软件入门与提高丛书

MATLAB 5.x 入门与提高

龚剑　朱亮　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

MATLAB 是集数学计算、图形处理和程序语言设计于一体的著名数学软件。本书循序渐进地介绍了 MATLAB 的主要函数命令，包括入门知识、数值计算功能、图形可视化处理功能、Notebook、程序设计、高级语言程序接口、图形用户界面以及 SIMULINK 动态仿真系统等高级功能。为使用户能融会贯通，本书最后给出了几个运用 MATLAB 解决实际问题的实例。

本书示例丰富，语言简洁，重点突出，可以作为高等院校理工科专业学生以及科研人员、工程技术人员学习 MATLAB 软件的参考书，尤其适用于对 MATLAB 有了一定初步了解，并希望进一步提高使用 MATLAB 技能的用户。

书　　名：MATLAB 5.x 入门与提高
作　　者：龚剑　朱亮
出 版 者：清华大学出版社（北京清华大学学研楼，邮编 100084）
　　http://www.tup.tsinghua.edu.cn
印 刷 者：北京市丰华印刷厂
发 行 者：新华书店总店北京科技发行所
开　　本：787×1092 1/16　印张：24.75　字数：584 千字
版　　次：2000 年 3 月第 1 版　2000 年 3 月第 1 次印刷
书　　号：ISBN 7-302-01208-3 /TP・452
印　　数：0001~5000
定　　价：33.00 元

《软件入门与提高丛书》特色提示

- ☑ 精选国内外著名软件公司的流行产品，以丰富的选题满足读者学用软件的广泛需求
- ☑ 以中文版软件为介绍的重中之重，为中国读者度身定制，从而便捷地掌握国际先进的软件技术
- ☑ 紧跟软件版本的更新，连续推出配套图书，使读者轻松自如地与世界软件潮流同步
- ☑ 明确定位面向初、中级读者，由“入门”起步，侧重“提高”，愿新手老手都能成为行家里手
- ☑ 围绕用户实际使用之需取材谋篇，着重技术精华的剖析和操作技巧的指点，使读者深入理解软件的奥秘，举一反三
- ☑ 追求明晰精练的风格，用醒目的步骤提示和生动的屏幕画面使读者如临操作现场，轻轻松松地把软件用起来

丛书编委会

《软件入门与提高丛书》序

使普通用户用电脑最关键也最头疼的恐怕就是学用软件了。软件范围之广，版本更新之快，功能选项之多，体系膨胀之大，往往令人目不暇接，无从下手；而每每看到专业人士在电脑前如鱼得水，把软件玩得活灵活现，一定又是惊羡不已。

“临渊羡鱼，不如退而结网”。道路只有一条：动手去用！选择您想用的软件和一本配套的好书，然后坐在电脑前面，开机、安装，按照书中的指示去用、去试，很快就会发现您的电脑也有灵气了，您也是个出色的舵手，能自如地在软件之海中航行了。

《软件入门与提高丛书》的推出就是为了给您一套畅游软件之海的导航器。它是一套包含了现今主要流行软件的易学易用的使用指导书。既可循序学习，亦可随查随用，使您学有所依，用有所循，快速便捷地掌握软件的操作方法和编程技术，得心应手地解决实际问题。

让我们来看一下本丛书的特色吧。

软件领域

本丛书所精选的软件皆为国内外著名软件公司的知名产品，也是时下国内应用面较广的软件，同时也是各领域令人注目的佼佼者。目前本丛书所涉及的软件领域主要有操作平台、办公软件、编程工具、数据库软件、网络和 Internet 软件、多媒体和图形图像软件等。本丛书还将密切注视新软件的面世，及时推出新软件以及虽然应用面稍窄但技术重要的软件产品的配套书。

版本选择

本丛书对于软件版本的选择原则是：紧跟软件更新步伐，以最近半年推出和未来半年即将推出的最新版本为重点，充分保证图书的技术先进性；兼顾经典主流软件，给广受青睐、深入人心的产品以一席之地；对于兼有中西文版本的软件，尽量取中文版而舍西文版，以全力满足中国用户的需要。

读者定位

本丛书明确定位于初、中级用户。不管您以前是否使用过本丛书所述的软件，这套书对您都非常合适。

本丛书名中“入门”的含义是指，对于每个软件的讲解都从必备的基础知识和基本操作讲起，新用户无需参照其他书即可轻松入门；老用户亦可从中快速了解新版本的新特色和新功能，自如地踏上新的台阶。至于书名中的“提高”，则蕴涵了图书内容的重点所在。以我们的经验，当前软件的功能日趋复杂，不学到一定的深度和广度是难以在实际工作中应付自如的。因此本丛书在让读者快速入门之后，就以大量明晰的操作步骤和典型的应用实例，教会读者更丰富全面地使用软件技术和应用技巧，使读者真正对所学软件融会贯

通、熟练在手。

内容设计

本丛书内容设计的策略是在仔细分析用户使用软件的困惑所在，并结合目前电脑图书市场现状的基础上确定的。简而言之，就是实用、明确和透彻。既不是面面俱到的“用户手册”，也并非详解原理的“功能指南”，而是独具实效的操作和编程指导书。一切围绕用户的实际使用需要选择内容，使读者在每个复杂的软件体系面前能“避虚就实”，直指目标；对于每个功能的讲解，则力求以明确的步骤指导和丰富的应用实例准确地指明如何去做，读者只要按书中的指示和方法做成、做会、做熟，再举一反三，就能扎扎实实地轻松过关。

风格特色

本丛书在风格上力求文字精练、图表丰富、脉络清晰、版式明快。另外，在策划写作时还特别设计了一些非常有特色的段落，以在正文之外为读者指点迷津。这些段落包括：


注　意——提醒可能出现的问题和容易犯的错误，以及如何避免，让您少一些傻眼的时刻和求救的烦恼。


提　示——提示可以进一步参见的章节，以及有关某个内容的详细信息，使您可深可浅，收放自如。


技　巧——指点一些捷径，透露一些高招，让您事半功倍，技高一筹。


试一试——精心设计各种操作练习，只要照猫画虎，试上一试，就不仅能在您的电脑上展现出书中出现的美妙画面，还能了解书中未详述的其他实现方法和可能出现的其他操作结果。随处可见的“试一试”，让您边学边用，时有所得，常有所悟。

经过紧张的策划、设计和创作，本套丛书已陆续面市，市场反应良好。许多书在两个月内迅速重印，本丛书自面世经来已累计售出两百多万册。在大量的读者反馈卡和来信中给我们提出了很多好的意见和建议，使我们受益匪浅。严谨、求实、高品味、高质量，一直是清华版图书的传统品质，也是我们在策划和创作中孜孜以求的目标。尽管倾心相注，精心而为，但错误和不足在所难免，恳请读者不吝赐教和指正，我们定会全力改进，在后续工作中提高。

本丛书在创作过程中得到了微软中国公司产品部的大力支持，对于他们在软件和技术资料的提供及有关目录的审定方面所给予的协助，表示衷心的感谢。

《软件入门与提高丛书》编委会

1999 年 4 月

目 录

引　言

MATLAB 原先是作为 Matrix 实验室使用 LINPACK 和 EISPACK 矩阵软件工具包的接口，后来才逐渐发展成为集通用科学计算、图形交互、系统控制和程序语言设计为一体的软件。

MATLAB 的基本单位是矩阵。它的表达式与数学、工程计算中常用的形式十分相似，极大地方便了用户学习和使用，故 MATLAB 深受用户欢迎。在欧美高等院校，MATLAB 已经成为线性代数、自动控制系统、数理统计、数字信号处理、时间序列分析和动态系统仿真等高级课程的基本教学工具；也是攻读学位的大学生、硕士生和博士生必须掌握的工具。在设计单位和科研部门，MATLAB 被广泛用来研究和解决各种具体的工程问题。

MATLAB 之所以受到广泛的欢迎，其开放性功不可没。除了内部函数之外，所有的 MATLAB 主包文件和各工具包文件都是可读可改的源文件(它们多以.m 为后缀)。用户可以研究源文件，掌握其用法，然后对其进行修改以适应自己的需要，还可加入自己编写的文件构成新的工具包。

近年来，随着 MATLAB 的升级和改进，国内的 MATLAB 用户在迅速增加。对于这些用户来说，手头有一本详尽介绍 MATLAB 中文版的书无疑是如虎添翼。基于此，我们编写了这本书。本书介绍了 MATLAB 的数值计算功能(第 2~5 章)、图形可视化处理功能(第 6~7 章)、程序设计功能(第 8~10 章)以及系统动态仿真(第 11 章)和工具箱(第 12 章)。同时在解释 MATLAB 各函数命令时结合了大量的例子和图片，而且根据作者实际使用 MATLAB 的经验给用户提出了有效的建议，最后还给出 2 个实例(第 13 章)。本书示例丰富、语言简洁、重点突出。

作为商业软件，MATLAB 的覆盖面极广，其内容之丰富远非一本书所能包括。因此本书重点介绍 MATLAB 比较基础和通用的部分。一方面避免面面俱到而失去重点，另一方面方便用户在读完本书后掌握 MATLAB 的精髓，并根据自己的需要自学 MATLAB 中专业性较强的部分。

一方面，本书能引导初级用户迅速掌握 MATLAB，达到事半功倍的效果；另方面，本书还能为中、高级用户、提供一些行之有效的使用 MATLAB 的经验和心得。

第 1 章
MATLAB 简介

本章要点：

本章将全面介绍 MATLAB 的各个方面，使用户对 MATLAB 有一个基本的认识，为后面的介绍打下基础。

本章具体包括以下内容：

- MATLAB 发展史
- MATLAB 的安装
- MATLAB 快速入门
- MATLAB 的联机帮助
- MATLAB 中的环境变量
- MATLAB 5.3 的新特性

作为一名工程技术人员，每天会与大量的科学计算打交道。为了更快更好地完成任务，人们开发出了众多数学工具软件(数学工具软件在这里是作者用来区别文字处理、绘图、通信类软件的，随着软件工程的发展，软件功能的不断增强，也许一个软件就包含了以上各种功能)。这些数学处理软件的原始内核大致可以分为三类：

(1) 以数值计算为主(Number Crunching 型)的软件，如 MATLAB、Xmath、Gauss 和 MLAB 等。这类软件对大批数据有较强的管理、计算和可视化的能力，运行效率高。尤其是 MATLAB，市场占有率很高，在一些高校，MATLAB 甚至作为理工科学生的必修课程，其重要性可见一斑。

(2) 以数学分析为主(Math Analysis 型)的软件，如 Mathematica、Maple 和 Macsyma。这类软件以符号计算为主，可以得到解析解和任意精度解，但是在处理大批量数据时运行效率较低。

(3) 一些辅助性的软件，如 Visio 等。

在国际流行的科学技术应用软件中，MATLAB 独具特色。该软件的开发商 MathWorks 公司顺应时代潮流，将 MATLAB 不断完善，加入各种功能。该公司使 MATLAB 不但具有卓越的数值计算功能和强大的图形处理能力，而且还具有在专业水平上开发符号计算、文字处理、可视化建模仿真和实时控制能力，使 MATLAB 成为适合多学科、多部门要求的新一代科技应用软件。

1.1 MATLAB 发展史

MATLAB 原先是作为 Matrix 实验室使用 LINPACK 和 EISPACK 矩阵软件工具包的接口，后来才逐渐发展为集通用科学计算、图形交互、系统控制和程序语言设计于一体的软件。

MATLAB 的基本单位是矩阵。它的表达式与数学、工程计算中常用的形式十分相似。例如，在 MATLAB 中，若 A 为矩阵，b，x 为向量，构造矩阵方程 b=Ax 被写为 b=A*x。这极大地方便了用户的学习。如果给出这个方程，需要求解，只需用 x=A\b 或是 x=b/A 即可(这是两种不同的除法，将在以后的章节中予以介绍)，完全不需对矩阵的求逆进行编程，比 C 和 FORTRAN 语言要简捷易用。

现在，MATLAB 已经成为一个系列。它包括 MATLAB 主程序和各种可选的工具包(Toolbox)。主程序中包含数百个核心内部函数。到目前为止，这数十个工具包又可以分为两类：功能性工具包和学科性工具包。功能性工具包主要用于扩展 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能和与硬件的实时交互过程。无论哪个学科都会用到这样的功能，因而是用好 MATLAB 的基础，本书中将详细介绍。学科性工具包专业性比较强，例如，控制工具包(Control Toolbox)、信号处理工具包(Signal Processing Toolbox)、通信工具包(Communication Toolbox)和专有图形处理工具包(Specgraph Toolbox)。另外还有一些工具包是 MATLAB 与系统的接口，例如用户图形界面开发工具(Uitools Toolbox)、字符串处理(Strfun Toolbox)与文件输入输出(Iofun Toolbox)，可以归入第一类中。

1.2 MATLAB 的安装

1.2.1 MATLAB 对系统的要求

MATLAB 从 3.0 版开始(1987 年),历经了 12 年的改进,目前的最高版本为 5.3 版(1999 年 7 月)。资料表明,它适用于以下平台(但不一定是 5.3 版本)。

1. Microsoft Windows

在 Windows 95、Windows 98 或 Windows NT(安装 Service Pack 3 或更高)上运行 MATLAB 的基本要求如下:

- 基于 Intel 486、Pentium、Pentium Pro 或 Pentium II 的个人计算机(PC 机)。测试表明,对于 Intel 系列的兼容芯片,如 AMD K5/K6 或 Cyrix 6x86/MII,MATLAB 也可以正常运行。

CD-ROM (安装用)。

- 至少 16 MB 内存,推荐 24 MB 以上。
- 所需的硬盘空间与硬盘簇(cluster)的大小有关。当簇为 512 字节时,需要 30 MB(安装 MATLAB 核心系统)+70 MB(安装联机帮助文件);当簇为 64KB 时,需要 145 MB(安装 MATLAB 核心系统)+250MB(安装联机帮助文件)。为了安装各种工具包,以及考虑到为硬盘留出足够的空间作为虚拟内存,建议安装前能有 1GB 左右的剩余空间。
- 8 位以上的显卡,以及 Windows 支持的图形加速卡、打印机和声卡。

为了运行 MATLAB HELP DESK,需要安装 Netscape Navigator 3.0 或其更高版本或 Microsoft Internet Explorer 4.0。

为了查看、打印随机的 PDF 格式的文档,需要安装 Adobe Acrobat Reader(在 MATLAB 光盘中提供)。

为了运行 MATLAB NOTEBOOK,需要安装 Microsoft Word 7.0 (Office 95)或 8.0 (Office 97),但是 NOTEBOOK 与中文 Word 存在一些兼容问题。

为了编译自己的 MEX 文件,需要以下软件之一:

DEC Visual Fortran 5.0

Microsoft Visual C/C++ 4.2 或 5.0

Borland C/C++ 5.0 或 5.02

WATCOM 10.6 或 11

注意: MATLAB 可以通过 TCP/IP 协议从网络进行安装。

2. UNIX

运行 MATLAB 需要满足如下基本要求:

- 40 MB 空余磁盘空间(如果要安装所有的帮助文件，需要 100 MB)。
- 64 MB 内存，推荐更多。
- 64 MB 交换分区(必需)。
- CD-ROM (安装用)。

为了运行 MATLAB HELP DESK，需要安装 Netscape Navigator 3.0 或其更高版本或 Microsoft Internet Explorer 4.0。

为了查看、打印随机的 PDF 格式的文档，需要安装 Adobe Acrobat Reader(在 MATLAB 光盘中提供)。

表 1.1 所示为能够运行 MATLAB 的 UNIX 系统配置。

表 1.1 UNIX上的MATLAB

品　牌	硬　件	操作系统	版　本
Sun	SPARC	Solaris	2.5.1， 2.6， 2.7
HP	HP9000 PA.RISC	HP.UX	10.20， 11.0
SGI	R4000/R5000	IRIX	6.3， 6.5
SGI	R8000/R10000	IRIX64	6.4， 6.5
IBM	RS/6000	AIX	4.2， 4.3
DEC	Alpha	Digital Unix	4.0d
RedHat & Debian Distributions	Intel 486， Pentium， Pentium Pro 或 Pentium II	Linux	2.0.34 kernel RedHat 4.2， 5.1 Debian 2.0

3. Macintosh

在 Macintosh 机上，目前的最高版本为 5.2。系统要求如下：

- CPU Power Macintosh 或 Macintosh 68000 系列(安装以下芯片：68020/68030 处理器和 68881/68882 数学协处理器、68040 处理器，但在 68LC040 上无法运行 MATLAB)。
- 操作系统　System 7.6，7.6.1，8.0 或 8.1。
- CD-ROM 驱动器(安装用)。
- 16 MB 内存。
- 所需的硬盘空间与安装时的选择有关，系统会在安装时给出提示。
- 显示适配器为 8 位或更高(推荐 16 位或 24 位彩色)。
- 其余硬件　Apple LaserWriter 或其他的 PostScript 打印机；附加的 RAM(内存)。

为了运行 MATLAB HELP DESK，需要安装 Netscape Navigator 3.0 或其更高版本或 Microsoft Internet Explorer 4.0。

为了查看、打印随机的 PDF 格式的文档，需要安装 Adobe Acrobat Reader(在 MATLAB 光盘中提供)。

为了运行 MATLAB NOTEBOOK，需要安装 Microsoft Word 6.0。

为了编译自己的 MEX 文件，需要以下软件之一：

对于 Power Macintosh 系列：

Macintosh Programmer's Workshop (MPW) MrC compiler version 2.0 from E.T.O. #21

Metrowerks CodeWarrior 10 or 11 (CW10 or CW11)， or CodeWarrior Pro 1

对于 Macintosh 68000 系列：

Fortner LS Fortran compiler version 3.3

Metrowerks Code Warrior 10 or 11 (CW10 or CW11)

Fortner LS Fortran for the Power Macintosh version 1.1

Macintosh Programmers Workshop (MPW) SC version 8.1.2 from E.T.O. #21

1.2.2　开始安装

下面以一台典型的 PC 机(操作系统为 Windows 98 中文版)为例，介绍 MATLAB 的安装过程。

(1) 放入 CD，Windows 会自动运行安装程序。如果不行，可以单击左下角的【开始】按钮，选择【运行】命令，输入 X:\setup 后按 Enter 键(其中 X 为光驱的盘符)。初始化完成后，出现如图 1.1 所示界面。

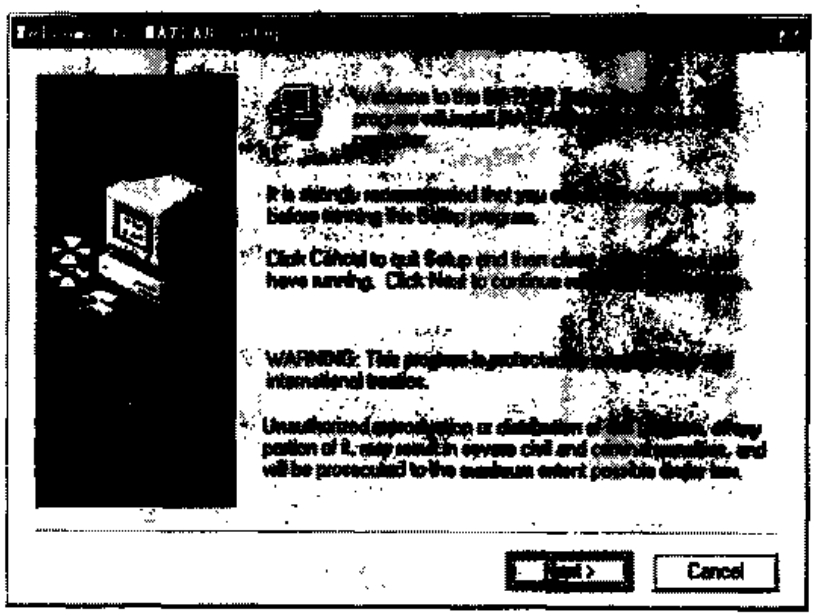


图 1.1　MATLAB的安装(1)

(2) 单击 Next 按钮，出现一个对话框，如图 1.2 所示。输入用户的姓名、单位及序列号。

注意： 序列号应该随着购买的 MATLAB 一起到用户的手中，如果没有，请与软件零售商联系。

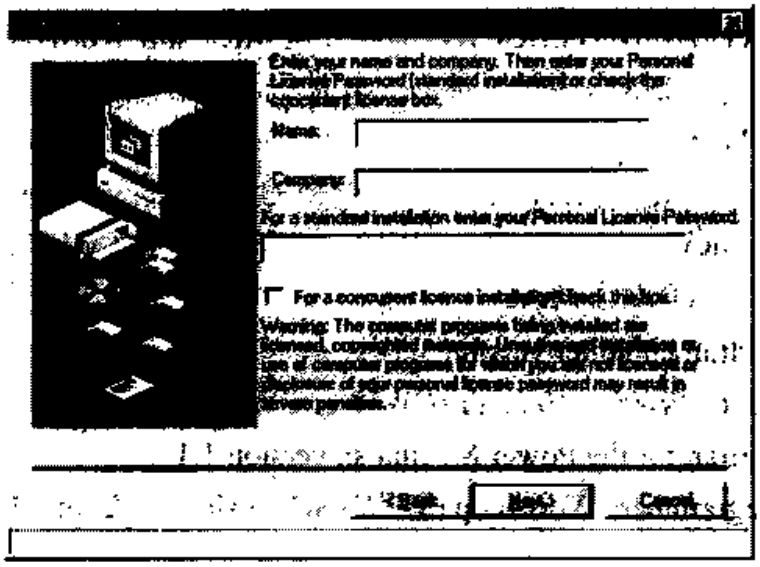


图 1.2 MATLAB的安装(2)

(3) 选择要安装的各种部件。单击各个部件前的复选框，可以选择是否安装这个部件，注意其中的 MATLAB WEBSEVER 只能在 Windows NT 下运行。如前所述，如果安装 HTML 格式的帮助文件，需要 Netscape Navigator 3.0 或其更高版本或 Microsoft Internet Explorer 4.0；而 PDF 格式的帮助文件，则要 Adobe Acrobat Reader。最下面两行显示的是以目前的配置安装所需要的磁盘空间和磁盘的剩余空间，如图 1.3 所示。

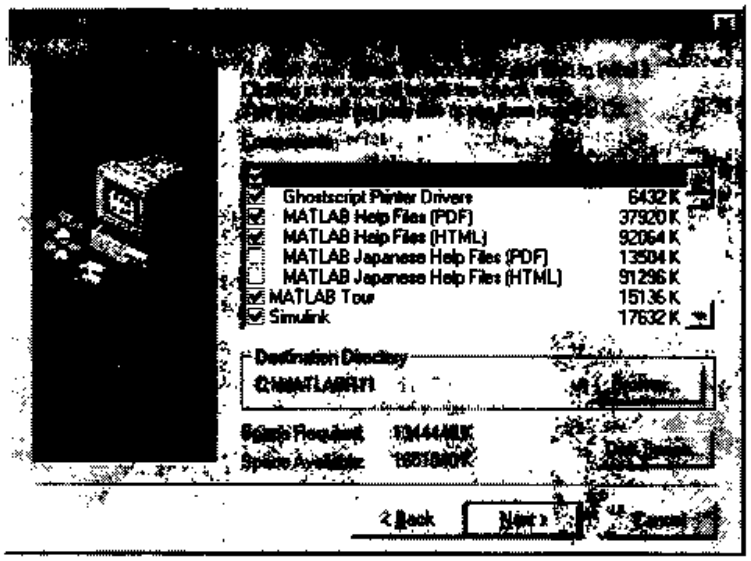


图 1.3 MATLAB的安装(3)

(4) 开始安装，系统显示进度。中间会有一个对话框，询问用户是否要安装 Microsoft Visual Java Machine(Microsoft Java 虚拟机)。建议安装。

(5) 安装完毕，重新启动机器，完成安装。

注意： 在 5.3 版的 MATLAB 中，NOTEBOOK 为默认安装项，默认安装至 C:\MATLABR11\notebook\pc\M.BOOK95.DOT 中。

1.3 MATLAB 快速入门

从本节开始，将进行对 MATLAB 的具体学习。MATLAB 是一个很复杂的程序，功能强大。要用好 MATLAB，需要有一定的数学基础，还要熟悉 Windows 的基本操作。

随着时代的发展，MATLAB 也在不断进步。在新的 5.3 版本中，加入了许多令人激动的新特点，并且解决了和中文 Windows 98 兼容的问题(对于 5.2 版，需要一个补丁程序，用户可在http://www.mathworks.com站点下载)。

1.3.1 MATLAB 的启动

1. 以快捷方式启动

具体步骤如下：

(1) 启动 Windows。

(2) 安装完 MATLAB 后，会在桌面上出现一个名为 MATLAB 5.3 的快捷方式(shortcut)。双击图标，启动 MATLAB。

启动完成后，出现 MATLAB 的工作窗口(MATLAB Command Window)，如图 1.4 所示。

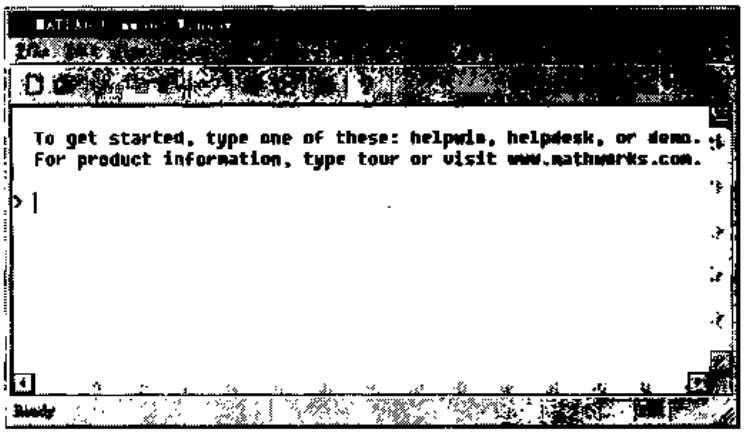


图 1.4 MATLAB的工作窗口

2. 以菜单方式启动

如果删除了桌面的快捷方式，也可以用菜单启动，具体步骤如下：

(1) 启动 Windows。

(2) 单击桌面左下角的【开始】按钮。

(3) 通过鼠标或键盘操作，依次选择【程序】、MATLAB 和 MATLAB 5.3 命令，进入 MATLAB 如图 1.4 所示的工作窗口。

说明：

- MATLAB 工作窗口的最上面两行是系统初始提示信息。
- 如果 MATLAB 运行在英文的 Windows 平台上，那么 MATLAB 工作窗口的第三行就会出现 MATLAB 环境提示符号“》”和光标位置符。
- 如果 MATLAB 运行在中文的 Windows 平台上，那么 MATLAB 工作窗口的第三行就会只有光标位置符，而没有 MATLAB 环境提示符号“》”。

1.3.2 MATLAB 工作窗口和指令行的操作

MATLAB 是一个标准的 Windows 界面，可以利用菜单中的命令完成对工作窗口的操作。它的使用方法与 Windows 的一般应用程序相同，参看图 1.5。下面将对菜单进行介绍。同时，考虑用户的实际情况，本节对菜单的介绍只到第二级，某些复杂操作将在后面的高级内容中介绍。

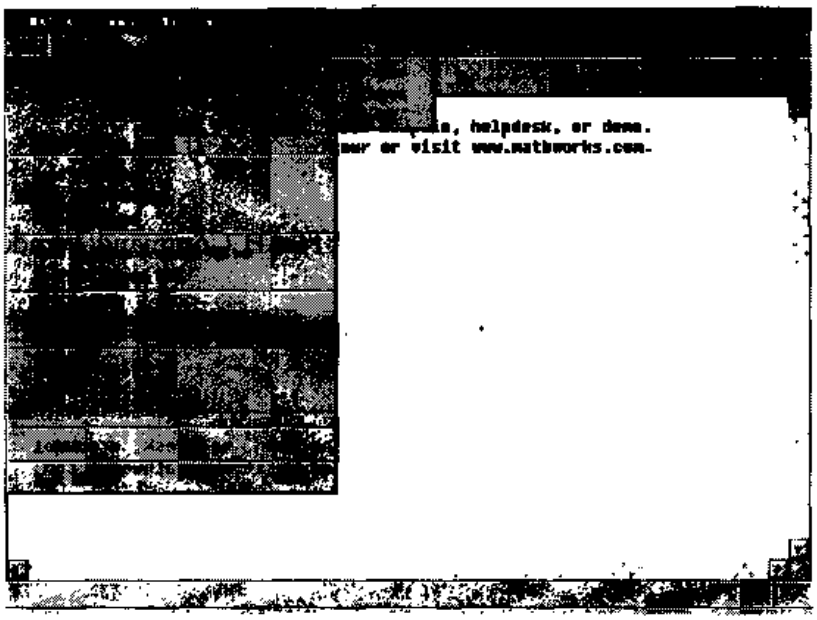


图 1.5 菜单的操作

1. **菜单命令**

- File(文件)菜单的内容如表 1.2 所示。

表 1.2 File菜单的使用

菜单命令	功 能
New	新建一个 M 文件、图形或 Simulink 模块
Open	打开一个已有的文件，可以是 M 文件、图形或 Simulink 模块
Open Selection	打开指定的文件
Run Script	运行一个已有的 M 文件

续表1.2

菜单命令	功 能
Load Workspace	将文件中内容放入 MATLAB 的工作区中
Save Workspace As	将 MATLAB 工作区中的内容存入文件
Show Workspace	显示 MATLAB 的工作区
Show Graphics Property Editor	显示图形属性编辑器
Show GUI Layout Tool	显示 GUI 界面布局管理器
Set Path	设置工作路径
Preferences	定义工作环境
Print Setup	打印设置
Print	打印
Print Selection	打印指定的文件
Exit MATLAB	退出 MATLAB

- Edit(编辑)菜单的内容如表 1.3 所示。

表 1.3 Edit菜单的使用

菜单命令	功 能
Undo	撤消上一步的操作
Cut	将选中的内容删除，放入剪贴板
Copy	将选中的内容放入剪贴板，但不删除所选的内容
Paste	将剪贴板的内容放入 MATLAB 工作窗口
Clear	清除
Select All	全部选中
Clear Session	清除屏幕内容，但是保留变量

- View(视图)菜单的内容如下所示：
  Toolbar 是否显示工具条
- Window(窗口)菜单

这个菜单应用于打开了多个 MATLAB 窗口的情况，这时可以用它在各个窗口之间切换。

- Help(帮助)菜单的内容如下所示：
  Help Windows 显示帮助窗口
  Help Tips 关于帮助的技巧

Help Desk (HTML) 显示 HTML 格式的帮助文件
Examples and Demos 实例
About MATLAB... 显示版权信息
Show License 显示用户授权协议
Join MATLAB Access 加入 MATLAB 用户协会

2. 工具条简介

参见图 1.4，工具条位于菜单的下方，它的图标所代表的常用功能如表 1.4 所示。

表 1.4 工具条简介

图 标	功 能
	新建一个 M 文件
	打开一个已有的文件，可以是 M 文件
	将选中的内容删除，放入剪贴板
	将选中的内容放入剪贴板，不删除所选的内容
	将剪贴板的内容放入 MATLAB 工作窗口
	撤消上一步的操作
	工作区管理器
	路径管理器
	SIMULINK 类管理器
	显示帮助窗口

3. 若干通用操作指令

除了通过菜单命令对工作窗口进行控制以外，MATLAB 还提供了许多通过键盘输入的控制指令。表 1.5 所示为 MATLAB 工作窗口中的一些通用操作指令。

表 1.5 MATLAB工作窗口中的部分通用指令

指令名称	指令功能
cd	改变当前工作目录
clear	清除内存中的所有变量和函数
clc	擦除 MATLAB 工作窗口中所有显示的内容
clf	擦除 MATLAB 当前窗口中的图形
dir	列出指定目录下的文件和子目录清单
disp	在运行中显示变量或文字内容
echo	控制运行文字指令是否显示

续表1.5

指令名称	指令功能
hold	控制当前图形窗口对象是否被刷新
pack	收集内存碎块以扩大内存空间
quit	关闭并退出 MATLAB
type	显示所指定文件的全部内容

4. 指令行的编辑

启动 MATLAB 以后，就可以利用它进行工作。由于 MATLAB 是一种交互式的语言，因此随时输入指令后即时给出运算结果成为它的主要工作方式之一。

表 1.6 列出了控制光标位置及对指令进行操作的一些常用操作键。

表 1.6　常用操作键

键盘操作		作　用
↑	Ctrl+p	调出前一个命令行
↓	Ctrl+n	调出后一个命令行
←	Ctrl+b	光标左移一个字符
→	Ctrl+f	光标右移一个字符
Ctrl+→	Ctrl+r	光标右移一个单词
Ctrl+←	Ctrl+l	光标左移一个单词
Home	Ctrl+a	光标移至行首
End	Ctrl+e	光标移至行尾
Esc	Ctrl+u	清除当前行
Del	Ctrl+d	清除光标所在位置后的字符
Backspace	Ctrl+h	清除光标所在位置前的字符
	Ctrl+k	删至行尾

【例 1】绘制 MATLAB 的图标。

通过这个例子，说明 MATLAB 指令输入的基本方法，同时演示 MATLAB 的强大功能。

在 MATLAB 命令窗口中输入以下内容。注意，每一行输入结束后，必须按 Enter 键。也可以在别的字处理软件(如 Notepad)中写好程序段，通过剪贴板将它们放入 MATLAB 的命令窗口求解。

```
load logo
surf(L, R), colormap(M), n=length(L(:, 1));
```

```
axis off, axis([1 n 1 n ..2 .8]), view(.37.5, 30)
title('MATLAB 5.x 的图标')
```

结果如图 1.6 所示。

图 1.6 MATLAB的图标

【例 2】计算

$$\frac{2\cos(0.3*\pi)}{1+\sqrt{7}} 和 \frac{2\cos(0.4*\pi)}{1+\sqrt{7}}$$

(1) 在 MATLAB 命令行中输入：

```
2*cos(0.3*pi)/(1+sqrt(7))
```

(2) 计算出第一个表达式的结果后，再计算第二个表达式的结果。

- 按↑键(或用 Ctrl+p)，调出上次的输入。
- 用←或→键移动光标，将 3 改为 4。
- 按 Enter 键，MATLAB 给出计算结果。

1.4 MATLAB 的联机帮助

1.4.1 基本帮助指令

在启动 MATLAB 时，可以看到第一行的提示信息中给出了 3 个指令：helpwin、helpdesk 和 demo。它们对于用户迅速掌握 MATLAB 帮助极大。下面分别介绍。

1. helpwin指令

在工作窗口中输入 helpwin，显示如图 1.7 所示的帮助窗口。

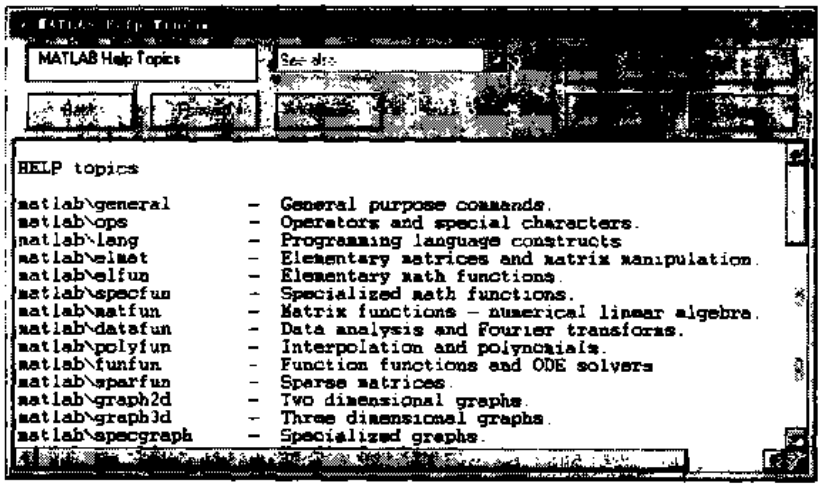


图 1.7 帮助窗口

- 在左上角的文本框中，输入想要查询的命令名，按 Enter 键，显示帮助内容。
- 右边的列表框显示相关主题。
- 单击 Back 按钮显示上一条帮助内容。
- 单击 Forward 按钮显示下一条帮助内容。
- 单击 Home 按钮回到开始。
- 单击 Go to Help Desk 按钮显示 HTML 格式的帮助内容。
- 单击 Tips 按钮显示相关技巧。
- 单击 Close 按钮关闭此窗口。

提示： *双击帮助内容可以显示相应的内容。*

2. helpdesk指令

在工作窗口中输入 helpdesk，显示 HTML 格式的帮助内容。如果用户有网上冲浪的经验，将会发现使用这个窗口与 www 的使用完全相同。在此不再赘述。

3. demo指令

在工作窗口中输入 demo，显示一个名为 MATLAB Demo Window 的窗口，如图 1.8 所示。从左边列表框中选择感兴趣的内容，右上的列表框内显示关于此内容的简介，右下方的列表框列出具体内容。双击相关的主题(或选中后单击 Run 按钮)，开始演示。

【例 3】MATLAB 的 Demo 功能使用示例。

(1) 在左边列表中选择 MATLAB 下的 Languang/Graphics。

(2) 在右边列表中双击 3.D surface plots。

(3) 显示一个名为 3.D Plots in Handle Graphics 的窗口。

(4) 在右边可以进行各种设置，从而改变显示的图形。

(5) 左下的窗口显示源码。

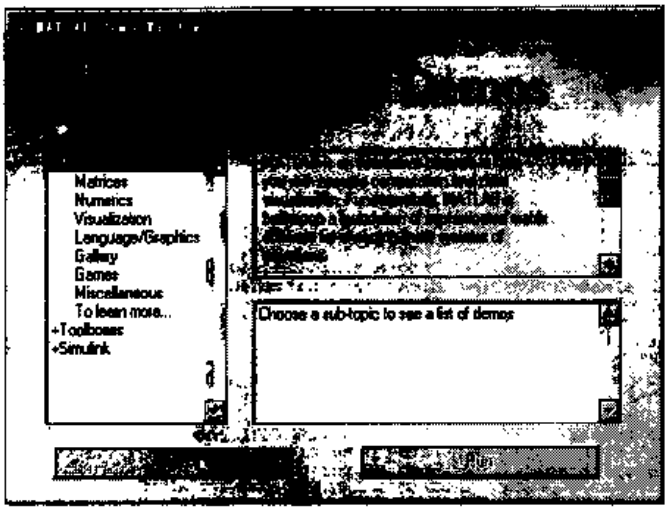


图 1.8 MATLAB的Demo窗口

1.4.2 MATLAB 的联机查询

理解、掌握和运用 MATLAB 的联机查询功能，对于新老用户都是十分重要的。MATLAB 的联机帮助系统相当完备。就查询系统的调用方式而言，可以分为两种：

- 在 MATLAB 指令窗口内输入 helpwin。这种方法在前面已有介绍，而且其使用方法与 Windows 的普通帮助相似，这里就不再介绍了。
- 在 MATLAB 的指令窗口内，直接输入帮助命令求助。这种方法最常用，下面详细介绍。

1. help 指令

help 是最常用的帮助指令。它可以提供绝大部分 MATLAB 指令的使用方法的联机说明。下面举例介绍 help 的应用方法。

【例 4】MATLAB 中 Help 指令的使用。

(1) 在 MATLAB 指令窗口内输入 help，得到联机帮助的总览。

(2) 输入 help elfun，寻求关于基本函数的帮助信息。

(3) 输入 help exp，得到指数函数指令 exp 的详细信息，如下所示：

```
EXP    Exponential.
EXP(X) is the exponential of the elements of X,  e to the X.
For complex Z=X+i*Y,  EXP(Z) = EXP(X)*(COS(Y)+i*SIN(Y)).
See also LOG,  LOG10,  EXPM,  EXPINT.
 Overloaded methods
help sym/exp.m
help demtseries/exp.m
```

说明：

- 如果想得到更详细的专题帮助，可在 help 指令后面再输入欲寻求的专题名字，例如 help simulink 。

- 上例中的 See also LOG， LOG10， EXPM， EXPINT，给出了与 exp 相关的指令。
- help 的工作机理是把指定名字的 M 文件的第一段注释内容显示出来。用户可以采用这种注释结构，以构成自己文件的联机帮助。

2. lookfor 指令

当要查找具有某种功能但又不知道准确名字的指令时，help 的能力就不够了。为此，MATLAB 设计了一个 lookfor 指令。它可以根据用户提供的完整或不完整的关键词，去搜索出一组与之相关的指令。

【例 5】lookfor 指令使用示例。

(1) 查找有关积分的指令 lookfor integral。

(2) 查找能进行傅立叶变换的有关指令 lookfor fourier。

说明：

- 由以上两个举例可以看出，lookfor 与 help 相配合就形成了相当完备的联机帮助系统。
- lookfor 的机制是对 MATLAB 目录中的每个 M 文件注释区的第一行进行扫描，一旦发现这行中包含欲查询的字符串，那么该文件名以及注释的第一行将被显示。当用户想建立自己文件的联机帮助时，可以利用这种机制。

3. 其他帮助指令

MATLAB 还提供了其他一些帮助指令，如表 1.7 所示。这些指令暂时只给出名字，在后面的章节中将会详细介绍。

表 1.7 其他帮助指令

指令名称	指令功能
exist	检查指定名字的变量或函数文件的存在性
what	按扩展名分类列出(在搜索路径中)指定目录上的文件名
which	列出指定名字文件所在的目录
who	列出工作内存中的变量
whos	列出工作内存中的变量名及其细节

1.5 MATLAB 中环境变量的设置

本节内容便于用户进一步理解和掌握 MATLAB，初学者可以跳过本节，使用 MATLAB 的默认设置足以完成大部分工作。

1. 环境变量的作用

环境变量是与 MATLAB 工作相关的一些变量，它们的设置影响着 MATLAB 的正常工作。以路径变量为例说明它们的作用。

在 MATLAB 中，如果输入一个命令，例如 logo，那么 MATLAB 将按照以下步骤进行搜索。

(1) 在工作内存中搜索，看它是否是变量。

(2) 检查它是否是内部函数。

(3) 在当前目录下，检查是否有 logo.m 文件。

(4) 沿着 matlabrc.m 文件指定的路径，再逐个目录查找是否有 logo.m 文件。

2. 环境变量的初始化

MATLAB 的环境变量在 matlabrc.m 文件中定义。matlabrc.m 是在 MATLAB 启动后自动执行的第一个 M 文件。它定义了 MATLAB 环境下的路径结构、MATLAB 的图形大小、图元默认值和 MATLAB 工作窗口的初始提示信息等重要参数。该文件定义的搜索路径部分如下：

```
p = ['$toolbox/matlab/general:', …
    '$toolbox/matlab/ops:', …
    '$toolbox/matlab/lang:', …
    '$toolbox/matlab/elmat:', …
    '$toolbox/matlab/elfun:', …
    '$toolbox/matlab/specfun:', …
    '$toolbox/matlab/matfun:', …
    '$toolbox/matlab/datafun:', …
    '$toolbox/matlab/polyfun:', …
    '$toolbox/matlab/funfun:', …
    '$toolbox/matlab/sparfun:', …
    '$toolbox/matlab/graph2d:', …
    '$toolbox/matlab/graph3d:', …
    '$toolbox/matlab/specgraph:', …
    '$toolbox/matlab/graphics:', …
    '$toolbox/matlab/uitools:', …
    '$toolbox/matlab/strfun:', …
    '$toolbox/matlab/iofun:', …
    '$toolbox/matlab/timefun:', …
    '$toolbox/matlab/datatypes:', …
    '$toolbox/matlab/winfun:', …
    '$toolbox/matlab/demos:', …
    '$toolbox/simulink/simulink:', …
```

```
'$toolbox/simulink/blocks:', ...
'$toolbox/simulink/simdemos:', ...
'$toolbox/simulink/dee:', ...
'$work:', ...
'$toolbox/local:', ...
];
```

注意：对应于不同的安装选择，搜索路径可能会有所不同。上例中给出的是一个基本系统的情况(包括核心函数和 SIMULINK)。

说明：

- 事实上，matlabrc.m 文件是通过对同一目录下的 pathdef.m 文件的调用来实现搜索路径设置的；
- MATLAB 安装时，路径自动生成，不需用户编写。因此，请勿随便改动此文件，以免 MATLAB 工作出错。

3. 环境变量的设置

环境变量仍然是 MATLAB 的变量，也可以在运行中对它们进行赋值。但是，必须使用 MATLAB 提供的某些专用工具来完成对它们的修改。

仍以搜索路径变量 path 为例，说明这一点。

在 MATLAB 中，可以动态增减搜索路径，方法有两种：

(1) 通过命令 path 来实现。path 命令的语法为 path(P1，P2)，其中 P1、P2 是两个字符串。功能是将 P2 连接到 P1 后，构成新的搜索路径。

说明：

- 输入 cd 'C:\MATLABR11'，将当前目录改至 C:\MATLABR11(MATLAB 5.3 的默认安装目录)。
- 输入 path(path，cd)，将当前目录加入搜索路径。
- 输入 mkdir MyWorks，在当前目录下建立一个名为 MyWorks 的目录。
- 输入 path(path，'C:\MATLABR11\MyWorks')，将此目录加入搜索路径中。

(2) 通过 pathtool 实现。在命令行中输入 pathtool，启动搜索路径管理器；也可以单击工具条上的图标，两者是等效的。这时，出现图 1.9 所示的窗口。

这是一个 GUI 界面的工具，从 Path 菜单下选择 Add path 命令，出现一个对话框，如图 1.10 所示。

单击按钮，选择一个目录。下面的两个单选按钮决定所加入目录在搜索路径中的位置(最前或最后)。单击 OK 按钮，完成操作。

在主窗口中单击 Browse 按钮，可以选择当前目录(相当于 cd 命令)，这时右边列出此目录下的相应文件。

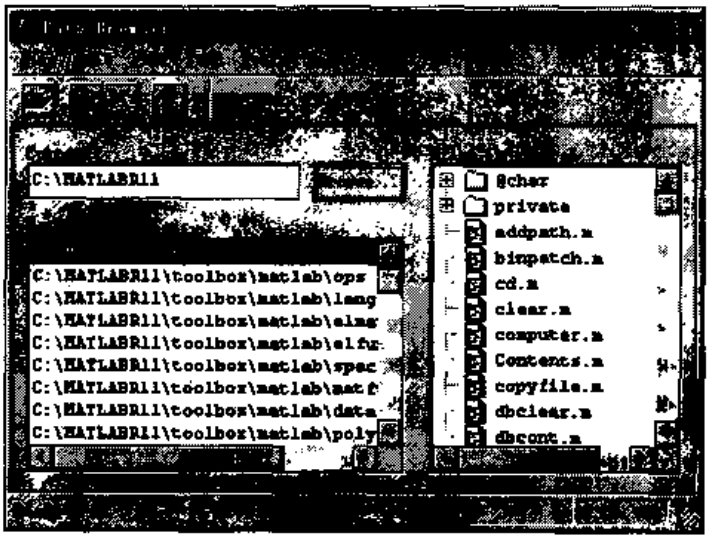


图 1.9 搜索路径管理器

图 1.10 搜索路径的添加

该管理器的其他功能，由于与本节关系不大，故不予介绍。请有兴趣的用户参看MATLAB 的帮助文档。

提示： 常用的目录可以放在搜索路径的前面，这样可以提高查询执行的速度。

1.6 MATLAB 5.3 的新特性

相对于以前的版本，MATLAB 5.3 提供了许多新特性。这里仅介绍一些常用的新特性，包括：对整数数据的支持；字符至双精度数值的转换；稀疏矩阵操作；改进的图形窗口和改进的应用程序接口(API)。

1. 对整数数据的支持

在 5.3 版中，扩展了对整型数据类型的支持，增加了对应于现有的 8 位、16 位和 32 位整数(包括有符号和无符号数)的数组类。

这些类主要是用来存储整数值的。大多数不改变数组元素的操作的返回值被定义为这些数据类型(例如reshape、size、逻辑运算及其相关操作)。另外，MATLAB支持针对整数类型的find函数，但是返回的数组类型为double型。

在 5.3 版中，sum 函数支持所有的整数类型。而以前，sum 函数仅仅支持无符号的 8 位整数(uint8 类)，但是现在，sum 支持 int8、int16、int32 以及 uint16、uint32 数据类。当

sum 函数用于整数类时，返回值为 double 类。

2. 字符至双精度数值的转换

新的 str2double 函数，能够将一个字符串转换为一个双精度数值。原始字符串包含一个由 ASCII 字符组成的数值(实数或复数)。Mathworks 公司推荐本函数，用以取代原来的 str2num 函数。

新的 texlabel 函数，能够从一个字符串得到一个 Tex 格式的输出。

3. 稀疏矩阵操作

在 5.3 版中，对操作稀疏矩阵的迭代方法进行了改进，使它可以使用函数作为一个参数。例如，可以用下式

```
x = pcg(A, b)
```

解线性方程组Ax = b中的未知向量x。当A并不明显确定为一个矩阵时，可以将A表达为一个操作符，从而可以接受向量x的输入，返回矩阵A和向量Ax的乘积。这个操作符可以是M文件、字符表达式甚至一个内建的对象。下面是一个简单的例子。

```
function y = afun(x)
% AFUN(X) 返回A*X， 这里， A = diag(7*ones(n，1))，是一个对角阵。
y = 7*x;                    % y = A*x
```

下面这个例子，调用pcg函数解方程组。在原本用矩阵做参数处采用了afun函数。

```
A = diag(7*ones(n, 1)).
x = pcg('afun', b);
```

4. 改进的图形窗口

在 MATLAB 5.3 版中，图形窗口有了显著的改进。

File 菜单中增加了两个命令，实现以下功能：

- 将一个图形以标准的图形格式保存下来(例如 TIFF 格式)。
- 页面打印设置。

图形窗口另外的改进包括：

- 增加了一个 Tools 菜单。
- 增加了一个新的工具条。
- 增加了一个新的略图编辑器。
- 框架打印编辑器(PrintFrame Editor)的界面改进。

5. 改进的应用程序接口 (API)

- 对 ActiveX 支持的改进

支持交互式的运用 get 以得到属性列表，运用 send 以得到一个界面的所有事件列表；改进的数据类型转换；改进的事件/回调(event/callback)机制。

- MATLAB 5.3 版的 API 引擎(API Engine)支持 5.0 中的所有数据类型，包括单元数组(cell arrays)、多维数组(multidimensional arrays)和结构体(structures)。

第 2 章
MATLAB 的数值计算功能

本章要点：

本章将介绍一些与 MATLAB 数值计算有关的内容。MATLAB 数值计算是使用 MATLAB 的基础，是 MATLAB 强大计算功能的体现。使用数值计算功能可以帮助用户解决一些学习和工作中常遇到的计算问题。

本章具体包括以下内容：

- 如何使用 MATLAB 的表达式和变量
- 如何使用 MATLAB 的基本运算函数
- 如何在 MATLAB 中创建、使用和保存矩阵
- 如何在 MATLAB 中创建、使用和保存数组
- 如何进行矩阵的运算
- 如何进行数组的运算
- 如何使用矩阵运算函数
- 如何使用数组运算函数

数学计算可以分为数值计算和符号计算。在 MATLAB 中，这两者的根本区别是：数值计算中不允许有未定义的变量，而符号计算允许。本章将专门介绍数值计算，符号计算在以后的章节中再介绍。

MATLAB 之所以成为世界上主导的计算软件，成为众多数学软件中的佼佼者，正是因为其强大的数值计算功能。自从 Mathworks 公司从 80 年代推出 MATLAB 以来，不断完善 MATLAB 的数值计算功能。现在的 MATLAB 5.3 版本，数值计算功能已经十分成熟了。

MATLAB 最初是为了矩阵的数值运算而推出的，MATLAB 是从 MATtrix 和 LABoratory 各取前 3 个字母组成的，意思是“矩阵实验室”。可见矩阵是 MATLAB 的精髓，掌握矩阵的各种数值运算和函数，是以后高级运用的基础。本章着重介绍 MATLAB 的矩阵数值运算功能，并以此为基础详细介绍 MATLAB 的各种数值计算功能。

2.1 MATLAB 的表达式与变量

表达式和变量是使用 MATLAB 的基础，通过本节用户可以学习如何定义和使用表达式、定义和使用变量、在 MATLAB 中如何查看已经保存在内存中的变量、复数变量的定义和使用以及在 MATLAB 中有关永久变量的规定。另外，还有与之相关的在屏幕中显示表达式和变量的格式设置。

2.1.1 MATLAB 的表达式

MATLAB 采用的是表达式语言，用户输入的语句由 MATLAB 系统解释运行。MATLAB 的语句是由表达式和变量组成的。MATLAB 语句有 2 种最常见的形式：

- 表达式
- 变量=表达式

表达式由运算符、函数、变量名和数字组成。它在 MATLAB 中占有很重要的地位，几乎所有的运算都必须借助表达式来进行。

在第一种形式中，表达式运算后产生的结果如果是矩阵或其他的数值类型，MATLAB 系统将会自动赋给名为 ans 的变量，并显示在屏幕上。ans 是一个默认的变量名，它会在以后的类似操作中被自动覆盖掉。所以，对于重要的结果一定要记录下来，也就是要使用第二种形式。

在第二种形式中，等号右边的表达式计算后产生的结果，MATLAB 系统会将其赋给等号左边的变量后放入内存中并显示在屏幕上。

提示： 在书写表达式时，运算符两侧允许有空格，以增加可读性。表达式的末尾可以加上“；”，也可以不加。有“；”时，MATLAB 系统不显示计算的结果，而是直接把数值赋给变量，如果没有用变量就无法看到结果了。没有“；”，MATLAB 系统将会在语句的下面显示运算的结果。

1. 数字的表达

MATLAB 的数值采用十进制表示，可以带小数点和负号；也可以使用科学计数法，用 e 表示位数。以下记述的数都是合法的：

```
4          -65234     0.00001
9.762684   1.3e-4     5.677e67
```

数值的相对精度为 eps=2^{-52}

数值的表示范围是 10^{-308}～10^{308}

2. MATLAB常用算符

+ 加法　　- 减法　　^ 幂

* 乘法　　/ 右除　　\ 左除

在矩阵运算中有左除和右除的区别；对于数字运算没有区别。

【例 1】表达式的创建

```
x=(3*8.4)/8.9;
x
x =
6.5000
```

2.1.2 MATLAB 的变量

和表达式紧密相关的是变量。除了 MATLAB 自定义的一些保留字以外，可以用一个字母打头，后面最多可接 19 个字母或数字定义一个变量。注意，在 MATLAB 中是区分大小写的。MATLAB 中不需要专门定义变量的类型，系统可以自动根据表达式的值或输入的值来确定变量的数据类型。因此，可以自由方便地使用变量。但是，如果使用和原来定义的变量一样的名字赋值，原变量将自动被覆盖，系统不会给出出错信息。使用变量时，要自觉地避免重复。2.1.3 小节，可以看到如何避免这一问题。

【例 2】表达式的计算结果

```
1999/19
ans =
 105.2105
```

没有用变量赋值，系统用默认变量 ans。

【例 3】表达式的赋值

```
W=1+2+3+4+5;
W
W =
    15
```

2.1.3 who、whos、永久变量和复数

1. who和whos

who 和 whos 都是用来列出 MATLAB 工作区中已驻留的变量清单。而 whos 还给出变量的维数和性质。利用这个命令，可以很快地知道已经使用过的变量，从而防止变量名的重复使用，造成数据的丢失。

【例 4】用 who 检查内存变量

```
who
Your variables are:
W        ans
```

这两个都是上面例子中产生的变量，注意 ans 是系统默认的变量名。

【例 5】 用 whos 检查内存变量的详细情况

```
whos
Name      Size          Bytes  Class

W         1x1               8  double array
ans       1x1               8  double array
Grand total is 2 elements using 16 bytes
```

2. 永久变量

上面提到定义变量时有些 MATLAB 的保留字不能用，其中有一部分就是 MATLAB 中的永久变量(有时也称为预定义的变量)。它们是在 MATLAB 启动时，系统自动定义的变量，驻留于内存中。它们不会被内存清除命令 clear 清除(永久变量的名称就源于此)。可以为这些永久变量赋值，可是所赋的值可以用 clear 命令清除，从而恢复系统预定义的值(预定义变量的名字就反映这样的意思)。用 who 是看不到永久变量的(除非使用的是 MATLAB 3.0)。表 2.1 列出了永久变量。

表 2.1 永 久 变 量

变量名称	变量含义
Realmin	最小的浮点数，2^{-1022}
Realmax	最大的浮点数，2^{1023}
Eps	容差变量，定义为 1.0 到最近浮点的距离。PC 机上等于 2^{-52}
Pi	圆周率的近似值 3.14159
Inf 或 inf	正无穷大，定义为(1/0)
NaN	非数(Not a Number)，产生于 0/0，∞/∞，0*∞等运算
i, j	虚数单位，定义为 i=$\sqrt{-1}$， j=$\sqrt{-1}$

【例 6】无穷大的使用

```
r=1/0
Warning: Divide by zero.
r =
   Inf
```

注意： 在 MATLAB 中这样的操作不会引起程序执行中断，只是在给出警告信息的同时，用一个特殊的符号 Inf 来表示。而且这个符号和其他的变量一样，可以在以后的运算中发挥作用。

【例 7】无穷大的作用

```
1/r
ans =
0
```

说明：

r(Inf)同样可以当作一个变量来使用，这个特点可以在编程中发挥巨大的作用。

3. 复数

上面指出 i 和 j 都是虚数的单位，使用它们可以定义复数。同样可以用复数来组成复数表达式、复数变量和复数矩阵等。

【例 8】复数表达和运算

```
z1=3.4+4.6*i, z2=6*exp(i*pi/6)
z=z1*z2
z1 =
   3.4000 + 4.6000i
z2 =
   5.1962 + 3.0000i
z =
   3.8669 +34.1023i
```

2.1.4　数据的输出格式

虽然在 MATLAB 系统中数据的储存和计算都是以双精度进行的，但是用户可以改变屏幕上显示的格式。控制数据显示格式的命令是 format，具体应用方法如下所示：

- format/format short　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　系统选择 5 位定点和 5 位浮点中更好的表示
- format long g　系统选择 15 位定点和 15 位浮点中更好的表示
- format rat　近似的有理数的表示

- format hex 十六进制的表示
- format +(plus) 表示大矩阵是分别用+、-和空格表示矩阵中的正数、负数和零
- format bank 用元、角、分(美制)定点表示
- format compact 变量之间没有空行
- format loose 变量之间有空行

用户可以自己试验各种格式，选择适合自己的格式。

另外，MATLAB 也提供在对话框中选择显示格式。单击 MATLAB 窗口的 File 菜单，再选择其中的 Preferences 命令，可以看到如图 2.1 所示的界面。

图 2.1 显示格式的设置

用户可以根据需要在左边的选项组中选择所需要的格式，其中 Rationa 是上面提到的 Rat 的全称，而 Compact 形式必须由命令来指定。

2.2 MATLAB 的基本计算功能

MATLAB 带有强大的函数库，一般的数学运算都能很容易地实现。表 2.2 列出了一些常用的函数。

表 2.2 MATLAB常用的基本数学函数

函数名称	函数功能
abs(x)	纯量的绝对值或向量的长度
angle(z)	复数 z 的相角(Phase angle)
sqrt(x)	开平方
real(z)	复数 z 的实部
imag(z)	复数 z 的虚部

续表2.2

函数名称	函数功能
conj(z)	复数 z 的共轭复数
round(x)	四舍五入至最近整数
fix(x)	无论正负，舍去小数至最近整数
floor(x)	地板函数，即舍去正小数至最近整数
ceil(x)	天花板函数，即加入正小数至最近整数
rat(x)	将实数 x 化为分数表示
rats(x)	将实数 x 化为多项分数展开
sign(x)	符号函数 (Signum function) 当 x<0 时，sign(x)=－1 当 x=0 时，sign(x)=0 当 x>0 时，sign(x)=1
rem(x，y)	求 x 除以 y 的余数
gcd(x，y)	整数 x 和 y 的最大公因数
lcm(x，y)	整数 x 和 y 的最小公倍数
exp(x)	自然指数
pow2(x)	2 的指数
log(x)	以 e 为底的对数，即自然对数
log2(x)	以 2 为底的对数
log10(x)	以 10 为底的对数

表 2.3　MATLAB常用的三角函数

函数名称	函数功能
sin(x)	正弦函数
cos(x)	余弦函数
tan(x)	正切函数
asin(x)	反正弦函数
acos(x)	反余弦函数
atan(x)	反正切函数
atan2(x，y)	四象限的反正切函数
sinh(x)	双曲正弦函数
cosh(x)	双曲余弦函数
tanh(x)	双曲正切函数
asinh(x)	反双曲正弦函数

续表2.3

函数名称	函数功能
acosh(x)	反双曲余弦函数
atanh(x)	反双曲正切函数

注意： 上面的 x，y，z 都必须是已定义的变量。

2.3 MATLAB 矩阵和数组的创建和保存

从本节开始介绍 MATLAB 最重要、最基础的部分——矩阵运算。在开始之前，再谈谈 MATLAB 中矩阵运算的特点，使用户对此有更深的印象。首先，MATLAB 中矩阵是运算的基本单元，该单元是定义在复数域上的。其次，MATLAB 中所有的矩阵事先都不必定义维数大小。系统会根据用户的输入自动配置，在运算中自动调整矩阵的维数。

2.3.1 直接输入创建的矩阵

当需要的矩阵的维数比较小的时候，最方便、最直接的方法是在 MATLAB 工作区中直接手工输入矩阵。输入的格式如下所示：

整个矩阵以“[ ]”作为首尾，行与行之间必须用分号“；”或按 Enter 键分隔。每行中的元素用逗号“，”或空格分隔。

矩阵中元素可以是数字或表达式。但表达式中不可包含未知的变量，MATLAB 用表达式的值为元素赋值。如果矩阵中没有元素，这样的矩阵称为“空阵”(Empty matrix)。空阵有极其广泛的作用。

【例 9】用指令产生数值矩阵

```
x=9;y=pi/6;              %定义 x，y 变量
A=[3 5 sin(y)
  cos(y) x^2 7
    x/2 5 1]               %产生矩阵 a
A=
   3.0000    5.0000    0.5000
   0.8660   81.0000    7.0000
   4.5000    5.0000    1.0000
```

说明：%作为 MATLAB 注释的开始标志，以后的文字不影响计算的过程。

矩阵中的元素可以用它的行和列数表示，例如 A(2，3)表示矩阵 A 的第二行第三列的元素 7。对于一个已经建立的矩阵，MATLAB 有许多方法修改它的内容。

【例 10】矩阵元素的修改

```
A(3，3)=0
```

```
A =
    3.0000    5.0000    0.5000
    0.8660   81.0000    7.0000
    4.5000    5.0000         0
```

说明： A(3，3)=0 将矩阵 A 的第三行第三列的元素变成 0。

```
A(2，6)=1
A=
    3.0000    5.0000    0.5000         0         0         0
    0.8660   81.0000    7.0000         0         0    1.0000
    4.5000    5.0000         0         0         0         0
```

A(2，6)=1 在矩阵的第二行第六列置为 1。由于矩阵 A 没有 6 列，MATLAB 系统自动增加 A 的列数以适应需要，并将其他位置置为 0。

前面谈到复数的时候，提到矩阵中也可以使用复数，下面看看如何用复数建立矩阵。复数矩阵的建立和输入如下所示：

```
c=[1，2+I*3;4，5*j];
```

复数矩阵的输入还有一个特殊的方法，即用一个矩阵表示复数的实部，而用另一个矩阵表示复数的虚部。最后将两个矩阵相加，可以得到所需的矩阵。

【例 11】复数矩阵的特殊输入法

```
R=[2，4，6;7，8，9];I=[6，9，2;34，56，89];
F=R+I*i;
```

显示 F 矩阵的内容如下：

```
F =
   2.0000 + 6.0000i   4.0000 + 9.0000i   6.0000 + 2.0000i
   7.0000 +34.0000i   8.0000 +56.0000i   9.0000 +89.0000I
```

2.3.2　由矩阵编辑器创建和修改矩阵

当输入的矩阵很大，不适合用手工直接输入时，MATLAB 提供了一个矩阵编辑器(Matrix Editor)来方便用户创建和修改比较大的矩阵。在调用矩阵编辑器之前，需要预先定义一个变量，无论是个数值还是个矩阵均可。

```
w=[2  1
   3  4];          %定义一个名为 w 的变量
```

以下是具体的操作步骤：

(1) MATLAB 提供工具栏按钮来查看工作区变量，如图 2.2 所示。

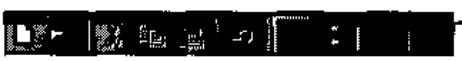


图 2.2　工具栏

(2) 单击图示的按钮会显示图 2.3 所示的工作区浏览器。

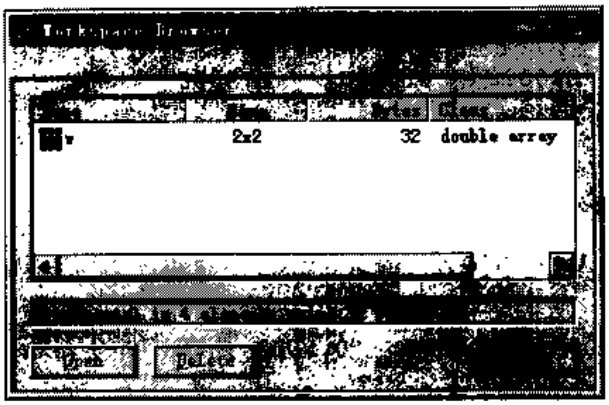


图 2.3 工作区浏览器

(3) 选中 W 变量就可以打开或者删除 W。双击鼠标左键或者单击 Open 按钮，就可以打开矩阵编辑器，如图 2.4 所示。

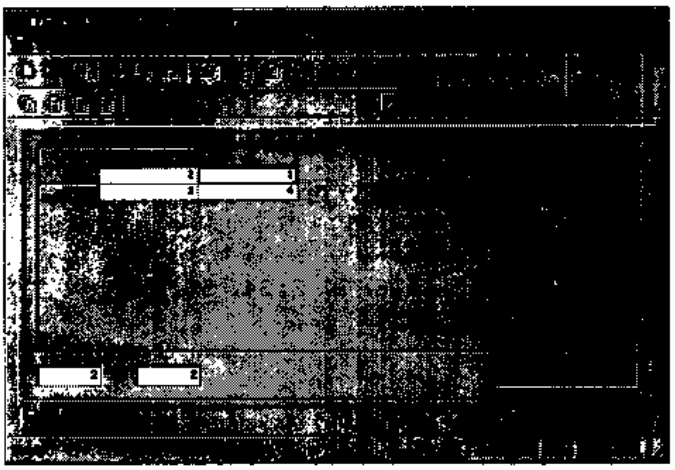


图 2.4 矩阵编辑器

(4) 改变矩阵的维数。在左下方的 2 个文本框中(用 by 连接在一起)分别表示矩阵的行和列，这里 W 维数是 2×2。这个维数值可根据用户的要求调整。

(5) 窗口右下方文本框显示的“(1，2)”是现在选中的元素的标号(Index)。通过选中该元素，用户可以改变元素的值。改变数值可以直接输入数值，也可以输入表达式，MATLAB 系统会自动计算该值。

(6) 当确认所有的元素都正确无误后，可以按下 Close 按钮，于是 W 就定义保存好了。

矩阵编辑器可以方便地创建和修改矩阵。还可以把原矩阵“裁减”为它的左上方的主子矩阵，或反过来把原矩阵“扩展”为更大的矩阵，MATLAB 系统会自动地把扩展的部分设置为 0。

2.3.3 由函数创建和修改矩阵

MATLAB 提供了大量的函数用于创建一些常用的特殊矩阵，像对角阵、单位阵和零矩阵等在数值计算中常用的矩阵以及由其经变换和截取产生的新矩阵。下面分别介绍这些内容。

1. 一些常用的特殊矩阵的生成

【例 12】 0~1 分布的随机矩阵，用 rand 函数可以产生任意行列的 0~1 分布的随机矩阵

```
ra=rand(2, 3)
ra =
0.9501    0.6068    0.8913
0.2311    0.4860    0.7621
```

这样产生了 2 行 3 列的矩阵，当然这里产生的矩阵和在用户的机器上产生的矩阵几乎不可能一样。

以下是 MATLAB 中用于产生常用特殊矩阵的函数：

- zeros(m，n)　零矩阵
- ones(m，n)　全部元素都为 1 的矩阵
- eye(m，n)　单位阵
- randn(m，n)　正态分布的随机矩阵
- compan(A)　矩阵 A 的伴随矩阵
- gallery　测试矩阵
- hankel(m，n)　n 维 Hankel 矩阵
- invhilb(n)　n 维逆 Hilbert 矩阵
- magic(n)　n 维 Magic 方阵
- toeplitz(m，n)　Toeplitz 矩阵
- wilkinson(n)　n 维 Wilkinson 特征值测试矩阵
- handamard(n)　n 维 Hadamard 矩阵
- hilb(n)　n 维 Hilbert 矩阵
- kron(A，B)　Kronecker 张量积
- pascal(n)　n 维 Pascal 矩阵
- vander(A)　由矩阵 A 产生 Vandermonde 矩阵

说明：它们的使用方法请参看 MATLAB 联机帮助，用户可以自己试验。

【例 13】利用 diag 产生对角阵

对角阵是线性代数中经常使用的矩阵。MATLAB 中有专门产生对角阵的函数 diag(M)，M 是一个矩阵或是一个向量。对于矩阵，diag 函数取矩阵的对角元产生一个列向量；而对于向量则产生一个对角阵。

```
a=randn(5, 5);
d=diag(a);
D=diag(d);
a =
   -0.4326    1.1909   -0.1867    0.1139    0.2944
   -1.6656    1.1892    0.7258    1.0668   -1.3362
    0.1253   -0.0376   -0.5883    0.0593    0.7143
```

```
    0.2877    0.3273    2.1832   -0.0956    1.6236
   -1.1465    0.1746   -0.1364   -0.8323   -0.6918
d =
   -0.4326
    1.1892
   -0.5883
   -0.0956
   -0.6918
D =
   -0.4326         0         0         0         0
         0    1.1892         0         0         0
         0         0   -0.5883         0         0
         0         0         0   -0.0956         0
         0         0         0         0   -0.6918
```

【例 14】求 Kronecker 张量积

```
A=[1 2 3;4 5 6];
B=[2 4;6 8];
C=kron(A, B)
C =
    2    4    4    8    6   12
    6    8   12   16   18   24
    8   16   10   20   12   24
   24   32   30   40   36   48
```

2. 矩阵标识和子矩阵

矩阵的子阵可以通过标量、向量、冒号的标识来引用和赋值。

- 子阵的序号向量标识方式 A(v，w)

v，w 可以是任何排列的向量。用户最好使用序号单调向量，否则可能产生混乱。v，w 中任一个可以是冒号“：”，它表示全部行(在 v 的位置)或列(在 w 的位置)。

v，w 所用的值必须大于或等于 1 而且小于或等于矩阵的维数，否则失败。

- “0~1”向量标识方式 A(L1，：)、A(:，L2)、A(L1，L2)

向量 L1，L2 的长度分别为矩阵 A 的行数和列数；向量 L1、L2 中的元素或取 1(表示提取相应的行或列)或取 0(不提取)；作这种标识方法主要为了与关系运算配合使用。

【例 15】提取矩阵的子阵

```
B=magic(5)              % 产生 5 维的魔方阵 B
B1=B(1:2, [1, 3, 5])          % 提取矩阵 B 的第一行和第二行的第 1，3，5 个元素
B2=B([3, 1], : )            % 提取矩阵 B 的第三行和第一行全部元素
B([1, 3], [2, 4])=zeros(2)    % 使得矩阵 B 的第一行和第三行的第 2，4 个元素为 0
B =
```

```
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
B1 =
    17     1    15
    23     7    16
B2 =
     4     6    13    20    22
    17    24     1     8    15
B =
    17     0     1     0    15
    23     5     7    14    16
     4     0    13     0    22
    10    12    19    21     3
    11    18    25     2     9
L=B(1, : <5)      %标出 B 矩阵第一行中小于 5 的元素
B3=B(1, L)        %获得矩阵 B 中第一行小于 5 的子向量
L =
     0     1     1     1     0
B3 =
     0     1     0
```

3. 矩阵的结构变换

可以通过矩阵的旋转、改变维数和截取部分元素来产生用户需要的新矩阵。MATLAB 提供以下矩阵变换命令和函数：

- B=rot90(A)　B 由矩阵 A 逆时针方向旋转 90° 而得
- B=rot90(A，k)　B 由矩阵 A 逆时针方向旋转 k×90° 而得
- B=fliplr(A)　B 由矩阵 A 左右翻转而得
- B=flipud(A)　B 由矩阵 A 上下翻转而得
- B=reshape(A，m，n)　B 阵的维数为(m×n)，m×n 等于矩阵 A 的列维和行维之积
- L=tril(A，k)　L 第 k 条对角线及以下的元素取矩阵 A 元素，其余为 0
- L=tril(A)　L 主对角线及以下的元素取矩阵 A 元素，其余为 0
- U=triu(A，k)　U 第 k 条对角线及以上的元素取矩阵 A 元素，其余为 0
- U=triu(A)　U 主对角线及以上的元素取矩阵 A 元素，其余为 0

说明：k 取 0 为主对角线；k 取正整数为主对角线上第 k 条对角线；k 取负整数为主对角线下第 k 条对角线。

以下分别举例说明各函数的作用和用法。

【例 16】矩阵的旋转和转置的区别

```
A=[1, 2, 3, 4;5, 6, 7, 8;9, 10, 11, 12]
B1=rot90(A)
BT=A'
B2=rot90(A, 2)
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
B1 =
     4     8    12
     3     7    11
     2     6    10
     1     5     9
BT =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
B2 =
    12    11    10     9
     8     7     6     5
     4     3     2     1
```

【例 17】矩阵的变维

```
B=reshape(A, 2, 6)          % 矩阵 A 取自上例
B =
     1     9     6     3    11     8
     5     2    10     7     4    12
```

【例 18】部分元素的截取

```
LA=tril(A, -1)   % 矩阵 A 取自上例
UA=triu(A, 1)
LA =
     0     0     0     0
     5     0     0     0
     9    10     0     0
UA =
     0     2     3     4
     0     0     7     8
```

```
     0     0     0    12
```

2.3.4　矩阵的保存和提取

对于大型的项目来说，经常有大量的矩阵需要保存和再次使用时提取。当然最好的办法是把辛苦得来的数据保存于文件中。在 MATLAB 中用 Mat 文件来保存二进制的数据。可以使用 Mat 来保存矩阵，具体步骤如下：

(1) 保存矩阵。如果 A，B 矩阵都已存在，用 save 命令保存，具体格式如下：

```
save mymatrix A B
```

注意： mymatrix 是用户自定义的文件名，MATLAB 系统会自动地加上后缀.MAT，而系统默认的路径是 MATLABr11\bin。如果用户想要改变路径，可以在文件名前加上路径。

(2) 提取矩阵。在重新启动 MATLAB 后，用 load 命令可以将保存在文件中的矩阵读到 MATLAB 工作区的内存中来。

```
load mymatrix
```

注意： load 命令中不能指定变量名，系统仍然将 A，B 作为矩阵的名称。用户可以用 who 命令来查看 MATLAB 工作区的变量。事实上数值矩阵还可以借助 save 命令以 ASCII 码的形式保存，以供其他应用程序使用。这部分内容将留在 MATLAB 与其他程序的接口中介绍。

2.3.5　数组的建立和保存

在 MATLAB 中数组可以看作是行向量，即只有一列的矩阵。前面介绍的所有矩阵的建立和保存的方法对于数组都可以使用，同时 MATLAB 系统还提供了一些创建数组的特殊命令。

【例 19】创建等差数列

在数学计算中，有时会用到等差数列。MATLAB 中可以很方便地创建等差数列如下所示：

```
a=0:0.05:1;              % 以“：”间隔起始值(=0)、增量值(=0.0.5)、终止值(=1)
x=linspace(0，1，75);  % 利用 linspace 以间隔起始值(=0)、终止值(=1)和元素数目(=75)
```

说明：

- 在第一种方式中，如果增量值省略不写，直接表达为 0:1，那么 MATLAB 默认的增量值为 1。
- 第二种方式常用于绘图中区间的分割。

还可以从已知的数组构造新的数组。

【例 20】从原来的数组创建新的数组

```
a=1:4;b=1:2:7;
```

```
c=[b, a];
c=
    1    3    5    7    1    2    3    4
%将两个数组 a，b 连接在一起
d=[a(1:2:4) 4 .2 8]
d =
    1    3    4   .2    8
```

这里 d 的前两项是由 a 的 1，3 两项组成的。需要指出的是，数组的元素与矩阵一样也可以通过下标来访问。

MATLAB 中还有一个与 linspace 相似的函数 logspace，用来创建等比数列。

【例 21】用 logspace 创建等比数列

```
logspace(0, 2, 11)
ans =
  Columns 1 through 7
    1.0000    1.5849    2.5119    3.9811    6.3096   10.0000
15.8489
  Columns 8 through 11
   25.1189   39.8107   63.0957  100.0000
```

这里 logspace 函数产生了一个起点为 10^0、终点为 10^2、包含 11 个数据的等比数列，其公比为 $10^{0.2}$。

2.4 矩阵运算及数组运算

数组是一种特殊的矩阵，数组的运算和矩阵的运算既有相似的地方又有很多不同。总的来说，矩阵的运算按照代数中矩阵的运算规则进行；而数组的运算都是按单个元素进行的，MATLAB 通过不同运算符来识别这两种运算。把这两种运算结合在一起介绍，有助于用户灵活地掌握它们。

2.4.1 MATLAB 的矩阵运算

在 MATLAB 系统中提供了如下的矩阵运算符：

+ 加法　－ 减法　* 乘法　^ 幂　\ 左除　/右除　' 转置

这些矩阵运算符要符合矩阵运算的规律：即具有相同的行与列的矩阵可以进行“加法”和“减法”运算；矩阵 A 的列数和矩阵 B 的行数相同时 A*B 才有意义；而矩阵的幂要求矩阵是个方阵(矩阵的行数等于列数)。如果矩阵的行数和列数不符合运算符的要求，则会产生错误的信息。

这里有一个例外的情况：如果两个矩阵中有一个是标量矩阵(1×1 矩阵)，则矩阵的上述运算依然可以进行，不过它的含义有所不同。这时的矩阵运算结果是标量和矩阵的每一

个元素“相加”、“相减”、“相乘”或“相除”。

由于加法和减法的计算很简单，这里就不作介绍。下面给出一个矩阵乘法的小例子，用户可以从中体会到 MATLAB 系统的一些特点。

【例 22】 矩阵的乘法

```
A=rand(3, 3);              %产生 3×3 的随机矩阵
B=rand(3, 2);              %产生 3×2 的随机矩阵
C=rand(2, 3);              %产生 2×3 的随机矩阵
M=C*A^2*B;              %计算矩阵的乘积
M =
    0.4446  0.7546
    0.4568  0.7723
```

根据矩阵运算的规则，C 和 A^2 相乘是 2×3 的矩阵，最后得到的矩阵必定是个 2×2 的矩阵。

2.4.2 矩阵的除法运算

在 MATLAB 中矩阵的除法有其特别之处。联机性代数中，只有矩阵逆的定义，并没有矩阵除法的运算。MATLAB 引入矩阵的除法运算，有着丰富的内涵。MATLAB 系统提供了 2 种除法指令：左除和右除。其中右除定义为：B\A=(A'/B')'，用户只要掌握了左除的使用就可以举一反三了。

矩阵除法可以看作是矩阵乘法的逆运算，当然可以像做算术一样应用矩阵的除法。但是在实践中， 矩阵除法的主要作用在于解 n 元一次方程组。 定义方程组 Ax=b，A 是 m×n 的矩阵，b 是 n 行向量。

为说明除法运算的意义，先介绍求逆命令。在 MATLAB 中，对于任何方阵都可以用下述命令来求逆：

inv(A)　求矩阵的逆，但 A 必须是方阵

注意：

- 由于 MATLAB 采用 IEEE 算法，所以即使 A 是奇异阵(A 的行列式值是 0)，运算照样进行，但会给出警告信息“Warning: Matrix is singular to working precision.”；而求出的矩阵所有元素的值为 Inf(无穷大)。
- 当 A 是病态阵时，MATLAB 使用的算法计算产生的误差可能会很大，因此 MATLAB 系统此时会给出警告信息“Warning: Matrix is badly scaled to working precision.”

1. 矩阵求逆和矩阵除法解静定方程

- 利用矩阵求逆来解方程组

```
x=inv(A)*b
```

- 采用左除运算解方程组

```
x=A\b
```

【例 23】矩阵求逆和矩阵除法解方程组

```
A=rand(10)        %产生(10×10)均匀分布随机矩阵
b=ones(10, 1)     %产生全为 1 的 10 元列向量
x1=inv(A)*b;      %矩阵求逆解方程组
x2=A\b;           %矩阵除法解方程组
x1 =
   -0.0952
    0.5617
    0.4973
    0.0967
    0.3024
    0.4356
    0.3196
   -0.0188
    0.1808
   -0.1646
x2 =
   -0.0952
    0.5617
    0.4973
    0.0967
    0.3024
    0.4356
    0.3196
   -0.0188
    0.1808
   -0.1646
```

提示： 由于两种解法所用的算法不一样，用除法解方程的速度大大快于矩阵求逆法，而所得解的精度差不多。MATLAB 中求逆函数(Inv)时，用的是高斯消去法；而用除法解方程时是直接用高斯消去法求解，有效地加快了速度。

2. 求解的方程是超静定方程

有如下 2 种解法：

- 求正则方程(Normal equations)$(A^TA)X=A^Tb$ 的解。
- 用 Householder 变换(Householder Transformation)直接求解方程的最小二乘解。

由于第二种方法采用的是正交变换，由最小二乘法理论可知，第二种方法所得的解准确、可靠。MATLAB 系统解超静定方程的除法运算算法就是用的第二种方法。

【例 24】 用矩阵除法解超静定方程

```
A=[3, 4, 6;5, 7, 9;2, 5, 8;3, -9, 6];
b=[2 3 4 7]'
```

```
x=A\b;
x =
   -0.4025
   -0.3624
    0.8226
```

提示：　用户可以利用这一方法来做直线拟合：矩阵 A 的第一列表示直线的横坐标，A 的第二列全部置为 1，b 是对应的纵坐标。解出 x 的两项分别是斜率和截距。

3. 求解的方程是欠静定方程

欠静定方程是不确定的。用除法运算所得的解将有两个重要特征：在解中至多有 Rank(A)(矩阵 A 的秩)个非零元素；而该解是这样类型解中范数(norm)最小的一个。

【例 25】用矩阵除法解欠静定方程

```
A=A';   % A取自上例
b=[2 3 4]';
x=A\b;
x =
         0
    0.3636
    0.0909
   -0.0000
```

2.4.3　矩阵的乘方运算

1. 方阵的标量乘方A^p

- 当 p 取整数时，该指令的运算结果可作如下的理解：
  当 p>0 的时候，A^p 表示方阵 A 的直接自乘 p 次；
  当 p<0 的时候，A^p 表示方阵 A 的直接自乘 p 次后的逆，仅对非奇异阵成立；
  当 p=0 的时候，A^0 将给出与 A 阵同维的单位阵。
- 当 p 取非整数时，该指令的运算结果可作如下的理解：

若 A 可以分解为 $A=WDW^{-1}$，D 为对角阵，那么可以定义

$$A^p=WD^pW^{-1}$$

说明：

- 如果 A 的特征值方程有重根，以上指令不适合。
- 对于有些矩阵的非整数次方，可能存在多个解，MATLAB 仅给出一个解。
- MATLAB 有一个命令专门求矩阵的平方根 sqrtm(A)，算法和此一样。

【例 26】求矩阵的非整数乘方

```
A=[3, 6, 7;9, 2, 5;1, 6, 3];
Ap=A^0.3;
```

```
Ap =
   1.4908 - 0.0000i   0.2551 - 0.0000i   0.6711 + 0.0000i
   2.1879 + 0.0000i   0.5145 - 0.0000i  -0.6590 + 0.0000i
  -1.4515 - 0.0000i   1.3756 - 0.0000i   2.2378 + 0.0000i
```

下面是按矩阵非整数乘方的定义进行验证的指令：

```
[W, D]=eig(A); % 求矩阵 A 的特征值和特征向量
Aap=W*D^.3/W;
Aap =
   1.4908 - 0.0000i   0.2551 - 0.0000i   0.6711 + 0.0000i
   2.1879 + 0.0000i   0.5145 - 0.0000i  -0.6590 + 0.0000i
  -1.4515 - 0.0000i   1.3756 - 0.0000i   2.2378 + 0.0000i
```

2. 标量的矩阵乘方p^A

若 A 可以分解为 $A=WDW^{-1}$，D 为对角阵，那么可以定义求标量矩阵乘方为

$$p^A = W\begin{bmatrix} p^{d_{11}} & & \\ & \ddots & \\ & & p^{d_{nn}} \end{bmatrix}W^{-1}$$

【例 27】求标量矩阵乘方

```
pA=(0.5)^A;        % A 阵取自上例
pA =
   1.4867 + 0.0000i   0.2124 - 0.0000i  -2.5695 - 0.0000i
 -13.2981 - 0.0000i   8.4944 + 0.0000i   6.7506 + 0.0000i
  11.3044 + 0.0000i  -8.2394 + 0.0000i  -4.1478 - 0.0000I
```

可以看到若 A 矩阵的分解有复数，那么标量矩阵乘方计算出的结果可能有复数。

2.4.4 数组运算

数组运算无论对于哪种运算操作都是对元素逐个进行的。抓住这个特点就不难理解数组运算的特点。MATLAB 设计这种运算的目的在于使大批数据的处理和标量情况相似，可以大大简化使用和编程，并便于阅读。

在 MATLAB 系统中提供了如下的数组运算符：

.+加法　.— 减法　.* 乘法　.^ 幂　.\ 除(./)　.' 共轭

注意： 运算符中的小黑点绝对不能遗漏，否则将不按数组运算规则进行计算。不管执行什么数组计算，所计算结果数组总是与参与运算的数组同维。数组运算中所有的二元运算必须是同维的数组或者其中有一个是标量。

数组的运算十分简单，除一些特殊情况外，其他各种运算请用户参阅帮助文件和其他资料。

1. 数组除

数组除的运算规则：

- 当参与运算的是两个同维的数组时，运算为数组对应的元素相除，所得的结果是和参加运算的数组同维的数组。
- 当参与运算的是有一个是标量时，运算为标量和数组中每一个元素相除，所得的结果是和参加运算的数组同维的数组。
- 左除和右除的关系为 A./B=B.\A，A 是被除数，B 是除数。

2. 数组乘方

数组乘方的运算规则：

- 数组的标量乘方 A.^p，运算为数组对应的元素 p 次方，所得结果为和数组 A 维数相同的数组。
- 标量的数组乘方 p.^A，运算为 p 的数组对应元素的次方，所得结果为和数组 A 维数相同的数组。

【例 28】数组的乘方

```
A=[3, 6, 7;9, 2; 5;1, 6, 3];
Ap=A.^0.4
Ap =
   1.5518    2.0477    2.1779
   2.4082    1.3195    1.9037
   1.0000    2.0477    1.5518
pA=0.4.^A
pA =
   0.0640    0.0041    0.0016
   0.0003    0.1600    0.0102
   0.4000    0.0041    0.0640
```

2.5　数组函数和矩阵函数

MATLAB 提供的函数有 2 种：一是按照数组运算法则设计的，称为数组函数；另一种是按照矩阵运算法则设计的，称为矩阵函数。

2.5.1　数组函数

对于数组中每一个元素进行函数运算，运算结果为和原数组维数一样的数组。除了所有标量的运算函数可以运用于数组外(参看 2.2 节)，MATLAB 还提供表 2.4 所列的特殊数组函数。

表 2.4 特殊数组函数

函数名称	功能简介
besselj(NU，Z)	解第一类 Bessel 微分方程
bessely(NU，Z)	解第二类 Bessel 微分方程
beta(Z，W)	计算 Beta 函数值
gamma(X)	计算 Gamma 函数值
rat(X，tol)	计算 X 的有理近似 (tol 表示容差)
erf(X)	计算 X 误差函数
erfinv(Y)	求逆误差函数
ellipke(M，tol)	求第一类、第二类全椭圆积分 (tol 表示容差)
ellipj(U，M)	Jacobi 椭圆函数

说明：

以上函数调用的参数中除了 tol 表示容差外，其他参数必须是维数一样的数组或者其中任一为标量。有关函数变量和输出结果的具体含义，请用户参看 MATLAB 联机帮助文件。

2.5.2 基本矩阵函数

矩阵行列式的值、矩阵的秩、特征值等在代数中应用广泛的概念，MATLAB 系统提供了相应的函数来求其值。表 2.5 给出了常用的矩阵函数命令。

表 2.5 常用的矩阵函数命令

函数名称	功能简介
cond(A)	矩阵 A 的条件数
det(A)	方阵 A 的行列式值
dot(A，B)	矩阵 A，B 的点积
eig(A)	方阵 A 的特征值和特征向量
norm(A，1)	矩阵 A 的 1-范数
norm(A)或 norm(A，2)	矩阵 A 的 2-范数
norm(A，inf)	矩阵 A 的无穷大-范数
norm(A，'fro')	矩阵 A 的 F-范数
rank (A)	矩阵 A 的秩
rcond(A)	矩阵 A 的倒条件数
svd(A)	矩阵 A 的奇异值分解
trace(A)	矩阵 A 的迹

续表2.5

函数名称	功能简介
expm(A)	矩阵的指数 e^A
expm1(A)	用 Pade 法求矩阵的指数 e^A
expm2(A)	用 Taylor 级数求矩阵的指数 e^A，精度差，对任何矩阵都适用
expm3(A)	用特征值和特征向量求矩阵的指数 e^A，仅当独立特征向量个数等于矩阵秩时适用
logm(A)	求矩阵 A 的对数
sqrtm(A)	求矩阵的平方根
funm(A，'fun')	一般的方阵函数

说明：

有关函数的意义请参考有关代数的教科书，具体调用形式请参考 MATLAB 的联机帮助文件。

注意： funm 函数的问题：

F = FUNM(A，'fun') 对于方阵 A，计算用函数'fun'定义的矩阵函数。例如：FUNM(A，'sin') 就是计算矩阵 A 的 sin 函数。

FUNM 使用的算法有潜在的不稳定性。如果 A 是接近于一个多重特殊值和病态特征向量的矩阵，FUNM 会产生不精确的值。MATLAB 系统会因此提示警告信息，但是系统有时候太敏感了，当结果正确时也会提示警告信息。如果 A 是对称阵或是 Hermit 阵，FUNM 可以产生比较精确的值。S=SQRTM(A)和 L=LOGM(A)使用 FUNM 做计算，可以得到更可靠的结果，而 EXPM(L)使用完全不同的算法。

【例 29】矩阵常用函数举例

```
A=rand(5);  % 产生 5×5 的随机分布矩阵
s= svd(A)   % 方阵 A 的奇异值分解
s=
2.8858    1.0439    0.6915    0.4568    0.0075
d=det(A)    %  方阵 A 的行列式值
d =
   -0.0071
t=trace(A)  % 方阵 A 的迹
t =
2.8776
rk=rank(A)  %  方阵 A 的秩
rk =
     5
```

```
c=cond(A)  %  方阵 A 的条件数
c =
  386.0703
n1=norm(A, 1) % 方阵 A 的 1-范数
n1 =
3.5620
n2=norm(A, 2)  % 方阵 A 的 2-范数
n2 =
2.8858
ninf=norm(A, inf)  % 方阵 A 的无穷大-范数
ninf =
3.2772
nf=norm(A, 'fro')  % 方阵 A 的 F-范数
nf =
3.1788
```

矩阵和数组函数运算的含义不同，请用户注意区分。

【例 30】矩阵和数组函数运算的区别

```
B=[1/6, 1/2;1/3, 5/6]*pi;
sinb=sin(B);                % 数组函数运算，对每个元素求 sin 值。
sinb =
    0.5000    1.0000
    0.8660    0.5000
fsinmb=funm(B, 'sin');      % 矩阵函数运算，先进行矩阵特征值分解
fsinmb =
   -0.0849    0.0000
    0.0000   -0.0849
```

2.5.3 矩阵分解函数

矩阵的分解在代数中占有重要的地位，是矩阵和数据分析的基础。MATLAB 系统提供了大量的矩阵分解函数，如表 2.6 所示。

表 2.6 基本的矩阵分解函数

函数指令	功能简介
cdf2rdf(V, D)	复数对角形转换成实数块对角形
chol(A)	矩阵 A 的 Cholesky 分解
eig(A)	矩阵 A 的特征值分解
Hess(A)	矩阵 A 的 Hessenberg 形式

续表2.6

函数指令	功能简介
LU(A)	矩阵 A 的 LU 分解
null(A)	由奇异值分解得出的矩阵 A 的零空间的标准正交基
orth(A)	矩阵 A 行向量的标准正交基
pinv(A)	求矩阵 A 的伪逆
qr(A)	矩阵 A 的 QR 正交三角分解
qz(A)	矩阵 A 的 QZ 分解，用于广义特征值
rref(A)	将矩阵 A 转换为逐行递减的阶梯阵
rsf2csf(V，D)	实数块对角形转换成复数对角形
schur(A)	矩阵 A 的 Schur 分解
subspace(A，B)	计算由 A，B 张成的子空间的夹角
svd(A)	方阵 A 的奇异值分解

下面比较详细地介绍特征值分解和奇异值分解指令，其他函数的具体调用形式请参考 MATLAB 的联机帮助文件。

1. 特征值分解

特征值分解命令有如下 5 种形式：

- E = eig(X)

E 是个包含矩阵特征值的向量。

- [V，D] = eig(X)

产生一个对角元是特征值的对角阵 D 和一个行向量是特征值对应特征向量的满秩矩阵，因此满足 XV = VD。

- [V，D] = eig(X，'nobalance')

禁止“平衡”程序的运行，用于在 X 阵中有的元素小到与截断误差相当时，可以减少计算的误差。

- E = eig(A，B)

E 是个包含矩阵 A，B 广义特征值的向量。

- [V，D] = eig(A，B)

产生一个对角元是广义特征值的对角阵 D 和一个行向量是广义特征值对应特征向量的满秩矩阵， 因此满足 A*V = B*V*D。

【例 31】特征值运算

```
A=[3    -2      eps  1
  1      eps     5   3
  -eps/4  eps/2   -4   0
  3      8*eps   -1  eps ];
```

```
[v1, d1]=eig(A);
Er1=A*v1.v1*d1;                    % 一般特征值分解的误差
Er1 =
  1.0e-006 *
  -0.0000 - 0.0000i  -0.0000 + 0.0000i   0.0000   -0.0048
   0.0000 + 0.0000i   0.0000 - 0.0000i  -0.0000    0.1355
   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000    0.0000
  -0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000    -0.0122
[v2, d2]=eig(A, 'nobalance');    % 无“平衡”特征值分解的误差
Er2=A*v2-v2*d2
Er2 =
  1.0e-014 *
  -0.1221 - 0.2220i  -0.1221 + 0.2220i  -0.1665   -0.2442
  -0.0666 - 0.0167i  -0.0666 + 0.0167i   0.0666    0.0444
   0.0168 - 0.0041i   0.0168 + 0.0041i  -0.0065    0
  -0.1332 . 0.1110i  -0.1332 + 0.1110i  -0.1110    -0.1110
```

提示： ●在本例中，若采用一般的特征值分解指令，那么所得的结果有比较大的误差。从理论上来说,A*v1.v1*d1 得出的应该是零矩阵。这是由于在执行本命令时要调用使原矩阵各元素数量大致相当的平衡程序。该程序使得本例中本可忽略的小元素作用放大了。

●当A阵来自计算结果而且其中含有“零元素”时，会发生本例中情况，这时建议用户使用 nobalance 命令。但是一般情况下，“平衡”程序的作用是减小计算误差。

2. 复数特征值对角阵与实数块特征值对角阵的转换

即使是实数阵也可能会有复数特征值。在实际应用中，常需要把这一对共轭复数特征值相互转换为一个 2×2 的实数块。为此 MATLAB 系统提供了下面 2 个指令用于这两种形式的相互转换：

```
[VR, DR]=cdf2rdf(VC, DC)        把复数对角形转换为实数块对角形
[VC, DC]=rsf2csf(VR, DR)        把实数块对角形转换为复数对角形
```

其中，DC 是含复数的特征值对角阵，VC 是与之相应的特征向量阵；DR 是含实数块的特征值对角阵，VR 是与之对应的实特征向量阵。

【例 32】把复数对角形转换为实数块对角形

```
A=[1,   -3
   2,  2/3];
[V, D]=eig(A)    % 求矩阵 A 的特征值分解，注意特征值是复数
V =
  -0.7728 + 0.0527i  -0.7728 - 0.0527i
        0 + 0.6325i        0 - 0.6325i
```

```
D =
   0.8333 + 2.4438i        0
    0                  0.8333 - 2.4438i
[VR, DR]=cdf2rdf(V, D)        % 求矩阵 A 的实数块形式特征值
VR =
   -0.7728    0.0527
         0    0.6325
DR =
    0.8333    2.4438
   -2.4438    0.8333
Er1=VR*DR/VR-A            % 验证 VR，DR 确实满足关系：VR*DR=A*VR
Er1 =
  1.0e-015 *
         0    0.4441
         0.4441   -0.2220
```

3. 奇异值分解

奇异值分解在矩阵分析中占有极重要的作用。有关奇异值分解的定义请参看任何一本高等代数书。MATLAB 系统提供的奇异值分解函数 svd 是利用 LINPACK 程序库中的 ZSVDC 编制而成的。假如经过 75 步 QR 分解迭代仍得不到一个奇异值，那么 MATLAB 系统会给出“不收敛”的提示。奇异值分解指令的调用格式如下：

- [U，S，V] = svd(X)

产生和矩阵 X 同维的对角阵 S 和酉矩阵 U 和 V，其中 S 的对角元非负并且按降序排列，满足 X = U*S*V'。

- S = svd(X)

返回 S 是包含奇异值的列向量。

- [U，S，V] = svd(X，0)

产生“经济大小”的分解。如果 X 是 m×n 的矩阵，并且 m>n，那么只计算 U 矩阵的前 n 行，而 S 也是个 n×n 的矩阵。

【例 33】奇异值分解

```
A=[61   38
   6    9];
 [U, S, V]=svd(A)
U =
    0.9907   -0.1363
    0.1363    0.9907
S =
   72.5425         0
         0    4.4250
```

```
V =
    0.8443   -0.5359
    0.5359    0.8443
```

4. 伪逆

和奇异值分解联系紧密的概念是 Moor-Penrose 伪逆(Pseudoinverse)。在 MATLAB 系统中提供求伪逆的函数 pinv，具体调用格式为：

- X = pinv(A)

产生和矩阵 A 的转置同维的矩阵 X，使得 A*X*A = A， X*A*X = X， 同时 A*X 和 X*A 是 Hermitian 矩阵。该算法基于奇异值分解命令 svd(A) 并且任何小于容差的奇异值都当作 0 处理。默认的容差为 tol=max(size(A)) * norm(A) * eps。

- pinv(A，tol)

使用用户指定的容差 tol。当用伪逆去求解列不满秩超静定最小二乘问题 Ay=b 的时候，它给出的解 y'满足：

$$\|Ay'-b\| = \min\|Ay-b\|$$

即为最小范数的最小二乘解。

【例 34】用伪逆求解最小二乘问题

```
A=magic(8);A=A(:, 1:6);
b=260*ones(8, 1);
yp=(pinv(A)*b)'        %伪逆法解最小二乘问题
yd=(A\b)'
np=norm(yp)            %求伪逆法解的范数
nd=norm(yd)            %求除法解的范数
yp =
   1.1538    1.4615    1.3846    1.3846    1.4615    1.1538
Warning: Rank deficient, rank = 3  tol =  1.8829e-013.
yd =
   4.0000    5.0000         0         0         0   -1.0000
np =
   3.2817
nd =
   6.4807
```

提示： 用伪逆法得出的最小二乘解范数最小，而用除法得出的最小二乘解非零元素的个数等于矩阵 A 的秩。

第 3 章
高级数值计算

本章要点：

本章将介绍一些有关数值计算的高级内容。第 2 章介绍了 MATLAB 的基本运算功能，而高级数值计算提供了更多更有效的函数和命令，是 MATLAB 强大的数值功能扩展和提高。使用高级数值计算可以帮助用户实现一些比较难以实现的科学计算。

本章具体包括以下内容：

- 如何使用关系运算
- 如何使用逻辑运算
- 如何进行多项式的定义和运算
- 如何进行基本的数据分析
- 如何在 MATLAB 中使用稀疏矩阵
- 如何进行基本的数值分析

在掌握了基本的矩阵运算后，就可以学习更多的数值运算功能了。MATLAB 的数值运算不仅体现在矩阵的运算上，其关系运算和逻辑运算为编制强大的应用程序提供了极大的便利，同时也是在实际工作中常要使用到的功能。随着版本的提升，MATLAB 提供了更多更加完善的关系运算和逻辑运算。

数据和数值分析是数值计算中重要的部分，也是和科研工作紧密相关的部分。MATLAB 系统提供的数据和数值分析函数已经集成到其主包中，其强大而多样的功能可以满足多方面的需求。

3.1 关系运算和逻辑运算

除了传统的数学运算，MATLAB 支持关系运算和逻辑运算。如果用户已经有些编程经验，就会熟悉这些运算。这些操作符和函数的目的是提供求解真/假命题的答案。一个重要的应用是控制基于真/假命题的一系列 MATLAB 命令(通常在 M 文件中)的流程或执行的次序。

作为所有关系和逻辑表达式的输入，MATLAB 把任何非 0 数值当作真，把 0 当作假。所有关系和逻辑表达式的输出，对于真，输出为 1；对于假，输出为 0。

3.1.1 关系操作符

MATLAB 关系操作符列于表 3.1 中，包括所有常用的比较操作。

表 3.1 关系操作符

关系操作符	功能说明	关系操作符	功能说明
<	小于	>=	大于或等于
<=	小于或等于	==	等于
>	大于	~=	不等于

MATLAB 关系操作符能用来比较 2 个同样大小的数组，或用来比较一个数组和一个标量。

1. 运算法则

- 当两个变量是标量 a 和 b 时，则：
  若 a、b 间关系成立，那么关系运算结果为 1；
  若 a、b 间关系不成立，那么关系运算结果为 0；
- 当比较量是两个维数相同的数组 A 和 B 时，则：

比较的是 A、B 数组相同位置的元素，按标量的运算规则逐个进行。关系运算的结果是一个维数和 A 相同的数组，它的元素由 0 和 1 组成。

● 当比较的一个是数组 A，一个是标量 b 时，则：

把标量 b 和数组 A 的每一个元素按标量关系运算规则逐个比较。关系运算的结果是一个维数和数组 A 相同的数组，它的元素由 0 和 1 组成。

● 优先级：由高到低为算术运算、关系运算、逻辑运算。

【例 1】关系操作符示例

```
A =
 2     3     4     5     6     7     8
B =
 6     5     4     3     2     1     0
t=A>4
t =
 0     0     0     1     1     1     1
```

说明：找出 A 中大于 4 的元素。0 出现在 A<=4 的地方，1 出现在 A>4 的地方。

```
t=(A= =B)
t=
 0     0     1     0     0     0     0
```

说明：找出 A 中的元素等于 B 中的元素。

注意：=和==意味着两种不同的事：== 比较两个变量，当它们相等时返回 1，当它们不相等时返回 0；在另一方面，= 被用来将运算的结果赋给一个变量。

```
t=B-(A>2)
t =
      6     4     3     2     1     0    -1
```

说明：找出 A>2，并从 B 中减去所求得的结果向量。这个例子说明，由于逻辑运算的输出是 1 和 0 的数组，它们也能用在数学运算中。

```
B=B+(B==0)*eps
B =
6.0000    5.0000    4.0000    3.0000    2.0000    1.0000    0.0000
```

注意：这是一个演示，表明如何用特殊的 MATLAB 数 eps 来代替在一个数组中的零元素，这种特殊的表达式在避免被 0 除时是很有用的。

【例 2】eps 的作用

```
x=(-2:2)/3
 x =
-0.6667   -0.3333         0    0.3333    0.6667
sin(x)./x   %求 sin(x)/x
Warning: Divide by zero
 ans =
    0.9276    0.9816       NaN    0.9816    0.9276
```

由于第三个数是 0，计算函数 sin(x)/x 时给出了一个警告。sin(0)/0 是没定义的，在该处 MATLAB 结果返回 NaN。用 eps 替代 0 以后，再试一次。

```
x=x+(x==0)*eps;
    sin(x)./x
    ans =
        0.8415    0.9276    0.9816    1.0000    0.9816    0.9276
```

现在 sin(x)/x 在 x=0 处给出了正确的极限。

3.1.2 逻辑操作符

逻辑操作符提供了一种组合或否定关系表达式。MATLAB 逻辑操作符如表 3.2 所示。

运算法则：

- 在逻辑运算中：
  非零元素的逻辑量为“真”，用 1 表示；
  零元素的逻辑量为“假”，用 0 表示。
- 若参与运算的是两个标量 a 和 b，那么：

表3.2 逻辑操作符

逻辑操作符	说明
&	与
|	或
~	非

a&b a，b 全是非零时，运算结果是 1；否则为 0。

a|b a，b 中只要有一个非零，运算结果为 1。

~a 当 a 是零时，运算结果为 1；否则为 0。

- 当参加逻辑运算的是 2 个维数相同的数组 A 和 B 时，则：

运算的是 A、B 数组相同位置的元素，按标量的运算规则逐个进行。逻辑运算的结果是一个维数和 A 相同的数组，它的元素由 0 和 1 组成。

- 当参加逻辑运算的一个是数组 A，一个是标量 b 时，则：

把标量 b 和数组 A 的每一个元素按逻辑关系运算规则逐个进行，逻辑运算的结果是一个维数和数组 A 相同的数组，它的元素由 0 和 1 组成。

- 逻辑“非”运算是一元运算符，服从数组运算规则。

【例 3】逻辑操作符用法

```
t=~(A>4)    % A 矩阵取自例 1
t =
1    1    1    0    0    0    0
```

说明：对上面的结果取非，也就是 1 替换 0，0 替换 1。

```
t=(A>2)&(A<6)
t =
0    1    1    1    0    0    0
```

说明：在 A 大于 2 且 A 小于 6 处返回 1。

可以利用上面的功能产生数组来表示不连续信号，或由多段其他信号所组成的信号。方法是把数组中要保持的那些值与 1 相乘，所有其他值与 0 相乘。

【例 4】表示不连续信号

```
x=linspace(0, 5, 100);          % 产生数据
y=cos(x) ;                      % 计算 cos 函数
z=(y>=0).*y ;                   % 将 cos 函数的负数置为零
z=z+0.3*(y<0) ;                 % 将 cos 函数的正数值增加 30%
z=(x<=4).*z ;                   % 将 x 超过 4 时 z 的值置为零
plot(x, z)
xlabel(' x '), ylabel(' z=f(x) '), title(' 一个不连续的信号 ')
% 绘图
```

结果如图 3.1 所示。

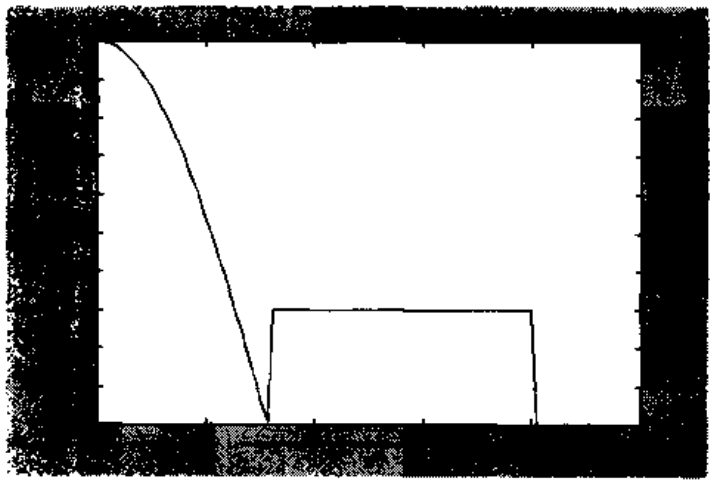


图 3.1　不连续信号

3.1.3　关系与逻辑函数

除了上面的关系与逻辑操作符外，MATLAB 还提供了大量的其他关系与逻辑函数，参见表 3.3。

表 3.3　其他关系与逻辑函数

函数名称	功能简介
xor(x, y)	异或运算。x 和 y 都是零(假)或都是非零(真)返回 0，否则返回 1
any(x)	如果在一个向量 x 中，任何元素是非零，返回 1；矩阵 x 中的每一列有非零元素，返回 1
all(x)	如果在一个向量 x 中，所有元素非零，返回 1；矩阵 x 中的每一列所有元素非零，返回 1

除了这些函数，MATLAB 还提供了大量的函数，测试特殊值或条件的存在，返回逻

辑值。测试函数见表 3.4 所示。

表 3.4 测 试 函 数

函数名称	功能简介
find	找出向量或矩阵中非零元素的位置和标识
finite	元素有限，返回真值
isempty	参量为空，返回真值
isglobal	参量是一个全局变量，返回真值
ishold	当前绘图保持状态是‘ON’，返回真值
isieee	计算机执行 IEEE 算术运算，返回真值
isinf	元素无穷大，返回真值
isletter	元素为字母，返回真值
isnan	元素为不定值，返回真值
isreal	参量无虚部，返回真值
isspace	元素为空格字符，返回真值
isstr	参量为一个字符串，返回真值
isstudent	MATLAB 为学生版，返回真值
isunix	计算机为 UNIX 系统，返回真值
isvms	计算机为 VMS 系统，返回真值
exist	检查变量、函数、文件的存在性及类型

这些指令在 MATLAB 程序设计和直接交互式运算中非常有用。用户可以从用户指南和联机帮助中找到具体的调用格式。本书不再作具体介绍。

3.1.4 NaN 和空矩阵

NaN 和空矩阵“[ ]”在 MATLAB 中做特殊处理，特别是用于逻辑或关系表达式中。根据 IEEE 数学标准，对 NaN 的几乎所有运算都得出 NaN。

【例 5】NaN 示例

```
    A=[5  3  nan  inf  nan]   % 注意实际中，NaN 可以写成小写字母
A =
     5     3   NaN   Inf   NaN
B=2*A  %乘法运算
B =
    10     6   NaN   Inf   NaN
C=sqrt(A)  %取平方根值
C =
```

```
2.2361    1.7321       NaN       Inf       NaN
D= (A==nan)
D =
 0     0     0     0     0
F = (A~=nan)
F =
 1     1     1     1     1
```

说明：

上面的第一和第二计算式对 NaN 输入给出 NaN 结果。然而，最后 2 个计算式产生有点令人惊讶的结果。当 NaN 与 NaN 相比较时，(a= =nan)产生全部为 0 或假的结果，同时(a~=nan)产生全部 1 或真值。于是，单个 NaN 相互不相等。由于 NaN 的这种特性，MATLAB 有一个包含逻辑函数寻找 NaN。

```
G=isnan(A)
G =
 0     0     1     0     1
```

这个函数用 find 命令能找出 NaN 的下标值。

```
H=find(isnan(A))       %  找出 NaN 的下标值
H =
     3     5
A(H)=zeros(size(H))    %  将 NaN 转变为零
A =
 5     3     0   Inf     0
```

空矩阵“[ ]”是 MATLAB 系统中定义大小为 0 的变量。当没有其他合适的结果时，在 MATLAB 中的许多函数返回空矩阵。

【例 6】函数 find 产生的空阵

```
x=(2:5)-3     % 产生新的数据
x =
-1     0     1     2
y=find(x>2)
y =
   [ ]
```

说明：x 没有包含大于 2 的值，所以没有返回下标。为了测试空结果，MATLAB 提供了逻辑函数 isempty。

```
isempty(y)
ans =
     1
```

注意： 在 MATLAB 里，空矩阵不等于任何非零矩阵(或标量)。尽管空阵大小为零，但是有一个下标值。

3.2 多 项 式

多项式在很多学科的计算中有着重要的应用。众多的方程和定理都是多项式的形式。MATLAB 系统提供多项式操作工具，包括多项式的求根、分解、求导数以及多项式的拟合。

3.2.1 多项式的表达和求根

在 MATLAB 里，多项式由一个行向量表示，它的系数是按降序排列。

【例 7】输入多项式 $x^4-17x^3+34x^2+0x+16$

```
p=[1  -17  34  0  16]
p =
        1   -17    34    0  16
```

注意： 行向量中，必须包括具有零系数的项。因为除非特别地辨认，MATLAB 无法知道哪一项为零。

除了可以直接输入行向量，还可以利用指令 P=poly(AR)，产生多项式系数向量。

- 若 AR 是方阵，则多项式为方阵 AR 的特征多项式；
- 若 AR 是行向量，$AR=[a_{r1}a_{r2}\Lambda\ a_m]$，则所得的多项式满足关系式：

$$P=[a_0a_1\Lambda\ a_{n-1}a_{n-2}]$$

$$(x-a_{r1})(x-a_{r2})\Lambda\ (x-a_m)=a_0x^n+a_1x^{n-1}+\Lambda+a_{n-1}x+a_n$$

【例 8】求方阵 A 的特征多项式

```
A=[3, 5, 7;9, 3, 5;11, 14, 8];
PA=poly(A)
PA =
   1.0000  -14.0000 -135.0000 -428.0000
```

说明： n 阶方阵的特征多项式系数向量一定是(n+1)维的，第一个元素的值一定是 1。

若令多项式的值为零，就可以把多项式看成一元 n 次方程。MATLAB 系统用函数 roots 找出一个多项式的根。

【例 9】求解例 1 中多项式的根

```
r=roots(p)
r =
 14.6787
  2.5221
 -0.1004 + 0.6497i
 -0.1004 - 0.6497i
```

说明： MATLAB 在求多项式根时，并不是按经典方法直接对多项式进行计算，而是先把多项式转换为伴随阵，然后再求特征值，其可靠性和精度都高于经典方法。在 MATLAB

中，无论是一个多项式，还是它的根，都是向量。MATLAB 按惯例规定，多项式是行向量，根是列向量。

【例 10】由给定的根求多项式的系数行向量

```
pp=poly(r)          % 利用上例中求出的根
pp =
    1.0000  -17.0000   34.0000    0.0000+0.0000i   16.0000
pp=real(pp)          %考虑到多余的虚数部分
pp =
    1.0000  -17.0000   34.0000    0.0000   16.0000
```

3.2.2　多项式的运算

表 3.5 列出多项式运算有关的函数。

表 3.5　多项式函数

函数名称	功能简介
conv(a，b)	乘法
[q，r]=deconv(a，b)	除法
poly(r)	用根构造多项式
polyder(a)	对多项式或有理多项式求导
polyfit(x，y，n)	多项式数据拟合
polyval(p，x)	计算 x 点处多项式值
[r，p，k]=residue(a，b)	部分分式展开式
[a，b]=residue(r，p，k)	部分分式组合
roots(a)	求多项式的根

1.　乘法

多项式的乘法实际上和卷积(Convolution)的机理完全相同。长度为 m 的向量 a 和长度为 n 的向量 b 的卷积(Convolution)定义为：

$$c(k)=\sum_{i=1}^{k}a(i)b(k+1-i)$$

得出的结果 c 向量的长度为(m+n−1)。

MATLAB 系统中用函数 c=conv(a，b)支持多项式乘法(执行两个数组的卷积)。

【例 11】两个多项式乘积

考虑两个多项式 $a(x)=x^3+2x^2+3x+4$ 和 $b(x)=x^3+4x^2+9x+16$ 的乘积。

```
a=[1  2  3  4] ;  b=[1  4  9  16];
c=conv(a ,  b)
```

```
c =
     1     6    20    50    75    84    64
```

结果是 $c(x)=x^6+6x^5+20x^4+50x^3+75x^2+84x+64$。两个以上的多项式的乘法需要重复使用 conv。

2. 加法(减法)

对多项式加法(减法)，MATLAB 不提供一个直接的函数。如果 2 个多项式向量大小相同，标准的数组加法有效。把多项式 a(x)与上面给出的 b(x)相加。

【例 12】数组的加法

```
d=a+b   % a，b 取自例 11
    d =
        2     6    12    20
```

说明：

结果是 $d(x)= 2x^3+6x^2+12x+20$。若两个多项式阶次不同，低阶的多项式必须用首零填补，使其与高阶多项式有同样的阶次。考虑上面多项式 c 和 d 相加：

```
e=c+[0  0  0  d]
e =
    1     6    20    52    81    96    84
```

说明：

结果是 $e(x)= x^6+6x^5+20x^4+52x^3+81x^2+96x+84$。要求首零而不是尾零，是因为相关的系数像 x 幂次一样，必须整齐。

3. 除法

在一些特殊情况，一个多项式需要除以另一个多项式。MATLAB 利用卷积的逆运算解卷(Deconvolution)来进行除法运算。用向量 a 对向量 c 进行解卷将得到“商(Quotient)”向量 q 和“余(Remainder)”向量 r，并且满足：

$$c(k)-r(k)=\sum_{i=1}^{k}a(j)q(k+1-i)$$

在 MATLAB 中，这由函数 deconv(c，a)完成。具体的调用格式如下：

[q，r]=deconv(c，a)　　q，r 分别表示商多项式和余多项式
　　c，a 分别表示被除多项式和除多项式

【例 13】两个多项式相除

```
[q ,  r]=deconv(c ,  b)   % c，b 取自上例
q =
 1     2     3     4
r =
 0     0     0     0     0     0     0
```

说明：这个结果是 c 除以 b，给出商多项式 q 和余数 r。在现在情况下 r 是零，因为 b 和 q 的乘积恰好是 c。

4. 求导数

MATLAB 为多项式 A(n 次)求导提供了函数 polyder(A)，返回的是个长度为 n－1 的向量。

【例 14】　求多项式的导数

```
g =[3  8  30  88  61  62  4]
h=polyder(g)
h =
18    40   120   264   122    62
```

5. 估值

在 MATLAB 中，有 2 种估值运算：

- 按数组运算规则计算多项式的值

PA=polyval(p，S)　　p 为多项式，S 为矩阵，矩阵 S 中的每一个值代入多项式求出值，结果是和 S 同维的矩阵。

- 按矩阵运算规则计算多项式的值

PAM=polyvalm(p，S)　p 为多项式，S 为矩阵，矩阵 S 作为变量代入多项式求出值，结果是和 S 同维的矩阵。

【例 15】多项式计算

```
s=rand(3);
p=poly(s)
pa=polyval(p, s)
pm=polyvalm(p, s)
p =
    1.0000   -2.6628    1.9560   -0.4289
pa =
   -0.1166    0.0075    0.0042
   -0.1067   -0.0929   -0.3936
    0.0009   -0.0422   -0.0647
pm =
  1.0e-015 *
   -0.8327   -0.7772   -0.6106
   -0.2836   -0.1665   -0.1457
   -0.9437   -0.8882   -0.6661
```

说明：

pm 中的元素都很小，它是运算误差造成的。从理论上讲，它应该是零矩阵。因为任何一个矩阵满足它自己的特征多项式方程。

3.2.3 有理多项式

在许多应用中，例如傅立叶(Fourier)、拉普拉斯(Laplace)和 Z 变换中，出现有理多项式或两个多项式之比。在 MATLAB 系统中，有理多项式由它们的分子多项式和分母多项式表示。对有理多项式进行运算的 2 个函数是 residue 和 polyder。

1. 有理多项式的分式展开

MATLAB 中提供函数 residue 执行部分分式展开。具体形式为：

[r，p，k]=residue(b，a) 或者相应的逆运算 [b，a]=residue(r，p，k)。

b，a 分别是分子、分母多项式系数向量；r，p，k 分别是留数、极点和直项向量。

【例 16】部分分式展开

```
num=10*[1  4  5  6  7];          %  分子多项式
den=poly([-2 ;  -1;  -0.5 ]) ;          %  分母多项式
[res,  poles,  k]=residue(num,  den)
res =
   -6.6667
  -60.0000
   64.1667
poles =
   -2.0000
   -1.0000
   -0.5000
k =
10   5
```

上面的结果说明了这个问题：

$$\frac{10(s^4+4s^3+5s^2+6s+7)}{(s+2)(s+1)(s+0.5)}=\frac{-6.6667}{s+2}+\frac{-60.0000}{s+1}+\frac{64.16667}{s+0.5}+10s+5$$

这个函数也可执行逆运算。

```
[n,  d]=residue(res,  poles,  k)  % n 表示分子多项式，d 表示分母多项式
n =
   10.0000   40.0000   50.0000   60.0000   70.0000
d =
   1.0000    3.5000    3.5000    1.0000
roots(d)
ans =
   -2.0000
   -1.0000
```

```
    -0.5000
```

说明：

这与开始时的分子和分母多项式一致。当然 residue 也能处理重极点的情况，尽管这里没有考虑。

2. 有理多项式求导

如前所述，函数 polyder 对多项式求导。除此之外，如果给出两个输入，则它对有理多项式求导。

【例 17】有理多项式求导

```
[b , a]=polyder(num , den) % num 分子多项式，den 分母多项式
b =
    10    70   195   200  -125  -390  -185
a =
1.0000    7.0000   19.2500   26.5000   19.2500    7.0000    1.0000
```

该结果证实：

$$\frac{d}{ds}\left\{\frac{10(s^4+4s^3+5s^2+6s+7)}{(s+2)(s+1)(s+0.5)}\right\}=\frac{10s^6+70s^5+195s^4+200s^3-125s^2-390s-185}{s^6+7s^5+19.25s^4+26.5s^3+19.25s^2+7s+1}$$

3.2.4　多项式拟合

在大量的数据处理时，数据的函数拟合是一项很重要的工作。MATLAB 系统提供数据的多项式拟合函数 polyfit，具体调用形式如下：

```
P=polyfit(x, y, n)          用 n 阶多项式拟合 x，y 向量给定的数据
```

当然如果事先知道所要的拟合函数是多项式，那么拟合的精度会比较高。但是在一般情况下，所需拟合的数据并不满足简单的多项式的函数形式，因此不能将拟合的函数随意扩展到更大的区间上去。下面的例子就证明了这一点。

【例 18】多项式拟合及其有效区间

```
x=0:0.1:2.5;
y=erf(x);
P=polyfit(x, y, 6)
x=0:0.1:5;
y=erf(x);
f=polyval(P, x);
plot(x, y, 'bo', x, f, 'r-');
axis([0, 5, 0, 2]);
legend('拟合曲线', '原始数据')
P =
0.0084   -0.0983    0.4217   -0.7435    0.1471    1.1064    0.0004
```

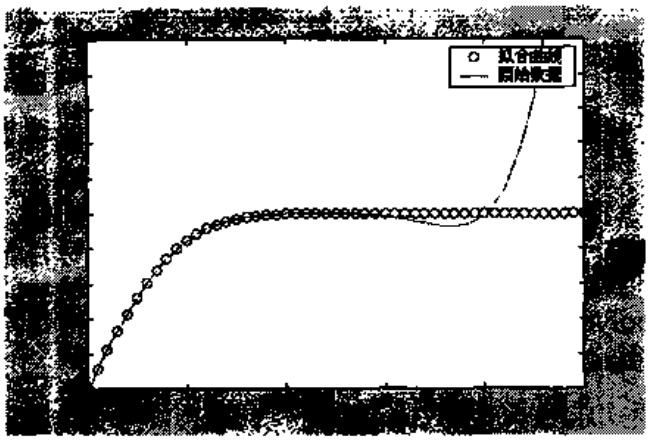


图 3.2 有效拟合的区间性

提示： 一般来说，多项式拟合中阶数 n 取得越大，拟合的精度就越高。用户可以利用上例，增大 n 看看拟合的效果是否有所提高。

3.3 数据分析函数

由于 MATLAB 面向矩阵，所以它很容易对数据集合进行统计分析。按规定，数据集存储在面向列的矩阵里。也就是，一个矩阵的每一列代表不同的被测变量，每一行代表各个样本或观察值。

3.3.1 基本数据分析指令

在 MATLAB 里的数据分析是按面向列矩阵而进行的。不同的变量存储在各列中，而每行表示每个变量的不同观察。MATLAB 数据分析函数见表 3.6。

表 3.6 数据分析函数

函数名称	功能简介
corrcoef(x)	求相关系数
cov(x)	协方差矩阵
cplxpair(x)	把向量分类为复共轭对
cross(x， y)	向量的向量积
cumprod(x)	列累计积
cumsum(x)	列累计和
del2(A)	五点离散拉氏算子
diff(x)	计算元素之间差
dot(x， y)	向量的点积
gradient(Z， dx， dy)	近似梯度

续表3.6

函数名称	功能简介
histogram(x)	直方图和棒图
max(x)，　max(x，　y)	最大分量
mean(x)	均值或列的平均值
median(x)	列的中值
min(x)，　min(x，　y)	最小分量
prod(x)	列元素的积
rand(x)	均匀分布随机数
randn(x)	正态分布随机数
sort(x)	按升序排列
std(x)	列的标准偏差
subspace(A，　B)	两个子空间之间的夹角
sum(x)	各列的元素和

提示： 数据分析是按面向列矩阵进行的这一约定，不仅应用于本节中的指令，而且反映在 MATLAB 工具箱中。因此，用户在编制自己的程序时，请尽量遵循以上的约定，以便更好地利用 MATLAB 现有的函数和命令。

【例 19】基本统计函数

```
A=rand(10，2);  %产生 10 行 2 列的矩阵 A
AMAX=max(A) %求矩阵 A 各列的最大值
AAMAX=max(AMAX) %求矩阵 A 中的最大值
AMED=median(A)  %求矩阵 A 各列的中位元素
AMEAN=mean(A)   %求矩阵 A 各列元素的平均值
ASTD=std(A) %求矩阵 A 各列元素的标准差
AMAX =
    0.8998    0.8385
AAMAX =
    0.8998
AMED =
    0.5895    0.5947
AMEAN =
    0.5848    0.5823
ASTD =
    0.2198    0.1683
```

3.3.2 协方差矩阵和相关阵

协方差矩阵及其相关阵在概率和数理统计中有很重要的应用，MATLAB 系统提供如下的函数实现计算协方差矩阵及其相关阵：

- C=cov(X) 返回一个协方差矩阵，X 的每行看作不同的变量值
- C=ding(cov(X)) 每行变量的向量
- C=sqrt(ding(cov(X))) 标准协方差
- C=cov(X，Y) X，Y 是同维的向量，等效于 cov([X(:) Y(:)])
- P=corrcoef(X) 返回一个相关阵，X 的每行看作不同的变量值
- P=corrcoef(X，Y) X，Y 是同维的列向量等效于 corrcoef([X Y])

说明：

若 C 是协方差矩阵，C = cov(X)，它的每一个元素为 C(i，j)，那么矩阵 p=corrcoef(X) 的 (i，j)元素为：

$$p(i,j)=\frac{C(i,j)}{\sqrt{C(i,j)}\sqrt{C(j,i)}}$$

【例 20】协方差阵和相关阵

```
x=rand(5, 2);
y=rand(5, 2);
CX=cov(x)
CY=cov(y)
CXY=cov(x, y)
PX=corrcoef(x)
PXY=corrcoef(x, y)
CX =
    0.1079    0.0495
    0.0495    0.0307
CY =
    0.1211   -0.0036
   -0.0036    0.0599
CXY =
    0.0617   -0.0047
   -0.0047    0.0805
PX =
    1.0000    0.8590
    0.8590    1.0000
PXY =
    1.0000   -0.0665
   -0.0665    1.0000
```

3.3.3　统计频数函数

对于随机样本$\{y_1,y_2\cdots y_n\}$构成的向量 Y，记 $y_{max}=\max\{Y\}$，$y_{min}=\min\{Y\}$。令 $L=y_{max}-y_{min}$，并把 L 分成 N 段，即 $l=\frac{L}{N}$；在把落在区间中的随机样本数记为 K_i；该区间的中心值记为 $x_i=\left(y_{min}+\frac{2i-1}{2}l\right)$。于是获得构成统计频数函数的两个统计向量 $K=[k_1,k_2,\Lambda,k_3]$ 和 $x=[x_1,x_2,\Lambda,x_3]$。

MATLAB 系统提供两组命令实现这种计算功能：

- [K，X] = hist(Y，N)　在 N 个子区间上计算 Y 直方频数函数
- hist(Y，N)　用直方图表现在 N 个子区间上的频数函数
- [TK，RK]=rose(THETA，N)　在 N 个子扇形上计算 THETA 的扇形频数函数
- rose(THETA，N) 用扇形图表示在 N 个子扇形上的频数函数

【例 21】统计频数函数

```
y=randn(10000, 1);          %
产生正态分步的随机向量 y
subplot(1, 2, 1)
hist(y, 50)                 %
直方图
subplot(1, 2, 2)
rose(y, 50)
%扇形图
```

输出结果如图 3.3 所示。

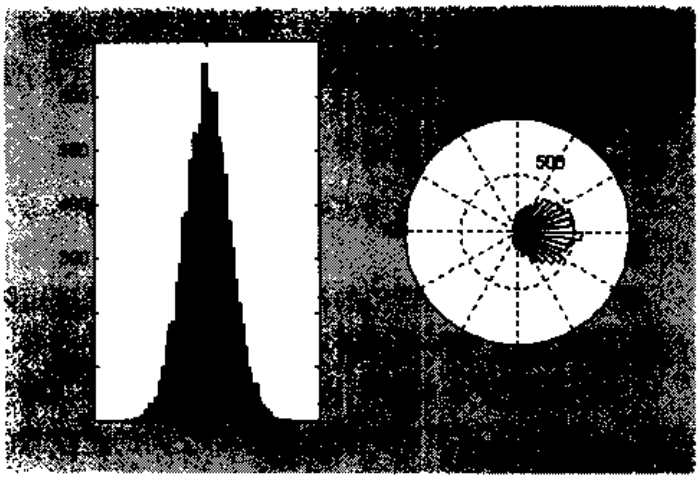


图 3.3　统计频数直方图和扇形图

3.4　稀 疏 矩 阵

对于一个用矩阵描写的线性方程来说，n 个未知数的问题就涉及一个 n×n 的方程组。解这个方程就需要 n^2 个数字的内存和正比于 n^3 的计算时间。这样解一个上千阶的方程就很不容易处理(即使在这样一个计算机技术不断提高的年代)。但是在大多数情况下，一个未知数只和数量不多的其他变量有关，即关系矩阵是稀疏矩阵。

稀疏矩阵及其算法不存储那些 0 元素，也不对它们进行操作，从而节省了内存和时间的开销。稀疏矩阵的计算复杂性和代价仅取决于稀疏矩阵的非零元素数目，而与该矩阵的总元素个数无关。

3.4.1 稀疏矩阵的创建和存储

1. 矩阵的存储方式

在 MATLAB 中有 2 种存储矩阵的方式：一种是全元素(Full)存储；另一种是稀疏(Sparse)存储。稀疏存储有以下特点：

- 只存储矩阵的非零元素，并且存储按列进行。
- 对于每列，用一个实数(或复数)数组记述非零元素值，再用一个整数数组记述相应非零元素的行下标。

实现 2 种存储方式转换的指令如下：

SM=sparse(M)把矩阵存储方式从任何一种形式(含全元素形式)转变为稀疏形式；

FM=full(A) 把矩阵存储方式从任何一种形式(含稀疏形式)转变为全元素形式。

观察这两种存储方式的差别可以用 whos 指令实现。

全元素矩阵经运算后仍然是全元素存储。因此，假如不经过专门定义和运算，稀疏矩阵是不会自动产生的。但是，一旦某个矩阵以稀疏方式存储，那由它参与运算的结果将仍以稀疏方式存储，除非运算本身(如经多次加运算)使稀疏性消失。

2. 稀疏矩阵创建指令：sparse

SM＝sparse(I，J，S，m，n，nzmax)

说明：

- 用矩阵[I，J，S]的行创建(m×n)维稀疏矩阵 SM。
- S 是按列排列的所有非零元素构成的向量。I，J 分别是非零元素的行下标和列下标。这 3 个向量的长度相同。m,n 分别是待生成稀疏矩阵 SM 的行与列维数。nzmax 是用来为非零元素指定存储空间的正整数(它一定不小于非零元素总数)。
- 调用格式中的 6 个输入量在一定条件下可以默认。详见 MATLAB 系统的用户指南或联机帮助。

3. 稀疏带状矩阵创建指令：spdiags

该函数的调用格式为：

SM =spdiags(B，d，m，n)

说明：

- m，n 分别指定矩阵 SM 的行与列维数。d 是长度为 p 的整数向量，它指定 SM 的对角线位置。B 是全元素阵(但不必一定)，用来给定 SM 的对角线位置上的元素，其行维为 min(m，n)，列维为 p。
- 该指令的其他调用格式请看用户指南或联机帮助。
- MATLAB 系统还提供了其他几种特殊稀疏矩阵的创建指令：稀疏单位阵(speye)、稀疏随机阵(sprandn)和稀疏对称随机阵(spandsym)。详见用户指南或联机帮助。

4. 外部数据转换为稀疏矩阵的指令：spconvert

该函数的调用格式：

SM = spconvert(T)

说明：

- T 来自外部数据文件(可以是 mat 文件，也可以是 dat 文件)。数据文件由 load 命令装载于 MATLAB 内存空间。
- T 数组的行维为 nnz(即非零元素数)或 nnz+1，列维为 3(对实数而言)或 4(对复数而言)。T 数组的每一行(以[I，J，Sre，Siml]形式)指定一个稀疏矩阵元素。
- load，save 指令本身不区分矩阵的存储形式，都能正确操作。

【例 22】用 2 种不同方式创建三对角稀疏矩阵

```
n=4;
SM1=sparse(1:n, 1:n, -2*ones(1, n), n, n, n);
SM2=sparse(2:n, 1:n-1, ones(1, n-1), n, n, n-1);
S1=SM1+SM2+SM2'
e=ones(n, 1);
S2=spdiags([e, -2*e, e], [-1, 0, 1], n, n)
SF=full(S1)
S1 =
   (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
S2 =
   (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
SF =
    -2     1     0     0
     1    -2     1     0
     0     1    -2     1
     0     0     1    -2
```

3.4.2 稀疏矩阵的运算

1. 运算规则

MATLAB 系统的各种命令都可以运用于稀疏矩阵的运算。当有稀疏矩阵参加运算时，所得的结果将遵循以下的规则：

- 把矩阵变为标量或者定长向量的函数总是给出全元素结果。
- 把标量或者定长的向量变换到矩阵的函数(diag、eyes、ones 等)总是给出全元素结果；而能给出稀疏矩阵结果的相应命令有 spdiags、speye、sprandn、sparse。
- 从矩阵到矩阵或向量的“单矩阵”变换函数将以原矩阵(是稀疏还是全元素)形式给出结果，如 chol(S)、diag(S)、max(S)、sum(S)和～S 等。
- 两个矩阵运算(如+、－、*、\、|)操作后的结果一般是全元素形式，除非参加运算的两个矩阵都是稀疏的，或除非操作本身(如.*、&)保留稀疏性。
- 参与的矩阵扩展(如[AB;CD])的子矩阵中，只要有一个是稀疏的，那么所得的结果也是稀疏的。
- 在矩阵引用中，将仍以原矩阵(是稀疏还是全元素)形式给出结果。若 S 阵稀疏而 Y 阵是全元素形式，不管 I、J 是标量还是向量，那么“右引用”Y=S(I，J)产生稀疏矩阵而“左引用”Y(I，J)=S 产生全元素结果。

2. 常用指令及应用举例

常用的稀疏矩阵指令见表 3.7。

表 3.7 常用稀疏矩阵指令

函数名称	功能简介
issparse	当矩阵稀疏时给出 1，否则为 0
nnz	矩阵的非零元素总数
nonzeroc	矩阵的非零元素数值
nzmax	指定存放非零元素所需内存
spalloc	为非零元素配置内存
sprun	求各非零元素的函数值
spones	用 1 置换非零元素

续表3.7

函数名称	功能简介
colmmd	列最小度排序
colperm	计算列排序置换向量
dmperm	矩阵的 DulTnageMendelsohn 分解
randperm	随机置换向量
symmmd	对称最小度排序
symrcm	反向 Cuthill・McKee 排序
condest	估计 1—范数条件数
normest	估计 2—范数
sprank	结构秩
gplot	依图论法则画“(无向)图”
spy	画稀疏结构图

说明：关于上述指令的详细解释请查用户指南或联机帮助。

【例 23】全元素矩阵、稀疏矩阵、最小排序稀疏矩阵三角分解所需时间的比较

```
n=200;                                    %给出矩阵的阶数
A=sprandsym(n, 0.03)+100*speye(n, n)      %建立(100×100)随机正定稀疏矩阵
subplot(1, 2, 1)
spy(A, 'b', 10)                           %画稀疏矩阵结构图
title('矩阵 A 的结构图')
subplot(1, 2, 2)
d=symmmd(A)                               %采用最小度排序法
spy(A(d, d), 'b', 10)
title('最小度排序算法')
B=full(A)                                 %给出 A 的全元素形式
format long e
tic, L1=chol(B);t1=toc                    %全元素时，cholesky 分解的计算时间
tic, L2=chol(A);t2=toc                    %稀疏矩阵时，cholesky 分解的计算时间
tic, L3=chol(A(d, d));t3=toc              %最小度排序时，cholesky 分解的计算时间
t1 =
   3.799999999999990e-001
t2 =
   1.600000000000001e-001
t3 =
   5.000000000000071e-002
```

作出的图如 3.4 所示。

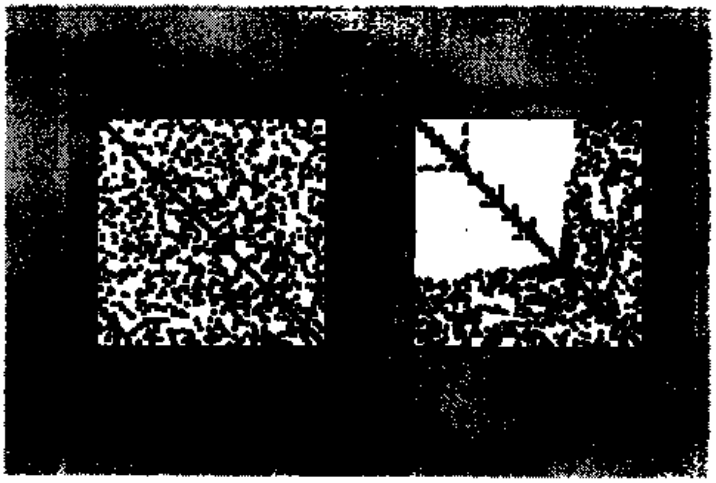


图 3.4 稀疏矩阵结构

说明：

指令 symmmd 采用最小度排序法。其中心思想是根据消元的每一个阶段和选择可能的主元，获得填充项和运算次数的最小化，因而列高斯消元法能够与最小度算法较好地结合，其运算时间最短。

用 who 指令查看各变量的如下储存信息，进一步说明稀疏矩阵节省内存的作用。

Name	Size	Bytes	Class
A	200×200	17448	sparse array
B	200×200	320000	double array
L1	200×200	320000	double array
L2	200×200	111960	sparse array
L3	200×200	44676	sparse array

3.5 数 值 分 析

当难以对一个函数进行积分、微分或者解析上确定一些特殊的值时，就可以借助计算机在数值上近似所需的结果。这在计算机科学和数学领域，称之为数值分析。MATLAB 提供了解决这些问题的工具。本节将介绍这些工具的使用。

3.5.1 求极小值

在许多应用中，特别感兴趣的是确定函数的极值，即最大值(峰值)和最小值(谷值)。数学上，可通过确定函数导数(斜率)为零的点，解析上求出这些极值点。显然，如果定义的函数简单，则这种方法常常奏效。然而，即使很多容易求导的函数，也常常很难找到导数为零的点。在这种情况下，以及很难或不可能解析上求得导数的情况下，必须数值上寻找函数的极值点。MATLAB 提供了 2 个完成此功能的函数 fminbnd 和 fminsearch。这 2 个

函数分别寻找一维或 n 维函数的最小值。因为 f(x)的最大值等于－f(x)的最小值，所以，上述 fminbnd 和 fminsearch 可用来求最大值和最小值。

注意：　在 MATLAB 的早期版本中没有这 2 个函数，而对应的分别为 fmin 和 fmins。如果用户熟悉这 2 个函数，请注意，MATLAB 今后会逐渐取消这 2 个函数。

具体调用格式为：

fminbnd 单变量非线性函数最小值函数。

- x =fminbnd(FUN，$x1$，$x2$)

返回的数值 x 是 FUN 函数(在 M 文件中定义的)在区间 $x1 < x < x2$ 上的局部最小值点。FUN 函数必须是单值非线性函数。

- x = fminbnd(FUN，$x1$，$x2$，options)
- x = fminbnd(FUN，$x1$，$x2$，options，p1，p2，...)

options 是用来控制算法的参数向量，其中有用的几个元素的意义是：

options(1)　默认值为 0，若被赋予 1，则显示运算中间步骤

options(2)　寻优的容差值，默认值为 0.0001

options(3)　终止寻优的目标函数容差值，默认值为 0.0001

options(14)　允许的最大迭代的次数，默认值为 500

p1，p2，...　向目标函数传递的参数，允许默认

fminsearch 多变量无束缚非线性最小值函数：

x =fminsearch(FUN，$x0$，options，p1，p2，...)

fminsearch 采用著名的 Nelder-Mead 单纯形算法求解多变量函数的最小值。$x0$ 是最小值点的初始猜测值。其余各项的含义和 fminbnd 一样。

【例 24】求解函数的最小值

```
fn='2*exp(-x)*sin(x)';  %  定义 fn 函数
      xmin=fminbnd(fn ,  2 ,  5)   %  在区间 2<x<5 内寻找最小值
      emin=5*pi / 4-xmin           %  求最小值的误差
      x=xmin;                      %  令 x 为最小值点
      ymin=eval(fn)                %  计算最小值点的函数值
      fx=' -2*exp(-x)*sin(x) ';  %  为求最大值，函数值取负
      xmax=fminbnd(fx ,  0 ,  3)   %  在区间 0<x<3 寻找最大值
      emax=pi / 4-xmax             %  求最大值的误差
      x=xmax;                      %  令 x 为最大值点
      ymax=eval(fn)                %  求最大值点的函数值
      Optimization terminated successfully:
       the current x satisfies the termination criteria using OPTIONS.TolX
  of 1.000000e-004                 %MATLAB 系统给出的正确结束提示信息
      xmin =
         3.9270
      emin =
```

```
    1.4523e-006
ymin =
   -0.0279
Optimization terminated successfully:
 the current x satisfies the termination criteria using OPTIONS.TolX
of 1.000000e-004              %MATLAB 系统给出的正确结束提示信息
xmax =
    0.7854
emax =
   -1.3781e-005
ymax =
    0.6448
```

说明：

本例中使用了函数 eval，它获取一个字符串并解释它，如同在 MATLAB 提示符下输入该字符串。由于要计算的函数以 x 为自变量的字符串形式给出，那么设置 x 等于 xmin 和 xmax，允许 eval 计算该函数，找到 ymin 和 ymax。

注意： 求数值上的最小值包含一个搜索过程，fmin 不断计算函数值，寻求其最小值。如果计算的函数需要很大的计算量，或者该函数在搜索区间不止一个最小值，则该搜索过程所花的时间比较长。在有些情况下，搜索过程根本找不到结果。当 fmin 找不到最小值时，它会停止运行并提供解释。

【例 25】多维函数求最小值

```
f = inline('norm(x)')          %norm(x)是 MATLAB 系统中内置的函数
x = fminsearch(f, [1;2;3]);    %从点[1, 2, 3]开始搜索
x =
  1.0e-004 *
   -0.4807
   -0.1109
   -0.1866
Optimization terminated successfully:
 the current x satisfies the termination criteria using OPTIONS.TolX
of 1.000000e-004
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun
of 1.000000e-004 %MATLAB 系统提供的正确结束信息
```

说明：

函数 fminsearch 利用单纯形法求最小值，它不需要精确的梯度计算。任何一种优化工具箱中具有更多扩展的优化算法。

3.5.2　求零点

正如人们对寻找函数的极点感兴趣一样，有时寻找函数过零或等于其他常数的点也非常重要。一般试图用解析的方法寻找这类点非常困难，而且很多时候是不可能的。MATLAB 再一次提供了该问题的数值解法。函数 fzero 寻找一维函数的零点。

```
z=fzero('fname', x0, tol, trace)
```

fname 是待求零点的函数文件名。x0 有两个作用：一是预定待搜索零点的大致位置；二是作为搜索起始点。tol 控制结果的相对精度，可以默认，默认值为 eps。trace 指定迭代信息是否在运算中显示，可以默认，默认为 0，表示不显示迭代信息。

注意： 该指令能比求多项式根的指令求更一般的单变量函数的零点。一个函数可能会有多个零点，但本指令的结果 z 只给出离 x0 最近的那个零点。

【例 26】求函数 humps 的零点

```
xzero=fzero('humps ',  1.2)% 在 1.2 附近寻找零点
xzero=
    1.2995
yzero=humps(xzero ,  1.2)          %在所求零点计算函数值
yzero=
   3.5527e-15
```

说明：寻找零点的过程可能失败。如果 fzero 没有找到零点，它将停止运行并提供解释。当调用函数 fzero 时，必须给出该函数的名称。但由于某种原因，它不能接受以 x 为自变量的字符串来描述的函数。

提示： fzero 不仅能寻找零点，它还可以寻找函数等于任何常数值的点，仅仅要求一个简单的再定义。例如，为了寻找 f(x)=c 的点，定义函数 g(x)=f(x)—c，然后，在 fzero 中使用 g(x)，就会找出 g(x)为零的 x 值，它发生在 f(x)=c 时。

3.5.3　数值积分

一个函数的积分或面积也是它的另一个有用的属性。MATLAB 提供了在有限区间内，数值计算某函数下的面积的 3 种函数：trapz、quad 和 quad8。

函数 trapz 通过计算若干梯形面积的和来近似某函数的积分：

z = trapz(x，y)　　计算变量 x 函数值 y 的积分

【例 27】用 trapz 在区间-1<x<2 上计算 y=humps(x)下面的面积

```
x=-1 : 0.1: 2;              % 粗略计算积分产生 x 的序列
y=humps(x);
area=trapz(x ,  y)          % 计算积分
area =
   26.4601
x=-1 : 0.01: 2;             % 以更好的精度计算
```

```
y=humps(x);
area=trapz(x , y)
area =
   26.3449
```

说明：粗略近似可能低估了实际面积。除非特别精确，没有准则说明哪种近似效果更好。很明显，如果人们能够以某种方式改变单个梯形的宽度，以适应函数的特性，即当函数变化快时使梯形的宽度变窄，这样就能够得到更精确的结果。

MATLAB 的函数 quad 和 quad8 是基于数学上的正方形概念来计算函数的面积。为获得更准确的结果，这 2 个函数在所需的区间都要计算被积函数。此外，与简单的梯形比较，这 2 个函数进行更高阶的近似，而且 quad8 比 quad 更精确。具体调用格式如下：

```
S=quad('fname', a, b, tol, trace, p1, p2, ...)
S=quad8('fname', a, b, tol, trace, p1, p2, ...)
```

fname 是被积函数表达式字符串或函数文件名。a、b 分别是积分的上、下限。tol 用来控制积分精度，默认时，tol=0.001。trace 取 1，用图形展现积分过程；取 0 则无图形；默认时不画图。p1，p2...是向函数传递的参数，可以默认。

【例 28】用 quad 和 quad8 在区间－1<x<2 上计算 y=humps(x)下面的面积

```
format long
area=quad('humps' , -1 , 2)
area =
  26.34497558341242
area=quad8('humps' , -1 , 2)
area =
 26.34496024631924
```

注意：这 2 个函数返回非常相近的估计面积，而且这个估计值在 2 个 trapz 面积的估计值之间。有关 MATLAB 的积分函数的其他信息参见联机帮助。

3.5.4 数值微分

与积分相反，数值微分非常困难。积分描述了一个函数的整体或宏观性质，而微分则描述一个函数在一点处的斜率，这是函数的微观性质。因此积分对函数的形状在小范围内的改变不敏感。而微分却很敏感。一个函数微小的变化，容易产生相邻点的斜率的大改变。

由于微分这个固有的困难，所以尽可能避免数值微分，特别是对实验获得的数据进行微分。在这种情况下，最好用最小二乘曲线拟合这种数据，然后对所得到的多项式进行微分。或用另一种方法，对该数据进行三次样条拟合，然后寻找样条微分。

【例 29】运用多项式微分函数 polyder 求数值微分

```
x=0:0.1:1;
y=[-.447  1.978  3.28  6.16  7.08  7.34  7.66  9.56  9.48  9.30
```

```
11.2];
    %试验数据
    n=2;                    %设定拟合的阶数
    p=polyfit(x , y , n)    %多项式拟合
    pd=polyder(p)
    pd =
      -19.6217   20.1293
```

说明：$y=-9.8108x^2+20.1293x-0.0317$ 的微分是 dy/dx=−19.6217x+20.1293。由于多项式的微分是另一个低一阶的多项式，在这种情况下拟合的多项式为二阶，其微分为一阶多项式。这样，微分为一条直线，它意味该微分与 x 成线性变化。

给定一些描述某函数的数据，MATLAB 提供了一个计算其非常粗略的微分的函数。这个函数命名为 diff，它计算数组中元素间的差分。具体调用格式为：

D=diff(X)　　求向量或矩阵的有限差分

因为微分定义为：

$$\frac{dy}{dx}=\lim_{h\to 0}\frac{f(x+h)-f(x)}{(x+h)-(x)}$$

则 y=f(x)的微分可近似为：

$$\frac{dy}{dx}\approx\frac{f(x+h)-f(x)}{(x+h)-(x)}$$　　这里 h>0

它是 y 的有限差分除以 x 的有限差分。因为 diff 计算数组元素间的差分，所以在 MATLAB 中，可近似求得函数的微分。

【例 30】用 diff 函数近似求得函数的微分

```
dy=diff(y) ./ diff(x);      %  应用数组近似计算微分，y，x 取自上例
dy =
Columns 1 through 7
24.2500  13.0200  28.8000  9.2000  2.6000  3.2000  19.0000
Columns 8 through 10
-0.8000   -1.8000   19.0000
```

说明：

由于 diff 计算数组元素间的差分，所以，其所得输出比原数组少了一个元素。这样，画微分曲线时，必须舍弃 x 数组中的一个元素。当舍弃 x 的第一个元素时，上述过程给出向后差分近似。而舍弃 x 的最后一个元素，则给出向前差分近似。如果比较上述两条曲线，显而易见，用有限差分近似微分会导致很差的结果，特别是被噪声污染了的数据。

3.5.5　微分方程的数值解

一般微分方程式描述系统内部变量的变化率如何受系统内部变量和外部激励(如输入)的影响。当常微分方程式能够解析求解时，可用 MATLAB 的符号工具箱中的功能找到精

确解，但是获得解析解一般是很困难的。在微分方程难以获得解析解的情况下，可以方便地在数值上求解。

MATLAB 系统提供如下 3 个函数解微分方程数值解：

```
[t, y] = ode23('F', tspan, yo, options, p1, p2, ...)
```

解一阶常微分方程组，采用低维方法

```
[t, y] = ode45('F', tspan, yo, options, p1, p2, ...)
```

解一阶常微分方程组，采用高维方法。

```
[t, y] = ode113('F', tspan, yo, options, p1, p2, ...)
```

解一阶常微分方程组，采用变维方法。

F 是定义函数文件名。该函数必须以 dx 为输出，以 t，y 为输入量。tspan = [t0 tfinal] 表示积分的起始值(t0)和终止值(tfinal)。y0 是初始状态列向量。options 可以定义函数运行时的参数，可以默认。p1，p2，...是向函数 F 输入的参数，可以默认。一般来说，ode113 比 ode45 运算速度高，而 ode45 比 ode23 运算速度高。

【例 31】解经典的范得波(Van der Pol)微分方程

$$\frac{d^2x}{dt^2} - w(1-x^2)\frac{dx}{dt} + x = 0$$

(1) 与所有的数值求解微分方程组的方法一样，高阶微分方程式必须等价地变换成一阶微分方程组。对于上述微分方程，通过重新定义两个新的变量来实现这种变换。

令 y1=x 且 $y_2=dy/dx$

则 $dy1/dt=y_2$

$dy2/dt=w(1-y2)-y2$

(2) 编写一个函数 M 文件，给定当前时间及 y1 和 y2 的当前值，该函数返回上述导数值。MATLAB 中，这些导数由一个列向量给出。在本例中，这个列向量为 yprime。同样，y1 和 y2 合并写成列向量 y。所得函数 M 文件是：

```
function yprime=vdpol(t , y);
yprime=[y(2)  2*(1-y(1)^2)*y(2)-y(1)]';  % 输出的结果必须是列向量，令 w=2
```

(3) 计算结果如下：

```
[t,  y]=ode23('vdpol',  [0 30], [1;0], [], 3);  %  to=0 ,  tf=30 ,
yo=[1; 0]
y1=y(:, 1);  % 第一列向量，速度
y2=y(:, 2);  % 第二列向量，位移
plot(t, y1, ':b', t, y2, '-r')
legend('速度', '位移')
```

结果如图 3.5 所示。

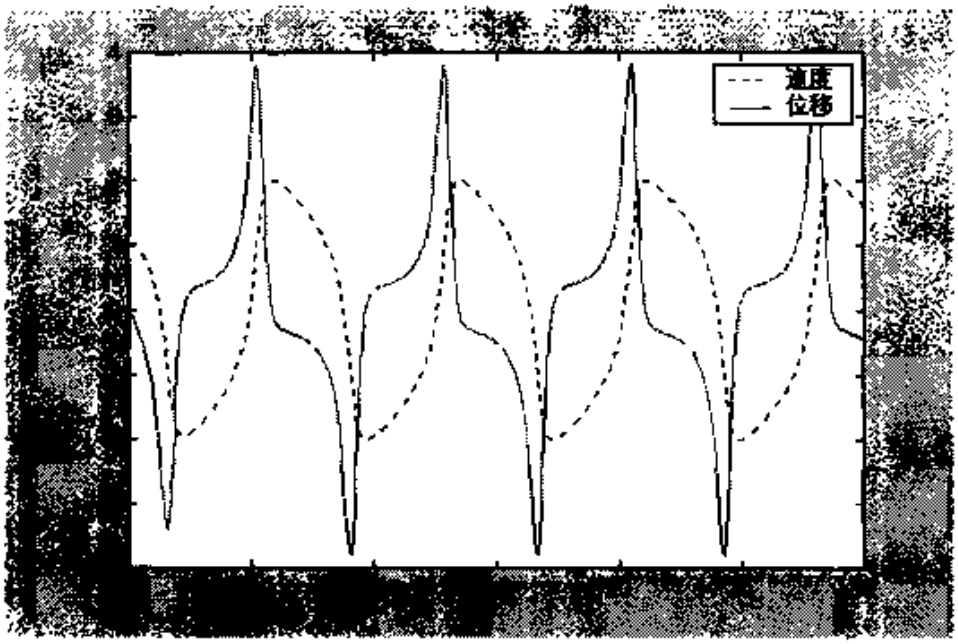


图 3.5　Van der Pol方程的解

说明：

函数 ode45 的使用与 ode23 完全一样。两个函数的差别在于必须与所用的内部算法相关。两个函数都运用了基本的龙格-库塔(Runge-Kutta)数值积分法的变形。ode23 运用一个组合得 2/3 阶龙格-库塔-芬尔格(Runge-Kutta-Fehlerg)算法，而 ode45 运用组合的 4/5 阶龙格-库塔-芬尔格算法。一般，ode45 可取较多的时间步。所以，要保持与 ode23 相同误差时，在 t0 和 tf 之间可取较少的时间步。然而在同一时间，ode23 每时间步至少调用 f_name 3 次，而 ode45 每时间步至少调用 f_name 6 次。

正如使用高阶多项式内插常常得不到最好的结果一样，ode45 也不总是比 ode23 好。如果 ode45 产生的结果，对作图间隔太大，则必须在更细的时间区间对数据进行内插，比如用函数 interp1。这个附加时间点会使 ode23 更有效。作为一条普遍规则，在所计算的导数中，如有重复的不连续点，为了保持精度，致使高阶算法减少步长，这时低阶算法更有效。正是由于这个原因，电子电路分析按系统默认的设置，就用一阶算法编程，并且最多提供二阶算法来解决暂态时间响应问题。此外，通过对 tol 设置更小的值，以达到更高的精度，但没有必要使绝对误差更小。这是因为 tol 设置每步的相对精度，不一定引起绝对误差减少。

总之，不要盲目使用数值方法。对于给定的问题，在决定最好的方法之前，要试一试各种可能的方法。有关微分方程数值解法的更进一步信息，请参考数值分析的相关书籍。

第 4 章
MATLAB 的符号计算功能

本章要点：

符号计算有两大特点：符号解和任意精度解。MATLAB 的符号计算借助符号工具箱(Symbolic)实现。本章介绍了符号计算 M 文件的特点和使用规则，并专门讲述了十分方便的“符号函数计算器”，另外还有一些关于扩展符号计算功能的方法。

本章具体包括以下内容：

- 符号表达式和符号矩阵的创建
- 符号矩阵的基本运算
- 因式分解、展开和简化
- 符号矩阵分解
- 符号微积分
- 符号代数方程求解
- 符号函数的二维图形
- 符号计算的扩展
- 图示的符号函数计算器
- 符号计算指令的联机帮助

MATLAB 所具有的符号数学工具箱与其他所有工具不同，它有广泛的用途，而不是针对一些特殊专业或专业分支。另外，MATLAB 符号数学工具箱与其他的工具箱区别还在于它使用字符串来进行符号分析，而不是基于数组的数值分析。为此，本章将对该工具箱进行具体介绍。

符号数学工具箱是操作和解决符号表达式的符号数学工具箱(函数)集合，有复合、简化、微分、积分以及求解代数方程和微分方程的工具。另外还有一些用于线性代数的工具，求解逆、行列式、正则型式的精确结果，找出符号矩阵的特征值而无由数值计算引入的误差。工具箱还支持可变精度运算，即支持符号计算并能以指定的精度返回结果。

符号数学工具箱中的工具建立在功能强大的称作 MAPLE 软件的基础上，它最初是由加拿大的沃铁卢(Waterloo)大学开发的。当要求 MATLAB 进行符号运算时，它就请求 MAPLE 去计算并将结果返回到 MATLAB 命令窗口。因此，在 MATLAB 中的符号运算是 MATLAB 处理数值的自然扩展。

4.1 符号表达式和符号矩阵的创建

4.1.1 符号表达式和符号方程的创建

在数值计算(包括输入、输出及中间计算在内)的过程中，所运作的变量都是被赋了值的数值变量。而在符号计算的整个过程中，所运作的是符号变量(Symbolic variable)。

符号表达式是代表数字、函数、算子和变量的MATLAB字符串或字符串数组，不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则这些符号表达式几乎完全像基本的MATLAB命令。表4.1列出几个符号表达式例子以及MATLAB等效表达式。

符号计算中出现的数字也都是当作符号处理的。

符号表达式(Symbolic expression)和符号方程(Symbolic equation)是两种不同的对象。它们的区别在于前者不包含等号(=)，而后者必须带等号。但这两种对象的创建方式是相同的。它们最简单和最常用的创建方式与 MATLAB 中的字符串变量的创建几乎相同。例如：

```
f='sin(x)^2'   %所创建的函数 sin²(x)赋给变量 f
eq='a*x^2+b*x+c=0'   %所创建的方程 ax²+bx+c=0 赋给变量 eq
de='dy+y^2=1'   %所创建的微分方程 y'+y²=1 赋给变量 de
```

表 4.1　符号表达式实例

符号表达式	MATLAB 表达式
$\frac{1}{2x^n}$	'1/(2*x^n)'
$y=\frac{1}{\sqrt{2x}}$	y='1/sqrt(2*x)'
$\cos(x^2)-\sin(2x)$	'cos(x^2)-sin(2*x)'
$M=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym('[a, b; c, d]')
$\int_0^b \frac{x^3}{\sqrt{1-x}}dx$	f=int('x^3/sqrt(1-x)', 'a', 'b')

注意：

- 第一个等号右边的部分，既可以是符号表达式也可以是符号方程。以上 3 个例子都把创建的符号表达式或符号方程赋给了符号变量。引入符号变量的目的是为以后调用方便，但这并不是必须的。事实上，符号表达式可以直接参与运算。
- 符号表达式和符号方程对空格都是敏感的。因此，在创建符号表达式时，不要在字符间任意乱加“修饰性”空格符。由于符号表达式可以看作 1×1 的符号矩阵，因此它也可以从 sym 指令创建。如 f=sym('sin(x)^2')与 f='sin(x)^2' 的作用完全一样。

4.1.2　符号变量

当字符表达式中含有多于一个的变量时，只有一个变量是独立变量。如果不告诉 MATLAB哪一个变量是独立变量，MATLAB将基于以下规则选择一个：

在符号表达式中默认的独立变量是唯一的。MATLAB 对单个英文小写字母(除 i、j 外)进行搜索，且以 x 为首选独立变量。如字符不是唯一的，就选择在字母顺序中最接近 x 的字母。如果有相连的字母，就选择在字母表中较后的那一个。

默认的独立变量，有时称作自由变量，在表达式 '1/(5+cos(x))' 中是'x'；在'3*y+z' 中是'y' ；在 'a+sin(t) '是' t '。在表达式 'sin(pi/4)-cos(3/5)' 中自由符号变量是' x '，因为此式是一个符号常数而无符号变量。可利用函数 symvar 询问 MATLAB 认为在符号表达式中哪一个变量是独立变量。例如：

```
symvar('a*x+y')   %返回默认的独立变量，这里是x
```

如果symvar利用规则不能找到一个默认独立变量，它便假定无独立变量并返回x。这一结论对含有由多个字母组成的变量(如alpha或s2的表达式)或不含变量的符号常数均成立。如果需要，绝大多数命令都使用用户选项以指定独立变量。

4.1.3 符号矩阵的创建和修改

本小节着重叙述 sym 指令在创建、引用和修改符号矩阵中的使用。

1. sym指令创建符号矩阵的直接输入法

这是模仿 MATLAB 数值矩阵的直接输入法设计的。矩阵元素可以是任何(不带等号)的符号表达式；各矩阵元素的长度可以不同；矩阵行之间用分号(；)分开；各矩阵元素间用逗号(，)分隔。例如，在 MATLAB 环境下运行：

```
msy=sym('[1/(a+x),sin(x),(b-x)/(a+x);1,exp(x),x^2]')
```

显示结果为：

```
msy =

[    1/(a+x),     sin(x), (b-x)/(a+x)]
[          1,     exp(x),         x^2]
```

2. 创建符号矩阵的字符串直接输入法

这是模仿MATLAB字符串矩阵的直接输入法设计的。在这种方法中，不需要调用sym指令，但要保证同一列中各元素字符串有同样的长度。为此，在较短字符串的前后可用空格符填充。例如，上述的msy矩阵也可以用以下指令创建：

```
msy=['[1/(a+x), sin(x), (b-x)/(a+x)]'; '[1, exp(x), x^2]'];
```

注意： 符号矩阵的每一行(row)的两端都有方括号“[ ]”，这是与 MATLAB 字符串矩阵的一个非常重要的区别。

3. 把数值矩阵转换为符号矩阵

MATLAB中的数值矩阵不能直接参与符号运算，必须通过转换才行。指令msy(Mnu)就把数值矩阵Mnu转换为符号矩阵。不管数值矩阵Mnu的元素原先是用分数还是浮点数表示，转换后的符号矩阵都将以最接近的精确有理形式给出。例如有数值矩阵

```
Mnu=[2/3,sqrt(3)/3,0.333;2.5,1/0.7,log(3)]
Mnu =
    0.6667    0.5774    0.3330
    2.5000    1.4286    1.0986
```

可用以下指令将把 Mnu 转换为符号矩阵：

```
Msy(Mnu)
ans =
[2/3,            sqrt(1/3),              333/1000]
[5/2,10/7,log(3)]
```

把这个指令与MATLAB中许多数值特殊矩阵生成指令(如eye，ones，zeros，magic)配合使用，可以产生许多特殊的符号矩阵。

4.　符号矩阵的引用和修改

在数值计算中，通过一个指令就可实现对矩阵任何一个子阵的引用和修改。但在符号计算中，引用(Quote)和修改(Modify)只能对符号矩阵的元素一个一个地进行。

引用指令：

msy(S，i，j)

给出符号矩阵 S 的第(i，j)个元素。

修改指令：

msy(S，i，j，'expr')

用 expr 替代符号矩阵 S 中原先的第(i，j)个元素。

4.2　符号矩阵的基本运算

一旦创建了一个符号表达式，也许想以某些方式改变它，也许希望提取表达式的一部分、合并两个表达式或求得表达式的数值。有许多符号工具可以帮助完成这些任务。

所有符号函数(例外的情况很少，讨论于后)都作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可能看起来像一个数字，但事实上它是一个内部用字符串表示的符号表达式。可以运用MATLAB函数isstr来找出像似数字的表达式是否真是一个数值或是一个字符串。

在 MATLAB 的数值计算中，矩阵的加、减、乘、除等运算的操作指令都很直观简单。在符号计算中，情况就不同了，所有涉及符号计算的操作都要借助专用函数进行。

以下对符号矩阵适用的指令，都适用于符号表达式。当然，这些指令对符号方程是没有意义的。

4.2.1　符号矩阵的加、减、乘、除运算

实现符号矩阵加(Add)、减(Subtract)、乘(Multiply)的指令分别是：

symadd(A，B)　给出两个符号矩阵的和 A+B

symsub(A，B)　给出两个符号矩阵的差 A–B

symmul(A，B)　给出两个符号矩阵的乘积 A×B

注意：

- symadd(A–B)，决不意味着 A+(–B)=A–B，因为(–B)对于符号运算是非法的。
- 相乘规则是：符号表达式可以与符号矩阵相乘；两个内维数相同的符号矩阵可以相乘。

【例 1】 计算 $\frac{1}{(s+1)(s+3)}$ 与 $\begin{bmatrix}(s+1) & s\\ & (s+1.5)\end{bmatrix}$ 相乘，并把结果赋给 G

```
G=symmul('1/(s+1)/(s+3)',sym('[s+1,s;0,s+1.5]'))
G =
    [ 1/(s+3),1/(s+1)/(s+3)*s]
    [ 0,  1/(s+1)/(s+3)*(s+1.5)]
```

4.2.2 符号矩阵的逆和除运算

符号矩阵的求逆(Inverse)指令和符号矩阵的除(Divide)运算指令如下：

inverse(B)	求 B^{-1}
symdiv(A,B)	求 A/B，即 AB^{-1}

注意： ● A 可以是符号表达式或(m × n)维符号矩阵；B 可以是符号表达式或(n × n)维满秩符号阵。

● inverse(B)和 symdiv('1',B)等价，而 symmul(A,inverse(B))与 symdiv(A,B)等价。

4.2.3 符号矩阵的幂运算

与数值计算相比，符号计算对幂(Power)运算所加的限制条件更多，否则计算结果就很难保证正确。具体如下：

sympow(S，p) 求幂运算指令 S^p

注意： 若 S 为标量符号表达式，p 可为标量符号或数值表达式；若 S 为符号方阵，p 必须为整数。

4.2.4 符号矩阵的综合运算指令

除了前面几小节介绍的加、减、乘、除、幂等单种符号计算指令外，在符号数学工具包中还有一个综合运算指令 symop。提供此指令是出于以下两个考虑：

- 实际使用的需要。对于多个符号矩阵或混有数值矩阵，要实现相互间多种运算时，假如仅仅依靠单种运算指令去做，就显得缺乏效率。
- 软件编制上的方便。实际上，那些单种运算文件，都是在综合运算函数文件 symop.m 的基础上开发的。

符号矩阵的综合运算(Operations)指令如下：

symop(s1,s2,s3,…)	符号矩阵的综合运算
s1,s2,s3,…	分别是符号矩阵或数值矩阵或 '+','-','*','/'、'^'、'('和')'

【例 2】　这里以一个简单的例子来说明综合运算与多次调用单种运算指令的比较

```
A=sym('[a1,a2;a3,a4]');B=sym('[b1,b2;0,b4]');N=[1,2;3,4];
F=symop(A,'/','(',B,'+',N,')');      %使用综合运算指令
FF=symdiv(A,symadd(B,sym(N)))   %由单种运算指令复合而成的输入指令
```

可以看到二者的结果相同，均为(以变量 F 为例)

```
F =
[ -(-b4*a1-4*a1+3*a2)/(-3*b2-2+b4*b1+b4+4*b1),
(b1*a2-2*a1+a2-a1*b2)/(-3*b2-2+b4*b1+b4+4*b1)]
[(b4*a3+4*a3-3*a4)/(-3*b2-2+b4*b1+b4+4*b1),
 -(-b1*a4-a4+2*a3+a3*b2)/(-3*b2-2+b4*b1+b4+4*b1)]
```

注意：
- 当综合运算指令中同时存在符号表达式和符号矩阵时，所得结果不一定正确。
- 当综合运算指令中的括号“()”两侧同时存在乘、除运算时，所得结果不一定正确。

有鉴于此，本书建议用户首先采用单种运算指令的复合调用去完成比较复杂的符号矩阵运算。

4.3　因式分解、展开和简化

4.3.1　因式分解和展开

对符号表达式、符号矩阵可以进行如下操作：

factor(S)　对 S 进行因式分解操作。S 可以是单个表达式、有理分式、方程或矩阵

expand(S)　对符号矩阵进行展开

collect(S)　把 S 符号矩阵元素中 x 的同幂项系数进行合并

collect(S,v)　把 S 符号矩阵元素中 v 的同幂项系数进行合并

【例 3】符号矩阵的分解

```
G=sym('[(s+1)/(s^2+5*s+6),2*(s+1)/(s^2+2*s-3);(s^2-
s)/(s^2+2*s+1),s/(s+2)]');
factor(G)
ans =
[   (s+1)/(s+3)/(s+2),  2*(s+1)/(s+3)/(s-1)]
[    s*(s-1)/(s+1)^2, s/(s+2)]
```

【例 4】符号矩阵的展开

```
GS=sym('[sin(x+y)/cos(2*x);exp(x-
y);log(a*b^2/c);(s+1)*(s+3)/((s+2)*(s+4))]');
expand(GS)
```

```
ans =
[  1/(2*cos(x)^2-1)*sin(x)*cos(y)+1/(2*cos(x)^2-1)*cos(x)*sin(y)]
[             exp(x)/exp(y)]
[             log(a*b^2/c)]
[             1/(s+2)/(s+4)*s^2+4/(s+2)/(s+4)*s+3/(s+2)/(s+4)]
```

注意： 展开将对该矩阵元素的因式乘积、三角函数、指数函数、对数函数进行，但在分母的因式乘积将不展开.

【例 5】同幂项的合并

```
S=sym('a*sin(t)*x^2+b/t*sin(t)*x^2+cos(t)*x+sin(t)^2*x+log(b)
*sin(t)*2');
XS=collect(S)                          %合并 x 的同幂项系数
SS=collect(S,'sin(t)')          %合并 sin(t)的同幂项系数
XS =(b/t*sin(t)+a*sin(t))*x^2+(cos(t)+sin(t)^2)*x+2*log(b)*sin(t)
SS =sin(t)^2*x+(a*x^2+b/t*x^2+2*log(b))*sin(t)+cos(t)*x
```

4.3.2 符号矩阵的简化

本小节介绍综合性简化函数 simple(S)，该指令将使符号矩阵 S 中的每个元素以最简短的形式给出。为了实现这一目的，该 simple.m 函数依次从 MAPLE 中调用以下功能：

- simplify 利用函数规则(如三角函数恒等式)对表达式进行简化。
- radsimp 对包含根式的表达式进行简化。
- combine 把表达式中以求和形式、乘积形式、幂次形式表示的各项(Terms)归并成单项。在许多情况下，该变换是 expand 的反变换。
- factor 实现多变量多项式的因式分解。
- convert 把表达式从一种形式转换为另一种形式。例如，它可实现从多项式到嵌套式(Horner 或 Nested)的转换。
- collect 合并同幂项系数。

综合性简化函数有以下 2 种调用形式：

simple(S)　显示 S 被简化的过程，并给出最简结果

[R,HOW]=simple(S)　不显示简化过程。R 为简化结果，HOW 为特别的简化操作

【例 6】　符号向量的简化

```
S=sym('[sqrt(x^2+2*x+1);sin(x)^2+cos(x)^2;(a+b*i)*(a-b*i)]');
[R,HOW]=simple(S)
R =
[           1+x]
[             1]
[       a^2+b^2]
HOW =
combine(trig)
```

最后一个显示结果表示，简化过程中用了三角函数简化工具。

4.4　符号矩阵分解

与数值计算一样，在符号计算中，符号矩阵分解也特别有用。计算符号矩阵维数、转置、行列式、特征值分解、奇异值分解、零空间和列空间分解的指令如下所示：

symsize(S)	求 S 矩阵维数的指令
transpose(S)	求 S 的符号转置矩阵
determ(S)	求 S 阵的行列式
colspace(S)	给出 S 列空间的基
symsize(colsize(S),2)	给出 S 的秩
nullspace(S)	给出 S 零空间的基
symsize(nullspace(S),2)	给出 S 的零化度
[VE, E]=eigensys(S)	VE 给出矩阵 S 的特征向量；E 给出 S 的特征值
[VJ, J]=jordan(S)	VJ 给出符号矩阵 S 的广义特征向量；J 给出相应的约当标准形
singvals(A)	给出一般符号阵 A 的奇异值
[U,S,V]=singvals(A)	给出(不带自由变量的)A 阵的奇异分解三对组

【例 7】　无重根阵的分解

```
D=sym('[1,-3;2,2/3]');
[VE,E]=eigensys(D)
VE =
[1,1]
[ 1/18-1/18*i*215^(1/2), 1/18+1/18*i*215^(1/2)]
E =
[5/6+1/6*i*215^(1/2),                0]
[0, 5/6-1/6*i*215^(1/2)]
```

【例 8】　有重根阵的分解

```
C=sym('[1,1,2;0,1,3;0,0,2]');
[VJ,J]=jordan(C)
VJ =
[  5,      -5,  -5]
[  3,       0,  -5]
[  1,       0,   0]
J =
[   2,     0,   0]
[   0,     1,   1]
[   0,     0,   1]
```

4.5 符号微积分

微分和积分是微积分学研究和应用的核心，并广泛地用在许多工程学科。MATLAB符号工具能帮助解决许多这类问题。

4.5.1 符号微分

符号表达式的微分以4种形式利用函数diff。

【例9】 符号表达式的微分

```
f='a*x^3+x^2-b*x-c ';          % 定义一个符号表达式
diff(f)  % 对默认的变量x求微分
ans=
3*a*x^2+2*x-b
diff(f, 'a ')  % 对变量a求微分
ans=
x^3
diff(f, 2)  % 对默认的变量x求二次微分
ans=
6*a*x+2
diff(f, 'a', 2)          %  对变量a求二次微分
ans=
0
```

函数diff也可对数组进行运算。如果F是符号向量或数组，diff(F)对数组内的各个元素进行微分。

【例10】 符号数组的微分

```
F=sym( '[a*x, b*x^2; c*x^3, d*s]')   %建立一个符号数组
diff(F)                              %以x为变量,对数组元素求微分
ans=
[     a,          2*b*x]
[     3*c*x^2,        0]
```

注意函数 diff 也用在计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵 M，diff(M)计算 M(2：m，：)-M(1：m-1，：)的数值差分。

【例11】 数值向量的差分

```
M=[(1:  8).^2)]  % 建立一个向量
M=
1   4   9   16   25   36   49   64
diff(M)   % 计算元素之间的差分
```

```
ans=
3   5   7   9   11   13   15
```

如果diff的表达式或可变参量是数值，MATLAB就非常巧妙地计算其数值差分；如果参量是符号字符串或变量，MATLAB就对其表达式进行微分。

4.5.2 符号积分

积分函数 int(f)，其中 f 是一符号表达式。它力图求出另一符号表达式 F，使 diff(F)=f。正如从研究微分学所了解的，积分比微分复杂得多。积分或逆求导不一定以封闭形式存在；或者存在但软件也许找不到；或者软件可明显地求解，但超过内存或时间限制。当 MATLAB 不能找到逆导数时，它将返回未经计算的命令。

```
int('log(x)/exp(x^2)')    %试图对函数log(x)/exp(x^2)进行积分运算
ans=
log(x)/exp(x^2)
```

同微分一样，积分函数有多种形式。形式 int(f)相对于默认的独立变量求逆导数；形式 int(f, ' s ')相对于符号变量 s 求积分；形式 int(f, a, b)和 int(f, ' s ', a, b)求解符号表达式从 a 到 b 的定积分，其中 a、b 是数值；形式 int(f, 'm' , 'n')和形式 int(f, 's', 'm', 'n')，求解符号表达式从 m 到 n 的定积分，其中 m、n 是符号变量。

正如函数diff一样，积分函数int对符号数组的每一个元素进行运算。

【例12】 符号表达式的积分运算

```
f=sym('sin(s+2*x)') ;  %  建立一个符号表达式
int(f)    %  以x为变量进行积分
ans=
-1/2*cos(s+2*x)
int(f, 's')    %  以s为变量进行积分
ans=
-cos(s+2*x)
int(f, pi/2, pi)    %  以x为变量从pi/2 到pi进行积分
ans=
-cos(x)
int(f, ' s ', pi/2, pi)    %  以s为变量从pi/2 到pi进行积分
ans=
cos(2*x)-sin(2*x)
int(f, ' m ', ' n ')    %  以x为变量从m到n进行积分
ans=
-1/2*cos(s+2*n)+1/2*cos(s+2*m)
F=sym('[a*x, b*x^2; c*x^3, d*s]')    %  建立一个符号型数组
int(F)    %  对F中的各元素以x为变量进行积分
```

```
ans=
[1/2*a*x^2,    1/3*b*x^3]
[1/4*c*x^4,     d*s*x]
```

4.5.3 符号矩阵的代数运算

除了上面介绍的几种微积分函数之外，还有一些函数在对符号矩阵进行处理时也会用到。本节着重介绍求符号和、符号导数、符号积分、泰勒级数以及符号雅可比(Jacobian)矩阵。具体指令如下：

symsum(S,v)	关于指定变量v对通项S求不定和
symsum(S,v,a,b)	指定变量v在[a,b]之间取值时，对通项S求和
taylor(F,v)	求F对变量v的泰勒展开，6阶小量以余项形式给出
taylor(F,v,n)	求F对变量v的泰勒展开，n阶小量以余项形式给出
jocobian(F,v)	求F的雅可比矩阵，F是向量函数，是指定变量构成的向量

说明：上述指令中的v可以省略。默认时，运算将对默认变量进行。

【例 13】 求无限项级数的和

```
symsum(1/k^2,1,Inf)
ans =
1/6*pi^2
```

【例 14】 求 $\sin(x)e^{-x}$ 的泰勒级数

```
FT=taylor(sin(x)*exp(-x),8)
FT =
x-x^2+1/3*x^3-1/30*x^5+1/90*x^6-1/630*x^7
```

【例 15】 求 $[xyz \quad y\log(z) \quad \sin(x)e^{y/x}]^T$ 对 $[x \quad y \quad z]^T$ 的雅可比矩阵

```
F=sym('[x*y*z;y*log(z);sin(x)*exp(y/z)]');
v=sym('[x;y;z]');
FJ=jacobian(F,v)
FJ =
[      y*z,              x*z,                    x*y]
[        0,           log(z),                    y/z]
[ cos(x)*exp(y/z),  sin(x)/z*exp(y/z),  -sin(x)*y/z^2*exp(y/z)]
```

4.6 符号代数方程求解

本节介绍两部分内容：线性方程组(Linear equations)的符号解(Symbolic solutions)和一般代数方程组(Algebraic equations)的符号解。

4.6.1　线性方程组的符号解

本小节只讨论 A 阵至少行满秩时的线性方程组 A*X=B 的解。下面是指令的使用格式：

X＝linsolve(A，B)　　只给出特解(Particular　solution)

[X，Z]=linsolve(A，B)　　将给出由 X 及 Z 构成的通解(General　solution)

说明：

- 当 A 阵列数大于行数时，将给出解不唯一的警告提示。
- X 是方程组的一个特解，Z 是 A 阵零空间的基，通解形式是(X＋p*Z)，式中 p 是可任意取值的自由参数。

【例 16】　求给定线性方程组的解

```
A=sym('[1,1/2,1/3;3,1,1;1,2,1]');
B=sym('[1,2;1/3,1;1,1/7]');
X=linsolve(A,B)
X =
[    7/3,    38/7]
[   16/3,      10]
[    -12, -177/7]
```

【例 17】　求欠定方程的通解

```
A=sym('[1,1/2,1/3;3,1,1]');
B=sym('[1;1]');
X=linsolve(A,B)
X =
[   0]
[   4]
[  -3]
```

4.6.2　一般代数方程的解

本小节所讲的 solve 指令，可以解一般代数方程，包括线性(Linear)方程、非线性(Nonlinear)方程和超越方程(Transcedental equation)。当方程组不存在解析解时，若又无其他自由参数，则 solve 将给出数值解。具体使用格式如下：

solve(S)	对一个方程的默认变量求解
solve(S,v)	对一个方程的指定变量 v 求解
solve(S1,S2,…,SN)	对 N 个方程的默认变量求解
solve(S1,S2,…,SN,v1,v2,…,vn)	对 N 个方程的 v1,v2,…,vn 变量求解
[x1,x2,…,xn]=solve(S1,S2,…,SN)	对默认变量求解的结果赋给 x1,x2,…,xn
[x1,x2,…,xn]=solve(S1,S2,…,SN, v1,v2,…,vn)	对 v1,v2,…,vn 求解的结果赋给 x1,x2,…,xn

【例 18】 求符号特征多项式的根

```
A=sym('[1,2,1/3;0,5,0;7,0,a]');
CA=poly(A)
RA=solve(CA)
CA =
x^3-x^2*a-6*x^2+6*x*a+8/3*x-5*a+35/3
RA =
[                                    5]
[    1/2*a+1/2+1/6*(9*a^2-18*a+93)^(1/2)]
[ 1/2*a+1/2-1/6*(9*a^2-18*a+93)^(1/2)]
```

【例 19】 求三元非线性方程组的解

```
S1='x^2+sqrt(2)*x+1=0';S2='x+3*z=4';S3='y*z=-1';
[x,y,z]=solve(S1,S2,S3)
xv=vpa(x,6)
yv=vpa(y,6)
zv=vpa(z,6)                    %以上三个指令给出解析解的 6 位浮点近似
x =
[ -1/2*2^(1/2)+1/2*i*2^(1/2)]
[ -1/2*2^(1/2)-1/2*i*2^(1/2)]
y =
[12/257*2^(1/2)*(-1/2*2^(1/2)+1/2*i*2^(1/2))+45/514*2^(1/2)-51/514
*i*2^(1/2)-180/257]
[12/257*2^(1/2)*(-1/2*2^(1/2)-1/2*i*2^(1/2))+45/514*2^(1/2)+51/514*
i*2^(1/2)-180/257]
z =
[ 1/6*2^(1/2)-1/6*i*2^(1/2)+4/3]
[ 1/6*2^(1/2)+1/6*i*2^(1/2)+4/3]
xv =
[ -.707105+.707105*i]
[ -.707105-.707105*i]
yv =
[ -.623269-.936276e-1*i]
[ -.623269+.936276e-1*i]
zv =
[ 1.56903-.235702*i]
[ 1.56903+.235702*i]
```

【例 20】 求解超越方程组

```
[x,y]=solve('x^x=1/7','x/y=3')
x =
```

```
-log(7)/lambertw(-log(7))
y =
-1/3*log(7)/lambertw(-log(7))
```

说明：

在上述符号解中的 lambertw(ω)代表ω函数ω(x)e^{ω(x)}=x 的解。有关内容，请用 mhelp 指令访问 MAPLE 库的 lambertw 条目。

4.7　符号微分方程求解

本节介绍常微分方程的计算机求解指令 dsolve。该指令的使用格式为：

```
[y1, y2, …]=dsolve(a1, a2,…,a12)
```

说明：

- 输入变量包括三部分内容：微分方程、初始条件和指定独立变量，其中微分方程是必不可少的输入内容。后两个内容视需要而定，可有可无。
- 关于指定独立变量的规定：若要指定独立变量，则总是由全部输入变量 a1，a2,…中的最后一个变量定义。若不对独立变量加以专门的定义，则本指令默认小写英文字母 x 或 t 为独立变量。
- 微分方程的记述规定：当 y 是应变量时，用 Dny 表示“y 的 n 阶导数”。比如：

  Dy 表示形如 $\frac{dy}{dx}$ 或 $\frac{dy}{dt}$ 的 y 的一阶导数；

  Dny 表示形如 $\frac{d^n y}{dx^n}$ 或 $\frac{d^n y}{dt^n}$ 的 y 的 n 阶导数。
- 关于初始条件的规定：初始条件被写成 y(a)=b，Dy(c)=d 等。a、b、c、d 可以是变量使用符外的其他字符。当初始条件少于微分方程数时，在所得解中将出现任意常数符 C1，C2，……，解中任意常数符的数目等于所缺少的初始条件数。
- 输出变量可有可无。当有输出变量时，MATLAB 工作内存中将在 y1，y2，…… 定义的输出变量中保存计算结果。
- 输入变量 a1,a2,…的个数不超过 12 个。但这不意味着该指令可以解的联立方程数为 12。因为，一个输入变量可以一次定义多个微分方程或初始条件。

【例 21】　求 $\frac{dx}{dt}=y,\frac{dy}{dt}=-x$ 的解

```
[x,y]=dsolve('Dx=y,Dy=-x')
x =
cos(t)*C1+sin(t)*C2
y =
-sin(t)*C1+cos(t)*C2
```

说明：

- t 是默认的独立变量。

- C1、C2 为任意常数。

【例 22】 求$\frac{df}{dx}=3f+4g,\frac{dg}{dx}=-4f+3g,f(0)=0,g(0)=1$的解

```
[f,g]=dsolve('Df=3*f+4*g, Dg=-4*f+3*g', 'f(0)=0,g(0)=1')
f =
exp(3*t)*sin(4*t)
g =
exp(3*t)*cos(4*t)
```

【例 23】 求解三阶微分方程$\frac{d^3y}{dt^3}=-y$，初始条件为 $y(0)=1$，$\frac{dy(0)}{dt}=0$，$\frac{d^2y(0)}{dt^2}=0$。

```
y=dsolve('D3y=-y','y(0)=1,Dy(0)=0,D2y(0)=0','t')
y =
1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*3^(1/2)*t)
```

4.8 符号函数的二维图形

通过对符号函数的数值计算，利用下一章介绍的 MATLAB 图形指令，很容易对符号函数进行可视化表现。因此，本节不对可视化指令做全面地介绍，而是专门介绍一个易于使用的一元符号函数 ezplot。其基本使用格式如下：

```
ezplot(F,[xmin,xmax],fig)
```

说明：

- 输入变量 F 是待绘的符号函数。
- [xmin,xmax]是界定绘图的自变量范围，可以默认，默认值为[−2π，2π]。
- 输入变量 fig 是指定图形窗口的，默认时默认为当前图形窗口。

【例 24】 绘制上节最后一个解函数 y(t)的图形(图 4.1)

```
ezplot(y)  %绘制符号函数 y(t)在[-2*pi, 2*pi]中图形的指令
```

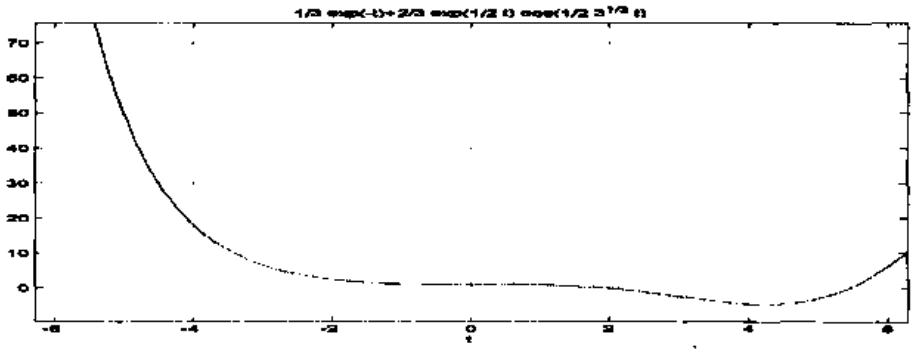


图 4.1 符号函数图形

4.9　符号计算的扩展

相对 MAPLE 软件的 2000 余条符号计算指令而言，前面几节所介绍的内容仅利用了 MAPLE 最常用计算指令中的一部分。为了在 MATLAB 工作环境中，进一步利用 MAPLE 的其他符号计算能力，本节将介绍把 MATLAB 与 MAPLE 联接起来的 2 个重要指令 maple 和 mpa。前者用于调动 MAPLE 的符号计算“引擎”和它庞大的函数库，并把最终的计算结果送到 MATLAB 工作区；后者用于向 MAPLE 工作内存输送 MATLAB 中已有的变量内容。

与前面一些符号计算指令相比，MAPLE 的调用要求 MATLAB 用户对 MAPLE 软件有更多的理解和认识。

4.9.1　直接调用 MAPLE 的符号计算能力

在 MATLAB 环境下，为了实现对 MAPLE 绝大多数符号计算指令的调用，该符号数学工具包提供了一个通用指令 maple。该指令的主要使用格式如下：

maple(MAPLEStatement)　　调用 MAPLE 中的完整指令 MAPLEStatement

maple(a0,a1,a2,...,a10)　　调用 MAPLE 中名为 a0 的指令

说明：a0 是字符串；a1,a2,...是 a0 指令使用格式中依次相应的输入变量。

【例 25】　求递推方程 f(n)=−3f(n−1)−2f(n−2)的通解

调用格式一：

```
MAPLE('rsolve(f(n)=-3*f(n-1)-2*f(n-2),f(k))')
ans =
   (2*f(0)+f(1))*(-1)^k+(-f(0)-f(1))*(-2)^k
```

调用格式二：

```
MAPLE('rsolve','f(n)=-3*f(n-1)-2*f(n-2),f(k)')
ans =
    (2*f(0)+f(1))*(-1)^k+(-f(0)-f(1))*(-2)^k
```

说明：

- 在现有的 MATLAB 符号数学工具包中，MATLAB 虽没有提供解“递推方程”的专用函数，但是可以通过 MAPLE 这个通用符号接口，直接调用 MAPLE 的 rsolve 来解“递推方程”，从而使 MATLAB 的符号计算能力得到进一步的开拓。
- 无论是 MAPLE 的哪种调用格式，都需要对 MAPLE 的相关指令有直接的认识和理解。

【例 26】　求 f=xyz 的 Hessian 矩阵

```
FH=MAPLE('hessian(x*y*z,[x,y,z])')
FH =
   matrix([[0, z, y], [z, 0, x], [y, x, 0]])
```

注意： 在现有的版本中，由于 maple.m 文件编写上的原因，对于像本例这种在一个参数位置上含有方括号的多变数的情况，MAPLE 的调用格式二将无法运用。有兴趣的用户不妨试试。

- 请记住，MAPLE 的调用格式一对 MAPLE 软件指令的适应性要比调用格式二强得多。

【例 27】 求 $\sin(x^2+y^2)$在 x=0,y=0 处展开的截断 8 阶小量的泰勒近似式

过程 1：

```
MAPLE('mtaylor(sin(x^2+y^2),[x=0,y=0],8)')
ans =
    mtaylor(sin(x^2+y^2),[x = 0, y = 0],8)
```

可以看到，展开没有进行。

过程 2：

```
MAPLE('readlib(mtaylor);');
MAPLE('mtaylor(sin(x^2+y^2),[x=0,y=0],8)')
ans =
    x^2+y^2-1/6*x^6-1/2*y^2*x^4-1/2*y^4*x^2-1/6*y^6
```

所得的展开结果正确。

注意： MAPLE 软件所有的函数，在初始化时并未全部装入内存．因此，有些读库函数(Readlib-defined functions)在第一次调用前，需要先运用 MAPLE 的 readlib 指令把该函数由库读入内存。readlib 在 MATLAB 中的使用格式是 MAPLE('readlib(FunName);')，FunName 为待安装的函数名。

- 如果出现类似于本例过程一的现象(运算结果恰为输入的 MAPLE 指令本身)，则说明所用的函数也许还没有读入。
- mtaylor 指令不但能求在 x=0，y=0 处泰勒展开，而且能求在 x=x0，y=y0 处的一般展开式。

4.9.2 MAPLE 的调试

MAPLE 提供了 2 个功能函数用以调试(debugging)，它们是跟踪模式(trace mode)和输出返回值(status output argument)。

1. 跟踪模式

命令maple traceon可以显示所有对MAPLE调用的中间过程并将中间结果显示在屏幕上。例如，使用以下命令可以将计算exp(2*a)的过程显示出来。

```
maple traceon
a = sym('a');
exp(2*a)
statement =
```

```
(2)*(a);
result =
2*a
statement =
exp(2*a);
result =
exp(2*a)
ans =
exp(2*a)
```

使用maple traceoff指令可以关闭跟踪模式，使过程不再显示出来。

2. 输出返回值

maple函数可返回2个参数result 和 status。如果对maple过程的所有调用成功，则参数result的值为计算结果，而参数status的值为0。如果调用失败，则MAPLE返回一个错误代码(一个正整数)，保存于status参数中，而在result参数中，则是对应的警告(错误)信息。例如，maple的discrim函数(计算一个多项式的判别式)的语法为discrim(p,x)，这里p是关于x的多项式。如果在调用discrim时忘记输入该函数的第二个参数，例如：

```
syms a b c x
[result, status] = maple('discrim', a*x^2+b*x+c)
result =
Error, (in discrim) invalid arguments
status =
2
```

如果正确调用，如下所示：

```
[result, status] = maple('discrim', a*x^2+b*x+c, x)
result =
-4*a*c+b^2
status =
0
```

4.10　图形化的符号函数计算器

本节要介绍 MATLAB 符号数学工具包所提供的第三种符号计算方式——图形化的函数计算器。它的外形如图 4.2 所示。该图形化计算器是由 funtool.m 文件生成的，在 MATLAB 环境下，输入以下指令即可：

```
funtool
```

该图形化计算器由 2 个函数曲线视窗(图 4.2 中的 Figure No.1 和 Figure No.2)和一个函数运算控制器(图 4.2 中的 Figure No.3)构成。

4.10.1 函数曲线视窗的激活

在任何时候，两个函数视窗中只有一个是激活的，或是 Figure No.1，或是 Figure No.2。在图 4.2 中，Figure No.2 正处于激活状态。每个函数曲线视窗的激活均由鼠标直接控制。在该窗口上任意位置单击鼠标左键，激活该函数视窗。

在函数运算控制器上的任何操作，都只对激活的函数视窗起作用。换句话说，随运算控制器上的不同操作，激活的函数视窗中的图示曲线将做相应的变化。

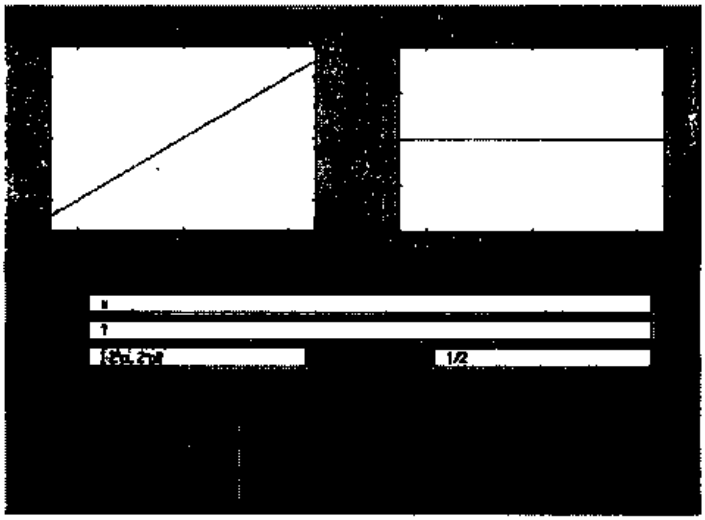


图 4.2 图形化函数计算器

4.10.2 运算控制器上被控栏的操作

被控栏是指在函数运算控制器(图 4.2 中的 Figure No.3)上半部分的那四个栏目：f、g、x、a。f、g 栏分别显示着相应视窗 Figure No.1 和 Figure No.2 中所视曲线的函数表达式；x 栏显示着函数曲线视窗中作为横坐标变量的取值范围；a 栏显示着可能与函数运算的自由参数值。

被控栏内容有以下 3 种来源和操作方式：

- 在图形化函数计算器打开之前，若 MATLAB 工作区中已有同名字符串变量 f、g、x、a 存在，那么这些变量的内容会被即将打开的图形化函数计算器所调用，反之，若 MATLAB 中没有已定义的同名变量，则新打开的图形化函数计算器将默认 f='x'，g='1'，x=[-2*pi, 2*pi]，a='1/2'。
- 在已打开的图形化函数计算器上，除 x 栏以外，被控栏 f、g 和 a 的内容可以用下述方法随时修改。移动鼠标使光标移到要修改内容之处，通过键盘操作就可写入用户所希望的内容。在以上操作结束之后，依次激活相应的函数曲线视窗，便可看到修改后所得的函数曲线。
- 当用运算控制器进行运算操作时，被激活的曲线视窗中的函数曲线和该视窗对应的被控栏中的函数表达式将实时地反映运算结果。同时，MATLAB 的变量 f 和 g 中的内容也做相应地改变。

4.10.3　单函数运算操作键

在运算控制器(图 4.2 中的 Figure No.3)上的第一排按钮都是单函数操作键。它们只对激活的那个函数进行如下功能的运算(如表 4.2 所示)。

表 4.2　单函数运算操作键

键　名	功　能
df/dx	求 f(x)相对于 x 的符号导数
int f	求 f(x)相对于 x 的符号积分
simple f	使 f(x)的表达式尽可能简化
num f	取 f(x)的分子表达式
den f	取 f(x)的分母表达式
1/f	求 1/f(x)
finv	求 f(x)的反函数，使 g(f(x))=x

如果在 intf 或是 finv 操作中，得不到收敛的结果，那么相应的函数栏里会出现 NaN，表示运算失败。

4.10.4　函数和参数运算操作键

在运算控制器(图 4.2 中的 Figure No.3)上的第二排按钮用以实现激活函数和自由参数 a 的操作，具体如表 4.3 所示。

表 4.3　函数和参数运算操作键

键　名	功　能
f+a	计算 f(x)+a
f-a	计算 f(x)-a
f*a	计算 af(x)
f/a	计算 f(x)/a
f^a	计算 f^(x)
f(x+a)	计算 f(x+a)
f(x*a)	计算 f(ax)

4.10.5　两个函数间的运算操作键

运算控制器上的第三排按钮用以实现两个函数间的运算。运算结果一方面图示在激活

的函数视窗里，一方面显示在相应的被控函数栏里。运算操作键的具体功能如表 4.4 所示。

表 4.4 两个函数间的运算操作键

键 名	功 能
f+g	求 f(x)+g(x)
f-g	求 f(x)-g(x)
f*g	求 f(x)g(x)
f/g	求 f(x)/g(x)
f(g)	求复合函数 f(g(x))
g=f	用 f(x)取代 g(x)
swap	交换 f(x)、g(x)

4.10.6 辅助操作键

运算控制器上的第四排按钮的功能如表 4.5 所示。

表 4.5 辅助操作键

键 名	功 能
Insert	把当前 Figure No.1 视窗里的函数插入到内含的典型函数演示表中
Cycle	在 Figure No.1 视窗里依次演示内含的典型函数演示表中的函数曲线
Delete	从内含的典型函数演示表中删除当前 Figure No.1 视窗中的函数
Reset	把整个函数计算器重置成初始调用状态
Help	在主 MATLAB 窗里给出函数计算器的联机帮助
Demo	自动演示函数计算器的运算功能
Close	关闭函数计算器

4.11 符号计算指令的联机帮助

关于符号计算指令的联机帮助要分两个层次来介绍。第一层次是关于 MATLAB 符号数学工具包中所有 M 文件的联机帮助；第二层次是关于 MAPLE 库函数指令的联机帮助。

4.11.1 符号数学工具包中 M 文件的联机求助

与 MATLAB 其他工具包联机帮助方法相同，获取 MATLAB 符号数学工具包中 M 文件的联机帮助信息有以下几种手段：

- 利用 dir c:\matlabr11\toolbox\symbolic 指令列出符号数学工具包中的所有文件,从中可以知道该包 M 文件的概貌和各 M 文件的确切名称。
- 利用 lookfor symbolic 指令可列出进行符号计算的各 M 文件的名称和简短说明。
- 利用 type c:\matlabr11\toolbox\symbolic\contents.m 获得关于各符号计算 M 文件功能的分类信息。
- 利用 help Mname(这里 Mname 是指具体的符号计算 M 函数名)获得关于具体函数使用方法的详细说明。

4.11.2　MAPLE 库函数联机帮助的检索树

获取 MAPLE 库函数指令联机帮助的主要手段是 mhelp，至于前面所介绍的获取符号计算 M 文件信息的 help 和 lookfor 在此几乎完全无用。

下面将沿着检索树自上而下地介绍 MAPLE 的联机帮助方式。

- mhelp index

index[expression]	表达式描述指令集
index[function]	函数指令集
index[misc]	辅助功能指令集
index[package]	库函数包指令集
index[procedure]	程序操作指令集
index[statement]	语句描述指令集

- mhelp index[CategoryName]

该指令中的 CategoryName 是指各目录名，如 function 等。该指令的执行将给出更详细的次下层信息。

- mhelp SubName

这里的 SubName 是指次下层的树枝名或是子包名。这些名字由使用 mhelp index [CategoryName] 得到。

- mhelp Fname

这里的 Fname 是指具体的 MAPLE 函数的名称。该 mhelp 将给出关于该指令的详细说明(包括定义、使用格式、使用举例及相关指令)。MAPLE 函数的确切名称可以用前面的方法联机查阅。

4.11.3　MATLAB 提供的 MAPLE 特殊函数名清单

为了用户查阅方便，MATLAB 编制了一个纯说明文件 mfunlist.m。该文件不做任何实质性的运算，仅采用 MATLAB 注释语句列出了几十条常用的 MAPLE 特殊函数名及简单的说明。在 MATLAB 环境下，help mfunlist 与 type mfunlist 相同。

第5章
MATLAB 程序设计

本章要点：

MATLAB 除了指令行操作的直接交互使用外，作为一种高级应用软件有自己的编程语言。要充分体现和发挥 MATLAB 的能力，必须掌握MATLAB程序设计。

本章具体包括以下内容：

- M文件的功能和特点
- M文件的形式
- 数据结构和全局变量
- 程序结构
- 程序流控制
- 字符与字符串
- 函数调用及变量传递
- M文件的调试

用 MATLAB 语言写的简单程序在前面章节中已有所介绍，本章将系统介绍如何用 MATLAB 语言进行程序设计。MathWork 公司将 MATLAB 语言称为第四代编程语言，足见其简洁有效。MATLAB 的编程效率比常用的 BASIC、C、FORTRAN 和 PASCAL 等语言要高得多，而且维护容易。

5.1 M 文件的功能和特点

MATLAB 有两种常用工作方式：一种是直接交互的指令行操作方式；另一种是 M 文件的编程工作方式。在前一种工作方式下，MATLAB 被当作一种高级“数学演算和图示器”来使用。这种指令行交互操作方式如何在 MATLAB 视窗中工作，在前面几章已经做了比较充分的论述。本章着重介绍与后一种工作方式有关的内容。

MATLAB 是一个强有力的操作环境。它集中 MATLAB 所提供的完整而易于使用的编程语言。从形式上讲，MATLAB 程序文件是一个 ASCII 码文件(标准的文本文件)，扩展名一律为.m(M 文件的名称由此而来)。用任何字处理软件都可以对它进行编写和修改。从特征上讲，MATLAB 是解释性编程语言。其优点是语法简单，程序容易调试，人机交互性强。缺点是由于逐句解释运行程序，故速度比编译型的慢。但是较慢运行速度仅明显表现在 M 文件初次运行时。因为 M 文件一经运行便变成代码存放在内存中；再次运行该文件时，MATLAB 将直接从内存中取出代码运行，大大加快了运行速度。从功能上讲，M 文件大大扩展了 MATLAB 的能力。MathWork 公司推出的一系列工具箱(Toolbox)就是明证。通过工具箱，MATLAB 才被应用到控制、信号处理、小波分析、系统辨识、图像处理、优化、样条分析、神经网络和金融财政等各个方面。而这些工具箱全部是由 M 文件构成的。从这点上来讲，如果不了解 M 文件，那么可以说，MATLAB 的能力仅应用了很小一部分。

由于 M 文件是解释性的程序语言，且以复数矩阵为基本运算单位，所以 M 文件无论从形式、结构和语法规则等方面都比一般的计算机语言简单、易写、易读。另外， MATLAB 本身是用 C 语言写的，M 文件的语法又与 C 语言十分相像，因此熟悉 C 语言的用户可以轻松地掌握 MATLAB 的编程技巧。

5.2 M 文件的形式

M 文件有两种形式：命令文件(Script File)和函数文件(Function File)。这两种文件的扩展名相同，均为“.m”。

当用户要运行的指令较多时，直接从键盘上逐行输入指令显得比较麻烦，而命令文件可以较好地解决这一问题。用户可以将一组相关命令编辑在同一个 ASCII 码命令文件中，运行时只需输入文件名字，MATLAB 就会自动按顺序执行文件中的命令。

函数文件是另一种形式的 M 文件，它的第一句可执行语句是以 function 引导的定义语句。在函数文件中的变量都是局部变量。

5.2.1 命令文件

命令文件中的语句可以访问 MATLAB 工作区(Workspace)中的所有数据。运行过程中，产生的所有变量均是全局变量。这些变量一旦生成，就一直保存在内存空间中，除非用户运用 Clear 命令将它们清除。

运行一个命令文件等价于从指令窗(Command Window)中按顺序连续运行文件里的指令。由于命令文件只是一串指令的集合，因此程序不需要预先定义，而只需按在指令窗中的指令输入顺序将指令编辑在命令文件中就可以。注意，不要忘记文件扩展名“.m”。

【例 1】绘制一幅“花瓣图案”

(1) 用记事本(Notebook)编写以下内容

```
% 一个用以绘制花瓣(flower petal)图案的命令文件
theta = -pi:0.01:pi;
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,:))
pause
end
```

(2) 选择 File 菜单中的 Save 命令，将所写文件保存在磁盘中，并命名为 petal.m。

(3) 在指令窗中输入文件名 petal，运行结束后可看到一幅花瓣图案如图 5.1 所示。

说明：

- 符号“%”引导的是注释行，不予执行。
- 不需要用 end 作为 M 文件结束的标志。
- 若用户把文件 petal.m 存放在自己的工作目录(假如名为 d:\mywork)上，那么在运行 petal.m 之前，应该先使 d:\mywork 处于 MATLAB 的搜索路径上。最简单的方法是在 MATLAB 指令窗中先运行 cd d:\mywork，或按照第 1 章介绍的方法将其加到 MATLAB 的搜索路径中去。

petal 运行后存放在内存的变量，可以用 whos 指令看到。

对于本程序，一幅图案出现后再按 Enter 键，可以看到下一幅图案。图 5.1 给出的是第一幅图案。

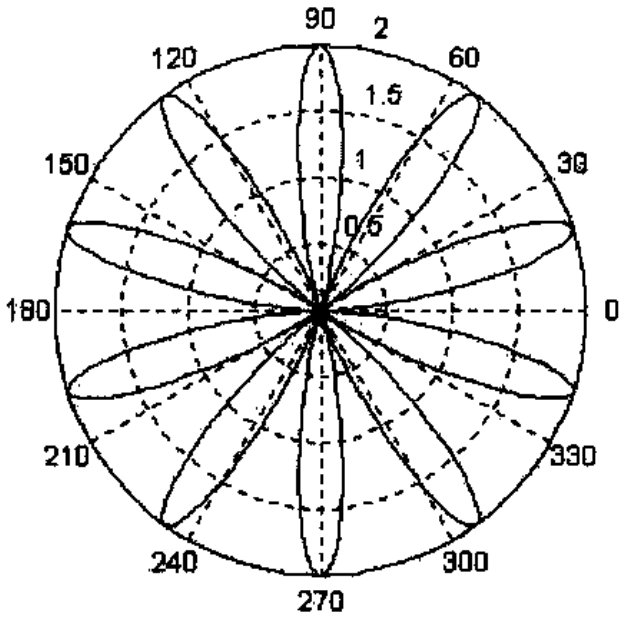


图 5.1 花瓣图案

5.2.2 函数文件

如果 M 文件的第一行包含 function，此文件就是函数文件。每一个函数文件都定义一个函数。事实上，MATLAB 提供的函数指令大部分都是由函数文件定义的，这说明函数文件的重要。从使用的角度看，函数是一个“黑箱”，把一些数据送进并经加工处理，再把结果送出来。从形式上看，函数文件区别于命令文件之处是：命令文件的变量在文件执行完后保存在内存中，而函数文件内定义的变量仅在函数文件内部起作用，当函数文件执行完后，这些内部变量将被清除。

【例 2】计算向量元素的平均值

(1) 在字处理软件中编写以下内容：

```
function y = average(x)
% 向量元素的平均值
% 语法：average(X)，其中，X 为输入向量
% 当输入非向量时，给出错误信息
[m,n] = size(x);
if (~((m==1) | (n==1)) | (m==1 & n==1))
error('Input must be a vector')    % 判断输入是否为向量
end
y = sum(x)/length(x);        % 实际计算过程
```

(2) 将文件 average.m 存盘。该文件定义了名为 average 的新函数。此函数的用法与别的函数一样。

(3) 在指令窗中运行以下指令，可以求得 1~100 的平均值。

```
average(1:100)
ans =
  50.5000
```

说明：

- 第一行执行指令的作用：指明该文件是函数文件；定义函数名、输入参数和输出参数。函数名可以是 MATLAB 中任何合法的字符。输入和输出的参数根据实际需要设置，参数类型可以是数值，也可以是字符串。在本例中，输入参数是向量 x，输出参数是数值型的。在后面将进一步讨论如何进行函数参数的传递。
- 变量 x 对函数文件 average.m 来讲是局部的。当该函数被调用结束后，变量 x 不再存在(如果变量 x 在程序运行前就已经存在的话，程序运行后它不会受到影响)。这一点可以用 who 指令检证。
- 在 M 文件前面，连续几行带符号“%”的注释行有两个作用：一是随 M 文件全部显示或打印时，直接起解释提示作用；二是供 help 指令联机查询用。

【例 3】联机查阅 average 函数的使用说明

(1) 在 MATLAB 指令窗中运行以下 help 指令，可得到帮助信息。

```
help average
% 向量元素的平均值
语法：average(x)，其中 x 为输入向量。
当输入非向量时，给出错误信息
```

(2) 利用 lookfor 指令对关键词进行搜索，获取帮助信息。

```
lookfor average
...(省略部分输出)
average.m: % 向量元素的平均值(average)
...(省略部分输出)
```

说明：

- 该例 help 指令运行后所显示的是 M 文件注释语句中的第一个连续块。至于与第一连续块被空行所隔离的其他注释语句，将被 MATLAB 联机帮助系统忽略。
- 该例 lookfor 指令运行后，显示出 average 函数文件的第一行注释。一般来说，为了利用 MATLAB 对关键词的搜索功能，用户在编制 M 文件时，应在第一行注释中尽可能多地包含该函数的特征信息。
- 为了使 lookfor 指令能对用户目录上的 M 文件的关键词进行搜索，必须使用户目录处在 MATLAB 的搜索路径上，最简单的处理方法是运行指令 path (path,'d:\mywork')。这里，d:\mywork 是假定的用户目录名。或者使用目录管理器，添加所需目录。

5.3 数据结构和全局变量

5.3.1 数据结构

MATLAB 是一种面向矩阵的编程语言，因此它将任何数据都看作是矩阵。在 MATLAB 5.3 版中，数据间的关系如图 5.2 所示。

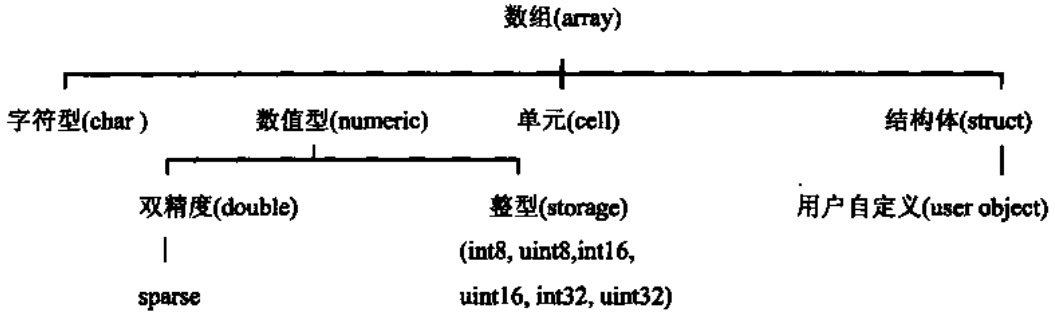


图 5.2 MATLAB 5.3的数据结构

数值变量有整型和双精度型。它们的显示可以用指令窗口中的 File 菜单中 Preference 子菜单下的 General 命令来控制，也可以用指令 format 控制。在程序设计中，不需预定义数据类型，MATLAB 的程序解释器就会自动根据所输入的数据决定数据类型。

【例 4】字符变量的输入和检查

```
a='this is a string';
isstr(a)
ans =
   1
```

说明：

- 指令 isstr 用以检测变量是否为字符型。如果返回值为 1，表示被检测的变量是字符；返回 0，表示被检测的变量是数值。
- 对于其他变量，可以用 class(v)来得到其变量类型。返回值为一个字符型的变量，可能的取值为 cell、char、double、sparse、struct、int8、uint8、int16、uint16, int32 和 uint32。

【例 5】预定义变量，加快运算速度。

```
x=rand(10);
y=zeros(1,2000);
tic;for i=1:2000, z(i)=det(x); end; toc
tic;for i=1:2000, y(i)=det(x); end; toc
elapsed_time =
```

```
    0.5000

elapsed_time =

    0.3300
```

说明：

- 由于 M 文件是面向矩阵的语言，所以 MATLAB 将任何一个变量都看成一个矩阵。如果变量是一个数，就认为是 1×1 的矩阵。通常情况下，变量不需要定义。但是预定义一个变量，可使程序在执行循环结构时速度加快。
- 由于变量没有预定义，所以程序解释器在每次循环中都重新定义 z 的维数。而 y 的预定义，使程序做循环时，省略定义维数这一步。为了仔细比较两者运行时间的区别，这里使用了指令 tic 和 toc 分别计算两个循环的运行时间。显然后者小于前者。
- 循环时间随机器配置不同而不同。

5.3.2 全局变量

全局变量用指令 global 定义，如运行指令 global X Y Z，就会将 X，Y，Z 定义为全局变量。

正如前面所说的，函数文件的内部变量是局部的，与其他函数文件及 MATLAB 内存相互隔离。但是，如果在若干函数中，都把某一变量定义为全局变量，那么这些函数将公用这个变量。全局变量的作用域是整个 MATLAB 的工作区，即全程有效，所有的函数都可以对它们进行存取和修改。因此，定义全局变量是函数之间传递数据的一个手段。

值得指出的是：程序设计中，全局变量固然可带来某些方便，但却破坏了函数对变量的封装，降低了程序的可读性和可靠性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当设计程序较大，子函数较多时，全局变量将给程序调试和维护带来不便，故而不提倡使用全局变量。如果一定要用全局变量，那么最好给它起一个特别的名字，以避免和其他变量混淆。

5.4 程序结构

从理论上讲，只要有顺序、循环和分支 3 种基本程序结构，MATLAB 构成任何一种程序并完成相应的工作。与大多数计算机语言一样，MATLAB 已有设计程序所必需的程序结构：顺序结构；循环结构；分支结构。

在 MATLAB 语言中，循环由 while 和 for 语句实现，分支结构由 if 语句实现。

MATLAB 虽然不像 C 语言那样具有丰富的控制结构，但是 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。MATLAB 语言是一种完善易用的高水平矩阵编程语言。

5.4.1 顺序结构

MATLAB 的顺序结构实际上就是复合表达式构成的语句。复合表达式由分号或逗号隔离的几个表达式构成。当表达式后面接分号时，表达式的计算结果虽不显示，但中间结果仍保留在内存中。若程序是命令文件，则程序运行完后，中间变量都予以保留；若程序是函数文件，那么程序运行完后，中间变量将被全部删除。

5.4.2 循环结构

在很多实际问题中会遇到许多有规律的重复运算，因此在程序中就需要将某些语句重复执行。一组被重复执行的语句称为循环体。它每循环一次，都必须做出是继续重复或是停止的判断，这个判断所依据的条件称为循环的终止条件。MATLAB 语言提供了两种循环方式：for-end 循环和 while-end 循环。

1. for-end循环

for 循环允许一组命令以固定的和预定的次数重复。for 循环的一般形式是：

```
for  x = array
    {commands}
end
```

在 for 和 end 语句之间的{commands}按数组中的每一列执行一次。在每一次迭代中，x 被指定为数组的一列，即在第 n 次循环中，x=array(:, n)。例如：

```
for n=1:10
    x(n)=sin(n*pi/10)
end
x
x=
columns [ through ]
0.3090   0.5878   0.8090   0.9511   1.0000   0.9511   0.8090
columns   8   through  10
0.5878    0.3090    0.0000
```

换句话，第一语句含义是：对 n 等于 1~10，求所有语句的值，直至下一个 end 语句。第一次通过 for 循环 n=1、第二次 n=2，如此继续，直至 n=10。在 n=10 以后，for 循环结束，然后求 end 语句后面的任何命令值。在这种情况下显示所计算的 x 元素。

说明：

- for 循环不能用 For 循环内重新赋值循环变量 n 的方法来终止。
- 语句 1：10 是一个标准的 MATLAB 数组创建语句。在 for 循环内接受任何有效的 MATLAB 数组。
- 当有一个等效的数组方法来解给定的问题时，应避免用 for 循环。例如，上面的

第一个例子可被重写为：

```
n=1:10;
x=sin(n*pi/10);
x =
    Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
0.8090
    Columns 8 through 10
    0.5878    0.3090    0.0000
```

两种方法得出同样的结果，而后者执行更快、更直观，要求较少的输入。

- 为了得到最大的速度，在 for 循环(while 循环)被执行之前，应预先分配数组。例如，前面所考虑的第一种情况，在 for 循环内每执行一次命令，变量 x 增加 1。迫使 MATLAB 每通过一次循环要花费时间对 x 分配更多的内存。为了消去这个步骤，for 循环的例子应重写为

```
x=zeros(1,10);    % 预先为 x 分配内存
for n=1:10
    x(n)=sin(n*pi/10);
end
```

现在，只有 x(n)的值需要改变。

- for 循环可按需要嵌套。

2. while-end循环

for 循环以固定次数求一组命令的值。相反，while 循环以不定的次数求一组语句的值。while 循环的一般形式是：

```
while expression
{commands}
end
```

只要在表达式里的所有元素为真，就执行 while 和 end 语句之间的{commands}。通常，表达式的求值给出一个标量值，但数组值也同样有效。在数组情况下，所得到数组的所有元素必须都为真。

【例 6】计算特殊 MATLAB 值 eps

```
EPS=1;
num=0;
while (1+EPS)>1
EPS=EPS/2;
num=num+1;
end
num
num =
```

```
53
EPS=2*EPS
EPS=
 2.2204e-016
```

说明：

MATLAB 值 eps 是一个可加到 1，而使结果以有限精度大于 1 的最小数值。这个例子表明了计算特殊 MATLAB 值 eps 的一种方法。这里用大写 EPS，因此 MATLAB 的 eps 的值不会被覆盖掉。

在这个例子里，EPS 以 1 开始。只要(1+EPS)>1 为真(非零)，就一直求 while 循环内的命令值。由于 EPS 不断地被 2 除，EPS 逐渐变小以致于 EPS+1 不大于 1(记住，发生这种情况是因为计算机使用固定数的数值来表示数。MATLAB 使用 16 位，因此，只能期望 EPS 接近 10^{-16})。在这一点上，(1+EPS)>1 是假(零)，于是 while 循环结束。最后，EPS 与 2 相乘，因为最后除 2 使 EPS 太小。

5.4.3 分支结构

很多情况下，命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种 if-else-end 结构来提供。最简单的 if-else-end 结构是：

```
if  expression
 {commands}
end
```

如果在表达式中的所有元素为真(非零)，那么就执行 if 和 end 语言之间的{commands}。在表达式包含有几个逻辑子表达式时，即使前一个子表达式决定了表达式的最后逻辑状态，仍要计算所有的子表达式，例如下面的程序段。

【例 7】折扣问题

```
apples=10;          %  apple 的数目
cost=apples*25      %  购买 apple 的费用
cost =
250
if apples>5         %  如果购买量超过 5，给以 20%折扣
cost=(1-20/100)*cost;
end
cost
cost =
200
```

假如有两个选择，if-else-end 结构将是：

```
if  表达式
表达式为真时的指令序列
else
```

```
表达式为假时的指令序列
end
```

在这里，如果表达式为真，则执行第一组命令；如果表达式为假，则执行第二组命令。当有三个或更多的选择时，if-else-end 结构采用以下形式：

```
if 表达式 1
表达式 1 为真时的指令序列
elseif 表达式 2
表达式 2 为真时的指令序列
elseif 表达式 3
表达式 3 为真时的指令序列
elseif …
…
else
所有的表达式均不为真时的指令序列
end
```

注意： 最后的这种形式，只有和所碰到的、与第一个真值表达式相关的命令被执行；接下来的关系表达式不检验，跳过其余的 if-else-end 结构。而且，最后的 else 命令可有可无。

现在知道了如何用 if-else-end 结构来控制循环，就有可能提出一种合理的方法来跳出或中断 for 循环和 while 循环。

【例 8】求 eps 的另一种方法

```
EPS=1;
for num=1:1000
EPS=EPS/2;
if (1+EPS)<=1
EPS=EPS*2
break
end
end
```

结果与上例相同。

说明：

这个例子演示了估算 eps 的另一种方法。在这种情况下，for 循环构造成要执行足够多的次数。if-else-end 结构检验要看 EPS 是否变得足够小。如果是，EPS 乘 2，break 命令强迫 for 循环提早结束，num=53。在这个例子里，当执行 break 语句时，MATLAB 跳到循环外下一个语句。在现在情况下，它返回到 MATLAB 的提示符并显示 EPS。如果一个 break 语句出现在一个嵌套的 for 循环或 while 循环结构里，那么 MATLAB 只跳出 break 所在的那个循环，不跳出整个嵌套结构。

5.5 程序流控制

有关程序流控制的语句和函数，在前面已经见过一些。本节将系统介绍 echo、input、pause、break 和 keyboard 指令。

5.5.1 echo 指令

通常，M 文件执行时，文件的指令不会显示在指令窗口中。用 echo 指令可以使文件指令在执行时可见，这对程序的调试和演示极为有用。对应于命令文件和函数文件，echo 的作用稍微有些不同，参见表 5.1。

表 5.1 echo指令的使用

指令格式	用 途	备 注
echo on	显示其后的所有被执行命令文件的指令	仅用于命令文件
echo off	不显示其后的所有被执行命令文件的指令	
echo	在上面两种状态中切换	
echo FileName on	使 FileName 指定文件的指令在执行中被显示出来	适用于命令文件和函数文件
echo FileName off	终止显示 FileName 文件的执行过程	
echo on all	文件的执行过程是否被显示的切换开关	
echo off all	使其后所有被执行文件的过程不被显示	

注意： 当把 echo 运用于某一函数文件时，该文件将不被编译执行，而是被解释执行。这样，函数文件在执行过程中，每一行都可被观察到。由于这种解释执行不大有效，因而仅用于程序调试。

5.5.2 input 指令

指令 input 提示用户从键盘输入数值、字符串或表达式，并接受该输入。下面是几种常用的格式：

```
a=input('please input a number: ')
```

该指令运行后，将给出如下文字提示，并等待键盘输入：

```
Please input a number:
```

用户可以输入数字或表达式，也可以输入字符串(两端必须有单引号)，按 Enter 键确认后，该输入被赋给变量 a。

```
a=input('please input a string:','s')
```

该指令运行后，也给出提示：

```
Please input a string:
```

等待用户输入。在此提示后输入的任何内容(不管是数字还是字符)一律被当作字符串

赋给变量 a。

注意：　当输入为一个字符串时，可以在输入中使用转义符，这样，“/”之后的字符将和“/”一道，构成新的含义(例如：/n 代表回车)。如果要输入“/”，则要使用“//”。事实上，这与大多数程序设计语言是相同的。

5.5.3　pause 指令

pause 指令使程序运行暂停，等待用户按任意键继续。pause 命令在程序调试以及需要看中间结果时特别有用。pause 的用法有 2 种：pause 暂停执行程序；pause(n)在继续执行前，暂停 n 秒(不同于以前的版本，这里 n 可以不是一个整数，例如 8.7)。

5.5.4　keyboard 指令

keyboard 指令与 input 一样有用。在程序遇到 keyboard 指令时，MATLAB 将会暂停程序的运行，并且调用机器的键盘命令进行处理。一旦处理完自己的工作之后，输入 return，然后按下 Enter 键，程序将继续运行。M 文件中有了它之后，便于在程序调试或在程序执行时修改变量。

5.5.5　break 指令

break 语句导致包含 break 指令的最内层 while、for、if 语句终止。通过使用 break 语句，可以不必等循环的自然结束，而是根据循环内部另设的条件，决定是否退出循环，是否结束 if 语句。在很多情况下，这是必须的。

关于 break 指令，在 5.4.3 节已经举过一个例子(例 3)，这里再举一个例。

【例 9】使用 break 求解数学问题(鸡兔同笼问题)：求 2 个自然数，它们的和等于 100，且第一个数被 2 除的商与第二个数被 4 除的商和为 36。

```
i=1;
while 1
    if rem(100-i*2,4)==0 & (i+(100-i*2)/4)==36
        break;
    end
    i=i+1;
end
a1=i*2
a2=100-i*2
a1 =
44
a2 =
56
```

说明：

在上述第三行中的条件满足后，立即执行 break，使循环终止并跳出循环体。如果没有 break，本循环将无休止地进行下去。

5.5.6 外部系统命令

在 MATLAB 环境中，可以发出 Windows 或 DOS 系统命令，“!”命令能起到这种作用。它使得 MATLAB 将后面的命令传到相应的操作系统。这个过程通常称作使用外部系统命令。MATLAB 5.x 有 4 种形式的外部系统命令，可以根据命令形式的尾部参数区分。

1. 同步实时处理命令

如果在外部系统命令(“！”)之后没有其他附加的参数，那么 MATLAB 会打开一个新的窗口作为该命令的运行窗口。MATLAB 系统要等到该命令完成之后，才开始接受新的命令。例如，!dir 将在其中列出当前目录中的内容。MATLAB 执行完上述命令后，便回到了 MATLAB 的提示符状态，等待新的命令。

2. 后台处理命令

如果外部命令行以字符“&”结束，MATLAB 则将此命令作为后台命令处理，不必等到该命令完成之后，才接受和执行新的 MATLAB 命令。在需要打开新的 Windows 应用程序时，可以使用该命令。例如，发出命令“！notepad &”后，将启动 Notepad 作为一个新的 Windows 任务。类似地，也可以发出 DOS 命令，这时将打开一个 DOS 窗口，但作为后台处理命令，此时可以在 MATLAB 命令窗口中执行其他的命令。

3. 图标后台处理命令

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

4. MATLAB的DOS命令

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

5.6 字符与字符串

MATLAB 有强大的字符处理能力，特别是在引入符号计算(Symbolic Math)工具包以后。本节将介绍字符处理在程序设计中的应用。对于编程语言来讲，字符处理是必不可

少的。

在 MATLAB 中关于字符串有以下几点规则：

- 在 MATLAB 中所有字符串都用单引号界定后输入或赋值。

  如指令 s='Hello'的运行结果是：

```
s=
  Hello
```

- 字符串的每个字符(空格也是字符)都是响应矩阵的一个元素。如上述变量 s 是 1×5 的矩阵，这可用指令 size(s)查得。
- 字符以 ASCII 码储存。用 abs 指令可看到字符的 ASCII 码值。

如运行 abs(s)可得如下结果：

```
abs =
  72   101   108   108   111
```

- 可以用指令 setstr 实现 ASCII 码值向字符的转换。
- 字符变量也可以用方括号合并成更大的“串”。

  例如，运行指令 s=[s,' world']，可得下面结果：

```
s =
   Hello world
```

- 用 eval/feval 函数将字符变量转换为宏功能。eval(t)/feval(t)就是运行包含在 t 中的内容。

【例 10】用 eval 函数产生 5 阶的 Hilbert 矩阵

```
n=5;
t='1/(i+j-1)';
a=zeros(n);
for i=1:n
for j=1:n
a(i,j)=eval(t);
end
end
a
a =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

【例 11】feval 函数的使用

```
fun = ['sin';'cos';'log'];
k = input('Choose function number:');
x = input('Enter value:');
```

```
feval(fun(k,:),x)
```

说明：

feval 与 eval 不同，它以输入变量作为某个函数的函数名来执行，因此，可以使用 feval 和 input 指令从几个 M 文件定义的命令中选择一个。

表 5.2 列出了字符串操作函数。

表 5.2 字符串操作函数

函数名	作 用
isstr	判断是否为字符
blanks	空白字符
deblank	移去空白字符
eval	运行字符串
feval	运行字符串
strcmp	比较字符串
findstr	从一个字符串中寻找是否包含另一个字符串
strrep	用一个字符串代替另一个字符串
upper	将字符串变为大写形式
lower	将字符串变为小写形式
abs	将字符串变为 ASCII 码值
setstr	将 ASCII 码值变为字符串
num2str	将数字变为字符串
str2num	将字符串变为数字
str2mat	将字符串变为文本矩阵
sprintf	将带格式的数字转变为字符串
sscanf	将字符串转变为带格式的数字
hex2num	将十六进制的字符串转变为 IEEE 浮点数
hex2dec	将十六进制的字符串转变为十进制数
dec2hex	将十进制数转变为十六进制的字符串

5.7 函数调用及变量传递

MATLAB 中的函数调用及变量传递比较复杂。但是这两件工作又是编写高质量的 M 文件所不可少的。一个较大的计算任务可以分成若干个较小的任务。这意味着，一个程序可以由若干个函数组成，并通过函数调用来实现控制转移和相互之间的数据传递。

5.7.1　函数调用

在 MATLAB 中，调用函数的常用形式是：

[输出参数 1，输出参数 2，...]=函数名(输入参数 1，输入参数 2，...)

注意： *函数调用时各参数出现的顺序，应该与函数定义时的顺序一样，否则出错。*

函数调用可以嵌套，一个函数可以调用别的函数，甚至调用它自己(递归调用)。

【例 12】给定两个实数 a、b，一个正整数 n，给出 k=1，...，n 时的所有$(a+b)^n$和$(a-b)^n$(本例限定不大于 10)。

(1) 建立函数文件 power.m

```
function[out1,out2]=power(a,b,n)
% power.m 计算(a+b)^n 和(a-b)^n
out1=(a+b)^n;
out2=(a-b)^n;
```

(2) 建立调用上述函数文件的命令 example1.m

```
a=input('Please input a=:');
b=input('Please input b=:');
addpow=zeros(1: 10);
subpow=zeros(1:10);
for k=1:10
   [addpow(k),subpow(k)]=power(a,b,k);
end
addpow
subpow
```

说明：

在本例中，命令文件 example1.m 对函数 power 每做一次调用，传入 3 个参数 a、b 和 k，送出 2 个结果 addpow 和 subpow。

【例 13】用递归调用形式计算 n 的阶乘

(1) 编写递归调用函数文件 factor.m

```
function f=factor(n)
% factor.m  计算 n 的阶乘
if n==1
    f=1;
    return;
else
    f=n*factor(n-1);
    return;
end
```

(2) 运行函数文件

```
factor(6)
ans=
720
```

说明：递归函数是一类算法编程的最直接方式，在程序设计中有重要地位。递归调用可使程序简洁易读。

5.7.2 参数传递

MATLAB 在函数调用上有一个与众不同之处：函数所传递的参数具有可调性。凭借这种特性，一个函数可以完成多种功能。

传递参数数目的可调性来源于如下 2 个 MATLAB 永久变量：

nargin 函数体内的 nargin 给出调用该函数时的输入参数数目

nargout 函数体内的 nargout 给出调用该函数时的输出参数数目

只要在函数文件中包含这 2 个变量，就可以准确地知道该函数文件被调用时的输入参数和输出参数的数目，从而决定如何处理。

【例 14】以 MATLAB 自带的 comet.m 为例说明 nargin 的用法

```
function comet(x, y, p)
if nargin == 0, error('Not enough input arguments.'); end
% 如果没有输入参数，给出错误信息，程序结束
if nargin < 2, y = x; x = 1:length(y); end
if nargin < 3, p = 0.10; end
ax = newplot;
if ~ishold,
  axis([min(x(isfinite(x))) max(x(isfinite(x))) ...
        min(y(isfinite(y))) max(y(isfinite(y)))])
end
co = get(ax,'colororder');
if size(co,1)>=3,
  % 选择彗星的颜色
  head = line('color',co(1,:),'marker','o','erase','xor', ...
              'xdata',x(1),'ydata',y(1));
  body = line('color',co(2,:),'linestyle','-','erase','none', ...
              'xdata',[],'ydata',[]);
  tail = line('color',co(3,:),'linestyle','-','erase','none', ...
              'xdata',[],'ydata',[]);
else
  % 选择彗星的颜色
  head = line('color',co(1,:),'marker','o','erase','xor', ...
```

```
            'xdata',x(1),'ydata',y(1));
    body = line('color',co(1,:),'linestyle','--','erase','none', ...
            'xdata',[],'ydata',[]);
    tail = line('color',co(1,:),'linestyle','-','erase','none', ...
            'xdata',[],'ydata',[]);
end
m = length(x);
k = round(p*m);
% Grow the body
for i = 2:k+1
   j = i-1:i;
   set(head,'xdata',x(i),'ydata',y(i))
   set(body,'xdata',x(j),'ydata',y(j))
   drawnow
end
% 主循环
for i = k+2:m
   j = i-1:i;
   set(head,'xdata',x(i),'ydata',y(i))
   set(body,'xdata',x(j),'ydata',y(j))
   set(tail,'xdata',x(j-k),'ydata',y(j-k))
   drawnow
end
% 清除尾部轨迹
for i = m+1:m+k
   j = i-1:i;
   set(tail,'xdata',x(j-k),'ydata',y(j-k))
   drawnow
end
```

说明：

- 该函数的 3 个输入参数中，向量 x 和 y 定义了彗星的运动轨迹，变量 p 定义了彗星的长度。
- 运行不带参数的命令 comet 时，程序出错，给出错误信息。
- 运行带一个输入参数的命令 comet(y)时，函数在采用 y 向量的同时，再定义相应的 x 和 p。
- 运行带 2 个参数的命令 comet(x,y)时，函数采用默认的 p 值。
- 参数输入的数目可变，但是顺序不可变。

5.8 M文件的调试

从程序设计的角度来讲，MATLAB 语言比其他的程序设计语言在说明结构上要简单得多。但是 MATLAB 有自己严密的语法，用户必须按语法的要求来编写 MATLAB 程序，否则同样会产生一些错误。

许多的语法错误可以在程序执行之前查出，这样的错误也容易处理。由于 MATLAB 是运行解释环境，没有独立的编译过程，所以在程序运行过程中，很可能会碰到在编程时不易发现的错误，而且很难按照运行的过程去追踪出错的地方。原因之一是 MATLAB 函数调用对 MATLAB 的工作区是局部的和封闭的，即使要求 MATLAB 输出中间的计算结果，也很难发现是在什么地方出现错误。

MATLAB 提供了 M 文件的调试功能，可以对 M 文件函数进行调试。MATLAB 的调试功能帮助用户确定 MATLAB 程序代码中的错误，可以在函数运行期间的任何时刻用调试查看 MATLAB 工作区的变量值，查看函数调用的栈管理，以及逐行地运行 M 文件。

调试为用户提供了命令行交互式接口，可以通过命令窗口的菜单进行操作。

5.8.1 调试主要功能

调试主要是分析 MATLAB 程序中的隐含错误，可以发现和更正如下两类错误。

- 语法错误

这类错误主要包括函数名拼写和括号遗漏等错误。当然，这类错误在程序运行时，MATLAB 系统自己可以检测到，并且会指出 M 文件中可能出错的行号。

- 运行错误

这类错误通常是算法错误，而在语法上是正确的。这类错误会导致不正确的计算结果，而 MATLAB 系统不会发现出错的地方，这就必须使用调试。

一般说来，运行错误往往难于跟踪。因为当被调用的函数由于出错而退出函数体时，函数局部工作区的变量就全部消失了。可以用下列技巧发现某些运行错误：

- 在 M 文件中，将某些语句后的分号去掉，迫使 M 文件输出一些中间计算结果，以便发现可能的算法错误；
- 在 M 文件中，加入 keyboard 语句。keyboard 语句可以设置程序的断点；
- 将函数 M 文件改为子程序 M 文件，这样可以在 MATLAB 工作区查看中间计算结果。

最后一种方法是使用调试，调试比其他技巧优越的地方在于它可以深入到函数的工作区，可以设置或清除断点、按行执行程序等。

5.8.2　调试主要命令

MATLAB 的调试命令主要有以下几种，如表 5.3 所示。

表 5.3　调试基本指令

指 令 名 称	指 令 功 能
dbstop	设置程序断点
dbclear/dbclear all	清除程序(所有)断点
dbcont	恢复程序运行至程序结束或到另一个断点
dbdown/dbstep in	深入下层局部工作区
dbstack	列函数调用关系
dbstatus	列出所有断点
dbstep	指定执行步数
dbtype	列出行号内的 M 文件
dbup	向上层改变局部工作区
dbquit	退出调试状态

5.8.3　调试的使用

如果要检测函数 M 文件是否有某些错误，可以对该函数进行调试。在函数中设置断点，帮助分析和发现程序的错误原因。在调试时，若函数子程序遇到断点，则将子程序暂时停下来，并将断点处的命令行显示出来，同时显示键盘输入的提示符。用户可以在该提示符之后输入任何合法的 MATLAB 命令。

使用 MATLAB 的调试命令时，应该记住以下几点：

- 调试命令只能对函数 M 文件使用，不能对其他的 M 文件使用。
- M 文件的断点是与编译过的 M 文件相关联的，如果 M 文件被清除或是被重新编辑，那么所有的断点即被取消。

5.8.4　GUI 界面的调试

MATLAB 5.x 版提供了一个基于 GUI 界面的调试。使用它，可以对函数进行调试。它的基本界面如图 5.3 所示。它的使用比较简单，下面仅仅介绍一些基本功能，并通过一个实例帮助用户掌握。

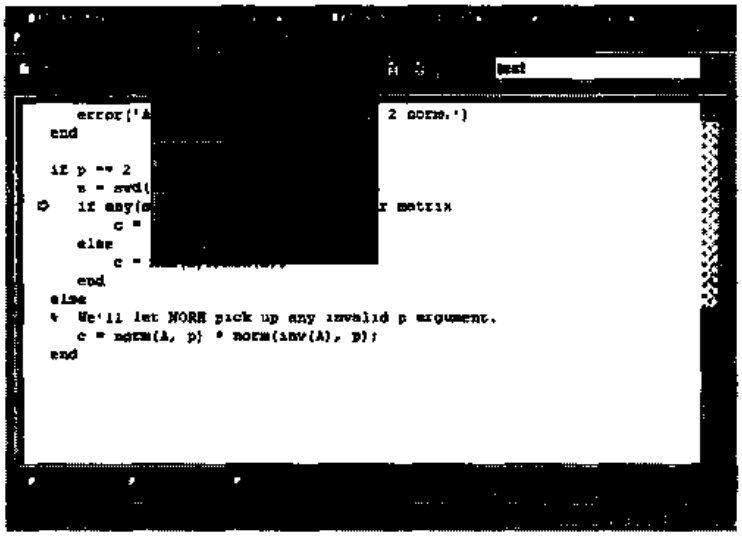


图 5.3 调试界面

1. 调试的使用

下面仅仅介绍 Debug 菜单和工具栏上部分按钮的使用，参见表 5.4 和表 5.5。

表 5.4 Debug菜单的使用

菜单命令名称	意 义	对应的调试命令
Continue	恢复程序运行至结束或另一个断点	Dbcont
Single Step	单步执行函数	Dbstep
Step In	深入下层局部工作区	dbdown/dbstep in
Quit Debugging	退出调试状态	Dbquit
Set/Clear Breakpoint	设置/清除光标处的断点	dbstop/dbclear
Clear All Breakpoints	清除程序中的所有断点	dbclear all
Stop if Error	运行至出错或结束	dbstop if error
Stop if Warning	运行至警告消息或结束	dbstop if warning
Stop if NaN of Inf	运行至运算结果出现 NaN 或 Inf	dbstop if naninf

表 5.5 调试工具按钮的使用

图 标	功 能	对应的 Debug 菜单命令
	设置/清除光标处的断点	Set/Clear Breakpoint
	清除程序中的所有断点	Clear All Breakpoints
	深入下层局部工作区	Step In
	单步执行函数	Single Step
	恢复程序运行至结束或另一个断点	Continue
	退出调试状态	Quit Debugging

2. 实例

编写函数 M 文件并对其进行调试。

(1) 函数 M 文件的编写

新建 2 个 M 文件 test.m 和 test1.m，准备开始调试。

```
test.m
  function a=test(b)
  c=sqrt(b)*cos(b)
  a=test1(b,c)
test1.m
  function a=test1(b,c)
  q=cond(b)
  [w,e]=eig(c)
  a=w*q
```

(2) 设置断点

确认将文件放在搜索路径下，在 MATLAB 主窗口中输入：

```
edit cond
edit test
edit test1
```

在 test.m 的第 2 行(c=sqrt(b)*cos(b))、cond.m 的第 36 行(s = svd(A);)设置断点。

(3) 启动 M 文件及查看堆栈

在 MATLAB 的主窗口中输入：

```
test(magic(4))
```

可以看到，程序停在 test.m 的第 2 行，并且工具栏最右边的列表栏中显示目前所在的工作区为 test/Base Workspace。说明这时还没有进行其他的调用过程。对于这一点，可以在 MATLAB 主窗口中输入 dbstack 指令来验证。

选择 Debug 菜单的 Continue 命令，继续运行至 cond.m 的第 36 行停下。这时可以看到，工作区按 cond/test1/test/Base Workspace 来排列的，而这正是调用的先后顺序。

回到 MATLAB 主窗口，输入 who，可知当前变量为

```
Your variables are:
A       m       n       p
```

输入 A，可以看到变量 A 的内容。这里，A 是一个由 magic(4)产生的魔方阵。

```
A =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
```

选择 Debug 菜单的 Single Step 命令，执行第 36 行，并且停在第 37 行。

```
if any(s == 0)   % Handle singular matrix
```

```
检查变量 s，为
s =
   34.0000   17.8885    4.4721    0.0000
```

这时，再次查看 cond 的工作区，发现新增加了变量 s。

```
who
Your variables are:
A        m        n        p        s
```

(4) 改变工作区

从列表栏中选择相应的名称，就可以改变函数所处的工作区，相当于 dbup 和 dbdown 的组合。对应于每一个工作区，都可以查看、修改变量内容。

现在，从列表栏中选择 Base Workspace，将工作区转到 MATLAB 的工作区，即调用函数 test.m 的工作区。

逐步执行函数，直至 cond.m 函数结束，转到 test1.m。在此过程中，可以看到工作区和变量的作用域的改变。

如果没有发现错误，可以退出调试状态。

注意：
- dbquit 不清除断点，在使用 dbquit 之前，必须先用 dbclear 命令清除所有断点。
- 当然，也可以用对应的菜单命令完成这项工作。

第 6 章
MATLAB 中的计算结果可视化

本章要点：

图形和可视化是当代应用软件发展的主要方向。随着 MATLAB 版本的不断提高，MATLAB 的图形功能也越来越强。本章主要介绍 MATLAB 的高层图形命令。通过绘图，有助于理解结果，直观地反映问题。

本章具体包括以下内容：

- 二维曲线图形
- 三维曲面图形
- 四维表现和切片图
- 图形的标注

人们很难直接从一大堆原始的离散数据中感受到它们的含义；数据图形恰能使人们直接感受到数据的许多内在本质。因此，数据可视化是人们研究科学、认识世界不可缺少的手段。

MATLAB 可以给计算数据以二维、三维乃至四维的图形表现。通过对图形线型、立面、色彩、渲染、光线、视角等属性的处理，可把计算数据的特征表现得淋漓尽致。

MATLAB 图形系统的这种能力是建立在一组“图形对象”(Graphics Objects)基础之上的。它的核心是图形的句柄(Graphic Handle)操作。MATLAB 有两个层次的绘图指令：一组是直接对句柄进行操作的底层(Low-Level)绘图指令；另一组是在底层指令基础上建立起来的高层(High-Level)绘图指令。

高层绘图指令简单明了。它不仅容易为用户所掌握，而且也是今后最常用的。本章将主要介绍高层绘图指令的作用和调用格式。

6.1 二维曲线图形

6.1.1 基本绘图指令 plot

MATLAB 函数 plot 是最简单而且使用最广泛的一个线型绘图函数。利用它可以生成线段、曲线和参数方程曲线的函数图形。以下详细介绍 plot 的几种基本命令形式。

1. 向量式plot(v)

这是最简单的一种调用方式，其中，v 是长度为 n 的数值向量，它的作用是在坐标系中顺序地用直线段连接顶点{(i，v(i)) (i=1，…，n)}，生成一条折(曲)线。坐标系的范围由 MATLAB 系统根据向量的长度及向量元素的大小自动生成(下同)。当向量的元素充分多时，即可以得到一条外观光滑的曲线。

2. 参数式plot(x, y)

在 plot(x,y)中，参数 x 和 y 都是长度为 n 的向量，它的作用是在坐标系中生成顺序连接顶点{(x(i),y(i))}(i=1，…，n)的折(曲)线。这种调用可以用来生成参数方程的图形。

【例 1】绘制 y=sin(x)在一个周期内的图形(见图 6.1)

```
t = 0:pi/100:2*pi;
y = sin(t);
plot(t,y)
grid on
```

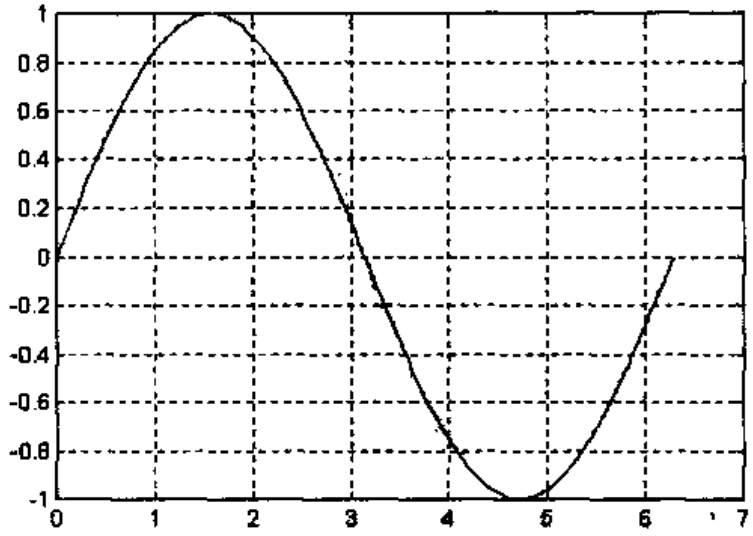


图 6.1　参数方程的图形

3. 矩阵式plot(Y)

在 plot(Y)中，Y 是一个 m×n 的矩阵。MATLAB 为矩阵的每一列画出一条线，同时以矩阵的行向量为基准对 x 轴进行分度和标注。标注时，采用向量 1:m，这里 m 是矩阵的行数。

【例 2】矩阵图形的绘制

```
Z = peaks;                    % 产生一个 49×49 的矩阵
plot(Z)                       % 绘图
```

这个例子比较简单，就不再给出图形了。

4. 混合式plot(X, Y)

在 plot(X,Y)中，如果 X 和 Y 都是向量，则长度必须相等，亦即参数式；如果 X 是向量，而 Y 是一个矩阵，X 的长度与矩阵 Y 的行数或列数相等，则它的作用是将向量 X 与矩阵 Y 的每列或每行的向量相对应作折(曲)线，当 Y 是方阵时，则将向量 x 与矩阵 Y 的列向量相对应作图；如果 X 是矩阵，Y 是向量，Y 的长度等于 X 的行数或列数，则将 X 的每列或每行的向量与 Y 相对应作图。同样，当 X 是方阵时，则将 X 的各列与 Y 相对应作图；如果 X 和 Y 都是矩阵，且维数相同，那么按列与列的对应方式来作图。可以参看下面这个例子。

【例 3】混合式图形的绘制

```
y = 1:length(peaks);
plot(peaks,y);
```

上述语句生成的图形如图 6.2 所示。

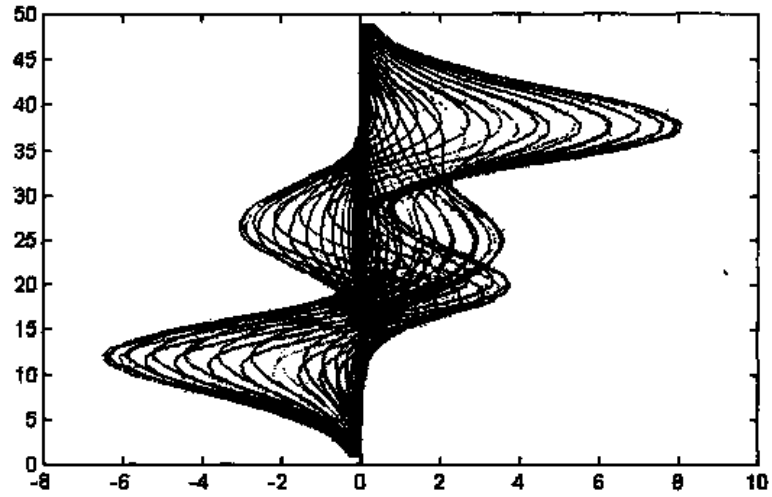


图 6.2 矩阵对向量的图形

5. 复向量式plot(Z)

当参量为一个复向量时，MATLAB 将会忽略向量的虚部，除非函数 plot 在调用时单独给出一个复参数。这时，一个指令相当于是两个指令的组合。例如，plot(Z)和plot(real(Z),imag(Z))是等效的(这里，Z 为一个复向量)。

为了说明这一点，下面举一个例子。

【例 4】随机矩阵特征值分布(见图 6.3)

```
plot(eig(randn(20,20)),'o','MarkerSize',6)
```

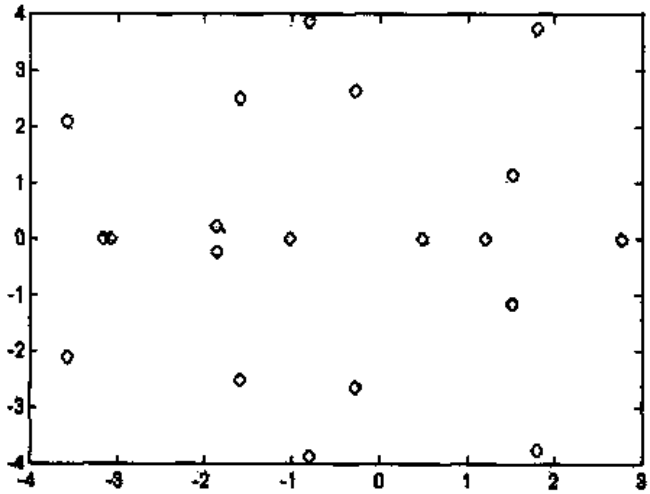


图 6.3 特征值分布

6. 综合调用方式

一般地，函数 plot 有多个矩阵对的调用方式，即

```
plot(X1,Y1,X2,Y2,…)
```

这时，每一对矩阵(Xi,Yi)均是按照前面的 4 种方式之一进行解释，不同的矩阵对之间，其维数可以不同。

6.1.2　线型、顶点标记和颜色

前面介绍了 plot 函数的多种调用方式(准确地说应该是 5 种)。使用这几种调用方式时，MATLAB 自动安排作图的线型和线段的颜色，包括线段顶点的标记。事实上，MATLAB 的 plot 函数可以设置和管理曲线的线段类型、顶点标记和线段颜色(在 MATLAB 中，它们通称为线型—Line Style，在本书中为了方便加以区分)。

MATLAB 定义的线段类型、顶点标记和线段颜色如表 6.1 所示。

表 6.1　线段、颜色与顶点标记

颜　色		线型顶点标记		顶点标记	
类型	符号	类型	符号	类型	符号
黄色	y	实线	-	实点标记	.
洋红色	m	点线	:	圆圈标记	o
蛋青色	c	点虚线	-.	乘号标记×	×
红色	r	虚线	--	加号标记+	+
绿色	g			星号标记*	*
蓝色	b			方块标记□	s
白色	w			钻石形标记◇	d
黑色	k			向下的三角形标记	V
				向上的三角形标记	^
				向左的三角形标记	<
				向右的三角形标记	>
				五角星标记☆	p
				六边形标记⬡	h

在 plot 函数中，可以传递一个类型参数，类型参数是字符串，由表 6.1 中列出的符号组成。这样可以控制线段类型、顶点标记和线段颜色。plot 的最典型的调用方式是所谓的三元组参数，即

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

其中，X 为横坐标矩阵，Y 为纵坐标矩阵，S 为类型说明字符串参数。例如 plot(x, y,'m^') 在所在顶点处(x(i),y(i))标上洋红色的三角符号标记。要注意的是，S 字符串可以是三种类型的符号之一，也可以是线型与颜色或顶点标记与颜色的组合。但是，三种类型的符号不能同时出现在 S 中。正如 MATLAB 所区分的那样，顶点标记是一种特殊线型。只在顶点处画标记，没有顶点间的连接线。例如，设 x 和 y 是长度相同的向量，要画出一条红色的虚线，而且在顶点上用黄色的圆圈标记，则可用下面的语句来实现。

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

这个例子表明 plot 函数可以有如下更一般的调用形式。

```
plot(X1,Y1,S1,X2,Y2,S2,…)
```

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

通常，在当前坐标系中绘图时，每调同一次绘图函数，如调用 plot 时，MATLAB 将擦掉坐标系中已有的图形对象。为了在一个坐际系中增加新的图形对象，可以用 MATLAB 的 hold 命令达到这个目的。在设置了 hold on 后，MATLAB 在生成新的图形时保留当前坐标系中已存在的图形对象。此时，MATLAB 根据新图形的大小，可能会重新改变坐标系的比例。

【例 5】hold on 命令的使用

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

上述程序将在同一个坐标系画出 3 条曲线，使用 MATLAB 默认的颜色设置，如图 6.4 所示。当然，下面的语句也可以达到同样的目的。

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

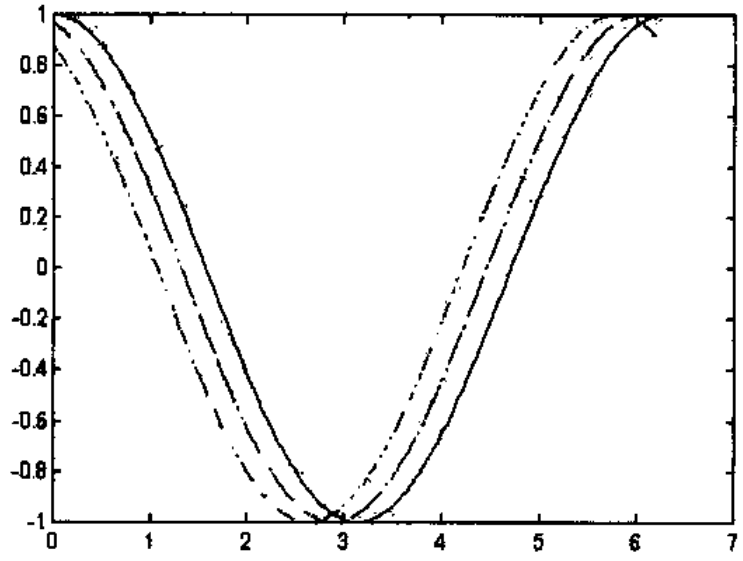


图 6.4 加入新图形

6.1.3　二维特殊图形

MATLAB 5.3 提供了很多绘制二维图形的指令，详见表 6.2。

表 6.2　特殊二维图形指令

函数名称	功　能
area	填充的函数折(曲)线图
bar	直方图
barh	垂直的直方图
bar3	三维直方图
bar3h	垂直的三维直方图
comet	彗星轨迹状的图形
errorbar	误差棒图
ezplot	符号函数二维曲线
feather	沿 x 轴分布的复数向量图
fill	平面多边形填色
fplot	数值函数二维曲线
hist	向量的统计直方图
pareto	带有标准的直方图
pie	饼图
pie3	三维饼图
plotmatrix	矩阵折(曲)线图
ribbon	带状图
scatter	点图(与 plot 的绘制结果相似，但是只有数据点)
stem	火柴杆图
stairs	阶梯图

【例 6】用 bar 函数绘制向量 y 的直方图(见图 6.5)

```
x=0:pi/10:2*pi;
y=cos(x);
bar(y);
x=0:pi/10:2*pi
y=cos(x);
bar(x,y);
```

说明：

- 如果使用指令 bar(y)，则取 y 元素的下标作为 x 坐标值。
- 本指令可像 plot 一样，可以加入指令开关，选择线型和颜色。

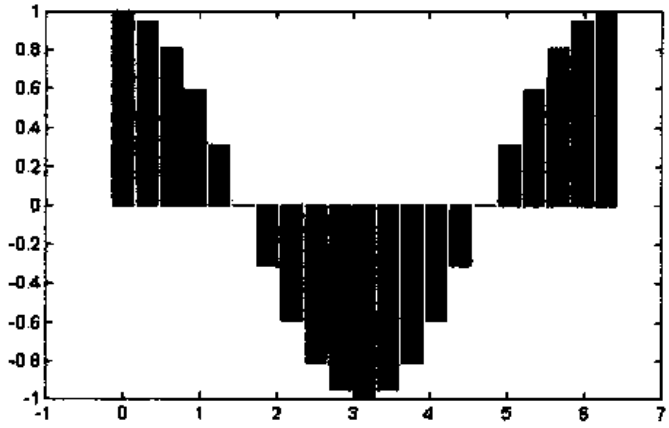


图 6.5 直方图

【例 7】用 errorbar(x,y,e)绘制误差棒图(见图 6.6)

```
x = 1:10;
y = sin(x);
e = std(y)*ones(size(x));
errorbar(x,y,e)
```

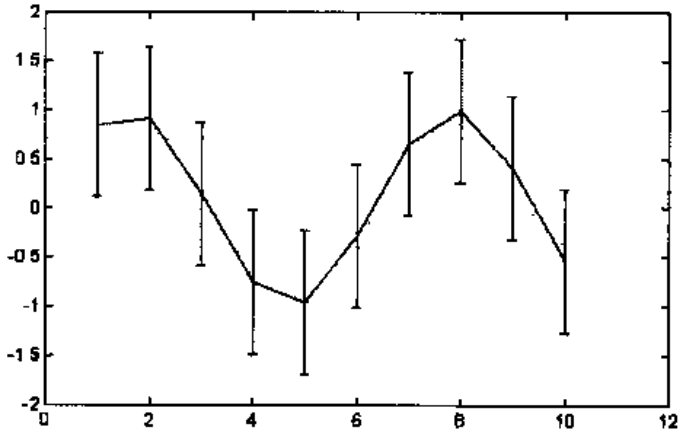


图 6.6 误差棒图

说明：

- x，y，e 必须是同维。
- 该指令常用于数理统计离散数据有理化。

6.1.4 绘制数值函数二维曲线的专用指令

前面所介绍的绘图指令在绘制一个函数 y=f(x)的图形时，必须先定义自变量调的一组取值点，再求出这组取值点上的函数值，然后根据这两组数值确定的数据点绘制出所需的图形。

本节要介绍绘制函数 y=f(x)图形的一个专用指令：fplot。该指令的特点在于它的绘图数据点是自适应产生的。在函数曲线平坦处，它所取数据点比较稀疏；在函数变化剧烈处，它将自动取较密的数据点。因而，对于那些导数变化较大的函数，用 fplot 所绘出的曲线比等分取点所画出的曲线更加接近真实。fplot 的用法如下：

```
[X,Y]=fplot(fun,lims,tol,n,'linespec',p1,p2,...)
```

其中各项的含义是：

fun　函数名称字符串，它可以是一个由多个分量函数构成的函数行向量。分量函数可以是 MATLAB 已有函数，也可以是用户自定义函数。

lims　定义 x 的取值区间，lims=[xmin,xmax]。

tol　相对误差，默认值为 2e−3

n　绘图的最少点数(n+1)

'linespec'　线型设置

p1,p2,..　函数传递的参数

X，Y　输出数据点坐标

注意：●如果按照上述函数形式调用，则将会把数据点坐标输入 X、Y，并没有图形显示出来。如果要得到图形，可以不加这两个参数。

●如果要使用 tol、n 或是'linespec'的默认参数，可以给函数传递一个空矩阵作为参数。

●在相同数据点数下，自适应取点所绘的图更真实。

●自适应绘图所需的时间较长。

【例 8】fplot 与一般绘图指令的绘图效果比较

(1) 横坐标自适应取点绘图

为比较需要，自适应绘图分两步进行。

```
[X,Y]=fplot('cos(tan(pi*x))',[-.4,1.4],0,2e-3);
n=length(X)
plot(X,Y);
n=
  274
```

绘制的图形如图 6.7 所示。

(2) 横坐标等分取点绘图

```
t=(-.4:1.8/n:1.4)';
n=length(t)
plot(t,cos(tan(pi*t)));
```

绘制的图形如图 6.8 所示。

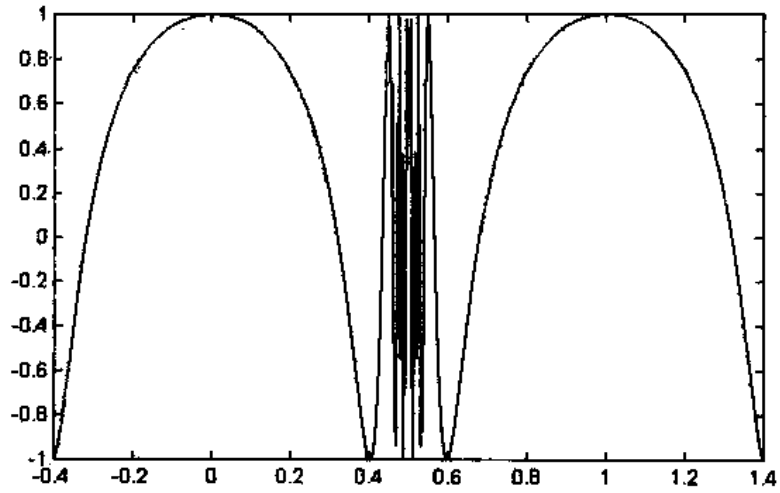


图 6.7 横坐标自适应取点绘图

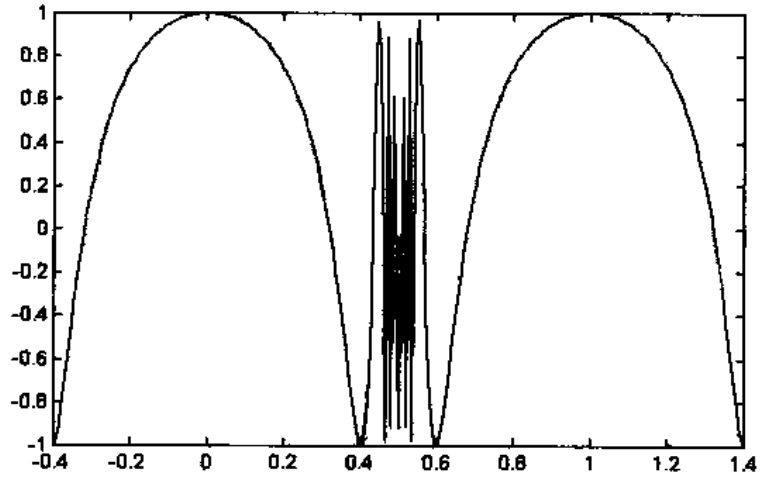


图 6.8 横坐标等分取点绘图

6.1.5 一个窗口中多个图形的绘制

在实际应用中，常常会需要在一个窗口中绘制多个图形，以便对几个函数进行直观、便捷地比较。这就要用到 subplot 指令。

subplot 指令不是仅用于二维图形，它对三维图形一样适用。其本质是将 Figure 窗口分为几个区域，再在每个区域内分别绘图，这是一个很有用的指令。

subplot 函数的用法：

h = subplot(m,n,p)或是 subplot(mnp)

它将图形窗口分成一个 m×n 的矩阵，每个小块使用自己的坐标系。同时指定第 p 块区域为活动区域，将随后的一条绘图指令输出的图形显示在这块区域上。p 是根据从上到下、从左到右的顺序来规定的。例如，subplot(2，2，3)，将窗口分为 4 块，选择第三块(即是左下角的一块)作为绘图区。函数调用成功后，返回这块区域的图形句柄值，保存在 h 中。

注意： 对于指令 subplot(m,n,p)，如果坐标系存在，则将其设为当前坐标系。也可以用 subplot(h)，其中 h 为坐标系的图形句柄来实现同样的功能。

说明：

- 指令 subplot('position',[left bottom width height]) 在普通坐标系中创建一个新的坐标系。其中，各个参量在 0 到 1 之间取值。
- 如果 subplot 指令所指定的区域与原有的区域重合(全部或部分)，原区域将被删除。例如，使用 subplot(1,2,1)将会删除图形窗口左边的所有已经存在的坐标系，并创建一个新的。
- subplot(111) 是一个特殊的情况，它与 subplot(1,1,1)不同。该调用并不立刻创建坐标系，而是使得下一条绘图指令在窗口中执行 clf 和 reset 指令(删除当前图形的所有子对象)，然后在默认位置创建一个坐标系。这种调用没有返回值。

【例 9】subplot 指令示例

下面一段程序，将图形窗口分为 4 个区域，在每个区域中，采用横坐标自适应取点绘图绘制图形。

```
subplot(2,2,1), fplot('humps',[0 1])
subplot(2,2,2), fplot('abs(exp(-j*x*(0:9))*ones(10,1))',[0 2*pi])
subplot(2,2,3), fplot('[tan(x),sin(x),cos(x)]',2*pi*[-1 1 -1 1])
subplot(2,2,4), fplot('sin(1 ./ x)', [0.01 0.1],1e-3)
```

所得的图形如图 6.9 所示。

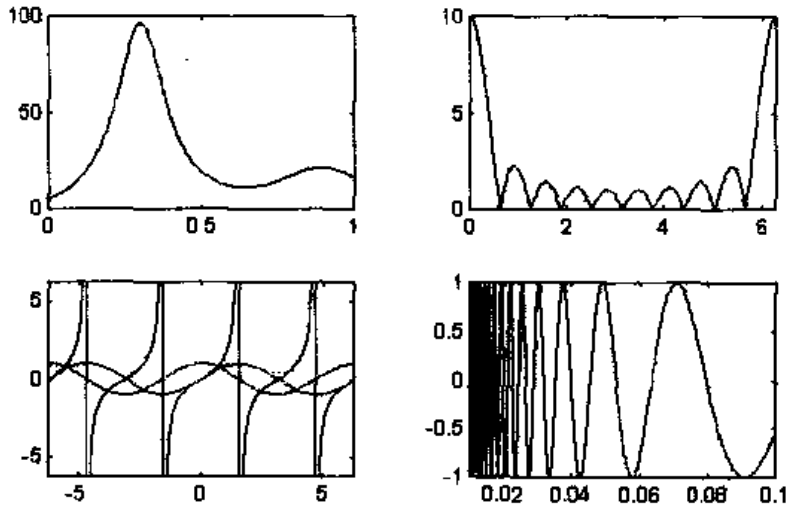


图 6.9　subplot指令的使用

6.2　三维曲面图形

MATLAB 具有强大的三维图形处理功能，包括三维数据显示、空间曲线、曲面、分块、填充以及曲面光顺着色、视点变换、旋转、隐藏等功能和操作。本节将对上面的功能进行介绍。

6.2.1 三维线性图形

plot3 命令将绘制二维图形的函数 plot 的特性扩展到三维空间。函数格式除了包括第三维的信息(如 Z 方向)之外，其他与二维函数 plot 相同。plot3 一般语法调用格式是 plot3(x1,y1,z1,S1,x2,y2,z2,S2,…)，这里 xn、yn 和 zn 是向量或矩阵，Sn 是可选的字符串，用来指定颜色、标记符号和线形。如果省略，MATLAB 将会按照默认值给出设置。

总的来说，plot3 可用来画一个单变量的三维函数，如下例所示。

【例 10】绘制一个三维螺旋线

```
t=0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
title('Helix'),xlabel('sint(t)'),ylabel('cos(t)'),zlabel('t')
text(0,0,0, 'Origin' )  % 图形标注命令
grid                    % 给图形加上网格
v = axis
v =
 -1     1    -1     1     0    40
```

输出的结果如图 6.10 所示。

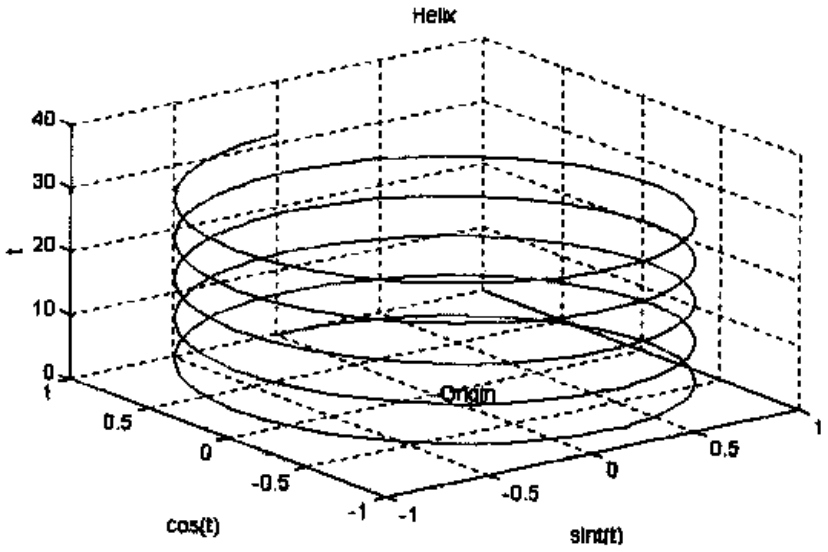


图 6.10 螺旋线图

从上例可明显看出，二维图形的所有基本特性在三维中仍都存在。axis 命令扩展到三维只是返回 Z 轴界限(0 和 40)，在数轴向量中增加两个元素。函数 zlabel 用来指定 z 轴的数据名称，函数 grid 在图底绘制三维网格；函数 text(x，y，z，string)在由三维坐标 x,y,z 所指定的位置放一个字符串。另外，子图和多图形窗口可以直接应用到三维图形中。

通过指定 plot 命令的多个参量或使用 hold 命令，可以把多条直线或曲线重叠画出。plot3 以及其他的三维图形函数都可以提供相同的能力。例如，增加维数的 plot3 命令可以使多个二维图形沿一个轴排列起来，而不是直接将二维图形叠到另一个的上面。

【例 11】二维图形在三维空间的排列

```
x=linspace(0,3*pi);     %  x 轴的数据
z1=sin(x);              %  在 x-z 平面内画图
z2=sin(2*x);
z3=sin(3*x);
y1=zeros(size(x));
y3=zeros(size(x));      %  变量的预定义
y2=y3/2;
plot3(x,y1,z1,x,y2,z2,x,y3,z3);
grid,xlabel( 'x-axis' ),ylabel( 'y-axis' ) , zlabel('z-axis' )
title('sin(x),sin(2x),sin(3x) ' )
```

输出的结果如图 6.11 所示

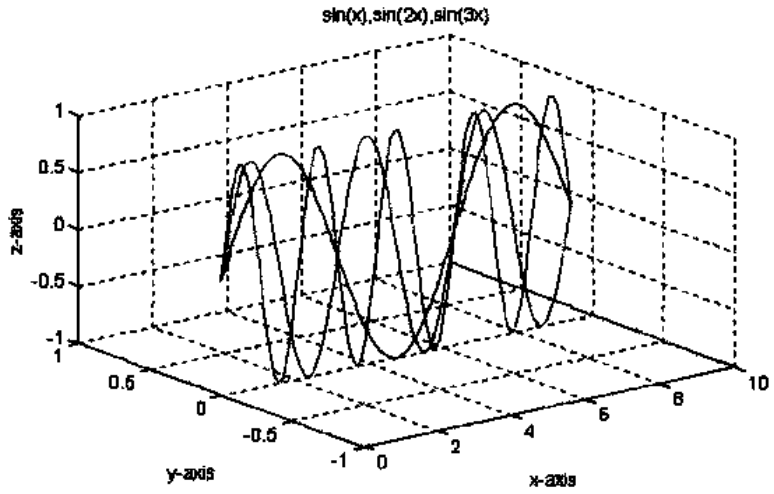


图 6.11　正弦曲线图

6.2.2　三维曲面

1.　平面网格点的生成

在数学上，函数 z=f(x,y)的图形是三维空间的曲面，在 MATLAB 中，总是假设函数 z=f(x,y)是定义在一个矩形的区域 D=[x0,xm]×[y0,yn]上的。为了绘制在区域 D 上的三维曲面，MATLAB 的方法是首先将[x0,xm]在 x 方向分成 m 份，将[y0,yn] 在 y 方向分成 n 份，由各分划点分别作平行于坐标轴的直线，将区域 D 分成 m×n 个小矩形块，计算出在网格点的函数值。对于每个小矩形，在空间中决定出 4 个顶点(xi,yi,f(xi,yi))，连接 4 个顶点得到一个空间中的四边形片。所有这些四边形片一起构成函数 z=f(x，y)定义在区域 D 上的空间网格曲面。

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

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

或是

```
{x,Y}=meshgrid(x)
```

第二种形式等价于 meshgrid(x,x)。这里，x 是区间[x0,xm]上分划点组成的向量，而 y 是区间[y0,yn]上分划点组成的向量。输出变量 X 和 Y 都是矩阵，矩阵 X 的行向量都是向量 x，Y 的列向量都是向量 y。于是，X 和 Y 的元素组(X(i，j)，Y(i，j))恰好是区域 D 的第(i,j)网格顶点。例如，(X(1，1)，Y(1，1))对应于(x0，y0)，而(X(m+1，n+1)，Y(m+1，n+1))对应于(xm，yn)。换句话说，函数 meshgrid 将由两个向量决定的区域转换成对应的网格点矩阵。

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

【例 12】数学函数 $z=\sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$，定义在区域[-8,8]×[-8,8]上。在生成网格点后，计算网格点上的函数值

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

注意： 由于在邻近原点处，R 的某些元素可能会很小，因此加入 eps 可以避免出现零为除数。

2. 网格曲面

在得到了网格点上的函数值矩阵后，即可以用 MATLAB 函数 mesh 来生成函数的网格曲面，即各网格线段组成的曲面。例如对上述的例子，下列语句生成的图形如图 6.12 所示。

```
mesh(Z)
```

函数 mesh 有多种调用形式，在图 6.12 中，x 轴和 y 轴是按矩阵 Z 的行列序号定义的。为了使其与该函数的定义域一致，需要使用其他的调用方式。下面分别给予介绍。

- mesh(X，Y，Z，C)，这是最一般的调用形式。X、Y、Z、C 是同维数的矩阵，C 称为颜色矩阵。网格曲面的顶点对应于空间的顶点(X(i，j)，Y(i，j)，Z(i，j))，而网格曲面的网格线的颜色由 C 值根据当前的色谱来着色。这种调用形式还可以用来生成参数曲面片.
- mesh(X，Y，Z)，这是在调用形式(1)中，取 C=Z 的简单调用形式。
- mesh(x，y，Z，C)，其中，x 和 y 是向量，Z 和 C 是同维数的矩阵，且向量 x 的长度等于矩阵 Z 的列数，而向量 y 的长度等于矩阵 Z 的行数。在这种情况下，网格曲面的网格顶点是(x(j)，y(i)，Z(i，j))，网格线的颜色由矩阵 C 决定。
- mesh(x，y，Z)，这是在调用形式(3)中，取 C=Z 的简单调用形式。
- mesh(Z，C)，Z 和 C 都是 m×n 的矩阵，该形式与 mesh(x，y，Z，C)等价，其中

向量 x=1:n，向量 y=1:m。

- mesh(Z)，这是在调用形式 5 中，取 C=Z 的简单调用形式。

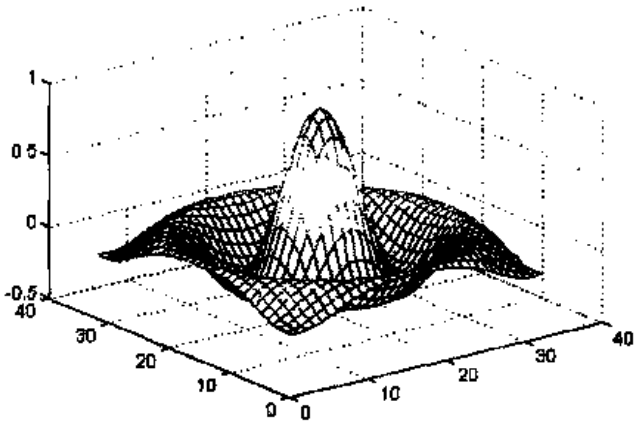
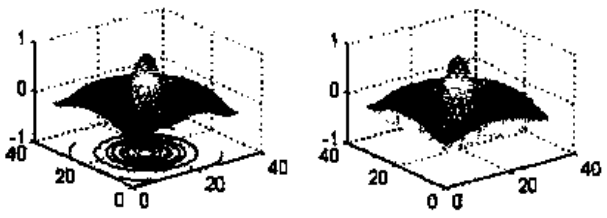


图 6.12　网格曲面

与 mesh 相关的另外两个函数是 meshc 和 meshz，它们的调用形式与 mesh 相同。meshc 除了生成网格曲面外，还在 x - y 平面上生成曲面的等高线图形，如图 6.13a 所示。而函数 meshz 的作用除了生成与 mesh 相同的网格曲面之外，还在曲面下面加上一个长方体的台柱，使图形更加美观，如图 6.13b 所示。

【例 13】meshc 和 meshz 的演示(接例 3)

```
subplot(2,2,1); meshc(Z)
subplot(2,2,2);meshz(Z)
```



a 曲面与等高线　　b 曲面与台柱

图 6.13　meshc与meshz生成的图形

3. 实曲面的绘制

实曲面就是对网格曲面的网格块(四边形片)区域进行着色的结果。MATLAB 的函数 surf 可提供这种功能。它的调用方式与 mesh 相同，这里不再重复。

surf 的曲面生成过程与 mesh 相似，但着色机制与 mesh 不同。mesh 仅对网格线着色，而 surf 是对网格片着色，而网格线用黑色标出(可以修改)。通常，surf 用默认的着色方式对曲面片着色。此外，还可以用 MATLAB 的函数 shading 来改变着色方式。例如：

```
shading faceted
```

这是默认的着色方式，网格线为黑色。而

```
shading flat
```

与 faceted 模式类似，只是网格线分块着色。又如：

```
shading interp
```

最后一个命令的着色光顺性最好，效果如图 6.14 所示。

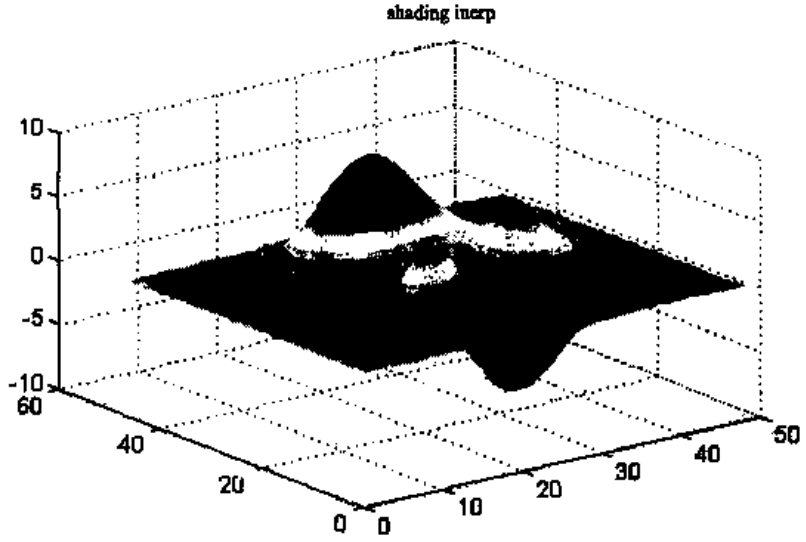


图 6.14 着色的曲面

网格块的内部像素的颜色由该片的 4 个顶点的颜色值做双线性插值得出。另一个很重要的函数是 surfl，它能生成具有光照效应的表面，使图形更加美观。除了与 surf 具有相同的输入参数之外，还可以指定三维坐标系的光源点坐标[x1，y1，z1]，或光源方向的球面坐标值，即经度和纬度(仰角)向量[azimuth,elevation]。

【例 14】用 surfl 指令创建一个图形，其中光源的方向为经度=−10°，纬度=50°

程序为：

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

生成的图形如图 6.15 所示。

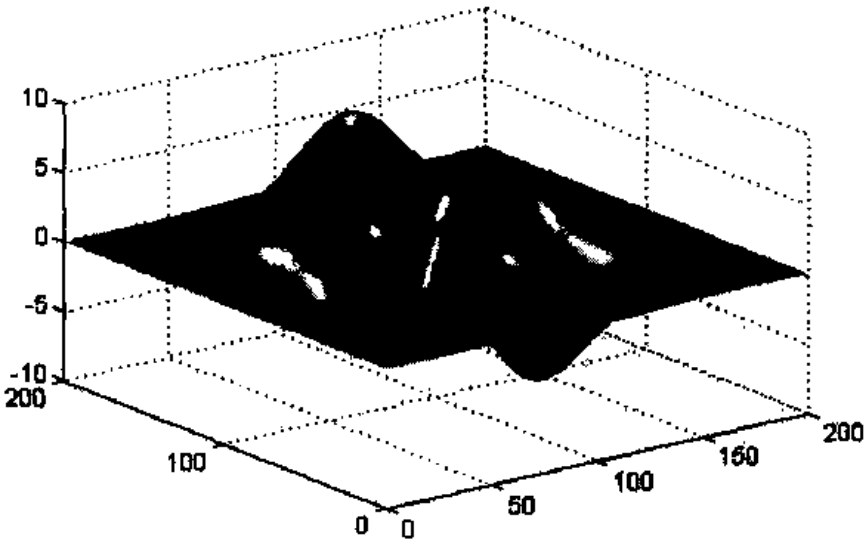


图 6.15 光照效应

还有两个与 meshc 和 meshz 对应的函数 surfc 和 surfz，这里不再详述。

6.2.3　等高线图形

MATLAB 支持二维和三维的等高线图形。函数 contour 和 contour3 被用来实现这种功能。它们将输入的矩阵变量 M 看作是定义在[1,m]×[1,n]上的函数，生成若干条常数值的等值线段。用户也可以指定等高线的条数、坐标系的比例以及某值上的等高线。

等高线的线型、顶点标记、颜色类型类似于 plot 函数定义。contour3 适用于三维等高线的生成，调用方式与 contour 相同。

【例 15】二维和三维等高线的生成见图 6.16 和图 6.17

```
contour(peaks,30)                    % 生成二维等高线
contour3(peaks,20)                   % 生成三维等高线
```

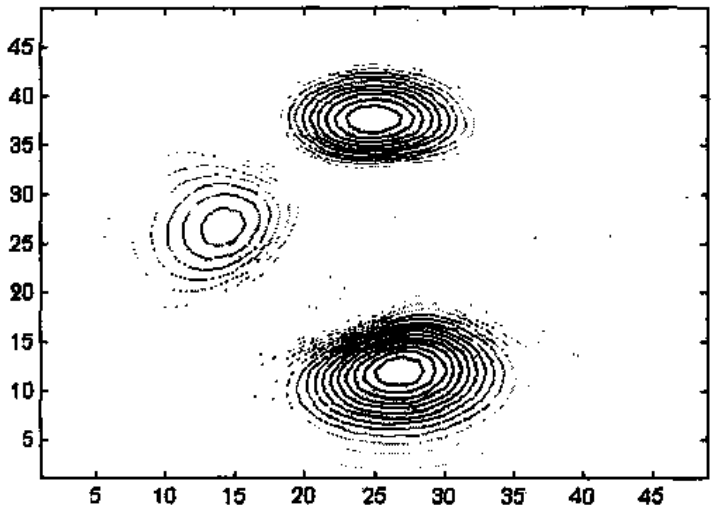


图 6.16　二维等高线

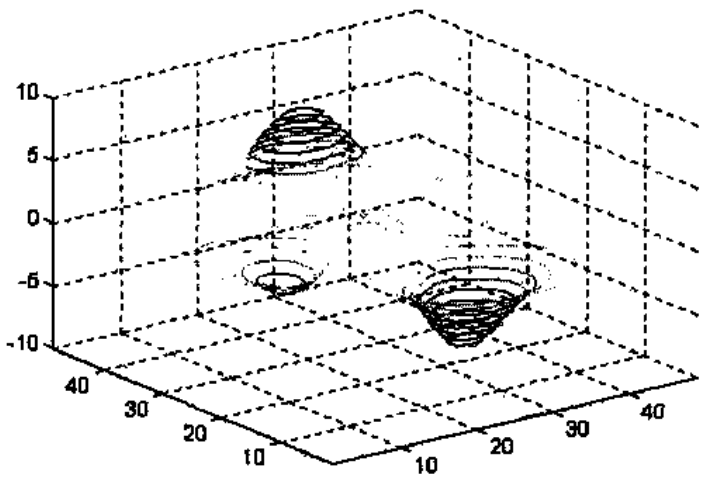


图 6.17　三维等高线

6.2.4 改变视角

注意两个图形，一个是以 30 度视角向下看 z=0 平面,一个是以 37.5 度视角向上看 x=0 平面。这是对所有三维图形的默认视角。与 z=0 平面所成的方向角叫仰角，与 x=0 平面的夹角叫做方位角。这样，默认的三维视角方向仰角为 30 度，方位角为-37.5 度。而默认的二维视角仰角为 90 度，方位角为 0 度。

在 MATLAB 中，函数 view 改变所有类型的二维和三维图形的图形视角。view(az,el)和 view([az,el])将视角改变到所指定的方位角 az 和仰角 el。

【例 16】view 的使用

```
x=linspace(0,3*pi).';
Z=[sin(x)  sin(2*x)  sin(2*x)];
Y=[zeros(size(x))  ones(size(x))/2  ones(size(x))];
% 生成矩阵 Y、Z
subplot(2,2,1)
plot3(x,Y,Z)
grid,xlabel(" X-axis "),ylabel(" Y-axis "),zlabel(" Z-axis ")
title('默认视角：方位角= -37.5,仰角 = 30 ')
view(-37.5,30)          % 变换视角至-37.5 度,30 度
subplot(2,2,2)
plot3(x,Y,Z)
grid,xlabel( " X-axis " ),ylabel(" Y-axis " ),zlabel(" Z-axis " )
title( ' 方位角旋转至 52.5 度 ')
view(-37.5+90,30)       % 变换视角至 52.5 度,30 度
subplot(2,2,3)
plot3(x,Y,Z)
grid,xlabel(" X-axis " ),ylabel(" Y-axis " ),zlabel(" Z-axis " )
title( ' 仰角变为 60 度 ')
view(-37.5,60)          % 变换视角至-37.5 度,60 度
subplot(2,2,4)
plot3(x,Y,Z)
grid,xlabel(" X-axis " ),ylabel(" Y-axis " )
title( ' 方位角为 0 度，仰角为 90 度' )
view(0,90)              % 变换视角至 0 度,90 度
```

输出的结果如图 6.18 所示。

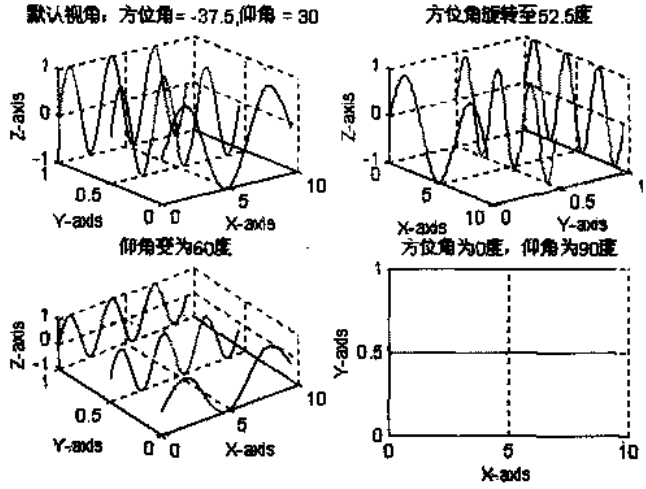


图 6.18　view函数的使用

函数 view 的其他用法见表 6.3。

表 6.3　函数view的使用

函数调用	功　能
view(az,el) view([az,el])	将视图设定为方位角 az 和仰角 el
view([x,y,z])	在笛卡儿坐标系中将视图设为沿向量[x,y,z]指向原点，例如 view([0 0 1])=view(0,90)
view(2)	设置默认的二维视角，az=0，el=90
view(3)	设置默认的三维视角，az=-37.5，el=30
[az,el]=view	返回当前的方位角 az 和仰角 el
view(T)	用一个 4×4 的转矩阵 T 来设置视图角
T=view	返回当前的 4×4 转矩阵

6.2.5　透视效应

MATLAB 在绘制图形时，在默认方式下，前面的图形会挡住后面的图形，即消去后面的隐藏线，但是可以用命令

```
hidden off
```

改变这种模式。相反，也可以用

```
hidden on
```

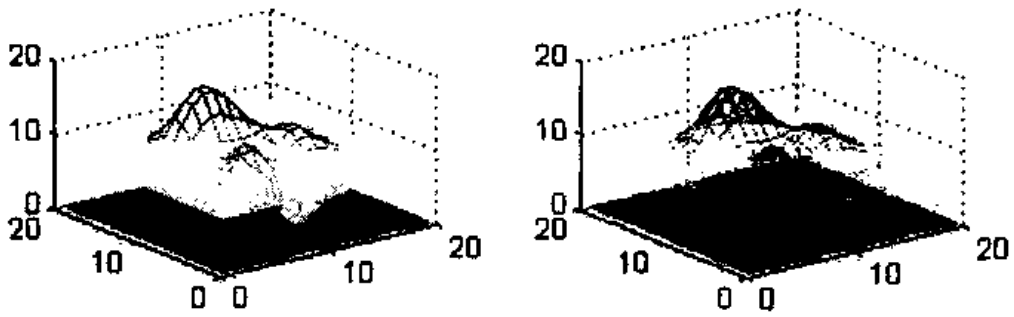
重新返回到默认的模式。

【例 17】hidden on 和 hidden off 模式的图形

下面的语句首先生成 peaks 的网格曲面，然后在 x-y 平面上生成 peaks 的伪图。显然，网格曲面挡住了部分伪图，如图 6.19a 所示。使用 hidden ff 之后，被挡住部分成为可见的，

如图 6.19b 所示。

```
mesh(peaks(20)+7);
hold on
pause
pcolor(peaks(20));
hidden off
```



a 部分伪图被挡　　b 伪图可见

图 6.19 hidden on与hidden off模式的图形

6.2.6 曲面的裁剪方法

因为曲面图不能作成透明，但在一些情况下可以很方便地移走一部分表面以便看到表面以下的部分。这在 MATLAB 中，通过在所期望的洞孔的所在位置，将数据置为特定的 NaN 来实现。由于 NaN 没有任何值，所有的 MATLAB 作图函数都忽略 NaN 的数据点，在该点出现的地方留下一个洞孔。

【例 18】利用 NaN 进行曲面裁剪

```
[X,Y,Z]=peaks(30);
x=X(1,:);  % vector of x axis
y=Y(:,1);  % vector of y axis
i=find(y>.8 & y<1.2);  % find x-axis indices of hole
j=find(x>-.6 & x<.5);  % find x-axis indices of hole
Z(i,j)=nan*Z(i,j);  % set values at hole indices to NaNs
surf(X,Y,Z)
grid,xlabel(" x-axis "),ylabel(" y-axis "),zlabel(" z-axis ")
title( " SURF of PEAKS with a Hole " )
```

输出结果见图 6.20。

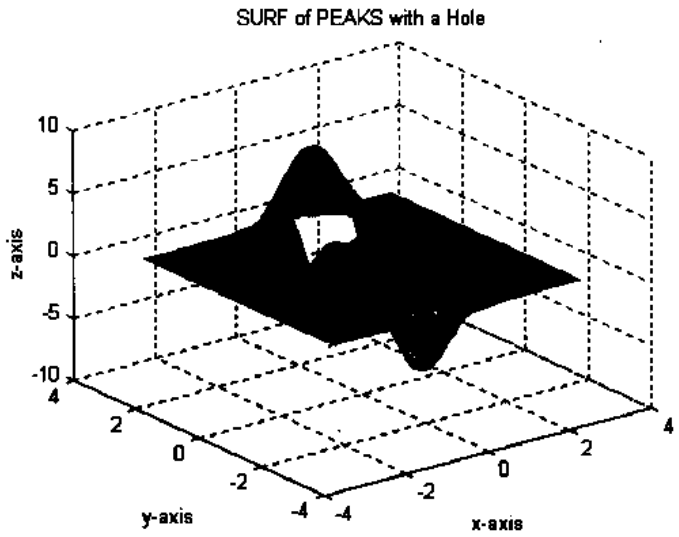


图 6.20　peaks曲面的裁剪图形

6.3　四维表现和切片图

对于一般的定义在 x-y-z 坐标系上的四维可视化，可以用指令 slice 来实现。

为了实现三元函数 R=f(x,y,z)的可视化表现，MATLAB 提供了一个绘制三维物体切片图指令及与之配合的三维网格坐标生成指令。

```
[X,Y,Z]=meshgrid(x,y,z)          三维网格坐标的生成
slice(X,Y,Z,V,xi,yi,zi,n)        绘制三维物体切片图
```

其中：

x，y，z　决定“网线”的位置，分别是(1×n)、(1×m)和(1×p)向量

X，Y，Z　三维网格坐标，它们都是(n×m)×p 维数组

V　在网线节点上的三元函数值数组，维数也为(n×m)×p

xi，yi，zi　分别决定垂直于 x、y、z 轴切面的位置向量。它们的维数可以不同。当取 0 维空阵时，表示没有切面存在。

切片上的函数值的大小可以用色轴上对应的颜色表示。V 数组中的最大有限值和最小有限值定义了色轴的范围。又由于切片的位置可以任意设置，因此 slice 通过三维坐标点上的色彩变化把图形的表现能力扩展到四维。

【例 19】函数$R=xe^{-(x^2+y^2+z^2)}$的四维表现图 6.21 所示。

```
x=-2:0.1:2; y=-2:0.25:2; z=-2:0.25:2; n=length(x);
[X,Y,Z]=meshgrid(x,y,z);
V=X.*exp(-X.^2-Y.^2-Z.^2);
xi=[-0.7,0.7]; yi=0.5;zi=-0.5;
slice(X,Y,Z,V,xi,yi,zi);
xlabel('x'); ylabel('y'); zlabel('z');
```

```
hold on
colorbar('horiz')
view([-30 45])
```

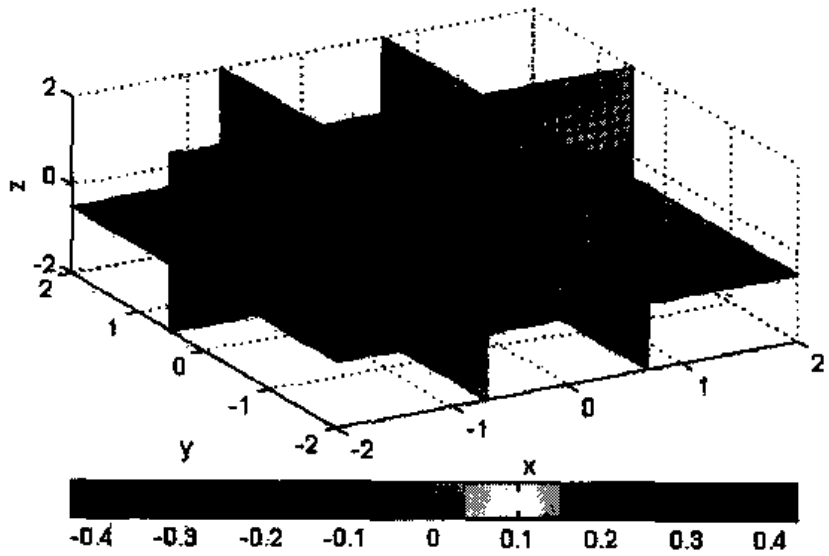


图 6.21 四维表现

说明：

- 解析计算表明：

在 x=0.7071，y=0，z=0 时，max(R)=0.4289

在 x= −0.7071，y=0，z=0 时，max(R)= −0.4289

- 色轴在图的最下方，它表明了颜色和数组的对应关系。

6.4 图形的标注

在 MATLAB 5.3 中，提供了一个图形用户界面(GUI 界面)的图形标注工具。但是，传统的基于命令行的标注方式依然可用。本节将分别介绍这两种标注方法。

6.4.1 使用命令行进行标注

MATLAB 可以给画出的图形加上各种标注及文字说明，以丰富图形的表现力。在中文 Windows 环境下，可以用汉字进行标注和文字说明。但是需要注意，用于标记文字的单引号应该是中文的而不是英文的单引号。

1. 图名和坐标名的标注

MATLAB 为定义图形名称和坐标轴名称提供了专门的标注指令。在新版本中，标注的内容不仅可以是外文字符，还可以是中文字符，具体如表 6.4 所示。

表 6.4　图形和坐标轴的标注

函数调用方式	功　能
title('String')	在图形顶端加注文字作为图形名
title('Sting','Property'Propertyvalue,…)	定义图名所用字体、大小、标注角度
xlabel('Sting')	在当前图形的 x 轴旁加入文字内容
xlabel('Sting','Property'Propertyvalue,…)	定义轴名所用字体、大小、标注角度
Legend('Sting1','Sting2'…)	对当前图进行图例标注

说明：

- 给 y、z 轴加标注的指令是 ylabel、zlabel，它们的调用格式和 xlabel 完全一样。
- 标注的内容 String 可以是中文。
- 第二种调用涉及到图形句柄的操作，这里不再介绍。另外，可以在 GUI 界面中直接修改属性。
- 如果用 legend 进行了标注，又需要改变图例说明框的位置，可以用鼠标将它拖动到合适的地方。

2. 所绘图形的标注

MATLAB 还提供对于所绘图形的文字标注功能——指令 text 和在图形中指定的点上加注文字的指令 gtext。先用鼠标定位，再在此位置加注文字，但不支持三维图形。

- text 指令

```
text(x,y,z,'Sting')                在点(x,y,z)上标注文字 String
text('PropertyName',PropertyValue,…)    对标注文字的属性加以定义
```

说明：

- 如果 x、y、z 是向量，则在三个向量定义的每一点上标注文字 String。如果 String 也是由字符串组成的向量，则在每一点上标注对应的字符。
- 在第一种调用格式中，如果没有 z，那么就是在二维平面图形上加注。
- gtext 指令

```
gtext('String')                    在鼠标指定位置上加注
```

说明：

使用这个指令后，会在当前图形上出现一个十字叉，等待用户选定位置进行标注。移动鼠标到所需的位置按下鼠标左键，MATLAB 就在选定位置标上文字。

3. 分格线

分格线的控制指令是一个对于二维和三维图形都适用，且使用频率较高、使用方法简便的指令。介绍如下：

```
grid on     打开分格线控制开关，以后绘制的图形都带有分格线
grid off    关闭分格线控制开关，以后绘制的图形都不带分格线
grid        实现分格线绘制与否的切换
```

6.4.2 GUI 界面下的图形标注

在 MATLAB 5.3 中，Figure 窗口增加了一个工具条，可以在图形环境下对图形进行标注。表 6.5 列出了它的部分功能。

表 6.5 标注工具条的使用

图 标	功 能
	图形编辑模式的开关
	为所绘图形加注文字
	为所绘图形加上箭头
	为所绘图形加上线段
	放大所绘的图形
	缩小所绘的图形
	旋转视角

在 Figure 窗口菜单中，增加了 Tools 选项(5.3 版)，其中有 3 个命令比较有用，下面分别介绍。

1. Axes Porperies命令

单击此命令后，出现一个对话框，如图 6.22 所示。对话框的各个选项功能如表 6.6 所示。

表 6.6 坐标轴属性编辑对话框功能

选 项	功 能
Title	输入的文字成为图形的标题
Label	各个坐标轴的标注
Limits	各个坐标轴的刻度范围，通常由系统指定，但是也可以选中 Manual 复选框，手工指定
Tick Step	分度值，可以自动给出，也可手工指定，与 Limits 类似
Scale	选择分度方式
Grid	分格线开关

说明：

在 Scale 选项组中，可以选择：

Linear 线性分度

Log 对数分度

Normal 正常标注方式(刻度值从小到大)

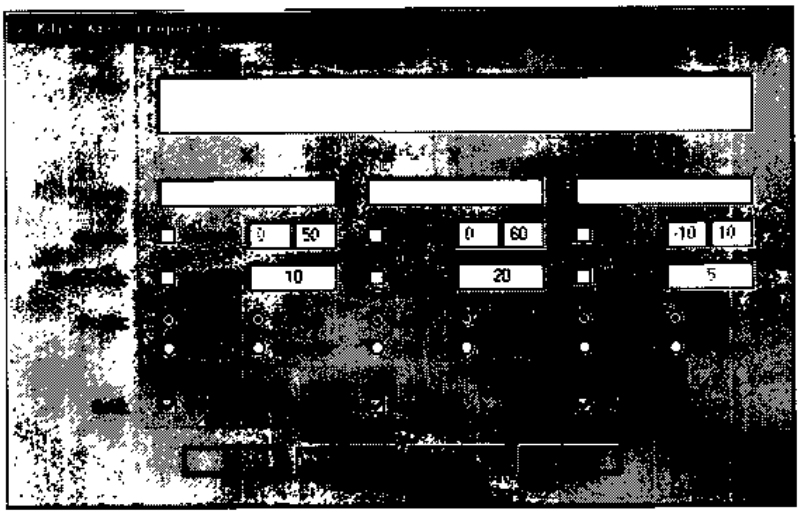


图 6.22　坐标轴属性的编辑对话框

2. Line Porperies命令

单击此命令后，出现一个对话框，如图 6.23 所示。

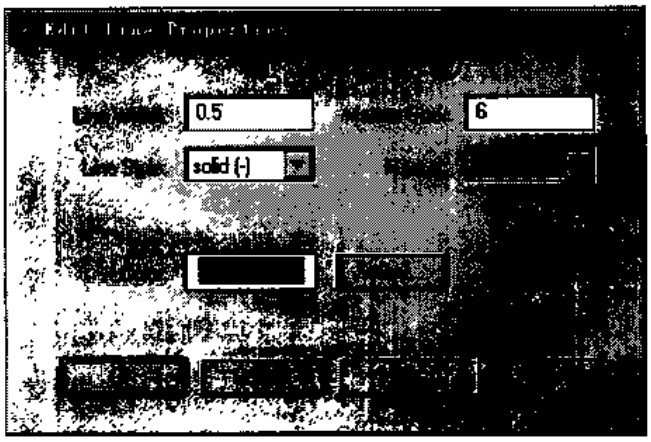


图 6.23　线段属性的编辑对话框

对话框的各个选项功能如下所示：

Line Width　　选择线宽

Marker Size　　数据标记的大小

Line Style　　选择线型

Marker　　选择数据标记的形状

Color　　选择线段的颜色

Select...　　GUI 界面的颜色选择

3. Text Porperies命令

单击此命令后，出现一个对话框，如图 6.24 所示。

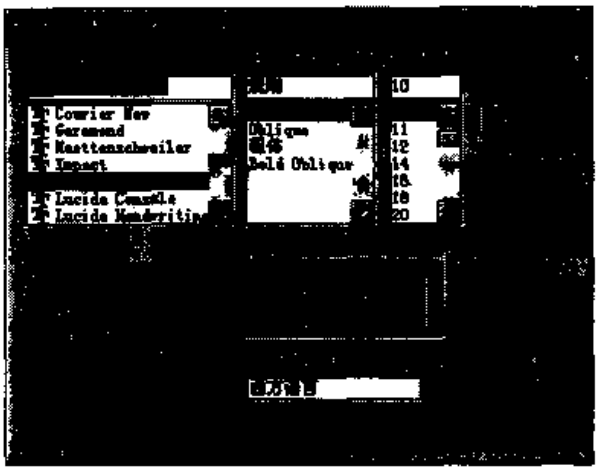


图 6.24 文字属性的编辑对话框

这个对话框是 Windows 标准对话框，与其他应用软件的文字属性对话框相同，用法也相似。这里就不再详述了。

第7章
高级图形处理

本章要点：

本章将介绍一些比较高级的图形处理函数，可以利用MATLAB 强大的图形处理功能，获得更好的图形效果和色彩表现。

本章具体包括以下内容：

- 如何在 MATLAB 中控制和使用色彩
- 如何在 MATLAB 中利用句柄图形
- 如何在 MATLAB 中使用动画

MATLAB 提供了许多在二维和三维空间内显示可视信息的工具。MATLAB 不仅是一个强大的计算工具，并且在以引人入胜和直观方式可视地表示数据方面也很有特色。很多时候，一个简单的二维或三维图形不能一次显示出想要提供的全部信息。这时，颜色可以对图形提供一个附加的维数。前面章节讨论的许多绘图函数都可以接受一个可用的颜色参量，来增加这附加的维数。

而通过句柄图形，可以更自如地控制和表现 MATLAB 的图形。本章将揭开句柄图形的神秘面纱，让用户进入更广阔的图形想象空间。

动画作为活动的图形，一直是个让人感兴趣的领域，MATLAB 同样可以高效地制作出生动、活泼的动画。

7.1 色彩的控制和表现

本节的讨论以研究颜色映像开始，如何创建、使用、显示和修改用户自己的颜色映像。然后分别介绍色彩的渲染和图像的表现。

7.1.1 颜色映像原理

MATLAB 利用颜色映像的数据结构来代表颜色值。颜色映像定义为一个有 3 列和若干行的矩阵。利用 0 到 1 之间的数，矩阵的每一行都代表了一种色彩。任何一行的数字都指定了一个 RGB 值，即红、黄、蓝三种颜色的强度，形成一种特定的颜色。表 7.1 给出一些有代表性的 RGB 值。

表 7.1 简单颜色

Red(红)	Green(绿)	Blue(蓝)	颜 色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
.5	0	0	深红
.5	.5	.5	灰色

MATLAB 系统有 10 个函数可产生预定的颜色映像，参见表 7.2。

表 7.2　标准颜色映像

标准颜色映像	说　明
hsv	色彩饱和值(以红色开始和结束)
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	hsv 的一种变形(以蓝色开始和结束)
copper	线性铜色度
prim	三棱镜，交替为红色、橘黄色、黄色、绿色和天蓝色
flag	交替为红色、白色、蓝色和黑色

说明：

默认情况下，表 7.2 中所列的各个颜色映像产生一个 64×3 的矩阵，指定了 64 种颜色 RGB 的描述。这些函数都接受一个参量来指定所产生矩阵的行数。比如 hot(m)产生一个 m×3 的矩阵，它包含的 RGB 颜色值的范围从黑经过红、橘红和黄到白。其他的函数调用请参看 MATLAB 的联机帮助文件。

大多数计算机在一个 8 位的硬件查色表中一次可以显示 256 种颜色，当然有些计算机的显示卡可以同时显示更多的颜色。这就意味着在不同的图中，一般一次可以用 3 或 4 个 64×3 的颜色映像。如果使用了更多的颜色映像输入项，计算机必须经常在它的硬件查色表中调出输入项。比如，当在画 MATLAB 图形时背景图案发生了变化，就是发生了这种情况。所以，除非计算机有一次显示更多种颜色的显示卡，最好任何一次所用的颜色映像输入项数都小于 256。

7.1.2　颜色映像函数

MATLAB 系统提供了一系列函数用来处理和使用颜色映像函数，这些函数列于表 7.3 中。

表 7.3　颜色映像函数

函 数 名 称	功 能 简 介
colormap(map)	在当前的图形窗口中安装一个颜色映像
colorbar	在当前的图形上显示一个水平的或垂直的颜色标尺
rgbplot(map)	颜色映像中红、绿、蓝分量的直线图
brighten(a)	0<a<1，当前颜色映像加亮；−1<a<0，当前颜色映像加暗

续表7.3

函 数 名 称	功 能 简 介
m=brighten(map,a)	返回加亮的颜色映像 m
[cmin,cmax]=caxis	返回颜色轴的界限
caxis([cmin,cmax])	设置颜色轴的界限

下面结合各种应用具体介绍各函数的功能。

1. 颜色映像的使用

函数 colormap(M)将矩阵 M 作为当前图形窗口所用的颜色映像。例如，colormap(cool)装入了一个有 64 个输入项的 cool 颜色映像。colormap default 装入了默认的颜色映像(hsv)。

接受颜色参量的绘图函数中的颜色参量通常采用以下三种形式之一：字符串，代表 plot 颜色或线型表中的一种颜色，比如 r 代表红色；三维行向量，它代表一个单独的 RGB 值，比如[.25.50.75]；矩阵，如果颜色参量是一个矩阵，其元素作了调整，并把它们用作当前颜色映像的下标。最后一种形式以后将做更多讨论。

2. 显示一个颜色映像

可以用多种途径来显示一个颜色映像，其中一个方法是观察颜色映像矩阵的元素。

【例 1】显示颜色映像矩阵。

```
hot(8)                  %产生从黑到红到黄到白的颜色映像
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

上面的数据显示出第一行是 1/3 红色，而最后一行是白色。

另外，函数 pcolor 可以用来显示一个颜色映像。具体调用格式为：

```
pcolor(C)               %产生矩阵 C 的伪色图
```

该函数的作用是将矩阵 C 作为颜色矩阵，利用着色原理在平面网格点(i,j)的右上角小区域内用 C(i,j)对应的色谱矩阵的颜色着色。

【例 2】函数 pcolor 显示一个颜色映像(见图 7.1)。

```
n=16;
colormap(pink(n))
```

```
pcolor([1:n+1;1:n+1])
title('用 pcolor 函数显示色图')
```

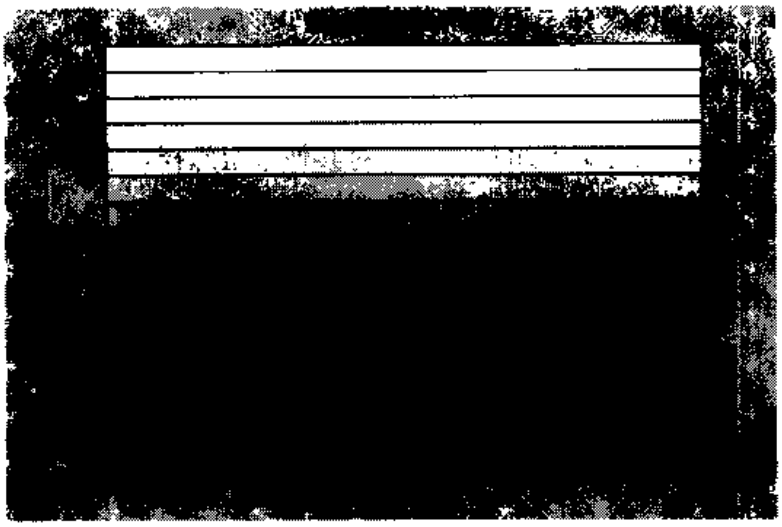


图 7.1　用伪彩色来显示颜色映像

MATLAB 的函数 rgbplot 可以把颜色映像的各列分别画成红、绿和蓝色。函数 colorbar 在当前的图形窗口中增加水平或垂直的颜色标尺以显示当前坐标轴的颜色映像。

说明：

- colorbar('horiz') 在当前的图形下面放一个水平的颜色条。
- colorbar('vert') 在当前的图形右边放一个垂直的颜色条。
- 对无参量的 colorbar，如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条。

【例 3】用 rgbplot 和 colorbar 函数显示 hsv 颜色映像(见图 7.2)。

```
rgbplot(hsv)          %用红蓝绿三色分别画 hsv 矩阵的三列向量。
colormap(hsv)         %把 hsv 指定为当前图形窗口的颜色对照表。
axis([1,64,0,1])      %hsv 行维为 64 和元素值在[0,1]间。
colorbar('horiz')     %画水平色轴。
```

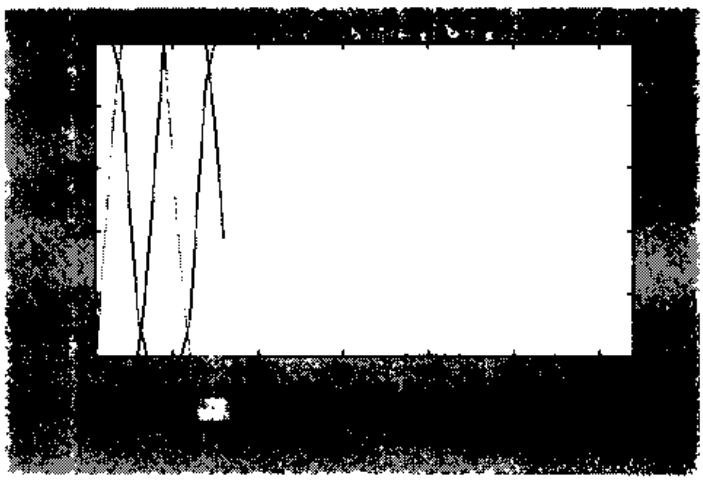


图 7.2　用红、绿和蓝色画颜色映像

说明：图中横坐标是 hsv 色图矩阵的“行下标”。纵坐标表示基色的亮度，0 最暗，1 最亮。图中的红线、绿线和蓝线分别表示在不同行下标上红色、绿色和蓝色的亮度值。色轴上的颜色正是由相应的横坐标上的 3 个基色调和而形成的。

3. 颜色映像的建立和修改

颜色映像就是矩阵，这意味着用户可以像对其他数组那样对它们进行操作。MATLAB 系统提供如下的函数命令，建立和修改颜色映像矩阵。

- 函数 brighten

调整一个给定的颜色映像来增加或减少暗色的强度。

brighten(n)(0<n<=1)　　使当前颜色映像变亮。

brighten(n)(-1<=n<0)　　使它变暗。

说明：brighten(n)后加一个 brighten(-n)使颜色映像恢复原来状态。newmap=brighten(n)命令创建一个比当前颜色映像更暗或者更亮的新的颜色映像，而并不改变当前的颜色映像。命令 ewmap=brighten(cmap,n)对指定的颜色映像创建一个已调整过的式样，而不影响当前的颜色映像或指定的颜色映像 cmap。

- 用户自定义颜色映像

可以通过生成 m×3 的矩阵 mymap 来建立用户自己的颜色映像，并用 colormap(mymap)来安装它。颜色映像矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小或比 1 大的任意值，函数 colormap 会提示一个错误信息然后退出。

也可以在算术上来组合颜色映像，其结果有时是不可预料的。比如，一个叫 pink 的颜色映像是：

```
pinkmap=squr(2/3*gray+1/3*hot);
```

只有当所有元素都在 0 与 1 之间时，才能保证结果是一个有效的颜色映像。

- 色轴范围设置函数 caxis

一个颜色映像定义了用于绘制图形的调色板。一个默认的颜色映像允许对数据使用 64 种不同的 RGB 值。MATLAB 使用函数 caxis 来决定哪一个数据值映射到颜色映像中输入项。

通常，颜色映像进行过调节，把数据从最小扩展到最大，也就是说，整个颜色映像都用于绘图。有时也许想改变颜色使用的方法。函数 caxis 代表颜色轴，因为颜色轴增加了另一个维数，它允许对数据范围的一个子集使用整个颜色映像或者对数据的整个集合只使用当前颜色映像的一部分。

[cmin,cmax]=caxis 返回映射到颜色映像中第一和最后输入项的最小和最大的数据。它们通常被设成数据的最小值和最大值。比如，函数 mesh(peaks) 会画出函数 peaks 的网格图，并把颜色轴 caxis 设为[-6.5466，8.0752]，即 Z 的最小值和最大值。这些值之间的数据点，使用从颜色映像中经插值得到的颜色。

caxis([cmin，cmax])对 cmin 和 cmax 范围区内的数据使用整个颜色映像。比 cmax 大的数据点用与 cmax 值相关的颜色绘图；比 cmin 小的数据点的颜色用与 cmin 值相关的颜色绘图。如果 cmin 小于 min(data)和/或 cmax 大于 max(data)，那么与 cmin 和/或 cmax 点相关的颜色将永远用不到。也就是说，只用到和数据相关的那一部分颜色映像。caxis('auto')

设置 cmin 和 cmax 的默认值。

【例 4】默认的颜色范围(见图 7.3)

```
pcolor([1:17;1:17] ),colormap(jet(8))
title( '默认的颜色范围' )
caxis( 'auto' )
colorbar
caxis
ans =
        1    17
```

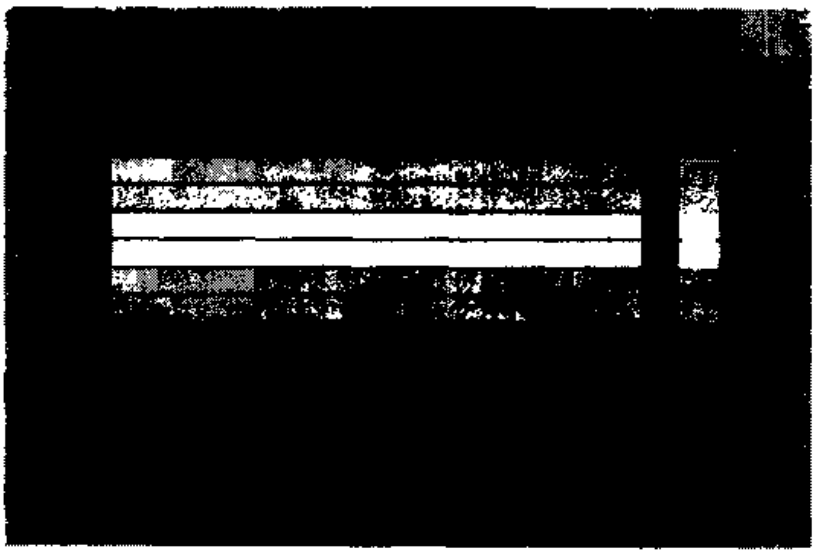


图 7.3　默认的颜色范围

由图 7.3 可见，对整个数据集合，当前颜色映像使用了所有 8 种颜色。每种颜色有两条。如果颜色被映射到从−3 到 23 的数据，那么图中只用到 5 种颜色。这可以通过下面的命令实现。

【例 5】扩展的颜色范围(见图 7.4)。

```
title( '扩展的颜色范围' )
caxis([-3,23])  %扩展颜色范围
colorbar        %重绘颜色条
```

如果颜色映射到从 5 到 12 的值，会用到所有的颜色。但是，比 5 小的数据和比 12 大的数据分别映射到与数据值 5 和 12 相关的颜色。这可以通过下面的命令产生。

【例 6】受限的颜色范围(见图 7.5)。

```
title( '受限的颜色范围' )
caxis([5,12])    % 限制颜色范围
colorbar  %      %重绘颜色条
```

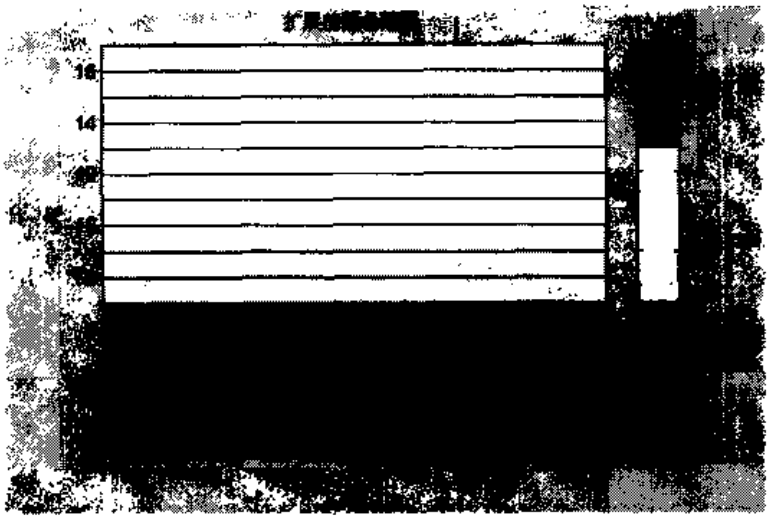


图 7.4 扩展的颜色范围

图 7.5 受限的颜色范围

7.1.3 色彩的渲染

函数 shading 设定由 mesh、surf、pcolor、fill 和 fill3 所创建的图形渲染方式。该函数有如下 3 种调用方式：

- shading flat

将图形色彩渲染为平坦方式，即根据线段两端(或三维图形的表面小方块四角)的值决定网格线段(或小方块面)取一对应的值。

- shading interp

将图形色彩渲染为插补方式，即网格线段(或小方块面)上每一个点的颜色是根据线段两端(或三维图形的表面小方块四角)的值通过双线性插补所算得的值决定的。因此线段 (或三维图形的表面小方块)的颜色是渐变的。

- shading faceted

默认的渲染方式，它是在 shading flat 方式的基础上再加上黑色网格线。这是最有效

的渲染方式。

【例 7】两种渲染方式的比较(见图 7.6 和图 7.7)。

```
clf;x=-6:0.4:6;y=x;[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps;        %加 eps 防止在零点产生出错信息
z=sin(R)./R;
surf(z);colormap(hsv);
shading flat                  %使用 flat 渲染方式画图
shading interp                %使用 interp 渲染方式画图
```

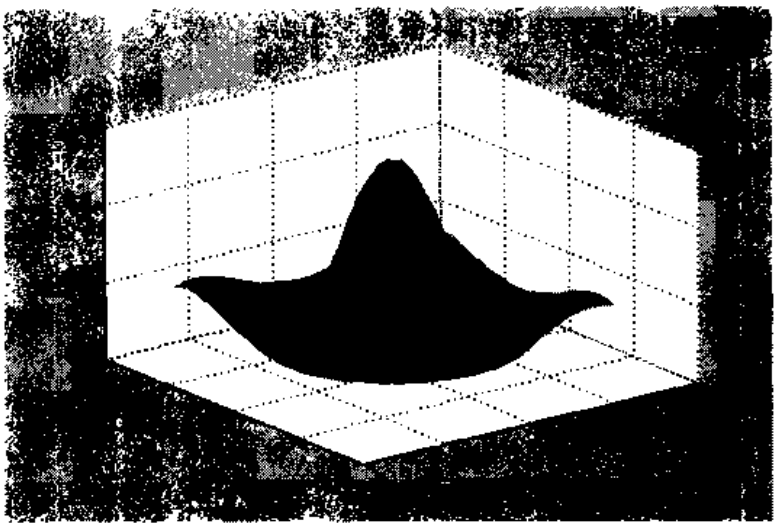


图 7.6　使用flat渲染方式画图

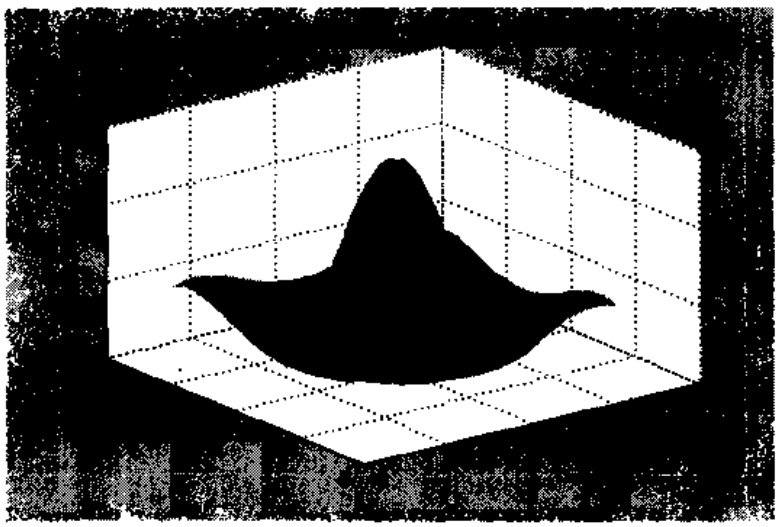


图 7.7　使用interp渲染方式画图

说明：

由图 7.6 和 7.7 可以看到，采用 interp 渲染方式画图，颜色的过渡平和多了。

7.1.4　图像显示技术

1.　image与imagesc函数

数字图像是指将一幅二维的图像表示成一个数值矩阵，矩阵的元素被解释为像素的颜色值(或灰度值)，或被解释为调色板颜色的索引号。为了显示由矩阵表示的数字图像，MATLAB 最一般的作法是将矩阵的每个元素对应到当前色谱的某个行标号，并取该行的颜色值作为图像相应点的颜色。一般说来，每幅图的色调不同，因此作为图像必须有自己特殊的色图，这样才能真实地显示图像。

MATLAB 用函数 image 显示图像，其命令形式为：

```
image(X)          %表现图形的矩阵
colormap(map)     %为表现该图所给定的特定色图
```

【例 8】图像的表现(图 7.8)。

在 MATLAB 中，有一幅图像的数值 MAT 文件是 gatiin.mat，它包含图像矩阵 X 和调色板矩阵 map。于是，下列语句就可以真实地再现这幅图像。

```
load  clown          %提取在 MATLAB\DEMOS 目录下的一组图形数据
image(X);
colormap(map);
axis  equal          %保持图形比例
axis('off')              %隐藏坐标轴
```

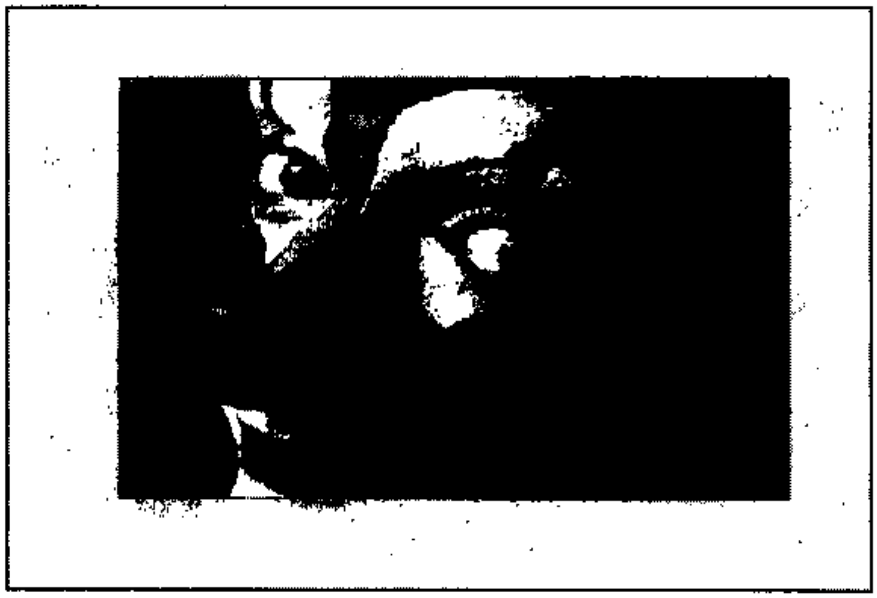


图 7.8　图像的表现

说明：

一般情况下，要保证图像正确的纵、横方向比例。上例中的语句 axis equal 就是为了达到这样的目的。

提示： 在没有装入图像的调色板之前，image(x)用的是当前的色谱。它的效果与 pcolor 的效果相类似，但它们之间有明显的区别。首先，image 是用来显示真正的图像的，包含装入调色板；而 pcolor 则是用图形的方式表达抽象的数学对象的数量关系。此外，它们还存在以下几个方面的差别(矩阵 A 是 m × n 的数值矩阵):

- image(A)对 m × n 片小矩形进行着色，横轴和纵轴网格的分点在 k+1/2 点处，而 pcolor(A)在(m − 1) × (n − 1)片小矩形上着色，且对网格线着黑色(默认色)。
- image(A)对每小片矩形区域按 flat 模式(分片常值)着色，而 pcolor(X)可以有多种着色模式，用 shading 函数来设置。
- image(A)采用 i − j 矩阵坐标系，而 pcolor(A)的默认方式是 x − y 笛卡尔坐标系。
- image(A)总是生成等分的网格，而 pcolor(X,Y,A)可以生成参数型的网格。
- image(A)生成的二维图像，只能从标准的二维视点(0，90)观看，即不能改变视点，而 pcolor(A)生成的图形可以从任何视点观看。
- image(A)直接用 A 值作为索引在色谱中对应颜色值，即 A 的元素被截成整数，其范围是 0 到 length(colormap)；而 pcolor 是根据坐标系的颜色区间(可以由 caxis 设置)按线性变换方式计算索引值来对应色谱矩阵的颜色值。image(A)的图像不受 caxis 函数的影响。
- image(A)能固定与图像同样大小的坐标系，使图像充满坐标系，而 pcolor 不具备这种功能。

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

2. 图像类型

从前面的分析已经看到，MATLAB 的图像由两部分构成，一个是数据矩阵，另一个是图像的色谱，即调色板。为了有效地使用全真颜色的图像，MATLAB 5.x 扩展了它的图像处理功能。它将图像分成以下 3 种类型：

- 索引式图像

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

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

- 强度式图像

图像数据矩阵 X 的元素值落在某个强度范围内，一般情况下，强度范围是[0，1]或[0，

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

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

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

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

- 全真颜色图像

与通常的全真颜色图像文件一样，全真颜色图像没有调色板，也就没有索引方式。只有一个 m×n×3 的三维数组，定义图像像素颜色的 RGB 值。记这个数组为 RGB，那么，RGB(：，：，1)、RGB(：，：，2)、RGB(：，：，3)分别记录 R(红色)值、G(绿色)值和 B(蓝色)值。例如，像素点(10，5)的 R、G、B 值分别为 RGB(10，5，1)、RGB(10，5，2)和 RGB(10，5，3)。在 MATLAB 5.x 的系统中，用 image 函数显示全真颜色图像，例如：

```
image(RGB)
```

3. 图像数据的格式

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

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

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

imagesc(X，[0 255])；colormap(gray)

对于全真颜色图像，如果三维数组 RGB 的类型是 unit8，那么任何一种 R、G、B 单色值的范围是[0，255]。在这种情形下，如果一个像素的 RGB 值是(255，255，255)，那么该像素就被着白色。对于全真颜色图像来说，无论 RGB 的类型是 unit8，还是 64 位的浮点型数，都用下面的命令显示该图像：

```
image(RGB)
```

可以用函数 unit8 和 double 进行两种类型之间的转换。例如：

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

4. 读写图像文件

MATLAB 5.x 改进了原来的两个函数，即 imread 和 imwrite。它们分别用于将图像文件读入 MATLAB 工作区，以及将图像数据和调色板数据一起写成一定格式的外部图像文件。在 MATLAB 4.2 版本中，imread 和 imwrite 只支持 RAW 图像格式。新版本扩展了它

的功能，能支持下列几类图像格式：

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

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

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

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

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

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

当图像文件是全真颜色的图像时，imread 将每个像素的 RGB 值直接读入到一个三维矩阵 X 中。例如，image.jpg 是全真颜色的 JPEG 格式的图像文件，那么，下列语句就把全部像素的 R、G、B 值装入到 3 维矩阵 X 中：

```
x=imread('image.jpg')
```

反过来，imwrite 则将 MATLAB 工作区中的图像数据及其调色板数据按指定的图像格式写入到一个外部图像文件中：

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

这里是将 MATLAB 的图像 clown 按 BMP 格式重新存储起来。

7.2　句 柄 图 形

句柄图形是对底层图形例程集合的总称，它实际上进行生成图形的工作。这些细节通常隐藏在图形 M 文件的内部，但如果想使用它们也是可以得到的。句柄图形可以被用来改变 MATLAB 生成图形的方式，不论是只想在一幅图里做一点小变动，还是想做影响所有图形输出的全局变动。

句柄图形允许定制图形的许多特性，而这用高级命令和前几章里描述的函数是无法实现的。例如，如果想用橘黄色来画一条线，而不是 plot 命令中可用的任何一种颜色，该怎么做呢？句柄图形就可以提供一种方法。

本节不对句柄图形做详细讨论，因为那样涉及问题太细。这里的目的只是对句柄图形概念做基本了解，并提供足够多的信息，使得即使是偶尔使用 MATLAB 的用户也可以利用句柄图形。

7.2.1 图形对象

句柄图形是基于这样的概念，即一幅图的每一组成部分都是一个对象，每一个对象有一系列句柄和它相关，每一个对象有按需要可以改变的属性。一个对象可以定义为由一组紧密相关、形成唯一整体的数据结构或函数集合。在 MATLAB 中，图形对象是一幅图中很独特的成分，它可以被单独地操作。

由图形命令产生的每一件东西都是图形对象。它们包括图形窗口或仅仅是图形，还有坐标系、线条、曲面、文本和其他。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其他对象的父亲。图形窗口是根对象的子对象；坐标系和用户界面对象是图形窗口的子对象；线条、文本、曲面、贴片和图像对象是坐标系对象的子对象。这种层次关系在图 7.9 中给出。

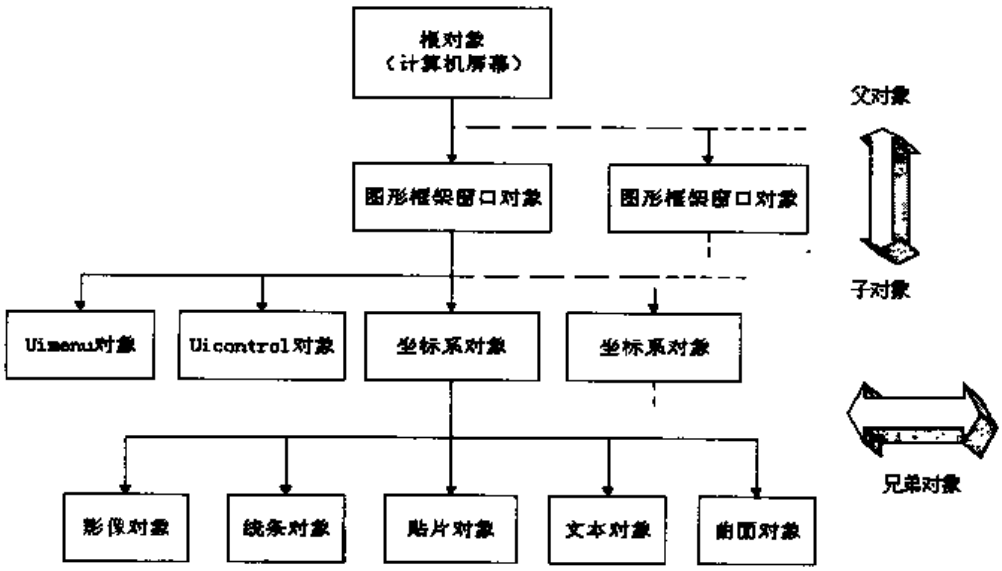


图 7.9 对象层次结构

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其他的对象(uicontrol 和 uimenu 外)都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，如果没有图形窗口，plot(rand(size([1：10])))函数会用默认属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

7.2.2 句柄对象

1. 句柄的概念

假设已打开了 3 个图形窗口，其中两个有两幅子图，并要改变其中一幅子图坐标轴内一条线的颜色，如何认定想要改变的那条线？在 MATLAB 中，每一个对象都有一个数字来标识，叫做句柄(handle)。

每次创建一个对象时，就为它建立一个唯一的句柄。计算机屏幕作为根对象，其句柄常常是零，而其他对象的句柄则是浮点数。

注意： 由于精度的要求，最好把浮点数句柄赋值给变量，以备后用。但千万不要试图从键盘输入屏幕所见句柄的浮点数。

所有产生对象的 MATLAB 函数都为所建立的每个对象返回一个句柄(或句柄的列向量)。这些函数包括 plot、mesh、surf 及其他。有些图形由一个以上的对象组成，比如，一个网格图由一个曲面组成，它只有一个句柄；而 waterfall 图形由许多线条对象组成，每个线条对象都有各自的句柄。

2. 对象创建函数

在MATLAB中除了根Root外，所有的对象都由与之对应的内置函数(Build-in Function)创建。表 7.4 给出了这些底层函数的功能和调用方式，每个函数都返回相应的图形句柄 h。

表 7.4　创建图形对象的底层函数

函数名称	函数功能	调用格式
axes	创建轴对象	Ha_axes=axes('position',[left,bottom,width,height])
figure	创建图形对象	Hf_fig=figure(n)
image	显示图形	Hi_imag=image(X)
line	创建直线对象	Hl_line=line(x,y,z)
patch	填充多边形	Hp_pat=patch(x,y,z,c)
surface	创建曲面对象	Hs_sur=surface(x,y,z,c)
text	创建文字对象	Ht_title=text(x,y,'string')
uicontrol	用户界面控制	Hu_uic=uicontrol('property',value)
uimenu	用户界面菜单	Hu_uim=uimenu('property',value)

提示： 为了提高可读性，建议用户为包含句柄对象的变量取名时以大写的 H 开头，后接一个辨识对象类型的字母，然后是一个下划线，最后是一个或几个描述符。因此，Hf_fig 是一个图形窗口的句柄，Ha_axis 是坐标轴对象的句柄，而 Ht_title 是一个文本对象的句柄。当不知道对象类型时，用字母 x，比如 Hx_obj。虽然句柄变量可以取任意名字，遵循这种规则在 M 文件中能很容易找到句柄变量。

每一个底层函数创建的图形对象被置于适当的父对象中。如果在函数运行之前已经有了相应的父对象，新建的图形对象就会在父对象之中，而且不影响父对象中已经存在的子对象。但是如果适当的父对象不存在，MATLAB 系统会自动地创建所有子对象必需的父对象。

3. 图形对象句柄的获得和删除

正如上面指出的，图形对象句柄可以通过创建过程中读取图形对象底层创建函数的返

回值来获得。MATLAB 还提供以下 3 个函数来获得对象句柄：

gcf 返回当前图形窗口的句柄。

gca 返回当前轴的句柄。

gco 获取当前对象的句柄。

获取图形句柄后，就可以对图形对象进行各种操作，其中包括删除图形对象，其函数为 delete。例如，delete(gca)将删除当前轴和它的所有子对象。

7.2.3 图形对象的属性

所有对象都由属性(property)来定义它们的特征，通过设定这些属性可以修正图形显示的方式。尽管许多属性是所有的对象都有的，但与每一种对象类型(比如坐标轴、线和曲面)相关的属性列表都是独一无二的。对象属性可包括诸如对象的位置、颜色、类型、父对象、子对象及其他内容。每一个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。

对象属性包括属性名和与它们相关联的值。属性名是字符串，它们通常按混合格式显示，每个词的开头字母大写，比如 LineStyle。但是，MATLAB 识别一个属性时是不分大小写的。另外，只要用足够多的字符来唯一地辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用 Position、position，甚至是 pos 来调用。

表 7.5~7.12 是 MATLAB 中常用的图形对象的属性列表。当然只能介绍图形对象属性中最重要的一小部分。具体的属性列表，用户可以参看 MATLAB 系统的帮助文件。

表 7.5 根对象属性

属性名称	含 义
BlackAndWhite	自动硬件检测标志。on：检测显示类型；{off}：认为显示是单色的，不检测
CurrentFigure	当前图形的句柄
Diary	会话记录。on：将所有的键盘输入和大部分输出拷贝到文件中；{off}：不将输入和输出存入文件
DiaryFile	一个包含 diary 文件名的字符串，默认的文件名为 diary
Echo	脚本响应模式。on：在文件执行时，显示脚本文件的每一行；{off}：除非指定 echo on，否则不响应
PointerLocation	相对于屏幕左下角指针位置的只读向量[left,bottom]或[X,Y]，单位由 Units 属性指定
PointerWindow	含有鼠标指针的图形句柄，如果不在图形窗口内，值为 0
ButtonDowFcn	MATLAB 回调字符串，当对象被选择时传给函数 eval，初始值是一空矩阵
Children	所有图形对象句柄的只读向量
Interruptible	ButtonDowFcn 回调字符串的可中断性。{no}：不能被其他回调中断；{yes}：可以被其他回调中断
Visible	对象可视性。{on}：对根对象有效果；{off}：对根对象无效果

表 7.6　图形对象属性

属性名称	含　义
Color	图形背景色，一个三元素的 RGB 向量或 MATLAB 预定的颜色名，默认的颜色是黑色
Colormap	m×3 的 RGB 向量矩阵，参阅函数 colormap
CurrentAxes	图形的当前坐标轴的句柄
CurrentCharacter	当鼠标指针在图形窗口中，键盘上最新按下的字符键
CurrentMenu	最近被选择的菜单项的句柄
CurrentObject	图形内最近被选择的对象的句柄，即由函数 gco 返回的句柄
CurrentPoint	一个位置向量[left,bottom]或图形窗口的点的[X,Y]，该处是鼠标指针最近一次按下或释放时所在的位置
KeyPressFcn	当鼠标指针处在图形内时，按下键，传递给函数 eval 的 MATLAB 回调字符串
MenuBar	将 MATLAB 菜单在图形窗口的顶部显示，或在某些系统中在屏幕的顶部显示。{figure}：显示默认的 MATLAB 菜单；{none}：不显示默认的 MATLAB 菜单
MinColormap	颜色表输入项使用的最小数目，它影响系统颜色表。如设置太低，会使未选中的图形以伪彩色显示
Name	图形框架窗口的标题(不是坐标轴的标题)。默认时是空串，如设为 string(字符串)，窗口标题变为“Figure　No.n：string”
NextPlot	决定新图作图行为。{new}：画前建立一个新的图形窗口；{add}：在当前的图形中加上新的对象；{replace}：在画图前，将除位置属性外的所有图形对象属性重新设置为默认值，并删除所有子对象
PaperPosition	代表打印页面上图形位置的向量[left,bottom,width,height]，[left,bottom]代表了相对于打印页面图形左下角的位置，[width,height]是打印图形的大小，单位由 PaperUnits 属性指定
PaperSize	向量[width,height]代表了用于打印的纸张大小，单位由 PaperUnits 属性指定，默认的纸张大小为[8.5　11]
PaperType	打印图形纸张的类型。当 PaperUnits 设定为归一化坐标时，MATLAB 使用 PaperType 来按比例调整图形的大小。{usletter}：标准的美国信纸；uslegal1：标准的美国法定纸张；a3：欧洲 A3 纸；a4letter：欧洲 A4 信纸；a5：欧洲 A5 纸；b4：欧洲 B4 纸；tabloid：标准的美国报纸
Pointer	鼠标指针形状。crosshair：十字形指针；{arrow}：箭头；watch：钟表指针；top1：指向左上方的箭头；topr：指向右上方的箭头；bot1：指向左下方的箭头；botr：指向右下方的箭头；circle：圆；cross：双线十字形；fleur：4 箭头形或指南针形
Position	位置向量[left,bottom,width,height]，[left,bottom]代表了相对于计算机屏幕的左下角窗口左下角的位置，[width,height]是屏幕大小，单位由 Units 属性指定

续表7.6

属性名称	含 义
Resize	允许不允许交互图形重新定大小。{on}：窗口可以用鼠标来重新定大小；{off}：窗口不能用鼠标来重新定大小
ResizeFcn	MATLAB 回调字符串，当窗口用鼠标重新定大小时传给函数 eval
WindowButtonDownFcn	当鼠标指针在图形内时，只要按一个鼠标按键，MATLAB 将回调字符串传递给函数 eval
WindowButtonMotionFcn	当鼠标指针在图形内时，只要移动鼠标指针，MATLAB 将回调字符串传递给函数 eval

表 7.7 坐标轴对象属性

属性名称	含 义
AspectRatio	纵横比向量[axis_ratio,data_ratio]，这里 axis_ratio 是坐标轴对象的纵横比(宽度/高度)，data_ratio 是沿着水平轴和垂直轴的数据单位的长度比。如设置，则 MATLAB 建立一个最大的坐标轴，保留这些比率，该最大轴将在 Position 定义的矩形内拟合。该属性的默认值为[NaN,NaN]
Box	坐标轴的边框。{on}：将坐标轴包括在一个框架或立方体内；{off}：不包括坐标轴
Clim	颜色界限向量[cmin cmax]，它确定将数据映射到颜色映像。cmin 是映射到颜色映像第一个入口项的数据，cmax 是映射到最后一项的数据。参阅函数 cmais
ClimMode	颜色限制模式。{auto}：颜色界限映成轴子对象的数据整个范围；{manual}：颜色界限并不自动改变。设置 Clim 就把 CLimMode 值设为 manual
DrawMode	对象生成次序。{normal}：将对象排序，然后按照当前视图从后向前绘制；fast：按已建立的次序绘制对象，不首先排序
LineStyleOrder	指定线形次序的字符串，用在坐标轴上画多条线。例如：' .- | : | -- | - ' 将通过点划线、点线、虚线和实线循环。LineStyleOrder 默认值为 '-'，即只有实线
LineWidth	X，Y 和 Z 坐标轴的宽度，默认值为 0.5 点
Title	坐标轴标题文本对象的句柄
View	向量[az el]，它代表了观察者的视角，以度为单位。az 为方位角或视角相对于负 Y 轴向右的转角；el 为 X-Y 平面向上的仰角。详细细节见第 6.2 节
Xcolor	RGB 向量或预定的颜色字符串，它指定 X 轴线、标志、刻度标记和格栅线的颜色。默认为 white(白色)
Xdir	X 值增加的方向。{normal}：X 值从左向右增加；{reverse}：X 值从右向左增加
Xform	一个 4×4 的视图转换矩阵。设置 view 属性影响 Xform
Xgrid	X 轴上的格栅线。on：X 轴上每个刻度标记处画格栅线；{off}：不画格栅线
Xlabel	X 轴标志文本对象的句柄

续表7.6

属性名称	含　义
Xlim	向量[xmin xmax]，指定 X 轴最小值和最大值
Xtick	数据值向量，按此数据值将刻度标记画在 X 轴上，将 Xtick 设为空矩阵就撤消刻度标记
XtickLabels	文本字符串矩阵，用在 X 轴上标出刻度标记。如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
Y…	与 X 轴一样
Z…	与 X 轴一样

表 7.8　线条对象属性

属性名称	含　义
LineStyle	线形控制。{-}：画通过所有数据点的实线；--：画通过所有数据点的虚线；:：画通过所有数据点的点线；-.：画通过所有数据点的点划线；+：用加号作记号，标出所有的数据；o：用圆圈作记号，标出所有的数据点；*：用星号作记号，标出所有的数据点；.：用实点作记号，标出所有的数据点；X：用 X 符号作记号，标出所有的数据点
LineWidth	以点为单位的线宽，默认值是 0.5
MarkerSize	以点为单位的记号大小，默认值是 6 点
Xdate	线的 X 轴坐标的向量
Ydate	线的 Y 轴坐标的向量
Zdate	线的 Z 轴坐标的向量

表 7.9　文本对象属性

属性名称	含　义
Extent	文本位置向量[left,bottom,width,height]，[left,bottom]代表了相对于坐标轴对象左下角的文本对象左下角的位置，[width,height]是包围文本串的矩形区域的大小，单位由 Units 属性指定
HorizontalAlignment	文本水平对齐。{left}：文本相对于它的 Position 左对齐；{center}：文本相对于它的 Position 中央对齐；{right}：文本相对于它的 Position 右对齐
Position	两元素或三元素向量[X　Y　(Z)]，指出文本对象在三维空间中的位置，单位由 Units 属性指定
Rotation	以旋转度数表示的文本方向。{0}：水平方向；±90：文本旋转±90 度；±180：文本旋转±180 度；±270：文本旋转±270 度
String	要显示的文本串
VerticalAlignment	文本垂直对齐。{top}：文本串放在指定的 Y 位置顶部；{cap}：字体的大写字母的高度在指定的 Y 位置；{middle}：文本串放在指定的 Y 位置中央；{baseline}：字体的基线在指定的 Y 位置；{bottom}：文本串放在指定的 Y 位置底部

表 7.10 曲面对象属性

属性名称	含 义
Cdata	指定 Zdata 中每一点颜色的数值矩阵。如果 Cdata 的大小与 Zdata 不同，Cdata 中包含的图像被映射到 Zdata 所定义的曲面
LineWidth	边缘线的宽度，默认值是 0.5 点
MarkerSize	边缘线的记号大小，默认值是 6 点
Xdata	曲面中点的 X 坐标
Ydata	曲面中点的 Y 坐标
Zdata	曲面中点的 Z 坐标

表 7.11 块对象属性

属性名称	含 义
Cdata	指定沿块边缘每一点颜色的数值矩阵。只有 EdgeColor 或 FaceColor 被设为 interp 或 flat 时才使用
LineWidth	轮廓线的宽度，以点为单位。默认值为 0.5 点
Xdata	沿块边缘点的 X 坐标
Ydata	沿块边缘点的 Y 坐标
Zdata	沿块边缘点的 Z 坐标

表 7.12 图像对象属性

属性名称	含 义
Cdata	指定图像中各元素颜色的值矩阵。Image(c)将 c 赋给 Cdata。Cdata 中的元素是当前颜色映像的下标
Xdata	图像 X 数据；指定图像中行的位置。如忽略，使用 Cdata 中的行下标
Ydata	图像 Y 数据；指定图像中列的位置。如忽略，使用 Cdata 中的列下标

7.2.4 图形对象属性的设置和使用

当建立一个对象时，它有一组默认属性值，该值可以用两种方法来改变。一是可以用{属性名，属性值}对来建立对象生成函数；另一个是在对象建立后改变属性。

第一种方法很简单，不再另做具体介绍。值得注意的是句柄图形对象创建函数(例如 figure、axis、line 等)接受多个属性名和属性值对。下面的一个小例子可以说明问题。

【例 9】建立对象生成函数

```
Hf_fig1=figure('Color','blue','NumberTitle','off','Name',
'My Figure')
```

说明：

创建一个图形窗口，背景为蓝色，标有 My　Figure 而不是默认标题 Figure　No.　1。

本节着重介绍第二种方法。MATLAB 系统中为了获得和改变句柄图形对象的属性，只需要两个函数 get 和 set。

- 函数 get 返回某些对象属性的当前值。使用函数 get 的格式是：

  get(handle, 'PropertyName')　　返回指定对象属性的当前值。

  get(handle)　列出对象所有的属性。

【例 10】：利用函数 get 返回某些对象属性。

```
Hf_fig1=figure(3);              %产生一个句柄为 3 的图形窗口
x=[1,2];
y=[122,144];
Hl_line1=line(x,y);             %在图形窗口中画一条二维直线
p=get(Hf_fig1, 'position' )     %返回具有句柄 Hf_fig1 图形窗口的位置向量
c=get(Hl_line1, 'color' )       %返回具有句柄 Hl_line1 对象的颜色
p =
   40    15   560   375
c =
    0     0     1
```

输入结果如图 7.10 所示。

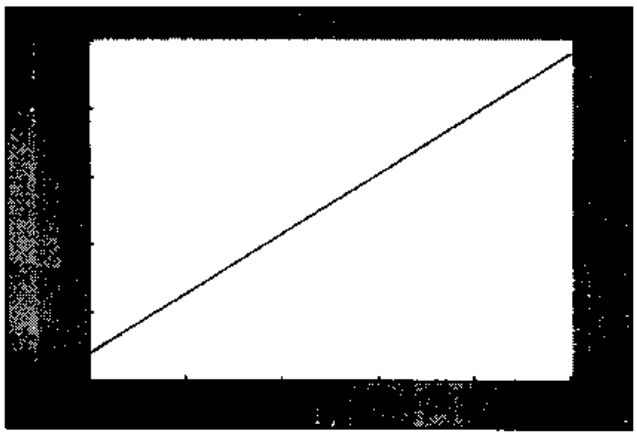


图 7.10　获得图形对象的属性

说明：

图形窗口中线条的颜色是蓝色，因此返回的颜色值是[0 0 1]。

- 使用函数 set 的格式是：

  set(handle, 'PropertyName' ,value)　　改变句柄图形对象属性。

  set(handle, 'PropertyName')　　返回一个可赋给对象的属性值列表。

  set(handle)　列出对象所有可设置的属性和可能的取值。

一般情况下，函数 set 的第一种调用格式可以有任意的('PropertyName' ,PropertyValue)对。而在第二种格式中如果指定一个没有固定值的属性，那么 MATLAB 系统就会给出提示信息通知用户。

提示： 注意到函数 set 和函数 get 返回不同的属性列表。函数 set 只列出可以用 set 命令改变的属性，这一类属性只可读，但不能被改变，它们叫做只读属性；而 get 命令列出所有对象的属性。

【例 11】使用函数 set 来设置图形对象的属性。

```
p_vect=[20  45  300  285]
    set(Hf_fig1, 'Position' ,p_vect)
```

说明：

将具有句柄 Hf_fig1 的图形位置设为向量 p_vect 所指定的值。由于屏幕显示的关系，无法向用户演示图形窗口的变化，用户可以自己试验。

```
set(Hl_line1, 'Color', 'r', 'Linewidth',2, 'LineStyle', '--' )
```

说明：将具有句柄 Hl_line1 的线条变成红色，线宽为 2 点，线型为破折号。

```
set(Hf_fig1, 'Units' )
[ inches | centimeters | normalized | points | {pixels} | characters ]
```

说明：表明由 Hf_fig1 所引用的图形的 Units 属性是 6 个可允许的字符串，而其中 'pixels' 是默认值。

```
set(Hf_fig1, 'Position' )
A figure's "Position" property does not have a fixed set of property
values.
```

说明：表明由 Hf_fig1 所引用的图形的 Postion 属性没有固定值，MATLAB 系统提示用户信息，但请注意不是出错信息。

图 7.11 是本例执行完成后的图形。

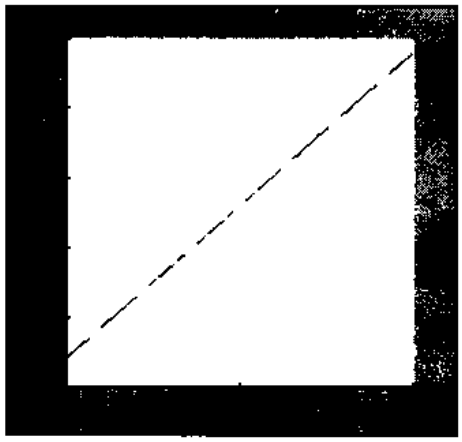


图 7.11 图形对象属性的改变

综合上节提供的图形对象的属性和本节中的属性设置函数，可以看出如何在 MATLAB 中创建和使用一个图形对象。考虑下面的例子。

【例 12】

```
Hf_fig =figure        %产生一个具有整数句柄的图形窗口对象
Hl_line=line          %产生一个具有浮点句柄的直线对象
Hf_fig =
```

```
     1
Hl_line =
72.0007
```

它要用非标准颜色画一条线。在这里，线的颜色用 RGB 值[1.50]来指定，它是适中的橘黄色。

```
x=[-2:0.01:2]*pi;
y=sin(x);                                    %产生数据
Hl_sin=plot(x,y)                             %画图并保存图形句柄
Hl_sin =
   72.0013
set(Hl_sin, 'Color' ,[1 .5 0], 'LineWidth' ,3)  %改变线段颜色和线宽
```

现在加一个紫色的 cos 曲线：

```
z=cos(x);                                    %cos 曲线的数据
hold on                                      %保留 sin 曲线
Hl_cos=plot(x,z);                            %画出 cos 函数曲线并获得句柄
set(Hl_cos, 'Color' ,[0.75  0  1])           %设置颜色为紫色
hold off
title('Sin 和 Cos 函数曲线' , 'Fontsize' ,16, 'Color' , 'green' )
%给当前图加一个 16 点的绿色标题
```

输出结果是图 7.12。

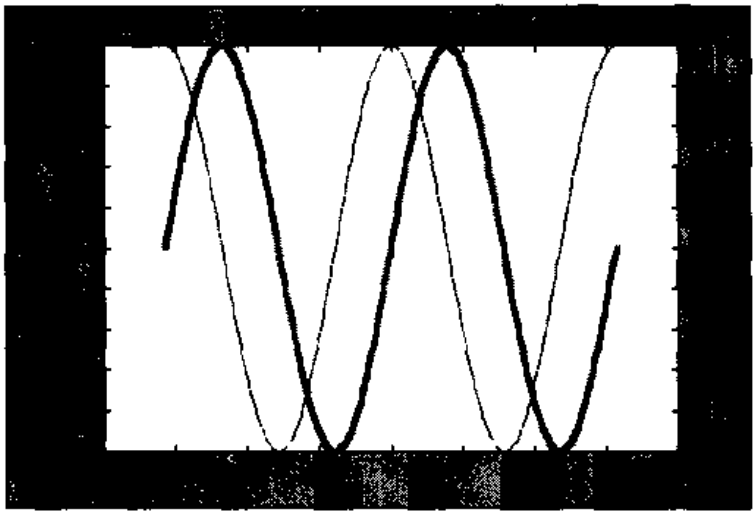


图 7.12　句柄图形对象

提示： 每一个对象都含有 Parent 属性和 Children 属性，该属性包含属于派生对象的句柄。画在一组坐标轴上的线，具有当作 Parent 属性值的坐标轴对象的句柄，而 Children 属性值是一个空矩阵。同时，这个坐标轴对象具有当作 Parent 属性值的图形句柄，而 Children 属性值是线条对象的句柄。标题字符串和坐标轴的标志不包含在坐标轴的 Children 属性值里，而是保存在 Title、Xlabel、Ylabel 和 Zlabel 的属性内。创建坐标轴对象时，这些文本对象就建立。title

命令设置当前坐标轴内标题文本对象的 String 属性。最终，标准 MATLAB 的函数 title、xlabel、ylabel 和 zlabel 不返回句柄，而只接受属性和数值参量。

7.3 动　画

动画是图形处理中最让人激动的部分。图形由静到动是质的飞跃。它要求用户有更全面的知识和灵感，同时也给用户提供科学和美相互协调的全新想象空间，从而激发出前所未有的创造力。

动画制作有两种方法。一是预先将图形制作好，放到图形缓冲区中，然后一帧一帧地播放。这种方法的计算量大，占用的内存多。另一种是保持整个背景图案不变，只更新运动部分的图形，以便加快整幅图的实时生成速度，这就是实时动画技术。本节将分别介绍这两种动画的制作，即动态图形的生成和实时动画的制作。

7.3.1 动态图形

在 MATLAB 系统中彗星轨迹函数、色图变幻函数和影片动画函数，能很方便地使图形及色彩产生动态变化效果。下面分别具体介绍这 3 种函数的格式和功能。

1. 彗星轨迹函数

彗星轨迹函数能动态地展现质点的运动轨迹。该函数的基本调用格式是：

comet(x,y,p)　　长为 p*length(y)的二维彗星轨迹，p 的默认值为 0.1。

comet3(x,y,z,p)　　长为 p*length(z)的三维彗星轨迹，p 的默认值为 0.1。

在二维和三维函数中 x,p 以及 x,y,p 都可以省略。

【例 13】二维彗星曲线，见图 7.13。

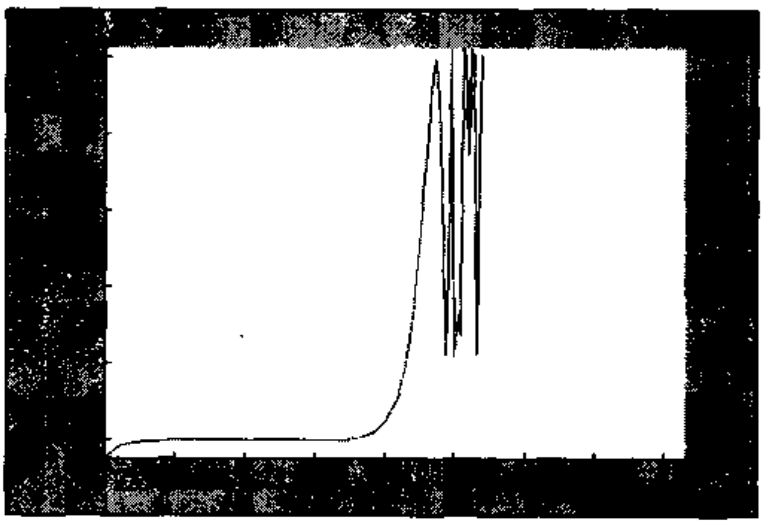


图 7.13　彗星函数演示

```
t=-1:0.02:1*pi;
```

```
comet(t,tan(sin(t))-sin(tan(t)))
```

2. 色图的变幻

MATLAB 系统为颜色的动态变幻提供函数 spinmap。它的功能是使当前图形的色图做循环变幻，以产生动态效果。该函数只涉及对色图的操作。spinmap 函数的调用格式有以下 4 种：

spinmap　　使色图周期旋转约 3s。

spinmap(t)　　使色图周期旋转约 ts。

spinmap(inf)　　使色图周期无限地旋转下去，用 Ctrl+C 键中断。

spinmap(t,inc)　　分别用 t,inc(默认为 2)控制色图旋转的时间和快慢，inc = 1 慢速变幻, inc = 3 快速变幻,inc = -2 反方向变幻。

【例 14】色彩的变幻(见图 7.14)。

```
pcolor(peaks)
spinmap(inf,2)
```

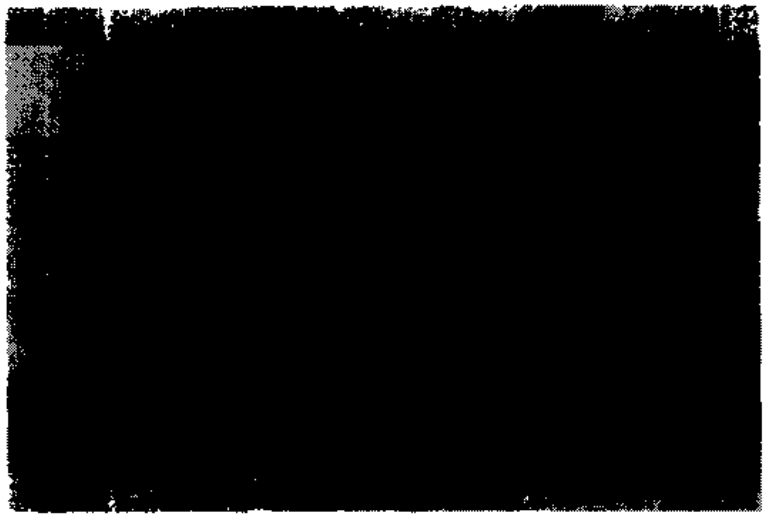


图 7.14　色图变幻中

提示： 色图的动画仅仅在 256 色的屏幕上才可以看到，请用户在使用函数时把 Windows 屏幕设置为 256 色，否则 MATLAB 系统会给出如下的警告信息：“Warning: Colormap animation is only possible for 256 color screens”。

3. 影片动画

MATLAB 系统支持影片动画(movie)的制作和放映。影片动画的制作函数有两个：moviein 和 getframe。而 movie 函数用来放映制作好的影片动画。具体过程如下：

(1) 由 M=moviein(n)创建具有 n 列的矩阵 M，准备用于储存 n 帧画面。

(2) 用 plot_command，M(j)=getframe 把制作影片动画用的第 j 帧画面像素(用 plot_command 函数绘出的图形)以列的方式保存在矩阵 M 中。这一步的作用主要在于开辟内存空间，以防止占有太多的内存。

(3) 运行影片放映函数 movie(M,k)。它使得矩阵 M 中储存的画面连续播放 k 次。

【例 15】影片动画的制作和放映。

```
n=12;
m=moviein(n);
t=0:2/n*pi:4*pi;
x=0:1/n*pi:4*pi;
nj=length(x);
for i=1:n
  for j=1:nj
   y(j)=sin(x(j)-t(i));
  end
  plot(x,y)
axis([0,4*pi,-1.5,1.5]);
m(i)=getframe;
end
movie(m,20)
```

说明：

本例制作的动画是向右移动的正弦行波，如图 7.15 所示。

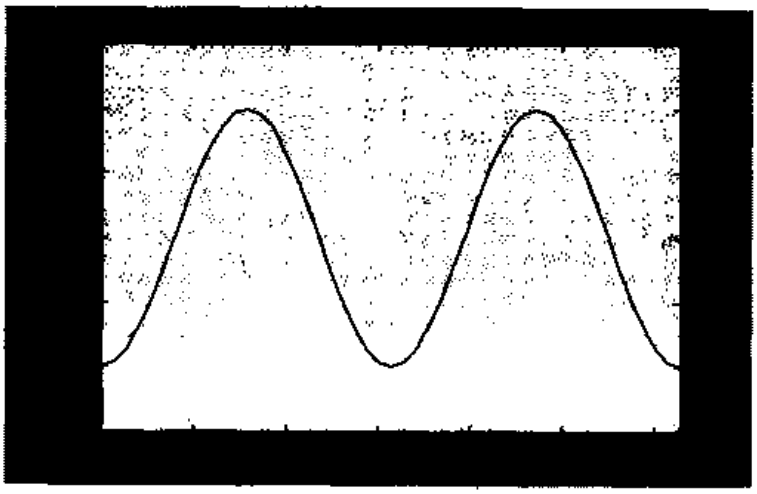


图 7.15　影片动画演示中

7.3.2　实时动画制作

改变某几个图形对象而不破坏图形的具体步骤：(1)计算活动对象的新位置，并在新位置上把它显示出来；(2)擦除原位置上原有的对象，刷新屏幕；(3)重复步骤 1 和步骤 2。

对于一般的计算机编程语言来说，显示新对象，擦除旧对象，而又不破坏背景图案不是一件轻松的事情。然而在 MATLAB 图形系统中却轻而易举，只需在创建对象时指定它的擦除方式(EraseMode)就可以了。

MATLAB 为 EraseMode 属性设置了 4 个值：

- normal 方式

使用该选项后，重画整个显示区。这种模式产生的图形最准确，但速度较慢。

- background 方式

将旧对象的颜色变成背景颜色，从而达到擦除的目的。这种模式将损坏被擦除对象下面的图形对象。

- xor 方式

对象的绘制和擦除由该对象的颜色和屏幕的颜色的异或而定。只画和屏幕颜色不一致的新对象点；只擦除与屏幕的颜色不一致的原对象点。该方式不损坏被擦除对象下面的其他图形对象。

- none 方式

不作任何擦除。

当新对象的属性设置好后，应该刷新屏幕，使新对象显示出来。MATLAB 用函数 drawnow 来刷新屏幕。drawnow 函数迫使 MATLAB 系统暂停目前的任务而去刷新屏幕。若没有 drawnow 函数，MATLAB 系统要等任务序列执行完成后才会去刷新屏幕。

【例 16】实时动画制作——弹簧振动系统模拟。

当质量为 M 的物体连接在刚度为 K 的弹簧上，物体初始距平衡位置有一个位移。如果不考虑物体和平面的摩擦，则物体将做相对平衡位置的理想简谐运动。其运动方程是：

$$M\ddot{x} + Kx = 0$$

其解为：

$$x = x_0 \cos\sqrt{\frac{K}{M}}t$$

其中，x_0=x|t=0。参数选择 K=M，x_0=5。

程序如下：

crtanim.m

```
function  crtanim
offset=4;                    %滑块的大小
animinit('onecart Animation');
onecart = findobj('Type','figure','Name','onecart Animation');
axis([-10 20 -7 7]);
hold on;
xySpr1=[ ...
        0         0
        .4        0
        .8        0.65
        1.6       -0.65
        2.4       0.65
        3.2       -0.65
        3.6        0
        4.0        0];    %弹簧的横纵坐标
```

```
    xyBx11=[ ...
          0          1.2
          0          -1.2
          0          0];          %滑块的数据
    xyBx21=[ ...
          0          1.2
          2          1.2
          2          -1.2
          0          -1.2
          0          1.2];        %滑块的数据
    xBx11=xyBx11(:,1);
    yBx11=xyBx11(:,2);
    xBx21=xyBx21(:,1);
    yBx21=xyBx21(:,2);
    xSpr1=xySpr1(:,1);
    ySpr1=xySpr1(:,2);
    x=[xBx11; xSpr1; xBx21(:,1)+offset];
    y=[yBx11; ySpr1; yBx21];
    %画出滑动物体下的水平面
    plot([-10 20],[-1.4 -1.4],'blue', ...
    [-10:19;-9:20],[-2 -1.4],'blue','LineWidth',2);
    hold on;
    %画出弹簧和滑块
    hndl=plot(x,y,'y','color','blue','EraseMode','background','LineW
idth',3);
    set(gca,'UserData',hndl);
    set(gcf,'color','yellow');          %设置背景颜色
    t=0;
    dt=0.02;
    u0=5;
    while t<10
        if any(get(0,'Children')==onecart),
            if strcmp(get(onecart,'Name'),'onecart Animation'),
            set(0,'currentfigure',onecart);
            t=t+dt;
            u=u0+5*cos(t);
            u=u+offset;
            distance=u;
            hndl=get(gca,'UserData');
```

```
                x=[xBx11;
                  xSpr1/4*distance;
                  xBx21+distance];
                set(hndl,'XData',x);
                drawnow;
                    if(t==0)
                    for(p=0:0.1:10*pl);
                    end                                    %在初始位置停留一段时间
                end
            end
        end
```

输出结果如图 7.16 所示。

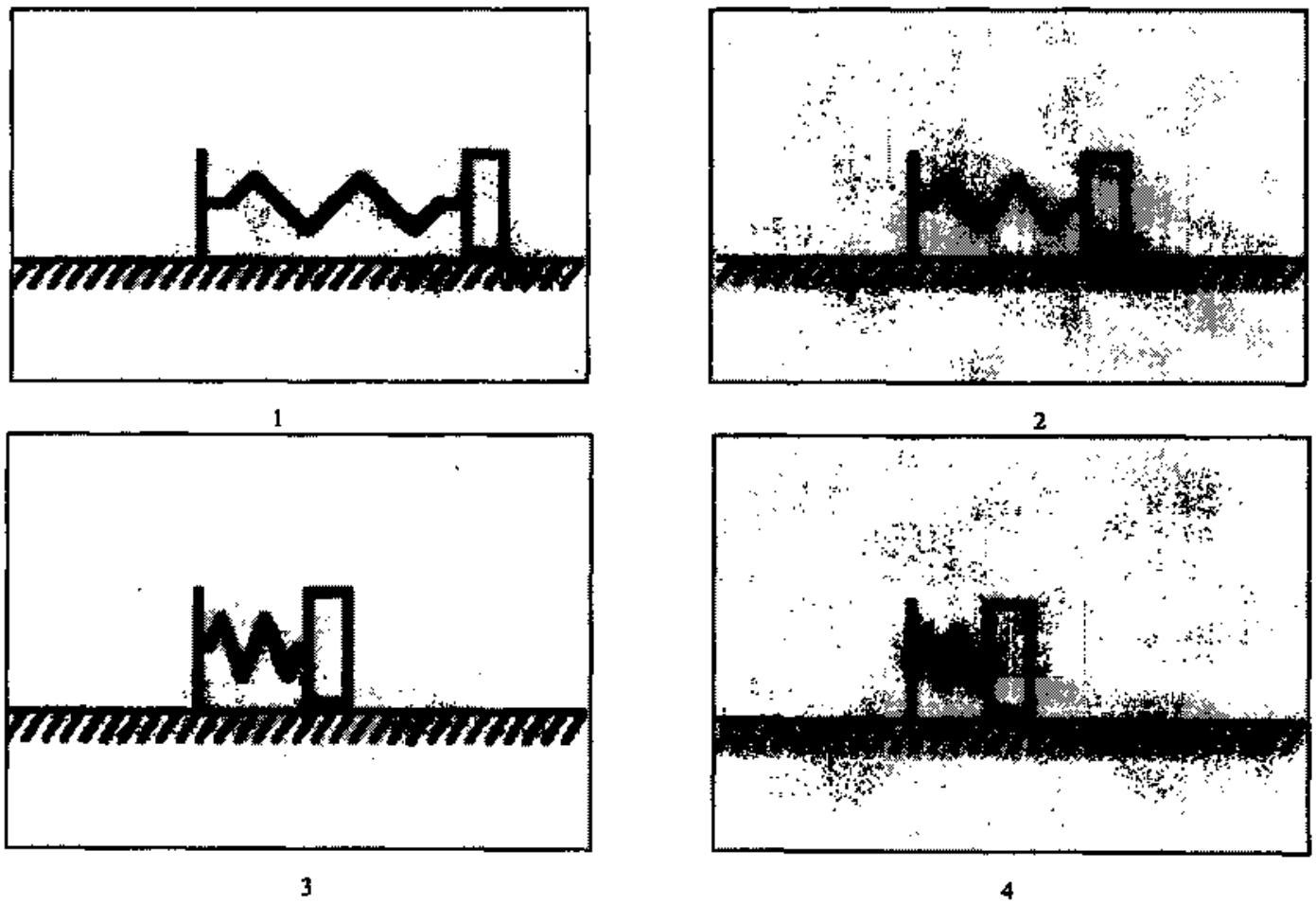


图 7.16　实时动画弹簧振动系统的演示

说明：

本例中设置了运行的时间，大约为 10 秒。用户可以通过改变循环的次数来改变运行的时间。

第 8 章
MATLAB 的接口

本章要点：

MATLAB 是一个开放的系统，它具有多种接口功能，使用户可以十分方便地与其他的应用程序交换数据和信息。理解 MATLAB 的文件格式和操作方法，有助于对 MATLAB 的进一步开发。而掌握 MATLAB 的各种接口，则能充分利用系统资源，编写出功能更强大，结构更简洁的程序。

本章具体包括以下内容：

- MATLAB 的数据接口
- 文件的 I/O 操作
- MEX 程序的编写

MATLAB 是一个开放的系统，它具有多种接口功能，用户可以十分方便地与其他的应用程序交换数据和信息。它的接口主要有数据输入输出接口、子程序库调用接口、图形图像输入输出接口和动态链接接口等。本章主要介绍数据输入输出接口和子程序调用接口。

8.1 MATLAB 的数据接口

MATLAB 的数据对象是数组。许多科学计算任务都需要处理大批量的数据。如何将这些数据输入到 MATLAB 的工作区呢？在 MATLAB 环境中，要视具体情况而定。

MATLAB 接受两种形式的数据：一种是 ASCII 码的文本数据文件，这种文件每一行的数据项数都是相同的，每个数据项由一个或多个空格分隔开，其内容恰好构成一个矩阵；另一种数据形式是 MATLAB 定义的 MAT 型数据，即所谓的 MAT 文件。首先介绍 MATLAB 系统对 MAT 数据文件的处理方法，而后介绍 MATLAB 的数据文件操作功能。

8.1.1 数据结构

了解 MATLAB 的内部数据结构和数据对象，对于进行数据交换操作是十分必要的。MATLAB 5.x 只支持单一的数据对象类型——数组(Array)。MATLAB 5.x 定义了如下几种变量类型：矩阵(包括数量和向量)、字符串、单元数组和结构。这些变量类型在 MATLAB 内部都是用数组来存储和管理的。

在 C 语言的意义下，变量类型 mxArray 定义 MATLAB 数组的内部数据结构，即任何一个 MATLAB 变量定义为 C 语言意义下的 mxArray 结构类型。mxArray 结构包含以下的结构域：

- MATLAB 变量名 Name，这是指向一个字符串的指针，字符串的最大长度由 MATRIX.H 头文件中的常数 mxMAXNAM 规定。这个字符串就是 MATLAB 的变量名字。
- 变量的维数 Dimensions，定义各维数大小。例如数量、向量和矩阵都视为二维的。
- 变量的类型 ClassName，这是一个标识值，指明变量被显示时是数值变量还是字符型变量，即将变量元素看成是字符的 ASCII 码。
- 变量的实或复数类型。如果变量包含有复数值作为其元素，那么该变量对应的 MATLAB 内部数组就包含有实部向量和虚部向量。
- 变量的存储属性 Storage，即是否是稀疏矩阵。指明变量的存储类型，即变量是按全元素矩阵还是按稀疏矩阵的方式存储。它的取值是 Full 或 Sparse。

数据矩阵定义为 M×N 数组，它是 MATLAB 语言的基本操作对象。矩阵的含义与通常的数学意义相同，可以是长方矩阵，也可以是方阵，以及单个数量。矩阵的类型可以是复数型矩阵、实数型矩阵或由字符组成的字符型矩阵。在 MATLAB 系统中，矩阵有两种不同的存储类型，即全元素矩阵的存储方式或稀疏矩阵的存储方式等。数量、向量和字符串等都是特殊的矩阵，它们都是按矩阵的结构来组织和管理的。

如果 MATLAB 变量是全元素矩阵，那么 mxArray 数组结构类型包含有参数 pr 和 pi。pr 是指向实部向量的指针，pi 是指向虚部向量的指针。如果 pi 指针为空(NULL)，则表示矩阵是实数型变量。如果 MATLAB 变量是稀疏矩阵，则 mxArray 除了有 pr 和 pi 参数外，还有另外 3 个参数：nzmax、ir 和 jc。它们的具体含义如下：

nzmax　定义 pr 和 pi(如果 pi 存在)所指向量的长度，是稀疏矩阵中“非零”元素的最大可能总数

pr　指向长度为 nzmax 的双精度型向量的指针，该向量按列由稀疏矩阵中“非零”元素的实部构成

pi　指向长度为 nzmax 的双精度型向量的指针，该向量按列由稀疏矩阵中“非零”元素的虚部构成。如果指针为 NULL，表示是实数型矩阵

ir　指向长度为 nzmax 的整数型向量的指针，其向量元素值为 pr 和 pi 所指的向量中的对应元素在原矩阵中的行指标

jc　指向长度为 N+1 的整数型向量的指针，包含矩阵元素的列信息。对于 0≤j≤N-1，jc[j]记录着矩阵第 j+1 列的第一个“非零”元素在 pr 和 pi 向量中的位置指标；jc[j+1]-1 是该列最后一个“非零”元素在 pr 和 pi 向量中的位置指标；jc[N]定义为原矩阵变量中“非零”元素的总数 nnz

【例 1】矩阵对象的属性值

在 MATLAB 中，提供了一个很有用的命令 explore，它不在默认的路径中，可以按照如下方法使用：

```
cd([matlabroot '/extern/examples/mex']);
explore(struct('name', 'Joe Jones', 'ext', 7332))
```

执行结果如下：

```
Name: Unnamed
Dimensions: 1x1
Class Name: struct
------------------------------------------------
(1,1).name
------------------------------------------------
                    Name: Unnamed
Dimensions: 1x9
Class Name: char
------------------------------------------------
(1,1) Joe Jones
(1,1).ext
------------------------------------------------
                    Name: Unnamed
Dimensions: 1x1
Class Name: double
------------------------------------------------
```

```
(1,1) = 7332
```

8.1.2 MATLAB 的数据输入

用户可以用多种方式向 MATLAB 系统输入数据。最佳的输入方式依赖于用户的数据格式。下面给出几种可以选用的方法。

1. 显式的输入

针对数据量比较小的情形，可以在 MATLAB 的命令窗口中，从键盘直接输入待输入的数据。这种方法就是前面介绍的矩阵的直接输入法，即使用方括号将矩阵的元素按行的顺序括起来，每行用分号隔开，行内各元素用空格或逗号分开。

2. M文件形式输入

如果数据量较大，且不是以计算机可读形式存在时，最有效的办法是用某种文本编辑器直接编写一个包含数据矩阵的 M 文件。这种方法与 8.1.1 节所述的方法是类似的。所不同的是，前者是在命令窗口中实时地写 MATLAB 的矩阵输入语句，后者是编写矩阵输入语句的 M 文件，最后可以通过执行 M 文件达到数据输入的目的。

3. ASCII码数据文件的输入

MATLAB 可以直接读入 ASCII 码的数据文件。ASCII 码的数据文件中的数据形式必须是一个矩阵，要求数据文件每一行的数据个数必须相同，每行数据对应于矩阵的每一行，每行的元素用空格分开。用户可以将数据编辑成一个 ASCII 码文件，或者由其他的程序将所输入的数据写到一个 ASCII 码文件中。ACSII 码文件中的数据由 MATLAB 的命令 load 装入，命令形式是：

```
load 文件名(扩展名)
```

该语句在 MATLAB 工作区中创建一个与文件名(无文件扩展名)相同的变量，该变量表示的矩阵即是 ASCII 码文件的数据组成的矩阵。

4. 低层I / O输入方式

MATLAB 提供了文件低层操作函数。可以直接打开文件(fopen)和读文件(fread)，这种方法主要用于装入某种特定格式的数据文件，这种数据文件可能是其他的应用程序生成和创建的。

5. MEX动态程序输入

如果已经有一些子程序(如 C 子程序或 FORTRAN 子程序)可以用来读取某些特定格式的数据文件，那么可以开发 MATLAB 的动态链接 MEX 子程序，与已有的子程序链接在一起，将数据文件转换成 MATLAB 的 MAT 数据文件，再用 load 函数将有关的数据装入到 MATLAB 系统中。

6. 外部程序转换

如果已有的数据文件格式比较复杂。用户可以开发 FORTRAN 或 C 程序将数据文件直接转换成 MATALB 的 MAT 数据文件，再用 load 命令装入到 MATLAB 系统中。在这种情况下，用户须对 MAT 文件结构有较深入的了解。

8.1.3 MATLAB 的数据输出

MATLAB 系统输出数据的方式主要有以下 5 种。

1. 小型矩阵输出

对于数据量较小的数据矩阵，通过输出的设置，可以用 diary 命令生成命令窗口部分内容的副本文件。该文件是文本文件，可以直接用普通文本编辑软件修改。它记录了命令窗口的命令行，以及 MATLAB 的屏幕输出内容。这样可以将 diary 文件剪接到其他的文件或报告中。

2. ASCII码数据输出

利用 ASCII 选项的 save 命令，可以生成一个 ASCII 码的数据文件。在这种方法中，用户可以存储某些指定的变量。例如下列语句

```
A=rand(4,3)
save temp.dat A -ascii
```

将生成名字为 temp.dat 的 ASCII 码文件，包含矩阵 A 的全部数据，如：

```
0.9501    0.8913    0.8214
0.2311    0.7621    0.4447
0.6068    0.4565    0.6154
0.4860    0.0185    0.7919
```

3. 低层I/O输出

利用 MATLAB 的低层 I/O 函数 fopen 和 fwrite，或者其他低层 I/O 函数，可将数据写入某个特定格式的文件中。这种方式主要用于将输出的数据写成某些其他应用程序所需要的数据格式。

4. MEX程序输出

如果已经有某些子程序可以将数据写成特定格式，那么可以开发 MEX 子程序直接调用那些子程序，把最后形成的数据格式作为某些应用程序的输入数据。

5. MAT格式输出

利用 MATLAB 的 save 命令，可以将数据存储成 MAT 格式文件，再开发一些 C 或 FORTRAN 程序将 MAT 文件转换成某些所需要的特殊格式。例如，将 MATLAB 的图像

数据存入到 MAT 文件中，然后利用 C 或 FORTRAN 程序将 MAT 的图像数据转换成一些标准的图像格式文件，如 PCX、TIF 和 GIF 格式等。

注意： 方式 2 中所述的方法仅仅适用于数值型的矩阵，对于其他形式，例如结构(struct)或是单元(cell)数组，均不适用。但是第 5 种方式具有通用性。

8.1.4 MAT 数据格式

MAT 数据格式是 MATLAB 的数据存储的标准格式。一个 MAT 文件中可以存储一个或多个矩阵数据，矩阵顺序地存在一片连续的磁盘空间上。在每个矩阵的开始处，有一个固定长度的矩阵信息头，这个信息头完整地描述该矩阵的全部特征信息。信息头之后才是矩阵的数据部分。数据占用的磁盘空间字节长度由信息头的长度决定。整个这样的结构就构成 MAT 文件中的一个矩阵。

MATLAB 的 save 命令可以将 MATLAB 系统内部数据写为 MAT 文件，而 load 命令可以将磁盘上的 MAT 文件正确地读入到 MATLAB 系统中。除此之外，为了有效地管理 MAT 文件，以及在 MATLAB 外部读取和创建 MAT 文件，MATLAB 提供了一个子程序库，用户可以在 C 或 FORTRAN 程序中直接调用这些子程序来创建和读取 MAT 文件。当然，如果用户了解 MAT 文件的结构，则完全可以开发自己的子程序读写 MAT 文件。但是，使用 MATLAB 提供的子程序库中的标准子程序会更有效。

MATLAB 子程序库中包含了 11 个子程序，它们的名字和所提供的功能说明见表 8.1。

表 8.1 MAT文件操作子程序

C 语言库函数	FORTRAN 语言库函数	功　能
matOpen	matOpen	打开 MAT 文件
matClose	matClose	关闭 MAT 文件
matGetDir	matGetDir	取得 MAT 文件中的数组列表
matGetFp		取得 MAT 文件的 C 语言文件句柄
matGetArray	matGetMatrix	从 MAT 文件中取一个数组
matPutArray	matPutMatrix	向 MAT 文件中存一个数组
matGetNextArray	matGetNextMatrix	从 MAT 文件中取下一个数组
matDeleteArray	matDeleteMatrix	从 MAT 文件中删除当前数组
matPutarrayAsGlobal		向 MAT 文件中存入一个数组。当用 load 命令装入这个 MAT 文件时，该数组对应的变量成为 Global 变量
matGetArrayHeader		读取 MAT 文件中数组的头信息
matGetNextArrayHeader		读取 MAT 文件中下一个数组的头信息
	matGetString	从 MAT 文件中取出一个字符串
	matPutString	向 MAT 文件中写入一个字符串

注意：●由于 MATLAB 5.x 将数组结构作为 MATLAB 的基本数据对象，所以在 MAT 文件中以数组变量为储存单元。MAT 子程序库中包含"Array"。

●有关的程序及目标程序放在 MATLAB 目录下的 extern 目录中。目录 extern 的子目录 include 下是描述数组访问的有关头文件 matrix.h 和 mat.h。在子目录 bin 下存有子程序动态连接库文件:

libmat.dll	MAT 文件子程序库
libmx.dll	MAT 文件数组访问子程序库

【例 2】MAT 子程序库的使用

```
#include <stdio.h>
#include <stdlib.h>
#include "mat.h"

int diagnose(const char *file) {
 MATFile *pmat;
 char    **dir;
 int     ndir;
 int     i;
 mxArray *pa;

 printf("Reading file %s...\n\n", file);
 /*  打开MAT文件   */
 pmat = matOpen(file, "r");
 if (pmat == NULL)
{
   printf("Error opening file %s\n", file);
   return(1);
 }
 /* 取得MAT文件中的数组列表   */
 dir = matGetDir(pmat, &ndir);
 if (dir == NULL)
{
   printf("Error reading directory of file %s\n", file);
   return(1);
 }
else
{
   printf("Directory of %s:\n", file);
   for (i=0; i < ndir; i++)
```

```
    printf("%s\n",dir[i]);
 }
 mxFree(dir);
 /* 为了能够正确读取某个数组信息，将文件重新打开，指针定位于文件头 */
 if (matClose(pmat) != 0)
{
  printf("Error closing file %s\n",file);
  return(1);
 }
 pmat = matOpen(file, "r");
 if (pmat == NULL)
{
  printf("Error reopening file %s\n", file);
  return(1);
 }
 /* 取得所有变量的头信息 */
 printf("\nExamining the header for each variable:\n");
 for (i=0; i < ndir; i++)
{
  pa = matGetNextArrayHeader(pmat);
  if (pa == NULL)
{
   printf("Error reading in file %s\n", file);
   return(1);
  }
  /* 分析数组的维数以及变量类型 */
  printf("According to its header, array %s has %d dimensions\n",
     mxGetName(pa), mxGetNumberOfDimensions(pa));
  if (mxIsFromGlobalWS(pa))
    printf("  and was a global variable when saved\n");
  else
    printf("  and was a local variable when saved\n");
  mxDestroyArray(pa);
 }
 /* 重新打开文件，读入数组 */
 if (matClose(pmat) != 0)
{
  printf("Error closing file %s\n",file);
  return(1);
```

```
 }
 pmat = matOpen(file, "r");
 if (pmat == NULL)
{
  printf("Error reopening file %s\n", file);
  return(1);
 }
 /* 读入每一个数组的内容 */
 printf("\nReading in the actual array contents:\n");
 for (i=0; i<ndir; i++)
{
    pa = matGetNextArray(pmat);
    if (pa == NULL)
{
     printf("Error reading in file %s\n", file);
     return(1);
    }
 if (matClose(pmat) != 0)
{
    printf("Error closing file %s\n",file);
    return(1);
 }
 printf("Done\n");
 return(0);
}
/* 主程序 */
int main(int argc, char **argv)
{ int result;
 if (argc > 1)
   result = diagnose(argv[1]);
 else
{
   result = 0;
   printf("Usage: matdgns <matfile>");
   printf(" where <matfile> is the name of the MAT-file");
   printf(" to be diagnosed\n");
 }
 return (result==0)?EXIT_SUCCESS:EXIT_FAILURE;
}
```

说明：

- 函数的用法：matdgns MAT-文件名。
- 本例演示了 matClose，matGetDir，matGetNextArray，matGetNextArrayHeader 和 matOpen 的用法。
- 具体的编译方法见后。

8.2 文件的 I/O 操作

MATLAB 系统具有直接对磁盘文件进行访问的功能。这样，用户不仅可以进行高层的程序设计，必要时也可以进行低层次的磁盘文件读/写操作，极大地增强了 MATLAB 程序设计的灵活性。

MATLAB 的文件 I/O 操作是基于 C 语言的文件 I/O 函数的，熟悉 C 语言的用户将会很快学会这种方法。即使用户不很熟悉 C 语言的文件操作功能，也很容易学会 MATLAB 的文件 I/O 操作。

8.2.1 文件的打开和关闭

根据操作系统的要求，在程序要求使用或创建一个磁盘文件时，必须向操作系统发出打开文件的命令。文件使用完毕后，必须告诉操作系统关闭使用过的文件。

在 MATLAB 中，用户可以使用 C 语言的同名函数 fopen 打开文件，供系统读写操作之用。函数 fopen 使用两个参数，第一个参数指明要打开的文件名，第二个参数是操作说明字符，指明操作方式。例如，下列命令

```
fid=fopen('logo.dat','r')
```

表示以只读方式打开 logo.dat 文件，供读数据操作之用。操作说明字符见表 8.2。

表 8.2 数据操作说明字符及其功能

操作字符	功　能
'r'	表示对文件进行读数据操作
'a'	表示对文件进行数据追加操作，在文件尾添加数据
'w'	表示对数据进行写操作，当文件不存在时创建文件
'r+'	表示对文件进行读与写操作
'w+'	表示创建一个新文件或是删除已存在的文件内容，并进行数据读写操作
'a+'	表示创建一个新文件、打开一个已存在的文件，并进行数据追加操作
'W'	表示对数据进行写操作，当文件不存在时创建文件，但是不自动移动指针
'A'	表示对文件进行数据追加操作，在文件尾添加数据，但是不自动移动指针

在有些操作系统上，要求在操作说明中区分文本和二进制文件。例如，rb 表示打开一个二进制文件，进行读数据操作。

正常情况下，函数 fopen 的返回值是一个非负的整数，是由操作系统设定的，作为打开的文件的标识值，或称为文件句柄值。对于该文件的任何操作，都是通过这个句柄值来传递的。MATLAB 通过这个句柄值来标识已打开的文件，实现对该文件的读、写和关闭等操作。

由下列命令

```
fids=fopen('all')
```

可以取得所有已打开的文件句柄值。

如果文件不能被正确地打开，例如，如果文件以 r 方式打开，而该文件并不存在，则会出现文件打开错误。这时，fopen 函数返回－1 作为该文件的句柄值。程序设计的良好习惯之一是每次打开文件时，都要进行打开操作是否正确的测定。fopen 函数还可以按下列方式调用：

```
[fid,message]=fopen('pen.dat','r')
```

输出变量 message 中包含了打开文件操作结果的有关信息。例如，如果文件 pen.dat 不存在，那么上述语句完成后，fid 的值为－1，而 message 的值是如下字符串：

```
No such file or directory
```

注意： 错误信息是与操作系统有关．MATLAB 函数 ferror 也可以提供有关的错误信息。一旦文件被打开，该文件就可以被用于读／写等操作。在完成对文件的读／写操作之后，必须关闭该文件．关闭文件的方法很简单，例如：

```
status=fclose('fid')
```

用函数 fclose 即可关闭句柄值为 fid 的已打开的文件。为了关闭所有已打开的文件，只需执行下列命令：

```
status=fclose('all')
```

如果关闭文件操作成功，则 status 的值为 0；否则 status 的值为－1。所以，与打开文件的操作一样，用户在程序设计中最好检查一下文件是否正常地关闭。

8.2.2 二进制数据文件的读／写操作

1. 读文件操作

MATLAB 函数 fread(与 C 语言函数同名)，可以用来读二进制文件，它有多种调用形式。最简单的形式为

```
fid=fopen('data.dat','r');
A=fread(fid);
status=fclose(fid);
```

将数据文件 data.dat 中的数据逐字节地按无符号字符型数据读入到矩阵变量 A 中。fread 还有两个可选择的操作参数，利用这两个参数可以控制读入数据的个数以及每个数据的精度。例如，以下语句

```
fid=fopen('data.dat','r');
A=fread(fid,100);
status=fclose(fid);
```

将 data.dat 文件中的前 100 个数据读入到列向量 A 中。如果参量 100 用维数向量[10，10]代替，则这 100 个数据将被按列的顺序读入到 10×10 的矩阵 A 中。另外，

```
A=fread(fid,Inf);
```

表示读至文件的结尾，生成一个列向量 A。

在 fread 中，还可以加入第三个参数，用来控制读入数据的精度。按数据精度可把数据分为字符型数据、整数型数据和浮点型数据。数据精度与计算机的硬件有关，用户对自己使用的计算机数据精度应有所了解。

MATLAB 支持各种数据精度类型，用户可以指明读入数据时采用的精度类型，有些与 C 语言或 FORTRAN 语言支持的数据精度类型相同。常用的数据精度类型见表 8.3。

表 8.3 MATLAB支持的数据精度类型

数据类型	说　明
char 和 uchar	有符号和无符号字符类型，数据精度为一个字节(8 bits)。对有符号字符类型，数据取值范围是-128～127；对无符号字符类型，数据取值范围是 0～255
single 和 float32	32 位精度的浮点数
double 和 float64	64 位精度的浮点数
intX	X 取 8、16、32、64，代表相应字长的整数
uintX	X 取 8、16、32、64，代表相应字长的无符号整数

如果 fid 表示的是已打开的浮点类型的数据文件，那么，

```
A=fread(fid,10,'float')
```

将读入前 10 个浮点型的数据保存到列向量 A 中。又如，下面的例子用来读包含函数 fread 的使用说明的文本文件 fread．m，然后在命令窗口中打印出来：

```
fid = fopen('fread.m','r');
F = fread(fid);
disp(setstr(F))
status=fclose(fid)
```

上述第二条语句等价于 F=fread(fid,Inf,'uchar')，即按无符号字符型数据处理。

2. 写文件操作

函数 fwrite 的作用是将一个矩阵的元素按一定的数据精度类型写入到某个打开的文件中，该函数返回写入的数据个数。例如：

```
fid = fopen('magic5.bin','wb')
count=fwrite(fid,magic(5),'integer*4');
status=fclose(fid);
```

将生成 100 字节长度的二进制文件，包含 5×5 个数据，即 5 阶方矩阵的数据。每个数据占用 4 个字节的存储单位，数据类型为整型，输出变量 count 的值为 25。

8.2.3 文件内的位置控制

根据操作系统的规定，在读/写数据时，默认的方式总是从磁盘文件的开始处顺序向后地在磁盘空间上读/写数据。操作系统控制一个文件指针，指出当前的文件位置。C 或 FORTRAN 语言都有专门的函数来控制和移动文件指针，达到随机访问磁盘文件的目的。

MATLAB 的函数 fseek 和 ftell 可以用来设置文件指针的位置，并取得文件指针的当前位置，该位置是下一次读/写操作的起始点。

函数 ftell 给出文件指针相对于文件中某个指定点的偏移量。fseek 函数根据指定的位置来重新设置文件的当前位置。这样，用户就可以跳过部分数据或返回到文件中前面的位置，达到随机访问的目的。这些函数的输入参数包括待操作的文件句柄值、正或负的偏移量和偏移相对点位置。相对点位置可以是文件指针的当前点，用 cof 表示；文件的开始点，用 bof 表示；文件的结束点，用 eof 表示。正的偏移量表示向文件尾的方向偏移，负的偏移量表示向文件头的方向偏移。偏移量的单位是字节。

【例 3】函数 fseek 和 ftell 使用举例

```
A=[1:5];
fid=fopen('five.bin','w');
fwrite(fid,A,'short');
status=fclose(fid);
fid=fopen('five.bin','r');
status=fseek(fid,6,'bof');
four=fread(fid,1,'short');
position=ftell(fid);
status=fseek(fid,-4,'cof');
```

```
three=fread(fid,1,'short');
status=fclose(fid);
```

说明：

在这个例子中，前面四条语句的作用是生成一个包含 5 个数据的文件，每个数据的二进制长度为 2 个字节。第一次调用 fseek 函数时，让文件指针跳过 6 个字节(包含数据 1、2 和 3)，以便进一步直接读数据 4。读数据的过程使文件指针向后移动了 2 个字节，所以 ftell 的结果应该是当前指针指到文件起始位置与文件首的偏移量，即 8 个字节。第二次调用 fseek 是将文件指针相对当前的位置向文件首的方向回移 4 个字节，即在数据 3 的位置，以便读取数据 3。

8.2.4 格式文件的输入和输出

1. 格式输出

MATLAB 的函数 fprintf(与 C 函数同名)的作用是将数据转换成字符串，输出到命令窗口屏幕或写入到一个文件中，通过一个格式说明字符串，说明输出的数据格式。格式说明控制矩阵数据的输出格式。格式说明通常由字符和以%开头的格式类型说明符组成。常用的格式说明符有：

%e　数据指数表示形式

%f　固定小数点位置的数据格式

%g　在前两种中自动选择比较短的一种

在格式说明中，还包含数据占用的最小宽度和数据精度的说明。

【例 4】fprintf 函数使用举例

```
x = 0:.1:1;
y = [x; exp(x)];
fid = fopen('exp.txt','w');
fprintf(fid,'指数函数表\n');
fprintf(fid,'%6.2f  %12.8f\n',y);
status=fclose(fid);
```

说明：

- 本例创建了一个文本文件 exp.txt，包含指数函数值的函数表。第一条 fprintf 语句输出一行标题，随后空一行；第二条 fprintf 语句输出函数表自身，每组变量和函数值占一行，都是固定小数点的格式。自变量占 6 个字符位，其小数点后的精度是 2 个字符位；函数值占 12 个字符位，小数点后的精度是 8 个字符位。自变量和函数值之间空两格。
- 矩阵 Y 的元素按照列的顺序转换成格式化输出，函数反复使用格式说明，直至将矩阵 A 的数据全部转换完毕。
- “\n”表示换行，“\r”、“\t”、“\b”、“\f”分别表示回车、制表符、退格、格式化文本；如果要输出“\”，可以用“\\”；如果要输出“%”，可以用“%%”。

与这个函数相关的另一个函数是 sprintf，它的作用是将输出的结果转换为一个字符串。例如：

```
roots=sprintf('The squrar root of %f is %6.4e.\n',2,sqrt(2));
dims=sprintf('The array is %dx%d.',2,3);
```

这里，roots 和 dims 都是字符串，其内容分别是

```
roots=The squrar root of 2.000000 is 1.4142e+000.
dims=The array is 2×3.
```

可以看到，这个函数对于字符串的构造十分有用。

2. 格式输入

MATLAB 的格式输入函数是 fscanf。其用法是以文件句柄作为第一个输入参数，格式说明作为第二个输入参数。格式说明通常由字符和以%开头的格式类型说明符组成。常用的格式说明符有：

%s　按字符进行输入转换

%d　按十进制数据进行转换

%f　按浮点数据进行转换

在格式说明中，除了单个的空格字符可以匹配任意个数的空格字符外，通常的字符在输入转换时与输出的字符进行一一匹配。函数 fscanf 将输入的文件看作是一个输入流(stream)，MATLAB 根据格式来匹配输入流，并将在流中匹配的数据读入到 MATLAB 系统中。下面是一个文件输入的例子。

【例 5】fscanf 函数使用示例

```
fid=fopen('exp.txt','r')
title=fscanf(fid,'%s');
[table,count]=fscanf(fid,'%d%f');
status=fclose(fid);
```

说明：

第一次调用 fscanf 时，文件 exp.txt 的标题行与“%s”匹配，解释为字符串，直到遇到回车符为止。第二次调 fscanf 时，读入文件数据表，每行数据按“%d %f” 方式匹配，即第一个数据按十进制数，第二个数据按浮点型数据解释，读入到变量 table 中，直到文件结束。count 返回匹配的或输入的数据个数。

另外，fscanf 中可以加入一个选项参数，用它来控制要读入的数据个数。例如，如果 fid 是一个已打开的文件，包含十进制的数据，那么，

```
A=fscanf(fid,'%5d',100);
```

将读入 100 个数据到列向量 A 中，而

```
A=fscanf(fid,'%5d',[10 10]);
```

将这 100 个数据送到 10×10 的矩阵 A 中。

另一个相关的函数是 sscanf，它的作用与 sprintf 相反，可以用它从一个字符串中读取有关的数据，例如：

```
rootvalue=sscanf(roots,'The squrare root of %f is %f.');
```

将返回包含 2 和其平方根的向量 rootvalue。

8.3 MEX 程序的编写

MATLAB 是自成体系的编程系统，有自己的数据结构。它不仅有如前面几节介绍的外部数据接口，还具有调用其他子程序的功能。这些功能被集成在它的应用程序接口(API)中，其中 MEX 动态链接是它的主要功能。

在 MATLAB 中，可以调用用户自己开发的 C 或 FORTRAN 子程序，通过 MATLAB 的 API 函数库将 C 或 FORTRAN 子程序编译成动态链接函数(库)，即 MEX 文件，以便在 MATLAB 环境中直接调用或链接这些子程序，达到提高计算效率的目的。

MEX 是一种动态链接的子程序，如同 MATLAB 的内置函数一样，能被 MATLAB 的解释器根据命令自动地装入和执行。MEX 文件具有以下几个方面的应用：

- 对于某些已有的 C 或 FORTRAN 子程序，可以通过 MEX 方式在 MATLAB 环境中直接调用，而不必重新编写相应的 M 文件。
- 对于影响 MATLAB 执行速度的 for 等循环体，可以编写相应的 C 或 FORTRAN 子程序完成相同功能，并编译成 MEX，提高运行速度。
- 对于 A/D 或是 D/A 卡，或其他 PC 硬件，可以直接用 MEX 文件进行访问，扩大 MATLAB 的功能。更重要的是，在这种对实时性要求较高的场合，使用 MEX 文件能够满足各种苛刻的要求。
- 利用 MEX 文件，还可以使用一些软件，如 Windows 的用户界面资源等。

当然，并不是所有的应用都适于用 MEX 文件来实现。MATLAB 的主要优越性在于可以节省编程时间。所以在一般情况下，还是以 MATLAB 编程为主。本书中将简单介绍如何开发 C 语言 MEX 文件。

8.3.1 MEX 文件的使用

MEX 文件是由 C 或 FORTRAN 源程序经过编译生成的 MATLAB 动态链接子程序，它的作用十分类似于 MATLAB 的内部函数。在 Windows 95 系统下，MEX 文件是 32 位 DLL 格式的。

使用 DLL 格式的优点是可以使用户直接访问 Windows 资源环境的各种功能。可以用 DLL 格式的 MEX 文件生成基于 Windows 95 的用户图形界面，也可以利用 Windows 的动态数据交换(DDE)能力，与其他的 Windows 应用程序交换必要的数据。

MATLAB 的 MEX 文件的扩展名可以是 MEX 或是 DLL。在 MATLAB 中，有一个名字为 ode23.mex 的文件，这个子程序的运行速度比 ode23.m 快。由于 MATLAB 调用函数时的顺序是在同一目录下先执行 MEX 文件，其次是 DLL 文件，最后是 M 文件，因此，当向 MATLAB 系统发出 ode23 函数调用命令时，一般是使用 ode23.mex。

Windows 的 SDK 被要求用来创建 DLL 格式的 MEX，而不论用户是否要访问 Windows 的用户界面的功能。在 SDK 中，可以从用户的 MEX 文件中访问 Windows 的应用函数。

所以，可以在 MEX 程序中加入一些 Windows 的资源，如对话框、标题栏、动态数据交换功能等。如果在 MEX 文件中使用 Windows 的功能，必须在 MEX 的源程序中加入：

```
#include <windows.h>
```

下面介绍如何配置系统，从而可以顺利地编译用户自己的 MEX 文件。

对于 5.3 版以前的 MATLAB，需要机器里已经安装了 VC、BC 或 Watcom C 中的一种。如果在安装 MATLAB 时已经设置过编译器，那么就可以使用 mex 命令来编译 C 语言的程序了。否则，在 MATLAB 里输入 mex -setup ，就会出现一个 DOS 方式窗口，然后根据提示一步步进行设置。但是需要注意的是，在设置编译器路径时，只能使用路径名称的 8.3 字符形式。比如 VC5 装在路径 C:\Program Files\DevStudio 下，则在设置路径时就要写成：C:\PROGRA~1\DevStudio。

对于 5.3 版，MATLAB 提供了一个编译器 lcc，可以直接使用。

这样设置完之后，MEX 就可以编译了。为了测试路径设置正确与否，可以把下面的程序存为 hello.c。

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[ ], int nrhs, const mxArray
*prhs[ ])
{
mexPrintf("Hello,This is my first MEX program!\n");
}
```

假设把 hello.c 放在了 C:\TEST\下，在 MATLAB 里用 CD C:\TEST\ 将当前目录改为 C:\TEST\(注意，仅将 C:\TEST\加入搜索路径是没有用的)。现在输入 mex hello.c。

如果一切顺利，编译应该在出现编译器提示信息后正常退出。例如，对于 Microsoft Visual C++ 5.0 的使用者，会出现如下所示的提示信息：

```
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 11.00.7022 for
80x86
Copyright (C) Microsoft Corp 1984-1997. All rights reserved.
```

然后输入 hello，程序会在屏幕上打出一行：

```
Hello,This is my first MEX program!
```

8.3.2 C 语言的 MEX 文件

1. 目录结构

这一小节将讨论如何生成基于 C 语言的 MEX 文件，MEX 文件是通过将用户的 C 源程序与 MATLAB 提供所需的 API 库链接在一起的方式得到的。

有关生成 MEX 文件的辅助文件和 MATLAB 链接库文件组织在 MATLAB 根目录下的子目录\EXTERN 下。它主要有 3 个子目录，其中：

\INCLUDE 目录中放有相关的源文件：

mex.h MEX 文件的函数原型头文件

matrix.h 访问矩阵变量的函数原型头文件

mxArray.h 访问数组变量的函数原型头文件

\LIB 目录下存放有 API 函数库文件，主要用于 MEX 文件的链接。

\SRC 目录存放有一些例子的源程序文件。

注意： 在 MEX 文件的源程序中必须包含头文件 mex.h；mxArray.h 仅能用于 5.x 版本。

2. MEX源文件结构与工作原理

MEX 文件是由 MEX 源代码文件经过适当的编译器编译和链接器链接而生成的动态链接子程序。MEX 源代码文件由两部分组成，第一部分称为入口子程序(Gateway Routine)，第二部分称为计算功能子程序(Computational Routine)。它们分工明确，入口子程序的作用是在 MATLAB 系统与被调用的外部子程序之间建立通信联系，可以看作是其通信协议。它定义被 MATLAB 调用的外部子程序的入口地址，定义 MATLAB 系统向子程序传递的子程序参数，还定义子程序向 MATLAB 系统返回的结果参数，以及调用计算功能程序等。而计算功能子程序就是要链接的外部子程序，它用于完成一些特定的计算，它由入口子程序调用。

MEX 源码文件的两部分既可以分开，又可以组合在一起。但无论怎样，入口子程序的函数名必须是 mexFunction，其构成形式为：

```
void mexFunction(
int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[])
{
/* 必要的 C 代码*/
}
```

下面对 mexFunction 函数中的参数作具体说明：

nrhs 整数型变量，记录调用 MEX 文件时的 mxArray(或 Matrix)类型输入参数的个数，即 MATLAB 函数调用的右端变量个数

prhs 指针数组变量，其元素是指向 mxArray(或 Mairix)类型的输入参数变量的指针；

nlhs 整数型变量，记录调用 MEX 文件时的 mxArray(或 Matrix)类型的输出参数的个数，即 MATLAB 函数调用的左端变量个数；

plhs 指针数组变量，其元素是指向 mxArray(或 Matrix)类型的输出参数变量的指针

这些参数是用来传递 MATLAB 启动 MEX 文件的参数，按 MATLAB 的语法规则，函数(包括 MEX 文件)调用的一般形式是：

```
[a,b,c,…]=fun(d,e,f,…)
```

这里 fun 是 MEX 文件名或 M 文件名，三连点表示参数个数可以是任意的，变量 a、b、

c 等称为输出变量(左变量)，而 d、e、f 等称为输入变量(右变量)。

例如，如果要调用一个名为 funmex 的 MEX 文件，则在 MATLAB 环境中的命令形式为：

```
x = fun(y,z);
```

随后，MATLAB 从 fun 的 mexFunction 处开始执行，且参数传递如下所示：

```
nlhs = 1
nrhs = 2
plhs = 指针→'NULL'
prhs = (指针→y  指针→z)
```

在执行子程序调用之前，由于输出变量 x 对象还没有创建，所以参数 plhs 的各指针指向的地址暂未定义。在入口子程序中，先根据需要创建 mxArray 类型的输出变量，其地址分别保留在 prhs[0]和 prhs[1]中。再通过两个指针，就可以在计算功能子程序中使用这些变量作计算。

【例 6】用 MEX 文件计算一个数乘以 2 的值

下面是完成计算的 C 代码：

```
#include <math.h>
void timestwo(double y[], double x[])
{
y[0] = 2.0*x[0];
return;
}
```

对应于上述函数的 MEX 源代码文件如下定义，入口子程序与计算功能子程序合在一起，代码文件名为 timestwo.c。

```
#include "mex.h"
void timestwo(double y[], double x[])
{
y[0] = 2.0*x[0];
}
void mexFunction( int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[] )
{
    double *x,*y;
    int mrows,ncols;

    /* 检查变量个数 */
    if(nrhs!=1)
     {
    mexErrMsgTxt("One input required.");
    }
```

```
    else if(nlhs>1)
    {
    mexErrMsgTxt("Too many output arguments");
    }

    /* 输入变量必须是一个双精度的实向量*/
    mrows = mxGetM(prhs[0]);
ncols = mxGetN(prhs[0]);
if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
!(mrows==1 && ncols==1) )
 {
mexErrMsgTxt("Input must be a noncomplex scalar double.");
}

/* 创建矩阵，作为输出向量*/
plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

/* 赋输入/输出变量指针 */
x = mxGetPr(prhs[0]);
y = mxGetPr(plhs[0]);

/* 调用功能计算子程序 */
timestwo(y,x);
}
```

说明：

从这个例子可以看出，入口子程序起到链接 C 子程序与 MATLAB 系统的作用：同时入口子程序完成大部分的辅助性工作，特别是判断变量类型。例如，在本入口子程序中，有好几条语句用于检测输入/输出变量的个数，以及变量的数据类型。这一步骤是必要的，因为 MATLAB 不像 C 语言那样，提供变量类型的说明。进行必要的类型检查可以避免由于指针与数据不匹配而导致的错误。

利用上一节所讲的方法编译这个文件，然后在 MATLAB 下就可以调用这个 MEX 文件，例如：

```
x = 2;
y = timestwo(x)
y =
   4
```

MATLAB 5.x的API不但支持MATLAB系统的最基本数据结构——矩阵，而且支持更广泛的数据结构和数据类型。MATLAB支持的任何数据结构和数据类型都可以用MEX子程序来进行传递、处理。下面进行详细说明。

- 字符串变量

【例7】传递并处理字符串变量的程序示例

下面是计算功能子程序的MEX源代码文件，它的功能是将输入的字符串按照反序输出：

```
#include "mex.h"
void revord(char *input_buf, int buflen, char *output_buf)
{
int i;
/* 将输入的字符串反序 */
for(i=0;i<buflen-1;i++)
*(output_buf+i) = *(input_buf+buflen-i-2);
}
```

在MEX文件结束时，MATLAB能够自行回收不再使用的内存空间。如果用户不想这样，可以使用mexMakeMemoryPersistent来自行回收。

以下是入口子程序的源代码文件：

```
void mexFunction( int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[])
{
char *input_buf, *output_buf;
int buflen,status;

/* 检查输入变量的个数 */
if(nrhs!=1)
mexErrMsgTxt("One input required.");
else if(nlhs > 1)
mexErrMsgTxt("Too many output arguments.");

/* 输入变量必须为字符串*/
if ( mxIsChar(prhs[0]) != 1)
mexErrMsgTxt("Input must be a string.");

/* 输入必须是一个行向量 */
if (mxGetM(prhs[0])!=1)
mexErrMsgTxt("Input must be a row vector.");

/* 取得该输入字符串的长度 */
buflen = (mxGetM(prhs[0]) * mxGetN(prhs[0])) + 1;

/* 为输入/输出字符串分配内存 */
```

```
input_buf=mxCalloc(buflen, sizeof(char));
output_buf=mxCalloc(buflen, sizeof(char));

/* 将输入数据从prhs[0]开始，复制到一个C字符串input_buf中。
* 如果字符串包含多行，则一次转换一列，将结果写入一个长的字符串中 */
status = mxGetString(prhs[0], input_buf, buflen);
if(status != 0)
mexWarnMsgTxt("Not enough space. String is truncated.");

/* 调用C子程序 */
revord(input_buf, buflen, output_buf);

/* 将C语言的输出转换为MATLAB MEX格式的输出 */
plhs[0] = mxCreateString(output_buf);
return;
}
```

说明：

- 在入口子程序中，分配的 char 字符型内存 buf 用来将输入的字符串传递给计算功能子程序 revord，函数 revord 返回一个字符型指针，然后 mxCreateString 函数生成返回的字符串 rev_String 的副本，并将输出变量 mxArray 指针指向该副本值。
- 本例中，API 函数 mxCalloc 代替了标准 C 中的 calloc 函数来进行动态内存分配。mxCalloc 使用 MATLAB 的内存管理器来动态分配内存，并且将它们初始化为 0 值。在任何需要使用 calloc 来分配内存的地方都应当改用 mxCalloc。同样，在 C 语言使用 malloc 时应当使用 mxMalloc；在使用 realloc 时应当使用 mxRealloc。
- 结构型变量和单元数组

结构(Structure)数据、单元(cell)数组类型是MATLAB 5.x定义的一种新的数据类型。像其他的MATLAB数据类型一样，结构型和单元数组数据可以在C语言MEX文件中传递。

向MEX子程序输入结构型数据时，如同传递其他类型的数据，结构的每个域(field)本身是mxArray型数据。所以API函数mxGetField返回一个mxArray类型的指针，再将这个指针当作通常的数组类型指针处理。但是，如果希望将结构域值传递到C子程序中，则必须调用API函数，例如mxGetString创建一个字符型指针，并将其指向域值。

MATLAB 5.x的API函数mxCreateStructArray为结构型变量分配内存，但是它不能写入单个域值，而必须用API函数mxSetField。这与字符串内存创建函数mxCreateSting不同，mxCreateString函数不仅为字符串分配内存，而且还将mxArray指针指向给定值。一般说来，对于较复杂的MATLAB数据结构，如单元数组和结构，MATLAB 5.x的API函数不是在同一步里完成内存分配和写入变量之值的任务的。

【例8】结构型数据和单元数组的传递和处理(phonebook.c)

```
#include "mex.h"
#include "string.h"
```

```
#define MAXCHARS 80 /* 每一个域中字符串的最大个数 */
/* 入口子程序 */
void mexFunction( int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[] )
const char **fnames; /* Pointers to field names */
const int *dims;
mxArray *tmp, *fout;
char *pdata;
int ifield, jstruct, *classIDflags;
int NStructElems, nfields, ndim;
/* 输入/输出变量的检查. */
{…}
/* 取得输入变量*/
nfields = mxGetNumberOfFields(prhs[0]);
NStructElems = mxGetNumberOfElements(prhs[0]);
/* 为输入变量的类名分配储存空间 */
classIDflags = mxCalloc(nfields, sizeof(int));
/* 检查空的域、适当的数据类型、取得每个域的数据类型 */
for(ifield=0; ifield<nfields; ifield++)
{for(jstruct = 0; jstruct < NStructElems; jstruct++)
{tmp = mxGetFieldByNumber(prhs[0], jstruct, ifield);
if(tmp == NULL)
 {
mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :",
jstruct+1);
mexErrMsgTxt("Above field is empty!");
}
if(jstruct==0)
 {
if( !mxIsChar(tmp) && !mxIsNumeric(tmp))
 {
mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1, "STRUCT INDEX :",
jstruct+1);
mexErrMsgTxt("Above field must have either string or
numeric data.");
}
classIDflags[ifield]=mxGetClassID(tmp);
}
 else
```

```
    if (mxGetClassID(tmp) != classIDflags[ifield])
     {
    mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1,"STRUCT INDEX :",
jstruct+1);
    mexErrMsgTxt("Inconsistent data type in above field!");
    }
     else if(!mxIsChar(tmp) &&
    ((mxIsComplex(tmp) || mxGetNumberOfElements(tmp)!=1))){
    mexPrintf("%s%d\t%s%d\n", "FIELD: ", ifield+1,"STRUCT INDEX :",
jstruct+1);
    mexErrMsgTxt("Numeric data in above field must be scalar
    and noncomplex!");
    }}}}

    /* 为指针分配内存空间 */
    fnames = mxCalloc(nfields, sizeof(*fnames));
    /* Get field name pointers. */
    for (ifield=0; ifield< nfields; ifield++)
    {
    fnames[ifield] = mxGetFieldNameByNumber(prhs[0],ifield);
    }
    /* 为输出创建一个1×1的矩阵. */
    plhs[0] = mxCreateStructMatrix(1, 1, nfields, fnames);
    mxFree(fnames);
    ndim = mxGetNumberOfDimensions(prhs[0]);
    dims = mxGetDimensions(prhs[0]);
    for(ifield=0; ifield<nfields; ifield++)
     {
    /* 建立单元/数值型数组 */
    if(classIDflags[ifield] == mxCHAR_CLASS)
     {
    fout = mxCreateCellArray(ndim, dims);
    }
    else
     {
    fout = mxCreateNumericArray(ndim, dims,classIDflags[ifield],
mxREAL);
    pdata = mxGetData(fout);
    }
```

```
    /*从输入的结构数组中取得数据*/
    for (jstruct=0; jstruct<NStructElems; jstruct++)
    {
    tmp = mxGetFieldByNumber(prhs[0],jstruct,ifield);
    if( mxIsChar(tmp))
    {
    mxSetCell(fout, jstruct, mxDuplicateArray(tmp));.3 Creating C
Language MEX-Files
    }
    else
    {
    size_t sizebuf;
    sizebuf = mxGetElementSize(tmp);
    memcpy(pdata, mxGetData(tmp), sizebuf);
    pdata += sizebuf;
    }
    }
    /*为输出的结构数组创建相应的域 */
    mxSetFieldByNumber(plhs[0], 0, ifield, fout);
    }
    mxFree(classIDflags);
    return;
```

为了测试这个程序，在 MATLAB 窗口中输入下面的结构体：

```
    friends(1).name = 'Jordan Robert';
    friends(1).phone = 3386;
    friends(2).name = 'Mary Smith';
    friends(2).phone = 3912;
    friends(3).name = 'Stacy Flora';
    friends(3).phone = 3238;
    friends(4).name = 'Harry Alpert';
    friends(4).phone = 3077;
```

输出结果为：

```
    phonebook(friends)
    ans =
      name: {1x4 cell }
      phone: [3386 3912 3238 3077]
```

说明：

- 这个程序以一个 m×n 的结构型数据矩阵作为输入参数，返回一个新的 1×1 的结构型矩阵，包括：

字符串输入产生的一个 m×n 的单元数组；

数值型变量产生的一个 m×n 的向量，使用与输入变量相同的数据类型。

- 对单元数组进行访问和管理，分别使用 API 函数 mxGetCell 和 mxSetCell；而函数 mxCreateCellArray 创建数组对象。
- 单元数组本身是 mxArray 类型的，因此，可以使用前面介绍过的对 mxArray 操作的任何函数来处理单元数组，如 mxGetData。
- 复数型变量

MATLAB中的复变量被分为实部和虚部。MATLAB提供了2个API函数mxGetPr和mxGetPi来对数据进行操作，它们分别返回指向数据实部和虚部的指针。

下面通过一个例子，说明MATLAB对复数变量的操作。

【例9】计算复数向量的卷积(convec.c)

```
#include "mex.h"
/* 计算子程序*/
void convec( double *xr, double *xi, int nx,double *yr,
   double *yi, int ny,double *zr, double *zi)
{
int i,j;
zr[0]=0.0;
zi[0]=0.0;
/* 进行卷积计算 */
for(i=0; i<nx; i++)
{
for(j=0; j<ny; j++)
{
*(zr+i+j) = *(zr+i+j) + *(xr+i) * *(yr+j) - *(xi+i)
* *(yi+j);
*(zi+i+j) = *(zi+i+j) + *(xr+i) * *(yi+j) + *(xi+i)
* *(yr+j);
}
}
}
```

下面是对计算子程序进行调用的入口子程序：

```
void mexFunction( int nlhs, mxArray *plhs[],int nrhs, const mxArray
*prhs[] )
{
double *xr, *xi, *yr, *yi, *zr, *zi;
int rows, cols, nx, ny;
/* 检查输入变量的个数*/
if(nrhs != 2)
```

```
mexErrMsgTxt("Two inputs required.");
if(nlhs > 1)
mexErrMsgTxt("Too many output arguments.");
/* 判断两个输入变量是否均为行向量 */
if( mxGetM(prhs[0]) != 1 || mxGetM(prhs[1]) != 1 )
mexErrMsgTxt("Both inputs must be row vectors.");
rows = 1;
/* 判断两个变量是否均为复向量 */
if( !mxIsComplex(prhs[0]) || !mxIsComplex(prhs[1]) )
mexErrMsgTxt("Inputs must be complex.\n");
/* 取得每个向量的长度 */
nx = mxGetN(prhs[0]);
ny = mxGetN(prhs[1]);
/* 取得指向输入向量实部和虚部的指针 */
xr = mxGetPr(prhs[0]);
xi = mxGetPi(prhs[0]);
yr = mxGetPr(prhs[1]);
yi = mxGetPi(prhs[1]);
/* 建立一个新的数组，并将输出数据的指针指向它 */
cols = nx + ny - 1;
plhs[0] = mxCreateDoubleMatrix(rows, cols, mxCOMPLEX);
zr = mxGetPr(plhs[0]);
zi = mxGetPi(plhs[0]);
/* Call the C subroutine. */
convec(xr, xi, nx, yr, yi, ny, zr, zi);
return;
}
```

为了测试这个程序，在MATLAB提示符下输入：

```
x = [3.000-1.000i, 4.000+2.000i, 7.000-3.000i];
y = [8.000-6.000i, 12.000+16.000i, 40.000-42.000i];
```

然后调用这个MEX-文件进行计算：

```
z = convec(x,y)
z =
   1.0e+02 *
   Columns 1 through 4
   0.1800-0.2600i 0.9600+0.2800i 1.3200-1.4400i 3.7600-0.1200i
   Column 5
1.5400 - 4.1400i
```

- 多维数值数组变量

多维数值数组也是 MATLAB 5.x 新提供的一种数据类型。为了和 C 语言 MEX 文件进行多维数组变量的交换，可以使用 API 函数 mxGetData 和 mxGetImagData，返回指向变量实部和虚部的指针。相反的，可以用 API 函数 mxCreateNumericArray 在 MATLAB 中创建多维数值数组对象，并传递到 MATLAB 系统中去。

【例 10】多维数值数组变量的传递和操作示例(findnz.c)

该程序以一个 N 维数值数组为输入变量，输出其中非零元素的位置。

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[])
{
/* 变量的定义 */
int elements, j, number_of_dims, cmplx;
int nnz=0, count=0;
double *pr, *pi, *pind;
const int *dim_array;
/*检查输入、输出变量的数目 */
if (nrhs != 1)
{ .
mexErrMsgTxt("One input argument required.");
}
if (nlhs > 1)
{
mexErrMsgTxt("Too many output arguments.");
}
/* 检查输入变量的类型 */
if (!(mxIsDouble(prhs[0])))
{
mexErrMsgTxt("Input array must be of type double.");
}
/* 得到输入变量的个数 */
elements = mxGetNumberOfElements(prhs[0]);
/* 取得输入变量的值 */
pr = (double *)mxGetPr(prhs[0]);
pi = (double *)mxGetPi(prhs[0]);
cmplx = ((pi==NULL) ? 0 : 1);
/* 计算矩阵中非零元素的数目，以便为输出变量分配内存 */
for(j=0;j<elements;j++)
{
```

```
if(IsNonZero(pr[j]) || (cmplx && IsNonZero(pi[j])))
{
nnz++;
}
}
/* 取得输入变量的维数，为返回值分配内存 */
number_of_dims = mxGetNumberOfDimensions(prhs[0]);
plhs[0] = mxCreateDoubleMatrix(nnz, number_of_dims, mxREAL);
pind = mxGetPr(plhs[0]);
/* 取得输入变量的维数. */
dim_array = mxGetDimensions(prhs[0]);
/* 通过循环，统计零的数目. */
for(j=0;j<elements;j++)
{
if(IsNonZero(pr[j]) || (cmplx && IsNonZero(pi[j])))
{
int temp=j;
int k;
for (k=0;k<number_of_dims;k++)
{
pind[nnz*k+count]=((temp % (dim_array[k])) +1);
temp/=dim_array[k];
}
count++;
}
}
}
```

为了测试这个函数，可以在MATLAB工作窗口中输入：

```
matrix = [ 3 0 9 0; 0 8 2 4; 0 9 2 4; 3 0 9 3; 9 9 2 0];
nz=findnz(matrix)
```

用户可以自己看到运行结果。

- 调用MATLAB函数和自定义函数

在MEX源代码中，可以调用MATLAB函数和MEX文件，其调用是由API函数mxCallMATLAB实现的，如下例所示。

【例11】MEX文件中sin函数的调用(sincall.c)

```
#include "mex.h"
#define MAX 1000
/* 等分数值区间 */
void fill( double *pr, int *pm, int *pn, int max )
```

```
{
int i;
/* 将区间分为MAX份 */
*pm = max/2;
*pn = 1;
for (i=0; i < (*pm); i++)
pr[i]=i*(4*3.14159/max);
}
/* 入口子程序 */
void mexFunction( int nlhs, mxArray *plhs[],
int nrhs, const mxArray *prhs[] )
{
int m, n, max=MAX;
mxArray *rhs[1], *lhs[1];
rhs[0] = mxCreateDoubleMatrix(max, 1, mxREAL);
/* 传递指针 */
fill(mxGetPr(rhs[0]), &m, &n, MAX);
mxSetM(rhs[0], m);
mxSetN(rhs[0], n);
/* 取得正弦函数值，然后绘图 */
mexCallMATLAB(1, lhs, 1, rhs, "sin");
mexCallMATLAB(0, NULL, 1, lhs, "plot");
/* 清除已分配的内存 */
mxDestroyArray(rhs[0]);
mxDestroyArray(lhs[0]);
return;
}
```

在命令窗口中输入sincall，就可以看到结果(图略)。

说明：

- 本程序的作用是创建一个 mxArray 对象，并将它相应的指针传递给 fill 子函数，填充数据，最后调用 mexCallMATLAB，计算 sin 函数的函数值并画图。
- 使用 mexCallMATLAB，可能会创建一个 mxUNKNOWN_CLASS 的对象。例如，如果创建一个 M 文件，返回两个变量，但是只给其中的一个赋值，然后再用 mexCallMATLAB 调用这个函数，那个未赋值的变量类型就会变为 mxUNKNOWN_CLASS。

从以上的例子可以看出，计算功能子程序按照 C 语言规则编写，处理 C 语言的各种

数据结构，完成各项计算功能。而入口子程序的功能是作为 MATLAB 与 C 语言子程序之间的接口，处理 MATLAB 向 MEX 子程序传递的变量。用 API 函数从这些输入变量中取出 C 语言子程序，即计算功能子程序所需要的各种数据及其数据结构；负责调用计算功能子程序完成必要的计算任务；按照 MATLAB 支持的个数创建输出变量并将计算功能子程序的计算结果通过所创建的数据对象传递到 MATLAB 系统中去。

第 9 章
图形用户界面(GUI)编程

本章要点:

现代的主流应用程序已经从命令行的交互方式转变为以图形界面为主的交互方式，这主要是由于它给用户带来了操作和控制的方便与灵活性。MATLAB 能够实现以比较简单的方式实现一系列的图形界面功能。通过对控件属性的设置和 Callback 的编写，就能够满足大多数用户的需求。

本章具体包括以下内容:

- 控件对象及其属性
- 菜单对象及其属性
- GUI 的设计方法
- 单一选择的单选按钮组设计
- 中断 Callback 的操作
- 鼠标的操作
- GUI M 文件的调试
- GUI 程序设计的其他问题

传统的用户界面定义为用户与计算机之间交互通信联系的平台。但在最近几年内，这种概念发生了巨大的变化，出现了多种形式的人机交互方式，从命令行的交互方式转变至以图形界面为主的交互形式。现在，图形界面已在人机交互方式中占主导地位，这主要是由于它给用户带来了操作和控制的方便与灵活性。

图形用户界面(GUI)是包含图形对象(如窗口、图标、菜单和文本)的用户界面。以某种方式选择或激活这些对象，通常引起动作或发生变化。最常见的激活方法是用鼠标或其他单击设备去控制屏幕上的鼠标指针的运动。按下鼠标按键，标志着对象的选择或其他动作。

MATLAB 也提供了在 MATLAB 应用程序中加入 GUI 的功能。本章的目的是阐述 MATLAB 的图形用户界面特性，并说明如何在应用程序中加入一些必要的图形用户界面单元，以方便其他用户使用。

本章中阐述 MATLAB 的 GUI 特性是 MATLAB 图形句柄系统的子系统。在本书的前面已经较详细地介绍了 MATLAB 图形句柄系统。为方便起见，再回顾一下与图形对象相关的几个概念：

- 图形对象
- 对象句柄
- 对象属性及对象属性值

MATLAB 的 GUI 的基本图形对象分为两类——用户界面控件对象和用户界面菜单对象，简称为控件和菜单对象。设计一个高效的用户界面包含以下的工作：选择恰当的图形对象，并将它们有逻辑地组织起来，使得用户界面容易操作和使用。

本章将说明图形句柄uicontrol 和uimenu对象的使用，把图形界面加到MATLAB的函数和M文件。uicontrol对象能建立如按钮、滚动条、弹出式菜单以及文本框等对象，uimenu对象能在图形窗口中产生下拉式菜单和子菜单。

9.1 控件对象及属性

9.1.1 控件对象类型

控件对象是这样一类图形界面对象：用户用鼠标在控件对象上进行操作，单击鼠标时，将会使应用程序作出响应并执行某些预定的功能子程序(Callback)。控件的操作结果是可见的，有时可以改变应用程序的初始状态。MATLAB 支持 10 种控件对象，它们的添加方式有 2 种：基于命令行的和基于 GUI 界面的，这将在下面的章节中介绍。下面详细介绍它们自身，包括每种控件的目的。

1. 坐标轴(Axes)

坐标轴对象如图9.1 所示，它的特性前面已经有过详细地讨论，这里就不再详述了。

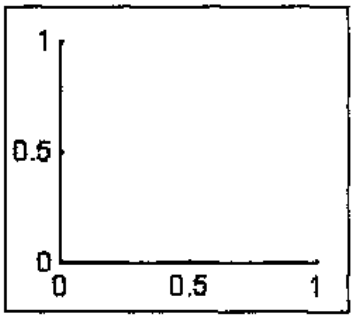


图 9.1　坐标轴对象

2. 静态文本框(Text)

静态文本框是仅仅显示一个文本字符串的uicontrol，该字符串是由String属性所确定的。静态文本框的Style属性值是text。静态文本框一般用于显示标志、用户信息及当前值。

静态文本框之所以称为“静态”，是因为用户不能动态地修改所显示的文本。文本只能通过改变String属性来更改。

3. 可编辑文本框(Edit)

可编辑文本框，像静态文本框一样，在屏幕上显示字符。但与静态文本框不同，可编辑文本框允许用户动态地编辑或重新安排文本串，就像使用文本编辑器或文字处理器一样。在String属性中有该信息。可编辑文本框uicontrol的Style属性值是edit。可编辑文本框一般用在让用户输入文本串或特定值。

可编辑文本框可包含一行或多行文本。单行可编辑文本框只接受一行输入，而多行可编辑文本框可接受两行以上的输入。单行可编辑文本框的输入以Enter键结尾。在X Windows和MS-Windows系统中，多行文本输入以Control-Return键结尾，而在Macintosh中用Command-Return键。

4. 弹出式菜单(PopupMenu)

弹出式菜单一般用于向用户提出互斥的一系列选项清单，让用户可以选择。弹出式菜单不受菜单条的限制，可位于图形窗口内的任何位置。弹出式菜单的Style属性值是popupmenu。

当关闭时，弹出式菜单以矩形或按钮的形式出现，按钮上含有当前选择的标志，在标志右侧有一个向下的箭头或凸起的小方块来表明uicontrol对象是一个弹出式菜单。当指针处在弹出式uicontrol之上并按下鼠标左键时，出现其他选项。移动指针到不同的选项，松开鼠标左键就关闭弹出式菜单，显示新的选项。MS-Windows 和某些X Windows系统平台允许用户单击弹出式菜单，打开它，而后单击另一个选项来进行选择。

5. 滑标(Slider)

滑标，或称滚动条，包括3个独立的部分，分别是滚动槽或长方条区域，代表有效对象值范围；滚动槽内的指示器代表滑标当前值；以及槽的两端的箭头。滑标uicontrol的Style属性值是Slider。

滑标一般用于从几个值域范围中选定一个。滑标值有3种方式设定。

方法1：鼠标指针指向指示器，移动指示器。拖动鼠标时要按住鼠标按键，当指示器位于期望位置后再松开鼠标按键。

方法2：当指针处于槽中但在指示器的一侧时，单击鼠标按键，指示器按该侧方向移动距离约等于整个值域范围的10% 。

方法3：在滑标不论哪端单击箭头，指示器沿着箭头的方向移动大约为滑标范围的1%。

滑标通常与所用文本uicontrol对象一起显示标志、当前滑标值及值域范围。

6. 框架(Frame)

框架uicontrol对象仅是带色彩的矩形区域。框架提供了视觉的分隔性。这点，框架与uimenu的Sepatator属性相似。框架一般用于组成单选按钮或其他uicontrol对象。在其他对象放入框架之前应事先定义框架，否则框架可能覆盖控制框使它们不可见。

7. 命令按钮(PushButton)

命令按钮，又称按钮键或按钮，是小的长方形屏幕对象，常常在对象本身标有文本。将鼠标指针移动至对象来选择命令按钮uicontrol，单击按钮，执行由回调字符串所定义的动作。命令按钮的Style属性值是pushbutton。

命令按钮一般用于执行一个动作而不是改变状态或设定属性，它与Windows系统的命令按钮的作用相同。

8. 单选按钮(RadioButton)

单选按钮，又称选择按钮或切换按钮，它由一个标志并和标志文本左端的一个小圆圈或小菱形所形成。当选择时，圆圈或菱形被填充，且Value属性值设为1；若未被选择，指示符被清除，Value属性值设为0。单选按钮的Style的属性值是radiobutton。

单选按钮一般用于在一组互斥的选项中选择一项。为了确保互斥性，各单选按钮uicontrol的回调字符串必须不选组中其他项，将它们各项的Value属性值设为0。然而，这只是一个约定，如果需要，单选按钮可与复选框交换使用。

9. 复选框(CheckBox)

复选框，又称切换按钮，它由具有标志并在标志左边的一个小方框所组成。激活时，uicontrol在检查和清除状态之间切换。在检查状态时，根据平台的不同，方框被填充或在框内含×，Value属性值设为1。若为清除状态，则方框变空，Value属性值设为0。

复选框一般用于表明选项的状态或属性。通常复选框是独立的对象，如果需要，复选框可与单选按钮交换使用。

10. 列表框(ListBox)

这是 MATLAB 5.x 新增加的一个控件对象。列表框列出字符串表，用户可以在这个列表中选取单个列表项或多个列表项。对应于选择，MATLAB 执行相应的功能或操作。

9.1.2　控件对象的创建

有 2 种方式创建控件对象，下面分别介绍。

1.　基于命令行的方式

创建控件对象的基本方法是使用 MATLAB 的函数 uicontrol。该函数的调用形式为

```
h=uicontrol(hfig,'属性名','属性值',…)
```

设计程序时，可根据需要向函数 uicontrol 提供必要的属性名和属性值参数对。MATLAB 系统不区别属性字符串中的大小写字母。例如，Style 和 style 都表示是属性 Style 的名字。

由于控件对象都是图形窗口对象的子对象，所以函数 uicontrol 调用中，第一个参数 hfig 应该是某个图形窗口句柄值。所创建的控件对象将是该图形窗口的子对象，并出现在该图形窗口中。如果省略这个句柄值参数，则生成的图形对象将是当前的图形窗口的子对象。如果此时无图形窗口存在，MATLAB 系统将自动创建一个图形窗口，并在其中创建相应的控件对象。

h 是所创建控件对象的句柄值，MATLAB 通过这个句柄值来管理该控件对象，如同 MATLAB 任何类型的图形对象一样。在 MATLAB 图形句柄系统中，控件对象也是一种图形对象，因此对于图形对象的操作方法都能用于控件对象。例如，可以用 get 函数取得控件对象的某些属性的当前值，用 set 函数可以改变和重新设置控件对象的某些当前的属性值。

【例 1】控件对象的创建。

```
[x,y] = meshgrid([-2:.4:2]);
Z = x.*exp(-x.^2-y.^2);
fh = figure('Position',[350 275 400 300],'Color','w');
ah = axes('Color',[.8 .8 .8],'XTick',[-2 -1 0 1 2],'YTick',[-2 -
1 0 1 2]);
sh = surface('Xdata',x,'YData',y,'ZData',Z,...
'FaceColor',get(ah,'Color')+.1,...
'EdgeColor','k','Marker','o',...
'MarkerFaceColor',[.5 1 .85]);
```

结果如图 9.2 所示。

说明：

这里只是对于 uicontrol 函数一个简单介绍，在后面的章节中，将会结合具体属性进行详细介绍。

提示：　"…" 在这里是表示换行，因而可以将每个属性的设置单独写在一行之中，这样有利于增强程序的可读性。

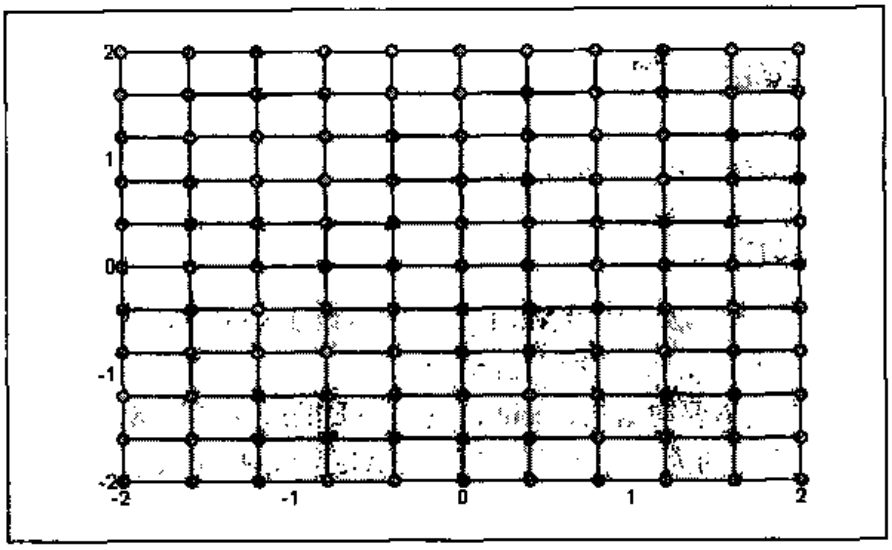


图 9.2 控件对象的创建

与控件对象相关联的功能操作是通过控件对象的 Callback 属性来定义的，具体使用方式见后面的介绍。

2. 基于GUI的方式

从 MATLAB 的命令窗口中选择 File 菜单下的 Show GUI Layout Tool 命令，或是在 MATLAB 命令窗口中输入 guide 命令，出现如图 9.3 所示的窗口。

Guide 控制板是 Guide 工具的集成环境，它使得图形对象的生成和管理变得简单、直接。例如，可以很方便地访问图形对象的属性；可以方便地控制图形对象的位置、创建任一种图形对象。

要创建一个图形对象，先单击 Add Figure 按钮，或是从列表中选择一个图形窗口。然后在图形对象创建工具栏(Guide 控制板最下面的部分)中单击想要添加的图形对象，在图形窗口中“画出”所需的位置和大小就可以了。

对于该控制板中的其他工具，将在后面介绍。

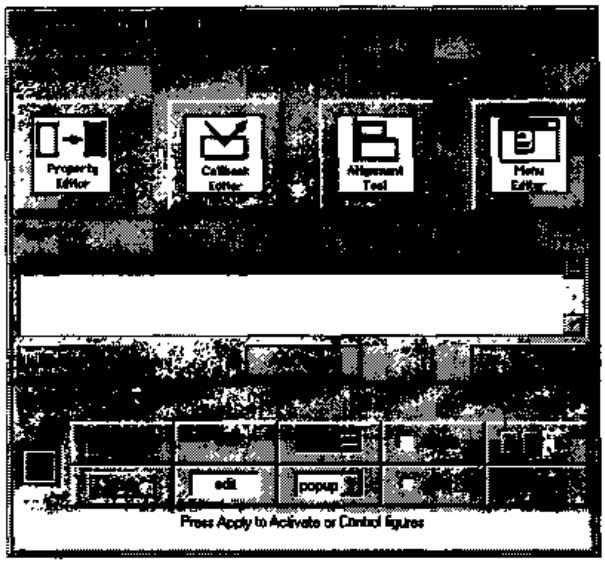


图 9.3 Guide控制板图形窗口

9.1.3　控件对象的属性

在 MATLAB 的图形句柄系统中，可使用属性对对象的外型和特性进行描述。用户可以在创建控件对象时设定这些属性的属性值，对未指定的属性值，MATLAB 将使用系统提供的默认属性值。可以用设置图形对象默认属性值的方法设置控件对象的默认属性。MATLAB 的控件对象使用相同的属性类型，但是这些属性对于不同类型的控件对象，其含义不尽相同。

控件对象的属性分为两大类，第一类是所有的控件对象都具有的公共属性，第二类是把控件对象作为图形对象所具有的公共属性。下面分别予以介绍。

1. 公共属性

- Children 属性

Children 属性的取值为空矩阵，因为控件对象自己没有子对象。

- Parent 属性

Parent 属性的取值是某个图形窗口对象的句柄值，因为控件对象总是图形窗口的子对象，该句柄值标明了控件对象所在的图形窗口。如果用 set 函数将 Parent 属性设置为另一个图形窗口的句柄值，则相当于将控件对象移到另一个图形窗口中。

- Tag 属性

Tag 属性的取值是字符串。它定义了该控件的一个标识值。定义了 Tag 属性后，在任何程序中都可以通过这个标识值找出该控件对象。

- Type 属性

Type 属性的取值总是 uicontrol，这个属性值表明图形对象的类型。对于控件对象，其类型就是 uicontrol，用户不能改写这个属性。

- UserData 属性

UserData 属性的取值是一个矩阵，默认值为空矩阵。用户可以在这个属性中保存与该控件对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。可以用 set 和 get 函数访问该属性。

- Visible 属性

Visible 属性的取值是 on(默认值)或 off。如果该属性值为 off，那么控件对象在图形窗口中是不可见的。但是该控件对象仍存在，可以用 set 和 get 函数访问它的各个属性。

2. 基本控制属性

- BackgroundColor 属性

BackgroundColor 属性的取值是某些颜色的预定义字符或颜色 RGB 数值。它定义控件对象区域的背景色。它的默认值是系统定义的浅灰色。

- Callback 属性

Callback 属性的取值是字符串，可以是某个 M 文件名或一小段 MATLAB 语句。当用户激活控件对象(例如，在控件对象图标上单击鼠标左键或移动滑动槽标尺)时，应用程序

就运行 Callback 属性定义的子程序(callback 调用)，但是框架静态文本框控件对象不定义 Callback 属性。

- Enable 属性

Enable 属性的取值是 on(默认值)、inactive 和 off。当它的取值是 on 时，无论何时激活控件对象，MATLAB 都执行 Callback 属性定义的 callback 子程序。但如果鼠标是在控件对象图标区域外的 5 个像素宽的边界范围内按下时，就执行由 ButtonDownFcn 属性定义的 callback 子程序；若 Enable 属性的取值是 inactive，那么当用户在控件对象图标及 5 个像素边界的范围内单击鼠标时，MATLAB 执行 ButtonDownFcn 定义 callback 子程序；当 Enable 属性的取值是 off 时，控件对象的标题字是有阴影的，它的 callback 执行功能与 inactive 情形相同，即运行 ButtonDownFcn 定义的 callback 子程序。这样定义是有好处的，例如可以通过定义 ButtonDownFcn 的 callback 子程序来移动控制对象。

- Extent 属性

Extent 属性的取值是 4 元素向量［0,0,width,height］，记录控件对象标题字符的位置大小，其度量单位由 Units 属性定义。只能读取该属性值，不能改写。

- ForegroundColor 属性

ForegroundColor 属性的取值是某些颜色的预定义字符或颜色 RGB 数值，这个属性定义控件对象标题字符的颜色。标题字符的默认颜色是黑色。

- Max、Min 属性

Max 和 Min 属性的取值都是数值，其默认值分别是 1 和 0。这两个属性值对于不同的控件对象类型意义是不同的。以下分别予以介绍：

- 当单选按钮控件对象被选中时，它的 Value 属性值为 Max 属性定义的值；当控件对象处于未选中状态时，它的 Value 属性定义为 Min 属性定义的值。
- 当复选框控件对象被选中时，它的 Value 属性值为 Max 属性定义的值；当控件对象处于未选中状态时，它的 Value 属性定义为 Min 属性定义的值。
- 对于滑标控件对象，Max 属性值必须比 Min 属性值大，Max 定义滑动条最大值，Min 定义滑动条的最小值。
- 对于可编辑文本框控件对象，如果 Max-Min＞1，那么对应的文本框接受多行字符输入；如果 Max-Min＜1，那么文本框仅接受单行字符输入。
- 对于列表框控件对象，如果 Max-Min＞1，那么在列表框中允许多行选择，如果 Max-Min＜1，那么在列表框中只允许单项选择。

另外，框架、弹出式菜单和静态文本框控件对象不使用 Max 和 Min 属性。

- String 属性

String 属性的取值是字符串矩阵或单元数组，它定义控件的标题或选项内容，特别是对于弹出式菜单和列表框这样的多选择的控件对象，它的 String 属性可接受多种形式的输入，如字符串单元数组、字符串矩阵和用“|”分隔的字符串向量。对于 Edit Boxes 和 Static text 这样的多行文字控件对象，String 矩阵的每行、单元数组的每块及字符串中的“\n”字符定义的文字行分隔符，即从该处分行。对于 Editable Text 和 String 属性值可以由键盘输入。

- Style 属性

Style 属性的取值可以是 pushbutton(按钮，默认值)、RadioButton(单选按钮)、Check Boxes(复选框)、edit(文本框)、text(静态文本框)、slider (滚动条)、frame(框架)、listbox(列表框) 和 popupmenu(弹出式菜单)。这个属性值定义控件对象的类型，由相应的值决定。

- Units 属性

Units 属性的取值可以是 pixel(像素，默认值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)、points(磅)。除了 normalized(相对单位)以外，其他都是绝对单位。这个属性值作为解释 Extend 和 Position 属性值的度量单位。所有单位的度量都是从图形窗口的左下角处开始。在相对单位下，图形窗口的左下角对应(0,0)，而右上角对应(1.0,1.0)。

- Value 属性

Value 属性的取值可以是向量值，也可以是数值。它的含义以及解释依赖于控件对象的类型。对应于单选按钮和复选框对象，当它们处于选中状态时，Value 属性值由 Max 属性值定义，反之由 Min 属性值定义。对于滑动条对象，Value 属性值处于 Min 和 Max 属性值之间，由滑动条标尺位置对应的值定义。对于弹出式菜单对象，Value 属性值是被选项的序号，例如 1 对应第一项。所以由 Value 值，可知弹出式菜单的选项。同样，对于列表框对象，Value 属性值定义了列表框中高亮度选项的序号。其他的控件对象不使用这个属性值。

3. 修饰控制属性

- FontAngle 属性

FontAngle 属性的取值是 normal(默认值)、italic 和 oblique，这个属性值定义控件对象标题等的字体。其值为 normal 时，选用系统默认的正字体；其值为 italic 或 oblique 时，使用方头斜字体。

- FontName 属性

FontName 属性的取值是控件对象标题等使用字体的字库名，必须是系统支持的各字库。默认字库是系统的默认字库。

- FontSize 属性

FontSize 属性的取值是数值，它定义控件对象标题等字体的字号。字号单位由 FontUnits 属性值定义，默认值与系统有关。

- FontUnits 属性

FontUnits 属性的取值是 points(磅，默认值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 pixels(像素)，该属性定义字号单位。相对单位将 FontSize 属性值解释控件对象图标高度百分比，其他单位都是绝对单位。

- FontWeight 属性

FontWeight 属性的取值是 normal(默认值)、Light、demi 或 bold，它定义字符字体的粗细。

- HorizontalAlignment 属性

HorizontalAlignment 属性的取值是 left、center(默认值)或 right，它定义控件对象标题等的调整方向，即标题在控件对象图标上居左(left)、居中(center)或居右(right)。

4. 辅助属性

- ListboxTop 属性

ListboxTop 属性的取值是数量值，这个属性只用于 Listbox 控件对象，它定义了位于列表框中的最上方的字符串在 String 属性中的序号。非整数值被截断为大于该值的最小整数值。

- SliderStep 属性

SliderStep 属性的取值是二元素向量[minstep，maxstep]，这个属性只对 Sliders 控件对象有定义。maxstep 定义了在滑动槽内单击鼠标左键时，滑动槽标尺应该移动的距离相对于滑槽长度的百分比；minstep 定义了在滑动槽两端箭头上单击鼠标左键时，滑动槽标尺应该移动距离相对于滑动槽长度的百分比。它的默认值是[0.01 0.10]。

- Selected 属性

Selected 属性的取值是 on 或 off(默认值)。当 Selected 属性值是 on 时，如果 SelectionHighligh 属性值也是 on，那么 MATLAB 就显示一个对象被选中的标记。如果将 ButtonDownFcn 定义为设置这个属性为 on，那么当选中控件对象时，控件对象就会有一个被选中的方框。

- SelectionHighligh 属性

SelectionHighligh 属性的取值是 on(默认值)或 off。该属性决定控件对象被选中时，是否显示被选中的状态。这个属性必须与 Selected 属性配合使用，共同控制控件对象被选中时的状态。

5. Callback管理属性

- BusyAction 属性

BusyAction 属性的取值是 cancel 或 queue(默认值)，该属性决定 MATLAB 采取的控制中断执行控件对象的 Callback 调用的方式。在 MATLAB 环境中，如果有一个 Callback 调用在运行，那么随后的 Callback 调用总是试图中断正在运行的 Callback 调用。如果此时 Interruptible 属性值为 on，那么 MATLAB 在处理事件队列时，会中断正在运行的 Callback 调用；如果 Interruptible 的属性值为 off，那么 BusyAction 属性就决定 MATLAB 处理事件的方式。此时，如果它的值是 cancel，就从事件队列中取消企图运行第二个 Callback 调用的事件；如果它的值是 queue，就将企图运行第二个 Callback 调用的事件加入事件队列，直至当前的 Callback 调用完成为止。

- ButtonDownFcn 属性

ButtonDownFcn 的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当用户在围绕控件对象的 5 个像素宽的边界内按下鼠标左键时，程序执行该属性定义的 Callback 调用。控件一般是矩形区域。如果控件对象的 Enable 属性值是 inactive 或 off，则当在该区域以及 5 个像素宽的边界内按下鼠标左键时，就执行 ButtonDownFcn 属性定义的 Callback。如果这个 Callback 定义为表达式，那么计算结果就属于 MATLAB 工作区。

- CreateFcn 属性

CreateFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句，即

当 MATLAB 生成控件对象时，MATLAB 事先应该执行的程序段。这个属性只能按设置默认值的方式定义，例如：

```
Set(0, 'DefaultUicontrolCreateFcn', 'set(gcf, "IntegerHandle ",
"off ")')
```

从根对象层设置控件对象的默认值。对于已经存在的控件对象，改变该属性值，对该对象无任何影响。MATLAB 在生成控件对象并完成所有默认属性值的设置后，再执行由 CreateFcn 定义的 Callback。

如果一个控件对象 CreateFcn 属性定义的 Callback 正在运行，那么该控件对象的句柄值不可访问，必须用函数 gcbo 取得。

- DeleteFcn 属性

DeleteFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当删除控件对象时，MATLAB 在清除该对象属性之前，要执行由 DeleteFcn 属性定义的 Callback。如果一个控件对象 DeleteFcn 属性定义的 Callback 正在运行，那么该控件对象的句柄值不可访问，必须用函数 gcbo 取得。

- HandleVisibility 属性

HandleVisibility 属性的取值为 on(默认值)、off 或 callback，这个属性定义控件对象句柄的可访问权限，决定其句柄值是否在父对象的 Children 属性中出现。如果 HandleVisibility 属性值是 on，那么它的句柄值总是可见的。如果属性值为 callback，那么该句柄值只在该对象的 Callback 调用以及 Callback 调用的函数内是可见的，但是命令行内调用的函数不能访问这样的控件句柄。也就是说，这样的句柄对于控件对象自己的 Callback 调用是私有的。这对于保护交互式的程序十分有用。如果 HandleVisibility 属性值被设置为 off，那么控件对象的句柄在任何时候都是不可见的。如果句柄是不可见的，那么任何查询句柄的函数都不能返回它的值，这些函数包括 get、findobj、gcf、gca、gco、newplot、cla、clf 和 close 等。当 HandleVisibility 属性值是 callback 或 off 时，这样的控件对象不会出现在图形窗口对象的 Children 和 CurrentObject 属性中，也不会出现在根对象 CallbackObject 属性中。但是可以将根对象的 ShowHiddenHandles 属性设置为 on，临时使所有的句柄都是可见的，而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍然是有效的，只要知道这样的句柄值，一切关于句柄的操作都是合法的。

- Interruptible 属性

Interruptible 属性的取值是 on 或 off(默认值)，这个属性决定控件对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 和 Callback 属性定义的 Callback 受 Interruptible 属性值的影响。在运行程序时，只有遇到 drawnow、figure、getframe 和 pause 等命令，MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

9.1.4　控件对象属性的修改

控件对象属性的创建和修改有两种方式——传统的基于命令行的方式和使用 GUI 界面的方式。两种方式的本质是相同的，都是调用 set 函数对句柄进行操作。但是，在编制程序界面时，GUI 界面显得比较简单好用；而在程序运行中，如果要创建新的对象、改变

已有的对象属性，就只能使用 uicontrol 和 set 语句了。因而这两种方法都很重要，而 set 函数的使用则是基础。本小节将通过例子进行具体介绍。

1. 使用命令行处理对象的属性

使用这种方法，控件对象的属性可以在程序开始时设置，也可以在程序运行中设置。在程序开始时，系统对于控件对象的属性给出默认值，但是往往不能满足要求，因而需要用户进行自定义。对此，可以使用函数 uicontrol，它的语法是：

```
H=uicontrol('PropertyName1',value1,'PropertyName2',value2,...)
```

H 是该函数的返回值，为所创建对象的句柄。

uicontrol 函数的具体用法前面已经介绍过，这里就不再详述了。

要创建某种控件对象，只要如前所述，将它的 Style 设为所需的值就可以了。为了方便查阅，列出表 9.1。

表 9.1 Style属性取值表

属性取值	所创建的对象
pushbutton	命令按钮
togglebutton	切换按钮
radiobutton	单选按钮
checkbox	复选框
edit	可编辑文本框
text	静态文本框
slider	滑动条
frame	框架
listbox	列表框
popupmenu	弹出式菜单

在 MATLAB 命令窗口中输入下面命令可以创建一个按钮对象：

```
Hbutton=uicontrol('Style','pushbutton');
```

说明：

- uicontrol 函数使用系统的默认值(Factory Value)作为默认值，但是系统默认值是可以修改的。因此在需要创建大量对象时，可以考虑修改默认值(方法见后)，进而创建出大量符合自己要求的对象。
- 可以通过返回的句柄值，对图形对象进行操作。

下面是一些的例子，通过它们，用户可以了解和熟悉 uicontrol 的用法。

【例 2】 创建一个按钮，响应滑标条的输入值。假设滑标条的句柄是 sli8。

```
pbgetval=uicontrol(gcf,'Style','pushbutton',…
    'Position',[10 10 100 25],…
    'String','Get Slider Value',…
    'Callback','disp(get(sli8,"Value"))');
```

说明：

- Position 属性指明所创建的按钮对象在当前的图形窗口中的位置和大小；String 属性为按钮对象上的说明文字；Callback 属性定义了当用户单击按钮对象时，应用程序执行的操作和功能。
- 对于 Callback 属性，它可以是一小段 M-程序(如本例)，也可以是一个 M 文件名，例如 myfunction.m(后缀.m 可以省略)。

【例 3】　创建复选框

这个例子说明如何生成 2 个复选框，用来设置当前坐标系的某些属性值。其中，文本对象给出了控件的一个标题，控件对象的位置和大小的单位是相对单位(normalized)，Callback 属性定义的操作是设置一些属性值。

```
% 静态文本框的创建
txtaxes=uincontrol(gcf,'Style','text',…
'Position',[.1 .4 .25 .1],'Units','normalized',…
'String','Set Axes Properties');
% 复选框的创建
cbbox=uicontrol(gcf,'Style','checkbox',…
      'Position',[.1 .3 .25 .1],'Units','normalized',…
      'String','Box=on', …
      'Callback',['set(gca,"Box","off"),'…
               'if get (bcbbox,"Value")==1,',…
               'set(gca,"Box","on")',…
               'end']);
cbtdir=uicontrol(gcf,'Style','checkbox',…
      'Position',[.1 .2 .25 .1],'Units','normalized',…
      'String','TickDir=out',..
      'Callback',['set(gca,"TickDir","in"),',…
               'if get(cbtdir,"Value")==1,',…
               'set(gca,"TickDir","out"),',…
               'end']);
```

【例 4】　单选按钮的创建

下面的例子说明如何创建一个具有 2 个单选按钮的单选按钮组，用来设置当前坐标系坐标轴刻度的方向，即标记是朝向坐标系内还是坐标系外，二者必居其一。每个单选按钮检查另外一个单选按钮的状态，保证只有一个按钮的状态是 on。

```
% 静态文本框的创建
txtdir=uicontrol(gcf,'Style','text',…
'String','Select Tick Direction',…
'Position',[200 100 150 200]);
% 单选按钮的创建
tdin=uicontrol(gcf,'Style','radio',…
```

```
        'String','In',…
        'Position',[200 75 150 25],…
        'Value',1,…
        'Callback',['set(tdin,"Value",1),',…
                'set(tdout,"Value",0),',…
                'set(gca,"TickDir","in"),']);
tdout=uicontrol(gcf,'Style','radio',…
      'String','out',…
      'Position',[200 50 150 25],…
'Callback',['set(tdout,"Value",1),',…
         'set(tdin,"Value",0),',…
         'set(gca,"TickDir","out"),']);
```

说明：

Callback 执行的结果保证只有一个单选按钮的状态为 on，因为单选按钮的 Value 属性是这样定义的：如果单选按钮的状态是 on，那么属性 Value 的值是另一个属性 Max 的属性值，该属性值的默认值是 1；如果状态是 off，那么属性 Value 的值是单选按钮的另一个属性 Min 的值，它的默认值是 0。这样，通过对属性 Value 设定不同的值，可以得到单选按钮的不同状态。

在以上的三个例子中，都使用了 set 函数。set 函数的基本使用方法很简单，语法为：

```
SET(H,'PropertyName',PropertyValue)
```

其中，H 为图形句柄，PropertyName 为属性名，PropertyValue 为属性值。另外，set 函数还有 3 种调用方法：

- set(H,a)：这里，a 是一个结构型数组，它的域名是对象属性名，从而可以将域名所指示的对象属性用 a 中包含的、对应的属性值来赋值。
- set(H,pn,pv)：将在单元数组 pv 中定义的数值赋给 pn 中定义的属性名，H 为相应对象的句柄。其中，pn 必须是一个 1×N 维的矩阵，而 pv 可以是 M×N 维的矩阵，其中 M 等于 H 的长度(可以用 length(H)来得到)，这样才能保证正确赋值。
- set(H,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2,...)：使用本调用方法可以对同一对象的多个属性同时赋值，用法就不再详述了。

说明：

- A =set(H)将会返回句柄为 H 的对象的所有属性和可能的属性值，并保存在一个单元数组 A 中。
- 一个属性的默认值可以改变，方法是在属性名的前面加上一个字符串 Default。例如，要将一个静态文本框对象的默认颜色改为红色，可以使用下面的命令：

```
set(gcf,'DefaultTextColor','red')
```

- 有 3 个字符串，如果将它们赋给属性值，则有特殊的意义：

  default　使用默认值(从最近的父对象开始)

  factory　使用默认值

  remove　删除默认值

下面再看几个例子。

【例 5】静态文本标志和一个单行可编辑文本框的创建。利用它，用户可以在文本框中输入颜色映像名，而后回调字符串把它放到在图形中

```
Hc_label=uicontrol(gcf,' Style',' text ' ,...
                'Position', [10  10  70  20],...
                'String', 'Colormap: ' );
Hc_map =uicontrol(gcf,' Style',' edit  ' ,...
                'Position',[80  10  60  20],...
                'String' , ' hsv ' ,...
                'callback','coloormap(eval(get(Hc_map,''String''))) ' );
```

【例 6】下面的例子实现了一个滑标，可以用于设置视点方位角。用 3 个文本框分别指示滑标的最大值、最小值和当前值

```
vw  =  get(gca,'View' );
Hc_az  =  uicontrol(gcf,'Style','slider',...
          'Position', [10  5  140  20],...
          'Min',-90,'Max',90,'Value',vw(1)' ,...
           'CallBack',[...
     'set(Hc_cur,''String'',num22str(get(Hc_az,''Value''))),' ...
     'set(gca,''View'',[get(Hc_az,''Value'') vw(2)]) ' ]);
Hc_min  =  uicontrol(gcf,'Style','text',...
          'Position ' ,[10  25  40  20],...
          'String ' ,num2str(get(Hc_az,'Min')));
          num2str(get(Hc_az,'Min')));
Hc_max  =  uicontrol(gcf,'Style','text',...
          'Position',[110  25  40  20],...
          'String',num2str(get(Hc_az,'Max));
          Hc_cur  =  uicontrol(gcf, ' Style ' , ' text ' ,...
          'Position',[60  25  40  20],...
          'String' num2str(get(Hc_az,'Value')));
```

【例 7】建立图形颜色的弹出式菜单。回调函数把图形的 Color 属性值设定为所选值。每种与颜色相关的 RGB 值存储在弹出控制框的 UserDate 属性中。所有句柄图形对象的 UserData 属性仅仅为单独矩阵提供孤立的存储

```
Hc_fcolor  =  uicontrol(gcf,''Style'',''popumenu'',...
'Position',[20 20 80 20],...
'String','Black|Red|Yellow|Green|Cyan|Blue|Magenta|White',...
'Value',1,...
'UserData' ,[[0 0 0]; ...
 [1 0 0];...
```

```
 [1 1 0];...
 [0 1 0];...
 [0 1 1];...
 [0 0 1];...
 [1 0 1];...
 [1 1 1];...
'CallBack' ,['UD=get(Hc_fcolor,'UserData' );', ...
'set(gcf,'Color',UD(get(Hc_fcolor,'Value'),:))']);
```

【例8】建立一个框架，并把两个按钮和一个标志放入其中

```
Hc_frame  =  uicontrol(gcf,'Style','frame','Position',[250 200 95
65]);
Hc_pb1  =  uicontrol(gcf,'Style', 'pushbutton' , ...
           'Position',[255 205 40 40], 'String', 'OK' );
Hc_pb2  =  uicontrol(gcf,'Style','pushbutton' ,...
           'Position',[300 205 40 40], 'String', ' NOT ' );
Hc_lb1  =  uicontrol(gcf,'Style' ,'text ' ,...
           'Position',[255 250 85 10],'String' , ' Push Me ' );
```

说明：

在这个例子中，框架对象的位置向量是这样计算的：text 控件对象为 85 个像素宽，而根据定位和对象的大小，frame 对象的两边比它各宽出 5 个像素，保证框架是可见的。同理，可以得到框架对象的高度值。

2. 使用GUI界面处理对象的属性

MATLAB 提供了一个 GUI 界面的工具，用以创建修改控件对象。而控件对象的创建方法在前面已经介绍过，这里着重介绍属性的修改。

如前所述，先用 Guide 控制板创建一个活动的图形窗口，然后单击最上面的 Property Editor 图标，启动属性编辑器。也可以在命令窗口中输入 propedit 启动属性编辑器。

下面通过一个例子，说明属性编辑器的使用方法。

首先，用下面的命令创建图形窗口，并打开属性编辑器。

```
surf(peaks(25));
propedit(gcf)
```

打开的属性编辑器窗口如图 9.4 所示。

为了将图形窗口的背景颜色改为红色，可以使用下面的命令：

```
set(gcf,'Color','red');
```

用属性编辑器可以做同样的工作。在属性编辑器的 figure (#1)标题下，有一个文本框，称为属性域。在其中输入属性名 Color，不要引号。按 Enter 键后，Color 属性的默认值就会出现在另一个文本框中，称为值域。将该默认值改为 red(带单引号)或[1 0 0]，按 Enter 键后，相应的图形窗口背景颜色变为红色。如同使用 set 命令一样，也必须提供“属性名/属性值”对。

在上述的方法中，由于必须向属性编辑器提供相应的属性名，故使用起来不够方便。为了避免这一点不足，属性编辑器提供了属性列表框。在属性编辑器中，用 Show Property List 复选框打开属性列表框。图 9.4 给出的是属性列表框打开时的属性编辑器。在属性列表框中，列出了被编辑的图形对象的全部属性和当前的属性值。为了编辑某个属性，只要单击相应的属性列表项，相应的属性名和属性值就会出现在属性域和值域文本框中，然后改变相应的属性值就可以了。

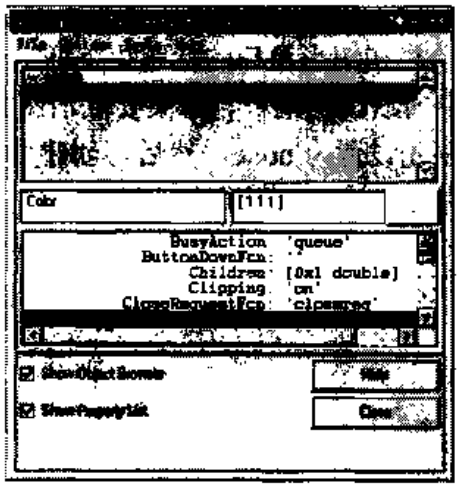


图 9.4　属性编辑器窗口

在前面的例子中，如果要改变图形窗口中坐标系对象的某个属性，可以用 propedit 命令：

```
propedit(gca)
```

将属性编辑器对准图形窗口中的当前坐标系对象。如前所述，对坐标系对象的属性进行编辑。另一方面，属性编辑器、Callback 编辑器和位置调整工具还提供了图形对象的浏览功能。这项功能可以在图形对象树型结构层次上管理图形对象。

可以用 Show Object Browser 复选框来打开或关闭图形对象列表框。在图形对象列表框中会列出当前图形窗口中的图形对象及其层次结构。对于前面的例子，只要在图形对象列表框中选择坐标系对象，即可将属性编辑器对准到坐标系，再对其属性进行必要的编辑。如果此时属性列表框是打开的，那么该图形对象的所有属性就在属性列表框中列出，如图 9.4 所示。

提示： 属性编辑器可以提供一些智能化的操作，以加快工作。

- 在属性编辑器中的图形列表框中可以选择一个或多个图形对象，甚至可以是不同类型的图形对象。此时，这些被选择的图形对象的公共属性出现在属性列表框中。
- 在属性编辑器的属性域文本框中可以输入部分属性名，长度只要可以用此将不同属性区分开来即可。例如，如果正在对图形窗口对象的属性进行编辑，那么在属性域中输入 Col，再按 Enter 键或 Tab 键，就可以代替输入整个 Color 属性名。
- 在属性域中，属性编辑器忽略含有空格的输入。所以，如果出现了输入错误，只要输入一个空格，然后按下 Enter 键，就可以清除原先的输入。

9.2 菜单对象及其属性

菜单对象(或称下拉式菜单对象)可以让用户在运行应用程序时，从一批功能选择项中浏览和选择某项功能。在 MATLAB 的每个图形窗口上，有一个主菜单栏，所有的主菜单都列在菜单栏上。菜单的标题或名字简单地描述了该菜单的功能。

在 Windows 系统中，菜单栏在图形窗口的标题栏下面。File、Edit、Windows 和 Help 菜单是 MATLAB 图形窗口中的默认主菜单。如果要取消图形窗口的默认的主菜单，可以将图形窗口的 MenuBar 属性事先设置为 none，然后再创建用户自己要定义的主菜单。

当用户单击主菜单条上的主菜单时，菜单会下拉式地打开，显示出该主菜单对象的命令或它的子菜单。选择一个命令就意味着让应用程序执行某种预定义的功能和操作。例如，在 File 主菜单中有一个命令 Print，当用户选择该命令时，即可以打印该图形窗口中的图形。

9.2.1 菜单对象的创建

与上面所讨论过的控件对象一样，菜单对象的创建也有两种方式——基于命令行的编程方式和基于 GUI 的方式。下面分别介绍这两种方法。

1. 基于命令行的方式

菜单对象的创建函数是 uimenu，可以用于创建菜单对象和创建命令对象，但其调用方式不同。这里所说的命令对象，简称菜单项，是指本身不再具有子菜单的功能选项，只对应于某种功能操作。而菜单对象或称子菜单项是指自身包含有下一级命令，它的功能就是打开它的子项。两者的区别在于：命令对象的 Children 属性值为空矩阵，而菜单对象的 Children 属性值为图形窗口句柄值或另一个菜单对象的句柄值。

具体地说，生成主菜单对象的调用形式是：

```
hmenu=uimenu(hfig,'属性名',属性值,…);
```

生成命令或子菜单对象的调用形式为：

```
hsub=uimenu(hfig,'属性名',属性值,…);
```

这两种形式的区别在于：创建主菜单时，要给出图形窗口的句柄值，如果省略了这个句柄值，MATLAB 就在当前的图形窗口中生成这个主菜单。如果此时不存在当前的图形窗口，MATLAB 会自动打开一个图形窗口，并将该菜单作为它的主菜单对象。在创建命令时，必须指定父菜单对象的句柄值。

其他属性的设置方式与控件对象属性的设置方式相同。

- 创建主菜单对象

使用上述第一种调用形式创建主菜单对象时，要注意返回一个句柄值，以备后面定义命令或子菜单对象之用。例如，为了定义一个名为 Plot Options 的主菜单，可以执行以下语句：

```
plotopt=uimenu(gcf,'Lable','Plot Options');
```

说明：

Label 属性值就是显示在图形窗口菜单条中的主菜单的标题或菜单对象的名字，plotopt 是该主菜单对象的句柄值，供定义命令或子菜单对象用。

- 创建命令

使用上述第二种调用形式创建命令时，必须提供对应的父菜单对象的句柄值，作为该函数的第一个输入参数。如果命令又是一个子菜单对象或其他的语句要引用，则应该记录下 uimenu 函数返回的句柄值。

下面的例子定义一个子菜单对象 Line Types 作为父菜单的命令。该父菜单对象的句柄值为 plotlt，各子菜单的 Callback 的工作是设置变量 ltype 之值：

```
plotlt(plotopt,'Lable','Line Types');
solid=uimenu(plotlt,'Lable','Solid line',…
             'Callback','ltype="—";');
dotted=uimenu(plotlt,'Lable','Dotted line',…
             'Callback','ltype=":";');
dashed=uimenu(plotlt,'Lable','Dashed line',…
             'Callback','ltype="——";');
```

- 创建子菜单对象

在创建子菜单对象的 uimenu 函数语句中，必须指明父菜单的句柄值，并返回子菜单对象的句柄值，供定义子菜单对象的命令之用。下面的例子定义菜单对象 Plot Options 的 3 个命令，每一个命令都是子菜单，用分隔条将每个子菜单分开。

```
plotlt=uimenu(plotopt,'Lable','Line Type');
plotsym=uimenu(plotopt,'Lable','Plot Symbol','Separator','on');
plotcol=uimenu(plotopt,'Lable','Plot Color','Seperator','on');
```

- 创建带检测标记的命令对象

MATLAB 对命令对象规定了一个属性 Checked，其属性值可以是 on 或 off，由此定义命令对象的两种不同的状态。当命令的状态为 on 时，该命令名字的左端具有一个检测的标记，可利用这个标记指示出对该命令所作出的选择。

例如，下面的语句使得当用户选择 Solid Line 命令时，就在该命令的左端作一个标记。由于一次只能选择一个命令，因此，标记只出现在一个命令上。

```
solid=uimenu(plotlt,'Lable','Solid Line',…
    'Callback',['ltype="-";'…
                'set(solid,"Checked","on"),',…
                'set(dotted,"Checked","off"),',…
                'set(dashed,"Checked","off")']);
```

- 创建切换式命令对象

MATLAB 可以让创建的命令对象在两种状态之间进行切换，而命令的不同状态是通过命令不同的标题来标识的。

下面的例子定义的命令具有 on 和 off 两种状态，由命令的标题的改换来表明命令所处的状态。两种状态的标题分别是 On Now 和 Off Now。当命令被选择时，菜单的名字就会改变，根据不同的状态来调用不同的子程序。

```
toggle=uimenu(uimenu,'Lable','On Now',…
    'Callback',['if strcmp(get(toggle,"Label"),'…
                "On Now"),',…
                'set(toggle,"Label","Off Now"),',…
                'fnoff,',…
              'else,',…
                'set(toggle,"Label","On Now"),',…
                'fnon,',…
              'end']);
```

2. 基于GUI的方式

菜单编辑器的界面如图 9.5 所示。使用菜单编辑器，既可以在某个图形窗口的菜单条上添加用户自定义的菜单，也可以编辑已定义的某个用户菜单。有一点要注意，在图形窗口成为活动窗口之前，用菜单编辑器加入或修改的用户菜单不会出现在图形窗口的菜单条上。因此，要看到所做的修改，必须进行切换。

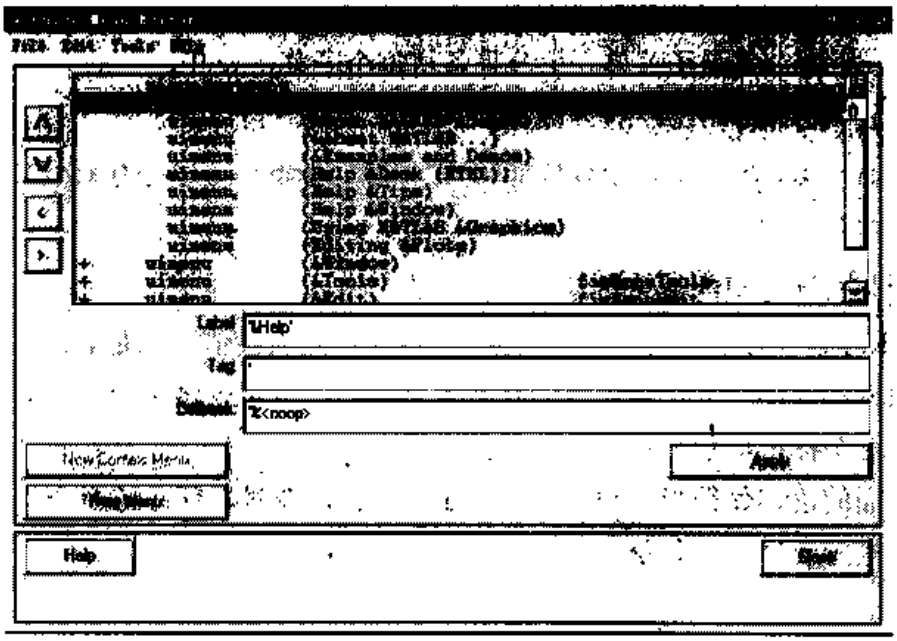


图 9.5 菜单编辑器窗口

菜单编辑器的使用比较简单，根据对应按钮上的提示，可以基本了解它们的使用功能和目的。这里就不再详述了。注意在左上方有 4 个小按钮，从上到下，它们对应的意义分别是：

- 使选中的命令向上移动一个位置，如果当前菜单已经是某个子菜单中的最上面的一项，移动后，它将成为前面一个比它高一级的子菜单对象的命令。
- 使选中的命令向下移动一个位置，如果当前菜单已经是某个子菜单中的最下面的一项，移动后，它将成为后面一个比它高一级的子菜单对象的命令。
- 使选中的命令提高一级。例如，二级子菜单变为一级子菜单。但是如果这个按钮变为灰色，则表示所选的菜单对象级别无法再提高了。
- 使选中的命令下降一级。例如，一级子菜单变为二级子菜单。但是如果这个按钮变为灰色，则表示所选的菜单对象级别无法再降低了。

注意： 使用菜单编辑器，只能看到菜单的 Label、Tag 和 Callback 这 3 个属性。如果要对别的属性进行修改，可以使用 Callback 编辑器或属性编辑器。

9.2.2　菜单对象的属性

1. 公共属性

- Children 属性

Children 属性的取值为空矩阵或句柄值向量，因为菜单对象可以有自己的子菜单对象。对于命令来说，Children 的取值就是空矩阵。改变向量元素的顺序，可以改变菜单对象的子菜单及命令的顺序。

- Parent 属性

Parent 属性的取值是对象句柄值，因为菜单对象总是图形窗口对象的子对象或另一个菜单对象的子菜单，该句柄值标明了菜单对象所在的图形窗口或其父菜单。如果用 set 函数将 Parent 属性值设置为另一个图形窗口的句柄值，则相当于将菜单对象转移到另一个图形窗口中。

- Tag 属性

Tag 属性的取值是字符串，它定义该菜单对象的一个标识值。定义了 Tag 属性后，在任何程序中都可以通过这个标识值找出该菜单对象。

- Type 属性

Type 属性的取值总是 uimenu，这个属性值标明图形对象的类型。对菜单对象，其类型就是 uimenu，用户不能改写这个属性。

- UserData 属性

UserData 属性的取值是一个矩阵，默认值为空矩阵。用户可以在这个属性中保存与该菜单对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。可以用 set 和 get 函数访问该属性。

- Visible 属性

Visible 属性的取值是 on(默认值)或 off。如果该属性值为 off，那么菜单对象是不可见的，但是该菜单对象仍存在，可以用 set 和 get 函数访问这个菜单对象的每个属性。

2. 基本控制属性

- Accelerator 属性

Accelerator 属性的取值是字符，只对 Children 属性值为空的对象，即命令对象才有定义。它定义该命令的热键。在 Windows 环境下，用 Ctrl 键加该字符，即可激活该命令的 Callback 功能。

- Callback 属性

Callback 属性的取值是字符串，可以是某个 M 文件名或一小段 MATLAB 语句。当用户激活菜单对象时，如果菜单对象没有子菜单就运行 Callback 属性定义的子程序(Callback 调用)。如果是子菜单对象，那么先运行 Callback 属性定义的子程序，再显示子菜单对象的子项。

- Checked 属性

Checked 属性的取值是 on 或 off(默认值)，该属性为命令定义一个指示标记，可以用这个特性指明某种选择状态。

- Enable 属性

Enable 属性的取值是 on(默认值)或 off，这个属性控制菜单对象的可选择性。如果它的值是 off，则此时不能使用该菜单。此时，该菜单标题的外观是阴影。

- Label 属性

Label 属性的取值是字符串，它定义菜单对象的名字。可以在字符串中加入&字符，那么在该菜单标题上，跟随&字符后的字符有一条下划线，&字符本身不出现在标题中。对于这种有带下划线字符的菜单，可以用 Alt 键加该字符键来激活。

- Position 属性

Position 属性的取值是数值，它定义主菜单对象在菜单条上的相对位置，或子菜单对象与命令对象在菜单组内的相对位置。例如，对主菜单，Position 属性值为 1，表示该菜单位于图形窗口菜单条的可用位置的最左端。

- Separator 属性

Separator 属性的取值是 on 或 off(默认值)。如果该属性值为 on，则在该菜单项上方添加一条分隔线，可以用分隔线将各菜单按功能分开。

3. Callback管理属性

- BusyAction 属性

BusyAction 属性的取值是 cancel 或 queue(默认值)，该属性决定 MATLAB 采取的控制中断执行菜单对象的 Callback 调用的方式。在 MATLAB 环境中，如果有一个 Callback 调用在运行，那么随后的 Callback 调用总是试图中断正在运行的 Callback 调用。如果此时 Interruptible 属性值为 on，那么 MATLAB 在处理事件队列时，中断正在运行的 Callback 调用；如果 Interruptible 属性值为 off，那么 BusyAction 属性就决定 MATLAB 处理事件的方式。如果它的值是 cancel，就从事件队列中取消企图运行第二个 Callback 调用的事件；如果它的值是 queue，就将企图运行第二个 Callback 调用的事件加入事件队列，直到当前的 Callback 调用完成为止。

- CreateFcn 属性

CreateFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当 MATLAB 生成菜单对象时，即 MATLAB 事先应该执行的程序段。这个属性只能按设置默认值的方式定义，例如，下列语句

```
set(0,'DefaultUicontrolCreateFcn',…
    set(gcf,"IntegerHandle","off")');
```

从根对象层设置菜单对象的默认值。对于已经存在的菜单对象，改变该属性的值，对该对象无任何影响。MATLAB 在生成菜单对象，完成所有默认属性值的设置后，再执行 CreateFcn 属性定义的 Callback。

如果一个菜单对象 CreateFcn 定义的 Callback 正在运行，那么该菜单对象的句柄值不可访问，必须用函数 gcbo 获取。

- DeleteFcn 属性

DeleteFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当删除菜单对象时，MATLAB 在清除该对象属性之前，先执行由 DeleteFcn 属性定义的 Callback。如果一个菜单对象 DeleteFcn 定义的 Callback 正在运行，那么该菜单对象的句柄值不可访问，必须用函数 gcbo 获取。

- HandleVisibility 属性

HandleVisibility 属性的取值为 on(默认值)、Callback 或 off。这个属性定义菜单对象句柄的可访问权限，决定其句柄值是否在父对象的 Children 属性中出现。如果 HandleVisibility 属性值是 on，那么它的句柄值总是可见的。如果属性值为 Callback，那么该句柄值只在该对象的 Callback 调用以及 Callback 调用的函数内是可见的，但是命令行内调用的函数则不能访问这样的菜单句柄值。也就是说，这样的句柄对于该菜单对象自己的 Callback 调用是私有的。这对于保护交互式的程序是十分有用的。如果 HandleVisibility 属性值被设置为 off，那么菜单对象的句柄植在任何时刻都是不可见的。如果句柄是不可见的，那么任何查询句柄的函数都不能返回它的值。这些函数包括 get、findobj、gcf、gca、gco、newplot、cla、clf 和 close 等。当 HandleVisibility 属性值是 Callback 或 off 时，这样的菜单对象不会出现在图形窗口对象的 Children 和 CurrentObject 属性中，也不会出现在根对象 CallbackObject 属性中。但是可以将根对象的 ShowHiddenHandles 属性设置为 on，临时使所有句柄都是可见的，而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍是有效的，如果知道这样的句柄值，则一切关于句柄的操作都是合法的。

- Interruptible 属性

Interruptible 属性的取值是 on 或 off(默认值)，这个属性决定着菜单对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 和 Callback 属性定义的 Callback 受 Interruptible 属性值的影响。在运行程序时，只有遇到 drawnow、figure、getframe 和 pause 等命令时，MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

9.2.3 菜单属性的修改

1. 基于命令行的方式

方法与前面相同，这里不再重复。总之，菜单对象作为 GUI 编程的一个基本构件，具有和其他 GUI 构件一样的控制方法。

2. 基于GUI的方式

如前所述，启动属性编辑器(如图 9.6 所示)，可以看到，uimenu 作为一个基本构件出现在对象列表中，可以用前面介绍过的方法来进行修改。

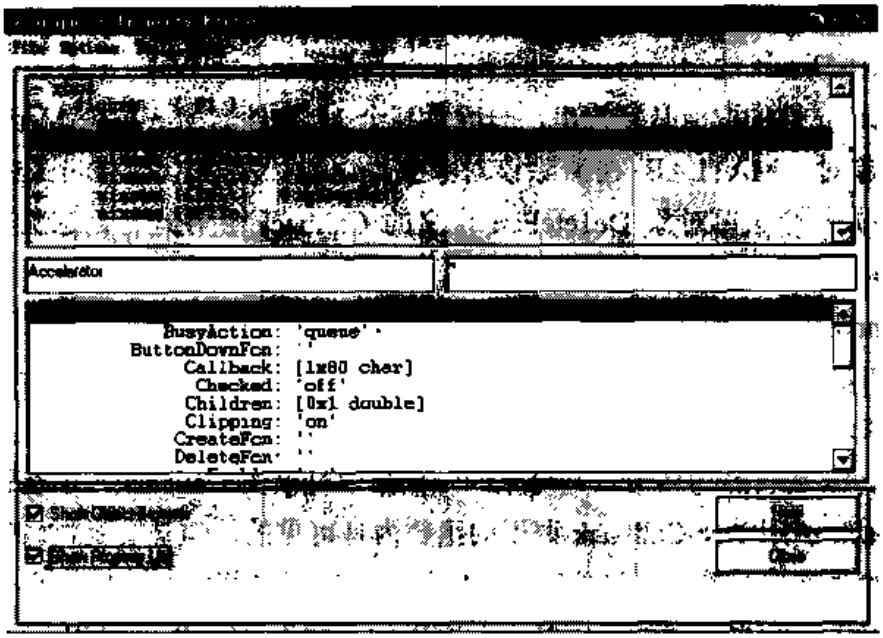


图 9.6 菜单属性的编辑器

9.3 GUI 的设计方法

9.3.1 使用函数替代 Callback

到目前为止，大部分例子中的 CallBack 属性值是由多条语句组成的字符串。当 Callback 要完成的工作很复杂时，这种方法是不适应的。一种有效的办法是根据某些规则，单独编写图形界面的应用程序，将用户界面的设计局限在一个函数内，在这个函数中按不同的选择来创建用户界面对象和定义有关的 Callback。这种方法至少有以下几个好处：

- 由于 MATLAB 的 Clear 命令会清除工作区中的有关变量，使用函数技巧可以保护图形界面中的有关变量不会被清除。

- 由于 Callback 与对象创建函数分开，更易于编写和修改，故对 Callback 函数进行调试时并不要求同时生成使用它的有关图形对象元。
- 执行速度更快，因为 MATLAB 只需要编译函数，而无需将 Callback 的值传送给 eval 函数解释。

在编写用户界面程序时，建议使用下面的规则：

- 用下列的语句定义一个函数，它的输入参数决定选用的函数和操作。

```
function uifun(action)
```

- 利用条件语句将生成界面控件对象的操作和 Callback 的操作分开。例如，在创建用户界面对象的代码之前使用下面的语句：

```
if strcmp(action,'initialize')
```

又如，对使用 ButtonDown 属性的 CallBack，可以用下列语句引导出有关的代码。

```
elseif strcmp(action,'linedown')
```

- 通过发布类似如下的命令来启动用户界面。

```
uifun("initialize")
```

- 定义特定的 CallBack 属性值，用适当的功能说明字符启动 uifun 函数。例如，下面的语句定义 lineobj 的 ButtonDownFcn 操作。

```
set(lineobj,'ButtonDownFcn','uifun("linedown")')
```

- 保护好用户界面的各层对象的句柄值和应用的有关数据，防止应用程序清除某些变量。为了使得每个函数启动时，这些句柄值和数据可供调用，或者将这些变量宣称为全程变量，或者将它们保存在 UserData 属性中。

1. 使用全程变量

使用全程变量是一种最为直接和有效的保护句柄值和应用数据的方法，但是它仍有以下两个缺点：

- 用户可能会输入命令 clear global，从而意外地清除某些图形界面中有用的全程变量。
- 如果应用程序打开多个图形窗口，则在没有其他的判别方法时，MATLAB 难以区分各个图形窗口的用户界面对象。

2. 使用UserData矩阵

同属性 Tag 一样，UserData 属性可在函数之间或递归函数的不同部分之间传递信息。如果需要多个变量，这些变量可以在一个容易辨识的对象的属性 UserData 中传递。如前所述，对与句柄图形对象在一起的单个数据矩阵 UserData 属性提供了存储。

将有关图形界面的句柄值存在 UserData 矩阵中是一种较为复杂的方法，而且要求一些额外的执行时间。但是与全程变量的方法相比，UserData 矩阵的方法至少有两条优点。

第一，只要图形窗口是打开的，则所有的句柄值和有关的变量值都存在。

第二，当用户打开另一个图形窗口时，它的控件对象的句柄值和有关的变量值不会与前一个图形窗口的有关值相混淆。

下面，针对上面的讲解给出一些例子。

【例 9】以脚本文件建立一个方位角的滑标

```
Avw=get(gca,'View');
Hc_az=uicontrol(gcf,'Style' , 'Slider',...
                'Position', [10 5 140 20],...
                'Min' ,-90, "Max",90,"Value",vw(1),...
                'Callback',[...
                'set(Hc_cur,"String",num2str(get(Hc_az,'Value'))),
' ,...
                'set(gca,"View",[get(Hc_az,"Value") vw(2)])']);
Hc_min=uicontrol(gcf, 'Style', 'text', ...
                'Position',[10 25 40 20],...
                'String' , num2str(get(Hc_az, 'Min')));
Hc_max=uicontrol(gcf, 'Style', 'text', ...
                'Position',[110 25 40 20],...
                'String' , num2str(get(Hc_az, 'Max')));
Hc_cur=uicontrol(gcf, 'Style', 'text' ,...
                'Position', [60 25 40 20],...
                'String' , num2str(get(Hc_az, 'Value' )));
```

【例 10】采用 Tag 属性来辨别控制框，并使用独立的 M 文件来执行回调

```
funtion setview( )
vw=get(gca, 'View');
Hc_az=uicontrol(gcf, 'Style' , 'Slider', ...
      'Position' , [10  5  140  20],...
      'Min' , -90, 'Max', 90, 'Value', vw(1), ...
      'Tag' , 'Azslider', ...
      'Callback' , 'svcback');
Hc_min=uicontrol(gcf, 'Style', 'text', ...
      'Position', [10 25 40 20], ...
      'String' , num2str(get(Hc_az, 'Min')));
Hc_max=uicontrol(gcf, ' Style', 'text', ...
      'Position' , [110 25 40 20], ...
      'String' , num2str(get(Hc_az, 'Max')));
Hc_cur=uicontrol(gcf, 'Style' , 'text' ,...
      'Position', [60 25 40 20], ...
      'Tag' , 'Azcur', ...
      'String' , num2str(get(Hc_az, 'Value')));
```

回调函数本身如下：

```
function svcback( )
```

```
vw = get(gca, 'View');
Hc_az = findobj(gcf, 'Tag' , 'AZslider');
Hc_cur = findobj(gcf, 'Tag', 'AZcur');
str = num2str(get(Hc_az, 'Value'));
newview =[get(Hc_az, 'Value') vw(2)];
set(Hc_cur, 'String', str)
set(gca, 'View', newview)
```

上面的例子并不节省很多代码，但却得到了用函数而不用脚本文件的优点：回调函数可以利用临时变量，而不使命令窗口工作区拥挤；不需要 eval 所需的引号和字符串；在回调函数中命令的句法变得十分简单。使用独立回调函数技术，越复杂的回调(函数)越简单。

独立回调函数的缺点是：需要很大数目的 M 文件以实现一个含有若干控制框和菜单项的 GUI 函数，所有这些 M 文件必须在 MATLAB 路径中可得，且每一个文件又必须要有一个不同的文件名。在对文件名大小有限制且对大小写不敏感的平台上，如 Windows，文件冲突的机会就增加了。而且回调函数只能被 GUI 函数调用而不能被用户调用。

9.3.2　递归函数调用

利用单独的 M 文件并递归地调用该文件，既可以避免多个 M 文件的复杂性，又可以利用函数的优点。使用开关 switches 或 if elseif 语句，可将回调函数装入调用函数内。通常这样一种函数调用的结构为

```
function guifunc(switch)
```

其中 switch 确定执行哪一个函数开关的参量，它可以是字符串 startup、close 和 sectolor 等，也可以是代码或数字。

【例 11】switch 控制的编程

- 如 switch 是字符串，则可如下面所示的 M 文件片段那样进行编程。

```
if nargin<1, switch = 'startup'; end;
if ~isstr(switch), error('Invalid argument'); end;
if strcmp(switch, 'startup'),
    % 此处的代码完成如下功能:
    % 创建 GUI 控制对象(包括控件对象和菜单对象)，完成 GUI 的基本功能
elseif strcmp(switch, 'setcolor'),
    % 对应于 SetColor 的回调函数
elseif strcmp(switch, 'close'),
    % 对应于 Close 的回调函数
end
```

- 如果是代码或字符串，也可以使用相似方式编程。

```
if nargin<1, switch = 'startup'; end;
```

```
if ~isstr(switch), error('Invalid argument'); end;
if switch == 0
     % 此处的代码完成如下功能:
     % 创建 GUI 控制对象(包括控件对象和菜单对象),完成 GUI 的基本功能
if switch == 1
     % 对应于 SetColor 的回调函数
if switch == 2
     % 对应于 Close 的回调函数
end
```

【例 12】使用全局变量、switch 等技术来完成对滑标条的设置

```
function setview(switch)
global HC_AZ HC_CUR
% 建立全局变量
if nargin < 1, switch = 'startup'; end;
if ~isstr(switch, error('Invalid argument.'); end;
vw = get (gca, 'View' );
% 这个句柄在两个程序段中都要用到
if strcmp(switch, 'startup' )
% 创建控件对象
Hc_AZ=uicontrol(gcf, 'Style', 'Slider', ...
               'Position', [10 5 140 20], ...
               'Min' , -90 , 'Max' , 90 , 'Value' , vw(1), ...
               'Callback' , 'setview('set')');
Hc_min=uicontrol(gcf, 'Style', 'text' ...
               'Position', [10 5 40 20], ...
               'String' , num2str(get(Hc-AZ, 'Min' , )));
HcMax=uicontrol(gcf, 'Style' , 'text', ...
               'Position' , [110 25 40 20], ...
               'String' , num2str(get(Hc_AZ, 'Max')));
Hc_cur=uicontrol(gcf, 'Style' , 'text', ...
               'Position' , [60 25 40 20], ...
               'String' , num2str(get(HC_AZ, 'Value')));
elseif strcmp(switch, 'set')
               % 执行回调函数
               str=num2str(get(HC_AZ, 'Value'));
               newview= [get(HC_AZ, 'Value') vw(2)];
               set(HC_CUR, 'String' , str)
               set(gca, 'View' , newview)
end
```

【例 13】利用 UserData 属性存储、处理数据

```
function setview(switch)
if nargin < 1, switch =  'startup' ; end;
vw = get(gca, 'View');
% 这个句柄在两个程序段中都要用到
if strcmp(switch, 'startup')
% 创建控件对象
Hc_AZ=uicontrol(gcf, 'Style', 'Slider', ...
                'Position', [10 5 140 20], ...
                'Min' , -90 , 'Max' , 90 , 'Value' , vw(1), ...
                'Callback' , 'setview('set')');
Hc_min=uicontrol(gcf, 'Style', 'text' ...
                'Position', [10 25 40 20], ...
                'String' , num2str(get(Hc-AZ, 'Min' , )));
Hc_Max=uicontrol(gcf, 'Style' , 'text', ...
                'Position' , [110 25 40 20], ...
                'String' , num2str(get(Hc_AZ, 'Max')));
Hc_cur=uicontrol(gcf, 'Style' , 'text', ...
                'Position' , [60 25 40 20], ...
                'String' , num2str(get(Hc_AZ, 'Value')));
                set(gcf, 'UserData' [Hc_Az Hc_cur]);
                % 利用 UserData 属性保存图形句柄
elseif strcmp(switch, 'set')
                 % 执行回调函数
                 Hc_all = get(gcf, 'UserData');
                 % 取出对象图形句柄
                 Hc_Az = Hc_all(1);
                 Hc_cur = Hc_all(2);
                 str = num2str(get(Hc_Az, 'Value'));
                 newview = [get(Hc_Az, 'Value') vw(2)];
                 set(Hc_cur,'String', str)
                 set(gca, 'View', newview)
end
```

注意： 全局变量遵循 MATLAB 的规定，变量名要大写。不需要 tag 属性，且不使用它；另外因为对象句柄已经存在，不需要用函数 findobj 去获取，故回调代码比较简单。全局变量通常使一个函数更有效。

说明：

- 尽管一个变量在函数内说明为全局的，变量并不能自动地在命令窗口工作区中利

用，也不能在回调字符串内使用。但是，如果用户发命令 clear global，则所有全局变量都被破坏，包括在函数内定义的那些变量。

- 当单独的一个图形或有限个变量要被所有的回调(函数)利用时，全局变量使用和递归性函数调用都是有效的技术。对于包含多个图形的更复杂的函数，或用独立对象回调函数实现的情况，UserData 属性更合适。另外，只要可获得对象句柄，对象 UserData 的属性值在命令窗口工作区中是存在的。

9.4 单一选择的单选按钮组设计

通常，单选按钮组的功能是使用户从一组可选状态中选出单一的状态(on)，而其他的状态被自动设置为 off。然而，MATLAB 单选按钮组自身并不具有这种功能，尽管每个单选按钮可以是 on 与 off 两种状态，但不同的单选按钮的状态之间无任何必然的联系。要想让一组单选按钮的状态具有择一性，必须使用专门的程序技巧来实现。在上一节的例子中，提供了一种较为简单的方法。这里将介绍一种较为系统的设计方法。

如果应用程序要求通过多个单选按钮提供不同的选择状态，首先应该将有关的单选按钮组成一个逻辑组，然后设计一段程序，保证该组中的单选按钮只有一个按钮的状态是 on，而其他的按钮状态是 off。

例如，假设应用程序用一组单选按钮来控制联机段图形的三种顶点标记中择一地选择一种顶点标记，第一个单选按钮对应默认的顶点标记，它的 Value 属性值可以设置为 1。

于是，有

```
sym(1) = uicontrol ( fig, 'Style', 'radio',…
        'position', [100 300 75 25],…
        'String', 'Circle', 'Value' , 1 );
sym(2) = uicontrol ( fig, 'Style', 'radio',…
        'position', [100 275 75 25],…
        'String', 'Plus', 'Value' , 10);
sym(3) = uicontrol ( fig, 'Style', 'radio',…
        'position', [100 250 75 25],…
        'String', 'Star', 'Value' , 10);
```

为了使这组单选按钮中只有一个按钮的状态是 on，可采用下面的技巧。即将每个单选按钮的 UserData 属性值定义为组中其他单选按钮的句柄值构成的矩阵，然后，当用户选择一个单选按钮时，它的 CallBack 就检查组中其他的单选按钮，将原来状态为 on 的单选按钮的状态改变为 off。下面的语句用来设置句柄值：

```
for i =1 :3
    set(sym(i), 'UserData', sym( :, [1 : (i-1), (i+1) :3]))
end
```

而以下的语句定义一个 CallBack：

```
call = ['me = gcf, "CurrentObject"]; ' ,…
```

```
'if ( get(me, "Value" )==1), ' ,…
   'set( get (me, "UserData"), "Value", 0),', …
'else,',…
   'set(me, "Value", 1)',…
   'set( get (me, "UserData"), "Value", 0),', …
'end'] ;
set (sym, 'Callback', call);
```

如果 Min 和 Max 属性值被修改过，那么定义 call 变量的语句中的数值 0 和 1 最好换成 Min 和 Max 属性的值。

9.5 中断 Callback 的操作

通常情况下，一旦 MATLAB 启动某个 Callback，则 Callback 不会中途被打断地执行完毕。中途打断 Callback 的过程称为中断，引起中断的事件称为中断事件。程序设计者可以在设计某个对象的 Callback 时，指明它的 Callback 执行是否可以被其他的 Callback 或其他的某些操作所中断。当然，某些情况下，不允许中断(打断)正在执行的 Callback 过程是十分必要的。例如，当定义一个按钮启动某个较长的初始化过程时，就不希望被用户的某些操作，如鼠标的操作所打断。

对大部分的图形对象来说，它的 Interruptible 属性可以被用来设置是否可以对它的 Callback 过程进行中断操作。属性 Interruptible 的取值可以是 on 或 off。

当 Interruptible 属性值是 off 时，图形对象的 Callback 操作不能被其他的 Callback 操作所中断。在 MATLAB 5.x 中，图形对象的 Callback 操作是否能被其他的中断事件所中断，要由图形对象的 BusyAction 属性决定。

当 Interruptible 属性值是 on 时，意味着该图形对象的 Callback 操作可以被其他的 Callback 操作中断。

例如，下面的语句定义了一个按钮对象，它的 Callback 操作不允许被其他的 Callback 操作中断。

```
pbstart = uicontrol(fig, 'Style', 'push',…
        'Position' , [10 10 75 25],…
        'Interrutptible', 'no',…
        'Callback', 'myprog1' , 'ButtonDownFcn' , 'myprog2');
```

一般说来，Callback 的执行过程不会被中断，除非它遇到某种特别的 MATLAB 命令。这时，MATLAB 系统开始处理存储在事件队列中的某些等待处理的事件。下面将就 MATLAB 的事件以及事件队列的概念做些讨论。

9.5.1 事件及事件队列

MATLAB 有关数值计算和图形句柄设置操作的命令，一旦被系统获得后，即刻会被解释与执行。有些命令或动作，例如鼠标操作和重新刷新图形窗口的命令，在 MATLAB 系统中被分割成一个一个的事件(events)。

MATLAB 将这些事件临时存储在一个事件队列中。在图形对象的 Callback 程序执行过程中，MATLAB 遇到如下命令时开始检测命令队列：drawnow、figure(包括 gcf 和 gca 命令)、getframe 和 puase。遇到上述命令之一，MATLAB 将暂时挂起 Callback 程序的执行，转而处理事件队列中的事件。MATLAB 如何处理每件事件，依赖于各事件的类型和图形对象的 Interruptible 的属性值。如果图形对象的 Interruptible 属性值是 on，则对输入型事件，如按下鼠标按键、释放鼠标按键、移动鼠标指针等，仅执行与这些事件相关的 Callback 程序，如 WindowButtonDowFcn、WindowButtonUpFcn 和 WindowButtonMotionFcn 定义的 Callback；如果图形对象的 Interruptible 属性值是 off，则视 BusyAction 属性值进行事件处理。不论对象的 Interruptible 属性值如何，刷新图形的事件总是立即执行。

但是，如果引起 MATLAB 挂起 Callback 执行过程的是带 discard 参数的 drawnow 命令，那么上述三种类型的事件都不会被处理。

9.5.2 MATLAB 处理 Callback 的过程

MATLAB 按照下列步骤启动和处理 Callback。

(1) 在一个 Callback 的执行过程中，当 MATLAB 遇到 drawnow、figure、getframe 和 pause 等命令时，MATLAB 会挂起当前的 Callback 执行，转向处理事件队列。

(2) 如果导致挂起 Callback 执行的是带 discard 参数的 drawnow 命令，或图形对象的 BusyAction 属性值为 cancel，MATLAB 将删除事件队列中的所有事件，然后转向步骤(4)。

(3) 如果事件队列的第一个事件是刷新图形的事件，MATLAB 将执行有关的刷新工作，并开始处理下一个事件。如果事件队列中的第一个事件是启动另一个 Callback 的执行，MATLAB 将检测正在被挂起的 Callback 对应的图形对象的 Interruptible 属性。如果 Interruptible 属性值为 on，那么 MAILAB 就执行该中断事件的 Callback；如果该 Callback 中还包含有 drawnow、figure、getframe 和 pause 等命令时，MATLAB 将按步骤(1)到(4)循环执行。如果 Interruptible 属性 off，那么 MATLAB 将从事件队伍中删除该事件或根据 BusyAction 属性值对事件队列进行处理，然后再处理事件队列中的下一个事件，直至将事件处理完为止。

(4) 当事件队列为空时，MATLAB 重新启动被挂起的 Callback，继续执行该程序。

MATLAB 持续地处理 Callback，如果到达至另一个 drawnow、figure、getframe 和 pause 等，重复上述步骤。下面看两种情况。

1. 不可中断的callback

假设定义了一个按钮对象，它的 Interruptible 属性值为 off，该按钮对象的 Callback 程

序中包含一条 drawnow 命令。假设 MATLAB 正在执行按钮对象的 Callback，此时，如果用户在某个已经定义了 WindowButtonDownFcn 属性的图形窗口中按下鼠标按键，将按下列步骤处理有关的事件：

(1) 当 MATLAB 遇到 drawnow 命令时，MATLAB 检查事件队列，就会发现用户在某个图形窗口中按下鼠标按键的事件存在，并且该图形窗口的 WindowButtonDownFcn 属性定义了 callback。

(2) MATLAB 检查按钮对象的 Interruptible 属性，看看是否可以执行那个图形窗口的 WindowButtonDownFcn 定义的 Callback。

(3) 根据假设，由于现在 Interruptible 属性值为 off，于是 MATLAB 从事件队列中去调用按下鼠标按键这个事件。如果是在 MATLAB 5.x 系统中，还必须检查 BusyAction 属性值，再做处理。

(4) MATLAB 继续处理事件队列，并且执行刷新图形的事件，但不执行其他的 Callback。

(5) 原来的 callback 继续执行。

注意： 当 MATLAB 遇到的是带 discard 参数的 ghdrawnow 命令时，甚至连刷新图形的操作也不执行。

如果原来的 Callback 中不包含 drawnow(或 figure、getframe 和 pause)命令，那么不会处理按下鼠标按键这一事件，直到 MATLAB 执行完 Callback，回到它的起始状态。

2. 可中断的callback

假设定义一个按钮对象，它的 Interruptible 属性值为 on，该按钮对象的 Callback 程序中包含一条 drawnow 命令。又设某个图形窗口的 WindowButtonDownFcn 属性已被定义。一旦用户在这个图形窗口中按下鼠标按键时，MATLAB 便按下列步骤处理有关的事件：

(1) 当 MATLAB 执行到 drawnow 命令时，MATLAB 检查事件队列，就会发现在某个图形窗口中“按下鼠标按键”的事件存在，并且该图形窗口的 WindowButtonDownFcn 有定义。

(2) MATLAB 检查按钮对象的 Interruptible 属性，看看是否可以执行那个图形窗口的 WindowButtonDownFcn 定义的 Callback。

(3) Interruptible 属性值为 on，于是 MATLAB 开始执行 WindowButtonDownFcn 定义的 Callback。

(4) 在 WindowButtonDownFcn 定义的 Callback 执行完毕时，MATLAB 继续处理事件队列中的事件。处理完毕后，返回到原来的 Callback。

然而，在中断过程中，MATLAB 中断前的状态未被保留下来。如果原来的 Callback 中不包含 drawnow(或 figure、getfame 和 pause)命令，那么就不会处理按下鼠标按键的事件，直到 MATLAB 执行完 Callback，回到它的起始状态。

9.5.3 事件的处理

- 中断事件对 MATLAB 运行状态的影响

当 MATLAB 中断某个 Callback，转而处理其他某项事件时，MATLAB 不会保护它的当前状态及其相关的变量，如当前图形窗口、当前坐标系、对象属性、工作区的变量等。因此，被中断的 Callback 再次启动时，它的某些执行条件可能发生了变化。所以，必要时，Callback 程序应该自己设法保护它被中断前的某些状态和重要的变量值。

- 移动鼠标指针与事件队列

移动一次鼠标指针的操作过程可能在事件队列中被分割成多个“移动鼠标”的事件，这取决于计算机的系统设置，以及鼠标移动的单步距离大小和移动速度。在处理事件队列中的事件时，如果 MATLAB 发现移动鼠标指针的事件，它将检查事件队列中余下的事件，并会删除相邻近的移动鼠标事件，即把连续的几个移动鼠标事件当作一个移动事件来处理。

- 保护 Callback 不被中断

当执行某个 Callback 遇到 drawnow 命令时，MATLAB 开始处理事件队列。MATLAB 执行有关的刷新图形的命令时，不论该对象的 Interruptible 的属性值是什么，都可以用带 discarde 参数的 drawnow 命令来防止刷新图形事件的发生。在这种情况下，MATLAB 将删除事件队列中的所有事件。

9.6 鼠标的操作

根据 Windows 系统的体系结构，下面的这些鼠标动作被称为事件：

- 当鼠标指针落在图形窗口中，按下鼠标按键时。
- 释放鼠标按键时。
- 在图形窗口中移动鼠标指针时。

MATLAB 系统保持了对这种概念的支持。对图形对象，MATLAB 定义了许多 Callback 属性，以便程序对这些事件做出反应。

本节将具体介绍如何实现对上述情况的处理。

9.6.1 鼠标指针的位置

鼠标指针的位置和用户的操作方式决定了 MATLAB 如何执行它的某些特定的 Callback。例如，当用户单击鼠标按键时，鼠标指针的位置将确定出图形窗口中的一个当前的图形对象。用户对鼠标按键的操作方式和应用程序将决定出某个 Callback，并启动执行。为了准确地对鼠标指针所在位置进行描述，MATLAB 将其位置状态分为 4 种情况，并根据此时用户的动作来决定应该执行的 Callback。这 4 种情况是：

- 当指针落在一个控件对象上或菜单对象上时。
- 当指针邻近一个控件对象时。
- 当指针邻近或在某些图形对象上时。
- 当指针在图形窗口中，但既不邻近也不在控件对象或某些图形对象上时，这种情况称为鼠标指针落在图形窗口内。

这里，对“邻近”的概念做如下的解释：图形对象在图形窗口中占据一定的区域，如

果鼠标指针是落在某个图形对象区域的 5 个像素宽的外边界(不是对象区域本身)内，就称为鼠标是“邻近”该图形对象。

MATLAB 提供了专门的属性来定义图形窗口对象、控件对象、菜单对象和某些图形对象的 Callback。MATLAB 根据鼠标指针的不同位置，来决定启动不同层次的 Callback 程序。表 9.2 列出了鼠标指针位置、鼠标的操作方式和 Callback 启动属性之间的关系。

表 9.2　Callback调用方式

鼠标指针位置	用户处理方式	Callback 启动属性
落在控件对象或菜单对象上	按下鼠标按键	启动控件或菜单的 Callback
邻近控件对象	按下鼠标按键	启动图形窗口的 WindowButtonDownFcn 定义的程序，再启动控件对象的 ButtonDownFcn 定义的程序
邻近控件对象	释放鼠标按键	启动图形窗口的 WindowButtonUpFcn 定义的程序
邻近控件对象	移动鼠标指针	启动图形窗口的 WindowButtonMotionFcn 定义的程序
邻近或在图形对象(除控件对象和图形窗口对象外)上	按下鼠标按键	启动图形窗口的 WindowButtonDownFcn 定义的程序
邻近或在图形对象(除控件对象和图形窗口对象外)上	释放鼠标按键	启动图形窗口的 WindowButtonUpFcn 定义的程序
邻近或在图形对象(除控件对象和图形窗口对象外)上	移动鼠标指针	启动图形窗口的 WindowButtonMotionFcn 定义的程序
图形窗口内	按下鼠标按键	启动图形窗口的 WindowButtonDownFcn 定义的程序
图形窗口内	释放鼠标按键	启动图形窗口的 WindowButtonUpFcn 定义的程序
图形窗口内	移动鼠标指针	启动图形窗口的 WindowButtonMotionFcn 定义的程序

9.6.2　按下鼠标按键的处理

当用户在图形窗口中按下鼠标按键时，MATLAB 按顺序做下列处理：

(1) MATLAB 检测鼠标指针的位置，决定是否选择某个图形对象。如果一个图形对象被选择，则 MATLAB 将图形窗口的属性 CurrentObject 设置为该图形对象的句柄值。如果无图形对象被选择，属性 CurrentObject 就被置为图形窗口本身。

(2) MATLAB 设置图形窗口的 CurrentPoint 属性和 SelectionType 属性，以及根对象的 CurrentFigure 属性。这些属性的具体含义在后面讨论。

(3) MATLAB 将当前的图形对象置于选择栈的栈首，栈内顺序在“选取对象”一段中讨论。

(4) 如果用户选择了控件对象，则 MATLAB 仅启动该控件对象的 Callback，不再执行以下的处理工作。

(5) MATLAB 执行图形窗口的 WindowButtonDownFcn 定义的程序。

(6) MATLAB 执行所选对象的 ButtonDownFcn 定义的程序。

9.6.3 释放鼠标按键的处理

当用户释放鼠标按键，且鼠标指针所在的图形窗口的 WindowButtonUpFcn 的程序已定义时，MATLAB 按下列顺序进行处理：首先，MATLAB 更新图形窗口的 CurrentPoint 属性值；然后，MATLAB 执行图形窗口 WindowButtonUpFcn 定义的程序。如果该属性未被定义，使鼠标指针移到另一个图形窗口中，MATLAB 也不做任何处理。即要执行的是按下鼠标按键时，鼠标指针所在的图形窗口的 WindowButtonUpFcn 定义的程序。

9.6.4 移动鼠标指针的处理

当用户在图形窗口内移动鼠标指针，且图形窗口 WindowButtonMotionFcn 已定义时，根据已打开的图形窗口的数目和用户移动鼠标指针的方式，MATLAB 分别执行不同的处理过程。当应用程序使用一个图形窗口，或打开了多个图形窗口却只在一个图形窗口内移动鼠标指针时，执行下列处理工作：

(1) MATLAB 更新图形窗口的 CurrentPoint 属性值。

(2) MATLAB 执行图形窗口的 WindowButtonMotionFcn 定义的程序。如果使用多个图形窗口，且用户将鼠标指针从一个图形窗口移到另一个图形窗口，则执行下列的处理工作：

- 当用户移动鼠标指针但不按下鼠标按键时，称为纯移动。对包含移动指针的图形窗口，MATLAB 将更新它的 CurrentPoint 属性值，并执行该窗口的 WindowButtonMotionFcn 定义的程序。
- 当用户按下鼠标按键时，称为拖动。对于拖动时鼠标指针所在的图形窗口，MATLAB 会更新它的 CurrentPoint 属性值，并执行该窗口 WindowButtonMotionFcn 定义的程序；当指针移动到另一个图形窗口中时，该图形窗口的属性值不受任何影响。用户按下鼠标按键时所在的图形窗口称为这个过程的主控图形窗口。

注意： 如果包含鼠标指针的窗口未定义 WindowButtonMotionFcn，MATLAB 就不会更新它的 CurrentPoint 属性。

9.6.5 相关属性总结

本部分再介绍几个受鼠标操作影响的属性。

- 根对象的属性

根对象的 CurrentFigure 属性值为用户按下鼠标按键时鼠标指针所在的图形窗口的句柄值，这个属性不受释放鼠标按键和鼠标移动操作的影响。而 PointerWindow 属性值为包含鼠标指针的图形窗口的句柄值。

- 图形窗口的属性

WindowButtonDownFcn 属性定义当用户在图形窗口内(不在控件对象上)按下鼠标按键时 MATLAB 要执行的 Callback。

WindowButtonUpFcn 属性定义当用户释放鼠标按键时，MATLAB 要执行的 Callback。该图形窗口应该是用户按下鼠标按键时鼠标指针所在的图形窗口。

WindowButtonMotionFcn 属性定义当用户在图形窗口内移动鼠标指针时 MATLAB 要执行的 Callback。

CurrentObject 属性指出鼠标指针选择的图形对象，如果无图形对象被选择，它的值是当用户按下鼠标键时鼠标指针所在的图形窗口的句柄值。

CurrentMenu 属性指出当前所选择的菜单或子菜单。

CurrentPoint 属性值是由鼠标指针的位置定义的 x、y 坐标值，其单位由图形窗口的 Units 属性决定。在以下三种情况下，MATLAB 将更新 CurreritPoint 的属性值：第一，当用户按下鼠标按键时；第二，当用户释放鼠标按键，且 WindowButtonUpFcn 有定义时；第三，当用户移动鼠标指针，且 WindowButtonMotionFcn 有定义时。

SelectionType 属性指明按下鼠标按键的方式，即是否有附加的组合键 Shift 和 Ctrl 键，以及是否单击或双击鼠标按键。

- 坐标系的属性

坐标系的 CurrentPoint 属性包含图形窗口 CurrentPoint 属性定义的两个点的坐标值。坐标系 CurrentPoint 属性是将图形窗口的 CurrentPoint 属性值变换到坐标系的坐标值时导出的。

- 图形对象的属性

CallBack 属性定义用户在一定的条件下，应用程序应该执行的预定义的操作或功能。

ButtonDownFcn 属性定义用户在图形对象附近(称为热区)按下鼠标按键时，应用程序应该执行的操作或功能。

9.6.6　对象选择规则

如果某个对象(如线段对象)与其他的对象重叠，因而难以选择该对象时，MATLAB 使用两条规则决定出用户按下鼠标按键时所选择的对象：第一，由对象的栈序决定；第二，由对象的热区决定。在多个对象重叠的情况下，仍按上述的规则决定每次选择的图形对象。

- 对象栈序

当鼠标指针落在多个对象的重叠区中，用户按下鼠标按键时，对象的栈序被用来确定应该选择的对象。开始时，栈序反映了对象创建的顺序，最后生成的对象在栈首。当鼠标指针落在某个对象上，并按下鼠标按键时，就将该对象移到了栈序的首位。在三维空间中，并不是离观察者近的对象就在栈序的高位。为了确定出图形窗口中图形对象的栈序，可以查看图形窗口的 Children 属性值。子对象的句柄值向量的顺序即是当前的图形对象的栈序，栈首对象对应着该向量的第一个元素。

- 热区

一个对象的热区由该对象本身以及该对象图形的一定宽度的边界区域组成。不同类型的对象，其热区的形状不同。热区是计算机屏幕上的一个矩形区域，用户可以将鼠标指针置于对象的热区内，按下鼠标按键就选中热区中的对象。具体定义如下：

对线段对象，热区是由线段本身以及围绕线段周围 5 个像素宽的周边区域构成的。所

以线段对象的热区是线段本身加上 10 个像素宽度的区域。

坐标系对象的热区是坐标系框加上坐标系外的坐标轴标题区(Label 区)和坐标系标题区(Title 区)，但是不包括坐标系内的子对象或控件等。

曲面、平面片区域和文字说明对象的热区是包含这些对象的最小矩形区域。

在三维坐标系中，这个最小矩形区域是屏幕上的二维矩形区域，它恰好包含了曲面对象、平面片区域对象或文字说明对象。

控件对象的热区定义为控件对象本身占据的矩形区域再加上四周 5 个像素宽的区域。

- 热区操作的响应

当用户在图形对象的热区中按下鼠标按键时，MATLAB 将针对不同的热区类型做出不同的响应。下面对此略做分析。

对于前面定义的前三类的热区，当用户在热区按下鼠标按键时，就表明用户选中了相应的图形对象，MATLAB 会将该对象移动至对象栈序的栈首，也就是把这个图形对象移至了屏幕的最前端。如果它与其他的图形对象是重叠的，那么此时这个图形对象将挡住其他的图形对象。另一方面，如果被选中的图形对象的 ButonDownFcn 属性被定义了一个 Callback 子程序，MATLAB 还会启动该程序的执行。例如，下面的语句

```
line([0, 1], [1,0], 'ButtonDownFcn', Idisp("line")');
```

创建了一个线段对象，并定义了一个 Callback，Callback 的操作是在命令窗口中显示字符串 line。然后，用户将鼠标指针移到图形窗口中该线段附近(即线段的热区中)按下鼠标按键时，MATLAB 命令窗口中就会显示一行字符串 line。每按一次，字符串就被显示一次。

但有时并不能很顺利地选中某个希望被选中的图形对象。在这样情况下，改变坐标系的视点可以使选中图形对象的可能性增大。下面来分析一个较复杂的例子。

以下两条语句

```
patch('ButtonDownFcn','disp("line")');
line([ 1,0 ], [0, 1], [0, 2], 'ButtonDownFcn' , 'disp("line")');
```

创建 2 个图形对象。第一条语句以默认的方式定义一个平面片对象。执行第一条语句之后，MATLAB 在打开的图形窗口中创建一个二维的坐标系，并在其中创建一个三角形区域的平面片对象。执行第二条语句时，就在当前的坐标系中绘出一条直线段。由于使用的是低层线段创建函数 line，故它不改变当前的坐标系。尽管线段对象是三维空间中的线段，但是仍在二维坐标系中输出线段对象的俯视图，因为 line 函数不改变坐标系的视点。最后的输出图形如图 9.7 所示。

由于平面片对象的热区包含它的最小矩形区域，即图 9.7 中整个矩形部分。很明显，线段对象的热区包含在平面片对象的热区中。由于线段对象是后生成的，因而当前线段对象处在栈首的位置。联机段对象附近(即热区)单击时，MATLAB 命令窗口中显示出字符串 line，表明了线段对象被选中，运行了 ButtonDownFcn 属性定义的 Callback。但是，一旦偏离了线段对象(因为它的热区范围很小)，并单击了鼠标左键，那么平面片对象便被选中，该平面片对象将被移到对象栈首。此后，就不能再在平面片对象的热区中选线段对象了。

为了能够再次选中上述的线段对象，需要改变坐标系的视点。由于平面片对象是落在 x-y 平面内的，而线段对象是连接点(1，0，0)和点(0，1，2)的空间线段，所以可以改变坐标系的视点(目前是［0，90］)。将坐标系变成三维的坐标系，使得在新的坐标系中，线

段对象的热区不完全包含在平面对象的热区中。

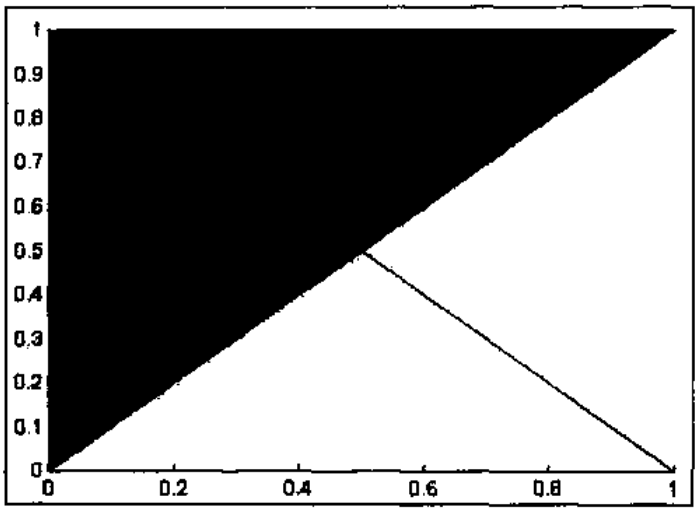


图 9.7　平面片对象和线段对象

对于第四类热区，即控件对象的热区，MATLAB 对于单击鼠标左键的响应与前述有所不同，其不同之处主要在于对 Callback 的处理。由于控件对象有两个属性，即 Callback 属性和 ButtonDownFcn 属性，可定义单击鼠标左键的 Callback 调用。所以，MATLAB 规定：在不同的位置上按下鼠标左键，执行不同的 Callback。

如果用户是在控件对象上单击鼠标左键，就执行 Callback 属性所定义的 Callback 调用；如果用户是在控件对象的热区，但又不是在控件对象上，即在控件对象周边 5 个像素宽的区域内单击鼠标左键，MATLAB 就执行由 ButtonDownFcn 属性所定义的 Callback 调用。在这种情况下还必须注意的是：虽然是在控件对象的热区中单击鼠标左键，但是，由于不是在控件对象上单击鼠标左键，因此，还相当于是在图形窗口中单击鼠标左键。当图形窗口的 WindowButtonDownFcn 属性有定义时，必须首先执行这个属性定义的 Callback，再执行控件对象的 ButtonDownFun 属性定义的 Callback。

9.6.7　应用举例

下面的例子说明如何实现在图形窗口中移动一个按钮对象的功能。

```
fig=figure('Color','w');
posvec=[10 10 75 25];
pb=uicontrol(fig,'Style','push','Position',posvec);
wbdcb=['set(fig,"WindowButtonDownFcn",['…
          ' "pv = get(fig, " "CurrentPoint" ");" ',…
          ' "posvec(1) = pv(1);" ',…
          ' "posvec(2) = pv(2);"])' ];
curpos = 'set(pb,"Position",posvec)';
set(fig, 'WindowButtonDownFcn', wbdcb);
set(fig, 'WindowButtonUpFcn', […
```

```
'set(fig, "WindowButtonMotionFcn", " "),', curpos])
```

下面对上述的语句略微解释。第一条语句打开一个白色背景的图形窗口。第三条语句创建一个控制按钮对象，它在图形窗口中的位置由向量 posvec 定义。第四、五条语句各定义了由一组语言构成的字符串向量。第六条语句将 wbdcb 的内容作为图形窗口 WindowButtonDownFcn 属性值。这样，如果用户在图形窗口按下鼠标按键，MATLAB 就执行 wbdcb 中的 MATLAB 语句。在 wbdcb 中定义的语句是设置图形窗口对象的 WindowButtonMotionFcn 属性的属性值，即定义在图形窗口移动鼠标时应该运行的 Callback。这样，一旦用户在控制按钮上按下鼠标按键，MATLAB 就为图形窗口定义 WindowButtonMotionFcn 属性值。当用户继续移动(此时还未放开鼠标按键)鼠标时，MATLAB 就会执行刚刚定义的 WindowButtonMotionFcn 属性的 Callback。也就是获取图形窗口的 Current Point 属性值，即鼠标指针的当前位置的坐标值，随后将坐标值赋给向量 posvec 的前两个分量。最后一条语句定义图形窗口的 WindowButtonUpFcn 属性。这样，当用户释放鼠标按键时，就执行该属性的 Callback，也就是重新将 WindowButtonMotionFcn 属性定义为空矩阵，即无任何操作。再运行 curpos 中的语句，即将控制按钮 pb 的 Position 属性置为更新后的 posvec，相当于将控制按钮移到了 posvec 决定的位置上。

这个例子定义了 3 个 Callback，分别关联于按钮对象、移动鼠标指针操作和释放鼠标按键操作。还有几点值得说明：

- 在上例中，WindowButtonDownFcn 的 Callback 操作定义了与 WindowButtonMotionFcn 相关的 Callback 操作。如果 WindowBuitonMotionFcn 是在创建图形窗口的语句中定义的，那么鼠标指针的任何移动都会移动按钮对象。这里是按下鼠标键时，才让 MATLAB 定义图形窗口的鼠标移动操作的 Callback。
- 毫无疑问，这种方法可以用来移动任何图形对象，如坐标系等。这是进行交互式程序设计的技巧。

9.7 GUI M 文件的调试

回调字符串在命令窗口工作区中计算并执行，这个情况对编写和调试GUI函数和脚本文件有某种隐含意义。回调字符串可以很复杂，尤其是在脚本文件中，这为语法错误提供了许多机会。记录单引号、逗号、括号是令人头痛的事。如果出现了语法错误，MATLAB给出提示。只要对象的Callback属性值是一个真正的文本串，MATLAB就认可了。只有当对象被激活并将回调字符串传给eval时，才检查回调字符串内部的语法错误。

这样让用户定义回调字符串，它涉及还未曾定义过的对象句柄和变量，这使编写相互参照的程序变得更容易。但是每个回调函数必须分别测试，以保证回调字符串是合法的MATLAB命令，并且回调字符中涉及的所有变量可在命令窗口工作区中是可利用的。

将回调函数像M文件一样编程或像GUI函数本身内的开关一样编程，就可以不运行整个GUI函数而对各个回调进行改变或测试。

因为回调字符串是在命令窗口工作区中而不在函数本身内计算，在函数与各回调之间传递数据就变得十分复杂。例如，函数test包含如下程序：

```
function test()
tpos1=[20  20  50  20];
tpos2=[20  80  50  20];
Hc_text=uicontrol('Style', 'text' , 'String' , 'Hello' , 'Position',
tpos1);
Hc-push=uicontrol('Style' , 'push' , 'String' , 'Move Text', ...
                'Position' , [15 50 100 25], ...
                'Callback', 'set(Hc-text, "Position", tpos2)');
```

所有语句都是有效的MATLAB命令，且回调字符串也为有效的MATLAB语句。文本对象和按钮出现在图形上。但当激活按钮时，MATLAB就提示错误。

如果test是个脚本文件，就不会出现这样问题，因为所有变量可在命令窗口工作区中使用。因为test是个函数，Hc_text和tpos2在命令窗口工作区中均未定义，回调字符串执行失败。

一种解决方法是使用各个字符串元素来建立回调，该字符串元素由数值而非变量建立。例如，改变回调字符串如下：

```
'Callback', ['set9', sprintf('&.15g ', Hc_text), ...
        ', "Position", ', ...
      sprintf( ' [%.15g  %.15g  %.15g  %.15g]', tpos2)')']);
```

建立了包括Hc_text对象句柄值的一个字符串，该值变换成具有15位精度的字符串，而tpos2变量转换成矩阵表示的字符串。在函数内计算sprinf语句，然后将所得的字符串用在回调中。在命令窗口工作区执行的实际命令如下所示：

```
eval('set(89.000244140625, "Position", [20 80 50 100])')
```

将一个对象句柄转换为字符串，必须保持全精度。上例中的变换，使用了小数点后15位数字的精度。在MATLAB中，句柄对象转换应使用这样的精度。

要记住，变量随后的变换不会改变回调字符串。在前面的例子中，在控制框定义之后改变tpos2的值就无效果。例如，在函数结尾处加命令

```
tpos2=[20  200  50  20 ]
```

就无效果。因为在tpos2重新定义之前，通过计算tpos2，回调字符串已经建立。

9.8 GUI 程序设计的其他问题

9.8.1 GUI 工具集中的其他工具

在 GUI 工具集中，还有其他一些有用的工具，下面分别介绍。

1. Callback编辑器

Callback 编辑器界面如图 9.8 所示。用 Callback 编辑器可以编辑修改某个图形对象的 Callback 属性值。在 Callback 编辑器中可以编辑多行的 GUI 代码。最方便的是，在 Callback 属性编辑器的文本框中可以省略 Callback 代码中所需的单引号。

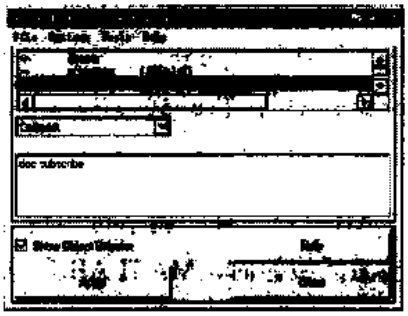


图 9.8 Callback属性编辑器的窗口

同样，还可以用 cbedit 命令启动 Callback 编辑器，也可以从 Guide 控制板中启动 Callback 编辑器。

可以直接用 Callback 编辑器对图形窗口的 Callback 进行修改。例如，为了修改 ButtonDownFcn 属性，首先用 Show Object Browser 复选框列出图形对象，选择 ButtonDownFcn，在 Callback 属性编辑器的文本框中编辑希望执行的代码，然后单击 Apply 按钮完成操作。

2. 位置调整器

位置调整器主要用于对图形对象的几何位置进行调整，使得 GUI 程序的界面更加美观。在命令行式的 GUI 程序设计中，需要仔细计算每个图形对象在窗口中的位置坐标，而且不能直接面对所设计的界面。无疑，这给程序设计带来了许多麻烦。利用位置调整器，这个难题迎刃而解。

位置调整器的窗口如图 9.9 所示。它由对象列表框和调整工具按钮组成。在选择了图形对象之后，就可以用各种按钮来移动被选择的图形对象，或调整它与其他图形对象的相对位置关系。

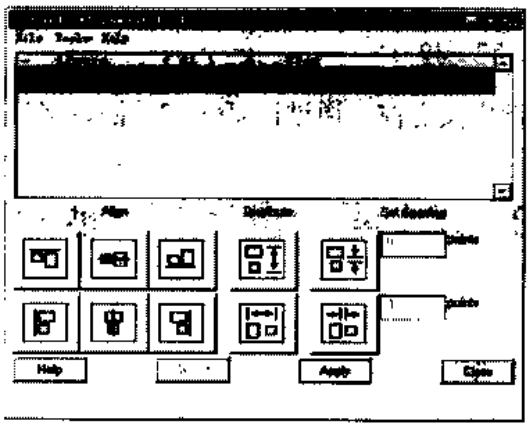


图 9.9 位置调整器窗口

9.8.2　对话框和请求程序

MATLAB具有建立对话框和“请求”的几个有用工具。对话框是弹出显示的单独窗口，它显示信息字符串。对话框含有一个或多个按钮以供用户输入。请求框是弹出显示的单独窗口，利用鼠标或键盘获得输入，并返回信息给调用函数。

1.　对话框

所有MATLAB的对话框都是基于函数dialog，它本身不是句柄图形对象，而是由一系列句柄图形对象构成的M文件。对话窗口是图形，包括与框架、编辑和按钮等控件对象共存的坐标轴。

预先定义的对话框是由函数helpdlg、errordlg、warndlg和questdlg建立。helpdlg和warndlg接受文本字符串和窗口标题字符串作为输入参量。errordlg接收replace变量作为输入。除了questdlg，所有上述函数都产生类似的对话框，它有各自默认的标题和文本字符串。单击OK按钮则关闭对话框。

对于helpdlg，它显示一个帮助文本框，语法为：

```
handle = helpdlg(helpstring, dlgname)
```

在对话框中显示标题为dlgname的帮助信息helpstring。如果名为dlgname的帮助对话框已在屏幕上显示，则引到屏幕正面，否则就建立该帮助对话框。

对于warndlg，它显示一个警告文本框，语法为：

```
handle = warndlg(warnstring,dlgname)
```

在对话框中显示标题为dlgname的警告信息warnstring。如果名为dlgname的警告对话框已在屏幕上显示，则引到屏幕正面，否则就建立该帮助对话框。

对于errordlg，它显示一个错误信息提示文本框，语法为：

```
handle=errordlg(errorstr, dlgame, replace)
```

建立显示出错信息errorstr、名为dlgname的出错对话框。要消除出错信息对话框，必须单击OK按钮，如果replace='on'并且名为digname的出错对话框已经存在，则引到屏幕正面，不再建立新的对话框。

对于questdlg稍有不同。前三种函数只显示一个按钮键并返回对话框图形对象的句柄。函数questdlg显示2个或3个按钮键，返回由用户所选按钮的标志字符串。questdlg的语法为：

```
click=questdlg(q, yes, no, cancel, default)
```

建立显示信息q的问题对话框。至多有3个按钮具有由yes，no和cancel给定的字符串，按钮与q一起显示在对话框中。根据所单击的按钮返回字符串click，对话框消失。

2.　请求程序

请求程序通过对话框获取用户的输入。请求程序是内置式GUI函数，它使用平台原有的窗口系统建立外观熟悉的请求程序。

- uigetfile 和 uiputfile 函数

函数uigetfile和uiputfile是所有平台上都有的内置式函数，用于交互地获得文件名，从而调用函数用它读取文件中数据或将数据存于文件中。

uigetfile通过显示对话框交互式地检索文件名，语法为：

```
[filename, pathname]=uigetfile('filterspec' , 'dialogtitle' , x, y)
```

它显示一个对话框，让用户输入并返回路径和文件名字符串。仅当文件存在时，才成功地返回。如果用户选择了一个并不存在的文件，就显示出错信息，控制框返回到对话框。用户可以输入另一个文件名或单击Cancel按钮。

所有输入参数都是可任选的，如果用其中之一，也必须使用所有先前参数。参数filterspec决定对话框中文件的初始显示。例如“*.m”列出所有的M文件。

参数dialogtitle是对话框标题字符串。x，y以像素为单位参数，定义对话框的初始位置。有些系统可能不支持这个选项。

输出变量filename是对话框内所选文件的名称字符串。如果用户按了取消按钮或有错误发生，filename的值设置为0。

输出参数pathname是对话框内所选文件的路径名字符串。如果用户按了取消按钮或有错误发生，pathname的值设置为0。

函数uiputfile与函数uigetfile相似，具有相似的输入参量，而且两者均返回文件和路径字符串，这里就不再详述了。如果用户选择了已经存在的文件，则出现一个对话框，询问用户是否要删除存在的文件。如果回答no，则返回原来的请求程序，等待另一次尝试；如果回答yes，则关闭请求程序和对话框并把文件名返回，但文件并未被请求程序删除。如果需要，调用函数必须删除或覆盖文件。

注意： *这些函数中不论哪一个都未真正地读或写任何文件，调用函数必须做这些工作。这些函数仅仅是将文件名和路径返回给调用函数。*

- 函数 uisetcolor

函数uisetcolor让用户交互式地选择颜色并选择性地将此颜色施加到一个对象上。它显示对话框，交互式地设置ColorSpace。语法为：

```
c=uisetcolor(arg, 'dialogtitle')
```

它显示一个对话框，让用户输入，并将所选颜色用于输入的图形对象。参数可任选，并可以以任何次序指定。

arg可以是一个图形对象的句柄或是RGB三原色。如果使用句柄，必须指定支持颜色的图形对象；如果使用RGB，必须是有效的RGB三原色(如[1 0 0]是红色)。在这两种情况下，所指定的颜色用于对话框的初始化。如果没有指定初始的RGB，则将对话框初始化为黑色。

如果使用参数dialogtitle，该参数是对话框名称的字符串。

输出值c是所选的RGB三原色。如果输入参数是句柄，则图形对象的颜色就设定为所选的颜色。

如果用户单击对话框中的Cancel按钮或有错误发生，输出值就设定为输入的RGB三原色；如果没有输入RGB三原色，则输出值设置为0。

例如：`c=uisetcolor(htext, 'settextcolor')`

第 10 章
Notebook

本章要点：

本章将介绍一些与 MATLAB 的 Notebook(笔记本)有关的内容。Notebook 是 MATLAB 与 Miscrosoft Word97 的完美结合，使用 MATLAB Notebook 可以同时利用 MATLAB 的强大计算功能和 Word97 完善的文本编辑功能。

本章具体包括以下内容：

- 如何安装 MATLAB Notebook
- 如何启动 MATLAB Notebook
- 如何使用 Notebook 的菜单
- 如何使用 Notebook 的格式
- 如何使用 Notebook 中的单元
- 如何解决 Notebook 使用中的问题

Mathwork 公司开发的 MATLAB Notebook 成功地把 Microsoft Word 与 MATLAB 集成为一个整体，为文字处理、科学计算、工程设计营造了一个完美统一的工作环境。它不仅拥有 Word 的全面文字处理功能，而且具备 MATLAB 无与伦比的数学运算能力和灵活自如的计算结果可视化能力。既可以把它看作能解决各种计算问题的字处理软件，也可以看作具备完善功能的科技应用软件。

从整体上来讲，几乎没有一个软件可以和 MATLAB Notebook 相提并论。拿流行的 Mathcad 来说，其字处理能力不如 Word，其计算可视化能力远在 MATLAB 之下，其数学计算能力更是无法和 MATLAB 相比。

MATLAB Notebook 是活的笔记本。在该笔记本中的计算指令可随时修改、即时计算并图示。对于讲授、编写理工学科教材讲义的教师，对于学习演算理工学科习题的广大学生，Notebook 的这个特性是十分宝贵的。

10.1 Notebook 的安装和运行

本节主要介绍 Notebook 的安装和启动，以及在 Word 中的 Notebook 运行环境。Notebook 简单易学，只要用户熟悉 Microsoft Word 和本书中介绍的 MATLAB 的使用环境，使用 Notebook 就易如反掌。

10.1.1 Notebook 的安装

MATLAB Notebook 是 MATLAB 系统中一个相对独立的部分，其安装程序和 MATLAB 主包的安装集成为一体。当 MATLAB 将 MATLAB 的主包和所有用户选取的 ToolBox 工具箱安装完成后，在安装程序的最后会检测用户的 Windows 系统。如果用户已经安装了 Microsoft Word 7.0 或者 Microsoft Word 97(考虑到 Microsoft Word 2000 即将推出，Mathwork 公司一定会在下一个版本中推出与之适应的 Notebook)，安装程序会自动弹出一个对话框，询问用户是否要安装 Notebook，如图 10.1 所示。

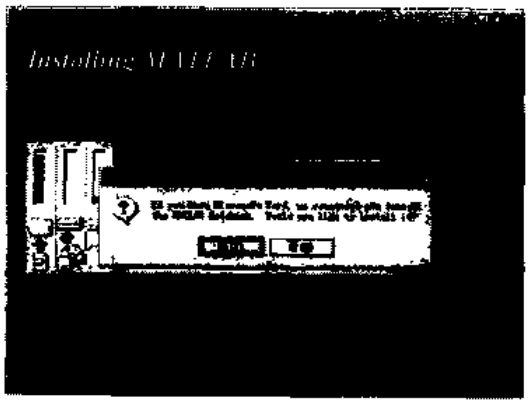


图 10.1 MATLAB安装

如果用户选择【是】，MATLAB 安装程序会继续询问用户安装对应什么 Word 版本的 Notebook，如图 10.2 所示：

图 10.2 Notebook选项

用户选择自己安装的 Word 的版本，这时 MATLAB 安装程序要求用户指定 Windows 中安装 Word 的目录，而其默认的目录为 c:\msoffice\winword，如图 10.3 所示。

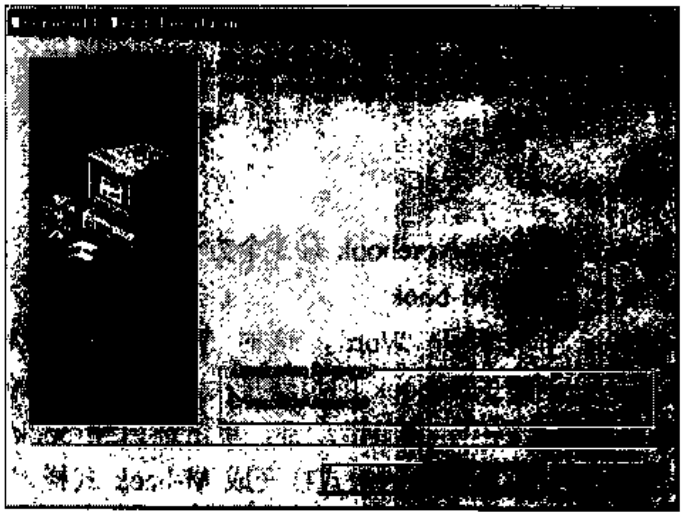


图 10.3 选择Word目录

单击 Browse 按钮，在对话框中选择正确的 Word 目录，安装程序就会正确地安装 Notebook。其过程为在 MATLAB 目录中建立 Notebook 子目录，然后把 Matlab\notebook 目录下的 m-book.dot 文件复制(或移动)到 winword\template 目录下。

如果程序不是靠安装软件 setup. exe 自动生成，而是复制上去的，那么应在 MATLAB 目录中建立 Notebook 子目录，然后将文件逐一复制到该目录，并在 Windows 目录下的 Matlab.ini 中的[Notebook Settings]组中增加如下内容：

```
Word path=C:\msoffice\winword
Word dot path=c:\program files\microsoft office
matlab-path=C:\MATLAB\bin
```

注意 在安装过程中，可能会弹出一个对话框如图 10.4，提示用户 Microsoft Word 97 程序有一个 Bug，在使用 Notebook 的过程中，MATLAB 输出到 Word 97 中的图形将不会正确地被打印。但是根据笔者的经验，这部分问题暂时还不是很大，用户可以放心地使用 MATLAB 的 Notebook。有关这个问题的解决方法将在本章有关图形输出的部分介绍。

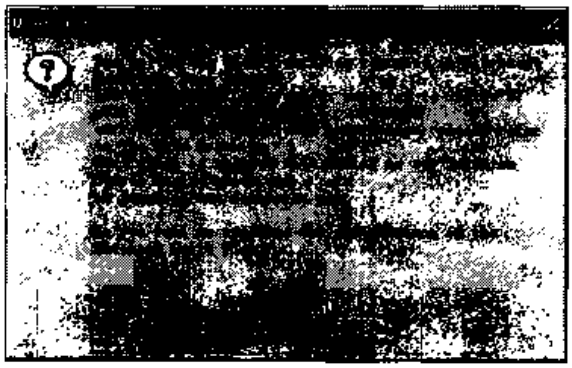


图 10.4 MATLAB安装弹出的警告信息

10.1.2 启动 Notebook

启动 Notebook(即 M-book)有两种方法：一是从 Word 中启动，一是从 MATLAB 指令窗中启动，详述如下。

1. 从Word中启动Notebook

一般来说，用户从 Word 中启动 Notebook 有 3 个方法。

- 在 Word 默认窗口下创建 M-book

假如用户在 Windows 下先打开 Word，这时 Word 窗口的默认模板是常用的 Normal.dot。在这种情况下，创建 Notebook 的步骤是：从 Word 窗口的【文件】菜单中选择【新建】命令；在弹出的对话框中，调节滑动键，单击选择 M-book 模板，单击【确定】按钮；于是，Word 的窗口布置由原先的默认式样变成 M-bcok 式样。假如此前 MATLAB 尚未启动，则 MATLAB 会自动被启动，用户可看到 MATLAB 的启动图标。MATLAB 启动结束后，便进入新的 M-book 文档。

- 在 Word "M-book" 窗口下创建 M-book

如果当前 Word 已经工作在 M-book 模板下，那么 Word 的窗口布置将与默认状态不同。在这种情况下创建新 M-book 文档的步骤是在【文件】菜单中选择 New M-book 命令。

- 在 Word 默认窗口下打开已有的 M-book 文件

在 Word 默认的窗口下打开已有 M-book 文件的方法与打开一般文件没有两样。最常用的方法是从【文件】菜单中选择【打开】命令，然后从弹出的对话框中选择所需要编辑的 M-book 文件。以后的步骤与方法 1 相同。至于其他打开文件的方法，请用户参考有关介绍 Word 的书籍。

2. 从MATLAB中启动Notebook

从 MATLAB 中启动 Notebook 比较简单，只需在指令窗中输入 Notebook 指令然后按 Enter 键就可以了，或者用 Notebook <M-BookFileName>命令打开一个已经存在的 Notebook 文件。为了更好地理解该指令的实质与用法，读一下 notebook.M 文件是有帮助的。

```
function notebook(fileName)
%NOTEBOOK Open an m-book in Microsoft Word (Windows only).
%       NOTEBOOK foo opens the M-book 'foo.doc' in the Notebook
%
%       NOTEBOOK, by itself, opens the Notebook for a new
%       M-book called 'document 1'

%  Copyright (c) 1984-98 by The MathWorks, Inc.
%  $Revision: 1.10 $

% get Word's startup path
wordPath        =        getprofl('Notebook        Settings','Word
path','c:\msoffice\winword','matlab.ini');

% get Word's template path
wordTempPath    =    getprofl('Notebook    Settings','Word    dot
path','c:\msoffice\template','matlab.ini');

if nargin == 0
   % create a new m-book
  eval(['!"'  wordPath  '\winword.exe"  '  '"'wordTempPath  '\m-
book.dot" ' '/mmwNewNotebookFromCmdLine&'])
else
   % open named m-book
   eval(['!"' wordPath '\winword.exe" ' fileName '&'])
end
```

从 notebook.m 可知，在运行 notebook 指令后，MATLAB 是把 Word 运行文件 winword.exe 作为外部命令来运行的。并且，当 notebook 指令后没有输入参量时，MATLAB 指定 m-book.dot 作为 Word 启动时的模板。如果有输入参量，即为打开一个已有的 Notebook 文件，从而启动 Notebook。

10.1.3 M-book 模板

如果用户不选用其他模板，Word 将会自动把 normal.dot 作为默认模板。

Notebook 的核心是 M-book 模板。M-book 模板为用户提供了在 Word 环境下使用 MATLAB 的功能。该模板定义了 Word 与 MATLAB 进行通讯的宏指令、文档样式和工具栏。

当调用该模板时，Word 便自动将 M-book 模板中定义的宏装入内存、运行 MATLAB、启动 NotetBook 用户界面以及定义文本的样式和单元。同其他 Word 的模板一样，用户既

可以修改 Notebook 模板原有样式，也可以加入新样式。

比如在现成的 M-book 模板中，输入(Input)单元是绿色的，输出(Output)单元和自活(AutoInit)单元是蓝色的，错误(Error)信息是红色的。如果用户想把输出单元的颜色变为黑色，那么可用以下操作实现：

(1) 选取【格式】菜单中的【样式】命令。

(2) 当【样式】对话框弹出后，在【样式】选项组中选择 Output，然后单击【更改】按钮。

(3) 当【更改样式】对话框弹出后，选择【添加到模板】选项，然后再选择【格式】菜单中的【字体】命令。

(4) 将其中的【颜色】选项改为黑色，单击【确定】按钮。

这样以后所有的输出单元都将是黑色的，除非再次更改它。

10.1.4 Notebook 菜单命令

M-book 模板窗口菜单栏，如图 10.5 与默认 Normal 模板不同之一是增加了 Notebook 菜单，它提供如下的命令：

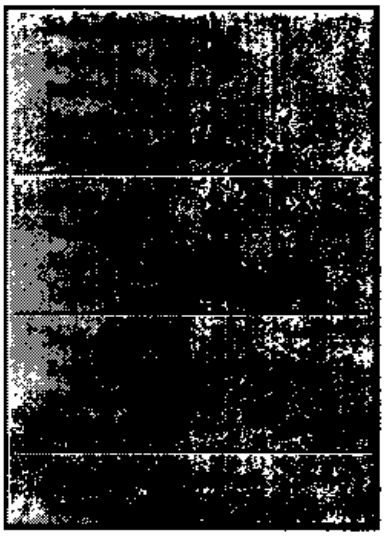


图 10.5 Notebook 菜单

Define Input Cell	Alt+I	自定义输入单元
Define AutoInit Cell	Alt+A	自定义自活单元
Define Calc Zone	Alt+Z	自定义计算区
Undefine Cells	Alt+U	将单元转换为文本
Purge Output Cells	Alt+P	清除输出单元
Group Cells	Alt+G	定义单元组
Ungroup Cells	Alt+p	将单元组转换为单个单元
Hide Cell Markers	Alt+H	隐藏单元标志

Show Cell Markers	Alt+C	显示单元标志
Toggle Graph Output for Cell		为每一个单元锁定图形输出
Evaluate Cell	Ctrl+Enter	运行当前单元或单元组
Evaluate Calc Zone	Alt+Enter	运行当前计算区
Evaluate M-book	Alt+R	运行 M-book 中所有的单元
Evaluate Loop	Alt+L	循环运行单元
Bring MATLAB to Front		将 MATLAB 系统置于屏幕之前
Notebook Options	Alt+O	定义输出显示的选项

M-book 模板窗口与通常不同之二是，在【帮助】菜单中加进了以下命令：

About Notebook	Notebook 版权说明

M-book 模板窗口异常的第三个地方是，在 File 菜单中增设了 New M-book 命令。

注意： 在以上新增菜单命令中，定义的快捷键与原先 Word 定义的某些菜单快捷键有冲突。凡发生冲突的快捷键都将按以上新定义发挥作用。要改变这种状况，用户必需修改有关快捷键的命名，具体的操作请参考有关 Word 的书籍。

10.2 Notebook 的使用方法

Notebook 定义了几种输入和输出的格式以供 MATLAB 系统分辨将要执行的函数和命令，这些格式包括 Input、Output、 Calc、AutoInit、 Nograph、 Normal、 Error 以及 Word97 中 Normal 模板中默认的格式。本节将具体介绍这些格式的功能和用法以及上节中提到的 Notebook 菜单命令的具体用法。

10.2.1 Notebook 格式的使用方法

本小节将介绍 Notebook 中各种文本格式的输入及其基本使用方法。

1. Word文档的输入

Notebook 启动后，可以按 Word 的正常方式输入文本。中文 Word 中默认段落文本的样式是“正文”，英文 Word 中默认段落文本的样式是 Normal，显示的文字外观采用 Word 中的默认设置。

2. MATLAB命令的输入与运行

在 Notebook 中创建和运行 MATLAB 命令有以下 2 种方法。

- 一步法

像平常输入文本那样，在新的一行里输入一条 MATLAB 命令；按组合键 Ctrl＋Enter，或在 Notebook 菜单中单击 Evaluate Cell。此后，该指令会变成绿色(即成为被选中的输入单元)，并在其下面行给出运算结果(即输出单元)。

- 二步法

在新的一行里输入一条 MATLAB 命令；在 Notebook 菜单中单击 Define Input Cell 命令，则该 MATLAB 命令的字体和颜色都将变成输入单元样式；再在此后单击 Notebook 菜单中的 Evaluate Cell 命令，于是在其下给出计算结果。

注意： 如果 MATLAB 命令被嵌在一段文本中，那么无论是想直接运行它还是想定义它为输入单元，都必须先把该命令(文本)选亮。否则，会出错，或者把嵌有该 MATLAB 命令的整个段落都定义为输入单元，显然有文字的段落是无法运行的。

下面举例说明命令的执行过程。为了表现 Notebook 中各种格式的颜色区别，这里没有按本书的样式来编排例子中的文字。

【例 1】产生一个四阶 Hilbert 矩阵

正文格式：本例用于产生一个四阶的 Hilbert 矩阵

```
hilb(4)   输入(Input Cell)格式

ans =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429   输出(Output Cell)格式
```

说明：

在 Notebook 中，MATLAB 命令被执行后所得的结果与在 MATLAB 指令窗中的结果完全一致。在没有输出参量时，输出结果自动赋给预定义变量 ans。

10.2.2 Notebook 中单元的使用

在 Notebook 中，MATLAB 命令有两种基本形式：一是输入单元(Input Cell)，一是自活单元(AutoInit Cell)。它们都可送到 MATLAB 环境中去运行。所得结果一方面保存在 MATLAB 的工作内存中，另一方面(假如要求的话)送回 Notebook。运算结果成为输出单元(Output Cell)。

1. 输入单元

凡在 MATLAB 环境中合法的命令、注释都可以定义为 Notebook 中的输入单元。一个输入单元可以是：一条单独的 MATLAB 命令；在一行内的几条 MATLAB 命令；包括注释在内的多行命令；嵌在文本中的命令。

输入单元在 Notebook 模板中预定义为绿色、10 磅大小、Courier New 粗体英文；作为注释的中文是默认段落字体。段落对齐方式、缩进与页面设置等其他格式采用 Normal 模板的设置。如前所述，这些定义是可以更改的。

下面介绍创建和激活输入单元的两种方法。

- 输入单元的创建

不管 MATLAB 命令在独立行还是嵌在文本中，先选中命令，然后在 Notebook 菜单中单击 Define Input Cell 命令，于是该 MATLAB 命令的字体和颜色都发生变化。这表明，MATLAB 命令已成为 Notebook 中的输入单元。这样创建的输入单元并没有送去运算。

- 输入单元创建与激活的同步实现

将光标置于独成一行的文本型 MATLAB 命令中，或选亮嵌在文本中的 MATLAB 命令，选取 Notebook 菜单中的 Evaluate Cell 命令，则不但该命令成为输入单元，而且被送去计算。运用快捷键 Ctrl +Enter 操作可替代以上菜单命令的选取。

【例 2】输入单元的创建

(1) 在英文状态下输入以下文本型指令：

```
x=0:0.1:2;y=x
```

(2) 把光标放在上述命令中，或用鼠标把它们选亮。

(3) 在 Notebook 菜单中单击 Define Input Cell 命令，于是该文本型命令就成为以下形式的输入单元：

```
x=0:0.1:2;y=x
```

(4) 把输入单元送去计算，用 Notebook 菜单中的 Evaluate Cell 命令实现，并产生以下结果：

```
y =
  Columns 1 through 7
         0    0.1000    0.2000    0.3000    0.4000    0.5000
0.6000
  Columns 8 through 14
    0.7000    0.8000    0.9000    1.0000    1.1000    1.2000
1.3000
  Columns 15 through 21
   1.4000   1.5000   1.6000   1.7000   1.8000   1.9000   2.0000
```

说明：

Define Input Cell 命令的功能仅仅是把文本命令变成输入单元。

【例 3】输入单元创建与激活的同步实现

在本例中先以文本方式输入以下(第一行)命令，然后按 Ctrl+Enter 键，得到以下形式

的输入、输出单元。

```
z=sin(x),p=exp(x)

z =
  Columns 1 through 7
        0      0.0998      0.1987      0.2955      0.3894      0.4794
0.5646
  Columns 8 through 14
    0.6442      0.7174      0.7833      0.8415      0.8912      0.9320
0.9636
  Columns 15 through 21
    0.9854      0.9975      0.9996      0.9917      0.9738      0.9463
0.9093
p =
  Columns 1 through 7
    1.0000      1.1052      1.2214      1.3499      1.4918      1.6487
1.8221
  Columns 8 through 14
    2.0138      2.2255      2.4596      2.7183      3.0042      3.3201
3.6693
  Columns 15 through 21
    4.0552    4.4817    4.9530    5.4739    6.0496    6.6859    7.3891
```

说明：

- 该例中变量 x 取自 MATLAB 工作内存。而变量 x 的保存是由上例最后的 Evaluate 计算指令完成的。
- 在 Notebook 中，是否显示计算结果的控制方法与在 MATLAB 指令窗中一样。命令后有分号“；”，则计算结果不会以输出单元显示；命令后有逗号“，”或不带分号，则显示。

2. 自活单元

自活单元与输入单元功能的唯一不同是：当用户启动一个 M-book 文件时，包含在该文件中的自活单元会自动被送去运算，而输入单元不具备这种功能。若用户需要在打开文件时，对 MATLAB 工作内存进行初始化工作，那么自活单元特别有用。

自活单元在 M-book 模板中预定义为深蓝色、10 磅大小、Courier，New 英文粗体。自活单元有两种来源，即文本形式的 MATLAB 命令和已经存在的输入单元。为把它们变成自活单元，先用鼠标选亮它们，然后运行 Notebook 菜单中的 Define AutoInit Cell 命令即可。

提示： 在此要强调指出：在文件启动以后，新定义的自活单元并不会自动运算，需另行运算操作。运行自活单元的方法同输入单元一样，选择 Evaluate Cell 菜

单命令或按 Ctrl+Enter 键。

3. 输出单元

输出单元包含 MATLAB 的输出结果：数据、图形和错误信息。数据的输出样式在 Notebook 模板中定义为蓝色、10 磅大小、Courier New 英文粗体；错误信息的输出样式是红色、10 磅大小、Courier New 英文粗体；图形的输出格式则通过 Notebook 菜单中的 Notebook Options 命令来设置，稍后详述。

输出单元是输入单元或单元组运算后产生的而不是人为定义的。输出单元总是紧跟在产生它的输入单元或单元组之后。假如，已带有输出单元的输入单元经修改后重新运行，那么将用新的输出单元替换原有的输出单元。

【例 4】输出单元的产生和修改

(1) 输入以下命令第一行并运行，得到输出单元

```
A=[5,6,7;2,8,9;-3,8,2],eig(A)
A =
     5     6     7
     2     8     9
    -3     8     2
ans =
    4.3150
   -3.1220
   13.8070
```

(2) 修改上面输入单元矩阵 A，再运行，则新的输出结果会覆盖掉原来的输出结果。由于这种修改运行操作只能在屏幕上表现，因此在书面上只能把修改运行后的最终结果重写如下。

```
A=[1,4,8;8,2,-4;12,3,-8],eig(A)

A =
     1     4     8
     8     2    -4
    12     3    -8
ans =
    9.7636
  -14.3349
   -0.4287
```

4. 单元组

单元组(Cell group)是多行输入单元或自活单元组成的一个整体。它有 3 种来源：

- 对输入的多行文本型 MATLAB 命令，用鼠标把它们同时选亮，然后在 Notebook 菜单中单击 Define cell 或 Define Autolnit Cell 命令，便生成单元组或自活单元组。

- 对输入的多行文本型 MATLAB 命令，用鼠标把它们同时选亮，然后在 Notebook 菜单中单击 Evaluate Cell 或操作快捷键 Ctrl+Enter，于是单元组被创建并激活。
- 把已有的多个独立输入单元或自活单元同时选亮，然后在 Notebook 菜单中单击 Group Cells 命令，于是便获得以第一个独立单元的(自活)性质组合而成的单元组。

假如被选亮的多个独立单元区域内夹带有独立成行的文本，那么在把独立单元组合成群的同时，将把原先夹在中间的文本内容移到单元组之后。

提示： Notebook 菜单中的 Group Cells 命令不能直接把文本型 MATLAB 命令组合成单元组。单元组被激活后将拥有一个输出单元(组)。在这个输出单元(组)中，输出数据的次序与命令在(输入)单元组中的前后次序相同，而图形输出总在数据之后。

单元组的用途主要有两个：

- 为保证 MATLAB 命令结构(如循环结构、条件结构)的完整，必须使用单元组。
- 为保证输出结果(如图形)的完整，必须使用单元组。

【例 5】对循环结构使用单元组，如图 10.6 所示

```
clear
x=0:10;
for k=1:10
    y=k*x;
    plot(x,y);
    hold on
end
hold off
```

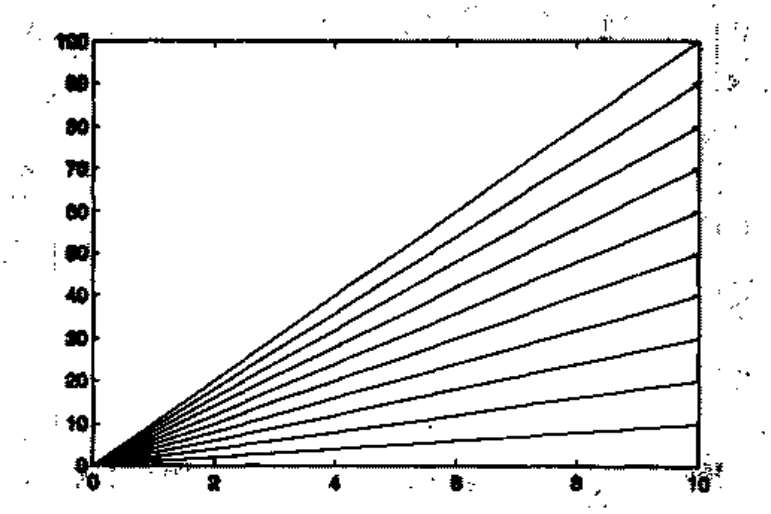


图 10.6 用循环语句画多线图

说明：

- 假如本例中的单元组用一行行独立的输入单元替代，那么运行它们时，将显示出错警告。

- 单元组运行后，不管输出数值结果的 MATLAB 命令在单元组中的什么位置，单元组在输出结果时，总是把数值结果放在图形结果的前面。
- 要想区分一个单元组和一组独立的输入单元，可以使用 Notebook 菜单中的 Show Cell Markers 命令。这个命令将把“整体”标志“[]”加在单元组的首尾两端；而对于一行行单独的输入单元，“整体”标志将分别显示在每个独立输入单元的首尾。
- 可以用 Notebook 菜单中的 Ungroup Cells 命令将单元组转换为一组单独的输入单元。

5. 计算区

计算区(Calc Zone)是一个由文本、输入单元和输出单元组成的连续区，用于描述某个具体的作业和问题。在计算区里，用户可以根据描述问题的需要安排段落、标题、格式和分栏，而不受计算区外有关格式定义的束缚。

Notebook 将计算区定义为 Word 中的一个节，并将节的分隔符号置于计算区的首尾，但是在文档的开始和结束的地方 Word 不显示节分隔符号。

创建计算区的方法是先选定包含输入单元、输出单元和文本的一个连续区，然后选择 Notebook 菜单中的 Define Calc Zone 命令。

要运行计算区，只需将光标置于计算区中的任何位置，然后选择 Evaluate Calc Zone 命令便可。

6. 单元转换为文本

单元(包括输入单元和输出单元)与文本不同，它们是“活”的。所谓“活”单元是指当用户改变输入单元并再次运行时，它的输出单元也会更新，即新的结果会覆盖掉原来的结果。假如用户希望保存原来的结果，就应该将“活”单元转换为“死”单元(即文本)。

单元转换为文本的方法是选定单元，运行 Noteback 菜单中的 Undefine Cells 命令；或将光标置于单元之中，按组合键 Alt+U。单元转换为文本后其样式会改变，由原来的样式改变为 Normal 样式，因此它的字体、大小、颜色均为 Normal 样式。

当某输入单元或单元组被转换为文本时，与之相应的输出单元也被自动转换为文本。然而当输出单元被转换为文本时，就切断了它与输入单元的任何联系，它不会随着输入单元的变化而变化，而输入单元的性质却不会因此而改变。此后，若再次运行输入单元，Notebook 会紧跟在输入单元之后产生一个新的输出单元。

10.2.3 Notebook 中 MATLAB 的使用

本小节将介绍有关 Notebook 使用 MATLAB 的特殊用法，这些用法与一般的 MATLAB 命令或者函数不同。

1. 工作内存的初始化

M-book 的所有计算是在 MATLAB 中进行的，参与运算的所有变量都储存在 MATLAB 工作内存里。M-book 文件和 MATLAB 指令窗口分享同一个“计算引擎”和同一个工作内存。

工作内存中的变量是各 M-book 文件和 MATLAB 指令窗口工作后共同产生的。对此，用户应有清醒的认识。

当用户同时打开几个 M-book 文件或在 MATLAB 指令窗口和 M-book 文件间交互运作时，要特别注意不同文件和窗口之间变量的相互影响。假如要保证某 M-book 文件独占 MATLAB 工作内存，保证该文件的输入输出数据间的一致性，一个有效的办法是把 clear 作为该文件中第一个输入单元或自活单元，以清除已经存在的变量对单元的影响。

2. 单元的循环运行

Notebook 提供循环运行单元的命令，具体步骤如下：

(1) 选择要重复运行的输入单元，可以包含合法的文字或输出单元。

(2) 选择 Evaluate Loop 命令，弹出如图 10.7 的对话框。

(3) 在 Stop After 文本框中输入重复运行次数，然后单击 Start 按钮。Notebook 开始运行命令，并显示已运行的次数。

如果要在每一个循环后加入延迟，可以单击 Slower 按钮；反之单击 Faster 按钮。若要暂停命令的运行，单击 Pause 按钮。停止运行命令，单击 Close 按钮。

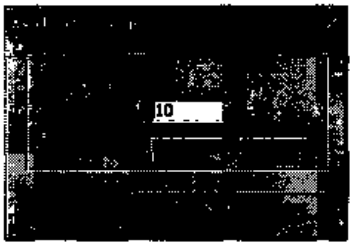


图 10.7 循环对话框

【例 6】利用循环运行命令绘制例 5 中的图形

先运行以下指令：

```
clear;x=0:10;k=1;hold on
```

选亮以下三行指令，打开循环运行对话框，并取循环次数为 10，便可启动。待循环完成后，关闭循环对话框，图形便绘制结束。所绘制的图形和原来的一样。

```
y=k*x;
plot(x,y);
k=k+1
hold on
```

3. 对M-book中所有单元的操作

在 Notebook 菜单中有两条对 M-book 文件中所有单元实现整体操作的命令，一条是

Evaluate M-book；另一条是 Purge Output Cells。

Notebook 菜单中的 Evaluate M-book 命令可以运行整个 M-book 文件，即把文档中所有单元送入 MATLAB 中去运行。不管光标处在该文档的什么地方，运行总是从文件首部开始。在整个 M-book 文件运行时，它不但会把所有原输出单元中的内容更新；而且会补写新的输出单元。这个命令在保证整个 M-book 文件中所有指令、数据、图形的一致性方面十分有用。

Purge Output Cells 命令的作用是删去 M-book 文件中的所有输出单元。在【编辑】菜单中选择【全选】命令，使整个文件被选中，然后再运行 Notebook 菜单中的 Purge Output Cells 命令，所有输出单元就被删去。这个指令在撰写报告、布置作业时常会用到。当然这个命令对于部分单元也可用，只要选中需要清除的区域，该部分的输出单元就会全部删除。

10.2.4 输出控制与文档的打印

本小节将介绍在 Notebook 中如何利用 Notebook Options 菜单命令来控制数据和图形的输出格式，以及在 Word 中有关打印 M-book 文件的问题。

1. 数据输出格式

在 Notebook 中，输出数据的格式是通过 Notebook 菜单中的 Notebook Options 命令进行的。该命令运行后，会出现如图 10.8 所示的对话框。

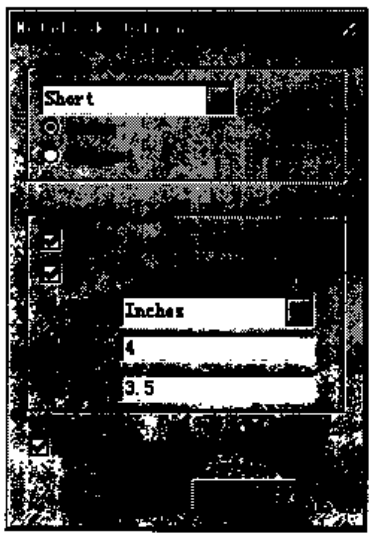


图 10.8 Notebook Options 对话框

- 数据格式控制

Numeric Format 选项组中的下拉列表中有 8 种可选格式：Short，Long，Hex，Bank，Plus，Shorte，Longe，Rational。它们的作用与 MATLAB 指令窗口中相应的 format 指令完全一样，详见第 2.1 节。

事实上，这 8 种格式也可以通过输入单元中的 format 命令来控制，其效果相同。

- 输出数据间的空行控制

Numeric Format 选项组中的 Loose 和 Compact 单选按钮用来控制输入单元与输出单元之间的空白区间。比如，选择 Loose 后，在 M-book 文档的输入单元和输出单元之间加入一个空行。

注意： 这种控制方法与输入单元组中的 format loose 和 format compact 命令有不同的功能。后者控制的将是输出单元与输出单元之间的空行。

【例 7】试比较几个不同的空行输出格式

在本例中，数据格式通过 Numeric Format 选项组中 Short 选项实现。

- 在 Numeric Format 选项组中选择 Compact 单选按钮后运行以下单元组。在输入单元和输出单元之间、以及输出单元和输出单元之间都没有空行。为了保持数据的原貌，在这里没有编排格式。

```
A=magic(3)
B=sqrt(A)
A =
     8     1     6
     3     5     7
     4     9     2
B =
    2.8284    1.0000    2.4495
    1.7321    2.2361    2.6458
    2.0000    3.0000    1.4142
```

- 假若用户在 Numeric Format 选项组中选 Loose 单选按钮，再运行以上单元组，将能看到在输入单元和输出单元间有空行。
- 若用户在 Numeric Format 选项组中选 Compact 单选按钮后，再运行 format loose 指令，则可看到输出单元和输出单元之间有空行。

【例 8】输入单元中数据格式指令的控制作用

在本例中，Numeric Format 选项组中选定 Short 和 Loose 选项不变。

- 短格式浮点表示：

```
format short e
A=sqrt(magic(3))
A =
  2.8284e+000  1.0000e+000  2.4495e+000
  1.7321e+000  2.2361e+000  2.6458e+000
  2.0000e+000  3.0000e+000  1.4142e+000
```

- 有理表示：

```
format rational
A=sqrt(magic(3))
A =
```

```
3363/1189        1          2158/881
1351/780     2889/1292     2024/765
   2             3          1393/985
```

提示： 在此再需强调，不同输出格式给出不同的数据显示精度，但内部存储及运算都是以相同的双精度进行的。

2. 图形输出

图 10.8 的 Notebook Options 对话框也实现对单元运算输出图形的控制。

- 黑白反色的控制

在中文 Word 的情况下，假如在图形输出前不对 Notebok Options 对话框进行操作，并单击 OK 按钮；也没在 MATLAB 指令窗中运行 Whitebg 指令，那么在 M-book 文档中单元运算输出图形将以黑色为背景。这种图在黑白打印机上的输出效果将是一片灰黑。

用户如果想得到白色背景的图形，必须进行以下两种操作中的一种：打开 Notebok Options 对话框，单击 OK 按钮；或在 MATLAB 命令行窗口中，运行 Whitebg 指令。

- 曲面色彩控制

假如在图形输出前，在 Notebook Options 对话框中不选择 Embed Figures in M-book 复选框(即选择小方块为空白)，那么生成的 256 色曲面图就可能得不到正确的色彩表达。更严重的是，在单色打印机上所得到的曲面图将是一片黑色。

- 图形镶嵌的控制

M-book 的默认设置是在 Notebook Options 对话框中选择 Embed Figures in M-book 复选框(即选择小方块中打钩)。在这种设置下，输出的图形被镶嵌在 M-book 文档中。假如不选择 Embed Figures in M-book 复选框，那么图形将输出到另外的图形窗口中。

- 图形大小的控制

在如图 10.8 所示的 Notebook Options 对话框下方有 3 个控制图形精确大小的控件：Unit、Width、Heigh。

在 Units 中可以设置 3 种单位：Inches、Centimetres 和 Points。Inches 是以英寸为单位，Centimetres 是以厘米为单位，Points 是以像素点为单位。

当然，所得图形也可用 Word 97 工具对它进行移动、缩放、剪裁和编辑。用户可以参考有关 Word 书籍。

【例 9】采用 16 色设置画复数立方函数图(图 10.9)

```
z=cplxgrid(20);
cplxmap(z,z.^3);
title('立方图形');
```

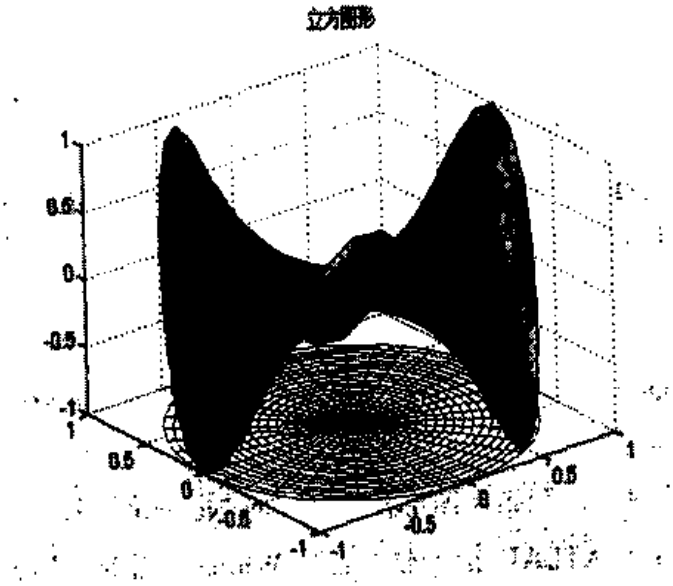


图 10.9 采用16色设置画复数立方函数图

说明：

MATLAB 的绝大多数图形都能在 M-book 中实现。但动画和某些交互式的图形指令例外，稍后详述。

3. M-book文档的打印输出

M-book 的打印输出与其他 Word 文档一样，按标准打印方式进行。当用户在【文件】菜单中选择【打印】命令后，会出现如图 10.10 所示的【打印】对话框。

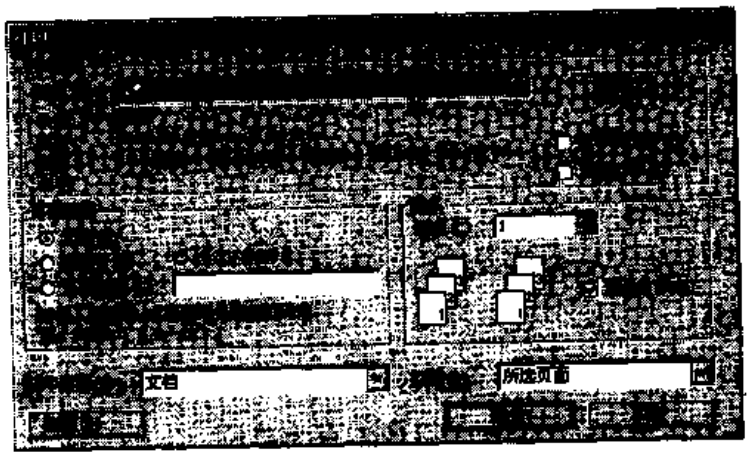


图 10.10 【打印】对话框

在这个对话框里可实现以下打印设置：

- 【打印内容】列表中有打印摘要信息、批注、样式、自动图文集词条或键值等选项。
- 【页面范围】选项组中有 3 种选择。
- 文档备份数目和奇偶页面选择。
- 是否以文件形式输出选择。假若在【打印到文件】复选框中打钩，则在【打印】对话框中单击【确定】按钮后，会出现【打印输出到文件】对话框。在此对话框

里，填写生成文件应在的驱动器名、目录名和文件名，然后单击【确定】按钮。这样所得的输出文件可以直接在 Windows 环境下输出到打印机上打印，即便不在 Word 环境下也行。

- 单击【打印】对话框中的【选项】按钮，可对是否后台打印、送纸方式进行设置。

在默认安装下，中文 Windows，提供两种 TrueType 汉字字体——宋体和黑体。这种比例缩放字体，使屏幕上显示的与打印纸上的实际结果一致。假如用户想安装其他汉字字体，请参阅有关手册。

最后，还要指出：单元标志(Cell Markers)属于 Word 控制符号，将不被打印出来。

10.3 Notebook 中的使用问题

Notebook 的使用涉及到 Windows 两大应用程序 MATLAB 和 Word 97 之间的通讯，其中一些较复杂的通讯问题有待解决。本节将对 Notebook 现行版本中的一些问题及使用中应该注意的问题给予说明。

10.3.1 Notebook 现行版本的问题

Notebook 现行版本的问题表现在对 MATLAB 包容上的缺陷。Notebook 已经包容了 MATLAB 的绝大多数功能，但尚有以下一些功能无法运行。

- Notebook 中无法运行的指令有：交互式指令(或与键盘交互，或与鼠标交互)；动画指令；程序调试指令，如 comet，dbclear，dbcont，dbdwnimput，dbstep，dbtype，dbup，ginput，gtext，input，keyboard，more，movie，pause 和 zoom 等。当然，包含上述指令的程序也不能在 Notebook 中运行。假如用户计算中必须使用上述的命令，那必须在 MATLAB 命令行窗口中进行，然后把结果复制到 Notebook 中。
- 带 GUI 控制和菜单的 MATLAB 图形被嵌入 Notebook 后不起作用。
- SIMULINK 不能在 Notebook 运作。

10.3.2 标点符号的问题

用户要特别注意中英文标点符号的区别。在输入中文文本时，当然应使用中文标点符号。但要记住：MATLAB 中的命令和命令组所用的是英文标点符号。千万不要在 MATLAB 命令、命令组、输入单元、单元组中错误地使用中文标点符号。不然，轻则出错，重则死机。

在标点符号中，尤其要注意冒号、分号、逗号的中英之分。

【例 10】MATLAB 对中文和西文标点符号的反应(由于出版印刷原因，无法从外观上形象地表达标点符号的区别)

```
x=1:7;                    %英文标点符号
y=log(x)
y =
        0    0.6931    1.0986    1.3863    1.6094    1.7918    1.9459
x=1:7;                    %中文标点符号
y=log(x)
rmat compact;x=1:7?                    |
Missing operator, comma, or semi-colon.
```

10.3.3 长文档中的输出单元问题

在长文档中，如果链接着输出单元的输入单元较多，那么它们间的链接关系有可能会被搞乱。当某个输入单元被运作时，它的输出结果有可能错误地更新其他输入单元的输出。

为防范可能的混乱，有两种措施可供使用。一种是将单元转换为文本，从根本上切断输入单元与输出单元之间的链接；另一种是对一定数量的单元用“计算区”界定，以确保在各计算区内的单元链接关系过于繁杂。

第 11 章
SIMULINK 仿真初步

本章要点：

本章将介绍一些与 SIMULINK 仿真系统有关的内容。SIMULINK 是个 MATLAB 的仿真工具箱，是 MATLAB 另外一个比较强大的功能。使用 SIMULINK 仿真系统可以帮助实现一些对现实系统进行仿真的功能。

本章具体包括以下内容：

- 如何在 SIMULINK 中创建一个简单的仿真模型
- 如何在 SIMULINK 中构造一个模型
- 如何给一个构造好的模型进行仿真
- 如何进行仿真结果的分析
- 如何封装产生一个新的模块

SIMULINK 是一个用来对动态系统进行建模、仿真和分析的软件包，它支持连续、离散及两者混合的线性和非线性系统，也支持具有多种采样速率的多速率系统。

SIMULINK 为用户提供了用方框图进行建模的图形接口，使得建模就像用纸和笔来画一样容易。它与传统的仿真软件包相比，具有更直观、方便、灵活的优点。SIMULINK 包含有 Sink(输出方式)、Source(输入源)、Linear(线性环节)、Nonlinear(非线性环节)、Connectors(连接与接口)和 Extra(其他环节)子模型库，而且每个子模型库中包含有相应的功能模块。用户也可以定制和创建用户自己的模块。

用 SIMULINK 创建的模型是分层的，因此用户可以采用从上到下或从下到上的结构来创建模型。用户可以从高级查看模型，然后双击模块来查看下一级中更加详细的内容。这种方法使得用户可以深入地理解模型的组织结构和各部分是如何相互作用的。

在定义完模型后，就可以利用 SIMULINK 的菜单或者 MATLAB 的命令窗口输入命令对模型进行仿真。菜单方式对于交互工作特别方便，而命令行方式对大量重复仿真很有用(如 Monte Carlo 仿真)。利用 Scope 或者其他的显示模块，可以在仿真的同时看到仿真的结果。另外，用户还可以改变参数并立即看到结果。仿真的结果存放到 MATLAB 系统的工作区以作后处理和可视化。

模型分析工具包括线性化和平衡点分析工具、MATLAB 的许多工具及 MATLAB 的应用工具箱。由于 MATLAB 和 SIMULINK 是集成在一起的，因此用户可以在这两种环境下对自己的模型进行仿真、分析和修改。

MATLAB 5.3 提供了一个新的 SIMULINK 工具箱，也就是最新的 SIMULINK 3.0 版本。这个新版本在用户界面、模块、建模、仿真、S 函数以及 SB2SL 2.0 上都做了新的改进，功能更加强劲，使用更加方便。本章将分别介绍 SIMULINK 的建模和仿真分析指令。还将介绍在 SIMULINK 中运用越来越广泛的 S-FUNCTION(S 函数)的编写。

11.1 快速入门

本节通过运行一个演示程序和创建一个简单的模型来帮助 SIMULINK 的新用户快速入门。

11.1.1 运行一个演示程序

与 SIMULINK 一起提供给用户的一个演示程序，模拟的是一个房子的热动力特性。要运行这个程序，可采用下面的步骤：

(1) 双击 MATLAB 图标打开 MATLAB 程序。

(2) 在MATLAB的命令行窗口输入thermo运行这个演示程序。这个命令启动SIMULINK并且产生包含这个模型的窗口，如图11.1所示。当用户打开模型时，SIMULINK打开一个Scope模块，包含有 Indoor vs. Outdoor Temp和Heat Cost ($)两个子图。

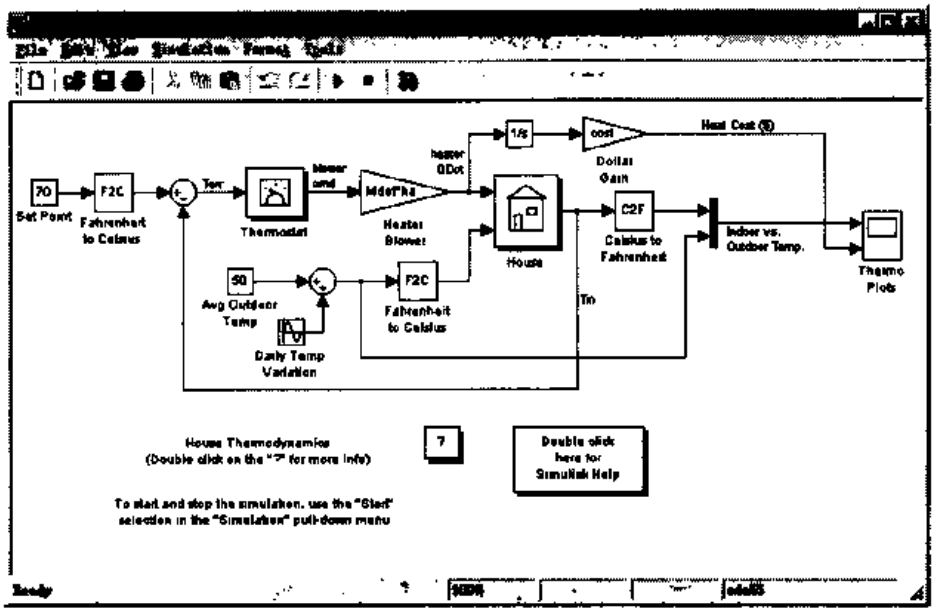


图 11.1　SIMULINK3.0 界面

(3) 要进行仿真，在 Simulation 菜单中选择 Start 命令，SIMULINK 就在 Indoor vs. Outdoor Temp 子图中输出房子内外的温度，而在 Heat Cost($)子图中输出累积的热损耗。

(4) 要终止仿真可在 Simulation 菜单中选择 Stop 命令。

(5) 在 Simulate 的 File 菜单中选择 Close 命令来关闭模型。

11.1.2　演示程序的说明

本演示程序采用一个简单的模型来模拟一个房子的热动力学特性。恒温器的温度设为 2℃，并且会受到外界温度干扰的影响。这个干扰用基本温度为 10℃、幅值为 9℃的正弦波来描述。

1.　各子系统介绍

内部和外部的温度输入给 House 这个子系统，并不断地更新内部的温度。用鼠标双击 House 子系统块，就可以看到这个子系统所包含的功能模块，如图 11.2 所示。

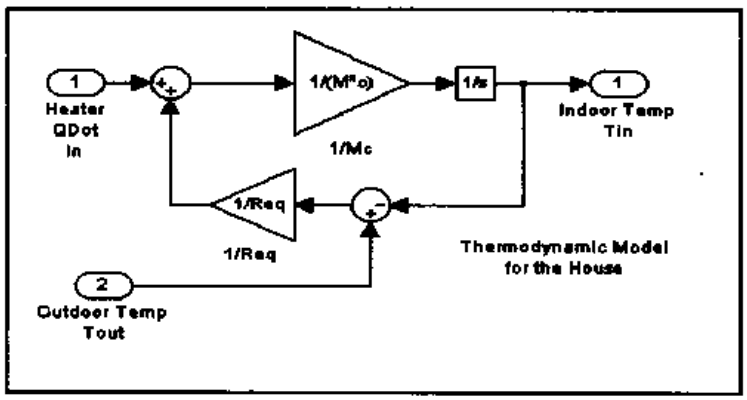


图 11.2　房子的热动力特性系统

Thermostat 子系统用来模拟一个恒温器的工作，确定何时加热器打开或关闭，用鼠标

双击 Thermostat 子系统模块，就可以看到这个子系统所包含的模块，如图 11.3 所示。

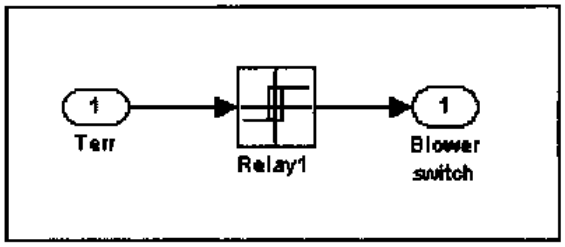


图 11.3 恒温器子模型

通过 Centigrade 子系统模块可以把外部温度和内部温度转换为摄氏温度。Centigrade 子系统所包含的模块如图 11.4 所示。

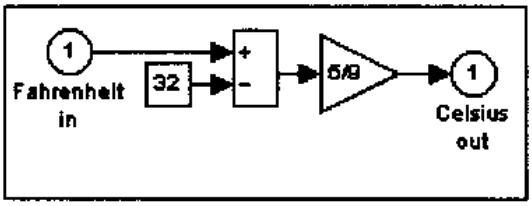


图 11.4 华氏温度转换为摄氏温度子模型

当加热器开始工作时，就计算其消耗的热量，并把所得的结果在 Heat Cost($)模块中显示出来，同时在内部温度 Indoor temp scope 模块中显示出来。

2. 仿真中的模型参数的修改

在仿真过程中，也可以修改模型中的参数来看系统的反映，例如：

- 打开 Floating Scope 模块。Floating Scope 模块可允许用户选择模型中的任何连线来检查该连线中的信号。如果要检查它的工作方式，可在仿真开始以后，选择不同的连线来观察它所显示的信号。
- 扩大 Scope 模块。Scope 模块含有显示区域和用户用来控制显示信号的控制区域。水平轴代表时间，垂直轴代表信号。
- 模型左上端标号为 Set Point 的模块用来设置内部温度的希望值。在仿真进行的同时把该模块打开，并把其值重新设置为 80℃，来观看内部温度和热损耗的变化情况。同样，也可改变外部温度来看它对仿真的影响。
- 打开标号为 Daily Temp Variation 模块来调节日常温度，看改变幅值以后，仿真所发生的变化。

3. 从中获得的一些启示

从上面这个演示程序中，可以得到以下一些有用的启示。

- 进行仿真需要包括两个步骤：一是定义参数，二是用 Start 命令启动仿真。
- 子系统(如 Thermostat、House、Temp Convert 等)可以把一组复杂的模块“隐藏”到一个子系统中。打开这些子系统模块，SIMULINK 就在一个新的窗口显示其所包含的基本模块。
- Scope 模块就像示波器一样，用图形方式显示输出。它显示的是驱动块的输出。

而悬浮的 Scope 模块与模型中的任何模块都不连，这样允许用户选择不同的连线来监视其中的信号，从而方便用户的使用。

- 模块中的参数可以由工作区里的变量来定义。这些变量通过 Load Data 模块，利用 eval 命令调用 thermdat.m 文件，把数据放在工作区里后，来获得它们的值。

11.1.3 创建一个简单的模型

这个例子显示用户如何利用命令和操作来创建自己的模型。在这一部分介绍的创建模型的命令是很简单的，下一节将详细介绍这些命令。从建立模型到仿真的步骤如下：

(1) 要创建这个模型，首先在 MATLAB 命令行窗口中输入 SIMULINK，这时弹出 SIMULINK Library Browser 窗口。在 SIMULINK Library Browser 工具条中选中“新建”按钮。这时 SIMULINK 就创建一个新的窗口，如图 11.5 所示。用户可以将窗口移到屏幕的右上角，以便能同时看到窗口中的内容和子模型库中的内容。

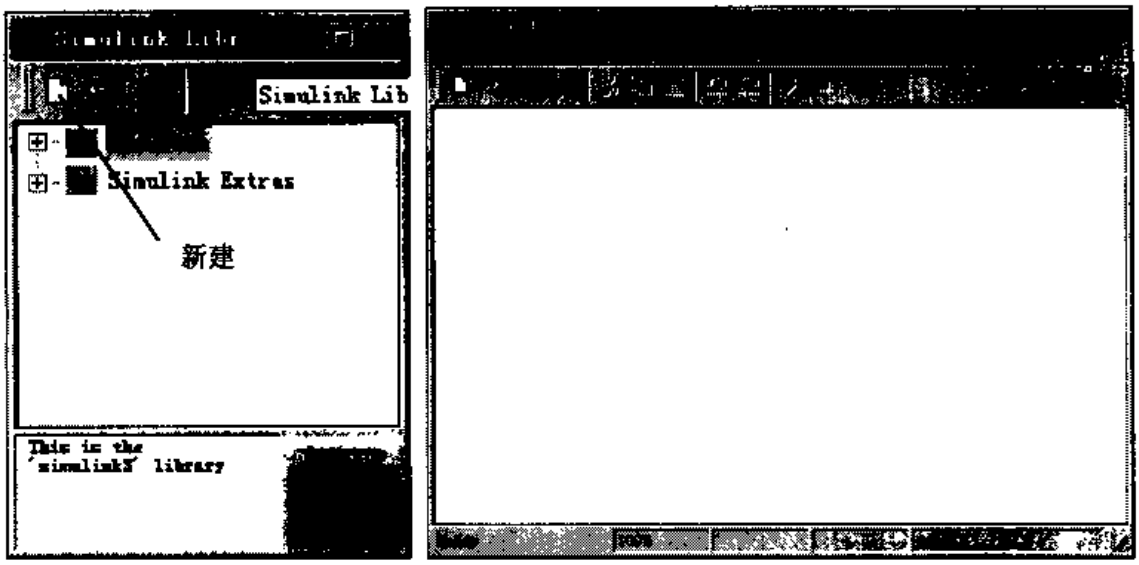


图 11.5　SIMULINK窗口

(2) 要建立这个模型，用户需要从以下这些SIMULINK模块库中复制模块到模型中去：源模块库(Sources Library)中的正弦波模块(Sine Wave Block)、显示模块库(Sinks Library)中的示波器模块(Scope Block)、连续模块库(Continuous Library)中的积分模块(Integrator Block)以及信号和系统模块库(Signals & Systems Library)中的整合模块(Mux Block)。

(3) 用户可以使用 Library Browser 从源模块库中复制正弦波模块(Sine Wave Block)。在 Library Browser 窗口中双击 SIMULINK 节点，展开 SIMULINK 节点下的子节点，可以看到有 Sources 节点。再次双击展开它，就可以看到正弦波发生器(Sine Wave Block)，单击并选中之。此时窗口如图 11.6 所示。

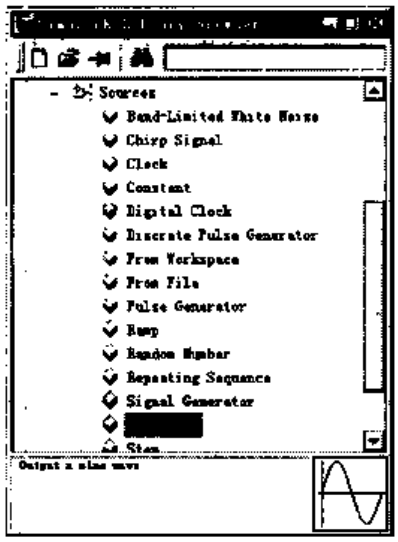


图 11.6 正弦波发生器模块

(4) 选中这个模块后，拖动鼠标把它移到自己的模型窗口中。当释放鼠标按键以后，SIMULINK就在用户的模型窗口中显示Sine Wave Block模块的图标。用同样的方法将其他模块从不同的模块库复制到模型窗口中。用户可以将模块从一个模型窗口拖动到另一个模型窗口。也可以选中这个模块，然后用方向键来短距离地移动模块。图11.7就是所有模块都放置好的情形。

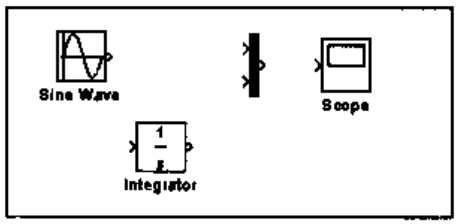


图 11.7 仿真模型1

(5) 下一步的工作是把不同的两个模块连接起来。为此，首先把鼠标指针定位在 Sine Wave 模块的输出端口并按下鼠标按键，然后拖动鼠标指针到 Mux 模块的输入端口并释放鼠标按键，SIMULINK 就在这两个模块之间画一条连线，连线中箭头的方向代表信号的流向。如图 11.8 所示，将所有的模块之间用线段连接起来。

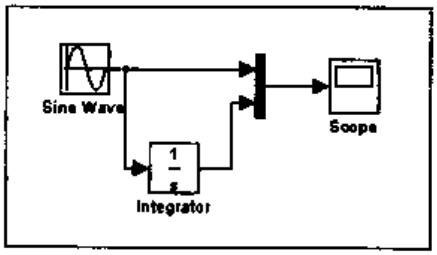


图 11.8 连接图

(6) 当用户建完模型并觉得满意之后，就可以开始仿真。打开示波器模块(Scope Block)显示仿真的输出，让 Scope 窗口打开着，接着设置 SIMULINK 让仿真持续 10 秒。首先，从 Simulation 菜单中选中 Parameters 命令。注意在弹出的对话框中，Stop Time(停止时间)应设置为 10 秒(这是默认值)，如图 10.9 所示。然后用鼠标单击 OK 按钮来关闭对话框。

这时就可通过 Simulation 菜单中的 Start 命令来启动仿真。在示波器上可以看到如图 11.10 所示的逼真的波形。

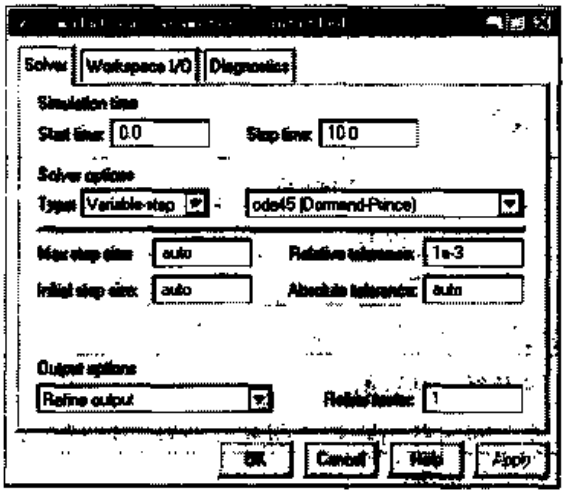


图 11.9　仿真参数设置

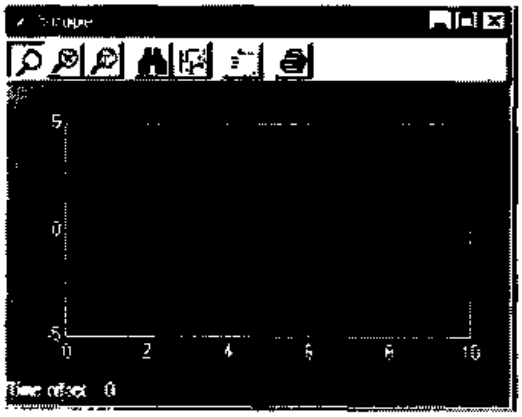


图 11.10　仿真示波器的输出

(7) 仿真终止的方式有两种：一种是自动的方式，即仿真时间和 Parameters 对话框中定义的时间相等时，就自动终止仿真；另一种是人工方式，即通过 Simulation 菜单中的 Stop 命令终止仿真。

11.1.4　SIMULINK 的界面和菜单

SIMULINK 工作窗口是 MATLAB 窗口系统中的一种新类型。下面分别简要介绍 SIMULINK 窗口中各菜单命令的含义，其中有些菜单的操作方法在后面的有关部分将陆续详细介绍。

1. File(文件)菜单

File 菜单包含一组标准的文件 I/O 命令和打印操作，如表 11.1 所示。

表 11.1　File菜单

命　令	快 捷 键	功能简介
New	Ctrl+N	创建新的 SIMULINK 模型窗口或模块库
Open	Ctrl+O	打开已经存在的 SIMULINK 模型文件
Close	Ctrl+W	关闭当前的 SIMULINK 窗口
Save	Ctrl+S	保存当前的 SIMULINK 模型文件
Save As		将文件另存
Mode Properties		设置描述模型的属性
Print	Ctrl+P	打印
Print Setup		打印设置
Exit MATLAB	Ctrl+Q	退出 MATLAB 系统

2. Edit(编辑)菜单

Edit 菜单除了包含大量的标准编辑操作外，还有很多 SIMULINK 独有的菜单操作，如子块操作和库模块操作等，如表 11.2 所示。

表 11.2 Edit菜单

命令	快捷键	功能简介
Undo	Ctrl+U	撤消上一步所做的操作
Redo	Ctrl+R	重做上一步所做的操作
Cut	Ctrl+X	剪切选定的内容
Copy	Ctrl+C	将选定的内容复制到剪贴板上
Paste	Ctrl+P	将剪贴板上的内容粘贴到当前光标所在的位置
Clear	Delete	清除所选定的内容
Select All	Ctrl+A	选择所有的内容
Copy Model		复制选定的模型
Signal Properties		设置信号线的属性
Block Properties		设置模块的属性
Create SubSystem	Ctrl+G	产生子系统
Mask SubSystem	Ctrl+M	封装子系统
Look Under Mask	Ctrl+U	查看封装子系统的详细内容
Go to Library Link	Ctrl+L	建立与模型库连接
Break Library Link		切断与模型库连接
Unlock Library		解锁模型库
Update Diagram	Ctrl+D	更新图像

3. View(查看)菜单

View 菜单包含一组定制 SIMULINK 窗口的命令和查看当前窗口模块的命令，如表 11.3 所示。

表 11.3 View菜单

命　令	备　注	工具简介
Toolbar		选择是否显示工具条
Status Bar		选择是否显示状态栏
Model Browser	含有子菜单	模型浏览，可从子菜单中选择所需的选项
Block Data Tips	含有子菜单	模块数据工具，可从子菜单中选择所需的选项
Show Library Browser		显示模块库浏览窗口
Zoom In		放大视图
Zoom Out		缩小视图
Fit System to View		固定系统以便观察
Normal(100%)		设置视图为一般状态(100%显示)

4. Simulation(仿真)菜单

SIMULATION 菜单包含了一组用来运行和设置仿真过程的命令，如表 11.4 所示。

表 11.4　Simulation菜单

命　令	快 捷 键	功能简介
Start	Ctrl+T	启动仿真
Stop		停止仿真
Parameters	Ctrl+E	仿真参数的设置

5. Format(格式)菜单

Format 菜单包含一组定制 SIMULINK 仿真系统显示格式和各个模块的显示格式的命令，如表 11.5 所示。

表 11.5　Format菜单

命令	快捷键/备注	功能简介
Font		设置字体的格式
Flip Name		翻转模块名称
Show/Hide Name		显示/隐藏模块名称
Filp Block	Ctrl+F	翻转模块
Rotate Block	Ctrl+R	旋转模块
Show/Hide Drap Shadow		显示/隐藏拖动阴影
Show/Hide Port Labels		显示/隐藏端口标志
Foreground Color	含有子菜单	设置前景颜色，在子菜单中选择
Background Color	含有子菜单	设置背景颜色，在子菜单中选择
Screen Color	含有子菜单	设置屏幕颜色，在子菜单中选择
Sample Time Color		设置不同的采样时间序列的颜色
Wide Vector Lines		加宽向量线条的宽度
Vector Lines Widths		设置向量线条的宽度
Port Data Types		设置端口数据的类型

6. Tools(工具)菜单

Tools 菜单包含了 MATLAB 为 SIMULINK 提供的附加工具包。目前有 Report Generator 工具包和 Realtime Work 工具包，是用来向用户提供数据报表和将仿真模型转换为实时运行的系统。而这个菜单的命令将取决于用户安装 MATLAB 时的选择，相信 MATLAB 公司还会继续提供新的工具包以及由第三方提供的工具包。

11.2 SIMULINK 模型的构造

为了掌握 SIMULINK，应学习如何操作以及如何建立模型，本节着重对构造模型的过程作详细地介绍。

SIMULINK 完全采用方框图的“抓取”功能来构造动态系统。系统的创建过程就是绘制方框图的过程。在 SIMULINK 的环境中方框图的绘制完全依赖于鼠标操作。鼠标的不同操作体现在光标的形状上。在默认状态下鼠标是一个箭头，但在图形的操作中鼠标呈现出不同的形态。

11.2.1 创建模型文件

如果用户要建立一个新的模型，可以像其他 Windows 98 程序一样移动和改变窗口的大小。可以在 MATLAB 命令行窗口中单击 File 菜单中的 New 子菜单的 Model 命令，这时 MATLAB 会打开一个新的 SIMULINK 窗口。或者在 MATLAB 命令行窗口中输入 SIMULINK，打开 SIMULINK 模型库窗口，然后再单击【新建】按钮，创建新的方框图窗口。

如果要编辑一个已经存在的模型框图，则有下面两种方法：

- 在 SIMULINK 的 File 菜单中单击 Open 命令，然后输入存放模型的文件名(其扩展名为 m)。
- 在 MATLAB 的命令行窗口中输入存放模型的文件名。

这样，SIMULINK 就会为用户建立一个新的模型窗口，并把模型在这个窗口中显示出来。

11.2.2 选择对象

在建模和编辑过程中，经常遇到选择一个或多个模块和连线(这些模块和连线统称为对象)的问题。下面具体介绍选择对象的方法。

1. 选择一个对象

要选择一个对象，只要用鼠标在对象上单击即可。这时就会发现被选中对象的角上有一个“把手”标记。例如，图 11.11 表示 Sine Wave 块和一条连线被选中。当在某个对象上用鼠标单击选中这个对象后，任何以前已被选中的对象就不再被选中。

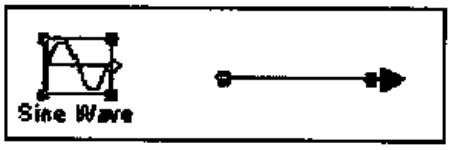


图 11.11 选择一个对象

2. 选择一个以上对象

用户可以选择一个以上对象，这有两种方法可以选择。一种方法是逐个选择法，另一种方法是用方框选择相邻的几个对象。下面分别介绍这两种方法。

- 逐个选择法

要选择一个以上对象，只要按住 Shift 键，然后在要选择的对象上用鼠标逐个单击，于是用鼠标单击过的对象就都被选中；反之，如果要放弃某个已被选中的对象，那么只需在按下 Shift 键的同时，在已被选中的对象上用鼠标单击，则这个对象就不再被选中。

- 用方框选择

选择一个以上的对象，一种比较简单的方法是在要选择的对象周围用方框框起来。要定义一个这样的方框，可采用下面的步骤：

(1) 首先定义方框的起始角，方法是把鼠标指针移到方框的一个起始角上，然后按下鼠标左键，如图 11.12 所示。

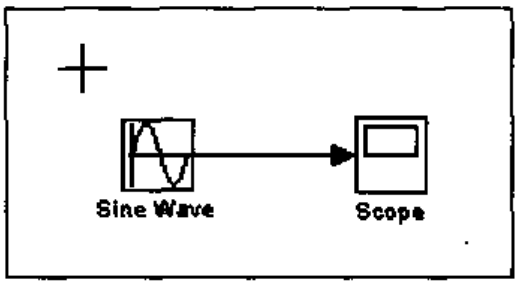


图 11.12 选择一个以上对象1

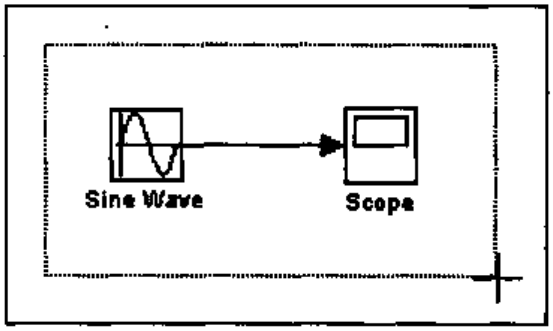


图 11.13 选择一个以上对象2

(2) 拖动鼠标，把鼠标指针移到方框起始角的对角上，如图 11.13 所示。

(3) 释放鼠标左键，于是在方框里面的所有模块和连线就都被选中，如图 11.14 所示。

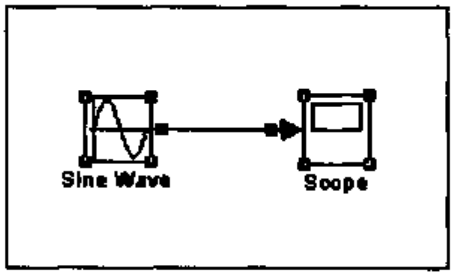


图 11.14 选择一个以上对象3

3. 选择整个模型

要想在模型的活动窗口中选择所有的对象，只要在模型窗口的 Edit 菜单中选择 Select All 命令或按快捷键 Ctrl+A 即可。

11.2.3 模块的操作

本节讨论建模中有关模块的各种操作问题。模块是仿真模型创建的基本元素。用户可以通过创建模块并且以适当的方式将其连接起来，模拟几乎所有的动态系统。

1. 有关模块数据的使用技巧

在 Windows 中，当用户将鼠标放置在 SIMULINK 窗口的模块上时，MATLAB 系统就会弹出一个窗口用来显示模块的信息。用户要使这项功能不起作用或者指定在信息窗口

中需要什么信息，可以从 SIMULNK 的 View 菜单中选取 Block Data Tips 命令，在其子菜单中可以选择(Enable)以及所要选取的信息。

2. 虚拟模块

当创建一个模型时，用户需要明白SIMULINK模块分为两种基本的类型，即非虚拟模块和虚拟模块。非虚拟模块在系统的仿真中起着积极的作用。如果用户加入或者移去一个非虚拟模块，那么用户就改变了一个模型的性质。与之相反，虚拟模块在仿真中没有起真正的作用。它们仅仅起着在图形上组织一个模块的作用。而一些SIMULINK模块可以在一些情况下起非虚拟模块的作用，也可以在另一些情况下起虚拟模块的作用。这样的模块被称为条件虚拟模块。表11.6列出了SIMULINK系统的虚拟和条件虚拟模块。

表 11.6 虚拟模块

模块名称	在什么条件下是虚拟模块
Bus Selector	总是虚拟的
Data Store Memory	总是虚拟的
Demux	总是虚拟的
Enable Port	总是虚拟的
From	总是虚拟的
Goto	总是虚拟的
Ground	总是虚拟的
Inport	除了模块在一个条件执行的子系统中并且直接和输出模块连接外，都是虚拟的
Mux	总是虚拟的
Outport	如果模块位于一个子系统模块内(无论是否是条件执行的)并且不是位于 SIMULINK 的根(顶层)窗口时，该模块是虚拟的
Selector	总是虚拟的
Subsystem	如果该模块不是条件执行的，则该模块是虚拟的
Terminator	总是虚拟的
Test Point	总是虚拟的
Trigger Port	如果输出端不是当前的，则该模块是虚拟的

3. 两个窗口之间的模块复制和移动

- 用鼠标复制模块

当用户建立自己的模型时，经常遇到这样的问题，即需要从 SIMULINK 库中或其他的模型窗口中把模块复制到自己的模型窗口中。要解决这样的问题，可以采用下面的步骤：

(1) 打开相应的库或源模型窗口。

(2) 把要复制的模块拖到自己的目标模型窗口中。方法是先用鼠标选中所要复制的对象，然后拖动鼠标，并把鼠标指针移到自己的模型窗口中，然后释放鼠标左键即可。

- 用菜单中的命令复制模块

用户也可以用 Edit 菜单中的 Copy 和 Paste 命令来复制模块，其方法如下:

(1) 选择要复制的模块。

(2) 在 Edit 菜单中选择 Copy 命令。

(3) 选择用户的模型窗口，使之成为活动窗口。

(4) 选择 Edit 菜单中的 Paste 命令。

这样用户所要复制的模块就出现在自己的模型窗口中。

- 模块的命名

SIMULINK 会给每一个复制过来的模块进行命名。如果在整个模型当中，用户复制过来的模块是这一类型模块中的第一块，那么它的名字和原来所在窗口中的名字是完全相同的。例如，如果用户把 Gain 模块从 Linear 库中复制到自己的模型窗口中，那么这一模块的名字就是 Gain。如果用户的模型中已经包含了一个 Gain 模块，那么 SIMULINK 就在模块名字的后面跟上一个按照顺序排列的数字(如 Gain1、Gain2 等)。用户也可以对模块进行重新命名，这将在后面详细论述。

- 复制模块之间的连线问题

如果用户要复制一个模块，而这个模块又和其他模块连接着，那么 SIMULINK 不会复制它们之间的连线。如果用户要复制与这个模块相连的那个模块，那么它们之间的连线也将一起复制过来。

当用户复制一个模块时，那么复制过来的模块将继承源模块中的所有参数值。

- 模块的移动

SIMULINK采用一个不可见的5个像素的网格来简化对象的移动。模型中的所有模块和网格中的一条线对齐。用户也可以采用上、下、左、右4个箭头键来慢速移动一个模块的位置。如果希望在模型窗口显示网格，可以在MATLAB命令行窗口输入以下的命令：

```
set_param('<model name>','showgrid','on')
```

如果要改变网格的大小，可用如下命令：

```
set_param('<model name>','gridspacing',<number of pixels>)
```

对于以上的任何一个命令，用户也可以选择模型，然后输入：

```
gcs instead of <model name>
```

用户也可以用 Copy，Cut 和 Paste 命令来复制或移动模块，这时只是它们的图形复制过来，而其参数并不进行复制。

要在一个窗口把一个以上的模块复制到另一个窗口，其方法和复制一个模块是类似的。唯一的区别是在选择模块时，要同时按下 Shift 键。

4. 在同一个模型中移动模块

在同一个模型窗口中把一个模块从一个地方移到另一个地方，其方法很简单。首先是选中该模块，然后拖动鼠标指针到一个用户所希望的位置，最后释放鼠标按键即可，SIMULINK 会自动重新布置连到该模块上的连线。

要移动一个以上的模块(包括它们之间的连线)，其方法如下：

(1) 选中需要移动的模块和连线。

如果用户用逐个选择法来选中一个以上的对象，在选中最后一个对象后，不要释放鼠标左键。否则，当用鼠标单击一个已被选中的对象时，将导致那个对象不在选中之列。

如果用户用定义方框的办法来选中一个以上的对象，只需用鼠标单击任何一个已选中的模块，那么在拖动该模块的同时，就等于拖动方框内的所有对象。注意不要用鼠标单击方框内的连线，否则，只有那条连线被选中。另外，如果在模型窗口中任何对象未曾占有的区域内单击，那么所有的对象就不再被选中。

(2) 拖动选中的模块和连线到希望的位置，然后释放鼠标左键。

5. 在同一个模型中复制模块

在同一个模型中复制模块的方法如下：

(1) 按下 Ctrl 键，然后用鼠标把要复制的模块选中。

(2) 用鼠标把要复制的模块移到所要复制的位置。

这时就在要复制的位置上把原来的模块复制了一份。

6. 定义模块中的参数

一个模块的某些功能是由模块中的参数来定义的，用户可以通过其对话框来定义这些参数值。只要用鼠标双击该模块把它打开，SIMULINK 就显示出该模块的对话框，并列出这些参数和它们的当前值。用户可以改变这些值或接受这些默认值。

用户还可以先选中该模块然后在 Edit 菜单中选择 Block Properites 命令，SIMULNK 会弹出一个图 11.15 所示的模块属性对话框，在此改变该模块的一些通用的属性。

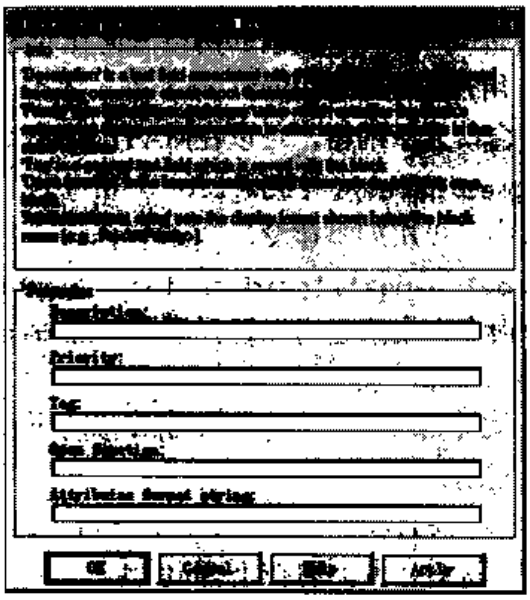


图 11.15　模块属性对话框

模块属性对话框包含如下的选项：

Description　简述模块的作用

Priority　在模型中，该模块相对于其他模块的运行优先权

Tag 在模块中存储的一般文字域

Open function　指定用户打开这个模块时调用的 MATLAB 函数

Attributes format string　　指定模块 AttributesFormatString 参数的值。该参数确定在模块的图标下显示什么参数。

属性格式字符串(Attributes Format String)可以是可植入参数名任何文本字符串。所谓的可植入的参数名是一个参数名前面是%<后面跟着是>的形式，例如%<priority>。SIMULINK 在模块图标的下面显示属性格式字符串，而将每一个参数名替换为相应的参数值。用户可以使用换行符(\n)在不同的行显示参数。例如指定属性格式字符串：

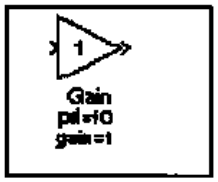


图 11.16　模块参数设置

pri=%<priority>\ngain=%<Gain>

对于 Gain 模块，在模型窗口中将显示如图 11.16 所示。

7. 删除模块

要删除一个或几个模块，首先必须选中需要删除的模块，然后按 Delete 键或者从 Edit 菜单中选择 Clear 或 Cut 命令。Cut 选项把要删除的模块移到了剪贴板上，而 Delete 键或 Clear 命令则不会影响剪贴板上的内容。

8. 断开模块间的连接

要使一个模块断开与另一个模块之间的连接，而不是删除它，方式是按下 Shift 键，然后选中该模块并从模型中的原始位置拖动该模块即可。

9. 改变模块的方向

SIMULINK 中默认，信号从模块的左边流进，从模块的右边流出，即输入在左边而输出在右边。用户也可以采用下面的方法改变模块的方向。

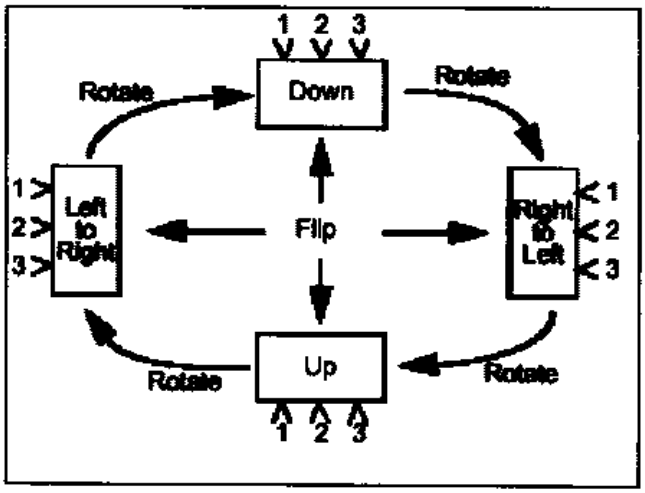


图 11.17　改变模块的方向

在 Format 菜单中选取 Flip Block 命令旋转模块 180°，或用 Rotate Block 命令顺时针使模块旋转 90°。图 11.17 显示在旋转后 SIMULINK 是如何组织模块端口的。

10. 重新定义模块的大小

要改变一个模块的大小，必须首先选中它，然后用鼠标拖动该模块的任何一个手柄即可。一个模块最小可定义为一个 5×5 的像素，而最大只受计算机屏幕的限制。当用户改变一个模块大小的时候，一个形状如箭头的标志表示所拖动的角和拖动的方向。当模块被重新定义大小时，一个长方形虚框表示其真正的大小。

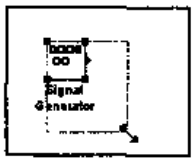


图 11.18 改变大小

例如，图 11.18 表示重新定义 Signal Generator 模块的大小。其含义是选中了右下角的手柄，且朝箭头的方向进行拖动，当鼠标按键释放后，长方形虚框的大小就是 Signal Generator 模块真正的大小。

11. 模块名的操作

在一个模型当中，所有的模块名必须是唯一的，而且必须至少含有一个字符。SIMULINK 默认，如果一个模块的端口在其右边的话，那么它的名字就在它的下面；如果一个模块的端口在其上边或下边的话，那么它的名字就在它的右边。用户也可以改变模块名和模块名的位置。

- 改变模块名

用户可以采用下面 3 种方法中的任何一种来改变模块的名字：选中显示名称的模块显示区，然后输入新的字符；把光标移到名字中间，然后输入新的字符；拖动鼠标，选中所要替换字符所在的区域，然后输入新字符。

当用户用鼠标在其他模块上单击或者进行其他的操作后，改变后的名字要么得到承认，要么被拒绝接受。如果用户试图改变一个模块名为一个已经被其他模块所用的名字，或者是一个没有字符的名字，SIMULINK 就显示一个出错信息。

用户可以修改一个模块名或几个模块名的字体。方法是首先选中这些模块，然后在 Format 菜单中选择 Fonts 子菜单，从这个子菜单中选择所需要的字体即可。

- 改变模块名的位置和隐藏

用户可以用以下两种方法改变选中的模块名称的位置：拖动模块名到模块现在位置的相对的一面；从 Format 菜单选择 Flip Name 命令，将模块名翻转 180° 。

另外用户可以从Format菜单中选择Hide Name命令来隐藏可见的模块名。当用户对一个模块选择了Hide Name命令后，当再次选中这个模块时，这个命令变成Show Name命令，这时可以利用这个命令来显示隐藏的模块名。

12. 模块的向量化

几乎所有的模块既能接受标量输入，也能接受向量输入，并允许用户定义标量或向量参数。用户可以通过 Format 菜单中的 Wide Vector Lines 命令来定义模型中的哪一条线传递的是向量信号，SIMULINK 就把传递向量信号的线画得比传递标量信号的线宽一些。若用户选择了上面的命令，并希望更新模型的显示，就必须在 Edit 菜单中选择 Update Diagram 命令来更新显示。另外，在仿真开始时也进行这样的更新显示。

13. 输入和参数的标量扩展

标量扩展是指把一个标量变成一个具有相同元素的向量。SIMULINK 只对一个模块的输入和参数进行标量扩展。具体哪些模块可以做扩展，用户可以在使用时注意模块的说明。

- 输入的标量扩展

当某个模块(如 Sum、Relational Operator 等)有一个以上的输入时，用户可以把向量输入和标量输入混合起来。在这种情况下，那个标量输入信号就要进行标量扩展，形成一个具有和向量输入信号维数一样的具有相同元素的向量。下面这个例子用来演示 Sum 模块的一个输入进行标量扩展后形成一个向量输入的情况。原来输入的标量 3 已经被扩展为向量[3,3,3]，同时请注意较粗的那些向量线(图 11.19)。

- 参数的标量扩展

对于可以进行标量扩展的那些模块，其参数既可以定义为标量，也可定义为向量。当定义为一个向量参数时，向量参数中的每一个元素与输入向量中的每一个元素相对应。而当定义为一个标量参数时，SIMULINK 就对标量参数进行标量扩展，自动形成一个具有相应维数的向量。

下面这个例子用来演示 Gain 模块的一个标量参数进行标量扩展后，形成一个与输入向量维数相同的具有三个元素的向量参数的情况(图 11.20)。

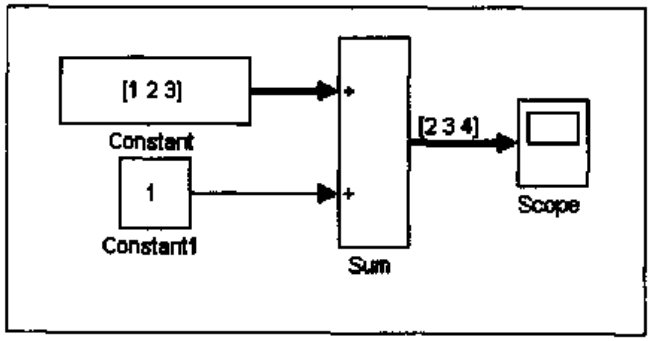


图 11.19　输入标量扩展

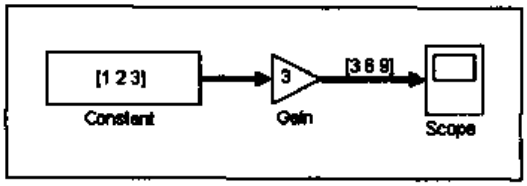


图 11.20　参数的标量扩展

11.2.4　连线的操作

连线既可以把一个模块的输出和另一个模块的输入连接起来，也可以把其他的连线与一个模块的输入连接起来。每一个输出端口都可以有任意条连线，而每一个输入端口却只能有一条连线。

提示： Mux 模块可把几个标量连线合并成一个向量连线，这个功能非常有用。

1. 模块间连线

- 连线操作

要把一个模块的输出和另一个模块的输入连接起来，可采用下面的步骤：

(1) 把鼠标指针移到第一个模块的输出端口。其实用户只需把鼠标指针移到靠近端口的任何位置，而没有必要把指针非常精确地移到那个端口位置上。

(2) 按下鼠标左键，这时鼠标指针就变成十字形，如图 11.21 所示。

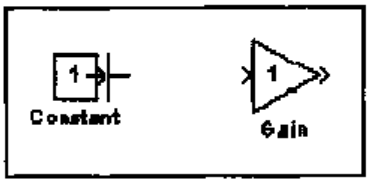


图 11.21 块间连线1

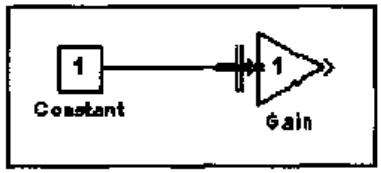


图 11.22 块间连线2

(3) 拖动鼠标指针到第二个模块(Gain)的输入端口。用户只需把鼠标指针移到靠近模块输入端口的任何位置。如果用户把鼠标指针移到了第二个模块的里边，则把连线连到第二个模块第一个未曾占有的输入端口。如果想把连线连到指定的端口，就必须在释放鼠标左键以前把鼠标指针定位到那个输入端口，如图 11.22 所示。

(4) 释放鼠标左键，SIMULINK 就用一个带箭头的连线代替端口的符号，用来表示信号的流向。用户既可以把连线从一个模块的输出端口连到另一个模块的输入端口，也可以反方向连接。在这两种情况下，箭头都出现在输入端口，而且信号的流向不会因为连线的方向不同而发生改变。

注意： SIMULINK 在布线时自动在模块的周围进行布线，而不是把连线从模块的中间穿过。不过，如果用户在画连线或线段的同时，按下了 Shift 键，那么 SIMULINK 就根据用户的要求进行连线，不再遵循上面的原则。

- 从其他连线画连线

用户可以在一条已经存在的连线上引出另一条连线，那么这两条连线就传送相同的信号给它们各自的对象。

例如，图 11.23 所示的两张框图中，左边的那张框图有一根单线，它把 Product 模块的输出端口连到 Scope 模块的输入端口。而右边的那张框图增加了一根连线，这根连线把 Product 模块的输出端口连到 To Workspace 模块的输入端口。事实上，Product 模块和 Scope 模块之间的连线与 Product 模块和 To Workspace 模块之间的连线上所流过的信号都是 Product 模块的输出信号。

要从某一根连线上引出另一根连线，可采用下面的方法：

(1) 把鼠标指针移到这根连线上的某个位置，这个位置就是引出新线的起始位置。

(2) 在按下 Ctrl 键的同时，按下鼠标左键。

(3) 拖动鼠标到目标端口，然后释放鼠标左键和 Ctrl 键，那么 SIMULINK 就在起始位置和目标端口之间创建了一条新线。

提示： 用户也可在按下鼠标左键的同时，按下鼠标右键，替代按下 Ctrl 键的功能来引出新线。

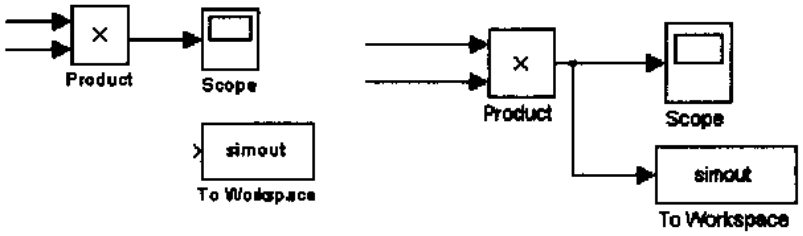


图 11.23　多重连线

- 画一根线段

画一根线段，就等价于画一根连线，而这根连线不连到框图中的任何一个对象上。在画完一根线段后，就联机段的末端出现一个箭头，以表示该线段还没有终止。如果要在刚才的线段后面接着画线段，只要把鼠标指针移到箭头上面并按下鼠标左键，其余的就是重复上面的步骤。如果模型中有任何未曾连完的线段，仿真时 SIMULINK 就给出警告信息。

- 连线的角度

除了下面这些情况以外，SIMULINK 总是以 45° 的倍数的角度来画连线。

- ◆ 如果鼠标指针移到了一个可用端口的附近，那么那根连线就连到该端口。
- ◆ 如果在创建新线的同时，按下了 Shift 键，SIMULINK 就根据用户的要求进行画线。
- ◆ 如果用户是通过移动鼠标指针到某个模块上并且释放鼠标来创建新线，那么这根连线就连到该模块最上面或最左边未曾占用的端口上。

2. 删除连线

要删除一根或几根连线，首先选中它们，然后按下 Delete 键或者在 Edit 菜单中选择 Clear 或 Cut 命令即可。

3. 移动线段

要移动一根线段，可采用下面的步骤：

(1) 把鼠标指针移到线段上面，如图 11.24 所示。

(2) 按下鼠标左键，这时鼠标指针变成十字形，如图 11.25 所示。

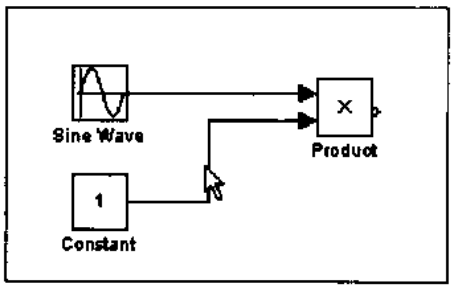


图 11.24　移动线段步骤1

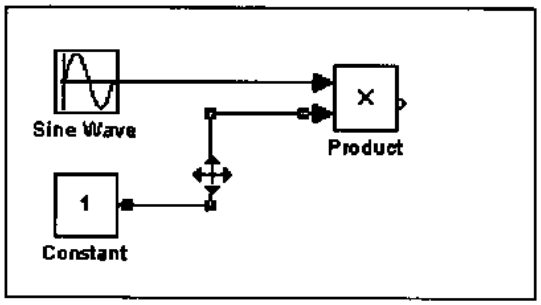


图 11.25 移动线段步骤2

(3) 拖动鼠标指针到用户所希望的位置，如图 11.26 所示。

(4) 释放鼠标左键，如图 11.27 所示。

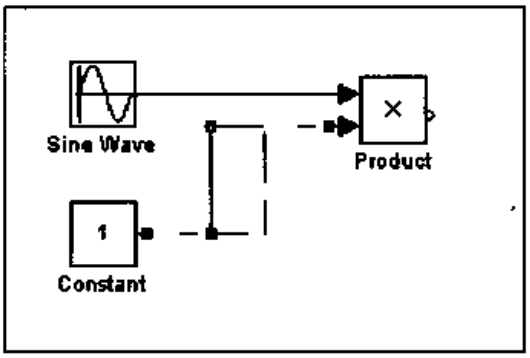


图 11.26 移动线段步骤3

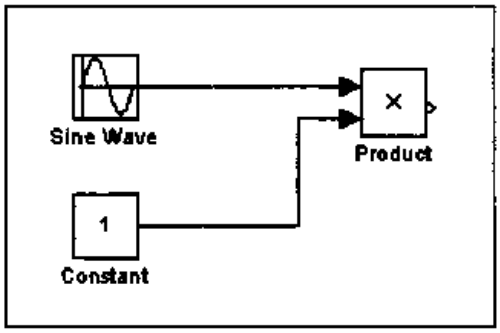


图 11.27 移动线段步骤4

注意： 用户不能移动直接连到某个模块端口上的线段。

4. 移动顶点

要移动一根连线的顶点，可采用如下的步骤：

(1) 首先把鼠标指针移到该顶点上面，然后按下鼠标左键，这时鼠标变成圆圈，如图 11.28 所示。

(2) 拖动鼠标指针到用户所希望的位置，最后释放鼠标左键即可，如图 11.29 所示。

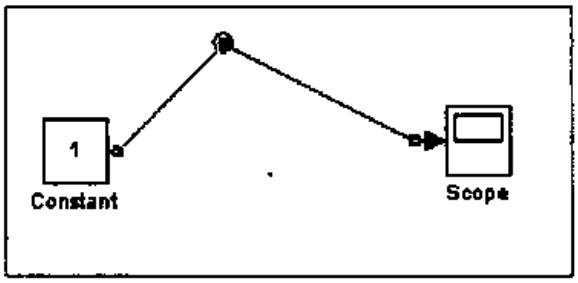


图 11.28 移动顶点步骤1

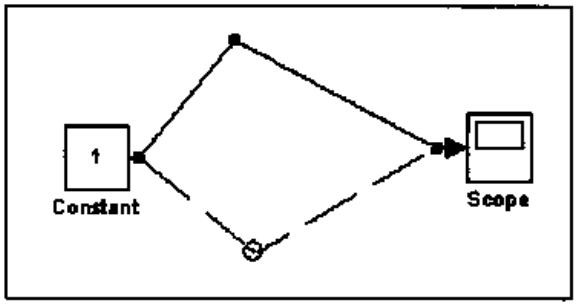


图 11.29 移动顶点步骤2

注意： 用户不能移动一根连线上两个端点处的这些顶点。

5. 把一根连线分割成线段

用户可以把一根连线(或者一根线段)分成两根线段，而这根线段的两个端点仍放在原来的位置。SIMULINK 就会自动创建这些线段和连接这些线段的顶点。要把一根连线分割成线段，可采用以下的步骤：

(1) 把鼠标指针移到连线上的某一点，且这一点是用户所需要的点。

(2) 按下 Shift 键的同时，按下鼠标左键。

(3) 拖动鼠标指针到用户所希望的位置。

(4) 释放鼠标左键和 Shift 键。

11.2.5　给模型框图添加文本注释

1.　添加文本注释

注释可以提供模型的文本信息。用户可以在模型的框图中添加文本注释，方法是：

(1) 把鼠标指针移到需要添加文本注释的位置。

(2) 单击鼠标左键。

(3) 输入注释文本。

(4) 再单击鼠标左键。

SIMULINK 把文本注释放在用户单击时鼠标指针所在位置略微偏下一点的地方。

注意： 文本注释在模型中必须是唯一的。

2.　修改文本注释的字体

用户也可以修改文本注释的字体，方法是：

(1) 用定义方框的办法来选中需要修改的文本注释。

(2) 在 Style 菜单中选择 Fonts 子菜单。

(3) 在 Fonts 子菜单中选择所需要的字体。

11.2.6　创建子系统

1.　模块的分组

随着系统规模和复杂性的增加，模型也在不断增大。为了使问题得到简化，可以对各个模块进行分组，然后把各组分别组成一个子系统来使复杂问题简单化。对各个模块进行分组的好处主要有以下几点：有助于在模型窗口中减少模块的个数；有助于把功能有关的模块编入同一组；有助于建立分层结构的框图。

用户有两种方法建立子系统：在模型窗口中添加一个 Subsystem 子块，然后把该模块包含的模块添加进去即可；如果组成一个子系统的模块已经添加进去了，那么可把这些模块归入到同一个子系统中。

2.　通过加入子系统模块来创建子系统

如果在一个子系统所包含的模块中还未加入要创建的子系统，首先必须添加一个 Subsystem 模块，然后再创建组成子系统的各个模块。具体方法如下：

(1) 从 Connections 库中把 Subsystem 模块复制到用户模型窗口中。

(2) 双击 Subsystem 模块，使之打开。

(3) 在空白的子系统窗口创建子系统。用 Import 模块表示从子系统外面输入，用 Output 模块表示子系统的输出。例如，下面框图中的 Sum 模块是子系统中唯一的一个模块，In1

和 In2 表示子系统的两个输入端口，Out1 表示子系统的输出端口(图 11.30)。

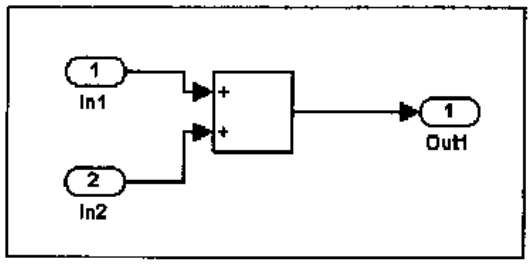


图 11.30　子系统的创建

3. 模块转变为子系统的方法

如果用户创建完了一些模块，又想把这些模块变成一个子系统，那么可采用下面的步骤：

(1) 用定义一个方框的办法选中用户需要变成一个子系统的模块和连线，注意不能用逐个选中它们的办法来对这些模块创建子系统。例如，图 11.31 这个模型代表一个加法器，其中在方框中的 Sum 模块和 Unit　Delay 模块被选中。

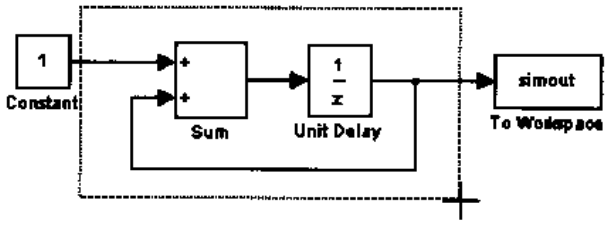


图 11.31　子系统的创建1

当释放鼠标左键的时候，这两个模块和它们之间的所有连线都被选中。

(2) 在 Edit 菜单中选择 Create Subsystem 命令，SMULINK 就用一个 Subsystem 模块来代替上面这两个模块和它们之间的连线。图 11.32 是在选择了 Group 命令之后所显示的模型。

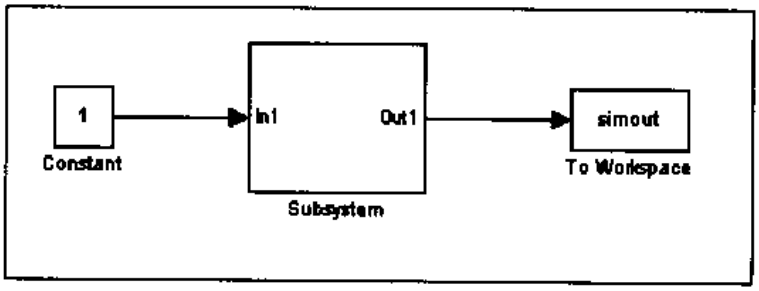


图 11.32　子系统创建2

和所有其他的模块一样，Subsystem 模块的名字也可以改变。同样，也可以用 SIMULINK 的封装功能来定做其图标和对话框，这方面详细内容见 11.4 节。

如果用鼠标双击 Subsystem 模块，SIMULINK 就显示其基本模型。图 11.33 就是在打开前面那个模型中的 Subsystem 模块后所得的结果。

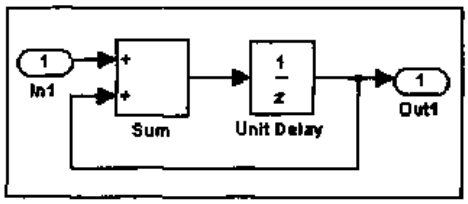


图 11.33　子系统下的模块

11.2.7　建模技巧

在建立模块中如果能注意以下这些问题，可以使得建立的模块更好。所谓更好，就是有良好的可读性和可扩充性。这些都是用户所建立的模型是否可以得到更广泛运用的关键。以下就是一些对建立模型有用的提示。

- 内存问题

一般来说，内存越大SIMULINK的表现越好。

- 使用分层模式

对于复杂的模型，最好采用分层次的子系统建模。成组的模块可以简化顶层模块的结构，使其他用户更容易理解模型。模型浏览器给复杂的模型提供了大量的有用信息。

- 使模型更加清楚

组织良好并有适当文档的模型具有良好的可读性和可理解性。在模型中使用信号标识和模块的注释可以帮助用户描述在模型中到底发生了什么。

- 建模策略

如果用户的一些模型要使用一些相同的模块，那么最好把这些模块保存在一个模型中。那么当用户新建一个模型的时候，就可以打开这个模型然后从其中复制公共模块。用户还可以创建一个模块库，把所需的模块复制进去，然后再保存这个系统。以后用户就可以在MATLAB命令行窗口输入系统的名称来进入模块。

一般来说，要建立一个模型，首先在纸上设计，然后用计算机创建模型。把所有的模块都放置到模型窗口中，再用信号线连接它们。这样，用户就可以减少打开模型库的次数。

11.2.8　模拟方程

对于一个 SIMULINK 的新用户来说，最费解的问题之一是如何来模拟方程。下面一些例子将帮助用户理解这个问题。

1.　把摄氏温度转变为华氏温度

首先，来模拟一个将摄氏温度转变为华氏温度的方程：$T_f=9/5(T_c)+32$。

- 确定建模所需的模块

在进行模拟以前，首先要确定建立上面这个模型所需要的模块：

- 一个 Gain 模块，用来定义常数增益 9/5，Gain 模块来源于 Linear Library。
- 一个 Constant 模块，用来定义一个常数 32，Constant 模块来源于 Source Library。
- 一个 Sum 模块，用来把两项相加，Sum 模块来源于 Linear Library。

- 一个 Ramp 模块，用来作为输入信号，Ramp 模块来源于 Source Library。
- 一个 Scope 模块，用来显示输出，Scope 模块来源于 Sinks Library。

● 模块的复制

把上面这些模块从各自相应的库中复制到用户的模型窗口中，如图 11.34 所示。

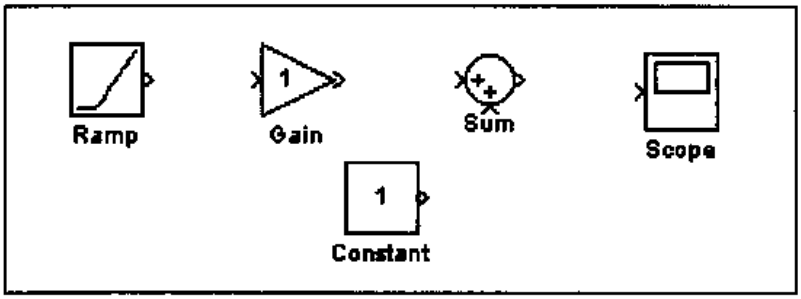


图 11.34 模拟方程模型1

分别打开 Gain 模块和 Constant 模块(方法是用鼠标在它们上面双击)，系统弹出对话框。用户可以在对话框中设置模块的属性值，这里分别把 Gain 模块的增益值设为 9 / 5 和将 Constant 模块的常数值设为 32，然后，单击 OK 按钮。打开 Ramp 模块，把其初始输出参数(Initial output)设置为 0。

● 模块的连接

按上节所示的方法，把各个模块连接起来，得到图 11.35 所示的框图。

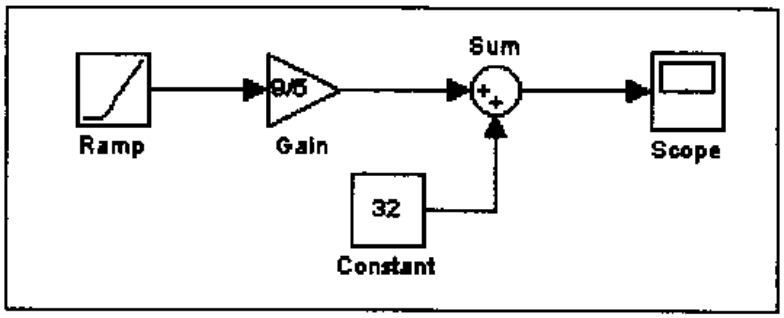


图 11.35 模拟方程模型2

Ramp 模块代表摄氏温度，Gain 模块输出为 9/5，这个值在 Sum 模块中和 Constant 模块中的常数 32 相加后输出，这个输出就是华氏温度。打开 Scope 模块就可以观看这个输出值的变化曲线。其中，Scope 模块的 x 轴设为比较小的时间(如 10s)，而把 y 轴设置得比输出幅值略微大一些，以便能看到整个曲线，在此可选为 60。

● 在用户窗口的菜单中选择

在用户窗口的 Simulation 菜单中选择 Parameters 菜单命令，定义 Stop time 为 10s，Maximum step size 为 0.1s。然后在 Simulation 菜单中选择 Start 命令，仿真就开始进行了。这时在 Scope 中就可以看到输出的曲线图。

2. 模拟一个简单的连续系统

下面再来模拟一个微分方程：

$$\dot{x}(t) = -2x(t) + u(t)$$

在这个模型中，需要对 $\dot{x}$ 进行积分并产生输出，因此需要一个 Integrator 模块。另外需要 Gain 模块和 Sum 模块。为了产生一个输入信号 u，可采用 Signal Generator 模块来产生方波。除此之外，还需要一个 Scope 模块来观察模型的输出。把这些模块按前面的所述的方法从各自的库中复制到模型窗口中，并用连线连接起来，再把 Gain 模块的增益值设为－2，结果如图 11.36 所示。

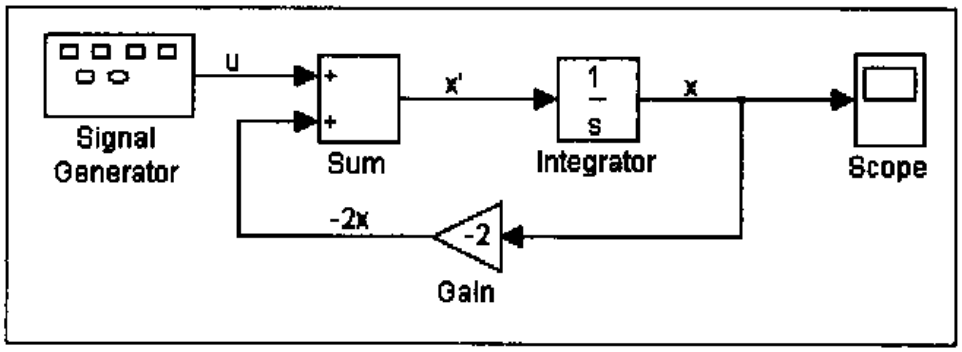


图 11.36　简单连续模型1

在这个模型中，可采用 Format 菜单中的 Flip Horizontal 命令来改变 Gain 模块的方向。另外，把连线从 Integrator 模块的输出端口连到 Gain 模块的输入端口时，在画线的同时要按下 Ctrl 键。

在这个模型中，最重要的概念是包括 Sum 模块、Integrator 模块和 Gain 模块的那个闭环回路。在这个环路中，x 是 Integrator 模块的输出，同时 $\dot{x}$ 是 Integrator 模块的输入且依赖于 x。因此，只有构成闭环回路，才能算出 x 的值。因此 x 和 $\dot{x}$ 之间的这种关系必须构成一个闭环才能实现。在仿真的过程中，每计算一步，Scope 块就显示一步 x 的值。如果仿真时间设为 10 s，Scope 块的 y 轴范围为－0.5 到 0.5，那么其输出如图 11.37 所示。

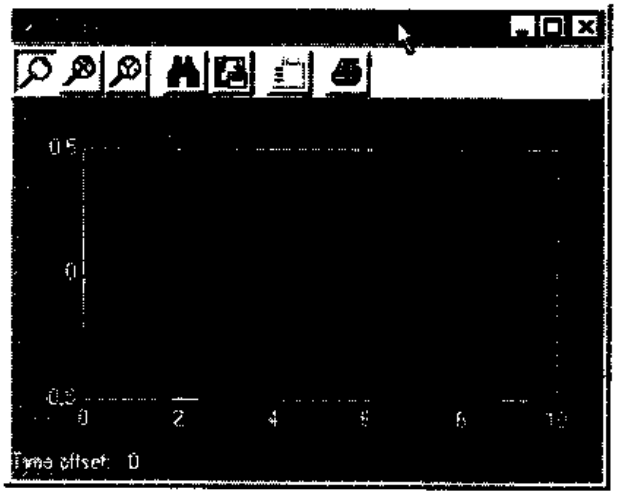


图 11.37　仿真结果输出显示

在这个例子中，所模拟的方程也可以用传递函数的形式来表示，模拟仿真要采用

Transfer Fcn 模块，u 作为该块的输入，x 作为该块的输出。因为 Transfer Fcn 实现的系统为 x/u，因此对该微分方程进行拉氏变换，可得

$$sx=-2x+u$$

由此可得

$$x=u/(s+2)$$

因此

$$x/u=1/(s+2)$$

即传递函数为1/(S+2)。传递函数模块使用参数来确定分子和分母的系数。在这个例子中，分子为1，分母为(S+2)。指定这两个值要用向量来代表s的连续递减次方的系数值。本例中分子用[1](或直接是1)来表示，分母用[1,2]来表示。 这样模型可以变成图11.38所示。

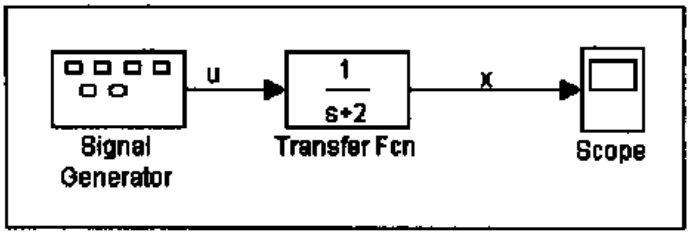


图 11.38 简单连续模型2

仿真的结果和前面的模型完全相同。

11.2.9 保存模型

用户可以在模型窗口的 File 菜单中选择 Save 命令或 Save As 命令来保存模型。保存模型窗口见图 11.39。模型保存后，SIMULINK 就生成一个特殊类型的文件，称为模型文件(model file)，以.mdl 为后缀，这个文件包括模块的图示和模块的属性。

如果用户第一次保存模型，使用 Save 命令提供文件名和保存的路径。模型的名称必须以字母开始并且不能超过 31 个字母、数字和下划线。如果用户的文件已经保存过，使用 Save 命令替换原有的文件或者用 Save As 命令以新的名称和路径保存文件。

SIMULINK 系统采用以下的步骤来保存一个模型：

(1) 如果.mdl 文件已经存在，系统将其重命名为一个临时文件。

(2) SIMULINK 系统执行所有模块的 PreSaveFcn 回调函数过程，然后执行模块图示的 PreSaveFcn 回调函数过程。

(3) SUMULINK 系统向模型文件写入新的文件并且加上扩张名.mdl。

(4) SIMULINK 系统执行所有模块的 PostSaveFcn 回调函数过程，然后执行模块图示的 PostSaveFcn 回调函数过程。

(5) SIMULINK 系统删除临时文件。

如果在以上过程中产生了任何错误，SIMULINK 系统将临时文件写回原来的文件名，将现在的文件写入一个以.err 为后缀的文件并且将错误信息写入文件中。

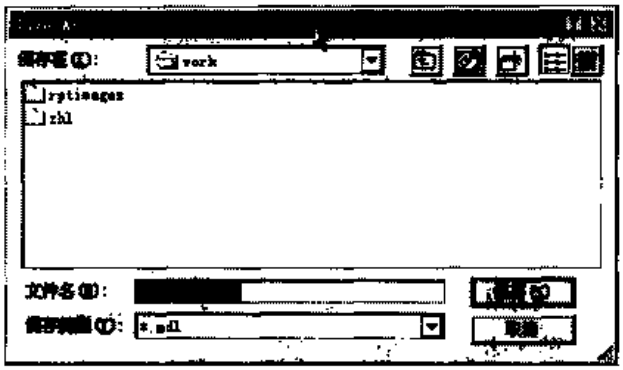


图 11.39　保存模型的窗口

11.2.10　打印框图

SIMULINK 有两种方法可以打印框图，一种方法是在 File 菜单中选择 Print 命令；另一种方法是在 MATLAB 的命令窗口输入 Print 命令。

在File菜单中选择Print命令，系统将会弹出一个打印对话框，如图11.40所示，使得用户可以在模型中选择性地打印系统。用户可以有如下的选择：

- 仅仅打印当前选择的系统。
- 打印现在的系统和所有模型层次上高于它的系统。
- 打印现在的系统所有模型层次上低于它的系统，可以通过选项确定所需封装模块和库模块的具体内容。
- 打印模型中的所有系统，可以通过选项确定所需封装模块和库模块的具体内容。
- 打印每一个框图中的重叠框架。

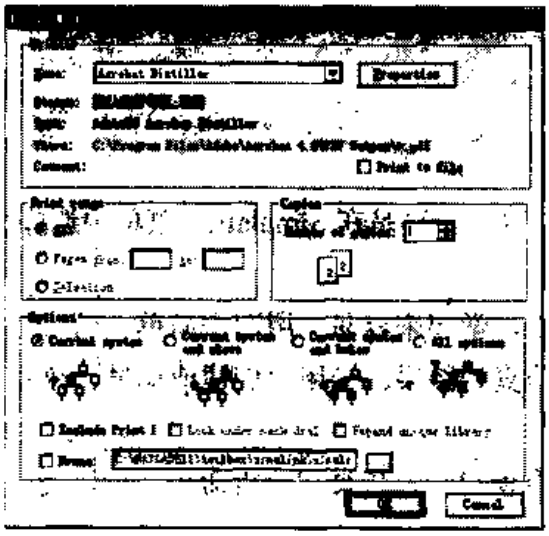


图 11.40　打印模型对话框

在以上这个对话框中的选项决定只有当前的系统才被打印。

在 MATLAB 的命令窗口用 Print 命令打印指定的文件到打印机或文件中。这个命令不打印任何打开的 Scope 模块。Print 命令的格式为：

```
print -smodel -device filename
```

其中这些参数的含义为：

- smodel 要打印的文件名字。如果省略，则打印当前系统，这时系统必须打开。
- device Windows 设备，包括：

win	当前安装的打印机，以单色打印。
winc	当前安装的打印机，以彩色打印。
meta	剪贴板，以 M 文件形式打印。
bitmap	剪贴板，以位图形式打印。
setup	打开 Print setup 对话框，但不打印。

- filename 保存输出的文件名。如果指定文件名，则把输出写到指定的文件中。如果文件已经存在了，则改写这个文件。如果指定的文件不包括扩展名，则添加一个适当的扩展名。如果直接把文件打印输出到打印机上，那么就不能控制其大小。如果要控制打印输出的大小，最好把打印输出先送到一个位图中，然后用 Word 程序来改变其大小。

11.3 仿真和结果分析

本节主要讨论模型的仿真与结果的分析这两个部分，包括以下一些内容：仿真分析、模型的线性化和模型平衡点确定。

在“仿真”这一小节将讨论如何从 Simulation 窗口菜单中和 MATLAB 命令窗口定义仿真参数和启动一个仿真的问题，同时还讨论几种不同的积分方法。在其后的两小节将讨论 SIMULINK 提供的用于线性化的工具和平衡点的确定问题。

11.3.1 仿真

启动一个仿真有两种方法，一种是在 Simulation 菜单中选择命令；一种是在 MATLAB 命令窗口输入命令。

- 在菜单中选择命令，学起来容易，用起来简单，而且可以很快得到结果，用户也可以用屏幕观察仿真的结果。当用户创建或调试一个系统时，用这种方法比较方便。
- 在 MATLAB 命令窗口输入仿真和分析命令，可以使用户便于观察改变模块或者积分参数后变化的结果。

上面这两种方法在模型的不同开发阶段具有不同的用途。以下分别介绍这两种仿真方法的使用特点。

1. 用菜单进行仿真

- 用菜单进行仿真的基本过程

这一部分讨论如何使用SIMULINK菜单命令和SIMULINK参数对话框来进行仿真。

用户可以从 Simulation 菜单中选择 Parameters 命令，SIMULINK 系统将会弹出 Simulation Parameters 对话框，用户可以通过 3 个选项卡来管理仿真的参数。

- Solver 选项卡允许用户设置开始和结束的时间，选择积分方法和确定积分方法的参数以及选择一些输出的选项。
- Workspace I/O 选项卡管理向 MATLAB 工作区的输入和输出。
- Diagnostics 选项卡允许用户在仿真时选择警告信息的水平。

用户可以使用有效的MATLAB表达式来指定参数。表达式由常量、工作区的变量名、MATLAB函数和数学运算符组成。

当定义完了所有的仿真参数后，就可以进行仿真。这可以从 Simulation 菜单中选择 Start 命令。要启动一个仿真，也可通过键盘上的热键 Ctrl+T 来实现。当仿真开始以后，Start 命令就变成了 Stop 命令。

如果模型中有 Scope 模块，要显示其输出的结果，首先必须把它们打开。而对于 Auto Scale Graph Scope，Graph Scope 或 X-Y Graph Scope 模块，SIMULINK 系统会自动把它们打开。

如果要终止一个仿真，可在 Simulation 菜单中选择 Stop 命令即可，也可采用键盘上的热键 Ctrl+T 来实现。

如果需要把一个仿真暂停运行，可在 Simulation 菜单中选择 Pause 命令。如果要把一个暂停运行的仿真继续进行下去，可在 Simulation 菜单中选择 Continue 命令。

注意： *SIMULINK 新用户常犯的一个错误是当 SIMULINK Library Browser 窗口还是活动窗口时就开始仿真。一定要在仿真前，确定仿真窗口是活动窗口。*

- 仿真诊断对话框(Simulation Diagnostics)

当在仿真的过程中发生错误时，SIMULINK系统停止仿真并且在仿真诊断对话框(Simulation Diagnostics)中显示错误信息，参见图11.41。

对话框有2个窗格，上面的窗格由以下的行组成，各行显示每一个错误的相关信息。

Message　显示信息类型，例如块错误、警告信息和日志等

Source　导致错误的模型元素名称(例如一个模块)

Fullpath　导致错误的元素路径

Summary　简要的错误信息

Reported by　报告错误信息的组件(例如SIMULINK、Stateflow和Real-Time Workshop等)

下面的窗格包含上窗格中第一个错误信息的全文。用户可以在上窗格中选择所需的错误信息，就可以在下窗格中看到错误信息的全文。

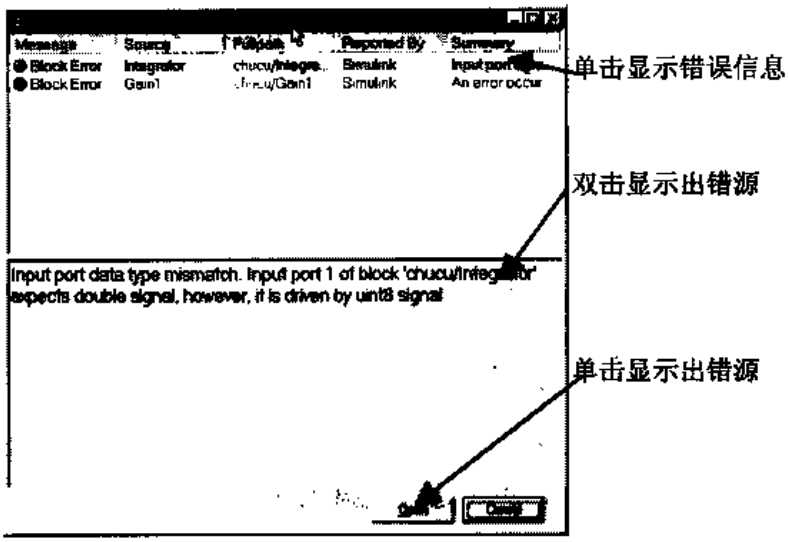


图 11.41 仿真诊断对话框

另外如果需要，SIMULINK系统会在弹出仿真诊断对话框的同时，打开一个包括出错源并把其显示为高亮的方框图（图11.42）。用户也可以双击诊断对话框的下窗格中的错误信息来显示出错源或者单击窗口中的Open按钮。

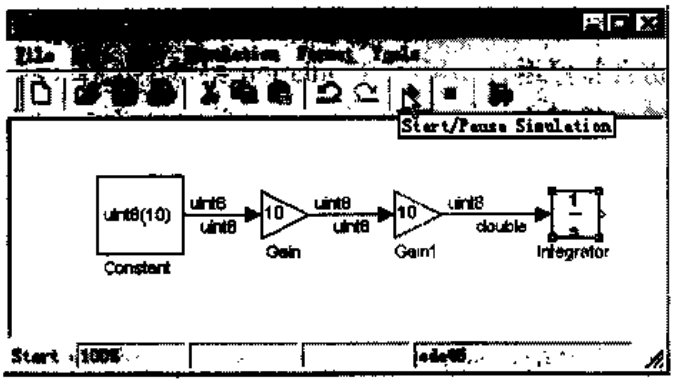


图 11.42 出错源显示方框图

2. 仿真参数设置对话框(The Simulation Parameters Dialog Box)

下面分别介绍如何在仿真参数对话框中的 3 个选项卡中设置仿真的参数。

- 设置积分方法(Solver)选项卡(图 11.43)

Solver 选项卡允许用户做如下的操作：设置仿真的开始和结束时间；选择仿真的积分方法并指定起始参数；选择输出选项。

◆ 设置开始时间和结束时间

参数 Start Time 和 Stop Time 定义了仿真开始的时间和仿真结束的时间。仿真时间和仿真所用的时间是两个不同的概念。例如要进行一个 10s 的仿真，通常计算机运行的时间不为 10s。实际上，运行一个仿真所需的时间取决于很多因素，包括模型的复杂性、最大和最小步长、计算机时钟的频率等。

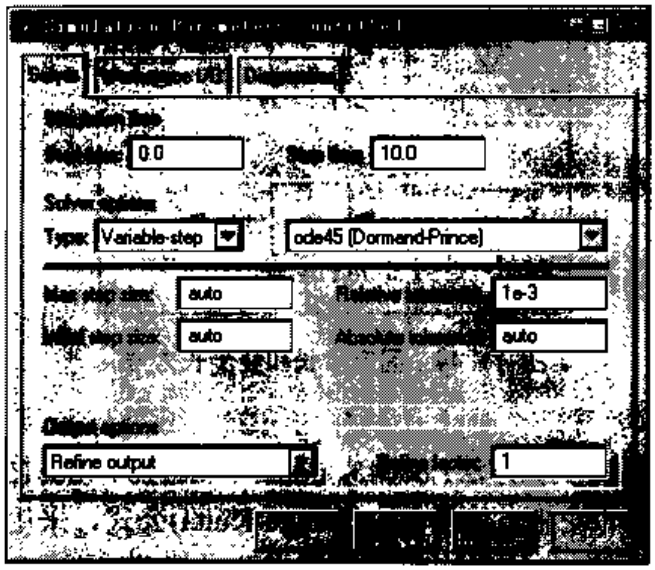


图 11.43　仿真参数设置Solver选项卡

◆　选择仿真中使用的积分方法

SIMULINK 模型的仿真涉及到一组常微分方程(ODEs)的数值积分。SIMULINK 为这些模型的仿真提供了一组积分方法。由于系统动态特性的多样性，因此没有一种方法能够精确而有效地适用于各种模型的仿真。要获得快速、准确的仿真结果，就必须精心选择适当的方法和适当的仿真参数。

对于不同的模型和不同的条件，仿真的性能(如仿真的速度、精度等)是千变万化的。因此，在选择仿真方法时，必须时刻牢记这些差异。

用户可以选择变步长和固定步长积分方法。变步长方法可以在仿真的过程中改变其步长，它们有误差控制和零穿越探测功能。固定步长方法在仿真过程中采用相同的步长，而且不具备变步长的两项功能。如果用户没有选择积分方法，SIMULINK 会根据用户模型的状态来选择步长，以下是其选择的原则：

如果模型具有连续的状态，使用 ode45。ode45 是一种优越的用于一般目的的方法。然而，如果用户知道系统是困难的或者 ode45 无法提供可接受的结果，最好改为 ode15s 方法。

如果模型不具有连续状态，SIMULINK 使用变步长的 discrete 方法并且显示信息指出没有使用 ode45 方法。SIMULINK 同时也提供一个使用固定步长的 discrete 方法。以下这个模块说明了这两种 discrete 方法的区别。

由于系统有 2 个采样时间 0.5s 和 0.75s，因此系统的基本采样时间是 0.25s。而变步长和固定步长这两种方法的区别在于它们产生的时间向量不一样，如图 11.44 所示。

固定步长方法产生如下的时间变量(以 0.25s 为采样步长)：

```
[0.0 0.25 0.5 0.75 1.0 1.25 ...]
```

变步长方法产生如下的时间变量(采用尽可能少的步骤)：

```
[0.0 0.5 0.75 1.0 1.5 2.0 2.25 ...]
```

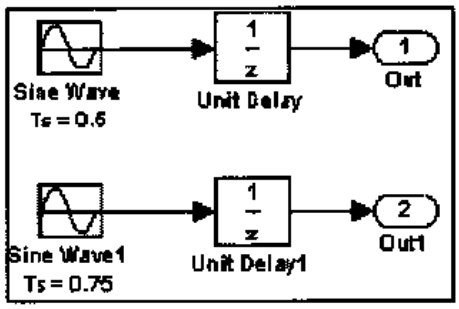


图 11.44 固定步长和变步长方法

◆ 变步长的方法

用户可以选择如下的变步长方法：ode45、ode23、ode113、 ode15s、 ode23s 和 discrete。

ode45 方法基于直接的龙格—库塔(4,5)(Runge-Kutta)公式，采用 Dormand-Prince 对，这是个一步的方法。在计算 y(t,n)时，该方法只需要紧接着前面的时间点 y(t,n－1)。一般来说，ode45 是解大多数一般问题的最佳选择。

ode23 方法也是基于直接的龙格—库塔(2,3)(Runge-Kutta)公式，采用 Bogacki 和 Shampine 对。在解容差比较大和存在很大困难时更有效。ode23 也是一步方法。

ode113 方法是可变阶的 Adams-Bashforth-Mouton PECE 方法。在解容差比较小的问题时更有效。ode113 是一种多步方法。

ode15s 方法是基于数值微分公式的可变阶公式(NDFs)。它比向后微分公式(BDFs 又称 Gear 方法)更加有效。ode15s 方法也是一种多步方法。如果用户怀疑自己的系统是困难的或者 ode45 方法失败了或者效率很低，试试 ode15s 方法。

ode23s 方法基于经修改的二阶 Rosenbrock 公式。由于它是一步方法，在解一些容差比较大的问题更有效。也可以解一些 ode15s 方法无效的困难问题。

ode23t 方法使用自由内插值实现了梯形积分法。如果问题难度中等并且用户需要一个没有数值跳跃的结果时可使用该方法。

ode23tb 方法使用 TR-BDF2 积分法，一个内置的 Runge-Kutta 公式。

dicrete(变步长) 是系统不具有连续状态时 SIMULINK 系统选择的方法。

◆ 固定步长方法

用户可以选择如下的固定步长方法：ode5、ode4、 ode3、ode2、 ode1 和 discrete。

ode5 方法是固定步长的 ode45 方法。

ode4 方法是 4 阶龙格—库塔(Runge-Kutta)公式(RK4)。

ode3 方法是固定步长的 ode23 方法。

ode2 方法是 Heun's 解法，也称为改进的欧拉(Euler)公式。

ode1 方法是欧拉(Euler)公式。

discrete (固定步长) 是固定步长的 discrete 方法。

◆ 积分方法参数的设置

默认的积分方法参数对大多数问题都能提供精确和有效的结果。然而在一些情况中，改变参数可以提高仿真的性能。

◆ 步长设置

对于变步长方法，用户可以设置最大步长和初始步长参数。默认时，这些参数是系统

自动选择的，显示为auto。对于固定步长的方法，用户可以选择固定步长，默认也是auto。

最大步长是用于控制积分中可使用的最大时间步长。默认值由开始和结束时间决定：

$$h_{\max} = \frac{t_{stop} - t_{start}}{50}$$

一般来说默认值足够了，假如用户确定积分中失去了重要的步骤，可以改变参数阻止积分采用过大的步长。如果仿真的时间太长了，默认的步长就有可能过长。另外，如果用户的系统是个周期系统，那么把最大步长设置为其周期值的一部分(如1/4)是很好的。

默认时积分方法通过检查初始时间状态的派生来选择初始步长。初始步长太大，积分中可能会失去重要的步骤。初始步长是个建议值，也就是说积分法会尝试这个步长。如果错误发生，就要减小这个步长。

◆ 容许误差

积分方法使用标准局部误差控制技术在每一步控制误差。在每一步中，积分方法在每一步末计算状态值和局部误差值，并且评估这些状态的误差。然后比较局部误差和可接受的误差，如果在任何一状态中误差大于可接受的误差，积分方法减少步长并且重试。

相对误差是误差值和每一个状态大小的相对比值。相对误差代表了状态值误差的百分比。默认值为1e－3，意味着计算的状态值将精确到 0.1%。

绝对误差表示当状态接近于零的时候的可接受误差。

◆ ode15s 方法中的最大阶数

ode15 方法是基于 1 到 5 阶 NDF 公式。虽然越高阶数的公式越精确，但是这样也越不稳定。如果用户的系统是困难的并且需要更稳定，将最大阶数减小为 2。当用户选择 ode15s 方法时，对话框就会显示这个参数选择。

◆ 输出选项(Output options)

输出选项使用户可以控制仿真输出的产生，用户可以选择以下 3 个选项：产生经过优化的输出；产生附加的输出；产生指定的输出。

◆ 优化输出选项

当仿真输出太稀疏的时候，优化输出选项提供附加的输出点。这个参数在时间步长中提供一个整数输出点。例如，优化输出值为 2 在时间步长中提供一个中点输出值。默认的优化值为 1。要得到更平滑的输出，采用优化输出比减小步长快得多。

◆ 产生附加输出选项

这个选项使用户可以直接在仿真产生的结果中指定附加的时间点。当用户选择这个选项，SIMULINK 系统在 Solver 选项卡上显示输出时间的文本框。在这个文本框中输入 MATLAB 表达式来指定附加的时间或者是附加时间向量。附加的输出是在附加时间上用连续扩展公式产生的。与优化输出不同，这个选项改变了仿真的步长，因此仿真步数和用户指定的附加输出的时间点相对应。

◆ 仅产生指定的输出选项

这个选项仅在用户指定的时间点输出。同样，它也改变了仿真的步长，仿真的步数也与之对应。当用户想要比较不同的仿真模型在同一时间的不同输出时，这个选项特别有用。

以下举一个例子说明这 3 种选项的不同。假如有一个仿真模型在以下时间输出：

```
0, 2.5, 5, 8.5, 10
```

选择 Refine Output 选项并且指定优化值为 2，则输出以下的时间：

```
0, 1.25, 2.5, 3.75, 5, 6.75, 8.5, 9.25, 10
```

选择 Produce additional output 选项并指定文本框为[0:10]，则输出以下的时间：

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

也许会有其他的时间，这取决与用户在 Solver 选项卡中选择的步长。

选择 Produce Specified Output Only 选项并指定[0:10]，则输出以下的时间

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

- 设置工作区 I/O 参数(Workspace I/O) 选项卡

用户可以指定 SIMULINK 系统向 MATLAB 工作区变量输出仿真值，也可以从工作区输入仿真的初始值。在仿真参数设置对话框中选择 Workspace I/O 选项卡，则该选项卡显示如图 11.45 所示。

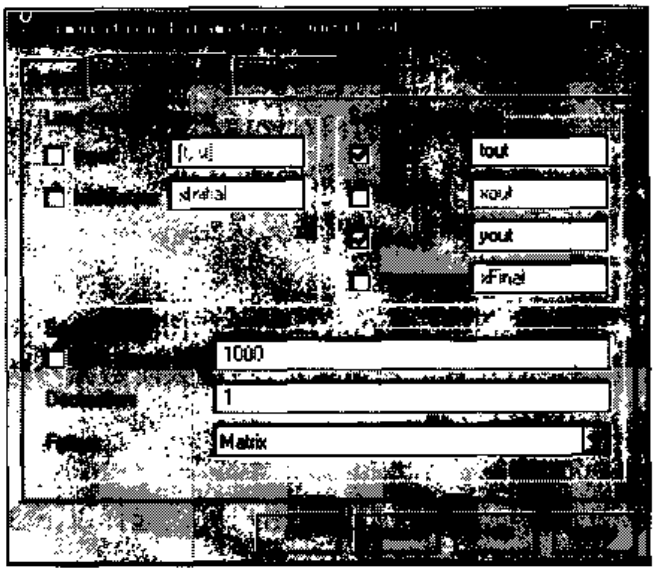


图 11.45 仿真参数对话框—工作区I/O选项卡

◆ 从基本工作区载入输入值

SIMULINK 系统可以在仿真运行中从模型的基本工作区向模块顶层的输入端输入数值。要指定这项功能，单击 WorkSpace I/O 选项卡中 Load from workspace 选项组中的 Input 复选框在其旁边的文本框中输入指定的外部输入值，然后单击 Apply 按钮。

外部输入值可以有以下几种形式：

外部输入矩阵

该矩阵的第一列必须是一个升序的时间向量，剩下各列指定输入值。每一列代表了模型中不同输入模块的信号，而每一行是在相应时间点各信号的值。如果在相应的输入端选择了插值选项，必要时 SIMULINK 系统会线性地内插或外插输入值。

当模型的输入端为 n 时，输入矩阵必须有 n+1 列。如果用户在基本工作区中定义了 t 和 u，就不必再指定外部输入值了，因为模型默认的外部输入值为[t,u]。

例如，假定一个模型有两个输入端：一个输入端接受两个信号，另一个接受一个信号。而在基本工作区中定义了如下两个变量u和t：

```
t = (0:0.1:1)';
u = [sin(t), cos(t), 4*cos(t)];
```

然后在对话框中指定由外部输入数据，具体的仿真结果请用户自己试验。

带时间的结构体

SIMULINK 系统可以用结构体从工作区读取数据，该结构体在 Input 文本框中指定名称。该输入结构体必须有两个顶层域——时间和信号。时间域包含一列仿真时间向量，信号域包含一列子结构体，各自与一个输入端相对应。

例如，考虑图 11.46 所示这个系统，有两个输入端。

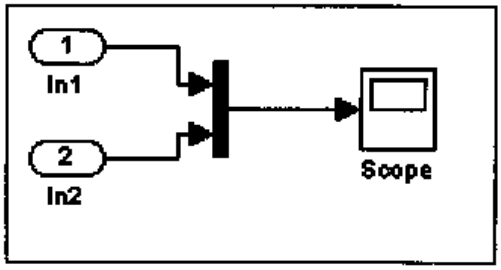


图 11.46　外部输入矩阵

假如基本工作区定义如下的模型输入向量：

```
a.time = (0:0.1:1)';
a.signals(1).values = sin(a.time);
a.signals(2).values = cos(a.time);
```

然后指定 a 为这个模型的外部输入值，选中 Input 复选框，然后在旁边的文本框中输入 a，就可开始仿真了。

Per-Port 结构体

这个格式由带时间的或者不带时间的相应端口分离结构体组成。每一个端口的输入数据结构体仅仅有一个信号域。要指定这个选项，在 Input 文本框中输入由逗号分割的数据表 in1，in2，…，inN。in1 是模型第一个端口的数据，in2 是模型第二个端口的数据，依此类推。

外部输入时间表达式

时间表达式可以是任何的 MATLAB 表达式，其计算出的行向量的长度必须等于模型的输入端数。具体的例子这里就不再讨论了。

◆　工作区中保存输出

用户可以通过在 Output 复选框中选择时间(Time)和状态(States)及指定返回的变量，使得 SIMULINK 向工作区写入时间、状态和输出向量值。

在复选框旁的文本框中输入变量名，向该变量指定数值。如果向多个变量写入输出值，可以在文本框中用逗号分割不同的变量名。SIMULINK 会把仿真的时间向量保存在 Time 文本框中指定的变量中。

在 Save Options 选项组中用户可以指定保存输出的数量限制和格式。具体的含义可以参看上面所讲的输入的各种数据格式。

● 诊断设置(Diagnostics)选项卡

在该选项卡中，用户可以指定在仿真中遇到各种情况时所需要的系统操作。诊断设置对话框如图 11.47 所示。

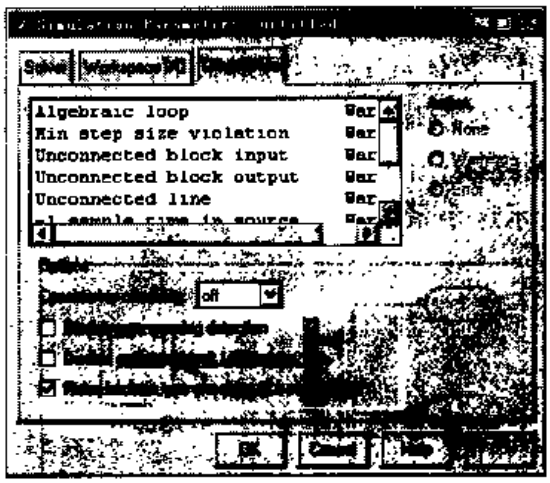


图 11.47 诊断设置对话框

对于任何类型的错误，用户可以选择是不需要给出信息、给出警告信息或者是给出出错信息。警告信息不会终止仿真，但是出错信息将会终止仿真。

◆ 一致性检验(Consistency checking)

一致性检验是一种调试手段，它使得一些由仿真ODE方法决定的假设有效。其主要的作用是保证S函数与SIMULINK内置的模块具有相同的规则。由于一致性检验极大地降低系统的表现(大约可降低40%)，因此一般都设置其为关闭状态。使用一致性检验使用户的S函数更有效并可防止一些不可预测的结果。

一致性检验的另一个目的是保证当输入一个给定的时间值时模块产生稳定的输出。对于存在困难的积分方法(ode23s和ode15s)这一点很重要。因为在计算雅可比行列式的时候，模块的输出函数可能会多次调用同样的时间值。

当设置了一致性检验后，SIMULINK 系统重新计算近似值并把它们和高速缓存中的值比较，如果数值不一样，一致性错误就会产生。SIMULINK 比较以下这些量的值：输出、零穿越、方差和状态。

◆ 禁止零穿越检测(Disable Zero Crossing Detection)

用户可以禁止仿真的零穿越检测。对于一个有零穿越的模块，禁止零穿越可以加快仿真的速度，但是可能对仿真结果的精度有负作用。

这一选项可以禁止那些具有内置零穿越检测功能模块的检测功能。但是不能禁止 Hit Crossing 模块的这一功能。

◆ 禁止模块优化 I/O 储存(Disable Opmized Block I/O Storage)

选中这个选项将导致 SIMULINK 向每一个模块的 I/O 值分配一个断续的内存缓冲区，而不是重新使用内存缓冲区。这样做可能会显著地增加大型模型仿真所需的内存数。因此除非用户需要调试一个模型，一般不要选择这个选项。当用户需要以下工作时，尤其需要禁止重新使用缓冲区：调试一个 C-MEX S 函数以及在调试的模型中使用了浮动示波器或者显示器来观察信号。

当允许缓冲区重用或者用户试图使用缓冲区已经被重用的浮动示波器和显示器时，SIMULINK 系统会打开一个出错对话框。

◆　放松布尔变量检查(与 2.x 版本兼容)(Relax boolean type checking)

选择这个选项将会使得需要布尔型变量输入的模块也可以接受双精度变量的输入。这使得原来由早期的版本(SIMULINK 2.x)创建的模块和新版本兼容。例如考虑图 11.48 所示的模块。

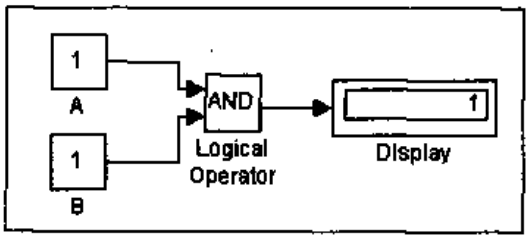


图 11.48　一个模型

这个模型将一个双精度的信号输入一个逻辑操作模块，它原本需要一个布尔型的信号输入。如果用户选择了 Relax boolean type checking 选项，在仿真的过程中系统就不会给出出错信息。

3. 提高仿真的效率和精度

仿真的效率和精度由很多因素决定，包括模型的设计和仿真参数选择。积分方法以及其默认的参数精确而有效地操纵了大多数模型的仿真。然而，对于一些模型，如果用户调整积分方法和参数可以得到更好的结果。同时，如果用户知道模型行为的信息，将这些信息提供给积分法，可以提高仿真的结果。

●　加速仿真

有很多原因会减慢仿真的信息，这里有如下一些原因：

◆　用户模型包含了 MATLAB Fcn 模块。这时，MATLAB 解释程序在每一步都会执行，大大降低了仿真的速度。因此在任何时候都要使用内置的 Fcn 模块或者基本数学计算模块。

◆　用户模块包含了 M 文件的 S 函数。M 文件 S 函数也导致每一步都执行 MATLAB 解释程序。解决的方法是把 S 函数转换为子系统，是转换为 C-MEX 文件的 S 函数。

◆　用户模型包含了 Memory 模块。使用 Memory 模块导致可变阶的积分方法(ode15 和 ode113)在每一步开始时都设置回 1。

◆　用户把最大步长改得太小了，试着使用默认值(auto)。

◆　用户把精度定得太高了，一般来说默认的相对容差 0.1%足够了。对于有接近零的状态的模型，如果绝对误差参数太小了，仿真可能会在接近零状态值附近计算太多步。

◆　仿真时间太长了，试着减少时间间隔。

◆　问题可能是困难的而用户采用了非困难的方法，试试 ode15s 方法。

◆　模型包含了代数环。代数环的解决方案在每一步都反复地计算，因此严重地降低了效率。

- 用户模型将一个 Random Number 模块传回一个积分模块。解决的方法是在连续系统中使用 Sources 库的有限带宽的白噪音模块(Band-Limited White Noise block)。
- 提高仿真精度

要检测一个仿真的精度，在一个合理的时间内运行仿真。然后，减小相对精度为 1e－4(默认为 1e－3)或者绝对精度再次运行，比较两次仿真的结果。如果结果不是显著的不同，用户可以确信解法是收敛的。

如果仿真在开始时错过重要的步骤，减少初始的步长以保证仿真不会错过重要的步骤。如果仿真结果在时间上变得不稳定，可能有以下的原因：

- 用户系统可能是不稳定的。
- 如果用户使用 ode15s，应当取仿真最大阶数为 2 或者使用 ode23s 方法。
- 对于一个有接近零状态的模型，如果绝对误差参数太大，到仿真在接近零状态值附近计算的步骤太少。减少这个参数值或者在积分器对话框中对每个分立状态调整其值。
- 如果减少绝对误差值不能有效地提高精度，请减少相对误差参数来减小可接受误差并强迫更小的步长和更多的步骤。

4. 用命令进行仿真

用命令进行仿真比用菜单进行仿真具有以下一些优点：

- 可以不理睬模块中的初始条件(参数 x0)。
- 如果只用来启动仿真命令，而对其左边的任何选项不作定义的话，MATLAB 将自动打印其输出。如果系统没有输出的话，则打印其状态。
- 可以定义任何外部输入(用参数 ut)。
- 可以由一个 M 文件来启动一个仿真，并且允许模块中的参数发生改变。
- 仿真可以稍微进行得快一些。

用来进行仿真的命令有 4 个。

- 使用 set_param 命令

用户可以使用 set_param 命令来开始、停止或者继续仿真或者更新一个模块的方框图。同样，用户可以使用 get_param 命令来检查一个仿真的状态。

set_param 命令的使用格式是：

```
set_param('sys', 'SimulationCommand', 'cmd')
```

这里 sys 是系统的名称而 cmd 是命令的名称，包括 start(开始仿真)、pause(暂停仿真)、continue(继续仿真)和 update(更新方框图)。

get_param 命令的使用格式是：

```
get_param('sys', 'SimulationStatus')
```

执行该命令，SIMULINK 系统将会返回此时仿真的状态，包括 stopped(仿真停止)、initializing(仿真初始)、running(仿真运行)、paused(仿真暂停)、terminating(仿真终止)和external(仿真外部)。

● 使用 Sim 命令

用来启动仿真的命令，其一般格式为：

```
[t,x,y] = sim(model, timespan, options, ut);
```

说明：

除了模型的名称外，用户可以使用空矩阵([])来设置右边的各参数，这时 sim 命令使用默认的参数值。用户可以使用 options 参数提供附加的仿真参数，包括积分方法和容许误差。用户使用 simset 命令来定义 options 参数。如果用户想要仿真一个连续系统，必须使用 simset 命令来指定 solver 参数。对于完全的离散模型，积分法(solver)默认为可变步长离散法(Variable Step Discrete)。

参数：

t　返回仿真的时间向量。

x　返回仿真的状态矩阵，前面是连续状态，接着是离散状态。

y　返回仿真的输出向量。每一列包含根层输出模块的输出量，以端口的数字为序。如果一个输出端口的输入是向量，则其输出具有相应数量的列。

model　模块方框图的名称。

timespan　仿真开始和停止的时间。可以用以下这些形式指定参数：

tFinal　指定停止的时间，开始时间为零。

[tStart tFianl]　指定开始和停止的时间。

[tStart Output Times tFinal]　指定开始和停止的时间和在t中返回的时间点。

options　由simset命令产生的一个结构体用来指定仿真的参数。

ut　向顶层输入端口模块输入的外部数据。ut可以是MATLAB的函数(以字符串的形式表达)，用来指定每一个时间点的输入u=UT(t)。这是个包含所有输入端口输入值的表或者用逗号分隔的列表ut1，ut2，...，每一个都对应于特定的端口。

【例1】用命令仿真Van der Pol方程，SIMULINK自带了vdp模型。

以下这个命令使用所有的默认参数：

```
[t,x,y] = sim('vdp')
```

以下这个命令定义了Refine参数的值为2：

```
[t,x,y] = sim('vdp', [], simset('Refine',2));
```

以下这个命令仿真1000s，保存返回变量的最后100行。仿真输出值只有t和y，但是在xFinal变量中保存最后的状态向量。

```
[t,x,y] = sim('vdp', 1000, simset ('MaxRows',100,'OutputVariables ','
ty','FinalStateName', 'xFinal'));
```

● 使用 Simset 命令

Simset 命令是用来向 Sim 命令产生或者编辑仿真参数和积分法属性的命令，其调用格式为：

options = simset(property, value, ...)　设置指定的参数值并保存在 options 中

options = simset(old_opstruct, property, value, ...)　修改名为 old_opstruct 结构体中指定参数值并保存在新的 options 中

options = simset(old_opstruct, new_opstruct)　将 old_opstruct 和 new_opstruct 结

构体结合在新的 options 中，所有两个结构体相同的参数都将用 new_opstruct 结构体的值来代替

simset 没有任何输入参数的命令，将会显示仿真的所有属性名称和所有可能的取值

说明：

Simset 命令产生一个名为 options 的结构体，其中可以指定仿真的参数和积分法的属性。所有用户没有指定的参数都使用默认值。该命令需要用 property 来指定所需参数的名称，而参数值要视参数名称而定。

参数：

◆ AbsTol 正数，默认值为 1e－6

绝对容许误差，仿真中用于状态向量的所有元素，该参数仅用于可变步长的积分法。

◆ DstWorkspace 可选 base | {current} | parent，默认为 current

这个属性指定工作区来分配仿真返回的变量。

◆ FinalStateName 字符串，默认为''

最终状态名称变量用于指定仿真结束后 SIMULINK 保存模型状态的变量名称。

◆ FixedStep 正数

固定步长。该参数仅用于固定步长的积分法。如果模型包含离散的元素，默认值为基本采样时间，否则默认值是仿真间隔的五十分之一。

◆ InitialState 向量，默认为空[ ]

连续和离散状态的初始值。先是连续状态的初始状态向量，接着是离散状态。InitialState 向量将取代在模型中指定的初始状态。

◆ InitialStep 正整数，默认为 auto

用户建议的初始步长。该参数在前面参数设置对话框中有详细的介绍。

◆ MaxOrder 默认为 5

ode15 积分法的最大运算阶数。

◆ MaxRows 非负整数，默认为 0

输出行的限制数。这个参数限制在 t、x 和 y 中返回的行向量中记录的时间点数。当为默认值时，表示没有限制。

◆ MaxStep 正数，默认为 auto

步长的上限。默认为系统自动设置的仿真时间的五十分之一。

◆ OutputPoints 默认为 all

确定的输出点。当指定该参数，仅仅在指定的时间点输出 t、x 和 y 值。如果设置为 all，则输出积分中每一步计算出的结果。

◆ OutputVariables 默认为 txy 可在 txy | tx | ty | xy | t | x | y 这些选项中选择

设置输出的变量。可以没有 t、x、y，仿真会相应地输出空矩阵。

◆ Refine 正整数，默认为 1

输出优化参数。详见参数设置对话框。

◆ RelTol 正数，默认为 1e－3

相对容许误差。

◆ Solver 选择积分法

可变步长系统可选择 Discrete |ode45 | ode23 | ode113 | ode15s | ode23s，固定步长系统可选择 ode5 | ode4 | ode3 | ode2 | ode1

◆ SrcWorkspace 可选{base} | current | parent，默认为 base。

设置计算模型中 MATLAB 表达式的工作区。

◆ Trace 可选'minstep'，'siminfo'，'compile' ，{''}，默认为空('')。

追踪工具。该参数启动仿真的追踪工具。设置 minstep 追踪标志指定当结果突然变化以致可变步长积分法不能进行以保证安全的误差限时终止仿真。默认时，SIMULINK 会给出警告信息，但会继续仿真。设置 siminfo 追踪标志，在仿真开始时提供仿真参数的简短摘要。设置 compile 追踪标志，显示模块中的编辑状态。

◆ ZeroCross {on} | off

启动/停止检测零穿越。详见参数设置对话框。

【例 2】这个命令产生一个名为 myopts 的结构体来定义 MaxRows 和 Refine 参数，其他的参数都取默认值

```
myopts = simset('MaxRows', 100, 'Refine', 2);
```

使用在 myopts 中定义的参数仿真 vdp 模型，仿真时间为 10s。

```
[t,x,y] = sim('vdp', 10, myopts);
```

具体的仿真结果请用户亲自试一试。

● 使用 Simget 命令

使用该命令可以得到选项结构体属性和参数。其具体的调用格式为：

```
struct = simget(model)
value = simget(model, property)
value = simget(OptionStructure, property)
```

说明：

Simget 命令得到仿真的参数和指定 SIMULINK 模型的积分法属性值。如果参数或属性是由变量定义的，Simget 返回变量值，而不是变量名。如果在工作区中不存在相应的变量，SIMULINK 系统提示出错信息。

【例 3】以下命令重新得到 vdp 模型的选项结构体

```
options = simget('vdp');
options =
            AbsTol: 1.0000e-006
             Debug: 'off'
        Decimation: 1
      DstWorkspace: 'current'
    FinalStateName: ''
         FixedStep: 'auto'
      InitialState: []
       InitialStep: 'auto'
          MaxOrder: 5
        SaveFormat: 'Matrix'
```

```
            MaxRows: 0
            MaxStep: 'auto'
       OutputPoints: 'all'
    OutputVariables: ''
             Refine: 1
             RelTol: 0.0010
             Solver: 'ode45'
       SrcWorkspace: 'base'
              Trace: ''
          ZeroCross: 'on'
```

以下命令得到 Refine 参数：

```
refine = simget('vdp', 'Refine');
refine =
    1
```

11.3.2 线性化分析

1. 线性化命令

SIMULINK系统利用线性化分析函数linfun在操作点附近来提取系统的线性状态空间模型。这一类函数的调用格式为：

```
[A,B,C,D] = linfun('sys')
[A,B,C,D] = linfun('sys', x, u)
[A,B,C,D] = linfun('sys', x, u, pert)
[A,B,C,D] = linfun('sys', x, u, pert, xpert, upert)
```

参数：

- linfun　线性化函数，包括 linmod，dlinmod 和 linmod2
- model　需要线性化的模型名称
- x 和 u　状态和输入向量。如果定义这些量，那么这些量就设置了工作点，并在这个点上提取线性模型
- pert　可选标量。用来设置 x 和 u 的干扰。如果这个值没有定义，就采样默认值 1e−5
- xpert 和 upert　可选向量。用来为每一个状态和输入设置扰动。如果定义了这个向量，那么就忽略上面的标量 pert，在这种情况下，状态 x 的第 i 个元素受到扰动后就变成 x(i)+xpert(i)；输入 u 的第 i 个元素受到扰动后就变成 u(j)+upert(j)。

说明：

linmod 得到的是用常微分方程描述的 SIMULINK 模型的线性模型，返回的模型是以状态空间 A、B、C、D 形式来表示其输入和输出之间的关系。这样的线性模型可用下面

的式子来表示：

［A，B，C，D］＝linmod('model')可以得到在状态变量 x＝0 和输入 u=0 这个工作点附近的线性化模型。

linmod 是在工作点附近对状态施加扰动后来确定状态导数和输出的变化速率(即雅可比矩阵)，并把得到的结果用来计算状态空间矩阵，每一个状态 x(i)收到扰动后变成：

$$x(i)+\Delta(i)$$

其中

$$\Delta(i)=\delta(1+|x(i)|)$$

同样，第 j 项输入受到扰动后，变成

$$u(j)+\Delta(j)$$

其中

$$\Delta(j)=\delta(1+|u(j)|)$$

2. 离散系统的线性化

dlinmod 能够以任意给定的采样时间对离散系统、多速率系统以及连续和离散混合系统进行线性化。除了第二个选项需要插入采样时间来对系统线性化以外，dlinmod 的调用格式和 linmod 是相同的。例如：

```
[Ad, Bd, Cd, Dd]=dlinmod('model', Ts, x, u)
```

上面这条命令可以产生一个以状态向量 x 和输入向量 u 为工作点、采样时间为 Ts 的离散状态空间模型。要想得到一个离散系统的近似连续模型，只要把 Ts 设为 0 即可。

如果一个模型是由线性模块、多速率模块、离散模块和连续模块组成的，那么在满足下列条件的情况下，dlinmod 产生一个采样时间为 Ts 且具有相同频率和时间响应的线性模型：

- Ts 是系统中所有采样时间的整数倍。
- Ts 不小于系统中最慢的采样时间。
- 系统是稳定的。

不过，在上面这些条件不满足的情况下，也有可能得到有效的线性模型，但这要看一个系统是否稳定。实际上只要计算线性化后所得矩阵 Ad 的特征值就可以了。因此，如果 Ts>0 而且特征值在单位圆内，即

```
all(abs(eig(Ad)))<1
```

那么系统就是稳定的。同样，如果 Ts＝0，而且所有的特征值在左半平面，那么系统也是稳定的，即：

```
all(abs(eig(Ad)))<0
```

当系统不稳定且采样时间不是其他采样时间整数倍的时候，dlinmod 就可能产生复矩阵 Ad 和 Bd。然而在这种情况下，仍然可以通过矩阵 Ad 的特征值来验证系统的稳定性。dlinmod 可以用来把一个系统的采样时间变成其他值，或者把一个线性离散系统变成一个连续系统。或者反过来，把一个连续系统变成一个离散系统。

如果需要求一个连续系统的频率响应，可用 MATLAB 的 bode 命令。

3. 线性化的一种高级形式

程序 linmod2 提供了一种线性化的高级形式。这个程序和 linmod 相比，运行需要更长的时间，但产生的结果要比 linmod 精确。

linmod2 的调用格式和 linmod 调用格式相似，但功能不同。例如，linmod2('model'，x，u)和 linmod 一样产生一个线性模型。然后，状态空间矩阵的每一个元素的干扰是逐个设置的，以便使与舍入误差和截断误差有关的误差达到最小。

linmod2 设法平衡舍入误差(由于小扰动引起的，并与计算精度有关的误差)与截断误差(由于大扰动引起的，使分段线性近似无效而产生的误差)，并在两者之间进行折衷。

对于调用命令[A，B，C，D]=linmod2('model'，x，u，pert)，变量 pert 表示可以施加的最小扰动，默认值为 1e－8。linmod2 的优点是能够探测到不连续性，并产生一个警告信息，例如：

```
warning: discontinuity detected ad A(2, 3)
```

当出现上述这样的警告信息时，可在其他的工作点附近进行试验来得到线性化模型。

对于调用命令：

```
[A, B, C, D] =linmod2('model', x, u, pert, Apert, Bpert, Cpert, Dpert)
```

其中变量 Apert、Bpert、Cpert 和 Dpert 是用来为每一个状态和每个输入的组合而设置的干扰矩阵。因此，xpert 的第 ij 个元素是与矩阵 A 的第 ij 个元素有关的干扰。默认的干扰可用下面的命令来获得：

```
[A, B, C, D, Apert, Bpert, Cpert, Dpert]=linmod2('model', x, u)
```

【例 4】有如下的一个需要仿真的方程：

$$x=Ax+Bu$$

$$y=Cx+Du$$

其中 x、u 和 y 分别为状态、输入和输出向量。例如，图 11.49 所示为一个名为 lmod 的模型。

要提取这个 SIMULINK 模型的线性化模型，可以使用下面的命令：

```
[A,B,C,D]=linmod('lmod')
A =
    -1     0     1
    -1    -2    -1
     0     1     0
B =
     0
     1
     0
C =
    -1     0     0
     0     0     1
D =
```

```
1
0
```

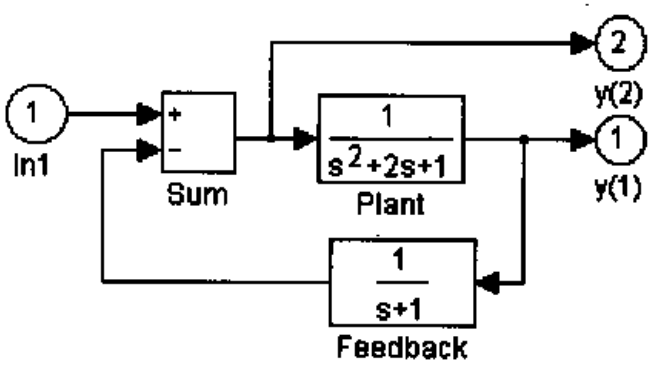


图 11.49　线性化模型

系统的输入和输出必须由 Signals&Systems 库中的 In1 和 Out1 模块来定义。Signal Generator 和 Scope 模块不能作为系统的输入和输出。一旦得到状态空间形式的数据，就可以用控制系统工具箱(Control System Toolbox)中的函数对其进一步的如下分析。

- 转换成传递函数形式

```
sys= ss(A,B,C,D);
```

- 伯特图(Bode plot)

```
bode(A,B,C,D);或者 bode(sys)
```

- 线性化的时间响应

```
step(A,B,C,D) 或 step(sys)
impulse(A,B,C,D) 或 impulse(sys)
lsim(A,B,C,D,u,t) 或 lsim(sys,u,t)
```

工具箱中的其他一些函数可用来对线性系统进行控制器设计。

当模型是非线性时，首先必须选定一个工作点，然后在这个工作点附近提取线性模型。非线性模型对于工作点处的干扰是非常敏感的，因此必须在截断和舍入误差之间进行折衷。linmod 命令的另外一些选项分别定义了工作点和干扰点。

```
[A,B,C,D]=linmod('model',x,u,pert,xpert,upert)
```

对于离散系统或者离散、连续这样的混合系统，可用函数 dlinmod 进行线性化。除了右边第二个选项必须具有采样时间以外，这个命令和 linmod 一样，具有相同的调用格式。

对于模型中含有 Derivative 或 Transport　Delay 模块的系统，用 linmod 进行线性化比较麻烦。因此联机性化之前，用特殊设计的模块来替换它们，以便避免出现这种问题。这些特殊设计的模块可在 Extras library 的 Sublibrary 中找到。

例如，可以用右边的模块代替左边的两个模块。

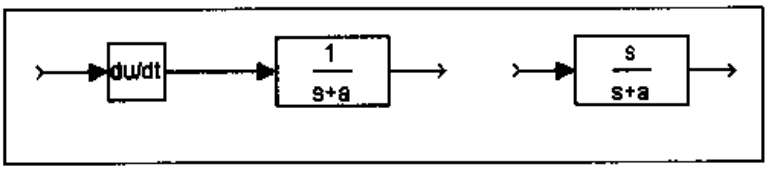


图 11.50　模块的替换

11.3.3　平衡分析

SIMULINK 提供的函数 trim 可以用来确定稳态平衡点。命令的基本格式是：

```
[x,u,y,dx] = trim('sys')
[x,u,y,dx] = trim('sys',x0,u0,y0)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx,options)
[x,u,y,dx] = trim('sys',x0,u0,y0,ix,iu,iy,dx0,idx,options,t)
[x,u,y,dx,options] = trim('sys',...)
```

说明

trim 的功能是试图找到输入 u 和状态 x 的值来使状态的导数为 0，这样的工作点被称作平衡点，即系统在稳态时的工作点。既然这样的平衡点通常不是唯一的，因此有必要对状态 x、输入 u 和输出 y 的值进行固定。

命令[x,u,y]＝trim('model')的功能是设法找到一个平衡点，使[x;u;y]的绝对值达到最小。

命令[x,u,y]＝trim('model',x0,u0,y0)为状态 x,输入 u 和输出 y 定义了各自初始的猜测值。在这种情况下，trim 就使[x-x0;u-u0;y-y0]的最大绝对值达到最小。

x、u、y 的每一个元素可用下面的调用格式进行固定：

```
trim('model',x0, u0,y0,ix,iu,iy)
```

整数向量 ix、iu 和 iy 用来指出对 x0、u0 和 y0 的哪一个元素进行固定。既然无法保证平衡点一定存在，因此问题就转换为寻找一个稳态值，使得

```
abs( [x(ix)-x0(ix);u(iu)-u0(iu);y(iy)-y0(iy)]
```

最大绝对值达到最小。

trim 在限制状态导数为零的情况下，用一个有约束的优化算法来求解一个由 x、u 和 y 的希望值所组成的最大绝对值最小问题。对于这样一个问题，有可能没有可行解。在这种情况下，trim 就在状态导数偏离 0 这种最坏条件下，使上面的最大绝对值达到最小。

要使导数固定为一个非 0 值，可以采用

```
[x,u,y,dx]=trim('model', x0, u0, y0.ix, iu, iy, dx0, idx)
```

其中 dx0 表示希望的偏离值，idx 用来表示对 dx 中的哪一个元素进行固定。

【例 5】下面仍以 lmod 的模型为例(如上节中图 11.49 所示)

在这个模型中，用户可以用 trim 函数来找到使输出为 1 时输入和状态所对应的值。首先输入 u 和状态变量 x 作为初始假设，然后设置输出 y 的希望值：

```
x=[0;0;0];
u=0;
y=[1;1];
```

可用指针变量指出哪个变量是固定的，哪个变量是可以变化的，如：

```
ix=[];          %没有固定任何状态
iu=[];          %没有固定输入
iy=[1;2];       %固定输出端口 1 和输出端口 2
```

调用 trim 函数，返回所得的解。

```
[x,u,y,dx]=trim('lmod',x,u,y,ix,iu,iy);
x =
    1.0000
         0
    1.0000
u =
    2.0000
y =
    1.0000
    1.0000
dx =
  1.0e-015 *
   -0.2220
    0.2220
         0
```

由于舍入误差的影响，所得的结果可能会有所差异。

注意： 有时会遇到没有平衡点的问题，在这种情况下，在首先设置偏差为 0 以后返回的解便是使最大偏差为最小的那个解。

11.4　封装定制新模块

利用 SIMULINK 的封装(Mask)功能，可以定做一个模块或者一个子系统的对话框和图标。本节主要讨论以下一些问题：

- 封装过程的概述。
- 用封装的办法创建新模块。
- 为新模块定义图标。

11.4.1 封装过程概述

1. 封装的作用

一般来说，采用封装的办法，有下列好处：

- 使模型的用户与模型不必要的复杂性隔绝开来。
- 提供一个描述性的、有益的用户接口。
- 保护模块的内容免受意外干扰的影响。

对于一个具有一个以上模块的子系统来说，封装的一个重要用途是帮助用户创建一个对话框来接受参数。这样就无需打开子系统中各个模块的对话框，然后再逐个输入参数。而是把这些模块组成一个子系统，然后定义这个子系统的对话框来接受这些参数值。

2. 封装的步骤

一般来说，整个封装过程包括下列步骤：

(1) 把各个模块打开，然后给需要在新的对话框中进行赋值的那些参数指定一个变量名。

(2) 创建子系统。有关子系统的创建方法详见第 11.2.6 节的有关内容。

(3) 选择子系统模块，然后在 Edit 菜单中选择 Edit Mask 命令。

(4) 填写封装对话框。

(5) 单击 OK 按钮来创建新模块，子系统模块所显示的图标就是封装对话框中定义的图标。如果打开了新的子系统模块，SIMULINK 就显示用户创建的新对话框。

11.4.2 用封装的办法创建模块

1. 封装模块功能

图 11.51 所示是个线性方程 y=mx+b 的模型。

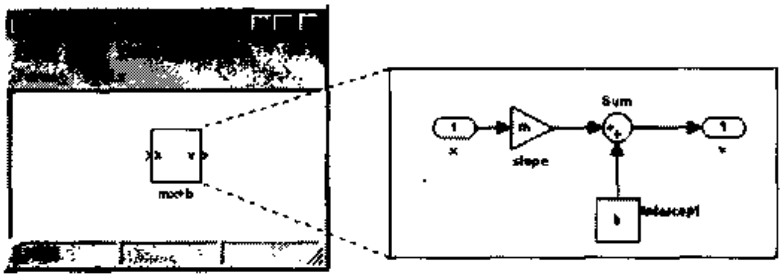


图 11.51 mx+b模型

一般来说，用户双击子系统模块便可打开它，将其显示在另一个窗口中。mx+b 子系统包含了一个 Gain 模块，命名为 Slope，其增益参数指定为 m；一个 Constant 模块，命名为 Intercept，其常数值参数指定为 b。这些参数代表一条直线的斜率和截距。在本例中将

为这个子系统产生一个用户对话框和图标。对话框包含对斜率和截距的提示。在完成封装后，用户只要双击子系统模块就可以打开该对话框。封装对话框和图标如图 11.52 所示。

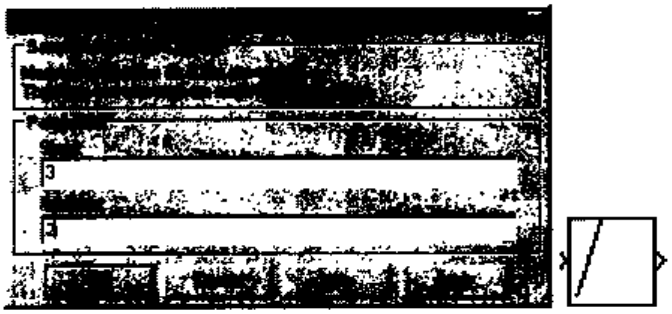


图 11.52　参数对话框

用户向封装对话框输入 Slope 和 Intercept 的值。SIMULINK 系统使这些值对下层子系统的所有模块都有效。封装这个子系统产生一个子包含的函数单元，拥有自己的特定函数参数——斜率(Slope)和截距(Intercept)。封装将这些封装参数映射给底层模块的一般参数。子系统这一复杂的功能封装于一个新的界面中，而该界面与 SIMULINK 内置的模块看上去和感觉上都是一样的。

2.　产生封装提示对话框

要产生这个系统的封装，先选取子系统模块然后从 Edit 菜单中选取 Mask Subsystem 命令。

封装对话框开始时大都显示 Mask Editor 对话框的 Initialization 选项卡。在本例中该显示如图 11.53 所示。

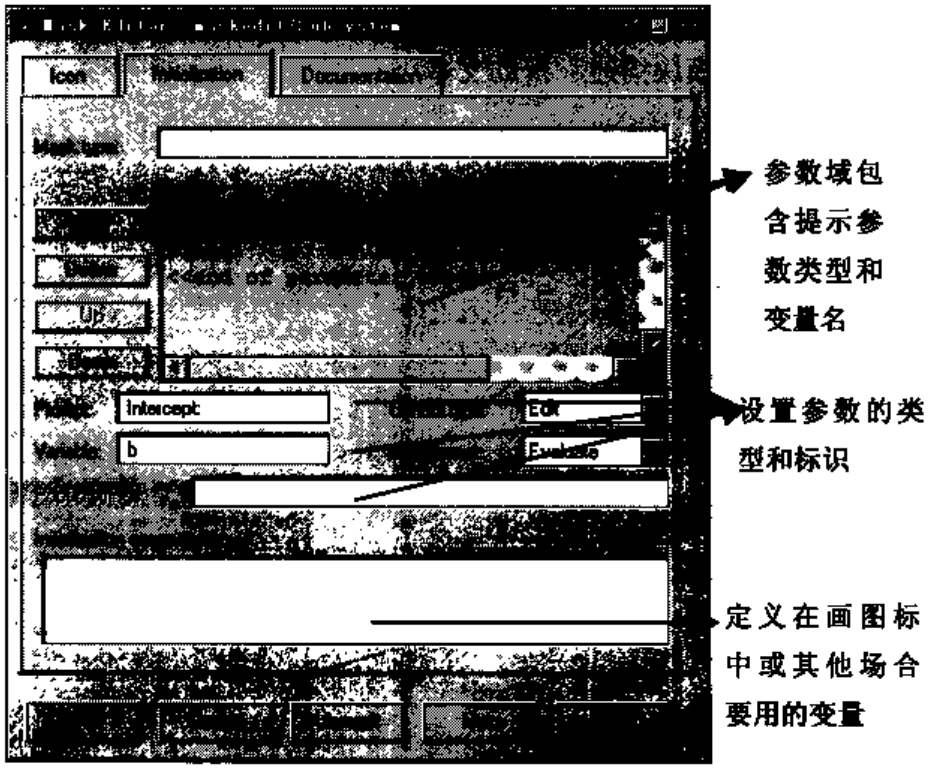


图 11.53　封装模块

封装编辑器让用户指定封装的如下属性：

- prompt： 描述参数的文本标识。
- Control type： 用户界面的控制风格用来决定在对话框中参数值是如何输入或选中的。
- Variable： 储存参数值的变量名。

Slope 和 Intercept 定义为 Edit 控件。这意味着在封装对话框中用户在文本框中输入其值。这些值保存在封装工作区的变量中。被封装的模块只能在封装工作区中存取变量。在本例中，在 Slope 文本框中输入的值分配到变量 m 中。封装子系统中的 Slope 模块从封装工作区中得到 Slope 参数。图 11.54 显示了在 Mask Editor 对话框中定义的 Slope 参数如何映射到实际的封装对话框参数中。

图 11.54 参数的映射

在为斜率(Slope)和截距(Intercept)产生封装参数后，单击OK按钮。然后双击子系统新建立的系统模块，在弹出的对话框中输入Slope值为3，Intercept值为2。

3. 产生封装模块描述和帮助文本

在 Documentation 选项卡中可以定义模块的封装类型、模块描述和帮助文本。在本例中，该选项卡以及其与封装对话框的关系，如图 11.55 所示。

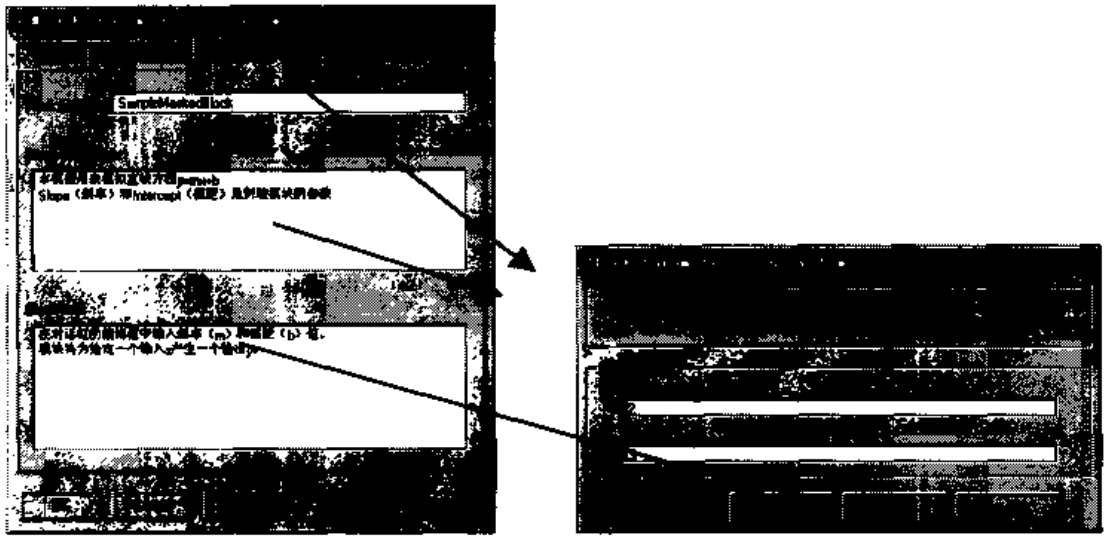


图 11.55 描述和帮助文本

4. 产生模块的图标

至此已经为 mx+b 子系统产生了一个子定义的对话框。然而，子系统模块仍然显示一般的 SIMULINK 子系统图标。这个封装模块适当的图标是标识直线斜率的图形。对于斜率为 3 的系统，图标应该如图 11.56 所示。

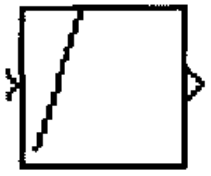


图 11.56　模块的图标

可以在 Mask Editor 对话框的 Icon 选项卡中定义图标，该选项卡如图 11.57 所示。

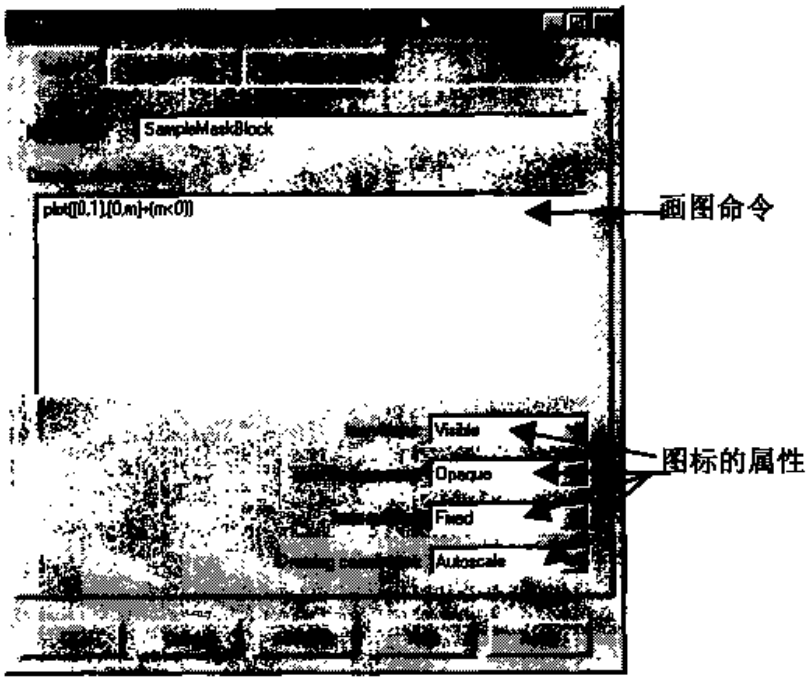


图 11.57　产生图标

画图命令 plot([0,1],[0,m]+(m<0))从(0，0)到(0，m)画一条直线。如果斜率是负值，SIMULINK 将直线移动 m，使直线一直在模块的可见作图范围内。画图命令可以作用于封装工作区的所有变量。当用户输入不同的斜率，图标就会更新重画直线。在图标参数的列表选项中选择 Normalized 和 Drawing Coordinates 选项，以保证图标画在一个左下角为(0，0)右上角为(1，1)的图框中。

第 12 章
工具箱初步

本章要点：

本章将介绍一些与 MATLAB 系统的工具箱(TOOLBOX)有关的内容。工具箱是 MATLAB 功能的进一步扩展，是 Mathwork 公司和第三方在 MATLAB 主包提供的强大的数值运算的基础上，对具体的工程问题提供的特殊函数集。用工具箱可以帮助解决特殊的工程问题。

本章具体包括以下内容:

- MATLAB 各种工具箱概述
- 控制系统工具箱简介
- 信号处理工具箱简介
- 优化工具箱简介

现在，MATLAB 已经有一个系列产品——MATLAB 主包和各种工具箱(TOOLBOX)。功能丰富的工具箱将不同领域、不同方向的研究者吸引到 MATLAB 的编程环境中来，成为 MATLAB 的忠实用户。迄今所有的 30 多个工具箱大致可分为两类——功能型工具箱和领域型工具箱。

功能型工具箱主要用来扩充 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能以及与硬件实时交互功能，能用于多种学科。而领域型工具箱是专业性很强的，如控制工具箱(Control Toolbox)、信号处理工具箱(Sign Processing Toolbox)等。

12.1 工具箱现状

下面，将 MATLAB 工具箱所包含的主要内容做简要介绍。

1. **通讯工具箱(Communication Toolbox)**

- 提供 100 多个函数和 150 多个 SIMULINK 模块用于通讯系统的仿真和分析
  - —— 信号编码
  - —— 调制解调
  - —— 滤波器和均衡器设计
  - —— 通道模型
  - —— 同步
- 可由结构图直接生成可应用的 C 语言源代码

2. **控制系统工具箱(Control System Toolbox)**

- 连续系统设计和离散系统设计
- 状态空间和传递函数
- 模型转换
- 频域响应：Bode 图、Nyquist 图和 Nichols 图
- 时域响应：冲击响应、阶跃响应和斜波响应等
- 根轨迹、极点配置和 LQG

3. **财政金融工具箱(Financial Toolbox)**

- 成本、利润分析，市场灵敏度分析
- 业务量分析及优化
- 偏差分析
- 资金流量估算
- 财务报表

4. **频率域系统辨识工具箱(Frequency Domain System Indentification Toolbox)**

- 辨识具有未知延迟的连续和离散系统
- 计算幅值 / 相位、零点 / 极点的置信区间

- 设计周期激励信号、最小峰值、最优能量谱等

5. **模糊逻辑工具箱**(Fuzzy Logic Toolbox)

- 友好的交互设计界面
- 自适应神经——模糊学习、聚类以及 Sugeno 推理
- 支持 SIMULINK 动态仿真
- 可生成 C 语言源代码用于实时应用

6. **高阶谱分析工具箱**(Higher Order Spectral Analysis Toolbox)

- 高阶谱估计
- 信号中非线性特征的检测和刻画
- 延时估计
- 幅值和相位重构
- 阵列信号处理
- 谐波重构

7. **图像处理工具箱**(Image Processing Toolbox)

- 二维滤波器设计和滤波
- 图像恢复增强
- 色彩、集合及形态操作
- 二维变换
- 图像分析和统计

8. **线性矩阵不等式控制工具箱**(LMI Control Toolbox)

- LMI 的基本用途
- 基于 GUI 的 LMI 编辑器
- LMI 问题的有效解法
- LMI 问题解决方案

9. **模型预测控制工具箱**(Model Predictive Control Toolbox)

- 建模、辨识及验证
- 支持 MISO 模型和 MIMO 模型
- 阶跃响应和状态空间模型

10. **μ 分析与综合工具箱**(μ—Analysis and Synthesis Toolbox)

- μ 分析与综合
- H2 和 H 最优综合
- 模型降阶
- 连续和离散系统
- μ 分析与综合理论

11. 神经网络工具箱(Neural Network Toolbox)

- BP、Hopfield、Kohoncn、自组织、径向基函数等网络
- 竞争、线性、Signloidal 等传递函数
- 前馈、递归等网络结构
- 性能分析及应用

12. 优化工具箱(Optimization Toolbox)

- 线性规划和二次规划
- 求函数的最大值和最小值
- 多目标优化
- 约束条件下的优化
- 非线性方程求解

13. 偏微分方程工具箱(Partial Differential Equation Toolbox)

- 二维偏微分方程的图形处理
- 几何表示
- 自适应曲面绘制
- 有限元方法

14. 鲁棒控制工具箱(Robust Control Toolbox)

- LQG / LTR 最优综合
- H2 和 H 最优综合
- 奇异值模型降阶
- 谱分解和建模

15. 信号处理工具箱(Signal Processing Toolbox)

- 数字和模拟滤波器设计、应用及仿真
- 谱分析和估计
- FFT、DCT 等变换
- 参数化模型

16. 样条工具箱(Spline Toolbox)

- 分段多项式和 B 样条
- 样条的构造
- 曲线拟合及平滑
- 函数微分、积分

17. 统计工具箱(Statistics Toolbox)

- 概率分布和随机数生成
- 多变量分析

- 回归分析
- 主元分析
- 假设检验

18. 符号数学工具箱(Symbolic Math Toolbox)

- 符号表达式和符号矩阵的创建
- 符号微积分、线性代数、方程求解
- 因式分解、展开和简化
- 符号函数的二维图形
- 图形化函数计算器

19. 系统辨识工具箱(System Identification Toolboox)

- 状态空间和传递函数模型
- 模型验证
- MA、AR、ARMA 等
- 基于模型的信号处理
- 谱分析

20. 小波工具箱(Wavelet Toolbox)

- 基于小波的分析和综合
- 图形界面和命令行接口
- 连续和离散小波变换及小波包
- 一维、二维小波
- 自适应去噪和压缩

随着 MATLAB 的不断扩充，工具箱将会越来越多。而工具箱实际上不过是在 MATLAB 系统上开发的一组实用的 M 文件函数命令或者是 SIMULINK 仿真模型。因此只要用户有兴趣和要求，自己也可以开发特殊用途的工具箱。而且目前的趋势是出现大量的由第三方开发的工具箱。

本章的目的不是详细介绍各种工具箱的功能和使用方法，而是旨在向用户展示最常用工具箱的一部分内容，以方便用户更深入地学习使用工具箱。

12.2　控制系统工具箱简介

控制系统的计算机辅助设计技术自从成为一项专门的学科以来已有 20 多年的历史，它一直受到控制界的普遍重视，在其发展过程中出现了各种各样的实用工具和理论成果。MATLAB 正是进行这项设计技术的方便可行的工具。MATLAB 已经成为国际控制界应用最广的语言和工具。

MATLAB 中含有极丰富的专用于控制工程与系统分析的函数。一些常见的运算如复数运算、求特征值、求根、矩阵逆运算与 FFT 等，一般会觉得它们特别麻烦，但在 MATLAB

的控制系统工具箱中，它们仅仅是几个简单例子，用一条语句即可解决。MATLAB 中的线性代数、矩阵计算和数值分析等功能为控制系统的分析与设计提供了可靠和方便的支持。

控制系统工具箱实际上是一个算法的集合，它使用关于复数矩阵的函数来提供控制工程的专用函数，其中大部分是 M 文件，都可以直接调用。利用这些函数就可以完成控制系统的时域或频域设计、分析与建模。

无论是对于连续的还是离散的系统，在控制系统工具箱中都能用传递函数或状态空间等形式来表示，利用经典或现代控制技术来处理。不仅如此，利用工具箱还可以进行模型之间的转换。大家熟知的控制系统分析中的时间响应、频域响应、根轨迹等都能够进行方便的计算并画出图形。利用其他的一些函数还可以进行如极点配置、最优控制与估计方面的设计。更为重要的一点是，用户可以通过编制 M 文件来任意添加工具箱中原来没有的工具函数。

12.2.1 安装

MATLAB 的安装指示中有控制系统工具箱的安装提示，可以在安装 MATLAB 时按提示插入含控制系统工具箱的磁盘进行工具箱安装，也可以在安装完 MATLAB 后随时添加控制系统工具箱，只需在 Window 98 环境下运行光盘中的 setup.exe 即可。其他工具箱的安装方式都是一样的，用户只要正确地按照 MATLAB 系统的安装提示进行，就可以安装上所需要的工具箱。

用户在此工具箱中可以找到 5 个演示用的 M 文件，其中示例程序 ctrldemo.m 演示一些基本的控制设计函数，而 jetdemo.m、diskdemo.m、boildemo.m 和 kalnldcmo.m 则是一个设计实例。可以通过在 MATLAB 的命令窗口中直接输入文件名并按 Enter 键来进行控制系统设计实例的演示。

12.2.2 控制系统分析

控制系统工具箱提供了用于对控制系统的时域和频域特征进行分析的工具函数，对连续和离散系统均能进行分析，而且适用于用传递函数或状态空间表示的模型。

1. 时间响应

对控制系统来说，系统的数学模型实际上是某种微分方程或差分方程模型。因而在仿真过程中需要以某种数值算法从给定的初始值出发，逐步计算出每一个时刻系统的响应，即系统的时间响应。最后绘制出系统的响应曲线，由此来分析系统的性能。时间响应探究系统对输入和扰动在时域内的瞬态行为，系统特征如上升时间、调节时间、超调和稳态误差都能从时间响应上反映出来。控制系统工具箱提供了对系统阶跃响应、脉冲响应等进行仿真的函数，如表 12.1 所示。

表 12.1　时间响应函数及说明

函　数	函数说明
covar	连续系统对白噪声的方差响应
dcovar	离散系统对白噪声的方差响应
impluse	连续系统的脉冲响应
dimpluse	离散系统的脉冲响应
initial	连续系统的零输入响应
dinitial	离散系统的零输入响应
lsim	连续系统对任意输入的响应
dlism	离散系统对任意输入的响应
step	对连续系统的单位阶跃响应
dstep	对离散系统的单位阶跃响应
filter	数字滤波器

下面详细介绍这些函数的用法。

- impulse：连续系统的脉冲响应。有如下 3 种调用格式：

```
y=impulse(A, B, C, D, iu, T)
```

说明：

用来求得系统对第三个输入的脉冲响应。其中向量 T 为均匀间隔的时间值，指明要计算响应的时间点，y 的列数与输出的个数相同，每列对应一个输出。

```
[x,y]=impulse(A, B, C, D, iu, T)
```

说明：

返回 y 的同时返回状态 x 的变化过程，x 的输出和 y 的格式一样。

```
y=impulse(num, den, t)
```

说明：

用来对传递函数形式进行计算。

- dimpulse：离散系统的脉冲响应。有如下 3 种调用格式：

```
y=dimpulse(A, B, C, D, iu, n)
```

说明：

用来求得系统

$$x(n+1)=Ax(n)+Bu(n)$$
$$y(n)=Cx(n)+Du(n)$$

对第 iu 个输入的脉冲响应。其中整数 n 为要计算响应的点数，y 的列数与输出的个数相同，每列对应一个输出。

```
[y, x]=dimpulse(A, B, C, D, iu, n)
```

说明：

在得到 x 的脉冲响应 y 的同时，返回状态 x 的变化过程。

```
y=dimpulse(num, den, n)
```

说明：

用来对传递函数形式进行计算。

- lsim：连续系统对任意输入的响应。有如下 3 种调用格式：

```
y=lsim(A, B, C, D, U, T)
```

说明：

用来计算出系统对于输入序列 U 的响应，矩阵 U 的每列对应于一个输入，每行对应于一个新的时间点，其行数与 T 的长度相同。

```
[y, x]=lism(A, B, C, D)
```

说明：

在返回输入 x 的响应 y 的同时，返回状态 x 的变化过程。

```
y=lism(num, den, U, T)
```

说明：

用来对传递函数形式进行计算。

- dlsim：离散系统对任意输入的响应。有如下 3 种调用格式：

```
y=lism(A, B, C, D, U)
```

说明：

用来计算出系统

$$x(n+1)=Ax(n)+Bu(n)$$

$$y(n)=Cx(n)+Du(n)$$

对于输入序列的响应每列为一个输出。其中矩阵 U 的每列对应一个输入序列，每行对应一个新时间点。

```
[y,x]=dlsim(A, B, C, D, U)
```

说明：

在返回输入 x 的响应 y 的同时，返回状态 x 的变化过程。

```
y=dlsim(num, den, U)
```

说明：

用来对传递函数形式进行计算。

- step：对连续系统的单位阶跃响应。有如下 2 种调用格式：

```
y=step(A, B, C, D, iu, T)
[y, d]=siep(A, B, C, D, iu, T)
```

各参数的意义参考 impulse。

- dstep：对离散系统的单位阶跃响应

调用方式同 dimpulse 函数。

上述大部分函数都能自动生成时间响应图。

2. 频率响应

研究系统的频率响应是经典控制领域的一个重要组成部分。它的基本原理是，若一个线性系统受到频率为 ω 的正弦信号激励时，它的输出仍然为正弦信号。但是其幅值与输入

信号成 M(ω)比例关系，而且输出与输入信号之间有一个相位差Φ(ω)。M(ω)和Φ(ω)是关于ω的有理函数，这样就可以通过 M(ω)和Φ(ω)来表示系统的特征了。频率响应研究系统的频率行为，从频率响应中可得带宽、增益、转折频率、闭环稳定性等系统特征。控制系统工具箱提供了伯德响应、尼柯尔斯响应、奈奎斯特响应等分析函数，如表 12.2 所示。

表 12.2　频率响应函数及说明

函　数	函数说明
bode	连续系统伯德图
dbode	离散系统伯德图
fbode	连续系统快速伯德图
freqs	拉普拉斯变换
nichols	连续系统的尼柯尔斯曲线
dnichols	离散系统的尼柯尔斯曲线
nyquist	连续系统的奈奎斯特图
dnyquist	离散系统的奈奎斯特图
sigma	连续奇异值频率图
dsigma	离散奇异值频率图
margin	求增益余度和相位余度及对应的转折频率
ngrid	尼柯尔斯方格图

这些函数的具体使用方法如下所示。

- bode：连续系统的伯德图

该函数的调用格式为：

```
[mag, phase]=bode(A, B, C, D, iu, w)
```

说明：

用来求得系统对于第 iu 个输入的频率响应。其中向量 w 包含欲计算响应的频率点(rad)、输出响应的幅度 mag 和相位 phase，它们的列数与输出的个数相同，每列为一个输出的响应。而下面的形式：

```
[mag, phase] =bode(nurn, den, w)
```

是对传递函数模型进行的同样操作。

- dbode：离散系统的伯德图

该函数的调用格式为：

```
[mag, phase]=dbode(A, B, C, D, h, w)
```

其中 A，B，C，D 为离散状态方程的系统阵，而

```
[mag, phase] =dbode(num, den, w)
```

是对传递函数形式进行的相同过程。

- nyquist：连续系统的奈奎斯特图

该函数的调用格式为：

```
[re, im] =nyquist(A, B, C, D, iu, w)
```

用来求得系统对于第 iu 个输入的频率响应。其中向量 w 包含带计算响应的频率点(rad)。输出响应结果为 re 和 im，它们的每列对应一个输出，而用法：

```
[re, im]=nyquist(num, den, w)
```

是对传递函数形式进行的相同过程。

- margin：求增益余度和相位余度及对应的转折频率

该函数的调用格式为：

```
[Gm, Pm, Wcg, Wcp]=margin(mag, phase, w)
```

用来输入参数为伯德响应的幅度 mag、相位 phase 和频率向量 w，返回结果 Gm 为增益余度，Pm 为相位余度，Wcg 和 Wcp 为对应的转折频率。

3. 根轨迹

关于控制系统的根轨迹分析，控制系统工具箱提供了几个函数，用它们可用根轨迹法进行增益的选择，如表 12.3 所示。

表 12.3　根轨迹函数及说明

函数	函数说明
pzmap	零极点映射关系
rlocfind	根轨迹增益检测
rlocus	求根轨迹
sgrid	连续系统的网格根轨迹
zgrid	离散系统的网格根轨迹

下面举例子说明以上函数的用法。

rlocus：求根轨迹。调用的格式为：

```
r=rlocus(num, den, K)
```

说明：

对向量 K 指定的增益计算出 H(s)=1+K*num(s) / den(s)零点根轨迹，r 的每一行对应 K 的一个增益。根轨迹命令 rlocus 自动画出根轨迹图并检测增益向量。

4. 极点配置

控制系统工具箱包含 2 个进行极点配置的函数，如表 12.4 所示。

表 12.4　极点配置函数及说明

函　数	函数说明
acker	用 Ackermann 公式进行极点配置
place	极点配置

函数的具体使用方法如下所示。

- acker：用 Ackermann 公式进行极点配置

调用格式为：

```
K=acker(A, B, P)
```

说明：

用来计算出反馈增益阵 K，使得单输入系统

$$\dot{x} = Ax + Bu$$

在反馈控制律 u=－Kx 下由向量 P 指定极点，即 P=eig(A－B*K)。

- place：极点配置

调用格式为：

```
K=place(A, B, P)
```

说明：

用来计算反馈增益阵 K，使得 A－BK 的特征值为 P 中的元素，而 P 中的复特征值必须共轭成对。

5. 线性二次型调节器和估计器设计

用于线性二次型调节器和估计器设计的函数如表 12.5 所示。

表 12.5　线性二次型调节器如估计器函数及说明

函数	函数说明
LQE	连续系统的线性二次估计器设计
DLQE	离散系统的线性二次估计器设计
LQR	连续系统的线性二次调节器设计
DLQR	离散系统的线性二次调节器设计
LQEW	连续系统线性二次估计器设计
DLQEW	离散系统线性二次估计器设计
LQRY	连续系统线性二次调节器设计
DLQRY	离散系统线性二次调节器设计
LQE2	使用 Schur 法的线性二次估计器设计
LQR2	使用 Schur 法的线性二次调节器设计
LQRD	从连续损耗函数设计离散线性二次调节器
LQED	从连续损耗函数设计离散线性二次估计器

用法举例如下。

- LQE：连续系统的线性二次估计器设计

有如下 3 种调用方式：

用法 1：

```
L=LQE(A, B, C, Q, R)
```

说明：

对于连续系统：

$$\dot{x} = Ax + bu + Gw$$

$$z = Cx + Du + v$$

过程噪声和测量噪声的协方差满足：

$$E\{w\} = E\{v\} = 0, E\{ww'\} = Q, E\{vv'\} = R, E\{wv'\} = 0$$

则用此函数求得增益矩阵 L 使得平稳 Kalman 滤波器

$$\dot{x} = Ax + Bu + L(z - Cx - Du)$$

产生对 L 的 LQE 最优估计。

用法 2：

```
[L, P, E]=LOE(A, G, C, Q, R)
```

说明：

用来求得增益矩阵 L，Riccati 方程的解 P 和估计器闭环特征值 E=EIG(AL*C)。

用法 3：

```
[L, P, E]=LOE(A, B, C, Q, R, N)
```

说明：

用来求当过程噪声和测量噪声满足 $E\{wv'\} = N$ 时系统的估计器问题。

- DLQE：离散系统的线性二次估计器设计

该函数的功能是对于系统：

$$x(n+1) = Ax(n) + Bu(n) + Gw(n)$$

$$z(n) = Cx(n) + Du(n) + v(n)$$

过程噪声和测量噪声的协方差满足：

$$E\{w\} = E\{v\} = 0, E\{ww'\} = Q, E\{vv'\} = R, E\{wv'\} = 0$$

用法 1：

```
L=DLQE(A, B, C, Q, R)
```

说明：

用来求得增益矩阵 L，使得带时间和观测更新的离散平稳 Kalman 滤波器，产生对 x 的 LQG 最优估计。

用法 2：

```
[L, M, P, E]=DLQE(A, G, C, Q, R)
```

说明：

用来求得增益矩阵 L，Riccati 方程的解 M，估计误差的方差为 P=E{[xi-xj][xi-xj]'}(i<>j) 和估计器闭环特征值 E=EIG(A－L*C)。

用法 3：

```
[L, M, P, E] =DLQE(A, G, C, Q, R, N)
```

说明：

用来求当过程噪声和测量噪声满足 $E\{wv'\} = N$ 时离散系统的估计器问题。

- LQR：连续系统的线性二次调节器设计

用法 1：

```
[K, S, E]=LQR(A, B, Q, R)
```

说明：

用来计算最优反馈增益矩阵 K，使得反馈律 u=－Kx 对于状态方程 $\dot{x}=Ax+Bu$ 的目标函数 $J=\int(x'Qx+u'Ru)dt$ 最小。其中 S 是如下代数 Riccati 方程

$$SA+A'S^{-1}-SBRB'S+Q=0$$

的稳态状态解，E 为闭环特征值 E=EIG(A－B*K)。

用法 2：

```
[K, S, E]=LQR(A, B, Q, R, N)
```

说明：

在目标函数中包括 u 到 x 的交叉项 N，此时的目标函数成为：

$$J=\int\{x'Qx+u'Ru+2*x'Nu\}dt$$

- DLQR：离散系统的线性二次调节器设计

用法 1：

```
[K, S, E]=DLQR(A, B, Q, R)
```

说明：

用来计算最优反馈增益矩阵 K，使得反馈律 u=－Kx 对于限制方程

$$x(n+1)=Ax(n)+Bu(n)$$

目标函数 $J=\sum x'Qx+uR'u$ 最小。其中 S 是如下代数 Riccaii 方程 $S-A'SA+A'SB(R+B'SB)BS'A-Q=0$ 的稳态状态解，E 为闭环特征值 E=EIG(A－B*K)。

用法 2：

```
[K, S, E] =DLQR(A, B, Q, R, N)
```

说明：

在目标函数中包括 u 到 x 的交叉项 N，则其目标函数成为：

$$J=\sum\{x'Qx+u'Ru+2*x'Nu\}$$

关于其他函数的详细使用方法，可使用 MATLAB 系统中的 help 命令进行查询。

12.3　信号处理工具箱简介

本节介绍如何使用信号处理工具箱(Signal Processing Toolbox)。MATLAB 信号处理工具箱主要包含以下功能：

- 滤波器的实现与分析。滤波运算和对滤波器进行分析的工具箱函数，包括求取频率响应、零极点分析等。
- 线性系统模型。连续和离散模型及相应的 MATLAB 函数模型描述。
- 滤波器的设计。包含设计 IIR 和 FIR 滤波器的方法。
- 信号变换。工具箱中提供的用于信号变换的传递函数，包括离散傅立叶变换、ChirpZ 变换、离散余弦变换和 Nilbert 变换等。
- 统计信号处理。描述互相关、协方差和谱分析等工具函数并给出典型应用实例。
- 窗口函数。包含工具箱中提供的窗口函数。
- 参数化模型。包含基于系统的时域或频域信息的模型参数的估计方法。

由于篇幅所限，本节不可能对以上内容做全面完整的介绍。本节拟只介绍信号处理和统计信号处理这两个运用最广泛的部分，其他更为专业化的功能请用户在需要时，参考MATLAB 的帮助文件或者更加具体的有关 MATLAB 工具箱的书籍。

12.3.1 信号变换

在信号处理工具箱中有如下几类变换函数：

- 离散傅立叶变换 DFT。即单位圆上的 Z 变换，此变换使离散序列的描述和处理变得更加容易。
- ChirpZ 变换。此变换在沿轮廓线而非单位圆的变换中很有用，特别是计算 prime-length 变换时，ChirpZ 变换比 DFT 算法效率更高。
- 离散余弦变换 DCT。此变换与 DFT 的关系很接近，其能量压缩特性在信号代码应用方面非常有用。
- Hilbert 变换。能简化解析信号的格式。解析信号广泛使用在通讯领域的带通信号处理方面。

1. 离散傅立叶变换

离散傅立叶变换 DFT 是数字信号处理的主要工具。信号处理工具箱的基础是快速傅立叶 FFT，它是对 DFT 进行快速计算的一种有效算法。

工具箱提供 fft 函数和 ifft 函数分别计算离散傅立叶变换及其逆变换。输入序列 x 及其变换 X，它们有下面的关系：

$$X(k+1)=\sum_{n=0}^{N-1}x(n+1)W_N^{kn} \qquad x(n+1)=\frac{1}{N}\sum_{k=0}^{N-1}X(k+1)W_N^{-kn}$$

在这些等式中，下标都是从 1 开始的，而且 $W_N=e^{-j\left(\frac{2\pi}{N}\right)}$。

调用格式为：

```
y=fft(x, n)
```

说明：

对单输入 x 为标量，函数 fft 用来计算输入向量或矩阵的 DFT。若 x 为向量，fft 计算向量的 DFT，若 x 为矩阵，fft 计算每一矩阵列的 DFT。

其中的 n 代表进行变换的点数，即 DFT 的长度。在这种情况下，若输入序列比 n 短，则函数 fft 用 0 填充输入序列，反之则截短输入序列。若 n 值默认，则默认 fft 长度为输入序列的长度。

提示： fft 的执行时间决定于 DFT 的长度 n:

●对任意为 2 的幂数的 n，fft 使用高速 radix－2 算法，这样执行时间最短。

●对任意非 2 的幂数的 n，ffi 使用主因子算法，此算法的速度取决于点数 n

的大小和它具有的主因子的数目。例如，尽管 1013 和 1000 在大小上很接近，但 fft 对长为 1000 的序列进行变换比对长为 1013 的序列变换要快得多。

●对任意质数 n，fft 不能使用 FFT 算法，它直接执行较慢的方法来计算 DFT。

逆离散傅立叶变换函数 ifft 也包含输入序列、输入选项、进行变换的期望点数等参数。其调用的方式和 fft 函数很类似。

【例 1】傅立叶变换

```
t=(0:1/255:1);
x=sin(2*pi*120*t);
y=real(ifft(fft(x)));          %对信号 x 进行傅立叶和逆变换，只取实数部分。
A=(x==y)
A =
  Columns 1 through 12
     1     1     1     1     1     1     1     1     1     1     1     1
.....................
Columns 253 through 256
     1     1     1     1
```

这说明变换后的向量和原来的向量是一样的。

二维傅立叶变换及其逆变换的函数是 fft2 和 ifft2，这些函数用于二维的信号处理或图像处理。有时需将 fft 或 fft2 的输出进行重排以使其零频部分处于序列的中心。使用函数 fftshift 可以将零频部分移到向量或矩阵的中心。

2. ChirpZ变换

ChirpZ 变换(czT)，指在 Z 域内对输入序列沿螺旋轮廓线进行 Z 变换。与 DFT 不同，CZT 并不限制在单位圆上，而可以沿如下描述的轮廓线进行 Z 变换：

$$z_l = Aw^{-l} \quad l = 0,\Lambda\ ,M-1$$

其中 A 是复数起点，W 是描述轮廓线上点间复比率的复数标量，M 是变换的长度。其中一条可能的螺旋线是：

```
A=0.8*exp(j*pi/6)
W=0.995*exp(-j*pi*0.05);
M=91;
z=A*(W.^(-(0:M-1)));
zplane([-2,4/91,2]',z.');
```

画出的图形如图 12.1 所示。

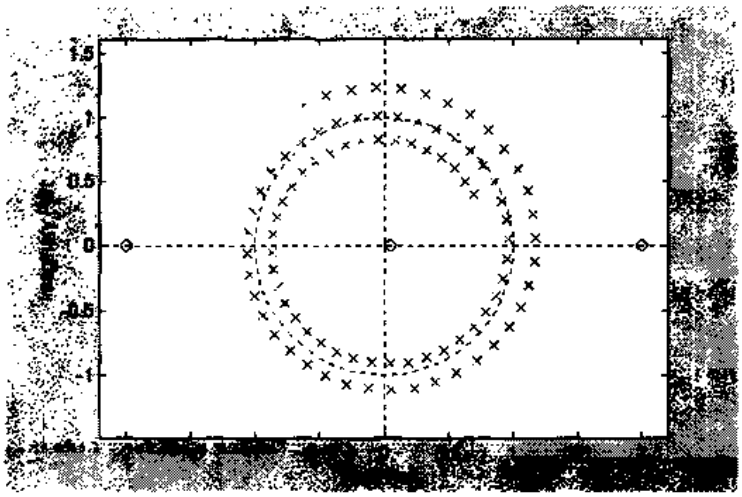


图 12.1 ChirpZ变换

3. 离散因果变换

dct 函数(DCT)计算输入向量或矩阵的离散因果变换。在数学上，输入序列 x 的 DCT 是：

$$y(k)=\sum_{n=1}^{N}2x(n)\cos\frac{\pi}{2N}k(2n+1)\quad k=0,\Lambda\ ,N-1$$

DCT 与离散傅立叶变换很接近，DFT 实际上是计算序列 DCT 的一步。DCT 具有更好的能量压缩性，仅用几个变换系数即可代表序列能量的总体。DCT 的能量压缩性使得它在数据通讯中非常有用。

idct 函数对输入序列计算逆 DCT 变换，从完全或部分的 DCT 系数中重建信号。逆离散因果变换是：

$$x(n)=\sum_{k=1}^{N}\omega(k)y(k)\cos\frac{\pi}{2N}k2(n+1)\quad n=0,\Lambda\ ,N-1$$

因为 DCT 具有能量压缩性，所以仅用部分 DCT 系数来重建信号是可能的。

【例 2】用 DCT 来重建信号

具体步骤如下：

(1) 在本例中先产生一个频率为 10Hz 的正弦信号，其采样频率为 1000Hz。

```
t=(0:1/999:1);
x=sin(2*pi*25*t);
```

(2) 计算此序列的 DCT 并仅用值大于 53 的部分重建信号。

```
y=dct(x);                          %对于信号进行 dct 变换
y2=find(abs(y)<0.053);             %找出 y 中小于 0.053 的部分
y(y2)=zeros(size(y2));             %将 y 中小于 0.053 的部分指定为零
z=idct(y);                         %对于 y 作 idct 变换
```

(3) 画出原始的和经过重新复原的信号的图形。

```
subplot(2,1,1),plot(t,x)
subplot(2,1,2),plot(t,z),axis([0 1 -1 1]);
```

原始信号和重建信号分别如图 12.2 所示。

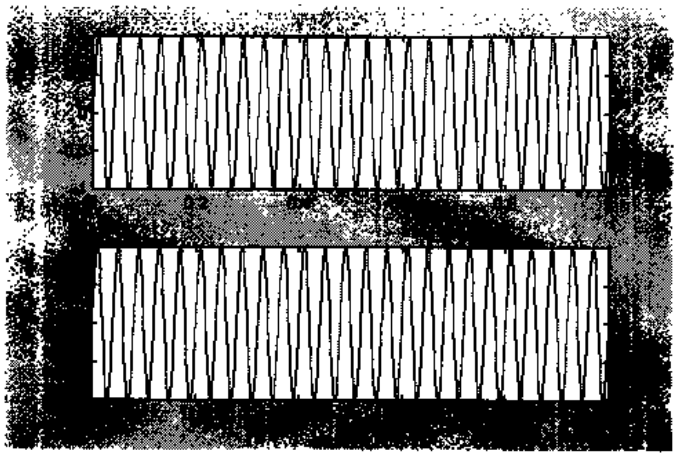


图 12.2　离散因果变换

(4)　量度重建的精度：

```
erro=norm(x-z)/norm(x)
erro =
    0.0120
```

说明：

用原始信号和重建信号之间差值的范数除以原始信号的范数。在这种情况下，重建的相对误差是 0.0120，重建信号大约保留了原始信号 98.8%的能量。

4. Hilbert变换

函数 hilbert 计算实数输入序列 x 的 Hilberi 变换，返回一个同样长度的复数结果。其调用格式为：

```
y=hilbert(x)
```

其中 y 的实部就是原始的实信号，虚部是实际的 Hilbert 变换，y 有时被称为解析信号。离散解析信号的主要特性是，其在单位圆下半部分的 z 变换为零。解析信号的许多应用与此性质有关。

Hilbert 变换与实际信号之间有 90° 的相位移动。

【例 3】Hilbert 变换

本例的目的是画出信号图形(实线)和其 Hilberi 变换图(虚线)，如图 12.3 所示。

```
t=(0:1/1023:1);
x=sin(2*pi*60*t);
y=hilbert(x);
plot(t(1:50),real(y(1:50))),hold on;
plot(t(1:50),imag(y(1:50)),':');hold off;
```

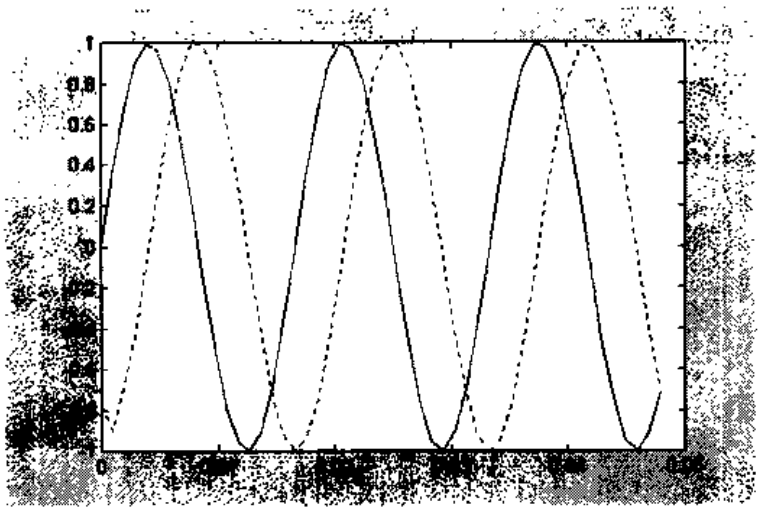


图 12.3 Hilbert变换

12.3.2 统计信号处理

工具箱提供了对随机信号进行处理的函数，特别是能估计离散序列的互相关、协方差序列和谱密度函数。本部分介绍并演示互相关和协方差函数，然后介绍计算信号功率谱的数学函数。

1. 互相关和协方差

- 函数的基本格式

函数 xcorr 和 xcov 分别估计随机过程的互相关和互协方差序列。因为仅能得到无限长随机过程的有限部分，因此在实际应用中必须利用这些序列进行估计。基于 x(n)和 y(n)的 M 个采样值的一个常用的估计是确定其互相关序列：

$$\hat{R}_{xy}(m)=\sum_{n=0}^{M-|m|-1}x(n)y^{*}(n+m)$$

给定长为 M 的输入 x(n)和 y(n)，xcorr 函数用基于 FFT 的算法可估计出上面的和值。

【例 4】互相关函数

```
x=[1 2 5 7]'
y=x+2;
xyc=xcorr(x,y)'      %注意这里进行了转置
xyc =
    21.0000  43.0000  75.0000  109.0000  63.0000  25.0000  9.0000
```

注意： 计算结果的长度是输入序列长度的两倍减 1。

函数 xcov 估计互协方差序列，计算的方法与 xcorr 函数相同，唯一的区别是 xcov 函数在计算时从序列中移去均值。

● 估计的偏差和标准化

若估计量的期望值不等于被估计量的期望值，则它们之间的差称为偏差。xcorr 的输出期望值是：

$$E\{(\hat{R}_{xy}(m)\} = \sum_{n=0}^{M-|m|-1} E\{x(n)y^*(n+m)\} = (M-|m|)R_{xy}(m)$$

若在 xcorr 函数的输入序列之后加上 unbiased 标志，则其输出是无偏估计，语句如下：

```
xcorr(x,y,'unbiased');
```

尽管该格式是无偏的，但因为它仅用几个点计算这些值，因此在位于－(M－1)和(M－1)附近的结束点仍有较大的方差。

解决的简单办法是加上 biased 标志，语句如下：

```
xcorr(x,y,'biased');
```

用这一方法，只有在零滞后的互相关样本中才是无偏的。这一估计通常比无偏估计更理想，因为它避免了互相关序列在结束点随机的、比较大的变化。

xcorr 函数还提供一个输出标准化的方法，语句如下：

```
xcorr(x,y,'coeff');
```

该格式用 norm(x)*norm(y)去除输出，使得对零滞后的样本其互相关是 1。

● 多通道信号

对多通道信号，函数 xcorr 和 xcov 同时估计所有通道的互相关和互协方差序列。若 S 是一个 M×N 信号矩阵，有 N 个通道，则 xcorr(S)返回一个(2M－1)×N^2矩阵，其 N^2列是 N 个通道的互协方差。例如，若 S 是一个三通道信号

```
s=[s1  s2  s3]
```

则 xcorr(S)的结果照如下的形式安排：

```
R=[Rs1s1  Rs1s2  Rs1s3  Rs2s1  Rs2s2  Rs2s3  Rs3s1  Rs3s2  Rs3s3]
```

另两个相关的函数 cov 和 corrcoef 分别估计同通道之间的协方差和标准化协方差，并将它们放置在一个方阵中。这两个函数在前面的数值计算中已经介绍过。

2. 谱密度函数

谱分析是基于有限数据来估计的信号、随机过程或系统输出的频率。功率谱估计在很多应用中十分有用，如检测淹没在宽带噪声中的信号等。

稳定随机过程的功率谱密度(PSD)通过离散傅立叶变换与自相关序列建立关系：

$$P_{xx}(\omega) = \sum_{m=-\infty}^{\infty} R_{xx}(m)e^{-j\omega m}$$

此频率函数具有这样的特性，其在全频率段的积分等于信号在此频率段的功率。PSD 是互谱密度(CSD)函数的特殊情况。

许多估计 PSD 的不同方法都可以归入参数化或非参数化类。信号处理工具箱的基本技术是 Welch 发展的非参数化方法，用函数 Psd、csd、tfe 及 cohere 等来实现。

● Welch 方法

估计过程功率谱的一个方法是求出过程样本的离散傅立叶变换，并将结果幅值平方。

【例 5】本例中信号 xn 有 1001 个元素，包括 2 个正弦信号和噪声

首先用下面的语句产生输入含噪声的信号：

```
Fs=1000;                                    %设置采样频率
t=0:1/Fs:1;
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(size(t));
Pxx=abs(fft(xn,1024)).^2/1001;        %xn 的粗略 PSD 估计
```

此估计也称为周期图法，它用数据窗的范数来标称 FFT 的平方幅值(此处是长度为 1001 的矩形窗)来保证估计是渐进无偏的。也就是说，当采样点的数目增加时，周期图的期望值接近 PSD 真值。用周期图法估计的缺点是其协方差较大，且当采样点数增加时协方差并不减小。

下面这个例子可以说明当 FFT 长度增加时，周期图不会变得平滑。

```
Pxx_short=abs(fft(xn,256)).^2/256;
subplot(2,1,1);
plot((0:255)/256*Fs10*log10(Pxx_short));
title('Short PSD');
subplot(2,1,2);
plot((0:1023)/1024*Fs,10*log10(Pxx));
title('Long PSD');
```

得到的短周期图和整段数据的周期图如图 12.4 所示。

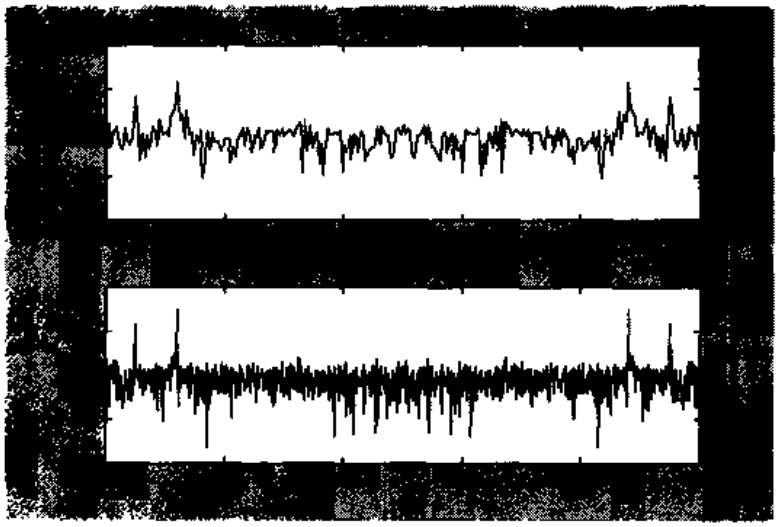


图 12.4　Welch方法求功率谱

减小 PSD 估计误差的方法有好几种，一是将信号分段进行估计，而且所分的段数越多，所估计的值就越精确；另一个方法是在计算周期图之前，对数据分段加非矩形窗，形成修正周期图法。

- 功率谱密度函数

下面用 psd 函数在默认 FFT 长度(256)、窗(长 256 的海明窗)、混迭样本(元)条件下对序列 xn 做 PSD 估计，其语句为：

```
Pxx=psd(xn);
```

提示： 若原序列调 n 的单位为 Hz，则 Pxx 的单位为 V^2/Hz。

【例 6】下面精确地重建例 5，混迭样本数为 128

```
Fs=1000;                    %采样频率
t=0:1/Fs:1;
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(size(t));
nfft=256;
window=hanning(256);
noverlap=128;
dflag='none';
Pxx=psd(xn,nfft,Fs,window,noverlap,dflag);
```

除了 dflag 字串可放在任何地方以外，psd 函数输入参数的顺序是很重要的。采样频率不影响 PSD 估计，但它影响 PSD 在画图时的频率轴，不加输出变量的 PSD 函数可直接画图：

```
Pxx=psd(xn,nfft,Fs,window,noverlap,dflag);
```

得到的图形如图 12.5 所示。

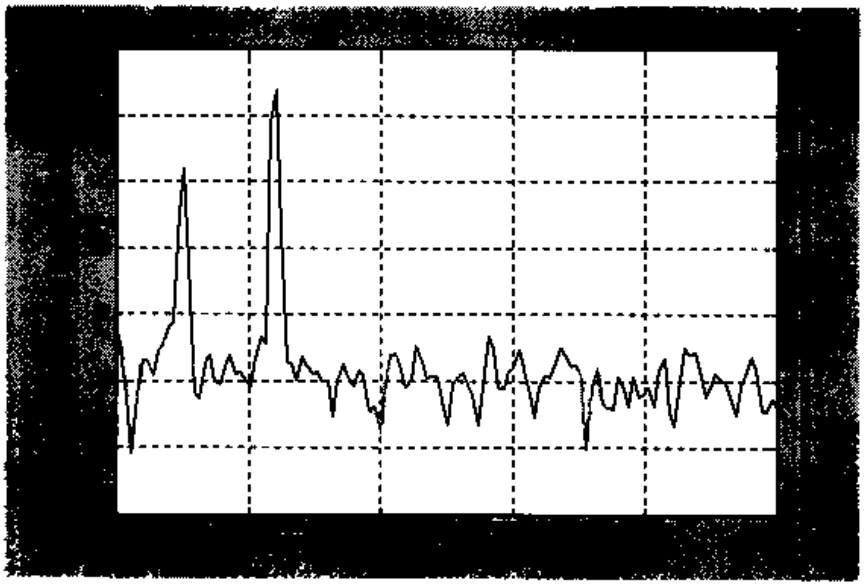


图 12.5　PSD函数求功率谱密度

若要自己生成 PSD 图形，可通过一个附加输出变量得到频率向量：

```
[Pxx,f]=psd(xn,nfft,Fs,window,noverlap,dflag);
```

所得图形与图 12.5 一样。

因为 xn 是实信号，PSF 函数仅返回从 0 到奈氏频率的频率。

- 互谱密度函数

函数 csd 使用 Welch 方法估计两个等长信号 x 和 y 的互谱密度，其基本格式为：

```
Pxy=csd(x, y, nfft, Fs, window, noverlap, dflag);
```

说明：

各参数的含义和功率谱密度函数 PSD 一样，使用时的基本方法也与 PSD 相同。其中最后一项 dflag 可以放在任何位置，其值是个字符串，一般为 none。

- 置信区间

psd 和 csd 函数均能计算置信区间，在输入参数项中增加 p 来说明置信区的百分比。它们具体的调用格式为：

```
[Pxx, pxxc, f]=psd(x, nfft, Fs, window, noverlap, p, dflag)
[pxy, pxyc, f]=csd(x, y, nfft, Fs, window, noverlap, p, dflag)
```

其中的参数 p 必须是 0～1 之间的一个标量。

- 传递函数估计

Welch 方法的一个应用是非参数化系统辨识。假定 H 是一个线性时不变系统，x(n)和 y(n)分别是 H 的输入和输出，则 x(n)的 PSD 与 x(n)和 y(n)的 CSD 有如下关系：

$$P_{xy}(\omega) = H(\omega)P_{xx}(\omega)$$

x(n)和 y(n)间传递函数的一个估计是

$$\hat{H}(\omega) = \frac{\hat{P}_{xy}(\omega)}{\hat{P}_{xx}(\omega)}$$

这一方法可用来估计幅值和相位信息。tfe 函数使用 Welch 方法计算 CSD 和 PSD，然后求得它们的传递函数的估计，使用 tfe 的方法与 csd 的使用方法一样。

【例 7】本例用来实现 FIR 滤波器对 x(n)进行滤波，然后画出实际的幅值响应和估计响应图

```
Fs=1000;                              %采样频率
t=0:1/Fs:1;
xn=sin(2*pi*50*t)+2*sin(2*pi*120*t)+randn(size(t));
h=ones(1,10)/10;
yn=filter(h,1,xn);
[HEST,f]=tfe(xn,yn,256,Fs,256,128,'none');      %用函数 tfe 估计的响应
H=freqz(h,1,f,Fs);                    %求出 x 的实际响应
subplot(1,2,1);
plot(f,abs(H));
title('实际的响应');
subplot(1,2,2);
plot(f,abs(HEST));
title('估计的响应');
```

图形如图 12.6 所示。

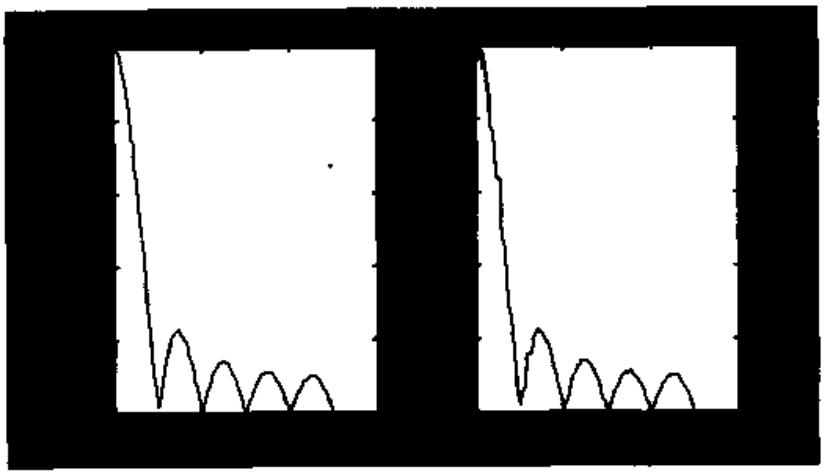


图 12.6　传递函数的估计

12.4　优化工具箱简介

优化工具箱(Optimization　Toolbox)涉及函数的最小化或最大化问题，也就是函数的极值问题。MATLAB 的优化工具箱由一些对普通非线性函数求解最小化或最大化极值的函数和解决诸如线性规划等标准矩阵问题的函数组成。

12.4.1　基本函数简介

优化工具箱中求非线性函数极小值的函数如表 12.6 所示。其中有些函数是 MATLAB 内核包中自带的函数，在前面的数值计算一章中已经做了介绍，下面就不再详细讲解了。如果用户还不太了解，可以参看 MATLAB 的帮助信息。

表 12.6　求函数极小值的函数

类　型	含　义	调用格式
无条件标量问题	$\min_x f(x)$，其中 x 为标量	x=fmin('f',x)
无限定条件矩阵问题	$\min_x f(X)$，其中 X 为矩阵	x=fminu('f',x)
有限定条件问题	$\min_x f(X)$，条件为 $G(X)\le 0$	x=constr('fg',x)
目标条件问题	$\min_x \gamma$，条件为 $F(X)-W'\le goal$	x=attgoal('f',x,goal,w)
最小最大极值	$\min_x \{\max F(X)\}$，条件为 $G(X)\le 0$	x=minimax('fg',x)
非线性二次平方极值	$\min\sum(F(X)\times F(X))$	x=leastsq('f',x)
非线性方程	$F(x)=0$	x=fsolvex('f',x)
半无穷条件	$\min_x f(X)$，条件为 $\phi(X,\omega)\le 0,\forall\omega$	x=seminf('ft',n,x)

问题的原理算法是 Nelder-Mead 单纯形搜索方法和 BFGS 拟牛顿(qusi-Newton)方法。限定条件下的最小、最大最小、目标法和半无穷优化等问题，所用的原理算法是二次规划法。非线性二次平方问题的原理算法是 Gauss-Newton 法和 Levenberg-Marquardt 法。非线性最小和非线性二次平方问题，可进行线性搜索策略的选择，线性搜索策略使用的是三次或四次内插和外插方法。

优化工具箱还能解决几类求矩阵的极小值问题，此时仅需要将相应的系数矩阵和向量传递到函数中。MATLAB 系统能解决的矩阵问题如表 12.7 所示。

表 12.7　求解矩阵问题的函数

类　型	含　义	语　法
非负二次平方问题	$\min_x \|Ax-b\|^2$，条件为 $x>0$	x=nnls(A,b)
二次问题	$\min_x\left(\frac{1}{2}x^THx-c^Tx\right)$，条件为 $Ax\le b$	x=qp(H,c,A,b)
线性规划问题	$\min_x(f^Tx)$，条件为 $Ax\le b$	x=lp(f,A,b)

12.4.2　函数功能举例

下面举几个例子来说明上述函数的用法。

1.　无限定条件的极值问题

【例 8】无限定条件的极值问题

考虑如下的问题：求合适的集合[x1,x2]，使问题 1

$$\min_x f(x)=e^{x1}\left(4x_1^2+2x_2^2+4x_1x_2+2x_2+1\right)$$

成立。

为求解此问题，可先编写一个能返回函数值的 M 文件，将函数表达式写入 MATLAB 系统中，然后调用非限定最小程序 fminu。具体的步骤如下：

(1) 利用文件编辑器编写 M 文件。

```
function f=fun(x)
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

程序中 function　f=fun(x)表示将此 M 文件定义为名为 fun.m 的文件。

(2) 在命令窗口中调用优化程序 fminu。

```
x=[-1,1];          %估计初值
x=fminu('fun',x)
```

经过 36 次函数调用后，得到极值点为：

```
x =
0.5000   -1.0000
```

在命令窗口中接着输入 fun(x)得到极值点处的函数值为：

```
ans =
  1.3029e-010
```

注意： 当函数存在一个以上的局部最小点时，[x1,x2]的初值会影响迭代的次数和解的值，本例中的初值为[－1,1]。

说明：

为了改变优化解的特征，可在函数 fminu 的说明参数中加入可选的参数说明(options)，语句如下：

```
x=fminu('fun',x,options)
```

options 是一个包含允许误差值和算法选择等选项的向量。它的第一个元素所起的作用是控制在优化周期中显示输出的数量，将它设置为 1 时会以表格形式显示函数值和收敛信息。options 的第二和第三个元素建立终止判据。options 中的其他元素用于选择算法和设置迭代的最大次数等。关于 options 的具体使用方法，以后会有详细的说明。

2. 限定条件极值问题

【例 9】限定条件极值问题

使用 constr 函数，可在上面例子中加入不等式限定条件。如对问题 2

$$\min_{x} f(x) = e^{x1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

限定条件为：

$$1.5 + x_1x_2 - x_1 - x_2 \le 0$$
$$-x_1x_2 - 10 \le 0$$

将上面编写的 M 文件进行修改，使其能返回目标函数值和限定函数值下的极值函数 constr 求出极值。具体过程如下：

(1) 利用文件编辑器编写 M 文件

```
function [f,g]=fun(x)
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
g(1)=1.5+x(1)*x(2)-x(1)-x(2);
g(2)=-x(1)*x(2)-10;
```

(2) 在命令窗口中调用优化程序

```
x=[-1,1];          %估计初值
x=constr('fun',x)
```

得到的结果为：

```
x=
    -9.5474    1.0474
```

在命令窗口中接着输入

```
[f,g]=fun(x)
```

得到的结果是在极值点处的函数值和限定条件的值

```
f =
```

```
    0.0236
g =
  1.0e-015 *
  -0.8882
    0
```

3. 下界条件与上界条件

使用函数的有界语法可将变量 x 限定在某一区域之内。例如对于函数 constr，下面的语句

```
x=constr('fun',x,options,vlb,vub);
```

可以将 x 限定在 vlb<x<vub 范围之内。

【例 10】下界条件与上界条件

如在例 2 中，要将 x 限定为大于 0，可在命令窗口中使用如下的命令：

```
x=[-1,1];              %初值
options=[];            %使用默认选项
vlb=[0,0];             %下界
vub=[];                %没有上界
x=constr('fun',x,options,vlb,vub)
```

经过七次迭代后，问题的解为：

```
x =
    0    1.5000
```

在 MATLAB 命令行窗口中输入如下内容：

```
[f,g]=fun(x)
```

得到的结果是在极值点处的函数值和限定条件的值：

```
f =
   8.5000
g =
    0
   -10
```

在上面的例子中，x 没有上界，因此 vub 被置为空矩阵。也可以使用如下形式的命令省掉上界，其调用格式为：

```
constr('fun',x,options,vlb,)
```

当 vlb 或 vub 的元素数目 n 比向量 x 的元素数目少时，则 x 中只有前 n 个元素被限定有界。另外，上界和下界也可以用线性不等式表示。当仅有几个元素有界时，用这样的方法是比较方便的。具体的调用格式为：

上界：$x_i \le U_s$，表示为：$x_i - U_s \le 0$

下界：$x_i \le L_s$，表示为：$-x_i + L_s \le 0$

4. 梯度计算

一般说来，最小化程序都要使用由有限微分近似法计算出的梯度，这一过程会自动作用到每个变量以计算函数和限定条件的偏导数。若所编写的程序能返回函数和限定条件的偏导数，则问题的求解将会更精确和有效。

【例 11】本例用梯度来解决前面的问题 2

具体过程如下：

(1) 利用文件编辑器为目标函数编写 M 文件。

```
function [f,g]=fun(x)
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
g(1)=1.5+x(1)*x(2)-x(1)-x(2);
g(2)=-x(1)*x(2)-10;
```

(2) 编写求梯度的 M 文件。

```
function [df,dg]=grad(x)
t=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
df=[t+4*exp(x(1))*(2*x(1)+x(2)),
4*exp(x(1))*(x(1)+x(2)+0.5)];          %定义目标函数的梯度函数
dg=[x(2)-1,-x(2)
   x(1)-1,-x(1)];                      %定义限定函数的梯度
```

(3) 在命令窗口中调用函数求限定条件下的极值问题。

```
x=[-1,1]              %初始估值
options=[]        %使用默认选项
vlb=[];           %没有下界
vub=[];           %没有上界
x=constr('fun',x,options,vlb,vub,'grad')
```

说明：

其中变量 df 包含目标函数 fun(x)对于 x 中每一个元的偏导数，变量 dg 的列向量包含对每个限定条件的偏导数，例如第 j 列是第 i 个限定条件对向量 x 的偏导数，上面的例子中变量 dg 为：

$$\begin{bmatrix} \frac{dg_1}{dx_1} & \frac{dg_2}{dx_1} \\ \frac{dg_1}{dx_2} & \frac{dg_2}{dx_2} \end{bmatrix}$$

变量 vlb 和 vub 分别给出自变量 x 的上界和下界。经过 11 次函数计算和梯度计算后，得到的极值点为

```
x =
-9.5474    1.0474
```

在命令窗口中继续输入

```
[f,g]=fun(x)
```

得到极值点处的函数值和限定条件值为：

```
f=
    0.0236
g =
  1.0e-014 *
    0.1110
   -0.1776
```

5. 梯度检查

当进行梯度计算时，可以将它与由有限微分法计算的结果进行比较，这对于检查目标函数或者梯度函数公式中的错误非常有用。

若要进行这样的检测，options(9)要赋值为 1，在优化检测的第一个周期中会自动检查梯度。若它们与给定的允许误差不匹配，将会出现警告信息，指出偏差大小并给出是放弃优化还是继续优化的选择项。

提示： 大于零的限定条件：优化工具箱只使用形如 g(x)<0 的限定条件，对于大于零的限定条件用小于等于零的限定条件乘以 -1 来处理。例如，形式 g(x)>0 的限定条件等价于 -g(x)<0。

6. 等式限定条件

等式限定条件式要放在矩阵 g 的前面位置，此时 options(13)要赋初值为条件等式的个数。

【例 12】在例 2 中加入 x1+x2=0 的限定条件，需要采用如下的步骤：

(1) 利用文件编辑器为目标函数编写 M 文件。

```
function f=fun(x)
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
g(1)=x(1)+x(2);                    %等式限定条件
g(2)=1.5+x(1)*x(2)-x(1)-x(2);      %不等式限定条件
g(3)=-x(1)*x(2)-10;                %不等式限定条件
```

(2) 在命令窗口中调用限定条件下的最优化程序。

```
x=[-1,1]                    %初始估值
options(13)=1;      %指出在限定条件中有一个是等式限定条件
x=constr('fun',x,options);
```

经过 11 次迭代后，得出的解为：

```
x =
   -9.5474    1.0474
```

在命令窗口中继续输入

```
[f,g]=fun(x)
```

得到的极值点处的函数值及限定条件值为：

```
f =
```

```
    0.0236
g =
  1.0e-014 *
    0.1110
   -0.1776
```

12.4.3　优化参数的设置

参数说明向量 options 共有 18 个元素，包含了在优化程序中需要用到的参数。若在对某优化程序的第一次调用中，向量 options 为空向量，则会自动产生并使用一组默认的参数。若向量 options 的元素为零，则这些元素也被赋值为默认值。若 options 被定义但其元素个数小于 18，则其他的元素将会被赋值为默认值。

options 中的一些参数根据问题大小的情况而确定。一些参数由在进行优化时的收敛性计算而得到；而另外一些参数要依据对优化程序所做的说明。向量 options 中的所有说明参数如表 12.8 所示。

表 12.8　参数的默认值和其功能

序号	功能	默认值	功能说明
1	显示	0	在优化周期中控制输出的数目。0 不显示输出；1 按表格输出结果；−1 隐藏警告信息
2	对 x 终止	1e−4	自变量 x 的最低精度的终止判据，当所有的终止判据都满足时，优化将会终止
3	对 f 终止	1e−4	在求解时对目标函数 f 所要求的精度的终止判据
4	对 g 终止	1e−7	由函数 attgoal、constr、minimax 和 seminf 使用的终止判据，也就是在最坏情况下限定条件精度的量度
5	主要算法	0	选择主要优化算法
6	SD 算法	0	选择搜索方向算法
7	搜索算法	0	选择线性搜索算法
8	函数		在最后极值点的函数值，对函数 attgoal 和 minirnax 而言它包含一个到达因子
9	梯度检验	0	当置为 1 时，在最初的几个迭代周期，梯度将与由有限微分计算的结果比较，此时梯度函数必须存在
10	函数计数		函数计算计数器
11	梯度计数		函数梯度计算或有限微分梯度计算的总数
12	限定计数		限定梯度计算或有限微分梯度计算的总数
13	等式限定条件	0	等式限定条件的数目。等式限定条件必须放置在变量 g 的前几个元素中

续表12.8

序号	功能	默认值	功能说明
14	最大迭代次数	100n	迭代的最大次数，此值被置为 n 的 100 倍，其中 n 为自变量的数目。在函数 fmins 中，默认值为 200 n，在函数 fmin 中此默认值为 500n
15	使用的目标	0	尽可能接近 goals 的目标数，由函数 attgoal 使用
16	最小摄动	1e−8	在有限微分梯度计算中变量的最小变化。对梯度计算而言，实际使用的摄动将会自动调整以提高精度。它将在最小摄动和最大摄动之间变换
17	最大摄动	0.1	在有限微分梯度计算中变量的最大变化
18	步长		步长参数。一般来说在第一次迭代中它被保守地赋值为 1 或更小，这取决于导数的情况。

12.4.4 常见问题及推荐的解决办法

优化问题可能要经过很多次迭代才能收敛，也可能对诸如有限微分梯度计算中的截短或截断误差很敏感。大部分优化问题都能得益于有一个好的初始估值，这能提高执行的效率并有利于找到全局最小而非局部最小。

优化工具箱可被应用于解决大量的问题。利用此工具箱，大部分与优化技术有关的局限性都能被克服。另外，非标准形的问题经过合适的变换后也能通过本工具箱求解。下面是一些常见的问题和推荐的解决办法。

1. 解不是全局最小

解决方法：

除非问题是连续的且只有一个最小值，否则存在全局最小这一点并不能得到保证。从不同的初始值开始进行优化能确定全局最小，或验证只有一个最小值。在可能的情况下可使用不同的方法来验证结果。

2. 函数fminu给出警告信息，并且在极值附近表现出较慢的收敛速度

解决方法：

若没有提供梯度，并且中断判据是严格的，函数 fminu 在极值点附近常常会因为在梯度计算中的截断误差而表现出较慢的收敛速度。放松中断条件则可以加快收敛速度，随之而来的是解的精度会降低。另外改变有限微分干扰水平 options(16：17)可能会提高梯度计算的精度。

3. 对得到的极值点不能计算函数f或g的值

解决方法：

当不能计算函数值时，可对自变量加上范围条件或者使用罚函数对 f 和 g 给一个较大的值。对梯度计算而言，罚函数必须是光滑连续的。

4. 欲进行极值计算的函数是不连续的

解决方法：

求极值的基本方法需要依赖连续函数的一阶和二阶导数。当极值点不在间断点附近，或者有限微分参数被调整以跳过不连续的点时，对某些类型的不连续情况也能得到正确的计算结果。参数 options(16)和 options(17)控制有限微分梯度计算中变量调的扰动水平。扰动总是在如下的范围之内：

```
options(16)<x<options(17)
```

5. 显示警告信息

解决方法：

当中断判据过于严格或者极值问题对自变量调的变化特别敏感时可能会发生此种情况。它通常会指出有限微分梯度计算中的截断误差或者多项式内插运算中的问题。这些问题通常可以忽略，然而，这种情况下的收敛将会比正常情况下花费更多的时间。

第13章
综合实例

本章要点：

本章将全面介绍两个通过 MATLAB 编程的例子，为用户进一步灵活使用 MATLAB 解决实际问题提供基础。

本章具体包括以下内容：

- 例 1：用传递矩阵法解扭转振动
- 例 2：温度转换器

13.1 用传递矩阵法解扭转振动

本节将介绍一个在机械工程中运用的一种振动方程的解法。该振动系统如图 13.1 所示。通过这个实例，用户可以看到 MATLAB 在工程中应用的方便和快捷。

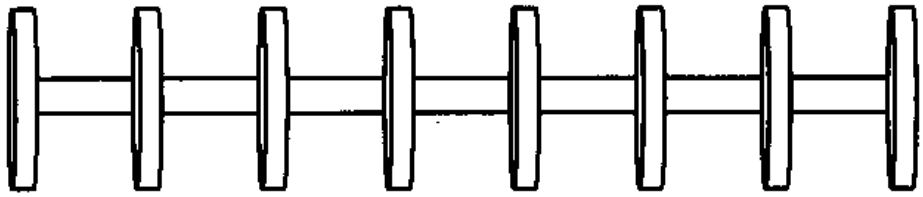


图 13.1 扭转振动系统

13.1.1 问题的工程背景

工程上有些结构是由很多单元一环一环地结合而成的，呈一种链状结构的形式，如连续梁、汽轮发动机转轴、发动机曲轴等。这些结构可以离散化为轴上带有集中质量的横向振动系统或轴上带有圆盘的扭转振动系统。对这种多自由度系统进行振动分析时，可以采用一种很有效的方法—— 传递矩阵法。

用传递矩阵法进行振动分析时，只需对一些阶次很低的传递矩阵进行连续的矩阵乘法运算。在数值求解时只需计算低阶次的传递矩阵和行列式值，这就大大节省了计算工作量。不管离散化以后的系统自由度数有多少，也不管系统有没有一些中间支座条件，这些对传递矩阵的阶数都没有什么影响。传递矩阵的阶数只取决于构成结构的单元的性质。描写单元运动的微分方程为四阶，轴单元扭转振动微分方程式为二阶，则相应的传递矩阵也为四阶或二阶。

13.1.2 算法分析

对于该结构的具体分析，可以在任何一本振动理论的教科书中找到，本书不再赘述。用户如果有兴趣可以参考清华大学出版社出版的《机械振动》。

根据分析，可以得到如下的算法。

假定系统的振动频率为 p，各盘的转动惯量为 I_i，轴的刚度为 K_i，则有矩阵：

$$\begin{bmatrix} 1 & 1/K_i \\ -p^2 I & 1-\dfrac{p^2 I}{K} \end{bmatrix}_i$$

由于该矩阵将第 $i-1$ 个点的状态矢量“传递”给了第 i 个点的状态矢量，称之为第 i 段的传递矩阵。该矩阵满足以下关系：

$$\begin{bmatrix}\theta\\M\end{bmatrix}_i^L=\begin{bmatrix}1 & 1/K_i\\ -p^2I & 1-\dfrac{p^2I}{K}\end{bmatrix}_i\begin{bmatrix}\theta\\M\end{bmatrix}_{i-1}^R$$

将所有的传递矩阵乘起来就可以得到这个结构由第一个节点到最后一个节点的传递矩阵 H。只要再满足已知的边界条件就可以求出轴盘扭转振动系统的固有频率及主振型了。而这些边界条件就是已知的第一节点和最后一个节点的某些状态量。它们有可能是转矩、位移和速度等。注意，现在有了传递矩阵 H，它把所有第一个节点的状态矢量传递到最后一个节点的状态矢量，这就为解决问题提供了出路。通过这种传递关系可以建立一个矩阵方程：

$$\begin{bmatrix}\theta\\M\end{bmatrix}_1^L=H*\begin{bmatrix}\theta\\M\end{bmatrix}_M^R$$

其中的未知量只有该系统的固有频率 P。

从本质上说，这个方程是个一元 n 次的方程，用一般的方法解这样的方程几乎是不可能的。因此我们假设了一个振动频率 P，看看它是否能够满足该方程。但是一般来说，P 并不是系统的固有频率。我们可以在一个有限的区间内查找，直到找到固有频率，或者查找到的值能以一定的精度满足该方程。还有一个值得注意的问题是系统的固有频率不止一个，一般该系统有几个自由度就有几个固有频率，因此必须有一定的判断条件来找到问题所有的解。

当然我们可以用固定的步长来解决这个问题。但从算法的角度来看，如果步长太大所得的解就会很粗糙。而且有可能两个固有频率很接近，以致步长太大而丢失几个固有频率。但是如果步长太小(一般为了不丢失解，总是取很小的步长)，这样的开销太大。因为实际中轴的个数会很多(也就是系统的自由度很大，其固有频率很多)，以致问题的计算量大得无法预知。尤其是当 n 大于 5 时，计算所耗费的时间会很长。比如本节这道简单的例题，如果采用 1e−6 的步长在奔腾 200 的机器上求解需要 0.5 小时。

因此在这里采用了二分法，以一个比较大的步长搜索解的区间，当两次得到的值正负相间，可以确定这个区间中有解。发现解的区间后，将步长折半，继续在此区间中搜索。直到满足给定的精度或者找到固有频率。

13.1.3　算法的实现

传递矩阵法主程序：

```
function [P,An,Yn]=MatriT2(I,K,h,ee)
//写一段好的 HELP 可以让使用程序的人更好地理解和使用它
//注意格式
%%This Function use the transmit Matrix method to
%%solve the multi-dimension vibrancy problem.
%%The declaration of the Variance
```

```
%%Input :I K h ee
%%I the rotate inertia of every object
%%K the rigidity of every axes
%%h the step
%%ee the tolerate
%%Output :P Yn An
%%P the Inhere Frequency of the Vibrancy System
%%An the Vibrancy Format of the Vibrancy System
%%format [P,An,Yn]=MatriT2(I,K,h,ee)
  format long;                %定义长整形格式
  n=length(I);                %获取矩阵的维数
  l=0;
  u=0;

  for(p=0:h:2);                    %以 h 为步长在(0, 2)区间内搜索解
    l=l+1;
    H=[1 0
     -p^2*I(i) 1 ];
    flag=0;

    for(j=2:n)
      Hi=[1 1/K(j)
        -p^2*I(j) 1-I(j)*p^2/K(j)];  %场矩阵
      H=Hi*H;
    end                                %求出传递矩阵 H

    M=[1
      0];                              %最左端的振动参数(边界条件)

    Mn=H*M;
    y=Mn(2,1);
    Y(l)=y;          %用传递矩阵计算最右端的振动参数(另一个边界条件)

     if Y(l)==eps;
       u=u+1;
       P(u)=p;
       flag=1;
     end;   %如果右端恰好为自由(满足边界条件)则记录下此时的 P
```

```
if l>1
   if Y(l-1)*Y(l)<eps;                    %如果该区间之间有根
       flag=0;
       p1=p-h;
       p2=p;
       y1=Y(l-1);
       y2=Y(l);                           %二分法的初始参数

       while(flag==0)
         p3=(p1+p2)/2;        %计算新的 P 值
          H=[1 0
             -p3^2*I(1) 1 ];
          for(j=2:n)
            Hi=[1 1/K(j)
               -p3^2*I(j) 1-I(j)*p3^2/K(j)];
            H=Hi*H;
          end                             %求出传递矩阵 H
         M=[1 0]';
         Mn=H*M;
         y3=Mn(2,1);          %传递矩阵法求出此时对应的值
         if y3*y1<eps;
           y2=y3;
           p2=p3;
           flag=0;
         end                             %若根在 y1,y3 之间则 y2=y3
         if y3*y2<eps;
           y1=y3;
           p1=p3;
           flag=0;
         end                             %若根在 y2,y3 之间则 y1=y3
         if abs(y3)<=eps|abs(p1-p2)<ee
%若 y3 的值为 0 或 p1，p2 之间的距离小于给定的误差，说明 p 已经达到预定的精度
           u=u+1;
           P(u)=p3;
           flag=1;
           H=[1 0
            -p3^2*I(1) 1 ];
           M=[1
             0];
```

```
                M=H*M;
                An(1,u)=M(1,1);
                  for(j=2:n)
                    Hi=[1 1/K(j)
                      -p3^2*I(j) 1-I(j)*p3^2/K(j)];
                     M=Hi*M;
                      An(j,u)=M(1,1);
                    end
                    Yn(u)=M(2,1);          %计算出此时各阶的主振型
              end
            end
          end
        end
      end
x=0:h:2;
grid,plot(x,Y);                            %画出 x，Y 的图像
```

13.1.4 计算及结果

在本例中使用了很多的 MATLAB 的矩阵运算函数，从而大大减少了编程的工作量。这样可以把更多的精力放到算法的改进和精度的提高上来，这正体现了 MATLAB 的优越性。在以后的例题中会用到更多的专业函数和工具箱，用户将会更深刻地体会到这一点。

在 Medit 的窗口中输入并调试该段函数，准确无误后以文件名 matriT2.m 保存在自定义的工作目录下，就可以在 MATLAB 命令行窗口中运用该程序了。

在 MATLAB 命令行窗口，先定义要求输入的变量：

```
I=[1 1 1 1 1 1 1 1];              %系统的转动惯量矩阵
K=[1 1 1 1 1 1 1 1];              %系统的转动刚度矩阵
eps=10-6;                       %设置算法的计算精度
h=0.01;                           %设置初始的搜索精度
[P,An,Yn]=MatriT2(I,K,h,eps);     %调用计算程序
```

得到的结果如表 13.1 和表 13.2 所示。

表 13.1 输出的结果1

p1	2	3	4
0.00003906250000	0.39018064498901	0.76536686897278	1.11114047050476
An			
1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
0.99999999847412	0.84775906427596	0.41421355587881	−0.23463314519354

续表13.1

An			
0.99999999542236	0.56645449533800	−0.41421357424743	−1.17958043237013
0.99999999084473	0.19891236398470	−1.00000000760853	−0.66817862032083
0.99999998474121	−0.19891237178871	−0.99999999239147	0.66817866328626
0.99999997711182	−0.56645450195392	−0.41421353751018	1.17958042228901
0.99999996795654	−0.84775906869657	0.41421359261605	0.23463309451236
0.99999995727539	−1.00000000155231	1.00000001521707	−1.00000002870858
Yn　1.0e−006 *			
−0.01220703105442	0.00310462761055	−0.03043413210335	0.05741716946339

表 13.2　输出结果2

p5	6	7	8
1.41421357154846	1.66293923377991	1.84775906562805	1.96157055854797
An			
1.00000000000000	1.00000000000000	1.00000000000000	1.0000000000000
−1.00000002595186	−1.76536689524451	−2.41421356461065	−2.84775905616221
−0.99999997404814	0.35115337958072	2.41421357093942	4.26197258579166
1.00000005190371	1.49660572336020	−1.00000001527899	−5.02733938634940
0.99999994809628	−1.49660585547407	−0.99999998472101	5.02733929373640
−1.00000007785557	−0.35115314635126	2.41421355828189	−4.26197232205216
−0.99999992214442	1.76536684885227	−2.41421357726819	2.84775866144815
1.00000010380742	−1.00000019772238	1.00000003055798	−0.99999953440303
Yn 1.0e−006 *			
−0.20761486063670	0.39544477736797	−0.06111596739800	−0.93119392730046

面对这一大堆的数据谁看了都会皱眉头。下面把这些数据用图形显示出来，看看到底得到了些什么。在上面的主程序中，返回了一个图。该图显示了频率 P 在搜索区间中，矩阵方程的误差 Yn，如图 13.2 所示。

下面再利用一个子程序来显示计算出的振动系统的相位图。作图的子程序如下所示，将其保存为 draw.m。

```
draw.m
function draw(An)
x=1:8;
subplot(2,4,1);                          %8 个子图中的第一个
```

```
plot(x,ones(1,8)),title('An1');      %画出第一个名为 An1 的子图
grid                                 %以网格显示
subplot(2,4,2);
plot(x,An(:,2)),title('An2');
grid
```

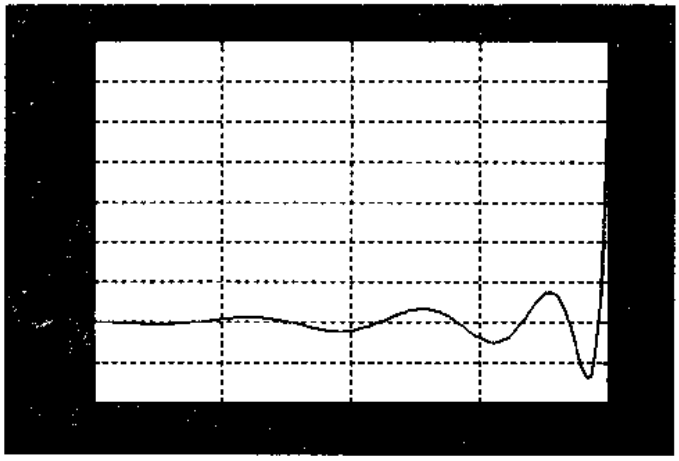


图 13.2 P-M8图

```
subplot(2,4,3);
plot(x,An(:,3)),title('An3');
grid
subplot(2,4,4);
plot(x,An(:,4)),title('An4');
grid
subplot(2,4,5);
plot(x,An(:,5)),title('An5');
grid
subplot(2,4,6);
plot(x,An(:,6)),title('An6');
grid
subplot(2,4,7);
plot(x,An(:,7)),title('An7');
grid
subplot(2,4,8);
plot(x,An(:,8)),title('An8');
grid
end
```

然后在 MATLAB 的命令行窗口中调用该函数：

```
draw(An);
```

图 13.3 就是本程序画出的振动相位图。

是不是颇有些专业水平？其实这里只是用到 MATLAB 很小一部分的功能，不过已经可以解决很深奥的问题了。从这个比较复杂的例子可以把握 MATLAB 编制程序的一些门径。

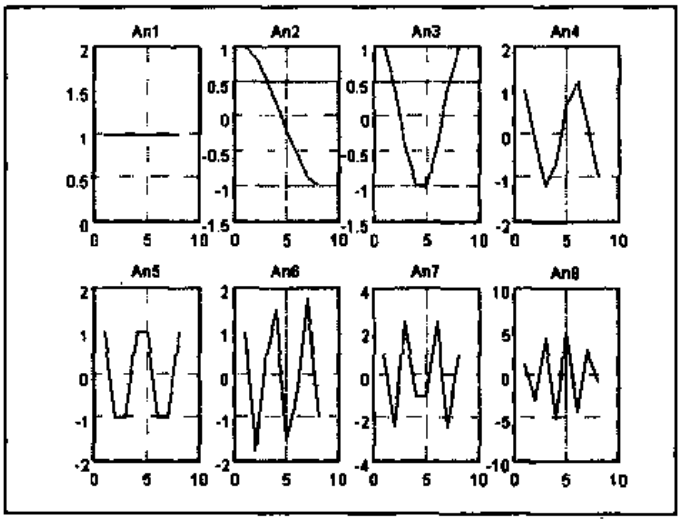


图 13.3　振动的相位图

通过以上的实例可以得到一些有益的启示，可以用来帮助解决实际问题。

- 充分利用 MATLAB 的运算工具和强大矩阵运算能力。凡是有关矩阵的问题，MATLAB 提供的函数都能高效地解决问题。
- 提供良好的程序接口，这样其他使用者可以更好地利用现成的程序。
- 注意算法的选择和研究。工程中大量的计算是不可避免的，为了让程序能更好地运用于实际中，必须有高效稳定的算法。
- 尽量将结果可视化，因为冗长枯燥的数据既难懂又无法检查错误。

13.2　GUI 示例：温度转换器

本节将通过对一个 GUI 程序编制全过程的讲解，帮助用户掌握 GUI 编程。同时，也是对前面所讲知识的复习与综合实践。

一个 GUI 程序的编写，大致包括 3 个步骤：界面绘制、各个属性的设置和具体 Callback 代码编写的思路和过程。下面分别进行介绍。

13.2.1　GUI 界面的绘制

1.　GUI界面的布置

这是编制程序的第一步。要编写一个好的 GUI 程序，必须综合考虑各方面的要求。一个简单明了、操作简便的界面则是这一切的基础。

由于在 MATLAB 5.x 版本中，集成了 GUI 控制工具集，因而界面的编写变得简单易行。图 13.4 为一个绘制好的 GUI 界面。

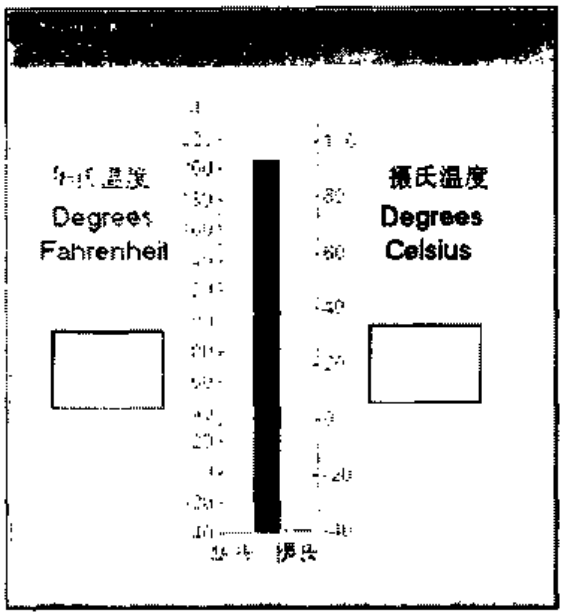


图 13.4 温度转换器的界面

在这个 GUI 界面中，可以实现以下的一些功能：

- 在左边的文本框中输入华氏温度，按 Enter 键后，右边的文本框中出现对应的摄氏温度；反之亦然。同时，中间的温度指示条显示出对应的温度。
- 在中间指示条所在的区域上单击，指示条指示鼠标位置对应的温度，同时两边的文本框显示出对应的华氏温度和摄氏温度。

为了实现这些功能，使用了以下 GUI 控件对象：

- 两个 axes 对象，分别指示华氏温度和摄氏温度的高低。
- 两个 text 对象，两个 edit 对象，用来直接输入已知的温度。
- 两个 text 对象，一个的前景色设置为红色(图 13.4 的中间部分)，用来指示温度的高低，另外一个设置为白色，与背景色相同。设置它是出于编程的考虑。

提示： 由于指示温度的 text 对象长度会发生变化，因而，鼠标操作并不能保证得到它的响应。因此采用另外一个 text 对象来接收鼠标动作，控制程序运行。

2. 控件的放置

首先按照上面的内容绘制出大概的界面，然后进行最后的调整。在这一过程中，有一些具体要注意的问题：

- 如果一个控件挡住了另外一个，可以从 Option 菜单中选择 Bring To Font 命令，反之，可以选择 Send To Back 命令。
- 坐标轴的 x 轴，在默认情况下始终会有刻度标记。可以修改它的属性，使得刻度的分度值大于刻度的范围(例如，默认刻度范围是[0 1]，将分度值设为 2)，就可以消除 x 轴上的刻度标记。
- 要保证美观，两个 edit 对象的大小必须一样。一个简单的方法是，先放置好一个的位置，然后选择 Edit 菜单下的 Copy 命令，然后再选择 Paste 命令，就会出现一个相同的 edit 对象了。这个方法也适用于 GUI 界面布置的一般情况。

- 给每个对象设置好合适的颜色，使界面美观好用。

13.2.2　属性的设置

属性的设置是很关键的一步。这里仅仅介绍和编程有关的属性设置，至于其他的一些控制外观的属性，请参考图 13.4 自行设置。

前面讲过 Tag 属性，它的目的是为了在编程时可以方便地取得相应对象的句柄。这里，几个有关对象的“Tag”属性值为：

- 左右两个 edit 对象：fahrenheittext、celsiustext
- 接收鼠标操作的 text 对象：trigger
- 指示温度的红色 text 对象：redline

在本例中，事件比较少，而且处理机制上有相似之处。因而并不单独为每个事件编写独立的 Callback 代码，而是使用函数 temperature(带参数)进行集中处理。在每个相关控件对象中，在其 Callback 属性中写出相应的函数调用。具体如下所示：

- 左边的 edit 对象：temperature fahrenheittext
- 右边的 edit 对象：temperature celsiustext
- 接收鼠标操作的 text 对象，ButtonDownFcn 的 Callback 设为 temperature start

可以参见图 13.5。

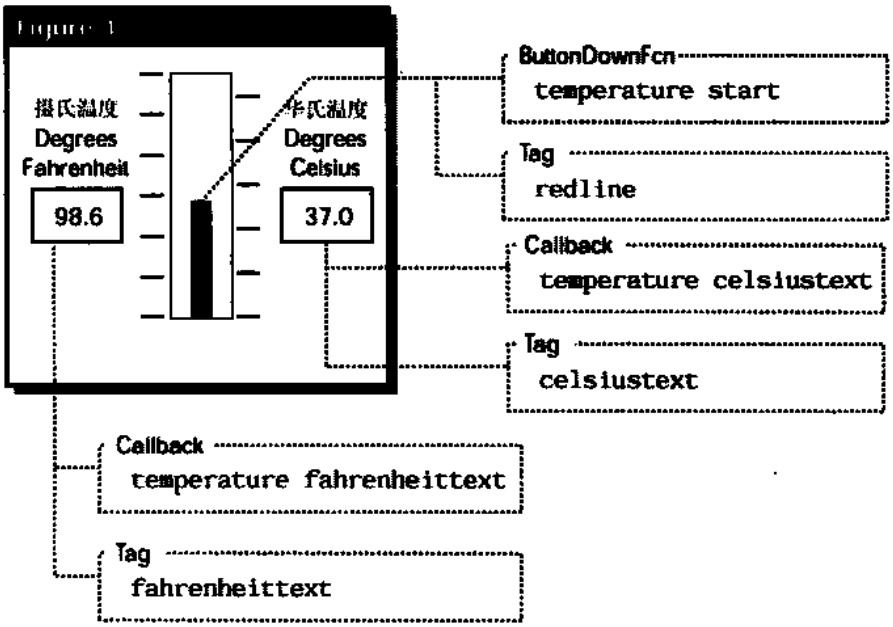


图 13.5　控件对象的属性

13.2.3 编写代码

打开 temperaturefig.m 文件，加入函数 temperature。它以 action 为参数，控制各种不同情况。

```
function temperature(action)
switch action
case 'start'
% 在此处加入初始化的代码

case 'move'
% 取得当前点的 Y 坐标，据此进行转换
% 在对应的文本框中显示数值

case 'stop'
% 结束部分，清除变量

case 'fahrenheit'
% 激活左边文本框时，调用的代码，完成功能：
% 温度的转换、温度的显示

case 'celsius'
% 激活右边文本框时，调用的代码，完成功能：
% 温度的转换、温度的显示

end
```

观察上面的框架体系可以看到，有一部分是相同的。例如在每一次转换之后，都要修改两个 edit 对象的显示内容。因此，可以考虑将这一部分写成一个函数，通过不同的参数来进行调用。例如采用函数 LocalSetDisplay( )来完成这一构想。它以温度、对象的 Tag 属性为参数来实现。

另外，还有一些别的问题需要考虑：当温度输入过高，超过指示范围时，必须保证最高值不超出原始设计图所给出的范围。可以以华氏温度为基准进行控制(不超过 240°)。edit 对象中字符输出的格式也是一个要考虑的问题。

下面，具体实现这个 GUI 编程。首先是 temperature 函数。

```
function temperature(action)
switch action

case 'start'
set(gcbf,'WindowButtonMotionFcn','temperature move')
```

```
set(gcbf,'WindowButtonUpFcn','temperature stop')
temperature move

case 'move'
% 当在温度指示条上单击鼠标时进行的操作
currentPoint = get(gca,'CurrentPoint');
newY = currentPoint(3);
% 控制温度的指示范围
fah = min(240,max(-40,newY));
% fah 代表所得到的华氏温度值
cels = (fah-32)*(5/9);
% cels 代表所得到的摄氏温度值
LocalSetDisplay(fah,cels)
% 调用函数 LocalSetDisplay，对 edit 对象进行设置

case 'stop'
% 程序结束，清除相关变量和操作
set(gcbf,'WindowButtonMotionFcn','')
set(gcbf,'WindowButtonUpFcn','')

case 'fahrenheit'
% 在左边文本框中输入温度时进行的操作
fah = eval(get(gcbo,'String'));
% 与上面一段类似，不再详述
fah = min(240,max(-40,fah));
cels = (fah-32)*(5/9);
LocalSetDisplay(fah,cels)

case 'celsius'
% 在右边文本框中输入温度时进行的操作
cels = eval(get(gcbo,'String'));
cels = min(120,max(-40,cels));
fah = cels*(9/5)+32;
LocalSetDisplay(fah,cels)
end
```

下面是函数 LocalSetDisplay 进行显示的设置。

```
function LocalSetDisplay(fah,cels)
% 设置文字显示的格式
pointerHandle = findobj(gcbf,'Tag','redline');
```

```
set(pointerHandle,'YData',[-40 fah])
fahHndl = findobj(gcbf,'Tag','fahrenheittext');
set(fahHndl,'String',sprintf('%3.1f',fah))
celsHndl = findobj(gcbf,'Tag','celsiustext');
set(celsHndl,'String',sprintf('%3.1f',cels));
```

根据上面的程序，再加上前面对属性的设置，就可以运行这个 GUI 的小程序了。

注意： 华氏温度和摄氏温度的转换关系：

$$T_{\text{摄氏温度}} = \frac{5}{9}\left(T_{\text{华氏温度}} - 32\right)$$

说明：

- 在下一次运行时，必须先进入保存它的目录，然后输入命令：

```
load temperature.mat
```

否则程序会因缺少相关数据而无法运行。

- 编程时注意变量的作用范围，处理好全局变量和局部变量的关系。注意函数内部的变量名都是局部变量(本例中没有全局变量)。

《软件入门与提高丛书》99新书导引

☞ Windows 2000 中文版入门与提高

- ✧ Microsoft 公司最新操作系统。
- ✧ 真正的 Web 集成环境，更快速、更易用、更稳定、更个性化。

☞ Office 2000 中文版入门与提高

- ✧ 真正的多语言支持功能，使 Office 2000 不再需要语言化（如汉化）。
- ✧ Microsoft 公司推出的一个最新的办公自动化集成软件。
- ✧ 集成了 Internet Explorer 5.0 中文版，使用户上网更加方便。

☞ AutoCAD 2000 入门与提高

- ✧ 目前世界上最流行的计算机辅助设计软件。
- ✧ 与前一版本 AutoCAD R 14 相比增加了 400 多种新的功能，给用户一个全新的感觉。

☞ WPS 2000 入门与提高

- ✧ WPS 2000 相对 WPS 97 增添了许多功能和最新技术，是一套性能突出的集成软件。
- ✧ WPS 2000 将以全新的界面、个性化的操作向导、强大的网络功能、丰富的字体、更加强大的表格功能等，给用户一个耳目一新的感觉。

☞ Visual Basic 6.0 中文版入门与提高

- ✧ 微软公司的可视化开发环境，拥有强大的多媒体数据库与 Internet 功能。
- ✧ 易学、易用、功能强，是 Windows 应用程序优秀的开发平台。

☞ Visual FoxPro 6.0 中文版入门与提高

- ✧ 著名的数据库管理系统，具有可视化的开发环境和面向对象的支持。
- ✧ 简捷、易用、功能强劲，是 Windows 环境下数据库开发的利器。

☞ Photoshop 5.0 入门与提高

- ✧ 多个功能的增加使 Photoshop 更加完善。
- ✧ 友好的界面以及更方便的操作工具为用户制作图像提供了广阔的空间。

《软件入门与提高丛书》已出书目

- Windows 98 中文版入门与提高
- Excel 2000 中文版入门与提高
- Visual C++ 6.0 入门与提高
- Photoshop 5.0 入门与提高
- Excel 97 中文版入门与提高
- Netscape Communicator 4.x 中文版入门与提高
- FreeHand 8.0 入门与提高
- Windows NT 3.51 中文版入门与提高
- AutoCAD R 14 中文版入门与提高
- Internet Explorer 4.0 中文版入门与提高
- Delphi 4.0 入门与提高
- Borland C++ Builder 3 入门与提高
- Windows C 程序设计入门与提高
- Authorware 5.0 入门与提高

读者意见交流卡

亲爱的读者：

感谢您对我们的支持与爱护。您购买的这本书是《软件入门与提高丛书》之一。我们出版这套丛书是为了将目前最新的流行软件以最快的速度、最详尽的讲解介绍给您，引导您轻松进入计算机之门。为了今后给您提供更优秀的图书，请您抽出宝贵的时间填写这份调查表，然后剪下寄到：北京清华大学出版社计算机编辑室（100084）；传真：010-6278 1731；电子信箱(E-mail)：hoosf@mail.tup.tsinghua.edu.cn。我们将充分考虑您的意见和建议，并尽可能地给您满意的答复。谢谢！

本书名：MATLAB 5.x 入门与提高

个人资料：

姓名：_____性别：1□男 2□ 女 出生年月(或年龄)：_____

职业：_____文化程度：_____通讯地址：_____

电话（或寻呼）：_____传真：_____电子信箱（E-mail）：_____

您是如何得知本书的：

□别人推荐 □书店 □出版社图书目录

□杂志、报纸等的介绍（请指明）_____

□其他（请指明）_____

您从何处购得本书：

□书店 □电脑商店 □软件销售处 □邮购

□商场 □其他_____

您购买过本套丛书中的几本：

□一本 两本 □三本 □四本及四本以上

影响您购买本书的因素（可复选）：

□封面封底 □装帧设计 □价格

□内容提要、前言或目录 □书评广告

□出版社名声 □作者名声 □责任编辑

□其他_____

您对本书封面设计的满意度：

□很满意 □比较满意 □一般 □较不满意

□不满意 □改进建议_____

您对本书印刷质量的满意度：

□很满意 □比较满意 □一般 □较不满意

□不满意 □改进建议_____

您对本书的总体满意度：

从文字角度 □很满意 □比较满意

□一般 □较不满意 □不满意

从技术角度 □很满意 □比较满意

□一般 □较不满意 □不满意

本书最令您满意的是：

_____

_____

您希望本书在哪些方面进行改进：

_____

_____

您希望增加什么系列或软件的图书：

_____

_____

_____

您是否希望本书配光盘：

□希望 □不希望 □无所谓

您对使用中文版软件或外文版软件介意吗？更喜欢用哪一种版本？

□介意 □无所谓； □中文版 □外文版

您对图书所用软件版本是否很介意？是否要求用最新版本？

□是，要求是最新版本 □无所谓

□不，因为硬件或软件跟不上要求

您更喜欢阅读哪些类型和层次的计算机书籍？

□入门类 □提高类 □技巧类

□实例类 □精通类 □综合类

您的其他要求：

_____