

---

高职高专“十五”规划教材系列

# MATLAB 基础及应用

于润伟 主编



机械工业出版社

# 目 录

出版说明

前言

第 1 章	MATLAB 概述 .....	1
1.1	MATLAB 的发展历史 .....	1
1.2	MATLAB 的主要特点 .....	2
1.3	MATLAB 操作桌面介绍 .....	3
1.3.1	桌面布局 .....	3
1.3.2	命令窗口 .....	3
1.3.3	工作空间窗口 .....	5
1.3.4	当前目录浏览器 .....	6
1.3.5	命令历史窗口 .....	6
1.3.6	启动平台 .....	6
1.4	帮助系统 .....	7
1.4.1	帮助浏览器 .....	7
1.4.2	help 命令 .....	8
1.4.3	lookfor 命令 .....	10
1.4.4	模糊查询 .....	10
1.4.5	在线帮助页 .....	11
1.5	上机实践 .....	11
1.5.1	跟我学 .....	11
1.5.2	自己练 .....	12
1.6	本章小结 .....	13
1.7	习题 .....	14
第 2 章	MATLAB 基础知识 .....	15
2.1	数据结构 .....	15
2.2	变量和数据操作 .....	17
2.2.1	变量 .....	17
2.2.2	预定义变量 .....	18
2.2.3	内存变量的管理 .....	18
2.2.4	常用数学函数 .....	18
2.2.5	数据的输入输出函数 .....	20
2.3	矩阵 .....	22
2.3.1	矩阵的建立 .....	22
2.3.2	复数与复数矩阵 .....	25
2.3.3	矩阵的基本运算 .....	26
2.3.4	矩阵的操作 .....	27

2.4	关系运算和逻辑运算 .....	29
2.4.1	关系运算符 .....	29
2.4.2	逻辑运算符 .....	30
2.4.3	其他关系与逻辑函数 .....	30
2.5	文件操作 .....	31
2.5.1	文件的打开与关闭 .....	31
2.5.2	二进制文件的读写操作 .....	32
2.5.3	文本文件的读写操作 .....	33
2.5.4	图像文件的读写操作 .....	33
2.6	上机实践 .....	34
2.6.1	跟我学 .....	34
2.6.2	自己练 .....	36
2.7	本章小结 .....	36
2.8	习题 .....	37
<b>第 3 章</b>	<b>MATLAB 程序设计 .....</b>	<b>39</b>
3.1	M 命令文件 .....	39
3.1.1	M 文件的建立 .....	39
3.1.2	M 文件的调试 .....	41
3.2	程序流程语句 .....	43
3.2.1	if 语句 .....	43
3.2.2	switch 语句 .....	45
3.2.3	while 语句 .....	47
3.2.4	for 语句 .....	47
3.2.5	循环的嵌套 .....	48
3.2.6	其他流程控制语句 .....	48
3.3	函数文件 .....	50
3.3.1	函数文件的基本结构 .....	50
3.3.2	函数调用 .....	51
3.3.3	函数所传递参数的可调性 .....	52
3.3.4	全局变量 .....	53
3.4	MATLAB 语言编程技巧 .....	53
3.4.1	测定程序执行时间 .....	53
3.4.2	加快 MATLAB 程序执行的方法 .....	54
3.5	上机实践 .....	54
3.5.1	跟我学 .....	54
3.5.2	自己练 .....	58
3.6	本章小结 .....	59
3.7	习题 .....	60
<b>第 4 章</b>	<b>MATLAB 绘图 .....</b>	<b>61</b>

4.1	二维绘图 .....	61
4.2	图形修饰与控制 .....	64
4.2.1	坐标轴的调整 .....	64
4.2.2	文字标示 .....	65
4.2.3	图例注解 .....	66
4.2.4	图形的保持 .....	67
4.2.5	网格控制 .....	67
4.2.6	图形窗口的分割 .....	68
4.2.7	图形的填充 .....	68
4.3	特殊坐标二维图形 .....	69
4.4	特殊二维图形 .....	70
4.5	三维图形 .....	73
4.5.1	三维数据的产生 .....	73
4.5.2	三维曲线图 .....	74
4.5.3	三维曲面图形 .....	75
4.6	上机实践 .....	76
4.6.1	跟我学 .....	76
4.6.2	自己练 .....	78
4.7	本章小结 .....	78
4.8	习题 .....	79
<b>第5章</b>	<b>MATLAB 符号计算 .....</b>	<b>81</b>
5.1	符号对象 .....	81
5.1.1	创建符号变量和符号矩阵 .....	81
5.1.2	符号表达式的基本运算函数 .....	82
5.1.3	符号表达式的化简函数 .....	83
5.1.4	符号表达式的替换函数 .....	84
5.2	符号微积分 .....	86
5.2.1	符号极限 .....	86
5.2.2	符号求导 .....	86
5.2.3	符号积分 .....	87
5.2.4	积分变换 .....	88
5.3	符号方程求解 .....	89
5.3.1	代数方程 .....	89
5.3.2	符号微分方程求解 .....	90
5.4	级数 .....	91
5.4.1	级数的符号求和 .....	91
5.4.2	函数的泰勒级数 .....	92
5.5	可视化符号函数计算器 .....	93
5.6	上机实践 .....	95

5.6.1	跟我学	95
5.6.2	自己练	97
5.7	本章小结	98
5.8	习题	98
<b>第 6 章</b>	<b>MATLAB 数值计算</b>	<b>100</b>
6.1	矩阵处理	100
6.1.1	矩阵分析	100
6.1.2	矩阵的线性变换	101
6.1.3	矩阵分解	101
6.1.4	稀疏矩阵	103
6.2	数据分析	105
6.2.1	基本数据分析函数	105
6.2.2	离差函数和相关函数	106
6.3	数值分析	107
6.3.1	多项式	107
6.3.2	数值插值与曲线拟合	108
6.3.3	函数的极值和零点	110
6.4	常微分方程的数值求解	111
6.4.1	常微分方程的解法	111
6.4.2	龙格-库塔法的实现	112
6.5	上机实践	113
6.5.1	跟我学	113
6.5.2	自己练	116
6.6	本章小结	118
6.7	习题	118
<b>第 7 章</b>	<b>Simulink 仿真</b>	<b>120</b>
7.1	认识 Simulink	120
7.1.1	Simulink 简介	120
7.1.2	Simulink 的启动和退出	120
7.2	Simulink 的基本模块	122
7.3	Simulink 模块的操作	127
7.3.1	添加和选取模块	127
7.3.2	模块位置和外形的调整	127
7.3.3	模块名的处理	128
7.3.4	复制和删除模块	129
7.3.5	模块属性和参数的设置	129
7.3.6	模块间的连线	130
7.4	仿真模型的参数设置	131
7.4.1	Solver 选项卡	132

7.4.2	Workspace I/O 选项卡 .....	133
7.4.3	Diagnostics 选项卡 .....	134
7.5	上机实践 .....	135
7.5.1	跟我学 .....	135
7.5.2	自己练 .....	141
7.6	本章小结 .....	141
7.7	习题 .....	141
第 8 章	MATLAB 应用实例 .....	143
8.1	数字滤波器的设计 .....	143
8.1.1	FIR 滤波器的设计 .....	143
8.1.2	IIR 滤波器的设计 .....	144
8.2	在信号处理中的应用 .....	145
8.2.1	快速傅立叶变换 (FFT) .....	145
8.2.2	离散余弦变换 (DCT) .....	147
8.3	在图像处理中的应用 .....	148
8.3.1	图像的几何操作 .....	148
8.3.2	图像的消噪处理 .....	149
8.3.3	图像融合 .....	149
8.3.4	图像压缩 .....	150
8.4	在通信技术中的应用 .....	151
8.4.1	信源编码 .....	151
8.4.2	单边带调幅 (SSB) .....	153
8.5	在控制系统分析中的应用 .....	155
8.5.1	线性反馈系统 .....	155
8.5.2	自动控制系统 .....	156
8.6	本章小结 .....	158
8.7	习题 .....	158
附录	.....	159
附录 A	MATLAB 命令参考 .....	159
附录 B	工具箱函数 .....	170
参考文献	.....	187

# 第 1 章 MATLAB 概述

在本章中，读者可以了解 MATLAB 语言的主要特点，熟悉其操作桌面和命令窗口；学会使用 MATLAB 的帮助系统，能够利用帮助系统查询函数的用法；能够在命令窗口中进行简单的数学运算。本章重点是 1.4 帮助系统，难点是 1.4.1 帮助浏览器。

## 1.1 MATLAB 的发展历史

MATLAB 是美国 MathWorks 公司推出的一套高性能的数值分析和计算软件，它将矩阵运算、数值分析、图形处理、编程技术结合在一起，为用户提供了一个强有力的分析、计算和程序设计的工具。随着时间的推移，MATLAB 本身也在不断改进和创新，特别是 2001 年推出的 6.1 版本在界面设计、计算方法、编程手段和工具等方面都有了巨大的突破，全面引入了面向对象编程的概念和方法，使 MATLAB 真正成为了具有全部高级语言功能和特点的新一代软件开发平台。6.1 版本新增了多种专用工具箱，如有限元分析、控制系统、系统辨识、信号处理、鲁棒性控制、 $\mu$ 分析与综合、模糊控制、神经网络、小波分析、定量反馈理论、多变量频域设计等工具箱。对原有软件，如 Simulink 仿真环境进行了较大规模的改进，使其仿真功能得到全面优化。同时，还增加了符号运算功能、图形处理功能，使 MATLAB 的应用更为广泛。

由于在 6.1 版本中增加了由 MATLAB 语言直接转为 C 语言代码的功能，使 MATLAB 在工程设计和实现方面具有了实用性，增加了软件的竞争优势，受到广大工程技术人员的重视。今天，MATLAB 已经走出了校园，深入到工业生产、科学研究等各领域，成为世界范围内公认的、具有高可靠性的高级计算机编程语言，成为众多新型项目开发和产品研制的首选软件环境，成了很多专业领域科技人员必须掌握的一门计算机技术。

在国内，MATLAB 语言也得到越来越多的师生和科研、工程技术人员的认可，在教学、科研、工程技术中得到了广泛应用。

MATLAB 自 1984 年由 MathWorks 公司推向市场以来，历经十几年的发展和改造，其中在 1993 年推出 Windows 界面下的 MATLAB4.0 版本以及在 1999 年推出的 MATLAB5.3 版本，这两个版本被广泛的使用，对于 MATLAB 的发展具有标志性的意义。表 1-1 列出了 MATLAB 各版本的升级历程。

表 1-1 MATLAB 版本升级历程

日 期	版 本	操 作 平 台
1984 年	1.0 版	Dos
1987 年	3.0 版	
1991 年	3.5 版	

(续)

日 期	版 本	操 作 平 台
1993 年 1 月	4.0 版	Windows3.x
1993 年 11 月	4.1 版	
1994 年 5 月	4.2 版	
1995 年 5 月	4.2c 版	
1997 年 5 月	5.1 版	Windows95/98
1998 年 1 月	5.2 版	
1999 年初	5.3 版	
2000 年 9 月	6.0 版	Windows98/2000/XP
2001 年 6 月	6.1 版	

## 1.2 MATLAB 的主要特点

MATLAB 是数学计算的强有力工具，它以矩阵作为数据操作的基本单位，是以矩阵运算为主要工作方式的数理统计、自动控制、数字信号处理、动态系统仿真等方面的重要工具。MATLAB 操作简单、功能强大、应用广泛。它的特点体现在如下几个方面。

### 1. 编程效率高

MATLAB 是一种面向科学与工程计算的高级语言，允许用数学形式的语言编写程序，且比 BASIC、FORTRAN 和 C 语言等更加接近人们书写计算公式的思维方式。用 MATLAB 编写程序犹如在演算纸上排列出公式与求解问题，因此 MATLAB 语言也被通俗的称为“演算纸式”的科学算法语言。由于它拥有丰富的库函数，所以编写简单、编程效率高、易学易懂。

### 2. 用户使用方便

MATLAB 语言是一种解释执行的语言，它灵活、方便，其调试程序的手段丰富，调试速度快，调试方法简单，使用者在短时间即可学会。人们用任何一种语言编写和调试程序一般都要经过 5 个步骤：编辑、编译、连接、执行和调试，但 MATLAB 语言把编辑、编译、连接、执行和调试融为一体，较好地解决了上述问题。它能在同一窗口上进行灵活操作，快速查找输入程序中的书写错误和语法错误，从而加快了用户编写、修改和调试程序的速度，可以说在编程和调试过程中它是一种比 VB 还要简单的语言。

具体地说，MATLAB 运行时，如直接在命令窗口输入 MATLAB 语句命令，包括调用 M 文件的语句，每输入一条语句，就立即对其进行处理，完成编译、连接和运行的全过程。又如，将 MATLAB 原程序编辑为 M 文件，由于 MATLAB 磁盘文件也是 M 文件，所以编辑后的源文件就可直接运行，而不需进行编译和连接。在运行 M 文件时，如果有错，计算机屏幕上会给出详细的出错信息，但 MATLAB 并非一次显示所有的错误，而是每次运行只显示第一条错误，用户可以边修改边执行，直到正确为止。

### 3. 扩充能力强

高版本的 MATLAB 语言（尤其是 MATLAB6.1 版）有丰富的库函数，在进行复杂的数学运算时可以直接调用，而且 MATLAB 的库函数同用户文件在形式上一样，用户文件也可



作为 MATLAB 的库函数来调用。因而，用户可以根据自己的需要方便地建立和扩充新的库函数，以便提高 MATLAB 使用效率和扩充 MATLAB 的功能。另外，为了充分利用 FORTRAN、C 语言等的资源，包括用户已编好的 FORTRAN、C 语言程序，可通过建立 Mex 调用文件格式，进行混合编程，能够方便地调用有关 FORTRAN、C 语言的子程序。

#### 4. 语句简单

MATLAB 语言中最基本、最重要的成分是函数。同一个函数名，不同数目的输入变量（包括无输入变量）及不同数目的输出变量，代表着不同的含义（有点像面向对象中的多态性）。这不仅使 MATLAB 的库函数功能更丰富，而且大大减少了所需的磁盘空间，使得 MATLAB 编写的 M 文件简单、短小而实用。

#### 5. 高效方便的矩阵和数组运算

MATLAB 语言像 BASIC、FORTRAN 和 C 语言一样规定了矩阵的算术运算符、关系运算符、逻辑运算符、条件运算符及赋值运算符，而且这些运算符大部分可以毫无改变地照搬到数组间的运算，有些运算符（如算术运算符）只要适当改变形式就可用于数组间的运算。另外，它不需定义数组的维数，并给出矩阵函数和特殊矩阵专用的库函数，使之在求解诸如信号处理、建模、系统识别、控制、优化等领域的问题时，显得极为简捷、高效、方便，这是其他高级语言所不能比拟的。

#### 6. 方便的绘图功能

利用 MATLAB 绘图十分方便，它有一系列绘图函数。例如建立线性坐标、对数坐标及极坐标，均只须调用不同的绘图函数；图上的标题、坐标轴标注、栅格绘制也只需调用相应的命令，简单易行。另外，在调用绘图函数时调整绘图参数可绘出不同颜色的点、线、复线或多重线，这种为科学研究着想的设计是大多数编程语言所不及的。

#### 7. 开放的源程序

开放性也许是 MATLAB 最受人们欢迎的特点。除内部函数以外，所有 MATLAB 的核心文件和工具箱文件都是可读可改的源文件，用户可通过对源文件的修改以及加入自己的函数文件来构成新的工具箱。

## 1.3 MATLAB 操作桌面介绍

### 1.3.1 桌面布局

与一般的 Windows 程序一样，双击桌面上的 MATLAB 图标，即可启动 MATLAB 系统。这时将看到如图 1-1 所示的 MATLAB 操作桌面。

操作桌面包括命令窗口（Command Window）、工作空间窗口（Workspace）、当前目录浏览器（Current Directory）、命令历史窗口（Command History）、启动平台（Launch Pad）等 5 个窗口，其中工作空间窗口（Workspace）和启动平台（Launch Pad）共用一个窗口，当前目录浏览器（Current Directory）、命令历史窗口（Command History）共用一个窗口。

### 1.3.2 命令窗口

命令窗口（Command Window）用于输入 MATLAB 命令、函数、矩阵、表达式等信息，

并显示除图形以外的所有计算结果，是 MATLAB 的主要交互窗口。

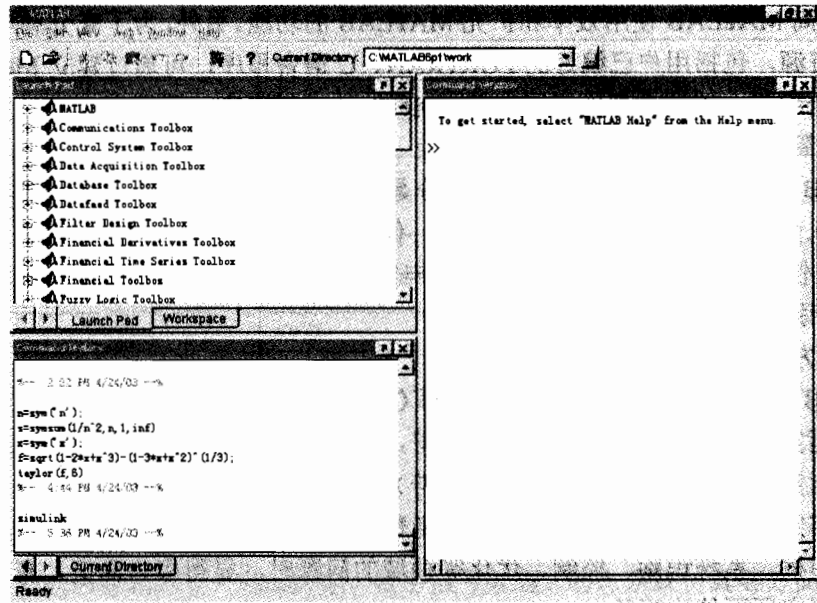


图 1-1 MATLAB 操作桌面

命令窗口提供了命令行编辑功能，即通过键盘输入控制命令、建立矩阵、运算表达式等。

MATLAB 是一种交互式语言，输入命令即给出运算结果。当命令窗口出现提示符>>时，表示 MATLAB 已准备好，可以输入命令、变量或运行函数。下面主要介绍命令行的编辑功能和方法。

例 1-1 建立一个 3×3 的矩阵。

```
>> a=[1 3 5;7 9 11;13 15 17] %从键盘输入，并按回车键
```

```
a = %屏幕显示的结果
```

```
1     3     5
7     9    11
13    15    17
```

例 1-2 计算  $\sin(\pi/5)+4 \cos(\pi/4)$

```
>> sin(pi/5)+4*cos(pi/4) %从键盘输入，并按“Enter”键，pi 代表 $\pi$ 
```

```
ans = %屏幕显示的结果，ans 表示默认变量名
```

```
3.4162
```

注意：

在 MATLAB 程序设计中规定，如果程序行中的第一个符号是百分号“%”，则该行为注释行，是用来进行说明而不能执行的行。

### 1.3.3 工作空间窗口

工作空间窗口 (Workspace) 是 MATLAB 用于存储各种变量和结果的内存空间。需要注意的是: MATLAB 的函数在运行中会调用一些临时变量, 这些变量在函数运行结束后将被释放, 因此, 这些临时变量不会占用工作空间。

工作空间窗口用于显示工作空间中所有变量的名称、大小、字节数及数据类型, 可对变量进行观察、编辑、保存和删除。


在命令窗口输入变量, 通过运行文件建立变量或返回计算结果的变量时, 这些变量都将被存储在工作空间 (内存) 中, 直到使用了 “clear” 命令清除工作空间或关闭了 MATLAB 系统为止。通过工作空间窗口可以观察数据名称、尺寸及数据类型等信息, 为了对变量的内容进行观察、编辑与修改, 可以打开内存数组编辑器。打开内存数组编辑器有三种方法: 双击变量名、选择该窗口工具栏上的图标、将鼠标指向要观察的变量名, 单击鼠标右键, 弹出一个包含 8 项内容的菜单。菜单的选项如表 1-2 所示。

表 1-2 内存数组编辑器

选 项	内 容
Open Selection...	在内存数组编辑器中打开所选的变量
Graph Selection	图形显示所选择的变量
Select All	选中全部变量
Import Data...	导入数据
Save Selection As...	将选择的变量存为...
Save Workspace As...	将工作空间存为...
Delete Selection	删除选择的变量
Clear Workspace	清除工作空间

表中第二个选项 (Graph Selection) 中, 还有多个选项:

- plot (二维曲线绘图)
- surf (三维表面绘图)
- 2-D Graphics (二维绘图)
- 3-D Graphics (三维绘图)
- special 2-D Graphics (特殊二维绘图)
- special 2.5-D Graphics (特殊三维绘图)

其中除了 plot 和 surf 两项外, 其余四项都要做进一步的选择, 以画出各种不同的图形 (如 loglog, mesh, bar, pie, contour 等)。可选择该菜单上的 “Open Selection...” 选项, 即可打开内存数组编辑器。

**提示:**

在命令窗口中, 键入 “whos” 命令, 可以显示出保存在工作空间中所有变量的名称、大小、数据类型等信息, 如果键入 “who” 命令则只显示出变量的名称。

### 1.3.4 当前目录浏览器

当前目录是指 MATLAB 运行文件时的工作目录，只有在当前目录或搜索路径下的文件及函数可以被运行或调用，如果没有特殊指明，数据文件也将存储在当前目录下。通常很多人都习惯于建立自己的工作目录，以便于文件和数据的管理，因此在运行文件前要将该文件所在的目录设置为当前目录。

当前目录浏览器（Current Directory）用于显示及设置当前工作目录，同时显示当前工作目录下的文件名、文件类型及目录的修改时间等信息。

如果在运行的文件中需要调用多个目录下的文件或函数，可以将这些目录加入到搜索路径中，这样在运行这些文件时，就不需要再将其设置成当前目录了。需要注意的是，在存储数据文件时，默认状态下数据文件将直接存储在当前目录下。

### 1.3.5 命令历史窗口

命令历史窗口（Command History）是为记录已运行过的 MATLAB 命令历史而设计的，命令历史窗口记录已经运行过的命令、函数、表达式等信息，可以进行命令历史的查找、检查等工作。也可以在该窗口中对命令历史进行复制及重运行，窗口中除了保留了输入的命令外，还记录了每次打开系统的时间。如果要清除掉这些记录，可以选择“Edit”菜单中的“Clear Command History”项。

此外，对命令历史窗口中的命令进行操作时，可在所要操作的命令处单击鼠标右键，弹出一个下拉式菜单，菜单中有 6 项有关命令的内容。

- 1) 复制 (Copy): 将选定的内容复制到剪贴板中，以便粘贴到其他地方。
- 2) 执行所选命令 (Evaluate Selection): 直接将选定的命令送到窗口中执行。
- 3) 建立 M 文件 (Create M-File): 打开一个新的 M 文件，并将选定内容复制过去。
- 4) 删除所选命令 (Delete Selection): 从命令历史窗口中删除选定的命令。
- 5) 删除到所选命令 (Delete to Selection): 删除所选命令之前输入的所有命令。
- 6) 删除全部历史 (Delete Entire History): 删除命令历史窗口中的全部命令。

其中第二项操作也可以在所选命令处双击鼠标完成。第三、第四项操作的目的是为了删除命令历史窗口中错误和无用的命令，以使窗口中保留的命令更加整齐、易于查找。

#### 提示:

当光标在命令窗口中，按光标移动键 (<↑>)，可以调出输入的前一命令行；按光标移动键 (<↓>)，可以调出输入的最后一命令行。

### 1.3.6 启动平台

启动平台 (Launch Pad) 是 MATLAB6.0 新增加的功能，它可以帮助用户方便地打开和调用 MATLAB 的各种程序、函数和帮助文件。启动平台列出了系统中安装的所有 MATLAB 产品的目录，包括 MATLAB 产品的帮助界面、演示界面、各种应用工具界面及网站的产品页等 (见图 1-2)，可以通过双击来启动相应的选项。每个选项中都有一个 Demos，Demos

是一个产品演示工具。

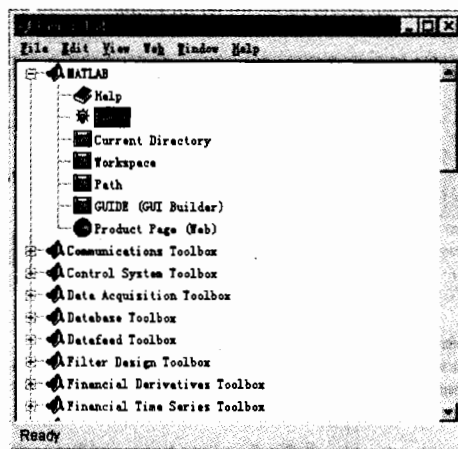


图 1-2 启动平台窗口

**提示:**

启动平台 (Launch pad) 是一个展示 MATLAB 功能和产品工具箱的平台, 可以通过 Demos 来了解 MATLAB。

## 1.4 帮助系统

MATLAB 提供了数目繁多的函数和命令, 要全部把它们记下来是不现实的。可行的办法是先掌握一些基本内容, 然后在实践中不断总结和积累, 掌握其他内容。通过软件系统本身提供的帮助系统来学习软件的使用是重要的学习方法。MATLAB 也提供了丰富的帮助功能, 通过这种功能可以很方便地获得有关函数和命令的使用方法。

### 1.4.1 帮助浏览器

MATLAB 6.1 设计了全新的帮助浏览器, 浏览器为用户提供了方便快捷的帮助信息获取途径和图文并茂的帮助内容。当勾选了“View”菜单中的“Help”选项、在启动平台的树状列表选择了“Help”项或在“Help”菜单中选择“MATLAB Help”菜单项时, MATLAB 都将打开一个独立的交互式帮助浏览器, 如图 1-3 所示。

帮助浏览器的左侧有四个重叠的页面:

- 第一页 Contents 是一个树状列表, 按内容列出了所有的帮助内容, 单击相应的内容就会在右侧窗口中显示超文本格式的帮助内容。
- 第二页 Index 是按字母顺序列出条目, 同时指出每一条目内容所在的位置 (如属于基本平台中的某一部分或属于某一工具箱等)。可先按字母顺序查找到需要的条目, 单击该条目就会在右侧的窗口中显示帮助内容。
- 第三页 Search 允许用户输入指定内容进行搜索。在“Search for”栏中输入指定内容

为关键词后按“Go”按钮，该页面就会显示出搜索到的内容，再单击其中具体的条目，就可以在右侧窗口中显示出详细的帮助内容。

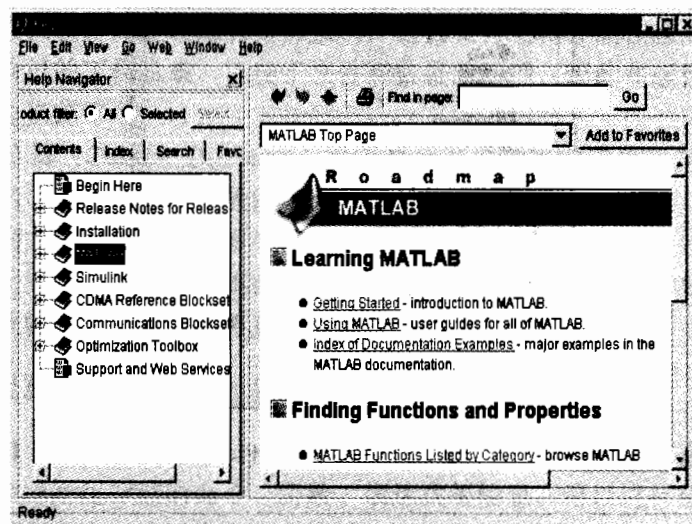


图 1-3 帮助浏览器

- 第四页 Favorites 可存放常用内容（便于用户快速地查找），存放的方法是，在前面三个页面中的任何一个页面中查找到感兴趣的内容，将鼠标指向该内容后按鼠标右键，弹出一个标有“Add to favorites”字样的按钮，单击该按钮，将选定的内容加到第四页中。

## 1.4.2 help 命令

在命令窗口输入“help”命令，也是 MATLAB 寻求在线帮助的一种方便快捷的方法，“help”命令的用法主要有以下三种。

- `>> help`      % 在命令窗口直接输入 help，显示在线帮助总览

HELP topics:

matlab\general	- General purpose commands.
matlab\ops	- Operators and special characters.
matlab\lang	- Programming language constructs.
matlab\elmat	- Elementary matrices and matrix manipulation.
matlab\elfun	- Elementary math functions.
matlab\specfun	- Specialized math functions.
matlab\matfun	- Matrix functions - numerical linear algebra.
matlab\datafun	- Data analysis and Fourier transforms.

- `>> help elfun`      % 寻求关于基本函数的帮助

Elementary math functions.  
Trigonometric.

---

sin	- Sine.
sinh	- Hyperbolic sine.
asin	- Inverse sine.
asinh	- Inverse hyperbolic sine.
cos	- Cosine.
cosh	- Hyperbolic cosine.
acos	- Inverse cosine.
acosh	- Inverse hyperbolic cosine.
tan	- Tangent.
tanh	- Hyperbolic tangent.
atan	- Inverse tangent.
atan2	- Four quadrant inverse tangent.
atanh	- Inverse hyperbolic tangent.
sec	- Secant.
sech	- Hyperbolic secant.
asec	- Inverse secant.
asech	- Inverse hyperbolic secant.
csc	- Cosecant.
csch	- Hyperbolic cosecant.
acsc	- Inverse cosecant.
acsch	- Inverse hyperbolic cosecant.
cot	- Cotangent.
coth	- Hyperbolic cotangent.
acot	- Inverse cotangent.
acoth	- Inverse hyperbolic cotangent.
Exponential.	
exp	- Exponential.
log	- Natural logarithm.
log10	- Common (base 10) logarithm.
log2	- Base 2 logarithm and dissect floating point number.
pow2	- Base 2 power and scale floating point number.
sqrt	- Square root.
nextpow2	- Next higher power of 2.
Complex.	
abs	- Absolute value.
angle	- Phase angle.
complex	- Construct complex data from real and imaginary parts.
conj	- Complex conjugate.
imag	- Complex imaginary part.
real	- Complex real part.
unwrap	- Unwrap phase angle.
isreal	- True for real array.
cplxpair	- Sort numbers into complex conjugate pairs.
Rounding and remainder.	
fix	- Round towards zero.

floor	- Round towards minus infinity.
ceil	- Round towards plus infinity.
round	- Round towards nearest integer.
mod	- Modulus (signed remainder after division).
rem	- Remainder after division.
sign	- Signum.

● >> help round    %显示具体函数的详细信息，本例为 round 函数

```
ROUND   Round towards nearest integer.
ROUND(X) rounds the elements of X to the nearest integers.

See also FLOOR, CEIL, FIX.
Overloaded methods
help quantizer/round.m
```

注意：

MATLAB 对字母的大小写是敏感的，变量 *A* 与变量 *a* 表示的是两个不同的变量。在 MATLAB 中所有的命令和函数都必须用小写（如应写成 round，而不能写成 Round 或 ROUND）。

“help”的工作原理是把指定名字的 M 文件（或函数）的第一段注释内容显示出来。因此，用户也可以采用这种注释结构，构成自己文件的在线帮助。

### 1.4.3 lookfor 命令

当用户希望查找具有某种功能的命令或函数，但又不知道准确的名字的时候，“help”的能力就不够用了。为此 MATLAB 设计了一个“lookfor”命令，它可以根据用户提供的完整或不完整的关键词，搜索出一组与之相关的命令和函数。

例 1-3 查找有关图像的命令和函数，将 image 作为关键词来查找。

```
>> lookfor image

CONTRAST Gray scale color map to enhance image contrast.
FRAME2IM Convert movie frame to indexed image.
IM2FRAME Convert indexed image into movie format.
IMAGE   Display image.
IMAGESC Scale data and display as image.
IND2RGB Convert indexed image to RGB image.
PRINT Print figure or model. Save to disk as image or M-file.
```

### 1.4.4 模糊查询

MATLAB 6.0 以后的版本提供了另一种方便的查询方法，即模糊查询方法。用户只需要输入命令的前几个字母，然后按〈Tab〉键，MATLAB 就会列出所有以这几个字母开始的命令。



例 1-4 查询与以 plot 开头的命令。

>> plot    %键入 plot 然后按〈Tab〉键

plot	plotbode	plotep	plotfrsp	plotmf	plotpv	plottr
plot3	plotbr	ploterr	plotlr	plotnic	plotscale	plotv
plot3m	plotchar	plotes	plotm	plotnyq	plotsm	plotvec
plotall	ploteach	plotfa	plotmap	plotpc	plotsom	plotyy
plotbintree		plotedit	plotfis	plotmatrix	plotperf	plotstep

这样用户就知道了某一条命令或函数的确切写法，然后再通过“help”命令查询其详细的解释。

### 1.4.5 在线帮助页

帮助桌面的所有文件均有相应的 PDF 格式文件，称为在线帮助页。PDF 格式文件可用 Adobe Acrobat Reader 阅读。用户选中帮助桌面上关于 PDF 格式文件的选项，或是在命令窗口中键入命令“doc”，便会自动打开 Adobe Acrobat Reader。

对于 Internet 用户，还可以通过帮助桌面很方便地访问 MathWorks 公司的主页，向 MathWorks 公司咨询；访问国内一些大学的 BBS，比如清华大学、哈尔滨工业大学、上海交通大学和西安电子科技大学，里面都有 MATLAB 讨论区；东北大学薛定宇教授主持的 MATLAB 大观园网站，有很多精彩内容（尤其是控制方面）；MathWorks 公司在中国的代理北京恒润公司的网站，也能够提供很多帮助。

#### 提示：

在命令窗口中，键入“clear”命令，可以清除工作空间内所有变量，并释放系统内存空间。如果键入“clc”命令则只清除命令窗口屏显内容，而保留工作空间内容。

## 1.5 上机实践

### 1.5.1 跟我学

例 1-5 求解线性方程组：

$$\begin{cases} 10x + 3y + z = 14 \\ 2x - 10y + 3z = -5 \\ x + 3y + 10z = 14 \end{cases}$$

解：

```
>> A=[10,3,1;2,-10,3;1,3,10];  
>> B=[14;-5;14];
```

```
>> root=inv(A)*B      %inv 为求逆矩阵函数
```

```
root =  
1.0000      %x 的解  
1.0000      %y 的解  
1.0000      %z 的解
```

**例 1-6** 利用快速傅立叶变换分析语音信号的频谱（见图 1-4）。语音信号可通过计算机的声卡读入，也可以使用 MATLAB 工具箱中提供的语音文件 mtlb.m。

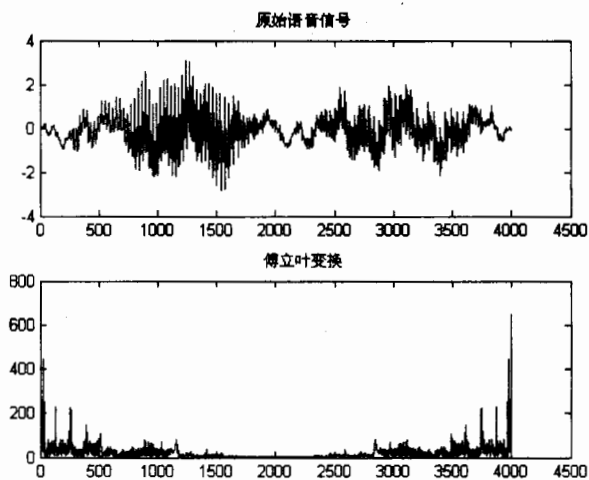


图 1-4 语音信号的频谱

解：

```
>> load mtlb;          %读入原始语音信号 mtlb  
>> subplot(2,1,1);    %把窗口分成 2 行 1 列，选中第一个行  
>> plot(mtlb);         %画出原始语音信号的波形  
>> title('原始语音信号'); %波形标题  
>> y=fft(mtlb);        %对原始语音信号作快速傅立叶变换  
>> subplot(2,1,2);    %把窗口分成 2 行 1 列，选中第二个行  
>> yy=abs(y);          %求复数的幅值  
>> plot(yy);           %画出语音信号傅立叶变换的波形  
>> title('傅立叶变换');
```

**例 1-7** 绘制三维立体曲面（如图 1-5 所示）。

解：

```
>> z=peaks(40);        %建立 40×40 的双峰三维数据矩阵  
>> mesh(z);            %以 Z 矩阵元素和其下标为数据点绘制网格线  
>> surf(z);            %生成着色表面图
```

## 1.5.2 自己练

1. 安装 MATLAB 6.1 软件。
2. 启动、退出 MATLAB。

3. 通过帮助桌面访问 MathWorks 公司的主页。

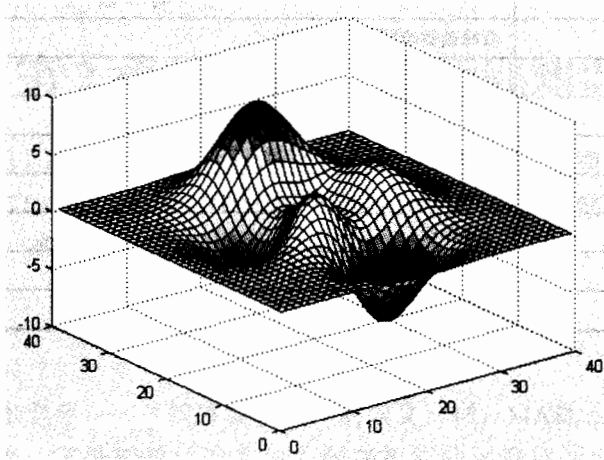


图 1-5 三维立体曲面

4. 通过 [www.hirain.com](http://www.hirain.com) 访问北京润恒公司的网站，了解 MATLAB 在国内的应用。
5. 通过帮助浏览器查找 max 函数的用法。
6. 通过帮助浏览器查找并比较 ceil、floor、fix、round、mod、rem 和 sign 函数的用法。
7. 解线性方程组：

$$\begin{cases} 3x + 4y - 2z = 12 \\ 6x + 2y - 3z = 4 \\ 45x + 5y + 4z = 23 \end{cases}$$

### 1.6 本章小结

本章介绍了 MATLAB 的主要特点、发展历程、MATLAB 操作桌面的主要窗口和功能菜单。重点讲解了 MATLAB 帮助系统的使用方法。通过几个简单的例子，表现了 MATLAB 在数值计算、绘图和信号处理上的应用，展示了 MATLAB 的特点。表 1-3 列出了与本章相关的命令或函数。

表 1-3 本章命令或函数一览

命令（或函数）名称	功 能
who	列出工作空间中的变量名
whos	列出工作空间中的变量详细内容
help	M 文件和函数文件的在线帮助
lookfor	搜索指定的关键词
clc	清除命令窗口屏显内容
clear	清除工作空间内所有变量，释放系统内存
inv	矩阵求逆

(续)

命令(或函数)名称	功 能
peaks	双峰函数绘图
mesh	三维网格曲面
surf	三维曲面图
load	从 MAT 或 ASCII 文件中读入数据
subplot	绘制子图
plot	二维绘图
title	文本标题
fft	一维快速傅立叶变换

## 1.7 习题

1. 与其他高级语言相比, MATLAB 有哪些显著特点?
2. 试将 MATLAB 系统的用户界面、操作方式与其他 Windows 应用程序(如 Word)做一对比, 有哪些共同规律?
3. 先建立自己的工作目录, 再通过路径浏览器将自己的工作目录设置到 MATLAB 搜索路径下, 用“help”命令能查询到自己的工作目录吗?
4. 查看 MATLAB 的目录结构, 并检查自己的机器上安装了哪些工具箱?
5. 利用 MATLAB 的帮助功能分别查询 inv 函数、plot 函数? 并了解 MATLAB 操作系统命令的功能及用法。
6. gcd 函数可以用于两个整数的最大公因数。先用“help”命令查看该函数的用法, 然后利用该函数求 15、35 的最大公因数。
7. 求解下列线性方程组:

$$(1) \begin{cases} x+3y+2z=1 \\ x+y-13z=14 \\ 5x-5y+z=2 \end{cases} \quad (2) \begin{cases} 2x+3y+21=z \\ y-13z=4x \\ 5x+3z=2y \end{cases}$$

## 第2章 MATLAB 基础知识

在本章中读者可以了解 MATLAB 的数据结构，熟悉整型、实型、字符型等基本数据类型的特点；学会 MATLAB 数据和文件的输入、输出操作；能够创建矩阵和进行矩阵运算。本章重点是 2.3 矩阵，难点是 2.5 文件操作。

### 2.1 数据结构

正如 MATLAB 的名字——“矩阵实验室”的含义一样，MATLAB 是由专门用于矩阵运算的软件发展起来的，它最初的目的是为了解决矩阵的运算问题而开发的，所以矩阵是 MATLAB 最基本、最重要的数据对象。MATLAB 的大部分运算或命令都是在矩阵运算的意义下执行的，而且这种运算是定义在复数域上的。MATLAB 的矩阵运算功能非常丰富，可以支持线性代数所定义的全部矩阵运算，许多含有矩阵运算的复杂计算问题，在 MATLAB 中很容易得到解决。

通常的矩阵是指含有  $m$  行  $n$  列数据的矩形结构。通过一定的转化方法，可以将一般的数学运算转化成相应的矩阵运算来处理。在 MATLAB 中，单个数值（标量）被看作是只有一行一列仅含一个元素的矩阵；列向量是只有一列的矩阵；行向量（矢量）是只有一行的矩阵。其他语言中常用到的数组和 MATLAB 中的矩阵在数据结构上没有区别，只是运算规则不同。数组运算是元素对元素的运算，也就是说无论什么运算，对数组中的元素都是平等进行的；矩阵运算是强调整体的运算，采用线性代数的运算方法。MATLAB 可以进行上述两种运算。那么，MATLAB 是如何区别这两种运算呢？MATLAB 是通过运算符的不同来区别的，带有小黑圆点的运算符就代表相应的数组运算。

对于数值数据，MATLAB 中最常用的类型为双精度型，占 64 位（8 个字节），用 `double` 函数实现转换。此外还有单精度数，占 32 位（4 个字节），用 `single` 函数实现转换。还有带符号整数和无符号整数，其转换函数有 `int8`、`int16`、`int32`、`uint8`、`uint16`、`uint32`，每一个函数名后面的数字表示相应数据类型所占位数，其含义不难理解。除数值数据以外，还有字符数据，在 MATLAB 中用 `char` 函数实现转换。在一般情况下，数组的每个元素必须具有相同的数据类型，但在实际应用中，有时需要将不同类型的数据构成数组的元素，因此 MATLAB 提供了结构（Structure）和单元（Cell）数据类型，用户也可以根据实际应用需要自定义数据类型。此外，MATLAB 还提供多维数组以及工程中应用十分广泛的稀疏矩阵（Sparse）。

在表 2-1 中列出了各种数据类型的简单例子，并对每种数据类型进行简要说明。需要说明的是：MATLAB 的数据类型还有一些其他的表达形式，表中没有列出。如果没有定义，MATLAB 默认的数据类型是双精度型。表 2-1 中的数据例子可以在命令窗口直接输入。

表 2-1 数据类型一览表

类型名称	函数	举 例	说 明
字符型	char	'A' 'happy'	字符型数组（每个字符 16 位），也可以用于字符串操作
整型（有符号）	int8 int16 int32	int8(32)	存储型变量为 8 位、16 位及 32 位的整数数组，帮助用户有效地管理内存，实现整型变量的操作，这些变量不能用于数学运算
整型（无符号）	uint8 uint16 uint32	uint8(2)	
单精度	single	single(32.3)	单精度数值数组，单精度所需的存储空间较双精度小，但精度差，数值的范围小，单精度不能用于数学运算
双精度	double	32 double(44.5)	双精度数值数组，为最常用的 MATLAB 变量类型
稀疏矩阵	sparse	sparse(6)	稀疏双精度矩阵，稀疏矩阵只存储少数的非零元素，较常规矩阵的存储节约了大量的存储空间，稀疏矩阵引入了特殊的算法
单元数组	cell	{10,'h',3.4}	单元数组，单元数组元素的尺寸、性质可以不同
结构数组	struct	a.color='Red'	结构数组，结构数组包括域名，域中可以包括其他数组，与单元数组类似

在工作空间浏览器中，不同的数据类型有着不同的图标标识，图 2-1 中显示了部分数据类型。在图 2-1 中，每种数据都表达成矩阵的形式，其中 Size 代表矩阵中元素的个数，Bytes 代表矩阵在内存中所占字节数，Class 代表矩阵的类型。

Name	Size	Bytes	Class
a	1x1	2	char array
b	1x1	1	int8 array
c	1x1	1	uint8 array
d	1x1	4	single array
e	1x1	8	double array
f	1x3	204	cell array
g	1x1	464	struct array
h	10x1		java.awt.Frame[][]
k	1x1	16	function_handle array
m	5x5	84	sparse array

图 2-1 在工作空间浏览器中显示的数据类型

图 2-1 中 a 是字符矩阵、b 是有符号整型矩阵、c 是无符号整型矩阵、d 是单精度矩阵、e 是双精度矩阵、f 是单元数组、g 是结构数组、h 是 Java 类数组、k 是函数句柄矩阵、m 是稀疏矩阵。有一些不同类型的数据之间不能够直接运算，还有一些函数对数据的类型有严格的要求，这就需要对数据的类型进行强制转换。例如：uint8(24)就是把默认的双精度数据“24”转化为无符号 8 位整型数据“24”。对变量也可进行同样的操作。

**注意：**

MATLAB 用十进制表示一个常数，可采用日常记数法和科学记数法两种表示方法。对于角度的输入，则采用弧度制。

## 2.2 变量和数据操作

### 2.2.1 变量

#### 1. 变量的命名

变量就是在程序运行过程中，其数值可以变化的数据。它可代表一个或若干个内存单元（变量的地址）中的数据，为了对变量所对应的存储单元进行访问，需要给变量命名。在 MATLAB 中，变量名可以由字母、数字或下划线组成的字符序列，最多可包含 31 个字符，但第一个字符必须是字母。例如：myfile13、ab\_lcd、EXAMP1-12 等均为合法变量名，而 3dat、\_mydat、1234 等都不是合法字符。

#### 注意：

在 MATLAB 中，变量名区分字母的大小写，大小写不同的两个变量名被认为是两个不同的变量。另外，MATLAB 不支持汉字，汉字不能出现在变量名和文件名中。

#### 2. 赋值语句

MATLAB 赋值语句有两种格式。

- 变量=表达式
- 表达式

说明：其中表达式是用运算符把特殊字符、函数名、变量名等有关运算量连接起来的式子，其结果是一个矩阵。在第一种语句形式下，MATLAB 将右边表达式的值赋给左边的变量，而在第二种语句形式下，将表达式的值赋给 MATLAB 的预定义变量 ans。例如：

```
>> 1200/25
```

```
ans =  
    48
```

运算结果在命令窗口中显示出来。如果在语句的最后加分号，那么 MATLAB 仅仅执行赋值操作，不显示运算的结果，以抑制不必要的信息显示。如果运算的结果是一个很大的矩阵或根本不需要运算结果，则可以在语句的最后加上分号。如果表达式较长，在一行中放不下，则可以在行末键入三个小黑点表示的续行符（...），指明下一行为续行。例如：

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

计算结果为  $s=0.6532$ ，并没有显示出来。

#### 提示：

如果续行符前面是数字，直接使用续行符会出现错误，有两种解决的方法，一种是再加一个点（共四个点），另一种是先空一格然后再加续行符。

### 2.2.2 预定义变量

在 MATLAB 工作空间中，还驻留几个由系统本身定义的变量。它们有特定的含义，在使用时，应尽量避免对这些变量重新赋值。除前面出现过的 `ans` 外，还有一些常用的预定义变量，现将它们列于表 2-2 中。

表 2-2 预定义变量

预定义变量名	含 义
<code>ans</code>	计算结果的默认赋值变量
<code>eps</code>	容差变量，定义为 1.0 到最近浮点数的距离，在 pc 机上等于 $2^{-52}$
<code>Pi</code>	圆周率 $\pi$ 的近似值
<code>i, j</code>	虚数单位
<code>inf, Inf</code>	正无穷大，定义为 $(1/0)$
<code>NaN, nan</code>	非数。在 IEEE 运算规则中，它产生于 $0/0$ 、 $0 \times \infty$ 等的结果
<code>nargin</code>	函数输入参数个数
<code>nargout</code>	函数输出参数个数
<code>realmax</code>	最大正实数
<code>realmin</code>	最小正实数
<code>lasterr</code>	存放最新的错误信息
<code>lastwarn</code>	存放最新的警告信息

**注意：**

函数名、预定义变量名和自定义变量名可以相同，但在清除该自定义变量之前，相应的函数和预定义变量都无效。

### 2.2.3 内存变量的管理

MATLAB 工作空间浏览器（见图 2-1）专门用于内存变量的管理。在工作空间浏览器上可以显示所有内存变量的属性。当选中某个变量后，再单击〈delete〉键，可删除该变量；双击某个变量，将进入矩阵编辑器（如图 2-2 所示），可以直接观察变量中的具体元素值，也可以直接修改这些元素。但是 MATLAB6.1 的矩阵编辑器对矩阵的大小有要求，只能显示最多 10 000 个元素的矩阵，使用时可以把一个大矩阵分成几个小矩阵来分别显示。

### 2.2.4 常用数学函数

MATLAB 提供了许多数学函数，函数的自变量规定为矩阵变量，运算法则是将函数逐项作用于矩阵的元素上，因而运算的结果是一个与自变量相同维数的矩阵。MATLAB 的常用数学函数可分为 4 部分：三角函数、指数函数（包括对数函数）、复数函数、取整和求余函数（如表 2-3 所示）。



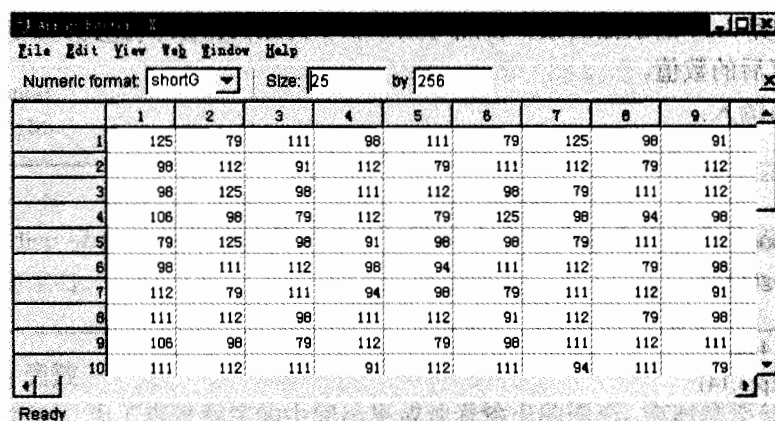


图 2-2 矩阵编辑器

表 2-3 MATLAB 的基本数学函数

	函数名称	功能		函数名称	功能
三角函数	sin	正弦	三角函数	sec	正割
	asin	反正弦		asec	反正割
	cos	余弦		csc	余割
	acos	反余弦		acsc	反余割
	tan	正切		cot	余切
	atan	反正切		acot	反余切
	atan2	第四象限的反正切			
指数函数	exp	以 e 为底的指数	指数函数	pow2	2 的幂次
	log	自然对数		sqrt	开平方
	log10	以 10 为底的对数		nextpow2	大于或等于变量的 2 的下一个幂次
	log2	以 2 为底的对数			
复数函数	abs	绝对值或复数的模	复数函数	real	复数的实部
	angle	相位角		unwrap	相位展开
	complex	由实部和虚部构造复数		isreal	是否为实数组
	conj	复数的共轭		cplxpair	整理为共轭对
	imag	复数的虚部			
取整函数	fix	朝零方向取整	取整函数	mod	模数（带符号余数）
	floor	朝负无穷方向取整		rem	除后取余数
	ceil	朝正无穷方向取整		sign	符号函数
	round	四舍五入到最近的整数		gcd	最大公倍数
	lcm	最小公倍数			

**注意：**

三角函数按弧度计算。另外， $\text{mod}(x, y)$  与  $y$  符号相同， $\text{rem}(x, y)$  与  $x$  符号相同；当  $x$  与  $y$  同号时， $\text{mod}(x, y)$  等于  $\text{rem}(x, y)$ 。

例 2-1 计算 3.14 的余弦函数、自然对数、以  $e$  为底的指数、朝零方向取整和朝正无穷大方向取整运算后的数值。

在命令窗口输入：

```
>> a=cos(3.14)
a =
    -1.0000
>> b=log(3.14)
b =
    1.1442
>> c=exp(3.14)
c =
    23.1039
>> d=fix(3.14)
d =
     3
>> e=ceil(3.14)
e =
     4
```

### 2.2.5 数据的输入输出函数

MATLAB 的数据输入输出方式包括从命令窗口的输入输出以及通过文件操作的输入输出。此外，为了顺应现代软件开发技术的潮流，MATLAB 还提供了图形界面（GUI）的输入输出方式。这里先介绍通过命令窗口的输入输出。

#### 1. input 函数

MATLAB 提供了一些输入输出函数，允许用户和计算机之间进行数据交换。如果用户想从键盘输入数据，则可以使用 input 函数来进行，该函数的调用格式为：

变量名=input（‘提示信息’，选项）；

说明：其中提示信息为一个字符串，用于提示用户输入什么样的数据。串中若有“\n”则表示换行输入。如果在 input 函数调用时采用's'选项，则允许用户输入一个字符串。例如：

```
>> a=input('How many apples\n','s')
How many apples
two apples    %通过键盘输入
a =

two apples
```

#### 2. disp 函数

MATLAB 提供的命令窗口输出函数主要是 disp 函数，其调用格式为：

disp（输出项）

说明：其中输出项既可以是字符串，也可以是矩阵。

例如输出上例创建的字符矩阵 **a**。

```
>> disp(a)
two apples
```

**注意：**

用 `disp` 函数显示矩阵时将不显示矩阵的名字，而且其格式更紧密，不留没有意义的空行。

### 3. pause 函数

当程序运行时，为了查看程序的中间结果或观看输出的图形，有时需要暂停程序的执行。这时可以使用 `pause` 函数，其调用格式为：

```
pause (n)
```

说明：*n* 是一个常数，表示延迟多少秒。如果省略延迟时间，直接使用 `pause`，则将暂停程序，直到用户按任意键后程序继续执行。若要强行中止程序的运行可使用 `<Ctrl> + <C>` 命令。

### 4. save 函数

`save` 命令是将 MATLAB 工作空间中的变量存入磁盘。具体格式如下。

#### ● save

说明：将当前 MATLAB 工作空间中所有变量以二进制格式存入名为 “matlab.mat”（默认的文件名）的文件中。

#### ● save dfile (文件名)

说明：将当前工作空间中所有变量以二进制格式存入名为 “dfile.mat” 文件，扩展名自动产生。

#### ● save dfile x

说明：只把变量 *x* 以二进制格式存入 “dfile.mat” 文件，扩展名自动产生。

#### ● save dfile.dat x -ascii

说明：将变量 *x* 以 8 位 ASCII 码形式存入 “dfile.mat” 文件。

#### ● save dfile.dat x -ascii -double

说明：将变量 *x* 以 16 位 ASCII 码形式存入 “dfile.mat” 文件。

#### ● save (fname, 'x', '-ascii')

说明：*fname* 是一个预先定义好的包含文件名的字符串，该用法将变量 *x* 以 ASCII 码格式存入由 *fname* 定义的文件中，由于在这种用法中，文件名是一个字符变量，因此可以方便地通过编程的方法存储一系列数据文件。

**注意：**

当 `save` 函数使用 `-ascii` 选项时，只能保存当前工作空间中的数值型变量，如有其他类型的变量不会被保存，同时 MATLAB 将发出警告。

## 5. load 函数

load 命令是将磁盘上的数据读入到工作空间。具体格式如下所示。

- load

说明：把磁盘文件“matlab.mat”（默认的文件名）的内容读入内存，由于存储.mat 文件时已包含了变量名的信息，因此调回时已直接将原变量信息带入，不需要重新赋值变量。

- load dfile

说明：把磁盘文件“dfile.mat”的内容读入内存。

- load dfile.dat

说明：把磁盘文件“dfile.dat”的内容读入内存，这是一个 ASCII 码文件，系统自动将文件名（dfile）定义为变量名。

- x=load (fname)

说明：fname 是一个预先定义好的包含文件名的字符串，将由 fname 定义文件名的数据文件读入变量 x 中，使用这种方法可以通过编程方便地调入一系列数据文件。

例 2-2 定义三个变量  $a=2$ ,  $b=4$ ,  $c=6$ ，全部存入一个文件中，再把  $a$ 、 $b$  存入另一个文件中，清空工作空间后，检查工作空间，再调入变量  $a$ ，再一次检查工作空间。

```
>> a=2;
>> b=4;
>> c=6;
>> save mydate1
>> save mydat2 a b
>> clear           %清空工作空间
>> whos           %检查工作空间，已没有任何变量
>>
>> load mydat2 a
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array

Grand total is 1 elements using 8 bytes

## 2.3 矩阵

### 2.3.1 矩阵的建立

MATLAB 把矩阵作为基本运算对象。数值（标量）可看作  $1 \times 1$  的矩阵，矢量（一维数组）可看成  $n \times 1$  或  $1 \times n$  阶的矩阵。在 MATLAB 中，不需要对矩阵的维数和类型进行说明，MATLAB 会根据用户所输入的内容进行配置，创建矩阵有以下三种方法。

#### 1. 直接输入创建矩阵

可以通过键入矩阵中每个元素的值来建立并输入一个矩阵，只须以左方括号开始，以逗

号或空格为间隔输入元素值，行与行之间用分号隔开，最后以右方括号结尾即可。当矩阵中的元素个数比较少时，这种方法非常适用。另外，用单引号界定后的字符或字符串可创建字符矩阵。

**例 2-3** 创建  $3 \times 3$  数值矩阵  $a$ 、 $b$  和字符矩阵  $c$ 。

```
>> a=[1, 2, 3;4, 5, 6;7, 8, 9]
a =
     1     2     3
     4     5     6
     7     8     9
>> b=[1.5 12 3.3;14 55 0.6;-7 -0.8 11.9]
b =
  1.5000  12.0000   3.3000
 14.0000  55.0000   0.6000
 -7.0000  -0.8000  11.9000
>> c='string'
c =
string
```

**提示：**

当矩阵较大时，可以分行输入，用〈Enter〉键代替分号，这样的输入形式比较接近线性代数中的矩阵。任何矩阵的元素内部都不能有空格，否则会被 MATLAB 认为是两个元素而出错。

## 2. 向量法创建矩阵

向量可以由冒号和数字产生。其格式为：

向量名=初值：增量：终值

说明：向量的全部成员是从初值开始，以增量为步长，直到不超过终值的所有元素所构成的序列。默认步长为“1”。

当矩阵中的元素很多且有规律时，则可以通过向量来建立一个矩阵。其基本格式为：

矩阵名=向量

**例 2-4** 建立一个 10 以内的奇数矩阵。

```
>> a=1:2:10
a =
     1     3     5     7     9
```

## 3. 函数法创建矩阵

利用内部语句和函数还可以快速产生一些特别有用的矩阵，如单位矩阵、随机矩阵、零矩阵等，如表 2-4 所示。

表 2-4 特殊矩阵

命 令	说 明
<code>[]</code>	空矩阵
<code>eye</code>	单位矩阵
<code>ones</code>	全部元素都为 1 的常数矩阵
<code>rand</code>	元素服从 0 和 1 之间均匀分布的随机矩阵
<code>randn</code>	元素服从零均值单位方差正态分布的随机矩阵
<code>zeros</code>	全部元素都为 0 的矩阵

**注意:**

当某一项操作无结果时, MATLAB 返回一个空矩阵。空矩阵的大小为零, 但其确实存在于工作空间中, 可以通过变量名访问。

**例 2-5** 建立空矩阵 *a*、单位矩阵 *b*、常数矩阵 *c*、均匀分布随机矩阵 *d*、正态分布的随机矩阵 *e*、零矩阵 *f*。

```
>> a=[]
a =
[]
>> b=eye(3,4)

b =
     1     0     0     0
     0     1     0     0
     0     0     1     0
>> c=4*ones(5)

c =
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
     4     4     4     4     4
>> d=rand(2,3)

d =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
>> e=randn(2,3)

e =
   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909
>> f=zeros(3,4)
```

```
f =
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

**提示:**

输入后的矩阵将保存在 MATLAB 的工作空间中, 并可以随时被调用, 如果用户不用“clear”命令清除它, 或对它重新赋值, 该矩阵将一直保存在工作空间中直到 MATLAB 被关闭为止。另外, 矩阵函数中只有一个参数, 则为方阵。

### 2.3.2 复数与复数矩阵

MATLAB 允许在运算和函数中使用复数或复数矩阵。复数的表示借助于特殊的字符 i 或 j, 其值在工作空间中都显示为 0+1.0000i。

1. 输入复数可由以下两种方式输入

● >> z=1+2i

```
z =
1.0000 + 2.0000i
```

● >> z=3\*exp(i\*3.14)

```
z =
-3.0000 + 0.0048i
```

其中 3.14 为复数幅角的弧度, 3 为复数的模。

2. 输入复数矩阵有下列两种方法

● >> a=[1+2i 3+4i; 5+6i 7+8i]

```
a =
1.0000 + 2.0000i    3.0000 + 4.0000i
5.0000 + 6.0000i    7.0000 + 8.0000i
```

● >> a=[1 3; 5 7]+i\*[2 4; 6 8]

```
a =
1.0000 + 2.0000i    3.0000 + 4.0000i
5.0000 + 6.0000i    7.0000 + 8.0000i
```

两式具有相同的结果。

**注意:**

当复数的虚部为一个确定的数值 (不是变量或矩阵) 时, 输入时可以省略 i (或 j) 前面的“\*”符号。另外, 如果 i、j 被定义与其他变量 (例如在程序设计中常常习惯于将 i 和 j 作为循环变量), 则应定义另一个新的复数单位, 如 i1=sqrt(-1), z=3+4\*i1, 这里将 i1 定义成新的复数单位。

### 2.3.3 矩阵的基本运算

#### 1. 矩阵与标量的运算

运算包括加、减、乘、除和乘方运算。矩阵与标量运算是矩阵的每个元素对该标量的运算。MATLAB 用符号“^”计算乘方时，按照矩阵运算规则计算乘方，要求矩阵为方矩阵；用符号“./”计算乘方时，按照数组运算规则计算乘方，对矩阵没有限制。

例 2-6 已知矩阵  $a = [1\ 2\ 3; 4\ 5\ 6]$ ，标量  $b = 3$ ，计算  $a+b$ 、 $a*b$ 、 $a/b$  和  $a.^b$ 。

```
>> a=[1 2 3;4 5 6];
>> b=3;
>> c=a+b
c =
     4     5     6
     7     8     9
>> d=a*b
d =
     3     6     9
    12    15    18
>> e=a/b
f =
    0.3333    0.6667    1.0000
    1.3333    1.6667    2.0000
>> g=a.^b
g =
     1     8    27
    64   125   216
```

#### 2. 矩阵与矩阵的运算

##### ● 矩阵加减法运算

两个矩阵的维数完全相同时，可以进行矩阵加减法运算。它会自动地使得  $A$  和  $B$  矩阵的相应元素相加减。如果两个矩阵的维数不相等，则 MATLAB 将自动地给出错误信息，提示两个矩阵的维数不相等。

##### ● 矩阵乘法运算

两个矩阵的维数相容时（ $A$  的列数等于  $B$  的行数），可以进行  $A$  乘  $B$  的乘法运算。

##### ● 矩阵除法运算

矩阵的除法运算包括左除和右除两种运算。其中

左除： $A \setminus B = A^{-1}B$ ， $A$  为方矩阵

右除： $A / B = AB^{-1}$ ， $B$  为方矩阵

可见，左除和右除的运算过程以及对矩阵的要求是不一样的，其结果也应该不同。

##### ● 点运算

按照数组运算规则计算，两个矩阵之间的点运算是该矩阵对应元素的直接运算，要求参加运算的矩阵的大小必须相同。有“.\*”、“./”和“.\”三种运算符。



例 2-7 已知矩阵  $a = [1 \ 2; \ 3 \ 4]$ ，矩阵  $b = [5 \ 6; \ 7 \ 8]$ ，求  $a*b$ 、 $a.*b$ 、 $a\b b$ 、 $a/b$ 、 $a.\b$  和  $a./b$  的运算结果。

```
>> a=[1 2;3 4];
>> b=[5 6;7 8];
>> c=a*b
c =
    19    22
    43    50
>> d=a.*b           %矩阵点乘
d =
     5     12
    21     32
>> f=a\b           %矩阵左除
f =
   -3.0000   -4.0000
    4.0000    5.0000
>> e=a/b           %矩阵右除
e =
     3.0000   -2.0000
     2.0000   -1.0000
>> h=a.\b           %矩阵点左除
h =
     5.0000     3.0000
     2.3333     2.0000
>> k=a./b           %矩阵点右除
k =
     0.2000     0.3333
     0.4286     0.5000
```

提示：

矩阵的除法运算实际上是求  $AX=B$  的解的过程。当  $A$  为非奇异矩阵时，结果是最小二乘解，即矩阵除法可以找到使  $\|AX-B\|$  绝对值最小的  $X$ 。

### 2.3.4 矩阵的操作

MATLAB 中提供了很多简便、智能的方式，可以对矩阵进行元素操作、提取子块、合并矩阵、转置等操作。

#### 1. 元素操作

MATLAB 允许用户对一个矩阵的单个元素进行操作，可以通过元素的下标进行（行、列的序号是从 1 开始的），修改某些元素的值不会影响其他元素的值。如果用户给出的行下标或列下标超出了原来的行数或列数，将自动扩展原来的矩阵，并将扩展后未赋值的矩阵元素置为零。

## 2. 提取子块

提取矩阵的某一部分，可以使用冒号表达式。在 MATLAB 中，冒号“:”表示“全部”。

**例 2-8** 输入一个  $4 \times 3$  的矩阵，选出前三行构成一个新矩阵，再选出前两列构成另一个矩阵。

```
>> a=[1 2 3;4 5 6;7 8 9;10 11 12];
>> b=a(1:3,:)
b =
     1     2     3
     4     5     6
     7     8     9
>> c=a(:,1:2)
c =
     1     2
     4     5
     7     8
```

## 3. 矩阵合并

把两个矩阵合并成一个大矩阵，有两种形式。

### ● $C=[A; B]$

说明： $A$  矩阵与  $B$  矩阵的列数必须相同， $B$  矩阵补在  $A$  矩阵的下面。

### ● $C=[A, B]$

说明： $A$  矩阵与  $B$  矩阵的行数必须相同， $B$  矩阵补在  $A$  矩阵的右面。

## 4. 矩阵的转置

用符号“'”（单引号）可以进行矩阵的转置运算。

## 5. 矩阵的展开

矩阵的展开是按照矩阵在内存中的实际存放形式展开的。矩阵的元素在内存中是按列存放的，即先存放第一列，接着存放第二列……把一个矩阵内的所有元素统一展开成一个列向量，其指令格式为：

$B=A(:)$

**例 2-9** 把矩阵  $A = [1\ 3\ 5; 7\ 9\ 11]$  和矩阵  $B = [2\ 4\ 6]$  合并成一个矩阵，再转置后展开。

```
>> A=[1 3 5;7 9 11];
>> B=[2 4 6];
>> C=[A;B]
C =
     1     3     5
     7     9    11
     2     4     6
>> C=C'
C =
     1     7     2
     3     9     4
     5    11     6
```

```

      3     9     4
      5    11     6
>> D=C(:)
D =
     1
     3
     5
     7
     9
    11
     2
     4
     6

```

## 2.4 关系运算和逻辑运算

### 2.4.1 关系运算符

MATLAB 共有 6 种关系运算符用于关系运算，见表 2-5。

表 2-5 关系运算符

运 算 符	说 明	运 算 符	说 明
<	小于	<=	小于或等于
>	大于	>=	大于或等于
==	等于	~=	不等于

MATLAB 关系运算符能用来比较两个同样大小的矩阵，或用来比较一个矩阵和一个标量。在后一种情况，标量和矩阵中的每一个元素相比较，结果是一个与原矩阵大小一样的矩阵。

例 2-10 已知矩阵  $A = [1\ 3\ 5\ 7\ 9]$ ，找出大于 4 的元素的位置。

```

>> A=[1,3,5,7,9];
>> b=A>4
b =
     0     0     1     1     1

```

可见，0 出现在  $A \leq 4$  的地方，1 出现在  $A > 4$  的地方，形成了一个与原矩阵同样大小的新的矩阵。

#### 提示：

=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；= 被用来将运算的结果赋给一个变量。

## 2.4.2 逻辑运算符

MATLAB 提供了三种逻辑运算符。见表 2-6。

表 2-6 逻辑运算符

名 称	运 算 符	说 明
与运算	&	两个元素同为非零时，结果为 1；否则为 0。
或运算		两个元素同为零时，结果为 0；否则为 1。
非运算	~	单目运算符。元素为零，结果为 1；元素为非零，结果为 0。

逻辑运算的方法与关系运算相类似，都是对同阶矩阵中的对应元素进行逻辑运算，如果其中一个为标量，则标量逐个与矩阵中的每一个元素进行逻辑运算。

例 2-11 建立  $A$ 、 $B$  两个矩阵，计算  $A \& B$ 、 $A | B$  和  $\sim B$ 。

```
>> A=[1 -3 5;0 1 0];
>> B=[1 50 0;-3 0.5 12];
>> C=A&B
C =
     1     1     0
     0     1     0
>> D=A|B
D =
     1     1     1
     1     1     1
>> ~B
ans =
     0     0     1
     0     0     0
```

## 2.4.3 其他关系与逻辑函数

除了上面的关系与逻辑运算符以外，MATLAB 提供了一些其他关系与逻辑函数，见表 2-7。

表 2-7 其他关系与逻辑函数

函 数 格 式	说 明
$\text{xor}(x, y)$	异或运算。 $x$ 和 $y$ 都是零（假）或都是非零（真）返回 0， $x$ 或 $y$ 不同则返回 1
$\text{any}(x)$	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。否则返回零
$\text{all}(x)$	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。否则返回零

例 2-12 已知矩阵  $a = [0 -3.4 5; 0 11 0]$  和矩阵  $b = [1 50 0; -3.14 0.5 12]$ ，查看  $a$  的零元素的情况，并与  $b$  进行异或运算。

```

>> a=[0 -3.4 5;0 11 0];
>> b=[1 50 0;-3.14 0.5 12];
>> any(a)
ans =
     0     1     1
>> all(a)
ans =
     0     1     0
>> xor(a,b)
ans =
     1     0     1
     1     0     1

```

## 2.5 文件操作

文件操作是一种重要的输入输出方式，即从数据文件读取数据或将结果写入数据文件。MATLAB 提供了一系列低层输入输出函数，专门用于文件操作。

### 2.5.1 文件的打开与关闭

#### 1. 打开文件

在读写文件之前，必须先用 `fopen` 函数打开或创建文件，并指定对该文件进行的操作方式。`fopen` 函数的调用格式为：

```
fid=fopen(文件名, '打开方式')
```

说明：其中 `fid` 用于存储文件句柄值，如果返回的句柄值大于 0，则说明文件打开成功。文件名用字符串形式，表示待打开的数据文件。常见的打开方式如下。

- ‘r’：只读方式打开文件（默认的方式），该文件必须已存在。
- ‘r+’：读写方式打开文件，打开后先读后写。该文件必须已存在。
- ‘w’：打开后写入数据。该文件已存在则更新；不存在则创建。
- ‘w+’：读写方式打开文件。先读后写。该文件已存在则更新，不存在则创建。
- ‘a’：在打开的文件末端添加数据。文件不存在则创建。
- ‘a+’：打开文件后，先读入数据再添加数据。文件不存在则创建。

另外，在这些字符串后添加一个“t”，如‘rt’或‘wt+’，则将该文件以文本方式打开；如果添加的是“b”，则以二进制格式打开，这也是 `fopen` 函数默认的打开方式。

#### 2. 关闭文件

文件在进行完读、写等操作后，应及时关闭，以免数据丢失。关闭文件用 `fclose` 函数，调用格式为：

```
sta=fclose(fid)
```

说明：该函数关闭 `fid` 所表示的文件。`sta` 表示关闭文件操作的返回代码，若关闭成功，返回 0，否则返回-1。如果要关闭所有已打开的文件用 `fclose('all')`。

## 2.5.2 二进制文件的读写操作

### 1. 写二进制文件

`fwrite` 函数按照指定的数据精度将矩阵中的元素写入到文件中。其调用格式为：

```
COUNT=fwrite(fid, A, precision)
```

说明：其中 `COUNT` 返回所写的数据元素个数（可默认），`fid` 为文件句柄，`A` 用来存放写入文件的数据，`precision` 代表数据精度，常用的数据精度有 `char`、`uchar`、`int`、`long`、`float`、`double` 等。默认数据精度为 `uchar`，即无符号字符格式。

**例 2-13** 将一个二进制矩阵存入磁盘文件中。

```
>> a=[1 2 3 4 5 6 7 8 9];
>> fid=fopen('d:\test.bin','wb')    %以二进制数据写入方式打开文件
fid =
     3    %其值大于 0，表示打开成功
>> fwrite(fid,a,'double')
ans =
     9    %表示写入了 9 个数据
>> fclose(fid)
ans =
     0    %表示关闭成功
```

### 2. 读二进制文件

`fread` 函数可以读取二进制文件的数据，并将数据存入矩阵。其调用格式为：

```
[A, COUNT]=fread(fid, size, precision)
```

说明：其中 `A` 是用于存放读取数据的矩阵、`COUNT` 是返回所读取的数据元素个数、`fid` 为文件句柄、`size` 为可选项。若不选用则读取整个文件内容；若选用则它的值可以是下列值：`N`（读取 `N` 个元素到一个列向量）、`inf`（读取整个文件）、`[M, N]`（读数据到  $M \times N$  的矩阵中，数据按列存放）。`precision` 用于控制所写数据的精度，其形式与 `fwrite` 函数相同。

**例 2-14** 将上例数据文件中的前 5 个数据读入到矩阵 `b` 中。

```
>> fid=fopen('d:\test.bin','rb')    %以读方式打开文件
fid =
     3
>> b=fread(fid,5,'double')    %读入前 5 个数据
b =
     1
     2
     3
     4
     5
>> fclose(fid);
```

注意:

矩阵在内存中是按列存放的, 所以, 存入文件的行向量读出后, 变成了列向量。在进行文件操作时, 一定要小心, 必要时可把矩阵转置。

### 2.5.3 文本文件的读写操作

#### 1. 读文本文件

`fscanf` 函数可以读取文本文件的内容, 并按指定格式存入矩阵。其调用格式为:

```
[A, COUNT]=fscanf(fid, format, size)
```

说明: 其中  $A$  用来存放读取的数据,  $COUNT$  返回所读取的数据元素个数,  $fid$  为文件句柄,  $format$  用来控制读取的数据格式, 由 % 加上格式符组成。常见的格式符有:  $d$  (整型)、 $f$  (浮点型)、 $s$  (字符串型)、 $c$  (字符型) 等, 在 % 与格式符之间还可以插入附加格式说明符, 如数据宽度说明等。size 为可选项, 决定矩阵  $A$  中数据的排列形式, 它可以取下列值:  $N$  (读取  $N$  个元素到一个列向量)、 $inf$  (读取整个文件)、 $[M, N]$  (读数据到  $M \times N$  的矩阵中, 数据按列存放)。

#### 2. 写文本文件

`fprintf` 函数可以将数据按指定格式写入到文本文件中。其调用格式为:

```
fprintf(fid, format, A)
```

说明:  $fid$  为文件句柄, 指定要写入数据的文件,  $format$  是用来控制所写数据格式的格式符, 与 `fscanf` 函数相同,  $A$  是用来存放数据的矩阵。

例 2-15 创建一个字符矩阵并存入磁盘, 再读出赋值给另一个矩阵。

```
>> a='string';
>> fid=fopen('d:\char1.txt','w');
>> fprintf(fid,'%s',a);
>> fclose(fid);
>> fid1=fopen('d:\char1.txt','rt');
>> fid1=fopen('d:\char1.txt','rt');
>> b=fscanf(fid1,'%s')
b =
    string
```

### 2.5.4 图像文件的读写操作

#### 1. imread 函数

用于从文件中读入图像, 图像可以是 `bmp` (Windows 位图文件)、`hdf` (层次数据格式图像文件)、`jpg` 或 `jpeg` (压缩图像文件)、`pcx` (Windows 画笔图像文件)、`tif` 或 `tiff` (标签图像格式文件) 等。其函数格式如下:

● `A = imread` (文件名, “图像文件格式”)

说明: `A` 为无符号 8 位整数 (uint8) 矩阵。如果读入文件为灰度图像, 则 `A` 为二维矩阵; 如果读入图像为真彩色 RGB 图像, 则 `A` 为三维矩阵。

● `[A, map]=imread` (文件名, “图像文件格式”)

说明: `map` 为双精度浮点数 (double), 其值在 0~1 之间。表示图像的颜色值。

## 2. imwrite 函数

用于将图像写入文件, 图像格式同 `imread` 函数。格式如下:

● `imwrite` (`A`, 文件名, “图像文件格式”)

● `imwrite` (`A`, `map`, 文件名, “图像文件格式”)

说明: 与 `imread` 函数相同。

例 2-16 读入图像文件名为 “cameraman.tif” 的图像文件, 改文件名为 “image”、改格式为 jpg 后, 存入 D 盘。

```
>> A=imread('cameraman.tif','tif');
>> imwrite(A,'d:\image.jpg','jpg');
```

打开 D 盘, 可以看到该图像。

## 2.6 上机实践

### 2.6.1 跟我学

例 2-17 计算  $\frac{\sin(|x|+|y|)}{\sqrt{\cos(|x+y|)}}$  的值, 其中  $x=-1.42$ ,  $y=0.54$ 。

```
>> x=-1.42;y=0.52;
>> sin(abs(x)+abs(y))/sqrt(cos(abs(x+y)))    %abs 为取绝对值、sqrt 为开根号函数

ans =
    1.1829
```

例 2-18 求解方程  $ax^2+bx+c=0$  的根, 其中:  $a=1$ ,  $b=2$ ,  $c=3$ 。

```
>> a=1;b=2;c=3;
>> d=sqrt(b^2-4*a*c);    %利用求根公式
>> x1=(-b+d)/(2*a)

x1 =
    -1.0000 + 1.4142i
>> x2=(-b-d)/(2*a)

x2 =
```



-1.0000 - 1.4142i

例 2-19 建立矩阵  $A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ -1 & 0 & -2 \\ -3 & 0 & 0 \end{bmatrix}$ 、 $B = \begin{bmatrix} -1 & 3 \\ -2 & 2 \\ 2 & 1 \end{bmatrix}$  求  $C=A \times B$ ，并把  $C$  转置后存盘。

```
>> A=[1 3 5;2 4 6;-1 0 -2;-3 0 0]; %建立矩阵
>> B=[-1 3;-2 2;2 1];
>> C=A*B

C =
     3    14
     2    20
    -3     5
     3    -9
>> C=C' %矩阵转置

C =
     3     2    -3     3
    14    20    -5    -9
>> save mydat C %存入 mydat 文件中
>> clear %清除工作空间变量。观察工作空间窗口 (Workspace)
>> load mydat C %读入数据 (比较工作空间窗口的变化)
```

例 2-20 建立矩阵  $U=[1 \ -2 \ 0 \ -1.4 \ 0 \ 0 \ 8]$  和矩阵  $V=[-1 \ 0 \ 0 \ 7.8 \ -3 \ 0 \ -6.2]$ ，计算  $U \geq V$ ， $U \& V$  和  $\text{xor}(U,V)$  的值，并查看  $U \setminus V$  的结果是否存在非零元素。

```
>> U=[1 -2 0 -1.4 0 0 8];
>> V=[-1 0 0 7.8 -3 0 -6.2];
>> U>=V

ans =
     1     0     1     0     1     1     1
>> U&V

ans =
     1     0     0     1     0     0     1
>> xor(U,V)

ans =
     0     1     0     0     1     0     0
>> any(U\V)

ans =
```

例 2-21 创建整数随机矩阵  $B$ ，并以二进制格式存入 D 盘，文件名是 test1.bin。

```
>> A=randn(3,4)           %先建立随机矩阵 A

A =
   -0.5883    0.1139   -0.0956   -1.3362
    2.1832    1.0668   -0.8323    0.7143
   -0.1364    0.0593    0.2944    1.6236

>> B=round(A)             %对 A 矩阵取整，变成整数矩阵

B =
    -1     0     0    -1
     2     1    -1     1
     0     0     0     2

>> fid=fopen('d:\test1.bin','wb');   %以写方式打开文件
>> fwrite(fid,B)
>> fclose(fid)
```

## 2.6.2 自己练

1. 已知  $A=2.1$ ,  $B=-4.5$ ,  $C=6$ ,  $D=3.5$ ,  $E=-5$ , 计算  $\arctan\left(\frac{2\pi A + \frac{E}{2\pi BC}}{D}\right)$  的值。

2. 已知矩阵  $A=\begin{bmatrix} 1 & 0 & -1 \\ 2 & 4 & 1 \\ -2 & 0 & 5 \end{bmatrix}$ 、 $B=\begin{bmatrix} 0 & -1 & 0 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$  求  $2A+B$ 、 $A^2-3B$ 、 $A*B$ 、 $B*A$ 、 $A./B$ 、 $A\backslash B$ 、 $A/B$ 、 $A\backslash B$  和  $A./B$ 。

$A/B$ 、 $A\backslash B$  和  $A./B$ 。

- 利用函数产生  $3\times 4$  阶单位矩阵和全部元素都是 4.5 的  $4\times 4$  阶常数矩阵。
- 利用函数产生  $5\times 5$  阶随机分布的矩阵和  $5\times 5$  阶正态分布的随机矩阵。
- 利用画图软件画一幅画，存盘后，读入 MATLAB 工作空间，并用矩阵编辑器查看这幅图画的像素值的分布情况。

## 2.7 本章小结

MATLAB 作为优秀的数学计算和分析软件，其中以矩阵运算为代表的基本运算功能是它的核心和基础。本章主要介绍了 MATLAB 的数据结构、矩阵、关系运算、逻辑运算和文件的操作。通过本章的学习，为后续章节的学习做好准备。本章涉及到的函数（或命令）列表 2-8 如下。

表 2-8 本章主要函数一览

函数（或命令）名称	说 明
input	提示用户输入
disp	显示矩阵或文本
pause	等待用户响应
save	保存工作空间变量
load	从磁盘文件中恢复变量
xor	异或运算
all	向量的所有元素为真，则其值为真
any	向量的任一元素为真，则其值为真
fopen	打开文件
fclose	关闭文件
fwrite	二进制数据写入到文件
fread	从文件中读二进制数据
fprintf	将格式化数据写入到文件
fscanf	从文件中读格式化的数据
imread	图像文件读入
imwrite	图像写出

## 2.8 习题

1. MATLAB 的数据结构特点？
2. 创建矩阵的方法有哪些？
3. 矩阵运算的乘和点乘、左除和右除、左除和点左除、右除和点右除的区别？
4. 已知  $a = 4.96$ ,  $b = 8.11$ , 计算  $\frac{e^{a+b}}{\lg(a+b)}$  的值。
5. 已知三角形的三个边  $a = 9.6$ ,  $b = 13.7$ ,  $c = 19.4$ , 求三角形的面积。  
利用公式  $area = \sqrt{s(s-a)(s-b)(s-c)}$ , 其中  $s = (a+b+c) / 2$ 。

6. 创建矩阵  $a = \begin{bmatrix} -1 & 0 & -6 & 8 \\ -9 & 4 & 0 & 12.3 \\ 0 & 0 & -5.1 & -2 \\ 0 & -23 & 0 & -7 \end{bmatrix}$ , 取出其前两列构成矩阵  $b$ , 取出其前两行构成

矩阵  $c$ , 转置矩阵  $b$  构成矩阵  $d$ , 计算  $a*b$ 、 $c<d$ 、 $c\&d$ 、 $cld$  和  $\sim cl\sim d$  的值。

7. 已知矩阵

$$A = \begin{bmatrix} 8 & 9 & 5 \\ 36 & -7 & 11 \\ 21 & -8 & 5 \end{bmatrix}, B = \begin{bmatrix} -1 & 3 & -2 \\ 2 & 0 & 3 \\ -3 & 1 & 9 \end{bmatrix}$$

---

求下列表达式的值:

(1)  $A+5*B$  和  $A-B$

(2)  $A*B$  和  $A.*B$

(3)  $A^3$  和  $A.^3$

(4)  $A/B$  和  $B\backslash A$

(5)  $[A,B]$

8. 已知  $A=1:9$ ,  $B=10:-1:1$ , 下列表达式的值分别是多少?

(1)  $A==B$

(2)  $A \leq 5$

(3)  $A > 3 \& A < 7$

(4)  $\text{find}(B > 3 \& B < 7)$

## 第3章 MATLAB 程序设计

在本章中，读者可以学会 M 文件和函数文件的建立方法，掌握选择结构语句和循环语句，能够读懂 MATLAB 的程序，并且可以编写简单的 MATLAB 程序。本章重点是 3.2 程序流程语句，难点是 3.3 函数文件。

### 3.1 M 命令文件

MATLAB 命令有两种执行方式：一种是交互式的命令执行方式，用户在命令窗口逐条输入命令，MATLAB 逐条解释执行，这种方式操作简单、直观，但速度慢，中间过程无法保留；另一种是 M 命令文件的程序设计方式，用户将有关命令编成程序存储在一个文件（扩展名为 .m）中，MATLAB 会自动依次执行该文件中的命令，这种方式可编写调试复杂的程序，是实际应用中主要的执行方式。

#### 3.1.1 M 文件的建立


M 文件是一个文本文件，它可以用任何编辑程序来建立和编辑，而一般常用且最为方便的是使用 MATLAB 提供的文本编辑器。

##### 1. 建立新的 M 文件

为建立新的 M 文件，启动 MATLAB 文本编辑器有 3 种方法。

（1）菜单操作：从 MATLAB 操作桌面的“File”菜单中选择“New”菜单项，再选择“M-file”命令，屏幕将出现 MATLAB 文本编辑器的窗口。

（2）命令操作：在 MATLAB 命令窗口输入命令“edit”，按〈Enter〉键后，即可启动 MATLAB 文本编辑器。

（3）命令按钮操作：单击 MATLAB 命令窗口工具栏上的新建命令按钮，启动 MATLAB 文本编辑器后，文本编辑器窗口如图 3-1 所示。

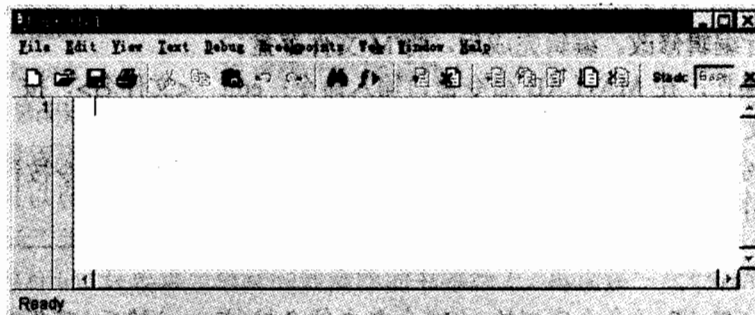


图 3-1 MATLAB 文本编辑器

MATLAB 文本编辑器是一个集编辑与调试两种功能于一体的工具环境。利用它不仅可

以完成基本的文本编辑操作，还可以对 M 文件进行调试。

启动 MATLAB 文本编辑器后，在文档窗口中输入 M 文件的内容，输入完毕后，选择文本编辑器窗口“File”菜单的“Save”或“Save As”命令存盘，默认名字是“Untitled”。注意，M 文件存放的位置一般是 MATLAB 默认的用户工作目录，当然也可以是别的目录。如果要选择别的目录，则应该将该目录设定为当前目录或将其加到搜索路径中，以便于查找。

### 例 3-1 编写一个命令文件，将变量 a, b 值互换。

首先打开文本编辑器，输入以下内容后，按〈F5〉或在 Debug 菜单中选择“Save and Run”命令项，以文件名“myfile.m”存盘。

```
clear          %清除工作空间变量
clc           %清屏幕

a=[1 3 4 7 9]; %建立 a 矩阵
b=[2 4 6 8 10]; %建立 b 矩阵
c=a;          %矩阵 a 与矩阵 b 交换，设中间变量 c
a=b;
b=c;
a             %输出 a 矩阵、b 矩阵
b
```

然后激活 MATLAB 的命令窗口，可看到输出为：

```
>>
a =

     2     4     6     8    10
b =


     1     3     4     7     9
```

## 2. 打开已有的 M 文件

打开已有的 M 文件，也有 3 种方法。

(1) 菜单操作：从 MATLAB 命令窗口的“File”菜单中选择“Open”命令，则屏幕出现“Open”对话框，在文件名对话框中选所需打开的 M 文件，在文本编辑窗口可以对打开的 M 文件进行编辑修改。编辑完成后，将 M 文件存盘。

(2) 命令操作：在 MATLAB 命令窗口输入命令：edit <文件名>，按〈Enter〉键后则打开指定的 M 文件。

(3) 命令按钮操作：单击 MATLAB 命令窗口工具栏上的打开命令按钮，再从弹出的对话框中选择所需打开的 M 文件。

#### 提示：

在 MATLAB 的命令窗口输入 M 命令文件名按〈Enter〉键后，可用命令 who-whos 查看工作空间中的变量。

### 3.1.2 M 文件的调试

#### 1. 文本编辑器窗口

在文本编辑器窗口菜单栏和工具栏的下面有三个区域，右侧的大区域是程序窗口，用于编写程序；最左面区域显示的是行号，每行都有数字，包括空行，行号是自动出现的，随着命令行的增加而增加；在行号和程序窗口之间的区域上有一些小横线，这些横线只有在可执行行上才有，而空行、注释行、函数定义行等非执行行的前面都没有，在进行程序调试时，可以直接在这些横线上点击鼠标以设置或去掉断点，图 3-2 中的菱形点即为断点。

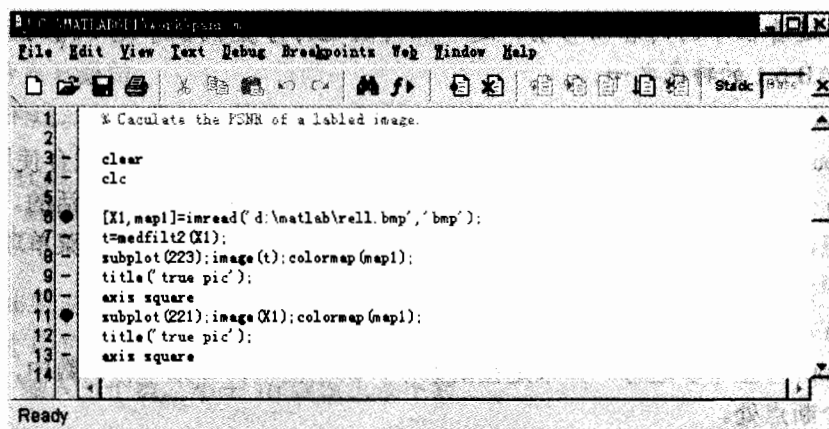


图 3-2 文本编辑器窗口

#### 2. 文本编辑器窗口的调试功能

在编辑调试器的菜单中有许多通用的菜单选项，如“File”，“Window”，“Help”等，这些菜单的使用方法与其他软件的使用方法相似，这里就不再介绍，下面介绍几个有特定功能的菜单项。

(1) Edit 菜单：该菜单中有一组特别的操作用于程序行的查找，这些操作对于程序内容的查找和修改是非常有用的。

- Go To Line... 到达指定行号处
- Set/Clear Bookmark 设置或清除标签
- Next Bookmark 找到下一个标签处
- Prev Bookmark 找到前一个标签处
- Set/Clear Bookmark 用来设置或清除书签，首先将光标移到设置书签的行上，然后选择该项，在光标所在的程序行前面就会出现一个蓝色的矩形书签标记。一个程序中可以设置多个书签标记，通过选择菜单中的“Next Bookmark”选项或“Prev Bookmark”选项，可以找到光标所在位置的后一个或前一个书签所在的行。如果要清除书签标记，只要将光标移到要清除的书签行上，再选择一次“Set/Clear Bookmark”选项，书签即被清除，书签的标记也随之消失。

当选定“Go To Line...”选项时，弹出如图 3-3 所示的对话框，输入行号选择“OK”后，光标自动移到指定的行上，并将该行作为当前行。

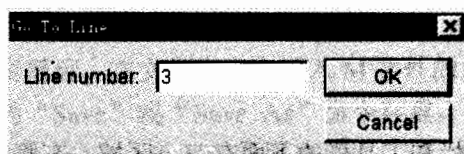


图 3-3 Go To Line...对话框

**提示:**

设置书签的目的是为了快速查找某些程序内容所在的位置, 书签只按程序内容标识, 而不随行号移动, 当在书签上面的位置添加或删除程序行时, 书签标记会随着程序的内容而移动, 这一点与通过行号查找程序是不同的, 也是 MATLAB 6.1 的特色之一。

(2) Debug 菜单: 是用来进行程序调试的, 需要与“Breakpoints”菜单联合使用。“Debug”菜单中共有 6 个菜单项, 在程序运行之前, 仅有“Save and Run”项是激活的。只有当程序中设置了断点, 且程序停止在第一个断点处时其他菜单项才被激活, 这些菜单项为:

- Step 单步运行。每单击一次, 程序运行一次, 但不进入函数。
- Step in 单步运行。遇到函数时进入函数内, 仍单步运行。
- Step out 停止单步运行。如果是在函数中, 跳出函数; 如果不在函数中, 直接运行到下一个断点处。
- Save and Run 存储文件并开始运行。如果文件是已经存储过的, 该选项变为“Run”, 当暂停在断点处时, 该选项变为“Continue”。
- Go Until Cursor 直接运行到光标所在的位置。
- Exit Debug Mode 退出调试方式。

(3) Breakpoints 菜单: 共有 6 个菜单选项, 前两个是用于在程序中设置和清除断点的。后 4 个菜单项是设定停止条件的, 用于临时停止 M 文件的执行, 并给用户一个检查局部变量的机会, 相当于在文件编辑过程中在指定的行号前加入了一个“keyboard”(键盘) 命令。

- Set/Clear Breakpoint 设置或清除断点。
- Clear All Breakpoint 清除所有断点。
- Stop If Error 出现错误时停止程序运行, 不包括 try...catch 语句中的错误。
- Stop If Warning 出现警告时停止程序运行。
- Stop If NaN or Inf 出现非数或无穷大时停止程序运行。
- Stop If All Error 与 Stop If Error 相同, 但包括了 try...catch 语句中的错误。

**提示:**

设置断点是高级语言中程序调试的重要手段之一, 断点是在程序中特定位置设置的中断点, 当程序运行至某一断点处时会暂停运行, 此时可通过检查相关变量的内容等方法确定程序的运行是否正确。在断点处一般可以控制程序按程序行逐行向后继续运行, 也可以控制程序继续运行到指定的程序行, 根据需要, 可以在程序中设置一个或多个断点。try...catch 语句是一种试探性执行语句。



## 3.2 程序流程语句

MATLAB 的程序流程语句主要包括选择结构和循环结构。选择结构是根据给定的条件成立或不成立，分别执行不同的语句，MATLAB 提供的选择结构语句有 if 语句、switch 语句和 try 语句；循环结构是根据给定的条件来决定执行语句的次数，MATLAB 提供的循环结构语句有 while 语句和 for 语句。MATLAB 的程序流程语句都以 end 为结束标志。

### 3.2.1 if 语句

MATLAB 语言中，if 语句有 3 种不同的格式。

#### 1. 单分支 if 语句

最简单的选择结构语句，其基本格式为：

```
if 表达式
    语句组
end
```

说明：如果表达式为真（非零），就执行 if 和 end 之间的语句组，如果表达式为假（零），就执行 end 之后的语句。

例 3-2 输入一个数，小于 10 就输出这个数。

```
n=input('enter a number,n=');    %键盘输入一个数
if n<10
    n
end
```

激活命令窗口，通过键盘输入数字“9”，观看运行结果为：

```
enter a number,n=9
n =

    9
```

再次运行 M 文件后，激活命令窗口，通过键盘输入数字“15”观看运行结果为：

```
enter a number,n=15    %没有输出
```

#### 提示：

在文本编辑器中，输入 M 文件内容后，按〈F5〉或单击“Debug”菜单下的“Save and Run”，在出现的对话框中，输入文件名按〈Enter〉键后，即可建立并运行 M 文件，运行后，文件自动存盘。查看运行结果，要到命令窗口。

#### 2. 双分支 if 语句

前面提供的 if 条件语句只能处理较简单的条件，功能很不全面。为此 MATLAB 还提供

了双分支 if 语句结构。其基本格式为：

```
if 表达式
    语句组 1
else
    语句组 2
end
```

说明：如果表达式为真（非零），则先执行语句组 1，再执行 end 后面的语句；如果条件表达式为假（为零），则先执行语句组 2，再执行 end 后面的语句。

**例 3-3** 给定两个实数，按代数数值的大小输出其中大的。

```
a=input('enter a number,a=');    %键盘输入数据
b=input('enter a number,b=');
if a>b
    max=a;
else
    max=b;
end
max                                %输出最大的实数
```

激活命令窗口，通过键盘输入数字“9”和数字“6”，观看运行结果为：

```
enter a number,a=9
enter a number,b=6
max =
9
```

**注意：**

表达式可以是任何合法的表达式，多为逻辑表达式或关系表达式，只要表达式的值不等于零（可以是负数），就认为表达式为真。

### 3. 多分支 if 语句

当有 3 个或更多的选择时，可采用 if 语句的嵌套，也可以采用多分支 if 语句。其基本格式为：

```
if 表达式 1
    语句组 1
elseif 表达式 2
    语句组 2
...
elseif 表达式 n
    语句组 n
else
    语句组 n+1
end
```

说明：先判断表达式 1 的值，若为真，则执行语句组 1，执行完语句组 1 后，跳出该选择结构，继续执行 end 后的命令；当表达式 1 为假时，跳过语句组 1，进而判断表达式 2，若为真，则执行语句组 2，并跳出选择结构。如此下去，若所有表达式都为假，则执行 else 后的语句组 n+1。当然最后的 else 命令有时可能没有。

例 3-4 给定三个数 A, B, C，要求按由大到小的顺序输出，其中最大数放入 A，最小数放入 C 中。

```
A=15;B=24;C=-45;
if A<B
    T=A;A=B;B=T;    %实现 A 和 B 的互换
elseif A<C
    T=A;A=C;C=T;    %实现 A 和 C 的互换
elseif B<C
    T=B;B=C;C=T;    %实现 B 和 C 的互换
end
A
B
C
```

激活命令窗口，观看运行结果为：

```
A =
    24
B =
    15
C =
   -45
```

### 3.2.2 switch 语句

switch 语句是多分支选择语句。if 语句只有两个分支可供选择，如果分支较多，则嵌套的 if 语句层数多，程序冗长而且可读性降低。MATLAB 从 5.0 版开始提供了 switch 语句，其基本格式为：

```
switch 表达式
    case 表达式 1
        语句组 1
    case 表达式 2
        语句组 2
    ...
    case 表达式 n
        语句组 n
    otherwise
        语句组 n+1
end
```

说明：当表达式的值等于某个 case 语句后面的表达式时，程序将转移到该组语句去执

行，执行完后，程序直接跳出开关体，执行 end 后的语句。程序的执行结果与各个 case 语句的次序无关，但当表达式的值不等于任何一个 case 语句后面的表达式时，程序将执行 otherwise 语句后的语句组，然后，执行 end 后的语句。

例 3-5 某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用 price 来表示）：

$\text{price} < 200$	没有打折
$200 \leq \text{price} < 500$	3%折扣
$500 \leq \text{price} < 1000$	5%折扣
$1000 \leq \text{price} < 2500$	8%折扣
$2500 \leq \text{price} < 5000$	10%折扣
$5000 \leq \text{price}$	14%折扣

输入某件商品的价格，求所售商品的实际销售价格。

程序如下：

```
price=input('请输入商品价格 price=')
switch fix(price/100) %fix 为向零方向取整函数
case{0,1} %价格在 200 以内
    rate=0; %没有打折
case{2,3,4} %价格在 200 到 500 之间
    rate=3/100; %折扣率 3%
case num2cell(5:9) %num2cell 是数值数组转化为单元数组函数，这里
    %相当于 case {5,6,7,8,9}
    rate=5/100;
case num2cell(10:24)
    rate =8/100;
case num2cell(25:49)
    rate =10/100;
otherwise
    rate=14/100;
end
price=price*(1-rate) %输出之实际销售价格
```

激活命令窗口，通过键盘输入数字“2000”，观看运行结果为：

```
请输入商品价格 price=2000
price =
    2000
price =
    1840
```

注意：

MATLAB 无需像 C 语言那样在每一个 case 语句后加上 break 语句，而是直接跳出整个语句结构。另外，当要在表达式满足某个表达式之一时执行某一程序段，则应把这样的表达式组用大括号括起来，中间加上逗号分隔。

### 3.2.3 while 语句

while 语句是条件循环语句，while 循环使语句体在表达式（多为逻辑条件或关系条件）控制下重复不确定次，直到表达式的值等于零为止。while 循环的一般形式是：

```
while 表达式
    语句体
end
```

说明：只要表达式的结果非零，语句体就重复执行。当表达式的结果不是标量时，可以用 any、all 等函数处理，如果表达式的值总是等于 1，该循环将无休止进行（即死循环）。

例 3-6 求  $1+2+3+\dots+100$  的和。

```
i=0;
s=0;
while i<=100
    s=s+i;
    i=i+1;
end
s
```

激活命令窗口，观看运行结果为：

```
s =
    5050
```

### 3.2.4 for 语句

for 语句为记数循环语句，在许多情况下，循环条件是有规律变化的，通常把循环条件初值、终值和变化步长放在循环的开头，这种形式就是 for 语句的循环结构。概括地讲，for 循环的一般形式是：

```
for 循环变量=表达式 1: 表达式 2: 表达式 3
    语句体
end
```

说明：其中表达式 1 的值是循环变量的初值，表达式 2 的值是循环步长，表达式 3 的值是循环变量的终值。初值、步长和终值可以取整数、小数、正数和负数，步长可以默认，默认值为 1。

例 3-7 利用 for 语句，求解例 3-6。

```
sum=0;
for i=1:100
    s=s+i;
end
s
```

激活命令窗口，观看运行结果为：

```
s =
    5050
```

### 3.2.5 循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。任一种循环语句的循环体部分都可以包含另一个循环语句，多重循环嵌套的层数可以是任意的。习惯上按照嵌套层数，分别叫做二重循环、三重循环等。处于内部的循环叫做内循环，处于外部的循环叫做外循环。在设计多重循环时，要特别注意内、外循环之间的关系，以及语句放置次序，不要搞错。

**例 3-8** 有一数列： $1^1+1^2+1^3+\dots+1^{10}+2^1+2^2+2^3+\dots+2^{10}+3^1+3^2+3^3+\dots+3^{10}$ ，求这些项的和。

```
s=0;
for i=1:3          %外层循环，分别产生 1、2、3
    for j=1:10      %内层循环，分别产生 1~9
        s=s+i^j;    %求和
    end
end
s                  %输出结果
```

激活命令窗口，观看运行结果为：

```
s =
    90628
```

**注意：**

for 语句可以嵌套使用，但在嵌套过程中每一个 for 都必须与其下方最近的一个 end 相匹配，否则程序将出错。

### 3.2.6 其他流程控制语句

#### 1. continue 语句

continue 语句用于控制 for 循环和 while 循环跳过某些执行语句，在 for 循环和 while 循环中，当出现 continue 语句时，则跳过循环体中所有剩余的语句，继续下一次循环。在嵌套循环中，continue 控制执行本嵌套中的下一次循环。

**例 3-9** 把 100 到 120 之间的能被 7 整除的整数输出。

```
for i=100:120
    if rem(i,7)~=0    %rem 为求余数函数，判断能否被 7 整除
        continue    %把不能被 7 整除的数去掉，判断下个数字
    end
    i                %输出能被 7 整除的数
end
```

激活命令窗口，观看运行结果为：

```
i =  
105  
i =  
112  
i =  
119
```

## 2. break 语句

**break** 语句用于终止 **for** 循环和 **while** 循环的执行。当遇到 **break** 语句时，则退出循环体，继续执行循环体外的下一个语句。在嵌套循环中，**break** 往往存在于内层的循环中。

**例 3-10** 将上例稍作改动，则输出 100 到 150 之间第一个能被 7 整除的整数。

```
for i=100:150  
    if rem(i,7)~=0  
        continue  
    end  
    break          %得到第一个数后，就中止循环  
end  
i
```

激活命令窗口，观看运行结果为：

```
i =  
105
```

**注意：**

**break** 语句不能用于 **for** 语句和 **while** 语句之外的任何语句。

## 3. return 语句

**return** 语句用于终止当前的程序序列，并返回到调用的函数或键盘操作中，也用于终止 **keyboard** 方式，在 **MATLAB** 中，被调用函数运行结束后自动返回调用函数。但使用 **return** 语句时，将 **return** 插入到被调用函数中的某一位置，可根据某种条件迫使被调用函数提前结束并返回主调函数。

## 4. try 语句

**MATLAB** 从 5.2 版开始提供了 **try** 语句，这是一种试探性执行语句。语句格式为：

```
try  
    语句组 1  
catch  
    语句组 2  
end
```

说明：**try** 语句先试探性执行语句组 1，如果语句组 1 在执行过程中出现错误，则将错误信息赋给保留的 **lasterr** 变量，并转去执行语句组 2。这种试探性执行语句是其他高级语言所没有的。

**例 3-11** 矩阵乘法运算要求两矩阵的维数相容，否则会出错。先求两矩阵的乘积，若出错，则自动转去求两矩阵的点乘。

程序如下：

```
>> A=[1,2,3;4,5,6]; B=[7,8,9; 10,11,12];
>> try
    C=A*B;
catch
    C=A.*B;
end
C
lasterr                                %显示出错原因
```

激活命令窗口，观看运行结果为：

```
C =
     7     16     27
    40     55     72
ans =
Error using ==> *
Inner matrix dimensions must agree.    %显示出错原因
```

### 3.3 函数文件

函数文件是另一种形式的 M 文件，每一个函数文件都是定义的一个函数。事实上，MATLAB 提供的标准函数大部分都是由函数文件定义的，这足以说明函数文件的重要性。从使用的角度看，函数是一个“黑箱”，把一些数据送进去，经加工处理，把结果送出来。从形式上看，函数文件区别于命令文件之处，在于命令文件的变量在文件执行完成后保留在工作空间中，而函数文件内定义的变量，只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。因此，在应用函数文件时，只关心函数的输入和输出即可。

#### 3.3.1 函数文件的基本结构

函数文件由 function 语句引导，其基本结构为：

```
function [输出形参表]=函数名(输入形参表)
    注释说明部分
    函数体语句
return
```

其中以 function 开头的一行为引导行，表示该 M 文件是一个函数文件。函数名的命名规则与变量名相同。输入形参为函数的输入参数，输出形参为函数的输出参数。当输出形参只有一个时，直接输入变量名而不用方括号。输入结束后，不能像 M 文件那样按〈F5〉或在“Debug”菜单中选择“Save and Run”命令项运行，而是要直接存盘。说明如下。

- 函数文件名。通常由函数名再加上扩展名.m 组成，不过函数文件名与函数名也可以



不相同。当两者不同时，MATLAB 将忽略函数名而确认函数文件名，因此，调用时使用函数文件名。不过最好把文件名和函数名统一，以免出错。

● 注释说明包括三部分内容：

① 紧随函数文件引导行之后以 % 开头的第一注释行。这一行一般包括大写的函数文件名和函数功能的简要描述，供 “lookfor” 关键词查询和 “help” 在线帮助用。

② 第一注释行之后连续的注释行。通常包括函数输入输出参数的含义及调用格式说明等信息，构成全部在线帮助文本。

③ 与在线帮助文本相隔一个空行的注释行。包括函数文件编写和修改的信息，如作者、修改日期、版本等内容，用于软件档案管理。

● 如果在函数文件中插入了 return 语句，则执行到该语句就结束函数的执行，程序流程转到调用该函数的位置。通常在函数文件中也可不使用 return 语句，这时在被调函数执行完成后自动返回。

### 3.3.2 函数调用

函数文件编制好后，就可以调用函数进行计算了。函数调用的一般格式为：

[输出实参表]=函数名(输入实参表)

要注意的是，函数调用时各实参出现的顺序、个数应与函数定义时形参的顺序、个数一致，否则会出错。函数调用时，先将实参传递给相应的形参，从而实现参数传递，然后再执行函数的功能。

**例 3-12** 编写一个函数文件，对输入的两个数进行加、减运算。

```
建立函数文件 datfunction.m
function [add,mul]=datfunction(a,b)
    add=a+b;
    mul=a-b;
```

然后在 MATLAB 的命令窗口调用该函数文件：

```
>> [m,n]=datfunction(7,5)
m =
    12
n =
     2
```

在 MATLAB 中，函数可以嵌套调用，即一个函数可以调用别的函数，甚至调用它自身。在调用一个函数的过程中又出现直接或间接地调用该函数本身，称为函数的递归调用。

**例 3-13** 有 5 个人坐在一起，问第 5 个人多少岁？他说比第 4 个人大 2 岁。问第 4 个人岁数，他说比第 3 个人大 2 岁。问第 3 个人岁数，他说比第 2 个人大 2 岁。问第 2 个人岁数，他说比第 1 个人大 2 岁。最后问第 1 个人， he 说是 12 岁。问第 5 个人多大？

```
function c=age(n)
if n==1
    c=12;
```

```

else
    c=age(n-1)+2;
end

```

然后在 MATLAB 的命令窗口调用该函数文件：

```

>> hisage=age(5)

hisage =
    20

```

**提示：**

当函数调用完毕后，可以用命令“who”或“whos”查看工作空间中的变量，也可以双击工作空间窗口（Workspace）的变量图标，在打开的矩阵编辑窗口（Array Editor）中查看。

### 3.3.3 函数所传递参数的可调性

MATLAB 在函数调用上有一个与一般高级语言不同之处，就是函数所传递参数数目的可调性。凭借这一点，一个函数可完成多种功能。在调用函数时，MATLAB 用两个永久变量 nargin 和 nargout 分别记录调用该函数时的输入实参和输出实参的个数。只要在函数文件中包含这两个变量，就可以准确地知道该函数文件被调用时的输入输出参数个数，从而决定函数如何处理。

**例 3-14** 编写一个函数文件 test，比较输入一个参数、两个参数和三个参数时的结果。

```

function apple=test(a,b,c)
if nargin==1           %输入一个参数
    apple=a;           %返回参数 a
elseif nargin==2       %输入两个参数
    apple=a+b;         %返回参数 a+b
elseif nargin==3
    apple=a+b+c;
end

```

在命令窗口输入以下内容：

```

>> x=[2,4,6];          %参数 x
>> test(x)              %调用函数，仅输入一个参数，显示结果为 x

ans =
     2     4     6

>> y=[3,6,9];          %另一个参数
>> test(x,y)            %调用函数，输入两个参数，显示结果为 x+y

ans =

```

```

5    10    15
>> test(x,y,1)           %调用函数，输入三个参数，显示结果为 x+y+1

ans =
6    11    16

```

**注意：**

函数有它们自己的专用工作空间，它与 MATLAB 的工作空间分开。函数内变量与 MATLAB 工作空间之间惟一的联系是函数的输入和输出参数。如果函数任一输入参数值发生变化，其变化仅在函数内出现，不影响 MATLAB 工作空间的其他变量。函数内所创建的变量只驻留在函数的工作空间，而且只在函数执行期间临时存在，函数调用结束后就释放了。

### 3.3.4 全局变量

由于函数空间和 MATLAB 的工作空间是独立分开的，如果函数文件的内部变量都是局部的，从一个调用到下一个调用，在函数工作空间中定义的变量不能直接被另一个函数文件调用。为了解决这个问题，MATLAB 使用全局变量。

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一个变量。全局变量的作用域是整个 MATLAB 工作空间，即全程有效。所有的函数都可以对它进行存取和修改。因此，定义全局变量是函数间传递信息的一种手段。

定义全局变量的方法是使用 global 命令，格式为：

global 变量名

值得指出，在程序设计中，全局变量固然可以带来某些方便，但却破坏了函数对变量的封装，降低了程序的可读性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当程序较大，子程序较多时，全局变量将给程序调试和维护带来不便，故不提倡使用全局变量。如果一定要用全局变量，最好给它起一个能反映变量含义的名字，以免和其他变量混淆。

在实际编程时，可在所有需要调用全局变量的函数里定义全局变量，这样就可实现数据共享。在函数文件里，全局变量的定义语句应放在变量使用以前，为了便于了解所有的全局变量，一般把全局变量的定义语句放在文件的前部。

**注意：**

实际编程中，全局变量名多用大写字母，并有选择地以首次出现的 M 文件的名字开头，这样，可把多个全局变量之间不必要的互作用减至最小。

## 3.4 MATLAB 语言编程技巧

### 3.4.1 测定程序执行时间

测定程序执行时间通常是使用 tic 和 toc 函数，tic 用于启动秒表，toc 用于停止秒表，tic

和 `toc` 函数返回的是变量 `elapsed_time`。

**例 3-15** 建立一个  $100 \times 100$  的魔方矩阵，并测定运行时间。

```
>> tic           %开始计时
>> A=magic(100); %运行程序，magic 为魔方矩阵函数
>> toc           %结束计时，并显示所耗时间
```

在命令窗口观察运行结果为：

```
elapsed_time =
           0.0600
```

### 3.4.2 加快 MATLAB 程序执行的方法

毕竟 MATLAB 是一种解释性语言，同其他解释性语言一样，都存在着执行速度不够理想的问题。下面是一些加快 MATLAB 程序执行的方法。

- 避免使用循环。

MATLAB 的一个不足是当对矩阵的单个元素进行循环操作时运算速度很慢，因此应尽量避免使用循环。编程时，尽量对矩阵或向量编程，把循环向量化，这样不仅能够缩短程序长度，而且能提高程序执行效率。在必须使用多重循环的情况下，若两个循环执行的次数不同，则应在循环的内层执行次数多的，外层执行次数少的。

- 对大型矩阵预先定义维数。

在程序执行的过程中，有时要动态改变矩阵的维数，这将非常浪费时间。为此，应在定义大矩阵时，首先用 MATLAB 的内在函数（如 `zeros` 或 `ones`）对矩阵定义好维数，然后再进行赋值，这样会提高程序的运行效率。

- 优先使用内层函数。

矩阵运算要首先考虑使用 MATLAB 内层函数，因为内层函数是由更底层的 C 语言构成，执行速度快于使用循环的矩阵运算。

- 考虑接口语言。

MATLAB 支持同其他语言进行编译连接，如果已经采取了相应的措施，程序执行速度仍然很慢，则应考虑使用 C 语言或 FORTRAN 语言进行编程。然后编译连接，这样能显著地提高程序的运行速度。

## 3.5 上机实践

### 3.5.1 跟我学

**例 3-16** 计算分段函数  $y = \begin{cases} x^2 & x < 1 \\ x^2 - 1 & 1 \leq x < 2 \\ x^2 - 2x + 1 & x \geq 2 \end{cases}$  的值。

程序如下：

```

x=input('输入 x 的值 x=')
if x<1
    y=x^2;
elseif x>=1&x<2
    y=x^2-1;
else
    y=x^2-2*x+1;
end
y

```

激活命令窗口，通过键盘输入数字“3”，观看运行结果为：

```

输入 x 的值 x=3
x =
    3
y =
    4

```

**例 3-17** 给出一个学生的百分制成绩，要求转化成成绩等级输出，等级分为‘A’、‘B’、‘C’、‘D’、‘E’五等。90 分以上为‘A’，80~89 分为‘B’，70~79 分为‘C’，60~69 分为‘D’，60 分以下为‘E’。

程序如下：

```

score=input('请输入学生成绩 score=:')
switch fix(score/10)    %fix 为向零取整函数
case{10,9}
    grade='A';
case 8
    grade='B';
case 7
    grade='C';
case 6
    grade='D';
otherwise
    grade='E';
end
grade

```

激活命令窗口，通过键盘输入数字“3”，观看运行结果为：

请输入学生成绩 score=:78

```

score =
    78
grade =
    C

```

**例 3-18** 求  $\sum_{n=1}^{20} n!$ 。

程序如下:

```
sum=0;
temp=1;
for n=1:20
    temp=temp*n;
    sum=sum+temp;
end
sum
```

激活命令窗口, 观看运行结果为:

```
>>
sum =                %程序运行结果

2.5613e+018
```

**例 3-19** 输出 100~1000 之间的所有“水仙花数”。“水仙花数”是一个三位数, 其各位数字的立方和等于该数本身。如  $153=1^3+5^3+3^3$ 。

程序如下:

```
for n=100:999
    a=floor(n/100);
    b=floor(n/10-a*10);
    c=floor(rem(n,10));
    if n==(a^3+b^3+c^3)
        n
    end
end
```

本题的“水仙花数”是 153、370、371、407。

**例 3-20** 求一个  $3 \times 3$  矩阵  $a = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$  对角线元素之和。

程序如下:

```
sum=0;
a=[1 2 3;4 5 6;7 8 9];
for i=1:3
    sum=sum+a(i,i);
end
end
sum
```

激活命令窗口, 观看运行结果为:

```
sum =
15
```

**例 3-21** 求  $S_n = a + aa + aaa + \cdots + aa \cdots a$  之值, 其中  $a$  是一个数字, 由键盘输入; 表达式中位数最多项  $a$  的个数, 也由键盘输入。

程序如下：

```
i=1;sn=0;tn=0;
a=input('请输入 a 的数值:')
n=input('请输入位数最多项中 a 的个数:')
while i<=n
    tn=tn+a;
    sn=sn+tn;
    a=a*10;
    i=i+1;
end
sn
```

激活命令窗口，通过键盘输入数字“2”和数字“5”，观看运行结果为：

请输入  $a$  的数值:2

```
a =
    2
```

请输入位数最多项中  $a$  的个数:5

```
n =
    5
sn =
   24690
```

**例 3-22** 编写一个计算任意正整数阶乘的函数文件，并在命令窗口调用。

函数文件如下：

```
function mul=fact(n)
mul=1;
for i=1:n
    mul=mul*i;
end
```

在命令窗口调用：

```
>> fact(5)
ans =
   120
```

**例 3-23** 建立一个函数文件 sub，比较下面两个文件的运行结果，说明全局变量的作用。

函数文件：function fun=sub(z)

```
global x
z=3*x;
x=x+z;
```

文件 1 如下：

```

global x
x=1:2:5;
y=2:2:6
sub(y)
x
y

```

激活命令窗口，观看运行结果为：

```

>>                                %文件 1 的运行结果
x =
    4    12    20
y =
    2     4     6

```

文件 2 如下：

```

clear                                %清除工作空间变量
clc                                  %清屏幕
x=1:2:5;
y=2:2:6;
sub(y)
x
y

```

激活命令窗口，观看运行结果为：

```

>>                                %文件 2 的运行结果
x =
    1     3     5
y =
    2     4     6

```

**注意：**

运行完文件 1，再运行文件 2 时，一定要在文件 2 的开头加上 `clear` 函数，否则结果相同，因为全局变量 `x` 已经在工作空间内。

### 3.5.2 自己练

1. 计算分段函数的值。

$$y = \begin{cases} x-1 & x \geq 0 \\ x^2+1 & x < 0 \end{cases}$$

2. 求下面表达式的值。



$$\sum_{k=1}^{100} k + \sum_{k=1}^{50} k^2 + \sum_{k=1}^{10} \frac{1}{k}$$

3. 求下面这个分数序列前 20 项之和。

$$\frac{2}{1}, \frac{3}{2}, \frac{5}{3}, \frac{8}{5} \dots$$

4. 输入四个整数，要求按由大到小的顺序排序。

5. 建立一个 4×4 阶的矩阵，编程输出最大元素的行号、列号和元素值。

6. 对一个含有 10 个元素的行向量按由小到大的顺序排序。

7. 有一群鸡和兔子，加在一起头的数量是 36，脚的数量是 100，编程序解答鸡和兔子的数量各是多少？

### 3.6 本章小结

MATLAB 语言是解释性程序设计语言，程序中的语句边解释边执行。其程序设计具备自由、灵活、简洁的风格，而且拥有丰富的库函数，因此避免了复杂的函数编程调试工作。本章主要讲解了 M 文件、分支控制语句（if 和 switch）、循环控制语句（for 和 while）、M 函数文件以及编程技巧等内容。本章涉及到的函数或命令列表 3-1 如下。

表 3-1 本章主要函数或命令一览

函数或命令名称	说 明
if	条件运行语句
else	条件运行语句（配合 if 使用）
elseif	条件运行语句（配合 if 使用）
switch	多选开关语句
case	多选开关语句
otherwise	开关语句的默认部分
end	终止条件语句，或指示最大下标
while	重复执行若干次，直到表达式等于零为止
continue	运行下一次循环
return	返回调用函数
for	重复执行指定次数
break	终止循环语句
global	定义全局变量
try...catch	检测语句
tic	秒表计时开始
toc	秒表计时结束

### 3.7 习题

1. 如何建立、运行、保存 M 文件？
2. M 文件和 M 函数文件有哪些区别？如何建立 M 函数文件？
3. 如何调试 MATLAB 程序？
4. 如何优化 MATLAB 程序？
5. 输入 20 个数，输出其中最大数和最小数。
6. 已知  $S=1+2+2^2+2^3+\cdots+2^{63}$ ，求  $S$  的值。
7. 分别用 for 和 while 循环结构编写程序并计算。

$$\sum_{n=1}^{100} (2n-1)^2$$

8. 计算如下的分段函数。

$$f(x) = \begin{cases} 0 & x \leq 5 \\ \frac{x-3}{4} & 5 < x \leq 10 \\ 2x & x > 10 \end{cases}$$

9. 从键盘输入一个四位整数，按规则加密后输出。加密规则：每位数字都加上 7，然后用和除以 10 的余数取代该数。
10. 产生 20 个两位随机整数，输出其中小于平均值的奇数。
11. 写出下面程序的运行结果。

```
s=0;
a=[1 2 3;4 5 6;7 8 9;10 11 12];
for k=a
    for j=1:4
        if rem(k(j),2)~=0
            s=s+k(j);
        end
    end
end
s
```

## 第4章 MATLAB 绘图

在本章中，读者可以掌握二维平面图形和三维立体图形的绘制方法，能够使用这些方法对实验数据进行图表化处理；能够利用不同的颜色、线条和模块绘制图形。本章重点是 4.1 二维绘图，难点是 4.5.1 三维数据的产生。

### 4.1 二维绘图

MATLAB 最常用的二维绘图函数是 `plot` 函数。该函数将各个数据点用直线连接绘制图形。MATLAB 的其他二维绘图函数中的绝大多数都是以 `plot` 为基础构造的。`plot` 函数可打开一个默认的图形窗口，它还自动将数值标尺及单位标注加到两个坐标轴上。如果已经存在一个图形窗口，`plot` 函数将刷新当前窗口的图形。

`Plot` 函数有以下几种常用形式。

#### 1. `plot(x, y)`

说明： $x$ 、 $y$  可以是向量或矩阵。

- 当  $x$ 、 $y$  均为向量时，要求向量  $x$  与向量  $y$  的长度一致，则 `plot(x, y)` 绘制出以  $x$  为横坐标， $y$  为纵坐标的二维图形。
- 当  $x$  为向量、 $y$  为矩阵时，`plot(x, y)` 用不同颜色的曲线绘制出  $y$  行或列对于  $x$  的图形。 $y$  矩阵的行或列的选择取决于  $x$ 、 $y$  的维数，若  $y$  为方阵或  $y$  矩阵的列向量长度与  $x$  向量的长度一致，则绘制出  $y$  矩阵的各个列向量相对于  $x$  的一组二维图形；若  $y$  矩阵的行向量长度与  $x$  向量的长度一致，则绘制出  $y$  矩阵的各个行向量相对于  $x$  的一组二维图形。
- 若  $x$  为矩阵， $y$  为向量，则按类似于上条的规则处理。
- 若  $x$ 、 $y$  是同维的矩阵，则 `plot(x, y)` 绘制出  $y$  列向量相对于  $x$  的列向量之间的一组二维图形。

例 4-1  $x$ 、 $y$  是同样长度的向量，绘制  $y$  元素对应于  $x$  元素的曲线图（如图 4-1 所示）。

```
>> x=0:0.05:4*pi;    %给出 x 向量，步长 0.05
>> y=sin(x);         %y 为 x 的正弦曲线函数
>> plot(x,y)
```

例 4-2  $x$  为向量， $y$  是列长与  $x$  相同的矩阵，绘制  $y$  对应于  $x$  的曲线图（如图 4-2 所示）。

```
x=0:pi/50:2*pi;
y(1,:)=sin(x);
y(2,:)=0.3*sin(x);
y(3,:)=0.6*sin(x);
plot(x,y)
```

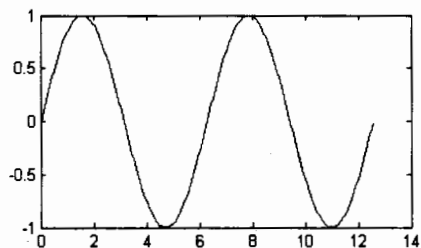


图 4-1 双向量曲线图

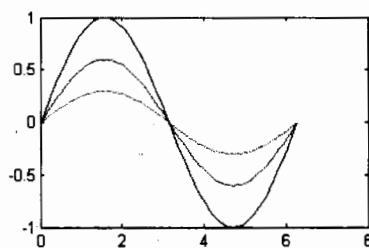


图 4-2  $x$  为向量  $y$  为矩阵的曲线图

## 2. plot(x)

说明:  $x$  可以是向量或矩阵。

- 若  $x$  为向量, 则绘制出一个  $x$  元素和  $x$  元素排列序号之间关系的线性坐标图。
- 若  $x$  为矩阵, 则绘制出  $x$  的列向量相对于行号的一组二维图形。

例 4-3 单向量绘图 (如图 4-3 所示)。

```
>> x=[0 0.2 0.5 0.7 0.6 0.7 1.2 1.5 1.6 1.9 2.3];
>> plot(x)
```

例 4-4 二维矩阵绘图 (如图 4-4 所示)。

```
>> x=[1 2 3 4 5 6;7 8 9 10 11 12;13 14 15 16 17 18];
>> plot(x)
```

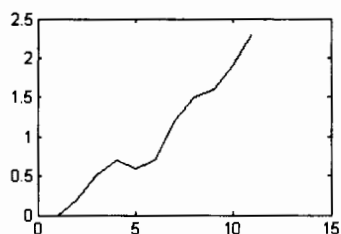


图 4-3 单向量曲线图

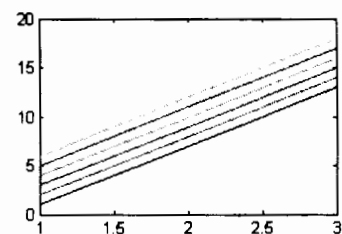


图 4-4 二维矩阵的曲线图

## 3. plot(x, y, '参数')

说明:  $x, y$  可以是向量或矩阵, 与前相同。参数选项为一个字符串, 它决定了二维图形的颜色、线型及数据点的图标。表 4-1、表 4-2、表 4-3 分别给出颜色、线型和标记的控制字符。

表 4-1 颜色控制符

字 符	颜 色	字 符	颜 色
b	蓝色	m	紫红色
c	青色	r	红色
g	绿色	w	白色
k	黑色	y	黄色

表 4-2 线型控制符

符 号	线 型	符 号	线 型
—	实线 (默认)	:	点连线
-.	点画线	---	虚线

表 4-3 数据点标记字符

控 制 符	标 记 符	控 制 符	标 记 符
.	点	h	六角形
+	十字号	p	五角星
o(字母)	圆圈	v(字母)	下三角
*	星号	^	上三角
x(字母)	叉号	>	右三角
s	正方形	<	左三角
d	菱形		

**注意:**

线型、颜色和标记点三种属性的符号必须放在同一个字符串内, 属性的先后顺序没有关系, 可以只指定一个或两个, 但同种属性不能同时指定两个。

例 4-5 用红颜色、点连线、叉号画出正弦曲线 (如图 4-5 所示)。

```
>> x=0:0.2:8;
>> y=sin(x);
>> plot(x,y,'r:x')
```

4. plot(x1, y1, '参数 1', x2, y2, '参数 2' ...)

说明: 可以用同一函数在同一坐标系中画多幅图形, x1, y1 确定第一条曲线的坐标值, 参数 1 为第一条曲线的选项参数; x2, y2 为第二条曲线的坐标值, 参数 2 为第二条曲线的选项参数……

例 4-6 用不同的线型在同一坐标内绘图 (如图 4-6 所示)。

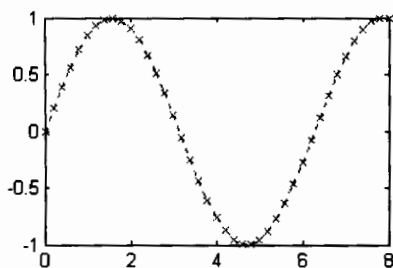


图 4-5 正弦曲线

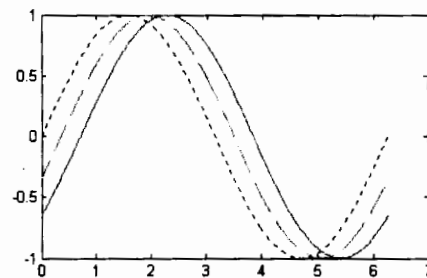


图 4-6 同一坐标内的三条曲线

```
t=0:pi/100:2*pi;
```

```

y1=sin(t);
y2=sin(t-0.35);
y3=sin(t-0.7);
plot(t,y1,'.',t,y2,'--',t,y3,'-')

```

## 4.2 图形修饰与控制

MATLAB 提供了一系列图形修饰函数，用于对 plot 函数绘制的图形进行修饰和控制。

### 4.2.1 坐标轴的调整

MATLAB 用 axis 函数对绘制图形的坐标轴进行调整。axis 函数的功能非常丰富，可用它控制坐标轴的比例和特性。

#### 1. 坐标轴比例控制

函数：axis ( $[x_{\min} \ x_{\max} \ y_{\min} \ y_{\max}]$ ) 将图形的  $x$  轴范围限定在  $[x_{\min}, x_{\max}]$  之间， $y$  轴的范围限定在  $[y_{\min}, y_{\max}]$  之间。MATLAB 绘制图形时，按照给定的数据值确定坐标轴参数范围。对坐标轴范围参数的修改，也就相当于对原图形进行放大或缩小处理。

#### 2. 坐标轴特性控制

函数：axis ('控制字符串') 根据表 4-4 所示的功能控制图形。

表 4-4 axis 控制符

控制字符串	函数功能
auto	自动设置坐标系（默认）： $x_{\min}=\min(x)$ 、 $x_{\max}=\max(x)$ 、 $y_{\min}=\min(y)$ 、 $y_{\max}=\max(y)$
square	将图形设置为正方形图形
equal	将图形的 $x$ 、 $y$ 坐标轴的单位刻度设置为相等
normal	关闭 axis(square)和 axis(equal)函数的作用
xy	使用笛卡尔坐标系
ij	使用“matrix”坐标系。即坐标原点在左上方， $x$ 坐标从左向右增大， $y$ 坐标从上向下增大
on	打开所有轴标注、标记和背景
off	关闭所有轴标注、标记和背景

#### 3. 坐标刻度标示

- 函数：set(gca, 'xtick', 标示向量)

set(gca, 'ytick', 标示向量)

说明：按照标示向量设置  $x$ 、 $y$  轴的刻度标示。

- 函数：set(gca, 'xticklabel', '字符串|字符串...')

set(gca, 'yticklabel', '字符串|字符串...')

说明：按照字符串设置  $x$ 、 $y$  轴的刻度标注。

例 4-7 分别改变  $x$  轴和  $y$  轴的标注点绘制函数曲线（如图 4-7 所示）。

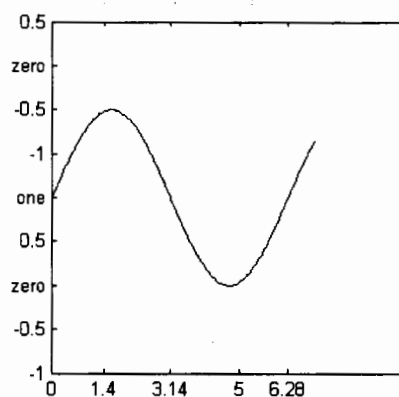


图 4-7 改变标注点的正弦曲线

```
x=0:0.05:7;
y=sin(x);
plot(x,y)
axis([0 3*pi -2 2])           %坐标轴比例控制
axis('square')                %坐标轴特性控制
set(gca,'yticklabel',{'-1|-0.5|zero|0.5|one'}) %改变 y 轴的标注点
set(gca,'xtick',[0 1.4 3.14 5 6.28])          %改变 x 轴的标注点
```

## 4.2.2 文字标示

有关图形的标题、轴线标注等函数有：

`title`（‘字符串’）——图形标题。

`xlabel`（‘字符串’）—— $x$  轴标注。

`ylabel`（‘字符串’）—— $y$  轴标注。

`text`（ $x$ ,  $y$ , ‘字符串’）——在坐标（ $x$ ,  $y$ ）处标注说明文字。

`gtext`（‘字符串’）——用鼠标在特定处标注说明文字。

若要输入特定的文字需要用反斜杠（\）开头，用法如表 4-5 所示。

表 4-5 特殊字符

输入字符	表示的特殊字符
<code>\pi</code>	$\pi$
<code>\alpha</code>	$\alpha$
<code>\beta</code>	$\beta$
<code>\leftarrow</code>	←（左方向箭头）
<code>\rightarrow</code>	→（右方向箭头）
<code>\bullet</code>	●

**例 4-8** 编辑一个 M 文件，画出正弦函数图形，图形上包括坐标轴标题、图形标题，并在曲线的过零点处做文字标示（如图 4-8 所示）。

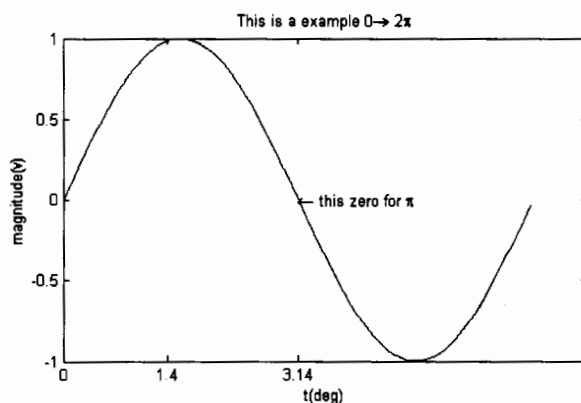


图 4-8 带有文字标示的正弦曲线

```
t=0:0.05:2*pi;
plot(t,sin(t))
set(gca,'xtick',[0 1.4 3.14 56.28])
xlabel('t(deg)')
ylabel('magnitude(v)')
title('This is a example 0\rightarrow 2\pi')
text(3.14,sin(3.14),'\leftarrow this zero for \pi')
```

### 4.2.3 图例注解

当在一个坐标系上画出多幅图形时，为区分各个图形，MATLAB 提供出了图例的注解说明函数。其格式如下：

`legend(字符串 1, 字符串 2, ..., 参数)`

此函数在图中开启一个注解视窗，依据绘图的先后顺序，依次输出字符串对各个图形进行注解说明。如字符串 1 表示第一个出现的线条，字符串 2 表示第二个出现的线条，参数字符串确定注解视窗在图形中的位置，其含义见表 4-6。同时，注解视窗可以用鼠标拖动，以便将其放置在一个合适的位置。

表 4-6 参数字符串的含义

参数字符串	含 义
0	尽量不与数据冲突，自动放置在最佳位置
1	放置在图形的右上角（默认）
2	放置在图形的左上角
3	放置在图形的左下角
4	放置在图形的右下角
-1	放置在图形视窗外右边

**例 4-9** 在同一坐标内，绘出两条函数曲线并有图例注解（如图 4-9 所示）。

```
x=0:0.2:12;
```



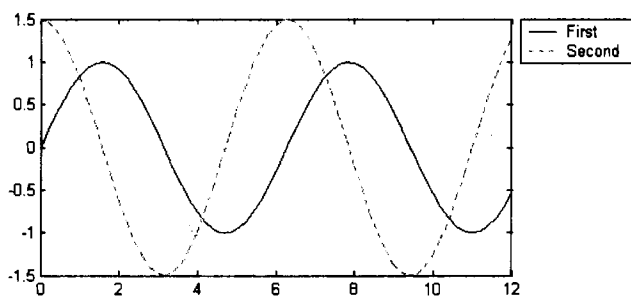


图 4-9 带有图例的函数曲线

```
plot(x,sin(x),'-',x,1.5*cos(x),'-');
legend('First','Second',-1); % 强行将注解视窗放置在图形视窗外的右方结果
```

#### 4.2.4 图形的保持

hold 函数用于保持当前图形。用 plot 函数绘图时，首先将当前图形窗口清屏，再绘制图形，所以只能见到最后一个 plot 函数绘制的图形。为了能利用多条 plot 函数绘制多幅图形，就需要保持窗口上的图形。

hold on 表示保持当前图形及轴系的所有特性。hold off 表示解除 hold on 函数。

例 4-10 在同一个窗口，使用两次 plot 函数绘制出两条曲线（如图 4-10 所示）。

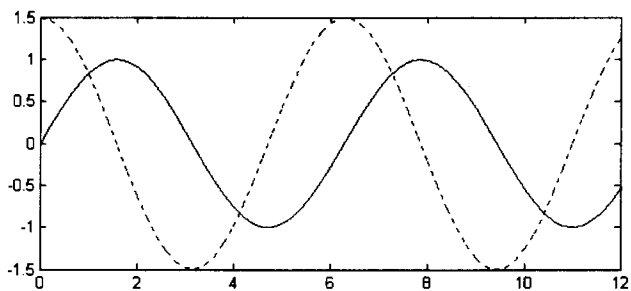


图 4-10 两次绘制的函数曲线

```
x=0:0.2:12;
plot(x,sin(x),'-')
hold on
plot(x,1.5*cos(x),'-');
```

#### 4.2.5 网格控制

网格是在坐标刻度标示上画出的格线。画出网格，便于对曲线进行观察和分析。设置或取消网格需要使用网格控制函数。函数如下：

grid on —— 在所画的图形中添加网格线。  
grid off —— 在所画的图形中去掉网格线。

也可以只输入函数 `grid` 添加网格线，再一次输入函数 `grid`，则取消网格线。

## 4.2.6 图形窗口的分割

MATLAB 的绘图指令可以将窗口分割成几个区域，在各个区域中分别绘图。

函数：`subplot(m, n, p)`

将当前窗口分割成  $m$  行  $n$  列区域，并指定第  $p$  个编号区域为当前绘图区域。区域的编号原则是“先上后下，先左后右”。MATLAB 允许每个编号区域可以以不同的坐标系单独绘图。 $m$ 、 $n$  和  $p$  前面的逗号可以省略。

例 4-11 把当前窗口分割成四个区域，绘制四条函数曲线（如图 4-11 所示）。

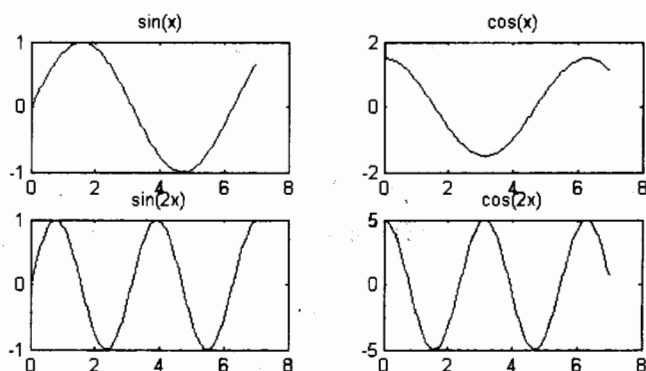


图 4-11 同一窗口的四条函数曲线

```
x=0:0.05:7;
y1=sin(x);
y2=1.5*cos(x);
y3=sin(2*x);
y4=5*cos(2*x);
subplot(221);plot(x,y1);title('sin(x)')
subplot(222);plot(x,y2);title('cos(x)')
subplot(223);plot(x,y3);title('sin(2x)')
subplot(224);plot(x,y4);title('cos(2x)')
```

## 4.2.7 图形的填充

`fill` 函数用于填充二维封闭多边形。其函数格式如下所示。

函数：`fill(x, y, 'color')`

说明：在由数据所构成的多边形内，用所指定的颜色填充。如果该多边形不是封闭的，可以用初始点和终点的连线封闭。颜色控制符同 `plot` 函数，见表 4-1。

例 4-12 绘制正弦函数，并用黑色充填（如图 4-12 所示）。

```
x=0:0.05:7;
y=sin(x);
fill(x,y,'k')
```

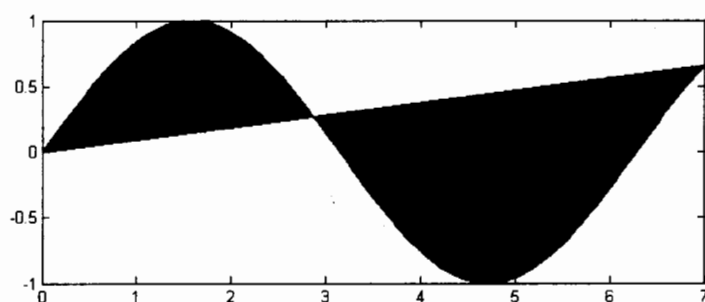


图 4-12 用黑色充填的正弦曲线

可以看到，由于该图形不是封闭的，MATLAB 用初始点和终点连线将其封闭，并填充黑色。

### 4.3 特殊坐标二维图形

MATLAB 提供一些特殊坐标的二维图形函数，如 `semilogx`、`semilogy` 和 `polar` 函数。这些函数与 `plot` 函数功能类似，也可以带参数，参数是对图形的修饰与控制，与 `plot` 函数的参数完全相同。这些绘图函数与 `plot` 函数的区别是将数据绘制到不同的图形坐标上（如表 4-7 所示）。

表 4-7 特殊坐标二维图形函数

函数名称	命令格式	说明
对数坐标图形	<code>semilogx(x, y, 参数)</code>	绘制半对数坐标图形，其中横轴取以 10 为底的对数坐标，纵轴为线性坐标。对 $x$ , $y$ 的要求与 <code>plot</code> 函数相同
	<code>semilogy(x, y, 参数)</code>	绘制半对数坐标图形，其中纵轴取以 10 为底的对数坐标，横轴为线性坐标。对 $x$ , $y$ 的要求与 <code>plot</code> 函数相同
	<code>loglog(x, y, 参数)</code>	绘制坐标轴都取以 10 为底的对数坐标图形。对 $x$ , $y$ 的要求与 <code>plot</code> 函数相同
极坐标图形	<code>polar(theta, radius, 参数)</code>	函数绘制相角为 $\theta$ 、半径为 $\text{radius}$ 的极坐标图形。相角为弧度制

例 4-13 对同一向量分别绘制线性坐标图和三种对数坐标图（如图 4-13 所示）。

```
clear      %清除工作空间变量
clc
y=[0,0.55,2.5,6.1,8.5,12.1,14.6,17,20,22.1];
subplot(221);plot(y);
title('线性坐标图');
subplot(222);semilogx(y);
title('x 轴对数坐标图');
subplot(223);semilogy(y);
title('y 轴对数坐标图');
subplot(224);loglog(y);
title('双对数坐标图');
```

例 4-14 绘制极坐标图（如图 4-14 所示）。

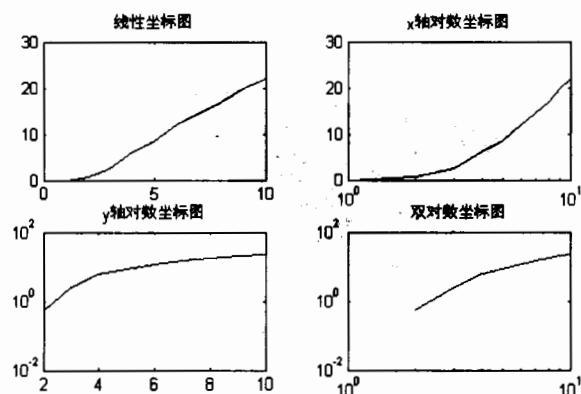


图 4-13 线性坐标图和三种对数坐标图

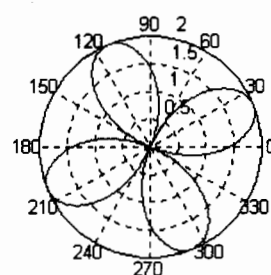


图 4-14 极坐标图

```
clear      %清除工作空间变量
clc
t=0:0.01:2*pi;
r=2*cos(2*(t-pi/8));
polar(t,r)
```

## 4.4 特殊二维图形

MATLAB 支持各种类型的图形绘制，以便于将数据信息更准确而有效地表达出来，图形类型的选择通常取决于数据的特点和表现的形式。在直角坐标系中，MATLAB 能够绘制的特殊二维图形主要有饼图、梯形图、条形图、概率分布图和向量图。饼图常用来表示各种因素所占的百分比，还可以把其中的几个部分突出显示；梯形图用来表示系统中的采样数据；条形图用来表示一些数据的对比情况，有垂直方向的条形图和水平方向的条形图两种；概率分布图多用于研究随机系统的数据分布情况；向量图用于复数绘图。所采用的函数如表 4-8 所示。

表 4-8 特殊二维图形函数

函数名称	命令格式	说明
绘图函数	<code>fplot('x',[xmin,xmax])</code>	$x$ 为函数名。用来绘制给定函数 $x$ 在区间 $[x_{min}, x_{max}]$ 内的变化图形
饼图	<code>pie(x, 参数)</code>	根据矩阵或向量 $x$ 绘制饼图。若 $x$ 为向量，该函数绘制 $x$ 的每一元素占全部向量元素总和的百分比饼图；若 $x$ 为矩阵，该函数绘制 $x$ 的每一元素占全部矩阵元素总和的百分比的饼图。参数表示某元素对应的扇块是否从整个饼图中分离出来，若为零，表示不分离；非零，则分离出来。它的维数应与 $x$ 相同
条形图	<code>bar(x, 参数)</code>	垂直方向的条形图。根据矩阵或 $x$ 向量绘制条形图。若 $x$ 为向量，则以其各元素的序号为各个数据点的横坐标，以 $x$ 向量的各个元素为纵坐标，绘出一个垂直方向的条形图。若为矩阵，同时参数字符串为 <code>group</code> 或默认，则以其各序号为横坐标，每列在其序号坐标上分别以列的各元素为纵坐标，绘出一组垂直方向的条形图；若参数字符串为 <code>stack</code> ，则以其各列序号为横坐标，每列在其列序号坐标上以列向量的累加值为纵坐标，绘出一个垂直方向的分组式条形图。参数 <code>width</code> 给定条形的宽度，默认值为 0.8，若大于 1，则条形图重叠
	<code>barh(x, 参数, )</code>	水平方向的条形图。与垂直方向条形图函数用法相同

(续)

函数名称	命令格式	说明
梯形图	stairs(x)	x 为向量。绘制以 x 向量序号为横坐标, 以 x 向量的各个对应元素为纵坐标的梯形图
	stairs(x,y)	x, y 均为向量。绘制以 x 向量的各个对应元素为横坐标, 以 y 向量的各个对应元素为纵坐标的梯形图
概率分布图	hist(y,x)	x, y 均为向量。绘制 y 在以 x 为中心的区间中分布个数的条形图
原子向量图	compass(x)	x 为向量。绘制相对原点的向量图
	compass(x,y)	以复数坐标系的原点为起点, 绘制出有箭头的一组复数向量, 其中向量 x 表示复数的实部, 向量 y 表示复数的虚部
水平向量图	feather(x)	x, y 均为向量。与 compass 函数的用法相同, 两者的区别是起点不同, compass 函数起始于坐标原点, feather 函数起始于向量各元素的序号
	feather(x, y)	

### 1. 绘图函数 fplot

例 4-15 绘制函数  $f(x) = \cos(\tan(\pi x))$  的曲线 (如图 4-15 所示)。

```
fplot('cos(tan(pi*x))',[-0.4,1.4])
```

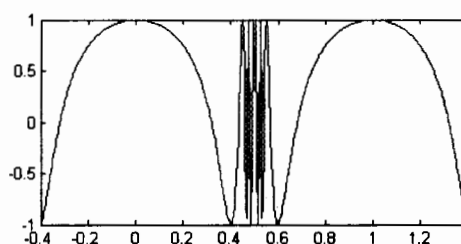


图 4-15 自适应的函数图形

fplot 函数不同于前面介绍的绘图函数, 是能对函数自适应采样的绘图函数。

#### 注意:

fplot 函数可自适应地对函数进行采样, 能够发现并对曲线变化率大的区段进行密集采样, 可以更好地反映函数的变化规律。

### 2. 饼图和条形图

饼图和条形图都常用在统计中, 饼图表示各因素所占的百分比; 条形图用来表示一些数据的对比情况。MATLAB 提供了两类条形图的函数: 一类是垂直方向的条形图; 另一类是水平方向的条形图。另外, bar3 函数可绘制三维条形图。

例 4-16 某次考试学生成绩优秀的占 8%、良好的占 20%、中等的占 36%、及格的占 24%、不及格的占 12%。分别用饼图、条形图表示 (见图 4-16)。

```
x=[8 20 36 24 12];
subplot(221);pie(x,[1 0 0 0 1]);
title('饼图');
subplot(222);bar(x,'group');
title('垂直条形图');
```

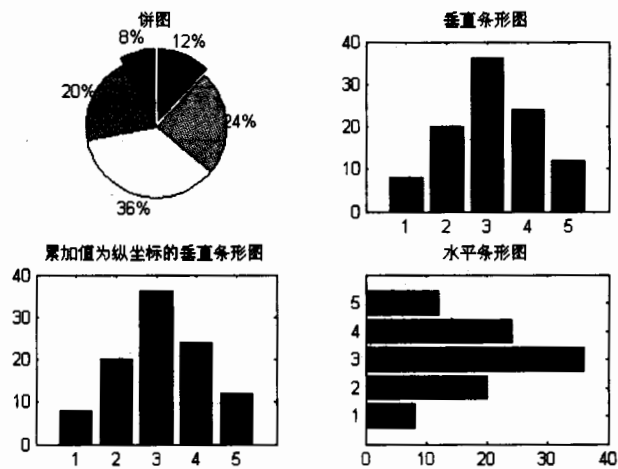


图 4-16 学生成绩统计（饼图的第 1 块和第 5 块分离出来突出显示）

```
subplot(223);bar(x,'stack');
title('累加值为纵坐标的垂直条形图');
subplot(224);barh(x,'group');
title('水平条形图');
```

### 3. 梯形图

梯形图可以用来表示系统中的采样数据。

例 4-17 用梯形图分别表示向量和正弦函数（如图 4-17 所示）。

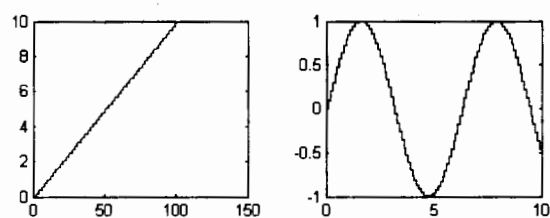


图 4-17 向量和正弦函数

```
x=0:0.1:10;
y=sin(x);
subplot(121);stairs(x);
subplot(122);stairs(x,y);
```

### 4. 概率分布图

研究随机系统时，常常用到概率分布图。

例 4-18 绘出 1000 个随机点的概率分布图（如图 4-18 所示）。

```
x=randn(1,1000);
y=-2:0.1:2;
hist(x,y)
```

## 5. 向量图

向量图有原点向量图和水平向量图两种，二者的区别仅在于坐标的起点不同。向量图可以绘制复数图形。

例 4-19 绘出复数向量的原子向量图和水平向量图（如图 4-19 所示）。

```
x=[-2+3j, 3+4j, 1-7j];
subplot(121);compass(x);
real=[-2 3 1];
imag=[3 4 -7];
subplot(122);feather(real,imag);
```

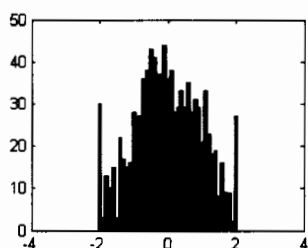


图 4-18 概率分布图

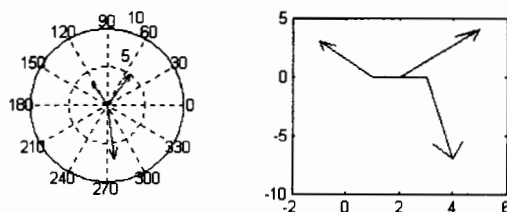


图 4-19 复数向量的原子向量图和水平向量图

## 4.5 三维图形

### 4.5.1 三维数据的产生

MATLAB 提供了两个用于生成三维数据的函数。

#### 1. peaks 函数

用于创建双峰函数和用双峰函数绘图，其基本调用格式为：

```
[x, y, z]=peaks(n)
```

说明：peaks 函数创建  $x$ 、 $y$ 、 $z$  均为  $n \times n$  阶的方阵数据。其中  $x$  的每一列的元素都相同，每一行的元素均是在  $[-3, 3]$  区间内的  $n$  等分； $y$  的每一行的元素都相同，每一列的元素均是在  $[-3, 3]$  区间内的  $n$  等分； $n$  的默认值为 49。 $z$  是  $x$  和  $y$  的函数，有如下关系：

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

#### 2. meshgrid 函数

按指定方式创建网格矩阵，其基本调用格式为：

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

说明：将向量  $x(1 \times m)$ 、 $y(1 \times n)$ 、 $z(1 \times k)$  转换为三维网格数据。 $X$ 、 $Y$ 、 $Z$  是三个  $m \times n \times k$  阶的矩阵。如果默认了参数  $Z$ ，则创建二维网格数据。

例 4-20 创建二维网格数据。

```

>>clear      %清除工作空间变量
>>clc
>> x=[1,2,3,4];
>> y=[5,6,7,8];
>> [X,Y]=meshgrid(x,y,z)
X =
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
Y =
     5     5     5     5
     6     6     6     6
     7     7     7     7
     8     8     8     8

```

## 4.5.2 三维曲线图

MATLAB 提供了 `plot3` 函数绘制三维曲线图形。该函数将绘制二维图形的函数 `plot` 的特性扩展到了三维空间，其功能和使用方法类似于绘制二维图形的函数。其格式为：

```
plot3(x1, y1, z1, '参数 1', x2, y2, z2, '参数 2', ...)
```

(1) 如果  $x$ ,  $y$  和  $z$  是同样长度的矢量，则绘制出一条在三维空间贯穿的曲线。

**例 4-21** 建立并绘制一条三维曲线（如图 4-20 所示）。

```

clear      %清除工作空间变量
clc
z=0:pi/50:10*pi;
x=sin(z);
y=cos(z);
plot3(x,y,z)

```

(2) 如果  $x$ ,  $y$  和  $z$  是  $m \times n$  阶的矩阵，则绘制出  $m$  条三维空间曲线。

**例 4-22** 绘出多条三维空间曲线图（如图 4-21 所示）。

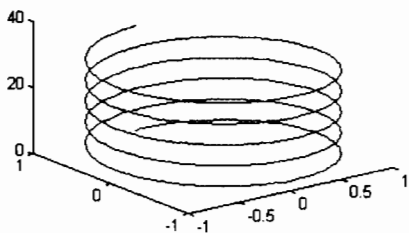


图 4-20 一条三维曲线

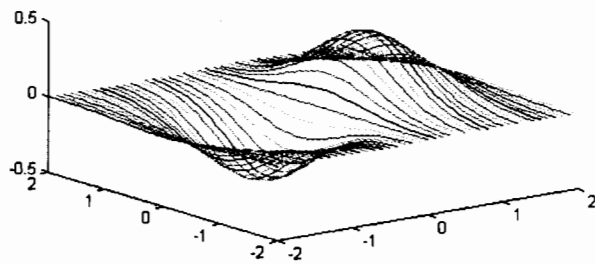


图 4-21 多条三维曲线图

```
clear      %清除工作空间变量
```



```
clc
```

```
[x,y]=meshgrid([-2:0.1:2]);
z=x.*exp(-x.^2-y.^2);
plot3(x,y,z)
```

### 4.5.3 三维曲面图形

MATLAB 还提供了一些函数，可在三维空间中画函数曲面或网格框架的函数。主要函数如表 4-9 所示。

表 4-9 三维曲面图形函数

函数名称	命令格式	说明
三维网格曲面	mesh(x, y, z, c) mesh(x, y, z) mesh(z, c) mesh(z)	当 x, y 为 $n \times m$ 维矩阵时，且 x 矩阵的所有行向量相同、y 矩阵的所有列向量相同时，mesh 函数将自动执行 meshgrid(x, y)，将 x, y 转换为三维网格数据矩阵。z 和 c 分别为 $(m \times n)$ 维矩阵，c 表示网格曲面的颜色分布，若省略，则网格曲面的颜色亮度与 z 方向上的高度值成正比。x, y 若均为省略，则三维网格数据矩阵取值 $x=1:n, y=1:m$
带等高线的三维网格曲面	meshc(x, y, z, c) meshc(x, y, z) meshc(z, c) meshc(z)	绘制带有等高线（XY 平面）的三维网格曲面。这些函数类似于 mesh 函数，不同的是该函数还在 XY 平面上绘制曲面在 Z 轴方向上的等高线
带底座的三维网格曲面	meshz(x, y, z, c) meshz(x, y, z) meshz(z, c) meshz(z)	绘制带有底座的三维网格曲面。这些函数类似 mesh 函数，不同的是该函数还在 XY 平面上绘制曲面的底座
填充颜色的三维网格曲面	surf(x, y, z, c) surf(x, y, z) surf(z, c) surf(z)	函数 mesh 绘制的是连接三维空间的一些四边形所构成的曲面，该曲面只有四边形的边用某种颜色绘出，四边形的内部是透明的。surf 函数绘制的曲面也由一些四边形所构成，不同的是四边形的边是黑色的，其内部用不同的颜色填充

例 4-23 绘制函数  $Z = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$  四种三维网格曲面（如图 4-22 所示）。

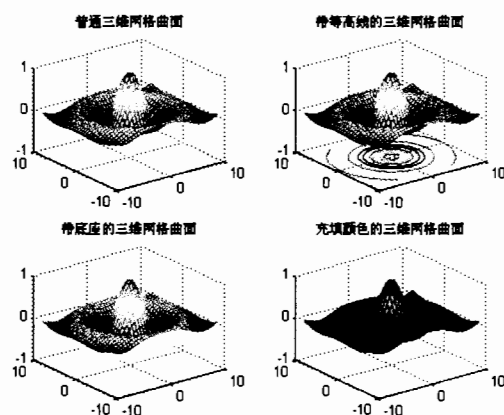


图 4-22 四种三维网格曲面

```
x=-10:0.5:10;
y=-8:0.5:8;
```

```

[X,Y]=meshgrid(x,y);
Z=sin(sqrt(X.^2+Y.^2))./sqrt(X.^2+Y.^2);
subplot(221);
mesh(X,Y,Z);
title('普通三维网格曲面');
subplot(222);
meshc(X,Y,Z);
title('带等高线的三维网格曲面');
subplot(223);
meshz(X,Y,Z);
title('带底座的三维网格曲面');
subplot(224);
surf(X,Y,Z);
title('充填颜色的三维网格曲面');

```

## 4.6 上机实践

### 4.6.1 跟我学

**例 4-24** 在同一坐标内，画出一条正弦曲线和一条余弦曲线，要求正弦曲线用红色实线、数据点用“+”号显示；余弦曲线用黑色点线、数据点用“\*”号显示。并给图形加入网格和标注（如图 4-23 所示）。

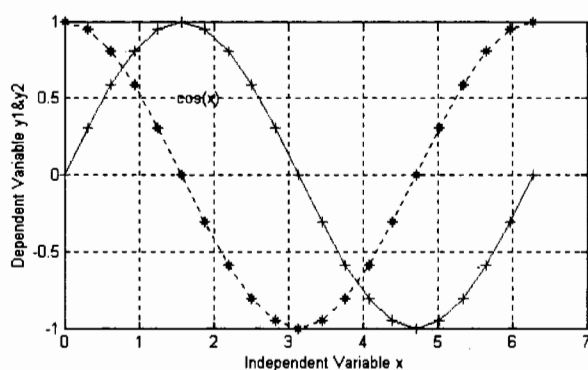


图 4-23 同一坐标内的正弦曲线和余弦曲线

```

clear          %清除工作空间变量
clc

x=0:pi/10:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'r+-',x,y2,'k*:')
grid on        %添加网格
xlabel('Independent Variable x') %横坐标名

```

```
ylabel('Dependent Variable y1&y2') %纵坐标名
text(1.5,0.5,'cos(x)') %指定位置加标注
```

例 4-25 某商场的电脑销售情况如表 4-10 所示，用直方图表示，如图 4-24 所示。

表 4-10 电脑销售一览表

月 份	四 月	六 月	八 月
联想	18	24	15
TCL	5	12	6
同方	28	36	30
实达	17	14	9

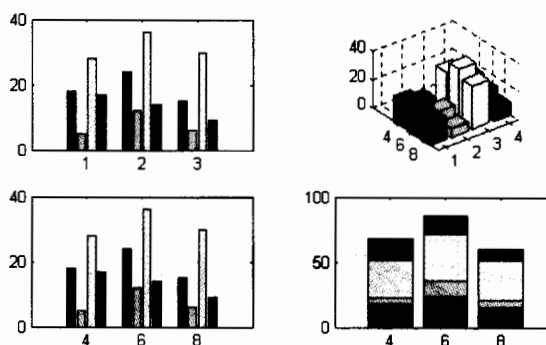


图 4-24 四种直方图

```
clear %清除工作空间变量
clc
y=[18,5,28,17;24,12,36,14;15,6,30,9];
subplot(221);bar(y) %默认横坐标的直方图
x=[4,6,8];
subplot(222);bar3(x,y) %三维直方图
subplot(223);bar(x,y,'grouped')
subplot(224);bar(x,y,'stack')
```

例 4-26 绘制方程  $\begin{cases} x = t \\ y = \sin(t) \\ z = \cos(t) \end{cases}$  在  $t=[0, 2\pi]$  区间的三维曲线（见图 4-25）。

```
x=0:pi/10:2*pi;
y1=sin(x);
y2=cos(x);
plot3(y1,y2,x,'m:p') %画三维曲线并修饰
grid on %添加网格
```

例 4-27 绘制方程  $\sqrt{4 - \frac{x^2}{9} - \frac{y^2}{4}}$  在  $x = [-2, 2]$ ,  $y = [-1, 1]$  区间的图形（如图 4-26 所示）。

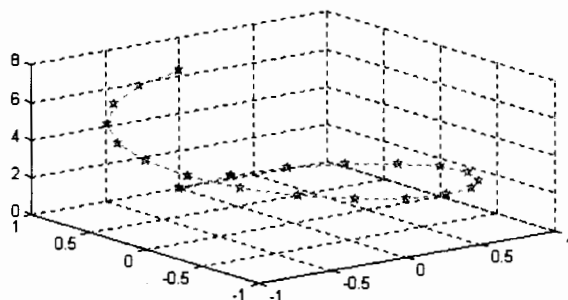


图 4-25 三维曲线图

```
x=-2:0.4:2;
y=-1:0.2:1;
[x,y]=meshgrid(x,y);      %生成三维数据
z=sqrt(4-x.^2/9-y.^2/4);
surf(x,y,z)
grid on
```

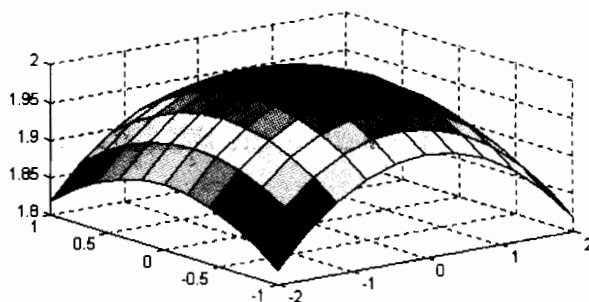


图 4-26 椭圆形着色表面

#### 4.6.2 自己练

1. 某班计算机考试成绩，90 分以上的同学有 8 人，80 分至 90 分的同学有 25 人，70 分至 80 分的同学有 15 人，60 分至 70 分的同学有 12 人，60 分以下的同学有 9 人，画出饼图并让不及格的人数突出显示。
2. 把当前窗口分成四个区域，用不同的颜色和线条分别绘制  $\sin(x)$ 、 $\cos(x)$ 、 $e^x$ 、 $\log(x)$  的函数图形，并加入文字标示和网格。
3. 在极坐标中绘制函数  $\cos(t)\sin(t)$ ， $t = [0 \ 2\pi]$  区间的曲线图。
4. 绘制方程  $\sqrt{5 - \frac{x^3}{3} - \frac{y^2}{7}}$  在  $x = [-2 \ 2]$ ， $y = [-1 \ 1]$  区间的图形。

#### 4.7 本章小结

MATLAB 具有强大的图形绘制能力，用户只需指定绘图方式，提供绘图数据，即可绘出二维或三维图形。本章主要讲解了二维平面绘图及图形修饰函数、对数坐标图形、极坐标

图形；饼图、条形图等特殊的二维图形。三维图形中讲解了三维数据的生成、三维曲线和三维曲面的绘制方法。本章涉及到的主要函数如表 4-11 所示。

表 4-11 本章主要函数

函数名称	功能	函数名称	功能
plot	二维绘图	axis	坐标轴控制
set	坐标轴刻度	hold	图形保持或解除
title	图形标题	xlabel	x 轴标注
text	指定位置添加标注	ylabel	y 轴标注
gtext	在鼠标处添加标注	legend	图例注解
grid	添加或取消网格	subplot	窗口分割
fill	填充图形	polar	极坐标
semilogx	对数坐标	fplot	绘图函数
semilogy		pie	饼图
loglog		bar (barh)	条形图
hist	概率分布图	bar3	三维条形图
compass	原子向量图	stairs	梯形图
peaks	三维双峰数据	feather	水平向量图
plot3	三维曲线	meshgrid	三维网格矩阵
meshc	带等高线的曲面	mesh	三维网格曲面
meshz	带底座的曲面	surf	充填颜色的曲面

## 4.8 习题

1. 选择合适的步长绘制出下列函数的图形。

- (1)  $\ln \frac{1-x}{1+x}, x \in (-1, 1)$     (2)  $\sqrt{\cos x}, x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$   
 (3)  $\sin\left(\frac{1}{t}\right), t \in (-1, 0) \cup (0, 1)$     (4)  $\frac{\sin(x)}{x}, x \in (-0.5, 0) \cup (0, 0.5)$

2. 在同一坐标下绘制函数  $x, x^2, -x^2, x \sin(x)$  在  $x \in (0, \pi)$  的曲线。

3. 绘制如下函数的图形  $y = \begin{cases} x, & x \in (-\infty, 1) \\ x^2, & x \in [1, 4] \\ 2^x, & x \in (4, +\infty) \end{cases}$

4. 在极坐标系中绘制下列函数的曲线：

- (1)  $\cos^3(t)-1$     (2)  $\cos(t)\sin(t)$     (3)  $2t^2+1$

5. 绘制极坐标曲线  $\rho = a \sin(b + n\theta)$ ，并分析参数  $a, b, n$  对曲线形状的影响。

6. 分别用 plot 和 fplot 函数绘制  $y = \sin \frac{1}{x}$  的曲线，并分析两条曲线的差别。

7. 绘制下列函数的带底座的三维图形和带等高线的三维图形。

---

(1)  $f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2}$     (2)  $f(x, y) = xy$     (3)  $f(x, y) = \sin(xy)$

8. 绘制二维正态分布密度函数

$$f(x, y) = \frac{1}{2\pi} \exp\left\{-\frac{1}{2}(x^2 + y^2)\right\}$$
 的三维图形。

9. 用不同的线型和颜色在同一坐标内绘制曲线  $y = 2e^{-0.5x} \sin(2\pi x)$  图形。

## 第5章 MATLAB 符号计算

在本章中，读者可以熟悉符号计算的基本命令，学会运用符号计算进行基本的数学运算。本章重点是 5.2 符号微积分，难点是 5.2.4 积分变换。

### 5.1 符号对象

#### 5.1.1 创建符号变量和符号矩阵

MATLAB 的符号数学工具箱提供了 `sym` 和 `syms` 两个基本函数，用来创建符号变量和符号矩阵。

- 函数 `sym` 的调用格式为：

符号变量名=`sym`（‘表达式’）

说明：函数 `sym` 可创建一个符号变量，表达式可以是字符、字符串、数学表达式或字符表达式等。

- 函数 `syms` 的调用格式：

`syms` 符号变量名 1 符号变量名 2 符号变量名 3 ...

说明：函数 `syms` 可一次创建多个符号变量。

**例 5-1** 创建符号变量。

```
>> a=sym('matlab')
a =
    matlab
>> b=sym('3*x^2+4*x+7')
b =
    3*x^2+4*x+7
```

**例 5-2** 创建符号变量 A、B、C。

```
>> syms A B C
```

在工作空间浏览器上可以看到 A、B、C 三个符号变量。使用 `sym` 函数和 `syms` 函数也可以创建符号矩阵。符号矩阵是一个数组，它的元素是符号表达式。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。

**例 5-3** 创建符号矩阵。



```
>> e=[1 3 5;2 4 6;7 9 11];    %建立数值矩阵
>> m=sym(e)                  %创建符号矩阵

m =
```

```
[ 1, 3, 5]
[ 2, 4, 6]
[ 7, 9, 11]
```

#### 例 5-4 创建循环符号矩阵。

```
>> syms a b c d
>> n=[a b c d;b c d a;c d a b;d a c b]
n =
    [ a, b, c, d]
    [ b, c, d, a]
    [ c, d, a, b]
    [ d, a, c, b]
```

在命令窗口的显示中，数值矩阵只显示元素的数值，而符号矩阵的每行元素放在一对括号内；在工作空间窗口显示的变量图标两者也不同，数值矩阵的图标为 ，符号矩阵（也称为符号对象）的图标为 ，二者很容易区分。

### 5.1.2 符号表达式的基本运算函数

符号表达式的运算与普通数值运算的方式不同，它的运算结果是符号表达式或符号矩阵。在 MATLAB 运算中，浮点运算速度最快，而符号计算占用时间和内存都比较多，但它的计算结果最精确。在默认情况下，当用函数 sym 生成符号变量后，MATLAB 将对这些变量进行符号计算。在 MATLAB 符号计算工具箱中提供了很多函数用于符号计算。下面将介绍一些常用的符号运算函数，如表 5-1 所示。

表 5-1 常用的符号函数

函 数 格 式	说 明
symadd( $S_1$ , $S_2$ )	符号表达式 $S_1$ 加上符号表达式 $S_2$
symsub( $S_1$ , $S_2$ )	符号表达式 $S_1$ 减去符号表达式 $S_2$
symmul( $S_1$ , $S_2$ )	符号表达式 $S_1$ 乘上符号表达式 $S_2$
symdiv( $S_1$ , $S_2$ )	符号表达式 $S_1$ 除符号表达式 $S_2$
sympow( $S$ , $p$ )	符号表达式 $S_1$ 的 $p$ 次幂， $p$ 可以是表达式

#### 例 5-5 计算表达式 $x^3-1$ 与表达式 $x-1$ 的和、差、积、商和乘方。

```
>> syms x
>> s1=x^3-1;
>> s2=x-1;
>> symadd(s1,s2)
ans =
    x^3-2+x
>> symsub(s1,s2)
ans =
    x^3-x
```



```
>> symmul(s1,s2)
ans =
(x^3-1)*(x-1)
>> symdiv(s1,s2)
ans =
(x^3-1)/(x-1)
>> sympow(s1,s2)
ans =
(x^3-1)^(x-1)
```

注意:

MATLAB 将基于以下规则确定自变量: 除 i 和 j 的小写字母外, 在符号表达式中默认的独立变量是惟一的; 如果没有这种字母, 就选择 x 作为独立变量; 如字符不是惟一的, 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。

### 5.1.3 符号表达式的化简函数

符号数学工具箱提供了符号表达式的因式分解、展开、合并、化简、通分等函数, 如表 5-2 所示。

表 5-2 符号表达式的化简函数

函 数 格 式	说 明	函 数 格 式	说 明
collect(s, x)	合并自变量 x 的同幂系数	simple(s)	寻找表达式的最简型
expand(s)	符号表达式 s 的展开	simplify(s)	符号表达式的化简
factor(s)	因式分解	radsimp(s)	对含根式的表达式 s 化简
numden(s)	符号表达式 s 的分式通分	horner(s)	符号表达式 s 的嵌套形式

例 5-6 对表达式  $f = x^3 - 1$  进行因式分解。

```
>> syms x %在命令窗口创建符号变量 x
>> f=factor(x^3-1)

f =
(x-1)*(x^2+x+1)
```

例 5-7 展开三角表达式  $\sin(a+b)$ 。

```
>> s=sym('sin(a+b)'); %用 sym 函数创建符号变量
>> expand(s)
ans =
sin(a)*cos(b)+cos(a)*sin(b)
```

例 5-8 对表达式  $f = x(x(x-8)+6)t$ , 分别将自变量 x 和 t 的同类项合并。

```

>> syms x t          %创建符号变量 x, t
>> f=x*(x*(x-8)+6)*t;
>> collect(f)         %按默认的变量 x 合并
ans =
      t*x^3-8*t*x^2+6*t*x
>> collect(f,t)       %按变量 t 合并
ans =
      x*(x*(x-8)+6)*t

```

例 5-9 化简表达式  $f = \sin^2(x) + \cos^2(x)$ 。

```

>> syms x
>> f=sin(x)^2+cos(x)^2;
>> simplify(f)
ans =
      1

```

例 5-10 化简分式  $(4x^2+8x+3)/(2x+1)$ 。

```

>> syms x
>> s=(4*x^2+8*x+3)/(2*x+1);
>> simplify(s)
ans =
      2*x+3

```

例 5-11 对表达式  $f = \frac{x}{y} - \frac{y}{x}$  进行通分。

```

>> syms x y
>> f=x/y-y/x;
>> [m,n]=numden(f)    %m 为分子, n 为分母
m =
      x^2-y^2
n =
      y*x

```

例 5-12 对表达式  $f = x^4 + 3x^3 - 7x^2 + 12$  进行嵌套形式重写。

```

>> syms x
>> f=x^4+3*x^3-7*x^2+12;
>> horner(f)
ans =
      12+(-7+(3+x)*x)*x^2

```

#### 5.1.4 符号表达式的替换函数

MATLAB 的符号数学工具箱提供了两个符号表达式的替换函数 `subexpr` 和 `subs`，可以通过符号替换使表达式的输出形式化简，得到一个简单的表达式。

- 函数 `Subexpr` 的调用格式为：

```
[R, SYM]=subexpr(S, SYM)
```

说明：此函数用变量 SYM（字符或字符串）的值代替符号表达式 S 中重复出现的字符串，R 是返回替换后的结果。

**例 5-13** 求解并化简三次方程  $ax^3+bx+1=0$  符号解。

在命令窗口创建符号表达式，并通过符号替换进行化简。

```
>> t=solve('a*x^3+b*x+1=0')           %化简的符号解

t =
[1/6/a*((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(1/3)-2*b/((-108+12*3^(1/2))*((4*b^3+
27*a)/a)^(1/2))*a^2)^(1/3)]
[-1/12/a*((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(1/3)+b/((-108+12*3^(1/2))*((4*b^3+
27*a)/a)^(1/2))*a^2)^(1/3)+1/2*i*3^(1/2)*(1/6/a*((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(
1/3)+2*b/((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(1/3))]
[-1/12/a*((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(1/3)+b/((-108+12*3^(1/2))*((4*b^3+
27*a)/a)^(1/2))*a^2)^(1/3)-1/2*i*3^(1/2)*(1/6/a*((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(
1/3)+2*b/((-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2)^(1/3)]]

>> [r,s]=subexpr(t,'s')               %化简后符号解

r =
[
1/6/a*s^(1/3)-2*b/s^(1/3)]
[-1/12/a*s^(1/3)+b/s^(1/3)+1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2*b/s^(1/3))]
[-1/12/a*s^(1/3)+b/s^(1/3)-1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2*b/s^(1/3))]
s =
(-108+12*3^(1/2))*((4*b^3+27*a)/a)^(1/2))*a^2
```

● 函数 subs 的调用格式：

```
R=subs(S, old, new)
```

说明：该函数是用新的符号变量 new 替换原来符号表达式 S 中的变量 old，R 是替换后的符号表达式。需要注意的是，当变量 new 是数值形式时，显示的结果虽然是数值，但它事实上是符号变量。要强制地求值需要用 vpa 函数。

**例 5-14** 求表达式  $\frac{3x^3+x^2-1}{x^2+1}$  在  $x=1$  时的代数值。

```
>>clear
>>clc
>>syms x
>>s=(3*x^3+x^2-1)/(x^2+1);
>>r=subs(s,'x','1')

r =
(3*(1)^3+(1)^2-1)/((1)^2+1)

>>vpa(r)           %强制求值
ans =
```



diff(s, x, n)

说明：其中  $s$  为符号表达式， $x$  为自变量， $n$  为求导的阶数。

例 5-16 分别计算表达式  $x^5$  的一阶导数和三阶导数。

```
>>clear
>>clc
>>syms x
>>diff(x^5)
ans =
    5*x^4
>>diff(x^5,3)
ans =
    60*x^2
```

注意：

$x$  和  $n$  都可默认。默认  $x$  时，取系统默认自变量；默认  $n$  时为求一阶导数。

### 5.2.3 符号积分

积分算法是非结构性的，许多函数的原函数存在，但不可用有限解析表达式表示，即使可以求积分的函数，其求积分过程也可能很复杂，但利用 MATLAB 求积分就非常容易。在 MATLAB 的符号数学工具箱中，表达式的积分由函数 `int` 实现，该函数可求不定积分和定积分，其调用格式如表 5-4 所示。

表 5-4 符号积分的函数格式

函数格式	说明
<code>int(s)</code>	求符号表达式 $s$ 对于默认自变量的不定积分
<code>int(s, x)</code>	求符号表达式 $s$ 对于自变量 $x$ 的不定积分
<code>int(s, a, b)</code>	求符号表达式 $s$ 对于默认自变量从 $a$ 到 $b$ 的定积分
<code>int(s, x, a, b)</code>	求符号表达式 $s$ 对于自变量 $x$ 从 $a$ 到 $b$ 的定积分

例 5-17 分别计算下列表达式的积分：

$$(1) \int (4-3x^2)^2 dx \quad (2) \int \frac{x}{x+y} dx \quad (3) \int \frac{x}{x+y} dy \quad (4) \int_1^3 \frac{x^2}{x+2} dx$$

在命令窗口创建符号变量  $x$  和  $y$ ，分别计算上面各表达式的积分。

```
>>clear
>>clc
>>syms x y
>>s=(4-3*x^2)^2;
>>int(s)
ans =
```

```

          9/5*x^5-8*x^3+16*x
>> int(x/(x+y),x)
ans =
      x-y*log(x+y)
>> int(x/(x+y),y)
ans =
      x*log(x+y)
>> int(x^2/(x+2),x,1,3)
ans =
      4*log(5)-4*log(3)
>> double(ans)
ans =
      2.0433

```

## 5.2.4 积分变换

积分变换就是通过积分运算把一个函数  $f$  (原函数) 变成另外一个函数  $F$  (像函数)。变化的过程是:

$$F(t) = \int_a^b f(x)K(x,t)dx \quad (5-1)$$

其中二元函数  $K(x,t)$  称为变换的核, 变换的核决定了变换的不同名称。在一定的条件下原函数和像函数之间是一一对应的, 可以相互转化。积分变换的意义是换一个角度来认识函数, 积分变换的一项基本应用是解微分方程, 求解过程是基于这样一种想法: 假如不容易从原方程直接求得解  $f$ , 则对原方程进行变换, 如果能从变换后的方程中求得解  $F$ , 则对  $F$  进行逆变换, 即可求得原方程的解  $f$ , 当然, 在选择变换的核时, 应该使得变换以后的方程比原方程容易求解。MATLAB 提供的变换函数如表 5-5 所示。

表 5-5 常用的积分变换函数

函数名称	函数格式	说明
傅立叶变换	fourier(fx, x, t)	fx 为函数 $f(x)$ 的符号表达式、 $x$ 为自变量、 $t$ 为像函数 $F(t)$ 的自变量。结果为函数 $f(x)$ 的傅立叶像函数 $F(t)$
	ifourier(Fw, t, x)	Fw 为函数 $F(t)$ 的符号表达式、 $t$ 为自变量、 $x$ 为原函数 $f(x)$ 的自变量。结果为函数 $F(t)$ 的傅立叶原函数 $f(x)$
拉普拉斯变换	laplace(fx, x, t)	结果为函数 $f(x)$ 的拉普拉斯像函数 $F(t)$
	ilaplace(Fw, t, x)	结果为函数 $F(t)$ 的拉普拉斯原函数 $f(x)$
Z 变换	ztrans(fx, x, t)	结果为函数 $f(x)$ 的 Z 变换像函数 $F(t)$
	iztrans(Fw, t, x)	结果为函数 $F(t)$ 的 Z 变换原函数 $f(x)$

例 5-18 求函数  $y = e^{-x^2}$  的傅立叶变换及其逆变换。

```

>> syms x t
>> y=exp(-x^2);
>> Ft=fourier(y,x,t)      %傅立叶变换
Ft =
      pi^(1/2)*exp(-1/4*t^2)

```

```
>> fx=ifourier(Ft,t,x)    %傅立叶逆变换
fx =
    1/2*4^(1/2)*exp(-x^2)
```

例 5-19 计算  $y = -x^3$  的拉普拉斯变换及其逆变换。

```
>> syms x t
>> y=-x^3;
>> Lt=laplace(y,x,t)      %对函数 y 进行拉普拉斯变换
Lt =
    -6/t^4
>> Lx=ilaplace(Ft,t,x)    %对函数 Ft 进行拉普拉斯逆变换
Lx =
    -x^3
```

例 5-20 求数列  $y = 3e^{2x}$  的 Z 变换及其逆变换

```
>> syms x t
>> y=3*exp(2*x);
>> Zt=ztrans(y,x,t)      %求 y 的 Z 变换
Zt =
    3*t/exp(2)/(t/exp(2)-1)
>> Zx=iztrans(Ft,t,x)    %求 Ft 的逆 Z 变换
Zx =
    3*exp(2)^x
```

## 5.3 符号方程求解

### 5.3.1 代数方程

代数方程是指未涉及微积分运算的方程，相对比较简单。在 MATLAB 符号数学工具箱中，求解用符号表达式表示的代数方程可由函数 solve 实现，其调用格式如表 5-6 所示。

表 5-6 符号方程的函数

函数格式	说 明
<code>solve(s)</code>	求解符号表达式 $s=0$ 的代数方程，自变量为默认自变量
<code>solve(s, x)</code>	求解符号表达式 $s=0$ 的代数方程，自变量为 $x$
<code>solve(s<sub>1</sub>,s<sub>2</sub>,...,s<sub>n</sub>, x<sub>1</sub>,x<sub>2</sub>,...,x<sub>n</sub>)</code>	求解由符号表达式 $s_1,s_2,...,s_n$ 组成的代数方程组，自变量分别为 $x_1,x_2,...,x_n$

例 5-21 分别求解代数方程  $ax^2+bx+c=0$ 。

在命令窗口创建符号变量  $a$ 、 $b$ 、 $c$  和  $x$ 。

```
>>clear
>>clc
>> syms a b c x
>> s=a*x^2+b*x+c;
```

```
>> solve(s)

ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

例 5-22 求解代数方程组：

$$\begin{cases} 2x^2 + y^2 - 3z = 4 \\ x + z = 3 \\ x - 2y = 3z \end{cases}$$

在命令窗口创建符号变量  $x$ 、 $y$ 、 $z$ ，求解方程组。

```
>> syms x y z
>> s1=2*x^2+y^2-3*z-4;
>> s2=y+z-3;
>> s3=x-2*y-3*z;
>> [x,y,z]=solve(s1,s2,s3)
x =
[ 7/2+1/6*i*699^(1/2)]
[ 7/2-1/6*i*699^(1/2)]
y =
[ 11/2-1/6*i*699^(1/2)]
[ 11/2+1/6*i*699^(1/2)]
z =
[ -5/2+1/6*i*699^(1/2)]
[ -5/2-1/6*i*699^(1/2)]
```

### 5.3.2 符号微分方程求解

在 MATLAB 中，用大写字母 D 来表示微分方程的导数。例如：Dy 表示  $y'$ ，D2y 表示  $y''$ 。D2y+Dy+x-10=0 表示微分方程  $y'' + y' + x - 10 = 0$ 。Dy(0)=3 表示  $y'(0) = 3$ 。

在符号数学工具箱中，求解表达式微分方程的符号解由函数 dsolve 实现，其调用格式为：

● r=dsolve('eq','cond','var')

说明：式中 eq 代表常微分方程，cond 代表常微分方程的边界条件或初始条件，var 代表自变量，默认是按系统默认原则处理。该函数可求解微分方程的特解。

● r=dsolve('eq1','eq2'...'eqN','cond1','cond2'...'condN','var1'...'varN')

说明：该函数求解由 eq1, eq2, ... 指定的常微分方程组在条件 cond1, cond2, ..., condN 下的符号解，若不给出初始条件，则求方程组的通解。var1, ..., varN 为求解变量，如果不指定，将为默认自变量。

例 5-23 求微分方程  $\frac{dy}{dt} = \frac{t^2 + y^2}{2t^2}$  的通解。

在命令窗口分别输入表达式，求解方程。



```
>> y=dsolve('Dy-(t^2+y^2)/t^2/2','t')      %方程右端的零可以不写
y =
t*(-log(t)+2+C1)/(-log(t)+C1)      %通解
```

**例 5-24** 求微分方程  $\frac{dy}{dx} = 2xy^2$  的通解和当  $y(0)=1$  时的特解。

在命令窗口输入表达式，求解方程。

```
>> y=dsolve('Dy=2*x*y^2','x')              %求通解
y =
-1/(x^2-C1)
>> y=dsolve('Dy=2*x*y^2','y(0)=1','x')      %求特解
y =
-1/(x^2-1)
```

**例 5-25** 求微分方程的通解。

$$\begin{cases} \frac{dx}{dt} = 4x - 2y \\ \frac{dy}{dt} = 2x - y \end{cases}$$

```
>> [x,y]=dsolve('Dx=4*x-2*y','Dy=2*x-y','t')

x =
-1/3*C1+4/3*C1*exp(3*t)-2/3*C2*exp(3*t)+2/3*C2
y =
2/3*C1*exp(3*t)-2/3*C1+4/3*C2-1/3*C2*exp(3*t)
```

## 5.4 级数

### 5.4.1 级数的符号求和

数值级数和函数级数是高等数学的重点研究内容，也是物理学以及其他工程技术学科的重要理论基础和分析工具。在 MATLAB 符号数学工具中，级数表达式的求和由函数 `symsum` 实现，其调用格式如表 5-7 所示。

表 5-7 级数求和函数格式

函数格式	说明
<code>symsum(S)</code>	计算符号表达式 $S$ （表示级数的通项）对于默认自变量的不定和
<code>symsum(S, x)</code>	计算符号表达式 $S$ 对于自变量 $x$ 的不定和
<code>symsum(S, a, b)</code>	计算符号表达式 $S$ 对于默认自变量从 $a$ 到 $b$ 的有限和
<code>symsum(S, x, a, b)</code>	计算符号表达式 $S$ 对于自变量 $x$ 从 $a$ 到 $b$ 的有限和

**例 5-26** 分别计算表达式  $\sum k$ 、 $\sum_{k=1}^{10} (k^2 - 3)$ 、 $\sum_{k=1}^{\infty} \frac{x^k}{k}$ 。

在命令窗口创建符号变量  $k$  和  $x$ ，分别计算上面各表达式。

```
>>clear
>>clc
>>syms x k
>>symsum(k)
ans =
      1/2*k^2-1/2*k
>>symsum(k^2-3,0,10)
ans =
      352
>>symsum(x^k/k,k,1,inf)
ans =
    -log(1-x)
```

## 5.4.2 函数的泰勒级数

泰勒级数可以将一个任意函数表示为一个幂级数，并且，在许多情况下，只需要取幂级数的前有限项来表示该函数，这对于大多数工程应用问题来说，精度已经足够。在 MATLAB 符号数学工具中，表达式的 Taylor 级数展开由函数 `taylor` 实现，其调用格式如表 5-8 所示。

表 5-8 泰勒级数函数格式

函数格式	说 明
<code>taylor(s)</code>	计算符号表达式 $s$ 在默认自变量等于 0 处的 5 阶 Taylor 级数展开式
<code>taylor(s,n)</code>	计算符号表达式 $s$ 在默认自变量等于 0 处的 $n-1$ 阶 Taylor 级数展开式
<code>taylor(f,n,a)</code>	计算符号表达式 $s$ 在默认自变量等于 $a$ 处的 $n-1$ 阶 Taylor 级数展开式
<code>taylor(f,x,n,a)</code>	计算符号表达式 $s$ 在自变量 $x$ 等于 $a$ 处的 $n-1$ 阶 Taylor 级数展开式

**例 5-27** 分别计算表达式  $\frac{1-x+x^2}{1+x+x^2}$  的 5 阶泰勒级数展开式和 12 阶泰勒级数展开式。

在命令窗口创建符号变量，分别计算上面各表达式。

```
>>clear
>>clc
>>syms x
>>s=(1-x+x^2)/(1+x+x^2);
>>taylor(s)

ans =
      1-2*x+2*x^2-2*x^4+2*x^5
>>taylor(s,x,12)

ans =
      1-2*x+2*x^2-2*x^4+2*x^5-2*x^7+2*x^8-2*x^10+2*x^11
```

## 5.5 可视化符号函数计算器

MATLAB 的符号数学工具箱中还提供了一个可视化符号函数计算器，它具有功能简单实用、操作方便的优点，同时由于具有可视化界面，为用户提供了直观的函数计算工具。

在 MATLAB 的命令窗口输入“funtool”，即可启动可视化符号函数计算器，其界面如图 5-1 所示。可视化符号函数计算器由三个独立的窗口组成，分别为两个图形窗口和一个函数功能的控制窗口。图形窗口 1 是函数  $f$  的显示窗口，图形窗口 2 是函数  $g$  的显示窗口。下面详细介绍可视化符号计算器的使用。

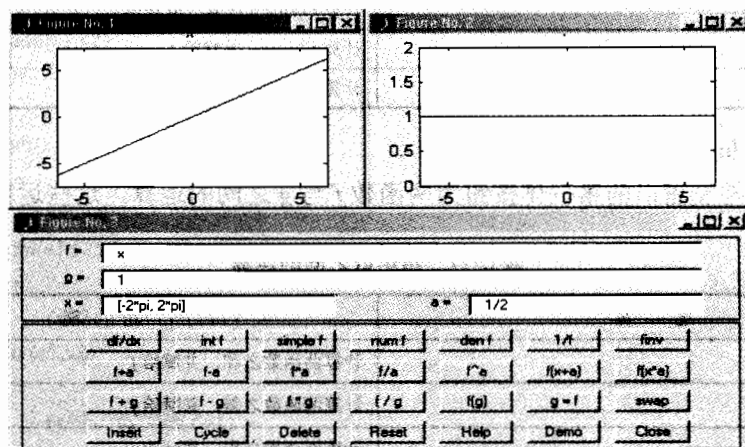


图 5-1 可视化符号函数计算器

### 1. 输入框

在控制窗口的上方有 4 个输入框，用户可以在输入框中输入函数，4 输入框分别为：

- $f=$  为图形窗口 1 的控制函数，其默认值为  $x$ ；
- $g=$  为图形窗口 2 的控制函数，其默认值为  $1$ ；
- $x=$  为两窗口函数自变量取值范围，默认为  $[-2\pi, 2\pi]$ ；
- $a=$  为常数的值，默认值为  $1/2$ 。

### 2. 计算器的功能

- 函数的自运算。

函数功能的控制窗口的第一排按钮为函数的自运算，这些运算见表 5-9。

表 5-9 函数的自运算

函 数	功 能
$df/dx$	计算函数 $f$ 对 $x$ 的导数，并赋给 $f$
$\text{int } f$	计算函数 $f$ 的积分函数，并赋给 $f$
$\text{simple } f$	计算函数 $f$ 的最简表达式，并赋给 $f$
$\text{num } f$	取表达式 $f$ 的分子，并赋给 $f$
$\text{den } f$	取表达式 $f$ 的分母，并赋给 $f$
$1/f$	求 $f$ 的倒数函数，并赋给 $f$
$\text{finv}$	求 $f$ 的反函数，并赋给 $f$

● 函数与常数的运算。

函数功能的控制窗口的第二排按钮为函数与常数之间的运算，这些运算如表 5-10 所示。

表 5-10 函数与常数的运算

函 数	功 能
$f + a$	计算 $f(x)+a$ ，并赋给 $f$
$f - a$	计算 $f(x)-a$ ，并赋给 $f$
$f * a$	计算 $f(x)*a$ ，并赋给 $f$
$f / a$	计算 $f(x)/a$ ，并赋给 $f$
$f^a$	计算 $f(x)$ 的 $a$ 次幂，并赋给 $f$
$f(x+a)$	计算 $f(x+a)$ ，并赋给 $f$
$f(x*a)$	计算 $f(a*x)$ ，并赋给 $f$

● 两函数之间的运算。

函数功能的控制窗口的第三排按钮为两函数  $f$  与  $g$  之间的运算，这些运算见表 5-11。

表 5-11 两函数之间的运算

运 算	功 能
$f + g$	计算两函数之和，并赋给 $f$
$f - g$	计算两函数之差，并赋给 $f$
$f * g$	计算两函数之积，并赋给 $f$
$f / g$	计算两函数之比，并赋给 $f$
$f(g)$	求复合函数 $f(g(x))$
$g = f$	将 $f$ 的函数值赋给 $g$
swap	交换 $f$ 与 $g$ 的函数表达式

● 函数计算器的系统操作。

函数功能的控制窗口的第四排按钮为函数计算器的系统操作，这些按钮是对函数计算器的参数进行设定。这些操作见表 5-12。

表 5-12 函数计算器的系统操作

操 作	功 能
Insert	将当前窗口 1 中的函数加到计算器的典型函数表中
Cycle	在窗口 1 依次演示计算器典型函数表中的函数
Delete	从计算器的典型函数表中删除当前窗口 1 中的函数
Reset	符号函数计算器的功能重置
Help	符号函数计算器的在线帮助
Demo	符号函数计算器功能演示（演示中相应功能按钮变白）
Close	关闭符号函数计算器

## 5.6 上机实践

### 5.6.1 跟我学

例 5-28 计算表达式  $x^2+2x+1$  与表达式  $x+1$  的和、差、积、商和乘方。  
解：

```
>> syms x
>> s1=x^2+2*x+1;s2=x+1;
>> symsadd(s1,s2)
ans =
    x^2+3*x+2
>> symsub(s1,s2)
ans =
    x^2+x
>> symmul(s1,s2)
ans =
    (x^2+2*x+1)*(x+1)
>> symdiv(s1,s2)
ans =
    (x^2+2*x+1)/(x+1)
>> sympow(s1,s2)
ans =
    (x^2+2*x+1)^(x+1)
```

例 5-29 化简表达式  $f=\cos 2x+2\sin^2 x$   
解：

```
>> syms x
>> f=cos(2*x)+2*sin(x)^2;
>> simplify(f)
ans =
    1
```

例 5-30 求下列极限值：(1)  $\lim_{x \rightarrow 0} \frac{\sin 2x}{\sin 5x}$       (2)  $\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^{2x}$

解：

```
>> syms x a;
>> s1=sin(2*x)/sin(5*x);
>> limit(s1,x,0)
ans =
    2/5
>> s2=(1+1/x)^(2*x);
>> limit(s2,x,inf)
```

```
ans =
exp(2)
```

**例 5-31** 分别计算表达式  $x\cos x$  的一阶导数、二阶导数和三阶导数。

解:

```
>> syms x
>> s=x*cos(x);
>> diff(s)
ans =
cos(x)-x*sin(x)
>> diff(s,2)
ans =
-2*sin(x)-x*cos(x)
>> diff(s,3)
ans =
-3*cos(x)+x*sin(x)
```

**例 5-32** 分别计算表达式: (1)  $\int \frac{x^4}{1+x^2} dx$  (2)  $\int_0^2 (3x^2 - x + 1) dx$

解:

```
>> syms x
>> s1=x^4/(1+x^2);
>> int(s1)
ans =
1/3*x^3-x+atan(x)
>> s2=3*x^2-x+1;
>> int(s2,0,2)
ans =
8
```

**例 5-33** 求解代数方程组:

$$\begin{cases} 5x + 6y + 7z = 16 \\ 4x - 5y + z = 7 \\ x + y + 2z = 2 \end{cases}$$

解:

```
>> syms x y z
>> s1=5*x+6*y+7*z-16;
>> s2=4*x-5*y+z-7;
>> s3=x+y+2*z-2;
>> [x,y,z]=solve(s1,s2,s3)
x =
129/34
y =
```

$$z = \frac{45}{34} - \frac{53}{34}i$$

例 5-34 求微分方程  $y' = e^{2x-y}$  的通解和当  $y(0)=0$  时的特解。  
解：

```
>> syms x y
>> y=dsolve('Dy=exp(2*x-y)','x')
y =
log(1/2*exp(2*x)+C1)
>> y=dsolve('Dy=exp(2*x-y)','y(0)=0','x')
y =
log(1/2*exp(2*x)+1/2)
```

例 5-35 求级数之和。

$$s = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \dots + \frac{1}{n^2} + \dots$$

解：

```
>> n=sym('n');
>> s=symsum(1/n^2,n,1,inf)
s =
1/6*pi^2
```

例 5-36 求表达式的泰勒展开式，展开到含  $x^5$  的项。

$$\sqrt{1-2x+x^3} - \sqrt[3]{1-3x+x^2}$$

解：

```
>> x=sym('x');
>> f=sqrt(1-2*x+x^3)-(1-3*x+x^2)^(1/3);
>> taylor(f,6)
ans =
1/6*x^2+x^3+119/72*x^4+239/72*x^5
```

### 5.6.2 自己练

1. 计算表达式  $x+1$  与表达式  $x^2-3x+1$  的和、差、积、商和乘方，并对所得结果进行展开、化简。

2. 求极限  $\lim_{x \rightarrow 0} \ln \frac{\sin x}{x}$ 。

3. 求极限  $\lim_{x \rightarrow +\infty} \left( \sqrt{x^2+x} - \sqrt{x^2-x} \right)$ 。

4. 已知  $y = \tan^2 \sqrt{x + \sqrt{x + \sqrt{2x}}}$ ，求  $y'$ 。

5. 求下列积分 (1)  $\int \frac{1}{x} \sqrt{\frac{x+1}{x-1}} dx$  (2)  $\int_0^\pi \sqrt{\sin x - \sin^3 x} dx$
6. 求微分方程  $y'' + 4y' + 4y = e^{-2x}$  的通解。

## 5.7 本章小结

符号运算功能强大，是 MATLAB 的一个特色。对于许多工程计算问题，MATLAB 中既提供了数值计算方法，也提供了符号计算方法，这两者各有用途：数值计算方法的主要特征是“可行性”，但不能给出解析解；符号方法能给出解析解，但结果一般较复杂、冗长，且对问题的选择性强，很多问题（特别是求解微分方程的问题）无法用符号方法求解。读者应根据问题灵活选择解决方案，一般情况下，符号方法使用要方便一些。本章主要介绍了符号运算的复合、化简、微积分、级数求和以及求解代数方程式、方程组、微分方程式等。学习本章时应加强上机实践练习，熟悉基本的符号运算命令。表 5-13 列出了与本章相关的函数。

表 5-13 本章函数一览

函数名称	功能	函数名称	功能
sym	创建符号变量或表达式	syms	一次创建多符号变量
symadd	符号表达式加	symsub	符号表达式减
symmul	符号表达式乘	symdiv	符号表达式除
sympow	符号表达式乘方	collect	符号表达式的合并
expand	符号表达式的展开	factor	符号表达式因式分解
simple	寻找符号表达式的最简型	simplify	符号表达式化简
radsimp	化简含根式的符号表达式	numden	符号表达式通分
horner	符号表达式嵌套形式	subexpr	符号表达式替换函数
subs	符号表达式的替换函数	vpa	强制求值
limit	符号求极限	diff	符号求导
int	符号求积分	fourier	傅立叶变换
ifourier	傅立叶逆变换	laplace	拉普拉斯变换
ilaplace	拉普拉斯反变换	ztrans	Z 变换
iztrans	Z 逆变换	solve	求解符号表达式的代数方程或方程组
dsolve	求解表达式微分方程的符号解	symsum	符号表达式的求和
taylor	符号表达式的 Taylor 级数的展开		

## 5.8 习题

1. 求极限  $\lim_{x \rightarrow 0} \sqrt{x^2 - 2x + 5}$ 。
2. 求极限  $\lim_{x \rightarrow 0} \frac{\sqrt{1+x^2} - 1}{x}$ 。



3. 求下列积分 (1)  $\int 2xe^{x^2} dx$  (2)  $\int x\sqrt{1-x^2} dx$

(3)  $\int_0^3 \frac{1}{3+2x} dx$

4. 已知  $y = \cos(x^2) \sin^2 \frac{1}{x}$ , 求  $y'$ 。

5. 解微分方程  $x^2 y' + xy = y^2$ ,  $y|_{x=1} = 1$  的特解。

6. 求下列级数之和。

(1)  $s = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + (-1)^{n+1} \frac{1}{n} + \dots$

(2)  $s = x + 2x^2 + 3x^3 + \dots + nx^n + \dots$

(3)  $s = 1 + 4 + 9 + 16 + \dots + 10\,000$

7. 求下面表达式的泰勒展开式, 展开到含  $x^4$  的项。

$$\sqrt[3]{2+x-3x^3} - \sqrt{1-3x+2x^2} + 3x$$

8. 求解下列方程组:

$$(1) \begin{cases} 3x+4y-2z=12 \\ 45x+5y+4z=23 \\ 6x+2y-3z=4 \end{cases} \quad (2) \begin{cases} x-4y+z=1 \\ 6x+4y+14=2z \\ y-13z+5=4x \end{cases}$$

## 第6章 MATLAB 数值计算

在本章中，读者可以了解 MATLAB 强大的数值计算功能，掌握数值计算常用函数的用法，并能够根据实际要求，合理的选择函数，满足计算精度和运算速度的要求。本章的重点是 6.3 数值分析，难点是 6.4.2 龙格—库塔法的实现。

### 6.1 矩阵处理

矩阵运算是 MATLAB 的核心，MATLAB 在提供强大的矩阵运算功能的同时，也提供了许多矩阵处理的函数。

#### 6.1.1 矩阵分析

MATLAB 提供了一些用于矩阵分析和计算矩阵特征的函数，如表 6-1 所示。

表 6-1 矩阵分析和计算矩阵特征的函数

函 数	功 能	函 数	功 能
det(A)	矩阵 A 的行列式	trace(A)	矩阵 A 的迹
norm(A)	矩阵 A 的范数	rank(A)	矩阵 A 的秩
cond(A)	矩阵 A 的条件数	eig(A)	矩阵 A 的全部特征值

例 6-1 求矩阵  $a = [2, 2, -1, 1; 4, 3, -1, 2; 8, 5, -3, 4; 3, 3, -2, 2]$  的行列式、迹、范数、条件数、秩和特征值。

```
>> a=[2,2,-1,1;4,3,-1,2;8,5,-3,4;3,3,-2,2];
>> adet=det(a)
    adet =
         2
>> atrace=trace(a)
    atrace =
         4
>> anorm=norm(a)
    anorm =
    13.3478
>> acond=cond(a)
    acond =
    69.3925
>> arank=rank(a)
    arank =
         4
>> eiga=eig(a)
```

```
eiga =
    4.4971
   -0.3992 + 1.1474i
   -0.3992 - 1.1474i
    0.3013
```

## 6.1.2 矩阵的线性变换

MATLAB 提供了一些矩阵变换函数，可以对矩阵作形式上的变换。具体函数如表 6-2 所示。

表 6-2 矩阵的变换函数

函 数	功 能	函 数	功 能
diag(A)	提取矩阵 A 的对角元素	triu(A)	提取矩阵 A 的上三角矩阵
diag(A, K)	提取矩阵 A 的第 K 条对角元素	tril(A)	提取矩阵 A 的下三角矩阵
fliplr(A)	矩阵 A 左右翻转	flipud(A)	矩阵 A 上下翻转

例 6-2 建立一个 3×3 的魔方矩阵，提取其对角元素和下三角矩阵，并上下翻转。

```
>> A=magic(3)           %建立魔方矩阵，magic 为魔方矩阵函数
A =
     8     1     6
     3     5     7
     4     9     2
>> Adiag=diag(A)
Adiag =
     8
     5
     2
>> Atril=tril(A)
Atril =
     8     0     0
     3     5     0
     4     9     2
>> flipud(A)
ans =
     4     9     2
     3     5     7
     8     1     6
```

## 6.1.3 矩阵分解

矩阵分解是把一个大的矩阵分解为几个比较简单的矩阵的连乘积。可用于解线性方程组。常用矩阵的分解方法有：针对实对称矩阵的 Cholesky 分解、针对一般矩阵的三角分解法和正交分解法。这三种分解方法都是基于三角矩阵的使用，下面就分别介绍。

## 1. Cholesky 分解

MATLAB 的 Cholesky 分解函数是 chol, 其格式如下:

```
G=chol(A)
```

说明:  $A$  表示要分解的实对称矩阵。

例 6-3 建立一个  $4 \times 4$  阶的 pascal 矩阵, 进行 Cholesky 分解。

```
>> A=pascal(4)           %建立 pascal 矩阵, pascal 为建立 pascal 矩阵函数
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20

>> B=chol(A)
B =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
```

提示:

如果一个矩阵的转置矩阵和其自身相等, 称该矩阵为对称矩阵。如果对称矩阵的所有元素均为实数, 则称这样的矩阵为实对称矩阵。

## 2. 三角分解法

矩阵的三角分解是将一个矩阵分解为一个下三角矩阵和一个上三角矩阵的乘积。函数是 lu, 其格式如下:

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

说明:  $A$  表示要分解的矩阵,  $L$  表示分解后得到的下三角矩阵,  $U$  表示分解后得到的上三角矩阵。

例 6-4 对矩阵  $A = [1 \ 3 \ 5; 2 \ 4 \ 6; 7 \ 9 \ 11]$  进行三角分解。

```
>> A=[1 3 5;2 4 6;7 9 11];
>> [L,U]=lu(A)
L =
    0.1429    1.0000         0
    0.2857    0.8333    1.0000
    1.0000         0         0
U =
    7.0000    9.0000   11.0000
     0    1.7143    3.4286
     0         0   -0.0000
```

注意:

上例中,  $U$  是上三角矩阵,  $L$  不是下三角矩阵, 但可以通过行变换变成下三角矩阵。

### 3. 正交分解法

矩阵的三角分解是将一个矩阵分解为一个正交矩阵和一个上三角矩阵的乘积。函数是 `qr`, 其格式如下:

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

说明:  $A$  表示要分解的矩阵,  $Q$  表示分解后得到的正交矩阵,  $R$  表示分解后得到的上三角矩阵。

例 6-5 对矩阵  $A=[1\ 3\ 5; 2\ 4\ 6; 7\ 9\ 11; 2\ 5\ 8]$  进行正交分解。

```
>> A=[1 3 5;2 4 6;7 9 11;2 5 8];
>> [Q,R]=qr(A)
Q =
   -0.1313   -0.5076    0.8326   -0.1784
   -0.2626   -0.3610   -0.4296   -0.7849
   -0.9191    0.3722    0.0995    0.0819
   -0.2626   -0.6881   -0.3350    0.5877
R =
   -7.6158  -11.0297  -14.4437
         0   -3.0569   -6.1139
         0         0   -0.0000
         0         0         0
```

#### 6.1.4 稀疏矩阵

在许多工程实际中, 经常会出现一些只包含几个非零元素, 而其他大量的元素都是为零值的矩阵, 这种矩阵被称为稀疏矩阵。如果按普通的矩阵处理方法来处理这些矩阵, 不但会占用许多存储空间, 同时也严重的影响运行速度。为了避免这些缺点, 对于那些具有大量零元素的矩阵, MATLAB 采用一种不同于一般矩阵的处理方法, 只存储非零元素的数值以及这些元素所对应的下标。设矩阵

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

是具有稀疏特征的矩阵, 其完全存储方式是列存储每个元素, 共有 12 个元素:

2, 0, 0, 0, 0, 0, 3, 0, 0, 1, 0, 0

其稀疏存储方式如下:

(1, 1), 2, (2, 3), 3, (3, 2), 1

稀疏存储方式占用 9 个元素空间，当原矩阵更大且更加“稀疏”时，稀疏存储方式会更有效地提高空间利用率。

虽然 MATLAB 对稀疏矩阵采用了特殊的处理，但其各种运算的规则仍然和完全矩阵一样，另外，也有专门的函数来处理稀疏矩阵。见表 6-3。

表 6-3 处理稀疏矩阵的主要函数

函数格式	说 明
<code>sparse(A)</code>	$A$ 为完全矩阵。将完全矩阵转化为稀疏矩阵
<code>sparse(m,n)</code>	生成 $m \times n$ 阶所有系数为零的稀疏矩阵
<code>sparse(U,V,S)</code>	$U, V, S$ 为等长的向量， $U, V$ 是 $S$ 中元素行和列的下标。建立一个 $U$ 行、 $V$ 列和以 $S$ 为元素的稀疏矩阵
<code>speye(m,n)</code>	生成 $m \times n$ 阶的稀疏矩阵，只有对角线元素为 1
<code>full(A)</code>	$A$ 为稀疏矩阵。将稀疏矩阵转化为完全矩阵
<code>[m,n,s]=find(A)</code>	$A$ 为稀疏矩阵。查看稀疏矩阵， $m, n$ 为非零元素的下标， $s$ 为非零元素

例 6-6 将矩阵  $A$  转化为稀疏矩阵  $B$ ，并查看；再将稀疏矩阵  $B$  转化为完全矩阵  $C$ 。

$$A = \begin{bmatrix} 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 \\ 0 & 22 & 0 & 9 \end{bmatrix}$$

在 MATLAB 命令窗口，输入以下命令：

```
>> A=[0,5,0,0;0,0,0,0;12,0,0,0;0,22,0,9];
>> B=sparse(A)
B =
      (3,1)      12
      (1,2)       5
      (4,2)      22
      (4,4)       9
>> [m,n,s]=find(B)
m =
     3
     1
     4
     4
n =
     1
     2
     2
     4
s =
    12
     5
```

```

22
9
>> C=full(B)
C =
    0     5     0     0
    0     0     0     0
   12     0     0     0
    0    22     0     9

```

## 6.2 数据分析

由于 MATLAB 是面向矩阵运算的，可以让矩阵的每列代表不同的被测变量，相应的行代表被测向量的观测值，这样就很容易通过对矩阵元素的访问进行数据的统计分析。

### 6.2.1 基本数据分析函数

MATLAB 提供的基本数据分析函数主要有求各列的最大元素、最小元素、均值和中值等函数。如表 6-4 所示。

表 6-4 基本统计命令

函 数 名 称	功 能	函 数 名 称	功 能
max(x)	找 x 各列的最大元素	prod(x)	求 x 各列元素之积
mean(x)	求 x 各列的平均值	sum(x)	求 x 各列元素之和
median(x)	找 x 各列的中间值元素	sort(x)	使 x 的各列元素按递增排序
min(x)	找 x 各列的最小元素		

说明：如果输入量  $x$  是向量，则不论是行向量还是列向量，运算是对整个向量进行的；如果输入量  $x$  是矩阵，则运算是按列进行的，即认为每个列是由一个变量的不同情况所得的数据的集合。

**例 6-7** 对矩阵  $A = [4 \ 8 \ -9; 11 \ -12 \ 4; -8 \ 0 \ 5; 0.6 \ 5 \ 10]$ ，求各列的最大元素、中值和平均值。

```

>> A=[4 8 -9;11 -12 4;-8 0 5;0.6 5 10];
>> maxA=max(A)
maxA =
    11     8    10
>> medA=median(A)
medA =
    2.3000    2.5000    4.5000
>> meanA=mean(A)
meanA =
    1.9000    0.2500    2.5000

```

**注意:**

如果想得到整个矩阵的统计结果, 而不是针对每列的, 可以对每列的统计结果再计算一次, 或者使用嵌套。如想得到矩阵 A 的最小元素可用:  
 $\text{minA}=\text{min}(\text{min}(\text{A}))$ 。

## 6.2.2 离差函数和相关函数

离差是描述样本中的数据偏离其中心值的程度, 主要有方差、标准差、极差和协方差。相关是表示两个矩阵的线性联系密切程度的一个统计量, 相关系数值是小于或等于 1 的正数, 当值为 1 时, 表示两个矩阵的线性联系最密切; 当值为 0 时, 表示两个矩阵的线性联系最弱。有自相关和互相关两种。见表 6-5。

表 6-5 离差函数和相关函数

函数名称	功能	函数名称	功能
<code>var(x)</code>	求 $x$ 各列的方差	<code>cov(x, y)</code>	求两个矩阵 $x$ 和 $y$ 的协方差
<code>std(x)</code>	求 $x$ 各列的标准差	<code>corrcoef(x)</code>	求 $x$ 的自相关阵
<code>range(x)</code>	求 $x$ 各列的极差	<code>corrcoef(x, y)</code>	求两个矩阵 $x$ 和 $y$ 的互相关系数
<code>cov(x)</code>	求 $x$ 的协方差阵		

**例 6-8** 建立一个  $3 \times 4$  阶随机矩阵, 求方差、标准差、极差、协方差和自相关阵。

```
>> A=rand(3,4)           % 建立随机矩阵
A =
    0.1389    0.6038    0.0153    0.9318
    0.2028    0.2722    0.7468    0.4660
    0.1987    0.1988    0.4451    0.4186
>> B=var(A)
B =
    0.0013    0.0466    0.1351    0.0804
>> C=std(A)
C =
    0.0358    0.2158    0.3676    0.2836
>> D=range(A)
D =
    0.0639    0.4050    0.7315    0.5132
>> E=cov(A)
E =
    0.0013   -0.0075    0.0123   -0.0100
   -0.0075    0.0466   -0.0658    0.0610
    0.0123   -0.0658    0.1351   -0.0912
   -0.0100    0.0610   -0.0912    0.0804
>> F=corrcoef(A)
F =
    1.0000   -0.9743    0.9337   -0.9902
```



-0.9743	1.0000	-0.8289	0.9962
0.9337	-0.8289	1.0000	-0.8745
-0.9902	0.9962	-0.8745	1.0000

## 6.3 数值分析

### 6.3.1 多项式

多项式是一种基本的数值分析工具，也是一种极简单的函数，很多复杂的函数都可以用多项式逼近。一个  $n$  次多项式有  $n+1$  个系数，在 MATLAB 中，用一个长度为  $n+1$  的行向量表示一个  $n$  次多项式，其中多项式中的各元素按降幂顺序排列，缺少系数的要补 0。如果多项式表示为：

$$P(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\dots+a_{n-1}x+a_n$$

则 MATLAB 表示的系数向量为：

$$P=[a_0 \ a_1 \ a_2 \ \dots \ a_{n-1} \ a_n]$$

如果知道多项式的根为： $ar_1, ar_2, \dots, ar_n$ 。则可以用多项式的根组成的向量来表示多项式：

$$ar=[ar_1 \ ar_2 \ \dots \ ar_n]$$

根向量与系数向量之间满足如下的关系式：

$$(x-ar_1)(x-ar_2)\dots(x-ar_n)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\dots+a_{n-1}x+a_n$$

在 MATLAB 中，与多项式有关的函数如表 6-6 所示。

表 6-6 多项式相关函数

名 称	函 数 格 式	说 明
创建多项式	$P=[a_0 \ a_1 \ a_2 \ \dots \ a_{n-1} \ a_n]$	$P$ 为多项式（以下各函数中 $P$ 均为多项式）， $a_0 \ a_1 \ a_2 \ \dots \ a_{n-1} \ a_n$ 为按降幂顺序排列的多项式系数
	$P=\text{poly}(A)$	$A$ 为向量。创建以向量 $A$ 中元素为根的多项式
求根	$\text{roots}(P)$	求该多项式的根，以列向量的形式给出
求值	$\text{polyval}(P, A)$	当 $A$ 为标量时，求多项式 $P$ 在自变量 $x=A$ 时的值；当 $A$ 为向量时，求 $x$ 分别等于 $A$ 中每个元素时，多项式的值
	$\text{polyvalm}(P, m)$	$m$ 为 $n \times n$ 阶方阵。求 $x$ 分别等于 $m$ 中每一个元素时，多项式的值（结果为 $n \times n$ 阶方阵）
多项式乘法	$\text{conv}(P_1, P_2)$	$P_1$ 多项式与 $P_2$ 多项式相乘
多项式除法	$[q, r]=\text{deconv}(P_1, P_2)$	$P_1$ 多项式与 $P_2$ 多项式相除。 $q$ 为商， $r$ 为余数
多项式求导	$p=\text{polyder}(P)$	多项式 $P$ 的导函数
	$P=\text{polyder}(P_1, P_2)$	$P_1$ 多项式与 $P_2$ 多项式乘积的导函数
	$[q, r]=\text{polyder}(P_1, P_2)$	$P_1$ 多项式与 $P_2$ 多项式相除后的导函数，导函数的分子放入 $q$ ，分母放入 $r$

例 6-9 在 MATLAB 中建立多项式  $f(x)=4x^3-3x^2+2x-5$ ，并求出  $f(x)=0$  时的根及  $x=3$ 、 $x=3.6$  的值。

```
>> P=[4,-3,2,-5];
>> x=roots(P)
```

```

x =
    1.2007
   -0.2253 + 0.9951i
   -0.2253 - 0.9951i
>> x=[3,3.6];
>> f=polyval(P,x)
f =
    82.0000   149.9440

```

#### 注意:

创建多项式时, 必须包括具有零系数的项。除非特别辨认, 否则 MATLAB 无法知道其中的哪一项为零。多项式的加、减运算就是其对应向量的加、减运算, 若两个向量的长度不同, 短的向量要补零, 使两个向量等长。

**例 6-10** 已知多项式  $f(x) = x^4 + 2x^3 - 4x^2 + 3x - 1$  和  $g(x) = x^2 - 1$ , 求两个多项式的和、差、积、商及  $f(x)$  的导数。

命令如下:

```

>> f=[1,2,-4,3,-1];           %创建多项式
>> g=[1,0,1];                 %短多项式补零
>> g1=[0,0,1,0,1];           %多项式加法
>> f+g1
ans =
     1     2    -3     3     0
>> f-g1                         %多项式减法
ans =
     1     2    -5     3    -2
>> conv(f,g)                     %多项式乘法
ans =
     1     2    -3     5    -5     3    -1
>> [q,r]=deconv(f,g)           %多项式除法
q =
     1     2    -5
r =
     0     0     0     1     4
>> polyder(f)                   %多项式求导
ans =
     4     6    -8     3

```

### 6.3.2 数值插值与曲线拟合

在大量的应用领域中, 人们所得到的数据通常都是离散的。如果想得到这些离散点以外的其他数值点, 就需要对这些已知数据点进行插值, 来近似那些未知的点, 这就是数值插值。与数值插值类似, 曲线拟合就是用一个较简单的函数去逼近一个复杂的或未知的函数, 所依

据的条件也是在一个区间上的有限个采样点的函数值，用这些有限的点构造逼近函数。数学上，常常把最佳拟合解释为数据点的误差平方和最小，且所用的曲线限定为多项式，就称为多项式的最小二乘曲线拟合。

插值和拟合都是根据已知数据来构造未知数据，两种方法的最大区别之处在于，曲线拟合要找出一个曲线方程式，而插值只要求得到内插数值。MATLAB 的数值插值与曲线拟合函数如下：

● 一维插值

$$Y_1 = \text{interp1}(X, Y, X_1, \text{'参数'})$$

说明：X 是采样点，Y 是样本值，X<sub>1</sub> 可是向量或标量，描述欲插值的点，Y<sub>1</sub> 是与 X<sub>1</sub> 等长的插值结果。

● 二维插值

$$Z_1 = \text{interp2}(X, Y, Z, X_1, Y_1, \text{'参数'})$$

说明：X、Y 是描述两个参数的采样点，Z 是与参数采样点对应的样本值，X<sub>1</sub>、Y<sub>1</sub> 是两个向量或标量，描述欲插值的点。二维插值是对双变量函数同时做插值。这在图像处理和数据可视化方面有很重要的应用。

插值函数参数如表 6-7 所示。

表 6-7 线性插值主要参数

参数名称	说明
nearest	最近点插值法。根据已知两点间的插值点与这两点之间的位置远近插值。当插值点距离前点近时，取前点的值，否则取后点的值
linear	线性插值。把相邻的数据点用直线连接，按所生成的曲线进行插值，是默认的插值方法
spline	三次样条插值。用已知数据求出样条函数后，按照样条函数插值
cubix	三次多项式插值。用已知数据构造出三次多项式进行插值
bicubic:	双立方插值法。利用已知的数据点拟合一个双立方曲面，然后根据插值点的坐标插值，每个插值点的值由该点附近的六个点的坐标确定，插值点处的值和该点值的导数都连续（二维插值独有）

注意：

两种插值方法都要求 x 必须为单调的升序或者降序排列。插值点的取值范围不能超过函数自变量的范围，否则，会给出“Nan”（非数）的错误。

● 曲线拟合

$$[P, S] = \text{polyfit}(X, Y, N)$$

说明：X、Y 是两个等长的向量，X 是采样点，Y 是采样点函数值，P 是一个长度为 N+1 的向量，代表 N 次多项式，S 是采样点的误差向量。该函数利用向量 X、Y 所确定的原始数据构造 N 阶多项式 P，使 P 与已知数据点间的函数值之差的平方和最小。当 N=1 时，多项式拟合就是线性拟合。一般而言，N 个数据点可以确定一个 N-1 阶多项式，便可做 N-1 阶拟合。

例 6-11 用不同的插值方法计算  $\sin(x)$  在  $x = \pi/4$  时的值。

```
>> X=0:0.1:pi/2;
>> Y=sin(X);
>> interp1(X,Y,pi/4)           %用默认方法（即线性插值）计算
ans =
    0.7067
>> interp1(X,Y,pi/4,'nearest') %用最近方法计算
ans =
    0.7174
>> interp1(X,Y,pi/4,'spline')  %用三次样条方法计算
ans =
    0.7071
>> interp1(X,Y,pi/4,'cubic')   %用三次多项式方法计算
ans =
    0.7071
```

例 6-12 用一个 4 次多项式在区间  $[0, 2\pi]$  内拟合函数  $\cos(x)$  并绘曲线（如图 6-1 所示）。

```
>> X=0:0.1:2*pi;
>> Y=cos(X);
>> [p,s]=polyfit(X,Y,4)
p =
   -0.0261    0.3275   -1.0894    0.3861    0.9477
s =
R: [5x5 double]
df: 58
normr: 0.1479
>> plot(X,Y,'k*',X,polyval(p,X),'r-') %实线是拟合后的函数，*是原函数
```

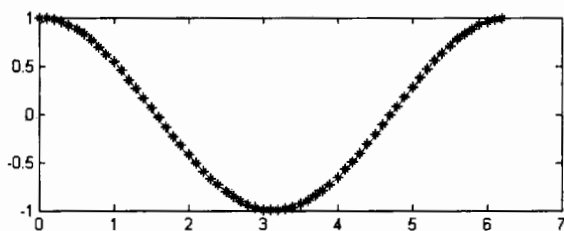


图 6-1 原函数和拟合后的函数曲线

### 6.3.3 函数的极值和零点

求函数在给定区间的最大值、最小值和零点是常见的运算，MATLAB 只提供了用于求极小值和零点的函数，但可通过求其倒数函数的极小值来得到函数的极大值。函数如表 6-8 所示。

表 6-8 函数的极值和零点的主要函数

函数名称	函数格式	说明
单个变量 函数极小值	$x = \text{fminbnd}('fun', a, b)$	fun 为待求极值的单变量函数, $a$ 、 $b$ 为求极值的区间。 $x$ 为函数极值点, $y$ 为极值点的函数值
	$[x, y] = \text{fminbnd}('fun', a, b)$	
多个变量 函数极小值	$x = \text{fminsearch}('fun', a)$	fun 为待求极值的多变量函数, $a$ 为极值点附近的初始值, $x$ 为函数极值点, $y$ 为极值点的函数值
	$[x, y] = \text{fminsearch}('fun', a)$	
函数零点	$x = \text{fzero}('fun', a)$	$a$ 为极值点附近的初始值, $[a, b]$ 为求零点的区间, $x$ 为函数零点, $y$ 为零点的函数值。若没有零点, 则返回 Nan (非数)
	$x = \text{fzero}('fun', [a, b])$	
	$[x, y] = \text{fzero}('fun', a)$	
	$[x, y] = \text{fzero}('fun', [a, b])$	

例 6-13 求函数  $f(x) = x^3 - 2x + 1$  在  $[-1, 1]$  之间的极小值和  $-1$  附近的零点。

```
>> x=-2:0.1:2; %定义自变量的范围
>> [x,y]=fminbnd('x.^3-2.*x+1',-1,1) %求极小值
x =
    0.8165
y =
   -0.0887
>> [x,y]=fzero('x.^3-2.*x+1',-1)
x =
   -1.6180
y =
    0
```

提示:

若要求单变量函数的极大值, 仍可以使用 fminbnd 函数, 只是要把原函数变成其倒数函数即可。

## 6.4 常微分方程的数值求解

### 6.4.1 常微分方程的解法

以一阶常微分方程为例讲解常微分方程的基本概念。一阶常微分方程的初值问题的一般形式是:

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases} \quad (6-1)$$

多数情况下, 一阶常微分方程初值问题的解是存在且唯一的, 但存在解并不意味着可以用有限形式表示其解。事实上, 在大多数情况下, 一阶常微分方程初值问题的解是不能用有限形式表示的。从高等数学中的知识可以知道, 常微分方程的解是一个函数, 既然不能找到

解的解析表达式, 而只有求解函数在指定点的函数值, 这就导致了对微分方程数值解的研究。数值解法的基本思想是: 先取一系列离散的点  $x_0 < x_1 < x_2 \dots < x_n < \dots$ , 通常取等步长  $h$ , 使  $x_n = x_0 + nh$ , 再求每个点对应的  $y(x_n)$ , 用  $y(x_n)$  近似  $y_n$  ( $n=1, 2, \dots$ ) 的值。主要有欧拉法、线性多步法、预估校正法、龙格—库塔法等, 其中以龙格—库塔法使用最多, 这里只对该方法进行介绍。

在求解式 6.1 中的未知数  $y$  时,  $y$  在  $x_0$  点的值  $y(x_0)=y_0$  是已知的, 根据高等数学中的中值定理, 应有:

$$\begin{cases} y(x_0 + h) = y_1 \approx y_0 + hf(x_0, y_0) \\ y(x_0 + 2h) = y_2 \approx y_1 + hf(x_1, y_1) \end{cases} \quad (6-2)$$

式中:  $h>0$ , 称为步长。

将式 6.2 推广, 在任意点  $x_i = x_0 + ih$ , 有:

$$y(x_0 + ih) = y_i \approx y_{i-1} + hf(x_{i-1}, y_{i-1}), \quad i=1, 2, \dots, n \quad (6-3)$$

当  $x_0$ 、 $y_0$  确定后, 根据式 6.3 能计算出未知函数  $y$  在点  $x_i = x_0 + ih$ ,  $i=0, 1, \dots, n$  的一系列数值解:

$$y_i = y_0, y_1, y_2, \dots, y_n, \quad i=0, 1, \dots, n \quad (6-4)$$

显然, 上述递推过程中有一个误差累计的问题, 在实际计算过程中, 使用的递推公式都进行了改造, 通常应用的龙格—库塔公式是:

$$y(x_0 + ih) = y_i = y_{i-1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (6-5)$$

其中:

$$\begin{aligned} k_1 &= f(x_{i-1}, y_{i-1}) \\ k_2 &= f\left(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}hk_2\right) \\ k_4 &= f(x_{i-1} + h, y_{i-1} + hk_3) \end{aligned}$$

## 6.4.2 龙格—库塔法的实现

基于龙格-库塔法, MATLAB 提供了求常微分方程数值解的函数, 其函数格式如下:

$$\begin{aligned} [X, Y] &= \text{ode23}('f', [x_0, x_n], y_0) \\ [X, Y] &= \text{ode45}('f', [x_0, x_n], y_0) \end{aligned}$$

说明:  $X$ 、 $Y$  是两个向量。 $X$  对应自变量  $x$  在求解区间  $[x_0, x_n]$  的一组采样点, 其采样密度是自适应的, 无需指定;  $Y$  是与  $X$  对应的一组解。 $f$  是一个 M 函数文件, 代表待求解方程。 $[x_0, x_n]$  代表自变量的求解区间。 $y_0=y(x_0)$ , 由方程的初值给定。函数在求解区间  $[x_0, x_n]$  内, 自动设立采样点向量  $X$ , 并求出解函数  $y$  在采样点  $X$  处的样本值。

ode23 函数采用了二阶、三阶龙格—库塔法, ode45 函数采用了四阶、五阶龙格—库塔法, 这两个函数都采用自适应变步长的求解方法, 即当解的变化较慢时采用较大的步长, 从

而提高了计算速度；当解的变化较快时步长会自动地变小，可以提高计算精确度。

**例 6-14** 求微分方程  $\begin{cases} \frac{dy}{dx} = -3y + 2x \\ y(1) = 2 \end{cases}$  初值问题在  $[1, 3]$  区间内的数值解。

先建立一个该方程的函数文件，启动 MATLAB 文本编辑器，输入以下命令：

```
function f=f(x,y)
f=-3.*y+2*x;           %使用点运算
```

按默认文件名存盘后，在命令窗口输入命令：

```
>> [X,Y]=ode45('f',[1 3],1);           %采用四阶、五阶龙格—库塔法
>> X'                                     %转置后，显示自变量的一组采样点
ans =
Columns 1 through 7
    1.0000    1.0500    1.1000    1.1500    1.2000    1.2500    1.3000
Columns 8 through 14
    1.3500    1.4000    1.4500    1.5000    1.5500    1.6000    1.6500
Columns 15 through 21
    1.7000    1.7500    1.8000    1.8500    1.9000    1.9500    2.0000
Columns 22 through 28
    2.0500    2.1000    2.1500    2.2000    2.2500    2.3000    2.3500
Columns 29 through 35
    2.4000    2.4500    2.5000    2.5500    2.6000    2.6500    2.7000
Columns 36 through 41
    2.7500    2.8000    2.8500    2.9000    2.9500    3.0000
>> Y'                                     %转置后，显示与采样点对应的一组数值解
ans =
Columns 1 through 7
    1.0000    0.9559    0.9226    0.8987    0.8827    0.8735    0.8703
Columns 8 through 14
    0.8722    0.8785    0.8885    0.9017    0.9178    0.9363    0.9568
Columns 15 through 21
    0.9791    1.0030    1.0282    1.0545    1.0818    1.1099    1.1388
Columns 22 through 28
    1.1683    1.1983    1.2287    1.2596    1.2908    1.3224    1.3541
Columns 29 through 35
    1.3861    1.4183    1.4506    1.4831    1.5157    1.5484    1.5812
Columns 36 through 41
    1.6140    1.6470    1.6799    1.7130    1.7460    1.7792
```

## 6.5 上机实践

### 6.5.1 跟我学

**例 6-15** 求矩阵  $A = [-2, 0, -11; -4, 13, 2; 3, -2, 2]$  的迹、范数、条件数、秩，

左右翻转后提取下三角矩阵。

```
>> A=[-2,0,-11;-4,13,2;3,-2,2];
>> traceA=trace(A)           %求矩阵的迹
    traceA =
         13
>> normA=norm(A)             %求范数
    normA =
    14.0201
>> condA=cond(A)             %求条件数
    condA =
     8.0179
>> ranfA=rank(A)             %求矩阵的秩
    ranfA =
         3
>> tril(fliplr(A))           %左右翻转后提取下三角矩阵
    ans =
     -11     0     0
       2    13     0
       2    -2     3
```

**例 6-16** 建立一个  $3 \times 5$  阶随机矩阵，求矩阵的最大值、最小值、方差和标准差。

```
>> A=rand(3,5)               %建立  $3 \times 5$  阶随机矩阵
    A =
    0.9501    0.4860    0.4565    0.4447    0.9218
    0.2311    0.8913    0.0185    0.6154    0.7382
    0.6068    0.7621    0.8214    0.7919    0.1763
>> ma=max(max(A))            %求矩阵的最大元素
    ma =
    0.9501
>> na=min(min(A))            %求矩阵的最小元素
    na =
    0.0185
>> B=var(A)                   %求方差
    B =
    0.1293    0.0429    0.1616    0.0301    0.1509
>> C=std(A)                   %求标准差
    C =
    0.3596    0.2070    0.4020    0.1736    0.3884
```

**例 6-17** 在 MATLAB 中建立多项式  $f(x) = x^3 + 2x - 5$ ，并求出  $f(x) = 0$  时的根及与多项式  $f(x) = x^2 - 1$  的乘积。

```
>> f=[1,0,2,-5];            %创建多项式
>> g=[1,0,-1];               %短多项式补零
>> g1=[0,1,0,-1];            %多项式求根
>> x=roots(f)
```



```

x =
    -0.6641 + 1.8230i
    -0.6641 - 1.8230i
    1.3283
>> conv(f,g)          %多项式相乘
ans =
     1     0     1    -5    -2     5

```

**例 6-18** 求函数  $f(x) = x^3 + x^2 - 3x + 3$  在  $[-2, 2]$  之间的极小值和  $-2$  附近的零点。

```

>> x=-2:0.1:2;          %定义自变量的范围
>> [x,y]=fminbnd('x.^3+x.^2-3.*x+3',-2,2) %求极小值
x =
    0.7208
y =
    1.7316
>> [x,y]=fzero('x.^3+x.^2-3.*x+3',-2) %求-2附近的零点
x =
   -2.5987
y =
  -3.5527e-015

```

**例 6-19** 求微分方程  $\begin{cases} \frac{dy}{dx} - \frac{3x}{y} = -2x \\ y(1) = 1 \end{cases}$  初值问题在  $[1, 2]$  区间内的数值解。

先建立一个该方程的函数文件，启动 MATLAB 文本编辑器，输入以下命令：

```

function f=f(x,y)
f=3.*x./y-2*x; %使用点运算

```

按默认文件名存盘后，在命令窗口输入命令：

```

>> [X,Y]=ode45('f',[1 2],1); %采用四阶、五阶龙格-库塔法
>> X' %转置后，显示自变量的一组采样点
ans =
Columns 1 through 7
    1.0000    1.0250    1.0500    1.0750    1.1000    1.1250    1.1500
Columns 8 through 14
    1.1750    1.2000    1.2250    1.2500    1.2750    1.3000    1.3250
Columns 15 through 21
    1.3500    1.3750    1.4000    1.4250    1.4500    1.4750    1.5000
Columns 22 through 28
    1.5250    1.5500    1.5750    1.6000    1.6250    1.6500    1.6750
Columns 29 through 35
    1.7000    1.7250    1.7500    1.7750    1.8000    1.8250    1.8500
Columns 36 through 41
    1.8750    1.9000    1.9250    1.9500    1.9750    2.0000
>> Y' %转置后，显示一组数值解

```

ans =

Columns 1 through 7

1.0000 1.0244 1.0476 1.0698 1.0909 1.1110 1.1303

Columns 8 through 14

1.1487 1.1662 1.1830 1.1991 1.2144 1.2291 1.2431

Columns 15 through 21

1.2565 1.2692 1.2815 1.2931 1.3043 1.3149 1.3250

Columns 22 through 28

1.3347 1.3439 1.3527 1.3611 1.3690 1.3766 1.3838

Columns 29 through 35

1.3907 1.3972 1.4034 1.4093 1.4148 1.4201 1.4251

Columns 36 through 41

1.4298 1.4343 1.4385 1.4425 1.4463 1.4499

## 6.5.2 自己练

1. 求下列矩阵的迹、秩、特征根和特征向量。

$$(1) A = \begin{bmatrix} 1 & -1 & 2 & 3 \\ 1 & 5 & -4 & 1 \\ 2 & 3 & 0 & 7 \\ 14 & 7 & -2 & 11 \end{bmatrix} \quad (2) A = \begin{bmatrix} 2 & -2 & 3 & 12 \\ 6 & 8 & 0 & 4 \\ 3 & 0 & 1 & 2 \\ -1 & 2 & 1 & 13 \end{bmatrix}$$

2. 用数值方法解线性方程组。

$$\begin{cases} 6x + 5y - 2z + 5u = -4 \\ 9x - y + 4z - u = 13 \\ 3x + 4y + 2z - 2u = 1 \\ 3x - 9y + 2u = 11 \end{cases}$$

3. 使用函数，实现左旋  $90^\circ$  或右旋  $90^\circ$  的功能。例如，原矩阵为  $A$ ， $A$  左旋后得到矩阵  $B$ ，右旋后得到矩阵  $C$ 。

$$A = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 11 & 12 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \quad C = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \\ 12 & 11 & 10 \end{bmatrix}$$

4. 将 10 个学生 5 门功课的成绩存入矩阵  $P$  中，进行如下处理。
  - (1) 分别求每门课的最高分、最低分及相应的学生序号。
  - (2) 分别求每门课的平均分和标准方差。
  - (3) 5 门课总分的最高分、最低分及相应学生序号。
  - (4) 将 5 门课总分按从大到小顺序存入  $zcx$  中，相应学生序号存入  $xzcx$  中。
5. 有 3 个多项式  $P_1(x)=x^4+2x^3+4x^2+5$ ,  $P_2(x)=x+2$ ,  $P_3(x)=x^2+2x+3$ ，试进行下列操作。

- (1) 求  $P(x)=P_1(x)+P_2(x)*P_3(x)$ 。  
 (2) 求  $P(x)$  的根。  
 (3) 当  $x$  取矩阵  $A$  的每一元素时, 求  $P(x)$  的值。其中

$$A = \begin{bmatrix} -1 & 1.2 & -1.4 \\ 0.75 & 2 & 3.5 \\ 0 & 5 & 2.5 \end{bmatrix}$$

- (4) 当以矩阵  $A$  为自变量时, 求  $P(x)$  的值。其中  $A$  的值与 (3) 相同。  
 6. 求微分方程的数值解。

$$\begin{cases} \frac{d^2 y}{dx^2} - 5 \frac{dy}{dx} + y = 0 \\ y(0) = 0 \\ y'(0) = 0 \end{cases}$$

7. 求代数方程的数值解。  
 (1)  $3x + \sin x - e^x = 0$  在  $x_0 = 1.5$  附近的根。  
 (2) 在给定的初值  $x_0 = 1, y_0 = 1, z_0 = 1$  下, 求下列方程组的数值解。

$$\begin{cases} \sin x + y^2 + \ln z - 7 = 0 \\ 3x + 2y - z^3 + 1 = 0 \\ x + y + z - 5 = 0 \end{cases}$$

8. 求下列函数在指定区间的最大值。  
 (1)  $f(x) = \frac{1+x^2}{1+x^4}, x \in (0, 2)$  (2)  $f(x) = \sin x + \cos x^2, x \in (0, \pi)$

9. 求下列多项式的根与导数。  
 (1)  $f(x) = x^3 - 2x - 5$  (2)  $f(x) = x^4 + 5x^3 + 5x^2 - 5x - 6$

10. 已知正弦函数如下表所示。

$x$	21°	22°	23°	24°
$y$	0.355 37	0.374 61	0.390 73	0.406 74

求  $\sin 21^\circ 30'$  的近似值。

11. 化工生产中, 常常需要知道丙烷在各种温度  $T$  和压力  $P$  下的导热系数  $K$ 。下面是实验得到的一组数据:

$T/^\circ\text{C}$	$P/(\text{MN}/\text{m}^2)$	$K$	$T/^\circ\text{C}$	$P/(\text{MN}/\text{m}^2)$	$K$
68	0.7981	0.0848	106	9.7908	0.0695
69	3.324	0.0895	106	14.757	0.0770
87	9.0078	0.0761	140	9.6536	0.0625
87	13.355	0.0810	140	12.364	0.0652

计算在  $T=95^\circ\text{C}$  和  $P=10.2\text{MN}/\text{m}^2$  下的导热系数  $K$  值。

## 6.6 本章小结

数值计算在科学研究与工程应用中有着广泛的应用。许多数值计算问题，用其他程序设计语言编程求解非常麻烦，并且需要具备专门的数学知识及一定的程序设计技能，而用 MATLAB 编程，往往只要很少几个语句即可完成求解任务，而且它编程效率高，对程序员的数学背景知识要求低。MATLAB 这种强大的数值计算能力，使其在科学计算方面成为了用户的首选工具。本章首先介绍了线性代数中的数值计算问题，包括特殊矩阵的建立、矩阵分析、线性方程组求解等内容，然后介绍数据处理和多项式计算，包括数据的分析与统计、数值插值、曲线拟和及多项式计算等内容，最后介绍常微分方程数值求解以及应用龙格-库塔方法求解常微分方程等内容。本章涉及到的函数列表 6-9 如下。

表 6-9 本章主要函数一览

函 数	功 能	函 数	功 能
det(A)	矩阵 A 的行列式	trace(A)	矩阵 A 的迹
norm(A)	矩阵 A 的范数	rank(A)	矩阵 A 的秩
cond(A)	矩阵 A 的条件数	eig(A)	矩阵 A 的全部特征值
diag(A)	提取矩阵 A 的对角元素	triu(A)	提取矩阵 A 的上三角矩阵
fliplr(A)	矩阵 A 左右翻转	tril(A)	提取矩阵 A 下三角矩阵
magic	建立魔方矩阵	flipud(A)	矩阵 A 上下翻转
pascal	建立 pascal 矩阵	full	把稀疏矩阵转化成普通矩阵
chol	Choleky 分解	sparse	创建稀疏矩阵
lu	LU 矩阵分解	qr	正交三角阵分解
max(x)	找 x 各列的最大元素	prod(x)	求 x 各列元素之积
mean(x)	求 x 各列的平均值	sum(x)	求 x 各列元素之和
median(x)	找 x 各列的中位元素	sort(x)	使 x 的各列元素按递增排序
min(x)	找 x 各列的最小元素	var(x)	求 x 各列的方差
std(x)	求 x 各列的标准差	cov(x, y)	求两个矩阵 x 和 y 的协方差
range(x)	求 x 各列的极差	corrcoef(x)	求 x 的自相关阵
cov(x)	求 x 的协方差阵	corrcoef(x, y)	求两个矩阵 x 和 y 互相关系数
poly	指定根的多项式	roots	求多项式的根
polyval	多项式计算	conv	卷积或多项式乘法
polyder	多项式微分	interp1	一维插值
interp2	二维插值	polyfit	多项式曲线拟合
fminbnd	求单变量函数极小值	fminsearch	求多变量极小值
fzero	求单变量函数的零点	ode23	2 阶、3 阶龙格-库塔法
ode45	4 阶、5 阶龙格-库塔法		

## 6.7 习题

1. 求下列矩阵的主对角元素，上三角阵，下三角阵，行列式的值、秩、范数、条件数、迹、特征值和特征向量。

$$(1) A = \begin{bmatrix} 4 & 6 & 0 \\ -3 & -5 & 0 \\ -3 & -6 & 1 \end{bmatrix} \quad (2) B = \begin{bmatrix} 1 & 2 & 2 \\ 1 & -1 & 1 \\ 4 & -12 & 1 \end{bmatrix}$$

2. 用 MATLAB 提供的 randn 函数生成符合正态分布的  $10 \times 5$  随机矩阵 A, 进行如下操作。

- (1) A 矩阵各列元素的均值和标准方差。
- (2) A 矩阵的最大元素和最小元素。
- (3) 分别对 A 矩阵的每列元素按升序、每行元素按降序排序。

3. 已知多项式  $P_1(x)=3x+2$ ,  $P_2(x)=5x^2-x+2$ ,  $P_3(x)=x^2-0.5$ 。

- (1) 求  $P(x)=P_1(x)P_2(x)P_3(x)$ 。
- (2) 求  $P(x)=0$  的全部根。
- (3) 计算  $x_i=0.2i$ ,  $i=0, 1, 2, \dots, 10$  各点上的  $P(x_i)$ 。

4. 求下列多项式的根与导数。

$$(1) f(x)=x^3-2x^2-5 \quad (2) f(x)=x^3+2x^2+10x-20$$

5. 求下列方程在区间  $[0, 1]$  的一个解。

$$(1) x-2^{-x}=0 \quad (2) e^x-x^2+3x=0$$

6. 求函数  $f(x,y)=(x+8y^2)^2+5(x^2+x)^2$  的极小值。

7. 求下列微分方程的数值解。

$$(1) \begin{cases} \frac{dy}{dx} - 2x = \frac{2x}{y} \\ y(1) = 0 \end{cases} \quad (2) \begin{cases} \frac{x^2 d^2 y}{dx^2} - 5 \frac{dy}{dx} + y = 0 \\ y(0) = 0 \\ y'(0) = 0 \end{cases}$$

8. 随机地从一批铁钉中抽取 16 枚, 测得其长度 (单位为 cm) 为:

2.14   2.10   2.13   2.15   2.13   2.12   2.13   2.10  
2.15   2.12   2.14   2.10   2.11   2.13   2.14   2.10

试求样本均值、样本方差、样本标准差、样本中位数和极差。

---

## 第7章 Simulink 仿真

通过对本章的学习，读者可以充分了解 Simulink 的基本模块，能够根据实际系统的特性合理选择模块和设置参数，进行系统仿真，并对仿真结果进行分析。本章的重点是 7.3 Simulink 模块的操作，难点是 7.4 仿真模型的参数设置。

### 7.1 认识 Simulink

#### 7.1.1 Simulink 简介

Simulink 是 MATLAB 环境下对动态系统进行建模、仿真和分析的一个软件包。顾名思义，Simulink 的名字表明了该系统的两个主要功能：Simu（仿真）和 Link（连接）。Simulink 提供了图形化的用户界面，用户只需点击鼠标，调用现成的图形模块，并将它们适当地连接起来以构成动态系统模型，然后对系统进行仿真，并可以随时观察仿真结果和干预仿真过程。这样就大大降低了仿真的难度，用户不需要为了完成仿真工作而去学习某种程序设计语言。


Simulink 提供了大量的系统模块，包括信号、运算、显示和系统等多方面的功能，可以创建各种类型的仿真系统，实现丰富的仿真功能。用户也可以定义自己的模块，进一步扩展模型的范围和功能，以满足不同的需求。为了创建大型系统，Simulink 提供了系统分层排列的功能，类似于系统的设计，在 Simulink 中可以将系统分为从高级到低级的几个层次，每层又可以细分为几个部分，每层系统构建完成后，将各层连接起来构成一个完整的系统。模型创建完成之后，可以启动系统的仿真功能分析系统的动态特性，Simulink 内置的分析工具包括各种仿真算法、系统线性化、寻求平衡点等，仿真结果可以以图形的方式显示在示波器窗口，以便于用户观察系统的输出结果；Simulink 也可以将输出结果以变量的形式保存起来，并输入到 MATLAB 工作空间中以完成进一步的分析。

Simulink 可以支持多采样频率系统，即不同的系统能够以不同的采样频率进行组合，可以仿真较大、较复杂的系统。

Simulink 软件包是 20 世纪 90 年代初由 MathWorks 公司开发。现在较为流行的版本有：与 MATLAB5.2 配用的 Simulink2.2、与 MATLAB5.3 配用的 Simulink3.0 和与 MATLAB6.0 配用的 Simulink4.0。

#### 7.1.2 Simulink 的启动和退出

##### 1. Simulink 的启动

在 MATLAB 命令窗口下，点击工具栏中的图标或键入命令“Simulink”，就会弹出一个名为“Simulink Library Browser”的浏览器窗口（如图 7-1 所示），该窗口以树状列表的形式列出了当前 MATLAB 系统中已安装了的所有 Simulink 模块，这些模块包含 Simulink 模块库中的各种模块及其他 Toolbox（工具箱）和 Blockset（模块组件）中的模块。展开树状列

表，用鼠标单击所需的模块项，列表窗口的上方会显示所选模块的信息。也可以在浏览器窗口右上角的输入栏中直接键入模块名并单击“Find”按钮进行查询。如果输入一个不存在的模块，将弹出提示对话框。

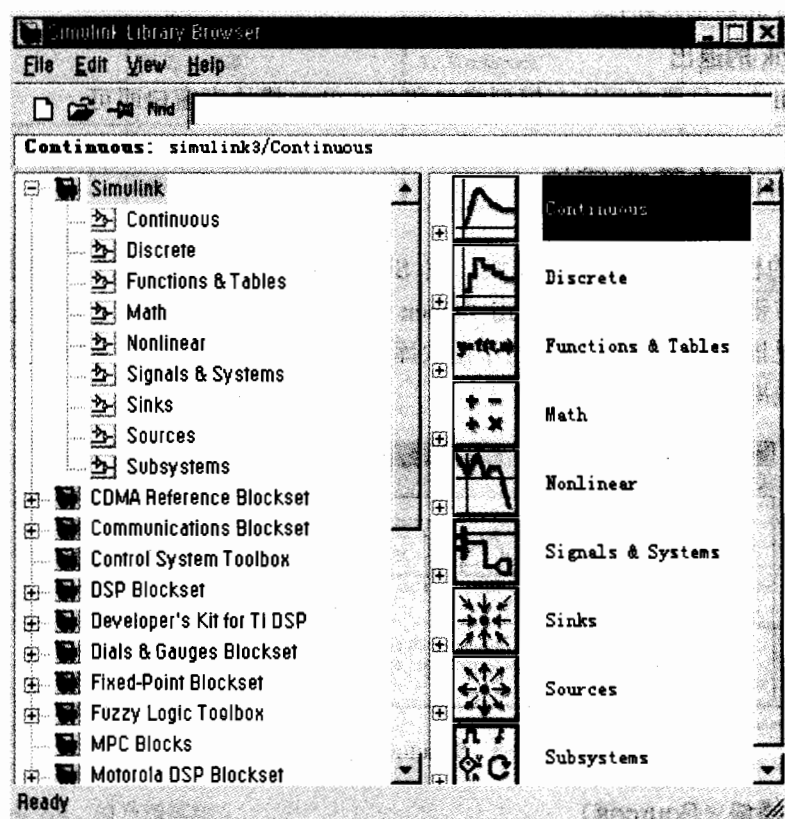



图 7-1 Simulink 模块库浏览器

在创建新模型时，先在“Simulink Library Browser”浏览器上方的工具栏中选择“建立新模型”的图标，或者在 MATLAB 命令窗口“File”菜单中选择“New”菜单项下的“Model”命令，则会弹出一个名为“Untitled”（无标题）的空白窗口，所有控制模块都创建在这个窗口中，如图 7-2 所示。

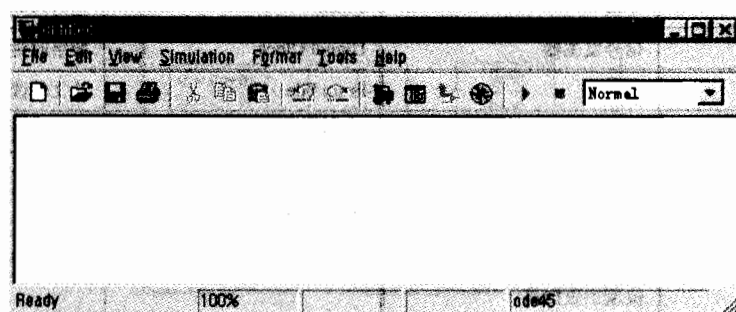



图 7-2 建立的模型窗口

如果要对一个已经存在的模块文件进行编辑修改，需要打开该模型文件，可以在 MATLAB 命令窗口直接输入该模型文件名（不要加文件扩展名.mdl）；也可以在模型窗口的“File”菜单中选择“Open”命令，然后选择或输入欲编辑模型的名字；也可以单击模型窗口工具栏上的打开命令按钮.

### 2. Simulink 的退出

退出 Simulink，只要关闭所有模型窗口和 Simulink 模块库窗口即可。

## 7.2 Simulink 的基本模块

Simulink 的模块库提供了大量模块。在 Simulink 的模块库浏览器窗口左侧的 Simulink 项上单击鼠标右键，在弹出的菜单单击“Open the ‘Simulink’ Labray”选项，将打开 Simulink 模块库窗口（见图 7-3），单击其中的子模块库图标，打开子模块库，找到仿真所需的模块。下面对各子模块库作介绍。

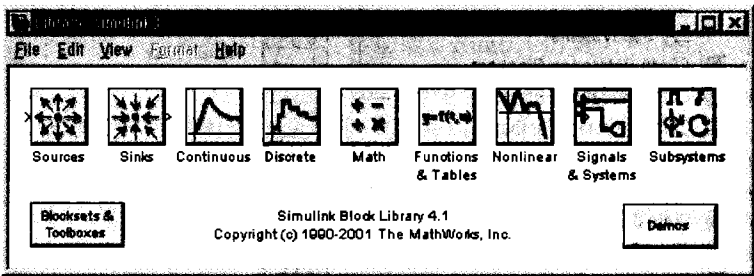


图 7-3 Simulink 模块库窗口

### 1. 信号源模块（Sources）

双击 Simulink 模块库窗口中的“Sources”模块，即可打开信号源模块（如图 7-4 所示）。信号源模块中的各子模块的功能如表 7-1 所示。

表 7-1 信号源模块

模 块	功 能	模 块	功 能
In1	创建输入端	Ground	接地
Constant	常数	Clock	当时时间
Signal Generator	信号发生器	Digital Clock	数字时钟
Ramp	斜坡	From File	从文件读数据
Sine Wave	正弦波	From Workspace	从工作空间读数据
Step	阶跃信号	Random Number	随机信号
Repeating Sequence	重复序列	Uniform Random Number	均匀随机信号
Pulse Generator	脉冲发生器	Band-Limited White Noise	带限白噪声
Chirp Signal	快速正弦扫描		

### 2. 输出模块（Sinks）

输出模块也可称为接收模块，用于显示仿真结果或输出仿真数据。输出模块及其功能简



介如图 7-5 和表 7-2 所示。

表 7-2 输出模块的功能

模 块	功 能	模 块	功 能
Scope	示波器	To File	输出到文件
Floating Scope	可选示波器	To Workspace	输出到工作空间
XY Graph	XY 关系图	Terminator	通用终端
Outl	创建输出端	Stop Simulation	输出不为 0 时停止仿真
Display	实时数值显示		

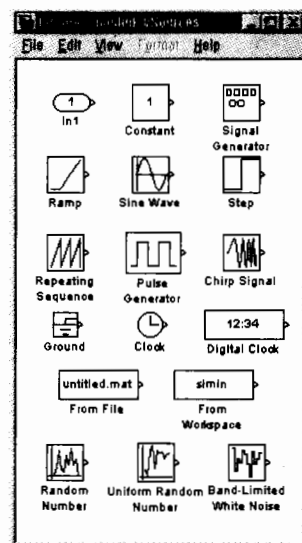


图 7-4 信号源模块

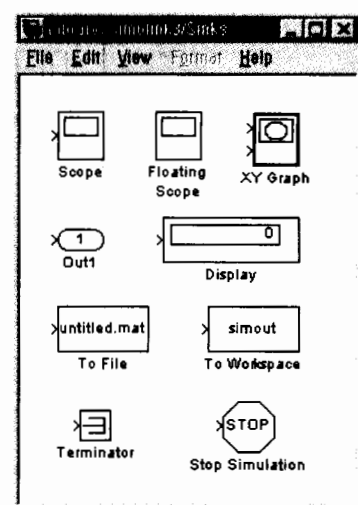


图 7-5 输出模块

### 3. 连续系统模块 (Continuous)

连续系统模块及其功能简介如图 7-6 和表 7-3 所示。

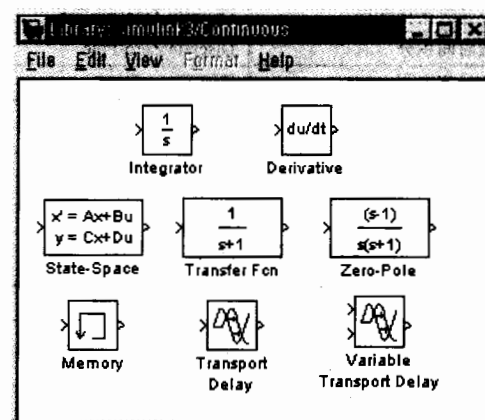


图 7-6 连续系统模块

表 7-3 连续系统模块的功能

模 块	功 能	模 块	功 能
Integrator	积分	Zero-Pole	零极点
Derivative	微分	Memory	延时输出
State-Space	状态方程	Transport Delay	传输延时
Transfer Fcn	传递函数	Variable Transport Delay	可变传输延时

#### 4. 离散系统模块 (Discrete)

离散系统模块及其功能简介如图 7-7 和表 7-4 所示。

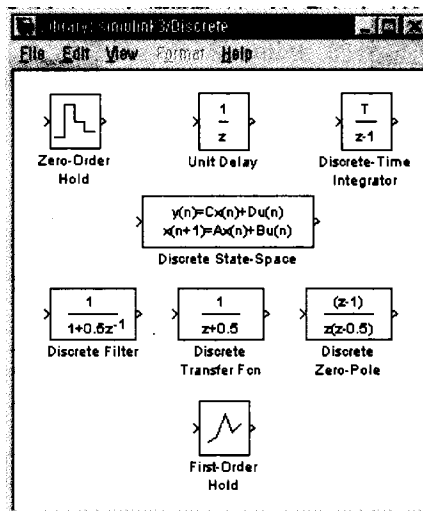


图 7-7 离散系统模块

表 7-4 离散系统模块的功能

模 块	功 能	模 块	功 能
Zero-Order Hold	零阶保持器	Discrete Filter	离散滤波器
Unit Delay	单位延时采样保持	Discrete Transfer Fcn	离散传递函数
Discrete-Time Integrator	离散时间积分	Discrete Zero-Pole	离散零极点
Discrete State-Space	离散状态方程	First-Order Hold	一阶保持器

#### 5. 函数和表模块 (Functions & Tables)

函数和表模块及其功能简介如图 7-8 和表 7-5 所示。

表 7-5 函数和表模块的功能

模 块	功 能	模 块	功 能
Look-Up Table	线性插值查表	Fcn	C 语言形式的表达式
Look-Up Table(2-D)	二维线性插值	MATLAB Fun	MATLAB 形式的表达式
Look-Up Table(n-D)	N 维线性插值	S-Function	调用 S 函数
Prelook-Up Index Search	预查下标	Polynomial	多项式
Interpolation(n-D) Using Prelook-Up	N 维插值	S-Function Builder	用 C 语言代码创建 S 函数
Direct Look-Up Table(n-D)	直接查表		

## 6. 非线性系统模块 (Nonlinear)

非线性系统模块及其功能简介如图 7-9 和表 7-6 所示。

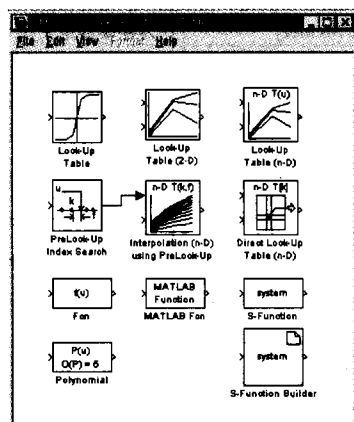


图 7-8 函数和表模块

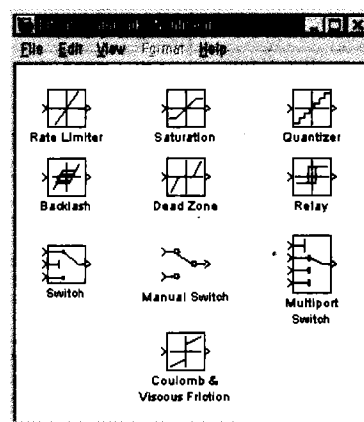


图 7-9 非线性系统模块

表 7-6 非线性系统模块的功能

模 块	功 能	模 块	功 能
Rate Limiter	速率限制器	Relay	继电器
Saturation	饱和元件	Switch	开关
Quantizer	量化元件	Manual Switch	手动开关
Backlash	间隙元件	Multiport Switch	多选开关
Dead Zone	死区元件	Coulomb & Viscous Friction	库仑和粘性摩擦

## 7. 信号与系统模块 (Signals & Systems)

信号与系统模块及其功能简介如图 7-10 和表 7-7 所示。

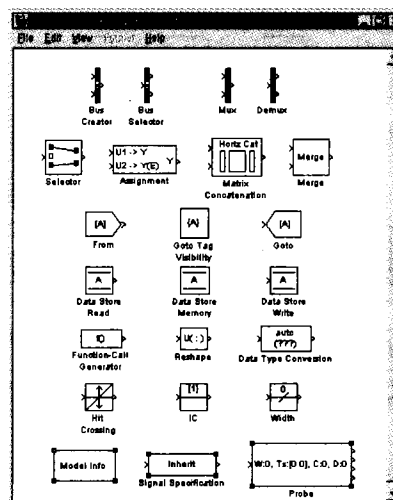


图 7-10 信号与系统模块

表 7-7 信号与系统模块的功能

模 块	功 能	模 块	功 能
Bus Creator	创建信号总线	Data Store Memory	为存储器定义尺寸
Bus Selector	从信号总线中选择信号	Data Store Write	向存储器写数据
Mux	多路传输器	Function-Call Generator	函数调用发生器
Demux	多路分离器	Reshape	改变信号大小
Selector	选择输入信号	Data Type Conversion	数据类型转换
Assignment	赋值	Hit Crossing	检测零交叉点
Matrix Concatenation	矩阵串联	IC	信号的初始值
Merge	信号合并	Width	信号的宽度
Form	从 Goto 模块接收信号	Model Info	显示模型信息
Goto Tag Visibility	定义 Goto 模块的范围	Signal Specification	检查信号参数
Goto	把信号送到 Form 模块	Probe	检查连线
Data Store Read	从存储器读数据		

## 8. 数学运算模块 (Math)

数学运算模块提供了基本数学运算函数、三角函数、复数运算函数以及矩阵运算函数。数学模块及其功能简介如图 7-11 和表 7-8 所示。

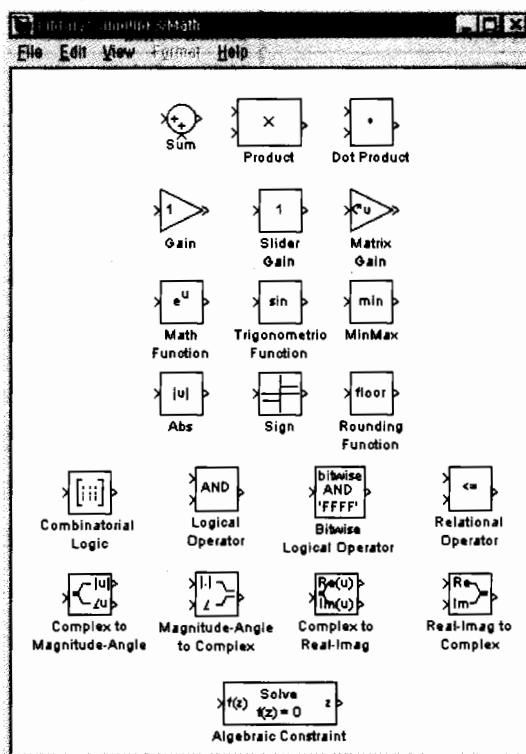


图 7-11 数学运算模块

表 7-8 数学运算模块的功能

模 块	功 能	模 块	功 能
Sum	求和	Rounding Function	取整函数
Product	积或商	Combinatorial Logic	逻辑真值表
Dot Product	点积	Logic Operator	逻辑算子
Gain	常数增益	Bitwise Logical Operator	位逻辑算子
Slider Gain	可变增益	Relational Operator	关系算子
Matrix Gain	矩阵增益	Complex to Magnitude-Angle	复数的模和辐角
Math Function	数学运算函数	Magnitude-Angle to Complex	模和辐角合成函数
Trigonometric function	三角函数	Complex to Real-Imag	复数的实部和虚部
MinMax	求最大值	Real-Imag to Complex	实部和虚部合成复数
Abs	求绝对值	Algebraic Constraint	强迫输入信号为零
Sign	符号函数		

## 7.3 Simulink 模块的操作

### 7.3.1 添加和选取模块

#### 1. 添加模块

当要把一个模块添加到模型中时，首先在 Simulink 模块库中找到它，然后在模块库中用鼠标单击该模块，不要放开鼠标，将这个模块拖入模型窗口中即可。

#### 2. 选取模块

当模块已经位于模型窗口中时，只要用鼠标在模块上单击就可以选中该模块，这时模块的角上出现一些黑色的小方块，这些小方块就是该模块的关键点，拖动这些黑色小方块可以改变模块的大小。

如果要选中多个模块，可以在这些模块所占区域的一角按下鼠标左键不放，拖向该区域的对角，在此过程中会出现虚框，当虚框包住了要选中的所有模块后，放开鼠标左键，这时在所有被选模块的角上以及连线上都会出现小黑方块，表示模块都被选中了。

### 7.3.2 模块位置和外形调整

为了连线的方便和美观，经常需要调整模块的位置和外形。

#### 1. 调整模块位置

改变调整模块很简单，用鼠标左键点住模块，按住左键拖动它即可。在拖动过程中，会显示该模块的虚框，将模块放到适当位置，松开鼠标即可。

#### 2. 调整模块大小

若要改变单个模块的大小，首先选中该模块，会看到模块四角的关键点，然后用鼠标点住模块的关键点，按住拖动即可改变模块大小。在拖动过程中，可以看到模块外形的虚框，拖到合适大小松开鼠标即可。

若要改变整个模型所有模块的大小，可以打开模型窗口中的“View”菜单，看到菜单

的第三部分的 4 个命令：“Zoom in”、“Zoom out”、“Fit selection to view”、“Normal(100%)”。前两条命令分别用来将整个模型放大、缩小；第三条命令用来将当前选中的模块或当前系统放大到整个窗口大小来观察；最后一条命令用来将整个模型恢复到原始的正常大小。

### 3. 调整模块方向

有时需要改变模块的方向，首先选中这一个或多个模块，然后打开“Format”菜单，选择“Rotate block”使模块旋转 90°；选择“Flip block”使模块旋转 180°。也可以在模块上右击鼠标，在弹出菜单中选择“Format”子菜单中的相应选项。同时这两种操作有快捷键〈Ctrl〉+〈R〉和〈Ctrl〉+〈F〉。

### 4. 调整模块颜色和效果

模块的颜色也可以修改，选中某一模块，打开“Format”菜单，指向“Foreground color”，在弹出菜单中选择模块的背景色，即模块的图标、边框和模块名的颜色。指向“Background color”，在弹出菜单中选择模块的背景色，即模块的背景填充色。在“Format”中还有一项“Screen color”，用来改变模型的背景色。

最后还可以给模块加上阴影，产生立体效果。选中模块后，选择“Format”菜单中“Show drop shadow”的命令即可。

#### 提示：

在模块上右击鼠标，在弹出菜单中选择“Format”子菜单中的相应选项，也可以完成以上操作。

## 7.3.3 模块名的处理

### 1. 改变模块名

用鼠标左键单击模块名的区域，这时会在此处出现编辑状态的光标，在这种状态下能够对模块名进行修改。模块名和模块图标字体也可以更改，方法是选定模块，在“Format”菜单下选取“Font”命令，这时会弹出“Set Font”对话框，在对话框中选取字体的类型和大小。

### 2. 隐藏模块名

选定模块，在“Format”菜单下选取“Hide name”命令，模块名会被隐藏。若要把隐藏的模块名再显示出来，再次打开“Format”菜单，会发现原来的“Hide name”命令变成了“Show name”命令，单击它就可以显示模块名。

### 3. 改变模块名的位置

模块名的位置有一定的规律，模块的接口在左右两侧时，模块名只能位于模块的上下两侧，默认在下侧；当模块的接口在上下两侧时，模块名只能位于模块的左右两侧，默认在左侧。

因此模块名只能从原始位置移到相对的位置。可以用鼠标拖动模块名到其相对应的位置；也可以选定模块后，选择模型窗口菜单栏“Format”菜单下的“Flip name”命令来翻转模块名。

### 7.3.4 复制和删除模块

#### 1. 复制模块

在模型制作的过程中，可能某一个模块使用多次，总是从 Simulink 模块库中添加无疑是很麻烦的。其实只要添加一个，其余的用这个复制即可。

在同一个模型窗口中复制模块时，最简单的方法是，首先按下〈Ctrl〉键不放，用鼠标左键点住要复制的模块，按住左键拖动该模块，在拖动过程中，会显示该模块的虚框和一个加号，最后将模块放到适当的位置，松开鼠标和〈Ctrl〉键即可。

当然也可以使用“Edit”菜单下的“Copy”和“Paste”命令来进行复制和粘贴。在不同窗口之间也可以复制模块，最简单的方法是直接将模块拖到目标窗口即可（不用按〈Ctrl〉键）。也可以使用复制、粘贴命令。

#### 注意：

在进行粘贴之前，可以用鼠标在模型窗口中的适当位置单击一下，选择模块粘贴的位置，鼠标单击点是粘贴模块的左上角。

#### 2. 删除模块

选定模块，选择“Edit”菜单下的“Cut”命令，将模块剪切到剪贴板；选择“Clear”命令将彻底删除模块，也可以按〈Delete〉键进行彻底删除；右击模块，弹出菜单中也有相应命令。

### 7.3.5 模块属性和参数的设置

#### 1. 模块参数的设置

Simulink 中几乎所有模块的参数(Parameters)都允许用户进行设置，只要双击要设置的模块或在模块上单击鼠标右键并在弹出的快捷菜单中选择“Block parameters”后，就可以打开模块参数设置对话框。可以发现对话框基本分为两部分（如图 7-12 所示），上面一部分是模块功能的说明，下面一部分用来进行模块参数设置。同样在模型窗口的菜单栏上，选择“Edit”菜单下的“Block parameters”命令也可以打开模块参数设置对话框。

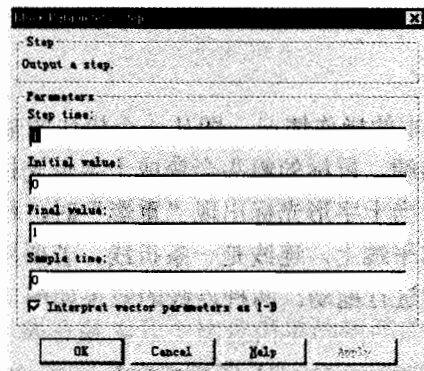


图 7-12 模块参数设置对话框

## 2. 模块属性的设置

与参数设置对话框不同，所有模块的属性（Properties）设置对话框都是一样的。选定要设置属性的模块，然后选择“Edit”菜单下的“Block properties”命令，将得到如图 7-13 所示的属性设置对话框。

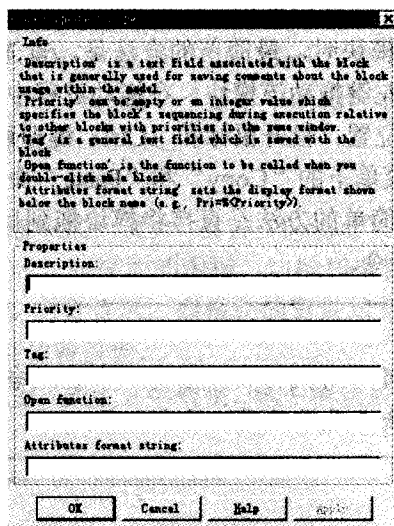


图 7-13 模块属性设置对话框

该对话框包括根据需要设定的 5 个基本属性。

- **Description**（说明）：对该模块在模型中的用法进行说明。
- **Priority**（优先级）：规定该模块在模型中相对于其他模块的优先顺序，优先级的数值必须是整数（可以是负数），该数值越小，优先级越高。
- **Tag**（标记）：用户为模块添加的文本格式的标记。
- **Open function**（调用函数）：当用户双击该模块时调用的 MATLAB 函数。
- **Attributes format string**（属性格式字符串）：用来在模块旁边显示字符串，一般是关于模块属性的字符串，这里填入的任何内容都将在模块的旁边显示出来，但不是模块名的一部分。

### 7.3.6 模块间的连线

#### 1. 连接两个模块

这是 Simulink 仿真最基本的操作情况，即从一个模块的输出端连到另一个模块的输入端。方法是先移动鼠标到输出端，鼠标的箭头会变成十字形光标，这时按住鼠标左键，拖动鼠标到另一个模块的输入端，当十字形光标出现“重影”时，释放鼠标即完成了连接。

如果两个模块不在同一水平线上，连线是一条折线。若要用斜线表示，需要在连线后，选中连线，再按住〈Shift〉键进行拖动。两种连线的效果如图 7-14 所示。

#### 2. 模块间连线的调整

用鼠标单击连线，可以选中该连线。这时会看到线上的一些黑色小方块，这些是连线的关键点。用鼠标点住关键点，拖动即可以改变连线的方向。



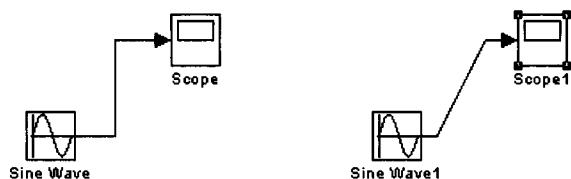


图 7-14 模块的连接

### 3. 连线的分支

仿真时经常会碰到需要把信号输送到不同的接收端的情况，这时就需要分支结构的连线。可以先连好一条线，然后把鼠标移到支线的起点位置，先按下〈Ctrl〉键，然后按住鼠标，将连线拖到目标模块，释放鼠标和〈Ctrl〉键即可。

### 4. 标注连线

双击某一条连线，可以看到一个文本框，在里面输入标注文字，按〈Enter〉键确定。还可以将这个文本框拖到合适的位置。

另外在“Format”菜单下还有命令与连线标注有关。

- Sample Time Color: 将采样时间不同的模块和连线用不同的颜色显示。
- Port Data Types: 在连线上显示传输数据的类型，如 double、int32 等。

### 5. 删除连线

如果想要删除某条连线，可点击要删除的连线，此时连线上出现标记点，表示该连线已经被选中，然后按模型窗口工具栏中的剪切按钮或者直接单击键盘上的〈Delete〉键，即可删除该连线。

## 7.4 仿真模型的参数设置

在仿真系统设计过程中，事先还必须对仿真算法、输出模式等各种模型参数进行设置，可以在模型窗口通过菜单命令进行设置。选择模型窗口“Simulation”菜单中的“Simulink parameters...”选项，将出现仿真参数对话框，如图 7-15 所示。

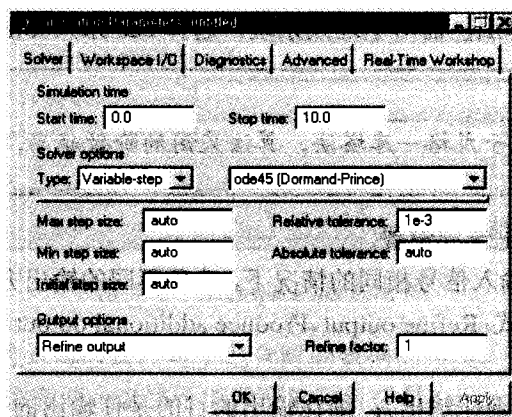


图 7-15 仿真模型参数设置选项卡（Solver）

对话框分 5 个选项卡: Solver (算法)、Workspace I/O (工作空间输入输出)、Advanced (高级)、Diagnostics(诊断)和 Real-TimeWorkshop (实时工作空间), 其中前三项经常用到, 下面就分别介绍。

### 7.4.1 Solver 选项卡

Solver 选项卡如图 7-15 所示。

● Simulink time: 设置仿真起始时间和停止时间。

设置起始和停止时间分别是在“Star time”和“Stop time”两个编辑框内, 通过直接输入数值来完成的, 时间单位是秒。

● Solver options: 仿真算法的选择。

仿真算法根据步长 (Type) 的变化分为固定步长算法 (Fixed—step) 和变步长算法 (Variable—step)。固定步长是指在仿真过程中计算的步长不变, 而变步长算法是指在仿真过程中要根据计算的要求改变步长。对于这两种算法, 它们所对应的相关选项以及具体算法都有所不同。

在采用变步长算法时, 首先应该指定允许的误差限, 包括相对误差限 (Relative tolerance) 和绝对误差限 (Absolute tolerance), 当计算过程中的误差超过该误差限时, 系统将自动调整步长, 步长的大小将决定仿真的精度。在采用变步长算法时还要设置最大步长 (Max step size), 在默认值 (auto) 的情况下, 系统所给定的最大步长为:

最大步长=(终止时间一起始时间)/50

在一般情况下, 系统给的最大步长已经足够, 但如果用户所进行的仿真时间过长, 则默认步长值就非常大, 有可能出现失真的情况, 这时应根据需要设置较小的步长。

在采用固定步长算法时, 要先设置固定步长。由于固定步长的步长不变, 所以此时不设定误差限, 而多了一个模型类型 (Mode) 的选项, 该选项包括: 多任务 (Multi Tasking)、单任务 (Single Tasking) 和默认值 (Auto)。多任务是指在模型中模块具有不同的采样速率, 系统同时检测模块之间采样速率的传递; 单任务是指各模块的采样速率相同, 不检测采样速率的传递; 默认值则根据模块的采样速率是否相同决定采用单任务还是多任务。

变步长和固定步长都是算法的模型, 这两种模型分别对应许多种不同算法。一般对于离散系统, 要选择 discrete 算法; 而对于连续系统, 选择 ode 系列算法。

提示:

ode 系列算法基于龙格—库塔法, 算法采用的阶数越高, 计算越精确。

● Output options: 输出选项设置。

对于同样的模型, 在输入信号相同的情况下, 选项不同的输出方式可以产生不同的效果。MATLAB 提供 3 种输出方式: Refine output、Produce additional output 和 Produce specified output only。

(1) Refine output (细化输出): 细化输出的目的是使输出的数据曲线更加平滑。它根据细化系数 (Refine factor) 可以在每个时间内加入若干资料, 使得输出曲线更加连续、平滑。例如, 细化系数是 3, 则把每个步长分成 3 等份, 在 1/3 和 2/3 的地方各加入一个资料。

细化系数越大，细化后的曲线越平滑。但无限的增大细化系数会导致数据曲线的失真。细化输出的方式比较适合于变步长的算法。

(2) Produce additional output (产生附加输出)：由用户指定产生输出的附加时刻。选择该项后，右边会产生输出时刻编辑框，在此可以输入附加时刻向量。这种输出方式在仿真计算时改变步长和指定的附加输出时刻相一致。

(3) Produce specified output only (仅在指定的时刻产生输出)：仅提供在指定的时间点上的输出值。当要比较同一个模型采用不同的仿真方法所得到的结果时，使用这一选项可以产生在一些共同时间点上的输出。

## 7.4.2 Workspace I/O 选项卡

Workspace I/O 选项卡如图 7-16 所示。

● Load from workspace: 从工作空间中载入数据。

在仿真过程中，如果模型中有输入端口 (In 模块)，可从工作空间直接把数据载入到输入端口，即先选中 “Input” 复选框，再在后面的编辑框内输入数据的变量名。变量名的输入形式有许多种，下面对各种形式分别介绍。

(1) 矩阵的形式。如果以矩阵的形式输入变量名，则矩阵的列数必须比模型的输入端口数多一个，MATLAB 把矩阵的第一列默认为时间向量，后面的每一列对应每一个输入端口，矩阵的第一行表示某一时刻各输入端口的输入状态。另外，也可以把矩阵分开来表示，即 MATLAB 默认的表达方法  $(t, u)$ ， $t$  是一维时间列向量； $u$  是和  $t$  长度相等的  $n$  维列向量 ( $n$  表示输入端口的数量)，表示状态值。

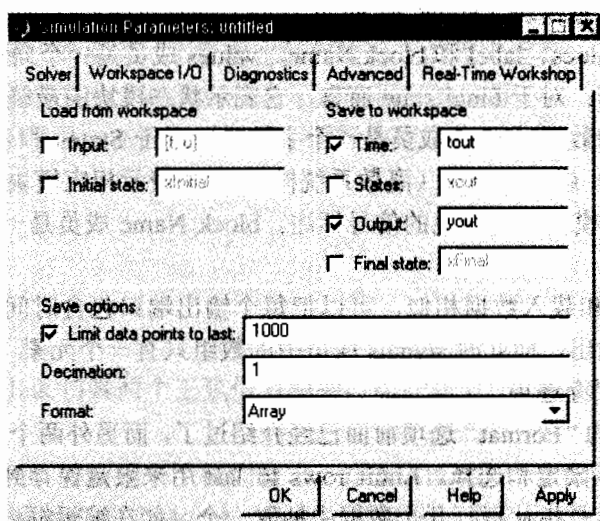


图 7-16 Workspace I/O 选项卡

(2) 包含时间数据的结构形式。对于包含时间数据的结构，在 MATLAB 中有非常严格的规定，即在结构中必须有两个名字不能改变的顶级成员：time 和 signals。在 time 成员中包含一个列向量，表示仿真时间；在 signals 成员中包含一个数组，数组中的每个元素对应一个输入端口，并且每个元素必须包含一个名字同样不能改变的 values 成员，values 成员

也包含一个列向量，对应于输入端口的输入数据。

(3) 综合形式。在综合形式中，可以把包含时间数据的结构和不包含时间数据的结构分别输入到不同的端口，即为上面两种结构形式的综合。

在 Input 的下面，还有一个“Initial state”的复选框，它表示的是模块的初始化状态。对模块进行初始化的方法是：先选中“Initial state”复选框，然后在后面的编辑框中输入初始化数据的变量名，对于变量要求的几种形式，与前面的输入端口数据的变量形式基本相同，但变量中的数据个数必须和状态模块数相同。

● Save to workspace: 将输出保存到工作空间。

在 Save to workspace 区域中，可以选择的选项有：Time（时间）、States（状态）、Output（输出端口）和 Final state（最终状态）。同载入数据的形式一样，保存的数据也有矩阵、包含时间数据的结构和综合结构几种形式，在 Save options 区域中的 Format 下可以根据需要进行选择。对于不同的保存形式来说 Time 的格式是不变的，总是对应仿真的采样时间。下面介绍一下其他几个选项在不同形式下的变化情况。

(1) 矩阵形式。在使用矩阵的形式时，States 输出的也是矩阵，每一列对应一个模块的状态值，每一行对应一个时刻各个模块的状态值；Options 输出的也是矩阵，矩阵行列的意义与 States 相似，但它只能是输出端口（Output）的值，如果模型中没有输出端口，则 Output 不能生成矩阵；Final state 同样也能输出矩阵，但输出的矩阵只有一行，表示每一个模块的最终状态。

(2) 包含时间数据的结构形式。同载入数据时介绍的一样，结构包含两个顶级成员：time 和 signals。time 成员为一个列向量，表示仿真时间；signals 成员包括一个子结构数组，子结构的个数应该等于状态模块（States 和 Final state）或输出端口（Output）的个数。每个子结构含有 3 个成员：values、label 和 block Name。values 成员是一个向量，对于 States 而言，它表示状态模块的数据；对于 Final state 而言，它表示状态模块的最终数据；对于 Output 而言，它表示输出端口的数据。label 成员是一个字符串，对于 States 和 Final state，其内容是 Cstate（连续系统模块）或 Dstate\_n（离散系统模块），n 表示相应模块中有 n 组离散状态；对于 output，其内容是模块间连线上的信号标注。block Name 成员是一个字符串，表示相应模块的名称。

(3) 综合形式。和载入数据相似，可以把每个输出端口包含时间数据的结构和不包含时间数据的结构一起输出，而此时 signals 成员中的数组只有一个元素。

● Save options: 保存选项

Save options 区中的“Format”选项前面已经介绍过了，而另外两个参数 Limit rows to last 和 Decimation 分别表示限定和选择。Limit rows to last 用来限定保存到工作空间中的数据的最大行数；Decimation 是指从每行几个数据中抽取一个，如在编辑框内输入 3，则表示输出数据时，每 3 个数据取一个，也就是每隔两个数据取一个数据。

### 7.4.3 Diagnostics 选项卡

在 Diagnostics 选项卡中，主要是指定系统对一些事件或仿真过程中可能遇到的情况做出什么反映。Diagnostics 选项卡如图 7-17 所示。

● Simulink options: 仿真选择。

Consistency checking 表示一致性检验，主要是保证模块对一个固定时间的输出不变。  
Bounds checking 表示范围检测，主要是保证模块之间不发生冲突。

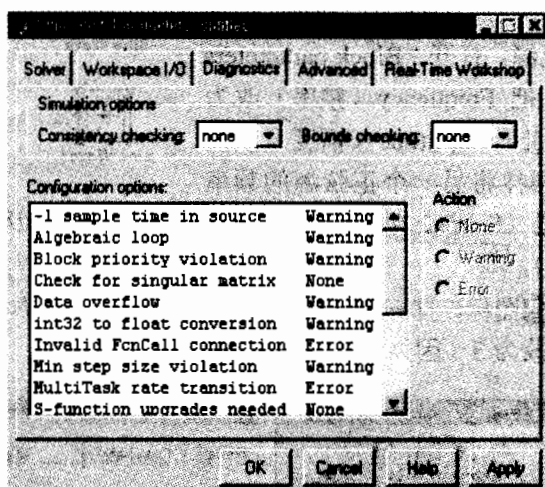


图 7-17 Diagnostics 选项卡

● Configuration options: 系统配置选择。

在该项中列出了系统中各种可能产生错误的项，用户可以选中任何一项后，在右边的 Action 中根据程序需要进行选择。

在 Diagnostics 选项卡中，主要是指定系统对一些事件或仿真过程中可能遇到的情况做出什么反应，反应的类型有以下几种。

- (1) None: 不做任何反应，不影响在任何情况下的程序运行。
- (2) Warning: 提示警告，但警告信息不影响程序的运行。
- (3) Error: 提示错误，在提示错误后，运行的程序将停止。

## 7.5 上机实践

### 7.5.1 跟我学

例 7-1 用 Simulink 仿真两个正弦信号相乘，即计算  $x(t) = \sin(t) * \sin(10t)$ 。  
仿真过程如下。

(1) 运行 Simulink 并新建一个模型窗口。在 MATLAB 命令窗口菜单上，选择“File”菜单下的“New”子菜单中的“Model”命令。可打开一个名为“untitled (无标题)”的模型窗口。

(2) 将所需模块添加到模型中。单击模块浏览器 (Simulink Library Browser) 中“Simulink”前面的“+”号，将看到 Simulink 基本模块库中包含的子模块库，单击“Sources (信号源模块)”，在右边的窗口中找到“Sine Wave (正弦源)”，然后用鼠标将其拖到模型窗口 (重复一次，得到第二个正弦源)。按照同样的方法，在“Sinks (输出模块)”中把 Scope (示波器) 拖到模型窗口；在“Math (数学模块)”中把 Product (乘法器) 拖到模型窗口。

在模型窗口得到的结果如图 7-18 所示。

(3) 编辑模块组成模型。根据题意的要求，需要两个正弦源的频率分别是 1Hz 和 10Hz，幅度均为 1。先双击一个正弦源，打开“Block parameters (模块参数)”对话框，把 Frequency (频率) 改为  $2\pi$  (角频率弧度制)；把 Amplitude (幅度) 改为 1，其他参数不用改。同样将另一个正弦源的频率改为  $20\pi$ 。双击示波器 (Scope) 看到一个图形界面 (如图 7-19 左图所示)。

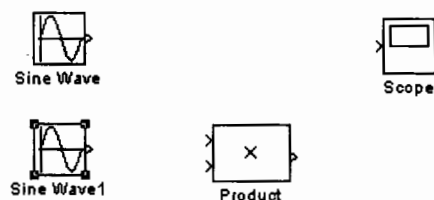



图 7-18 选好的仿真模块

单击其工具栏上的  图标，打开示波器属性窗口 (如图 7-19 右图所示)，将“Number of axes (坐标轴的数量)”改为 3 (因为要看三个波形)。

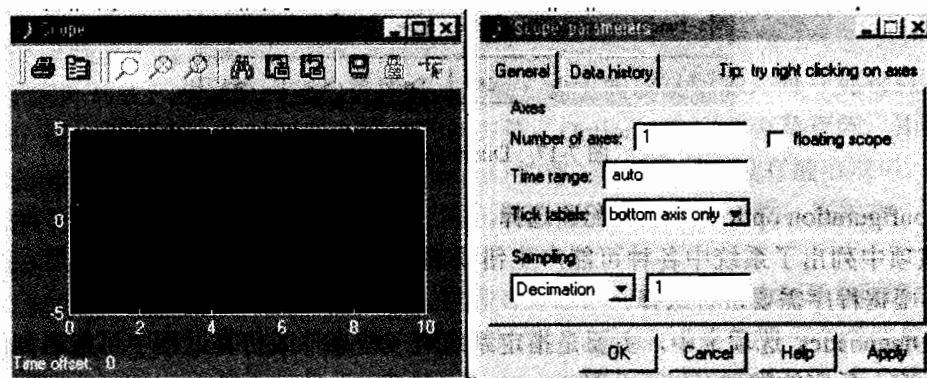


图 7-19 示波器及其属性窗口

然后，将各个模块连接起来。如图 7-20 所示。

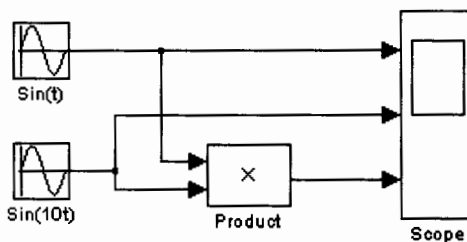



图 7-20 实现信号相乘的仿真系统

(4) 进行系统仿真参数设置。仿真之前，还要设置仿真的时间、步长和算法。单击模型窗口的“Simulink”菜单下的“Simulation parameters...”对话框，打开仿真参数设置对话框，如图 7-21 所示。

把仿真结束时间设置为 2，即仿真时间为 2 秒；把算法选择中的 Type 选择为“Fixed-step (固定步长)”，并在其右边的算法框选择“ode5 (龙格-库塔法的 5 阶算法)”，再把“Fixed step size (固定步长尺寸)”设置为 0.001 秒。

(5) 进行系统仿真。系统仿真参数设置完成后，单击模型窗口中  图标或单击模型

窗口的“Simulink”菜单下的“Star”命令进行仿真。

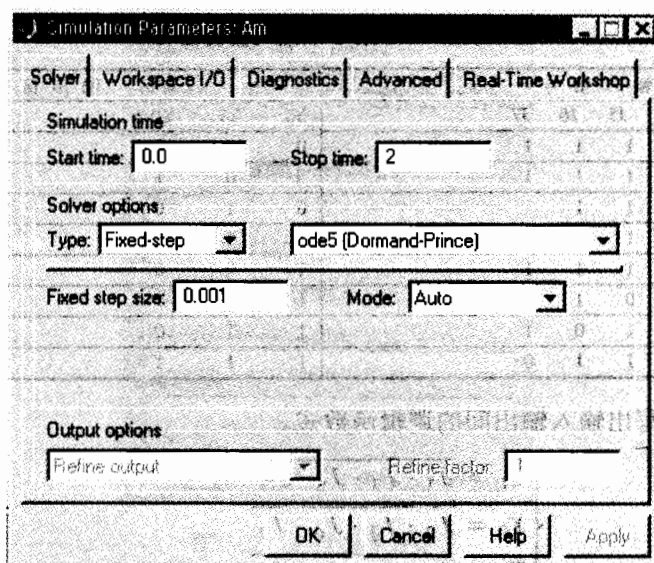


图 7-21 仿真参数设置对话框

(6) 观察系统仿真结果。系统仿真结束后（系统仿真的时间取决于系统的复杂程度），双击模型窗口的示波器图标，可见仿真结果，如图 7-22 所示。

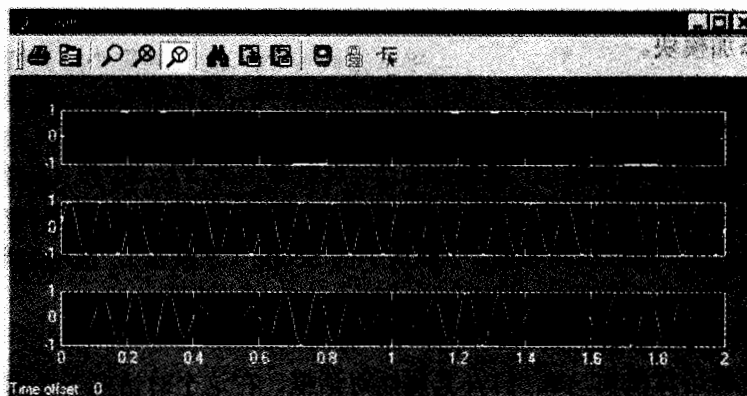



图 7-22 仿真结果

由于示波器的坐标轴刻度设置不同，看到的波形可能会不一样。直接单击示波器窗口工具栏上的  图标，可以自动调整坐标来使波形刚好完整显示。

**例 7-2** 设计一个数字电路的 8 线-3 线编码器，并用 Simulink 仿真。  
设计过程如下。

(1) 分析编码器的功能，写出编码器的逻辑表达式。

编码就是在选定的一系列二值代码中赋予每个代码以固定的含义，执行编码功能的电路统称为编码器。8 线-3 线编码器是对输入端的 8 个信号进行编码，输出三位二进制数，要求输入信号每次只有一个为 0，其余 7 个为 1，其中值为 0 的信号是待编码的信号。根据要求

写出输入输出对应的真值表（如表 7-9 所示）。

表 7-9 8 线-3 线编码器真值表

输入信号								输出信号		
J0	J1	J2	J3	J4	J5	J6	J7	Y2	Y1	Y0
0	1	1	1	1	1	1	1	0	0	0
1	0	1	1	1	1	1	1	0	0	1
1	1	0	1	1	1	1	1	0	1	0
1	1	1	0	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	1	1	1	0	1
1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1

利用真值表，写出输入输出间的逻辑函数式。

$$\begin{cases} Y_0 = \overline{J_1 \cdot J_3 \cdot J_5 \cdot J_7} \\ Y_1 = \overline{J_2 \cdot J_3 \cdot J_6 \cdot J_7} \\ Y_2 = \overline{J_4 \cdot J_5 \cdot J_6 \cdot J_7} \end{cases} \quad (7-1)$$

在写出逻辑表达式之后，可以用与非门来实现这个表达式。

(2) 用 Simulink 仿真实现编码器。


在进行仿真时，在 8 个输入端依次加一个低电平，然后用示波器观察 3 个输出信号的波形。一共分四个步骤。

第一步：添加模块。

在 MATLAB 中运行 Simulink，打开模块库浏览器，然后新建一个模型。将本次仿真需要的模块添加到模型中。首先找到 Simulink 模块库的 Math 子模块库中的 Logical Operator（逻辑运算），拖动到新建的模型窗口，复制成 3 个，重新命名为 Y1、Y2 和 Y3；然后找到 Sources 子模块库中的 Pulse Generator（脉冲激励源），拖动到新建的模型窗口，复制成 8 个，重新命名为 J0、J1、…、J8；最后找到 Sinks 子模块库中的 Scope（示波器），拖动到新建的模型窗口，复制成 3 个。

第二步：修改模块参数。

首先双击逻辑运算模块 Y0，打开其属性对话框，在这个对话框下半部分的参数设置栏内，将操作（Operator）修改为“NAND（与非）”，输入节点数（Number of input ports）修改为 4，然后点击“OK”确定。其余两个逻辑运算模块 Y1、Y2 也同样修改。

然后通过双击示波器模块 Scope，得到一个图形界面，在其工具栏上单击  图标，可以打开示波器属性设置对话框，将坐标轴数（Number of axes）改为 3，同样地，将示波器 Scope1 和 Scope2 的坐标轴数改为 4。

最后修改脉冲源的属性。双击离散脉冲源 J0，将看到关于它的属性对话框，如图 7-23 所示。

可以从对话框中看到 5 个参数需要设置，分别解释如下。

1) Amplitude: 方波信号的幅度。

2) Period: 方波信号的周期（以采样时间为单位）。



3) Pulse width: 脉冲宽度（即电平为 1 的时间，以采样时间为单位）。

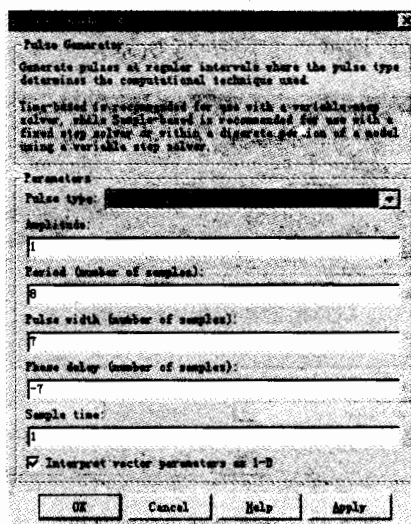


图 7-23 脉冲源参数设置

4) Phase delay: 相位延迟（即信号开始的时间，以采样时间为单位）。

5) Sample Time: 采样时间长度，以秒为单位。

根据编码器的设计要求，J0 到 J7 应依次为低电平，所以 J0 到 J7 的周期设为 8，脉冲宽度设为 7，相位延迟依次为-7 到 0（J0 为-7，J1 为-6，J2 为-5，...），幅度和采样时间采用默认值。这样在零时刻，J0 为低电平，其余输入为高电平；过一个采样时间，J1 变为低电平。这样下去，到第七个采样时间，J7 变为低电平。

第三步：连线及仿真。

根据输入输出的逻辑表达式，可以知道，将 J1、J3、J5、J7 接至 Y0 的输入，将 J2、J3、J6、J7 接至 Y1 的输入，将 J4、J5、J6、J7 接至 Y2 的输入。用示波器 Scope 监测 Y2、Y1、Y0 的输出，Scope1 用来监视 J0~J3 这 4 个波形，Scope2 用来监视 J4~J7 这 4 个波形。完成连线（如图 7-24 所示）。

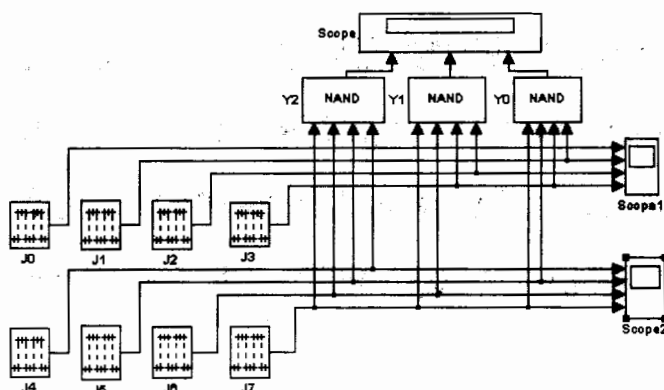


图 7-24 8 线—3 线编码器

第四步：仿真参数设置及系统仿真。

单击模型窗口“Simulink”菜单下的“Simulink parameters...”命令（或者直接用快捷键〈Ctrl〉+〈E〉），打开参数设置对话框，如图 7-25 所示。

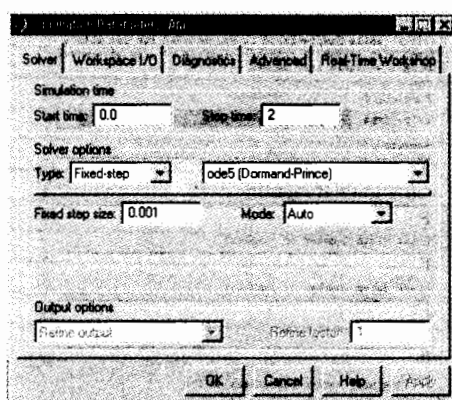


图 7-25 仿真参数设置对话框

将仿真时间（Simulink time）设置为 0 到 20 秒。其他参数采用默认值即可。

然后双击打开 3 个示波器，点击 ▶ 图标开始仿真。仿真结束后，可以从示波器 Scope1 和 Scope2 中看到编码器 8 个输入端的波形（如图 7-26 和图 7-27 所示），可以看出，J0 到 J7 是以 8 个脉宽为周期，依次出现 0 电平。

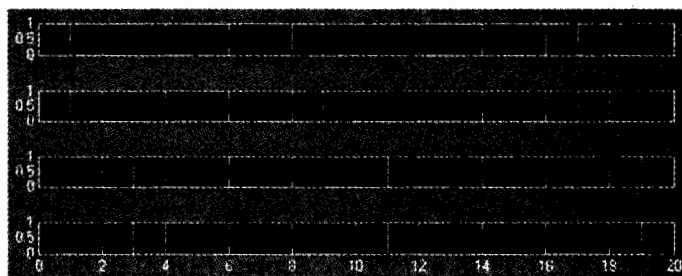


图 7-26 编码器输入波形（前四个脉冲信号）

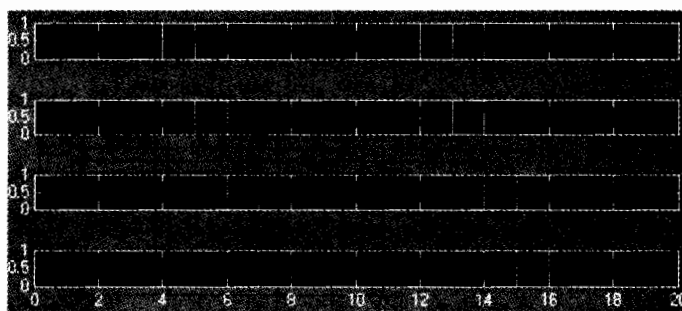


图 7-27 编码器输入波形（后四个脉冲信号）

在示波器 Scope 中可以看到编码器输出波形，如图 7-28 所示。由图中可以清楚地看到，输出的三位二进制码（Y2Y1Y0）依次为：000、001、010、011、100、101、110、111，从

而实现了编码的功能。

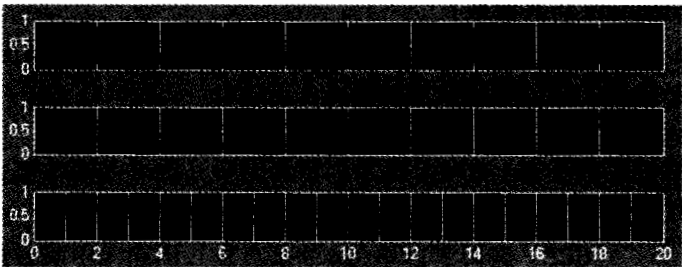


图 7-28 编码器输出波形

注意：

在进行仿真之前，应先将连接完毕的模型存盘，以免由于仿真时出现错误，突然死机，使得前面的工作付诸东流。可在模型窗口打开“File”菜单，选择命令将模型存盘。

7.5.2 自己练

- 1. 浏览 Simulink 模块库。
- 2. 用 Simulink 仿真一个正弦信号和一个余弦信号相加，即计算  $x(t) = 2\sin(2t) + 3\cos(5t)$
- 3. 设计一个数字电路的 3 线-8 线译码器，并用 Simulink 仿真。

7.6 本章小结

本章主要讲解了 Simulink 仿真系统的基本模块、模块的建立及其操作方法、参数设置等内容。通过编码器设计的仿真过程，系统的介绍了从建模到参数设置仿真的全部过程。让读者对 Simulink 有个全面的了解，Simulink 模块库包含的子模块库名称如表 7-10 所示。

表 7-10 Simulink 子模块库

模块名称	说明	模块名称	说明
Simulink	启动 Simulink 模块库浏览器	Source	信号源模块
Sinks	输出模块	Discrete	离散系统模块
Continuous	连续系统模块	Functions&Tables	函数和表模块
Math	数学模块	Signals&Systems	信号和系统模块
Nonlinear	非线性系统模块	Subsystems	创建子系统模块

7.7 习题

- 1. 什么是 Simulink？其主要功能是什么？

- 
2. 应用 Simulink 仿真的主要步骤有哪些?
  3. 如何设置 Simulink 仿真参数?
  4. 利用 Scope (示波器) 观察 Source (信号源) 中的各种信号并画出波形。
  5. 用 Simulink 仿真二位二进制数加法器。

## 第8章 MATLAB 应用实例

本章通过一些应用实例的介绍,使读者可以掌握 MATLAB 程序设计的基本方法。了解 MATLAB 在相关领域的应用价值,能够应用 MATLAB 来解决自己专业领域的实际问题。本章重点是 8.3 在图像处理中的应用,难点是 8.4.2 单边带调幅(SSB)。

### 8.1 数字滤波器的设计

滤波器是指用来对信号进行滤波的硬件或软件。如果滤波器的输入、输出都是离散时间信号,则滤波器的冲激响应也必然是离散的,这样的滤波器定义为数字滤波器。数字滤波器在数字信号处理的各应用中起着重要的作用,它是通过对采样数据信号进行数学运算处理来达到频域滤波的目的。从数字滤波器的实现方法上看,由有限冲激响应所表示的数字滤波器被称为 FIR 滤波器;具有无限冲激响应的数字滤波器则称为 IIR 滤波器。数字滤波器的设计包括以下三个基本步骤:

- 给出所需要的滤波器的技术指标。
- 设计一个系统函数使其逼近所需要的技术指标。
- 实现所设计的系统函数。

但 FIR 滤波器和 IIR 滤波器在设计方法上又有各自的特点,下面分别介绍。

#### 8.1.1 FIR 滤波器的设计

FIR 滤波器可以对给定的频率特性直接设计,有 Fourier 级数展开设计法、窗函数法和频率采样法等。

**例 8-1** 用窗函数法设计一个多带通滤波器,归一化的通带是 $[0, 0.2]$ 、 $[0.4, 0.6]$ 和 $[0.7, 1.0]$ ,滤波器的阶数为 40。滤波器的频率响应见图 8-1。

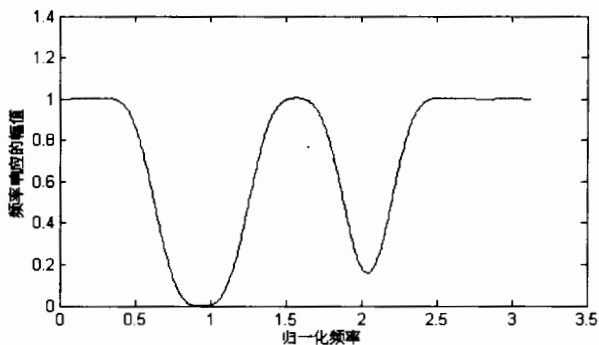


图 8-1 滤波器的频率响应

设计程序如下:

```

clear                %清除工作空间变量
clc                  %清屏幕
w=[0.2,0.4,0.6,0.7];
b=fir1(40,w,'dc-1'); %基于窗函数的 FIR 标准响应滤波器设计
[h,f]=freqz(b);      %数字滤波器的频率响应
plot(f,abs(h));
xlabel('归一化频率');
ylabel('频率响应的幅值');

```

注意:

基于窗函数的 FIR 标准响应滤波器设计函数 `fir1` 中的“1”是数字“1”而不是字母“l”，二者不太好区别。

### 8.1.2 IIR 滤波器的设计

IIR 滤波器目前通用的设计方法是借助模拟滤波器原型，再将其转换成数字滤波器。MATLAB 工具箱提供了几个设计数字滤波器的函数，可以直接调用。

**例 8-2** 设计一个低通 Butterworth 数字滤波器，要求达到的指标： $w_p$ （通带上限临界频率）=30Hz， $w_s$ （通带下限临界频率）=35Hz， $F_s$ （采样频率）=100Hz， $r_p$ （通带内的最大衰减）=0.5dB， $r_s$ （阻带内的最小衰减）=40dB。Butterworth 数字滤波器频率响应如图 8-2 所示。

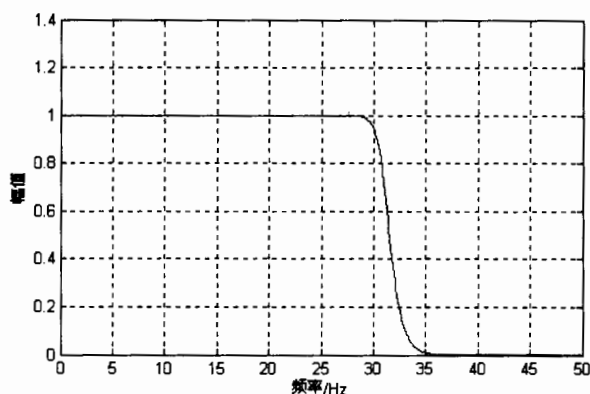


图 8-2 Butterworth 数字滤波器频率响应

设计程序如下:

```

clear                %清除工作空间变量
clc                  %清屏幕
wp=30;ws=35;Fs=100;rp=0.5;rs=40;
[N,wn]=buttord(wp/(Fs/2),ws/(Fs/2),rp,rs,'z'); %Butterworth 滤波器阶的选择
[num,den]=butter(N,wn); %设计 Butterworth 滤波器
[H,W]=freqz(num,den); %数字滤波器的频率响应

```

```

plot(W*Fs/(2*pi),abs(H));grid;
xlabel('频率/Hz');
ylabel('幅值')

```

## 8.2 在信号处理中的应用

信号是信息的载体，信息是通过各种各样的信号来进行传递的，携带着不同信息的不同信号之间可能相互叠加，而且信号在传递过程中还不可避免地受到噪声的干扰。因此，如何从所收到的信号中提取所需要的信息，是信号处理的基本任务。MATLAB 的信号处理工具箱和数字信号处理组件提供了许多函数可供使用。

### 8.2.1 快速傅立叶变换 (FFT)

离散傅立叶变换（简称 DFT）是信号分析与处理中的一种重要变换。但直接计算 DFT 的运算量与变换的长度  $N$  的平方成正比，当  $N$  较大时，计算量太大。在快速傅立叶变换（简称 FFT）出现以前，直接用 DFT 算法进行频谱分析和信号的实时处理是不实际的。FFT 使得 DFT 的运算效率大大提高，为数字信号处理技术应用于各种信号的实时处理创造了条件，推动了数字信号处理技术的发展。

MATLAB 提供 fft 函数来计算数字信号的快速傅立叶变换，提供 ifft 函数来计算数字信号的快速傅立叶逆变换。这两种函数是用机器语言编写的，因此它的执行速度很快。其函数格式如下：

●  $y = \text{fft}(x)$

说明：计算信号  $x$  的离散傅立叶变换  $y$ 。当  $x$  为矩阵（多通道信号）时，计算  $x$  中每一列信号的离散傅立叶变换。

●  $y = \text{fft}(x, n)$

说明：计算  $n$  点傅立叶变换，当  $x$  的长度大于  $n$  时，截断  $x$ ；当  $x$  的长度小于  $n$  时，补零  $x$ 。使  $n$  和  $x$  等长。

●  $x = \text{ifft}(y)$

说明：当  $y$  为向量时，计算信号  $y$  的一维离散傅立叶逆变换。当  $y$  为矩阵（多通道信号）时，计算  $y$  中每一列信号的离散傅立叶逆变换。

●  $x = \text{ifft}(y, n)$

说明：计算  $n$  点离散傅立叶逆变换，在变换前对  $y$  进行补零或截去多余元素，使  $y$  的长度与  $n$  相等。

**例 8-3** 有一个由 50Hz 和 300Hz 正弦信号构成的信号，受到均值随机噪声的污染，现对该信号进行采样，采样频率为 1000Hz，通过快速傅立叶变换来分析其信号频率成分（如图 8-3 所示）。

程序如下：

```

clear
clc
t=0:0.001:1.5;                                %时间间隔 0.001，说明采样频率为 1000Hz

```

$x = \sin(2\pi \cdot 50 \cdot t) + \sin(2\pi \cdot 300 \cdot t);$     %产生主要频率为 50Hz 和 300Hz 的信号

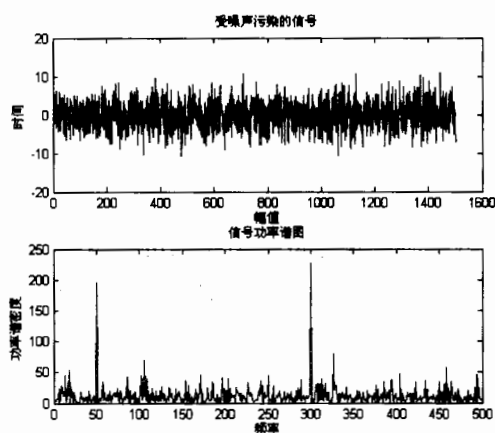


图 8-3 原始信号与功率谱

```
f=x+3.5*randn(1,length(t)); %加入均值随机噪声
subplot(211);plot(f);
Xlabel('幅值');Ylabel('时间');
title('受噪声污染的信号');
y=fft(f,1024); %快速傅立叶变换, 采样点为 1024 个
p=y.*conj(y)/1024; %计算功率谱密度
ff=1000*(0:511)/1024; %计算变换前后不同点所对应的频率值
subplot(212);plot(ff,p(1:512)); %画出信号的频率谱
Xlabel('频率');
Ylabel('功率谱密度');
title('信号功率谱图');
```

从图可见信号集中在 50Hz 和 300Hz。

例 8-4 用 FFT 变换分析语音信号的频谱 (如图 8-4 所示)。

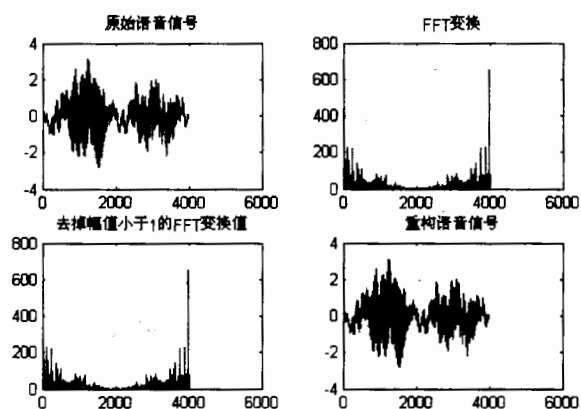


图 8-4 语音信号的频谱

用 MATLAB 实现如下:



```

load mtlb; %读入原始语音信号
subplot(221);plot(mtlb);title('原始语音信号')
y=fft(mtlb); %快速傅立叶变换
subplot(222);plot(abs(y));title('FFT 变换');
y(abs(y)<1)=0; %把绝对值小于1的系数量化为0
x=ifft(y); %快速傅立叶逆变换
subplot(223);plot(abs(y));title('去掉幅值小于1的FFT变换值')
subplot(224);plot(real(x));title('重构语音信号')

```

## 8.2.2 离散余弦变换 (DCT)

离散余弦变换 (DCT) 是仅次于 K-L 变换的次最佳正交变换, 且已获得广泛的应用, 并成为许多图像编码国际标准的核心。DCT 的变换核为余弦函数, 计算速度较快, 有利于图像压缩和其他处理。在大多数情况下, DCT 用于图像的压缩操作中。JPEG (Joint Photographic Experts Group) 图像格式的压缩算法采用的就是离散余弦变换。其相关函数格式如下。

●  $y = \text{dct}(x)$

说明:  $x$  为向量, 计算信号  $x$  的一维离散余弦变换  $y$ 。当  $x$  为矩阵 (多通道信号) 时, 计算  $x$  中每一列信号的离散余弦变换。

●  $y = \text{dct}(x, n)$

说明: 计算  $n$  点离散余弦变换。当  $x$  的长度大于  $n$  时, 截断  $x$ ; 当  $x$  的长度小于  $n$  时, 补零  $x$ 。使  $n$  和  $x$  等长。

●  $x = \text{idct}(y)$

说明: 当  $y$  为向量时, 计算信号  $y$  的一维离散余弦逆变换。当  $y$  为矩阵 (多通道信号) 时, 计算  $y$  中每一列信号的离散余弦逆变换。

●  $y = \text{idct}(y, n)$

说明: 计算  $n$  点离散余弦逆变换, 在变换前对  $y$  进行补零或截去多余元素, 使  $y$  的长度与  $n$  相等。

例 8-5 计算信号  $x(n) = n + 50 \cos\left(\frac{2\pi}{40}n\right)$   $n=1, 2, \dots, 500$  的离散余弦变换 (如图 8-5 所示)。

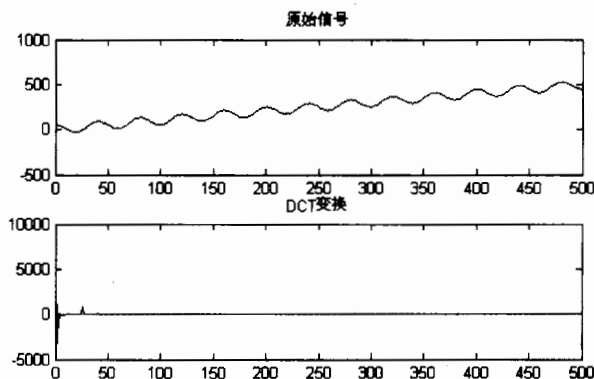


图 8-5 信号的离散余弦变换

```

n=1:500;
x=n+50*cos(2*pi*n/40);    %建立原始信号
y=dct(x);                  %离散余弦变换
subplot(211);plot(x);title('原始信号');
subplot(212);plot(y);title('DCT 变换');

```

从图 8-5 可见，在 DCT 变换域上，信号的能量主要集中在几个系数上，而绝大多数系数接近于零，仅用少数几个变换系数就可表征信号的总体。这一特点是用 DCT 变换进行数据压缩的基本依据。

### 8.3 在图像处理中的应用

图像处理是指利用数字技术，对图像施加某种运算或处理，从而达到某种预想的目的。主要处理有几何操作、消噪、增强、恢复、压缩和分析等。

#### 8.3.1 图像的几何操作

图像的几何操作主要有剪裁、大小调整和旋转，MATLAB 图像工具箱提供了 `imcrop` 函数、`imresize` 函数和 `imrotate` 函数完成上述操作。

**例 8-6** 对一幅图像进行剪裁、放大二倍和旋转  $15^\circ$  的处理（如图 8-6 所示）。

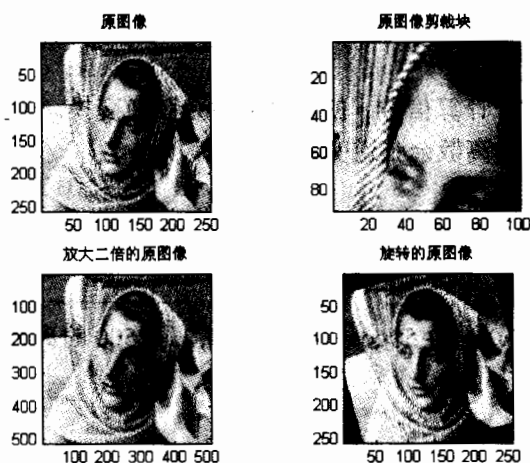


图 8-6 图像几何操作

程序如下：

```

clear
clc
load woman;                %调入原图像
subplot(221);subimage(X,map);title('原图像');    %显示原图像
X1=imcrop(X,map,[60 40 100 90]);                %[60 40 100 90]为剪裁区域
subplot(222);subimage(X1,map);title('原图像剪裁块');
X2=imresize(X,2,'bilinear');                      %参数'bilinear'为双线性插值

```

```
subplot(223);subimage(X2,map);title('放大二倍的原图像');
X3=imrotate(X,15,'bilinear','crop');           %参数'crop'为返回同样大小的图像
subplot(224);subimage(X3,map);title('旋转的原图像');
```

### 8.3.2 图像的消噪处理

图像的消噪可以采用二维中值滤波函数（medfilt2）。给图像增加噪声主要用 imnoise 函数，该函数有 gaussian（高斯白噪声）、salt & pepper（黑白像素点噪声）和 specakle（乘性噪声）三个参数。

**例 8-7** 给 woman 图像加入黑白点噪声，然后做消噪处理（如图 8-7 所示）。  
程序如下：

```
clear
clc
load woman;      %调入 woman 图像
X1=imnoise(uint8(X),'salt & pepper');
subplot(121);subimage(X1,map);title('加入噪声的图像');
X2=medfilt2(X1);
subplot(122);subimage(X2,map);title('处理后的图像');
```

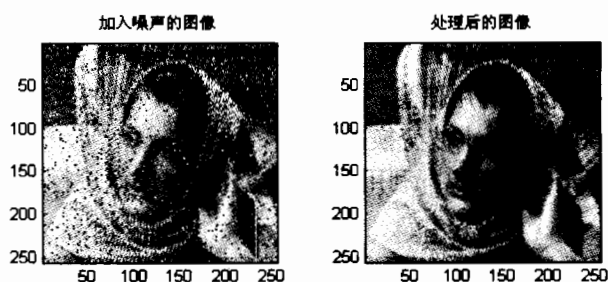


图 8-7 图像的消噪处理

**注意：**

imnoise 函数的参数 salt & pepper，在 & 的前后必须有空格。

### 8.3.3 图像融合

图像融合是将同一对象的两个或更多的图像合成在一幅图像中，以便更好地理解图像。是图像分析的一种手段。

**例 8-8** 利用矩阵加法，将两幅图像融合在一起（如图 8-8 所示）。  
程序如下：

```
clear
clc
load woman;      %调入第一幅 woman 图像
```

```

X1=X;
subplot(131);subimage(X1,map);
title('woman 图像');
load wbarb;           %调入第二幅 wbarb 图像
X2=X;
subplot(132);subimage(X2,map);
title('wbarb 图像');
X3=0.5*(X1+X2);       %矩阵相加, 并减小亮度
subplot(133);subimage(X3,map);
title('融合后的图像');

```

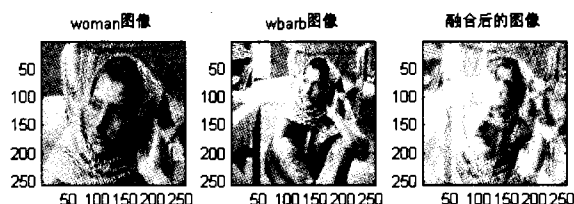


图 8-8 二幅图像和融合后的图像

### 8.3.4 图像压缩

图像数据往往存在各种信息的冗余,如空间冗余、信息熵冗余、视觉冗余和结构冗余等。压缩就是去掉各种冗余,保留有用信息。在图像压缩方面,小波变换得到了广泛的重视,最新的压缩标准 JPEG2000 就采用小波变换。

**例 8-9** 给出一幅 woman 图像,利用二维小波分析对图像进行压缩(如图 8-9 所示)。

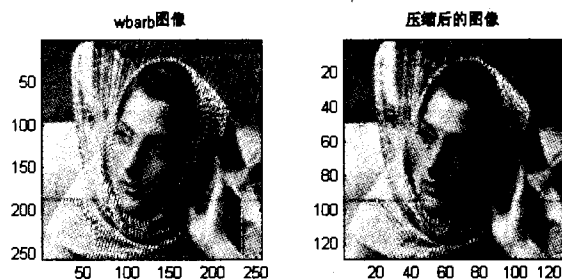


图 8-9 原图像与压缩后的图像

程序如下:

```

clear
clc
load woman;           %调入 woman 图像
subplot(121);image(X);colormap(map);
title('wbarb 图像');
axis square
disp('压缩前图像的大小为:');
whos('X')

```

```

[c,s]=wavedec2(X,1,'haar');    %对图像用 haar 小波进行 1 层小波分解
ca1=appcoef2(c,s,'haar',1);    %提取第一层的低频系数
X2=wcodemat(ca1,220,'mat',0);  %量化编码, 系数变化范围为 0~220
subplot(122);image(X2);colormap(map);
title('压缩后的图像');
axis square
disp('压缩后图像的大小为:');
whos('X2')

```

在命令窗口观察压缩前后图像大小的变化:

压缩前图像的大小为:

```

Name      Size      Bytes  Class
X         256x256    524288  double array
Grand total is 65536 elements using 524288 bytes

```

压缩后图像的大小为:

```

Name      Size      Bytes  Class
X2        128x128    131072  double array
Grand total is 16384 elements using 131072 bytes

```

## 8.4 在通信技术中的应用

信息技术革命是伴随着通信技术的发展而进行的, 现代通信系统是信息时代的生命线。MATLAB 中有专用的通信工具箱, 其中提供了相当多的通信方面的函数。

### 8.4.1 信源编码

通信的根本问题是如何将信源输出的信息在接收端的信宿精确或近似地复制出来。为更有效地实现复制, 信源编码就相当重要, 因为它能实现通信系统与信源统计特性的匹配。若接收端要求无失真地精确复制信源输出的消息, 这时的信源编码是无失真编码。只有对离散信源可以实现无失真编码, 对连续信源其输出的信息量可以为无限大, 因此是不可能实现无失真编码的。离散信源的无失真编码实际上是一种概率匹配编码, 它可以进一步分为有记忆和无记忆的编码。下面主要讨论无失真编码中的最佳变长编码—哈夫曼编码。哈夫曼编码步骤如下:

- (1) 信源消息(符号)按概率大小顺序排队;
- (2) 从概率最小的两个消息开始编码, 并给以一定的规则, 例如小概率的下支路若编为 1, 则大概率的上支路就应编为 0;
- (3) 将已编码的两个消息对应的概率合并, 并重新按概率大小排序, 重复步骤(2);
- (4) 重复步骤(3), 直到合并概率得到了 1 为止;
- (5) 编程的码字按“后出先编”的方式, 即从概率归一的树根逆行至对应的信源消息。

**例 8-10** 设有一个离散无记忆信源 S, 其信源空间为:

$$\begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} U1 & U2 & U3 & U4 & U5 & U6 & U7 \\ 0.20 & 0.19 & 0.18 & 0.17 & 0.15 & 0.10 & 0.01 \end{bmatrix}$$

求其作为一元信源的哈夫曼编码。

(1) 在哈夫曼编码中，为出现概率小的信源分配较长的码字，而对出现概率较大的信源分配较短的码字。对该信源得到的编码方法如图 8-10 所示。

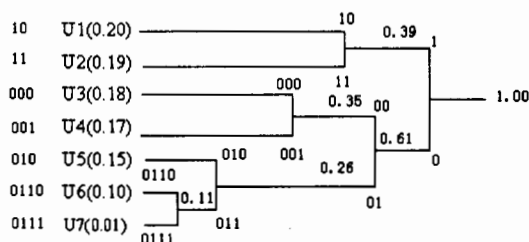


图 8-10 哈夫曼编码树

编码结果为：

$U1$	$U2$	$U3$	$U4$	$U5$	$U6$	$U7$
10	11	000	001	010	0110	0111

(2) MATLAB 程序如下。

● 先编写一个函数文件 huffman.m。

```
function [h,l]=huffman(p);
if length(find(p<0))~=0 %判断输入概率有无负数
    error('Input is not a prob.vector, there is negative component')
end
if abs(sum(p)-1)>10e-10 %判断输入概率之和是否等于 1
    error('Input is not a prob.vector, the sum of the components is not equal to 1.')
end
n=length(p); %得到输入的元素个数
q=p;
m=zeros(n-1,n);
for i=1:n-1
    [q,l]=sort(q); %将输入的概率排序
    m(i,:)=l(1:n-i+1),zeros(1,i-1);
    q=[q(1)+q(2),q(3:n),1];
end
for i=1:n-1
    c(i,:)=blanks(n * n); %加入空白字符
end
c(n-1,n)='0';
c(n-1,2*n)='1';
for i=2:n-1
    c(n-i,1:n-1)=c(n-i+...
1,n*(find(m(n-i+1,:)==1))-(n-2):n*(find(m(n-i+1,:)==1)));
    c(n-i,n)='0';
```

```

c(n-i,n+1:2*n-1)=c(n-i,1:n-1);
c(n-i,2*n)='1';
for j=1:i-1
c(n-i,(j+1)*n+1:(j+2)*n)=...
c(n-i+1,n*(find(m(n-i+1,:)==j+1)-1)+1:n*find(m(n-i+1,:)==j+1));
end
end
for i=1:n
h(i,1:n)=c(1,n*(find(m(1,:)==i)-1)+1:find(m(1,:)==i)*n);
ll(i)=length(find(abs(h(i,:))~=32));
end
l=sum(p.*ll);

```

- 在命令窗口调用哈夫曼函数，完成编码。

```

>> p=[0.20,0.19,0.18,0.17,0.15,0.10,0.01]; %定义概率序列
>> [h,l]=huffman(p) %调用函数，完成编码

h = %得到的结果
    01 %按概率的大小排列
    00
    111
    110
    101
    1001
    1000
l = %信源的平均信息量
    2.7200

```

## 8.4.2 单边带调幅 (SSB)

模拟信号的连续波调制是以正弦波为载波的调制方式，常规双边带调幅和抑制载波双边带调幅都有一个缺点就是占用频带资源过宽，从频谱图能看出有一半的频带资源被浪费了，为此又设计出了单边带 (SSB) 调幅调制方式，此种方式的优点是调制效率高，占用频带资源相对较少。利用 MATLAB 可以实现单边带调幅，还可对信号进行分析。

**例 8-11** 一个未调制带限信号  $f(t) = \begin{cases} \sin c(200t)^2 & |t| \leq t_0 \\ 0 & \text{其他} \end{cases}$

式中：  $t_0=2s$ ，载波  $C(t) = \cos 2\pi f_c t$ ，  $f_c=100\text{Hz}$ ，作出单边带调幅波  $M(t)$  的波形（如图 8-11 所示）。

单边带的时域表达式中含有希尔伯特变换，其表示式如下：

$$\begin{aligned}
 M_{USSB}(t) &= \frac{1}{2}[S(t)\cos 2\pi f_c t - S(\hat{t})\sin 2\pi f_c t] \\
 M_{LSSB}(t) &= \frac{1}{2}[S(t)\cos 2\pi f_c t + S(\hat{t})\sin 2\pi f_c t]
 \end{aligned}
 \tag{8-1}$$

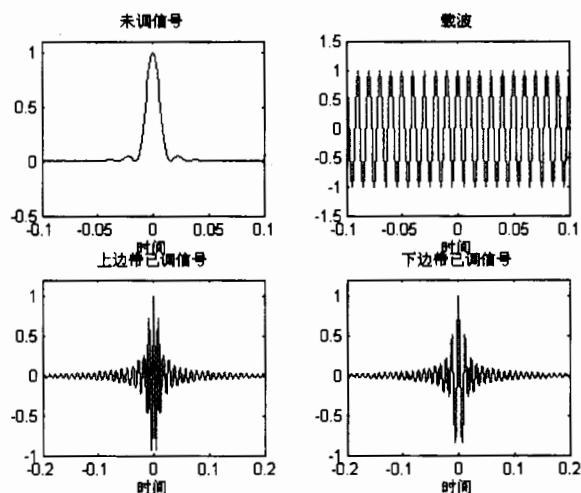


图 8-11 单边带调幅波形

式中:  $S(\hat{t})$  为  $S(t)$  的希尔伯特变换。

MATLAB 程序如下:

- 先编写一个函数文件 fftseq。

```
function [M,m,df]=fftseq(m,tz,df)
fz=1/tz;
if nargin==2 %判断输入参数的个数是否符合要求
    n1=0;
else ni=fz/df; %根据参数个数决定是否使用频率缩放
end
n2=length(m);
n=2^(max(nextpow2(n1),nextpow2(n2)));
M=fft(m,n); %进行离散傅立叶变换
m=[m,zeros(1,n-n2)];
```

- 编辑 M 文件, 实现单边带调幅 (SSB)。

```
t0=2; %信号持续时间
ts=0.001; %采样时间
fc=100; %载波频率
fs=1/ts; %采样频率
df=0.3; %频率分辨率
t=[-t0/2:ts:t0/2];
x=sin(200*t); %未调制信号
m=x./(200*t);
m(1001)=1;
m=m.*m;
c=cos(2*pi*fc.*t); %载波同向分量
b=sin(2*pi*fc.*t); %载波正交分量
v=m.*c+imag(hilbert(m)).*b; %计算出下边带调幅分量
u=m.*c-imag(hilbert(m)).*b; %计算出上边带调幅分量
```



```

[M,m,df1]=fftseq(m,ts,df);           %调用 fftseq 函数进行傅立叶变换
M=M/fs;
[U,u,df1]=fftseq(u,ts,df);
U=U/fs;
[V,v,df1]=fftseq(v,ts,df);
V=V/fs;
f=[0:df1:df1*(length(m)-1)]-fs/2;
subplot(221);plot(t,m(1:length(t)));
axis([-0.1,0.1,-0.5,1.1])
xlabel('时间');title('未调信号')
subplot(222);plot(t,c(1:length(t)));
axis([-0.1,0.1,-1.5,1.5])
xlabel('时间');title('载波')
subplot(223);plot(t,u(1:length(t)));           %作出上边带信号波形
axis([-0.2,0.2,-1,1.2]);xlabel('时间');
title('上边带已调信号')
subplot(224);plot(t,v(1:length(t)));           %作出下边带信号波形
axis([-0.2,0.2,-1,1.2]);xlabel('时间');
title('下边带已调信号');

```

## 8.5 在控制系统分析中的应用

MATLAB 的控制系统工具箱 (Control System Toolbox) 和 Simulink 仿真软件都可以进行控制系统分析, 其实质是对描述系统的数学模型进行求解。

### 8.5.1 线性反馈系统

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, 因此在仿真过程中需要以某种数值算法从给定的初始条件出发, 逐步地算出每一个时刻系统的响应, 最后绘出系统的响应曲线, 由此分析系统的性能。

**例 8-12** 一个典型线性反馈控制系统结构如图 8-12 所示, 图中  $R(s)$  为输入函数,  $Y(s)$  为输出,  $G_C(s)$  为控制器模型,  $G(s)$  为对象模型,  $H(s)$  为反馈模型。各个模块分别为:

$$G_C(s) = \frac{4}{s^3 + 3s^2 + 3s + 4}, \quad G(s) = \frac{s-3}{s+3}, \quad H(s) = \frac{1}{0.01s+1}$$

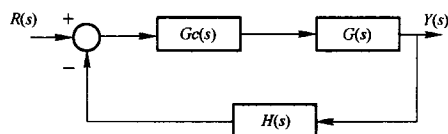


图 8-12 线性反馈控制系统结构

用 Simulink 仿真软件求出开环和闭环系统的阶跃响应曲线。

Simulink 软件仿真步骤如下:

- (1) 打开模块库浏览器, 并新建一个模型窗口。

(2) 在模块库浏览器窗口中双击“Control System Toolbox”图标，即打开控制系统工具箱，并将其中的 LTI 模型拖动到新建的模型窗口中，共需要 3 个（可再拖动两次，也可以复制两个），重新命名后分别作为  $G_c(s)$ 、 $G(s)$  和  $H(s)$  的函数模块。由于  $H(s)$  是反向模块（即表示负反馈），所以需选中该模块后，按快捷键  $\langle \text{Ctrl} \rangle + \langle \text{R} \rangle$  两次以改变其传输方向。

(3) 双击其中的  $G_c(s)$  模块，将出现设置模块参数对话框，将“LTI system variable”一栏中原来系统默认的传递函数修改为  $tf(4,[1,2,3,4])$ ，同样把  $G(s)$  的传递函数修改为  $tf([1,-3],[1,3])$ ， $H(s)$  的传递函数修改为  $tf(1,[0.01,1])$ 。

(4) 在模块库浏览器窗口中，双击 Sources 模块（信号源），将其中的 Step 模块（阶跃信号）拖动到模型窗口；双击 Math 模块（数学运算），将其中的 Sum 模块（求和运算）拖动到模型窗口；双击 Sinks 模块（输出），将其中的 Scope 模块（示波器）拖动到模型窗口。并按图 8-13 连接好系统。

(5) 选择模型窗口“Simulation”菜单中的“Star”命令，即可得到与图 8-14 完全一致的闭环系统阶跃响应曲线。

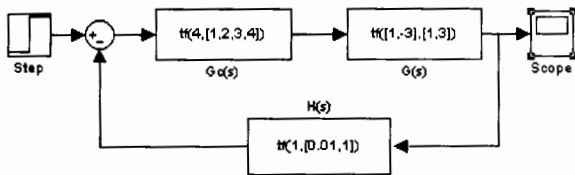


图 8-13 线性反馈控制系统仿真模型

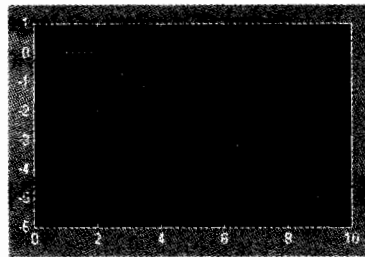


图 8-14 闭环系统的阶跃响应曲线

断开图 8-13 中  $H(s)$  模块左侧或右侧的连线，使其成为开环系统（如图 8-15 所示）。

(6) 再进行仿真，即可得到如图 8-16 的开环系统阶跃响应曲线。

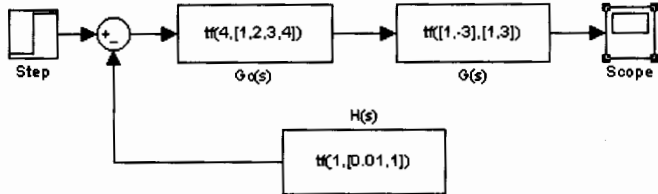


图 8-15 线性反馈控制系统仿真模型

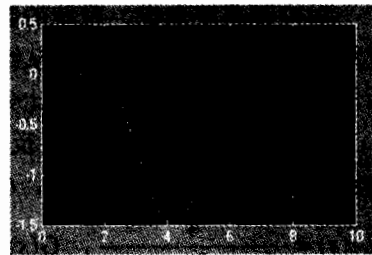


图 8-16 开环系统的阶跃响应曲线

从这个例子可以看出，开环系统是稳定的，而闭环系统是不稳定的。因此，并不是所有的控制器和闭环结构都能够改善原系统的性能，事实上，如果控制器设计不当，则将使闭环系统的特性恶化。

## 8.5.2 自动控制系统

利用 MATLAB 控制系统工具箱提供的仿真函数，同样可以分析自动控制系统。

例 8-13 有一高阶系统：

$$x' = \begin{bmatrix} -1.6 & -0.9 & 0 & 0 \\ 0.9 & 0 & 0 & 0 \\ 0.4 & 0.5 & -5.0 & -2.45 \\ 0 & 0 & 2.45 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} u$$

$$y = [1 \ 1 \ 1 \ 1]x$$

设初始状态为  $x_0 = [1 \ 1 \ 1 \ -1]^T$ ，求这个系统的单位阶跃响应、单位冲激响应及零输入响应（如图 8-17 所示）。

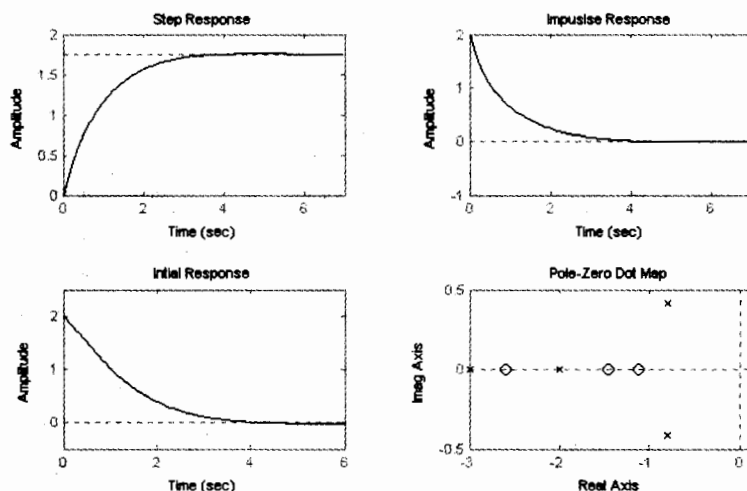


图 8-17 连续系统的响应

程序如下：

```
clear
clc

a=[-1.6,-0.9,0,0;0.9,0,0,0;0.4,0.5,-5.0,-2.45;0,0,2.45,0];
b=[1;0;1;0];
c=[1,1,1,1];
d=[0];                                %以上四条语句为定义系统参数
subplot(221);
step(a,b,c,d);                        %求高阶系统的阶跃响应
title('Step Response');
subplot(222);
impz(a,b,c,d);                        %求高阶系统的冲激响应
title('Impulse Response');
subplot(223);
x0=[1;1;1;-1];                        %初始条件
initial(a,b,c,d,x0);                 %求零输入响应
axis([0 6 -0.5 2.5]);
title('Initial Response');
```

```
subplot(224);
pzmap(a,b,c,d);          %作系统的零点一极点图
title('Pole-Zero Dot Map');
```

从图 8-17 中可以看出该系统是稳定的。

## 8.6 本章小结

本章简单介绍了 MATLAB 数字滤波器的设计、信号处理、图像处理、通信和控制系统中的应用例子。让读者对 MATLAB 的实用价值有个感性的认识。本章的相关函数如下表 8-1 所示。

表 8-1 本章相关函数

函 数 名 称	说 明	函 数 名 称	说 明
fir1	基于窗函数的 FIR (标准响应) 滤波器的设计	buttord	Butterworth 滤波器阶的选择
freqz	数字滤波器频率响应	butter	Butterworth 滤波器设计
fft	一维快速傅立叶变换	dct	一维离散余弦变换
ifft	一维快速傅立叶逆变换	idct	一维离散余弦逆变换
imcrop	剪裁图像	imresize	改变图像大小
imrotate	旋转图像	imnoise	添加噪声
medfilt2	二维中值滤波	wavedec2	多尺度二维小波分解
appcoef2	提取二维小波分解低频系数	wcodemat	对矩阵进行量化编码
length	求向量的长度	blanks	空格字符处理
nextpow2	最靠近 2 的幂次	pzmap	零极点图
hilbert	离散希尔伯特变换	initial	连续时间零输入响应
step	阶跃响应	impulse	冲激响应

## 8.7 习题

1. 什么是数字滤波器？数字滤波器主要分为几类？设计步骤有哪些？
2. 什么是 FFT？
3. 离散余弦变换 (DCT) 为何能用于数据压缩？
4. 图像融合的意义是什么？
5. 利用计算机的声卡输入一段声音信号，用 FFT 分析其频谱？