

# MATLAB 5.3 精要、编程及高级应用

程卫国 冯峰 王雪梅 刘艺 编



机械工业出版社

本书分为三篇。基础篇主要介绍 MATLAB5.3 的基本功能和操作；中级篇使读者更深一步了解应用 MATLAB 进行编程和数值计算的方法，以及动态系统仿真工具 Simulink 的功能；高级篇深入介绍了用 Simulink 进行系统仿真的方法，并着重介绍了最常用的几个工具箱（Toolbox）。

本书从 MATLAB 的基本功能讲起，与具体应用相结合，如信号处理、控制系统设计分析、符号数学、系统仿真等。本书主要读者对象为广大的科技工作者和理工科大学的本科生、研究生。

## 图书在版编目（CIP）数据

MATLAB 5.3 精要、编程及高级应用 / 程卫国等编. —北京：机械工业出版社，2000.4

ISBN 7-111-08012-2

I . M … II . 程 … III . 计算机辅助计算 - 软件包，MATLAB 5.3 IV . TP391.75

中国版本图书馆 CIP 数据核字(2000)第 06539 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑：武 江 封面设计：姚 燕

责任印制：路 琳

北京市密云县印刷厂印刷 · 新华书店北京发行所发行

2000 年 4 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 29.5 印张 · 718 千字

0 001—4000 册

定价：46.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话 (010) 68993821、68326677-2527

# 前　　言

随着计算机在电力、航空、机械、汽车、制造等各个行业的深入广泛地应用，及微型机的日渐普及，用计算机解决数学模型成为了一个对各行业的革命性话题，这场革命的方向是高度自动化、智能化、科学规划化、集成化。比如电力系统的优化，科学规划，以及对电网潮流、稳定性的计算。一个现代化的城市电网即使简化处理，用手工计算也基本上不大可能，计算机的出现使这成为可能，并且也为对电网的安全、即时监控也提供了条件，而由 MathWork 公司推出的 MATLAB，使得为解决这些具体问题而建立数学模型变得轻松、便捷，为科学和工程技术人员节省了宝贵的精力，并赢得时间。清华大学数学科学系一位著名教授说：“以前科学工作者和工程技术人员往往为了繁琐的计算耗费大量时间，比如要研究一个函数的性质，写出它的泰勒级数、傅里叶级数等等，用一周甚至更长的时间都在进行无谓的计算，现在有了计算机和 MATLAB 等数学软件，几条命令就可以分析清它的连续性，找出零点、极点、驻点等等，手脚慢点没关系，计算机快，10 分钟顶多 20 分钟完事。”是的，计算机的出现引发了一场科学和技术革命，而 MATLAB 以它强大的数值计算功能和简捷易学的语句、函数获得了广大科学工作者和工程技术人员的一致认可，并被一些名家誉为第四代编程语言。现在，该软件已经在国外的许多大学普及，在清华大学也已经成为大部分系的必修课或者必备知识。现在我们有幸编著这本书，希望这一书能为在我国高校和科学工作者中推广 MATLAB 软件产生积极深远的影响。

## 本书的两大特点：

### 一、循序渐进

本书分为三篇，第一篇基础篇，主要介绍 MATLAB5.3 的基本功能和操作，第二篇中级篇，使读者更深一步了解应用 MATLAB 进行编程和数值计算的方法，以及动态系统仿真工具 Simulink 的功能；第三篇高级篇，深入介绍了用 Simulink 进行系统仿真的方法，并着重介绍了最常用的几个工具箱（Toolbox）。读完这一篇，相信一些读者会有如释重负的感觉，因为他们以前需要一周甚至更长时间耗在那些繁琐的计算上，现在只需要几条命令，用几分钟即可！如果果真如此，我们也如释重负了，读者对本书和对 MATLAB5.3 的认可与工作效率的提高，将是对我们几位作者长时间挑灯夜战的最好奖赏。

### 二、深入浅出

在编著本书时，我们考虑到读者不同的知识和专业背景，在介绍 MATLAB5.3 命令的同时，也介绍了这些运算和函数的数学背景，还有一些介绍了它的工程技术背景，深入剖析原理，浅出提供操作指令，一气呵成，既是一本带有科普性质的数学知识读本，又是指导实际操作和深入学习的计算机教材，还可以作为高校数学模型、数学实验、数字信号处

理、控制系统设计与计算机应用课程的参考资料。

参与本书编写的人员还包括：姚东、徐昕、赵宇明、梁辉、苏海东、高子军、王延国、国胜军、果永振、赵春立、金燕、田华、杨德永、巩晓君、缪银昌、毛银杰、王军、康长楠。

作 者

2000年3月

# 目 录

前言

## 基础篇

<b>第1章 概论</b>	1
1.1 MATLAB简介	1
1.1.1 MATLAB的广泛应用	1
1.1.2 MATLAB软件系统的构成	1
1.1.3 MATLAB各个版本的特点	3
1.2 MATLAB的安装	5
1.2.1 MATLAB对硬件和软件的要求	5
1.2.2 安装过程	6
1.2.3 MATLAB的启动和退出	10
1.2.4 卸载 MATLAB	11
1.3 MATLAB的目录结构	11
1.4 MATLAB应用入门	14
1.4.1 命令窗口的菜单系统	14
1.4.2 命令窗口的工具栏	17
1.4.3 矩阵和常用命令的输入	17
1.4.4 图形窗口	19
1.5 使用 MATLAB帮助	20
1.5.1 help命令	21
1.5.2 演示和漫游	22
1.5.3 其他帮助命令	24
1.5.4 帮助窗口(MATLAB Help Window)	27
1.5.5 Help Desk	28
<b>第2章 基本数值运算</b>	31
2.1 矩阵的生成	31
2.1.1 简单矩阵的生成	31
2.1.2 常用矩阵的生成	34
2.1.3 特殊矩阵的生成	35
2.1.4 向量的生成	36
2.2 数值运算基本函数及其应用	37
2.2.1 加, 减, 乘, 除, 乘方	37

2.2.2 三角和超越函数 .....	42
2.2.3 指数和对数函数 .....	43
2.2.4 复数函数 .....	43
2.2.5 数值处理 .....	43
2.3 矩阵变换 .....	43
2.3.1 矩阵旋转 .....	44
2.3.2 矩阵的产生或提取 .....	45
2.4 输出 .....	47
2.4.1 输出格式 .....	47
2.4.2 特殊变量和常数 .....	48
<b>第3章 数据可视化 .....</b>	<b>49</b>
3.1 基本平面图形 .....	49
3.1.1 怎么作图 .....	49
3.1.2 图形的标注 .....	55
3.2 特殊平面图形 .....	60
3.2.1 条形图 .....	61
3.2.2 复数向量图 .....	63
3.2.3 直方图 .....	64
3.2.4 极坐标曲线图 .....	65
3.2.5 扇形图 .....	66
3.3 三维图形 .....	68
3.3.1 基本三维图 .....	68
3.3.2 线、面填色 .....	69
3.3.3 三维数据的等高线和其二维表现 .....	70
3.3.4 曲面与网线图 .....	72
3.3.5 图的表现 .....	75
3.3.6 其他三维图形 .....	86
<b>第4章 工作环境管理 .....</b>	<b>88</b>
4.1 搜索路径管理 .....	88
4.1.1 用户目录的建立 .....	88
4.1.2 搜索文件的顺序 .....	89
4.2 工作空间管理 .....	89
4.2.1 工作空间浏览器 .....	89
4.2.2 保存和载入 MATLAB 工作空间的内容 .....	90
4.2.3 保存和载入变量 .....	90
4.3 命令窗口管理 .....	91
4.3.1 环境参数设置 .....	91
4.3.2 执行外部应用程序 .....	93
4.3.3 命令窗口的分页输出 .....	93

## 中 级 篇

<b>第 5 章 基本编程.....</b>	<b>94</b>
5.1 变量、语句 .....	94
5.1.1 变量类型 .....	94
5.1.2 基本语句 .....	94
5.2 数据类型 .....	95
5.2.1 字符 .....	96
5.2.2 结构 .....	99
5.2.3 单元数组 .....	101
5.3 程序控制语句 .....	105
5.3.1 循环语句 .....	105
5.3.2 条件转移语句 .....	107
5.4 MATLAB 函数 .....	109
5.4.1 函数 .....	110
5.4.2 子函数 .....	115
5.4.3 函数的执行 .....	116
<b>第 6 章 数据分析.....</b>	<b>117</b>
6.1 线性方程组 .....	117
6.1.1 线性方程求解 .....	117
6.1.2 矩阵分解 .....	118
6.1.3 矩阵特征值与特征向量 .....	123
6.1.4 其他矩阵函数 .....	125
6.2 非线性数值计算 .....	132
6.2.1 非线性函数最小值点 .....	132
6.2.2 单变量函数零点 .....	135
6.2.3 绘制函数曲线 .....	135
6.2.4 常微分方程（组）数值解 .....	137
6.2.5 函数的数值积分 .....	139
6.3 多项式 .....	140
6.3.1 多项式表示法 .....	140
6.3.2 求多项式的根，由根创建多项式 .....	141
6.3.3 多项式乘和除 .....	141
6.3.4 多项式导数 .....	142
6.3.5 多项式的值 .....	143
6.3.6 多项式曲线拟合 .....	144
6.3.7 部分分式展开 .....	144
6.4 插值 .....	145
6.4.1 一维插值 .....	145

6.4.2 二维插值(interp2) .....	147
6.4.3 三次样条 .....	148
6.5 数据分析和傅立叶变换 .....	151
6.5.1 基础运算 .....	151
6.5.2 有限差分 .....	153
6.5.3 向量运算 .....	154
6.5.4 协方差阵和相关阵 .....	155
6.5.5 傅立叶变换初步 .....	156
6.6 稀疏矩阵 .....	158
6.6.1 稀疏矩阵的存储 .....	158
6.6.2 创建稀疏矩阵 .....	159
6.6.3 稀疏矩阵的查看 .....	161
6.6.4 稀疏矩阵的运算 .....	163
<b>第 7 章 编程进阶</b> .....	<b>169</b>
7.1 句柄图形 .....	169
7.1.1 句柄图形的结构层次 .....	169
7.1.2 访问对象句柄 .....	170
7.1.3 图形对象的属性和设置 .....	174
7.2 图形对象属性编辑器 .....	177
7.2.1 图形窗口的交互操作方式 .....	177
7.2.2 图形属性编辑器 .....	180
7.3 GUI 设计向导 .....	183
7.3.1 GUI 设计向导控制面板 .....	183
7.3.2 利用向导设计菜单 .....	184
7.3.3 利用向导设计控件 .....	189
7.4 编程设计 GUI .....	198
7.4.1 编程建立菜单 .....	198
7.4.2 编程序建立控件 .....	202
7.5 对话框 .....	206
7.5.1 专用对话框的设计 .....	207
7.5.2 标准对话框 .....	212
7.6 低级文件 I/O .....	216
7.6.1 打开和关闭文件 .....	216
7.6.2 读写二进制数据 .....	218
7.6.3 有格式文件 .....	220
7.6.4 文件位置指针 .....	224
<b>第 8 章 Simulink 入门</b> .....	<b>226</b>
8.1 Simulink 概述 .....	226
8.2 Simulink 基本操作 .....	226

8.2.1 运行 Simulink .....	226
8.2.2 Simulink 模块的操作 .....	227
8.2.3 模块的连接 .....	230
8.2.4 在连线上反映信息 .....	231
8.3 Simulink 的几类基本模块 .....	232

## 高 级 篇

<b>第 9 章 信号处理工具箱 .....</b>	<b>235</b>
9.1 波形产生 .....	235
9.1.1 常用周期波形 .....	235
9.1.2 Sinc 函数和 Dirichlet 函数 .....	236
9.1.3 脉冲信号 .....	238
9.1.4 扫频信号 .....	240
9.2 线性系统模型 .....	241
9.2.1 离散时间系统模型 .....	241
9.2.2 连续时间系统模型 .....	244
9.2.3 线性系统变换 .....	244
9.3 信号变换 .....	245
9.3.1 Chirp z 变换 .....	245
9.3.2 离散余弦变换 (DCT) .....	245
9.3.3 Hilbert 变换 .....	247
9.4 数字滤波器的应用与分析 .....	247
9.4.1 数字滤波的应用 .....	247
9.4.2 滤波器的分析 .....	250
9.5 滤波器设计 .....	253
9.5.1 IIR 滤波器设计 .....	253
9.5.2 FIR 滤波器设计 .....	257
9.6 统计信号处理和谱分析 .....	263
9.6.1 相关和协方差 .....	263
9.6.2 谱分析 .....	265
9.6.3 Welch 方法及函数 .....	265
9.6.4 其他 PSD 估计方法 .....	269
9.7 窗函数 .....	273
9.7.1 矩形窗、巴特利特窗、三角窗 .....	273
9.7.2 广义余弦窗 .....	273
9.7.3 凯塞窗 .....	274
9.7.4 切比雪夫窗 .....	274
9.8 交互工具 .....	275
9.8.1 SPTool 主窗口 .....	276

9.8.2 信号浏览器 .....	277
9.8.3 滤波器观察器 .....	278
9.8.4 滤波器设计器 .....	279
9.8.5 谱观察器 .....	280
<b>第 10 章 控制系统工具箱 .....</b>	<b>283</b>
10.1 LTI 模型 .....	283
10.1.1 建立 LTI 模型 .....	283
10.1.2 LTI 模型的属性 .....	289
10.1.3 模型转换 .....	292
10.2 模型的运算 .....	293
10.2.1 算术运算 .....	293
10.2.2 模型的连接 .....	295
10.2.3 连续模型和离散模型的相互转换 .....	298
10.3 模型分析 .....	300
10.3.1 模型通用特征 .....	300
10.3.2 模型动态特性 .....	301
10.3.3 状态空间实现 .....	302
10.3.4 时间响应 .....	305
10.3.5 频率响应 .....	307
10.3.6 模型降阶 .....	309
10.4 LTI Viewer .....	311
10.4.1 菜单 .....	311
10.4.2 应用举例和上下文菜单 .....	313
10.5 控制系统设计 .....	315
10.5.1 根轨迹法 .....	316
10.5.2 极点配置 .....	317
10.5.3 LQG 设计 .....	319
10.6 根轨迹设计工具 .....	322
10.6.1 窗口功能 .....	322
10.6.2 输入输出模型 .....	323
10.6.3 设置网格和边界线 .....	325
10.6.4 转换为 Simulink 框图 .....	325
<b>第 11 章 系统仿真 .....</b>	<b>326</b>
11.1 系统仿真及参数设置 .....	326
11.1.1 Solver 的设置 .....	326
11.1.2 设置 Workspace I/O Page .....	328
11.1.3 Diagnostics (诊断) 参数设置 .....	332
11.1.4 在命令窗口输入命令进行仿真 .....	333
11.2 子系统的建立和封装 .....	337

11.2.1 子系统的建立 .....	337
11.2.2 条件执行子系统 .....	338
11.2.3 子系统的封装（Masking） .....	342
11.3 构造仿真模型的命令和参数 .....	348
11.3.1 构造模型的命令 .....	348
11.3.2 设置参数的命令 set_param .....	349
11.3.3 参数设置 .....	349
11.4 Real-Time Workshop 简介 .....	350
11.4.1 Real-Time Workshop 简介 .....	350
11.4.2 Real-Time Workshop 的几个基本概念 .....	351
11.5 Real-Time Workshop 的设置 .....	352
11.5.1 System Target File（系统目标文件） .....	352
11.5.2 内联参数和可调参数 .....	354
11.5.3 模板 make 文件（Template Makefile） .....	356
11.5.4 make 命令 .....	359
11.5.5 Options 按钮 .....	359
11.5.6 文件拆分和函数拆分 .....	360
11.6 外部模式（External Mode） .....	362
11.6.1 Target Interface（目标连接）对话框 .....	363
11.6.2 Signal & Triggering（信号和触发）对话框 .....	363
11.6.3 Data Archiving（数据存档）对话框 .....	365
11.7 Real-Time Workshop 函数库 .....	366
11.7.1 Real-Time Workshop 函数库概述 .....	366
11.7.2 自定义代码库（Custom Code Library） .....	368
<b>第 12 章 符号数学工具箱 .....</b>	<b>374</b>
12.1 建立符号对象 .....	374
12.1.1 建立符号变量、表达式和矩阵 .....	374
12.1.2 把数值标量或矩阵转换为符号形式 .....	376
12.1.3 建立符号数学函数 .....	378
12.2 因式分解和替换 .....	379
12.2.1 因式分解和展开 .....	379
12.2.2 简化 .....	382
12.2.3 替换 .....	384
12.3 符号微积分 .....	387
12.3.1 符号自变量的确定 .....	387
12.3.2 微分 .....	388
12.3.3 极限 .....	389
12.3.4 积分 .....	390
12.3.5 符号求和 .....	391

12.3.6 泰勒级数	392
12.4 线性代数	392
12.4.1 基本算术运算	392
12.4.2 线性代数运算	394
12.4.3 特征值和特征向量	397
12.4.4 约当标准型	398
12.4.5 奇异值分解	399
12.5 方程求解	399
12.5.1 代数方程	399
12.5.2 代数方程组	400
12.5.3 微分方程	402
12.5.4 微分方程组	403
12.6 积分变换	403
12.6.1 傅立叶变换和傅立叶逆变换	403
12.6.2 拉普拉斯变换和拉普拉斯逆变换	404
12.6.3 z 变换和逆 z 变换	404
12.7 符号函数的图形	405
12.7.1 绘制符号函数的图形	405
12.7.2 可视化函数计算器	407
12.8 访问 Maple 函数	408
12.8.1 访问 Maple 中的函数	408
12.8.2 特殊数学函数	410
12.9 扩展符号数学工具箱	411
12.9.1 Maple 软件包	411
12.9.2 Maple 程序	413
附录 A MATLAB 函数命令索引	416
附录 B 图形对象属性	422
附录 C Simulink 模型模块参数	450

# 第1章 概论

## 1.1 MATLAB 简介

### 1.1.1 MATLAB 的广泛应用

具有 Fortran 语言和 C 语言编程经验的读者可能有这样的体会，当涉及到矩阵运算或画图时，编程会很麻烦。例如，想要求解一个线性代数方程组，得首先编写一个主程序，然后编写一个子程序去读入各个矩阵的元素，之后再编写一个子程序，求解相应的方程，最后输出计算结果。如果计算子程序不是很可靠，则所得的计算结果往往可能会出现问题。如果没有标准的子程序可以调用，则用户往往要将自己编好的子程序逐条地敲入计算机，然后进行调试，最后进行计算。这样一个简单的问题往往需要用户编写 100 条左右的源程序，仅键入和调试就是很繁琐的，而且还无法保证所键入的程序一次就全部可靠。

1980 年前后，MATLAB 的首创者 Cleve Moler 博士在 New Mexico 大学讲授线性代数课程时，看到了用高级语言编程解决工程计算问题的诸多不便，因而构思开发了 MATLAB 软件 (MATrix LABoratory, 矩阵实验室)，该软件利用了 Moler 博士在此前开发的 LINPACK (线性代数软件包) 和 EISPACK (基于特征值计算的软件包) 中可靠的子程序，用 fortran 语言编写而成，集命令翻译、工程计算功能于一身。

与 Fortran 和 C 等高级语言比较，MATLAB 的语法规则更简单，更重要的是其贴近人思维方式的编程特点，使得用 MATLAB 编写程序有如在便笺上列公式和求解。

80 年代初期，Cleve Moler 和 John Little 采用 C 语言改写了 MATLAB 的内核。不久，他们成立了 Mathworks 软件开发公司并将 MATLAB 正式推向市场。

现在的 MATLAB 新版本早已不只停留在工程计算的功能上了，它由主包、Simulink 以及功能各异的工具箱组成，以矩阵运算为基础，把计算、可视化、程序设计融合到了一个简单易用的交互式工作环境中。在这里可以实现工程计算、算法研究、符号运算、建模和仿真、原型开发、数据分析及可视化、科学和工程绘图、应用程序设计（包括图形用户界面设计）等等功能。

正是凭借 MATLAB 的这些突出的优势，它现在已成为世界上应用最广泛的工程计算软件。在美国等发达国家的大学里 MATLAB 是一种必须掌握的基本工具，而在国外的研究设计单位和工业部门，更是研究和解决工程计算问题的一种标准软件。在国内也有越来越多的科学技术工作者参加到学习和倡导这门语言的行列中来。在大家的共同努力下，MATLAB 正在成为计算机应用软件中的一个新热点。

### 1.1.2 MATLAB 软件系统的构成

MATLAB 软件主要由主包、Simulink 和工具箱三大部分组成。其结构如图 1-1 所示。

## 1. MATLAB 主包

MATLAB 主包包括以下五个部分：

### (1) MATLAB 语言

MATLAB 语言是一种基于矩阵/数组的高级语言，它具有流程控制语句、函数、数据结构、输入输出，并且具有面向对象的程序设计特性。用 MATLAB 语言可以迅速地建立临时性的小程序，也可以建立复杂的大型应用程序。

### (2) MATLAB 工作环境

MATLAB 工作环境集成了许多工具和程序，用户用工作环境中提供的功能完成他们的工作。

MATLAB 工作环境给用户提供了管理工作空间内的变量和输入、输出数据的功能，并给用户提供了不同的工具用以开发、管理、调试 M 文件和 MATLAB 应用程序。

### (3) 句柄图形

句柄图形是 MATLAB 的图形系统。它包括一些高级命令，用于实现二维和三维数据可视化、图像处理、动画等功能；还有一些低级命令，用来定制图形的显示以及建立 MATLAB 应用程序的图形用户界面。

### (4) MATLAB 数学函数库

MATLAB 数学函数库是数学算法的一个巨大集合，该函数库既包含了诸如求和、正弦、余弦、复数运算之类的简单函数；也包含了矩阵转置、特征值、贝塞尔函数、快速傅立叶变换等等复杂函数。

### (5) MATLAB 应用程序接口 (API)

MATLAB 应用程序接口是一个 MATLAB 语言同 C 和 Fortran 等其它高级语言进行交互的库。包括从 MATLAB 调用其它程序（动态链接），把 MATLAB 作为计算引擎来调用，还包括读写 MATLAB 数据文件 (MAT 文件)。

## 2. Simulink

Simulink 是用于动态系统仿真的交互式系统。Simulink 允许用户在屏幕上绘制框图来模拟一个系统，并能够动态地控制该系统。Simulink 采用鼠标驱动方式，能够处理线性、非线性、连续、离散、多变量以及多级系统。

此外，Simulink 还为用户提供了两个附加项：Simulink Extensions (扩展) 和 Blocksets

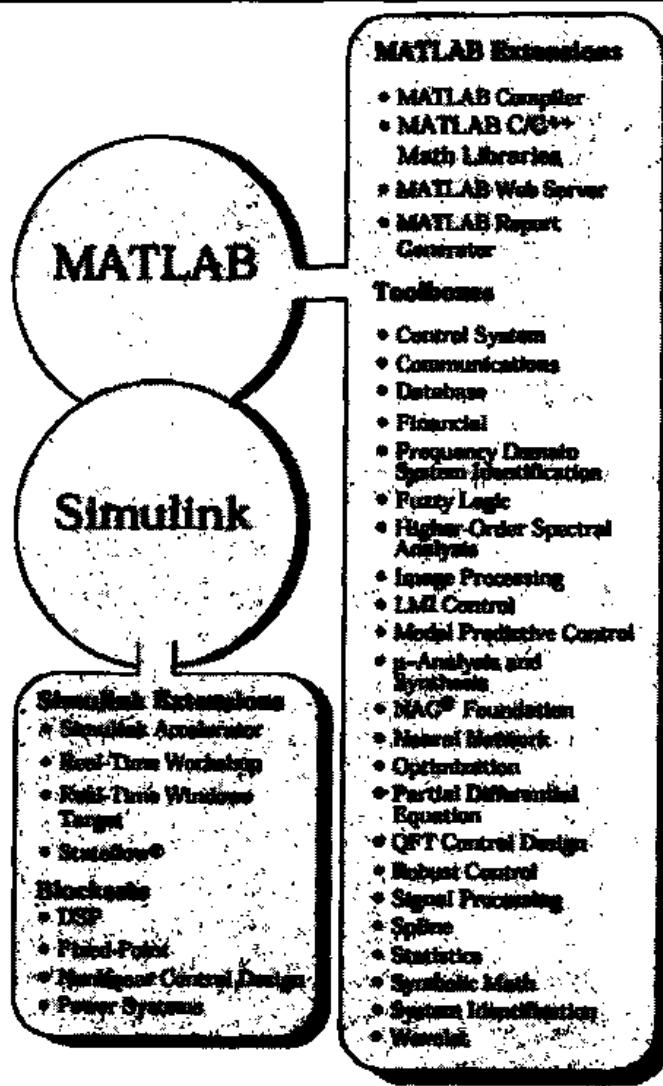


图 1-1 MATLAB 软件系统结构

(模块集)。

Simulink Extensions 是一些可选择的工具，支持在 Simulink 环境中开发的系统的具体实现，包括：

- Simulink Accelerator
- Real-Time Workshop
- Real-Time Windows Target
- Stateflow

Blocksets 是为特殊应用领域中设计的 Simulink 模块的集合。Blocksets 包括以下几个领域的模块集：

- DSP(数字信号处理)
- Fixed-Point(定点)
- Nonlinear Control Design(非线性控制设计)
- Communications(通信)

### 3. MATLAB 工具箱

工具箱是 MATLAB 用来解决各个领域特定问题的函数库，它是开放式的，可以应用，也可以根据自己的需要进行扩展。

MATLAB 提供的工具箱为用户提供了丰富而实用的资源，工具箱的内容非常广泛，涵盖了科学的研究的很多门类。目前，已有涉及数学、控制、通信、信号处理、图像处理、经济、地理等多种学科的二十多种 MATLAB 工具箱投入应用。这些工具箱的作者都是相关领域的顶级专家，这当然地确定了其权威性。应用 MATLAB 的各种工具箱可以在很大程度上减小用户编程时的复杂度。而 Mathworks 公司也一直致力于追踪各学科的最新进展，并及时推出相应功能的工具箱。毫无疑问，MATLAB 能在数学应用软件中成为主流是离不开各种功能强大的工具箱的。

#### 1.1.3 MATLAB 各个版本的特点

Mathworks 公司自 1984 年正式推出 MATLAB 后，经过这些年的不断更新，交互性越来越好，功能越来越强大，已经成为国际公认的最优秀的数学应用软件之一。

MATLAB 软件从 1984 年推出的一个版本到目前已经发布了 11 个版本，第 11 个版本是 MATLAB 5.3 (Release 11)。

在这些版本中，具有划时代意义的是 Mathworks 公司于 1992 年推出的 MATLAB 4.0 版，其微机版于 1993 年完成。从该版本开始，MATLAB 由 DOS 下的应用程序改进为 Windows 下的应用程序，更易于操作，从而大大拓展了其应用范围，并加速了该软件的更新过程。同时，在这个版本中，Mathworks 公司推出了用于控制系统仿真和设计的交互式模型输入与仿真环境 simulink 1.0，为控制系统的计算机辅助设计打开了崭新的局面。1994 年推出的 4.2 版本扩充了 4.0 版本的功能，尤其在图形界面设计方面提供了新的方法。

MATLAB 在国内的大范围推广就是从 4.x 版本开始的。此后的 MATLAB 5.0、5.1、5.2 到 1999 年 1 月推出的 5.3 版本，分别在前面版本的基础上前进了一大步。这里我们把各个版本的新增特性以表格的形式加以比较，以便读者对 MATLAB 的发展过程有一个概要的认识。如表 1-1 所示。

表 1-1 几个 MATLAB 版本比较

	5.0	5.1	5.2 (Release 10)	5.3 (Release 11)
语言和开发环境	完整的 M 文件编辑器 可视化的 M 文件调试器 建立 M 文件执行简档的工具 搜索路径浏览器/编辑器 工作空间浏览器 基于网络的在线帮助 P 代码文件 面向对象的程序设计 多函数和私有的 M 文件 增强的 API	函数 find 当找不到指定内容时返回一个空矩阵，在以前版本中返回[0,1]; 支持多字节字符 不再需要 Windows TCP/IP 支持 Notebook 可以支持 Office 97 更新了 PC 机编辑器/调试器工具栏上的图标	try/catch 错误处理 M 文件锁定技术 改进了 Help Desk，以很快的搜索引擎支持全文搜索 支持两个新的 ActiveX 技术：ActiveX 控件容器和 ActiveX 自动化客户功能 支持 HDF (分级数据格式) 文件	简化了安装过程 文件 I/O 功能增强 开发工具增强 在线帮助文件增强 日文界面
新的数据类型	多维数组 用户自定义的数据结构 单元数组——包含多种数据类型的数组 字符串数组 用于图像的单字节数据类型 可变长度的参数列表 函数和操作符重载 switch/case 句型			支持整数数据类型
可视化功能	GUI 构造器 句柄图形属性编辑器 支持真彩色 (RGB) Z 缓冲区算法 Flat、Gouraud 和 Phong 三种光照方式 三维造型的量化补片 模拟照相机视图 高效的 8 位图像显示 输入/输出图像文件 用希腊字符、上下标、多行文本做标注 采用多坐标轴 可绘制饼形图、三维直方图 扩充了曲线标记符号	分散绘图函数 函数 uisetcolor 支持 X-Windows 改进了补片 (patch) 和表面的打印 增加了 tiff 和 jpeg 格式的设备驱动程序 内嵌的 PostScript (EPS) 可以包含 TIFF 预览图像	支持 OpenGL 透视图，显著改善了很多可视化程序的性能； 改进了照相机控制、简化了光源的定位； 新增的打印画面编辑器可以让用户打印 Simulink 模型框图时自己进行设计； 能够基于缺省的系统字体大小来确定用户界面的位置和大小，并可以在 GUI 中添加工具提示、触发按钮、上下文菜单等	改进了图形窗口 新的绘图编辑器 新的绘图和三维可视化函数 支持 HDF 文件的开发工具 支持 PNG (Portable Network Graphics, 可移植的网络图像文件格式) 图像
数学工具	新的 ODE 解法 德洛内三角剖分算法 对不规则采样数据采用网格内插法处理 集合论函数 二维积分 时间和日期的处理函数 多维插值、卷积、FFT 位运算 稀疏矩阵特征值和奇异值		增加 ODE (常微分方程) 函数	稀疏矩阵处理能力增强 数值分析能力增强

(续)

工具箱	图像处理工具箱 2.0 控制系统工具箱 4.0 信号处理工具箱 4.0 最优化工具箱 2.0		通信工具箱 1.3 控制系统工具箱 4.1 DSP 模块集 2.2 金融工具箱 1.1 模糊逻辑工具箱 2.0 图像处理工具箱 2.1 神经网络工具箱 3.0 信号处理工具箱 4.1 样条工具箱 2.0	通信工具箱 1.4 控制系统工具箱 4.2 DSP 模块集 3.0 金融工具箱 2.0 固定点模块集 2.0 图像处理工具箱 2.2 地图绘制工具箱 1.1 Excel 连接 1.0.8 最优化工具箱 2.0 动力系统模块集 1.1 信号处理工具箱 4.2 统计工具箱 2.2 符号数学工具箱 2.1
	Simulink 2.0		2.2 版  支持 2 级 S 函数 新的用户界面特性 新增了仿真功能 一些新的模块和命令	3.0 版  改进了 GUI，包括新的库浏览器和模型浏览器、可以改变观察框图的焦距、在句柄图像窗口中再现保存的数据的新工具  几个新的增强的模块 增强了建模功能 增强了仿真功能 SB2SL 2.0
			2.2 版  支持中断服务程序 可以为目标系统定制中断服务程序 MIMO 的 S 函数 运行时进行参数检测	3.0  生成高质量的嵌入式代码 增强了外部模式 完全兼容 Simulink 中的数据类型
		Stateflow 1.0 地图绘制工具箱 1.0 固定点 (Fixed-Point) 模块集 1.0	动力系统模块集 (Power System Blocksets)	MATLAB 报告发生器 Simulink 报告发生器 Real-Time Windows Target

## 1.2 MATLAB 的安装

### 1.2.1 MATLAB 对硬件和软件的要求

MATLAB 可以安装到下列各种类型的机器上：PC 及兼容机、Macintosh 机、Sun 工作站、VAX 机、HP 工作站、Apollo 工作站、DEC 工作站、SGI 工作站、RS/6000 工作站、Convex 工作站及 Cray 计算机等，而且无论是单机还是网络环境都可发挥其卓越的性能。如果单纯地使用 MATLAB 语言进行编程，不连接其它外部语言，则用 MATLAB 语言编写出来的程序可以不做任何修改直接移植到其它机型上去使用，所以 MATLAB 和其它语言不同，它是

和机器类型无关的，即使去设计图形绘制或图形界面编辑这样往往依赖于机器类型的程序在 MATLAB 下也可以简单地将程序拷贝到其它机器上，而不必去担心它是否能够在新的机器环境中正确运行。

本书中仅讨论在 PC 机环境下 MATLAB 的应用。

MATLAB5.3 对 PC 机的系统要求如下：

- 操作系统：Microsoft Windows95, Windows98 或 Windows NT 4.0(带 Service Pack 3)
- Intel 486 以上 CPU，建议使用奔腾处理器
- 光驱（用来安装 MATLAB）
- 最小 16MB 内存，在 Windows NT 下强烈建议使用 24MB 以上内存
- 8 位以上显卡
- 需要的硬盘空间和硬盘的分区格式有关：
  - 如果是 FAT32 分区，只安装 MATLAB 需要 30MB，安装 MATLAB 在线帮助需要 70MB，完全安装共需要约 570MB；
  - 如果是 FAT16 分区，需要的硬盘空间大约是 FAT32 分区的一倍，完全安装需要约 1.2GB 的硬盘空间。
- 建议安装的其它设备：
  - 图形加速卡
  - 打印机
  - 声卡
- 需要安装 Netscape Navigator 3.0 或更高版本或 Internet Explorer 4.0，用来运行 MATLAB Help Desk。
- 需要安装 Adobe Acrobat Reader 来阅读和打印 MATLAB 中 PDF 格式的在线帮助文档。
- 需要安装 Microsoft Word 7.0 (Office 95) 或 8.0 (Office 97) 来运行 MATLAB Notebook。
- 如果要构造自己的 MEX 文件，则需要下列的至少一种产品：
  - DEC Visual Fortran 5.0
  - Microsoft Visual C/C++ 4.2 或 5.0
  - Borland C/C++ 5.0 或 5.02
  - WATCOM 10.6 或 11

### 1.2.2 安装过程

随着 MATLAB 版本的更新，安装也越来越简便。对于 MATLAB 5.3，用户只要按照安装界面的提示逐步进行。

将 MATLAB 5.3 光盘放入光驱，在 MATLAB 目录下直接运行“Setup.exe”程序，显示如图 1-2 所示的初始画面，随之同屏显示安装准备进度条。

随后出现“Welcome to MATLAB Setup”（欢迎安装 MATLAB）对话框，如图 1-3 所示。选择“Next”继续安装，“Cancel”中断安装。

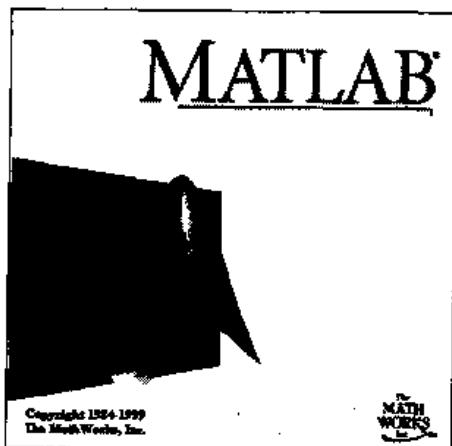


图 1-2 安装初始画面

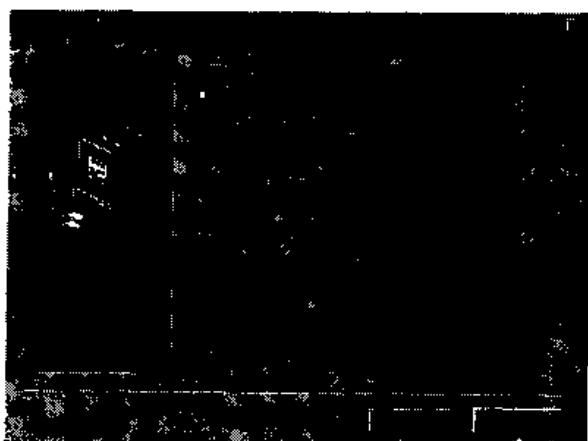


图 1-3 欢迎安装对话框

下一步出现的是“Software License Agreement”（软件许可协议）对话框，如图 1-4 所示，选择“No”不接受协议，中断安装，“Back”返回上一步，“Yes”接受协议内容，出现“Customer Information”（用户信息）对话框，如图 1-5。

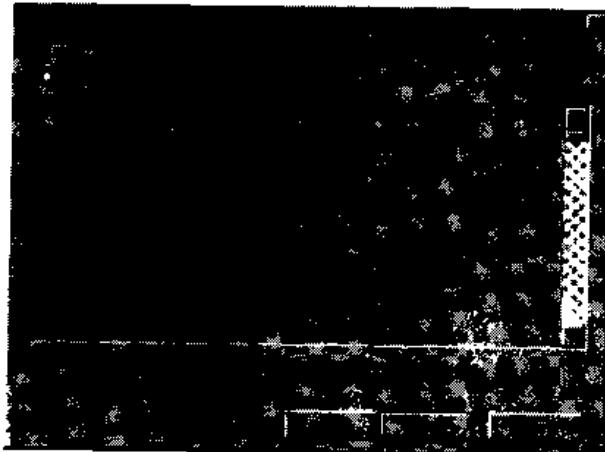


图 1-4 软件许可协议

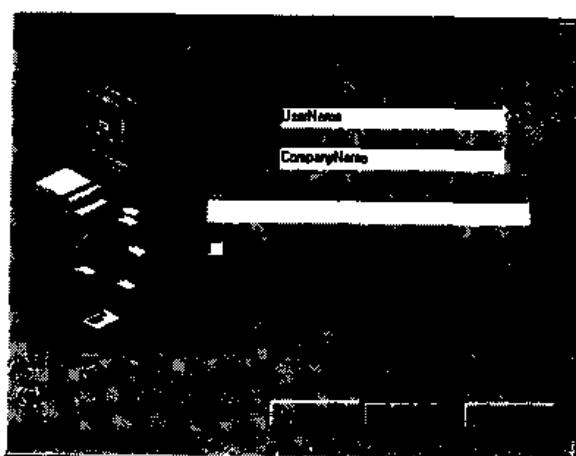


图 1-5 用户信息设置

在“Name”和“Company”中分别输入自己的姓名和公司名字，在下面的空格中输入安装密码，选择“Next”，如果密码输入正确，安装过程将继续，这时出现如图 1-6 所示的“Select MATLAB Components”（选择安装组件）对话框。

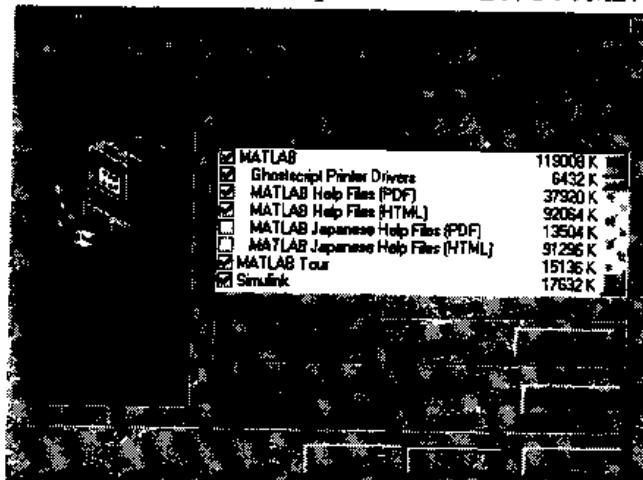


图 1-6 选择要安装的 MATLAB 组件

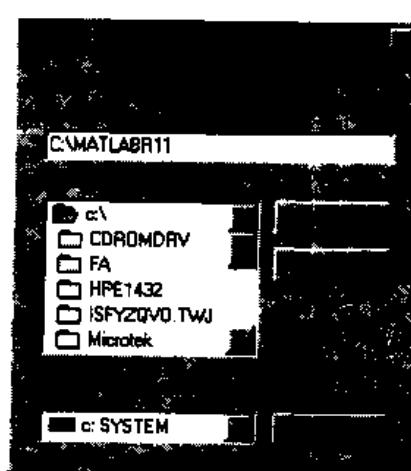


图 1-7 选择安装路径

在选择安装组件对话框中，显示 MATLAB 的组件、安装目录及安装所需要的空间。用户可以通过选中组件前的复选框来选择要安装的 MATLAB 组件，注意在 Windows 95 或 98 上安装时，不能选择“MATLAB Web Server”组件，该组件只能在 Windows NT 平台上使用。

MATLAB 的安装程序将显示安装 MATLAB 时所需的空间并计算用户硬盘的可用空间，安装所需空间与选择的组件数目、硬盘分区方式有关。对于 FAT16 分区，完全安装 Release 11 (MATLAB 5.3) 中除日文部分和“MATLAB Web Server”外的所有组件和帮助，需要约 1.3GB 的硬盘空间。这里讲述的安装就是以这种情况为例。

单击“Browse”按钮，浏览安装 MATLAB 的目录，打开如图 1-7 所示的“Choose Directory”(选择目录) 对话框，用户可以在该对话框中指定 MATLAB 的安装目录。

用户不但可以在本地计算机上安装 MATLAB，也可以将 MATLAB 安装到网络上指定的计算机上。在图 1-7 的对话框中单击“Network”按钮，将打开如图 1-8 所示的对话框。

窗口中的“驱动器”栏指定用于与网络计算机相连接的驱动器号。Windows 将自动选定一个可用的驱动器号。要指定不同的驱动器号，请直接键入驱动器号或从列表中选择某一个。“路径”栏用于指定希望连接到的共享文件夹路径。在大多数情况下，用户可以按照如下格式键入路径：“\计算机名\目录名”。如果用户要选定最近连接到的共享文件夹，请在该栏右侧的列表中单击并且选择一个文件夹。

单击“确定”按钮返回“Choose Directory”对话框。单击“Next”按钮

在为 MATLAB 指定安装路径后，单击“OK”按钮，将返回“Select MATLAB Components”对话框，单击“Next”，MATLAB 将开始安装，安装进程画面如图 1-9。



图 1-8 网络安装



图 1-9 安装进程

当安装即将完成时，将自动弹出如图 1-10 所示的对话框，提示用户安装完成后配置

Real-Time Windows Target 的方法（启动 MATLAB，在命令窗口输入命令“rtwintgt -setup”）。

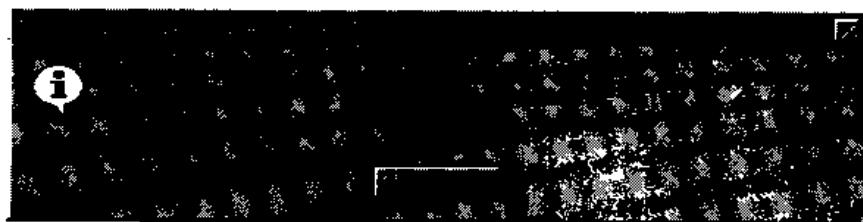


图 1-10 配置 Real-Time Windows Target 的提示

如果用户在安装时选择了需要 Java 支持的组件，如 Database（数据库）工具箱，而系统中没有安装 Java，那么安装程序会询问用户是否要安装 Java，如图 1-11。安装 Java 并不同时安装 IE 浏览器。

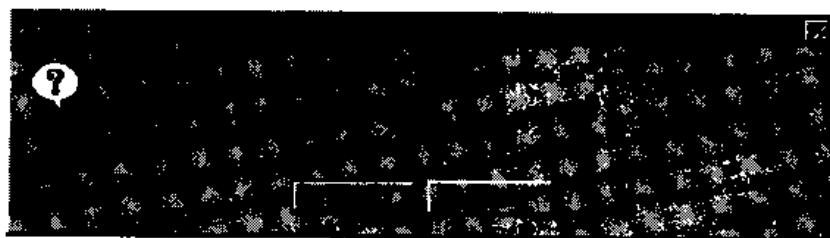


图 1-11 询问是否安装 Java



图 1-12 提示重启计算机

如果用户需要安装 Java，下一步将弹出另一个对话框进一步确认，然后是软件许可协议，接受协议后安装程序开始进行安装。安装完成后，出现图 1-12 所示的对话框，提示用户需要重新启动计算机才能使设置生效，这里选择“是”或“否”都不会立即重新启动，随后弹出 MATLAB 的总安装过程结束对话框，如图 1-13 所示。

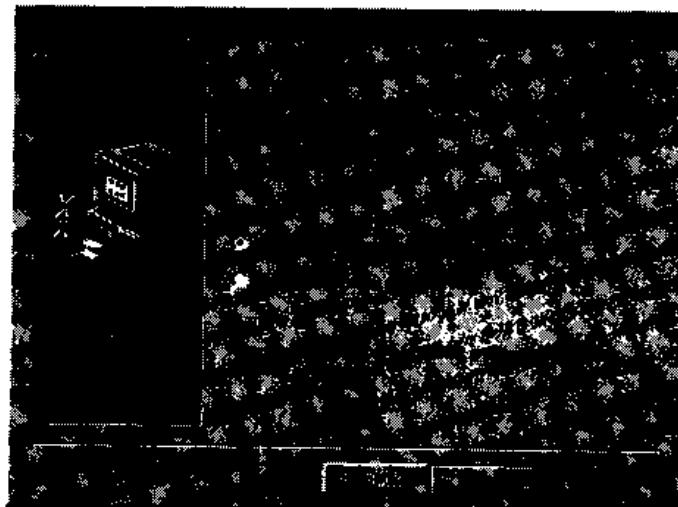


图 1-13 安装结束对话框

在安装结束对话框中提示用户安装完成后，如果需要使用 MATLAB API，则首先启动 MATLAB，然后在命令窗口中运行“mex -setup”；如果需要使用 Notebook，则在命令窗口中运行“notebook -setup”。

同时，在这个对话框中选择是否要重启计算机，并按下“Finish”以结束安装。至此，MATLAB 的安装过程全部结束。

安装的最后，程序会自动创建如图 1-14 所示的 MATLAB 程序组，同时在桌面上建立快捷方式。

### 1.2.3 MATLAB 的启动和退出

#### 1. 启动 MATLAB

在 Windows 98 操作系统中，双击桌面上的 MATLAB 快捷方式图标或单击 Windows 98 的开始菜单，依次指向“程序”、“MATLAB”，单击“MATLAB 5.3”，将进入 MATLAB 的命令窗口，如图 1-15 所示。

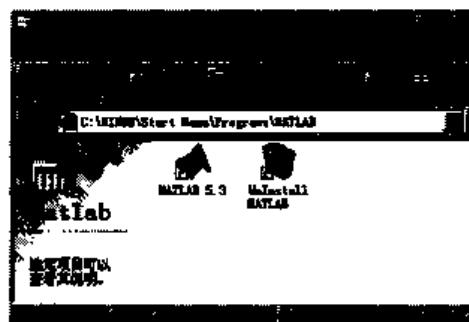


图 1-14 MATLAB 程序组

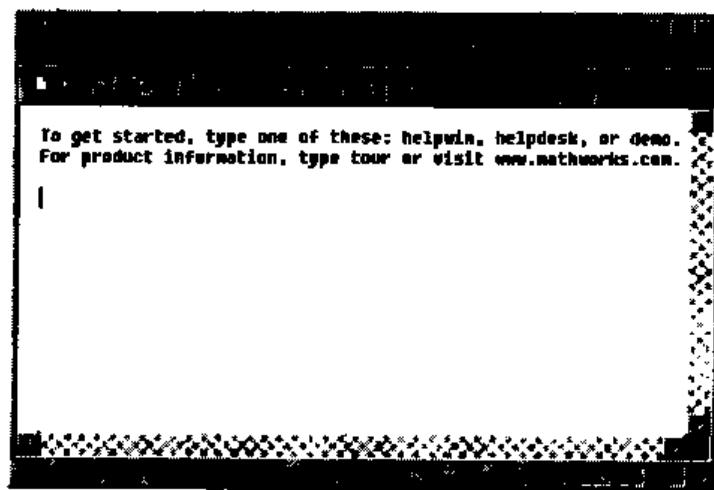


图 1-15 MATLAB 命令窗口

在 MATLAB 命令窗口的上方两行文字是初始提示信息。若 MATLAB 运行在英文 Windows 平台上，那么 MATLAB 命令窗口中的第三行将出现 MATLAB 命令提示符“»”和光标，而在中文 Windows 平台上将不显示命令提示符“»”，而只有光标。

#### 2. 退出 MATLAB

退出 MATLAB 的方式有多种：

- 1) 在 MATLAB 命令窗口的“File”菜单下选择“Exit MATLAB”；
- 2) 快捷键“Ctrl + q”；
- 3) 在命令窗口输入“quit”命令；
- 4) 在命令窗口输入“exit”命令；
- 5) 用鼠标单击 MATLAB 命令窗口右上角的“X”按钮；
- 6) 用鼠标双击 MATLAB 命令窗口左上角的图标。

#### 1.2.4 卸载 MATLAB

当用户需要卸载 MATLAB 时，单击 Windows 98 的开始菜单，依次指向“程序”、“MATLAB”，单击“UnInstall MATLAB ...”，将出现“Comfirm File Deletion”对话框，如图 1-16 所示。该对话框将让用户确认是否真的需要卸载 MATLAB。

单击“否”按钮，关闭该对话框。单击“是”按钮，将卸载 MATLAB 5.3，如图 1-17 所示。



图 1-16 确认文件删除



图 1-17 开始卸载 MATLAB

### 1.3 MATLAB 的目录结构

MATLAB 在安装盘上以压缩方式分布，安装程序把这些文件解压缩并安装到 Windows 环境下。安装完成后，MATLAB 安装目录下将包括以下一些文件和子目录(这里以在 Windows 98 系统下的完全安装为例)：

DeIsL1.isu  
license.txt  
\BIN  
\EXLINK  
\EXTERN  
\HELP  
\JAVA  
\NOTEBOOK  
\RTW  
\SIMULINK  
\STATEFLOW  
\SYS  
\TOOLBOX  
\WORK  
(1) DeIsL1.isu (文件)

该文件记录了本次 MATLAB 安装的信息，在卸载 MATLAB 时必须存在该文件。

(2) license.txt (文件)

该文件为软件许可协议的内容。

(3) \BIN

该目录包含 MATLAB 系统运行文件“MATLAB.EXE”，建立 MEX 文件所需的批处理文件，及多种相关的二进制文件。

(4) \EXLINK

该目录下的文件用于在 MATLAB 中连接 Excel。

(5) \EXTERN

建立 MATLAB 的外部程序接口所需的工具。该目录包括四个子目录：

\EXAMPLES：应用 C 语言和 FORTRAN 语言的 API 程序实例。

\INCLUDE：外部接口程序库的头文件。

\LIB：外部接口程序的目标连接库。

\SRC：建立 MEX 文件的 C 源程序举例。

(6) \HELP

帮助文件系统。

(7) \JAVA

Java 的支持程序。

(8) \NOTEBOOK

Notebook 是用来实现 MATLAB 数学工作环境与 Word 字处理环境信息交换的软件，是一个兼备数学计算、图形显示、文字处理能力的集成环境。

这个目录包括了 Notebook 的模板“M-book.dot”和示范文件。

(9) \RTW

该目录下是 Real-Time Workshop 软件包。

Real-Time Workshop 是一种实时开发环境，可以直接从 Simulink 的模型产生出可移植的程序源代码（C 语言或 Ada 语言代码）并自动构造出能在多种环境中（包括实时系统和单机仿真）实时执行的程序。它为系统从设计到实现提供了一条快捷的途径，而且简单易用。通过 Real-Time Workshop 可以在远程处理器上实时运行仿真模型，也可以在主机或外部计算机上运行高速单机仿真。

(10) \SIMULINK

该目录下是 Simulink 软件包。

Simulink 是对动态系统进行建模、仿真和分析的一个软件包。它支持线性和非线性系统、连续时间系统、离散时间系统、连续和离散混合系统，而且系统可以是多进程的。

(11) \STATEFLOW

该目录下是 Stateflow 软件包。

Stateflow 是一个功能强大的图形化开发和设计工具，用于解决复杂的控制系统、管理系统设计问题。

(12) \SYS

该目录下是 MATLAB 需要的工具和操作系统库。

(13) \WORK

该目录是 MATLAB 的当前工作目录，是缺省的 M 文件和数据文件保存目录。

#### (14) \TOOLBOX

该目录下的子目录是 MATLAB 的各种工具箱，它包括如下一些子目录：

\MATLAB： MATLAB 核心部分工具包，该目录中的内容下面详细介绍。

\RUNTIME： MATLAB 运行时间服务器开发工具包

\DAQ： 数据采集工具箱

\DIALS： 计量仪表模块集

\RPTGEN： MATLAB 报告发生器

\RPTGENEXT： Simulink 报告发生器

\DATABASE： 数据库工具箱

\POWERSYS： 动力系统模块集

\COMPILER： MATLAB 编译器

\COMM： 通信工具箱

\SYMBOLIC： 符号数学工具箱

\NAG： 数值和统计（Numerical & Statistical）工具箱

\MAP： 地图绘制工具箱

\WAVELET： 小波工具箱

\PDE： 偏微分方程工具箱

\FINANCE： 金融工具箱

\LMI： 线性矩阵不等式工具箱

\QFT： QFT（Quantitative Feedback Theory，定量反馈理论）控制系统设计工具箱

\FIXPOINT： 固定点模块集

\DSPBLKS： 数字信号处理模块集

\FUZZY： 模糊逻辑工具箱

\MPC： 模型预测控制（Model Predictive Control）工具箱

\FDIDENT： 频域识别工具箱

\HOSA： 高阶谱分析工具箱

\STATS： 统计工具箱

\NCD： 非线性控制系统设计模块集

\IMAGES： 图像处理工具箱

\NNET： 神经网络工具箱

\MUTOOLS：  $\mu$  分析与综合工具箱

\SIGNAL： 信号处理工具箱

\SPLINES： 样条工具箱

\OPTIM： 最优化工具箱

\ROBUST： 鲁棒控制工具箱

\IDENT： 系统识别工具箱

\CONTROL： 控制系统工具箱

\RTW： Real-Time Workshop 工具箱

\SB2SL: SystemBuild 到 Simulink 的转换器

\TOUR: MATLAB 漫游

\STATEFLOW: Stateflow 工具箱

\SIMULINK: Simulink 工具箱

\LOCAL: 用于局部环境设置的 M 文件

#### (15) \TOOLBOX\MATLAB

该目录下的各个子目录中的内容为 MATLAB 核心部分工具包，这些子目录有：

\DATAFUN: 数据分析和傅立叶变换函数

\DATATYPES: 数据类型和结构

\DEMOS: 例子

\ELFUN: 基本的数学函数

\ELMAT: 基本矩阵和矩阵操作函数

\FUNFUN: 功能函数

\GENERAL: 通用命令

\GRAPH2D: 绘制二维图形的函数

\GRAPH3D: 绘制三维图形的函数

\GRAPHICS: 通用绘图命令

\IOFUN: 低级文件 I/O 函数

\LANG: 语言结构设计和调试函数

\MATFUN: 矩阵函数——数值线性代数

\OPS: 运算符和特殊符号

\POLYFUN: 多项式和插值函数

\SPARFUN: 稀疏矩阵函数

\SPECFUN: 特殊数学函数

\SPECGRAPH: 特殊图形函数

\STRFUN: 字符串函数

\TIMEFUN: 时间、日期和日历函数

\UITOOLS: GUI 设计工具

\WINFUN: Windows 操作系统接口函数

## 1.4 MATLAB 应用入门

本节通过介绍 MATLAB 工作环境，使初学者初步掌握 MATLAB 软件的基本操作方法，并具备运用 MATLAB 的基本工作能力。

### 1.4.1 命令窗口的菜单系统

MATLAB 启动后出现命令窗口，参见前面的图 1-15。MATLAB 的命令窗口是用户使用 MATLAB 进行工作的交互界面，同时也是实现 MATLAB 各种功能的窗口，MATLAB 的各种操作都是从命令窗口开始的，因而有必要首先了解命令窗口的基本环境。

本节只介绍基本的内容，更深入的操作在后面的相关章节会讲到。

MATLAB 命令窗口的菜单由 File、Edit、View、Window、Help 五个主菜单组成。

### 1. File 菜单

#### (1) New

File 菜单下的子菜单 New 有三个选项：

- “M-file” 新建一个 M 文件，该命令将打开 MATLAB 的 M 文件编辑/调试器。通过 M 文件编辑/调试器，用户可以创建自己的 M 文件，也可以编辑已有的 M 文件，并可以调试 MATLAB 程序。
- “Figure” 新建一个图形窗口，参见 1.4.4 节。
- “Model” 新建一个 Simulink 模型窗口。

#### (2) Open

弹出打开文件对话框。用户可以搜寻要打开的 MATLAB 文件所在的目录，选中该文件，然后单击“打开”按钮，将打开该 MATLAB 文件。

#### (3) Open Selection

用户可以在 MATLAB 命令窗口中选中需要编辑的 M 文件名，然后选择“Open Selection”选项，就可以直接打开该文件。

#### (4) Run Script

用来运行脚本文件，用户可以在弹出的“Run Script”对话框中直接输入脚本文件名，然后单击“OK”按钮运行脚本文件。如图 1-18 所示。

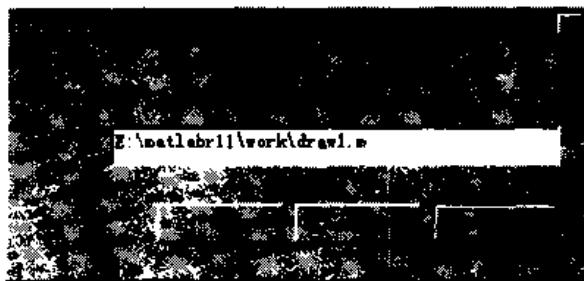


图 1-18 “Run Script” 对话框

#### (5) Load Workspace

用来载入 MATLAB 数据文件中的内容到工作空间。

#### (6) Save Workspace As

使用二进制的 MAT 文件保存 MATLAB 工作空间的内容。

#### (7) Show Workspace

打开 MATLAB 工作空间浏览器。MATLAB 工作空间浏览器能够显示用户在 MATLAB 中可以通过命令行进行操作的当前变量的集合。

#### (8) Show Graphics Property Editor

显示图形属性编辑器。属性编辑器允许交互地修改图形对象属性

#### (9) Show GUI Layout Tool

显示图形用户界面设计向导，给用户提供了一个控制面板用于打开其它设计向导。

#### (10) Set Path

打开路径浏览器。

(11) Preferences

打开参数设置对话框。通过参数设置对话框，用户可以设置 MATLAB 工作环境的外观和操作的相关属性。

(12) Print Setup

进行打印设置。

(13) Print

对屏幕的内容进行打印。

(14) Print Selection

打印在命令窗口中选择的语句和运行结果。

(15) Exit Matlab

退出 MATLAB。

2. Edit 菜单

Edit 菜单的各选项和通用的 Windows 编辑菜单功能类似：

- 1) Undo: 撤消上一次的操作
- 2) Cut: 将选中的内容剪切到剪贴板上
- 3) Copy: 复制选中的内容
- 4) Paste: 将剪贴板上的内容粘贴到指定位置
- 5) Clear: 清除工作空间中的指定变量
- 6) Select All: 选中命令窗口中的所有内容
- 7) Clear Session: 清除命令窗口里所有显示的内容

3. View 菜单

View 菜单较为简单，仅用于控制是否显示工具栏。当选中“Toolbar”时，则显示工具栏；否则，隐藏工具栏。

4. Window 菜单

Window 菜单使用户可以在打开的窗口之间方便地进行切换。

5. Help 菜单

MATLAB 为用户提供了非常详尽的帮助信息，而且获取帮助的方式也有多种，这些内容将在后面的 1.5 节中详细介绍。这里只简要说明帮助菜单的选项。

MATLAB 的帮助菜单是获得 MATLAB 帮助信息的方式之一，它包括如下七个选项：

- 1) Help Window: 显示帮助窗口
- 2) Help Tips: 显示帮助提示
- 3) Help Desk (HTML): 进入帮助面板
- 4) Examples and Demos: 进入 MATLAB 演示窗口
- 5) About MATLAB: 显示 MATLAB 的版本和用户信息
- 6) Show License: 在 MATLAB 编辑器中显示许可协议的内容。
- 7) Join MATLAB Access: 打开 subscribe.html 文件，提示用户加入 MATLAB 访问成员组。如果用户能够连接到 Internet 上，可以通过注册成为 MATLAB 访问成员。当用户成为

MATLAB 访问成员后，除了可以得到最新的 MATLAB 开发动态，并且可以获得许多非访问成员无法享受的利益。

### 1.4.2 命令窗口的工具栏

工具栏是新版本的 MATLAB 中新增的功能，它位于 MATLAB 命令窗口菜单的下面。工具栏是 MATLAB 为用户提供的常用命令的快捷方式。这里我们简单列出工具栏中各个按钮的功能：

- 用 MATLAB 的 M 文件编辑/调试器新建一个文件。
- 用 MATLAB 的 M 文件编辑/调试器打开一个文件。
- 把选中的内容剪切到剪贴板中。
- 把选中的内容复制到剪贴板中。
- 把剪贴板中的内容粘贴到命令窗口的光标所在位置。
- 撤消最近的一次操作。
- 打开工作空间浏览器。
- 打开路径浏览器。
- 打开 Simulink 库浏览器。
- 打开 MATLAB 帮助窗口。

### 1.4.3 矩阵和常用命令的输入

#### 1. 输入矩阵

MATLAB 最基本的功能就是矩阵运算，因而我们首先应该学会如何输入矩阵。在 MATLAB 中矩阵的输入方法很多，这里只简单介绍在命令窗口直接输入矩阵的方法，更详细的内容参见第 2 章。

在 MATLAB 命令窗口中输入矩阵需要遵循以下基本规则：

- 1) 把矩阵元素列入方括号“[ ]”中
- 2) 矩阵每行内的元素间用逗号或空格分开
- 3) 矩阵的行与行之间用分号隔开，或者分行输入

**本书约定：在例子中，凡是输入的内容一律加黑显示。**

在命令窗口输入如下内容：

**a = [1 2 3;4 5 6;7 8 9]**

或 **a = [1,2,3;4,5,6;7,8,9]**

按回车键后，命令窗口将显示矩阵“a”的值，同时把它的值保存在 MATLAB 的工作空间中，直到退出 MATLAB 或用“clear”命令清除它。

上面这两种输入方式的结果是相同的，如下所示：

**a =**

1	2	3
4	5	6

7      8      9

注意：如果不需要在窗口中立即显示输入语句的结果，那么在行尾加分号“;”即可。读者可以用上面的例子自己实验。

对于比较大的矩阵，也可以分行输入，这种输入方式更为直观。

在命令窗口中分行输入如下内容（每个行尾都要加回车键）：

`x = [4 5 7 8`

`6 1 2 5`

`2 5 6 7`

`8 3 5 6]`

到最后一行的中括号输入完成并按下回车键后，将在命令窗口中显示如下结果：

`x =`

4	5	7	8
6	1	2	5
2	5	6	7
8	3	5	6

## 2. 常用命令

用户可以直接在 MATLAB 命令窗口输入并执行命令，常用的命令如表 1-2 所示。

表 1-2 常用命令

命    令	功    能
<code>cd</code>	显示或改变当前工作目录
<code>dir</code>	列出当前目录或指定下的文件和子目录清单
<code>clc、home</code>	擦除 MATLAB 命令窗口中的所有显示内容，并把光标移到命令窗口的左上角
<code>clf</code>	擦除 MATLAB 当前图形窗口中的图形
<code>clear</code>	清除内存中的变量和函数
<code>disp</code>	显示变量的内容
<code>type</code>	列出指定文件的全部内容
<code>exit</code>	退出 MATLAB
<code>quit</code>	退出 MATLAB

下面介绍几个主要命令的用法。

### (1) cd

- `cd` 目录名：把指定目录设为当前目录。
- `cd ..`：把当前目录上移一层（注意，这种用法要求在命令“`cd`”后必须加一个空格，而在两个点之间不能有空格）。

- cd: 后面不带任何内容, 给出当前工作目录。
- a = cd: 把当前目录作为一个字符串返回给变量“a”。
- cd ('目录名'): 这是该命令的函数形式, 把当前目录改为由“目录名”指定的目录。

#### (2) dir

- dir 目录名: 列出指定目录中的文件和子目录, 不指定目录名则显示当前目录中的内容, 类似 DOS 下的使用方式, 该命令也可以使用通配符“\*”。
- d = dir('目录名'): 将结果返回给一个结构变量“d”。(结构的概念)

#### (3) clear

- clear: 清除工作空间中的所有变量。
- clear 变量名: 从工作空间中清除指定的变量。

#### (4) disp

disp(变量名): 显示变量的内容, 不显示变量名。

### 3. 输入内容的编辑

在 MATLAB 命令窗口中, 为了便于对输入的内容进行编辑, MATLAB 提供了一些控制光标位置和进行简单编辑的一些常用编辑键和组合键, 见表 1-3。

表 1-3 常用编辑键

编辑键	组合键	作用
↑	Ctrl+p	恢复前一行
↓	Ctrl+n	恢复后一行
←	Ctrl+b	光标向左移动一个字符
→	Ctrl+f	光标向右移动一个字符
Home	Ctrl+↑	光标移动至行首
End	Ctrl+↓	光标移动至行尾
Delete	Ctrl+d	删除光标后字符
Backspace	Ctrl+h	删除光标前字符
Esc	Ctrl+u	清除当前行
Ctrl+←	Ctrl+l	光标向左移动一个单词
Ctrl+→	Ctrl+r	光标向右移动一个单词

### 1.4.4 图形窗口

MATLAB 的绘图功能十分强大, 而且使用非常方便。这也是它的重要特点之一。

和 MATLAB4.x 版本相比, MATLAB 5.3 在这方面有了更大的进步, 不但新增了很多图形图像处理功能, 而且为图形窗口增加了菜单和工具条, 增强了交互处理能力。

图 1-19 就是 MATLAB 5.3 的图形窗口。它在 MATLAB4.x 版本的基础上增加了菜单和工具条, 其中包括通用的文件操作命令、编辑命令, 对图形的坐标轴、线型等特性进行设

置的专用工具，还可以为图形添加标注。用菜单命令操作图形窗口简便、直观，可以直接观察效果。

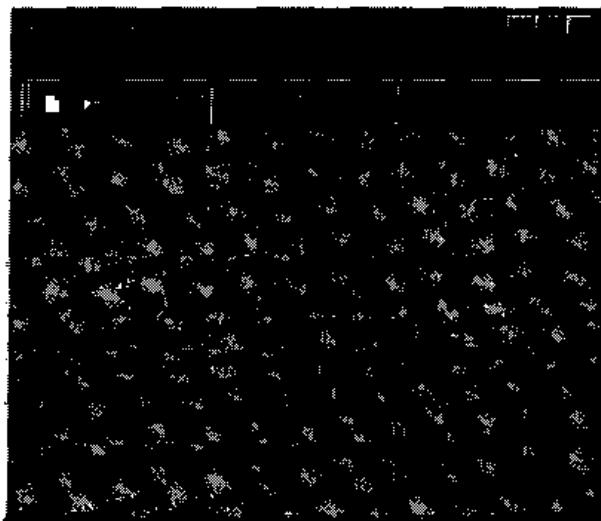


图 1-19 MATLAB 5.3 图形窗口

在 MATLAB 下建立图形窗口由命令 `figure` 完成，每执行一次 `figure` 命令就产生一个图形窗口，可以同时产生若干个图形窗口，MATLAB 自动在这些窗口的名字上添加序号（No.1, No.2, ...）作为区别。

关闭图形窗口由 `close` 命令来完成，每执行一次 `close` 命令关闭一个当前的图形窗口，要同时关闭所有窗口，使用 `close all`。

图形窗口工具条上有几个专用按钮，是针对图形进行操作的，其功能分别为：

按钮■：允许对图形进行编辑；

按钮■：在图形窗口中添加文本；

按钮■：在图形窗口中添加箭头；

按钮■：在图形窗口中添加直线；

按钮■：允许对图形进行放缩操作，该按钮按下后，在图形中单击鼠标左键，图形会被放大，单击鼠标右键，图形会被缩小；

按钮■：允许对图形进行缩放操作，该按钮按下后，在图形中单击鼠标右键，图形会被放大，单击鼠标左键，图形会被缩小；

按钮■：允许把图形旋转为三维图形。

## 1.5 使用 MATLAB 帮助

MATLAB 为用户提供了非常详尽的多种帮助信息，例如 MATLAB 的在线帮助、帮助窗口、帮助提示、HTML 格式的帮助、pdf 格式的帮助文件以及 MATLAB 的例子和演示等等。

通过使用 MATLAB 的帮助菜单或在命令窗口输入帮助命令，用户都可以很容易地获得 MATLAB 的帮助信息，并能够通过帮助进一步学习 MATLAB。常用的帮助命令如表 1-4 所示。

表 1-4 常用命令

帮助命令	功能
help	获取在线帮助
demo	运行 MATLAB 演示程序
tour	运行 MATLAB 漫游程序
who	列出当前工作空间中的变量
whos	列出当前工作空间中变量的更多信息
what	列出当前目录或指定目录下的 M 文件、MAT 文件和 MEX 文件
which	显示指定函数或文件的路径
lookfor	按照指定的关键字查找所有相关的 M 文件
exist	检查指定的变量或函数文件的存在性
helpwin	运行帮助窗口
helpdesk	运行 HTML 格式的帮助面板 Help Desk
doc	在网络浏览器中显示指定内容的 HTML 格式帮助文件，或启动 helpdesk

### 1.5.1 help 命令

help 命令是最常用的在线帮助命令。使用 help 命令可以查询所有 MATLAB 函数的用法。这样，无论用户在何时有疑惑的问题，都可以使用使用 MATLAB 的 help 命令来获得帮助。help 命令的使用方法如下：

- help 帮助主题：获取指定主题的帮助信息。“帮助主题”可以是命令名、目录名或者 MATLAB 搜索路径中的部分路径名（注意用“/”和“\”分隔路径都允许）。如果是命令名，将会显示该命令的信息；如果是目录名或部分路径名，将会列出指定目录下的文件名和每个文件中第一行的简要说明。
- help 函数名：显示该函数的帮助信息。
- help：后面不带内容，将列出所有的帮助主题，每个帮助主题对应于 MATLAB 搜索路径中的一个目录。
- T = help(‘帮助主题’)：把帮助信息作为一个字符串返回给指定变量“T”。

(1) help 命令后加命令名

**help clc**

CLC Clear command window.

CLC clears the command window and homes the cursor.

See also HOME.

在这里我们看到，帮助信息的第一部分（第一行）是介绍该命令的功能，第二部分说明该命令的用法，第三部分给出相关命令。

(2) help 命令后加函数名

**help inv**

**INV** Matrix inverse.

INV(X) is the inverse of the square matrix X.

A warning message is printed if X is badly scaled or  
nearly singular.

See also SLASH, PINV, COND, CONDEST, NNLS, LSCOV.

Overloaded methods

```
help sym/inv.m
help zpk/inv.m
help tf/inv.m
help ss/inv.m
help lti/inv.m
help frd/inv.m
```

函数的帮助信息一般比命令多一部分：提示如何获取函数名重载的帮助信息。

(3) help 命令后加部分路径名或目录名

以下几种输入方式的输出结果是一样的 (MATLAB 的安装路径为 “e:\matlabr11”):

```
help lang
help matlab/lang
help toolbox/matlab/lang
help matlabr11/toolbox/matlab/lang
help e:/matlabr11/toolbox/matlab/lang
```

输出结果为目录 “e:\matlabr11\toolbox\matlab\lang” 下的文件名及其简要说明 (文件的第一行内容)，也就是该目录下的文件 “Contents.m” 中的所有内容。读者可以自己实验。

(4) help 命令后不加参数

这种情况下显示 MATLAB 所有搜索路径及简要说明。

### 1.5.2 演示和漫游

Mathworks 公司精心设计了两个旨在介绍 MATLAB 功能的演示和漫游程序：demo 和 tour。这两个程序的界面友好，示例典型，表现方式生动直观，对新老用户都十分有益，是有关的参考书籍不能代替的。

下面我们分别介绍这两个命令的用法。

#### 1. demo

直接在命令窗口输入命令

**demo**

就将打开 MATLAB 演示窗口，如图 1-20 所示。

图中有三个文本框，左边的文本框中是总的结构，在这里选择需要演示的类别；右上方的文本框是对选中的类别的介绍；右下方的文本框是供选择的示例名称，在这里选择需要演示的例子，然后单击右下方的按钮 “Run xxx” 即打开该例子的演示窗口。

例如，我们在左边文本框中选择 MATLAB 下的 “Matrices”，在右下方的文本框中选择

“Basic matrix operations”，然后按下右下方的按钮“Run Basic matrix ...”，这时会打开如图 1-21 所示的演示画面。

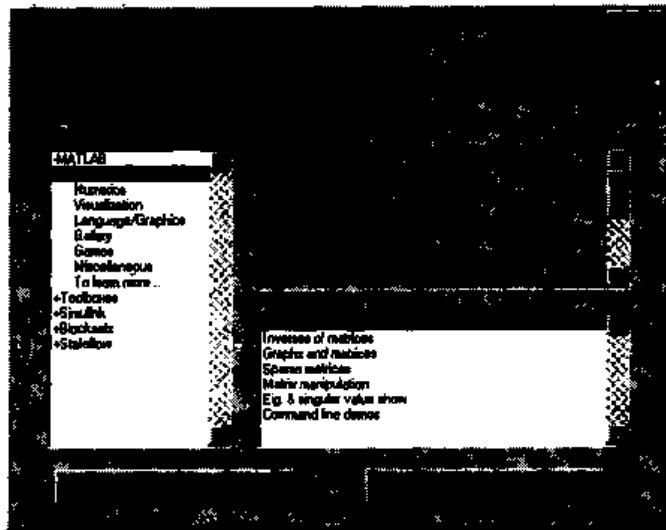


图 1-20 MATLAB 演示窗口

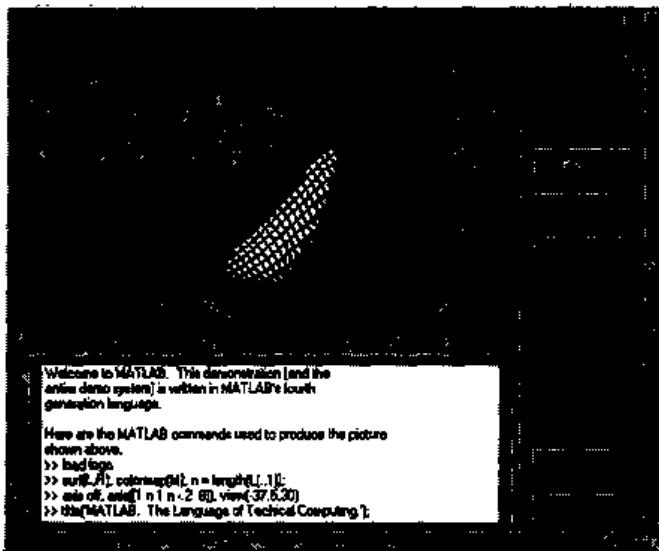


图 1-21 基本矩阵操作演示

在图 1-21 中，如果要逐页观察演示，可以在窗口的右上方按“Start”按钮，“Start”就会变成“Next”，然后再逐次按下“Next”；也可以只按下“AutoPlay”按钮由演示程序自动播放各页的内容。

窗口左下方的文本框中给出了每页内容的说明和相应的 MATLAB 程序。窗口右下方的按钮“Info”给出这个演示程序的相关信息，“Close”关闭这个演示窗口。

demo 程序中的内容非常丰富，这里不可能详细介绍，读者可以就感兴趣的方面自己观看该程序的演示。

## 2. tour

该程序的用法和 demo 类似，在命令窗口直接输入：

**tour**

然后就会打开如图 1-22 所示的窗口。该窗口为用户提供了比 demo 窗口更为全面的介绍。用户可以单击该窗口中的相应主题来打开相应的内容。

例如，单击该窗口左侧的“Overview of Products”区域，将打开 Mathworks 公司系列产品的介绍窗口，如图 1-23 所示。

如果单击主窗口中“MATLAB”左侧的按钮，则会打开如图 1-24 所示的 MATLAB 基本内容演示窗口，它包括了 demo 程序中的相应内容，而且有所补充。

主窗口下方的三个图案分别对应三个具有代表性的 MATLAB 图形，它们形象地说明了 MATLAB 的绘图能力。例如，我们用鼠标单击右侧的图案，会出现一幅具有光照效果的三维绳结图，如图 1-25 所示。

关于 tour 程序中的详细内容，请读者根据提示自己观看。



图 1-22 MATLAB 漫游窗口

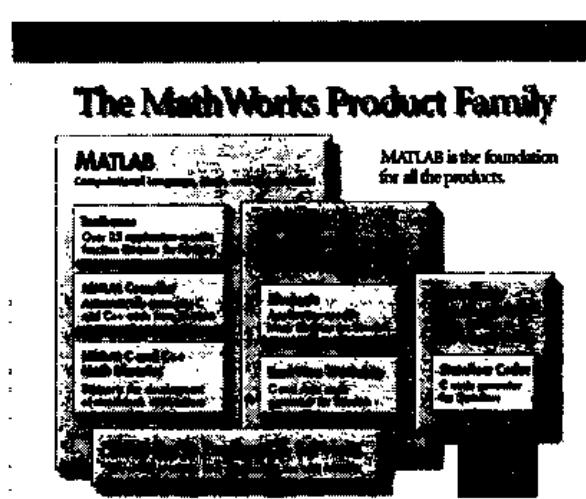


图 1-23 Mathworks 公司系列产品

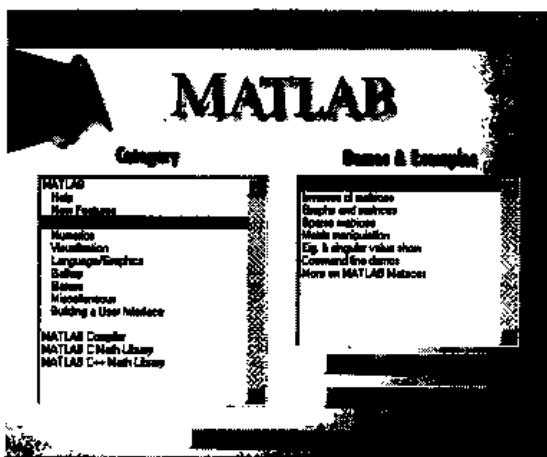


图 1-24 MATLAB 基本内容



图 1-25 三维绳结图

### 1.5.3 其它帮助命令

除了 help、demo、tour 之外，在表 1-4 中还列出了其它几种获取帮助的命令，这里分别加以介绍。

### 1. lookfor

当要查找具有某种功能的命令或函数，但又不知道该命令或函数的确切名字时，`help` 命令就不能用了。这时就要借助于 `lookfor` 命令，它允许用户通过完整的或部分关键字来搜索相关内容。一般情况下，该命令仅搜索各个文件帮助文本的第一行。

例如，在用 MATLAB 设计程序时，需要为用户提供鼠标绘图的功能，但不知道 MATLAB 中的相关函数有哪些，这时就可以借助于 `lookfor` 命令。

在命令窗口输入：

`lookfor mouse`

然后 MATLAB 就会自动在每个函数和命令文件的第一行进行查找，找到包含关键字“`mouse`”的文件就列出文件名和文件的第一行内容。下面列出了其中的一部分信息：

`GTEXT Place text with mouse.`

`DRAGRECT Drag XOR rectangles with mouse.`

`GINPUT Graphical input from mouse.`

`MOUSEDRV Hardware driver setup for the MOUSEDRV board.`

`GTEXTM Place text on a 2-D map using a mouse`

`INPUTTM Return mouse click positions from displayed map`

`scircleg.m: %SCIRCLEG: Display of small circles defined via a mouse input`

`TRACKG Display of great circles and rhumb lines defined via a mouse input`

### 2. exist

`exist` 命令用来检查变量或函数是否被定义。它的用法如下：

- `exist ('名字')`: 检查指定名字的变量或函数是否存在于 MATLAB 的搜索路径内。
- `exist ('名字', 'var')`: 只检查变量。
- `exist ('名字', 'builtin')`: 只检查嵌入函数。
- `exist ('名字', 'file')`: 检查文件名和目录名。
- `exist ('名字', 'dir')`: 只检查目录名。

该函数的返回值有以下几种情况：

- 返回 0: 检查的内容不存在。
- 返回 1: 检查的内容是工作空间内的变量。
- 返回 2: 检查的内容是 MATLAB 搜索路径内的 M 文件或不知类型的文件。
- 返回 3: 检查的内容是 MATLAB 搜索路径内的 MEX 文件。
- 返回 4: 检查的内容是 MATLAB 搜索路径内的 MDL 文件。
- 返回 5: 检查的内容是 MATLAB 的嵌入函数。
- 返回 6: 检查的内容是 MATLAB 搜索路径内的 P 文件。
- 返回 7: 检查的内容是一个目录。

在命令窗口输入：

`exist('dir')`

返回值为：

```
ans =
```

```
5
```

表示“dir”为 MATLAB 的嵌入函数。如果输入：

```
exist('dir.m')
```

则返回值为：

```
ans =
```

```
2
```

说明“dir.m”为 MATLAB 搜索路径内的 M 文件。

### 3. who 和 whos

who 和 whos 这两个命令的作用都是列出在 MATLAB 工作空间内驻留的变量名，whos 还同时给出变量的维数、大小和类型。

```
who
```

显示内容为：

```
Your variables are:
```

a	tout	xFinal	yout
t	u	xout	

```
whos
```

显示更详细的信息：

Name	Size	Bytes	Class
a	10x2	160	double array
t	10x1	80	double array
tout	217x1	1736	double array
u	10x1	80	double array
xFinal	1x1	8	double array
xout	217x1	1736	double array
yout	217x1	1736	double array

Grand total is 692 elements using 5536 bytes

### 4. what

命令 what 用于列出指定目录下的 MATLAB 所特有的文件类型，包括 M 文件、MAT 文件、MEX 文件、MDL 文件以及 P 文件等。

如果不指定目录名，那么 what 命令列出当前工作目录下的相关文件。

在命令窗口输入如下几条命令中的任一条：

```
what control
```

```
what toolbox\control
```

```
what e:\matlab\toolbox\control
```

输出结果将是相同的，请读者自己实验。

### 5. which

命令 `which` 用于定位函数和文件，即给出指定的函数或文件的完整路径。如果是工作空间中的变量或嵌入式函数，则给出相应的提示信息。

命令 `which` 可以加开关 “`-all`”，用来显示出具有指定名字的所有函数和命令。

**例 1-3 使用 `which` 命令**

请读者比较一下下面两种用法：

**which eval**

eval is a built-in function.

**which eval -all**

eval is a built-in function.

E:\MATLAbR11\toolbox\symbolic\@sym\eval.m % sym method

E:\MATLAbR11\toolbox\matlab\lang\eval.m % Shadowed

#### 1.5.4 帮助窗口 (MATLAB Help Window)

MATLAB 帮助窗口也是获取在线帮助的方式之一。在 MATLAB 命令窗口输入命令 “`helpwin`” 或在 Help 菜单下选择 “Help Window” 都可以进入 MATLAB 帮助窗口。

MATLAB 帮助窗口显示出 MATLAB 可以提供的帮助主题，如图 1-26 所示，用鼠标双击其中的某个主题就会显示相应的帮助主题的内容。这时在窗口上方的 “See also” 下拉式菜单中可以看到相关的内容。

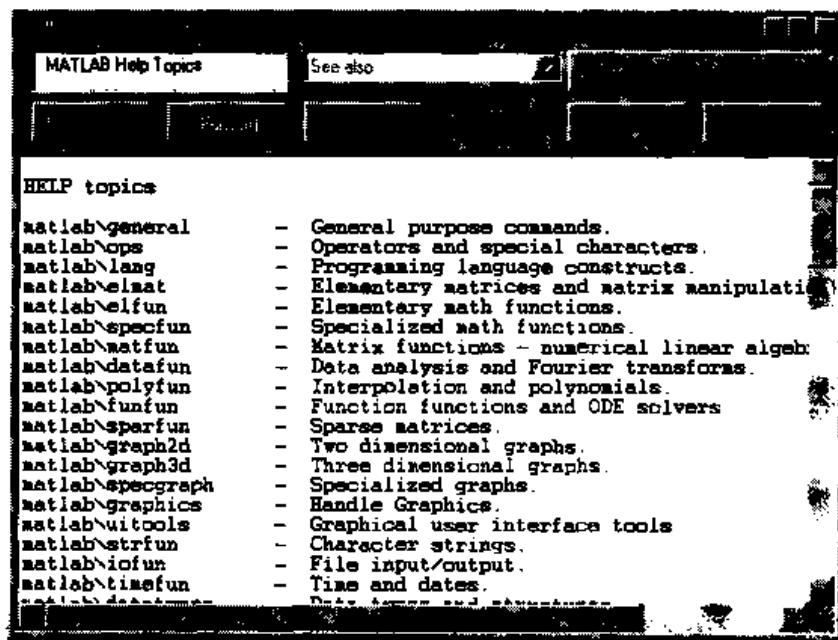


图 1-26 MATLAB 帮助窗口

单击窗口右上方的 “Tips” 按钮或在 Help 菜单中选择 “Help Tips” 可以显示 MATLAB 帮助提示，如图 1-27 所示。通过帮助提示窗口，用户可以了解 MATLAB 各种获取帮助方式的特点和用法。

单击帮助窗口右上方的 “Go to Help Desk” 按钮，可以直接进入 MATLAB Help Desk。关于 Help Desk 的具体介绍见下一小节。

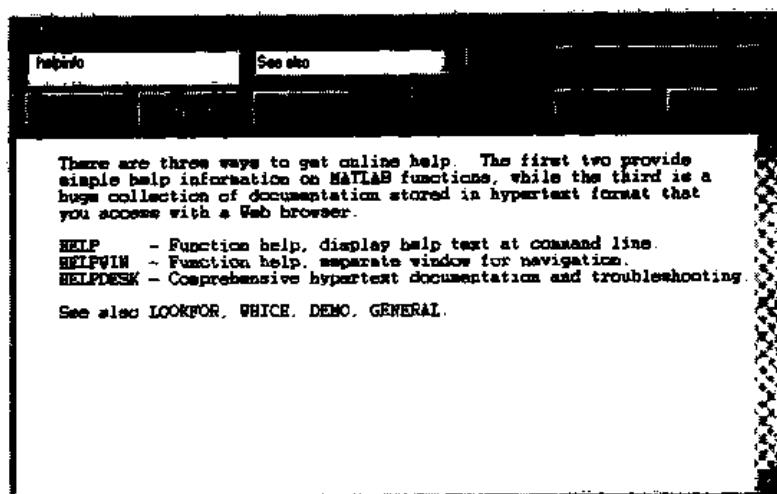


图 1-27 帮助提示窗口

### 1.5.5 Help Desk

Help Desk 是 MATLAB 提供的 HTML 格式的帮助，它是一个超文本格式帮助文件的巨大集合，通过网络浏览器查看这些帮助文件。

通过在 Help 菜单中选择“Help Desk”选项，或者直接在命令窗口内输入如下命令：

**helpdesk**

都可以打开 Help Desk，如图 1-28。

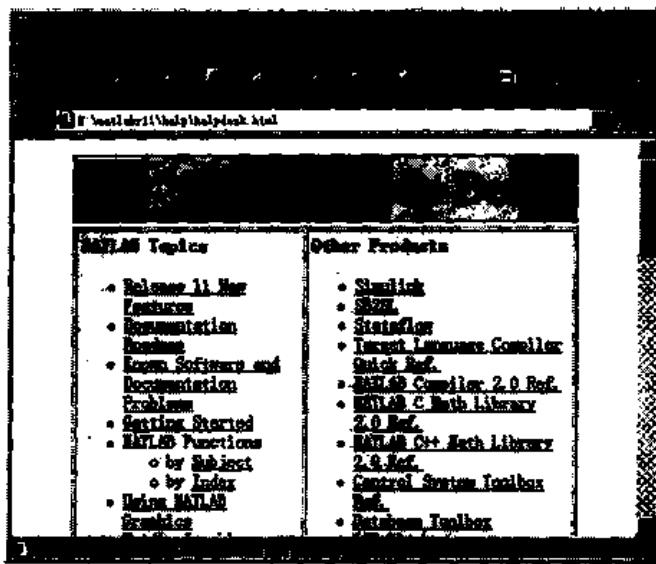


图 1-28 MATLAB Help Desk 首页

MATLAB 的 Help Desk 和相关的大量 pdf 格式的文件为用户提供的帮助信息在所有获取帮助的方式中是最全面、最深入的。

Help Desk 的用法也很简单，和在网上浏览一样，只要用鼠标在某个有下划线的内容上点一下，就会显示相应的帮助内容。

例如，MATLAB 的初学者可以在 Help Desk 首页上点击“Getting Started”来学习 MATLAB 的入门知识，这时浏览器会进入如图 1-29 所示的窗口。左侧的窗口中是这部分内容的结构，

右侧是具体内容。

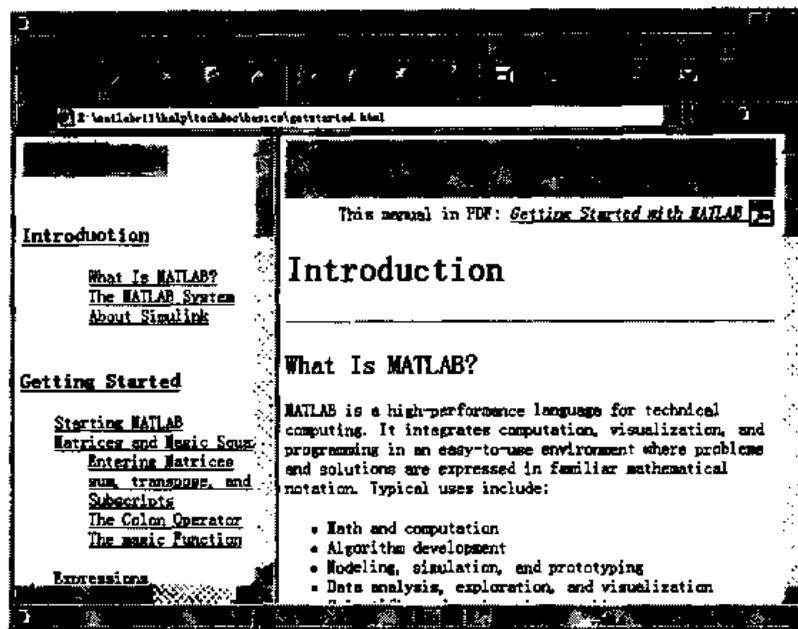


图 1-29 Getting Started 窗口

在右侧窗口的上方有一行提示文字“*This manual in PDF: Getting Started with MATLAB*”，提示用户和这部分内容相同的 pdf 格式帮助文件名，用鼠标点击有下划线的部分“*Getting Started with MATLAB*”就可以打开该 pdf 文件，打开后的窗口如图 1-30 所示。pdf 格式的文件中的内容相对比较系统，而且具有 Help Desk 中所没有的相关理论内容。

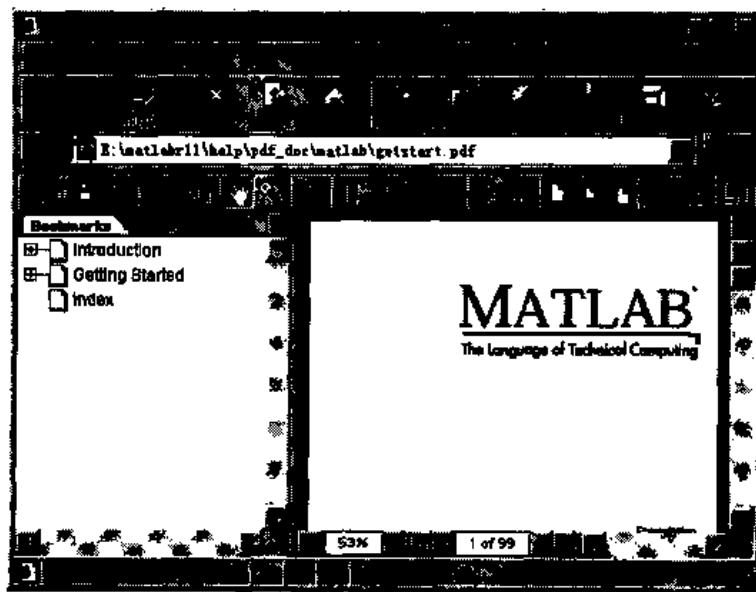


图 1-30 pdf 格式的帮助文件

在 Help Desk 中还可以实现全文搜索：用鼠标点 Help Desk 首页中左侧窗口中部的“Full-text Search”，进入全文搜索窗口，如图 1-31 所示。在“Enter query”后输入要查询的表达式，表达式中可以含有通配符“\*”，并且可以是多个表达式用逻辑符号进行组合。

就象是在图书馆查找资料一样。

从 MATLAB 安装的目录结构上看, Help Desk 的文件都在目录“help”下, Help Desk 的首页就是该目录下的文件“helpdesk.html”, 直接打开这个文件就可以运行 Help Desk, 而不必启动 MATLAB。如果经常要利用 Help Desk 进行学习, 可以在桌面上建立该文件的快捷方式。

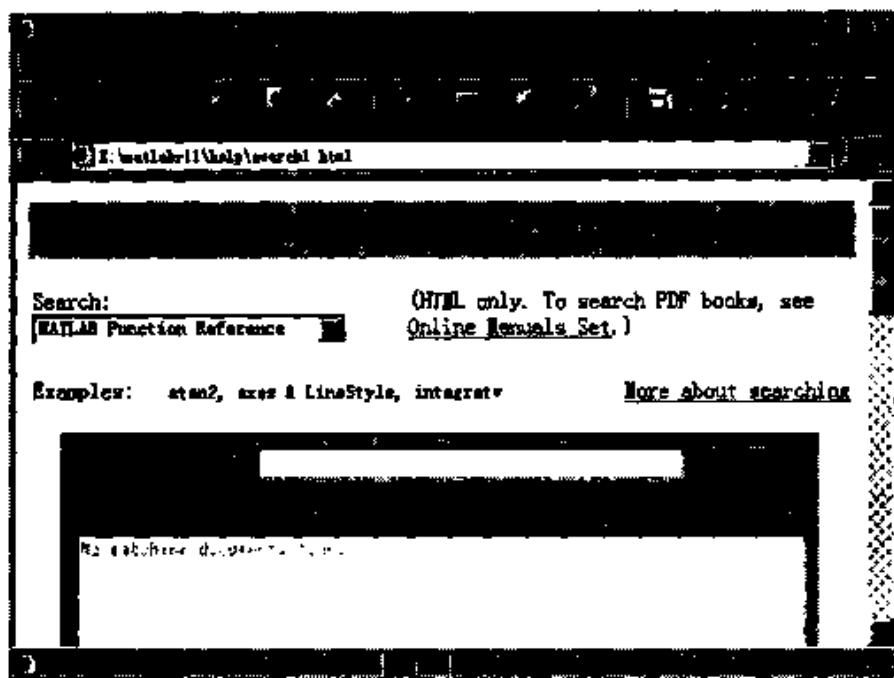


图 1-31 Help Desk 全文搜索窗口

所有 pdf 格式的帮助文件都在“help”目录的子目录“pdf\_doc”下。在完全安装的情况下, 这些 pdf 文件共有约 250MB 字节, 可见其内容非常丰富。

若用户连接在 Internet 上, 还可以通过 Help Desk 使用浏览器直接访问 Mathworks 公司的网上站点, 以获得最新的信息和网上在线帮助。

# 第2章 基本数值运算

数值运算是 MATLAB 最基本、最重要的功能，MATLAB 能够成为世界上最优秀的数学软件之一，和它出色的数值运算能力是分不开的。

MATLAB 以复数矩阵作为基本的运算单元，向量和标量都作为特殊的矩阵来处理：向量看作只有一行或一列的矩阵；标量看作只有一个元素的矩阵。

矩阵和数组的概念要注意加以区别的。非正式情况下，这两个术语通常可以互换。确切地说，矩阵是数组的一种特例，它是二维的数值型数组，表示了一种线性变换的关系。定义在矩阵概念上的数学运算也就构成了线性代数这门课程。

从运算的角度来看，矩阵运算和数组运算有显著的不同，它们在 MATLAB 中，属于两类不同的运算。矩阵运算是从矩阵的整体出发，依照线性代数的运算规则进行，而数组运算是从数组的元素出发，针对每个元素进行的运算。

本章将为读者具体讲述 MATLAB 中矩阵运算和数组运算的方法。

## 2.1 矩阵的生成

### 2.1.1 简单矩阵的生成

在 MATLAB 中，可以采用多种不同的方式生成矩阵。

#### 1. 直接输入矩阵元素

对于较小的简单的矩阵，从键盘上直接输入矩阵是最常用、最方便和最好的数值矩阵创建方法。直接从键盘输入一系列元素生成矩阵，只要遵循下面几个基本原则：

- (1) 矩阵每一行的元素必须用空格或逗号分开；
- (2) 在矩阵中，采用分号或回车表明每一行的结束；
- (3) 整个输入矩阵必须包含在方括号中。

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

```
A =
```

4	5	7	8
6	1	2	5
2	5	6	7
8	3	5	6

```
A=[4 5 7 8
```

```
6 1 2 5
```

```
2 5 6 7
```

```
8 3 5 6]
```

A =

4	5	7	8
6	1	2	5
2	5	6	7
8	3	5	6

第一种方式中，矩阵每一行用分号分隔，每一行的各个元素用逗号分隔；而在第二种方式中，矩阵每一行用回车分隔，每一行的各个元素用空格分隔，两种情况的输出结果是一样的。

一旦输入了矩阵，它将被自动地储存在 MATLAB 工作空间中，可以简单地使用矩阵名“A”来访问它。

### 2. 从外部数据文件调入矩阵

在 MATLAB 中，还可以从外部数据文件读入数据生成矩阵。这些文件可以是以前 MATLAB 生成的矩阵存储成二进制文件，也可以是包含数值数据的文本文件。在文本文件中，数据必需排成一个矩形表，数据之间用空格分隔，文件的每行仅包含矩阵的一行，并且每一行的元素个数必需相等。

16.0	3.0	2.0	13.0
5.0	10.0	11.0	8.0
9.0	6.0	7.0	12.0
4.0	15.0	14.0	1.0

然后用下述命令将 magik.dat 中的内容调入工作空间并生成变量 magik。

```
load magik.dat      % 将 magik.dat 中的内容调入工作空间
```

```
magik                % 显示变量 magik
```

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

说明：

(1) 采用本方法可以创建和保存矩阵的大小没有限制，还可以将其它的程序生成的矩阵直接调入 MATLAB 中进行处理。

(2) 在 MATLAB 中，每行中“%”以后的内容仅做注释用，对 MATLAB 的计算不产生任何影响。

### 3. 利用用户创建的 M 文件中的函数生成矩阵。

用户可以使用 M 文件生成自己的矩阵，M 文件是一种包含 MATLAB 代码的文本文件，这种文件的扩展名为“.m”，它所包含的内容就是把在 MATLAB 的命令行上键入的矩阵生成的命令存入一个文件。下面将举例说明 M 文件的创建过程。

(1) 使用编辑器输入 magik.m 文件所需内容，或在 MATLAB 的命令行中敲入如下命

令启动 MATLAB 的编辑器窗口，并新建文件“magik.m”：

**edit magik.m**

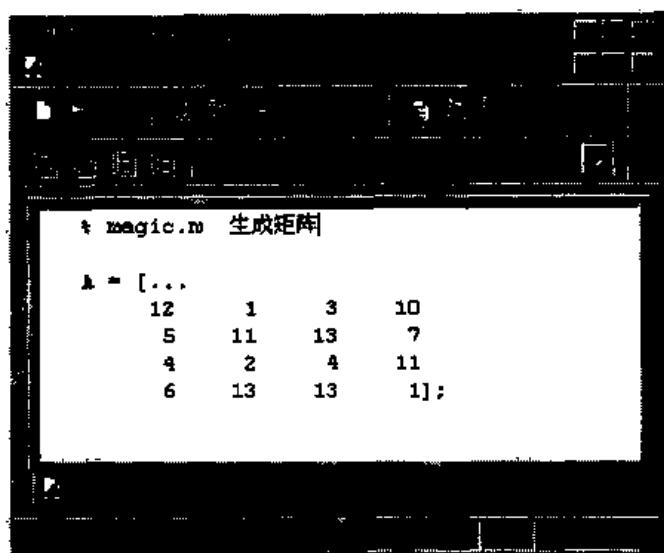


图 2-1 在编辑器窗口输入矩阵

- (2) 在编辑器窗口输入矩阵 A，并将文件保存为 magik.m。
- (3) 运行文件 magik.m，即在命令窗口输入“magik”，生成一个变量 A，包含上面生成的矩阵。

其中，在 magik.m 文件中以“%”开头的是注释行。注释行的格式通常是：“%”符号后紧接着写该 M 文件的文件名，接着写该文件的主要功能。注释文字可以是中文，也可以是英文。MATLAB 的所有 M 文件一般都是这种结构，这样可以增加 M 文件的可读性。

当一个命令过长时或者为了增加可读性而需要分行书写时，可用由三个以上小黑点组成的续行号“...”实现。

#### 4. 生成大矩阵

在 MATLAB 中，可以将小矩阵联接起来生成一个较大的矩阵。事实上，前面直接输入法生成矩阵就是将单个元素联接起来生成矩阵。方括号“[ ]”就是联接算子。

**B = [A A+32; A+48 A+16]**

**B =**

12	1	3	10	44	33	35	42
5	11	13	7	37	43	45	39
4	2	4	11	36	34	36	43
6	13	13	1	38	45	45	33
60	49	51	58	28	17	19	26
53	59	61	55	21	27	29	23
52	50	52	59	20	18	20	27
54	61	61	49	22	29	29	17

四个 4×4 的子矩阵组合成一个 8×8 的矩阵 B。

### 2.1.2 常用矩阵的生成

下面介绍一些常用矩阵的生成命令: zeros, ones, eye, rand 和 randn。

#### 1. zeros 生成全 0 阵

调用格式为:

- $B = \text{zeros}(n)$ : 生成  $n \times n$  的全 0 矩阵; 如果  $n$  不是标量将给出出错信息。
- $B = \text{zeros}(m,n)$ : 生成  $m \times n$  的全 0 阵。
- $B = \text{zeros}([m\ n])$ : 生成  $m \times n$  的全 0 阵。
- $B = \text{zeros}(d_1,d_2,d_3\dots)$ : 生成  $d_1 \times d_2 \times d_3 \times \dots$  的全 0 阵。
- $B = \text{zeros}([d_1\ d_2\ d_3\dots])$ : 生成  $d_1 \times d_2 \times d_3 \times \dots$  的全 0 阵。
- $B = \text{zeros}(\text{size}(A))$ : 生成与矩阵 A 大小相同的全 0 阵。

注意: 在 MATLAB 中不需要预先定义矩阵的维数, MATLAB 自动为矩阵分配存储空间。尽管如此, 如果采用全 0 阵为矩阵生成的全部元素或某一行、某一列的元素保留存储空间, 则大多数 MATLAB 程序将运行得更快。

#### 2. ones 生成全 1 阵

调用格式为:

- $Y = \text{ones}(n)$ : 生成  $n \times n$  的全 1 矩阵; 如果  $n$  不是标量将给出出错信息。
- $Y = \text{ones}(m,n)$ : 生成  $m \times n$  的全 1 阵。
- $Y = \text{ones}([m\ n])$ : 生成  $m \times n$  的全 1 阵。
- $Y = \text{ones}(d_1,d_2,d_3\dots)$ : 生成  $d_1 \times d_2 \times d_3 \times \dots$  的全 1 阵。
- $Y = \text{ones}([d_1\ d_2\ d_3\dots])$ : 生成  $d_1 \times d_2 \times d_3 \times \dots$  的全 1 阵。
- $Y = \text{ones}(\text{size}(A))$ : 生成一个与矩阵 A 大小相同的全 1 阵。

#### 3. eye 生成单位阵

调用格式为:

- $Y = \text{eye}(n)$ : 生成  $n \times n$  的单位阵。
- $Y = \text{eye}(m,n)$ : 生成  $m \times n$  的矩阵, 其中对角线元素为 1, 其它元素为 0。
- $Y = \text{eye}([m\ n])$ : 生成  $m \times n$  的矩阵, 其中对角线元素为 1, 其它元素为 0。
- $Y = \text{eye}(\text{size}(A))$ : 生成一个与矩阵 A 大小相同的单位阵。

注意: 对于多维数组没有定义单位阵, 命令  $y = \text{eye}([2,3,4])$  将给出错误信息。

#### 4. rand 生成均匀分布的随机阵

调用格式

- $Y = \text{rand}(n)$ : 生成一个  $n \times n$  的随机阵; 如果  $n$  不是标量将给出出错信息。
- $Y = \text{rand}(m,n)$ : 生成一个  $m \times n$  的随机阵。
- $Y = \text{rand}([m\ n])$ : 生成一个  $m \times n$  的随机阵。
- $Y = \text{rand}(m,n,p,\dots)$ : 生成一个  $m \times n \times p \times \dots$  的随机阵。
- $Y = \text{rand}([m\ n\ p\dots])$ : 生成一个  $m \times n \times p \times \dots$  的随机阵。
- $Y = \text{rand}(\text{size}(A))$ : 生成一个与矩阵 A 大小相同的随机阵。
- $\text{rand}$ : 不带任何参数将产生一个随机数。

- `s = rand('state')`: 返回有 35 个元素的向量，包含当前均匀随机数发生器的状态。

说明：

- (1) `rand` 函数产生 0、1 之间均匀分布的随机数。

- (2) MATLAB5 以上的版本中采用一个新的多种子随机数发生器生成封闭区域

$[2^{-53}, 1 - 2^{-53}]$  内的随机浮点数。

- (3) `rand('state',s)`: 将当前的状态设为 `s`，`s` 必需是标量或是个有 35 个元素的列向量

- (4) `rand('state',0)`: 将发生器状态设为它的初始状态

- (5) `rand('state',j)`: 对于整数 `j`，将发生器设为第 `j` 个状态

- (6) `rand('state',sum(100*clock))`: 每次将发生器设为不同的状态

### 5. `randn` 正态分布随机阵

命令的形式为：

- `Y = randn(n)`: 生成  $n \times n$  的随机矩阵；如果 `n` 不是标量将给出出错信息

- `Y = randn(m,n)`: 生成一个  $m \times n$  的随机阵

- `Y = randn([m n])`: 生成一个  $m \times n$  的随机阵

- `Y = randn(m,n,p,...)`: 生成一个  $m \times n \times p \times \dots$  的随机阵

- `Y = randn([m n p ...])`: 生成一个  $m \times n \times p \times \dots$  的随机阵

- `Y = randn(size(A))`: 生成一个与矩阵 `A` 大小相同的随机阵

- `randn`: 不带任何参数，每次引用将产生一个随机数

- `s = randn('state')`: 返回包含当前正态随机数发生器的状态的有 2 个元素的向量

说明：

- (1) `randn` 函数产生 -1、1 之间正态分布的随机数；

- (2) `randn('state',s)`: 将当前的状态设为 `s`

- (3) `randn('state',0)`: 将发生器状态设为它的初始状态

- (4) `randn('state',j)`: 对于整数 `j`，将发生器设为第 `j` 个状态

- (5) `randn('state',sum(100*clock))`: 每次将发生器设为不同的状态

### 2.1.3 特殊矩阵的生成

在 MATLAB 中，提供了一些特殊矩阵。表 2-1 给出了生成特殊矩阵的一些命令，在此不作详细说明，请在使用时参看联机帮助。

表 2-1 特殊矩阵生成命令

命 令	功 能	命 令	功 能
<code>compan</code>	伴随阵	<code>magic</code>	魔方阵
<code>gallery</code>	Hightam 检验矩阵	<code>pascal</code>	Pascal 阵
<code>hadamard</code>	Hadamard 阵	<code>rosser</code>	经典对称特征值检验矩阵
<code>hankel</code>	Hankel 阵	<code>toeplitz</code>	Toeplitz 阵
<code>hilb</code>	Hilbert 阵	<code>vander</code>	Vander 阵
<code>invhilb</code>	逆 Hilbert 阵	<code>wilkinson</code>	Wilkinson 特征值检验矩阵

### 2.1.4 向量的生成

在 MATLAB 中，有多种方法生成向量，本节介绍下面几种。

#### 1. 利用冒号“：“生成向量

冒号“：“是 MATLAB 中最最有用的算子之一，它不仅可以用来作为数组下标，对数组元素进行引用、增加和删除，还可以用来生成向量。

冒号使用下列格式生成均匀等分向量：

(1)  $x = j:k$

如果  $j < k$ ，则生成向量  $x = [j, j+1, \dots, k]$ ；

如果  $j > k$ ，则  $x$  为空向量。

(2)  $x = j:i:k$

如果  $i > 0$  且  $j < k$  或  $i < 0$  且  $j > k$ ，则生成向量  $x = [j, j+i, j+2i, \dots, k]$ ；

如果  $i > 0$  且  $j > k$  或  $i < 0$  且  $j < k$ ，则  $x$  为空向量；

```
x1 = 1:5
x2 = 1:0.5:3
x3 = 5:-1:1
x1 =
    1     2     3     4     5
x2 =
    1.0000    1.5000    2.0000    2.5000    3.0000
x3 =
    5     4     3     2     1
```

#### 2. 利用 linspace 函数生成向量

`linspace` 函数生成线性等分向量，它的功能类似与于冒号算子“：“，但是它不象冒号算子那样，根据给定的起始值、增量和终止值那样控制生成的向量元素的个数，而是直接给出元素的个数，从而给出各个元素的值。

`linspace` 函数的格式为：

(1)  $x = \text{linspace}(a, b)$

生成有 100 个元素的行向量  $x$ ，它的元素在  $a$  和  $b$  之间线性分布。

(2)  $x = \text{linspace}(a, b, n)$

生成有  $n$  个元素的行向量  $x$ ，它的元素在  $a$  和  $b$  之间线性分布。

#### 3. 利用 logspace 函数生成向量

`logspace` 函数生成对数等分向量，它和 `linspace` 函数一样直接给出元素的数目。该函数用法如下：

(1)  $x = \text{logspace}(a, b)$

生成有 50 个元素的对数等分行向量  $x$ ， $x(1)=10^a$ ， $x(50)=10^b$ 。

(2)  $x = \text{logspace}(a, b, n)$

生成有  $n$  个元素的对数等分行向量  $x$ ， $x(1)=10^a$ ， $x(n)=10^b$ 。

(3)  $x = \text{logspace}(a, pi)$

生成有 50 个元素的对数等分行向量 x,  $x(1)=10^0$ ,  $x(50)=\pi$ .

```
x1 = linspace(1.2, 5, 4)
x2 = logspace(1, 2, 4)
x1 =
    1.2000    2.4667    3.7333    5.0000
x2 =
    10.0000   21.5443   46.4159  100.0000
```

## 2.2 数值运算基本函数及其应用

### 2.2.1 加, 减, 乘, 除, 乘方

本小节讲述矩阵和数组的算术运算。矩阵的算术运算操作是按照线性代数运算法则定义的；数组算术运算操作是按元素逐个执行的。

在 MATLAB 语言中，算术运算既可以使用运算符，也可以使用等效的算术运算函数，如表 2-2 所示。

表 2-2 算术运算符与等效的算术运算函数

名 称	算术运算符	算术运算函数
加法	A+B	plus(A,B)
一元运算加	+A	uplus(A)
减法	A-B	minus(A,B)
一元运算减	-A	uminus(A)
矩阵乘法	A*B	mtimes(A,B)
数组乘法	A.*B	times(A,B)
矩阵右除	A/B	mrddivide(A,B)
数组右除	A./B	rdivide(A,B)
矩阵左除	A\B	mlddivide(A,B)
数组左除	A.\B	ldivide(A,B)
矩阵乘方	A^B	mpower(A,B)
数组乘方	A.^B	power(A,B)

下面我们分别介绍这些运算符和函数的用法。

#### 1. 加法和减法运算

对于矩阵运算和数组运算来说，加法和减法是相同的，都可以由下面的命令执行加减法：

$C = A + B$  或  $C = \text{plus}(A, B)$

$C = A - B$  或  $C = \text{minus}(A, B)$

这里要求 A 和 B 的大小必须相同，因为加减运算是把 A 和 B 的对应元素相加减。如果 A 与 B 的大小不同，MATLAB 将自动给出错误信息。

特殊情况是 A 和 B 中有一个是标量，MATLAB 允许标量和任意大小的矩阵相加减，结果是把矩阵中的每个元素和这个标量相加减。

在 MATLAB 中算术运算符+和-可以作为一元运算符使用。+A 就是取 A，而-A 则是对 A 中的每个元素取负。

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

```
B = [1 0 1; 0 1 0];
```

```
C = A + B
```

结果为：

```
C =
```

2	2	4
4	6	6

```
C = C - 2
```

结果为：

```
C =
```

0	0	2
2	4	4

```
C = -C
```

结果为：

```
C =
```

0	0	-2
-2	-4	-4

## 2. 乘法运算

对于乘法以及后面的除法、乘方、转置运算采用点号“.”来区别矩阵运算和数组运算。

矩阵乘和数组乘的运算命令分别为：

矩阵乘： $C = A * B$  或  $C = \text{mtimes}(A, B)$

数组乘： $C = A.^{*}B$  或  $C = \text{times}(A, B)$

矩阵乘  $A * B$  是矩阵 A 和 B 的线性代数乘。更精确地说，对于非标量 A 和 B，矩阵 A 的列数必须等于矩阵 B 的行数。标量可以和任意大小的矩阵相乘。

数组乘  $A.^{*}B$  是数组 A 和 B 的对应元素相乘。除非其中一个是标量，否则 A 和 B 必须有相同的大小。

首先建立矩阵 A 和全 1 矩阵 B，它们的大小都是  $3 \times 3$  的：

```
A = [1 2 1; 2 1 2; 1 2 1];
```

```
B = ones(3)
```

```
B =
```

1	1	1
1	1	1
1	1	1

## (1) 矩阵乘

**C = A\*B**

结果为:

**C =**

4	4	4
5	5	5
4	4	4

## (2) 数组乘

**C = A.\*B**

结果为:

**C =**

1	2	1
2	1	2
1	2	1

## 3. 矩阵求逆

行数和列数相等的矩阵称为方阵，只有方阵有逆矩阵。方阵的求逆函数为：

**B = inv (A)**

该函数返回方阵 A 的逆阵。如果 A 是病态的或接近奇异的，则会给出警告信息。

在实际应用中，很少显式地使用矩阵的逆。在 MATLAB 中很少使用逆阵  $x = \text{inv}(A)*b$  来求线性方程组  $Ax = b$  的解，而是使用矩阵除法运算  $x = A\b$  来求解。因为 MATLAB 设计求逆函数 `inv` 时，采用的是高斯消去法，而设计除法解线性方程组时，并不求逆，而是直接采用高斯消去法求解，有效地减小了残差，并提高了求解的速度。因此，MATLAB 推荐尽量使用除法运算，少用求逆运算。

## 4. 除法运算

在线性代数中，只有矩阵的逆的定义，而没有矩阵除法的运算。而在 MATLAB 中，定义了矩阵的除法运算。矩阵除法的运算在 MATLAB 中是一个十分有用的运算。根据实际问题的需要，定了两种除法命令：左除和右除。

## (1) 矩阵除法

矩阵除的运算命令为：

矩阵左除：**C = A\b** 或 **C = mldivide (A, B)**矩阵右除：**C = A/B** 或 **C = mrdivide (A, B)**

通常矩阵左除不等于右除，如果 A 是方阵， $A\b$  等效于 A 的逆阵左乘 B 矩阵，也就是  $\text{inv}(A)*B$ 。如果 A 是一个  $n \times n$  的矩阵，B 是一个 n 维列向量，或是有若干这样的列的矩阵，则  $A\b$  就是采用高斯消去法求得的方程  $AX = B$  的解 X。如果 A 是病态的或接近奇异的，MATLAB 将会给出警告信息。

如果 A 是一个  $m \times n$  矩阵，其中 m 近似等于 n，B 是一个 m 维列向量，或是有若干这样的列的矩阵，则  $X = A\b$  是不定或超定方程组  $AX = B$  的最小二乘解。通过 QR 分解确定矩阵 A 的秩 k，方程组的解 X 每一列最多只有 k 个非零元素。如果  $k < n$ ，方程的解是不唯一的，用矩阵除法求得的最小二乘解是这种类型解中范数最小的。

$B/A$  大体等效于  $B*inv(A)$  ( $B$  右乘  $A$  的逆阵), 但在计算方法上存在差异, 更精确地,  
 $B/A = (A'\backslash B')'$ 。

## (2) 数组除法

数组除的运算命令分别为:

数组左除:  $C = A \backslash B$  或  $C = mldivide(A, B)$

数组右除:  $C = A / B$  或  $C = mrdivide(A, B)$

数组左除  $A \backslash B$  是数组  $B$  和  $A$  中对应元素相除, 即  $B(i,j)/A(i,j)$ 。除非其中一个是标量, 否则  $A$  和  $B$  必需有相同的大小。

数组右除  $A / B$  是数组  $A$  和  $B$  中对应元素相除, 即  $A(i,j)/B(i,j)$ 。除非其中一个是标量, 否则  $A$  和  $B$  必需具有相同的大小。

## (1) 矩阵除法

$A = [1 2 3; 0 1 0; 3 2 1];$

$B = [1 2 1]'$ ;

矩阵左除

$A \backslash B$

$ans =$

```
-0.7500
2.0000
-0.7500
```

矩阵右除

$B = B'$

$B =$

```
1 2 1
```

$B / A$

$ans =$

```
0.2500 1.0000 0.2500
```

## (2) 数组除法

首先使矩阵  $B$  的大小和  $A$  相同, 做如下组合:

$B = [B; B; B]$

$B =$

```
1 2 1
1 2 1
1 2 1
```

然后分别计算  $A$  和  $B$  的数组左除和右除

$A \backslash B$

Warning: Divide by zero.

$ans =$

```
1.0000 1.0000 0.3333
```

```

Inf 2.0000 Inf
0.3333 1.0000 1.0000
A ./ B (或 B.\ A)
ans =
1.0000 1.0000 3.0000
0 0.5000 0
3.0000 1.0000 1.0000

```

注：在  $A \setminus B$  的结果中，“inf”是 MATLAB 用来表示无穷大的专用变量，在 MATLAB 中，被零除或浮点溢出都不会按错误处理，而只是给出警告信息，同时用“inf”做出标记。

## 5. 乘方运算

矩阵乘方和数组乘方运算命令分别为：

矩阵的乘方运算： $C = A^p$  或  $C = \text{mpower}(A, p)$

数组的乘方运算： $C = A.^B$  或  $C = \text{power}(A, B)$

矩阵的乘方运算要求  $A$  是一个方阵，且  $p$  是一个标量， $A^p$  的意思是矩阵  $A$  的  $p$  次方。如果  $p$  是个整数，则  $A^p$  是矩阵  $A$  自乘  $p$  次；如果  $p$  是一个负整数，则首先对  $A$  求逆，然后将它自乘  $p$  次。

如果  $p$  不是整数，则涉及到特征值和特征向量的求解问题。例如，若已知矩阵  $A$  的特征值矩阵为  $D$ ，特征向量矩阵为  $V$ ，则有

$$A^p = V * D.^p / V$$

其中  $D$  是对角阵， $D.^p$  为数组的乘方。

数组的乘方  $A.^B$  就是矩阵  $A$  和  $B$  对应元素的乘方，即  $A(i,j)$  的  $B(i,j)$  次方。除非其中一个是标量，否则  $A$  和  $B$  必需有相同的大小。

如果  $p$  是标量， $A$  是矩阵， $p.^A$  是  $p$  的  $A(i,j)$  次方，而  $A.^p$  则是  $A$  中每个元素  $A(i,j)$  的  $p$  次方。

### (1) 矩阵乘方

$A = [1 2 3; 0 1 0; 3 2 1];$

$A ^ 2$

ans =

10	10	6
0	1	0
6	10	10

$A ^ 0.5$

ans =

1.0000 + 0.7071i	0.6667	1.0000 - 0.7071i
0	1.0000	0
1.0000 - 0.7071i	0.6667	1.0000 + 0.7071i

**A ^ -2**

```
ans =
0.1563 -0.6250 -0.0938
0 1.0000 0
-0.0938 -0.6250 0.1563
```

(2) 数组乘方

**B = eye(3);**

**A.^ B**

```
ans =
1 1 1
1 1 1
1 1 1
```

**2.^ A**

```
ans =
2 4 8
1 2 1
8 4 2
```

**A.^ 2**

```
ans =
1 4 9
0 1 0
9 4 1
```

## 2.2.2 三角和超越函数

MATLAB 提供的三角函数和超越函数及其功能如表 2-3 所示。严格地说，这些函数都是按照数组的运算规则进行运算的，即它们都是对数组中的每个元素做函数运算。这些函数的用法都很简单，这里不再举例。

表 2-3 三角和超越函数

函数名	功能	函数名	功能
sin	正弦	acsc	反余割
cos	余弦	sinh	双曲正弦
tan	正切	cosh	双曲余弦
cot	余切	tanh	双曲正切
sec	正割	coth	双曲余切
csc	余割	asinh	反双曲正弦
asin	反正弦	acosh	反双曲余弦
acos	反余弦	atanh	反双曲正切
atan	反正切	acoth	反双曲余切
atan2	四象限反正切	asech	反双曲正割
acot	反余切	acsch	反双曲余割
asec	反正割		

### 2.2.3 指数和对数函数

表 2-4 列出 MATLAB 中的指数和对数函数，这些函数是针对数组，按数组运算规则进行运算的，取数组内每个元素的函数值。

表 2-4 指数和对数函数

函数名	功 能	函数名	功 能
log2	以 2 为底的对数	log	自然对数
log10	常用对数	sqrt	平方根
exp	指数	pow2	以 2 为底的指数

### 2.2.4 复数函数

MATLAB 中用于复数运算的函数如表 2-5 所示，和前面讲到的三角函数一样，这些函数也是按照数组运算的规则进行运算的。

表 2-5 复数函数

函数名	功 能	函数名	功 能
abs	绝对值(复数的模)	imag	复数的虚部
angle	复数的相角	real	复数的实部
conj	复数的共轭	~isreal	是否为复数数组

### 2.2.5 数值处理

MATLAB 还提供了一些常用的数值处理函数，如表 2-6 所示。这些函数都比较简单，但在编程时也会经常用到。它们也是按照数组运算规则进行运算的。函数的使用很容易，这里就不再具体叙述了。

表 2-6 基本数组函数

函数名	功 能	函数名	功 能
fix	向零取整	mod	除法求余(与除数同号)
floor	向负无穷方向取整	rem	除法求余(与被除数同号)
ceil	向正无穷方向取整	sign	符号函数
round	四舍五入		

## 2.3 矩阵变换

常用的矩阵变换函数如表 2-7 所示。

表 2-7 矩阵变换函数

函数名	功 能	函数名	功 能
flplr	矩阵左右翻转	diag	产生或提取对角阵
flipud	矩阵上下翻转	tril	提取下三角阵
flipdim	矩阵沿特定维翻转	triu	提取上三角阵
rot90	矩阵逆时针旋转 90°		

### 2.3.1 矩阵旋转

矩阵旋转函数的用法如下：

- (1)  $B = \text{flplr}(A)$ : 矩阵 A 关于垂直轴沿左右方向进行列维翻转。
- (2)  $B = \text{flipud}(A)$ : 矩阵 A 关于水平轴沿上下方向进行行维翻转。
- (3)  $B = \text{flipdim}(A, \text{dim})$ : 矩阵 A 沿着特定的维进行翻转，A 可以是多维的。当 dim=1 时，数组按行维进行翻转，对于二维数组等同于命令  $\text{flipud}(A)$ ；当 dim=2 时，数组按列维进行翻转，对于二维数组等同于命令  $\text{flplr}(A)$ 。
- (4)  $B = \text{rot90}(A)$ : 矩阵 A 逆时针方向旋转 90°
- (5)  $B = \text{rot90}(A, k)$ : 矩阵 A 逆时针旋转  $k * 90^\circ$ ，其中 k 为整数

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

```
A =
```

```
1 2 3
4 5 6
```

```
B = flplr(A) % 矩阵 A 左右翻转
```

```
B =
```

```
3 2 1
6 5 4
```

```
C = flipud(A) % 矩阵 A 上下翻转
```

```
C =
```

```
4 5 6
1 2 3
```

```
D = flipdim(A, 1) % 矩阵 A 沿行翻转，等同于 flipud(A)
```

```
D =
```

```
4 5 6
1 2 3
```

```
E = rot90(A) % 矩阵 A 逆时针旋转 90°
```

```
E =
```

```
3 6
2 5
1 4
```

```
F = rot90(A,3) %矩阵 A 逆时针旋转 3*90°
```

```
F =
```

4	1
5	2
6	3

### 2.3.2 矩阵的产生或提取

#### 1. diag 函数

diag 函数用于产生对角矩阵或提取对角线元素。它的用法有如下几种：

(1)  $X = \text{diag}(v, k)$ : 当  $v$  是有  $n$  个元素的向量，返回方阵  $X$ ，它的大小为  $n + \text{abs}(k)$ ，向量  $v$  的元素位于  $X$  的第  $k$  条对角线上。 $k = 0$  表示主对角线， $k > 0$  为主对角线以上， $k < 0$  为主对角线以下。

(2)  $X = \text{diag}(v)$ : 将向量  $v$  的元素放在矩阵  $X$  的主对角线上，等同于上面  $k = 0$  的情况。

(3)  $v = \text{diag}(X, k)$ : 对于矩阵  $X$ ，返回列向量  $v$ ，它的元素由  $X$  的第  $k$  条对角线的元素构成。

(4)  $v = \text{diag}(X)$ : 返回  $X$  的主对角线元素，等同于上面  $k = 0$  的情况。

#### (1) 产生对角矩阵

```
v=[1 2 3];
```

```
diag(v, 0)
```

```
ans =
```

1	0	0
0	2	0
0	0	3

```
diag(v, -1)
```

```
ans =
```

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

#### (2) 提取矩阵的对角线元素

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

```
diag(X)
```

```
ans =
```

```

9
diag(X, 1)
ans =
2
6

```

## 2. 函数 tril 和 triu

函数 tril 用于提取下三角矩阵，用法如下：

- (1) L = tril(X): 返回 X 的下三角部分，其余部分用“0”补齐。
- (2) L = tril(X,k): 返回 X 的第 k 条对角线以下的元素，其余部分用“0”补齐。k = 0 是主对角线，k > 0 位于主对角线以上，k < 0 位于主对角线以下。

函数 triu 用于产生或提取上三角矩阵，用法如下：

- (1) U = triu(X): 返回 X 的上三角部分元素，其余部分用“0”补齐。
- (2) U = triu(X,k): 返回 X 的第 k 条对角线以上的元素，其余部分用“0”补齐。k = 0 是主对角线，k > 0 位于主对角线以上，k < 0 位于主对角线以下。

```

X = [1 2 3
      4 5 6
      7 8 9];
tril(X)
ans =

```

```

1   0   0
4   5   0
7   8   9

```

```

tril(X,1)
ans =

```

1	2	0
4	5	6
7	8	9

```

triu(X)
ans =

```

1	2	3
0	5	6
0	0	9

```

triu(X,-2)
ans =

```

1	2	3
4	5	6
7	8	9

## 2.4 输出

### 2.4.1 输出格式

MATLAB 语句的执行结果都可以在屏幕上显示，同时赋值给指定的变量，数据的显示格式由 `format` 命令来控制。`format` 只影响结果的显示，不影响其计算与存储。MATLAB 中所有的计算都采用双精度格式进行。

在 MATLAB 的命令窗口输入 `format` 命令，指定数据的输出格式，其后的数据输出都采用这种格式，直到出现下一个 `format` 命令为止。

表 2-8 给出了 `format` 命令的各种选项用来控制不同的输出格式。

如果输出矩阵中所有元素都是整数，那么不管 `format` 命令指定小数的位数是多少，该矩阵都将显示整数，但矩阵的元素只要有一个不是整数，输出就都按照 `format` 命令指定的位数进行。

表 2-8 `format` 命令的格式

<code>format</code> 命令	输出结果	例子
<code>format</code>	缺省输出，与 <code>short</code> 相同	
<code>format short</code>	5 位定点数	2.7183
<code>format long</code>	15 位定点数	2.71828182845905
<code>format short e</code>	5 位浮点数	2.7183e+000
<code>format long e</code>	15 位浮点数	2.718281828459045e+000
<code>format short g</code>	最佳 5 位定点数或浮点数	2. 7183
<code>format long g</code>	最佳 15 位定点数或浮点数	2.71828182845905
<code>format hex</code>	十六进制数	4005bf0a8b145769
<code>format bank</code>	(金融)精确到分	2.72
<code>format rat</code>	用有理分式表示	1457/536
<code>format +</code>	用 +, -, 空格表示正、负数和为零元素	+
<code>format compact</code>	压缩额外的空行	
<code>format loose</code>	显示变量之间插入空行	

```
format long
a = [1 0 1]
a =
    1     0     1
a = [1 0 1.01]
a =
1.000000000000000          0          1.010000000000000
```

### 2.4.2 特殊变量和常数

MATLAB 环境下有一些经常用到的特殊变量和常数，如下：

**ans:** 最近的结果。这是在 MATLAB 命令空间最常遇到的变量，用于自动存储未指定输出变量的表达式的输出结果。

**eps:** 浮点运算的相对精确度，这是 MATLAB 计算中的误差。

**realmax:** 计算机能够显示的最大浮点数。

**realmin:** 计算机能够显示的最小浮点数。

**pi:** 圆周率  $3.1415926535897\dots$ 。

**i, j:** 虚数单位。

**inf:** 无穷大，即“ $n/0$ ”或溢出的结果，MATLAB 定义了这个特殊的变量，可以避免由于被 0 除造成的浮点溢出。

**NaN:** 不是一个数值 (Not-a-Number)，指无效的数值。“ $0/0$ ”和“ $inf - inf$ ”、“ $inf/inf$ ”都会导致 NaN。

**flops:** 浮点运算的次数。比如“A”为  $N \times N$  矩阵，则“A+A”的运算次数 flops 等于  $N^2$  次。

**computer:** 计算机类型。

**version:** MATLAB 版本，字符串格式。

在命令行输入：

```
sin(pi/2)
```

则显示：

```
ans =
```

```
1
```

又如，我们计算机的最大浮点数为：

```
realmax
```

```
ans =
```

```
1.7977e+308
```

如果对该数值乘 1.1 倍，则结果为：

```
realmax*1.1
```

```
ans =
```

```
Inf
```

这表示该数值已经超出了该计算机的浮点数范围（溢出）。

如下内容说明了“NaN”的含义：

```
realmax*1.1 - realmax*1.1
```

```
ans =
```

```
NaN
```

```
realmax*1.1/(realmax*1.1)
```

```
ans =
```

```
NaN
```

# 第3章 数据可视化

对于大量的原始数据，很难直接从中找出内在的规律，而把这些数据用各种形式的图形表示出来，就可以把数据的很多内在规律直观地展示在人们面前。因此，数据可视化是一种不可缺少的重要手段。

MATLAB 在数据可视化方面提供了很强的功能，它可以把数据以多种形式加以表现。

本章将为大家介绍如何利用 MATLAB 的图形功能来处理自己的数据。

## 3.1 基本平面图形

### 3.1.1 怎么作图

#### 1. 线性坐标平面图形

绘制线性坐标平面图形的函数是 `plot`，对于不同的输入参数，该函数有不同的形式可以实现不同的功能，我们在这里分别介绍。

##### (1) `plot(y)`

当只有一个参数时，`plot` 以该参数的值为纵坐标，横坐标从 1 开始自动赋值为向量 [1 2 3 ...] 或其转置向量，向量的方向和长度与参数 `y` 相同。例如，下面的命令：

```
y = [4 2 9 3 7 5 0];
```

```
plot(y)
```

可以绘出如图 3-1 的曲线，其横坐标为向量 [1 2 3 4 5 6 7]。

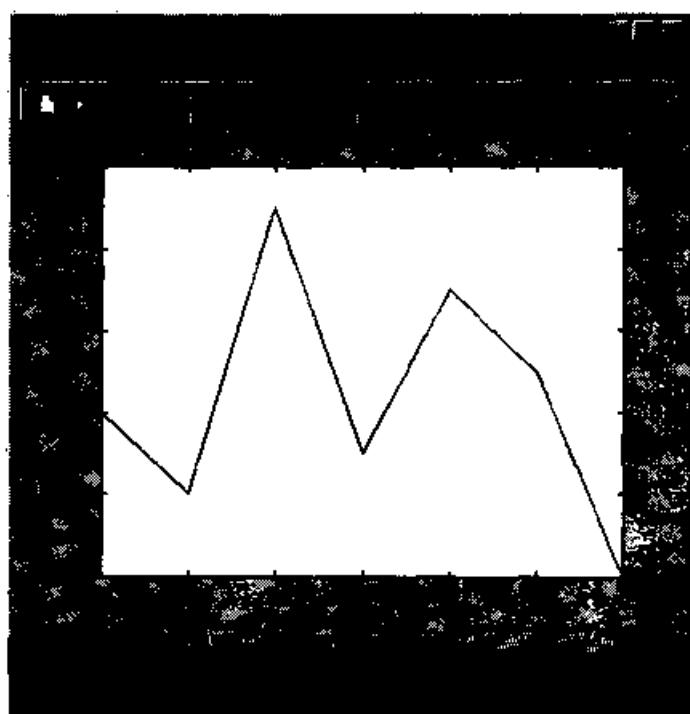


图 3-1 `plot(y)` 函数绘制的曲线

(2) `plot(x, y)`

这是最常用的形式。 $x$  为横坐标向量,  $y$  为纵坐标向量。例如, 下面的命令:

```
t=0:0.1:4*pi;
```

```
y=sin(t);
```

```
plot(t, y)
```

将绘出如图 3-2 的两个周期的正弦曲线。

如果  $x$  和  $y$  为标量, 那么该函数在窗口中画一个点。

**※ 注意:**  $x$  和  $y$  必须方向相同(行或列), 长度相等, 否则 MATLAB 将提示错误。

参数  $y$  还可以包括多个长度都和向量  $x$  相等的列向量, 这样可以在一个图形窗口同时绘制多条曲线, 这些曲线具有相同的横坐标。例如, 命令:

```
t=0:0.1:2*pi;
```

```
y=[sin(t); sqrt(t)];
```

```
plot(t, y)
```

可以绘制出如图 3-3 所示的正弦和平方根两条曲线。MATLAB 自动把不同的曲线绘制成不同的颜色, 而且在黑白打印机上输出时也会以不同的灰度来表示。

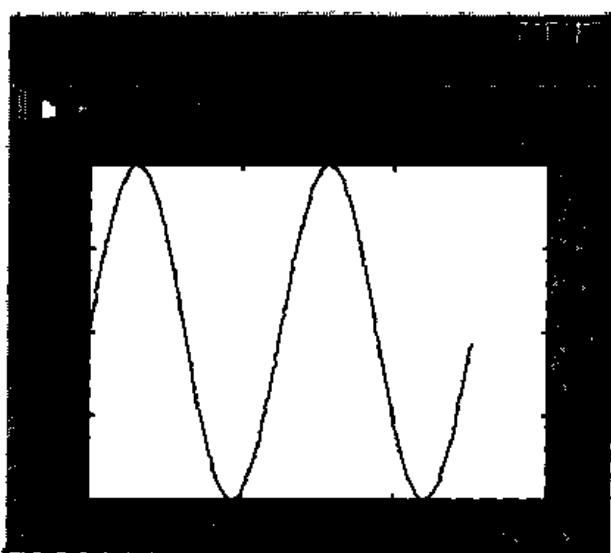


图 3-2 `plot` 函数绘制的正弦曲线

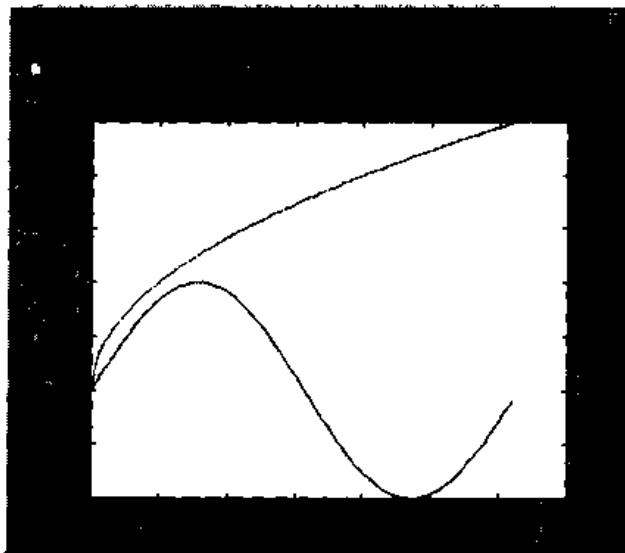


图 3-3 在一个窗口绘制两条曲线

(3) `plot(x1,y1,x2,y2,...)`

用这种形式也可以在同一窗口绘制多条曲线, 而且每条曲线的横坐标可以不同, 每一组向量也都可以有不同的长度。例如, 下面的命令:

```
t1=0:0.1:2*pi;
```

```
t2=0:0.1:4;
```

```
plot(t1,sin(t1),t2,sqrt(t2))
```

可以绘制出如图 3-4 的两条曲线, 它们的坐标位置不同, 而且长度也不同。

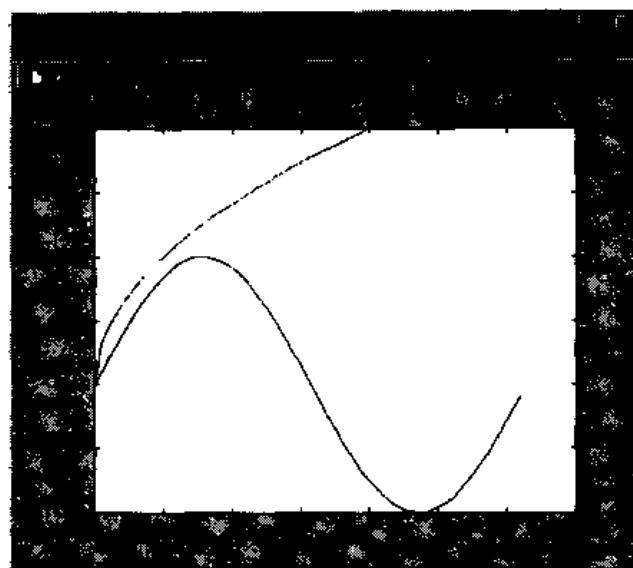


图 3-4 两条曲线具有不同的横坐标

(4) `plot(x, y, '选项')`

这里的‘选项’包括线型、颜色、数据点标记符号等特性的设置。常用的几种选项如表 3-1 所示。

表 3-1 常用的绘图选项

选 项	含 义	选 项	含 义
'-'	实线	'.'	用点号标出数据点
'--'	虚线	'o'	用圆圈标出数据点
'-.'	点线	'x'	用叉号标出数据点
'-.'	点划线	'+'	用加号标出数据点
'r'	红色	's'	用小正方形标出数据点
'g'	绿色	'd'	用菱形标出数据点
'b'	蓝色	'v'	用下三角标出数据点
'y'	黄色	'^'	用上三角标出数据点
'm'	洋红	用左三角标出数据点	
'c'	青色	用右三角标出数据点	
'w'	白色	'h'	用六角形标出数据点
'k'	黑色	'p'	用五角形标出数据点
'*''	用星号标出数据点		

利用表中这些选项可以把同一窗口中不同的曲线设置为不同的线型和颜色，可以只画出数据点，也可以在绘制的曲线上同时标出数据点。这些选项可以组合使用，例如选项“-r”表示绘制红色的虚线，“:yx”表示绘制黄色点线，同时用符号‘x’标记数据点。

下面我们用不同的线型和标注来重新画图 3-4 中的两条曲线，命令如下：

```
t1=0:0.4:2*pi;
t2=1:0.1:4;
plot(t1, sin(t1), ':ob', t2, cos(t2), '--g');
```

结果如图 3-5 所示，图中正弦曲线用蓝色点线表示，其数据点用‘o’绘制，余弦曲线

用绿色虚线表示。

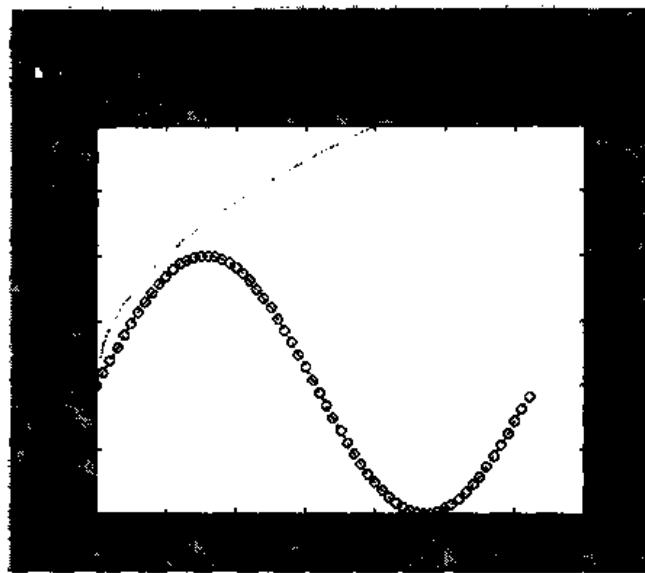


图 3-5 用不同的线型绘制曲线

## 2. 对数坐标曲线

函数 `semilogx`、`semilogy` 和 `loglog` 可以用来绘制二维对数坐标曲线，这几个函数的用法和 `plot` 函数相同。例如，如下命令：

```
t=0.1:0.1:2*pi;  
y=sin(t);  
semilogx(t,y)  
grid on
```

绘制出的正弦曲线的横坐标为对数坐标，并用“`grid on`”命令为图形窗口添加了网格。如图 3-6 所示。函数 `semilogy` 绘制的图形纵坐标为对数坐标，`loglog` 绘制的图形横纵坐标均为对数坐标。

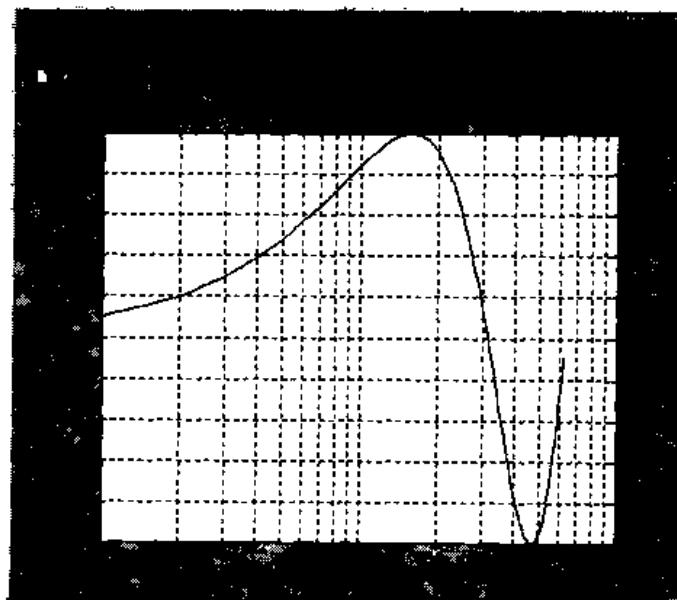


图 3-6 绘制对数坐标曲线

### 3. 双 y 轴图形

用 `plotyy` 函数可以绘制左右各有一个 y 轴的图形。该函数有以下几种常用的形式：

(1) `plotyy(x1,y1,x2,y2)`:

在一个图形窗口同时绘制两条曲线  $(x_1, y_1)$  和  $(x_2, y_2)$ ，曲线  $(x_1, y_1)$  用左侧的 y 轴，曲线  $(x_2, y_2)$  用右侧的 y 轴。

(2) `plotyy(x1,y1,x2,y2,FUN)`:

“`FUN`”是字符串格式，用来指定绘图的函数名，如 `plot`、`semilogx` 等。例如，命令：

`plotyy(x1,y1,x2,y2,'semilogy')`

就是用函数 `semilogy` 来绘制曲线  $(x_1, y_1)$  和  $(x_2, y_2)$ 。

(3) `plotyy(x1,y1,x2,y2,FUN1,FUN2)`:

和第二种形式类似，只是用“`FUN1`”和“`FUN2`”可以指定不同的绘图函数分别绘制这两条曲线。

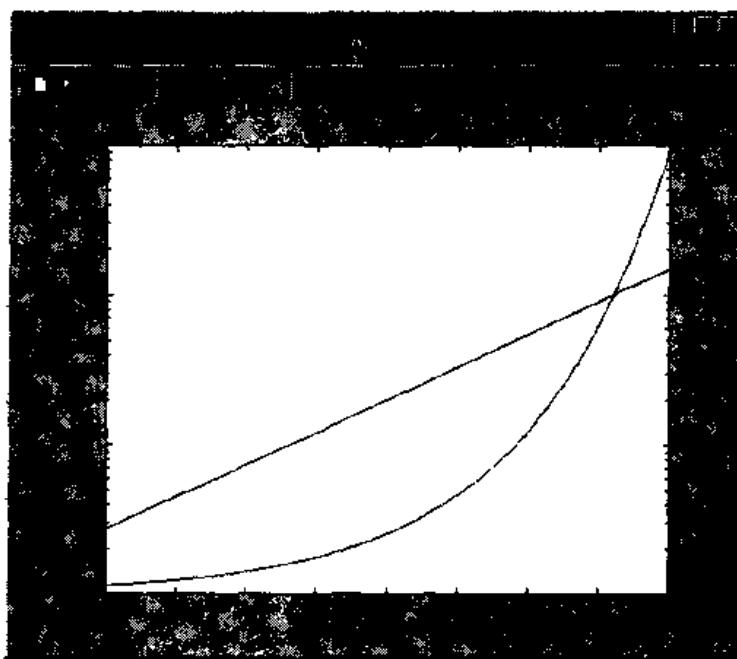


图 3-8 双 y 轴图形

例如，下面的命令：

```
x=1:0.01:5;
y=exp(x);
plotyy(x,y,x,y,'semilogy','plot')
```

用不同的函数、不同的 y 坐标轴绘制同一条指数曲线，如图 3-8 所示。左侧的 y 轴为对数坐标，图中由 `semilogy` 函数绘制的直线对应着这个 y 轴（在对数坐标系中，指数函数  $y=\exp(x)$  为一条直线）；右侧的 y 轴为线性坐标，图中由 `plot` 函数绘制的曲线对应着这个 y 轴。在彩色显示方式下每条曲线的颜色和相应的坐标轴颜色相同，而两个 y 轴的颜色（两条曲线的颜色）互不相同，以示区别。

注意：在 `plotyy` 函数中不能加入设置曲线的线型、颜色以及标出数据点的参数。如果要进行这种设置可以借助我们在第 7 章将要讲到的 `set` 函数。

#### 4. 图形窗口的分割

有时需要在一个图形窗口显示几幅图，这可以通过用 subplot 函数把图形窗口分割为几个子窗口来实现。

函数 subplot 的调用格式为：

`subplot(m, n, i)`

意为把图形窗口分割为 m 行 n 列子窗口，然后选定第 i 个窗口为当前窗口。例如：

`subplot(2, 2, 3)`

指的是把图形窗口分为 2 行 2 列共四个子窗口，选择第二行第一列（排序是 3）的子窗口为当前窗口进行操作。

下面我们举一个例子，用 subplot 函数把两种不同的图形综合在一个图形窗口。命令如下：

```
subplot(2,1,1)
```

```
t=0.1:0.1:2*pi;
```

```
y=sin(t);
```

```
semilogx(t,y)
```

```
grid on
```

```
subplot(2,1,2)
```

```
x=1:0.01:5;
```

```
y=exp(x);
```

```
plotyy(x,y,x,y,'semilogy','plot')
```

这一组命令绘制出图 3-12 中所示的两幅不同的图形。

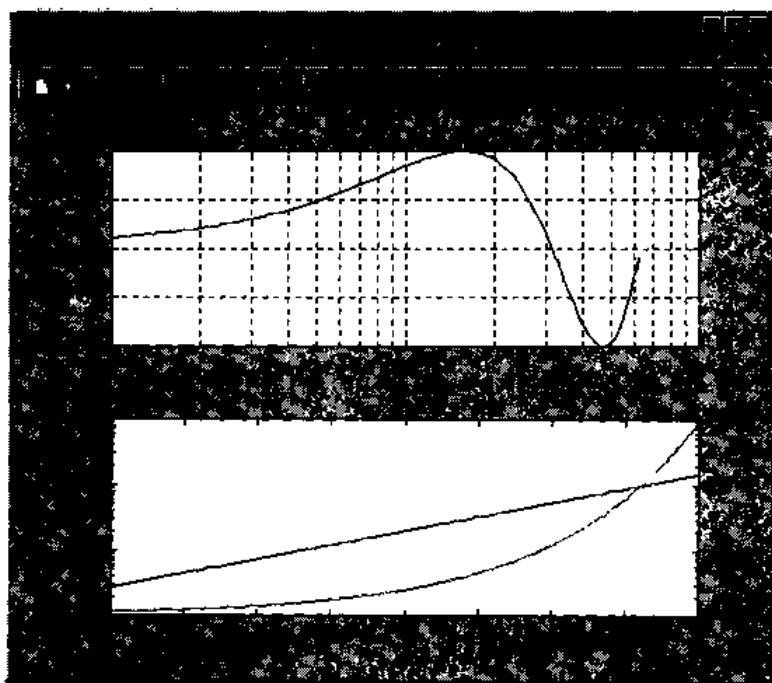


图 3-12 把一个图形窗口分为几个子窗口

## 5. 坐标系的调整

在前面的例子中我们已经看到, MATLAB 的绘图函数可以根据要绘制曲线数据的范围自动选择合适的坐标系, 使得曲线尽可能清晰地显示出来, 所以一般情况下用户不必自己选择绘图坐标。但对有些图形, 如果用户觉得自动选择的坐标不合适, 则可以用手动的方式选择新的坐标系。手动选择坐标系的工作可以由函数 `axis` 来完成, 该函数的调用格式为:

```
axis([xmin,xmax,ymin,ymax,zmin,zmax])
```

注意: 坐标的最小值 (`xmin, ymin, zmin`) 必须小于相应的最大值 (`xmax, ymax, zmax`), 否则会出错。

在这个函数中可以输入 4 个或 6 个参数, 分别对应于二维或三维坐标系的最小值和最大值。参数可以是数值也可以是某些函数。MATLAB 会按照用户指定的值来选择合适的坐标系。

例如, 如下命令绘制了正弦曲线, 其坐标系为自动坐标系, 图形可参见前面的图 3-2。

```
t=0:0.1:4*pi;  
y=sin(t);  
plot(t, y)
```

而我们现在想要调整该坐标系使得正弦曲线能充满整个窗口, 那么可以由下面的命令来完成:

```
axis([0,max(t),min(y),max(y)])
```

该命令执行后, 图 3-2 变为图 3-13 所示的坐标系。

坐标系的调整还可以用命令 `set` 来完成, 关于该命令的用法我们将在后面具体介绍。

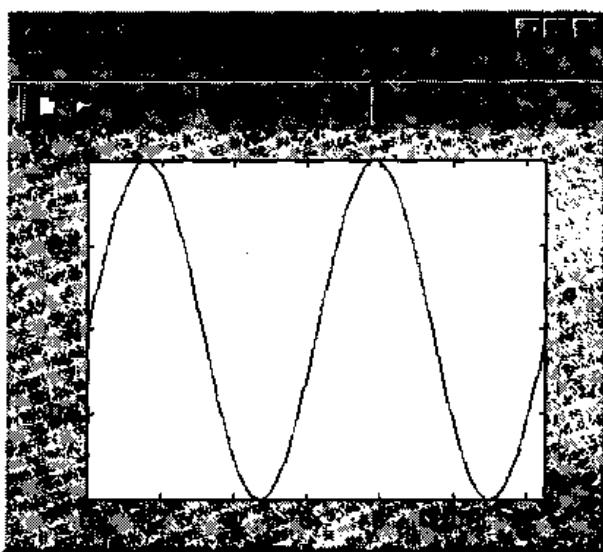


图 3-13 用 `axis` 函数调整后的坐标系

### 3.1.2 图形的标注

MATLAB 可以在画出的图形上加各种标注及文字说明, 以丰富图形的表现力。在中文 Windows 环境下, 还可以用汉字进行标注。

1. 为坐标轴加标注, 给图形加标题

给坐标轴 x、y、z 加标注只要调用相应的函数 xlabel、ylabel、zlabel。以函数 xlabel 为例，其调用格式为：

```
xlabel('text','property1',PropertyValue1,'property2',PropertyValue2,...)
```

其中 ‘text’ 为要添加的标注文本，‘property’ 指该文本的属性，“PropertyValue”为相应的属性值。该命令把文本 ‘text’ 按照指定的格式添加到 x 轴下方。

给图形加标题的函数为 title，它的调用格式和 xlabel 类似：

```
title('text','property1',PropertyValue1,'property2',PropertyValue2,...)
```

区别只是 title 函数把文本 ‘text’ 加到图形上方。

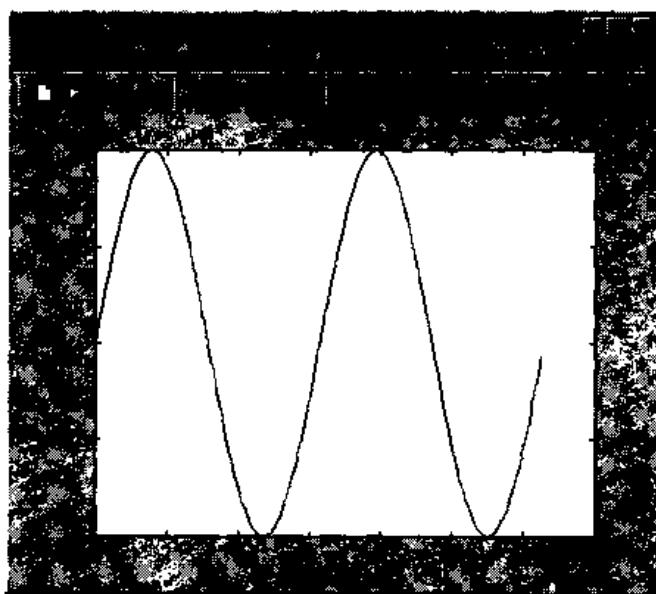


图 3-14 为图形加基本标注

例如，我们用如下命令来为图形添加标题和坐标轴标注：

```
t=0:0.1:4*pi;
y=sin(t);
plot(t,y)
xlabel('X 轴 (0~4\pi)')
ylabel('Y 轴')
title('正弦波','FontSize',12,'FontWeight','bold','FontAngle','italic')
```

结果如图 3-14 所示。

可以在函数 xlabel、ylabel 和 title 中的设置属性有很多，比如这里用到的 “FontWeight”（字体粗细）、“FontAngle”（字体角度）、“FontSize”（字体大小）等，我们将在后面结合图形属性一并介绍。

另外，在函数 title 的文本字符串中我们看到的 “\pi” 显示到图形标题上是字符 “π”，这里用特征字符串来表示 TeX 格式的字符，斜线 “\” 是引导特征字符串的标志符号。使用特征字符串可以把很多数学公式或工程中的 TeX 符号标注到图形上，大大增加了标注的灵活性。MATLAB 中的特征字符串及其对应的 TeX 符号如表 3-2 所示。

注意：特征字符串的大小写是有区别的。

表 3-2 特征字符串和 TeX 符号

特征字符串	符号	特征字符串	符号
\alpha	a	\beta	β
\gamma	γ	\delta	δ
\epsilon	ε	\zeta	ζ
\eta	η	\iota	ι
\theta	θ	\vartheta	ϑ
\kappa	κ	\lambda	λ
\mu	μ	\nu	ν
\xi	ξ	\pi	π
\rho	ρ	\tau	τ
\sigma	σ	\varsigma	ς
\upsilon	υ	\phi	ϕ
\chi	χ	\psi	ψ
\omega	ω	\Gamma	Γ
\Delta	Δ	\Theta	Θ
\Lambda	Λ	\Xi	Ξ
\Pi	Π	\Sigma	Σ
\Upsilon	Υ	\Phi	Φ
\Psi	Ψ	\Omega	Ω
\forall	∀	\exists	Ξ
\ni	∋	\equiv	≡
\cong	≅	\approx	≈
\neq	≠	\propto	∞
\leq	≤	\geq	≥
\pm	±	\div	÷
\times	×	\surd	￣
\otimes	⊗	\oplus	⊕
\cdot	·	\neg	¬
\cap	∩	\cup	∪
\supseteq	⊇	\subset	⊆
\subseteq	⊆		
\int	∫	\in	∈
\partial	∂	\wp	℘
\circ	○	\circ	◦
\emptyset	∅	\infty	∞
\Im	ℐ	\Re	ℜ
\aleph	ℵ	\wp	℘
\wedge	∧	\vee	∨
\langle	⟨	\rangle	⟩
\nabla	∇	\bullet	•
\sim	~	\leftrightarrow	↔
\leftarrow	←	\rightarrow	→
\uparrow	↑	\downarrow	↓
\clubsuit	♣	\diamondsuit	♦
\heartsuit	♥	\spadesuit	♠
\ceil	⌈	\ceil	⌈
\floor	⌋	\floor	⌋
\perp	⊥	\mid	
\dots	...	\prime	‘
\copyright	©		

## 2. 在图形窗口添加文本字符串

用 `text` 函数可以在图形窗口的任何位置加入文本字符串。该函数调用格式为：

```
text(x,y,'string')
```

这里，`x`、`y` 用于指定加入字符串的位置，“`string`”是需要添加的字符串，该字符串中也可以添加由“\”引导的特征字符串来表示特殊符号。

例如，我们用如下一组命令对前面的图 3-14 做标注：

```
text(pi, 0, '\leftarrow sin(pi) = 0')
```

```
text(4*pi-2, 0.8, '图形标注示例')
```

结果如图 3-15 所示。

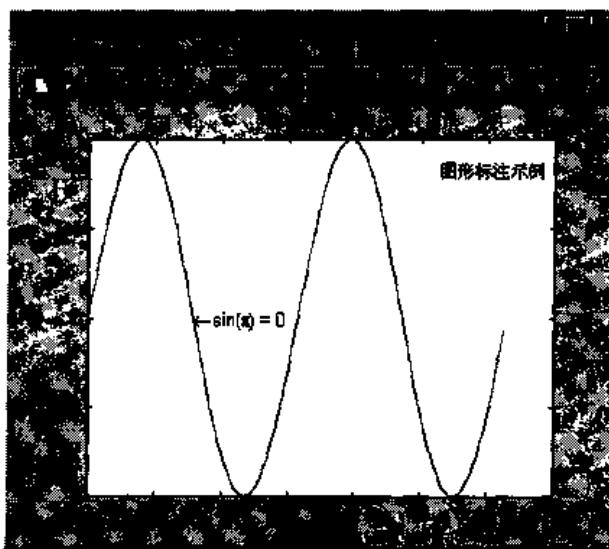


图 3-15 在图形中添加文本

从标注字符串的位置和 `x`、`y` 值之间的关系可见，`x`、`y` 是插入点在图形的坐标系中的坐标值。我们注意到在标注字符串中使用了特征字符串“\leftarrow”和“\pi”，分别表示左箭头“←”和符号“π”。

标注字符串还具有更加灵活的功能：可以在字符串中使用字符变量或返回值为字符形式的函数。例如，下面这组命令：

```
sinvalue=[-1;0';1'];
text(7*pi/2,-0.9,['\downarrow sin(pi*7/2)= ',sinvalue(1,:)])
text(pi/2,0.9,['\uparrow sin(pi/2)= ',num2str(sin(pi/2))])
text(0,-0.7,[' 绘图日期 : ',date])
text(0,-0.85,[' MATLAB 版本 : ',version])
```

在图形窗口的标注结果如图 3-16 所示。

需要注意在几条命令中字符串变量的使用方法：

命令“`text(7*pi/2,-0.9,['\downarrow sin(pi*7/2)= ',sinvalue(1,:)])`”中的“`sinvalue(1,:)`”是字符串数组“`sinvalue`”中的第一行的字符串“-1”，它和前面的字符串在中括号里连接为一个整体；

命令“`text(pi/2,0.9,['\uparrow sin(pi/2)= ',num2str(sin(pi/2))])`”中“`sin(pi/2)`”的结果是一

个实数，不是字符串，这种情况下不能直接和前面的字符串连接，必须首先用“`num2str`”函数把它转化为字符串格式：

最后两条命令类似，`date` 函数给出当前日期，`version` 函数返回使用的计算机类型，它们的返回值都是字符串格式，因而不用进行类型转换，可以直接和其它字符串连接。

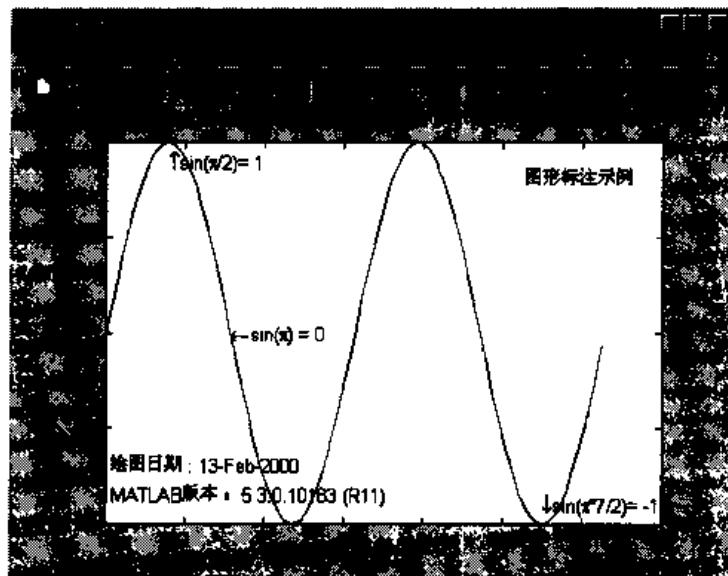


图 3-16 在标注字符串中使用变量

### 3. 图例

为图形添加图例便于对图形的观察和分析。添加图例由 `legend` 函数来完成，该函数的一般调用格式为：

```
legend(string1, string2, string3, ...)
```

只要指定标注字符串，该函数就会按顺序把字符串添加到相应的曲线线型符号之后。例如，输入如下一组命令：

```
t = 0:0.1:4*pi;
plot(t, sin(t), ':', t, cos(t), '*')
legend('Sin Wave', 'Cos Wave', 'None')
```

其标注结果如图 3-17 所示。从图中可见，图例在缺省情况下自动置于图形的右上角，并自动把线型和标注字符串按顺序排列到一起，由于图中只有两条曲线，所以，第三个字符串“`None`”不对应图中的任何曲线，因而其前面是空的。

MATLAB 还允许很方便地对图例进行调整：用鼠标左键点住图例拖动即可移动图例到需要的位置；用鼠标左键双击图例中的某个字符串就可以对该字符串进行编辑。

在 `legend` 函数中加入一个参数也可以对图例的位置作出设置，格式如下：

```
legend(string1, string2, string3, ..., Pos)
```

式中“`Pos`”的取值及含义如下：

`0` = 自动把图例置于最佳位置，即和图中曲线重复最少

`1` = 置于图形右上角（缺省值）

`2` = 置于图形左上角

`3` = 置于图形左下角

**4 = 置于图形右下角**

**-1 = 置于图形右侧（外部）**

当 Pos=0 时，每改变一次图形大小，图例都会重新选择一次最佳位置。图 3-18 所示即为 Pos=0 的图例位置。命令为：

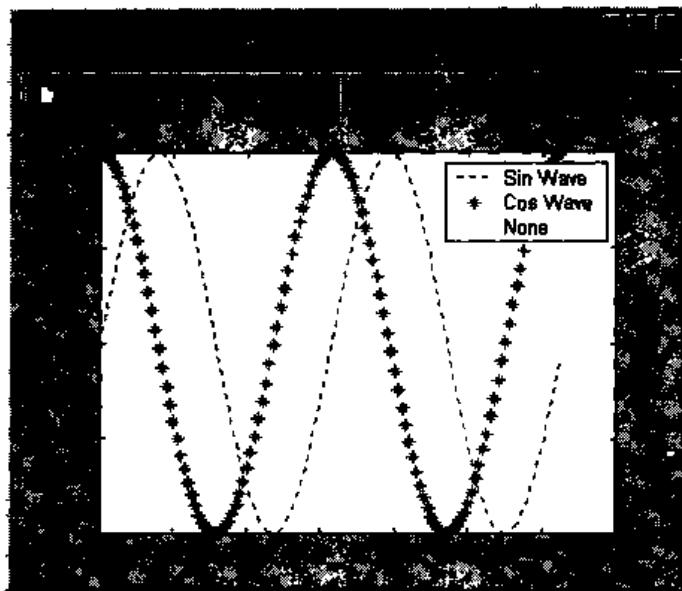


图 3-17 添加图例

```
legend('Sin Wave', 'Cos Wave', 'None', 0)
```

当 Pos=-1 时，图例会被置于图形外，这样层次更加分明，如图 3-19 所示。

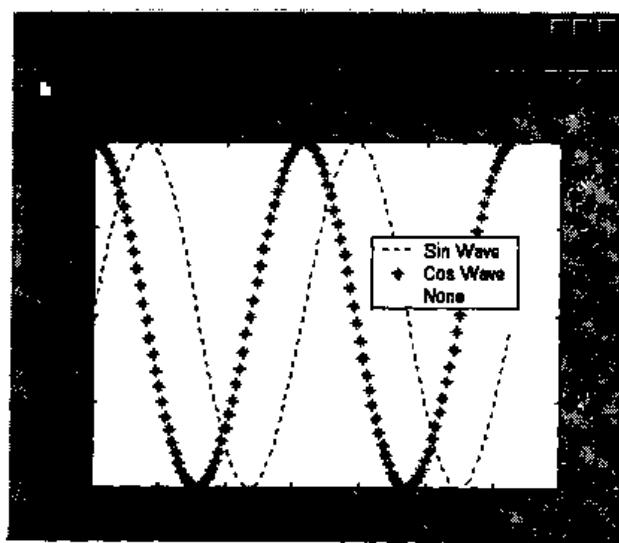


图 3-18 参数 Pos 为 0 时的图例位置

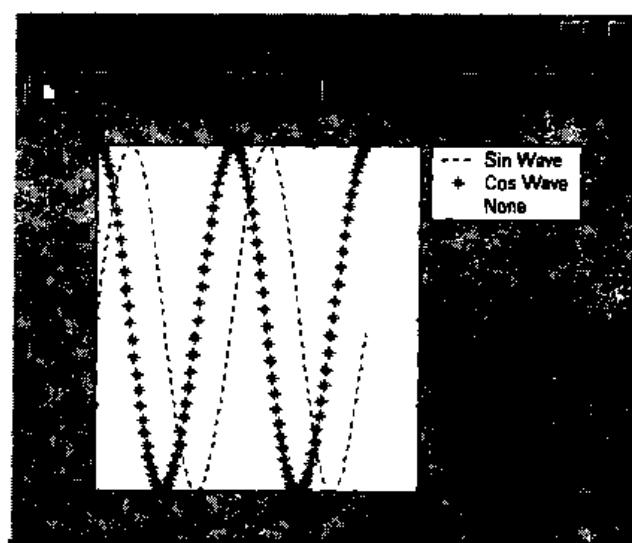


图 3-19 图例置于图形外

### 3.2 特殊平面图形

在很多研究领域，常常为满足一些特殊要求而需要采用特殊的平面图形，MATLAB 为此设计了几个专用于绘制特殊平面图形的函数，使得这项工作变得非常简单。

### 3.2.1 条形图

条形图常用于统计数据的作图。绘制条形图的函数如表 3-3 所示。

表 3-3 条形图函数

函数	功能	函数	功能
bar	竖直条形图	bar3	三维竖直条形图
barh	水平条形图	bar3h	三维水平条形图

这几个函数的调用方法类似，以函数 bar 为例，它有以下几种用法：

- 1) bar(X,Y): “X”是横坐标向量。“Y”可以是向量或矩阵，“Y”是向量时，每一个元素对应一个竖条；“Y”是 m 行 n 列的矩阵时，将画出 m 组竖条，每组包括 n 个条。
- 2) bar(Y): 横坐标使用缺省值 1:1:m。
- 3) bar(X,Y,width)或 bar(Y,width): 用“width”指定竖条的宽度，缺省宽度是 0.8。如果宽度大于 1，那么条与条之间将重合
- 4) bar(...,'grouped'): 产生组合的条形图。
- 5) bar(...,'stacked'): 产生堆叠的条形图。
- 6) bar(...,linespec): 指定条的颜色。
- 7) h = bar(...): 返回补片对象的句柄向量。

我们通过例子来具体体会一下条形图函数的用法。

先看一个最简单的情况：

```
x=1:12;
y=[-10,-6,5,10,20,25,30,24,22,19,10,6];
bar(x,y)
```

这里我们假想了某城市一年 12 个月的日平均气温，画出的条形图如图 3-20 所示。

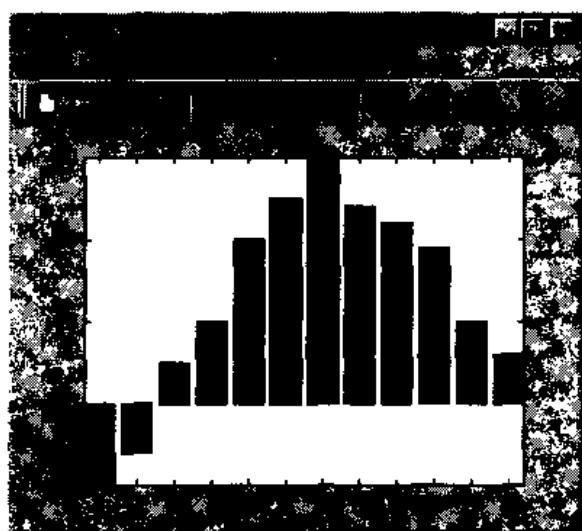


图 3-20 平均气温条形图

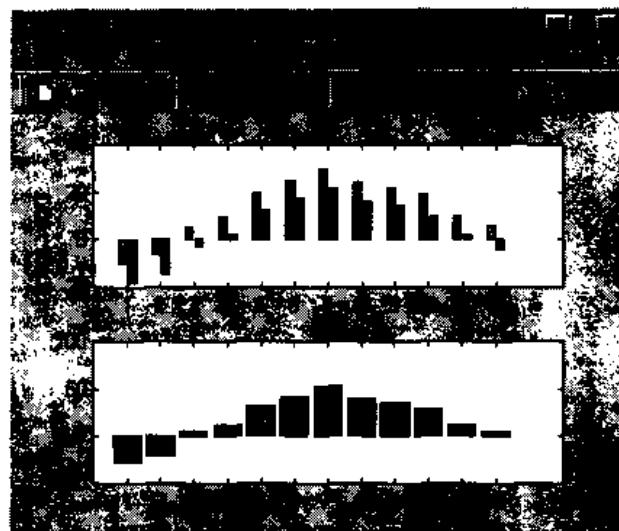


图 3-21 两种形式的条形图比较

现在我们加上夜平均气温的数据，看一下画出的条形图：

```
x=1:12;
Y=[-10,-6,5,10,20,25,30,24,22,19,10,6;
   -18,-14,-3,2,12,17,22,16,14,10,2,-4];
Y=Y';
colormap(cool)
subplot(2,1,1)
bar(x,y,'grouped')
subplot(2,1,2)
bar(x,y,'')
```

注意这里的 Y 必须转置为 12 行 2 列，它的行数须等于向量 x 的长度。命令 colormap 用来把当前颜色映像的色调改为“cool”。我们分别画出了“grouped”和“stacked”两种形式的条形图，结果如图 3-21 所示。

我们再来看一下三维条形图的画法，用如下一组命令：

```
x=1:12;
Y=[-10,-6,5,10,20,25,30,24,22,19,10,6;
   -18,-14,-3,2,12,17,22,16,14,10,2,-4];
Y=Y';
Y=[Y Y Y];
bar3(x,Y)
bar3(x,Y,'grouped')
bar3(x,Y,'stacked')
```

在这组命令中，我们把 Y 扩成 6 列的矩阵以便于比较，然后我们用三种形式分别绘制了这组数据的三维条形图，如图 3-22、3-23、3-24 所示。

画好的条形图也同样可以用 set 和 get 函数设置其状态，而且还可以用 shading 命令设置颜色的平滑方式。大家可以自己试验一下效果。

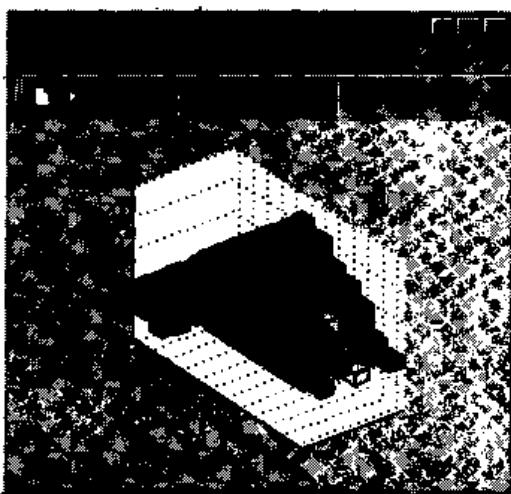


图 3-22 缺省形式的三维条形图

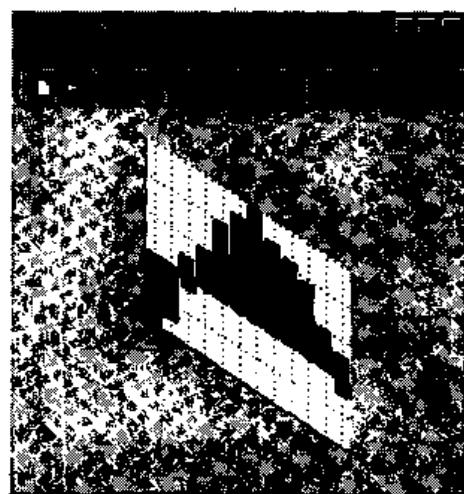


图 3-23 组合形式的三维条形图

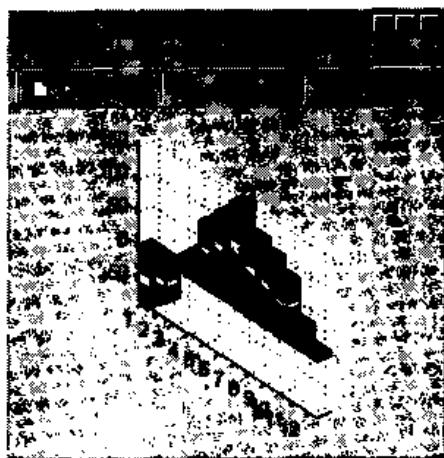
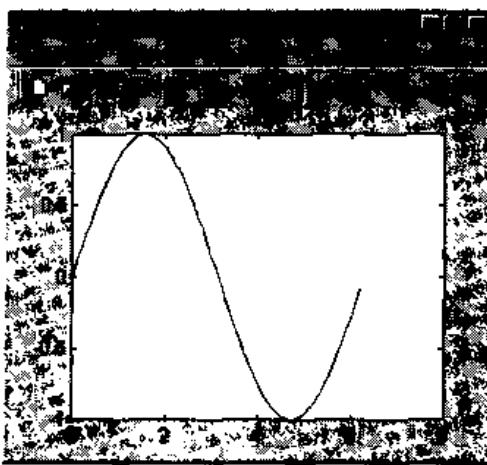


图 3-24 堆叠形式的三维条形图

### 3.2.2 复数向量图

当数据为复数时，使用 `plot` 函数绘制图形根据 `plot` 函数的不同调用方式会有两种不同的情况：

- 1) `plot(x,y)`: 这种方式下不论参数 `x` 和 `y` 哪个是复数，都将被忽略掉虚部数据，只绘制该复数的实部数据。其余用法和绘制一般实数数据相同。
- 2) `plot(z)`: 用这种方式绘制复数 `z` 的图形时，会以复数的实部为横坐标、虚部为纵坐标绘制实部和虚部的关系曲线，其作用相当于使用 `plot(real(z), imag(z))` 命令。

图 3-25 用 `plot(t, z)` 命令绘制的复数图形

我们来举例说明两种方式的区别。例如，如下一组命令在定义了复数 `z` 之后，用上述第一种方式画图：

```
t=0:0.1:2*pi;
x=sin(t);
y=cos(t);
z=x+i*y;
plot(t,z)
```

命令的执行结果为图 3-25 所示正弦曲线（复数 `z` 的实部），同时显示警告信息，提示

用户复数的虚部数据被忽略；如果把命令 `plot(t,z)` 改为 `plot(z)`，那么结果将为图 3-10 所示的正余弦关系曲线。

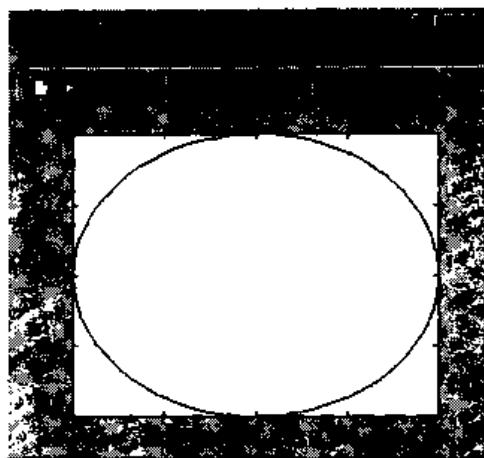


图 3-26 用 `plot(z)` 命令绘制的复数图形

### 3.2.3 直方图

直方图和条形图的形状类似，但作用不同，它主要用于显示数据的分布规律。

用于建立直方图的函数有 `hist` 和 `rose`，其中函数 `hist` 在直角坐标系中建立直方图，函数 `rose` 在极坐标系中建立直方图。

函数 `hist` 有如下几种主要用法：

- 1) `N = hist(Y)`: 把“Y”按其中数据的大小分为 10 个长度相等的段，统计每段中的元素个数并返回给“N”。如果“Y”是矩阵，那么按列分段。
- 2) `N = hist(Y,m)`: “m”是标量，用来设置分段的个数。
- 3) `N = hist(Y,X)`: “X”是向量，用于指定所分的每个数据段的中间值。
- 4) `hist(...)`: 不带输出参数时，直接绘制直方图。

下面我们来举例说明该函数的用法。

考虑随机数的分布规律，函数 `rand` 用于产生具有均匀分布规律的随机数，函数 `randn` 用于产生具有正态分布的随机数，我们来看一下二者的区别。如下一组命令用于产生不同的随机数并绘制直方图：

```
Y1=rand(10000,1);
hist(Y1,20)
Y2=randn(10000,1);
hist(Y2,30)
```

图 3-27 显示的是均匀分布随机序列的分布情况，图 3-28 是正态分布的随机序列。

如果“Y”是矩阵，我们来看结果如何：

```
Y=randn(10000,2);
hist(Y)
```

上述命令产生了一个 10000 行 2 列的正态分布随机数矩阵，它的分布规律的表示方式见图 3-29。

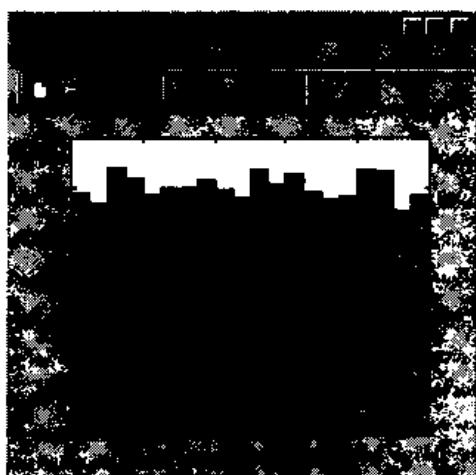


图 3-27 均匀分布随机序列

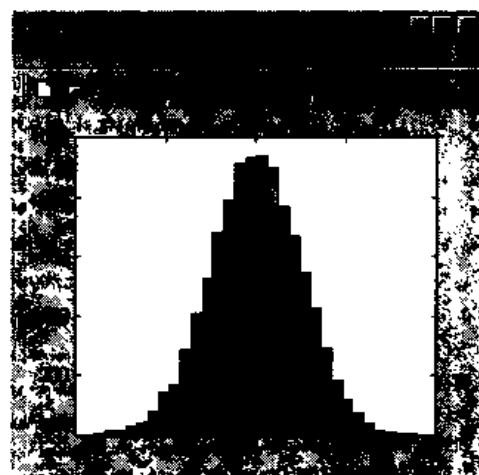


图 3-28 正态分布随机序列

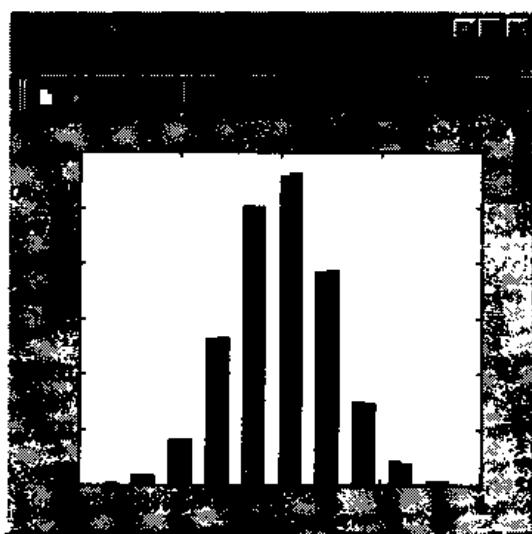


图 3-29 两列随机数的分布直方图

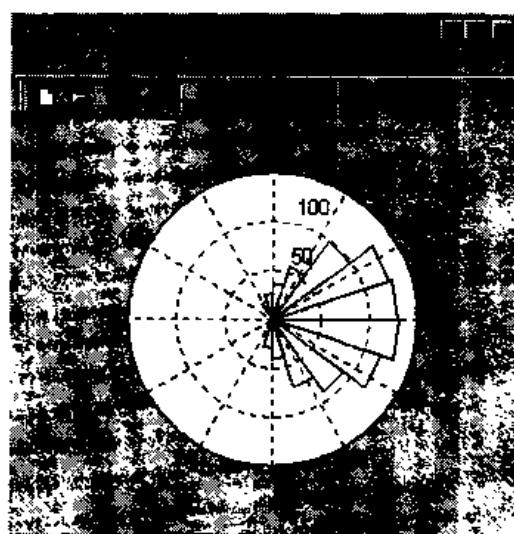


图 3-30 极坐标系中的直方图

在极坐标系中建立直方图的函数 `rose` 用法和 `hist` 类似，只是把数据作为弧度值处理，在极坐标系中建立直方图。

例如，如下命令用来在极坐标系中建立随机序列的直方图：

```
Theta=randn(1000,1);
Theta=pi*Theta/max(Theta);
rose(Theta)
```

式中的“`Theta`”是归一到区间 $[-\pi, \pi]$ 上的随机弧度值，随机数正态分布，这时的结果如图 3-30 所示。

### 3.2.4 极坐标曲线图

用 `polar` 函数绘制极坐标曲线。该函数的调用格式为：

```
polar(theta,rho,选项)
```

其中“`theta`”和“`rho`”分别为角度向量和幅值向量，要求向量“`theta`”和“`rho`”的长度相同，其“选项”的内容和用法与 `plot` 函数基本一致。

例如，下面的命令绘制了如图 3-31 所示的极坐标曲线：

```
thita=0:0.1:4*pi;
rho=(cos(thita/6)+0.5);
polar(thita, rho)
```

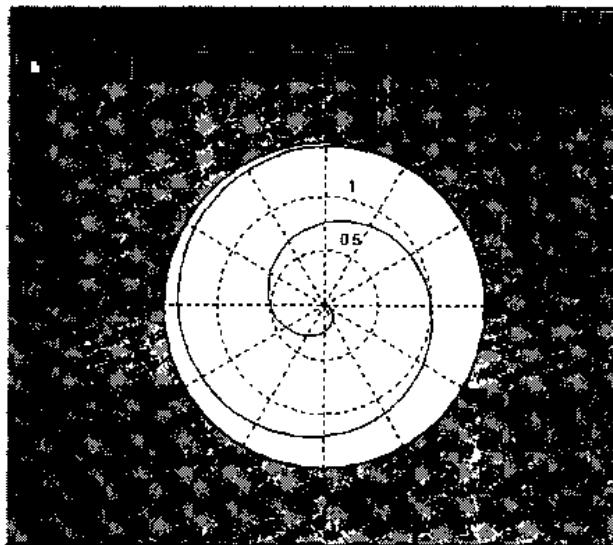


图 3-31 极坐标曲线

### 3.2.5 扇形图

扇形图用于显示向量中的元素所占向量元素总和的百分比。

函数 pie 和 pie3 分别用于绘制二维和三维扇形图。它们的用法类似，以函数 pie 为例，它常用以下几种调用方式：

- 1) pie(*x*): 绘制向量“*x*”中数据的扇形图。
- 2) pie(*x*, *explode*): 向量“*explode*”用于指定从扇形图中抽出一部分的块。它和向量“*x*”必须长度相等，向量“*explode*”中的非零向量对应的块将被抽出。
- 3) pie(..., *labels*): “*labels*”是用于标注扇形图的字符串数组，其长度必须和向量“*x*”相等。
- 4) *h* = pie(...): 返回包括补片和文本对象的句柄的向量。

例如，下面的命令用于建立某公司四个季度销售额“Sale”的二维扇形图：

```
Sale=[100 150 400 250]
```

```
pie(Sale, [0 0 0 1])
```

我们在向量 “[0 0 0 1]” 中指定把第四季度的扇形图块移出一些，结果如图 3-32 所示。

如下一组命令使用 pie3 函数绘制三维扇形图，为图中的饼块设置了标注文本，并对这些文本的位置进行调整以利于显示：

```
pie3(Sale, [0 0 1 0], {'Spring','Summer','Autumn','Winter'})
```

```
get(findobj(gca,'Type','Text','String','Autumn'),'Position')
```

```
ans =
```

```
-0.0000 -1.3500 0.3500
```

```
set(findobj(gca,'Type','Text','String','Autumn'),...  
    'Position',[0.6,-1.5,0.35])  
  
get(findobj(gca,'Type','Text','String','Summer'),'Position')  
ans =  
    -1.1746    0.4275    0.3500  
set(findobj(gca,'Type','Text','String','Summer'),...  
    'Position',[-1.4,0.4275,0.35])
```

在用 set 命令进行修改之前，首先用 get 命令获取文本对象的当前位置，以便于比较修改。结果如图 3-33 所示。

如果我们需要把图中的某一块区域去掉，那么首先要把绘图数据归一化，每个数据用其所占的百分比表示，然后把不显示的区域的数据从向量中去掉。

例如，如下一组命令把扇形图中的第一季度数据去掉：

```
sale=[0.11 0.17 0.44 0.28];
```

```
pie(sale(2:4))
```

结果如图 3-34 所示。

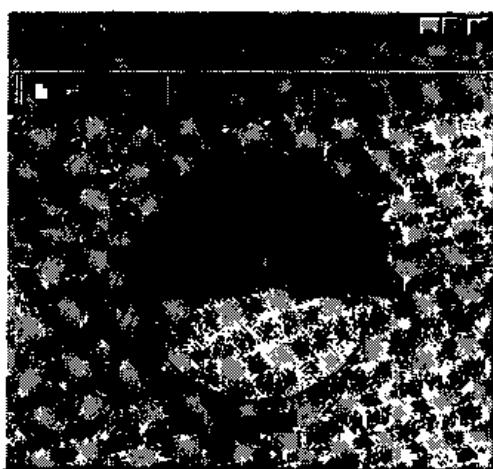


图 3-32 二维扇形图

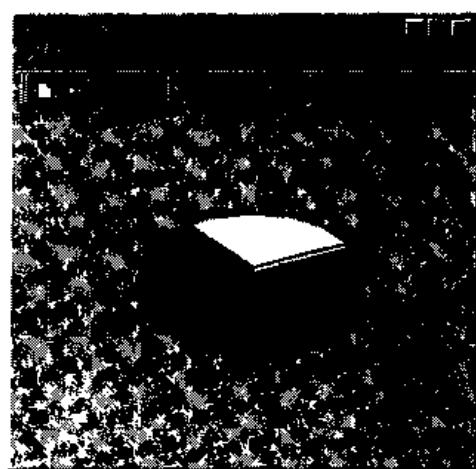


图 3-33 三维扇形图

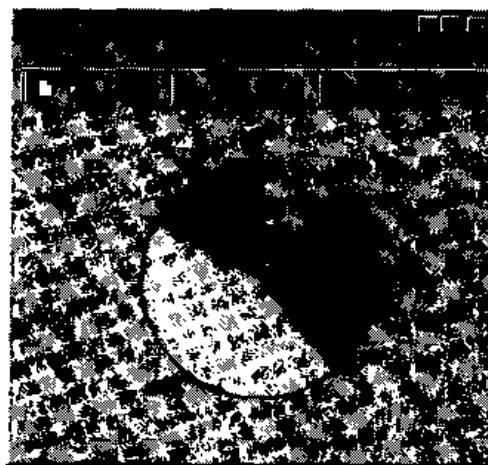


图 3-34 去掉了一块的扇形图

### 3.3 三维图形

#### 3.3.1 基本三维图

建立三维线条图的函数 `plot3` 和二维绘图函数 `plot` 相比区别只在于 `plot3` 多了第三维数据，调用格式都相同。

`plot3` 的一般调用方式为：

```
plot3(x, y, z, s)
```

式中 `x`、`y`、`z` 分别为第一到三维数据，可以是向量也可以是矩阵，但必须尺寸相等；`s` 是设置线型、颜色、数据点标记的字符串。

如下命令将绘制一个经典的三维螺旋线，并为其坐标轴加上标注：

```
t=0:0.1:8*pi;
plot3(sin(t),cos(t),t)
 xlabel('sin(t)', 'FontWeight', 'bold', 'FontAngle', 'italic')
 ylabel('cos(t)', 'FontWeight', 'bold', 'FontAngle', 'italic')
 zlabel('t', 'FontWeight', 'bold', 'FontAngle', 'italic')
```

结果如图 3-35 所示。

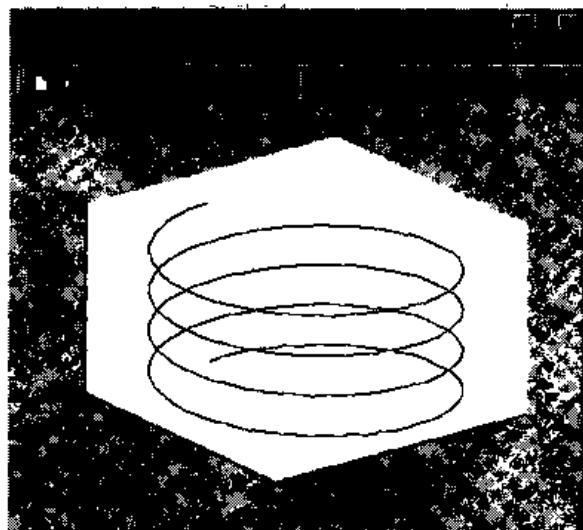


图 3-35 三维螺旋线

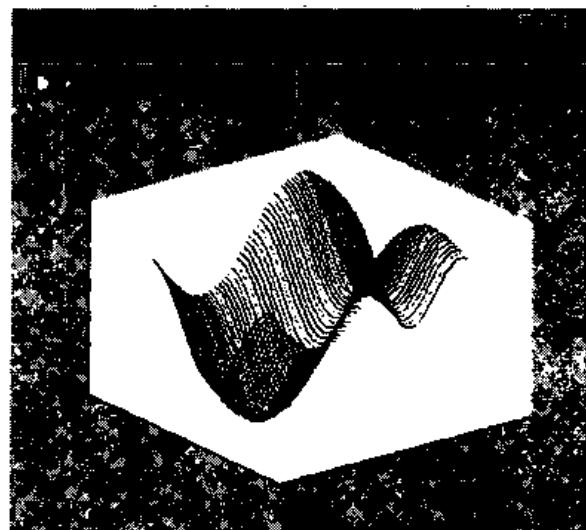


图 3-36 `plot3` 的参数为矩阵

我们想象如果图 3-35 中的 `z` 轴 (`t`) 去掉，也就是相当于从上往下看这幅图，那么它将是一个圆，这就和前面曾讲过的 `plot(sin(t), cos(t))` 绘制的曲线相同，由此可见，`plot3` 实际上就是二维函数 `plot` 在三维空间上的扩展。

下面再看一下 `x`、`y`、`z` 是矩阵的例子：

```
[X, Y]=meshgrid(-pi:0.1:pi);
Z=sin(X)-cos(Y);
plot3(X,Y,Z)
```

这里的函数 `meshgrid` 用于从向量生成方阵，如果输入参数是两个向量，那么生成的矩阵 `X` 的行和第一个输入向量相同，矩阵 `Y` 的列和第二个输入向量相同；输入参数是一个向

量时，相当于输入两个相同的向量，比如这里的矩阵 X 的每一行和矩阵 Y 的每一列都是向量 [-pi:0.1:pi]。绘制的三维曲线见图 3-36。在彩色显示方式下，这些曲线是用多种颜色绘制的，相邻曲线的颜色都不相同，便于区分。

### 3.3.2 线、面填色

图形的填充函数有两个：fill 和 patch，它们的用法基本相同，我们以函数 patch 为例说明一下这两个函数的使用。

使用如下一组命令，用“patch”函数建立一个五边形，并填充为蓝色，结果如图 3-37 所示。

```
patch([0 0.2 0.8 1 0.5 0], [1 0 0 1 1.8 1], 'b')
```

实质上 patch 函数是一个建立补片图形对象(patch)的低级函数，补片对象指坐标系中建立的多边形。如果多边形没有封闭，那么 patch 自动对它进行封闭。

在三维坐标系中建立补片，patch 函数的一般调用方式为：

```
patch(X, Y, Z, C)
```

式中“C”用来指定颜色。

下面举例说明函数 patch 在三维空间中建立补片的方法。首先用如下一组命令绘制一个立方体的框架，注意只是线条，没有面。

```
x=
[0 1 1 0 0 0
 1 1 0 0 1 1
 1 1 0 0 1 1
 0 1 1 0 0 0]

y=
[0 0 1 1 0 0
 0 1 1 0 0 0
 0 1 1 0 1 1
 0 0 1 1 1 1]

z=
[0 0 0 0 0 1
 0 0 0 0 0 1
 1 1 1 1 0 1
 1 1 1 1 0 1]

plot3(x, y, z)
xlabel('X 轴','FontWeight','bold')
ylabel('Y 轴','FontWeight','bold')
```

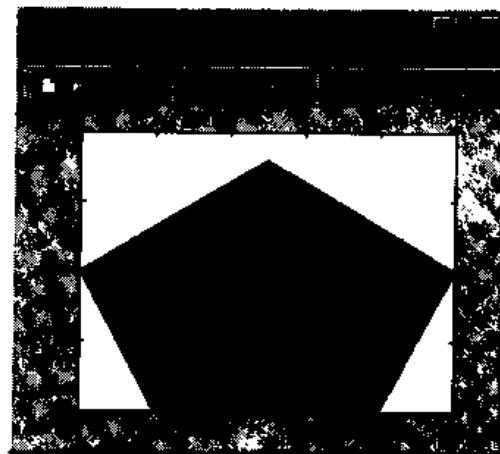


图 3-37 填充的图形

```
zlabel('Z 轴','FontWeight','bold')
```

下面我们用 `patch` 函数填充立方体框架位于 x 轴、y 轴和顶部的四边形，从而建立三个补片对象，命令如下：

```
patch(x(:,1),y(:,1),z(:,1),'y')
patch(x(:,4),y(:,2),z(:,1),'b')
patch(x(:,1),y(:,6),z(:,6),'g')
```

填充后的图形如图 3-38 所示。

函数 `fill3` 也用于填充三维多边形，其用法和 `patch` 基本相同，这里不再赘述。

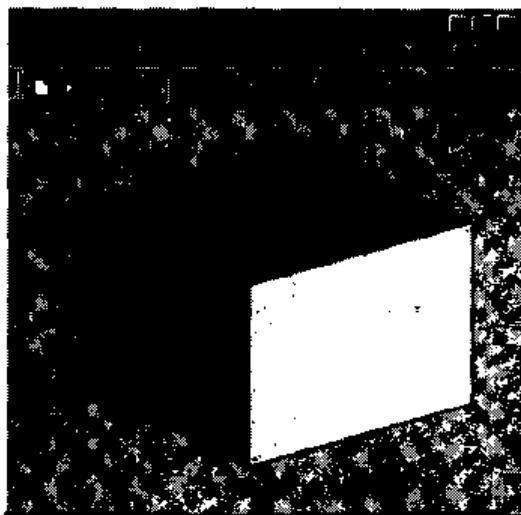


图 3-38 填充后的图形

### 3.3.3 三维数据的等高线和其二维表现

函数 `contour` 和 `contour3` 用来绘制二维和三维等高线，它们的调用方式相同，只是函数 `contour3` 要绘制相应的 z 轴。以 `contour` 为例，它的调用方式有：

- 1) `contour(Z)`: 直接绘制矩阵“Z”的等高线。
- 2) `contour(X,Y,Z)`: 用“X”和“Y”指定等高线的 x、y 坐标。
- 3) `contour(Z,n)` 和 `contour(X,Y,Z,n)`: 用标量“n”指定绘制等高线的线条数（从最低位置到最高位置所用的线条总数）。
- 4) `contour(Z,V)` 和 `contour(X,Y,Z,V)`: 向量“V”中的元素指定绘制等高线的位置，该向量的长度对应绘制的线条数。
- 5) `[C,H] = contour(...)`: 返回等高线矩阵“C”和列向量“H”，“H”是线条对象或补片对象的句柄。

例如，我们来绘制高斯三维分布曲面的等高线，绘制二维等高线的命令为：

```
[X,Y,Z]=peaks;
contour(X,Y,Z,15)
```

式中，我们用“X,Y”规定了等高线的坐标，并把等高线的线条数设为“15”。如图 3-39 所示。

如下命令绘制相应的三维等高线：

**contour3(Z,20)**

式中，我们把等高线条数设为 20，而没有规定 x-y 平面的坐标，如图 3-40 所示，请大家对比一下二者的区别。

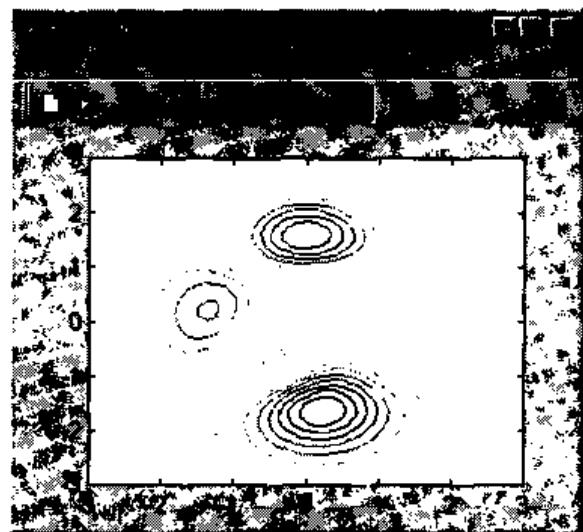


图 3-39 二维等高线

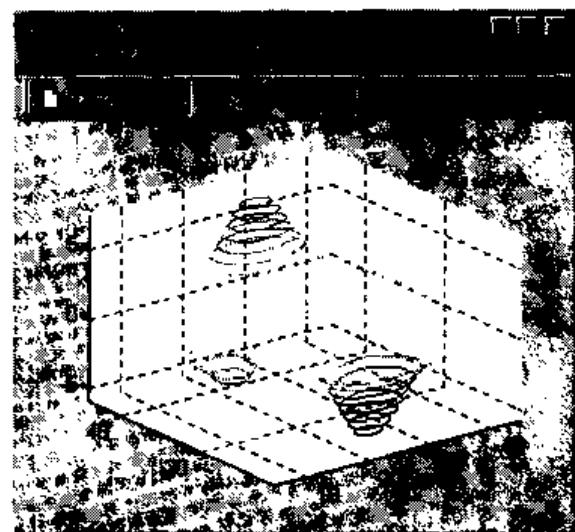


图 3-40 三维等高线

绘制好的等高线可以用 **clabel** 函数标注高度值，如下一组命令示意了 **clabel** 函数的基本用法：

**[C,H]=contour(Z);**

**clabel(C,H)**

标注结果如图 3-41 所示。

MATLAB 下还可以用 **contourf** 函数绘制填充的二维等高线图，该函数基本用法和函数 **contour** 相同。例如，下面的命令即可绘制填充的二维等高线图：

**contourf(Z,15)**

结果见图 3-42。

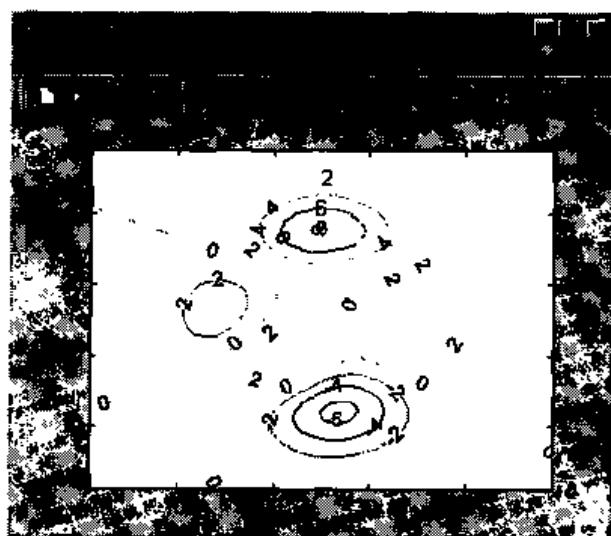


图 3-41 标注等高线的高度值



图 3-42 填充的二维等高线图

### 3.3.4 曲面与网线图

所谓网格图，是指把相邻的数据点连接起来形成网状曲面。建立网格图的常用函数是 `mesh`，还有两个建立特殊网格图的函数：`meshc` 和 `meshz`。

下面我们来看一个用 `mesh` 函数产生网格图的例子：

```
[X,Y] = meshgrid(-8:0.5:8);
R = sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(X,Y,Z)
```

上述命令产生的网格图如图 3-43 所示。这里绘制的是经典的 sinc 函数的三维网格图，形状类似草帽。

如果在彩色显示方式下，我们会看到网格图的线条具有不同的颜色，而且色彩随着高度（沿着 z 轴）变化，MATLAB 有一套缺省的着色方案，当然也可以自己规定如何着色，关于颜色的调节我们将在下一节讲述。

在图 3-43 中，我们发现网格线之间的区域是不透明的，因而显示的网格只是前面的部分，被遮住的部分没有显示出来。MATLAB 用 `hidden` 函数控制网格图的这个特性：`hidden on` 表示隐藏被遮住的部分，`hidden off` 表示显示遮住的部分。例如，在绘制图 3-43 的一组命令之后加上命令：

`hidden off`

结果将如图 3-44 所示。

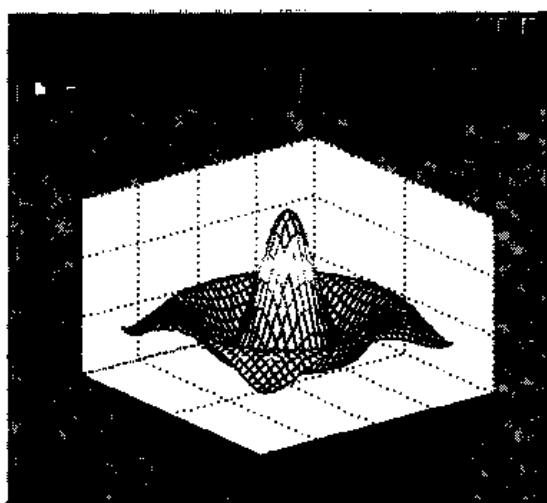


图 3-43 草帽图

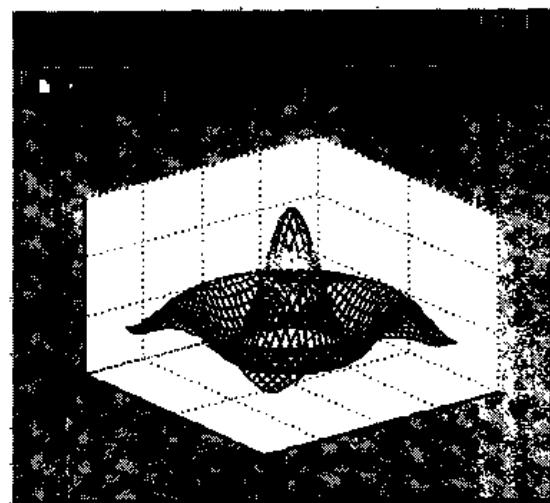


图 3-44 显示被遮住部分的效果

函数 `mesh` 也可以在调用时使用更多的参数来设置网格的特性，用法和其它绘图函数类似，这里不再赘述，请大家在使用时参考 MATLAB 帮助。

还有两个和 `mesh` 类似的函数：`meshc` 和 `meshz`。下面的例子说明了这两个函数的功能：

```
[X,Y,Z]=peaks(30);
meshc(X,Y,Z)
meshz(X,Y,Z)
```

上述命令中的 **peaks** 函数用于生成三维高斯型分布的数据，其参数为生成数据矩阵的维数。

命令“**meshc(X,Y,Z)**”生成具有基本等高线的网格图，如图 3-45 所示；命令“**meshz(X,Y,Z)**”生成带有基准平面的网格图，如图 3-46 所示。

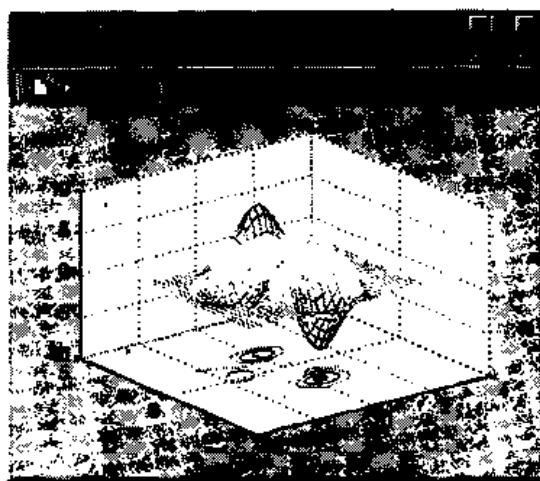


图 3-45 带有基本等高线的网格图

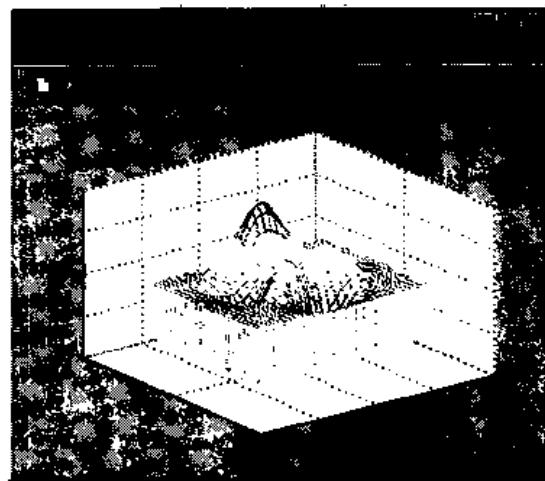


图 3-46 带有基准平面的网格图

表面图是指把网格图表面的网格围成的小片区域（补片）用不同的颜色填充形成的彩色表面。建立网格图的常用函数是 **surf**，还有两个类似的函数：**surfc** 和 **surfl**。

函数 **surf** 的用法和 **mesh** 基本相同，例如，如下一组命令将建立填充了网格的草帽图，如图 3-47 所示：

```
[X,Y] = meshgrid(-8:0.5:8);
R = sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
surf(X,Y,Z)
```

我们看到表面图的线条色为黑色，补片是彩色的，这也是它和网格图的区别所在。

有时想要得到表面图的整体效果，那么其中的线条应该去掉，并可以对颜色作平滑和插值处理，这需要借助函数 **shading**。它有三种用法：

- 1) **shading flat**: 去掉各片连接处的线条，平滑当前图形的颜色。
- 2) **shading interp**: 去掉连接线条，在各片之间使用颜色插值，使得片与片之间以及片内部的颜色过渡都很平滑。
- 3) **shading faceted**: 缺省值，带有连接线条的曲面。即如图 3-47 的效果。

如果在建立图 3-47 的一组命令后加上命令：

**shading flat**

那么将产生如图 3-48 所示的效果，网格去掉了，但片与片之间的过渡还比较明显。

若加上如下命令：

**shading interp**

**axis off**

那么将产生一个非常平滑、完整的效果，如图 3-49 所示。

函数 `surf` 也有两个类似的函数：`surf` 和 `surfl`。下面的例子说明了这两个函数的功能。如下命令：

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

绘制具有基本等高线的表面图，如图 3-50 所示。

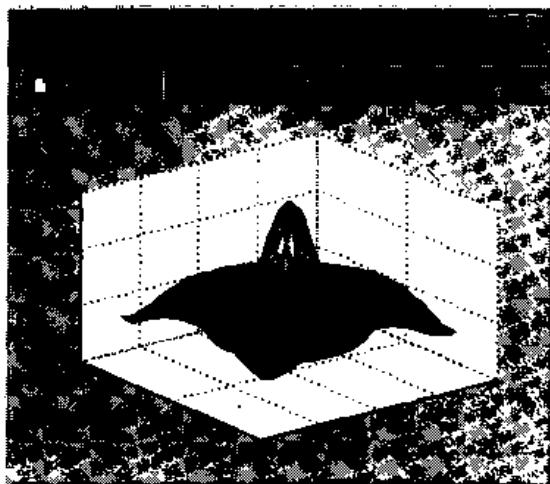


图 3-47 草帽形的表面图

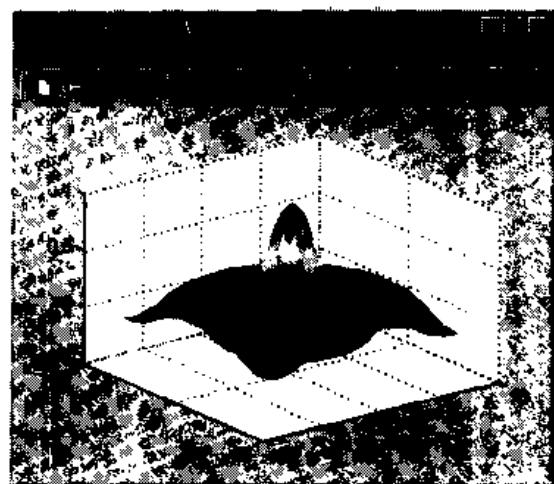


图 3-48 平滑后的效果

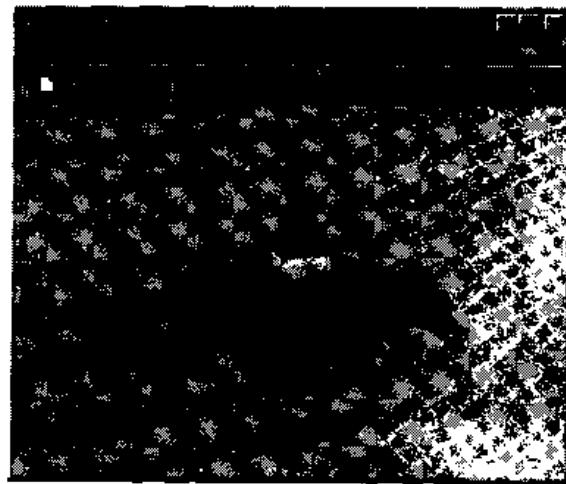


图 3-49 使用颜色插值后的效果

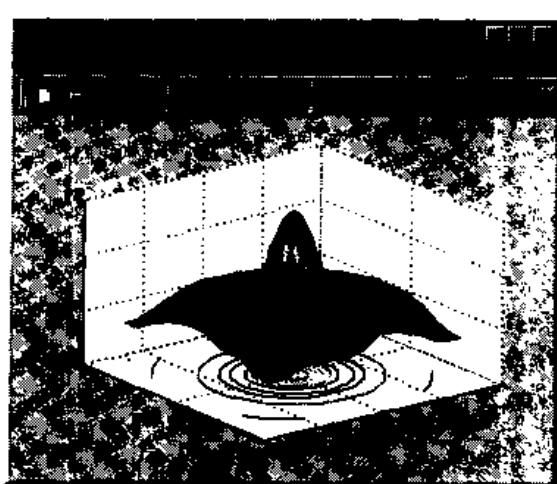


图 3-50 具有基本等高线的表面图

如下命令：

```
surfl(X,Y,Z)
```

绘制具有光照效果的表面图，如图 3-51 所示。这方面的知识我们将在下一节具体介绍。

MATLAB 还提供了一个建立表面对象的低级函数：`surface`。该函数用于把表面对象添加到当前坐标系中，例如，如下命令的执行结果和调用 `surf` 函数是一样的，也生成如图 3-47 所示的彩色表面图：

```
[X,Y] = meshgrid(-8:0.5:8);
R = sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(X,Y,Z)
```

**surface(X,Y,Z)**

但如果在调用 `surface` 函数之前没有用 `mesh` 命令生成网格图，那么直接运行 `surface` 的结果将如图 3-52 所示，这实际上是图 3-47 中的表面图向 x-y 平面的投影。

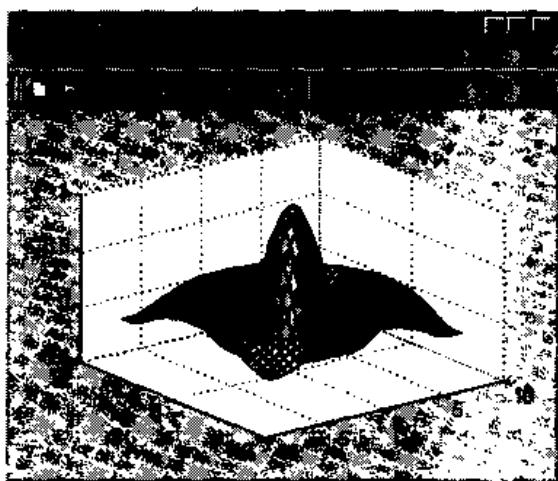


图 3-51 具有光照效果的表面图

图 3-52 直接执行 `surface` 函数的结果

### 3.3.5 图的表现

#### 1. 颜色映像

对于比较复杂的图形或图像，仅使用几种固定的颜色（如：绿色（'g'），蓝色（'b'））是远远不够的，MATLAB 提供了一种灵活而且通用的方式：颜色映像，也就是通常所说的 RGB 颜色。

颜色映像就是把三基色——红色（'R'）、绿色（'G'）、蓝色（'B'）按照不同的比例组合起来，形成新的颜色。颜色映像的数据结构是若干行三列的矩阵，矩阵元素为 0~1 之间的数，这些数表示相应颜色的强度。

常用的一些颜色的映像如表 3-4 所示。

表 3-4 常用颜色的映像

R (红色)	G (绿色)	B (蓝色)	映像的颜色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青
2/3	0	1	天蓝
1	1/2	0	橘黄
0.5	0	0	深红
0.5	0.5	0.5	灰色

MATLAB 提供了几种典型的颜色映像，它们各侧重于不同的色调，这些颜色映像矩阵见表 3-5。

表 3-5 几种典型色调的颜色映像

颜色映象	颜色范围
hsv	色调饱和值：从红色开始，依次经过黄、绿、青、蓝、紫，最后再回到红色。这种颜色映像尤其适合于周期函数。
hot	从黑到红到黄到白
gray	线性灰度
bone	带一点蓝色调的灰度
copper	线性铜色调
pink	粉红，柔和的色调
white	白色
flag	交替的红色、白色、蓝色和黑色
lines	线性颜色
colorcube	增强的颜色立方
vga	Windows 的 16 位颜色映像
jet	hsv 的一种变形（以蓝色开始和结束）
prism	棱镜。交替的红色、橘黄色、黄色、绿色和天蓝色
cool	青和洋红的色调
autumn	红、黄色调
spring	洋红、黄色调
winter	蓝、绿色调
summer	绿、黄色调

表中的这些颜色映像在缺省情况下生成的矩阵都是 64 行 3 列，当然行数是可以用参数设置的。

MATLAB 中使用的缺省颜色映像是“jet”。

在前面讲过的 mesh、surf、patch 等一些函数中可以直接运用颜色映像，在调用函数时添加相应的参数即可：

```
mesh(X, Y, Z, C)
surf(X, Y, Z, C)
patch(X, Y, Z, C)
```

其中的矩阵“C”用于设置图形的颜色变化，它的大小应该和矩阵“Z”相同，其中的元素指定矩阵“Z”中的数据如何使用当前颜色映像中的颜色。如果不带有参数“C”，即如以前的调用格式，那么 MATLAB 自动用矩阵“Z”本身的数据来调节颜色映像中的颜色，也就相当于如下调用格式：

```
mesh(X, Y, Z, Z)
```

颜色映像中的色彩和数据的对应关系前面已经讲过，这种方式只反映了 z 方向的数据变化，不够灵活，我们可以根据需要自己规定使用颜色的方式。

下面我们以函数 mesh 为例说明设置颜色矩阵 “C” 的用法。

仍考虑前面讲过的草帽图，用下面一组命令来绘制：

```
[X,Y] = meshgrid(-8:0.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R) ./ R;
```

下面的命令以不同方向的数据来控制颜色的运用，因而颜色是沿着各个方向按照指定的函数规律变化的，如图 3-53、3-54、3-55 所示。

```
mesh(X, Y, Z, X+Y)
```

```
mesh(X, Y, Z, diff(z))
```

```
mesh(X, Y, Z, cos(z))
```

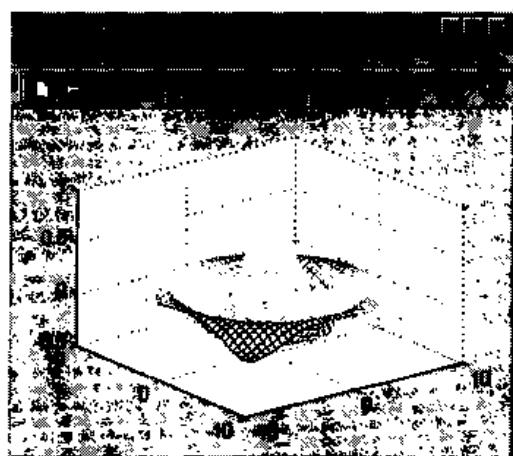


图 3-53  $\text{mesh}(X, Y, Z, X+Y)$

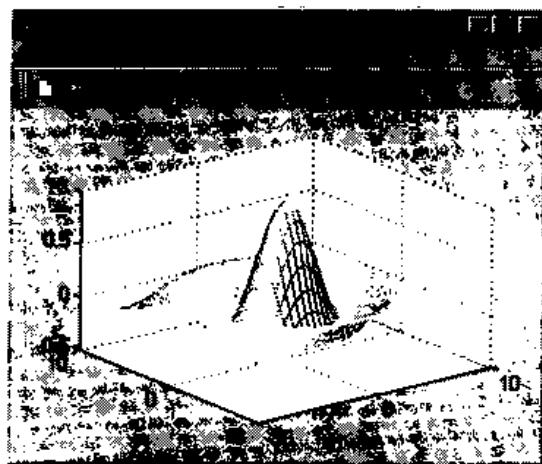


图 3-54  $\text{mesh}(X, Y, Z, \text{diff}(Z))$

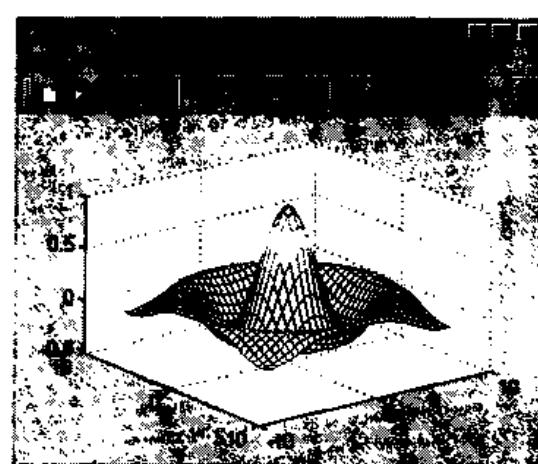


图 3-55  $\text{mesh}(X, Y, Z, \cos(Z))$

## 2. 调整视角

所谓视角简单讲就是观察（显示）图形的方向。调整视角可以使一幅图显示出来自不同方向的观察效果。

### (1) 用函数 view 调整视角

设置视角的基本函数是 view，它有如下几种不同的调用方式：

- view(*az, el*)或 view([*az, el*]): 设置观察图形的视角。
- view(2): 设置缺省的二维视角。
- view(3): 设置缺省的三维视角。
- [*az, el*] = view: 返回当前的视角。

下面我们就来介绍如何用第一种调用方式来设置图形的视角。式中的“az”是方位角 (azimuth)，指在 x-y 平面内从 y 轴负方向绕 z 轴旋转的角度，逆时针方向为正；“el”是仰角 (elevation)，指从 x-y 平面沿 z 轴方向仰起的角度，向 z 轴正方向的仰角为正。“az”和“el”的单位都是“度”。

二维图形视角的缺省值是：az = 0°，el = 90°。

三维图形视角的缺省值是：az = -37.5°，el = 30°。

例如，我们首先用如下命令建立表面图：

```
[X,Y,Z]=peaks(30);
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('Y 轴','FontWeight','bold')
zlabel('Z 轴','FontWeight','bold')
```

如图 3-56 所示，该图的视角为缺省值：az = -37.5，el = 30。下面我们来调整视角，使用命令：

```
view(90, 0)
```

该命令把方位角在 x-y 平面内从 y 轴负方向逆时针旋转 90 度，转到了 x 轴的正方向，而仰角为 0°，也就是说视线从 x 轴的正方向水平看过去的效果，如图 3-57 所示。

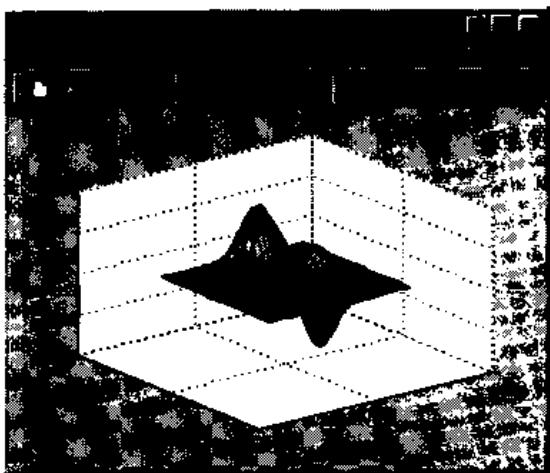


图 3-56 缺省视角

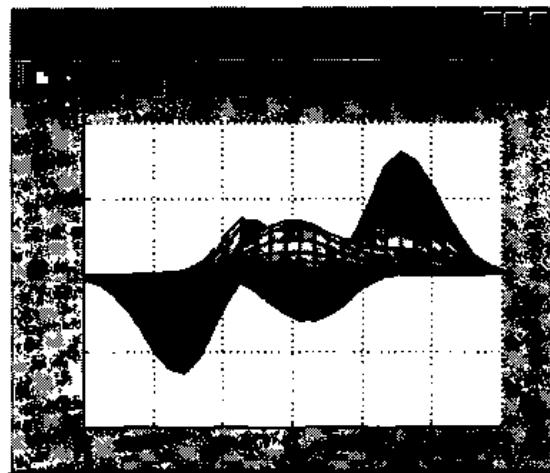


图 3-57 方位角 90 度，仰角为 0

如下命令：

```
view(-37.5, 80)
```

把方位角恢复为缺省值，仰角设为 80°，也就是从斜上方往下看，这时的效果如图 3-58 所示。

如下命令：

**view(0, 90)**

设置观察方向为从正上方向下看，如图 3-59 所示。

用 view 函数靠调整方位角和仰角来调整观察点是有一定限制的，它不能指定观察点的距离，只能指定方向，而且 z 轴总是指向上的。

要实现对观察点的高级控制，还需要借助于下面讲到的 MATLAB 照相制图技术（camera graphics）。

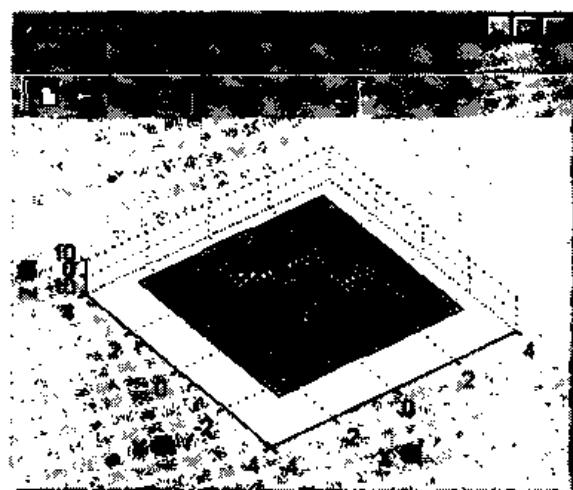


图 3-58 方位角-37.5°，仰角 80°

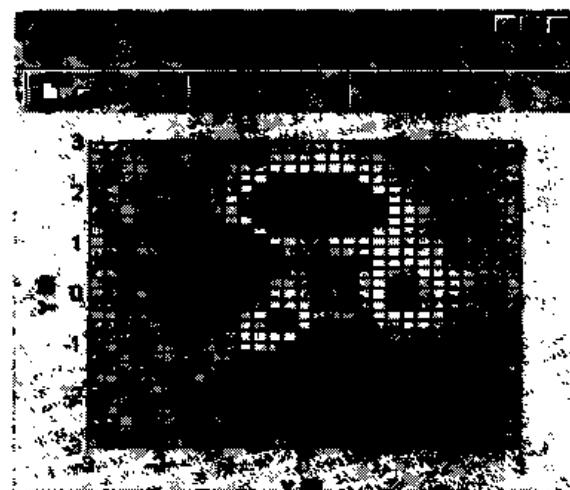


图 3-59 方位角为 0，仰角 90°

## (2) 照相制图技术

为了实现对观察点的控制，MATLAB 提供了类似照相机变焦透镜的高级技术，这种技术由表 3-6 中的函数实现。

表 3-6 照相制图的相关函数

函 数	功 能
<b>camdolly</b>	移动照相机的位置和被照目标
<b>camlookat</b>	移动照相机和目标以观察指定对象
<b>camorbit</b>	在被照目标周围旋转照相机
<b>campan</b>	在照相机的位置周围旋转被照目标
<b>campos</b>	设置或获取照相机的位置
<b>camproj</b>	设置或获取投影类型（正交或远景）
<b>camroll</b>	绕着观察坐标轴旋转照相机
<b>camtarget</b>	设置或获取被照目标的位置
<b>camup</b>	设置或获取照相机向上向量的值
<b>camva</b>	设置或获取照相机视角的值
<b>camzoom</b>	推近（放大）或拉远（缩小）照相机

例如，首先建立一个缺省情况下的三维高斯分布曲线：

```
[X,Y,Z]=peaks(30);
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('Y 轴','FontWeight','bold')
zlabel('Z 轴','FontWeight','bold')
```

参见图 3-56，下面我们首先获取当前照相机的位置和视角，命令和结果如下：

```
campos
ans =
-36.5257 -47.6012 86.6025
camva
ans =
10.3396
```

命令 `campos` 返回的值是照相机的位置坐标 [x, y, z]，`camva` 返回的是照相机观察目标的视线夹角大小。

下面我们首先把照相机的位置移到和当前位置关于坐标原点对称的地方，命令如下：

```
campos([36.5257, 47.6012, -86.6025])
```

结果如图 3-60 所示。

在上一步基础上，我们把观察的角度减小，从而相当于放大了目标，命令如下：

```
camva(5)
```

结果如图 3-61 所示。

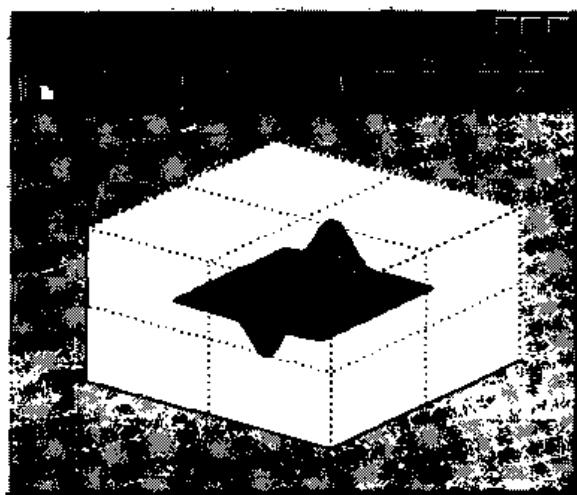


图 3-60 移动照相机位置

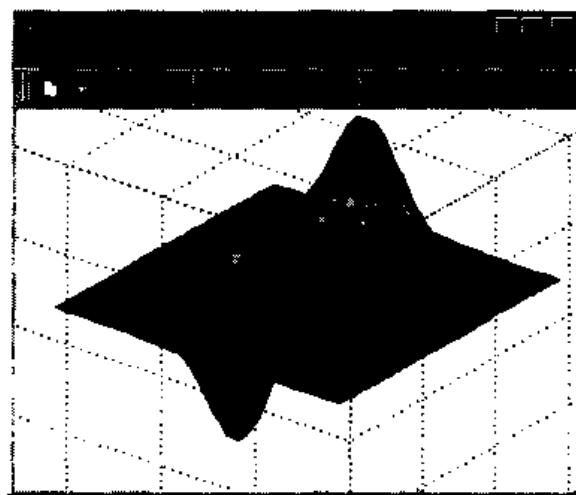


图 3-61 减小视角

下面我们再把放大了的目标绕观察轴旋转 180 度，命令如下：

```
camroll(180)
```

结果如图 3-62 所示。

从这个例子我们可以看出，照相制图技术功能要强大很多，如果能够灵活运用，会得到逼真的效果。

### (3) 用 axis 函数设置坐标轴的缩放比例

在调整坐标轴的属性时，函数 `axis` 非常有用，它有很多种用法分别实现不同的功能。

函数 `axis` 最基本的功能是设置坐标轴的缩放比例。

通常 MATLAB 按绘图数据的最大值和最小值来确定合适的坐标刻度范围，用户也可以用 `axis` 命令直接指定坐标刻度范围，格式如下：

- 1) `axis([xmin xmax ymin ymax])`: 二维图形。
- 2) `axis([xmin xmax ymin ymax zmin zmax])`: 三维图形。
- 3) `v = axis`: 返回当前坐标系的坐标刻度范围。

如下几种调用格式用于指定选取坐标刻度的方式：

- 1) `axis auto`: 设为自动选取刻度的方式，缺省值。
- 2) `axis manual`: 使坐标刻度保持当前状态不变。
- 3) `axis tight`: 限定坐标刻度等于数据范围。
- 4) `axis fill`: 使坐标轴充满边框。这种方式只有当属性“PlotBoxAspectRatioMode”或“DataAspectRatioMode”设为“manual”时有效。

例如，对于前面图 3-56 中的图形，如果在执行完绘图命令后使用命令：

**`axis tight`**

那么，该图形的坐标范围将位于曲面的边缘，曲面就显得大了，如图 3-63 所示。

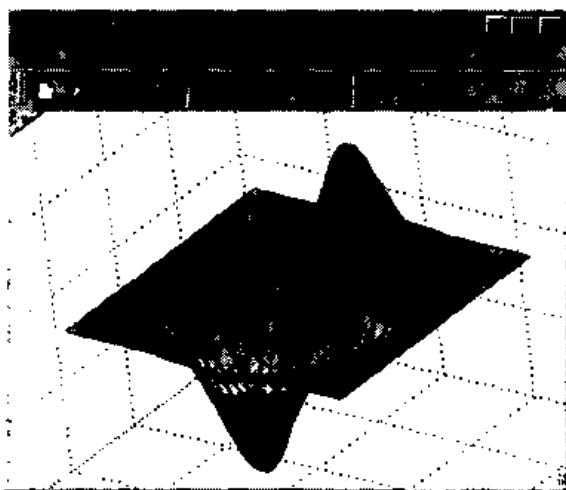


图 3-62 沿观察轴旋转 180 度

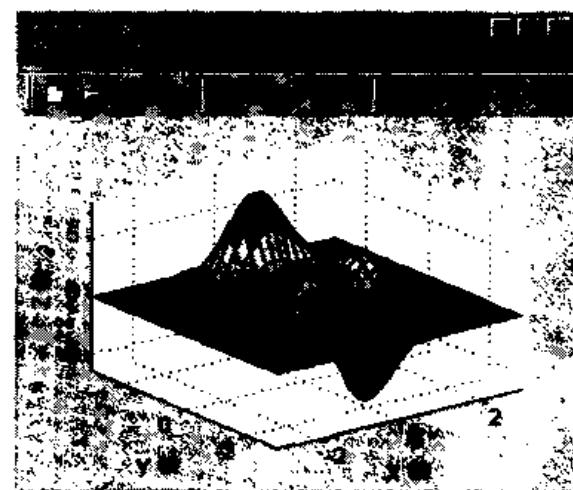


图 3-63 调整了坐标范围的表面图

函数 `axis` 还可以用于调整坐标模式：

- `axis ij`: 设为“矩阵”坐标模式，这时坐标系的起点在左上角。
- `axis xy`: 缺省值，“笛卡儿”坐标模式，这时坐标系的起点在左下角。

图 3-64 就是图 3-63 的“矩阵”坐标模式。

`axis` 函数还能用于设置坐标轴的纵横比。

通常情况下，坐标轴的缩放比例是根据图形窗口的大小选取的，图形窗口的尺寸改变时，为了保持坐标轴在图形窗口中的比例，坐标轴的尺寸会随之改变，纵横比也可能发生变化。而有的图形需要按照确定的纵横比例显示，在图形窗口纵横比发生变化时它的比例也保持不变，甚至有的图形的尺寸都不能发生变化。这些特殊的要求可以借助 `axis` 函数来

实现，相关调用如下：

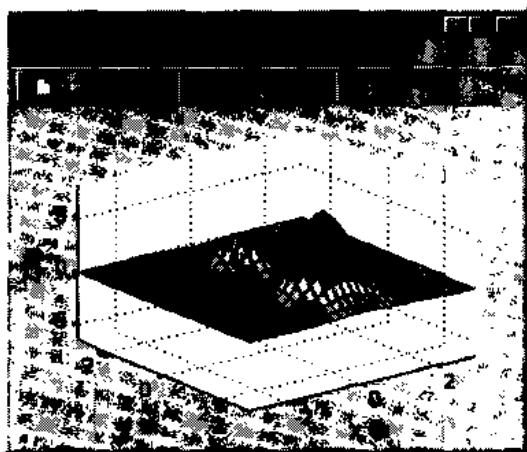


图 3-64 “矩阵”坐标模式

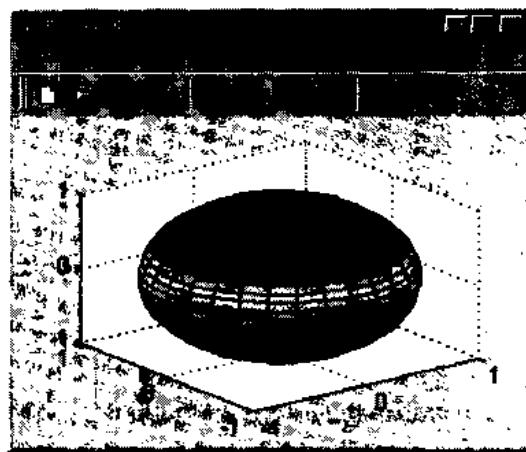


图 3-65 缺省情况下的球体形状

- `axis equal`: 使得在几个方向的坐标轴上对于相等的刻度增量尺寸也相同。
- `axis image`: 在 `axis equal` 的基础上, 还使得坐标轴的边框紧靠数据曲线。
- `axis square`: 使坐标轴的边框尺寸相等。
- `axis normal`: 恢复缺省状态。
- `axis vis3d`: 锁定纵横比。

例如, MATLAB 下的 `sphere` 函数会生成一个球体, 缺省情况下, 运行命令:

`sphere(30)`

绘出的球体如图 3-65 所示, 我们看到这时的球体是扁的, 在调用函数 `axis equal` 之后, 球体才会变为真正的圆球, 如图 3-66 所示, 这时如果再改变图形窗口的大小, 球体也会随之改变大小, 但其纵横比不会变。

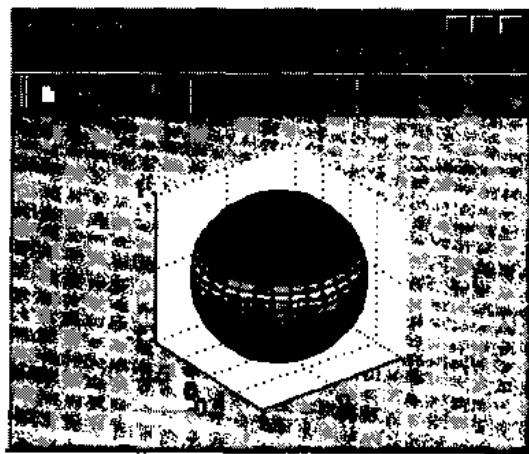


图 3-66 调用 `axis equal` 后球体变圆了

### 3. 光源的设置

为图形加上光源能够得到更加逼真的效果。MATLAB 提供了放置光源和调整光照目标特性的一些函数, 如表 3-7 所示。

表 3-7 光源的设置函数

函 数	功 能
camlight	根据照相机的位置建立或移动光源
lightangle	在球形坐标系中建立或放置光源
light	建立光源对象
lighting	选择光照方法
material	设置被照目标的反射特性

合理地运用这些函数会得到很真实的视觉效果。

### (1) 建立光源

在设置光照效果之前要首先建立光源，可以用 light、camlight 或 lightangle 函数来完成。

light 函数一般有三种调用方式：

- light: 使用缺省值建立光源。
- light(param1, value1, ..., paramn, valuen): 在建立光源的同时设置参数。
- H=light(...): 返回建立光源对象的句柄。

camlight 函数用于建立类似照相时的光源，其调用方式有：

- camlight headlight: 在当前坐标系中照相机的位置建立光源。
- camlight right: 在照相机的右上方建立光源。
- camlight left 在照相机的左上方建立光源。
- camlight: 缺省情况，在照相机的右上方建立光源。
- camlight(az, el): 在相对照相机方位角为“az”，仰角为“el”的位置建立光源。
- camlight(..., style): 设置光源的类型，可以是“local”（缺省值）或“infinite”。
- camlight(h, ...): 把指定的光源放在指定的位置。
- h = camlight(...): 返回光源对象的句柄。

lightangle 函数是在球形坐标系中建立光源，用法如下：

- lightangle(az, el): 在指定位置建立光源。
- h = lightangle(az, el): 建立光源并返回其句柄。
- lightangle(h, az, el): 设置指定光源的位置。
- [az el] = lightangle(h): 获取指定光源的位置。

例如，下面的例子建立一个三维高斯分布表面图，然后用 light 函数在指定位置添加光源，效果见图 3-67。

```
[X,Y,Z]=peaks(30);
surf(X,Y,Z)
axis tight
light('Position',[-4 -6 10])
```

而如下一组命令首先改变图形的视角，然后用 lightangle 函数指定光源位置，效果见图 3-68。

```
[X, Y, Z]=peaks(30);
```

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

```
view(90, 0)
```

```
lightangle(45, 30)
```

## (2) 设置照明方式

在为图形添加了光源后，还要考虑选取合适的照明方式，不同的照明方式效果也不一样。

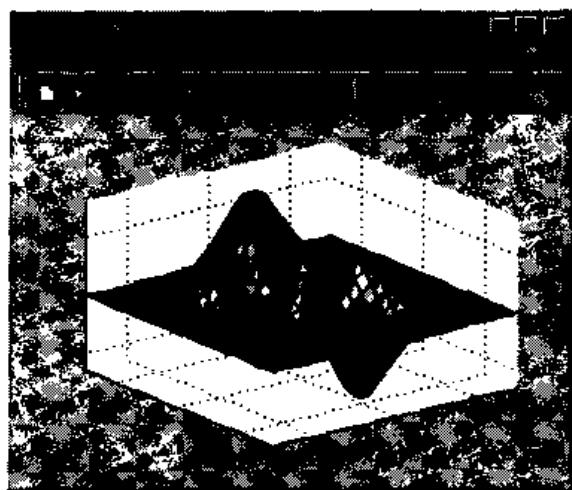


图 3-67 用 light 函数加光源

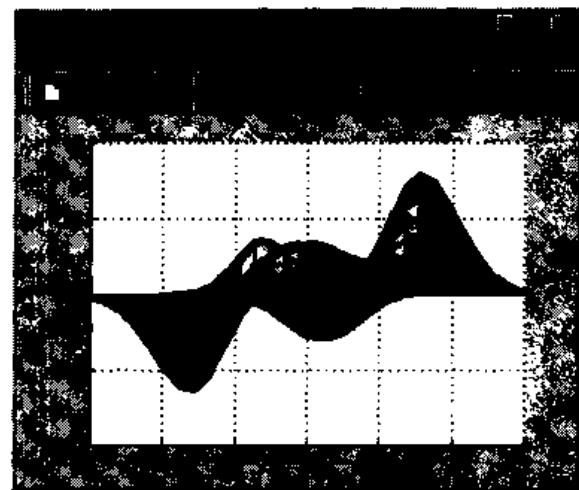


图 3-68 用 lightangle 函数加光源

设置照明方式由 lighting 函数完成，MATLAB 提供了三种照明方式，分别由 lighting 函数的不同调用方式来设置：

- `lighting flat`: “flat” 方式——均衡图形中每个小表面的交叉颜色，这是缺省方式。
- `lighting gouraud`: “gouraud” 方式——对小表面的交叉颜色做颜色插值。
- `lighting phong`: “phong” 方式——对小表面的交叉颜色做颜色插值，并计算每个像素的反射比。
- `lighting none`: 关闭照明。

我们来比较一下几种方式的区别，首先绘制一个球体，去掉网格，然后用 `camlight` 命令在照相机的左上方和右上方各设置一个光源，最后用 `lighting` 命令指定照明的不同方式。为便于比较，我们把四种情况用 `subplot` 函数合到一幅图中，如图 3-69 所示。程序如下：

```
subplot(2,2,1)
sphere
shading flat
camlight left
camlight right
lighting flat
```

```
subplot(2,2,2)
sphere
shading flat
camlight left
```

```
camlight right  
lighting gouraud  
  
subplot(2,2,3)  
sphere  
shading flat  
camlight right  
camlight left  
lighting phong  
  
subplot(2,2,4)  
sphere  
shading flat  
camlight left  
camlight right  
lighting none  
  
subplot(2,2,1)  
title('Flat Mode')  
subplot(2,2,2)  
title('Gouraud Mode')  
subplot(2,2,3)  
title('Phong Mode')  
subplot(2,2,4)  
title('None Mode')
```

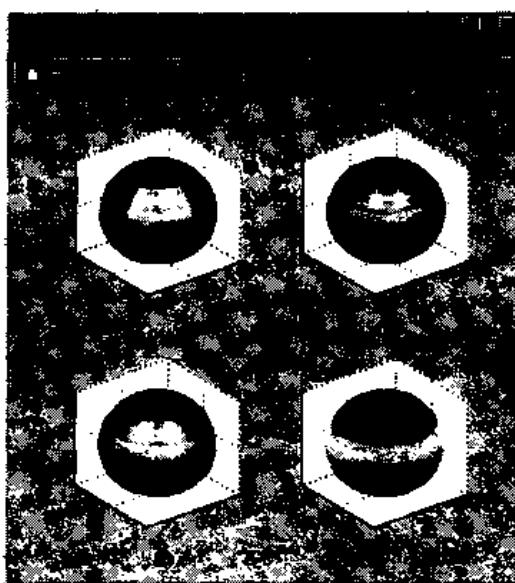


图 3-69 三种照明方式和无照明的比较

### 3.3.6 其他三维图形

#### 1. 离散数据图

在工程中经常会遇到离散数据的绘图问题, MATLAB 专门为这提供了几个函数: stem、stem3 和 stairs。

函数 stem 和 stem3 分别用来绘制二维和三维的离散图形, 它们的用法与前面讲过的 plot 和 plot3 基本一致。区别是函数 stem 和 stem3 中可以用选项 “filled” 来填充图中的离散点的标记。

函数 stairs 用于绘制类似楼梯形状的步进图形。

下面几个例子分别使用了这三个函数。

如下一组命令绘制一组正弦数据的二维离散图形:

```
x=0:0.1:2*pi;
stem(x,sin(x))
```

结果如图 3-70 所示。

下面的命令用以建立三维离散图形, 并填充离散点的标记符号:

```
x=0:0.1:10;
stem3(exp(x),x,exp(x),'filled')
```

结果如图 3-71 所示。

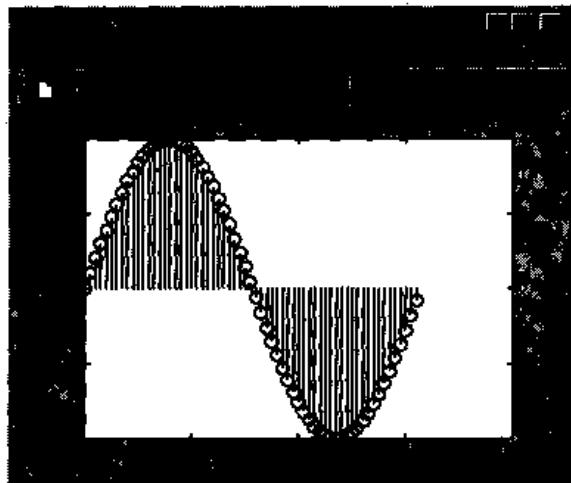


图 3-70 正弦离散图形

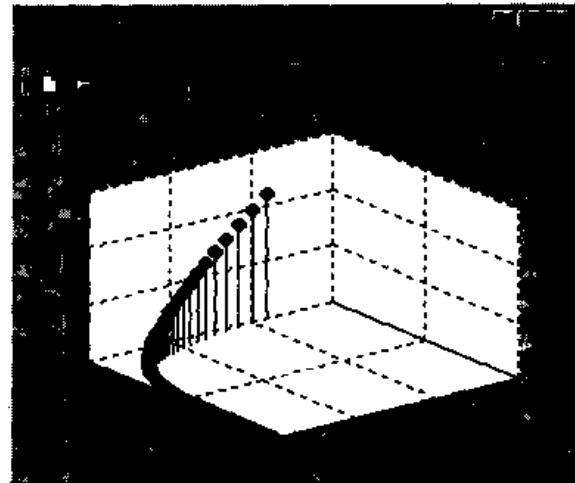


图 3-71 三维离散图形

如下一组命令用于绘制步进图形, 同时我们用相同的数据绘制了一条普通曲线以作参照:

```
x=0:0.3:2*pi;
stairs(x,sin(x))
hold on
plot(x,sin(x),'r:')
hold off
```

结果如图 3-72 所示。

#### 2. 漩布图

绘制瀑布图的函数为 waterfall，它和函数 mesh 基本相同，只是它不画出纵向的线条，因而产生类似“瀑布”的效果。

例如，对于三维高斯分布，我们来画它的瀑布图：

```
waterfall(peaks)
```

```
axis tight
```

结果如图 3-73 所示。

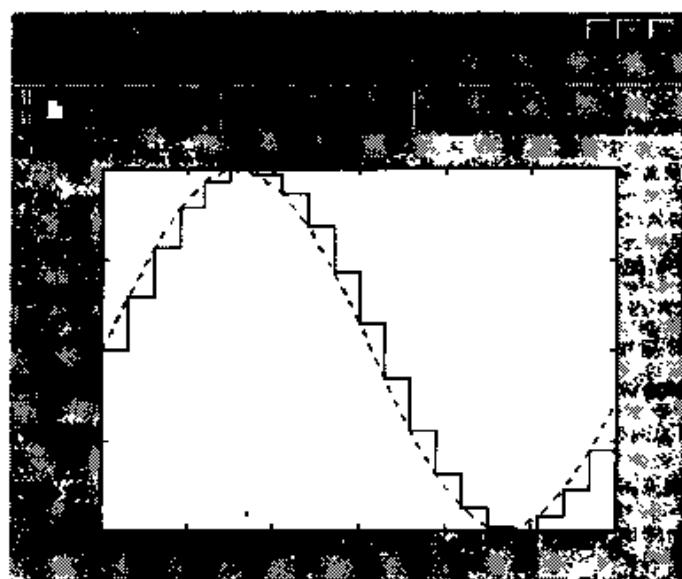


图 3-72 步进图形

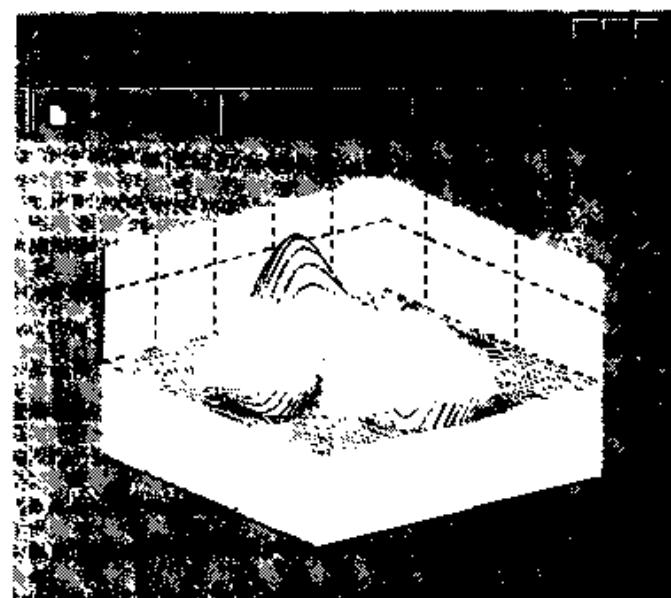


图 3-73 瀑布图

# 第 4 章 工作环境管理

## 4.1 搜索路径管理

### 4.1.1 用户目录的建立

在第 1 章的 1.3 节我们曾经介绍过 MATLAB 自身的目录结构，这是软件安装时建立的。用户在使用 MATLAB 时一般要建立自己的目录，而且希望目录中的文件能够在 MATLAB 环境中直接调用，这就需要借助相应的 MATLAB 路径管理工具把建立的目录添加到 MATLAB 搜索路径中。

MATLAB 5.3 为用户提供了具有交互式图形用户界面的路径管理工具——路径浏览器（Path Browser），在 MATLAB 5.3 命令窗口的“File”菜单下选择“Set Path”就可以打开路径浏览器，如图 4-1 所示。也可以单击命令窗口工具栏上的图标。

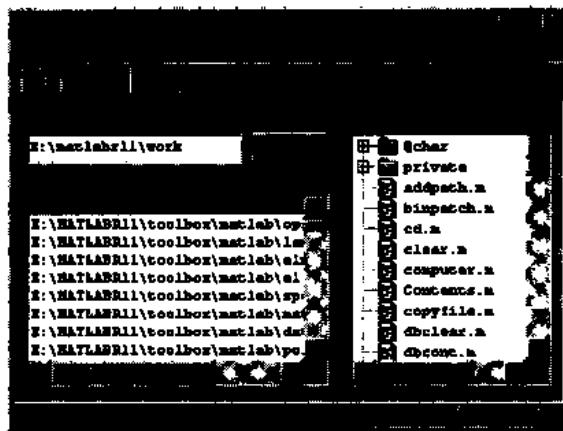


图 4-1 路径浏览器

MATLAB 5.3 路径浏览器的界面比较友好，用户可以方便的在已有的 MATLAB 搜索路径基础上添加、删除路径，还可以查看、编辑或运行路径中的文件。

浏览器窗口的左侧有两个窗口，上面的小窗口“Current”显示当前目录，按下后面的“Browse”按钮可以改变当前目录，下面的窗口中列出了 MATLAB 搜索范围内的所有路径；浏览器右侧的窗口显示的是在搜索路径列表中选中的目录中的文件。

#### 1. 添加新路径

添加新路径有以下两步：

- (1) 在路径浏览器的“Path”菜单下选择“Add to Path”，或在工具栏上单击图标，都会弹出如图 4-2 所示的对话框，让用户指定要添加的目录名，并可以在对话框下方选择添加到路径列表的前面（Add to front）还是后面（Add to back）。然后按下“OK”。



图 4-2 指定要添加的路径

(2) 在浏览器的“File”菜单下选择“Save Path”，这样添加的路径才会保存下来。新路径添加完成后，该目录下的 M 文件在 MATLAB 环境中就可以直接调用了。

#### 2. 删 除 路 径

删除路径的步骤和添加路径类似，首先在路径列表中选择要删除的路径，然后选择“Path”菜单下的“Remove from Path”或工具栏上的~~剪切~~按钮，最后保存即可。

#### 3. 查 看 和 编 辑 文 件

如果用户需要在路径浏览器中查看或编辑 M 文件，可以首先在路径列表中选择该 M 文件所在的目录，然后在浏览器右侧的窗口中选择该 M 文件，最后在“File”菜单下选择“Open”或按下工具栏上的~~剪切~~按钮，也可以用鼠标双击该文件，这样 MATLAB 就会用编辑器打开该文件，可以查看或编辑。

#### 4. 运 行 文 件

用上述方法选中了某个 M 文件后，选择 Tools 菜单下的选项 Run，将会直接运行该文件。

除了这几种主要功能外，在路径浏览器中还可以实现一些常规设置，读者如果感兴趣，不妨自己实验。

用户也可以在命令窗口使用命令来完成添加和删除路径的工作，相关命令是：path、addpath 和 rmpath。path 用于获取或设置搜索路径；addpath 用于添加路径；rmpath 用于删除路径。它们的用法都很简单，请读者参看在线帮助。

### 4.1.2 搜索文件的顺序

当 MATLAB 遇到一个输入的名称时，会按照如下的顺序搜索：

- (1) 检查它是否工作空间中的变量
- (2) 检查是否嵌入函数
- (3) 检查是否子函数
- (4) 检查是否私有函数
- (5) 是否位于 MATLAB 搜索路径范围内的函数文件或脚本文件

如果在用户的搜索路径中，有不止一个函数具有同一个函数名，MATLAB 将只执行搜索中遇到的第一个函数，而其他的同名函数将被屏蔽且不能执行。

用第一章介绍的“which 名称 -all”命令可以列出所有的同名函数和变量。

## 4.2 工作空间管理

### 4.2.1 工作空间浏览器

工作空间浏览器是 MATLAB 用以管理工作空间中变量的工具。选择 File 菜单下的“Show

“Workspace”选项或工具栏上的按钮■，就可以打开工作空间浏览器窗口，如图 4-3 所示。

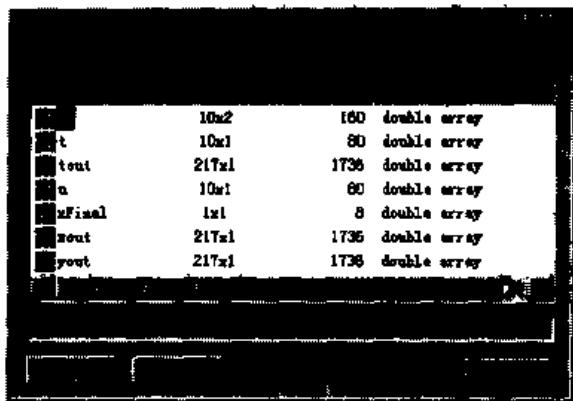


图 4-3 工作空间浏览器

在浏览器窗口中列出 MATLAB 工作空间中所有变量的主要信息，选中该变量，单击窗口下方的“Open”按钮，则用 MATLAB 编辑器打开该变量，如图 4-4 所示，在编辑器中可以对变量内容进行编辑；单击“Delete”按钮则从工作空间中删除该变量。

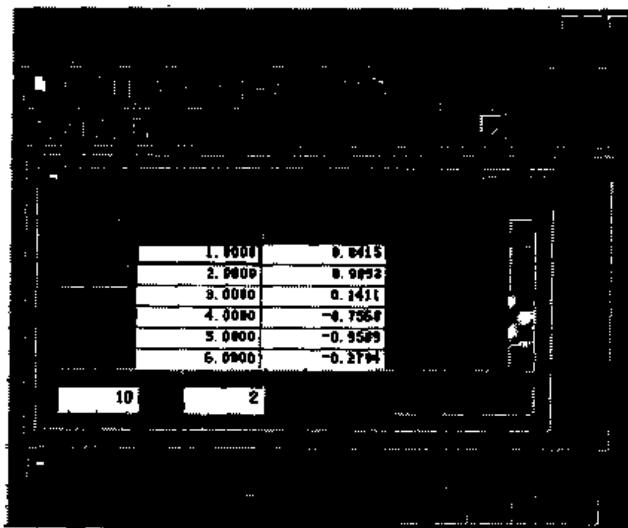


图 4-4 编辑工作空间中的变量

#### 4.2.2 保存和载入 MATLAB 工作空间的内容

用户可以随时对 MATLAB 工作空间的变量进行保存，并且随时把保存的数据重新载入到 MATLAB 的工作空间内。

MATLAB 命令窗口 File 菜单的“Save Workspace As”选项用来把 MATLAB 工作空间中的所有变量保存到 MATLAB 数据文件中，该数据文件为二进制的 MAT 文件。

“Load Workspace”选项用来把包含 MATLAB 变量的数据文件载入到工作空间，载入后的工作空间中就会含有该数据文件中的变量。

#### 4.2.3 保存和载入变量

前面介绍了用 MATLAB 菜单实现工作空间中的变量保存和载入的方法，这种方法只限于工作空间的变量，而且只能是全部变量，因而有很大的局限性。

本节介绍用 `save` 命令和 `load` 命令实现变量的保存和载入，这是一种通用而灵活的方法。`save` 命令将工作空间中的指定变量以指定格式存储成 MAT 文件。`save` 命令有下列几种调用方式：

- `save`: 把工作空间的全部变量以二进制形式存储在隐含的文件 `matlab.mat` 中；
- `save filename`: 将工作空间的全部变量以二进制形式存储在指定文件 `filename.mat` 中；
- `save filename variables`: 将工作空间的指定变量 `variables`（可以是一个或多个）以二进制形式存储在文件 `filename.mat` 中；
- `save filename options`: 用表 4-1 中的格式存储工作空间中的全部变量。
- `save filename variables options`: 用表 4-1 中的格式存储工作空间中的指定变量 `variables`。
- `save ('filename', 'variables')`: `save` 命令的函数形式，当文件名或变量名是变量时，可以把 `save` 命令以这种形式调用，但这种形式不能用附加选项指定存储格式。

表 4-1 数据存储格式

选项	存储格式
<code>-ascii</code>	8 位 ASCII 码形式
<code>-ascii -double</code>	16 位 ASCII 码形式
<code>-ascii -tabs</code>	8 位 ASCII 码形式，用制表符 tab 分隔
<code>-ascii -double -tabs</code>	16 位 ASCII 码形式，用制表符 tab 分隔
<code>-append</code>	在已有的 MAT 文件中添加指定变量

`load` 命令将 MAT 文件中的变量载入工作空间。它的用法和 `save` 命令类似，请读者参看在线帮助。以下几点在使用时需要注意：

- (1) 以二进制形式存储的文件，直接使用 `load` 命令调用即可。
- (2) 以 ASCII 形式存储的多个变量将会合并成一个变量，变量名就是存储成的文件名。在用 `load` 命令调用时，必需使用 `-ascii` 选项，并且调入工作空间时，只有一个变量，只能使用冒号算子访问每一个变量。因此，采用 ASCII 形式存储时，最好将每一个变量分别存储。
- (3) 采用 ASCII 形式存储复数，将造成虚部丢失，因为 MATLAB 不能调入非数值数据“i”或“j”。

## 4.3 命令窗口管理

### 4.3.1 环境参数设置

MATLAB 命令窗口 File 菜单下的 Preferences 选项用于对 MATLAB 工作环境中的参数进行设置。单击该选项会弹出如图 4-5 所示的参数设置窗口。

#### 1. 常规设置

在图 4-5 所示的“General”夹中，用户可以设置 MATLAB 的数据格式、编辑器以及帮助文件所在目录等。

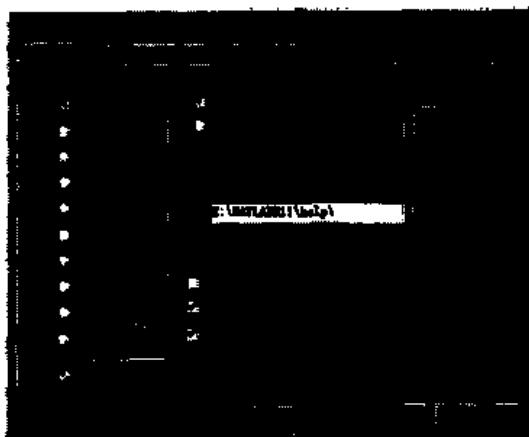


图 4-5 参数设置窗口——“General”夹

尽管在 MATLAB 内部，总是用双精度数执行所有的运算，但用户可以设置显示在屏幕上的 MATLAB 语句的输出格式。

在“Numeric Format”栏内，用户可以通过 MATLAB 提供的十一个单选按钮来设置在 MATLAB 命令窗口显示的数据格式。

用户可以通过“General”夹的“Editor Preference”栏来指定优先编辑器。用户既可以使用 MATLAB 本身的编辑器，也可以指定编辑器。

此外，用户可以在“Help Directory”栏中指定 MATLAB 帮助文件的目录。

选择“Echo On”复选框，将使 M 文件在运行时把其中的命令显示在命令窗口中；选择“Show Toolbar”复选框显示工具栏，选中“Enable Graphical Debugging”复选框用来启用图形化调试功能。

## 2. 字体设置

“Command Window Font”夹用于设置命令窗口的字体。它是 Windows 通用的字体设置对话框，这里不再赘述。

## 3. 图形拷贝设置

“Copying Options”夹内容如图 4-6 所示，用于设置图形拷贝的选项。

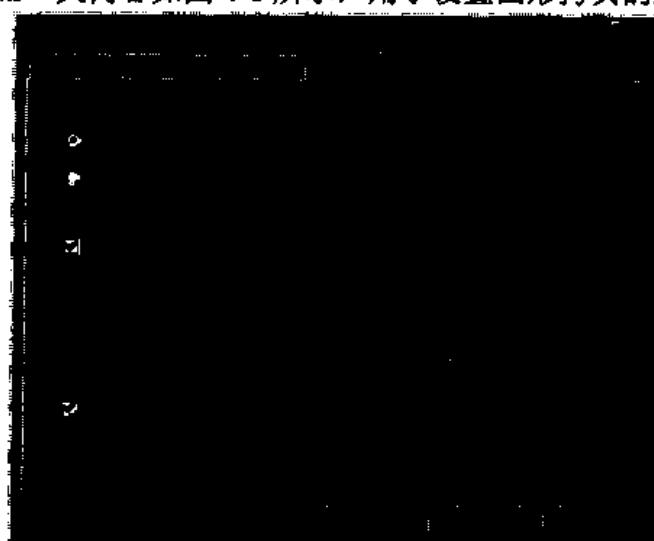


图 4-6 “Copying Options”夹

“Clipboard Format”栏内的单选按钮“Windows Metafile”和“Windows Bitmap”可以将剪贴板格式设置为 Windows 图元文件或 Windows 位图文件。

选择“Match Figure Screen Size”复选框时，将按照图形在屏幕上显示的尺寸来复制图形。

如果选择“White background”复选框，则复制图形时使用白色背景。

### 4.3.2 执行外部应用程序

在 MATLAB 命令窗口中可以直接运行外部应用程序，方法是在程序名前加感叹号“!”。例如，在命令窗口输入：

**!notepad**

就会启动 Windows 记事本，并且当前的程序进程切换到记事本，MATLAB 命令窗口处于停止状态，不能输入任何内容，直到把记事本关闭为止。

如果在命令的结尾添加符号“&”，则外部程序将在另外的窗口执行，这时 MATLAB 命令窗口仍然保持活动状态。例如：

**!notepad &**

这条命令在打开记事本的同时，MATLAB 命令窗口仍然可以输入命令。类似的，如下命令：

**!dir &**

另外打开一个 DOS 窗口运行 DOS 命令 dir，执行完毕后 DOS 窗口并不关闭，而且 MATLAB 命令窗口处于活动状态。

### 4.3.3 命令窗口的分页输出

在命令窗口查看较大的矩阵时，矩阵前面的内容一般无法看到，这时需要控制命令窗口显示的内容分页输出，MATLAB 为此提供了命令 more，该命令的用法为：

- more off：不允许分页输出
- more on：允许分页输出
- more(n)：指定每页输出的行数

当允许分页输出时，每显示一页的内容，在命令窗口底部会显示“--more--”标记，这时按回车键前进一步，按空格键翻到下一页，按“q”键结束当前内容的显示。

## **中级篇**

# **第 5 章 基本编程**

MATLAB 不但是一个功能强大的工具软件，更是一种高效的编程语言。MATLAB 软件就是 MATLAB 语言的编程环境，M 文件也就是用 MATLAB 语言编写的程序代码文件。

本章将从语言的角度介绍编写基本 MATLAB 程序的规则和方法。

## **5.1 变量、语句**

变量和语句在前面几章中实际上一直在用，这一节我们从语言的角度把相关内容进行归纳和总结。

### **5.1.1 变量类型**

#### **1. 变量命名规则**

为变量（包括函数）命名时应该遵循以下的规则：

- (1) 必须以字母开头
- (2) 可以由字母、数字和下划线混合组成
- (3) 字符长度应不大于 31 个

注意：MATLAB 区分大小写

#### **2. 局部变量和全局变量**

通常，每个函数体内都有自己定义的变量，不能从其它函数和 MATLAB 工作空间访问这些变量，这些变量就是局部变量。如果要使某个变量在几个函数中和 MATLAB 函数空间都能使用，可以把它定义为全局变量。

全局变量就是用关键字“global”声明的变量。全局变量名尽量大写，并能够反应它本身的含义。

如果需要在几个函数中和 MATLAB 工作空间都能访问一个全局变量，那么必须在每个函数中和 MATLAB 工作空间内都声明该变量为全局的。

全局变量需要在函数体的变量赋值语句之前说明，整个函数以及所有对函数的递归调用都可以利用全局变量。

注意：实际编程中，应尽量避免使用全局变量，因为全局变量的值一旦在一个地方被改变，那么在其它包括该变量的函数中都将改变，这样有可能会出现不可预见的情况。如果需要用全局变量，建议全局变量名要长，能反映它本身的含义，并且最好所有字母都大写，并有选择地以首次出现的 M 文件的名字开头。

### **5.1.2 基本语句**

MATLAB 可以认为是一种解释性语言，用户可以在 MATLAB 命令窗口键入命令，也

可以在编辑器内编写应用程序，这样 MATLAB 软件对此命令或程序中各条语句进行翻译，然后在 MATLAB 环境下对它进行处理，最后返回运算结果。

MATLAB 语言的基本语句结构为：

变量名列表 = 表达式

其中等号左边的变量名列表为 MATLAB 语句的返回值，等号右边的是表达式的定义，它可以是 MATLAB 允许的矩阵运算，也可以是函数调用。

等号右边的表达式可以由分号结束，也可以由逗号或回车结束，但它们的含义是不同的，如果用分号结束，则左边的变量结果将不在屏幕上显示出来，否则将把结果全部显示出来。

MATLAB 语言和 C 语言不同，在调用函数时 MATLAB 允许一次返回多个结果，这时等号左边是用 [ ] 括起来的变量列表，例如第三章用过的函数：

**[X, Y, Z] = peaks;**

注意：表达式中的运算符号两侧允许有空格，以增加可读性。但在复数或符号表达式中要尽量避免，以防出错。

在 MATLAB 的基本语句结构中，等号左边的变量名列表和等号一起可以省略，这时将把表达式的执行结果自动赋给变量“ans”，并显示到命令窗口中。

## 5.2 数据类型

MATLAB 下的数据类型最大的特点是每一种类型都以数组为基础，都是从数组派生出来的，MATLAB 事实上把每种类型的数据都作为数组处理。

MATLAB 下有六种基本数据类型，分别是：double（双精度数值），char（字符），sparse（稀疏数据），storage（存储型），cell（单元数组），struct（结构）。各种数据类型的层次关系如图 5-1 所示。

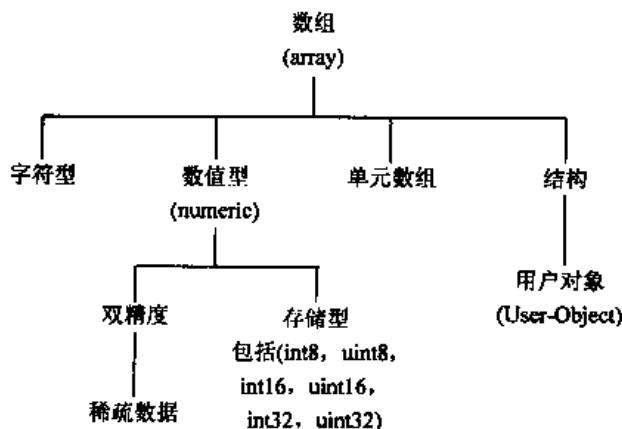


图 5-1 MATLAB 基本数据类型结构图

其中存储型是一个虚拟数据类型，这是 MATLAB5.3 版中新增的定义，它包括：int8（8 位整型），uint8（无符号 8 位整型），int16（16 位整型），uint16（无符号 16 位整型），int32（32 位整型），uint32（无符号 32 位整型）。

这些类型中最常用的一般只有双精度型和字符型，所有的 MATLAB 计算都把数据作为双精度型处理。

其它的数据类型只在一些特殊的条件下使用，如：无符号 8 位整型数据（uint8）一般用于存储图象数据，稀疏数据（sparse）一般用于处理稀疏矩阵，单元数组（cell）和结构（struct）一般只在大型程序中使用。

存储型数据（storage）只用于内存的有效存储。可以对这些类型的数组进行基本操作，但不能对它们执行任何数学运算，在执行数学运算之前必须用 double 函数把这类数组转换为双精度型。

在 MATLAB 结构类型（struct）的基础上，还可以定义用户的类和对象，关于类和对象的知识请读者参考相关书籍。

本节侧重为读者介绍 MATLAB 中的字符、结构和单元数组三种数据类型。

### 5.2.1 字符

对于编程语言来讲，字符处理是必不可少的。MATLAB 有强大的字符处理能力，其字符处理函数如表 5-1 所示。

表 5-1 字符处理函数

函数	功能	函数	功能
abs	将字符串转变为 ASCII 码值	texlabel	用特征字符串产生 TeX 格式的符号
eval	解释执行字符串	char	建立或转换为字符数组
deblank	删除字符串末尾的空格	int2str	整数转换为字符串
findstr	从一个字符串中查找另一个字符串	mat2str	矩阵转换为字符串
lower, upper	将字符串转变为小写，大写	num2str	数字转换为字符串
strcat	字符串水平连接（串联）	sprintf	将带格式的数字转换为字符串
strcmp	比较字符串	sscanf	将字符串转换为带格式的数字
strcmpi	忽略大小写比较字符串	str2double	字符串转换为双精度数
strncmp	比较两个字符串的前 n 个字符	str2num	字符串转换为数字
strmatch	查找匹配的字符串	bin2dec	二进制数转换为十进制数
strrep	替换字符串	dec2bin	十进制数转换为二进制数
strjust	对齐字符串数组（包括左对齐、右对齐、居中）	dec2hex	十进制数转换为十六进制数
strtok	返回字符串中第一个分隔符（包括空格、回车和 Tab 键）前的部分	hex2dec	十六进制数转换为十进制数
strvcat	垂直连接字符串	hex2num	十六进制数转换为双精度数
symvar	确定字符串中的符号变量（除常量和函数外的内容）		

MATLAB 中的字符串有几条基本规则：

- (1) 所有字符串都用单引号括起来。
- (2) 字符串中的每一个字符（不是单词）都是该字符串变量（矩阵或向量）中的一个元素。

(3) 字符串中的字符以 ASCII 码形式存储，因而大小写是有区别的。用 abs 函数可以看到字符的 ASCII 码值。

下面我们介绍 MATLAB 中对字符串的几类常用操作方法。有些很直观的用法不再介绍。

### 1. 建立

建立字符串可以通过直接赋值，也可以使用函数 char、int2str、mat2str、num2str 和 sprintf 建立或转换得到。同样，字符串也可以用函数 abs、double、str2double、str2num 和 sscanf 转换为数值形式。

```
s = 'China'                                % 直接给s赋字符串，s中的每个字符都是一个元素
s =
China
char([97 98 99 100])                      % 把数字按照ASCII码值转换为字符串
ans =
abcd
num2str([97.01 98 99 100])                 % 把数字直接转换为字符串，每个数为一个独立字符串
ans =
97.01          98           99           100
int2str([97.01 98 99 100])                 % 把数字取整后再转换为字符串，注意和num2str的区别
ans =
97  98  99 100
mat2str([97 98; 99 100])                  % 把矩阵转换为字符串，得到的结果中，...
ans =                                         % 包括“[]”、“;”和空格在内的每个字符都是其中的元素
[97 98;99 100]
abs('abcd') 或 double('abcd')    % 字符串转换为ASCII码值
ans =
97    98    99    100
```

### 2. 连接

字符串可以连接到一起组成更大的字符串。水平连接通过函数 strcat 或在中括号内用逗号连接；垂直连接通过函数 strvcat 或在中括号内用分号连接。它们的作用不尽相同，请看下面的例子。

※ 注意：垂直连接方式得到的是字符串数组，水平连接方式得到的只是更长的字符串。

```
['Red ','Yell ow']        % 在中括号中直接水平连接，结果包括原字符串结尾处的空格
ans =
Red Yell ow
strcat('Red ','Yell ow')   % 函数strcat连接，忽略原字符串结尾处的空格
ans =
```

```

RedYellow
v1 = ['Red'; 'Yellow'] % 在中括号中用分号实现垂直连接，必须保证每个被连接
v1 =
% 的字符串长度相等，否则要把较短的字符串用空格补齐
Red
Yellow
v2 = strvcat('Red','Yellow') % 用该函数实现垂直连接，自动为较短的字符串补足空格
v2 =
Red
Yellow
size(v1(1,:)) % 查看垂直连接的结果v1的第一行的大小,
ans = % 发现“Red”后面有三个空格，它们也是v1中的元素
      1      6
size(v2(1,:)) % 函数strvcat虽然在连接前不用补足较短的字符串,
ans = % 但其结果中也会加入空格，使得各行长度相等
      1      6

```

### 3. 执行字符串

字符串的执行函数 eval 是很有用的，比如可以把用户输入的命令字符串直接执行。请看下面的例子。

```

a = 10; b = 20;
% 让用户交互输入a和b的运算关系，把结果赋给变量s
s = input('如何对a和b作运算: ','s');
如何对 a 和 b 作运算: (a-b)/a+1
eval(s) % 执行s中的内容
ans =
      0

```

### 4. 比较和查找

比较字符串的函数 strcmp、strcmpi 和 strncmp 按同样顺序比较字符串（数组）中的字符，相同则返回 1，不同则返回 0，它们的用法简单，不再赘述。

函数 strrep 用法为：

```
str = strrep(str1, str2, str3)
```

用字符串 str3 替换字符串 str1 中所有的字符串 str2。

函数 strmatch 的用法为：

```
i = strmatch('str', STRS)
```

查找字符串数组 STRS 中所有以字符串 'str' 开头的字符串，并返回其次序行向量，如果附加参数 'exact'，则查找完全相同的字符串。

函数 findstr 的用法为：

```
k = findstr(str1, str2)
```

返回所有在字符串 str1 中的字符串 str2 的起始位置。

```

str1 = 'To get started, type: helpwin, helpdesk, or demo.'
str2 = 'help'
str3 = 'turn'
k = findstr(str1, str2)
k =
    23    32
str = strrep(str1, str2, str3)
str =
To get started, type: turnwin, turndesk, or demo.
i = strmatch(str2, str1)      % str1 是一个字符串, 它的开头不是 str2, 因而返回空向
量
i =
[]
STR = strvcat('helpwin', 'helpdesk', 'demo');
i = strmatch(str2, STR)      % STR 是字符串数组, 在 STR 中查找以 str2 开头的字符串
i =
    1
    2

```

### 5.2.2 结构

MATLAB 中结构的概念和 C 语言中类似, 它也包含一个或多个域(数据容器), 每一个域可以包含任何类型的数据, 而且互相独立。

#### 1. 结构的建立和访问

我们通过下面的例子来说明如何建立并访问结构。

我们用如下命令建立一个结构, 用来记录一个城市一年的温度变化, 它包括三个域, 域中的数据类型各不相同:

```

temp.city = 'Beijing';          % 域city, 字符串, 记录城市名
temp.year = 1999;              % 域year, 标量, 记录年份
temp.data = [-10,-6,5; 10,20,25; 30,24,22; 19,10,6];  % 域data, 矩阵, 一年的温度数
据

```

**temp** % 得到的结构内容如下

```

temp =
  city: 'Beijing'
  year: 1999
  data: [4x3 double]

```

从这个例子中可以看出建立结构的一种方法: 直接建立结构和各个域, 同时给各域赋值(也可以不赋值), 结构和域之间用点“.”连接。同样, 访问结构的各个域时, 也是“结

构名。域名”的格式，结构的各个域可以按照其本身的数据类型进行相应的各种运算。

我们知道，MATLAB 中的数据类型都以数组为基础，同样，上面建立的结构 temp 在 MATLAB 中认为是包含一个结构的结构数组，也就是说，这时的 temp 等价于 temp(1)。那么，如果要在结构数组 temp 中再添加一个结构，即 temp(2)，其方法如下：

```
temp(2).city = 'Shanghai';
temp(2).year = 1999;
temp(2).data = [-7,-3,8; 13,23,28; 33,27,25; 22,13,9];
temp % 含有两个结构的结构数组将不直接显示域的内容
temp =
1x2 struct array with fields:
    city
    year
    data
temp.city % 这时结构数组的每个域都包括两个结构的域的内容
ans =
Beijing
ans =
Shanghai
temp(1).data - temp(2).data % 结构域的运算，注意访问结构数组中域的方式
ans =
    -3    -3    -3
    -3    -3    -3
    -3    -3    -3
    -3    -3    -3
```

除了这种方法外，结构还可以用函数 struct 建立，该函数的用法为：

```
s = struct('field1', values1, 'field2', values2, ...)
```

该函数用指定的域名和各个域的数据建立结构数组，如果数组中包含多个结构，而且各个结构域中的数据不尽相同，那么域的数据 values1、values2 ... 必须是单元数组（见下一节）。建立的结构数组将和单元数组的大小相同。

**注意** 注意比较下面建立的结构数组 tempmean 中的域 data 在两种赋值方式下的区别，其中第一种（单元数组）是正确的方法。

```
tempmean = struct('city',{'Beijing','Shanghai'},'year',{1999},'data',{13,16})
```

```
tempmean =
```

```
1x2 struct array with fields:
```

```
    city
    year
    data
```

```
tempmean(1).data % 该数据是正确的
```

```

ans =
13
tempmean = struct('city',{'Beijing','Shanghai'},'year',{1999},'data',[13,16]);
tempmean(1).data           % 该数据是错误的
ans =
13    16

```

访问和设置结构的域还可以通过函数 `getfield` 和 `setfield` 进行，另外，利用函数 `rmfield` 可以删除结构的域，它们的用法请参看帮助。

## 2. 数据的分配

函数 `deal` 把输入数据分配给输出数据，其用法为：

`[Y1,Y2,Y3,...] = deal(X)`: 把输入赋给所有的输出，等价于  $Y_1 = X, Y_2 = X, Y_3 = X, \dots$ 。

`[Y1,Y2,Y3,...] = deal(X1,X2,X3,...)`: 等价于  $Y_1 = X_1; Y_2 = X_2; Y_3 = X_3; \dots$ 。

这个函数对于结构和单元数组很有用，下面举例说明。

### 例 5.7 函数 `deal` 的应用

```

[city1, city2] = deal(tempmean.city)      % 把结构数组中的两个域分配给两个变量
city1 =
Beijing
city2 =
Shanghai
SS = {rand(2) ones(2,1) eye(2) zeros(2,1)};    % 建立单元数组SS（参见下一节）
[a,b,c,d] = deal(SS{:})        % 把SS中的每一个“单元”分配给一个变量
a =                               % 注意SS{:}和SS不同，SS是将整个单元数组分配给每个变量
    0.8913    0.4565
    0.7621    0.0185
b =
    1
    1
c =
    1    0
    0    1
d =
    0
    0

```

## 5.2.3 单元数组

单元数组 (cell array) 是一种比较特殊的 MATLAB 数组，它的每个元素都是一个“单元”，单元中包含其它的 MATLAB 数组。单元数组和结构的概念有些类似，只是单元数组中没有域名，而且更加灵活。

### 1. 单元数组的建立

建立单元数组有三种方法：

- 利用赋值语句
- 利用单元数组语法（花括号“{}”）
- 利用函数 cell

下面通过例子具体介绍。

### 例 5.1 建立单元数组的三种方法

#### (1) 利用赋值语句

赋值语句可以采用两种不同的方式，第一种是用小括号括起单元的下标，在赋值语句的右侧用花括号括起单元的内容，赋值语句及得到的单元数组 A 如下所示：

```
A(1, 1) = {'Beijing'}; % 单元(1, 1), 字符串, 记录城市名
A(1, 2) = {1999}; % 单元(1, 2), 标量, 记录年份
A(2, 1) = {13}; % 单元(2, 1), 标量, 全年平均数据
A(2, 2) = {[ -10,-6,5; 10,20,25; 30,24,22; 19,10,6]}; % 单元(2, 2), 矩阵, 一年的温度
A =
    'Beijing'    [      1999]
    [      13]    [4x3 double]
```

第二种方式是用花括号括起单元的下标，在赋值语句的右侧直接指定单元的内容，对于和上面相同的单元数组，赋值语句如下：

```
A{1, 1} = 'Beijing'; % 单元(1, 1)中的内容
A{1, 2} = 1999; % 单元(1, 2)中的内容
A{2, 1} = 13; % 单元(2, 1)中的内容
A{2, 2} = [-10,-6,5; 10,20,25; 30,24,22; 19,10,6]; % 单元(2, 2)中的内容
```

请大家仔细体会上述两种方式的区别，从中可以理解单元数组中花括号和小括号的不同含义。

#### (2) 利用单元数组的语法

也就是在花括号中直接赋值，单元与单元之间用逗号（或空格）和分号隔开，如下语句得到和上面相同的单元数组：

```
A = {'Beijing', 1999; 13, [-10,-6,5; 10,20,25; 30,24,22; 19,10,6]}
```

#### (3) 利用cell函数预分配

cell函数用来预分配指定大小的单元数组，有如下几种用法：

c = cell(n): 建立n×n的单元数组，单元是空矩阵。

c = cell(m, n) 或 c = cell([m, n]): 建立m×n的单元数组，单元是空矩阵。

c = cell(m, n, p, ... ) 或 c = cell([m n p ...]): 建立m×n×p×...的单元数组，单元是空矩阵。

c = cell(size(X)): 建立和X大小相同的单元数组。

用cell函数建立了空的单元数组以后，可以再用前面介绍的方法为单元数组赋值。

## 2. 单元数组的访问和显示

#### (1) 访问单元数组有两种方式：

- 用内容下标访（花括号）问单元的内容
- 用单元下标（小括号）访问单元子集

我们通过举例来比较这两种方法的区别。

### 例 5.9 单元数组的访问

仍然参照上面例子中建立的单元数组 A，如下一些命令分别用两种不同的方式访问 A，注意对比其不同的含义：

```
A{1,1}          % 访问第(1,1)个单元中的内容
ans =
Beijing
A(1,1)          % 访问第(1,1)个单元
ans =
'Beijing'
A{2,2}{1,:}      % 访问
ans =
-10   -6   5
B = A(1:2, 1)    % 把单元数组A第一列的两个单元赋给B，B也成为单元数组
B =
'Beijing'
[     13]
size(A(2, 2))    % 获取第(2,2)个单元的整体的大小
ans =
1   1
size(A{2, 2})    % 获取第(2,2)个单元中的内容（矩阵）的大小
ans =
4   3
```

(2) 一般单元数组中的内容是以压缩形式显示的（参见例 5.8），要查看单元数组中的具体内容可以用函数 `celldisp` 和 `cellplot`，其中 `cellplot` 以图形方式显示单元数组的结构。例如，如下命令将以图形方式显示单元数组 A 的结构，而且添加了图例，结果如图 5-2 所示：



图 5-2 以图形方式显示单元数组

```
cellplot(A, 'legend')
```

### 3. 基本操作

通过把该单元（注意不是单元内容）设置为空可以删除单元数组中的单元。

改变单元数组的大小可以用函数 `reshape` 来实现，但要注意改变形状时不能改变数组中的元素个数。

另外，单元数组也可以象一般数组那样用方括号“[ ]”进行连接。

下面来看相应的例子。

```
A(2, :) = [] % 删除 A 中第二行的单元
A =
    'Beijing'    [1999]
A = [] % 删除 A 中的所有单元
A =
    []
B = cell(4,3); % 建立一个 4×3 的单元数组 B
C = reshape(B,2,6) % 把 B 的形状改为 2×6，结果如下
C =
    []    []    []    []    []    []
    []    []    []    []    []    []
X = {1,2; 3,4} % 建立单元数组 X，每个单元只有一个标量
X =
    [1]    [2]
    [3]    [4]
Y = X' % 转置单元数组，和矩阵转置含义相同
Y =
    [1]    [3]
    [2]    [4]
[X, Y] % 水平连接单元数组，要求 X 和 Y 行数相同
ans =
    [1]    [2]    [1]    [3]
    [3]    [4]    [2]    [4]
[X; Y] % 垂直连接单元数组，要求 X 和 Y 列数相同
ans =
    [1]    [2]
    [3]    [4]
    [1]    [3]
    [2]    [4]
```

### 4. 用单元数组代替变量列表

单元数组可以代替用逗号（或空格）分隔的变量列表，例如函数的输入输出参数。

如果和花括号一起用冒号引用数组中的多个单元，那么 MATLAB 把每个单元的内容看

作一个独立的变量。下面举例说明这个特性。

请读者注意比较如下函数的输入输出参数中花括号、中括号和小括号的含义：

```
A(1) = {[1 2 3]};  
A(2) = {[1 1 1]};  
A(3) = {'string'};  
A(4) = {[4 5 6]};  
A{2:4} % 分别显示单元数组A中第2到4个单元  
ans =  
    1     1     1  
ans =  
string  
ans =  
    4     5     6  
conv(A{1:2}) % 单元数组的第1、2个单元分别作为函数conv的第1、2个参数  
ans =  
    1     3     6     5     3  
B = [A{1};A{2};A{4}] % 利用单元数组的单元建立矩阵  
B =  
    1     2     3  
    1     1     1  
    4     5     6  
[E{1:2}] = eig(B) % 函数eig的两个输出参数返回给单元数组的两个单元  
E =  
    [3x3 double]    [3x3 double]
```

### 5.3 程序控制语句

和其它高级语言一样，MATLAB 也提供了循环语句、条件转移语句等一些常用的控制语句，从而使得 MATLAB 语言的编程显得十分灵活。

MATLAB 支持的控制语句和 C 语言中的控制语句格式很相似，本节将介绍这些控制语句及其用法。

#### 5.3.1 循环语句

MATLAB 中的循环语句包括 for 循环和 while 循环两种类型。

##### 1. for 循环

for 循环的基本格式为：

```
for 循环变量 = 起始值 : 步长 : 终止值  
    循环体
```

```
end
```

注意 MATLAB 的 for 循环是以“end”结尾的，这和 C 语言的结构不同。

步长的缺省值是 1。步长可以在正实数或负实数范围内任意指定，对于正数，循环变量的值大于终止值时，循环结束；对于负数，循环变量的值小于终止值时，循环结束。

下面的循环的步长为-1，它将执行 10 次：

```
for i = 10 : -1 : 1
    y(i) = i;
end
```

执行后 y 的值为：

```
y =
    1     2     3     4     5     6     7     8     9     10
```

下面的双重循环为矩阵 a 赋值：

```
for i=1:3
    for j=1:3
        a(i,j)=i+j;
    end
end
```

循环执行后矩阵 a 的值为：

```
a =
    2     3     4
    3     4     5
    4     5     6
```

## 2. while 循环

while 循环的格式为：

```
while 表达式
    循环体
end
```

其执行方式为：若表达式为真（1），则执行循环体的内容，执行后再判断表达式是否为真（1），若不是则跳出循环体，向下继续执行。

如下的循环想要求出从 1 到多少的自然数和大于或等于 100：

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

结果为：

```
sum =
```

```
105
i =
14
```

### 5.3.2 条件转移语句

条件转移语句有 if 语句和 switch 语句两种。

1. if, else, elseif 语句

最简单的 if 语句格式为：

```
if 逻辑表达式
    执行语句
end
```

当逻辑表达式的值为真（1）时，则执行该结构中的执行语句内容，执行完之后继续向下进行；若逻辑表达式的值为假（0）时，跳过结构中的执行语句继续向下进行。

**例 5.14** 赋值语句

如下语句为向量 a 赋值：

```
for i = 1:10
    a(i) = i;
    if i>5
        a(i)=10-i;
    end
end
```

执行结果为：

```
a =
1     2     3     4     5     4     3     2     1     0
```

MATLAB 还提供了功能更完整的 if - else 格式和 if - elseif 格式，这两种格式分别如下所示：

```
if 逻辑表达式
    执行语句 1
else
    执行语句 2
end
```

和：

```
if    逻辑表达式 1
    执行语句 1
elseif 逻辑表达式 2
    执行语句 2
.....
end
```

if-else 格式的执行方式为：如果逻辑表达式的值为真，则执行语句 1，然后跳过语句 2

向下执行；如果为假则执行语句 2，然后向下执行。

**if - elseif** 格式的执行方式为：如果逻辑表达式 1 的值为真，则执行语句 1；如果为假，则判断逻辑表达式 2 的值是否为真，如果为真，则执行语句 2，否则向下执行。

为下式写赋值程序：

$$y = \begin{cases} 10 & x \geq 1 \\ 0 & -1 < x < 1 \\ -10 & x \leq -1 \end{cases}$$

程序为：

```
if x>=1
    y=10;
elseif x>-1 & x<1
    y=0;
else
    y=-10;
end
```

## 2. switch 语句

switch 语句的格式为：

switch 表达式（标量或字符串）

```
case 值1
    语句1
case 值2
    语句2
...
otherwise
    语句3
end
```

其执行方式为：表达式的值和哪种情况（case）的值相同，就执行哪种情况中的语句，如果都不同，则执行 otherwise 中的语句。switch 语句中也可以不包括 otherwise，这时如果表达式的值和列出的每种情况都不同，则继续向下执行。

下面的程序让用户输入三角函数名进行相应计算并画图：

```
t=-pi : 0.1 : pi;
trigname = input('Input trig function name : ');
switch trigname
    case 'sin'
        plot(t, sin(t))
    case 'cos'
```

```

plot(t, cos(t))
case 'tan'
    plot(t, tan(t))
otherwise
    break
end

```

式中，函数 `input` 常用于获取用户的键盘输入，输入的字符串要注意加单引号。用户输入列出的三种三角函数名后，即画出相应的函数图形，如果输入内容不在三种情况中，则用 `break` 命令跳出 `switch` 语句，当然这里的 `break` 命令也可以省略。

## 5.4 MATLAB 函数

建立一个新的 M 文件的一般方法是在 MATLAB 主菜单 File 下选择“New”→“M-file”，然后会出现 MATLAB 编辑器窗口，如图 5-2 所示。在该编辑器中输入程序代码后，在编辑器的 File 菜单下选择 Save 命令，出现保存文件对话框，指定文件名来保存输入的内容，这样就建立了一个新的 M 文件。

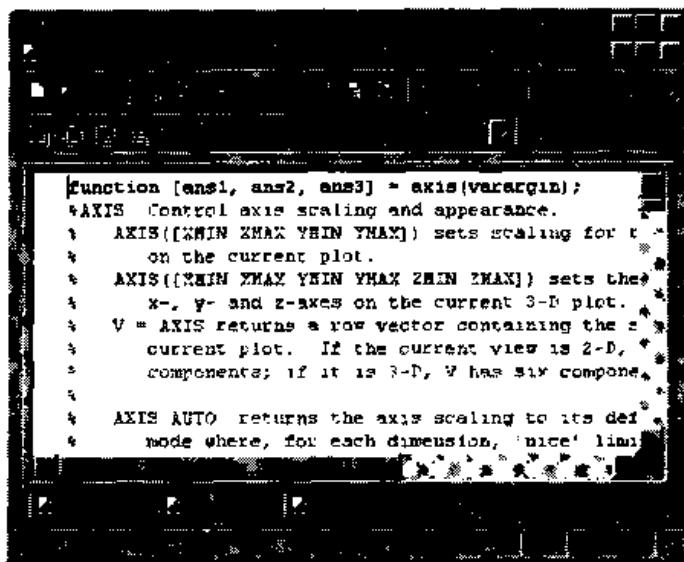


图 5-2 MATLAB 编辑器界面

MATLAB 编辑器/调试器（Editor/Debugger，一般简称编辑器）提供了基础文本编辑功能和 M 文件的调试工具，它具有 Windows 标准的多文档界面。

MATLAB 编辑器对于编写 M 文件比较方便，它有自动缩排功能，而且把关键字、字符串、注释用不同的颜色表示，便于区别。该编辑器提供的调试功能，可以在程序中设置多个断点进行在线调试。

当然，在其它任何文本编辑器中输入程序代码，然后保存为 M 文件的形式（扩展名为 “.m”）也同样可以建立 M 文件。

M 文件有两种形式：脚本和函数。

### 5.4.1 函数

#### 1. 脚本与函数

##### (1) 脚本

脚本是 M 文件的简单类型，它们没有输入输出参数，只是一些函数和命令的组合。类似于 DOS 下的批处理文件。

脚本可以在 MATLAB 环境下直接执行，它们可以访问存在于整个工作空间内的数据，由脚本建立的变量在脚本执行完后仍将保留在工作空间中，可以继续对其进行操作。直到使用 clear 命令清除了这些变量为止。

实际上我们前面各章用于举例的程序段都是脚本。例如，下面的一段程序就是脚本：

```
% A simple script
L=membrane(1);
surf(L)
pause
L=membrane(2);
surf(L)
pause
L=membrane(3);
surf(L)
pause
L=membrane(4);
surf(L)
```

这个脚本依次绘制各阶隔膜特征函数图，中间用 pause 命令暂停，按下任意键后继续绘制下一阶。我们看到这段程序只有执行的主体（除了帮助信息），不包括其它的任何格式限制，这就是脚本的特点。

##### (2) 函数

和其它高级语言一样，函数是 MATLAB 语言中最重要的组成部分，MATLAB 提供的各种工具箱中的 M 文件几乎都是以函数的形式给出的，MATLAB 的主体和各个工具箱本身就是一个庞大的函数库。

函数接收输入参数，返回输出参数。函数只能访问该函数本身工作空间中的变量，从 MATLAB 命令窗口或其它函数中也不能对该函数工作空间中的变量进行访问。

函数文件与脚本文件类似之处在于它们都是一个有 “.m” 扩展名的文本文件，而且函数文件和脚本文件一样，都是由文本编辑器所创建的外部文本文件。

#### 2. 函数的结构

下面是一个 MATLAB 提供的上下翻转矩阵的函数，文件名是 flipud.m:

```
function y = flipud(x)
%FLIPUD Flip matrix in up/down direction.
% FLIPUD(X) returns X with columns preserved and rows flipped
% in the up/down direction. For example,
%
```

```
% X = 1 4      becomes 3 6
%
%      2 5          2 5
%      3 6          1 4
%
% See also FLIPLR, ROT90, FLIPDIM.

%
% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.5 $ $Date: 1997/11/21 23:28:50 $

if ndims(x)~=2, error('X must be a 2-D matrix.'); end
[m,n] = size(x);
y = x(m:-1:1,:);
```

我们以 flipud 函数为例介绍 MATLAB 函数 M 文件的结构。

MATLAB 函数 M 文件通常由以下几个部分组成：

- 函数定义行
- H1 行
- 函数帮助文本
- 函数体
- 注释

下面我们分别介绍各部分的内容：

### (1) 函数定义行

函数 M 文件的第一行用关键字“function”把 M 文件定义为一个函数，并指定它的名字，它与文件名相同。同时也定义了函数的输入和输出参数。

注意：函数 M 文件的函数名和文件名必须相同

函数 flipud 的定义行是：

```
function y = flipud(x)
```

其中“flipud”为函数名，输入参数为“x”，输出参数为“y”。

如果函数有多个输入参数和（或）输出参数，那么参数之间用逗号分隔，多个输出参数用方括号括起来。如：

```
function [pos, newUp]=camrotate(a,b,dar,up,dt,dp,coordsys,direction);
function [ans1, ans2, ans3] = axis(varargin);
```

如果函数没有输出或没有输入，可以不写相应的参数，如：

```
function grid(opt_grid);
```

### (2) H1 行

所谓 H1 行指帮助文本的第一行，它紧跟在定义行之后。它以“%”符号开头，用于概括说明函数名和函数的功能。

函数 flipud 的 H1 行为：

```
%FLIPUD Flip matrix in up/down direction.
```

当在 MATLAB 命令行使用 lookfor 命令时，找到的相关函数将只显示 H1 行。

### (3) 帮助文本

帮助文本指位于在 H1 行之后函数体之前的说明文本。它同样以“%”符号开头，一般用来比较详细地介绍函数的功能和用法。

函数 flipud 的帮助文本为：

```
% FLIPUD(X) returns X with columns preserved and rows flipped
% in the up/down direction. For example,
%
%   X = 1 4      becomes  3 6
%           2 5             2 5
%           3 6             1 4
%
% See also FLIPLR, ROT90, FLIPDIM.
%
% Copyright (c) 1984-98 by The MathWorks, Inc.
% $Revision: 5.5 $ $Date: 1997/11/21 23:28:50 $
```

当在 MATLAB 命令行键入“help 函数名”时，就会同时显示 H1 行和帮助文本，也就是在定义行和函数体之间的文本（都以“%”符号开头）。

### (4) 函数体

函数体就是函数的主体部分，函数体包括进行运算和赋值操作的所有 MATLAB 程序代码。

函数体中可以有流程控制、输入输出、计算、赋值、注释，还可以包括函数调用和对脚本文件的调用。

函数 flipud 的函数体为：

```
if ndims(x)~=2, error('X must be a 2-D matrix.'); end
[m,n] = size(x);
y = x(m:-1:1,:);
```

### (5) 注释

除了函数开始独立的帮助文本外，还可以在函数体中添加对语句的注释。注释必须以“%”开头，MATLAB 在编译执行 M 文件时把每一行中“%”后面的内容全部作为注释，不进行编译。

我们可以为函数添加合法的注释文本，例如：

```
if ndims(x)~=2, error('X must be a 2-D matrix.'); end    % 判断输入矩阵维数是否正确
[m,n] = size(x);          % 求输入矩阵的大小
y = x(m:-1:1,:);         % 将矩阵上下翻转
```

为显示区别，我们对这几行在不同的位置添加了注释，这几种情况都是合法的。

» 注意：在前面讲过的这几部分中，定义行是一个 MATLAB 函数所必须的，其它各部分内容可以没有，这种函数称为空函数。

## 3. 函数参数

### (1) 参数传递规则

函数调用的过程实际上就是参数传递的过程。例如，在 MATLAB 工作空间用如下方式

调用函数 flipud:

```
a = [1 2; 3 4; 5 6];
y = flipud(a)
```

这个调用过程是把赋了值的变量“a”传递给函数中的输入参数“x”，然后，函数运算的返回值传递给输出参数“y”。

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 工作空间之间唯一的联系就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，以后就消失。因此，从一个调用到下一个调用，在函数工作空间的变量中存储信息是不可能的。不过，可以定义全局变量来解决这个问题，参见 5.1.1 节。

总之，在 MATLAB 工作空间定义的变量不会延伸到函数的工作空间；在函数内定义的变量也不会延伸到 MATLAB 的工作空间中。

从函数 M 文件内可以调用脚本文件。在这种情况下，脚本文件查看函数工作空间，不查看 MATLAB 工作空间。从函数 M 文件内调用的脚本文件不必用调用函数编译到内存。函数每调用一次，它们就被打开和解释。因此，从函数 M 文件内调用脚本文件减慢了函数的执行速度。

函数还可以递归调用，即函数能调用它们本身。递归调用函数功能在许多应用场合是有用的。在编制要递归调用的函数时，必须确保会终止，否则 MATLAB 会陷入死循环。

## (2) 检查函数参数的个数

函数空间内有两个固定的变量“nargin”和“nargout”分别等于实际的输入参数个数和输出参数个数。

而函数 nargchk 用于检查输入参数的个数是否正确，调用格式为：

```
msg = nargchk(low, high, n)
```

该函数的作用是：如果个数“n”不在“low”和“high”之间就返回错误信息。其中“n”一般是输入参数的个数。

例如，下面的格式检查输入参数的个数是否在 1 和 3 之间（包括 1 和 3），如果不是，比如 0 或 4 个，就会提示错误信息。

```
msg = nargchk(1, 3, nargin)
```

注意：函数可以按少于函数 M 文件中所规定的输入和输出参数个数进行调用，但不能用多于函数 M 文件中所规定的输入和输出参数数目。如果输入和输出参数数目多于函数 M 文件中定义行所规定的数目，则调用时自动返回一个错误提示。

下面我们看一下 MATLAB 中的 title.m 文件如何使用变量“nargin”和“nargout”来处理参数个数不同的情况，我们省去了函数中的帮助文本，在相关部分添加了注释：

```
function hh = title(string,varargin)
%
ax = gca;
```

```

h = get(ax,'title');
% 当参数大于 1 时, 如果第一个参数后的参数个数是奇数, 则提示错误
% 函数 fix 对数据向 0 的方向取整
if nargin > 1 & (nargin-1)/2-fix((nargin-1)/2),
    error('Incorrect number of input arguments')
end
%Over-ride text objects default font attributes with
%the Axes' default font attributes.
set(h, 'FontAngle', get(ax, 'FontAngle'), ...
    'FontName', get(ax, 'FontName'), ...
    'FontSize', get(ax, 'FontSize'), ...
    'FontWeight', get(ax, 'FontWeight'), ...
    'Rotation', 0, ...
    'String', string, varargin{:});
% 如果输出参数个数大于 0, 那么把变量 h 作为返回值赋给输出参数 hh
% 如果输出参数个数为 0 (没有输出), 则不执行此操作
if nargout > 0
    hh = h;
end

```

### (3) 传递可变个数的参数

用固定变量“varargin”和“varargout”可以传递任意个输入参数给函数, 也能够返回任意多个输出。

下面我们通过函数 zoom 来说明可变个数参数的使用, 我们去掉了原函数的一部分内容, 添加了一些相关的注释:

```

function out = zoom(varargin)
% 按照输入参数的个数分情况处理
switch nargin,
case 0,
    fig=get(0,'currentfigure');
    if isempty(fig), return, end
    zoomCommand='toggle';
case 1,
    % 如果输入参数是 1 个, 根据参数的类型决定如何处理
    if isstr(varargin{1}),
        fig=get(0,'currentfigure');
        if isempty(fig), return, end
        zoomCommand=varargin{1};
    else
        scale_factor=varargin{1};
    end
end

```

```

zoomCommand='scale';
fig = gcf;
end % if
case 2,
    % 如果输入参数为 2 个，那么第一个参数是图形句柄，第二个参数是 zoom 命令
    fig=varargin{1};
    zoomCommand=varargin{2};
otherwise,
    % 用函数nargchk判断，如果输入参数个数不在0和2之间，则提示错误
    error(nargchk(0, 2, nargin));
end % switch nargin

```

## 5.4.2 子函数

### 1. 子函数

函数文件可以包含一个以上的函数，文件中的第一个函数是主函数，其后的所有函数都是子函数，子函数只能被同一文件中的主函数和其它子函数访问。函数文件名和主函数名相同。

例如，下面的第一个函数 caldemo 是主函数，在它里面调用子函数 rsp 来进行计算并画图。

```

function caldemo()
%% Main function
y1 = 1:0.1:10;
y2 = y1.^2;
yy = rsp(y1,y2);
plot(y1,yy)

function y = rsp(y1,y2)
% Sub function
H1=y2./y1;
H2=y1./y2;
y=(H2+H1)/2;

```

主函数对子函数的调用和同一文件内的子函数之间的互相调用都是通过参数传递实现的，都满足同样的参数传递规则。

### 2. 私有函数

私有函数指位于 private 子目录下的函数。它们只能被上一层目录的函数访问，对于其它目录的函数都是不可见的，因而私有函数可以和其它目录下的函数重名。

根据私有函数的特点，我们可以在自己的工作目录下建立一个名为 private 的子目录，这个目录下的函数名可以根据需要任意指定，不必担心会和其它目录下的函数重名，因为 MATLAB 在查找一般 M 函数文件之前查找私有函数。

**注意：**私有函数的目录 private 不要添加到 MATLAB 查找路径之中。

### 5.4.3 函数的执行

#### 1. 函数的执行

MATLAB 头一次执行一个函数文件时，它打开相应的文本文件并将命令编译后存到存储器的内部，以加速执行以后所有的调用。如果函数包含了对其它函数的调用，它们也同样被编译到存储器。普通的脚本文件不被编译，即使它们是从函数内调用；打开脚本文件，调用一次就逐行解释执行。

当函数执行到达 M 文件终点，或者碰到返回命令 return，就结束执行和返回。return 命令提供了一种结束一个函数的简单方法，而不必到达文件的终点。

MATLAB 运行时，它缓存了存储在 Toolbox 目录和 Toolbox 目录内的所有子目录中所有的 M 文件的名字和位置。这使 MATLAB 能够很快地找到和执行 M 文件。被缓存的 M 文件函数当作是只读的。如果执行这些函数，以后又发生变化，MATLAB 将只执行以前编译到内存的函数，不管已改变的 M 文件。而且，在 MATLAB 执行后，如果 M 文件被加到 Toolbox 目录中，那么它们将不出现在缓存里，因此不可利用。所以，在 M 文件函数的使用中，最好把它们存储在 Toolbox 目录外。

在 Toolbox 目录外，MATLAB 跟踪 M 文件的修改日期。所以，当遇到一个以前编译到内存的 M 文件函数时，MATLAB 把已编译的 M 文件的修改日期与在磁盘上的 M 文件比较。如果日期是相同的，MATLAB 执行已编译的 M 文件。相反，如果在磁盘上的 M 文件是新的，MATLAB 清除以前已编译的 M 文件，且编译这个新的 M 文件。

#### 2. 命令/函数二元性

MATLAB 下的很多命令在执行时可以指定操作对象，如：

```
help plot
clear t
which demo
```

这些带操作对象的命令都可以用字符串的形式把操作对象作为参数，把命令以函数的形式表示，如下：

```
help('plot')
clear('t')
which('demo')
```

这就是 MATLAB 的命令/函数二元性，也就是如下的命令形式：

command argument

都可以写成函数的形式，如下：

```
command('argument')
```

这个特点使得 MATLAB 命令可以具有很多灵活性，因为函数的参数可以是变量，那么我们就可以设置不同的条件，传递给命令随条件变化的字符变量作为操作对象。

# 第 6 章 数据分析

## 6.1 线性方程组

### 6.1.1 线性方程求解

在工程计算中，一个很重要的问题是线性方程组的求解。在矩阵表示方法中，上述问题可以表述为：给定两个矩阵  $A$  和  $B$ ，求  $X$  的唯一解使得：

$$AX = B \text{ 或 } XA = B$$

MATLAB 求解这种问题时并不计算矩阵的逆。尽管在标准数学中没有矩阵除法概念，在此 MATLAB 求解线性方程组时，采用前面介绍的除法运算符 “/” 和 “\” 求解。

$X = A \setminus B$ ： 表示求矩阵方程  $AX = B$  的解。

$X = B / A$ ： 表示求矩阵方程  $XA = B$  的解。

对于  $X = A \setminus B$ ，要求矩阵  $A$  和  $B$  有相同的行数， $X$  和  $B$  有相同的列数，它的行数等于矩阵  $A$  的列数。对于  $X = B / A$ ，则行和列的角色相互交换。

实际上，线性方程形式  $AX = B$  比形式  $XA = B$  出现的频率更高一些，相应地，左除 “\” 比右除 “/” 用的更多一些。并且由于  $(B/A)' = (A \setminus B)'$ ，因此下面仅讨论 “\” (左除)。

矩阵  $A$  并不要求是方阵，如果矩阵  $A$  的维数是  $m \times n$ ，则有三种情况：

1)  $m = n$ : 适定方程组，寻求精确解

2)  $m > n$ : 超定方程组，寻求最小二乘解

3)  $m < n$ : 不定方程组，寻求基本解，其中至多有  $m$  个非零元素

针对不同的情况，左除算子采用不同的算法求解。

#### 1. 适定方程组

一般形式的线性方程组为：

$$Ax = b \text{ 或 } AX = B$$

其中  $A$  是个方阵， $b$  是一个列向量， $B$  是个方阵， $X$  是一个与  $A$ ， $B$  同样大小的方阵。它们的求解命令为：

$$x = A \setminus b$$

$$X = A \setminus B$$

如果  $A$  是奇异的，则  $AX = B$  的解或者不存在，或者存在但不唯一。

当  $A$  接近奇异时， $A \setminus B$  将给出警告信息，如果发现  $A$  是奇异的，一方面给出警告信息，另一方面给出结果为  $\inf$ 。

#### 2. 超定方程组

线性超定方程组经常遇到的问题就是试验数据的曲线拟合问题。对于超定方程，在 MATLAB 中，利用左除命令寻求它的最小二乘解。其调用格式为：

$c = E \setminus y$

$E^*c$  并不是精确的等于  $y$ , 但是它们的差值小于原始数据的测量误差。

### 3. 不定方程组

不定线性方程组未知量的个数多于方程数。解是不唯一的。MATLAB 将寻求一个基本解, 其中至多只有  $m$  个非零元素。特解由列主元 QR 分解求得。

#### 6.1.2 矩阵分解

把一个给定的矩阵分解成几个“较简单的”矩阵连乘积形式, 无论在理论证明和科学上都是十分重要的, 本节介绍几个矩阵分解的方法, 相关函数见表 6-1。

表 6-1 矩阵分解函数

命 令	功 能	说 明
chol	Cholesky 分解	对称正定矩阵的分解, 用于求解方程组
lu	矩阵 LU 分解	将矩阵采用 LU 分解, 直接求解线性方程组
qr	正交三角分解	将矩阵分解为正交矩阵或酉矩阵和上三角矩阵
svd	奇异值分解	将矩阵分解为一个对角阵和两个酉矩阵
schur	Schur 分解	将矩阵分解为一个 Schur 矩阵和一个酉矩阵

在 MATLAB 中, 线性方程组的求解主要基于三种基本的矩阵分解: 对称正定矩阵的 Cholesky 分解, 一般方阵的 Gaussian 消去法和矩形矩阵的正交分解。这三种分解通过函数 chol, lu 和 qr 进行。所有这三种分解都使用三角矩阵, 其中所有的元素位于对角线以上或以下的都为 0。包含三角矩阵的线性方程组不论使用左除还是右除都可以简单、快速地求解。

本小节将具体介绍表 6-1 列出的五种矩阵分解方法。

##### 1. Cholesky 分解

设  $A$  为一个  $n \times n$  的对称正定矩阵, 则存在对角线为正的上三角矩阵  $R$ , 使得:

$$A = R' R$$

并不是所有的对称矩阵都可以进行 Cholesky 分解, 能进行此种分解的矩阵必需是正定的。这意味着  $A$  的所有对角线元素都是正的, 而非对角线元素不会太大。Cholesky 分解同样也可以应用于复矩阵, 任何能进行 Cholesky 分解的复矩阵满足  $A' = A$ , 是 Hermitian 正定的。

Cholesky 分解的函数调用格式为:

1)  $R = \text{chol}(X)$ : 其中是  $X$  对称正定矩阵,  $R$  是上三角阵, 使得  $R'^*R = X$ 。如果  $X$  是非正定的, 将给出出错信息。

2)  $[R, p] = \text{chol}(X)$ : 返回两个输出参数, 不会给出出错信息。如果  $X$  是正定的, 则  $p$  等于 0,  $R$  同上; 如果  $X$  不是正定的, 则  $p$  等于正整数,  $R$  是个上三角矩阵, 它的阶数为  $q = p-1$ , 使得  $R'^*R = X(1:q, 1:q)$ 。

Cholesky 分解允许对线性方程组  $A^*x = b$  进行如下替换:

$$R'^*R^*x = b$$

由于左除算子可以处理三角矩阵, 因此可以快速地解出:

$x = R \setminus (R' \backslash b)$

如果  $A$  是  $n \times n$  方阵,  $\text{chol}(A)$  的计算复杂性是  $O(n^3)$ , 而接下来的左除算子的复杂性只有  $O(n^2)$ .

$X = [3 \ 2 \ 3$

$2 \ 4 \ 1$

$3 \ 1 \ 7];$

$R = \text{chol}(X)$

$R =$

1.7321	1.1547	1.7321
0	1.6330	-0.6124
0	0	1.9039

$R' * R$

$ans =$

3.0000	2.0000	3.0000
2.0000	4.0000	1.0000
3.0000	1.0000	7.0000

## 2. LU 分解

Gaussian 消去法或 LU 分解, 可以将任何方阵表示为一个下三角矩阵  $L$  和一个上三角矩阵  $U$  的乘积。

$A = L * U$

其中  $L$  矩阵为下三角阵, 当对它进行置换以后, 对角线元素为 1。

函数  $lu$  的调用格式为:

$[L, U] = lu(A)$ : 在  $U$  中存储一个上三角矩阵, 在  $L$  中存储一个“心理上的”下三角矩阵, 实际上它是下三角矩阵和置换矩阵的乘积, 使得  $A = L * U$ 。

$[L, U, P] = lu(A)$ : 得到一个下三角矩阵  $L$ , 上三角矩阵  $U$  和置换矩阵  $P$ , 使得  $P * A = L * U$ 。

LU 分解允许线性方程系统  $A * x = b$  如下进行快速计算:

$x = U \setminus (L \setminus b)$

矩阵行列式的值和矩阵的逆也可以利用 LU 分解如下进行:

$\det(A) = \det(L) * \det(U)$

$\text{inv}(A) = \text{inv}(U) * \text{inv}(L)$

$A = [1 \ 2 \ 3$

$4 \ 5 \ 6$

$7 \ 8 \ 9];$

$[L, U, P] = lu(A)$

$L =$

1.0000	0	0
--------	---	---

```

0.1429    1.0000      0
0.5714    0.5000    1.0000
U =
7.0000    8.0000    9.0000
0    0.8571    1.7143
0        0    0.0000
P =
0    0    1
1    0    0
0    1    0

```

### 3. 正交分解

正交分解或 QR 分解，将矩形矩阵分解为一个正交矩阵和一个上三角矩阵的乘积：

$$A = Q R \quad \text{或} \quad A P = Q R$$

其中 Q 是正交矩阵，R 是一个上三角矩阵，P 是置换矩阵。

QR 分解的函数 qr 对矩阵进行三角分解，它适用于方阵和矩形矩阵。它将一个矩阵表示成实型的正交阵或复数酉矩阵和上三角矩阵的乘积。函数用法为：

- 1)  $[Q, R] = qr(X)$ : 生成一个上三角矩阵 R 和正交矩阵 Q，使得  $X = Q^* R$ 。其中 R 与 X 同维。
- 2)  $[Q, R, E] = qr(X)$ : 生成置换矩阵 E、上三角矩阵 R 和正交矩阵 Q，使得  $X^* E = Q^* R$ 。列置换矩阵 E 如此选取，可使  $\text{abs}(\text{diag}(R))$  是递减的。
- 3)  $[Q, R] = qr(X, 0)$  和  $[Q, R, E] = qr(X, 0)$ : 生成一种“经济”的分解。其中 E 是一个置换向量，使得  $Q^* R = X(:, E)$ 。列置换向量如此选取，使得  $\text{abs}(\text{diag}(R))$  递减。

```
A = [1 2 3 4
```

```
    3 4 5 6
```

```
    5 6 7 8];
```

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

```
Q =
```

```
-0.1690    0.8971    0.4082
-0.5071    0.2760   -0.8165
-0.8452   -0.3450    0.4082
```

```
R =
```

```
-5.9161   -7.4374   -8.9586   -10.4799
0        0.8281    1.6562     2.4842
0        0       -0.0000        0
```

```
[Q, R, E] = qr(A, 0)
```

```
Q =
```

```
-0.3714    0.8339    0.4082
-0.5571    0.1516   -0.8165
```

-0.7428 -0.5307 0.4982

R =

-10.7703	-5.7566	-7.4278	-9.0991
0	-1.3646	-0.9097	-0.4549
0	0	-0.0000	0.0000

E =

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

#### 4. 奇异值分解

奇异值分解在矩阵分析中占有极为重要的位置。奇异值分解的定义是：

对矩阵  $A \in C^{m \times n}$  (不失一般性, 设  $m > n$ ), 若存在酉矩阵  $U \in C^{m \times m}$ ,  $V \in C^{n \times n}$  使得

$$A = U \Sigma V^T$$

其中,  $\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ 0 & & & \ddots \\ & & & 0 \end{bmatrix}_{m \times n}$ ,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$

那么,  $U$ 、 $\Sigma$ 、 $V$  称为矩阵  $A$  的奇异值分解三对组。这里的上标 “T” 表示共轭转置。

MATLAB 提供的奇异值分解函数 svd 的调用格式如下:

SVD (A): 得到一个包含 diag (S) 的向量。

[U, S, V] = SVD (A): 产生一个对角阵 S, 它与矩阵 A 维数相同, 且其有一个降阶的非负对角元素, 而 U 与 V 是酉矩阵, 使得  $X = U * S * V'$ 。

[U, S, V] = SVD (A, 0): 得到一种“有效大小”的分解结果, 如果 A 是  $m \times n$  矩阵 ( $m > n$ ), 那么只计算 U 矩阵的前 n 行, 且 S 矩阵是  $n \times n$  阶的。

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

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

U =

0.3435	0.7629	-0.5449	-0.0551
0.4400	0.3262	0.6855	0.4796
0.5365	-0.1105	0.2638	-0.7940
0.6329	-0.5472	-0.4044	0.3695

S =

16.9804	0	0
0	1.2902	0

```

    0      0      0.0000
    0      0      0

```

V =

```

0.3159 -0.8565 0.4082
0.5459 -0.1878 -0.8165
0.7760 0.4808 0.4082

```

[U, S, V] = svd(A, 0)

U =

```

0.3435 0.7629 -0.5449
0.4400 0.3262 0.6855
0.5365 -0.1105 0.2638
0.6329 -0.5472 -0.4044

```

S =

```

16.9804      0      0
      0 1.2902      0
      0      0 0.0000

```

V =

```

0.3159 -0.8565 0.4082
0.5459 -0.1878 -0.8165
0.7760 0.4808 0.4082

```

## 5. Schur 分解

Schur 分解的含义为：

$$A = USU^T$$

其中 A 必须是方阵，U 是一个正交矩阵，S 是块上三角矩阵，由对角线上的  $1 \times 1$  和  $2 \times 2$  块组成。特征值由 S 的对角线和块给出，而 U 的列比一系列特征向量给出了更多的数值特性。Schur 分解可以对缺陷矩阵进行。

Schur 分解函数的调用格式为：

1) [U,S] = schur(A): 生成 Schur 形式矩阵 S 和酉矩阵 U，使得  $A = U^*S^*U^T$  和  $U^*U = eye(size(A))$ 。

2) S = schur(A): 仅仅返回 Schur 形式矩阵 S。

复数 Schur 形式矩阵是一个上三角阵，矩阵的特征值位于对角线上。实数 Schur 形式矩阵实特征值在对角线上，而复特征值位于  $2 \times 2$  块对角线上。

如果 A 实矩阵，schur 返回实数 Schur 形式矩阵，如果 A 是复矩阵，schur 返回复数 Schur 形式矩阵。函数 rsf2csf 可以将实数形式转化为复数形式。

```

A = [1   2   3   4
      3   4   5   6
      5   6   7   8
      7   8   9   0];

```

**[U, S] = schur(A)**

**U =**

0.2638	0.2846	0.8263	0.4082
0.4536	0.3258	0.1464	-0.8165
0.6434	0.3670	-0.5335	0.4082
0.5574	-0.8235	0.1057	0.0000

**S =**

20.2105	3.2312	3.0171	-0.0000
-0.0000	-7.4091	-0.6756	-0.0000
0	0	-0.8014	0.0000
0	0	0	-0.0000

### 6.1.3 矩阵特征值与特征向量

如果 A 是  $n \times n$  方阵,  $n$  个  $\lambda$  值满足关系式:

$$Av = \lambda v$$

则  $\lambda$  为 A 的特征值,  $v$  为 A 的特征向量。

特征值问题是指方程  $Ax = \lambda x$  的非平凡解问题。其中 A 是一个  $n \times n$  矩阵, x 是一个长度为 n 的列向量,  $\lambda$  是一个标量。满足方程的 n 个值是特征值, 对应的 x 值是特征向量。在 MATLAB 中, eig 函数求解特征值, 给出特征向量。

广义特征值问题是指方程  $Ax = \lambda Bx$  的非平凡解问题。其中 A 和 B 都是  $n \times n$  的矩阵,  $\lambda$  是一个标量。满足方程的  $\lambda$  是广义特征值, 对应的 x 值是广义的特征向量。

一些有缺陷的矩阵不能够对角化, 不能进行特征值分解。

由 A 的特征值构成的对角矩阵, 以及由对应的特征向量构成矩阵 V 的各列, 满足:

$$AV = VD$$

如果 V 是非奇异的, 则这就是矩阵 A 的特征值分解。

特征值分解的函数 eig 调用格式为:

1) D = eig(A): 返回矩阵 A 的特征值

2) [V, D] = eig(A): 生成特征值矩阵 D 和特征向量构成的矩阵 V, 使得  $A * V = V * D$ 。矩阵 D 由 A 的特征值在主对角线构成的对角矩阵。V 是由 A 的特征向量按列构成的矩阵。

3) [V, D] = eig(A, 'nobalance'): 计算矩阵的特征值和特征向量, 而不采用预先平衡。通常, 预先平衡增加了特征值和特征向量的计算精度。然而, 如果一个矩阵包含由于舍入误差引入的小的元素, 平衡过程有可能将它们放大, 使得它们和原始矩阵中的其它元素大致相当, 从而导致错误的特征值。在这种情况下, 使用 'nobalance' 选项。

4) D = eig(A, B): 如果 A 和 B 是方阵, 则返回一个向量, 其中包含广义特征值。

5) [V, D] = eig(A, B): 生成对角矩阵 D 和全元素矩阵 V, 其中 D 包含广义特征值, V 由相应的特征向量按列构成。使得  $A * V = B * V * D$ 。特征向量单位化使得每一个向量的范数等于 1。

A = [1 2 3]

```
2 3 4  
3 4 5];  
B = [1 2 3  
    4 5 6  
    7 8 9];  
D = eig(A)  
D =  
-0.0000  
-0.6235  
9.6235  
[V, D] = eig(A)  
V =  
0.4082 -0.8277 0.3851  
-0.8165 -0.1424 0.5595  
0.4082 0.5428 0.7339  
D =  
-0.0000 0 0  
0 -0.6235 0  
0 0 9.6235  
[V, D] = eig(A, B)  
V =  
0.9636 -0.8111 0.4082  
-0.1482 0.3244 -0.8165  
-0.2224 0.4867 0.4082  
D =  
0.3333 0 0  
0 1.0000 0  
0 0 -0.4286  
A*V  
ans =  
-0.0000 1.2978 0.0000  
0.5930 1.2978 0  
1.1860 1.2978 0  
B*V*D  
ans =  
-0.0000 1.2978 0.0000  
0.5930 1.2978 0.0000  
1.1860 1.2978 0
```

### 6.1.4 其它矩阵函数

在 MATLAB 中提供了一些用于矩阵分析和计算的函数，参见表 6-2。

表 6-2 矩阵函数

函数	功能	函数	功能
<code>funm</code>	计算矩阵的函数值 <code>funm(A, 'function')</code>	<code>null</code>	矩阵的零空间
<code>pinv</code>	求矩阵的伪逆	<code>rank</code>	矩阵的秩
<code>expm</code>	矩阵的指数	<code>condeig</code>	对应于特征值的条件数
<code>logm</code>	矩阵的对数	<code>norm</code>	向量和矩阵的范数
<code>sqrtm</code>	矩阵的平方根	<code>orth</code>	矩阵的列空间
<code>cond</code>	矩阵的条件数	<code>subspace</code>	子空间的夹角
<code>det</code>	方阵的行列式值		

下面分别介绍几个主要函数的用法。

#### 1. `sqrtm`

该函数的调用格式为：

`X = sqrtm(A)`

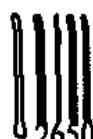
计算出的 `X` 是矩阵 `A` 的平方根，也就是 `X*X = A` 的解。

如果矩阵 `A` 的特征值的实部都是非负的，那么 `X` 中的元素是实数。如果矩阵 `A` 的一些特征值有负的实部，那么将得到复数结果。如果矩阵 `A` 是奇异的，那么它不具有平方根，MATLAB 将给出相应的警告信息。

```

A = [1 2 4; 2 4 1; 4 1 2];
eig(A)
ans =
    16.1168
   -1.1168
   -0.0000
sqrtm(A)
ans =
    1.0143 + 0.9521i  0.5084 - 0.1686i  1.1231 - 0.7834i
    0.5084 - 0.1686i  1.9364 + 0.0299i  0.2010 + 0.1388i
    1.1231 - 0.7834i  0.2010 + 0.1388i  1.3216 + 0.6447i
B = [3 2 3; 2 4 1; 3 1 7];
eig(B)
ans =
    0.7943
    3.9408

```



```
sqrtm(B)
```

```
ans =
```

1.4661	0.5630	0.7304
0.5630	1.9146	0.1322
0.7304	0.1322	2.5395

### 2. expm

**expm(A)** 用来计算矩阵 A 的幂。该函数是借助 Pade 近似法用比例和自乘算法进行运算的。

但当 A 有一个特征值矩阵 D 和一个特征向量矩阵 V 时，则不采用上述算法，而是由下式计算它的幂：

```
expm(A) = V*diag(exp(diag(D)))/V
```

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

```
    0 1 0
```

```
    1 0 1];
```

```
B = expm(A)
```

```
B =
```

2.7183	1.0000	2.7183
1.0000	2.7183	1.0000
2.7183	1.0000	2.7183

```
Bm = expm(A)
```

```
Bm =
```

4.1945	0	3.1945
0	2.7183	0
3.1945	0	4.1945

### 3. logm

**L=logm(A)** 是求矩阵 A 的对数。它是函数 **expm(A)** 的反函数。

如果矩阵 A 的特征值的实部都是非负的，那么 L 是唯一的。如果矩阵 A 的一些特征值有负的实部，那么将得到复数结果。如果 A 是奇异阵，那么它没有对数，MATLAB 将给出相应的警告信息。

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

```
B = [ 3 2 3; 2 4 1; 3 1 7];
```

```
eig(A)
```

```
ans =
```

16.1168
-1.1168
-0.0000

```
eig(B)
```

```

ans =
    0.7943
    3.9408
    9.2650
logm(A)
ans =
-5.3738 + 2.7896i 11.9341 - 0.4325i -5.3475 - 0.5129i
12.2439 - 0.7970i -22.1906 + 2.1623i 12.5537 - 1.1616i
-4.7279 - 1.2421i 12.8634 - 1.5262i -4.1346 + 1.3313i
logm(B)
ans =
    0.4430    0.7083    0.7951
    0.7083    1.1880    0.0045
    0.7951    0.0045    1.7362

```

#### 4. funm 函数

函数 funm 的调用格式为：

```
F = funm(A, 'function')
```

其中要求 A 必需是方阵，'function' 是按照数组运算规则进行运算的函数名。例如 funm(A, 'sqrt') 等同于 sqrtm(A)。

当然，如果有专用的矩阵函数，一般不必用函数 funm 进行计算。但 MATLAB 中大多数数学函数都是采用数组的运算规则进行运算的，而且没有相应的矩阵运算函数，如三角函数，这种情况下只有借助于 funm 函数来完成相应的矩阵运算。

为便于比较，我们用例 6.7 中的矩阵 A，结果和该例是一样的，如下：

```

A = [1 2 4; 2 4 1; 4 1 2];
funm(A, 'sqrt')
ans =
    1.0143 + 0.9521i  0.5084 - 0.1686i  1.1231 - 0.7834i
    0.5084 - 0.1686i  1.9364 + 0.0299i  0.2010 + 0.1388i
    1.1231 - 0.7834i  0.2010 + 0.1388i  1.3216 + 0.6447i

```

下面我们比较一下矩阵函数和原数组函数的区别：

```

A = [pi/6 pi/4
      pi/3 3*pi/4];
sin(A)
ans =
    0.5000    0.7071
    0.8660    0.7071
funm(A, 'sin')
ans =

```

```
0.1864    0.0764
0.1018    0.3646
```

### 5. norm 函数

函数 norm 用来求矩阵或向量的范数。

1) 对于矩阵 X, 它的调用格式为:

**norm(X)**: 计算 X 的最大奇异值, 即  $\max(\text{svd}(X))$ 。

**norm(X,2)**: 和 **norm(X)**功能相同。

**norm(X,1)**: 计算 X 的 1-范数, 最大列之和, 即  $\max(\sum(\text{abs}((X))))$ 。

**norm(X,inf)**: 计算 X 的无穷大范数, 最大行之和, 即  $\max(\sum(\text{abs}((X'))))$ 。

**norm(X,'fro')**: 计算 F-范数, 即  $\sqrt{\sum(\text{diag}(X^*X)))}$ 。

2) 对于向量 V, 它的调用格式为:

**norm(V,P)**: 即  $\sum(\text{abs}(V).^P)^{(1/P)}$ 。

**norm(V)**: 即 **norm(V,2)**。

**norm(V,inf)**: 即  $\max(\text{abs}(V))$ 。

**norm(V,-inf)**: 即  $\min(\text{abs}(V))$ 。

#### (1) 矩阵的范数

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

```
X =
```

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

```
norm(X)
```

```
ans =
```

```
20.1409
```

```
norm(X, 1)
```

```
ans =
```

```
25
```

```
norm(X, inf)
```

```
ans =
```

```
24
```

```
norm(X, 'fro')
```

```
ans =
```

```
20.1742
```

#### (2) 向量的范数

```
V = X(:, 1)
```

```
V =
```

```

3
6
7
norm(V, 1)
ans =
    17
norm(V, 1/2)
ans =
    61.2665
norm(V)
ans =
    9.7468
norm(V, inf)
ans =
    7
norm(V, -inf)
ans =
    1

```

### 6. pinv 函数

函数 pinv 用于求矩阵的伪逆 (Pseudo-inverse)，它的调用格式为：

**X = pinv (A) 或 X = pinv (A, tol)**

它产生一个矩阵 X 与 A' 有同样的维数，使得  $A * X * A' = A$ ， $X * A * X' = X$  且  $AX$  和  $X'A$  是 Hermitian 矩阵，计算基于奇异值分解 svd (A)，而任何小于零奇异值阈值的奇异值被处理为零。阈值可以由用户通过参数 “tol” 指定，如果不指定，则使用缺省阈值，它的计算方法是：

**max (size (A)) \* norm (A) \* eps**

当用伪逆取求解列不满秩超定最小二乘问题时，它所给出的解是最小范数最小二乘解。

```

A = [1 2 3
      2 3 4
      3 4 5
      4 5 6];
X = pinv(A)
X =
   -0.7500   -0.3333    0.0833    0.5000
   -0.1000   -0.0333    0.0333    0.1000
    0.5500    0.2667   -0.0167   -0.3000
A*X
ans =
   0.7000    0.4000    0.1000   -0.2000

```

```

0.4000 0.3000 0.2000 0.1000
0.1000 0.2000 0.3000 0.4000
-0.2000 0.1000 0.4000 0.7000

```

**X\*A****ans =**

```

0.8333 0.3333 -0.1667
0.3333 0.3333 0.3333
-0.1667 0.3333 0.8333

```

**A\*X\*A****ans =**

```

1.0000 2.0000 3.0000
2.0000 3.0000 4.0000
3.0000 4.0000 5.0000
4.0000 5.0000 6.0000

```

### 7. null 函数

函数 **null** 计算零空间，它的用法为：

**Z = null(A)**: 通过奇异值分解得到矩阵 A 零空间的正交基。A\*Z 近似为 0，而 Z'\*Z = I。

**Z = null(A, 'r')**: 零空间的“有理”基，A\*Z 等于 0。

**A = [1 2 3; 3 4 5; 4 5 6; 6 7 8];****Z1 = null(A)****Z1 =**

```

0.4082
-0.8165
0.4082

```

**A\*Z1****ans =**

```

1.0e-015 *
-0.4441
-0.4441
-0.8882
-0.4441

```

**Z2 = null(A, 'r')****Z2 =**

```

1
-2
1

```

**A\*Z2****ans =**

```
0
0
0
0
```

### 8. cond 函数

cond 函数用于计算矩阵的条件数，它的调用格式为：

**cond(X)**: 计算 2-范数条件数，即矩阵 X 最大奇异值与最小奇异值的比。条件数越大说明矩阵越奇异。

**cond(X, p)**: 计算 p-范数条件数，即  $\text{norm}(X, p) * \text{norm}(\text{inv}(X), p)$ ，其中 p 为 1、2、inf 或 'fro'。

请比较以下几种矩阵和它们的条件数大小：

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

```
cond(A)
```

```
ans =
```

```
11.6646
```

```
B = [1 2 3; 1 2 3; 1 2 3];
```

```
cond(B)
```

```
ans =
```

```
Inf
```

```
C = [1 2 3; 2 3 4; 3 4 5];
```

```
cond(C)
```

```
ans =
```

```
3.2244e+016
```

### 9. 矩阵的行列式和秩

矩阵的行列式函数 **det** 和秩的计算函数 **rank** 都很简单，来看下面的例子。

仍然用例 6.14 中的 A、B、C 三个矩阵：

```
rank(A)
```

```
ans =
```

```
3
```

```
rank(B)
```

```
ans =
```

```
1
```

```
rank(C)
```

```
ans =
```

```
2
```

```
det(A)
```

```
ans =
```

```

29
det(B)
ans =
0
det(C)
ans =
0

```

## 6.2 非线性数值计算

非线性数值计算是 MATLAB 的重要功能, MATLAB 把它作为基本功能之一置于目录“\toolbox\matlab\funfun”下, 本节中讲述的函数除 fplot 外都在此目录下。

### 6.2.1 非线性函数最小值点

#### 1. 求单变量函数最小值点

fminbnd 函数用于求单变量函数的最小值点, 它采用黄金分割查找和抛物线插值算法, 其调用格式为:

```
[x, fval, exitflag, output] = fminbnd(fun, x1, x2, options, p1, p2, ...)
```

其中输入参数 fun、x1、x2 是必需的, 其余参数允许缺省。fun 是被计算最小值点的单变量函数(目标函数)名称字符串, x1、x2 是目标函数自变量的取值范围。

p1, p2, ... 是向目标函数传递的附加参数。

options 是一个结构类型的变量, 用于指定算法的优化参数。该结构的内容用函数 optimset 进行定义(关于 optimset 函数的用法请参看在线帮助), 如果没有优化参数要设置, 可以在 options 的位置用 “[ ]” 做为占位符。函数 fminbnd 使用了结构 options 的四个域, 它们的含义分别为:

- Display: 显示的层次, “off” 不显示输出内容, “iter” 显示每次迭代的输出, “final” 只显示最后一次输出。

- MaxFunEvals: 所允许的函数的最大求值次数。

- MaxIter: 所允许的最大迭代次数。

- TolX: 终止迭代的容差值(x)。

输出参数 fval 是目标函数 fun 在 x 处的值。

exitflag 描述函数 fminbnd 退出的情况, 大于 0 表示函数收敛到了解 x, 0 表示达到了所允许的函数最大求值次数。

output 是一个结构变量, 包含了计算的信息, 它共有三个域:

- output.algorithm: 使用的算法。
- output.funcCount: 函数求值的次数。
- output.iterations: 迭代次数。

(1) 目标函数是 MATLAB 函数, 注意函数名要加引号:

```
x = fminbnd('sin', pi, 2*pi)
```

Optimization terminated successfully:

the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004

x =

4.7124

## (2) 目标函数为自定义的内联函数

首先用 inline 函数定义内联函数 f，同时指定 f 的参数顺序，如下命令定义内联函数  $f(x, a) = x^4 - ax - 5$ ：

```
f = inline('x.^4-a*x-5', 'x', 'a');
```

下面在区间[0, 2]上求内联函数 f (注意函数名不加引号) 的最小值点，设置终止迭代的容差为 1e-12，迭代过程中显示每一步的输出，函数的最大求值次数为 2 次，给内联函数 f 的参数 “a” 赋值为 2，命令如下：

```
[xout, fval, exitflag, output] = fminbnd(f, 0, 2, optimset('TolX', 1e-12, 'Display', 'iter', 'MaxFunEvals', 2), 2)
```

执行结果为：

% 显示每一次迭代的输出，包括次数、自变量值、相应的目标函数值和算法

Func-count	x	f(x)	Procedure
1	0.763932	-6.18728	initial
2	1.23607	-5.13777	golden
3	0.472136	-5.89458	golden

% 最后一步退出时显示退出的状态

Exiting: Maximum number of function evaluations has been exceeded

- increase MaxFunEvals option.

% 给出在迭代过程中目标函数值最小的点

xout =

0.7639

% 给出相应的目标函数值

fval =

-6.1873

% 退出标志，0 表示达到了允许的函数求值次数而退出

exitflag =

0

% 相关信息

output =

iterations: 3

funcCount: 3

下面我们对函数的执行次数不做限制，直到解收敛为止，并只显示迭代的最后一步输出，命令如下：

```
[xout, fval, exitflag, output] = fminbnd(f, 0, 2, optimset('TolX', 1e-12, 'Display', 'end'), 2)
```

执行结果如下，请注意正常结束时输出信息的区别：

```
Optimization terminated successfully:
the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-012
xout =
    0.7937
fval =
    -6.1906
exitflag =
    1
output =
    iterations: 12
    funcCount: 12
    algorithm: 'golden section search, parabolic interpolation'
```

## 2. 求多变量函数最小值点

fminsearch 函数用于求多变量函数的最小值点，它采用的是 Nelder-Mead 单纯形算法，该函数的调用格式为：

```
[x, fval, exitflag, output] = fminsearch(fun, x0, options, p1, p2,...)
```

其中的大部分参数和函数 fminbnd 中的参数相同，这里只介绍有区别的参数：

x0 是最小值点的初始值，一般是猜测的。

这里的结构 options 的域比函数 fminbnd 中增加了一个：TolFun，指终止计算的目标函数容差值。

退出标志 exitflag 也比函数 fminbnd 中增加了一种情况：小于 0，表示目标函数不收敛。fminsearch 中必需的参数为 fun 和 x0。

经典的多维寻优测试函数——Rosenbrock “香蕉” 函数的表达式为： $f(x) = 100(x_2 - x_1^2)^2 + (a - x_1)^2$ ，我们来看一下求该函数最小值点的方法：

首先，在建立 M 文件“banana.m”（当然也可以构造内联函数，这里只是要介绍另外一种常用的方法），并保存到 MATLAB 搜索路径范围内，banana.m 内容如下：

```
function f = banana(x,a)
if nargin < 2, a = 1; end
f = 100*(x(2)-x(1)^2)^2+(a-x(1))^2;
```

然后，用函数 fminsearch 求最小值点，给参数“a”赋值为  $\sqrt{2}$ ，调用方式如下：

```
[x, fval] = fminsearch('banana', [-1.2, 1], optimset('TolX', 1e-8), sqrt(2))
```

执行结果为：

Optimization terminated successfully:

```
the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-008
and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-004
```

```
x =
```

```
1.4142    2.0000
fval =
4.2065e-018
```

### 6.2.2 单变量函数零点

函数 fzero 用于求单变量函数的零点，其调用格式为：

```
[x,fval,exitflag,output] = fzero(fun,x0,options,p1,p2,...)
```

其中参数  $x_0$  指定搜索的起始点，同时也预定搜索零点的大致位置，一个函数可能有多个零点，但只给出离  $x_0$  最近的那个零点。

结构 options 这里只有两个域：Display 和 TolX。

输出参数 exitflag 大于 0 表示找到了函数的零点，小于 0 表示没有找到零点或在搜索过程中遇到了无穷大的函数值。

其余几个参数和求最小值点的函数相同。

首先建立 M 文件：

```
function f = fun1(x,a)
f = x.^4-a*x-5;
```

给参数 “a” 赋值为 4，结构 options 使用缺省值，求函数 fun1 的零点：

```
[x,fval,exitflag,output] = fzero('fun1',2,[],4)
```

执行结果为：

```
Zero found in the interval: [1.84, 2.1131].
```

```
x =
```

```
1.8812
```

```
fval =
```

```
-1.7764e-015
```

```
exitflag =
```

```
1
```

```
output =
```

```
iterations: 14
```

```
funcCount: 14
```

```
algorithm: 'bisection, interpolation'
```

### 6.2.3 绘制函数曲线

前面讲过的绘图函数在绘制一个函数  $y = f(x)$  的图形时，必需首先定义自变量  $x$  的一组取值点，再求出这些点上的函数值，然后根据这两组数值确定的数据点绘制出所需的图形。

这里要介绍绘制函数曲线的专用函数：fplot。该函数在指定范围内绘制函数曲线，绘图数据由 fplot 函数在指定范围内自适应产生，根据函数曲线的平滑程度自动调整数据点的密度。因而，对于那些不够平滑的函数，用 fplot 绘制的函数曲线比较准确。

fplot 函数有如下几种调用格式：

- `fplot('function', limits)`: 直接绘制函数曲线, `function` 是函数名或函数表达式, `limits` 是指定 `x` 轴范围的向量 “[`xmin xmax`]”, 也可以同时指定 `x` 轴和 `y` 轴的范围, 向量格式为 “[`xmin xmax ymin ymax`]”。
- `fplot('function', limits, LineSpec)`: 用 `LineSpec` 指定线条的属性。
- `fplot('function', limits, tol)`: `tol` 是相对误差容限, 缺省值为 `2e-3`。通过 `tol` 可以确定 `x` 的最大点数为 “(`1/tol`)+1”。
- `fplot('function', limits, tol, LineSpec)`: 同时指定相对误差容限和线条属性。
- `[x, Y] = fplot(...)`: 不画图, 返回计算出的横纵坐标数据。可以用 `plot(x, Y)` 命令绘制曲线。

为便于比较, 我们在一幅图中分四个子窗口, 然后分别用 `fplot` 和 `plot` 绘图, 命令如下:

% 在第一个子窗口(左上角)用fplot绘制第一条函数曲线

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

% 在第二个子窗口(右上角)用plot也绘制第一条函数曲线, 要先给横坐标赋值

```
x = (0:0.1:2*pi)';
```

```
y = abs(exp(-j*x*(0:9))*ones(10,1));
```

```
subplot(2,2,2), plot(x,y)
```

% 在第三个子窗口(左下角)用fplot绘制第二条函数曲线, 指定相对误差容限

```
subplot(2,2,3), fplot('sin(1 ./ x)', [0.01 0.1], 1e-3)
```

% 在第四个子窗口(右下角)用plot也绘制第二条函数曲线, 先给横坐标赋值

```
x = (0.01:0.001:0.1)';
```

```
subplot(2,2,4), plot(x,sin(1 ./ x))
```

结果如图 6-1 所示, 可以看出, 用 `fplot` 绘制的函数曲线比用一般的 `plot` 命令更加精确, 后者没能完全绘制出函数的极值点。

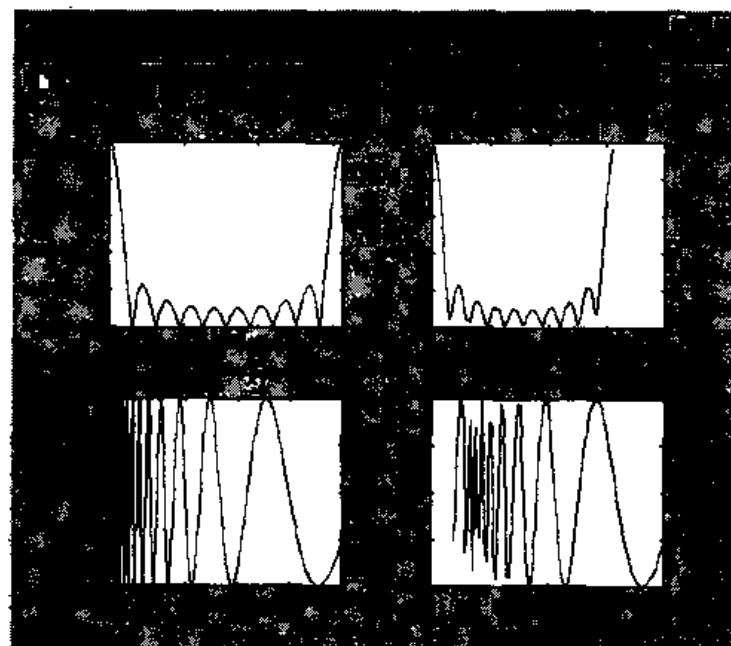


图 6-1 函数 `fplot` 与 `plot` 的绘图效果比较

### 6.2.4 常微分方程(组)数值解

求微分方程数值解是数值计算的基本内容,由于微分方程的多样性也有很多不同的解法, MATLAB 给出了七种解法,分别由七个不同的函数来完成,它们是: `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`。

在介绍这些函数的适用范围之前,我们首先介绍一个概念——刚性(stiff)问题——定性的讲,对于一个常微分方程组,如果其 Jacobian 矩阵的特征值相差十分悬殊,那么这个方程组就称为刚性方程组。对于刚性方程组,为了保持解法的稳定,步长选取很困难。有些解法不能用来解刚性方程组,有的解法由于对稳当性的要求不严格,可以用来解决刚性问题。下面我们分别介绍这七种函数对应解法的特点:

1) `ode45`: 四阶/五阶龙格-库塔(Runge-Kutta)法。它属于单步解法(只需要前一步的解就可以计算出当前的解),不需要附加初始值,因而,计算过程中随意改变步长也不会增加任何附加计算量,这是它的重要优点。通常 `ode45` 对很多问题是首先试用的最好方法。但它不能解刚性问题。

2) `ode23`: 二阶/三阶龙格-库塔算法,属于单步解法。在误差容许范围较宽而且存在轻微刚性时比 `ode45` 效果好。

3) `ode113`: 可变阶次的 Adams-Basforth-Moulton PECE 算法,属于多步解法(需要前几步的解来计算当前的解)。比 `ode45` 更适合于误差容许范围要求比较严格的情况。不能解刚性问题。

4) `ode15s`: 可变阶次的数值微分公式(NDFs)算法,属于多步解法。如果采用 `ode45` 效果很差或者失败,可以考虑试用 `ode15s`,它可以解刚性问题。

5) `ode23s`: 基于修正的 Rosenbrock 公式,属于单步解法。比 `ode15s` 更适用于误差容许范围较宽的情况。可以解决一些采用 `ode15s` 效果不好的刚性问题。

6) `ode23t`: 采用自由内插法实现的梯形规则(trapezoidal rule)。适用于系统存在适度刚性并要求无数值衰减的情况。

7) `ode23tb`: TR-BDF2 方法的实现,即龙格-库塔公式的第一级采用梯形规则、第二级采用 Gear 法。对于误差容限比较宽的情况,比 `ode15s` 效果更好。可以解刚性问题。

这些函数的用法完全相同,以 `ode45` 为例,其调用格式为:

`[T, Y] = ode45('F', tspan, y0, options, p1, p2...)`

式中的参数'F'、`tspan`、`y0` 是必需的,其它参数可选。这些参数的含义如下:

'F' 定义是常微分方程组的文件名字符串。这些函数是针对一阶常微分方程组设计的,对于高阶微分方程,必须先化为形如  $\dot{x} = f(x, t)$  的一阶微分方程组。注意定义常微分方程组的函数必须以 `t`、`x` 为输入参数(顺序不能变),以  $\dot{x}$  为输出参数,且输出  $\dot{x}$  的必须是列向量。

`tspan` 是一个向量,一般形如: `[t0 tfinal]`,指定积分的起始值和终止值;如果需要得到指定时刻的结果,也可以这样取值: `[t0, t1, ..., tfinal]`。

`y0` 是初始状态向量。

`options` 是一些可选的综合参数,由函数 `odeset` 进行设置(该函数的用法请参看在线帮助)。

`p1`、`p2` 是传递给'F'的参数。

输出参数 T 为时间列向量, Y 是数值解的矩阵, 它的每一行对应着列向量 T 中的一个时间值。

### (1) 非刚性问题

已知常微分方程组及初始条件:  $\begin{cases} y_1' = y_2 y_3 & y_1(0) = 0 \\ y_2' = -y_1 y_3 & y_2(0) = 1 \\ y_3' = -2y_1 y_2 & y_3(0) = -1 \end{cases}$

这是一阶微分方程组, 故不必再转化, 第一步就可以直接建立该方程组的描述文件 `ode1.m`:

```
function dy = ode1(t, y) % t 和 y 的先后顺序不能变
dy = zeros(3,1); % 首先定义 dy 为三个零元素的列向量
dy(1) = y(2) * y(3); % 输入微分方程的表达式
dy(2) = -y(1) * y(3);
dy(3) = -2 * y(1) * y(2);
```

第二步, 调用适当的函数解方程组并绘图, 结果如图 6-2 所示。

% 选用 `ode45` 算法, 在 0 到 12 的时间段上求解

```
[T, Y] = ode45('ode1',[0 12],[0 1 -1]);
% 分别用不同的线形绘制微分方程组的解
plot(T,Y(:,1),'-',T,Y(:,2),'*',T,Y(:,3),'-')
```

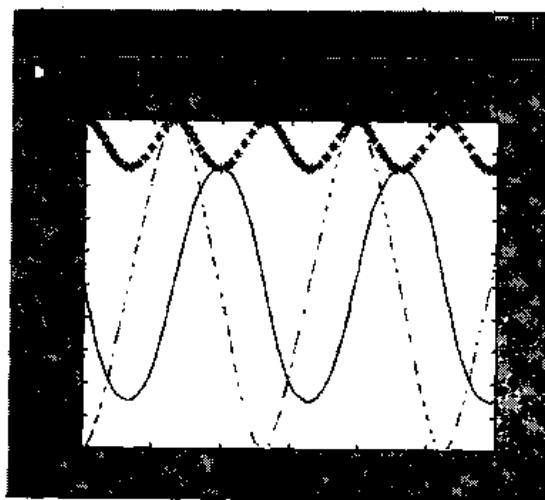


图 6-2 非刚性微分方程组的解

### (2) 刚性问题

已知二阶微分方程及初始条件:  $y'' = 1000(1 - y^2)y' - y \quad y(0) = 0, y'(0) = 1$

求该方程的解, 首先要把该方程降为一阶, 令  $y_1 = y, y_2 = y_1'$ , 则该方程可化为如下的  
一阶常微分方程组:

$$\begin{cases} y_1' = y_2 \\ y_2' = 1000(1 - y_1^2)y_2 - y_1 \end{cases}$$

第二步，建立该方程组的描述文件 `ode2.m`:

```
function dy = ode2(t, y)
dy = zeros(2,1);
dy(1) = y(2);
dy(2) = 1000*(1 - y(1)^2)*y(2) - y(1);
```

第三步，解方程组。这是一个刚性问题，因而选用 `ode15s` 算法；求解的区间设为 [0 3000]，并用 `odeset` 函数设置相对误差容限为 `1e-4`，绝对误差容限  $y_1$ 、 $y_2$  分别为 `1e-4` 和 `1e-5`。解的结果如图 6-3 所示。

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-5]);
[T,Y] = ode15s('ode2',[0 3000],[2 0],options);
plot(T,Y(:,1),'o') % 矩阵 Y 的第一列  $y_1$  为原二阶微分方程的解
```

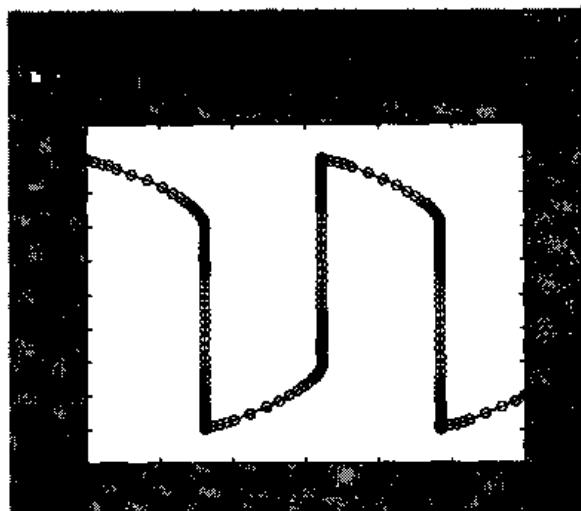


图 6-3 刚性微分方程的解

### 6.2.5 函数的数值积分

MATLAB 提供了 `quad` 和 `quad8` 两个函数用于求函数的数值积分。这两个函数使用了不同的算法：函数 `quad` 采用低阶法——自适应递推辛普生法，函数 `quad8` 采用高阶法——自适应递推牛顿-柯西法。对于“软奇异值”函数，比如  $\int \sqrt{x} dx$ ，`quad8` 比 `quad` 效果更好。

但是，这两个函数都不能解决可积的奇异值问题，如  $\int \frac{1}{\sqrt{x}} dx$ 。

函数 `quad` 和 `quad8` 递推的层次限制在 10 层以内，达到这个限制时会提示警告信息。

函数 `quad` 和 `quad8` 的调用格式是完全相同的，以 `quad` 为例：

```
q = quad('fun', a, b, tol, trace, p1, p2,...)
```

其中，'fun' 是被积函数文件名或表达式字符串； $a$ 、 $b$  分别是积分的上、下限； $tol$  是相对误差容限，用来控制积分精度，缺省值是 `1e-3`； $trace$  若取 1 则用图形展现积分过程，取 0 则不画图，缺省是 0； $p1$ 、 $p2$  是传递给被积函数的参数。

用函数 quad8 在区间[0.01 1]上求  $\sqrt{x}$  的数值积分，设置相对误差容限为 1e-4，并画图显示积分过程，命令如下：

```
quad8('sqrt',0.01,1,1e-4,1)
```

积分过程如图 6-4 所示，积分结果为：

```
ans =
```

```
0.6660
```

下面，改用函数 quad 加以比较，命令如下：

```
quad('sqrt',0.01,1,1e-4)
```

该函数虽然最后也计算出了积分结果，但在积分过程中，给出了警告信息，提示递推层次达到了限制，如下所示（这里省略了一些中间的内容）：

```
Warning: Recursion level limit reached in quad. Singularity likely.
```

```
> In E:\MATLABR11\toolbox\matlab\funfun\quad.m (quadstp) at line 102
```

```
In E:\MATLABR11\toolbox\matlab\funfun\quad.m (quadstp) at line 141
```

```
.....
```

```
Warning: Recursion level limit reached 2 times.
```

```
> In E:\MATLABR11\toolbox\matlab\funfun\quad.m at line 80
```

```
ans =
```

```
0.6660
```

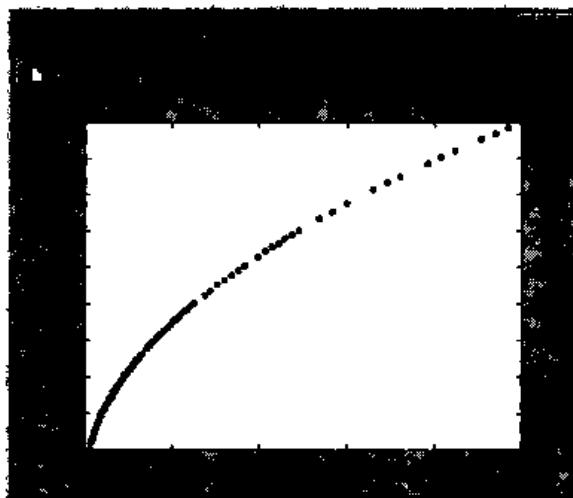


图 6-4 用图形展示积分过程

## 6.3 多项式

MATLAB 中的多项式函数位于目录\toolbox\matlab\polyfun 下，这是 MATLAB 提供的基本运算功能之一。

### 6.3.1 多项式表示法

MATLAB 采用行向量表示多项式，将多项式的系数按降幂次序存放在行向量中。多项

式  $P(x) = a_0x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$  的系数行向量为:  $P = [a_0 \ a_1 \ \dots \ a_n]$ , 注意顺序必须是从高次幂向低次幂。

为了将整个多项式输入 MATLAB, 可以采用类似于向量创建的方法, 直接输入多项式的系数行向量。例如, 多项式  $x^4 + 3x^2 - 5x + 1$  可以用系数向量 [1 0 3 -5 1] 表示, 注意多项式中缺少的幂次要用“0”补齐。

### 6.3.2 求多项式的根, 由根创建多项式

函数 roots 用来求多项式的根。它的调用形式为:

```
R = roots(C)
```

其中输入参数 C 是多项式的系数, 用行向量表示, 输出参数 R 是多项式的根, 一般用列向量表示。

函数 poly 由给定的根创建多项式。它的调用形式为:

```
C = poly(R)
```

按照输入参数 R 的不同形式, 该函数的返回值有两种情况:

1) R 为  $N \times N$  矩阵, 那么返回值 C 将是  $N+1$  个元素的行向量, 是该矩阵特征多项式的系数。

2) R 为向量, 那么返回值将是多项式的系数行向量, 该多项式的根向量为 R。

以多项式  $x^3 + 5x^2 - 9x + 3$  为例:

```
roots([1 5 -9 3])
```

```
ans =
```

```
-6.4641  
1.0000  
0.4641
```

```
poly([-6.4641,1,0.4641])
```

```
ans =
```

```
1.0000 5.0000 -9.0000 3.0000
```

下面看一下 poly 函数求矩阵特征多项式系数的用法:

```
poly(rand(4))
```

```
ans =
```

```
1.0000 -2.7334 1.2135 -0.6543 0.1155
```

### 6.3.3 多项式乘和除

多项式的乘法和除法运算实质就是多项式系数向量的卷积和解卷运算。函数 conv 和 deconv 用来实现这些运算。

长度为 m 的向量 a 和长度为 n 的向量 b 的卷积 c 定义为:

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

式中，向量  $c$  的长度为  $(m+n-1)$ 。

相应的 conv 函数调用格式为：

$c = \text{conv}(a, b)$

解卷是卷积的逆运算，向量  $a$  对向量  $c$  进行解卷将得到商向量  $q$  和余量  $r$ ，并且满足：

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

相应的 deconv 函数调用格式为：

$[q, r] = \text{deconv}(c, a)$

以多项式  $a(x) = x^2 - 2x + 3$  和  $b(x) = 2x^3 + 4x^2 - 1$  为例：

```
a = [1 -2 3]; b = [2 4 0 -1];
c = conv(a, b)
```

$c =$

2 0 -2 11 2 -3

$[q, r] = \text{deconv}(c, a)$

$q =$

2 4 0 -1

$r =$

0 0 0 0 0 0

$[q, r] = \text{deconv}(b, a)$

$q =$

2 8

$r =$

0 0 10 -25

#### 6.3.4 多项式导数

polyder 函数用来计算多项式的导数，它不但可以计算单个多项式的导数，也可以用来计算两个多项式的乘积和商的导数。

该函数的调用格式有三种：

$\text{polyder}(p)$ : 只有一个输入参数时，计算系数向量为  $p$  的多项式的导数。

$\text{polyder}(a, b)$  或  $c = \text{polyder}(a, b)$ : 有两个输入参数、一个或零个输出参数时，计算多项式  $a * b$  的导数。

$[q, d] = \text{polyder}(a, b)$ : 有两个输出参数时，计算多项式的商  $a/b$  的导数，用  $q/d$  表示。

仍以上例的两个多项式为例：

```
a = [1 -2 3]; b = [2 4 0 -1];
```

$p = \text{polyder}(a)$

$p =$

```

2      -2
q = polyder(b)
q =
6      8      0
c = polyder(a, b)
c =
10      0     -6     22      2
[q, d] = polyder(b, a)
q =
2      -8      10     26      -2
d =
1      -4      10     -12      9

```

### 6.3.5 多项式的值

`polyval` 函数用来计算对于给定的自变量，多项式的值。该函数调用方式为：

```
y = polyval(p, x)
```

其中，`p` 为多项式的系数向量，`x` 是指定的自变量的值，可以是标量、向量或矩阵。如果 `x` 是向量或矩阵，那么该函数将对向量或矩阵中的每一个元素计算多项式的值，并返回给 `y`。

`polyvalm` 函数计算矩阵多项式的值，即以矩阵整体（注意不是矩阵元素）做为多项式的自变量进行计算。该函数用法为：

```
Y = polyvalm(p, X)
```

相当于用矩阵 `X` 去替换多项式  $p(x)$  中的 `x`。

在下面的例子中，请比较两个函数的区别。

```
p = [1 -2 3];
```

```
x = [1 -1];
```

```
polyval(p, x)
```

```
ans =
```

```
2      6
```

```
A = [1 -1; -1 1];
```

```
polyval(p, A)
```

```
ans =
```

```
2      6
```

```
6      2
```

```
polyvalm(p, A)
```

```
ans =
```

```
3      0
```

```
0      3
```

### 6.3.6 多项式曲线拟合

函数 `polyfit` 采用最小二乘法对给定数据进行多项式曲线拟合，最后给出拟合的多项式系数。该函数的调用方式为：

```
p = polyfit(x, y, n)
```

其中，`x` 和 `y` 是数据的横纵坐标向量，`n` 是规定的拟合多项式的阶数。返回多项式  $p(x)$  的系数向量 `p`。

```
x = 1:10; % 给出数据点的横纵坐标 x, y
y = sqrt(x)+3*sin(x);
p = polyfit(x, y, 5) % 把数据点拟合为 5 阶多项式
p =
    0.0074   -0.2320    2.6027   -12.4498   23.5114   -10.0216
% 把原始数据点和拟合结果绘制在同一幅图上，原始数据点用“o”表示，见图 6-5
plot(x, y, 'o', x, polyval(p, x), '-')
```

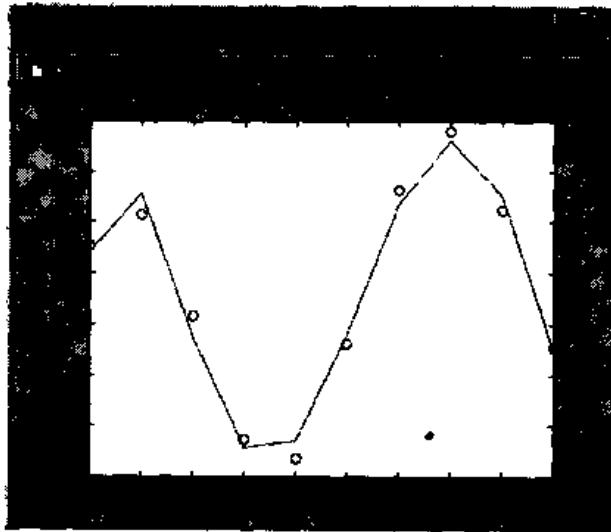


图 6-5 多项式拟合

### 6.3.7 部分分式展开

`residue` 函数用来将两个多项式之比用部分分式展开。它有两种调用方式：

(1) `[r, p, k] = residue(b, a)`: 求多项式之比  $b(s)/a(s)$  的部分分式展开，返回值 `r` 是留数、`p` 是极点、`k` 是直接项。如果没有重根，就是如下的形式：

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \cdots + \frac{r_n}{s - p_n} + k_s$$

向量 `r`、`p` 的长度和向量 `a`、`b` 之间有如下关系：

$$\text{length}(r) = \text{length}(p) = \text{length}(a)-1$$

当向量 `b` 的长度小于 `a` 时，向量 `k` 中没有元素，否则满足如下关系：

$$\text{length}(k) = \text{length}(b) - \text{length}(a) + 1$$

(2) `[b, a] = residue(r, p, k)`: 从部分分式得出多项式系数向量。

```

a = [1 0];
b = [1 1 -2];
[r, p, k] = residue(a, b)          % 求 a/b 的部分分式展开
r =
    0.6667
    0.3333
p =
    -2
    1
k =
    []
[r, p, k] = residue(b, a)          % 求 b/a 的部分分式展开
r =
    -2
p =
    0
k =
    1     1

```

## 6.4 插值

插值就是在原始数据点之间按照一定的关系插入新的数据点，以便更准确地分析数据的变化规律。MATLAB 中的插值函数位于目录\toolbox\matlab\polyfun 下，属于 MATLAB 提供的基本运算功能。

### 6.4.1 一维插值

一维插值就是对一维函数  $y = f(x)$  的数据进行插值，图 6-6 示意了插值的含义，图中的实心数据点  $x$  和  $Y$  表示原始数据，空心数据点  $xi$  和  $yi$  是插值数据点。

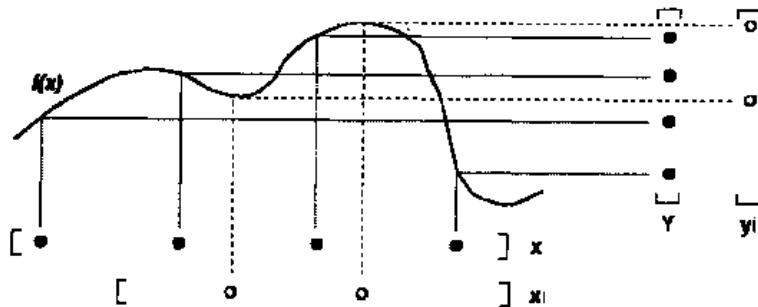


图 6-6 一维插值示意图

MATLAB 中的一维插值函数是 `interp1`，该函数调用格式为：

`yi = interp1(x, Y, xi, method)`

输入参数是原始数据点， $x$  是横坐标向量，必须是单调的； $Y$  是纵坐标向量或矩阵，如果  $Y$  是矩阵，那么插值按照  $Y$  的列向量进行，返回值  $yi$  将和矩阵  $Y$  的列数相等。

$xi$  是指定插值点的横坐标， $yi$  是在  $xi$  指定的位置计算出的插值结果。

参数  $method$  用来指定插值的方法，可供选择的方法有如下四种：

'nearest'：最近邻域插值（nearest neighbor interpolation），只选择最接近各估计点的粗略数据点。

'linear'：线性插值（linear interpolation），仅用作连接图上数据点。

'spline'：三次样条插值（cubic spline interpolation），更详细的内容见 6.4.3 节。

'cubic'：三次内插（cubic interpolation），使用三次多项式插值。

通过下面的例子，我们可以比较这几种方法的区别，并体会插值函数的用法。

如下一段程序首先生成一个周期正弦波的 7 个数据点，然后对这些点进行插值并绘制插值结果，如图 6-7 所示。

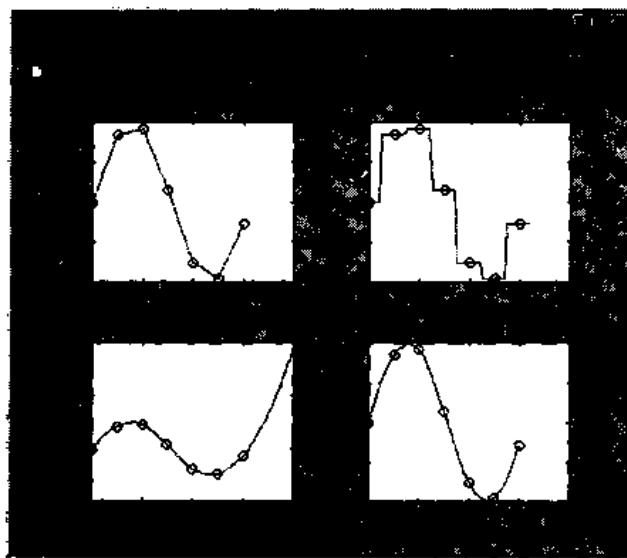


图 6-7 一维插值

```

x = 0:1:2*pi;
y = sin(x);
xi = 0:0.1:8;          % 设置插值点的横坐标，注意有在原数据点范围外的点 (>2π)
yi1 = interp1(x,y,xi);           % 缺省方式为线性插值
yi2 = interp1(x,y,xi,'nearest');
yi3 = interp1(x,y,xi,'spline');
yi4 = interp1(x,y,xi,'cubic');
subplot(2,2,1)           % 绘制插值结果和原数据点（用 ‘o’ 表示）
plot(x,y,'o',xi,yi1)
title('线性插值')
subplot(2,2,2)
plot(x,y,'o',xi,yi2)

```

```

title('最近邻域插值')
subplot(2,2,3)
plot(x,y,'o',xi,yi3)
title('三次样条插值')
subplot(2,2,4)
plot(x,y,'o',xi,yi4)
title('三次内插')

```

从图中看出，对同样的数据点，不同的插值方法得到的结果不同。因为插值是一个估计或猜测的过程，应用不同的估计规则会导致不同的结果。样条插值的结果更平滑（但不一定更精确），而且，这几种方法中只有样条插值可以估计原始数据点外侧的值，其它方法在数据点外侧会给出结果“NaN”。

#### 6.4.2 二维插值(interp2)

二维插值是基于与一维插值同样的基本思想，它是对两变量的函数  $z = f(x, y)$  进行插值。

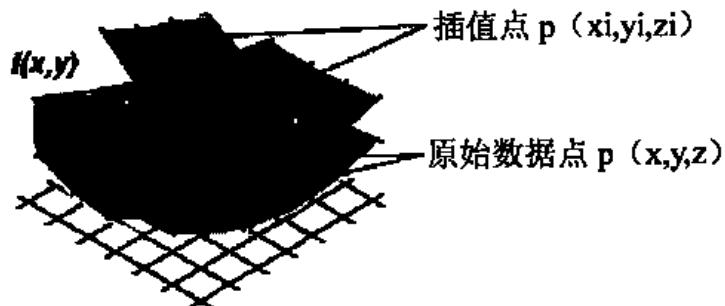


图 6-8 二维插值示意图

MATLAB 中二维插值的函数为 `interp2`，该函数的基本调用格式为：

`ZI = interp2(X, Y, Z, XI, YI, method)`

这里  $X$  和  $Y$  是两个独立向量，它们必须是单调的。 $Z$  是矩阵，是由  $X$  和  $Y$  确定的点上的值。 $Z$  和  $X$ 、 $Y$  之间的关系是：

$$Z(i, :) = f(X, Y(i)) \quad \text{和} \quad Z(:, j) = f(X(j), Y)$$

即：当  $X$  变化时， $Z$  的第  $i$  行与  $Y$  的第  $i$  个元素  $Y(i)$  相关，当  $Y$  变化时， $Z$  的第  $j$  列与  $X$  的第  $j$  个元素  $X(j)$  相关。如果省略  $X$  和  $Y$ ，就相当于假定  $X = 1:n$ ,  $Y = 1:m$ ，这里的  $m$ 、 $n$  分别是矩阵  $Z$  的行数和列数。

$XI$  是沿  $X$  轴插值的一个数值数组； $YI$  是沿  $Y$  轴插值的一个数值数组。

参数 `method` 指定插值方法，二维插值可供选择的插值方法有如下四种：

'linear': 双线性插值 (bilinear interpolation)，缺省值

'nearest': 最近邻域插值

'spline': 三次样条插值

'cubic': 双三次插值 (bicubic interpolation)

本例通过对 MATLAB 的经典函数 `peaks` 产生的三维高斯型分布进行插值来说明二维插

值函数的用法，插值前后的图形如图 6-9 所示，程序如下：

```
[X,Y] = meshgrid(-3:3;3);
Z = peaks(X,Y);
[XI,YI] = meshgrid(-3:1:3);
ZI = interp2(X, Y, Z, XI, YI, 'spline');
mesh(X,Y,Z)
hold on
mesh(XI,YI,ZI+15)
hold off
axis tight
```

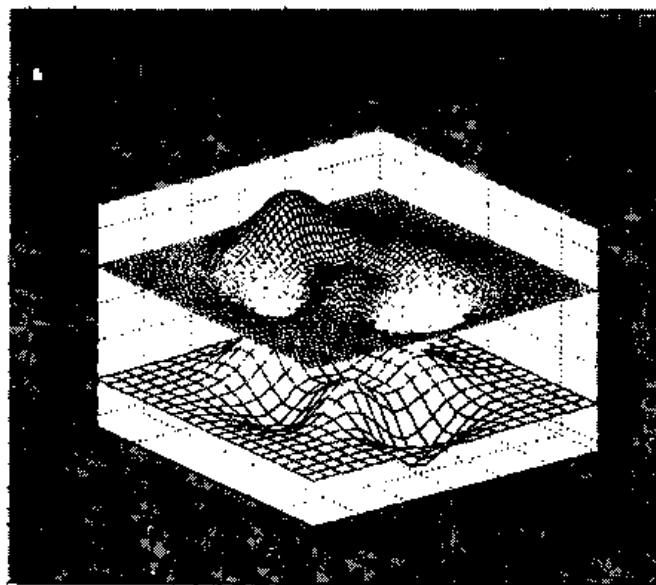


图 6-9 二维插值

除了一维和二维插值函数，MATLAB 还提供了三维插值函数 `interp3`、多维插值函数 `interpn`、应用 FFT 法的一维插值函数 `interpft` 等多个类似的函数，这些函数应用范围较窄，本书不再详细介绍，请读者参考 MATLAB 在线帮助。

#### 6.4.3 三次样条

为了用更光滑的曲线来拟合数据点，最常用的方法是用一个三阶多项式，即三次多项式，来对相邻数据点之间的各段建模，每个三次多项式的头两个导数与该数据点相一致。这种类型的插值被称为三次样条或简称为样条。

众所周知，使用高阶多项式的插值常常产生病态的结果。目前，有多种消除病态的方法。在这些方法中，三次样条是最常用的一种。

在三次样条中，要寻找三次多项式，以逼近每对数据点间的曲线。在样条术语中，这些数据点称之为断点。因为，两点只能决定一条直线，而在两点间的曲线可用无限多的三次多项式逼近。因此，为使结果具有唯一性，在三次样条中，增加了三次多项式的约束条件。通过限定每个三次多项式的一阶和二阶导数，使其在断点处相等，就可以较好地确定所有内部三次多项式。此外，近似多项式通过这些断点的斜率和曲率是连续的。然而，第

一个和最后一个三次多项式在第一个和最后一个断点以外，没有伴随多项式。因此必须通过其它方法确定其余的约束。最常用的方法，就是采用非扭结(not-a-knot)条件。这个条件强迫第一个和第二个三次多项式的三阶导数相等。对最后一个和倒数第二个三次多项式也做同样地处理。MATLAB 中提供的三次样条函数 `spline` 就采用了这种方法，它可以在计算三次多项式所覆盖的区间外，计算三次样条。当数据出现在最后一个断点之后或第一个断点之前时，则分别运用最后一个或第一个三次多项式来寻找内插值。

基于上述描述，人们可能猜想到，寻找三次样条多项式需要求解大量的线性方程。实际上，给定  $N$  个断点，就要寻找  $N-1$  个三次多项式，每个多项式有 4 个未知系数。这样，所求解的方程组包含有  $4*(N-1)$  个未知数。把每个三次多项式列成特殊形式，并且运用各种约束，通过求解  $N$  个具有  $N$  个未知系数的方程组，就能确定三次多项式。这样，如果有 50 个断点，就有 50 个具有 50 个未知系数的方程组。幸好，用稀疏矩阵（参见 6.6 节），这些方程组能够简明地列出并求解，这就是函数 `spline` 所使用的计算未知系数的方法。

函数 `spline` 调用方式有如下两种：

- 1) `yi = spline(x, y, xi)`
- 2) `pp = spline(x, y)`

一般情况下，使用第一种调用方式，`spline` 函数获取数据 `x` 和 `y` 以及插值点横坐标 `xi`，寻找拟合 `x` 和 `y` 的三次样条内插多项式，然后，计算这些多项式，对每个 `xi` 的值，寻找相应的 `yi`。

这种方式适合于只需要一组内插值的情况。不过，如果需要从相同数据里获取另一组内插值，再次计算三次样条系数是没有意义的。在这种情况下，可以采用第二种调用方式，这时，`spline` 返回一个称之为三次样条的 `pp` 形式或分段多项式形式的数组。这个数组包含了对于任意一组所期望的内插值和计算三次样条所必须的全部信息。

如下一段程序采用三次样条进行插值，插值结果和原数据的对比如图 6-10 所示：

```
x = -5:5;
y = x.^2;
xi = -5:0.25:5;
yi = spline(x, y, xi);
plot(x, y, 'o', xi, yi)
```

下面计算三次样条的 `pp` 形式，命令语句及结果如下：

```
pp = spline(x, y)
pp =
    form: 'pp'
    breaks: [-5 -4 -3 -2 -1 0 1 2 3 4 5]
    coefs: [10x4 double]
    pieces: 10
    order: 4
    dim: 1
```

这里的三次样条 `pp` 形式，它以结构变量的形式存储了断点和多项式系数，以及关于三

次样条表示的其它信息。其中 `breaks` 是断点位置；`coefs` 是多项式系数矩阵，它的第  $i$  行是第  $i$  个多项式的系数；`pieces` 是多项式的数目；`order` 是每个多项式的阶数；`dim` 是插值的维数。

注意：pp 形式具有一般性，样条多项式不必是三次。

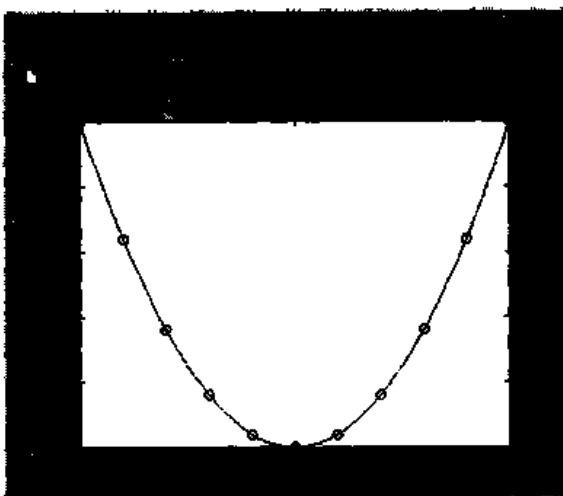


图 6-10 三次样条插值

得到 pp 形式后，可以用函数 `ppval` 计算该三次样条。该函数的用法如下：

`yi = ppval(pp, xi)`

对于相同的 pp 形式，指定不同的 `xi` 可以得到不同的三次样条插值。

当要计算三次样条表示时，必须把 pp 形式分解成它的各个部分。在 MATLAB 中，通过函数 `unmkpp` 完成这一功能。该函数的调用格式如下：

`[breaks, coefs, pieces, order, dim] = unmkpp(pp)`

输出参数的个数是 1 个到 5 个之间都可以，该函数会按式中的次序输出这些内容。输出参数缺省时只返回第一个参数，即断点位置。

反之，如果给定 pp 形式的各个部分，可以用函数 `mkpp` 重构完整的 pp 形式。

继续上面的例子，我们用如下的语句可以分解和重构 pp 形式：

```
[breaks, coefs]=unmkpp(pp) % 只返回断点位置 breaks 和多项式系数矩阵 coefs
breaks =
-5    -4    -3    -2    -1     0     1     2     3     4     5
coefs =
      0    1.0000   -10.0000   25.0000
      0    1.0000   -8.0000   16.0000
      0    1.0000   -6.0000    9.0000
      0    1.0000   -4.0000    4.0000
      0    1.0000   -2.0000    1.0000
-0.0000    1.0000    0.0000      0
  0.0000    1.0000   2.0000   1.0000
```

```

0    1.0000    4.0000    4.0000
0    1.0000    6.0000    9.0000
0    1.0000    8.0000   16.0000
pp1 = mkpp(breaks, coefs)      % 重构 pp 形式
pp1 =
form: 'pp'
breaks: [-5 -4 -3 -2 -1 0 1 2 3 4 5]
coefs: [10x4 double]
pieces: 10
order: 4
dim: 1

```

可以看出，只要断点位置向量 `breaks` 和多项式系数矩阵 `coefs` 两个参数就可以重构出完全相同的 `pp` 形式，这时因为矩阵 `coefs` 的大小确定了参数 `pieces` 和 `order`，所以 `mkpp` 不需要 `pieces` 和 `order` 去重构 `pp` 形式。

## 6.5 数据分析和傅立叶变换

MATLAB 中的基本数据分析函数位于目录`\toolbox\matlab\datafun` 下，本节我们来分类介绍部分函数的用法。

MATLAB 对这些函数有两条约定：

- (1) 若输入参数是向量，那么不管是行向量还是列向量，运算是对向量整体进行的。
- (2) 若输入参数是数组（包括矩阵），那么运算是按列进行的。

这两条约定不只适用于本节提到的函数，而且对 MATLAB 各工具箱的函数具有普遍意义。建议用户编程时尽量遵循这种约定。

### 6.5.1 基础运算

基础运算函数很多，用法简单，表 6-3 列出了这些函数的名称和功能。对这些函数的用法我们只举几个典型的例子加以说明，大部分函数请读者在使用时参看在线帮助。

表 6-3 矩阵函数

函数	功 能	函数	功 能
<code>convhull</code>	返回位于凸包上点的序号	<code>perms</code>	求向量按所有可能的变换组合出的矩阵
<code>cumprod</code>	计算累积积	<code>polyarea</code>	求多边形区域的面积
<code>cumsum</code>	计算累积和	<code>primes</code>	列出小于或等于某个数值的所有素数
<code>cumtrapz</code>	计算累积梯形数值积分	<code>prod</code>	求积
<code>delaunay</code>	德洛内（Delaunay）三角分解	<code>sort</code>	把数组的列元素按递增顺序排列
<code>dsearch</code>	搜索三角分解网格中最近的点	<code>sortrows</code>	把数组的元素以某一列为基准，按递增顺序排列，缺省是第一列
<code>factor</code>	计算素数因子	<code>std</code>	求标准差

(续)

函数	功能	函数	功能
inpolygon	检测指定点是否在多边形区域内	sum	求和
max	求最大元素	trapz	梯形数值积分
mean	求平均值	tsearch	在封闭的德洛内三角中搜索
median	找出中位元素	var	求方差
min	求最小元素	voronoi	沃龙诺依图

```
B = [3 8 4; 2 1 2; 4 6 1]
```

```
B =
```

```
3     8     4
2     1     2
4     6     1
```

```
mean(B)          % 计算按列进行，得到每一列的均值
```

```
ans =
```

```
3.0000    5.0000    2.3333
```

```
median(B)        % 找出每列的中位元素
```

```
ans =
```

```
3     6     2
```

```
std(B)           % 每列的标准差
```

```
ans =
```

```
1.0000    3.6056    1.5275
```

```
cumsum(B)        % 按列计算的累积和
```

```
ans =
```

```
3     8     4
5     9     6
9    15     7
```

```
sortrows(B, 2)    % 以第二列为基准，按递增顺序排列
```

```
ans =
```

```
2     1     2
4     6     1
3     8     4
```

```
x = [1 2 3];
```

```
perms(x)         % 把向量的所有可能变换组合为矩阵
```

```
ans =
```

```
3     2     1
2     3     1
3     1     2
1     3     2
```

```
2     1     3
1     2     3
```

### 6.5.2 有限差分

#### 1. 差分

函数 `diff` 计算向量或数组（按列）的有限差分，该函数的完整调用格式为：

```
Y = diff(X, n, dim)
```

其中 `X` 是向量或数组，`n` 是差分的阶数，`dim` 指定沿着数组的哪一维进行差分运算。其中 `n` 和 `dim` 可以省略。

```
A = [1 2 3 4];
diff(A)                                % 向量的一阶差分
ans =
1     1     1
B(:,:,1) = [1 2; 3 4];      % 定义三维数组 B
B(:,:,2) = [5 6; 7 8];
diff(B, 1, 2)                          % 沿数组的第二维做一阶差分
ans(:,:,1) =
1
1
ans(:,:,2) =
1
1
diff(B, 1, 3)                          % 沿数组的第三维做一阶差分
ans =
4     4
4     4
```

#### 2. 拉普拉斯微分算子

MATLAB 中的函数 `del2` 类似于离散拉普拉斯微分算子，其一般用法为：

```
L = del2(U)
```

其中 `U` 是数组，如果把数组 `U` 看成多元函数  $u(x, y, z, \dots)$ ，数组的维数就是函数中变量的个数，那么，函数 `del2` 相当于进行如下运算：

$$L = \frac{\nabla^2 u}{2N} = \frac{1}{2N} \left( \frac{d^2 u}{dx^2} + \frac{d^2 u}{dy^2} + \frac{d^2 u}{dz^2} + \dots \right)$$

式中的 `N` 就是数组的维数，也就是函数中的变量个数。

```
x =
```

```
1     0     1
```

```

0   1   1
1   1   0
def2(x)
ans =
1.0000  0.2500  0.2500
0.2500 -0.5000 -0.5000
0.2500 -0.5000 -0.5000

```

### 3. 数值梯度

不失一般性，函数  $F(x, y, z, \dots)$  的梯度定义为：

$$\nabla F = \frac{\partial F}{\partial x} \hat{i} + \frac{\partial F}{\partial y} \hat{j} + \frac{\partial F}{\partial z} \hat{k} + \dots$$

在 MATLAB 中用函数 gradient 来进行数值梯度运算，该函数的一般用法为：

$[Fx, Fy, Fz, \dots] = \text{gradient}(F)$

如果数组  $F$  是一维的（向量），那么只返回  $Fx$ ，即  $\frac{\partial F}{\partial x}$ ；如果是二维（矩阵），则返回

$Fx$  和  $Fy$ ，即  $\frac{\partial F}{\partial x}$  和  $\frac{\partial F}{\partial y}$ ，分别对应矩阵的列方向和行方向；依次类推。

**A =**

```

1   1   0
1   0   1
0   1   1

```

$[Fx, Fy] = \text{gradient}(A)$

$Fx =$

```

0   -0.5000  -1.0000
-1.0000      0    1.0000
1.0000    0.5000      0

```

$Fy =$

```

0   -1.0000  1.0000
-0.5000      0    0.5000
-1.0000    1.0000      0

```

## 6.5.3 向量运算

### 1. 向量的点积

函数 dot 计算向量的点积，用法如下：

$C = \text{dot}(A, B, \text{dim})$

如果  $A$  和  $B$  是向量，则返回  $A$  和  $B$  的标量积，要求  $A$  和  $B$  必需长度相同。如果  $A$  和  $B$  是多维数组，标量积沿着  $A$  和  $B$  第一个长度非 1 的维进行，要求  $A$  和  $B$  必需有相同的大小。参数 dim 是可选的，如果指定了该参数，则表示在第 dim 维上计算  $A$  和  $B$  的标量积。

## 2. 向量的叉积

函数 `cross` 计算向量的叉积，用法如下：

```
C = cross(A, B, dim)
```

如果 A 和 B 是向量，则返回 A 和 B 的叉积，要求 A 和 B 必需是长度为 3 的向量。如果 A 和 B 是多维数组，叉积沿着 A 和 B 第一个长度为 3 的维进行，要求 A 和 B 必需有相同的大小。参数 dim 是可选的，如果指定了该参数，则表示在第 dim 维上计算 A 和 B 的叉积，要求 A 和 B 必需有相同的大小，而且 `size(A, dim)` 和 `size(B, dim)` 必需等于 3。

```
a = [1 2 3]; b = [3 2 1];
c1 = dot(a, b)
c1 =
    10
c2 = cross(a, b)
c2 =
    -4      8     -4
```

### 6.5.4 协方差阵和相关阵

MATLAB 中计算协方差阵和相关阵的相应函数分别为 `cov` 和 `corrcoef`，下面分别介绍这两个函数的用法。

#### 1. cov 函数

1) `C = cov(x)`: 如果输入参数 x 是向量，那么返回值 C 是该向量的方差；如果 x 是矩阵，那么它的每一列相当于一个变量，返回值 C 就是该矩阵的列与列之间的协方差矩阵。这时，`diag(cov(x))` 是该矩阵每个列向量的方差。

2) `C = cov(x, y)`: 相当于 `cov([x y])`。

#### 2. corrcoef 函数

1) `S = corrcoef(X)`: 计算输入矩阵 X 的相关系数矩阵 S，X 的每一列相当于一个变量。对于相同的输入矩阵 X，相关系数矩阵 S 和协方差矩阵 C 之间有如下关系：

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

2) `S = corrcoef(x, y)`: 相当于 `corrcoef([x y])`。

```
A = perms([1 2 3]) % 由向量任意变换组成矩阵
```

```
A =
```

3	2	1
2	3	1
3	1	2
1	3	2
2	1	3

```

1   2   3
cov(A)          % 计算协方差阵
ans =
    0.8000  -0.4000  -0.4000
   -0.4000   0.8000  -0.4000
   -0.4000  -0.4000   0.8000
corrcoef(A)      % 计算相关系数矩阵
ans =
    1.0000  -0.5000  -0.5000
   -0.5000   1.0000  -0.5000
   -0.5000  -0.5000   1.0000

```

### 6.5.5 傅立叶变换初步

由于傅立叶变换有明确的物理意义，即变换域反映了信号包含的频率内容，因此傅立叶变换是信号处理中最基本也是最常用的变换。

MATLAB 提供了 fft (内置函数), ifft, fft2, ifft2, fftn, ifftn, fftshift, ifftshift 等一些函数来计算数据的离散快速傅立叶变换。在数据的长度是 2 的幂次时，采用基-2 算法进行计算，计算速度会显著增加，因此，只要可能，就应当尽量使数据长度为 2 的幂次或者用零来填补数据。

函数  $X = \text{fft}(x)$  和  $x = \text{ifft}(X)$  分别做数据的傅立叶变换和反傅立叶变换，如果  $x$  和  $X$  都是长度为  $N$  的向量，那么两个函数采用的公式分别为：

$$X(k) = \sum_{j=1}^N x(j) W_N^{(j-1)(k-1)} \quad k = 1, 2, 3, \dots, N$$

$$x(j) = \frac{1}{N} \sum_{k=1}^N X(k) W_N^{-(j-1)(k-1)} \quad j = 1, 2, 3, \dots, N$$

式中， $W_N = e^{-2\pi j/N}$

注意：MATLAB 中没有零下标，因而 MATLAB 采用的公式下标（从 1 开始）比一般工科教材上讲的公式下标（从 0 开始）后移一位。

下面我们分别介绍 MATLAB 中这些用于快速傅立叶变换的函数用法。

#### 1. fft 和 ifft

函数 fft 的调用格式有如下几种：

$Y = \text{fft}(X)$ ：如果  $X$  是向量，则采用快速傅立叶变换算法做  $X$  的离散傅立叶变换；如果是矩阵，则计算矩阵每一列的傅立叶变换；如果是多维数组，则对第一个非单元素的维进行计算。

$Y = \text{fft}(X, n)$ ：用参数  $n$  限制  $X$  的长度，如果  $X$  的长度小于  $n$ ，则用 0 补足，如果  $X$  的长度大于  $n$ ，则去掉长出的部分。

$Y = \text{fft}(X, [], \text{dim})$  或  $Y = \text{fft}(X, n, \text{dim})$ : 在参数 dim 指定的维上进行操作。

函数 ifft 的用法和 fft 完全相同。

我们首先产生一组数据 y, 它含有 50Hz 和 120Hz 的两个正弦信号和一些随机噪声, y 的图形如图 6-11 的上半部分, 从图中已经很难看出正弦波的成分。程序如下:

```
t = 0:0.001:0.6; % 采样周期为 0.001 秒, 即采样频率为 1000Hz
x = sin(2*pi*50*t)+sin(2*pi*120*t); % 产生正弦波
y = x + 2*randn(size(t)); % 叠加随机噪声
subplot(2, 1, 1) % 画出 y 的曲线
plot(y(1:50))
```

为了识别信号 y 中的正弦成分, 我们对 y 做傅立叶变换, 把时域信号变换到频域进行分析, 结果如图 6-11 的下半部分, 从中可以明显看出信号中 50Hz 和 120Hz 的两个频率分量。程序如下,

```
Y = fft(y, 512); % 对 y 做傅立叶变换, 取 512 个点
f = 1000*(0:256)/512; % 设置频率轴(横轴)坐标, 1000 是采样频率
subplot(2, 1, 2) % 绘制频域图形
plot(f, Y(1:257))
```

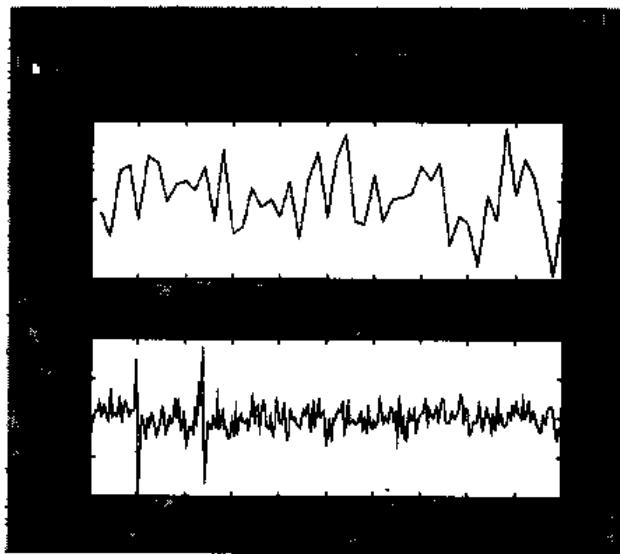


图 6-11 时域信号和频域信号比较

## 2. fft2 和 ifft2

函数 fft2 和 ifft2 对数据做二维快速傅立叶变换和反傅立叶变换。

数据的二维傅立叶变换  $\text{fft2}(X)$  相当于  $\text{fft}(\text{fft}(X))'$ , 即先对 X 的列做一维傅立叶变换, 然后对变换结果的行做一维傅立叶变换。

两个函数 fft2 和 ifft2 的调用格式完全相同, 以 fft2 为例:

$Y = \text{fft2}(X)$ : 二维快速傅立叶变换

$Y = \text{fft2}(X, m, n)$ : 通过截断或用 0 补足, 使得 X 成为  $m \times n$  的矩阵。

和 fft、ifft 类似, fftn、ifftn 对数据做多维快速傅立叶变换, 相关内容请参看帮助。

### 3. fftshift 和 ifftshift

函数 `fftshift(Y)` 用于把傅立叶变换结果 `Y` (频域数据) 中的直流分量 (频率为 0 处的值) 移到中间位置。

如果 `Y` 是向量, 则交换 `Y` 的左右半边; `Y` 是矩阵, 则交换其一三象限和二四象限; `Y` 是多维数组, 则在数组的每一维交换其“半空间”。

函数 `ifftshift` 相当于把 `fftshift` 函数的操作逆转, 用法相同。

```
X = rand(5,3);
Y = fft(X)
Y =
    1.8261      2.3754      3.1822
    0.5019 - 0.2915i  0.4312 + 0.1461i  -0.3732 - 0.4985i
    0.3587 - 0.0291i  0.0867 + 0.6039i  -0.2719 + 0.0337i
    0.3587 + 0.0291i  0.0867 - 0.6039i  -0.2719 - 0.0337i
    0.5019 + 0.2915i  0.4312 - 0.1461i  -0.3732 + 0.4985i

Ys = fftshift(Y)
Ys =
    -0.2719 - 0.0337i  0.3587 + 0.0291i  0.0867 - 0.6039i
    -0.3732 + 0.4985i  0.5019 + 0.2915i  0.4312 - 0.1461i
    3.1822            1.8261      2.3754
    -0.3732 - 0.4985i  0.5019 - 0.2915i  0.4312 + 0.1461i
    -0.2719 + 0.0337i  0.3587 - 0.0291i  0.0867 + 0.6039i
```

## 6.6 稀疏矩阵

对于一个用矩阵描写的线性方程组来说,  $n$  个未知数的问题就涉及到一个( $n \times n$ )的方程组, 存储这个方程组就需要  $n^2$  个字的内存和正比于  $n^3$  的计算时间。这样, 解上百阶、上千阶的方程就不那么容易处理。幸运的是, 大多数情况下, 一个未知数只与数量不多的其它变量有关, 即关系矩阵是稀疏的。

MATLAB 支持稀疏矩阵, 只存储矩阵的少量的非零运算, 这个特征在矩阵的存储空间和计算时间上都有很大的优点。本章将介绍在 MATLAB 中如何创建稀疏矩阵, 如何在特定的和一般的数学运算中使用它们。稀疏矩阵函数位于 MATLAB 的 toolbox 目录下的 `sparfun` 目录下。

稀疏矩阵及其算法, 就是不存储那些“0”元素, 也不对它们进行操作, 从而节省内存和计算时间。稀疏矩阵计算的复杂性和代价仅仅取决于稀疏矩阵的非零元素数目, 而与该稀疏矩阵的总元素数目无关。

### 6.6.1 稀疏矩阵的存储

在 MATLAB 中有两种存储矩阵的方式: 一种是全元素存储, 另一种是稀疏存储。对于

全元素存储，MATLAB 将存储矩阵的每一个元素，零值元素和其它矩阵元素一样需要相同的存储空间。然而，对于稀疏存储，MATLAB 仅存储那些非零元素及其下标。对于有较多零元素的大矩阵，这种存储方式将显著的减少所需的存储空间。

MATLAB 需要三个数组存储实型的稀疏矩阵。考虑一个  $m \times n$  的稀疏矩阵，它有  $nnz$  个非零元素：第一个数组采用浮点形式存储整个数组的非零元素，这个数组的长度为  $nnz$ ；第二个包含非零元素所对应的行标，它的长度也是  $nnz$ 。第三个数组包含指向每一列开始的指针，它的长度等于等于  $n$ 。这个矩阵的储存包含  $nnz$  个浮点数和  $nnz+n$  个整数。如果用 8 位表示一个浮点数，4 位表示一个整数，则存储这个稀疏矩阵所需的总的位数位： $8*nnz + 4*(nnz+n)$ 。复数矩阵也可以采用稀疏形式存储。在这种情况下，MATLAB 还需要第四个数组存储非零复数的虚部部分，它的大小也是  $nnz$ 。一个元素，只要它的实数部分或复数部分有一个不为零则它就是非零元素。

### 6.6.2 创建稀疏矩阵

全元素矩阵经过运算后仍然是全元素存储的。因此，假如不经过专门定义和运算，稀疏矩阵是不会自动产生的，稀疏矩阵的创建需要用户来决定。用户要判断在矩阵中是否有大量的零元素，是否需要采用稀疏存储技术。但是，一旦确定了某个矩阵以稀疏方式存储，那么它参与运算的结果也将仍以稀疏方式存储，除非运算本身使稀疏性消失。

矩阵的密度定义为矩阵中非零元素的个数除以矩阵中总的元素个数。对于密度很低的矩阵，采用稀疏形式存储是一种好的选择。

#### 1. 全元素存储转化为稀疏存储

使用 `sparse` 函数可以将一个全元素存储矩阵转化为稀疏存储。其调用格式为：

`S = sparse(A)`

假定矩阵不是太大，也可以将使用 `full` 函数将稀疏存储的矩阵转化为全元素存储。其命令为：

`A = full(S)`

```
A = [0 0 0 5
      0 2 0 0
      1 3 0 0
      0 0 4 0];
S = sparse(A)    %输出矩阵 S 的非零元素，以及对应的行和列
S =
(3,1)      1
(2,2)      2
(3,2)      3
(4,3)      4
(1,4)      5
```

元素按列存储，这反映出了内部的数据存储结构。

`A = full(S)` %将稀疏矩阵重新转化为全元素存储

```
A =
    0   0   0   5
    0   2   0   0
    1   3   0   0
    0   0   4   0
```

将全元素矩阵转化为稀疏矩阵并不是生成稀疏矩阵经常使用的方法。如果矩阵的阶数足够小，使得矩阵可以采用全元素存储，则将它转化为稀疏存储将不会节省太多的存储空间。

## 2. 直接创建稀疏矩阵

可以直接采用 `sparse` 函数利用非零元素列表创建稀疏矩阵。其调用格式为：

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

其中 `i` 和 `j` 分别是矩阵非零元素的行和列指标向量，`s` 是非零值向量，它的下标由对应的数对(`i, j`)确定。`m, n` 分别是矩阵的行维和列维。

因此，前面例子中的矩阵 `S` 可以直接利用下列命令生成：

```
S = sparse([3 2 3 4 1],[1 2 2 3 4],[1 2 3 4 5],4,4)
```

```
S =
```

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

`sparse` 命令还有其它的一些形式，上面的例子采用的形式，由 `length(s)` 给定了矩阵最大的非零元素个数。如果需要的话，可以再加上第六个参数，给定一个较大的数，它确定了矩阵非零元素的最大个数。以后，可以继续增加非零元素而不需要改变存储要求。

## 3. 从 MATLAB 外部输入稀疏矩阵

可以从 MATLAB 外部输入稀疏矩阵。将 `load` 命令和 `spconvert` 函数共同使用，可以输入包含一系列下标和非零元素的文本文件。`save` 命令和 `load` 命令也可以处理以二进制形式储存在 MAT-文件中的稀疏矩阵。一个矩阵在 MATLAB 内存空间中以稀疏形式存在，当用 `save` 命令存储成 MAT-文件以后，用 `load` 命令调用后仍然以稀疏形式存在。

考虑一个有三列的文本文件 `T.dat`，它的第一列是一些行下标，第二列是一些列下标，第三列是一些非零元素值。下面的命令将 `T.dat` 调入 MATLAB，然后将它转化为稀疏矩阵。

下面是文件“`T.dat`”的内容：

```
1  2  1.0
2  3  2.0
3  4  2.5
1  4  3.0
load T.dat
S = spconvert(T)      %生成稀疏矩阵
```

```

S =
(1,2)      1.0000
(2,3)      2.0000
(1,4)      3.0000
(3,4)      2.5000
save t S %将工作内存空间中的变量 S 存储在文件 t.mat 中。
clear %清除工作空间的变量
load t S %将 T.mat 中的变量读入工作空间
S %查看变量 S 是否仍然为稀疏矩阵
S =
(1,2)      1.0000
(2,3)      2.0000
(1,4)      3.0000
(3,4)      2.5000

```

#### 4. 稀疏带状矩阵创建命令

**SM=spdiags(B, d, m, n)**

其中, m、n 分别指定矩阵 SM 的行、列; d 是长度为 p 的整数向量, 它指定 SM 的对角线位置; B 是全元素阵 (但不必一定) 用来给定 SM 对角线位置上的元素, 其行维为  $\min(m,n)$ , 列维为 p。

MATLAB 还提供了其它的几种特殊稀疏矩阵的创建命令: 稀疏单位阵 (speye); 稀疏随机阵 (sprandn); 稀疏对称随机阵 (sprandsym)。

### 6.6.3 稀疏矩阵的查看

MATLAB 提供了一系列函数, 对稀疏矩阵进行定量的了解和得到关于它的图形信息。本节只介绍关于稀疏矩阵信息的定量的了解。

#### 1. 通常的存储信息

**whos** 命令提供了关于稀疏矩阵的大小、存储类型等较全面的信息。例如, 下面的 whos 命令给出了同一个矩阵采用全元素存储和稀疏存储的存储信息。

**whos**

Name	Size	Bytes	Class
A	3x4	96	double array
S	3x4	68	sparse array

Grand total is 16 elements using 164 bytes

注意: 由于只保存非零元素, 存储稀疏矩阵所需的字节数小于全元素存储, 尤其对于大型稀疏矩阵更是如此。

#### 2. 非零元素的信息

在 MATLAB 中有一些命令提供了关于稀疏矩阵非零元素的信息。

**nnz:** 返回稀疏矩阵非零元素的数目

**nonzeros:** 返回列向量, 包含稀疏矩阵中所有的非零元素

**nzmax:** 返回分配给稀疏矩阵中非零项的总的存储空间

以前面的矩阵 S 为例:

```
nnz(S) % 给出非零元素的个数
```

```
ans =
```

```
4
```

```
S
```

```
S =
```

(1,2)	1.0000
(2,3)	2.0000
(1,4)	3.0000
(3,4)	2.5000

```
nonzeros(S) % 列出矩阵 S 的非零元素
```

```
ans =
```

1.0000
2.0000
3.0000
2.5000

对于大型的矩阵，可以使用 Ctrl-C 停止 nonzeros 的输出。

- 注意：初始 nnz 隐含地和 nzmax 具有相同的值。即非零元素的数目等价于分配给非零元素的存储空间数目。然而，当其它的元素被给定零值时，MATLAB 并不会动态地释放内存。将矩阵的一些元素值改变成零只会改变 nnz 的值，而不会改变 nzmax 的值。然而，用户可以给矩阵增加任意多的非零元素，这并不受 nzmax 的初始值的限制。

### 3. find 函数和稀疏矩阵

对于一个矩阵，不论是全元素的还是稀疏的，find 函数将返回非零元素的下标和数值。

其调用格式为：

```
[i,j,s] = find(S)
```

find 返回值 i 为非零元素的行下标向量，j 为非零元素的列下标向量，非零元素的值保存在向量 s 中。

```
[i,j,s] = find(S)
```

```
i =
```

```
1  
2  
1  
3
```

```
j =
```

```
2
```

```
3
4
4
s =
    1.0000
    2.0000
    3.0000
    2.5000
[m,n] = size(S)
m =
    3
n =
    4
sparse 函数使用 find 函数的输出以及矩阵的大小，重新创建矩阵 S。
S = sparse(i, j, s, m, n)
S =
    (1,2)      1.0000
    (2,3)      2.0000
    (1,4)      3.0000
    (3,4)      2.5000
```

#### 6.6.4 稀疏矩阵的运算

大多数的 MATLAB 标准数学函数，对于稀疏矩阵的运算就象对全元素矩阵运算一样。另外，MATLAB 还提供了一些针对稀疏矩阵进行运算的函数。本节主要讨论以下几个方面：

- 标准数学运算
- 置换和重新排列
- 矩阵分解
- 线性方程组
- 特征值和奇异值

稀疏矩阵的计算复杂性与矩阵中非零元素的个数成正比，计算的复杂性也线性地依赖于矩阵行、列的大小，但是与矩阵总的元素个数无关。这种复杂性使得运算变得和复杂，例如稀疏线性方程组的求解，包括元素的排序和填充等因素。然而，通常的稀疏矩阵运算所需的时间正比于对于非零元素进行的算术操作的数目。这就是说，计算时间正比于浮点运算次数每秒。

##### 1. 标准数学运算

在运算中，稀疏矩阵的变化主要遵循下面几个原则：

- 1) 对于一个函数，输入参数是矩阵，而输出的参数是标量或向量，则输出结果采用全元素形式存储。例如，size 函数的输入可以是全元素矩阵也可以是稀疏矩阵，但它总是返回一个全元素的向量。

2) 函数输入参数是标量或向量, 输出的参数是矩阵型, 如 zeros, ones, rand 和 eye 等, 总是返回全元素形式。这对于避免引入意外的稀疏性是必需的。zeros(m,n)的稀疏形式是 sparse(m, n), rand 和 eye 的稀疏形式分别是 sprand 和 speye。函数 ones 没有稀疏形式。

3) 一元函数的输入是矩阵, 输出为矩阵或向量, 它保留操作数的存储特性。如果 S 是一个稀疏矩阵, 则 chol(S)也是一个稀疏矩阵, diag(S)是一个稀疏向量。按列进行运算的函数, 如 max 和 sum 也返回稀疏向量, 尽管这些向量可能整个都是非零的。sparse 和 full 函数对于这个规则是一个例外。

4) 二元运算符, 如果两个操作数都是稀疏的, 则产生的结果也是稀疏的, 如果两个操作数都是全元素的, 则产生的结果也是全元素的。对于混合的操作数, 除非运算保留稀疏性, 否则将给出全元素结果。如果 S 是稀疏的, F 是全元素的, 则 S+F, S\*F 和 F\ S 也是全元素的, 而 S.\*F 和 S&F 是稀疏的。在有些情况下, 尽管矩阵有很多的零元素, 结果仍然是稀疏形式的。

5) 矩阵采用 cat 函数或方括号连接对于混合算子将产生稀疏的结果。

6) 在赋值语句的右侧的子矩阵索引保留操作数的存储形式。对于 T = S(i, j), 如果 S 是一个稀疏矩阵, 而 i 和 j 是向量或标量, 将产生一个稀疏的结果。在赋值语句的左侧的子矩阵索引, 如在 T(i, j) = S 中, 将不会改变左侧矩阵的存储形式。

## 2. 置换和重新排列

稀疏矩阵 S 的行和列的置换可以用下列两种方式表示:

- 1) 置换矩阵 P 作用于 S 的行表示为 PS, 或者作用于 S 的列表示为 SP'。
- 2) 置换向量 p, 是一个包含 1:n 的置换的全元素向量, 作用于 S 的行表示为 S(p,:), 作用于 S 的列表示为 S(:,p)。

```
p = [1 3 4 2 5]; % 置换向量 p
I = eye(5,5);
```

```
P = I(p,:); % 将置换向量 p 作用于矩阵 I 得到置换后的矩阵 P
```

```
P =
```

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

```
e = ones(4,1);
```

```
S = diag(11:11:55) + diag(e,1) + diag(e,-1)
```

```
S =
```

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

现在可以用置换向量 p 和置换矩阵 P 进行一些置换运算。

**S(p, :)** %行置换

ans =

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

**S(:, p)** %列置换

ans =

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

**P\*S** %行置换

ans =

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

**S'\*P'** %列置换

ans =

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

如果 P 是一个稀疏矩阵，则两种表示方式的存储量与 n 成正比，将二者应用于 S 所用时间与 nnz(S)成正比。两种表示方式中向量表示法更简洁和有效，因此各种稀疏矩阵的置换了 LU 分解中的主元素置换返回一个与早期的 MATLAB 版本兼容的矩阵外，将都返回整个行向量。

为了将两种表达方式进行转换，首先考虑一个稀疏单位阵 I = speye(n)，则：

P = I(p,:)

P' = I(:,p)

p = (1:n)\*P'

p = (P\*(1:n))'

通过上面的几个命令可以将两种表达方式进行转换。

将矩阵的列重新排列经常使得矩阵的 Cholesky 分解, LU 和 QR 分解矩阵变得更加稀疏。最简单的重新排列方式是重新按列检索非零元素数目。这对于结构非常无规则的矩阵来说是一个很好的重新排序方法。尤其当矩阵的行、列的非零元素数目有很大的变化。

### 3. 矩阵分解

本节将讨论四种重要的稀疏矩阵分解技术: LU 或三角分解, Cholesky 分解, QR 或正交分解, 不完全分解。

#### 1) LU 分解

如果 S 稀疏矩阵, 下面的命令将返回三个稀疏矩阵 L, U 和 P, 使得  $PS = LU$ .

$[L, U, P] = lu(S)$

lu 函数通过部分主元素高斯消去法获得结果。置换矩阵 P 只有 n 个非零元素。对于全元素矩阵, 下面的命令  $[L, U] = lu(S)$  返回一个单位下三角矩阵 L 和一个上三角阵 U, 满足  $S = L * U$ 。本身, lu(S) 返回矩阵 L 和 U 并不包含主元素的信息。

事实上, 下面的稀疏分解和全元素分解给出相同的 L 和 U, 尽管计算时间和存储要求有很大的不同。

$[L, U] = lu(S)$  %稀疏分解

$[L, U] = sparse(lu(full(S)))$  %全元素分解

在分解过程中, MATLAB 将自动地给稀疏矩阵 L 和 U 分配所需内存。MATLAB 并不预先使用符号 LU 预分解来决定所需的内存和建立数据结构。

#### 2) Cholesky 分解

如果 S 是对称正定稀疏矩阵, 则下面的命令将返回一个稀疏的上三角矩阵 R, 使得  $R^*R = S$ :

$R = chol(S)$

由于 Cholesky 算法对于稀疏性和数值稳定性并不需要选主元, 在进行分解的时候可以快速计算出所需的内存数量并分配所有的内存。

#### 3) QR 分解

MATLAB 将对稀疏矩阵 S 进行完全的 QR 分解:

$[Q, R] = qr(S)$

但是, 这经常是不现实的。正交矩阵 Q 经常不可能有太多的零元素。更现实的方法, 有时被称作“较少 Q 的 QR 分解”是可行的。

采用一个稀疏矩阵输入和一个输出参数:

$R = qr(S)$

仅返回 QR 分解的上三角部分。矩阵 R 为与正态方程关联的矩阵提供了 Cholesky 分解:

$R^*R = S^*S$

然而, 这样做避免了计算  $S^*S$  时, 内在的数值信息的丢失。

当输入的两个参数有相同的行数, 并给出两个输出参数时, qr 分解的命令为:

$[C, R] = qr(S, B)$

对 B 进行正交变换, 得到  $C = Q^*B$  而没有计算 Q。

“较少 Q 的 QR 分解”允许分两步求解稀疏最小二乘问题: 使  $|Ax - b|$  最小化。

$[c, R] = qr(A, b)$

$x = R \backslash c$

如果  $A$  是个稀疏矩阵，但不是方阵，则 MATLAB 使用下面的步骤求解线性方程： $x = A \backslash b$  或者，用户也可以自己做矩阵分解，并且验证  $R$  是有缺陷的。

也可以求解一系列的具有不同的右端项  $b$  的最小二乘线性系统  $L$ ，因为当计算  $R = qr(A)$  时并不需要知道  $b$ 。这个方法可以求解下列方程：

$R^T R^* x = A^T b$

首先：

$x = R \backslash (R^T \backslash (A^T b))$

然后，使用一步迭代修正降低迭代误差。

$r = b - A^T x$

$e = R \backslash (R^T \backslash (A^T r))$

$x = x + e$

#### (4) 不完全分解

`luinc` 和 `cholinc` 函数提供了近似的不完全分解，它作为稀疏矩阵的迭代方法的预条件是很有用的。

如果  $A$  是个稀疏矩阵， $tol$  是一个小的公差，则：

$[L, U] = luinc(A, tol)$

计算一个近似的 LU 分解，其中所有的元素小于  $tol$  乘以矩阵  $A$  中相应的列设为零时的矩阵范数。

$[L, U] = cholinc(A, '0')$

计算近似的 LU 分解，其中  $L+U$  的稀疏形式是  $A$  的稀疏形式的置换。

`luinc` 和 `cholinc` 函数还有其它的一些选项，参见联机帮助。

### 4. 求解线性方程组

线性方程组可以有两种不同的求解方法：直接法和迭代法。

直接法通常都是高斯消去法的各种变体，经常表现为矩阵分解，如 LU 分解或 Cholesky 分解。迭代法，经过有限的迭代步后得到方程的近似解。系数矩阵被间接地包括，通过矩阵和向量的乘或作为抽象线性算子的结果。

#### (1) 直接法

直接法是一种快速的方法，如果有足够多的存储空间可用，经常被使用。迭代法经常应用于比较严格的方程情况，依赖于类似主对角占优的矩阵情况。

直接解法通过 MATLAB 的核心实现，对于一般的矩阵尽可能的有效。迭代法通常通过 MATLAB 的 M-文件来实现，使用子问题或预条件的直接求解。

通常并不是通过 `lu` 函数或 `chol` 函数访问 MATLAB 中的直接方法，而是通过矩阵的除法算子  $/$  和  $\backslash$ 。如果  $A$  是一个方阵，则  $X = A \backslash B$  的结果就是线性方程  $A^T X = B$  的解。如果  $A$  不是方阵，则给出最小二乘解。

如果  $A$  是方阵，全元素或稀疏矩阵，则  $A \backslash B$  与  $B$  具有相同的存储类别。它的计算包含了各种算法的选取。在此不多介绍。

不管是采用什么样的算法，都是通过矩阵的除法算子求解线性方程组。

## (2) 迭代法

在 MATLAB 中，有六个函数可用于实现稀疏线性方程组的迭代解法。所有的六个方法都设计求解  $Ax = b$ 。预条件共轭梯度法 pcg 只适用于对称正定矩阵 A。其它的五个可以处理非对称方阵。表 6-4 给出了六个函数。

表 6-4 迭代函数表

函数	描述
bicg	双共轭梯度法
bicgstab	稳定化双共轭梯度法
cgs	平方共轭梯度法
gmres	广义最小残差法
pcg	预条件共轭梯度法
qmr	准最小残差法

所有的六种方法都可以使用左右预条件。

# 第 7 章 编程进阶

## 7.1 句柄图形

句柄图形是一种面向对象（Object-Oriented）的图形系统概念，它是建立计算机图形的必要成分。

例如 `figure` 函数，如果用变量来获取该函数的返回值，即采用如下形式的命令：

```
h1 = figure  
h2 = figure
```

那么在绘图的同时两个 `figure` 函数会返回不同的值给 `h1`、`h2`，这里的 `h1`、`h2` 就是相应图形窗口的句柄，每个图形窗口的句柄各不相同，因而可以通过句柄来确定图形窗口并对其属性进行设置。

### 7.1.1 句柄图形的结构层次

句柄图形基于这样的概念：一幅图的每一组成部分是一个对象，每一个对象有一系列句柄和它相关，每一个对象有可以设置和改变的属性。

由图形命令产生的每一部分都是图形对象。它们包括图形窗口或仅仅说是图形，还有坐标轴、线条、文本等组成部分，以及其它交互式设备，如菜单、界面等。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其它对象的父对象。图形窗口是根对象的子对象；坐标轴和用户界面等对象是图形窗口的子对象；线条、文本、表面和图象等对象是坐标轴对象的子对象。这种层次关系在图 7-1 中给出。

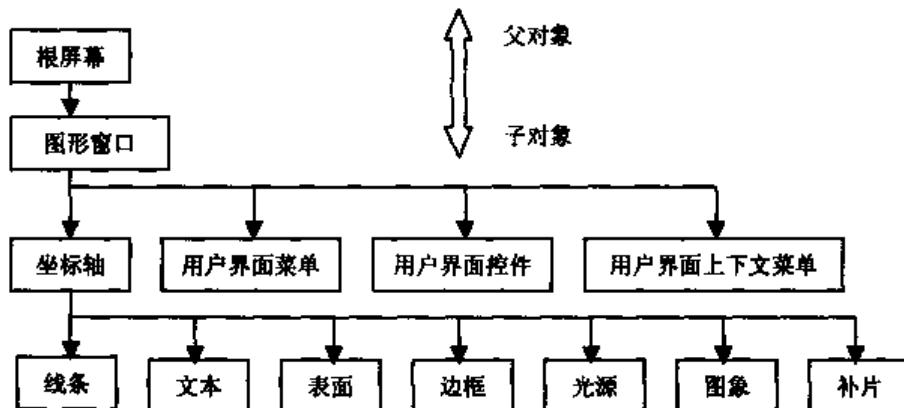


图 7-1 各种图形对象层次结构图

每个父对象中可以包含一个或多个子对象，比如，根屏幕可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有创建对象的函数当父对象不存在时，都会首先创建它们。例如，如果没有图形窗口，“`plot(x, y)`” 函数会用缺省属性创建一个新的图

形窗口和一组坐标轴，然后在这组坐标轴内画线。

对于图 7-1 中的各种图形对象，MATLAB 都提供了相应的创建函数，这些函数如表 7-1 所示。

表 7-1 建立图形对象的函数

函数（图形对象）	建立的图形对象
axes（坐标轴）	直角坐标系的坐标轴
figure（图形窗口）	显示图形的窗口
Image（图象）	由颜色映象（colormap）索引值或 RGB 值定义的二维图画。其数据格式可以是 8 位、16 位整型或双精度型
Light（光源）	在坐标轴内光源的方向
Line（线条）	依照指定的顺序，把坐标数据用直线段连接成的线条
Patch（补片）	按指定的方式填充的多边形
Rectangle（矩形）	图形的矩形、圆角矩形或椭圆形边框，注意还包括其中的填充
Surface（表面）	由小的矩形面组成的图形表面
Text（文本）	位于图形窗口的文本字符串
Uicontextmenu（用户界面上下文菜单）	可以与其它图形对象联系的上下文菜单
Uicontrol（用户界面控件）	可编程的用户界面设备，如按钮、列表框、对话框等
Uimenu（用户界面菜单）	图形窗口上部的可编程菜单

如何应用这些函数建立和修改图形对象，我们将在后面作介绍。

### 7.1.2 访问对象句柄

句柄实际上就是分配给每个对象的数字标识。每次创建一个对象时，就为它建立一个唯一的句柄。

计算机屏幕作为根对象，它的句柄一般是 0。图形窗口的句柄为整数，通常显示在图形窗口标题条中的“Figure No.”之后的数值就是该图形窗口的句柄。其它对象句柄是 MATLAB 满精度的浮点数。

#### 1. 获取图形对象的句柄

对于有些图形对象，MATLAB 可以用相应的函数获取它们的句柄。这类函数如表 7-2 所示。

表 7-2 获取图形对象句柄的函数

函数	功能
gcf	获取当前图形窗口的句柄
gca	获取当前坐标轴的句柄
gco	获取当前对象的句柄
gcb	获取当前正在执行调用的对象的句柄
gcbf	获取包括正在执行调用的对象的图形的句柄

直接调用这些函数就会返回对象的句柄值，如在 MATLAB 命令窗口中键入：

```
gcf
```

则 MATLAB 命令窗口会用固定变量 ans 显示返回的句柄值，如下所示：

```
ans =
```

```
1
```

若用如下的调用方式：

```
h=gcf
```

则 MATLAB 就会把返回的句柄值赋给变量 h，如下所示：

```
h =
```

```
1
```

这些函数还可以作为参数被其它函数调用，如 set、get 等对图形句柄操作的函数，格式一般为：

```
set(gcf, 'propertynname', propertynvalue)
```

```
propertynvalue = get(gcf, 'propertynname')
```

关于函数 set 和 get 的具体用法在下一小节介绍。

## 2. 查找对象

用函数 findobj 可以快速遍历对象层并获取指定了属性值的对象句柄。该函数有如下几种调用方式：

1) h = findobj('propertynname', propertynvalue,...): 在所有的对象层中查找符合指定属性值的对象，返回起句柄给变量 h。

2) h = findobj(ObjectHandle, 'propertynname', propertynvalue,...): 把查找的范围限制在句柄“ObjectHandle”指定的对象及其子对象中。

3) h = findobj(ObjectHandles, 'flat', 'propertynname', propertynvalue,...): 把查找的范围限制在句柄“ObjectHandle”指定的对象中，但不包括及其子对象。

4) h = findobj: 返回根对象和所有子对象的句柄。

5) h = findobj(ObjectHandles): 返回“ObjectHandle”指定的对象和其所有子对象的句柄。

我们举例说明一下此函数的用法。先用如下命令建立一个图形对象：

```
t=0:0.1:2*pi;
y=sin(t);
plot(t,y)
xlabel('Time (0~2\pi)', 'FontWeight', 'bold')
text(pi,0,'\leftarrow sin(\pi)=0')
```

图形对象中包括坐标轴、线条、文本和标注，如图 7-2 所示。

现在如果运行命令：

```
h=findobj(gcf)
```

那么 MATLAB 命令窗口显示的返回值为图形对象及其所有子对象的句柄：

```
h =
```

```
1.0000
```

```
73.0009
115.0005
1.0017
```

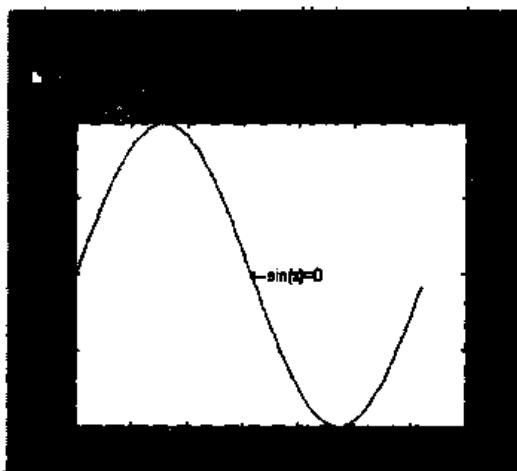


图 7-2 图形对象

其中  $h(1)=1$  为图形对象的句柄， $h(2)=73.0009$  为图形的下一级子对象——坐标轴的句柄， $h(3)=115.0005$  为坐标轴的下一级子对象——线条的句柄， $h(4)=1.0017$  为坐标轴的下一级子对象——文本的句柄。可见句柄中的元素排列顺序是由各个对象在整个对象层次中的位置决定的。

如果使用命令：

```
h=findobj('String','\leftarrow sin(pi)=0')
```

那么返回值将是：

```
h =
```

```
1.0017
```

即为文本对象的句柄，因为只有文本对象的属性“String”的值为“\leftarrow sin(pi)=0”。

我们注意到句柄中不包括用函数 xlabel 为 x 轴添加的标注，这是由于坐标轴的标注不在图形对象的几个层次中，用函数 findobj 无法遍历到。当然，坐标轴的标注也是一种对象，也能通过句柄对其进行操作。例如，我们在用函数 xlabel 为 x 轴添加标注时使用如下调用格式：

```
h1=xlabel('Time (0~2*pi)','FontWeight','bold')
```

这样就会在标注的同时得到其句柄“h1”，然后使用 get 或 set 命令（见下一节）都可以对该句柄进行操作。

### 3. 复制对象

用函数 copyobj 可以把对象从一个父对象中复制到另一个父对象中。新的对象和原对象只是句柄和属性“Parent”的值不同，其它属性均不变。

用函数 copyobj 可以把几个对象复制到同一个新的父对象，也可以把一个对象同时复制到几个不同的父对象，但要注意必须保持正确的“父子关系”。

如果复制的对象中包含子对象，那么 MATLAB 将把其所有的子对象一并复制。

函数 copyobj 有以下几种调用方式：

1)  $lC = \text{copyobj}(H, P)$ : 参数 H 和 “P” 都是向量，向量的元素为对象的句柄，向量 H 中的每一个句柄对应的图形对象都将被复制到向量 “P” 中的相应句柄对应的图形对象之下，分别成为这些父对象的子对象。而这些新的对象的句柄将会赋给向量 C 中的相应元素。注意 H 和 “P” 长度必须相等，而且其中的每个对应元素 “H (i)” 和 “P (i)” 必须具有合法的“父子关系”。

2)  $C = \text{copyobj}(H, p)$ : 参数 H 是向量，参数 p 是标量。向量 H 中的每一个句柄对应的图形对象都将被复制到句柄 p 对应的图形对象之下，同时成为这个父对象的子对象。而这些新的子对象的句柄将会赋给向量 C 中的相应元素。

3)  $C = \text{copyobj}(h, P)$ : 参数 h 是标量，参数 “P” 是向量。句柄 h 对应的图形对象将被复制到向量 “P” 中的每个句柄对应的图形对象之下，分别成为这些父对象的子对象。而这些新的子对象的句柄将会赋给向量 C 中的相应元素。

我们结合函数 `findobj` 来具体说明一下函数 `copyobj` 的用法。

结合前面的例子，首先生成如前图 7-2 所示的图形对象，然后用如下命令另外再建立一个图形对象：

```
figure
t=0:0.1:4*pi;
y=sin(t);
plot(t,y)
```

使用 `figure` 命令是为了另外建立一个图形对象，如果不使用 `figure` 命令，后面 `plot` 命令绘制的线条就会覆盖前一个图形对象中的线条。绘制的正弦曲线如图 7-3 所示，它只是比前面的曲线多了一个周期。我们看到该图形的标题栏显示了此图形对象的句柄为 “2”。

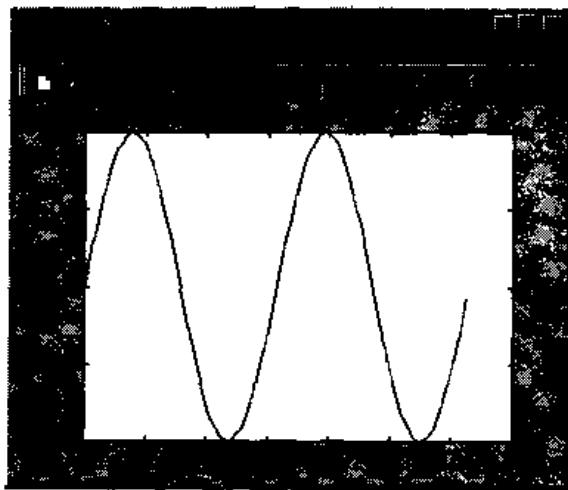


图 7-3 应用 `copyobj` 命令前的图形对象

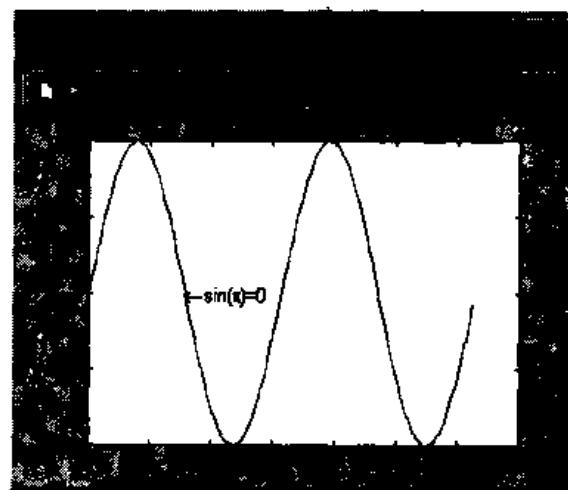


图 7-4 应用 `copyobj` 命令后的图形对象

下面我们要把图 7-2 中的文本对象复制到图 7-3 中来，使用如下命令：

```
h1=copyobj(h, gca)
```

结果如图 7-4 所示。命令中的句柄 h 是在前面用命令 “`h = findobj ('String', 'leftarrowsin(pi)=0')`” 得到的文本对象的句柄，而 `gca` 用来在目的图形对象中获取文本对象的父对象——坐标轴的句柄。

注意：必须保证要获取句柄的对象或其父对象为当前对象。

#### 4. 删除对象

删除对象由函数 `delete` 完成。调用格式为：

`delete(h)`

该命令将删除句柄所指的对象及其所有子对象，而且不提示确认，所以使用时要小心。

### 7.1.3 图形对象的属性和设置

所有的图形对象都有属性，正是通过设定这些属性来定义或修改图形的特征。每个不同的对象都有和它相关的属性，可以改变这些属性而不影响同类型或不同类型的其它对象。

对象属性包括属性名和与相应的值，属性名是字符串。而且，只要用足够多的字符来唯一地标识一个属性名即可，例如图形对象中的文件名属性一般用“`FileName`”表示，也可以用“`filename`”，甚至是“`file`”来表示。

注意：MATLAB 识别图形对象的属性时是不区分大小写的。

附录 B 给出了 MATLAB5.3 版本中各种图形对象的属性及其取值，供读者参考。

本节我们主要介绍如何获取、设置和修改这些属性，这就要用到对对象进行操作时非常常用的两个函数：`get` 和 `set`。

#### 1. get 函数

`get` 函数用于获取指定对象的属性值。它有如下几种用法：

1) `PropertyValue = get(H,'PropertyName')`: 获取句柄为 H 的对象中名为 `PropertyName` 的属性的值。例如命令“`V=get(gca, 'Position')`”用来返回当前图形对象中坐标轴左下角和右上角的位置坐标。如果 H 是向量，那么函数 `get` 将同时返回向量 H 中每个句柄对应的图形对象指定属性的值。

2) `get(h)`: 显示句柄为 h 的对象的所有属性名及其当前的取值。这里 h 只能为标量。

3) `PropertyValue = get(h)`: 返回一个结构，结构的每个域名就是句柄为 h 的对象的所有属性名，每个域又包括属性的值。这里的 h 为标量。

3) `PropertyValue = get(0, 'Factory<ObjectType><PropertyName>')`: 对于所有类型的对象，返回其所有可以由用户设置缺省值的属性的“出厂值”，所谓“出厂值”指的是未经过任何用户改动的最初的缺省值。这里的参数 `ObjectType` 和 `PropertyName` 是可选的。

4) `PropertyValue = get(h, 'Default<ObjectType><PropertyName>')`: 返回缺省的属性值。这里的句柄 h 必须是标量。这里的参数 `ObjectType` 和 `PropertyName` 是可选的。如果两个参数都不选，那么该函数返回句柄为 h 的对象所有属性的缺省值。缺省值不能从对象的后代或对象本身来查询，而只能从对象的前辈查询。

#### 2. set 函数

`set` 函数用来设置对象的属性值。它的调用方法有：

1) `set(H,'PropertyName', PropertyValue)`: 把句柄为 H 的对象中名为 `PropertyName` 的属性的值设置为“`PropertyValue`”。句柄 H 可以是向量，这种情况下，函数 `set` 为所有对象设置属性值。

2) `set(h,a)`: 这里的 a 为结构，其域名就是对象的属性名，属性值包括在域中，`set` 函数

数把属性值赋给和域名相同的属性。句柄 h 为标量。

3) set(H,PN,PV): 式中 PN 是  $1 \times N$  维数组, 其中的元素为需要设置的属性名, PV 中的元素是要设置的属性值, 此式把在句柄中指定的所有对象的属性(在数组 PN 中指定)设置为“PV”中的指定值。注意: 如果向量 H 长度是 M, 那么 PV 的维数应该是  $M \times N$ 。

4) set(H,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2,...): 同时设置多个属性值。

5) A = set(H, 'PropertyName') 或 set(H,PropertyName): 返回或显示句柄为 H 的对象的指定属性的值。

6) A = set(H) 或 set(H): 返回或显示句柄为 H 的对象的所有属性和可能的取值。

7) set(h, 'DefaultObjectTypePropertyName', PropertyValue): 设置对象属性的缺省值。式中的 ObjectType 和PropertyName 分别要代入相应的对象类型和属性名。注意: 对象属性的缺省值只能从前辈对象中设置。

我们来举例说明这两个函数的用法。首先用如下命令在同一个窗口中绘制一条正弦曲线、一条余弦曲线, 为 x 轴加标注, 并添加两个文本类型的对象作为曲线的标注(如图 7-5 所示)。注意这里的 hold 函数的用法: “hold on”保持住当前的图形和坐标轴属性, 使得后面的绘图命令在当前的图形基础上增添新的图形, 不改变当前图形的状态; “hold off”是缺省状态, 后面的绘图命令会重新设置图形的属性, 并覆盖掉前面的图形。

```
t=0:0.1:2*pi;
y1=sin(t);
h_line1=plot(t,y1,:')
hold on
y2=cos(t);
h_line2=plot(t,y2,'*')
h_label=xlabel('Time (0~2\pi)', 'FontWeight', 'bold')
h_text1=text(pi,0,'\\leftarrow sin wave')
h_text2=text(pi/2,0,'\\leftarrow cos wave')
hold off
```

在这段程序中我们为线条、文本、坐标轴标注都分配了相应的句柄, 这些句柄的值如下所示:

```
h_line1 =
    1.0024
h_line2 =
    74.0052
h_label =
    75.0056
h_text1 =
    76.0028
h_text2 =
    77.0033
```

用函数 set 和 get 对这些句柄进行操作就能够实现对相应用对象属性的修改。

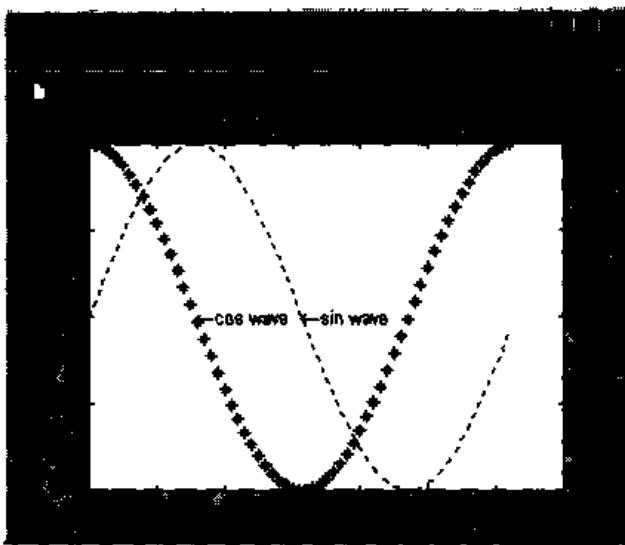


图 7-5 在同一窗口绘制正弦、余弦两条曲线

下面列举了一些命令及其作用，命令中用到的属性请参见附录 B 中的表格。

如下命令用来获取坐标轴的刻度位置：

**h\_XTick = get(gca, 'XTick')**

其返回值为：

**h\_XTick =**

0      1      2      3      4      5      6      7

如下命令用来获取坐标轴标注的字体大小：

**h\_text1\_FontSize = get(h\_label, 'FontSize')**

其返回值为：

**h\_text1\_FontSize =**

10

如下命令用来获取线条对象“Line”的线型属性“LineStyle”值，注意这里的句柄是由函数“gca”得到的坐标轴对象句柄，它是线条对象的父对象，不能在这里使用线条本身的句柄“h\_line1”或“h\_line2”。

**h\_line\_default = get(gca, 'DefaultLineStyle')**

其返回值为：

**h\_line\_default =**

如下命令用来设置坐标轴的刻度位置和刻度的标记，结果见图 7-6：

**set(gca, 'XTick', 0:pi/2:2\*pi)**

**set(gca, 'XTickLabel', {'0' 'pi/2' 'pi' '3pi/2' '2pi'})**

如下命令用来设置坐标轴标注和文本对象的字体形状和大小，设置效果如图 7-6 所示：

**set(h\_label, 'FontAngle', 'italic')**

**set([h\_text1 h\_text2], 'FontAngle', 'italic', 'FontSize', 12)**

注意第二条命令的句柄为向量，这样就同时设置了向量中的每个句柄对应的对象。

如下命令设置线条的缺省线型为“\*”型：

```
set(gca, 'DefaultLineLineStyle', '*')
```

在执行这条命令后，其它的绘图命令绘制的曲线缺省情况下都由“\*”表示。

用函数 `set` 和 `get` 来设置对象的属性非常灵活方便，而且使用简单，功能强大，因而这两个函数非常常用，大家可以在应用中仔细体会它们的使用技巧。

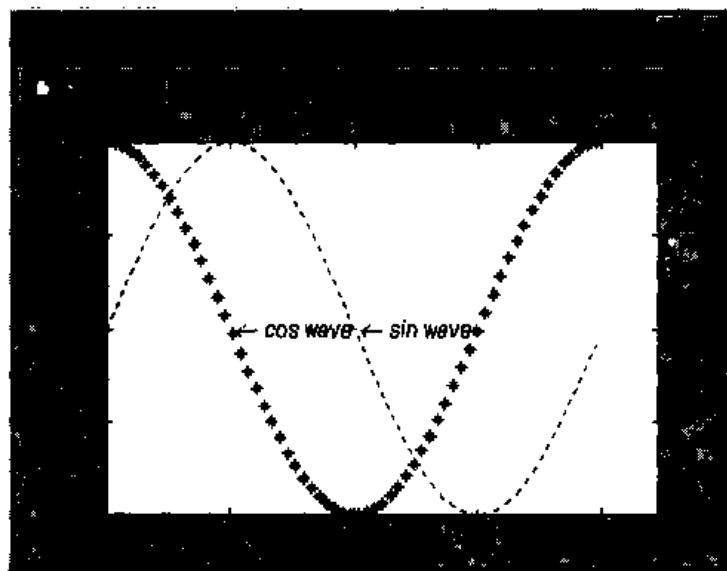


图 7-6 用 `set` 命令设置后的图形

## 7.2 图形对象属性编辑器

### 7.2.1 图形窗口的交互操作方式

我们曾经简单介绍过 MATLAB5.3 在图形窗口中新增的交互操作方式：按下按钮■就允许对窗口中的图形对象进行交互式编辑。本节我们将举例说明如何进行这种交互操作。

首先绘制一个简单的二维图形，同时加上文本对象做标注。程序如下：

```
t=0:0.1:2*pi;
y1=sin(t);
plot(t,y1,:')
hold on
y2=cos(t);
plot(t,y2,'*')
xlabel('Time (0~2\pi)', 'FontWeight', 'bold')
text(pi,0,'leftarrow sin wave')
text(pi/2,0,'leftarrow cos wave')
hold off
```

结果如图 7-7 所示。下面我们按下其中的■按钮便可以分别对图形窗口中的坐标轴、

线条和文本对象进行交互式编辑。

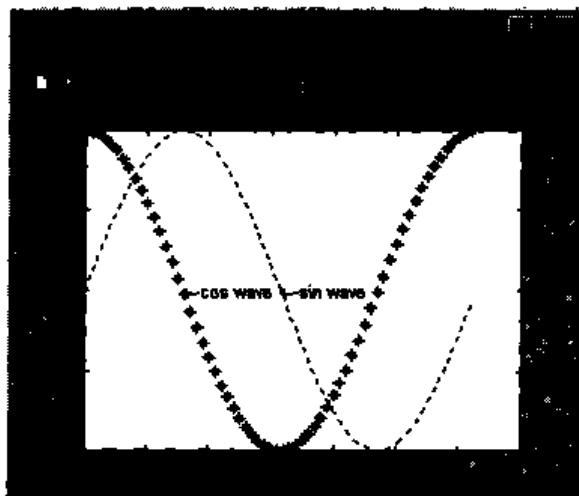


图 7-7 包含坐标轴、线条和文本对象的图形窗口

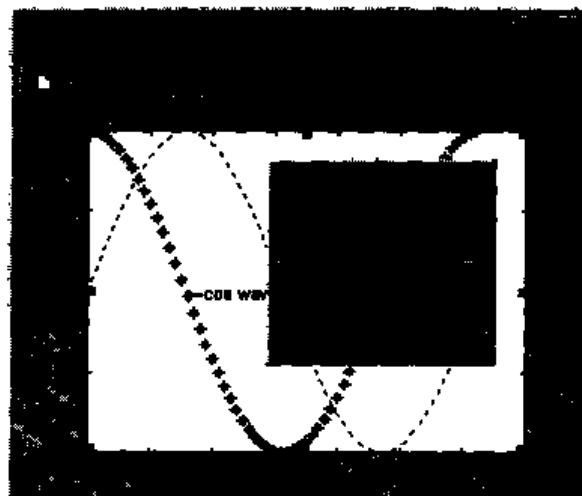


图 7-8 成为编辑状态的坐标轴及弹出的菜单

#### 1. 编辑坐标轴属性

在坐标轴范围内的空白区域或坐标轴的边框处单击鼠标右键，坐标轴在四个角和四条边的中央会各出现一个黑色的小方块，表示坐标轴已被选中，同时弹出一个菜单（这种菜单称为上下文菜单（Contextmenu）），此时的状态如图 7-8 所示。

弹出菜单中的 Show Legend 选项用于显示缺省的图例。Unlock Axes Position 用于解锁坐标轴的位置，选中这一项后，可以用鼠标拖动坐标轴来改变其位置，缺省情况下坐标轴的位置是锁定的，不能直接移动。

选择菜单中的 Properties 选项，将弹出如图 7-9 所示的编辑坐标轴属性对话框。这个对话框也可以通过用鼠标左键双击坐标轴范围内的空白区域或坐标轴的边框得到。还可以先选中要编辑的坐标轴，然后选择菜单 Tools 下的 Axis Properties 选项。

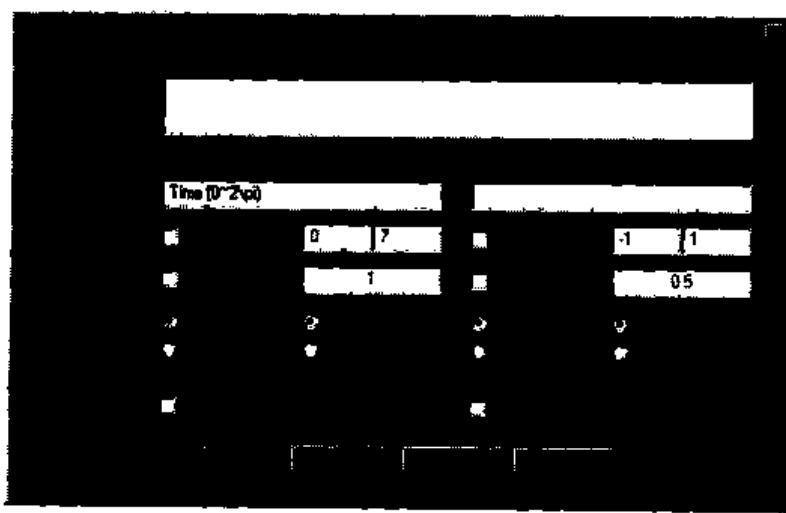


图 7-9 编辑坐标轴属性对话框

在这个对话框中的各个区域分别可以编辑坐标轴的如下几个属性：

- Title: 标题

- Label: 坐标轴的标注
- Limits: 坐标轴的刻度范围
- Tick Step: 坐标轴的刻度线间隔
- Scale: 坐标类型（线性、对数）和坐标轴的方向
- Grid: 网格线

在这里设置坐标轴属性和使用相关命令进行设置的效果是相同的。

## 2. 编辑线条属性

在需要编辑的线条上单击鼠标右键，线条的所有数据点都会成为一个黑色的小方块，表示线条已被选中，同时弹出一个菜单，此时的状态如图 7-10 所示。下一步选择菜单中的 Properties 选项，将弹出如图 7-11 所示的编辑线条属性对话框。这个对话框也可以通过用鼠标左键双击该线条得到。还可以先选中要编辑的线条，然后选择菜单 Tools 下的 Line Properties 选项。

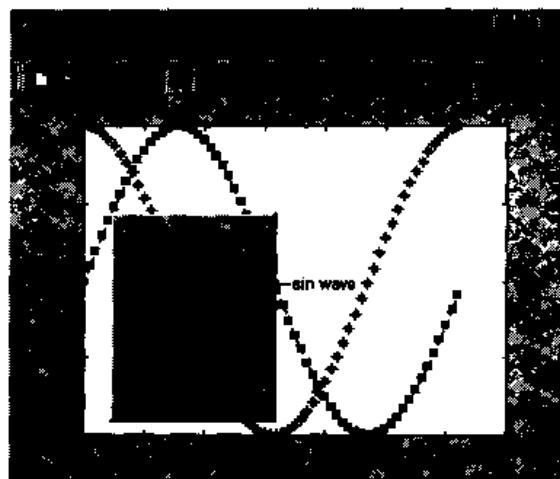


图 7-10 选中线条的状态及弹出式菜单

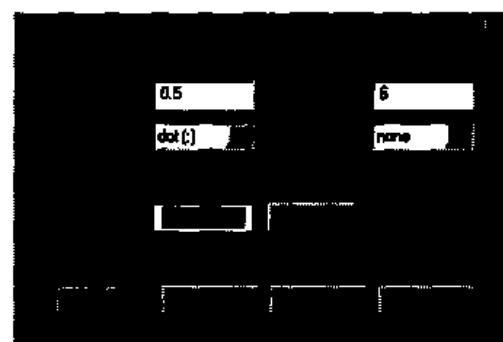


图 7-11 编辑线条属性对话框

从图 7-11 中可以看到在此对话框的区域中能够编辑的线条属性有如下几种：

- Line Width: 线条宽度
- Line Style: 线型
- Marker: 标记符号
- Marker Size: 标记符号的大小
- Color: 颜色

在选择了 Color 后的 Select 按钮时，会弹出一个通用的颜色设置对话框，里面有许多种供选择的颜色，还可以自己定义颜色，只要按下其中的“规定自定义颜色”按钮就会在原对话框右侧出现一个自定义颜色的面板，在这里可以选择各种喜欢的颜色。如图 7-12 所示。

## 3. 编辑文本属性

在需要编辑的文本上单击鼠标右键，文本框的四个角和四条边中央会各出现一个黑色的小方块，表示文本已被选中，同时弹出一个菜单，此时的状态如图 7-13 所示。下一步选择菜单中的 Properties 选项，将弹出如图 7-14 所示的文本的字体属性编辑对话框。这等效于先选中要编辑的文本，然后选择菜单 Tools 下的 Text Properties 选项。

如果用鼠标左键双击文本对象，或者从图 7-13 所示的弹出式菜单中选择 String，那么文本字符串将还原为原始字符串的内容，并能够对其中的字符进行编辑。

用鼠标左键选中文本并拖动鼠标可以任意改变文本的位置。

在图 7-14 所示的编辑字体属性对话框中可以设置文本字符串的字体名称、样式、大小等属性。

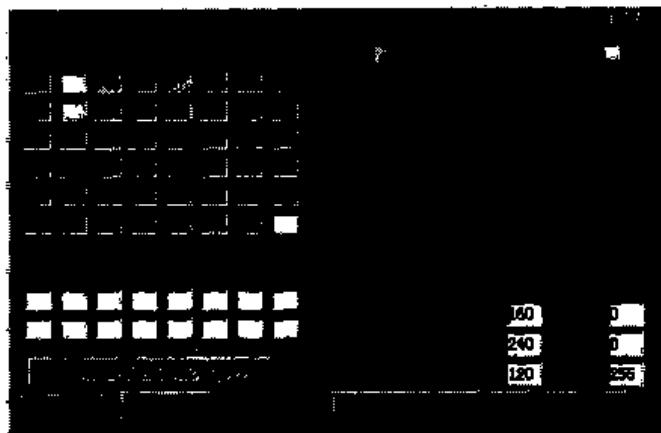


图 7-12 颜色设置对话框

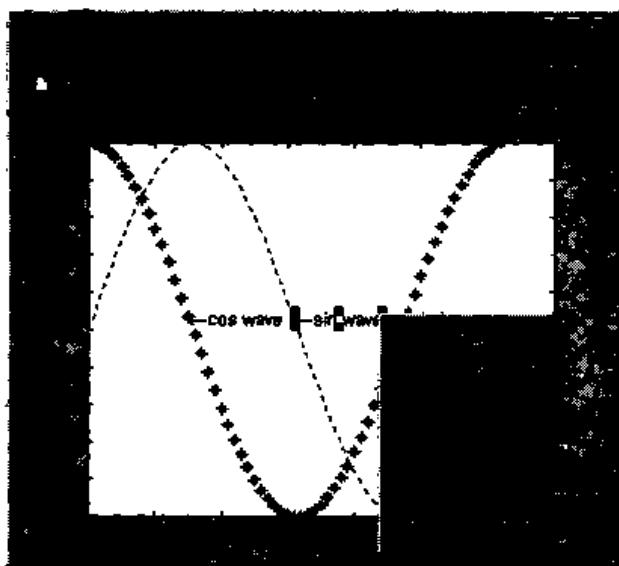


图 7-13 文本对象被选中的状态和弹出式菜单

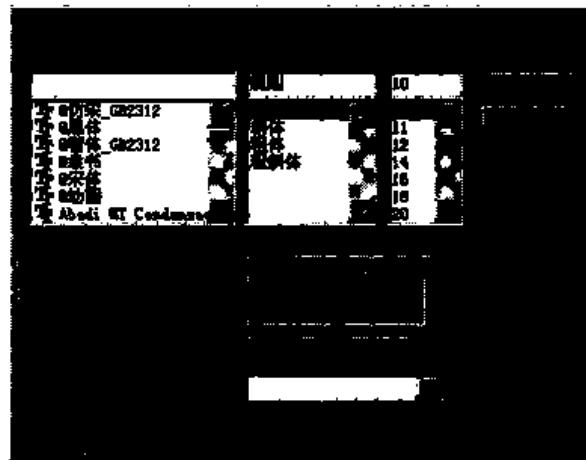


图 7-14 编辑字体属性对话框

## 7.2.2 图形属性编辑器

采用 7.2.1 节讲到的方法可以对图形窗口中的坐标轴、线条和文本对象的几个基本属性进行交互式编辑，用法简单，但有很大的局限性：一方面只能对这三种对象进行编辑，另一方面只能编辑几个基本属性。

本节我们将介绍功能强大的图形属性编辑器，它可以对所有图形对象的所有属性进行交互式编辑，实现 `set` 和 `get` 函数的所有功能。

在图形窗口的 File 菜单下选择 Property Editor 选项即弹出图形属性编辑器，如图 7-15 所示。

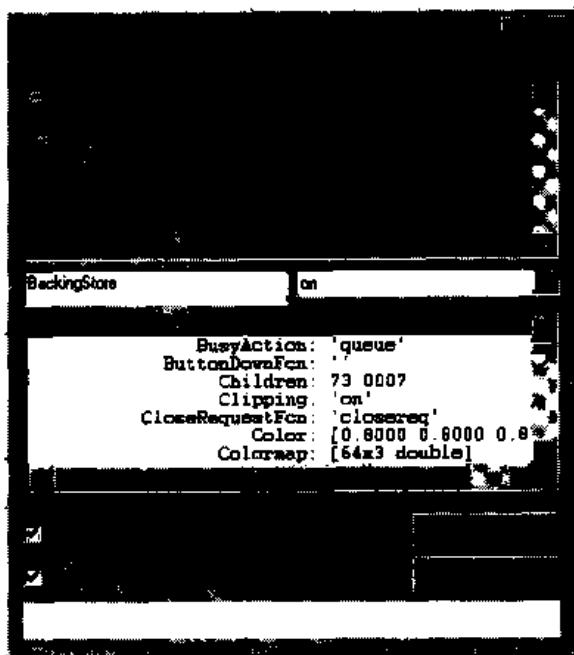


图 7-15 图形属性编辑器

在图 7-15 中，图形属性编辑器上面的大窗口称为对象浏览器（Object Browser），编辑器下面的复选框“Show Object Browser”控制该窗口是否显示。对象浏览器显示了在当前图形窗口中的所有图形对象，各种对象按层次排列，双击左侧有“+”号的对象，该对象的下一层即被展开。用鼠标左键单击要编辑的对象，该对象就被选中，同时在图形窗口的显示成为编辑状态。

编辑器下面的大窗口为属性列表（Property List），编辑器下面的复选框“Show Property List”控制该窗口的显示。属性列表显示的是选中的图形对象的所有属性，包括只读属性。在这个窗口中用鼠标左键单击来选取需进行编辑的属性，该属性的名字和可能的取值即分别出现在编辑器中央的左右两个小区域中，就在这两个区域对指定的属性进行编辑。

例如，考虑图 7-16 所示的网格图，其中的网面属于表面对象，采用上一节讲的方法无法对其进行编辑。我们在这里使用图形属性编辑器来编辑网面的颜色属性，当前网面的颜色是白色。

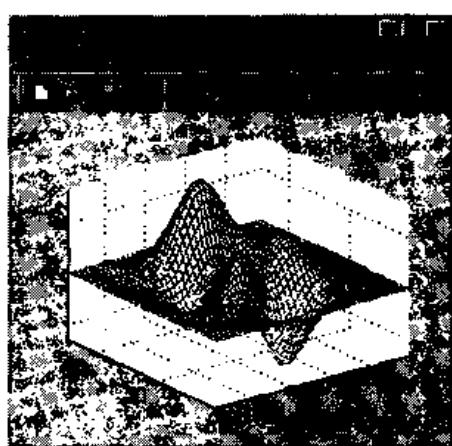


图 7-16 网格图

首先在菜单 File 下选择 Property Editor 运行图形属性编辑器，依次双击对象浏览器中的 figure、axes 对象，显示出 axes 对象的子对象，并选择其中的 surface 对象，在属性列表中找到 FaceColor（表面颜色）属性，这时中间的区域显示出 FaceColor 的当前属性值，并在最下面的小窗口中提示 FaceColor 的可能取值。然后我们单击 FaceColor 后带有省略号“...”的白色按钮（该按钮的颜色和设置的颜色一致，当前 FaceColor 的颜色值是白色），这时弹出颜色设置对话框，参见前面的图 7-12，选择其中的黄色，并按“确定”按钮返回，网面的颜色已经改为黄色了。

这时的图形属性编辑器参见图 7-17，图 7-18 显示了现在的网格图状态。网面周围的虚线框表示它处于编辑状态，退出编辑过程后该虚线框自行消失。

用同样的方法还可以对图形的菜单属性进行编辑。首先在对象浏览器中打开 Tools 菜单，选择其中的 Show Toolbar 命令，如图 7-19 所示。然后在属性列表中选取要编辑的属性。例如，我们要为该命令添加加速键“Ctrl+t”，则需要选取 Accelerator 属性，并输入在图形实现编辑器中间的区域输入加速键的字符“t”即可。然后用相同的方法给 Zoom In 和 Zoom Out 命令分别添加加速键“Ctrl+i”和“Ctrl+o”。设置完成后，Tools 菜单将成为图 7-20 的样子。这时按下加速键就会执行相应的命令。

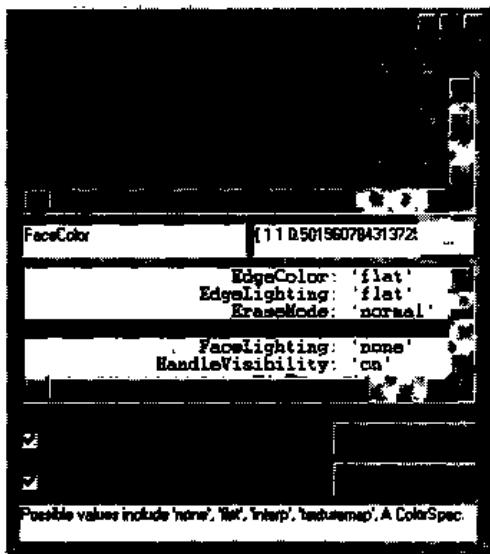


图 7-17 编辑网面的颜色属性

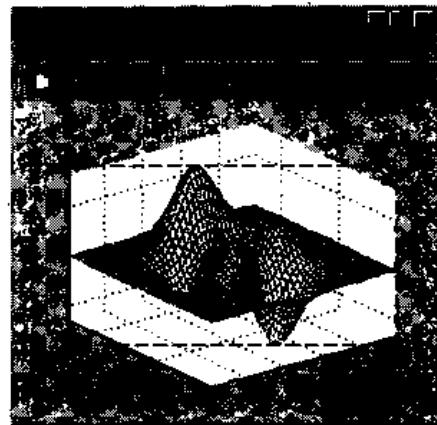


图 7-18 编辑过程中的效果

图形属性编辑器共有四种启动方式，分别为：

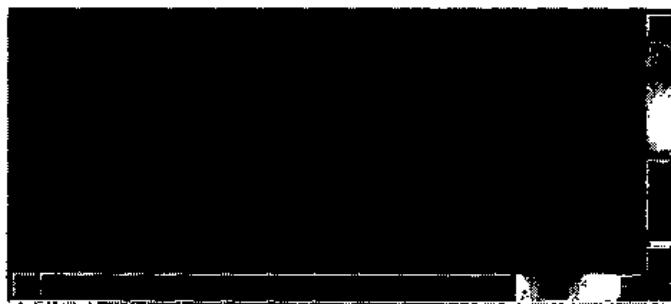


图 7-19 显示在对象浏览器中的 Tools 菜单

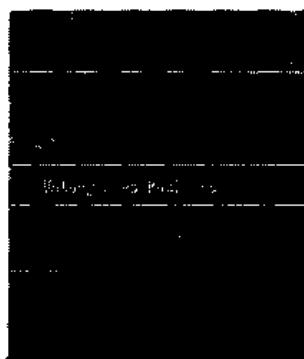


图 7-20 添加了加速键的菜单

- 1) 从图形窗口的 File 菜单下选择 Property Editor 选项 (如前所述);
  - 2) 从 MATLAB 主命令窗口的 File 菜单下选择 Show Graphics Property Editor 选项;
  - 3) 在 MATLAB 命令窗口或 M 文件中执行命令 propedit;
  - 4) 在后面将要讲到的 GUI 设计向导控制面板中按下 Property Editor 按钮;
- 选用其中的哪种方式取决于用户的需要, 之间没有明显区别。

## 7.3 GUI 设计向导

在上一节, 我们利用图形属性编辑器编辑图形对象的属性, 编辑器本身就是一个很好的图形用户界面。利用它编辑图形对象, 交互性好、使用方便, 节省了很多编程的工作量。如果我们的程序也能具有良好的用户界面, 那将给用户提供很大的方便。

### 7.3.1 GUI 设计向导控制面板

#### 1. 控制面板的功能

MATLAB5.3 为图形用户界面的设计提供了一个功能强大的设计向导, 该向导也是以图形用户界面的形式提供给用户的, 从 MATLAB 的 File 菜单下选择 Show GUI Layout Tool 命令, 就会弹出图 7-21 所示的 GUI 设计向导控制面板。

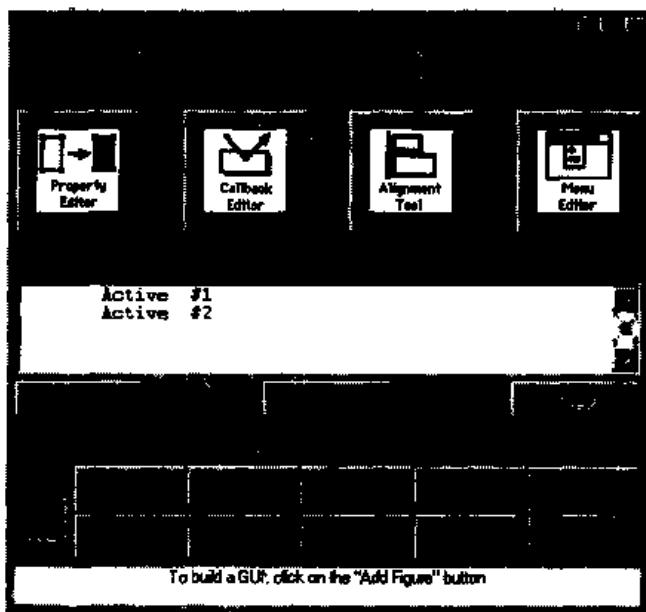


图 7-21 GUI 设计向导控制面板

**注:** 在 MATLAB 命令窗口或 M 文件中打开 GUI 设计向导控制面板的命令是 guide.

在这个控制面板中包括了编辑图形属性、设计 GUI 菜单和 GUI 控件的交互式工具。面板上方的四个按钮的功能分别为:

- **Property Editor:** 编辑图形属性 (见上一节)
- **Callback Editor:** 编辑菜单、控件以及某些事件的回调函数
- **Alignment Tool:** 调整控件排列方式的工具

- **Menu Editor:** 编辑和设计菜单

面板中间的 **Figure List** 列表框列出了当前根屏幕下的所有图形窗口，这里显示出当前根屏幕下有两个图形窗口，其中的 **Active** 表示图形窗口处于活动状态。下面的按钮 **Refresh List** 用于刷新列表；**Add Figure** 添加图形窗口到根屏幕，该图形窗口处于被控制状态（**Controlled**）；**Apply** 用于把状态改变（**Active** 或 **Controlled**）应用到图形窗口。

面板下面的 **New Object Palette** 中包括了可供添加的对象，包括坐标轴对象和各种 GUI 控件。

面板的最下方是提示栏，显示操作的简单帮助信息。

## 2. 控制面板的基本使用方法

如果要设计一个新的图形用户界面，那么需要在 GUI 设计向导控制面板中按下按钮 **Add Figure**，这时 MATLAB 会自动建立一个新的图形窗口，该窗口处于被控制状态，同时在 **Figure List** 中列出该图形窗口的状态“**Controlled #1**”，其中“#1”为该图形窗口的序号，**Controlled** 表示该图形窗口处于被控状态。如图 7-22 所示。

同时我们还看到下面 **New Object Palette** 中的内容从不可操作状态成为可操作状态。

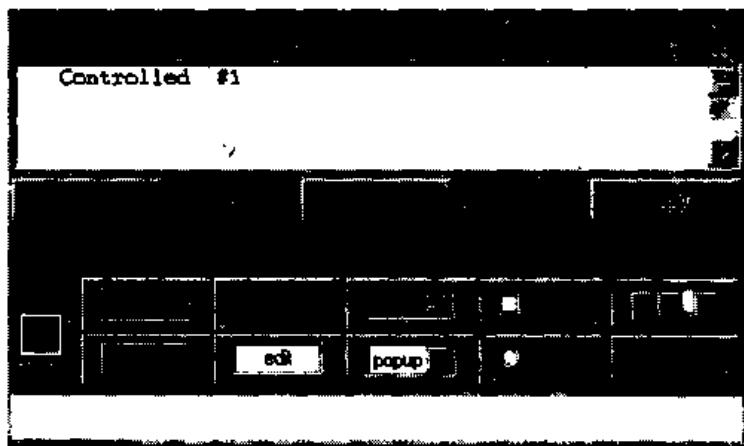


图 7-22 添加图形窗口

如果要编辑已有的图形用户界面，首先要在 MATLAB 命令窗口运行该界面的 M 文件打开用户界面窗口。出现该用户界面后，再执行 GUI 设计向导控制面板，这时在 **Figure List** 中列出的图形窗口状态是 **Active**，这表示该图形窗口处于正常的活动状态。用鼠标单击 **Active** 的行，列表框中的 **Active** 会立即变为 **Controlled**，同时 **Apply** 按钮成为可操作状态，按下 **Apply** 按钮后图形窗口即成为被控制状态。

对被控制状态的图形即可以利用控制面板中的各种设计向导来设计、修改自己的图形用户界面。

### 7.3.2 利用向导设计菜单

按下控制面板中的 **Menu Editor** 按钮即弹出菜单编辑器。本节我们将介绍如何利用该编辑器在图形用户界面中设计自己的菜单。

注：在 MATLAB 命令窗口或 M 文件中打开 GUI 菜单设计编辑器的命令是 **menuedit**。

### 1. 添加新菜单

每个 MATLAB 的图形窗口本身在建立时就都带有五个基本的菜单：File, Edit, Tools, Windows, Help。利用菜单编辑器我们可以在这几个菜单的基础上添加自己的菜单。

首先按下 GUI 设计向导控制面板中的 Add Figure 按钮，添加一个处于被控制状态的图形窗口。然后按下 Menu Editor 按钮启动菜单设计向导，如图 7-23 所示。

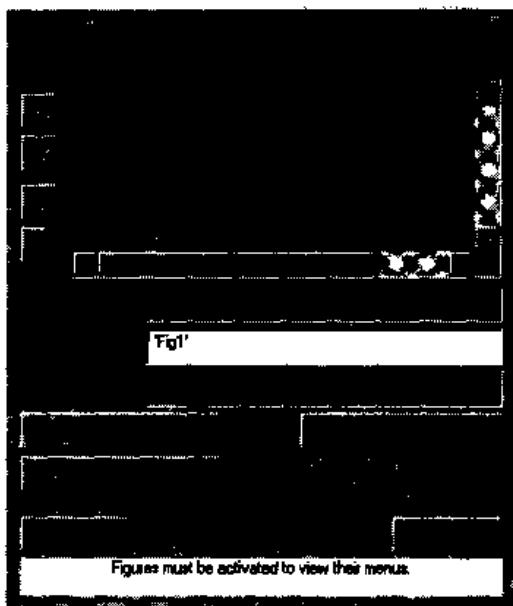


图 7-23 菜单编辑器

菜单编辑器上部的窗口按层次显示了当前图形窗口中的所有菜单项。窗口左侧的四个按钮分别用来调整菜单项的顺序和层次，它们的作用如下：

- ：菜单项向上移动一位
- ：菜单项向下移动一位
- ：菜单项提升一层
- ：菜单项降下一层

菜单编辑器中间的几个编辑框分别是菜单项的标注（Label）、标记（Tag）和回调程序（Callback）的内容。

在菜单项的标注字符串中可以定义相应的快捷键，从而能够使用组合键“Alt+字符”打开相应的菜单项。方法是指定标注字符串时，在该字符的前面加上符号“&”。注意，在同一层次的菜单中不能有相同的快捷键。

回调程序的内容可以是 MATLAB 的函数、命令、可执行的表达式，也可以调用自己编写的函数。

下部的按钮 New Context Menu 用于新建一个上下文菜单。

按钮 New Menu 用于在当前菜单项的下一层新建一个常规菜单。

按钮 Apply 把当前的编辑结果应用于菜单。

注意：这时不会在图形窗口显示新建的菜单，必须要使图形窗口的状态成为 Active 后，这些菜单才会显示出来。

下面我们举例说明在图形窗口添加菜单的方法。

首先建立一个处于被控制状态的图形窗口，如图 7-24 所示。然后按下控制面板中的按钮 Menu Editor 启动如图 7-23 的菜单编辑器。

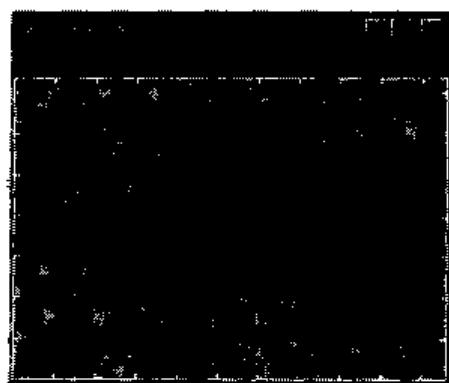


图 7-24 处于被控制状态的图形窗口

在菜单编辑器上面的窗口中用鼠标单击“figure”所在的行，然后按下 New Menu 按钮，这时将在“figure”下面出现新菜单项的内容，其缺省名为“uimenu”，和图形窗口的基本菜单处在同一个层次（第一层，也就是图形窗口的下一层）。

这时下面的三个编辑框都成为可编辑状态，如图 7-25 所示。在编辑框中输入菜单的标注、标记和回调程序，然后单击 Apply 按钮，输入内容就会自动添加到上面的列表框中。

完成一个菜单项的设置后，用相同的方法在适当的层次上可以继续建立菜单项。下面列出了我们建立每一个菜单项的简要步骤和输入内容：

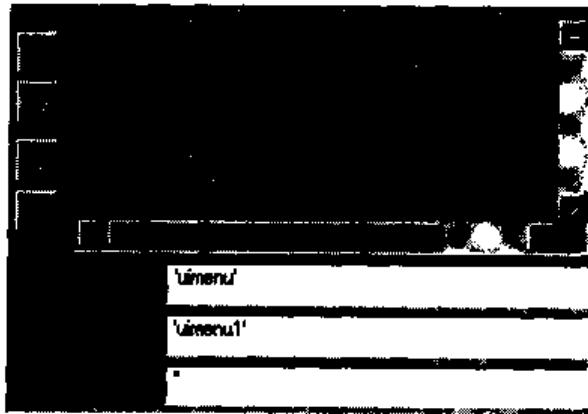


图 7-25 未编辑的新菜单项

### (1) 建立第一层菜单

单击“figure”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Draw'

**Tag:** '绘图'

**Callback:** ''

### (2) 建立第二层菜单

单击“Draw”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Mesh'

**Tag:** '三维网格'

**Callback:** 'mesh(peaks); axis tight'

单击“Draw”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Contour'

**Tag:** '三维等高线'

**Callback:** 'contour3(peaks,20); axis tight'

单击“Draw”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** 'C&olorMap'

**Tag:** '颜色映象'

**Callback:** ''

(3) 建立第三层菜单（第二层菜单 ColorMap 的子菜单）

单击“ColorMap”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Summer'

**Tag:** '暖色调'

**Callback:** 'colormap(summer)'

单击“ColorMap”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Cool'

**Tag:** '冷色调'

**Callback:** 'colormap(cool)'

单击“ColorMap”所在的行，然后按下 New Menu 按钮。在各编辑框中输入：

**Label:** '&Default'

**Tag:** '还原缺省色调'

**Callback:** 'colormap("default")'

至此我们的几层菜单就设置好了。这时菜单编辑器的列表框中包括了我们设计的菜单内容，如图 7-26 所示。

下面按下 Close 按钮关闭菜单编辑器，到 GUI 设计向导控制面板中把图形窗口的状态改为 Active，并按下 Apply 键，这时 MATLAB 弹出对话框提示在使图形窗口成为活动状态前保存图形，如图 7-27 所示，按下 Yes 按钮，出现保存文件对话框，输入要保存的文件名（这里输入“draw1”），该图形窗口包括自己设计的菜单被保存到 M 文件中。以后只要在 MATLAB 命令窗口运行 draw1，就会出现该图形窗口。

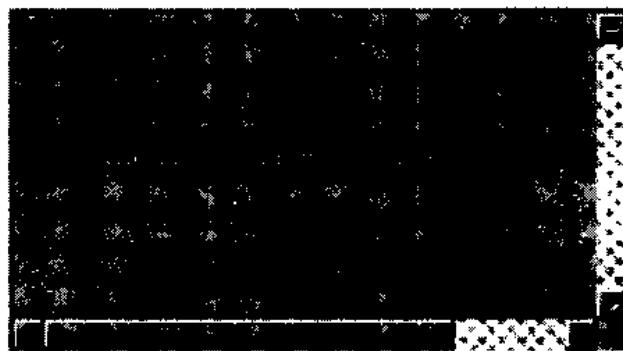


图 7-26 菜单设计完成后的列表框

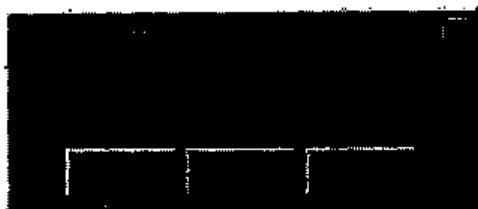


图 7-27 提示保存图形的对话框

图形窗口成为活动状态后，我们新添加的菜单将会出现在该窗口中，在菜单 Draw 下选择 Contour 命令，将会在窗口中绘制三维等高线，相应的菜单和图形如图 7-28 所示。

下一步先用菜单 Draw 下的 Mesh 命令绘制三维网格图，然后利用子菜单 ColorMap 中的命令来调整图形颜色映象的色调，选择其中的 Cool 选项，这时网格图的颜色映象将调整为冷色调，相应的菜单和图形如图 7-29 所示。

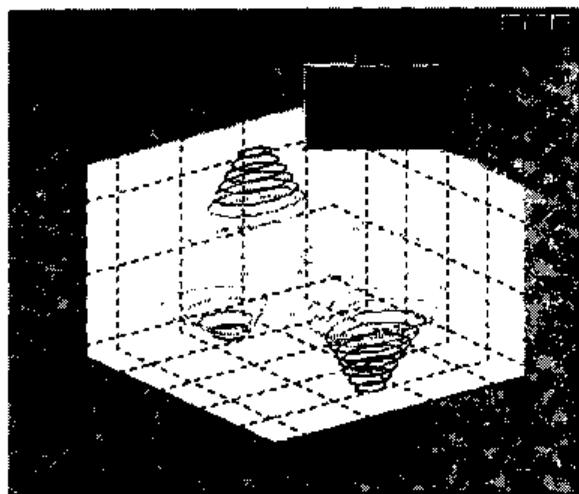


图 7-28 用自己设计的菜单绘制等高线

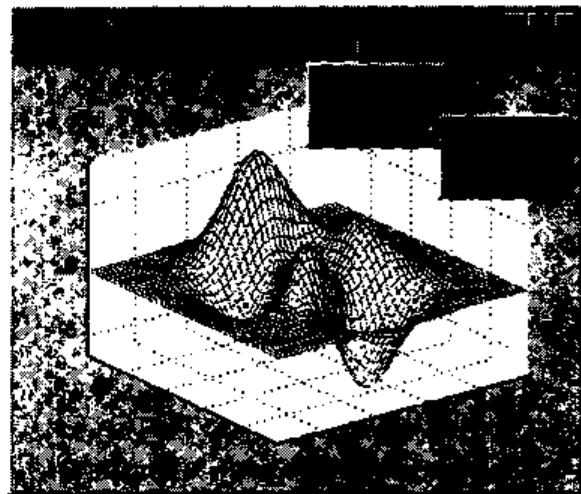


图 7-29 绘制网格图并改变其颜色映象

## 2. 菜单的进一步处理

设计好的菜单如果需要进一步处理根据要调整的内容有两种不同的方法。

(1) 调整菜单项的顺序和层次关系、标注、标记、回调程序等内容。

这些调整仍可以通过菜单编辑器来完成。

首先运行已经设计好的菜单的 M 文件，显示出菜单所在的图形窗口。然后启动 GUI 设计向导控制面板，把图形窗口设置为被控制状态，再运行菜单编辑器。

从编辑器的列表框中选择要调整的菜单项，要调整菜单项的顺序和层次关系可以通过列表框左侧的四个按钮来完成；要调整菜单项的标注、标记、回调程序等内容可以直接在下面的编辑框中进行修改。

(2) 调整菜单项的其它属性

如果要为菜单项添加加速键、分隔符，或在菜单项前标记选中状态等，那么需要借助于我们在前面讲过的图形属性编辑器。

首先仍然需要运行设计好的 M 文件，显示出菜单所在的图形窗口。然后调出图形属性编辑器，选中要编辑的菜单项，并修改相应的属性。

为菜单项添加加速键的方法前面已经讲过。

菜单的属性“Separator”用于设置是否在该菜单项前添加分隔符，把该属性设为“on”即在该菜单项之前添加分隔符。

菜单的属性“Checked”如果设为“on”就在菜单项前添加选中标记。

如图 7-30 就是我们经过进一步调整后的菜单。

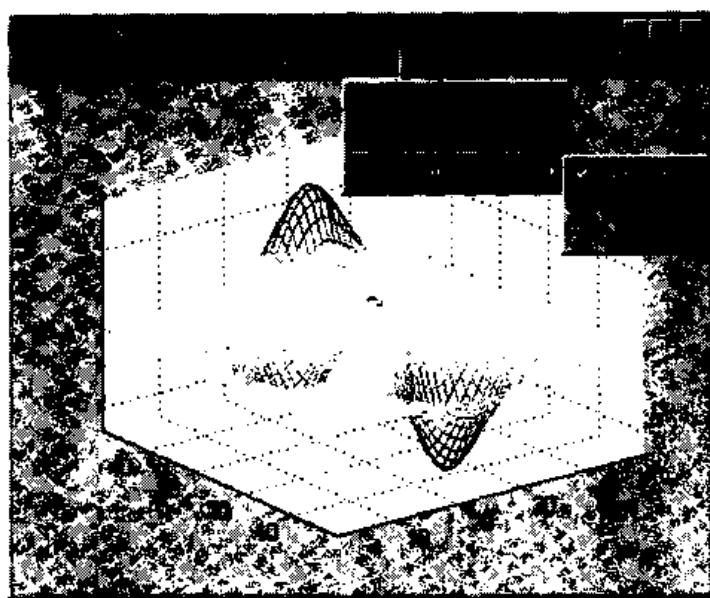


图 7-30 调整后的菜单

### 3. 建立独立的菜单体系

每个 MATLAB 的图形窗口本身在建立时就都带有五个基本的菜单：File, Edit, Tools, Windows, Help。前面我们讲了如何在此基础上添加自己的菜单。那么如果我们要建立一个独立的菜单体系，File、Edit、Windows、Help 等最基本的菜单都由自己设计，那么首先要把图形窗口本身的基本菜单去掉。

去掉图形窗口的基本菜单需要借助于图形属性编辑器（Property Editor），选择对象浏览器中的 figure 对象，然后把它的属性“MenuBar”设为“none”即可（属性“MenuBar”的缺省值是“figure”）。

例如，如果把前面例子中图形对象的属性“MenuBar”设为“none”，那么图形窗口的菜单条上将只有一个菜单“Draw”。这时再打开菜单编辑器，几个基本菜单也从列表框中去掉了。

现在添加菜单的方法和前面讲的一样，而且可以从 File 菜单开始，设计一个完整而独立的菜单体系。

#### 7.3.3 利用向导设计控件

象 GUI 菜单一样，GUI 控件也提供了一种友好的交互方式，给用户的操作带来很多方便。而且，控件比菜单更加直观。

利用 GUI 设计向导控制面板可以实现多种控件的设计。本节将介绍各种控件的设计方法。

### 1. 控件的种类

在 MATLAB 的 GUI 设计向导控制面板下部，以按钮的形式排列着可供选择的控件，如图 7-31 所示。

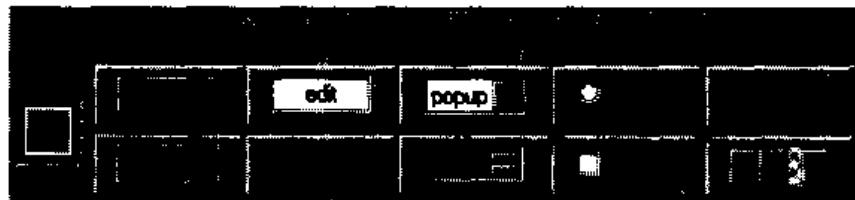


图 7-31 供选择的控件

#### (1) 各种控件介绍

这些图标对应控件的中英文名称和含义分别为：

：按钮（pushbutton），是小的矩形面，常常在上面标有文本。将鼠标指针移动至按钮，单击鼠标，按钮被按下随即自动弹起，并执行回调程序。

按钮的“Style”属性值是“pushbutton”，在属性“String”中定义它上面标注的文本。

：开关按钮（togglebutton），和一般按钮形状相同，区别在于它有两种状态——“开”（按钮按下）和“关”（按钮弹起），用鼠标单击按钮，它会从一种状态变成另一种状态，并执行相应的回调程序（两种状态各对应不同的回调程序）。

开关按钮“开”时，Value属性的值为在Max属性中指定的值；“关”时，Value属性的值为在Min属性中指定的值。

按钮的“Style”属性值是“togglebutton”，在属性“String”中定义它上面标注的文本。

：编辑框（edit），允许用户动态地编辑文本字符串或数字，就象使用文本编辑器或文字处理器一样。编辑框一般用于让用户输入或修改文本字符串和数字。

编辑框的“Style”属性值是“edit”，在“String”属性中设置开始显示的字符串。

：弹出式菜单（popupmenu），向用户提出互斥的一系列选项清单，用户可以选择其中的某一项。弹出式菜单不同于前面讲过的菜单（下拉式菜单），它不受菜单条的限制，可以位于图形窗口内的任何位置。

通常状态下的弹出式菜单以矩形的形式出现，矩形中含有当前选择的选项，在选项右侧有一个向下的箭头来表明该对象是一个弹出式菜单。当指针处在弹出式菜单的箭头之上并按下鼠标时，出现所有选项。移动指针到不同的选项，单击鼠标左键就选中了该选项，同时关闭弹出式菜单，显示新的选项。

当选择一个选项时，弹出式菜单的“Value”属性值为该选项的序号。

弹出式菜单的“Style”属性值是“popupmenu”。在“String”属性中设置弹出式菜单的选项字符串，不同的选项之间用“|”分隔，类似于换行。

：单选按钮（radiobutton），又称无线按钮，它由一个标注字符串（在“String”属性中设置）和字符串左侧的一个小圆圈组成。当选择时，圆圈被填充一个黑点，且属性“Value”的值为1；若未被选择，圆圈为空，属性“Value”的值为0。

单选按钮一般用于在一组互斥的选项中选择一项。为了确保互斥性，各单选按钮的回

调程序需要将其它各项的“Value”值设为0。

单选按钮“style”的属性值是“radiobutton”。

■：图文框（frame），图文框是填充的矩形区域。一般用来把其它控件放入图文框中，组成一组。图文框本身没有回调程序。注意只有用户界面控件可以在图文框中显示。

由于图文框是不透明的，因而定义图文框的顺序就很重要，必须先定义图文框，然后定义放到图文框中的控件。因为先定义的对象先画，后定义的对象后画，后画的对象覆盖到先画的对象之上。

■：静态文本框（text），静态文本框用来显示文本字符串，该字符串是由属性“string”所确定的。静态文本框之所以称之为“静态”，是因为文本不能被动态地修改，而只能通过改变“String”属性来更改。静态文本框一般用于显示标记、提示信息及当前值。

静态文本框的“Style”属性值是“text”。

■：列表框（listbox），列表框列出一些选项的清单，并允许用户选择其中的一个或多个选项，一个或多个的模式由Min和Max属性控制。

Value属性的值为被选中选项的序号，同时也指示了选中选项的个数。

当单击鼠标按钮选中选项后，Value属性的值被改变，释放鼠标按钮的时候 MATLAB 执行列表框的回调程序。

列表框的“Style”属性值是“listbox”。

■：复选框（checkbox），又称检查框，它由一个标注字符串（在“String”属性中设置）和字符串左侧的一个小方框所组成。选中时在方框内添加“√”符号，“Value”属性值设为1；未选中时方框变空，“Value”属性值设为0。复选框一般用于表明选项的状态或属性。

■：滑动条（slider），又称滚动条，包括三个部分，分别是滑动槽，表示取值范围；滑动槽内的滑块，代表滑动条的当前值；以及在滑动条两端的箭头，用于改变滑动条的值。

滑动条一般用于从一定的范围内取值。改变滑动条的值有三种方式，一种是用鼠标指针拖动滑块，当滑块位于期望位置后松开鼠标；另一种是当指针处于滑动槽中但不在滑块上时，单击鼠标按钮，滑块沿该方向移动一定的距离，距离的大小在属性“SliderStep”中设置，为该属性值的第二个元素，缺省情况下等于整个范围的10%；第三种方式是在滑动条的某一端用鼠标单击箭头，滑块沿着箭头的方向移动一定的距离，距离的大小在属性“SliderStep”中设置，为该属性值的第一个元素，缺省情况下为整个范围的1%。

滑动条的“Style”属性值是“slider”。

## （2）控件的几个重要属性

和其它各种图形对象一样，GUI控件也有自己的很多可以设置的属性，请参见附录B。这里我们对其中的几个重要属性给以比较详细的介绍和总结。

Value属性：控件的当前值，格式为标量或向量。该属性对不同的控件有不同的取值方式，分别为：

- 复选框：被选中时Value的值为属性Max中设置的值；未选中时Value的值为属性Min中设置的值。
- 列表框：被选中选项的序号，当有多个选项被选中时，Value属性的值为向量。序

号指的是选项的排列次序，最上面的选项序号为 1，第二个选项序号为 2……。

- 弹出式菜单：和列表框类似，也是被选中选项的序号，只是弹出式菜单只能有一个选项被选中，因而 Value 属性的值为标量。
- 单选按钮：被选中时 Value 的值为属性 Max 中设置的值；未选中时 Value 的值为属性 Min 中设置的值。
- 滑动条：Value 的值等于滑块指定的值。
- 开关按钮：“开”时 Value 的值为属性 Max 中设置的值；“关”时 Value 的值为属性 Min 中设置的值。
- 按钮、编辑框、图文框、静态文本不设置这个属性值。

**Max 属性：**指定 Value 属性中可以设置的最大值，格式为标量。该属性对不同的控件有不同的含义，分别如下所述：

- 复选框：当复选框被选中时 Value 属性的取值。
- 编辑框：如果 Max 的值减去 Min 的值大于 1，那么编辑框可以接受多行输入文本；如果 Max 的值减去 Min 的值小于或等于 1，那么编辑框只能接受一行输入文本。注意：只能输入一行文本时用回车键结束输入；可以输入多行文本时用 Ctrl+回车键结束输入。
- 列表框：如果 Max 的值减去 Min 的值大于 1，那么允许选取多个选项；如果 Max 的值减去 Min 的值小于或等于 1，那么只能选取一个选项。
- 单选按钮：当单选按钮被选中时 Value 属性的取值。
- 滑动条：滑动条的最大值，缺省值是 1。
- 开关按钮：当开关按钮“开”（被选中）时 Value 属性的取值。缺省值是 1。
- 文本框、弹出式菜单、按钮和静态文本框不使用 Max 属性。

**Min 属性：**指定 Value 属性中可以设置的最小值，格式为标量。该属性对不同的控件有不同的含义，分别如下所述：

- 复选框：复选框未被选中时 Value 属性的取值。
- 编辑框：如果 Max 的值减去 Min 的值大于 1，那么编辑框可以接受多行输入文本；如果 Max 的值减去 Min 的值小于或等于 1，那么编辑框只能接受一行输入文本。
- 列表框：如果 Max 的值减去 Min 的值大于 1，那么允许选取多个选项；如果 Max 的值减去 Min 的值小于或等于 1，那么只能选取一个选项。
- 单选按钮：单选按钮未被选中时 Value 属性的取值。
- 滑动条：滑动条的最小值，缺省值是 0。
- 开关按钮：当开关按钮“关”（未被选中）时 Value 属性的取值。缺省值是 0。
- 文本框、弹出式菜单、按钮和静态文本框不使用 Min 属性。

## 2. 控件的建立

上面我们介绍了几种控件的功能和用法，本小节我们将举例说明如何在图形窗口中建立控件以得到良好的图形用户界面。

当图形窗口处于被控制状态时，在控制面板中按下需要添加的控件的图标，然后把鼠标指针移动到图形窗口的适当位置，单击鼠标左键，在该位置出现了控件的图标，双击图标即弹出图形属性编辑器显示出该控件的属性，并可以进行编辑。

这里我们将在前面设计了菜单的图形窗口基础上添加功能类似的控件，增强界面的交互能力。

### (1) 建立控件并设置基本属性

运行 draw1 命令调出该图形窗口，并启动 GUI 设计向导控制面板，把该窗口设为被控制状态。这时控制面板下部的控件图标成为可选状态。

先改变原图形窗口坐标轴对象的大小和位置，以给建立控件留出位置。改变坐标轴的大小可以直接用鼠标选中坐标轴，出现黑色边框，然后用鼠标拖动边框上的黑色方块来改变大小，用鼠标拖动整个边框就可以移动坐标轴的位置。移动后如图 7-32 所示。

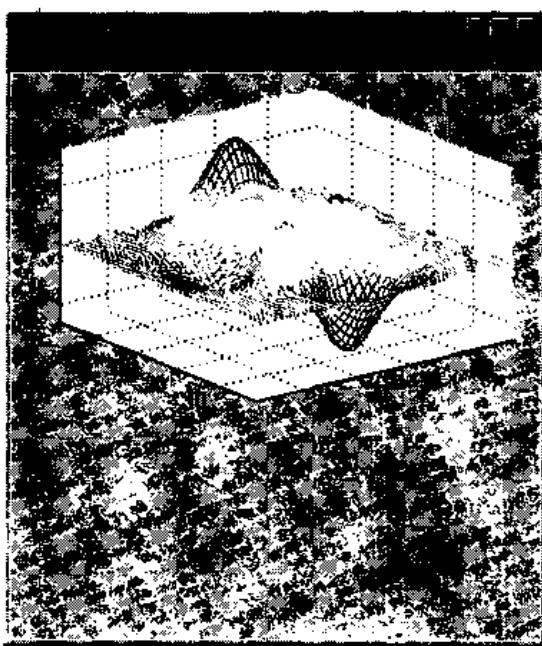


图 7-32 调整坐标轴对象

现在把需要的控件图标复制到图形窗口中坐标轴的下方的适当位置，并调整到合适的尺寸，我们选择了三个按钮、一个静态文本框、一个弹出式菜单。然后双击图形窗口中控件的图标编辑它们的属性。

三个按钮分别用于绘制网格图、等高线和退出操作，因而在它们的“String”属性中分别输入：“画网格图”、“画等高线”、“退出”，这几个字符串将作为该按钮的标注，同时在属性“FontSize”中设置字体大小为“12”。

弹出式菜单列出几种颜色映象的色调供用户选择，因而在其属性“String”中输入“缺省色调|冷色调|暖色调”，这些字符串是弹出式菜单的选项，同时在属性“FontSize”中把字体大小设为“12”。

静态文本框放在弹出式菜单上方起到提示作用，在其属性“String”中输入文本框中的文本“选择色调：”，并在属性“FontSize”中把字体大小设为“12”。缺省情况下，控件中的标注文本都是居中排列的，而在这个静态文本框中我们希望文本靠左排列，因而还需要把属性“HorizontalAlignment”设为“left”。

这些属性设置好后，控件的状态如图 7-33 所示。

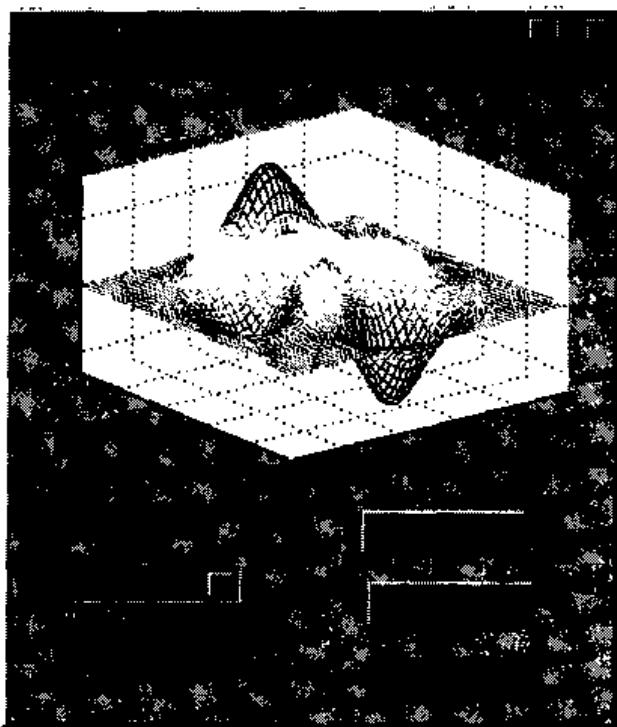


图 7-33 经初步设置的控件

## (2) 设置对齐方式

从图 7-33 中可以看出，手工排列的控件很难真正对齐，因而我们需要借助 MATLAB 提供的排列对象的工具“Alignment Tool”。

Alignment Tool 也是在 GUI 设计向导控制面板中提供的，按下控制面板中的 Alignment Tool 按钮就出现该工具的对话框，如图 7-34 所示。

注：在 MATLAB 命令窗口或 M 文件中打开 GUI 图形对象排列工具的命令是 align。

该对话框上方的窗口列出了图形窗口中可以进行排列的对象，除了控件外还包括坐标轴，在这里选择要处理的对象。在该窗口中按下鼠标左键并向上或向下拖动鼠标可以连续选中几个对象；还可以选中一个对象，在另外一个位置按下“Shift”键再选中另外—个对象，这样将选中两个对象之间的所有对象（包括两个对象在内）；如果需要同时选择几个不相邻的对象，可以按下“Ctrl”键逐个选择。

对话框中部的图标按钮有三种：

- Align 下的六个按钮用于设置对象的对齐方式，包括水平方向和竖直方向的排列；
- Distribute 下的两个按钮用于使几个对象等间距分布，包括水平和竖直两个方向的分布；
- Set Spacing 下的两个按钮用于指定对象的水平或竖直间距，以“点”为单位指定数值。每个按钮设置的排列方式在按钮上的图标中标注得很清楚。

对话框下面的 Apply 按钮把设置的排列方式应用到选中的对象，按下排列方式按钮后，要再按下 Apply 按钮，设置才会生效。Revert 按钮取消上次 Apply 按钮的操作，使对象恢复到 Apply 执行前的状态。

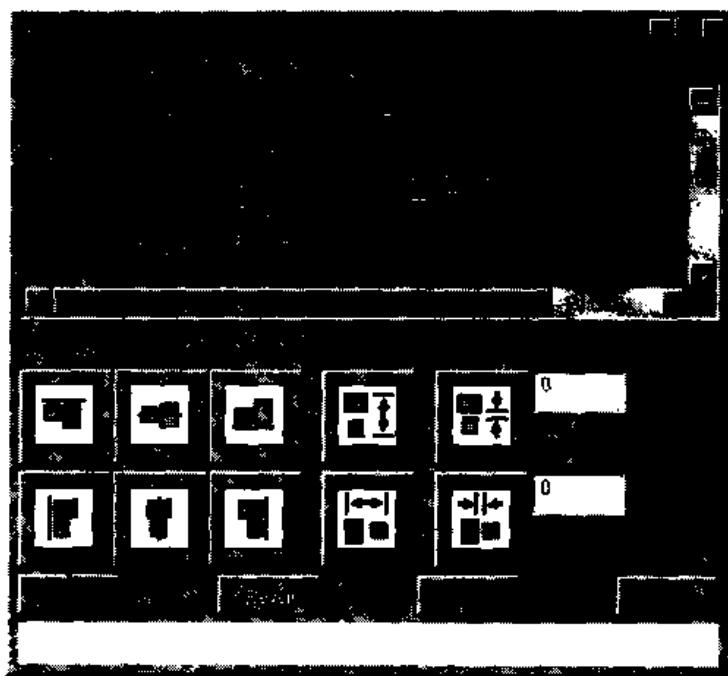


图 7-34 排列工具对话框

例如，我们要把图 7-33 中的三个按钮排列整齐，在选中三个按钮之后（参见图 7-34），首先按下 Distribute 下的第一个图标，使选定的三个按钮上下的间隔相等，并按 Apply 使格式生效；然后按下 Align 下第二排的第三个图标，使得选中的三个按钮靠右对齐。

用同样的方法把静态文本框和弹出式菜单排列好，全部排列整齐后的结果如图 7-35 所示。

### (3) 编写回调程序

控件可以象菜单一样在属性“Callback”中输入回调程序，这里我们介绍如何利用 GUI 设计向导控制面板中的工具“Callback Editor”（回调程序编辑器）编写回调程序。

在控制面板中按下 Callback Editor 按钮就弹出回调程序编辑器的对话框，如图 7-36 所示。

**注意：**在 MATLAB 命令窗口或 M 文件中打开 GUI 回调程序编辑器的命令是 guide。

该编辑器可以编写任何图形对象的回调程序，除了菜单、控件外还包括图形窗口、坐标轴、线条、文本标注等等。

对于不同的事件 (event) 有不同类型的回调程序，而且每一类图形对象包括的事件种类也是不同的，图 7-36 中显示了图形窗口对象包含的九类事件，事件“Callback”是用户界面菜单和控件特有的，关于图形对象的事件请参见附录 B。回调程序编辑器可以编写每一类事件的回调程序。

首先选择要编写程序的对象，然后选择相应的事件，比如用户界面菜单或控件的“Callback”事件或一般图形对象的“ButtonDownFcn”事件等。在中间的窗口输入回调程序的内容。

下面我们将为在图形窗口中添加的几个控件编写相应的回调程序。

按钮“画网格图”的回调程序为：

```
mesh(peaks);
```

```
axis tight
```

按钮“画等高线”的回调程序为：

```
contour3(peaks, 20);
```

```
axis tight
```

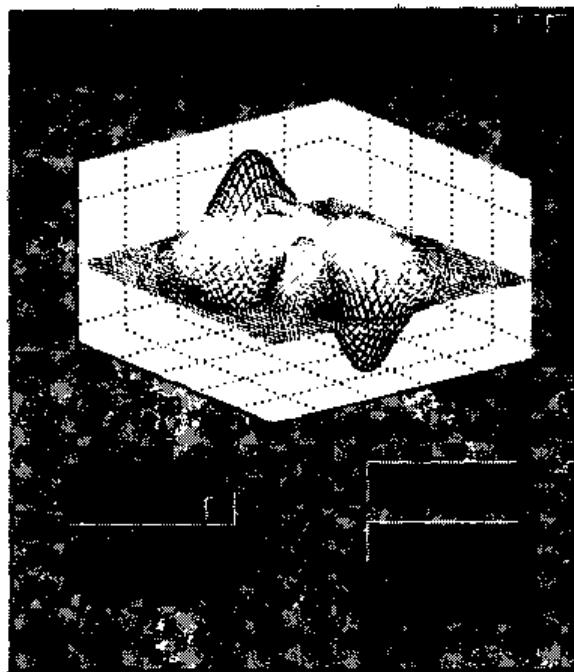


图 7-35 控件排列整齐后的效果

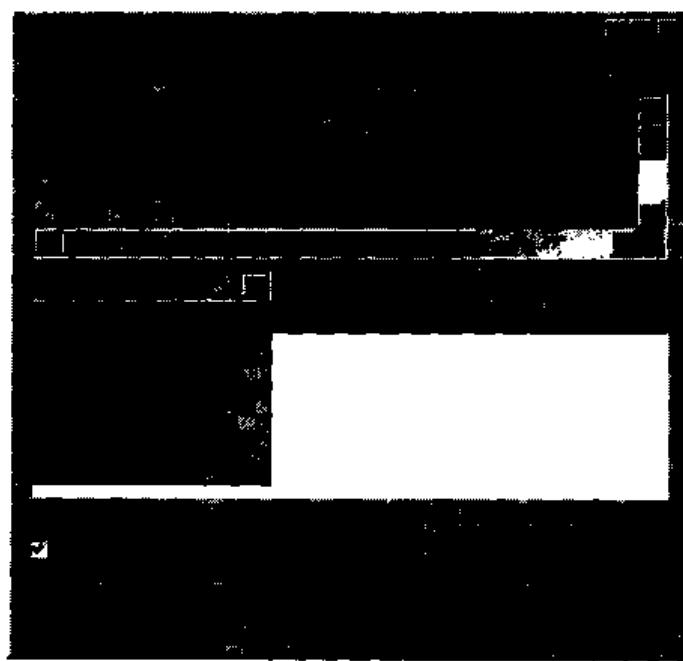


图 7-36 回调程序编辑器

按钮“退出”的回调程序为：

```
close
```

弹出式菜单的回调程序为：

```
v = get(findobj(gcf, 'style', 'popupmenu'), 'Value');
switch v
    case 1
        colormap('default')
    case 2
        colormap(cool)
    case 3
        colormap(summer)
end
```

在弹出式菜单的回调程序中，我们用 `findobj` 函数来获取弹出式菜单的句柄，因为该控件是通过交互方式建立的，所以我们不知道它的句柄，而用 `findobj` 函数则可以通过唯一识别该控件的信息——属性“Style”的值来获得该控件的句柄。函数 `get` 用来获取控件的“Value”属性的值，也就是用户选择的选项序号，命令 `switch` 用这个序号来控制各个选项和回调程序的对应关系。

至此，一个功能比较完整的图形用户界面就建立起来了。它包括用户界面菜单和控件，用户可以通过任意一种方式来控制程序的执行，绘制相应的曲线并选择喜欢的颜色映象色调。

例如，在弹出式菜单下选取“冷色调”，然后单击按钮“画等高线”，则最后结果将如图 7-37 所示。

这是我们设计的图形用户界面的全貌，按下其中的“退出”按钮结束操作，关闭该界面。

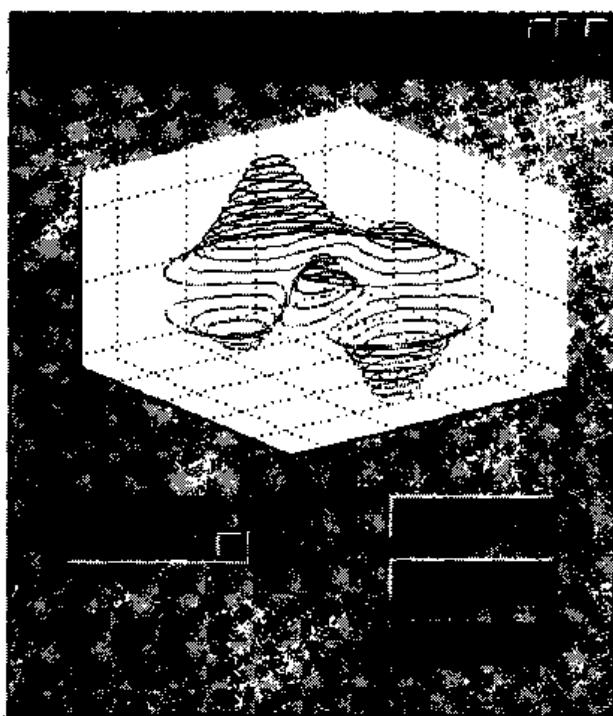


图 7-37 设计完成后的图形用户界面

### 3. 图形用户界面的程序文本

本节我们利用 GUI 设计向导设计了一个自己的图形用户界面，包括菜单和控件，MATLAB 把这个用户界面以 M 文件的形式保存在当前目录下。实际上 GUI 设计向导的作用根据用户的设计产生相应功能的 M 文件，然后运行 M 文件生成用户界面。

下一节我们将介绍如何用函数编写用户界面。

## 7.4 编程设计 GUI

### 7.4.1 编程建立菜单

在上一节列出的图形用户界面 M 文件中我们已经看到，用于建立菜单的函数是 uimenu。本节我们介绍如何使用函数 uimenu 编写程序来建立菜单。

#### 1. 函数 uimenu 的用法

函数 uimenu 的用法和其它建立图形对象的函数类似，一般它有如下两种调用格式：

1) `h = uimenu(PropertyName1,value1,PropertyName2,value2,...)`: 在当前图形窗口顶部的菜单条上建立菜单，同时返回该菜单的句柄。参数 “PropertyName” 和 “value” 分别是菜单对象的属性和设置值。

2) `h = uimenu(Parent, ...)`: 用参数 “Parent” 指定菜单的上一级菜单的句柄，或是图形窗口的句柄。如果 “Parent” 是一个图形窗口的句柄，那么 uimenu 在对应图形窗口顶部的菜单条上建立菜单；如果 “Parent” 是菜单的句柄，那么 uimenu 建立该菜单的子菜单。

菜单对象包括的属性及取值参见附录 E 的表 E-4。

我们在这里强调几种属性的设置：

#### (1) Label 和 Callback

这两个属性是菜单的基本属性，编写一个具有基本功能的菜单必须要设置 Label 和 Callback 属性。

Label 是在菜单项上显示的标注文本，在文本中同时可以设置该菜单项的快捷键：把符号 “&” 放在标注字符串中用于快捷键的字符前，执行该菜单项时就可以用 “Alt+该字符” 来完成。

例如，如下命令在当前图形窗口的菜单条上建立一个菜单并设置其快捷键：

```
h1 = uimenu('Label', '&Graph');
```

建立的菜单项显示将是 “Graph”，快捷键定义为 “Alt+g”。

属性 Callback 用来设置菜单项的回调程序。如下命令在菜单 “Graph” 下建立子菜单，并给出回调程序：

```
h2-1 = uimenu(h1, 'Label', 'Grid o&n', 'Callback', 'grid on');
```

此命令建立的菜单项是 “Grid on”，快捷键定义为 “Alt+n”，回调程序 “grid on” 由于给图形添加网格。

#### (2) Position

属性 Position 用来调整菜单项的相对位置。

例如，在菜单 “Graph” 下增加一个子菜单 “Grid off”，比较一下它和子菜单 “Grid on”

的属性 Position 的值，命令如下：

```

h2-2 = uimenu(h1, 'Label', 'Grid on', 'Callback', 'grid off');
get(h2-1, 'position')
ans =
1
get(h2-2, 'position')
ans =
2

```

可见属性 Position 的值等于菜单项的排列顺序，排在最上面的子菜单“Grid on”的 Position 值为 1，排在第二位的子菜单“Grid off”的 Position 值为 2。如果需要调整相对位置可以用 set 函数来完成：

```
set(h2-1, 'position', 2)
```

该命令把子菜单“Grid on”的 Position 值设为 2，从而把它移到了第二位，子菜单“Grid off”自动移到第一位。

同样，横向排列在图形窗口顶部菜单条上的菜单也可以用 Position 属性来设置其相对位置，方法和纵向排列的菜单相同，只是这时的 Position 值表示菜单横向排列的相对位置。

例如，在菜单条上的“Graph”菜单位于图形窗口的五个基本菜单之后，那么它的 Position 属性值为：

```
get(h1, 'position')
```

```
ans =
```

```
6
```

### (3) Checked 和 Separator

属性 Checked 设置是否在菜单项前添加选中标记，设为“on”表示添加，设为“off”表示不添加。

比如，如下命令给子菜单“Grid on”添加选中标记：

```
set(h2-1, 'Checked', 'on')
```

需要注意，有些菜单的选中标记是相斥的，“Grid on”和“Grid off”就属于这种类型，两个菜单项不能同时标记为选中或不选中，这就要求给一个菜单项添加选中标记的同时去掉另一个菜单项的标记。

属性 Separator 用于在菜单项前添加分隔符，是菜单的层次更加清晰。

### (4) BackgroundColor (背景色) 和 ForegroundColor (前景色)

这两个属性值均为 RGB 向量，BackgroundColor 是菜单本身的颜色，ForegroundColor 是菜单上标注字符串的颜色。

## 2. 用函数 uimenu 设计菜单

下面我们将举例介绍用函数 uimenu 设计菜单的具体过程。

首先建立一个图形窗口，去掉窗口本身的菜单条和工具条，并命名为“My First GUI”。程序如下：

```

h0 = figure('MenuBar', 'none',...
'ToolBar', 'none',...

```

**'Name', 'My First GUI');**

前两行后面的省略号“...”在 MATLAB 语句中表示和下一行相连，用这种方式把比较长的语句分开表示，或者把语句中每段功能比较独立的部分作为一行，这样可以使程序的可读性更强。

下面从左到右依次建立各级菜单。

第一个菜单名为“Mesh”，下面三个选项“Membrane”、“Peaks”、“Sinc”，分别用来绘制隔膜的特征函数、三维高斯分布、Sinc 函数（草帽图）的网格图。程序如下：

```

h1 = uimenu(h0, 'Label', '&Mesh');
h21 = uimenu(h1, 'Label', '&Membrane',...
    'callback','mesh(membrane);axis tight');
h21 = uimenu(h1, 'Label', '&Peaks',...
    'callback','mesh(peaks);axis tight');
h21 = uimenu(h1, 'Label', '&Sinc',...
    'callback',...
    '[X,Y] = meshgrid(-8:0.5:8);',...
    'R = sqrt(X.^2+Y.^2)+eps;',...
    'Z=sin(R)./R;',...
    'mesh(X,Y,Z)']);

```

注意“Sinc”菜单的回调程序书写格式，“callback”属性值是一个 MATLAB 字符串，MATLAB 将它传给函数 eval 并在命令窗口工作空间执行。

字符串可以串接起来生成一个合法 MATLAB 字符串，只是如果该字符串需要分行，那么每一行的语句必须都加上单引号，而且要把整个字符串的几行括在方括号中。

同时注意，除了最后一句命令，其它引号内的各语句必须用分号或逗号结尾，同时在引号后要用空格或逗号结尾。实际上“Sinc”菜单“callback”属性中的几条语句后可以不要逗号，下一行的空格可以起到相同的作用。

下面我们来建立第二个菜单“Colormap”，该菜单下有五个选项，分别用来选择不同的颜色映象。当某个选项被选中时，我们为这个选项添加选中标记，同时去掉其它选项上的选中标记。为了操作方便，每个选项都设置了相应的快捷键。程序如下：

```

h1 = uimenu(h0, 'Label', '&Colormap');
h22(1) = uimenu(h1, 'Label', '&Default',...
    'Accelerator', 'd',...
    'Checked','on',...
    'Callback',...
    ['set(h22,"Checked","off");',...
     'set(h22(1),"Checked","on");',...
     'colormap("default")']);
h22(2) = uimenu(h1, 'Label', '&Spring',...
    'Accelerator', 's',...
    'Callback',...

```

```

['set(h22,"Checked","off");',...
 'set(h22(2),"Checked","on");',...
 'colormap(spring)'});

h22(3) = uimenu(h1, 'Label', 'S&ummer',...
 'Accelerator', 'u',...
 'Callback',...
 ['set(h22,"Checked","off");',...
 'set(h22(3),"Checked","on");',...
 'colormap(summer)'});

h22(4) = uimenu(h1, 'Label', '&Autumn',...
 'Accelerator', 'a',...
 'Callback',...
 ['set(h22,"Checked","off");',...
 'set(h22(4),"Checked","on");',...
 'colormap(autumn)'});

h22(5) = uimenu(h1, 'Label', '&Winter',...
 'Accelerator', 'w',...
 'Callback',...
 ['set(h22,"Checked","off");',...
 'set(h22(5),"Checked","on");',...
 'colormap(winter)']);

```

注意程序中各菜单项的回调程序字符串的格式：在单引号内的字符串用两个单引号（注意：不等于双引号）表示单引号。

在设置选中标记时，我们首先用命令“`set(h22,"Checked","off");`”把向量“`h22`”中的五个句柄对应的菜单项都设为未选中状态，然后，把选择的菜单项设为选中状态，这样保证了几个选项之间的互斥性。

最后建立一个控制坐标轴显示的菜单“Axis”，用于选择是否显示坐标轴。和菜单“Colormap”类似，它的两个选项也添加了选中标记，并且指定了快捷键。程序如下：

```

h1 = uimenu(h0, 'Label', '&Axis');

h23(1) = uimenu(h1, 'Label', 'Axis o&ff',...
 'Accelerator', 'f',...
 'callback',...
 ['axis off;',...
 'set(h23,"checked", "off");',...
 'set(h23(1),"checked", "on");']);

h23(2) = uimenu(h1, 'Label', 'Axis o&n',...
 'Accelerator', 'n',...
 'callback',...
 ['axis on;...

```

```
'set(h23,"checked","off");...
'set(h23(2),"checked","on"));]
```

把这些程序组合起来，就形成了一个功能比较完整的用户界面菜单，三个菜单的菜单项如图 7-38 所示。图 7-39 显示了整个用户界面在选择图 7-38 所示的选项后的执行结果。



图 7-38 三个菜单的菜单项

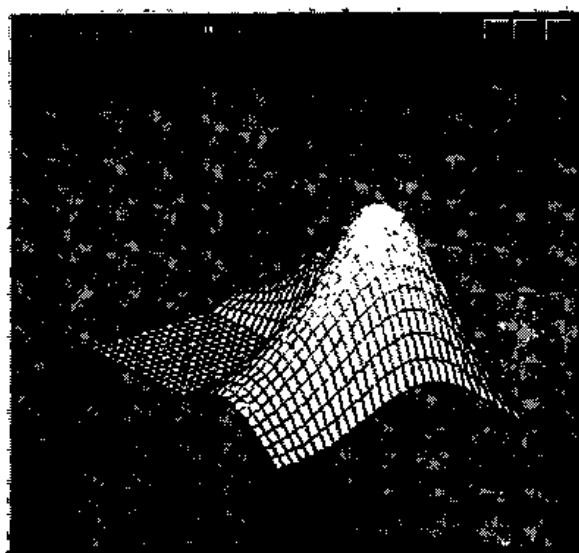


图 7-39 执行结果

#### 7.4.2 编程序建立控件

在上一节我们已经对各种控件一一作了介绍。虽然不同类型的控件属性各不相同，但它们都是 uicontrol 对象，都由函数 uicontrol 来建立，在 uicontrol 的属性“Style”中指定不同的控件类型就可以建立不同的控件。

函数 uicontrol 的用法和函数 uimenu 相同，这里不再重复。

各种控件的“Style”属性值我们在前面已经讲过，下面我们举例说明如何用 uicontrol 函数编写建立控件的程序。

##### 1. 建立图形窗口和坐标轴

首先建立一个图形窗口，指定窗口的大小，以便后面设置坐标轴和控件的位置：

```
h0 = figure('MenuBar','none',...
'ToolBar','none',...
'Position',[198 56 408 468],...
'Name','My Second GUI');
```

指定一个坐标轴对象的位置，这样后面的绘图将在这个坐标轴范围内，而不会自动充

满整个图形窗口，而且为后面建立控件预留位置：

```
h1 = axes('Parent',h0, ...
    'Position',[0.15 0.45 0.7 0.5],...
    'Visible', 'off');
```

### 2. 建立编辑框

在坐标轴区域下建立一个编辑框，用户可以在里面输入图形的标题，编辑框前放置文本框起到提示作用，程序如下：

```
htext1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[20 110 45 15], ...
    'String','Input Title:',...
    'Style','text');

hedit = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[65 110 70 15], ...
    'callback','title(get(hedit,"String"))',...
    'Style','edit');
```

### 3. 建立按钮

在编辑框下方建立三个按钮对象，分别用于建立三种不同的网格图：隔膜的特征函数、三维高斯分布、Sinc 函数（草帽图），程序如下：

```
hpush1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'String','Membrn', ...
    'Position',[15 65 50 18], ...
    'callback','mesh(membrane);axis tight');

hpush2 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'String','Peaks', ...
    'Position',[75 65 50 18], ...
    'callback','mesh(peaks);axis tight');

hpush3 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'String','Sinc', ...
    'Position',[135 65 50 18], ...
    'callback',...;
    ['[X,Y] = meshgrid(-8:0.5:8);'...
    'R = sqrt(X.^2+Y.^2)+eps;'...
    'Z=sin(R)./R;'...
    'mesh(X,Y,Z)']);
```

#### 4. 建立滑动条

按钮下面安排一个滑动条，调节图形颜色的亮度，滑块从中间向右移动图形变亮，向左移动图形变暗。同时，在滑块左侧建立了一个静态文本，用以提示用户滑动条的功能。程序如下：

```
htext2 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[20 30 45 15], ...
    'String','Brightness',...
    'Style','text');

hslider = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[65 30 115 15], ...
    'Min',-1,...
    'Max',1,...
    'Style','slider', ...
    'callback','brighten(get(hslider,"Value")));
```

上面的程序中我们使用 brighten 函数改变颜色的亮度，该函数的调用方式为：

```
brighten(beta)
```

参数 beta 指示调整的幅度，取值范围在 [-1 1] 之间，其中 [-1 0] 表示把当前颜色变暗，(0 1] 表示把当前颜色变亮。

#### 5. 建立单选按钮

现在我们在坐标轴下方的右侧建立五个单选按钮，用于选取五种颜色映象中的一种，每个单选按钮对应一种颜色映象，按钮之间是相斥的关系，不能同时选取多个按钮。在五个按钮上方安排一个静态文本框，提示用户在这里选择颜色映象。把缺省颜色映象（'Default'）按钮的初始状态设为选中状态。程序如下：

```
htext3 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[200 110 80 15], ...
    'String','Select Color :',...
    'Style','text');

hradio1(1) = uicontrol('Parent',h0, ...
    'Units','points', ...
    'Position',[200 95 80 15], ...
    'Style','radiobutton', ...
    'String','Default',...
    'Value',1,...
    'Callback',...
    ['set(hradio1,"Value",0);',...
    'set(hradio1(1),"Value",1);']);
```

```
'colormap("default")]);  
hradio1(2) = uicontrol('Parent',h0, ...  
'Units','points', ...  
'Position',[200 80 80 15], ...  
'Style','radiobutton', ...  
'String','Spring',...  
'Callback',...  
['set(hradio1,"Value",0); '...  
 'set(hradio1(2),"Value",1);'...  
 'colormap(spring)])];  
  
hradio1(3) = uicontrol('Parent',h0, ...  
'Units','points', ...  
'Position',[200 65 80 15], ...  
'Style','radiobutton', ...  
'String','Summer',...  
'Callback',...  
['set(hradio1,"Value",0); '...  
 'set(hradio1(3),"Value",1);'...  
 'colormap(summer)])];  
  
hradio1(4) = uicontrol('Parent',h0, ...  
'Units','points', ...  
'Position',[200 50 80 15], ...  
'Style','radiobutton', ...  
'String','Autumn',...  
'Callback',...  
['set(hradio1,"Value",0); '...  
 'set(hradio1(4),"Value",1);'...  
 'colormap(autumn)])];  
  
hradio1(5) = uicontrol('Parent',h0, ...  
'Units','points', ...  
'Position',[200 35 80 15], ...  
'Style','radiobutton', ...  
'String','Winter',...  
'Callback',...  
['set(hradio1,"Value",0); '...  
 'set(hradio1(5),"Value",1);'...  
 'colormap(winter)])];
```

到此为止，我们的用户界面控件就建立好了，把上述这些程序保存为一个 M 文件，然后在 MATLAB 命令窗口运行此 M 文件就会出现我们设计的用户界面，如图 7-40 所示。

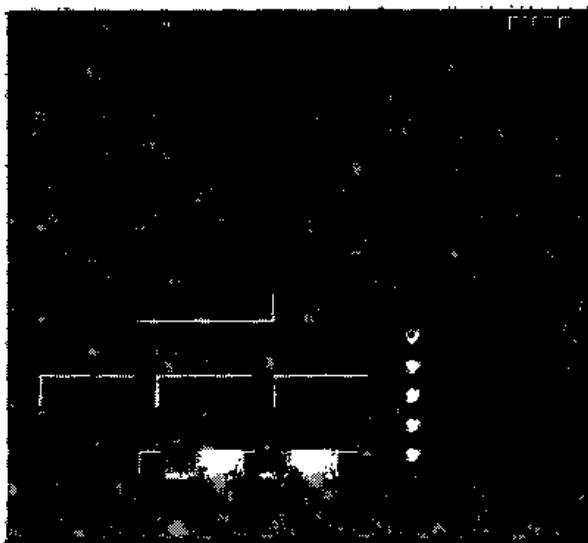


图 7-40 图形用户界面控件

在这个图形窗口中，我们去掉了菜单和工具条，开始运行时不显示坐标轴，但实际上已经指定了坐标轴的位置，绘图时会自动定位到该位置。

首先按下其中的按钮“Membrn”，在设置的坐标轴位置会画出一幅隔膜特征函数图，该图的属性采用缺省值。然后为图形加上标题“隔膜特征函数”，并选择颜色映象“Summer”，调整色彩的亮度使之稍暗一些，最后效果如图 7-41 所示。

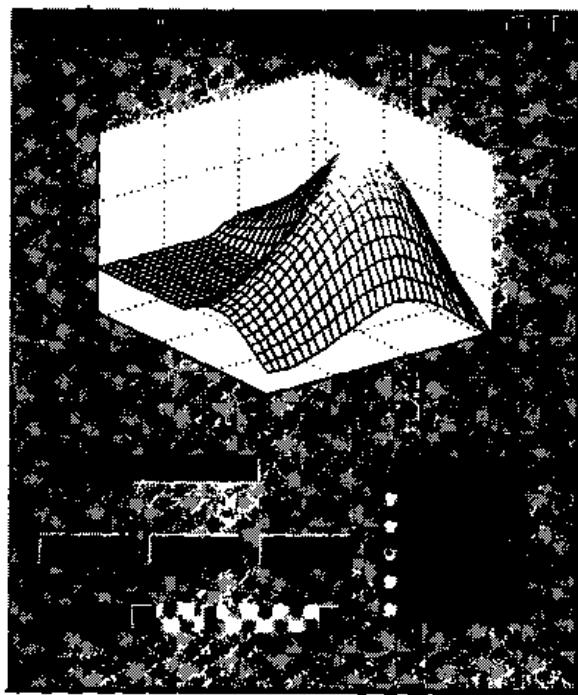


图 7-41 最后的用户界面

## 7.5 对话框

几乎所有的 Windows 程序都要借助于对话框来和用户打交道，所谓对话框实际上就是

一种带有各种控件的用户界面，一般是弹出显示的单独窗口，用来要求或提供信息。通过使用对话框，用户可以通知计算机一些自己的选择，也可以将一些参数赋给计算机，而计算机也可以通过对话框将一些信息反馈给用户。

### 7.5.1 专用对话框的设计

理论上，每个对话框都可以通过编程从最基本的图形窗口开始逐步建立起来，但这样的工作量很大，而且有很多有专门作用的对话框格式都很固定，因而 MATLAB 提供了多种建立专用对话框的函数，这些函数给用户提供了很大的方便，同时也保证了界面的规范统一。

表 7-3 中列出了几种建立专用对话框的函数。

表 7-3 建立专用对话框的函数

函数	建立的对话框类型
errordlg	错误提示对话框
helpdlg	帮助对话框
inputdlg	输入对话框
pagedlg	设置图形位置对话框
printdlg	打印对话框
questdlg	提问对话框
warndlg	警告对话框
msgbox	消息框

下面分别介绍这些函数的用法。

#### 1. errordlg 函数

该函数的调用格式为：

- 1) errordlg：建立一个错误提示对话框，如果已有错误对话框存在，则把该对话框弹出到前面来显示。
- 2) errordlg('errorstring')：建立一个包含字符串“errorstring”的错误对话框。
- 3) errordlg('errorstring','dlgname')：建立名为“dlgname”，包含字符串“errorstring”的错误对话框。
- 4) errordlg('errorstring','dlgname','on')：指定是否代替已经存在的同名对话框，“on”表示把已有的对话框弹出到前面来显示，不建立新的对话框；如果是“off”则新建一个对话框。
- 5) h = errordlg(...): 返回对话框的句柄。

例如，如下命令建立一个如图 7-42 所示的错误提示对话框：

```
errordlg('Invalidation Operation !','My error dialog')
```

按下“OK”按钮该对话框自动关闭，否则将一直存在。

注意：错误提示对话框的大小根据它显示的错误信息字符串长短自动设置，用户不能改变其大小。

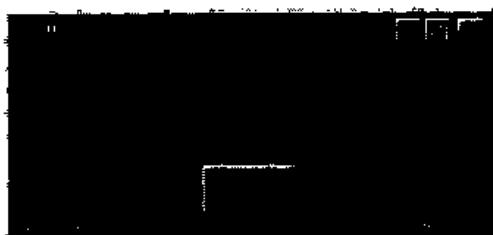


图 7-42 错误提示对话框

## 2. helpdlg 函数

该函数的调用格式为：

- 1) helpdlg：建立一个缺省帮助对话框，如果已有帮助对话框存在，则把该对话框弹出到前面来显示。
- 2) helpdlg('helpstring')：建立一个包含字符串“helpstring”的帮助对话框。
- 3) helpdlg('helpstring','dlname')：建立名为“dlname”，包含字符串“helpstring”的帮助对话框。
- 4) h = helpdlg(...): 返回对话框的句柄。

例如，下面的命令建立如图 7-43 所示的帮助对话框：

```
helpdlg('Use Ctrl+d to draw graph .','My help dialog')
```

按下“OK”按钮对话框消失，否则将一直显示。

» 注意：帮助对话框的大小根据它显示的帮助字符串长短自动设置，用户不能改变其大小。

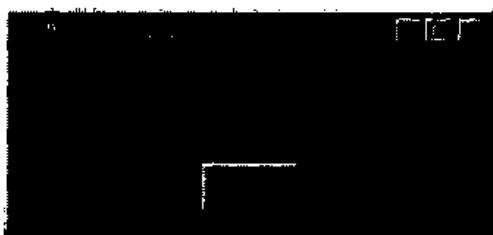


图 7-43 帮助对话框

## 3. inputdlg 函数

该函数的调用格式为：

- 1) answer = inputdlg(prompt)：建立一个输入对话框，用户输入返回到“answer”，“prompt”是提示字符串。
- 2) answer = inputdlg(prompt,title)：带有标题“title”的输入对话框。
- 3) answer = inputdlg(prompt,title,lineNo)：“lineNo”用来指定用户输入值的行数。
- 4) answer = inputdlg(prompt,title,lineNo,defAns)：“defAns”用于指定每个输入项的缺省值，格式为字符串。
- 5) answer = inputdlg(prompt,title,lineNo,defAns,Resize)：“Resize”指示对话框能否被改变大小，取值包括“on”和“off”，分别表示可以改变和不能改变。缺省情况下输入对话框大小不能改变，它是有模式对话框；如果选中了“on”，那么输入对话框自动成为无模式对话框。

所谓有模式对话框，指在对话框被关闭之前，用户无法在程序的其它地方进行工作，比如“Open File”对话框就是典型的有模式对话框；而无模式对话框在保留在屏幕上的同时，用户还可以在程序的其它窗口中进行工作，“Find”对话框是典型的无模式对话框。

如下一组命令建立了如图 7-44 所示的输入对话框：

```
prompt={'Enter Graph Title:','Enter XLabel:','Enter YLabel:'}
```

```
title=['Setup Graph Label ... '];
```

```
line=1;
```

```
def={'Graph1','X axis','Y axis');
```

```
glabel=inputdlg(prompt,title,line,def)
```

在按下该对话框的“OK”或“Cancel”按钮之前，对话框会一直显示在屏幕上，而且由于输入对话框缺省情况下是有模式对话框，这时用户无法进行其它操作。



图 7-44 输入对话框

#### 4. pagedlg 函数

该函数的调用格式为：

1) **pagedlg**: 显示当前图形窗口的页面位置对话框。

2) **pagedlg(fig)**: 显示指定句柄的图形窗口页面位置对话框。

如果当前窗口中没有图形窗口，那么该函数会自动建立一个缺省的图形窗口。

如下命令建立一个已有的图形窗口的页面位置对话框：

```
pagedlg(gcf)
```

结果如图 7-45 所示。

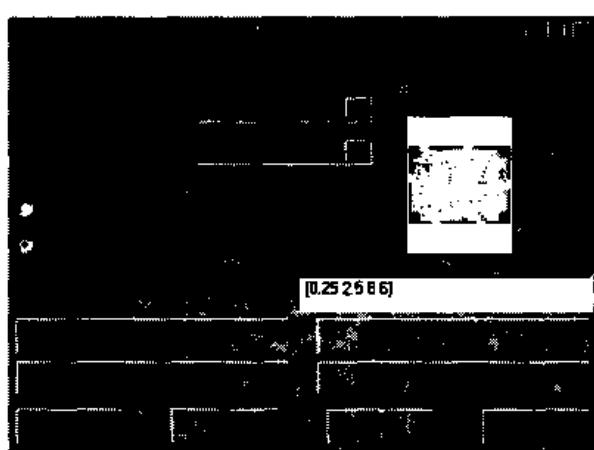


图 7-45 页面位置调整对话框

### 5. printdlg 函数

该函数的调用格式为：

- 1) printdlg：显示当前图形窗口的打印对话框。
- 2) printdlg(fig)：建立指定句柄图形窗口的打印对话框。
- 3) printdlg('-crossplatform',fig)：显示标准的跨平台打印对话框而不是 Windows 平台内置的打印对话框，该对话框不受操作系统平台限制。

打印对话框是有模式对话框。

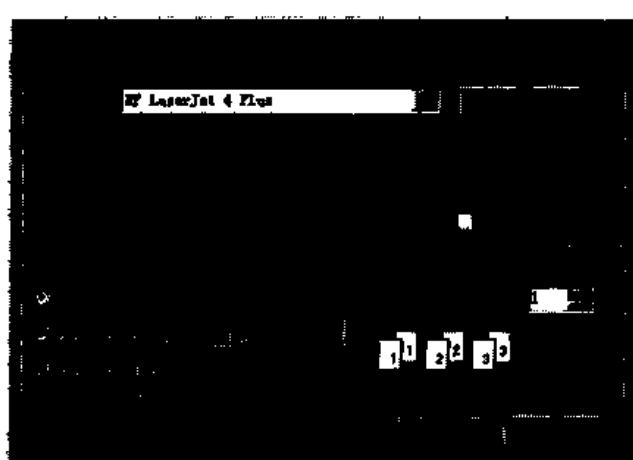


图 7-46 Windows 内置的打印对话框



图 7-47 标准跨平台打印对话框

如下命令建立当前图形窗口的 Windows 打印对话框，如图 7-46 所示：

**printdlg(gcf)**

如下命令建立标准的跨平台打印对话框，如图 7-47 所示：

**printdlg('-crossplatform',gcf)**

### 6. questdlg 函数

该函数的调用方式为：

- 1) button = questdlg('qstring')：建立提问对话框，提示问题 “qstring”。对话框有三个缺省的按钮 “Yes”、“No” 和 “Cancel”。“button” 包含按下的按钮的名字。
- 2) button = questdlg('qstring','title')：为提问对话框加上标题 “title”。
- 3) button = questdlg('qstring','title','default')：在 “default” 中指定当按下回车键时的缺省按钮。
- 4) button = questdlg('qstring','title','str1','str2','default')：建立有两个指定按钮的对话框，按钮的标注为 “str1” 和 “str2”，缺省按钮在 “default” 中指定。
- 5) button = questdlg('qstring','title','str1','str2','str3','default')：建立有三个按钮的对话框，按钮的标注为 “str1”、“str2” 和 “str3”，缺省按钮为 “default”。

注意：提问对话框是有模式对话框。

如下命令建立一个如图 7-48 所示的提问对话框：

```
queststring={'This operation may take a long time!';'Are you sure ?'};
title='Are you sure?';
button=questdlg(queststring,title,'Yes','No','Why','No')
```

按下不同的按钮就会返回不同的按钮名给“button”。

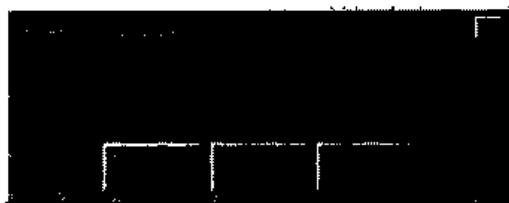


图 7-48 提问对话框

### 7. warndlg 函数

该函数的调用格式如下：

- 1) warndlg: 建立一个缺省的警告对话框，对话框标题为“Warning Dialog”，显示的警告字符串为“This is the default warning string”。
- 2) warndlg('warningstring'): 用“warningstring”指定警告字符串。
- 3) warndlg('warningstring','dlgname'): 用“warningstring”指定警告字符串，用“dlgname”指定对话框标题。
- 4) h = warndlg(...): 返回对话框句柄。

例如，下面的命令建立如图 7-49 所示的警告对话框。

```
warnstr='This operation may take a long time !!!';
title='Warning!!!';
warndlg(warnstr,title)
```

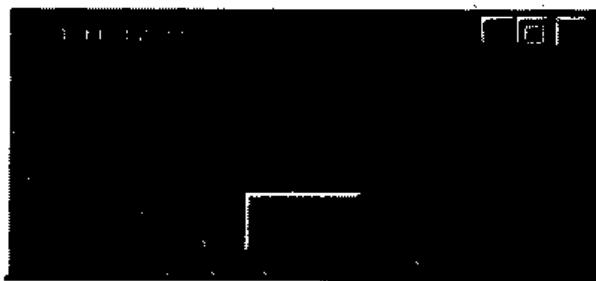


图 7-49 警告对话框

注意提问对话框也可以发出警告信息，图 7-48 中的提问对话框就是这种情况，它和警告对话框的区别是：警告对话框仅是提示信息，不对下一步的操作做出反应，而提问对话框可以对不同按钮被按下的事件进行不同的操作。

### 8. msgbox 函数

该函数的调用格式为：

- 1) msgbox(message): 建立一个消息框，显示“message”中的信息。“message”可以是字符串的向量、矩阵或数组。消息框带有一个“OK”按钮。
- 2) msgbox(message,title): 用“title”指定消息对话框的标题。
- 3) msgbox(message,title,'icon'): 用“icon”指定在消息框中使用的图标。“icon”的取值可以为：“none”、“error”、“help”、“warn”或“custom”，分别对应：不显示图标、错误图标（错误对话框中的图标）、帮助图标（帮助对话框中的图标）、警告图标（警告对话框中的图标）。

中的图标)或自定义图标。缺省情况是“none”。

4) `msgbox(message,title,'custom',iconData,iconCmap)`: 自定义一个自己的图标，“iconData”是定义图标的数据，“iconCmap”是图标采用的颜色映象。

5) `msgbox(...,'createMode')`: 指定消息框是有模式还是无模式对话框。“createMode”的取值包括“modal”(有模式)和“non-modal”(无模式)。不指定参数“createMode”时消息框是无模式对话框。

6) `h = msgbox(...)`: 返回消息框的句柄。

如下命令建立如图 7-50 所示的消息框:

```
msgbox('This is my test message box.','Test msgbox','help')
```

按下“OK”按钮消息框消失。

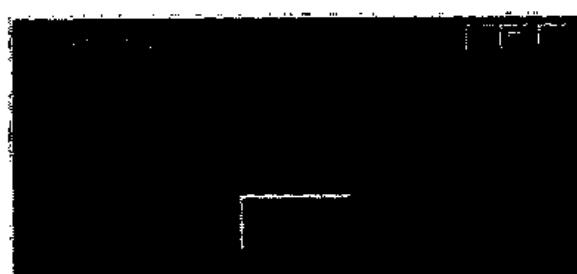


图 7-50 消息框

### 7.5.2 标准对话框

具有用户界面的应用软件一般都有 File 菜单，该菜单下一般又含有 Open 和 Save、Save as 选项，在很多软件中，这些选项都有相同样式的对话框，这类对话框都是 Windows 下内置的资源，只要使用相应的函数就能调用。

MATLAB 也为用户提供了调用标准对话框的函数，通过它们，用户可以很方便地为自己编制的菜单选项加上标准对话框作为回调程序。

MATLAB 提供的这类函数如表 7-4 所示。

表 7-4 调用标准对话框的函数

函数	建立的标准对话框类型
<code>uigetfile</code>	打开已有的文件对话框
<code>uiputfile</code>	保存文件对话框
<code>uisetfont</code>	设置字体对话框
<code>uisetcolor</code>	设置颜色对话框

下面我们将介绍如何用这几个函数调用相应的对话框。

#### 1. `uigetfile` 函数

该函数的调用格式为:

1) `uigetfile`: 显示一个打开文件对话框，该对话框自动列出当前路径下的目录和具有缺省扩展名“.m”的文件。

2) `uigetfile('FilterSpec')`: “FilterSpec”指定初始时显示的文件名或文件类型，可以使用通配符“\*”。

- 3) `uigetfile('FilterSpec','DialogTitle')`: 对话框的标题由“DialogTitle”指定。
- 4) `uigetfile('FilterSpec','DialogTitle',x,y)`: 指定对话框在屏幕上的位置, “x, y”分别是到屏幕左边和上边的距离, 单位是像素。
- 5) `[fname, pname] = uigetfile(...)`: 在按下对话框中的执行按钮(“打开”)后, 返回选择的文件名和路径, 分别保存到“fname”和“pname”中。如果按下取消按钮或发生错误, 则返回值都是0。

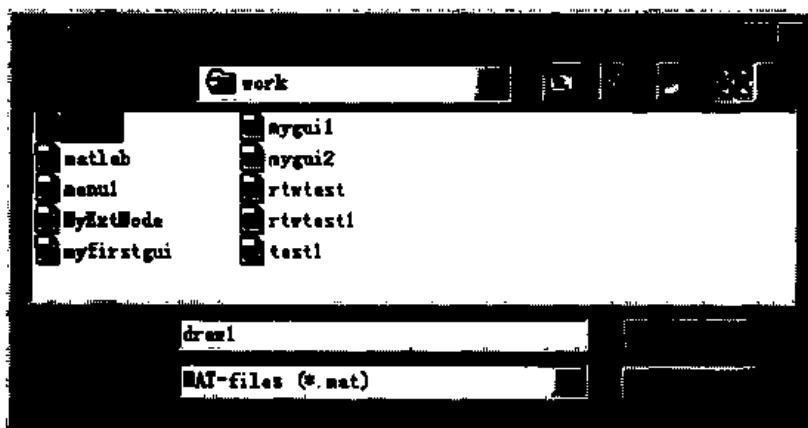


图 7-51 打开文件对话框

例如, 如下命令调用打开文件对话框来打开一个数据文件:

```
[name,path] = uigetfile('.mat','打开数据文件')
```

显示的对话框如图 7-51 所示, 在该对话框中选择“draw1”文件, 则返回值为:

```
name =
    draw1.mat
path =
    D:\MATLABR11\work\
```

注意: 得到文件名和路径后, 并没有读取文件中的数据, 要真正读出数据必须要在 `uigetfile` 函数后使用相关的文件输入输出函数, 参见下一节。

## 2. uiputfile 函数

该函数的调用格式为:

- 1) `uiputfile`: 显示一个保存文件对话框, 该对话框自动列出当前路径下的目录和具有缺省扩展名“.m”的文件。
- 2) `uiputfile('InitFile')`: “InitFile”指定初始时显示的文件名或文件类型, 可以使用通配符“\*”。
- 3) `uiputfile('InitFile','DialogTitle')`: 对话框的标题由“DialogTitle”指定。
- 4) `uiputfile('InitFile','DialogTitle',x,y)`: 指定对话框在屏幕上的位置, “x, y”分别是到屏幕左边和上边的距离, 单位是像素。
- 5) `[fname, pname] = uiputfile(...)`: 在按下对话框中的执行按钮(“保存”)后, 返回选择的文件名和路径, 分别保存到“fname”和“pname”中。如果按下取消按钮或发生错误, 则返回值都是0。

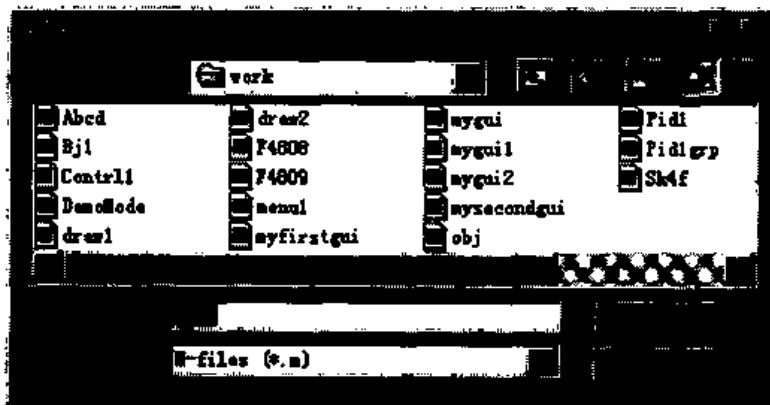


图 7-52 保存文件对话框

例如，如下命令调用保存文件对话框来保存一个文件：

```
[name,path] = uiputfile('test.m','保存程序')
```

显示的对话框如图 7-52 所示，对话框中出现了设置的保存文件名“test.m”，按下“保存”按钮保存该文件，并返回该文件的文件名和路径：

```
name =
    test.m
path =
    D:\MATLABR11\work\
```

注意：实际上现在“test.m”还是一个空文件，并没有保存数据，要真正把数据写到文件中必须要在 uiputfile 函数后使用相关的文件输入输出函数，参见 7.6 节。

### 3. uisetcolor 函数

该函数的调用格式为：

```
c = uisetcolor(handle 或 color, 'DialogTitle');
```

显示一个对话框让用户选择或自定义颜色。“handle”指图形对象的句柄，该对象必须有“Color”属性；“color”是 RGB 颜色的三元素向量，用于指定一种缺省的颜色，初始化对话框。“DialogTitle”指定了对话框标题。返回值“c”是所选颜色的 RGB 值。

例如，如下命令调用如图 7-53 所示的颜色设置对话框，并指定初始颜色为黄色：

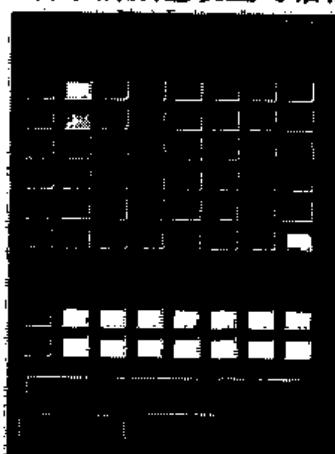


图 7-53 颜色设置对话框

**uisetcolor([1 1 0], '选择颜色')**

如果我们选择了其中的红色，那么返回值将是：

ans =

$$1 \quad 0 \quad 0$$

当然还可以按下“规定自定义颜色”按钮定义更多的颜色。

#### 4. `uiSetFont` 函数

函数 `uisetfont` 用于改变文本、坐标轴或用户界面控件的字体属性，包括字体的种类、单位、大小、粗细和角度。该函数返回一个结构，包括字体的属性和相应的值。

该函数的调用格式为：

- (1) `uisetfont`: 显示字体属性设置对话框, 返回所选的字体属性。
  - (2) `uisetfont(h)`: 用句柄为 `h` 的对象的字体属性值初始化对话框中的字体属性。设置后的字体属性也应用到这个对象。
  - (3) `uisetfont(S)`: 用指定的结构 “`S`” 中的字体属性值初始化对话框中的字体属性。结构 “`S`” 中必须定义了一个或多个字体属性的合法值, 结构的域名和属性名 (包括 `FontName`, `FontUnits`, `FontSize`, `FontWeight` 和 `FontAngle`) 必须严格相同。
  - (4) `uisetfont('DialogTitle')`: 对话框的标题由 “`DialogTitle`” 指定。

例如，如下命令设置当前图形对象的字体：

**uisetfont(gco,'设置图形对象字体')**

字体设置对话框如图 7-54 所示。我们选择了合适的字体属性值后，函数的返回值为：

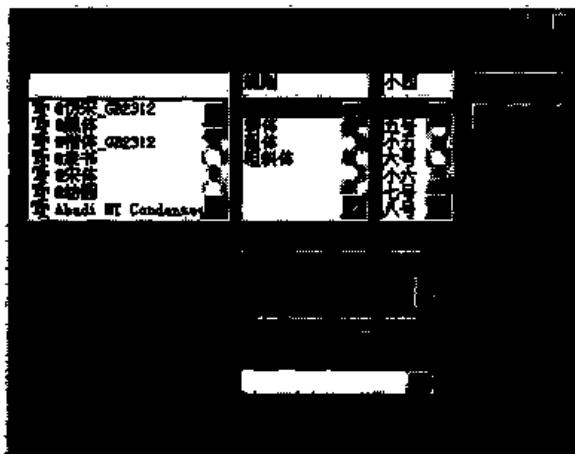


图 7-54 字体设置对话框

```
ans =  
    FontName: '隶书'  
    FontUnits: 'points'  
    FontSize: 10  
    FontWeight: 'normal'  
    FontAngle: 'italic'
```

这是结构的域名和相应的值，在自己定义结构时也要按照这样的格式。

## 7.6 低级文件 I/O

文件的输入输出在编程中是经常要碰到的问题，要对保存在文件中数据进行分析，分析结果要保存到文件中，这都需要文件的输入输出操作。当然，对于简单的情况可以用 MATLAB 的 `load` 和 `save` 命令来完成，但这两条命令的局限性很大，只能读写 MATLAB 本身的数据格式。

本节我们将介绍 MATLAB 提供的能够读写一般类型文件的几个函数，这几个函数位于 MATLAB 根目录下的`\toolbox\MATLABIO\FUN` 目录。表 7-5 列出了这些低级文件 I/O 函数。

表 7-5 低级文件 I/O 函数

函数名	功能
<code>fopen</code>	打开文件
<code>fclose</code>	关闭文件
<code>fread</code>	读文件中的二进制数据
<code>fwrite</code>	向文件中写入二进制数据
<code>scanf</code>	从文件中读有格式数据
<code>fprintf</code>	向文件中写有格式数据
<code>fgetl</code>	读文件的下一行（作为字符串格式），不包括行结束符
<code>fgets</code>	读文件的下一行（作为字符串格式），包括行结束符
<code>feof</code>	测试文件结尾
<code>fseek</code>	设置文件的位置指针
<code>ftell</code>	获取文件的位置指针
<code>frewind</code>	重置文件指针到文件头

### 7.6.1 打开和关闭文件

上一节介绍了如何用函数 `uigetfile` 和 `uiputfile` 来调用打开文件对话框和保存文件对话框。实际上这两个对话框的作用是找到指定文件的位置，它们的返回值也只是文件名和路径，并没有真正打开该文件。具体的打开关闭文件和读写文件操作只能由文件的输入输出函数来完成。

#### 1. 打开文件

打开文件的函数是 `fopen`，其调用格式为：

1) `fid = fopen(filename, permission)`: 打开文件名为 `filename` 的文件，`permission` 用于指定打开文件的模式，返回文件指针给 `fid`。如果文件不在 MATLAB 的搜索路径范围内，那么需要在 `filename` 中同时指定文件的路径。

2) `[fid, message] = fopen(filename, permission)`: 返回打开文件的信息。

3) `Fids = fopen('all')`: 返回一个行向量，包括所有已打开文件的指针，但不包括标准输出（指屏幕，文件指针总是为 1）和标准错误提示（文件指针总是为 2）。向量的元素个数等于打开文件的个数。

4) `Filename = fopen(fid)`: 根据打开的文件指针返回文件名, 如果文件不在 MATLAB 当前工作路径下, 则文件名中包括路径。

函数 `fopen` 的参数 `permission` 指打开文件的模式, 它的值可以为:

- 'r': 打开文件读。
- 'r+'. 打开文件读写。
- 'w': 删除已经存在的文件的内容或建立一个新文件, 并打开文件写。
- 'w+'. 删除已经存在的文件的内容或建立一个新文件, 并打开文件读写。
- 'a': 建立并打开一个新文件或打开一个已存在的文件写, 在文件的结尾处添加内容。
- 'a+'. 建立并打开一个新文件或打开一个已存在的文件读写, 在文件的结尾处添加内容。

文件可以以二进制代码的格式打开, 也可以以文本代码的格式打开。在文本格式下, 行的分隔符在从文件读出时自动删除, 在写入文件时自动添加; 而二进制代码格式不删除或添加分隔符。

对于 PC 机系统, 必须在用 `fopen` 函数打开文件时标明: 打开文本格式的文件, 在 `permission` 参数指定打开模式的字符之后添加字母“t”, 打开二进制代码格式的文件, 在 `permission` 参数指定打开模式的字符之后添加字母“b”。

成功打开一个文件后, 返回的文件指针是一个非负的整数, 如果打开文件失败, 则文件指针的值为-1。

例如, 下面的语句用来打开一个文件读, 并返回指针和信息:

```
[fid, message]=fopen('mytest.mat', 'r')
```

如果成功打开了这个文件, 则将显示:

```
fid =
3
message =
"
```

如果打开文件失败, 则将显示:

```
fid =
-1
message =
Cannot open file. Existence? Permissions? Memory? ...
```

## 2. 关闭文件

打开的文件在进行读写操作后一般需要立即关闭, 删除文件指针, 以免未关闭文件的指针造成混乱。

关闭文件的函数是 `fclose`, 它的用法是:

- 1) `status = fclose(fid)`: 关闭指针所指的文件, 成功关闭返回 0, 关闭失败则返回-1。
- 2) `status = fclose('all')`: 关闭所有打开的文件 (不包括标准输出和标准错误提示文件), 成功关闭返回 0, 关闭失败则返回-1。

例如, 前面例子中打开的文件指针为 “fid”, 值为 3, 那么以下两种方式都能关闭该文

件：

```
fclose(fid);
fclose(3);
```

关闭后返回 0。

### 7.6.2 读写二进制数据

函数 **fread** 和 **fwrite** 分别用于读、写文件中的二进制数据。

#### 1. 读数据

读二进制数据的函数 **fread** 用法为：

**[A, count] = fread(fid, size, precision, skip):** 从文件指针 **fid** 所指的文件读取二进制数据，并写到矩阵 **A** 中。可选参数 **count** 返回成功读取的元素个数。

可选参数 **size** 指定需要读的数据个数，如果不指定这个参数，那么 **fread** 将读到文件的结尾。参数 **size** 的取值包括如下几种：

- **n:** 读 **n** 个数据到一个列向量。
- **inf:** 一直读到文件的结尾，读出的数据放到一个列向量中。
- **[m, n]:** 读出的数据个数等于一个  $m \times n$  矩阵中的元素总数，读出的数据按照列的顺序排列，如果文件中没有足够的数据，则用 0 来填补剩下的部分。

可选参数 **precision** 是一个字符串，用来指定读取数据的精度，也就是数据类型，如'**int**'（整数），'**float**'（浮点数），'**char**'（字符）等。表 7-6 列出了和 C 语言兼容的可移植的数据类型表示方法，表 7-7 是限于 MATLAB 平台的不可移植的数据类型表示方法，表中列出了相应的 C 语言数据类型。

可选参数 **skip** 指定在每读取一个数据之后跳过的字节数。指定了 **skip** 的值以后，函数 **fread** 读取一个数据，跳过指定的字节数，然后再读取一个数据……。该参数常用来在一批数据中等间隔抽取一部分数据。

如下几条命令读取一个二进制文件中的数据：

```
fid = fopen('test1.mat', 'rb');
A1 = fread(fid, 5, 'int8');
A2 = fread(fid, 5, 'char', 2);
```

如果文件 **test1.mat** 中的数据是从 1 到 100 的连续整数，那么返回结果为

**A1 =**

```
1
2
3
4
5
```

**A2 =**

```
6
9
12
```

15

18

从返回结果我们可以体会出 fread 函数中各个参数的含义。

注意：第二个 fread 语句读数据的位置是接着前一个 fread 语句读数据的位置之后，不是从文件开始的位置读。

表 7-6 可移植的数据类型

MATLAB 语言	C 语言	含义
'schar'	'signed char'	有符号字符，8位
'uchar'	'unsigned char'	无符号字符，8位
'int8'	'integer*1'	整数，8位
'int16'	'integer*2'	整数，16位
'int32'	'integer*4'	整数，32位
'int64'	'integer*8'	整数，64位
'uint8'	'integer*1'	无符号整数，8位
'uint16'	'integer*2'	无符号整数，16位
'uint32'	'integer*4'	无符号整数，32位
'uint64'	'integer*8'	无符号整数，64位
'float32'	'real*4'	浮点数，32位
'float64'	'real*8'	浮点数，64位
'double'	'real*8'	浮点数，64位

表 7-7 限于 MATLAB 平台的数据类型

MATLAB 语言	C 语言	含义
'char'	'char*1'	字符，8位
'short'	'short'	整数，16位
'int'	'int'	整数，32位
'long'	'long'	整数，32位或64位
'ushort'	'unsigned short'	无符号整数，16位
'uint'	'unsigned int'	无符号整数，32位
'ulong'	'unsigned long'	无符号整数，32位或64位
'float'	'float'	浮点数，32位

## 2. 写数据

写二进制数据的函数 fwrite 用法如下：

count = fwrite(fid, A, precision, skip): 把矩阵 A 中的数据写到文件指针 fid 所指的文件中，返回值 count 为成功写入数据的个数。

参数 precision 和 skip 的含义与 fread 函数中的参数相同。

我们举一个例子，综合说明函数 fwrite 和 fread 的用法，程序如下：

```
t = 1:100;

fid1 = fopen('test.mat','wb+');
count = fwrite(fid1,t,'int');
fclose(fid1);

fid2 = fopen('test.mat','rb')
A1 = fread(fid2,[5,4],'int')
A2 = fread(fid2,5,'int',4)
fclose(fid2);
```

这段程序首先把 1 到 100 的整数以 int 格式（32 位整数）保存到二进制文件 test.mat 中，然后从文件中读出数据，先读一个 5 行 4 列的矩阵（共 20 个数），再从第 21 个数起每隔 4 个字节（注意：每个字节 8 位，四个字节 32 位，正好是一个 int 格式的数据）读一个数，共读取 5 个数。

- » 注意：在向文件中写入数据后，必须首先关闭文件，然后再打开读数据，否则，如果写完数据后立刻读数据，那么实际读数据的位置将从写入的最后一一个数据之后开始，而不会从文件头开始。

上一段程序的执行结果为

```
A1 =
    1     6    11    16
    2     7    12    17
    3     8    13    18
    4     9    14    19
    5    10    15    20
```

```
A2 =
    21
    23
    25
    27
    29
```

### 7.6.3 有格式文件

读写有格式文件的函数是 fscanf 和 fprintf，另外两个函数 fgets 和 fgetl 用于读文件的行，下面我们分别介绍。

#### 1. fscanf 函数

函数 fscanf 的调用格式为：

[A, count] = fscanf(fid, format, size)：从文件指针 fid 指向的文件中读取数据并返回给矩阵 A，count 为成功读取的数据个数。

参数 size 和函数 fread 中的用法类似，包括以下几种取值方式：

- n：读 n 个数据到一个列向量。
- inf：一直读到文件的结尾，读出的数据放到一个列向量中。
- [m, n]：读出的数据个数等于一个  $m \times n$  矩阵中的元素总数，读出的数据按照列的顺序填充矩阵，其中 n 可以是无穷大 (inf)，表示读到文件结尾，而 m 不可以。

参数 format 指定读取数据格式，指定的格式必须和文件中的数据格式相同，否则读取的数据可能会出现错误。

MATLAB 的格式字符串和 C 语言的格式字符串基本相同，都以符号“%”开头，以针对各种数据格式的变换符号结尾。如“%s”、“%f”、“%ld”、“%10.3e”都是合法的 MATLAB 格式字符串。

MATLAB 下的格式变换符号如表 7-8 所示。

表 7-8 格式变换符号

符号	含义
%c	字符
%d	十进制数
%e	指数格式的浮点数
%f	一般格式的浮点数
%g	%e 和 %f 的紧凑格式
%i	有符号整数
%o	有符号八进制整数
%s	字符串
%u	无符号十进制整数
%x	有符号十六进制整数

在 MATLAB 格式字符串中的“%”之后，格式变换符号之前的部分可以有如下三种形式：

- 星号 (\*): 跳过格式相匹配的数据，这些数据不读到输出矩阵中。
- 阿拉伯数字：读出数据的最大位数。
- 字母：如 h 表示 short (短型)，“l”表示 long (长型)。组合起来 “%ld” 表示长整型。

例如，如下程序用来读取一个 3 列的数据文件，该文件头有 3 个说明字符串：

```
fid = fopen('test.dat', );
[A1,Count1]=fscanf(fid,'%s',3);
[A2,Count2]=fscanf(fid,'%e, %e, %e', [3 inf]);
fclose(fid);
```

读出到矩阵 A2 中的数据也将是 3 列。

## 2. fprintf 函数

函数 fprintf 的调用格式为：

**count = fprintf(fid, format, A, ...):** 把矩阵 A 中的数据按照 format 规定的格式写到 fid 指向的文件中。返回成功写入文件的字节数给 count。

**fprintf(format, A, ...):** 把矩阵写到标准输出设备——屏幕。

函数 fprintf 的参数和 fscanf 类似，只是函数 fprintf 比函数 fscanf 的参数 format 更加复杂，可以从很多角度更确切地定义写入文件的数据格式。

函数 fprintf 的参数 format 可以包括转义字符，如表 7-9 所示。而且比函数 fscanf 中增加了几种更确切的格式变换符号：

- %E: 在指数格式表示时使用大写字母 E。
  - %G: 用大写字母 E。
  - %X: 用大写字母 A~F 表示相应的十六进制数。
- 而且，在“%”后，设置位数的数字前可以使用如下几个特殊符号指定特殊格式：
- 减号 (-): 使得写入的数据左对齐，如“%-10.4e”。
  - 加号 (+): 无论数据是正或是负，都显示其符号，如“%+10.4e”。
  - 零 (0): 用“0”补齐位数，如“%010.4e”。

表 7-9 转义字符

字 符	含 义
\b	退后一格
\f	换页
\n	换行
\r	回车
\t	水平制表符
\\"	反斜线
\' or "(two single quotes)"	单引号
\%	百分号

例如，下面一段程序先在一个文件中按一定的格式写入数据，然后按照相应的格式读出数据：

```
t=1:5;
s1=sin(t);
s2=cos(t);
s=[t;s1;s2];

fid1=fopen('test.dat','wt');
fprintf(fid1,'This is a Formated file\n');
fprintf(fid1,'%4d %+12.5E %+12.5E\n',s);
fclose(fid1);

fid2=fopen('test.dat','rt');
```

```
[A1,count1]=fscanf(fid2,'%s',5);
[A2,count2]=fscanf(fid2,'%4d %e %e',[3,inf]);
A2=A2';
fclose(fid2);
```

程序执行后，生成的文件 test.dat 内容如下：

This is a Formated file

```
1 +8.41471E-001 +5.40302E-001
2 +9.09297E-001 -4.16147E-001
3 +1.41120E-001 -9.89992E-001
4 -7.56802E-001 -6.53644E-001
5 -9.58924E-001 +2.83662E-001
```

从该文件中读出的矩阵 A1 和 A2 内容如下：

A1 =

This is a Formated file

A2 =

```
1.0000 0.8415 0.5403
2.0000 0.9093 -0.4161
3.0000 0.1411 -0.9900
4.0000 -0.7568 -0.6536
5.0000 -0.9589 0.2837
```

我们看到读出来的 A1 合成了一个字符串，这个问题可以用后面讲到的两个函数 fgets 和 fgetl 来解决。

### 3. 函数 fgetl 和 fgets

函数 fgetl 和 fgets 用来读取文件的下一行，它们的调用格式如下所示：

- 1) line = fgetl(fid): 把 fid 指向文件的下一行以字符串的形式返回给 line，不包括行的结束符，如果遇到文件结尾则返回-1。
- 2) line = fgets(fid): 把 fid 指向文件的下一行以字符串的形式返回给 line，包括行的结束符，如果遇到文件结尾则返回-1。
- 3) line = fgets(fid, nchar): 限制最多返回的字符个数为 nchar 个。

例如，对于前面生成的文件 test.dat，我们用这两个函数来读取它的行：

```
fid=fopen('test.dat','rt');
for i=1:3
    line=fgets(fid);
end
fclose(fid);
```

读出的前三行内容为：

line =

line =

```
This is a Formated file
```

```
line =
1 +8.41471E-001 +5.40302E-001
```

其中，第一行为空行，用 fgets 函数返回一个行结束符，因而 line 这时并不是空向量。如果用函数 fgetl 代替 fgets，那么第一行的返回结果将为：

```
line =
Empty string: 1-by-0
```

是一个空的字符串向量。

在很多情况下，经常把上面讲过的这四个函数结合在一起使用，这样就可以准确的读出比较复杂的文件格式了。

#### 7.6.4 文件位置指针

每打开一个文件时，MATLAB 就为该文件分配一个位置指针，按照这个指针所指的位置来进行读写数据的操作。每读写完一个数据后，文件位置指针就向后移动一个数据所占的字节数。

例如，下面的一段程序把含有 10 个元素的向量 t 以 16 位整数的格式写入一个文件：

```
t = 1:10;
fid = fopen('test.mat','wb+');
count = fwrite(fid, t, 'int16');
```

这个操作完成后，文件位置指针将位于文件尾部。如果下面继续进行读写数据的操作，都是从文件的尾部继续。

如果我们需要移动位置指针，需要借助函数 fseek，该函数的调用格式为：

`status = fseek(fid, offset, origin)`: 把 fid 所指的文件的位置指针移动由参数 offset 指定的字节数。如果 offset 是正数，则向文件尾部移动，如果 offset 是负数，则向文件头的方向移动。操作成功后返回给 status 的值为 0，失败则返回-1。

参数 origin 用来指定移动位置指针的参考起点，它可以取值为：

'bof' 或 -1: 文件的开头。

'cof' 或 0: 文件的当前位置。

'eof' 或 1: 文件的结尾。

例如，下面的语句把上面的文件的位置指针从文件结尾向前移动 6 个字节：

```
fseek(fid, -6, 'eof');
```

当前指针的位置可以用函数 ftell 来获取：

```
pos=ftell(fid)
```

结果为：

```
pos =
```

14

即当前的位置指针位于第 14 个字节之后，由于数据类型是 16 为整型，一个数据占两个字节，因而位置指针也就是位于第 7 个数据 (7) 之后。现在用函数 fread 来读文件当前位置的数据，进而检验位置指针的位置：

```
A = fread(fid,1,'int16')
```

结果：

```
A =
```

```
8
```

这证明位置指针正是位于 7 之后。

除了函数 `fseek` 和 `ftell` 外，还有两个常用的和位置指针有关的函数：`frewind` 和 `feof`。它们的用法分别为：

`frewind(fid)`：把 `fid` 指向的文件的位置指针移到文件的开头，这相当于 `fseek` 函数的如下调用：

```
fseek(fid, 0, 'bof');
```

函数 `feof` 的用法为：

`eofstat = feof(fid)`：检查 `fid` 指向的文件是否设置了文件尾指示符，如果设置了，则返回 1，否则返回 0。文件尾指示符当文件不再有输入时设置。

# 第 8 章 Simulink 入门

## 8.1 Simulink 概述

作为 MATLAB 的重要组成部分, Simulink 具有相对独立的功能和使用方法。确切的说, 它是对动态系统进行建模、仿真和分析的一个软件包。它支持线性和非线性系统、连续时间系统、离散时间系统、连续和离散混合系统, 而且系统可以是多进程的。现在, Simulink 的版本已经发展到 3.0 (MATLAB 5.3), 功能更加强大, 使用也越来越方便。

Simulink 提供了友好的图形用户界面 (GUI), 模型由模块组成的框图来表示, 用户建模通过简单的单击和拖动鼠标的动作就能完成。Simulink 的模块库为用户提供了多种多样的功能模块, 这是一笔非常丰富的资源。其中基本功能模块有连续系统 (Continuous)、离散系统 (Discrete)、非线性系统 (Nonlinear) 几类基本系统构成模块, 还包括连接、运算类模块: 函数与表 (Functions & Tables)、数学运算模块 (Math)、信号与系统 (Signals & Systems)。而输入源模块 (Sources) 和接收模块 (Sinks) 则为模型仿真提供了信号源和结果输出设备。便于用户对模型进行仿真和分析。

模型建立好后, 可以直接对它进行仿真分析。可以选择合适的输入源模块 (如: 正弦波 (Sine Wave)) 做信号输入, 用适当的接收模块 (如: 示波器 (Scope)) 观察系统响应、分析系统特性, 仿真结果输出到接收模块上。如果仿真结果不符合要求, 则可以修改系统模型的参数, 继续进行仿真分析。

后面几节我们将为大家介绍 Simulink 的具体功能和操作。

## 8.2 Simulink 基本操作

### 8.2.1 运行 Simulink

运行 Simulink 的方式有三种:

- 1) 在 MATLAB 的命令窗口直接键入 “simulink”
- 2) 单击 MATLAB 工具条上的 Simulink 图标 
- 3) 在 MATLAB 菜单上选择 File→New→Model

运行后会显示如图 8-1 所示的 Simulink 模块库浏览器, 方式(3)还同时显示如图 8-2 所示的新建模型窗口。单击图 8-1 工具条左边的图标  (建立新模型), 也会弹出如图 8-2 的窗口。

打开已经存在的模型文件也有几种方式:

- 1) 在 MATLAB 命令窗口直接键入模型文件名 (不要加扩展名 “.mdl”), 这要求该文件在当前的路径范围内;

- 2) 在 MATLAB 菜单上选择 File→Open;  
 3) 单击图 8-2 的工具条上的图标 。

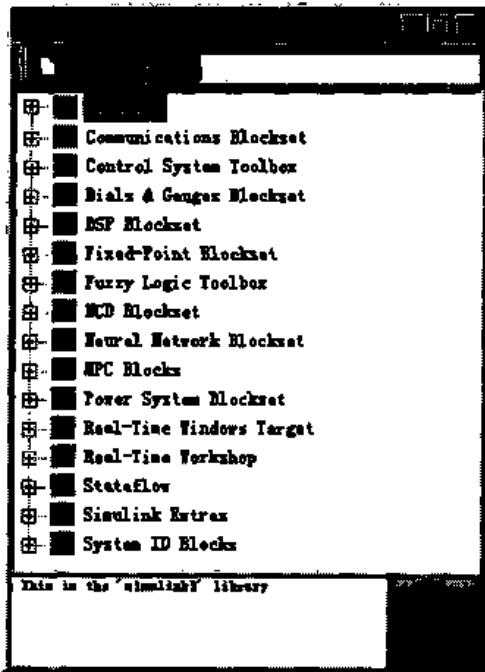


图 8-1 Simulink 模块库浏览器

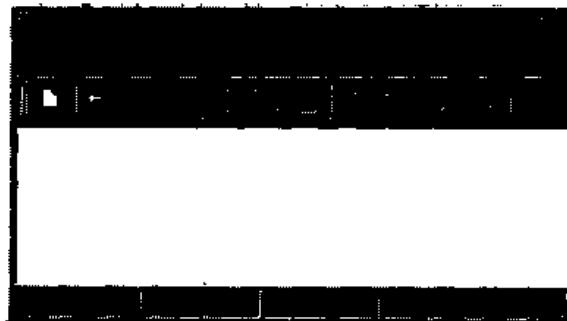


图 8-2 新建模型窗口

注意: Simulink 中的几类窗口: 模块库浏览器、模块库、模型和仿真结果输出窗口的性质相对独立, 不属于 MATLAB 的图形对象, 不能用句柄图形 (Handle Graphics) 的命令设置或修改这类窗口的属性。

和其它很多 Windows 窗口的特点类似, 在 Simulink 的模型窗口和模块库窗口的 View 菜单下选择或取消 Toolbar 和 Status Bar 选项, 就可以显示或去掉工具条和状态条。在进行仿真的过程中, 模型窗口的状态条会显示仿真状态、仿真进度、仿真时间等相关信息。

### 8.2.2 Simulink 模块的操作

模块是建立 Simulink 模型的基本单元。用适当的方式把各种模块连接在一起就能够建立动态系统的模型。本节将介绍对模块的操作方法。

#### 1. 选取模块

当选取单个模块时, 只要用鼠标在模块上单击即可, 这时模块的角上出现黑色的小方块。选取多个模块时, 在所有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, 当虚框包住了要选的所有模块后, 放开鼠标左键, 这时在所有被选模块的角上都会出现小黑方块, 表示模块都被选中了。此过程如图 8-3 所示。

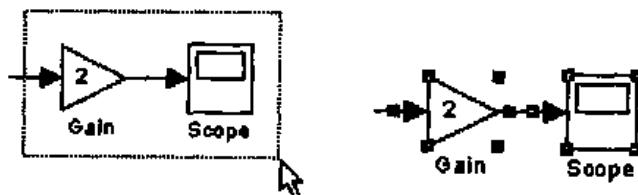


图 8-3 选取多个模块

## 2. 复制、删除模块

### (1) 在不同的窗口之间复制

当我们建立模型时，需要从模块库窗口或者已经存在的模型文件窗口把需要的模块复制到新建模型文件的窗口。要对已经存在的模型进行编辑时，有时也需要从模块库窗口或另一个已经存在的模型文件窗口复制模块。

最简单的办法是用鼠标左键点住要复制的模块（首先打开源模块和目标模块所在的窗口），按住左键移动鼠标到相应窗口（不用按住 Ctrl 键），然后释放，该模块就会被复制过来，而源模块不会被删除。

当然还可以用 Edit 菜单下的 Copy 和 Paste 命令来完成复制：先选定要复制的模块，选择 Edit 菜单下的 Copy 命令，到目标窗口的 Edit 菜单下选择 Paste 命令。

### (2) 在同一模型窗口内复制

有时一个模型需要多个相同的模块，这时的复制方法如下：

用鼠标左键点住要复制的模块，按住左键移动鼠标，同时按下 Ctrl 键，到适当位置放开鼠标，该模块就被复制到当前位置。更简单的方法是按住鼠标右键（不按 Ctrl 键）移动鼠标。

另一种方法是先选定要复制的模块，选择 Edit 菜单下的 Copy 命令，然后选择 Paste 命令。这时我们会发现复制出的模块名称在原名称的基础上又加了编号，这是 Simulink 的约定：每个模型中的模块和名称是一一对应的，相同的模块或不同的模块都不能用同一个名字。

### (3) 删除模块

选定模块，选择 Edit 菜单下的 Cut（删除到剪贴板）或 Clear（彻底删除）命令。或者在模块上单击鼠标右键，在弹出菜单上选择 Cut 或 Clear 命令。

## 3. 模块的参数和属性设置

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按鼠标右键并在弹出的上下文菜单中选择“Block Parameters”就会弹出参数设置对话框，图 8-4 所示的是正弦波模块的参数设置对话框，用户可以设置它的幅值、频率、相位、采样时间等参数。模块参数还可以用 set\_param 命令设置，这在后面将会讲到。

每个模块都有一个内容相同的属性（Properties）设置对话框，如图 8-5 所示。它的打开方式是在模块上按鼠标右键并在弹出的上下文菜单中选择“Block Properties”。该对话框包括如下几项内容：

### (1) 说明 (Description)

是对该模块在模型中用法的注释。

### (2) 优先级 (Priority)

规定该模块在模型中相对于其它模块执行的优先顺序。优先级的数值必须是整数或不输入数值，这时系统会自动选取合适的优先级。优先级的数值越小（可以是负整数），优先级越高。

### (3) 标记 (Tag)

用户为模块添加的文本格式的标记。

### (4) 调用函数 (Open function)

用户双击该模块时调用的 MATLAB 函数。

#### (5) 属性格式字符串 (Attributes format string)

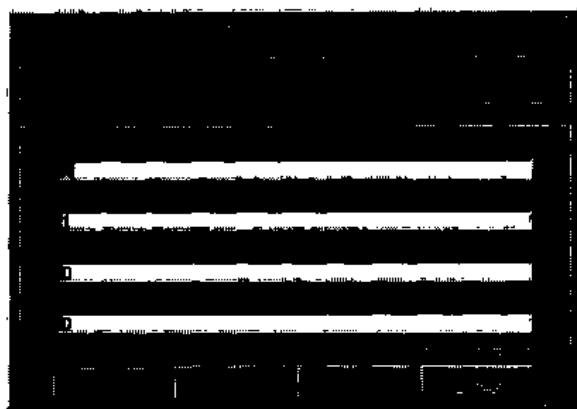


图 8-4 模块参数设置对话框

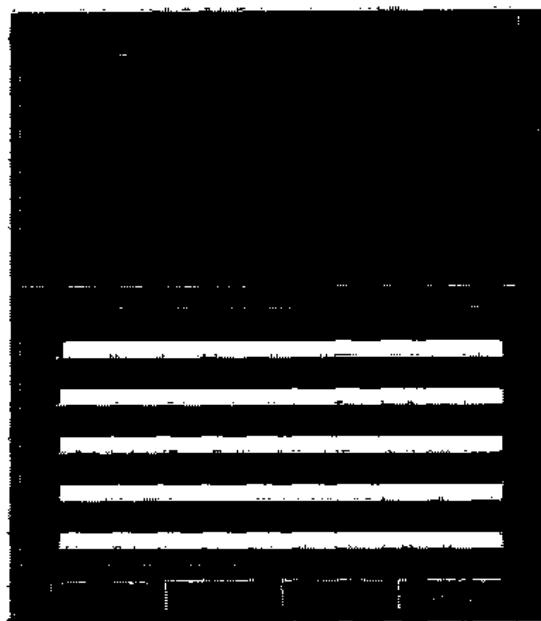


图 8-5 模块属性设置对话框

指定在该模块的图标下显示模块的哪个参数，以什么格式显示。属性格式字符串由任意的文本字符串加嵌入式参数名组成。例如，对一个传递函数模块指定如下的属性格式字符串：

**优先级=%<priority>\n 传函分母=%<Denominator>**

该模块的图标将显示如下的内容：



图 8-6 设置属性格式字符串后的效果

如果参数的值不是字符串或数字，相应位置会显示 N/S (not supported)。如果参数名无效，该位置将显示“??？”。

#### 4. 模块外形的调整

##### (1) 改变模块的大小

选定模块，用鼠标点住其周围的四个黑方块中的任意一个拖动，这时会出现一个虚线的矩形表示新模块的位置，到需要的位置后释放鼠标即可。

##### (2) 调整模块的方向

选定模块，选取菜单 Format 下的 Rotate Block 使模块旋转 90°，Flip Block 使模块旋转 180°。效果如图 8-7 所示。



图 8-7 调整模块方向

### (3) 给模块加阴影

选定模块，选取菜单 Format 下的 Show Drop Shadow 使模块产生阴影效果。

## 5. 模块名的处理

### (1) 模块名的显示与否

选定模块，选取菜单 Format 下的 Hide Name，模块名就会被隐藏，同时 Hide Name 改为 Show Name。选取 Show Name 就会使模块隐藏的名字显示出来。

### (2) 修改模块名

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随意修改。

模块名和模块图标中的字体也可以更改，方法是选定模块，在菜单 Format 下选取 Font，这时会弹出 Set Font 对话框，在对话框中选取想要的字体。

### (3) 改变模块名的位置

模块名的位置有一定的规律，当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，缺省在下侧；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，缺省在左侧。

因此模块名只能从原位置移到相对的位置。可以用鼠标拖动模块名到其相对的位置；也可以选定模块，用菜单 Format 下的 Flip Name 实现相同的移动。

## 8.2.3 模块的连接

上一节我们介绍了对模块本身的各种操作。当我们设置好了各个模块后，还需要把它们按照一定的顺序连接起来才能组成一个完整的系统模型。这一节我们将讨论模块连接的相关问题。

### 1. 连接两个模块

这是最基本的情况：从一个模块的输出端连到另一个模块的输入端。方法是在移动鼠标到输出端，鼠标的箭头会变成十字形光标，这时点住鼠标左键，移动鼠标到另一个模块的输入端，当十字形光标出现“重影”时，释放鼠标左键就完成了连接。

如果两个模块不在同一水平线上，连线是一条折线。要用斜线表示，需要在连接时按住 Shift 键。两种连线的结果见图 8-8。

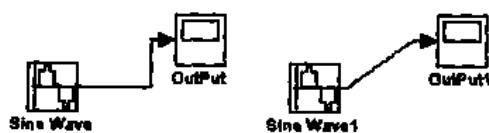


图 8-8 两模块不在同一水平线上

### 2. 模块间连线的调整

如图 8-9，这种调整模块间连线位置的情况采用鼠标简单拖动的办法实现。即先把鼠标

移到需要移动的线段的位置，按住鼠标左键，移动鼠标到目标位置，释放鼠标左键。

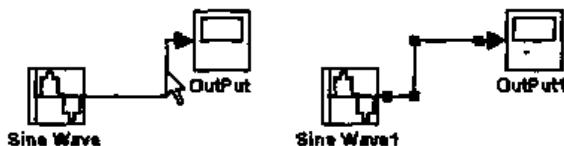


图 8-9 调整连线的位置（一）

还有一种情况，如图 8-10 所示，要把一条直线分成斜线段。调整方法和前一种情况类似，不同之处在于按住鼠标左键之前要先按下 Shift 键，出现小黑方块之后，鼠标点住小黑方块移动，移好后释放 Shift 键和鼠标。



图 8-10 调整连线的位置（二）

### 3. 在连线之间插入模块

把该模块用鼠标拖到连线上，然后释放鼠标即可。

### 4. 连线的分支

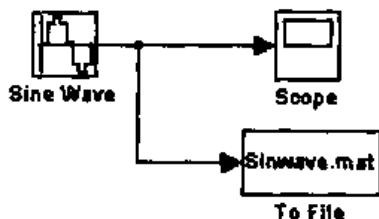


图 8-11 连线的分支

我们经常会碰到一些情况，需要把一个信号输送到不同的模块，这时就需要分支结构的连线。如图 8-11 所示，要把正弦波信号实时显示出来，同时还要保存到文件。

这种情况的步骤是：在先连好一条线之后，把鼠标移到支线的起点位置，先按下 Ctrl 键，然后按住鼠标拖到目标模块的输入端，释放鼠标和 Ctrl 键。

## 8.2.4 在连线上反映信息

### 1. 用粗线表示向量

为了能够比较直观的区别各个模块之间传输的数据是向量还是数组，可以选择模型文件菜单 Format 下的 Wide Vector Lines 选项，这样传输数组的连线就会变粗。如果再选择 Format 下的 Vector Lines Widths 选项，在传输矩阵的连线上方会显示出通过该连线的数组维数。

### 2. 显示数据类型

在连线上可以显示前一个模块输出的数据类型：选择菜单 Format 下的 Port Data Types 选项。

### 3. 信号标记

为了使模型更加直观、可读性更强，我们可以为传输的信号做标记。

建立信号标记的办法是：双击要做标记的线段，出现一个小文本编辑框，在里面输入标记的文本，这样就建立了一个信号标记。

信号标记可以随信号的传输在一些模块中进行传递。支持这种传递的模块有 Mux、Demux、Import、From、Selector、Subsystem 和 Enable。

要实现信号标记的传递，需要在上面列出的模块的输出端建立一个以“<”开头的标记。当开始仿真或执行 Edit 菜单下的 Update Diagram 命令时，传输过来的信号标记就会显示出来。图 8-12 示意了这个传递的过程。

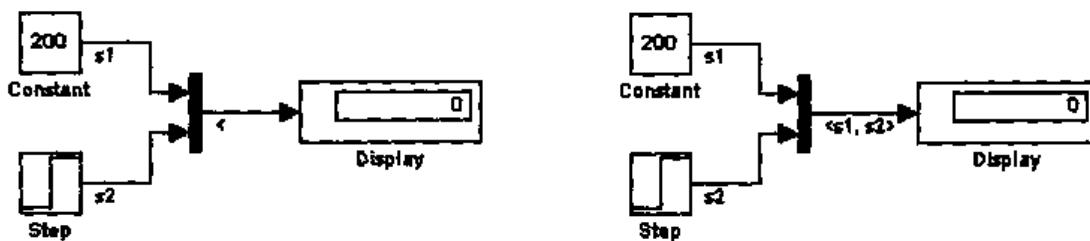


图 8-12 信号标记的传递

## 8.3 Simulink 的几类基本模块

本节以表格的形式给出了 Simulink 几个基本模块库中的模块功能简介，从而用户可以对 Simulink 的一些主要模块有一个初步的认识。如表 8-1 到 8-6 所示，这些表格中的模块名和模块库中的模块图标下的名称一致。

打开模块库(图标)窗口的方法很简单，以连续系统模块库(Continuous)为例，在 Simulink 模块库浏览器窗口中选中 Simulink，然后单击鼠标右键，这时会在鼠标指针的位置弹出一行上下文菜单，内容是“Open the ‘Simulink’ Library”，用鼠标左键单击该项，出现如图 8-13 所示的 Simulink 基本库窗口，在此窗口中双击 Continuous 模块库的图标即可打开该模块库窗口，如图 8-14 所示。也可以在模块库浏览器窗口中 Simulink 下选中 Continuous 项，然后用和打开 Simulink 基本库窗口同样的方法直接打开连续系统模块库。

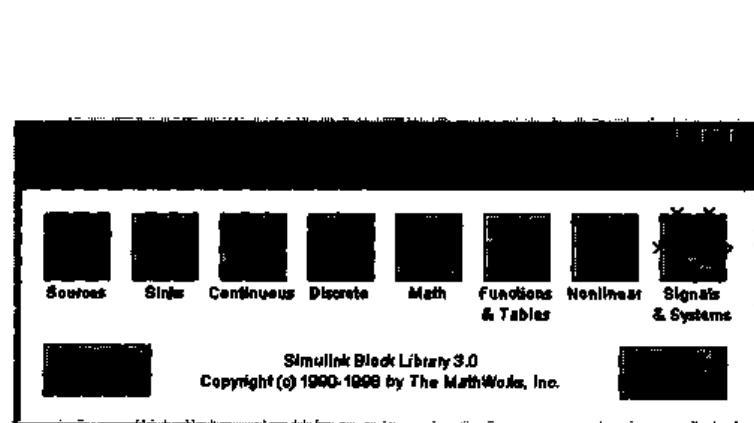


图 8-13 Simulink 基本库

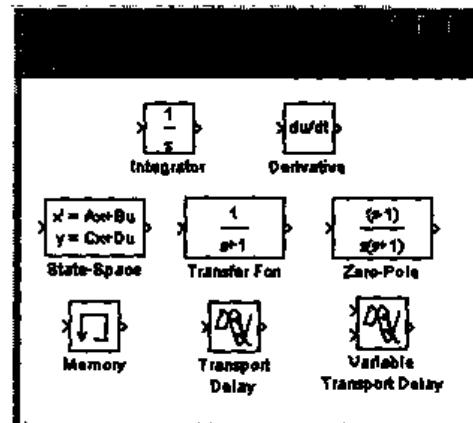


图 8-14 连续系统模块库

表 8-1 输入源模块 (Sources)

模块名	功能简介	模块名	功能简介
Constant	常数	Chirp Signal	频率不断变化的正弦信号
Signal Generator	信号发生器	Clock	输出当前的仿真时间
Step	阶跃信号	Digital Clock	按指定速率输出当前仿真时间
Ramp	线性增加或减小的信号	From File	从文件读数据
Sine Wave	正弦波	From Workspace	从当前工作空间定义的矩阵读数据
Repeating Sequence	重复的线性信号, 类似锯齿形	Random Number	高斯分布的随机信号
Discrete Pulse Generator	离散脉冲发生器和采样时间有关	Uniform Random Number	平均分布的随机信号
Pulse Generator	脉冲发生器, 和采样时间无关	Band-Limited White Noise	带限白噪声

表 8-2 接收模块 (Sinks)

模块名	功能简介	模块名	功能简介
Scope	示波器	To File	保存到文件
XY Graph	两个信号的关系图, 用 MATLAB 图形显示	To Workspace	输出到当前工作空间的变量
Display	实时数值显示	Stop Simulation	输入不为零时停止仿真

表 8-3 连续系统模块 (Continuous)

模块名	功能简介	模块名	功能简介
Integrator	积分环节	Zero-Pole	零-极点模型
Derivative	微分环节	Memory	把前一步的输入作为输出
State-Space	状态方程	Transport Delay	把输入信号按给定的时间做延迟
Transfer Fcn	传递函数	Variable Transport Delay	按第二个输入指定的时间把第一个输入做延迟

表 8-4 离散系统模块 (Discrete)

模块名	功能简介	模块名	功能简介
Zero-Order Hold	零阶保持器	Discrete Filter	离散滤波器 (IIR、FIR)
Unit Delay	采样保持, 延迟一个周期	Discrete Transfer Fcn	离散传递函数
Discrete-Time Integrator	离散时间积分	Discrete Zero-Pole	离散零-极点模型
Discrete State-Space	离散状态方程	First-Order Hold	一阶保持器

表 8-5 信号与系统模块 (Signals &amp; Systems)

模块名	功能简介
In1,Out1	为子系统或模型提供输入、输出端口
Enable	放到子系统中建立使能子系统
Trigger	放到子系统中建立触发子系统
Mux	把向量或标量组合为大的向量

(续)

模块名	功能简介
Demux	把向量分为标量或小的向量
Bus Selector	从输入中选择信号
Selector	选择输入的元素
Merge	把输入信号合并为输出信号(标量)
From	接收标记(Tag)相同的 Goto 模块的信号
Goto Tag Visibility	定义 Goto 模块标记(Tag)的有效范围
Goto	把信号输出到标记(Tag)相同的 From 模块
Data Store Read	从指定的数据存储器读数据
Data Store Memory	为数据存储器定义内存区域
Data Store Write	写数据到指定的数据存储器
Ground	给未连接的输入端接地, 输出 0
Terminator	连到未连接的输出端, 终止输出信号
Data Type Conversion	数据类型转换
Function-Call Generator	函数调用发生器
SubSystem	空的子系统
Configurable Subsystem	表示从用户指定的模块库里选择的任何模块
Model Info	显示模型的修改信息
Hit Crossing	检测输入信号的零交叉点
IC	设置信号的初始值
Width	检查输入信号的宽度
Probe	检测连线的宽度、采样时间和(或)复数信号标记

表 8-6 数学运算模块(Math)

模块名	功能简介	模块名	功能简介
Sum	对输入求代数和	Sign	取输入的正负符号
Product	对输入求积或商	Rounding Function	取整函数
Dot Product	点积(内积)	Combinatorial Logic	建立逻辑真值表
Gain	常量增益(输入乘一个常数)	Logical Operator	逻辑操作符
Slider Gain	可以用滑动条改变的增益	Relational Operator	比较操作符
Matrix Gain	矩阵增益(输入乘一个矩阵)	Complex to Magnitude-Angle	求复数的幅值、相角
Math Function	数学运算函数	Magnitude-Angle to Complex	根据幅值、相角得到复数
Trigonometric Function	三角函数	Complex to Real-Imag	求复数的实部、虚部
MinMax	求最值	Real-Imag to Complex	根据实部、虚部得到复数
Abs	求绝对值, 或求模(复数)	Algebraic Constraint	强制输入信号为零

# 第 9 章 信号处理工具箱

简单的说，信号处理是以数值计算的方法对信号进行采集、变换、综合、估计与识别等加工处理，借以达到提取信息和便于应用的目的。

MATLAB 的信号处理工具箱是信号处理算法文件的集合，它处理的基本对象是信号与系统。信号处理工具箱中的文件位于目录\toolbox\signal 下。

利用工具箱中的文件可以实现信号的变换、滤波、谱估计，滤波器设计，线性系统分析等等功能。工具箱还提供了图形用户界面工具，可以交互地完成很多信号处理的功能。

信号处理工具箱约定以列向量表示单通道信号，在多通道情况下，每一列代表一个通道，每一行对应一个采样点。

## 9.1 波形产生

工具箱中提供了很多函数用于产生各种波形，如表 9-1 所示。本节将分类讨论一些主要波形的特点和产生办法。

表 9-1 波形产生函数

函数	产生的波形	函数	产生的波形
chirp	扫频余弦	sawtooth	锯齿波或三角波
diric	Dirichlet 函数或周期 sinc 函数	sinc	Sinc 函数
gauspuls	高斯调制正弦波脉冲	square	方波
pulstran	脉冲序列	strips	带状条纹图
rectpuls	非周期的矩形脉冲	tripuls	非周期的三角脉冲

### 9.1.1 常用周期波形

#### 1. 方波

产生方波的函数 square 有两种调用方式：

`x = square(t)`: 相对时间数组 t 中的元素生成周期为  $2\pi$ 、峰峰值为  $\pm 1$  的方波。

`x = square(t, duty)`: duty 是“占空比”，即信号为正的区域在一个周期内所占的百分比。

#### 2. 锯齿波和三角波

产生锯齿波和三角波的函数 sawtooth 有如下两种调用方式：

`x = sawtooth(t)`: 相对时间数组 t 中的元素生成周期为  $2\pi$ 、峰峰值为  $\pm 1$  的锯齿波。锯齿波在  $2\pi$  整数倍的位置定义为 -1，随时间变化按照  $1/\pi$  的斜率增加到 1。

`x = sawtooth(t, width)`: 根据 width 值的不同产生不同形状的三角波，参数 width 是 0~1 之间的标量，指定在一个周期之间最大值的位置，width 是该位置的横坐标和周期的比值。因而，当 width = 0.5 时产生标准的对称三角波，当 width = 1 时产生锯齿波。

如下一段程序分别产生方波、锯齿波、三角波，并画图。结果如图 9-1 所示。

```
t = 0:0.01:3;
subplot(2,2,1)
y = square(2*pi*t); % 产生周期为1的标准方波
plot(t,y)
axis([0 3 -1.1,1.1])
subplot(2,2,2)
y = square(2*pi*t,70); % 方波大于0的区域在周期中占的百分比为70%
plot(t,y)
axis([0 3 -1.1,1.1])
subplot(2,2,3)
y = sawtooth(2*pi*t); % 产生周期为1的标准锯齿波
plot(t,y)
subplot(2,2,4)
y = sawtooth(2*pi*t,0.5); % 产生周期为1的标准三角波
plot(t,y)
```

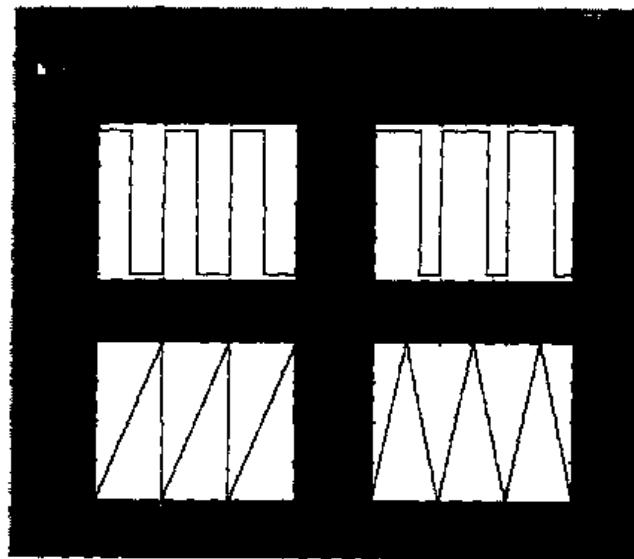


图 9-1 周期波形

### 9.1.2 Sinc 函数和 Dirichlet 函数

#### 1. Sinc 函数

Sinc 函数是信号处理中一个很重要的函数，它的表达式为：

$$\text{Sinc}(t) = \begin{cases} 1 & t = 0 \\ \frac{\sin(\pi t)}{\pi t} & t \neq 0 \end{cases}$$

它实际上是宽为  $2\pi$ ，高为 1 的矩形脉冲的连续反傅立叶变换，即：

$$\text{Sinc}(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} 1 \times e^{j\omega t} d\omega$$

求取 Sinc 函数的格式为:  $y = \text{sinc}(t)$ 。

通过下面的程序我们可以看到 Sinc 函数的典型形状, 如图 9-2:

```
t = -6:0.01:6;
y = sinc(t); % Sinc 函数的调用方式
plot(t,y)
```

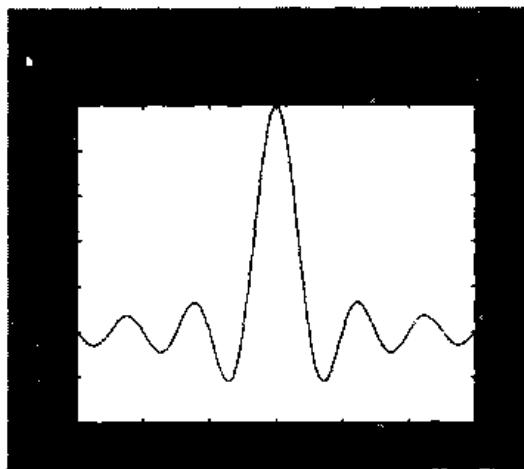


图 9-2 Sinc 函数的形状

## 2. Dirichlet 函数

Dirichlet 函数又称周期 Sinc 函数, diric 用来求取 Dirichlet 函数, 调用格式为:

$y = \text{diric}(x, n)$ : 其中  $n$  必须是正整数。

Dirichlet 函数的定义如下:

$$\text{diric}(x, n) = \begin{cases} \frac{x}{1^{2\pi}(n-1)} & x = 0, \pm 2\pi, \pm 4\pi, \dots \\ \frac{\sin(nx/2)}{n \sin(x/2)} & \text{其它} \end{cases}$$

可以看出, 当  $n$  为奇数时, Dirichlet 函数的周期为  $2\pi$ , 当  $n$  为偶数时, Dirichlet 函数的周期为  $4\pi$ 。

下面的程序在  $0 \sim 8\pi$  之间绘制 Dirichlet 函数的图形, 结果如图 9-3 所示。

```
x = 0:0.01:8*pi;
y1 = diric(x,7); % n 为奇数, 见图 9-3 上半部分
y2 = diric(x,8); % n 为偶数, 见图 9-3 下半部分
subplot(2,1,1)
plot(x,y1)
axis tight
subplot(2,1,2)
```

```
plot(x,y2)
axis tight
```

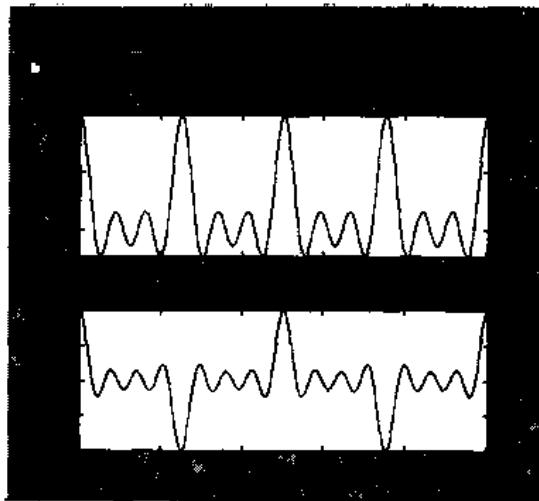


图 9-3 Dirichlet 函数

### 9.1.3 脉冲信号

#### 1. 非周期矩形脉冲和三角脉冲

函数 `rectpuls` 和 `tripuls` 分别用于产生非周期矩形脉冲和三角脉冲，其用法分别为：

1) `rectpuls(t, w)`: 产生连续、非周期、单位高度的矩形脉冲，宽度为 `w`，缺省时宽度为 1。`t` 是采样时间数组，脉冲信号的中心位置在 `t = 0` 处。要注意，该函数产生的矩形脉冲非零幅值的区间左边是闭区间，右边是开区间，即：以缺省脉冲宽度 1 为例，`rectpuls(-0.5) = 1`，而 `rectpuls(0.5) = 0`。

2) `y = tripuls(T, w, s)`: 产生连续、非周期、单位高度的三角脉冲，`T` 是采样时间数组。`w` 是脉冲宽度，缺省值为 1；`s` 是倾斜程度， $-1 < s < 1$ ，缺省值是 `s=0`，产生对称三角脉冲，脉冲信号的中心位置在 `t = 0` 处。

#### 2. 高斯调制正弦波脉冲

函数 `gauspuls` 产生高斯调制 (Gaussian-modulated) 正弦波脉冲。该函数调用格式有：

1) `yi = (t, fc, bw)`: 产生单位幅值的高斯 RF 脉冲，`t` 是时间数组，`fc` 是中心频率（单位 Hz），`bw` 是带宽占参考带宽的比例（小数形式）。缺省情况下：`fc = 1000 Hz, bw = 0.5`。

2) `yi = gauspuls(t, fc, bw, bwr)`: 参数 `bwr` 指定相对于规格化的信号峰值的参考水平，该水平上的带宽就是 `bw` 所依据的参考带宽。`bwr` 的单位是 dB，缺省是 -6 dB。

3) `[yi, yq, ye] = gauspuls()`: `yq` 是积分脉冲，`ye` 是信号的包络。

4) `tc = gauspuls('cutoff', fc, bw, bwr, tpe)`: 带有参数 'cutoff' 时，该函数不产生脉冲信号，而是返回截止时间 `tc`（大于或等于 0），脉冲信号的包络低于参数 `tpe` 指定的分贝数的时间就是截止时间。参数 `tpe` 的数值是相对于峰值包络的，它必须小于 0，缺省值是 -60 dB。

#### 3. 脉冲序列

脉冲序列函数 `pulstran` 是前面三种脉冲函数的综合。它有以下几种用法：

1) `y = pulstran(t, d, 'func')`: 产生基于函数 `func` 的脉冲序列，`func` 的取值可以是 `gauspuls`、

`rectpuls`、`tripuls` 三者中的任意一个。`pulstran` 用函数 `func` 计算 `length(d)` 次，结果相加，即  $y = \text{func}(t-d(1)) + \text{func}(t-d(2)) + \dots$

2) `pulstran(t,d,'func',p1,p2,...)`:  $p_1, p_2\dots$  是传递给函数'func'的参数。

3) `pulstran(t, d, p, Fs)`: 向量  $p$  中是原始脉冲信号，通过对  $p$  的多次延迟并相加最后得到脉冲序列。 $F_s$  是采样速率，单位 Hz，缺省值是 1Hz。

1) 如下一段程序分别产生几种脉冲信号，结果如图 9-4 所示。

```
t = -2:0.01:2;
yrect = rectpuls(t);
ytri = tripuls(t,1,-0.5);
subplot(2,2,1)
plot(t,yrect) % 矩形脉冲
subplot(2,2,2)
plot(t,ytri) % 三角脉冲
% 高斯分布1kHz的RF脉冲，带宽60%，采样频率1e5Hz,
% 信号包络低于峰值40dB时截止
tc = gauspuls('cutoff',1000,0.6,[],-40);
t = -tc : 0.00001 : tc;
[yi, yq, ye] = gauspuls(t,1000,0.6);
subplot(2,2,3)
plot(t,yi)
subplot(2,2,4)
plot(t,yq) % 脉冲信号的积分
```

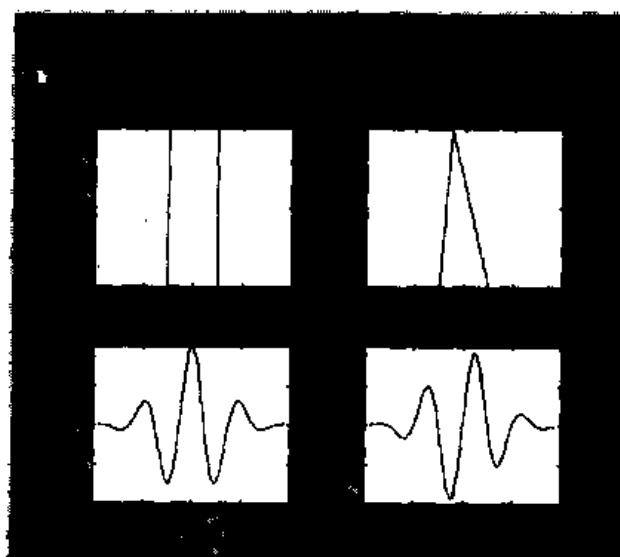


图 9-4 脉冲信号

(2) 下面的程序产生各种脉冲序列，结果如图 9-5 所示。

% 产生三角波脉冲序列

```

t = -1 : 0.001 : 1; % 采样频率1kHz, 从-1秒到1秒
d = -1 : 0.5 : 1; % 循环频率2Hz
y = pulstran(t,d,'tripuls',0.2,1); % 三角波宽度0.2秒, 峰值在最右端(后锯齿)
subplot(2,2,1)
plot(t,y)
% 产生高斯脉冲序列
t = 0 : 0.001 : 1; % 采样频率1kHz, 从0到1秒
d = [0 : 0.2 : 1; 0.8.^ (0:5) ]'; % 循环频率5Hz, 按照0.8的比例峰值逐渐减小
y = pulstran(t,d,'gauspuls',50,0.5); % 信号频率50Hz, 50%带宽
subplot(2,2,2)
plot(t,y)
% 从给定信号产生脉冲序列
p = sin(0:0.1:3.1); % 给定半正弦作为原始脉冲信号, 共32个点
t = 0:0.1:31; d = (0:9)'*3.1; % 采样频率10Hz, 重复10次
y = pulstran(t, d, p, 10); % 设置Fs=10
subplot(2,2,3)
plot(0:0.1:3.1,p)
subplot(2,2,4)
plot(t,y)

```

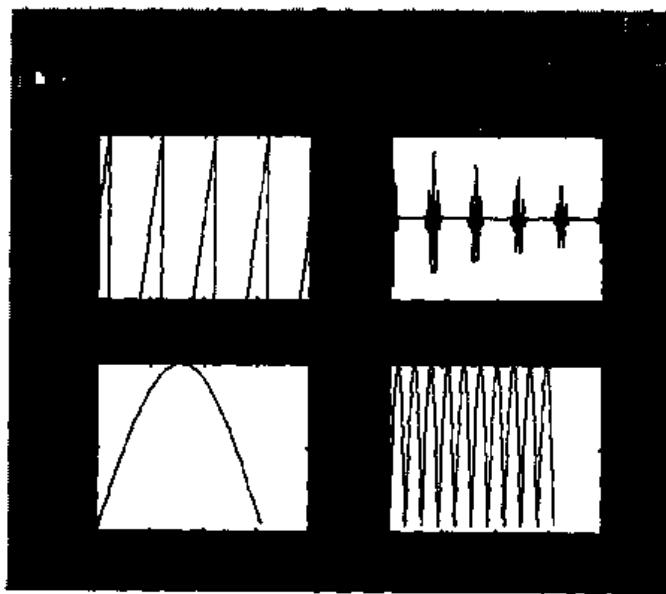


图 9-5 产生脉冲序列

#### 9.1.4 扫频信号

扫频信号在系统实验分析中很有用, chirp 函数生成扫频余弦信号, 函数用法如下:

- 1)  $y = \text{chirp}(t, f_0, t_1, f_1)$ : 在数组  $t$  指定的时间向量上产生线性扫频余弦信号,  $f_0$  是 0 时刻的瞬时频率,  $f_1$  是  $t_1$  时刻的瞬时频率。 $f_0$  和  $f_1$  的单位都是 Hz, 缺省情况下,  $f_0$  是 0,  $t_1$  是 1,  $f_1$  是 100。

2)  $y = \text{chirp}(t, f_0, t_1, f_1, \text{'method'}, \phi)$ : 参数  $\phi$  指定初始相位, 单位是度, 缺省值为 0。参数  $\text{method}$  用来指定扫频的方式, 它的取值有如下三种:

- linear: 线性扫频(缺省值), 其瞬时扫描频率  $f_i(t)$  由下式确定

$$f_i(t) = f_0 + \beta t \quad \text{式中 } \beta = (f_1 - f_0)/t_1, \text{ 确保 } t_1 \text{ 时刻的瞬时频率为 } f_1。$$

- quadratic: 二次扫频, 瞬时扫描频率由下式确定

$$f_i(t) = f_0 + \beta t^2 \quad \text{式中 } \beta = (f_1 - f_0)/t_1$$

- logarithmic: 对数扫频, 瞬时扫描频率由下式确定

$$f_i(t) = f_0 + 10^{\beta t} \quad \text{式中 } \beta = [\log_{10}(f_1 - f_0)]/t_1$$

如下一段命令按二次扫频方式产生扫频信号并绘制其谱图, 如图 9-6 所示, 从图中可以明显看出信号频率的变化规律。

```
t = -2:0.001:2; % 在±2 秒区间上, 采样频率 1kHz
y = chirp(t,100,1,200,'quadratic'); % 起始频率 100Hz, 1 秒时达到 200Hz
specgram(y,128,1e3,128,60) % 显示谱图, 此函数的用法请参看帮助
```



图 9-6 扫频信号的谱图

## 9.2 线性系统模型

MATLAB 在信号处理工具箱中提供了多种模型用于表示线性系统, 这样用户可以根据需要在模型之间自由转换。这部分将介绍 MATLAB 支持的线性系统模型以及如何实现这些模型的相互转换。

### 9.2.1 离散时间系统模型

MATLAB 支持的离散时间系统模型有如下几种:

- 传递函数
- 零极点模型
- 状态方程

- 部分分式(留数)

- 二次分式

- 网格结构

- 卷积矩阵

1. 传递函数

传递函数是主要的离散时间模型，它是离散系统的基本  $z$  域表示形式，为两个多项式之比。以传递函数形式表示的离散系统模型如下：

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \cdots + a(na+1)z^{-na}} X(z)$$

这里， $b(i)$ 与 $a(i)$ 是离散系统的系数，分别存储在两个向量  $b$  和  $a$  中（一般为行向量）， $na$  和  $nb$  中较大的一个是离散系统的阶次。

注意向量  $b$  和  $a$  中的系数是按照  $z^{-1}$  的升幂顺序排列的，这是数字信号处理中的习惯。

2. 零极点模型

传递函数的零极点模型（因式分解）为：

$$H(z) = \frac{q(z)}{p(z)} = k \frac{(z - q(1))(z - q(2)) \cdots (z - q(n))}{(z - p(1))(z - p(2)) \cdots (z - p(n))}$$

其中  $q(i)$  为零点， $p(i)$  为极点，它们分别存储在两个向量  $q$  和  $p$  中（一般为列向量）， $k$  为系统的增益。

3. 状态方程

线性系统总可以用一组一阶微分方程来表示，这就是状态方程形式，如下所示：

$$\begin{aligned} x(n+1) &= Ax(n) + Bu(n) \\ y(n) &= Cx(n) + Du(n) \end{aligned}$$

这里， $u$  是输入， $x$  是状态向量， $y$  是输出。对于单输入单输出系统（单通道），即  $u$  和  $y$  是标量，这时  $A$  为  $m \times m$  矩阵， $B$  是列向量， $C$  是行向量， $D$  是标量；对于多通道系统，即  $u$  和  $y$  是向量，这时  $A$ 、 $B$ 、 $C$ 、 $D$  都是矩阵。

对状态方程进行  $z$  变换，就可以得到系统的传递函数形式，传递函数和状态矩阵之间的关系如下：

$$H(z) = C(zI - A)^{-1}B + D$$

4. 部分分式

每一个传递函数都有相应的部分分式或留数形式，如下：

$$H(z) = \frac{r(1)}{1 - p(1)z^{-1}} + \cdots + \frac{r(n)}{1 - p(n)z^{-1}} + k(1) + k(2)z^{-1} + \cdots + k(m-n+1)z^{-(m-n)}$$

这里的传递函数没有重极点。 $n$  是传递函数分母多项式的次数。

关于部分分式的相关内容，请参见 6.3.7 节。

5. 二次分式

任何传递函数  $H(z)$  都可以用如下形式的二次分式表示：

$$H(z) = \prod_{k=1}^L H_k(z) = \prod_{k=1}^L \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}}$$

其中 L 为描述系统的二次分式的数目。

MATLAB 用一个  $L \times 6$  的矩阵 sos 来表示离散系统的二次分式形式, sos 矩阵的每一行的六个元素代表一个二次分式, 前三个元素是分子的系数, 后三个元素是分母的系数。矩阵形式如下:

$$sos = \begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix}$$

## 6. 网格结构

对于离散系统的 N 阶全零点或全极点传递函数, 用一个多项式系数向量 a(n) 描述, 其中  $n = 1, 2, \dots, N+1$ 。存在 N 个相应的网状结构映射系数 k(n),  $n = 1, 2, \dots, N$ , 利用 k(n) 可以按照图 9-7 所示的网状结构表示这种离散系统。

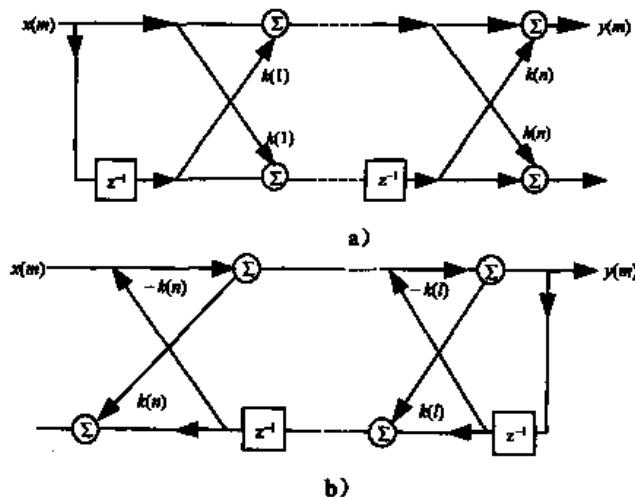


图 9-7 特殊离散系统的网状结构

a) FIR 滤波器 b) IIR 滤波器

图 9-7a 图所示的结构实际上是 FIR 滤波器, b 图所示的结构是 IIR 滤波器, 参见后面的 9.4 和 9.5 节。

对于一般性的离散系统, 其传递函数有两个多项式系数 a 和 b, 这样的系统也可以用网状结构表示, 如图 9-8 所示。

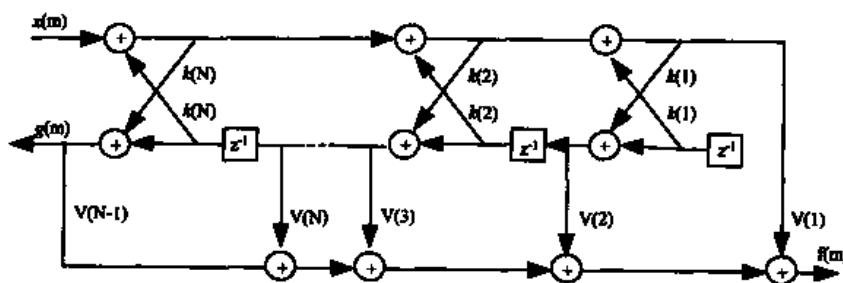


图 9-8 一般离散系统的网状结构

图中的向量  $v(n)$  为阶梯系数。

### 7. 卷积矩阵

在信号处理中，两个向量或矩阵的卷积等价于用一个输入对另一个输入进行滤波，这种关系使得可以用卷积矩阵来表示数字滤波器，也就是离散系统的模型。

给定任意向量，工具箱中的函数 `convmtx` 产生一个矩阵，其与另一个向量的内积等于这两个向量的卷积。这个矩阵就是表示系统的卷积矩阵。

### 9.2.2 连续时间系统模型

MATLAB 中在连续时间域表示线性系统的模型和离散域类似，有以下几种形式：

- 状态方程
- 部分分式
- 传递函数
- 零极点模型

对于任何线性时不变系统，总可以用一组一阶微分方程表示，形式如下：

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

和离散系统相同，这里的 A、B、C、D 就是表示系统的状态矩阵。

对状态方程进行拉氏变换，就可以得到连续系统的传递函数形式，传递函数和状态矩阵之间的关系如下：

$$H(s) = C(sI - A)^{-1}B + D$$

对于单输入单输出系统， $H(s)$  可以表示为如下形式：

$$H(s) = \frac{b(1)s^{nb} + b(2)s^{nb-1} + \dots + b(nb+1)}{a(1)s^{na} + a(2)s^{na-1} + \dots + a(na+1)}$$

给定  $H(s)$  的系数  $b(n)$  和  $a(n)$ ，可以用函数 `residue` 得到系统的部分分式形式，如果没有重极点，则可以表示为：

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \dots + \frac{r_n}{s - p_n} + k_s$$

连续系统传递函数的零极点模型为：

$$H(s) = k \frac{(s - z(1))(s - z(2)) \cdots (s - z(n))}{(s - p(1))(s - p(2)) \cdots (s - p(n))}$$

其中向量  $z$  是零点， $p$  是极点， $k$  是增益。

### 9.2.3 线性系统变换

信号处理工具箱中提供了在不同的线性系统模型之间进行变换的函数，表 9-2 列出了从一种模型（表左侧）变换为另一种模型（表上方）所用的函数，这些函数的用法都比较

简单, 请参见在线帮助。

表 9-2 线性系统变换函数

	传递函数	状态方程	零极点	部分分式	网格结构	二次分式	卷积矩阵
传递函数		tf2ss	tf2zp roots	residuez residue	tf2latc		convmtx
状态方程	ss2tf		ss2zp			ss2sos	
零极点	zp2tf poly	zp2ss				zp2sos	
部分分式	residuez residue						
网格结构	latc2tf						
二次分式	sos2tf	sos2ss	sos2zp				
卷积矩阵							

### 9.3 信号变换

除了傅立叶变换(参见第六章)之外, MATLAB 工具箱还提供了几种其它类型的变换, 包括 Chirp z 变换、离散余弦变换(DCT)、Hilbert 变换, 它们有着各自的理论与应用背景。本节将简要介绍这几种变换的含义及其应用。

#### 9.3.1 Chirp z 变换

z 变换是离散信号与系统分析与综合的重要工具, 其地位和作用犹如拉普拉斯变换对于连续信号和连续系统。

给定一个离散信号  $x(n)$ ,  $n = 0 \sim \infty$ , 可直接给出  $x(n)$  的(单边) z 变换定义为:

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n}$$

Chirp z 变换指在 z 域内对输入序列沿螺旋轮廓线进行 z 变换。该变换的 MATLAB 函数为 czt, 用法如下:

$y = \text{czt}(x, m, w, a)$ : 计算信号 x 的 Chirp z 变换, 参数 m 是变换数据的长度, 参数 w 和 a 按如下关系式确定变换的螺旋轮廓线:

$$z = a * (w.^{(0:m-1)})$$

其中, a 是 Z 平面轮廓线上复数的起点, w 是沿轮廓线上点与点之间的比率。参数 m、w、a 的缺省值是:  $m = \text{length}(x)$ ,  $w = \exp(j * 2 * \pi / m)$ ,  $a = 1$ 。

缺省情况下, z 变换沿着单位圆进行, 这时等同于离散傅立叶变换, 即 fft(x)。

#### 9.3.2 离散余弦变换(DCT)

DCT 的定义是 1974 年 Ahmed 和 Rao 提出的。对于给定的序列  $x(n)$ ,  $n = 1, 2, \dots, N$ (这

里我们按照 MATLAB 中的习惯，下标从 1 开始），其 DCT 定义为：

$$y(k) = w(k) \sum_{n=1}^N x(n) \cos \frac{\pi(2n-1)(k-1)}{2N} \quad k = 1, 2, \dots, N$$

其中，

$$w(k) = \begin{cases} \frac{1}{\sqrt{N}} & k = 1 \\ \sqrt{\frac{2}{N}} & 2 \leq k \leq N \end{cases}$$

MATLAB 中用于计算 DCT 的函数是 `dct`，用法为：

`y = dct(x)` 或 `y = dct(x, n)`：参数 `n` 用于指定变换的数据长度，根据 `n` 对原数据进行增删。

DCT 的反变换由函数 `idct` 完成，该函数从 DCT 的系数重建一个序列，用法为：

`x = idct(y)` 或 `x = idct(y, n)`：参数 `n` 的作用和在 `dct` 函数中相同。

DCT 具有很强的压缩能力，用少数的几个 DCT 系数就可以非常准确地重建一个序列。

如下程序先生成一组信号，然后用 DCT 和反 DCT 重建该信号，结果如图 9-9 所示。

```
t = 0:0.001:1;
x = t + sin(10*2*pi*t).*cos(2*pi*t); % 原始信号, 9-7左侧图
y = dct(x); % DCT变换
n = fix(length(y)/20); % 取数据长度的1/20
y(n+1:length(y))=0; % DCT变换结果中, 1/20以后的部分全设为0
z = idct(y); % 仅用前1/20的数据量进行反DCT变换, 重建原始信号
subplot(1,2,1)
plot(t,x)
subplot(1,2,2)
plot(t,z) % 重建结果如图9-9右侧
```

信号重建精度的量度是：

`norm(x-z)/norm(x)`

结果为：

`ans =`

`0.0316`

这说明，重建的误差是 0.0316，即用 1/20 的数据量保留了原始信号约 97% 的能量。

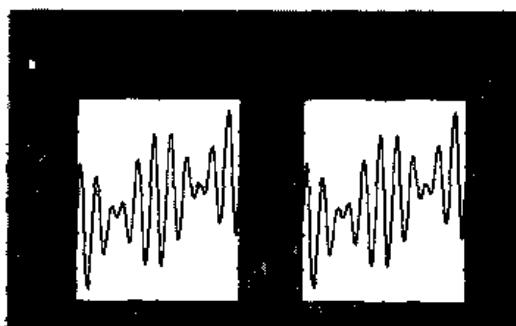


图 9-9 用 DCT 变换实现信号重建

### 9.3.3 Hilbert 变换

Hilbert 变换是信号分析中的重要工具，对于给定的信号  $x(n)$ ，Hilbert 变换定义为：

$$\hat{x}(n) = \frac{2}{\pi} \sum_{m=-\infty}^{\infty} \frac{x(n-2m-1)}{2m+1}$$

MATLAB 中进行 Hilbert 变换的函数是 `hilbert`，该函数的用法为：

```
y = hilbert(x)
```

其中返回值  $y$  一般称为解析信号，它的实部是原始信号的实部，虚部是 Hilbert 变换结果，Hilbert 变换与原始信号之间有  $90^\circ$  的相移。解析信号的主要性质是其在 Z 平面上单位圆的下半部分的 z 变换为零。

```
t = 0:0.001:2*pi;
x = sin(3*t);
y = hilbert(x);
plot(t, real(y), ':', t, imag(y)) % 在图 9-10 中同时画出原始信号和 Hilbert 变换结果
```

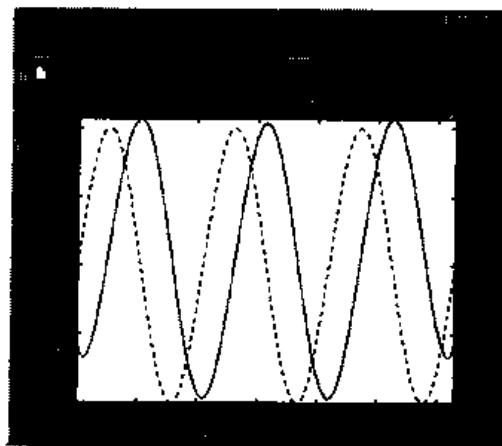


图 9-10 Hilbert 变换

## 9.4 数字滤波器的应用与分析

本节介绍如何利用 MATLAB 中的 `filter` 函数（内置函数）及信号处理工具箱的其它一些函数对离散信号进行数字滤波，并讨论怎样分析滤波器特征。

### 9.4.1 数字滤波的应用

#### 1. 滤波原理

滤波器，顾名思义，其目的是对输入信号起到滤波的作用。滤波器的输出  $y(n)$  与输入  $x(n)$  之间的关系是脉冲响应  $h(n)$ ，它们的关系如下：

$$y(n) = x(n) * h(n) \quad (\text{卷积})$$

若滤波器的输入输出都是离散信号，那么该滤波器的脉冲响应也必然是离散的，这样的滤波器就称为数字滤波器。

可以看出，滤波的数学基础是卷积，若数字滤波器的脉冲响应  $h(n)$  为有限长，且输入  $x(n)$  也是有限长，则可以用卷积运算来进行滤波。

通常，数字滤波器输出  $y(n)$  的  $z$  变换  $Y(z)$  与输入  $x(n)$  的  $z$  变换  $X(z)$  有如下关系：

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

其中， $H(z)$  是滤波器的传递函数，向量  $b(k)$  与  $a(k)$  是滤波器系数， $na$  和  $nb$  中较大的数是滤波器的阶次。

## 2. filter 函数

滤波函数 `filter` 是 MATLAB 的内置函数，它可以对实数或复数进行数字滤波，IIR（无限冲激响应）滤波器和 FIR（有限冲激响应）滤波器的功能都能够实现。

该函数是按照图 9-11 所示的框图来进行滤波的，其中  $n-1$  是滤波器的阶次， $z^{-1}$  模块称为移位寄存器。

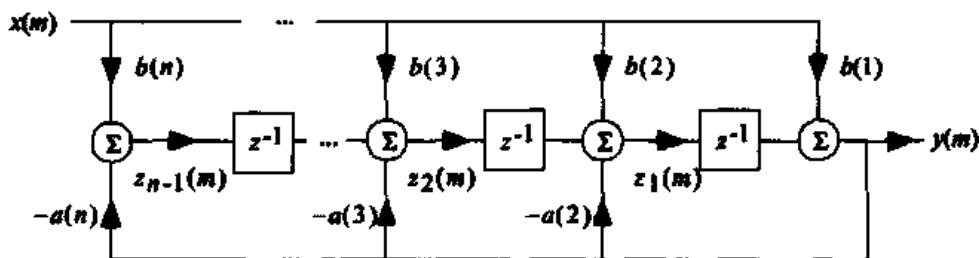


图 9-11 滤波算法框图

该函数的调用格式有：

- 1)  $y = \text{filter}(b, a, x)$ : 参数  $x$  是被滤波的数据，向量  $a$  和  $b$  是用来定义滤波器的系数，分别对应滤波器的分母与分子的系数向量。
- 2)  $[y, zf] = \text{filter}(b, a, x)$ : 返回值  $zf$  是移位寄存器的最终状态值。 $zf$  的主要用途是，对特别长的信号需要分段滤波时，使衔接不影响总的滤波质量。
- 3)  $[...] = \text{filter}(b, a, x, zi)$ : 参数  $zi$  指定移位寄存器的初始值，缺省时，认为移位寄存器的初始值全部为 0。参数  $zi$  和  $zf$  的大小是  $\max(\text{length}(b), \text{length}(a)) - 1$ 。
- 4)  $[...] = \text{filter}(b, a, x, zi, dim)$ : 参数  $dim$  指定进行滤波的  $x$  的维数。

下面的程序在原始信号  $x1$  中加入噪声  $x2$ ，然后对合成的信号进行滤波，结果如图 9-12。

```
t = 0:0.0005:0.05;
x1 = sin(80*2*pi*t); % 产生80Hz正弦信号
x2 = 0.2*rand(size(t)); % 添加随机噪声
x = x1 + x2;
A = [1 -1.143 0.4128]; % 滤波器分母系数
B = [0.06745 0.1349 0.06745]; % 滤波器分子系数
```

```

y = filter(B, A, x);      % 滤波
plot(t, x, ':', t, y)    % 滤波前后信号比较, 虚线为滤波前信号波形

```

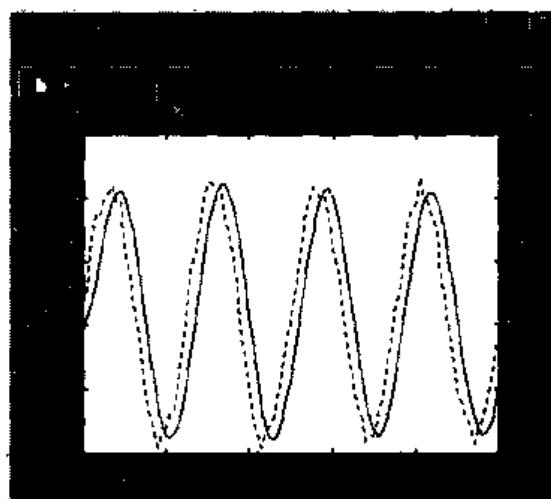


图 9-12 滤波前后信号比较

### 3. 其它滤波函数

除了 filter 函数外, 还有几个相关的滤波函数: filtfilt、fftfilt 和 filter2, 函数 filtfilt 和 fftfilt 位于信号处理工具箱中, 函数 filter2 位于目录\toolbox\matlab\datafun 下。

#### (1) filtfilt

函数 filtfilt 实现零相位数字滤波, 它沿前后两个方向处理输入数据, 从而使得滤波结果没有相位失真而且滤波器的阶数加倍。该函数的调用方式和 filter 相同:

```
y = filtfilt(b, a, x)
```

采用例 9.8 中的数据和滤波器系数, 比较一下 filtfilt 函数和 filter 函数的滤波效果。如图 9-13 所示, 图中虚线是输入数据, 点画线是 filter 函数的输出, 实线是 filtfilt 函数的输出, 可见, 函数 filtfilt 消除了相位失真。

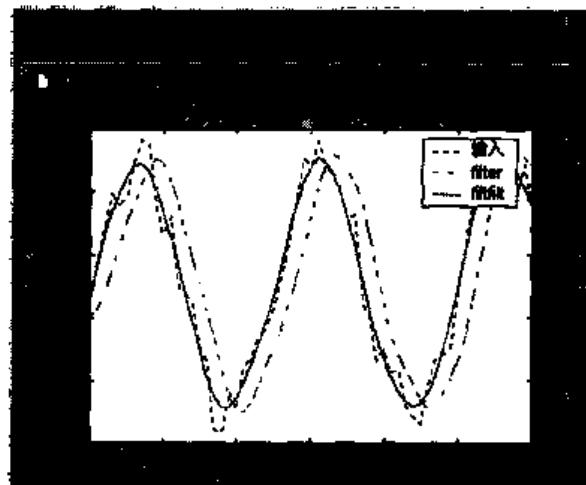


图 9-13 消除滤波的相位失真

#### (2) fftfilt

函数 `fftfilt` 用基于 FFT 的频域滤波技术对数据进行滤波, 它效率比较高, 但只能用于 FIR 滤波器。函数的一般用法为:

```
y = fftfilt(b, x)
```

它的输出等同于 `filter(b, 1, x)`。

### (3) filter2

`filter2` 函数对输入信号做二维数字滤波, 采用二维卷积算法。该函数只能用于 FIR 滤波器, 调用格式如下:

`Y = filter2(B, X, 'shape')`: 矩阵 `X` 是输入的二维数据, 矩阵 `B` 中是二维 FIR 滤波器的系数。返回值 `Y` 的大小由参数 `shape` 设置, 该参数的取值有三种:

`same` (缺省值): 返回卷积的中间部分, 和输入矩阵 `X` 的大小相同;

`full`: 返回二维卷积的全部结果, `size(Y) > size(X)`。

`valid`: 只返回卷积的一部分, 这部分是计算时不带零插值边缘的, `size(Y) < size(X)`。

## 9.4.2 滤波器的分析

### 1. 脉冲响应

当数字滤波器的输入是脉冲信号时, 滤波器的输出 (响应) 就是该滤波器的脉冲响应。脉冲信号就是满足如下关系的信号:

$$x(n) = \begin{cases} 1 & n = 1 \\ 0 & n \neq 1 \end{cases}$$

在 MATLAB 中产生 `N` 个数据的脉冲信号通常用 `x = [1; zeros(n-1, 1)]` 的方式。把产生的脉冲信号带入 `filter` 函数即得到滤波器的脉冲响应。

在信号处理工具箱中还有专用于求滤波器脉冲响应的函数 `impz`, 该函数通常的用法为:

`[h, t] = impz(b, a, n)`: 参数 `n` 指定脉冲信号的长度, 返回参数 `h` 是脉冲响应数据, `t` 是相应的时间向量。如果不带返回参数, 那么该函数将直接画出脉冲响应的图形。

本例首先设计一个 4 阶巴特沃思滤波器, 然后计算其脉冲响应并画图, 结果如图 9-14。

```
[b, a] = butter(4, 0.05);
impz(b, a, 100);
```

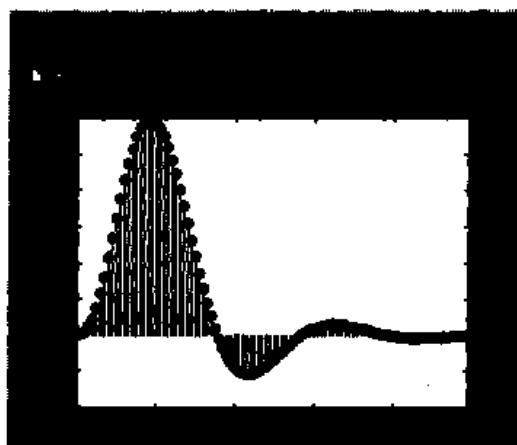


图 9-14 滤波器脉冲响应

## 2. 频率响应

信号处理工具箱中的函数 freqz 和 freqs 分别用于计算数字滤波器和模拟滤波器的复频率响应。

### (1) 数字滤波器频率响应

函数 freqz 采用 FFT 算法计算数字滤波器的复频率响应  $H(e^{j\omega})$ ，即滤波器的传递函数  $H(z)$  当  $z = e^{j\omega}$  时的值。该函数的用法有如下几种：

- $[h, w] = \text{freqz}(b, a, n)$ : 计算数字滤波器的 n 点复频率响应，b 和 a 是滤波器系数向量，整数 n 是频率响应的点数，输出参数 h 是复频率响应，w 是频率点。
- $[h, f] = \text{freqz}(b, a, n, Fs)$ : 参数 Fs 是采样频率，单位 Hz，缺省值为 1。输出参数 f 包含 0 到  $F_s/2$  (奈奎斯特频率) 的实际频率点，长度为 n。
- $[h, w] = \text{freqz}(b, a, n, 'whole')$ : 采用单位圆上的 n 个点，这样输出参数 w 的范围是  $[0, 2\pi]$ 。
- $[h, f] = \text{freqz}(b, a, n, 'whole', Fs)$ : 输出参数 f 的范围是  $[0, Fs]$ 。
- $h = \text{freqz}(b, a, w)$ : 计算在由向量 w (单位 rads/sample) 指定的频率点的复频率响应。
- $h = \text{freqz}(b, a, f, Fs)$ : 计算在向量 f (单位 Hz) 指定的频率点的复频率响应。
- freqz(...): 不带输出参数时，直接绘制出滤波器的幅频响应和相频响应图。

如下命令对一 FIR 滤波器进行频率响应分析，频率范围 0~1000Hz，采样频率 2048Hz，结果如图 9-15。

```
b = fir1(80,0.5,kaiser(81,3));
f = 0:0.1:1000;
freqz(b, 1, f, 2048)
```

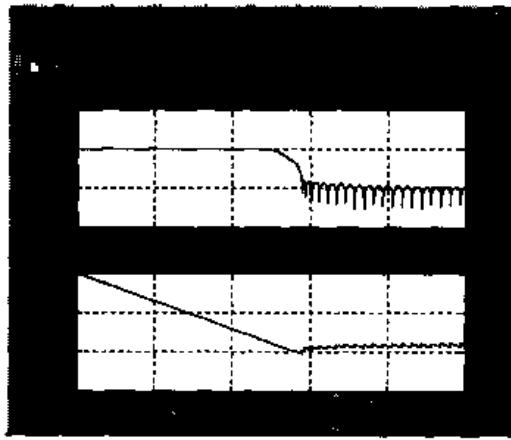


图 9-15 频率响应分析

函数 freqz 计算的复频率响应 h 是一个复数向量，它包含着幅值和相位信息，MATLAB 提供了两个函数 abs 和 angle，分别用于从复频率响应数据中提取幅值信息和相位信息。

如果例中的 freqz 函数返回了复频率响应 h，而没有直接绘图，那么可以用如下一组命令提取幅值和相位信息，并绘制和图 9-15 一样的图形：

```
amp = abs(h);
```

```
pha = angle(h);
subplot(2,1,1); plot(f, amp)
subplot(2,1,2); plot(f, pha)
```

#### (2) 模拟滤波器的频率响应

函数 freqs 计算模拟滤波器的频率响应，它的用法和函数 freqz 类似，请读者参看相关帮助。

#### 3. 群延迟

滤波器的群延迟是滤波器平均延迟的度量，它是滤波器相位响应的负一阶导数，若滤波器的复频率响应是  $H(e^{j\omega})$ ，那么群延迟为：

$$\tau_g(\omega) = \frac{d\theta(\omega)}{d\omega}$$

其中， $\omega$  是频率， $\theta$  是  $H(e^{j\omega})$  的相角。

在 MATLAB 中计算群延迟的函数是 grpdelay，该函数的用法和 freqz 类似，我们看一个例子。

```
[b, a] = butter(5, 0.5);
grpdelay(b, a, 512) % 计算 512 个点的群延迟，结果如图 9-16
```

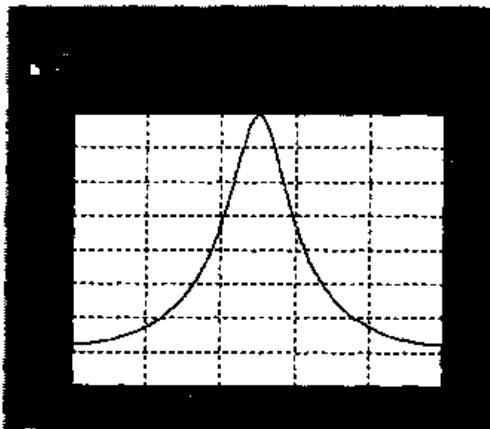


图 9-16 群延迟

#### 4. 零极点分析

函数 zplane 可以在 z 平面上画出线性系统的零点和极点。在图中用“o”表示零点，用“x”表示极点。该函数的用法如下：

`zplane(z, p):` z 和 p 是列向量时，分别为系统的零点和极点。

`zplane(b, a):` b 和 a 是行向量时，分别为系统传递函数分子和分母的系数向量。这种情况下，zplane 用 roots 函数求出系统的根，并画出零极点图。

```
[b, a] = butter(5, 0.7);
zplane(b, a)
```

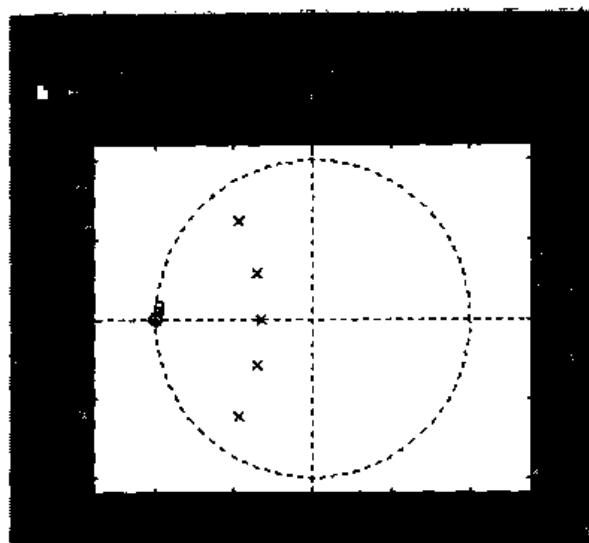


图 9-17 零极点图

## 9.5 滤波器设计

滤波器设计函数我们在前面的例子中已经接触到了，简单讲，滤波器的设计就是为了满足不同的要求而选用不同类型的滤波器，并计算适当的参数。设计滤波器一般的要求是从信号中消除噪声，更高的要求是要滤波器具有指定的通带、阻带或过渡带宽度，有时还要求以最小的滤波器阶次满足一定的性能指标。

本节分别介绍 IIR 滤波器和 FIR 滤波器的设计方法及 MATLAB 中的相关函数。包括数字滤波器和模拟滤波器。

### 9.5.1 IIR 滤波器设计

IIR 滤波器的优点在于，在满足相同性能要求的条件下，它比相应的 FIR 滤波器阶次低很多。IIR 滤波器是非线性相位滤波器。

MATLAB 信号处理工具箱中用于 IIR 滤波器设计的函数如表 9-3 所示，从表中可以看出，MATLAB 的滤波器设计函数主要分为四类，对应滤波器设计的四种方法：经典 IIR 滤波器设计，广义设计，根据频域指标设计 IIR 滤波器（阶次估计），直接设计。

表 9-3 IIR 滤波器设计函数

设计方法	滤波器	函数及调用格式
经典设计	贝塞尔 (Bessel) 模拟滤波器	$[b,a] = \text{besself}(n,W_n,\text{'ftype'})$ $[z,p,k] = \text{besself}(\dots)$ $[A,B,C,D] = \text{besself}(\dots)$
	巴特沃思 (Butterworth)	$[b,a] = \text{butter}(n,W_n,\text{'ftype','s'})$ $[z,p,k] = \text{butter}(\dots)$ $[A,B,C,D] = \text{butter}(\dots)$

(续)

设计方法	滤波器	函数及调用格式
经典设计	切比雪夫 (Chebyshev) I型	$[b, a] = \text{cheby1}(n, R_p, W_n, 'ftype', 's')$ $[z, p, k] = \text{cheby1}(\dots)$ $[A, B, C, D] = \text{cheby1}(\dots)$
	切比雪夫 (Chebyshev) II型	$[b, a] = \text{cheby2}(n, R_s, W_n, 'ftype', 's')$ $[z, p, k] = \text{cheby2}(\dots)$ $[A, B, C, D] = \text{cheby2}(\dots)$
	椭圆 (Elliptic)	$[b, a] = \text{ellip}(n, R_p, R_s, W_n, 'ftype', 's')$ $[z, p, k] = \text{ellip}(\dots)$ $[A, B, C, D] = \text{ellip}(\dots)$
广义设计	广义数字巴特沃思滤波器	$[b, a] = \text{maxflat}(nb, na, W_n)$ $b = \text{maxflat}(nb, 'sym', W_n)$ $[b, a, b1, b2] = \text{maxflat}(nb, na, W_n)$ $[...] = \text{maxflat}(nb, na, W_n, 'design_flag')$
根据频域指标设计	巴特沃思	$[n, W_n] = \text{buttord}(W_p, W_s, R_p, R_s, 's')$
	切比雪夫I型	$[n, W_n] = \text{cheb1ord}(W_p, W_s, R_p, R_s, 's')$
	切比雪夫II型	$[n, W_n] = \text{cheb2ord}(W_p, W_s, R_p, R_s, 's')$
	椭圆	$[n, W_n] = \text{ellipord}(W_p, W_s, R_p, R_s, 's')$
直接设计	递归数字滤波器	$[b, a] = \text{yulewalk}(n, f, m)$

从表中可以看出，每一类滤波器设计函数的用法都大致相同，下面我们分类加以介绍。

### 1. 经典设计函数

利用这些函数可以简单地创建任意阶次的低通、高通、带通和带阻滤波器。

函数中的输入参数 n 指定滤波器的阶数；Wn 是期望的截止频率（注意，要以归一化频率形式给出，即把奈奎斯特频率设为 1，指定的数值是实际频率和奈奎斯特频率的比值）；Rp 是通带的波动范围 (dB)，Rs 是阻带的衰减 (dB)。

ftype 是可选参数，缺省情况下如果 Wn 是标量，则该函数产生低通滤波器；如果 Wn 是两个元素的向量，则产生带通滤波器。若需要设计高通滤波器，则须把参数 ftype 赋值为 high；若把 ftype 赋值为 stop，且参数 Wn 是两个元素的向量，则该函数设计 2n 阶的带阻滤波器。

这些函数的输出都有三种方式：

- [b, a]: 两个元素，b 和 a 分别是滤波器传函的分子和分母的系数向量。
- [z, p, k]: 三个元素，z、p、k 分别为零极点形式表示的滤波器的零点、极点和增益。
- [A, B, C, D]: 四个元素，A、B、C、D 分别表示滤波器状态方程的系数矩阵。

输入参数 s 是可选参数，如果带有该参数，则设计模拟滤波器，否则设计数字滤波器。函数 besself 除外，它只设计模拟滤波器。

```
% 5 阶低通贝塞尔模拟滤波器，通带为 0.2 奈氏频率
```

```
[b, a] = besself(5, 0.2)
```

```
% 4 阶切比雪夫I型带通模拟滤波器，通带 0.4~0.6 奈氏频率，通带波动范围 3dB
```

```
[b, a] = cheby1(4, 3, [0.4 0.6], 's')
```

```
% 4 阶切比雪夫II型高通数字滤波器，阻带衰减 60dB
```

```
[b, a] = cheby2(4, 60, 0.8, 'high')
```

```
% 3阶椭圆带阻数字滤波器，通带波动范围2dB，阻带衰减50dB
```

```
[A, B, C, D] = ellip(3, 2, 50, [0.3 0.7], 'stop')
```

### 2. 广义设计函数

MATLAB 的较新版本中提供了广义数字巴特沃思滤波器的设计函数 maxflat。所谓广义巴特沃思滤波器是指零点数和极点数不同的巴特沃思滤波器。

在函数 maxflat 中要指定两个阶次，一个是分子的阶数 nb，对应零点的个数，一个是分母的阶数 na，对应极点的个数。当 nb = na 时，该函数的作用和 butter 函数相同。

函数 maxflat 中的截止频率 Wn 是半功率频率 (half-power frequency)，即幅频响应为  $1/\sqrt{2}$  的点的频率。

在函数 maxflat 中使用'sym'选项可以设计具有最平特性的 FIR (线性相位) 滤波器。

参数 design\_flag 使用户可以观察滤波器设计结果，它有三个取值：

- trace：以文本的形式显示设计中采用的设计表。
- plots：画出滤波器的幅频特性、群延迟和零极点图。
- both：同时显示文本和图形。

函数如果有四个返回参数，那么后两个参数 b1 和 b2 是两个多项式的系数向量，它们的积等于分子多项式 b，即  $b = \text{conv}(b1, b2)$ 。其中 b1 包含了  $z = -1$  处的所有零点，b2 包含了其它零点。

如下程序设计八个零点，两个极点的巴特沃思数字滤波器，该滤波器的幅频特性、群延迟和零极点图如图 9-18 所示。

```
nb = 8; na = 2;
Wn = 0.1*pi;
[b, a, b1, b2] = maxflat(nb, na, Wn, 'plots')
```

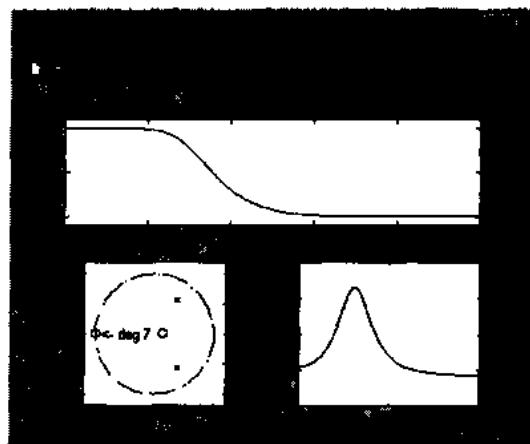


图 9-18 滤波器特性

### 3. 根据频域指标设计

表 9-3 中列出的这一类函数的作用是根据给定要求估计滤波器所需的最小阶次。这些函数和滤波器的一般设计函数结合使用可以设计出比较优化的滤波器。

这些函数的返回值 n 和 Wn 分别是估计出的滤波器最小阶次和相应的截止频率。

输入参数 Wp 是通带截止频率（归一化），Ws 是阻带转角频率（归一化），Rp 是通带波动（dB），Rs 是阻带衰减（dB）。

如果要设计模拟滤波器，可在输入中附加参数's'。

对采样频率是 1000Hz 的数据设计巴特沃思低通滤波器，通带截止频率 100Hz，波动不大于 3dB，从 150Hz 到奈氏频率（采样频率/2）的衰减不小于 30dB。设计程序如下：

```
Wp = 100/500; Ws = 150/500;
```

```
[n, Wn] = buttord(Wp, Ws, 3, 30)
```

```
n =
```

```
8
```

```
Wn =
```

```
0.2034
```

```
[b, a] = butter(n, Wn); % 根据估计结果计算滤波器的系数
```

#### 4. 直接设计

直接设计是在离散域内基于规范法设计 IIR 滤波器，它不受标准低通、高通、带通、带阻等的限制，并以任意可能多的频率响应来设计滤波器。

函数 yulewalk 对给定的频率响应用最小二乘拟合设计递归 IIR 数字滤波器。

该函数的输入参数 n 仍然是滤波器的阶次；参数 f 是从 0 到 1 的频率点向量，1 对应奈氏频率；m 是在向量 f 中指定的频率点上期望的幅值响应，f 和 m 长度必须相同。用命令 plot(f, m) 可以显示要设计的滤波器的理想形状。也就是说，用户只要给出滤波器的形状，就可以用函数 yulewalk 来实现该滤波器。

下面的程序用 yulewalk 函数设计滤波器，并画出期望的和实际的频率响应曲线，如图 9-19 所示。

```
f = [0 0.3 0.4 0.7 0.8 1];
m = [0 0 1 1 0 0];
[b, a] = yulewalk(12, f, m);
[h, w] = freqz(b, a, 256);
plot(f, m, w/pi, abs(h))
```

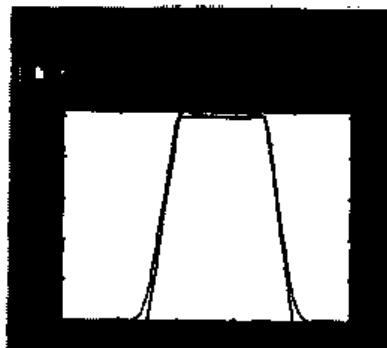


图 9-19 频率响应回比

### 9.5.2 FIR 滤波器设计

FIR（有限冲激响应）滤波器和 IIR 滤波器比较有如下几个主要优点：

- 具有精确的线性相位
- 总是稳定的
- 设计方法一般是线性的
- 硬件实现比较容易
- 滤波器过渡过程具有有限区间

FIR 滤波器的主要缺点是它在同样的条件下，阶次比 IIR 滤波器要高，相应的，其延迟比同样性能的 IIR 滤波器要大得多。

MATLAB 信号处理工具箱中用于 FIR 滤波器设计的函数如表 9-4 所示。

表 9-4 FIR 滤波器设计函数

设计方法	函数及主要用法
窗函数法	<pre>b = fir1(n,Wn,'ftype',window) b = fir1(...,'noscale')</pre> <pre>b = fir2(n,f,m,npt,window) b = fir2(n,f,m,npt,lap) b = fir2(n,f,m,npt,lap,window)</pre> <pre>[n,Wn,beta,ftype] = kaiserord(f,a,dev,Fs) c = kaiserord(f,a,dev,Fs,'cell')</pre>
带过渡带的多带滤波器设计法	<pre>b = fircls(n,f,a,w,'ftype') b = remez(n,f,a,w,'ftype')</pre> <pre>[n,fo,so,w] = remezord(f,a,dev,Fs) c = remezord(f,a,dev,Fs,'cell')</pre>
约束最小二乘法	<pre>b = fircls(n,f,amp,up,lo) fircls(n,f,amp,up,lo,'design_flag')</pre> <pre>b = fircls1(n,wo,dp,ds,'high') b = fircls1(n,wo,dp,ds,'design_flag')</pre>
任意响应法	<pre>b = cremez(n,f,'fresp',w) b = cremez(n,f,{ 'fresp', p1, p2, ... }, w) b = cremez(n,f,a,w) b = cremez(...,'sym') b = cremez(...,'skip_stage2') b = cremez(...,'debug') b = cremez(...,{ 'grid' })</pre>
升余弦法	<pre>b = fircos(n,F0,dF,Fs) b = fircos(n,F0,r,Fs,'rolloff') b = fircos(...,'type',delay,window)</pre>

除了函数 `cremez` 之外，其它所有的 FIR 滤波器设计函数只设计线性相位滤波器，这种滤波器的系数遵从奇对称或偶对称关系。根据这种对称性和滤波器阶次 n 的奇偶，线性相

位滤波器的频率响应有确定的内在限制，这种限制关系如表 9-5 所示。

表 9-5 线性相位滤波器频率响应的限制关系

滤波器类型	滤波器阶次 n	系数对称性	响应 $H(f)$ , $f=0$	响应 $H(f)$ , $f=1$ (奈氏频率)
I型	偶数	偶数: $b(k) = b(n+2-k)$ $k = 1, \dots, n+1$	无限制	无限制
II型	奇数		无限制	$H(1) = 0$
III型	偶数	奇数: $b(k) = -b(n+2-k)$ $k = 1, \dots, n+1$	$H(0) = 0$	$H(1) = 0$
IV型	奇数		$H(0) = 0$	无限制

### 1. 窗函数法

函数 fir1 和 fir2 根据给定的滤波器阶次和理想滤波器的描述，返回理想滤波器的加窗逆傅立叶变换。函数 kaiserord 估计采用 Kaiser 窗设计的 FIR 滤波器的参数。

#### (1) fir1

函数 fir1 是加窗线性相位 FIR 数字滤波器设计的经典方法。它针对标准频率响应。

函数的参数 n、Wn 的含义和用法和 IIR 滤波器的经典设计函数相同；可选参数 window 用来指定设计滤波器的窗函数，是长度为 n+1 的列向量，缺省时是海明窗（Hamming）；这里的参数'ftype'比前面讲到的函数增加了两个取值（以前是'high'和'stop'）：

'DC-1': 使得多带滤波器的第一个带是通带。

'DC-0': 使得多带滤波器的第一个带是阻带。

参数'noscale'的作用是取消默认的缩放比例。

#### (2) fir2

函数 fir2 针对任意形状的线性频率响应设计加窗 FIR 数字滤波器。

函数的参数 n、f、m 和 IIR 滤波器设计函数 yulewalk 中的参数相同。

函数 fir2 在计算之前先要把频率响应数据进行插值，参数 npt 指定插值后的数据点数，缺省值为 512；如果两个相邻的频率点相等，那么需要有一个很陡但平滑的过渡带，参数 lap 设置这个过渡带的点数，缺省值是 25。

#### (3) kaiserord

函数 kaiserord 估计滤波器的最小阶次 n、截止频率向量 Wn 和 Kaiser 窗的参数 beta，供函数 fir1 进行滤波器设计。

函数的输入参数 f 是频带边缘频率向量，a 是在各个频率带的期望幅值，它们的长度满足关系  $\text{length}(f) = 2 * \text{length}(a) - 1$ ，f 和 a 共同定义了一个分段的频率响应关系。dev 是和 a 大小相同的向量，指定每个频带的最大允许误差或输出滤波器的频率响应和期望值的偏差。

输出参数 ftype 是传递给函数 fir1 用于指定滤波器类型的。

如果在输入参数中使用字符串'cell'，那么函数的返回值将是一个单元数组，它的元素是传递给函数 fir1 的参数。

要求设计低通滤波器，通带 0~200Hz，阻带 300~1000Hz，通带波动 7%，阻带衰减 60dB。设计过程如下：

fs = 2000;

% 采样频率

```

f = [200, 300];
a = [1, 0];
dev = [0.07, 0.001];      % 阻带衰减 60dB 相当于 0.001
[n, Wn, beta, ftype] = kaiserord(f, a, dev, fs);
hh = fir1(n, Wn, ftype, kaiser(n+1, beta));
freqz(hh)                % 画出滤波器幅频和相频特性

```

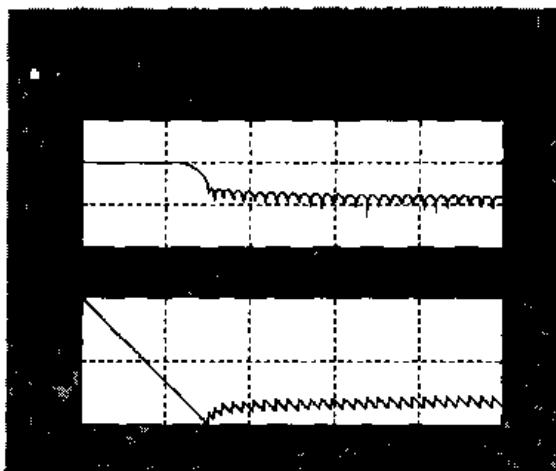


图 9-20 窗函数法设计 FIR 滤波器

## 2. 带过渡带的多带滤波器设计法

函数 `firls` 和 `remez` 提供了带过渡带的多带滤波器设计方法，这两个函数的语法是基本一样的。函数 `remezord` 为 `remez` 函数提供阶次估计。

函数 `firls` 是 `fir1` 和 `fir2` 的扩展，它的设计准则是使期望频率响应和实际频率响应之间的整体方差最小。

函数 `remez` 使用 Parks-McClellan 算法，该算法用契比雪夫(Chebyshev)逼近理论和 Remez 交换法设计在期望的和实际的频率响应之间配合最好的滤波器。因而这种滤波器是期望的和实际的频率响应误差最小的滤波器，从这个意义上说，它是最优滤波器，也称为最小滤波器。而且这种滤波器在频率响应中表现出等波纹特性，因此也称等波纹滤波器。

函数 `firls` 和 `remez` 的输入参数 `n`、`f`、`a` 是必须的，`n` 是阶数，`f` 是频率点向量(成对的)，`a` 是在每个频率点上的幅值，`a` 和 `f` 长度相等。在这种缺省方式下，它们设计 I型或 II型线性相位滤波器，这取决于滤波器的阶次是奇数还是偶数。

参数 `w` 在各个频带内设置权值，每个频带对应 `w` 中的一个元素，`w` 的长度是 `a` 和 `f` 的一半。

参数'ftype'的取值有两种，分别为：

`hilbert`: 设计奇对称线性相位滤波器，III型(偶数阶)和IV型(奇数阶)。这种滤波器包含希尔伯特变换器，其在整个频带内幅值为 1。

`differentiator`: 采用特殊的加权技术设计 III型和 IV型滤波器，即微分器。在非零频带，函数用系数  $1/f$  对误差加权，以使得低频部分误差比高频小很多。

```
f = [0 0.1 0.15 0.3 0.35 0.6 0.65 1];
```

```

a = [0.6 0.6 0 0 1 1 0 0];
weight = [5 1 5 1];      % 通带比阻带的波动小 5 倍
b = remez(40, f, a, weight);
[h, w] = freqz(b, 1, 512);
plot(f, a, w/pi, abs(h))

```

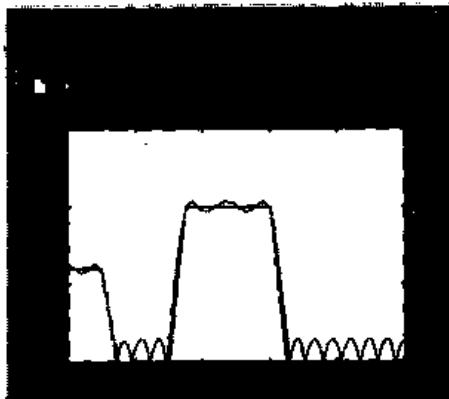


图 9-21 多带滤波器

### 3. 约束最小二乘法

采用约束最小二乘法 (Constrained Least Squares, CLS) 设计 FIR 滤波器可以不用明确定义过渡带的幅值响应，而只要指定截止频率或通带和阻带的边缘频率，这样，CLS 法可以间接定义过渡带。

CLS 法的关键特性是能够定义幅值响应中允许波动的上下限，给定这个约束以后，该方法在滤波器响应的整个频率范围上应用最小二乘的误差最小化技术，误差最小化包括理想滤波器的任何区域。

工具箱中的函数 fircls 和 fircls1 实现这种设计技术，fircls1 设计低通和高通线性相位滤波器，fircls 设计多带 FIR 滤波器。

#### (1) fircls

函数 fircls 产生一个 n 阶 FIR 滤波器，其中的参数 f 是频带之间的过渡频率，amp 是在各个频带的期望幅值，它们的长度满足关系  $\text{length}(a) = \text{length}(f)-1$ 。

up 和 lo 是和 amp 长度相同的向量，它们指定每个频带频率响应的上下限。

参数 design\_flag 的取值包括 trace、plots、both，前面的函数 maxflat 中已经讲过。

#### (2) fircls1

参数 wo 是归一化截止频率，dp 是相对于 1 的最大通带衰减，ds 是相当于 0 的最大阻带衰减。如果带参数'high'则生成高通滤波器，缺省是低通滤波器。

参数 design\_flag 和 fircls 函数中相同。

用函数 fircls 设计例 9.19 中的滤波器，结果如图 9-22 所示。

```

n = 40;                      % 40 阶
f = [0 0.15 0.35 0.65 1];    % 注意比较这里的 f、amp 和例 9.19 中的 f、a 的区别
amp = [0.6 0 1 0];

```

```
up = [0.61 0.03 1.01 0.03]; % 通过设置每个频带频率响应的上下限来设置频带的波动
lo = [0.59 -0.03 0.99 -0.03];
b = fircls(n,f,amp,up,lo,'plots');
```

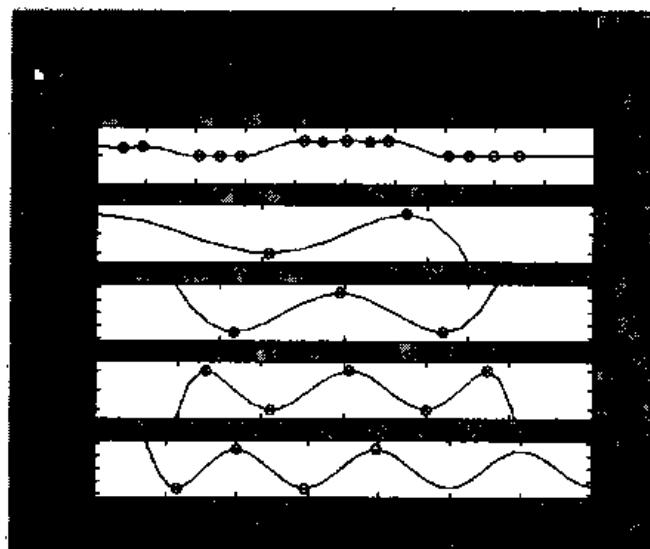


图 9-22 用 CLS 法设计滤波器

#### 4. 任意响应法

`cremez` 函数提供了根据任意复杂响应设计 FIR 滤波器的方法。它在指定滤波器频率响应的方式上和其它滤波器设计函数不同：它接受计算滤波器响应的函数名，因而用该函数设计滤波器具有通用性，而且功能强大。

这种设计方法可以用来设计非线性相位 FIR 滤波器，不对称频率响应滤波器（复系数），或用户定制频率响应的对称滤波器。

`cremez` 函数采用扩展的 Remez 交换算法使契比雪夫误差最优化。如果 Remez 交换算法不能获取最优解，则该算法会转换为 ascent-descent 算法，以保证收敛到最优解。

函数中的参数 `f` 是频带边缘向量（成对），范围从-1 到 1。`w` 是每个频带的非负加权系数，向量 `w` 的长度是 `f` 的一半。

参数 `fresp` 指定返回期望频率响应的函数，`p1`、`p2` 是该函数的可选参数。`fresp` 的可能取值有：

(1) `lowpass`, `highpass`, `bandpass`, `bandstop`

这些取值的语法格式为（以 `lowpass` 为例）：

`b = cremez(n,f,'lowpass',...)` 或 `b = cremez(n,f,{'lowpass',d},...)`

其中参数 `d` 设置滤波器的相位延迟为 `n/2+d`。

(2) `multiband`

设计多带滤波器，语法为：

`b = cremez(n,f,{'multiband',a},...)` 或 `b = cremez(n,f,{'multiband',a,d},...)`

参数 `a` 是和 `f` 对应的期望幅值向量。

(3) `differentiator`

设计线性相位微分器，这种设计的零频率必须在过渡带，频带加权设置为和频率成反

比。语法为：

`b = cremez(n,f,'differentiator',Fs,...)` 或 `b = cremez(n,f,'differentiator',Fs,d,...)`

采样频率 `Fs` 用于确定微分器响应的斜率，缺省时 `Fs=1`。

#### (4) hilbfilt

设计线性相位希尔伯特变换滤波器，要求零频率必须在过渡带。语法为：

`b = cremez(n,f,'hilbfilt',...)` 或 `b = cremez(N,F,'hilbfilt',d,...)`

参数'sym'在设计的冲激响应上强加一个对称约束，它的可能取值有：

- 'none': 没有对称约束。如果没有负频带，那么它是缺省值
- 'even': 偶数冲激响应。当参数 `fresp` 取值为 `highpass`、`lowpass`、`bandpass`、`bandstop` 和 `multiband` 时，这是缺省值。
- 'odd': 奇数冲激响应。参数 `fresp` 取值为 `Hilbert` 和 `differentiator` 时，这是缺省值。
- 'real': 频率响应共轭对称。

选用参数'skip\_stage2'则不采用第二阶段的算法（ascent-descent 算法），这样会加快计算速度，但有可能使结果的精确度降低。

参数'debug'允许在滤波器设计中显示中间结果，它的取值可以是'trace'、'plots'、'both'或'off'。缺省时是'off'。

{lgrid}控制频率点的密度，缺省值是 25，注意它是  $1 \times 1$  的单元数组。

如下程序设计一低通滤波器，设计过程及结果如图 9-23 所示。

```
n = 40;
f = [-1 -0.4 -0.3 0.6 0.7 1];
b = cremez(n, f, 'lowpass', 'plots');
```

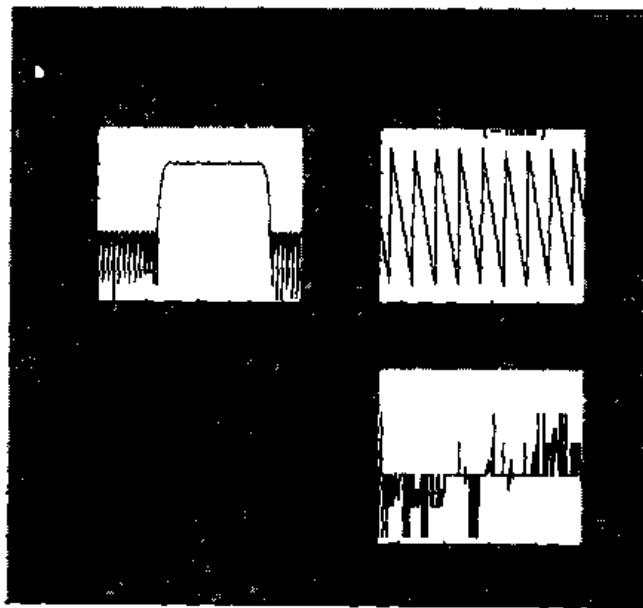


图 9-23 任意响应法设计过程

## 5. 升余弦法

升余弦法的函数 `firrcos` 设计带有升余弦过渡带的 FIR 滤波器。

$F_0$  是滤波器的截止频率， $df$  是过渡带带宽， $F_s$  是采样频率（缺省值是 2，按照归一化的形式，奈氏频率为 1），它们的单位都是 Hz。 $df$  必须足够小，以满足  $0 < F_0 \pm df/2 < F_s/2$ 。输出  $b$  中的系数是归一化的，使得名义上的通带增益总是为 1。

当函数中带有参数'rolloff'时，第三个输入参数  $r$ （代替  $df$ ）是频率响应下降的系数，范围在 0~1 之间。

参数'type'指定升余弦滤波器的类型，它的取值有两种：normal（缺省）指正规升余弦滤波器；sqrt 指均方差升余弦滤波器。

delay 在  $[0, n+1]$  区间指定整数延迟，缺省值是  $n/2$ （ $n$  是偶数）或  $(n+1)/2$ （ $n$  是奇数）。

window 是加窗的系数，它必须是长度为  $n+1$  的列向量。缺省时使用矩形窗。

## 9.6 统计信号处理和谱分析

信号处理工具箱提供了对随机信号进行估计的重要函数。具体包括离散信号的相关、协方差和谱密度函数。本节将介绍相关和协方差函数，并讨论功率谱估计的数学方法。

### 9.6.1 相关和协方差

#### 1. 基本概念

对于平稳随机过程  $x_n$  和  $y_n$ ，它们的互相关定义为：

$$r_{xy}(m) = E\{x_n y_{n+m}^*\}$$

这里的  $-\infty \leq n \leq +\infty$ ， $E$  是数学期望。互协方差就是去掉了均值的互相关，它的定义为：

$$C_{xy}(m) = E\{(x_n - \mu_x)(y_{n+m}^* - \mu_y^*)\}$$

等价于

$$C_{xy}(m) = r_{xy}(m) - \mu_x \mu_y^*$$

在实际应用中，不可能得到无限长的随机过程，而必须对有限长的序列进行估计。对于  $N$  个采样点的随机序列  $x_n$  和  $y_n$ ，常用的估计是确定性的互相关序列：

$$\hat{R}_{xy}(m) = \begin{cases} \sum_{n=0}^{N-|m|-1} x_n y_{n+m}^* & m \geq 0 \\ \hat{R}_{yx}^*(-m) & m < 0 \end{cases}$$

这里我们假设  $x_n$  和  $y_n$  从 0 到  $N-1$ ， $\hat{R}_{yx}^*(-m)$  从  $-(N-1)$  到  $N-1$ 。

#### 2. 函数用法

函数 `xcov` 和 `xcov` 分别用于估计随机过程的自互相关和自互协方差。函数 `xcov` 采用高

效的基于 FFT 的算法估计上式的和。函数的调用格式有如下几种：

1)  $c = \text{xcorr}(x, y)$ : 计算长为 N 的向量 x 和 y 的互相关序列，结果 c 长度为  $2N-1$ 。如果向量 x 和 y 的长度不同，则较短的向量补零以和另一个向量长度相同。

2)  $c = \text{xcorr}(x)$ : 计算向量 x 的自相关序列。如果 x 是  $N \times P$  的矩阵，则 c 是  $2N-1$  行  $P^2$  列的矩阵，c 中的列对应矩阵 x 中的列按照各种可能组合计算出的自相关序列。

3)  $c = \text{xcorr}(x, y, , 'option')$  或  $c = \text{xcorr}(x, \text{maxlags}, 'option')$ : 参数 maxlags 指定最大延迟，返回的相关序列 c 的延迟范围是  $[-\text{maxlags} : \text{maxlags}]$ ，长度为  $2 * \text{maxlags} + 1$ 。 $'option'$  是相关的标准化选项，其可能的取值有：

- biased: 有偏估计。
- unbiased: 无偏估计。
- coeff: 相关序列归一化，零延迟的自相关为 1。
- none: 使用原始计算结果（缺省值）。

(4)  $[c, lags] = \text{xcorr}(\dots)$ : 返回的向量 lags 是估计相关序列的延迟向量的位置，如果指定了 maxlags，则 lags 的范围是  $[-\text{maxlags} : \text{maxlags}]$ ，否则其范围是  $[-N+1 : N-1]$ 。

在所有这些情况下，计算的自相关或互相关序列的零延迟都位于序列的中间，即第 maxlags+1 个元素或第 maxlags+1 行，如果没有指定 maxlags，则是第 N 个元素或第 N 行。

函数 xcov 和 xcorr 用法相同。

```
x = [1 2 3]; y = [3 2 1];
xy = [1 2 3; 3 2 1]'; % 给定以向量x和y为列向量的3行2列矩阵xy
xcorr(x) % 计算向量x的原始自相关序列
ans =
    3.0000    8.0000   14.0000    8.0000    3.0000
xcorr(x, 'coeff') % 向量x的归一化自相关序列
ans =
    0.2143    0.5714    1.0000    0.5714    0.2143
xcorr(y, 'coeff') % 向量y的归一化自相关序列
ans =
    0.2143    0.5714    1.0000    0.5714    0.2143
[c, lags]=xcorr(x, y, 'coeff') % 向量x和y的归一化互相关序列，同时返回延迟向量
c =
    0.6429    0.8571    0.7143    0.2857    0.0714
lags =
    -2     -1      0      1      2
xcorr(xy, 'coeff') % 对矩阵xy的各列求相关，结果中的每一列是一种组合方式
ans =
    0.2143    0.6429    0.0714    0.2143
    0.5714    0.8571    0.2857    0.5714
    1.0000    0.7143    0.7143    1.0000
```

0.5714	0.2857	0.8571	0.5714
0.2143	0.0714	0.6429	0.2143

### 9.6.2 谱分析

谱分析基于有限的数据寻找信号、随机过程或系统的频率成分。功率谱估计在很多应用中都是很有用的，比如检测淹没在宽带噪声中的信号。

平稳随机信号的功率谱密度（PSD）在数学上是相关序列的离散傅立叶变换：

$$P_{xx}(\omega) = \sum_{m=-\infty}^{\infty} r_{xx}(m) e^{-j\omega m}$$

功率谱密度函数在一个频率段的积分等于信号在此频率段的能量。

PSD 是互谱密度（CSD）函数的特殊情况，互谱密度的定义为：

$$P_{xy}(\omega) = \sum_{m=-\infty}^{\infty} r_{xy}(m) e^{-j\omega m}$$

PSD 估计的方法很多，大致可以分为参数化和非参数化两种。信号处理工具箱中提供的是由 Welch 提出的最普遍采用的非参数化方法，在此基础上又出现了很多现代的非参数化方法，比如 MTM 法（multitaper method）、MUSIC（multiple signal classification）法和特征向量法。Yule-Walker 自回归（AR）法和 Burg 法都是参数化方法。MATLAB 中提供的 PSD 估计方法及其相应函数如表 9-6 所示。

表 9-6 PSD 估计方法及实现函数

方法	函数	方法	函数
Burg 法	pburg	MTM 法	pmtm
协方差法	pcov	MUSIC 法，特征向量法	pmusic
改进的协方差法	pmeov	Yule-Walker 自回归法	pyulear
Welch 法	pwelch(psd), csd, tfe, cohere		

在下面几节中我们将分别介绍这几种方法在 MATLAB 中的实现。

### 9.6.3 Welch 方法及函数

#### 1. Welch 方法

按照定义估计功率谱密度的方法是求出来样数据的离散傅立叶变换，然后将结果的幅值平方，这种估计方法称为周期图法。当采样点的数目很大时，周期图法的期望值很接近 PSD 的真值（理论值），但周期图法的问题是方差较大，而且采样点数增加方差不会减小。

把采样数据分段进行估计可以减小周期图法的方差。比如，采样数据  $x(n)$  长度为 512，把它分为互不重叠的四段，每段 128 个点，这时估计出的 PSD 的方差是用 128 个点的数据进行估计的  $1/4$ 。可见，能分的段数越多，结果的方差越小，但信号的长度限制了可分的段数。如果要增加段数，可以使数据段之间部分重叠，但这会导致每一段的方差增大，因而需要在分的段数和重叠率之间加以平衡。

提高周期图法估计效果的另一种方式是对采样数据分段使用非矩形窗，这就是修正周期图法，也就是 PSD 估计的 Welch 方法。

由于非矩形窗在边沿趋近于零，从而减小了分段对重叠的依赖。而且，非矩形窗减小旁瓣的干扰（谱泄漏），增加主瓣的宽度。人们发现，选用合适的窗函数，采用每段长度一半的重叠率能大大降低谱估计的方差。

```

Fs = 512;          % 采样频率
t = 0 : 1/Fs : 1;    % 采样时间
x = sin(2*pi*100*t) + 2*sin(2*pi*300*t) + randn(size(t));    % 正弦信号加随机噪声
w = hanning(256)';           % 256个点的海宁窗
Pxx = ( abs(fft(w.*x(1:256))).^2 + ...
         abs(fft(w.*x(129:384))).^2 + ...
         abs(fft(w.*x(257:512))).^2 ) / (norm(w)^2*3);
plot((0:255)/256*Fs, 10*log10(Pxx))    % 结果如图9-24所示

```

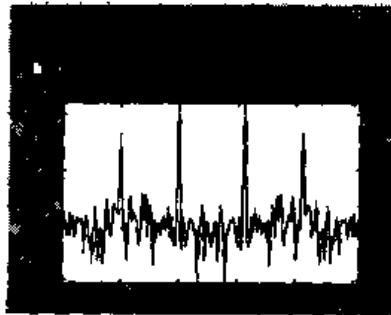


图 9-24 用 Welch 方法估计 PSD

## 2. PSD 和 CSD 估计函数

MATLAB 中用于 PSD 估计的函数是 `pwelch` 或 `psd`，用于 CSD 估计的函数是 `csd`。其中 `psd` 函数在新版本的 MATLAB 中已经被 `pwelch` 函数取代，`pwelch` 函数主要在以下几方面做了改进：谱的幅值和  $1/F_s$  成比例； $F_s$  的缺省值是 1；不再消除趋势项。

函数 `pwelch` 是 PSD 估计中最常用的函数，其用法有如下几种：

1)  $P_{xx} = \text{pwelch}(x)$ : 用 Welch 方法估计  $x$  的 PSD。如果  $x$  是实向量，则 `pwelch` 函数只在正频率上估计谱值，这时，若  $x$  的长度  $n_{fft}$  为偶数，则输出  $P_{xx}$  是长为  $n_{fft}/2+1$  的列向量，若  $n_{fft}$  为奇数，则  $P_{xx}$  长为  $(n_{fft}+1)/2$ 。如果  $x$  是复向量，则 `pwelch` 函数在正负频率上都估计谱值，这时  $P_{xx}$  的长度为  $n_{fft}$ 。

2)  $P_{xx} = \text{pwelch}(x, n_{fft})$ : 参数  $n_{fft}$  指定估计  $x$  的 PSD 时，FFT 运算使用的数据长度，该值确定了估计 PSD 的频率点个数。一般把该参数指定为 2 的幂，以使执行速度更快。把  $n_{fft}$  指定为空矩阵 “[ ]”，则使用缺省值—— $\min(256, \text{length}(x))$ 。

3)  $[P_{xx}, w] = \text{pwelch}(x, n_{fft})$ : 输出参数  $w$  是和估计 PSD 的位置对应的归一化角频率，单位“弧度/秒”。如果  $x$  是实向量， $w$  的范围是  $[0, \pi]$ ；如果  $x$  是复向量， $w$  的范围是  $[0, 2\pi]$ 。由于向量  $w$  和  $P_{xx}$  的长度相同，因而可以用 `plot(w, Pxx)` 命令绘制 PSD 和归一化的角频率之间的关系曲线。

4)  $[P_{xx}, f] = \text{pwelch}(x, \text{nfft}, \text{Fs})$ : 参数  $\text{Fs}$  指定采样频率,  $\text{Fs}$  指定为空矩阵时, 使用缺省值 1Hz,  $P_{xx}$  和  $1/\text{Fs}$  成比例。返回向量  $f$  是和估计 PSD 的位置对应的线性频率, 单位 Hz。如果  $x$  是实向量,  $f$  的范围是  $[0, \text{Fs}/2]$ ; 如果  $x$  是复向量,  $f$  的范围是  $[0, \text{Fs}]$ 。

5)  $[P_{xx}, f] = \text{pwelch}(x, \text{nfft}, \text{Fs}, \text{window}, \text{noverlap})$ : 参数  $\text{window}$  指定窗函数和向量  $x$  加窗的数据段的采样点数, 例如  $\text{hanning}(256)$ 。窗的长度必须小于或等于  $\text{nfft}$ 。把参数  $\text{window}$  设置为一个标量, 则表示使用该长度的海宁窗, 如果设置为空矩阵, 则使用缺省值  $\text{hanning}(\text{nfft})$ 。参数  $\text{noverlap}$  指定数据重叠的长度, 指定为空矩阵时, 使用缺省值 0。

6)  $[P_{xx}, P_{xxc}, f] = \text{pwelch}(x, \text{nfft}, \text{Fs}, \text{window}, \text{noverlap}, p)$ : 这里的参数  $p$  是 0 到 1 之间的标量, 指定 PSD 估计的置信度, 返回值  $P_{xxc}$  是对于该置信度的 PSD 估计值。 $P_{xxc}$  是一个两列的矩阵, 和  $P_{xx}$  长度相等, 由  $P_{xxc}$  的两列确定的区间以概率  $p$  覆盖在 PSD 的真值上。 $p$  的缺省值为 0.95。

7)  $[P_{xx}, P_{xxc}, f] = \text{pwelch}(..., \text{'range'})$ : 参数 ' $\text{range}$ ' 指定包含在  $f$  中的频率值的范围, 它的取值有:

'half': 如果未指定  $\text{Fs}$ ,  $x$  是实向量或复向量都在区间  $[0, \pi]$  上计算 PSD。如果指定了  $\text{Fs}$ ,  $x$  是实向量或复向量都在区间  $[0, \text{Fs}/2]$  上计算 PSD, 如果  $\text{Fs}$  指定为空矩阵 (缺省值 1Hz), 则该区间为  $[0, 1/2]$ 。

'whole': 如果未指定  $\text{Fs}$ ,  $x$  是实向量或复向量都在区间  $[0, 2\pi]$  上计算 PSD。如果指定了  $\text{Fs}$ ,  $x$  是实向量或复向量都在区间  $[0, \text{Fs}]$  上计算 PSD, 如果  $\text{Fs}$  指定为空矩阵, 则该区间为  $[0, 1]$ 。

8)  $\text{pwelch}(x, ...)$ : 没有输出参数时, 绘制 PSD 和频率的关系曲线。

9)  $\text{pwelch}(..., \text{'magunits'})$ : 用参数 ' $\text{magunits}$ ' 指定 PSD 图形幅值坐标轴的单位, 包括:

'db': 分贝 (dB), 缺省值。

'squared': 线性单位 (幅值的平方)。

函数  $\text{csd}$  估计两个输入信号的互谱密度, 其用法和  $\text{pwelch}$  类似, 请读者参看帮助。

下面一段程序在置信度为 0.9 的区间上估计有色噪声  $x_n$  的 PSD 并画图, 结果如图 9-25 所示。

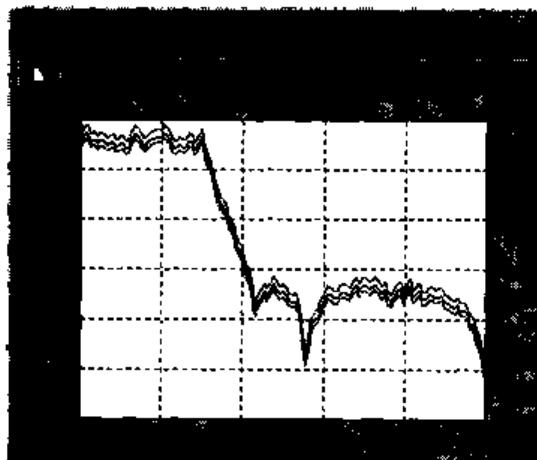


图 9-25 PSD 估计

```
[b, a] = ellip(5, 3, 60, 0.3); % 设计5阶椭圆滤波器
r = randn(2048, 1); % 产生白噪声
xn = filter(b, a, r); % 对白噪声滤波(染色), 得到信号xn
% 估计xn的PSD, 采样频率1000, nfft为256, 窗函数用缺省值, 置信度0.9
pwelch(xn, 256, 1000, [], 0, 0.9)
```

### 3. 传递函数估计

Welch 方法的应用之一是非参数化系统辨识。假定  $H$  是一个线性时不变系统,  $x(n)$  和  $y(n)$  分别是该系统的输入和输出, 则  $x(n)$  和  $y(n)$  的互谱密度 CSD 与  $x(n)$  的自谱密度 PSD 之间有如下关系:

$$P_{xy}(\omega) = H(\omega)P_{xx}(\omega)$$

因而,  $x(n)$  和  $y(n)$  之间传递函数的估计是:

$$\hat{H}(\omega) = \frac{\hat{P}_{xy}(\omega)}{\hat{P}_{xx}(\omega)}$$

用这种方法可以估计幅值和相位信息。MATLAB 中的函数 `tfe` 就采用这种方法估计传递函数, 它用 Welch 方法计算 CSD 和 PSD, 然后求得它们的比, 即输入与输出的传递函数估计。该函数的用法和 `csd` 函数基本相同。

这里对例 9.24 中的滤波器传函进行估计, 并与其实际传递函数比较, 结果如图 9-26 所示。

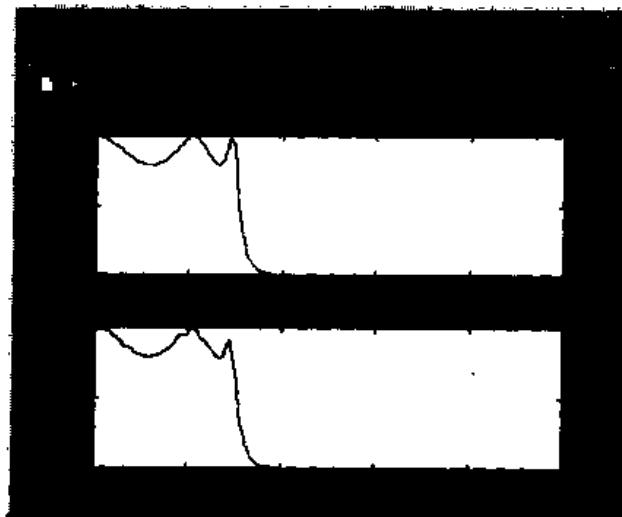


图 9-26 传函估计

```
Fs = 1000; nfft = 256; noverlap = nfft/2;
win = hanning(nfft);
[b, a] = ellip(5, 2, 50, 0.3);
r = randn(2048, 1); % 系统的输入信号
xn = filter(b, a, r); % 滤波后的信号, 即系统的输出信号
```

```
[HH, f] = tfe(r, xn, nfft, Fs, win, nooverlap); % 估计传函, HH是估计值, f是频率点
H = freqz(b,a,f,Fs); % 根据滤波器参数计算实际传函
subplot(2,1,1), plot(f,abs(H))
title('实际传递函数')
subplot(2,1,2), plot(f,abs(HH))
title('估计传递函数')
```

#### 4. 相干函数

对于给定的信号  $x(n)$  和  $y(n)$ , 相干函数定义为:

$$C_{xy}(\omega) = \frac{|P_{xy}(\omega)|^2}{P_{xx}(\omega)P_{yy}(\omega)}$$

信号  $x(n)$  和  $y(n)$  的相干函数表示信号  $y(n)$  在多大程度上来源于  $x(n)$ , 它反映的是信号的频域特性, 取值在 0 到 1 之间, 数值越大, 表示  $x(n)$  和  $y(n)$  的相干性越强。

MATLAB 中求信号相干的函数是 `cohere`, 该函数的用法和函数 `csd`、`tfe` 基本相同。

我们来估计例 9.25 中输入信号  $r$  和输出信号  $xn$  的相干函数, 命令如下, 结果如图 9-27 所示。

```
cohere(r, xn, nfft, Fs, win, nooverlap)
```

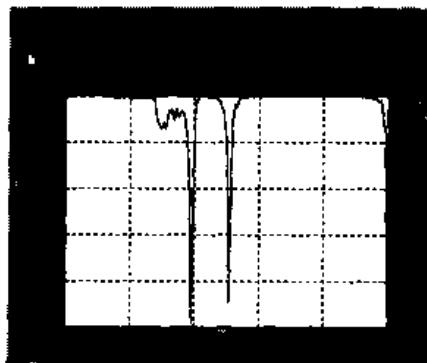


图 9-27 相干估计

#### 9.6.4 其它 PSD 估计方法

##### 1. MTM 法 (multitaper method)

MTM 法使用正交的窗 (离散长球形序列, 又称为 Slepian 序列) 获取近似独立的 PSD 估计, 然后把这些估计结合以得到最终的估计。这种估计显示了更多的自由度, 并且比较容易量化偏差和方差之间的平衡。很多谱估计方法只用一个窗, 这样会使序列开始和结尾处的一些信息丢失。在 MTM 法中, 增加的窗用于复原一些丢失的信息。

MTM 法最主要的参数是时间和带宽的积——NW, 这个参数直接关系到谱估计的窗的个数, 窗的个数总是  $2*NW - 1$  个。这意味着, 随着 NW 的增大, 会有更多的谱估计, 从而估计的方差将减小。然而, 每个窗的带宽也和 NW 成正比, 随着 NW 的增大, 谱泄漏会增大, 而且整个谱估计的结果将有更大的偏差。通常, 对于每一组数据, 总有一个 NW 值能

够在偏差和方差之间取得最佳的平衡。

采用 MTM 法估计 PSD 的函数 pmtm 用法和 pwelch 函数类似，其完整调用格式为：

$[P_{xx}, P_{xxc}, f] = \text{pmtm}(x, nw, nfft, Fs, 'method', p)$ ：这里和函数 pwelch 不同的参数是 nw 和'method'。前面已经介绍过，nw 是时间和带宽的乘积，缺省值为 4，其它几个典型的取值是 2、5/2、3、7/2。参数'method'指定把单独的谱估计结合起来的算法，包括

- adapt：Thomson 自适应非线性组合算法，缺省值。
- unity：相同加权的线性组合。
- eigen：特征值加权的线性组合。

本例采用 MTM 法估计 PSD，并比较 nw 取不同值时的估计效果，如图 9-28 所示。从图中可以看出，nw 越大（下面的图），曲线越平滑，说明方差比较小；但同时，波峰变宽，说明谱泄漏比较大。

```
Fs = 1000; t = 0 : 1/Fs : 1;
x = sin(2*pi*200*t) + randn(size(t));
[Pxx1, f] = pmtm(x, 3/2, 512, Fs); % nw = 3/2
[Pxx2, f] = pmtm(x, 9, 512, Fs); % nw = 9
subplot(2,1,1), plot(f, 10*log10(Pxx1))
subplot(2,1,2), plot(f, 10*log10(Pxx2))
```

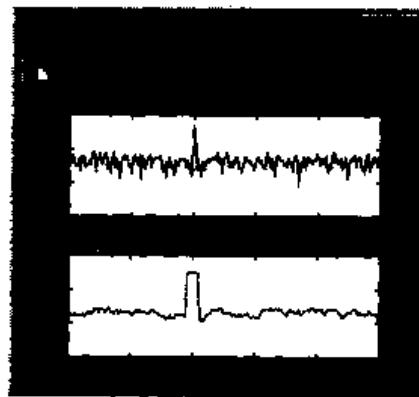


图 9-28 MTM 法谱估计

## 2. MUSIC 法和特征向量法

这两种方法都是基于特征分析和自相关矩阵的频率估计法。这种谱分析方法把相关或数据矩阵中的信息分类，把信息分配到信号子空间或噪声子空间。

MUSIC 法依据的公式为：

$$P_{music}(f) = \frac{1}{e^H(f) \left( \sum_{k=p+1}^N v_k v_k^H \right) e(f)} = \frac{1}{\sum_{k=p+1}^N |v_k^H e(f)|^2}$$

式中，N 是特征向量的维数， $v_k$  是输入信号相关矩阵的 k 阶特征向量，p 是信号子空间的维数，向量  $e(f)$  由复指数组成，因而，内积  $v_k^H e(f)$  等于傅立叶变换。实际计算一般采

用第二种形式，它计算每一个  $v_k$  的傅立叶变换，然后把幅值的平方相加。

在特征向量法中，用相关矩阵的特征值  $\lambda_k$  对求和进行加权，公式如下：

$$P_{ev}(f) = \frac{1}{\left( \sum_{k=p+1}^N |v_k^H e(f)|^2 \right) / \lambda_k}$$

这种方法依赖于信号矩阵的奇异值分解 (SVD)，它用 eig 函数分析相关矩阵。

在 MATLAB 中，这两种估计方法使用的函数相同，都是 pmusic，该函数的用法如下：

1)  $[Pxx, f] = \text{pmusic}(x, p)$  或  $[Pxx, f] = \text{pmusic}(x, [p \text{ thresh}])$ ：这里的输入信号  $x$  对于不同的格式有几种不同的含义：

- 行向量或列向量——表示过程的某一观察值。
- 矩阵（可能是方阵）——每一列是过程的一个观察值。
- 方阵，同时指定参数'corr'（见后）——表示相关矩阵。

第二个参数可以是标量或两个元素的向量。如果只指定了  $p$ ，则信号子空间的维数是  $p$ 。如果指定了  $[p \text{ thresh}]$ ，则把  $\text{thresh}$  乘最小特征值  $\lambda_{\min}$ ，在  $\lambda_{\min} * \text{thresh}$  界限之下的特征值分配到噪声子空间，这时， $p$  是信号子空间的最大维数。

2)  $[Pxx, f] = \text{pmusic}(x, [p \text{ thresh}], nfft, Fs, \text{window}, \text{noverlap})$ ：参数 nfft、Fs、window、noverlap 的含义和函数 pwelch 相同。

3)  $[Pxx, f] = \text{pmusic}(x, \dots, 'corr')$ ：强制  $x$  为相关矩阵，这时参数 window 和 nooverlap 要忽略。

4)  $[Pxx, f] = \text{pmusic}(x, \dots, 'ev')$ ：选择 MUSIC 估计的特征向量变量。

5)  $[Pxx, f, \text{evecs}, \text{svals}] = \text{pmusic}(x, \dots)$ ：返回参数 evecs 是跨越噪声子空间的特征向量矩阵（每列一个），svals 是奇异值（平方）向量或相关矩阵的特征值向量（当指定选项'corr'时）。

### 3. Yule-Walker AR 法

Yule-Walker AR 法又称为自相关法，通过最小化前向预报误差来给输入信号配置适合的自回归模型 (AR 模型)。这种方法用自相关法解 AR 模型的参数。Yule-Walker AR 估计通过如下正则方程的解获得：

$$\begin{bmatrix} r(1) & r^*(2) & \cdots & r^*(n) \\ r(2) & r(1) & \cdots & r^*(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(n) & r(n-1) & \cdots & r(1) \end{bmatrix} \begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(n+1) \end{bmatrix} = \begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(n+1) \end{bmatrix}$$

式中， $a = [1 a(2) \dots a(n+1)]$  是自回归系数向量，向量  $r = [r(1) r(2) \dots r(n+1)]$  中的元素是相关系数。左边的自相关矩阵是厄密共轭而且正定的。

Yule-Walker AR 法 PSD 估计的公式为：

$$P_{YuleAR}(f) = \frac{1}{|a^H e(f)|^2}$$

式中  $e(f)$  是复数正弦曲线。

工具箱中 Yule-Walker AR 法 PSD 估计函数是 `pyulear`, 其完整的调用格式如下:

```
[Pxx, freq] = pyulear(x, p, nfft, Fs, 'range')
```

这里, 参数  $p$  是全极点滤波器模型的阶数, 其余参数和函数 `pwelch` 的用法相同。

#### 4. Burg 法

Burg 法通过使前向和后向预报误差最小化来给信号配置适合的自回归 (AR) 模型, 同时强制 AR 参数满足 Levinson-Durbin 递归关系。Burg 法和其它 AR 估计法相比, 避免了计算自相关函数, 而直接估计映射系数。

Burg 法的主要优点是可以从噪声水平和信噪比都比较低的信号中提取正弦信号, 并且可以用比较少的数据记录进行估计, 这些情况下, 估计的结果很接近真值。另外, Burg 法确保 AR 模型的稳定和高效的计算。对于高阶模型、比较长的数据记录、比较高的信噪比, Burg 法的精确度会降低。

MATLAB 中采用 Burg 法的函数是 `pburg`, 该函数的用法和 `pwelch` 相同, 请读者参看帮助。

本例比较 Burg 法、Yule-Walker AR 法和 Welch 法的 PSD 估计, 结果如图 9-29 所示。实线是 Burg 法的估计结果, 虚线是 Yule-Walker AR 法的估计结果, 点划线是 Welch 法的估计结果。

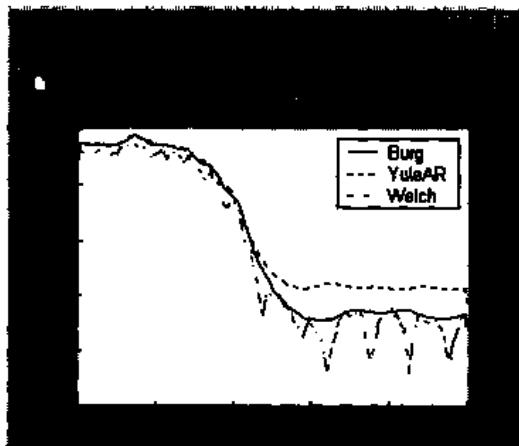


图 9-29 三种方法 PSD 估计比较

```

Fs = 1000;
h = fir1(20, 0.3); % 设计20阶FIR滤波器
r = randn(1024, 1);
x = filter(h, 1, r); % 把随机序列通过滤波器
[Pxx1, f] = pburg(x, 20, [], Fs); % Burg法, 选用20阶AR模型
[Pxx2, f] = pyulear(x, 20, [], Fs); % Yule-Walker AR法, 选用20阶AR模型
[Pxx3,f] = pwelch(x, 256, Fs, [], 128);
hold on
plot(f, 10*log10(Pxx1))

```

```

plot(f, 10*log10(Pxx2), ':')
plot(f, 10*log10(Pxx3), '-.')
hold off
legend('Burg', 'YuleAR', 'Welch')

```

### 5. 协方差法和修正协方差法

AR 谱估计的协方差法基于最小化前向预报误差，修正的协方差法基于最小化前向和后向预报误差。这两种方法的估计效果基本相同。

MATLAB 中相应的函数是 `pcov` 和 `pmcov`，它们的用法和函数 `pburg` 相同。

## 9.7 窗函数

在数字滤波器的设计和谱分析中，窗函数的选择对于决定最终结果的质量起着重要的作用。窗函数的主要作用是减弱由于对无穷数列截断而产生的吉布斯现象的影响。

工具箱中提供的窗函数如表 9-7 所示，本节我们将分类介绍这些窗的特点及函数用法。

表 9-7 窗及其 MATLAB 函数

窗	函数	窗	函数
矩形窗	<code>boxcar</code>	凯塞窗	<code>kaiser</code>
海宁窗	<code>hamming</code>	巴特利特窗	<code>bartlett</code>
海明窗	<code>hamming</code>	布兰克曼窗	<code>blackman</code>
切比雪夫窗	<code>chebwin</code>	三角窗	<code>triang</code>

### 9.7.1 矩形窗、巴特利特窗、三角窗

矩形窗是基本窗。它是一个全 1 的列向量，长度由用户指定。例如，要生成一个长 100 的矩形窗用如下命令：

```
w = boxcar(100);
```

该命令将生成长度为 100 的全 1 列向量 `w`。因而它又等价于命令：

```
w = ones(100, 1);
```

巴特利特窗和三角窗是两个矩形窗的卷积。函数 `bartlett` 和 `triang` 生成类似的三角窗，但它们有三个重要的区别：

1) 函数 `bartlett` 生成的窗两端总是 0，因此，如果 `n` 是奇数，则 `bartlett(n+2)` 的中间部分等于 `triang(n)`。

2) 如果 `n` 是偶数，`bartlett` 仍然是两个矩形窗的卷积。`n` 是偶数的情况下，对三角窗没有标准的定义，这时 `triang` 函数生成的三角窗的线段斜率比 `bartlett` 函数要稍陡一些。

3) `n` 是偶数时，巴特利特窗的傅立叶变换是负数，而三角窗的傅立叶变换总是非负的。

### 9.7.2 广义余弦窗

矩形窗、海宁窗、海明窗、布兰克曼窗可以用同一种通用的形式表示，这就是广义余弦窗，这些窗都是广义余弦窗的特例。

这些窗都是频率为  $0$ 、 $2\pi/(N-1)$  和  $4\pi/(N-1)$  的余弦曲线的合成，其中  $N$  为窗的长度。生成这些窗的常用方法是采用如下一组命令：

```
ind = (0:n-1) * 2*pi/(n-1);
w = A - B*cos(ind) + C*cos(2*ind);
```

其中， $A$ 、 $B$ 、 $C$  是由用户定义的常数。

根据  $A$ 、 $B$ 、 $C$  取值的不同，形成不同的窗函数，分别为：

- 矩形窗： $A = 1$ ,  $B = 0$ ,  $C = 0$
- 海宁窗（函数 hanning）： $A = 0.5$ ,  $B = 0.5$ ,  $C = 0$
- 海明窗（函数 hamming）： $A = 0.54$ ,  $B = 0.46$ ,  $C = 0$
- 布兰克曼窗（函数 blackman）： $A = 0.42$ ,  $B = 0.5$ ,  $C = 0.08$

采用海宁窗、海明窗或布兰克曼窗可以有效地降低旁瓣的高度，其副作用是增加了主瓣的宽度。这几种窗在随机信号的谱分析中很常用。

注意，按照定义，当  $A = 0.5$ ,  $B = 0.5$ ,  $C = 0$  时，广义余弦窗在第一个采样点和第  $n$  个采样点的值为零，为了消除这些窗边沿的零点，函数 hanning 用  $2\pi/(N+1)$  替代  $2\pi/(N-1)$ 。

### 9.7.3 凯塞窗

上面讨论的几种窗函数都是以增加主瓣宽度为代价来获得旁瓣抑制的，而凯塞窗可以在主瓣宽度和旁瓣高度之间选择它们的比重。凯塞窗类似于扁长球面（prolate-spheroidal）窗，其主瓣能量和旁瓣能量的比值最大。

凯塞窗的函数 kaiser 的调用格式为：

kaiser(n, beta)

其中  $n$  是窗的长度，参数  $\beta$  用于控制旁瓣的高度。 $n$  一定时， $\beta$  越大，其频谱的旁瓣越小，但主瓣宽度也相应增加； $\beta$  一定， $n$  变化时，旁瓣的高度保持不变。

用凯塞窗设计 FIR 滤波器时，为使旁瓣高度为  $-adB$ ，参数  $\beta$  应按如下关系进行计算：

$$\beta = \begin{cases} 0.1102(\alpha - 8.7) & \alpha > 50 \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 21) & 50 \geq \alpha \geq 21 \\ 0 & \alpha < 21 \end{cases}$$

如果滤波器的过渡带宽度为  $\Delta\omega$  rad/s，则凯塞窗的长度  $n$  应满足如下关系：

$$n = \frac{\alpha - 8}{2.285\Delta\omega} + 1$$

### 9.7.4 切比雪夫窗

对于给定的旁瓣高度，切比雪夫窗的主瓣宽度最小，这是由于它的旁瓣具有相同的高度，即具有等波纹性。

切比雪夫窗的函数 chebwin 有两个输入参数，一个是窗的长度，另一个是旁瓣和主瓣的高度差（单位 dB）。

切比雪夫窗在边沿的采样点有尖峰。

本例中，我们绘制了几种常用窗函数的波形和频谱，以便读者对这些窗的时域和频域

特性有一个形象的概念。结果如图 9-30 所示，图中从上向下依次为矩形窗、海宁窗、凯塞窗、切比雪夫窗的时域（左侧）和频域（右侧）波形。为便于观察，窗的点数取得比较少。

```
w1 = boxcar(21);
w2 = hanning(21);
w3 = kaiser(21, 8);
w4 = chebwin(21, 40);
[h1, f1] = freqz(w1/sum(w1), 1, 512, 2); % 除以w1的元素和目的是使频谱都从0dB开始
[h2, f2] = freqz(w2/sum(w2), 1, 512, 2);
[h3, f3] = freqz(w3/sum(w3), 1, 512, 2);
[h4, f4] = freqz(w4/sum(w4), 1, 512, 2);
subplot(4,2,1), stem(w1)
subplot(4,2,2), plot(f1, 20*log10(abs(h1)))
subplot(4,2,3), stem(w2)
subplot(4,2,4), plot(f2, 20*log10(abs(h2)))
subplot(4,2,5), stem(w3)
subplot(4,2,6), plot(f3, 20*log10(abs(h3)))
subplot(4,2,7), stem(w4)
subplot(4,2,8), plot(f4, 20*log10(abs(h4)))
```

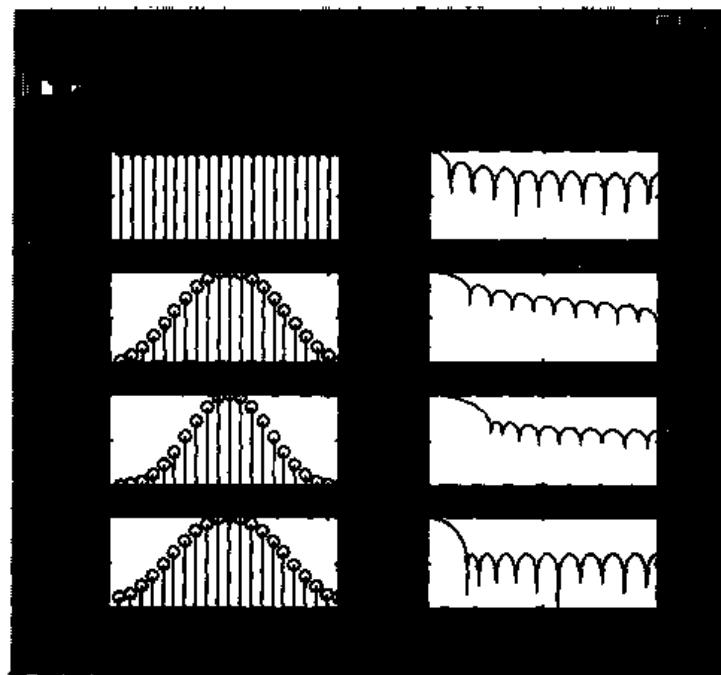


图 9-30 几种窗的时域和频域波形

## 9.8 交互工具

信号处理工具箱中提供了一个交互式的图形用户界面工具——SPTool，用来执行信号处理的任务。SPTool 为信号处理工具箱中的很多重要的函数提供了易于使用的界面，利用

它，使用鼠标就可以载入、观察、分析和打印数字信号，分析、实现和设计数字滤波器，还可以对信号进行谱分析。

### 9.8.1 SPTool 主窗口

在 MATLAB 命令窗口输入命令 `sptool` 来调用 SPTool，其主窗口如图 9-31 所示。

在 SPTool 的主窗口中有三个列表框：“Signals”、“Filters”和“Spectra”，列表框中列出的内容是供处理的信号。其下的按钮对应着 SRTTool 中包含的四个完整的信号处理工具，它们是：

- 1) 信号浏览器 (Signal Browser): 观察、测量、分析一个或多个时域信号的信息。
- 2) 滤波器设计器 (Filter Designer): 设计和修改不同长度和类型的 FIR 和 IIR 滤波器。
- 3) 滤波器观察器 (Filter Viewer): 分析滤波器的特性，包括幅值响应、相位响应、群延迟、脉冲响应和阶跃响应。
- 4) 谱观察器 (Spectrum Viewer): 把用各种 PSD 估计方法得到的频域数据以图形方式进行分析。

后面几节我们将分别介绍这四种工具。

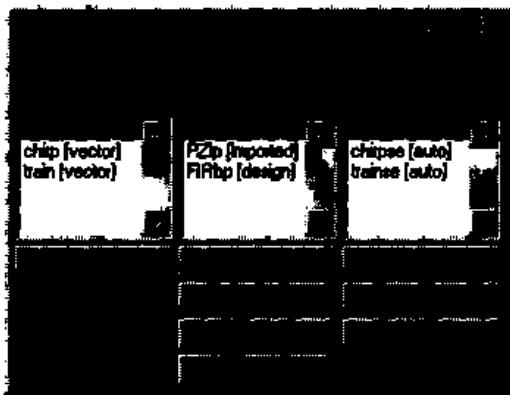


图 9-31 SPTool 主窗口

这里首先介绍 SPTool 窗口 File 菜单和 Edit 菜单的用法。两个菜单的内容如图 9-32 所示。



图 9-32 File 菜单（左）和 Edit 菜单（右）

#### 1. File 菜单

**Open Session:** 打开保存的主窗口设置。

**Import:** 从 MATLAB 工作空间或数据文件载入数据。

**Export:** 把数据输出到 MATLAB 工作空间或文件。

**Save Session 和 Save Session As:** 保存主窗口的当前设置。

**Preferences:** 一些通用参数的设置, 如图 9-33 所示, 请读者参照具体应用体会其中参数的作用。

## 2. Edit 菜单

**Duplicate:** 复制指定的信号。

**Clear:** 删 除 指 定 的 信 号。

**Name:** 给指定的信号重命名。

**Sampling Frequency:** 改变指定信号的采样频率。

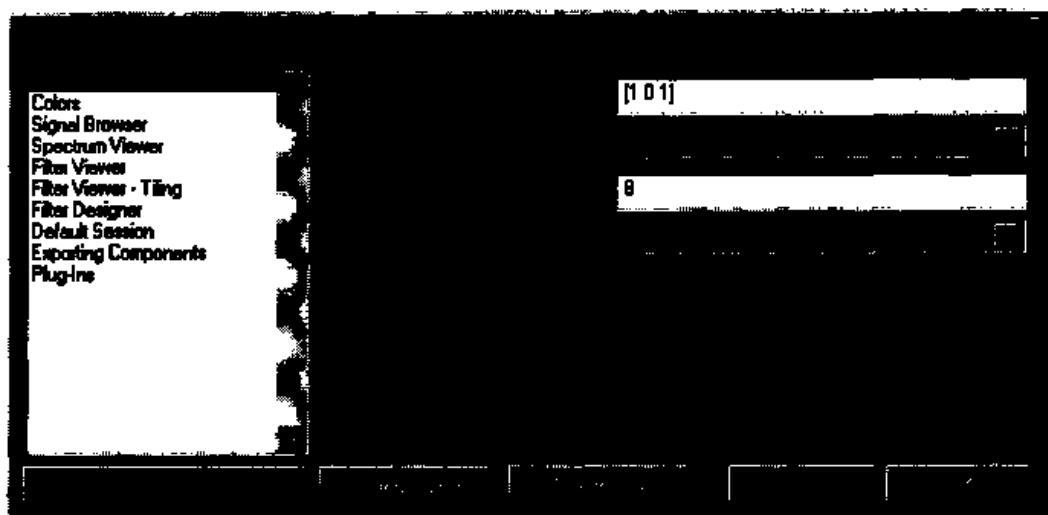


图 9-33 通用参数设置对话框

### 9.8.2 信号浏览器

在 SPTool 主窗口的 Signals 下的列表框中列出已经载入到 SPTool 中的信号, 现在列出的是其自身带有的三个信号, 从中选择一个(例如 mtlb)并按下该列表框下方的 View 按钮就可以调用信号浏览器, 其界面如图 9-34 所示。

界面中间的窗口是观察信号的主窗口, 界面中的其它工具都是针对主窗口中的曲线的。下面的小窗口显示信号中的所有数据。

界面上方的工具栏是对曲线进行放大和缩小的工具: 按下按钮“Mouse Zoom”可以通过拖放鼠标在主窗口中选择想要放大的区域, 按下按钮“Full View”恢复显示整条曲线, 后面的四个按钮分别是沿 Y 方向的放大、缩小和沿 X 方向的放大缩小。

“Help”按钮很有用, 按下它鼠标指针上就会增加一个“?”号, 把这个指针点到某个位置将弹出 MATLAB 帮助窗口, 显示对该位置内容的帮助信息。

界面右上角的 Color 按钮用于设置曲线的颜色和线条。按下该按钮会出现相应的对话框。

界面右侧靠上的四个按钮选择不同类型的标尺: “Vertical”是垂直标尺, 只显示横坐标的值; “Horizontal”是水平标尺, 只显示纵坐标的值; “Track”是点的轨迹, 同时显示横纵坐标的值; “Slope”在显示横纵坐标的同时还显示两个点连线的斜率。

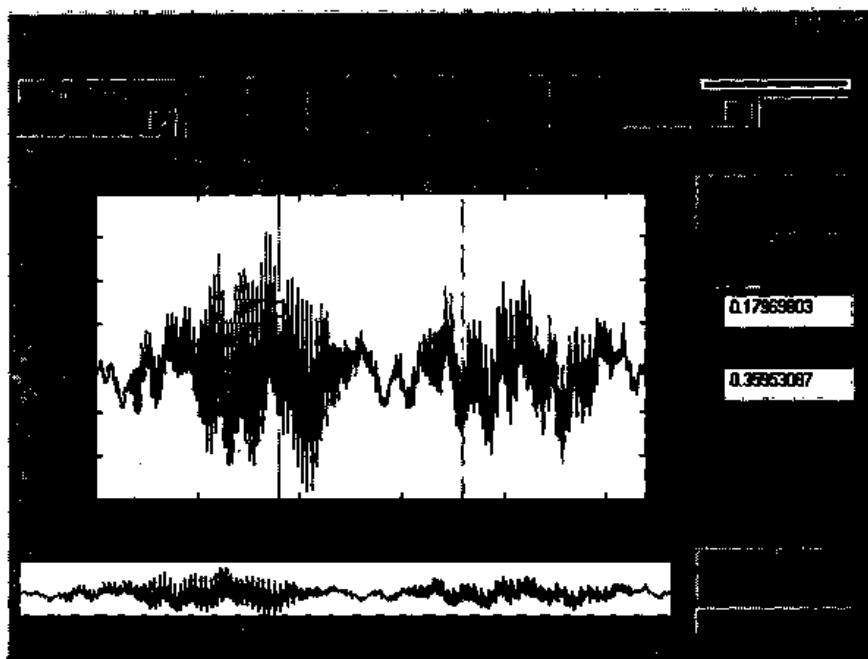


图 9-34 信号浏览器界面

界面右侧中间区域显示坐标值，其中的  $dx$ 、 $dy$  分别为两个标尺（或两个点）之间的横纵坐标的差， $m$ （标尺为 Slope 方式）为两个点连线的斜率，即  $dy/dx$ 。

界面右下方的按钮“Peaks”和“Valleys”分别用于在曲线上标出波峰和波谷。按钮“Save Rulers...”把当前标尺所在的位置坐标保存在 MATLAB 工作空间的变量中，变量名由用户在弹出的对话框中指定。

在界面的菜单 File 下包括页面设置、打印预览、打印三个选项用于信号波形的打印，它们的用法很简单，请读者自己操作。

如果要分析自己的数据，则首先要把该数据载入到 MATLAB 工作空间或保存为 MATLAB 数据文件的格式，然后在 SPTool 主窗口中用 File 菜单下的选项 Import 载入该数据到 SPTool 的工作环境中。

### 9.8.3 滤波器观察器

在 SPTool 主窗口的 Filters 下的列表框中选择一个示例的滤波器（例如 PZlp）并按下该列表框下方的 View 按钮就可以调用滤波器观察器来分析该滤波器的特性，其界面如图 9-35 所示。

该界面和信号浏览器界面的区别主要是界面左侧的内容。在滤波器观察器界面的左侧有两个组合框——“Plots”和“Frequency Axis”。

在组合框 Plots 中设置要绘制滤波器的哪些特性，其中的六个复选框依次为幅值响应、相位响应、群延迟、零极点、脉冲响应、阶跃响应。其中，幅值响应和相位响应还可以设置单位。

组合框 Frequency Axis 中的弹出式菜单 Scale 和 Range 分别用于设置频率轴的刻度和范围。

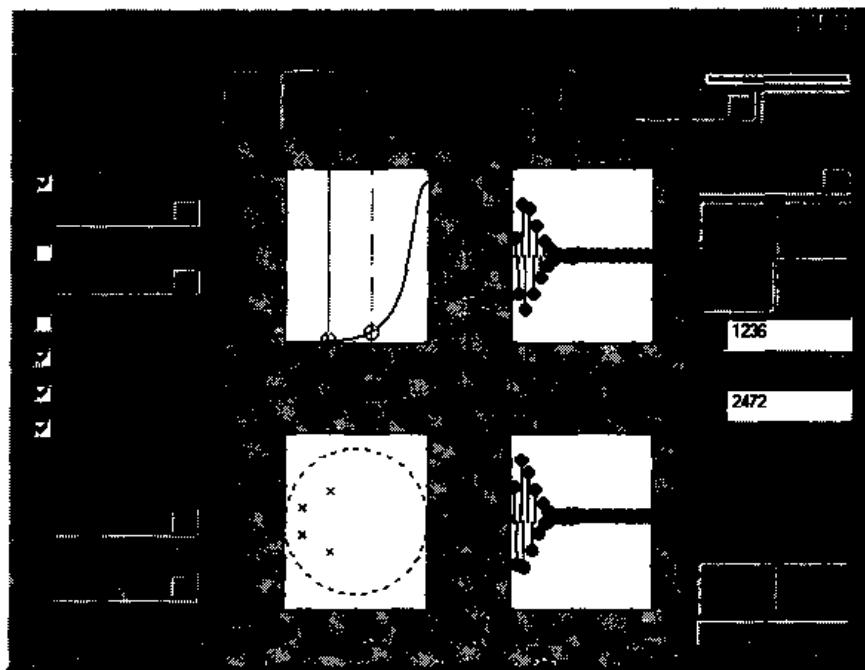


图 9-35 滤波器观察器

#### 9.8.4 滤波器设计器

在 SPTool 主窗口的 Filters 下的列表框中按下按钮 New Design 或选择一个示例的滤波器并按下该列表框下方的 Edit Design 按钮就可以调用滤波器设计器来设计新滤波器或编辑示例的滤波器设计指标。

滤波器设计器的界面如图 9-36 所示，图中的曲线是新设计的 FIRbp 滤波器的频率响应曲线。下面我们通过设计过程来介绍设计器的用法。

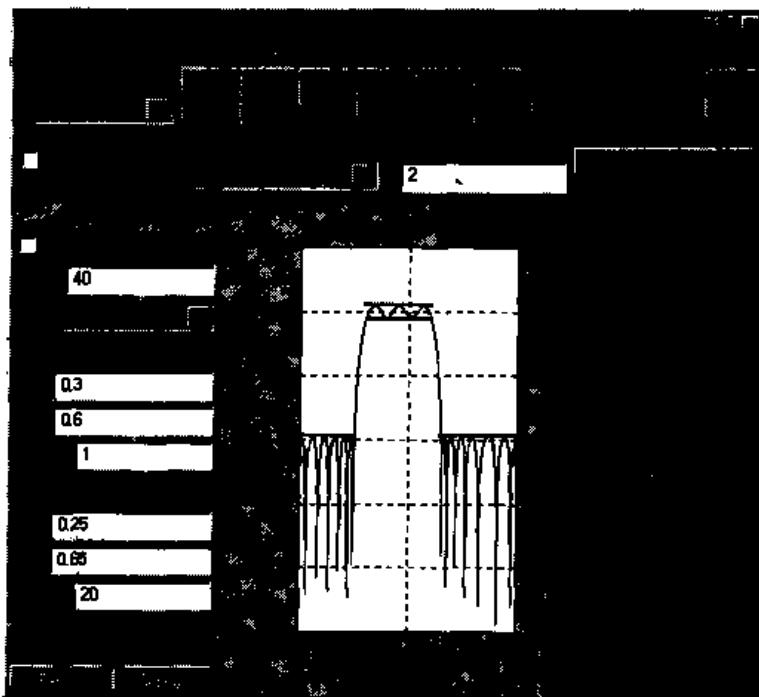


图 9-36 滤波器设计器

首先要在界面上方的 Algorithm 下选择采用的算法或滤波器类型，算法不同，界面左侧的“Specifications”区域中会显示不同类型的设计规格，这里选择设计等波纹 FIR 滤波器。同时在 Algorithm 右侧的编辑框中设置采样频率为标准形式：2（奈氏频率为 1）。

下面给出滤波器的设计规格。在 Specifications 区域中，先去掉“Minimum Order”左侧的复选框——不使用最小阶次——这时在其下方出现编辑框“Order”，在其中输入需要的滤波器阶次：40。在 Type 旁边选择滤波器的类型——包括低通、高通、带通、带阻——我们选择带通。

这时，在下面的组合框“Passband”和“Stopband”中分别列出需要设置的通带和阻带的规格：Fs1、Fp1 和 Fp2、Fs2 分别指定了两个过渡带的起始频率、截止频率；weight 指定了通带和阻带的权重。

这些规格设置好后，按下窗口下面的 Apply 按钮，设计器就会按指定的规格设计滤波器，并在中间窗口中显示设计出的滤波器的频率响应特性。

右侧窗口中显示滤波器的实际特性，现在显示的是我们设计的等波纹滤波器的实际通带波动“Rp 4.191”和阻带衰减“Rs 38.48”，它们的单位都是 dB。

至此，一个完整的滤波器就设计好了。关闭滤波器设计器窗口，回到 SPTool 主窗口，这时在 Filters 列表框下会增加一个滤波器“filt1”。

在 Filters 列表框下方的按钮“Apply”用于将滤波器应用于 Signals 列表框中指定的信号，也就是用指定的滤波器对该信号进行滤波，并把滤波的结果保存为一个新的信号。

例如，在 Signals 列表框中选择 mtlb（其曲线如图 9-34），在 Filters 列表框中选择上面设计的滤波器 filt1，按下 Apply 按钮，这时出现如图 9-37 所示的对话框，在该对话框中选择合适的算法并指定输出信号名，按下 OK 后 SPTool 将用滤波器 filt1 对信号 mtlb 进行滤波，输出信号保存到 Signals 列表框中，名为 sig1。用信号浏览器观察该输出信号的曲线如图 9-38 所示。

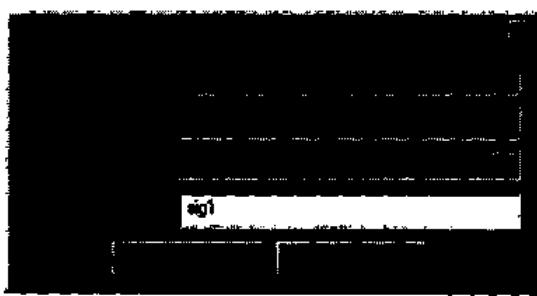


图 9-37 滤波器应用对话框

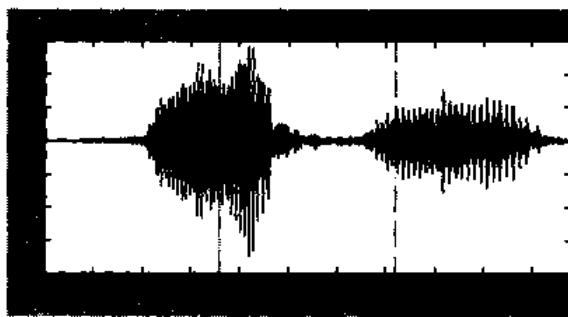


图 9-38 滤波后的曲线

### 9.8.5 谱观察器

在 SPTool 主窗口 Spectra 下的列表框中选择一个示例的谱（例如 mtlbse，它是 Signals 列表框中的信号 mtlb 的谱估计）并按下该列表框下方的按钮 View 就可以调用谱观察器对载入的谱进行观察和分析。谱观察器如图 9-39 所示。

在观察器界面左侧的“Parameters”组合框中，列出了谱估计的方法和相应的参数，这些参数和调用函数进行谱估计需要指定的参数相同，请读者参照 9.6 节讲过的相关内容。

窗口下方的弹出式菜单“Inherit from”用于继承其它谱估计的参数，在该菜单中选择其它某个谱估计，然后按下按钮 Apply，谱观察器采用指定谱估计的参数设置来估计当前的谱。

和信号浏览器类似，在谱观察器中的曲线也可以打印，File 菜单下提供了相应的菜单项。

如果要建立新的谱估计，则首先需要在 SPTool 主窗口的 Signals 列表框中指定要分析的信号，然后在 Spectra 列表框下按下按钮 Create（Update 按钮和 Create 按钮类似，区别是它用新建的谱估计覆盖已有的谱估计）。出现谱观察器界面后，在左侧的 Parameters 组合框中设置谱估计的方法和相应参数，然后按下 Apply 按钮，中间窗口中就会显示出谱估计的结果。

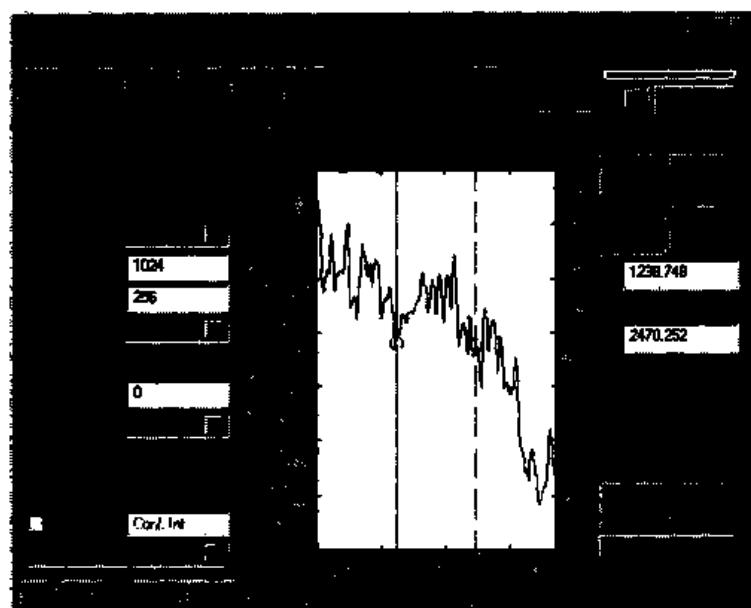


图 9-39 谱观察器

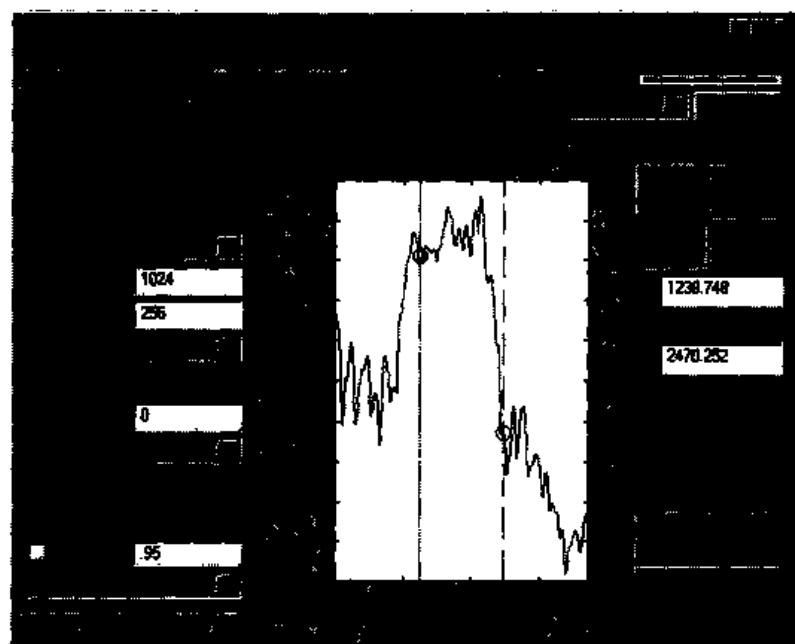


图 9-40 滤波后数据的谱估计

例如，我们在 SPTool 主窗口的 Signals 列表框中指定前面通过对信号 mtlb 滤波生成的信号 sig1，然后按下 Apply 按钮，在谱观察器窗口左侧的参数设置窗口中设置参数和图 9-39 中相同，以便于比较，然后按下按钮 Apply，结果如图 9-40 所示。对比图 9-39 和 9-40 中的频域曲线可以明显地看出带通滤波器的特性。

# 第 10 章 控制系统工具箱

控制系统工具箱提供了丰富的工具用于处理和分析线性时不变（LTI）模型。它支持连续系统和离散系统，单输入单输出（SISO）系统和多输入多输出（MIMO）系统。

利用工具箱中的函数不但可以实现系统模型的建立、转换、分析和处理，还可以进行控制系统设计。新版本的 MATLAB 工具箱中提供了系统分析和设计的交互式工具，可以大大简化分析和设计的过程。

控制系统工具箱的文件位于 MATLAB 的目录\toolbox\control 下。

## 10.1 LTI 模型

### 10.1.1 建立 LTI 模型

LTI 模型包括传递函数模型（TF）、零极点增益模型（ZPK）、状态空间模型（SS）和频率响应数据模型（FRD）。在新版本的 MATLAB 中，这些模型都可以用 MATLAB 函数基于“面向对象”的概念建立起来，在 MATLAB 中建立起来的每个模型都是一个对象，它可以完整的反映该模型的信息。在下面的详细介绍中读者可以体会其中的具体含义。

#### 1. 传递函数模型

##### (1) SISO 系统

SISO 系统的传递函数形式如下：

$$H(s) = \frac{num(s)}{den(s)} = \frac{num(1)s^{nn-1} + num(2)s^{nn-2} + \dots + num(nn)}{den(1)s^{nd-1} + den(2)s^{nd-2} + \dots + den(nd)}$$

其中 nn 和 nd 分别是分子和分母的项数。要在 MATLAB 中建立传递函数的模型，有两种方法：直接用函数 tf；或者通过拉普拉斯变量 s。

函数 tf 的用法为：

`h = tf(num, den)`

其中 num 和 den 分别是传递函数分子和分母多项式的系数向量，按照 s 的降幂排列。

返回值 h 是一个 TF 对象，该对象包含了传递函数的分子和分母信息。

传递函数： $h(s) = (s - 1)/(s^2 - 2s + 7)$ ，首先用函数 tf，命令如下：

`h = tf([1, -1], [1, -2, 7])`

命令执行后会显示如下内容：

`Transfer function:`

$s^2 - 2s + 7$

这就是 TF 对象  $h$  包含的信息，我们看到，它把传递函数的形式形象地表示出来。

如果通过拉普拉斯变量  $s$  来建立这个模型，则需要如下两条命令：

```
s = tf('s'); % 首先定义变量 s 为特殊的传函模型
h = (s-1)/(s^2-2*s+7) % 然后直接输入传递函数的公式形式
```

这两条命令的执行结果和第一种方法相同。

注意：这里定义的特殊传函模型也可以用其它变量名表示，建立的对象  $h$  中包含的信息自动转换为字母  $s$  表示。而且， $s$  只要定义一次，然后可以用它建立任意多个传函模型，直到再一次对它进行定义为止。

## (2) MIMO 系统

MIMO 系统的传递函数模型是以 SISO 系统传函模型为元素的二维数组。有两种方法可以建立 MIMO 系统的传函模型：串联 SISO 传函模型或以单元数组为参数调用 `tf` 函数。我们举例说明这两种方法。

传递函数为  $H(s) = \begin{bmatrix} \frac{s}{s+1} \\ \frac{s-1}{s^2-2s+7} \end{bmatrix}$ ，首先采用串联 SISO 模型的方法，命令如下：

```
h11 = tf([1], [1,1]);
h12 = tf([1,-1], [1,-2,7]);
H = [h11; h12]
```

建立的 MIMO 传函模型如下所示，这也就是 `TF` 对象  $H$  的内容：

Transfer function from input to output...

```
#1: 1
      -----
      s + 1
      s - 1
#2: -----
      s^2 - 2s + 7
```

这种方法能够比较形象地反映 MIMO 系统传递函数的特点。下面我们采用单元数组的方法，即把传递函数两部分的分子和分母的多项式系数向量分别组合为两个单元数组，然后以单元数组为参数调用 `tf` 函数。命令如下：

```
N = {[1]; [1,-1]}; % 两部分传函分子系数向量组成单元数组，注意按列的方向
D = {[1,1]; [1,-2,7]}; % 分母多项式向量组成单元数组
H = tf(N, D)
```

命令执行的结果和前面相同。

## 2. 零极点增益模型

### (1) SISO 系统

SISO 系统的零极点增益形式如下：

$$H(s) = k \frac{[s - z(1)][s - z(2)] \cdots [s - z(m)]}{[s - p(1)][s - p(2)] \cdots [s - p(n)]}$$

其中  $k$  是实型标量，表示系统增益；向量  $z$  和  $p$  中的元素分别是系统的零点和极点，它们可能是实数或共轭复数对。这里的零点和极点实际上分别是传递函数形式中分子多项式和分母多项式的根。

要在 MATLAB 中建立零极点增益的模型，同样有两种方法：直接用函数 `zpk`；或者通过拉普拉斯变量  $s$ 。这两种方法和建立传函模型的方法类似。

函数 `zpk` 的用法为：

```
h = zpk(z, p, k)
```

其中  $z$  和  $p$  分别是系统的零点和极点向量， $k$  是系统增益。返回值  $h$  是一个 ZPK 对象，该对象包含了零点、极点和增益的信息。

零极点增益形式： $H(s) = 2 \frac{s-1}{(s-(1+i))(s-(1-i))}$ ，首先用函数 `zpk`，命令如下：

```
h = zpk([1],[1+i,1-i],2)
```

返回的 ZPK 对象  $h$  内容为：

Zero/pole/gain:

```
2 (s-1)
-----
(s^2 - 2s + 2)
```

如果通过拉普拉斯变量  $s$  来建立这个模型，则需要如下两条命令：

```
s = zpk('s'); % 首先定义变量 s 为特殊的 ZPK 模型
h = 2*(s-1)/[(s-1-i)*(s-1+i)] % 然后直接输入 ZPK 的公式形式
```

这两条命令的执行结果和第一种方法相同。

## (2) MIMO 系统

与 MIMO 系统的 TF 模型建立方法相同，也可以用串联 SISO 系统和单元数组两种方法来建立 MIMO 系统的 ZPK 模型。

通过单元数组建立 MIMO 系统 ZPK 模型的语法和 SISO 系统相同：

```
H = zpk(Z, P, K)
```

只是，这里的  $H$  是二维数组， $Z$ 、 $P$ 、 $K$  都是单元数组。假设系统有  $p$  个输入， $m$  个输出，则  $H$  的大小是  $p \times m$ ，相应的  $Z$ 、 $P$ 、 $K$  分别为：

- $Z$  是  $p \times m$  的单元数组， $Z\{i,j\} = H_{ij}(s)$  的零点。
- $P$  是  $p \times m$  的单元数组， $P\{i,j\} = H_{ij}(s)$  的极点。
- $K$  是  $p \times m$  的矩阵， $K(i,j) = H_{ij}(s)$  的增益。

已知两输入两输出系统:  $H(s) = \begin{bmatrix} \frac{s-1}{s+1} & 0 \\ -1 & \frac{s}{(s-1)(s-2)} \end{bmatrix}$ , 首先通过 SISO 系统串联建立

ZPK 模型, 命令如下:

```
h11 = zpk(1, -1, 1);
h12 = zpk([], [], 0);
h21 = zpk([], 0, -1);
h22 = zpk(0, [1, 2], 1);
H = [h11, h12; h21, h22]
```

返回的 ZPK 模型对象 H 为:

Zero/pole/gain from input 1 to output...

```
#1:  $\frac{(s-1)}{(s+1)}$ 
#2:  $\frac{-1}{s}$ 
```

Zero/pole/gain from input 2 to output...

```
#1: 0
#2:  $\frac{s}{(s-1)(s-2)}$ 
```

采用单元数组法的命令如下:

```
Z = {1, []; [], 0};
P = {-1, []; 0, [1, 2]};
K = [1, 0; -1, 1];
H = zpk(Z, P, K)
```

### 3. 状态空间模型

状态空间模型用线性微分方程组来描述系统的动态特性, 它的形式如下:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

其中,  $x$  是状态向量,  $u$  和  $y$  是输入输出向量。A、B、C、D 都是实数矩阵, 它们的大小由  $x$ 、 $y$ 、 $u$  的长度决定, 设  $x$ 、 $y$ 、 $u$  的长度依次为  $N_x$ 、 $N_y$ 、 $N_u$ , 则 A 的大小为  $N_x \times N_x$ , B 的大小为  $N_x \times N_u$ , C 为  $N_y \times N_x$ , D 为  $N_y \times N_u$ 。

在 MATLAB 中用函数 ss 建立状态空间模型, 格式为:

```
sys = ss(A, B, C, D)
```

返回值 sys 为 SS 对象, 它包含状态空间矩阵 A、B、C、D 的信息。

已知状态空间形式:  $\dot{x} = \begin{bmatrix} 0 & 1 \\ -2 & 0.5 \end{bmatrix}x + \begin{bmatrix} 0 \\ 2 \end{bmatrix}u$ ,  $y = [0 \ 1]x$ , 则在 MATLAB 中建立该模型的命令为:

```
sys = ss([0, 1; -2, 0.5], [0; 2], [0, 1], 0)
```

建立的 SS 模型, 即 SS 对象 sys 的内容如下:

a =

$$\begin{array}{cc} & \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \end{matrix} & \begin{bmatrix} 0 & 1 \\ -2 & 0.5 \end{bmatrix} \end{array}$$

b =

$$\begin{array}{c} u_1 \\ \begin{matrix} x_1 & 0 \\ x_2 & 2 \end{matrix} \end{array}$$

c =

$$\begin{array}{cc} & \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{matrix} y_1 \\ \end{matrix} & \begin{bmatrix} 0 & 1 \end{bmatrix} \end{array}$$

d =

$$\begin{array}{c} u_1 \\ \begin{matrix} y_1 & 0 \end{matrix} \end{array}$$

Continuous-time model.

我们看到, 在 SS 模型中除了矩阵 A、B、C、D 的数据之外, 还包括状态向量和输入输出向量的名称, 这里显示的是默认名。后面我们将会讲到怎样根据自己的需要指定这些名称。

#### 4. 频率响应数据模型

某些情况下, 只有系统的频率响应采样数据, 而不知道系统的传递函数或状态空间模型, 针对这种情况, MATLAB 允许直接用频率响应数据来表示系统模型。

在 MATLAB 中, 用函数 frd 建立频率响应数据模型, 该函数用法为:

```
sys = frd(response, frequencies, units)
```

其中, frequencies 是长为 Nf 的实数向量, 为频率点, Nf 是频率点的个数; response 是复数向量, 是相对于 frequencies 中的各个频率的频率响应值, 它的长度和 frequencies 相同; units 是可选的字符串参数, 指定频率的单位, 取值为'rad/s' (缺省值) 或'Hz'。

对于 MIMO 系统, 函数 frd 的用法相同, 只是参数 response 成为大小是 p×m×Nf 的多维数组, 其中 p 是输出的个数, m 是输入的个数。

#### 5. 离散时间系统模型

前面讲到的四种模型都是连续时间系统模型, 而离散时间系统模型的建立方法和连续时间系统很相似, 使用的函数也相同。区别只是需要在函数中增加一个输入参数指定采样时间。采样时间应该是一个标量, 单位是秒。也可以把该参数设置为-1, 而不指定采样时

间的具体数值。

建立几种离散时间系统模型的函数调用格式如下：

`sys1 = tf(num, den, Ts)`

`sys2 = zpk(z, p, k, Ts)`

`sys3 = ss(a, b, c, d, Ts)`

`sys4 = frd(response, frequency, Ts)`

在建立 TF 模型和 ZPK 模型时，还有如下两个问题需要注意。

(1) 在建立连续系统的 TF 模型和 ZPK 模型时，可以利用拉普拉斯变量 s 建立一个特殊的连续系统模型。在离散系统中，变量 s 就要改为变量 z，而且建立特殊的离散系统模型时需要指定采样时间。

例如，下面的一组命令建立离散 TF 模型，采样时间为 0.1 秒：

`z = tf('z', 0.1);`

`h = (z-1)/(z^2-2*z+7)`

结果为：

Transfer function:

$$\frac{z - 1}{z^2 - 2z + 7}$$

Sampling time: 0.1

(2) 在数字信号处理领域，习惯于把离散系统的传递函数分子分母的系数以  $z^{-1}$  的升幂形式排列（参见第九章），这个习惯和函数 `tf` 存在冲突，对于连续系统和离散系统，函数 `tf` 都习惯于按降幂排列。

为了解决这个冲突，控制系统工具箱提供了函数 `filt` 来建立离散系统的 TF 模型，它和数字信号处理中的习惯相同——按照  $z^{-1}$  的升幂形式排列。

同样的系数向量，函数 `filt` 和 `tf` 的结果不同。请把如下命令的结果和上面的 TF 模型比较：

`h = filt([1, -1], [1, -2, 7], 0.1)`

输出结果为：

Transfer function:

$$\frac{1 - z^{-1}}{1 - 2z^{-1} + 7z^{-2}}$$

Sampling time: 0.1

## 6. 模型数据的还原

上面我们介绍了一些函数用于建立连续和离散 LTI 系统的各种模型，MATLAB 中还提供了相应的函数可以把建立模型时的数据（输入参数）还原出来。这些函数及用法如下：

`[num, den, Ts] = tfdata(sys)`

`[z, p, k, Ts] = zpkdata(sys)`

`[a, b, c, d, Ts] = ssdata(sys)`

```
[response, frequency, Ts] = frdata(sysfr)
```

其中, sys 可以是除 FRD 模型外的各种 LTI 模型对象, sysfr 只能是 FRD 模型的对象。这也意味着, 除 FRD 模型外, 其余几种模型都可以还原为其它模型的数据类型。

对于连续系统, 返回参数 Ts 将返回 0。这时也可以不带 Ts 参数。

如下命令建立 TF 模型然后还原:

```
h = tf([1, -1], [1, -2, 7]);
```

```
[num, den] = tfdata(h)
```

结果为:

```
num =
```

```
[1x3 double]
```

```
den =
```

```
[1x3 double]
```

如果希望直接显示还原的变量中的数据, 可以用参数 “v”, 命令如下:

```
[num, den] = tfdata(h, 'v')
```

这时将显示如下内容:

```
num =
```

```
0     1     -1
```

```
den =
```

```
1     -2      7
```

### 10.1.2 LTI 模型的属性

第 7 章我们曾介绍过图形对象的概念(句柄图形), 与之类似, MATLAB 中的各种 LTI 模型也是对象, 它们也有自己的属性, 并且也可以设置或修改这些属性的值。

LTI 模型的通用属性如表 10-1 所示。每种模型都有自己的专用属性, 如表 10-2 所示。

表 10-1 LTI 模型通用属性

属性	含义	属性值格式
IoDelayMatrix	输入输出延迟	矩阵
InputDelay	输入延迟	向量
InputGroup	输入通道组	单元数组
InputName	输入通道名	字符串单元向量
Notes	模型记录中的注释	文本
OutputDelay	输出延迟	向量
OutputGroup	输出通道组	单元数组
OutputName	输出通道名	字符串单元向量
Ts	采样时间	标量
Userdata	附加数据	任意

表 10-2 LTI 模型专用属性

模型类型	属性	含义	属性值格式
传递函数 (TF) 模型	num	分子系数	实型行向量单元数组
	den	分母系数	实型行向量单元数组
	Variable	TF模型中的变量	字符串: 's', 'p', 'z', 'q' 或 'z^-1'
零极点增益 (ZPK) 模型	z	零点	列向量单元数组
	p	极点	列向量单元数组
	k	增益	实型矩阵
	Variable	TF模型中的变量	字符串: 's', 'p', 'z', 'q' 或 'z^-1'
状态空间 (SS) 模型	a	矩阵A	实型矩阵
	b	矩阵B	实型矩阵
	c	矩阵C	实型矩阵
	d	矩阵D	实型矩阵
	StateName	状态名	字符串单元向量
频率响应数据 (FRD) 模型	Frequency	频率数据点	实数向量
	ResponseData	频率响应值	复数多维数组
	Units	频率单位	字符串: 'rad/s' 或 'Hz'

注意: 这些属性名不区分大小写, 而且在不会混淆的情况下, 可以缩写。

这些属性可以在建立模型时在函数参数中设置, 也可以在模型建立后, 用函数 set 来设置或直接设置。同样, 属性值既可以直接获取也可以通过函数 get 来获取。

下面的两个例子分别介绍属性的设置和获取, 同时说明一些属性的含义。

1) 如下三条命令用三种方法设置了 TF 模型的输入延迟、输入输出名、注释和表示传函模型的变量:

```
sys = tf(1, [1 1], 'Inputdelay', 0.3);
set(sys, 'inputname', 'in1', 'outputname', 'out1', 'notes', 'A example model')
sys.Variable = 'p' % 设置传函模型的变量符号设为p, 通常是s
```

这时模型对象 sys 的内容为:

Transfer function from input "in1" to output "out1":

$$\frac{1}{\exp(-0.3*p) * \frac{1}{p+1}}$$

2) 如下几条命令首先建立一个三输入三输出的连续系统随机状态空间模型 (函数 rss 的用法请参考帮助), 然后修改该模型的输入输出通道名和状态名, 最后把输入输出通道分组: 输入通道 1 和 2 分为一组, 名为 “Signal”, 通道 3 为一组, 名为 “Noise”; 输出通道 2 和 3 分为一组, 名为 “Measure”。命令如下:

```
h = rss(2,3,3);
set(h, 'InputN', {'in1','in2','in3'}, 'OutputN', {'out1','out2','out3'}, ...
      'StateN', {'STA1', 'STA2'})
set(h, 'InputGroup', {[1,2] 'Signal'; [3] 'Noise'}, 'OutputG', {[2,3] 'Measure'})
```

设置后的 SS 对象 h 内容如下：

a =

$$\begin{matrix} & \text{STA1} & \text{STA2} \\ \text{STA1} & -1.0319 & 0.059919 \\ \text{STA2} & 0.059919 & -0.93794 \end{matrix}$$

b =

$$\begin{matrix} & \text{in1} & \text{in2} & \text{in3} \\ \text{STA1} & 1.6236 & 0.858 & -1.5937 \\ \text{STA2} & -0.69178 & 1.254 & -1.441 \end{matrix}$$

c =

$$\begin{matrix} & \text{STA1} & \text{STA2} \\ \text{out1} & 0.57115 & 0 \\ \text{out2} & -0.39989 & 0.71191 \\ \text{out3} & 0.69 & 1.2902 \end{matrix}$$

d =

$$\begin{matrix} & \text{in1} & \text{in2} & \text{in3} \\ \text{out1} & 0 & 0 & 0 \\ \text{out2} & 0 & 0 & -1.0565 \\ \text{out3} & -1.2025 & -1.6041 & 0 \end{matrix}$$

I/O groups:

Group name	I/O	Channel(s)
Signal	I	1,2
Noise	I	3
Measure	O	2,3

Continuous-time model.

通过如下一些命令及其执行结果可以反映获取属性值的方法，这里用到的两个模型就是上例中建立的，请加以比较：

```
get(sys, 'variable')          % 用 get 函数
ans =
p
h.a
ans =
-1.0319    0.0599
 0.0599   -0.9379
h.statename                  % 直接获取
ans =
'STA1'
```

```
'STA2'
get(sys)          % 列出模型对象sys的所有属性
    num: {[0 1]}
    den: {[1 1]}
    Variable: 'p'
    Ts: 0
    InputDelay: 0.3
    OutputDelay: 0
    ioDelayMatrix: 0
        InputName: {'in1'}
        OutputName: {'out1'}
        InputGroup: {0x2 cell}
        OutputGroup: {0x2 cell}
        Notes: {'A example model'}
    UserData: []
```

### 10.1.3 模型转换

在 10.1.1 节我们曾介绍数据的还原，用于数据还原的几个函数能够把一种模型对象转换为其它类型的模型数据，这实际上就是一种模型的转换，它的方式是从模型对象转换为数据。

我们在信号处理中还曾经介绍线性系统模型的转换（参见第 9 章），从控制系统的角度看，这种转换实际上是构造模型的数据与数据之间的转换。当然，其中的有些函数在控制系统中也可以同样使用。

在控制系统工具箱中提供模型转换是在各种 LTI 模型对象之间相互转换，这种转换的函数就是建立模型对象的几个函数，只不过用法不同。几个函数的调用格式及功能如下：

```
sys = tf(sys)          % 转换为TF模型对象
sys = zpk(sys)         % 转换为ZPK模型对象
sys = ss(sys)          % 转换为SS模型对象
sys = frd(sys, frequency)  % 转换为 FRD 模型对象
```

其中 sys 为模型对象。注意，FRD 模型不能转换为其它类型的模型，而且当把其它模型转换为 FRD 模型时，需要给出频率向量。

首先建立一个 TF 模型对象 sys，然后比较三种转换方式：

```
sys = tf([1, -1], [1, -2, 7]);
```

(1) 信号处理中的方式——数据转换到数据

```
[z, p, k] = tf2zp([1, -1], [1, -2, 7])
```

```
z =
```

```
1
```

```
p =
```

```
1.0000 + 2.4495i
```

```
1.0000 - 2.4495i
```

```
k =
```

```
1
```

(2) 数据还原的方式——对象转换到数据

```
[z,p,k] = zpkdata(sys,'v')
```

```
z =
```

```
1
```

```
p =
```

```
1.0000 + 2.4495i
```

```
1.0000 - 2.4495i
```

```
k =
```

```
1
```

(3) 控制系统工具箱中的方式——对象转换到对象

```
sys1 = zpk(sys)
```

Zero/pole/gain:

```
(s-1)
```

```
-----  
(s^2 - 2s + 7)
```

在进行 LTI 模型转换时需要注意以下几个问题：

1) TF、ZPK、SS 三种模型并不同等适合数值计算。尤其是用高阶传递函数进行计算的精确度经常很差，因而常用状态空间模型来表示。

2) 转换为 TF 模型可能会导致精确度降低，造成 TF 模型的极点和原始的 ZPK 模型或 SS 模型不同。

3) 转换为状态空间时并不能保证最小实现，将一个 SS 模型转换为其它模型再转换回来，结果状态空间矩阵可能会不一样，设置对于 MIMO 系统，状态的个数都可能不同。因而应尽量避免在状态空间和其它模型之间来回转换。

## 10.2 模型的运算

### 10.2.1 算术运算

能够对 LTI 模型进行的算术运算如表 10-3 所示。这些运算是针对 LTI 模型的对象的。

表 10-3 LTI 模型的算术运算

符 号	含 义	符 号	含 义
+	加	inv	矩阵求逆
-	减	'	完全转置
*	乘	.'	转置
/	矩阵右除	^	幂
\	矩阵左除		

本节分别介绍这些运算对于 LTI 模型的具体含义。

### 1. 加和减

LTI 模型的加等价于把两个模型并行连接。如下命令等价于图 10-1 所示的连接。

`sys = sys1 + sys2`

如果 `sys1` 和 `sys2` 是两个状态空间模型，其数据分别为  $A1$ 、 $B1$ 、 $C1$ 、 $D1$  和  $A2$ 、 $B2$ 、 $C2$ 、 $D2$ ，则相加后的模型 `sys` 的数据是：

$$\begin{bmatrix} A1 & 0 \\ 0 & A2 \end{bmatrix} \quad \begin{bmatrix} B1 \\ B2 \end{bmatrix} \quad [C1 \ C2] \quad D1 + D2$$

如果 `sys1` 是 MIMO 模型，而 `sys2` 是 SISO 模型，那么它们的和 `sys` 将和 `sys1` 的维数相同，而且满足关系  $sys_{ij} = sys1(i, j) + sys2$ 。

同样，两个模型相减 `sys = sys1 - sys2` 的含义如图 10-2 所示。

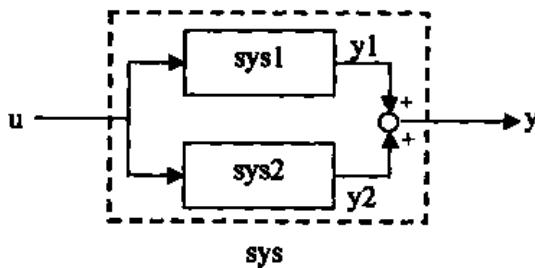


图 10-1 LTI 模型相加

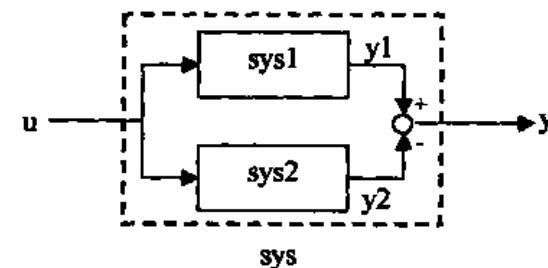


图 10-2 LTI 模型相减

如果是两个不同类型的模型对象相加减，那么 MATLAB 将按照如下的优先顺序，把优先级低的模型自动转换为优先级高的模型，再做运算，结果是优先级高的模型形式：

`FRD > SS > ZPK > TF`

这个优先顺序在对 LTI 模型进行算术运算或其它运算时同样适用。

### 2. 乘

两个模型相乘等价于它们的串行连接，如下运算等价于图 10-3 所示的连接方式。

`sys = sys1 * sys2`

注意模型 `sys1` 和 `sys2` 的顺序是颠倒的。

如果 `sys1` 和 `sys2` 是两个状态空间模型，其数据分别为  $A1$ 、 $B1$ 、 $C1$ 、 $D1$  和  $A2$ 、 $B2$ 、 $C2$ 、 $D2$ ，则它们的乘积 `sys` 的数据是：

$$\begin{bmatrix} A1 & B1C2 \\ 0 & A2 \end{bmatrix} \quad \begin{bmatrix} B1D2 \\ B2 \end{bmatrix} \quad [C1 \ D1C2] \quad D1D2$$

如果 `sys1` 是 MIMO 模型而 `sys2` 是 SISO 模型，那么它们的乘积 `sys` 将是和 `sys1` 维数相同的模型，且满足关系： $sys_{ij} = sys1(i, j) * sys2$ 。

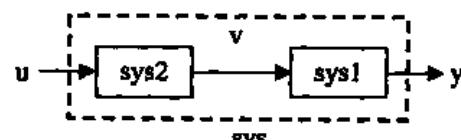


图 10-3 LTI 模型相乘

### 3. 求逆和除法

LTI 模型求逆就是颠倒模型的输入和输出关系，即：

$$y = Hu \rightarrow u = H^{-1}y$$

求逆运算只针对正方系统（输入和输出个数相等），对于模型 `sys`，相应的函数为：

**inv(sys)**

求逆的结果和 sys 类型相同。LTI 模型的除法和求逆运算有如下关系：

- 1) 左除:  $\text{sys1}\backslash\text{sys2} = \text{inv}(\text{sys1}) * \text{sys2}$
- 2) 右除:  $\text{sys1}/\text{sys2} = \text{sys1} * \text{inv}(\text{sys2})$

对于状态空间模型 sys, 其数据为 A、B、C、D, 当这些矩阵都是方阵, 且可逆时, 其逆运算的结果为:

$$A - BD^{-1}C, \quad BD^{-1}, \quad -D^{-1}C, \quad D^{-1}$$

**4. 转置**

对于 LTI 模型 sys, 转置运算的格式为:

`sys.'`

该运算对于各种模型的含义分别为:

- 1) TF 模型: 输入参数是 num 和 den, 则分别把单元数组 num 和 den 转置。
- 2) ZPK 模型: 输入参数是 z、p 和 k, 则分别把单元数组 z、p 和矩阵 k 转置。
- 3) SS 模型: 模型数据是 A、B、C、D, 则分别把这四个矩阵转置。
- 4) FRD 模型: 复频率响应矩阵为 Response, 则把每个频率上的频率响应数据矩阵转置。

**5. 完全转置 (Pertransposition)**

完全转置对于 TF 模型和 ZPK 模型来说, 就是把模型的变量取相反数, 然后再进行转置运算(上面讲到的)。对于传递函数为  $H(s)$  的连续系统, 完全转置后模型的传递函数为:

$$G(s) = [H(-s)]^T$$

对于传递函数为  $H(z)$  的离散系统, 完全转置后传递函数为:

$$G(z) = [H(z^{-1})]^T$$

对于 SS 模型, 完全转置就是把模型数据矩阵中的元素全部取相反数, 然后再做矩阵的转置运算。

可以利用完全转置来获得给定系统频率响应的共轭转置 (Hermitian 转置)。给定系统的传递函数  $H(s)$  和其完全转置  $G(s)$  之间有如下关系:  $G(s)$  的频率响应等于  $H(s)$  频率响应的共轭转置, 即:

$$G(jw) = H(jw)^H$$

要得到系统 sys 在向量 w 指定的频率范围上的频率响应的共轭转置, 可以用函数 freqresp 实现, 命令如下:

`freqresp(sys', w)`

**10.2.2 模型的连接**

控制系统工具箱提供了很多函数帮助建立系统模型, 其中用于模型连接函数如表 10-4 所示。其中包括 I/O 连接, 通用的并行和串行连接, 反馈连接等。这些函数对建立开环和闭环系统都很有用。我们在这一节将介绍几种常用的模型连接方式, 没有详细介绍的函数请读者参看相关帮助。

表 10-4 模型连接函数

函 数	功 能	函 数	功 能
[ , ]	水平级联	feedback	以反馈的形式连接模型
[ ; ]	垂直级联	lft	产生LFT连接
append	按照对角的位置添加模型	parallel	两个模型通用的并行连接
augstate	通过添加状态来增加输出	series	两个模型通用的串行连接
connect	把分块对角的SS模型按指定方式连接		

### 1. 级联

模型的级联就象在 MATLAB 中连接矩阵。有如下三种简单的方式：

- 1) 水平级联:  $\text{sys} = [\text{sys1}, \text{sys2}]$
- 2) 垂直级联:  $\text{sys} = [\text{sys1}; \text{sys2}]$
- 3) 对角级联:  $\text{sys} = \text{append}(\text{sys1}, \text{sys2})$

设模型  $\text{sys1}$  的传函为  $H_1$ ,  $\text{sys2}$  的传函为  $H_2$ ,  $\text{sys}$  的传函为  $H$ , 它们之间满足如下关系:

$$1) \text{ 水平级联: } H = [H_1 \quad H_2]$$

$$2) \text{ 垂直级联: } H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$$

$$3) \text{ 对角级联: } H = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$$

我们通过两个传函模型  $h1$  和  $h2$  的连接来比较各种级联的含义，并和前面讲过的模型加法相对比。

```

h1 = tf([1, -1], [1, -2, 7]);
h2 = tf([1], [1, -1]);
sys = h1+h2          % 模型相加

```

Transfer function:

$$2 s^2 - 4 s + 8$$

$$-----$$

$$s^3 - 3 s^2 + 9 s - 7$$

```
sys = append(h1, h2)          % 模型对角级联
```

Transfer function from input 1 to output...

$$s - 1$$

#1: -----

$$s^2 - 2 s + 7$$

#2: 0

Transfer function from input 2 to output...

#1: 0

```

1
#2: -----
s - 1
sys = [h1; h2]          % 模型垂直级联
Transfer function from input to output...
s - 1
#1: -----
s^2 - 2 s + 7
1
#2: -----
s - 1
sys = [h1,h2]          % 模型水平级联
Transfer function from input 1 to output:
s - 1
-----
s^2 - 2 s + 7
Transfer function from input 2 to output:
1
-----
s - 1

```

## 2. 串联和并联

模型的串联有两种方式：

- 1) sys = series(sys1,sys2): 这种方式相当于两个模型 sys1 和 sys2 相乘，参见上一节。
- 2) sys = series(sys1,sys2,outputs1,inputs2): 这种方式建立更一般的串联，其原理如图 10-4 所示。向量 outputs1 和 inputs2 指定模型 sys1 的哪些输出  $y_1$  和模型 sys2 的哪些输入  $u_2$  要连接到一起。连接成的模型 sys 以  $u$  为输入  $y$  为输出。

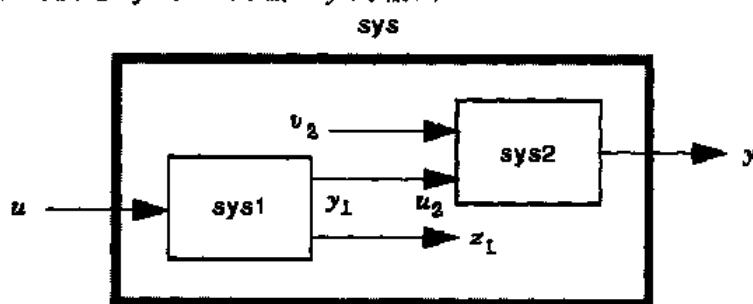


图 10-4 模型串行连接

模型的并联也有类似的方式：

- 1) sys = parallel(sys1,sys2): 相当于两个模型 sys1 和 sys2 相加，参见上一节。
- 2) sys = parallel(sys1,sys2,inpl1,inpl2,outl1,out2): 这种方式建立更一般的并联，其原理如图 10-5 所示。向量 inpl1 和 inpl2 指定模型 sys1 的哪些输入和模型 sys2 的哪些输入连接到一

起, 向量  $\text{out1}$  和  $\text{out2}$  指定模型  $\text{sys1}$  的哪些输出和模型  $\text{sys2}$  的哪些输出相加。连接成的模型  $\text{sys}$  输入为  $[v_1; u; v_2]$ , 输出为  $[z_1; y; z_2]$ 。

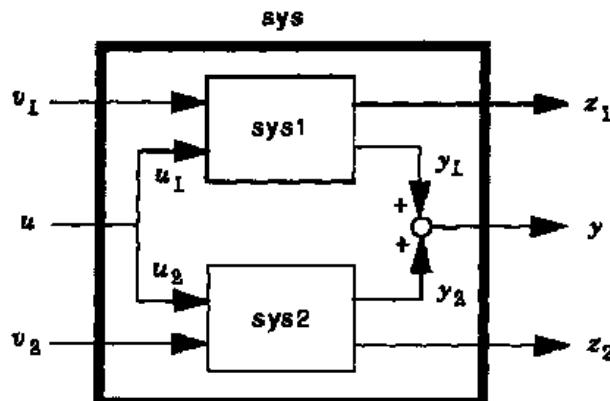


图 10-5 模型并行连接

### 3. 反馈连接

模型的反馈连接有如下两种方式:

1)  $\text{sys} = \text{feedback}(\text{sys1}, \text{sys2}, \text{sign})$ : 这是典型的反馈连接形式, 参数  $\text{sign}$  指定正负反馈,  $-1$  表示负反馈,  $+1$  表示正反馈, 缺省情况下是负反馈。模型  $\text{sys1}$  和  $\text{sys2}$  的输入输出个数要匹配, 如果  $\text{sys1}$  有  $m$  个输入  $p$  个输出, 则  $\text{sys2}$  应该是  $p$  个输入  $m$  个输出。连接后的模型具有和模型  $\text{sys1}$  相同的输入输出。反馈连接的原理如图 10-6 所示。

2)  $\text{sys} = \text{feedback}(\text{sys1}, \text{sys2}, \text{feedin}, \text{feedout}, \text{sign})$ : 这是更为一般的反馈连接, 其原理如图 10-7 所示。向量  $\text{feedin}$  指定模型  $\text{sys1}$  的哪些输入包含在反馈循环中, 向量  $\text{feedout}$  指定模型  $\text{sys1}$  的哪些输出被用来做反馈。连接后的模型具有和模型  $\text{sys1}$  相同的输入输出。

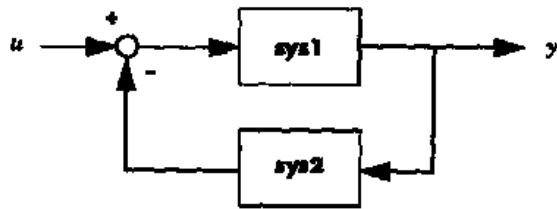


图 10-6 负反馈连接

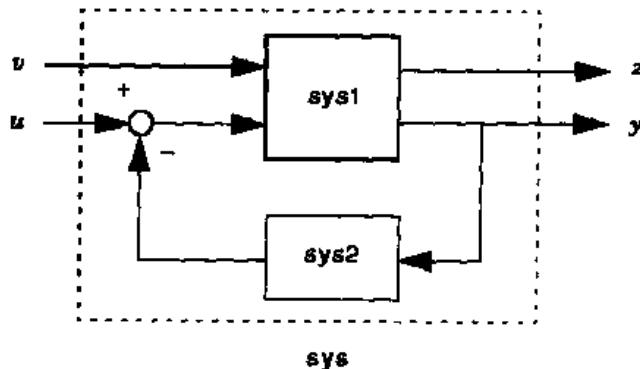


图 10-7 更一般的反馈连接

### 10.2.3 连续模型和离散模型的相互转换

#### 1. 模型转换

函数  $\text{c2d}$  用于将连续时间模型转换为离散时间模型,  $\text{d2c}$  用于将离散时间模型转换为连续时间模型。它们的调用格式如下:

$\text{sysd} = \text{c2d}(\text{sys}, \text{Ts}, \text{method})$

$\text{sysc} = \text{d2c}(\text{sysd}, \text{method})$

其中, 参数  $\text{method}$  为模型转换时采用的离散化方法和插值方法, 在  $\text{c2d}$  函数中其取值

有如下几种：

- 'zoh': 零阶保持 (Zero-order hold), 假设系统输入在采样周期  $T_s$  上为分段常数。
- 'foh': 一阶保持 (first-order hold), 假设系统输入在采样周期  $T_s$  上为分段线性。
- 'tustin': Tustin 近似 (双线性近似)。
- 'prewarp': 带频率预弯曲 (prewarping) 的 Tustin 近似。
- 'matched': 和零极点匹配。

d2c 函数中 method 参数没有'foh' (一阶保持) 算法。缺省情况下两个函数都采用零阶保持算法。

## 2. 离散模型重采样

利用函数 d2d 可以对已有的离散时间模型 TF、SS 或 ZPK 重采样，格式如下：

```
sys2 = d2d(sys1, Ts)
```

新的采样周期  $T_s$  可以任意指定。

我们对采样周期为 1 秒的离散传递函数  $H(z) = \frac{z+0.4}{z-0.6}$  进行重采样，程序如下：

```
h1 = tf([1 0.4],[1 -0.6],1); % 原始传递函数模型
h2 = d2d(h1,0.1) % 重采样，新的采样周期 0.1 秒
```

下面是重采样后的离散传递函数模型：

Transfer function:

$z - 0.8257$

-----

$z - 0.9502$

Sampling time: 0.1

用函数 step (参见 10.3.4 节) 绘制重采样前后两个离散模型的阶跃响应进行比较，命令如下，结果如图 10-8 所示：

```
step(h1, ':', h2, '-') % 虚线表示重采样前的模型，实线表示重采样后的模型
```

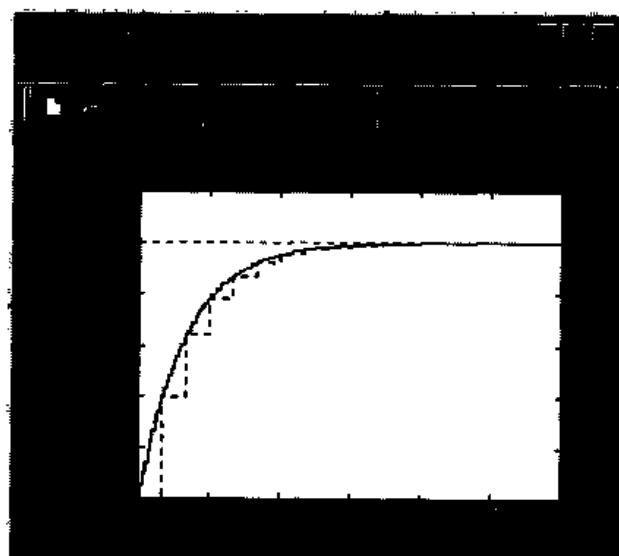


图 10-8 两离散模型阶跃响应

## 10.3 模型分析

### 10.3.1 模型通用特征

通用模型特征包括模型类型、I/O 维数和连续或离散状态等，获取模型特征的相关函数如表 10-5 所示。这些函数可以对任何类型的 LTI 模型进行操作。

表 10-5 模型通用特征函数

函数	功能	函数	功能
Class	显示模型类型 ('tf'、'zpk'、'ss'或'frd')	isempty	检测是否空的LTI模型
Hasdelay	检测LTI模型是否具有延迟	isproper	检测LTI模型是否适当
isa	检测LTI模型（对象）是否为指定的类型	issiso	检测是否SISO模型
isct	检测是否连续时间模型	Ndims	显示模型的维数
isdt	检测是否离散时间模型	size	返回输出/输入/数组的维数，也可以返回TF、ZPK、SS模型的阶数或FRD模型的频率点个数。 该函数是MATLAB内置函数size的重载。

这些函数的用法比较直观，我们通过下面的例子加以说明，更详细的内容请读者参看帮助。

```

sys = rss(2,3,4);          % 建立随机 SS 模型，有 2 个状态、3 个输出、4 个输入
class(sys)                  % 获取 sys 的类型
ans =
ss
isa(sys, 'ss')             % 判断 sys 是否 SS 类型，如果是，结果为 1，否为 0
ans =
1
size(sys)                  % 不带输出参数执行 size 函数，返回输入输出和 SS 模型的状态个数
State-space model with 3 outputs, 4 inputs, and 2 states.
d = size(sys)              % 带输出参数，返回向量中元素的顺序是[输出 输入]
d =
3      4
isct(sys)                  % 判断 sys 是否连续系统
ans =
1
ndims(sys)                 % 判断模型的维数，一般 LTI 模型都是二维：输出和输入
ans =
2
% 下面两条命令判断模型是否适当，SS 模型总是适当的，返回值为 1
% 对于 TF、ZPK 模型，分子向量比分母向量长时，认为不适当，否则是适当的
isproper(tf([1 2 3],[1 2]))
ans =
0

```

```
isproper(tf([1 2],[1 2 3]))
```

```
ans =
```

```
1
```

### 10.3.2 模型动态特性

控制系统动态特性指系统的零点、极点、直流增益和范数等特性。控制系统工具箱中相应的函数如表 10-6 所示。这些函数中除了  $L_\infty$  范数 (norm) 能够用于 FRD 模型之外，其它函数都只能用于 TF、ZPK、SS 模型。

表 10-6 模型动态特性函数

函 数	功 能
covar	模型对白噪声响应的协方差，对于SS模型，同时返回其状态的协方差
damp	模型极点的固有频率和阻尼
dcgain	直流增益，对连续系统传函模型即 $s=0$ 的值，对 SS 模型为 $D=CA^{-1}B$ ；离散系统即传函 $z=1$ 的值，对 SS 模型为 $D=C(I-A)^{-1}B$
dsort	按照幅值排列离散模型的极点
esort	按照实部排列连续模型的极点
norm	模型的范数，包括 $H_2$ （传递函数脉冲响应的均方根）和 $L_\infty$ （频率响应的峰值增益）
pole, eig	计算模型的极点，对于 SS 模型就是状态矩阵 A 的特征值
pzmap	绘制极点-零点图（不带输出参数）或返回极点和零点
zero	计算模型的零点（SISO 系统）或传输零点（MIMO 系统）

我们通过下面的例子简要说明这些函数的用法，更详细的内容请读者参看帮助。

```
H = tf([2 1],[1 2 3]); % 建立一个 TF 模型
covar(H, 3) % 计算模型对白噪声响应的协方差，3 为噪声强度
ans =
3.2500
damp(H) % 不带输出参数，计算模型的特征值、阻尼和固有频率
Eigenvalue Damping Freq. (rad/s)
-1.00e+000 + 1.41e+000i 5.77e-001 1.73e+000
-1.00e+000 - 1.41e+000i 5.77e-001 1.73e+000
[p, z] = pzmap(H) % 带输出参数，返回极点和零点
p =
-1.0000 + 1.4142i
-1.0000 - 1.4142i
z =
-0.5000
dcgain(H) % 直流增益，即  $s=0$  时的值
ans =
```

```

0.3333
norm(H, 2)          % 计算  $H_2$  范数，即脉冲响应的均方根
ans =
1.0408
[ninf, fpeak] = norm(H, inf) % 计算  $L_\infty$  范数（频率响应的峰值增益），并返回该频率
值
ninf =
1.0413
fpeak =
1.6511
sys = rss(3, 2, 2);           % 建立随机 SS 模型
pole(sys) 或 eig(sys.a) % 求 SS 模型的极点，等于求 A 的特征值，两条命令结果相同
ans =
-0.3178
-0.6488 + 0.5672i
-0.6488 - 0.5672i

```

### 10.3.3 状态空间实现

表 10-7 中列出了用于分析和执行状态空间各种实现方式的函数，这些函数只用于状态空间模型。

表 10-7 状态空间实现函数

函数	功能	函数	功能
canon	求状态空间的标准型	obsv	求可观矩阵
ctrb	求可控矩阵	obsvf	可观/不可观的阶梯形式分解
ctrbf	可控/不可控的阶梯形式分解	ss2ss	状态的相似变换
gram	求可控或可观的GRAM矩阵	ssbal	用对角相似变换均衡状态空间模型

#### 1. 可控和可观

设状态空间模型 sys 的数据为 A、B、C、D，则求模型 sys 的可控和可观矩阵方法为：

- 可控矩阵：Co = ctrb(A, B) 或 Co = ctrb(sys)
- 可观矩阵：Ob = obsv(A, C) 或 Ob = obsv(sys)

可控和可观的 GRAM 矩阵有比用函数 ctrb 和 obsv 计算的可控和可观矩阵更好的数值性能，更适合于研究模型的可控性和可观性，GRAM 矩阵常用于模型降阶和均衡实现。GRAM 矩阵的定义如下（ $W_c$  为可控矩阵， $W_o$  为可观矩阵）：

$$W_c = \int_0^{\infty} e^{At} BB^T e^{A^T t} dt, \quad W_o = \int_0^{\infty} e^{At} C^T C e^{A^T t} dt$$

计算 GRAM 矩阵的方法为：

- 可控矩阵：Wc = gram(sys, 'c')
- 可观矩阵：Wo = gram(sys, 'o')

## 2. 阶梯形式分解

如果状态空间模型可控矩阵的秩小于 A 的大小，那么必然存在如下的相似变换：

$$\bar{A} = TAT^T, \quad \bar{B} = TB, \quad \bar{C} = CT^T$$

变换后的阶梯形式模型中，所有不可控的都位于左上角，结果的形式如下：

$$\bar{A} = \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 \\ B_c \end{bmatrix}, \quad \bar{C} = [C_{uc} \quad C_c]$$

其中  $(A_c, B_c)$  是可控的，位于左上角的  $A_{uc}$  的所有特征值都是不可控的。

函数 ctrbf 对模型做可控性分解，其用法为

`[Abar, Bbar, Cbar, T, k] = ctrbf(A, B, C, tol)`

其中，Abar、Bbar、Cbar 就是分解后的阶梯形式矩阵  $\bar{A}$ 、 $\bar{B}$ 、 $\bar{C}$ ，T 是相似变换矩阵，k 是长度为 n（模型的阶数）的向量，其中的每一项表示在变换矩阵的每一步得到的可控状态个数，k 中的非零元素个数指示了计算 T 所需的步数，而 sum(k) 等于  $\bar{A}$  的可控部分  $A_c$  中的状态个数。参数 tol 指定计算的容差，缺省值是  $10*n*norm(A,1)*eps$ 。

与可控性阶梯形式分解类似，也可以进行可观性分解，其相似变换的形式和可控分解相同，分解的结果为：

$$\bar{A} = \begin{bmatrix} A_{vo} & A_{12} \\ 0 & A_o \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} B_{vo} \\ B_o \end{bmatrix}, \quad \bar{C} = [0 \quad C_o]$$

其中  $(C_o, A_o)$  是可观的，位于左上角的  $A_{vo}$  的所有特征值都是不可观的。

函数 obsvf 对模型做可观分解，其用法为

`[Abar, Bbar, Cbar, T, k] = obsvf(A, B, C, tol)`

其中，Abar、Bbar、Cbar 就是分解后的阶梯形式矩阵  $\bar{A}$ 、 $\bar{B}$ 、 $\bar{C}$ ，T 是相似变换矩阵，k 是长度为 n（模型状态的个数）的向量，其中的每一项表示在变换矩阵的每一步得到的可观状态个数，k 中的非零元素个数指示了计算 T 所需的步数，而 sum(k) 等于  $\bar{A}$  的可观部分  $A_o$  中的状态个数。参数 tol 指定计算的容差，缺省值是  $10*n*norm(A,1)*eps$ 。

## 3. 模型的均衡

在介绍模型均衡的函数 ssbal 之前，我们先看一下函数 ss2ss 的用法。

函数 ss2ss 对 SS 模型的状态向量 x 做相似变换  $\bar{x} = Tx$ ，生成等价的 SS 模型：

$$\dot{\bar{x}} = TAT^{-1}\bar{x} + TBu$$

$$y = CT^{-1}\bar{x} + Du$$

函数的调用格式为：

`sysT = ss2ss(sys, T)`

其中 T 是变换矩阵，它必须是可逆矩阵。sys 可以是连续的或离散的 SS 模型。

函数 ssbal 和 ss2ss 类似，也是对模型的状态向量 x 做相似变换  $\bar{x} = Tx$ ，但该函数在计算时引入了标量  $\alpha$ ，变换后的模型是均衡模型，形式如下：

$$\dot{\bar{x}} = TAT^{-1}\bar{x} + \frac{TB}{\alpha}u$$

$$y = \alpha CT^{-1}\bar{x} + Du$$

变换后的矩阵满足关系：矩阵  $\begin{bmatrix} TAT^{-1} & TB/\alpha \\ \alpha CT^{-1} & 0 \end{bmatrix}$  中行和列的范数近似相等。

函数 ssbal 的用法为：

`[sysb, T] = ssbal(sys, condT)`: 参数 condT 用于指定变换矩阵 T 的条件数的上限，由于用病态的矩阵 T 进行均衡会在无意中放大舍入误差，因而用 condT 来控制最坏情况下的舍入误差放大倍数，condT 的缺省值为 1/eps。

对如下的状态空间模型进行均衡，请比较程序后的均衡结果：

$$A = \begin{bmatrix} 1 & 10^4 \\ 0 & 100 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad C = [0.1 \quad 100]$$

`A = [1, 1e4; 0, 1e2];`

`B = [1; 1];`

`C = [0.1, 100];`

`sys = ss(A, B, C, 0);`

`ssbal(sys)`

均衡结果如下：

`a =`

	x1	x2
x1	1	9.7656
x2	0	100

`b =`

	u1
x1	0.015625
x2	16

`c =`

	x1	x2
y1	6.4	6.25

`d =`

	u1
y1	0

Continuous-time model.

#### 4. 模型的标准型

函数 canon 进行模型的标准型转换，该函数的调用格式为：

`[cys, T] = canon(sys, 'type')`

其中，返回值 T 是状态变换矩阵，可以缺省。参数'type'指定变换出的标准型的类型，它有两个选项：'modal'和'companion'，分别对应对角标准型和伴随矩阵标准型，这两种标准型都是针对模型的状态矩阵 A 所讲的。

1) 对角标准型：指矩阵 A 的实特征值都位于 A 的对角线上，复共轭特征值以  $2 \times 2$  的

模块形式位于 A 的对角线上。

2) 伴随矩阵标准型: 指原系统 sys 中矩阵 A 的特征多项式系数位于伴随矩阵的最右侧, 排列成一列。

#### 10.3.4 时间响应

时间响应研究 LTI 模型对特定类型的输入和扰动的时域瞬态行为。通过时间响应分析可以确定系统的上升时间、稳定时间、超调量和稳态误差等特征。控制系统工具箱中提供了相应的函数, 如表 10-8 所示, 这些函数使得分析过程变得相当简单。用这些函数可以对 TF、SS、ZPK 模型进行分析 (FRD 模型只能做频率响应分析)。

表 10-8 时间响应分析函数

函数	功能	函数	功能
impulse	脉冲响应	lsim	对任意输入的响应
initial	初始条件响应	step	阶跃响应
gensig	输入信号发生器		

##### 1. 信号发生器

函数 gensig 和其它几个函数不同, 它只是产生作为输入的信号, 调用格式为:

[u, t] = gensig(type, tau, tf, ts)

其中, type 是信号类型, 可以是 'sin' (正弦波)、'square' (方波)、'pulse' (脉冲); tau 是信号的周期, 单位是秒; tf 和 ts 是可选参数, 分别指定信号的持续时间和采样时间间隔, 缺省时自动指定; 输出 u 是产生信号的值, t 是采样时间点。

##### 2. 阶跃、脉冲、初始条件响应

###### (1) 基本用法

这三个函数 step、impulse 和 initial 自动产生适当的仿真范围对模型进行仿真, 并且一般不带输出参数, 自动绘制仿真曲线, 语法如下:

```
step(sys)
impulse(sys)
initial(sys, x0) % x0 是初始状态向量
```

注意: 如果想用这些函数产生的数据自己绘制图形, 只要带输出参数即可, 这种情况下, 该函数不再绘图。这也同样适用于频率响应分析函数 (见下一节)。这种用法比较简单, 请参看函数的帮助。

###### (2) 同时计算多个模型的响应

如果需要比较多个模型的时间响应, 可以把这些模型同时列在输入参数中, 还可以在每个模型名之后指定绘制该模型的曲线属性, 这样就会同时在一幅图中绘制出这些模型的曲线, 便于比较。例如:

```
step(sys1, '—', sys2, '--')
initial(sys1, sys2, ..., sysN, x0)
```

注意: 这种方式对所有的时间响应分析函数和频率响应分析函数 (见下一节)

都适用。以后不再重复。

### (3) MIMO 模型

对于 MIMO 模型，这些函数产生一个图形的阵列，每个 I/O 通道对应一幅图（对于函数 initial 和 lsim，每个输出对应一幅图），而且图形和通道的排列方式类似。

#### 例程 10-8 MIMO 系统脉冲响应分析

首先建立 MIMO 模型，然后绘制其脉冲响应图，如图 10-9 所示，注意比较图形和 I/O 通道的对应关系。

```
b = [tf(10,[1 2 10]), 0; tf(1,[-1 1]), tf(1,[1 -1])]; % 建立两输入两输出的TF模型
```

```
Transfer function from input 1 to output...
```

```
10
#1: -----
% 输入通道1到输出通道1，对应的图形在左上角
s^2 + 2 s + 10
-1
#2: -----
% 输入通道1到输出通道2，对应的图形在左下角
s - 1
```

```
Transfer function from input 2 to output...
```

```
#1: 0 %
% 输入通道2到输出通道1，对应的图形在右上角
1
#2: -----
% 输入通道2到输出通道2，对应的图形在右下角
s - 1
```

```
impulse(b)
```

### (4) 设定仿真范围

仿真范围可以由函数根据模型的动态特性自动确定，也可以直接指定仿真终止时间或采样时间向量。例如：

```
step(sys, 10) % 仿真从0到10秒
```

```
t = 0:0.1:10;
```

```
step(sys, t) % 仿真从0到10秒，且采样时间间隔0.1秒
```

注意，在指定时间向量  $t = [0:dt:tf]$  时，采样时间间隔  $dt$  要受如下限制：

- 对于离散系统， $dt$  必须和系统的采样时间匹配；
- 对于连续系统， $dt$  必须取得足够小以保证采样时获取系统瞬态响应的主要特征。
- 3. 对任意输入的响应

函数 lsim 可以仿真系统对任意输入信号的响应，其用法为：

```
lsim(sys, u, t)
```

其中， $t$  是采样时间向量，矩阵  $u$  的每一列表示一个输入，每一行对应一个采样时刻，因此它的列数应与系统 sys 的输入个数相同，它的行数应与向量  $t$  的长度相同。

该函数常和其它函数一起使用，用各种输入信号对系统进行仿真，观察响应。

下面计算一模型对三种不同输入信号的响应数据，并画图，结果如图 10-10 所示。

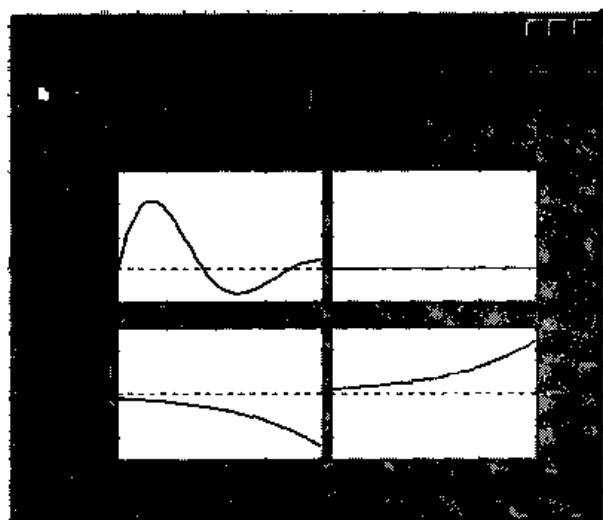


图 10-9 脉冲响应曲线

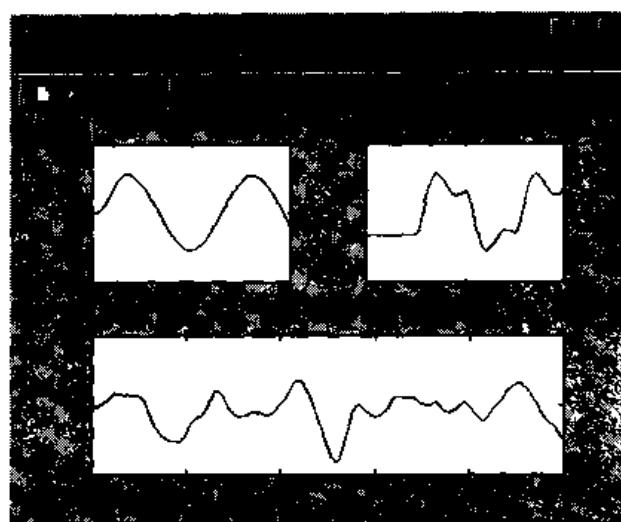


图 10-10 模型对不同输入信号的响应

```

t = 0:0.01:10;
s1 = sin(t); % 产生正弦信号
s2 = randn(size(t)); % 产生随机信号
[s3, tt] = gensig('square', 5, 10, 0.01); % 用gensig函数产生方波
sys = tf(10, [1 2 10]); % 建立TF模型
[y1, t1] = lsim(sys, s1, t); % 计算模型对正弦信号的响应, y1是响应数据, t1是时间
[y2, t2] = lsim(sys, s2, t);
[y3, t3] = lsim(sys, s3, tt); % 计算模型对方波的响应
subplot(2,2,1), plot(t1,y1), title('Input: Sine Sig') % 绘制模型对正弦信号的响应
subplot(2,2,2), plot(t3,y3), title('Input: Square Sig') % 绘制模型对方波的响应
subplot(2,1,2), plot(t2,y2), title('Input: Rand Sig') % 绘制模型对随机信号的响应

```

### 10.3.5 频率响应

表 10-9 列出了 MATLAB 控制系统工具箱中提供的频域分析函数，其中有直接计算系统频率响应的函数，也有绘制控制系统几种典型的频率响应曲线（如 Bode 图，Nyquist 曲线以及 Nichols 曲线）的函数。这大大简化了频域分析的过程。

表 10-9 频域分析函数

函数	功能
bode	计算并绘制Bode图
evalfr	计算模型在某个指定复频率点上的频率响应（不包括FRD模型）
fresp	计算在一组实频率点上的频率响应
margin	计算增益裕度和相位裕度
ngrid	在Nichols图上添加Nichols网格线
nichols	计算并绘制Nichols图
nyquist	计算并绘制Nyquist图
sigma	计算并绘制奇异值图

### 1. 频率响应的计算

在一个LTI系统受到频率为 $\omega$ 的正弦信号激励时，它的输出仍为正弦信号，但其幅值与输入信号成 $M(\omega)$ 的比例关系，而且输入信号与输出信号之间有一个相位差 $\phi(\omega)$ ，比例和相位差随 $\omega$ 的变化而变化，这样就可以通过 $M(\omega)$ 和 $\phi(\omega)$ 来表示系统的特性了，这种方法即为系统的频域分析方法。 $M(\omega)$ 和 $\phi(\omega)$ 分别为频率响应的幅值和相位数据。

对于连续系统，在频率 $\omega$ 处的频率响应就是传递函数 $H(s)$ 当 $s = j\omega$ 时的值。对于状态空间模型，频率响应按照如下关系计算：

$$H(j\omega) = D + C(j\omega I - A)^{-1}B$$

对于离散系统，频率点通过变换 $z = e^{j\omega T}$ 映射到单位圆上。

函数 freqresp 计算模型的频率响应，其用法为：

`H = freqresp(sys, w)`

其中，向量`w`是用于指定实频率点的向量，频率的单位必须是 rad/s。返回值`H`是一个三维数组，它的维数是“输出的个数×输入的个数×向量`w`的长度”，对于 SISO 系统，`H(1, 1, 5)`是在频率点`w(5)`上的响应数据，对于 MIMO 系统，频率点`w(5)`上的响应数据是`H(:, :, 5)`。

函数 evalfr 计算指定的某个频率（复数）处的频率响应，它相当于函数 freqresp 的简化。

### 2. 频率响应曲线

绘制各种频率响应曲线的函数有 bode、nichols、nyquist 和 sigma，它们的用法基本相同，以 bode 为例，其调用格式有如下几种情况：

1) `bode(sys)`: 绘制任意 LTI 模型的 Bode 图，模型可以是连续的或离散的，也可以是 SISO 或 MIMO。对于 MIMO 模型，生成 Bode 图阵列，每幅图对应一个特定的 I/O 通道。频率范围由函数基于系统的零点和极点自动确定，频率向量是对数间隔的。

2) `bode(sys, w)`: 用参数`w`指定频率范围或频率点，指定范围时，`w`的格式为`{wmin, wmax}`，指定频率点时，`w`为向量形式，一般用函数 logspace 来产生对数间隔的频率向量。

3) `[mag, phase, w] = bode(...)`: 不画图，返回频率响应的幅值`mag`，相位`phase`和频率点`w`。

这里介绍的 bode 函数的用法也同样适用于其它几个函数，这里不再重复了。

函数 ngrid 用于添加 Nichols 网格，调用该函数不带任何参数。

下面举例说明这些函数的用法。

如下一组命令分别绘制模型的 Bode 图、Nyquist 图和 Nichols 图，并用 ngrid 函数绘制了完整的 Nichols 网格。结果如图 10-11 所示。

```
sys = tf(10, [1 2 10]);
subplot(2,2,1), bode(sys) % Bode 图
subplot(2,2,2), nyquist(sys) % Nyquist 图
subplot(2,2,3), ngrid % 完整的 Nichols 网格
subplot(2,2,4), nichols(sys), ngrid % Nichols 图，带网格
```

### 3. margin 函数

margin 函数计算模型的增益裕度和相位裕度。该函数用法如下：

1)  $[G_m, P_m, W_{cg}, W_{cp}] = \text{margin}(\text{sys})$ : 由模型求增益裕度  $G_m$  和相位裕度  $P_m$ , 并求出增益裕度和相位裕度处相应的频率值  $W_{cg}$  和  $W_{cp}$ 。

2)  $[G_m, P_m, W_{cg}, W_{cp}] = \text{margin}(\text{mag}, \text{phase}, \text{w})$ : 由频率响应的幅值和相位数据来求增益裕度和相位裕度,  $w$  是相应的频率点向量。

3)  $\text{margin}(\text{sys})$ : 画出开环响应 Bode 图, 用竖直线标出增益裕度和相位裕度。

#### 例 10.18 计算增益裕度和相位裕度

```
sys = tf([0.05 0.045], [1 -1.8 0.9], 0.1); % 离散 TF 模型, Ts = 0.1
[Gm, Pm, Wcg, Wcp] = margin(sys) % 求模型的增益裕度和相位裕度
```

$G_m =$

2.2222

$P_m =$

15.4799

$W_{cg} =$

5.6527

$W_{cp} =$

4.4159

margin(sys) % 绘制 Bode 图, 如图 10-12 所示, 并标出裕度

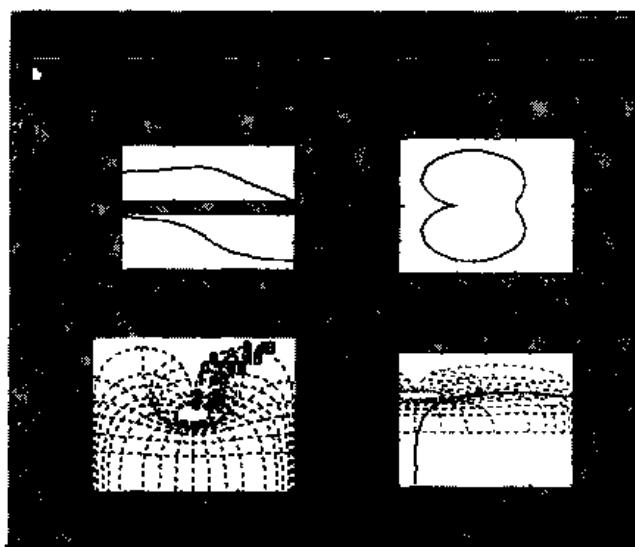


图 10-11 频率响应曲线

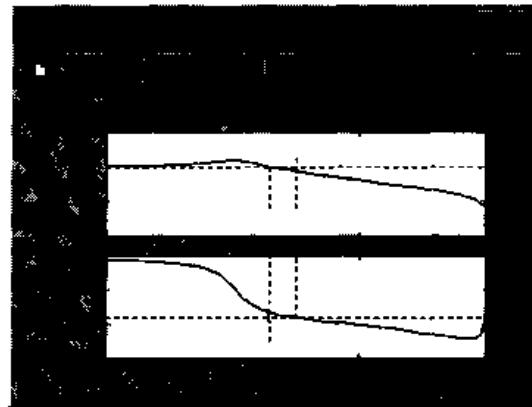


图 10-12 模型的裕度

### 10.3.6 模型降阶

在控制系统的研究中, 模型降阶技术起着很重要的作用。模型降阶的目的就是使高阶系统由一个相对低阶的模型尽可能好地近似, 使得高阶模型可以按照对低阶模型的设计方法进行近似设计。

在 MATLAB 控制系统工具箱中提供了如表 10-10 所示的几种模型降阶算法程序, 由这些程序很容易求出降阶模型。

表 10-10 模型降阶函数

函数	功 能	函数	功 能
balreal	状态空间的均衡实现	modred	从均衡实现的SS模型中删除状态
minreal	最小实现或零极点对消	sminreal	在结构上的最小实现

### 1. 最小实现

函数 minreal 删 除 SS 模型中不可控或不可观的状态，或对消 TF 模型和 ZPK 模型中的零极点对。函数 sminreal 从 SS 模型中删除不影响 I/O 响应的任何状态。它们都可以得到模型的最小实现。两个函数的用法如下：

1)  $\text{sysr} = \text{minreal}(\text{sys}, \text{tol})$ : 参数 sys 可以是 SS、TF、ZPK 中的任何类型，可选参数 tol 用于指定计算中的误差容限，缺省值为  $\text{sqrt}(\text{eps})$ 。输出模型 sysr 有最小的阶数以及和原始模型 sys 相同的响应特性。

2)  $\text{msys} = \text{sminreal}(\text{sys})$ : 该函数的参数 sys 必须是 SS 模型，输出模型 msys 的所有状态就是原始模型 sys 中的状态，而且 msys 的输入/输出响应和 sys 相同。

### 2. 均衡实现

对于已经是最小实现的模型，可以用函数 balreal 和 modred 进一步降低模型的阶数。

函数 balreal 用法为：

1)  $\text{sysb} = \text{balreal}(\text{sys})$ : 利用 GRAM 可控和可观矩阵计算 sys 的均衡实现 sysb，如果 sys 不是 SS 模型，则该函数自动把它转换为 SS 模型，结果 sysb 必然是 SS 模型。

2)  $[\text{sysb}, \text{g}, \text{T}, \text{Ti}] = \text{balreal}(\text{sys})$ : 同时返回 GRAM 矩阵的对角线向量 g，T 是相似变换矩阵，Ti 是 T 的逆阵。

函数 modred 通常和 balreal 函数联合使用，该函数有两种降阶算法，对应的调用格式为：

1)  $\text{rsys} = \text{modred}(\text{sys}, \text{elim})$  或  $\text{rsys} = \text{modred}(\text{sys}, \text{elim}, \text{'mdc'})$ : 产生的降阶模型 rsys 和原始模型 sys 的直流增益（或阶跃响应的稳态）互相匹配。向量 elim 指定要除去的状态，降阶后的模型 rsys 有比较少的状态。注意该函数的输入模型 sys 只能是 SS 模型。

2)  $\text{rsys} = \text{modred}(\text{sys}, \text{elim}, \text{'del'})$ : 简单地删除指定的状态，不保证直流匹配，该方法在频率响应上的近似性更好。

如果 SS 模型 sys 经过函数 balreal 做均衡，而 GRAM 矩阵有 m 个较小的对角线元素，则可以用 modred 函数除去模型最后的 m 个状态。

下面一段程序先求模型的均衡实现，并得到 GRAM 矩阵的对角线值 g，由于 g 的后两个值比较小，因而用 modred 进一步除去模型的最后两个状态。最后比较原始模型和两次降阶后的 Bode 图，如图 10-13 所示。图中实线表示原始模型，‘o’

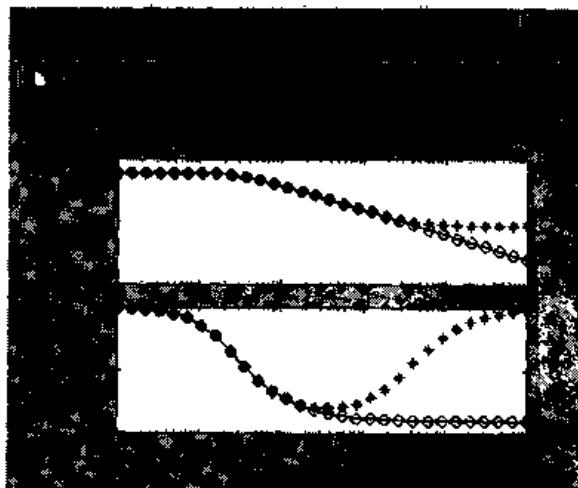


图 10-13 降阶前后的模型特性比较

表示均衡实现的模型，‘\*’表示用 modred 函数降阶后的模型。

```
sys = tf([1 11 36 26],[1 15 75 154 100]);
[sys1, g] = balreal(sys)
g =
    0.1400
    0.0112
    0.0015
    0.0004
sysm = modred(sys1, 3:4, 'mdc')      % 除去模型的第三个和第四个状态
bode(sys, '-', sys1, 'o', sysm, '*')
```

## 10.4 LTI Viewer

上一节我们介绍了 LTI 模型时域和频域分析的各种方法和实现函数，这些 MATLAB 函数使得 LTI 模型的分析过程相当简单。

除了这些函数，控制系统工具箱还提供了更为简单和直观的方式：LTI Viewer（观察器），一个具有良好图形用户界面（GUI）的交互式 LTI 模型分析工具，利用 LTI Viewer 可以在图形方式下交互地进行 LTI 模型的各种时域和频域特性分析，并能够直接观察图形方式的分析结果。

调用 LTI Viewer 的方法是在 MATLAB 命令窗口输入命令：ltiview。其窗口如图 10-14 所示。本节将介绍该窗口界面的用法。

### 10.4.1 菜单

LTI Viewer 有三个菜单：File、Tools 和 Help，Help 菜单为用户提供帮助，不再多做介绍。File 和 Tools 两个菜单的内容如图 10-15 所示。

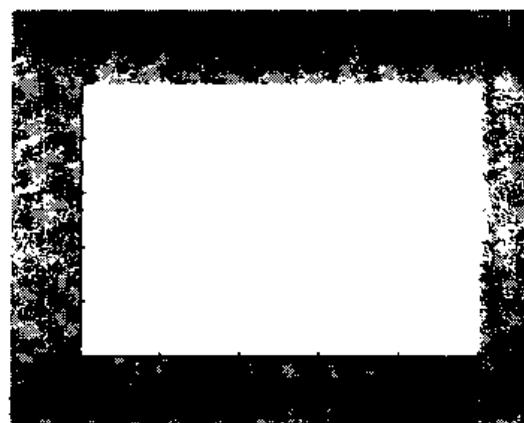


图 10-14 LTI Viewer 窗口

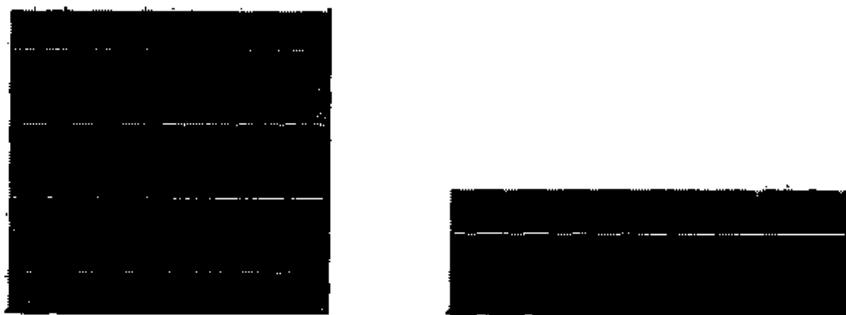


图 10-15 LTI Viewer 菜单

#### 1. File 菜单

File 菜单的主要作用是在 LTI Viewer 的工作空间对模型对象和分析结果的图形进行操作。各菜单项的功能分别如下：

**New Viewer:** 新建一个 LTI Viewer 窗口。

**Import:** 从 MATLAB 工作空间引入 LTI 模型对象到 LTI Viewer 工作空间。

**Export:** 把 LTI 模型对象从 LTI Viewer 工作空间输出到 MATLAB 工作空间或保存为 MAT 文件。

**Delete Systems:** 从 LTI Viewer 工作空间删除模型。

**Refresh Systems:** 刷新 LTI Viewer 工作空间中的模型。

**Print:** 打印 LTI Viewer 中的图形。

**Print to Figure:** 把 LTI Viewer 中的图形拷贝为 MATLAB 图形。

**Close Viewer:** 关闭当前窗口。

## 2. Tools 菜单

Tools 菜单主要用于对 LTI Viewer 做各种设置。下面介绍其菜单项。

### 1) Viewer Configuration

选择该项后出现图 10-16 所示的窗口，在该窗口中直观地设置 LTI Viewer 中显示的响应图形个数和各种响应的排列方式。窗口左侧是图形子窗口的排列格式，右侧选择每个子窗口对应的响应类型。

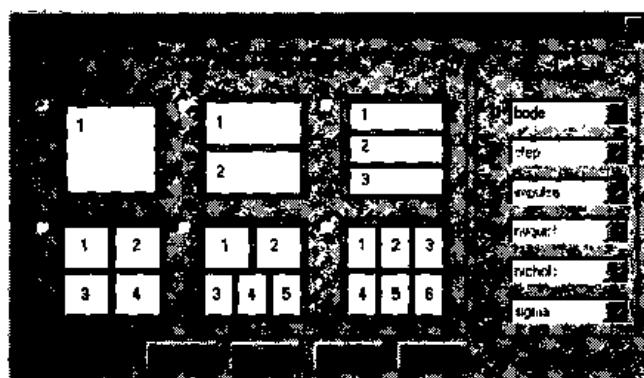


图 10-16 LTI Viewer 配置

### 2) Response Preferences

选择该项后出现图 10-17 所示的窗口，在该窗口中设置时间响应（左侧）和频率响应（右侧）分析的相关参数。

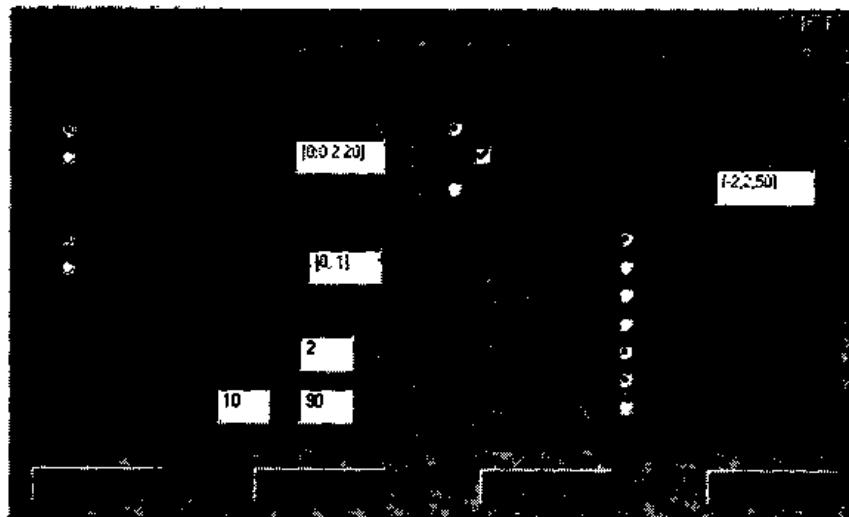


图 10-17 响应参数设置

时间响应中可以设置的参数有：

- 时间向量 (Time Vector): 可以选择为自动设置 (缺省) 或自己指定。
- Y 周坐标范围 (Y-axis Range): 可以选择为自动设置 (缺省) 或自己指定。
- 绘图选项 (Plot Option): 其中 settling time (稳定时间) 中设置的百分数是指响应值和稳态值相差的百分比，把达到这个百分比以内的时刻作为系统的稳定时间参数。rise time (上升时间) 中设置的两个百分数仍然是和响应稳态值的比，由这两个百分数确定的区间作为系统的上升时间参数。这里的设置主要是用于在时间响应曲线上标注稳定时间和上升时间两个参数的。参见 10.4.2 节。

频率响应中可以设置的参数有：

- 频率向量 (Frequency Vector): 可以选择为自动设置或自己指定。
- 幅值单位 (Magnitude in): 可选分贝 (decibels)、绝对单位 (absolute) 或对数 (logarithmic)。
- 相位单位 (Phase in): 可选弧度 (radians) 或度 (degrees)。
- 频率单位 (Frequency in): 可选弧度/秒 (Radians/second) 或 Hz。

### 3) Linestyle Preferences

选择该项后出现图 10-18 所示的窗口，在该窗口中设置曲线的线型、颜色，标记的形状等。

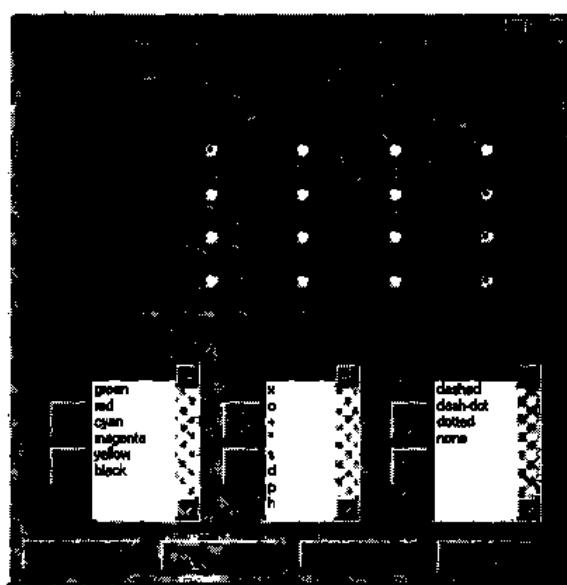


图 10-18 线型设置

窗口的上方设置区分系统、输入、输出和通道的方法。缺省时，只用颜色区别系统。输入、输出和通道不做区分。

窗口下方设置颜色 (color)、标记 (marker) 和线型 (linestyle) 的选用次序。

## 10.4.2 应用举例和上下文菜单

本节通过一个分析实例来说明 LTI Viewer 的用法，并进一步介绍如何使用 LTI Viewer 中的上下文菜单（在窗口中按下鼠标右键弹出的菜单）。

首先在 MATLAB 工作空间建立两个 LTI 模型，命令如下：

```
sys = tf([1 11 36 26],[1 15 75 154 100]); % 连续 TF 模型
sys_z = tf([0.05 0.045], [1 -1.8 0.9], 0.1); % 离散 TF 模型
```

然后在 LTI Viewer 的 File 菜单下选择 Import，在出现的对话框中同时选择两个模型（按住 Ctrl 键），并按 OK。这时两个模型被引入到 LTI Viewer 工作空间，同时在 LTI Viewer 的窗口中按照缺省设置画出这两个模型的阶跃响应曲线。

在图形窗口上方按下鼠标右键，出现如图 10-19 所示的上下文菜单。

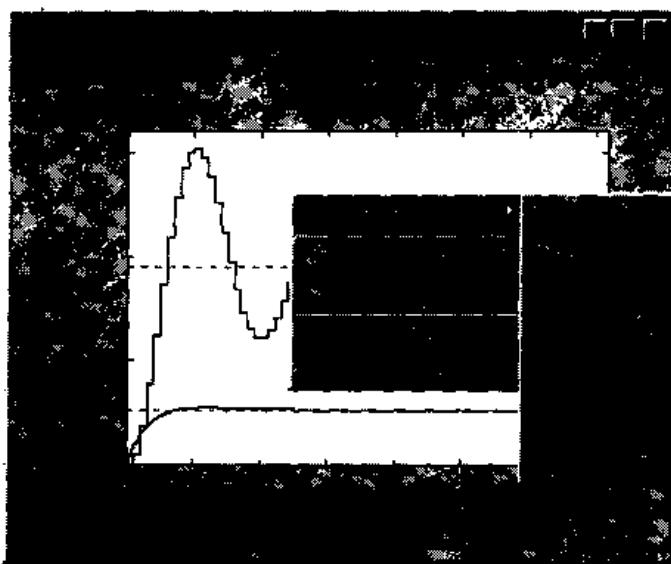


图 10-19 图形窗口和上下文菜单

该菜单主要用来调整图形窗口的显示。它的几个菜单项的功能如下：

- 图 10-19 中列出了 Plot Type 菜单项的内容，可以在这里选择需要绘制的各种时间响应和频率响应类型。每幅图只能选择一种类型。
- Systems 菜单项包括当前 LTI Viewer 工作空间中的全部模型，在这里选择要分析的模型，可以是一个或多个。
- Characteristics 菜单项对不同类型的响应曲线，标出相关的特征量。例如，阶跃响应的特征量有四个：峰值响应（Peak Response）、稳定时间(Settling Time)、上升时间(Rise Time)、稳定状态(Steady State)。而对于频率响应曲线，主要的特征量是稳定性裕度（Stability Margins）。
- 菜单项 Zoom 用来对图形进行放大和缩小，Grid 在图形窗口添加网格线。

下面我们把窗口中显示的图形个数改为 4 个，在 Tools 菜单下的 Viewer Configuration 中做相应的设置。

然后对 LTI Viewer 窗口中的每一幅图分别用上下文菜单进行调整。

左上角的图形只绘制连续模型 sys 的阶跃响应（在 Systems 菜单项中去掉 sys\_z 模型的选项，下同），并在 Characteristics 菜单项中选择标出系统的峰值响应和上升时间。

左下角的图形绘制模型 sys 的 Bode 图，在 Plot Type 菜单项中选择 Bode，并在 Characteristics 菜单项中选择标出系统的稳定性裕度。

右上角的图形绘制离散模型 sys\_z 的脉冲响应（在 Systems 菜单项中去掉 sys 模型的选

项, 下同), 在 Plot Type 菜单项中选择 Impulse, 并在 Characteristics 菜单项中选择标出峰值响应和稳定时间。

右下角的图形绘制离散模型 sys\_z 的 Nyquist 图, 在 Plot Type 菜单项中选择 Nyquist, 并在 Characteristics 菜单项中选择标出稳定性裕度。

设置后的最终结果如图 10-20 所示。

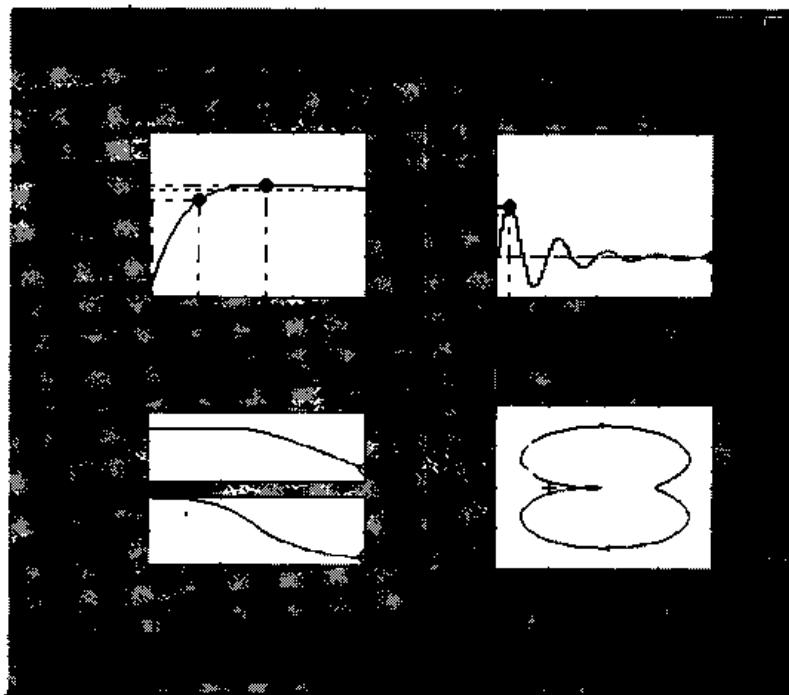


图 10-20 LTI Viewer 窗口的显示

## 10.5 控制系统设计

控制系统设计的理论和算法多种多样, 在 MATLAB 中除了控制系统工具箱中提供的经典设计方法之外, 另外还有其它几个专门的控制系统设计工具箱, 它们是:

**QFT 工具箱:** 采用定量反馈理论 (Quantitative Feedback Theory) 的设计方法, 常用于带有很大对象不确定性的单变量系统、多变量系统以及各种高度非线性系统和时变系统的鲁棒设计, 这种方法既可以用于最小相位系统的设计, 也可以用于非最小相位系统的设计。该工具箱在 MATLAB 路径中的位置是\toolbox\qft。

**鲁棒控制工具箱:** 侧重于控制系统的现代时域设计方法, 主要用于LQG鲁棒控制器和  $H_{\infty}$  控制器设计。该工具箱在MATLAB路径中的位置是\toolbox\robust。

**$\mu$  分析与综合工具箱:** 在鲁棒控制工具箱出现之后,  $H_{\infty}$  等技术又有了很大的发展, 为了将最新的理论研究成果用 MATLAB 加以实现, 就出现了  $\mu$  分析与综合工具箱。所谓  $\mu$  分析即系统的结构奇异值分析。该工具箱主要针对连续与离散系统的  $H_{\infty}$  最优设计、 $H_{\infty}$  最小熵设计、 $H_{\infty}$  回路整形设计等。该工具箱在 MATLAB 路径中的位置是\toolbox\mutools。

上面几个工具箱不在本书的范围内, 请读者参看相关书籍或 MATLAB 帮助。本节将介绍控制系统工具箱中提供的几种经典设计方法。

### 10.5.1 根轨迹法

根轨迹法用于研究闭环系统极点位置增益变化对系统的影响，而这些位置为时间和频率响应提供了直接的信息。

控制系统工具箱中关于根轨迹的函数如表 10-11 所示。

表 10-11 根轨迹函数

函 数	功 能
pzmap	绘制零极点图（参见 10.3.2 节）
rlocut	根轨迹设计 GUI 工具（在 10.6 节中将专门介绍）
rlocfind	从根轨迹图中选择反馈增益
rlocus	计算并绘制 Evans 根轨迹
sgrid	在根轨迹或零极点图中添加 s 平面网格
zgrid	在根轨迹或零极点图中添加 z 平面网格

#### 1. 计算并绘制根轨迹

函数 rlocus 计算 SISO 开环模型的 Evans 根轨迹，根轨迹以反馈增益的函数形式给出了闭环极点的轨道（假定为负反馈）。该函数的一般用法为：

**rlocus(sys)**

计算并绘制开环 SISO 模型 sys 的根轨迹，sys 可以是如图 10-21 所示的三种模型中的任何一种。对于模型 (a)， $sys = G$ ；对于模型 (b)， $sys = FG$ ；对于模型 (c)， $sys = GC$ 。

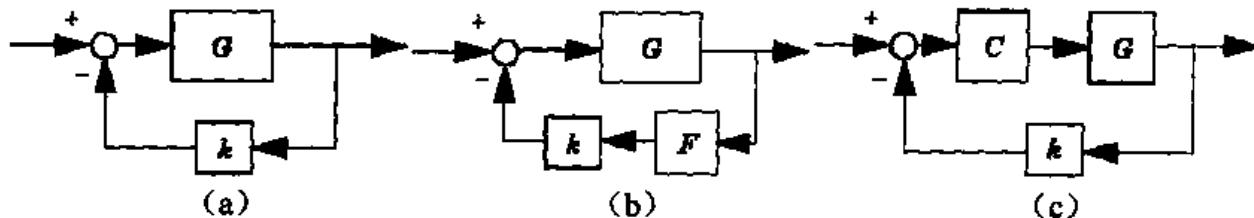


图 10-21 三种负反馈模型

如果模型 sys 有传递函数形式： $h(s) = \frac{n(s)}{d(s)}$ ，则闭环极点就是如下方程的根：

$$d(s) + kn(s) = 0$$

rlocus 函数可以自适应选择一组正增益 k，产生平滑的曲线，用户也可以自己指定向量 k，格式为：

**rlocus(sys, k)**

函数 rlocus 还可以带输出参数，格式为：

**[r, k] = rlocus(sys)** 或 **r = rlocus(sys, k)**

这时，返回自适应选择的增益 k 和对于这些增益的复数根的位置 r。矩阵 r 的列数和向量 k 的长度相同，它的第 m 列的元素是对于增益 k(m) 的闭环系统的根。

#### 2. 选择反馈增益

函数 rlocfind 返回在根轨迹中和一些极点相关的反馈增益。其用法为：

1) **[k, poles] = rlocfind(sys)**：从函数 rlocus 生成的模型 sys 的根轨迹图中交互选择反馈

增益。运行 rlocfind 函数后，在根轨迹图上出现十字形光标，用来选择极点位置，和选中的点相关联的根轨迹增益返回到 k 中，列向量 poles 中包含了该增益的闭环极点。使用该函数时，模型 sys 的根轨迹图必须在当前图形窗口。

2)  $[k, \text{poles}] = \text{rlocfind}(\text{sys}, p)$ : 用向量 p 指定期望的极点位置，并计算这些位置上的根轨迹增益。向量 k 的第 m 项是根据极点位置  $p(m)$  计算的增益，矩阵 poles 的第 m 列是相应的闭环极点。

#### 例 10.29 根轨迹的绘制：

下面一段程序计算并绘制离散模型 sys 的根轨迹图，如图 10-22 所示。然后用 rlocfind 函数在图中选择极点位置（十字光标如图所示），得到反馈增益的值。

```
sys = tf([0.05 0.045], [1 -1.8 0.9], 0.1);
rlocus(sys)
k = rlocfind(sys) % 执行后出现下面一条信息，等待选择极点位置
Select a point in the graphics window
selected_point = % 选择极点位置后得到位置和增益值
0.8896 + 0.2881i
k =
0.1029
```

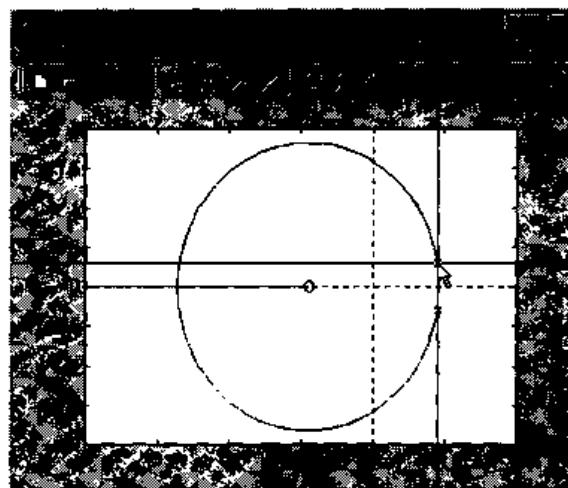


图 10-22 根轨迹图和交互选择反馈增益

### 10.5.2 极点配置

对于给定的控制系统模型，经常希望引入某种控制器（动态补偿器），使得该系统的闭环极点移动到某个指定的位置，因为闭环极点的位置对系统的时间响应性能（如上升时间、稳定时间等）有直接影响。

极点配置是针对状态空间模型的（设其数据为 A、B、C、D），其它模型先要用 ss 函数进行转换。

#### 1. 极点配置方法

极点配置的过程就是设计动态补偿器的过程，包括两步：状态反馈增益选择，状态估计器设计。

### 1) 状态反馈增益选择

当状态反馈  $u = -Kx$  时, 闭环系统的动态特性为:

$$\dot{x} = (A - BK)x$$

闭环极点就是  $A - BK$  的特征值。采用极点配置算法, 可以计算得到一个增益矩阵  $K$  把这些极点配置到复平面上任何希望的位置 (只要  $A, B$  是可控的)。

### 2) 状态估计器设计

如果不是系统的所有状态  $x$  都可以测定, 则不能实现状态反馈规则  $u = -Kx$ 。这时, 可以建立一个状态估计  $\xi$ , 使得  $u = -K\xi$  保留同样的极点配置属性。这可以通过建立如下的状态估计器来实现:

$$\dot{\xi} = A\xi + Bu + L(y - C\xi - Du)$$

估计器的极点是  $A - LC$  的特征值, 通过选择合适的估计器增益矩阵  $L$  可以任意配置。经验表明, 估计器的动态性能应比控制器快。

在  $u = -Kx$  中用状态估计  $\xi$  代替  $x$ , 产生如下的动态输出反馈补偿器:

$$\begin{aligned}\dot{\xi} &= [A - LC - (B - LD)K]\xi + Ly \\ u &= -K\xi\end{aligned}$$

得到的闭环系统动态特性是:

$$\begin{bmatrix} \dot{x} \\ e \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}, \quad e = x - \xi$$

## 2. 极点配置函数

极点配置用控制系统工具箱中的函数很容易实现, 这些函数可以计算增益矩阵  $K$  和  $L$  以实现期望的闭环极点配置, 还可以用这些增益产生状态估计器和动态补偿器。函数如表 10-12 所示。

表 10-12 极点配置函数

函数	功 能	函数	功 能
acker	SISO系统极点配置	place	MIMO系统极点配置
estim	根据给定的增益产生状态估计器	reg	根据给定的状态反馈和估计器增益产生输出反馈补偿器

### (1) acker 函数

该函数利用 Ackermann 公式对 SISO 系统进行极点配置, 用法为:

$k = \text{acker}(A, b, p)$

其中,  $A, b$  为 SS 系统模型的数据, 向量  $p$  中是期望的闭环极点位置, 返回值是增益向量  $k$ 。

### (2) place 函数

该函数对 MIMO 系统（也可以是 SISO 系统）进行极点配置，用法为：

$K = \text{place}(A, B, p)$

其中， $A$ 、 $B$  为 SS 系统模型的数据，向量  $p$  中是期望的闭环极点位置，返回值是增益矩阵  $K$ 。

#### (3) estim 函数

该函数用法如下：

$est = \text{estim}(\text{sys}, L)$  或  $est = \text{estim}(\text{sys}, L, \text{sensors}, \text{known})$

其中， $L$  是估计器增益矩阵，返回值为相应的估计器。参数  $sensors$  和  $known$  是向量，它们指定可以测定的输出和已知的输入，产生的估计器  $est$  用它们计算输出和状态的估计。缺省情况下，函数  $\text{estim}$  认为所有的输入都是已知的，所有的输出都是可测定的。

#### (4) reg 函数

该函数用法为：

$r\text{sys} = \text{reg}(\text{sys}, K, L)$  或  $r\text{sys} = \text{reg}(\text{sys}, K, L, \text{sensors}, \text{known}, \text{controls})$

其中， $K$  和  $L$  分别为状态反馈增益矩阵和估计器增益矩阵。返回值  $r\text{sys}$  是模型  $\text{sys}$  的动态补偿器。向量  $sensors$  和  $known$  的作用与函数  $\text{estim}$  中的参数相同，参数  $controls$  指定可控的输入。

已知状态方程  $A$ 、 $B$ ，要引入状态反馈矩阵  $K$ ，使得闭环系统的极点位置为  $-3, -1 \pm i$ ，程序如下：

```

A = [0 3 1; 10 2 8; -2 0 -2];
B = [-2 0; 4 -3; -1 1];
p = [-3; -1-i; -1+i];
K = place(A, B, p)
place: ndigits= 15
K =
    -1.7630   -2.0007   -1.9523
    -5.6457   -4.3761   -5.6039
eig(A-B*K)'          % 验证极点的位置
ans =
    -1.0000 - 1.0000i  -1.0000 + 1.0000i  -3.0000

```

### 10.5.3 LQG 设计

LQG (Linear-Quadratic-Gaussian, 线性二次型高斯) 设计是一种基于状态空间的现代最优控制设计技术。它是经典的 LQ (线性二次型) 最优控制设计方法的拓展，考虑比经典 LQ 设计更复杂的情况，即状态变量和输出变量测定时存在随机扰动的情况。对于 LQG 问题，当扰动信号为零时，就是经典的 LQ 问题，因而 LQ 问题是 LQG 问题的特例。

本节以连续系统的为例介绍 LQG 设计，离散系统与之类似，不做专门讨论。

#### 1. LQG 设计原理

LQG 设计的系统框图如图 10-23 所示。

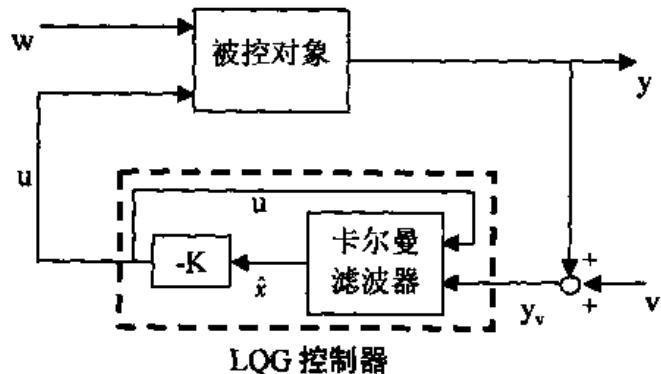


图 10-23 LQG 控制结构

设计的目标是把输出  $y$  调整到近似为 0。被控对象由过程噪声  $w$  和控制信号  $u$  驱动，LQG 控制器依靠噪声测量  $y_v = y + v$  产生控制信号。

被控对象的 SS 模型可以表示为：

$$\dot{x} = Ax + Bu + Gw$$

$$y_v = Cx + Du + Hw + v$$

其中， $w$ （过程噪声）和  $v$ （测量噪声）均为白噪声信号。

LQG 控制器由两部分组成：最优状态反馈增益和卡尔曼（Kalman）滤波器。这也就是 LQG 控制器设计的两个步骤。

### (1) 最优状态反馈增益

在 LQG 设计中，系统的性能由线性二次型控制指标来衡量，也就是下面的目标函数：

$$J(u) = \int_0^{\infty} \{x^T Q x + 2x^T N u + u^T R u\} dt$$

其中， $Q$ 、 $N$ 、 $R$  是用户定义的加权矩阵，用于均衡控制器的性能和控制效果。当  $N$  为 0 时，就是经典的 LQ 问题。

当前设计的目的就是找到合适的状态反馈规则  $u = -Kx$  使目标函数  $J(u)$  达到最小。也就是要计算出矩阵  $K$ （称为 LQ 最优化增益），这要通过解如下的 Riccati 方程（自变量  $S$ ）：

$$A^T S + S A - (S B + N) R^{-1} (B^T S + N^T) + Q = 0$$

得到解  $S$  后，再由下面的关系式计算出  $K$ ：

$$K = R^{-1} (B^T S + N^T)$$

### (2) 卡尔曼滤波器

与极点配置中类似，状态反馈规则  $u = -Kx$  没有全部的状态测量是不能实现的。可以用状态估计  $\hat{x}$  来代替状态  $x$ ，使得  $u = -K\hat{x}$  对于输出反馈问题仍保持最优。状态估计  $\hat{x}$  由如下的卡尔曼滤波器产生：

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_v - C\hat{x} - Du)$$

这个滤波器用已知的输入  $u$  和可测定的  $y_v$  来产生状态估计  $\hat{x}$ 。其中，滤波器增益  $L$  通

过解 Riccati 方程得到，方程的系数 Q、R、N 根据过程噪声 w 和测量噪声 v 如下确定：

$$E(w) = E(v) = 0, \quad E(ww^T) = Q, \quad E(vv^T) = R, \quad E(wv^T) = N$$

卡尔曼滤波器对于处理 Gaussian 白噪声是最优估计器。它使状态估计误差  $x - \hat{x}$  的渐进协方差  $P = \lim_{t \rightarrow \infty} E((x - \hat{x})(x - \hat{x})^T)$  最小。

设计出的卡尔曼滤波器和被控对象组成的系统称为卡尔曼估计器，后面介绍的 MATLAB 函数就是直接给出估计器的模型，该模型的方程如下：

$$\begin{bmatrix} \dot{\hat{x}} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D \\ 0 \end{bmatrix} u$$

其中输出估计  $\hat{y}$  就是被控对象的真实输出。

### (3) 最终的 LQG 控制器

上面的两步完成后，只要把 LQ 最优化增益 K 和卡尔曼滤波器象图 10-23 中那样连到一起，就生成了 LQG 控制器。LQG 控制器完整的状态空间方程如下：

$$\begin{aligned} \dot{\hat{x}} &= [A - LC - (B - LD)K] \hat{x} + Ly, \\ u &= -K\hat{x} \end{aligned}$$

这就是 LQG 设计需要得到的最终结果。

## 2. LQG 设计函数

控制系统工具箱中提供了比较全面的 LQG 设计函数，如表 10-13 所示。

表 10-12 极点配置函数

函数	功 能	函数	功 能
care	求解连续 Riccati 方程	lqr	计算连续系统的 LQ 最优化增益
dare	求解离散 Riccati 方程	dlqr	计算离散系统的 LQ 最优化增益
kalman	卡尔曼估计器	lqrds	计算连续被控对象的离散 LQ 增益
kalmd	连续被控对象的离散卡尔曼估计器	lqry	计算带输出加权的 LQ 最优化增益
lqgreg	根据给定的 LQ 增益和卡尔曼滤波器求 LQG 控制器		

以前面介绍的 LQG 设计原理为基础，这些函数的用法都很直观。我们把针对连续系统的函数给以简单介绍。离散系统函数与之类似。

### (1) 求解 Riccati 方程

函数 care 解连续 Riccati 方程，它的不同调用格式对应着不同类型的 Riccati 方程：

- $[X, L, G, rr] = \text{care}(A, B, Q)$ : 求解的方程形式如下（注意以 X 为自变量）：

$$Ric(X) = A^T X + XA - XBB^T X + Q = 0$$

- $[X, L, G, rr] = \text{care}(A, B, Q, R, S, E)$ : 求解的方程形式如下（注意以 X 为自变量）：

$$Ric(X) = A^T X E + E^T X A - (E^T X B + S) R^{-1} (B^T X E + S^T) + Q = 0$$

两种调用方式的返回参数相同，它们的含义分别为：

- X 是 Riccati 方程的解；
- L 是  $A - BB^T X$  的特征值；
- G 是增益矩阵  $G = B^T X$ ；
- rr 是相对留数  $rr = \frac{\|Ric(X)\|_F}{\|X\|_F}$ 。

### (2) 计算 LQ 增益

计算连续 LQ 增益的函数 lqr 调用格式有如下两种：

$[K, S, e] = \text{lqr}(A, B, Q, R)$

$[K, S, e] = \text{lqr}(A, B, Q, R, N)$

其中输入参数 A、B、Q、R、N 的含义和前面介绍 LQG 设计原理中的相应字母一致。输出 K 是得到的 LQ 增益矩阵，S 是 Riccati 方程的解，e 是闭环系统特征值  $e = \text{eig}(A - B^* K)$ 。

### (3) 卡尔曼估计器

函数 kalman 计算连续的或离散的卡尔曼估计器，调用格式如下：

$[kest, L, P] = \text{kalman}(\text{sys}, Q, R, N)$

$[kest, L, P] = \text{kalman}(\text{sys}, Q, R, N, \text{sensors}, \text{known})$

其中，Q、R、N 就是在 LQG 设计原理中提到的方程系数，参数 sensors 和 known 是向量，它们指定可以测定的输出和已知的输入，其它输入都认为是随机的。

返回值  $kest$  就是设计出的卡尔曼估计器，L 是卡尔曼滤波器增益，P 为稳态误差协方差矩阵。

### (4) LQG 控制器

经过上面几个函数的运用，LQG 控制器的各部分都已经设计完成了，函数 lqgreg 只是简单地把 LQ 增益和卡尔曼估计器连接到一起，用法为：

$rlqg = \text{lqgreg}(kest, k)$

其中，kest 为卡尔曼估计器，k 是 LQ 增益矩阵。返回值 rlqg 就是最终设计完成的 LQG 估计器（状态空间模型）。

## 10.6 根轨迹设计工具

类似于 LTI 模型分析的 GUI 工具 LTI Viewer，控制系统工具箱中也提供了控制系统的 GUI 工具：根轨迹设计 GUI (Root Locus Design GUI)。该工具用根轨迹法交互式设计 SISO 补偿器。

调用根轨迹设计工具的方法是在 MATLAB 命令窗口输入命令：rltool。其窗口如图 10-24 所示。

### 10.6.1 窗口功能

注意在窗口右上角的系统框图，图中的模块“P”为被控对象的模型，模块“K”就是

要设计的补偿器模型，窗口左上角显示着设计过程中的补偿器模型（ZPK 模型）。设计的最终目的就是为被控对象“P”设计出一个适当的补偿器“K”，使得整个系统（右上角的系统框图）的性能满足要求。

根轨迹设计 GUI 中间的窗口是进行设计的主窗口，它显示整个系统的根轨迹图（由于还没有设计补偿器，而且“F”和“H”都等于 1，因而图 10-24 中的根轨迹图就是被控对象的根轨迹）。

主窗口左上角有四个按钮，分别对应四种不同的编辑功能。

缺省时按钮 被按下，这时在主窗口中可以用鼠标交互式地移动闭环极点（根轨迹分支上的红色方块）和设计出的补偿器的零极点。

按钮 或 被按下时，可以在主窗口中添加补偿器的极点或零点。

按钮 被按下时，可以在主窗口中擦除补偿器的极点或零点。

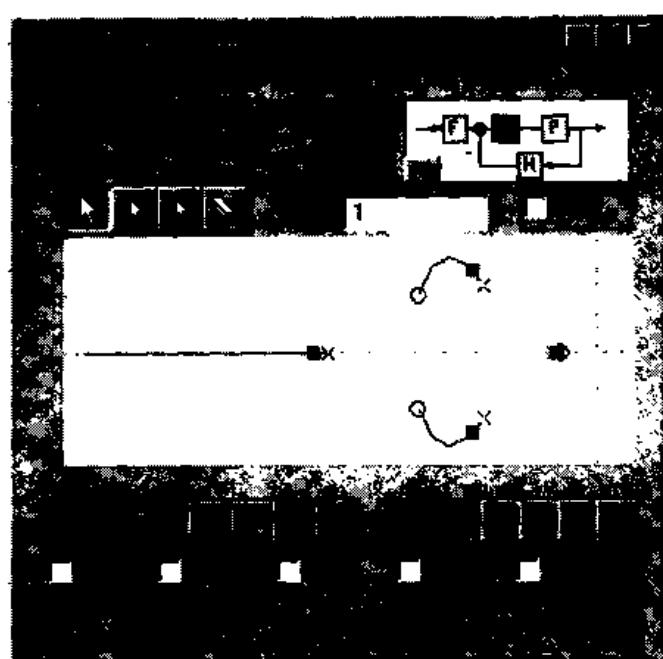


图 10-24 根轨迹设计 GUI

主窗口的左下方是坐标轴设置按钮，它们的作用分别为： 储存当前的坐标范围； 载入储存的坐标范围； 把坐标轴设置为正方形； 设置坐标轴的纵横比相等。

主窗口右下方是放大工具按钮，它们的作用分别为： 沿 X 方向和 Y 方向都放大； 沿 X 方向放大； 沿 Y 方向放大； 观察全部图形（恢复初始大小）。

根轨迹设计 GUI 下方的复选框      选择需要显示的系统时间响应和频率响应分析。选中其中的一个或几个响应后，将调用 LTI Viewer 来显示响应的曲线，随着设计过程中系统模型的变化响应也随之变化，这样便于在设计过程中随时观察系统的特性。

## 10.6.2 输入输出模型

根轨迹设计 GUI 窗口菜单 File 下的 Import Model 用来输入 LTI 设计模型，其窗口如图 10-25 所示。窗口左上角显示的是系统的结构框图，它允许两种反馈结构，另一种如图 10-24

右上角所示。在这里输入的模型包括被控对象 P, 前置滤波器 F, 传感器动态特性 H。缺省情况下, P、F、H 的值是 1。

在窗口左下角指定输入模型的位置, 包括工作空间、MAT 文件和 Simulink 框图。窗口中间显示了指定位置(如工作空间)中的所有模型名称。

如果要使被控对象 P 的模型为 sys1, 则首先选定模型 sys1, 然后在窗口右下方 P 前面的箭头上点一下, 被控对象 P 的模型就成为 sys1。同样可以为 H 和 F 指定相应的模型。系统设置完成后, 按下 OK 按钮返回主窗口, 这时主窗口中间会显示出输入的 LTI 模型系统的根轨迹图(参见图 10-24), 其中红色的小方块表示闭环极点。

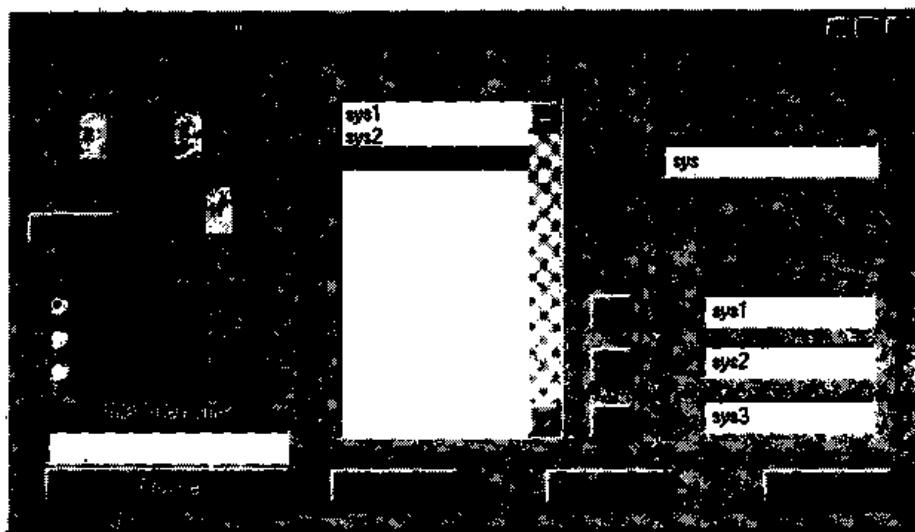


图 10-25 载入 LTI 设计模型

类似地, 菜单 File 下的 Import Compensator 选项用来输入要进行设计的补偿器 K 的模型, 模型输入后, 主窗口中显示出整个系统的根轨迹图, 在图中可以对补偿器 K 的零极点进行增删和编辑, 但不能编辑原系统的零极点, 只能沿轨迹移动闭环极点。

模型的输出用菜单 File 下的 Export 选项, 其窗口如图 10-26 所示。模型可以输出到磁盘 MAT 文件, 也可以输出到工作空间。窗口左侧显示的几项内容的含义分别为: DM 指设计的模型; OL 指开环模型; CL 指闭环模型; K 指补偿器模型。

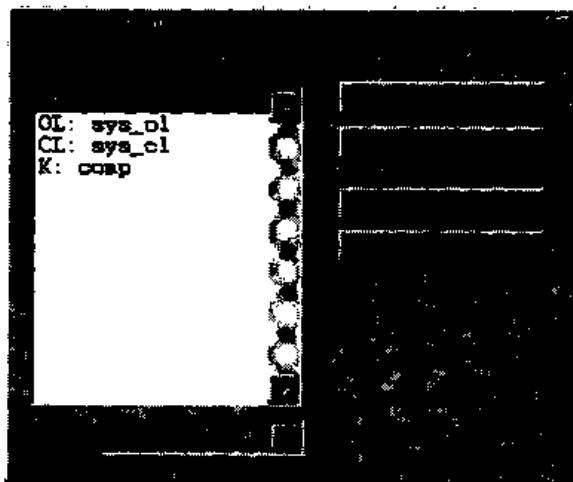


图 10-26 输出模型

### 10.6.3 设置网格和边界线

利用 Tools 菜单下的 Add Grid/Boundary 选项可以在根轨迹图中添加网格并设置边界线。其设置窗口如图 10-27 所示。

例如，我们需要显示阻尼比和固有频率的网格，则只要选中该项前的单选按钮，然后选择下面的“Grid on？”前的复选框即可。

同时，设计的系统需要满足条件：稳定时间小于 0.05 秒，阻尼比小于 0.5，那么只要在该窗口的 Settling Time 后输入 0.05，在 Damping Ratio 后输入 0.5 即可。设置的内容见图 10-27，设置后的根轨迹图如图 10-28 所示。

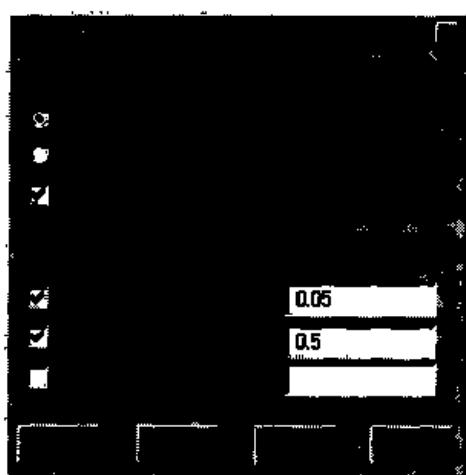


图 10-27 网格和边界线设置窗口

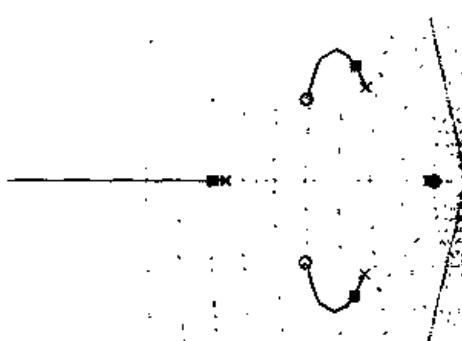


图 10-28 添加了网格和边界的根轨迹图

### 10.6.4 转换为 Simulink 框图

主窗口 File 菜单下的 Draw Simulink Diagram 选项能够把 LTI 系统模型转换为相应的 Simulink 仿真框图，这样便于在 Simulink 中对系统做更进一步的仿真分析。

由 LTI 系统模型转换得到的 Simulink 框图如图 10-29 所示。关于 Simulink 的相关内容请参见第八章和第十一章。

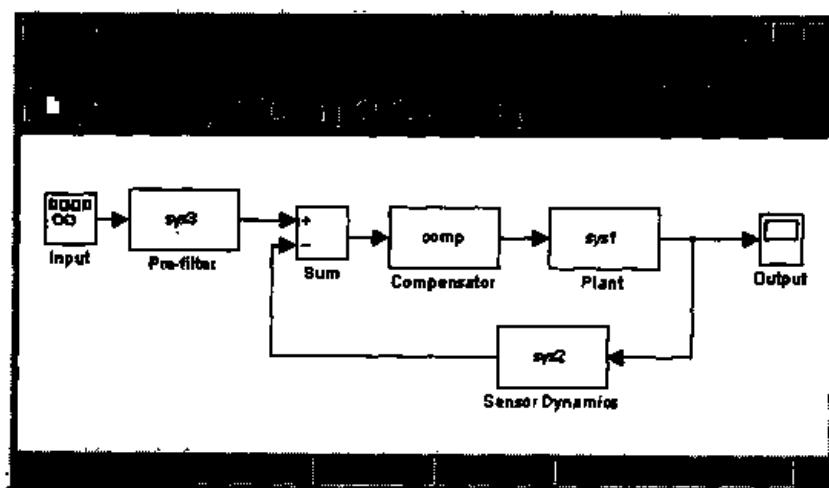


图 10-29 转换得到的 Simulink 框图

# 第 11 章 系统仿真

第 8 章介绍了如何利用 Simulink 建立仿真的框图模型，本章将更深入地介绍在 Simulink 中进行系统仿真的方法，同时介绍 Simulink 中功能强大的实时开发环境 Real-Time Workshop（实时工作间）。

## 11.1 系统仿真及参数设置

在 Simulink 中建立起系统模型框图之后，就可以用 Simulink 对模型进行动态仿真。进行仿真有两种方式：

- 1) 使用 Simulink 模型窗口的菜单 Simulation 下的选项
- 2) 在 MATLAB 命令窗口输入命令

这两种方法中以第一种方法较为常用，但第二种方法的有些优点也是第一种不能代替的。我们将在后面分别加以介绍。

运行菜单 Simulation 下的 Start 命令就可以开始仿真，仿真开始后 Start 变为 Pause，选择 Pause 可以暂停执行仿真，要停止仿真选择 Stop。

在仿真运行之前一般需要对仿真参数进行设置，这就是 Simulation 下的 Parameters 选项（当然对简单模型可以使用缺省值而不用做过多设置）。采用菜单方式进行仿真最主要的就是设置参数。

本节将主要介绍参数设置的方法。仿真参数设置包括 Solver（解法）、Workspace I/O Page（到工作空间的输入输出）、Diagnostics（诊断）和 Real-Time Workshop（实时工作间）。其中 Real-Time Workshop 我们将在后面几节详细介绍。

### 11.1.1 Solver 的设置

在 Solver 里需要设置仿真的起始和停止时间、选择合适的解法（solver）并指定参数、设置一些输出选项。该界面的内容如图 11-1 所示。

#### 1. 设置起始时间和停止时间

设置起始和停止时间就是在 Start time 和 Stop time 后的编辑框内输入相应的数值。它们的单位是“秒”，但实际运行时间和计算机的性能、模型的复杂程度、解法、步长、要求的误差水平等很多因素有关，因而一般和时钟的秒不一致。

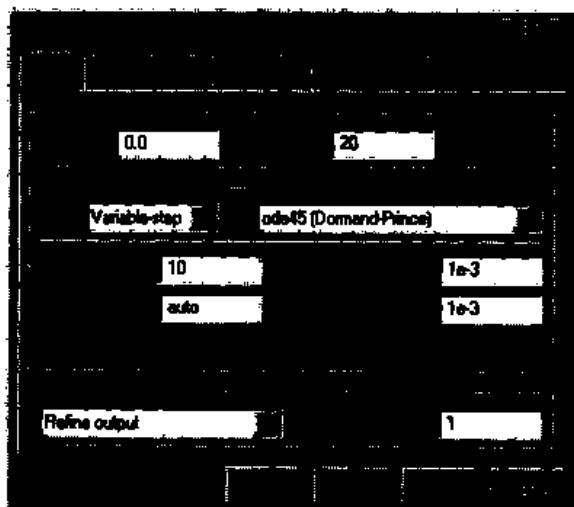


图 11-1 Solver 设置

## 2. 设置仿真步长

仿真的主要过程一般是求解常微分方程组。窗口中 Solver options 区域的内容就是针对解常微分方程组的。

其中的 Type 用来选择仿真的步长是变化的还是固定的。变步长解法可以在仿真过程中根据要求调整运算步长，在采用变步长解法时，应该先指定一个容许误差限（在 Relative tolerance 或 Absolute tolerance 中设置），使得当误差超过这个误差限时自动修正仿真步长，所以说，在变步长仿真时误差限的设置将关系到常微分方程组解的精度。同时，采用变步长解法还要设置最大步长（Max step size）。在缺省情况下（输入“auto”），系统按下式决定最大步长：

$$\text{最大步长} = (\text{停止时间} - \text{起始时间}) / 50$$

通常这个缺省值就够了。如果仿真时间很长，缺省最大步长就很大，有可能会出现失根，这时可以根据需要设置较小的步长。如果模型具有周期或近似周期的特性，而且已知周期的大小，则可以设置最大步长的值为该周期的一部分（比如 1/3 周期）。

在一些特定的场合下往往还需要采用定步长的方法进行仿真，这时不能对误差限做出要求，只能设置固定的步长长度。选取定步长解法后，出现了一个选项：Mode（模型类型）。它包括 MultiTasking（多任务）、SingleTasking（单一任务）和 Auto。多任务模型指模型中的有些模块具有不同的采样速率，并对模块之间采样速率的传递进行检测；单一任务模型的各个模块采样速率是相同的，不检测采样速率的传递；Auto 则根据模型中模块的采样速率是否相同决定采用单一任务模型还是多任务模型。

## 3. 选择仿真解法

有了模型以后，确定一个合适的解法是至关重要的。

如果模型全部是离散的，那么对变步长和定步长，解法都采用 discrete (no continuous state)。

如果模型含有连续状态，则可供选择的解法很多，而且对于变步长和定步长的情况供选择的解法是不同的。

可供选择的变步长解法有：ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb 和 discrete (variable-step)。这些解法的具体内容参见第 6 章。

可以选用的定步长解法有：ode5, ode4, ode3, ode2, ode1 和 discrete (fixed-step)。分别为：

- ode5：定步长的 ode45 解法。
- ode4：四阶 Runge-Kutta 算法 (RK4)。
- ode3：定步长 ode23 解法。
- ode2：Huen 方法，即改进的欧拉法。
- ode1：即欧拉法。
- discrete (fixed-step)：不含积分运算的定步长方法，适用于没有连续状态的系统。

## 4. 设置输出选项 (Output Options)

对同样的信号，选择不同的输出选项后，到输出设备上的信号是不完全一样的。要根据需要选择合适的输出选项以达到满意的输出效果。

输出选项包括三种：Refine output (细化输出)，Produce additional output (产生附加输

出), Produce specified output only (只产生指定输出)。下面分别加以介绍。

(1) 细化输出: 可以增加输出数据的点数, 使输出数据更加平滑。例如, 如果细化系数 (Refine factor) 定为 2, 则会在每段时间步长的中间插入一个平滑数据; 如果是 3, 则会在每段时间步长的  $1/3$ 、 $2/3$  位置各插入一个数据。细化系数越大, 输出数据越平滑。缺省的细化系数为 1。细化输出适用于变步长解法, 尤其是常采用大步长的 `ode45` 法。总之, 如果输出效果粗糙, 最简单有效的办法就是增加细化系数。

(2) 产生附加输出: 允许指定产生输出的附加时刻。选择这一项后, 出现输出时刻 (Output times) 编辑框, 在这里可以输入计算附加时刻的表达式或附加时刻向量。和细化输出不同的是, 这种方式改变仿真步长以使其和指定的附加输出的时刻相一致。

(3) 只产生指定输出: 只在指定输出的时刻产生仿真输出。这种方式改变仿真步长以使其和指定的产生输出的时刻相一致。有的情况下需要对不同的仿真做比较, 以确定它们同时产生输出, 这时就要采用这种方式。

### 11.1.2 设置 Workspace I/O Page

如图 11-2 所示, 在这一部分可以设置 Simulink 和当前工作空间 (workspace) 的数据输入输出。通过设置, 可以从工作空间输入数据、初始化状态模块 (state), 也可以把仿真结果、状态模块数据、时间数据保存到当前工作空间。输入输出的数据格式可以是矩阵、结构 (structure)、包含时间数据的结构 (structure with time)。

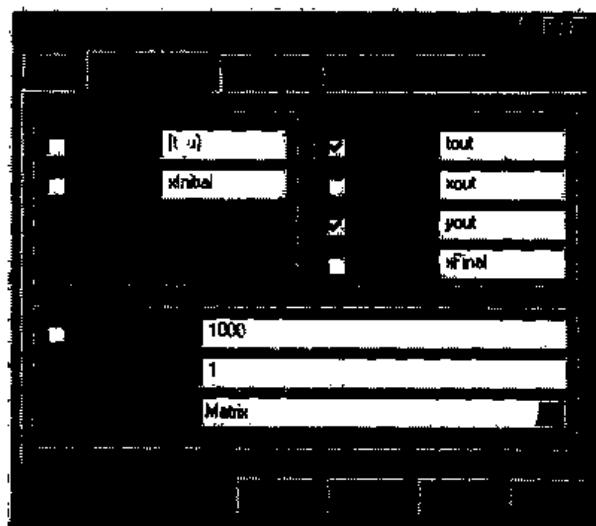


图 11-2 设置 Workspace I/O Page

下面我们将对 Simulink 和 MATLAB 工作空间之间输入输出数据、状态的设置方法分别给以详细说明。

#### 1. 从工作空间载入数据

在仿真过程中可以从工作空间载入数据到模型的输入端口, 但前提是模型必须有输入端口, 即信号与系统模块库中的模块 In1。

设置的方法是选中 Input 前的复选框, 然后在后面的编辑框键入输入数据的变量名。输入数据可以有几种格式: 结构、矩阵、包含时间数据的结构。各种格式都有比较严格的要求。

### (1) 矩阵

矩阵的列数要求等于模型输入端口的个数加1，Simulink自动把第一列当成时间向量，后面几列按顺序依次对应各个端口。例如，对于有两个输入端 In1、In2 的模型，输入矩阵就必须是三列。如果用缺省的表示方式：[t, u]，那么 t 应当是一维时间列向量，u 应是和 t 长度相同的二维列向量。下面的 t 和 u 是合理的：

```
t=(1:1:10)';
u=[t.^2, t.^3];
```

### (2) 包含时间数据的结构

对于包含时间数据的结构，Simulink 有更加具体的规定。首先结构必须有两个顶级域：time 和 signals。顶级域 time 包含一个列向量，表示仿真时间。顶级域 signals 包含一个数组，数组的每个元素都是一个子结构，每个子结构对应模型的一个输入端口，每个子结构下必须包含域 values，values 域包含一个列向量，就是相应于该子结构的输入端口的输入数据。

注意：结构的域名是固定的：“time”、“signals”、“values”，不能更改。

对结构的赋值应该分别按照结构的各个域进行。例如，对矩阵输入的数据，现在要用结构输入，则应按下述过程进行：

```
t = (1:1:10)';
a.time = t;
a.signals(1).values = t.^2;
a.signals(2).values = t.^3;
```

在这里我们令结构名为 a，同样，在参数设置窗口 Input 后的编辑框中就应该输入 a。结构的名称是任意的。

### (3) 结构

对于一般结构，就是使包含时间数据的结构的 time 域为空（注意不能去掉 time 域）。对于上面的例子，就是把

```
a.time=t;
```

改为

```
a.time=[ ];
```

这样，Simulink 根据输入端口（In1, In2 等）的参数中设置的采样时间读取输入数据。因而要注意，In1, In2 等的参数设置对话框中的采样时间（Sample Time）必须输入确定的值，不能用“-1”来选用动态方式；同时，内插数据（Interpolate data）前的复选框一定要去掉。

还有一种综合方法，可以把包含时间数据的结构和不含时间数据的结构输入不同的端口，每个输入端口对应一个结构。这实际上是前两种结构的组合，只需要分别给不同的结构赋值，并在参数设置窗口中 Input 后的编辑框中输入相应的结构名，中间用逗号分隔开。用这种方法要注意，结构的顶级域 signals 的数组中只能包含一个元素。仍然用上面的例子，这时应该如下赋值：

```
t=(1:1:10)';
a.time=t;
a.signals.values=t.^2;
```

```
b.time=[ ];
b.signals.values=t.^3;
```

然后在参数设置窗口 Input 后的编辑框中输入“a, b”。

## 2. 保存数据到工作空间

在参数设置的 Save to workspace 区域，可以选择保存的选项有：时间（Time）、端口输出（Output）、状态（States）、最终状态（Final state）。选中选项前面的复选框并在选项后面的编辑框键入变量名，就会把相应数据保存到指定的变量。

保存到工作空间的数据也有矩阵、包含时间数据的结构、一般结构这几种格式，在 Save options 区域中的 Format 下可以选择需要的格式。选择不同的数据格式，Time 的格式是不变的，总是仿真的采样时间。下面介绍一下其它几个选项的变化。

使用矩阵格式时：States 输出的矩阵（缺省是 xout）每一列对应于模型的一个状态模块，每一行对应于一个确定时刻的状态；Final state 输出的矩阵（缺省是 xFinal）只有一行，表示最终状态，每一列对应于模型的一个状态模块；Output 输出的矩阵（缺省是 yout）每一列对应于模型的一个状态模块的输出端口，每一行对应于一个确定时刻的输出。需要注意，Output 输出的矩阵是到输出端口（Outport 模块）的值，因而模型中需要有 Outport 模块，否则不能生成输出矩阵。

如图 11-3 所示的系统，由于只有一个输出端口（Out1），因而 Output 输出的矩阵是一列，而 Transfer Fcn 和 Discrete-Time Integrator 两个模块都属于状态模块（注意 Zero-Order Hold 模块不是状态模块），因而 States 和 Final state 输出的矩阵都是两列。

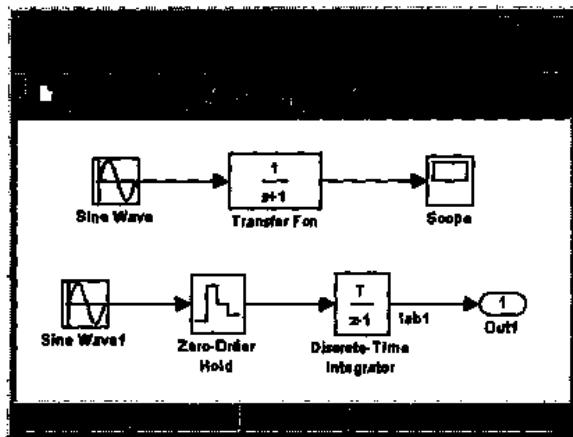


图 11-3 保存数据到工作空间

对于包含时间数据的结构：结构有两个顶级域：time 和 signals。顶级域 time 为一个列向量，对应于仿真时间。顶级域 signals 下包含一个子结构数组，对于 States 和 Final state 选项，每个子结构对应一个状态模块；而 Output 选项的子结构个数和输出端口的个数相同，每个子结构对应一个输出端口。虽然这些项目的子结构的含义不同，但它们都含有三个域：values、label、blockName。域 values 是一个向量，表示相应的数据（对 Output 选项是输出端口的数据，对 States 选项是状态模块的数据，对 Final state 选项是状态模块的最终数据）。域 label 是一个字符串，对 Output 选项是连线上的信号标记；对 States 和 Final state 选项 label 域的内容是 Cstate（连续系统模块）或 Dstate\_n（离散系统模块），n 表示响应模块中有 n

组离散状态；域 `blockName` 是一个字符串，表示相应模块的名称。

对于不包含时间数据的结构，和包含时间数据的结构区别仅仅是时间域为空。

和载入数据类似，输出数据到工作空间也有综合方法，可以把来自每个输出端口（Outport）包含时间数据的结构和不含时间数据的结构一起输出，这时结构的顶级域 `signals` 的数组中只包含一个元素。这实际上是前两种结构的组合，只需要在参数设置窗口中 Output 后的编辑框中输入相应的结构名，中间用逗号分隔开。

为了对每种数据输出的结构有一个形象具体的认识，我们把图 11-3 中的模型输出到工作空间的选项 States 和 Output 的数据结构列出来，便于分析。假定输出变量名都用缺省值，选项 States 输出为 `xout`，选项 Output 输出为 `yout`：

```
xout = % xout 有两个顶级域  
        time: [201x1 double]  
        signals: [1x2 struct]  
  
yout =  
        time: [201x1 double]  
        signals: [1x1 struct]  
  
xout.signals = % xout 的顶级域 signals 是 1x2 的结构数组，它又包含三个域  
1x2 struct array with fields:  
    values  
    label  
    blockName  
  
xout.signals.label % 顶级域 signals 下的 label 域，是 1x2 的数组  
  
ans =  
CState  
  
ans =  
DState_1  
  
xout.signals.blockName % 顶级域 signals 下的 blockName 域，1x2 的数组  
  
ans =  
savedata/Transfer Fcn  
  
ans =  
savedata/Discrete-Time  
  
Integrator
```

在参数设置对话框 Save options 下还有两个选项：Limit row to last 和 Decimation。Limit row to last 用来限定保存到工作空间数据的最大行数。Decimation（抽选）指从几个数中选择一个。例如，如果在 Decimation 的编辑框中键入 3，则表示输出数据时，每三个数据中取一个，也就是隔两个数取一个数。

### 3. 初始化状态模块

状态模块初始化的方法有两种：使用模块本身的参数设置对话框；从工作空间载入。有的状态模块本身的参数设置对话框中含有设置初始值的选项，例如 Integrator 模块，这时可以直接在对话框中输入初始值。从工作空间载入数据对模块进行初始化适用于所有状态模块，方法是选中图 11-16 中 Load from workspace 下的 Initial state 选项前的复选框，在其后的编辑框中键入含有初始化数据的变量名，变量类型可以是矩阵、带时间数据的结构、一般结构，内容和前面讲到的基本相同，不再赘述。只是要注意，用于初始化的变量中的元素个数要和状态模块数目一致。

以图 11-3 中的系统为例，首先在 Workspace I/O Page 中选择 Initial state 前的复选框，然后在其后的编辑框中键入变量名“init”，系统含有两个状态模块 Transfer Fcn 和 Discrete-Time Integrator，因而变量 init 应含有两个元素，在工作空间中给 init 赋值语句应该是如下形式：

`init = [10 1] 或 init = [10 1]'`

当从工作空间载入数据时，模块本身的参数设置对话框中的初始值无效。另外，我们还注意到，为 init 赋值为行向量或列向量均可，init 中的元素按状态模块排列的先后顺序给模块赋值。

#### 11.1.3 Diagnostics (诊断) 参数设置

如图 11-4 所示，在这部分可以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 None (不做反应)、Warning (警告)、Error (提示错误)，警告信息不影响程序的执行，而提示错误后程序就要停止运行。大家可以根据自己程序的调试需要做出相应的选择。

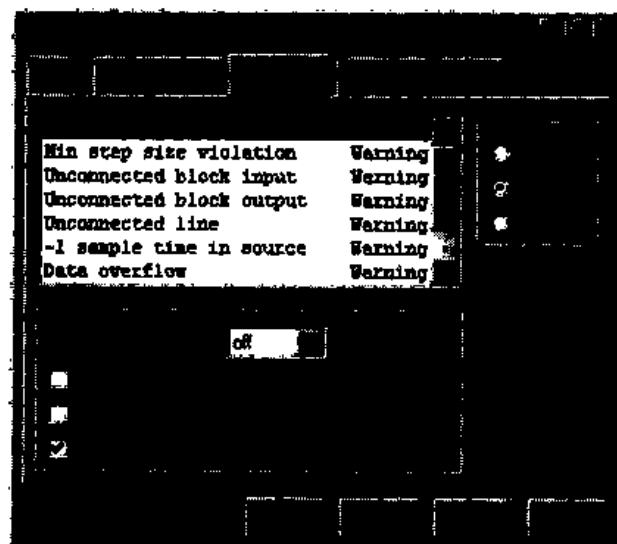


图 11-4 设置 Diagnostics 参数

Options 区域下的 Consistency checking (一致性检验) 让用户选择是否对模型进行一致性检验。一致性检验是一个调试工具，用以验证 Simulink 微分方程解法中的某些假设。它主要用于保证 S 函数和 Simulink 内置模块具有相同的规则。一致性检验的另一个目的是确保模块对于固定的时间  $t$  输出是不变的。这一点对于刚性问题的解是很重要的，因为当计算 Jacobian 矩阵时，在相同的时间  $t$ ，输出函数会被多次调用。

由于进行一致性检验会使系统性能明显降低，因此这一项一般设为 off。

Zero crossing detection (零交叉检验) 可以提高仿真精度，但仿真速度会有所下降。选项 Disable zero crossing detection 使系统不对自身具有零交叉检验功能的模块进行检验。

Optimized I/O storage (优化 I/O 存储) 指对模块的 I/O 数据优化使用内存缓冲区，这样有些数据的缓冲区会重复使用以节省内存空间。若选取 Disable optimized I/O storage，则表示给每个模块的 I/O 数据分配单独的内存缓冲区。这样会增加内存的使用量，对于大的模型，增加的量将很大，因而一般只在需要对模型进行调试的时候才选择这一项。

最后一项 Relax boolean type checking (2.x compatible) 是指放宽对逻辑数据类型的检查，以和 Simulink 3.0 以前的版本兼容。当选取这一项时，系统会允许对 double 类型的数据进行逻辑操作，也就是可以把 double 类型的数据作为逻辑类型处理。如果不选这一项，系统会对这类操作提示错误信息。

#### 11.1.4 在命令窗口输入命令进行仿真

在前几节中我们介绍了使用菜单命令进行仿真和参数设置的方法，可以看出它主要有以下几个优点：

- 可以直接设置和修改很多仿真参数，包括仿真时间、解法、仿真步长等等
- 可以同时对几个系统进行仿真
- 可以实时监视连线上传输的数据
- 可以方便地修改各个模块的参数

而本节将要介绍的方法是在命令窗口输入命令进行仿真，它和菜单命令相比不够简便、直观，但在下述两个方面，它是菜单命令不能替代的：

- 仿真 M 文件、MEX 文件（用 C 或 Fortran 编写的可以被 MATLAB 调用或执行的动态连接子程序）的模型
- 从 M 文件中运行仿真，允许仿真和模块参数反复改变，或者在设定的条件下开始或停止仿真

从命令窗口运行仿真的命令有 sim、simset、simget 和 set\_param。

##### 1. sim 命令

该命令完整的调用格式是：

`[t, x, y] = sim(model, timespan, options, ut);`

其中各个参数的意义如下：

`t`: 仿真的时间向量。

`x`: 状态模块的状态矩阵。

`y`: 仿真的输出矩阵，每一列对应一个输出端口 (Outport) 的输出数据。

`model`: 模型的名字，用单引号括起来。

**timespan:** 时间带宽。有三种格式：

[tFinal]指定仿真停止时间，仿真开始时间缺省为 0；

[tStart tFinal]指定仿真开始和停止时间；

[tStart OutputTimes tFinal]指定开始时间、停止时间和要输出的时间点。

**options:** 由 simset 命令（后面将介绍）设置的仿真参数，数据格式为结构。

**ut:** 外部输入到输入端口 (Inport) 的数据。如果输入到所有端口，则 ut 可以是矩阵或结构；如果是输入到单个端口，则 ut 只能是结构。

这些参数中只有 model 是必须的，当其它参数没有赋值时，系统会自动以 Simulink Parameters 中设置的参数值为准。

例如，对于仿真模型“Simul1”，如图 11-5 所示，我们来比较一下用不同的参数运行 sim 命令的含义：

(1) 在命令窗口输入：

[t, x, y] = sim('Simul1')

这时在 MATLAB 命令窗口用 whos 命令来查看工作空间中的变量：

**whos**

Name	Size	Bytes	Class
t	205x1	1640	double array
x	205x3	4920	double array
xFinal	1x3	24	double array
y	205x2	3280	double array

状态矩阵 x 有三列表示模型 Simul1 中含有三个状态向量，模块 Integrator 含有一个状态向量，而 Transfer Fcn 含有两个状态向量，这是因为它是二阶传递函数。而输出矩阵 y 有两列，对应着输出端口 Out1 和 Out2 的值。

(2) 指定仿真开始时间、停止时间和中间输出数据的时刻：

[t, x, y] = sim('Simul1', [2, 8]) % 时间范围为 2 到 8 秒

[t, x, y] = sim('Simul1', [2, 4, 6, 8])

这时仿真时间向量 t 大小为 4\*1，x 为 4\*3，y 为 4\*2。这种情况下只在四个指定的时刻（包括开始和停止时间）有值。这种方式便于对某些特定的时刻作分析。

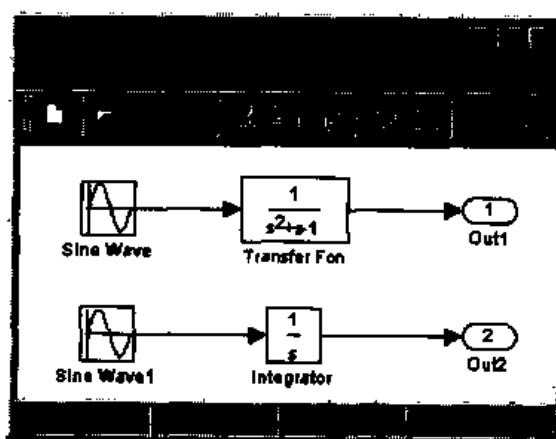


图 11-5 仿真模型 Simul1

## 2. simset 命令

simset 命令用来为 sim 命令建立或编辑仿真参数，并把设置结果保存在一个结构变量中。它有如下四种用法：

(1) options = simset(property, value, ...)

把“property”的参数赋值为“value”。结果保存在 options 中。

(2) options = simset(old\_opstruct, property, value, ...)

把已有的结构“old\_opstruct”(由 simset 产生)中的参数“property”重新赋值为“value”。结果保存在新结构 options 中。

(3) options = simset(old\_opstruct, new\_opstruct)

用结构“new\_opstruct”的值替代已经存在的结构“old\_opstruct”的值。

(4) simset

显示所有的参数名和它们可能的值。

可以用 simset 命令设置的参数“property”和可能的取值“value”如下所示，其中由“{}”括起来的是缺省值：

(1) Solver: 'VariableStepDiscrete'|

'ode45' | 'ode23' | 'ode113' | 'ode15s' | 'ode23s'

'FixedStepDiscrete'|

'ode5' | 'ode4' | 'ode3' | 'ode2' | 'ode1'

指定仿真解法，取值为字符串。

(2) RelTol: [正数 {1e-3}]

相对误差容限 (Relative error tolerance)，缺省值 1e-3。

(3) AbsTol: [正数 {1e-6}]

绝对误差容限 (Absolute error tolerance)，缺省值 1e-6。

(4) Refine: [正整数 {1}]

输出细化因子，缺省值 1。

(5) MaxStep: [正数 {auto}]

最大步长，只对应变步长解法，缺省值 auto。

(6) InitialStep: [正数 {auto}]

初始步长，只对应变步长解法，缺省值 auto。

(7) MaxOrder: [1|2|3|4|{5}]

解法 ODE15s 的最大阶数，只对应解法 ODE15s，缺省值 5。

(8) FixedStep: [正数]

固定步长，只对应变定步长解法，如果模型中有离散状态，缺省值为基本采样时间，否则缺省值为仿真时间间隔的 1/50。

(9) OutputPoints: [{specified} | 'all']

确定输出数据点，取值为字符串格式，指定缺省值 ‘specified’ 时，解法只产生在 timespan 中指定的时间点的值，指定 ‘all’ 时，输出还会包括解法的时间步长对应的值。

(10) OutputVariables: [{txy} | 'tx' | 'ty' | 'xy' | 't' | 'x' | 'y']

设置输出变量，取值为字符串格式。

(11) SaveFormat: [ {'Matrix'} | 'Structure' | 'StructureWithTime' ]

设置保存的状态和输出的数据格式，取值为字符串格式，缺省值是 ‘Matrix’。

(12) MaxRows: [ 非负整数 {0} ]

限定输出的最大行数，缺省值为 0。

(13) Decimation: [ 正整数 {1} ]

设置抽选系数，即在几个数中抽取一个，缺省值为 1。

(14) InitialState: [ 向量 {[ ]} ]

确定连续或离散模块的初始状态值，缺省为空向量。

(15) FinalStateName: [ 字符串 {""} ]

确定一个变量名用来在仿真结束后保存状态模块的状态值，取值为字符串格式，缺省为空。

(16) Trace: [ comma separated list of 'minstep', 'siminfo', 'compile' {""} ]

Trace 包括 ‘minstep’、‘siminfo’、‘compile’ 中的某一个或者用逗号分隔的几个选项。其中 ‘minstep’ 指当变步长解法的某一步不能满足误差容限时就停止仿真并提示错误；‘siminfo’ 指的是在仿真开始时对仿真参数给出一个简略的总结；‘compile’ 是指显示对模块框图的编译过程。

(17) SrcWorkspace: [ {'base'} | 'current' | 'parent' ]

指定在哪个工作空间计算模型中定义的 MATLAB 表达式的值，取值为字符串格式，缺省值为 ‘base’（基本工作空间，指 MATLAB 命令窗口）。

(18) DstWorkspace: [ 'base' | {'current'} | 'parent' ]

指定在哪个工作空间为模型中定义的变量赋值，取值为字符串格式，缺省值为 ‘current’（当前工作空间）。

(19) ZeroCross: [ {'on'} | 'off' ]

允许或不允许零交叉检验，零交叉检验只对应于变步长解法，取值为字符串格式，缺省值为 ‘on’。

下面我们来举例说明 simset 的用法。仍然以图 11-5 所示的仿真模型为例，在命令窗口输入如下语句：

```
option1 = simset('OutputVariables', 'xy', 'OutputPoints', 'all');
```

```
[t, x, y] = sim('Simul1', [2,4,6,8], option1);
```

这时，由于设置的 OutputVariables 中不包含 t，因而 [t, x, y] 中的 t 为空；而 OutputPoints 设为 all 则使得输出中除去指定的几个点，还包括了仿真时间步长对应的点。

式中 option1 的值为：

```
option1 =
```

```
.....
```

```
OutputPoints: 'all'
```

```
OutputVariables: 'xy'
```

```
.....
```

### 3. simget 命令

simget 命令用来获得模型的参数设置值。它有如下三种用法：

(1) `struct = simget(model);`

返回指定模型 `model` 的参数设置的 `options` 结构。

(2) `value = simget(model, property);`

返回指定模型 `model` 的参数 `property` 的值。

(3) `value = simget(OptionStructure, property);`

获取参数设置的 `options` 结构 `OptionStructure` 中的参数 `property` 的值。

#### 4. `set_param` 命令

`set_param` 命令的功能很多，这里只介绍如何用 `set_param` 命令设置 Simulink 仿真参数以及如何开始、暂停、终止仿真进程。

##### (1) 设置仿真参数

语法如下式所示：

```
set_param(mode, property, value, ...);
```

式中 `mode` 为设置的模型名，`property` 为要设置的参数，`value` 是设置值。这里设置的参数可以有很多种，而且和用 `simset` 设置的内容不尽相同，相关参数见附录 C，这里不再一一介绍。

##### (2) 控制仿真进程

语法如下式所示：

```
set_param(mode, 'SimulationCommand', 'cmd')
```

式中 `mode` 为仿真模型名称，而'cmd'就是控制仿真进程的各个命令，包括'start', 'stop', 'pause', 'continue'或者'update'。

在用这两个命令的时候，需要注意必须先把模型打开。仍以前面提到的模型 `Simul1` 为例，我们首先把模型打开，然后运行如下命令：

```
set_param('Simul1', 'StartTime', '10', 'StopTime', '20')
```

```
set_param('Simul1', 'SimulationCommand', 'start')
```

这时模型就会开始仿真。

这一节我们介绍了系统仿真的两种方式：菜单命令的方式简便直观，易于使用，但不够灵活；从命令窗口输入命令的方式比较复杂，但它可以很灵活地用于 M 文件或 MEX 文件中，这样就可以编制结构比较复杂的程序，仿真过程可以在各种流程中调用。

## 11.2 子系统的建立和封装

当建立的模型比较大，或者模型中的一些模块的功能可以组合为相对完整的功能时，可以把模型中的这些模块组合在一起，成为一个新的模块，这就是子系统（Subsystem）。

### 11.2.1 子系统的建立

子系统类似于一般编程语言中的子函数，它们也具有类似的优点，一方面有助于减少模型中的模块数目，使模型的结构和层次更加清晰，可读性更强，也易于调试和维护；另一方面把功能上相关的模块组合到一起，形成了比较完善的功能，在其它模型中还可以直接作为一个功能模块调用。

子系统的建立有两种方法：

### 1. 通过在模型中添加 Subsystem 模块建立子系统

如果需要在模型中新建一个子系统，模型本身不包含组成子系统的模块，则可以按下列的步骤进行：

(1) 在 Simulink Library Browser 中打开 Simulink 库，从其中的 Signals & Systems 库中拷贝 Subsystem 模块到模型中。

(2) 双击 Subsystem 模块打开 Subsystem 窗口。

(3) 把要组合的模块拷贝到 Subsystem 窗口，然后在该窗口中加入 Import 模块表示从子系统外部到内部的输入，加入 Outport 模块表示从子系统内部到外部的输出，把这些模块按顺序连接起来，子系统就建立成功了。

例如，我们要把逻辑模块 OR 和 NOT 的连接组合为一个子系统，组合后的子系统模型将象图 11-6 这样，该子系统的图标位于图 11-6 的右侧。

我们看到，在建成的子系统的图标中标出了该子系统的输入输出端口，这样便于和其它模块连接。如果不标出端口，可以选定子系统模块，然后在菜单 Format 下选择 Hide Port Labels。另外，对一般模块的常规操作，同样适用于子系统模块。

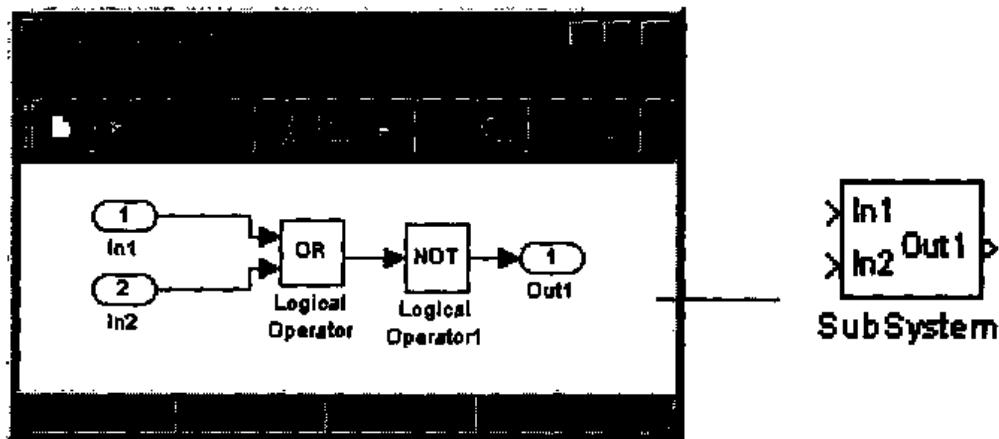


图 11-6 子系统模型及其图标

### 2. 通过组合已有的模块建立子系统

如果是把模型中已经存在的模块组合为一个子系统，那么操作就更加简单。

第一步：同时选取所有要组合的模块。

第二步：在 Edit 菜单下选择 Create Subsystem 或按快捷键 Ctrl+g，子系统就建成了。

这种情况下，Simulink 自动把 Import 模块和 Outport 模块添加到子系统中，并把原来的模块替换为建成的子系统图标。

## 11.2.2 条件执行子系统

条件执行子系统指的是子系统的执行由输入信号的值来控制。控制子系统执行的信号称为控制信号。在一个复杂模型中，有的模块的执行依赖于其它模块，这种情况下，条件执行子系统是很有用的。

条件执行子系统包括使能子系统、触发子系统和使能加触发子系统。

### 1. 使能子系统 (enabled subsystem)

使能子系统在控制信号从负数朝正方向穿过零时开始执行，直到控制信号变为负数为止。使能子系统的控制信号可以是标量也可以是向量。如果控制信号是标量，当该标量的值大于 0 时子系统执行；如果是向量，向量中的任何一个元素大于 0，子系统都执行。

建立使能子系统的方法是从 Signals & Systems 库中拷贝 Enable 模块到该子系统。图 11-7 中的子系统就是一个使能子系统。使能子系统在图标中加上了“几”字型的标志，并在该标志处增加了控制信号输入端口，见图 11-8 所示子系统在原模型中的图标。

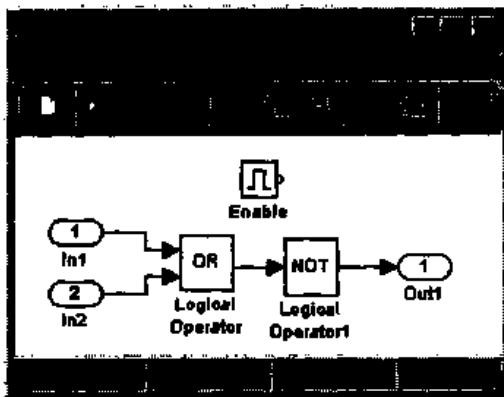


图 11-7 使能子系统

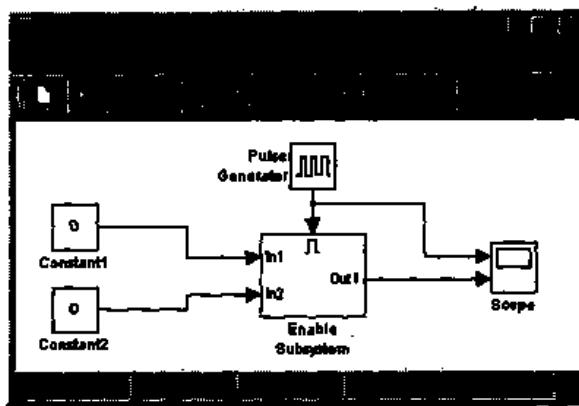


图 11-8 使能子系统所在的模型

使能子系统建立后，一般还需要对其中的 Enable 模块和输出端口模块 (Outport) 进行参数设置。我们以图 11-7 所示的子系统（其所在模型如图 11-8 所示）为例，讲一下如何设置这些参数才能达到满意的效果。

双击 Enable 模块打开其参数设置对话框，如图 11-9。在参数 States when enabling 后有两个选项：held 和 reset。held 表示当使能子系统再次开始执行时，保持上一次执行后子系统的状态；reset 表示使能子系统再次执行时，把子系统状态重置为初始状态。本例中设为 held。

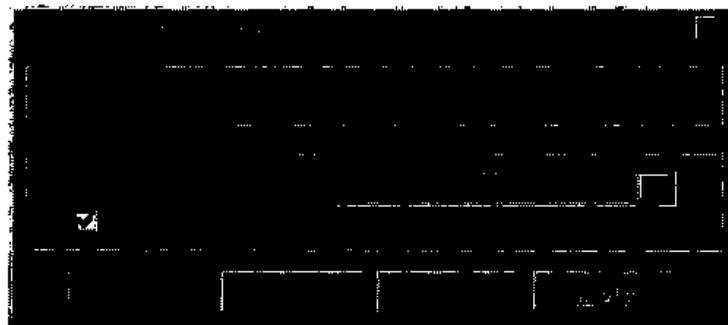


图 11-9 Enable 模块参数设置

在这个对话框中还有一个复选框 Show output port，选中该项后，Enable 模块会添加一个输出端，从这个输出端输出控制信号，从而可以对控制信号进行监视和分析。

打开输出端口模块 Out1 的参数设置对话框，如图 11-10。在参数 Output when disabled 后同样有两个选项：held 和 reset。这里 held 表示当使能子系统停止执行后，输出端口的值保持该端口最近的输出值；reset 表示当使能子系统停止执行后，输出端口的值重置为初始输出值，初始输出值在 Initial output 编辑框中设置。本例中把参数 Output when disabled 设

为 reset，初始输出值设为 0。

上述两个参数设置好后，我们来观察子系统的输出波形，如图 11-11。上面的波形是输入到 Enable 模块的脉冲控制信号，下面的波形是子系统的输出信号。当脉冲信号为“1”时，子系统执行，输出逻辑运算结果“1”，当脉冲信号下降为“0”时，子系统不执行，根据对子系统的输出端口 Out1 的设置（reset），输出值重置为“0”，因此输出信号和控制脉冲的波形是同步的。

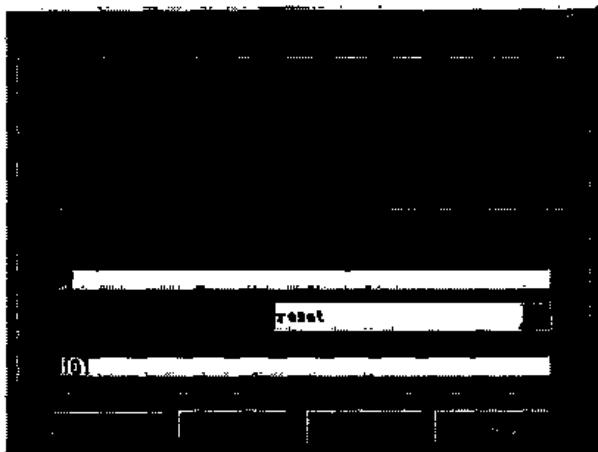


图 11-10 输出端口参数设置

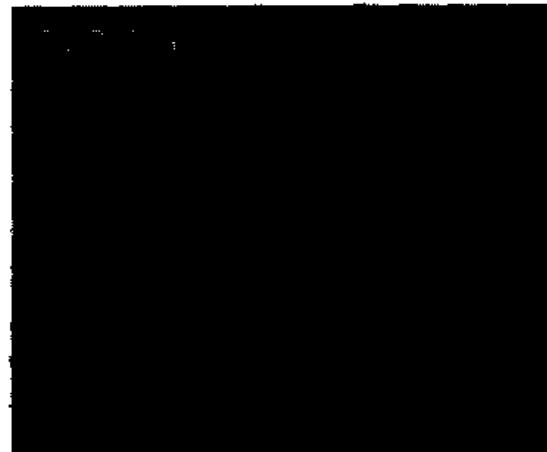


图 11-11 使能子系统的输出波形

如果我们把参数 Output when disabled 设为 held，那么子系统的输出波形将是值为 1 的一条直线，读者可以自己分析其中的原因。

## 2. 触发子系统 (triggered subsystem)

触发子系统指的是当触发事件发生时开始执行的子系统。建立一个触发子系统就是把 Signals & Systems 模块库中的 Trigger 模块拷贝到子系统框图中。如图 11-12，就是一个触发子系统。

触发子系统有一个控制输入端，控制信号从这里输入子系统，当触发事件发生时子系统开始执行。触发事件发生的条件可以设为三种类型：

- 上升沿（rising）触发：控制信号从负值或 0 上升到正值时触发，子系统开始执行。
- 下降沿（falling）触发：控制信号从正值或 0 下降到负值时触发，子系统开始执行。
- 上升沿和下降沿均触发：当控制信号满足上升沿触发或下降沿触发的条件时均触发，子系统开始执行。

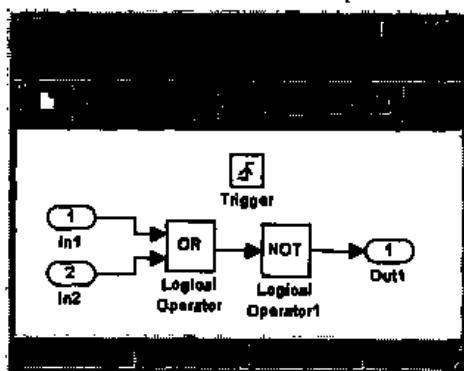


图 11-12 触发子系统

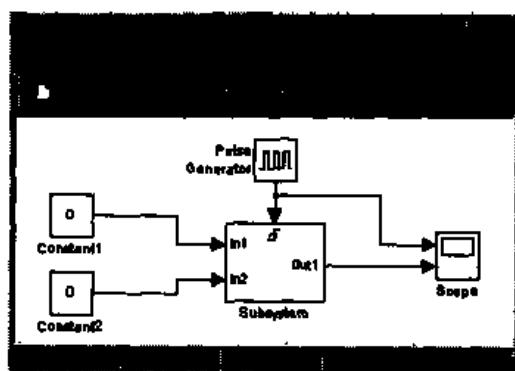


图 11-13 触发子系统所在模型

触发类型在 Trigger 模块的参数设置对话框中进行设置, 如图 11-14。在参数 Trigger Type 后面有四个选项, 它们分别对应几种触发类型: rising 是上升沿触发, falling 是下降沿触发, either 是上升沿和下降沿均触发, function-call 是一种特殊的类型, 这种子系统的触发不是由信号的值决定的, 而是由 S 函数内部的逻辑决定。

对于不同的触发类型, 子系统模块的图标有不同的标志, 图 11-15 列出了这几类不同的子系统。

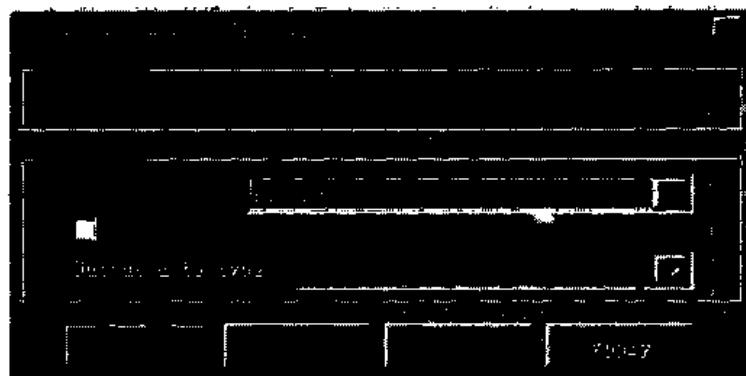


图 11-14 Trigger 模块参数设置

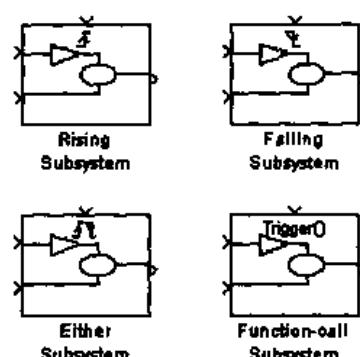


图 11-15 不同触发类型的子系统

Trigger 模块的参数设置对话框中还包括 Show output port 复选框, 选中这一项后, Trigger 模块添加一个输出端, 用于输出控制信号。选中这个复选框后, 下面的参数 Output data type 被激活, 在这里选取输出控制信号的类型, 其中包括 auto、int8 和 double 三种类型, auto 指输出的数据类型自动和输入保持一致。

对于两次触发之间子系统的状态和输出的设置, 触发子系统和使能子系统不同。触发子系统在每次触发结束到下次触发发生之间总是保持 (hold) 上一次结束时的输出数据, 触发再次开始时也不能重置子系统的状态。

我们以图 11-12、11-13 中的模型为例来分析触发子系统的工作过程。触发方式选择上升沿触发, 仿真过程中示波器的显示如图 11-16 所示。上面的曲线是触发信号波形, 下面的是子系统的输出波形。可以看出, 在  $t=1$  (秒) 时子系统被触发, 开始执行, 输出为 “1”;  $t=1.5$  (秒) 到  $t=2$  (秒) 之间系统第一次触发结束, 等待第二次触发, 这时子系统停止执行, 但由于它的输出保持触发结束时的值, 所以仍然是 “1”。

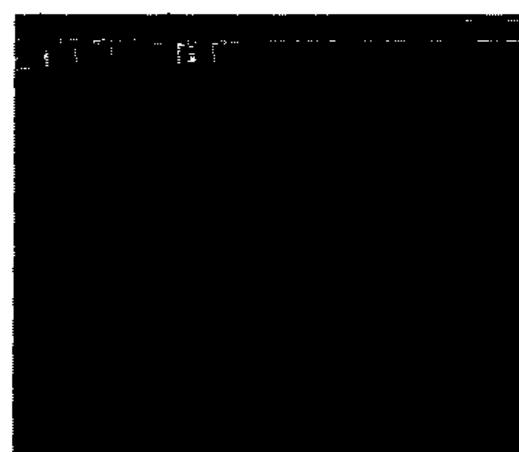


图 11-16 触发信号和子系统的输出波形

### 3. 触发加使能子系统 (triggered and enabled subsystem)

顾名思义, 触发加使能子系统就是使能子系统和触发子系统的组合。把 Trigger 模块和 Enable 模块同时加入到子系统中就形成了触发加使能子系统。图 11-17 即为触发加使能子系统, 图 11-18 为它所在的模型。

触发加使能子系统含有触发信号和使能信号两个控制信号输入端。触发事件发生后，Simulink 检查使能信号是否大于 0，大于 0 即开始执行。

触发加使能子系统的参数设置可以分别进行：在 Trigger 模块中设置触发类型，在 Enable 模块中设置子系统再次开始执行时的状态值。这时，输出端口模块的参数设置和使能子系统相同，这里不再重复。

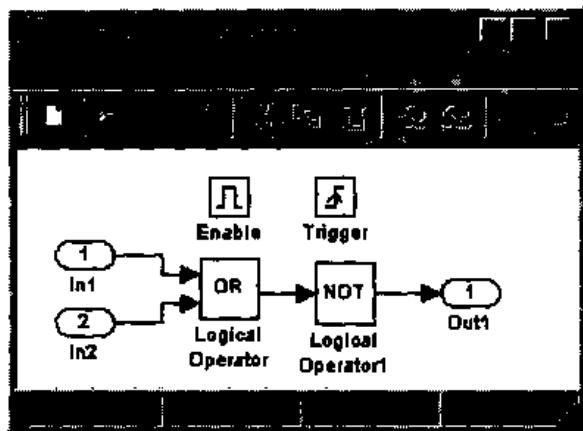


图 11-17 触发加使能子系统

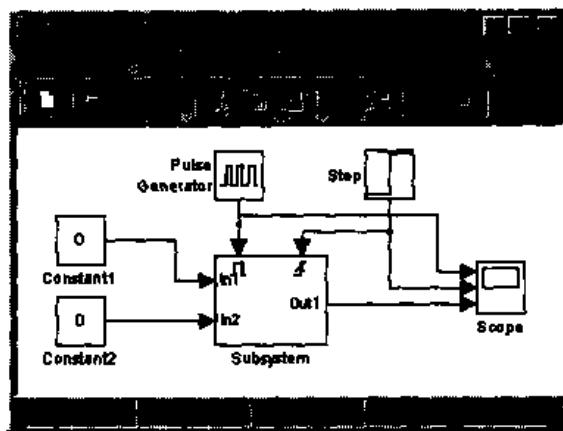


图 11-18 子系统所在的模型

我们以图 11-18 所示的系统为例来说明这类子系统的执行。触发类型选上升沿触发，输出端口中的参数 Output when disabled 设为 reset，重置的初始值为 0。

在图 11-19 的示波器中同时列出了使能控制信号（脉冲）、触发控制信号（阶跃）和子系统的输出信号。从图中可以看出子系统的工作过程： $t=1$  (秒) 前，没有触发信号，这时即使使能信号为正，子系统也不会执行，因而输出结果为“0”；当  $t=1$  (秒) 时，阶跃信号从 0 跳到 1，触发事件发生，这时使能控制信号——脉冲为正，因而子系统开始执行，子系统输出逻辑运算结果“1”； $t=1.5$  (秒) 时，使能信号从“1”变为“0”，子系统停止执行，输出重置为“0”；这以后，虽然有时使能信号为正，但由于再没有发生新的触发，因此该子系统不会再次启动，输出一直为“0”。

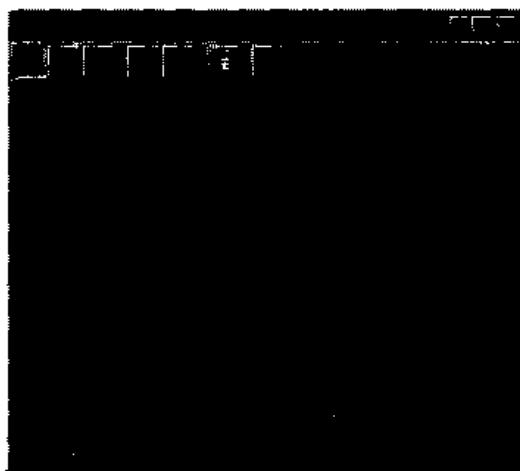


图 11-19 使能信号、触发信号和输出信号

### 11.2.3 子系统的封装 (Masking)

上一节介绍了如何建立子系统。建立的子系统在设置时需要打开其中的每个模块分别输入参数，子系统本身没有基于整体的独立操作界面，这会使子系统的应用受到很大的限制。为解决这些问题，Simulink 提供了子系统封装技术。

所谓封装，就是指为子系统定制对话框和图标，使其具有良好的用户界面。对子系统进行封装可以把子系统中模块的对话框合成为一个，在这个对话框中可以包括模型描述、参数设置等自己定制的内容，界面良好，使用简便；还可以为子系统定制一个能反映其特

性的图标。

封装的过程就是选中子系统模块，在Edit菜单下选择Mask Subsystem项，这时出现一个对话框：Mask Editor，把这个对话框的参数设置好，模块就封装成功了。

Mask Editor对话框有三个夹：Icon、Initialization和Documentation，封装的过程主要就是设置这三个夹中的参数。

首先建立一个Simulink模型，如图11-20所示，图11-21为其中的子系统。这是一个简单的PID（比例积分微分）控制系统模型，其中的子系统实际上是一个PID控制器，通过调节其中的参数p、i和d，来实现对被控对象—Object的有效控制。在模型中选取Subsystem模块，选择Edit菜单下的Mask Subsystem命令，就会弹出如图11-22所示的Mask Editor对话框，下面我们就以这个模型为例讲解如何在Mask Editor对话框中进行设置来建立自己的封装子系统。

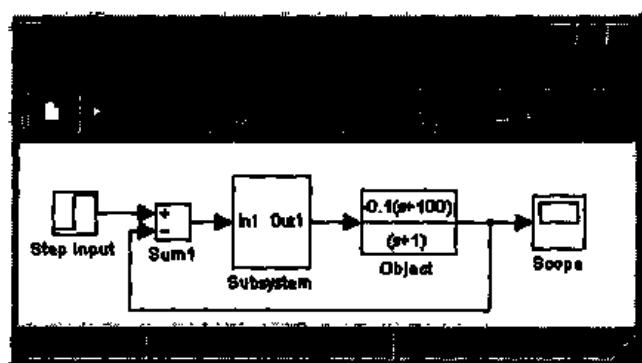


图11-20 封装前的子系统所在的模型

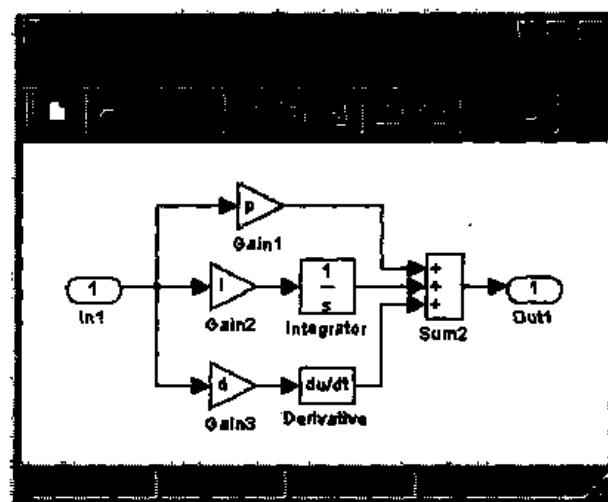


图11-21 PID子系统

### 1. Initialization夹的参数设置

在Initialization夹中可以定义参数的提示符（Prompt）、变量名（Variable）并给出初始化命令（Initialization commands）。如图11-22所示。

#### （1）提示符和相关变量的设置

Initialization夹中部的编辑框Variable和Prompt用来指定需要用户设置的变量名和它的提示符。在封装子系统的参数设置对话框中，提示符提示用户设置什么内容，变量名指定变量来接收用户设置的内容。

参数Assignment有两个选项：Evaluate和Literal。选择Evaluate时，用户输入的内容先由MATLAB进行计算，然后把结果赋给相关变量；选择Literal时，用户输入的内容不经过计算，以字符串格式直接赋给相关变量。其缺省设置为Evaluate。

在本例中，我们需要设置三个变量：比例系数“p”、积分系数“i”和微分系数“d”。设置过程分为以下几步：

第一步，在Prompt后输入提示符“比例系数”，在Variable后输入变量名“p”；

第二步，按下设置对话框左上方的Add按钮，提示符“比例系数”和变量名“p”加入到上面的列表中，同时编辑框Variable和Prompt变空，可以继续输入；

然后重复上面的步骤输入积分系数“i”和微分系数“d”。

由于变量 p、i、d 都是数值变量，因而参数 Assignment 均使用缺省值 Evaluate，不必再做设置。

现在我们按下参数设置对话框下面的 OK 按钮结束封装，在模型框图中双击子系统模块就会弹出如图 11-23 所示的子系统模块参数设置对话框，而不再象封装之前那样显示子系统的内部框图了。

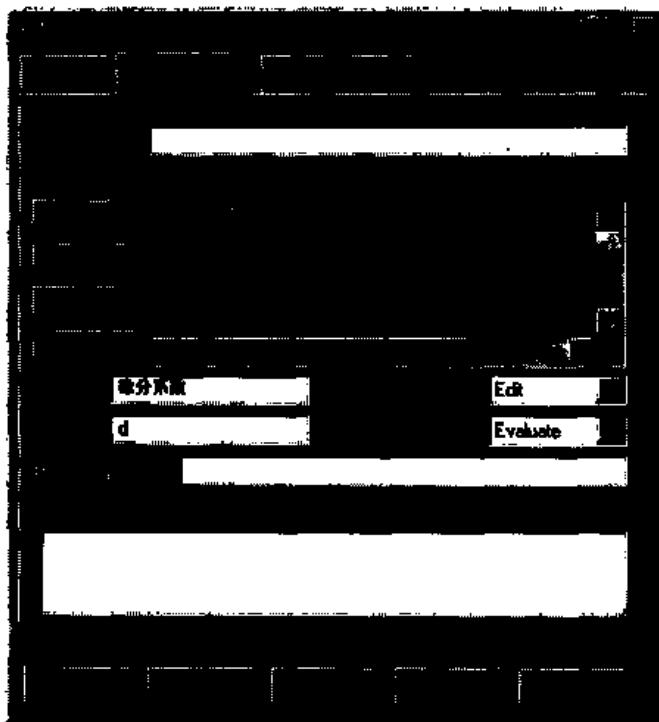


图 11-22 Initialization 夹的参数设置



图 11-23 设置提示符和变量后的子系统模块参数对话框

设置完成的变量和提示符如果需要修改，在列表中选择该变量所在的行，然后在编辑框 Variable 和 Prompt 中做相应修改。若需要改变提示符和变量在封装后的子系统对话框中显示的顺序，可以选中该行后按下 Up 或 Down 按钮进行调节。

### (2) 选取适当的控件类型 (Control type)

可供选择的控件有三种，参数 Assignment 取不同的值它们有不同的含义：

**Edit (编辑框):** 如果参数 Assignment 设为 Evaluate，那么编辑框的返回值是在其中键入的内容的计算结果；如果参数 Assignment 设为 Literal，那么编辑框的返回值是在其中键入的内容本身（字符串格式）。

**Checkbox (复选框):** 如果参数 Assignment 设为 Evaluate，那么选中复选框时，返回值为 1，不选中时返回值为 0；如果参数 Assignment 设为 Literal，那么选中复选框时，返回值为字符串“on”，不选中时返回值为字符串“off”。

**Popup (弹出式菜单):** 如果参数 Assignment 设为 Evaluate，那么选择弹出菜单的某个选项时，返回值是该选项在所有选项中的序号；如果参数 Assignment 设为 Literal，那么选择弹出菜单的某个选项时，把该选项作为一个字符串返回给相应的变量。

### (3) 设置初始化命令 (Initialization commands)

在 Initialization 夹下部的 Initialization commands 编辑框中可以输入初始化命令，这些命令将在开始仿真、更新模块框图、载入模型或重绘封装子系统的图标时被调用。

初始化命令用来定义位于封装工作空间的变量。这些变量可以被所有为封装定义的初始化命令、封装子系统中的模块和绘制封装图标的命令（在 Icon 页的 drawing commands 区域定义）使用。

初始化命令可以由有效的 MATLAB 表达式组成，包括 MATLAB 函数、操作符和在封装工作空间中定义的变量。初始化命令不能访问 MATLAB 工作空间的变量。

所谓封装工作空间，指的是当封装含有初始化命令或封装定义了提示符及相关变量时，Simulink 建立的一个局部工作空间。封装工作空间包括和封装参数相关的变量以及由初始化命令定义的变量。

封装模块及子系统内部包含的模块可以访问本封装工作空间的变量；但不能访问基本工作空间或其它封装工作空间的变量。

对于前面提到的例子中在封装工作空间中定义的变量 p、i、d，它们是和封装子系统中的模块参数相联系的。但是，这两个变量的值并不会直接赋给模块参数，模块是通过访问封装工作空间中的所有变量来获取参数值的。可以这样理解：封装变量和模块参数之间通过封装工作空间来交换数据。

## 2. Icon 夹的参数设置

在 Icon 夹中定制封装模块的图标。如图 11-24，它包括两个编辑框：Mask type 和 Drawing commands，还有几个设置封装图标特性的弹出式菜单。

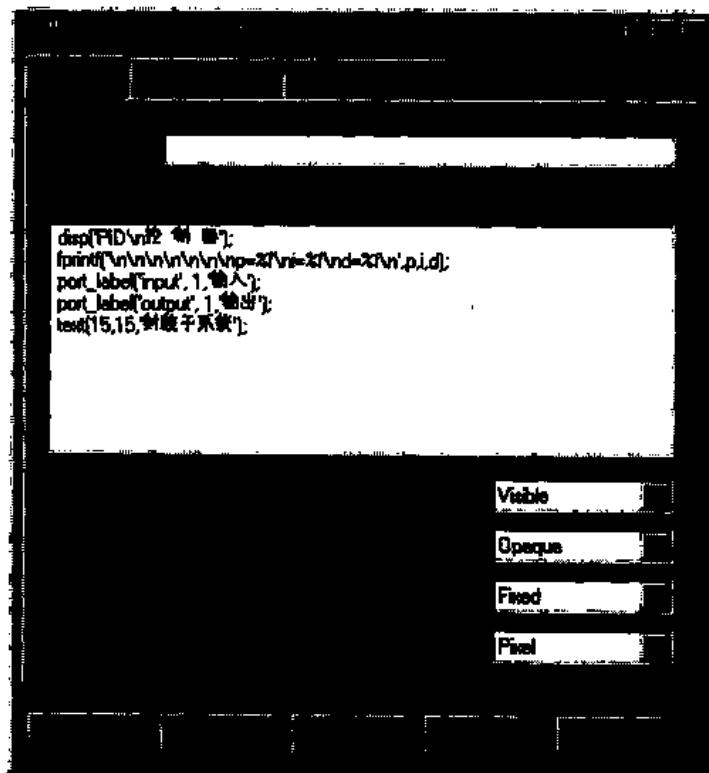


图 11-24 Icon 夹的参数设置

### (1) Drawing commands 编辑框

在编辑框 Drawing commands 中输入命令能够建立用户化的图标，可以在图标中显示文本、图象、图形或传函。

用来在封装图标中显示文本的命令有四个： disp、 text、 fprintf 和 port\_label。分别有如下几种用法：

- disp(variable): 在图标正中显示变量 variable 的值。
- disp('string'): 在图标中显示字符串。
- text(x, y, variable): 在图标的点 (x, y) 处显示变量 variable 的值。
- text(x, y, 'string'): 在图标的点 (x, y) 处显示字符串。
- fprintf('string'): 在图标正中显示字符串。
- fprintf('format', variable): 在图标正中按指定格式显示变量 variable 的值。
- port\_label(port\_type, port\_number, label): 根据指定的端口类型 port\_type、端口号 port\_number 为该端口添加标记 label。port\_type 包括“input”和“output”，对应输入端口和输出端口；port\_number 是相应端口的序号；label 是一个自己指定的字符串，用来标记相应的端口。

命令 text 显示文本或变量时，还可以用如下格式（以文本为例）来限定文本或变量相对于指定点 (x, y) 的排列方式：

```
text(x, y, 'string', 'horizontalAlignment', HOption, 'verticalAlignment', VOption);
```

下面我们举例说明如何在图标上标注文本，如下一组命令的标注结果如图 11-25 所示。

```
disp('PID\n控 制 器');
fprintf('\n\n\n\n\n\nnp=%fni=%fnd=%f\n',p,i,d);
port_label('input', 1,'输入');
port_label('output', 1,'输出');
text(15,15,'封装子系统');
```

除了文本以外，还能在封装图标中显示图形或图象。

显示图形用 plot 函数，显示图象可以用 image 或 patch 函数。

在封装的图标中还可以显示传递函数。命令格式有：

dpoly(num, den): num 为传函的分子向量，den 为传函的分母向量。

dpoly(num, den, 'character'): 当传函需要按“z”的降幂排列显示时，character 的值应取为“z”；当需要按“1/z”的升幂排列显示时，character 的值应取为“z-”。

droots(z, p, k): 显示零极点模型的传函，z 为零点，p 为极点，k 为增益。

这些方法我们不再一一举例，请读者自己实验。

### (2) 设置封装图标的特性

在编辑框 Drawing commands 下方有几个弹出式菜单，在这里对封装图标的特性进行设置。

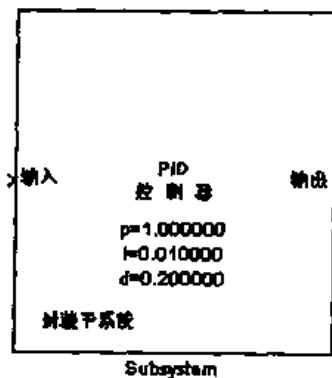


图 11-25 在图标中标注文本

- **Icon frame:** 指的是图标外的矩形边框。设置为 **Visible** 显示边框，设置为 **Invisible** 隐藏边框。
- **Icon transparency:** 指图标的透明度。设置为 **Opaque**（不透明），不显示图标下的内容，比如端口标记（“In1”、“Out1”等）；设置为 **Transparent**，显示图标下的内容。
- **Icon rotation:** 当被封装的模块旋转或翻转时，选择 **Icon rotation** 中的选项 **Fixed** 表示图标不随之转动；若选择 **Rotates** 则表示图标随被封装模块的转动而转动。
- **Drawing coordinates:** 用来设置 **Drawing commands** 中 **plot** 和 **text** 命令使用的坐标系。它包括 **Autoscale**、**Normalized** 和 **Pixel** 三个选项。

**Autoscale:** 根据绘制的点的坐标自动选取坐标系，使坐标中最小的 x 和 y 位于图标左下角，最大的 x 和 y 位于图标右上角。当模块大小改变时，用这种方式绘制的图标随之发生改变。

**Normalized:** 规定图标左下角的坐标为 (0, 0)，右上角的坐标为 (1, 1)。要绘制的点的坐标必须归一化到 [0 1] 之间。当模块大小改变时，用这种方式绘制的图标随之发生改变。

**Pixel:** 以像素为单位绘制图形。当模块大小改变时，用这种方式绘制的图标不随之变化。

### 3. Documentation 夹的参数设置

如图 11-26，在 Documentation 夹的三个编辑框中可以分别为封装模块设置封装类型、描述文本和帮助文本。

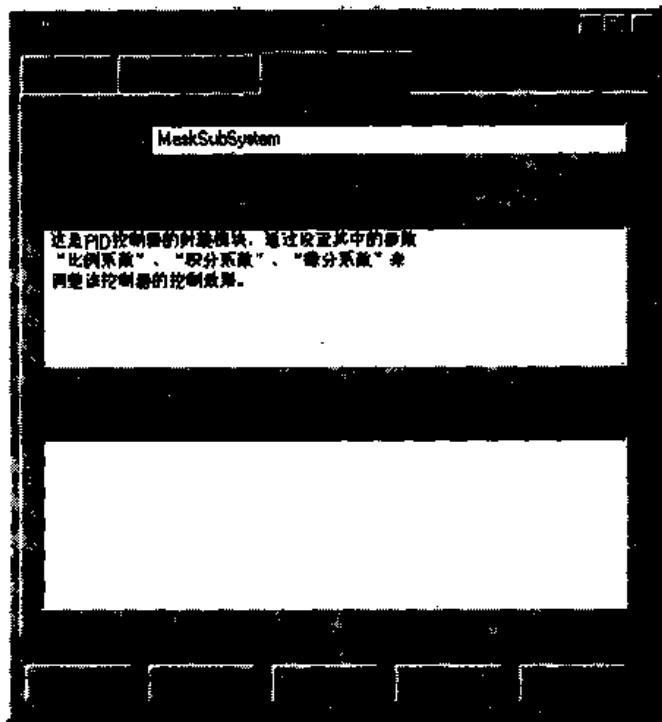


图 11-26 Documentation 夹的参数设置

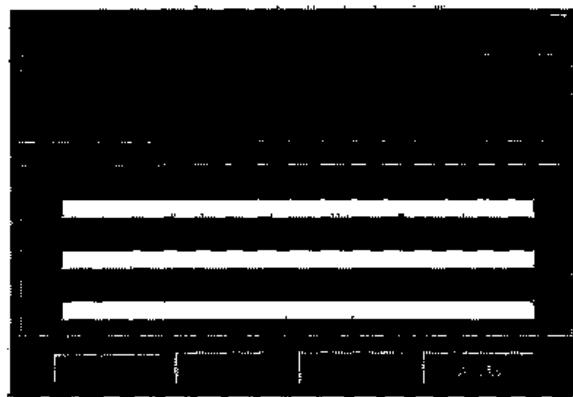


图 11-27 设置出的封装模块对话框

#### (1) Mask Type 编辑框

编辑框 **Mask Type** 在这三个夹中都有，用于设置模块的封装类型，封装类型只是一个

文本意义上的概念，没有什么实际意义。可以输入任何字符串，其作用就是和内置的封装模块区别开。这里输入的字符串加上“(mask)”字符串显示在封装模块的对话框顶部。

### (2) Block Description 编辑框

在这里输入描述文本，输入的内容显示在封装模块对话框的上部，位于 Mask type 之下的边框内。输入的文本一般是对模块的目的或功能的描述。参见图 11-26 中的设置和图 11-27 中设置出的结果。

### (3) Block help 编辑框

在这里输入帮助文本，当封装模块对话框的 Help 按钮被按下时，就会显示这些输入的内容。

## 11.3 构造仿真模型的命令和参数

这一节我们将介绍构造仿真系统模型、设置模块参数和模型参数的另一种方式：使用命令编写程序。这种方式虽然不象构造 Simulink 框图那样简便、直观，但它可以灵活地控制程序的流程，并且参数可以动态设置和修改，这些优点使这种方式可以实现很多框图方式无法实现的目的。

### 11.3.1 构造模型的命令

在表 11-1 中列出了这些命令和它们的功能。命令的具体用法这里不再一一介绍，使用时请参考在线帮助。

表 11-1 构造模型的命令列表

命 令	功 能	命 令	功 能
add_block	在系统中加入一个新模块	gcbh	获取当前模块的句柄
add_line	在系统中加入一条线	gcs	获取当前系统的路径名
bdclose	关闭一个系统窗口	get_param	获取参数值
bdroot	获取根系统的名字	new_system	建立一个新的 Simulink 系统
close_system	关闭一个系统窗口	open_system	打开一个存在的系统
delete_block	从系统中删除模块	replace_block	替换系统中的模块
delete_line	从系统中删除连线	save_system	保存系统
find_system	查找系统、模块、连线或注释	set_param	设置参数值
gcb	获取当前模块的路径名	simulink	打开仿真模块库

有很多命令需要指定 Simulink 系统或模块，这需要同时指定它们的位置。有以下几种情况：

- 指定一个系统：只要指定包含该系统的文件名和文件的路径。
- 指定一个子系统：要指定该子系统所处的系统和其上的各级子系统，假定需要指定的子系统为 subsystem，那么格式为：  
system/subsystem1/subsystem2/.../subsystem
- 指定一个模块：需要指定该模块所处的系统和各级子系统，假定要指定的模块名

为 block，那么格式为：

```
system/subsystem1/subsystem2/.../subsystem/block
```

如果模块名不止一行，那么需要加入“sprintf('\n')”作为分行的标志。例如，如果要指定的模块是 Pulse Generator(脉冲发生器)，那么如下命令可以获取模型 MyMode 中的 Pulse Generator 模块的参数 Period 的值。

```
get_param(['MyModel/Pulse', sprintf('\n'), 'Generator'], 'Period')
```

有的模块名中含有斜线“/”，要指定该模块名时，需要把斜线重复一次。

### 11.3.2 设置参数的命令 set\_param

命令 set\_param 用来设置仿真系统或模块的各种参数，命令格式如下：

```
set_param('obj', 'parameter1', value1, 'parameter2', , ...)
```

其中 obj 为设置的系统或模块的路径名。用这个命令把值 value1、value2……赋给相应的参数 parameter1、parameter2……。

参数 parameter1、parameter2……可以是仿真模型的参数、模块的通用参数、模块的特定参数、模块的回调(callback) 参数。

### 11.3.3 参数设置

#### 1. 模型参数

模型参数用来设置模型的仿真参数、打印选项、回调(callback) 参数等内容。参见附录 C。

模型的回调参数设置在什么时候执行指定的回调程序。参数值就是要执行的程序名或 MATLAB 表达式。

例如，下面的命令设置仿真的开始时间为 10 秒，终止时间为 100 秒：

```
Set_param('MyModel', 'StartTime', 10, 'StopTime', 100)
```

下面的命令设置开始仿真时执行命令“notebook”：

```
Set_param('MyModel', 'StartFcn', 'notebook')
```

#### 2. 通用模块参数

通用模块参数指所有 Simulink 模块都具有的参数。参见附录 C，其中模块的回调参数和模型回调参数类似，只不过它是用于每个模块而不是模型。

例如，下面的命令为模型中的模块“Sine Wave”设置在窗口中的位置：

```
set_param(['MyModel', '/', 'Sine Wave'], 'position', [20, 20, 60, 80])
```

下面的命令使得当模块“State-Space”被打开(双击鼠标左键)时为其参数“A”赋值为 3×3 的随机数矩阵：

```
set_param(['MyModel', '/', 'State-Space'], 'OpenFcn', 'A=randn(3,3)')
```

#### 3. 模块的特定参数

模块的特定参数指每个不同的模块特有的参数。在附录 C 列出了 Simulink 通用库下几个子库中的一些常用模块的参数和取值，封装模块不能用命令 set\_param 设置参数，这些模块在表的“参数”一列中标记了“(masked)”。

## 11.4 Real-Time Workshop 简介

### 11.4.1 Real-Time Workshop 简介

Real-Time Workshop(实时工作间)是一种可以在多平台上(包括 Microsoft Windows 95, Windows 98, Windows NT 和 UNIX)运行的产品。安装 Real-Time Workshop 3.0 需要系统首先安装 MATLAB 5.3 和 Simulink3.0, 而且系统中要有下面三种之中的一种 Windows 环境下的 C 语言编译器, 否则 Real-Time Workshop 将不能正常工作。

- Microsoft Visual C/C++
- Watcom C
- Borland C/C++

作为 Simulink3.0 的一个重要功能模块, Real-Time Workshop 是一种实时开发环境, 可以直接从 Simulink 的模型产生出可移植的程序源代码(C 语言或 Ada 语言代码)并自动构造出能在多种环境中(包括实时系统和单机仿真)实时执行的程序。它为系统从设计到实现提供了一条快捷的途径, 而且简单易用。通过 Real-Time Workshop 可以在远程处理器上实时运行仿真模型, 也可以在主机或外部计算机上运行高速单机仿真。

Real-Time Workshop 具有以下这些主要特性:

1) 能够从定步长的 Simulink 连续时间系统模型、离散时间系统模型和混合系统模型直接产生源代码。

2) 产生的代码是经过优化的, 从而保证了执行速度。

3) 代码具有可移植性, 能够在多种环境中使用。而且可读性强, 便于维护。

4) 自动构造程序, 构造过程完全用户化。

5) 使用可定制的 make 文件建立目标文件, 并可以自动移植到硬件。

6) 具有可扩充的支持多种硬件的驱动程序库。并支持第三方的硬件和工具。

7) 从 Simulink 移植到外部硬件的参数可以在系统运行过程中进行在线调整。

8) 自动函数内联, 允许直接把函数嵌入到生成的代码中。从而可以消除函数调用的开销。

9) 菜单驱动的良好图形用户界面(GUI)使得操作更加简便。

基于 Real-Time Workshop 的这些特性, 它主要用于以下几个方面:

1) 实时控制: 首先在 MATLAB 和 Simulink 下建立控制系统模型, 然后通过 Real-Time Workshop 生成程序源代码, 把源代码编译后可以直接移植到控制系统的硬件上。

2) 实时信号处理: 先用 MATLAB 和 Simulink 设计信号处理的算法, 然后生成程序源代码, 编译后移植到硬件上。

3) 交互式实时参数调整: 可以把 Simulink 作为实时模型的前端, 这样就能够在程序执行过程中调整参数。

4) 实时仿真: 比如模拟系统的训练、实时模型验证和测试。

5) 高速单机仿真

6) 产生可移植的 C 语言代码

### 11.4.2 Real-Time Workshop 的几个基本概念

在讨论如何使用 Real-Time Workshop 之前，我们先介绍几个相关的概念。

Real-Time Workshop 位于菜单 Simulation 下的 Parameters 设置对话框中。还可以从 Tools 菜单下选择 RTW Options 启动 Simulation Parameters 的这一项。如图 11-28 所示。

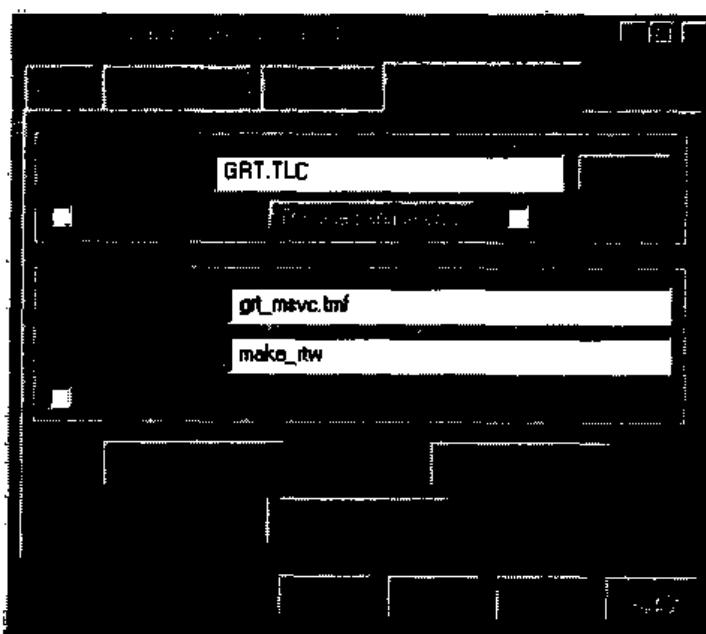


图 11-28 Real-Time Workshop 的设置

为便于理解，我们先来看如下几个概念：

#### 1. 定位 (Targeting)、目标 (Target) 和主机 (Host)

使用 Real-Time Workshop 必须要确定把生成的代码放到什么环境中，这就叫定位 (Targeting)，而这个环境本身则称作目标 (Target)。

主机 (Host) 指的是运行 MATLAB、Simulink 和 Real-Time Workshop 的环境。利用主机中的构造工具，可以生成代码和在目标系统上运行的可执行文件。

#### 2. 目标语言编译器文件 (Target Language Compiler files)

目标语言编译器 (TLC) 文件是指由目标语言编译器编译并执行，描述如何生成目标代码的文件。Real-Time Workshop 利用 TLC 文件把仿真模型翻译为代码。

#### 3. 构造过程 (The Build process)

Real-Time Workshop 的构造过程由 make\_rtw 控制，当按下图 11-28 中的 Build 按钮时 Real-Time Workshop 自动调用 make\_rtw 开始构造。

首先，make\_rtw 编译模型的模块框图，生成 model.rtw 文件，然后 make\_rtw 启动目标语言编译器 (TLC) 生成代码。

#### 4. 模板 make 文件 (Template makefiles)

Real-Time Workshop 用模板 make 文件 (Template makefiles) 把代码编译为可执行文件。模板 make 文件的扩展名是 “.tmf”，名字和系统目标文件有关。例如 grt\_msvc.tmf 指的是一般实时 (Generic Real-Time) 目标的 Visual C/C++ 模板 make 文件。

make 文件是由模板 make 文件生成的，生成的 make 文件的名字是 *model.mk*。通过修改模板 make 文件可以对构造过程进行设置。

具体的设置方法和内容我们将在下一节作详细介绍。

## 11.5 Real-Time Workshop 的设置

在菜单 Simulation 下打开 Parameters 对话框，选择 Real-Time Workshop 选项就进入了如图 11-28 所示的 Real-Time Workshop 设置对话框。还可以在模型窗口的 Tools 菜单下选择 RTW Options，也同样会打开 Real-Time Workshop 设置的对话框。

▲ 注意：应用 Real-Time Workshop 必须在 Solver 设置中把解法设为定步长（Fixed\_step），否则 Real-Time Workshop 将不能运行。

下面我们分别介绍 Real-Time Workshop 设置的各部分内容。

### 11.5.1 System Target File（系统目标文件）

在 System target file 区域指定代码的类型和产生代码的目标。可供选择的系统目标文件类型如图 11-29 所示。选定了系统目标文件，Real-Time Workshop 就会自动为目标文件选择相应的模板 make 文件和 make 命令。

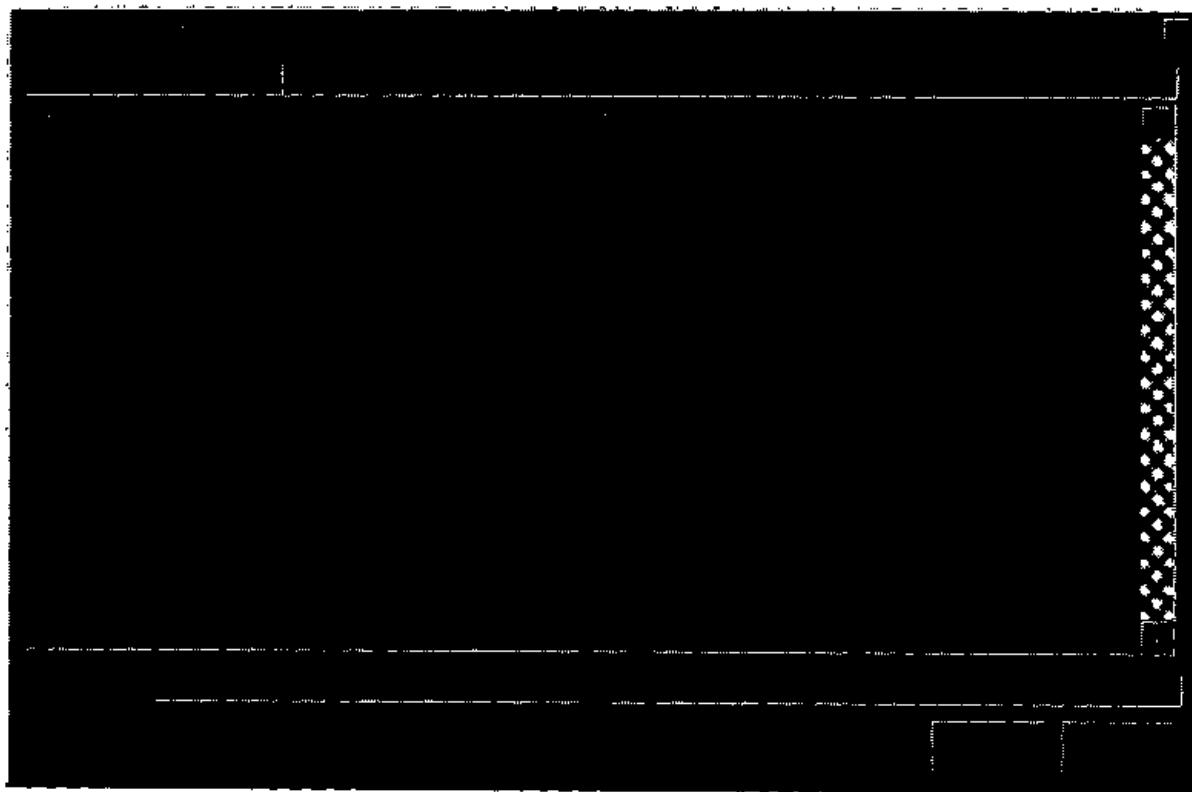


图 11-29 可供选择的系统目标文件类型

在下面的表 11-2 中我们列出了常用的系统目标文件及其代码格式、模板 make 文件和 make 命令。

表 11-2 系统目标文件

系统目标文件	目标/代码格式	模板 make 文件	make 命令
DRT.TLC	DOS (4GW)	drt_watc.tmf	make_rtw
ERT.TLC	一般嵌入式 C 语言	ert_default_tmf	make_rtw
ERT.TLC	适用于 Watcom 的嵌入式 C 语言	ert_watc.tmf	make_rtw
ERT.TLC	适用于 Visual C/C++ 的嵌入式 C 语言	ert_vc.tmf	make_rtw
ERT.TLC	适用于 Visual C/C++ Project Make 文件的嵌入式 C 语言	ert_msvc.tmf	make_rtw
ERT.TLC	适用于 Borland 的嵌入式 C 语言	ert_bc.tmf	make_rtw
GRT.TLC	一般实时目标	grt_default_tmf	make_rtw
GRT.TLC	适用于 Watcom 的一般实时目标	grt_watc.tmf	make_rtw
GRT.TLC	适用于 Visual C/C++ 的一般实时目标	grt_vc.tmf	make_rtw
GRT.TLC	适用于 Visual C/C++ Project Make 文件的一般实时目标	grt_msvc.tmf	make_rtw
GRT.TLC	适用于 Borland 的一般实时目标	grt_bc.tmf	make_rtw
grt_malloc.tlc	带有动态内存分配的一般实时目标	grt_malloc_default_tmf	make_rtw
grt_malloc.tlc	适用于 Watcom 的一般实时目标(动态)	grt_malloc_watc.tmf	make_rtw
grt_malloc.tlc	适用于 Visual C/C++ 的一般实时目标(动态)	grt_malloc_vc.tmf	make_rtw
grt_malloc.tlc	适用于 Visual C/C++ Project Make 文件的一般实时目标(动态)	grt_malloc_msvc.tmf	make_rtw
grt_malloc.tlc	适用于 Borland 的一般实时目标(动态)	grt_malloc_bc.tmf	make_rtw
OSEK_LEO.TLC	LE/O (Lynx embedded OSEK) 实时目标	osek_leo.tmf	MAT_FILE=1 RUN=1 LEO_NODE=osek
RSIM.TLC	快速仿真目标	rsim_default_tmf	make_rtw
RSIM.TLC	适用于 Watcom 的快速仿真目标	rsim_watc.tmf	make_rtw
RSIM.TLC	适用于 Visual C/C++ 的快速仿真目标	rsim_vc.tmf	make_rtw
RSIM.TLC	适用于 Visual C/C++ Project Make 文件的快速仿真目标	rsim_msvc.tmf	make_rtw
RSIM.TLC	适用于 Borland 的快速仿真目标	rsim_bc.tmf	make_rtw
rt_ada_sim.tlc	Ada 语言仿真目标	gnat_sim.tmf	make_rtw -ada
rt_ada_tasking.tlc	Ada 语言多任务实时目标	gnat_tasking.tmf	make_rtw -ada
rtwsfcn.tlc	S 函数目标	rtwsfcn_default_tmf	make_rtw
rtwsfcn.tlc	适用于 Watcom 的 S 函数目标	rtwsfcn_watc.tmf	make_rtw
rtwsfcn.tlc	适用于 Visual C/C++ 的 S 函数目标	rtwsfcn_vc.tmf	make_rtw
tornado.tlc	Tornado (VxWorks) 实时目标	tornado.tmf	make_rtw
win_watc.tlc	Windows 95/98/NT 实时目标	win_watc.tmf	make_rtw

在选定了系统目标文件之后，还可以为目标语言编译器（TLC）选择产生代码的选项，常用的一些选项可以在 Code Generation Options 对话框中设置（在 Real-Time Workshop 中按下 Options 按钮，具体内容将在后面介绍）。其它选项可以在 System target file 区域中目标文件名的后面输入，或者直接添加到系统目标文件中。常用的选项有：

- **-Ipath:** 把路径 *path* 增加到搜索目标文件的路径列表中。例如，“-IE:\matlabprogram”表示把路径 E:\matlabprogram 加到搜索路径范围中。
- **-m[N|a]:** 规定出现错误时报告错误的最多个数（缺省是 5）。例如，“-m2”表示最多报告两个错误。而“-ma”表示报告所有错误。
- **-d[g|n|o]:** 指定调试（debug）的模式——generate、normal 或 off。缺省是 off，当选用 generate 或 normal 时，启动调试模式，为每个 TLC 文件生成记录文件 (\*.log)。
- **-aVariable=expr:** 把变量 *Variable* 赋值为 *expr*。这是最常用的一种方式，式中的变量和取值见表 11-3。

表 11-3 变量及其取值

变量赋值表达式	含义及取值
<b>-aMaxStackSize = <i>N</i></b>	用来限制在函数中声明的局部变量的总字节数不能超过 <i>N</i> ， <i>N</i> 为任意正整数。
<b>-aMaxStackVariableSize = <i>N</i></b>	用来限制在函数中声明的给定的局部变量的字节数不能超过 <i>N</i> ， <i>N</i> 为任意正整数。
<b>-aFunctionInlineMode = "mode"</b>	用来控制函数内联的模式，包括：Automatic、Manual、Either、None 四种模式。 Automatic：用自动内联阈值（AutoInlineThreshold）来决定是否内联。 Manual：把 RTWData 加入到子系统中决定是否内联。
<b>-aFunctionInlineType = "mode"</b>	函数内联的类型，控制函数如何内联，包括两种类型：CodeInsertion、Pragmainline。 使用 CodeInsertion 类型，代码在执行函数调用的位置自动嵌入。 使用 Pragmainline 类型，当适当的编译命令出现时，控制目标语言编译器声明函数。
<b>-aParameterTuning = <i>value</i></b>	参数调整，在运行模式下调整参数。 <i>Value</i> 可以取值为 0、1，分别表示： 0：使这一项无效 1：生成 <i>model.pt</i> 文件
<b>-aAutoInliningThreshold = <i>N</i></b>	当 FunctionInlineMode 是 Automatic 或者 Either 时，这个变量用来设置连接个数的阈值。 <i>N</i> 是任意正整数。
<b>-aWarnNonSaturatedBlocks = <i>value</i></b>	设置一个标志，为具有饱和能力的模块控制溢出警告的显示。 <i>Value</i> 可以取值为 0、1、2，分别表示： 0：不显示警告 1：在模型代码产生过程中显示一个警告 2：显示一个警告，列出所有被警告的模块。
<b>-aBlockIOSignals = <i>value</i></b>	模块 IO 信号，在运行模式下监视信号。 <i>Value</i> 可以取值为 0、1，分别表示： 0：使这一项无效 1：生成 <i>model.bio</i> 文件
<b>-PragmainlineString = "string"</b>	当 FunctionInlineType 设置为 Pragmainline 时，这里应当设置为编译器用来内联函数的命令。

### 11.5.2 内联参数和可调参数

内联参数（Inline parameters）指从运行状态的模型中移动采样时间为常数的模块的方式。这些模块的输出信号在模型启动过程中设置。当这个选项被选中时，Simulink 自动把

所有内联参数设置为恒定常量。

选中了内联参数选项后，可调参数（Tunable parameters）按钮自动激活，按下这个按钮打开 RTW Tunable Parameters 对话框，如图 11-30 所示。RTW Tunable Parameters 对话框支持下面几种功能：

### 1. 参数调整

在 RTW Tunable Parameters 对话框的 Variable 域输入任何模型参数的名字并单击 Add 按钮，这时就解除了该参数的内联。这实际上取消了对于该变量的内联参数（Inline parameters）复选框。而其它模型参数仍保持内联。

### 2. 存储类型（Storage Class）

在这个下拉式菜单中可以为选择的变量改变存储类型。打开下拉式菜单我们可以看到其中包括四种类型，它们分别是：

- Auto：使 Real-Time Workshop 把变量保存在一个稳定的数据结构中。这是 Real-Time Workshop 缺省的存储类型。
- Exported Global：把变量声明为全局变量（global），从而能够从生成的代码外部访问该变量。
- Imported Extern：把变量声明为外部变量（extern），它必须从代码外部声明。
- Imported Extern Pointer：把变量声明为外部变量指针（extern pointer），它必须从代码外部声明。

这些类型对其它 C 代码和 Real-Time Workshop 生成的代码的连接很有用。

### 3. 存储类型限定词（Storage Type Qualifier）

在 Storage Type Qualifier 域中输入任何和变量类型声明相应的字符串。注意，不管输入是否正确，Real-Time Workshop 不对这个字符串作检查。

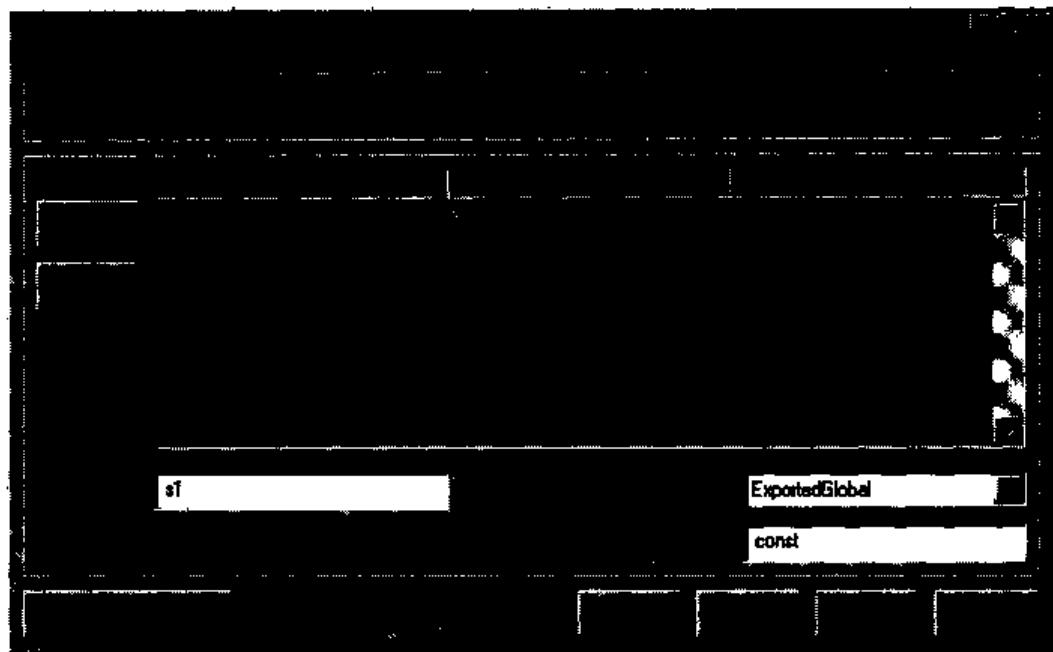


图 11-30 RTW Tunable Parameters 对话框

### 11.5.3 模板 make 文件 (Template Makefile)

当 Generate code only 复选框不被选中时, 就要使用模板 make 文件。它可以唯一地识别要建立可执行文件的目标。

#### 1. 模板 make 文件的记号扩充

模板 make 文件是基本的 make 文件, 把它的一些记号 (token) 扩充就可以建立自己模型的 make 文件 (*model.mk*)。把 Real-Time Workshop 中指定的模板 make 文件的每一行拷贝过来, 并扩充 Real-Time Workshop 的记号就生成了 *model.mk* 文件。表 11-4 列出了这些记号和它们的扩充内容。模板 make 文件中的记号两端用 “>” 和 “<” 标记。

表 11-4 模板 make 文件的记号和扩充

记 号	扩 充
>COMPUTER<	计算机类型
>MAKEFILE_NAME<	从模板 make 文件生成的 make 文件名: <i>model.mk</i>
>MATLAB_ROOT<	MATLAB 安装的目录
>MATLAB_BIN<	MATLAB 执行的位置
>MEM_ALLOC<	内存如何分配, 有两种: RT_MALLOC (动态) 和 RT_STATIC (静态)
>MEXEXT<	MEX 文件扩展名
>MODEL_NAME<	进行构造的 Simulink 模型名
>MODEL_MODULES<	产生的附加源模块 (*.c) 的文件名。
>MODEL_MODULES_OBJ<	附加源模块的 Object 文件名 (*.obj)。
>MULTITASKING<	解法是否为多任务模式: 是——1, 不是——0。
>NUMST<	模型中的采样次数。
>RELEASE_VERSION<	MATLAB 的发行版本。
>S_FUNCTIONS<	非内联的 S 函数源文件列表。
>S_FUNCTIONS_LIB<	可以用来连接的 S 函数库。
>S_FUNCTIONS_OBJ<	非内联的 S 函数源文件对应的 Object 文件名 (*.obj)。
>SOLVER<	解法的源文件名
>SOLVER_OBJ<	解法的 Object 文件名。
>TID01EQ<	连续任务和第一个离散任务的采样速率是否相等: 1——相等, 0——不相等。
>NCSTATES<	连续状态的个数。
>BUILDARGS<	传递给 make_rtw 命令的选项。

下面举出模板 make 文件 *grt\_vc.tmf* 中的记号和在自己建立的模型 *rtwttest.mdl* 的 make 文件 *rtwttest.mk* 中的扩充。

在 *grt\_vc.tmf* 中的记号部分如下:

#----- Tokens expanded by make\_rtw -----

.....

MODEL =>MODEL\_NAME<

```

MODULES      =|>MODEL_MODULES<|
MAKEFILE     =|>MAKEFILE_NAME<|
MATLAB_ROOT  =|>MATLAB_ROOT<|
MATLAB_BIN   =|>MATLAB_BIN<|
S_FUNCTIONS  =|>S_FUNCTIONS<|
S_FUNCTIONS_LIB =|>S_FUNCTIONS_LIB<|
SOLVER       =|>SOLVER<|
NUMST        =|>NUMST<|
TID01EQ      =|>TID01EQ<|
NCSTATES    =|>NCSTATES<|
BUILDARGS    =|>BUILDARGS<|
MULTITASKING =|>MULTITASKING<|
EXT_MODE     =|>EXT_MODE<|

```

在 rtwtest.mk 中的扩充部分如下：

```

MODEL        = rtwtest
MODULES      =
MAKEFILE     = rtwtest.mk
MATLAB_ROOT  = D:\MATLAbR11
MATLAB_BIN   = D:\MATLAbR11\bin
S_FUNCTIONS  =
S_FUNCTIONS_LIB =
SOLVER       = ode5.c
NUMST        = 2
TID01EQ      = 1
NCSTATES    = 2
BUILDARGS    =
MULTITASKING = 0
EXT_MODE     = 0

```

对比这两个文件就可以看出它们之间的区别和相互关系。除了这部分内容，两个文件的其它部分是一样的。

## 2. 模板 make 文件的结构

模板 make 文件的结构包括四部分：

第一部分主要包括描述此 make 文件目标的注释。

第二部分定义了一些宏指令 (macro)，告诉 make 命令 “make\_rtw” 如何处理模板 make 文件。这些宏指令有：

(1) MAKECMD——用来调用 make 命令。例如，对于 Visual C/C++ 编译器，这条指令为：

MAKECMD = nmake

相应的 make 命令为：

nmake -f *model.mk*

(2) HOST——模板 make 文件定位的平台，可以为：

HOST = PC、UNIX 或其它

(3) BUILD——告诉 make\_rtw 是否在 Real-Time Workshop 构造过程中调用 make 命令。这条指令为：

BUILD = yes (调用) 或 no (不调用)

(4) SYS\_TARGET\_FILE——系统目标文件名。make\_rtw 使用这条指令来进行一致性检验，验证在 Real-Time Workshop 参数设置中设定的系统目标文件是否正确。

(5) BUILD\_SUCCESS——这是一条可选宏指令，用来指定构造成功后显示的内容。例如：

BUILD\_SUCCESS = ### Celebrate build success !!!

(6) BUILD\_ERROR——可选宏指令，用来指定构造过程中出现错误时显示的提示信息。例如：

BUILD\_ERROR = 'Some Error occur while building!

(7) DOWNLOAD——可选宏指令，可以指定为 yes 或 no。当选 yes 时，make 命令会再次被下载目标调用，这时 make 命令的表达式为：

nmake -f *model.mk* download

(8) DOWNLOAD\_SUCCESS——可选宏指令，用来指定下载成功后显示的内容。例如：

DOWNLOAD\_SUCCESS = ### Celebrate download success !!!

(9) DOWNLOAD\_ERROR——可选宏指令，用来指定下载过程中出现错误时显示的提示信息。例如：

DOWNLOAD\_ERROR = 'Some Error occur while downloading!'

第三部分定义了记号及其扩充。如前面所述。

第四部分包括在构造代码的可执行文件时用到的编译连接选项、make 命令规则等等内容，这部分和所用的编译器类型有关。

模板 make 文件的一般结构如下：

**第一部分：注释**

```
# Copyright (c) 1994-1998 by The MathWorks, Inc. All Rights Reserved.
```

```
# File      : grt_vc.tmf   $Revision: 1.43 $
```

```
# Abstract:
```

**第二部分：宏指令**

```
#----- Macros read by make_rtw -----
```

```
MAKECMD      = nmake
```

```
HOST         = PC
```

```
BUILD        = yes
```

```
SYS_TARGET_FILE = grt.tlc
```

**第三部分：记号及其扩充**

```
#----- Tokens expanded by make_rtw -----
```

#### 第四部分：其它设置

```
#----- Tool Specifications -----
工具
#----- Include/Lib Path -----
包含文件/库文件路径
#----- External mode -----
外部代码
#----- Compiler and Linker Options
编译、连接选项
# Optimization Options
OPT_OPTS = $(DEFAULT_OPT_OPTS)
# General User Options
OPTS =
.....
#----- Source Files -----
源文件
#----- Rules -----
命令规则
#----- Dependencies -----
相关文件
```

#### 11.5.4 make 命令

从模板 make 文件建立了 *model.mk* 文件之后，Real-Time Workshop 就会调用 make 命令，把源代码文件构造为可执行文件。

可以在 Real-Time Workshop 中使用的 make 命令有很多种版本，但任何版本都必须按照下面的命令格式：

*makecommand -f model.mk*

式中 *makecommand* 由模板 make 文件的 MAKECMD 宏指令定义。还可以为 make 命令指定其它一些选项，例如，在 *make\_rtw* 后指定 Visual C/C++ 用来优化代码的选项 OPT\_OPTS=-Ot，其表达式如下：

*make\_rtw OPT\_OPTS=-Ot*

这样就会使 make 命令成为如下格式：

*makecommand -f model.mk OPT\_OPTS=-Ot*

如果需要调试（debug），该项应设置为 OPT\_OPTS=-Zd。构造选项在所用模板 make 文件头的注释部分都有说明，使用时可以参考相应的注释。

#### 11.5.5 Options 按钮

在 Real-Time Workshop 设置窗口中有一个 Options 按钮，单击此按钮就会出现如图 11-31 所示的 Code Generation Options（代码产生选项）对话框。对不同类型的系统目标文件

这个对话框的内容也不尽相同，我们以 GRT.TLC 文件为例来说明其中一些选项的含义。

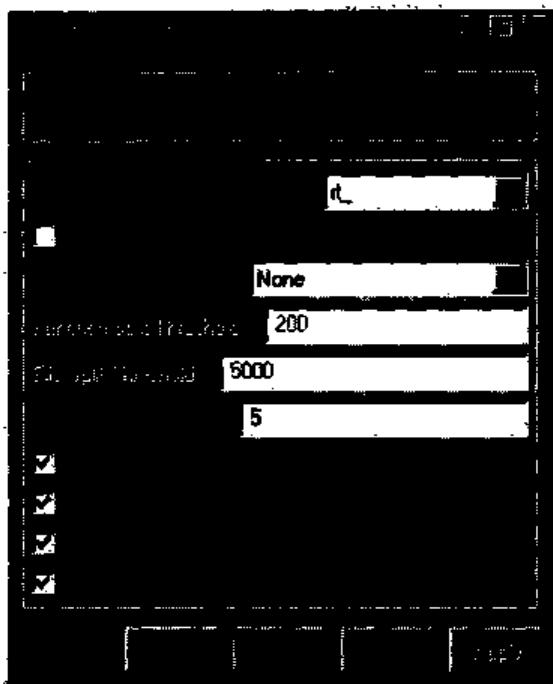


图 11-31 Code Generation Options 对话框

#### 1. Loop rolling threshold (滚动循环阈值)

设置一个发生滚动循环的阈值，正整数，缺省值是 5。当代码中有算法需要的增量个数超过这个阈值时，整个算法就会滚动为一个 for 循环。

#### 2. Show eliminated statements (显示删除语句)

在生成的代码中给出删除的语句（优化结果）的注释。

#### 3. Verbose builds (详细构造)

使得在命令窗口显示出代码产生的过程和编译器的输出。

#### 4. Inline invariant signals (内联恒定信号)

Invariant signals 指在 Simulink 仿真过程中不发生变化的模块输出信号。它和 invariant constants (恒定常量) 是不同的。

如果选中这个选项，Real-time Workshop 会在生成的代码中内联恒定信号。这和 Real-time Workshop 设置中的 Inline parameters 有所不同，后者内联的是恒定常量。但是它们之间有一定的联系，如果在 Real-time Workshop 设置中不选择 Inline parameters，那么 Inline invariant signals 将不执行，但反过来 Inline parameters 可以单独执行。因而需要在执行 Inline invariant signals 前选中 Inline parameters。

#### 5. Local block outputs (局部模块输出)

选中这一项后，Real-Time Workshop 会把尽可能多的模块输出信号作为局部变量处理。

### 11.5.6 文件拆分和函数拆分

在图 11-31 中我们看到还有一个 Function Management (函数管理) 下拉菜单及其相关设置。在这里可以设置文件和函数的拆分。

有些编译器对文件大小有限制，当代码长度超过指定范围时 Real-Time Workshop 可以对源代码进行拆分，使拆分后的每部分代码长度都在规定范围内。支持文件和函数拆分的系统目标文件类型有：一般实时目标（GRT.TLC）、嵌入式 C 语言目标（ERT.TLC）和快速仿真目标（RSIM.TLC）。

文件和函数拆分都需要工具 Perl 5.0 以上版本支持，这个工具位于 MATLAB 5.3 的 bin 目录下。

文件拆分用来缩减文件的长度。这可以在 Code Generation Options（代码产生选项）对话框中设置，如图 11-32 所示。

从图中可以看出，当选中 File Splitting（文件拆分）时，File Split Threshold（文件拆分阈值）自动成为可编辑状态，其缺省阈值为 5000。同时，Show eliminated statements（显示删除语句）选项自动成为不可选状态。

文件拆分有一定的局限性，主要在于它只能从函数的边界处拆分，不能拆开函数，因而当函数比较大时，文件拆分往往达不到目的，这时就要同时使用函数拆分和文件拆分。函数拆分用来把大的函数拆分为小的函数。

设置函数拆分也是在 Code Generation Options（代码产生选项）对话框中，如图 11-33 所示。

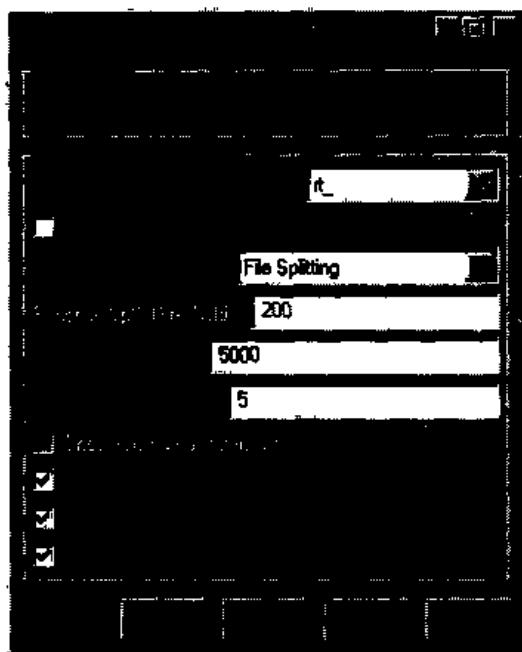


图 11-32 设置文件拆分

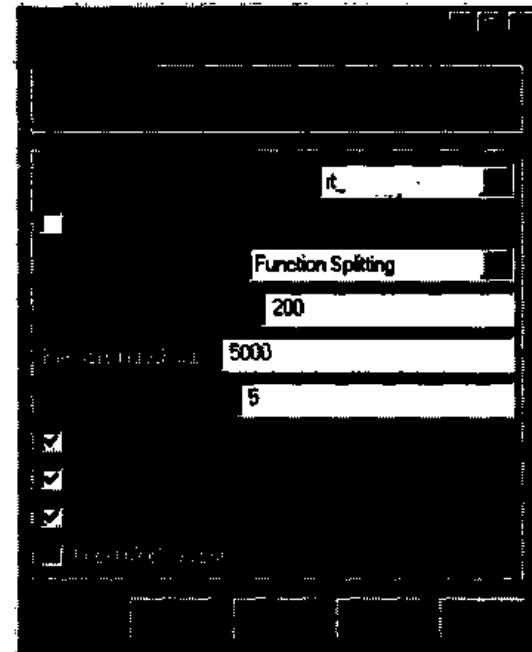


图 11-33 设置函数拆分

如图 11-33 中所示，当选中 Function Splitting（函数拆分）时，Function Split Threshold（函数拆分阈值）成为可编辑状态，其缺省阈值为 200。同时，Local block outputs（局部模块输出）选项成为不可选状态。

函数拆分一般只在模块的边界上，如果模块的边界不在函数拆分的阈值上，那么函数就在下一个模块的边界位置进行拆分。

函数拆分的对象只包括以 Mdl、Sys 或模型名开头的函数。拆分后的函数名在原函数名之后添加一个整数，从“1”开始，依次递增。例如，如果函数 MdlStart 被拆分为三个函数，

那么它们的名字依次为：MdlStart1、MdlStart2、MdlStart3。

还可以选择 Function and File Splitting（函数和文件拆分）来同时拆分函数和文件，这时函数拆分和文件拆分的阈值设置同时变为被激活，而 Show eliminated statements 和 Local block outputs 选项都成为不可选状态。

## 11.6 外部模式（External Mode）

外部模式是 Real-Time Workshop 提供的一种仿真模式，支持在实时环境中在线调整参数。“外部”指的是这种模式包括两个不同的环境：主机（host）和目标（target）。主机是 MATLAB 和 Simulink 运行的计算机，目标指用 Real-Time Workshop 生成的可执行文件运行的计算机。

外部模式通过在 Simulink 和 Real-Time Workshop 生成的代码之间建立一个通信通道进行工作。通信通道可以是网络协议（如 TCP——Transmission Control Protocol），也可以是共享内存。

在外部模式下，Simulink 等待参数的改变，一旦 Simulink 接收到参数的变化，它就把新参数提供给目标。

要使用外部模式，首先需作如下几方面的设置：

- 在 Simulation 菜单下选择 External 选项。
- 在 Simulation Parameters 对话框的 Solver 页中设置 Solver Options Type 为 Fixed-step（定步长）。
- 在 Simulation Parameters 对话框的 Real-Time Workshop 页中按下 Options 按钮，选择其中的 External mode 复选框。
- 在 Simulation Parameters 对话框的 Real-Time Workshop 页中不能选择 Inline parameters 复选框，否则外部模式将不能运行。

Real-Time Workshop 为外部模式提供了一个图形用户界面的控制面板来对外部模式进行设置。在 Tools 菜单下选择 External Mode Control Panel 就会弹出这个控制面板，如图 11-34 所示。

这里我们将着重介绍三个设置对话框的使用。



图 11-34 外部模式控制面板

### 11.6.1 Target Interface (目标连接) 对话框

在图 11-34 所示的控制面板中按下 Target Interface 按钮，会出现如图 11-35 所示的目标连接设置对话框。

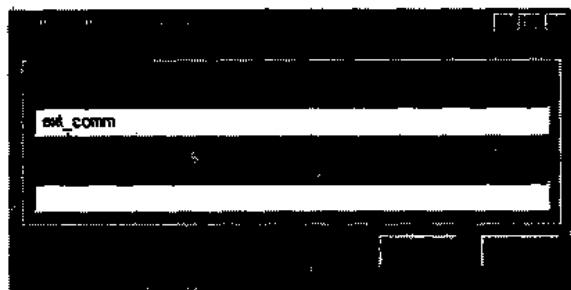


图 11-35 目标连接设置

对话框中有两个选项：MEX-file for external interface 和 MEX-file arguments。在上面的选项中设置外部连接的 MEX 文件名。例如：设置为 ext\_comm 时，对应的目标是 grt 和 Tornado 类型，通信通道采用 TCP；设置为 win\_tgt 时，对应于 Real-Time Windows 目标，通信通道采用共享内存模式。

下面的选项设置外部连接 MEX 文件的参数，它由三部分内容组成：目标网络名称+冗长级别+TCP 服务器端口号。

- 目标网络名 (Target network name)：指运行外部程序的计算机网络名。缺省情况下指 Simulink 运行的计算机。
- 冗长级别 (Verbosity level)：指在数据传递过程中显示的详细信息的级别。这一项可以取值为 0 或 1，取 0 表示不显示信息，取 1 表示显示详细信息。
- TCP 服务器端口号 (TCP server port number)：缺省值是 17725，可以改为在 245 到 65535 之间的任意整数。

### 11.6.2 Signal & Triggering (信号和触发) 对话框

在控制面板中按下 Signal & Triggering 按钮就会弹出如图 11-36 所示的信号和触发设置对话框。这个对话框中包括信号选择、触发信号选择、触发设置等几方面的内容。下面我们分别加以介绍。

#### 1. 信号选择

指定哪个示波器模块在外部模式下被激活。外部模式现在只支持示波器模块的信号选择。模型中的任何示波器模块都会显示在 Signal selection 列表框中。按下列表框右边的 Select all 和 Clear all 按钮就会选择或不选所有列出的信号。还可以用鼠标双击某个列出的信号或通过选择右边的单选按钮 on 和 off 来改变该信号的选中状态。当信号被选中时，会在信号名称的左侧出现“X”标记。

#### 2. 触发信号选择

选择信号指定何时在示波器模块显示数据。选择触发信号的方法是：首先在信号列表中选择一个信号，然后单击右侧的 Trigger signal 按钮，这个信号就被指定为触发信号。这时在信号名称的左侧会出现一个标记“T”。不过只有当 Trigger source (触发源) 设置为 signal

时触发信号才有效。

### 3. 触发选项

设置何时、如何在示波器模块显示数据。

在图 11-36 所示的对话框下方有一个 Trigger 面板，触发选项就是在这部分进行设置。下面我们分别介绍其中的选项。

- **Source:** 包括 manual 和 signal。选择 manual 时，一旦控制面板中的 Arm trigger 钮被按下，外部模式立刻开始记录数据。选择 signal 时，只有当所选的触发信号满足触发条件时，才会开始记录数据。
- **Duration:** 记录数据的持续时间，其中的数值为触发后外部模式记录数据所用的采样步长的个数。
- **Mode:** 包括 normal 和 one-shot 两种可选方式。在 normal 方式下，外部模式在每次触发结束后自动重置触发状态，等待下一次触发。在 one-shot 方式下，每次触发只记录一个缓冲区的数据。
- **Delay:** 触发延时。其中的数值为采样步长的个数。可以是正数或负数。正数延时表示滞后触发，指触发发生后到开始数据采集所延迟的时间；负数延时表示预触发，指在触发之前一段时间先开始数据采集。
- **Arm when connect to target:** 选中这个选项后，外部模式一旦连接到目标就自动开始等待触发。

当 Trigger 面板中的 Source 设置为 signal 时，图 11-36 所示的对话框右下方的 Trigger signal 面板就会被激活。

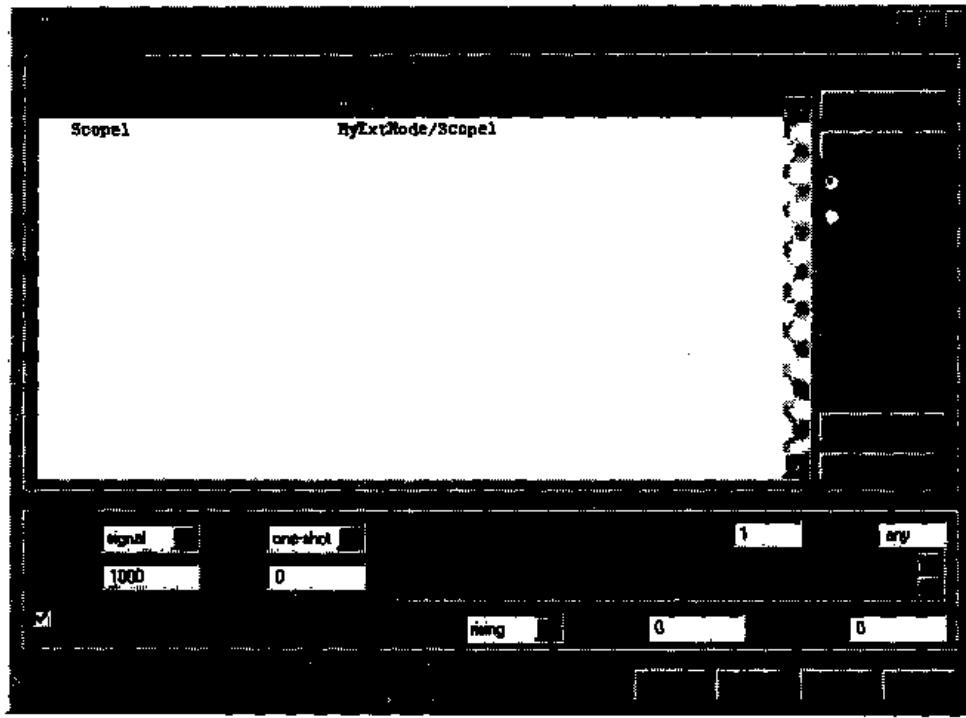


图 11-36 信号和触发设置

缺省条件下，指定模块的第一个输入端口的任何元素都会使触发发生，而很多情况下，这是不能满足要求的。为了使触发按照需要进行，就要对 Trigger signal 面板中的编辑框 Port

和 Element 进行设置。Port 可以设置为某个端口的序号（正整数）或者“last”（指最后一个端口）；Element 可以设置为任何数值或“any”（任何元素）或“last”（最后一个元素）。

在这部分的最下方还有三个设置，它们分别为：

- Direction：包括 rising、falling 和 either 三个选项。这用来指定触发事件，rising 指上升沿触发，falling 指下降沿触发，either 则指上升沿和下降沿都触发。
- Level：设置触发电平，缺省值是零。当信号沿指定的方向超过这个值时即被触发。
- Hold-off：设置触发保持时间，缺省值是零。指终止触发到下一次等待触发之间的时间间隔，仍用采样步长的个数表示。这项设置只用于 normal 方式，而不能用于 one-shot 方式。

### 11.6.3 Data Archiving（数据存档）对话框

在控制面板中按下 Data Archiving 按钮就会弹出如图 11-37 所示的数据存档设置对话框。

从图中可见，这个对话框包括目录注释（directory note）、文件注释（file note）、数据存档（data archiving）等几方面的设置。

- 目录注释：按下 Edit directory note 按钮会自动打开 MATLAB Editor，在该窗口输入目录注释文档，并保存到指定目录下。
- 文件注释：按下 Edit file note 按钮会自动弹出搜寻文件对话框，显示出当前工作目录下的 MAT 文件，选择需要做注释的文件，随之出现一个编辑窗口，在该窗口中输入对该文件的注释，按 OK 即可。
- 数据存档：选择 Enable Archiving 复选框后和数据存档相关的几个选项就被激活，图 11-37 所示就是选项被激活的状态，通过这些选项的设置来实现数据存档。



图 11-37 数据存档设置

在一般情况下（不采用数据存档），外部模式处理数据的方式有两种，一种是在 one-shot 触发方式下，在触发事件发生后，每个示波器模块把它的数据写到工作区，如果另一个 one-shot 触发发生，保存在工作区的数据就会被覆盖。另一种是在 normal 方式下，每个触发发生后自动重置触发状态，等待触发，因而除了最后的一个触发外，其余的每个触发都可以看成是中间状态。由于触发随时可能发生，因此保存到工作区的中间结果会被不可预测地覆盖，得到的结果一般就只能是最后一次触发的数据，而中间结果都被丢掉了。

如果已知在两次触发之间的时间足够检查中间结果，那么可以选择 Write intermediate results to workspace 复选框，把中间结果保存到工作区，但这种方法并不能保证工作区的数

据不被后发生的触发的数据覆盖。

要有效地保存结果，包括中间结果，需要进行如下这些设置：

- **Directory 编辑框：**指定保存数据的目录。如果选择了复选框 **Increment directory when trigger armed**，外部模式自动为目录名增加后缀。
- **File 编辑框：**指定保存数据的文件名。如果选择了复选框 **Increment file after one-shot**，外部模式自动为文件名增加后缀。
- **Increment directory when trigger armed 复选框：**选中这一项后，每次按下 **Arm trigger** 按钮开始触发，都在不同的目录保存数据，外部模式自动在目录名后面增加序号建立新目录。例如 **Directory1**, **Directory2**, **Directory3** 等等。
- **Increment file after one-shot 复选框：**选中这一项后，每次都把数据保存到不同的文件中，外部模式自动在文件名后面增加序号建立新文件。例如 **File1**, **File2**, **File3** 等等。
- **Append file suffix to variable names 复选框：**在前一个选项中，每次保存数据的文件名都不相同，但文件中的变量名是一样的，如果要在 MATLAB 中同时载入这些文件，对其中的数据进行比较，那么载入的变量由于变量名相同会相互覆盖。当选中这一项时，保存到文件中的变量名会随文件名增加序号而同时也增加序号。例如，对于一个变量 **tout**，保存到文件 **File1**, **File2**, **File3** 中的变量名依次为 **tout1**, **tout2**, **tout3**。

到这里我们已经讲述了在外部模式控制面板（External Mode Control Panel）中 Configuration 区域里几个项目的设置，外部模式的基本设置已经完成了。下面就可以运行外部模式，进行仿真了。

按下图 11-34 中控制面板上方的 **Connect** 按钮或者在 **Simulation** 菜单下选择 **Connect to target** 来把主机连接到指定的目标，然后就可以开始仿真。

仿真过程中的在线参数调整有两种方式，一种是选择控制面板中的 **Parameter tuning** 区域内的 **batch download**（批量下传）复选框，**Download** 按钮会被激活，一批参数调整好后按下 **Download** 按钮调整的参数才传递到目标；另一种是不选择 **batch download** 方式，那么参数调整后立刻下传到目标。

在这一节中我们对外部模式（External Mode）的基本操作和参数设置做了比较具体的介绍，这部分功能十分强大，它可以把仿真模型的运行提高到网络的层次上，虽然目前国内的应用范围还不广泛，但它的发展前景十分可观。

## 11.7 Real-Time Workshop 函数库

Real-Time Workshop 为用户提供了一个函数库，借助这个库可以更加灵活地建立实时系统模型和生成代码。

### 11.7.1 Real-Time Workshop 函数库概述

如图 11-38，在 Simulink Library Browser 中 Real-Time Workshop 函数库由以下这五部分子库组成：

- 1) Custom Code (自定义代码): 借助其中的模块可以在生成的源代码文件或相关函数中插入自定义代码。这部分我们将在后面做详细介绍。
- 2) DOS Device Drivers (DOS 设备驱动): DOS 方式下几种模拟输入输出设备、数字输入输出设备的驱动模块。在图 11-39 中列出了这个子库中包含的驱动模块类型。
- 3) Interrupt Templates (中断模板): 以其中的模块为模板可以建立自己的异步中断系统。图 11-40 为这个子库包含的内容。
- 4) S-Function Target (S 函数目标): 此模块用以和 Real-Time Workshop 的 S 函数的代码格式连接。图 11-41 是这个子库中的内容。
- 5) VxWorks Support (VxWorks 支持): 包括支持 VxWorks (Tornado) 的一些模块。其中包括 I/O 设备支持, 如图 11-42; 异步系统支持, 这一部分和中断模板子库的内容相同。

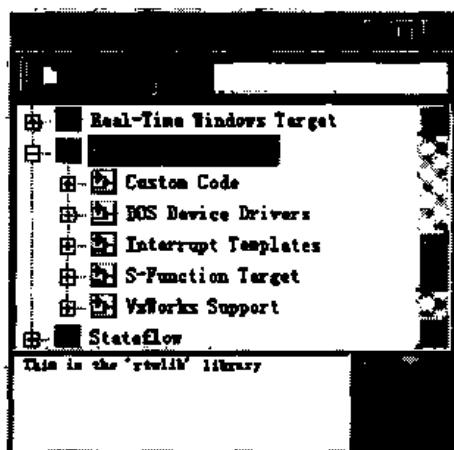


图 11-38 Real-Time Workshop 函数库

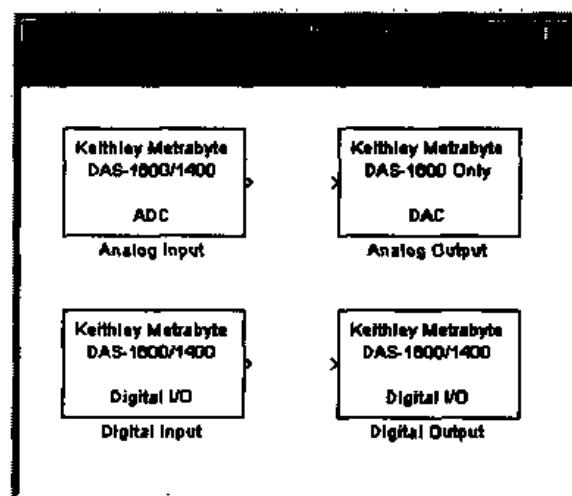


图 11-39 DOS 设备驱动模块

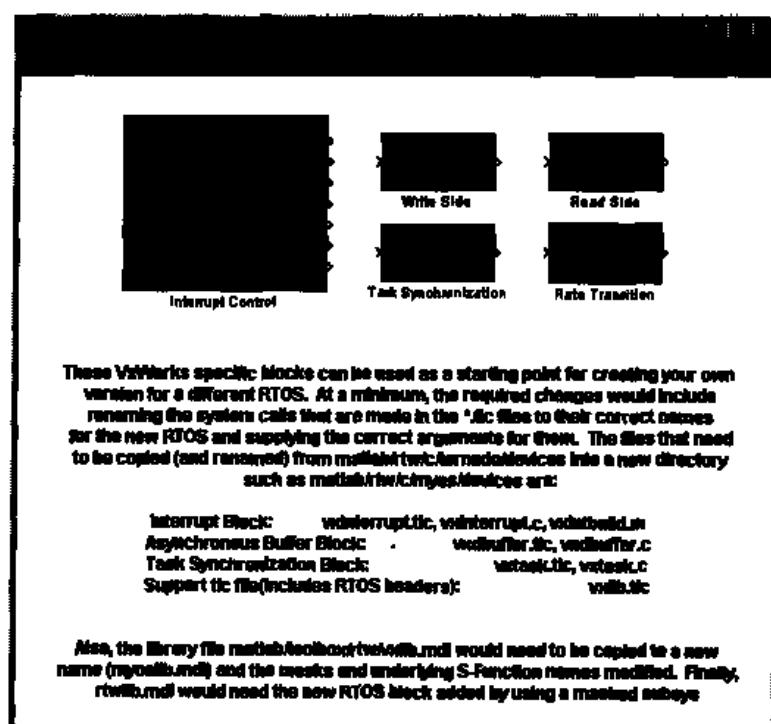


图 11-40 中断模板库

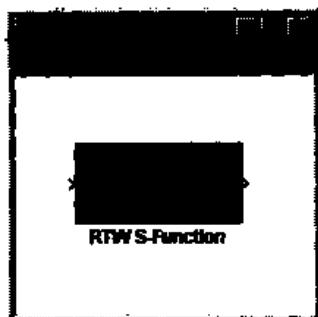


图 11-41 S 函数目标

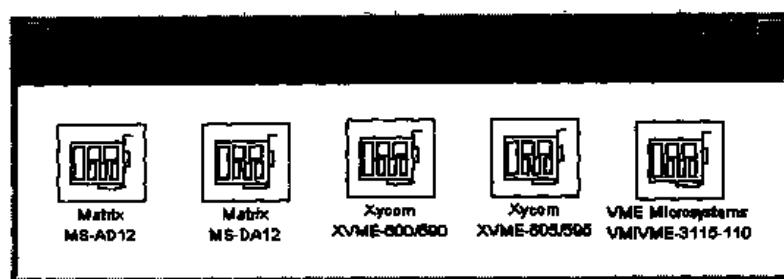


图 11-42 VxWorks 支持之 I/O 设备

### 11.7.2 自定义代码库 (Custom Code Library)

在这一节我们将以子库 Custom Code Library (自定义代码库) 为例讲述 Real-Time Workshop 函数库的应用，其它几个子库由于篇幅所限不再做更具体的介绍。

自定义代码库中包括 C 语言代码和 Ada 语言代码两种，我们这里只介绍 C 代码。在 C 代码库中又包括模型代码库 (Model Code) 和子系统代码库 (Subsystem Code) 两部分内容，我们先来看模型代码库。

#### 1. 模型代码库

在 Simulink Library Browser 中依次打开 Real-Time Workshop→Custom Code→C Custom Code→Model Code，然后在 Model Code 上通过单击鼠标右键的方式就可以打开模型代码库，如图 11-43 所示，模型代码库 (Model Code) 包含 11 个函数模块，用这些模块可以在生成的模型文件和函数中插入自定义代码。

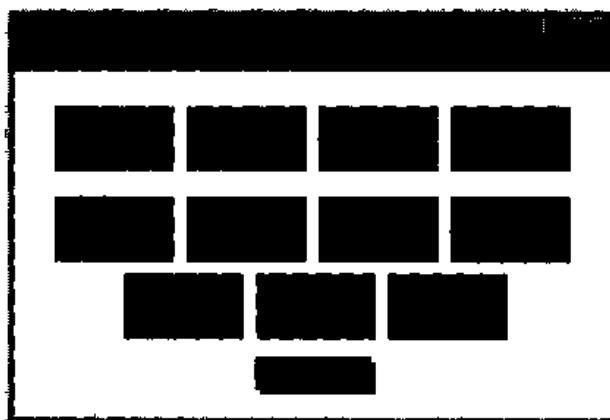


图 11-43 模型代码库

第一排的四个模块用来向文件中插入自定义代码，每个模块含有两个区域：Top of File 和 Bottom of File，在两个区域输入的代码分别被插入到文件顶部和文件底部。下面是模块和文件的对应关系：

- Header File 模块插入代码到文件 *model.h*;
- Parameter File 模块插入代码到文件 *model.prm*;
- Source File 模块插入代码到文件 *model.c*;
- Registration File 模块插入代码到文件 *model.reg*。

下面两排的七个模块把自定义代码插入到函数中，每个模块含有三个区域：Declaration

Code、Execution Code 和 Exit Code，在这三个区域中分别输入变量声明代码、执行代码和退出代码。下面是各个模块和函数的对应关系：

- Model Registration Function 模块把代码插入到文件 *model.reg* 的函数中；
- Model MdlStart Function 模块把代码插入到文件 *model.c* 的函数 *MdlStart*；
- Model MdlInitialize Function 模块把代码插入到文件 *model.c* 的函数 *MdlInitialize*；
- Model MdlTerminate Function 模块把代码插入到文件 *model.c* 的函数 *MdlTerminate*；
- Model MdlOutputs Function 模块把代码插入到文件 *model.c* 的函数 *MdlOutputs*；
- Model MdlUpdate Function 模块把代码插入到文件 *model.c* 的函数 *MdlUpdate*；
- Model MdlDerivatives Function 模块把代码插入到文件 *model.c* 的函数 *MdlDerivatives*；

下面我们举一个例子来说明这些模块的使用方法。

如图 11-44 所示，我们首先组成一个简单模型，然后从图 11-43 的模型代码库中把模块 Source File、Registration File、Model MdlStart Function、Model MdlInitialize Function、Model Registration Function 复制到该模型中，这样就组成了一个应用自定义代码的模型 rtwcustom1。

现在我们分别在这些自定义模型代码模块中输入自定义的代码。首先打开 Source File 模块，如图 11-45 所示。我们在 Top of Source File 区域输入如图内容：定义一个读文件标志 *ReadFlag* 为 1，声明一个浮点类型变量 *readrtw1* 用来存储从文件中读到的数据，由于这个区域输入的内容会被置于文件顶部，因而这里声明的变量是全局变量。

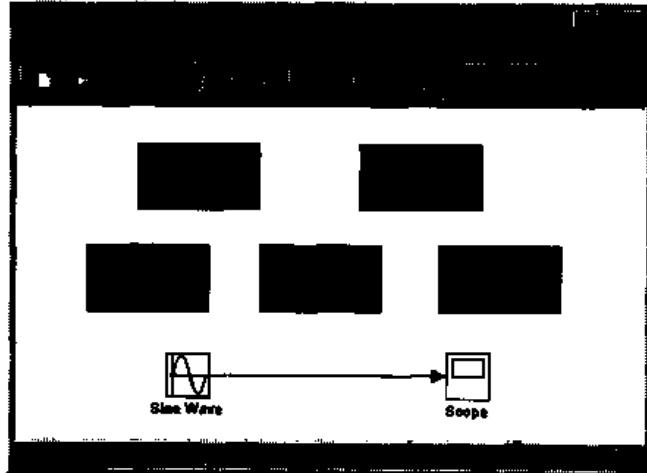


图 11-44 应用自定义代码的模型 rtwcustom1

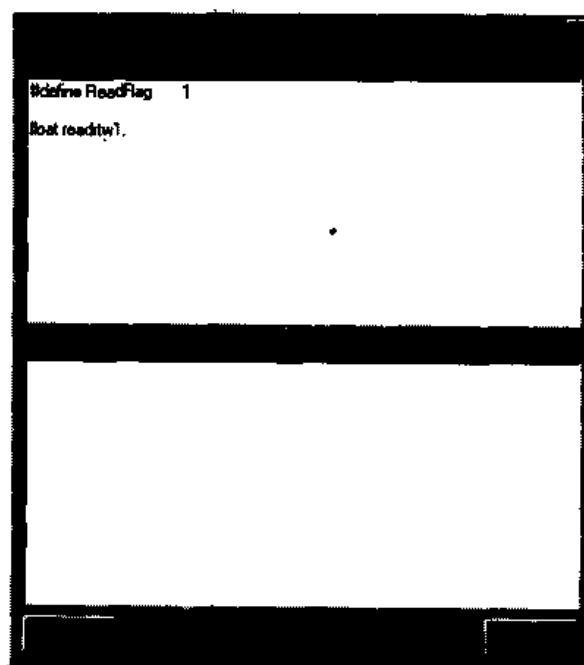


图 11-45 Source File 模块

下面打开 Model MdlStart Function 模块，如图 11-46 所示。在这个模块中可以加入变量声明、赋值、运算、退出等各种内容的代码。我们在这个模块的 Declaration Code 区域定义了一个文件指针 *fpo* 用于打开和关闭文件；在 Execution Code 区域输入代码来打开文件并读取数据，赋给在模块 Source File 中声明的变量 *readrtw1*；在 Exit Code 区域输入代码用来在退出函数时关闭文件。

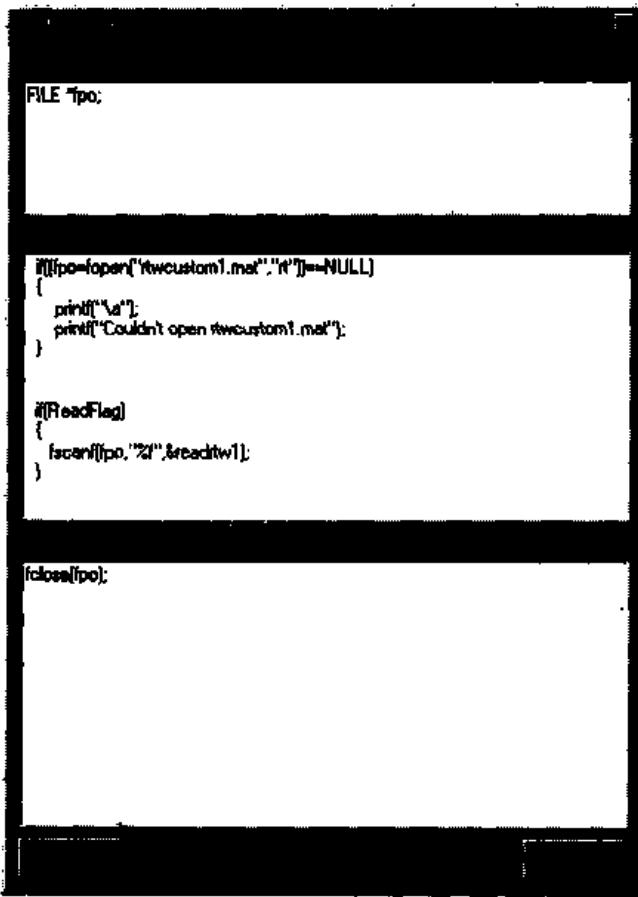


图 11-46 Model MdlStart Function 模块

前面已经讲过，模块 Registration File 和 Model Registration Function 都把代码插入到文件 rtwcustom1.reg 中，但它们插入的位置是不同的，这里我们同时把这两个模块列出以做区别。在模块 Registration File 的 Top of Registration File 区域我们输入注释代码 “//test for registration file”，如图 11-47 所示；类似地，在模块 Model Registration Function 的 Declaration Code 区域输入“//test for registration function”，如图 11-48 所示。在生成的文件 rtwcustom1.reg 中我们将会看到它们的区别。

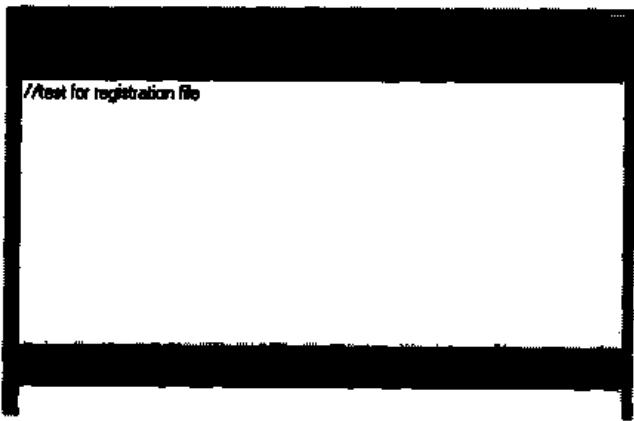


图 11-47 Registration File 模块

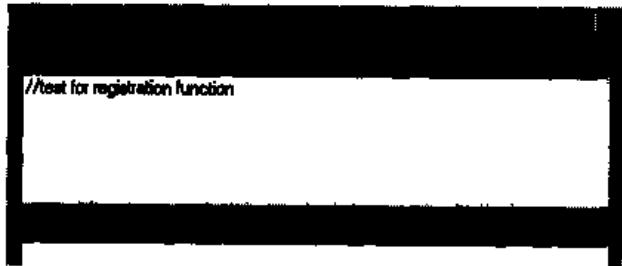


图 11-48 Model Registration Function 模块

同样，模块 Model MdIInitialize Function 也是为了说明其在文件中插入代码的位置，我

们在其 Declaration Code 区域输入 “//test for initialize function”，如图 11-49 所示。在生成的文件 rtwcustom1.c 中可以看到它插入的代码位置。

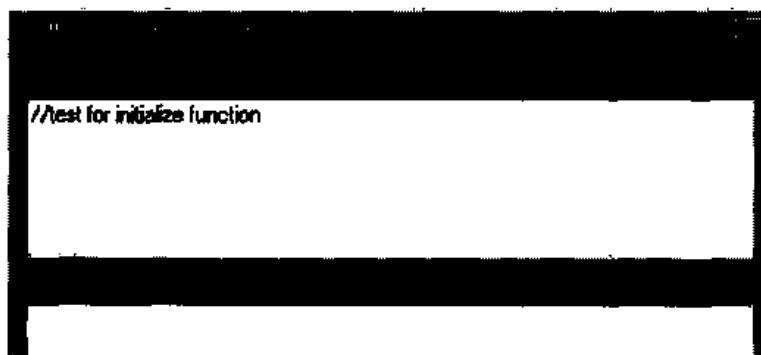


图 11-49 Model MdlInitialize Function 模块

这些模块设置好后，在 Real-Time Workshop 设置对话框中按下 Build 按钮（或在 Tools 菜单下选择 RTW Build，或按 Ctrl+b 快捷键）生成 C 代码，构造可执行程序。

首先打开文件 rtwcustom1.c，为便于理解，我们把和自定义代码相关的部分列在下面，同时把自定义的代码加黑，其余部分省略。

```

/*
 * rtwcustom1.c
 */
/* user code (top of source file) */
/* System: <Root> */
#define ReadFlag           1
float readrtw1;

void MdlStart(void)
{
    /* user code (Start function Header) */
    /* System: <Root> */
    FILE *fpo;
    int i;
    /* user code (Start function Body) */
    /* System: <Root> */
    if((fpo=fopen("rtwcustom1.mat","rt"))==NULL){   printf("\a");   printf("Couldn't
open rtwcustom1.mat"); }

    if(ReadFlag){
        fscanf(fpo,"%f",&readrtw1); }
    /* user code (Start function Trailer) */
    /* System: <Root> */
    fclose(fpo);
}

```

```

{
    /* user code (Initialize function Header) */
    /* System: <Root> */
    //test for initialize function
}
}

```

从列出的代码中我们看出了模块 Source File、Model MdlStart Function、Model MdlInitialize Function 中的自定义代码在文件中的相对位置，从而也就可以理解这几个模块以及类似模块的用法。

从另一个文件 rtwcustom1.reg 中我们来比较一下模块 Registration File 和 Model Registration Function 的区别。文件中的部分相关代码如下：

```

/*
 * rtwcustom1.reg
 */
/* user code (top of registration file) */
/* System: <Root> */
//test for registration file

/* Function to initialize sizes */
void MdlInitializeSizes(void){.....}

/* Function to initialize sample times */
void MdlInitializeSampleTimes(void){.....}

/* Function to register the model */
SimStruct *rtwcustom1(void)
{
    .....
    {
        /* user code (registration function declaration) */
        /* System: <Root> */
        //test for registration function
    }
}

```

从中可以看出，模块 Registration File 中的代码的插入位置是相对于整个 rtwcustom1.reg 文件而言的，而模块 Model Registration Function 的代码则插入到该文件用于注册模型的函数 rtwcustom1 (void) 的尾部。

通过上面这个例子我们介绍了模型代码库（Model Code）中模块的用法，下面我们简单介绍一下子系统代码库（Subsystem Code）的使用。

首先从 Simulink Library Browser 中打开子系统代码库，如图 11-50 所示。

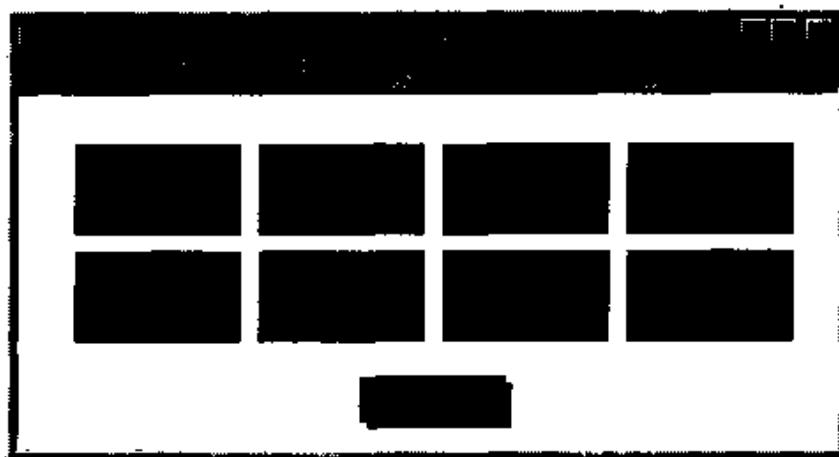


图 11-50 子系统代码库

如图中所示，子系统代码库含有 8 个模块，和模型代码库中的模块类似。这些模块也是向函数中插入代码，区别在于这些模块插入代码的位置和它们在模型中所处的位置有关。例如，我们考虑含有一个子系统的模型，如果把模块 Subsystem Start Function 置于这个子系统中，那么代码将插入到这个子系统生成的代码的 start 函数中；如果这个模块置于整个模型中，那么代码将插入到模型的代码文件 *model.c* 的 MdlStart 函数中。

这 8 个子系统代码模块都含有三个区域：Declaration Code、Execution Code 和 Exit Code，在这三个区域中分别输入变量声明代码、执行代码和退出代码。它们的用法和前面讲的模型代码类似，这里不再赘述。

# 第 12 章 符号数学工具箱

公式推导、因式分解这一类符号运算问题在工程领域和科学的研究中是很重要的一个环节。为了在以数值计算为主的 MATLAB 中增加这一项功能，1993 年，MathWorks 公司购买了著名的 Maple 软件的使用权，并以 Maple 的内核为符号运算的引擎，依赖 Maple 已有的库函数，开发了在 MATLAB 环境下实现符号处理的工具箱：符号数学工具箱（Symbolic Math Toolbox），将符号运算结合到 MATLAB 的数值运算环境。

最初的符号数学工具箱 1.0 版本是以 Maple V2 版本为基础的，可以在 MATLAB 4.0 到 4.2 版本下使用。MATLAB 5.3 中的符号数学工具箱是 2.1 版，它以 Maple V Release 5 版本为基础开发，同时使用了 MATLAB 5.0 以上版本中面向对象的概念，定义了一种新的数据类型——符号对象(symbolic object)或称为 sym 对象。

## 12.1 建立符号对象

符号对象类似于第十章介绍的 LTI 模型对象，它是一种数据结构，存储表示符号的字符串。符号数学工具箱用符号对象表示符号变量、表达式和矩阵。

建立符号对象的函数是 sym 和 syms，其中函数 syms 是 sym 的简捷方式。这两个函数是符号数学工具箱中其它函数的基础。本节分类介绍它们的几个功能。

### 12.1.1 建立符号变量、表达式和矩阵

我们先来介绍 sym 函数的几种基本用法：

(1)  $S = \text{sym}(A)$ : 建立 sym 对象 S。如果 A 是一个字符串，则 S 是符号变量或符号数；如果 A 是数值标量或矩阵，则 S 是这些数值的符号形式。这是 sym 函数最一般的形式，下面几种情况是这种调用形式的具体实现。

(2)  $x = \text{sym}'(x')$ : 建立符号变量 x，变量的值为单引号内的字符或字符串，这里是和变量名相同的字符'x'。

(3)  $x = \text{sym}'(x', 'real')$ : 设定符号变量 x 为实型变量，这时  $\text{conj}(x)$  和 x 相等。

(4)  $x = \text{sym}'(x', 'unreal')$ : 使 x 成为纯粹的形式变量，没有附加属性。一般用来清除 x 的实型属性。

(5) 类似  $\text{pi} = \text{sym}'(\text{pi}')$  和  $a = \text{sym}'(1/3)$  这种形式，建立符号数，这种方式避免了浮点数本身的近似，建立的符号数是数值的精确表示。例如，这种方式建立的符号数 pi 可以临时代替内置的同名数值函数 pi。

函数 syms 是 sym 的简捷方式，它适用于符号变量和变量的值相同的情况，该函数的用法及其与 sym 函数的关系如下：

(1)  $\text{syms arg1 arg2 ...}$  等价于： $\text{arg1} = \text{sym}'(\text{arg1}'); \text{arg2} = \text{sym}'(\text{arg2}'); \dots$

(2)  $\text{syms arg1 arg2 ... real}$  等价于： $\text{arg1} = \text{sym}'(\text{arg1}', 'real'); \text{arg2} = \text{sym}'(\text{arg2}', 'real'); \dots$

(3) `syms arg1 arg2 ... unreal` 等价于: `arg1 = sym('arg1', 'unreal');` `arg2 = sym('arg2', 'unreal');` ...

了解了函数 `sym` 和 `syms` 的用法, 我们举例说明如何用这两个函数建立符号变量、表达式和矩阵。

### 例 12.1 建立符号变量、表达式

(1) 如下命令建立了符号变量 `x` 和 `y`, 并设置附加属性为实型变量:

```
x = sym('x', 'real'); y = sym('y', 'real');
```

或采用如下简捷形式:

```
syms x y real
```

建立复数变量 `z`, 可以如下操作:

```
z = x + i*y;
```

下面的命令计算相应变量的共轭复数:

```
conj(x), conj(z), expand(z*conj(z))
```

结果为:

```
x, x-i*y, x^2 + y^2
```

可以利用下面的命令清除 `x` 的实型特性。

```
syms x unreal 或 x = sym('x','unreal')
```

(2) 要建立二次函数  $f = ax^2 + bx + c$  的符号表达式, 命令为:

```
f = sym('a*x^2 + b*x + c');
```

这个命令将符号表达式  $ax^2 + bx + c$  赋值给变量 `f`, 由于没有建立对应于表达式中 `a`、`b`、`c`、`x` 的变量, `f` 中的内容只是一个简单的字符串, 为了使 `f` 成为一个真正的符号表达式, 从而能够执行符号数学运算 (例如微积分、解方程等), 必需显式地建立这些变量。命令如下:

```
a = sym('a')
```

```
b = sym('b')
```

```
c = sym('c')
```

```
x = sym('x')
```

或者简单地用命令:

```
syms a b c x
```

### 例 12.2 建立符号矩阵

如下命令建立一个符号矩阵 `A`:

```
syms a b c
```

```
A = [a b c; b c a; c a b]
```

结果为:

```
A =
```

```
[ a, b, c]
```

```
[ b, c, a]
```

```
[ c, a, b]
```

现在用 `beta` 代替矩阵 `A(2, 3)` 位置的元素, 用变量 `alpha` 代替其中所有的元素 `b`, 命令

为：

```
syms alpha beta;
A(2,3) = beta;
A = subs(A, b, alpha)
```

结果为：

```
A =
[    a, alpha,      c]
[ alpha,      c,  beta]
[    c,      a, alpha]
```

通过这个例子，可以看到，符号对象的使用非常类似于 MATLAB 中数值对象的使用。

### 例 12.3 建立符号数

用符号变量表示黄金分割比率： $\rho = \frac{1 + \sqrt{5}}{2}$

```
rho = sym('(1 + sqrt(5))/2')
```

```
rho =
```

```
(1 + sqrt(5))/2
```

现在可以对 rho 进行各种数学运算。如：

```
f = rho^2 - rho - 1
```

```
f =
```

```
(1/2 + 1/2 * 5^(1/2))^2 - 3/2 - 1/2 * 5^(1/2)
```

```
simplify(f) % 简化结果
```

```
ans =
```

```
0
```

## 12.1.2 把数值标量或矩阵转换为符号形式

### 1. 形式转换

sym 函数的如下调用格式将数值转化为符号形式：

```
S = sym(A, flag)
```

其中，flag 有四个选项'f'、'r'、'e'和'd'对应不同的符号形式，缺省值是'r'。

#### (1) 选项'f'

十六进制浮点形式，格式为：'1.F\*2^(e)'或'-1.F\*2^(e)'，其中 F 是 13 位十六进制数组成的字符串（注意 F 前面的“1”是十进制数），e 是整数。

#### (2) 选项'r'

有理数形式，形如 p/q、p\*pi/q、sqrt(p)、2^q 和 10^q 的有理形式，这中形式有效地补偿了舍入误差，但有可能表示的浮点值和原值不相等。如果找不到简单的有理形式近似，则采用形式 p\*2^q 产生正确的浮点数，其中 p 是很大的整数。

#### (3) 选项'e'

有理数形式，同时根据 eps(浮点运算的相对精度)给出理论表达式和实际计算值的误差。

## (4) 选项'd'

十进制小数，有效数位数由 digits 函数定义，缺省有效位数是 32 位。

如果有效数位数小于 16，则会损失一些精度，例如：

```
digits(10), sym(4/3, 'd')
```

```
ans =
```

```
1.333333333
```

```
digits(20), sym(4/3,'d')
```

```
ans =
```

```
1.333333333333332593
```

当有效数位数超过 16 时，结果并不是以 3 的循环结尾，而是最接近  $4/3$  的浮点数的精确十进制形式。

## 2. 符号算术运算

在符号数学中有三种常用的算术运算类型：浮点运算，有理运算，任意精度运算（VPA）。

(1) 浮点运算是三者中速度最快的，需要最少的计算机内存，但是结果是不精确的。

MATLAB 双精度数输出的数位数由 format 命令控制，但是它的内部表示法总是采用由计算机硬件提供的八位浮点表示法。

(2) 有理运算的计算时间和内存花费都是最大的，但只要有足够的内存和足够长的计算时间，其产生的结果是最精确的。

(3) 任意精度运算在内存花费和计算精度方面，位于前面两种运算之间。采用函数 digits 来控制十进制结果的有效位数。增加有效位数将增加结果的精度，但是同时也增加了内存的需求和计算时间。函数 digits 的缺省值 32 大约对应于浮点精度。

符号数学工具箱中执行任意精度运算的函数是 vpa，其用法为：

- vpa(A)：计算 A 中的每个元素到任意位数，有效数位数是函数 digits 指定的当前值。结果中的每个元素都是一个符号表示。
- vpa(A, d)：用参数 d 来指定有效数位数，不用函数 digits 的设置。

下面我们举例说明数值到符号形式的转换以及几种类型的符号运算。

## 例 12.4 符号算术运算

```
sym(2, 'r')
```

```
ans =
```

```
'1.000000000000000'*2^(1)
```

```
sym(0.1, 'r') % 0.1 不能表示为精确的（十六进制）浮点形式，这是近似
```

```
ans =
```

```
'1.999999999999a'*2^(-4)
```

```
sym(1.33, 'r') 或 sym(1.33)
```

```
ans =
```

```
133/100
```

```
sym(1.33333, 'r') 或 sym(1.33333) % 到一定程度，会自动选择 p*2^q 的有理形式
```

```
ans =
```

```
6004784491161903*2^(-52)
```

```

sym(hilb(2))          % 采用有理形式表示矩阵中的元素
ans =
[ 1, 1/2]
[ 1/2, 1/3]
sym(pi, 'e')
ans =
pi-198*eps/359
digits(20)           % 后面的运算有效数位数都设置为20位
sym(4/3, 'd')        % 给出最接近4/3的20位浮点数, 注意和下面的vpa比较
ans =
1.333333333333332593
vpa(4/3)              % 计算4/3到第20位
ans =
1.333333333333333333
sym(pi, 'd')          % 用20位有效数字最近似地表示π
ans =
3.1415926535897931160
vpa(pi)               % 计算π到第20位
ans =
3.1415926535897932385
vpa(sym(sin(pi/6)), 10) % 指定计算到10位有效数字
ans =
.50000000000

```

### 12.1.3 建立符号数学函数

#### 1. 建立抽象函数

如果想要建立一个抽象函数  $f(x)$ , 输入命令:

```
f = sym('f(x)')
```

则  $f$  就象  $f(x)$ , 可以采用工具箱中的命令对它进行操作。

#### 例 12.5 建立一阶差分函数

```
f = sym('f(x)')
```

```
f =
```

```
f(x)
```

```
df = (subs(f,'x','x+h')-f)/'h'          % 建立一阶差分
```

```
df =
```

```
(f(x+h)-f(x))/h
```

或者采用下面的命令:

```
syms x h
```

```
df = (subs(f,x,x+h)-f)/h
```

```
df =
(f(x+h)-f(x))/h
```

在进行符号积分变换时, sym 的这个功能非常有用。

## 2. 建立一般函数

在符号数学工具箱中有两种方式建立一般符号数学函数: 利用符号表达式, 建立 M 文件。

### (1) 利用符号表达式

下面的命令利用符号表达式 r、t 和 f 表示相应的函数:

```
syms x y z
r = sqrt(x^2 + y^2 + z^2)
t = atan(y/x)
f = sin(x*y)/(x*y)
```

可以使用符号数学工具箱函数对它们进行操作。

### (2) 建立 M 文件

M-文件允许更一般地使用函数。例如, 假定要建立一个 sinc 函数求  $\sin(x)/x$ , 为了实现它, 在 sym 子目录下建立一个 M 文件:

```
function z = sinc(x)
%SINC The symbolic sinc function
if is equal(x, sym(0))
z = 1;
else
z = sin(x)/x;
end
```

可以将这个函数扩展到多变量的情况。

## 12.2 因式分解和替换

通常, 用户希望用最简单的形式书写的符号表达式。但是在很多的情况下确切地知道表达式的最简式是很困难的, 在 MATLAB 中提供了一系列的函数用于化简符号表达式和执行符号替换, 从而使得符号表达式变得相对比较简单。

### 12.2.1 因式分解和展开

下面给出了三种不同的符号表达式, 它们实际上是同一个三次多项式的三种不同的表达式:

```
syms x
f = x^3-6*x^2+11*x-6;
g = (x-1)*(x-2)*(x-3);
h = x*(x*(x-6)+11)-6;
```

三种形式中, 每一种都是针对不同情况的最佳的表达形式。f 是多项式的最常用的表达

形式，它是  $x$  的阶乘的简单线性组合； $g$  是因式形式，它给出了多项式的根，但是，如果多项式没有简单形式的根，则它的因式分解形式就不会如此方便； $h$  是嵌套的表达形式，它包含最少的数学运算，并且对于其它范围的  $x$  是最精确的。

因式分解的主要函数有：factor、collect 和 expand，下面将分别介绍这些函数。

### 1. factor 函数

factor( $f$ )：如果  $f$  是个多项式，系数为有理数，则该命令 将  $f$  表示成系数为有理数的低阶多项式相乘。如果  $f$  不能被分解成有理数，则返回的结果就是  $f$  自身。

另外，factor 也可以对符号整数进行因式分解。

#### 例 12.6 factor 函数用法

```
syms x
f = x^3-6*x^2+11*x-6;
factor(f)
ans =
(x-1)*(x-2)*(x-3)
f = x^3-6*x^2+11*x-1;
factor(f)
ans =
x^3-6*x^2+11*x-1
f = x^5-1;
factor(f)
ans =
(x-1)*(x^4+x^3+x^2+x+1)
one = '1'; % 建立符号数变量
for n = 1:4
N(n,:) = sym(one(1,ones(1,n)));
end
[N factor(N)] % 同时显示符号数和其因式分解
ans =
[ 1, 1]
[ 11, (11)]
[ 111, (3)*(37)]
[ 1111, (11)*(101)]
```

### 2. collect 函数

命令形式为：

collect( $f$ )

将  $f$  表示成关于符号变量的多项式形式，即关于符号变量合并同类项。如果有不只一个变量，则可以使用第二个参数确定对哪一个变量进行合并。

#### 例 12.7 collect 函数的使用

syms x y t

```

f = (x-1)*(x-2)*(x-3);
collect(f)
ans =
x^3-6*x^2+11*x-6
f = (1+x)*t + x*t;
collect(f)
ans =
2*x*t+t
f = (x+3)*(y+4)+x*y;
collect(f)
ans =
(2*y+4)*x+3*y+12
collect(f, y)
ans =
(2*x+3)*y+4*x+12

```

### 3. expand 函数

用于将表达式展开。

#### 例 12.8 expand 函数的应用

```

syms x y a b
f = (a+b)*(x+y);
collect(f) % 注意比较函数 collect 和 expand 的区别
ans =
(a+b)*x+(a+b)*y
expand(f)
ans =
a*x+a*y+b*x+b*y
f = exp(a+b);
expand(f)
ans =
exp(a)*exp(b)
f = sin(x+y);
expand(f)
ans =
sin(x)*cos(y)+cos(x)*sin(y)
f = sin(3*asin(x));
expand(f)
ans =
3*x-4*x^3

```

## 12.2.2 简化

### 1. simple 函数

simple 函数的目的是寻求表达式的最简形式，使之包含最少数目的字符。该函数通过将 simplify、collect、factor、horner 和其它简化函数应用于表达式并记录下结果的长度，最后 simple 从中选择最短的结果。函数的用法为：

(1) simple(S): 对符号表达式 S 应用不同的简化方法，显示简化的中间过程，包括每种简化方法和相应的简化结果，最后，simple 函数从中选择一个最短的结果作为返回值。如果 S 是矩阵，则返回结果是整个矩阵的最短表示形式，但不一定是每一个单独元素的最短表示。

(2) [r, how] = simple(S): 不显示简化的中间过程，返回最短的结果给参数 r，同时返回最短的形式所用的简化方法给参数 how。

### 2. 简化函数

#### (1) simplify 函数

simplify 函数是一个强有力的、具有普遍意义的工具。它应用于包含和式，方根，分数的乘方，指数函数，对数函数，三角函数，Bessel 函数，超越函数等的表达式，对表达式进行化简。

#### 例 12.9 simplify 函数用法

```
syms x y
f = x*(x*(x-6)+11)-6;
simplify(f)
ans =
x^3-6*x^2+11*x-6
f = (1-x^2)/(1-x);
simplify(f)
ans =
x+1
f = cos(x)^2 + sin(x)^2;
simplify(f)
ans =
1
```

#### (2) horner 函数

该函数将符号多项式转换成嵌套形式的表达式。

#### 例 12.10 horner 函数的用法

```
syms x
f = x^3-6*x^2+11*x-6;
horner(f)
ans =
x*(x*(x-6)+11)-6
```

```
f = x^2+5.5*x-2.4;
```

```
horner(f)
```

```
ans =
```

```
-12/5+(11/2+x)*x
```

### 3. simple 函数应用举例

#### 例 12.11 simple 函数的应用

```
syms x
```

```
simple(cos(x)^2+sin(x)^2) % 不带输出参数, 显示中间过程
```

```
% 下面是简化的中间过程
```

```
simplify:
```

```
1
```

```
radsimp:
```

```
cos(x)^2+sin(x)^2
```

```
combine(trig):
```

```
1
```

```
factor:
```

```
cos(x)^2+sin(x)^2
```

```
expand:
```

```
cos(x)^2+sin(x)^2
```

```
combine:
```

```
1
```

```
convert(exp):
```

```
(1/2*exp(i*x)+1/2/exp(i*x))^2-1/4*(exp(i*x)-1/exp(i*x))^2
```

```
convert(sincos):
```

```
cos(x)^2+sin(x)^2
```

```
convert(tan):
```

```
(1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2+4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2
```

```
collect(x):
```

```
cos(x)^2+sin(x)^2
```

```
% 下面是最终结果
```

```
ans =
```

```
1
```

```
[f,how] = simple(cos(x)^2+sin(x)^2) % 带输出参数, 不显示中间过程
```

```
f =
```

```
1
```

```
how =
```

```
combine
```

在一些情况下, 有必要使用 simple 函数两次 (使用两种不同的简化函数)。例如:

```
f = (1/x^3+6/x^2+12/x+8)^(1/3);
```

```
[g,how1] = simple(f)          % 第一次简化
g =
(2*x+1)/x
how1 =
radsimp
[h,how2] = simple(g)          % 进一步简化
h =
2+1/x
how2 =
collect(x)
```

### 12.2.3 替换

在 MATLAB 中，可以通过符号替换将表达式的输出形式简化，从而得到一个较简单的表达形式。有两个函数可以实现符号替换：subs 和 subexpr。

#### 1. subexpr 函数

该函数用法为：

$[Y, SIGMA] = \text{subexpr}(X, SIGMA)$  或  $[Y, SIGMA] = \text{subexpr}(X, 'SIGMA')$

通过替换表达式 X 的公用子表达式来简化 X 的表示，这些子表达式是通过命令 pretty(S) 得到的“%1、%2 ...”形式的式子。

pretty 函数把表达式表示为排版的形式，它继承了 Maple 中的%n (n 是整数) 表示法，用%n 代表在符号对象中多次出现的子表达式。

#### 例 12.12 函数 subexpr 的应用

通过替换公用子表达式对方程  $x^3+a*x+1=0$  的解进行简化：

```
syms a x
s = solve(x^3+a*x+1)          % 求解方程
s =
[1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3) - 2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3)]
[-1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3) + a/(-108+12*(12*a^3+81)^(1/2))^(1/3) +
1/2*i*3^(1/2)*(1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3) +
2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3))]
[-1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3) + a/(-108+12*(12*a^3+81)^(1/2))^(1/3) -
1/2*i*3^(1/2)*(1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3) +
2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3))]
```

这时用 pretty(s) 以排版形式表示 s，将得到如下的公用子表达式，其余部分略去，读者可以自己实验：

$\%1 := -108 + 12(12a^3 + 81)$

subexpr 函数允许象保存符号对象一样保存通用的表达子式。表达子式以列向量的形式存储，名字为 sigma。下面进行替换：

$r = \text{subexpr}(s)$

```

sigma =
-108+12*(12*a^3+81)^(1/2)
r =
[ 1/6*sigma^(1/3)-2*a/sigma^(1/3)]
[ -1/12*sigma^(1/3)+a/sigma^(1/3)+1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))]
[ -1/12*sigma^(1/3)+a/sigma^(1/3)-1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))]
» 注意：函数 subexpr 在 MATLAB 工作空间建立了一个变量 sigma。可以利用 whos 命令或 sigma 命令验证其是否存在。

```

## 2. subs 函数

该函数用法如下：

- $R = \text{subs}(S)$ ：用当前 MATLAB 工作空间中的变量替换 S 中的所有同名符号。
- $R = \text{subs}(S, \text{old}, \text{new})$ ：用 new 替换 S 中的所有符号 old。

### 例 12.13 subs 函数的应用

(1) 如下命令用工作空间中的同名变量替换符号表达式 y 中的符号：

```

a = 9;
C1 = 10;
y = dsolve('Dy = -a*y')
y =
C1*exp(-a*t)
subs(y)
ans =
10*exp(-9*t)

```

(2) 下面求矩阵 A 的特征值和特征向量，然后用 subs 和 subexpr 函数来替换其中的公共部分：

```

syms a b c
A = [a b c; b c a; c a b];
[v, E] = eig(A)
v =
[-(a+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c), -(a-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c), 1]
[ -(b-c-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c), -(b-c+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c),
1]
[ 1, 1, 1]
E =
[ (b^2-b*a-c*b-c*a+a^2+c^2)^(1/2), 0, 0]
[ 0, -(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2), 0]
[ 0, 0, b+c+a]

```

我们要替换在整个 v 和 E 中公共的表达式： $(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)$ 。首先使用 subexpr 函数，命令如下：

```
v = subexpr(v, 'S')
```

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

接着，将符号 S 代入 E：

```
E = subs(E, S, 'S')
E =
[ S, 0, 0]
[ 0, -S, 0]
[ 0, 0, b+c+a]
```

我们现在要计算  $a=10$  时的 v 值，可以如下使用 subs 函数，它将 v 中所有的 a 用 10 来替换：

```
subs(v, a, 10)
ans =
[ -(10+S-b)/(10-c), -(10-S-b)/(10-c), 1]
[ -(b-c-S)/(10-c), -(b-c+S)/(10-c), 1]
[ 1, 1, 1]
```

注意：用 S 代替的符号表达式不受上述替换的影响。也就是说，S 中的 a 没有被 10 替换。

subs 函数也可以用来将一个特殊表达式中的多个变量用多个值替换。

现在，让我们看一看 S。假定除了用 10 替换 a 以外，我们还想分别用 2 替换 b 和用 10 替换 c，解决这个问题的方法是在工作空间中给 a, b 和 c 赋值。然后，subs 用已经存在的符号变量和当前工作空间中的双精度变量计算输入的表达式的值。在这个例子中，首先给出：

```
a = 10; b = 2; c = 10;
subs(S)
ans =
8
```

使用 whos 命令看工作空间的变量：

```
whos
```

Name	Size	Bytes	Class
A	3x3	878	sym object
E	3x3	888	sym object
S	1x1	186	sym object
a	1x1	8	double array
ans	1x1	8	double array
b	1x1	8	double array

```
c           1x1           8 double array
v           3x3           982 sym object
```

从中可以看出, a、b、c 是双精度变量; 而 A、E、S、v 仍然是符号表达式。

如果想要保留 a、b、c 作为符号变量, 而且还要在 S 中改变它们的值, 使用下列命令:

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

## 12.3 符号微积分

符号数学工具箱提供了进行微积分运算的基本函数: 微分、积分、求和、极限和泰勒级数展开。本节将介绍这些功能。

### 12.3.1 符号自变量的确定

当进行数学函数运算时, 自变量的选取从上下文中很容易得到。

例如, 考虑表达式:  $f = x^n$ 、 $g = \sin(at + b)$  和  $J_v(z)$ , 如果我们要求这些表达式的导数, 而没有指定自变量, 则根据数学约定, 分别得到:  $f' = nx^{n-1}$ 、 $g' = a \cos(at + b)$  和  $h' = J_{v+1}(z)(v/z) - J_v(z)$ , 分别假设三个表达式中的自变量为 x、t 和 z, 其它的符号 n、a、b 和 v 看作常数或参数。

根据数学约定, 自变量通常都是小写字母, 并且靠近拉丁字母表的后面(如 x, y 和 z)。利用函数 `findsym` 可以找出自变量的符号。

#### 例 12.14 确定自变量

建立和前面的函数相匹配的符号表达式 f、g 和 h:

```
syms a b n nu t x z
f = x^n; g = sin(a*t + b); h = besselj(nu,z);
```

采用 `diff` 对这些表达式进行微分(关于微分和 `diff` 函数的具体讨论参见下一节):

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

上面, 我们并没有指定对应于微分的变量, 函数 `diff` 如何知道我们想要对 x 进行微分? 通过 `findsym` 命令来确定:

```
findsym(f, 1)          % 寻找第一个符号变量
ans =
x
```

类似地, `findsym(g, 1)` 和 `findsym(h, 1)` 将分别返回 t 和 z。在此, `findsym` 中的第二个参数代表了在符号对象 f 中想要寻找的符号变量个数。如果第二个参数缺省, `findsym` 将给出给定符号表达式中的所有的符号变量。

```
findsym(g)          % 给出表达式中的所有符号变量
ans =
a, b, t
```

`fndsym` 的规则：在符号表达式中的缺省符号变量为靠近小写字母  $x$  的优先，在  $x$  后面的字母优先。

### 12.3.2 微分

符号数学工具箱中的微分函数是 `diff`，其用法我们通过具体的例子来介绍。

#### 例 12.15 符号微分

首先建立一个符号表达式  $f$ :

```
syms a x
f = sin(a*x);
```

对  $f$  进行微分运算，自变量使用数学约定，在此为  $x$ :

```
df = diff(f)
df =
cos(a*x)*a
```

以  $a$  为自变量进行微分，命令形式如下:

```
dfa = diff(f, a)
dfa =
cos(a*x)*x
```

为了分别计算  $f$  对于  $x$  和  $a$  的二阶导数，使用如下命令:

```
diff(f, 2)          % 等价于 diff(f, x, 2)
ans =
-sin(a*x)*a^2
diff(f, a, 2)
ans =
-sin(a*x)*x^2
```

`diff` 函数也可以使用符号矩阵作为它的输入，在这种情况下，微分运算按矩阵的元素逐个进行。考虑下面的例子:

```
syms a x
A = [cos(a*x), sin(a*x); -sin(a*x), cos(a*x)]
A =
[ cos(a*x),  sin(a*x)]
[ -sin(a*x),  cos(a*x)]
dy = diff(A)
dy =
[ -sin(a*x)*a,  cos(a*x)*a]
[ -cos(a*x)*a, -sin(a*x)*a]
```

也可以对列向量求微分，它的自变量也是一个列向量。考虑从笛卡儿坐标系( $x, y, z$ )到球坐标系( $r, \lambda, \varphi$ )的变换，坐标系之间的关系为  $x = r \cos \lambda \cos \varphi$ ,  $y = r \cos \lambda \sin \varphi$  和  $z = r \sin \lambda$ 。注意  $\lambda$  对应于仰角或纬度，而  $\varphi$  代表方位角或经度。

使用 `jacobian` 函数计算变换的雅可比矩阵  $J$ ,  $J$  的数学表示法为:

$$J = \frac{\partial(x, y, z)}{\partial(r, \lambda, \varphi)}$$

为了满足工具箱的语法规则，使用 l、f 分别代替  $\lambda$  和  $\varphi$ ，命令为：

```
syms r l f
x = r*cos(l)*cos(f); y = r*cos(l)*sin(f); z = r*sin(l);
J = jacobian([x; y; z], [r l f]) % 计算雅可比矩阵
J =
[    cos(l)*cos(f), -r*sin(l)*cos(f), -r*cos(l)*sin(f)]
[    cos(l)*sin(f), -r*sin(l)*sin(f),   r*cos(l)*cos(f)]
[          sin(l),           r*cos(l),            0]
detJ = simple(det(J)) % 给出简化形式的雅可比矩阵的行列式值
detJ =
-cos(l)*r^2
```

注意：jacobian 函数的第一个参数必需是列向量，第二个参数必需是行向量。此外，由于 Jacobian 行列式是非常复杂的三角函数表达式，使用 simple 命令进行三角变换和简化。

### 12.3.3 极限

微分指的是：当自变量趋近某个特定的值时的函数值的变化。而导数的定义通过极限给出：

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(h)}{h}$$

假定极限存在。符号数学工具箱采用函数 limit 直接计算函数的极限。请看下面的例子。

#### 例 12.16 求极限

```
syms h n x
dc = limit( (cos(x+h)-cos(x))/h, h, 0 )
dc =
-sin(x)
limit( (1+x/n)^n, n, inf )
ans =
exp(x)
```

对于极限  $\lim_{x \rightarrow a} f(x)$ ，如果不管是从左侧趋近 a 还是从右侧趋近 a 结果是相同的，则说  $f(x)$  在 a 点的极限存在；如果从左侧趋近 a 和从右侧趋近 a 结果是不同的，则  $f(x)$  在 a 点的极限不存在。 $f(x)$  在它的奇异点没有极限。因此，下面的三个极限将产生不同的结果：

$$\lim_{x \rightarrow 0} \frac{1}{x}, \quad \lim_{x \rightarrow 0^-} \frac{1}{x}, \quad \lim_{x \rightarrow 0^+} \frac{1}{x}$$

求解程序如下：

```
limit(1/x, x, 0) % 等价于 limit(1/x)
ans =
```

```

NaN
limit(1/x, x, 0, 'left')      %左极限
ans =
-Inf
limit(1/x, x, 0, 'right')     %右极限
ans =
Inf
注意，缺省情况limit(f)等同于limit(f,x,0)。

```

#### 12.3.4 积分

如果 f 是一个符号表达式，那么如下命令：

`int(f)`

试图寻找另一个符号表达式 F，使得  $\text{diff}(F) = f$ ，也就是说 `int(f)` 返回 f 的不定积分，或者是 f 的微分的逆运算。和微分类似，`int(f, v)` 使用符号对象 v 作为积分自变量。

从表 5-1 中可以看出 `int` 函数的用法。

表 5-1 符号表达式积分

数学运算 f	MATLAB 命令
$\int x^n dx = \frac{x^{n+1}}{n+1}$	<code>int(x^n)</code> 或者 <code>int(x^n,x)</code>
$\int_0^{2\pi} \sin(2x) dx = 1$	<code>int(sin(2*x),0,pi/2)</code> 或者 <code>int(sin(2*x),x,0,pi/2)</code>
$g = \cos(at+b)$	<code>g=cos(a*t+b)</code>
$\int g(t) dt = \sin(at+b)/a$	<code>int(g)</code> 或者 <code>int(g,t)</code>
$\int J_1(z) dz = -J_0(z)$	<code>int(besselj(1,z))</code> 或者 <code>int(besselj(1,z),z)</code>

和微分相比，符号积分是一个更复杂的工作。在计算积分是会遇到很多困难。导数的逆 F，在封闭形式下也许不存在；也许会定义一个不熟悉的函数；也许可能存在，但是软件不能找到它；也许，软件可以在大型计算机上得到它，但在当前的机子上无法得到。尽管如此，在很多情况下，MATLAB 可以成功地进行符号积分。如果工具箱不能给导数的逆它将简单地返回命令，包含积分变量，而没有进行计算。

定积分也是可能的。表 5-1 中给出了定积分的计算。命令 `int(f, a, b)` 和 `int(f, v, a, b)` 分别用于求： $\int_a^b f(x) dx$  和  $\int_a^b f(v) dv$  的符号表达式。

表 5-1 中贝塞尔函数(`besselj`)的例子，可以使用 `double` 函数计算积分值的数值近似。命令为：

`a = int(besselj(1,z), 0, 1)`

```
a =
-besselj(0,1)+1
a = double(a) % 给出积分值 a 的数值近似。
a =
0.2348
```

在符号积分中很微妙的一点是各种参数的值。例如，表达式  $e^{-(kx)^2}$  显然是正的。对于任意实数 k，当 x 趋近于  $\pm\infty$  时，表达式趋近于 0。采用下面的命令对表达式进行积分：

```
syms x k;
f = exp(-(k*x)^2);
int(f, x, -inf, inf) % 等价于 int(f, -inf, inf)
Warning: Explicit integral could not be found.
> In C:\MATLAB\toolbox\symbolic\@sym\int.m at line 58
ans =
int(exp(-k^2*x^2),x = -inf .. inf)
```

然而，Maple 的内核不能预先把  $k^2$  或  $x^2$  当作正数处理，相反，它们是纯粹的符号变量，没有数学特性。因此上面的积分得不到显示的积分结果。那么如何使 k 成为一个实型变量，使得  $k^2$  为正数呢？我们可以通过 sym 定义实型变量。前面介绍了 sym 的一个非常有用的选择为“real”，允许将 k 声明为实型变量。

```
syms k real % 确保 k 为实型符号变量
int(f, x, -inf, inf)
ans =
signum(k)/k*pi^(1/2)
```

注意：k 现在在 MATLAB 工作空间中是一个符号对象，而在 Maple 的内核工作空间中是一个实型变量。命令“clear k”只能将 MATLAB 工作空间中的 k 清除掉。为了确保 k 没有正式的属性（也就是说，确保 k 是一个纯符号变量），可以采用命令“syms k unreal”将 k 从 Maple 的工作空间中清除掉。

### 12.3.5 符号求和

当符号变量的和存在时，可以使用 symsum 函数进行符号求和。

例如，p-级数：

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots$$

它的和为  $\pi^2/6$ ，而几何级数  $1 + x + x^2 + \dots$  的和为  $1/(1-x)$ ，假设  $|x| < 1$ 。

下面给出了这两个求和运算：

```
syms x k
```

```
s1 = symsum(1/k^2,1,inf) %  $1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots$ 
```

```
s2 = symsum(x^k,k,0,inf) % 1 + x + x2 + ...
s1 =
1/6*pi^2
s2 =
-1/(x-1)
```

### 12.3.6 泰勒级数

下面的命令将给出函数  $f(x)$  的 8 阶泰勒展开式:

```
syms x
f = 1/(2+sin(x));
T = taylor(f, 5)
T =
1/2-1/4*x+1/8*x^2-1/48*x^3-1/96*x^4
```

下列命令将给出  $g$  在  $x=1$  的泰勒展开式的前面 3 个非零项。

```
syms x
g = exp(sin(x))
t = taylor(g, 3, 1)
t =
exp(sin(1)) + exp(sin(1))*cos(1)*(x-1) + exp(sin(1))*(-1/2*sin(1)+1/2*cos(1)^2)*(x-1)^2
```

## 12.4 线性代数

### 12.4.1 基本算术运算

在 MATLAB 中, 符号对象的基本算术运算与双精度数的运算是相同的。包括符号对象的加、减、乘、点乘、左除(点除)、右除(点除)和乘方运算。请体会下面的例子。

#### 例 12.17 符号的基本算术运算

```
syms a b c d
A=sym('[a,b;c,d]');
B=sym('[a+b,a-b;c+d,c-d]');
A + B
ans =
[ 2*a+b,      a]
[ 2*c+d,      c]
A - B
ans =
[      -b, 2*b-a]
[      -d, 2*d-c]
```

**A \* B**

```
ans =
[ a*(a+b)+b*(c+d), a*(a-b)+b*(c-d)]
[ c*(a+b)+d*(c+d), c*(a-b)+d*(c-d)]
```

**A .\* B**

```
ans =
[ a*(a+b), b*(a-b)]
[ c*(c+d), d*(c-d)]
```

**A / B**

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

**A \ B**

```
ans =
[ 1, 1]
[ 1, -1]
```

**A ∙ B**

```
ans =
[ a/(a+b), b/(a-b)]
[ c/(c+d), d/(c-d)]
```

**A .\ B**

```
ans =
[ (a+b)/a, (a-b)/b]
[ (c+d)/c, (c-d)/d]
```

**A ^ 2**

```
ans =
[ a^2+b*c, a*b+b*d]
[ a*c+c*d, b*c+d^2]
```

**A .^ 2**

```
ans =
[ a^2, b^2]
[ c^2, d^2]
```

注意：

- (1) 在 MATLAB 中采用了运算符重载，这些双精度数运算的运算符同样也可以用于符号对象。
- (2) 符号对象的加法和减法必需满足下列原则：如果两个对象都是符号矩阵，它们必需大小相同。符号矩阵可以和符号标量进行加、减运算，运算按照数组运算规则进行。
- (3) 两个符号矩阵相乘，要求第一个矩阵的列维必需等于第二个矩阵的行维。

(4) 符号乘方运算  $S^p$ ，如果 S 为符号表达式表达式，则 p 可以为符号表达式或数值表达式；如果 S 为符号方阵，则 p 必需是整数。

#### 12.4.2 线性代数运算

符号矩阵和数值矩阵类似，也可以进行矩阵的线性代数运算，例如求逆。下面我们举例说明其特点。

##### 例 12.18 线性代数运算

###### (1) Hilbert 矩阵的运算

**format short**

**H = hilb(3)** % 生成  $3 \times 3$  的 Hilbert 矩阵，采用短格式输出数据

**H =**

1.0000	0.5000	0.3333
0.5000	0.3333	0.2500
0.3333	0.2500	0.2000

H 的元素是浮点数，是小整数的比，属于 MATLAB 双精度数组。将 H 转化为符号矩阵：

**H = sym(H)**

**H =**

[ 1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]

这样就允许对 H 进行符号运算，产生的结果对应于无穷精度的希尔伯特矩阵。  
**sym(hilb(3))** 并不是 **hilb(3)** 的浮点近似。

**inv(H)** % 符号矩阵求逆

**ans =**

[ 9, -36, 30]
[ -36, 192, -180]
[ 30, -180, 180]

**det(H)** % 求符号矩阵的行列式值

**ans =**

1/2160

我们可以使用左除算子求解线性方程组，命令为：

**b = [1 1 1]'**

**b =**

1
1
1

**x = H\b** % 求解线性方程组  $Hx = b$

```
x =
```

```
[ 3]
[-24]
[ 30]
```

这里，矩阵的逆、矩阵行列式的值和线性方程组的解都是精确解。另一方面，使用 `digits(16)`，利用下列命令：

```
V = vpa(hilb(3))
```

```
V =
```

```
[ 1., .5000000000000000, .3333333333333333]
[ .5000000000000000, .3333333333333333, .2500000000000000]
[ .3333333333333333, .2500000000000000, .2000000000000000]
```

矩阵中每一个元素中的小数点表明采用的是任意精度的算术运算。每一个算术运算的结果舍入到 16 位有效数字。对矩阵进行求逆运算时，这些误差将被矩阵的条件数放大。

```
inv(V)
```

```
ans =
```

```
[ 9.00000000000082, -36.0000000000039, 30.0000000000035]
[-36.0000000000039, 192.0000000000021, -180.0000000000019]
[ 30.0000000000035, -180.0000000000019, 180.0000000000019]
```

显示出精度损失了两位。因此，结果为：

```
det(V)
```

```
ans =
```

```
.462962962962958e-3
```

```
V\ b
```

```
ans =
```

```
[ 3.00000000000041]
[-24.0000000000021]
[ 30.0000000000019]
```

由于 H 是非奇异的，H 的零空间和列空间分别为：

```
null(H)
```

```
ans =
```

```
[ empty sym ]
```

```
colspace(H)
```

```
ans =
```

```
[ 0, 0, 1]
```

```
[ 1, 0, 0]
```

```
[ 0, 1, 0]
```

(2) 下面讨论这样一个问题：试图寻找一个数 s 位于 H(1, 1)使得 H 奇异：

```
syms s
```

```
H(1, 1) = s
```

```
H =
[ s, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

**Z = det(H)**

Z =

1/240\*s-1/270

**sol = solve(Z)**

sol =

8/9

然后, 用 sol 替换 H 中的 s:

**H = subs(H, s, sol)**

H =

```
[ 8/9, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

这时符号矩阵 H 的行列式值为 0:

**det(H)**

ans =

0

由于矩阵的行列式值为 0, 因此下面进行求逆将给出出错信息:

**inv(H) % 求矩阵 H 的逆, 给出出错信息。**

??? Error using ==> sym/inv

Error, (in inverse) singular matrix

因为 H 是奇异的, 因此矩阵的零空间和列空间是重要的。

**Z = null(H) % 矩阵 H 的零空间。**

Z =

```
[    1]
[   -4]
[ 10/3]
```

**C = colspace(H) % 矩阵 H 的列空间。**

C =

```
[    1,      0]
[    0,      1]
[-3/10,  6/5]
```

需要指出的是, 尽管 H 是奇异的, vpa(H)不是奇异的。对任何整数 d, 设置 digits(d), 然后计算 det(vpa(H))和 inv(vpa(H)), 返回行列式值大小为  $10^{-(d)}$ , 逆矩阵元素的值为  $10^d$  量级。例如, 当 d=10 时:

**digits(10)**

```

det(vpa(H))
inv(vpa(H))
ans =
-.4e-11
ans =
[ -1041666665., 4166666668., -3472222225.]
[ 4166666668., -.1666666668e11, .1388888888e11]
[ -3472222225., .1388888888e11, -.1157407408e11]

```

### 12.4.3 特征值和特征向量

在 MATLAB 中，分别使用函数 eig 计算方阵 A 的符号特征值和特征向量，用法如下：

**E = eig(A)**

**[V, E] = eig(A)**

相关的任意精度的运算的格式为：

**E = eig(vpa(A))**

**[V, E] = eig(vpa(A))**

A 的特征值是 A 的特征多项式  $\det(A - x^*I)$  等于零时的值，求 A 的特征多项式的函数为：

**poly(A)**

#### 例 12.19 特征值和特征向量

给定矩阵 H：

**H =**

**[8/9, 1/2, 1/3]**

**[1/2, 1/3, 1/4]**

**[1/3, 1/4, 1/5]**

求特征值和特征向量的命令为：

**[T, E] = eig(H)**

**T =**

```

[ 1,           1,           1]
[ -4, -7/81+1/162*12589^(1/2), -7/81-1/162*12589^(1/2)]
[ 10/3, -109/270+1/135*12589^(1/2), -109/270-1/135*12589^(1/2)]

```

**E =**

```

[ 0,           0,           0]
[ 0, 32/45+1/180*12589^(1/2), 0]
[ 0,           0, 32/45-1/180*12589^(1/2)]

```

生成矩阵 T 和 E。矩阵 T 的每一列就是 H 的特征向量。类似地，E 的对角线元素为 H 的特征值。由于矩阵是奇异的，因此它的一个特征值为 0。

如果用 s 替换  $12589^{(1/2)}$ ，则很容易了解特征向量矩阵 T 和特征值 E 的结构。

采用下面的命令：

**a = sym(12589^(1/2));**

```

syms s
Ts = subs(T, a, s)
Es = subs(E, a, s)
Ts =
[           1,           1,           1]
[           -4, -7/81+1/162*s, -7/81-1/162*s]
[      10/3, -109/270+1/135*s, -109/270-1/135*s]
Es =
[           0,           0,           0]
[           0, 32/45+1/180*s,           0]
[           0,           0, 32/45-1/180*s]

```

第一个特征值是 0, 相应的特征向量(Ts 的第一列)等同于零空间的基的第一列。其它的两个特征值是将二次方程式应用于  $x^2-64/45*x+253/2160$  得到的解。它是多项式因式分解 factor(poly(H)) 的二次方因子。

```

syms x
g = simple(factor(poly(H))/x);
solve(g)
ans =
[ 32/45+1/180*12589^(1/2)]
[ 32/45-1/180*12589^(1/2)]

```

只有当特征多项式能够被写成比较低阶的有理因式相乘时，才能给出符号表达式的形式。

#### 12.4.4 约当标准型

当用相似变换对角化矩阵时，就会产生约当标准型。

给定矩阵 A，寻找非奇异矩阵 V，使得  $\text{inv}(V)*A*V$ ，或更简洁地说，使  $J = V\backslash A*V$  尽可能接近对角形。对几乎所有的矩阵，约当标准型是特征值矩阵，它的变换矩阵的每一列就是特征向量。如果矩阵是对称的或者有相异的特征值，则经常是这样的。对于有重特征值的非对称矩阵，不能进行对角化。约当标准型在它的对角线上有特征值，但是一些对角线以上的元素是 1，而非 0。

计算矩阵的约当标准型使用函数 jordan，用法如下：

```

J = jordan(A)
[V, J] = jordan(A)

```

其中 J 是约当标准型，V 是相似变换矩阵，V 的列是 A 的一般化特征向量。

约当标准型对扰动非常敏感，对于 A 的任何改变将引起它的约当形式对角化。这使得很难用浮点算术运算可靠地计算出约当标准型。这也表明 A 必需准确地知道（例如，没有舍入误差）。它的元素必需是整数，或者是有理分式。而且，任意精度的计算 jordan(vpa(A)) 是不允许的。

#### 例 12.20 求矩阵的约当标准型

```

A = sym([12,32,66,116; -25,-76,-164,-294; 21,66,143,256; -6,-19,-41,-73])
A =
[ 12, 32, 66, 116]
[ -25, -76, -164, -294]
[ 21, 66, 143, 256]
[ -6, -19, -41, -73]
[V, J] = jordan(A)
V =
[ 4, -2, 4, 3]
[ -6, 8, -11, -8]
[ 4, -7, 10, 7]
[ -1, 2, -3, -2]
J =
[ 1, 1, 0, 0]
[ 0, 1, 0, 0]
[ 0, 0, 2, 1]
[ 0, 0, 0, 2]

```

从  $J$  看出,  $A$  有一个双特征值 1, 具有单一的约当块; 同样有一个双特征值 2, 具有单一的约当块。

矩阵只有两个特征向量  $V(:,1)$  和  $V(:,3)$ , 它们满足:

$$A \cdot V(:,1) = 1 \cdot V(:,1)$$

$$A \cdot V(:,3) = 2 \cdot V(:,3)$$

$V$  的其它列用来使特征向量一般化, 满足:

$$A \cdot V(:,2) = 1 \cdot V(:,2) + V(:,1)$$

$$A \cdot V(:,4) = 2 \cdot V(:,4) + V(:,3)$$

### 12.4.5 奇异值分解

在符号数学工具箱中, 只有任意精度的奇异值分解计算是可行的。一个原因在于由符号计算产生的公式通常都太长、太繁。如果  $A$  是一个浮点的或任意精度数字的矩阵, 那么:

$$S = svd(A)$$

计算  $A$  的奇异值, 它的精度由 digits 的当前设置决定。 $[U, S, V] = svd(A)$  将产生两个正交矩阵  $U$ 、 $V$  和对角阵  $S$ , 使得  $A = U \cdot S \cdot V'$ 。

## 12.5 方程求解

### 12.5.1 代数方程

如果  $S$  是一个符号表达式, 则:

**solve(S)**

求解当  $S=0$  时,  $S$  中缺省符号自变量的值。如果想要对特定的变量求解, 必需将特定变量作为附加参数。

#### 例 12.21 解代数方程

(1) 求解方程  $ax^2 + bx + c = 0$

```
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))]
```

如果求  $S$  中  $a$  的值, 使用下列命令:

**solve(S, a)**

ans =

```
-(b*x+c)/x^2
```

注意: 上面假定方程的形式为  $f(x) = 0$ 。如果想要求解的方程的形式为  $f(x) = q(x)$ , 必需在输入参数中用单引号将方程括起来, 其命令形式为:

```
s = solve('cos(2*x)+sin(x)=x')
```

s =

```
.75104954299588425376464666542097
```

(2) 求解方程  $x^3 - 2x^2 = x - 1$  并得到数值结果

```
s = solve('x^3-2*x^2=x-1')
```

s =

```
[1/6*(28+84*i*3^(1/2))^(1/3)+14/3/(28+84*i*3^(1/2))^(1/3)+2/3]
[-1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)+2/3 +
1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)-14/3/(28+84*i*3^(1/2))^(1/3))]
[-1/12*(28+84*i*3^(1/2))^(1/3)-7/3/(28+84*i*3^(1/2))^(1/3)+2/3 -
1/2*i*3^(1/2)*(1/6*(28+84*i*3^(1/2))^(1/3)-14/3/(28+84*i*3^(1/2))^(1/3))]
```

下面用double函数得到它的数值结果:

**double(s)**

ans =

```
2.2470 + 0.0000i
```

```
-0.8019 + 0.0000i
```

```
0.5550 - 0.0000i
```

#### 12.5.2 代数方程组

解代数方程组可以用函数 **solve** 或矩阵除法, 用 **solve** 函数时只要把方程组中的每个方程作为一个输入参数即可。

#### 例 12.22 解方程组

(1) 分别用 solve 函数和矩阵除法解线性方程组

```
syms u v x y
S = solve('x+y=u', 'x-y=v')      % 只有一个输出参数时，该参数是结构类型
S =
```

```
x: [1x1 sym]
y: [1x1 sym]
S.x          % 访问结构的域 x
ans =
1/2*v+1/2*u
S.y          % 访问结构的域 y
ans =
-1/2*v+1/2*u
```

下面采用矩阵除法求解

```
A = [1 1; 1 -1];
b = [u; v];
z = A\b
```

```
z =
[ 1/2*v+1/2*u]
[ -1/2*v+1/2*u]
```

(2) 非线性方程组

syms x y alpha

```
[x, y] = solve(x^2*y^2, x-(y/2)-alpha)      % 两个输出参数对应两个解
```

```
x =
[     0]
[     0]
[ alpha]
[ alpha]
```

```
y =
[ -2*alpha]
[ -2*alpha]
[     0]
[     0]
```

可以看出 x、y 有相同的解，这是因为第一个方程  $x^2 y^2 = 0$  有两个解： $x = \pm 0$ ， $y = \pm 0$ 。  
对方程添加如下形式的小扰动将产生四组不同的解。

```
eqs = 'x^2*y^2=1, x-1/2*y-alpha';
```

```
[x, y]=solve(eqs)
```

```
x =
[ 1/2*alpha+1/2*(alpha^2+2)^(1/2)]
[ 1/2*alpha-1/2*(alpha^2+2)^(1/2)]
```

```
[ 1/2*alpha+1/2*(alpha^2-2)^(1/2)]
[ 1/2*alpha-1/2*(alpha^2-2)^(1/2)]
y =
[ -alpha+(alpha^2+2)^(1/2)]
[ -alpha-(alpha^2+2)^(1/2)]
[ -alpha+(alpha^2-2)^(1/2)]
[ -alpha-(alpha^2-2)^(1/2)]
```

### 12.5.3 微分方程

函数 `dsolve` 可用于求常微分方程的符号解。

在符号方程中，符号表达式中包含字母“D”来代表微分运算，符号 D2、D3、... DN 分别对应于第二、第三、... 第 N 阶导数。这样  $D2y$  是  $d^2y/dt^2$  在符号数学工具箱中对应的形式。因变量是位于 D 后面的变量，缺省的自变量为 t。还可以通过在命令中加入包含自变量的参数将自变量由 t 变为其它的符号变量。

» 注意：符号变量名不包含 D。

初始条件可以在其它的方程中给定。如果初始条件未给定，则结果包含积分常数 C1，C2。

`dsolve` 的输出类似于 `solve` 的输出，用户调用 `dsolve` 时，输出变量的个数可以等于因变量的个数，或者将输出放在一个包含微分方程解的结构中。

例 12.23 解微分方程

(1) 方程  $y' = 1 + y^2$ ,  $y = y(t)$

```
dsolve('Dy=1+y^2')           % 将 y 作为因变量，缺省的自变量为 t
ans =
tan(t-C1)
```

给定初始条件  $y(0) = 1$ :

```
y = dsolve('Dy=1+y^2', 'y(0)=1')
y =
tan(t+1/4*pi)
```

» 注意：y 在 MATLAB 的工作空间，而自变量 t 并不在 MATLAB 的工作空间中。因此，命令 `diff(y, t)` 将给出出错信息。可以使用命令“`syms t`”将 t 放在工作空间中。

(2) 非线性微分方程

```
x = dsolve('(Dx)^2+x^2=1', 'x(0)=0')
```

```
x =
[ sin(t)]
[ -sin(t)]
```

(3) 二阶微分方程，给定两个初始条件

```
y = dsolve('D2y=cos(x)-y', 'y(0)=1', 'Dy(0)=0', 'x')      % 自变量是 x
```

```

y =
-1/2*sin(x)^2*cos(x)+(1/2*sin(x)*cos(x)+1/2*x)*sin(x)+cos(x)

(4) 三阶微分方程  $\frac{d^3u}{dx^3} = u$ ,  $u(0) = 1$ ,  $u'(0) = -1$ ,  $u''(0) = \pi$ 

u = dsolve('D3u=u','u(0)=1','Du(0)=-1','D2u(0)=pi','x')

u =
1/3*pi*exp(x) - 1/3*(1+pi)*3^(1/2)*exp(-1/2*x)*sin(1/2*3^(1/2)*x) +
(1-1/3*pi)*exp(-1/2*x)*cos(1/2*3^(1/2)*x)

在此使用 D3u 代表  $d^3u/dx^3$ , D2u(0) 代表  $u''(0)$ 

```

#### 12.5.4 微分方程组

函数 `dsolve` 可以用于求解有多个变量的常微分方程组，可以有或没有初始条件。

例 12.24 给定初始条件，解常微分方程组  $f' = f + g$ ,  $g' = f - g$ ,  $f(0) = 0$ ,  $g(0) = 1$

```
S = dsolve('Df = f+g', 'Dg = f-g', 'f(0) = 0, g(0) = 1')
```

```
S =
```

```

f: [1x1 sym]
g: [1x1 sym]

S.f
ans =
-1/4*2^(1/2)*exp(-2^(1/2)*t)+1/4*2^(1/2)*exp(2^(1/2)*t)
S.g
ans =
1/2*exp(2^(1/2)*t)+1/4*2^(1/2)*exp(-2^(1/2)*t)-1/4*2^(1/2)*exp(2^(1/2)*t)
+1/2*exp(-2^(1/2)*t)

```

## 12.6 积分变换

连续和离散的积分变换，在应用数学中是重要的工具。拉普拉斯变换作用于连续系统(微分方程)， $z$  变换作用于离散系统(差分方程)。类似地，傅立叶变换作用于连续函数；离散傅立叶变换(DFT)作用于有限数据采样。

符号数学工具箱提供了一些函数，能够进行解析的傅立叶、拉普拉斯和  $z$  变换，以及它们的逆变换。本节介绍这些函数的用法。

### 12.6.1 傅立叶变换和傅立叶逆变换

#### 1. 傅立叶变换

(1)  $F = \text{fourier}(f)$ : 符号表达式  $f$  的傅立叶变换。缺省的自变量为  $x$ ，缺省返回值是关于  $w$  的函数。如果  $f = f(w)$ ，`fourier` 函数返回关于  $t$  的函数。

(2)  $F = \text{fourier}(f, v)$ : 返回函数  $F$  是关于符号表达式对象  $v$  的函数，而不是缺省的  $w$ 。

(3)  $F = \text{fourier}(f, u, v)$ : 对于关于  $u$  的函数  $f$  进行变换，返回函数  $F$  是关于  $v$  的函数。

## 2. 傅立叶逆变换

(1)  $f = \text{ifourier}(F)$ : 符号表达式对象的傅立叶逆变换。缺省的自变量为  $w$ ，缺省返回是关于  $x$  的函数。如果  $F = F(x)$ ,  $\text{ifourier}$  返回关于  $t$  的函数。

(2)  $f = \text{ifourier}(F, u)$ : 返回函数  $f$  是关于符号表达式对象  $u$  的函数，而不是缺省的  $x$  的函数。

(3)  $f = \text{ifourier}(F, v, u)$ : 对于关于  $v$  的函数  $F$  进行变换，返回关于  $u$  的函数  $f$ 。

## 12.6.2 拉普拉斯变换和拉普拉斯逆变换

函数  $f(t)$  的拉普拉斯变换定义为：

$$L[f](s) = \int_0^{\infty} f(t) e^{-st} dt$$

$f(s)$  的拉普拉斯逆变换定义为：

$$L^{-1}[f](t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} f(s) e^{st} ds$$

其中， $c$  是一个实数，满足  $f(s)$  的所有奇异值所对应的  $s$  都小于  $c$ 。

$L[f]$  代表  $f$  的拉普拉斯变换， $L^{-1}[f]$  代表  $f$  的拉普拉斯逆变换。

拉普拉斯变换有许多的应用，包括求解常微分方程初值问题。

## 1. 拉普拉斯变换

(1)  $\text{laplace}(F)$ : 符号表达式  $F$  的拉普拉斯变换，缺省的自变量为  $t$ 。缺省的返回是关于  $s$  的函数  $L$ 。如果  $F = F(s)$ , 则  $\text{laplace}$  函数返回关于  $t$  的函数。

(2)  $\text{laplace}(F, t)$ : 返回关于  $t$  的函数  $L$ ，而不是关于缺省的  $s$  的函数。 $L$  作为符号表达式返回。

(3)  $\text{fourier}(F, w, z)$ : 对  $w$  的函数  $F$  进行变换，返回关于  $z$  的函数  $L$ 。

## 2. 拉普拉斯逆变换

(1)  $F = \text{ilaplace}(L)$ : 符号表达式对象  $L$  的逆变换，缺省的自变量是  $s$ 。缺省返回值是关于  $t$  的函数。如果  $L = L(t)$ ,  $\text{ilaplace}$  函数返回关于  $x$  的函数。

(2)  $F = \text{ilaplace}(L, y)$ : 返回关于  $y$  的函数，而不是缺省的  $t$ 。 $y$  是符号表达式对象。

(3)  $F = \text{ilaplace}(L, y, x)$ : 对于关于  $y$  的函数  $L$  进行变换，返回关于  $x$  的函数  $F$ 。

## 12.6.3 z 变换和逆 z 变换

函数的  $f(n)$  的  $z$  变换定义为：

$$Z[f](z) = \sum_{n=0}^{\infty} f(n) z^{-n}$$

记号  $Z[f]$  表示  $f$  在  $z$  处的  $z$  变换。令  $R$  为正数，使得函数  $g(z)$  在圆  $|z| = R$  的外面是解析

的，则函数  $g$  在  $n$  处的逆  $z$  变换 (IZT) 定义为：

$$Z^{-1}[g](n) = \frac{1}{2\pi i} \oint_{|z|=R} g(z) z^{n-1} dz, \quad n = 1, 2, \dots$$

记号  $Z^{-1}[f]$  表示  $f$  在  $n$  处的逆变换。符号数学工具箱中的命令 `ztrans` 和 `iztrans` 分别用于求符号表达式的  $z$  变换和逆  $z$  变换。 $z$  变换经常用于求解差分方程。

### 1. $Z$ 变换

(1)  $F = \text{ztrans}(f)$ : 符号表达式  $f$  的  $z$  变换，缺省的自变量为  $n$ ，缺省返回关于  $z$  的函数。如果  $f = f(z)$ ，则 `ztrans(f)` 返回关于  $w$  的函数。

(2)  $F = \text{ztrans}(f, w)$ : 返回关于符号变量  $w$  的函数  $F$ ，而不是关于缺省的符号变量  $z$  的。

(3)  $F = \text{ztrans}(f, k, w)$ : 对关于  $k$  的符号变量作  $z$  变换，返回关于符号变量  $w$  的函数  $F$ 。

### 2. 逆 $Z$ 变换

(1)  $f = \text{iztrans}(F)$ : 关于符号表达式对象  $F$  的逆  $z$  变换，缺省的自变量为  $z$ 。缺省返回是关于  $n$  的函数。如果  $F = F(n)$ ，`iztrans` 返回关于  $k$  的函数。

(2)  $f = \text{iztrans}(F, k)$ : 返回关于  $k$  的函数  $f$ ，而不是关于  $n$  的函数。在此  $k$  是符号表达式对象。

(3)  $f = \text{iztrans}(F, w, k)$ :  $F$  是关于  $w$  的函数，而不是隐含的 `findsym(F)` 确定的，返回关于  $k$  的函数。

## 12.7 符号函数的图形

### 12.7.1 绘制符号函数的图形

在符号数学工具箱 2.1 版本中增强了符号函数的绘图功能，可以对多种形式的符号函数直接绘制不同表现形式的图形。这些绘图函数及其绘制的曲线或曲面类型如表 12-2 所示。

表 12-2 符号绘图函数

函 数	绘制的曲线或曲面类型
<code>ezplot</code>	$y = f(x)$ ; $f(x, y) = 0$ ; $x = f(t)$ 和 $y = g(t)$ 构成的参数曲线
<code>ezpolar</code>	$r = f(\theta)$
<code>ezplot3</code>	$x = f(t)$ , $y = g(t)$ , $z = h(t)$ 构成的参数曲线
<code>ezsurf</code> , <code>ezsurf</code> , <code>ezmesh</code> , <code>ezmeshc</code>	$z = f(x, y)$ ; $x = f(s, t)$ , $y = g(s, t)$ , $z = h(s, t)$ 构成的参数网格或表面图
<code>ezcontour</code> , <code>ezcontourf</code>	绘制上面两种曲面的等高线图

下面我们举例说明这些函数的作用，函数的详细用法请参见帮助。

#### 例 12.25 绘制函数曲线

下列命令绘制平面曲线，结果如图 12-1。

```
syms x y t
subplot(1,2,1), ezplot(x^2-y^4)      % 绘制曲线  $x^2 - y^4 = 0$ ，缺省横坐标范围  $[-2\pi, 2\pi]$ 
x = t; y = t^3;                         % 以参数形式给定曲线
```

```
subplot(1,2,2), ezplot(x, y, [-2, 2]) % 绘制参数曲线, 指定参数t的范围[-2, 2]
```

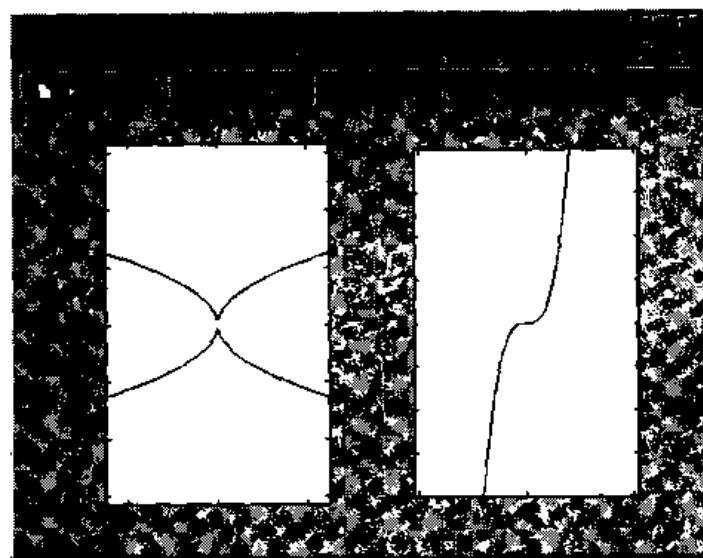


图 12-1 平面曲线

下面的命令绘制极坐标曲线, 结果如图 12-2 所示:

```
r = sin(s)+cos(s)+1;
ezpolar(r) % 使用缺省取值范围[0, 2π]
```

下面的命令绘制以参数形式给出的三维曲线, 如图 12-3 所示:

```
ezplot3('sin(t)', 'cos(t)', 't', [0, 8*pi]) % 参数 t 的取值范围[0, 8π]
```

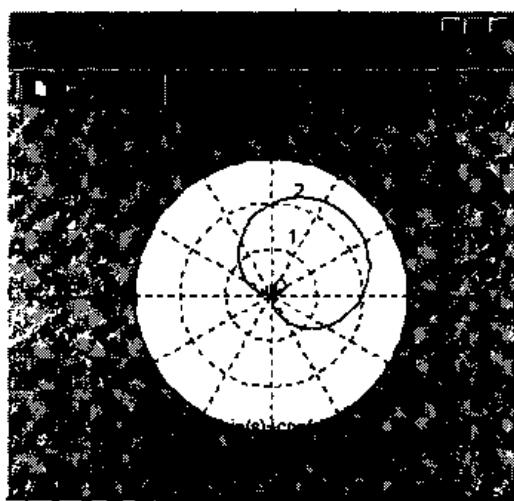


图 12-2 极坐标曲线

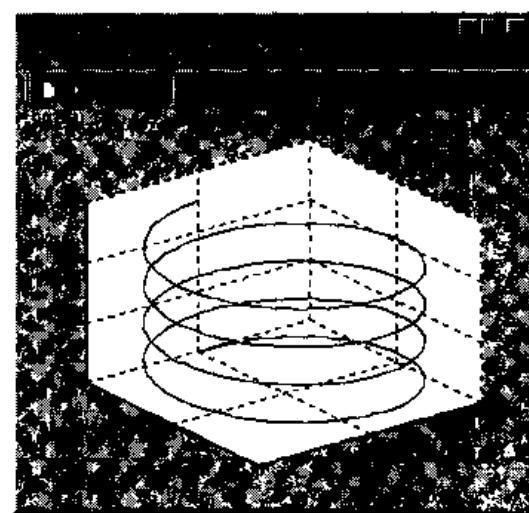


图 12-3 三维曲线

#### 例 12.26 绘制网格图和表面图

下列命令分别绘制函数  $z = x^2 - y^2$  的网格图和表面图, 使用缺省取值范围:  $-2\pi < x < 2\pi$ ,  $-2\pi < y < 2\pi$ , 结果如图 12-4、12-5 所示, 其中表面图带有等高线:

```
syms x y z t
ezmesh(x^2-y^2) % 网格图, ezmeshc 绘制带等高线的网格图
ezsurf(x^2-y^2) % 带等高线的表面图, ezsurf 绘制一般表面图
```

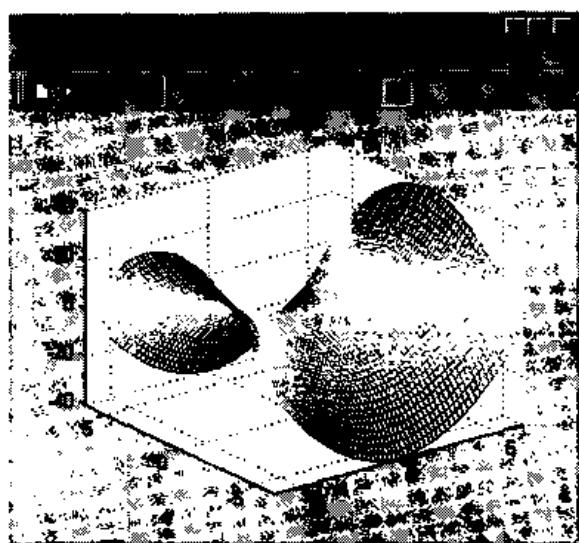


图 12-4 网格图

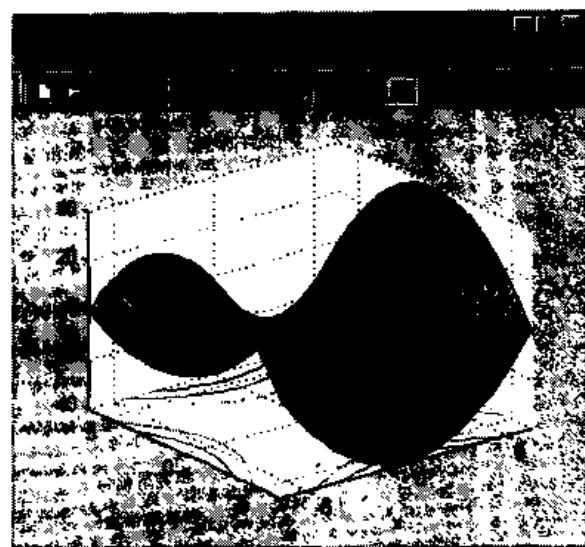


图 12-5 带等高线的表面图

### 例 12.27 绘制等高线

如下命令绘制等高线和填充等高线，结果如图 12-6 和 12-7 所示：

```
syms x y
z = peaks(x, y)      % 函数名重载，用于数值运算的函数可以用于符号运算
z =
3*(1-x)^2*exp(-x^2-(y+1)^2)-(2*x-10*x^3-10*y^5)*exp(-x^2-y^2)-1/3*exp(-(x+1)^2-y^2)
ezcontour(z, [-3, 3])    % 等高线，指定 x 和 y 的范围是[-3, 3]
ezcontourf(z, [-3, 3])   % 填充颜色的等高线
```

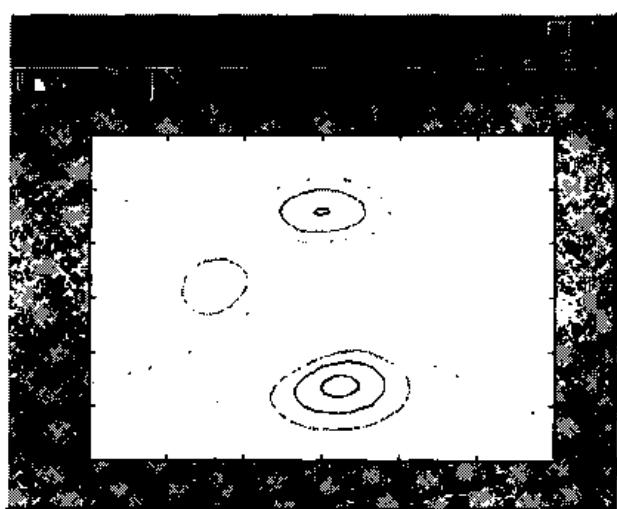


图 12-6 等高线

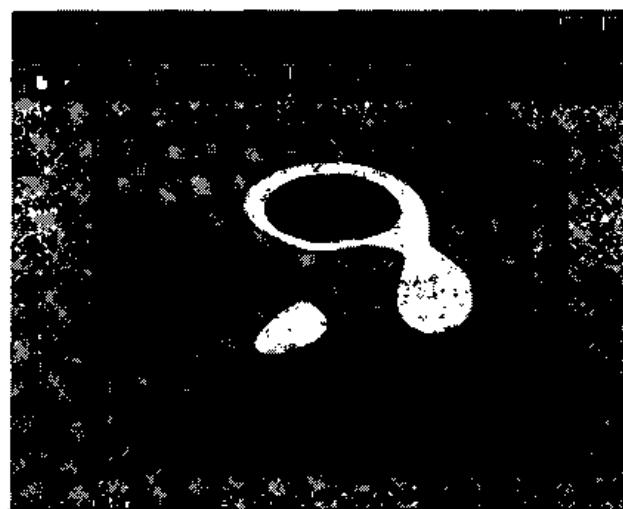


图 12-7 填充的等高线

### 12.7.2 可视化函数计算器

符号数学工具箱中的函数 `funtool` 提供了一个可视化的函数计算器，以图形方式处理和显示单变量函数。调用计算器的方式是在 MATLAB 命令窗口输入命令 `funtool`，该计算器的界面如图 12-8 所示。

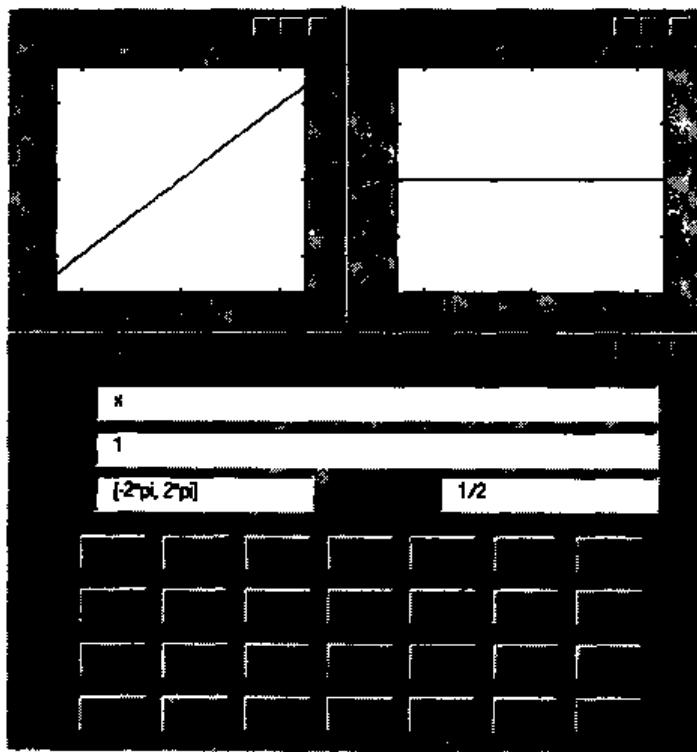


图 12-8 函数计算器界面

调用函数计算器时打开三个窗口，下面的窗口是计算器操作面板，上面两个窗口显示两个一元函数  $f$  和  $g$  的曲线，在对函数进行处理的过程中，上面的窗口中显示处理的结果。

计算器操作面板中上面两行是两个函数  $f$  和  $g$  的表达式，下面一行的  $x$  是函数自变量的取值范围， $a$  是指定的常数，在对函数处理时可能会用到。

操作面板中的按钮用于对当前的函数进行各种操作，按钮上的提示比较直观地给出了这些按钮的功能，按下按钮即完成相应的操作，就象使用一般计算器那样简单。

如果需要详细了解这些按钮的功能，可以使用最后一行按钮中的“Help”查看相应的帮助信息。

## 12.8 访问 Maple 函数

Maple 具有很强大的符号计算功能和丰富的经典应用数学函数，这些资源以库的形式提供给 MATLAB，由于不是 M 文件，因而不能直接在 MATLAB 中使用，MATLAB 为此提供了专用的函数作为接口，通过这些函数访问 Maple 的内核，从而可以很容易地调用 Maple 的绝大多数功能。

### 12.8.1 访问 Maple 中的函数

利用符号数学工具箱中的 `maple` 函数可以直接访问 Maple 中的任何函数。这个函数采用符号对象、字符串、双精度数作为输入。返回与输入相对应的符号对象、字符串和双精度数。也可以使用 `maple` 函数调试用户编写的符号数学程序。

`maple` 函数的调用格式如下：

- (1) `maple('statement')`: 把语句输送到 Maple 内核，并返回该语句的执行结果。
- (2) `maple('function', arg1, arg2, ...)`: 接收任何 Maple 函数的函数名'function'和相关输入参数，在需要时可以把参数转换为字符表达式，并按照指定的参数调用函数。如果输入参数是符号对象，则返回值也是符号对象，否则返回字符类型的结果。
- (3) `[r, status] = maple(...)`: 可选输出参数用于返回警告/错误信息，当执行成功时，`r` 是返回值，`status` 为 0；当执行失败时，`r` 是相应的警告/错误信息，`status` 是一个正整数。
- (4) `maple('traceon')` (或 `maple trace on`)：显示其后 Maple 语句的调用过程。用 `maple('traceoff')` (或 `maple trace off`) 来关闭这一功能。

这里，我们还需要了解另一个重要的函数：`mhelp`，该函数在 MATLAB 环境中获取 Maple 函数的详细帮助信息，用 MATLAB 中的 `help` 函数无法得到 Maple 函数的帮助。对于一些 MATLAB 和 Maple 中重名的函数，`help` 和 `mhelp` 分别得到 MATLAB 和 Maple 中的函数帮助，它们的内容是不同的。读者可以自己比较下面两条命令的执行结果：

```
mhelp maple
```

```
help maple
```

#### 例 12.28 访问 Maple 函数

- (1) 下列三条命令的作用相同，计算  $\pi$  到 20 位，注意函数 `maple` 的不同用法：

```
maple('evalf(Pi,20)')
```

```
maple evalf Pi 20
```

```
maple('evalf','Pi',20)
```

```
ans =
```

```
3.1415926535897932385
```

- (2) 下列命令显示了符号数学工具箱中的函数 `factor` 执行时调用 Maple 函数的过程：

```
syms x
```

```
f = [x^3-1; x^4-1];
```

```
maple traceon
```

```
factor(f)
```

```
statement:
```

```
map(ifactor, array([[x^3-1],[x^4-1]]));
```

```
result:
```

```
Error, (in ifactor) invalid arguments
```

```
statement:
```

```
map(factor, array([[x^3-1],[x^4-1]]));
```

```
result:
```

```
matrix([(x-1)*(x^2+x+1)], [(x-1)*(x+1)*(x^2+1)])
```

```
ans =
```

```
[ (x-1)*(x^2+x+1)]
```

```
[ (x-1)*(x+1)*(x^2+1)]
```

从调用过程我们看到，`factor` 函数首先调用 Maple 中的整数分解语句，判断输入参数是否是可分解的整数，如果返回错误信息，则继续调用表达式分解的语句进行分解。

### (3) 调用 Maple 中的应用数学函数

下列命令调用 Maple 中特有的应用数学函数 (MATLAB 中没有该函数), 请读者注意函数返回值的类型, 以便和下一节将要介绍的 mfun 函数比较:

```

syms m n
a = maple('binomial', m, n)          % binomial 为 Maple 中的二项式函数
a =
binomial(m,n)
b = maple('binomial', 4, 0.5)
b =
256/35/pi
c = maple('binomial', 6, 2)
c =
15
whos a b c

```

Name	Size	Bytes	Class
a	1x1	150	sym object
b	1x9	18	char array
c	1x2	4	char array

Grand total is 25 elements using 172 bytes

## 12.8.2 特殊数学函数

在符号数学工具箱中, 有 50 多个经典应用数学的特殊函数可供使用, 其中绝大部分在 MATLAB 中不能直接求解。这些函数可以通过 mfun 函数访问, 其用法为:

```
mfun('function', par1, par2, par3, par4)
```

该函数以数值方式计算 Maple 中的特殊数学函数'function'的值, 函数的参数由 par1, par2, par3, par4 指定, 最多可以指定四个参数。

符号数学工具箱中提供了函数 mfunlist 可以列出 Maple 中的特殊数学函数名称及其功能, 这些函数的用法等更加详细的信息可以用命令 “mhelp + 函数名” 来获取。

注意上一节介绍的函数 maple 也可以调用这些数学函数, 它和函数 mfun 的区别在于计算的方式不同: 函数 maple 采用符号计算, 计算结果是符号对象或字符型数据; 而函数 mfun 采用 16 位精度的数值计算, 结果是双精度型数值, 出现奇异值则返回为 NaN。

采用 mfun 可以调用下面这些特殊数学函数:

Airy 函数	Binomial 系数	Riemann Zeta 函数
Bernoulli 数和多项式	欧拉数和多项式	Harmonic 函数
指数积分	对数积分	正弦和余弦积分
双曲正弦和余弦积分	Shifted 正弦积分	Fresnel 正弦和余弦积分
Dawson 积分	误差函数	补充误差函数及重复积分
Gamma 函数	对数 Gamma 函数	不完全 Gamma 函数
Digamma 函数	Polygamma 函数	广义超几何函数

Bessel 函数

补模完全椭圆积分

Lambert W 函数

不完全椭圆积分

Beta 函数

Dirac Delta 函数

完全椭圆积分

Dilogarithm 积分

Heaviside 函数

下面列出的是只能用于扩展符号数学工具箱的正交多项式:

Gegenbauer 多项式

广义 Laguerre 多项式

第一类和第二类 Chebyshev 多项式

Hermite 多项式

Legendre 多项式

Laguerre 多项式

Jacobi 多项式

☞ 注意: Maple 函数区分大小写。

**例 12.29 特殊数学函数的计算**

下列命令根据不同的输入参数计算二项式函数的值, 请读者与例 12.28 (3) 比较:

**mfund('binomial', m, n)** % mfun 不能用于符号运算, 因而将给出错误信息

??? Error using ==&gt; isnan

Function 'isnan' not defined for variables of class 'sym'.

Error in ==&gt; E:\MATLABR11\toolbox\symbolic\mfund.m

On line 43 ==&gt; nans = find(isnan(x));

**mfund('binomial', 4, 0.5)** % 计算函数在[4, 0.5]处的值

ans =

2.3282

**mfund('binomial', 6, 2)** % 结果和例 12.28 中相同, 但数据类型不同

ans =

15

**mfund('Shi', 1:3)** % 分别计算[1 2 3]处的双曲正弦积分

ans =

1.0573 2.5016 4.9734

## 12.9 扩展符号数学工具箱

扩展符号数学工具箱 (Extended Symbolic Math Toolbox) 允许用户访问所有的非图形 Maple 软件包、Maple 编程特征和 Maple 程序。这样扩展工具箱提供了访问由 Maple 语言编写的大量数学软件的方法。

关于 Maple 编程特征请读者参阅 Maple 手册。

本节将介绍如何加载 Maple 软件包和如何使用 Maple 程序。

### 12.9.1 Maple 软件包

用户可以在 MATLAB 中使用的 Maple 软件包可以用如下命令显示出来:

**mhelp index[packages]**

显示出的可用 Maple 软件包如下:

**Index of descriptions for packages of library functions**

Description:

- The following packages are available:

<code>DEtools##DEtools</code>	differential equations tools
<code>Domains##Domains</code>	create domains of computation
<code>GF##GF</code>	Galois Fields
<code>GaussInt##GaussInt</code>	Gaussian Integers
<code>LREtools##LREtools</code>	manipulate linear recurrence relations
<code>Matlab##Matlab</code>	Matlab Link
<code>algcurves##algcurves</code>	Algebraic Curves
<code>codegen##codegen</code>	Code Generation
<code>combinat##combinat</code>	combinatorial functions
<code>combstruct##combstruct</code>	combinatorial structures
<code>diffalg##diffalg</code>	differential algebra
<code>diffforms##diffforms</code>	differential forms
<code>finance##finance</code>	financial mathematics
<code>genfunc##genfunc</code>	rational generating functions
<code>geom3d##geom3d</code>	Euclidean three-dimensional geometry
<code>geometry##geometry</code>	Euclidean geometry
<code>grobner##grobner</code>	Grobner bases
<code>group##group</code>	permutation and finitely-presented groups
<code>inttrans##inttrans</code>	integral transforms
<code>liesymm##liesymm</code>	Lie symmetries
<code>linalg##linalg</code>	Linear algebra
<code>logic##logic</code>	Boolean logic
<code>networks##networks</code>	graph networks
<code>numapprox##numapprox</code>	numerical approximation
<code>numtheory##numtheory</code>	number theory
<code>orthopoly##orthopoly</code>	orthogonal polynomials
<code>padic##padic</code>	p-adic numbers
<code>plots##plots</code>	graphics package
<code>plottools##plottools</code>	basic graphical objects
<code>powseries##powseries</code>	formal power series
<code>process##process</code>	(Unix)-multi-processing
<code>simplex##simplex</code>	linear optimization
<code>stats##stats</code>	statistics
<code>student##student</code>	student calculus
<code>sumtools##sumtools</code>	indefinite and definite sums
<code>tensor##tensor</code>	tensor computations and General Relativity
<code>totorder##totorder</code>	total orders on names

- For information see ?package where package is from the above list. This will give a list of the functions available in the package. To cause all functions in a package to be defined in the session, do: with(package);
- For information on a particular package function, see ?package,function

See Also:

`with##with, examples,binarytree##examples,binarytree`

要得到关于某个软件包的详细信息，以 `inttrans`（积分变换软件包）为例，可以使用如下命令：

**mhelp inttrans**

将返回该软件包的内容和基本用法。

要加载 Maple 软件包，仍以 `inttrans` 为例，使用如下命令：

**maple('with(inttrans)')**

将显示如下内容：

```
Warning: Warning, new definition for hilbert      % 警告hilbert函数被重新定义
ans =
```

```
[addtable, fourier, fouriercos, fouriersin, hankel, hilbert, invfourier, invhilbert, invlaplace,
inv mellin, laplace, mellin, savetable]
```

这是在 `inttrans` 软件包中的函数名列表。现在，这些函数已经载入 Maple 的工作空间，可以象调用通常的 Maple 函数一样调用它们。

### 12.9.2 Maple 程序

用户可以通过扩展符号数学工具箱访问 Maple 程序。

下面的 Maple 程序是采用由 Richard Brent 基于高斯几何算术平均算法导出的方法，采用符号运算或任意精度运算方法计算 $\pi$ ，函数的输入参数  $n$  表示算法的阶数。

```
pie := proc(n)
# pie(n) takes n steps of an arithmetic-geometric mean
# algorithm for computing pi. The result is a symbolic
# expression whose length roughly doubles with each step.
# The number of correct digits in the evaluated string also
# roughly doubles with each step.
# Example: pie(5) is a symbolic expression with 1167
# characters which, when evaluated, agrees with pi to 84
# decimal digits.
local a,b,c,d,k,t;
a := 1;
b := sqrt(1/2);
c := 1/4;
t := 1;
for k from 1 to n do
```

```

d := (b-a)/2;
b := sqrt(a*b);
a := a+d;
c := c-t*d^2;
t := 2*t;
od;
(a+b)^2/(4*c);
end;

```

假定这个 Maple 程序的源代码文件名为“pie.src”。使用扩展符号数学工具箱的 MATLAB 命令：

```
procread('pie.src')
```

读入给定的文件，删除注释行和换行符，将结果字符串传递给 Maple(MATLAB 的 ans 变量这时就包含一个字符串代表 pie.src 文件)。

现在可以在 maple 函数中调用 pie 函数，例如：

```
p = maple('pie', 3)
```

返回一个字符型变量 p：

```

p =
1/4*(1/8+1/16*2^(1/2)+1/8*2^(3/4)+1/4*2^(1/2)*((1/2+1/4*2^(1/2))*2^(3/4))^(1/2)+1/2*2^(1/2)*((1/4+1/8*2^(1/2)+1/4*2^(3/4))*2^(1/2)*((1/2+1/4*2^(1/2))*2^(3/4))^(1/2))^(1/2))^2/(1/4-(1/4*2^(1/2)-1/2)^2-2*(1/4*2^(3/4)-1/4-1/8*2^(1/2))^2-
4*(1/4*2^(1/2)*((1/2+1/4*2^(1/2))*2^(3/4))^(1/2)-1/8-1/16*2^(1/2)-1/8*2^(3/4))^2)

```

下面我们把字符型变量 p 转换为符号对象，并用任意精度计算其数值：

```
s = sym(p)
```

```
vpa(s) % 任意精度运算，缺省有效数位数32位
```

```
ans =
```

```
3.1415926535897932382795127748020
```

当 Maple 把一个源程序（ASCII 码文本）加载到 Maple 的工作空间，Maple 将它编译（转换）成内部格式，用户可以随后使用函数 maple 将程序以内部格式储存起来。这样做的优点是当下一次加载它时，避免了再次编译，加快了运行。

例如，可以将前面的例子中 pie.src 程序转换成预编译 Maple 程序，使用下面的命令：

```
clear maplemex
```

```
procread('pie.src')
```

```
maple('save(''pie.m'')');
```

其中，命令 clear maplemex 重新设置 Maple 工作空间为初始状态。因为 Maple 的 save 命令储存工作空间的所有变量，而我们想要除去无关的变量。

注意：pie.m 两侧的符号是键盘主区左上角（数字 1 左侧）的键表示的引号，不能用常用的引号代替。

为了将一个预编译的程序读入后来的 MATLAB 程序，输入命令：

```
maple('read','pie.m');
```

该函数仍然是 ASCII 文本形式，可以使用 `maple` 函数访问这个函数。

`p = maple('pie', 3)`

结果和上面相同。

» 注意：预编译 Maple 过程也具有.m 的扩展名，但和 MATLAB 的 M 文件完全不同，读者可以使用文本编辑器查看其内容，加以对比。

# 附录 A MATLAB 函数命令索引

表 A-1 MATLAB 函数命令索引

函数/命令	功 能	函数/命令	功 能
<code>abs()</code>	绝对值函数	<code>acos()</code>	反余弦函数
<code>acosh()</code>	反双曲余弦函数	<code>acot()</code>	反余切函数
<code>acoth()</code>	反双曲余切函数	<code>acsc()</code>	反余割函数
<code>acsch()</code>	反双曲余割函数	<code>airfoil()</code>	NASA 翼面稀疏矩阵显示
<code>all()</code>	测试向量中所有元素是否为真	<code>angle()</code>	相角函数
<code>any()</code>	测试向量中是否有为真元素	<code>ans()</code>	返回最新结果
<code>arith()</code>	MATLAB 的各种算术运算符信息	<code>asec()</code>	反正割函数
<code>asech()</code>	反双曲正割	<code>asin()</code>	反正弦函数
<code>asinh()</code>	反双曲正弦	<code>atan()</code>	反正切函数
<code>atan2()</code>	四个象限内反正切	<code>atanh()</code>	反双曲正切
<code>audioread()</code>	读声音文件	<code>auwrite()</code>	写声音文件
<code>axes()</code>	坐标轴任意形式的设定	<code>axis()</code>	坐标轴标度设定
<code>balance()</code>	改进特征精度的均衡变换	<code>bar()</code>	绘制条形图
<code>bench()</code>	MATLAB 测试基准问题	<code>blanks()</code>	设置一个由空格组成的字符串
<code>bone</code>	带有蓝色的灰度颜色表	<code>break</code>	中断循环执行的语句
<code>brighten()</code>	使图形色调变亮	<code>bucky</code>	Buckminster Fuller 拱形演示
<code>caxis()</code>	伪颜色坐标轴设定	<code>cd</code>	改变当前的工作目录
<code>cdf2rd()</code>	复块对角矩阵到实块对角阵转换	<code>ceedit</code>	设置命令行编辑与回调的参数
<code>ceil()</code>	对+∞ 方向取整数	<code>census</code>	2000 年美国人口普查预测
<code>chol()</code>	Cholesky 分解	<code>cla</code>	清除当前坐标轴
<code>clabel()</code>	等高线剖面标志	<code>clc</code>	清除命令窗口显示
<code>clear</code>	删除内存中的变量与函数	<code>clf</code>	清除当前图形窗口
<code>clock</code>	时钟	<code>close</code>	关闭图形窗口
<code>clommd()</code>	最小列的阶次	<code>colon</code>	冒号表达式的帮助信息
<code>colormap()</code>	设定颜色可查表	<code>colormenu</code>	颜色表演示
<code>colperm()</code>	由非零数据的计数来排列各列	<code>comet()</code>	彗星状轨迹绘制
<code>comet3()</code>	绘制三维彗星状的轨迹	<code>compan()</code>	生成伴随矩阵
<code>compass()</code>	绕行曲线绘制	<code>computer()</code>	计算机类型测试
<code>cond()</code>	求矩阵的条件数	<code>condeit()</code>	估算范数
<code>conj()</code>	共轭复数函数	<code>contour()</code>	等高线图形绘制
<code>contour3()</code>	三维等高线绘制	<code>contourc()</code>	等高线绘图计算
<code>contrast()</code>	灰度对比度设置	<code>conv()</code>	求多项式乘法的卷积
<code>conv2()</code>	二维卷积	<code>cool</code>	天蓝粉色基色颜色表

(续)

函数/命令	功    能	函数/命令	功    能
copper	线性铜色调颜色表	corrcoef()	相关函数系数
cos()	余弦函数	cosh()	双曲余弦
cot()	余切函数	coth()	双曲余切
cov()	协方差矩阵	cplxdemo	复变量函数映射函数演示
cplxpair()	将数据按共轭复数对重新排序	cputime()	所用的 CPU 时间
csc()	余割函数	csch()	双曲余割函数
cusum()	各元素累加积	cylinder()	产生柱体
date()	日期	dbclear	消除跟踪调试断点
dbcont	跟踪调试恢复执行	dbdown	改变局部工作空间内容
dbquit	退出跟踪调试模式	dbstack	列出函数调用关系
dbstatus	列出所有的断点情况	dbstep	跟踪调试单步执行
dbstop	设置跟踪调试断点	dbtype	列出带有命令行标号的.M 文件
dbup	改变局部工作空间内容	deblank	消除字符串中的空格
dec2hex()	十进制到十六进制的转换	deconv()	因式分解与多项式除法
de12()	离散 Laplace 变换	delete	删除文件
delsqdemo	各种域上的有限差分演示	demo	运行 MATLAB 演示程序
det()	求矩阵的行列式	diag()	建立对角矩阵或获取对角向量
diary	将 MATLAB 运行的命令存盘	diff()	差分函数与近似微分
diffuse()	图像柔焦处理	dir	列出当前目录的内容
disp()	显示矩阵或文本	dmperm()	Dulmage-Mendelsohn 分解
drawnow	刷新绘图指令队列	earthmap	地球拓扑图形的显示
echo	显示文件中的 MATLAB 命令	eig()	求矩阵的特征值与特征向量
eigmovie	对称矩阵特征值求解过程演示	else	与 if 一起使用的转移语句
elseif	与 If 一起使用的转移语句	end	结束控制语句块的命令
eps	浮点相对差限	error()	显示错误信息并中断函数
errorbar()	误差条型图绘制	etree()	矩阵消元树结构
etreeplot()	绘制消元路径	etime()	所用时间的函数
eval()	执行 MATLAB 语句构成的字符串	exist()	检验变量或文件是否已经定义
expm()	矩阵指数函数	expml()	expm()函数的.M 文件实现
expm2()	Taylor 级数求矩阵指数	expm3()	特征值特征向量法求矩阵指数
exp()	指数函数	eye()	产生单位阵
fclose()	关闭文件	feather	羽状图形绘制
feof()	测试文件是否结束	ferror()	查询文件输入输出错误状态
feval()	执行字符串指定的文件	fft()	离散 Fourier 变换
fft2()	二维离散 Fourier 变换	fftdemo	快速 Fourier 变换演示
fftsift()	去掉谱分析中的直流分量	fgetl()	从文件读入一行数据(忽略换行)
fgets()	从文件读入一行数据(保留换行)	figure()	生成绘图窗口
fill()	绘制充填的二维多边形	fill3()	绘制充填的三维多边形
filter()	一维数字滤波	filter2()	二维数字滤波
find()	查找非零元素的下标	findstr()	由一个字符串中查找

(续)

函数/命令	功 能	函数/命令	功 能
<code>finite()</code>	若参数为有限元素则为真	<code>fitdemo</code>	非线性最优化拟合演示
<code>fix()</code>	对零方向取整数	<code>flag</code>	红白蓝 黑基色颜色表
<code>fliplr()</code>	按左右方向翻转元素	<code>fliplr()</code>	按上下方向翻转矩阵元素
<code>floor()</code>	对负无穷方向取整数	<code>flops()</code>	浮点运算计数器
<code>fmin()</code>	单变量最优化函数	<code>fmins()</code>	多变量最优化函数
<code>fopen()</code>	打开文件	<code>for</code>	循环语句
<code>format</code>	设置输出格式	<code>fourier</code>	Fourier 级数展开图形演示
<code>fplot()</code>	给定函数绘图	<code>fplotdemo</code>	函数图形绘制演示
<code>fprintf()</code>	有格式地向文件写入数据, 参见 C	<code>fread()</code>	从文件读入二进制数据
<code>frewind()</code>	将文件指针至文件开头	<code>fscanf()</code>	从文件有格式地读入数据, 参见 C
<code>fseek()</code>	设置文件位置指针	<code>ftell()</code>	获得文件位置指针
<code>full()</code>	由稀疏矩阵变换常規矩阵	<code>function</code>	MATLAB 函数表达式的引导符
<code>funm()</code>	矩阵的任意函数	<code>fwrite()</code>	将二进制数据写入文件
<code>fzero()</code>	单变量函数求根	<code>gallery()</code>	生成一些小的测试矩阵
<code>gca()</code>	获得当前坐标轴的句柄	<code>gcf()</code>	获得当前图形的窗口句柄
<code>get()</code>	获得对象属性	<code>getenv</code>	获得环境参数
<code>getframe()</code>	获得一幅“电影”图像	<code>ginput()</code>	由鼠标器作图像输入
<code>global</code>	定义全局变量	<code>gplot()</code>	绘制图论图形
<code>gray</code>	线性灰度颜色表	<code>graymon()</code>	将图形窗口设置成灰度默认值
<code>grid</code>	给图形加网格线	<code>griddata()</code>	插值用数据网格生成
<code>gradient()</code>	近似梯度计算	<code>gtext()</code>	在鼠标指定的位置加文字说明
<code>hadamard()</code>	生成 Hadamard 矩阵	<code>hankel()</code>	生成 Hankel 矩阵
<code>help</code>	启动联机帮助文件显示	<code>hess()</code>	求取 Hessenberg 标准型
<code>hex2num()</code>	十六进制到 IEEE 浮点数的转换	<code>hex2dec()</code>	十六进制到十进制的转换
<code>hidden</code>	网格图隐含线设置开关	<code>hilb()</code>	生成 Hilbert 矩阵
<code>hist()</code>	直方图绘制	<code>hold</code>	当前图形保护模式
<code>home</code>	将光标移动到左上角位置	<code>hostid</code>	MATLAB 服务器的主机代号
<code>hot</code>	黑红黄白基色颜色表	<code>hsv</code>	色度饱和值 (HSV) 颜色表
<code>hsv2rgb()</code>	HSV 对 RGB 颜色的转换	<code>if</code>	条件转移语句
<code>ifft()</code>	离散 Fourier 逆变换	<code>ifft2()</code>	二维离散 Fourier 逆变换
<code>imag()</code>	求取虚部函数	<code>image</code>	创建图像
<code>imagedemo</code>	MATLAB 4.0 版图形处理功能演示	<code>inf</code>	无穷大 (保留变量)
<code>info</code>	显示 MATLAB 与 MathWorks 信息	<code>input()</code>	带有提示的键盘输入函数
<code>int2str()</code>	整数转换为字符串	<code>interp1()</code>	一维插值 (一维查表)
<code>interp2()</code>	二维插值 (二维查表)	<code>interpft()</code>	利用 FFT 的一维插值
<code>intro</code>	MATLAB 引言信息	<code>inv()</code>	矩阵求逆
<code>invhilb()</code>	生成逆 Hilbert 矩阵	<code>isempty()</code>	若参数为空矩阵, 则结果为真
<code>isglobal()</code>	若参数为全局变量则为真	<code>ishold()</code>	若屏幕处于保护状态则为真
<code>isieee()</code>	若有 IEEE 算术标准则为真	<code>isinf()</code>	若参数为 Inf, 则结果为真
<code>isletter()</code>	若字符串为字母组成则为真	<code>isnan()</code>	若参数为 NaN, 则结果为真

(续)

函数/命令	功 能	函数/命令	功 能
issparse()	若矩阵为稀疏表示则为真	isstr()	若参数为字符串，则结果为真
jet	HSV 色调的变化型	keyboard	启动键盘管理程序
knot	围绕三维结的柱形显示	kron()	Kronecke 乘积函数
lasterr()	查询上的一条错误信息	length()	查询向量的维数
life	Conway 生命假设的 MATLAB 版	linspace()	构造线性分布的向量
load	从文件中读入变量	log()	自然对数函数
log10()	常用对数函数	loglog()	全对数坐标图绘制
logm()	矩阵的对数	logspace()	构造等对数分布的向量
lookfor	对 HELP 信息中的关键词查找	lorenz	Lorenz 混沌吸引子的曲线
lower()	将一个字符串内容转换为小写	lsqco()	最小二乘方差
lu()	矩阵的三角 (LU 分解)	magic()	生成魔术矩阵
matlabrc	启动主程序	max()	求向量中最大元素
mean()	求向量各元素均值	median()	求向量各元素中间值
membrane	产生 MathWorks 公司标志	menu()	产生用户输入的菜单
mesh()	三维网格图形	meshc()	带有等高线的网格图形
mgshgrid()	用 $x, y$ 阵列构造三维图形	meshz()	带有零平面的三维网格图形
min()	求向量中最小元素	more	控制命令窗口的输出页面
movie()	播放存储的“电影”幅面	moviein()	初始化“电影”各幅图像内存
mu21in()	声音文件对线性标度文件的转换	NaN	不定式
nargchk()	函数输入输出参数个数检验	nargin	函数中实际输入变量个数
nargout	函数中实际输出变量个数	newplot()	Nextplot 特性的.M 文件前缀
mextpow2()	找出下一个 2 的指数	nnz()	非零最小二乘
nnz()	非零元素个数	nonzeros()	非零元素
norm()	求矩阵的范数	normest()	估算范数
null()	右零空间	num2str()	将数值转换为字符串
nzmax()	允许的非零元素存储空间	ode23()	微分方程低阶数值解法
ode23p()	微分方程低阶数值解法并画图	ode45()	微分方程高阶数值解法
odedemo	常微分方程演示	ones()	产生元素全部为 1 的矩阵
orient()	设置打印纸方向	orth	正交空间
pack	整理工作空间内存	patch()	低级填充多边形绘制函数
path	设置或查询 MATLAB 的路径	paren	各种括号的查询信息
pascal()	生成 Pascal 矩阵	pause()	暂停函数
pcolor()	伪颜色绘图	peaks	两变量的峰值函数演示
penny	便士硬币的各个角度视图	pi	圆周率( $\pi$ )
pink	粉色色调颜色表	pinv()	伪逆矩阵
plot()	线性坐标图形绘制	plot3()	绘制三维线或点型图形
polar()	极坐标图形绘制	poly()	求矩阵的特征多项式
polyder()	多项式求导	polyfit()	数据的多项式拟合
polyval()	多项式求值	polyvalm()	多项式矩阵求值
print()	打印图形或将图形存盘	printopt()	建立打印机默认值

(续)

函数/命令	功 能	函数/命令	功 能
prism	光谱颜色表	prod()	对向量中各元素求积
punct	各种标点符号的查询信息	qr()	矩阵的正交三角化 (QR) 分解
qrdelete()	QR 分解中删除一列	qrinsert()	QR 分解中插入一列
quad()	低阶数值积分算法	quad8()	高阶数值积分算法
quaddemo	自适应变步长数值演示	quake	Loma Prieta 地震模型
quit	退出 MATLAB 环境	quiver()	箭头图形
qz()	广义特征值问题求解 (QZ 算法)	rand()	产生随机矩阵
randn()	产生正态分布随机阵	randperm()	随机置换向量
rank()	求矩阵的秩	rbbox()	擦除框
rcond()	LINPACK 倒数条件估计	real()	求取实部函数
realmex	最大浮点数值	realmmin	最小浮点数值
relop	各种关系符号的查询信息	rem()	除法的余数
reset()	恢复对象特性	return	返回到主调函数的命令
rgb2HSV()	RGB 对 HSV 颜色的转换	rgbplot()	绘制颜色图
roots()	求多项式的根	rose()	极坐标 (角度) 直方图绘制
rosser()	典型的对称矩阵特征值问题测试	rot90()	将矩阵元素旋转 90 度
round()	截取到最近的整数	rref()	矩阵的行阶梯型实现
rrefmovie	消元法解方程过程演示	rsf2csf()	实块对角阵转移复块对角阵
save	将工作空间中变量存盘	saxis()	声音坐标轴处理
surfnorm()	表面图形规范化	schur()	Schur 分解
script	MATLAB 语句及文件信息	sec()	正割函数
sech()	双曲正割	semilogx()	x 轴半对数坐标图形绘制
semilogy()	y 轴半对数坐标图形绘制	sepdemo	有限元网格图演示
set()	设置对象属性	setstr()	将数值转换为字符串
shading	阴影模式	sigdemo1	离散 Fourier 变换演示
sigdemo2	连续 Fourier 变换演示	sign()	符号函数
sin()	正弦函数	sinh()	以曲正弦
size()	查询矩阵的维数	slash	求解线性方程 (左除右除) 信息
slice()	容量可视图形	sort()	对向量中各元素排序
sound()	将数据向量转换为声音	sounddemo	MATLAB4.0 的声音功能演示
spalloc()	给非零元素定位存储空间	sparse()	从常规矩阵转换稀疏矩阵
sparsity	稀疏矩阵排序效应演示	spaugment()	建立最小二乘增广系统
spconvert()	由稀疏矩阵外部格式进行转换	spdiags()	稀疏对角矩阵
specular()	反射	speye()	稀疏单位矩阵
spfunF()	对稀疏矩阵处理的非线性函数	sphere()	产生球面
spinmap()	使颜色旋转	spline2rd	二维样条函数演示
spnames()	将原稀疏矩阵非零元素用 1 取代	spparms()	设置稀疏矩阵参数

(续)

函数/命令	功    能	函数/命令	功    能
sprank()	结构秩数	sprandn()	稀疏随机矩阵
sprandsym()	稀疏对称随机矩阵	sprintf()	按照 C 语言格式书写字符串
spy()	绘制稀疏矩阵结构	sqdemo	超二次锥面的显示
sqrt()	平方根函数	sqrtm()	矩阵的平方根
sscanf()	按照 C 语言格式读字符串	stairs()	阶梯图形绘制
startup	MATLAB 自启动文件	std()	求向量中各元素标准方差
stem()	离散序列柄状图形绘制	str2mat()	字符串转换成矩阵
str2num()	字符串转换为实型数据	strcmp()	字符串比较
strings	关于 MATLAB 字符串的帮助信息	subplot	将图形窗口分成若干个区域
subscribe	成为 MATLAB 的签约用户	subspace()	子空间
sum()	对向量中各元素求和	sunspots	太阳黑子活动模拟
surf()	三维表面图形	surface()	创建曲面
surfc()	带有等高线的三维表面图形	surf3()	带有光照阴影的三维表面图形
svd()	奇异值分解 (SVD)	symbfact()	符号因式分解
symmd	对称最小阶次	symrcm()	逆序 Cuthill-McKee 排序
tan()	正切函数	tanh()	双曲正切
terminal	设置图形终端类型	text()	在图形上加文字说明
tic()	启动秒表计时器	title()	给图形加标题
toc()	读取秒表计时器	toeplitz()	生成 Toeplitz 矩阵
trace()	求矩阵的迹	trapz()	梯形法求数值积分
treelayout()	树状结构	treeplot()	画出分割路径的图形
tril()	提取矩阵的下三角部分	triu()	提取矩阵的上三角部分
type	列出.M 文件	uicontrol()	建立用户界面控制的函数
ungetfile()	标准读盘文件名处理对话框	uisetcolor()	标准颜色设置对话框
uisetfont()	标准字体设置树对话框	unix	执行操作系统命令并回结果
unwrap()	除去每 360° 的跳跃	upper()	将一个字符串内容转换为大写
vander()	生成 Vandermonde 矩阵	ver	显示程序版本号
version	显示 MATLAB 版本号	vibes	L 型振荡动画
view()	三维图形视角指定	viewmtx()	显示坐标变换矩阵
waterfall()	瀑布图	what	列出当前目录下的有关文件
whatsnew	手册中未给出的新特性	which	找出函数与文件所在的目录名
while	循环语句	whitebg	将图形窗口设置成白色背景
who	简要列出工作空间变量名	whos	详细列出工作空间变量名
why	给出简要的回答	wilkinson	生成 Wilkinson 特征值测试矩阵
xlabel()	给图形加 X 轴标注	xor()	逻辑异或
ylabel()	给图形加 Y 轴标注	zerodemo	求根演示
zeros()	产生零矩阵	zlabel()	给图形加 Z 轴标注

## 附录 B 图形对象属性

本附录给出了 MATLAB5.3 版本中各种图形对象的属性及其取值。  
表中用“{}”括起来的属性值是缺省值。

表 B-1 根屏幕属性

属性名	功    能	属性取值
BusyAction	对根屏幕对象不起作用	
ButtonDownFcn	对根屏幕对象不起作用	
CallbackObject	回调程序正在执行的对象的句柄	句柄
Children	子对象句柄，包括所有未被隐藏的图形对象	句柄
Clipping	对根屏幕对象不起作用	
CreateFcn	对根屏幕对象不起作用	
CurrentFigure	当前图形窗口的句柄	
DeleteFcn	对根屏幕对象不起作用	
Diary	日志文件模式。采用这种模式时，MATLAB 将保留一个文件（文件名由 DiaryFile 指定）用以保存所有的键盘输入和尽可能多的输出结果。	{off}：不保留日志文件 on：保留日志文件
DiaryFile	日志文件名	字符串
Echo	脚本文件的显示模式	{off}：脚本文件执行时不显示其内容 on：脚本文件执行时显示文件的每一行
ErrorMessage	最近的一个错误信息	字符串（只读）
Format	显示数据的格式	short：5位定点格式 shortE：5位浮点格式 shortG：5位定点或浮点格式 long：15位定点格式 longE：15位浮点格式 longG：15位定点或浮点格式 bank：元和分的定点格式 hex：十六进制格式 +：显示“+”和“-”号 rat：用小整数的比率逼近
FormatSpacing	输出间隔	{loose}：显示附加行的输入，使得可读性更强 compact：不显示附加输入
HandleVisibility	对根屏幕对象不起作用	
HitTest	对根屏幕对象不起作用	
Interruptible	对根屏幕对象不起作用	

(续)

属性名	功    能	属性取值
Language	系统环境设置	字符串
Parent	父对象句柄, 对根屏幕对象是空矩阵	
PointerLocation	鼠标指针位置	指针位置相对于屏幕左下角的坐标, 两元素向量: [x, y]
PointerWindow	含有鼠标指针的图形窗口句柄	句柄(只读)
Profile	测定 MATLAB 执行 M 文件(在 ProfileFile 中指定)中的每一行所用时间。请参见 profile 命令的用法。	{on} off
ProfileCount	执行 profile 的输出	向量
ProfileFile	相应的 M 文件名	字符串
ProfileInterval	执行 profile 的时间间隔	标量
ScreenDepth	屏幕颜色深度, 以比特为单位	整数, 2 的幂次。比如 1 表示黑白单色, 4 表示 16 色
ScreenSize	屏幕尺寸	四元素位置向量: [左边位置、右边位置、屏幕宽、屏幕高]。其中左边位置和右边位置一般为 0。
Selected	对根屏幕对象不起作用	
SelectionHighlight	对根屏幕对象不起作用	
ShowHiddenHandles	显示或不显示标记为隐藏的句柄	{off}: 不显示 on: 显示
Tag	用户指定的对象标记	字符串
TerminalDimensions	终端大小(X-Windows)	像素个数
TerminalHideGraphCommand	隐藏终端窗口命令(X-Windows)	字符串
TerminalOneWindow	是否终端上只有一个窗口(X-Windows)	{on}: 是 off: 否
TerminalProtocol	终端类型(X-Windows)	none x tek401x tek410x
TerminalShowGraphCommand	显示终端窗口命令(X-Windows)	字符串
Type	对象类型	root
UIContextMenu	对根屏幕对象不起作用	
Units	尺寸和位置数据的单位	{pixels}: 像素 normalized: 归一化坐标, 屏幕的左下角为[0 0], 右上角为[1 1] inches: 英寸 centimeters: 厘米 points: 点, 等于 1/72 英寸 characters: 缺省系统字体的字符尺寸, 宽等于一个字母的宽度, 高等于两行文本基线之间的距离。
UserData	用户指定的数据	矩阵或字符串
Visible	对根屏幕对象不起作用	

表 B-2 图形窗口属性

属性名	功 能	属性取值
BackingStore	存储图形窗口的拷贝到内存缓冲区，以实现快速重画	{on}: 当图形被覆盖时，存储被覆盖区域的拷贝。被覆盖区域重新显露时，调用拷贝。这种方式刷新窗口速度快，但需要较大的内存容量。 off: 被覆盖区域重新显露时，重画该区域的图形。这种方式刷新窗口速度慢，但节省内存。
BusyAction	回调程序中断方式，指当一个回调程序执行时，如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on，那么后面的中断请求自动以队列的方式进行；如果该属性设为 off，那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在图形窗口上时（不在它的子对象之上）按下鼠标按钮，这时要调用的程序	字符串，可以是 MATLAB 表达式或 M 文件名
Children	图形对象中所有坐标轴对象、用户界面菜单、用户界面控件的句柄	向量（只读）
Clipping	对图形对象不起作用	
CloseRequestFcn	图形窗口以任何方式被关闭时执行的程序	字符串，可以是 MATLAB 表达式或 M 文件名
Color	图形背景色	RGB 向量或 MATLAB 中固定的颜色名。缺省颜色是灰色 ([0.8 0.8 0.8])
Colormap	图形颜色映象（参见第六章）	mx3 矩阵或 RGB 向量
CreateFcn	建立对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
CurrentAxes	图形的当前坐标轴的句柄	坐标轴句柄
CurrentCharacter	最近在键盘上按下的键	字符（只读）
CurrentObject	在当前位置（见 CurrentPoint 属性）的对象句柄	对象句柄
CurrentPoint	最近按下鼠标按钮时，鼠标指针的位置	位置坐标[x, y]
DeleteFcn	删除对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
Dithermap	抖动处理的颜色映象	RGB 值的 mx3 矩阵
DithermapMode	抖动处理模式	{manual}: 根据属性 Dithermap 中定义的颜色映象显示色彩 auto: 根据当前显示的颜色自动产生一个抖动处理的颜色映象
DoubleBuffer	双缓冲器，实现对简单动画的快速着色。这个属性要求动画对象的属性 EraseMode 设为 normal，图形对象的 Renderer 属性设为 painters。	{off}: 不使用双缓冲器 on: 使用双缓冲器
FixedColors	定义在图形窗口中不从“颜色映象”中获取的颜色，包括：坐标轴线和标注、线条、文本、用户界面控件、用户界面菜单的颜色	RGB 值的 mx3 矩阵
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
MinColormap	使用颜色表输入项的最小个数	标量，缺省值 64
InvertHardcopy	改变图形对象的颜色以输出到打印设备。	{on}: 将图形背景色改为白色，坐标轴、线条、文本改为黑色以便打印。 off: 打印输出的颜色和屏幕颜色一致。

(续)

属性名	功 能	属性取值
HitTest	指定图形是否能够通过鼠标单击被选中，成为当前对象。	{on}: 图形可以被鼠标选中为当前对象，即成为 geo 命令和图形属性 CurrentObject 的返回值。 off: 鼠标在图形上单击后，geo 命令和图形属性 CurrentObject 的返回值为空矩阵
IntegerHandle	图形句柄的格式	{on}: 整数 off: 实数
Interruptible	回调程序中断模式。控制图形的回调程序是否可以被后面回调程序的请求中断。此属性只影响以下几个属性的回调函数：ButtonDownFcn, KeyPressFcn, WindowButtonDownFcn, WindowButtonMotionFcn, WindowButtonUpFcn。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
KeyPressFcn	按键的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
MenuBar	显示或不显示图形窗口的菜单	{figure}: 显示 none: 不显示
Name	图形窗口标题名	字符串（缺省情况是空的），如果 NumberTitle 属性为 on，则标题格式为：Figure No.n: 字符串
NextPlot	决定如何绘制下一幅图	{add}: 使用当前图形窗口的属性，添加新的图形。 replace: 在画图前，将除 Position 属性外的所有图形窗口属性重置为缺省值，并删除所有子对象。 replacechildren: 删除所有子对象，但不重置图形窗口属性。
NumberTitle	是否在图形窗口标题中使用序号	{on}: 使用； off: 不使用
Pointer	图形窗口内的鼠标指针形状	crosshair: 十字形指针 {arrow}: 箭头 watch: 钟表形 topl: 指向左上方的箭头 topr: 指向右上方的箭头 botl: 指向左下方的箭头 botr: 指向右下方的箭头 circle: 圆形 cross: 双线十字形 fleur: 指南针形 left: 指向左边的箭头 right: 指向右边的箭头 top: 指向上方的箭头 bottom: 指向下方的箭头 fullcrosshair: 中心为空的十字形指针 ibeam: 工字形 custom: 用户定制
PaperPosition	打印图形在页面上的位置	四元素的向量：[到左边框的距离，到右边框的距离，图形宽度，图形高度]
PaperPositionMode	图形在页面上的位置模式	auto: 按照图形在屏幕上的相同尺寸打印，不管属性 PaperPosition 的设置 {manual}: 按照 PaperPosition 的设置打印

(续)

属性名	功 能	属性取值
PaperSize	打印纸的尺寸	两元素向量: [宽, 高]。(只读) {usletter}: 标准美国信纸, 8.5×11 英寸 uslegal: 标准美国法定纸张, 11×14 英寸 tabloid: 标准美国报纸, 11×17 英寸 A0: 841×1189mm A1: 594×841mm A2: 420×594mm A3: 297×420mm A4: 210×297mm A5: 148×210mm B0: 1029×1456mm B1: 728×1028mm B2: 514×728mm B3: 364×514mm B4: 257×364mm B5: 182×257mm arch-A: 9×12 英寸 arch-B: 12×18 英寸 arch-C: 18×24 英寸 arch-D: 24×36 英寸 arch-E: 36×48 英寸 A: 8.5×11 英寸 B: 11×17 英寸 C: 17×22 英寸 D: 22×34 英寸 E: 34×43 英寸
PaperType	打印纸的类型	normalized: 归一化坐标 {inches}: 英寸 centimeters: 厘米 points: 点, 每点 1/72 英寸
PaperUnits	纸张尺寸的单位	painters: MATLAB 的原始着色方式, 当图形窗口只包括小的或简单的图形对象时比较快 zbuffer: 只给在屏幕上可见的像素着色, 因而速度比较快而且准确, 但要消耗大量的内存 OpenGL: 这是一种可以在很多种计算机系统上使用的着色方式, 比前两种方式都快, 而且有些情况下允许 MATLAB 访问绘图硬件。
Renderer	显示和打印的着色工具	{auto}: MATLAB 根据图形的尺寸和复杂程度自动选择着色工具。 manual: 用户选择着色工具。
Resize	允许或不允许改变图形窗口大小	{on}: 允许改变 off: 不允许改变
Parent	图形窗口父对象的句柄	句柄, 一般为 0

(续)

属性名	功 能	属性取值
PointerShapeCData	用户定制的指针图形数据, 16×16 像素	16×16 的矩阵, 元素的取值: 1: 黑色像素 2: 白色像素 NaN: 去掉像素点(透明)
PointerShapeHotSpot	指针的活动区域, 指定 PointerShapeCData 中的矩阵的那个点用来指示指针的确切位置	两元素向量: [x, y]
Position	图形窗口在屏幕中的位置	四元素的向量: [到屏幕左边的距离, 到屏幕右边的距离, 图形窗口宽度, 图形窗口高度]
ResizeFcn	图形窗口大小改变时的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Selected	是否被选中	on: 被选中; off: 未被选中
SelectionHighlight	对图形窗口不起作用	
SelectionType	提供鼠标选择类型的信息, 这个属性的值和相同类型有关. 只读属性。	{normal}: 按下鼠标左键 extend: Shift+按下鼠标左键或同时按下鼠标左右键 alt: Ctrl+按下鼠标左键或按下鼠标右键 open: 双击任何鼠标键
ShareColors	共享颜色表的槽	{on}: 在可能的情况下, 重复使用颜色表中的槽向量 off: 不和其它窗口共享颜色表的槽
Tag	用户指定的对象标记	字符串
Type	对象类型	figure
UIContextMenu	把图形和一个上下文菜单联系起来	图形中建立的上下文菜单句柄
Units	尺寸和位置数据的单位	{pixels}: 像素 normalized: 归一化坐标, 屏幕的左下角为[0 0], 右上角为[1 1] inches: 英寸 centimeters: 厘米 points: 点, 等于 1/72 英寸 characters: 缺省系统字体的字符尺寸, 宽等于一个字母的宽度, 高等于两行文本基线之间的距离。
UserData	用户指定的数据	矩阵或字符串
Visible	图形窗口是否可见	{on}: 可见; off: 不可见
WindowButtonDownFcn	当鼠标指针在图形窗口上时按下鼠标按钮, 这时要执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
WindowButtonMotionFcn	在图形窗口上移动鼠标指针时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
WindowButtonUpFcn	当鼠标指针在图形窗口上时释放鼠标按钮, 这时要执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
WindowStyle	窗口类型	{normal}: 常规 modal: 有模式窗口, 某个窗口设为这种类型后, MATLAB 将使鼠标和键盘只在该窗口有效, 在其它窗口(包括 MATLAB 命令窗口)不能操作。使用 Ctrl+C 键可以把这种类型还原为 normal.

表 B-3 坐标轴属性

属性名	功 能	属性取值
AmbientLightColor	景物的背景光，无方向、均匀地照射在所有目标上。如果坐标轴对象中未定义光源对象，那么这个属性将不起作用。	颜色矩阵
Box	坐标轴的边框，指定是否把坐标轴的范围封闭在一个边框内	{off}：不加边框 on：加边框
BusyAction	回调程序中断方式，指当一个回调程序执行时，如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on，那么后面的中断请求自动以队列的方式进行；如果该属性设为 off，那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}：排成队列等待执行 cancel：拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在坐标轴上时（不在其它对象之上）按下鼠标按钮，这时要调用的程序	字符串，可以是 MATLAB 表达式或 M 文件名
CLim	颜色轴界限，确定 MATLAB 如何把表面对象和补片对象的 Cdata 属性值映射到图形的颜色映象矩阵。	[cmin, cmax] cmin：映射到颜色映象的第一个颜色的数据值。 cmax：映射到颜色映象的最后一个颜色的数据值
CLimMode	颜色限制模式	{auto}：把 CLim 属性设置为颜色映象矩阵到 Cdata 数据的完全映射。 manual：按照 CLim 属性进行映射。只要设置了 CLim 属性的值，CLimMode 就自动成为 manual 模式。
CameraPosition	照相机的位置	[x, y, z]
CameraPositionMode	照相机的位置模式	{auto}：MATLAB 自动计算照相机的位置。 manual：设置了 CameraPosition 的值，就自动成为 manual 模式。
CameraTarget	被照目标的位置	[x, y, z]
CameraTargetMode	被照目标的位置模式	{auto}：MATLAB 自动把被照目标定位在坐标轴的质心。 manual：设置了 CameraTarget 的值，就自动成为 manual 模式。
CameraUpVector	照相机的向上向量	[x, y, z]
CameraUpVectorMode	照相机的向上向量模式	{auto}：对三维坐标 MATLAB 自动把 CameraUpVector 设为[0 0 1]，对二维坐标，自动设为[0 1 0]。 manual：设置了 CameraUpVector 的值，就自动成为 manual 模式。
CameraViewAngle	照相机的视角	大于 0 度小于或等于 180 度的度数值
CameraViewAngleMode	照相机的视角模式	{auto}：MATLAB 自动把 CameraViewAngle 设为捕获整个景物的最小视角。 manual：设置了 CameraViewAngle 的值，就自动成为 manual 模式。

(续)

属性名	功 能	属性取值
Children	坐标轴对象的子对象，包括 images、lights、lines、patches、surfaces 和 text	句柄
Clipping	对坐标轴对象无效	
Color	坐标轴背景颜色	{none}: 使用图形的背景色。 RGB 向量或 MATLAB 中固定的颜色。
ColorOrder	用来绘制多条曲线的颜色	RGB 颜色的 $m \times 3$ 矩阵
CreateFcn	建立对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
CurrentPoint	最近按下鼠标按钮时，鼠标指针的位置	位置坐标[x, y]
DataAspectRatio	数据单元的相对缩放比例，比如设为[2 1 1]意味着 x 方向的 2 个数据单元等于 y 方向和 z 方向的一个数据单元的长度	[dx, dy, dz]
DataAspectRatioMode	数据单元缩放比例的模式	{auto}: 自动设置相对缩放比例 manual: 设置了 DataAspectRatio 的值，就自动成为 manual 模式。在此模式下，不能进行拉伸操作。
DeleteFcn	删除对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
DrawMode	绘制对象的方式	{normal}: 在当前的视角下，从后向前绘制。 fast: 不考虑对象在三个方向的相互关系，按绘图命令指定的顺序绘制。这种方式速度快，但可能产生不理想的效果。
FontAngle	字体角度	{normal}: 正常 italic: 斜体 oblique: 在有些系统中为斜体
FontName	字体名	字符串，缺省值是 Helvetica
FontSize	字体尺寸	整数
FontUnits	尺寸单位	{points}: 点，1/72 英寸 normalized: 归一化 inches: 英寸 centimeters: 厘米 pixels: 像素
FontWeight	字体的粗细	{normal}: 正常 bold: 粗体 light: 淡字体 demi: 适中或粗体
GridLineStyle	网格线型	-: 实线 --: 虚线 {:}: 点线 -.: 点划线 none: 无

(续)

属性名	功 能	属性取值
HitTest	指定坐标轴是否能够通过鼠标单击被选中，成为当前对象。	{on}: 坐标轴可以被鼠标选中为当前对象，即成为 gco 命令和属性 CurrentObject 的返回值。 off: 鼠标在坐标轴上单击后，选择坐标轴下面的对象。
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
Layer	坐标轴线的层次	{bottom}: 在图形对象的下层画坐标轴线 top: 在图形对象的上层画坐标轴线
LineStyleOrder	设置绘图中使用的线条和标记类型的顺序	类型符号字符串
LineWidth	坐标轴线的宽度	以点为单位的数
NextPlot	决定如何绘制下一幅图	{add}: 使用当前的坐标轴，添加新的图形。 replace: 在画图前，将除 Position 属性外的所有坐标轴属性重置为缺省值，并删除所有子对象。 replacechildren: 删除所有子对象，但不重置图形窗口属性。
Parent	坐标轴的父对象句柄	图形窗口句柄
PlotBoxAspectRatio	坐标轴边框的相对缩放比例	[px, py, pz]
PlotBoxAspectRatioMode	坐标轴边框的相对缩放比例模式	{auto}: 自动设置相对缩放比例 manual: 设置了 PlotBoxAspectRatio 的值，就自动成为 manual 模式。
Position	坐标轴相当于图形窗口的位置	四元素向量：[坐标轴左下角相对于图形窗口的横坐标，坐标轴左下角相对于图形窗口的纵坐标，坐标轴的宽，坐标轴的高]
Projection	投影类型	{orthographic}: 正交 perspective: 远景
Selected	对象是否被选中	on: 被选中时在角落显示选中的对象句柄，如果 SelectionHighlight 也设为 on，那么将在中央显示句柄 off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
Tag	用户指定的对象标记	字符串
TickDir	刻度线的方向	in: 刻度线向内（二维坐标轴的缺省值） out: 刻度线向外（三维坐标轴的缺省值）
TickDirMode	刻度线的方向控制	{auto}: 自动设置刻度线的方向 manual: 设置了 TickDir 的值，就自动成为 manual 模式。

(续)

属性名	功 能	属性取值
TickLength	刻度线的长度	两个元素的向量: [二维图形刻度线的长度, 三维图形刻度线的长度]
Title	坐标轴的标题	文本对象的句柄
Type	图形对象的类型(只读)	axes
UserData	用户指定的数据	矩阵或字符串
Units	位置数据的单位	pixels: 像素 {normalized}: 归一化坐标, 屏幕的左下角为[0 0], 右上角为[1 1] inches: 英寸 centimeters: 厘米 points: 点, 等于 1/72 英寸 characters: 缺省系统字体的字符尺寸, 宽等于一个字母的宽度, 高等于两行文本基线之间的距离。
UIContextMenu	把坐标轴和一个上下文菜单联系起来	在坐标轴的父对象——图形对象中建立的上下文菜单句柄
Visible	坐标轴是否可见, 这个属性不影响坐标轴的子对象	{on}: 可见 off: 不可见
XAxisLocation	X 轴的位置	{bottom}: 下面 top: 上面
XColor	X 轴的颜色	RGB 向量或 MATLAB 中固定的颜色名
XDir	X 轴的正方向	{normal}: 从左向右 reverse: 从右向左
XGrid	X 轴的网格线	{off}: 无网格线 on: 添加垂直 x 轴的网格线
XLabel	X 轴的标注	文本对象句柄
XLim	X 轴的取值范围	[最小值, 最大值]
XLimMode	X 轴的取值模式	{auto}: 自动设置取值范围 manual: 设置了 XLim 的值, 就自动成为 manual 模式。
XScale	X 轴的刻度类型	{linear}: 线性 log: 对数
XTick	X 轴的刻度向量, 按此向量把刻度线画在坐标轴上	向量
XTickLabel	X 轴的刻度标记	字符串数组
XTickLabelMode	X 轴的刻度标记模式	{auto}: 自动设置刻度标记 manual: 设置了 XTickLabel 的值, 就自动成为 manual 模式。
XTickMode	X 轴的刻度模式	{auto}: 自动设置刻度 manual: 设置了 XTick 的值, 就自动成为 manual 模式。
YAxisLocation	Y 轴的位置	{left}: 左侧 right: 右侧
YColor	Y 轴的颜色	RGB 向量或 MATLAB 中固定的颜色名

(续)

属性名	功 能	属性取值
YDir	Y 轴的正方向	{normal}: 对二维坐标轴从下向上, 三维坐标轴从前向后 reverse: 和 normal 相反
YGrid	Y 轴的网格线	{off}: 无网格线 on: 添加垂直 y 轴的网格线
YLabel	Y 轴的标注	文本对象句柄
YLim	Y 轴的取值范围	[最小值, 最大值]
YLimMode	Y 轴的取值模式	{auto}: 自动设置取值范围 manual: 设置了 YLim 的值, 就自动成为 manual 模式。
YScale	Y 轴的刻度类型	{linear}: 线性 log: 对数
YTick	Y 轴的刻度向量, 按此向量把刻度线画在坐标轴上	向量
YTickLabel	Y 轴的刻度标记	字符串数组
YTickLabelMode	Y 轴的刻度标记模式	{auto}: 自动设置刻度标记 manual: 设置了 YTickLabel 的值, 就自动成为 manual 模式。
YTickMode	Y 轴的刻度模式	{auto}: 自动设置刻度 manual: 设置了 YTick 的值, 就自动成为 manual 模式。
ZColor	Z 轴的颜色	RGB 向量或 MATLAB 中固定的颜色名
ZDir	Z 轴的正方向	{normal}: 对二维坐标轴相对屏幕从里向外, 三维坐标轴从下向上 reverse: 和 normal 相反
ZGrid	Z 轴的网格线	{off}: 无网格线 on: 添加垂直 z 轴的网格线
ZLabel	Z 轴的标注	文本对象句柄
ZLim	Z 轴的取值范围	[最小值, 最大值]
ZLimMode	Z 轴的取值模式	{auto}: 自动设置取值范围 manual: 设置了 ZLim 的值, 就自动成为 manual 模式。
ZScale	Z 轴的刻度类型	{linear}: 线性 log: 对数
ZTick	Z 轴的刻度向量, 按此向量把刻度线画在坐标轴上	向量
ZTickLabel	Z 轴的刻度标记	字符串数组
ZTickLabelMode	Z 轴的刻度标记模式	{auto}: 自动设置刻度标记 manual: 设置了 ZTickLabel 的值, 就自动成为 manual 模式。
ZTickMode	Z 轴的刻度模式	{auto}: 自动设置刻度 manual: 设置了 ZTick 的值, 就自动成为 manual 模式。

表 B-4 图形界面控件 (Uicontrol) 属性

属性名	功    能	属性取值
BackgroundColor	对象的背景色，用于填充图形界面控件的矩形	RGB 向量或 MATLAB 中固定的颜色名
BusyAction	回调程序中断方式，指当一个回调程序执行时，如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on，那么后面的中断请求自动以队列的方式进行；如果该属性设为 off，那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在控件周围 5 个像素范围内时（不在其它对象之上）按下鼠标按钮，这时要调用的程序	字符串，可以是 MATLAB 表达式或 M 文件名
Callback	激活控件对象时执行的程序	字符串，可以是 MATLAB 表达式或 M 文件名
CData	在控件上显示的真彩色图象的 RGB 值矩阵	三维矩阵
Children	控件无子对象	空矩阵
Clipping	对控件对象无效	
CreateFcn	建立对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
Enable	控件是否处于工作状态	{on}: 工作状态 inactive: 控件不工作，但显示的状态和工作状态相同 off: 控件不工作，而且显示为不活动状态
Extent	用于标注控件的特征字符串的尺寸	四元素向量：[0, 0, 宽度, 高度]
FontAngle	字体角度	{normal}: 正常 italic: 斜体 oblique: 在有些系统中为斜体
FontName	字体名	字符串，缺省值是 Helvetica
FontSize	字体尺寸	整数
FontUnits	尺寸单位	{points}: 点, 1/72 英寸 normalized: 归一化 inches: 英寸 centimeters: 厘米 pixels: 像素
FontWeight	字体的粗细	{normal}: 正常 bold: 粗体 light: 淡字体 demi: 适中或粗体
ForegroundColor	前景色，指控件的标注文本的颜色	RGB 向量或 MATLAB 中固定的颜色名
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
HitTest	对控件对象无效	
HorizontalAlignment	控件标注字符串的水平排列方式	{center}: 相对于控件居中 left: 相对于控件靠左 right: 相对于控件靠右
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。

(续)

属性名	功    能	属性取值
ListboxTop	指定显示在列表框最上方的字符串的序号, 此属性只针对控件中的列表框类型	标量
Max	当前控件允许的最大值	标量
Min	当前控件允许的最小值	标量
Parent	父对象句柄	句柄
Position	控件相当于图形窗口的位置	四元素向量: [控件左下角相对于图形窗口的横坐标, 控件左下角相对于图形窗口的纵坐标, 控件的宽度, 控件的高度]
Selected	对象是否被选中	on: 如果 SelectionHighlight 也设为 on, 被选中时显示句柄 off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
SliderStep	滑动条步长范围	[最小步长, 最大步长]
String	标注字符串、列表框的项目、弹出式菜单的选项	字符串
Style	控件类型	{pushbutton}: 按钮 togglebutton: 开关按钮 radiobutton: 单选按钮 checkbox: 复选框 edit: 编辑框 text: 静态文本框 slider: 滑动条 frame: 图文框 listbox: 列表框 popupmenu: 弹出式菜单
Tag	用户指定的对象标记	字符串
TooltipString	控件的工具提示的内容	字符串
Type	对象类型	uicontrol
UIContextMenu	把控件和一个上下文菜单联系起来, 当右键在控件上单击鼠标右键时显示该上下文菜单	句柄
Units	区域和位置数据的单位	{pixels}: 像素 normalized: 归一化坐标, 屏幕的左下角为[0 0], 右上角为[1 1] inches: 英寸 centimeters: 厘米 points: 点, 等于 1/72 英寸 characters: 缺省系统字体的字符尺寸, 宽等于一个字母的宽度, 高等于两行文本基线之间的距离。
UserData	用户指定的数据	矩阵或字符串
Value	当前控件的值	标量或向量
Visible	控件是否可见	{on}: 可见 off: 不可见

表 B-5 图形界面菜单属性

属性名	功 能	属性取值
Accelerator	快捷键的字符，按下 Ctrl 键和指定的字符快速执行对应命令	字符
ButtonDownFcn	对菜单不起作用	
Clipping	对菜单不起作用	
Callback	菜单项的调用程序	字符串，可以是 MATLAB 表达式或 M 文件名
BusyAction	回调程序中断方式，指当一个回调程序执行时，如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on，那么后面的中断请求自动以队列的方式进行；如果该属性设为 off，那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
Checked	菜单项被选中时作标记	{off}: 不作标记 on: 作标记
Children	菜单对象无子对象	
CreateFcn	建立对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
Enable	菜单项是否处于可选状态	{on}: 可选状态 off: 不可选状态
ForegroundColor	前景色，指菜单的文本颜色	RGB 向量或 MATLAB 中固定的颜色名
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
Label	含有菜单项标记的文本串。标记中前面有‘&’的，定义了快捷键，它由 Alt+字符激活	字符串
Parent	父对象句柄	句柄
Position	uimenu 对象的相对位置。顶层菜单从左到右编号，子菜单从上至下编号	标量
Selected	对菜单不起作用	
SelectionHighlight	对菜单不起作用	
Separator	菜单项之间的分隔线	{off}: 不画分隔线 on: 分隔线在菜单项之上
Tag	用户指定的对象标记	字符串
Type	对象类型	uimenu
UIContextMenu	对菜单不起作用	
UserData	用户指定的数据	矩阵或字符串
Visible	菜单是否可见	{on}: 可见 off: 不可见

表 B-6 图形界面上下文菜单属性

属性名	功 能	属性取值
Callback	在定义了上下文菜单的对象上单击鼠标右键时调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Children	上下文菜单无子对象	
HitTest	对上下文菜单无效	
Clipping	对上下文菜单无效	
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
Parent	父对象	句柄
Selected	对上下文菜单无效	
SelectionHighlight	对上下文菜单无效	
Tag	用户指定的对象标记	字符串
Type	对象类型	uicontextmenu
UIContextMenu	对上下文菜单无效	
UserData	用户指定的数据	矩阵或字符串
Visible	上下文菜单是否可见	{off}: 不可见 on: 可见

表 B-7 线条属性

属性名	功 能	属性取值
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在线条上时按下鼠标按钮, 这时要调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Children	线条对象无子对象	

(续)

属性名	功 能	属性取值
Clipping	是否修剪超出坐标轴边框的线条	{on}: 剪掉超出坐标轴边框的线条 off: 不修剪
Color	线条颜色	RGB 向量或 MATLAB 中固定的颜色名。
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
EraseMode	擦除方式	{normal}: 重画影响显示的作用区域, 以保证所有的对象正确地画出。这是最精确的, 也是最慢的一种模式  background: 通过以图形背景色重画线来消除线条。这会破坏被消除的线条后面的对象  xor: 用线条下屏幕的颜色执行异或运算, 画出和消除线条。当画在其它对象上时, 可能造成不正确的颜色  none: 当移动或删除线条时该线不会被消除
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
HitTest	指定线条是否能够通过鼠标单击被选中, 成为当前对象。	{on}: 线条可以被鼠标选中为当前对象, 即成为 <code>gco</code> 命令和图形属性 <code>CurrentObject</code> 的返回值。 off: 鼠标在线条上单击后, 选择它下面的对象
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 <code>ButtonDownFcn</code> 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
LineStyle	线条类型	{-}: 实线 --: 虚线 .: 点线 -.: 点划线 none: 无
LineWidth	线条粗细	标量, 以点 (1/72 英寸) 为单位
Marker	标记符号	'*': 星号 '.': 点号 'o': 圆圈 'x': 叉 '+'.: 加号 's': 小正方形 'd': 菱形 'v': 下三角 '^': 上三角 '<': 左三角 '>': 右三角 'h': 六角形 'p': 五角形

(续)

属性名	功    能	属性取值
MarkerEdgeColor	标记符号的边缘颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名
MarkerSize	标记符号的大小	标量, 以点 (1/72 英寸) 为单位
Parent	父对象	句柄
MarkerFaceColor	标记符号的表面颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名
Selected	对象是否被选中	on: 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄。 off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
Tag	用户指定的对象标记	字符串
Type	对象类型	line
UIContextMenu	把图形和一个上下文菜单联系起来	图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
Visible	线条是否可见	{on}: 可见 off: 不可见
XData	X 方向坐标数据	向量
YData	Y 方向坐标数据	向量
ZData	Z 方向坐标数据	向量

表 B-8 文本属性

属性名	功能	属性取值
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在文本上时按下鼠标按钮, 这时要调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Children	文本对象无子对象	
Clipping	是否修剪超出坐标轴边框的文本	{on}: 不显示超出坐标轴边框的文本 off: 不修剪
Color	文本颜色	RGB 向量或 MATLAB 中固定的颜色名。
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名

(续)

属性名	功 能	属性取值
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Editing	允许或不允许对文本进行编辑	{off}: 不允许; on: 允许
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
HitTest	指定文本是否能够通过鼠标单击被选中, 成为当前对象。	{on}: 文本可以被鼠标选中为当前对象, 即成为 geo 命令和图形属性 CurrentObject 的返回值。 off: 鼠标在文本上单击后, 选择它下面的对象
EraseMode	擦除方式	{normal}: 重画影响显示的作用区域, 以保证所有的对象正确地画出, 这是最精确的, 也是最慢的一种模式  background: 通过以图形背景色重画文本来消除文本。这会破坏被消除的文本后面的对象  xor: 用文本下屏幕的颜色执行异或运算, 画出和消除文本。当画在其它对象上时, 可能造成不正确的颜色  none: 当移动或删除文本时该文本不会被消除
Extent	文本的位置和尺寸	四元素向量: [文本矩形区域左下角的横坐标, 文本矩形区域左下角的纵坐标, 宽度, 高度]
FontAngle	字体角度	{normal}: 正常 italic: 斜体 oblique: 在有些系统中为斜体
FontName	字体名	字符串, 缺省值是 Helvetica
FontSize	字体尺寸	整数
FontUnits	尺寸单位	{points}: 点, 1/72 英寸 normalized: 归一化 inches: 英寸 centimeters: 厘米 pixels: 像素
FontWeight	字体的粗细	{normal}: 正常 bold: 粗体 light: 淡字体 demi: 适中或粗体
HorizontalAlignment	文本的水平排列方式	{left}: 靠左 center: 居中 right: 靠右
Interpreter	把文本作为特征字符串 (TeX) 处理还是作为纯文本处理 (参见第六章 6.3 节)	{tex}: 作为特征字符串处理 none: 纯文本

(续)

属性名	功 能	属性取值
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
Parent	父对象	句柄
Position	文本的位置	[x, y, z]
Rotation	文本的旋转角度	度数
Selected	对象是否被选中	on: 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄。 off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
String	文本字符串	
Tag	用户指定的对象标记	字符串
Type	对象类型	text
UIContextMenu	把文本和一个上下文菜单联系起来	在文本所在的图形中建立的上下文菜单句柄
Units	区域和位置数据的单位	{data}: 文本的父对象坐标轴的数据单位 pixels: 像素 normalized: 归一化坐标, 屏幕的左下角为[0 0], 右上角为[1 1] inches: 英寸 centimeters: 厘米 points: 点, 等于 1/72 英寸
UserData	用户指定的数据	矩阵或字符串
VerticalAlignment	文本的垂直排列方式	{middle}: 文本区域的中央放在指定位置 top: 文本区域的顶部放在指定位置 cap: 文本中大写字母的顶部在指定位置 baseline: 文本中字体的基线在指定位置 bottom: 文本区域的底部在指定位置
Visible	文本是否可见	{on}: 可见 off: 不可见

表 B-9 矩形属性

属性名	功 能	属性取值
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在矩形区域上时按下鼠标按钮, 这时要调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Parent	父对象	句柄
Children	矩形对象无子对象	
Clipping	是否修剪超出坐标轴边框的矩形	{on}: 不显示超出坐标轴边框的矩形 off: 不修剪

(续)

属性名	功 能	属性取值
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
Curvature	矩形的水平和垂直曲率数值	[x, y]: x, y 可以是 0 到 1 之间的数, 0 表示不弯曲, 1 表示最大弯曲
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
EdgeColor	矩形边框的颜色	{RGB 向量或 MATLAB 中固定的颜色名} none: 无颜色
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
EraseMode	擦除方式	{normal}: 重画影响显示的作用区域, 以保证所有的对象正确地画出。这是最精确的, 也是最慢的一种模式  background: 通过以图形背景色重画矩形来消除矩形。这会破坏被消除的矩形后面的对象  xor: 用矩形下屏幕的颜色执行异或运算, 画出和消除矩形。当画在其它对象上时, 可能造成不正确的颜色  none: 当移动或删除矩形时该矩形不会被消除
FaceColor	矩形表面的颜色	{none}: 无颜色 {RGB 向量或 MATLAB 中固定的颜色名}
HitTest	指定文本是否能够通过鼠标单击被选中, 成为当前对象。	{on}: 文本可以被鼠标选中为当前对象, 即成为 gco 命令和图形属性 CurrentObject 的返回值。 off: 鼠标在文本上单击后, 选择它下面的对象
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
LineStyle	线条类型	{-}: 实线 {--}: 虚线 {:}: 点线 {-}: 点划线 none: 无
LineWidth	线条粗细	标量, 以点 (1/72 英寸) 为单位
Position	矩形的位置和尺寸	四元素向量: [矩形区域左下角的横坐标, 矩形区域左下角的纵坐标, 宽度, 高度]
Selected	对象是否被选中	on: 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄, off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
Tag	用户指定的对象标记	字符串
Type	对象类型	rectangle
UIContextMenu	把矩形对象和一个上下文菜单联系起来	在矩形对象所在的图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
Visible	矩形对象是否可见	{on}: 可见 off: 不可见

表 B-10 表面属性

属性名	功 能	属性取值
ButtonDownFcn	当鼠标指针在表面区域上时按下鼠标按钮, 这时要调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
AmbientStrength	环境光照的强度	0 到 1 之间的标量
BackFaceLighting	表面光照控制	unlit: 无光照 flat: 正常方式光照 reverselit: 好象表面的顶点朝向照相机的光照效果
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
CData	顶点位置(即 Zdata 中的数据对应的点)的颜色值	颜色矩阵
CDataMapping	设置颜色映射的方式	{scaled}: 按比例缩放, 根据坐标轴的 Clim 属性设置的范围线性映射数据和颜色 direct: 直接映射, 用 Cdata 中设置的颜色数据作为索引直接调用颜色映象中的颜色。这时的颜色数据应该在 1 到颜色映象矩阵的长度之间, 小于 1 的数据映射为颜色映象的第一个颜色, 大于矩阵长度的数据映射为颜色映象的最后一个颜色。
Children	表面对象无子对象	
Clipping	是否修剪超出坐标轴边框的表面	{on}: 不显示超出坐标轴边框的表面 off: 不修剪
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DiffuseStrength	由坐标轴中的光源对象产生的散射光的强度	0 到 1 之间的标量
EdgeColor	组成整个表面的单独面的边缘颜色	{RGB 向量或 MATLAB 中固定的颜色名} none: 不画出边缘 flat: 颜色平滑处理 interp: 颜色插值处理
EdgeLighting	组成整个表面的单独面的边缘光照	{none}: 光照不影响边缘 flat: 光线均匀穿过每个单独表面的边缘 gouraud: 线性插值处理 phong: 线性插值, 并计算每个象素点的反射系数
LineStyle	线条类型	{-}: 实线 --: 虚线 --: 点线 -.: 点划线 none: 无

(续)

属性名	功 能	属性取值
LineWidth	线条粗细	标量, 以点 (1/12 英寸) 为单位
EraseMode	擦除方式	{normal}; 重画影响显示的作用区域, 以保证所有的对象正确地画出。这是最精确的, 也是最慢的一种模式 background; 通过以图形背景色重画表面来消除表面。这会破坏被消除的表面后面的对象 xor; 用表面下屏幕的颜色执行异或运算, 画出和消除表面。当画在其它对象上时, 可能造成不正确的颜色 none; 当移动或删除表面时该表面不会被消除
FaceColor	表面的颜色	{flat}; 颜色平滑处理 none; 不画出面 interp; 颜色插值处理 RGB 向量或 MATLAB 中固定的颜色名
FaceLighting	表面光照	{none}; 光照不影响表面 flat; 光线均匀穿过每个单独表面 gouraud; 线性插值处理 phong; 线性插值, 并计算每个象素点的反射系数
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}; 通过回调程序或函数使句柄可见。 on; 总是可见 off; 总是不可见
HitTest	指定表面对象是否能够通过鼠标单击被选中, 成为当前对象。	{on}; 表面对象可以被鼠标选中为当前对象, 即成为 gco 命令和图形属性 CurrentObject 的返回值。 off; 鼠标在表面对象上单击后, 选择它下面的对象
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}; 可以被其它回调程序中断。 off; 不能被其它回调程序中断。
Marker	标记在每个单独面的顶点符号	'*'; 星号 '.'; 点号 'o'; 圆圈 'x'; 叉 '+'; 加号 's'; 小正方形 'd'; 菱形 'v'; 下三角 '^'; 上三角 '<'; 左三角 '>'; 右三角 'h'; 六角形 'p'; 五角形

(续)

属性名	功 能	属性取值
MarkerEdgeColor	标记符号的边缘颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名
MarkerSize	标记符号的大小	标量, 以点 (1/72 英寸) 为单位
MarkerFaceColor	标记符号的表面颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名
MeshStyle	边缘线的画法	{both}: 行和列的边缘线都画 row: 只画行的边缘线 column: 只画列的边缘线
NormalMode	法线向量的指定模式	{auto}: 根据数据自动指定 manual: 用户在属性 VertexNormals 中指定
Parent	父对象	句柄
Selected	对象是否被选中	on: 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄, off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
SpecularColorReflectance	镜面反射光的颜色设置	0 到 1 之间的标量: 0 表示镜面反射光的颜色依赖于反射光线的对象颜色和光源的颜色; 1 表示镜面反射光的颜色只依赖于光源的颜色。
SpecularExponent	镜面反射的粗糙度指数	大于或等于 1
SpecularStrength	镜面反射光的强度	0 到 1 之间的标量
Tag	用户指定的对象标记	字符串
Type	对象类型	surface
UIContextMenu	把表面对象和一个上下文菜单联系起来	在表面对象所在的图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
VertexNormals	组成表面对象的单独小表面法线向量的数据	向量或矩阵
Visible	表面对象是否可见	{on}: 可见 off: 不可见
XData	X 轴数据	向量或矩阵
YData	Y 轴数据	向量或矩阵
ZData	Z 轴数据	向量或矩阵

表 B-11 补片属性

属性名	功 能	属性取值
AmbientStrength	环境光照的强度	0 到 1 之间的标量
BusyAction	回调程序中断方式，指当一个回调程序执行时，如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on，那么后面的中断请求自动以队列的方式进行；如果该属性设为 off，那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在矩形区域上时按下鼠标按钮，这时要调用的程序	字符串，可以是 MATLAB 表达式或 M 文件名
Clipping	是否修剪超出坐标轴边框的补片对象	{on}: 不显示超出坐标轴边框的补片 off: 不修剪
CDATA	补片的颜色	颜色矩阵
CDATAMapping	设置颜色映射的方式	{scaled}: 按比例缩放，根据坐标轴的 Clim 属性设置的范围线性映射数据和颜色  direct: 直接映射，用 Cdata 中设置的颜色数据作为索引直接调用颜色映象中的颜色。这时的颜色数据应该在 1 到颜色映象矩阵的长度之间，小于 1 的数据映射为颜色映象的第一个颜色，大于矩阵长度的数据映射为颜色映象的最后一个颜色。
Children	补片对象无子对象	
CreateFcn	建立对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串，可以是 MATLAB 表达式或 M 文件名
DiffuseStrength	由坐标轴中的光源对象产生的散射光的强度	0 到 1 之间的标量
EdgeColor	补片边缘的颜色	{RGB 向量或 MATLAB 中固定的颜色名}  none: 不画出边缘 flat: 颜色平滑处理 interp: 颜色插值处理
EdgeLighting	补片边缘的光照	{none}: 光照不影响边缘 flat: 光线均匀穿过每个补片的边缘 gouraud: 线性插值处理 phong: 线性插值，并计算每个像素点的反射系数
EraseMode	擦除方式	{normal}: 重画影响显示的作用区域，以保证所有的对象正确地画出。这是最精确的，也是最慢的一种模式  background: 通过以图形背景色重画补片来消除补片。这会破坏被消除的补片后面的对象  xor: 用补片下屏幕的颜色执行异或运算，画出和消除补片。当画在其它对象上时，可能造成不正确的颜色  none: 当移动或删除补片时该补片不会被消除
FaceColor	补片表面的颜色	{flat}: 颜色平滑处理  none: 不画出面 interp: 颜色插值处理 RGB 向量或 MATLAB 中固定的颜色名

(续)

属性名	功 能	属性取值
FaceLighting	补片表面的光照	{none}: 光照不影响表面 flat: 光线均匀穿过补片 gouraud: 线性插值处理 phong: 线性插值，并计算每个象素点的反射系数
FaceVertexCData	补片表面和顶点的颜色	矩阵
Faces	补片顶点的连接	矩阵
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
HitTest	指定补片对象是否能够通过鼠标单击被选中，成为当前对象。	{on}: 补片对象可以被鼠标选中为当前对象，即成为 gco 命令和图形属性 CurrentObject 的返回值。 off: 鼠标在补片对象上单击后，选择它下面的对象
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
LineStyle	线条类型	{-}: 实线 --: 虚线 .: 点线 -.: 点划线 none: 无
LineWidth	线条粗细	标量，以点(1/72 英寸)为单位
Marker	标记在补片顶点的符号	'*': 星号 '.': 点号 'o': 圆圈 'x': 叉 '+'.: 加号 's': 小正方形 'd': 菱形 'v': 下三角 '^': 上三角 '<': 左三角 '>': 右三角 'h': 六角形 'p': 五角形
MarkerEdgeColor	标记符号的边缘颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名
MarkerFaceColor	标记符号的表面颜色	{auto}: 自动 none: 无颜色 RGB 向量或 MATLAB 中固定的颜色名

(续)

属性名	功 能	属性取值
MarkerSize	标记符号的大小	标量, 以点 (1/72 英寸) 为单位
NormalMode	法线向量的指定模式	{auto}: 根据数据自动指定 manual: 用户在属性 VertexNormals 中指定
SpecularColorReflectance	镜面反射光的颜色设置	0 到 1 之间的标量; 0 表示镜面反射光的颜色依赖于反射光线的对象颜色和光源的颜色; 1 表示镜面反射光的颜色只依赖于光源的颜色。
SpecularExponent	镜面反射的粗糙度指数	大于或等于 1
SpecularStrength	镜面反射光的强度	0 到 1 之间的标量
Tag	用户指定的对象标记	字符串
Type	对象类型	patch
Selected	对象是否被选中	on: 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄; off: 未被选中
SelectionHighlight	是否突出选中的对象	{on}: 突出 off: 不突出
Parent	父对象	句柄
UIContextMenu	把补片对象和一个上下文菜单联系起来	在补片对象所在的图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
VertexNormals	补片对象的表面法线向量的数据	向量或矩阵
Vertices	顶点坐标	[x, y, z]矩阵
Visible	补片对象是否可见	{on}: 可见 off: 不可见
XData	X 轴数据	向量或矩阵
YData	Y 轴数据	向量或矩阵
ZData	Z 轴数据	向量或矩阵

表 B-12 光源属性

属性名	功 能	属性取值
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	对光源对象无效	
Children	光源对象无子对象	
Clipping	对光源对象无效	
Color	光源颜色	RGB 向量或 MATLAB 中固定的颜色名。
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名

(续)

属性名	功 能	属性取值
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}: 通过回调程序或函数使句柄可见。 on: 总是可见 off: 总是不可见
HitTest	对光源对象无效	
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性不能影响属性 DeleteFcn 中的设置。	{on}: 可以被其它回调程序中断。 off: 不能被其它回调程序中断。
Position	光源的位置	向量[x, y, z], 单位是坐标轴的数据单位
Style	光源类型	{infinite}: 无限光源(平行) local: 局部光源
Parent	父对象	句柄
Selected	对光源对象无效	
SelectionHighlight	对光源对象无效	
Tag	用户指定的对象标记	字符串
Type	对象类型	light
UIContextMenu	把光源对象和一个上下文菜单联系起来	在光源对象所在的图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
Visible	光源对象是否可见	{on}: 可见 off: 不可见

表 B-13 图象属性

属性名	功 能	属性取值
BusyAction	回调程序中断方式, 指当一个回调程序执行时, 如何处理后面的回调程序的中断请求。如果正在执行回调程序的对象的属性 Interruptible 设为 on, 那么后面的中断请求自动以队列的方式进行; 如果该属性设为 off, 那么就需要设置正在执行回调程序的对象的 BusyAction 属性。	{queue}: 排成队列等待执行 cancel: 拒绝后面的中断请求
ButtonDownFcn	当鼠标指针在图象对象上时按下鼠标按钮, 这时要调用的程序	字符串, 可以是 MATLAB 表达式或 M 文件名
CData	图象的颜色数据	颜色矩阵
CDataMapping	设置颜色映射的方式	{direct}: 直接映射, 用 Cdata 中设置的颜色数据作为索引直接调用颜色映象中的颜色。这时的颜色数据应该在 1 到颜色映象矩阵的长度之间, 小于 1 的数据映射为颜色映象的第一个颜色, 大于矩阵长度的数据映射为颜色映象的最后一个颜色。 scaled: 按比例缩放, 根据坐标轴的 Clim 属性设置的范围线性映射数据和颜色

(续)

属性名	功    能	属性取值
CreateFcn	建立对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
DeleteFcn	删除对象时执行的回调程序	字符串, 可以是 MATLAB 表达式或 M 文件名
EraseMode	擦除方式	{normal}; 重画影响显示的作用区域, 以保证所有的对象正确地画出。这是最精确的, 也是最慢的一种模式  background; 通过以坐标轴或图形背景色重画图象来消除图象。这会破坏被消除的图象后面的对象  xor; 用图象下屏幕的颜色执行异或运算, 画出和消除图象。当画在其它对象上时, 可能造成不正确的颜色  none; 当移动或删除图象时该图象不会被消除
Clipping	是否修剪超出坐标轴边框的图象	{on}; 不显示超出坐标轴边框的图象  off; 不修剪
Children	图象对象无子对象	
HandleVisibility	指定何时对象句柄可见。此属性用于控制从 MATLAB 命令行或图形用户界面对对象句柄的访问。	{callback}; 通过回调程序或函数使句柄可见。 on; 总是可见  off; 总是不可见
HitTest	指定图象对象是否能够通过鼠标单击被选中, 成为当前对象。	{on}; 图象对象可以被鼠标选中为当前对象, 即成为 gco 命令和图形属性 CurrentObject 的返回值。  off; 鼠标在图象对象上单击后, 选择它下面的对象
Interruptible	回调程序中断模式。控制执行的回调程序是否可以被后面回调程序的请求中断。此属性只影响 ButtonDownFcn 的回调函数。	{on}; 可以被其它回调程序中断。  off; 不能被其它回调程序中断。
Parent	父对象	句柄
Selected	对象是否被选中	on; 被选中时如果 SelectionHighlight 也设为 on 则显示对象句柄。  off; 未被选中
SelectionHighlight	是否突出选中的对象	{on}; 突出  off; 不突出
Tag	用户指定的对象标记	字符串
Type	对象类型	image
UIContextMenu	把图象对象和一个上下文菜单联系起来	在图象对象所在的图形中建立的上下文菜单句柄
UserData	用户指定的数据	矩阵或字符串
Visible	图象对象是否可见	{on}; 可见  off; 不可见
XData	X 轴数据	向量或矩阵
YData	Y 轴数据	向量或矩阵

## 附录 C Simulink 模型模块参数

表 C-1 模型参数

参数	含义	取值
Name	模型名	文本
Version	Simulink的版本	版本号
SimParamPage	Simulation Parameters对话框显示哪一页	{Solver}   WorkspaceI/O   Diagnostics
SampleTimeColors	是否给采样时间不同的模块使用不同的颜色，和菜单Format下的Sample Time Colors选项作用相同	on   {off}
InvariantConstants	恒定常数设置	on   {off}
WideVectorLines	用粗线表示向量，和菜单Format下的Wide Vector Lines选项作用相同	on   {off}
ShowLineWidths	显示线的宽度，和菜单Format下的how Line Widths选项作用相同	on   {off}
PaperOrientation	打印纸的方向	portrait   {landscape}
PaperPosition	框图在打印页面上的位置	[left, bottom, width, height]
PaperSizeMode	打印纸的放置方式	auto   {manual}
PaperSize	打印纸的尺寸	[width height] (只读)
PaperType	打印纸的类型	{usletter}   uslegal   a0   a1   a2   a3   a4   a5   b0   b1   b2   b3   b4   b5   arch-A   arch-B   arch-C   arch-D   arch-E   A   B   C   D   E   tabloid
PaperUnits	打印纸的尺寸的单位	normalized   {inches}   centimeters   points
StartTime	仿真开始时间	标量 {0.0}
StopTime	仿真停止时间	标量 {10.0}
Solver	解法	{ode45}   ode23   ode113   odel5s   ode23s   ode5   ode4   ode3   ode2   odel   FixedStepDiscrete   VariableStepDiscrete
RelTol	相对误差容限	标量 {1e-3}
AbsTol	绝对误差容限	标量 {1e-6}
Refine	细化因子	标量 {1}
MaxStep	最大仿真步长	标量 {auto}
InitialStep	初始步长	标量 {auto}
FixedStep	固定步长大小	标量 {auto}
MaxOrder	算法ode15s的最大阶数	1   2   3   4   {5}
OutputOption	输出选项	AdditionalOutputTimes   {RefineOutputTimes}   SpecifiedOutputTimes
OutputTimes	所选输出选项的值	向量 {[]}

(续)

参数	含义	取值
LoadExternalInput	从工作空间载入输入数据	on   {off}
ExternalInput	时间和输入变量名	标量 或 向量 [t, u]
SaveTime	保存仿真时间	{on}   off
TimeSaveName	仿真时间名	变量 {tout}
SaveState	保存状态值	on   {off}
StateSaveName	输出状态名	变量 {xout}
SaveOutput	保存仿真输出	{on}   off
OutputSaveName	仿真输出名	变量 {yout}
LoadInitialState	载入初始状态	on   {off}
InitialState	初始状态名或数值	变量 或 向量 {xInitial}
SaveFinalState	保存最终状态	on   {off}
FinalStateName	最终状态名	变量 {xFinal}
LimitMaxRows	限制输出	on   {off}
MaxRows	最大输出行数	标量 {1000}
Decimation	抽选因子	标量 {1}
AlgebraicLoopMsg	代数循环诊断	none   {warning}   error
MinStepSizeMsg	最小步长大小诊断	{warning}   error
UnconnectedInputMsg	未连接输入端口诊断	none   {warning}   error
UnconnectedOutputMsg	未连接输出端口诊断	none   {warning}   error
UnconnectedLineMsg	未连接线诊断	none   {warning}   error
ConsistencyChecking	一致性检验	on   {off}
ZeroCross	零交叉检验	{on}   off
BooleanDataType	允许逻辑类型的数据	on   {off}
BufferReuse	允许再次使用模块的输入输出缓冲区	{on}   off

表 C-2 模型回调参数

模型回调参数	何时执行
CloseFcn	关闭模型框架前执行
PreLoadFcn	载入模型之前执行
PostLoadFcn	载入模型后执行
PreSaveFcn	模型被保存前执行
PostSaveFcn	模型被保存后执行
InitFcn	模型仿真开始时执行
StartFcn	开始仿真前执行
StopFcn	终止仿真后执行

表 C-3 通用模块参数

参数	含义	取值
Name	模块名	字符串
Type	仿真对象类型	'block'
Parent	模块所在系统的名字	字符串
BlockType	模块类型	文本
BlockDescription	模块说明文档	文本
Description	用户指定的说明文档	文本
InputPorts	输入端口位置数组	[h1,v1; h2,v2; ...]
OutputPorts	输出端口位置数组	[h1,v1; h2,v2; ...]
CompiledPortWidths	端口宽度的结构	标量和向量
Orientation	模块的方向	{right}   left   down   up
ForegroundColor	模块名字、图标、边框、输出信号、信号标记的颜色	{black}   white   red   green   blue   cyan   magenta   yellow   gray   lightBlue   orange   darkGreen
BackgroundColor	模块的背景色	black   {white}   red   green   blue   cyan   magenta   yellow   gray   lightBlue   orange   darkGreen
DropShadow	显示阴影	{off}   on
NamePlacement	模块名字的位置	{normal}   alternate
FontName	字体	{Helvetica}
FontSize	字体大小	{10}
FontWeight	字体粗细	light   {normal}   demi   bold
FontAngle	字体角度	{normal}   italic   oblique
Position	模块在模型窗口中的位置	向量[左 上 右 下]
ShowName	是否显示模块名	{on}   off
Tag	用户定义的标记	字符串
UserData	用户数据，可以是任何 MATLAB 数据类型	[]
Selected	模块的被选中状态	on   {off}
LinkStatus	模块的连接状态	none   resolved   unresolved   implicit
AttributesFormatString	在模块下面显示的字符串	字符串

表 C-4 模块回调参数

模块回调参数	何时执行
CloseFcn	当模块用 close_system 命令关闭时执行
CopyFcn	当模块被复制之后执行。这个参数和子系统模块是递归的，这就是说，如果一个定义了参数 CopyFcn 的模块包含在子系统中，那么当子系统被复制时，该模块中指定的程序也会执行。另外，用 add_block 命令复制模块时，程序也会执行。
DeleteFcn	在模块被删除之前执行，和子系统模块递归。
InitFcn	在计算出模块参数之前执行
LoadFcn	在模块框图载入之后执行，和子系统模块递归。
ModelCloseFcn	在模块框图被关闭之前执行，和子系统模块递归。
MoveFcn	当模块被移动或改变大小时执行
NameChangeFcn	在模块名或路径改变之后执行，和子系统模块递归。
OpenFcn	当模块被打开时执行。
ParentCloseFcn	在关闭包含该模块的子系统之前执行
PreSaveFcn	在保存模块框图之前执行，和子系统模块递归。
PostSaveFcn	在保存模块框图之后执行，和子系统模块递归。
StartFcn	在仿真开始之前执行
StopFcn	在仿真中断时执行
UndoDeleteFcn	当撤销对一个模块的删除操作时执行

表 C-5 输入源模块参数

模块	参数	取值
Constant	Value	标量或向量 {1}
Signal Generator	WaveForm	{sine}   square   sawtooth   random
	Amplitude	标量或向量 {1}
	Frequency	标量或向量 {1}
	Units	{Hz}   rad/sec
Step	Time	标量或向量 {1}

(续)

模 块	参 数	取 值
	Before	标量或向量 {0}
	After	标量或向量 {1}
Sine Wave		
	Amplitude	标量或向量 {1}
	Frequency	标量或向量 {1}
	Phase	标量或向量 {0}
	SampleTime	标量{-1} 或向量
Digital Clock		
	SampleTime	标量{1} 或向量
From File		
	FileName	Filename {untitled.mat}
From Workspace		
	VariableName	矩阵 {[T,U]}
Random Number		
	Seed	标量或向量 {0}
Uniform Random Number		
	Minimum	标量或向量 {-1}
	Maximum	标量或向量 {1}
	Seed	标量或向量 {0}
	SampleTime	标量或向量 {0}
Clock	无可设置参数	
Discrete Pulse Generator	(masked)	
Pulse Generator	(masked)	
Chirp Signal	(masked)	
Repeating Sequence	(masked)	
Ramp	(masked)	
Band-Limited White Noise	(masked)	

表 C-6 接收模块参数

模 块	参 数	取 值
Scope		
	Location	向量 {[左 上 右 下]}
	Open	{off}   on
	NumInputPorts	正整数
	TickLabels	{on}   off
	ZoomMode	{on}   xonly   yonly
	AxisTitles	向量 {auto}
	Grid	{on}   off
	TimeRange	标量 {auto}
	YMin	标量 {-5}
	YMax	标量 {5}
	SaveToWorkspace	{off}   on
	SaveName	变量 {ScopeData}
	DataFormat	{matrix   structure}
	LimitMaxRows	{on}   off
	MaxRows	标量 {5000}
	Decimation	标量 {1}
	SampleInput	{off}   on
	SampleTime	标量{0} 或 向量
Display		
	Format	{short}   long   short_e   long_e   bank
	Decimation	标量 {1}
	Floating	{off}   on
	SampleTime	标量{-1} 或 向量
To File		
	Filename	文件名 {untitled.mat}
	MatrixName	变量 {ans}
	Decimation	标量 {1}
	SampleTime	标量{-1} 或 向量
To Workspace		
	VariableName	变量 {simout}
	Buffer	标量 {inf}
	Decimation	标量 {1}
	SampleTime	标量{-1} 或 向量
Stop Simulation	无可设置参数	
XY Graph	(masked)	

表 C-7 连续系统模块参数

模 块	参 数	取 值
Integrator	ExternalReset	{none}   rising   falling   either
	InitialConditionSource	{internal}   external
	InitialCondition	标量或向量 {0}
	LimitOutput	{off}   on
	UpperSaturationLimit	标量或向量 {inf}
	LowerSaturationLimit	标量或向量 {-inf}
	ShowSaturationPort	{off}   on
	ShowStatePort	{off}   on
State-Space	AbsoluteTolerance	标量 {auto}
	A	矩阵 {1}
	B	矩阵 {1}
	C	矩阵 {1}
	D	矩阵 {1}
	X0	向量 {0}
Memory	X0	标量或向量 {0}
Transfer Fcn	Numerator	向量或矩阵 {[1]}
	Denominator	向量 {[1 1]}
Variable Transport Delay	MaximumDelay	标量或向量 {10}
	InitialInput	标量或向量 {0}
	MaximumPoints	标量 {1024}
Transport Delay	DelayTime	标量或向量 {1}
	InitialInput	标量或向量 {0}
	BufferSize	标量 {1024}
Zero-Pole	Zeros	向量 {[1]}
	Poles	向量 {[0 -1]}
	Gain	向量 {[1]}
Derivative	无可设置参数	

表 C-8 离散系统模块参数

模 块	参 数	取 值
Zero-Order Hold		
	SampleTime	标量{1} 或向量
Unit Delay		
	X0	标量或向量{0}
	SampleTime	标量{1} 或向量
Discrete-Time Integrator		
	IntegratorMethod	{ForwardEuler}   BackwardEuler   Trapezoidal
	ExternalReset	{none}   rising   falling   either
	InitialConditionSource	{internal}   external
	InitialCondition	标量或向量{0}
	LimitOutput	{off}   on
	UpperSaturationLimit	标量或向量{inf}
	LowerSaturationLimit	标量或向量{-inf}
	ShowSaturationPort	{off}   on
	ShowStatePort	{off}   on
	SampleTime	标量{1} 或向量
Discrete State-Space		
	A	矩阵{1}
	B	矩阵{1}
	C	矩阵{1}
	D	矩阵{1}
	X0	向量{0}
	SampleTime	标量{1} 或向量
Discrete Filter		
	Numerator	向量或矩阵{[1]}
	Denominator	向量{[1 2]}
	SampleTime	标量{1} 或向量
Discrete Transfer Fcn		
	Numerator	向量或矩阵{[1]}
	Denominator	向量{[1 0.5]}
	SampleTime	标量{1} 或向量
Discrete Zero-Pole		
	Zeros	向量{[1]}
	Poles	向量{[0 0.5]}
	Gain	标量{[1]}
	SampleTime	标量{1} 或向量
First-Order Hold	(masked)	

表 C-9 信号与系统模块参数

模 块	参 数	取 值
In1	Port	标量 {1}
	PortWidth	标量 {-1}
	SampleTime	标量{-1} 或向量
Out1	Port	标量 {1}
	OutputWhenDisabled	{held}   reset
	InitialOutput	标量或向量 {0}
Enable	StatesWhenEnabling	{held}   reset
	ShowOutputPort	{off}   on
	TriggerType	{rising}   falling   either   function-call
Trigger	ShowOutputPort	{off}   on
	Inputs	标量或向量 {3}
	Outputs	标量或向量 {3}
From	GotoTag	标记 {A}
	GotoTag	标记 {A}
Goto Tag Visibility	GotoTag	标记 {A}
	TagVisibility	{local}   scoped   global
Goto	GotoTag	标记 {A}
	TagVisibility	{local}   scoped   global
Data Store Read	DataStoreName	标记 {A}
	SampleTime	标量{-1} 或向量

(续)

模 块	参 数	取 值
Data Store Memory		
	DataStoreName	标记 {A}
	InitialValue	向量 {0}
Data Store Write		
	DataStoreName	标记 {A}
	SampleTime	标量{-1} 或 向量
SubSystem		
	ShowPortLabels	{on}   off
Configurable Subsystem		
	Choice	字符串
	LibraryName	字符串
Hit Crossing		
	HitCrossingOffset	标量或向量 {0}
	HitCrossingDirection	rising   falling   {either}
	ShowOutputPort	{on}   off
IC		
	Value	标量或向量 {1}
Probe		
	ProbeWidth	{on}   off
	ProbeSampleTime	{on}   off
	ProbeComplexSignal	{on}   off
Width	无可设置参数	
Selector	无可设置参数	
Bus Selector	无可设置参数	
Model Info	无可设置参数	
Data Type Conversion	无可设置参数	
Terminator	无可设置参数	
Ground	无可设置参数	
Function-Call Generator	无可设置参数	
Merge	无可设置参数	

表 C-10 数学运算模块参数

模 块	参 数	取 值
Sum		
	Inputs	标量或符号列 {++}
Product		
	Inputs	标量 {2}
Gain		
	Gain	标量或向量 {1}
Rounding Function		
	Operator	{floor}   ceil   round   fix
Math Function		
	Operator	{exp}   log   log10   square   sqrt   pow   reciprocal   hypot   rem   mod
Trigonometric Function		
	Operator	{sin}   cos   tan   asin   acos   atan   atan2   sinh   cosh   tanh
MinMax		
	Function	{min}   max
	Inputs	标量 {1}
Combinatorial Logic		
	TruthTable	矩阵 {{0 0;0 1;0 1;1 0;0 1;1 0;1 0;1 1}}
Logical Operator		
	Operator	{AND}   OR   NAND   NOR   XOR   NOT
	Inputs	标量 {2}
Relational Operator		
	Operator	==   !=   <   {<=}   >=   >
Complex to Magnitude-Angle	无可设置参数	
Magnitude-Angle to Complex	无可设置参数	
Complex to Real-Imag	无可设置参数	
Real-Imag to Complex	无可设置参数	
Abs	无可设置参数	
Sign	无可设置参数	
Algebraic Constraint	(masked)	
Slider Gain	(masked)	
Matrix Gain	(masked)	
Dot Product	(masked)	



