



张 琨 高思超 毕 靖 编著

MATLAB 2010

从入门到精通



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

MATLAB 2010

从入门到精通

张 琨 高思超 毕 靖 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书对 MATLAB 2010 进行了详细的介绍和讲解,以实际应用为导向,力求做到由简入繁,并达到快速入门和迅速提高的目的。本书共分为 2 篇,即基础篇和提高篇。前 7 章为基础篇,讲解有关 MATLAB 的基础知识,包括 MATLAB 的安装、卸载及系统功能的简述, MATLAB 的数值运算、符号运算和图形功能, M 文件编程、Simulink 框图仿真以及图形用户界面等内容。第 8 章至第 11 章为提高篇,第 8 章和第 9 章分别介绍了 MATLAB 2010 的科学计算, S-函数的概念、原理和应用。第 10 章和第 11 章分别介绍了物理系统的建模和仿真以及 MATLAB 外部接口。

本书条理明晰,深入浅出,并配有大量实用的例子,适合使用 MATLAB 的本科生、研究生和教师以及广大科技工作者作为参考用书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

MATLAB 2010 从入门到精通 / 张琨,高思超,毕靖编著. —北京:电子工业出版社,2011.5

ISBN 978-7-121-13413-5

I. ①M… II. ①张… ②高… ③毕… III. ①计算机辅助计算—软件包, MATLAB 2010 IV. ①TP391.75

中国版本图书馆 CIP 数据核字(2011)第 078828 号

责任编辑:李红玉

印 刷: 三河市鑫金马印装有限公司
装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编:100036

北京市海淀区翠微东里甲 2 号 邮编:100036

开 本:787×1092 1/16 印张:31.5 字数:823 千字

印 次:2011 年 5 月第 1 次印刷

定 价:64.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlt@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

前 言

MATLAB 是 Matrix Laboratory（矩阵实验室）的缩写，它是以著名的线性代数软件包 LINPACK 和特征值计算软件包 EISPACK 为基础发展而来的，于 1984 年由 MathWorks 公司推出，2010 年 3 月发布了 MATLAB 7.10（MATLAB R2010a）。MATLAB 是一种开放型程序设计语言，拥有集计算、可视化、编程和仿真于一体的开发环境。同时它具有功能强、学习易、效率高等特点，可以方便地应用到科学计算、算法研究、数据采集和处理、系统建模和仿真、数据分析和可视化、科学和工程绘图、图形用户界面建立等方面，是目前世界上最流行的仿真计算软件之一，广泛应用于各个领域。

目前的 MATLAB 可以说是科技工作者必不可少的工具之一，掌握了这一重要工具将使日常的学习和工作事半功倍。MATLAB 已经逐步发展成具有通用性和可扩展性的操作平台，并为主要应用领域提供专用工具箱。本书主要从实际应用角度和快速入门角度对 MATLAB 2010 进行通用性介绍，没有局限于某些具体领域介绍某个或某几个工具箱，而着重于讲清和讲透通用内容，为具体应用打下坚实的基础。

全书分为入门篇和提高篇。入门篇包括前 7 章，通过入门篇使读者对 MATLAB 最基本的内容有较深刻的理解，能够初步应用 MATLAB；提高篇包括后 4 章，在入门篇的基础上有针对性地拓展了 MATLAB 的功能，通过提高篇读者能够对 MATLAB 的通用内容熟练掌握，从而充分利用 MATLAB 的功能。

入门篇的第 1 章主要是 MATLAB 的概述，包括历史沿革、安装、工作环境、通用命令和帮助查询系统等内容；第 2 章主要介绍 MATLAB 的数值运算，包括 MATLAB 数据类型、数组运算、矩阵运算、多项式运算、关系和逻辑运算等内容；第 3 章主要介绍 MATLAB 的符号运算，包括符号表达式表示、符号表达式运算、符号表达式微积分、符号表达式积分变换、符号矩阵运算等内容；第 4 章主要介绍 MATLAB 的图形功能，包括二维绘图、三维绘图、绘图处理、图形窗口控制等内容；第 5 章主要介绍 MATLAB 的 M 文件编程，包括与外部数据的交换、流程控制、脚本、函数、子函数、程序调试、性能分析等内容；第 6 章主要介绍 MATLAB 的 Simulink 仿真环境，包括模型建立、封装子系统、回调函数、仿真运行等内容；第 7 章主要介绍 MATLAB 的图形用户界面。

提高篇针对入门篇进行功能拓展，第 8 章可以看做第 2、3 章的拓展，主要介绍 MATLAB 的科学计算，包括方程求解、数据统计、多项式操作、插值、数值积分、优化计算等内容；第 9 章可以看做第 6 章的拓展，主要介绍 S-函数以拓展 Simulink 的应用，包括工作原理以及各种不同文件型的编写方法等内容；第 10 章主要介绍物理系统的建模和仿真，包括物理元件库、机械系统、电气系统以及多域物理系统实例等内容。第 11 章可以看做 MATLAB 平台的拓展，这是因为入门篇的所有内容都是 MATLAB 平台内的应用，包括与其他应用程序的交互；同时提高篇可以使读者灵活应用第 4 章介绍的编程内容，以及加深理解第 1 章介绍的 MATLAB 功能。

本书条理清晰、深入浅出，提供了大量的实用例子，适合作为学习或使用 MATLAB 这一

重要工具的本科生、研究生、教师以及广大科技工作者的参考书。

全书由张琨、高思超和毕靖编著，刘抒和王晓芳审校。本书在编著过程中，尽管编者竭尽全力，但由于自身水平有限和时间仓促，书中不尽如人意的地方和错误在所难免，敬请指正，不胜感激！

注：考虑到计算机编程与系统界面的上下文内容，为求一致一些符号排成正体，以便对应。

为方便读者阅读，若需要本书配套资料，请登录“北京美迪亚电子信息有限公司”(<http://www.medias.com.cn>)，在“资料下载”页面进行下载。

目 录

第一篇 基础篇

第 1 章 MATLAB 概述	1	2.4 多项式及其函数	87
1.1 MATLAB 简介与发展历史	1	2.4.1 多项式的建立和操作	87
1.1.1 MATLAB 的基本功能及特点	1	2.4.2 多项式运算	88
1.1.2 MATLAB R2010a 的新功能及特点	3	2.4.3 多项式展开	93
1.1.3 MATLAB 的发展历史	4	2.4.4 多项式拟合	94
1.2 MATLAB 的安装、退出与卸载	5	2.5 关系和逻辑及其运算	94
1.2.1 MATLAB 安装	5	2.5.1 关系和逻辑运算符	95
1.2.2 MATLAB 退出	10	2.5.2 关系和逻辑函数	97
1.2.3 MATLAB 卸载	10	2.5.3 NaN 和空矩阵	97
1.3 MATLAB 的目录结构	11	第 3 章 MATLAB 符号运算	100
1.4 MATLAB 的工作环境	11	3.1 符号运算入门	100
1.5 MATLAB 的通用命令简介	22	3.1.1 符号对象的创建函数	100
1.6 MATLAB 的工具箱简介	26	3.1.2 符号对象的创建	101
1.7 MATLAB 的帮助查询功能	28	3.1.3 符号运算中的运算符	105
第 2 章 MATLAB 数值计算	33	3.1.4 符号表达式中自变量的确定	106
2.1 数据类型	33	3.2 符号表达式运算	107
2.1.1 字符串 (String) 类型	33	3.2.1 提取分子和分母	107
2.1.2 数值 (Numeric) 类型	35	3.2.2 标准代数运算	108
2.1.3 函数句柄 (Handle)	38	3.2.3 复合符号函数运算	109
2.1.4 逻辑 (Logical) 类型	38	3.2.4 数值转换	110
2.1.5 结构体 (Structure) 类型	39	3.2.5 变量替换	112
2.1.6 细胞数组 (Cell) 类型	44	3.2.6 化简与格式化	113
2.2 数组及其函数	47	3.3 符号运算精度	120
2.2.1 数组的建立和操作	47	3.4 符号矩阵的计算	122
2.2.2 数组运算	55	3.4.1 基本算术运算	122
2.2.3 数组函数	60	3.4.2 线性代数运算	124
2.3 矩阵及其函数	61	3.4.3 科学计算	139
2.3.1 矩阵的建立和操作	62	3.5 符号表达式积分变换	147
2.3.2 矩阵运算	71	3.5.1 Fourier 变换及其反变换	147
2.3.3 矩阵函数	73	3.5.2 Laplace 变换及其反变换	149
2.3.4 稀疏矩阵及其运算	84	3.5.3 Z 变换及其反变换	152
		3.6 符号函数的图形绘制	155

3.6.1 符号函数曲线的绘制	155	5.4 脚本文件	234
3.6.2 符号函数等值线的绘制	157	5.5 函数文件	237
3.6.3 符号函数曲面图及表面图的 绘制	158	5.5.1 基本结构	237
3.7 符号方程的求解	160	5.5.2 输入/输出参数	238
3.7.1 代数方程的求解	160	5.5.3 子函数	241
3.7.2 微分方程的求解	162	5.5.4 私有函数	243
3.7.3 复合方程的求解	164	5.5.5 嵌套函数	245
3.7.4 反方程的求解	165	5.5.6 重载函数	245
第4章 MATLAB 图形功能	167	5.6 P 码文件和变量使用范围	245
4.1 二维基本图形	167	5.6.1 P 码文件	245
4.1.1 基本绘图函数	167	5.6.2 局部变量和全局变量	246
4.1.2 特殊函数	174	5.7 M 文件调试	248
4.2 三维基本图形	181	5.7.1 M 文件出错信息	248
4.2.1 基本绘图函数	181	5.7.2 M 文件调试方法	249
4.2.2 特殊函数	185	5.8 M 文件性能分析	256
4.3 图形处理技术	188	5.9 编程技巧	259
4.3.1 坐标轴的调整	188	第6章 Simulink 仿真	261
4.3.2 文字标示	191	6.1 Simulink 介绍	261
4.3.3 图例注解及添加颜色条	193	6.1.1 Simulink 概述	261
4.3.4 图形的保持	195	6.1.2 Simulink 窗口介绍	263
4.3.5 网格控制及坐标轴封闭	197	6.1.3 Simulink 运行原理	271
4.3.6 图形窗口的分割	198	6.2 Simulink 常用模块	273
4.4 图形窗口	200	6.3 Simulink 其他模块	286
4.4.1 图形窗口的创建与控制	200	6.4 Simulink 模型创建	288
4.4.2 图形窗口的菜单操作	207	6.4.1 模块操作	289
第5章 M 文件编程	214	6.4.2 基本步骤	291
5.1 编程概述	214	6.5 子系统及其封装	292
5.1.1 M 文件的创建	214	6.5.1 子系统的创建	292
5.1.2 M 文件的打开	215	6.5.2 子系统的条件执行	292
5.1.3 M 文件内容的显示	216	6.5.3 子系统的封装	294
5.1.4 M 文件的分类	218	6.6 运行仿真	299
5.2 与外部数据的交换	218	6.6.1 过零检测和代数环	299
5.2.1 数据文件保存	218	6.6.2 仿真参数的设置	299
5.2.2 数据文件调用	220	6.6.3 仿真的运行	303
5.3 流程控制	221	6.7 模型调试	305
5.3.1 顺序结构	221	第7章 图形用户界面	310
5.3.2 分支结构	222	7.1 界面设计	310
5.3.3 循环结构	224	7.1.1 图形用户界面 (GUI) 概述	310
5.3.4 其他流程控制结构	226	7.1.2 GUIDE 的控件	313
		7.1.3 GUIDE 开发环境	314

7.2 程序设计.....	323	7.2.6 GUI 与 Simulink 仿真的数据交互.....	331
7.2.1 对象的回调函数.....	323	7.2.7 中断执行.....	337
7.2.2 程序的一般结构.....	324	7.2.8 多界面实例.....	339
7.2.3 对象属性的访问.....	325	7.3 GUI 应用.....	343
7.2.4 对象间数据传递.....	326	7.3.1 GUI 设计的一般步骤.....	343
7.2.5 GUI 与 M 文件的数据交互.....	328	7.3.2 GUI 设计实例.....	343
第二篇 提高篇			
第 8 章 MATLAB 科学计算.....	349	9.3 Level-1 M 文件型.....	397
8.1 方程求解.....	349	9.3.1 概述.....	397
8.1.1 线性方程组.....	349	9.3.2 编写方法.....	399
8.1.2 非线性方程.....	359	9.3.3 实例.....	401
8.1.3 常微分方程.....	363	9.4 Level-2M 文件型.....	413
8.2 数据统计处理.....	367	9.4.1 概述.....	413
8.2.1 最大值和最小值.....	368	9.4.2 编写方法.....	416
8.2.2 求和和求积.....	370	9.4.3 实例.....	419
8.2.3 平均值和中值.....	370	9.5 C MEX 文件型.....	424
8.2.4 标准方差.....	370	9.5.1 概述.....	424
8.2.5 相关系数.....	371	9.5.2 编写方法.....	433
8.2.6 排序.....	372	9.5.3 实例.....	437
8.3 数据插值.....	373	9.6 使用 S-函数创建器编写 C MEX 文件型.....	443
8.3.1 一维插值.....	373	第 10 章 物理系统的建模和仿真.....	446
8.3.2 二维插值.....	374	10.1 物理元件库.....	446
8.3.3 三维插值.....	377	10.2 机械系统.....	448
8.4 数值积分.....	378	10.2.1 主要的机械元件.....	449
8.4.1 一元函数积分.....	378	10.2.2 建模的基本要点及步骤.....	450
8.4.2 矢量积分.....	381	10.2.3 常用的机械系统.....	452
8.4.3 二元函数积分.....	382	10.3 电气系统.....	454
8.4.4 三元函数积分.....	383	10.3.1 主要的电气元件.....	454
8.5 最优化问题求解.....	383	10.3.2 建模的基本步骤.....	455
8.5.1 无约束非线性极小化.....	383	10.3.3 常用的电气系统.....	457
8.5.2 有约束极小化.....	384	10.4 多域物理系统实例.....	459
8.5.3 二次规划和线性规划.....	384	第 11 章 MATLAB 外部接口.....	462
8.5.4 线性最小二乘.....	387	11.1 文本文件.....	462
8.5.5 非线性最小二乘.....	389	11.1.1 打开/关闭文件.....	462
8.5.6 多目标寻优方法.....	390	11.1.2 二进制形式访问.....	465
第 9 章 S-函数.....	393	11.1.3 普通形式访问.....	469
9.1 基本概念.....	393	11.1.4 文件内的位置控制.....	473
9.2 工作原理.....	395		

11.2	MATLAB 与 Word 混合使用 ..	476	11.4	编译器	485
11.2.1	Notebook 的安装和使用	476	11.4.1	编译器的安装和配置	485
11.2.2	Notebook 的实际应用	477	11.4.2	编译命令	487
11.3	MATLAB 与 Excel 混合使用 ..	479	11.4.3	项目开发工具	488
11.3.1	Spreadsheet Link 的安装	479	11.5	MATLAB 与 C 语言混合使用 ..	489
11.3.2	Spreadsheet Link 的启动和 退出	482	11.6	MATLAB 与外部设备和互联 网交互	490
11.3.3	Spreadsheet Link 的实际应用 ..	482			

第一篇 基础篇

第 1 章 MATLAB 概述

MATLAB (Matrix Laboratory) 由美国的 Cleve Moler 博士首创, 是由 MathWorks 公司于 1982 年推出的一套高性能的数值计算和可视化软件, 它集数值分析、矩阵运算、信号处理和图形显示于一体, 构成了一个方便的、接口友好的用户环境。MATLAB 可以进行矩阵运算、函数计算和数据处理, 可以实现算法、创建用户界面、连接其他编程语言的程序等, 主要应用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域。

新版本的 MATLAB 语言是基于最为流行的 C++ 语言基础上的, 因此语法特征与 C++ 语言极为相似, 而且更加简单, 更加符合科技人员对数学表达式的书写格式, 使之更利于非计算机专业的科技人员使用。而且这种语言可移植性好、可拓展性极强, 这也是 MATLAB 能够深入到科学研究及工程计算各个领域的重要原因。

MATLAB 对许多专门的领域都开发了功能强大的模块集和工具箱。目前, MATLAB 已经把工具箱延伸到了科学研究和工程应用的诸多领域, 诸如数据采集、数据库接口、概率统计、样条拟合、优化算法、偏微分方程求解、神经网络、小波分析、信号处理、图像处理、系统辨识、控制系统设计等。

1.1 MATLAB 简介与发展历史

MATLAB 作为一种高效的工程计算语言, 不仅提供了高性能的数值计算和可视化功能, 而且提供了大量的内置函数, 广泛应用于信号处理、科学运算、控制系统等领域的分析、仿真和设计工作。本书使用的 MATLAB 7.10 是 2010 年 3 月发布的, 下面介绍它的基本功能及特点和新功能及特点。

1.1.1 MATLAB 的基本功能及特点

MATLAB 的基本功能及特点主要有以下几个方面:

1. 数学计算

数学计算是 MATLAB 的基础, 包括数值计算和矩阵计算等, 具体的 MATLAB 计算内容主要包括以下几个方面:

- 线性代数、矩阵分析与运算。
- 线性方程与微分方程求解。
- 稀疏矩阵运算。
- 三角函数和其他初等函数计算。
- Bessel、Beta 和其他特殊函数计算。
- 数据处理和基本统计。
- 傅里叶变换及相关性、协方差的分析。

2. 开发工具

MATLAB 提供了用于算法开发的工具，主要有以下几种。

- **MATLAB Editor (MATLAB 编辑器)**: 提供了标注的编辑、调试 M 文件的基本环境。
- **M-Lint Code Checker (M-Lint 代码检查器)**: 分析 M 文件，并向开发人员提出改善代码性能和增强维护性的建议。
- **MATLAB Profiler (MATLAB 分析器)**: 计算 M 文件代码执行的时间。
- **Directory Reports (目录报告)**: 扫描当前目录下的 M 文件，报告文件的代码效率及文件的相关性。

3. 数据可视化

MATLAB 提供了丰富的数据可视化功能函数，主要有以下几种功能：

- 绘制二维、三维图形，如直线图、直方图、饼图和极坐标图等。
- 图形标注和处理功能，包括对象对齐、连接注释和数据点的箭头等。
- 支持动画和声音。
- 数据探测工具，可以在图形窗口中查询图形上某一点的坐标值。
- 具有多种光源设置、照相机和透视控制等。

4. 工具箱功能函数

用户可以直接使用 MATLAB 利用 M 文件开发的专业工具箱。工具箱具有开放性和扩展性，用户可以修改现有算法，并且可以开发新算法来扩充自己的工具箱。MATLAB 的一些工具箱如下：

- 信号处理。
- 控制系统。
- 滤波器设计。
- 图像处理。
- 科学计算。
- 金融财务分析。
- 生物遗传工程。

5. Simulink 仿真功能

Simulink 可以实现对各种动态模型建模、仿真和分析。Simulink 提供了丰富的功能模块和专业模块，可以通过鼠标布置模块建立系统框图模型，并且 Simulink 可以对任何能够用数学描述的系统建模，它的一些应用领域如下：

- 通信与卫星的仿真。
- 航空航天系统的仿真。
- 生物系统的仿真。
- 船舶系统的仿真。

- 汽车系统的仿真。
- 金融系统的仿真。

6. 图形用户接口界面开发环境

图形用户接口是用户和计算机程序之间的交互通道，用户可以通过输入设备，如鼠标、键盘、麦克风或控制板等，实现和计算机之间的通信，图形用户接口具有以下一些特点：

- 支持多种界面元素，如按钮、复选框、文本编辑框和滚动条等。
- 支持下拉式菜单及弹出式菜单。
- 用户可以直接访问 ActiveX 控件。

1.1.2 MATLAB R2010a 的新功能及特点

MATLAB 7.10 属于 MATLAB R2010a，较 MATLAB R2009a 版本有较大的变化。总体来说，Release 2010a 包括 MATLAB 和 Simulink 的若干新功能、一款新产品以及对其他 85 款产品的更新和缺陷修复。已经购买软件维护服务的用户可以下载产品更新。从 R2010a 起，PolySpace 产品需要激活，用户可以访问许可中心下载产品、激活软件并管理许可证，设置用户信息。

下面首先基于 MATLAB R2010a 和 MATLAB R2009 的产品说明，分别列出新增产品、更名产品和未修改产品（不考虑是否修订错误），如表 1-1~表 1-3 所示。

表 1-1 新增产品

产品名称	产品功能描述
Simulink PLC Coder	用于生成 PLC 和 PAC IEC 61131 结构化文本的新产品

表 1-2 更名产品

原产品名称	现产品名称
Genetic Algorithm and Direct Search Toolbox	Global Optimization Toolbox

表 1-3 未修改产品

产品名称	产品名称
Aerospace Toolbox	MATLAB Report Generator
Bioinformatics Toolbox	Model Predictive Control Toolbox
Communications Toolbox	Model-Based Calibration Toolbox
Control System Toolbox	Mapping Toolbox
Curve Fitting Toolbox	Neural Network Toolbox
Data Acquisition Toolbox	OPC Toolbox
Database Toolbox	Optimization Toolbox
Datafeed Toolbox	Partial Differential Equation Toolbox
Econometrics Toolbox	Parallel Computing Toolbox
Filter Design HDL Coder	RF Toolbox
Filter Design Toolbox	Robust Control Toolbox
Financial Derivatives Toolbox	Signal Processing Toolbox
Financial Toolbox	SimBiology
Fixed-Income Toolbox	Spline Toolbox
Fixed-Point Toolbox	Spreadsheet Link EX
Fuzzy Logic Toolbox	Statistics Toolbox
Image Acquisition Toolbox	Symbolic Math Toolbox
Image Processing Toolbox	System Identification Toolbox
Instrument Control Toolbox	SystemTest
MATLAB Builder JA	Vehicle Network Toolbox
MATLAB Builder NE	Wavelet Toolbox
MATLAB Compiler	MATLAB Builder EX
MATLAB Distributed Computing Server	

这里需要说明的是，之所以列出未修改产品是因为大多数产品经过了修改和更新。

MATLAB 的新功能包括：

- 增加了更多多线程数学函数，增强了文件共享、路径管理功能，改进了 MATLAB 桌面。
- 新增用于在 MATLAB 中进行流处理的系统对象，并在 Video and Image Processing Blockset 和 Signal Processing Blockset 中提供超过 140 种支持算法。
- 针对 50 多个函数提供多核支持并增强性能，并对图像处理工具箱中的大型图像提供更多支持。
- 在全局优化工具箱和优化工具箱中提供新的非线性求解器。
- 能够从 Symbolic Math Toolbox 中生成 Simscape 语言方程。
- 在 SimBiology 中提供随机近似最大期望（SAEM）算法和药动学给药方案支持。

Simulink 的新功能包括：

- 在 Simulink 中提供可调参数结构、触发模型块以及用于大型建模的函数调用分支。
- 在嵌入式 IDE 链接和目标支持包中提供针对 Eclipse、嵌入式 Linux 及 ARM 处理器的代码生成支持。
- 在 IEC 认证工具包中提供对 Real-Time Workshop Embedded Coder 和 PolySpace 产品的 ISO 26262 认证。
- 在 DO 鉴定工具包中提供扩展至模型的 DO-178B 鉴定支持。
- 增加了 Simulink PLC Coder，用于生成 PLC 和 PAC IEC 61131 结构化文本的新产品。

1.1.3 MATLAB 的发展历史

MATLAB 经历了二十多年的研究与发展，现在已经从最初的“矩阵实验室”演变成为具有广泛前景的全新计算机高级编程语言。MATLAB 的发展经历了以下几个主要阶段：

（1）1992 年，MathWorks 公司推出了具有划时代意义的 MATLAB 4.0。

（2）1993 年，MathWorks 公司推出了可以配合 Microsoft Windows 使用的 MATLAB 4.x。此版本在继承和发展原有的数值计算和图形可视化能力的同时，推出了 Simulink，这是一个交互式操作的动态系统建模、仿真和分析的集成环境；开发了与外部直接进行数据交换的组件；推出了符号计算工具包；构造了 Notebook。

（3）1997 年，MathWorks 公司推出了 Windows 95 下的 MATLAB 5.0 和 Simulink 2.0。在原有版本的基础上，真正实现了 32 位运算，数值计算速度快，图形更加丰富，编程也更加简洁。

（4）1999 年初，MathWorks 公司推出了 MATLAB 5.x，与之前的 MATLAB 相比，它具有更丰富的数据类型、更友好的面向对象设计、更强大的数学计算能力和更多的应用开发工具。

（5）2000 年，MathWorks 公司推出了 MATLAB 6.0，使 MATLAB 拥有了强大的、成系列的交互式界面。

（6）2004 年，MathWorks 公司推出了 MATLAB 7.0 和 Simulink 6.0，MATLAB 在编程环境、数据可视化、数学计算和文件 I/O 方面进行了升级；Simulink 对大规模的系统开发进行了性能优化。

（7）2006 年 3 月，MathWorks 公司推出了 R2006a（MATLAB 7.2、Simulink 6.4），主要更新了 10 个产品模块，增加了 350 个新特性，并增加了对 64 位 Windows 的支持，还推出了 .NET 工具箱。

（8）2006 年 9 月，MathWorks 公司推出了 R2006b（MATLAB 7.3、Simulink 6.5），包含 6 个自 R2006a 开发以来的新产品，而且还对 7 个产品进行了升级，并对近 80 个产品进行了小的升级和缺陷的修复。

(9) 2007 年 3 月, MathWorks 公司推出了 R2007a (MATLAB 7.4, Simulink 6.6), 该产品不仅更新了 MATLAB R2006b 中 MATLAB 和 Simulink 的功能, 还更新了 82 项其他模块, 修复了相应的缺陷, 同时增加了对基于 Intel 的 Mac、Windows Vista 及 64 位 Sun Solaris SPARC 平台的支持。

(10) 2007 年 9 月, MathWorks 公司推出了 R2007b (MATLAB 7.5, Simulink 7), 包括新产品 Simulink Design Verifier、Link for Analog Devices VisualDSP 以及 82 个产品模块的更新升级及缺陷修复。

(11) 2008 年 3 月, MathWorks 公司推出了 R2008a (MATLAB 7.6, Simulink 7.1), 该产品包括了 MATLAB 和 Simulink 的新特性、两个新产品, 以及对 82 种其他产品的更新和缺陷修复。

(12) 2008 年 9 月, MathWorks 公司推出了 R2008b (MATLAB 7.7, Simulink 7.2), 包含 MATLAB 和 Simulink 的新特性、两个新产品, 以及 91 个其他产品的更新和缺陷修复, 包括 PolySpace 代码验证产品。

(13) 2009 年 3 月, MathWorks 公司推出了 R2009a (MATLAB 7.8, Simulink 7.3), 包括 MATLAB 和 Simulink 的若干新功能、两款新产品, 并对其他 91 款产品进行了更新和缺陷修复。

(14) 2009 年 9 月, MathWorks 公司推出了 R2009b (MATLAB 7.9, Simulink 7.4), 包括 MATLAB 和 Simulink 的若干新功能, 以及对其他 83 款产品的更新和缺陷修复, 增加了对 64 位 Mac 平台的支持。

(15) 2010 年 3 月, MathWorks 公司推出了 R2010a (MATLAB 7.10, Simulink 7.5), 该产品进一步完善了 MATLAB 和 Simulink。

1.2 MATLAB 的安装、退出与卸载

1.2.1 MATLAB 安装

1. 对硬件和软件的要求

MATLAB 的安装平台:

(1) 安装到 Windows 下:

- Windows XP (Service Pack 2 or 3)
- Windows 2003 (Service Pack 2 or R2)
- Windows Vista (Service Pack 1)
- Windows Server 2008

(2) 安装到 Linux 下:

- Debian 4.0 及以上
- Red Hat Enterprise Linux v.4 及以上
- OpenSuSE 9.3 及以上
- Ubuntu 8 及以上

(3) 安装到 Solaris 下:

- Solaris 10*

(4) 安装到 Mac 下:

- Mac OS X 10.4 (10.4.8 及以上)

- Mac OS X 10.5（10.5.2 及以上）

MATLAB 在单机环境或是网络环境都可发挥其卓越的性能，MATLAB 编程规则和 C 语言类似，使用 MATLAB 语言编写的程序可以直接送入其他机型使用。MATLAB 对 PC 的要求如表 1-4 所示。

表 1-4 MATLAB 对 PC 的要求

操作平台	Windows XP（Service Pack 2 or 3）、Windows 2003（Service Pack 2 or R2）、Windows Vista（Service Pack 1）、Windows Server 2008
处理器	Intel Pentium（Pentium 4 及以上）、Intel Celeron**、Intel Xeon、Intel Core、AMD Athlon 64**、AMD Opteron、AMD Sempron
存储空间	625 MB
内存	512 MB
显卡	16-bit、24-bit 或 32-bit 兼容 OpenGL 的图形适配卡
软件	为了运行 MATLAB Notebook，需要安装 Microsoft Word 2002、2003 或 2007，为了运行 MATLAB Builder for Excel、Excel Link，需要安装 Microsoft Excel 2002、2003 或 2007

2. 安装步骤

第一步：准备安装。

- 退出正在运行的其他版本的 MATLAB。
- 不要在安装过程中进行病毒扫描，否则会降低安装速度。

第二步：开始安装。

将 MATLAB 的安装盘插入光驱中，自动进入如图 1-1 所示的安装界面。此时有两种选择，即通过网络自动安装和不通过网络手动安装。

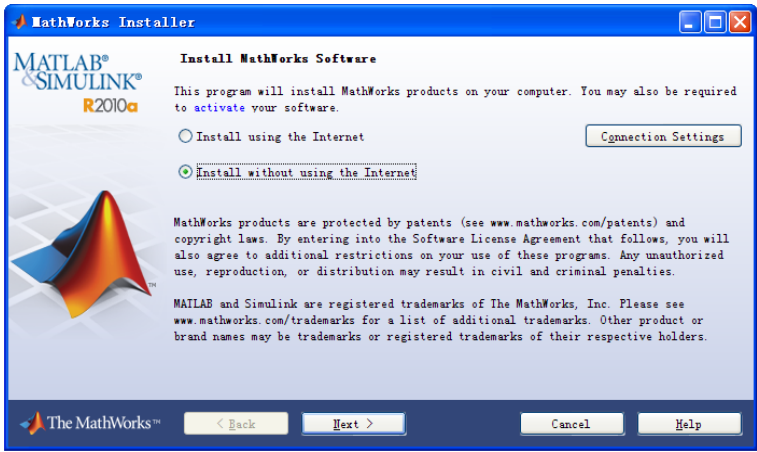


图 1-1 安装界面（一）

第三步：单击“Next”，出现如图 1-2 所示的软件许可证协议。

第四步：选择“**Yes**”，单击“Next”继续安装，出现如图 1-3 所示的“File Installation Key”（文件安装密码）对话框。

如果用户没有安装密码，选择“I do not have the File Installation Key. Help me whit the next steps”，单击“Next”出现如图 1-4 所示的对话框。

单击网址会链接到 Math Works 公司的主页，但是需要有该公司提供的访问账号和口令。

如果用户有安装密码，选择“I have the File Installation Key for my license”，输入安装密码并单击“Next”继续安装。

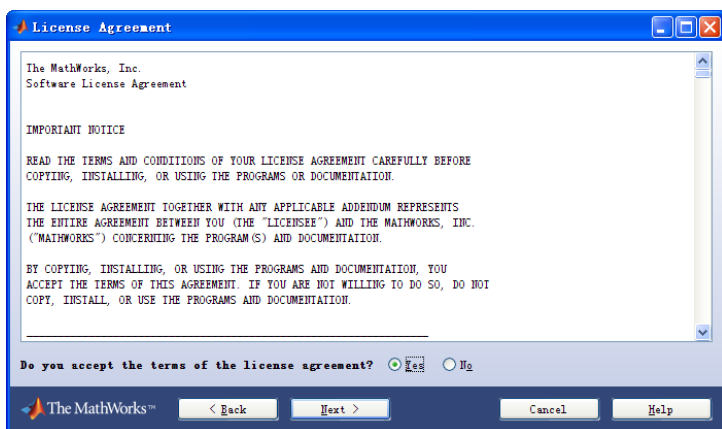


图 1-2 安装界面（二）



图 1-3 安装界面（三）

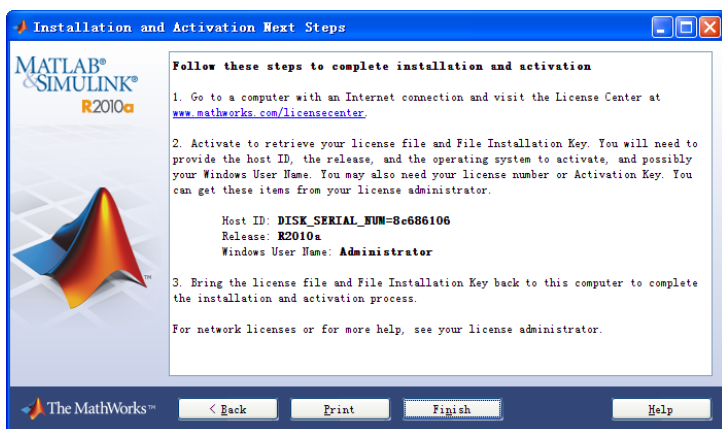


图 1-4 安装界面（四）

第五步:如图 1-5 所示选择安装类型。安装类型有两种,一种是典型安装类型“Typical”,另一种是自定义安装类型“Custom”。自定义安装类型允许用户选择所要安装的产品,并设定哪些需要访问安装选项。但是为了保证能够使用所有产品的功能,一般选择典型安装类型。

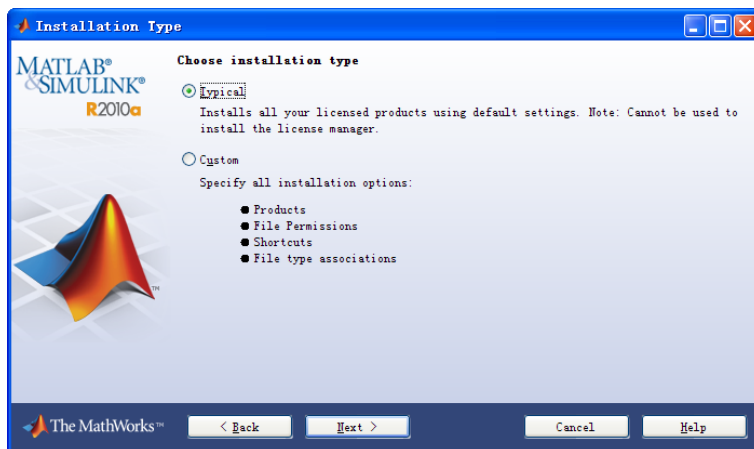


图 1-5 安装界面（五）

第六步：单击“Next”，出现如图 1-6 所示的定义安装目录的对话框。

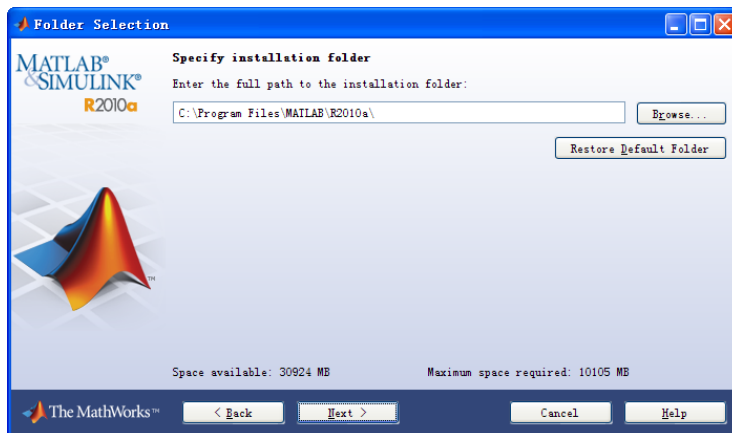


图 1-6 安装界面（六）

第七步：单击“Next”，出现如图 1-7 所示的“Folder Selection”（选择安装目录）对话框。

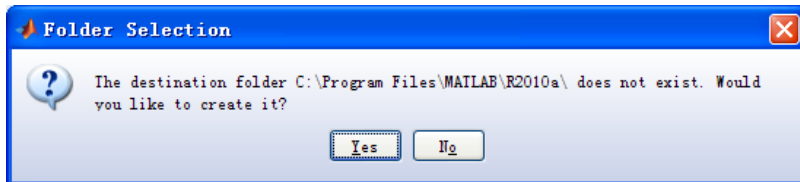


图 1-7 安装界面（七）

第八步：单击“Yes”，出现如图 1-8 所示的“Confirmation”（确认安装设置）对话框，用于确认前面设置的安装目录和需要安装的产品。

第九步：单击“Install”，进行 MATLAB 的安装并显示如图 1-9 所示的安装进度条。

第十步：完成安装后，需要激活所安装的程序，程序被激活后出现如图 1-10 所示的安装完毕的对话框。如果用户需要在退出安装后直接启动 MATLAB，选择“Start MATLAB”（运行 MATLAB）；如果用户不需要在退出安装后启动 MATLAB，则不必选择“Start MATLAB”。

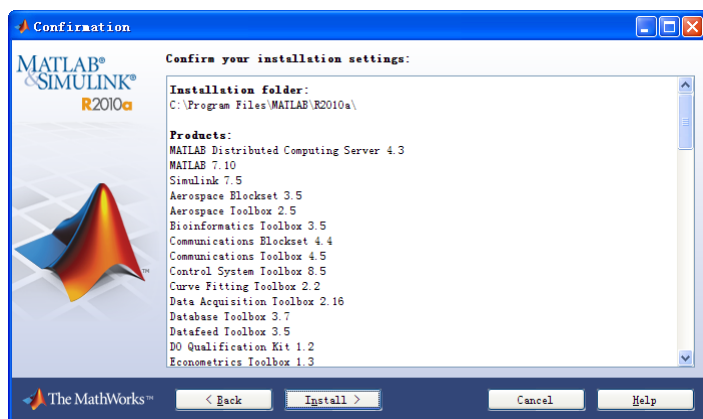


图 1-8 安装界面（八）

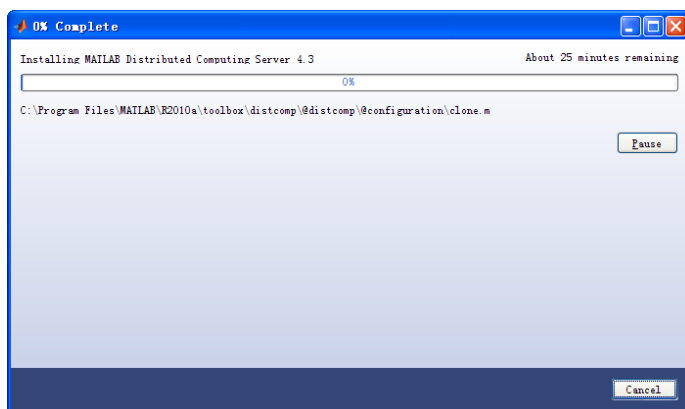


图 1-9 安装界面（九）

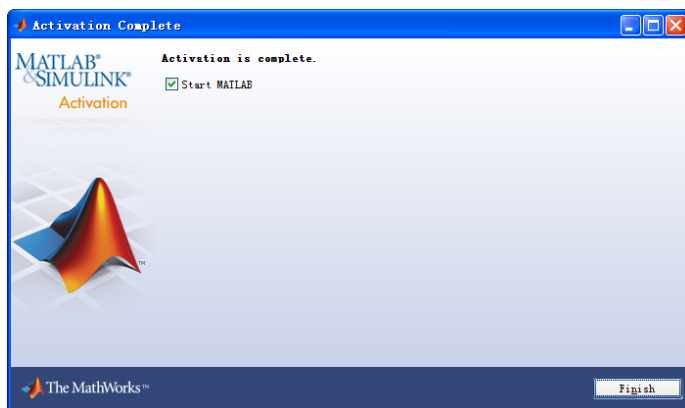


图 1-10 安装界面（十）

3. 启动步骤

本书以 Windows XP 系统为例说明 MATLAB 的启动、退出和卸载。启动 MATLAB，可以通过以下两种方式：

(1) 双击桌面上的图标。

(2) 选择“开始”→“所有程序”→“MATLAB”→“R2010a”→“MATLAB R2010a”如图 1-11 所示。

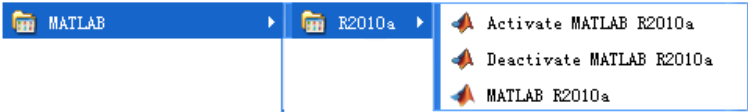


图 1-11 通过“开始”菜单启动 MATLAB

1.2.2 MATLAB 退出

用户启动了 MATLAB 以后，如需要退出 MATLAB，有以下几种方法：

- 选择如图 1-12 所示的“File”（文档）菜单中的“Exit MATLAB”（退出 MATLAB）选项。

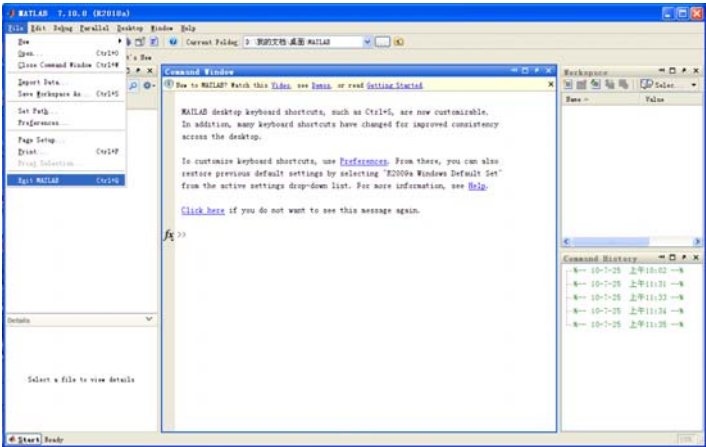


图 1-12 退出 MATLAB

- 在命令窗口中输入“quit”。
- 在命令窗口中输入“exit”。
- 使用快捷键“Ctrl+Q”。
- 单击 MATLAB 主窗口右上角的图标。

1.2.3 MATLAB 卸载

选择“开始”→“控制面板”→“删除/添加程序”→“MATLAB R2010a”，出现如图 1-13 所示的对话框，单击“Uninstall”开始卸载 MATLAB。

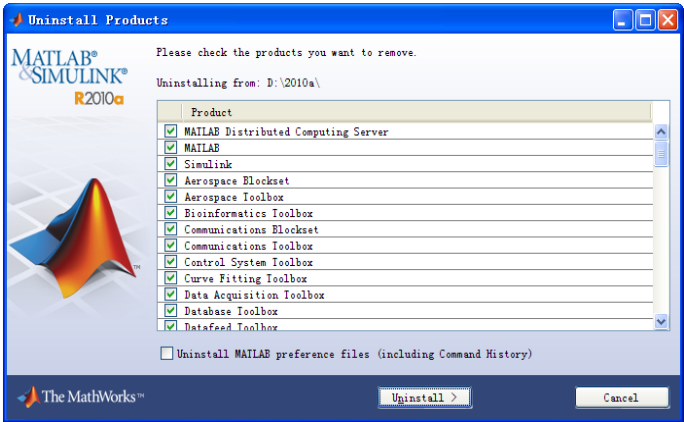


图 1-13 卸载 MATLAB

1.3 MATLAB 的目录结构

成功安装 MATLAB 以后, 用户可以在安装目录下找到如表 1-5 所示的文件夹目录。

表 1-5 MATLAB 的目录结构

文件夹	描述
\bin\win32	MATLAB 系统可执行的相关文件
\extern	创建 MATLAB 的外部接口程序
\help	帮助系统
\ja	MATLAB 的国际化文件
\java	MATLAB 的 Java 支持程序
\jhelp	MATLAB 的 Java 帮助文件
\licenses	MATLAB 的许可证文件
\notebook	用来实现 MATLAB 数学工作环境与 Word 字处理环境信息交互的软件, 是一个兼备数学计算、图形显示和文字处理能力的集成环境
\rtw	Real-Time Workshop 软件包
\simulink	Simulink 软件包, 用于动态系统的建模、仿真和分析
\stateflow	Stateflow 软件包, 用于图形化开发和设计
\sys	MATLAB 所需要的工具和操作系统库
\toolbox	MATLAB 的各种工具箱
\uninstall	MATLAB 的卸载程序
\work	默认当前目录

1.4 MATLAB 的工作环境

MATLAB 的工作环境界面由六部分组成, 如图 1-14 所示, 即菜单栏、工具栏、当前文件夹窗口、工作空间窗口、历史命令窗口和命令窗口。

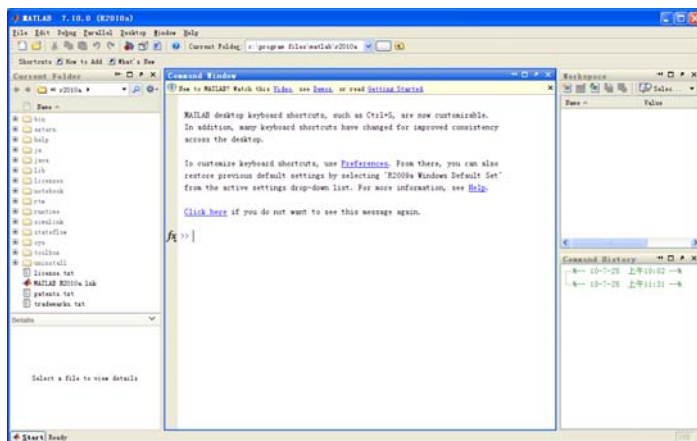


图 1-14 MATLAB 工作环境

1. MATLAB 菜单栏

MATLAB 的菜单栏如图 1-15 所示, 类似于 Windows 程序, 位于工作环境的最上方。

当执行不同的窗口操作时, 菜单的内容就会发生不同的变化。这里只介绍默认情况下的菜单。



图 1-15 MATLAB 菜单栏

(1) File 菜单

File 菜单如图 1-16 所示，包括以下几个命令：

- **New:** 可以打开 M 文件编辑器 (M-File)、创建 MATLAB 图形窗口 (Figure)、工作空间窗口 (Workspace), 创建新的变量 (Variable), 创建 Simulink 模型 (Model), 打开 MATLAB 的 GUI 编辑器 (GUI), 新建开发项目 (Development Project)。
- **Open:** 打开已经存在的文件。
- **Close Current Window:** 关闭当前文件夹窗口。
- **Import Data:** 向工作空间导入数据。
- **Save Workspace As:** 将工作空间的变量保存到文件中。
- **Set Path:** 打开如图 1-17 所示的路径浏览器。

当 MATLAB 对函数或文件等进行搜索时，是在其搜索路径下进行搜索的，如果用户调用的函数在搜索路径之外，MATLAB 将认为此函数不存在。一般情况下，系统的函数（包括工具箱函数）都在系统默认搜索路径之中，但是用户自己书写的函数有可能并没有保存在搜索路径下，对于这个问题，只需把程序所在的目录扩展成 MATLAB 的搜索路径即可。通过图 1-17 所示的路径浏览器，用户可以删除、添加和设置 MATLAB 的搜索路径。

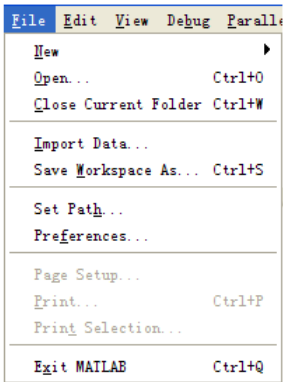


图 1-16 File 菜单

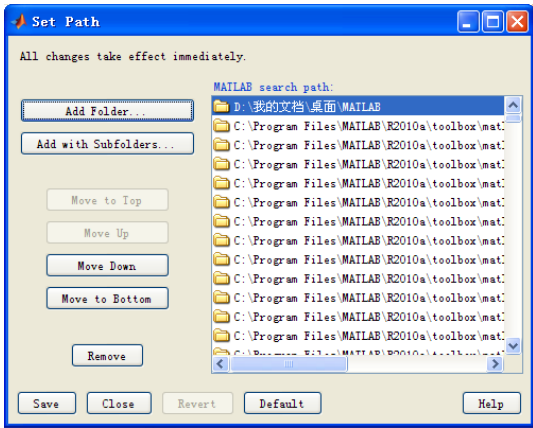


图 1-17 路径浏览器

- **Preferences:** 打开环境设置对话框。
- **Page Setup:** 打开打印设置对话框。
- **Print:** 开始打印。
- **Exit MATLAB:** 退出 MATLAB。

(2) Edit 菜单

Edit 菜单用于实现复制、粘贴等基本操作，类似于一般的 Windows 程序。

(3) Debug 菜单

Debug 菜单用于调试程序。

(4) Parallel 菜单

Parallel 菜单用于进行并行处理。

(5) Desktop 菜单

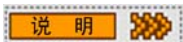
Desktop 菜单用于设置在工作环境中需要打开的窗口。

(6) Window 菜单

Window 菜单用于列出所有打开的窗口。

(7) Help 菜单

Help 菜单用于打开不同的帮助系统。



用户打开“Current Directory”（当前工作目录）窗口，会增加 View 菜单，用于显示当前目录下的文件。

用户“Workspace”（工作空间）窗口，会增加 View 菜单和 Graphics 菜单，View 菜单用于设置工作空间中的显示变量，Graphics 菜单用于打开绘图工具。

2. MATLAB 工具栏

MATLAB 工具栏如图 1-18 所示，为用户提供了更便捷的操作。

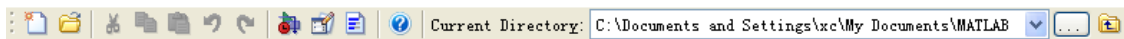












图 1-18 MATLAB 工具栏






- (1)  图标：新建 M 文件。
- (2)  图标：打开已经存在的文件。
- (3)  图标：剪切已选的对象。
- (4)  图标：复制已选的对象。
- (5)  图标：粘贴已选的对象。
- (6)  图标：打开 Simulink 主窗口。
- (7)  图标：打开图形用户界面设计窗口。
- (8)  图标：打开帮助系统。
- (9)  图标：用于设置当前目录。
- (10)  图标：可以直接打开 MATLAB 工具。

3. MATLAB 命令窗口

当 MATLAB 启动完成后，会出现 MATLAB 的工作环境，命令窗口位于工作环境的中间。MATLAB 的命令窗口是 MATLAB 进行工作的窗口，也是实现 MATLAB 各种功能的窗口。MATLAB 命令窗口如图 1-19 所示，其中“>>”是运算提示符，表示 MATLAB 处于准备编辑的状态，用户可以直接在 MATLAB 的运算提示符后输入语句，按“Enter”键后会实现相应语句的功能。

(1) 快捷操作

在 MATLAB 命令窗口的右上方，有四个图标：

-  图标：实现命令窗口最小化功能。
-  图标：实现命令窗口最大化功能。
-  图标：可以得到一个如图 1-20 所示的脱离了主窗口的独立窗口，独立的命令窗口主要包括文本的编辑区域和菜单栏。
-  图标：此图标位于图 1-20 中，用于将独立窗口嵌入主窗口。
-  图标：用于关闭命令窗口。

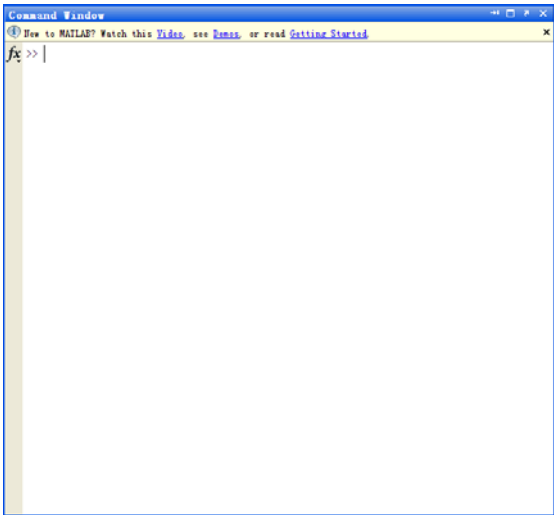


图 1-19 命令窗口

(2) 鼠标右键的快捷菜单

在命令窗口的空白处单击鼠标右键，会出现如图 1-21 所示的快捷菜单，通过菜单中的不同选项可以实现不同的操作。

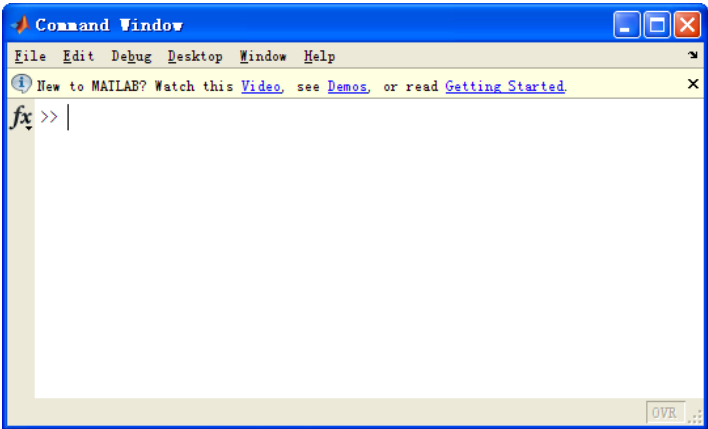


图 1-20 独立的命令窗口

Evaluate Selection	F9
Open Selection	Ctrl+D
Help on Selection	F1
Function Browser	Shift+F1
Show Function Browser Button	
Function Hints	Ctrl+F1
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Clear Command Window	

图 1-21 命令窗口的快捷菜单

- **Evaluate Selection:** 计算所选文本对应表达式的值。
- **Open Selection:** 打开所选文本对应的 MATLAB 文件。
- **Help on Selection:** 调用所选文本对应函数的帮助信息。
- **Cut:** 剪切命令。
- **Copy:** 复制命令。
- **Paste:** 粘贴命令。
- **Clear Command Window:** 清除命令窗口中的内容。

(3) 命令窗口的菜单栏

命令窗口的菜单栏和工作环境的菜单栏基本一致，在此不再介绍。

(4) 命令窗口的显示方式

命令窗口的显示方式一般为默认方式，用户也可以通过设置显示方式对它进行更改。

- 默认显示方式：在默认的情况下关键字是蓝色字体，输入的命令、表达式和计算结果是黑色字体，字符串是褐红色字体。
- 设置显示方式：用户可以根据需要，更改命令窗口中字符的字体、大小、颜色和数值计算结果的格式。

设置显示方式的具体方法如下：

选择命令窗口或工作环境中的“File”→“Preference”，出现如图 1-22 所示的环境设置对话框，通过相应的参数进行命令窗口显示方式的设置。

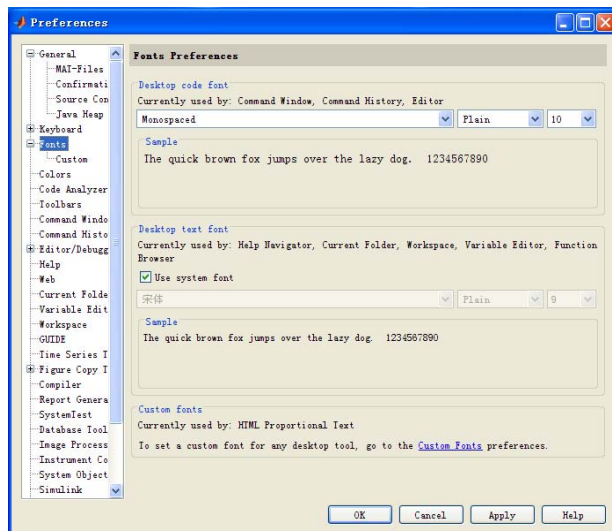


图 1-22 环境设置对话框

表 1-6 提供了数据格式及其含义，表 1-7 提供了数据显示格式及其含义。

表 1-6 数据格式及其含义

数据格式	含义
short	短格式，保证小数点后 4 位有效数字，最多不多于 7 位，大于 1000 的实数，使用 5 位有效数字的科学计数形式表示
long	长格式，15 位数组表示
short e	短格式 e 方式，5 位科学计数表示
long e	长格式 e 方式，15 位科学计数表示
short g	短格式 g 方式，从 format short 和 format short e 中选择最佳方式
long g	长格式 g 方式，从 format long 和 format long e 中选择最佳方式
short eng	短工程格式，小数点后 4 位且幂次采用 3 位的科学计数方法
long eng	长工程格式，小数点后 16 位且幂次采用 3 位的科学计数方法
hex	十六进制数表示
bank	（金融）元，角，分表示
+	+格式

表 1-7 数据显示格式及其含义

数据显示格式	含义
compact	紧凑格式，变量间没有空行
loose	稀松格式，变量间有空行

下面通过两个例子，具体说明通过环境设置对话框，如何更改命令窗口中字符的字体、大小和数值计算结果的格式。

(1) 打开如图 1-22 所示的对话框, 选择“Fonts”, 依次将命令窗口中的字体设置为“Monospaced”, 样式为“plain”且字号为“10”。在命令窗口中输入如下语句:

命令窗口中的输出结果如图 1-23 所示。



通过对图 1-23 和图 1-24 进行比较, 可以看出字体和大小都有很大变化。



(1) 打开“Preferences”对话框, 选择“Command Window”(命令窗口), 并将数据显示方式设置为“long e”和“loose”, 如图 1-25 所示, 在命令窗口中输入如下语句:

命令窗口中的输出结果如下所示:

5.555555555555555e+000

y=5.555555555555555555555555555555555555

$$\mathbf{y} \equiv$$

5.5555555555555556

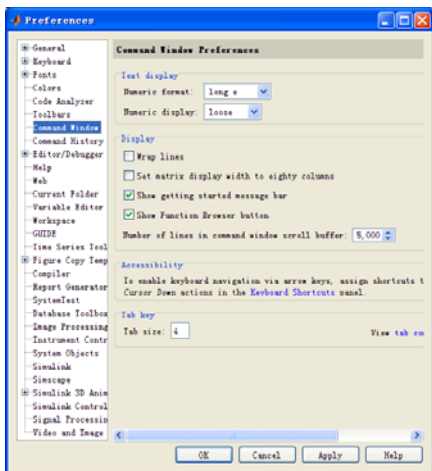


图 1-25 环境设置对话框

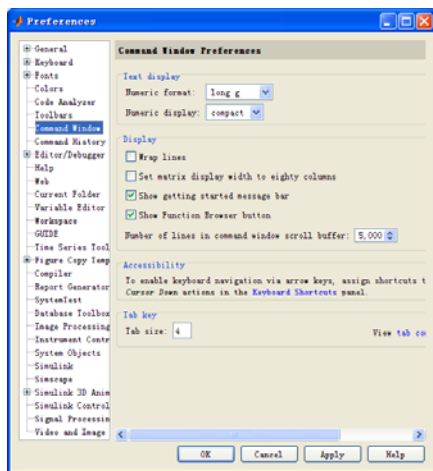


图 1-26 环境设置对话框






当前文件夹窗口用于显示当前路径下的文件，如图 1-27 所示。在当前文件夹窗口中可以如下操作：

- 显示或改变当前文件夹。
- 显示当前目录下的文件。
- 进行路径搜索。

下面介绍通过快捷操作和鼠标右键的快捷菜单两种方式实现指定功能。

(1) 快捷操作

当前文件夹工作窗口上方的、、、、图标和命令窗口中的功能相同。

-  图标：进入所显示文件夹的前一个文件夹。
-  图标：进入所显示文件夹的后一个文件夹。
-  图标：在当前文件夹中查找文件。
-  图标：显示所有文件夹。
-  图标：显示某一文件夹下的子文件夹，如图 1-28 所示。

（2）鼠标右键的快捷菜单

在当前文件夹窗口的空白处单击鼠标右键，会出现如图 1-29 所示的快捷菜单，通过菜单中的不同选项可以实现不同的操作，下面对一些选项做简单介绍。

- **Create Zip File:** 新建压缩文件。
- **New Folder:** 新建文件夹。
- **New File:** 建立新的 M 文件 (M-File)、Simulink 模型 (Model)。
- **Compare Against:** 与另一文件夹下的文件进行比较。

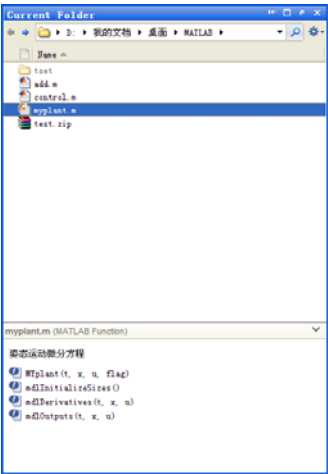


图 1-27 当前文件夹窗口

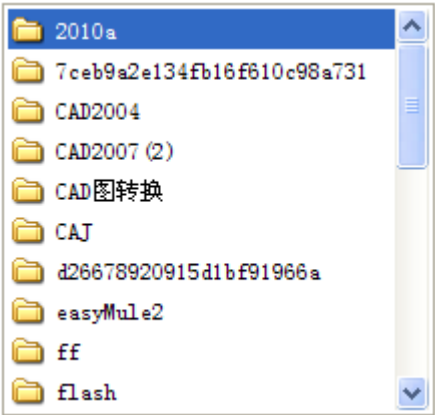


图 1-28 当前文件夹下的子文件夹

- Source Control: 设置数据源控制。
- Paste: 粘贴所选文件。
- Add to Path: 为所选文件添加目录。
- Indicate files not on path: 显示文件不在路径下。
- Locate on Disk: 在硬盘上定位文件。
- Find Files: 进行文件查找。
- Back: 回到当前目录。
- Up One Level: 上一级工作目录。
- Reports: 当前文件夹的文件报告。
- Refresh: 更新路径浏览器。
- Collapse All: 全部折叠。

选中当前文件夹窗口中某一文件夹，单击鼠标右键，会出现如图 1-30 所示的快捷菜单，通过菜单中的不同选项可以实现不同的操作，下面对一些选项做简单介绍。



图 1-29 当前文件夹窗口的快捷菜单

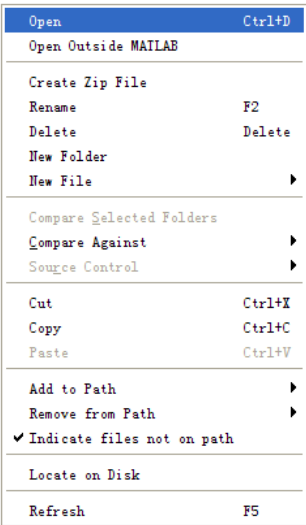


图 1-30 指定文件夹的快捷菜单

- Open: 打开所选择的文件。
- Open Outside MATLAB: 打开外部的 MATLAB 文件。
- Create Zip File: 新建压缩文件。
- Rename: 对所选文件夹进行重命名。
- Delete: 删除所选文件。
- New Folder: 新建文件夹。
- New File: 建立新的 M 文件 (M-File)、Simulink 模型 (Model)。
- Compare Against: 与另一文件夹下的文件进行比较。
- Cut: 剪切所选文件。
- Copy: 复制所选文件。
- Paste: 粘贴所选文件。
- Add to Path: 为所选文件添加目录。
- Remove from Path: 将所选文件从当前路径下移除。
- Indicate files not on path: 显示文件不在路径下。
- Locate on Disk: 在硬盘上定位文件。
- Refresh: 更新路径浏览器。







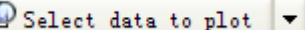
5. MATLAB 工作空间窗口

工作空间窗口如图 1-31 所示, 用来显示目前保存在内存中的 MATLAB 的变量名、数据结构、最小值以及最大值。

下面介绍通过快捷操作和鼠标右键的快捷菜单两种方式实现指定功能。

(1) 快捷操作

工作空间窗口上方的、、、、图标和命令窗口中的功能相同。

- 图标: 向工作空间添加新变量。
- 图标: 打开数组编辑器。
- 图标: 向工作空间载入数据
- 图标: 保存工作空间的变量。
- 图标: 删除工作空间的变量。
-  : 为选定的数据画图。

(2) 鼠标右键的快捷菜单

在工作空间窗口的空白处单击鼠标右键, 会出现如图 1-32 所示的快捷菜单, 通过菜单中的不同选项可以实现不同的操作。如选择工作空间窗口快捷菜单中的“Open Selection”, 会打开所选数据变量的数组编辑器, 如图 1-33 所示, 用户可以直接在数组编辑器内, 修改数据的结构数和属性等。

6. MATLAB 历史命令窗口

历史命令窗口如图 1-34 所示, 主要记录所有执行过的命令, 并标明使用的时间, 用户可以通过双击某一个历史命令来重新执行该命令。

下面介绍通过快捷操作和鼠标右键的快捷菜单两种方式实现指定功能。

(1) 快捷操作

历史命令窗口上方的、、、、图标和命令窗口中的功能相同。

(2) 鼠标右键的快捷菜单

在历史命令窗口的空白处单击鼠标右键, 会出现如图 1-35 所示的快捷菜单, 通过菜单中的

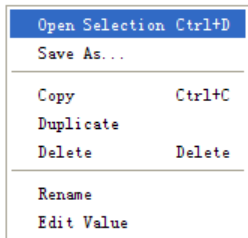


图 1-32 工作空间窗口的快捷菜单

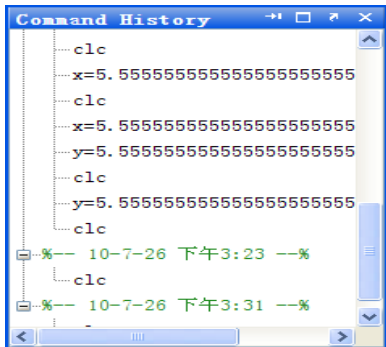


图 1-34 历史命令窗口

- **Cut:** 剪切命令。
- **Copy:** 复制命令。
- **Evaluate Selection:** 计算所选文本对应表达式的值。
- **Create M-File:** 将所选的历史命令写入到一个新的 **M** 文件中，并打开此文件。
- **Create Shortcut:** 进入如图 1-36 所示的快捷键设置对话框。
- **Profile Code:** 生成 **Profile** 代码命令。
- **Delete Selection:** 删除所选的历史命令。
- **Delete to Selection:** 删除所选对象之前的所有历史命令。
- **Clear Command History:** 删除历史命令。

由于图 1-37 中的工作空间窗口、当前文件夹窗口和命令窗口独立显示，所以工作环境看起来比较混乱。

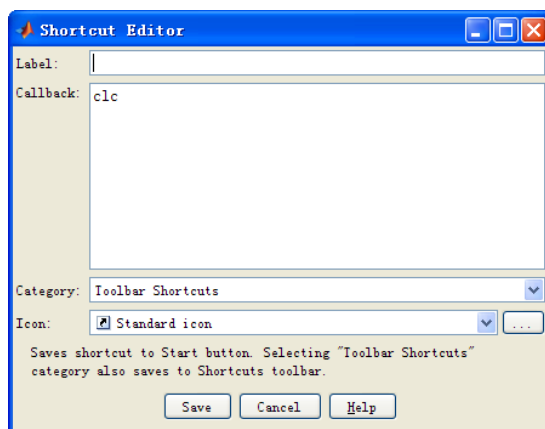


图 1-36 快捷键设置对话框

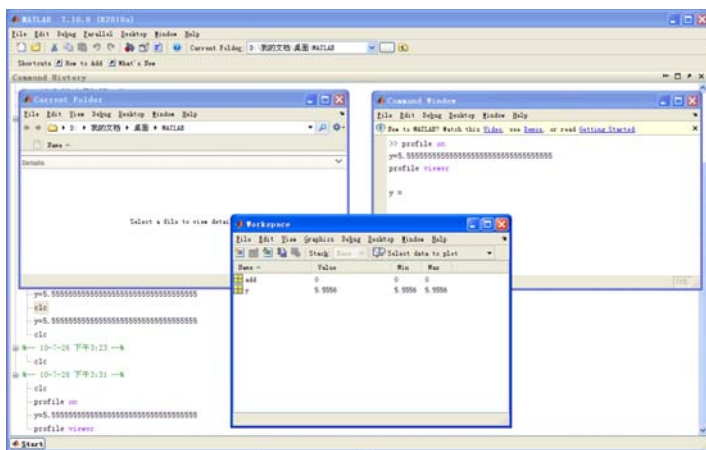


图 1-37 混乱的工作环境

可以通过上述操作，恢复到如图 1-38 所示的工作环境的初始状态。

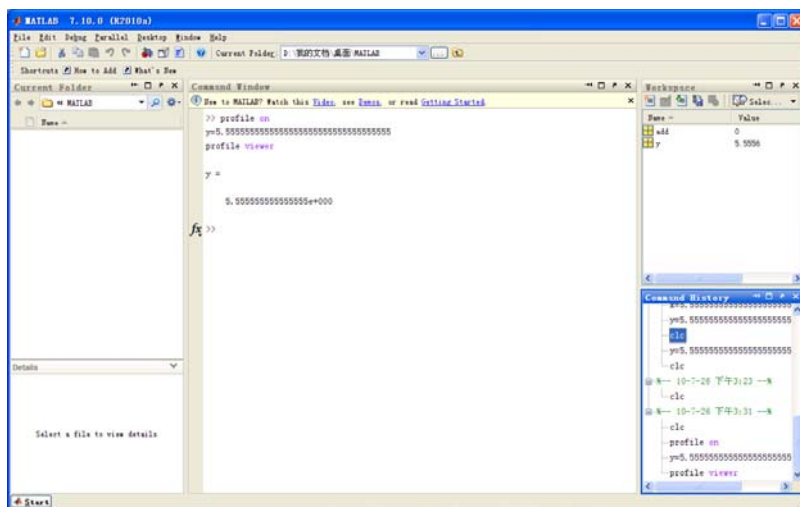


图 1-38 工作环境的初始状态

1.5 MATLAB 的通用命令简介

通用命令可以用来管理目录、命令、函数、变量、工作空间、文件和窗口，在 MATLAB 中会经常使用。为了能熟练运用 MATLAB，用户需要熟悉和掌握这些命令。

1. 常用命令

表 1-8 列出了 MATLAB 中常用的命令及其功能。

表 1-8 常用命令及其功能

常用命令	功能	常用命令	功能
cd	显示或改变当前工作目录	load	加载指定文件的变量
dir	显示当前目录或指定目录下的文件	diary	日志文件命令
clc	清除命令窗中的所有显示内容	!	调用 DOS 命令
home	将光标移至命令窗口的左上角	exit	退出 MATLAB 7.10
clf	清除图形窗口	quit	退出 MATLAB 7.10
type	显示文件内容	pack	收集内存碎片
clear	清除内存变量	hold	图形保持开关
echo	工作窗信息显示开关	path	显示搜索目录
disp	显示变量或文字内容	save	保存内存变量到指定文件

【例 1-3】利用 type 命令显示 color 函数的内容。

在命令窗口中输入如下语句：

```
type color
```

命令窗口中的输出结果如下所示：

```
function [group] = color(J,p)
%COLOR Column partition for sparse finite differences.
%
% GROUP = COLOR(J,P) returns a partition of the
% column corresponding to a coloring of the column-
% intersection graph. GROUP(I) = J means column I is
% colored J.
%
% All columns belonging to a color can be estimated
% in a single finite difference.
%
% Copyright 1990-2006 The MathWorks, Inc.
% $Revision: 1.1.6.1 $ $Date: 2009/07/06 20:45:55 $
%
[m,n] = size(J);
if nargin < 2,
```

```

    p = 1:n;
end
J = J(:,p);
group = zeros(n,1);
ncol = 0;
J = spones(J);
while any(group==0)
    % Build group for ncol
    ncol = ncol + 1;
    rows = zeros(m,1);
    index = find(group == 0);
    lenindex = length(index);
    for i = 1:lenindex
        k = index(i);
        inner = J(:,k)'*rows;
        if inner == 0
            group(k) = ncol;
            rows = rows + J(:,k);
        end
    end
end
end
group(p)= group;

```

【例 1-4】利用 `clear` 命令和 `clc` 命令，清除内存变量和命令窗口中的所有显示内容。在命令窗口中输入如下语句：

```

x=1;
y=8;
z=x+y

```

命令窗口中的输出结果如下所示，并且命令窗口和工作空间窗口分别如图 1-39 和图 1-40 所示。

```

z =
    9

```

继续在命令窗口中输入如下语句：

```
clear      %清除内存变量
```

工作空间窗口如图 1-41 所示。

最后在命令窗口中输入如下语句：

```
clc      %清除命令窗中的所有显示内容
```

命令窗口如图 1-42 所示

2. 编辑输入内容

为了便于对所输入的内容进行编辑，MATLAB 提供了一些控制光标位置和对内容进行编辑的组合键，有效地利用这些按键可以简化操作、提高效率。

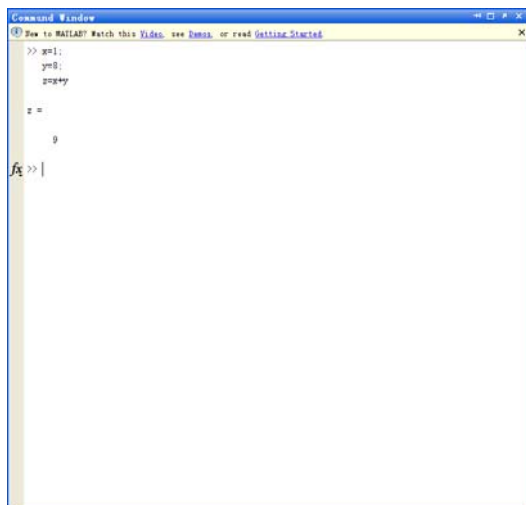


图 1-39 命令窗口

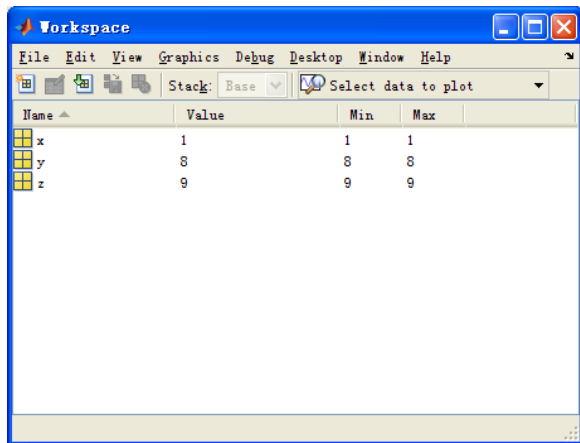


图 1-40 工作空间窗口

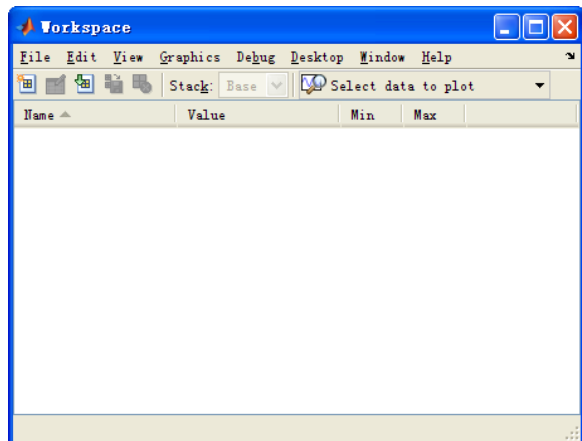


图 1-41 工作空间窗口

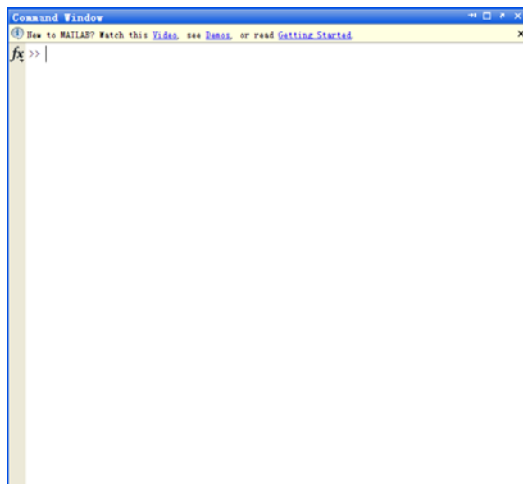


图 1-42 命令窗口

表 1-9 列出了常用的键盘按键及其功能。

表 1-9 常用键盘按键及其功能

键盘按键	功能	键盘按键	功能
↑	Ctrl+P, 调用上一行	Home	Ctrl+A, 光标置于当前行开头
↓	Ctrl+N, 调用下一行	End	Ctrl+E, 光标置于当前行末尾
←	Ctrl+B, 光标左移一个字符	Esc	Ctrl+U, 清除当前输入行
→	Ctrl+F, 光标右移一个字符	Del	Ctrl+D, 删除光标后的字符
Ctrl+←	Ctrl+L, 光标左移一个单词	Backspace	Ctrl+H, 删除光标前的字符
Ctrl+→	Ctrl+R, 光标右移一个单词	Alt+Backspace	恢复上一次删除

对于输入内容的编辑，需要说明的是：

- 当命令后面有分号时，按“Enter”键后，命令窗口中不显示运算结果；如果命令后面无分号，则在命令窗口中显示运算结果。
- 当用户需要输入多条语句后同时执行时，则在输入下一条语句时，按“Shift+Enter”键进行换行输入。

- 当用户需要使用复制或粘贴功能时，也可通过按“Ctrl+C”键或“Ctrl+V”键来实现。
- 当用户需要输入一条较长语句时，可以使用“...”来实现语句的换行，此形式和输入一行语句是等价的。

【例 1-5】比较使用“;”和不使用“;”的区别。

在命令窗口中输入如下语句：

```
x=rand(2,4);  
y=rand(2,4)  
A1=cos(x)  
A2=sin(y)
```

命令窗口中的输出结果如下所示：

```
y =  
    0.1299    0.4694    0.3371    0.7943  
    0.5688    0.0119    0.1622    0.3112  
A1 =  
    0.7291    0.8431    0.9985    0.7115  
    0.9285    0.9971    0.8624    0.5946  
A2 =  
    0.1295    0.4523    0.3308    0.7134  
    0.5386    0.0119    0.1615    0.3062
```

【例 1-6】输入多条语句，然后同时执行这些语句。

在命令窗口中输入如下语句：

```
x=rand(1,5)  
y=[0.01 1.23 3.45 5.67 7.89 ]  
A1=x-y  
A2=cos(x)  
A3=A1'*A2
```

命令窗口中的输出结果如下所示：

```
x =  
    0.1966    0.2511    0.6160    0.4733    0.3517  
y =  
    0.0100    1.2300    3.4500    5.6700    7.8900  
A1 =  
    0.1866   -0.9789   -2.8340   -5.1967   -7.5383  
A2 =  
    0.9807    0.9686    0.8162    0.8901    0.9388  
A3 =  
    0.1830    0.1807    0.1523    0.1661    0.1752  
   -0.9601   -0.9482   -0.7990   -0.8713   -0.9190  
   -2.7794   -2.7451   -2.3130   -2.5224   -2.6605  
   -5.0966   -5.0338   -4.2414   -4.6255   -4.8787
```

-7.3931 -7.3020 -6.1526 -6.7097 -7.0770

【例 1-7】当输入的语句较长时通过加“...”来实现换行。

在命令窗口中输入如下语句：

```
a=5.64166481234133+...
46413487456
b=9.54354798646156+...
66112346133
c=a+b
```

命令窗口中的输出结果如下所示：

```
a =
    4.641348746164166e+010

b =
    6.611234614254355e+010

c =
    1.125258336041852e+011
```

3. 标点符号

在 MATLAB 中，要合理地运用标点符号，因为一些标点符号被赋予特殊的意义或代表一定的运算，具体内容如表 1-10 所示。

表 1-10 标点用法

标点	功能	标点	功能
：（冒号）	具有多种应用功能	%（百分号）	注释标记
；（分号）	区分行及取消运行结果显示	！（惊叹号）	调用操作系统运算
，（逗号）	区分列及函数参数	=（等号）	赋值标记
()（括号）	指定运算的优先级	"（单引号）	字符串的标识符
[]（方括号）	定义矩阵	.（句号）	小数点及对象域访问
{ }（大括号）	构造单元数组	...（省略号）	续行符号

1.6 MATLAB 的工具箱简介

MATLAB 拥有几十个功能强大的工具箱，每一个工具箱都是为某一学科和专业应用而特别制定的，这是 MATLAB 语言能够快速发展的重要因素之一。

MATLAB 的工具箱大致可分为两类：一类是功能性工具箱，即通用型；另一类是领域性工具箱，即专用型。功能性工具箱主要用来扩充 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能以及 与硬件实时交互功能，能够用于多种学科；领域性工具箱是学科专用工具箱，如控制系统、信号处理、模糊逻辑和金融工具箱等，只适用于相关专业。

通过单击“Start”→“Toolboxes”，可以看到如图 1-43 所示的 MATLAB 工具箱。

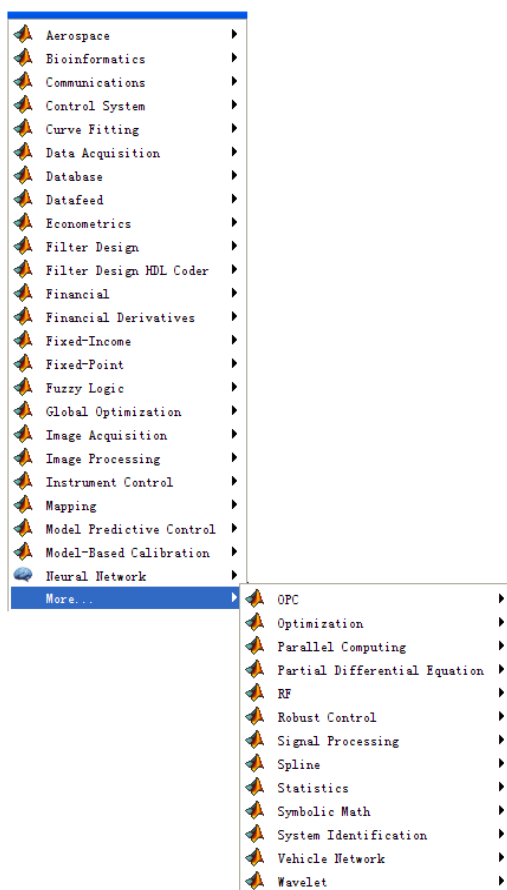


图 1-43 MATLAB 工具箱

下面列出一些 MATLAB 的工具箱：

- Bioinformatics Toolbox——生物工具箱。
- Control System Toolbox——控制系统工具箱。
- Curve Fitting Toolbox——曲线拟合工具箱。
- Communications Toolbox——通信工具箱。
- Database Toolbox——数据库工具箱。
- Fuzzy Logic Toolbox——模糊逻辑工具箱。
- Filter Design Toolbox——滤波器设计工具箱。
- Financial Toolbox——金融工具箱。
- Fixed-Income Toolbox——固定收入工具箱。
- Fixed-Point Toolbox——定点运算工具箱。
- Financial Derivatives Toolbox——金融误差分析工具箱。
- Instrument Control Toolbox——器具控制工具箱。
- Image Processing Toolbox——图像处理工具箱。
- Image Acquisition Toolbox——图像采集工具箱。
- Mapping Toolbox——地图绘制工具箱。
- Model Predictive Control Toolbox——模型预测控制工具箱。

- Model-Based Calibration Toolbox—— 基于模型标准工具箱。
- Neural Network Toolbox—— 神经网络工具箱。
- Optimization Toolbox—— 最优化工具箱。
- Partial Differential Equation Toolbox—— 偏微分方程工具箱。
- Robust Control Toolbox—— 鲁棒控制工具箱。
- System Identification Toolbox—— 系统辨识工具箱。
- Signal Processing Toolbox—— 信号处理工具箱。
- Spline Toolbox—— 样条工具箱。
- Statistics Toolbox—— 统计学工具箱。
- Symbolic Math Toolbox—— 符号数学工具箱。
- Wavelet Toolbox—— 小波工具箱。

1.7 MATLAB 的帮助查询功能

MATLAB 为用户提供了非常丰富的获得帮助信息的方式，如帮助窗口、在线帮助、帮助提示、HTML 格式帮助和 PDF 格式帮助等，极大地完善了该软件的功能，使用户能够通过帮助更好地学习 MATLAB，表 1-11 提供了常用的命令窗口中的帮助命令。

表 1-11 命令窗口中常用的帮助命令

命令	命令功能
Help	获得联机帮助
Demo	运行 MATLAB 演示程序
Doc	在帮助浏览器中显示指定函数的参考信息
Who	列出当前工作空间中的变量
Whos	列出当前工作空间中的变量的更多属性信息
What	列出当前目录下的 M 文件、MEX 文件和 MAT 文件
Which	显示指定函数或文件的路径
Lookfor	显示具有指定参数特征的函数的帮助信息
Helpbrowser	打开帮助内的浏览器
Helpdesk	运行 HTML 格式的帮助面板（Help Desk）
Helpwin	打开帮助内的窗口
Exist	查找指定变量或函数的存在性
Web	显示指定的网络页面

这里介绍三种获得帮助信息的方式：

- 帮助命令。
- 帮助窗口。
- Demo 演示。

1. 帮助命令

MATLAB 提供了 help 命令和 lookfor 命令用于查询帮助内容，这两个命令都是对 M 文件的第一行进行关键字搜索，不同的是 help 命令搜索与输入关键字完全匹配的内容，而 lookfor 命令搜索的条件较为宽松。

【例 1-8】在命令窗口中显示 MATLAB 的帮助信息。

在命令窗口中输入如下语句：

```
help demo
```

命令窗口中的输出结果如下所示：

DEMO Access product demos via Help browser.

DEMO opens the Help browser and selects the MATLAB Demos entry in the table of contents.

DEMO SUBTOPIC CATEGORY opens the Demos entry to the specified CATEGORY.

CATEGORY is a product or group within SUBTOPIC. SUBTOPIC is 'matlab', 'toolbox', 'simulink', 'blockset', or 'links and targets'. When SUBTOPIC is 'matlab' or 'simulink', do not specify CATEGORY to show all demos for the product.

Examples:

```
demo 'matlab'  
demo 'toolbox' 'signal'  
demo 'matlab' 'getting started'
```


See also echodemo, grabcode, help, helpbrowser.

Reference page in Help browser

doc demo

2. 帮助窗口

帮助窗口所给出的信息和帮助命令所给出的信息不同。进入帮助窗口有以下几种方式：

- 在命令窗口中输入 `helpwin`、`helpdesk` 或 `doc` 命令。
- 双击菜单栏中的问号图标 .
- 选择“Help”菜单中的“Product Help”、“Using the Desktop”和“Using the Command Window”选项。

【例 1-9】通过帮助命令给出余弦函数的信息。

在命令窗口中输入如下语句：

```
help cos
```

命令窗口中的输出结果如下所示：

COS Cosine of argument in radians.

COS(X) is the cosine of the elements of X.

See also acos, cosd.

Overloaded methods:

codistributed/cos

Reference page in Help browser

doc cos

【例 1-10】通过帮助窗口给出余弦函数的信息。
在搜索框中输入如下关键词：

COS

从 “Search Results” 选项卡中选择 “Relevance”，可以得到如图 1-44 所示的函数参考信息。

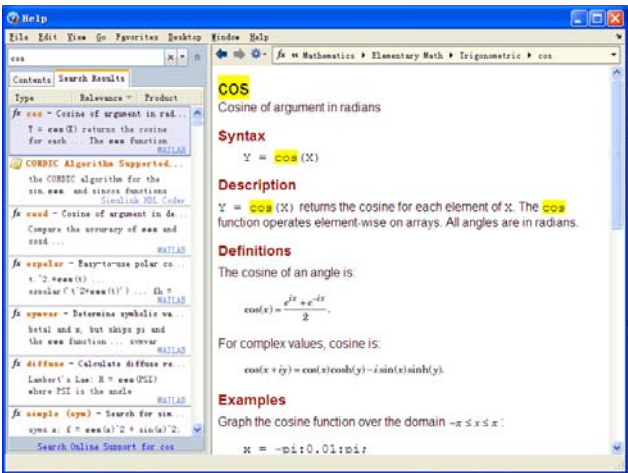


图 1-44 函数参考信息

【例 1-11】通过命令窗口进入帮助窗口。
在命令窗口中输入如下语句：

helpwin

进入的帮助窗口如图 1-45 所示。

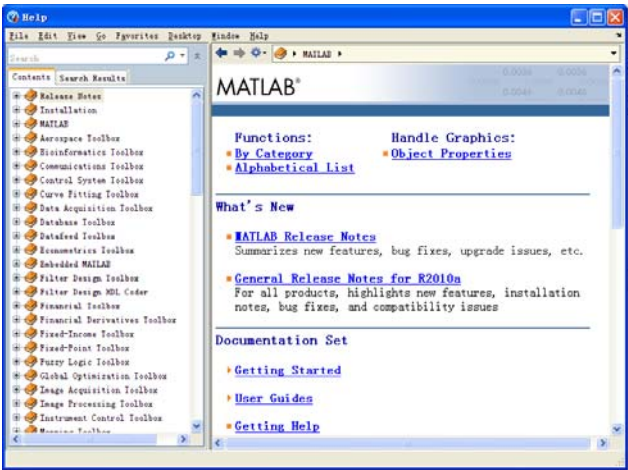


图 1-45 帮助窗口

帮助窗口所给出的信息还包括函数浏览器，通过函数浏览器可以快速调出函数信息。
选择 “Help” → “Function Browser”，可以打开如图 1-46 所示的窗口。

通过函数浏览器可以选择各种工具，如选择 “Optimization Toolbox” → “Minimization”，
如图 1-47 所示。



图 1-46 函数浏览器窗口

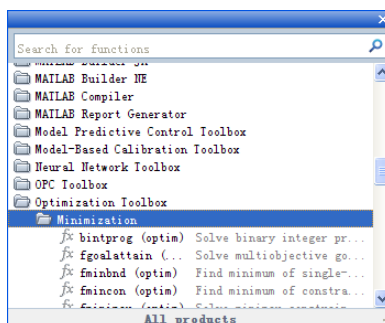


图 1-47 选择工具

3. Demo 演示

MATLAB 拥有很好的演示程序，该演示程序有交互式界面引导，操作很方便。可以通过选择“Help”→“Demos”，打开如图 1-48 所示的演示程序。



图 1-48 Demo 演示程序

【例 1-12】通过 MATLAB 的演示程序，使用户对单一的液压缸的模拟有一定的了解。

(1) 选择“Contents”→“Simulink”→“Demos”→“Single Hydraulic Cylinder Simulation”，可以得到如图 1-49 所示的演示系统。

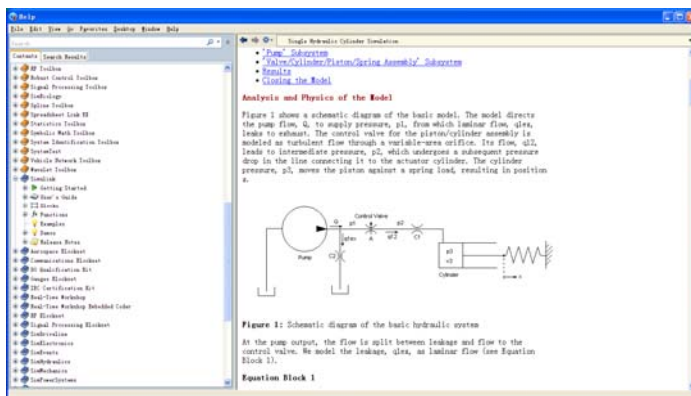


图 1-49 演示系统

(2) 单击右上方的“Open this model”，可以得到如图 1-50 所示 Simulink 模型。

(3) 单击 Simulink 模型上方的▶图标，运行 Simulink 模型后得到如图 1-51 所示的运行结果。

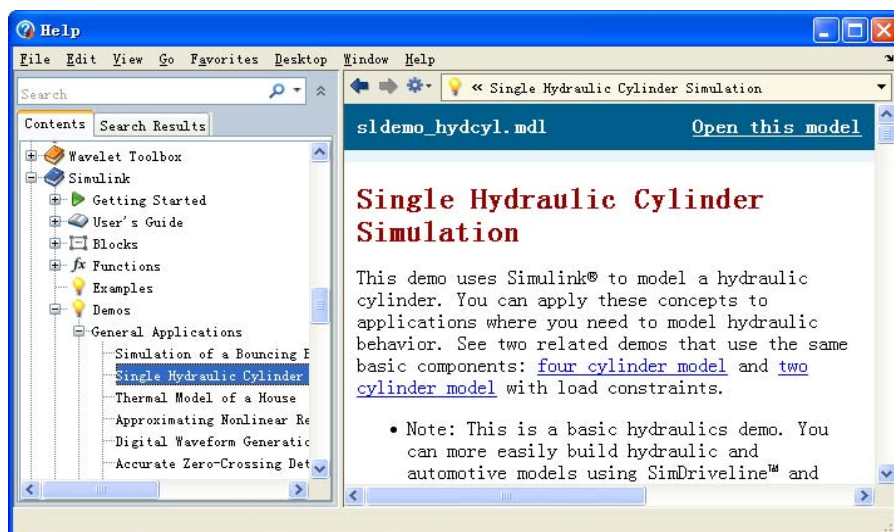


图 1-50 Simulink 模型

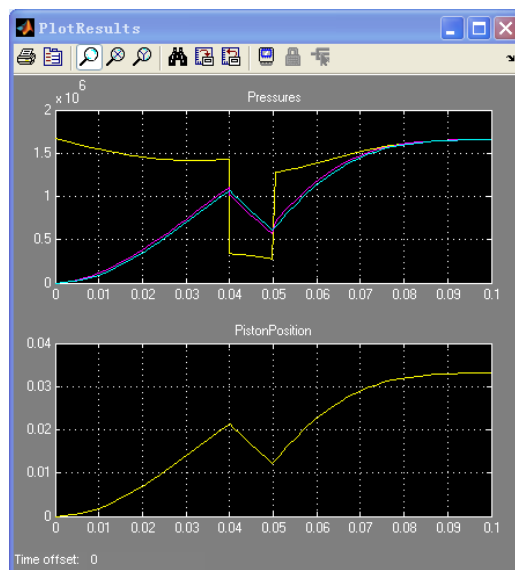


图 1-51 运行结果

第 2 章 MATLAB 数值计算

本章着重介绍 MATLAB 的数值计算，这部分是 MATLAB 的核心内容，包括 MATLAB 的数据类型、数组及其函数、矩阵及其函数、多项式及其函数、关系和逻辑及其运算等内容。本章的内容是 MATLAB 编程的基础，熟练掌握本章的内容对后面章节的学习会起到事半功倍的作用。

2.1 数据类型

MATLAB 中定义了很多数据类型，基本数据类型有字符串类型、数值类型、函数句柄、逻辑类型、结构体类型、Java 类型、细胞数组类型等，用户也可以根据需求定义自己的数据类型。本节讨论主要的数据类型及其在 MATLAB 中的使用方法。

MATLAB 中的数据都是通过数组的形式进行存储和运算的，并且对数组的维数不做要求。

2.1.1 字符串（String）类型

字符和字符串运算是各种高级语言极其重要的部分，应用广泛。MATLAB 提供了强大的字符串处理能力。

MATLAB 中的字符串是由多个字符按行向量的形式组成的，需要使用单引号对 “ ” 标识，并且字符串中的每个字符（包括空格）对应行向量的一个元素。

字符串的创建可以采用直接赋值和 char 函数两种方法。

【例 2-1】使用两种方法建立字符串。

在命令窗口中输入如下语句：

```
str1='34567'  
str2=char([51 52 53 54 55])
```

命令窗口中的输出结果如下所示：

```
str1 =  
34567  
  
str2 =  
34567
```

本例需要说明的是，使用 char 函数时应了解字符所对应的 ASCII 码。

对于字符串数组，每行的字符总数必须相等。

【例 2-2】建立字符串数组。

在命令窗口中输入如下语句：

```
str=['20','23','34','33','44','55']
```

命令窗口中的输出结果如下所示：

```
str =
```

202334

334455

字符串的水平合并可以使用中括号对 “[]” 和 `strcat` 函数两种方法。

【例 2-3】实现字符串的水平合并。

在命令窗口中输入如下语句：

```
str1=['Welcome to ','China ']  
str2=strcat('Welcome to ','China ')
```

命令窗口中的输出结果如下所示：

```
str1 =  
Welcome to China  
  
str2 =  
Welcome toChina
```

本例中，'Welcome to '包含两个空格，一个位于两单词之间，一个位于字符串结尾。从上面的结果不难看出，使用中括号对的方法将保留所有字符，且字符串的分隔符为逗号“，”，而使用 `strcat` 函数的方法将自动去除字符串结尾的空格，无论哪种方法得到的结果都是一个字符串。

字符串的垂直合并可以使用中括号对 “[]” 和 `strvcat` 函数两种方法。

【例 2-4】实现字符串的垂直合并。

在命令窗口中输入如下语句：

```
str3=['hello   '; 'everyone']  
str4=strvcat('hello','everyone')  
str5=['hello'; 'everyone']
```

命令窗口中的输出结果如下所示：

```
str3 =  
Name  
Class  
  
str4 =  
Name  
Class  
  
??? Error using ==> vertcat  
CAT arguments dimensions are not consistent.
```

从上面的结果不难看出，使用中括号对的方法必须保证每个字符串的长度相等（否则报错），且字符串的分隔符为分号“；”，而使用 `strvcat` 函数的方法将自动为较短的字符串补充空格，以满足每个字符串的长度相等，无论哪种方法得到的结果都是一个字符串。

在命令窗口中输入如下语句，同样可以得到上述 `str4` 的结果：

```
str41='hello';  
str42='everyone';  
str4= strvcat(str41,str42)
```

2.1.2 数值 (Numeric) 类型

MATLAB 的数值类型包括整数、复数、浮点数、无穷大和非数值量 5 种，其中正无穷大由 `Inf` 表示，负无穷大由 `-Inf` 表示，非数值量由 `NaN` 表示。

1. 整数类型

MATLAB 中整数类型分为有符号整数和无符号整数两类。两者的区别是有符号整数可以表示负数、正数和零，而无符号整数只能表示正整数和零，不能表示负数。MATLAB 支持 8、16、32、64 位的有符号整数和无符号整数。

需要说明的是，应用时在满足要求的情况下，尽可能用字节数少的类型来表示数据，这样可以提高运算速度，减少数据所占空间。

表 2-1 提供了上述 8 种整数类型的名称、表示范围和类型转换函数。

表 2-1 整数类型及其表示范围

类型名称	表示范围	类型转换函数
有符号 8 位整数	$-2^7 \sim 2^7-1$	<code>int8</code>
有符号 16 位整数	$-2^{15} \sim 2^{15}-1$	<code>int16</code>
有符号 32 位整数	$-2^{31} \sim 2^{31}-1$	<code>int32</code>
有符号 64 位整数	$-2^{63} \sim 2^{63}-1$	<code>int64</code>
无符号 8 位整数	$0 \sim 2^8-1$	<code>uint8</code>
无符号 16 位整数	$0 \sim 2^{16}-1$	<code>uint16</code>
无符号 32 位整数	$0 \sim 2^{32}-1$	<code>uint32</code>
无符号 64 位整数	$0 \sim 2^{64}-1$	<code>uint64</code>

表中类型转换函数有两个作用：一个是可以把其他类型的数值强制转换为指定的整数类型，另一个是可以生成指定类型的整数。

【例 2-5】将不同数据转换为有符号 8 位整数。

在命令窗口中输入如下语句：

```
x1=int8(20)
x2=int8(210)
x3=int8(23.4)
x4=int8(17.9)
x5=int8(-18.1)
x6=int8(-150.3)
```

命令窗口中的输出结果如下所示：

```
x1 =
    20

x2 =
   127

x3 =
    23
```

```
x4 =  
    18
```

```
x5 =  
   -18
```

```
x6 =  
  -128
```

【例 2-6】将不同数据转换为无符号 16 位整数。

在命令窗口中输入如下语句：

```
x1=uint16(20)  
x2=uint16(210)  
x3=uint16(23.4)  
x4=uint16(17.9)  
x5=uint16(-18.1)  
x6=uint16(-150.3)
```

命令窗口中的输出结果如下所示：

```
x1 =  
    20  
  
x2 =  
   210  
  
x3 =  
    23  
  
x4 =  
    18  
  
x5 =  
     0  
  
x6 =  
     0
```

从上面的结果不难看出，如果数据超出了该类型的表示范围，取值为表示范围的两个界限之中距离近的；如果数据是小数，取其距离最近的表示范围内的整数。

2. 复数类型

复数包含实部和虚部两个部分，MATLAB 中的虚部可以用 *i* 或 *j* 来表示。

建立复数可以使用直接输入和 `complex` 函数两种方法。

【例 2-7】使用两种方法建立复数。

在命令窗口中输入如下语句：

```
a=2+3i
b=2+3*i
c=2+3*sqrt(-1)
x=2;
y=3;
z=complex(x,y)
z1=complex(x)
z2=complex(x,0)
```

命令窗口中的输出结果如下所示：

```
a =
    2.0000 + 3.0000i

b =
    2.0000 + 3.0000i

c =
    2.0000 + 3.0000i

z =
    2.0000 + 3.0000i

z1 =
     2

z2 =
     2
```

本例需要说明的是，使用 `complex` 函数时 `complex(x)` 等价于 `complex(x, 0)`。

3. 浮点数类型

在 MATLAB 中浮点数类型分为单精度浮点数和双精度浮点数两类。单精度浮点数以 4 字节存储；双精度浮点数以 8 字节存储，它是 MATLAB 中的默认数据类型。浮点数的类型转换函数包括 `single`（将数据转换成单精度浮点数）和 `double`（将数据转换成双精度浮点数）。

4. Inf 和 NaN

在 MATLAB 中用 `Inf` 和 `-Inf` 表示正无穷大和负无穷大，主要应用在除 0、结果溢出等情况下。需要说明的是，MATLAB 遇到除 0 运算，不会报错和终止程序的运行，而是将结果赋值成 `Inf`。

`NaN` 是 Not a Number 的缩写，它是 MATLAB 中定义的非数值量，主要应用在 `0/0`、`Inf/Inf` 等情况下。

【例 2-8】实现运算结果为 `Inf` 或 `-Inf`。

在命令窗口中输入如下语句：

```
x=9/0  
y=tand(90)  
z = log(0)
```

命令窗口中的输出结果如下所示：

```
x =  
    Inf  
  
y =  
    Inf  
  
z =  
   -Inf
```

【例 2-9】实现运算结果为 NaN。

在命令窗口中输入如下语句：

```
0/0  
inf/inf
```

命令窗口中的输出结果如下所示：

```
ans =  
    NaN  
  
ans =  
    NaN
```

2.1.3 函数句柄（Handle）

MATLAB 中的函数句柄可以间接调用函数，它的具体用法如下：

- `fhandle=@functionname`：functionname 为函数名，fhandle 为该函数对应的函数句柄。

【例 2-10】利用函数句柄调用 MATLAB 中自带的正弦函数。

在命令窗口中输入如下语句：

```
mysin=@sin;  
mysin(0)
```

命令窗口中的输出结果如下所示：

```
ans =  
    0
```

其中，`mysin(0)`等价于 `sin(0)`。

2.1.4 逻辑（Logical）类型

在 MATLAB 中，用 1 表示逻辑真（true），用 0 表示逻辑假（false）。logical 函数可以将 0 转换为逻辑 0，把非零数值转换为逻辑 1。

【例 2-11】使用 logical 函数将数值转换为逻辑类型。

在命令窗口中输入如下语句：

```
x=logical(0)
y=logical(-1.5)
z=logical(-Inf)
```

命令窗口中的输出结果如下所示：

```
x =
    0

y =
    1

z =
    1
```

2.1.5 结构体 (Structure) 类型

在 MATLAB 中，结构体类型的作用在于，可以将不同类型但在逻辑上相关的数据组成一个有机的整体。换句话说，结构体可以看做一个数据容器，或是一种由若干属性组成的数组，其中每个属性都可以是任意数据类型。

结构体的示例如图 2-1 所示，结构体 `patient` 中包含 3 个字段：姓名字段 `name` 中存储了一个字符串类型的数据；账单字段 `billing` 中存储了一个浮点数值；`test` 字段中存储了一个二维数组。一个结构体可以具有多个字段，它们又可以存储不同类型的数据，通过这种方式，就可以把多个不同类型的数据组织在一个结构体对象中。

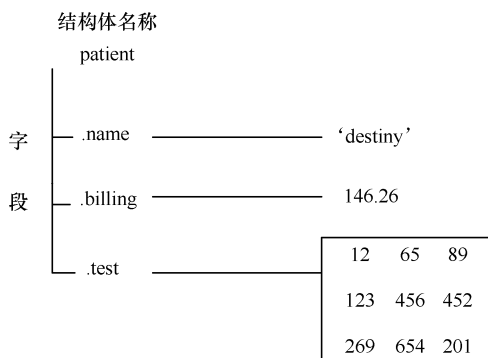


图 2-1 结构体 `patient` 的示例

在 MATLAB 中，一个结构体对象就是一个 1×1 的结构体数组，因此可以创建具有多个结构体对象的二维或多维结构体数组。

1. 结构体数组的创建

创建结构体数组有两种方法：一种是通过赋值语句，另外一种是通过 `struct` 函数。

(1) 通过赋值语句创建结构体数组

在对结构体的字段进行赋值时，赋值表达式的左边代表了结构体的字段变量名，用“结构体名称.字段名称”的形式书写，可以对一个结构体的多个字段赋值。

【例 2-12】创建如图 2-1 所示的结构体数组 `patient`。

在命令窗口中输入如下语句：

```
patient.name='destiny';
patient.billing=146.26;
patient.test=[12 65 89;123 456 452;269 654 201];
patient
```

命令窗口中的输出结果如下所示：

```
patient =
    name: 'destiny'
   billing: 146.2600
    test: [3x3 double]
```

本例通过分别对 name、billing 和 test 属性赋值，创建了一个具有 3 个字段的结构体对象 patient。

在命令窗口中输入 whos 语句，命令窗口中的输出结果如下所示：

Name	Size	Bytes	Class	Attributes
patient	1x1	466	struct	

用 whos 语句显示发现 patient 是一个 1×1 的结构体数组。

通过圆括号索引指派，用字段赋值的方法也可以创建结构体数组。

需要注意的是，同一个结构体数组中的所有结构体对象具有相同的字段，没有明确赋值字段，MATLAB 默认赋值为空数组。

【例 2-13】通过圆括号索引指派，用字段赋值的方法创建结构体数组。

在命令窗口中输入如下语句：

```
patient(1).name='destiny';
patient(1).billing=146.26;
patient(1).test=[12 65 89;123 456 452;269 654 201];
whos
```

命令窗口中的输出结果如下所示：

Name	Size	Bytes	Class	Attributes
patient	1x1	466	struct	

在命令窗口中输入如下语句：

```
patient(2).name='lili';
patient(2).billing=133.22;
patient(2).test=[20 32 45 ;111 222 333;456 120 320];
whos
```

命令窗口中的输出结果如下所示：

Name	Size	Bytes	Class	Attributes
patient	1x2	734	struct	

在命令窗口中输入如下语句：

```
patient
```

命令窗口中的输出结果如下所示：

```
patient =
```

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

```
    name  
    billing  
    test
```

命令窗口中输入如下语句：

```
patient(3).name='tom';  
patient
```

命令窗口中的输出结果如下所示：

```
patient =  
1x3 struct array with fields:  
    name  
    billing  
    test
```

在命令窗口中输入如下语句：

```
patient(3).billing
```

命令窗口中的输出结果如下所示：

```
ans =  
[]
```

【例 2-14】创建 2×2 的结构体数组 `patient`。

在命令窗口中输入如下语句：

```
patient(1,1).name='destiny';  
patient(1,1).billing=146.26;  
patient(1,1).test=[12 65 89;123 456 452;269 654 201];  
patient(1,2).name='lili';  
patient(1,2).billing=133.22;  
patient(1,2).test=[20 32 45 ;111 222 333;456 120 320];  
%patient(2,1).name='tom';  
%patient(2,2).billing=96.22;  
patient
```

命令窗口中的输出结果如下所示：

```
patient =  
2x2 struct array with fields:  
    name  
    billing  
    test
```

(2) 通过 `struct` 函数创建结构体数组

除了通过字段赋值，还可以用 `struct` 函数创建结构体数组。

`struct` 函数的具体用法如下：

```
strArray = struct('field1',val1,'field2',val2, ...)
```

【例 2-15】创建 2×2 的结构体数组 `patient`。

在命令窗口中输入如下语句：

```
patient=struct('name',{'destiny','lili','tom',[]},'billing',{146.26,133.22;96.22,[]},...
              'test',{[12 65 89;123 456 452;269 654 201], [20 32 45 ;111 222 333;456 120 320];[],[]})
```

命令窗口中的输出结果如下所示：

```
patient =
2x2 struct array with fields:
    name
    billing
    test
```

2. 结构体数组的访问

访问结构体数组可以通过下标索引和属性名的方式。

【例 2-16】读取如图 2-2 所示的结构体数组 `patient` 的数据。

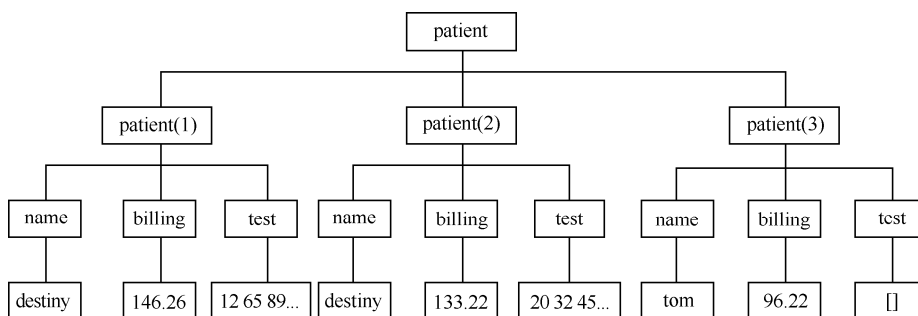


图 2-2 结构体数组示例

首先创建如图 2-2 所示的结构体数组，在命令窗口中输入如下语句：

```
patient=struct('name',{'destiny','lili','tom',[]},'billing',{146.26,133.22;96.22},...
              'test',{[12 65 89;123 456 452;269 654 201], [20 32 45 ;111 222 333;456 120 320];[],[]})
```

命令窗口中的输出结果如下所示：

```
patient =
1x3 struct array with fields:
    name
    billing
    test
```

其次读取结构体数组的任意元素，在命令窗口中输入如下语句：

```
New patient = patient (1)
```

命令窗口中的输出结果如下所示：

```
Newpatient =
    name: 'destiny'
  billing: 146.2600
    test: [3x3 double]
```

再次读取结构体数组的任意属性，在命令窗口中输入如下语句：

```
[A B C]= patient.name
E=[ patient.name]
```

```
F={ patient.name}
```

```
F(2)
```

命令窗口中的输出结果如下所示:

```
A =
```

```
destiny
```

```
B =
```

```
lili
```

```
C =
```

```
tom
```

```
E =
```

```
destinylilitom
```

```
F =
```

```
'destiny'    'lili'    'tom'
```

```
ans =
```

```
'lili'
```

最后读取结构体数组任意元素的任意属性, 在命令窗口中输入如下语句:

```
patient (2).name
```

命令窗口中的输出结果如下所示:

```
ans =
```

```
lili
```

通过上面的方法可以实现对结构体数组的读取, 对数据的修改与之类似。

3. 结构体数组的操作

对结构体数组的常见操作, 包括获取其尺寸信息和增删字段。

和一般的数组一样, `size` 函数可以获取结构体数组的尺寸, 也就是数组中包含多少行、多少列结构体对象。当然, 如果 `size` 函数的输入参数是结构体字段, 则返回的是该字段的尺寸信息。

向结构体中增加新的字段, 只需要添加新的字段赋值语句即可。没有被明确赋值的结构体中, 添加的新字段被初始化为空数组。

在结构体数组中删除字段, 利用 `rmfield` 函数即可实现, 具体用法如下所示:

- `newstrArray=rmfield(strArray,'fieldname')`: 表示从结构体数组 `strArray` 中删除 `fieldname` 字段, 并将结果赋值到新的结构体数组 `newstrArray` 中。

【例 2-17】在结构体中增加和删除字段, 并且为新字段赋值。

首先创建结构体, 在命令窗口中输入如下语句:

```
patient =struct('name',{ 'destiny'},'billing',{ 146.26},'test',{[ 12 65 89;123 456 452;269 654 201]})
```

命令窗口中的输出结果如下所示:

```
patient =  
    name: 'destiny'  
    billing: 146.2600  
    test: [3x3 double]
```

其次增加新属性并赋值，在命令窗口中输入如下语句：

```
patient.Age=27
```

命令窗口中的输出结果如下所示：

```
patient =  
    name: 'destiny'  
    billing: 146.2600  
    test: [3x3 double]  
    Age: 27
```

再次删除 test 属性，在命令窗口中输入如下语句：

```
patient=rmfield(patient,'test')
```

命令窗口中的输出结果如下所示：

```
patient =  
    name: 'destiny'  
    billing: 146.2600  
    Age: 27
```

2.1.6 细胞数组 (Cell) 类型

在 MATLAB 中细胞数组类型的作用在于，可以把不同类型的数据归并到一个数组中。细胞数组类型的每一个元素称做一个单元，每一个单元都可以包含一个任意数组，如数值数组、字符串数组、结构体数组或另一个细胞数组。

细胞数组示例如图 2-3 所示，它包含 6 个单元数组，显然其每个单元数组内的数据类型均不相同。

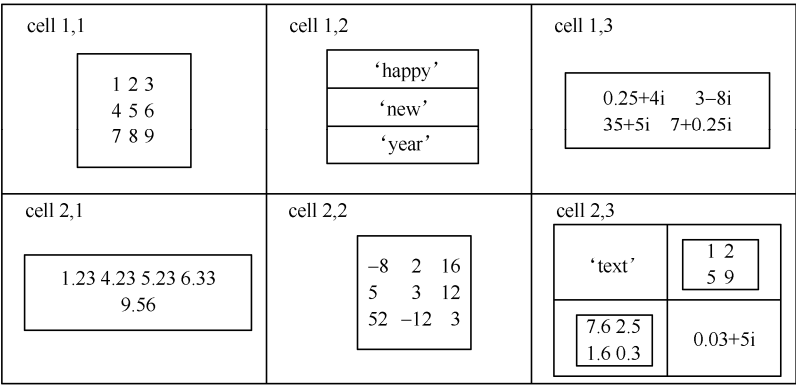


图 2-3 细胞数组示例

1. 细胞数组的创建
- 和结构体的创建类似，细胞数组的创建也有两种方法：直接对各个元胞赋值，或者用 cell 函数创建。

(1) 使用赋值语句创建细胞数组

赋值语句中需要用到花括号对“{}”。当花括号对用在赋值语句左侧的下标索引时，右侧为细胞单元的数据；当花括号对用在赋值语句右侧的细胞单元数据时，左侧为细胞单元的下标索引。具体用法如下所示：

- `c{1,1}='Clayton'`
- `c(1,1)={'Clayton'}`

(2) 使用 cell 函数创建细胞数组

在 MATLAB 中 cell 函数的具体用法如下：

- `arrayName=cell(m,n)`：创建一个 $m \times n$ 的空细胞数组，其单元为空矩阵。

【例 2-18】创建一个 2×3 的细胞数组并赋值。

首先创建一个 2×3 的细胞数组，在命令窗口中输入如下语句：

```
A=cell(3,3)
```

命令窗口中的输出结果如下所示：

```
A =
    []    []    []
    []    []    []
    []    []    []
```

其次对第 2 行第 2 列的单元赋值，在命令窗口中输入如下语句：

```
A{3,2}=eye(3)
```

命令窗口中的输出结果如下所示：

```
A =
    []          []    []
    []    [2x2 double]    []
    []    [3x3 double]    []
```

2. 细胞数组的显示

MATLAB 提供了 `celldisp` 函数显示细胞数组的具体数据。

【例 2-19】显示细胞数组的数据。

在命令窗口中输入如下语句：

```
A= eye(5,5);
b=1+2i;
str='MATLAB';
M={'Nancy', b, str,A};
celldisp(M)
```

命令窗口中的输出结果如下所示：

```
M{1} =Nancy

M{2} =
    1.0000 + 2.0000i

M{3} =MATLAB
```

M{4} =

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

3. 细胞数组的访问

MATLAB 主要利用下标索引来访问细胞数组元素。

【例 2-20】访问细胞数组中指定单元的数据。

在命令窗口中输入如下语句：

```
A= eye(5,5);
b=1+2i;
str='MATLAB';
M={'Nancy', b, str,A};
B=M{1,4}
C=M(1,4)
```

命令窗口中的输出结果如下所示：

```
B =
    1     0     0     0     0
    0     1     0     0     0
    0     0     1     0     0
    0     0     0     1     0
    0     0     0     0     1
```

```
C =
[5x5 double]
```

当用花括号括起下标索引时，返回细胞单元的数据；当用小括号括起下标索引时，返回细胞数组的子数组，结果仍为细胞数组。

4. 细胞数组的操作

如果要删除细胞数组的某个指定单元，只需将空矩阵赋给该单元。

【例 2-21】删除细胞数组的某个指定单元。

在命令窗口中输入如下语句：

```
A= eye(5,5);
b=1+2i;
str='MATLAB';
M={'Nancy', b, str,A};
M{1,3}=[]
```

命令窗口中的输出结果如下所示：

```
M =
```

```
'Nancy' [1.0000 + 2.0000i] [] [5x5 double]
```

如果要改变细胞数组的形状，可以利用 `reshape` 函数。

【例 2-22】改变细胞数组的形状。

在命令窗口中输入如下语句：

```
A=eye(5,5);  
b=1+2i;  
str='MATLAB';  
M={'Nancy', b, str,A};  
B=reshape(M,4,1)
```

命令窗口中的输出结果如下所示：

```
B =  
    'Nancy'  
    [1.0000 + 2.0000i]  
    'MATLAB'  
    [5x5 double]
```

2.2 数组及其函数

数组及数组运算都是 MATLAB 的重要内容，几乎所有的 MATLAB 数据，无论什么类型，都是以数组的形式储存的。数组函数是指可以对数组进行运算操作的函数，如三角函数、指数函数和对数函数等。

2.2.1 数组的建立和操作

数组可以是一维的“行”或“列”，也可以是二维的“矩形”，或是具有更高的维数，所以数组可以分为一维数组、二维数组和高维数组。针对不同维数的数组，MATLAB 可以分别创建不同要求的数组类型。

1. 一维数组的建立

(1) 直接输入法

这种方法是最直观、最通用的方法。

【例 2-23】利用直接输入法建立一维数组。

在命令窗口中输入如下语句：

```
x=[5.6 pi*4 sqrt(2) 1+2i]
```

命令窗口中的输出结果如下所示：

```
x =  
    5.6000    12.5664    1.4142    1.0000 + 2.0000i
```

(2) 冒号生成法

这种方法是通过设定采样起点、步长和终止边界生成一维“行”数组，其定义形式如下：

• $x=a:inc:b$: a 表示数组的第一个元素，即起点。 inc 是创建数组之间的间隔，也就是步长， inc 可以是正数也可以是负数，默认值是 1。如果 inc 是正数，要求 $a < b$ ；反之亦然。如果 $(b-a)/inc$

是整数, b 是生成数组的最后一个元素, 否则生成数组的最后一个元素不超过 a 和 b 构成的范围。 a 、 inc 和 b 之间用冒号“:”隔开。

【例 2-24】利用冒号生成法建立一维数组。

在命令窗口中输入如下语句:

```
x=2:2:16
y=-2:-2:-11.2
z=3:8
```

命令窗口中的输出结果如下所示:

```
x =
     2     4     6     8    10    12    14    16

y =
    -2    -4    -6    -8   -10

z =
     3     4     5     6     7     8
```

(3) 定数线性采样法

这种方法是使用 MATLAB 提供的 `linspace` 函数生成一维数组。

`linspace` 函数的具体用法如下:

- $x=\text{linspace}(a, b, n)$: a 、 b 分别是生成数组的第一个和最后一个元素, 含两个端点共计均匀采样 n 个点, 即生成的数组维数是 $1 \times n$ 。此指令与 $x=a:(a-b)/(n-1):b$ 作用相同。

(4) 定数对数采样法

这种方法是使用 MATLAB 提供的 `logspace` 函数生成一维数组。

`logspace` 函数的具体用法如下:

- $x=\text{logspace}(a, b, n)$: a 、 b 分别是指数系数的第一个和最后一个元素, 含两个端点共计均匀采样 n 个点。计算上述系数对应的以 10 为底的指数, 即 10^a 、 10^b 分别是生成数组的第一个和最后一个元素, 生成的数组维数是 $1 \times n$ 。

【例 2-25】利用 `linspace` 函数和 `logspace` 函数建立一维数组。

在命令窗口中输入如下语句:

```
x=0.1:0.1:0.6
y= linspace(0, 0.5, 8)
z= logspace(0, 0.5, 8)
```

命令窗口中的输出结果如下所示:

```
x =
    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000

y =
     0    0.0714    0.1429    0.2143    0.2857    0.3571    0.4286    0.5000

z =
```

1.0000	1.1788	1.3895	1.6379	1.9307	2.2758	2.6827	3.1623
--------	--------	--------	--------	--------	--------	--------	--------

2. 一维数组的访问

一维数组的访问主要是通过下标索引的方式。

【例 2-26】利用下标索引访问一维数组。

在命令窗口中输入如下语句：

```
rand('state',0);    % 设置随机数种子
x=rand(1,8)         % 随机生成行向量，每个元素都满足[0,1]上的均匀分布
y1=x(3)             % 访问第 3 个元素
y2=x([1 3 5 7])     % 访问第 1、3、5 和 7 个元素
y3=x(2:8)           % 访问第 2、3、4、5、6、7 和 8 个元素
y4=x(6:-2:2)        % 访问第 6、4 和 2 个元素，输出次序同访问次序
y5=x(4:end)         % 访问第 4 个到最后一个元素
y6=x(find(x<0.6))   % 访问小于 0.6 的元素
y7=x([5 4 3 2 1 2 3]) % 访问指定下标索引序列的元素，输出次序同访问次序
x(5)=0.5            % 为第 5 个元素赋值
```

命令窗口中的输出结果如下所示：

```
x =
    0.9501    0.2311    0.6068    0.4860    0.8913    0.7621    0.4565    0.0185

y1 =
    0.6068

y2 =
    0.9501    0.6068    0.8913    0.4565

y3 =
    0.2311    0.6068    0.4860    0.8913    0.7621    0.4565    0.0185

y4 =
    0.7621    0.4860    0.2311

y5 =
    0.4860    0.8913    0.7621    0.4565    0.0185

y6 =
    0.2311    0.4860    0.4565    0.0185

y7 =
    0.8913    0.4860    0.6068    0.2311    0.9501    0.9501    0.2311    0.6068
```

```
x =
    0.9501    0.2311    0.6068    0.4860    0.5000    0.7621    0.4565    0.0185
```

由上面的结果不难看出，一维数组的访问方法是非常灵活的，主要通过 $x(\text{index})$ 的方式进行访问， index 可以是一个正整数也可以是正整数数组， index 中的每个元素都要在 $[1, \text{end}]$ 内，其中 end 指数组的长度，即最后一个元素的下标。

3. 二维数组的建立

从数据结构的角，二维数组和矩阵没有什么本质区别。

(1) 直接输入法

对于维数比较小的数组，直接输入法较其他方法更加简便。用该方法建立二维数组时需要满足以下几点要求：

- 数组要以方括号对 “[]” 作为创建的首尾。
- 数组的行与行之间要用分号 “;” 隔开。需要注意的是，当分号位于中括号对内时，是数组行之间的分隔符，当位于语句后时是语句的结束符，并表示该语句的结果不显示在屏幕上。
- 每行中的元素要用逗号 “,” 或空格隔开。需要注意的是，当逗号位于中括号对内时，是每行元素之间的分隔符，当位于语句后时是语句的结束符，并表示该语句的结果显示在屏幕上。

【例 2-27】利用直接输入法创建数组。

在命令窗口中输入如下语句：

```
x=1.252;
y=3.256;
z=[2, 5*x+i*y, 2*y; 4.8+2i, y*sqrt(x), sin(pi/4)]
```

命令窗口中的输出结果如下所示：

```
z =
    2.0000         6.2600 + 3.2560i     6.5120
    4.8000 + 2.0000i     3.6432         0.7071
```

【例 2-28】利用直接输入法创建复数数组。

在命令窗口中输入如下语句：

```
a=[1,2,3;4,5,6];
b=[7,8,9;10,11,12];
c=a+i*b
```

命令窗口中的输出结果如下所示：

```
c =
    1.0000 + 7.0000i    2.0000 + 8.0000i    3.0000 + 9.0000i
    4.0000 +10.0000i    5.0000 +11.0000i    6.0000 +12.0000i
```

(2) 特殊函数生成法

这种方法利用 MATLAB 提供的函数生成二维数组，如 rand 函数和 randn 函数。

rand 函数的具体用法如下：

- $x=\text{rand}(m, n)$ ，随机生成 $m \times n$ 的二维数组，每个元素都满足 $[0,1]$ 上的均匀分布。

randn 函数的具体用法如下：

- `x=randn(m,n)`, 随机生成 $m \times n$ 的二维数组, 每个元素都满足标准正态分布。

4. 二维数组的访问

二维数组的访问方式主要有单下标索引、双下标索引和逻辑索引方式。其中, 单下标索引方式是只使用一个下标确定数组中元素的位置来访问原数组; 双下标索引方式是使用两个下标(行下标和列下标)确定数组中元素的位置来访问原数组; 逻辑索引方式是通过比较关系运算产生一个满足比较关系的索引数组, 利用此索引数组来访问原数组。

需要指出的是, 通过单下标索引方式访问数组时, 是将二维数组的所有元素按从左到右的顺序, 首尾相接排成一维的长列, 按照原数组自上而下编号, 通过编号访问原数组。MATLAB 提供了 `sub2ind` 函数和 `ind2sub` 函数, 实现单下标索引和双下标索引之间的转换。

`sub2ind` 函数将双下标索引转换为单下标索引, 具体用法如下:

- `IND=sub2ind(siz,I,J)`: `siz` 是转换数组的行列尺寸, 一般用 `size(A)`表示; `I`、`J` 分别是双下标索引的两个数字, `IND` 是转化后的单下标索引的数字。

`ind2sub` 函数将单下标索引转换为双下标索引, 具体用法如下:

- `[I,J]=ind2sub(siz,IND)`: `siz` 是转换数组的行列尺寸, 一般用 `size(A)`表示; `IND` 是单下标索引的数字; `I`、`J` 分别是转化后双下标索引的两个数字。

【例 2-29】利用下标索引访问二维数组。

在命令窗口中输入如下语句:

```
x=rand(3,4)
A=x(2,3)      %双下标索引方式访问数组第2行第3列的元素
A1=x(3,1:4)   %双下标索引方式访问数组第3行,第1~4列的元素
A2=x([2,1],3) %双下标索引方式访问数组第2、1行,第3列的元素
A3=x([3,1],[3,4]) %双下标索引方式访问数组第3、1行,第3、4列的元素
B=x(2)        %单下标索引方式访问数组第2个元素(即第2行第1列元素)
B1=x(6)       %单下标索引方式访问数组第6个元素(即第3行第2列元素)
B2=x(7:11)    %单下标索引方式访问数组第7到11位的元素
% (即第3列第1、2、3行和第4列第1、2行元素)
```

在命令窗口中的输出结果如下所示:

```
x =
    0.8214    0.7919    0.1763    0.9169
    0.4447    0.9218    0.4057    0.4103
    0.6154    0.7382    0.9355    0.8936

A =
    0.4057

A1 =
    0.6154    0.7382    0.9355    0.8936

A2 =
    0.4057
```

```
0.1763
```

```
A3 =
```

```
0.9355    0.8936
```

```
0.1763    0.9169
```

```
B =
```

```
0.4447
```

```
B1 =
```

```
0.7382
```

```
B2 =
```

```
0.1763    0.4057    0.9355    0.9169    0.4103
```

【例 2-30】利用逻辑索引访问二维数组。

在命令窗口中输入如下语句：

```
A=magic(5)      %生成 4×4 的魔方矩阵
```

```
B=A>7          %通过比较关系运算产生一个索引数组
```

```
A(B)=99        %利用索引数组访问原数组，并给原数组重新赋值
```

命令窗口中的输出结果如下所示：

```
A =
```

```
17    24     1     8    15
```

```
23     5     7    14    16
```

```
4      6    13    20    22
```

```
10    12    19    21     3
```

```
11    18    25     2     9
```

```
B =
```

```
1      1     0     1     1
```

```
1      0     0     1     1
```

```
0      0     1     1     1
```

```
1      1     1     1     0
```

```
1      1     1     0     1
```

```
A =
```

```
99    99     1    99    99
```

```
99     5     7    99    99
```

```
4      6    99    99    99
```

```
99    99    99    99     3
```

```
99    99    99     2    99
```

【例 2-31】实现单下标索引和双下标索引之间的转换。

在命令窗口中输入如下语句：

```
A=[1 2 3 4;-4 3 3.5 5.2;1.23 4.32 3.22 5.25]    %创建一个 3×4 的矩阵
IND=sub2ind(size(A),2,3)                        %将双下标索引转换为单下标索引
A(IND)                                             %显示单下标索引对应的数组元素
[I,J]=ind2sub(size(A),8)                         %将单下标索引转换为双下标索引
```

在命令窗口中的输出结果如下所示：

```
A =
    1.0000    2.0000    3.0000    4.0000
   -4.0000    3.0000    3.5000    5.2000
    1.2300    4.3200    3.2200    5.2500

IND =
     8

ans =
    3.5000

I =
     2

J =
     3
```

5. 高维数组的建立和访问

二维数组可以看做由行和列确定的“数组面”，高维数组是数组面在多维空间上的扩展。

高维数组的建立有三种方法：

- 通过索引将二维数组扩展成高维数组。
- 利用内联函数建立高维数组，表 2-2 提供了常见的内联函数。
- 利用 cat 函数建立高维数组。

表 2-2 常见内联函数

函数名称	功能介绍
ones (a_1, a_2, a_3, \dots)	生成维数是 $a_1 * a_2 * a_3 \dots$ 的全 1 数组
zeros ($a_1 * a_2 * a_3 \dots$)	生成维数是 $a_1 * a_2 * a_3 \dots$ 的全 0 数组
rand ($a_1 * a_2 * a_3 \dots$)	生成维数是 $a_1 * a_2 * a_3 \dots$ 的数组，数组元素服从[0,1]均匀分布
randn ($a_1 * a_2 * a_3 \dots$)	生成维数是 $a_1 * a_2 * a_3 \dots$ 的数组，数组元素服从[0,1]标准正态分布
repmat ($m, [a_1 * a_2 * a_3 \dots]$)	生成维数是 $a_1 * a_2 * a_3 \dots$ 的数组，数组元素是 m

【例 2-32】利用二维数组扩展建立 $4 \times 4 \times 2$ 的高维数组。

在命令窗口中输入如下语句：

```
A=[1 2 3;4 5 6;7 8 9]
A(:,:,2)=ones(3)    %创建 3×3×2 的数组
```

在命令窗口中的输出结果如下所示：

```
A(:,:,1) =
```

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

```
A(:,:,2) =
```

```
1     1     1
1     1     1
1     1     1
```

【例 2-33】利用内联函数建立高维数组。

在命令窗口中输入如下语句：

```
A=ones(3,3,4)           % 生成维数是 3×3×4 的全 1 数组
```

```
A1=repmat(rand(2,1),[2,3,2]) % 生成维数是 2×3×2 的数组，数组元素是 2×1 维的随机数组
```

命令窗口中的输出结果如下所示：

```
A(:,:,1) =
```

```
1     1     1
1     1     1
1     1     1
```

```
A(:,:,2) =
```

```
1     1     1
1     1     1
1     1     1
```

```
A(:,:,3) =
```

```
1     1     1
1     1     1
1     1     1
```

```
A(:,:,4) =
```

```
1     1     1
1     1     1
1     1     1
```

```
A1(:,:,1) =
```

```
0.8147    0.8147    0.8147
0.9058    0.9058    0.9058
0.8147    0.8147    0.8147
0.9058    0.9058    0.9058
```

```
A1(:,:,2) =
```

```
0.8147    0.8147    0.8147
```

```
0.9058    0.9058    0.9058
0.8147    0.8147    0.8147
0.9058    0.9058    0.9058
```

【例 2-34】利用 cat 函数建立高维数组。

在命令窗口中输入如下语句：

```
A = cat(4,magic(4),ones(4))
```

命令窗口中的输出结果如下所示：

```
A(:,:,1,1) =
```

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

```
A(:,:,1,2) =
```

```
1     1     1     1
1     1     1     1
1     1     1     1
1     1     1     1
```

【例 2-35】利用 cat 函数建立高维数组。

在命令窗口中输入如下语句：

```
a = magic(4);
```

```
b = ones(4);
```

```
c = cat(4,a,b)
```

命令窗口中的输出结果如下所示：

```
c(:,:,1,1) =
```

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

```
c(:,:,1,2) =
```

```
1     1     1     1
1     1     1     1
1     1     1     1
1     1     1     1
```

上述例 2-34 和例 2-35 分别使用两种方式建立了相同的高维数组。

2.2.2 数组运算

MATLAB 提供了强大的数组运算功能，所谓数组运算是指运算施加于每个元素，而非数

组整体，这是与矩阵运算的差异。

MATLAB 提供了丰富的算术运算符、关系运算符和逻辑运算符，表 2-3 列出运算符并给出它们对应的运算优先等级。

表 2-3 运算符的优先等级

运算符	优先等级
括号 ()	
转秩 (.'), 幂 (.^), 复共轭转秩 (') , 矩阵幂 (^)	
一元正号 (+), 一元负号 (-), 逻辑非 (~)	
元素相乘 (.*), 元素右除 (./), 元素左除 (.\), 矩阵乘法 (*), 矩阵右除 (/), 矩阵左除 (\)	优先等级由高到低
加法 (+), 减法 (-)	
冒号运算符 (:)	
小于 (<), 小于等于 (<=), 大于 (>) 大 z 于等于 (>=), 等于 (==), 不等于 (~=)	
逻辑与 (&)	
逻辑或 ()	
短路逻辑与 (&&)	
短路逻辑或 ()	

1. 算术运算

【例 2-36】实现数组的基本运算。

在命令窗口中输入如下语句：

```
A=[2 1 3;0 1 2;1 1 3]
B=[11 20 30;5 6 7;3 4 5]

C1=A+B           %求和
C2=A-B           %求差
C3=5*A            %数乘
C4= A.*B          %点乘
C5= 2.^A          %指数
C6= A.^2          %乘方
C7= A./B          %右除，A 的各元素/ B 对应的各元素
C8= A.\B          %左除，B 的各元素/ A 对应的各元素
```

命令窗口中的输出结果如下所示：

```
A =
     2     1     3
     0     1     2
     1     1     3

B =
    11    20    30
     5     6     7
     3     4     5
```

C1 =

13	21	33
5	7	9
4	5	8

C2 =

-9	-19	-27
-5	-5	-5
-2	-3	-2

C3 =

10	5	15
0	5	10
5	5	15

C4 =

22	20	90
0	6	14
3	4	15

C5 =

4	2	8
1	2	4
2	2	8

C6 =

4	1	9
0	1	4
1	1	9

C7 =

0.1818	0.0500	0.1000
0	0.1667	0.2857
0.3333	0.2500	0.6000

C8 =

5.5000	20.0000	10.0000
Inf	6.0000	3.5000
3.0000	4.0000	1.6667

下面列出一些常用的基本运算。

- $A=s$: 把标量 s 赋值给 A 的每一个元素。
- $s+A$: 把标量 s 分别与 A 中元素相加。
- $s-A$: 把标量 s 分别与 A 中元素相减。
- $\exp(A)$: 以 e 为底, 计算 A 的各元素指数。
- $\log(A)$: 求 A 的各元素对数。
- $\text{sqrt}(A)$: 求 A 的各元素平方根。

需要说明的是, 必须是同维数组才可以进行数组间运算, 且运算结果与它们也是同维的。

2. 关系运算

MATLAB 中提供的关系运算符: $>$ (大于)、 $<$ (小于)、 $>=$ (大于等于)、 $<=$ (小于等于)、 $==$ (等于)、 \sim (不等于), 运算结果为逻辑类型数据。

【例 2-37】数组应用关系运算符。

在命令窗口中输入如下语句:

```
A=[8 6 9;9 4 6;1 6 10;1 2 3]
B=[5 3 12;11 6 6;3 2 0;4 5 6]
C1=A>B
C2=A<=B
C3=A==B
C4=A~B
```

命令窗口中的输出结果如下所示:

```
A =
     8     6     9
     9     4     6
     1     6    10
     1     2     3

B =
     5     3    12
    11     6     6
     3     2     0
     4     5     6

C1 =
     1     1     0
     0     0     0
     0     1     1
     0     0     0

C2 =
     0     0     1
     1     1     1
     1     0     0
```

```
1    1    1
```

```
C3 =
```

```
0    0    0
0    0    1
0    0    0
0    0    0
```

```
C4 =
```

```
1    1    1
1    1    0
1    1    1
1    1    1
```

3. 逻辑运算

MATLAB 中提供的逻辑运算符： $\&$ （与）、 \sim （非）、 \mid （或），运算结果为逻辑类型数据。

【例 2-38】数组应用逻辑运算符。

在命令窗口中输入如下语句：

```
A=[1 0 0;0 1 0;0 0 1]
```

```
B=[1.2 2.3 3.4;4.5 5.6 6.7;7.8 8.9 9.0]
```

```
C1=A&B
```

```
C2=A|B
```

```
C3=~B
```

命令窗口中的输出结果如下所示：

```
A =
```

```
1    0    0
0    1    0
0    0    1
```

```
B =
```

```
1.2000    2.3000    3.4000
4.5000    5.6000    6.7000
7.8000    8.9000    9.0000
```

```
C1 =
```

```
1    0    0
0    1    0
0    0    1
```

```
C2 =
```

```
1    1    1
```

```
1      1      1
1      1      1

C3 =

0      0      0
0      0      0
0      0      0
```

需要说明的是，上述运算等效于先将 **B** 变换为逻辑类型数据，再与 **A** 进行逻辑运算。

2.2.3 数组函数

表 2-4 提供了 MATLAB 常用的数组函数，介绍了函数名称和功能。

表 2-4 常用数组函数

函数名称	功能介绍	函数名称	功能介绍
sin	正弦（弧度）	asech	反双曲正割
sind	正弦（角度）	acsch	反双曲余割
cos	余弦（弧度）	ceil	向正无穷大方向取整
cosd	余弦（角度）	floor	向负无穷大方向取整
tan	正切（弧度）	fix	向零方向取整
tand	正切（角度）	round	四舍五入为整数
cot	余切（弧度）	sign	符号函数
cotd	余切（角度）	abs	取绝对值
sec	正割（弧度）	angle	获得复数相角
secd	正割（角度）	imag	获得复数虚部
csc	余割（弧度）	real	获得复数实部
cscd	余割（角度）	conj	获得共轭复数
asin	反正弦（弧度）	log10	获得常用对数
asind	反正弦（角度）	log	获得自然对数
acos	反余弦（弧度）	exp	指数
acosd	反余弦（角度）	sqrt	平方根
atan	反正切（弧度）	rem	除运算余数
atand	反正切（角度）	beta	Beta 函数
acot	反余切（弧度）	gamma	Gamma 函数
acotd	反余切（角度）	rat	近似到有理数
sinh	双曲正弦	erf	误差函数
cosh	双曲余弦	ellipj	Jacobi 椭圆函数
tanh	双曲正切	bessel	第一、二类贝塞尔函数
coth	双曲余切	Cross	向量 x 和 y 的叉积
sech	双曲正割	Erfinv	误差逆函数
csch	双曲余割	Ellipke	第一、二类全椭圆积分
asinh	反双曲正弦	cart2sph	变换笛卡儿坐标为球坐标
acosh	反双曲余弦	cart2pol	变换笛卡儿坐标为极坐标
atanh	反双曲正切	pol2cart	变换极坐标为笛卡儿坐标
acoth	反双曲余切		

【例 2-39】分别画出 $\sin(x)$ 、 $\tan(x)$ 、 $\cos(x)$ 、 $\text{atan}(x)$ 的曲线。

在命令窗口中输入如下语句：

```
x=1:pi/4:5;
y=sin(x);
subplot(2,2,1)
plot(x,y)
title('sin 函数')
subplot(2,2,2)
fplot('tan(x)',4*pi*[-1,1,-1,1])
title('tan 函数')
subplot(2,2,3)
fplot('cos(x)',[-1,1,-2*pi,2*pi])
title('cos 函数')
subplot(2,2,4)
fplot('atan(x)',2*pi*[-1,1,-1,1])
title('atan 函数')
```

图形窗口中的输出结果如图 2-4 所示。

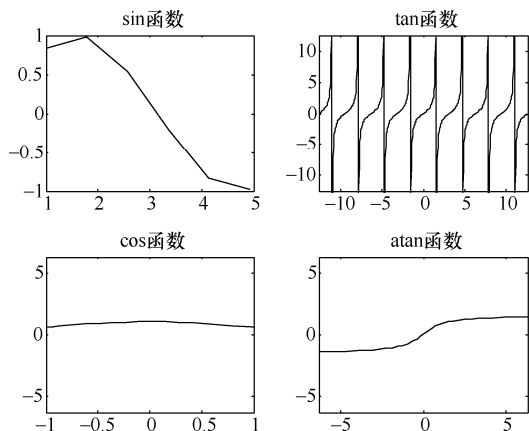


图 2-4 函数 $\cos(x)$ 、 $\sec(x)$ 、 $\text{acos}(x)$ 和 $\text{csch}(x)$ 的图像

`subplot` 函数和 `title` 函数将在第 4 章中详细介绍。

2.3 矩阵及其函数

MATLAB 是基于矩阵运算的一个软件，它的基本形式是二维矩阵，从单个数据到一组数据都可以用二维矩阵来存储。具体来说，MATLAB 使用 1×1 矩阵表示一个数据；使用 $1 \times n$ 矩阵表示一组数据（ n 为数据长度）。通常将 1×1 的矩阵称为标量；将 $1 \times n$ 的矩阵称为向量；将 0×0 的矩阵称为空矩阵，使用中括号对 “[]” 表示。

MATLAB 提供了大量函数用于矩阵运算，方便了用户的使用。

2.3.1 矩阵的建立和操作

1. 矩阵的建立

建立矩阵可以使用直接输入法和 MATLAB 函数法两种方法。

(1) 直接输入法

在 MATLAB 中直接创建矩阵有以下要求：

- 矩阵元素必须在中括号对 “[]” 内。
- 矩阵的同行元素之间用空格或逗号 “,” 隔开。
- 矩阵的行与行之间用分号 “;” 或回车符隔开。
- 矩阵的元素可以是数值、变量、表达式或者函数。
- 矩阵的尺寸不必提前定义。

输入 $1 \times m$ 的矩阵，有以下两种方式：

- `row = [E1, E2, ..., Em]`
- `row = [E1 E2 ... Em]`

输入 $m \times n$ 的矩阵，有以下两种方式：

- `row = [E11, E12, ..., E1n; E21, E22, ..., E2n; ...; En1, En2, ..., Enn]`
- `row = [E11 E12 ... E1n; E21 E22 ... E2n; ...; En1 En2 ... Enn]`

【例 2-40】使用直接输入法构造 1×4 的矩阵。

在命令窗口中输入如下语句：

```
a=[1 2 3 4]
a1=[1,2,3,4]
```

命令窗口中的输出结果如下所示：

```
a =
    1     2     3     4

a1 =
    1     2     3     4
```

本例需要说明的是，无论矩阵的元素之间使用逗号隔开还是空格隔开，两种情况的输出结果是相同的。

【例 2-41】使用直接输入法构造矩阵 4×4 矩阵。

在命令窗口中输入如下语句：

```
A=[1 2 3 4;5 6 7 8;1 3 5 7;2 4 6 8]
A1=[1 2 3 4
5 6 7 8
1 3 5 7
2 4 6 8]
```

命令窗口中的输出结果如下所示：

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

```

1    3    5    7
2    4    6    8

```

A1 =

```

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

```

本例需要说明的是，无论矩阵的行之间使用分号隔开还是空格隔开，两种情况的输出结果是相同的。

(2) MATLAB 函数法

表 2-5 提供了常用的特殊矩阵函数。

表 2-5 常用的特殊矩阵函数

函数名称	功能介绍	基本调用格式	
ones	矩阵元素全为 1	A=ones(n)	产生 $n \times n$ 的全 1 矩阵
		A=ones(m×n)	产生 $m \times n$ 的全 1 矩阵
zeros	矩阵元素全为 0	A=zeros(n)	产生 $n \times n$ 的全 0 矩阵
		A=zeros(m,n)	产生 $m \times n$ 的全 0 矩阵
eye	主对角线元素为 1，其他位置元素为 0 的单位矩阵	A=eye(n)	产生 $n \times n$ 的单位矩阵
		A=eye(m,n)	产生 $m \times n$ 的单位矩阵
diag	将向量转化为对角矩阵 或得到矩阵的对角元素	X=diag(v,k)	将向量 v 转换为一个对角矩阵
		X=diag(v)	将向量 v 转换为一个主对角矩阵
		v=diag(X,k)	得到矩阵 X 的对角元素
		v=diag(X)	得到矩阵 X 的主对角元素
magic	产生魔方矩阵	A=magic(n)	产生 $n \times n$ 的魔方矩阵
rand	产生 0~1 均匀分布的随机数	A=rand(n)	产生 $n \times n$ 的 0~1 均匀分布的随机数
		A=rand(m,n)	产生 $m \times n$ 的 0~1 均匀分布的随机数
randn	产生均值为 0、方差为 1 的标准高斯分布的随机数	A=randn(n)	产生 $n \times n$ 的标准高斯分布的随机数
		A=randn(m,n)	产生 $m \times n$ 的标准高斯分布的随机数
randperm	产生整数 1~n 的随机排列	A=randperm(n)	产生整数 1~n 的随机排列
compan	产生多项式的伴随矩阵	A=compan(n)	产生多项式 n 的伴随矩阵
hankel	产生 Hankel 矩阵	A=hankel(n)	产生 n 维的 Hankel 矩阵
vander	产生范德蒙矩阵	Y=vander(A)	由矩阵 A 产生范德蒙矩阵
pascal	产生 Pascal 矩阵	A=pascal(n)	产生 n 维的 Pascal 矩阵

【例 2-42】产生 4×4 的魔方矩阵，在命令窗口中输入如下语句：

```
A=magic(4)
```

命令窗口中的输出结果如下所示：

A =

```

16    2    3   13
 5   11   10    8
 9    7    6   12

```



```
4    14    15    1
```

【例 2-43】产生 4×4 单位矩阵，在命令窗口中输入如下语句：

```
A=eye(4)
```

命令窗口中的输出结果如下所示：

```
A =
```

```
1    0    0    0
0    1    0    0
0    0    1    0
0    0    0    1
```

【例 2-44】产生 4×4 的 0~1 均匀分布随机数。

在命令窗口中输入如下语句：

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

命令窗口中的输出结果如下所示：

```
A =
```

```
0.8147    0.6324    0.9575    0.9572
0.9058    0.0975    0.9649    0.4854
0.1270    0.2785    0.1576    0.8003
0.9134    0.5469    0.9706    0.1419
```

本例需要说明的是，rand 函数以机器时间为标准，每次运行结果不同。

【例 2-45】产生 4×4 的 Pascal 矩阵。

在命令窗口中输入如下语句：

```
A= pascal (4)
```

命令窗口中的输出结果如下所示：

```
A =
```

```
1    1    1    1
1    2    3    4
1    3    6   10
1    4   10   20
```

2. 矩阵的操作

(1) 矩阵的合并

矩阵的合并是指将两个或两个以上的矩阵合并成一个新的矩阵。中括号对 “[]” 不仅可以作为矩阵构造符，而且可以作为矩阵合并操作符。

矩阵有两种合并方式，即水平方向合并矩阵和竖直方向合并矩阵，具体用法如下：

- $C=[A\ B]$ ，在水平方向合并矩阵 A 和 B 。
- $C=[A;B]$ ，在竖直方向合并矩阵 A 和 B 。

【例 2-46】在水平方向合并矩阵 A 和 B 。

在命令窗口中输入如下语句：

```
A=[1 2 3;4 5 6;7 8 9]    %生成 3×3 的矩阵
B=[1.2 2.3;3.8 4; 2.9 6]  %生成 3×2 的矩阵
C=[A,B]                  %水平方向合并矩阵 A 和 B
```

```
A1=[1 2 3;4 5 6;7 8 9]      %生成 3×3 的矩阵
B1=[1.2 2.3 3.8;4 2.9 6]      %生成 2×3 的矩阵
C1=[A1,B1]                    %水平方向合并矩阵 A 和 B
```

命令窗口中的输出结果如下所示:

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

B =
     1.2000     2.3000
     3.8000     4.0000
     2.9000     6.0000

C =
     1.0000     2.0000     3.0000     1.2000     2.3000
     4.0000     5.0000     6.0000     3.8000     4.0000
     7.0000     8.0000     9.0000     2.9000     6.0000

A1 =
     1     2     3
     4     5     6
     7     8     9

B1 =
     1.2000     2.3000     3.8000
     4.0000     2.9000     6.0000
```

```
??? Error using ==> horzcat
```

```
CAT arguments dimensions are not consistent.
```

从上面的结果不难看出: 在水平方向合并矩阵 **A** 和 **B** 的时候, 要求矩阵 **A** 和 **B** 的行数相同。

【例 2-47】在竖直方向合并矩阵 **A** 和 **B**。

在命令窗口中输入如下语句:

```
A=[2 3.2 1;4 5 2;5.6 5 3;1.5 3 4]      %生成 4×3 的矩阵
B=[pi 9.89 5.5]                          %生成 1×3 的矩阵
C=[A;B]                                    %竖直方向合并矩阵 A 和 B
A1=[2 3.2 1;4 5 2;5.6 5 3;1.5 3 4]      %生成 4×3 的矩阵
B1=[pi 9.89 4 6.3]                       %生成 1×4 的矩阵
C1=[A2;B2]                                %竖直方向合并矩阵 A 和 B
```

命令窗口中的输出结果如下所示:

```
A =
    2.0000    3.2000    1.0000
    4.0000    5.0000    2.0000
    5.6000    5.0000    3.0000
    1.5000    3.0000    4.0000

B =
    3.1416    9.8900    5.5000

C =
    2.0000    3.2000    1.0000
    4.0000    5.0000    2.0000
    5.6000    5.0000    3.0000
    1.5000    3.0000    4.0000
    3.1416    9.8900    5.5000

A1 =
    2.0000    3.2000    1.0000
    4.0000    5.0000    2.0000
    5.6000    5.0000    3.0000
    1.5000    3.0000    4.0000

B1 =
    3.1416    9.8900    4.0000    6.3000
```

```
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

从上面的结果不难看出：在竖直方向合并矩阵 **A** 和 **B** 的时候，要求矩阵 **A** 和 **B** 的列数相同。合并矩阵可以使用矩阵合并符，还可以使用矩阵合并函数，表 2-6 提供了常用的矩阵合并函数。

表 2-6 常用的矩阵合并函数

函数名称	功能介绍	基本调用格式	
cat	在指定的方向合并矩阵	cat(1,A,B)	竖直方向合并矩阵
		cat(2,A,B)	水平方向合并矩阵
		cat(dim,A,B)	dim 维方向合并矩阵
horzcat	在水平方向合并矩阵	horzcat(A,B)	与[A B]含义相同
vertcat	在竖直方向合并矩阵	vertcat(A,B)	与[A;B]含义相同
repmat	通过复制矩阵来构造新的矩阵	B = repmat(A,m,n)	得到 $m \times n$ 个 A 的大矩阵
blkdiag	用已知矩阵来构造块对角化矩阵	Y = blkdiag(A,B,...)	得到以矩阵 A 、 B …为对角块的矩阵 Y

【例 2-48】在水平方向合并矩阵 **A** 和 **B**。

在命令窗口中输入如下语句：

```
A=[3 5 7 9]
B=rand(4)
C=cat(1,A,B)
```

命令窗口中的输出结果如下所示：

```
A =
     3     5     7     9

B =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419

C =
    3.0000    5.0000    7.0000    9.0000
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
```

【例 2-49】使用 `repmat` 函数合并矩阵。

在命令窗口中输入如下语句：

```
A=ones(2)
B = repmat(A,3,3)
```

命令窗口中的输出结果如下所示：

```
A =
     1     1
     1     1

B =
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
```

【例 2-50】构造矩阵 A 和 B 的对角块矩阵 C 。

在命令窗口中输入如下语句：

```
A=pascal(3);
B= rand(3);
```

```
C = blkdiag(A,B)
```

命令窗口中的输出结果如下所示：

```
C =
```

```

1.0000    1.0000    1.0000         0         0         0
1.0000    2.0000    3.0000         0         0         0
1.0000    3.0000    6.0000         0         0         0
         0         0         0    0.8147    0.9134    0.2785
         0         0         0    0.9058    0.6324    0.5469
         0         0         0    0.1270    0.0975    0.9575

```

(2) 矩阵行、列的删除

要从矩阵中删除某行或某列，将行或者列赋为一个空矩阵即可。

【例 2-51】删除一个 4×4 矩阵的第 2 行。

在命令窗口中输入如下语句：

```
A = rand(4,4)      %生成 3×4 的 0~1 均匀分布的随机数
A(2,:)=[]          %删除矩阵的第 2 行
```

命令窗口中的输出结果如下所示：

```
A =
```

```

0.7577    0.1712    0.0462    0.3171
0.7431    0.7060    0.0971    0.9502
0.3922    0.0318    0.8235    0.0344
0.6555    0.2769    0.6948    0.4387

```

```
A =
```

```

0.7577    0.1712    0.0462    0.3171
0.3922    0.0318    0.8235    0.0344
0.6555    0.2769    0.6948    0.4387

```

【例 2-52】删除一个 4×4 魔方矩阵的第 3 列。

在命令窗口中输入如下语句：

```
A=magic(4)        %生成 4×4 的魔方矩阵
A(:,3)=[]          %删除矩阵的第 3 列
```

命令窗口中的输出结果如下所示：

```
A =
```

```

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

```

```
A =
```

```

16     2    13
 5    11     8

```

```

9    7    12
4    14    1

```

(3) 利用冒号表达式获得子矩阵

- $A(:,j)$: 表示取 A 矩阵的第 j 列全部元素。
- $A(i,:)$: 表示取 A 矩阵的第 i 行全部元素。
- $A(i,j)$: 表示取 A 矩阵的第 i 行、第 j 列元素。
- $A(i:i+m,:)$: 表示取 A 矩阵的第 $i \sim i+m$ 行全部元素。
- $A(:,k:k+m)$: 表示取 A 矩阵的第 $k \sim k+m$ 列全部元素。
- $A(i:i+m,k:k+m)$: 表示取 A 矩阵的第 $i \sim i+m$ 行、第 $k \sim k+m$ 列全部元素。

【例 2-53】通过冒号表达式获得 5×5 pascal 矩阵的子矩阵。

在命令窗口中输入如下语句:

```

A=pascal(5)      %生成 5×5 的 pascal 矩阵
A1=A(:,2)        %取 A 矩阵的第 2 列全部元素
A2=A(3,:)        %取 A 矩阵的第 3 行全部元素
A3=A(1,4)        %取 A 矩阵的第 1 行、第 4 列全部元素
A4=A(1:1+3,:)    %取 A 矩阵的第 1~1+3 行全部元素
A5=A(:,2:1+2)    %取 A 矩阵的第 2~1+2 列全部元素
A6=A(1:2+1,4:3+2) %取 A 矩阵的第 1~2+1 行、第 4~3+2 列全部元素

```

命令窗口中的输出结果如下所示:

```

A =
    1     1     1     1     1
    1     2     3     4     5
    1     3     6    10    15
    1     4    10    20    35
    1     5    15    35    70

```

A1 =

```

1
2
3
4
5

```

A2 =

```

1     3     6    10    15

```

A3 =

```

1

```

A4 =

```

1    1    1    1    1
1    2    3    4    5
1    3    6   10   15
1    4   10   20   35

```

A5 =

```

1    1
2    3
3    6
4   10
5   15

```

A6 =

```

1    1
4    5
10   15

```

3. 矩阵结构的改变

表 2-7 提供了常用的矩阵结构改变函数。

表 2-7 矩阵结构改变函数

函数名称	功能介绍	基本调用格式	
reshape	按照列的顺序重新排列矩阵元素	B = reshape(A,m,n)	把矩阵 A 变为 $m \times n$ 大小
rot90	旋转矩阵 90°	B = rot90(A)	旋转矩阵 90°
		B = rot90(A,k)	旋转矩阵 $k \times 90^\circ$, k 为整数
fliplr	以竖直方向为轴做镜像	B = fliplr(A)	
flipud	以水平方向为轴做镜像	B = flipud(A)	
flipdim	以指定的轴做镜像	B = flipdim(A,dim)	$dim=1$ 以水平方向为轴做镜像, $dim=2$ 以竖直方向为轴做镜像
transpose	矩阵的转秩	B = transpose(A)	相当于 $B=A'$
ctranspose	矩阵的共轭转秩	B = ctranspose(A)	相当于 $B=A'$

【例 2-54】分别对 A 矩阵实现矩阵结构的改变。

在命令窗口中输入如下语句：

```

A = [1 2 3;4 5 6;7 8 9; 10 11 12]    %定义 4×3 的 A 矩阵
B = transpose(A)                      %计算矩阵 A 的转秩
B1 = ctranspose(A)                   %计算矩阵 A 的共轭转秩
B2 = reshape(A,3, 4)                 %将矩阵 A 重新排列成 3×4 的矩阵
B3 = rot90(A)                        %将矩阵旋转 90°
B4 = fliplr(A)                       %将矩阵以竖直方向为轴做镜像

```

命令窗口中的输出结果如下所示：

A =

```
1    2    3
4    5    6
7    8    9
10   11   12

B =
    1     4     7    10
    2     5     8    11
    3     6     9    12

B1 =
    1     4     7    10
    2     5     8    11
    3     6     9    12

B2 =
    1    10     8     6
    4     2    11     9
    7     5     3    12

B3 =
    3     6     9    12
    2     5     8    11
    1     4     7    10

B4 =
    3     2     1
    6     5     4
    9     8     7
   12    11    10
```

2.3.2 矩阵运算

MATLAB 具有强大的矩阵运算能力，支持许多线性代数中的操作。

1. 算术运算

(1) 基本算术运算

MATLAB 的基本算术运算有：+（加）、-（减）、*（乘）、/（右除）、\（左除）、和^（乘方）。需要说明的是，上述运算是在矩阵意义下进行的，单个数据的算术运算是矩阵运算的一个特例，常见的矩阵算术运算如表 2-8 所示。

(2) 点运算

点运算符有.*、./、.\和.^。两矩阵进行点运算是指它们的对应元素进行相关运算，要求两个矩阵的维数相同。

表 2-8 常见矩阵算术运算

书写形式	功能介绍
$s * A$	标量 s 分别与 A 矩阵每个元素的积
$s * \text{inv}(B)$	s 与 B 逆矩阵的积
A^n	方阵 A 的整数乘方
A^p	方阵 A 的非整数乘方
p^A	标量 p 的矩阵乘方，其中 A 为方阵
$A + B$	矩阵相加
$A - B$	矩阵相减
$A * B$	维数相同矩阵的乘积
A / B	A 右除 B
$A \backslash B$	A 左除 B
$\text{expm}(A)$	A 矩阵的指数函数
$\text{logm}(A)$	A 矩阵的对数函数
$\text{sqrln}(A)$	A 矩阵的平方根函数
$\text{funm}(A, 'FN')$	一般矩阵函数

2. 关系运算

MATLAB 为矩阵运算也提供了与数组运算相同的 6 种关系运算符。关系运算符的运算法则如下：

- （1）当两个比较量是标量时，直接比较两数的大小，如果运算关系成立，关系表达式返回 1，否则返回 0。
- （2）当两个比较量是维数相同的矩阵时，比较两个矩阵对应位置上的元素，并且按标量关系运算规则逐个比较，最后给出比较结果，比较结果是一个由 0 或 1 组成的维数与原矩阵相同的矩阵。
- （3）当两个比较量一个是标量一个是矩阵时，将标量与矩阵的每一个元素按标量关系运算规则逐个比较，最后给出比较结果，比较结果是一个由 0 或 1 组成的维数与原矩阵相同的矩阵。

【例 2-55】生成 4×4 的矩阵 A ，找出 A 中大于 0.5 的元素。

在命令窗口中输入如下语句：

```
A=rand(4) % 生成 4×4 的 A 矩阵
find(A>0.5) %找出 A 中大于 0.5 的元素
```

命令窗口中的输出结果如下所示：

```
A =
    0.3816    0.4898    0.7547    0.1626
    0.7655    0.4456    0.2760    0.1190
    0.7952    0.6463    0.6797    0.4984
    0.1869    0.7094    0.6551    0.9597

ans =

     2
     3
     7
     8
     9
    11
```

12

16

从上面的结果不难看出，find 函数返回的是矩阵元素的位置，不是矩阵元素的内容。

3. 逻辑运算

MATLAB 为矩阵运算也提供了与数组运算相同的 3 种逻辑运算符。逻辑运算符的运算法则如下：

- (1) 在逻辑运算中，非零元素为 true 用 1 表示，零元素为 false 用 0 表示。
- (2) 当两个标量 *a* 和 *b* 进行逻辑运算时：
 - 逻辑与 (*a*&*b*) 运算，*a* 和 *b* 全为非 0 时，输出结果是 1，否则是 0。
 - 逻辑或 (*a*|*b*) 运算，*a* 和 *b* 只要有一个为非 0，输出结果就是 1，否则是 0。
 - 逻辑非 (~*a*) 运算，当 *a* 是 0 时，输出结果是 1；当 *a* 为非 0 时，输出结果是 0。
- (3) 当参与逻辑运算的是两个同维矩阵时，对矩阵相同位置上的元素按标量规则逐个进行运算，输出结果是一个由元素 0 和 1 组成的与原矩阵同维的矩阵。
- (4) 当参与逻辑运算的一个是标量一个是矩阵时，在标量与矩阵中的每个元素之间按标量规则逐个进行运算，输出结果是一个由元素 0 和 1 组成的与原矩阵同维的矩阵。

2.3.3 矩阵函数

表 2-9 提供了 MATLAB 常用的矩阵函数。

表 2-9 矩阵函数	
函数名称	功能介绍
norm	计算矩阵或者向量的范数
normest	计算矩阵的 2 阶范数
rank	计算矩阵的秩
det	计算矩阵行列式的值
trace	计算矩阵的迹
null	零空间
orth	正交化空间
rref	约化行阶梯形式
subspace	计算两个矩阵空间的角度

1. 矩阵的范数

MATLAB 使用 norm 函数计算矩阵的范数。

norm 函数的具体用法如下：

- N=norm(A,p)：计算矩阵或向量 *A* 的 *p* 阶范数。

对于矩阵 *A*，求解其 1 阶范数为 N=norm(A,1)，即返回矩阵 *A* 各列元素和中的最大值；求解其 2 阶范数为 N=norm(A,2)，即返回矩阵 *A* 的最大奇异值，此为默认情况，即等价于 N=norm(A)；求解其无穷阶范数为 N=norm(A,inf)，即返回矩阵 *A* 各行元素和中的最大值。

对于向量 *A*，求解其 *p* 阶范数为 N=norm(A,p)，即返回 $N = \left(\sum_{i=1}^{\text{length}(A)} A^p(i) \right)^{\frac{1}{p}}$ ；默认情况为 *p*=2，

即 N=norm(A,2)等效于 N=norm(A)；求解其无穷阶范数为 N=norm(A,p)，即返回向量 *A* 各元素绝对值中的最大值。

【例 2-56】计算矩阵 *A* 的 1 阶、2 阶和无穷阶范数。

在命令窗口中输入如下语句：

```
A=magic(5)
N1=norm(A,1)
N2=norm(A,2)
Ninf=norm(A,inf)
```

命令窗口中的输出结果如下：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

N1 =
    65

N2 =
   65.0000

Ninf =
    65
```

`norm` 函数计算维数比较大的矩阵时，计算时间比较长。对于一个维数比较大并且计算的精度要求不高的矩阵，一个近似的范数值就可以满足要求，因此常常使用 `normest` 函数估计 2 阶范数值。

`normest` 函数的具体用法如下：

- `normest(S)`：估计矩阵 S 的 2 阶范数值，默认允许的相对误差为 $1e-6$ 。
- `normest(S,tol)`：使用 *tol* 作为允许的相对误差。

【例 2-57】计算 5×5 魔方矩阵 A 的 2 阶范数。

在命令窗口中输入如下语句：

```
A=magic(5)
N=normest(A)
```

命令窗口中的输出结果如下所示：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

N =
    65
```

2. 矩阵的秩

矩阵的秩反映了矩阵各行向量和各列向量之间的线性依赖关系，矩阵 A 中线性无关的列向

量个数称为列秩，线性无关的行向量个数称为行秩。求解矩阵秩的方法很多，有些算法稳定，有些算法不稳定，MATLAB 提供了 rank 函数计算矩阵的秩。

rank 函数的具体用法如下：

- rank(A)：用默认允许误差计算矩阵的秩。
- rank(A,tol)：给定允许误差计算矩阵的秩。

【例 2-58】分别计算下列矩阵的秩。

在命令窗口中输入如下语句：

```
A=[12 -2 1;-6 9.8 5;4.5 6 7.1];
a=rank(A)
B=eye(4);
b=rank(B)
C=zeros(3);
c=rank(C)
```

命令窗口中的输出结果如下所示：

```
a =
    3

b =
    4

c =
    0
```

3. 矩阵的行列式

MATLAB 提供了 det 函数求解矩阵的行列式。

【例 2-59】计算下列矩阵的行列式。

在命令窗口中输入如下语句：

```
A=[2.4 5 -5;1.6 5 4;7 0 8.5]
B=rand(4)
Y=det(A)
Z=det(B)
```

命令窗口中的输出结果如下所示：

```
A =
    2.4000    5.0000   -5.0000
    1.6000    5.0000    4.0000
    7.0000         0    8.5000

B =
    0.3404    0.2551    0.9593    0.2575
    0.5853    0.5060    0.5472    0.8407
    0.2238    0.6991    0.1386    0.2543
    0.7513    0.8909    0.1493    0.8143
```

Y =

349

Z =

-0.0802

由常数矩阵的行列式可以判断该矩阵是否为奇异矩阵，即行列式为 0 的矩阵为奇异矩阵。

4. 矩阵的逆

矩阵 A 的逆矩阵如果是 B ，有 $B*A=A*B=I$ 成立，表示一个矩阵和它的逆矩阵左乘或右乘的结果都是单位矩阵。矩阵的逆是线性代数中的重要概念，对于线性方程组 $A*X=b$ ，计算矩阵的逆是求解方程解的一个重要途径。

MATLAB 提供 `inv` 函数计算矩阵的逆，它的具体用法如下：

- `N=inv(A)`：计算矩阵 A 的逆。

【例 2-60】计算下列矩阵的逆。

在命令窗口中输入如下语句：

```
A=[3.2 5 -3 1;2 0 5 4.5;3 7 -2 8.5;3.2 6.24 5 8]
```

```
B=magic(5)
```

```
N=inv(A)
```

```
N1=inv(B)
```

命令窗口中的输出结果如下所示：

A =

3.2000	5.0000	-3.0000	1.0000
2.0000	0	5.0000	4.5000
3.0000	7.0000	-2.0000	8.5000
3.2000	6.2400	5.0000	8.0000

B =

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

N =

0.3232	0.3838	-0.0454	-0.2081
0.0076	-0.2851	-0.0891	0.2541
-0.0173	-0.0354	-0.1388	0.1695
-0.1244	0.0910	0.1744	-0.0959

N1 =

-0.0049	0.0512	-0.0354	0.0012	0.0034
---------	--------	---------	--------	--------

0.0431	-0.0373	-0.0046	0.0127	0.0015
-0.0303	0.0031	0.0031	0.0031	0.0364
0.0047	-0.0065	0.0108	0.0435	-0.0370
0.0028	0.0050	0.0415	-0.0450	0.0111

5. 矩阵的迹

矩阵的迹定义为矩阵对角线元素之和。

MATLAB 提供 `trace` 函数计算矩阵的迹，它的具体用法如下：

- `Y=trace(A)`：计算矩阵 **A** 的迹。

【例 2-61】计算矩阵 **A** 的迹。

在命令窗口中输入如下语句：

```
A=rand(4)
```

```
Y=trace(A)
```

命令窗口中的输出结果如下所示：

```
A =
    0.2435    0.2511    0.8308    0.2858
    0.9293    0.6160    0.5853    0.7572
    0.3500    0.4733    0.5497    0.7537
    0.1966    0.3517    0.9172    0.3804
```

```
Y =
    1.7897
```

6. 矩阵的点乘积

两矩阵的点乘积是两矩阵对应元素的乘积运算，要求两矩阵具有相同的维数。MATLAB 提供 `dot` 函数计算矩阵的点乘积，它的具体用法如下：

- `Y=dot(A,B)`：计算同维矩阵 **A** 和 **B** 的点乘积。

【例 2-62】计算 4×4 矩阵 **A** 和 **B** 的点乘积。

在命令窗口中输入如下语句：

```
A=rand(4)
```

```
B=[4 2 3 1;0.5 6 4 2;2 1 4 5;0.1 0.6 4 8]
```

```
Y=dot(A,B)
```

命令窗口中的输出结果如下所示：

```
A =
    0.4218    0.6557    0.6787    0.6555
    0.9157    0.0357    0.7577    0.1712
    0.7922    0.8491    0.7431    0.7060
    0.9595    0.9340    0.3922    0.0318
```

```
B =
    4.0000    2.0000    3.0000    1.0000
    0.5000    6.0000    4.0000    2.0000
    2.0000    1.0000    4.0000    5.0000
```

```
0.1000    0.6000    4.0000    8.0000

Y =
    3.8253    2.9353    9.6086    4.7827
```

7. 矩阵的特征值

MATLAB 提供 `eig` 函数计算矩阵的特征值，它的具体用法如下：

- `Y=eig(A)`: 计算矩阵 **A** 的特征值。

【例 2-63】计算 3×3 矩阵 **A** 和 **B** 的点乘积。

在命令窗口中输入如下语句：

```
A=[1 2 3;4 5 6;7 8 9]
Y=eig(A)
B=rand(4)
Z=eig(B)
```

命令窗口中的输出结果如下所示：

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

Y =
    16.1168
    -1.1168
    -0.0000

B =
    0.5678    0.7792    0.4694    0.7943
    0.0759    0.9340    0.0119    0.3112
    0.0540    0.1299    0.3371    0.5285
    0.5308    0.5688    0.1622    0.1656

Z =
     1.5507
    -0.2476
     0.0640
     0.6374
```

8. 矩阵的分解

矩阵的分解是根据一定原理将一个矩阵分解成几个矩阵的乘积，常见的矩阵分解有 Cholesky 分解、LU 分解、QR 分解、奇异值分解和 Schur 分解等。MATLAB 中线性方程组求解主要基于 3 种矩阵分解，即对称正定矩阵的 Cholesky 分解、矩阵的 LU 分解和 QR 分解。这 3 种分解分别通过 `chol` 函数、`lu` 函数和 `qr` 函数实现，并且都使用了三角矩阵。三角矩阵中对角线上方或下方元素均为 0。

(1) 对称正定矩阵的 Cholesky 分解

只有正定矩阵可以进行 Cholesky 分解, Cholesky 分解是用上三角矩阵与下三角矩阵乘积的形式表示一个对称矩阵。设上三角矩阵为 R , 下三角矩阵是 R 的转置 R' , 则对 A 矩阵的 Cholesky 分解可以表示为 $A=R \times R'$ 。

MATLAB 提供了 chol 函数进行 Cholesky 分解, 它的具体用法如下:

- $R=\text{chol}(A)$: 对矩阵 A 进行 Cholesky 分解。

【例 2-64】对 5 阶的 pascal 矩阵进行 Cholesky 分解。

在命令窗口中输入如下语句:

```
A=pascal(5)
```

```
R=chol(A)
```

命令窗口中的输出结果如下所示:

```
A =
```

```

1    1    1    1    1
1    2    3    4    5
1    3    6   10   15
1    4   10   20   35
1    5   15   35   70
```

```
R =
```

```

1    1    1    1    1
0    1    2    3    4
0    0    1    3    6
0    0    0    1    4
0    0    0    0    1
```

(2) 矩阵的 LU 分解

LU 分解就是将任意一个矩阵 A 分解为一个下三角矩阵 L 和一个上三角矩阵 U 的乘积, LU 分解可以表示为 $A=LU$ 。

MATLAB 提供了 lu 函数进行 LU 分解, 它的具体用法如下:

- $[L,U]=\text{lu}(X)$: L 是对角线元素为 1 的下三角矩阵, U 是上三角矩阵。

【例 2-65】对 4 阶随机矩阵进行 LU 分解。

在命令窗口中输入如下语句:

```
A=rand(4)
```

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

命令窗口中的输出结果如下所示:

```
A =
```

```

0.7094    0.6551    0.9597    0.7513
0.7547    0.1626    0.3404    0.2551
0.2760    0.1190    0.5853    0.5060
0.6797    0.4984    0.2238    0.6991
```


L =

```
0.9399    1.0000         0         0
1.0000         0         0         0
0.3657    0.1185   -0.7249    1.0000
0.9006    0.7007    1.0000         0
```

U =

```
0.7547    0.1626    0.3404    0.2551
         0    0.5023    0.6398    0.5115
         0         0   -0.5310    0.1109
         0         0         0    0.4325
```

(3) 矩阵的 QR 分解

QR 分解又称为正交分解,是将任意一个矩阵 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积,QR 分解可以表示为 $A=QR$ 。

MATLAB 提供了 `qr` 函数进行 QR 分解,它的具体用法如下:

- $[Q,R]=qr(A)$: Q 是正交矩阵, R 是上三角矩阵。

【例 2-66】对 4×4 的魔方矩阵进行 QR 分解。

在命令窗口中输入如下语句:

```
A=magic(4)
```

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

命令窗口中的输出结果如下所示:

A =

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

Q =

```
0.8230   -0.4186    0.3123    0.2236
0.2572    0.5155   -0.4671    0.6708
0.4629    0.1305   -0.5645   -0.6708
0.2057    0.7363    0.6046   -0.2236
```

R =

```
19.4422   10.5955   10.9041   18.5164
         0   16.0541   15.7259    0.9848
         0         0    1.9486   -5.8458
         0         0         0    0.0000
```

(4) 奇异值分解

奇异值分解是将矩阵 A 分解为 $A=U*S*V'$, 其中 U 和 V 是正交矩阵, S 是一个对角矩阵。

MATLAB 提供了 `svd` 函数进行奇异值分解，它的具体用法如下：

- `s=svd(A)`：返回由矩阵 A 的奇异值组成的列向量。
- `[U,S,V]=svd(A)`：把矩阵 A 分解为 3 个矩阵的乘积，可以表示为 $A=U*S*V'$ ，其中 U 和 V 是正交矩阵， S 是对角矩阵，并且 S 矩阵的对角元素值为矩阵 A 的奇异值。

【例 2-67】分别对以下矩阵进行奇异值分解。

在命令窗口中输入如下语句：

```
A=magic(4)
s=svd(A)
B=rand(4)
s1=svd(B)
```

命令窗口中的输出结果如下所示：

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

s =
    34.0000
    17.8885
     4.4721
     0.0000

B =
    0.6020    0.7482    0.9133    0.9961
    0.2630    0.4505    0.1524    0.0782
    0.6541    0.0838    0.8258    0.4427
    0.6892    0.2290    0.5383    0.1067

s1 =
     2.1551
     0.6041
     0.4258
     0.0290
```

【例 2-68】分别对以下矩阵进行奇异值分解。

在命令窗口中输入如下语句：

```
A=magic(4)
[U S V]=svd(A)
B=rand(4)
[U1 S1 V1]=svd(B)
```

命令窗口中的输出结果如下所示：

A =

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

U =

-0.5000	0.6708	0.5000	-0.2236
-0.5000	-0.2236	-0.5000	-0.6708
-0.5000	0.2236	-0.5000	0.6708
-0.5000	-0.6708	0.5000	0.2236

S =

34.0000	0	0	0
0	17.8885	0	0
0	0	4.4721	0
0	0	0	0.0000

V =

-0.5000	0.5000	0.6708	0.2236
-0.5000	-0.5000	-0.2236	0.6708
-0.5000	-0.5000	0.2236	-0.6708
-0.5000	0.5000	-0.6708	-0.2236

B =

0.9619	0.8687	0.8001	0.2638
0.0046	0.0844	0.4314	0.1455
0.7749	0.3998	0.9106	0.1361
0.8173	0.2599	0.1818	0.8693

U1 =

-0.6853	-0.2371	-0.5665	-0.3915
-0.1427	-0.1853	0.7096	-0.6647
-0.5484	-0.3837	0.3848	0.6355
-0.4575	0.8730	0.1659	0.0319

S1 =

2.2131	0	0	0
0	0.8061	0	0

```

0      0      0.3796      0
0      0      0      0.2117

```

V1 =

```

-0.6591    0.2323   -0.2841    0.6564
-0.4272   -0.1838   -0.6197   -0.6322
-0.5388   -0.5710    0.6150   -0.0728
-0.3045    0.7656    0.3962   -0.4052

```

(5) 方阵的 schur 分解

schur 分解只能对方阵进行分解, 是将方阵 A 分解为 $A=U*T*U'$, 其中 U 是一个正交矩阵, T 是一个上三角矩阵。

MATLAB 提供了 schur 函数进行舒尔分解, 它的具体用法如下:

- $T = \text{schur}(A)$: 返回上三角矩阵 T 。
- $[U, T] = \text{schur}(A)$: 返回正交矩阵 U 和上三角角矩阵 T 。

【例 2-69】对 4×4 的随机矩阵进行 schur 分解。

在命令窗口中输入如下语句:

```

A=rand(4)
T = schur(A)
[U,T] = schur(A)

```

命令窗口中的输出结果如下所示:

```

A =
    0.5797    0.6221    0.0760    0.2400
    0.5499    0.3510    0.2399    0.4173
    0.1450    0.5132    0.1233    0.0497
    0.8530    0.4018    0.1839    0.9027

T =
    1.6853    0.1894    0.4736    0.1632
         0   -0.2046    0.1016    0.4129
         0         0    0.1700    0.1057
         0         0         0    0.3060

U =
   -0.4315   -0.4799   -0.7620    0.0538
   -0.4511    0.4902   -0.1054   -0.7383
   -0.2122   -0.7049    0.5348   -0.4147
   -0.7518    0.1803    0.3496    0.5291

T =
    1.6853    0.1894    0.4736    0.1632

```

0	-0.2046	0.1016	0.4129
0	0	0.1700	0.1057
0	0	0	0.3060

2.3.4 稀疏矩阵及其运算

在实际问题中常常遇到含有很多元素为 0 的大规模矩阵，它们可以称为稀疏矩阵。对于一个 n 阶矩阵，需要 n^2 的存储空间，当 n 很大时，进行矩阵运算会占用大量的内存空间和运算时间，降低执行效率。MATLAB 支持稀疏矩阵及其操作，只存储和处理矩阵的非零元素，节省了内存空间和运算时间，此时计算的复杂性和代价只取决于稀疏矩阵的非零元素个数。对于低密度（矩阵的密度是指矩阵中非零元素的个数除以矩阵中总的元素个数）的矩阵，采用稀疏矩阵方式存储是一种很好的选择。也就是说，稀疏矩阵提供了一种只针对元素大多数为 0 的矩阵存储和处理方式。

用户可以创建双精度类型、复数类型和逻辑类型的稀疏矩阵，所有 MATLAB 数学函数、逻辑函数和引用操作都可以应用在稀疏矩阵上。

1. 稀疏矩阵的存储方式

MATLAB 通过三个矩阵存储稀疏矩阵（如 $m \times n$ 矩阵）的信息：

- 第一个矩阵存储稀疏矩阵中非零元素的值（ p 个）。它是浮点类型矩阵。
- 第二个矩阵存储这些非零元素的行索引。它是整型矩阵。
- 第三个矩阵存储原矩阵中每一列的第一个非零元素在前两个矩阵中的位置，以及最后一列的最后一个非零元素在前两个矩阵中的位置。它是整型矩阵。

比较普通矩阵和稀疏矩阵的字节单元，对于一个 $m \times n$ 的矩阵，要存储 p 个浮点数和 $p+n+1$ 个整数，用稀疏矩阵来存储需要 $8 \times p + 4 \times (p+n+1)$ 字节的单元，而用普通矩阵需要 $8 \times m \times n$ 字节的单元。很明显用稀疏矩阵来存储节约空间，使用的运算时间较少。

2. 稀疏矩阵的创建

(1) 满矩阵存储方式与稀疏矩阵存储方式的转化

`sparse` 函数将满矩阵转化为稀疏矩阵，其具体用法如下：

- $S = \text{sparse}(A)$ ：将满矩阵 A 转化为稀疏矩阵 S 。

【例 2-70】将魔方矩阵 A 转化为稀疏矩阵 S 。

在命令窗口中输入如下语句：

```
A=magic(4)
```

```
S = sparse(A)
```

命令窗口中的输出结果如下所示：

```
A =
```

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

```
S =
```

```
(1,1)    16
```

```

(2,1)      5
(3,1)      9
(4,1)      4
(1,2)      2
(2,2)     11
(3,2)      7
(4,2)     14
(1,3)      3
(2,3)     10
(3,3)      6
(4,3)     15
(1,4)     13
(2,4)      8
(3,4)     12
(4,4)      1

```

full 函数将稀疏矩阵转换为满矩阵，它的具体用法如下：

- $A = \text{full}(S)$ ：将稀疏矩阵 S 转化为满矩阵 A 。

【例 2-71】将上述稀疏矩阵 S 转化为满矩阵 A 。

在命令窗口中输入如下语句：

```
A = full(S)
```

命令窗口中的输出结果如下所示：

```

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

```

(2) 用 sparse 函数直接创建稀疏矩阵

sparse 函数的具体用法如下：

- $\text{sparse}(m,n)$ ：生成所有元素都是 0 的 $m \times n$ 稀疏矩阵。
- $S = \text{sparse}(i,j,s,m,n)$ ：其中 i 、 j 分别是稀疏矩阵非零元素的行、列下标， s 为相应的非零元素的值， m 和 n 分别是矩阵的行数和列数。

【例 2-72】使用 sparse 函数创建稀疏矩阵。

在命令窗口中输入如下语句：

```
S=sparse([3,2,3,2],[5,1,4,3],[8,6,7,9],3,5)
```

命令窗口中的输出结果如下所示：

```

S =
    (2,1)      6
    (2,3)      9
    (3,4)      7
    (3,5)      8

```

(3) 稀疏带状矩阵的创建

稀疏带状矩阵可以用 `spdiags` 函数创建，它的具体用法如下：

- `S=spdiags(B,d,m,n)`： m 和 n 分别是矩阵的行数和列数， d 是长度为 p 的整数向量，它指定矩阵 S 的对角线位置。 B 是全元素矩阵，用来给定 S 对角线位置上的元素，行数为 $\min(m,n)$ ，列数为 p 。

(4) 单位稀疏矩阵的产生

单位稀疏矩阵是一种具有稀疏特征的矩阵，对角线元素为 1，其他元素都为 0。`eye` 函数可以产生一个满秩的单位矩阵，MATLAB 还提供了 `speye` 函数产生稀疏存储方式的单位矩阵。

`speye` 函数的具体用法如下：

- `Y=speye(m,n)`：返回一个 $m \times n$ 的单位稀疏矩阵。

表 2-10 提供了部分用于创建特殊稀疏矩阵的函数。

表 2-10 创建特殊稀疏矩阵的函数			
函数名称	功能介绍	基本调用格式	
speye	创建单位稀疏矩阵	$S = \text{speye}(m,n)$	创建 $m \times n$ 单位稀疏矩阵
		$S = \text{speye}(n)$	创建 $n \times n$ 单位稀疏矩阵
spones	创建非零元素为 1 的稀疏矩阵	$R = \text{spones}(S)$	把矩阵 S 的非零元素的值改为 1
		$R = \text{sprand}(S)$	把矩阵 S 的非零元素的值改为均匀分布的随机数
sprand	创建非零元素为均匀分布的随机数的稀疏矩阵	$R = \text{sprand}(m,n,\text{density})$	创建非零元素密度为 density 的 $m \times n$ 大小的均匀分布的随机数
sprandn	创建非零元素为高斯分布的随机数的稀疏矩阵	$R = \text{sprandn}(S)$	把矩阵 S 的非零元素的值改为高斯分布的随机数
		$R = \text{sprandn}(m,n,\text{density})$	创建非零元素密度为 density 的 $m \times n$ 大小的高斯分布的随机数
sprandsym	创建非零元素为高斯分布的随机数的对称稀疏矩阵	$R = \text{sprandsym}(S)$	返回对称随机稀疏矩阵，其下三角和主对角结构与 S 相同
		$R = \text{sprandsym}(n,\text{density})$	返回 $n \times n$ 的对称随机稀疏矩阵，其非零元素密度为 density
spdiags	创建对角稀疏矩阵	$A = \text{spdiags}(B,d,m,n)$	把 B 中的值放在 d 中指定的对角线上，创建一个 $m \times n$ 的稀疏矩阵
spalloc	为稀疏矩阵分配空间	$S = \text{spalloc}(m,n,\text{nzmax})$	相当于 <code>sparse([],[],[],m,n,nzmax)</code>

3. 稀疏矩阵函数

MATLAB 提供了常用的稀疏矩阵函数，我们可以通过这些函数得到稀疏矩阵的定量信息和图形化信息，常用稀疏矩阵函数见表 2-11。

表 2-11 常用稀疏矩阵函数	
函数名称	功能介绍
spy	通过图像来显示稀疏矩阵非零元素分布
spalloc	为稀疏矩阵分配内存空间
nnz	统计矩阵中非零元素的个数
nzmax	存储非零元素的空间长度
issparse	判断是否为稀疏矩阵
speye	单位稀疏矩阵
spaument	建立最小二乘增广矩阵
spconvert	从外部载入稀疏矩阵
spparms	设置稀疏矩阵程序参数
sprand	创建均匀分布随机的稀疏矩阵
sprandn	创建高斯分布随机的稀疏矩阵
sprandsym	创建对称随机的稀疏矩阵

4. 稀疏矩阵的运算规则

MATLAB 系统中的各种命令都可以用于稀疏矩阵运算，但稀疏矩阵参加运算时，所得结果需要遵循以下规则：

- 将标量或者定长向量变换到矩阵函数，给出满矩阵结果相应的函数有 zeros 函数、ones 函数、eye 函数、rand 函数等；给出稀疏矩阵结果相应的函数有 speye 函数、sprand 函数等。
- 将矩阵变换到矩阵或者向量，其变换函数将以原矩阵的形式出现，即定义在稀疏矩阵上的运算生成稀疏矩阵，定义在满矩阵上的运算生成满矩阵，有 chol 函数、max 函数和 sum 函数等。
- 两个矩阵经运算符（如+、-、*、\、|）操作后的结果一般都是满矩阵，参加运算的矩阵都是稀疏矩阵，其操作后的结果是稀疏矩阵。
- 参与矩阵扩展的子矩阵中，只要有一个是稀疏矩阵，那么所得的结果也是稀疏矩阵。
- 在矩阵引用中，将仍以原矩阵形式给出结果。若 S 矩阵是稀疏矩阵， Y 矩阵是全元素矩阵，不管 I 、 J 是标量还是向量，则“右引用” $Y=S(I,J)$ 产生稀疏矩阵，而“左引用” $S(I,J)=Y$ 产生满矩阵。

2.4 多项式及其函数

MATLAB 提供了大量的多项式操作函数，以方便多项式的计算。

2.4.1 多项式的建立和操作

MATLAB 约定降幂多项式 $P(x)=a_1x^n+a_2x^{n-1}+\cdots+a_nx+a_{n+1}$ 用如下的系数行向量表示： $P=[a_1 \ a_2 \ \cdots \ a_n \ a_{n+1}]$ 。

多项式系数向量的两种创建方法如下：

(1) 直接输入法。将多项式的各项系数依次排放在行向量元素的位置上。

- 多项式系数以降幂次序排列。
- 多项式中缺某幂次项，则此项系数为零。

(2) 使用 poly 函数产生多项式系数向量，具体用法如下。

- $P=\text{poly}(AR)$ ：如果 AR 为方阵，则多项式 P 是该方阵的特征多项式，如果 AR 为向量，则 AR 的元素为多项式 P 的根。 n 阶方阵的特征多项式存放在行向量中，并且特征多项式最高次数的系数一定为 1，显示多项式可以采用 `poly2str(p,'x')` 语句。

【例 2-73】使用 poly 函数产生多项式系数向量。

在命令窗口中输入如下语句：

```
A=[1 2 3 4; 3 5 4 6; 7 6 4 2; 1 5 3 9]    %产生 4x4 的矩阵
P=poly(A)                                %产生多项式系数向量
PP=poly2str(A,'s')                       %显示多项式
```

命令窗口中的输出结果如下所示：

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



```

7      6      4      2
1      5      3      9

```

P =

```
1.0000 -19.0000 28.0000 133.0000 -31.0000
```

PP =

```
s^3 + 3 s^2 + 7 s + 1
```

【例 2-74】计算根 $[1.0000 + 2.0000i \quad 1.0000 - 2.0000i \quad 0 \quad -1.0000 \quad 2.0000 \quad 1.0000]$ 对应的多项式。

在命令窗口中输入如下语句：

```
r1=[1.0000 + 2.0000i 1.0000 - 2.0000i 0 -1.0000 2.0000 1.0000];
```

```
y1=poly(r1)
```

```
r2=[0 -1.0000 2.0000 1.0000 + 2.0000i 1.0000 - 2.0000i 1.0000];
```

```
y2=poly(r2)
```

命令窗口中的输出结果如下所示：

y1 =

```
1 -4 8 -6 -9 10 0
```

y2 =

```
1 -4 8 -6 -9 10 0
```

从本例可以看出，根的排列顺序不影响所创建的多项式。

2.4.2 多项式运算

1. 多项式的根

MATLAB 提供了 `roots` 函数计算多项式的根，计算多项式的根就是计算多项式为零的值。规定 MATLAB 中多项式由一个行向量表示，其系数按降序排列。

【例 2-75】计算多项式 $3x^5 + x^4 - 8.5x^3 + 6x^2 + 20$ 的根。

在命令窗口中输入如下语句：

```
p=[3 1 -8.5 6 0 20]
```

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

命令窗口中的输出结果如下所示：

p =

```
3.0000 1.0000 -8.5000 6.0000 0 20.0000
```

r =

```
-2.2491
```

```
1.3597 + 0.7764i
```

```
1.3597 - 0.7764i
```

```
-0.4018 + 1.0236i
```

```
-0.4018 - 1.0236i
```

在 MATLAB 中,无论是多项式还是它的根都是向量。按惯例规定,多项式是行向量,多项式的根是列向量。

2. 多项式的加减法

MATLAB 不提供直接的函数进行多项式加减法的计算,如果两个多项式向量大小相同,则可以利用数组加减法对这两个多项式进行加减。

【例 2-76】计算多项式 a 和 b 的和。

在命令窗口中输入如下语句:

```
a=[1 2 3 4 5]
b=[5 4 3 2 1]
d=a+b
PP=poly2str(d,'x')
```

命令窗口中的输出结果如下所示:

```
a =
     1     2     3     4     5

b =
     5     4     3     2     1

d =
     6     6     6     6     6

PP =
    6 x^4 + 6 x^3 + 6 x^2 + 6 x + 6
```

【例 2-77】计算多项式 c 和 d 的差。

在命令窗口中输入如下语句:

```
c=[12 23 56 45 13 41]
d=[1 2 5 6 7 8]
e=c-d
PP=poly2str(d,'x')
```

命令窗口中的输出结果如下所示:

```
c =
    12    23    56    45    13    41

d =
     1     2     5     6     7     8

e =
    11    21    51    39     6    33
```

PP =

$x^5 + 2x^4 + 5x^3 + 6x^2 + 7x + 8$

本例需要说明的是，当两个多项式阶次不同时，低阶的多项式必须用首零填补，达到与高阶多项式相同的阶次。

3. 多项式的乘法

MATLAB 提供了 `conv` 函数实现多项式的乘法（计算两个数组的卷积）。

【例 2-78】计算两个多项式 $a(x)=4x^4+3x^2+2x+1$ 和 $b(x)=3x^3-7x^2+8x+9$ 的乘积。

在命令窗口中输入如下语句：

```
a=[4 0 3 2 1]
```

```
b=[3 -7 8 9]
```

```
c=conv(a,b)
```

```
PP=poly2str(c,'x')
```

命令窗口中的输出结果如下所示：

a =

4 0 3 2 1

b =

3 -7 8 9

c =

12 -28 41 21 13 36 26 9

PP =

$12x^7 - 28x^6 + 41x^5 + 21x^4 + 13x^3 + 36x^2 + 26x + 9$

计算两个以上的多项式乘法需要重复使用 `conv` 函数。

4. 多项式的除法

MATLAB 提供了 `deconv` 函数实现多项式的除法。

【例 2-79】计算多项式 b 和 c 的商和余数。

在命令窗口中输入如下语句：

```
b=[1 6 8 4 4]
```

```
c=[12 36 59 42 32 46 17]
```

```
[q,r]=deconv(c,b)
```

命令窗口中的输出结果如下所示：

b =

1 6 8 4 4

c =

12 36 59 42 32 46 17

```

q =
    12    -36    179

r =
         0         0         0    -792   -1304   -526   -699

```

从以上结果不难看出，计算多项式 b 被多项式 c 除，给出商多项式 q 和余数 r ，因为 b 和 q 的乘积恰好是 c ，所以余数 r 是零。

5. 多项式的导数

MATLAB 提供了 `polyder` 函数计算多项式的导数。

【例 2-80】计算下面多项式的导数。

在命令窗口中输入如下语句：

```
g=polyder([4 5 3.2 8 5.5 2.4])
```

命令窗口中的输出结果如下所示：

```

g =
    20.0000    20.0000    9.6000    16.0000    5.5000

```

6. 多项式的积分

MATLAB 提供了 `polyint` 函数计算多项式的积分，它的具体用法如下：

- `i=polyint(p,k)`：返回多项式 p 的积分且常数项为 k ，结果仍为行向量。
- `i=polyint(p)`：返回多项式 p 的积分且常数项为 0，结果仍为行向量。

【例 2-81】计算给定多项式 $6x^2+4x+5$ 的积分。

在命令窗口中输入如下语句：

```

p=[6 4 5];
k1=polyint(p)
k2=polyint(p,5)

```

命令窗口中的输出结果如下所示：

```

k1 =
     2     2     5     0

k2 =
     2     2     5     5

```

7. 多项式的估值

MATLAB 提供了 `polyval` 函数对多项式进行估值运算。

【例 2-82】对多项式进行估值运算。

在命令窗口中输入如下语句：

```

x=linspace(-2,5);
p=[2 3 -5 -3.5 6.2];
v=polyval(p, x); %计算 x 值上的 p(x)，将运行结果存在 v 里
plot(x, v)       %使用 plot 函数给出结果
title(' 2x^4+3x^3-5x^2-3.5x+6.2 '), xlabel(' x '), ylabel(' v ')

```

图形窗口中的输出结果如图 2-5 所示。

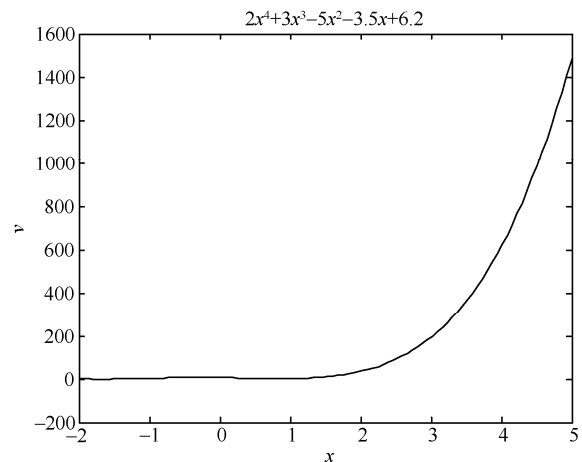


图 2-5 多项式估值

8. 多项式的值

计算多项式在给定点的值，分为代数多项式求值和矩阵多项式求值。

- `y = polyval(p,x)`: 计算多项式 p 在 x 点的值。
- `y=polyvalm(p,X)`: 计算矩阵多项式的值，采用的是普通矩阵运算并且矩阵 X 必须是方阵。

【例 2-83】计算多项式的值。

在命令窗口中输入如下语句：

```
p=p[2 3 -5 -3.5 6.2]
x=2
y = polyval(p,x)
```

命令窗口中的输出结果如下所示：

```
p =
    2.0000    3.0000   -5.0000   -3.5000    6.2000

x =
    2

y =
   35.2000
```

表 2-12 提供了常用的多项式函数，并且介绍了多项式函数的功能。

表 2-12 多项式函数及操作

函数名称	功能介绍	函数名称	功能介绍
<code>conv(a, b)</code>	乘法	<code>[a, b]=residue(r, p, k)</code>	部分分式组合
<code>deconv(a, b)</code>	除法	<code>mmp2str(a)</code>	从多项式向量变换到字符串
<code>poly(r)</code>	使用根构造多项式	<code>mmp2str(a, ' x ')</code>	从多项式向量变换到字符串
<code>polyder(a)</code>	多项式求导	<code>mmp2str(a, ' x ', 1)</code>	常数和符号多项式的变换
<code>polyfit(x, y, n)</code>	多项式数据拟合	<code>mmpadd(a, b)</code>	多项式加法
<code>polyval(p, x)</code>	计算多项式在 x 点的值	<code>mmpsim(a)</code>	化简多项式
<code>[r, p, k]=residue(a, b)</code>	部分分式展开式		

2.4.3 多项式展开

MATLAB 提供了 `residue` 函数来实现多项式展开，多项式展开就是将两个多项式相除的形式用部分分式展开的形式来表示。

`residue` 函数的具体用法如下：

- `[r,p,k] = residue(b,a)`：实现多项式之比 b/a 的部分分式展开， k 是商的多项式， r 是部分分式的留数， p 是部分分式的极点。

若 $a(x)$ 不存在重根，则多项式展开可表示为

$$\frac{b(x)}{a(x)} = \frac{r(1)}{x - p(1)} + \frac{r(2)}{x - p(2)} + \cdots + \frac{r(n)}{x - p(n)} + k(x)$$

若 $a(x)$ 存在 m 重根，即 $p_i = p_{i+1} = \cdots = p_{i+m-1}$ ，则多项式展开可表示为

$$p(x) = \frac{b(x)}{a(x)} = \cdots + \frac{r_i}{x - p_i} + \frac{r_{i+1}}{(x - p_{i+1})} + \cdots + \frac{r_{i+m-1}}{(x - p_{i+1})^m} + \cdots$$

MATLAB 还提供了 `residue` 函数实现多项式与其部分分式之间的转换，即把部分分式和的形式转化为两个多项式相除的形式，具体用法如下。

- `[r,p,k] = residue(b,a)`： b 和 a 都是多项式对应的行向量。
- `[b,a] = residue(r,p,k)`： r 、 p 和 k 都是多项式对应的行向量。

【例 2-84】将多项式进行部分分式展开。

在命令窗口中输入如下语句：

```
a=[6,2,8,3,5,7];
```

```
b=[4,2,4,5];
```

```
[r,p,k]=residue(b,a)
```

命令窗口中的输出结果如下所示：

```
r =
```

```
0.0087 - 0.1916i
```

```
0.0087 + 0.1916i
```

```
-0.0341 - 0.1839i
```

```
-0.0341 + 0.1839i
```

```
0.0506
```

```
p =
```

```
-0.3388 + 1.1240i
```

```
-0.3388 - 1.1240i
```

```
0.5651 + 0.8704i
```

```
0.5651 - 0.8704i
```

```
-0.7861
```

```
k =
```

```
[]
```

2.4.4 多项式拟合

曲线拟合是分析数据常用的方法，它的思想是：从一组或多组数据中找到一条可以用数学函数描述的曲线，这条曲线尽可能多地穿越这些已知数据点。通过判断测量数据点和该曲线上对应点之间的平方误差，来评价这条曲线是否准确描述了测量数据，平方误差越小，曲线拟合的效果越好。

MATLAB 提供了 `polyfit` 函数进行多项式曲线拟合，它的具体用法如下。

- `p = polyfit(x,y,n)`: 使用 n 次多项式 p 来拟合数据 x 和 y ，使得 $p(x)$ 与 y 的最小均方差最小。

【例 2-85】利用 `polyfit` 函数进行多项式拟合，并讨论采用不同多项式阶数对拟合结果的影响。在命令窗口中输入如下语句：

```
x=0:0.1:1;  
y=[0.02,1.2,2.5,3.32,3.45,4.51,4.93,5.7,6.9,8.3,9.26];  
p1=polyfit(x,y,1)  
y1=polyval(p1,x);  
p3=polyfit(x,y,3)  
y3=polyval(p3,x);  
p5=polyfit(x,y,5)  
y5=polyval(p5,x);  
plot(x,y,'.',x,y1,'-',x,y3,'-',x,y5,'-')  
legend('原始数据','1 次多项式拟合','3 次多项式拟合','5 次多项式拟合');
```

图形窗口中的输出结果如图 2-6 所示。

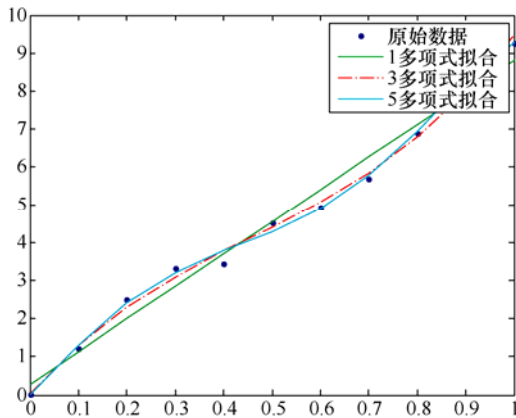


图 2-6 多项式拟合

从图 2-6 可以看出，使用 5 次多项式拟合得到的结果比较好。

2.5 关系和逻辑及其运算

前面在数组运算和矩阵运算中已经简单介绍了关系和逻辑运算操作符，下面将详细介绍关系和逻辑运算的内容，同时介绍 NaN 和空矩阵的差异。

2.5.1 关系和逻辑运算符

1. 关系运算符

在进行关系运算的时候，MATLAB 要求两个操作数具有相同的规模或其中一个是标量。当两个操作数据有相同规模的时候，两个矩阵的对应元素进行比较，返回的矩阵规模和原矩阵相同。MATLAB 的关系运算符与数组运算和矩阵运算中的一致。

【例 2-86】比较 4×4 随机矩阵中哪些元素的值大于 0.5。

在命令窗口中输入如下语句：

```
rand(4)>0.5*ones(4)
```

命令窗口中的输出结果如下所示：

```
ans =
     1     1     1     0
     0     0     0     0
     1     0     1     0
     1     0     0     1
```

返回结果中等于 1 的位置表示 rand(4)的矩阵元素大于 0.5，返回结果中等于 0 的位置表示 rand(4)的矩阵元素小于 0.5。

如果其中一个操作数为标量，MATLAB 在进行比较的时候，将此标量分别与矩阵进行比较，返回的矩阵规模 and 这个比较的矩阵相同。

同样可以通过矩阵和标量的关系运算来实现上述功能，在命令窗口中输入如下语句：

```
rand(4)>0.5
```

MATLAB 还提供了 isempty 函数来判断矩阵是否为空，如果 A 为空矩阵，则 isempty(A) 返回 1，否则返回 0。

2. 逻辑运算符

MATLAB 包括 3 种类型的逻辑运算符，分别为元素方式逻辑运算符、比特方式逻辑运算符和短路逻辑运算符。

(1) 元素方式逻辑运算符包括：逻辑与、逻辑或、逻辑非和逻辑异或，元素方式逻辑运算符只针对具有相同规模的两个操作数，或者其中一个操作数为标量的情况。

- 逻辑与 (&)：两个操作数同时为 1，运算结果为 1，否则为 0。
- 逻辑或 (|)：两个操作数同时为 0，运算结果为 0，否则为 1。
- 逻辑非 (~)：当操作数为 0 时，运算结果为 1，否则为 0。
- 逻辑异或 (xor)：当两个操作数相同时，运算结果为 0，否则为 1。

【例 2-87】分别求矩阵 A 和 B 的逻辑与、逻辑或、逻辑非和逻辑异或。

在命令窗口中输入如下语句：

```
A = [1 1 0 1 0 1];
B = [0 1 1 1 0 0];
C1=A&B
C2=A|B
C3=~A
C4=xor(A,B)
```


命令窗口中的输出结果如下所示：

```
C1 =  
    0    1    0    1    0    0  
  
C2 =  
    1    1    1    1    0    1  
  
C3 =  
    0    0    1    0    1    0  
  
C4 =  
    1    0    1    0    0    1
```

元素方式逻辑运算符有重载函数，符号&、|和~的重载函数分别是 and 函数、or 函数和 not 函数。

（2）比特方式逻辑运算符是指对操作数的比特位进行计算，其对应函数的功能如表 2-13 所示。比特方式逻辑运算符接受逻辑类型和非负整数变量输入，上述元素方式逻辑运算符只接受逻辑类型的变量输入。表 2-13 中使用的变量如下：

```
A = 30;           % binary 11110  
B = 3;            % binary 11
```

表 2-13 比特方式逻辑运算符对应函数

函数名称	功能介绍	例子
bitand	位与，返回操作数对应位相与操作	bitand(A,B) = 20 (binary 10100)
bitor	位或，返回操作数对应位相或操作	bitor(A,B) = 29 (binary 11101)
bitcmp	补码，返回 <i>n</i> 位整数表示的补码	bitcmp(A,5) = 3 (binary 00011)
bitxor	位异或，返回两个操作数的对应位异或操作	bitxor(A,B) = 9 (binary 01001)

（3）短路逻辑运算符。

- 逻辑与（&&）：两个操作数同时为 1，运算结果为 1，否则为 0。
- 逻辑或（||）：两个操作数同时为 0，运算结果为 0，否则为 1。

短路逻辑运算符和元素方式逻辑运算符的运算结果是一样的，但这两者是有区别的。在执行短路逻辑运算时，只有通过第一操作数运算结果不确定的情况下才会考虑第二操作数，在执行元素方式逻辑运算时，需要同时考虑这两个操作数。

例如 A//B 操作，当 A 为 1 时，直接返回 1，而不检查 B 的值；当 A 为 0 时，如果 B 为 0，则返回 0，否则返回 1。A&&B 的执行方式与 A||B 类似。在需要满足特定的条件才能被执行的语句中，均可使用短路逻辑运算符。

【例 2-88】利用短路逻辑运算符实现条件判断。

在命令窗口中输入如下语句：

```
a=3;
```

```
b=0;
x = (b ~= 0) && (a/b > 1)
```

命令窗口中的输出结果如下所示：

```
x =
    0
```

这里需要说明的是，当 $b=0$ 时函数直接返回 0，而不计算 a/b 。

2.5.2 关系和逻辑函数

除了上面的关系和逻辑操作符，MATLAB 还提供了一些关系和逻辑函数，如表 2-14 所示。

表 2-14 关系和逻辑函数

函数名称	功能介绍
xor(x,y)	异或运算， x 或 y 非零（真）返回 1， x 和 y 都是零（假）或都是非零（真）返回 0
any(x)	如果一个向量 x 中有非 0 元素返回 1，否则为 0
all(x)	如果一个向量 x 中所有元素都是非零返回 1，否则为 0
isequal(x,y)	x 和 y 对应元素相等时，相应元素位置为 1，否则为 0
ismember(x,y)	x 元素是 y 的子集，相应 x 元素位置为 1，否则为 0

MATLAB 还提供了大量函数来测试变量，如表 2-15 所示。

表 2-15 测试函数

函数名称	功能介绍	函数名称	功能介绍
isfinite	参量为有限元素，返回真值	isreal	参量为实数，返回真值
isempty	参量为空，返回真值	isspace	元素为空格字符，返回真值
isglobal	参量是全局变量，返回真值	ischar	参量为一个字符串，返回真值
ishold	当前绘图保持状态是'ON'，返回真值	istudent	MATLAB 为学生版，返回真值
isieee	计算机执行 IEEE 算术运算，返回真值	isunix	计算机为 UNIX 系统，返回真值
isinf	元素无穷大，返回真值	isvms	计算机为 VMS 系统，返回真值
isletter	元素为字母，返回真值	isprime	参量为质数，返回真值
isnan	元素为不定值，返回真值		

2.5.3 NaN 和空矩阵

非数值量 NaN 是 Not a Number 的缩写，规定 $0/0$ 、 ∞/∞ 和 $0 \times \infty$ 等运算均产生非数值量，由于非数值量没有大小概念，所以不能比较非数值量的大小。

【例 2-89】非数值量的运算。

在命令窗口中输入如下语句：

```
a= [Inf NaN 2 5 8 NaN ] ;
b=4*a
c=sqrt(a)
d=(a==nan)
f=(a~=nan)
```

命令窗口中的输出结果如下所示：

```
b =
    Inf    NaN     8    20    32    NaN
```

```
c =  
      Inf      NaN      1.4142      2.2361      2.8284      NaN  
  
d =  
      0      0      0      0      0      0  
  
f =  
      1      1      1      1      1      1
```

上面的前两个表达式对于输入 NaN 都给出 NaN 结果，然而最后两个表达式产生的结果不为 NaN。当 NaN 与 NaN 相比较时(a==NaN)，结果全部为 0 或假，同时(a~=NaN)结果全部为 1 或真。由于 NaN 的这种特性，MATLAB 提供了 isnan 函数判断 NaN。

【例 2-90】使用 find 命令找出非数值量 NaN 的下标值。

在命令窗口中输入如下语句：

```
a=[2 3 NaN Inf NaN];  
g=isnan(a)  
i=find(isnan(a))
```

命令窗口中的输出结果如下所示：

```
g =  
      0      0      1      0      1  
  
i =  
      3      5
```

空矩阵是 MATLAB 为表述和操作专门设计的一种矩阵，可以看做维数为 0 的矩阵。当运算无结果时，MATLAB 有些函数将返回空矩阵。

【例 2-91】通过以下命令对空矩阵的产生加深理解。

在命令窗口中输入如下语句：

```
x=(3:6)+2  
y=find(x>6)  
isempty(y)%判断矩阵是否为空  
class(y)%判断数据类别
```

命令窗口中的输出结果如下所示：

```
x =  
      5      6      7      8  
  
y =  
      3      4  
  
ans =  
      0
```

```
ans =  
double
```

本例中没有包含 x 大于 6 的值，所以没有返回下标，MATLAB 提供了 `isempty` 函数测试空矩阵。

【例 2-92】通过以下命令对空矩阵的性质加深理解。

在命令窗口中输入如下语句：

```
x=[]  
y=ones(3,0)  
x==y
```

命令窗口中的输出结果如下所示：

```
x =  
  
[]  
  
y =  
Empty matrix: 3-by-0  
  
??? Error using ==> eq  
Matrix dimensions must agree.
```

【例 2-93】通过以下命令对空矩阵的性质加深理解。

在命令窗口中输入如下语句：

```
x==0  
x~=0  
find(y==0)
```

命令窗口中的输出结果如下所示：

```
ans = %结果可解释为不等于  
  
[]  
  
ans = %结果可解释为不等于  
  
[]  
  
ans =  
Empty matrix: 0-by-1
```

本例需要说明的是，空矩阵不是全零矩阵，不等于任何非零矩阵（或标量）。

第3章 MATLAB 符号运算

MATLAB 符号运算是由集成在 MATLAB 中的符号数学工具箱 (Symbolic Math Toolbox) 来完成的, 符号数学工具箱使用字符串进行符号分析与运算。在 MATLAB 中, 符号数学工具箱中的工具都是建立在数学计算软件 Maple 的基础上的, 如果进行符号运算, MATLAB 会调用 Maple, 并将结果返回到 MATLAB 的命令窗口中。

符号数学工具箱包含基本的符号数学工具箱和扩展的符号数学工具箱两个子工具箱。基本的符号数学工具箱是 100 多个 MATLAB 函数的集合, 通过使用 MATLAB 语言的语法和类型来访问 Maple 内核入口, 调用 Maple 函数; 扩展的符号数学工具箱增加了访问所有 Maple 非图形化的程序包、Maple 编程特性和用户自定义程序等功能。

MATLAB 的符号数学工具箱可以完成几乎所有的符号运算功能, 这些功能主要包括符号表达式的运算, 符号表达式的复合和化简, 符号矩阵的运算, 符号微积分, 符号函数画图, 符号代数方程求解和符号微分方程求解等。此外, 符号数学工具箱还支持可变精度的运算, 并以指定的精度返回结果。

3.1 符号运算入门

在科学工程中, 数值运算是非常重要的内容, 但是在自然科学理论分析中, 各种各样的公式和关系式的推导等问题也是十分重要的, 这些就是符号运算解决的问题。

符号运算与数值运算的区别在于:

- MATLAB 数值运算的对象是数值, 而符号运算的对象是非数值的符号对象, 即非数值的符号字符串。
- 数值计算要先对变量赋值再计算, 而符号运算不需要赋值即可直接计算, 计算结果是符号表达式。
- 符号运算可以获得任意精度的解。

3.1.1 符号对象的创建函数

符号数学工具箱 (Symbolic Math Toolbox) 定义了 MATLAB 的一个新的数据类型就是符号对象, 符号对象可以是变量、表达式和矩阵等。符号对象是一种数据结构, 用来存储代表符号的字符串, 在进行符号运算时, 首先对符号对象进行定义, 其次利用这些符号对象构成表达式, 最后进行所需的符号运算。符号表达式由符号常量、符号变量和符号函数等符号对象构成, sym 函数和 syms 命令可以规定和创建符号对象以及符号表达式。

1. sym 函数

在进行符号运算之前, 首先需要定义符号变量或符号表达式, 然后利用这些符号对象构建表达式, 最后才进行符号运算。MATLAB 提供了 sym 函数将数值类型变量转换成符号类型变量, 它的具体用法如下所示:

- `s=sym(A,flag)`
- `s=sym('A',flag)`

即由 A 建立一个符号对象或符号表达式 s，具体说明如下：

• 不带单引号的 A 是一个数字、数值矩阵或数值表达式，输出是将数值对象转换成的符号对象或符号表达式。

- 带单引号的 A 是一个字符串，输出是将字符串转换成的符号对象或符号表达式。
- `sym` 函数可以用来定义抽象函数、指定符号变量类型、访问 Maple 中的函数，还可以创建符号矩阵。需要说明的是，定义常数的符号变量必须采用 `s=sym(A,flag)` 形式。
- `flag` 是指所转换符号对象的格式，也就是说符号对象建立时可以附加属性。

当被转换的对象是数值对象时，`flag` 有以下几种形式：

- (1) `flag` 表示为 'd'，指精确的十进制浮点数。
- (2) `flag` 表示为 'e'，指估计误差（进行数值计算时）是有理数。
- (3) `flag` 表示为 'f'，指十六进制浮点数。
- (4) `flag` 表示为 'r'，默认设置下，最接近的有理表示形式。

当被转换的对象是字符串时，`flag` 有以下几种形式：

- (1) `flag` 表示为 'positive'，指 A 是正的实型符号变量。
- (2) `flag` 表示为 'real'，指 A 是实型符号变量。
- (3) `flag` 表示为 'unreal'，指 A 是非实型符号变量。

2. syms 命令

MATLAB 还提供 `syms` 命令同时定义多个变量，它的具体用法如下所示：

```
syms s1 s2 s3 flag
```

s1、s2 和 s3 均为变量名，`flag` 是所转换符号对象的格式，规定同上。

3.1.2 符号对象的创建

1. 符号常量的创建

符号常量是一种符号对象。数值常量如果作为函数 `sym()` 的输入变量，这就建立了一个符号对象——符号常量，即看上去是一个数值量，但它已经是一个符号对象。创建的这个符号对象可以用 `class()` 函数检测其数据类型。

【例 3-1】对数值 1/6 创建符号对象并检测数据的类型。

在命令窗口中输入如下语句：

```
a=1/6;  
b='1/6';  
c=sym(1/6);  
d= sym('1/6');  
classa=class(a)  
classb=class(b)  
classc=class(c)  
classd=class(d)    %检验变量类型
```

命令窗口中的输出结果如下所示：

```
classa =
```

```
double
```

```
classb =  
char
```

```
classc =  
sym
```

```
classd =  
sym
```

从上面的结果不难看出，a 是双精度浮点数值类型；b 是字符类型；c 与 d 都是符号对象类型。

2. 符号变量的创建

变量是程序语言的基本元素之一，符号变量是指内容可变的符号对象。符号变量通常是指一个或几个特定的字符，不是指符号表达式，虽然可以将一个符号表达式赋值给一个符号变量。符号变量有时也叫做自由变量。符号变量的命名规则如下：

- 变量名可以由英文字母、数字和下画线组成。
- 变量名以英语字母开头。
- 组成变量名的字母长度不大于 31。
- MATLAB 中区分英语字母的大小写。

MATLAB 提供了 `sym` 函数和 `syms` 命令创建符号变量。

【例 3-2】 使用 `sym` 函数创建符号变量。

在命令窗口中输入如下语句：

```
a=sym('x');  
b=sym('y');  
c=sym('z');  
classa=class(a)  
classb=class(b)  
classc=class(c)
```

命令窗口中的输出结果如下所示：

```
classa =  
sym  
  
classb =  
sym  
  
classc =  
sym
```

【例 3-3】 使用 `syms` 命令创建符号变量。

在命令窗口中输入如下语句：

```
syms x y z;
```

```

classa=class(x)
classb=class(y)
classc=class(z)

```

命令窗口中的输出结果如下所示：

```

classa =
sym

classb =
sym

classc =
sym

```

3. 符号矩阵的创建

元素是符号对象（符号变量与符号常量）的矩阵叫做符号矩阵。符号矩阵既可以构成符号矩阵函数，也可以构成符号矩阵方程，它们都是符号表达式。

（1）使用 sym 函数创建符号矩阵

sym 函数创建的符号矩阵，其元素是任何不带等号的符号表达式。各元素字符串的长度可以不相等，符号矩阵内的元素之间使用逗号“,”或空格“ ”隔开，各行之间使用分号“;”隔开。

【例 3-4】使用 sym 函数创建符号矩阵。

在命令窗口中输入如下语句：

```

x=sym('aa bb cc;cos(x)^2 1/(3+x^2) exp(a);ee ff gg')
y=sym('[1 2 3;4 5 6;7 8 9]')

```

命令窗口中的输出结果如下所示：

```

x =
[ aa,      bb,      cc]
[ cos(x)^2, 1/(x^2 + 3), exp(a)]
[ ee,      ff,      gg]

y =
[ 1, 2, 3]
[ 4, 5, 6]
[ 7, 8, 9]

```

（2）使用直接输入法创建符号矩阵

直接输入法要求每一行用方括号对“[]”括起来，并且使用分号隔开，各元素之间使用逗号“,”隔开。同时要保证同一列上的各行字符长度相同，不够用空格补齐。

【例 3-5】使用直接输入法创建符号矩阵。

在命令窗口中输入如下语句：

```

s='[(a+b),sin(b),exp(a)]':['a*b ',log(b),b  ]':['abc ',cos(b),ab  ]']

```

命令窗口中的输出结果如下所示：

```

s =

```



```
[(a+b),sin(b), exp(a)]
```

```
[a*b ,log(b),b ]
```

```
[abc ,cos(b),ab ]
```

(3) 将数值矩阵转化成符号矩阵

由于数值矩阵不能直接参与符号运算，因此在符号运算中需要先将数值矩阵转化为符号矩阵，`sym` 函数可以实现将数值矩阵转化成符号矩阵。

【例 3-6】将数值矩阵转化为符号矩阵。

在命令窗口中输入如下语句：

```
s2=[2/3,sqrt(2),4.2; 5.2, log(3),sin(3);6.5,cos(pi),1.3]
```

```
s3=sym(s2)
```

命令窗口中的输出结果如下所示：

```
s2 =
```

```
0.6667    1.4142    4.2000
```

```
5.2000    1.0986    0.1411
```

```
6.5000   -1.0000    1.3000
```

```
s3 =
```

```
[ 2/3,                2^(1/2),                21/5]
```

```
[ 26/5, 2473854946935173/2251799813685248, 5084384125703515/36028797018963968]
```

```
[ 13/2,                -1,                13/10]
```

4. 符号表达式的创建

表达式也是程序设计语言的基本元素之一。在 MATLAB 的符号运算中，符号表达式是由符号常量、符号变量、符号函数运算符以及专用函数连接而成的符号对象。符号表达式有两类：符号函数和符号方程，两者的区别在于，符号函数不带等号，而符号方程带等号。使用 `sym` 函数和 `syms` 命令可以建立符号表达式。

【例 3-7】使用 `sym` 函数和 `syms` 命令建立符号函数并检测符号对象的类型。

在命令窗口中输入如下语句：

```
syms x y z;
```

```
f1 =x*y^x/z
```

```
classf1=class(f1)
```

```
f2 = sym(sin(x)*x+y^2+cos(z))
```

```
classf2=class(f2)
```

```
f3 = f1/f2
```

```
classf3=class(f3)
```

```
f4=pi+atan(y*z)
```

```
classf4=class(f4)
```

命令窗口中的输出结果如下所示：

```
f1 =
```

```
(x*y^x)/z
```

```
classf1 =  
sym  
  
f2 =  
y^2 + cos(z) + x*sin(x)  
  
classf2 =  
sym  
  
f3 =  
(x*y^x)/(z*(y^2 + cos(z) + x*sin(x)))  
  
classf3 =  
sym  
  
f4 =  
pi + atan(y*z)  
  
classf4 =  
sym
```

【例 3-8】使用 `sym` 函数建立符号方程。

在命令窗口中输入如下语句：

```
x = sym('a*x^2+b*x+c=1')  
y = sym('sin(x)^2+cos(x)=0')  
z = sym('m-2=n')
```

命令窗口中的输出结果如下所示：

```
x =  
a*x^2 + b*x + c = 1  
  
y =  
sin(x)^2 + cos(x) = 0  
  
z =  
m - 2 = n
```

3.1.3 符号运算中的运算符

1. 基本运算符

MATLAB 采用了重载技术，使符号运算符和数值运算符几乎完全相同，为我们进行计算提供了方便。

以下是对符号计算中基本运算符和函数的简要归纳：

(1) 运算符“+”、“-”、“*”、“\”、“/”和“^”分别实现矩阵的加、减、乘、左除、右除和求幂运算。

(2) 运算符“.*”、“./”、“.\”和“.^”分别实现数组乘、左除、右除和求幂运算。

(3) 运算符“'”和“.'”分别实现矩阵的共轭转置和转置。

2. 关系运算符

对于符号对象的关系运算，没有“大于”、“大于等于”、“小于”和“小于等于”的概念，只有“等于”的概念。

运算符“==”和“~=”分别对运算符两端的对象进行“相等”和“不等”的比较。当比较的结果为“真”时，返回结果 1，否则返回 0。

3.1.4 符号表达式中自变量的确定

在微积分、函数表达式化简、解方程中，确定自变量是必不可少的步骤。在不指定自变量的情况下，一般按数学常规来确定自变量，通常都是字母表中的小写字母，如 m、n、x、y 和 z 等。MATLAB 提供了 findsym 函数按数学常规确定表达式中的自变量。

findsym 函数的具体用法如下所示。

- findsym(f,n): 按数学常规确定符号函数 f 中的 n 个自变量。当 n=1 时，说明有一个自变量，于是从字母表中寻找与 x 最接近的字母为自变量；如果有两个字母与 x 的距离相等，要选靠后的一个；当 n 为默认值时，findsym 函数会给出符号函数 f 中所有的符号变量。
- findsym(e,n): 这种表达形式与上面的表达形式的不同之处在于，此表达式确定的是符号方程 e 中的 n 个自变量，其他与上面相同。

【例 3-9】使用 findsym 函数确定符号表达式的自变量。

在命令窗口中输入如下语句：

```
syms k m n x y z;  
f=n*y+m*y+x;  
ans1=findsym(f,1)  
f2=m*y+n*log(x)+exp(x*y*z);  
ans2=findsym(f2,2)
```

命令窗口中的输出结果如下所示：

```
ans1 =  
x  
  
ans2 =  
x,y
```

从输出结果可以看出，符号表达式中的自变量选择了与 x 距离最近的符号变量。

【例 3-10】使用 findsym 函数确定符号表达式的自变量。

在命令窗口中输入如下语句：

```
f=sym('w*y+y^2+w*z');  
findsym(f,1)
```

命令窗口中的输出结果如下所示：

```
ans =  
y
```

从输出结果可以看出，符号变量有 w 、 y 和 z 。很明显， x 与 w 和 y 的距离是相等的，但是符号表达式中的自变量选择了靠后的符号变量 y 。

【例 3-11】使用 `findsym` 函数确定符号表达式的自变量。

在命令窗口中输入如下语句：

```
e=sym('m*a+5*b+c+d=0');  
findsym(e,2)  
findsym(e)
```

命令窗口中的输出结果如下所示：

```
ans =  
m,d  
  
ans =  
a,b,c,d,m
```

从输出结果不难看出，默认情况下 `findsym` 函数给出符号方程中所有的符号变量。

3.2 符号表达式运算

符号表达式的运算包括求表达式的值，对表达式进行加、减、乘、除标准代数运算，数值转换，以及变量替换等。

3.2.1 提取分子和分母

进行提取分子和分母运算时，符号表达式要求是一个有理分式，即是两个多项式比的形式，其中包括分母为 1 的情况。`MATLAB` 提供了 `numden` 函数来提取分子和分母，必要的时候还可以在表达式合并之后再提取分子和分母。

`numden` 函数的具体用法如下所示。

- `[n,d]=numden(f)`： f 指要进行提取的有理分式， n 指提取后的分子， d 指提取后的分母。

【例 3-12】提取函数 $f = \frac{6x^2 + 3y}{2y + 1}$ 的分子和分母。

在命令窗口中输入如下语句：

```
f=sym(' (6*x^2+3*y)/(2*y+1) ')  
[n,d]=numden(f)
```

命令窗口中的输出结果如下所示：

```
f =  
(6*x^2 + 3*y)/(2*y + 1)  
  
n =  
6*x^2 + 3*y
```

```
d =  
2*y + 1
```

【例 3-13】提取函数 $f = \frac{2}{5}x^4 + \frac{3}{4}x^2 + 7$ 的分子和分母。

在命令窗口中输入如下语句：

```
f=sym('2/5*x^4+3/4*x^2+7')  
[n,d]=numden(f)
```

命令窗口中的输出结果如下所示：

```
f =  
(2*x^4)/5 + (3*x^2)/4 + 7  
  
n =  
8*x^4 + 15*x^2 + 140  
  
d =  
20
```

本例需要说明的是，`numden` 函数可以在对有理式进行通分、合并同类项之后再提取分子和分母，`d=20` 就是在对有理式通分之后得到的分母的值。

3.2.2 标准代数运算

标准的代数运算，如加、减、乘、除，同样适用在符号表达式中，符号表达式在执行标准运算时与代数运算相同。

【例 3-14】分别计算下列两个符号表达式的加、减、乘和除。

在命令窗口中输入如下语句：

```
a=sym('x^4+2*x^2-exp(4)')  
b=sym('x^3-4*x+sin(x)')  
a+b  
a-b  
a/b  
a*b
```

命令窗口中的输出结果如下所示：

```
a =  
x^4 + 2*x^2 - exp(4)  
  
b =  
sin(x) - 4*x + x^3  
  
ans =  
sin(x) - exp(4) - 4*x + 2*x^2 + x^3 + x^4  
  
ans =
```

```

4*x - exp(4) - sin(x) + 2*x^2 - x^3 + x^4

ans =
(x^4 + 2*x^2 - exp(4))/(sin(x) - 4*x + x^3)

ans =
(x^4 + 2*x^2 - exp(4))*(sin(x) - 4*x + x^3)

```

3.2.3 复合符号函数运算

设 z 是 y (自变量) 的函数 $z = f(y)$, 而 y 又是 x (自变量) 的函数 $y = \varphi(x)$, 则 z 对 x 的函数: $z = f(\varphi(x))$ 叫做 z 对 x 的复合函数。求 z 对 x 的复合函数 $z = f(\varphi(x))$ 的过程叫做复合函数运算。

MATLAB 提供了 `compose` 函数用于实现符号函数的复合, 其调用格式有以下 6 种。

- `fg= compose(f,g)`: 这种格式的功能是当 $f = f(x)$ 与 $g = g(y)$ 时, 返回复合函数 $f(g(y))$, 即用 $g = g(y)$ 代入 $f(x)$ 中的 x , 且 x 为命令 `findsym()` 确定的 f 的自变量, y 为 `findsym()` 确定的 g 的自变量。
- `fg= compose(f,g,z)`: 这种格式的功能是当 $f = f(x)$ 与 $g = g(y)$ 时, 返回以 z 为自变量的复合函数 $f(g(z))$, 即用 $g = g(y)$ 代入 $f(x)$ 中的 x , 且 $g(y)$ 中的自变量 y 改换为 z 。
- `fg= compose(f,g,x,z)`: 这种格式的功能同第 2 种格式的功能。
- `fg= compose(f,g,t,z)`: 这种格式的功能是当 $f = f(t)$ 与 $g = g(y)$ 时, 返回以 z 为自变量的复合函数 $f(g(z))$, 即用 $g = g(y)$ 代入 $f(t)$ 中的 t , 且 $g(y)$ 中的自变量 y 改换为 z 。
- `fg=compose(f,h,x,y,z)`: 这种格式的功能同第 2 种格式与第 3 种格式的功能。
- `fg=compose(f,g,t,u,z)`: 这种格式的功能是当 $f = f(t)$ 与 $g = g(u)$ 时, 返回以 z 为自变量的复合函数 $f(g(z))$, 即用 $g = g(u)$ 代入 $f(t)$ 中的 t , 且 $g(u)$ 中的自变量 u 改换为 z 。

【例 3-15】计算函数 $f = \ln\left(\frac{x^2 + 1}{t}\right)$ 和函数 $g = u + \cos(f)$ 的复合符号函数。

在命令窗口中输入如下语句:

```

syms f g t u x y z;
f=log((x^2+1)/t);
g=cos(f)
fg=compose(f,g)
fgt= compose(f,g,z)
fgxz=compose(f,g,x,z)
fgtz= compose(f,g,t,z)
fgxyz= compose(f,g,x,y,z)
fgxyz= compose(f,g,t,u,z)

```

命令窗口中的输出结果如下:

```

fg =
log((cos(log((x^2 + 1)/t))^2 + 1)/t)

fgt =

```

```
log((cos(log((z^2 + 1)/t))^2 + 1)/t)

fgxz =
log((cos(log((z^2 + 1)/t))^2 + 1)/t)

fgtz =
log((x^2 + 1)/cos(log((z^2 + 1)/t)))

fgxyz =
log((cos(log((x^2 + 1)/t))^2 + 1)/t)

fgxyz =
log((x^2 + 1)/cos(log((x^2 + 1)/t)))
```

3.2.4 数值转换

1. 常值转换函数

可以利用数据类型转换函数实现将符号常数转换为数值，表 3-1 提供了常见的数据类型转换函数。

表 3-1 常见的数据类型转换函数

函数名称	功能介绍
logical	将数值转化为逻辑值
char	创建字符串数组或者将其他类型变量转化为字符串数组
int8	转换为 8 字节整型数
uint8	转换为 8 字节数
int16	转换为 16 字节整型数
uint16	转换为 16 字节数
int32	转换为 32 字节整型数
uint32	转换为 32 字节数
int64	转换为 64 字节整型数
uint64	转换为 64 字节数
single	转换为单精度浮点型
double	转换为 64 字节浮点数
cell	转换为细胞数组
struct	转换为创建结构体类型

【例 3-16】利用数据类型转换函数对符号常数进行转化。

在命令窗口中输入如下语句：

```
syms c1 c2 c3;
c1=sym('5.4');
c2=sym('exp(2.6)');
c3=sym('pi');
ans1=int8(c1)
ans2=uint16(c2)
ans3=double(c3)
```

命令窗口中的输出结果如下所示：

```
ans1 =
     5

ans2 =
    13

ans3 =
    3.1416
```

2. sym2poly 函数

MATLAB 提供了 sym2poly 函数用于将符号表达式转换为数值多项式的系数向量，并且系数从高到低依次排列。

【例 3-17】使用 sym2poly 函数显示数值多项式的系数向量。

在命令窗口中输入如下语句：

```
syms x;
f=sym(6+5*x+3*x^3+2*x^4+x^5);
sym2poly(f)
```

命令窗口中的输出结果如下所示：

```
ans =
     1     2     3     0     5     6
```

3. poly2sym 函数

MATLAB 提供了 poly2sym 函数用于将数值多项式的系数向量转换为符号表达式，作用与 sym2poly 函数相反。

【例 3-18】使用 poly2sym 函数将数值多项式的系数向量转换为符号表达式。

在命令窗口中输入如下语句：

```
a=[1     2     3     0     5     6];
poly2sym(a)
```

命令窗口中的输出结果如下所示：

```
ans =
x^5 + 2*x^4 + 3*x^3 + 5*x + 6
```

4. eval 函数

MATLAB 提供了 eval 函数用于计算符号表达式的值。

【例 3-19】计算符号表达式 $2x+3y$ 的值。

在命令窗口中输入如下语句：

```
f=sym('2*x+3*y');
x=sym('4');
y=sym('6');
p=eval(f)
class(p)
```

命令窗口中的输出结果如下所示：


```
p =  
26
```

```
ans =  
sym
```

从上面的结果不难看出，`p` 的显示形式仿佛是数值，但是它本身仍然是符号变量。同时 `eval` 函数还可以用于生成矩阵。

【例 3-20】用 `eval` 函数生成 3 到 5 阶的魔方矩阵。

在命令窗口中输入如下语句：

```
for n=3:5  
eval(['M' num2str(n) '=magic(n)'])  
end
```

命令窗口中的输出结果如下所示：

```
M3 =
```

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

```
M4 =
```

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

```
M5 =
```

```
17    24     1     8    15  
23     5     7    14    16  
4      6    13    20    22  
10    12    19    21     3  
11    18    25     2     9
```

3.2.5 变量替换

计算中经常会遇到复杂的函数和复杂的方程，进行变量替换会使计算变得简单。MATLAB 提供了 `subs` 函数用于实现变量的替换，它的具体用法如下所示。

- `subs(S,old,new)`：用 `new` 来替换 `S` 中的 `old` 变量，`old` 必须是 `S` 中的符号变量，`new` 可以是符号变量、符号常量、数值和数组等。
- `subs(S,new)`：用 `new` 来替换 `S` 中的自变量，自变量的确定和 `findsym` 函数的要求相同。
- `subs(S)`：用工作空间中的变量值替换符号表达式 `S` 中的所有符号变量，如果符号变量的值未指定，则该符号变量不被替换。

【例 3-21】使用 `subs` 函数进行变量替换。

在命令窗口中输入如下语句：

```
syms x y;
f=x^2+y^2;
subs(f,x,2)
```

命令窗口中的输出结果如下所示：

```
ans =
y^2 + 4
```

【例 3-22】使用 subs 函数进行变量替换。

在命令窗口中输入如下语句：

```
syms t a;
subs(exp(a*t), 'a', -magic(2))
```

命令窗口中的输出结果如下所示：

```
ans =
[ 1/exp(t), 1/exp(3*t)]
[ 1/exp(4*t), 1/exp(2*t)]
```

从上面的结果不难看出，用于变量替换的部分可以是数组形式。

3.2.6 化简与格式化

符号运算中的表达式往往都是烦琐的，MATLAB 提供了多种手段对符号表达式进行简化，如因式分解、符号表达式的通分和化简、合并同类项等。这些都是对符号表达式的恒等变形，不会改变表达式本身的内容。

MATLAB 提供了多种函数实现上述化简。

1. factor 函数

MATLAB 提供了 factor 函数用于实现表达式的因式分解，它的具体用法如下所示。

- factor(x)：若 x 可以分解成多项式或多项式矩阵的形式，并且系数是有理数，factor 函数可以把 x 表示为系数是有理数的低阶多项式的乘积。若 x 不能分解成多项式或多项式矩阵的形式，则返回原来的 x。对于大于 2^{52} 整数的分解，可使用语句 factor(sym('N'))。

【例 3-23】使用 factor 函数对 $f=x^3+x^2-x-1$ 进行因式分解。

在命令窗口中输入如下语句：

```
f=sym('x^3+x^2-x-1');
f1=factor(f)
```

命令窗口中的输出结果如下所示：

```
f1 =
(x - 1)*(x + 1)^2
```

【例 3-24】使用 factor 函数对符号矩阵表达式进行因式分解。

在命令窗口中输入如下语句：

```
syms x m n ;
n=1:10
x=x(ones(size(n)));
m=x.^n+1;
```

```
[m;factor(m)].'
```

命令窗口中的输出结果如下所示：

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

ans =
[      x + 1,                                x + 1]
[   x^2 + 1,                                x^2 + 1]
[   x^3 + 1,                                (x + 1)*(x^2 - x + 1)]
[   x^4 + 1,                                x^4 + 1]
[   x^5 + 1,                                (x + 1)*(x^4 - x^3 + x^2 - x + 1)]
[   x^6 + 1,                                (x^2 + 1)*(x^4 - x^2 + 1)]
[   x^7 + 1,                                (x + 1)*(x^6 - x^5 + x^4 - x^3 + x^2 - x + 1)]
[   x^8 + 1,                                x^8 + 1]
[   x^9 + 1,                                (x + 1)*(x^2 - x + 1)*(x^6 - x^3 + 1)]
[ x^10 + 1,                                (x^2 + 1)*(x^8 - x^6 + x^4 - x^2 + 1)]
```

从上面的结果不难看出，输出的矩阵中每行的第一个元素为分解前的多项式，第二个元素为分解后的多项式。

2. collect 函数

MATLAB 提供了 `collect` 函数用于实现对符号表达式进行合并同类项运算，它的具体用法如下所示。

- `collect(S)`：将符号表达式 S 中相同幂次的项合并， S 可以是表达式，也可以是符号矩阵。
- `collect(S,v)`：这是一种恒等变换，其功能是将符号表达式 S 中 v 的相同幂次的项进行合并，如果 v 没有指定，则默认将含有 x 的相同幂次的项合并。

【例 3-25】 使用 `collect` 函数的第一种表达方法进行合并同类项运算。

在命令窗口中输入如下语句：

```
syms a b n x y;
f=(x^2*y+x*y-a*x^2-b*x)*(x+1);
collect(f)
```

命令窗口中的输出结果如下：

```
ans =
(y - a)*x^3 + (2*y - b - a)*x^2 + (y - b)*x
```

【例 3-26】 使用 `collect` 函数的第二种表达方法进行合并同类项运算。

在命令窗口中输入如下语句：

```
syms a b c x ;
f=-a*x*exp(-c*x)+b*exp(-c*x);
f1=collect(f,exp(-c*x))
collect(f)
```

命令窗口中的输出结果如下所示：

```
f1 =
(b - a*x)/exp(c*x)
```

```
ans =
(-a/exp(c*x))*x + b/exp(c*x)
```

3. horner 函数

MATLAB 提供了 `horner` 函数用于实现将多项式分解成嵌套的形式, 它的具体用法如下所示。

- `horner(S)`: `S` 是符号多项式矩阵, `horner` 函数将每个多项式转换成嵌套形式。

【例 3-27】使用 `horner` 函数将多项式分解成嵌套形式。

在命令窗口中输入如下语句:

```
syms x;
f=3*x^5+4*x^3-8*x+4;
horner(f)
```

命令窗口中的输出结果如下所示:

```
ans =
x*(x^2*(3*x^2 + 4) - 8) + 4
```

4. expand 函数

MATLAB 提供了 `expand` 函数用于实现对表达式进行多项式、指数函数、对数函数以及三角函数的展开, 它的具体用法如下所示。

- `expand(S)`: 若表达式 `S` 是多项式, 则展开为相应和的形式。若表达式 `S` 是三角函数、指数函数和对数函数, 可根据要求展开成相应形式。

【例 3-28】使用 `expand` 函数对多项式进行相应展开。

在命令窗口中输入如下语句:

```
syms x y;
f=(x+y)^4+x^2+32;
expand(f)
```

命令窗口中的输出结果如下所示:

```
ans =
x^4 + 4*x^3*y + 6*x^2*y^2 + x^2 + 4*x*y^3 + y^4 + 32
```

【例 3-29】使用 `expand` 函数对余弦函数进行相应展开。

在命令窗口中输入如下语句:

```
syms x y;
f=cos(x+y);
expand(f)
```

命令窗口中的输出结果如下所示:

```
ans =
cos(x)*cos(y) - sin(x)*sin(y)
```

5. simplify 函数

MATLAB 提供了 `simplify` 函数通过使用多种恒等式转换, 实现对表达式进行综合化简, 它的具体用法如下所示:

- `simplify(S)`: 表达式 `S` 可以是一般多项式, 也可以是符号表达式矩阵。`simplify` 函数应用范围广、功能强, 可以使用在指数函数、对数函数、三角函数、方根以及平方等表达式

中，并且能利用 Maple 化简规则对表达式进行简化。

【例 3-30】使用 simplify 函数对表达式进行综合化简。

在命令窗口中输入如下语句：

```
syms x;
e=sin(x)^2+cos(x)^2;
e1=simplify(e)
```

命令窗口中的输出结果如下所示：

```
ans =
1
```

6. simple 函数

MATLAB 提供了 simple 函数用于实现将表达式转变成最简形式，它的具体用法如下所示。

- **[r,how] = simple(S)**: 这条语句只显示寻找到的最短形式以及找到该形式所用的方法。返回值中，r 是最短形式的符号表达式，how 是一个描述简化方法的字符串。
- **r=simple(S)**: simple 函数可以在简化的基础上进一步给出多种简化的形式，返回表达式 S 的最短形式。如果 S 是符号表达式矩阵，则返回使整个矩阵变成最短的形式；如果输出参数 r 没有给定，simple 函数将显示表达式 S 所有的简化形式，并返回其中最短的一个。符号表达式简化的功能强大，由于简化方式有很多，因此有时产生的结果会不同。

【例 3-31】分别利用上面两条指令，对表达式 $f = \frac{(x+2) \times (x^3-1)}{x-1}$ 进行化简。在命令窗口

中输入如下语句：

```
syms x;
f=('(x+2)*(x^3-1)/(x-1)');
[r,how]=simple(f)
simple(f)
```

命令窗口中的输出结果如下所示：

```
r =
x^3 + 3*x^2 + 3*x + 2

how =
simplify

simplify:
x^3 + 3*x^2 + 3*x + 2

radsimp:
((x^3 - 1)*(x + 2))/(x - 1)

simplify(100):
(x + 2)*(x^2 + x + 1)

combine(sincos):
```

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

```
combine(sinhcosh):
```

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

```
combine(ln):
```

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

```
factor:
```

```
(x^2 + x + 1)*(x + 2)
```

```
expand:
```

```
(2*x^3)/(x - 1) - 2/(x - 1) - x/(x - 1) + x^4/(x - 1)
```

```
combine:
```

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

```
rewrite(exp):
```

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

```
rewrite(sincos):
```

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

```
rewrite(sinhcosh):
```

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

```
rewrite(tan):
```

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

```
mwcos2sin:
```

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

```
collect(x):
```

```
x^3 + 3*x^2 + 3*x + 2
```

```
ans =
```

```
x^3 + 3*x^2 + 3*x + 2
```

simple 函数的简化步骤包括执行下述函数。

- (1) collect 函数：合并同类项。
- (2) factor 函数：对表达式进行因式分解。
- (3) expand 函数：把表达式展开，并按降幂形式排列。
- (4) simplify 函数：对表达式进行简化。

(5) `radsimp` 函数：将表达式表示为复数形式。

(6) `convert` 函数：实现形式转换。

(7) `combine` 函数：对表达式以求和形式、乘积形式和幂形式出现的各项进行合并。

【例 3-32】通过对表达式 $e_1 = 2\cos^2 x - \sin^2 x$ 的化简，了解化简函数的巨大功能。
在命令窗口中输入如下语句：

```
syms x;  
e1=2*cos(x)^2-sin(x)^2;  
simple(e1)
```

命令窗口中的输出结果如下所示：

```
simplify:  
2 - 3*sin(x)^2  
  
radsimp:  
2*cos(x)^2 - sin(x)^2  
  
simplify(100):  
3*cos(x)^2 - 1  
  
combine(sincos):  
(3*cos(2*x))/2 + 1/2  
  
combine(sinhcosh):  
  
2*cos(x)^2 - sin(x)^2  
  
combine(ln):  
2*cos(x)^2 - sin(x)^2  
  
factor:  
2*cos(x)^2 - sin(x)^2  
  
expand:  
2*cos(x)^2 - sin(x)^2  
  
combine:  
2*cos(x)^2 - sin(x)^2  
  
rewrite(exp):  
2*((1/exp(x*i))/2 + exp(x*i)/2)^2 - (1/2*i*exp(i*x) - 1/2*i*exp(-i*x))^2
```

```

rewrite(sincos):
2*cos(x)^2 - sin(x)^2

rewrite(sinhcosh):
2*cosh(-x*i)^2 + sinh(-x*i)^2

rewrite(tan):
(2*(tan(x/2)^2 - 1)^2)/(tan(x/2)^2 + 1)^2 - (4*tan(x/2)^2)/(tan(x/2)^2 + 1)^2

mwcos2sin:
2 - 3*sin(x)^2

collect(x):
2*cos(x)^2 - sin(x)^2

ans =
2 - 3*sin(x)^2

```

通过 `simple` 函数，分别对函数 $e_1=2\cos^2x-\sin^2x$ 运用了多种化简方法，所以它能够改善函数 `simplify` 给出的结果。特别值得一提的是，`simple` 函数对于含有三角函数的表达式很有效。

7. pretty 函数

MATLAB 提供了 `pretty` 函数以习惯的“书写”方式（有理分式）显示表达式，它的具体用法如下所示。

- `p=pretty(f)`: 将 `f` 转化为我们习惯的形式显示。

【例 3-33】使用 `pretty` 函数将符号表达式 $f = a*x/b + c/(d*y)$ 与 $\text{sqrt}(b^2-4*a*c)$ 以习惯的“书写”方式显示。

在命令窗口中输入如下语句：

```

syms a b c d x y;
f=a*x/b+c/(d*y);
f1=sqrt(b^2-4*a*c);
pretty(f)
pretty(f1)

```

命令窗口中的输出结果如下所示：

```

      c      a x
    --- + ---
    d y      b

      2      1/2
    (b  - 4 a c)

```

从上面的结果不难看出，结果显示为常用形式。

3.3 符号运算精度

数值计算将引入舍入误差，因为数值的精度常常受所设位数的限制，如果计算出的数值位数很多，已经超出设定，系统会把多出的位数舍去，造成数值计算的结果不精确。而符号运算不存在这个问题，符号计算中不会出现数值计算，便不会有舍入误差的存在。只有在符号运算结束，把结果转换成数值型的时候才会产生误差。

MATLAB 计算精度一共有浮点运算的数值算法、精确运算的符号算法和可控精度的算法 3 种方法。

1. 浮点运算的数值算法

浮点运算的数值算法是运算速度最快的算法，此算法在机器内部是以二进制进行存储的，因此这种算法的表达和计算结果都不是完整的值，而是近似值。

【例 3-34】浮点运算。

在命令窗口中输入如下语句：

```
syms a b c
a=1/5;
b=3/8;
c=a+b
```

命令窗口中的输出结果如下所示：

```
c =
    0.5750
```

浮点运算虽然运算速度快，但是精度稍差。上面例题中有 4 处引入了截断误差，第一处是 $1/5$ ，第二处是 $3/8$ ，第三处是在求和的过程中产生的，第四处是在由二进制转换成十进制的时候产生的。

2. 精确运算的符号算法

精确运算的符号算法与另外两种算法比较，占用较大的内存空间，运算速度慢，但它的运算结果精确。

【例 3-35】精确运算。

在命令窗口中输入如下语句：

```
a=sym(1/5+3/8)
b=sym(1/3+1/2)
```

命令窗口中的输出结果如下所示：

```
a =
    23/40

b =
    5/6
```

很明显，结果均没有产生误差，为精确计算。

3. 可控精度的算法

可控精度的算法是通过规定计算量的有效数字位数，来达到对运算精度控制的目的。有效数字位数的不同，直接影响运算的精度，MATLAB 提供了两种可控精度的算法，它们的具体

用法如下所示。

- **digits(n)**: 此函数规定参加运算的有效数字的位数, 如果没有明确指出则认为是默认值 32, 一般采取默认值时的运算精度和浮点数差不多。这条语句规定 MATLAB 在以后的运算中取 **n** 个有效数字。
- **vpa(s)**: **s** 为所要计算的表达式, 是在 **digits(n)** 的情况下进行可控精度的计算, 如果 **n** 不确定则认为是默认值 32。

【例 3-36】可控精度的计算。

在命令窗口中输入如下语句:

```
digits(4)
syms a b c
a=1/5+3/8;
b=2*pi;
c=1.256349;
d=sym(3/7);
f1=vpa(a+b)
f2=vpa(a+c)
f3=vpa(c+d)
f4=vpa(b+d)
f5=vpa(a+d)
```

命令窗口中的输出结果如下所示:

```
f1 =
6.858

f2 =
1.831

f3 =
1.685

f4 =
6.712

f5 =
1.004
```

【例 3-37】重新定义例 3-35 中的有效数字位数。

在命令窗口中输入如下语句:

```
digits(20)
f1=vpa(a+b)
f2=vpa(a+c)
f3=vpa(a+d)
f4=vpa(b+c)
```

```
f5=vpa(d+b)
```

命令窗口中的输出结果如下所示：

```
f1 =  
6.8581853071795864096  
  
f2 =  
1.831348999999998936  
  
f3 =  
1.6849204285714285095  
  
f4 =  
6.7117567357510150484  
  
f5 =  
1.0035714285714285714
```

3.4 符号矩阵的计算

由于 MATLAB 采取了重载的技术，使符号对象的矩阵运算在形式上与数值运算十分相似。

3.4.1 基本算术运算

基本的算术运算包括加、减、乘、除、矩阵乘方以及矩阵指数运算等，具体说明如下：

(1) 符号矩阵运算和一般矩阵运算要求一样，在计算矩阵加减的时候，要求这两个矩阵的大小一样，允许参与运算的两矩阵之一是标量，此时标量与矩阵的所有元素分别进行加减操作。

(2) 在计算矩阵相乘的时候要求前一矩阵的列数等于后一矩阵的行数，允许参与运算的两矩阵之一是标量，标量可与任何矩阵相乘。

(3) 矩阵相除的运算在线性代数中是没有的，在线性代数中只有矩阵逆的运算，但是在 MATLAB 中存在两矩阵相除的运算。

(4) 矩阵的乘方表示为 a^p 。其中 a 是指方阵，矩阵的乘方就是 a 自乘 p 次所得的数值。

对于 p 的其他值，计算将涉及特征值和特征向量。如果 p 是矩阵， a 是标量， a^p 使用特征值和特征向量自乘到 p 次幂；如果 a 、 p 都是矩阵，则 a^p 无意义。

(5) 符号矩阵的指数运算由函数 `exp()` 来实现。

【例 3-38】符号矩阵的基本算术运算。

在命令窗口中输入如下语句：

```
A = sym(['a 2*b;1 5*c'])  
B = sym(['2*x 3*b; a m'])  
  
A+B  
A-B  
A*B  
A/B
```

命令窗口中的输出结果如下所示:

```
A =  
[ a, 2*b]  
[ 1, 5*c]  
  
B =  
[ 2*x, 3*b]  
[  a,   m]  
  
ans =  
[ a + 2*x,      5*b]  
[  a + 1, 5*c + m]  
  
ans =  
[ a - 2*x,      -b]  
[  1 - a, 5*c - m]  
  
ans =  
[ 2*a*b + 2*a*x, 3*a*b + 2*b*m]  
[  2*x + 5*a*c,   3*b + 5*c*m]  
  
ans =  
[ (a*(2*b - m))/(3*a*b - 2*m*x), (b*(3*a - 4*x))/(3*a*b - 2*m*x)]  
[ -(m - 5*a*c)/(3*a*b - 2*m*x), (3*b - 10*c*x)/(3*a*b - 2*m*x)]
```

【例 3-39】计算矩阵的 3 次方。

在命令窗口中输入如下语句:

```
A=[1 2 3;4 5 6;7 8 9]  
a=sym(A)      %将数值矩阵转换成为符号矩阵  
a^3
```

命令窗口中的输出结果如下所示:

```
A =  
     1     2     3  
     4     5     6  
     7     8     9  
  
a =  
[ 1, 2, 3]  
[ 4, 5, 6]  
[ 7, 8, 9]  
  
ans =  
[ 468, 576, 684]
```

```
[ 1062, 1305, 1548]
[ 1656, 2034, 2412]
```

当一个方阵有复数特征值或负实特征值时，则非整数幂是复数阵。

【例 3-40】计算符号矩阵的幂。

在命令窗口中输入如下语句：

```
A=sym('[a1 a2;a3 a4]');
b=3;
C1=A^b
C2=A*A*A
```

命令窗口中的输出结果如下所示：

```
C1 =
[ a1*(a1^2 + a2*a3) + a2*(a1*a3 + a3*a4), a2*(a4^2 + a2*a3) + a1*(a1*a2 + a2*a4)]
[ a3*(a1^2 + a2*a3) + a4*(a1*a3 + a3*a4), a4*(a4^2 + a2*a3) + a3*(a1*a2 + a2*a4)]

C2 =
[ a1*(a1^2 + a2*a3) + a3*(a1*a2 + a2*a4), a2*(a1^2 + a2*a3) + a4*(a1*a2 + a2*a4)]
[ a3*(a4^2 + a2*a3) + a1*(a1*a3 + a3*a4), a4*(a4^2 + a2*a3) + a2*(a1*a3 + a3*a4)]
```

【例 3-41】计算符号矩阵的指数。

在命令窗口中输入如下语句：

```
A=sym('[a1 a2;a3 a4]');
B=exp(A)
```

命令窗口中的输出结果如下所示：

```
B =
[ exp(a1), exp(a2)]
[ exp(a3), exp(a4)]
```

由运算结果可知，符号矩阵的指数运算的规则是得到一个与原矩阵行列数相同的矩阵，而以 e 为底、以矩阵的每一个元素作为指数进行运算的结果作为新矩阵的对应元素。

3.4.2 线性代数运算

符号矩阵的线性代数运算和矩阵的代数运算是相同的，具体运算如表 3-2 所示。

表 3-2 符号矩阵的线性代数运算

函数名称	功能介绍
inv	矩阵求逆
det	计算行列式的值
diag	对角矩阵
triu	抽取矩阵的上三角部分
tril	抽取矩阵的下三角部分
rank	计算矩阵的秩
rref	返回矩阵的缩减行阶梯矩阵
null	零空间的正交基
colspace	返回矩阵列空间的基
transpose	返回矩阵的转置

1. inv 函数

MATLAB 提供了 `inv` 函数用于计算符号矩阵的逆，它的具体用法如下所示。

- `inv(A)`: 计算符号矩阵的逆。

【例 3-42】计算随机矩阵的逆。

在命令窗口中输入如下语句：

```
A=rand(4)
inv(A)
```

命令窗口中的输出结果如下所示：

```
A =
    0.7577    0.1712    0.0462    0.3171
    0.7431    0.7060    0.0971    0.9502
    0.3922    0.0318    0.8235    0.0344
    0.6555    0.2769    0.6948    0.4387

ans =
    2.0990    0.3865    1.9524   -2.5074
    0.8852    8.1778   15.4969  -19.5678
   -0.9419   -0.2786    0.2319    1.2659
   -2.2030   -5.2978  -13.0653   16.3710
```

2. det 函数

MATLAB 提供了 `det` 函数用于计算符号矩阵的行列式，它的具体用法如下所示。

- `det(A)`: 计算矩阵 A 的行列式。

【例 3-43】使用 `det` 函数计算符号矩阵的行列式。

在命令窗口中输入如下语句：

```
A=magic(4); %生成三阶魔方数值矩阵
B=sym(A); %将数值矩阵转换成为符号矩阵
det(B)
```

命令窗口中的输出结果如下所示：

```
ans =
0
```

矩阵求逆和求行列式采用的都是符号计算，得到的结果都是精确解。

3. diag 函数

MATLAB 提供了 `diag` 函数用于实现对符号矩阵对角线元素的操作，它的具体用法如下所示。

- `diag(v,k)`: 若 v 是由 n 个元素组成的矢量，则结果是 $n+abs(k)$ 阶的方阵。当 $k=0$ 时，将矢量 v 置于主对角线上；当 $k<0$ 时，将矢量 v 置于主对角线之下；当 $k>0$ 时，将矢量 v 置于主对角线之上。
- `diag(v)`: 与 $k=0$ 相同，将矢量 v 置于主对角线上。
- `diag(A,k)`: A 是矩阵，结果是由矩阵 A 的第 k 条对角线上的元素组成的列矢量。
- `diag(A)`: A 是矩阵，此用法是 `diag(A,k)` 用法中 $k=0$ 的情况，结果是由矩阵 A 的主对角线元素组成的列矢量。

【例 3-44】如 v 是由 4 个元素组成的矢量，利用 `diag` 函数求符号矩阵的对角线。
在命令窗口中输入如下语句：

```
syms a b c
v=[1 a+b c 2];
diag(v,1)
```

命令窗口中的输出结果如下所示：

```
ans =
[ 0, 1,      0, 0, 0]
[ 0, 0,  a + b, 0, 0]
[ 0, 0,      0, c, 0]
[ 0, 0,      0, 0, 2]
[ 0, 0,      0, 0, 0]
```

从上面的结果不难看出，返回的矩阵是 $4+1=5$ 阶的方阵。因为当 $k>0$ 时，将矢量 v 置于主对角线之上。

【例 3-45】使用 `diag(v)` 将矢量 v 置于主对角线上。

在命令窗口中输入如下语句：

```
syms a b c
v=[1 a+b c 2];
diag(v)
```

命令窗口中的输出结果如下所示：

```
ans =
[ 1,      0, 0, 0]
[ 0,  a + b, 0, 0]
[ 0,      0, c, 0]
[ 0,      0, 0, 2]
```

【例 3-46】若 A 是魔方矩阵，分别找出由矩阵 A 的第 1、2、3、4 条对角线上的元素组成的列矢量。

在命令窗口中输入如下语句：

```
s=sym(4);    %定义了符号常量
class(s)     %显示定义的类型
A=magic(s)
diag(A,3)
diag(A,2)
diag(A,1)
diag(A,0)
```

命令窗口中的输出结果如下所示：

```
ans =
sym

A =
```

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

```
ans =
```

```
13
```

```
ans =
```

```
3
```

```
8
```

```
ans =
```

```
2
```

```
10
```

```
12
```

```
ans =
```

```
16
```

```
11
```

```
6
```

```
1
```

【例 3-47】利用 `diag(A)` 找出矩阵 `A` 主对角线上元素的列矢量。

在命令窗口中输入如下语句：

```
s=sym(4);
class(s)
A=magic(s)
diag(A)
```

命令窗口中的输出结果如下所示：

```
ans =
```

```
sym
```

```
A =
```

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

```
ans =
```

```
16
```

```
11
```

```
6
```


4. triu 函数

MATLAB 提供了 `triu` 函数生成一个新矩阵，该新矩阵是抽取原矩阵的上三角部分，其他部分用 0 来填充的矩阵，它的具体用法如下所示。

- `triu(A)`: 抽取矩阵 `A` 主对角线上的三角部分重新组成一个新矩阵，其他部分用 0 来填充。
- `triu(A,k)`: 抽取矩阵 `A` 的第 `k` 条对角线上的部分重新组成一个新矩阵，其他部分用 0 来填充。当 `k>0` 时，抽取的元素是在主对角线上且在 `k` 条对角线上的元素，其他部分用 0 来填充；当 `k<0` 时，抽取的元素是在主对角线下且在 `k` 条对角线上的元素，其他部分用 0 来填充。当 `k=0` 时，`triu(A,0)` 同 `triu(A)` 一样，抽取主对角线上的部分。

【例 3-48】若 `A` 是魔方矩阵，利用 `triu` 函数生成一个由矩阵 `A` 主对角线上的元素组成的矩阵。

在命令窗口中输入如下语句：

```
A=magic(4)      %生成魔方数值矩阵
B=sym(A)        %将数值矩阵转换成为符号矩阵
triu(B)
```

命令窗口中的输出结果如下所示：

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

B =
[ 16,  2,  3, 13]
[  5, 11, 10,  8]
[  9,  7,  6, 12]
[  4, 14, 15,  1]

ans =
[ 16,  2,  3, 13]
[  0, 11, 10,  8]
[  0,  0,  6, 12]
[  0,  0,  0,  1]
```

【例 3-49】若 `A` 是魔方矩阵，利用 `triu` 函数生成由矩阵 `A` 主对角线上的元素组成的矩阵，以及由矩阵 `A` 主对角线下的元素组成的矩阵和由矩阵 `A` 主对角线的元素组成的矩阵，对比它们的不同。

在命令窗口中输入如下语句：

```
A=magic(4)      %生成魔方数值矩阵
B=sym(A)        %将数值矩阵转换成为符号矩阵
triu(B,2)
```

```
triu(B,-3)
```

```
triu(B,0)
```

命令窗口中的输出结果如下所示:

```
A =
```

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

```
B =
```

```
[ 16,  2,  3, 13]
[  5, 11, 10,  8]
[  9,  7,  6, 12]
[  4, 14, 15,  1]
```

```
ans =
```

```
[ 0, 0, 3, 13]
[ 0, 0, 0,  8]
[ 0, 0, 0,  0]
[ 0, 0, 0,  0]
```

```
ans =
```

```
[ 16,  2,  3, 13]
[  5, 11, 10,  8]
[  9,  7,  6, 12]
[  4, 14, 15,  1]
```

```
ans =
```

```
[ 16,  2,  3, 13]
[  0, 11, 10,  8]
[  0,  0,  6, 12]
[  0,  0,  0,  1]
```

【例 3-50】利用 `triu` 函数生成由符号矩阵 **A** 主对角线上的元素组成的矩阵, 以及由符号矩阵 **A** 主对角线下的元素组成的矩阵和由符号矩阵 **A** 主对角线的元素组成的矩阵, 对比它们的不同。

在命令窗口中输入如下语句:

```
syms a b c
```

```
A=[a^2 b+c 6 exp(c); a+b b a 5 ;4 b c 1;a^b c a 8]
```

```
triu(A)
```

```
triu(A,1)
```

```
triu(A,-1)
```

命令窗口中的输出结果如下所示：

```
A =
[ a^2, b + c, 6, exp(c)]
[ a + b, b, a, 5]
[ 4, b, c, 1]
[ a^b, c, a, 8]

ans =
[ a^2, b + c, 6, exp(c)]
[ 0, b, a, 5]
[ 0, 0, c, 1]
[ 0, 0, 0, 8]

ans =
[ 0, b + c, 6, exp(c)]
[ 0, 0, a, 5]
[ 0, 0, 0, 1]
[ 0, 0, 0, 0]

ans =
[ a^2, b + c, 6, exp(c)]
[ a + b, b, a, 5]
[ 0, b, c, 1]
[ 0, 0, a, 8]
```

从本例可以看出，`triu(A,0)` 等价于 `triu(A)`，即生成由主对角线上的元素组成的矩阵。

5. tril 函数

MATLAB 提供了 `tril` 函数生成一个新矩阵，该新矩阵是抽取原矩阵的下三角部分，其他部分用 0 来填充的矩阵，它的具体用法如下所示。

- `tril(A)`: 抽取矩阵 A 的主对角线下的三角部分重新组成一个新矩阵，其他部分用 0 来填充。
- `tril(A,k)`: 抽取矩阵 A 的第 k 条对角线下的部分重新组成一个新矩阵，其他部分用 0 来填充。当 $k > 0$ 时，抽取的元素是在主对角线上且 k 条对角线下的元素，其他部分用 0 来填充；当 $k < 0$ 时，抽取元素是在主对角线下且 k 条对角线下的元素，其他部分用 0 来填充。当 $k = 0$ 时，`triu(A,0)` 同 `triu(A)` 一样，抽取主对角线下的部分。

【例 3-51】利用 `tril` 函数生成由希尔伯特矩阵 H 主对角线下的元素组成的矩阵。

在命令窗口中输入如下语句：

```
H= hilb(3)      %生成 3 阶希尔伯特数值矩阵
B= sym(H)       %将数值矩阵转换成为符号矩阵
tril(B)
```

命令窗口中的输出结果如下所示：

```
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

```
B =
[ 1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

```
ans =
[ 1, 0, 0]
[ 1/2, 1/3, 0]
[ 1/3, 1/4, 1/5]
```

【例 3-52】利用 `tril` 函数生成由符号矩阵 `A` 主对角线下的元素组成的矩阵。

在命令窗口中输入如下语句：

```
syms a b c
A=[a^2 b+c 6 exp(c); a+b b a 5 ;4 b c 1;a^b c a 8]
tril(A)
```

命令窗口中的输出结果如下所示：

```
A =
[ a^2, b + c, 6, exp(c)]
[ a + b, b, a, 5]
[ 4, b, c, 1]
[ a^b, c, a, 8]
```

```
ans =
[ a^2, 0, 0, 0]
[ a + b, b, 0, 0]
[ 4, b, c, 0]
[ a^b, c, a, 8]
```

【例 3-53】利用 `tril` 函数生成由符号矩阵 `A` 主对角线上的元素组成的矩阵，以及由符号矩阵 `A` 主对角线下的元素组成的矩阵和由符号矩阵 `A` 主对角线的元素组成的矩阵。

在命令窗口中输入如下语句：

```
H= hilb(3)
B= sym(H)
tril(B,1)
tril(B,2)
tril(B,-1)
```

```
tril(B,-2)
```

```
tril(B,0)
```

命令窗口中的输出结果如下所示：

```
H =
```

```
    1.0000    0.5000    0.3333  
    0.5000    0.3333    0.2500  
    0.3333    0.2500    0.2000
```

```
B =
```

```
[ 1, 1/2, 0]  
[ 1/2, 1/3, 1/4]  
[ 1/3, 1/4, 1/5]
```

```
ans =
```

```
[ 1, 1/2, 1/3]  
[ 1/2, 1/3, 1/4]  
[ 1/3, 1/4, 1/5]
```

```
ans =
```

```
[ 1, 1/2, 1/3]  
[ 1/2, 1/3, 1/4]  
[ 1/3, 1/4, 1/5]
```

```
ans =
```

```
[ 0, 0, 0]  
[ 1/2, 0, 0]  
[ 1/3, 1/4, 0]
```

```
ans =
```

```
[ 0, 0, 0]  
[ 0, 0, 0]  
[ 1/3, 0, 0]
```

```
ans =
```

```
[ 1, 0, 0]  
[ 1/2, 1/3, 0]  
[ 1/3, 1/4, 1/5]
```

【例 3-54】 利用 `tril` 函数生成由符号矩阵 **A** 主对角线上的元素组成的矩阵，以及由 **A** 主对角线下的元素组成的矩阵和由符号矩阵 **A** 主对角线的元素组成的矩阵。

在命令窗口中输入如下语句：

```
syms a b c
A=[c a 1 2 ;6 b a 3 ;5 4 b c;6 c b a ]
tril(A,1)
tril(A,3)
tril(A,-1)
    tril(A,-3)
tril(A,0)
```

命令窗口中的输出结果如下所示:

```
A =
[ c, a, 1, 2]
[ 6, b, a, 3]
[ 5, 4, b, c]
[ 6, c, b, a]
```

```
ans =
[ c, a, 0, 0]
[ 6, b, a, 0]
[ 5, 4, b, c]
[ 6, c, b, a]
```

```
ans =
[ c, a, 1, 2]
[ 6, b, a, 3]
[ 5, 4, b, c]
[ 6, c, b, a]
```

```
ans =
[ 0, 0, 0, 0]
[ 6, 0, 0, 0]
[ 5, 4, 0, 0]
[ 6, c, b, 0]
```

```
ans =
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 6, 0, 0, 0]
```

```
ans =
[ c, 0, 0, 0]
```

```
[ 6, b, 0, 0]
```

```
[ 5, 4, b, 0]
```

```
[ 6, c, b, a]
```

从本例可以看出，调用 `tril(A,0)` 生成由主对角线上的元素组成的矩阵，和调用 `tril(A)` 的作用相同。

6. rank 函数

矩阵 **A** 中线性无关的列向量个数称为列秩，线性无关的行向量个数称为行秩。可以证明矩阵的列秩和行秩是相等的。矩阵秩的求法很多，其中有些算法是稳定的，有些算法是不稳定的。**MATLAB** 提供了 `rank` 函数计算符号矩阵的秩，采用的算法是基于矩阵奇异值分解基础上的。它的具体用法如下所示。

- `rank(A)`: 返回矩阵 **A** 的秩。

【例 3-55】使用 `rank` 函数计算 4×4 希尔伯特矩阵的秩。

在命令窗口中输入如下语句：

```
H=hilb(4);
```

```
B=sym(H);
```

```
rank(B)
```

命令窗口中的输出结果如下所示：

```
ans =
```

```
4
```

从本例可以看出，希尔伯特矩阵为满秩矩阵。

【例 3-56】使用 `rank` 函数计算符号矩阵的秩。

在命令窗口中输入如下语句：

```
syms a b c
```

```
A=[c a 1 2;6 b a 3;5 4 b c;0 0 0 0]
```

```
rank(A)
```

命令窗口中的输出结果如下所示：

```
A =
```

```
[ c, a, 1, 2]
```

```
[ 6, b, a, 3]
```

```
[ 5, 4, b, c]
```

```
[ 0, 0, 0, 0]
```

```
ans =
```

```
3
```

7. rref 函数

矩阵的简化行阶梯形式是高斯-约旦消元法解线性方程组的结果，其形式为

$$\begin{pmatrix} 1 & \dots & 0 & * \\ \vdots & \ddots & \vdots & * \\ 0 & \dots & 1 & * \end{pmatrix}$$

MATLAB 提供了 `rref` 函数返回符号矩阵的简化行阶梯矩阵，它的具体用法如下所示。

- `R=rref(A)`: 在计算的过程中利用高斯-约当消元法和行主元素法，返回矩阵的简化行阶梯矩阵 `R`。
- `[R,jb]=rref(A)`: 返回矩阵的简化行阶梯矩阵 `R` 和矢量 `jb`。`R(1:r,jb)` 为 $r \times r$ 阶不确定矩阵，矩阵 `A` 的秩为 `r=length(jb)`，`x(jb)` 为线性方程组 $Ax=b$ 的边界向量。
- `[R,jb]=rref(A,tol)`: 返回矩阵的简化行阶梯矩阵 `R` 和矢量 `jb`，要求和以上提到的相同，`tol` 指明了返回矩阵元素的误差。

【例 3-57】 使用 `rref` 函数返回符号矩阵 `A` 的简化行阶梯矩阵。

在命令窗口中输入如下语句：

```
syms a b c
A=[a^2 b+c 6 exp(c); a+b b a 5 ;4 b c 1;a^b c a 8]
R=rref(A)
```

命令窗口中的输出结果如下所示：

```
A =
[ a^2, b + c, 6, exp(c)]
[ a + b, b, a, 5]
[ 4, b, c, 1]
[ a^b, c, a, 8]

R =
[ 1, 0, 0, 0]
[ 0, 1, 0, 0]
[ 0, 0, 1, 0]
[ 0, 0, 0, 1]
```

8. null 函数

MATLAB 提供了 `null` 函数计算零空间的正交基，它的具体用法如下所示。

- `N=null(A)`: 计算矩阵 `A` 的零空间的正交基，运算依赖矩阵 `A` 的奇异值分解。
- `N=null(A,'r')`: 计算矩阵 `A` 的零空间的正交基，运算依赖矩阵 `A` 的简化行阶梯矩阵。

【例 3-58】 使用 `null` 函数返回符号矩阵 `A` 的零空间正交基。

在命令窗口中输入如下语句：

```
B=magic(4);
A=sym(B)
N=null(A)
```

命令窗口中的输出结果如下所示：

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



```
N =
-1
-3
3
1
```

9. colspace 函数

MATLAB 提供了 `colspace` 函数计算矩阵列空间的基，它的具体用法如下所示。

- `C=colspace(A)`: 返回符号矩阵 `A` 的列空间的基。

【例 3-59】使用 `colspace` 函数计算符号矩阵 `A` 的列空间的基。

在命令窗口中输入如下语句：

```
A=rand(5);
B=sym(A);
C=colspace(B)
```

命令窗口中的输出结果如下所示：

```
C =
[ 1, 0, 0, 0, 0]
[ 0, 1, 0, 0, 0]
[ 0, 0, 1, 0, 0]
[ 0, 0, 0, 1, 0]
[ 0, 0, 0, 0, 1]
```

10. transpose 函数

MATLAB 提供了 `transpose` 函数计算矩阵的转置，它的具体用法如下所示。

- `T=transpose(A)`: 返回符号矩阵 `A` 的转置。

【例 3-60】计算符号矩阵的转置。

在命令窗口中输入如下语句：

```
syms a b c d
A=[a^2 b+c 6 exp(c); a+b b a 5 ;4 b c 1;a^b c a 8]
T=transpose(A)
```

命令窗口中的输出结果如下所示：

```
A =
[ a^2, b + c, 6, exp(c)]
[ a + b, b, a, 5]
[ 4, b, c, 1]
[ a^b, c, a, 8]

T =

[ a^2, a + b, 4, a^b]
[ b + c, b, b, c]
[ 6, a, c, a]
```

```
[ exp(c), 5, 1, 8]
```

这里需要说明的是, $\text{transpose}(A)$ 与 A' 的结果存在差别, $\text{transpose}(A)$ 是计算转置, A' 是计算共扼转置。在命令窗口中输入如下语句:

```
syms a b c d
A=[a^2 b+c 6 exp(c); a+b b a 5 ;4 b c 1;a^b c a 8]
C= A'
```

命令窗口中的输出结果如下所示:

```
C =
[ conj(a)^2, conj(a) + conj(b), 4, conj(a^b)]
[ conj(b) + conj(c), conj(b), conj(b), conj(c)]
[ 6, conj(a), conj(c), conj(a)]
[ exp(conj(c)), 5, 1, 8]
```

其中 conj 函数用于计算共扼。

11. 特征值分解

MATLAB 提供了 eig 函数进行特征值分解, 即计算矩阵的特征值和特征向量, 它的具体用法如下所示。

- $E=\text{eig}(A)$: 返回由方阵 A 的特征值组成的矩阵。
- $[V,D]=\text{eig}(A)$: 返回方阵 A 的特征值矩阵 D 和特征矢量矩阵 V , 其中特征值矩阵 D 是由 A 的特征值为对角线组成的对角矩阵, V 、 D 和 A 之间满足 $AV=VD$ 。

【例 3-61】利用 eig 函数计算 4 阶魔方矩阵的特征值和特征向量。

在命令窗口中输入如下语句:

```
A=magic(4);
E=eig(A)
[V,D]=eig(A)
```

命令窗口中的输出结果如下所示:

```
E =
34.0000
8.9443
-8.9443
0.0000

V =
-0.5000 -0.8236 0.3764 -0.2236
-0.5000 0.4236 0.0236 -0.6708
-0.5000 0.0236 0.4236 0.6708
-0.5000 0.3764 -0.8236 0.2236

D =
34.0000 0 0 0
0 8.9443 0 0
```

```

0      0    -8.9443      0
0      0      0     0.0000

```

12. 约当标准型

MATLAB 提供了 `jordan` 函数将矩阵变换为约当标准型，计算约当标准型也就是找一个非奇异矩阵 V ，使 $J = V/A * V$ 最接近对角矩阵，其中 V 称为转换矩阵。利用矩阵分块可以简化很多有关矩阵的证明和计算，任何方阵都可以通过相似变换，变为约当标准型。`jordan` 函数的具体用法如下所示：

- $J=jordan(A)$ ：返回矩阵 A 的约当标准型。
- $[V,J]=jordan(A)$ ：返回矩阵 A 的约当标准型，并且给出变换矩阵 V ，满足 $J = V/A * V$ 。

【例 3-62】利用 `jordan` 函数将 4 阶魔方矩阵 A 划为约当标准型。

在命令窗口中输入如下语句：

```

A=magic(4);
[V,J]=jordan(A)

```

命令窗口中的输出结果如下所示：

```

V =
-1.0000    1.0000   -2.1882   -0.4570
-3.0000    1.0000    1.1254   -0.0287
 3.0000    1.0000    0.0627   -0.5143
 1.0000    1.0000    1.0000    1.0000

J =
      0      0      0      0
      0 34.0000      0      0
      0      0   8.9443      0
      0      0      0  -8.9443

```

13. 奇异值分解

奇异值分解在矩阵分解中占有极其重要的地位，即对于 $m \times n$ 的矩阵 A ，若存在 $n \times m$ 的西矩阵 U 和 $n \times m$ 的西矩阵 V 使得 $A=U * \Sigma * V'$ ，其中 Σ 为一个 $m \times n$ 的非负对角矩阵，并且其对角元素的值按降序排列，则 $A=U * \Sigma * V'$ 即为矩阵 A 的奇异值分解， U 、 Σ 和 V 称为矩阵 A 的奇异值分解的三对组。

MATLAB 提供了 `svd` 函数对矩阵 A 进行奇异值分解。由于符号计算产生的公式一般都很长而且复杂，所以计算指定精度矩阵的奇异值分解才是有意义的。

`svd` 函数的具体用法如下所示。

- $S=svd(A)$ ：返回符号矩阵的奇异值对角矩阵，计算精度由函数 `digits` 来指定。
- $[U,S,V]=svd(A)$ ：输出的矩阵 U 和 V 是正交矩阵，并且它们满足关系式： $A = U * S * V'$ 。

【例 3-63】利用 $[U,S,V]=svd(A)$ 函数对符号矩阵 A 进行奇异值分解。

在命令窗口中输入如下语句：

```

A=[1 2 3;4 5 6;7 8 9]
digits(30);
[U,S,V]=svd(A)

```

命令窗口中的输出结果如下所示：

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

U =
    -0.2148    0.8872    0.4082
    -0.5206    0.2496   -0.8165
    -0.8263   -0.3879    0.4082

S =
    16.8481         0         0
         0     1.0684         0
         0         0     0.0000

V =
    -0.4797   -0.7767   -0.4082
    -0.5724   -0.0757    0.8165
    -0.6651    0.6253   -0.4082
```

3.4.3 科学计算

极限、微分和积分是微积分学的核心和基础，MATLAB 提供了专门的函数来支持微积分运算。

1. 符号极限的计算

极限理论即“无穷逼近”是高等数学的出发点和基础，高等数学的许多内容都建立在极限理论的基础上，如上面讲到的求微分，它的基本思想就是当自变量趋近某个值时，计算函数值的变化。MATLAB 提供了 `limit` 函数计算符号表达式的极限，它的具体用法如下所示。

- `limit(f,x,a)`：当变量 x 趋近于常数 a 时，计算符号函数 $f(x)$ 的极限值。
- `limit(f,a)`：相当于变量 x 趋近于 a 时，计算符号函数 $f(x)$ 的极限值。在没有指定符号函数 $f(x)$ 的自变量时，使用此函数来计算符号函数的极限，系统按 `findsym` 函数指示的默认变量来确定符号函数 $f(x)$ 的变量。
- `limit(f)`：在没有指定变量的目标值时，系统默认变量趋近于 0，相当于变量 x 趋近于 0 时，计算符号函数 $f(x)$ 的极限值，系统按 `findsym` 函数指示的默认变量来确定符号函数 $f(x)$ 的变量。
- `limit(f,x,a,'right')` 和 `limit(f,x,a,'left')`：由于求极限可以从两边趋近，对于某些函数从左面趋近和从右面趋近得到的结果是不同的，针对这种情况，`limit` 函数专门提供了本语句来计算函数的左极限和右极限。`'right'` 表示符号函数 $f(x)$ 的右极限，即变量 x 从右边趋近于 a ，`'left'` 表示符号函数 $f(x)$ 的左极限，即变量 x 从左边趋近于 a 。

【例 3-64】计算的极限。 $\lim_{x \rightarrow \infty} \left(\frac{2x+3}{2x+1} \right)^{x+1}$

在命令窗口中输入如下语句：

```
syms x
f=((2*x+3)/(2*x+1))^(x+1);
limit(f,x,inf)
```

命令窗口中的输出结果如下所示：

```
ans =
exp(1)
```

【例 3-65】计算 $\lim_{x \rightarrow 1} x[e^{\sin(x)} + 1] - 2[e^{\tan(x)} - 1]$ 的极限。

在命令窗口中输入如下语句：

```
syms x;
f=x*(exp(sin(x))+1)-2*(exp(tan(x))-1);
limit(f,x,1)
```

命令窗口中的输出结果如下所示：

```
ans =
3-2*exp(sin(1)/cos(1))+exp(sin(1))
```

【例 3-66】计算 $\lim_{x \rightarrow a} \frac{\sin x - \sin a}{x - a}$ 的极限。

在命令窗口中输入如下语句：

```
syms x a;
f=(sin(x)-sin(a))/(x-a);
limit(f,a)
```

命令窗口中的输出结果如下所示：

```
ans =
cos(a)
```

【例 3-67】计算 $\lim_{x \rightarrow 0} \frac{\tan(2x)}{\sin(5x)}$ 的极限。

在命令窗口中输入如下语句：

```
syms x
f=tan(2*x)/sin(5*x);
limit(f)
```

命令窗口中的输出结果如下所示：

```
ans =
2/5
```

【例 3-68】计算 $f = \lim_{x \rightarrow 0^+} \frac{\sqrt{x} - \sqrt{3} + \sqrt{x-3}}{\sqrt{x^2 - 9}}$ 的极限。

在命令窗口中输入如下语句：

```
syms x;
```

```
f=(sqrt(x)-sqrt(3)-sqrt(x-3))/sqrt(x*x-9);
limit(f,x,3,'right')
```

命令窗口中的输出结果如下所示:

```
ans =
-6^(1/2)/6
```

【例 3-69】对于上例的函数 $f = \lim_{x \rightarrow 0^+} \frac{\sqrt{x} - \sqrt{3} - \sqrt{x-3}}{\sqrt{x^2-9}}$, 计算从左边趋近于零的值。

在命令窗口中输入如下语句:

```
syms x;
f=(sqrt(x)-sqrt(2)-sqrt(x-2))/sqrt(x*x-4);
limit(f,x,3,'left')
```

命令窗口中的输出结果如下所示:

```
ans =
-6^(1/2)/6
```

从例 3-68 和例 3-69 可以看出, 由于函数的左右极限都存在并且相等, 说明函数 $f = \frac{\sqrt{x} - \sqrt{3} - \sqrt{x-3}}{\sqrt{x^2-9}}$ 存在极限。

下面的例子说明了另一种情况。

【例 3-70】计算函数 $f = \frac{1}{x^3}$, 当 x 趋近于零时的极限值, 并分别求出函数的左极限和右极限。

在命令窗口中输入如下语句:

```
syms x;
f=1/x^3;
limit(f,x,0)
limit(f,x,0,'right')
limit(f,x,0,'left')
```

命令窗口中的输出结果如下所示:

```
ans =
NaN

ans =
Inf

ans =
-Inf
```

从这个例子可以看出函数 $f = \frac{1}{x^3}$ 的左右极限都存在, 但是左右极限不相等, 说明该函数的极限不存在, `ans =NaN` 语句说明该函数的极限不存在。需要注意的是, MATLAB 对于没有定义的极限均返回 NaN。

2. 符号微分的计算

微分运算是高等数学中除极限运算外的最重要的基本内容。

MATLAB 的符号微分运算，实际上是计算函数的导（函）数。MATLAB 提供了 `diff` 函数计算符号表达式的微分，它的具体用法如下所示。

- `diff(s)`: 没有指定变量和导数阶数，则系统按 `findsym` 函数指定的默认变量对符号表达式 `s` 求一阶导数。
- `diff(s,'v')`: 以 `v` 为自变量，对符号表达式 `s` 求一阶微分。
- `diff(s,n)`: 按 `findsym` 函数指示的默认变量对符号表达式 `s` 求 `n` 阶微分，且 `n` 为正整数。
- `diff(s,'v',n)`: 以 `v` 为自变量，对符号表达式 `s` 求 `n` 阶微分，`n` 为正整数。

【例 3-71】计算 $f = ax^2 + bx + c$ 函数关于 x 的微分。

在命令窗口中输入如下语句：

```
f=sym('a*x^2+b*x+c');  
df=diff(f)
```

命令窗口中的输出结果如下所示：

```
df =  
2*a*x+b
```

【例 3-72】计算 $f = \log(3x^2 + 5x + 1)$ 函数关于 x 的微分。

在命令窗口中输入如下语句：

```
f=sym('log(3*x^2+5*x+1)');  
df=diff(f)
```

命令窗口中的输出结果如下所示：

```
df =  
(6*x + 5)/(3*x^2 + 5*x + 1)
```

【例 3-73】计算 $f = 5x^4 + 6x^2 + \tan(x) + \sin^2(x)$ 函数关于 x 的微分。

在命令窗口中输入如下语句：

```
f=sym('5*x^4+6*x^2+tan(x)+sin(x)^2');  
df=diff(f)
```

命令窗口中的输出结果如下所示：

```
df =  
12*x + 2*cos(x)*sin(x) + tan(x)^2 + 20*x^3 + 1
```

【例 3-74】计算 $f = x^3 + a \tan(x) + b \sin^2(x)$ 函数关于 a 的微分。

在命令窗口中输入如下语句：

```
f=sym('x^3+a*tan(x)+b*sin(x)^2');  
df=diff(f,'a')
```

命令窗口中的输出结果如下所示：

```
df =  
tan(x)
```

【例 3-75】计算上例函数关于 b 的微分。

在命令窗口中输入如下语句：

```
f=sym('x^3+a*tan(x)+b*sin(x)^2');
df=diff(f,'b')
```

命令窗口中的输出结果如下所示:

```
df =
sin(x)^2
```

【例 3-76】计算 $f = \sin(x) \cos^2(x) + \tan^2(x) + 1$ 函数关于 x 的二阶微分。

在命令窗口中输入如下语句:

```
f=sym('sin(x)*cos(x)^2+tan(x)^2+1');
df=diff(f,2)
simplify(f) %使用 simplify 函数进行化简
```

命令窗口中的输出结果如下所示:

```
df =
2*(tan(x)^2 + 1)^2 + 2*sin(x)^3 + 4*tan(x)^2*(tan(x)^2 + 1) - 7*cos(x)^2*sin(x)

ans =
- 1/(sin(x)^2 - 1) - sin(x)*(sin(x)^2 - 1)
```

【例 3-77】计算 $f = a^4 \log(a) \tan(x) + a \sin^2(x) + x$ 函数关于 x 的三阶微分。

在命令窗口中输入如下语句:

```
f=sym('a^4*log(a)*tan(x)+a*sin(x)^2+x');
df=diff(f,3)
```

命令窗口中的输出结果如下所示:

```
df =
2*a^4*log(a)*(tan(x)^2 + 1)^2 - 8*a*cos(x)*sin(x) + 4*a^4*log(a)*tan(x)^2*(tan(x)^2 + 1)
```

【例 3-78】计算 $f = a^4 \log(a) \tan(x) + a \sin^2(x) + 2a \sin(x)$ 函数关于 a 的三阶微分。

在命令窗口中输入如下语句:

```
f=sym('a^4*log(a)*tan(x)+a*sin(x)^2+2*a*sin(x)');
diff(f,'a',3)
```

命令窗口中的输出结果如下所示:

```
ans =
26*a*tan(x) + 24*a*log(a)*tan(x)
```

3. 符号积分的计算

积分运算是微分运算的逆运算,即由已知导数求原函数的过程。函数的积分有不定积分与定积分两种。定积分中,若是积分区间为无穷或被积函数在积分区间上有无穷不连续点,但积分存在或收敛者,叫做广义积分。MATLAB 提供了 `int` 函数计算符号表达式的积分。该函数既可以计算定积分,也可以计算不定积分和广义积分, `int` 函数的具体用法如下所示:

- `int(s)`: 没有指定积分变量和积分阶数,系统按 `findsym` 函数指示的默认变量对被积函数或符号表达式 s 求不定积分。
- `int(s,v)`: 以 v 为自变量,计算被积函数或符号表达式 s 的不定积分。
- `int(s,v,a,b)`: 计算表达式 s 的定积分。该函数求在 $[a,b]$ 区间上的定积分, a 和 b 分别是积分的下限和上限。 a 和 b 可以是两个具体的数,也可以是一个符号表达式或是无穷(`inf`),

当 a 、 b 中有一个或两个是 inf 时，函数返回一个广义积分；当 a 、 b 中有一个符号表达式时，函数返回一个符号函数。系统按 `findsym` 函数指示的默认变量来确定表达式的变量，当表达式 s 是符号矩阵时，则对矩阵的各个元素分别进行积分。

- `int(s,v,a,b)`: 符号表达式采用符号标量 v 作为标量，求 v 从 a 变到 b 时，符号表达式 s 的定积分值， a 和 b 的规定同上。

【例 3-79】计算 $f = \sin x \cos^2(x) + 2$ 函数关于 x 的不定积分。

在命令窗口中输入如下语句：

```
f=sym('sin(x)*cos(x)^2+2');
int(f)
```

命令窗口中的输出结果如下所示：

```
ans =
2*x - cos(3*x)/12 - cos(x)/4
```

【例 3-80】计算 $f = e^{ax+b} + 1$ 函数关于 x 的不定积分。

在命令窗口中输入如下语句：

```
f=sym('exp(a*x+b)+1');
int(f)
```

命令窗口中的输出结果如下所示：

```
ans =
x + exp(b + a*x)/a
```

【例 3-81】计算 $f = ax^2 + \sin(a) + ab + e^{a+b}$ 函数关于 a 的不定积分。

在命令窗口中输入如下语句：

```
syms a x b;
f=a*x^2+sin(a)+a*b+exp(a+b);
int(f,a)
```

命令窗口中的输出结果如下所示：

```
ans =
exp(a + b) - cos(a) + a^2*(x^2/2 + b/2)
```

【例 3-82】计算二重不定积分 $f = \iint y e^{-xy} dx dy$ 的值。

在命令窗口中输入如下语句：

```
syms x y;
f=int(int('y*exp(-2*x*y)','x'),'y')
```

命令窗口中的输出结果如下所示：

```
f =
1/(4*x*exp(2*x*y))
```

【例 3-83】计算 $f = \int_0^{\frac{\pi}{4}} \sin^6\left(\frac{x}{2}\right) dx$ 的定积分。

在命令窗口中输入如下语句：

```
syms x
f=sin(x/2)^6;
```

```
int(f,0,pi/4)
```

命令窗口中的输出结果如下所示：

```
ans =
(5*pi)/64 - (23*2^(1/2))/96 + 3/32
```

【例 3-84】计算 $f = \int_0^{\pi/4} \sin(x) \cos(y) dy$ 的定积分。

在命令窗口中输入如下语句：

```
syms x y
f=sin(x)*cos(y);
int(f,y,0,pi/4)
```

命令窗口中的输出结果如下所示：

```
ans =
(2^(1/2)*sin(x))/2
```

【例 3-85】计算广义积分 $f_1 = \int_{-\infty}^{+\infty} \frac{dx}{x^2 + 1}$ 、 $f_2 = \int_1^{+\infty} \frac{dx}{x^4}$ 的值。

在命令窗口中输入如下语句：

```
syms x
f1=1/(x^2+1);
f2=1/(x^4);
int(f1,-inf,inf)
int(f2,1,inf)
```

命令窗口中的输出结果如下所示：

```
ans =
pi

ans =
1/3
```

4. 级数求和的计算

收敛的幂级数，不论是常数项级数还是函数项级数，都有求和的问题。MATLAB 提供了 `symsum` 函数用于计算符号表达式的和，它的具体用法如下所示。

- `r = symsum(s)`：自变量是由 `findsym` 函数所确定的符号变量，默认自变量为 `k`，计算表达式 `s` 从 0 到 `k-1` 的和。
- `r = symsum(s,v)`：计算表达式 `s` 从 0 到 `v-1` 的和。
- `r = symsum(s,a,b)`：计算表达式 `s` 默认变量从 `a` 到 `b` 的和。
- `r = symsum(s,v,a,b)`：计算表达式 `s` 变量 `v` 从 `a` 到 `b` 的和。

【例 3-86】使用 `symsum` 函数计算符号表达式的和。

在命令窗口中输入如下语句：

```
syms x k
symsum(x^3+6*x^2)      % 计算符号表达式的和
symsum(1/x^k,k,0,inf)  % 计算无穷级数的和
```

```
symsum(1/x^k,k,1,inf) %计算无穷级数的和
```

命令窗口中的输出结果如下所示：

```
ans =
1/4*x^4+3/2*x^3-11/4*x^2+x

ans =
x/(x-1)

ans =
1/(x-1)
```

5. Taylor 级数的计算

给定函数 $f(x)$ ，是否能找到这样一个幂级数，它在某区间内收敛，且其和正好是给定函数 $f(x)$ 。若能找到这样的幂级数，则说函数 $f(x)$ 在该区间内能展开成幂级数。

若函数 $f(x)$ 在点 x_0 的某一临域内具有从 1 阶直到 $n+1$ 阶的导数，则在该临域内，函数 $f(x)$ 在点 $x = x_0$ 时，项数趋向无穷的幂级数如下：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \cdots$$

这个幂级数叫做函数 $f(x)$ 的 Taylor 级数（展开）。

MATLAB 提供了 `taylor` 函数用来计算符号表达式的泰勒级数展开式，它的具体用法如下所示。

- `r = taylor(f)`: f 是符号表达式，自变量是由 `findsym` 函数所确定的符号变量，该函数将返回 f 在变量等于 0 处做 5 阶泰勒展开时的展开式。
- `r = taylor(f,n,v)`: 符号表达式 f 以符号标量 v 作为自变量，返回 f 的 $n-1$ 阶麦克劳林级数（即在 $v = 0$ 处做泰勒展开）展开式。
- `r = taylor(f,n,v,a)`: 返回符号表达式 f 在 $v = a$ 处做 $n-1$ 阶泰勒展开的展开式。

【例 3-87】使用 `taylor` 函数计算符号表达式 $f(x) = e^x$ 的泰勒级数展开式。

在命令窗口中输入如下语句：

```
syms x
f=exp(x);
r=taylor(f) %返回 5 阶泰勒展开式
```

命令窗口中的输出结果如下所示：

```
r =
x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
```

【例 3-88】使用 `taylor` 函数计算符号表达式 $f(x) = \ln(x+1)$ 的泰勒级数展开式。

在命令窗口中输入如下语句：

```
syms x
f=log(x+1);
r=taylor(f,6,x)%以符号变量 x 为自变量，返回 5 阶泰勒展开式
```

命令窗口中的输出结果如下所示：

```
r =
x^5/5 - x^4/4 + x^3/3 - x^2/2 + x
```

【例 3-89】使用 `taylor` 函数将 $f(x) = \sin(x)$ 展开成 $\left(x - \frac{\pi}{4}\right)$ 的幂级数。

在命令窗口中输入如下语句：

```
syms x
f=sin(x);
r=taylor(f,pi/4,4)
```

命令窗口中的输出结果如下所示：

```
r =
(2^(1/2)*(pi/4 - x)^3)/12 - (2^(1/2)*(pi/4 - x)^2)/4 + 2^(1/2)/2 - (2^(1/2)*(pi/4 - x))/2
```

3.5 符号表达式积分变换

积分变换是工程设计和计算常用的工具，本节将介绍傅里叶（Fourier）变换、拉普拉斯（Laplace）变换和 Z 变换。

3.5.1 Fourier 变换及其反变换

1. Fourier 变换

函数 $f(x)$ 在区间 $(-\infty, +\infty)$ 上是连续的，并且积分 $\int_{-\infty}^{+\infty} f(x)e^{-j\omega x} dx$ 收敛于 $F(\omega)$ ，称 $F(\omega)$ 是 $f(x)$ 的 Fourier 变换式，时域中的 $f(x)$ 与它在频域中的 Fourier 变换 $F(\omega)$ 之间存在如下关系：

$$F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-j\omega x} dx$$

进行 Fourier 变换有两种方式：一种是直接调用 `fourier` 函数，另一种是根据定义，利用积分函数 `int` 实现，直接使用 `fourier` 函数实现变换比较简单。下面只介绍 `fourier` 函数的使用方法，它的具体用法如下所示。

- `fourier(f)`：默认函数 f 的自变量是 x ，对默认变量计算 Fourier 变换式，并且默认输出结果 F 是变量 ω 的函数，记为 $F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-j\omega x} dx$ 。
- `fourier(f,v)`：在默认函数 f 的自变量是 x 的情况下，指定参数变为 v ，对函数 f 进行 Fourier 变换，记为 $F(v) = \int_{-\infty}^{+\infty} f(x)e^{-jvx} dx$ 。
- `fourier(f,u,v)`：指定函数 f 的自变量是 u 、指定参数是 v ，求函数 f 的 Fourier 变换，记为 $F(v) = \int_{-\infty}^{+\infty} f(u)e^{-jvu} du$ 。

【例 3-90】分别用默认参数 `fourier(f)` 函数和指定参数 `fourier(f,v)` 函数计算 $f(x) = e^{-\frac{x}{2}} \cos(3x)$ 的 Fourier 变换。

在命令窗口中输入如下语句：

```
syms x u v
f=exp(-x/2)*cos(3*x)
```

```
fourier(f)
fourier(f,v)
```

命令窗口中的输出结果如下所示：

```
f =
cos(3*x)/exp(x^2)

ans =
transform::fourier(cos(x)^3/exp(x^2), x, -w) - 3*transform::fourier((cos(x)*sin(x)^2)/exp(x^2), x, -w)

ans =
transform::fourier(cos(x)^3/exp(x^2), x, -v) - 3*transform::fourier((cos(x)*sin(x)^2)/exp(x^2), x, -v)
```

【例 3-91】使用 `fourier(f,u,v)` 函数在指定自变量和变换参数的情况下计算 $f(x)$ 的 Fourier 变换。在命令窗口中输入如下语句：

```
syms t u v
f=exp(-(t+u)^2/3)
fourier(f,t,v)
```

命令窗口中的输出结果如下所示：

```
f =
1/exp((t + u)^2/3)

ans =
(3^(1/2)*pi^(1/2))/exp((3*(- v + (2*u*i)/3)^2/4 + u^2/3)
```

2. Fourier 反变换

时域中的 $f(x)$ 和频域中的 $F(\omega)$ 的 Fourier 反变换存在如下关系：

$$f(x) = \frac{1}{\pi} \int_{-\infty}^{\infty} F(\omega) e^{-j\omega x} d\omega$$

进行 Fourier 反变换有两种方式：一种是直接调用 `ifourier` 函数；另一种是根据定义，利用积分函数 `int` 实现，直接使用函数实现变换比较简单。下面只介绍 `ifourier` 函数的使用方法，它的具体用法如下所示。

- `ifourier(F)`：在系统默认自变量和变换参数的情况下，计算函数的 Fourier 反变换，记为

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{-j\omega x} d\omega。$$

- `ifourier(F,v)`：在系统默认自变量，并指定变换参数是 v 的情况下，计算函数的 Fourier 反变换，记为 $f(v) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{-j\omega v} d\omega。$

- `ifourier(F,w,v)`：在系统的自变量为 w ，并指定变换参数是 v 的情况下，计算函数的 Fourier 反变换，记为 $f(v) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(w) e^{-j\omega v} dw。$

【例 3-92】使用函数 `ifourier(F)` 在系统默认自变量和变换参数的情况下，计算函数的 Fourier 反变换。

在命令窗口中输入如下语句：

```
syms x w u v
F=exp(-x^2)*cos(3*x)
ifourier(F)
```

命令窗口中的输出结果如下所示:

```
F =
cos(3*x)/exp(x^2)

ans =
-(3*transform::fourier((cos(x)*sin(x)^2)/exp(x^2), x, t) - transform::fourier(exp(-x^2)*cos(x)^3, x, t))/(2*pi)
```

【例 3-93】使用函数 `ifourier(F,v)` 在系统默认自变量, 并指定变换参数是 v 的情况下计算函数的 Fourier 反变换。

在命令窗口中输入如下语句:

```
syms v w
F=exp(-(v+w)^2/5)
ifourier(F,v)
```

命令窗口中的输出结果如下所示:

```
F =
1/exp((v + w)^2/5)

ans =
(5^(1/2)*(1/exp(v^2*(5/4 - i))))/(2*pi^(1/2))
```

【例 3-94】使用函数 `ifourier(F,w,v)` 在系统的自变量为 w , 并指定变换参数是 v 的情况下计算函数的 Fourier 反变换。

在命令窗口中输入如下语句:

```
syms w v a
F=cos(w-v)*exp(-a*w^2/3);
ifourier(F,w,v)
```

命令窗口中的输出结果如下所示:

```
ans =
(cos(-v)*transform::fourier(cos(w)/exp((a*w^2)/3), w, v) + sin(-v)*transform::fourier(sin(w)/exp((a*w^2)/3), w, v))/(2*pi)
```

3.5.2 Laplace 变换及其反变换

1. Laplace 变换

如果函数 $f(t)$ 在区间 $[0, +\infty)$ 上有定义, 并且积分 $\int_0^{+\infty} f(t)e^{-st} dt$ 在 s 的某一区域内收敛, 则由这个积分确定函数 $F(s)$, 即 $F(s) = \int_0^{+\infty} f(t)e^{-st} dt$ 。此式称为函数 $f(t)$ 的 Laplace 变换式, 记为 $L[f(t)] = F(s)$ 。

与 Fourier 变换相似, 进行 Laplace 变换也有两种方式: 一种是直接调用 `laplace` 函数; 另

一种是根据上面的定义，利用积分函数 `int` 实现。直接使用 `laplace` 函数实现变换比较简单，下面只介绍 `laplace` 函数的使用方法，它的具体用法如下所示。

- `laplace(F)`: 在默认自变量（自变量默认为 x ）和参变量（参变量默认为 s ）的情况下，计算符号函数的 Laplace 变换，记为 $L(s) = \int_0^{+\infty} F(x)e^{-sx} dx$ 。
- `laplace(F,z)`: 在默认自变量（自变量默认为 x ），并指定参变量为 z 的情况下，计算符号函数的 Laplace 变换，记为 $L(z) = \int_0^{+\infty} F(x)e^{-zx} dx$ 。
- `laplace(F,w,z)`: 在指定自变量为 w ，并指定参变量为 z 的情况下，计算符号函数的 Laplace 变换，记为 $L(z) = \int_0^{+\infty} F(w)e^{-zw} dw$ 。

【例 3-95】使用函数 `laplace(F)` 在默认自变量和参变量的情况下，计算符号函数的 Laplace 变换。

在命令窗口中输入如下语句：

```
syms x
F=exp(x-1)*x^2+2*x+3;
L=laplace(F)
```

命令窗口中的输出结果如下所示：

```
L =
3/s + 2/s^2 + 2/(exp(1)*(s - 1)^3)
```

【例 3-96】使用函数 `laplace(F,w,z)` 在指定自变量为 w ，并指定参变量为 z 的情况下，计算符号函数的 Laplace 变换。

在命令窗口中输入如下语句：

```
syms w z
F=5*x^3*tan(w+2)+log(w+1);
L=laplace(F,w,z)
```

命令窗口中的输出结果如下所示：

```
L =
5*laplace(tan(w + 2), w, z)*x^3 + laplace(log(w + 1), w, z)
```

【例 3-97】使用函数 `laplace(F,z)` 在默认自变量，并指定参变量为 z 的情况下，计算符号函数的 Laplace 变换。

在命令窗口中输入如下语句：

```
syms w z
F=cos(x^2)+sin(x);
L=laplace(F,z)
```

命令窗口中的输出结果如下所示：

```
L =
1/(z^2 + 1) + laplace(cos(x^2), x, z)
```

2. Laplace 反变换

Laplace 反变换定义为： $F(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} L(s)e^{st} dt$ ，其中 c 为使函数 $L(s)$ 的所有奇点位于

直线 $s = c$ 左边的实数, Laplace 反变换记为 $F(t) = L^{-1}[L(s)]$ 。

进行 Laplace 反变换同样也有两种方式: 一种是直接调用指令 `ilaplace` 函数; 另一种是根据上面的定义, 利用积分函数 `int` 实现。直接使用 `ilaplace` 函数实现变换比较简单, 下面只介绍 `ilaplace` 函数的使用方法, 它的具体用法如下所示。

- `ilaplace(L)`: 在默认自变量 (自变量默认为 s) 和参变量 (参变量默认为 t) 的情况下, 计算函数 $L(s)$ 的 Laplace 反变换, 记为 $L^{-1}[L(s)] = F(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} L(s)e^{st} ds$, 其中 c 为使函数 $L(s)$ 的所有奇点位于直线 $s = c$ 左边的实数。
- `ilaplace(L,v)`: 在默认自变量 (自变量默认为 s) 并指定参变量 v 的情况下, 计算函数 $L(s)$ 的 Laplace 反变换, 记为 $L^{-1}[L(s)] = F(v) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} L(s)e^{sv} ds$, 其中 c 为使函数 $L(s)$ 的所有奇点位于直线 $s = c$ 左边的实数。
- `ilaplace(L,v,x)`: 在指定自变量为 x , 并指定参变量为 v 的情况下, 计算函数 $L(s)$ 的 Laplace 反变换, 记为 $L^{-1}[L(x)] = F(v) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} L(x)e^{sv} dx$, 其中 c 为使函数 $L(s)$ 的所有奇点位于直线 $s=c$ 左边的实数。

【例 3-98】使用函数 `ilaplace(L)` 在默认自变量和参变量的情况下, 计算函数 $L(s)$ 的 Laplace 反变换。

在命令窗口中输入如下语句:

```
syms s
L=1/(s-1);
F=ilaplace(L)
```

命令窗口中的输出结果如下所示:

```
F =
exp(t)
```

【例 3-99】使用函数 `ilaplace(L)` 在默认自变量和参变量的情况下, 计算函数 $L(s)$ 的 Laplace 反变换。

在命令窗口中输入如下语句:

```
syms s
L=1/(s^3+1);
F=ilaplace(L)
```

命令窗口中的输出结果如下所示:

```
F =
1/(3*exp(t)) - (exp(t/2)*(cos((3^(1/2)*t)/2) - 3^(1/2)*sin((3^(1/2)*t)/2))/3
```

【例 3-100】使用函数 `ilaplace(L,v)` 在默认自变量并指定参变量为 v 的情况下, 计算函数 $L(s)$ 的 Laplace 反变换。

在命令窗口中输入如下语句:

```
syms v s
L=s^(-sym(5/2));
ilaplace(L,v)
```


在命令窗口中的输出结果如下所示：

```
ans =
4/3*v^(3/2)/pi^(1/2)
```

【例 3-101】使用函数 `ilaplace(L,v,x)` 在指定自变量为 x ，并指定参变量为 v 的情况下，计算函数 $L(s)$ 的 Laplace 反变换，在命令窗口中输入如下语句：

```
syms v x w
L=v/(v^2+w^2+1);
ilaplace(L,v,x)
```

命令窗口中的输出结果如下所示：

```
ans =
cos(x*(w^2 + 1)^(1/2))
```

3.5.3 Z 变换及其反变换

1. Z 变换

Z 变换的算法是：当 $t < 0$ 时， $f^*(t) = f(t) = 0$ ；当 $t \geq 0$ 时， $f^*(t) = \sum_{k=0}^{+\infty} f(kT)\delta(t-kT)$ 。

对该式进行 Laplace 变换，得到 $F^*(s) = \sum_{k=0}^{+\infty} f(kT)e^{-skT}$ ，此时令 $z=e^{sT}$ ，于是函数变为

$F^*(z) = \sum_{k=0}^{+\infty} f(kT)z^{-k}$ ，在函数 $F^*(z)$ 收敛的情况下，称 $F^*(z)$ 是 $f^*(t)$ 的 Z 变换，Z 变换记

为 $F(z) = \sum_{n=0}^{\infty} f(n)z^{-n}$ 。

MATLAB 提供了 `ztrans` 函数实现 Z 变换，它的具体用法如下所示。

- `ztrans(f)`：在默认自变量（默认自变量为 n ）和参变量（参变量默认为 z ）的情况下，计算符号函数的 Z 变换，记为 $F(z) = \sum_{n=0}^{+\infty} f(n)z^{-n}$ 。
- `ztrans(f,v)`：在默认自变量（默认自变量为 n ）并指定参变量为 v 的情况下，计算符号函数的 Z 变换，记为 $F(v) = \sum_{n=0}^{+\infty} f(n)v^{-n}$ 。
- `atrans(f,k,v)`：在指定自变量为 k ，并指定参变量为 v 的情况下，计算符号函数的 Z 变换，记为 $F(v) = \sum_{k=0}^{+\infty} f(k)v^{-k}$ 。

【例 3-102】使用函数 `ztrans(f)` 在默认自变量和参变量的情况下，对函数进行 Z 变换。

在命令窗口中输入如下语句：

```
syms n
f=3^n+1;
ztrans(f)
```

命令窗口中的输出结果如下所示：

```
ans =
```

$$z/(z-1) + z/(z-3)$$

【例 3-103】使用函数 `ztrans(f,v)` 在默认自变量并指定参变量为 v 的情况下，对函数进行 Z 换。

在命令窗口中输入如下语句：

```
syms n k
f=cos(2*k*n);
ztrans(f,v)
```

命令窗口中的输出结果如下所示：

```
ans =
(v*(v - cos(2*k)))/(v^2 - 2*cos(2*k)*v + 1)
```

【例 3-104】使用函数 `ztrans(f,k,v)` 对指定自变量为 k ，并指定参变量为 v 的函数进行 Z 变换。

在命令窗口中输入如下语句：

```
syms k m v
f=sin(2*k*m);
ztrans(f,k,v)
```

输出结果如下：

```
ans =
(v*sin(2*m))/(v^2 - 2*cos(2*m)*v + 1)
```

2. Z 反变换

Z 反变换记为： $f(n)=\mathcal{Z}^{-1}(F(z))$ ，MATLAB 提供了 `iztrans` 函数实现 Z 反变换，其具体用法如下：

- `iztrans(F)`：在默认自变量（默认自变量为 n ）和参变量（参变量默认为 z ）的情况下，对函数进行 Z 反变换，记为 $f(n) = \frac{1}{2\pi i} \int_{|z|=R} F(z) z^{n-1} dz, n=1,2,3,\dots$
- `iztrans(F,v)`：在默认自变量（默认自变量为 n ）并指定参变量为 v 的情况下，对函数进行 Z 反变换，记为 $f(n) = \frac{1}{2\pi i} \int_{|v|=R} F(v) v^{n-1} dv, v=1,2,3,\dots$
- `iztrans(F,w,v)`：在指定自变量为 w ，并指定参变量为 v 的情况下，对函数进行 Z 反变换，记为 $f(w) = \frac{1}{2\pi i} \int_{|v|=R} F(v) v^{w-1} dv, v=1,2,3,\dots$

【例 3-105】使用函数 `iztrans(F)` 在默认自变量和参变量的情况下，对函数进行 Z 反变换。在命令窗口中输入如下语句：

```
syms z
f=z/((z-5)*(z-1));
iztrans(f)
```

命令窗口中的输出结果如下：

```
ans =
5^n/4 - 1/4
```

【例 3-106】使用函数 `iztrans(F)` 在默认自变量和参变量的情况下，对函数进行 Z 反变换。在命令窗口中输入如下语句：

```
syms x z
f=exp(a*x/z);
iztrans(f)
```

命令窗口中的输出结果如下所示：

```
ans =
(a*x)^n/factorial(n)
```

【例 3-107】使用函数 `iztrans(F,v)` 在默认自变量并指定参变量为 v 的情况下，对函数进行 Z 变换。

在命令窗口中输入如下语句：

```
syms x z
f=exp(a*x/z);
iztrans(f,v)
```

命令窗口中的输出结果如下所示：

```
ans =
(a*x)^v/factorial(v)
```

【例 3-108】使用函数 `iztrans(F,v)` 在默认自变量并指定参变量为 v 的情况下，对函数进行 Z 反变换。

在命令窗口中输入如下语句：

```
syms z
f=z/((z-5)*(z-1));
iztrans(f,v)
```

命令窗口中的输出结果如下所示：

```
ans =
5^v/4 - 1/4
```

【例 3-109】使用函数 `iztrans(F,w,v)` 在指定自变量为 w ，并指定参变量为 v 的情况下，对函数进行 Z 反变换。

在命令窗口中输入如下语句：

```
syms w v
f=w*(w+2)/(w^2+4*w+3);
iztrans(f,w,v)
pretty(ans)
```

命令窗口中的输出结果如下所示：

```
ans =
(-1)^v/2 + (-3)^v/2
```

$$\frac{(-1)^v}{2} + \frac{(-3)^v}{2}$$

3.6 符号函数的图形绘制

MATLAB 提供了数值运算和符号运算功能，还开发了强大的图形显示功能。本节只介绍符号函数的图形绘制，关于图形绘制的详细内容将在第4章详细讲解。

3.6.1 符号函数曲线的绘制

MATLAB 提供了 `ezplot` 函数和 `ezplot3` 函数用于绘制符号函数的二维曲线和三维曲线。

1. 二维曲线的绘制

`ezplot` 函数用于绘制符号函数的二维曲线，此函数可以绘制显函数图形、隐函数图形和参数方程的图形，具体用法如下所示。

- `ezplot(f)`: 绘制显函数 f 在区间 $[-2\pi, 2\pi]$ 的二维曲线；绘制隐函数 f 在区间 $-2\pi < x < 2\pi$ 和 $-2\pi < y < 2\pi$ 的曲线；绘制参数方程 $x = x(t)$ 、 $y = y(t)$ 在区间 $0 < t < 2\pi$ 的曲线。
- `ezplot(f,[min,max])`、`ezplot(f,[xmin,xmax, ymin,ymax,])` 和 `ezplot(x,y,[tmin,tmax])`: 第一种用法是绘制显函数 f 在指定区间 $[\min, \max]$ 的二维曲线；第二种用法是绘制隐函数 f 在指定区间 $x_{\min} < x < x_{\max}$ 和 $y_{\min} < y < y_{\max}$ 的曲线；第三种用法是绘制参数方程 $x = x(t)$ 、 $y = y(t)$ 在区间 $t_{\min} < t < t_{\max}$ 的曲线。

【例 3-110】使用函数 `ezplot(f)` 绘制显函数的二维曲线。

在命令窗口中输入如下语句：

```
syms x
f=cos(x);
ezplot(f)
grid
title('cos(x)')
```

图形窗口中的输出结果如图 3-1 所示。

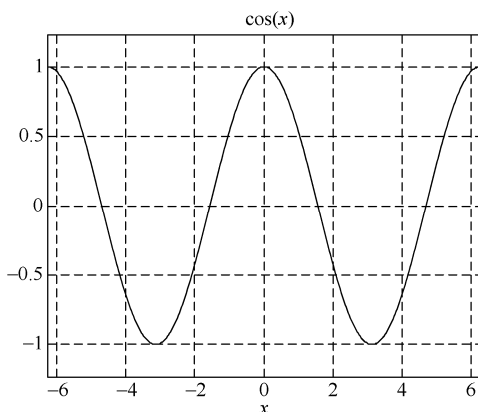


图 3-1 显函数的二维曲线

【例 3-111】使用函数 `ezplot(f)` 绘制隐函数 $2x^2 - 3y^3 + 1 = 0$ 的二维曲线。

在命令窗口中输入如下语句：

```
syms x y
```

```
f1=2*x^2-3*y^3+1;  
ezplot(f1)  
grid  
title('隐函数')
```

图形窗口中的输出结果如图 3-2 所示。

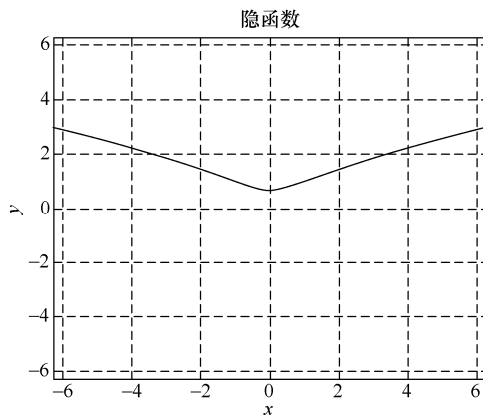


图 3-2 隐函数的二维曲线

【例 3-112】使用函数 `ezplot(f)` 绘制参数方程 $x^2 - y^3 = 0$ 的二维曲线。
在命令窗口中输入如下语句：

```
syms x y t  
x=2*sin(t);  
y=t*cos(t);  
ezplot(x,y)  
grid  
title('参数方程')
```

图形窗口中的输出结果如图 3-3 所示。

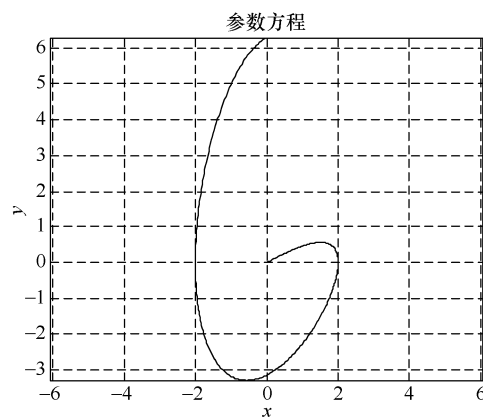


图 3-3 二维曲线

2. 三维曲线的绘制

`ezplot3` 函数用于绘制符号函数的三维曲线，具体用法如下所示。

- `ezplot3(x,y,z)`: 绘制参数方程 $x = x(t)$ 、 $y = y(t)$ 、 $z = z(t)$ 在默认区间 $0 < t < 2\pi$ 的三维曲线。
- `ezplot3(x,y,z[tmin,tmax])`: 绘制参数方程 $x = x(t)$ 、 $y = y(t)$ 、 $z = z(t)$ 在区间 $t_{\min} < t < t_{\max}$ 的三维曲线。
- `ezplot3(...,'animate')`: 生成空间曲线的动态轨迹。

【例 3-113】使用 `ezplot3` 函数绘制三维曲线。

在命令窗口中输入如下语句：

```
syms t
x=sin(t);
y=cos(t);
z=t;
ezplot3(x,y,z)
grid
title('三维曲线')
```

图形窗口中的输出结果如图 3-4 所示。

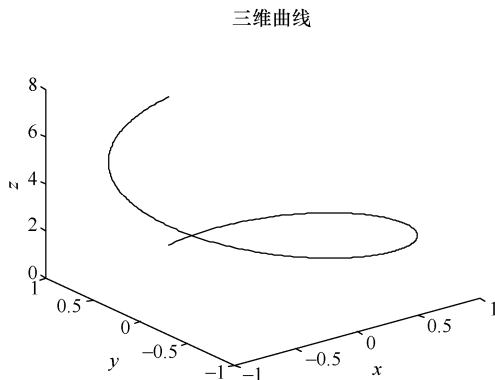


图 3-4 三维曲线

3.6.2 符号函数等值线的绘制

MATLAB 提供了 `ezcontour` 函数和 `ezcontourfd` 函数用于绘制符号函数的等值线，两个函数的使用方法类似，区别在于 `ezcontourfd` 函数绘制带有填充区域的等值线。`ezcontour` 函数的具体用法如下所示。

- `ezcontour(f)`: 绘制二元函数 $f(x,y)$ 在默认区域的等值线。
- `ezcontour(f,domain)`: 绘制二元函数 $f(x,y)$ 在指定区域的等值线。
- `ezcontour(...,n)`: 绘制等值线图，并指定等值线的条数。

【例 3-114】使用 `ezcontour` 函数绘制符号函数的等值线。

在命令窗口中输入如下语句：

```
syms x y
f = 3*(1-y)^2*exp(-(x^2)-(x+1)^2);
ezcontour(f)
```

图形窗口中的输出结果如图 3-5 所示。

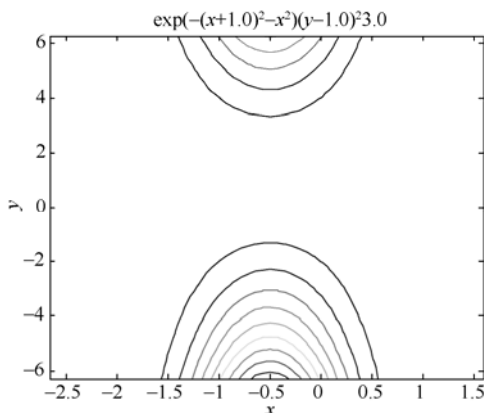


图 3-5 等值线图

【例 3-115】使用 ezcontourf 函数绘制符号函数的等值线。

在命令窗口中输入如下语句：

```
syms x y
f = 3*(1-y)^2*exp(-(x^2)-(x+1)^2);
ezcontourf(f)
```

图形窗口中的输出结果如图 3-6 所示。

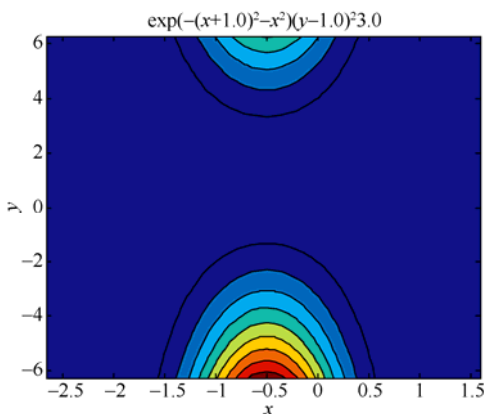


图 3-6 带填充区域的等值线图

3.6.3 符号函数曲面图及表面图的绘制

MATLAB 提供了 ezmesh 函数和 ezmeshc 函数用于绘制符号函数的三维曲面图,以及 ezsurf 函数和 ezsurf 函数用于绘制符号函数的三维表面图。

ezmesh 函数和 ezmeshc 函数的区别在于, ezmeshc 函数在绘制三维曲面图的同时绘制等值线; ezsurf 函数和 ezsurf 函数的区别在于, ezsurf 函数在绘制三维表面图的同时绘制等值线。

1. ezmesh 函数和 ezsurf 函数

MATLAB 提供了 ezmesh 函数用于绘制三维曲面图, ezsurf 函数绘制三维表面图, 两个函数使用的方法类似。ezmesh 函数的具体用法如下所示。

- ezmesh(f): 绘制 $f(x,y)$ 的图像。

- `ezmesh(f, domain)`: 在指定区域绘制 $f(x,y)$ 的图像。
- `ezmesh(x,y,z)`: 在默认区域绘制三维参数方程的图像。
- `ezmesh(x,y,z,[smin,smax,tmin,tmax])` 或 `ezmesh(x,y,z,[min,max])`: 在指定区域绘制三维参数方程的图像。

【例 3-116】利用 `ezmesh` 函数和 `ezsurf` 函数绘制三维曲面图和三维表面图。

在命令窗口中输入如下语句：

```
syms x y
subplot(2,1,1)
ezmesh(x*exp(-x^2+y^2),[-2.5,2.5])
title('三维曲面图')
subplot(2,1,2)
ezsurf(x*exp(-x^2+y^2),[-2.5,2.5])
title('三维表面图')
```

图形窗口中的输出结果如图 3-7 所示。

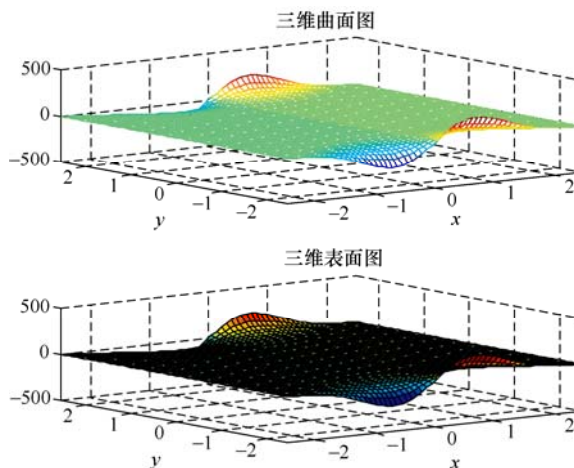


图 3-7 三维曲面图和三维表面

2. `ezmeshc` 函数和 `ezsurf` 函数

MATLAB 提供了 `ezmeshc` 函数用于绘制带等值线的三维曲面图，`ezsurf` 函数用于绘制带等值线的三维表面图，两个函数使用的方法类似。`ezmeshc` 函数的具体用法如下所示。

- `ezmeshc(f)`: 在默认区域 $-2\pi < x < 2\pi$ 和 $-2\pi < y < 2\pi$ 绘制二元函数 $f(x,y)$ 的图像。
- `ezmeshc(f, domain)`: 在指定区域绘制二元函数 $f(x,y)$ 的图像。
- `ezmeshc(x,y,z)`: 在默认区域 $-2\pi < s < 2\pi$ 和 $-2\pi < t < 2\pi$ 绘制三维参数方程 $x = x(s,t)$ ， $y = y(s,t)$ ， $z = z(s,t)$ 的图像。
- `ezmeshc(x,y,z,[smin,smax,tmin,tmax])` 或 `ezmeshc(x,y,z,[min,max])`: 在指定区域绘制三维参数方程的图像。

【例 3-117】使用 `ezmeshc` 函数和 `ezsurf` 函数绘制带等值线的三维曲面图和三维表面图。

在命令窗口中输入如下语句：


```
syms x y
subplot(2,1,1)
ezmeshc(x*exp(-x^2+y^2),[-2.5,2.5])
title('带等值线的三维曲面图')
subplot(2,1,2)
ezsurf(x*exp(-x^2+y^2),[-2.5,2.5])
title('带等值线的三维表面图')
```

图形窗口中的输出结果如图 3-8 所示。

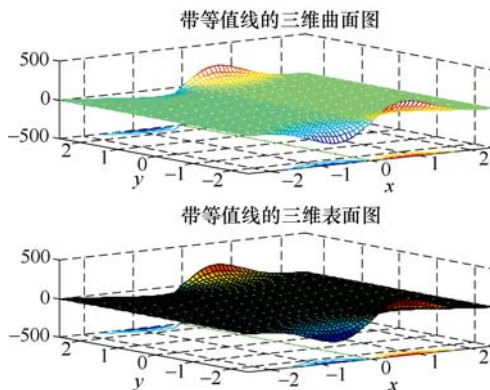


图 3-8 带等值线的三维曲面图和三维表面图

3.7 符号方程的求解

方程的求解是数学中的重要问题，MATLAB 为符号方程的求解提供了强有力的支持。符号方程求解可分为代数方程求解、微分方程求解、复合方程求解和反方程求解。

3.7.1 代数方程的求解

代数方程包括线性方程、非线性方程和超越方程。

1. 求解代数方程

MATLAB 提供了 solve 函数用于求解代数方程，它的具体用法如下所示。

- $g = \text{solve}(\text{eq})$: 其中 eq 可以是符号表达式或字符串，该函数用于求解方程 $\text{eq} = 0$ ，其自变量采用默认变量，自变量可以由 findsym 函数确定。
- $g = \text{solve}(\text{eq}, \text{var})$: 求解方程 $\text{eq} = 0$ ，其自变量由参数 var 指定。其中 eq 和上一种调用方式相同，返回值 g 是由方程的所有解构成的列向量。

【例 3-118】 使用 solve 函数求解代数方程 $4ax^2 + bx - c = 0$ 。

在命令窗口中输入如下语句：

```
syms x a b c
x=solve('4*a*x^2 + b*x - c') %默认变量
```

命令窗口中的输出结果如下所示：

```
x =
-(b + (b^2 + 16*a*c)^(1/2))/(8*a)
```

```
-(b - (b^2 + 16*a*c)^(1/2))/(8*a)
```

【例 3-119】 使用 solve 函数求解代数方程 $ax^2 + bx + 4c = 0$ 。

在命令窗口中输入如下语句：

```
syms a b c x
b=solve('a*x^2 + b*x +4*c','b') %以 b 为自变量
```

命令窗口中的输出结果如下所示：

```
b =
-(a*x^2 + 4*c)/x
```

2. 求解代数方程组

MATLAB 提供了 solve 函数用于求解代数方程组，它的具体用法如下所示。

- $g = \text{solve}(\text{eq1}, \text{eq2}, \dots, \text{eqn})$ ：求解由符号表达式或字符串 eq1, eq2, ..., eqn 组成的方程组。其中的自变量为整个方程组的默认变量，即将函数 findsym 作用于整个方程组时返回的变量。
- $g = \text{solve}(\text{eq1}, \text{eq2}, \dots, \text{eqn}, \text{var1}, \text{var2}, \dots, \text{varn})$ ：求解由符号表达式或不带等号的字符串 eq1, eq2, ..., eqn 组成的方程组。其自变量由输入参数 var1, var2, ..., varn 指定。

【例 3-120】 使用 solve 函数求解代数方程组关于 x 和 y 的解。

在命令窗口中输入如下语句：

```
S = solve('2*x + 3*y = 1','5*x -4*y = 2')
disp('S.x'),disp(S.x),disp('S.y'),disp(S.y)
```

命令窗口中的输出结果如下所示：

```
S =
    x: [1x1 sym]
    y: [1x1 sym]
```

```
S.x
10/23
```

```
S.y
1/23
```

【例 3-121】 使用 solve 函数求解代数方程组。

在命令窗口中输入如下语句：

```
A = solve('a*u^2 + v^2', 'u+v = 1', 'a^2 - 5*a + 6','u','v','a')
disp('A.a'),disp(A.a),disp('A.u'),disp(A.u),disp('A.v'),disp(A.v)
```

命令窗口中的输出结果如下所示：

```
A =
    a: [4x1 sym]
    u: [4x1 sym]
    v: [4x1 sym]
```

```
A.a
```

```
2
2
3
3
```

A.u

```
1/3-1/3*i*2^(1/2)
1/3+1/3*i*2^(1/2)
1/4-1/4*i*3^(1/2)
1/4+1/4*i*3^(1/2)
```

A.v

```
2/3+1/3*i*2^(1/2)
2/3-1/3*i*2^(1/2)
3/4+1/4*i*3^(1/2)
3/4-1/4*i*3^(1/2)
```

3.7.2 微分方程的求解

从数值计算角度看,与初值问题求解相比,微分方程边值问题的求解显得复杂和困难。此时,不妨通过符号计算指令进行求解尝试。对于符号计算来说,不论是初值问题,还是边值问题,其求解微分方程的指令形式都相同,且比较简单。但是,符号计算可能会花费较多的计算机资源,可能得不到简单的解析解或封闭形式的解,甚至无法求解。所以没有万能的求解微分方程的一般解法,可以认为微分方程的符号法和数值法的作用是互补的。

1. 求解微分方程

MATLAB 提供了 `dsolve` 函数用来求常微分方程的符号解,它的具体用法如下所示。

- `r = dsolve('eq1,eq2,...','cond1,cond2,...','v')`: 计算由 `eq1, eq2, ...` 指定的常微分方程的符号解。常微分方程以变量 `v` 作为自变量,参数 `cond1, cond2, ...` 用于指定方程的边界条件或者初始条件,如果不指定 `v`,将默认 `t` 为自变量。
- `r = dsolve('eq1','eq2',...,'cond1','cond2',...,'v')`: 计算由 `eq1, eq2, ...` 指定的常微分方程的符号解。这些常微分方程都以 `v` 作为自变量,这些单独输入的方程的最大允许个数为 12,其他参数调用方式与同上。

对于以上具体的用法,需要说明的是:

(1) 在方程中,用大写字母 D 表示一次微分, $D2$ 、 $D3$ 分别表示二次、三次微分。以此类推,符号 $D2y$ 表示为 $\frac{d^2y}{dt^2}$ 。函数 `dsolve` 把 `d` 后面的字符当做因变量,并默认所有这些变量对 t 进行求导。

(2) 微分方程的初始条件或边界条件都以变量 `v` 作为自变量,其形式为 $y(a)=b$ 或 $Dy(a)=b$,其中 y 是微分方程的因变量, a 和 b 是常数。如果指定的初始条件和边界条件比方程中的因变量个数少,那么所得的解中将包含积分常数 $C1$ 、 $C2$ 等。

(3) 函数 `dsolve` 的输出结果同 `solve` 类似,既可以用和因变量个数相同的输出参数分别接

收每个变量的解，也可以把方程的解写入一个结构数组中。

【例 3-122】使用 `dsolve` 函数求解微分方程 $\frac{dx}{dt} = -ax, \frac{d^2x}{dt^2} = \sin(t) + \cos(t), \left(\frac{dy}{ds}\right)^2 + y^2 = 5$ 。

在命令窗口中输入如下语句：

```
dsolve('Dx = -a*x')
dsolve('D2x = sin(t)+cos(t)')
dsolve('(Dy)^2+y^2=5','s')
```

命令窗口中的输出结果如下所示：

```
ans =
C2/exp(a*t)

ans =
C5 - cos(t) - sin(t) + C4*t

ans =
      5^(1/2)
     -5^(1/2)
5^(1/2)*sin(C12 + s)
5^(1/2)*sin(C8 - s)
```

【例 3-123】使用 `dsolve` 函数在设定初值的情况下求解微分方程 $\frac{d^2y}{dt^2} = -a^2y$ 。

在命令窗口中输入如下语句：

```
dsolve('D2y=-a^2*y','y(0)=1','Dy(pi/2)=0') %限定初值
```

命令窗口中的输出结果如下所示：

```
ans =
sin(1/2*a*pi)/cos(1/2*a*pi)*sin(a*t)+cos(a*t)
```

2. 求解微分方程组

MATLAB 提供了 `dsolve` 函数用来求解微分方程组，它的具体用法如下所示。

- `r = dsolve('eq1,eq2,...','cond1,cond2,...','v')`: 计算由 `eq1, eq2, ...` 指定的常微分方程组的解。`v` 作为自变量，参数 `cond1, cond2, ...` 用于指定方程的边界条件或者初始条件，其他要求同上。

【例 3-124】使用 `dsolve` 函数求解微分方程组。

在命令窗口中输入如下语句：

```
syms x y
S = dsolve('Dx = -y+2', 'Dy = 2*x', 'x(0)=0', 'y(0)=1')
S.x    %查看 x 的值
S.y    %查看 y 的值
```

命令窗口中的输出结果如下所示：

```
S =
```

```

x: [1x1 sym]
y: [1x1 sym]

ans =
(2^(1/2)*sin(2^(1/2)*t))/2

ans =

2 - cos(2^(1/2)*t)

```

3.7.3 复合方程的求解

MATLAB 提供了 `compose` 函数用于求复合方程的解，它的具体用法如下所示。

- `compose(f,g)`: 返回函数 $f(g(y))$ ，其中 x 是 f 的默认变量，即 $f=f(x)$ ， y 是 g 的默认变量，即 $g=g(y)$ 。
- `compose(f,g,z)`: 返回函数 $f(g(z))$ ，自变量指定为 z 。
- `compose(f,g,x,z)`: 返回函数 $f(g(z))$ ，指定 f 的自变量是 x 。
- `compose(f,gx,y,z)`: 返回函数 $f(g(z))$ ，指定 f 的自变量是 x ，指定 g 的自变量是 y 。

【例 3-125】使用 `compose` 函数求复合方程的解。

在命令窗口中输入如下语句：

```

syms x y z u v
f=cos(1 + x^2);
g = tan(2*y);
m=sin(x^u);
n=exp(-y/5*v);
compose(f,g)
compose(f,g,u)
compose(m,n,x,z)
compose(m,n,u,z)
compose(m,n,x,y,z)
compose(m,n,u,v,z)

```

命令窗口中的输出结果如下所示：

```

ans =
cos(tan(2*y)^2 + 1)

ans =
cos(tan(2*u)^2 + 1)

ans =
sin((1/exp((v*z)/5))^u)

```

```
ans =
sin(x^(1/exp((v*z)/5)))
```

```
ans =
sin((1/exp((v*z)/5))^u)
```

```
ans =
sin(x^(1/exp((y*z)/5)))
```

3.7.4 反方程的求解

MATLAB 提供了 `finverse` 函数用于计算反函数，它的具体用法如下所示。

- `g=finverse(f)`: 在 `f` 函数的反函数存在的情况下，返回 `f` 函数的反函数，自变量为默认变量。
- `g=finverse(f,v)`: 在 `f` 函数的反函数存在的情况下，返回 `f` 函数的反函数，自变量设置为 `v`。

【例 3-126】计算由反函数构成方程的解。

在命令窗口中输入如下语句：

```
syms x y u v X Y U V
f1=x^3-2*y;
f2=exp(2*u+v);
g1=finverse(f1,y)
g2=finverse(f2,u)
g3=finverse(f2,v)
X= solve(g1,x)
Y= solve(g1,y)
U= solve(g2,u)
V= solve(g3,v)
```

命令窗口中的输出结果如下所示：

```
g1 =
x^3/2 - y/2

g2 =
log(u)/2 - v/2

g3 =
log(v) - 2*u

X =
y^(1/3)

y^(1/3)*(- 1/2 + (3^(1/2)*i)/2)
-y^(1/3)*(1/2 + (3^(1/2)*i)/2)
```

Y =

x^3

U =

exp(v)

V =

exp(2*u)

第 4 章 MATLAB 图形功能

MATLAB 具有很强的图形功能，可以方便地实现数据的视觉化，数据图形能使用户对数据有直观的感觉，较方便地发现其中的规律。强大的计算功能与图形功能相结合为 MATLAB 在科学技术和教学方面的应用提供了更加广阔的天地。

本章主要介绍二维图形和三维图形的绘制、图形处理的基本技术、图形窗口的操作。

4.1 二维基本图形

二维图形的绘制是 MATLAB 图形处理的基础。MATLAB 提供了丰富的绘图函数，既可以绘制基本的二维图形，又可以绘制特殊的二维图形，如柱状图、饼图和直方图等。

绘制二维图形的基本步骤如下：

（1）准备数据

首先获取自变量（向量），然后通过计算得到相应的函数值（向量）。

（2）设置当前绘图区

在指定的位置创建新的绘图窗口，系统自动以此窗口作为当前绘图区，当绘制子图时需要指定子图在整个绘图区中的位置。

（3）调用绘图指令

创建坐标轴进行图形绘制。

（4）设置坐标轴，标注图形

设置坐标轴的范围、刻度和坐标分格线等，并且对图形进行标注，如图名、坐标名、图例和文字说明等。

（5）设置图形的曲线和标记点的形式

通过 set 函数设置图形曲线的颜色、线宽、线型和标记点的颜色、大小、形状等。

（6）保存并导出图形

按照指定文件的属性、格式保存图形，配置打印机可以实现图形的导出。

以上需要说明的是，第（1）和（3）步是绘图的基本步骤，第（2）步在图形较多或绘制子图时使用，第（4）和（5）步用户可以根据自身的需要更改和调整，并且顺序不完全固定。

4.1.1 基本绘图函数

MATLAB 基本的绘图函数有 line 函数、plot 函数和 polar 函数，line 函数是直角坐标系中的简单绘图函数，plot 函数是直角坐标系中的常用绘图函数，polar 函数是极坐标系中的绘图函数。二维绘图函数中最核心且最基本的是 plot 函数。

1. line 函数

MATLAB 提供了 line 函数用于在直角坐标系下简单画线，它的具体用法如下所示：

- line(X,Y): X 和 Y 都是一维数组，line 函数将数组 (X(i),Y(i)) 中的各点用线段连接起来，

形成一条折线。

【例 4-1】使用 line 函数实现已知点连线功能。

在命令窗口中输入如下语句：

```
figure           %figure 函数用于创建图形窗口
X= 10:-1.03:1.35 %生成 X 的一维数组
Y=cos(X)         %生成 Y 的一维数组
line(X,Y)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-1 所示的结果。

```
X =
    10.0000    8.9700    7.9400    6.9100    5.8800    4.8500    3.8200    2.7900    1.7600
Y =
   -0.8391   -0.8984   -0.0859    0.8099    0.9198    0.1372   -0.7786   -0.9388   -0.1881
```

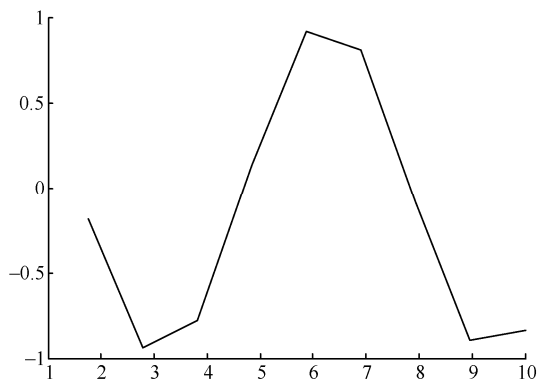


图 4-1 line 函数画线

2. plot 函数

plot 函数是直角坐标系中最基本的绘图函数。它有多种调用格式，不同的调用格式可以实现不同的功能。

(1) plot 函数最简单的形式是 plot(Y)，它的具体用法如下所示：

- plot(Y): 当 Y 为实向量时，plot(Y)是将各点(i,Y(i))依次连接起来，其中 i 从 1 到 length(Y)；当 Y 是实矩阵时，plot(Y)绘制矩阵 Y 的列元素值相对其下标的曲线，其中曲线数等于矩阵 Y 的列数；当 Y 是复数矩阵时，plot(Y)是绘制元素实部为横坐标，元素虚部为纵坐标的曲线。

【例 4-2】使用 plot 函数的最简单形式 plot(Y)绘制实向量的二维图形。

在命令窗口中输入如下语句：

```
figure
x=[-2.1  2.3  0  -2.5  2.9]
plot(x)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-2 所示的结果。

```
x =
   -2.1000    2.3000    0   -2.5000    2.9000
```

【例 4-3】使用 `plot` 函数的最简单形式 `plot(Y)` 绘制实矩阵的二维图形。

在命令窗口中输入如下语句：

```
figure
Y=[1 2 3;4 6 8;10 12 20]
plot(Y)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-3 所示的结果。

```
Y =
     1     2     3
     4     6     8
    10    12    20
```

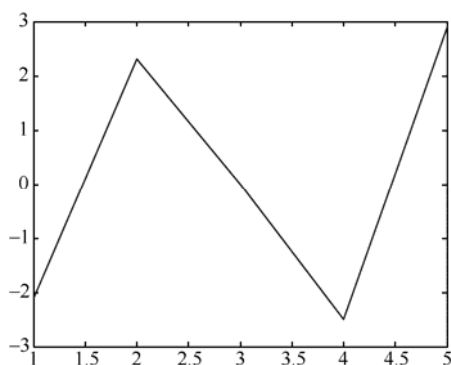


图 4-2 `plot` 函数绘制二维图形

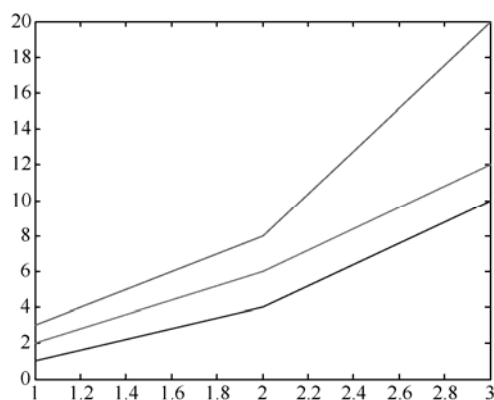


图 4-3 `plot` 函数绘制二维图形

【例 4-4】使用 `plot` 函数的最简单形式 `plot(Y)` 绘制复数矩阵的二维图形。

在命令窗口中输入如下语句：

```
figure
X=[1 2 3;4 6 8]
Y=[-1.2 0 2.1;3 5.2 8]
Z=X+Y*i;
plot(Z)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-4 所示的结果。

```
X =
     1     2     3
     4     6     8

Y =
    -1.2000     0     2.1000
     3.0000     5.2000     8.0000

Z =
    1.0000 - 1.2000i    2.0000    3.0000 + 2.1000i
    4.0000 + 3.0000i    6.0000 + 5.2000i    8.0000 + 8.0000i
```

(2) `plot` 函数常用的形式是 `plot(X,Y)`，它的具体用法如下所示：

- `plot(X,Y)`：当 `X` 和 `Y` 是同维向量时，`plot` 函数的功能和 `line` 函数相同；当 `X` 和 `Y` 是同

维矩阵时，`plot` 函数以矩阵 `X` 和 `Y` 的列元素作为横、纵坐标绘制曲线，其中曲线的条数等于矩阵的列数；当 `X` 是向量，`Y` 是有一维和 `X` 同维的矩阵时，`plot` 函数以 `X` 为曲线的横坐标，对 `X` 和 `Y` 的每一行或列画线，其中曲线的条数等于矩阵 `Y` 的另一维数；当 `X` 是矩阵，`Y` 是向量时情况同上；需要注意的是，`plot` 函数以 `Y` 为曲线的纵坐标。

【例 4-5】使用 `plot` 函数的常用形式 `plot(X,Y)` 绘制两同维向量的二维图形。

在命令窗口中输入如下语句：

```
figure
X=0:0.75:3.5
Y=3*X
plot(X,Y)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-5 所示的结果。

```
X =
    0    0.7500    1.5000    2.2500    3.0000
Y =
    0    2.2500    4.5000    6.7500    9.0000
```

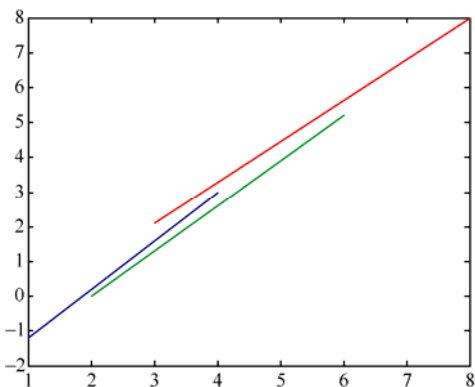


图 4-4 `plot` 函数绘制二维图形

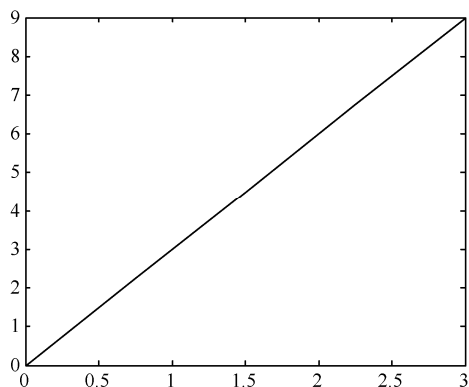


图 4-5 `plot` 函数绘制二维图形

【例 4-6】使用 `plot` 函数的常用形式 `plot(X,Y)` 绘制双矩阵的二维图形。

在命令窗口中输入如下语句：

```
figure
X=[-1.1 2 3 5.6;-2 1.3 5.7 8;-8 5 2.3 6]
Y=randn(3,4)
plot(X,Y)
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-6 所示的结果。

```
X =
   -1.1000    2.0000    3.0000    5.6000
   -2.0000    1.3000    5.7000    8.0000
   -8.0000    5.0000    2.3000    6.0000
Y =
   -0.4326    0.2877    1.1892    0.1746
```

-1.6656	-1.1465	-0.0376	-0.1867
0.1253	1.1909	0.3273	0.7258

【例 4-7】使用 plot 函数的常用形式 plot(X,Y) 绘制向量和矩阵的二维图形。

在命令窗口中输入如下语句：

```
figure
X=-1:0.1:5;
Y=[3*cos(X);cos(X);cos(X+0.1*pi)];
plot(X,Y)
```

图形窗口中的输出结果如图 4-7 所示。

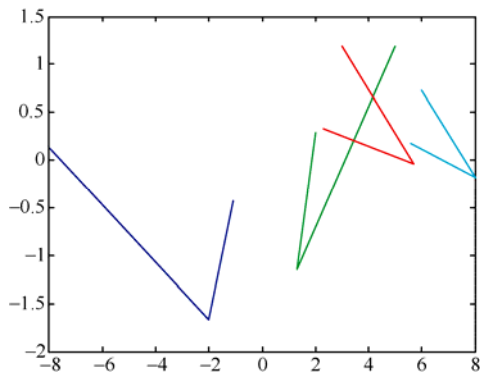


图 4-6 plot 函数绘制二维图形

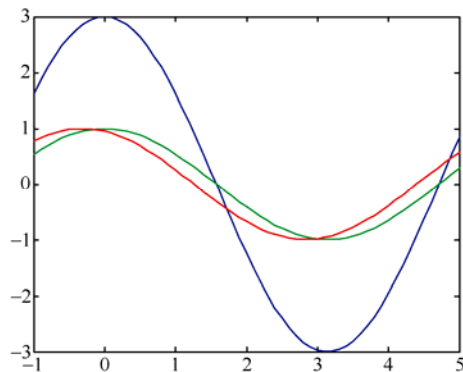


图 4-7 plot 函数绘制二维图形

(3) plot 函数可以同时多组变量进行绘图，它的具体用法如下所示：

- plot(X1,Y1,X2,Y2,...,Xn,Yn): 对多组变量同时进行绘图，不同组之间互不影响。

【例 4-8】使用 plot 函数对多组变量同时进行绘图。

在命令窗口中输入如下语句：

```
figure
X=0:0.5:10;
Y1=cos(X)
Y2=sin(X)
Y3=tan(X)
plot(X,Y1,X,Y2,X,Y3)
```

图形窗口中的输出结果如图 4-8 所示。

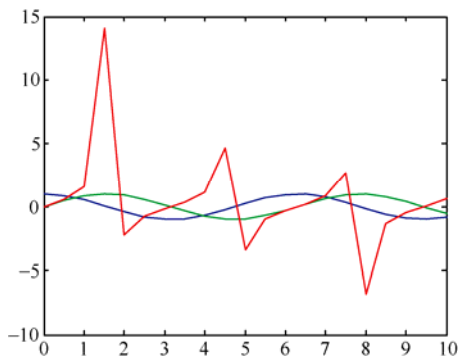


图 4-8 plot 函数绘制二维图形

(4) plot 函数还可以绘制具有不同线型、点标和颜色的图形，它的具体用法如下所示：

- plot(x, y, s): 其中 s 为字符，可以代表不同的线型、点标和颜色等，表 4-1 提供了常见的二维图形常用设置选项。

表 4-1 二维图形常用设置选项

选项	含义说明	选项	含义说明
-	实线	.	点
:	点线	o	圆
-.	点画线	+	加号
--	虚线	*	星号
Y	黄色	x	x 符号
M	紫红色	s	方形
C	蓝绿色	d	菱形
R	红色	v	下三角
G	绿色	^	上三角
B	蓝色	<	左三角
W	白色	>	右三角
K	黑色	p	正五边形

【例 4-9】使用 plot 函数分别绘制不同线型和颜色的二维图形。

在命令窗口中输入如下语句：

```
figure
X=0:0.1:5;
Y=cos(X);
subplot(2,2,1)
plot(X,Y,'-.')
title('点画线')
subplot(2,2,2)
plot(X,Y,'d')
title('菱形')
subplot(2,2,3)
plot(X,Y,'g-.')
title('绿色点画线')
subplot(2,2,4)
plot(X,Y,'-.g')
title('绿色点画线')
```

图形窗口中的输出结果如图 4-9 所示。

本例需要说明的是，figure 函数用于创建图形窗口，将在 4.4.1 小节详细讲解；title 函数用于标示图形的标题，将在 4.3.2 小节详细讲解；subplot 函数用于对图形窗口进行分割，将在 4.3.6 小节详细讲解。本例如果不使用 subplot 函数，需要绘制四幅图，来描述 plot 函数的绘图结果，而使用 subplot 函数将四个图形绘制到一幅图中，方便用户观察和比较不同之处。

plot 函数通过对 s 的设置可以绘制不同线型、点标和颜色的图形。s 可以单独使用，如第一幅图点画线和第二幅图菱形线中 s 的使用；s 也可以将线型、颜色和点标结合在一起使用，如第三幅图绿色点画线和第四幅图绿色点画线中 s 的使用。第三幅图和第四幅图的不同之处在于 s 中点标的顺序，可见 s 中线型、点标和颜色的顺序并不影响最终图形的绘制。

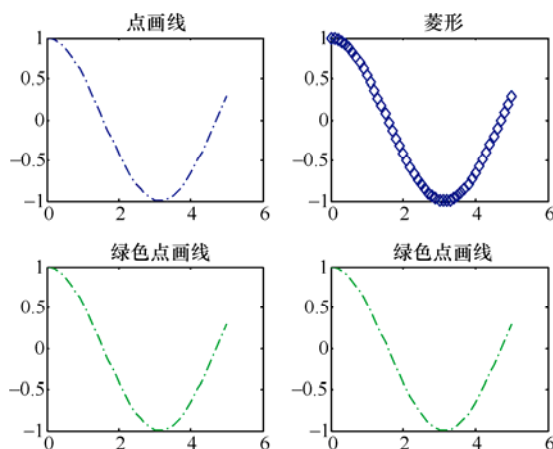


图 4-9 不同颜色和线型的二维图形

3. polar 函数

polar 函数是在极坐标系下进行绘图的函数，它的具体用法如下所示：

- **polar(theta,rho):** polar 函数以弧度为单位，功能与 plot 函数类似，但是 polar 函数不能对多组变量同时进行绘图。

【例 4-10】使用 polar 函数在极坐标系下绘制三叶和四叶玫瑰线。

(1) 用 polar 函数绘制三叶玫瑰线的 MATLAB 程序如下：

```
figure
t = 0:0.1:2*pi;
rho=sin(3*t);
polar(t,rho)
```

程序运行后绘制出的三叶玫瑰线如图 4-10 (a) 所示。

(2) 用 polar 函数绘制四叶玫瑰线的 MATLAB 程序如下：

```
figure
t = 0:0.1:2*pi;
rho=cos(2*t);
polar(t,rho)
```

程序运行后绘制出的四叶玫瑰线如图 4-10 (b) 所示。

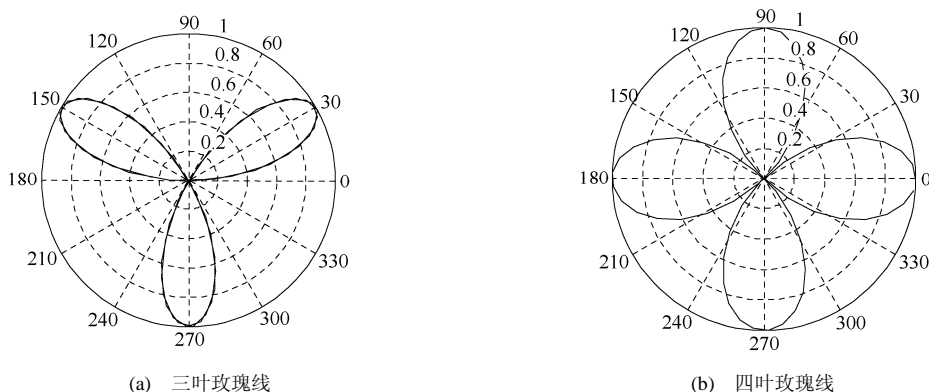


图 4-10 polar 函数绘图

【例 4-11】使用 `polar` 函数对多组变量同时进行绘图。

在命令窗口中输入如下语句：

```
figure
t=0:0.1:5;
y1=sin(X);
y2=cos(X);
polar(t,y1,t,y2)
```

命令窗口中的输出结果如下所示：

```
??? Error using ==> polar at 23
Too many input arguments.
```

从本例不难看出，`polar` 函数不能对多组变量同时进行绘图。

4. 对数坐标曲线

绘制对数坐标曲线的函数有三个：`semilogx()`、`semilogy()`、`loglog()`。这三个函数的输入参数与 `plot()` 函数完全类似，这里不再赘述。不同之处如下所示：

- `semilogx()`、`semilogy()` 分别将以 X 轴与 Y 轴为对数坐标。
- `loglog()` 是双对数坐标，即 X 轴与 Y 轴都是对数坐标。

【例 4-12】已知对数组 $x=y=0:1:1000$ ，试用 `semilogx()` 函数绘制其曲线。

在命令窗口中输入如下语句：

```
figure
x =0:1:1000;
y =0:1:1000 ;
semilogx(x,y)
```

图像窗口中的输出结果如图 4-11 所示。

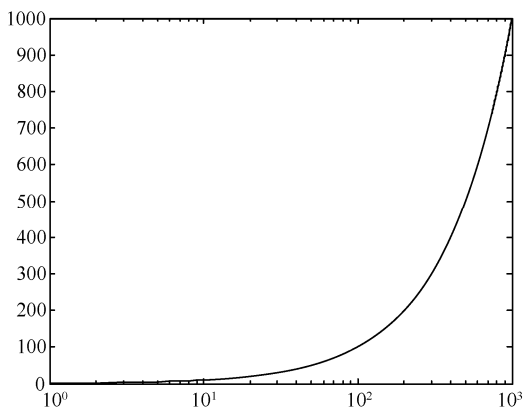


图 4-11 绘制的曲线

4.1.2 特殊函数

在一些特定场合，`plot` 函数绘制的线条图不能满足要求，如气象工作者需要显示若干地区的平均温度，销售人员需要了解每年的销售量等。为了满足实际的要求，MATLAB 提供了一些专门用于绘制特殊图形（如柱状图、饼图和直方图等）的函数，表 4-2 总结了常见的绘制特殊二维图形的函数。

表 4-2 绘制特殊二维图形函数

函数名	功能介绍	函数名	功能介绍
area	填充绘图	fplot	函数绘制
bar	柱状图	hist	直方图
barh	水平柱状图	pareto	Pareto 图
comet	彗星图	pie	饼图
errorbar	误差带图	plotmatrix	分散矩阵绘制
ezplot	简单绘制函数图	ribbon	三维图的二维条状显示
ezpolar	简单绘制极坐标图	scatter	散射图
feather	矢量图	stem	离散序列火柴杆状图
fill	多边形填充	stairs	阶梯图
gplot	拓扑图	rose	极坐标系下的直方图
compass	与 feather 函数类似的矢量图	quiver	向量场

1. 柱状图的绘制

柱状图主要应用在比较不同数据、显示单独数据点在总体中的比重和观察变量的变化趋势等方面。MATLAB 提供了 `bar` 函数和 `barh` 函数绘制柱状图，二者的区别是，`bar` 函数用于绘制竖直柱状图，`barh` 函数用于绘制水平柱状图。`bar` 函数的具体用法如下：

- `bar(x, y)`：在指定的横坐标 x 上画出 y ，其中 x 是向量， y 可以是向量或者矩阵。当 y 为向量时，在横坐标 x 上画出 y ，每个横坐标对应一个竖条；当 y 为 $m \times n$ 的矩阵时，`bar` 函数将画出 m 组竖条，每组有 n 个条。
- `bar(y)`，横坐标是默认值 $1:1:m$ 。
- `bar(x,y,width)`：通过 `width` 指定竖条宽度，竖条宽度的默认值是 0.8，如果竖条宽度大于 1，条与条之间将会重合。
- `bar(x, y, 'mode')`：通过改变参数 `mode`，可以接收不同的模式。参数 `mode` 有以下几种取值：`grouped` 产生组合的柱状图（默认情况）；`stacked` 产生堆叠的柱状图；`linespec` 指定条的颜色。
- `h = bar(...)`：返回一个 `patch` 图形对象句柄的向量。

`barh` 函数的具体用法与 `bar` 函数类似，这里不再赘述。

【例 4-12】绘制水平柱状图和竖直柱状图。

在命令窗口中输入如下语句：

```
Y=[5 2 1;8 7 3;9 8 6];
subplot(3,1,1)
bar(Y)%条形的宽度为默认值 0.8
subplot(3,1,2)
bar(Y,1.5)%设置条形宽度为 1.5，条形会出现交叠
subplot(3,1,3)
barh(Y)%二维水平柱状图
```

图形窗口中的输出结果如图 4-12 所示。

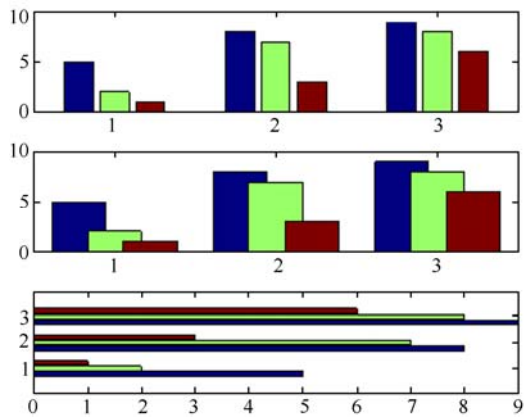


图 4-12 竖直柱状图和水平柱状图的比较

【例 4-13】使用 bar 函数绘制柱状图。

在命令窗口中输入如下语句：

```
figure
x=1:5;
y=rand(5,2);
colormap(summer)
subplot(2,1,1)
bar(x,y,'grouped')
subplot(2,1,2)
bar(x,y,'stacked')
```

图形窗口中的输出结果如图 4-13 所示。

本例分别画出了组合和堆叠两种形式的柱状图，需要说明的是 y 的行数必须与向量 x 的长度相等。另外，colormap 函数用来改变当前颜色的色调，内置的色调种类如图 4-14 所示。

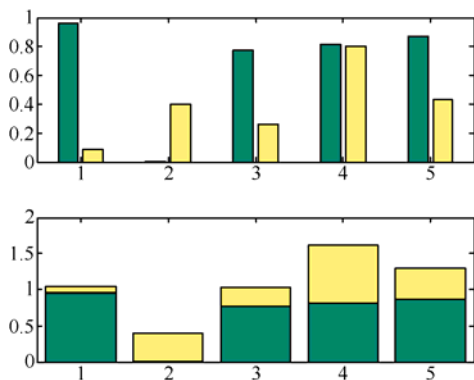


图 4-13 两种形式的柱状图比较

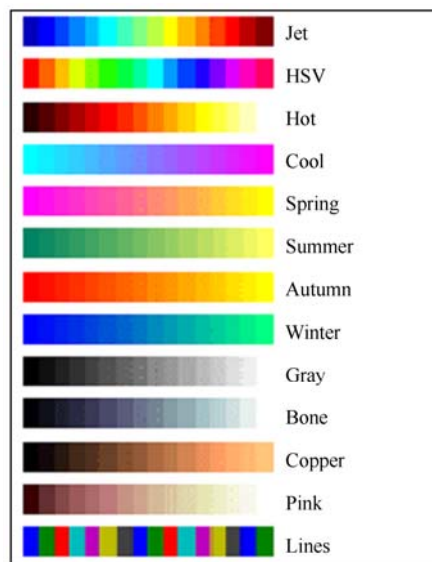


图 4-14 colormap 函数内置的色调种类

2. 饼图的绘制

在 MATLAB 中，饼图用于显示矢量或矩阵中的每一元素在所有元素的总和中所占的百分比。MATLAB 提供了 `pie` 函数用于绘制二维饼图，它的具体用法如下所示：

- `pie(x)`: 绘制向量 `x` 的饼图。
- `pie(x,explode)`: 向量 `explode` 与 `x` 的长度相同，若其中有非零元素，`x` 矩阵中相应位置的元素在饼图中对应的扇形将向外移出。
- `pie(x,labels)`: `labels` 用于定义相应块的标签。

【例 4-14】使用 `pie` 函数绘制饼图。

在命令窗口中输入如下语句：

```
figure
x=rand(1,7);
y=[0.32 0.11 0.28 0.22];
figure;
subplot(1,2,1)
pie(x)
subplot(1,2,2)
pie(y)
```

图形窗口中的输出结果如图 4-15 所示。

本例需要说明的是，当输入数据的总和超过 1 时，`pie` 函数自动计算每一部分在总体中所占的比例；当输入数据的总和不超过 1 时，`pie` 函数只绘制输入数据指定的各部分，不到 1 的部分作为空缺。

【例 4-15】使用 `pie` 函数绘制饼图。

在命令窗口中输入如下语句：

```
figure
x=rand(1,6);
explode = [2 1 1 0 0 3];
pie(x,explode)
```

图形窗口中的输出结果如图 4-16 所示。

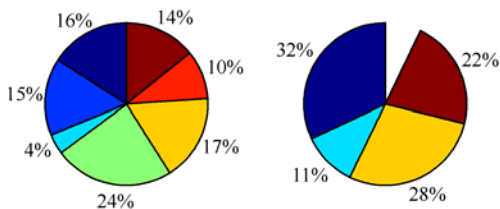


图 4-15 饼图的绘制

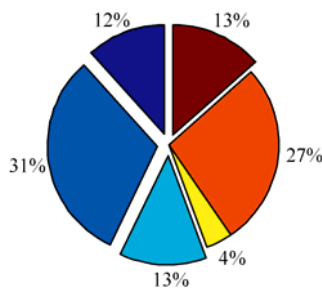


图 4-16 饼图的绘制

注意，向量 `explode` 与 `x` 的长度相同均为 6，其中有非零元素，表现为 `x` 矩阵中的相应位置的元素在饼图中对应的扇形向外移出。

【例 4-16】使用 `pie` 函数绘制饼图。

在命令窗口中输入如下语句：

```
figure
x=[1 2 3 4];
pie(x,{'A','B','C','D'})
```

图形窗口中的输出结果如图 4-17 所示。

3. 直方图的绘制

直方图经常用来显示数据的分布规律。MATLAB 提供了 `hist` 函数和 `rose` 函数绘制直方图，其中 `hist` 函数用于在直角坐标系下绘制直方图，`rose` 函数用于在极坐标系下绘制直方图。

`hist` 函数的具体用法如下：

- `n=hist(y)`: 使用 10 个等距区间统计向量 `y` 的分布，返回区间所含向量 `y` 中元素的个数。
- `n=hist(y,m)`: 使用 `m` 个等距区间统计向量 `y` 的分布。
- `n=hist(y,x)`: 参量 `x` 为向量，把 `y` 中元素放到 `m` (`m=length(x)`) 个由 `x` 中元素指定的位置为中心的条形中。
- `hist(...)`: 直接绘制直方图，不带返回值的调用格式。

`rose` 函数用于在极坐标系下绘制直方图，其用法和 `hist` 函数类似。需要注意的是，`rose` 函数中的数据是以弧度为单位的。

【例 4-17】使用 `hist` 函数绘制直方图。

在命令窗口中输入如下语句：

```
figure
x = -5:0.1:5;
y = randn(200,1);
hist(y,x)
```

图形窗口中的输出结果如图 4-18 所示。

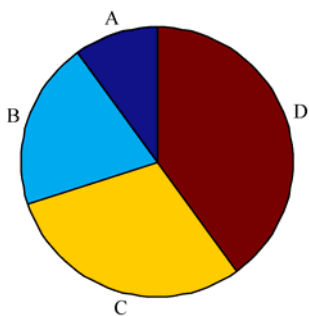


图 4-17 饼图的绘制

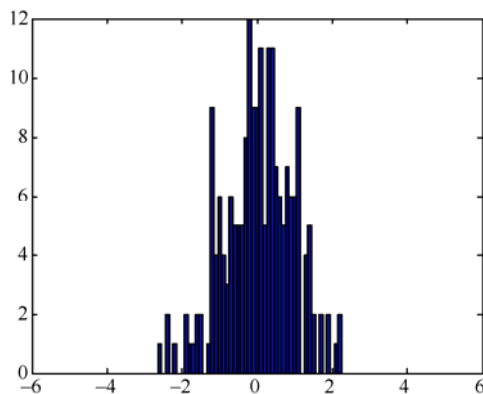


图 4-18 直方图的绘制

4. 等高线图的绘制

等高线图经常用来显示多元函数值的变化趋势。MATLAB 提供了 `contour` 函数和 `contourf` 函数用于绘制等高线图，其中 `contour` 函数绘制一般等高线图，`contourf` 函数绘制填充模式的等高线图。

`contour` 函数的具体用法如下所示：

- `contour(Z)`: 绘制数值矩阵 `Z` 的等高线图，等高线的数目和数值由系统根据数值矩阵 `Z` 自动确定。
- `contour(Z, n)`和 `contour(Z, v)`: `n` 和 `v` 是可选的输入变量，`n` 是所绘图形等高线的条数，

即按指定长度绘制等高线。 v 为一数值向量，等高线条数等于该向量的长度，且等高线的值对应向量的元素值。

- `contour(X,Y,Z)`, `contour(X, Y, Z, n)`, `contour(X, Y, Z, v)`: 绘制数值矩阵 Z 的等高线图，坐标值由矩阵 X 和 Y 指定，且矩阵 X 、 Y 、 Z 的维数相同。
- `[c, h] = contour(...)`: c 为等高线矩阵， h 为等高线的句柄。
- `contour(..., LineSpec)`: 利用指定的线型绘制等高线图。

这里需要说明的是，`clabel` 函数用于在等高线图中添加高度值，配合 `contour` 等函数使用，它的具体用法如下所示：

- `clabel(c,h)`: c 为等高线矩阵， h 为等高线的句柄。

`contourf` 函数的具体用法与 `contour` 函数类似，这里不再赘述。

【例 4-18】使用 `contour` 函数绘制等高线图。

在命令窗口中输入如下语句：

```
figure
z=peaks;
subplot(2,2,1)
contour(z,3)
title('contour(Z,n)')
subplot(2,2,2)
[c,h] = contour(z);
clabel(c,h)
title('contour(Z)')
subplot(2,2,3)
contourf(z,[0.3 1.5])
title('contourf(Z,[1])')
subplot(2,2,4)
[c,h] = contourf(z);
clabel(c,h)
title('contourf(Z)')
```

图形窗口中的输出结果如图 4-19 所示。

例子中调用了函数 `peaks`，这个函数是 MATLAB 中固有的，其具体的形式是：

$$z = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$
，在 MATLAB 命令窗口输入 `help`

`Peaks` 可以查看关于这个函数的信息。

5. 误差棒形图的绘制

误差棒形图经常用于显示工程中的大量误差数据。MATLAB 提供了 `errorbar` 函数用于绘制误差棒形图，它的具体用法如下所示：

- `errorbar(X,Y,E)`: X 、 Y 和 E 必须是同型参量。若同为向量，在点 $(X(i), Y(i))$ 处画出向下长为 E ，向上长为 E 的误差棒；若同为矩阵，则在曲面点 $(X(i,j), Y(i,j))$ 处，画出长度为 $E(i,j)$ 的误差棒。
- `errorbar(X,Y,L,U)`: X 、 Y 、 L 、 U 必须是同型参量。若同为向量，在点 $(X(i), Y(i))$ 处画出向下长为 $L(i)$ ，向上长为 $U(i)$ 的误差棒；若同为矩阵，则在点 $(X(i,j), Y(i,j))$ 处画出向下长

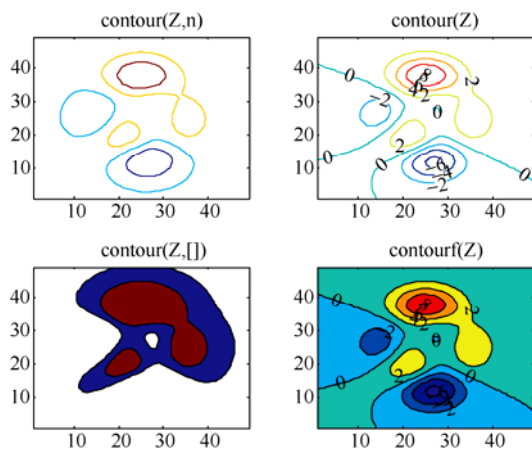


图 4-19 绘制等高线图

为 $L(i,j)$, 向上长为 $U(i,j)$ 的误差棒。

- `errorbar(...,LineStyle)`: 用 `LineStyle` 设定线型、标记符和颜色等画出误差棒。
- `h = errorbar(...)`: 返回线图形对象的句柄向量给 `h`。

【例 4-19】使用 `errorbar` 函数绘制误差棒形图。

在命令窗口中输入如下语句：

```
figure
x = -1:0.1:1;
y = cos(x);
e = rand(size(x))/10;
errorbar(x,y,e)
```

图形窗口中的输出结果如图 4-20 所示。

【例 4-20】使用 `errorbar` 函数绘制误差棒形图。

在命令窗口中输入如下语句：

```
figure
x = 0:0.5:2;
y = cos(x);
errorbar(x,y,rand(1,5),ones(1,5))
```

图形窗口中的输出结果如图 4-21 所示。

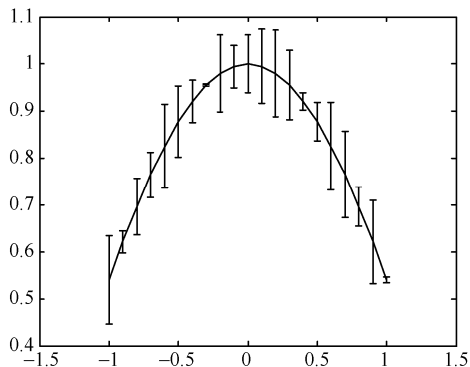


图 4-20 绘制误差棒形图

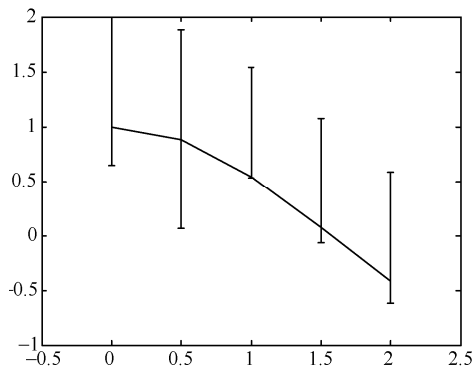


图 4-21 绘制误差棒形图

4.2 三维基本图形

在实际应用中,需要分析处理一些三维数据,这些数据往往比较抽象、复杂,利用 MATLAB 绘制三维图形可以很好地对这些数据进行分析。

MATLAB 具有强大的三维绘图能力,如绘制三维曲线、三维网格图和三维曲面图等,并提供了大量的三维绘图函数。绘制三维图形的过程与二维图形类似,不同的是三维图形可以设置和标注更多的元素,如光照、视角和颜色条等。

绘制三维图形的基本步骤如下:

(1) 准备数据

创建一般的数组。

(2) 设置当前绘图区

在指定的位置创建新的绘图窗口,系统自动以此窗口作为当前绘图区。

(3) 调用绘图指令

绘制三维曲线、三维网格图和三维曲面图。

(4) 设置视角

设置观察者查看图形的视角。

(5) 设置颜色表

通过颜色显示数值的变化。

(6) 设置光照效果

设置光源位置和类型。

(7) 设置坐标轴,标注图形

设置坐标轴的范围、刻度和坐标分格线等,并且对图形进行标注,如图名、坐标名、图例和文字说明等。

(8) 设置图形的曲线和标记点的形式

通过 set 函数设置图形曲线的颜色、线宽、线型和标记点的颜色、大小、形状等。

(9) 保存并导出图形

按照指定文件的属性、格式保存图形,配置打印机可以实现图形的导出。

4.2.1 基本绘图函数

最常见的是绘制三维曲线图、三维网格图和三维曲面图, MATLAB 相应提供了 plot3 函数、mesh 函数和 surf 函数进行绘制。

1. 三维曲线图的绘制

三维曲线图描述的是当自变量 x 和 y 沿着一条平面曲线变化的时候,函数值 z 随之变化的情况。MATLAB 提供了 plot3 函数用于绘制三维曲线图, plot3 函数是 plot 函数的扩展,它的具体用法如下所示:

- plot3(X,Y,Z): X 、 Y 和 Z 可以是向量或矩阵,尺寸必须相同。当 X 、 Y 和 Z 是尺寸相同的向量时, plot3 将绘制一条分别以向量 X 、 Y 和 Z 为 x 、 y 和 z 轴坐标值的空间曲线;当 X 、 Y 和 Z 均是 $m \times n$ 的矩阵时, plot3 将绘制 m 条曲线,其第 i 条曲线分别以 X 、 Y 和 Z 矩阵的第 i 列分量为 x 、 y 和 z 轴坐标值的空间曲线。

- `plot3(X,Y,Z,S)`: `S` 用来定义曲线线型、颜色和数据点等 (参见表 4-1), 其他要求同上。

【例 4-21】使用 `plot3(X,Y,Z)` 函数绘制三维螺旋曲线图。

在命令窗口中输入如下语句:

```
figure
t = 0:pi/50:20*pi;
plot3(sin(t),cos(2*t),sin(t)+cos(t))
grid //是否划分网格线的切换指令
```

图形窗口中的输出结果如图 4-22 所示。

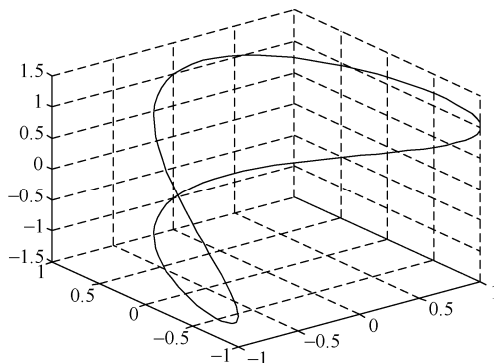


图 4-22 绘制三维曲线图

2. 三维网格图的绘制

三维网格图可以绘出在某一区间内完整的曲面, 而不是单根曲线。三维网格图是将邻近的网格顶点 (X, Y) 对应表面上的点 (X, Y, Z) 用线条连接起来形成的, 网格对应的曲面区域显示为空白。MATLAB 提供了 `mesh` 函数用于绘制三维网格图, 它的具体用法如下所示:

- `mesh(x,y,z,c)`: 绘制由 x 、 y 和 z 指定的参数曲面。 x 和 y 必须均为向量。若 x 和 y 的长度分别为 m 和 n , 则 z 必须为 $m \times n$ 的矩阵, c 是颜色映射数组, 决定图形的颜色。
- `mesh(x,y,z)`: 此方式相当于 $c=z$, 图形的颜色和网格高度成正比, 其他同上。
- `mesh(z)`和 `mesh(z,c)`: 没有指定 x 和 y 的平面区域, 系统自动赋予 $x=[1:n]$ 、 $y=[1:m]$ 使得 $[m,n]=\text{size}(z)$ 。

【例 4-22】使用 `mesh` 函数绘制三维网格图。

在命令窗口中输入如下语句:

```
figure
[x,y,z]=peaks(70);
subplot(2,1,1)
mesh(x,y,z)
title('1')
[x,y,z]=peaks(20);
subplot(2,1,2)
mesh(x,y,z)
title('2')
```

图形窗口中的输出结果如图 4-23 所示。

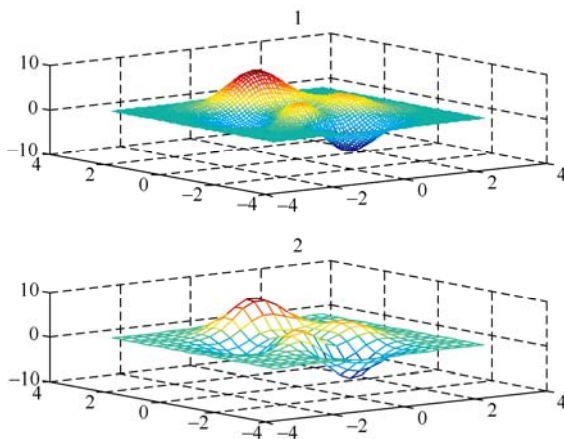


图 4-23 绘制三维网格图

从本例不难看出，`mesh` 函数通过颜色显示 z 值的大小。本例需要说明的是，`peaks` 函数用于生成三维高斯型分布的数据，`peaks` 函数的参数用于生成数据矩阵的维数。

MATLAB 还提供了两个与 `mesh` 函数相类似的函数，即 `meshc` 函数和 `meshz` 函数，它们的具体用法如下所示：

- `meshc(x,y,z)`：在绘制网格图的同时，在 X - Y 平面绘制函数的等值线。
- `meshz(x,y,z)`：在图形底部以及外侧绘制平行于 Z 轴的边框线。

【例 4-23】比较 `plots` 函数、`mesh` 函数、`meshc` 函数和 `meshz` 函数所绘制的图形。

在命令窗口中输入如下语句：

```
[x,y]=meshgrid(-5:0.5:5);
z=peaks(x,y);
figure
subplot(2,2,1)
plot3(x,y,z)
title('plot3')
subplot(2,2,2)
meshc(x,y,z)
title('meshc')
subplot(2,2,3)
meshz(x,y,z)
title('meshz')
subplot(2,2,4)
mesh(x,y,z)
title('mesh')
```

图形窗口中的输出结果如图 4-24 所示。

本例需要说明的是，`meshgrid` 函数用于生成数组，具体用法如下所示：

- `[X,Y]=meshcgrid(x,y)`：将 x 和 y 指定的区域转化为数组 X 和 Y 。
- `[X,Y]=meshcgrid(x)`：相当于 `[X,Y]=meshcgrid(x,x)`。
- `[X,Y,Z]=meshcgrid(x,y,z)`：用于三维数组。

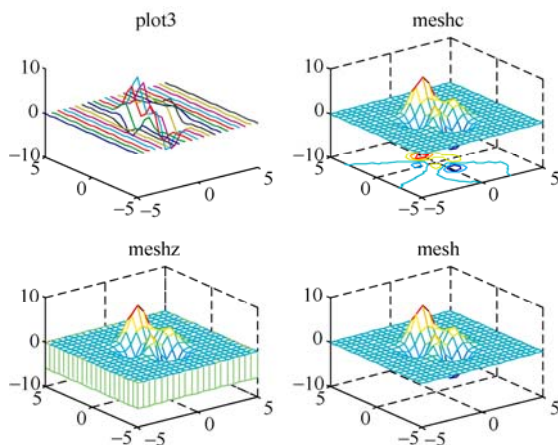


图 4-24 绘制三维网格图

3. 三维曲面图的绘制

三维曲面图是用颜色填充三维网格图表面网格围成的小区域，除了网格线条间的区域使用颜色填充外，三维曲面图和三维网格图效果是一样的，但三维曲面图更具有立体感。MATLAB 提供了 `surf` 函数用于绘制三维曲面图，其用法和 `mesh` 函数类似，它的具体用法如下：

- `surf(x, y, z, c)`: 各个参数的意义与 `mesh` 函数相同。
- `surf(x, y, z)`: 各个参数的意义与 `mesh` 函数相同。
- `surf(z)`和 `surf(z,c)`: 各个参数的意义与 `mesh` 函数相同。

需要说明的是，`c` 是具有三个分量的向量 `c=[cx, cy, cz]`，用来指明光线的方向。

【例 4-24】使用 `surf` 函数绘制三维曲面图。

在命令窗口中输入如下语句：

```
figure
[x,y]=meshgrid(-1:0.1:1);
z=peaks(x,y);
surf(x,y,z)
```

图形窗口中的输出结果如图 4-25 所示。

同样，MATLAB 还提供了两个与 `surf` 函数相类似的函数，即 `surfc` 函数和 `surfl` 函数，它们的具体用法如下所示：

- `surfc(x,y,z)`: 绘制具有基本等值线的曲线图。
- `surfl(x,y,z)`: 绘制有亮度的曲线图，其着色原理依据光照模型。

【例 4-25】分别使用 `surf` 函数、`surfc` 函数和 `surfl` 函数绘制三维曲面图并进行比较。

在命令窗口中输入如下语句：

```
[x,y]=meshgrid(-2:0.3:2);
z=peaks(x,y);
figure
subplot(3,1,1)
surf(x,y,z)
title('surf')
subplot(3,1,2)
```

```
surf(x,y,z)
title('surf')
subplot(3,1,3)
surfl(x,y,z)
title('surfl')
```

图形窗口中的输出结果如图 4-26 所示。

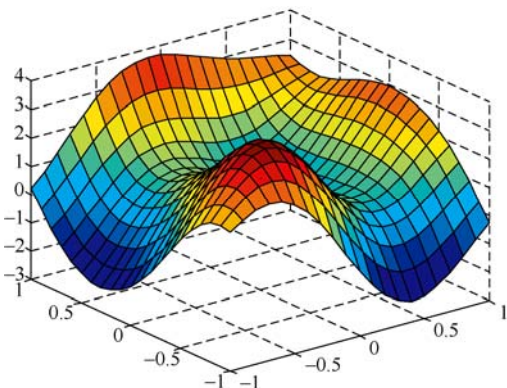


图 4-25 绘制三维曲面图

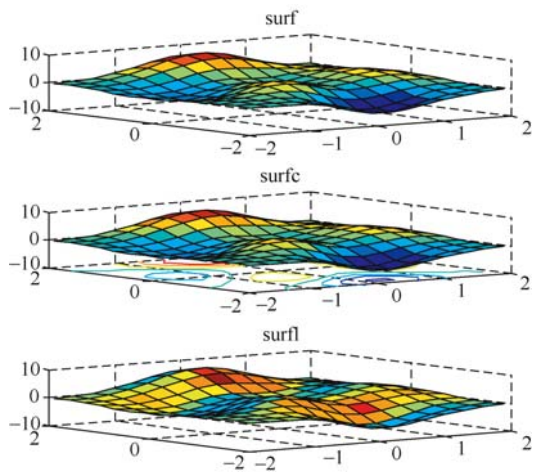


图 4-26 绘制三维曲面图

本例说明了三个绘制三维曲面函数的区别：

- surf: 对网线图的网格块区域着色得到刻画面。
- surfc: 和 meshc 类似，在刻画面下方绘上等值线。
- surfl: 对刻画面中单元颜色进行平滑处理，得到曲面图，使其更接近实体外观。

4.2.2 特殊函数

和绘制特殊二维图形相似，MATLAB 同样提供了绘制特殊三维图形的函数，如绘制三维柱状图、饼图和散点图等函数。

表 4-3 总结了常见的绘制特殊三维图形的函数。

表 4-3 特殊三维图形函数

函数名	说明	函数名	说明
bar3	三维柱状图	surfc	着色图与等高线图结合
comet3	三维彗星轨迹图	trisurf	三角形表面图
ezgraph3	函数控制绘制三维图	trimesh	三角形网格图
pie3	三维饼状图	waterfall	瀑布图
scatter3	三维散射图	cylinder	柱面图
stem3	三维离散数据图	sphere	球面图
quiver3	向量场	contour3	三维等高线

1. 三维柱状图的绘制

三维柱状图使用三维条柱表示每一个元素。MATLAB 提供了 bar3 和 bar3h 函数用于绘制三维柱状图。它们的用法与 bar 函数和 barh 函数类似，bar3 函数用于绘制竖直的三维柱状图，bar3h 函数用于绘制水平的三维柱状图。

【例 4-26】使用 `bar3` 函数和 `bar3h` 函数绘制三维柱状图，并比较二者的不同之处。
在命令窗口中输入如下语句：

```
x=rand(3,5)
figure
subplot(2,1,1)
bar3(x)
title('bar3(x)')
subplot(2,1,2)
bar3h(x)
title('bar3h(x)')
```

图形窗口中的输出结果如图 4-27 所示。

【例 4-27】分别绘制二维柱状图和三维柱状图并进行比较。
在命令窗口中输入如下语句：

```
x=-1:0.5:1;
y=rand(5,6);
figure
subplot(3,1,1)
bar(x,y)
title('bar')
subplot(3,1,2)
bar3(x,y)
title('bar3')
subplot(3,1,3)
bar3h(x,y)
title('bar3h')
```

图形窗口中的输出结果如图 4-28 所示。

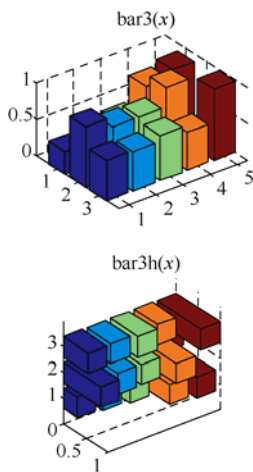


图 4-27 绘制三维柱状图

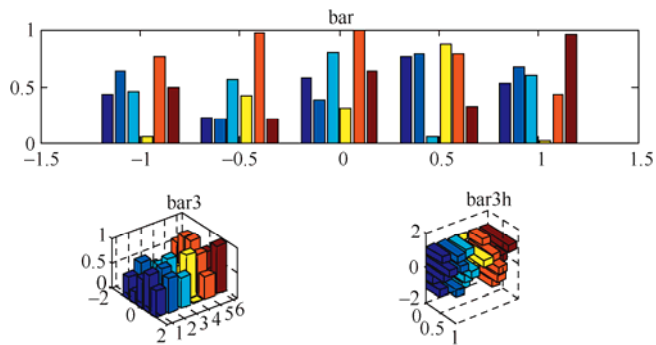


图 4-28 二维柱状图和三维柱状图的比较

2. 三维饼图的绘制

在统计学中，经常用饼形图来表示各个统计量占总量的份额，饼形图可以显示向量或矩阵中的元素占有所有元素总和的百分比。MATLAB 提供了 `pie3` 函数用于绘制三维饼图，`pie3` 函数的用法和 `pie` 函数类似，生成的效果立体感更强。

【例 4-28】使用 `pie3` 函数绘制三维饼图。

在命令窗口中输入如下语句：

```
x=rand(1,4);
explode = [0.32 0.11 0.28 0.15];
figure
subplot(1,2,1)
pie3(x)
title('A')
subplot(1,2,2)
pie3(x,explode)
title('B')
```

图形窗口中的输出结果如图 4-29 所示。

本例需要说明的是，向量 `explode` 与 `x` 的长度必须相同。

【例 4-29】使用 `pie` 函数和 `pie3` 函数分别绘制二维饼图和三维饼图。

在命令窗口中输入如下语句：

```
x=rand(1,4);
explode = [2 0 1 3];
figure
subplot(1,2,1)
pie(x,explode)
title('二维饼图')
subplot(1,2,2)
pie3(x,explode)
title('三维饼图')
```

图形窗口中的输出结果如图 4-30 所示。

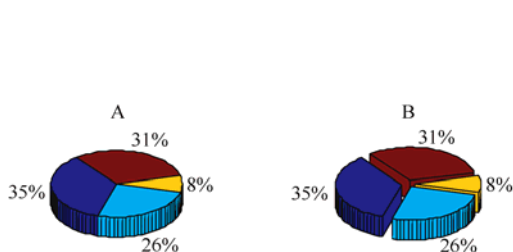


图 4-29 绘制三维饼图

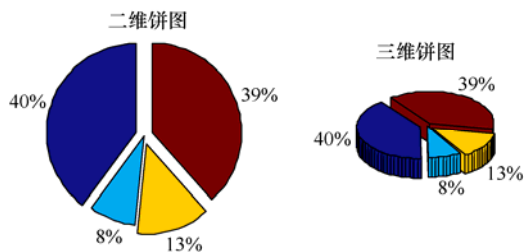


图 4-30 二维饼图与三维饼图的比较

3. 三维等高线图的绘制

等高线图常用来显示多元函数函数值的变化趋势，三维等高线图将等值线显示在平行于

X-Y 平面的每一个切面上。MATLAB 提供了 `contour3` 函数用于绘制三维等高线图，`contour3` 函数的用法和 `contour` 函数类似，只是生成的效果更加直观和立体。

【例 4-30】使用 `contour3` 函数绘制三维等高线并与使用 `contour` 函数绘制二维等高线相比较。

在命令窗口中输入如下语句：

```
z=peaks;  
figure  
subplot(1,2,1)  
contour(z,40)  
title('二维等高线')  
subplot(1,2,2)  
contour3(z,40)  
title('三维等高线')
```

图形窗口中的输出结果如图 4-31 所示。

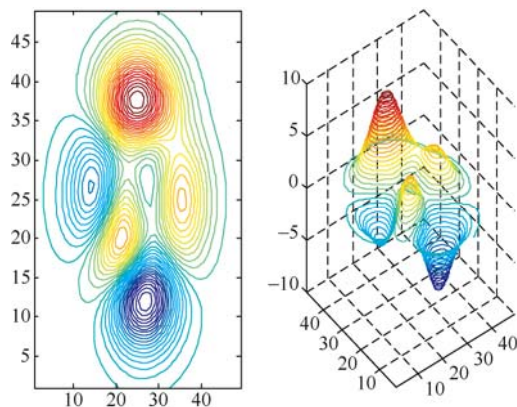


图 4-31 二维等高线与三维等高线的相比较

4.3 图形处理技术

MATLAB 提供了许多图形处理函数，用于处理前面绘制出的图形，以满足用户的更高级要求，使图形表达更直观，更方便用户观察和分析图形。

4.3.1 坐标轴的调整

MATLAB 提供了 `axis` 函数和 `zoom` 函数用于对所绘制图形的坐标轴进行调整，`axis` 函数可以控制坐标轴的刻度范围、控制坐标轴的特性，`zoom` 函数可以实现对二维图形坐标轴的缩放。

1. 控制坐标轴刻度范围

MATLAB 提供了 `axis` 函数用于控制坐标轴的刻度范围，它的具体用法如下所示：

- `axis([xmin xmax ymin ymax])`: 控制二维图形的坐标轴刻度范围，`xmin` 和 `xmax` 是图形 X 轴的限制范围，`ymin` 和 `ymax` 是图形 Y 轴的限制范围。在绘制图形的过程中，按照以上数据值确定坐标轴的刻度范围。
- `axis([xmin xmax ymin ymax zmin zmax])`: 控制三维图形的坐标轴刻度范围，`xmin` 和 `xmax`

是图形 X 轴的限制范围， $ymin$ 和 $ymax$ 是图形 Y 轴的限制范围， $zmin$ 和 $zmax$ 是图形 Z 轴的限制范围。

【例 4-31】使用 `axis` 函数控制坐标轴的刻度范围。

在命令窗口中输入如下语句：

```
figure
x=0:0.5:2;
y=rand(5);
bar(x,y)
title('水平柱状图')
axis([-0.5 2 0 1.5])           %使用 axis 函数控制坐标轴的刻度范围
```

图形窗口中的输出结果如图 4-32 所示。

不使用 `axis` 函数控制坐标轴的刻度范围，图形窗口中的输出结果如图 4-33 所示。

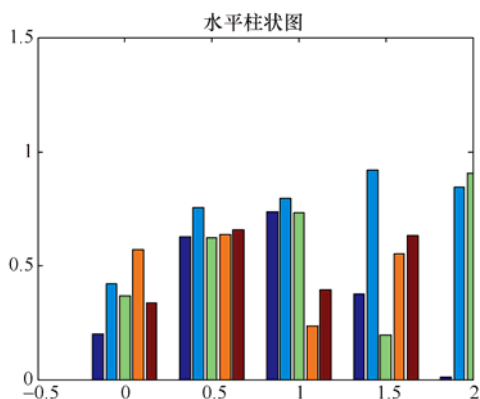


图 4-32 已控制坐标轴的刻度范围

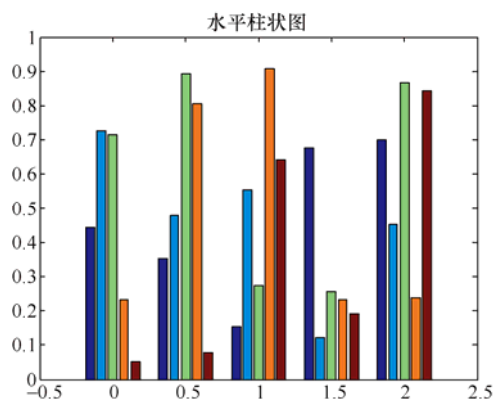


图 4-33 未控制坐标轴的刻度范围

2. 控制坐标轴特性

MATLAB 提供的 `axis` 函数还可以用来控制坐标轴的特性，它的具体用法如下所示：

- `axis('string')`：控制字符串 `string` 如表 4-4 所示。

表 4-4 `axis` 命令的控制字符串

字符串	功能介绍
<code>auto</code>	自动模式，使得坐标轴范围能容纳下所有的图形
<code>manual</code>	以当前的坐标范围限定图形的绘制，此后使用 <code>hold on</code> 命令再次绘图时保持坐标轴范围不变
<code>tight</code>	将坐标范围限制在指定的数据范围内
<code>fill</code>	设置坐标范围和 <code>PlotBoxAspectRatio</code> 属性以使坐标满足要求
<code>ij</code>	将坐标设置成矩阵形式，原点在左上角
<code>xy</code>	将坐标设置成直角坐标系
<code>equal</code>	将各坐标轴的刻度设置成相同
<code>Image</code>	与 <code>equal</code> 类似
<code>square</code>	设置绘图区为正方形
<code>vis3d</code>	使图形在旋转或拉伸过程中保持坐标轴的比例不变
<code>normal</code>	解除对坐标轴的任何限制
<code>off</code>	取消对坐标轴的一切设置
<code>on</code>	恢复对坐标轴的一切设置

【例 4-32】使用 `axis` 函数控制坐标轴的特性，以满足设置绘图区为正方形的要求。
在命令窗口中输入如下语句：

```
figure
x=0:0.5:2;
y=rand(5);
bar(x,y)
axis('square')
```

图形窗口中的输出结果如图 4-34 所示。

3. 控制坐标轴缩放

通过调整坐标轴的比例可以控制坐标轴的缩放，MATLAB 提供了 `zoom` 函数控制二维图形坐标轴的缩放，它的具体用法如下所示：

- `zoom('string')`：控制字符串 `string` 如表 4-5 所示。

表 4-5 zoom 命令的控制字符串	
字符串	功能介绍
空	在 <code>zoom on</code> 和 <code>zoom off</code> 之间切换
<code>factor</code>	以 <code>factor</code> 作为缩放因子进行坐标轴的缩放
<code>on</code>	允许对坐标轴进行缩放
<code>off</code>	禁止对坐标轴进行缩放
<code>out</code>	恢复到最初的坐标轴设置
<code>reset</code>	设置当前的坐标轴为最初值
<code>xon</code>	允许对 <i>X</i> 轴进行缩放
<code>yon</code>	允许对 <i>Y</i> 轴进行缩放

需要说明的是，当 `zoom` 处于 `on` 状态时，可以通过鼠标对图形进行缩放。单击鼠标左键可以以指定点为基础将图形放大一倍，单击鼠标右键选择 `zoom out` 可以将图形缩小一倍，选择 `Reset to Original View` 可以恢复到图形的初始状态，双击鼠标左键也可以恢复到初始状态。

【例 4-33】使用 `zoom` 函数控制坐标轴的缩放。

在命令窗口中输入如下语句：

```
figure
x=1:5
y=rand(5,2)
bar(x,y)
title('竖直柱状图')
xlabel('x 轴')
ylabel('y 轴')
zoom on
```

图形窗口中的输出结果如图 4-35 所示。

单击鼠标左键以指定点为基础将图形放大一倍，图形窗口中的输出结果如图 4-36 所示。

单击鼠标右键选择 `zoom out` 将如图 4-36 所示的图形缩小一倍，图形窗口中的输出结果如图 4-37 所示。

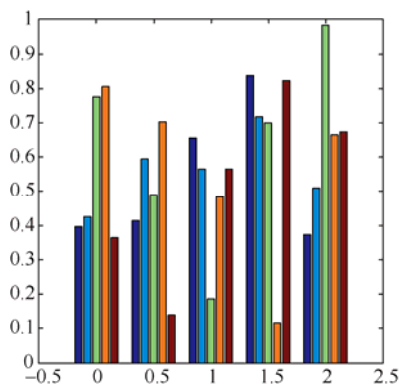


图 4-34 控制坐标轴的特性

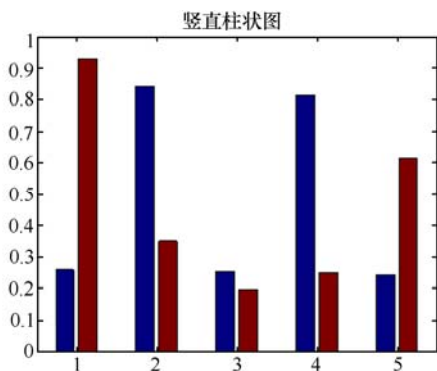


图 4-35 竖直柱状图

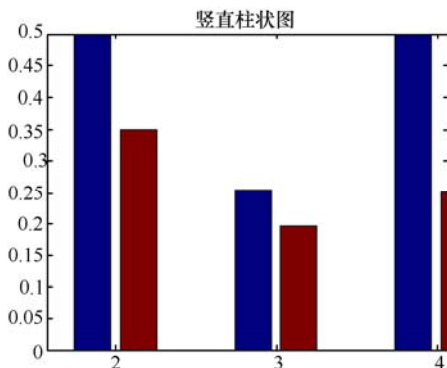


图 4-36 竖直柱状图放大一倍

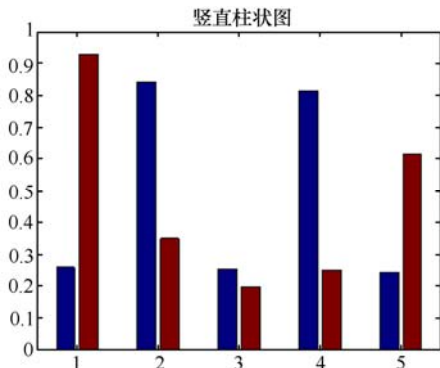


图 4-37 竖直柱状图缩小一倍

4.3.2 文字标示

MATLAB 提供了标题和坐标轴标示和文本标示的文字标示方式,表 4-6 列出了几种常用的图形标注函数及其功能,利用这些函数可以为图形加标题,为图形的坐标轴加标注,为图形加图例,也可以把说明、注释文本放到图形的任何位置。

表 4-6 常用图形标注函数

函数	函数功能
title	为图形添加标题
xlabel	为 X 轴添加标注
ylabel	为 Y 轴添加标注
zlabel	为 Z 轴添加标注
legend	为图形添加图例
text	在指定位置添加文本
otext	用鼠标在图形上放置文本

1. 标题和坐标轴标示

MATLAB 提供了 title 函数、xlabel 函数和 ylabel 函数用于标示图形的标题和坐标轴,它们的具体用法如下所示:

- title('string'): 标示图形的标题,字符串中可以添加由“\”引导的表示字符串特征的符号。
- xlabel('string'): 标示图形的 X 轴,字符串要求同上。
- ylabel('string'): 标示图形的 Y 轴,字符串要求同上。

- `zlabel('string')`: 标示图形的 Z 轴，字符串要求同上。
- 以上三个函数我们在前面的小节中已经用过，这里为避免赘述，不再另外给出例子说明。

2. 文本标示

MATLAB 提供了 `text` 函数和 `gtext` 函数在图形中添加文本，`text` 函数在命令中指定添加位置，`gtext` 函数允许用户通过鼠标在图形中选择添加位置。

(1) `text` 函数

使用 `text` 函数进行标示时，需要定义标示的文本字符串和添加位置，它的具体用法如下所示：

- `text(x,y,'string')`: 在 (x, y) 处添加文本。x 和 y 用于指定加入文本的位置，'string'是需要添加的文本内容。
- `text(x,y,z,'string')`: 在三维图形的指定位置添加文本。

表 4-7 提供了一些常用的表示字符串特征的符号。

表 4-7 特殊符号

字符名称	符号含义	字符名称	符号含义
<code>\alpha</code>	α	<code>\lambda</code>	λ
<code>\beta</code>	β	<code>\mu</code>	μ
<code>\gamma</code>	γ	<code>\xi</code>	ξ
<code>\delta</code>	δ	<code>\pi</code>	π
<code>\epsilon</code>	ϵ	<code>\omega</code>	ω
<code>\zeta</code>	ζ	<code>\tau</code>	τ
<code>\eta</code>	η	<code>\sigma</code>	Σ
<code>\theta</code>	θ	<code>\kappa</code>	κ
<code>\leftarrow</code>	\leftarrow	<code>\uparrow</code>	\uparrow

此外还可以对标注文本进行显示控制，具体方式如下：

- `\bf`: 黑体。
- `\it`: 斜体。
- `\sl`: 透视。
- `\rm`: 标准形式。
- `\fontname{fontname}`: 定义标注文字的字体。
- `\fontsize{fontsize}`: 定义标注文字的字体大小。

【例 4-34】使用 `text` 函数进行文本标示。

在命令窗口中输入如下语句：

```
figure
x=1:0.1*pi:2*pi;
y=sin(x);
plot(x,y)
xlabel('x(0-2\pi)');
ylabel('y=sin(x)','fontweight','bold');
title('正弦函数','fontsize',12,'fontweight','bold','fontname','隶书')
text(3*pi/4,sin(3*pi/4),'\leftarrowsin(t) = .707','FontSize',16)
text(pi,sin(pi),'\leftarrowsin(t) = 0','FontSize',16)
text(5*pi/4,sin(5*pi/4),sin(t) = -.707\rightarrow','HorizontalAlignment','right','FontSize',16)
```

图形窗口中的输出结果如图 4-38 所示。

(2) gtext 函数

gtext 函数通过鼠标来选择添加文本的位置，它的具体用法如下所示：

- gtext('string')：使用鼠标在指定处添加文本，文本内容通过'string'指定，'string'可以是字符串也可以是含有多个字符串的字符串数组。
- gtext({'string','PropertyName',PropertyValue,...})：'string'要求同上，PropertyName 为指定属性，PropertyValue 为指定属性的对应值。

【例 4-35】使用 gtext 函数进行文本标示。在命令窗口中输入如下语句：

```
figure
X=1:0.1*pi:2*pi;
Y=sin(X);
plot(X,Y)
xlabel('X')
ylabel('Y=sin(X)')
gtext('Y=sin(X)')
```

图形窗口中的输出结果如图 4-39 所示。

执行 gtext 函数后，会在图形上出现“+”形光标，在图形上任意移动光标，移到合适位置按下鼠标，则在此位置显示指定文本。

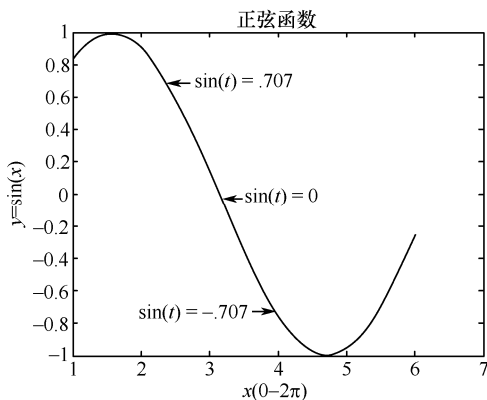


图 4-38 使用 text 函数进行文本标示

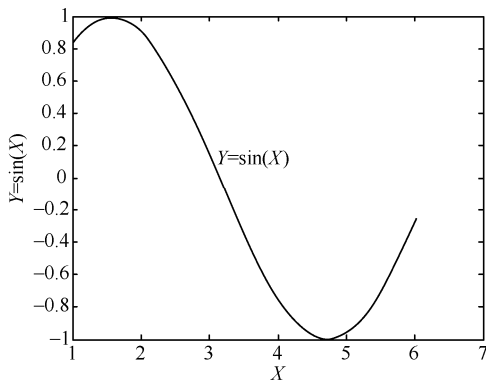


图 4-39 使用 gtext 函数进行文本标示

4.3.3 图例注解及添加颜色条

图例通过对每一条曲线标注不同的颜色和应用不同的线条，来区分一张图中绘制的多条曲线。颜色条主要用于显示图形中颜色和数值的对应关系，常用于三维图形和二维等高线图形中。

1. 图例注解

用户可以通过插入菜单的图例项 (Legend) 为曲线添加图例，也可以使用 legend 函数为曲线添加图例。

MATLAB 提供了 legend 函数用于为曲线添加图例，它的具体用法如下所示：

- legend('string1','string2',...): 其中'string1'和'string2'等分别用于设置各组数据对应的图例文字。
- legend(...,'Location',location): 'Location'和 location 用于定义标注的位置。
- legend('off'): 清除图例。

- legend('show'): 显示图例。
- legend('hide'): 隐藏图例。

表 4-8 提供了图例位置的标注定义。

表 4-8 图例位置标注定义

字符串	位置	字符串	位置
North	绘图区内的上中部	South	绘图区内的底部
East	绘图区内的右部	West	绘图区内的左中部
NorthEast	绘图区内的右上部	NorthWest	绘图区内的左上部
SouthEast	绘图区内的右下部	SouthWest	绘图区内的左下部
NorthOutside	绘图区外的上中部	SouthOutside	绘图区外的下部
EastOutside	绘图区外的右部	WestOutside	绘图区外的左部
NorthEastOutside	绘图区外的右上部	NorthWestOutside	绘图区外的左上部
SouthEastOutside	绘图区外的右下部	SouthWestOutside	绘图区外的左下部
Best	标注与图形重叠最少	BestOutside	绘图区外占用最小面积

需要说明的是，标注的位置还可以通过定位代号来定义，定位代号的说明如下：

- 0：自动定位，使得标注与图形重叠最少。
- 1：置于图形的右上方，默认值。
- 2：置于图形的左上方。
- 3：置于图形的左下方。
- 4：置于图形的右下方。
- -1：置于图形的右外侧。

【例 4-36】 使用 legend 函数为曲线添加图例并指定图例位置。

在命令窗口中输入如下语句：

```
figure
X=-1:0.1:10;
Y1=sin(X);
Y2=cos(X);
Y3=sin(X+2*pi);
Y=[Y1;Y2;Y3];
plot(X,Y1,'-ro',X,Y2,'--',X,Y3,'-b')
legend('sin(X)','cos(X)','sin(X+2*pi)','Location','Best')
```

图形窗口中的输出结果如图 4-40 所示。

需要说明的是，图例标注的位置可以用鼠标拖动来调整，以达到用户满意的效果。

2. 添加颜色条

用户可以通过插入菜单的颜色条项（Colorbar）为图形添加颜色条，也可以使用 colorbar 函数为图形添加颜色条。

【例 4-37】 使用 colorbar 函数为图形添加颜色条。

在命令窗口中输入如下语句：

```
figure
[x,y,z]=peaks(70);
mesh(x,y,z)
```

colorbar

图形窗口中的输出结果如图 4-41 所示。

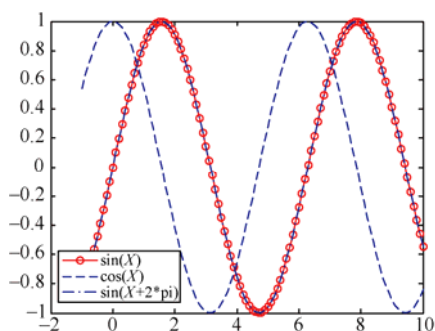


图 4-40 添加图例

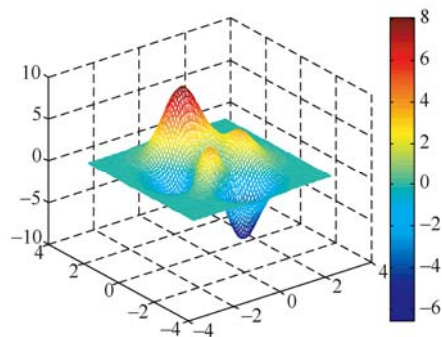


图 4-41 添加颜色条

【例 4-38】使用 colorbar 函数为图形添加颜色条。

在命令窗口中输入如下语句：

```
z=peaks;
figure
subplot(1,2,1)
contour(z,40)
title('二维等高线')
subplot(1,2,2)
contour3(z,40)
title('三维等高线')
colorbar
```

图形窗口中的输出结果如图 4-42 所示。

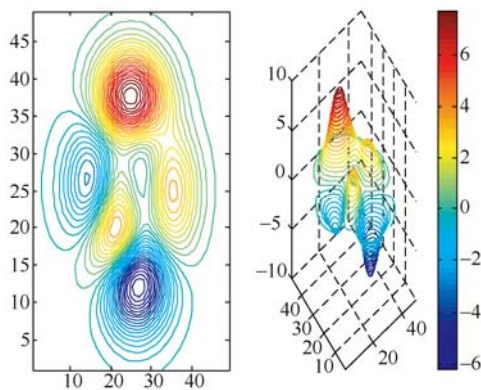


图 4-42 添加颜色条

4.3.4 图形的保持

MATLAB 提供了 hold 命令用来保持当前图形。系统默认的是在当前图形窗口中绘图，如果一个图形绘制完成后，需要继续绘图，系统会将原图形覆盖，并在原窗口中绘制图形。要想保持原有图形，并在图形中填加新的内容，就会用到 MATLAB 的保持当前图形功能。

- hold on: 保持当前图形。
- hold off: 解除 hold on 命令。
- hold: 在 hold on 和 hold off 之间切换。

【例 4-39】只使用 plot 函数画图而不使用 hold on 命令。

在命令窗口中输入如下语句：

```
X=-10:0.1:10;  
Y1=sin(X)+2;  
Y2=sin(X+2*pi);  
figure  
plot(X,Y1,'r')  
plot(X,Y2,'m-')  
xlabel('x')  
ylabel('y')
```

图形窗口中的输出结果如图 4-43 所示。

从上面的图形不难看出，使用 plot 函数绘制了两条曲线，而在图形中显示的只是最后一条曲线，为了能将多条曲线显示出来，可以使用 MATLAB 的 hold 命令保持当前图形。

【例 4-40】利用 plot 函数画图，使用 hold 命令保持当前图形。

在命令窗口中输入如下语句：

```
X=-10:0.1:10;  
Y1=sin(X)+2;  
Y2=sin(X+2*pi);  
figure  
plot(X,Y1,'r')  
xlabel('x')  
ylabel('y')  
hold on  
plot(X,Y2,'m-')
```

图形窗口中的输出结果如图 4-44 所示。

【例 4-41】针对上例，使用 hold off 命令。

在命令窗口中输入如下语句：

```
X=-10:0.1:10;  
Y1=sin(X)+2;  
Y2=sin(X+2*pi);  
figure  
plot(X,Y1,'r')  
xlabel('x')  
ylabel('y')  
hold off  
plot(X,Y2,'m-')
```

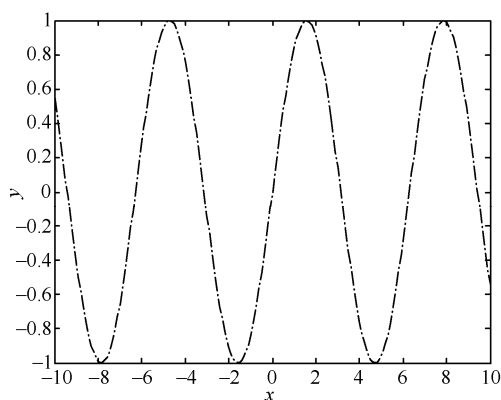


图 4-43 plot 函数画图

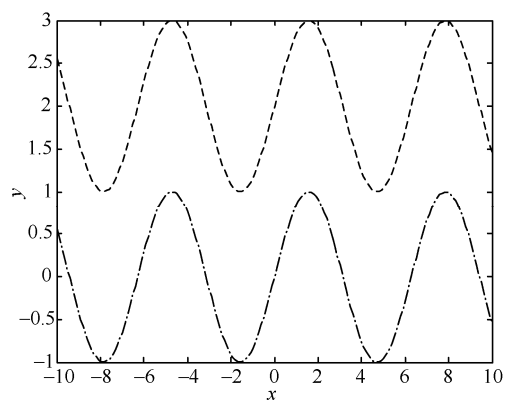


图 4-44 hold 命令保持当前图形

图形窗口中的输出结果如图 4-45 所示。

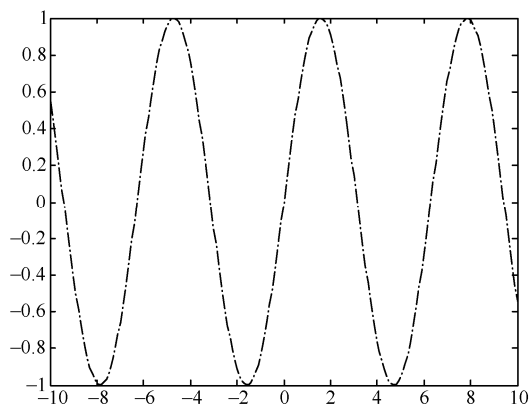


图 4-45 使用解除指令 hold off

4.3.5 网格控制及坐标轴封闭

MATLAB 提供了控制网格显示和坐标轴显示的函数，分别为 `grid` 函数和 `box` 函数，默认形式是不划分网格且坐标轴封闭。

1. grid 函数

MATLAB 提供了 `grid` 函数用于设置网格线，它的具体用法如下所示：

- `grid`: 是否划分网格线的切换指令。
- `grid on`: 添加网格线。
- `grid off`: 取消网格线。

2. box 函数

- `box`: 坐标形式在封闭和开启间切换。
- `box on`: 坐标呈封闭形式，默认形式。
- `box off`: 坐标呈开启形式。

【例 4-42】为图形添加网格线，并对比图形在使用 `box on` 和 `box off` 命令后的差异。

在命令窗口中输入如下语句：

```
figure
```

```
X=0:0.2*pi:2*pi;  
Y=3*sin(X);  
figure  
subplot(2,1,1)  
plot(X,Y);  
xlabel('x1')  
ylabel('y1')  
box on  
grid on  
title('坐标封闭')  
subplot(2,1,2)  
plot(X,Y)  
xlabel('x2')  
ylabel('y2')  
box off  
grid on  
title('坐标开启')
```

图形窗口中的输出结果如图 4-46 所示。

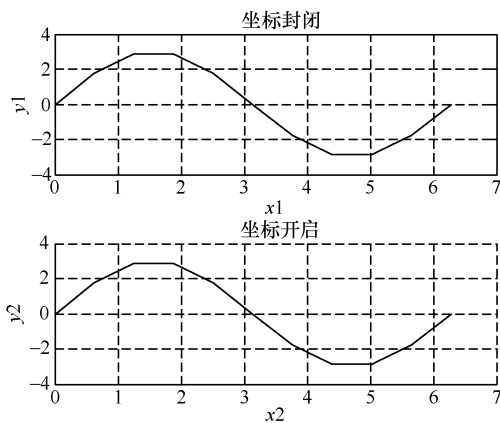


图 4-46 添加网格线和 box 命令的切换比较

4.3.6 图形窗口的分割

MATLAB 提供了 subplot 函数用于对图形窗口进行分割。在前面的章节中，我们已经使用过 subplot 函数，现在来详细讲解该函数的用法。subplot 函数的功能是将绘图窗口分割成多个矩形子区域，在指定的子区域绘图，它的具体用法如下所示：

- subplot(m,n,p): 将当前绘图窗口分割成 $m \times n$ 个子区域，并指定第 p 个编号区域是当前的绘图区域，区域编号的原则是“从上到下，从左到右”。若 p 是一向量，则将 p 指定的区域合并。
- subplot(m,n,p,'replace'): 如果指定区域已存在坐标系，则删掉已有的坐标系创建新坐标系。

- `subplot(m,n,p,'align')`: 将坐标系对齐。
- `subplot(h)`: 在句柄 `h` 指定的坐标系中绘图。
- `subplot('Position',[left bottom width height])`: 在由 4 个元素指定的位置上创建一坐标系。

【例 4-43】利用 `subplot` 函数将图形窗口分割为相等的两部分。

在命令窗口中输入如下语句：

```
figure
x=-30:pi/10:30;
subplot(2,1,1)
plot(x,sin(x))
xlabel('x1')
ylabel('y1')
grid
title('y=sin(x)')
subplot(2,1,2)
xlabel('x2')
ylabel('y2')
plot(x,cos(x))
title('y=cos(x)')
```

图形窗口中的输出结果如图 4-47 所示。

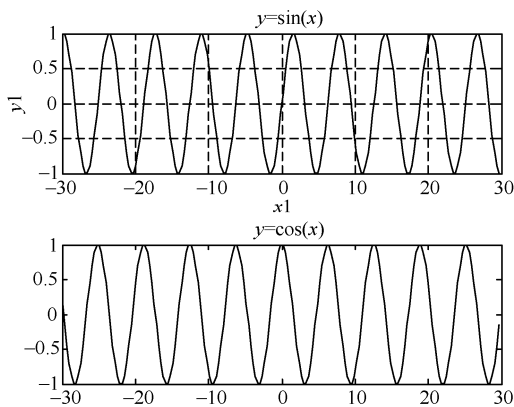


图 4-47 图形窗口的分割

【例 4-44】利用 `subplot` 函数将图形窗口分割为不均匀的 3 部分。

在命令窗口中输入如下语句：

```
figure
X=-1:0.1:10;
Y=[sin(X);cos(X);sin(X+2*pi)];
subplot(2,2,[1 3])
plot(X,Y)
xlabel('x')
ylabel('y=sin(X)')
```



```

title('图一')
subplot(2,2,2)
plot(X)
xlabel('x')
ylabel('y=cos(X)')
title('图二')
subplot(2,2,4)
plot(Y)
xlabel('x')
ylabel('y=sin(X+2*pi)')
title('图三')

```

图形窗口中的输出结果如图 4-48 所示。

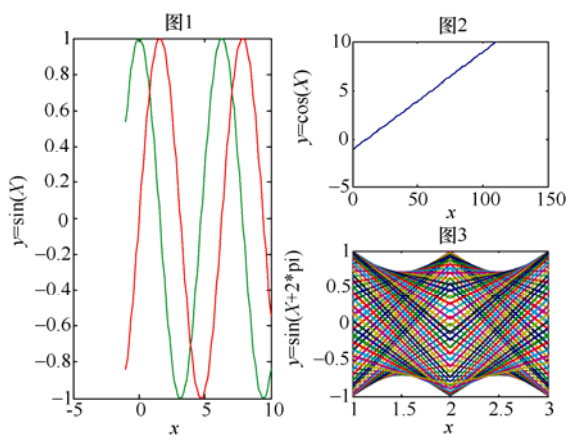


图 4-48 图形窗口的分割

4.4 图形窗口

MATLAB 的图形都是在图形窗口中绘制的，创建图形窗口有两种方式：一种是在绘制图形的时候，系统自动创建图形窗口，另外一种是采用创建图形窗口函数来创建图形窗口。图形窗口中包含菜单栏和工具栏，用户可以通过这些对图形对象进行操作。

4.4.1 图形窗口的创建与控制

1. 图形窗口的创建

MATLAB 提供了 **figure** 函数用于创建图形窗口，它的具体用法如下所示：

- **figure**：创建一个图形窗口。
- **figure(h)**：如果 **h** 句柄所对应的窗口对象已存在，则该命令使得该图形窗口成为当前窗口；如果不存在，则新建一个以 **h** 为句柄的窗口。
- **h = figure(...)**：返回图形窗口对象的句柄。

【例 4-45】使用 **figure** 函数建立一个新图形窗口。

在命令窗口中输入如下语句：

figure

图形窗口中的输出结果如图 4-49 所示。

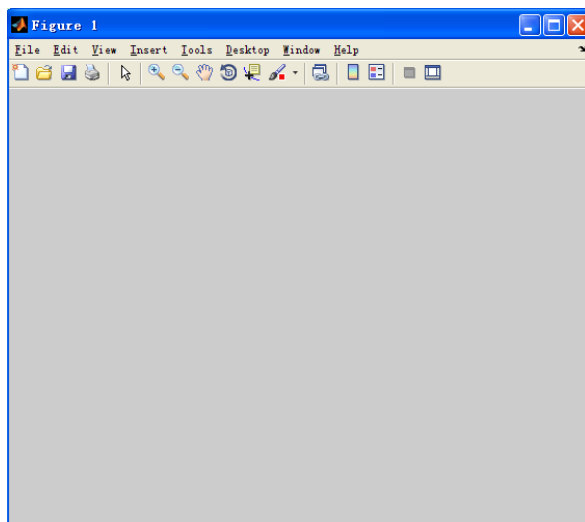


图 4-49 建立新的图形窗口

2. 图形窗口的控制

使用 figure 函数创建图形窗口后，要实现对图形窗口的控制，可以有两种方法：一种是使用属性编辑器，另外一种是使用 MATLAB 提供的 get 函数和 set 函数。

(1) 使用属性编辑器实现对图形窗口的控制

下面针对上例介绍如何使用属性编辑器，实现对图形窗口进行控制。

- 在图形窗口中选择菜单 View 中的 Property Editor 选项，激活属性编辑器，如需关闭属性编辑器，只要再选中菜单 View 中的 Property Editor 选项即可。激活属性编辑器的界面如图 4-50 所示，并可以在其中设置标题名、颜色表等属性。

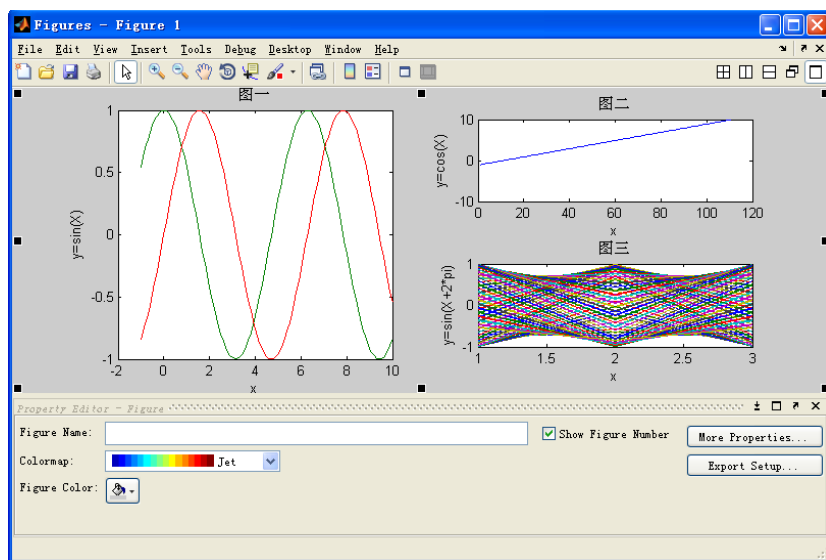


图 4-50 属性编辑器

- 如需设置更多属性，可选择如图 4-51 所示的 More Properties 选项，并可以在其中对图形窗口的属性进行设置。

(2) MATLAB 提供了 get 函数用于返回图形窗口的属性，它的具体用法如下所示：

- `v=get(h)`：返回句柄为 `h` 的图形窗口的所有属性值。
- `v=get(h,'PropertyName')`：返回 `PropertyName` 属性的值。

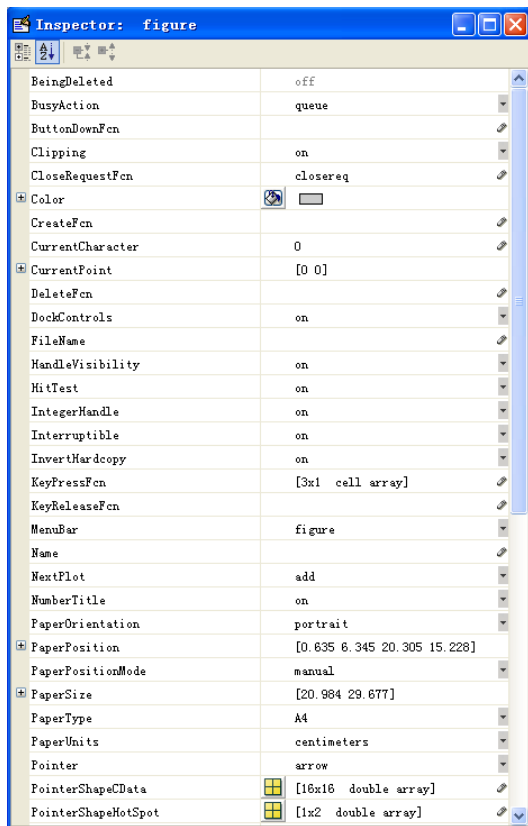


图 4-51 更多属性设置

- `v=get(0,'Factory')`和 `v=get(0,'FactoryObjectType Property Name')`：返回图形窗口属性的出厂设置。不同的是第一个函数返回图形窗口的所有属性设置，第二个函数返回图形窗口的指定属性设置。
- `v=get(h,'Default')`和 `v=get(h,'Default ObjectType Property Name')`：返回图形窗口的默认属性设置，二者的区别同上。

【例 4-46】使用函数 `figure()`和 `get()`函数实现图形窗口的创建与控制。

在命令窗口中输入如下语句：

```
figure
X=-1:0.1:10;
Y=[sin(X);cos(X);sin(X+2*pi)];
subplot(2,2,[1 3])
plot(X,Y)
xlabel('x')
```

```
ylabel('y=sin(X)')
title('图一')
subplot(2,2,2)
plot(X)
xlabel('x')
ylabel('y=cos(X)')
title('图二')
subplot(2,2,4)
plot(Y)
xlabel('x')
ylabel('y=sin(X+2*pi)')
title('图三')           %系统自动创建图形窗口 Figure 1
get(1)                  %返回指定图形的所有属性值
```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-52 的结果。

```
Alphamap = [ (1 by 64) double array]
CloseRequestFcn = closereq
Color = [0.8 0.8 0.8]
Colormap = [ (64 by 3) double array]
CurrentAxes = [185.002]
CurrentCharacter =
CurrentObject = [1]
CurrentPoint = [0 0]
DockControls = on
FileName =
IntegerHandle = on
InvertHardcopy = on
KeyPressFcn = [ (3 by 1) cell array]
KeyReleaseFcn =
MenuBar = figure
Name =
NextPlot = add
NumberTitle = on
PaperUnits = centimeters
PaperOrientation = portrait
PaperPosition = [0.634517 6.34517 20.3046 15.2284]
PaperPositionMode = manual
PaperSize = [20.984 29.6774]
PaperType = A4
Pointer = arrow
PointerShapeCDData = [ (16 by 16) double array]
```

```
PointerShapeHotSpot = [1 1]
Position = [1 1 712 252]
Renderer = painters
RendererMode = auto
Resize = on
ResizeFcn =
SelectionType = normal
ToolBar = auto
Units = pixels
WindowButtonDownFcn = [ (3 by 1) cell array]
WindowButtonMotionFcn =
WindowButtonUpFcn = [ (3 by 1) cell array]
WindowKeyPressFcn = [ (3 by 1) cell array]
WindowKeyReleaseFcn = [ (3 by 1) cell array]
WindowScrollWheelFcn =
WindowStyle = docked
WVisual = [ (1 by 94) char array]
WVisualMode = manual

BeingDeleted = off
ButtonDownFcn =
Children = [ (3 by 1) double array]
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [0]
Selected = on
SelectionHighlight = on
Tag =
Type = figure
UIContextMenu = []
UserData = []
Visible = on
```

(3) MATLAB 提供了 `set` 函数用于设置图形窗口的属性，它的具体用法如下所示：

- `a=set(h)`: 返回句柄为 `h` 的图形窗口中用户设置的属性，其中 `a` 是一个结构体，它的域名为属性名称，值为对应属性的设置值。

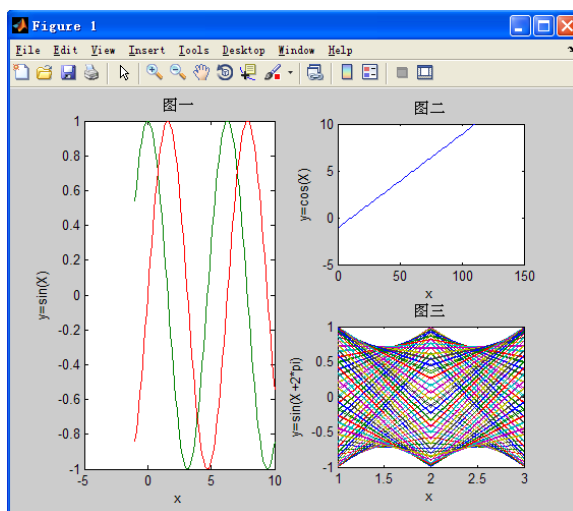


图 4-52 系统自动创建图形窗口 Figure 1

- `set(H,'Property Name', PropertyValue,...)`: 设置由 H 指定的窗口属性 Property Name 的值是 PropertyValue。
- `set(H,a)`: 设置 H 指定的窗口属性是 a, a 的要求同上。
- `a=set(0,'FactoryObjectType Property Name')`: 返回图形窗口指定属性的出厂设置。
- `a=set(h,'Default ObjectType Property Name')`: 返回图形窗口的默认属性设置。

【例 4-47】使用 set 函数返回图形窗口的属性值。

在命令窗口中输入如下语句：

```
figure
X=-5:0.1:5;
Y=[sec(X);sin(X);cos(X+pi/4)];
subplot(2,2,[1 3])
plot(X,Y)
xlabel('x')
ylabel('y=sec(X)')
title('图 1')
subplot(2,2,2)
plot(X)
xlabel('x')
ylabel('y=sin(X)')
title('图 2')
subplot(2,2,4)
plot(Y)
xlabel('x')
ylabel('y=cos(X+pi/4)')
title('图 3')
a=set(1)
```

% 系统自动创建图形窗口 Figure 1
% 返回指定图形的所有属性值

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 4-53 的结果。

a =

```
    Alphamap: {}
    BusyAction: {2x1 cell}
    ButtonDownFcn: {}
    Children: {}
    Clipping: {2x1 cell}
    CloseRequestFcn: {}
    Color: {'none'}
    Colormap: {}
    CreateFcn: {}
    CurrentAxes: {}
    CurrentCharacter: {}
    CurrentObject: {}
    CurrentPoint: {}
    DeleteFcn: {}
    DockControls: {2x1 cell}
    FileName: {}
    HandleVisibility: {3x1 cell}
    HitTest: {2x1 cell}
    IntegerHandle: {2x1 cell}
    Interruptible: {2x1 cell}
    InvertHardcopy: {2x1 cell}
    KeyPressFcn: {}
    KeyReleaseFcn: {}
    MenuBar: {2x1 cell}
    Name: {}
    NextPlot: {4x1 cell}
    NumberTitle: {2x1 cell}
    PaperOrientation: {3x1 cell}
    PaperPosition: {}
    PaperPositionMode: {2x1 cell}
    PaperSize: {}
    PaperType: {26x1 cell}
    PaperUnits: {4x1 cell}
    Parent: {}
    Pointer: {18x1 cell}
    PointerShapeCData: {}
    PointerShapeHotSpot: {}
```

```

Position: {}
Renderer: {4x1 cell}
RendererMode: {2x1 cell}
Resize: {2x1 cell}
ResizeFcn: {}
Selected: {2x1 cell}
SelectionHighlight: {2x1 cell}
SelectionType: {4x1 cell}
Tag: {}
ToolBar: {3x1 cell}
UIContextMenu: {}
Units: {6x1 cell}
UserData: {}
Visible: {2x1 cell}
WindowButtonDownFcn: {}
WindowButtonMotionFcn: {}
WindowButtonUpFcn: {}
WindowKeyPressFcn: {}
WindowKeyReleaseFcn: {}
WindowScrollWheelFcn: {}
WindowStyle: {3x1 cell}
WVisual: {}
WVisualMode: {2x1 cell}

```

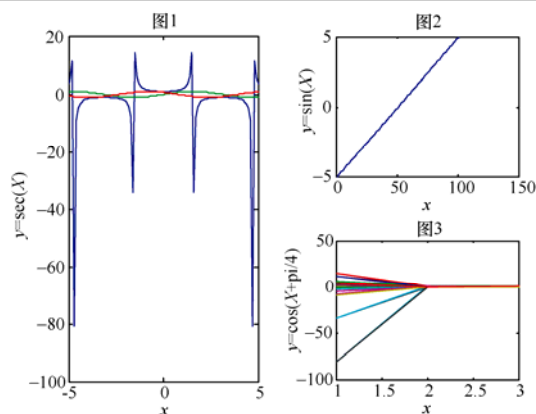


图 4-53 系统自动创建图形窗口 Figure 1

4.4.2 图形窗口的菜单操作

本小节介绍图形窗口中的常用菜单命令。

1. File 菜单

MATLAB 中 File 菜单的命令形式和 Windows 系统中 File 菜单的命令形式类似,包括 New、Open、Save、Close 和 Save As 等命令,如图 4-54 所示。

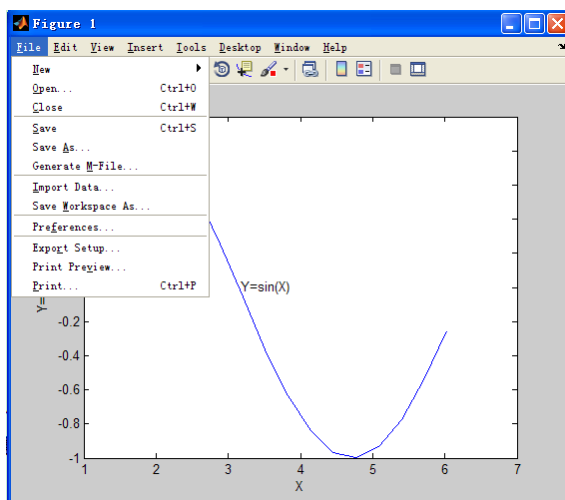


图 4-54 File 菜单的命令形式

File 菜单的具体命令功能如下：

- **New**: 新建 M 文件、图形窗口、变量图 and 图形用户接口，如图 4-55 所示。
- **Open**: 打开已经存在的文件。
- **Close**: 关闭当前窗口。
- **Save**: 保存文件。
- **Save As**: 另存文件。
- **Generate M-File**: 生成 M 文件。

【例 4-48】通过 Generate M-File 命令将图形窗口中的图形转化为 M 文件。

(1) 在生成的图形窗口中选择 Generate M-File 命令，如图 4-56 所示。

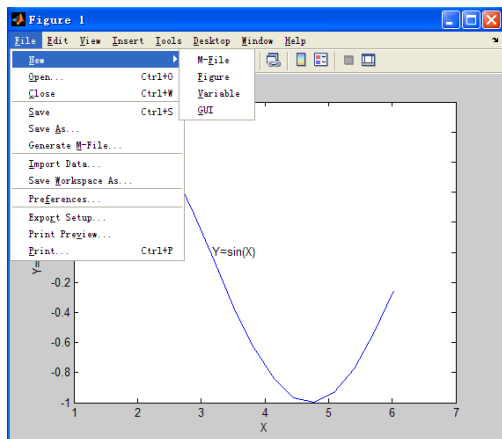


图 4-55 New 命令

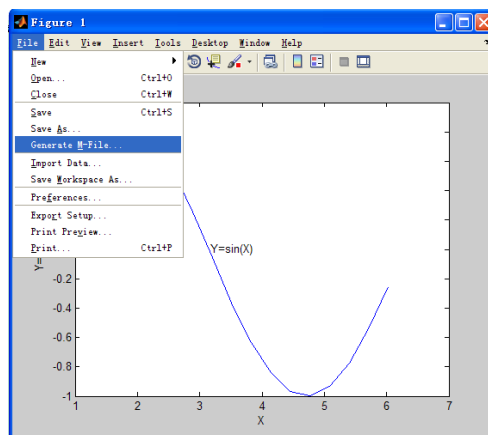


图 4-56 选择命令

(2) 生成如下所示的 M 文件。

```
function createfigure(X1,Y1)
%CREATEFIGURE(X1,Y1)
% X1: vector of x data
% Y1: vector of y data
```

```
% Auto-generated by MATLAB on 29-Aug-2010 11:17:33

% Create figure
figure1 = figure;

% Create axes
axes1 = axes('Parent',figure1);
box(axes1,'on');
hold(axes1,'all');

% Create plot
plot(X1,Y1);

% Create xlabel
xlabel('X');

% Create ylabel
ylabel('Y=sin(X)');

% Create text
text('Parent',axes1,'VerticalAlignment','baseline','String','Y=sin(X)',...
     'Position',[3.17741935483871 -0.0263157894736841 0]);
```

图 4-57 所示是 M 文件编辑器的 File 菜单。

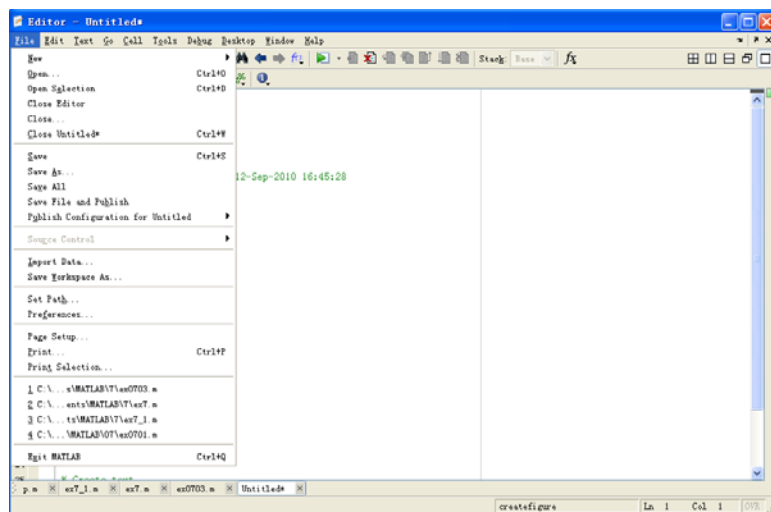


图 4-57 M 文件编辑器的 File 菜单

下面介绍一下 M 文件编辑器的 File 菜单命令：

- **Import Data:** 用于导入数据。
- **Save Workspace As:** 用于将图形窗口中的图形数据存储为二进制文件，以供其他的编程语言调用。

- Preferences: 定义图形窗口，如字体、颜色等，如图 4-58 所示。
- Page Setup: 进行页面设置，如图 4-59 所示。

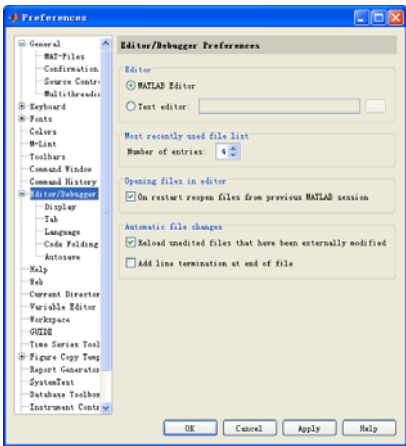


图 4-58 Preferences 对话框

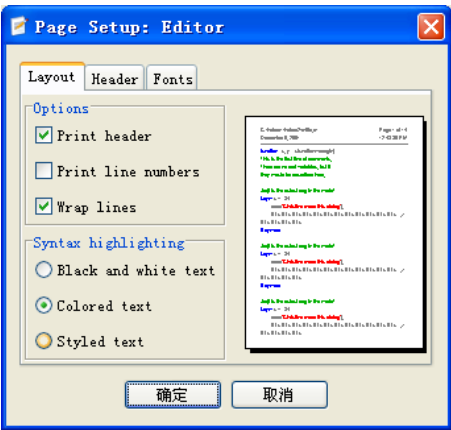


图 4-59 页面设置对话框

- Print: 打开打印对话框。

2. Edit 菜单

Edit 菜单中的命令如图 4-60 所示。

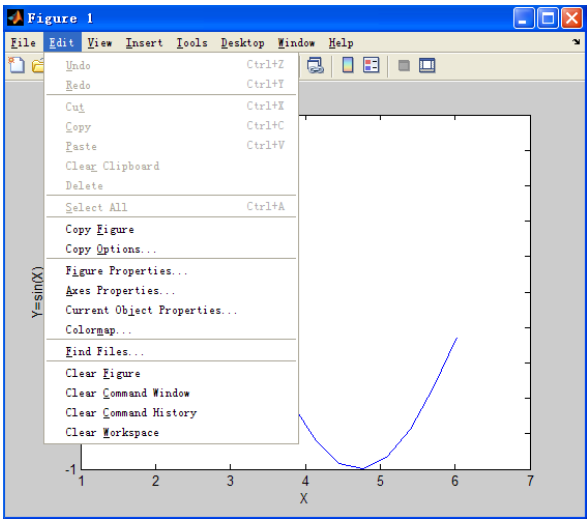


图 4-60 Edit 菜单

Edit 菜单的具体命令功能如下：

- Copy Figure: 用于复制图形。
- Copy Option: 用于打开复制设置对话框，可以设置图形复制的格式、图形背景颜色和图形大小等，如图 4-61 所示。
- Figure Properties: 用于打开图形窗口的属性设置对话框，如图 4-62 所示。可以对图形的属性进行设置，如图形窗口的标题、颜色映射表等。单击 More Properties 按钮可以获得更多属性设置，单击 Export Setup 按钮可以设置图形的导出属性。
- Axes Properties: 用于打开设置坐标轴属性对话框，如图 4-63 所示。

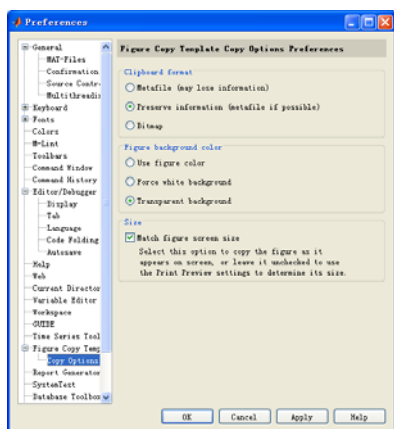


图 4-61 复制设置对话框

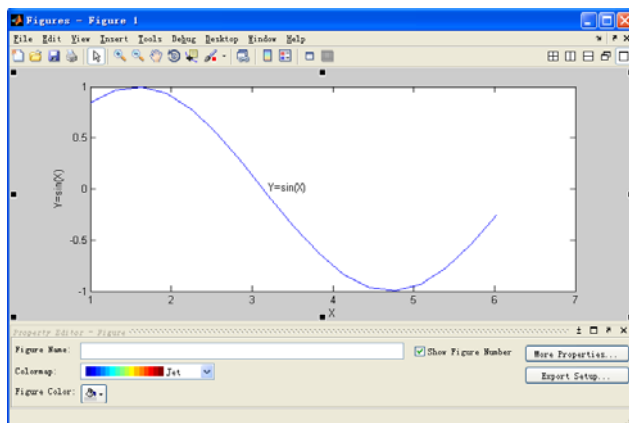


图 4-62 属性设置对话框

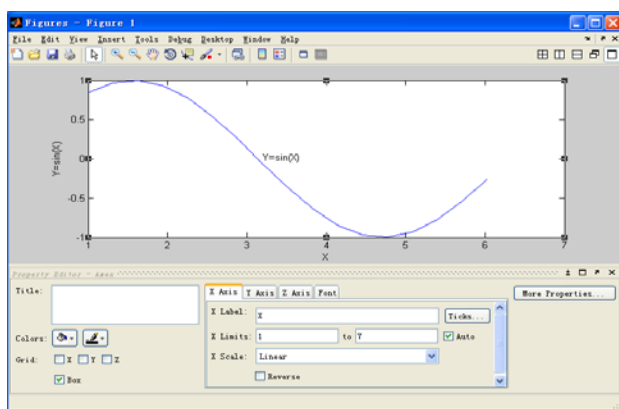


图 4-63 设置坐标轴属性对话框

- **Current Object Properties:** 用于打开设置图形窗口中当前对象（如窗口中的坐标轴、图形等）属性对话框，如图 4-64 所示。

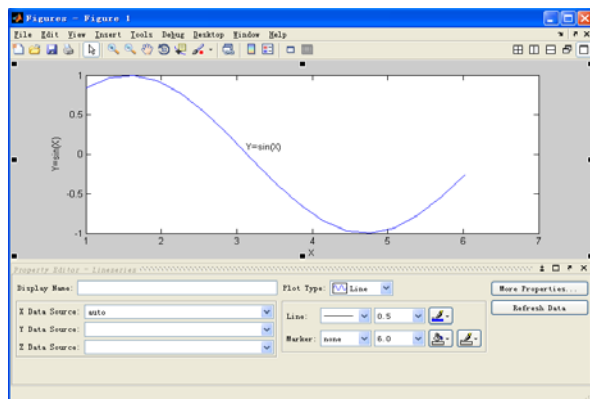


图 4-64 设置图形窗口中当前对象属性对话框

- **Colormap:** 用于打开色图编辑对话框，设置图形的颜色表，如图 4-65 所示。

3. Insert 菜单

Insert 菜单主要用于向当前图形窗口中插入各种标注图形，如坐标轴、箭头、标题、直线和图例等，表 4-9 给出了 Insert 菜单中的具体命令。

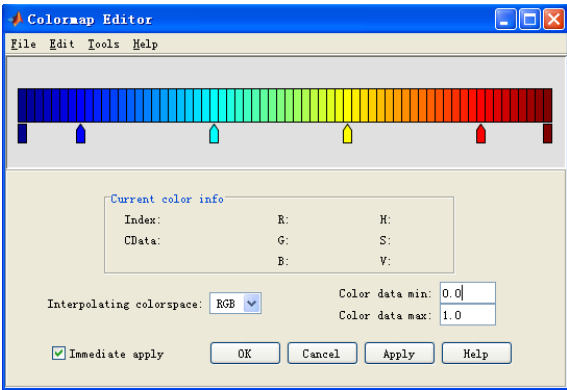


图 4-65 色图编辑对话框

表 4-9 图形窗口中 Insert 菜单命令

命令	功能介绍	命令	功能介绍
X Lable	插入 X 轴	Arrow	插入箭头
Y Lable	插入 Y 轴	Text Arrow	插入文本箭头
Z Lable	插入 Z 轴	Double Arrow	插入双箭头
Title	插入标题	TextBox	插入文本框
Legend	添加图例	Rectangle	插入矩形
Colorbar	添加颜色条	Ellipse	插入椭圆
Line	插入直线	Axes	添加坐标系
Light	亮度控制		

4. Tools 菜单

Tools 菜单包括一些常用的图形工具，如平移、旋转、缩放和视点控制等，并且 Tools 菜单还提供了两个图形分析工具：Basic Fitting 工具和 Data Statistics 工具，用于对图形中的数据进行拟合和分析。

数据拟合对话框和数据分析对话框分别如图 4-66 和图 4-67 所示。

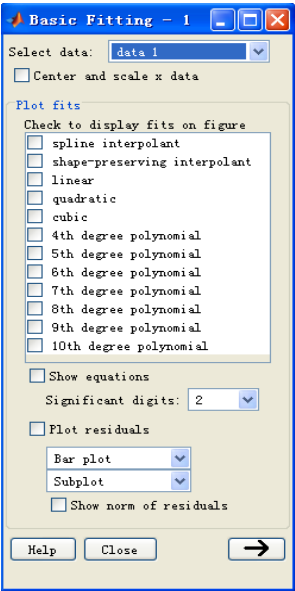


图 4-66 数据拟合对话框

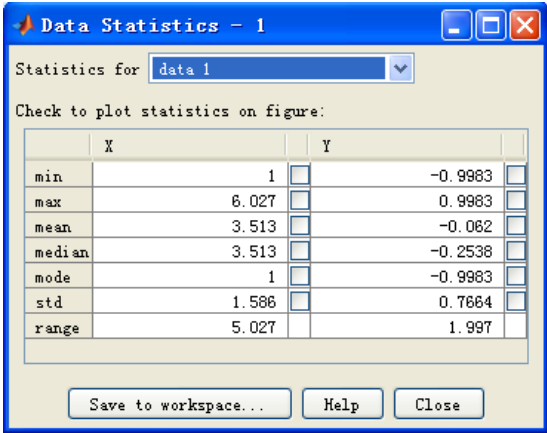


图 4-67 数据分析对话框

【例 4-49】选择相应的拟合方法对图形中的数据进行拟合，如对余弦曲线二阶多项式拟合。
在命令窗口中输入如下语句：

```
figure
X=-5:0.1:5;
Y=sin(X);
plot(X,Y)
```

图形窗口中的输出结果如图 4-68 所示。

在图形窗口中选择“Tools”→“Basic Fitting”→“5th degree polynomial”命令，进行四阶多项式拟合，如图 4-69 所示。

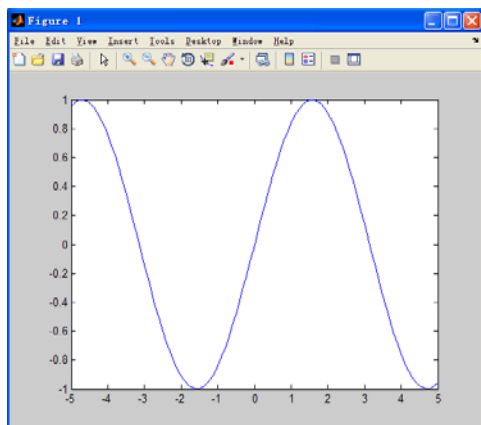


图 4-68 余弦曲线

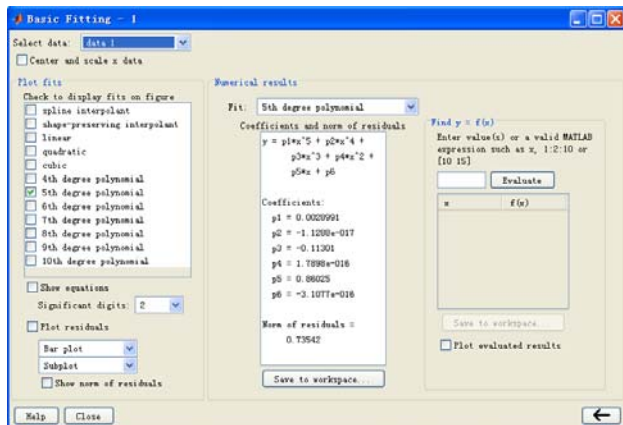


图 4-69 数据拟合对话框

在图形窗口得到的最终图形如图 4-70 所示。

5. Desktop 菜单

Desktop 菜单用于将窗口合并到 MATLAB 主界面的窗口中。

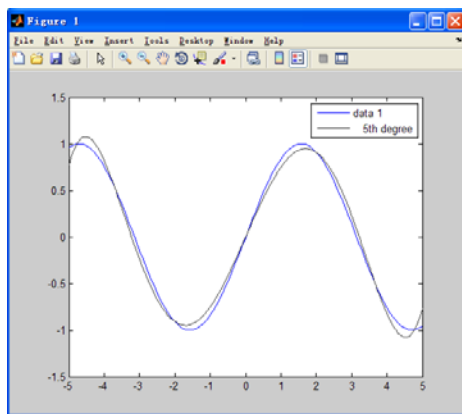


图 4-70 余弦曲线拟合结果

第 5 章 M 文件编程

通过前面几章的介绍，了解到 MATLAB 具有强大的数值运算能力、符号运算能力和丰富的绘图功能。作为科学计算的工具软件，为了实现更加复杂的运算，MATLAB 还可以像其他计算机高级语言一样进行程序设计，它通过 M 语言来实现完整的编写应用程序的能力。

5.1 编程概述

MATLAB 作为一种程序化的编程语言，比一般的高级语言更简单，程序更容易调试，有更好的交互性。

用户可以把所要运行的语句编制成文件，进而由 MATLAB 解释执行，最终返回结果。编制成的文件要求是以 .m 作为文件扩展名的 M 文件，为了方便用户直接调用，MATLAB 系统自带了大量的 M 文件。一个 M 语言文件是由若干 MATLAB 命令组合在一起构成的，M 语言文件是标准的纯文本格式的文件，其文件扩展名为 .m。

MATLAB 语言是由 C 语言开发的，因此 M 文件的语法规则和 C 语言的语法规则类似，对于熟悉 C 语言的用户来说，M 文件的编写会比较容易。

5.1.1 M 文件的创建

如果需要对 M 文件进行编辑、保存和运行，首先需要创建一个 M 文件。M 文件可以在任意的文本编辑器，如写字板、记事本或 Word 中编辑，但系统默认的是 MATLAB 自带的 M 文件编辑器。

创建 M 文件主要有如下三种方法：

- 通过菜单创建。单击“File”→“New”→“Script”菜单命令，如图 5-1 所示，进入如图 5-2 所示的 M 文件编辑器，进而可以编辑、保存和运行 M 文件。

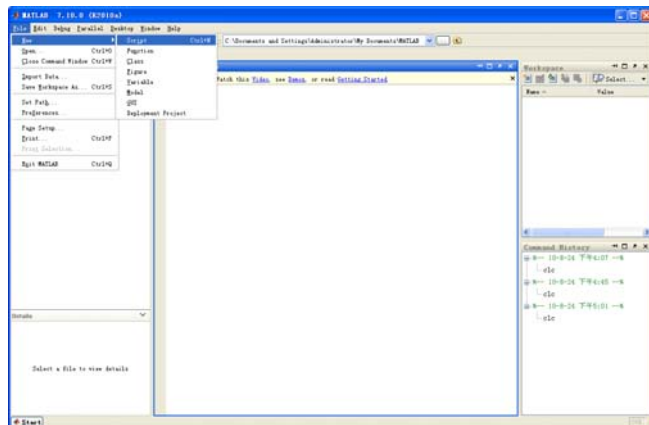



图 5-1 File 菜单

- 通过工具栏创建。单击如图 5-3 所示的工具栏中的图标，同样可以创建如图 5-2 所示的 M 文件。

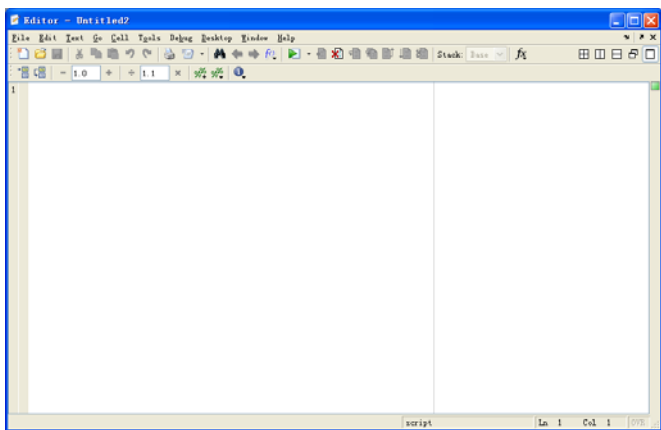


图 5-2 创建 M 文件

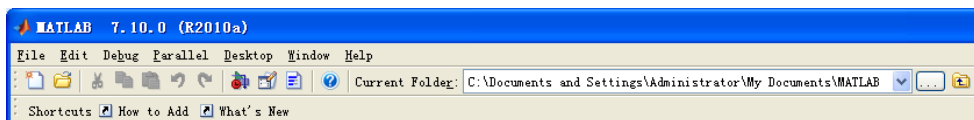


图 5-3 工具栏

- 通过命令创建。在 MATLAB 命令窗口中输入如下语句，同样可以创建如图 5-2 所示的 M 文件。

```
edit
```

5.1.2 M 文件的打开

打开 M 文件主要有如下三种方法：

- 通过菜单打开。单击“File”→“Open”菜单命令，如图 5-4 所示，进入如图 5-5 所示的打开文件对话框，选择需要打开的 M 文件后，进入如图 5-6 所示的 M 文件编辑器，同时显示该 M 文件的内容。

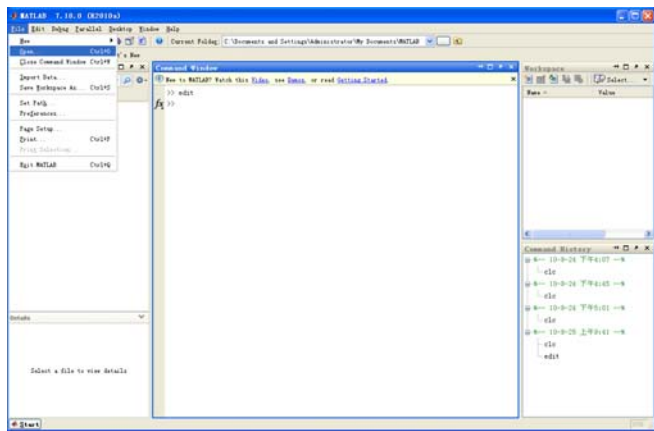


图 5-4 File 菜单

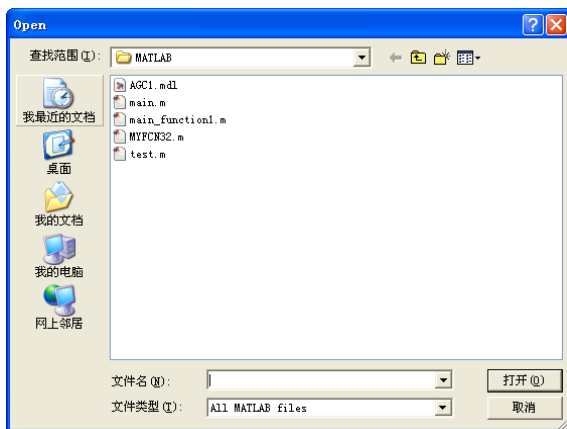


图 5-5 打开文件对话框

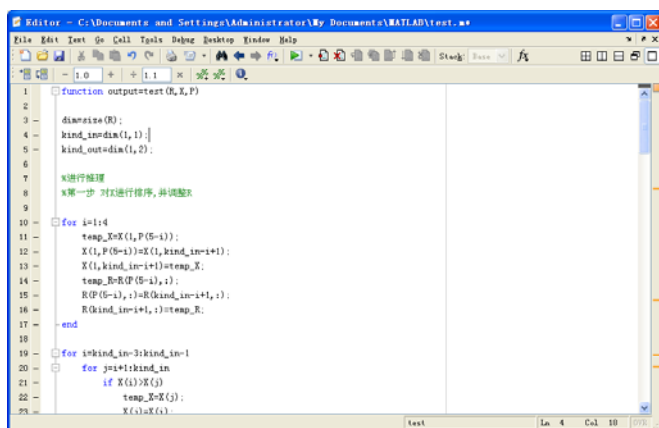



图 5-6 打开 M 文件

- 通过工具栏打开。单击如图 5-3 所示的工具栏中的  图标，进入如图 5-5 所示的打开文件对话框，后面的操作与通过菜单打开相同。
- 通过命令打开。在 MATLAB 命令窗口中输入如下语句，同样可以打开如图 5-6 所示的 M 文件。

```
edit test.m
```

需要说明的是，在运行 M 文件之前需要确定它是否在 MATLAB 的当前目录或搜索路径中。MATLAB 提供了 `which` 命令在当前目录和搜索路径中查找 M 文件。

【例 5-1】使用 `which` 命令查找 `test` 文件。

在命令窗口中输入如下语句：

```
which test
```

命令窗口中的输出结果如下所示：

```
C:\Documents and Settings\Administrator\My Documents\MATLAB\test.m
```

5.1.3 M 文件内容的显示

需要显示 M 文件的内容，可以使用第 1 章讲到的 `type` 命令。

【例 5-2】显示生成魔方矩阵的 `magic` 函数的内容。

在命令窗口中输入如下语句：

type magic

命令窗口中的输出结果如下所示:

```
function M = magic(n)
%MAGIC Magic square.
% MAGIC(N) is an N-by-N matrix constructed from the integers
% 1 through N^2 with equal row, column, and diagonal sums.
% Produces valid magic squares for all N > 0 except N = 2.

% Copyright 1984-2002 The MathWorks, Inc.
% $Revision: 5.15 $ $Date: 2002/04/15 03:44:23 $

% Historically, MATLAB's magic was a built-in function.
% This M-file uses a new algorithm to generate the same matrices.

n = floor(real(double(n(1))));

% Odd order.
if mod(n,2) == 1
    [J,I] = meshgrid(1:n);
    A = mod(I+J-(n+3)/2,n);
    B = mod(I+2*J-2,n);
    M = n*A + B + 1;

% Doubly even order.
elseif mod(n,4) == 0
    [J,I] = meshgrid(1:n);
    K = fix(mod(I,4)/2) == fix(mod(J,4)/2);
    M = reshape(1:n*n,n,n)';
    M(K) = n*n+1 - M(K);

% Singly even order.
else
    p = n/2;
    M = magic(p);
    M = [M M+2*p^2; M+3*p^2 M+p^2];
    if n == 2, return, end
    i = (1:p)';
    k = (n-2)/4;
    j = [1:k (n-k+2):n];
    M([i; i+p],j) = M([i+p; i],j);
```

```
i = k+1;  
j = [1 i];  
M([i; i+p],j) = M([i+p; i],j);  
end
```

5.1.4 M 文件的分类

MATLAB 编写的 M 文件可以分为两大类：一类是脚本文件，另一类是函数文件。

(1) 脚本文件

脚本文件可以包含 MATLAB 的各种语句，它没有输入和输出参数。脚本一般有两个用途：一是作为主程序，二是被其他程序调用。

作为主程序时，因为脚本文件没有独立的工作空间，所以其产生的变量都存放在 MATLAB 的基本工作空间中，除非用户运行 `clear` 命令将它们清除。它们可以与其他脚本、MATLAB 命令窗口、Simulink 等共享。

被其他程序调用时，脚本的作用在于将脚本内容简单插入到调用位置，其变量存放在调用它的程序对应的工作空间中。

本章后面的介绍都是针对作为主程序的情况。

(2) 函数文件

函数文件比脚本文件复杂一些，它一般包含输入和输出参数。没有特别指明的话，函数运行过程中产生的变量都存放在函数本身的工作空间。

关于脚本和函数的详细介绍请参看后面的章节。

5.2 与外部数据的交换

MATLAB 中的文件与其他系统一样，也有两类文件组成：一是文件，又称 M 文件，另一类是数据文件。数据文件包括文本文件和二进制文件，其中 MAT 格式的文件属于二进制文件。

MATLAB 有着强大的数据处理功能，经常需要从外部文件读取数据或将数据写到外部文件。这一节主要介绍数据文件的保存和调用。

5.2.1 数据文件保存

1. MAT 文件的保存

MATLAB 在科学计算和实际的工程应用中，有大量的数据需要处理。读取数据是进行数据处理的基础，处理完毕后，有时需要将处理结果保存起来，以便以后调用。有时，在清除变量或退出 MATLAB 后，变量不复存在。为了保存变量的值，需要把它们存储在数据文件中。

在 MATLAB 中，使用 `save` 命令可以将当前 MATLAB 工作区中的几个或全部变量存入指定的文件中，默认的文件格式是 `.mat`。当选择 ASCII 时按字符格式保存，保存文件的类型由后缀名来确定，还可以通过通配符来保存一类变量。

【例 5-3】利用 `save` 命令对数据文件进行保存。

在命令窗口中输入如下语句：

```
clear;  
clc;
```

```
X=magic(4)
```

```
Y=rand(4)
```

命令窗口中的输出结果如下所示:

X =

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

Y =

```
0.8147    0.6324    0.9575    0.9572
0.9058    0.0975    0.9649    0.4854
0.1270    0.2785    0.1576    0.8003
0.9134    0.5469    0.9706    0.1419
```

将上述 X、Y 矩阵保存起来, 在命令窗口中输入如下语句:

```
save data1 X;
```

```
save data2 Y;
```

上述指令实现了将矩阵 X 和 Y 中的数据分别以 data1.m 和 data2.m 文件名保存的功能。

需要说明的是, 用 save 命令保存的文件可以用 load 命令打开。load 命令会根据文件类型载入文件, 一般不需要声明文件类型。从广义上讲, load 命令可以载入任何字符类型的数据文件, 但文件中必须使用空格符分隔数值, 每行的数据个数必须相同。数据需要存放在与文件名同名的变量中。

2. 文本文件的保存

虽然 MATLAB 自带的 MAT 文件为二进制文件, 但为了便于和外部程序进行交换以及方便地查看文件中的数据, 也常常采用文本数据格式与外界交换数据。

fprintf 函数可用于将数据按指定的格式写入文本文件中并保存, 具体用法如下所示:

count=fprintf(fid,format,输出变量列表): 将输出变量按指定格式写入文件中, 若省略 fid, 则表示在屏幕上进行输出。

- count: 返回所写入数据的元素个数, 可以省略。
- format: 以%开头, 可以使用的特殊字符包括: \b-退后一格, \t-水平制表符, \f-换页, \-反斜杠, \n-换行, \r-回车, '单引号, %%-百分号。
- fprintf: 该命令的格式说明符包括: c-字符型, g-浮点数(自动), d-十进制整数, o-八进制, e-浮点数(科学计数法), s-字符串, f-浮点数(小数形式), x/X-十六进制。

【例 5-4】创建一个字符矩阵存入磁盘, 再将其读出赋值给另一个矩阵。

在命令窗口中输入如下语句:

```
a='hello world'
```

```
fid=fopen('c:\char.txt','w');
```

```
fprintf(fid,'%s',a);
```

```
fclose(fid);
```

```
fid1=fopen('c:\char.txt','rt');
```

```
b=fscanf(fid1,'%s')
```

命令窗口中的输出结果如下所示:

```
b=  
helloworld
```

5.2.2 数据文件调用

在 MATLAB 中进行数据处理或计算,将文件加载到 MATLAB 工作空间中,并赋给指定变量,方法有多种:

- load 命令

运用 MATLAB 调用二进制文件的命令 load,可以调用*.txt、*.dat、*.m 格式文件,格式如下:

```
>>load 文件名 %文件名即为变量名
```

- load 函数

load 函数可以以函数的形式打开文件,并可以指定变量名,格式如下:

```
>>a=load('文件名')
```

- dlmread 函数

dlmread 函数用于读取有界定符号的数据文件,并可以指定变量名,格式如下:

```
>>b=dlmread('文件名')
```

- textread 函数

textread 函数从数据文件中读取格式化的数据或字符,并可以指定变量名,格式如下:

```
>>[a,b,c,...]=textread('文件名',格式)
```

最简单的用法还是:

```
c=textread('文件名')
```

- xlsread 函数

xlsread 函数可以直接读取 Excel 文件,并可以指定变量名,格式如下:

```
>>d=xlsread('C:\matlab2010a\work\ls.xls');
```

- wklread 函数

wklread 函数可以直接读取数据库文件,并可以指定变量名,格式如下:

```
e=wklread('C:\matlab2010a\work\b.wkl')
```

- 如果数据能复制到剪贴板,直接在 MATLAB 命令窗口中粘贴到中括号[]中即可,格式如下:

```
f=[]
```

- 对于 MATLAB 的文本文件的读取函数 fscanf,也不得不提,不过与之配套的函数还有打开文件函数 fopen 和关闭文件函数 fclose,用法如下:

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

其中,A 用以存放读取的数据。COUNT 返回所读取的数据元素个数。fid 为文件句柄。format 用以控制读取的数据格式,由%加上格式符组成,常见的格式符有 g、d、s。如:

%g——表示浮点数值。

%d——表示十进制数值。

%s——表示字符串。

size 为可选项,用于决定矩阵 A 中数据的排列形式。

【例 5-5】利用 MATLAB 的文本文件的读取函数 `fscanf` 读取文件。

具体调用指令如下：

```
fid=fopen('d.txt','r');  
g=fscanf(fid,'%g');  
status=fclose(fid);
```

5.3 流程控制

流程控制是程序编写中最基本的内容，MATLAB 程序的流程控制与 C 程序的流程控制类似。MATLAB 同样提供了顺序结构、分支结构和循环结构等必不可少的流程控制，还包括特有的一些结构，如 `try-catch` 结构。

5.3.1 顺序结构

顺序结构是 MATLAB 最基本的流程控制结构。它是指在系统运行程序时，将按照程序的物理顺序依次执行。顺序结构的程序比较容易编制，但是结构比较单一，可以实现的功能有限。

【例 5-6】绘制以 `a` 为横轴，`b` 为纵轴的图形。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存后执行（按 `F5` 键）：

```
clear  
clc  
%定义变量 a  
a=[15 20 25 30 35];  
%定义变量 b  
b=[155.1 156.3 157.5 158.9 160.4];  
figure(1)  
%使用默认设置作图，以 a 为横轴，以 b 为纵轴  
plot(a,b)  
hold on  
%用红色标志'x'画出相关的点  
plot(a,b,'x');
```

图形窗口中的输出结果如图 5-7 所示。

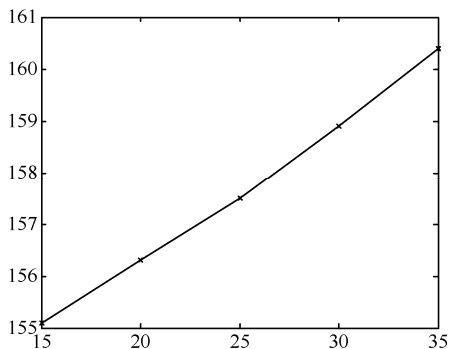


图 5-7 `plot` 函数绘制的二维图形

5.3.2 分支结构

在复杂的计算中，经常会遇到根据条件选择执行语句的情况，这时就需要用到分支结构。MATLAB 提供了 if 语句和 switch 语句来实现分支结构。

1. if 语句

if 语句通过检验逻辑表达式的真假，判断是否运行后面的语句组。执行 if 语句可分为以下两个步骤：

(1) 计算逻辑表达式的结果。如果值为 1，说明逻辑表达式为真；如果值为 0，说明逻辑表达式为假；当逻辑表达式使用矩阵时，要求矩阵元素必须都不为 0，逻辑表达式才为真。

(2) 根据逻辑表达式的结果，选择运行的语句组。

if 语句的语法结构如下所示：

```
if 逻辑表达式 1
    语句组 1
elseif 逻辑表达式 2
    语句组 2
else
    语句组 3
end
```

以上是 if 语句的一般结构。

```
if 逻辑表达式
    语句组
end
```

以上是 if 语句的简化结构，也是最简单的形式，其中只有一个判断语句和一个语句组。

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

以上的 if 语句包含一个判断语句和两个语句组。

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

以上的 if 语句包含多个判断语句和多个语句组。

【例 5-7】使用 if 语句计算 $x=5$ 时的表达式 $y(x) = \begin{cases} 2x+1 & x < 0 \\ 3x-2 & 0 \leq x \leq 4 \\ 4x+3 & x > 4 \end{cases}$ 的值。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存后执行：

```
clear
clc
x=5;
if x<0
    y=2*x+1;
elseif x>=0 & x<=4
    y=3*x-2;
else
    y=4*x+3;
end
y
```

命令窗口中的输出结果如下所示：

```
y =
    23
```

它等价于下述语句：

```
clear
clc
x=5;
if x<0
    y=2*x+1;
end
if x>=0 & x<=4
    y=3*x-2;
end
if x>4
    y=4*x+3;
end
y
```

2. switch 语句

switch 语句是将某表达式的值依次和提供的检测值范围比较，如果比较结果都不同，则取下一个检测值范围进行比较；如果比较结果包含相同的检测值，则执行相应的语句组，然后跳出结构。

switch 语句的语法结构如下所示：

```
switch expression
case value1
    语句组 1;
case value2
```



```
语句组 2;  
...  
case valuen  
语句组 n;  
otherwise  
语句组;  
end
```

其中，otherwise 表示除 value1~valuen 外未列出的所有情况，它可以省略。

需要说明的是，switch 指令后的 expression 是一个标量或者字符串，当 expression 是一个标量时，需要判断 expression==value 是否成立；当 expression 是一个字符串时，需要调用函数 strcmp 实现比较。case 指令后面的 value 可以是标量、字符串或细胞数组，当 value 是一个细胞数组时，系统将 expression 的值和此细胞数组中的所有元素比较，如果存在 expression 的值和细胞数组中的某个元素相等，则执行相应 case 指令后面的语句组。

【例 5-8】使用 switch 语句判断键盘输入值并给出相应提示。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear  
clc  
num=input('请输入一个数')  
switch num  
    case -1  
        disp('I am a teacher.');    case 0  
        disp('I am a student.');    case 1  
        disp('You are a teacher.');    otherwise  
        disp('You are a student.');end
```

命令窗口中的输出结果如下所示：

请输入一个数

当键入 5 并按回车后，命令窗口中的输出结果如下所示：

```
num =  
    5  
  
You are a student.
```

5.3.3 循环结构

循环结构是按照给定的条件，反复执行指定语句组进行重复运算。循环结构对于需要重复运算的场合十分有效，在很大程度上缩减了程序的代码量。

MATLAB 提供了两种循环结构：一种是 for 循环，另一种是 while 循环。

1. for 循环

for 循环的语法结构如下所示：

```
for 循环变量=表达式 1: 表达式 2: 表达式 3
循环体;
end
```

需要说明的是，循环体的执行次数是由表达式的值决定的，表达式 1 的值是循环变量的起点，表达式 2 的值是循环变量的步长（步长的默认值是 1），表达式 3 的值是循环变量按步长方向增加不允许超过的界限。另外，循环结构可以嵌套使用。

【例 5-9】使用单重 for 循环将一系列数的头尾次序颠倒。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行。

```
clear
clc
n=8;
x=rand(1,n)
for i=1:n
    y(i)=x(8+1-i);
end
y
```

命令窗口中的输出结果如下所示：

```
x =
    0.8147    0.9058    0.1270    0.9134    0.6324    0.0975    0.2785    0.5469
y =
    0.5469    0.2785    0.0975    0.6324    0.9134    0.1270    0.9058    0.8147
```

【例 5-10】使用双重 for 循环将一系列数从小到大排序。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear
clc
n=8;
x=rand(1,n)
for i=1:n-1
    for j=i+1:n
        if x(i)>x(j)
            tmpx=x(j);
            x(j)=x(i);
            x(i)=tmpx;
        end
    end
end
x
```

命令窗口中的输出结果如下所示：

```
x =  
    0.9575    0.9649    0.1576    0.9706    0.9572    0.4854    0.8003    0.1419  
  
x =  
    0.1419    0.1576    0.4854    0.8003    0.9572    0.9575    0.9649    0.9706
```

2. while 循环

while 循环的语法结构如下所示：

```
while 表达式  
    循环体;  
end
```

需要说明的是，while 循环执行时首先判断表达式的值，如果表达式的值不为 0，则执行循环体，执行完毕以后继续判断表达式的值；如果表达式的值为 0，循环将结束。通常来说，循环体中包含导致表达式值变化的语句。

表达式的值可以是标量和数组，若表达式的值是数组，则只有当数组中所有元素不为 0 时才可以执行循环体。

【例 5-11】使用 while 循环计算阶乘。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear  
clc  
n=10;  
result=1;  
while n>=1  
    result=result*n;  
    n=n-1;  
end  
result
```

命令窗口中的输出结果如下所示：

```
result =  
    3628800
```

5.3.4 其他流程控制结构

前三个小节主要介绍了常用的流程控制结构，但是在程序中还会经常遇到一些特殊情况，如提前终止循环、显示错误信息等。MATLAB 提供了一些特殊的控制结构来实现这些功能，下面将分别进行介绍。

1. continue 命令

continue 命令的作用是在循环控制中结束本次循环，即跳过本次循环中尚未执行的语句，进入下一次循环。

【例 5-12】使用 continue 命令读取数据，遇到大于 8 的数显示其位置（-1 表示全不大于 8）。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear
```

```
clc
num=20;
a=10*rand(1,num)
pos=-1;
n=0;
while n<num
    n=n+1;
    if a(n)<=8
        continue;
    end
    pos=n
end
```

命令窗口中的输出结果如下所示：

```
a =
    Columns 1 through 8
    8.1428    2.4352    9.2926    3.4998    1.9660    2.5108    6.1604    4.7329

    Columns 9 through 16
    3.5166    8.3083    5.8526    5.4972    9.1719    2.8584    7.5720    7.5373

    Columns 17 through 20
    3.8045    5.6782    0.7585    0.5395

pos =
    1

pos =
    3

pos =
   10

pos =
   13
```

2. break 命令

break 命令和 continue 命令的作用都是中断循环，但 break 命令和 continue 命令不同的是，break 命令是跳出当前循环，不再执行此循环。

【例 5-13】使用 break 命令读取矩阵数据，遇到大于 4 的数退出并显示退出时的位置（-1 表示全不大于 4）。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear
clc
a=10*rand(6)
size_a=size(a);
for i=1:size_a(1)
    pos(i)=-1;
    for j=1:size_a(2)
        if a(i,j)>4
            pos(i)=j;
            break;
        end
    end
end
pos
```

命令窗口中的输出结果如下所示：

```
a =
    5.3080    0.1190    1.6565    4.5054    5.3834    0.0463
    7.7917    3.3712    6.0198    0.8382    9.9613    7.7491
    9.3401    1.6218    2.6297    2.2898    0.7818    8.1730
    1.2991    7.9428    6.5408    9.1334    4.4268    8.6869
    5.6882    3.1122    6.8921    1.5238    1.0665    0.8444
    4.6939    5.2853    7.4815    8.2582    9.6190    3.9978

pos =
     1     1     1     2     1     1
```

3. return 命令

return 命令可以使正在执行的函数正常退出，并返回调用它的函数继续运行，同时可以用于正常和强制结束函数的运行。

【例 5-14】使用 **return** 命令读取矩阵数据（-1 表示全不大于 5）。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear
clc
a=6*rand(7,6)
size_a=size(a);
for i=1:size_a(1)
    pos(i)=-1;
    for j=1:size_a(2)
        if a(i,j)>5
            pos(i)=j;
            return;
        end
    end
end
```

```

        end
    end
end
pos

```

命令窗口中的输出结果如下所示：

```

a =
    2.2109    2.6152    3.8659    3.3009    5.0659    5.5403
    3.7537    2.6807    2.2717    3.7349    1.1686    2.5812
    4.6814    1.8381    4.8695    3.5223    1.3555    1.1089
    0.4868    3.0511    3.1970    1.2465    1.0242    5.4293
    5.5763    3.0646    2.1044    1.8075    1.3660    5.8785
    4.6543    4.9058    5.6340    2.8255    2.6142    2.6332
    2.9207    4.7690    5.2557    1.3829    1.8666    0.6667

pos =
    5

```

4. input 函数

input 函数的作用是用户可以通过键盘输入数据、字符串和表达式，并且通过按回车键将输入的内容送到工作空间。input 函数的语法结构如下所示：

- **R=input('message'):** 将用户输入的内容如数值、字符串和细胞数组等赋给变量 R。
- **R=input('message','s'):** 将用户输入的内容如数值、字符串和细胞数组等作为字符串赋给变量 R。

【例 5-15】使用 input 函数进行猜字谜小游戏设计。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```

clear
clc
disp('game is starting now! ');
x=fix(10*rand);

for n=1:5      % 用户只有 5 次机会
    a=input('please enter the number you guess: ');
    if a>x
        disp('the number is higher! ');
    elseif a<x
        disp('the number is lower! ');
    else
        disp('congratulations to you! ');
        return
    end
end
end

```

```
disp('haha, try again???');
```

命令窗口中的输出结果如下所示:

```
game is starting now!
```

```
please enter the number you guess:
```

输入 10 并按回车键后, 命令窗口中的输出结果如下所示:

```
please enter the number you guess: 10
```

```
the number is higher!
```

输入 2 并按回车键后, 命令窗口中的输出结果如下所示:

```
please enter the number you guess: 2
```

```
the number is lower!
```

输入 6 并按回车键后, 命令窗口中的输出结果如下所示:

```
please enter the number you guess: 6
```

```
congratulations to you!
```

```
haha, try again???
```

这里需要说明的是, 如果输入一直不满足退出条件, 则程序将无法正常工作结束, 此时可以通过按快捷键 Ctrl+C 终止程序。

5. keyboard 命令

keyboard 命令的作用是停止程序的执行, 并将控制权交给键盘, 用户可以通过键盘输入各种合法的指令, 并且可以修改程序中的指令。执行 return 命令后, 控制权将再次交给程序。

【例 5-16】使用 keyboard 命令获得键盘控制权。

在 M 文件编辑器中新建包含如下内容的 M 文件, 保存并执行:

```
clear
clc
k=3;
l=k^3
keyboard
m=l^2
```

命令窗口中的输出结果如下所示:

```
l =
```

```
27
```

```
K>>
```

接着在命令窗口中输入如下语句:

```
[k l]
```

命令窗口中的输出结果如下所示:

```
ans =
```

```
3    27
```

```
K>>
```

继续在命令窗口中输入如下语句:

```
l=4;  
return;
```

命令窗口中的输出结果如下所示:

```
m =  
    16
```

本例表明执行 `keyboard` 语句后, 可以查询和修改变量值。

6. error 函数

`error` 函数的作用是显示出错信息, 并终止当前运行的程序。`Error` 函数的语法结构如下所示:

```
error('message')
```

【例 5-17】使用 `error` 函数显示出错信息。

在 M 文件编辑器中新建包含如下内容的 M 文件, 保存并执行:

```
clear  
clc  
a=inf;  
if isinf(a)  
    error('a is an infinity number');  
    disp('display again');  
end
```

命令窗口中的输出结果如下所示:

```
??? a is an infinity number
```

本例表明执行 `error` 语句后, 程序将终止执行。

7. warning 函数

`warning` 函数的作用是显示警告信息, 但是会继续运行当前程序, `warning` 函数的语法结构如下所示:

- `warning('message')`

【例 5-18】使用 `warning` 函数显示警告信息。

在 M 文件编辑器中新建包含如下内容的 M 文件, 保存并执行:

```
clear  
clc  
a=inf;  
if isinf(a)  
    warning('a is an infinity number');  
    disp('display again');  
end
```

命令窗口中的输出结果如下所示:

```
Warning: a is an infinity number  
display again
```

本例表明执行 `warning` 语句后, 程序仍将继续执行。

8. echo 命令

通常在运行 M 文件时, 执行的语句是不显示在命令窗口中的。MATLAB 提供了 `echo` 命令

用来选择显示或不显示执行的语句，常用于调试和演示程序。对于脚本文件和函数文件，echo 命令的语法结构稍有不同。

对于脚本文件，echo 命令的语法结构如下所示：

- echo on: 显示其后所有执行的语句。
- echo off: 不显示其后所有执行的语句。
- echo: 在上述两种情况之间切换。

对于函数文件，echo 命令的语法结构如下所示：

- echo file on: 显示文件名为 file 的 M 文件的执行语句。
- echo file off: 不显示文件名为 file 的 M 文件的执行语句。
- echo file: 在上述两种情况之间切换。

对于 echo 命令，脚本文件和函数文件中都适用的语法结构如下所示：

- echo on all: 显示其后所有 M 文件的执行语句。
- echo off all: 不显示其后所有 M 文件的执行语句。

【例 5-19】 使用 echo 命令显示执行语句。

在 M 文件编辑器中新建包含如下内容的 M 文件，保存并执行：

```
clear
clc
echo on
x1=rand(3);
y1=sin(x1);
echo off
x2=rand(3)
y2=cos(x1)
```

命令窗口中的输出结果如下所示：

```
x1=rand(3);
y1=sin(x1);
echo off

x2 =
    0.0344    0.7655    0.4898
    0.4387    0.7952    0.4456
    0.3816    0.1869    0.6463

y2 =
    0.7609    0.9989    0.7682
    0.9995    0.9953    0.9501
    0.9619    0.6797    0.5815
```

9. pause 命令

pause 命令的作用是暂时中止运行程序，等待用户按任意键后继续进行。pause 命令经常用在程序调试的过程中和用户需要查询中间结果时，它的语法结构如下所示：

- `pause`: 暂时停止执行的程序, 等待用户按任意键继续。
- `pause(n)`: 使程序在暂时停止 n 秒后继续执行, n 为非负实数。
- `pause on`: 允许连续的 `pause` 命令暂时停止程序的执行。
- `pause off`: 使连续的 `pause` 命令变为无效。

【例 5-20】使用 `pause` 命令暂停程序运行。

在 M 文件编辑器中新建包含如下内容的 M 文件, 保存并执行:

```
clear
clc
tic %计时开始
for i=1:10
    pause(3) %暂停 3 秒
end
toc %计时结束, 并输出自 tic 语句至本语句间的执行时间
```

命令窗口中的输出结果如下所示:

```
Elapsed time is 30.010953 seconds.
```

这里需要说明的是, 采用 `tic` 和 `toc` 命令组合来计算执行时间是不准确的, 每次的结果都可能不同 (与系统内存使用率等情况有关)。

10. try-catch 结构

`try-catch` 结构的作用是检测程序是否出现异常, 并指定异常的处理方法, 它的语法结构如下所示:

```
try
语句体 1
catch
语句体 2
end
```

需要说明的是:

- (1) 语句体 1 总会被执行, 当执行出现异常时, 才会执行语句体 2。
- (2) 可以调用 `lasterr` 函数查询异常原因, 如果 `lasterr` 函数的运行结果为空, 说明语句体 1 被成功执行, 即没有出现异常。

【例 5-21】使用 `try-catch` 结构处理异常。

在 M 文件编辑器中新建包含如下内容的 M 文件, 保存并执行:

```
clear
clc
a=rand(3,4);
try
    b=a(5,4) %维数超界
catch
    disp('请检查 a 的值');
end
lasterr
```

命令窗口中的输出结果如下所示：

请检查 a 的值

ans =

Index exceeds matrix dimensions.

5.4 脚本文件

脚本文件是按用户的需求排列成的 MATLAB 语句组，5.3 节中所建立的 M 文件都是脚本文件。

脚本文件是扩展名为 .m 的文件，它可以包含 MATLAB 的各种语句。在运行的过程中，脚本文件所产生的变量都将保存在基本工作空间中，在不关闭 MATLAB 且不使用 clear 函数的情况下，变量会一直保存在基本工作空间中。需要说明的是，在 MATLAB 启动的时候，基本工作空间自动产生，在 MATLAB 关闭的时候，基本工作空间才被删除。

对于简单的计算，脚本文件的作用并不十分明显，但是随着计算复杂度的增加，所需要的语句越来越多，从命令窗口中直接输入语句会很烦琐，此时使用脚本文件就比较方便。

1. 脚本文件的基本结构

脚本文件的基本结构如下所示：

% Compute a factorial value	帮助文本
% FACT(N) returns the factorial of N,	
% usually deoted by N!	
% Put simple,FACT(N) is PROD(1:N).	注释
f=prod(1:n);	脚本内容

脚本文件具体说明如下：

- 帮助文本：当使用 help 命令查询该脚本文件的说明信息时，在窗口中显示此内容。
- 注释：在程序中解释程序功能的文字。
- 脚本内容：运行产生输出的语句。


2. 脚本文件的创建

创建脚本文件分为以下几个步骤：

- (1) 新建一个 M 文件。
- (2) 在 M 文件编辑器中输入程序。
- (3) 保存为扩展名.m 的文件。

3. 脚本文件的运行

运行脚本文件可分为两个步骤：

- (1) 将脚本所在的目录设置为当前目录或搜索路径。
- (2) 运行脚本文件，有以下五种方式：
 - 在 M 文件编辑器中打开脚本文件，单击“Debug”→“Run”菜单命令运行脚本文件。
 - 在 M 文件编辑器中打开脚本文件，单击工具栏中的图标，运行脚本文件。
 - 在 M 文件编辑器中打开脚本文件，按快捷键 F5，运行脚本文件。
 - 在命令窗口中直接输入脚本文件名，运行脚本文件。
 - 在 M 文件编辑器中打开脚本文件，复制脚本内容，在命令窗口中粘贴脚本内容后按回

车键，运行脚本文件。

【例 5-22】使用脚本近似计算 $\int_a^b \int_c^d \cos(xy)e^{xy^2} dx dy$ ，其中 $a=0$ ， $b=1$ ， $c=1$ ， $d=2$ 。

脚本中的内容如下所示，保存并执行：

```
clear;
clc;
xmin=1;    %指定积分下限，x 为内积分变量
xmax=2;    %指定积分上限
xnum=1001;%指定包含两顶点的采样个数
ymin=0;
ymax=1;
ynum=501;
xstep=(xmax-xmin)/(xnum-1);%相邻采样点的距离
x=xmin:xstep:xmax-xstep;%采样点间等距离
ystep=(ymax-ymin)/(ynum-1);
y=ymin:ystep:ymax-ystep;

result=0;%保存计算结果的变量，需累加
%外循环，y 变化
for i=1:ynum-1
    ytmp=y(i);

    resulttmp=0;%保存临时计算结果的变量，需累加

    %内循环，x 变化
    for j=1:xnum-1
        xtmp=x(j);
        resulttmp=resulttmp+cos(xtmp*ytmp)*exp(xtmp*ytmp^2)*xstep;%计算被积函数值并累加
    end

    result=result+resulttmp*ystep;
end
result
```

命令窗口中的输出结果如下所示：

```
result =
    0.9055
```

将第 2~8 行的语句做如下更改后执行：

```
xmin=-1;
xmax=1;
xnum=2001;
```

```
ymin=1;
ymax=2;
ynum=5001;
```

命令窗口中的输出结果如下所示：

```
result =
    1.9228
```

这里需要说明的是，改变第 2~8 行的赋值语句，将得到 $a=1$, $b=2$, $c=-1$, $d=1$ 的近似结果。

【例 5-23】使用脚本计算 $2s^3 + 3s^2 + s + k = 0$ 在 $k \in [0, 2]$ 时的最大实根。

脚本中的内容如下所示，保存并执行：

```
clear
close all
clc
k=0:0.1:2; %k 的采样点
for i=1:length(k)
    kroots=roots([2 3 1 k(i)]); %计算 3 个根，必有一实根
    maxroot(i)=-inf; %首先设置负无穷为最小值，以后将被修改
    for j=1:3
        if isreal(kroots(j)) %判断是否为实根
            maxroot(i)=max(maxroot(i),kroots(j)); %取两者最大值
        end
    end
end
end
figure %图形显示
plot(k,maxroot)
axis([0 2 min(maxroot) max(maxroot)])
xlabel('K')
ylabel('maxroot')
grid on
```

图形窗口中的输出结果如图 5-8 所示。

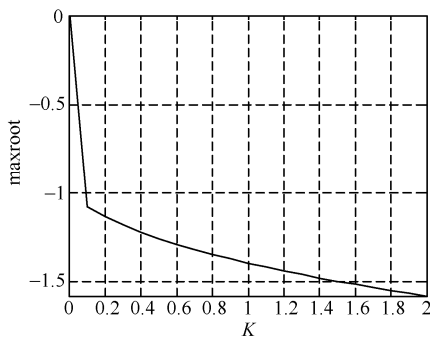


图 5-8 某连续系统的单位响应

5.5 函数文件

函数文件是通过获取外部参数进行运算的，并向外部返回运算的结果，从外部只能看到传入函数文件的输入量和传出的运算结果。

函数文件的扩展名为.m，函数文件中的变量一般是局部变量，即只存在于函数空间内部，不能被外部调用。当函数文件运行时，MATLAB 会为此函数开辟一个临时工作空间，即为函数本身的工作空间，并将所有的中间变量存放到了该工作空间中；当运行到最后一条语句或遇到 return 语句时，立即将该工作空间和所有的中间变量清除。

需要说明的是，函数名和保存的文件名必须相同。

5.5.1 基本结构

一个函数文件通常包含函数定义行、帮助文本、注释和函数体。

function f=fact(n)	函数定义行
% Compute a factorial value	帮助文本
% FACT(N) returns the factorial of N,	
% usually deoted by N!	
% Put simple,FACT(N) is PROD(1:N).	注释
f=prod(1:n);	函数体

函数文件的具体说明如下：

- 函数定义行：定义函数名、输入/输出参数的数量和顺序。
- 帮助文本：当使用 help 命令查询函数的说明信息时，在窗口中显示此内容。
- 注释：在程序中解释程序功能的文字。
- 函数体：运行产生输出的语句。

【例 5-24】比较两个数的大小的函数文件如下所示：

```
function nummax(a,b)
    a=input('请输入 a 的值');
    b=input('请输入 b 的值');
    if(a<=0|b<=0)
        disp('input error');
    elseif a>b
        fprintf('the large number is %f,a);
    elseif a<b
        fprintf('the large number is %f,b);
    else
        fprintf('%f=%f,a,b);
    end
```

命令窗口中的输出结果如下所示：

请输入 a 的值 5

请输入 b 的值 6

the large number is 6.000000

5.5.2 输入/输出参数

在 MATLAB 中编写函数，只需要确定形式参数的列表，不用具体确定实际的参数。只有当函数实际被调用时，才会将实际的数值赋给形式参数指定的变量，变量都存储在变量空间中。需要说明的是，变量空间和工作空间是相互独立的。

MATLAB 函数可以有多个输入和多个输出参数，当然也允许存在无输入、无输出的情况。在无输入/输出参数的时候，可以使用脚本运行的前 4 种方式运行，但是函数文件和脚本文件仍是不同的，主要是变量所属工作空间不同。

在函数被调用时，输入参数需要和函数定义对应给出，输出参数可以按照参数列表指定，也可以不指定。

【例 5-25】计算 $4s^3 + 3s^2 + 2s + k = 0$ 的最大实根和实根个数，其中 k 为输入参数，保存后的函数文件如下所示：

```
function [maxrealroot, realrootsnumber]=myroot(k)
%The function can gain the maximum one and the number
%of all real roots for polynomial
%f(s)=s^3+s^2+k, where k is a real number

P=[4 3 2 k];
kroots=roots(P);
maxrealroot=-inf;
for j=1:3
    if isreal(kroots(j))
        maxrealroot=max(maxrealroot,kroots(j));
        realrootsnumber=j;
    end
end
```

在命令窗口中输入如下语句：

```
kroot=roots([4 3 2 3])
myroot(3)
y=myroot(3)
[s,t]= myroot(3)
```

命令窗口中的输出结果如下所示：

```
kroot =
    -1.0000
    0.1250 + 0.8570i
    0.1250 - 0.8570i

ans =
```

```

-1.0000

y =
-1.0000

s =
-1.0000

t =
1

```

本例给出了包含多个输出参数的函数的编写方法。这里需要说明的是，当多个输出参数只返回一个的情况下，将返回第一个输出参数；每个输出参数都可以是任意的数组元素，如下例所示。

【例 5-26】计算 $4s^3 + 3s^2 + 2s + k = 0$ 的所有根和实根个数，其中 k 为输入参数，保存后的函数文件如下所示：

```

function [kroots realrootsnumber]=myroot1(k)

P=[4 3 2 k];
kroots=roots(P);
realrootsnumber=0;
for j=1:3
    if isreal(kroots(j))
        realrootsnumber= realrootsnumber+1;
    end
end

```

然后在命令窗口中输入如下语句：

```
[s,t]= myroot1(3)
```

命令窗口中的输出结果如下所示：

```

s =
-1.0000
0.1250 + 0.8570i
0.1250 - 0.8570i

t =
1

```

MATLAB 提供了 `nargin` 函数和 `nargout` 函数，主要用于读取函数参数传递时的输入变量个数和输出变量个数，它们的具体用法如下：

- `nargin`：返回函数的输入参数个数。
- `nargin(fun)`：返回 `fun` 函数的输入参数个数。
- `nargout`：返回函数的输出参数个数。
- `nargout(fun)`：返回 `fun` 函数的输出参数个数。

【例 5-27】 使用 nargin 函数和 nargout 函数灵活编写自用函数，保存后的函数文件如下所示：

```
function c=testarg(a,b)
%检测输入/输出参数个数
% 该函数根据不同的输入/输出参数个数进行相应的操作
if (nargout~=1)
disp('使用该函数必须指定一个输出参数! ');
return
end
switch nargin
case 0
disp('使用该函数必须指定一个输入参数! ');
c=[];
return
case 1
c=a.^2;
case 2
c=a+b;
end
```

然后在命令窗口中输入语句及显示结果如下：

```
clear
clc
使用该函数必须指定一个输出参数!
>> A=[1 2 3];B=[2 3 5];
>> testarg(A,B)
使用该函数必须指定一个输出参数!
>> C=testarg
使用该函数必须指定一个输入参数!

C =

     []

>> C=testarg(A)

C =

     1     4     9

>> D=testarg(A,B)

D =
```

3 5 8

>> E=testarg(A,B,C)

??? Error using ==> testarg

5.5.3 子函数

一个 M 文件可以包括多个函数，其中第一个出现的函数称为主函数，其他的函数称为子函数。M 文件的名称一般与主函数的名称保持一致。

主函数的范围大于子函数，主函数可以在 M 文件的外部调用，子函数没有在线帮助，且子函数只能被主函数和该 M 文件的其他子函数调用。

对于子函数需要说明以下几点：

(1) 在一个 M 文件中，主函数必须出现在最上方，它的后面可以包括任意多个子函数，子函数的第一行是函数定义行，子函数的排列顺序可以任意改变。

(2) 系统按照先判断是否为子函数，再判断是否为私有函数，最后判断是否为内置函数的顺序在函数中调用函数（后面两类函数将在后面介绍）。

(3) 在同一个 M 文件中的主函数和子函数的工作空间是彼此独立的，函数间传递信息可以通过输入/输出变量、全局变量或跨空间指令来完成。

【例 5-28】使用子函数近似计算 $\int_a^b \int_c^d \cos(xy) e^{xy^2} dx dy$ ，保存后的函数文件如下所示：

```
function output=mymethod1(X,Y)
xmin=X(1);    %指定积分下限，x 为内积分变量
xmax=X(2);    %指定积分上限
xnum=X(3);    %指定包含两顶点的采样个数
ymin=Y(1);
ymax=Y(2);
ynum=Y(3);
xstep=(xmax-xmin)/(xnum-1); %相邻采样点的距离
x=xmin:xstep:xmax-xstep;    %采样点间等距离
ystep=(ymax-ymin)/(ynum-1);
y=ymin:ystep:ymax-ystep;

result=0; %保存计算结果的变量，需累加
%外循环，y 变化
for i=1:ynum-1
    ytmp=y(i);

    resulttmp=0; %保存临时计算结果的变量，需累加

    %内循环，x 变化
    for j=1:xnum-1
        xtmp=x(j);
```

```

        resulttmp=resulttmp+mymethodsubfun(xtmp,ytmp)*xstep; %计算被积函数值并累加
    end

    result=result+resulttmp*ystep;
end
output=result;

%子函数
function output=mymethodsubfun(x,y)
output=cos(x*y)*exp(x*y^2);

```

在命令窗口中输入如下语句：

```

clear
clc
X=[0 1 1001];
Y=[1 2 501];
Z=mymethod1(X,Y)

```

命令窗口中的输出结果如下所示：

```

Z =
    1.5794

```

【例 5-29】子函数应用举例，保存后的函数文件如下所示：

```

function[avg,med]=newstats(u) % 主函数
%计算均值和中间值
n=length(u);
avg=mean(u,n); % 调用子函数
med=median(u,n); % 调用子函数
function a=mean(v,n) % 子函数
% 计算平均值
a=sum(v)/n;
function m=median(v,n) % 子函数
% 计算中间值
w=sort(v);
if rem(n,2)==1
m=w((n+1)/2);
else

```

在命令窗口中输入如下语句：

```

clear
clc
x=1:11;
[mean,mid]=newstats(x)

```

命令窗口中的输出结果如下所示：

```

mean =

```

```
6  
  
mid =  
6
```

若在命令窗口中输入如下语句：

```
clear  
clc  
x=1:10;  
[a,b]=newstats(x)
```

命令窗口中的输出结果如下所示：

```
a =  
5.5000  
  
b =  
5.5000
```

5.5.4 私有函数

私有函数是指位于子目录“private”下的函数，私有函数的定义与普通函数完全相同，只是它仅能被“private”父目录中的函数调用，其他目录中的函数或“private”父目录中的脚本都不能调用它。

【例 5-30】说明私有函数的使用方法，具体步骤如下所示：

首先设置新目录 C:\test，并在该目录下设置子目录 private，然后将 C:\test 及其子目录加入搜索路径。在加入过程中会发现子目录 private 没有被加入，当手工加入该子目录时，会得到如图 5-9 所示的对话框，提示无法加入。

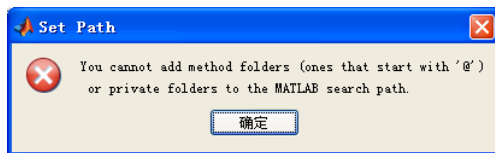


图 5-9 无法加入搜索路径的提示对话框

其次在目录 C:\test\private 中建立内容如下的脚本文件，保存为 test1.m 后在 M 文件编辑器中执行：

```
clear  
clc  
a=3;  
b=5;  
c=myadd(a,b)
```

命令窗口中的输出结果如下所示：

```
??? Undefined function or method 'myadd' for input arguments of type  
'double'.  
  
Error in ==> test1 at 5  
c=myadd(a,b)
```

修改当前目录为 C:\test\private 后执行 test1.m，命令窗口中的输出结果如下所示：

```
c =  
11
```

再次修改当前目录为 C:\Documents and Settings\Administrator\My Documents\MATLAB，在目录 C:\test 中建立内容如下的函数文件：

```
function output=testprivatefun1(x,y)  
if x<y  
    output=myadd(x,y);  
else  
    output=myadd(y,x);  
end
```

在命令窗口中输入如下语句：

```
clear  
clc  
X=9;  
Y=8;  
Z= testprivatefun 1(X,Y)
```

命令窗口中的输出结果如下所示：

```
Z =  
25
```

最后在当前目录下建立内容如下的函数文件：

```
function output=testprivatefun2(x,y)  
if x<y  
    output=myadd(x,y);  
else  
    output=myadd(y,x);  
end
```

在命令窗口中输入如下语句：

```
clear  
clc  
X=9;  
Y=8;  
Z= testprivatefun2 (X,Y)
```

命令窗口中的输出结果如下所示：

```
??? Undefined function or method 'myadd' for input arguments of type  
'double'.  
  
Error in ==> testprivatefun2 at 5  
    output=myadd(y,x);
```

由本例结果可以看出，调用私有函数的脚本文件与私有函数在一个目录下；调用私有函数的函数文件在私有函数的父目录下。

5.5.5 嵌套函数

在 MATLAB 中允许在函数内部定义其他函数，这些内部定义的函数称为嵌套函数。嵌套函数的定义形式和其他函数类似，所不同的是嵌套函数必须用 `end` 结尾。

【例 5-31】说明嵌套函数的使用方法，具体步骤如下所示：

建立内容如下的函数文件，保存后在 M 文件编辑器中或命令窗口中执行：

```
function testnestedfun
clc
x = 5;
y = nestedfun;
    function z = nestedfun
        z = 2*x;
    end
y
z
end
```

命令窗口中的输出结果如下所示：

```
y =
    10

??? Undefined function or variable 'z'.
Error in ==> testnestedfun at 9
z
```

由本例结果可以看出，`testnestedfun` 主函数中变量 `x` 的值传入 `nestedfun` 嵌套函数，而 `nestedfun` 嵌套函数中变量 `y` 的值无法传入 `testnestedfun` 主函数。

5.5.6 重载函数

重载函数即功能类似，但是参数类型或个数不同的函数。重载函数放置在不同的文件夹中，通常以 `@` 开头后面加上表示数据类型的字符，如 `@int16` 和 `@single`。当用户需要输入的参数是 16 位的整型变量和单精度浮点型变量时，可以编写两个相同文件名的函数。一个用来进行 16 位整型变量的输入，一个用来进行单精度浮点型变量的输入。当用户在调用函数时，系统会根据实际的变量类型，选择需要执行的函数。

5.6 P 码文件和变量使用范围

下面补充介绍两个重要的概念，一个是 P 码文件，一个是变量的使用范围。

5.6.1 P 码文件

当一个 M 文件首次被调用时，MATLAB 首先对其进行语法分析，然后生成相应的内部伪代码 (Pseudocode，简称 P 码)，最后将生成的伪代码存放在内存中。当此 M 文件再次被调用时，

系统会直接调用 M 文件在内存中的 P 码，而不会再对原 M 文件进行语法分析。

对于 P 码文件需要说明两点：

（1）生成的 P 码文件与原 M 文件具有相同的文件名，所不同的是 P 码文件的扩展名为.p。P 码文件的运行速度高于原 M 文件，一般在大规模的系统中效果较为明显。

（2）如果在 MATLAB 中存在同名的 P 码文件与原 M 文件，当 M 文件被调用时，系统调用的是 P 码文件。

【例 5-32】说明 P 码文件的使用方法，具体步骤如下所示：

首先建立内容如下的函数文件，并保存在 C:\Program Files\MATLAB\R2008a\work 目录下（该目录是搜索目录，可以不是当前目录）：

```
function output=testPcode(x)
if x>3
    output=x^2;
else
    output=2*x+3;
end
```

其次在命令窗口中输入如下语句：

```
pcode testPcode
```

此时在当前目录下生成 testPcode.p。当 testPcode 函数在一个线程中首次被调用时，MATLAB 将首先对其解析，然后将生成的 P 码存放在内存中。此后再次调用该函数时，将直接运行内存中的 P 码。

5.6.2 局部变量和全局变量

MATLAB 是一个完整的程序语言，有它自己的语句格式和语法结构。在编写程序的过程中，一般不需要事先定义变量（除全局变量外），但变量的命名必须满足如下规则：

- 以字母开头，之后可以是任意字母、数字或下画线，但是之间不能有空格。
- 变量名区分大小写。
- 变量名不能超过 63 个字符，第 63 个字符之后的部分将被忽略。

此外，MATLAB 还提供了如表 5-1 所示的一些特殊变量。

表 5-1 MATLAB 中的特殊变量

变量名称	变量含义
Ans	MATLAB 中的默认变量
Pi	圆周率
Eps	计算机中的最小数
Inf	无穷大
NaN	不定值，如 0/0
i(j)	复数中的虚数单位
Nargin	所用函数的输入变量数目
Nargout	所用函数的输出变量数目
Realmin	最小可用正实数
Realmax	最大可用正实数

用户使用的变量大致有两种类型，即局部变量与全局变量。

1. 局部变量

MATLAB 将每个变量保存在一块内存空间中，此空间称为工作空间。主工作空间，也称为基本工作空间，包括所有通过命令窗口创建的变量和脚本文件运行生成的变量。每个函数都有属于自己的局部变量，它们存放在各自的函数工作空间中，这些局部变量与主工作空间的变量和其他函数的变量分开存储，不会相互影响。局部变量只在函数内部使用，当函数调用结束后，局部变量即被删除。在函数文件中，变量默认为局部变量。

2. 全局变量

全局变量在全部工作空间内有效，当在某个工作空间内改变其值时，将改变它在其他工作空间的值。使用全局变量可以减少参数的传递，合理、有效利用全局变量将会提高程序的执行效率。但是全局变量会损害函数封装性，造成程序较难维护，所以不提倡大量使用全局变量。

MATLAB 提供了 `global` 命令用来声明全局变量，声明格式如下所示：

```
global var1 var2
```

`global` 函数需要说明以下几点：

- (1) 任何函数如果需要使用全局变量，必须首先声明。
- (2) 声明全局变量必须在该变量被使用之前。
- (3) MATLAB 对全局变量命名没有要求。
- (4) 当与全局变量联系的所有工作空间都被删除时，全局变量才会被删除。

【例 5-33】在近似计算 $\int_a^b \int_c^d \cos(xy)e^{-xy^2} dx dy$ 的函数中使用局部变量和全局变量：

```
function output=mymethod2(X,Y)
global xtmp ytmp
xmin=X(1);    %指定积分下限，x 为内积分变量
xmax=X(2);    %指定积分上限
xnum=X(3);    %指定包含两顶点的采样个数
ymin=Y(1);
ymax=Y(2);
ynum=Y(3);
xstep=(xmax-xmin)/(xnum-1); %相邻采样点的距离
x=xmin:xstep:xmax-xstep;    %采样点间等距离
ystep=(ymax-ymin)/(ynum-1);
y=ymin:ystep:ymax-ystep;

result=0; %保存计算结果的变量，需累加
%外循环，y 变化
for i=1:ynum-1
    ytmp=y(i);

    resulttmp=0; %保存临时计算结果的变量，需累加

    %内循环，x 变化
```



```
for j=1:xnum-1
    xtmp=x(j);
resulttmp=resulttmp+mymethodsubfun*xstep; %计算被积函数值并累加
end

result=result+resulttmp*ystep;
end
output=result;

%子函数
function output=mymethodsubfun
global xtmp ytmp
output=cos(xtmp * ytmp)*exp(xtmp * ytmp^2);
```

在命令窗口中输入如下语句:

```
clear
clc
global xtmp ytmp
X=[0 1 1001];
Y=[1 2 501];
Z1=mymethod2(X,Y)
Z2=xtmp+ytmp
Z3=xmin
```

命令窗口中的输出结果如下所示:

```
Z1 =
    1.5794

Z2 =
    2.9970
```

```
??? Undefined function or variable 'xmin'.
```

由本例的结果可以看出局部变量和全局变量的作用范围。

5.7 M 文件调试

用户在编写 M 文件时出现错误在所难免,能够熟练掌握调试的方法和技巧可以提高工作效率。M 文件的调试可以在文件编辑器中进行,因为文件编辑器不仅仅是一个文件编辑器,还是一个可视化的调试开发环境,M 文件的调试也可以在命令行中结合具体的命令进行。

5.7.1 M 文件出错信息

一般来说,M 文件错误可分为两种,即语法错误(Syntax Errors)和逻辑错误(Logic Errors)。

语法错误主要包括变量名、函数名的误写，标点符号的缺、漏等，这类错误在运行或 P 码编译时会被及时发现，程序会终止执行并给出错误原因，用户很容易根据提示信息发现并改正错误。

逻辑错误是程序本身的算法问题，它和语法错误相比难以处理。逻辑错误产生的原因很复杂，如算法是否正确，程序是否正确，MATLAB 指令的使用是否正确等。运行错误是动态的错误，对于 M 文件，一旦运行结束，所有函数工作空间便消失。由于逻辑错误产生的原因不定，影响因素较多，所以加大了用户调试的难度。

5.7.2 M 文件调试方法

通常程序的调试有两种方法，即直接调试法和工具调试法。

1. 直接调试法

MATLAB 的指令系统比较简单，所以程序通常也很简洁，对于比较简单的程序使用直接调试法是很有效的。

由于程序在运行的过程中只返回最后的输出结果，不返回中间变量，所以直接调试法可以通过分析中间变量的值，来查找具体的错误。下面介绍几种直接调试法，即查找中间变量值的方法。

(1) 单独调试一个函数时，将第一行的函数定义注释掉，并定义输入变量的值，然后以脚本的方式执行此 M 文件，这样可以保存中间变量，可以对这些结果进行分析，找出错误的原因。

(2) 在适当的位置添加 disp 函数用来显示输出变量值。

(3) 将重点怀疑语句后的分号“;”删掉，显示出运行结果，然后与预期值进行比较，找出错误的原因。

(4) 在原脚本文件或函数文件的适当位置添加 keyboard 指令。当 MATLAB 执行到 keyboard 指令时将暂停执行文件，并显示“k>>”提示符，用户可以查看或改变各个工作空间中存放的变量，在提示符后键入 return 指令可以继续执行原文件。

2. 工具调试法

MATLAB 提供了一些调试程序的工具，包括命令行形式和图形界面形式，很好地利用这些工具可以大大提高编程的效率。

(1) 命令行形式

命令行的程序调试适用于各种不同的平台，主要应用 MATLAB 提供的调试函数来调试程序，下面对 MATLAB 的一些命令调试方法进行简单介绍。

1) 设置断点

通过设置断点使程序执行到断点暂停，以便检查局部变量的值。MATLAB 提供了 dbstop 命令实现设置断点，表 5-2 列出了它的具体用法。

2) 清除断点

MATLAB 提供了 dbclear 命令实现清除断点，表 5-3 列出了清除断点的一些具体用法。

3) 恢复执行

MATLAB 提供了 dbcont 命令实现从断点处恢复程序的执行，直到遇到程序的下一个断点或错误后返回。

表 5-2 断点设置

命令	功能介绍
dbstop in mfile	在文件名为 mfile 的 M 文件的第一个可执行语句前设置断点。当程序运行到 mfile 的第一个可执行语句时，暂时中止 M 文件的运行，进入 MATLAB 的调试模式，此时用户可以使用各种调试工具、查看工作空间的变量等。需要说明的是，M 文件必须在 MATLAB 的搜索路径或当前目录内
dbstop in mfile at lineno	在文件名为 mfile 的 M 文件的第 lineno 行设置断点。当 lineno 行号所指向的语句是非执行语句的时候，程序运行到此行即停止执行，并在此行的下一个可执行语句前设置断点
dbstop in mfile at subfun	程序执行到子函数 subfun 处，暂时中止文件执行，进入 MATLAB 的调试模式
dbstop if error	当程序运行到 M 文件遇到错误时，终止文件执行，停止在产生错误的行，进入 MATLAB 的调试模式。这里提到的 error，不包括 try-catch 语句检测到的错误，用户不可以在 error 后重新开始程序的运行
dbstop if all error	遇到任何错误的时候，均终止文件执行，进入 MATLAB 的调试模式。这里提到的 all error，包括 try-catch 语句检测到的错误
dbstop if warning	遇到警告的时候，暂时中止文件执行，进入 MATLAB 的调试模式。运行暂停在 warning 出现的行，并且程序可以恢复运行
dbstop if caught error	try-catch 语句检测到错误时，暂时中止文件执行，并且程序可以恢复运行
dbstop if naninf(或 dbstop if infnan)	遇到无穷值或非数值时，暂时中止文件执行，进入 MATLAB 的调试模式

表 5-3 断点清除

命令	功能介绍
dbclear all	清除所有 M 文件的所有断点
dbclear all in mfile	清除文件名为 mfile 的 M 文件的所有断点
dbclear in mfile	清除文件名为 mfile 的 M 文件中第一个可执行语句前的断点
dbclear in mfile at lineno	清除文件名为 mfile 的 M 文件中第 lineno 行语句前的断点
dbclear in mfile at subfun	清除文件名为 mfile 的 M 文件中子函数 subfun 前的断点
dbclear if error	清除 dbstop if error 设置的断点
dbclear if warning	清除 dbstop if warning 设置的断点
dbclear if naninf（或 dbclear if infnan）	清除 dbstop if naninf（或 dbstop if infnan）设置的断点

4) 调用堆栈

MATLAB 提供了 dbstack 命令实现堆栈调用，表 5-4 列出了调用堆栈的具体用法。

表 5-4 堆栈调用

命令	功能介绍
dbstack	显示 M 文件名和行号。完整形式为[ST,I]=dbstack，其中 ST 是 M×1 的结构体，它将返回包含 3 个属性的堆栈信息 ST。I 是指当前工作空间的索引
dbstack(N)	省略了显示中的前 N 个帧
dbstack('completenames')	输出堆栈中每个函数的全名，函数的全名包括函数文件的名称和在堆栈中文件的包含关系

5) 执行多行语句

MATLAB 提供了 dbstep 命令实现多行语句执行，表 5-5 列出了执行多行语句的具体用法。

表 5-5 多行语句执行

命令	功能介绍
dbstep	执行当前的 M 文件的下一条可执行语句
dbstep nlines	执行当前的 M 文件的下 nlines 条可执行语句
dbstep in	当执行的语句中包含对另外一个函数的调用时, 此命令从被调用的函数的第一个可执行语句开始执行
dbstep out	执行函数剩余的部分, 直到离开此函数

6) 退出调试

MATLAB 提供了 dbquit 命令结束调试, 并返回到主工作空间。

(2) 图形界面形式

M 文件编辑器也是程序的编译器, 用户编写 M 文件后, 可以直接对 M 文件进行调试。MATLAB 的 M 文件编辑器是一个集成了代码编写和调试的开发环境, 使用户可以更方便、更直观地完成程序调试。

M 文件的调试主要通过如图 5-10 所示的 M 文件编辑器中的 Debug 菜单来实现。

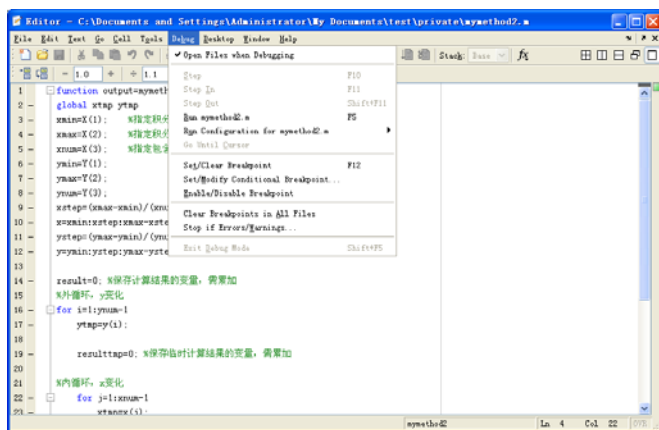


图 5-10 M 文件编辑器中的 Debug 菜单

下面分别介绍 Debug 菜单中各命令的功能。

1) Step

Step 用于在 M 文件的调试模式下, 执行 M 文件的当前行, 即单步执行。

2) Step In

Step In 用于在 M 文件的调试模式下, 如果执行 M 文件的当前行调用了另一个函数, 则深入被调用的函数内部执行程序。

3) Step Out

Step Out 用于跳出被调用的函数, 完成剩余函数的执行, 并退出函数。

4) Run

Run 用于执行当前 M 文件。

5) Go Until Cursor

Go Until Cursor 用于运行 M 文件到光标所在行。

6) Set/Clear Breakpoint

Set/Clear Breakpoint 用于在光标所在行设置或清除断点。

7) Set/Modify Conditional Breakpoint

Set/Modify Conditional Breakpoint 用于在光标所在行设置或修改条件断点,如图 5-11 所示。

8) Enable/Disable Breakpoint

Enable/Disable Breakpoint 用于设置当前行的断点为有效或无效。

9) Clear Breakpoint in All Files

Clear Breakpoint in All Files 用于清除所有 M 文件中设置的断点。

10) Stop if Errors/Warnings

Stop if Errors/Warnings 用于设置在什么情况下程序停止运行,如图 5-12 所示。

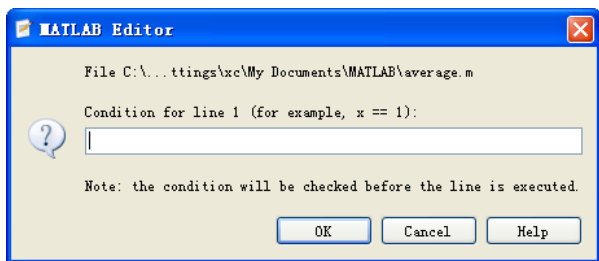


图 5-11 条件断点设置对话框

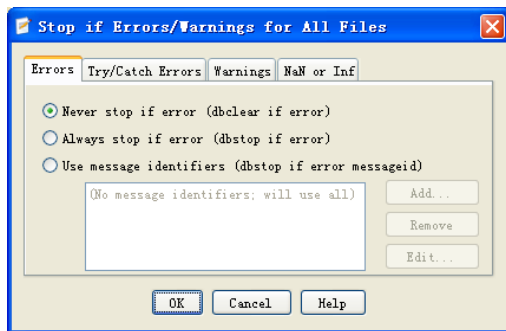


图 5-12 停止程序运行设置

11) Exit Debug Mode

Exit Debug Mode 用于退出调试模式。

对于以上命令,需要说明以下几点:

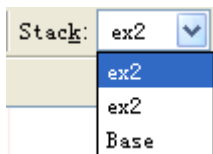










图 5-13 堆栈下拉菜单

①当 M 文件处于调试模式时,以上的命令才可以使用。

②工具栏的右侧有如图 5-13 所示的堆栈下拉菜单,通过菜单的选取可以对不同工作空间进行观察和操作。

③在 M 文件编辑器中,还提供了工具栏,用于对程序进行调试,工具栏会为用户调试程序提供很大的方便,下面给出工具栏图标和菜单命令的对应关系。

-  图标: Run。
-  图标: Set/clear breakpoint。
-  图标: Clear breakpoint in all files。
-  图标: Step。
-  图标: Step In。
-  图标: Step Out。
-  图标: Continue。
-  图标: Exit Debug Mode。

【例 5-34】使用以上调试方法调试近似计算 $\int_a^b \int_c^d \cos(xy)e^{xy^2} dx dy$ 的函数,具体步骤如下:

首先构建包含如下内容的 M 文件并保存:

```
function output= mymethod4(X,Y)
xmin=X(1);    %指定积分下限, x 为内积分变量
xmax=X(2);    %指定积分上限
```

```

xnum=X(3);    %指定包含两顶点的采样个数
ymin=Y(1);
ymax=Y(2);
ynum=Y(3);
xstep=(xmax-xmin)/(xnum-1); %相邻采样点的距离
x=xmin:xstep:xmax-xstep;    %采样点间等距离
ystep=(ymax-ymin)/(ynum-1);
y=ymin:ystep:ymax-ystep;

result=0; %保存计算结果的变量，需累加
%外循环，y 变化
for i=1:ynum-1
    ytmp=y(i);

    resulttmp=0; %保存临时计算结果的变量，需累加

    %内循环，x 变化
    for j=1:xnum-1
        xtmp=x(j);
        resulttmp=resulttmp+ mymethodsubfun(xtmp,ytmp); %计算被积函数值并累加
    %逻辑错误，应该为 resulttmp=resulttmp+ mymethodsubfun(xtmp,ytmp)*xstep;
    end

    result=result+resulttmp;
    %逻辑错误，应该为 result=result+resulttmp*ystep;
end
output=result;

```

%子函数

```

function output=mytsetsubfun(x,y)
output=cos(x*y)*exp(x*y^2); %语法错误，应该为 output=cos(x*y)*exp(x*y^2)

```

在命令窗口中输入如下语句：

```

X=[0 1 1001];
Y=[1 2 501];
mymethod4 (X,Y)

```

命令窗口中的输出结果如下所示：

```

??? Undefined function or variable 'y2'.

```

```

Error in ==> mymethod4>mymethodsubfun at 34

```

```

output=cos(x*y)*exp(x*y^2); %语法错误，应该为 output=cos(x*y)*exp(x*y^2)

```

Error in ==> mymethod4 at 23

```
resulttmp=resulttmp+mymethodsubfun(xtmp,ytmp); %计算被积函数值并累加
```

通过以上执行结果判断程序出现语法错误，对其进行修正：

```
output=sin(x*y)*exp(x^2*y);
```

然后保存并运行修正后的程序，在命令窗口中输入如下语句：

```
X=[0 1 1001];
```

```
Y=[1 2 501];
```

```
method1(X,Y)
```

命令窗口中的输出结果如下所示：

```
ans =
```

```
7.8972e+005
```

比较本例中的结果 $\text{ans} = 7.8972\text{e}+005$ 与例 5-33 中的结果 $Z1 = 1.5794$ ，结果的不同说明程序一定存在逻辑错误。

接下来使用工具调试法对以上程序进行调试。通过设置断点，检查变量的数值。如图 5-14 所示将断点设置在 `resulttmp=resulttmp+mymethodsubfun(xtmp,ytmp)` 之前。

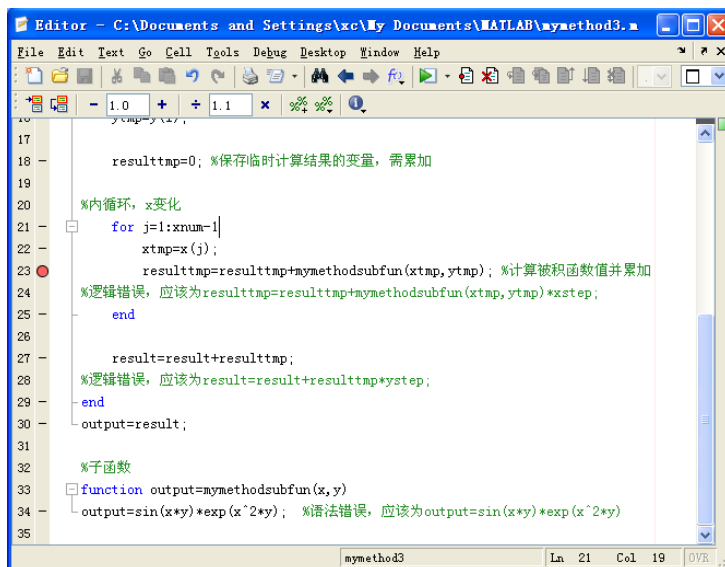


图 5-14 设置断点


在命令窗口中输入如下语句：

```
X=[0 1 5];
```

```
Y=[1 2 6];
```

```
mymethod4(X,Y)
```

当程序运行到断点处时，在断点和文本之间出现一个绿色箭头，将鼠标放置到变量处会显示变量的数值，此时 j 的值为 1，如图 5-15 所示。

单击  图标（单步执行）运行循环程序，如图 5-16 所示绿色箭头可以在循环体内移动，标明程序正在一步一步地执行，此时 j 的值为 2，程序完成了一次循环。

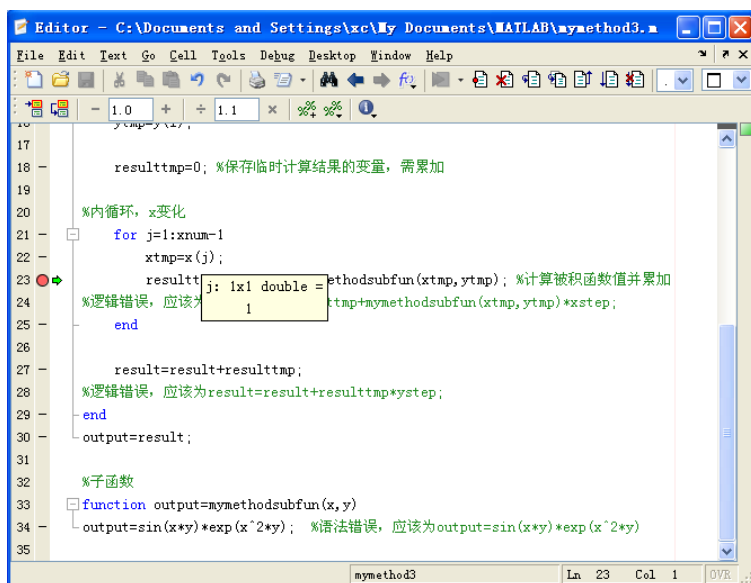


图 5-15 运行显示

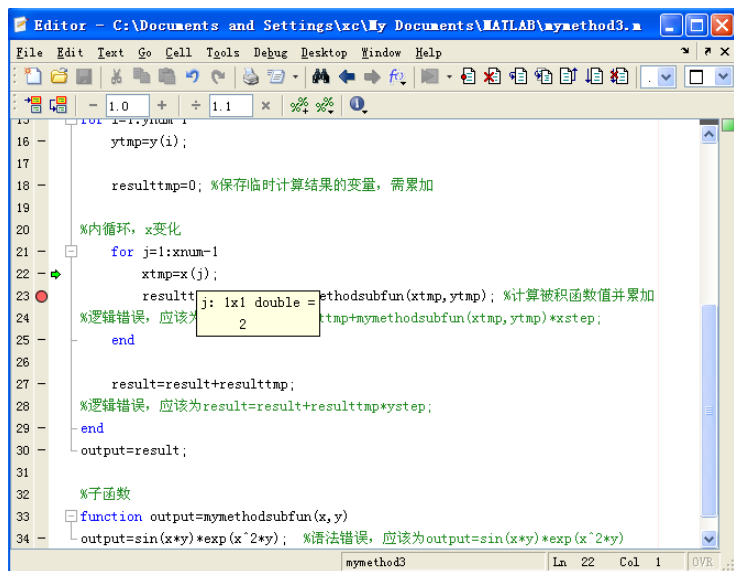


图 5-16 单步执行

在命令窗口中输入如下语句：

```
resulttmp
```

命令窗口中的输出结果如下所示：

```
resulttmp=1
```

通过单步执行和在命令窗口读取变量 `resulttmp` 值的配合，可以得到当 `j=4` 时 `resulttmp=4.0649`，此时可以发现 `resulttmp` 的值偏大。因此在此循环体内寻找逻辑错误，经排查发现如下语句有误，没有在调用函数的返回值上乘以步长：

```
resulttmp=resulttmp+mymethodsubfun(xtmp,ytmp)
```

根据上述逻辑错误，排查程序中是否还有类似错误，于是发现如下语句有误：


```
result=result+resulttmp;
```

最后运行修正后的程序，在命令窗口中输入如下语句：

```
X=[0 1 1001];
```

```
Y=[1 2 501];
```

```
Mymethod4(X,Y)
```

命令窗口中的输出结果如下所示：

```
ans =
```

```
1.5794
```

从上述运行结果可以看出，此结果与例 5-33 的结果相同，程序调试结束。

5.8 M 文件性能分析

用户通过 Debug 菜单可以对编写的程序进行调试，并改正程序的语法错误和逻辑错误，使程序按照指定功能运行。这里需要说明的是，程序虽然可以正常运行，但程序的运行效率或者性能未必是最优的。MATLAB 提供了 M-lint 工具和 Profiler 工具对程序进行分析和优化。

M-lint 工具和 Profiler 工具提供了图形操作界面，可以方便用户的使用，下面通过具体的实例来介绍这两个工具。

1. M-lint 工具

M-lint 工具用于分析 M 文件中的错误或者性能问题。

【例 5-35】通过 M-lint 工具分析以下脚本文件，保存为 mydata1.m:

```
clear
```

```
clc
```

```
[t,y] = ode23('lotka',[0 2],[20;20])
```

其中 ode23 函数实现微分方程的求解，lotka 函数为 MATLAB 自带的函数，其内容如下所示：

```
function yp = lotka(t,y)
```

```
%LOTKA Lotka-Volterra predator-prey model.
```

```
% Copyright 1984-2002 The MathWorks, Inc.
```

```
% $Revision: 5.7 $ $Date: 2002/04/15 03:33:21 $
```

```
yp = diag([1 - .01*y(2), -1 + .02*y(1)])*y;
```

使用 M-lint 工具进行分析的步骤：首先在 M 文件编辑器中打开所要分析的 M 文件，然后如图 5-17 所示选择“Tools”→“Code Analyzer”→“Show Code Analyzer Report”菜单命令，最后返回如图 5-18 所示的分析报告。报告包含两个方面的内容，即所分析的 M 文件的路径和分析结果，其中分析结果是由“行号：错误或问题报告”构成的。

分析报告一般包括语句没有以分号结束，变量在文件中并没有被其他的语句调用，循环过程中数组的尺寸增加等内容。

用户如果需要更改 M 文件来改善 M-lint 的分析结果，可以直接单击 M-lint 分析结果的行号，系统会直接定位到原 M 文件的相应内容。

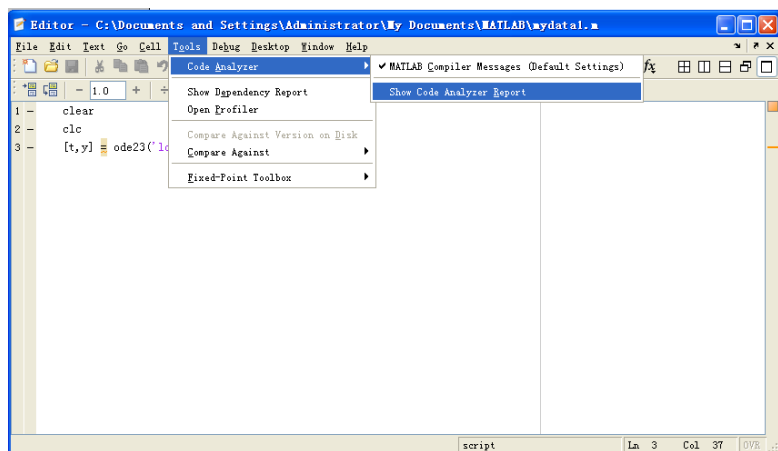


图 5-17 Tools 菜单

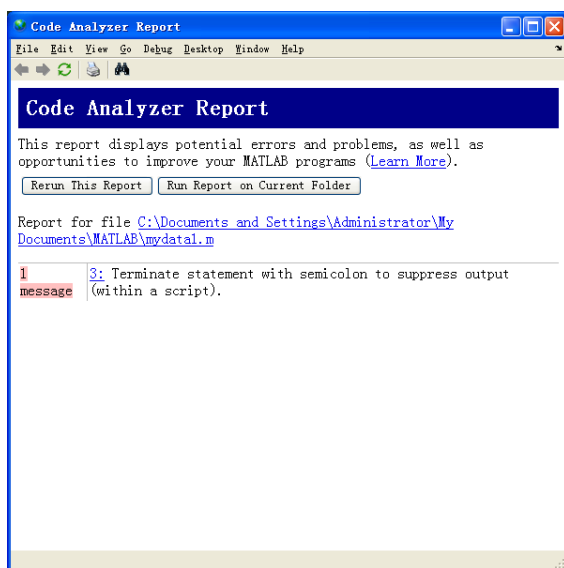


图 5-18 分析报告

2. Profiler 工具

MATLAB 提供的另一个代码分析工具是 Profiler 工具。

【例 5-36】通过 Profiler 工具分析例 5-35 的 mydata1.m 文件。

使用 Profiler 工具进行分析的步骤：首先在 M 文件编辑器中打开所要分析的 M 文件，其次如图 5-19 所示选择“Tools”→“Open Profiler”命令，进入如图 5-20 所示的 Profiler 工具图形界面，最后单击“Start Profiling”开始分析 M 文件，得到如图 5-21 所示的分析结果。

Profiler 的分析结果显示了所调用的函数名称、调用的次数和消耗时间等信息。

单击所调用的脚本名或函数名可以得到其更加详细的分析报告，如单击“lotka”可以得到如图 5-22 所示的结果。

详细分析结果显示了该文件在运行的过程中所耗费的时间和具体的耗时信息，用户可以充分利用这些信息优化程序，提高程序的运行效率。

同样单击函数调用的次数，可以显示每个函数在执行的过程中一共被调用了多少次，如图 5-23 所示。

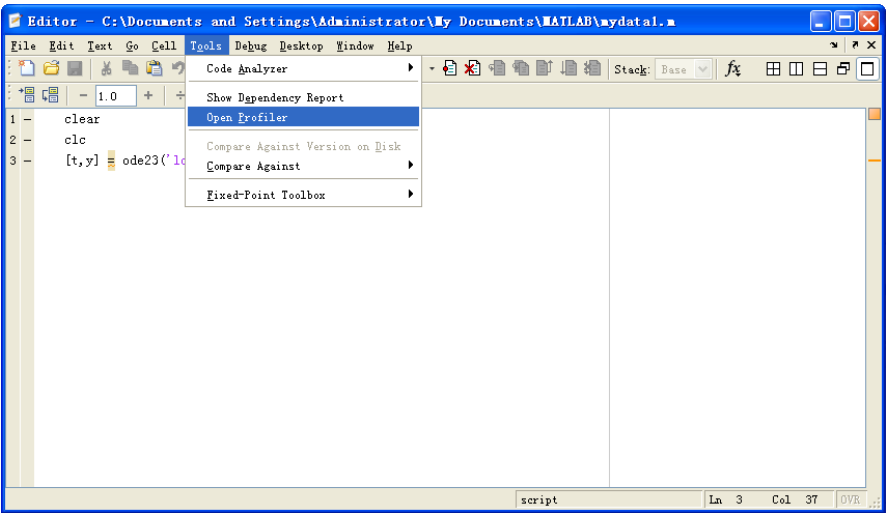


图 5-19 Tools 菜单

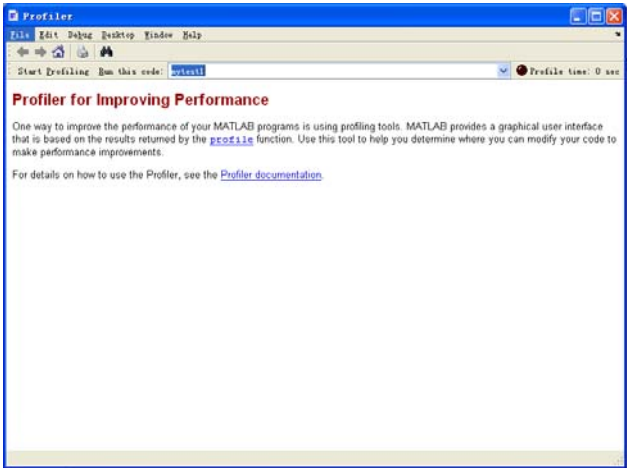


图 5-20 Profiler 工具图形界面

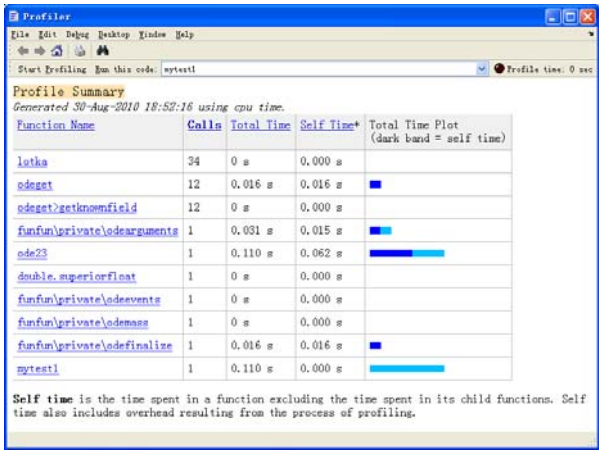


图 5-21 Profiler 的分析结果

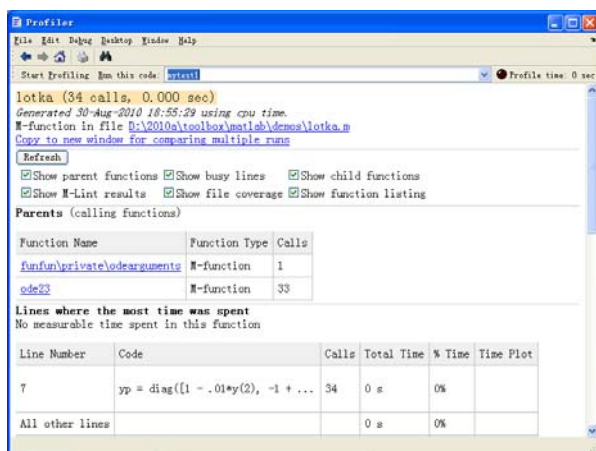


图 5-22 详细的分析结果

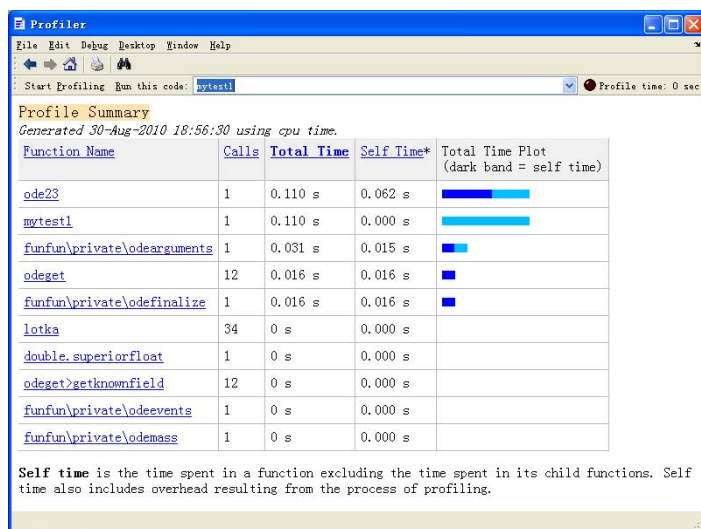


图 5-23 函数调用次数的分析结果

5.9 编程技巧

MATLAB 语言是一种解释性语言，有时 MATLAB 程序的执行速度不是很理想。为了使 MATLAB 程序执行的时间较少，可以采用以下几种方式：

(1) 尽量避免使用循环。

因为循环语句及循环体经常被认为是 MATLAB 编程的瓶颈问题，它会增加 MATLAB 的运行时间，改进这种状况有两种方式：

- 尽量使用向量化的运算来代替循环操作。
- 在必须使用多重循环的情况下，如果内外两重循环执行的次数不同，则使外层循环执行的次数少，内层循环执行的次数多，这样也可以显著提高速度。

(2) 大型矩阵的维数预先定义。

为大型矩阵动态地定义维数很费时间，应首先使用 MATLAB 的内置函数预先定义维数，然后再进行赋值，这样可以减少消耗的时间。

(3) 优先考虑内置函数。

进行矩阵运算的时候应该尽量使用 **MATLAB** 的内置函数，由于 **MATLAB** 的内置函数是由最基本的编程语言编写的，因此执行速度显然快于使用循环的矩阵运算。

(4) 应用 **MEX** 技术。

虽然采用了很多措施，但执行速度仍然很慢，比如使用循环有的时候是不可避免的，于是可以考虑使用其他语言，如 **C** 语言、**Fortran** 语言。用户可以按照 **MEX** 技术要求的格式编写相应部分的程序，然后通过编译形成由 **MATLAB** 直接调用的动态连接库（**DLL**）文件，这样可以很明显地加快运算速度。

(5) 采用有效的算法。

在实际应用中，解决同样的问题经常有很多种方法，在选择算法时应该加以比较，要选择最合适的方法。

第 6 章 Simulink 仿真

Simulink 是 MATLAB 的重要组成部分，它提供了集动态系统建模、仿真和综合分析于一体的图形用户环境。通过 Simulink 构造复杂仿真模型时，不需要书写大量的程序，只需要使用鼠标对已有模块进行简单的操作，以及使用键盘设置模块的属性。Simulink 是用于动态系统和嵌入式系统的多领域仿真和基于模型的设计工具，对各种时变系统，包括通信、控制、信号处理、视频处理和图像处理系统，Simulink 提供了交互式图形化环境和可定制模块库来对其进行设计、仿真、执行和测试。Simulink 可以非常容易地实现可视化建模，并把理论研究和工程实践有机地结合在一起，越来越受到人们的关注。

本章将系统地介绍 Simulink 的基本知识、常用模块集、特殊模块、子系统及其封装、模型仿真、M 文件与 Simulink 之间的数据交互、模型调试等内容。

6.1 Simulink 介绍

Simulink 是 Math Works 公司为 MATLAB 提供的系统模型化的图形输入与仿真工具，它使仿真进入到了模型化的图形阶段。Simulink 主要有两个功能，即 Simu（仿真）和 Link（连接），它可以针对自动控制、信号处理以及通信等系统进行建模、仿真和分析。Simulink 是一种基于 MATLAB 的框图设计环境，是实现动态系统建模、仿真和分析的一个软件包。Simulink 可以用连续采样时间、离散采样时间或两种混合的采样时间进行建模，它也支持多速率系统，也就是系统中的不同部分具有不同的采样速率。Simulink 的最新版本是 Simulink 7.5，本章将着重介绍这个版本的功能及其使用方法。


6.1.1 Simulink 概述

Simulink 具有独立的功能，它支持线性系统、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的建模和仿真，并且模型可以是多进程的。

Simulink 的显著特点是它具有模型化的图形输入，即 Simulink 提供了许多按功能分类的模块，Simulink 中的模型可以由这些模块组成的框图来表示。

用户使用 Simulink 建模，只需要知道模块的输入、输出、属性参数以及模块的功能，而不必清楚模块内部的操作。因此，用户使用 Simulink 进行系统建模的任务就是如何选择合适的模块并把它们按照模型结构连接起来，最后进行调试和仿真。建立模型后，可以利用 Simulink 中的菜单或 MATLAB 命令窗口中的命令对模型进行仿真，并且可以通过 Simulink 提供的分析工具对模型进行分析，如各种仿真算法、线性化、寻找平衡点等。

启动 Simulink 可以采用如下两种方式：

- 在 MATLAB 的命令窗口中输入“Simulink”命令。
- 单击 MATLAB 工具栏中的图标。

通过以上两种方式，都可以得到如图 6-1 所示的 Simulink 模块库浏览器。此模块库浏览器

列出了按功能分类的各种模块的名称。

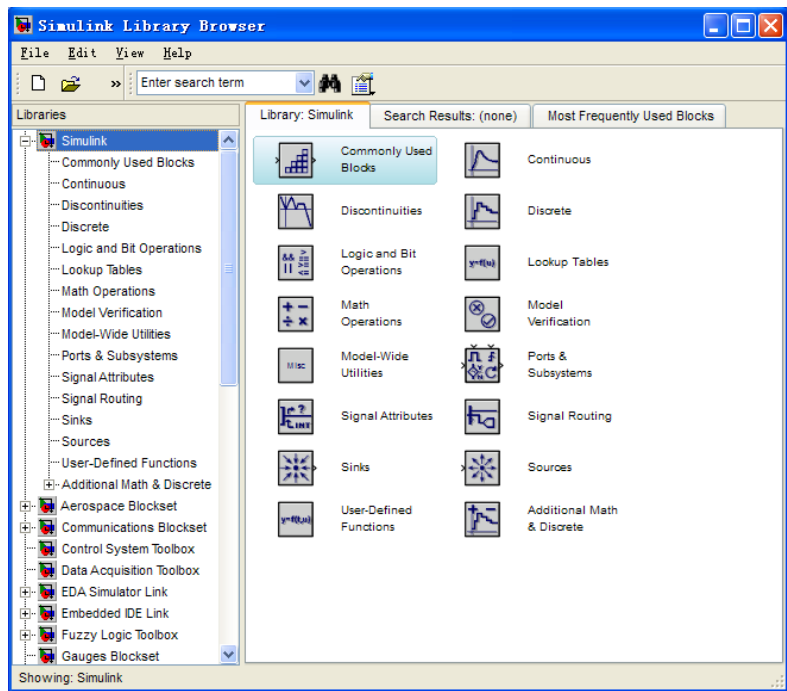



图 6-1 Simulink 模块库浏览器

新建模型窗口可以采用如下两种方式：

- 单击 Simulink 模块库浏览器左上方的  图标。
- 在 MATLAB 的菜单栏中选择“File”→“New”→“Model”命令。

通过以上两种方式，都可以得到如图 6-2 所示的新建模型窗口，所有的模型构建都在此窗口进行。

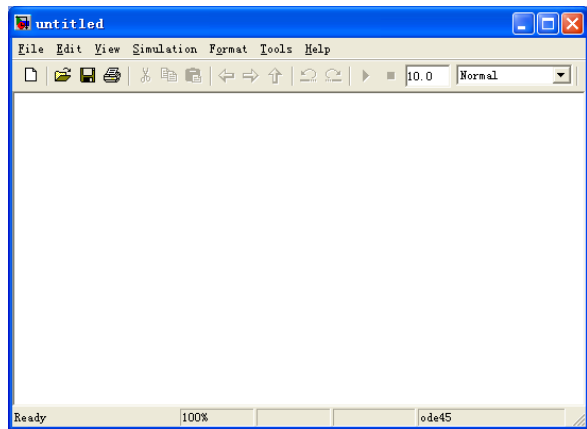



图 6-2 新建模型窗口

打开已经存在的模型文件可以采用如下三种方式：

- 在 MATLAB 的命令窗口输入模型文件名，要求此文件在当前目录或搜索路径下。

- 在 MATLAB 的菜单栏中选择“File”→“Open”命令。
- 单击 Simulink 模块库浏览器左上方的  图标。



6.1.2 Simulink 窗口介绍

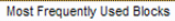

下面介绍 Simulink 模块库浏览器窗口和 Simulink 模型窗口。

1. Simulink 模块库浏览器窗口

启动 Simulink 以后,可以得到如图 6-1 所示的 Simulink 模块库浏览器。下面对 Simulink 模块库浏览器进行简单介绍,在 6.2 节将详细介绍 Simulink 模块库中的常用模块集。

Simulink 模块库浏览器窗口包括如下部分:

- Simulink 模块库的菜单栏: 。
- Simulink 模块库的工具栏: 。
- Simulink 模块库的搜索栏: ，通过在搜索栏中输入关键词,可以进行模块的搜索,此功能对于用户查找模块十分有用。
- Simulink 模块库的库目录树,如图 6-3 所示。
- Simulink 模块库中的模块,如图 6-4 所示。
- Simulink 7.10 相对于之前的 Simulink 7.1 版本,增加了

 标签。如果在实际应用中多次使用了某几个模块,则使用频率较高的这几个模块会出现在  标签的下拉菜单中,如图 6-5 所示,顶端最右边的标签即为高频使用模块标签。

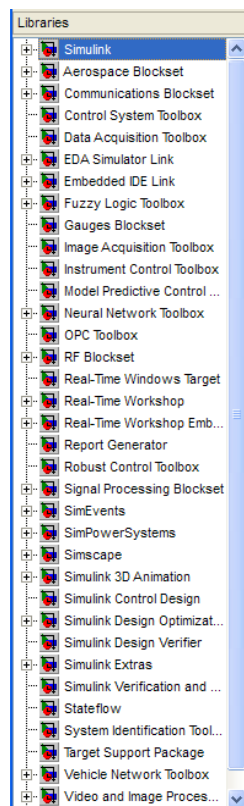


图 6-3 库目录树

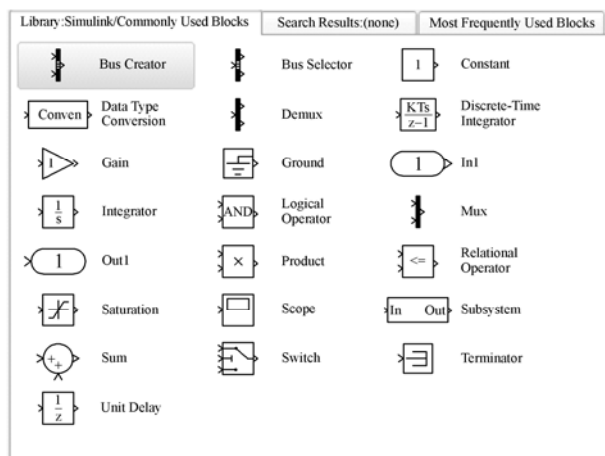


图 6-4 模块

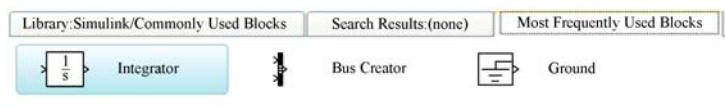



图 6-5 高频使用模块标签

2. Simulink 模型窗口

除了选择和查找模块外,用户建立和仿真模型的大部分工作将在模型窗口(如图 6-2 所示)

中完成，因此需要熟练掌握模型窗口中菜单和工具的功能。

下面介绍如图 6-2 所示的 Simulink 模型窗口的主要部分：

- Simulink 模型窗口的菜单栏：。
- Simulink 模型窗口的工具栏，如图 6-6 所示。

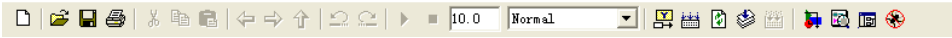


图 6-6 工具栏

- Simulink 模型窗口的状态栏，如图 6-7 所示。



图 6-7 状态栏

(1) Simulink 模型窗口的菜单栏

Simulink 模型窗口的菜单栏包括“File”菜单、“Edit”菜单、“View”菜单、“Simulation”菜单、“Format”菜单、“Tools”菜单和“Help”菜单。

1) “File”菜单

“File”菜单主要用于对模型文件进行处理，表 6-1 介绍了“File”菜单中主要的子菜单及其功能。

表 6-1 “File”菜单

主要子菜单	快捷键	功能
New	Ctrl+N	新建模型（Model）或库（Library）
Open	Ctrl+O	打开模型
Close	Ctrl+W	关闭模型
Save	Ctrl+S	保存模型
Save as		将模型另存为
Source Control		设置 Simulink 与 SCS 的接口
Model Properties		打开模型属性对话框（如图 6-8 所示）
Preferences		打开模型参数设置对话框（如图 6-9 所示），模型参数设置对话框主要用于设置一些用户界面的显示形式，如颜色、字体等
Print	Ctrl+P	打印模型或模块图标到一个文件
Print Details		生成 HTML 格式的模型报告文件，包括模块的图标和模块参数的设置等
Print Setup		打印模型或模块图标
Exit MATLAB	Ctrl+Q	退出 MATLAB

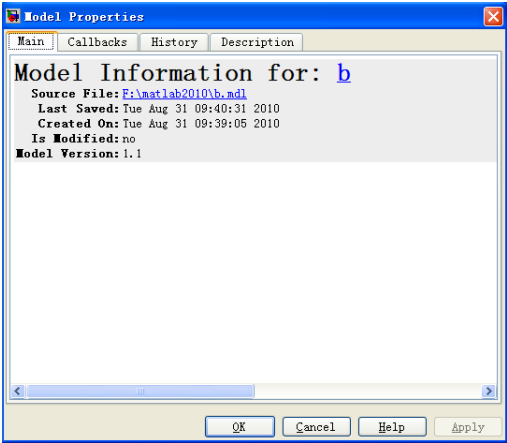


图 6-8 模型属性对话框

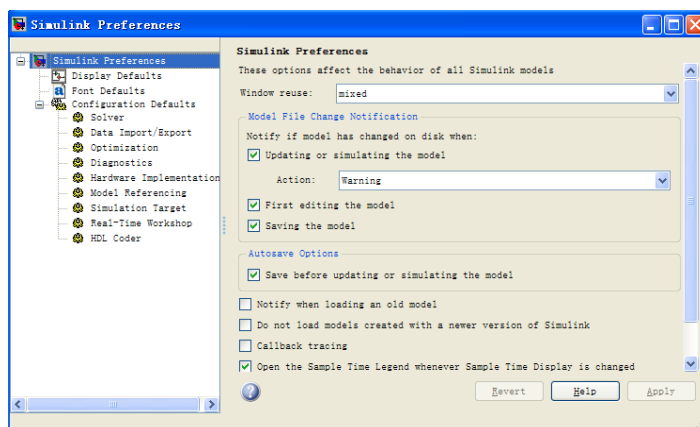


图 6-9 模型参数设置对话框

2) “Edit” 菜单

“Edit” 菜单主要用于对模块进行处理，表 6-2 介绍了“Edit” 菜单中主要的子菜单及其功能。

表 6-2 “Edit” 菜单

主要子菜单	快捷键	功能
Cut	Ctrl+X	剪切
Copy	Ctrl+C	复制
Paste	Ctrl+V	粘贴
Paste Duplicate Inport		粘贴复制的模块
Copy Model To Clipboard		将模型复制到剪贴板
Explore		打开模型浏览器，只有当模块被选中时才可用
Block Properties		打开模块属性对话框，只有当模块被选中时才可用
Create Subsystem	Ctrl+G	创建子系统，只有当模块被选中时才可用
Mask Subsystem	Ctrl+M	封装子系统，只有当子系统被选中时才可用
Look Under Mask	Ctrl+U	查看封装子系统的内部结构，只有当子系统被选中时才可用
Refresh Model Blocks	Ctrl+K	刷新输入、输出和参数设置
Update Diagram	Ctrl+D	更新模型框图的外观

3) “View” 菜单

“View” 菜单主要用于对模型进行处理，表 6-3 介绍了“View” 菜单中主要的子菜单及其功能。

表 6-3 “View” 菜单

主要子菜单	快捷键	功能
Go To Parent		展示其直接父模型
Toolbar		显示工具栏
State Bar		显示状态栏
Model Browser Options		显示模型浏览器
Block Data Tips Options		用于设定在鼠标指针移到某一模块时是否显示模块的相关提示信息（如模块名、模块参数名及其值和用户自定义描述字符串）
System Requirements		设置系统要求
Library Browser		激活如图 6-1 所示的模型库浏览器
Model Explorer	Ctrl+H	打开如图 6-10 所示的模型资源管理器，将模块的参数设置、仿真参数设置以及解法器选择、模块的各种信息等集成到一个界面
Zoom In		放大显示模型
Zoom Out		缩小显示模型
Fit System To View		自动选择合适的显示比例
Normal		显示正常比例
Port Values		设置通过鼠标操作显示模块端口的当前值

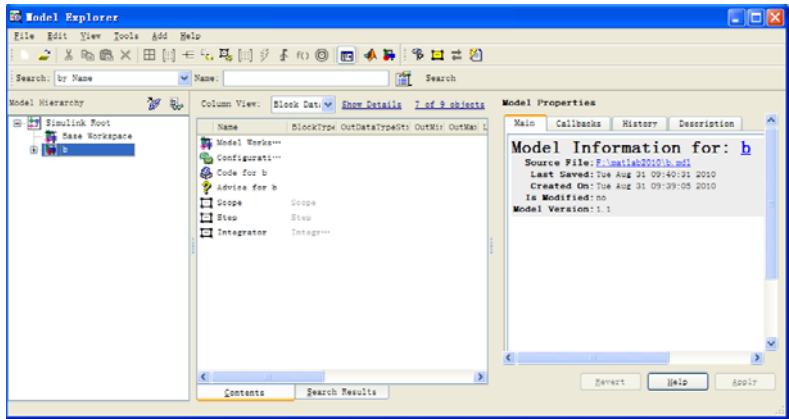


图 6-10 模型资源管理器

4) “Simulation” 菜单

“Simulation” 菜单主要用于对模型仿真进行操作，表 6-4 介绍了“Simulation” 菜单中主要的子菜单及其功能。

表 6-4 “Simulation” 菜单

主要子菜单	快捷键	功能
Start	Ctrl+T	开始仿真
Stop		停止仿真
Configuration Parameters	Ctrl+E	如图 6-11 所示设置仿真参数，如设定解法器及仿真步长
Normal		正常工作模式
Accelerator		加速仿真模式
Rapid Accelerator		快速仿真模式
External		外部工作模式

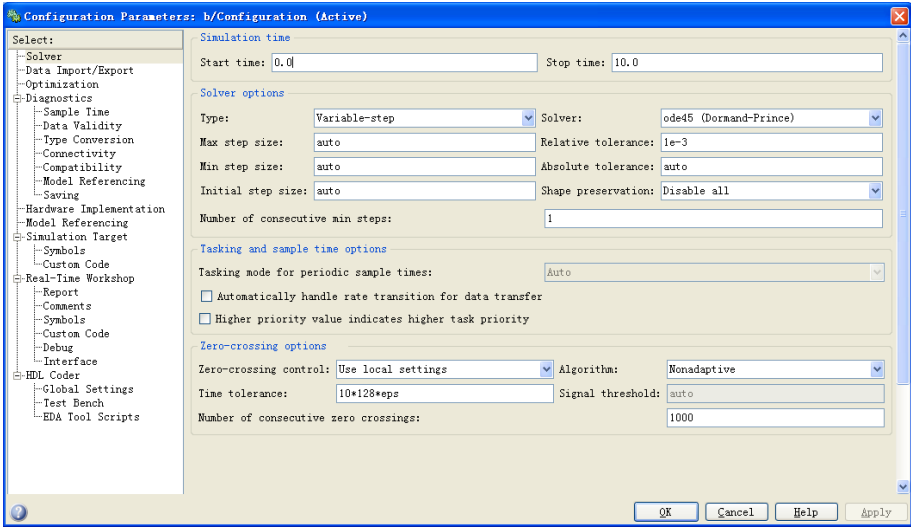


图 6-11 设置仿真参数

5) “Format” 菜单

“Format” 菜单主要用于设置字体、屏幕颜色、模块名的显示、模块显示颜色、信号和端口类型和宽度等，表 6-5 介绍了“Format” 菜单中主要的子菜单及其功能。

表 6-5 “Format” 菜单

主要子菜单	功能
Font	字体设置
Text Alignment	标注文字对齐工具
Enable Tex Commands	设置 Tex 控制命令生效
Flip Name	翻转模块名字
Flip Block	翻转模块图标
Rotate Block	旋转模块图标
Show Name	显示模块名
Show Drop Shadow	显示模块阴影
Show Port Labels	显示端口标签
Foreground Color	前景颜色，从子菜单中选取
Background Color	后景颜色，从子菜单中选取
Screen Color	屏幕颜色
Align Blocks	对齐模块
Distribute Blocks	分布模块
Resize Blocks	重新定义模块大小
Port/Signal Displays	显示端口和信号的相关信息，其中“Sample Time Colors”选项根据模块的采样时间来设置不同的显示颜色
Block Displays	显示模块相关信息
Library Link Display	库连接显示

6) “Tools” 菜单

“Tools” 菜单主要用于打开各种编辑器或设置对话框，以便设置与模型仿真相关的参数，表 6-6 介绍了“Tools”菜单中主要的子菜单及其功能。

表 6-6 “Tools” 菜单

主要子菜单	功能
Simulink Debugger	打开调试器
Model Advisor	打开模型分析器对话框，帮助用户检查和分析模型的配置
Fixed-Point Settings	打开定点设置对话框
Lookup Table Editor	打开查表编辑器，帮助用户检查和修改模型中的 Lookup Table (LUT) 模块的参数
Data Class Designer	打开数据类设计器，帮助用户创建 Simulink 类的子类，即创建自定义数据类
Bus Editor	打开 Bus 编辑器，帮助用户修改模型中 Bus 类型对象的属性
Profiler	选中此菜单项，当仿真运行结束后会自动生成并弹出一个仿真报告文件 (HTML 格式)
Coverage Settings	打开“Coverage Settings”对话框，用户可以通过该对话框设置在仿真结束后给出仿真过程中有关 Coverage Data 的一个 HTML 格式报告文件
Signal & Scope Manager	打开信号和示波器的管理器，帮助用户创建各种类型的信号生成模块和示波器模块
Real-Time WorkShop	用于将模块转换为实时可执行的 C 代码
External Mode Control Panel	打开外部模式控制板，用于设置外部模式的特性
Control Design	打开“Control and Estimation Tools Manager”和“Simulink Model Discretizer”窗口
Parameter Estimation	打开“Control and Estimation Tools Manager”窗口，如图 6-12 所示，可用于模型的参数分析
Report Generator	打开报告生成器

7) “Help” 菜单

“Help” 菜单主要用于查阅 Simulink 的联机帮助，表 6-7 介绍了“Help”菜单中主要的子菜单及其功能。

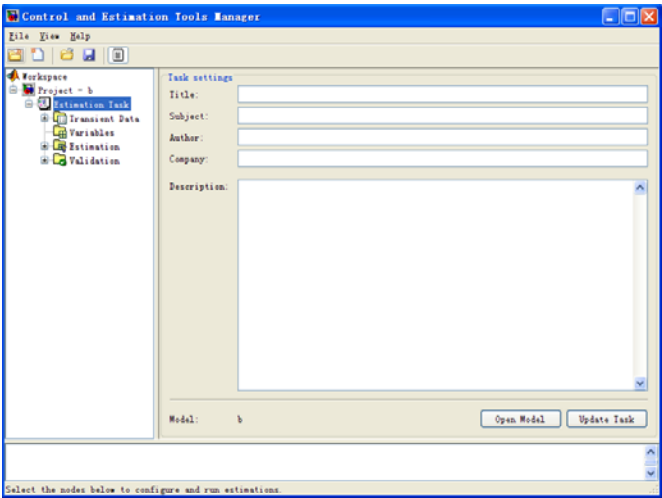


图 6-12 “Control and Estimation Tools Manager” 窗口

表 6-7 “Help” 菜单

主要子菜单	功能
Using Simulink	打开 MATLAB 的帮助，显示 Simulink 帮助部分
Blocks	打开 MATLAB 的帮助，显示按字母排序的 Blocks 帮助部分
Blocksets	打开按应用方向分类的帮助
Block Support Table（图 6-13）	打开如图 6-14 所示的模块支持的数据类型帮助文件
Shortcuts	打开 MATLAB 的帮助，显示鼠标和键盘快捷键设置的帮助部分
S-Function	打开 MATLAB 的帮助，显示 S-函数的帮助部分
Demos	打开 MATLAB 的帮助，显示 Demos 页的帮助部分，通过它可以打开许多有用的演示示例，如图 6-15～图 6-17 所示为某个 Demos 演示示例
About Simulink	显示如图 6-18 所示的 Simulink 版本

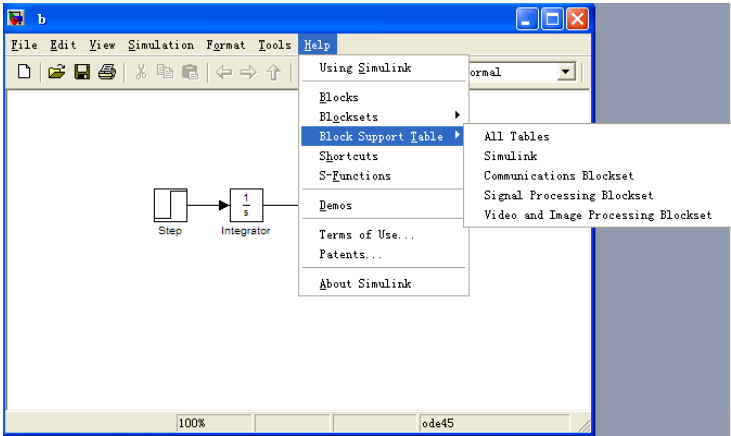


图 6-13 “Block Support Table” 下拉菜单

下面具体讲解一下 Demos 的用法：

单击图 6-15 中的“Simulation of a Bouncing Ball”，进入关于 Simulation of a Bouncing Ball 的详细讲解页面，如图 6-16 所示。单击图 6-16 中的“Open this model”，打开 Demos 演示示例的详细.mdl 文件，如图 6-17 所示。

Simulink Block Data Type Support

The following table describes the data types that the blocks in the main Simulink library support. All blocks that can generate code contain an "X" in the column titled Code Generation Support. A subset of these blocks are not recommended for production code as flagged by footnote 4. Guidelines to determine when a block is recommended for production code are listed below the table.

Some blocks have footnotes that should be considered when using them. Footnotes are indicated in the table by numbers in parenthesis "[#]", and are described below the table.

Table legend: If a column has an:

- "X", the block supports that data type or capability.

Sublibrary	Block	Double	Single	Boolean	Base Integer	Fixed-Point	Enumerated	Bus	Code Generation Support	Multidimension Support	Variable-Size Support	For Each Subsystem Support
Additional Math & Discrete/Additional Discrete	Fixed-Point State-Space	X	X	X (1)	X	X			X		X	X
	Transfer Fcn Direct Form I	X	X		X	X			X (2)			
	Transfer Fcn Direct Form II Time Varying	X	X	X (1)	X	X			X (2)			
	Unit Delay Enabled External IC	X	X	X	X	X	X		X (2)			X
	Unit Delay Enabled Resettable	X	X	X	X	X	X		X (2)			X
	Unit Delay Enabled Resettable External IC	X	X	X	X	X	X		X (2)		X	X
	Unit Delay External IC	X	X	X	X	X			X (2)		X	X
	Unit Delay External IC	X	X	X	X	X			X (2)		X	X
	Unit Delay External IC	X	X	X	X	X			X (2)		X	X
	Unit Delay External IC	X	X	X	X	X			X (2)		X	X

图 6-14 模块支持的数据类型帮助文件

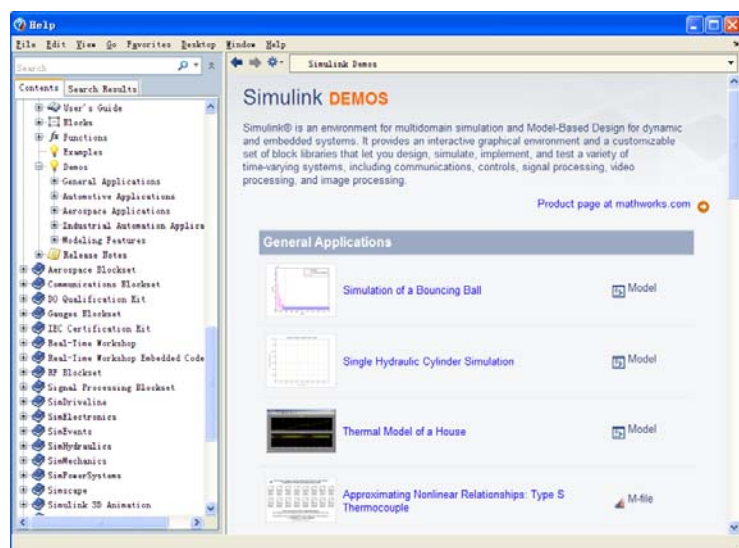


图 6-15 Demos 帮助界面

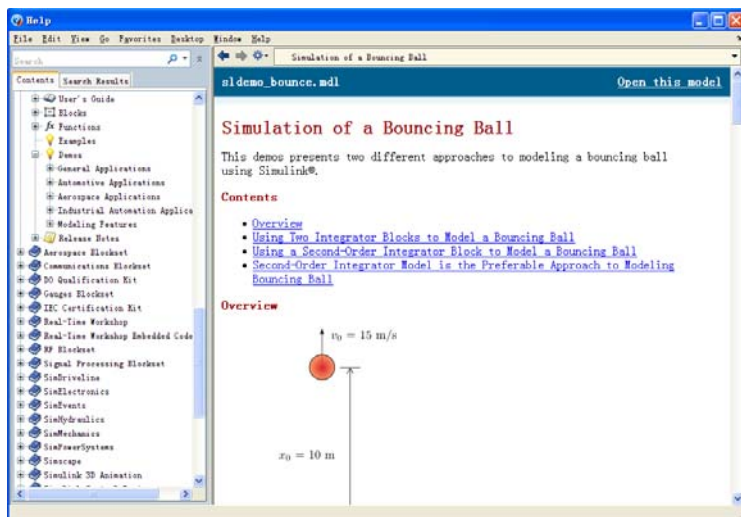


图 6-16 Demos 演示示例

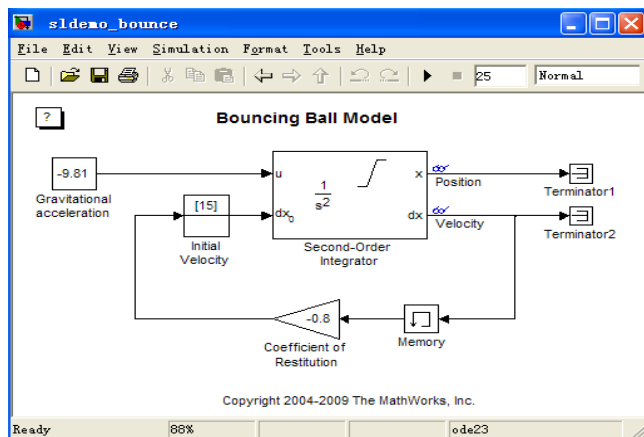


图 6-17 Demos 演示示例的详细文件



图 6-18 Simulink 版本

(2) Simulink 模型窗口的工具栏

- □图标：新建模型。
- 图标：打开模型。
- 图标：保存模型。
- 图标：打印。
- 图标：剪切。
- 图标：复制。
- 图标：粘贴。
- 图标：后退。
- 图标：前进。
- 图标：显示父模型。
- 图标：撤销。
- 图标：重复。
- 图标：开始仿真。
- 图标：停止仿真。
- 10.0 图标：设置仿真时间。

-  图标：设置仿真加速模式。
-  图标：准备。
-  图标：产生 RTW 代码。
-  图标：刷新。
-  图标：更新。
-  图标：为子系统产生代码。
-  图标：显示库浏览器。
-  图标：打开模型管理器。
-  图标：打开或隐藏模型浏览器。
-  图标：打开调试器。

(3) Simulink 模型窗口的状态栏

Simulink 模型窗口状态栏的“Ready”表示模型已经准备就绪，等待仿真指令；“100%”表示模型的显示比例；“ode45”表示仿真所选用的算法。

6.1.3 Simulink 运行原理

Simulink 提供了实现各种功能的模块，为用户提供了很大的方便。Simulink 建模大体可分为两步：创建模型的图标和控制 Simulink 对它进行仿真。如果用户希望灵活、高效地使用 Simulink，还需要了解 Simulink 的运行原理。

1. 图形化模型和现实系统之间的映射关系

每一个现实系统都有输入、输出和状态三个基本元素，以及它们之间随时间变化的数学函数关系。Simulink 的模型中每一个图形化模块都可以用图 6-19 表示，它代表了现实系统中某个部件或整体的输入、输出和状态间随时间变化的函数关系，即为系统的数学模型。系统的数学模型是由一系列的数学方程来描述的，每一个模块均代表一个数学方程，Simulink 称这些数学方程为模型的方法。

Simulink 模型的典型结构通常分三部分，即信源、系统和信宿，它们的关系如图 6-20 所示。

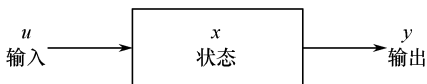


图 6-19 模块的图形化形式

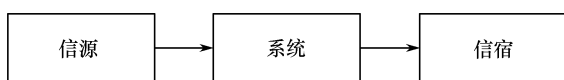


图 6-20 Simulink 模型的典型结构

- 信源：可以是常数、时钟、脉冲信号、正弦波、脉冲生成器等信号源。
- 系统：对仿真对象的数学抽象，可以是连续线性系统，也可以是连续非线性系统等。
- 信宿：接收信号的部分，接收器可以是示波器、图形记录仪等。

信源、系统和信宿可以直接从 Simulink 模型库中获取，也可以自行创建。当然，对于具体的 Simulink 模型而言，不一定完全包含这三部分。

2. 利用映射关系进行仿真

Simulink 对模型进行仿真，就是在用户定义的时间段内根据输入计算出状态和输出，并将计算结果予以显示或保存。

Simulink 的仿真过程包括：

- 模型编译阶段。任务是调用模型编辑器，将模型编译成可以执行的形式。
- 连接阶段。任务是创建按执行的次序排列的方法运行列表，同时定位和初始化存储在每

个模块的信息。

- 仿真环阶段。此阶段又分为两个子阶段：一个是仿真环初始化阶段，此阶段只运行一次，用于初始化模型的状态和输出；另一个是仿真环迭代阶段，此阶段在定义的时间段内每个采样点运行一次，用于在每个采样点计算模型新的输入、输出和状态，并且更新模型。

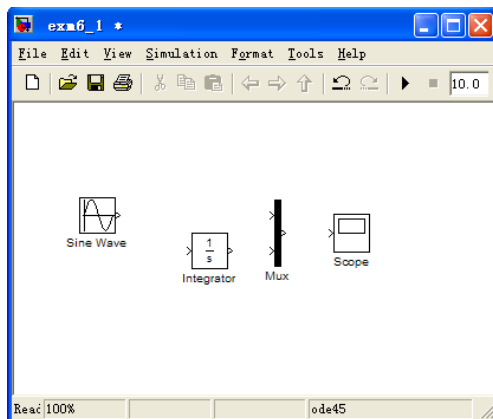
下面，通过一个简单的例子让读者对 Simulink 有一个直观的认识，之后再详细说明如何利用 Simulink 进行仿真。

【例 6-1】在 Simulink 中构建模型：对一个正弦波信号进行积分处理，然后将原始正弦信号和积分后的信号送到示波器中同时显示出来。

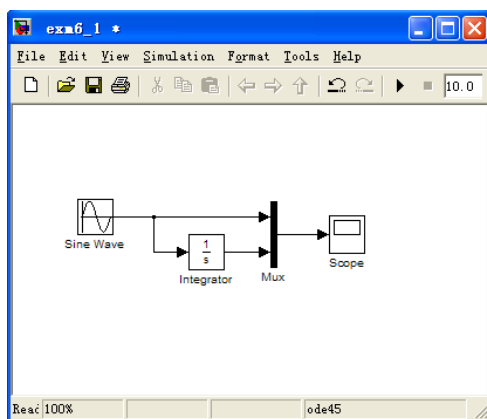
(1) 打开 Simulink 的模块库浏览器和模型编辑窗，默认情况下模型名为“Untitled”。然后展开模块库浏览器中的 Simulink 节点，并且选中 Sources 模块库中的 Sine Wave 模块，单击鼠标右键，在弹出的快捷菜单中选择“Add to Untitled”菜单项，就可以把 Sine Wave 模块添加到“Untitled”模型中去。也可以不通过右键菜单，直接将 Sine Wave 模块拖曳到模型编辑窗中，完成模块的添加操作。

(2) 采用与上一步相同的方法将 Continuous 模块库中的 Integrator 模块（积分器）、Sink 子模块库中的 Scope 模块（示波器）、Signal Routing 模块库中的 Mux 模块（复路器）等模块，一一添加到模型编辑窗中，并排列成如图 6-21 (a) 所示。

(3) 连接信号线。将鼠标放到模块的输出端口，等光标变为十字形状时按住鼠标左键移动到另一个模块的输入端口，释放鼠标即可。如果信号线存在分路，按住“Ctrl”键的同时将鼠标放到信号线上，等光标变为十字形状即可引出分支信号线。连线后的系统如图 6-21 (b) 所示。




(a) 模块置入图



(b) 信号连接线完成图

图 6-21 Simulink 仿真简例

(4) 保存模型文件。执行“File”→“Save as”菜单命令，将文件命名为“exm6_1.mdl”后保存。

(5) 执行“Simulation”→“Start”菜单命令或单击工具栏中的  图标，开始仿真，该操作的快捷键为“Ctrl+T”。

(6) Simulink 默认的仿真时间是 10s（注意这并不是实际流逝的时间），结束仿真后，双击 Scope 模块，可以看到仿真结果如图 6-22 所示。

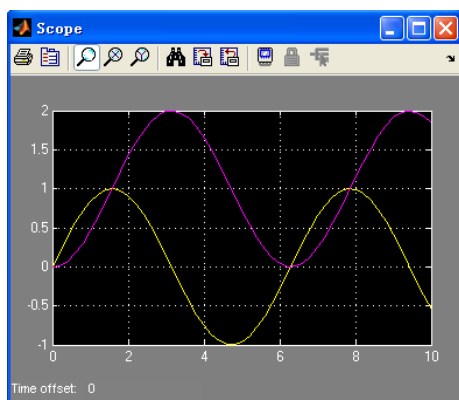


图 6-22 示波器输出的仿真结果

在图 6-22 所示的示波器输出图形中,靠下方的黄线代表 Sine Wave 信号源产生的正弦波形,靠上方的紫线表示的是正弦波积分后的输出波形。

注:模型中的 Mux 模块是复路器,其作用是将多路信号综合后加以输出,此模型就是将原始信号和它积分后的信号合成为一个矢量信号进行输出。

6.2 Simulink 常用模块

使用 Simulink 对模型进行仿真,首先要绘制仿真模型框图,其次确定仿真所需要的参数,然后进行调试。这个过程也可以简单地理解为从模块库里选择合适的模块,然后将它们连接在一起,最后进行调试。

模块是仿真模型框图的基本单元,Simulink 模块库提供了大量的常用模块用于构造仿真模型框图。模块库浏览器将各种模块库按树状结构进行罗列,以便用户快速地查询所需模块,同时它还提供了按名称查找的功能。

熟练掌握 Simulink 的前提是要了解各个模块的作用,下面主要介绍 Simulink 模块库常用模块集中部分模块的功能及使用。

1. 常用模块

从库目录树中选择“Simulink”→“Commonly Used Blocks”,可以得到如图 6-23 所示的常用模块,表 6-8 介绍了常用模块的模块名及其功能。

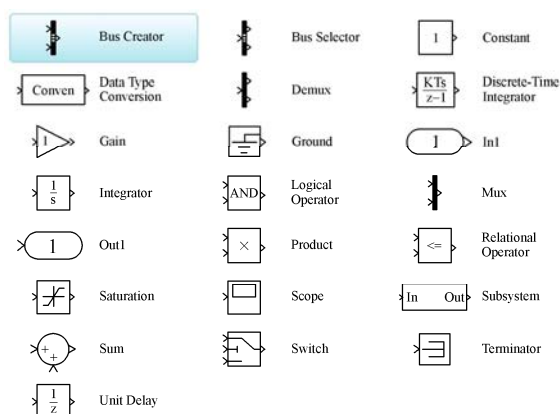


图 6-23 常用模块

表 6-8 常用模块

模块名	功能
Bus Creator	将输入信号合并成向量信号
Bus Selector	将输入向量分解成多个信号，输入只接受从 Mux 和 Bus Creator 输出的信号
Constant	输出常量信号
Data Type Conversion	数据类型的转换
Demux	将输入向量转换成标量或更小的标量
Discrete-Time Integrator	离散积分器模块
Gain	比例增益模块
Ground	接地
In1	输入模块
Integrator	连续积分器模块
Logical Operator	逻辑运算模块
Mux	将输入的向量、标量或矩阵信号合成
Out1	输出模块
Product	乘法器，执行标量、向量或矩阵的乘法
Relational Operator	关系运算，输出布尔类型数据
Saturation	定义输入信号的最大和最小值
Scope	输出示波器
Subsystem	创建子系统
Sum	加法器
Switch	选择器，根据第二个输入信号来选择输出第一个还是第三个信号
Terminator	终止输出，用于防止模型最后的输出端口没有接任何模块时报错
Unit Delay	单位时间延迟

2. 连续模块

从库目录树中选择“Simulink”→“Continuous”，可以得到如图 6-24 所示的连续模块，表 6-9 介绍了连续模块的模块名及其功能。

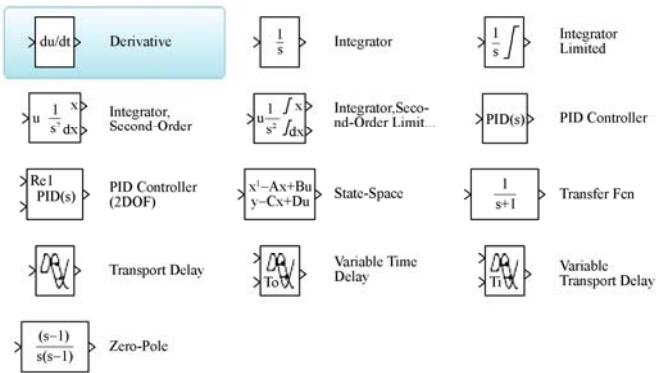


图 6-24 连续模块

3. 非连续模块

从库目录树中选择“Simulink”→“Discontinuities”，可以得到如图 6-25 所示的非连续模块，表 6-10 介绍了非连续模块的模块名及其功能。

表 6-9 连续模块

模块名	功能
Derivative	连续微分器模块
Integrator	连续积分器模块
Integrator Limited	带限幅的积分器模块
Integrator,Second-Order	二阶积分器模块
Integrator,Second-Order Limited	带限幅的二阶积分器模块
PID Controller	PID 控制器
PID Controller(2DOF)	带设定值加权的 PID 控制器
State-Space	创建状态空间系统模型 $dx/dt = Ax + Bu$ $y = Cx + Du$
Transfer Fcn	传递函数模型
Transport Delay	定义传输延迟
Variable Time Delay	可变时间延迟
Variable Transport Delay	定义传输延迟，第一个输入接收输入；第二个输入接收延迟时间
Zero-Pole	用矩阵描述系统零点，用向量描述系统极点和增益

4. 离散模块

从库目录树中选择“Simulink”→“Discrete”，可以得到如图 6-26 所示的离散模块，表 6-11 介绍了离散模块的模块名及其功能。

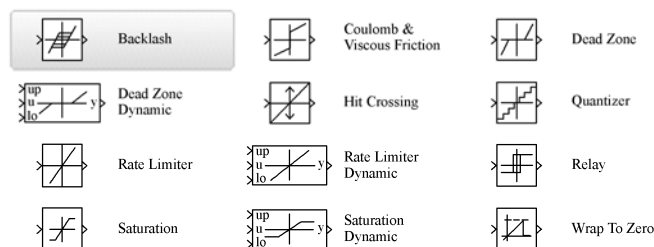


图 6-25 非连续模块

表 6-10 非连续模块

模块名	功能
Backlash	间隙非线性
Coulomb&Viscous Friction	库仑与黏度摩擦非线性
Dead Zone	产生死区，当输入在某一范围取值时输出为 0
Dead Zone Dynamic	产生死区，当输入在某一范围取值时输出为 0，与 Dead Zone 不同的是，它的死区范围在仿真过程中是可变的，即动态死区非线性
Hit Crossing	检测输入是上升经过某一值，还是下降经过这一值，或是固定在某一值，用于过零检测
Quantizer	按相同的间隔离散输入，即量化非线性
Rate Limiter	限制输入的上升和下降速率在某一范围
Rate Limiter Dynamic	限制输入的上升和下降速率在某一范围，与 Rate Limiter 不同的是，它的范围在仿真过程中是可变的
Relay	判断输入与某两阈值的大小关系，当大于开启阈值时，输出为 on；当小于关闭阈值时，输出为 off；当在两者之间时输出不变
Saturation	限制输入在最大和最小范围之内
Saturation Dynamic	限制输入在最大和最小范围之内，与 Saturation 不同的是，它的范围在仿真过程之中是可变的
Wrap To Zero	当输入大于某一值时输出 0，否则输出等于输入

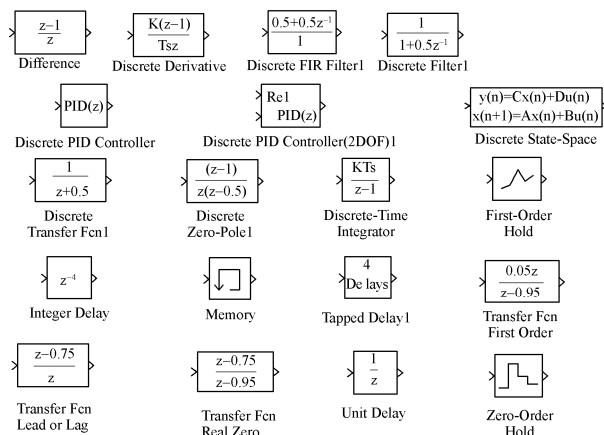


图 6-26 离散模块

5. 查找表模块

从库目录树中选择“Simulink”→“Lookup Tables”，可以得到如图 6-27 所示的查找表模块，表 6-12 介绍了查找表模块的模块名及其功能。

表 6-11 离散模块

模块名	功能
Difference	差分环节
Discrete Derivative	离散微分环节
Discrete Filter	离散滤波器
Discrete FIR Filter	离散 FIR 滤波器
Discrete PID Controller	离散 PID 控制器
Discrete PID Controller(2DOF)	带设定值加权的离散 PID 控制器
Discrete State-Space	创建离散状态空间系统模型 $x(n+1) = Ax(n) + Bu(n)$ $y(n) = Cx(n) + Du(n)$
Discrete Transfer Fcn	离散传递函数模型
Discrete Zero-Pole	以零极点表示的离散传递函数模型
Discrete-Time Integrator	离散时间积分器
First-Order Hold	一阶保持器
Integer Delay	整数倍采样周期的延迟
Memory	存储单元，当前输出是前一时刻的输入
Tapped Delay	延迟
Transfer Fcn First Order	离散一阶传输函数，单位的直流增益
Transfer Fcn Lead or Lag	传递函数
Transfer Fcn Real Zero	离散零点传递函数
Unit Delay	一个采样周期的延时
Zero-Order Hold	零阶保持器

6. 数学模块

从库目录树中选择“Simulink”→“Math Operations”，可以得到如图 6-28 所示的数学模块，同时表 6-13 介绍了数学模块的模块名及其功能。

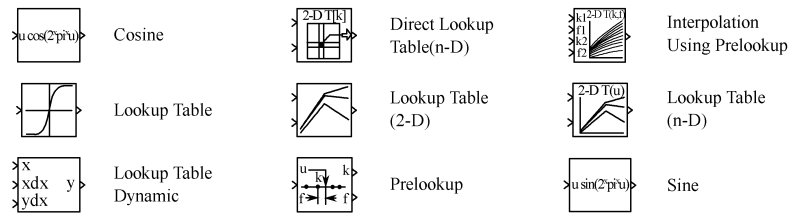


图 6-27 查找表模块

表 6-12 查找表模块

模块名	功能	模块名	功能
Cosine	余弦函数查询表	Lookup Table (n-D)	n 维输入信号的查询表
Direct Lookup Table (n-D)	n 个输入信号的查询表	Lookup Table Dynamic	动态查询表
Interpolation Using Prelookup	n 个输入信号的预插值	Prelookup	预查询
Lookup Table	输入信号的查询表	Sine	正弦函数查询表
Lookup Table (2-D)	两维输入信号的查询表		

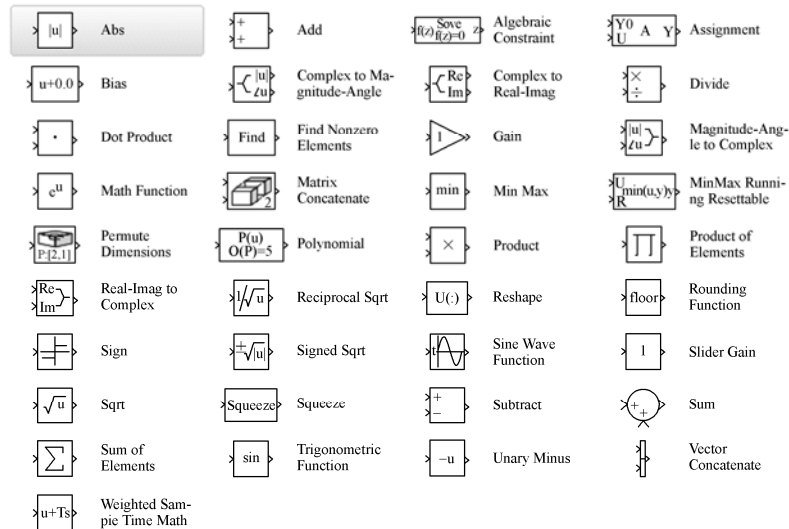


图 6-28 数学模块

表 6-13 数学模块

模块名	功能
Abs	取绝对值
Add	加法运算
Algebraic Constraint	代数约束
Assignment	赋值
Bias	偏移
Complex to Magnitude-Angle	将复数输入转换成幅值和幅角的输出
Complex to Real-Imag	将复数输入转换成实部和虚部的输出
Divide	实现除法或乘法
Dot Product	点乘运算
Find Nonzero Elements	求线性指数或非零输入值的下标
Gain	比例运算
Magnitude-Angle to Complex	将幅值和幅角的输入合成复数输出
Math Function	实现常用数学函数运算

(续表)

模块名	功能
Matrix Concatenation	实现矩阵的串联
Min Max	最值运算
MinMax Running Resettable	最小和最大值运算
Permute Dimensions	重新安排元素位置
Polynomial	多项式求值，多项式的系数以数组的形式定义
Product	乘运算
Product of Elements	元素乘运算
Real-Imag to Complex	将实部和虚部的输入合成复数输出
Reciprocal Sqrt	平方根函数的倒数
Reshape	改变输入信号的维数
Rounding Function	将输入的整数部分输出
Sign	判断输入的符号，若为正输出 1，为负输出-1，为零输出 0
Signed Sqrt	符号平方根函数
Sine Wave Function	产生一个正弦函数
Slider Gain	可变增益
Sqrt	平方根函数
Squeeze	去除数组中长度为 1 的维
Subtract	减法运算
Sum	加法运算
Sum of Elements	元素的和运算
Trigonometric Function	三角函数，包括正弦、余弦、正切和余切等
Unary Minus	一元减法
Vector Concatenate	数组数据的合成
Weighted Sample Time Math	根据采样时间实现输入的加法、减法、乘法和除法，只对离散信号适用

7. 信号接收器模块

从库目录树中选择“Simulink”→“Sinks”，可以得到如图 6-29 所示的信号接收器模块，表 6-14 介绍了信号接收器模块的模块名及其功能。

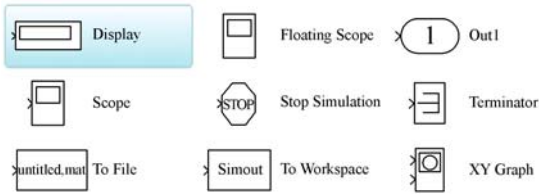


图 6-29 信号接收器模块

表 6-14 信号接收器模块

模块名	功能
Display	数字显示器
Floating Scope	浮置示波器，由用户来设置所要显示的数据
Out1	输出端口
Scope	示波器
Stop Simulation	当输入不为零时，停止仿真
Terminator	连接到未连接的输入端口
To File	将输入和时间写入 MAT 文件
To Workspace	将输入和时间写入 MATLAB 工作空间中的数组或结构中
XY Graph	将输入分别当成 X、Y 轴数据绘制成二维图形

8. 信号源模块

从库目录树中选择“Simulink”→“Sources”，可以得到如图 6-30 所示的输入源模块，表 6-15 介绍了信号源模块的模块名及其功能。

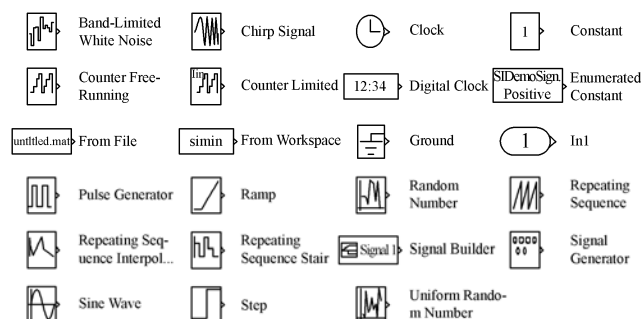


图 6-30 信号源模块

表 6-15 信号源模块

模块名	功能
Band-Limited White Noise	有限带宽的白噪声
Chirp Signal	产生 Chirp 信号
Clock	输出当前仿真时间
Constant	输出常数
Counter Free-Running	自动计数器，发生溢出后又从 0 开始
Counter Limited	有限计数器，当计数到某一值后又从 0 开始
Digital Clock	以数字形式显示当前的仿真时间
Enumerated Constant	输出枚举型常数
From File	从 MAT 文件中读取数据
From Workspace	从 MATLAB 的工作空间读取数据
Ground	接地
In1	输入信号
Pulse Generator	脉冲发生器
Ramp	斜坡信号输入
Random Number	产生正态分布的随机数
Repeating Sequence	重复输出某一数据序列
Repeating Sequence Interpolated	重复序列内插值
Repeating Sequence Stair	重复阶梯序列
Signal Builder	信号创建器
Signal Generator	信号发生器
Sine Wave	产生正弦波信号
Step	产生阶跃信号
Uniform Random Number	按某一分布在某一范围内生成随机数

9. 用户自定义函数模块

从库目录树中选择“Simulink”→“User-Defined Functions”，可以得到如图 6-31 所示的用户自定义函数模块，表 6-16 介绍了用户自定义函数模块的模块名及其功能。

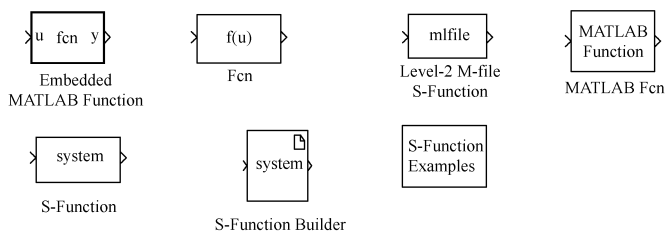


图 6-31 用户自定义函数模块

表 6-16 用户自定义函数模块

模块名	功能
Embedded MATLAB Function	内置 MATLAB 函数模块，在模型窗口中双击该模块图标就会弹出 M 文件编辑器
Fcn	简单的 MATLAB 函数表达式模块
Level-2 M-file S-Function	Level-2 型 M 文件 S-函数
MATLAB Fcn	用于指定自编函数
S-Function	用于指定 S-函数
S-Function Builder	具有 GUI 界面的 S-函数创建器
S-Function Examples	S-函数演示模块，在模型窗口中双击该模块图标可以看到多个 S-函数示例

需要说明的是，表 6-16 中提到的 S-函数将在第 8 章详细讲解，GUI 相关知识将在第 7 章详细讲解。

除了以上列举的模块外，在 Simulink 模块库浏览器下拉菜单中，还有许多其他的模块，用户可根据需要查看各个模块的功能。

10. 举例

由于模块单独使用没有太大的意义，所以需要利用上面提到的模块构造模型。下面通过具体的例子介绍一些模块的使用。

添加一个模块，只需在模块库浏览器中找到该模块，选中并拖放到模型窗口中即可；删除一个模块，只需在模型窗口中选中并删除即可；模块之间的连接比较简单，只需首先选中一个模块的输出端口，然后用鼠标拖动到另一个模块的输入端口即可，或者首先选中一个模块的输出端口，然后用鼠标拖动到已经存在的连线上即可；设置模块参数只需双击指定模块，然后设置相应参数即可。

【例 6-2】使用常用的信号接收器和信号源模块建立简单的 Simulink 模型。

首先建立如图 6-32 所示的模型并保存。

其次按如下方式设置模块的参数。

- Sine Wave 模块：如图 6-33 所示设置频率参数 Frequency 为 $\pi/4$ 和偏置角参数 Phase 为 $\pi/2$ 。
- Integrator 模块：连续积分器模块。
- Scope 模块：如图 6-34 所示取消勾选用于设定保存数据个数的参数 Limit data points to last，否则只保存最后 5000 个数据。

最后运行该模型。双击打开 Scope 模块并单击鼠标右键选择“Autoscale”选项，可以得到如图 6-35 所示的结果。

【例 6-3】建立连续系统的仿真模型。

首先建立如图 6-36 所示的模型并保存。

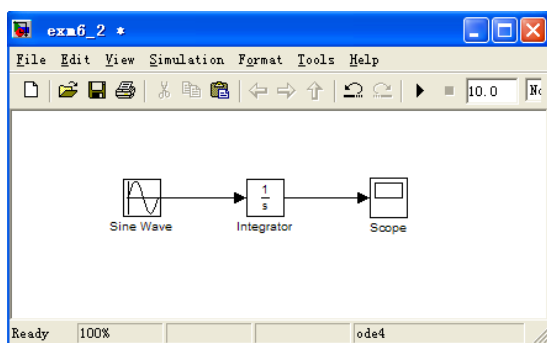


图 6-32 Simulink 模型

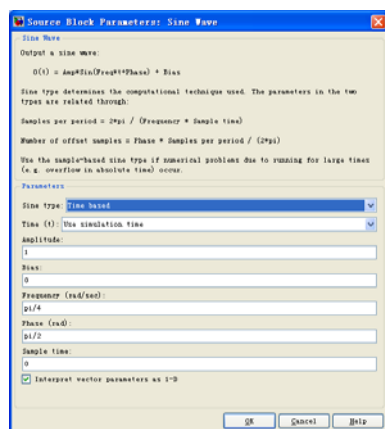


图 6-33 Sine Wave 模块设置

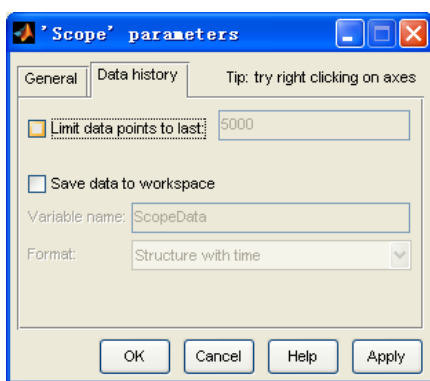


图 6-34 Scope 模块设置

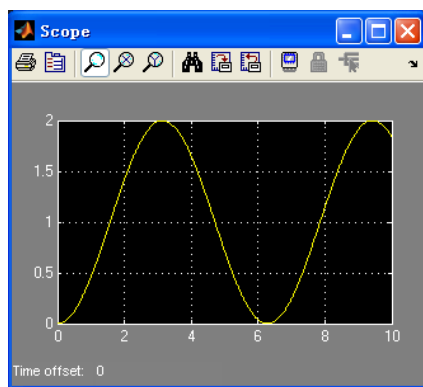


图 6-35 Scope 模块运行结果

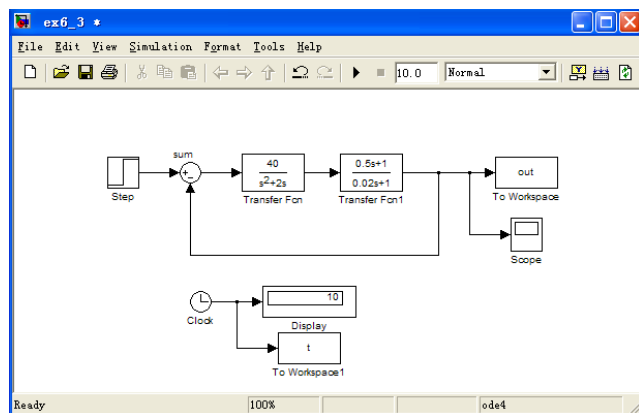


图 6-36 Simulink 模型

其中需要说明的是:

(1) 图中文字“sum”是手动填加的, 只需用鼠标单击要填加的区域, 写入要填加的文字即可。标志为的 sum 模块的设置如图 6-37 所示。在“Icon shape”下拉菜单中, 可以设置 sum 模块的形状为“round”和“rectangular”, 即圆形和矩形, 本例中我们选择的是“round”; 在 List of signs 中, 设置 sum 模块的“+”和“-”数目, 本例中是一个负反馈, 因此在“List of signs”中将默认的“++”改为“+-”, 如果需要相加的有三个量, 那么相应地应改为“+++”。

(2) Step 模块的阶跃开始时间“Step time”改为 0，终值“Final value”改为 2，如图 6-38 所示。

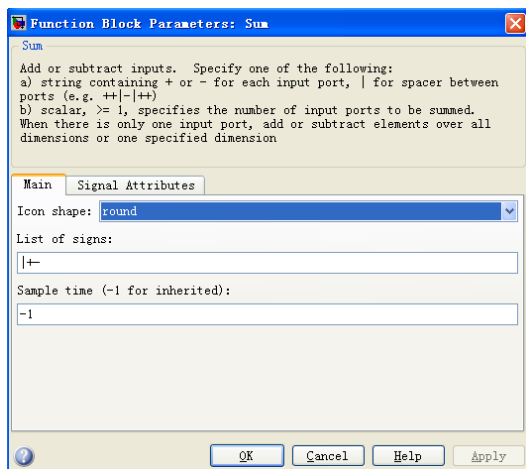


图 6-37 sum 模块设置

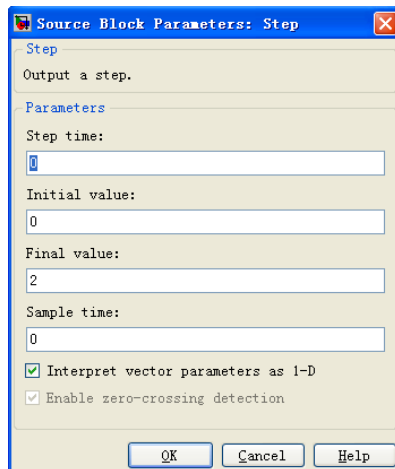


图 6-38 Step 模块设置

(3) Transfer Fcn 模块的设置如图 6-39 所示，本例中的两个传递函数要分别设置参数，其中传递函数的分子和分母以多项式的方式设置。分子和分母的多项式排列均为降阶顺序，注意第一个传递函数 $\frac{40}{s^2 + 2s}$ 的分母应设置为 [1 2 0]，即常数项为 0，这时不要忘记补 0，否则如果设置为 [1 2]，则传递函数将被错误地表达为 $\frac{40}{s + 2}$ 。

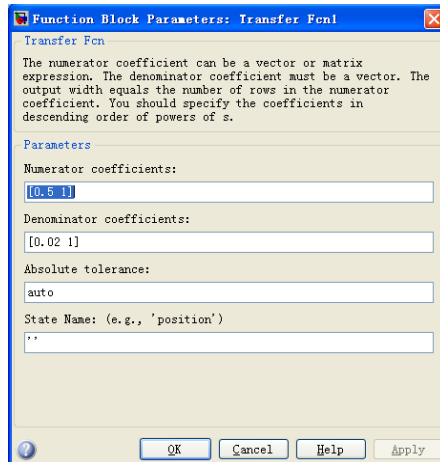
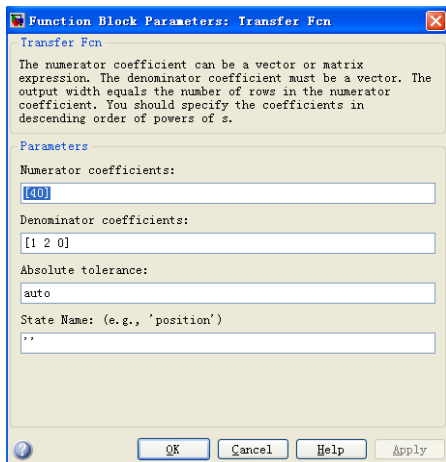


图 6-39 Transfer Fcn 模块设置

(4) To Workspace 模块的设置如图 6-40 所示，其中保存格式为数组类型。分别将两个 To Workspace 模块的变量名字设置为“out”和“t”。

(5) Display 模块用来显示时间，如图 6-41 所示为该模块的设置对话框。在某些情况下，用户需要观察或动态显示某个信号的数值结果时，可以选用 Display 模块，它既可以显示单个信号，也可以显示向量信号或矩阵信号（帧信号），此功能是通过“Decimation”选项来设置的，本例中只显示时间，所以将“Decimation”设为 1。

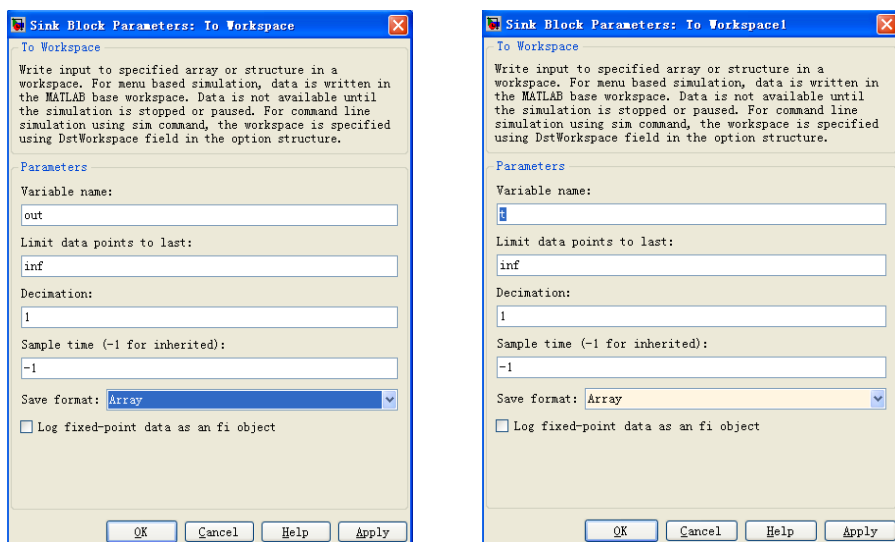


图 6-40 To Workspace 模块设置

(6)使用 Scope 模块显示仿真结果。单击  按钮可以使信号动态范围与 Scope 显示相匹配,从而方便用户对输出信号进行观察。双击 Scope 模块,该模型的仿真结果如图 6-42 所示。

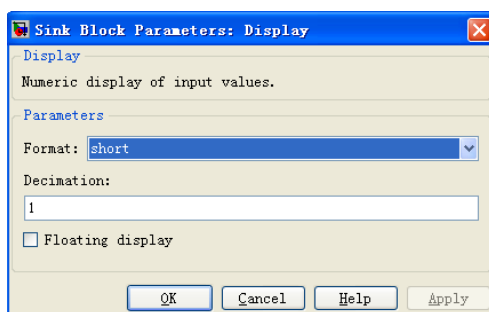


图 6-41 Display 模块设置

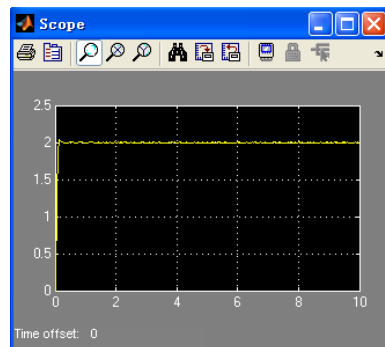


图 6-42 Scope 模块运行结果

此时,工作空间中已经存储了数组值,双击“out”查看存储结果,如图 6-43 所示。

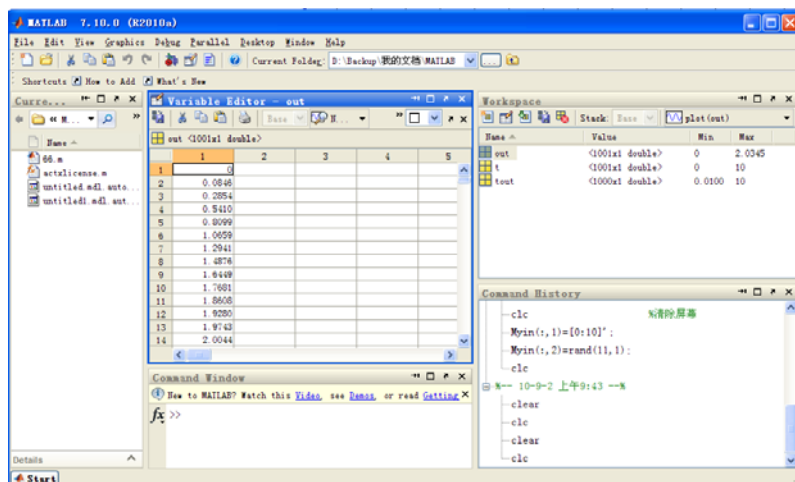


图 6-43 数据存储结果

【例 6-4】使用查找表模块。
首先建立如图 6-44 所示的模型并保存。

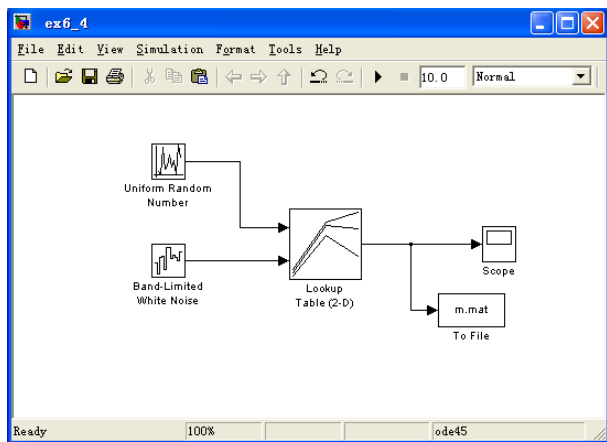


图 6-44 Simulink 模型

其中需要说明的是：

(1) Uniform Random Number 模块用于产生指定参数的均匀分布，Band-Limited White-Noise 模块用于产生有界白噪声，参数设置如图 6-45 所示。

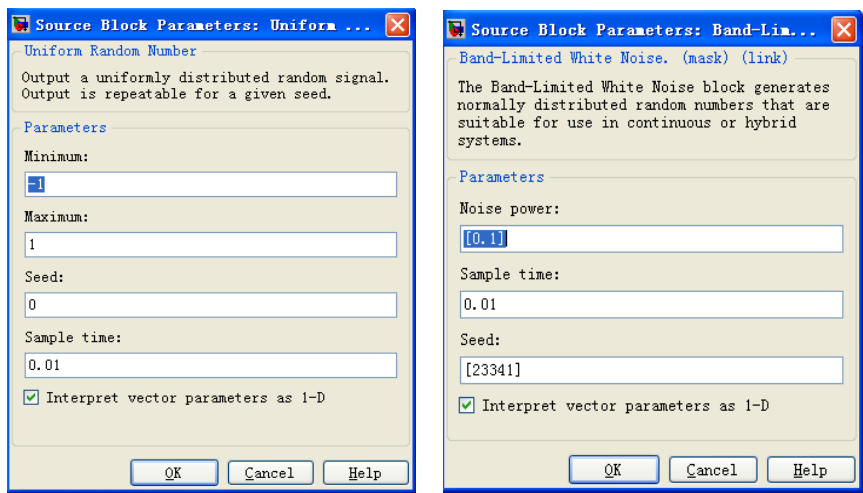


图 6-45 Uniform Random Number 模块和 Band-Limited White-Noise 模块设置

(2) Lookup Table 模块用于实现一维插值，参数设置如图 6-46 所示。其中，“Vector of input values”表示已知数据对 $(x,y=f(x))$ 中的 x ，“Table data”表示 y ，这里用变量 Table1data 表示，“Lookup method”表示插值方法。

(3) Lookup Table (2-D)模块用于实现二维插值，如图 6-47 所示的参数设置类似于一维插值。

(4) To file 模块用于实现将数据保存在 MAT 文件中，如图 6-48 所示设置保存的文件名和保存的变量名。

其次在命令窗口输入如下语句为变量 Table1data 赋值：

```
clear
clc
Table1data=cos(-5:5);
```

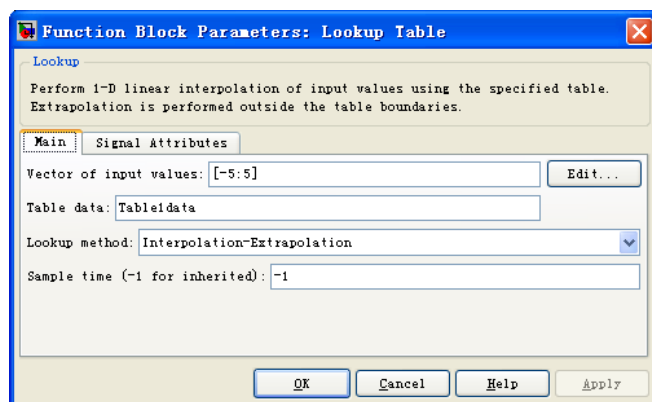


图 6-46 Lookup Table 模块设置

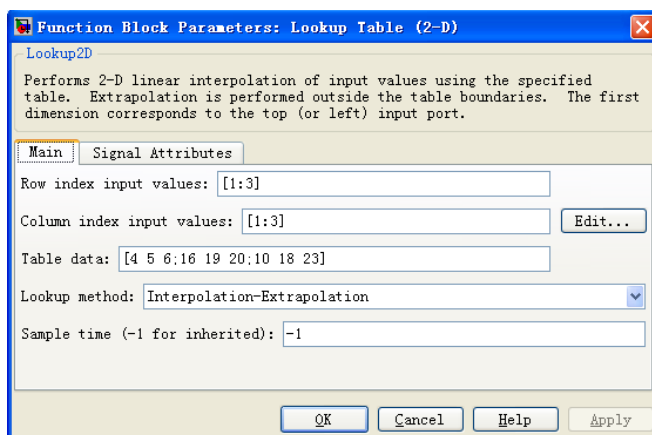


图 6-47 Lookup Table (2-D)模块设置

然后运行该模型，在图形窗口得到如图 6-49 所示的结果。

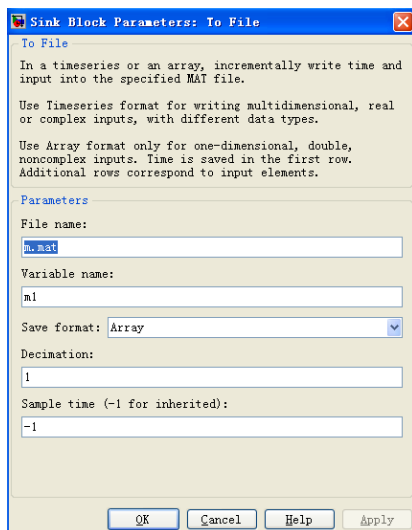


图 6-48 To File 模块设置

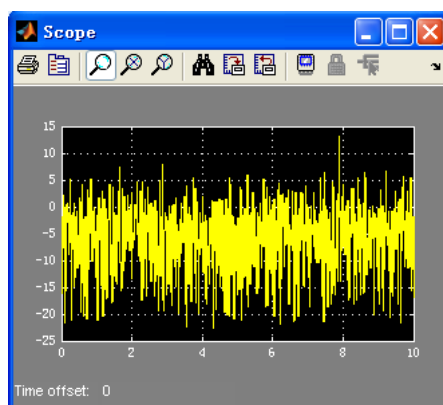


图 6-49 图形窗口结果

最后在命令窗口输入如下语句：

```
clear
clc
load m
sizedata=size(m1)
y=m1(:,1:5)
```

命令窗口中的输出结果如下所示：

```
sizedata =
         2         1001

y =
         0         0.0100         0.0200         0.0300         0.0400
-15.0156 -16.1355 -2.6512 -1.9151  2.1522
```

从结果不难看出，存放到 MAT 文件中的第一行为时间，第二行为数据。

6.3 Simulink 其他模块

Simulink 可以针对控制、信号处理以及通信等系统进行建模、仿真和分析，因此除了前面介绍的常用模块以外，还提供了其他的模块和工具箱。

Signal Processing Blockset（信号处理模块集）与 Video and Image Processing Blockset（视频和图像处理模块集）是 MATLAB 中执行流处理的新系统对象。系统对象支持 140 多种算法，占用的内存更少，可改善冗长信号和视频数据流的处理，简化流算法的开发。

【例 6-5】使用 Signal Processing Blockset 中的模块。

首先建立如图 6-50 所示的模型并保存。

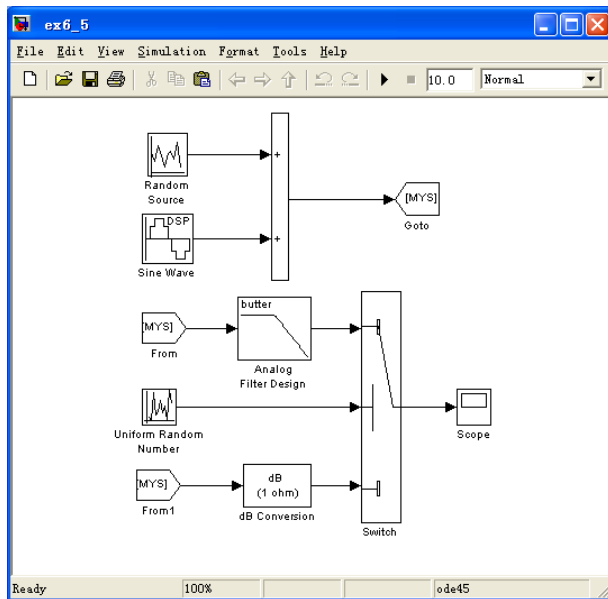


图 6-50 Simulink 模型

其中需要说明的是：

(1) Random Source 模块用于产生均匀和正态分布随机数，其参数设置如图 6-51 所示。

(2) Sine Wave 模块用于生成正弦函数，其参数设置如图 6-52 所示。

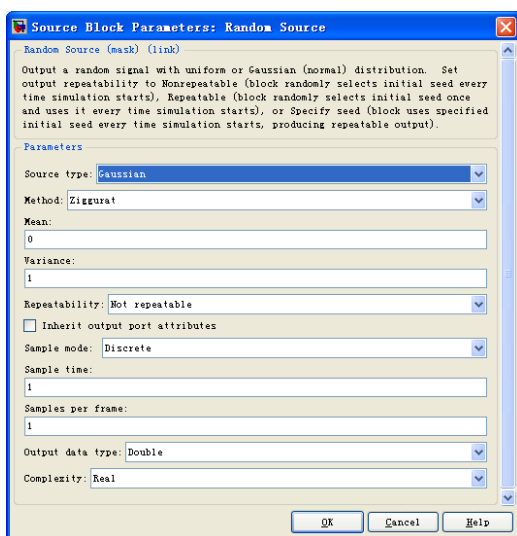


图 6-51 Random Source 模块设置

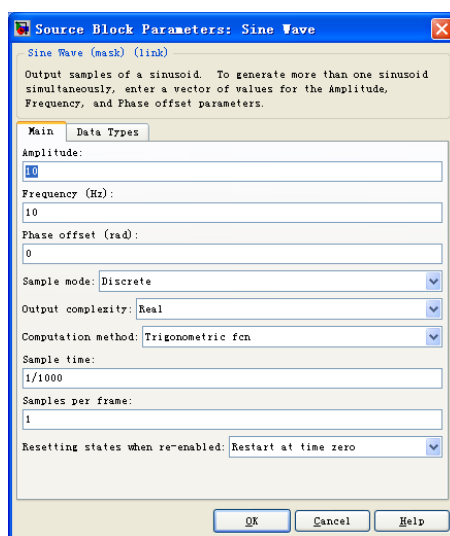


图 6-52 Sine Wave 模块设置

(3) Goto 模块用于在仿真过程中保存某个端口的输出，From 模块用于将保存的端口输出送入指定模块，对应的 Goto 模块和 From 模块应具有相同的参数 Tag，该参数可以如图 6-53 所示按照用户的需求设置。

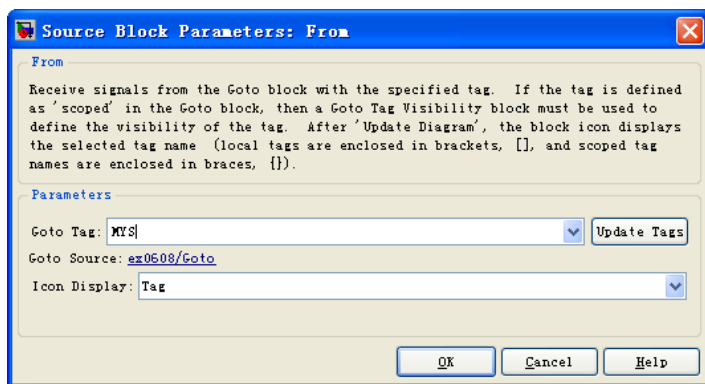
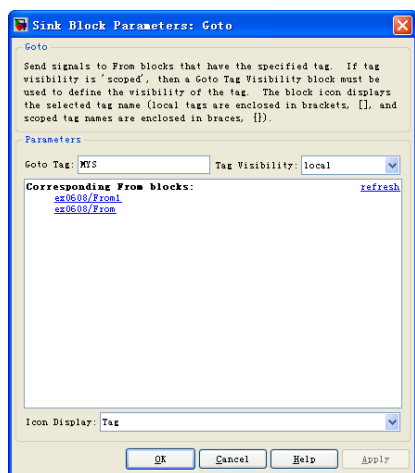


图 6-53 Goto 模块和 From 模块设置

(4) Analog Filter Design 模块用于设置滤波器形式和参数，可以选择低通、带通、高通等形式，还可以设置对应不同形式的参数，如图 6-54 所示。

(5) dB Conversion 模块用于计算分贝数。

(6) Switch 模块用于仿真的分支操作，当第二个输入不小于指定数（该值可以被设置，这里取为 0）时输出为第一个输入，否则为第三个输入。

其次运行该模型，然后双击 Scope 模块可得到如图 6-55 所示的结果。

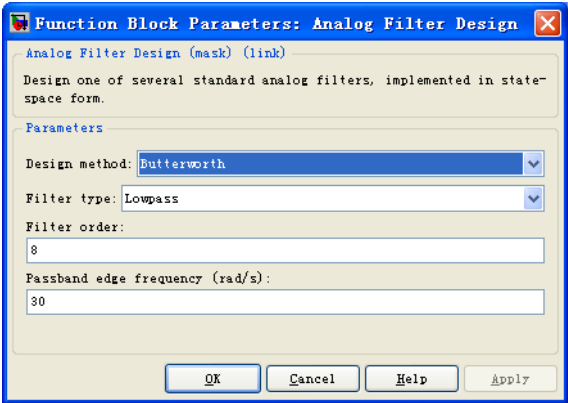


图 6-54 Analog Filter Design 模块设置

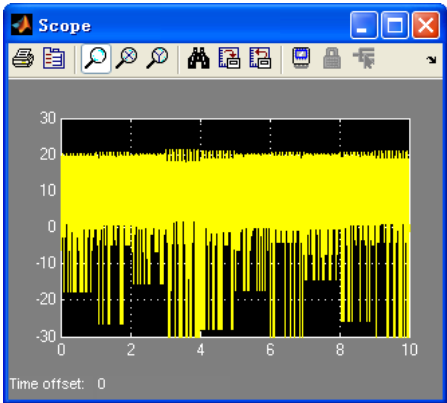


图 6-55 Scope 模块运行结果

6.4 Simulink 模型创建

通过前面的讲解我们对于 Simulink 模型已经有了一些初步的了解，简而言之，Simulink 模型有以下几层含义：

- (1) 在视觉上表现为直观的方框图。
- (2) 在文件上则是扩展名为.mdl 的 ASCII 代码。
- (3) 在数学上表现为一组微分方程或差分方程。
- (4) 在行为上则模拟了实际系统的动态特性。

简单模型的建立包括以下几个步骤：

- (1) 建立模型窗口。
- (2) 将功能模块由模块库复制到模型窗口。
- (3) 对模块进行连接，从而构成需要的系统模型。

建立 Simulink 模型的流程如图 6-56 所示。

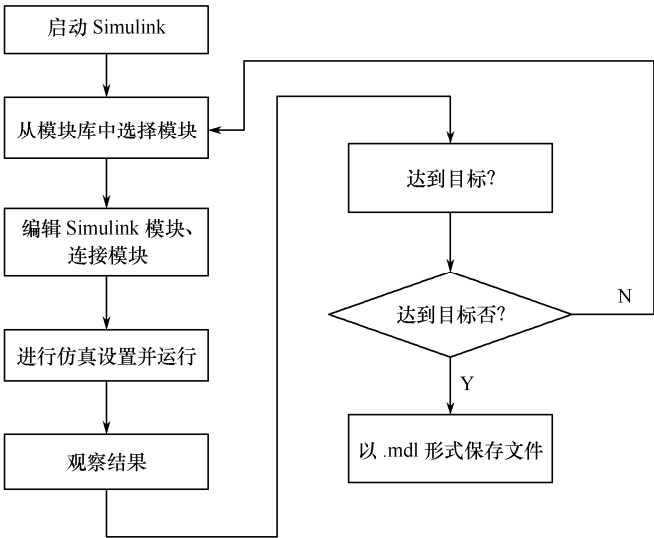


图 6-56 Simulink 模型建立流程

6.4.1 模块操作

Simulink 模块操作包括选择一个或多个模块，复制、删除和移动模块，模块外形的调整，模块名的操作，定义模块中的参数和属性，模块间的连接等。

1. 模块的选择

选择模块有两种情况，即选择一个模块和选择多个模块。

(1) 选择一个模块

选择一个模块只需要使用鼠标左键单击指定模块，当用户选中一个模块时，以前选中的模块就被放弃。

(2) 选择多个模块

选择多个模块有两种方法：一是逐个选择法，另一个是使用方框选择相邻的几个模块。

- 逐个选择法：按住“Shift”键，使用鼠标左键单击需要选中的模块。
- 方框选择法：使用鼠标单击和拖动画出方框，则选择方框内的所有模块。

2. 复制、删除和移动模块

模块复制、删除和移动操作如下所述。

(1) 复制模块

复制模块分两种情况：

- 在不同窗口间复制模块：选中模块后，直接将模块从一个窗口拖动到另一个窗口即可。
- 同一个窗口内复制模块：选中模块后，按下快捷键“Ctrl+C”实现复制，再按下快捷键“Ctrl+V”实现粘贴，还可以通过菜单“Edit”中的“Copy”和“Paste”命令来复制模块。

(2) 删除模块

删除模块有以下两种方法：

- 选中模块后，按“Delete”键删除模块。
- 选中模块后，通过菜单“Edit”中的“Cut”（将模块删除到剪贴板）和“Delete”（将模块永久删除）命令来删除模块。

(3) 移动模块

使用鼠标直接拖动模块到指定位置。

3. 模块外形的调整

模块外形的调整包括三种情形，即改变模块的大小、调整模块的方向和给模块添加阴影。

(1) 改变模块的大小

选中模块后，将鼠标移到模块边框的一角，当鼠标指针变成两端有箭头的线段时，按下鼠标左键拖动模块边框以改变模块大小。

(2) 调整模块的方向

选中模块后，选择“Format”→“Rotate Block”菜单命令使模块在水平方向顺时针旋转 90°，选择“Format”→“Flip Block”菜单命令使模块在垂直方向翻转 180°。

图 6-57 所示中间模块是最初模块，选择“Format”→“Rotate Block”→“Clockwise”和“Format”→“Rotate Block”→“Clockwise”菜单命令，最初的模块变为左边

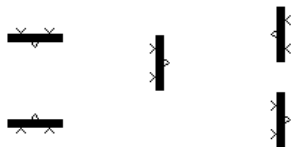


图 6-57 模块的旋转

的上下两个模块；右边的两个模块是中间的最初模块从上到下依次通过选择“Format”→“Flip Block”菜单命令旋转一次和两次后的结果。

(3) 给模块添加阴影

选中模块后，选择“Format”→“Show Drop Shadow”菜单命令给模块添加阴影。

4. 模块名的操作

模块名的操作包括修改模块名、显示模块名和改变模块名的位置。

(1) 修改模块名

使用鼠标左键双击模块名，即可修改模块名。

(2) 显示模块名

选中模块后，选择“Format”→“ShowName”菜单命令显示模块名；选择“Format”→“Hide Name”菜单命令隐藏模块名。

(3) 改变模块名的位置

选中模块后，选择“Format”→“Flip Name”菜单命令改变模块名的显示位置。

5. 定义模块中的参数

定义模块中的参数有以下三种方法：

- 用户通过双击需要设置参数的模块，得到如图 6-58 所示的模块参数设置对话框，定义模块中的参数。
- 右击模块并在弹出的菜单中选择“TransferFcn Parameters...”命令，定义模块中的参数。
- 选中要设置的模块后，选择“Edit”→“TransferFcn Parameters...”菜单命令，定义模块中的参数。

需要说明的是，每个模块的参数设置对话框都有对其自身功能的描述，如图 6-58 所示的上半部分。

6. 定义模块的属性

Simulink 中的每个模块都有一个内容相同的如图 6-59 所示的属性设置对话框，可以通过两种方法打开此属性设置对话框：

- 使用鼠标右键单击模块并在弹出的菜单中选择“Block Properties...”命令。
- 选中要设置的模块后，选择“Edit”→“Block Properties”菜单命令。

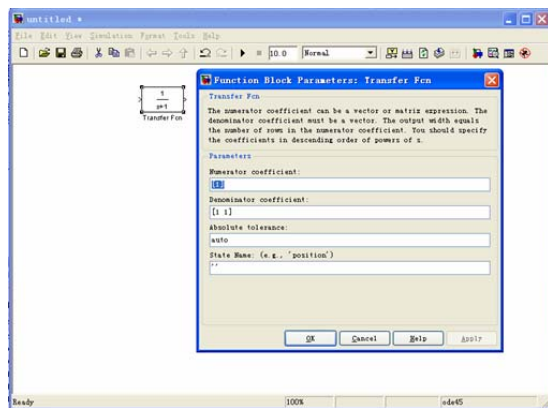


图 6-58 模块参数设置对话框

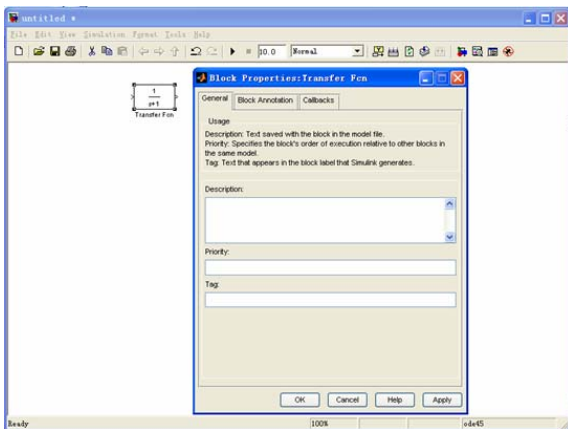


图 6-59 属性设置对话框

7. 模块的连接

模块之间的连接一般是通过直线完成的，下面通过表 6-17～表 6-19 介绍在 Microsoft Windows 环境下，对直线操作、直线信息和注释文字的处理。

表 6-17 直线操作

任务	Microsoft Windows 环境下的操作
选择多条直线	与选择多个模块的方法一样
选择一条直线	单击要选择的直线，当用户选择一条直线时，之前选择的直线被放弃
连线的分支	按下“Ctrl”键，然后拖动直线
移动直线	按下鼠标左键直接拖动直线
移动直线顶点	将鼠标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键并移动连线
直线调整为斜线段	按下“Shift”键，将鼠标指向需要移动的直线上的一点并按下鼠标左键直接拖动直线
直线调整为折线段	按住鼠标左键不放直接拖动直线

表 6-18 直线信息处理

任务	Microsoft Windows 环境下的操作
建立信号标签	在直线上双击，然后输入标签
复制信号标签	按下“Ctrl”键，然后按下鼠标左键选中标签并拖动
移动信号标签	按下鼠标左键选中标签并拖动
编辑信号标签	在标签框内双击，然后进行编辑
删除信号标签	按下“Shift”键，然后用鼠标选中标签，再按“Delete”键
用粗线表示向量	选择“Format”→“Port/Signal Displays”→“Wide Nonscalar Lines”菜单命令
显示数据类型	选择“Format”→“Port/Signal Displays”→“Port Data Types”菜单命令

表 6-19 注释文字处理

任务	Microsoft Windows 环境下的操作
建立注释	在模型图标中双击，然后输入文字
复制注释	按下“Ctrl”键，然后按下鼠标左键选中注释文字并拖动
移动注释	按下鼠标左键选中注释文字并拖动
编辑注释	单击注释文字，然后进行编辑
删除注释	按下“Shift”键，然后用鼠标选中注释文字，再按“Delete”键

6.4.2 基本步骤

通过前面内容的讲解，使用户了解到 Simulink 的一些基础知识，下面总结使用 Simulink 进行系统建模和仿真的步骤。

(1) 画出系统框图，将所要仿真的系统根据功能划分成子系统，然后选用模块来搭建每个子系统。

(2) 启动 Simulink 模块库浏览器，新建一个空白模型。

(3) 在模块库中找到所需模块并拖到空白模型窗口中，按系统框图的布局摆放好各模块并连接各模块。

(4) 如果系统较复杂、模块的数目太多，可以将同一功能的模块封装成一个子系统，子系统的封装将在第 6.5 节中详细介绍。

(5) 设置各模块的参数以及与仿真有关的参数，与仿真有关的参数设置将在第 6.6.2 小节中详细介绍。

(6) 将模型保存为后缀名为.mdl 的模型文件。

(7) 运行仿真, 观察结果。如果仿真出错, 请按弹出的错误提示来查看出错的原因, 进行修改; 如果仿真结果与预想的结果不符, 首先检查模块的连接是否有误, 选择的模块是否合适, 然后检查模块参数和仿真参数的设置是否合理。

(8) 调试模型。如果在第(7)步中没有出任何错误提示, 但是仿真结果与预想的结果不符, 那么就需要进行调试, 查看系统在每个采样点的运行情况, 以便找到导致仿真结果与预想的或实际情况不符的地方。修改后再仿真, 直到结果符合要求为止, 最后保存模型。调试模型的方法将在第 6.7 节中介绍。

6.5 子系统及其封装

下面分别介绍子系统和封装子系统的特点及其创建方法。

6.5.1 子系统的创建

随着系统规模的不断扩大, 复杂性不断增加, 模型的结构也变得越来越复杂, 用户很难轻易读懂以前所建的模型。在这种情况下, 将功能相关的模块组合在一起形成几个小系统, 将使整个模型变得非常简洁, 使用起来非常方便。

1. 子系统的优点

- (1) 减少模型窗口中模块的数目。
- (2) 把一些功能相关的模块集成在一起, 实现模块化。
- (3) 通过子系统可以实现模型图表的层次化。

2. 子系统的建立

在 Simulink 中创建子系统有以下两种方法:

- 通过子系统模块来创建子系统。
- 组合已存在的模块创建子系统。

通过子系统模块来创建子系统的步骤:

- (1) 打开 Simulink 模块库, 从 Ports&Subsystems 库中选取 Subsystem 模块拖到模型窗口中。
- (2) 双击 Subsystem 模块打开 Subsystem 模块的编辑窗口。
- (3) 直接在 Subsystem 模块的编辑窗口中建立模型, 构成子系统。

组合已存在的模块创建子系统的步骤:

- (1) 使用方框同时选中要组合的模块。
- (2) 选择“Edit”→“Creat Subsystem”菜单命令创建子系统。

【例 6-6】使用第二种方法创建 PI 控制器的子系统。

首先如图 6-60 所示同时选中要组合的模块, 然后选择“Edit”→“Creat Subsystem”菜单命令, 创建如图 6-61 所示的子系统, 双击生成的子系统, 如图 6-62 所示。

6.5.2 子系统的条件执行

1. 条件执行子系统的定义

在 Simulink 模块库里, 有两个特殊的模块: Enable 模块和 Trigger 模块, 如图 6-63 所示。如果把上述这两种模块放到某一个子系统中, 则该子系统是否起作用将取决于外界的某个条件(状态或事件)是否满足, 这样就构成了所谓的条件执行子系统。

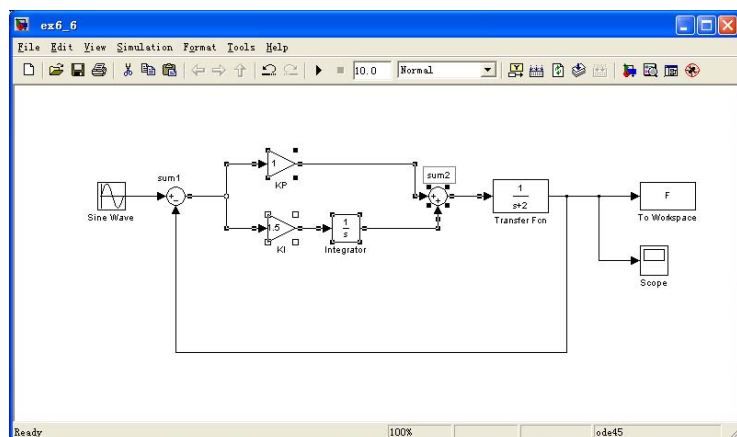


图 6-60 选择子系统包含的模块

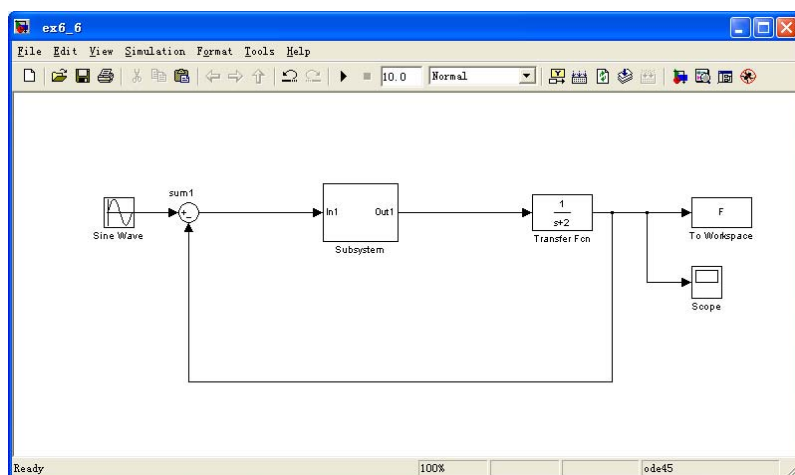


图 6-61 生成子系统后的模型图

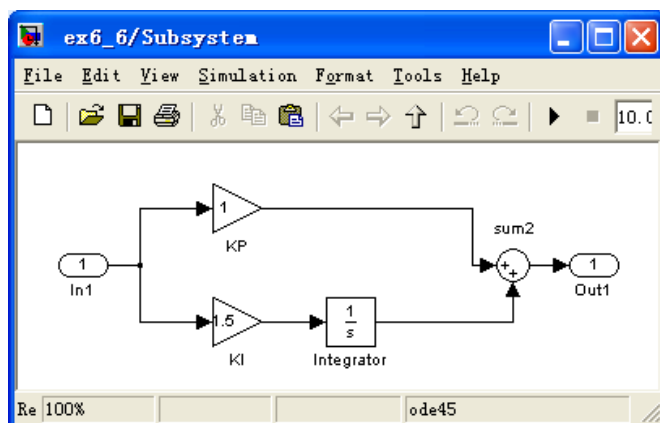


图 6-62 子系统模型

2. 条件执行子系统的分类

常用的条件执行子系统有：

- 使能子系统 (Enabled Subsystem)
- 触发子系统 (Triggered Subsystem)



Enable



Trigger

图 6-63 Enable 模块和 Trigger 模块

- 触发使能子系统 (Triggered and Enabled Subsystem)

【例 6-7】利用使能原理构成一个半波整流器，演示使能子系统的创建及其工作机理。

(1) 打开 Simulink 的新建模型窗口。

(2) 从 Simulink 模块库中提取三个模型 Sine Wave、Subsystem、Scope 到新建模型窗口。

然后进行文件保存操作，文件名为 ex6_7.mdl (为方便以后调用)。

(3) 双击空子系统模块 Enabled Subsystem，打开其结构模型窗口。Enable 模块无须进行任何连接，本例采用默认设置，便可实现题目所需使能子系统，如图 6-64 所示。

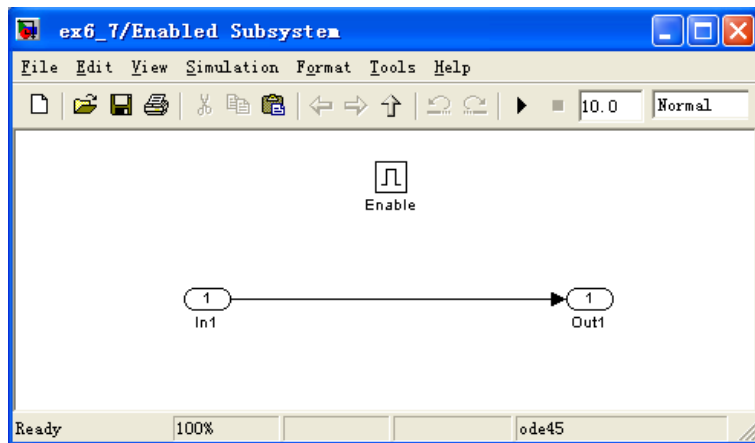


图 6-64 Enabled Subsystem 模块

(4) 完成各模块之间的连接，如图 6-65 所示。

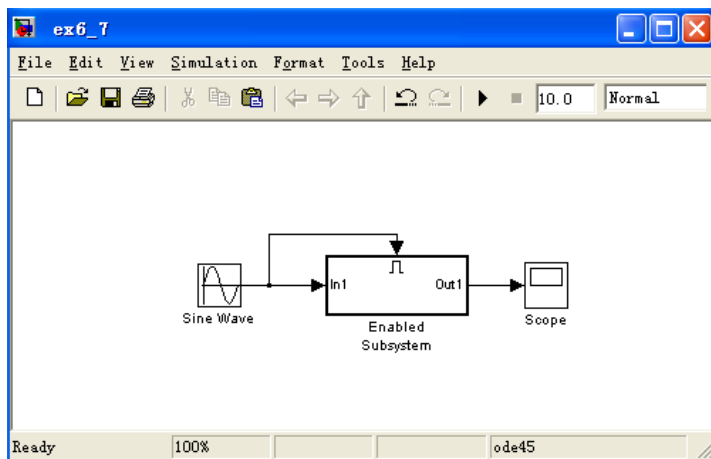


图 6-65 半波整流器模型

(5) 单击“Start”按钮，双击示波器模块，打开显示窗口，就可看到半波整流后的波形，如图 6-66 所示。

6.5.3 子系统的封装

子系统虽然可以使模型更简洁，但是更改子系统内部模块参数时，需要打开许多参数对话框，修改大量的参数时工作会变得十分烦琐，Simulink 提供了封装技术来解决这一问题。封装就是为子系统定制对话框和图标，使子系统具有良好的用户界面。

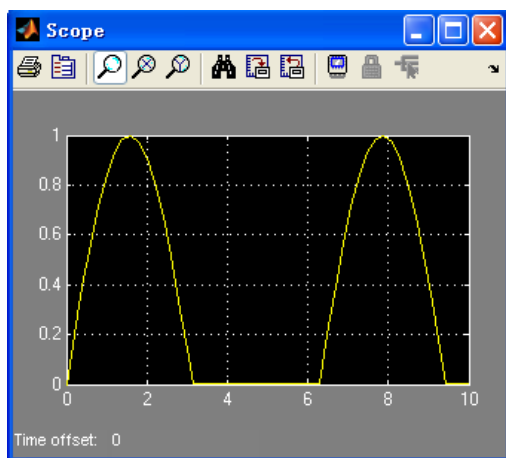


图 6-66 示波器显示结果

1. 封装子系统的优点

- (1) 在设置子系统内部模块参数时可以通过一个参数对话框完成。
- (2) 为子系统创建一个可以反映子系统功能的图标。
- (3) 可以避免用户在无意中修改子系统中模块的参数。

2. 封装子系统的建立

封装子系统的建立分为如下三个步骤：

- (1) 选中需要封装的子系统。
- (2) 选择“Edit”→“Edit mask”菜单命令，弹出如图 6-67 所示的封装编辑器，设置 Icon、Parameters、Initialization 和 Documentation。
- (3) 单击“Apply”或“OK”按钮保存设置。

3. 封装编辑器的设置

以上一小节中创建的子系统为例，介绍一下封装编辑器的设置。

- “Icon”页：图标设置页面如图 6-67 所示，用于设置显示的文字、图形和转换函数。

如果需要在例 6-6 的子系统显示文字“PI controller”，可如图 6-68 所示在编辑框“Icon Drawing commands”中输入相应文字，设置后系统如图 6-69 所示。

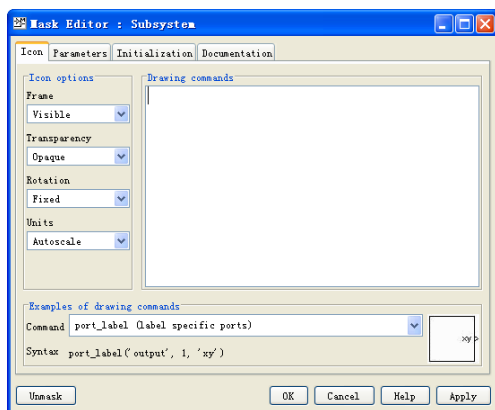


图 6-67 封装编辑器

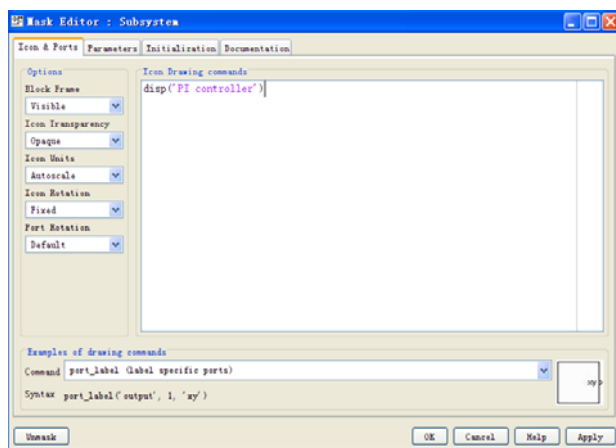


图 6-68 图标设置页面

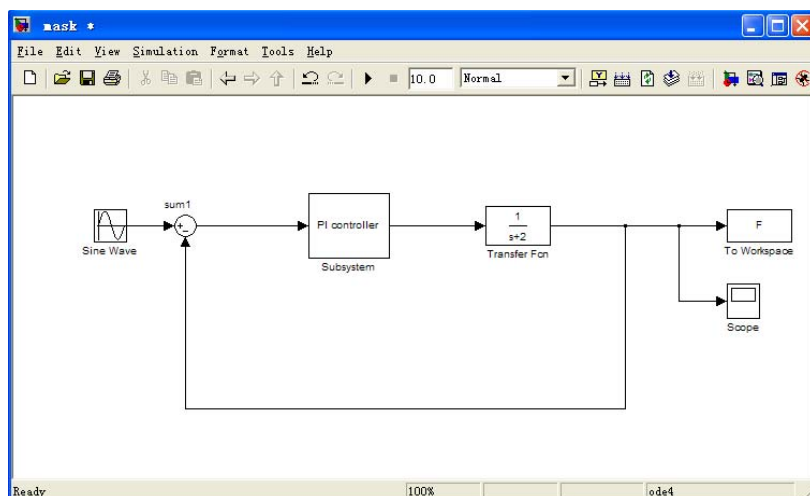


图 6-69 Simulink 模型

- “Parameters” 页：参数设置页面如图 6-70 所示，用于定义参数的提示符和变量名等。

✚图标：实现在“Dialog parameters”中添加变量。在编辑框“Dialog parameters”中，“Prompt”用来描述参数的文本标志；“Variable”用来存储参数值的变量名；“Type”用来选择用户的控制风格；选中“Evaluate”表示用户输入的内容先经 MATLAB 进行计算，再将输出结果赋给相关变量，如果不选中的话，用户输入的内容将不经过计算，直接以字符串的格式赋给相关变量；选中“Tunable”允许输入值在仿真过程中发生变化；在“Tab name”框内可以设置变量的标号。

✕图标：实现删除已经添加的变量。

⬆图标：实现将变量位置上移。

⬇图标：实现将变量位置下移。

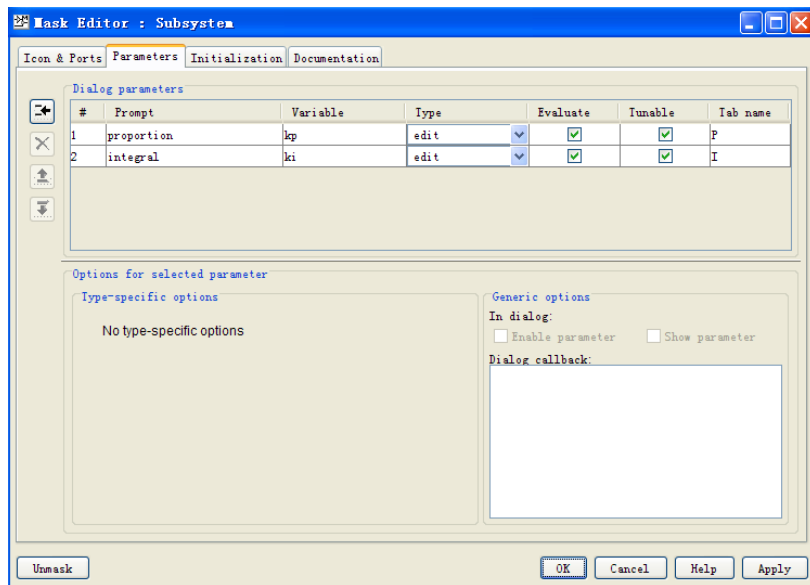


图 6-70 参数设置页面

此时将子系统修改为如图 6-71 所示的形式，这样就将子系统内部模块参数与参数设置页面中的参数连接在了一起。

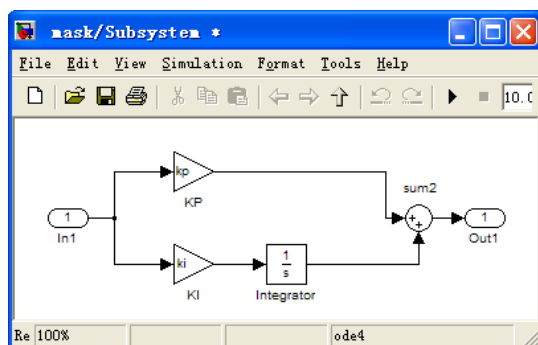


图 6-71 修改后的子系统

双击上述修改后的子系统，出现如图 6-72 所示的参数设置对话框。单击“P”设置比例系数，单击“I”设置积分系数。

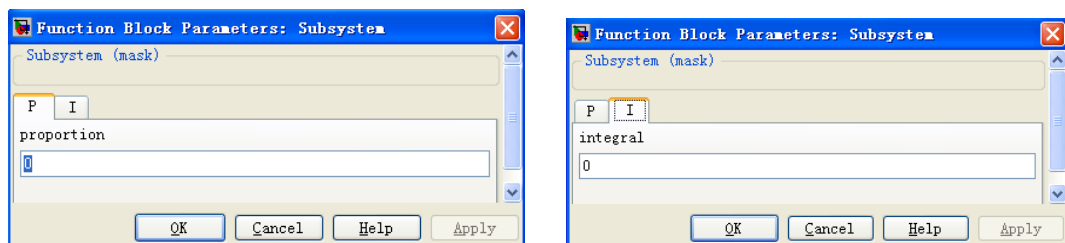


图 6-72 参数设置对话框

- “Initialization”页：初始化设置页面如图 6-73 所示。在“Initialization commands”编辑框中可以输入初始化命令，这些命令会在开始仿真、更新模块框图、载入模型和重新绘制封装子系统的图标时被调用。

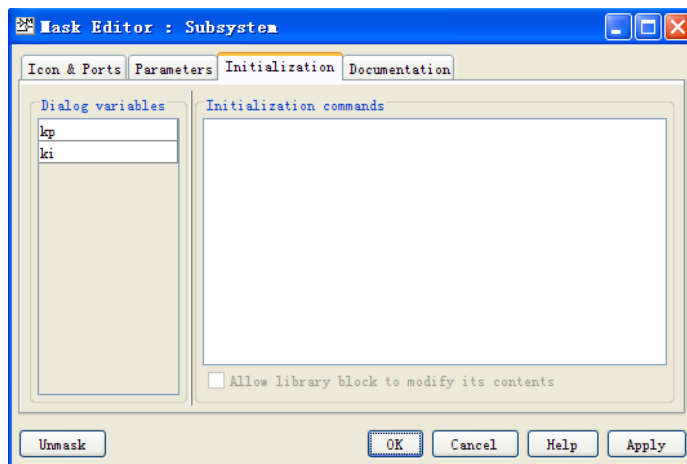


图 6-73 初始化设置页面

- “Documentation”页：封装类型、模块描述和帮助文本设置页面如图 6-74 所示。“Mask type”编辑框用于设置模块的封装类型，可以输入字符串，它的作用是：和 Simulink 内置的模块区分开，它显示在封装模块对话框的顶部；“Mask description”编辑框用于描述模块功能，输入的内容显示在封装模块对话框的上部；“Mask help”编辑框用于输入帮助文本，当单击“Help”按钮时将显示这些内容。

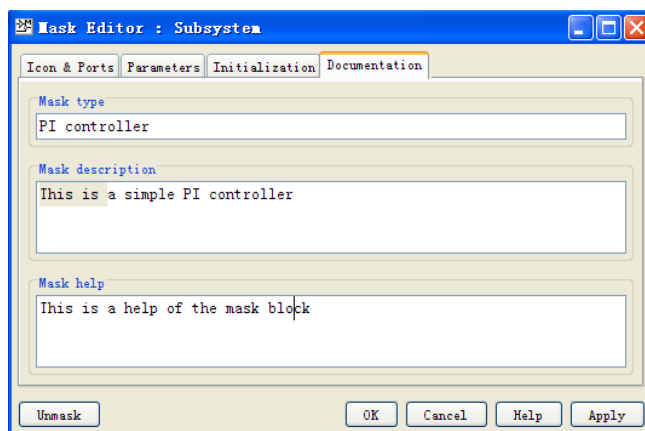


图 6-74 “Documentation” 页面

如上设置完毕后，双击子系统出现如图 6-75 所示的参数对话框，再单击“Help”按钮得到如图 6-76 所示的帮助信息。

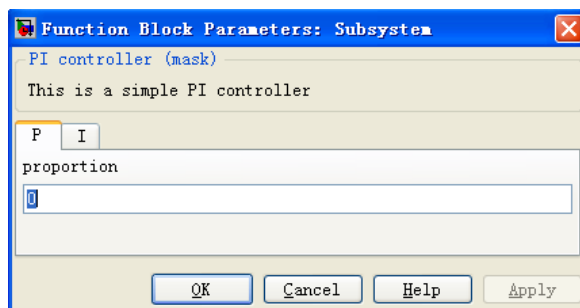


图 6-75 子系统参数对话框

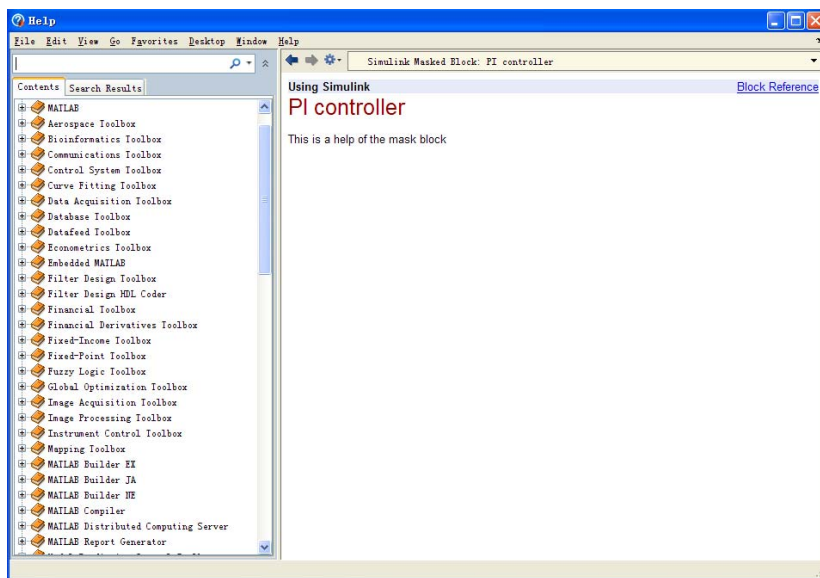


图 6-76 子系统的帮助

6.6 运行仿真

建立模型之后需要运行仿真模型，本节将详细介绍各种仿真参数的设置和仿真的运行。在介绍仿真运行之前先介绍两个重要的概念：一个是过零检测，另一个是代数环。

6.6.1 过零检测和代数环

下面分别介绍过零检测和代数环的含义。

1. 过零检测

动态系统仿真过程中，状态的不连续常常会导致动态系统的显著变化，因此对不连续点进行精确的仿真非常重要，否则会导致仿真得到错误的系统行为。为了避免得到错误的结论，需要使不连续点发生的时刻成为仿真的一个采样点。

Simulink 在每一个采样点使用过零检测技术来检测系统状态变量的间断点。如果 Simulink 在当前采样点内检测到了不连续的点，那么它将找到发生不连续的精确时间点，并且会在该时间点的前后附加采样点。

2. 代数环

使用 Simulink 模块库建立动态系统模型时，有些系统模块的输入端口具有直接馈通的特性，也就是说模块输出直接依赖于模块的输入。

系统模型中代数环的产生条件如下：

- 具有直接馈通特性的系统模块的输入，直接由此模块的输出来驱动。
- 具有直接馈通特性的系统模块的输入，由其他直接馈通模块所构成的反馈回路间接来驱动。

系统模型中代数环的消除方法如下：

- 对于含 Atomic Subsystem 和 Enabled Subsystem 模块的模型，Simulink 可以在模块参数设置对话框中选择“Minimize algebraic loop occurrences”来消除其中一些代数环。
- 对于含 Model 模块的模型，Simulink 可在“Configuration Parameters”对话框的“Model Referencing 面板”中选择“Minimize algebraic loop occurrences”来消除其中一些代数环。
- 还可尝试在具有直接馈通特性系统模块的输出部分增加延迟环节。

6.6.2 仿真参数的设置

当系统开始仿真时，Simulink 就开始按照指定设置来求解方程，如果用户不希望采用系统的默认参数，就要在系统仿真之前设置各种仿真参数。

Simulink 中需要设置的参数包括：起始时间和终止时间、仿真的步长、仿真容差和仿真使用的算法等，用户还可以设置系统是否从外界获得数据、是否向外界传递数据等。

Simulink 的参数设置都可以在仿真参数对话框中完成，打开仿真参数对话框有以下两种方法：

- 单击“Simulation”→“Configuration Parameters”菜单命令，可以得到如图 6-77 所示的仿真参数对话框。
- 直接按“Ctrl+E”快捷键，也可以打开仿真参数对话框。

下面主要介绍仿真器参数设置、仿真数据导入导出设置和仿真诊断设置。

1. 仿真器参数设置

单击仿真参数对话框左面的“Slover”，打开如图 6-78 所示的仿真器参数设置窗口。

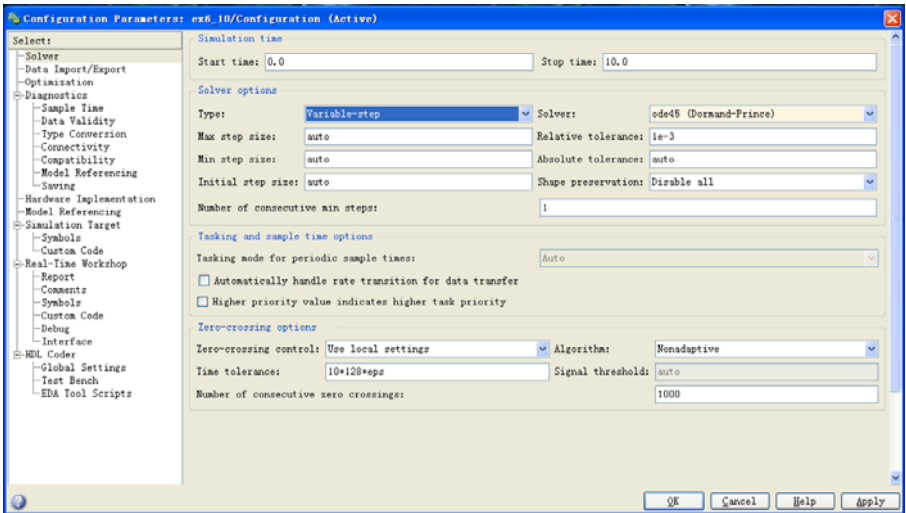


图 6-77 仿真参数对话框

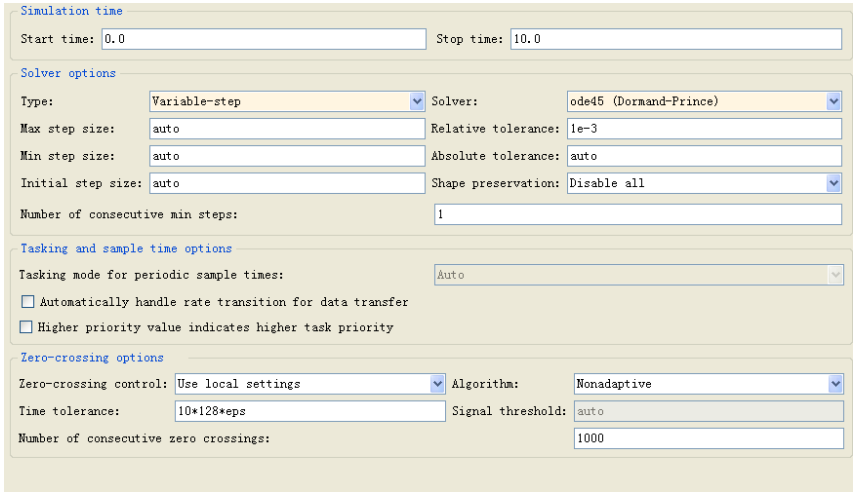


图 6-78 仿真器参数设置窗口

仿真器参数设置窗口可以用于设置仿真开始的时间、仿真结束的时间、选择解法器并设置它的相关参数。

(1) 仿真时间

在仿真器参数设置窗口的“Start time”中设置仿真的起始时间（默认为 0），在“Stop time”中设置仿真的终止时间（默认为 10 秒）。

计算机中的时间与真实的时间概念不同，计算机中的时间与采样步长有关，采样步长越小，采样点数就会增加，实际执行的时间也会增加。

(2) 仿真步长

在仿真器参数设置窗口的“Type”下拉菜单中指定步长方式，如图 6-79 所示。

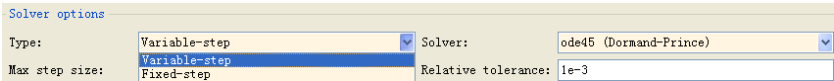


图 6-79 指定步长方式

• **Variable-step**（变步长模式）：可以在仿真过程中改变步长，提供误差控制和过零检测选择。变步长模式下的可选解法器如图 6-80 所示，它们的具体含义如表 6-20 所示。

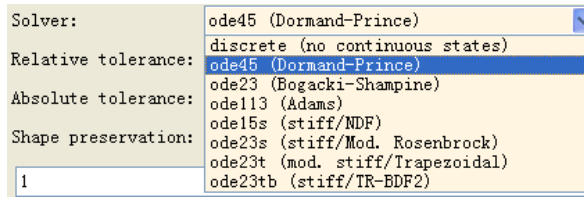


图 6-80 可选解法器

表 6-20 变步长模式解法器

名称	功能介绍
discrete	模型在离散无连续状态时使用
ode45	默认值，4/5 阶龙格-库塔法，应用在大多连续和离散系统，不适于刚性系统
ode23	2/3 阶龙格-库塔法，适用于精度要求不高的场合，不适于刚性系统
ode113	阶数可变的解法器，多步解法器，不适于刚性系统，
ode15s	基于数字微分公式的解法器，适用于刚性系统
ode23s	单步解法器，应用于刚性系统
ode23t	实现梯形规则的一种自由差值，应用于适度刚性系统
ode23tb	表示两阶隐式龙格-库塔公式，应用于刚性系统

• **Fixed-step**（固定步长模式）：在仿真的过程中采用固定的步长，不提供误差控制和过零检测选择。固定步长模式下的可选解法器如图 6-81 所示，它们的具体含义如表 6-21 所示。

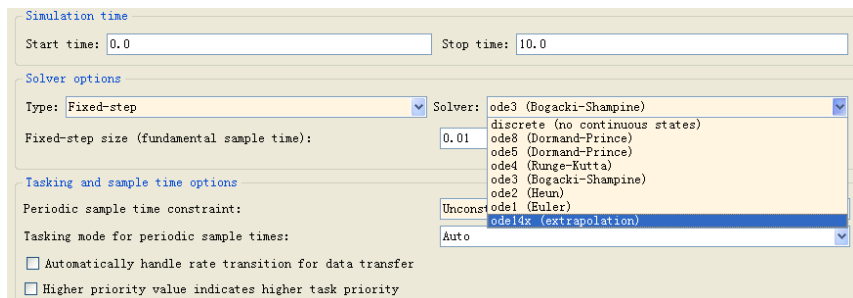


图 6-81 可选解法器

表 6-21 固定步长模式解法器

名称	功能介绍
discrete	模型在离散无连续状态时使用
ode5	默认值，4/5 阶龙格-库塔法，应用在大多连续和离散系统，不适于刚性系统
ode4	4 阶龙格-库塔法，具有一定计算精度
ode3	2/3 阶龙格-库塔法
ode2	改进的欧拉法
ode1	欧拉法
ode14x	支持物理模块仿真

固定步长和变步长两种解法器计算下一个仿真时间的方法都是在当前仿真时间上加上一个时间步长。不同的是固定步长解法器的时间步长是常数，而可变步长解法器的时间步长是根据模型动态特性可变的。当模型的状态变化特别快时，为了保证精度要降低时间步长，反之就增加时间步长。

当用户希望通过自建模型生成代码并在实时计算系统中运行这些代码时，用户就应该选择固定步长解法器来仿真模型。这是因为实时计算系统以固定的采样速率运行，若采用变步长将有可能使仿真器发生错误。

当用户并不从模型生成代码时，解法器类型的选择取决于模型的动态特性。当模型的状态变化特别快或是包括不连续状态时，选择变步长解法器可以缩短仿真时间。这是因为变步长相对于固定步长的解法器需要更少的时间，而两者仿真的精度相当。

(3) 步长参数

在变步长模式下，用户可以设置最大的和推荐的初始步长参数，默认值是 `auto`。

- **Maximum step size**（最大步长参数）：决定解法器能够使用的最大时间步长。
- **Initial step size**（初始步长参数）：建议使用默认值 `auto`。

在固定步长模式下，用户可以设置步长。

(4) 仿真精度

在变步长模式下，仿真精度有两种：

- **Relative tolerance**（相对误差）：指误差相对于状态的值。
- **Absolute tolerance**（绝对误差）：在状态值为零的情况下，可以接受的误差。

2. 仿真数据导入导出设置

单击仿真参数对话框左面的“Data Import/Export”，打开如图 6-82 所示的仿真数据导入导出设置窗口。该窗口主要用于向 MATLAB 工作空间输出模型仿真结果，或从 MATLAB 工作空间读入数据到模型。

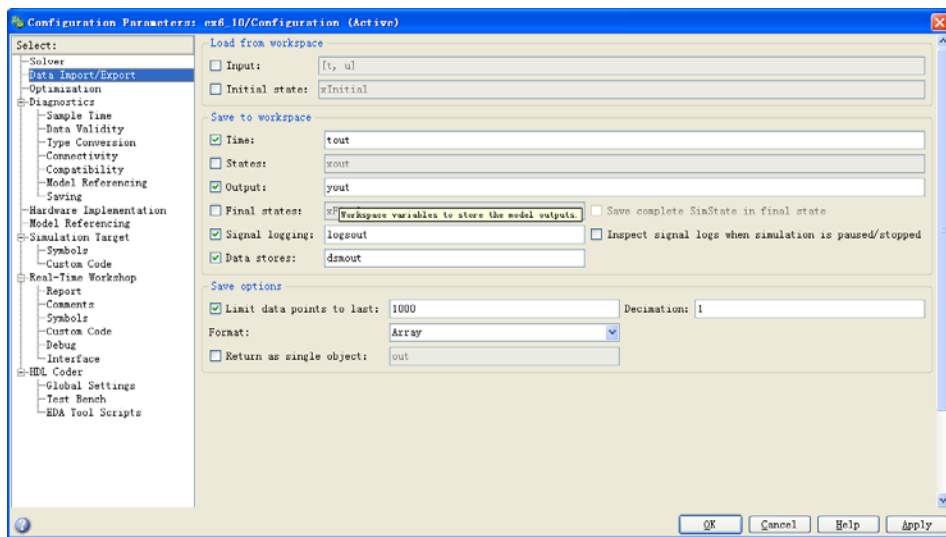


图 6-82 仿真数据导入导出设置窗口

仿真数据导入导出设置窗口包括以下三部分：

- **Load from workspace**：从 MATLAB 工作空间向模型导入数据，作为输入和系统的初始状态。
- **Save to workspace**：向 MATLAB 工作空间输出仿真时间、系统状态和系统输出等。
- **Save options**：向 MATLAB 工作空间输出数据的数据格式、数据量以及生成附加输出信号数据等。

3. 仿真诊断设置

单击仿真参数对话框左面的“Diagnostics”，打开如图 6-83 所示的仿真诊断设置窗口。

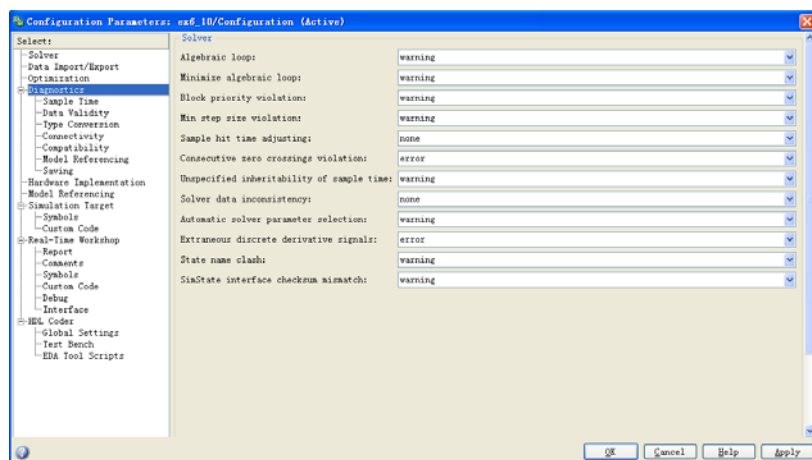


图 6-83 仿真诊断设置窗口

仿真诊断设置窗口包括对采样时间、数据完整性、转换、连接、兼容性和模型引用这几项的诊断。用户可以设置当模块在编译和仿真遇到异常时，Simulink 将采用哪种诊断动作，如是否进行一次检验、是否禁用过零检测、是否禁用复用缓存和是否进行不同版本的 Simulink 检验等。

6.6.3 仿真的运行

仿真的运行可以采用两种方式：一种是使用窗口运行仿真，另一种是使用 MATLAB 命令运行仿真。

1. 使用窗口运行仿真

建立模型后，如图 6-84 所示可以通过选择“Simulation”→“Start”菜单命令进行仿真，或如图 6-85 所示可以通过单击工具栏上的▶按钮进行仿真。

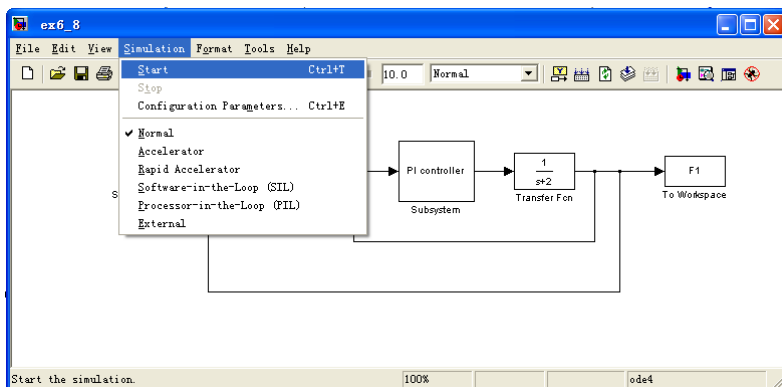


图 6-84 通过菜单进行仿真

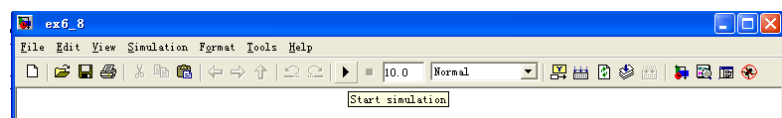


图 6-85 通过工具栏进行仿真

2. 使用 MATLAB 命令运行仿真

MATLAB 提供了 `sim` 函数在命令窗口、脚本中运行仿真模型，它的具体用法如下所示：

```
sim('model')
```

```
[T,X,Y] = sim('model',TIMESPAN,OPTIONS,UT)
```

```
[T,X,Y1,...,Yn] = sim('model',TIMESPAN,OPTIONS,UT)
```

其中，`model` 为要运行的模型文件名，`T` 为返回的时间向量，`X` 为返回的状态量，`Y` 为返回的输出量（必须对应位于主仿真窗口的输出模块），`Y1,...,Yn` 为返回的指定输出量（必须对应位于主仿真窗口的输出模块），`TIMESPAN` 为设置的运行时间段，`OPTIONS` 为设置的仿真参数（可以由 `simset` 函数创建），`UT` 为可选的外部输入。

MATLAB 可以通过基本工作空间和 Simulink 进行数据交互。下面通过例子介绍如何在脚本中运行模型，并与模型进行数据交互。

【例 6-8】在脚本中运行模型，并与模型进行数据交互，具体步骤如下：

首先建立如图 6-86 所示的模型并保存为 `ex6_8`，其中外环的子系统是前面介绍的 PI 控制器子系统，并且它的 `Gain` 模块参数由上至下分别设置为变量 `kp1` 和 `ki1`；内环的子系统也是按照外环的 PI 控制器封装而成的，并且它的两个参数由上至下分别设置为变量 `kp2` 和 `ki2`；`Transfer Fcn` 模块的分母参数设置为变量 `K`。

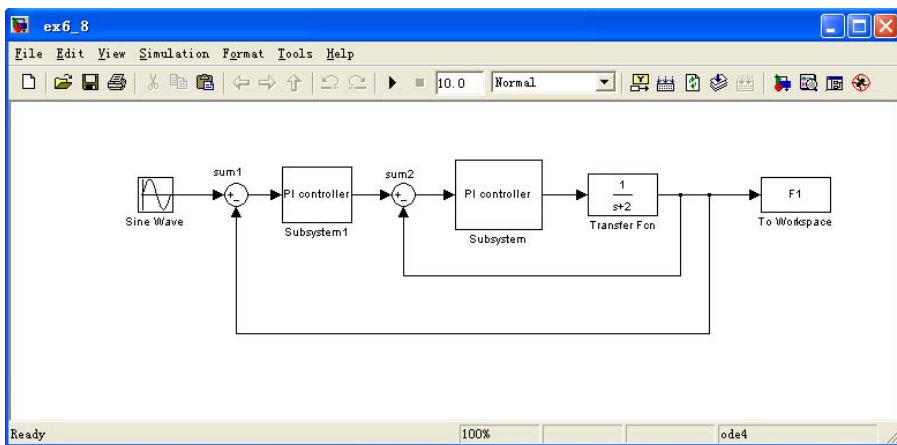


图 6-86 Simulink 模型

编写如下内容的脚本并保存：

```
clear
clc
kp1=0.5;
ki1=0.4;
kp2=0.3;
ki2=0.2;
K=3;
sim('ex6_10')
maxF1=max(F1)
minF1=min(F1)
```

最后运行脚本，命令窗口中的输出结果如下所示：

```
maxF1 =  
    0.3668  
minF1 =  
   -0.1488
```

单击 F1，数据存储如图 6-87 所示。

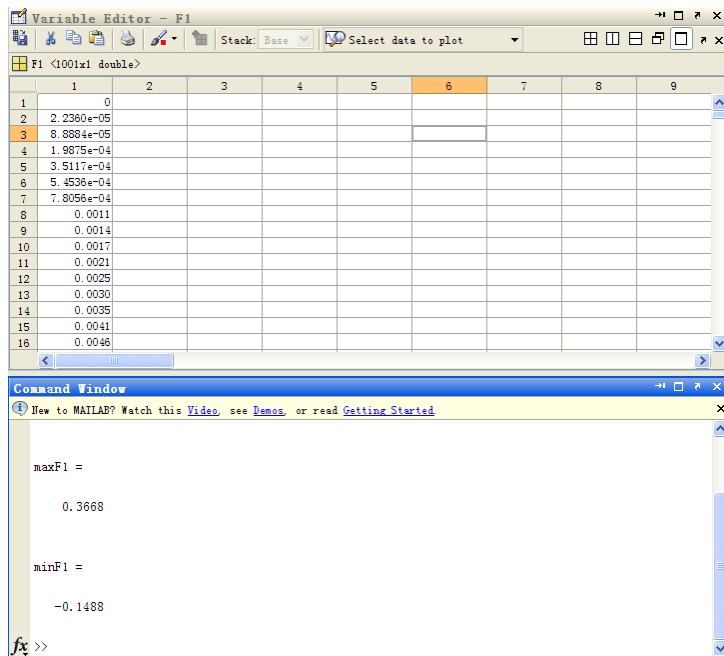


图 6-87 数据存储结果

由上面的例子可以看出，脚本可以与模型很方便地进行数据交互，包括模块、子系统、封装子系统、仿真参数等。

6.7 模型调试

Simulink 作为高性能的系统建模、仿真与分析平台，为用户提供了强大的模型调试工具。模型的调试可以通过两种方式来原因：一是 Simulink 调试器，二是命令行。由于经常使用 Simulink 调试器进行调试，下面以例 6-8 为基础，说明使用 Simulink 调试器进行调试的步骤。

(1) 另存例 6-8 对应的模型文件为 test.mdl，并修改仿真运行参数为 0.2 秒的固定步长，仿真时间为 20 秒，如图 6-88 所示。

(2) 单击“Tools”→“Simulink Debugger...”菜单命令，启动如图 6-89 所示的 Simulink 调试器。

Simulink 调试器可以分为两个部分：

- 调试器工具栏，表 6-22 介绍了 Simulink 调试器的工具栏按钮及其功能。
- 调试器窗口界面。

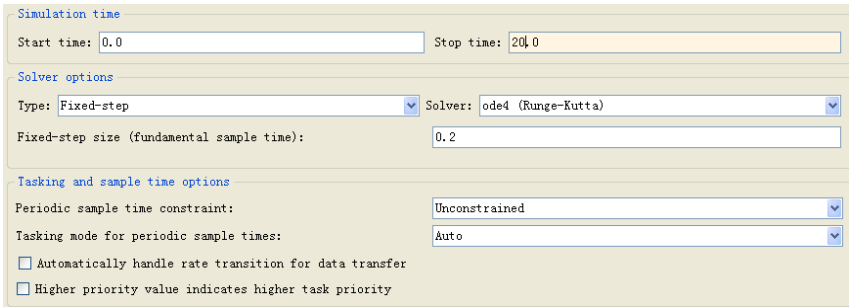


图 6-88 仿真参数设置

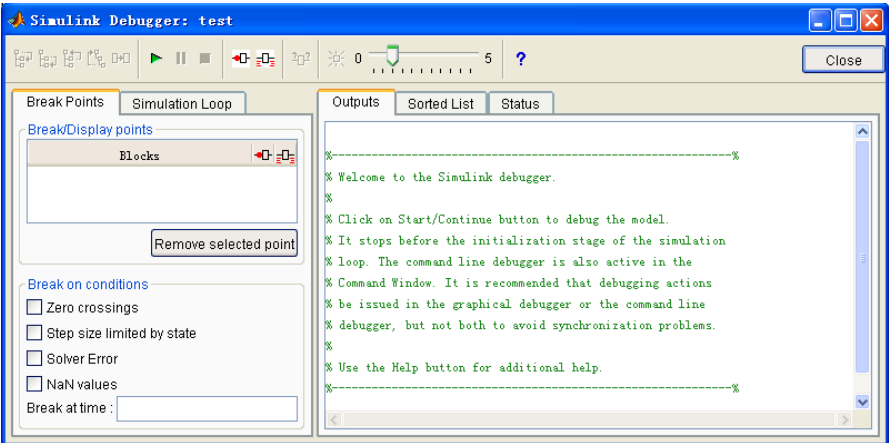


图 6-89 Simulink 调试器

表 6-22 调试器工具栏

工具栏按钮	功能
	进入当前方法
	跳过当前方法
	跳出当前方法
	在下一个采样点跳转到第一个方法
	跳转到下一个模块方法
	开始或继续调试
	暂停仿真
	停止仿真
	在指定模块之前设置断点
	当选中的模块被执行时显示其输入输出
	显示选中的模块的当前输入输出
	选择动画模式
	显示调试器的帮助
	关闭调试器

➤ “Break Points” 页：用于设置断点，即仿真运行到某个模块方法或满足某个条件时停止。断点显示框如图 6-90 所示，它用于显示断点的设置以及断点处模块的输入输出值。大多数情况下，用户需要在一定的条件下设置系统断点来进行系统调试，Simulink 提供了如图 6-91 所示的五种断点条件设置，即“Zero crossings”在系统发生过零处设置断点；“Step size limited by state”在仿真步长受到状态限制处设置断点；“Solver Error”在系统遇到错误处设置断点；“NaN values”在系统中出现无穷小值处设置断点；“Break at time”在指定的仿真时刻处设置断点。

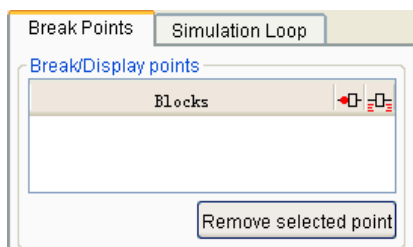


图 6-90 断点显示框

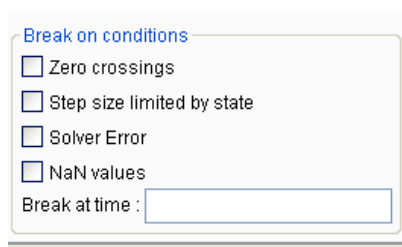




图 6-91 断点条件设置框

- “Simulation Loop” 页：用于显示当前仿真步正在运行的相关信息。
- “Outputs” 页：用于显示调试结果，包括调试命令提示、当前运行模块的输入输出和模块的状态。
- “Sorted List” 页：用于显示被调试的模块列表，该列表按模块执行的顺序排列。
- “Status” 页：用于显示调试器各种选项设置的值以及其他状态信息。

(3) 设置断点，在 test.mdl 文件中选中 Transfer Fcn 模块，即所要设置断点的模块，然后单击 Simulink 调试器工具栏中的 （在指定模块之前设置断点）按钮进行断点设置，并单击 （当选中的模块被执行时显示其输入输出）按钮，切换到如图 6-92 所示的界面。

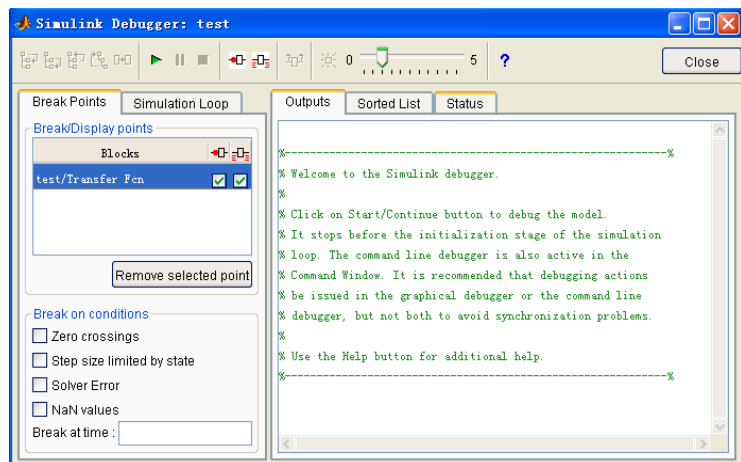

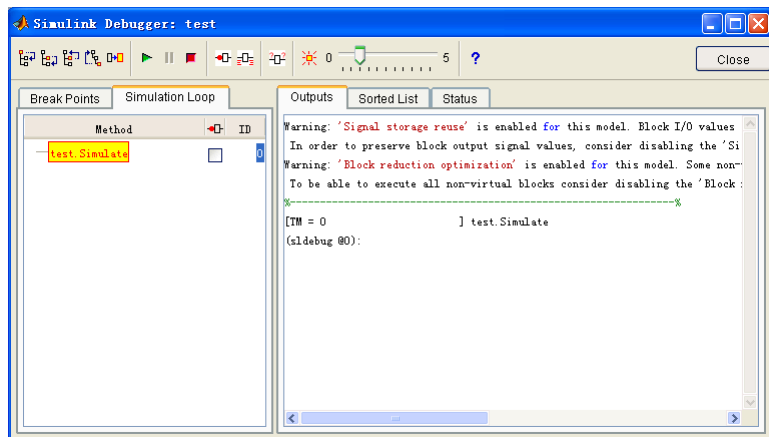


图 6-92 设置断点

(4) 单击  按钮开始进行调试，切换到如图 6-93 所示的界面。



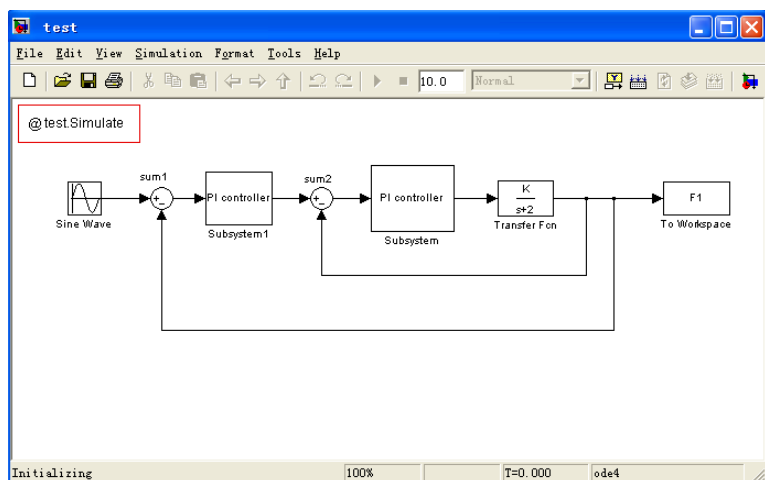



图 6-93 进行调试

(5) 单击  按钮进行单个模块仿真，不断地单击可以看到 Simulink 正执行模块及设置断点处的输入输出，如图 6-94 所示。由执行模块，可以明显地看出 Simulink 调用模块的次序。

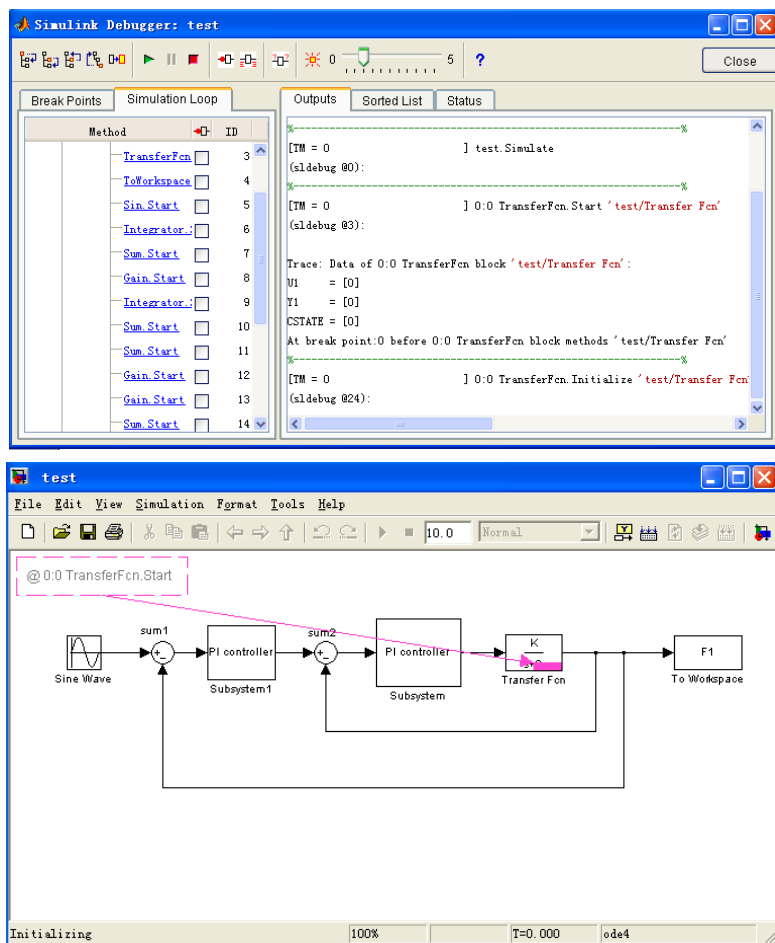


图 6-94 进行仿真

(6) 单击  按钮将再次运行到断点处暂停，单击  按钮退出调试，如图 6-95 所示。

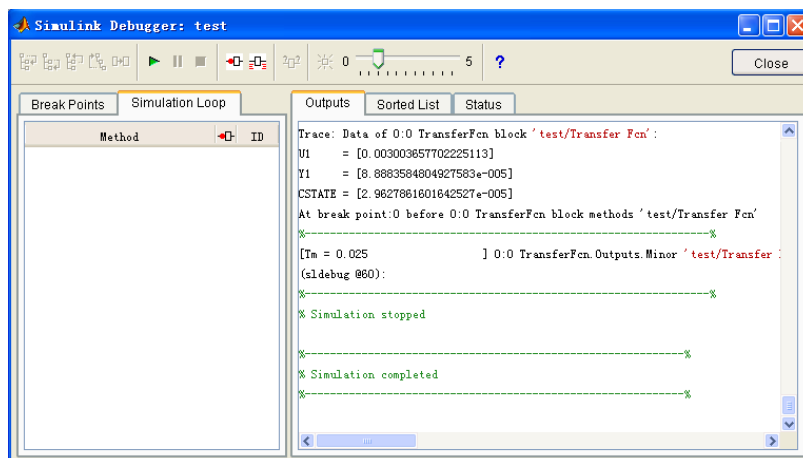


图 6-95 退出调试

第 7 章 图形用户界面

一个可发布的应用程序通常都需要具备一个友好的图形界面，本章就来介绍图形用户界面（GUI，Graphical User Interfaces）的设计，以便更好地进行人机交互。在 MATLAB 中有两种图形用户界面的设计方法，即纯 M 文件编程的方式和利用 GUIDE（Graphical User Interface Development Environment）的方式。由于利用 GUIDE 的方式在设计过程中更直观，所见即所得，同时减少了编码工作，所以本章着重介绍该方式。

下面给出利用 GUIDE 的方式设计的一个用户界面，如图 7-1 所示。从运行结果不难看出，它非常类似于 VB、Dephi 等编程语言设计的界面。

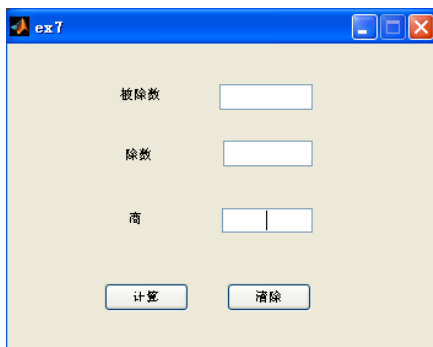


图 7-1 GUI 界面示例

7.1 界面设计

图形用户界面较之命令界面更加友好，通常在用户终端应用程序上采用图形界面。提供图形用户界面的应用程序能够使用户的学习和使用更为方便、容易。用户不需要知道应用程序究竟是怎样执行各种命令的，而只需要了解可见界面组件的使用方法；用户也不需要知道命令是怎样执行的，只要通过与界面交互就可以使指定的行为得以正确执行。

7.1.1 图形用户界面（GUI）概述

1. GUIDE 简介

在 MATLAB 中，图形用户界面是一种包含多种对象的图形窗口。用户必须对每一个对象进行界面布局和编程，从而使用户激活 GUI 每个对象时都能够执行相应的行为。另外，用户必须保存和发布所创建的 GUI，使得 GUI 能够真正地得到应用。

针对 GUI 而言，它首先实现界面布局，其次实现事件驱动，以使用户执行不同的操作激活相应的行为，最后实现保存和执行所创建的 GUI。

在 MATLAB 中，GUI 是一种包含多种对象的图形窗口，并且提供了开发 GUI 的集成开发环境 GUIDE。GUIDE 类似于 VB、Dephi 等编程语言，提供界面布局、对象属性和行为响应方

式的设置。同时 GUIDE 将 GUI 界面和对象（图形窗口、菜单、控件等）属性保存在一个.fig 文件中，同时将 GUI 初始化和对象行为响应方式的代码保存在一个同名的.m 文件中。这个 M 文件为实现对象行为响应方式的对应函数提供了一个程序架构，用户只需在指定的行为响应函数中，按照需求编写相应代码即可。

下面具体介绍这两个文件：

（1）FIG 文件：包括 GUI 界面布局及其对象属性的描述。它是一个二进制文件，可以通过下面介绍的界面设计编辑器的“File”→“Save”菜单命令保存该文件，可以通过“File”→“Open”菜单命令打开该文件。同时它可以实现图形对象句柄的保存和引用。

（2）M 文件：包括 GUI 初始化、行为响应方式等对应回调函数的代码，其中代码框架是自动生成的。可以通过命令窗口执行该 M 文件运行 GUI 界面。

创建 MATLAB 用户图形界面必须具有以下 3 个基本元素：

（1）组件——在 MATLAB GUI 中的每一个项目（按钮、标签、编辑框等）都是一个图形化组件。组件可分为 3 类：

- 图形化控件。如按钮、编辑框、列表、滚动条等。
- 静态元素。如窗口和文本字符串等。
- 菜单和坐标系。

（2）图形窗口——GUI 的每一个组件都必须安排在图形窗口中。在画数据图形时，图形通常会被自动创建。但还可以用函数 figure 来创建空图形窗口，空图形窗口经常用于放置各种类型的组件。

（3）回应——如果用户用鼠标单击或用键盘输入一些信息，那么程序就要有相应的动作。鼠标单击或输入信息是一个事件，如果 MATLAB 程序运行相应的函数，那么 MATLAB 函数肯定会有所反应。

2. 启动 GUIDE

在 MATLAB 中，在命令窗口中输入如下语句，即可得到如图 7-2 所示的 GUIDE 启动界面。

```
guide
```

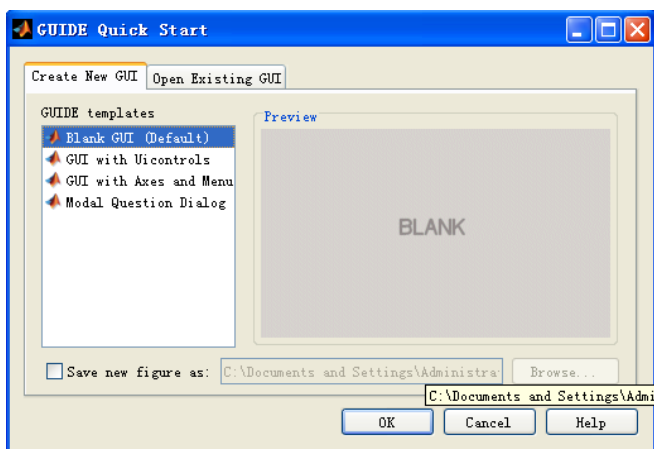


图 7-2 GUIDE 的启动界面

从图 7-2 中可以看到，GUIDE 提供了多种设计模板以方便用户定制新的 GUI，包括空白模板、带有界面控制的模板、带有坐标轴和菜单的模板以及问答式对话框模板，同时提供了 FIG

文件的保存路径和文件名。需要说明的是，每种模板都可以设置回调函数。如果 GUIDE 已经打开，选择“File”→“New”菜单命令，可以打开如图 7-2 所示的界面。

从图 7-2 中还可以看到，GUIDE 提供了打开现有 FIG 文件的方式。

在图 7-2 所示的界面上选择“Blank GUI (Default)”，单击“OK”按钮，即打开如图 7-3 所示的空白模板。

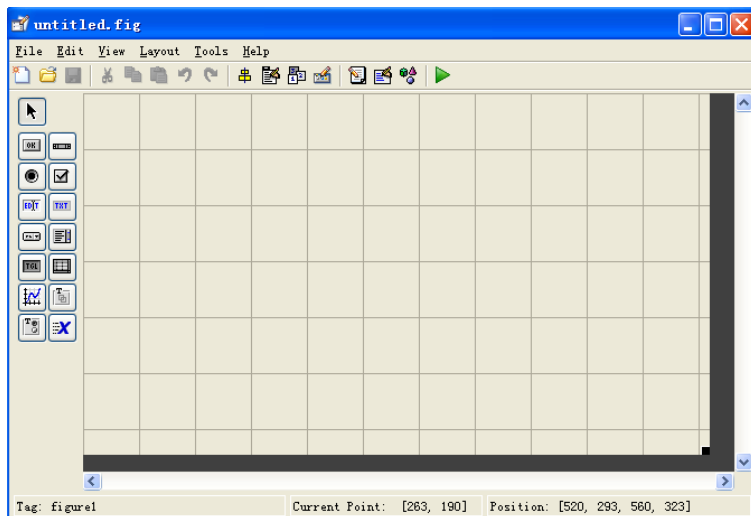


图 7-3 空白模板

在图 7-2 所示的界面上选择“GUI with Uicontrols”，即打开如图 7-4 所示的带有界面控制的模板。

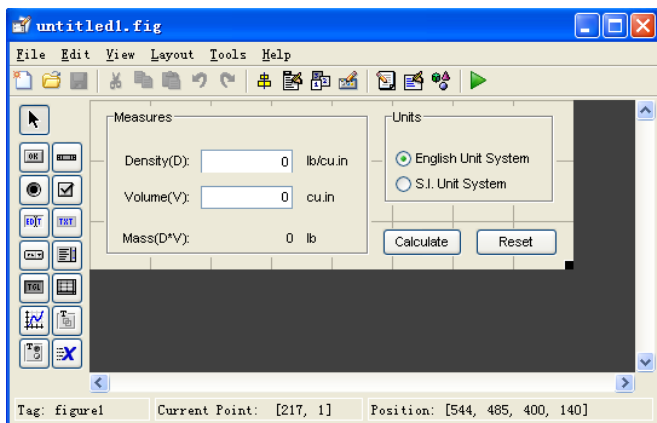


图 7-4 带有界面控制的模板

在图 7-2 所示的界面上选择“GUI with Axes and Menu”，即打开如图 7-5 所示的带有坐标轴和菜单的模板。

在图 7-2 所示的界面上选择“Modal Question Dialog”，即打开如图 7-6 所示的问答式对话框模板。

以空白模板为例，选择“File”→“Save As”菜单命令将图形保存为 ex7_1.fig 后，MATLAB 将自动生成 ex7_1.m，其代码的具体含义将在后面的小节中讲解。

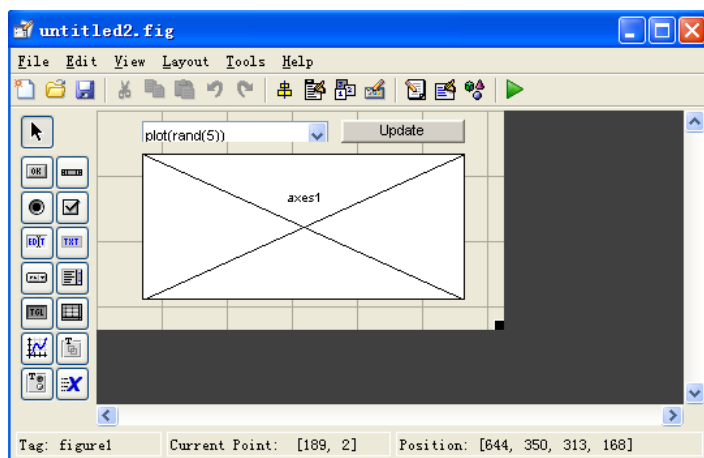


图 7-5 带有坐标轴和菜单的模板

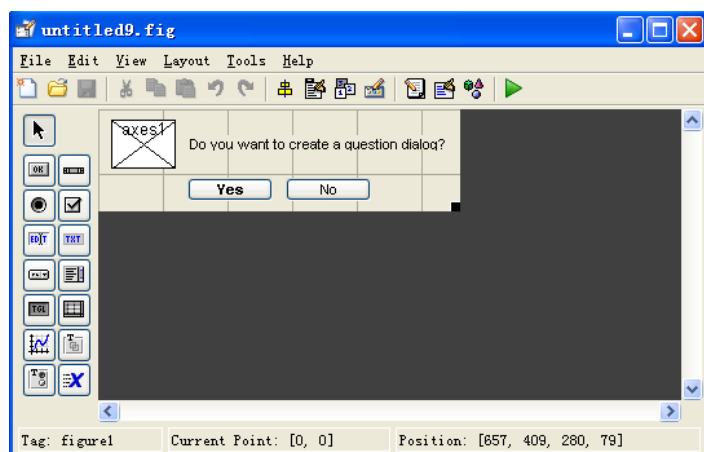


图 7-6 问答式对话框模板

7.1.2 GUIDE 的控件

在 4 种模板中 GUIDE 都提供了如图 7-7 所示的控件，以方便界面的设计，它们在图形界面上的显示形式如图 7-8 所示。



图 7-7 GUIDE 的控件

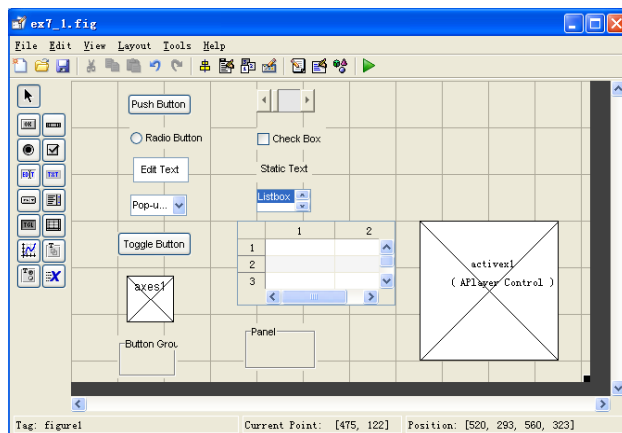


图 7-8 GUIDE 的控件显示形式

下面简单介绍各个控件的含义和用途。


- 按钮：通过单击调用相应的回调函数。
- 滚动条：通过移动滚动条的位置来改变指定范围内的数值输入。
- 单选框：通过单击调用相应的回调函数。单选框通常成组出现，配合按钮组使用，并且一组单选框中只有一个有效。
- 复选框：通过单击调用相应的回调函数。复选框通常成组出现，配合按钮组使用，并且一组复选框可以有多个有效。
- 编辑框：通过修改 **String** 属性由用户编辑或修改字符串的动态文本域，通常作为输入使用。
- 静态文本：通过修改 **String** 属性由用户编辑或修改字符串的静态文本域，通常作为标签使用。
- 弹出式菜单：通过 **String** 属性定义的选项列表提供菜单选项。
- 列表框：通过 **String** 属性定义的列表提供选择项。
- 套索按钮：通过单击能够产生一个二进制状态的行动，同时该按钮在外观上可以保持下陷状态和弹起状态，不同状态调用不同的回调函数。
- 表格：通过 **Data** 属性定义的数据填充表格。
- 坐标轴：通过坐标轴可以显示图形。
- 面板：通过面板可以构成图形窗口中的子区域，它将相关联的控件封闭在该区域中，以便用户界面更容易理解。
- 按钮组：通过按钮组可以构成由按钮和套索按钮构成的子区域。
- Active X 控件：通过 Active X 控件可以调用外部程序。

7.1.3 GUIDE 开发环境

GUIDE 开发环境主要包括如下几个部分：

- 界面设计器：在界面区域上添加指定控件，并可以通过菜单、鼠标或键盘操作控制版式。
- 菜单编辑器：创建窗口菜单和文本菜单。
- 工具条编辑器：创建窗口菜单和文本菜单。
- 属性检查器：对添加的控件进行参数设置，通过参数值也可以控制版式。
- 对象浏览器：查看界面中所有对象的信息及相互的关系。
- M 文件编辑器：编写、查看和修改作为初始化和回调函数的 M 文件代码。

1. 界面设计器

图 7-3 给出了界面设计器的外观，可以利用控件面板（含选择工具）、弹出式菜单、工具栏和菜单栏设计界面区域，激活后的界面区域就是 GUI 的图形窗口。

(1) 控件面板

控件面板中列出了所有可用的控件，如图 7-7 所示。当添加指定控件到界面区域时，只需将该控件拖放到界面区域即可，然后可以改变该控件的大小和位置。

(2) 弹出式菜单

选中界面区域的指定控件，单击鼠标右键可以显示与其相关联的弹出式菜单。与按钮相关联的弹出式菜单如图 7-9 所示。

从图 7-9 中可以看出，弹出式菜单除包括常用的“剪切”、“复制”、“粘贴”、“清除”和“实例复制”菜单项外，还包括后面要提到的与 Tab 次序相关的“送到最前端”和“放到最后端”、

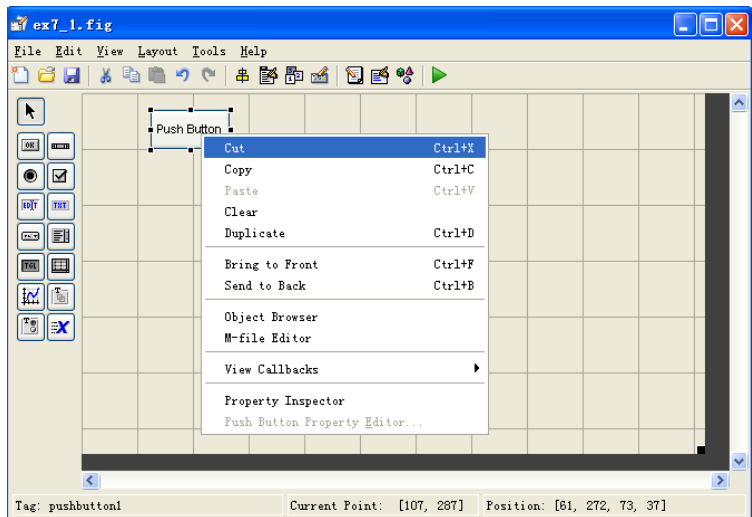


图 7-9 与按钮相关联的弹出式菜单

“对象浏览器”、“M 文件编辑器”、“查看回调函数”、“属性检查器”和“与按钮相关的属性编辑器”（不同控件该项不同）菜单项。

“查看回调函数”菜单项还包含如图 7-10 所示的子菜单项，不同的控件对应不同的子菜单项。

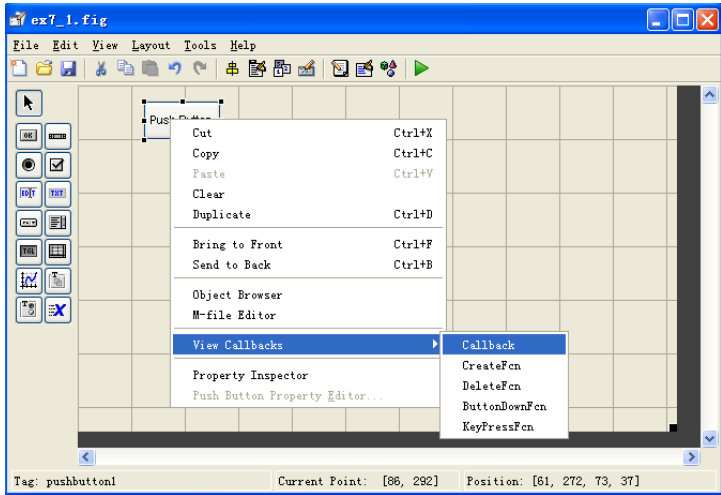



图 7-10 与按钮相关联的“查看回调函数”菜单项的子菜单

选中界面区域的空白位置，单击鼠标右键可以显示与图形窗口对象相关联的弹出式菜单，如图 7-11 所示。

(3) 工具栏

工具栏如图 7-12 所示，每个图标对应一个快捷方式，并且这些图标在菜单栏中都有实现相同功能的菜单项相对应。

前 8 个图标的功能和操作与 Windows 中经常用到的相同，这里不再赘述。

当单击图标  时，可打开如图 7-13 所示的对齐工具，其中包括水平和垂直两个方向的对齐，对齐的方式如小图标所示。该工具是为了多个控件的整齐排列而设置的，对单一控件无效果。使用时首先需要用鼠标将所要排列的控件选中，然后按照指定格式排列。

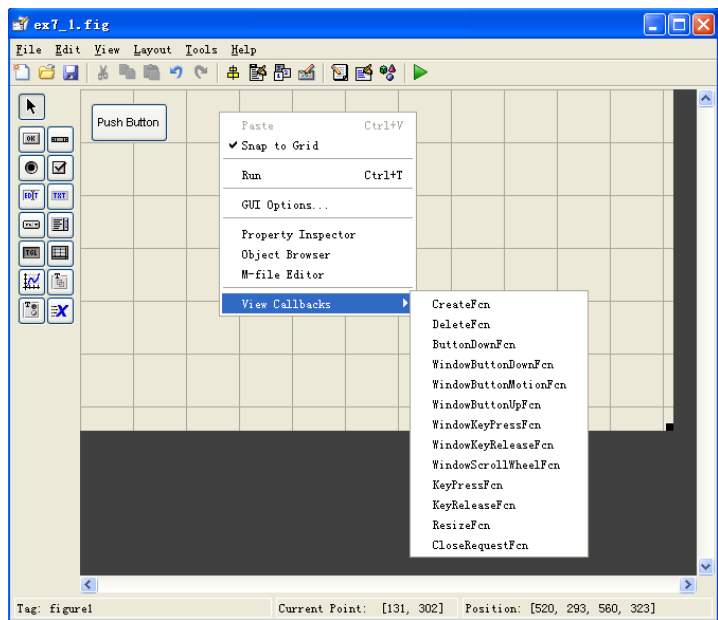


图 7-11 与图形窗口对象相关联的弹出式菜单

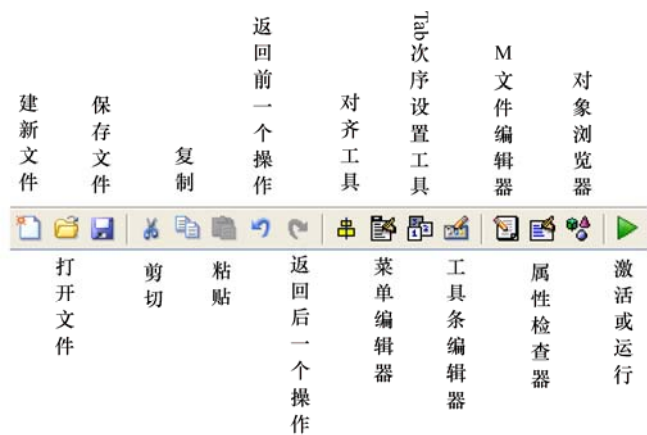


图 7-12 工具栏

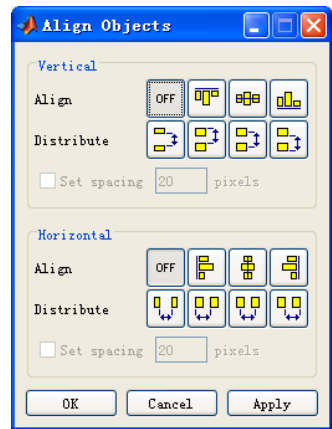






图 7-13 对齐工具


当单击图标时，可打开如图 7-14 所示的菜单编辑器，具体内容将在后面讲解。


当单击图标时，可得到如图 7-15 所示的 Tab 次序设置工具。当界面区域布置了多个控件并激活后，使用 Tab 键可以遍历所有控件，Tab 次序就是指遍历的次序。使用该工具可以设置和调整 Tab 次序。前面介绍的弹出式菜单中的“送到最前端”和“放到最后端”菜单项就是指这个次序。

当单击图标时，可打开如图 7-16 所示的工具条编辑器，具体内容将在后面讲解。

当单击图标时，可打开如图 7-17 所示的 M 文件编辑器，具体内容将在后面讲解。

当单击图标时，可打开如图 7-18 所示的属性编辑器，具体内容将在后面讲解。

当单击图标时，可打开如图 7-19 所示的对象浏览器，具体内容将在后面讲解。

当单击图标时，可以激活和运行设计的图形用户界面。

(4) 菜单栏

这里仅介绍没有工具栏图标对应的菜单栏中的菜单项。

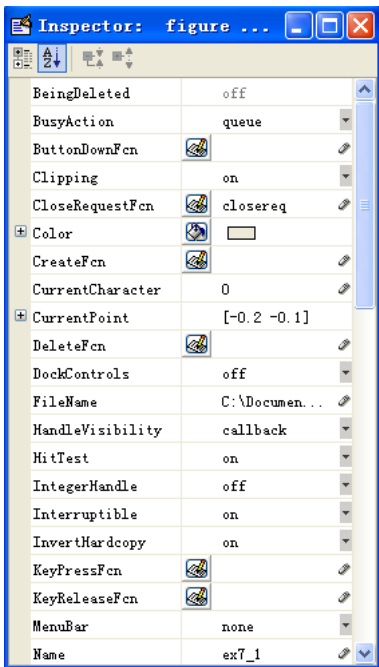


图 7-18 属性编辑器

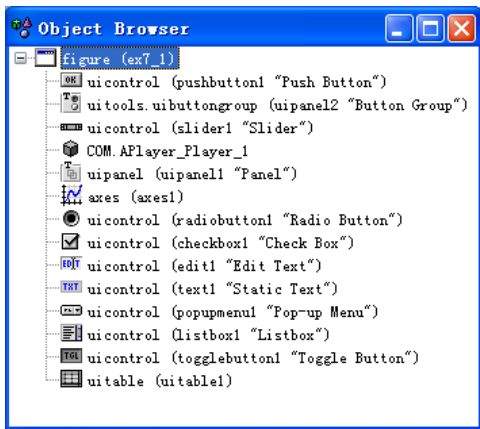


图 7-19 对象浏览器

菜单命令“Layout”→“Bring Forward”和“Layout”→“Bring Backward”用于单步移动所选控件的 Tab 次序。

单击“Tools”→“Grid and Rules”菜单命令，打开如图 7-20 所示的网格和标线对话框。

“Show rulers”选项用于设置标尺是否可见；“Show guides”选项用于设置标线是否可见；“Show grid”选项用于设置网格是否可见；“Grid Size”选项用于设置网格的间隔，默认值为 50 像素；“Snap-to-grip”选项用于将在标线附近 9 像素范围内移动或重画的对象自动放置在标线上，无论网格是否可见。

单击“Tools”→“GUI Options”菜单命令，打开如图 7-21 所示的 GUI 选项对话框。

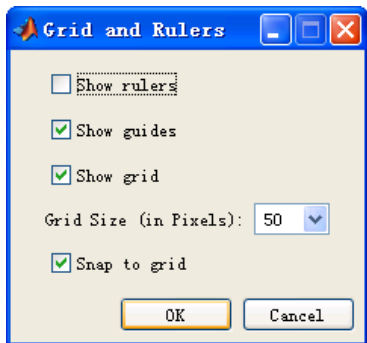


图 7-20 网格和标线对话框

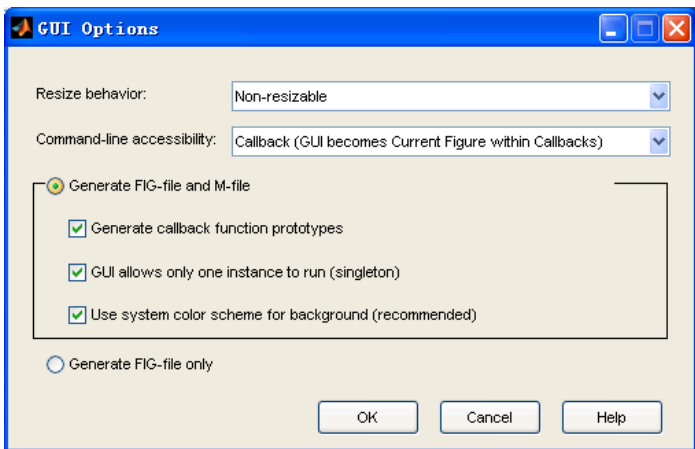


图 7-21 GUI 选项对话框

该对话框可以进行设置的选项如下：

- 窗口重画行为 (Resize behavior): 选择是否可以重画 GUI 所在的图形窗口以及如何管理重画过程。包含如下 3 种情况: Non-resizable (不能改变窗口大小, 默认值); Proportional (允许按照新的图形窗口尺寸按比例重新绘制 GUI 组件, 重画过程中将不改变控件标签字体的大小); Other (按照用户指定的方式变化, 需要用户编写 ResizeFcn 属性定义的回调函数)。
- 命令行访问 (Command-line accessibility): 若需要创建包含坐标轴等绘图工具的 GUI 时, GUI 要支持命令行的访问。包含如下 4 种情况: Callback (只可通过 GUI 回调函数访问); Off (禁止命令行对 GUI 图形窗口的访问); On (允许命令行对 GUI 图形窗口进行访问); Other (通过设置 Handle Visibility 和 IntergerHandle 这两个图形窗口属性值决定是否能被命令行访问)。
- 生成 FIG 文件和 M 文件 (Generate FIG-file and M-file): 若希望 GUIDE 同时创建 FIG 文件和应用程序 M 文件, 则选择该选项, 同时激活如下选项:
 - 生成回调函数原型 (Generate callback function prototypes): GUIDE 将在应用程序 M 文件中自动为每个控件添加回调函数。
 - 同一时刻仅允许运行一个应用程序实例 (GUI allows only one instance to run (singleton)): 该选项决定一次运行过程中仅有一个实例。
 - 使用系统背景颜色设置 (Use system colors scheme for background (recommended)): 该选项决定图形窗口的背景是否采用系统颜色配置方案。
- 仅生成 FIG 文件 (Generate FIG-file only): 该选项决定 GUIDE 仅创建 FIG 文件。

2. 菜单编辑器

GUIDE 菜单编辑器的外观如图 7-14 所示, 能够创建菜单栏菜单 (GUI 激活后显示在界面区域) 和上下文菜单 (单击鼠标右键显示) 两种菜单。

(1) 菜单栏菜单

创建菜单栏菜单的步骤如下: 首先单击左下角的“Menu bar”标签 (进入菜单栏菜单编辑模式); 其次使用如图 7-22 的工具条创建菜单。

当新增菜单项时, 界面如图 7-23 所示, 单击“Untitled 1”, 出现如图 7-24 所示的菜单项属性设置。

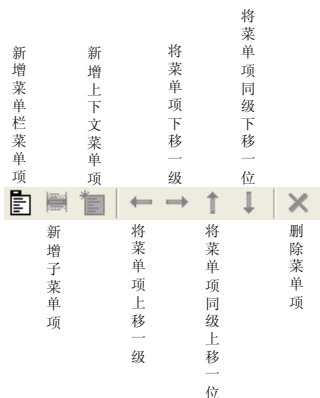


图 7-22 菜单编辑器工具条

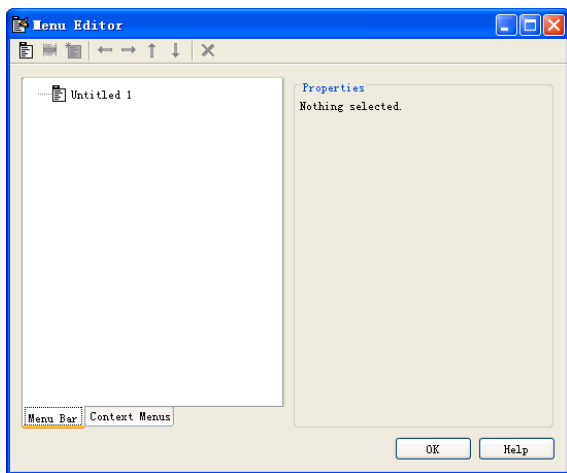


图 7-23 新增菜单项

在图 7-24 中单击“More Properties”按钮，可以设置菜单项显示名称、调用名称（如用于回调函数）、快捷键、是否在与同级上一个菜单项间增加分隔线、是否标志已选择的状态、是否激活、回调函数等属性，如图 7-25 所示。

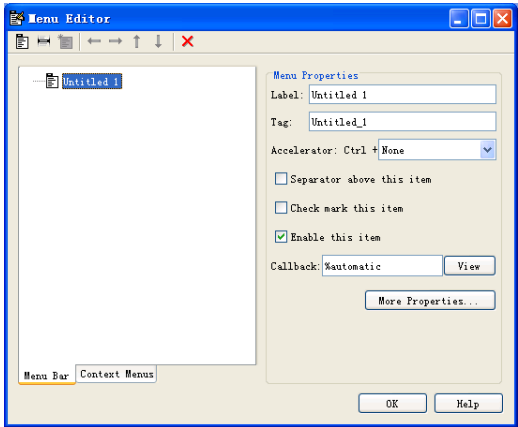


图 7-24 菜单项属性设置

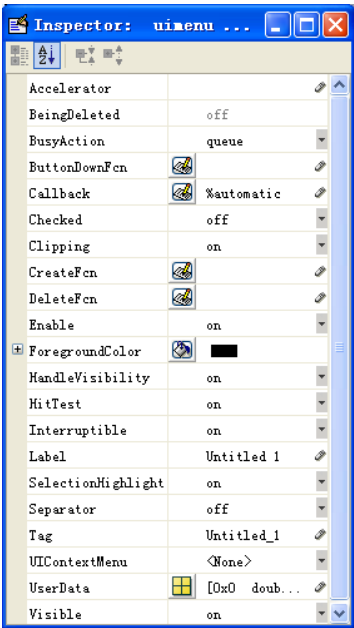


图 7-25 菜单项更多属性

(2) 上下文菜单

创建上下文菜单的步骤如下：首先单击左下角的“Context Menus”标签（进入上下文菜单编辑模式）；其次使用如图 7-22 的工具条创建菜单。

当新增上下文父菜单时，界面如图 7-26 所示，单击“Untitled 1”，出现如图 7-27 所示的上下文父菜单属性设置。

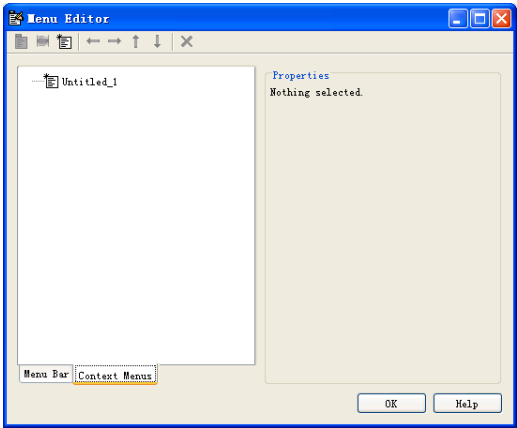


图 7-26 新增上下文父菜单

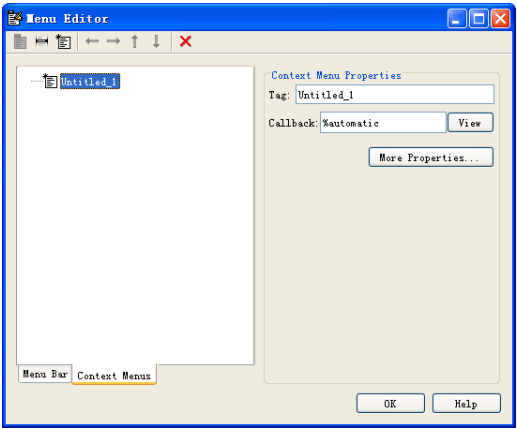


图 7-27 上下文父菜单属性设置

在图 7-27 中单击“More Properties”按钮，可以设置上下文父菜单调用名称（如用于回调函数）、回调函数等属性，如图 7-28 所示。

这里需要说明的是，上下文父菜单将不会显示，仅供其他对象调用（如通过设置其他控件

的 `UINavigationController` 属性调用), 而其子菜单项将会在调用时显示。子菜单项的建立和设置同菜单栏菜单相同, 这里不再赘述。

3. 工具条编辑器

GUIDE 工具条编辑器的外观如图 7-16 所示, 能够创建 GUI 激活后显示在界面区域的工具条。

创建工具条的步骤如下: 首先单击右边的“Toolbar Properties”标签, 设置工具条调用名称、是否可见及更多属性(如图 7-29 所示); 其次使用如图 7-16 所示的界面创建工具条。

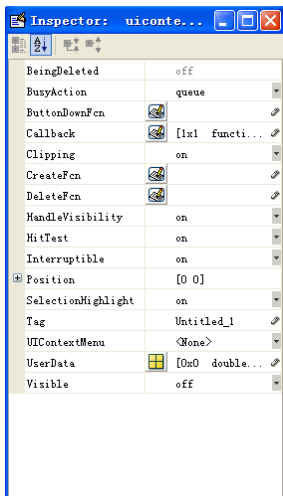


图 7-28 上下文菜单更多属性

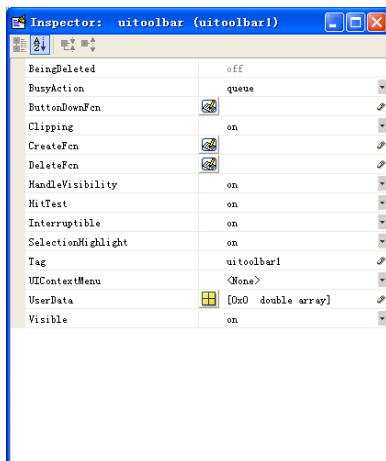





图 7-29 工具条更多属性

若需要增加如“New”等已有图标时, 只需选中  并单击“Add”按钮, 即得到如图 7-30 所示的界面, 并可以设置图标图形、调用名称、注释字符、是否激活、左侧是否有分隔线、回调函数、更多属性和恢复默认设置。

若需要增加如“Toggle Tool”等自定义图标时, 只需选中  并单击“Add”按钮, 或双击 , 即得到如图 7-31 所示的界面, 并可以设置图标图形、调用名称、注释字符、是否激活、左侧是否有分隔线、回调函数和更多属性。

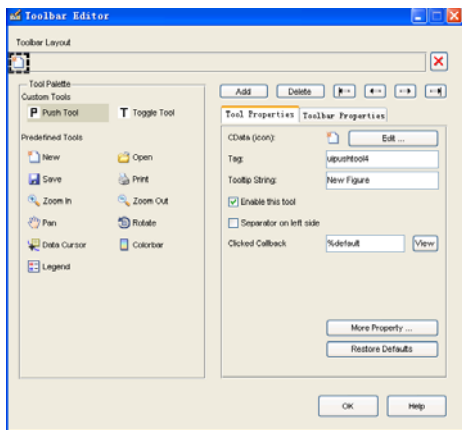


图 7-30 增加已有图标

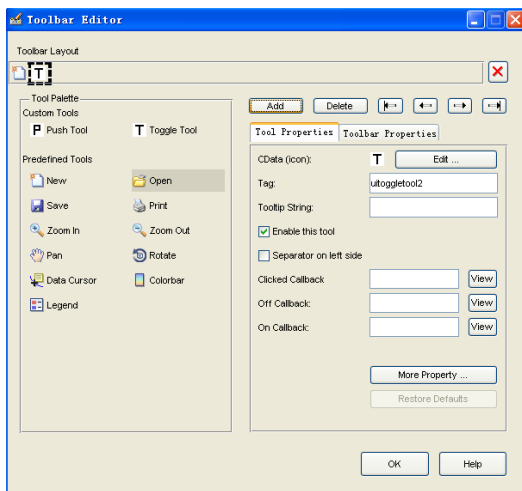



图 7-31 增加自定义图标

4. 属性检查器

属性检查器提供一个所有可设置属性的列表，并可以显示和手工修改当前的属性值，也可以通过代码读取和修改属性值。不同的对象对应不同的属性检查器，下面以按钮的一些属性为例，说明属性检查器的应用。

- **BackgroundColor**: 设置背景颜色。可以进行三元色设置 (red, green, blue)，直接填写或修改成其他数据，将改变背景颜色；单击  按钮出现更多的颜色种类并可以进行更多的颜色设置，如图 7-32 所示。

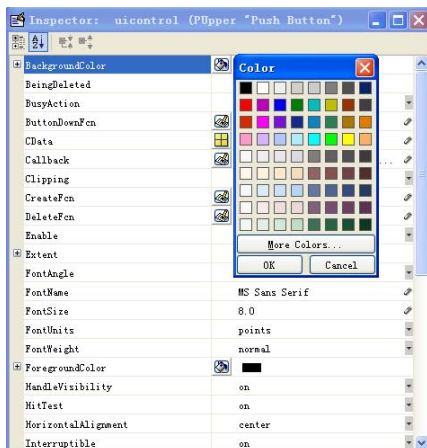


图 7-32 按钮背景颜色设置

- **BusyAction**: 设置在不可中断的回调函数执行期间发生事件的处理方式，queue 表示保存事件直至不可中断回调函数执行完毕后处理，cancel 表示放弃该事件并将事件从序列中删除。
- **ButtonDownFcn**: 设置鼠标单击的回调函数。
- **Enable**: 设置是否可用。
- **FontName**: 设置按钮显示标签的字体名。
- **Interruptible**: 设置回调函数是否可中断。
- **Position**: 可以以向量的形式设置按钮的位置，也可以打开下拉菜单，通过各分量设置按钮的位置。
- **String**: 设置按钮显示标签的文字。
- **Tag**: 设置按钮的调用名称。
- **UIContextMenu**: 选用已保存的上下文菜单，对于已保存的上下文菜单，通过下拉菜单可以看到它们对应的 Tag 值。
- **Visible**: 设置是否可见。

5. 对象浏览器

对象浏览器显示图形窗口中所有对象的继承关系。图形窗口处于最高层；图形窗口中菜单栏菜单的第一级目录、上下文菜单、工具条、独立的控件等处于第二级；菜单栏菜单的第二级目录、上下文菜单项、工具条各图标、按钮组或面板中的控件处于第三级；以此类推。

6. M 文件编辑器

单击  按钮或执行 “View” → “M-file Editor” 菜单命令启动 M 文件编辑器，如图 7-33

所示。用户可以在编辑器中编写自己的回调函数。下面的 7.2 节将详细介绍如何编写回调函数，以实现各种需求。

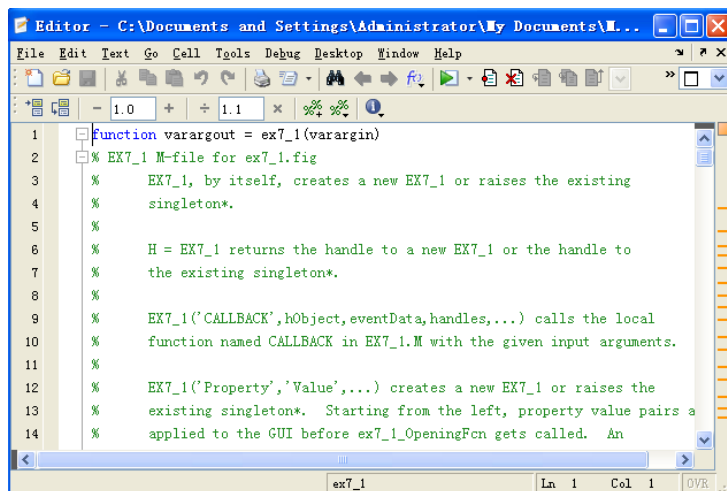


图 7-33 M 文件编辑器

7.2 程序设计

本节将着重介绍编程设计 GUI，包括对象的回调函数、程序的一般结构、对象属性的访问、对象间数据传递、GUI 与 M 文件的数据交互以及 GUI 与 Simulink 仿真的数据交互等。

7.2.1 对象的回调函数

在 MATLAB 中，对句柄图形对象还可以设置一些事件响应函数，它们可以在对象创建或对象删除等事件发生时执行，从而实现特定事件触发下需要的某些功能。这些事件响应函数称为句柄图形对象的回调函数，我们经常称之为对象的回调函数。

回调函数是由指定事件触发或激活的，前面已经介绍过不同的对象对应不同的回调函数，如按钮包含 Callback、CreateFcn、DeleteFcn、ButtonDownFcn 和 KeyPressFcn 回调函数，而按钮组包含 CreateFcn、DeleteFcn、ButtonDownFcn、ResizeFcn 和 SelectionChangeFcn 回调函数。表 7-1 列出了所有的回调函数及其触发事件和使用范围。

表 7-1 回调函数及其触发事件和使用范围

回调函数名称	触发事件	使用范围
ButtonDownFcn	当单击组件或图形窗口上或其边界 5 像素范围内时触发。对于控件，应将其 Enable 属性置为 on	坐标轴、图形窗口、按钮组、面板、按钮、滚动条、单选框、复选框、文本编辑框、静态文本框、列表框、套索按钮
Callback	当单击组件或菜单项时触发	上下文菜单、菜单栏菜单、按钮、滚动条、单选框、复选框、文本编辑框、静态文本框、列表框、套索按钮
CloseRequestFcn	在图形窗口关闭前触发	图形窗口
CreateFcn	在组件或图形窗口创建后，且在组件或图形窗口显示前触发，用于初始化相关参数	坐标轴、图形窗口、按钮组、上下文菜单、菜单栏菜单、面板、按钮、滚动条、单选框、复选框、文本编辑框、静态文本框、列表框、套索按钮

(续表)

回调函数名称	触发事件	使用范围
DeleteFcn	在组件或图形窗口删除前触发,用于清理相关参数	坐标轴、图形窗口、按钮组、上下文菜单、菜单栏菜单、面板、按钮、滚动条、单选框、复选框、文本编辑框、静态文本框、列表框、套索按钮
KeyPressFcn	当使用键盘单击处于焦点状态的组件或图形窗口时触发,可以返回所使用的键	图形窗口、按钮、滚动条、单选框、复选框、文本编辑框、静态文本框、列表框、套索按钮
KeyReleaseFcn	当使用键盘单击后松开处于焦点状态的图形窗口时触发,可以返回所使用的键	图形窗口
ResizeFcn	当图形窗口、面板和按钮组改变尺寸大小时触发,且其 Resize 属性为 On	图形窗口、面板和按钮组
SelectionChangeFcn	当在按钮组中选择不同的单选框和套索按钮时触发	按钮组
WindowButtonDownFcn	当单击图形窗口时触发	图形窗口
WindowButtonMotionFcn	当鼠标在图形窗口中移动时触发	图形窗口
WindowButtonUpFcn	当鼠标释放时触发	图形窗口
WindowScrollWheelFcn	当图形窗口处于焦点且鼠标滚动时触发	图形窗口

创建回调函数的方法非常简单：只需首先选中指定对象，然后单击鼠标右键，在上下文菜单的“View Callbacks”中选择回调函数即可。这时将进入到 M 文件编辑器的指定位置，在已有程序框架下编写恰当的代码即完成回调函数的创建。

7.2.2 程序的一般结构

当启动 GUIDE 且选择新建“GUI with Uicontrols”时，将可以看到如图 7-4 所示的带有界面控制的模板，当保存为 ex7_2.fig 后，MATLAB 自动生成 ex7_2.m。下面通过表 7-2 介绍生成代码的各部分及其含义。

表 7-2 示例程序中的各部分及其含义

各部分描述	含义
function varargout = ex7_2(varargin)	主函数声明，varargin 为输入变量，varargout 为输出变量， ex7_2 为自动生成的主函数，因为对应的 FIG 文件名为 ex7_2
%.....	注释文字，方便程序的阅读及调试
% Begin initialization code - DO NOT EDIT	
...	初始化代码，不要进行编辑
% End initialization code - DO NOT EDIT	
ex7_2_OpeningFcn 子函数	在 GUI 出现之前进行初始化设计，并且子函数名 ex7_2_OpeningFcn 是自动生成的 handles.output = hObject;%为主函数选择默认命令行输出，其中 hObject 表示图形界面，handles 表示图形界面及其所有对象，它的数据类型都是结构 guidata(hObject, handles);%更新 handles 数据 initialize_gui(hObject, handles, false);%调用子函数 initialize_gui
ex7_2_OutputFcn 子函数	在 GUI 运行结束返回命令行前进行输出设计 varargout{1} = handles.output; %返回默认命令行输出

(续表)

各部分描述	含义
density_CreateFcn 子函数	文本编辑框 density 的 CreateFcn 回调函数，其中 density 是指定文本编辑框的 Tag 属性值，density_CreateFcn 是自动生成的回调函数名
density_Callback 子函数	文本编辑框 density 的 Callback 回调函数
volume_CreateFcn 子函数	文本编辑框 volume 的 CreateFcn 回调函数
volume_Callback 子函数	文本编辑框 volume 的 Callback 回调函数
calculate_Callback 子函数	按钮 calculate 的 Callback 回调函数
reset_Callback 子函数	按钮 reset 的 Callback 回调函数
unitgroup_SelectionChangeFcn 子函数	按钮组 unitgroup 的 SelectionChangeFcn 回调函数
initialize_gui 子函数	供主函数和其他子函数调用的内部子函数

由表 7-2 可以看出，程序必有部分包括主函数声明、不能编辑的初始化代码、相应的 OpeningFcn 和 OutputFcn 子函数。同时增加对象（控件）可以设置其回调函数，还可以设置其他需调用的子函数。

7.2.3 对象属性的访问

对于对象属性的访问包括读取和设置两类，同时包括访问自身属性和其他对象属性两类，下面以 7.2.2 节中的代码为例说明访问方法。

• 访问自身属性

density_Callback 子函数中的代码 get(hObject, 'String')表示读取自身的 String 属性值，并可以将返回值赋予变量，其中 str2double 函数实现字符串到数值的转换；代码 set(hObject, 'String', 0)表示将自身的 String 属性值设置为 0。

• 访问其他对象属性

initialize_gui 子函数中的代码 set(handles.text4, 'String', 'lb/cu.in')表示设置图形窗口中 Tag 属性值为 text4 的对象的 String 属性值为 lb/cu.in。同理，get(handles.text4, 'String') 表示读取图形窗口中 Tag 属性值为 text4 对象的 String 属性值。

【例 7-1】实现两个按钮的交互。具体步骤如下：

(1) 利用空白模板创建如图 7-34 所示的界面，并保存为 ex0701.fig。

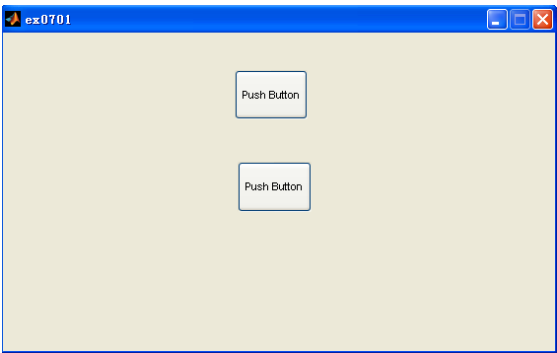


图 7-34 示例界面

(2) 利用属性编辑器将两个按钮的 Tag 属性值手工改为“Pupper”（上方按钮）和“Pdown”（下方按钮）。

(3) 设置上方按钮的 Callback 回调函数, 实现先将上方按钮的 String 属性值增加“(Upper)”, 再将下方按钮的 String 属性值增加“(Down)”。回调函数的内容如下所示:

```
function PUpper_Callback(hObject, eventdata, handles)
% hObject      handle to PUpper (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
oldUS=get(hObject,'String');
newUS=strcat(oldUS,'(Upper)');
set(hObject,'String',newUS);
oldDS=get(handles.PDown,'String');
newDS=strcat(oldDS,'(Down)');
set(handles.PDown,'String',newDS);
```

(4) 激活设计的界面并单击上方按钮, 可以得到如图 7-35 所示的界面。

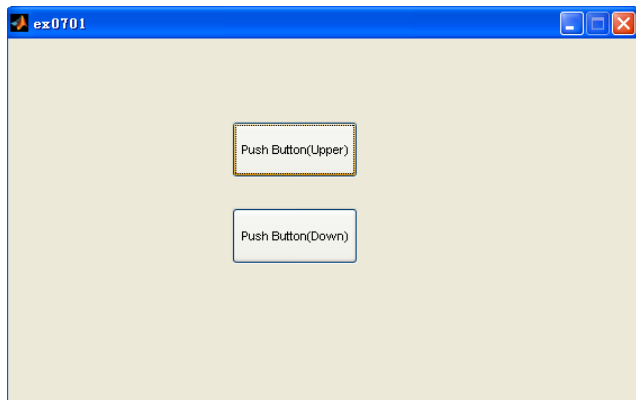


图 7-35 示例运行界面

7.2.4 对象间数据传递

通过前面的方法可以实现对象间属性的访问, 有时还需要变量值的交互, 即数据传递。

以 7.2.2 节中的代码为例, 按照执行次序, 变量 `metricdata` 首次出现在 `initialize_gui` 子函数的代码中, 它的数据类型是结构体。代码 `isfield(handles, 'metricdata')` 用于判断该变量是否存在, 同时表明若存在将以 `handles` 字段的形式出现, 即通过 `handles.metricdata` 调用。由于该变量为结构体, 其值或属性值可以如下设置:

```
handles.metricdata.density = 0;
handles.metricdata.volume = 0;
```

`density_Callback` 子函数中的代码对变量 `metricdata` 进行数据更改, 通常的做法如下:

```
handles.metricdata.volume = volume;
```

需要说明的是, 在初始化或数据更改后应该更新 `handles` 的数据。在对象的回调函数和 `OpeningFcn` 与 `OutputFcn` 子函数中, 可以使用下述语句更新:

```
guidata(hObject, handles);
```

在如 `initialize_gui` 的子函数中, 必须将 `handles` 作为参数传入子函数, 并且可以使用下述语句更新:

```
guidata(handles.FigureTag, handles);
```

其中 FigureTag 为对应图形窗口的 Tag 属性值，在 ex7_2 中为 figure1，即：

```
guidata(handles.figure1, handles);
```

【例 7-2】实现数据在不同控件中的传递。具体步骤如下：

(1) 利用空白模板创建如图 7-36 所示的界面，并保存为 ex0702.fig。

(2) 利用属性编辑器将两个静态文本的 Tag 属性值手工改为“STU”(上方)和“STD”(下方)，String 属性值手工改为“输入 1”(上方)和“输出 2”(下方)；两个文本编辑框的 Tag 属性值手工改为“ETU”(上方)和“ETD”(下方)；两个按钮的 Tag 属性值手工改为“PBL”(左边)和“PBR”(右边)，String 属性值手工改为“保存”(左边)和“计算”(右边)。

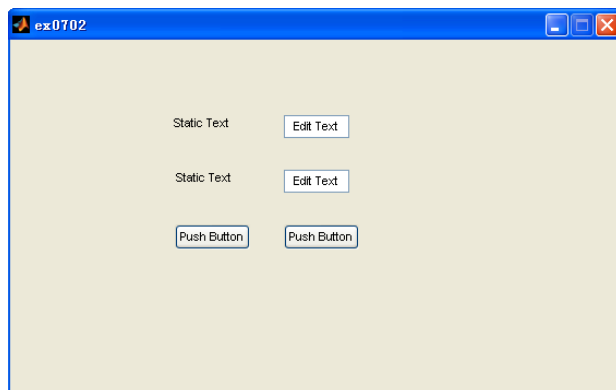


图 7-36 示例界面

(3) 设置 OpeningFcn 子函数，实现初始化，包括赋值给变量 ETUinput，将上方文本编辑框的 String 属性值设置为“ETUinput”，下方文本编辑框的 String 属性值设置为“空白”。子函数的内容如下所示：

```
function ex0902_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for ex0902
handles.output = hObject;

handles.ETUinput=15;
set(handles.ETU,'String',handles.ETUinput);
set(handles.ETD,'String','');

% Update handles structure
guidata(hObject, handles);
```

(4) 设置左边按钮的 Callback 回调函数，实现将上方文本编辑框的数据保存在变量 ETUinput 中。回调函数的内容如下所示：

```
function PB_L_Callback(hObject, eventdata, handles)
ETDstr=get(handles.ETU,'String');
ETDnum=str2double(ETDstr);
if isnumeric(ETDnum)
```



```
handles.ETUinput=ETDnum;  
guidata(hObject, handles);  
end
```

(5) 设置右边按钮的 Callback 回调函数, 实现将经过保存的变量 ETUinput 的 10 倍值写入下方文本编辑框中。回调函数的内容如下所示:

```
if isfield(handles, 'ETUinput')  
    set(handles.ETD, 'String', 10*handles.ETUinput);  
end
```

(6) 激活设计的界面, 如图 7-37 所示。

单击“计算”按钮, 则显示如图 7-38 所示的界面。

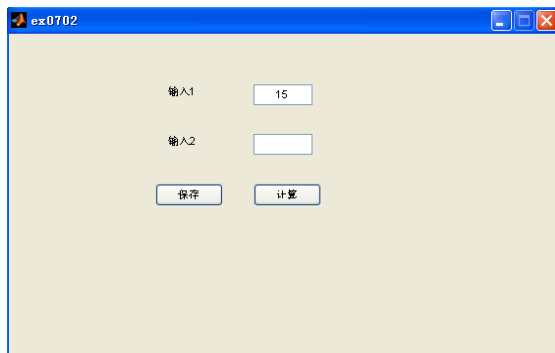


图 7-37 激活界面 1

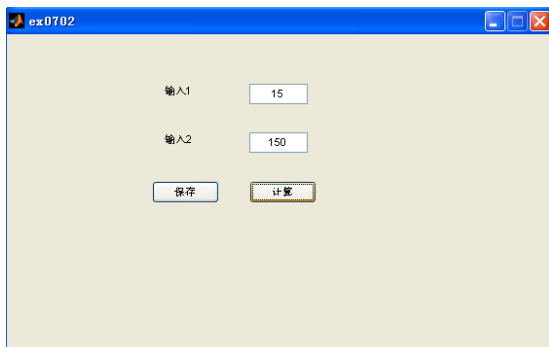


图 7-38 示例运行界面 1

在“输入 1”中重新输入 5, 单击“计算”按钮, 则在“输入 2”中显示的计算结果如图 7-39 所示。

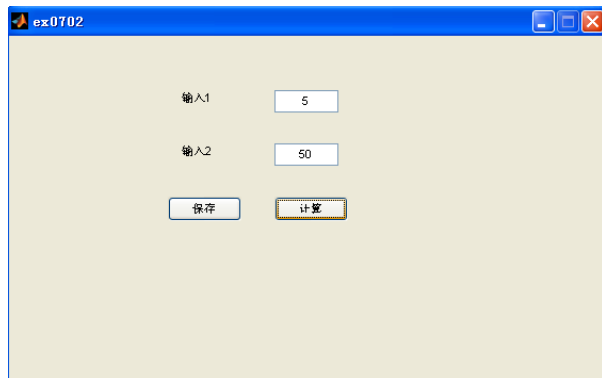


图 7-39 示例运行界面 2

7.2.5 GUI 与 M 文件的数据交互

这里提到的 GUI 与 M 文件的数据交互包括如下三个含义:

- (1) GUI 调用脚本或函数式的 M 文件。只需像使用 MATLAB 自带函数一样调用即可。
- (2) GUI 调用工作空间中的变量。直接使用即可。
- (3) 工作空间调用 GUI 中的数据。

通过 evalin 函数可以实现上述功能, 其具体用法如下:

```
evalin('base','expression'); %在 MATLAB 工作空间内执行表达式 expression, 其中表达式是字符串形式
vars = evalin('base','expression'); %返回在 MATLAB 工作空间内的执行结果
evalin('caller','expression'); %在调用函数工作空间内执行表达式 expression, 其中表达式是字符串形式
vars = evalin('caller','expression'); %返回在调用函数工作空间内的执行结果
```

【例 7-3】实现 GUI 与工作空间里的数据交互。具体步骤如下：

(1) 利用空白模板创建如图 7-40 所示的界面，并保存为 ex0703.fig。

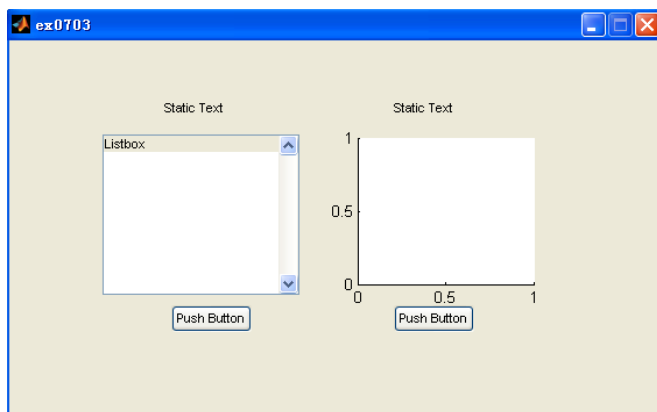


图 7-40 示例界面

(2) 利用属性编辑器将两个静态文本的 Tag 属性值手工改为“STL”（左边）和“STR”（右边），String 属性值手工改为“工作空间数据”（左边）和“图形显示结果”（右边）；列表框的 Tag 属性值手工改为“LB”；坐标轴的 Tag 属性值手工改为“AX”；两个按钮的 Tag 属性值手工改为“PBL”（左边）和“PBR”（右边），String 属性值手工改为“导入数据”（左边）和“绘制图形”（右边）。设置结果如图 7-41 所示。

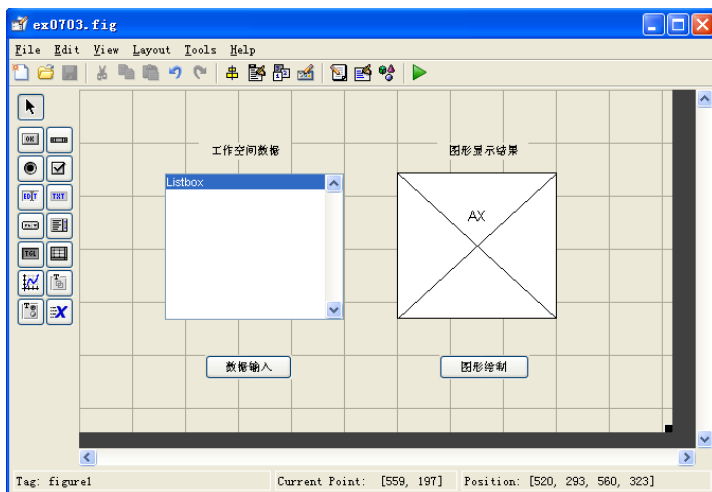


图 7-41 修改属性后的界面

(3) 设置 OpeningFcn 子函数，实现初始化，包括将目前的工作空间变量导入到列表框中，并将第一个列表项作为默认列表项。子函数的内容如下所示：

```
function lb_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
```

```
update_LB(handles)
```

```
guidata(hObject, handles);
```

其中，update_LB 子函数的内容如下所示：

```
function update_LB(handles)
vars = evalin('base','who');
set(handles.LB,'String',vars)
set(handles.LB,'Value',1)
```

(4) 设置左边按钮的 Callback 回调函数，实现将工作空间变量导入到列表框中，并将第一个列表项作为默认列表项。回调函数的内容如下所示：

```
function PBL_Callback(hObject, eventdata, handles)
update_LB(handles)
```

(5) 设置右边按钮的 Callback 回调函数，实现在坐标轴中绘制选中的工作空间变量，并将其值加倍后返回工作空间。回调函数的内容如下所示：

```
function PBR_Callback(hObject, eventdata, handles)
axes(handles.AX)
list_entries = get(handles.LB,'String');
index_selected = get(handles.LB,'Value');
if length(index_selected) ~= 1
    errordlg('You must select one variable','Incorrect Selection','modal')
else
    var = list_entries{index_selected};
end
evalin('base',['plot(',var,')']);
figure
evalin('base',['plot(',var,')']);
evalin('base',[2*var]);
```

其中，axes(handles.AX)表示选择 AX 作为绘图区域，figure 表示在 MATLAB 中生成新的图形窗口。

(6) 在命令窗口输入如下语句，执行后工作空间有 5 个变量，再激活设计的界面，如图 7-42 所示。

```
clear
clc
t=1:20;
a=tan(t);
b=cos(t);
c=exp(t);
d=sqrt(t);
```

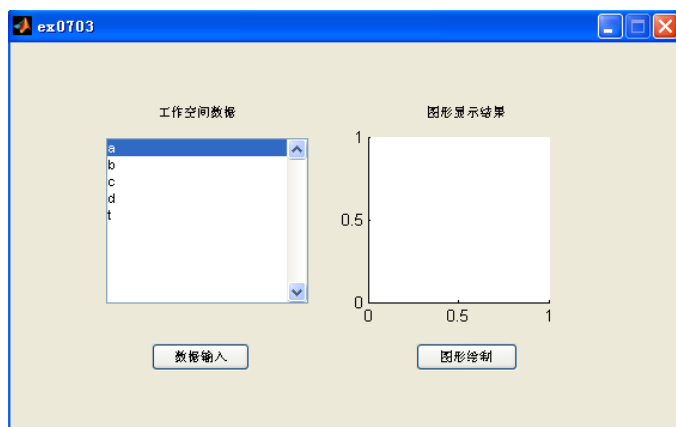


图 7-42 激活界面

选择“a”并单击右边按钮可以得到如图 7-43 所示的界面，同时在图形窗口可以得到如图 7-44 所示的图形，在命令窗口可以看到如下结果，并且工作空间增加了 `ans` 变量：

```
ans =
Columns 1 through 11
    3.1148   -4.3701   -0.2851    2.3156   -6.7610   -0.5820    1.7429   -13.5994   -0.9046
    1.2967  -451.9017
Columns 12 through 20
   -1.2717    0.9260   14.4892   -1.7120    0.6013    6.9878   -2.2746    0.3032    4.4743
```

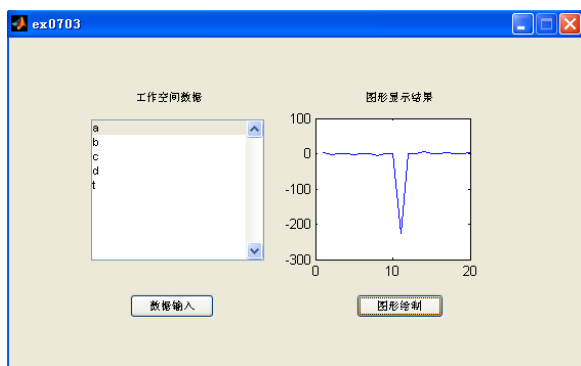


图 7-43 示例运行界面 1

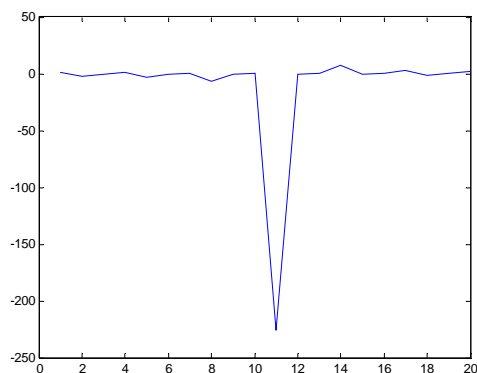


图 7-44 图形窗口运行结果

再单击左边按钮可以得到如图 7-45 所示的界面，增加了变量 `ans`。

单击“工作空间数据”框里的“c”，得到如图 7-46 所示的界面。

7.2.6 GUI 与 Simulink 仿真的数据交互

这里提到的 GUI 与 Simulink 仿真的数据交互包括如下两个含义：

(1) 通过 GUI 设置 Simulink 仿真参数

设置 Simulink 仿真参数包括环境参数、模块参数、子系统内的模块参数和封装子系统参数等情况。

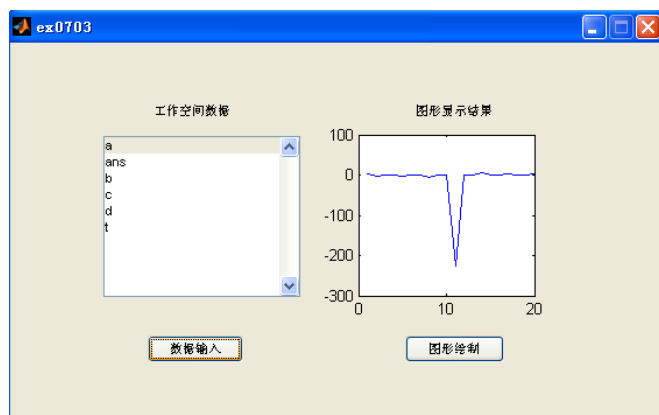


图 7-45 示例运行界面 2

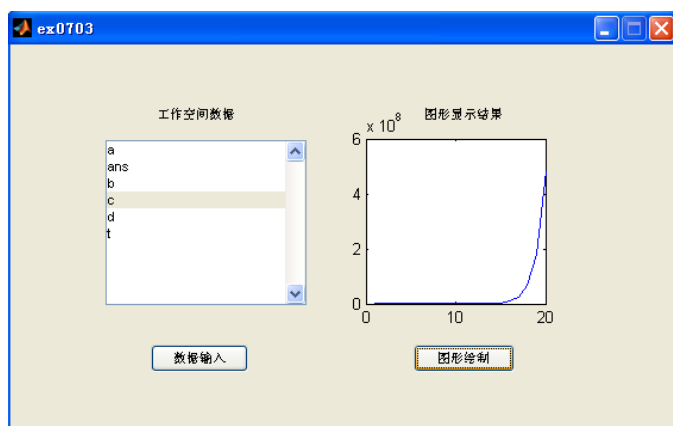


图 7-46 示例运行界面 3

设置环境参数（如仿真终止时间），可以利用 MATLAB 工作空间传递参数实现，因为 Simulink 也可以利用工作空间的数据。

设置模块参数和子系统参数，可以通过向 MATLAB 工作空间传递参数实现，也可以通过下述方法实现：

```
%打开 Simulink 文件
open_system('SimFilename')
%为按相对路径指定的模块的指定参数赋值
set_param('SimFilename/.../Blockname','Fieldname', Value)
%保存 Simulink 文件
save_system('SimFilename')
```

设置封装子系统参数，通过下述方法实现：

```
%打开 Simulink 文件
open_system('SimFilename')
%为按相对路径指定的封装子系统的指定参数赋值
set_param('SimFilename/Subsystemname','Paraname', Value);
%保存 Simulink 文件
save_system('SimFilename')
```

(2) 通过 GUI 调用 Simulink 仿真结果

当使用 ToWorkspace 模块时，其中的数据将传递到工作空间，GUI 直接调用即可。

【例 7-4】实现 GUI 与 Simulink 仿真的数据交互。具体步骤如下：

(1) 利用空白模板创建如图 7-47 所示的界面，并保存为 ex0704.fig。

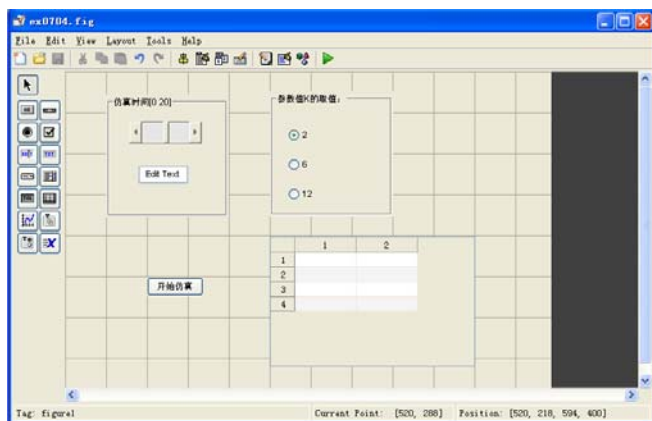


图 7-47 示例界面

(2) 利用 Simulink 环境搭建如图 7-48 所示的模型，并保存为 ex07041.mdl。

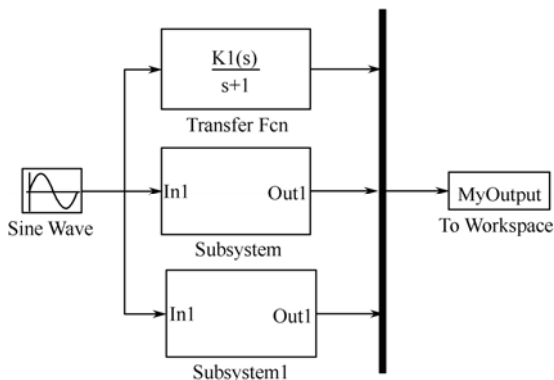


图 7-48 Simulink 模型

双击 Transfer Fcn 模块、Subsystem 子系统和 Subsystem1 封装子系统，可以分别看到如图 7-49~图 7-51 所示的界面，其中都包括增益变量（仿真时取相同的值），图 7-50 和图 7-52 表示被封装的子系统，图 7-53 表示封装子系统的参数设置，需选中 Evaluate 属性。同时单击“Simulation”→“Configuration Parameters”菜单命令，可以看到如图 7-54 所示的参数配置界面，其中包括变量 tend。运行结果送入数组 MyOutput 中。

(3) 利用属性编辑器将左上方面板的 Tag 属性值手工改为“Pan”，Title 属性值手工改为“仿真时间[0 20]”，其中滚动条的 Tag 属性值手工改为“Sli”，Max 属性值手工改为“20”，Min 属性值手工改为“0”，Value 属性值手工改为“10”，文本编辑框的 Tag 属性值手工改为“ET”；右边按钮组的 Tag 属性值手工改为“BG”，Title 属性值手工改为“参数值 K 的取值:”，其中三个单选框由上至下 Tag 属性值手工改为“RB1”、“RB2”和“RB3”，String 属性值手工改为“K=2”、“K=6”和“K=12”；按钮的 Tag 属性值手工改为“PB”，String 属性值手工改为“开始仿真”；表格的 Tag 属性值手工改为“Ta”。

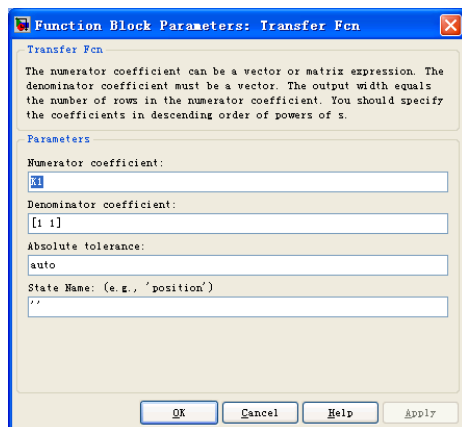


图 7-49 双击 Transfer Fcn 的界面

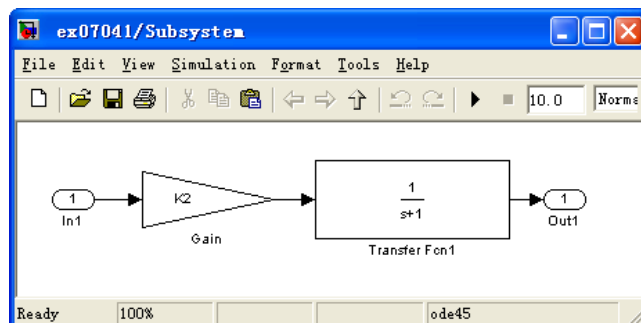


图 7-50 双击 Subsystem 的界面

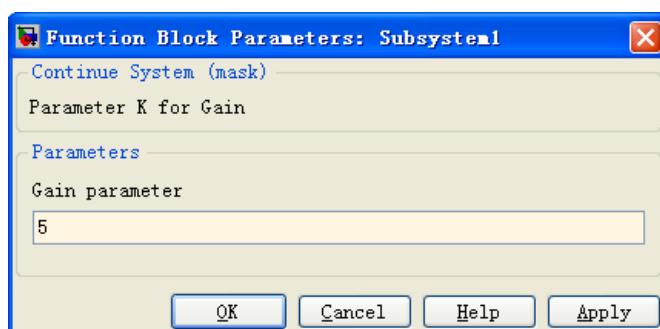


图 7-51 双击 Subsystem1 的界面

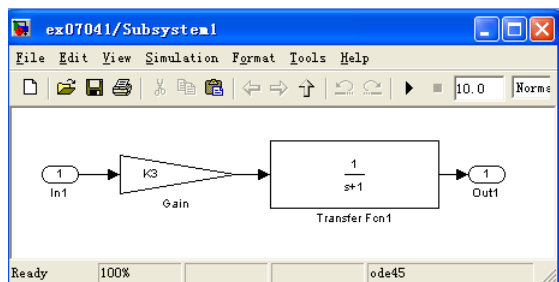


图 7-52 被封装的子系统

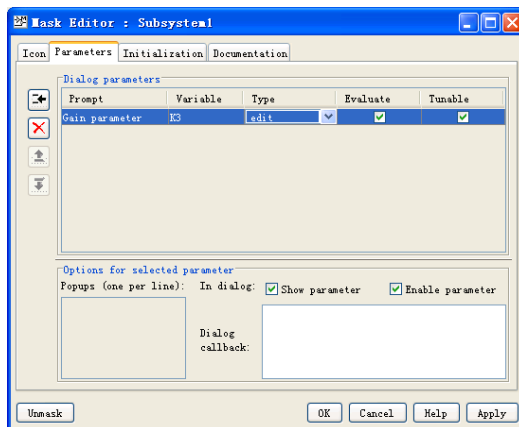


图 7-53 封装子系统的参数设置

(4) 设置 OpeningFcn 子函数，实现初始化，包括使文本编辑框显示的数据与滚动条一致，将单选框 RB2 作为初始选择，设参数 K 的值为 6，将表格数据置空。子函数的内容如下所示：

```
function ex0704_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
handles.output = hObject;
```

```
set(handles.ET,'String',get(handles.Sli,'Value'))
```

```

set(handles.BG,'SelectedObject',handles.RB2)
handles(paraK=6;
set(handles.Ta,'Data',[])

guidata(hObject, handles);

```

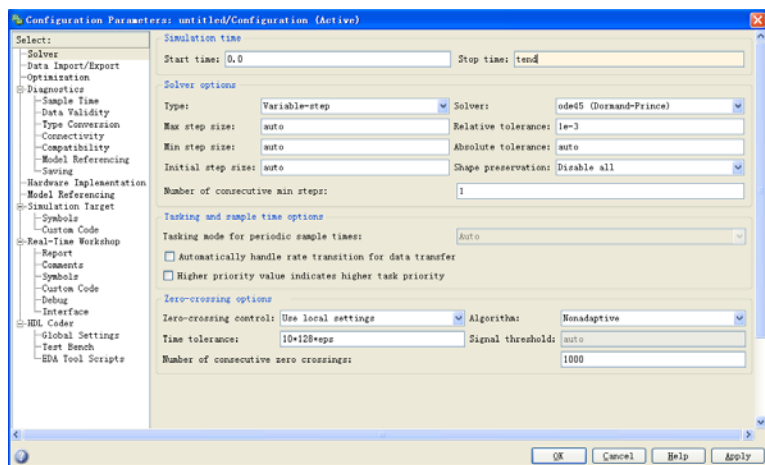


图 7-54 参数配置界面

(5) 设置滚动条的 Callback 回调函数, 实现将其值写入文本编辑框。回调函数的内容如下所示:

```
function Sli_Callback(hObject, eventdata, handles)
```

```
set(handles.ET,'String',get(handles.Sli,'Value'))
```

(6) 设置按钮的 Callback 回调函数, 实现读入仿真时间和参数 K, 输出参数并将其赋给 Simulink 模型, 运行后将数据的最后 10 行显示在表格中。回调函数的内容如下所示:

```
function PB_Callback(hObject, eventdata, handles)
```

```

tendvalue=get(handles.Sli,'Value');
evalin('base',['tend=',num2str(tendvalue)]);

```

```
switch get(handles.BG,'SelectedObject')
```

```
case handles.RB1
```

```
handles(paraK=2;
```

```
case handles.RB2
```

```
handles(paraK=6;
```

```
case handles.RB3
```

```
handles(paraK=12;
```

```
end
```

```
K=handles(paraK;
```

```
evalin('base',['K1=',num2str(K)]);
```



```
evalin('base',['K2=',num2str(K)]);
```

```
K3=K
```

```
open_system('ex07041')
```

```
set_param('ex07041/Subsystem1','K3',num2str(K3));
```

```
save_system('ex07041')
```

```
close_system('ex07041')
```

```
sim('ex07041');
```

```
set(handles.Ta,'Data',MyOutput(end-9:end,:))
```

(7) 激活设计的界面，如图 7-55 所示。拖动滚动条并选择参数值后，单击“开始仿真”按钮可以得到如图 7-56 所示的界面。

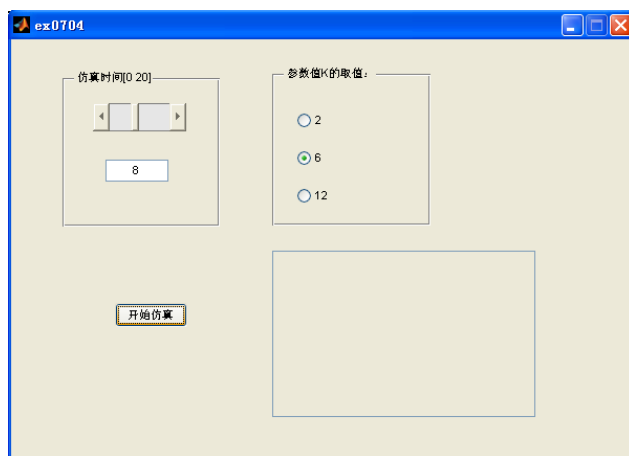


图 7-55 激活界面

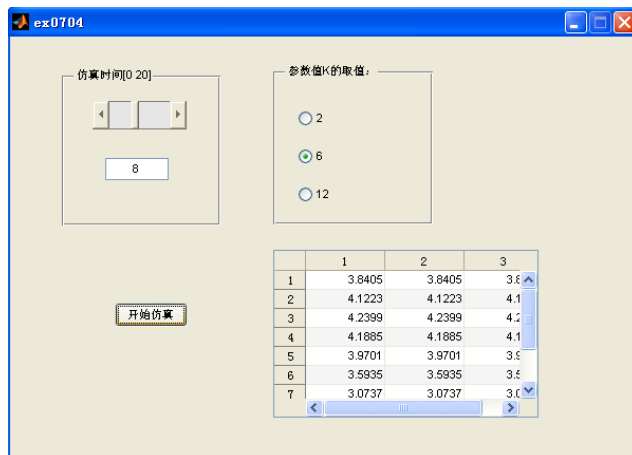


图 7-56 示例运行界面 1

(8) 在文本编辑框中修改数值为 10，并选择 K=12，单击“开始仿真”按钮可以得到如图 7-57 所示的界面。

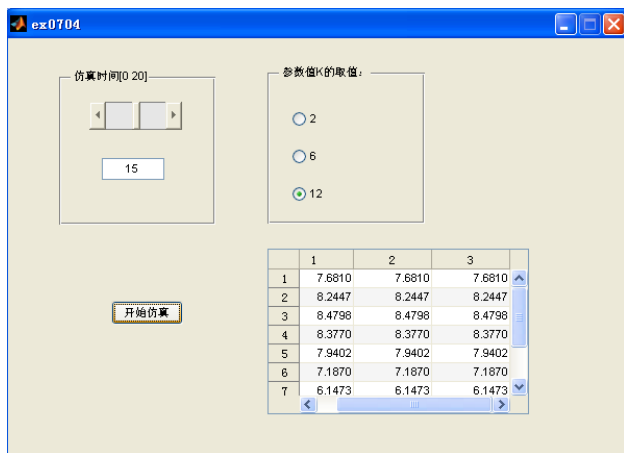


图 7-57 示例运行界面 2

7.2.7 中断执行

默认情况下, MATLAB 允许正在执行的回调函数被之后调用的回调函数中断。例如, 我们建立一个对话框, 用来作为加载数据的进度指示器。对话框中有一个“Cancel”命令按钮, 它能够停止加载操作。“Cancel”命令按钮的响应程序将中断当前正在运行的响应程序。

所有对象都有控制其回调函数能否被中断的属性 `Interruptible`, 默认值为 `on`, 表示回调函数可以中断。MATLAB 只有在遇到命令 `drawnow`、`figure`、`getframe`、`pause` 和 `waitfor` 时才会执行中断, 转而查询事件序列, 否则将会执行完回调函数。需要说明的是, 图形窗口的重画、`CloseRequestFcn` 和 `ResizeFcn` 事件, 对象的 `DeleteFcn` 和 `CreatFcn` 事件可以任意中断回调函数。

当回调函数被中断时, MATLAB 先将该回调函数挂起, 然后处理事件序列中的事件。同时所有对象都具有一个 `BusyAction` 属性, 该属性决定了不允许中断的回调函数的处理方式: 一是将新事件加入事件序列, 等待当前回调函数执行完毕再处理; 二是直接舍弃新事件。

【例 7-5】实现回调函数的中断。具体步骤如下:

(1) 利用空白模板创建如图 7-58 所示的界面, 并保存为 `ex0705.fig`。

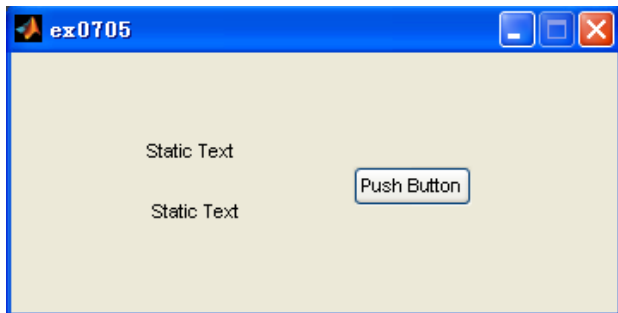


图 7-58 示例界面

(2) 利用属性编辑器将两个静态文本的 `Tag` 属性值手工改为“STU”(上方)和“STD”(下方), `String` 属性值手工改为“按钮事件状态”(上方)和“鼠标事件状态”(下方); 按钮的 `Tag` 属性值手工改为“PB”, `String` 属性值手工改为“测试”。

(3) 设置 `OpeningFcn` 子函数, 实现初始化, 包括保存初始值以及设置标志。子函数的内容如下所示:

```
function ex0705_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

handles.strU=get(handles.STU,'String');
handles.strD=get(handles.STD,'String');
handles.strflag=0;

guidata(hObject, handles);
```

(4) 设置按钮的 Callback 回调函数，实现中断的条件，并且显示中断的过程。回调函数的内容如下所示：

```
function PB_Callback(hObject, eventdata, handles)

handles.strflag=handles.strflag+1;
guidata(hObject, handles);

if handles.strflag==1
    newstrU=[handles.strU,'启动'];
    set(handles.STU,'String',newstrU);

    newstrD=[handles.strD,'未启动'];
    set(handles.STD,'String',newstrD);
    pause

    newstrU=[handles.strU,'结束'];
    set(handles.STU,'String',newstrU);

    newstrD=[handles.strD,'结束'];
    set(handles.STD,'String',newstrD);
end
```

(5) 设置图形窗口的 WindowButtonMotionFcn 回调函数，实现中断处理。回调函数的内容如下所示：

```
function figure1_WindowButtonMotionFcn(hObject, eventdata, handles)
if handles.strflag==1
    newstrU=[handles.strU,'中断'];
    set(handles.STU,'String',newstrU);

    newstrD=[handles.strD,'启动'];
    set(handles.STD,'String',newstrD);
end
```

(6) 激活设计的界面, 如图 7-59 所示。

使用鼠标在图形窗口上滑动时界面无变化, 单击“启动测试”按钮可以得到如图 7-60 所示的界面, 按下键盘任意键可以得到如图 7-61 所示的界面。

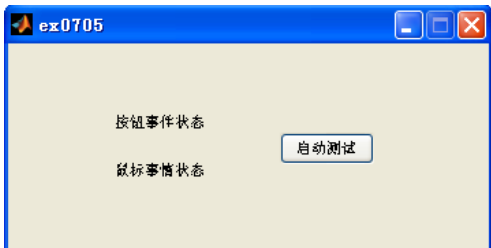


图 7-59 激活界面

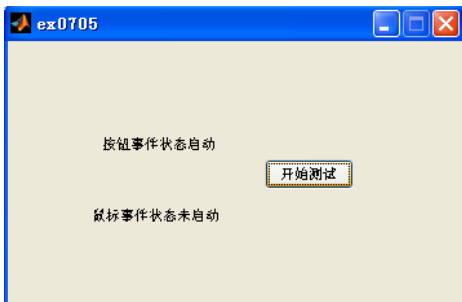


图 7-60 示例运行界面 1

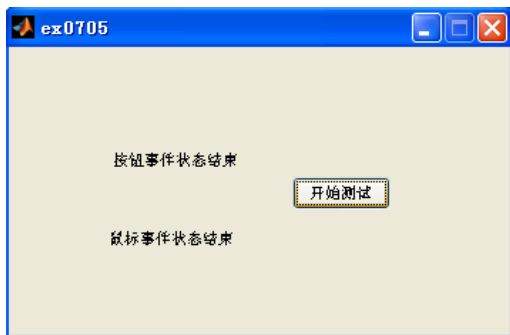


图 7-61 示例运行界面 2

7.2.8 多界面实例

前面的例子都是对于单界面的操作, 下面给出一个多界面的例子。在一个界面中需要调用其他界面时, 可以在某个回调函数中采用如下语句:

`figname % 界面对应的图形文件名`

需要关闭一个界面时, 只需在某个回调函数中采用如下语句:

`delete(handles.figurename) % figurename 为图形窗口的 Tag 属性值`

【例 7-6】实现多界面的操作。具体步骤如下:

(1) 利用空白模板创建如图 7-62 所示的界面, 并保存为 `ex0706.fig`。单击“Tools”→“GUI Options”菜单命令, 设置参数如图 7-63 所示, 即窗口大小可调整, 允许同时运行多个实例 (不设置此项, 将无法出现两个 `ex0706` 的实例)。

(2) 利用菜单编辑器创建如图 7-64 所示的菜单, 利用工具条编辑器创建如图 7-65 所示的工具条, 利用属性查看器得到图形窗口的 Tag 属性值为“figure1”。

(3) 利用空白模板创建如图 7-66 和图 7-67 所示的界面, 并分别保存为 `ex07061.fig` 和 `ex07062.fig`。

(4) 对于 `ex07061.fig`, 利用属性查看器得到图形窗口的 Tag 属性值为“figure1”, 设置静态文本的 String 属性值为“EX07061”, 按钮的 String 属性值为“退出”。对于 `ex07062.fig`, 利用属性查看器得到图形窗口的 Tag 属性值为“figure1”, 设置静态文本的 String 属性值为“EX07062”, 按钮的 String 属性值为“调用并退出”。

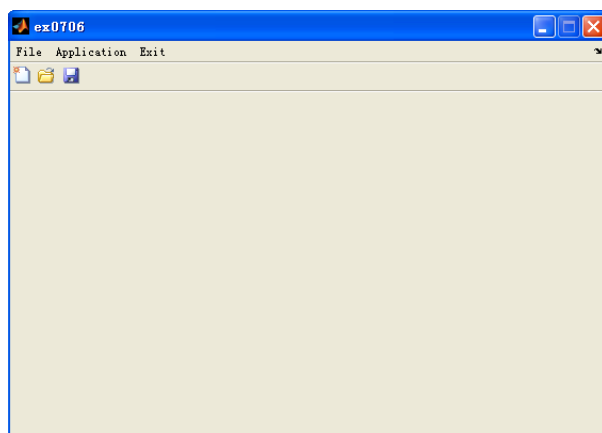


图 7-62 示例界面

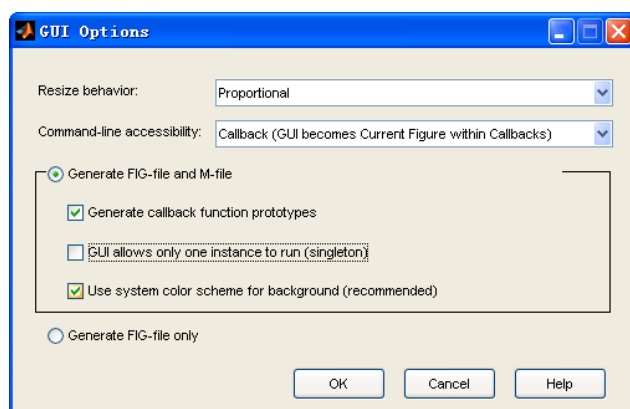


图 7-63 GUI 参数设置

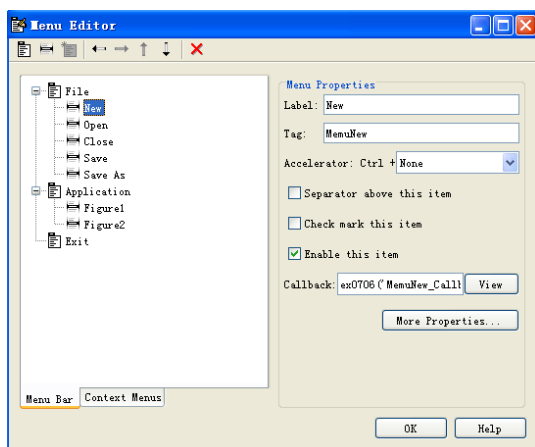


图 7-64 菜单设置

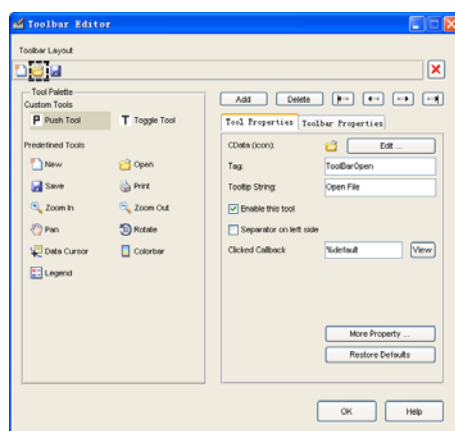


图 7-65 工具条设置

(5) 对于 ex0706.fig, 设置菜单命令“Application”→“Figure1”的 Callback 回调函数, 实现对 ex07061.fig 的调用和对自身的再次调用。回调函数的内容如下所示:

```
function MenuFigure1_Callback(hObject, eventdata, handles)
ex07061
ex0706
```

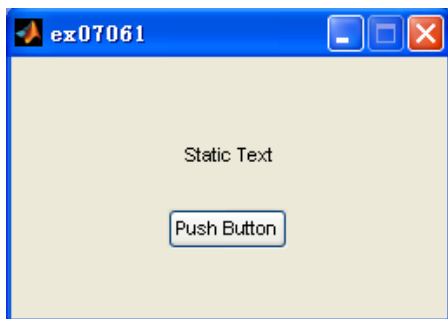


图 7-66 示例界面

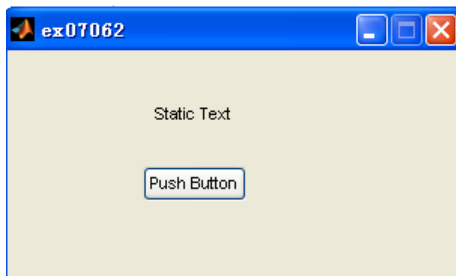


图 7-67 示例界面

设置菜单命令“Application”→“Figure2”的 Callback 回调函数，实现对 ex07062.fig 的调用。回调函数的内容如下所示：

```
function MenuFigure2_Callback(hObject, eventdata, handles)
ex07062
```

设置菜单命令“Exit”的 Callback 回调函数，实现关闭 ex0706.fig。回调函数的内容如下所示：

```
function MenuExit_Callback(hObject, eventdata, handles)
delete(handles.figure1)
```

（6）对于 ex07061.fig，设置按钮的 Callback 回调函数，实现对 ex07062.fig 的调用和自身的退出。回调函数的内容如下所示：

```
function pushbutton1_Callback(hObject, eventdata, handles)
ex07062
delete(handles.figure1)
```

（7）对于 ex07062.fig，设置按钮的 Callback 回调函数，实现自身的退出。回调函数的内容如下所示：

```
function pushbutton1_Callback(hObject, eventdata, handles)
delete(handles.figure1)
```

（8）激活 ex0706.fig 界面，如图 7-68 所示，可以改变图形窗口大小。选择菜单命令“Application”→“Figure2”后并移动图形窗口，可以得到如图 7-69 所示的界面。



图 7-68 激活界面



图 7-69 示例运行界面 1

激活 ex07061.fig 界面，如图 7-70 所示。

单击 ex07061 中的“退出”按钮，可以得到如图 7-71 所示的界面。



图 7-70 激活界面



图 7-71 示例运行界面 2

选择 ex0706 中的菜单命令“Exit”，可以得到如图 7-72 所示的界面，再选择 ex0906 中的菜单命令“Exit”，屏幕上将没有图形窗口。

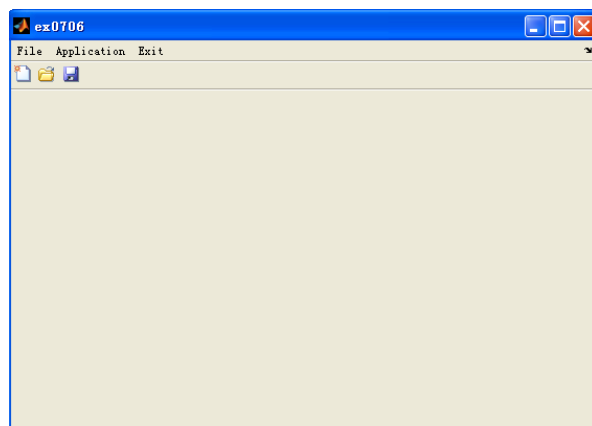


图 7-72 示例运行界面 3

7.3 GUI 应用

通过前面两节的介绍，特别是第 7.2 节中的例子，了解到使用 GUIDE 可以设计出界面精美和功能强大的 GUI。尽管与其他编程软件相比还有一点差距，如控件组中没有常见的下拉框、组合框等，但可以通过现有控件的组合和代码的编写实现相应功能。MATLAB 主要用于科学计算和模型仿真，现有的 GUI 设计能力应该能够满足要求。

下面首先介绍 GUI 设计的一般步骤，其次通过一个例子给出 GUI 对应的完整 M 文件。

7.3.1 GUI 设计的一般步骤

前面 7.2 节中的各个例子也体现了 GUI 设计的一般步骤，主要包括：

(1) 利用指定模板创建初始界面，在界面上布局控件、菜单和工具条，可以充分利用 MATLAB 提供的界面设计器、菜单编辑器与工具条编辑器设计出精美的界面。

(2) 利用属性编辑器和菜单编辑器与工具条编辑器为每个对象赋予属性值，最重要的属性是 Tag，它将作为该对象的标识出现在对象浏览器和 M 文件编辑器中。

(3) 利用 M 文件编辑器编写初始化函数、结束自函数、对象回调函数以及使用到的子函数，设计出具有强大功能的 GUI。

(4) 利用 M 文件的调试方法得到正常运行的 GUI。

7.3.2 GUI 设计实例

本小节利用一个简单的例子，详细讲解 GUI 的设计过程。

【例 7-7】初级 GUI 编程实例：使用 GUIDE 进行界面设计，绘制分析信号的频率和周期的图形。

- 建立 6 个静态文本，用于显示函数和标注相应控件的提示。
- 建立两个坐标轴对象，用于显示周期和时间的图形。
- 建立 3 个文本编辑框，用于输入数据。
- 建立一个按钮用于绘制图形。

针对上述步骤，完成的界面如图 7-73 所示，保存为 singal.fig 文件。

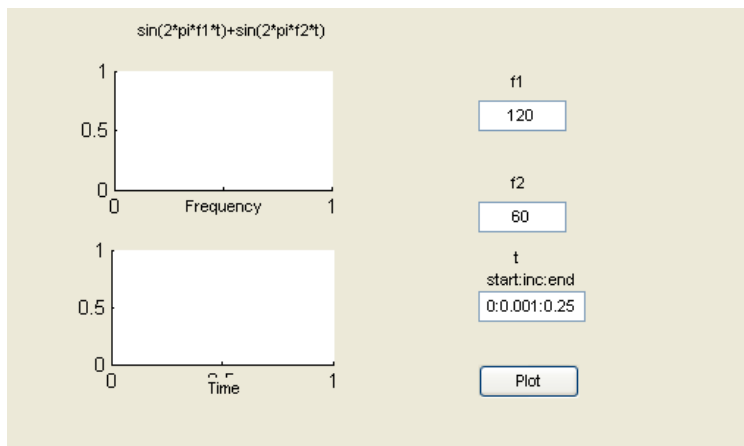


图 7-73 示例界面 1

设置控件的相关属性。每个控件在创建时都会由开发环境自动产生一个标识 (Tag)，在程序设计中，为了编辑、记忆和维护的方便，一般为控件设置一个新的标识。

- 设置第一个坐标轴的标识为 frequency_axes，用于显示频率图形，如图 7-74 所示。

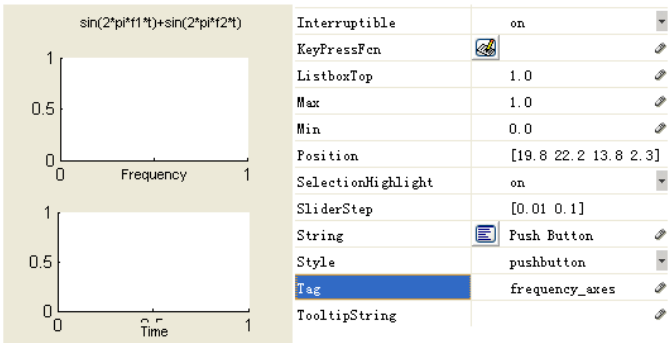


图 7-74 坐标轴的设置

- 设置第二个坐标轴的标识为 time_axes。用于显示时域图形。
- 三个文本编辑框的标识为 f1_input、f2_input 和 t_input，分别用于输入两个频率和自变量时间的间隔，如图 7-75 所示。

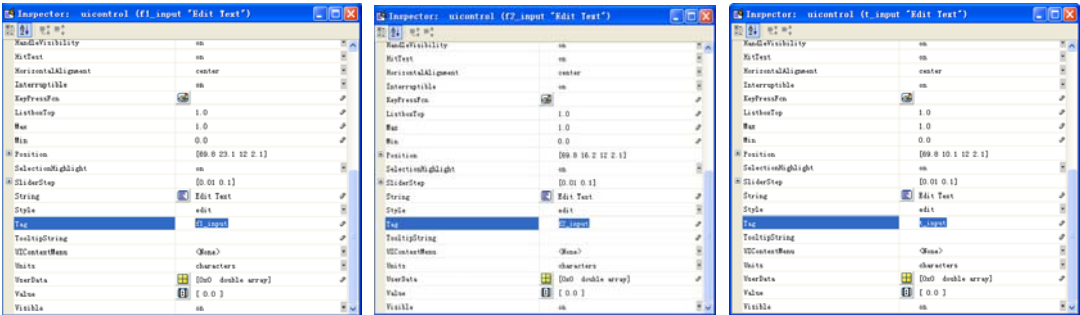


图 7-75 文本编辑框的设置

另外，设置其他几个静态文本控件和按钮的名称。单击“Property Inspector”→“String”菜单命令，设置显示名称，如图 7-76 所示。

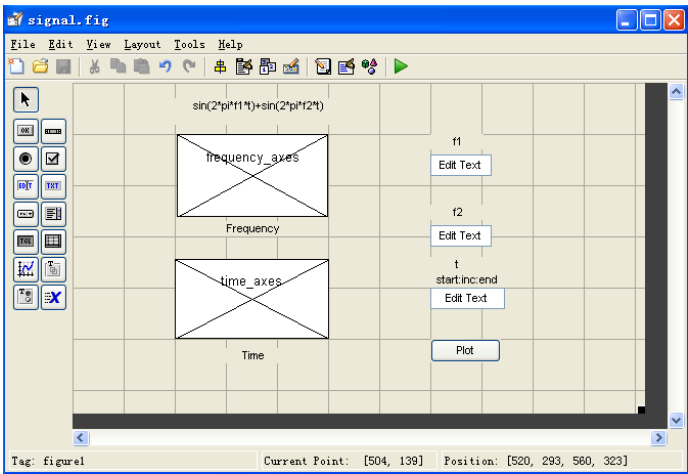


图 7-76 示例界面 2

添加菜单，如图 7-77 所示。

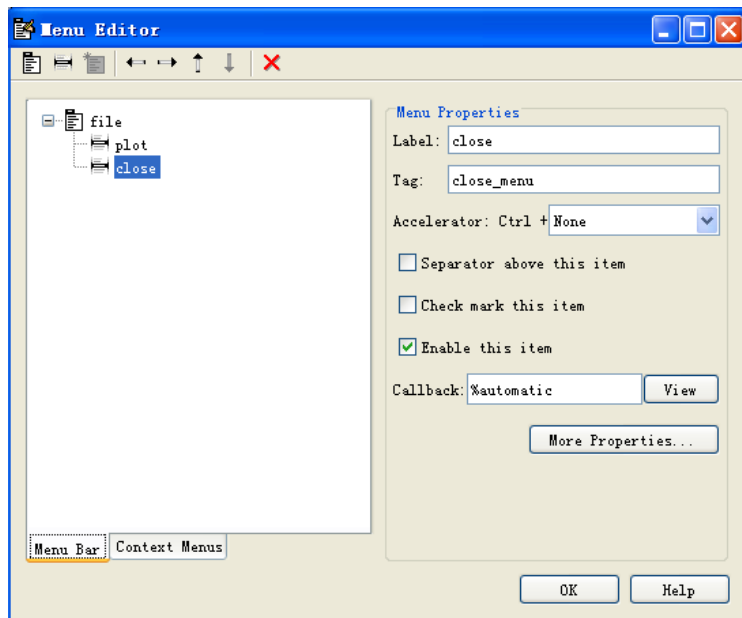


图 7-77 添加菜单

如图 7-77 所示，建立一级菜单 file，设置两个子菜单项 plot 和 close。

- 子菜单项 plot 的 Tag 设置为 “plot_menu”，调用绘图功能。
- 子菜单项 close 的 Tag 设置为 “close_menu”，执行关闭图形的功能。

编写代码完成程序中变量的赋值、输入、输出以及绘图等工作，打开 signal.m 文件，生成的代码如下所示：

```
function varargout = ex0708(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...           %GUI 结构
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ex0708_OpeningFcn, ...
                  'gui_OutputFcn',  @ex0708_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

function ex0708_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = ex0708_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
    function f1_input_Callback(hObject, eventdata, handles)
function f1_input_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function f2_input_Callback(hObject, eventdata, handles)
function f2_input_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function t_input_Callback(hObject, eventdata, handles)
function t_input_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Plot_pushbutton_Callback(hObject, eventdata, handles)
% -----
function file_menu_Callback(hObject, eventdata, handles)
% -----
function plot_menu_Callback(hObject, eventdata, handles)
% -----
function close_menu_Callback(hObject, eventdata, handles)

```

调用 `plot_pushbutton_Callback` 执行绘制图形的功能，函数代码如下：

```

function plot_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to plot_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f1=str2double(get(handles.f1_input,'String'));
f2=str2double(get(handles.f2_input,'String'));
t=eval(get(handles.t_input,'String'));
x=sin(2*pi*f1*t)+sin(2*pi*f2*t); % 计算数据
y=fft(x,512)
m=y.*conj(y)/512;
f=1000*(0:256)/512;
axes(handles.frequency_axes) % 选择显示周期的坐标轴

```

```

plot(f,m(1:257))
set(handles.frequency_axes,'XMinorTick','on')
grid on
axes(handles.time_axes) %选择显示时间的坐标轴
plot(t,x)
set(handles.time_axes,'XMinorTick','on')
grid on

```

调用 `f1_input_CreateFcn` 设置编辑框的初始值为 “120”，函数代码如下：

```

function f1_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f1_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
set(gcbo,'String','120')
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

调用 `f2_input_CreateFcn` 设置编辑框的初始值为 “60”，函数代码如下：

```

function f2_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f2_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
set(gcbo,'String','60')
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

调用 `t_input_CreateFcn` 设置编辑框的初始值为 “0:0.001:0.25”，函数代码如下：

```

function t_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
set(gcbo,'String','0:0.001:0.25')

```

```

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

调用 `close_menu_Callback` 结束程序，函数代码如下：

```

function close_menu_Callback(hObject, eventdata, handles)
% hObject    handle to close_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Close

```

在上述程序中，`f1_input_CreateFcn`、`f2_input_CreateFcn` 和 `t_input_CreateFcn` 在创建图形的时候依次给三个编辑文本框输入参数，`plot_pushbutton_Callback` 执行绘制图形的功能。

运行程序后，单击“Plot”按钮绘制图形，如图 7-78 所示。

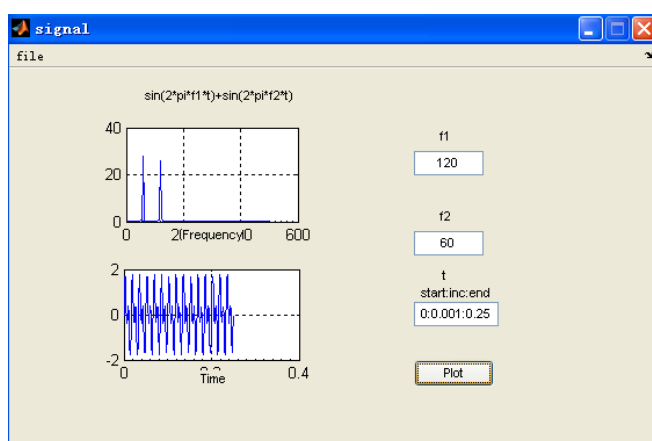


图 7-78 示例运行界面 1

可以在编辑框中改变频率和时间的数值，单击菜单命令“file”→“plot”，绘制的图形如图 7-79 所示。

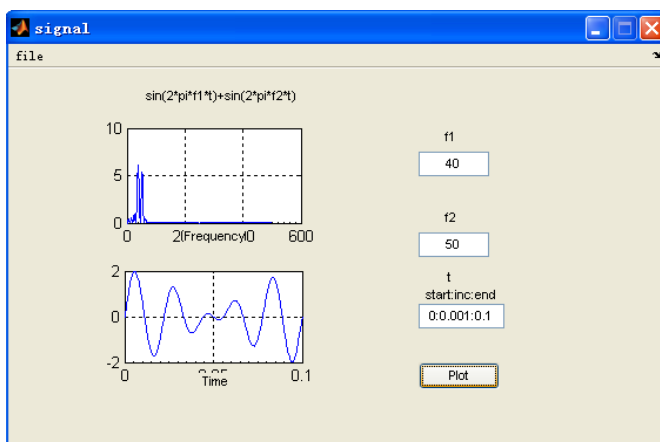


图 7-79 示例运行界面 2

第二篇 提高篇

第 8 章 MATLAB 科学计算

本章主要介绍经常用到的 MATLAB 科学计算问题的求解方法，其中包括线性方程与非线性方程以及常微分方程的求解、数据统计处理、数据插值、数值积分以及优化问题求解等方面的内容。

8.1 方程求解

本节将分别讨论线性方程组、非线性方程和常微分方程三种常见方程的解法。

8.1.1 线性方程组

线性方程组是线性代数中的主要内容之一，也是理论发展最为完整的部分。在 MATLAB 中也包含多种处理线性方程组的命令，下面详细进行介绍。

1. 问题描述

在实际应用中，经常需要求解如下所示的两类线性方程组，其中第一种更常见。

$$AX = B$$

$$XA = B$$

按照数学的严格定义，并没有矩阵除法的概念，而 MATLAB 为了书写简便提供了用除号求解线性方程组解的方式，其具体用法如下：

- $X=A \setminus B$ ：左除，计算方程组 $AX=B$ 的解。
- $X=B/A$ ：右除，计算方程组 $XA=B$ 的解。

下面针对 $AX = B$ 的形式进行说明。系数矩阵 A 是 $m \times n$ 的矩阵，根据其维数可以分为如下 3 种情况：

- $m=n$ 为恰定方程组，即方程数等于未知量数。
- $m>n$ 为超定方程组，即方程数大于未知量数。
- $m<n$ 为欠定方程组，即方程数小于未知量数。

线性方程组解的类型也可以分为 3 种情况：

- $\text{rank}(A)=\text{rank}([A,B])$ 且对应齐次方程组 $AX=0$ 不存在非 0 解，则方程组有唯一解，如矩阵

A 可逆。

- $\text{rank}(A)=\text{rank}([A,B])$ 且对应齐次方程组 $AX=0$ 存在非 0 解, 则方程组有无穷解, 如 $\text{rank}(A)=\text{rank}([A,B])$ 且为欠定方程组。
- $\text{rank}(A)=\text{rank}([A,B])$, 则方程组无解。

不难看出, 线性方程组解的类型是由对应齐次方程组的解、对应系数矩阵和增广矩阵间的关系决定的。

2. 解的形式

线性方程组 $AX=B$ 解的形式可以如下描述:

- 首先可以使用 `null` 函数求解对应齐次方程组 $AX=0$ 的基础解系, 也可称为通解, 则 $AX=B$ 的解都可以通过通解的线性组合表示。
- 其次求解非齐次线性方程组 $AX=B$ 的特解。
- 最后非齐次线性方程组 $AX=B$ 解的形式为通解的线性组合加上特解。

3. 除法及求逆的解法

(1) 除法解法

若线性方程组 $AX=B$ 的系数矩阵可逆, 则 $A \setminus B$ 给出方程组的唯一解。

【例 8-1】使用除法求解系数矩阵可逆的恰定线性方程组。

在命令窗口中输入如下语句:

```
A=pascal(4)
det_A=det(A)
B=rand(4,1)
X1=A\B
X2=inv(A)*B
```

命令窗口中的输出结果如下所示:

```
A =
    1     1     1     1
    1     2     3     4
    1     3     6    10
    1     4    10    20

det_A =
    1.0000

B =
    0.4218
    0.9157
    0.7922
    0.9595

X1 =
```

```

-1.5980
  4.4539
-3.3424
  0.9083

X2 =
-1.5980
  4.4539
-3.3424
  0.9083

```

本例需要说明的是， $\det(A)$ 用于计算矩阵 A 的行列式，不难看出矩阵 A 可逆；由于矩阵 A 是 4×4 ，所以矩阵 B 必须是 4×1 ，例中随机生成矩阵的语句是 $B=\text{rand}(4,1)$ ，而非 $B=\text{rand}(1,4)$ ； $A \setminus B$ 等价于 $\text{inv}(A)*B$ 。

若线性方程组 $AX = B$ 的系数矩阵不可逆，则方程组的解不存在或者不唯一。此时，执行 $A \setminus B$ 可能给出警告信息和不恰当的解。

【例 8-2】 使用除法求解系数矩阵不可逆的恰定线性方程组。

在命令窗口中输入如下语句：

```

A=[1 3 7;-1 4 4;1 10 18];
det_A=det(A)
B=[6;4;15];
X=A\B

```

命令窗口中的输出结果如下所示：

```

det_A =
    0

Warning: Matrix is singular to working precision.
X =
    NaN
    Inf
   -Inf

```

从以上的结果可以看出，MATLAB 会显示提示信息，表示该矩阵是奇异矩阵，因此无法得到精确的数值解。

【例 8-3】 使用除法求解欠定线性方程组。

在命令窗口中输入如下语句：

```

C=magic(4);
A=C(1:3,:);
B=[1;0;0];
X=A\B

```

命令窗口中的输出结果如下所示：

```

A =

```



```
16    2    3    13
    5   11   10    8
    9    7    6   12

X =
    0.1863
    0.0294
         0
   -0.1569
```

【例 8-4】使用除法求解超定线性方程组。

在命令窗口中输入如下语句：

```
T=magic(5)
A= T(:,1:4)
B=[1;0;0;0;0];
X=A\B
```

命令窗口中的输出结果如下所示：

```
T =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

A =
    17    24     1     8
    23     5     7    14
     4     6    13    20
    10    12    19    21
    11    18    25     2

X =
   -0.0041
    0.0437
   -0.0305
    0.0060
```

(2) 求逆解法

在例 8-1 中，已经介绍了通过求逆的方法求解方程组的解，这里着重介绍伪逆的用法。对于系数矩阵而言，它可能是方阵但不可逆，也可能不是方阵，上述情况都导致它的逆不存在或无定义，这就需要引入伪逆的概念。伪逆包含很多种形式（详见矩阵的有关书籍），下面介绍最常用的基于最小二乘意义下的最优伪逆，在 MATLAB 中通过 `pinv` 函数可以实现，即可以使用矩阵 **A** 的伪逆矩阵 `pinv(A)` 来得到方程的一个解，其对应的数值解为 `pinv(A)*B`。

【例 8-5】使用伪逆矩阵的方法求解奇异矩阵的线性方程的解。

在命令窗口中输入如下语句：

```
A=[1 3 7;-1 4 4;1 10 18];
B=[5;2;12];
X=pinv(A)*B
C=A*X
```

命令窗口中的输出结果如下所示：

```
X =
    0.3850
   -0.1103
    0.7066

C =
    5.0000
    2.0000
   12.0000
```

从上面的结果可以看出，通过使用伪逆矩阵的方法，可以求解得到数值解，同时该数值解可以精确地满足结果。

上面都是介绍如何计算特解，下面介绍如何计算线性方程组的所有解。

【例 8-6】使用求逆法计算线性方程组的所有解。

在命令窗口中输入如下语句：

```
A=[1 2 3 4;5 6 7 8;9 10 11 12];
B=[1;1;2];
X1=null(A)
X2=pinv(A)*B
```

命令窗口中的输出结果如下所示：

```
X1 =
    0.3170    0.4467
   -0.7556   -0.3592
    0.5602   -0.6215
   -0.1216    0.5341

X2 =
   -0.1250
   -0.0208
    0.0833
    0.1875
```

此时线性方程组的所有解为 $X=a*X1(:,1)+b*X1(:,2)+X2$ ，其中 a 、 b 为任意实数。

4. 矩阵分解的解法

(1) LU 分解

LU 分解又可称为 Gauss（高斯）消去法。若系数矩阵为方阵，它可以表示为下三角矩阵

和上三角矩阵的乘积, 即 $A=LU$, 其中 L 为下三角阵, U 为上三角阵。在 MATLAB 中通过 `lu` 函数可以实现 LU 分解。

针对 LU 分解, 线性方程组 $AX=B$ 可以表示为 $LUX=B$, 由于 L 和 U 的特殊性, 通过 $X=U/(L/B)$ 求解可以大大提高运算速度。

LU 函数的具体语法形式如下:

- `[L,U]=lu(X)`: X 是任意方阵, L 是下三角阵, U 是上三角阵, 这三个变量满足的条件式为 $X=LU$ 。
- `[L,U,P]=lu(X)`: X 是任意方阵, L 是下三角阵, U 是上三角阵, P 是置换矩阵, 满足的条件式为 $PX=LU$ 。
- `Y=lu(X)`: X 是任意方阵, 把上三角矩阵和下三角矩阵合并在矩阵 Y 中给出, 满足的条件式为 $Y=L+U-I$, 但是将损失置换矩阵 P 的信息。

【例 8-7】 使用 LU 分解法求解线性方程组
$$\begin{cases} 2x_1 + x_2 + x_3 = 1 \\ x_1 - 2x_2 + 3x_3 = 5 \\ 6x_1 - 5x_2 + x_3 = 7 \end{cases}$$

在命令窗口中输入如下语句:

```
A=[2 1 1;1 -2 3;6 -5 1];
B=[1;5;7];
C=det(A)
[L,U]=lu(A)
X=U\ (L\B)
```

命令窗口中的输出结果如下所示:

```
C =
    50

L =
    0.3333    1.0000         0
    0.1667   -0.4375    1.0000
    1.0000         0         0

U =
    6.0000   -5.0000    1.0000
         0    2.6667    0.6667
         0         0    3.1250

X =
    0.3600
   -0.7600
    1.0400
```

【例 8-8】使用 LU 分解法求解线性方程组
$$\begin{cases} -x_1 + 8x_2 + 5x_3 = 2 \\ 9x_1 - x_2 + 2x_3 = 3 \\ 2x_1 - 5x_2 + 7x_3 = 5 \end{cases}。$$

在命令窗口中输入如下语句：

```
A=[-1 8 5;9 -1 2;2 -5 7];
B=[2;3;5];
C=det(A)
[L,U]=lu(A)
X=U\ (L\B)
```

命令窗口中的输出结果如下所示：

```
C =
-690.0000

L =
-0.1111    1.0000         0
 1.0000         0         0
 0.2222   -0.6056    1.0000

U =
 9.0000   -1.0000    2.0000
 0       7.8889    5.2222
 0         0     9.7183

X =

 0.1913
-0.0957
 0.5913
```

(2) Cholesky 分解

若系数矩阵为对称正定矩阵，可以表示为上三角矩阵和其转置的乘积，即 $\mathbf{A}=\mathbf{R}'\mathbf{R}$ ，其中 \mathbf{R} 为上三角矩阵， \mathbf{R}' 为下三角矩阵。在 MATLAB 中通过 chol 函数可以实现 Cholesky 分解。

从理论角度来讲，并不是所有的对称矩阵都可以进行 Cholesky 分解，可以进行 Cholesky 分解的矩阵必须是正定的。针对 Cholesky 分解，线性方程组 $\mathbf{AX}=\mathbf{B}$ 可以表示为 $\mathbf{R}'\mathbf{RX}=\mathbf{B}$ ，由于 \mathbf{R} 的特殊性，通过 $\mathbf{X}=\mathbf{R}/(\mathbf{R}'/\mathbf{B})$ 求解可以大大提高运算速度。

Chol 函数的具体语法形式如下。

- $\mathbf{R}=\text{chol}(\mathbf{X})$: \mathbf{X} 是对称的正定矩阵， \mathbf{R} 是上三角矩阵，使得 $\mathbf{X}=\mathbf{R}'\mathbf{R}$ 。如果矩阵 \mathbf{X} 是非正定矩阵，该命令会返回错误信息。
- $[\mathbf{R},\mathbf{p}]=\text{chol}(\mathbf{X})$: 该命令返回两个参数，并不返回错误信息。当 \mathbf{X} 是正定矩阵时，返回的矩阵 \mathbf{R} 是上三角矩阵，而且满足 $\mathbf{X}=\mathbf{R}'\mathbf{R}$ ，同时返回参数 $\mathbf{p}=0$ ；当 \mathbf{X} 不是正定矩阵时，返回的参数 \mathbf{p} 是正整数， \mathbf{R} 是三角矩阵，矩阵阶数是 $\mathbf{p}-1$ ，并且满足 $\mathbf{X}(1:\mathbf{p}-1,1:\mathbf{p}-1)=\mathbf{R}'\mathbf{R}$ 。

【例 8-9】使用 Cholesky 分解法求解线性方程组
$$\begin{cases} 2x_1 + x_2 + x_3 = 1 \\ x_1 - 5x_2 + 4x_3 = 3 \\ 3x_1 - 4x_2 + 6x_3 = 5 \end{cases}$$

在命令窗口中输入如下语句：

```
A=[2 1 3;1 5 4;3 4 6];  
B=[1;3;5];  
C=det(A)  
R=chol(A)  
TR=R'  
X=R\'(TR\B)
```

命令窗口中的输出结果如下所示：

```
C =  
1.0000  
  
R =  
1.4142    0.7071    2.1213  
0    2.1213    1.1785  
0    0    0.3333  
  
TR =  
1.4142    0    0  
0.7071    2.1213    0  
2.1213    1.1785    0.3333  
  
X =  
-23.0000  
-10.0000  
19.0000
```

【例 8-10】对对称正定矩阵进行 Cholesky 分解。

在命令窗口中输入如下语句：

```
n=5;  
X=pascal(n)  
R=chol(X)  
C=transpose(R)*R
```

命令窗口中的输出结果如下所示：

```
X =  
1    1    1    1    1  
1    2    3    4    5  
1    3    6    10   15  
1    4    10   20   35
```

```
1    5    15    35    70
```

```
R =
```

```
1    1    1    1    1
0    1    2    3    4
0    0    1    3    6
0    0    0    1    4
0    0    0    0    1
```

```
C =
```

```
1    1    1    1    1
1    2    3    4    5
1    3    6    10   15
1    4    10   20   35
1    5    15   35   70
```

由结果可以看出， R 是上三角矩阵，同时满足 $X = R'R = C$ ，表明上面的 Cholesky 分解过程是正确的。

如果修改矩阵的信息，在命令窗口中输入如下语句：

```
X(n,n)=X(n,n)-1
```

```
[R1,p]=chol(X)
```

```
C1=transpose(R1)*R1
```

```
C2=X(1:p-1,1:p-1)
```

命令窗口中的输出结果如下所示：

```
X =
```

```
1    1    1    1    1
1    2    3    4    5
1    3    6    10   15
1    4    10   20   35
1    5    15   35   69
```

```
R1 =
```

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

```
p =
```

```
5
```

```
C1 =
```

```

1      1      1      1
1      2      3      4
1      3      6     10
1      4     10     20

C2 =
1      1      1      1
1      2      3      4
1      3      6     10
1      4     10     20

```

由此可见，当原来的正定矩阵的最后一个元素减 1 后，矩阵将不是正定矩阵，并且满足 $X(1:p-1,1:p-1)=R'R$ 。

(3) QR 分解

对于任何系数矩阵，可以表示为正交矩阵和上三角矩阵的乘积，即 $A=QR$ ，其中 Q 为正交矩阵， R 为上三角矩阵。在 MATLAB 中通过 qr 函数可以实现 QR 分解。

针对 QR 分解，线性方程组 $AX=B$ 可以表示为 $QRX=B$ ，由于 Q 和 R 的特殊性，通过 $X=R/(Q/B)$ 求解可以大大提高运算速度。

qr 函数的具体语法形式如下：

- $[Q,R]=qr(A)$ ：矩阵 R 和矩阵 A 的大小相同， Q 是正交矩阵，满足 $A=QR$ ，该调用方式适合于满矩阵和稀疏矩阵。

【例 8-11】使用 QR 分解法求解线性方程组
$$\begin{cases} 2x_1 + x_2 + 3x_3 + 4x_4 = 1 \\ 1x_1 + 5x_2 + 4x_3 + 2x_4 = 3 \\ 3x_1 + 4x_2 + 6x_3 - 5x_4 = 5 \end{cases}$$

在命令窗口中输入如下语句：

```

A=[2 1 3 4;1 5 4 2;3 4 6 -5];
B=[1;3;5];
[Q,R]=qr(A)
X=R/(Q\B)

```

命令窗口中的输出结果如下所示：

```

Q =

0.5345    -0.4257    -0.7301
0.2673     0.9047    -0.3319
0.8018    -0.0177     0.5974

R =

3.7417    5.0780    7.4833   -1.3363
0         4.0267    2.2351    0.1951
0          0         0.0664   -6.5709

```

```
X =
    0
    0.2846
    0.4878
   -0.1870
```

5. 共轭梯度的解法

MATLAB 还提供了一系列的线性方程组 $AX=B$ 的共轭梯度解法，这里主要介绍双共轭梯度法，它可由 `bicg` 函数实现。`bicg` 函数要求系数矩阵 A 必须为方阵。

【例 8-12】使用共轭梯度法求解线性方程组
$$\begin{cases} 5x_1 + 3x_2 + x_3 = 7 \\ 2x_1 - 3x_2 + 4x_3 = 9 \\ x_1 - 7x_2 + 2x_3 = 1 \end{cases}$$

在命令窗口中输入如下语句：

```
A=[3 1 1;2 -1 5;8 -4 0];
B=[2;4;1];
[X,flag,relres,iter,resvec]=bicg(A,B)
```

命令窗口中的输出结果如下所示：

```
X =
    0.5253
    0.6364
    2.4646

flag =
    0

relres =
    3.2537e-015

iter =
    3

resvec =
    11.4455
    33.8479
    35.9786
    0.0000
```

其中，`flag` 表示在默认迭代次数内收敛，`relres` 表示相对残差 $\text{norm}(B-A*x)/\text{norm}(B)$ ，`iter` 表示终止的迭代次数，`resvec` 表示每次迭代的残差。

8.1.2 非线性方程

1. 函数的零点

对于任意函数，在求解范围内可能有零点，也可能没有；可能只有一个零点，也可能有多

个甚至无数个零点。MATLAB 没有可以求解所有函数零点的通用命令，本节将分别讨论一元函数和多元函数的零点求解问题。

(1) 一元函数的零点

在所有函数中，一元函数是最简单的，在 MATLAB 中可以使用 `fzero` 函数来计算一元函数的零点，它的具体使用方法如下：

- `x = fzero(fun,x0)`: 在 `x0` 点附近寻找函数 `fun` 的零点。
- `x = fzero(fun,x0,options)`: 使用 `options` 设定优化器参数。
- `x = fzero(fun,[x0, x1])`: 在 `[x0, x1]` 区间寻找函数 `fun` 的零点。

【例 8-13】计算一元函数 $f(x) = x^2 \sin x - x + 1$ 在 $[-3,4]$ 区间的零点。

首先绘制函数的曲线，在命令窗口中输入如下语句：

```
x=-3:0.1:4;
y= x.*x.*sin(x)-x+1;
plot(x,y,'r')
xlabel('x');
ylabel('f(x)');
title('The zero of function')
hold on
h=line([-3,4],[0,0]);
set(h,'color','g')
grid;
```

图形窗口中的输出结果如图 8-1 所示。

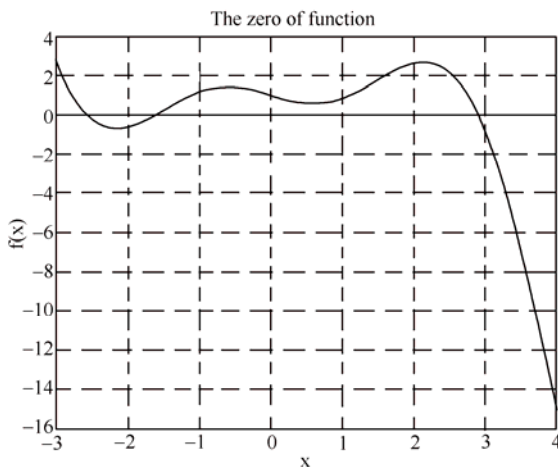


图 8-1 函数 $f(x)$ 的曲线

在求解函数零点之前，需要绘制函数的图形，是为了在后面的步骤中使用 `fzero` 命令时，更加方便地选择初始数值 `x0`。

由图 8-1 不难看出，曲线在 $[-3,4]$ 区间内包含 3 个零点。

其次计算函数某点附近的零点，在命令窗口中输入如下语句：

```
f = @(x) x-10*sin(x);
```

```
x1 = fzero(f,-2.5)
x2 = fzero(f,-1.5)
x3 = fzero(f,3)
```

命令窗口中的输出结果如下所示:

```
x1 =
    -2.5708

x2 =
    -1.6194

x3 =
     2.9142
```

求一元函数零点时, 可以使用 `optimset` 函数设置优化器参数, 它的具体使用方法如下:

- `optimset`: 显示优化器的现有参数名及其参数值。
- `options = optimset('param1',value1,'param2',value2,...)`: 使用参数名和参数值设定优化器的参数。
- `options = optimset(optimoptions,'param1',value1,...)`: 在现有优化器 `optimoptions` 的基础上, 使用参数名和参数值变更优化器参数。

`optimset` 函数中可以设置的主要优化器参数如表 8-1 所示。

表 8-1 优化器参数

参数名	有效参数值	功能描述
Display	'final'、'off'、'iter'和'notify'	'final': 只显示最终结果, 该选项为默认值 'off': 不显示计算结果 'iter': 显示每个迭代步骤的计算结果 'notify': 只在不收敛时显示计算结果
MaxFunEvals	正整数	最大允许的函数评估次数
MaxIter	正整数	最大允许的迭代次数
TolFun	正标量	函数值的截断阈值
TolX	正标量	自变量的截断阈值
OutputFcn	空矩阵或者用户定义函数句柄	空矩阵: 迭代过程采用 MATLAB 自带的函数 用户自定义函数句柄: 用该函数替换 MATLAB 自带的函数
FunValCheck	'off'和'on'	'off': 不检查输入函数的返回值, 该选项为默认值 'on': 如果输入函数的返回值为复数或者 NaN, 则显示警告信息

(2) 多元函数的零点

多元函数的零点问题比一元函数的零点问题更难解决, 但是当零点大致位置和性质比较好预测时, 也可以使用数值方法来搜索精确的零点。

非线性方程组的标准形式为 $F(x) = 0$, 其中 x 为向量, $F(x)$ 为函数向量。在 MATLAB 中, 求解多元函数的命令是 `fsolve`, 它的具体使用方法如下:

- `x = fsolve(fun,x0)`: 在向量 x_0 附近寻找函数 `fun` 的解。
- `x = fsolve(fun,x0,options)`: 使用 `options` 设定优化器参数。

其中 `options` 设定优化器参数的方法同前。

【例 8-14】求解二元方程组 $\begin{cases} 2x_1 - x_2 = e^{-x_1} \\ -x_1 + 2x_2 = e^{-x_2} \end{cases}$ 的零点。

首先绘制函数的曲线，在命令窗口中输入如下语句：

```
x=[-5:0.1:5];
y=x;
[X,Y]=meshgrid(x,y);
Z=2*X-Y-exp(-X);
surf(X,Y,Z)
xlabel('x')
ylabel('y')
zlabel('z')
title('The figure of the function')
```

图形窗口中的输出结果如图 8-2 所示。

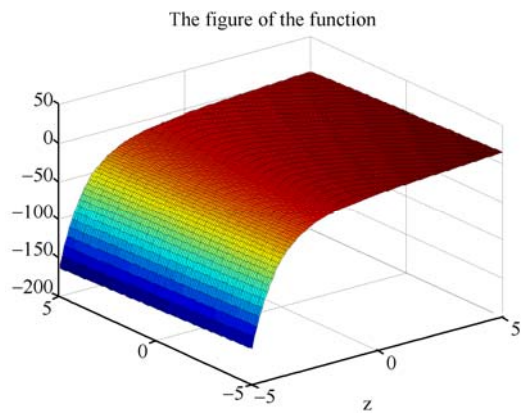


图 8-2 函数 $f(x)$ 的曲线

编写求解函数的 M 文件，在其中输入如下的程序代码：

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

将上述程序代码保存为 fsolvefun.m 文件。

下面求解二元函数的零点，在命令窗口中输入如下语句：

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

命令窗口中的输出结果如下所示：

Norm of		First-order	Trust-region		
Iteration	Func-count	f(x)	step	optimality	radius
0	3	47071.2		2.29e+004	1
1	6	12003.4	1	5.75e+003	1
2	9	3147.02	1	1.47e+003	1
3	12	854.452	1	388	1

4	15	239.527	1	107	1
5	18	67.0412	1	30.8	1
6	21	16.7042	1	9.05	1
7	24	2.42788	1	2.26	1
8	27	0.032658	0.759511	0.206	2.5
9	30	7.03149e-006	0.111927	0.00294	2.5
10	33	3.29525e-013	0.00169132	6.36e-007	2.5

Optimization terminated: first-order optimality is less than options.TolFun.

```
x =
    0.5671
    0.5671
```

由上面的结果可以看出，原来的二元函数是对称的，因此求解的未知数结果是相等的。

2. 非线性方程组的解

求非线性方程组的解的问题，也就是求多元函数的零点的问题。在 MATLAB 中使用 `fsolve` 函数计算非线性方程组的解，使用方法见上一小节。

【例 8-15】求解 $X^4 = \begin{bmatrix} 1 & 7 \\ -11 & 9 \end{bmatrix}$ 。

首先对非线性方程组进行函数描述，并保存为 `myfun8_15.m`，其内容如下所示：

```
function T = myfun8_15(x)
T = x^4 - [1,7;-11,9];
```

其次对非线性方程组进行求解，在命令窗口中输入如下语句：

```
x0 = [3 1;2 1];
options=optimset('Display','off');
x = fsolve(@myfun8_15,x0,options)
```

命令窗口中的输出结果如下所示：

```
x =
    1.4693    0.3875
   -0.6089    1.9121
```

8.1.3 常微分方程

1. 求解常微分方程的函数

在 MATLAB 中使用 `ode45`、`ode23`、`ode113`、`ode15s`、`ode23s`、`ode23t`、`ode23tb` 等函数求常微分方程（ODE）的数值解，这些函数的介绍如表 8-2 所示。它们的具体使用方法类似，为了方便后面的描述，这里用 `solver` 统一代替它们。

在介绍具体用法之前，首先介绍其涉及的参数，如表 8-3 所示。

其次介绍它们的具体用法，如下所述：

- `[T,Y] = solver(odefun,tspan,y0)`

在区间 `tspan=[t0,tf]` 上，使用初始条件 `y0` 求解常微分方程。常微分方程 `odefun` 解向量 `Y` 中的每行结果对应于时间向量 `T` 中每个时间点。

表 8-2 常微分方程的求解算法

算法名	含义	特点	说明
ode23	普通 2-3 阶法非刚性解	一步法；2、3 阶 Runge-Kutta 方程；累计截断误差达(Δx) ³	适用于精度较低的情形
ode23s	低阶法解刚性	一步法；2 阶 Rosebrock 算法；低精度	当精度较低时，计算时间比 ode15s 短
ode23t	解适度刚性	梯形算法	适用于刚性情形
ode23tb	低阶法解刚性	梯形算法；低精度	当精度较低时，计算时间比 ode15s 短
ode45	普通 4-5 阶法非刚性解	一步算法；4、5 阶 Runge-Kutta 方程；累计截断误差达(Δx) ³	大部分场合的首选算法
ode15s	变阶法解刚性	多步法；Gear's 反向数值微分；精度中等	若 ode45 失效，可尝试使用
ode113	普通变阶法非刚性解	多步法；Adams 算法；高低精度均可达到 $10^{-3} \sim 10^{-6}$	计算时间比 ode45 短

表 8-3 solver 中的参数

参数名	功能描述
odefun	表示常微分方程
tspan	表示求解区间或求解时刻，通常为 tspan=[t0,tf]或 tspan=[t0,t1,t2,...,tf]（要求单调）
y0	表示初始条件
options	表示使用 odeset 函数所设置的可选参数
p1,p2	表示传递给 odefun 的参数

在 tspan=[t0,t1,t2,...,tf]指定时间点上,使用初始条件 y0 求解常微分方程.常微分方程 odefun 解向量 Y 中的每行结果对应于时间向量 T 中每个时间点。

- [T,Y] = solver(odefun,tspan,y0,options)

利用 odeset 函数所设置的可选参数进行求解，odeset 函数可设置的参数如表 8-4 所示，其用法类似于 optimset 函数。

表 8-4 solver 中 options 的参数

参数名	取值	含义
absTol	有效值：正实数或向量 默认值：1e-6	绝对误差对应于解向量中的所有元素；向量则分别对应于解向量中的每一分量
relTol	有效值：正实数 默认值：1e-3	相对误差对应于解向量中的所有元素。在每步（第 k 步）计算过程中，误差估计为： $e(k) \leq \max(\text{RelTol} * \text{abs}(y(k)), \text{AbsTol}(k))$
normControl	有效值：on、off 默认值：off	为 on 时，控制解向量范数的相对误差，使每步计算中，满足： $\text{norm}(e) \leq \max(\text{RelTol} * \text{norm}(y), \text{AbsTol})$
events	有效值：on、off	为 on 时，返回相应的事件记录
outputFcn	有效值：odeplot、odephas2、odephas3、odeprint 默认值：odeplot	若无输出参量，则 solver 将执行下面操作之一： <ul style="list-style-type: none">• 画出解向量中各元素随时间的变化• 画出解向量中前两个分量构成的相平面图• 画出解向量中前三个分量构成的三维相空间图• 随计算过程，显示解向量
outputSel	有效值：正整数或向量 默认值：[]	若不使用默认设置，则 OutputFcn 所表现的是那些正整数指定的解向量中的分量的曲线或数据
refine	有效值：正整数或 k>1 默认值：k = 1	若 k>1，则增加每个积分步中的数据点记录，使解曲线更加光滑
jacobian	有效值：on、off 默认值：off	若为 on，返回相应的 ode 函数的 Jacobi 矩阵
jpattern	有效值：on、off 默认值：off	为 on 时，返回相应的 ode 函数的稀疏 Jacobi 矩阵
mass	有效值：none、M、M(t)、M(t,y) 默认值：none	M：不随时间变化的常数矩阵 M(t)：随时间变化的矩阵 M(t,y)：随时间、地点变化的矩阵
maxStep	有效值：正实数 默认值：tspans/10	最大积分步长

- `[T,Y]=solver(odefun,tspan,y0,options,p1,p2,...)`

利用传递给函数 `odefun` 的 `p1,p2,..` 等参数进行求解。

2. 求解常微分方程的类型

MATLAB 可以求解 3 种一阶常微分方程, 即显式常微分方程、线性隐式常微分方程和完全隐式常微分方程。

显式常微分方程的形式如下:

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

线性隐式常微分方程的形式如下:

$$\begin{cases} M(t, y)y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

完全隐式常微分方程的形式如下:

$$\begin{cases} f(t, y, y') = 0 \\ y(t_0) = y_0 \end{cases}$$

对于高阶常微分方程 $y^{(n)} = f(t, y, y', \dots, y^{(n-1)})$, 可以将其转换成如下所示的一阶常微分方程组:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_n' = f(t, y_1, y_2, \dots, y_n) \end{cases}$$

3. 具体解法

下面通过 3 个实例分别说明 3 种方程的解法。

【例 8-16】 已知微分方程 $y'' - \mu(1 - y^2)y' + y = 0$ ($y(0) = 0, y'(0) = 2; t \in [0, 30]$), 该方程为显式常微分方程, 分别取 $\mu = 3$ 和 $\mu = 5$ 求解该方程。

首先对微分方程进行变换得到如下形式:

$$\begin{cases} y_1' = y_2 \\ y_2' = \mu(1 - y_1^2)y_2 - y_1 \end{cases}$$

其次对方程组进行函数描述, 并保存为 `myfun8_16.m`, 其内容如下所示:

```
function output = myfun8_16(t,y,mu)
output=zeros(2,1);
output(1) = y(2);
output(2) = mu*(1-y(1)^2)*y(2)-y(1);
```

再次对方程组进行求解, 在命令窗口中输入如下语句:

```
[t1,y1] = ode45(@myfun8_16,[0 30],[0; 2], [],3); % mu=3
[t2,y2] = ode45(@myfun8_16,[0 30],[0; 2], [],5); % mu=5
plot(t1,y1(:,1),'-',t2,y2(:,2),'--')
```

```
title('显式常微分方程的解');
xlabel('t');
ylabel('y');
legend('mu=3','mu=5');
```

图形窗口中的输出结果如图 8-3 所示。

【例 8-17】求解微分方程 $(ty^2 + 1)y' = 3y^3 + y + 4$ ($t \in [0, 10]; y(0) = 2$)，该方程为线性隐式常微分方程。

首先根据微分方程 $(ty^2 + 1)y' = 3y^3 + y + 4$ 和通式 $M(t, y)y' = f(t, y)$ ，得到：

$$\begin{cases} f(t, y) = 3y^3 + y + 4 \\ M(t, y) = ty^2 + 1 \end{cases}$$

其次对 $f(t, y)$ 进行函数描述，并保存为 myfun8_17f.m，其内容如下所示：

```
function output= myfun8_17f(t,y)
output =3* y.^3 +y+4;
```

再次对 $M(t, y)$ 进行函数描述，并保存为 myfun8_17M.m，其内容如下所示：

```
function output = myfun8_17M (t,y)
output = t.*y.^2 +1;
```

最后对方程进行求解，在命令窗口中输入如下语句：

```
options = odeset('RelTol',1e-6,'OutputFcn','odeplot','Mass', @myfun8_17M);
[t,y] = ode45(@myfun8_17f,[0 10],2,options);
xlabel('t');
ylabel('y');
title('线性隐式常微分方程的解')
```

图形窗口中的输出结果如图 8-4 所示。

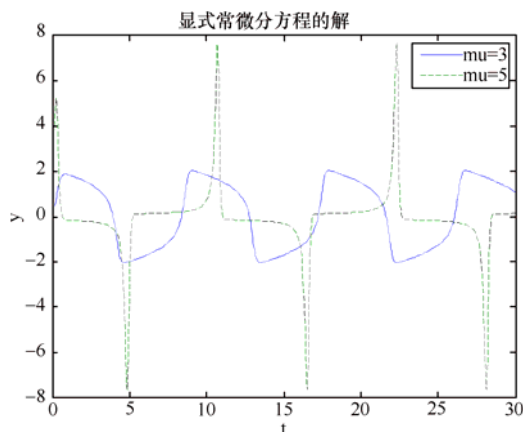


图 8-3 显式常微分方程 $y'' - 2(1 - y^2)y' + y = 0$ 的解

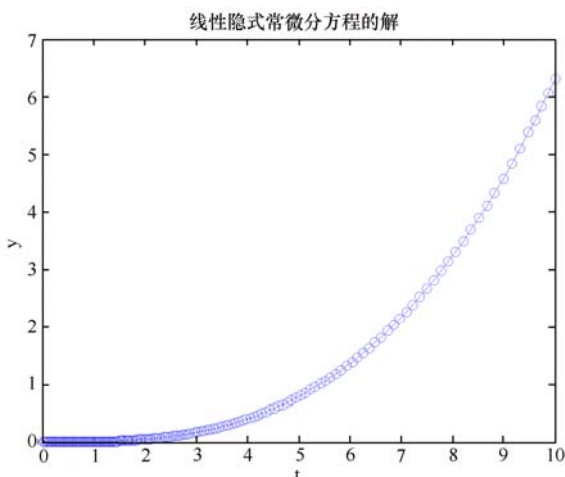


图 8-4 线性隐式常微分方程 $(ty^2 + 1)y' = 3y^3 + y + 4$ 的解

MATLAB 提供了函数 decic 计算自洽的初始值，其具体用法如下：

- `[t,Y] = ode15i(odefun,tspan,y0,yp0,options)`: `y0` 和 `yp0` 用于指定微分方程的初始值，初始值必须是自洽的，即满足 $f(t,y0,yp0)=0$ 。

MATLAB 提供了函数 `decic` 计算自洽的初始值，其具体用法如下：

- `[y0mod,yp0mod] = decic(odefun,t0,y0,fixed_y0,yp0,fixed_yp0,options)`: `t0` 为初始时间，`y0` 和 `yp0` 是猜测的初始值，当 `fixed_y0(i)=1` 时，`y0(i)` 不会被改变，`fixed_yp0` 的作用与 `fixed_y0` 相同。

【例 8-18】求解微分方程 $ty^2(y')^3 - 2y^3(y')^2 + 3t(t^2+1)y' - t^2y = 0$ ($t \in [1, 20]$; $y(0) = \sqrt{3/2}$)，

该方程为完全隐式常微分方程。

首先对方程进行函数描述，并保存为 `myfun8_18.m`，其内容如下所示：

```
function output = myfun8_18 (t,y,dydt)
output = t*y.^2*dydt.^3-2*y.^3*dydt.^2+3*t*(t^2+1)*dydt-t^2*y;
```

其次对方程进行求解，在命令窗口中输入如下语句：

```
t0 = 1;
y0 = sqrt(3/2);
yp0 = 0;
[y0,yp0] = decic(@myfun8_18,t0,y0,1,yp0,0);
[t,y] = ode15i(@myfun8_18,[1 20],y0,yp0);
plot(t,y);
xlabel('t');
ylabel('y');
title('完全隐式常微分方程的解');
```

图形窗口中的输出结果如图 8-5 所示。

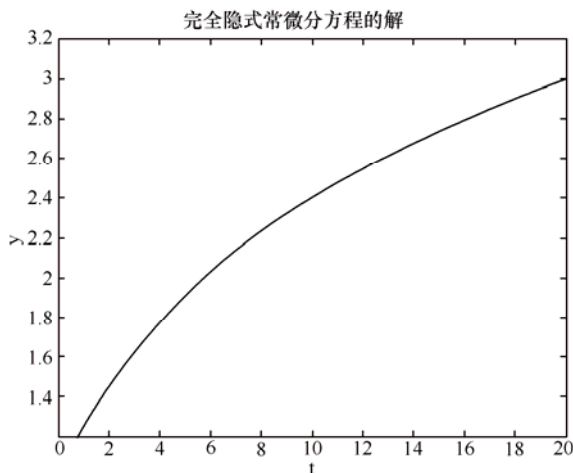


图 8-5 完全隐式常微分方程 $ty^2(y')^3 - 2y^3(y')^2 + 3t(t^2+1)y' - t^2y = 0$ 的解

8.2 数据统计处理

本节主要介绍 MATLAB 在数据统计处理方面的应用，包括最大值和最小值、求和和求积、平均值和中值、标准方差、相关系数以及排序等内容。

1. 随机数的生成

先来了解一下常见的随机数生成函数，如表 8-5 所示。

表 8-5 随机数生成函数

函数名	调用形式	注释
unifrnd	unifrnd (A,B,m,n)	[A,B]上均匀分布（连续）随机数
unidrnd	unidrnd(N,m,n)	均匀分布（离散）随机数
uxprnd	exprnd(Lambda,m,n)	参数为 Lambda 的指数分布随机数
uormrnd	normrnd(MU,SIGMA,m,n)	参数为 MU,SIGMA 的正态分布随机数
chi2rnd	chi2rnd(N,m,n)	自由度为 N 的卡方分布随机数
trnd	trnd(N,m,n)	自由度为 N 的 t 分布随机数
frnd	frnd(N1, N2,m,n)	第一自由度为 N1，第二自由度为 N2 的 F 分布随机数
gamrnd	gamrnd(A,B,m,n)	参数为 A,B 的 γ 分布随机数
betarnd	betarnd(A,B,m,n)	参数为 A,B 的 β 分布随机数
lognrnd	lognrnd(MU,SIGMA,m,n)	参数为 MU,SIGMA 的对数正态分布随机数
nbirnd	nbirnd(R,P,m,n)	参数为 R,P 的负二项式分布随机数
ncfrnd	ncfrnd(N1,N2, delta,m,n)	参数为 N1,N2,delta 的非中心 F 分布随机数
nctrnd	nctrnd(N, delta,m,n)	参数为 N,delta 的非中心 t 分布随机数
ncx2rnd	ncx2rnd(N, delta,m,n)	参数为 N,delta 的非中心卡方分布随机数
raylrnd	raylrnd(B,m,n)	参数为 B 的瑞利分布随机数
weibrnd	weibrnd(A, B,m,n)	参数为 A,B 的韦伯分布随机数
binornd	binornd(N,p,m,n)	参数为 N,p 的二项分布随机数
geornd	geornd(P,m,n)	参数为 p 的几何分布随机数
hygernd	hygernd(M,K,N,m,n)	参数为 M,K,N 的超几何分布随机数
poissrnd	poissrnd(Lambda,m,n)	参数为 Lambda 的泊松分布随机数

【例 8-19】生成[1 5]上均匀分布的 4×4 的随机数矩阵。

在命令窗口中输入如下语句：

```
x=unifrnd(1,5,4,4)
```

命令窗口中的输出结果如下所示：

```
x =
    4.0310    1.6847    1.1847    2.2684
    3.9725    3.8242    1.3885    4.8009
    2.5689    1.1273    4.2938    1.1378
    3.6219    2.1077    3.7793    2.7550
```

2. 数据分析基本函数

在 MATLAB 中提供了大量数据分析函数，在分类介绍这些函数之前，首先给出如下约定：

- 进行一维数据分析时，数据可以用行向量或者列向量来表示，无论哪种表示方法，函数的运算都是对整个向量进行的。
- 进行二维数据分析时，数据可以用多个向量或者二维矩阵来表示。对于二维矩阵，函数的运算总是按列进行的。

MATLAB 提供了包括计算随机变量数字特征在内的大量数据分析函数，详见以下小节内容。

8.2.1 最大值和最小值

在 MATLAB 中计算最大值和最小值的函数分别为 max 和 min，具体用法如下：

- 计算最大值函数：C = max(A)，如果 A 是向量，返回向量中的最大值；如果 A 是矩阵，

返回一个包含各列最大值的行向量。

- 计算最小值函数： $C = \min(A)$ ， \min 和 \max 函数使用方法类似。

【例 8-20】应用求最大值、最小值的函数。

在命令窗口中输入如下语句：

```
x=1:20;
y=randn(1,20);
figure;
hold on;
plot(x,y);
[y_max,I_max]=max(y)           %求向量最大值及其对应下标
plot(x(I_max),y_max,'ro');
[y_min,I_min]=min(y)          %求向量最小值及其对应下标
plot(x(I_min),y_min,'g*');
xlabel('x');
ylabel('y');
legend('原始数据','最大值','最小值');
```

命令窗口中的输出结果如下所示。

```
y_max =
    1.6821

I_max =
    11

y_min =
   -1.1742

I_min =
    15
```

图形窗口中的输出结果如图 8-6 所示。

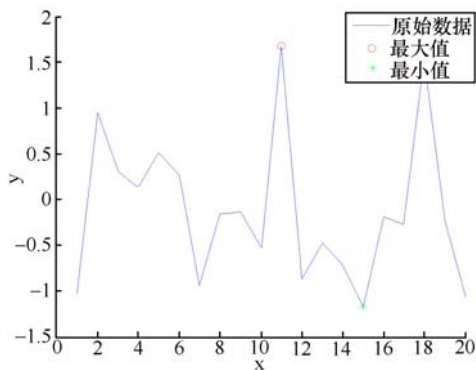


图 8-6 原数据及其最大值和最小值

8.2.2 求和和求积

在 MATLAB 中求和和求积的函数分别为 `sum` 和 `prod`，具体用法如下：

- 计算元素和： $B = \text{sum}(A)$ ，如果 A 是向量，返回向量 A 的各元素之和；如果 A 是矩阵，返回含有各列元素之和的行向量。
- 计算元素连乘积： $B = \text{prod}(A)$ ，如果 A 是向量，返回向量 A 的各元素连乘积；如果 A 是矩阵，返回含有各列元素连乘积的行向量。

【例 8-21】应用求和与求积的函数。

在命令窗口中输入如下语句：

```
x=1:20;  
y=randn(1,20);  
y_sum=sum(y)  
y_prod=prod(y)
```

命令窗口中的输出结果如下所示：

```
y_sum =  
    -2.2656  
  
y_prod =  
   -8.1722e-010
```

8.2.3 平均值和中值

在 MATLAB 中计算平均值和中值的函数分别为 `mean` 和 `median`，具体用法如下：

- 计算均值也叫数学期望： $M = \text{mean}(A)$ ，如果 A 是向量，返回向量 A 的平均值；如果 A 是矩阵，返回含有各列平均值的行向量。
- 计算中值： $M = \text{median}(A)$ ，与 `mean` 函数使用方法类似。

【例 8-22】应用求平均值和中值的函数，在命令窗口中输入如下语句：

```
x=1:20;  
y=randn(1,20);  
y_mean=mean(y)           %求向量平均值  
y_median=median(y)       %求向量中间值
```

命令窗口中的输出结果如下所示：

```
y_mean =  
    -0.2447  
  
y_median =  
     0.0405
```

8.2.4 标准方差

在 MATLAB 中求标准方差的函数为 `std`，具体用法如下：

- $s = \text{std}(A)$ ：如果 A 是向量，返回向量的标准差；如果 A 是矩阵，返回含有各列标准差的

行向量。

另外, MATLAB 计算方差(即标准差的平方)的函数为 `var`:

- `v=var(X)`: 如果 `A` 是向量, 返回向量的方差; 如果 `A` 是矩阵, 返回含有各列方差的行向量。

向量 \mathbf{x} 的标准差定义如下:

$$s = \left[\frac{1}{N-1} \sum_{k=1}^N (x_k - \bar{x})^2 \right]^{\frac{1}{2}}$$

向量 \mathbf{x} 的方差是标准差的平方, 即:

$$s^2 = \frac{1}{N-1} \sum_{k=1}^N (x_k - \bar{x})^2$$

其中 N 是向量 \mathbf{x} 的长度, $\bar{x} = \frac{1}{N} \sum_{k=1}^N x_k$, 即平均值。

当 N 较大时, 取 $s^2 = \frac{1}{N} \sum_{k=1}^N (x_k - \bar{x})^2$, 有时称此 s^2 为样本方差, 而称上式的 s^2 为样本修正方差。

【例 8-23】 计算向量 \mathbf{x} 的标准差。

在命令窗口中输入如下语句:

```
x=1:20;
mean_x=mean(x);
r=0;
for i=1:20
    r=r+(x(i)-mean_x)^2;
end
r1=sqrt(r/10)
r2=sqrt(r/9)
r3=std(x)
```

命令窗口中的输出结果如下所示:

```
r1 =
    8.1548
r2 =
    8.5959
r3 =
    5.9161
```

8.2.5 相关系数

MATLAB 提供了 `corrcoef` 函数计算相关系数, 具体用法如下:

- `corrcoef(X,Y)`: 计算列向量 `X,Y` 的相关系数, 等价于 `corrcoef([X Y])`; `corrcoef(A)` 计算矩阵 `A` 的列向量的相关系数矩阵。

另外, `cov` 函数用来计算协方差:

- `cov(X)`计算向量 X 的协方差；`cov(A)`计算矩阵 A 各列的协方差矩阵，该协方差矩阵的对角线元素是 A 的各列的方差；`cov(X,Y)`等价于 `cov([X Y])`。

【例 8-24】计算协方差与相关系数。

在命令窗口中输入如下语句：

```
X=[1 0 1 3]';  
Y=[-3 5 1 5]';  
A1=cov(X)  
A2=cov(X,Y)  
A3=corrcoef(X)  
A4=corrcoef(X,Y)
```

命令窗口中的输出结果如下所示：

```
A1 =  
    1.5833  
  
A2 =  
    1.5833    1.0000  
    1.0000   14.6667  
  
A3 =  
     1  
  
A4 =  
    1.0000    0.2075  
    0.2075    1.0000
```

8.2.6 排序

在 MATLAB 中用函数 `sort()`来实现数值的排序，具体用法如下：

- $B = \text{sort}(A)$ ：如果 A 是向量，升序排列向量；如果 A 是矩阵，升序排列各个列。
- $B = \text{sort}(\dots, \text{mode})$ ：用 `mode` 选择排序方式，'ascend'为升序，'descend'为降序。

【例 8-25】对随机产生的矩阵 A 分别进行降序和升序排序。

在命令窗口中输入如下语句：

```
A=unifrnd(1,2,4,5)  
Y1=sort(A,'descend')  
Y2=sort(A,'ascend')
```

命令窗口中的输出结果如下所示：

```
A =  
    1.2875    1.5466    1.6790    1.7093    1.4501  
    1.0911    1.4257    1.6358    1.2362    1.4587  
    1.5762    1.6444    1.9452    1.1194    1.6619  
    1.6834    1.6476    1.2089    1.6073    1.7703
```

Y1 =

1.6834	1.6476	1.9452	1.7093	1.7703
1.5762	1.6444	1.6790	1.6073	1.6619
1.2875	1.5466	1.6358	1.2362	1.4587
1.0911	1.4257	1.2089	1.1194	1.4501

Y2 =

1.0911	1.4257	1.2089	1.1194	1.4501
1.2875	1.5466	1.6358	1.2362	1.4587
1.5762	1.6444	1.6790	1.6073	1.6619
1.6834	1.6476	1.9452	1.7093	1.7703

8.3 数据插值

插值是指在所给的基准数据情况下,研究如何平滑地估算出基准数据之间其他点的函数数值。当其他点上函数值较难获取时,此时插值就体现出来它的优越性。

MATLAB 提供了大量的插值函数,下面从一维、二维和三维插值三个方面分别进行介绍。

8.3.1 一维插值

一维插值就是对一维函数 $y=f(x)$ 的插值,其含义如图 8-7 所示,其中实心点 (x,y) 表示已知数据,空心点 (x_i,y_i) 中的 x_i 代表待插值的自变量值, y_i 代表插值结果。

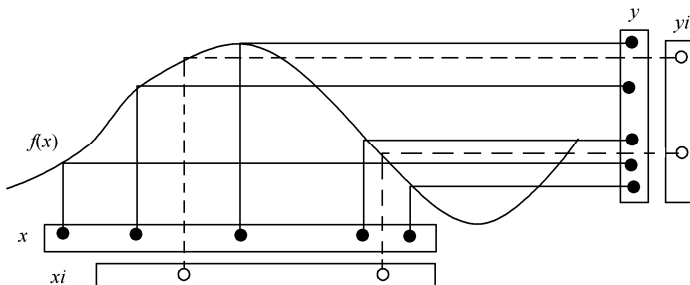


图 8-7 一维插值示意图

MATLAB 提供了函数 `interp1` 实现一维插值,它的具体用法如下所示:

- `y_i=interp1(x,y,x_i,method)`, `x` 和 `y` 是向量且具有相同的长度, `x_i` 可以是标量、向量和任意维矩阵, `y_i` 与 `x_i` 具有相同的大小。
- `y_i=interp1(x,y,x_i,method)`: `method` 用于指定插值的方法,包括最邻近插值 (`method='nearest'`)、线性插值 (默认方法, `method='linear'`)、三次样条插值 (`method='spline'`)、分段三次厄米多项式插值 (`method='pchip'`) 和三次多项式插值 (`method='cubic'`, 与 `method='pchip'` 等价)。
- `y_i=interp1(x,y,x_i,method,'extrap')`: 对超出数据范围的插值数据指定外推方法 `'extrap'`。

【例 8-26】用不同插值方法对数据进行一维插值。

在命令窗口中输入如下语句:

```

x = 0:1.2:10;
y = cos(x);
xi = 0:0.1:10;
yi_nearest = interp1(x,y,xi,'nearest');
yi_linear = interp1(x,y,xi);
yi_spline = interp1(x,y,xi,'spline');
yi_cubic = interp1(x,y,xi,'cubic');
figure;
hold on;
subplot(2,2,1);
plot(x,y,'ro',xi,yi_nearest,'b-');
title('最邻近插值');
subplot(2,2,2);
plot(x,y,'ro',xi,yi_linear,'b-');
title('线性插值');
subplot(2,2,3);
plot(x,y,'ro',xi,yi_spline,'b-');
title('三次样条插值');
subplot(2,2,4);
plot(x,y,'ro',xi,yi_cubic,'b-');
title('三次多项式插值');

```

图形窗口中的输出结果如图 8-8 所示。

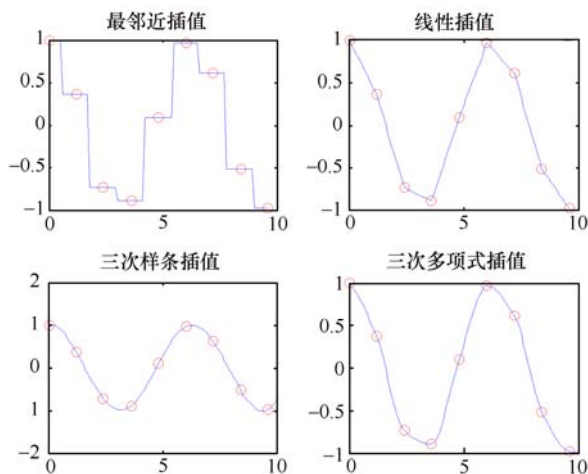


图 8-8 一维插值不同方法比较

由结果不难看出，不同插值方法导致不同的插值结果。

8.3.2 二维插值

二维插值的基本思想与一维插值是相同的，区别在于它是对二维函数 $z=f(x,y)$ 的插值，其含义如图 8-9 所示，其中实心点 (x,y,z) 表示已知数据，空心点 (xi,yi,zi) 中的 xi 和 yi 代表待插值的

自变量值, z_i 代表插值结果。

MATLAB 提供了函数 `interp2` 实现二维插值, 它的具体用法如下所示:

- `zi=interp2(x,y,z,xi,yi)`: 返回值 z_i 是对于自变量 (x_i, y_i) 的插值, 需要说明的是 x 、 y 和 z 应该是同维的。
- `zi=interp2(x,y,z,xi,yi,method)`: `method` 用于指定插值的方法, 包括最邻近插值 (`method='nearest'`)、双线性插值 (默认方法, `method='linear'`)、三次样条插值 (`method='spline'`) 和双三次多项式插值 (`method='cubic'`)。

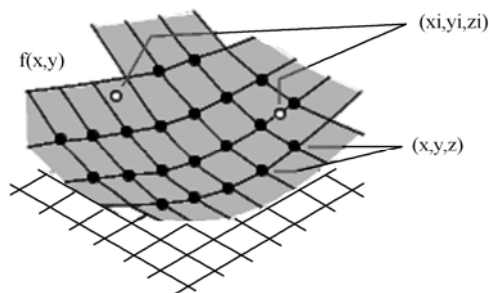


图 8-9 二维插值示意图

- `zi = interp2(x,y,z,xi,yi,method,extrapval)`: 当数据超过原始数据范围时, 输入 `extrapval` 指定一种外推方法。

【例 8-27】用不同插值方法对数据进行二维插值。

在命令窗口中输入如下语句:

```
[x,y] = meshgrid(1:0.2:2,0:0.2:1)           %构造原始数据的自变量, 两自变量必须同维
z=peaks(x,y)                                 %构造原始数据的因变量
[x1,y1] = meshgrid([1 1.5 2],[0.3 0.7])      %构造插值数据的自变量, 两自变量必须同维
z1= interp2(x,y,z,x1,y1,'nearest')           %不同方法插值, 显示数据结果
z2= interp2(x,y,z,x1,y1)
z3= interp2(x,y,z,x1,y1,'spline')
z4= interp2(x,y,z,x1,y1,'cubic')
figure
surf(x,y,z);                                %显示图形结果
title('原始数据');
figure
hold on;
subplot(2,2,1);
surf(x1,y1,z1);
title('最邻近插值');
subplot(2,2,2);
surf(x1,y1,z2);
title('双线性插值');
subplot(2,2,3);
surf(x1,y1,z3);
```



```
title('三次样条插值');
subplot(2,2,4);
surf(x1,y1,z4);
title('双三次多项式插值');
```

命令窗口中的输出结果如下所示:

```
x =
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000

y =
         0         0         0         0         0         0
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
    0.4000    0.4000    0.4000    0.4000    0.4000    0.4000
    0.6000    0.6000    0.6000    0.6000    0.6000    0.6000
    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000

z =
    2.9369    3.5333    3.4946    2.9494    2.1706    1.4122
    2.8229    3.3922    3.3501    2.8242    2.0768    1.3504
    2.5348    3.0267    2.9785    2.5056    1.8401    1.1955
    2.2486    2.5885    2.5024    2.0847    1.5221    0.9853
    2.1843    2.2681    2.0756    1.6758    1.2006    0.7678
    2.4338    2.1681    1.7959    1.3596    0.9338    0.5805

x1 =
    1.0000    1.5000    2.0000
    1.0000    1.5000    2.0000

y1 =
    0.3000    0.3000    0.3000
    0.7000    0.7000    0.7000

z1 =
    2.5348    2.5056    1.1955
    2.1843    1.6758    0.7678
```

z2 =

2.6788	2.9146	1.2730
2.2164	2.0846	0.8765

z3 =

2.6909	2.9779	1.2824
2.1806	2.0992	0.8749

z4 =

2.6896	2.9760	1.2822
2.1829	2.0993	0.8751

在图形窗口中显示如图 8-10 和图 8-11 所示的结果。

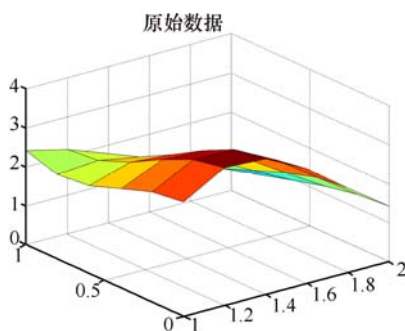


图 8-10 原始数据图形

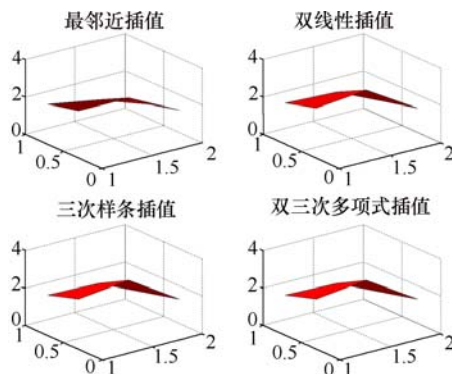


图 8-11 不同二维插值方法比较

8.3.3 三维插值

三维插值的基本思想与一、二维插值是相同的，区别在于它是对三维函数 $v=f(x,y,z)$ 的插值。MATLAB 提供了函数 `interp3` 实现三维插值，它的具体用法如下所示：

- `vi = interp3(x,y,z,v,xi,yi,zi)`: 返回值 `vi` 是对于自变量 (xi,yi,zi) 的插值，需要说明的是 `xi`、`yi`、`zi` 和 `vi` 应该是同维的。
- `vi=interp3(x,y,z,v,xi,yi,zi,method)`: `method` 用于指定插值的方法，包括最邻近插值 (`method='nearest'`)、线性插值(默认方法, `method='linear'`)、三次样条插值(`method='spline'`)和三次多项式插值 (`method='cubic'`)。
- `vi=interp3(x,y,z,v,xi,yi,zi,method,extrapval)`: 当数据超过原始数据范围时，输入 `extrapval` 指定一种外推方法。

【例 8-28】对函数 `flow(n)` 采用不同方法实现三维插值。

在命令窗口中输入如下语句：

```
[x,y,z,v] = flow(30);
[xi,yi,zi] = meshgrid(1:2:5, [0 1], [1 2]);
vi1= interp3(x,y,z,v,xi,yi,zi, 'nearest');
vi2= interp3(x,y,z,v,xi,yi,zi);
```

```

vi3= interp3(x,y,z,v,xi,yi,zi, 'spline');
vi4= interp3(x,y,z,v,xi,yi,zi, 'cubic');
figure
slice(x,y,z,v,2.5,[0.3 0.5],[1 1.5 2]);
title('原始数据');
figure
hold on;
subplot(2,2,1);
slice(xi,yi,zi,vi1, 2.5,[0.3 0.5],[1 1.5 2]);
title('最邻近插值');
subplot(2,2,2);
slice(xi,yi,zi,vi2, 2.5,[0.3 0.5],[1 1.5 2]);
title('线性插值');
subplot(2,2,3);
slice(xi,yi,zi,vi3, 2.5,[0.3 0.5],[1 1.5 2]);
title('三次样条插值');
subplot(2,2,4);
slice(xi,yi,zi,vi4, 2.5,[0.3 0.5],[1 1.5 2]);
title('三次多项式插值');
colormap hsv

```

在图形窗口中显示如图 8-12 和图 8-13 所示的结果。

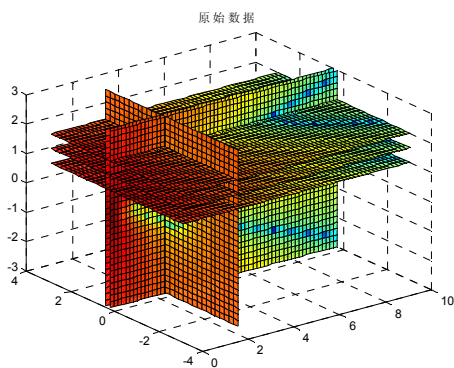


图 8-12 函数 flow 的原始数据

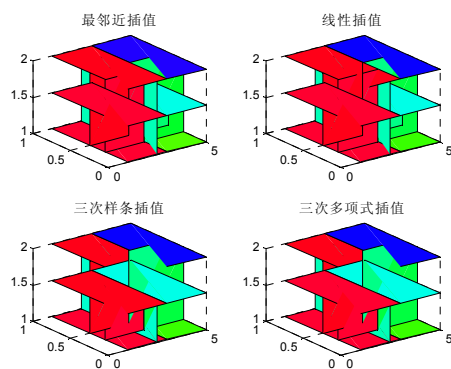


图 8-13 函数 flow 的三维插值结果

8.4 数值积分

数值积分是工程计算中经常应用到的，下面从一元、二元和三元函数数值积分三个方面分别进行介绍。

8.4.1 一元函数积分

在 MATLAB 中提供了函数 quad 和 quadl 计算一元函数的积分，其中函数 quad 采用低阶的

自适应递归 Simpson 方法, 函数 `quadl` 采用高阶的自适应 Lobatto 方法。一般来讲, `quadl` 比 `quad` 命令更加有效。

1. 函数 `quad`

函数 `quad` 的具体用法如下所示:

- `q = quad(fun,a,b)`: 计算函数 `fun` 在 `[a,b]` 区间内的定积分, 其中 `fun` 为函数句柄, `a` 和 `b` 分别是积分区间的下界和上界。
- `q = quad(fun,a,b,tol)`: 基于设定的绝对误差容限 `tol` 计算函数 `fun` 在 `[a,b]` 区间内的定积分, 绝对误差容限 `tol` 的默认值为 10^{-6} 。
- `q = quad(fun,a,b,tol,trace)`: 当 `trace` 为非零值时显示计算定积分的中间迭代结果。

【例 8-29】实现函数 $f(x) = \frac{e^x + \cos(x) + 1}{x}, x \in [1,2]$ 的数值积分计算。

在命令窗口中输入如下语句:

```
y=@(x) 1./x.*(exp(x)+cos(x)+1);
figure
fplot(y,[1 2],'b');
title('被积函数曲线');
q1=quad(y,1,2, 2e-6,1)
p_min=1;
p_max=2;
p_step=0.01;
p_num=(p_max-p_min)/ p_step+1;
q2(1)=0;
for i=2: p_num
    p=p_min+(i-1)*p_step;
    q2(i)=quad(y,1,p);
end
figure
plot(p_min:p_step:p_max,q2);
title('积分过程');
xlabel('上界')
ylabel('积分值')
```

命令窗口中的输出结果如下所示, 并在图形窗口中显示如图 8-14 和图 8-15 所示的结果:

```

9      1.0000000000    2.71580000e-001    1.0886998046
11     1.2715800000    4.56840000e-001    1.7036025323
13     1.2715800000    2.28420000e-001    0.8558853857
15     1.5000000000    2.28420000e-001    0.8477168661
17     1.7284200000    2.71580000e-001    1.0455386278

q1 =
    3.8378
```

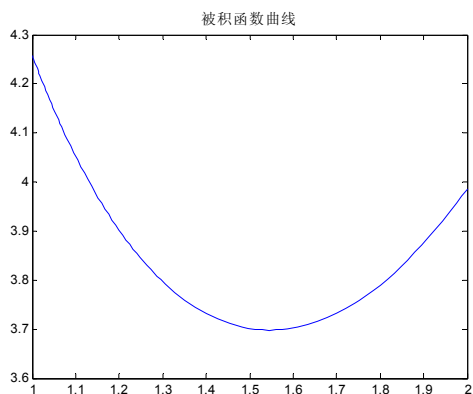


图 8-14 被积函数 $f(x) = \frac{e^x + \cos(x) + 1}{x}$ 曲线

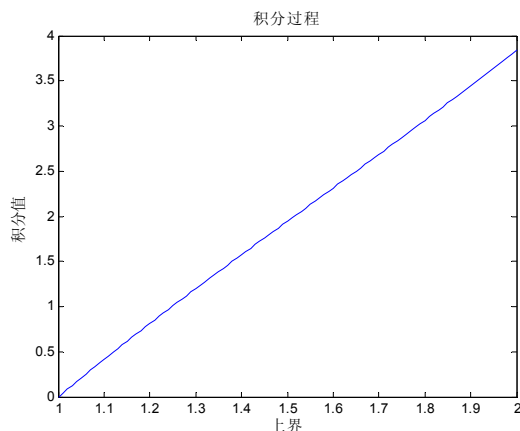


图 8-15 函数 $f(x) = \frac{e^x + \cos(x) + 1}{x}$ 积分过程曲线

本例需要说明的是，由积分理论可知 $\text{quad}(y,1,1)=0$ ，如在命令窗口中输入如下语句：

```
quad(y,1,1)
```

在命令窗口中给出如下警告信息和结果：

```
Warning: Minimum step size reached; singularity possible.
```

```
> In quad at 103
```

```
ans =
```

```
0
```

因此本例为了避免警告信息，人工进行赋值 $q2(1)=0$ 。

在计算一维积分时，有可能得到 3 种警告信息：

- 'Minimum step size reached': 已经达到了最小步长。这意味着积分的迭代区间比给定的定积分区间的截断误差值小。一般来说，这表明被积函数的可积性是奇异的。
- 'Maximum function count exceeded': 计算函数值的次数超过 10 000，一般来说，这也表明被积函数的可积性是奇异的。
- 'Infinite or Not-a-Number function value encountered': 积分过程中出现浮点数溢出或者被 0 除。

2. 函数 quadl

函数 `quadl` 的具体用法与函数 `quad` 类似，这里不再赘述。通过下面的例子比较两种函数的差异。

【例 8-30】通过函数 $f(x) = \frac{e^x + \cos(x) + 1}{x}$, $x \in [1,2]$ 的数值积分比较函数 `quadl` 和函数 `quad` 的计算结果。

在命令窗口中输入如下语句：

```
y=@(x) 1./x.*(exp(x)+cos(x)+1);
p_min=1;
p_max=2;
p_step=0.01;
p_num=(p_max-p_min)/p_step+1;
q1(1)=0;
```

```

q2(1)=0;
for i=2: p_num
    p=p_min+(i-1)*p_step;
    q1(i)=quad(y,1,p);
    q2(i)=quadl(y,1,p);
end
figure
plot(p_min:p_step:p_max,q1,'+',p_min:p_step:p_max,q2,'r*');
title('积分过程');
xlabel('上界')
ylabel('积分值')
legend('quad','quadl')
e=max(abs(q1-q2))

```

命令窗口中的输出结果如下所示，并在图形窗口中显示如图 8-16 所示的结果。

```

e =
    3.8756e-008

```

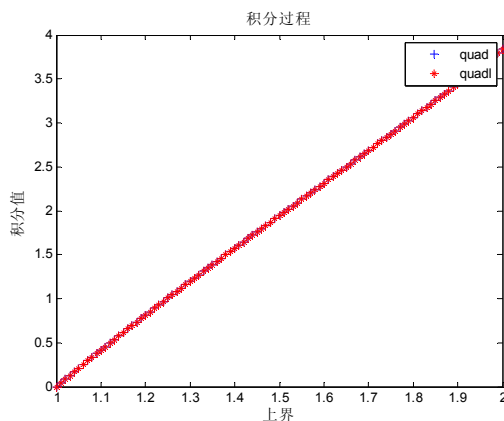


图 8-16 两种函数积分过程曲线

从本例结果可以看出，二者存在差异但差异很小。

8.4.2 矢量积分

矢量积分可以看做多个一元定积分，或是看做变参数一元定积分。

在 MATLAB 中用 `quadv` 函数计算一元函数的矢量积分，其用法类似于上一节所讲的一元函数积分函数，这里不再赘述

【例 8-31】 计算积分 $\int_{-1}^3 \frac{1}{n} \exp\left(-\frac{x}{n^2}\right) dx$ ， $n=1,2,3,4,5$ 。

在命令窗口中输入如下语句：

```

y=@(x,n)1./n.*exp(-x./((1:n).^2));
q=quadv(@(x)y(x,5),-1,3)

```

命令窗口中的输出结果如下所示：

```
q =  
    0.5337    0.6493    0.7218    0.7535    0.7695
```

8.4.3 二元函数积分

二元函数积分的形式如下：

$$Q = \int_{y_{\min}}^{y_{\max}} \int_{x_{\min}}^{x_{\max}} f(x, y) dx dy$$

在 MATLAB 中提供了函数 `dblquad` 计算二元函数的积分，根据 `dx dy` 的顺序，称 x 为内积分变量， y 为外积分变量。该函数按照先计算内积分值，再计算外积分值的顺序进行。它的具体用法如下所示：

- `q = dblquad(fun,xmin,xmax,ymin,ymax)`：对于内积分变量 x 下界和上界分别为 `xmin` 和 `xmax`，以及外积分变量 y 下界和上界分别为 `ymin` 和 `ymax` 构成的矩形区域，计算二元函数 `fun` 的积分。
- `q = dblquad(fun,xmin,xmax,ymin,ymax,tol)`：用 `tol` 指定绝对计算精度。
- `q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)`：用 `method` 指定计算一维积分时采用的函数，`method=@quad` 即为采用函数 `quad`，该方式也是默认方式；`method=@quadl` 即为采用函数 `quadl`；用户也可以定制一维积分函数，但该函数的使用方法必须与函数 `quad` 或函数 `quadl` 一致。

【例 8-32】计算积分 $\int_0^{\pi} \int_{\pi}^{2\pi} (y \sin x + 2x \cos y) dx dy$ 。

在命令窗口中输入如下语句：

```
f=@(x,y)y*sin(x)+2*x*cos(y);  
xmin=pi;  
xmax=2*pi;  
ymin=0;  
ymax=pi;  
q=dblquad(f,xmin,xmax,ymin,ymax)
```

命令窗口中的输出结果如下所示：

```
q1 =  
-9.8696
```

函数 `dblquad` 只能处理矩形积分区间的情况，若出现非矩形积分区间，则需经过一定的变换，最直接的方法就是用一个矩形包含指定积分区间，并将多余空间对应点的函数值赋为 0。

【例 8-33】计算函数 $f(x, y) = (2x + 3y)e^x$ 在圆形区域 $x^2 + y^2 \leq 1$ 上的数值积分。

在命令窗口中输入如下语句：

```
clear  
clc  
z=@(x,y) exp(x).*(2*x+3*y).*(sqrt(x.^2+y.^2)<=1);  
q=dblquad(z,-1,1,-1,1,1e-5)
```

命令窗口中的输出结果如下所示：

```
q1 =
```

1.7059

8.4.4 三元函数积分

三元函数积分的形式如下：

$$Q = \int_{z_{\min}}^{z_{\max}} \int_{y_{\min}}^{y_{\max}} \int_{x_{\min}}^{x_{\max}} f(x, y, z) dx dy dz$$

在 MATLAB 中提供了函数 `triplequad` 计算三元函数的积分，该函数按照先计算最内层积分值，再计算中间层积分值，最后计算最外层积分值的顺序进行。它的具体用法如下所示：

- `q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax)`: 对于内积分变量 x 下界和上界分别为 x_{\min} 和 x_{\max} ，中间积分变量 y 下界和上界分别为 y_{\min} 和 y_{\max} ，以及外积分变量 z 下界和上界分别为 z_{\min} 和 z_{\max} 构成的长方体区域，计算三元函数 fun 的积分。
- `q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol)`: 用 `tol` 指定绝对计算精度。
- `q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method)`: 用 `method` 指定计算一维积分时采用的函数，`method=@quad` 即为采用函数 `quad`，该方式也是默认方式；`method=@quadl` 即为采用函数 `quadl`；用户也可以定制一维积分函数，但该函数的使用方法必须与函数 `quad` 或函数 `quadl` 一致。

【例 8-34】计算函数 $f(x, y, z) = 2 + y \sin x + 3z \cos x (x \in [0, 2]); y \in [0, 2\pi]; z \in [-2\pi, 0]$ 的数值积分。

在命令窗口中输入如下语句：

```
clear
clc
q = triplequad(@(x,y,z) (2+y*sin(x)+3*z*cos(x)), 0, 2, 0, 2*pi, -2*pi, 0)
```

命令窗口中的输出结果如下所示：

```
q =
-4.7757
```

8.5 最优化问题求解

最优化问题是工程中经常遇到的问题，下面从无约束非线性极小化、有约束极小化、二次规划和线性规划、线性最小二乘、非线性最小二乘和多目标寻优方法等方面进行介绍。

8.5.1 无约束非线性极小化

在 MATLAB 中无约束非线性极小化问题的处理，使用的是 `fminsearch` 函数。具体用法如下：

- `x=fminsearch(fun,x0)`: fun 代表目标函数， $x0$ 为初值，它可以是标量、向量或矩阵。
- `x=fminsearch(fun,x0,options)`: `options` 为进行优化的各种属性，由 `optimset` 函数进行设置。
- `[x,fval]=fminsearch(...)`: x 代表极小值点， $fval$ 代表最小值。

另外，还有一个 `fminunc` 函数，也是解决无约束非线性极小化问题的函数，用法与 `fminsearch` 函数相似，该函数求的是局部解。

【例 8-35】求解正弦函数在 $x0=3$ 附近的极小值点。在命令窗口中输入如下语句：


```
clear
clc
X = fminsearch(@sin,3)
```

在命令窗口中的输出结果如下所示：

```
X =
    4.7124
```

8.5.2 有约束极小化

有约束条件的极小化问题相比于无约束条件极小化情况复杂很多，种类也比较繁多，这里只简单介绍函数 `fmincon` 的使用。

`fmincon` 函数用于解决如下约束条件的极小化问题：

```
min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq  (线性约束)
                        C(X) <= 0, Ceq(X) = 0    (非线性约束)
                        LB <= X <= UB          (边界)
```

`fmincon` 函数的具体用法如下：

- `x=fmincon (fun,x0,A,B)`: `fun` 代表目标函数，`x0` 为初值，它可以是标量，向量或矩阵，线性约束条件为 $A \cdot X \leq B$ 时，找到目标函数的极小值点。
- `x= fmincon (fun,x0,A,B,Aeq,Beq)`: 线性约束条件为 $A \cdot X \leq B$ 和 $Aeq \cdot X = Beq$ ，无不等式存在时设 $A=[]$ ， $B=[]$ 。
- `x= fmincon (fun,x0,A,B,Aeq,Beq,LB,UB,nonlcon)`: 计算非线性有约束条件 ($C(X) \leq 0$, $Ceq(X) = 0$) 下 `fun` 函数的极小值点，如果没有边界则设定 $LB = []$, $UB = []$ 。

【例 8-36】求解约束条件下函数的极小值点。

在命令窗口中输入如下语句：

```
clear;
clc;
X = fmincon(@(x) 3*sin(x(1))+exp(x(2)),[1;1],[],[],[],[],[0 0])
```

命令窗口中的输出结果如下所示：

```
Active inequalities (to within options.TolCon = 1e-006):
```

```
    lower    upper    ineqlin    ineqnonlin
```

```
    1
```

```
    2
```

```
X =
```

```
    0
```

```
    0
```

8.5.3 二次规划和线性规划

1. 二次规划

用下式对二次规划的标准问题进行描述，其中 X 、 b 、 b_{eq} 、 lb 和 ub 都为列向量， A 、 A_{eq} 和 H 都为符合维数要求的矩阵。

$$\begin{cases} \min_X \left(\frac{1}{2} X'HX + f'X \right) \\ AX \leq b, A_{eq}X = b_{eq} \\ lb \leq X \leq ub \end{cases}$$

函数 `quadprog` 用来处理二次规划问题，下面详细介绍该函数的具体用法：

- `x = quadprog(H,f,A,b)`: 计算 $\begin{cases} \min_X \left(\frac{1}{2} X'HX + f'X \right) \\ AX \leq b \end{cases}$ 线性规划的最优解。
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)`: 由指定设置 `options` 和初值 `x0` 开始计算 $\begin{cases} \min_X \left(\frac{1}{2} X'HX + f'X \right) \\ AX \leq b, A_{eq}X = b_{eq} \\ lb \leq X \leq ub \end{cases}$ 线性规划的最优解，其中 `options` 可以由函数 `optimset` 设定。
- `[x,fval] = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)`: 附加返回最小值。

【例 8-37】计算 $\begin{cases} \min \left(\frac{1}{3} \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 3x_1 - 5x_2 \right) \\ x_1 + 3x_2 \leq 2 \\ -x_1 + 2x_2 \leq 8 \\ x_1, x_2 \geq 0 \end{cases}$ 的线性规划。

在命令窗口中输入如下语句：

```
H = [1 -1; -1 2];
f = [-3; -5];
A = [1 3; -1 2];
b = [2; 8];
[x,fval] = quadprog(H,f,A,b,[],[],[0;0])
```

命令窗口中的输出结果如下所示：

```
Warning: Large-scale algorithm does not currently solve this problem formulation,
using medium-scale algorithm instead.
> In quadprog at 291
Optimization terminated.
x =
    1.2941
    0.2353

fval =
   -4.4706
```

2. 线性规划

用下式对线性规划问题进行数学描述, 其中 X 、 f 、 b 、 b_{eq} 、 lb 和 ub 都为列向量, A 和 A_{eq} 都为符合维数要求的矩阵。

$$\begin{cases} \min_X f^T X \\ AX \leq b, A_{eq} X = b_{eq} \\ lb \leq X \leq ub \end{cases}$$

函数 `linprog` 用来处理线性规划问题, 下面详细介绍该函数的具体用法:

- $x = \text{linprog}(f, A, b)$: 计算 $\begin{cases} \min_X f^T X \\ AX \leq b \end{cases}$ 线性规划的最优解。
- $x = \text{linprog}(f, A, b, A_{eq}, b_{eq})$: 计算 $\begin{cases} \min_X f^T X \\ AX \leq b, A_{eq} X = b_{eq} \end{cases}$ 线性规划的最优解。
- $x = \text{linprog}(f, [], [], A_{eq}, b_{eq})$: 计算 $\begin{cases} \min_X f^T X \\ A_{eq} X = b_{eq} \end{cases}$ 线性规划的最优解。
- $x = \text{linprog}(f, A, b, A_{eq}, b_{eq}, lb, ub)$: 计算 $\begin{cases} \min_X f^T X \\ AX \leq b, A_{eq} X = b_{eq} \\ lb \leq X \leq ub \end{cases}$ 线性规划的最优解。
- $x = \text{linprog}(f, A, b, A_{eq}, b_{eq}, lb, ub, x_0)$: 由初值 x_0 开始计算 $\begin{cases} \min_X f^T X \\ AX \leq b, A_{eq} X = b_{eq} \\ lb \leq X \leq ub \end{cases}$ 线性规划的最优解。
- $x = \text{linprog}(f, A, b, A_{eq}, b_{eq}, lb, ub, x_0, options)$: 由指定设置 `options` 和初值 x_0 开始计算 $\begin{cases} \min_X f^T X \\ AX \leq b, A_{eq} X = b_{eq} \\ lb \leq X \leq ub \end{cases}$ 线性规划的最优解, 其中 `options` 由函数 `optimset` 设置, 详见表 8-1。
- $[x, fval] = \text{linprog}(f, A, b, A_{eq}, b_{eq}, lb, ub, x_0)$: 附加返回 $f^T X$ 。

【例 8-38】计算 $\begin{cases} \min(x_1 + 3x_2 - 5x_3) \\ x_1 - x_2 + x_3 \leq 8 \\ 3x_1 + x_3 \leq 4 \\ x_1, x_2, x_3 \geq 0 \end{cases}$ 的线性规划。

在命令窗口中输入如下语句:

```
f=[1;3;-5];
A=[1 -1 1;3 0 1];
b=[8;4];
x = linprog(f,A,b,[],[],[0;0;0],[inf;inf;inf])
```

在命令窗口中的输出结果如下所示：

```
Optimization terminated.
```

```
x =  
    0.0000  
    0.0000  
    4.0000
```

需要说明的是，在有些情况下，无法寻找到最优解，这往往是由约束条件不够引起的。

MATLAB 还提供了函数 `bintprog` 进行二进制整数规划，但遗憾的是，没有提供进行整数规划的函数。

8.5.4 线性最小二乘

利用左除 (`\`) 可以求解线性最小二乘问题。

如果 A 为 $m \times n$ 矩阵 ($m \neq n$)，且 b 为具有 m 个元素的列向量或具有多个此类列向量的矩阵，则 $X=A \backslash b$ 为等式 $AX=b$ 的最小二乘意义上的解。

1. 非负线性最小二乘

非负线性最小二乘问题的数学描述如下所示：

$$\min_x \frac{1}{2} \|Cx - d\|_2^2$$

$$x \geq 0$$

其中，矩阵 C 和向量 d 为目标函数的系数。

在 MATLAB 中用 `lsqnonneg` 函数求线性问题的非负最小二乘解，具体用法如下所示：

- `x=lsqnonneg(C,d)`：返回向量 x ，使得范数 $(C*x-d)$ 最小化，约束条件为 $x \geq 0$ 。 C 和 d 必须为实数。
- `x=lsqnonneg(C,d,x0)`：若所有的 $x0 \geq 0$ ，则使用 $x0$ 为初值，否则使用默认值。默认的初值为原点（当 $x0=[]$ 或只提供两个输入变量时也使用默认值）。
- `x=lsqnonneg(C,d,x0,options)`：用 `options` 结构指定的优化参数进行最小化。
- `[x,resnorm]=lsqnonneg(...)`：返回残差的平方范数值 $\text{norm}(C*x-d)^2$ 。

【例 8-39】通过下面的 4×2 问题比较 `lsqnonneg` 函数解与无约束最小二乘解的不同。

在命令窗口中输入如下语句：

```
C=[0.0372 0.2869;0.6861 0.7071;0.6233 0.6245;0.6344 0.6170];  
d=[0.8587;0.1781;0.0747;0.8405];  
x1=lsqnonneg(C,d)  
x2=C\d
```

命令窗口中的输出结果如下所示：

```
x1 =  
    0  
    0.6929  
  
x2 =
```

-2.5627

3.1108

由此可见二者的解不完全一样，非负最小二乘解没有负值。

2. 有约束线性最小二乘

有约束线性最小二乘问题的数学描述如下所示：

$$\begin{aligned} \min_x \quad & \frac{1}{2} \|Cx - d\|_2^2 \\ & A \cdot X \leq b \\ & Aeq \cdot x = beq \\ & lb \leq x \leq ub \end{aligned}$$

其中， C 、 A 和 Aeq 为矩阵， d 、 b 、 beq 、 lb 、 ub 和 x 为向量。

在 MATLAB 中用 `lsqlin` 函数求有约束线性最小二乘解，具体用法如下所示：

- `x=lsqlin(C,d,A,b)`: 求解最小二乘意义上的线性系统 $C \cdot x = d$ ，约束条件为 $A \cdot X \leq b$ ，其中 C 为 $m \times n$ 的矩阵。
- `x=lsqlin(C,d,A,b,Aeq,beq)`: 加上等式约束 $Aeq \cdot x = beq$ 以后求解上面的问题，如果没有不等式存在，则令 $A=[]$, $b=[]$ 。
- `x=lsqlin(C,d,A,b,Aeq,beq,lb,ub)`: 为 x 定义一系列下界 lb 和上界 ub ，使得总有 $lb \leq x \leq ub$ 。
- `[x,resnorm,residual,exitflag,output]=lsqlin(...)`: 返回与优化信息有关的结构输出参数 `output`。

【例 8-40】求解下列问题的最小二乘解。

$$\begin{aligned} C \cdot x &= d \\ A \cdot x &\leq b \\ lb &\leq x \leq ub \end{aligned}$$

在命令窗口中输入如下语句：

```
C=[0.9501 0.7620 0.6153 0.4057;
    0.2311 0.4564 0.7919 0.9354;
    0.6068 0.0185 0.9218 0.9169;
    0.4859 0.8214 0.7382 0.4102;
    0.8912 0.4447 0.1762 0.8936];
d=[0.0578;0.3528;0.8131;0.0098;0.1388];
A=[0.2027 0.2721 0.7467 0.4659;
    0.1987 0.1988 0.4450 0.4186;
    0.6037 0.0152 0.9318 0.8462];
b=[0.5251;0.2026;0.6721];
lb=-0.1*ones(4,1);
ub=2*ones(4,1);
```

命令窗口中的输出结果如下所示：

```
Warning: Large-scale algorithm can handle bound constraints only;
        using medium-scale algorithm instead.
> In lsqlin at 286
```

Optimization terminated.

```
x =
    -0.1000
    -0.1000
     0.2152
     0.3502
```

8.5.5 非线性最小二乘

非线性最小二乘问题的数学描述如下所示:

$$\min_x f(x) = f_1(x)^2 + f_2(x)^2 + \cdots + f_m(x)^2 + L$$

式中, L 为常数。

在 MATLAB 中, 用 `lsqnonlin` 函数求解非线性最小二乘问题, 具体用法如下所示:

- `x=lsqnonlin(fun,x0)`: 初值为 `x0`, 求 `fun` 函数的最小平方和。`fun` 函数将返回一个数值向量但不是值的平方。
- `x=lsqnonlin(fun,x0,lb,ub)`: 定义一系列的下界 `lb` 和上界 `ub`, 使得总有 `lb<=x<=ub`。
- `x=lsqnonlin(fun,x0,lb,ub,options)`: 用 `options` 结构指定的优化参数进行最小化。
- `x=lsqnonlin(fun,x0,lb,ub,options,P1,P2,...)`: 将问题参数 `P1`、`P2` 等直接传递给 `fun` 函数, 将空矩阵传递给 `options` 参数作为其默认值。
- `[x,resnorm]=lsqnonlin(...)`: 返回 `x` 处残差的平方范数值 `sum(fun(x).^2)`。

【8-41】求解 x , 使得下式最小化:

$$\sum_{k=1}^{10} (2 + 2k - e^{kx_1} - e^{kx_2})^2$$

初值为 $x=[0.3 \ 0.4]$ 。

由于 `lsqnonlin` 函数假设用户提供的平方和不是显式表达的, 因此 `lsqnonlin` 函数中的函数应该变换为向量值函数, 如下所示:

$$F_k(x) = 2 + 2k - e^{kx_1} - e^{kx_2}$$

$$k=1,2,\cdots,10$$

编辑 M 文件并保存为 `myfun1.m` 文件, 程序代码如下所示:

```
function F=myfun1(x)
k=1:10;
F=2+2*k-exp(k*x(1))-exp(k*x(2));
```

其次在命令窗口中输入如下语句:

```
x0=[0.3 0.4]
[x,resnorm]=lsqnonlin(@myfun1,x0)
```

命令窗口中的输出结果如下所示:

```
x =
    0.2578    0.2578
```

```
resnorm =  
124.3622
```

8.5.6 多目标寻优方法

前面介绍的优化方法只有一个目标函数，我们称之为单目标优化方法。事实上，在许多实际工程问题中，很多时候我们希望多个指标都达到最优值，因此就出现了多个目标函数，这种问题被称为多目标寻优方法。多目标寻优问题的数学描述如下所示：

$$\begin{aligned} \min_{x \in R^n} & F(x) \\ G_i(x) &= 0 \quad i=1, \dots, m_e \\ G_i(x) &\leq 0 \quad i=m_e+1, \dots, m \\ x_i &\leq x \leq x_u \end{aligned}$$

其中， $F(x)$ 为目标函数。

多目标寻优问题往往没有唯一解，这就必须引进非劣解的概念。

若 $x^* \in \Omega$ ，且对于 x^* 不存在 Δx ，使得：

$$(x^* + \Delta x) \in \Omega$$

和

$$\begin{aligned} F_i(x^* + \Delta x) &\leq F_i(x^*) \quad i=1, \dots, m \\ F_j(x^* + \Delta x) &\leq F_j(x^*) \quad \text{for some } j \end{aligned}$$

不能同时成立，那么定义 x^* 为多目标寻优问题的非劣解。

1. 多目标寻优的解法

多目标寻优有多种解法，以下列举几种常用形式：

(1) 权和法

该方法将多目标向量问题转化为所有目标的加权求和的标量问题，数学描述如下所示：

$$\min_{x \in \Omega} f(x) = \sum_{i=1}^m \omega_i \cdot F_p(x)^2$$

其中， ω_i 为加权因子，它的选取方法很多，比如容限法和加权因子分解法等。

(2) ε 约束法

ε 约束法对目标函数向量中的主要目标 F_p 进行最小化，将其他目标用不等式约束的形式写出：

$$\begin{aligned} \min_{x \in \Omega} & \cdot F_p(x) \\ \text{sub. } & F_i(x) \leq \varepsilon_i \quad i=1, \dots, m \quad i \neq p \end{aligned}$$

(3) 目标达到法

目标函数为 $F(x) = \{F_1(x), F_2(x), F_3(x), \dots, F_m(x)\}$ ，对应的目标值为 $F^* = \{F_1^*, F_2^*, \dots, F_m^*\}$ 。允许目标函数有正负偏差，偏差的大小由加权系数 $W = (W_1, W_2, \dots, W_m)$ 控制，于是目标达到问题就可以表述为标准的寻优问题：

$$\begin{aligned} \min_{\gamma \in R, x \in \Omega} & \gamma \\ \text{sub. } & F_i(x) - \omega_i \gamma \leq F_i^* \quad i=1, \dots, m \end{aligned}$$

指定目标 $\{F_1^*, F_2^*\}$ ，定义目标点 P 。权重向量定义从 P 到可行域空间 $\Lambda(\gamma)$ 的搜索方向。在寻优过程中， γ 的变化改变可行域的大小，约束边界变为唯一解点 (F_{1s}, F_{2s}) 。

(4) 目标达到法的改进

目标达到法的一个好处是可以将多目标寻优问题转化为非线性规划问题。通过将目标达到问题变为最大最小化问题来获得更合适的目标函数：

$$\min_{x \in R^n} \max_i \{\Lambda_i\}$$

$$\text{其中, } \Lambda_i = \frac{F_i(x) - F_i^*}{W_i} \quad i = 1, \dots, m$$

2. 多目标寻优的有关函数

多目标达到寻优的数学描述如下所示：

$$\begin{aligned} & \min_{x, \gamma} \\ & F(x) - \text{weight} \cdot \gamma \leq \text{goal} \\ & c(x) \leq 0 \\ & \text{ceq}(x) = 0 \\ & A \cdot x \leq b \\ & A_{\text{eq}} \cdot x = \text{beq} \\ & lb \leq x \leq ub \end{aligned}$$

其中, x 、 weight 、 goal 、 b 、 beq 、 lb 、 ub 为向量, A 、 A_{eq} 为矩阵, $c(x)$ 、 $\text{ceq}(x)$ 、 $F(x)$ 为函数, 返回向量。 $c(x)$ 、 $\text{ceq}(x)$ 、 $F(x)$ 也可以是非线性函数。

在 MATLAB 中, 利用 `fgoalattain` 函数来处理多目标寻优问题, 具体用法如下所示:

- `x=fgoalattain(fun,x0,goal,weight)`: 试图通过改变 x 来使目标函数 `fun` 达到 `goal` 指定的目标。初值为 `x0`, `weight` 参数指定权重。
- `x=fgoalattain(fun,x0,goal,weight,A,b)`: 求解目标寻优问题, 约束条件为线性不等式 $A \cdot x \leq b$ 。
- `x=fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)`: 求解目标寻优问题, 除了 $A \cdot x \leq b$ 外, 还有 $A_{\text{eq}} \cdot x = \text{beq}$, 若无不等式存在, 则设置 $A=[]$, $b=[]$ 。
- `x=fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)`: 为设计变量 x 定义下界 lb 和上界 ub 集合, 使得 $lb \leq x \leq ub$ 。
- `x=fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)`: `nonlcon` 是一个用户定义的函数 `function [c,ceq]=mycon(x)`, 根据状态向量 x 计算非线性约束 $c(x) \leq 0$ 和非线性等式约束 $\text{ceq}(x) = 0$, 若不存在边界, 则设置 $lb=[]$, $ub=[]$ 。

下面举一个实际的例子来说明多目标寻优问题的解决。

【例 8-42】某玩具厂制作两种不同的涂料产品 A 和 B, 已知生产 A 涂料 100kg 需要 8 个工时, 生产 B 涂料 100kg 需要 10 个工时。限定每日的工时数为 40, 并希望不需要临时工, 也不需工人加班生产。这两种涂料每 100kg 可获利 100 元。此外, 有个顾客要求供应 B 涂料 600kg, 请问应如何制订生产计划, 达到最优?

分析: 假设制作 A 和 B 两种涂料的数量分别为 x_1 、 x_2 (均以 100kg 计), 为了使生产计划比较合理并用人尽量少, 使利润最大化, 并且 B 涂料的产量尽量多, 由以上分析可建立如下所示的数学描述:

$$\begin{aligned} & \min z_1 = 8x_1 + 10x_2 \\ & \min z_2 = 100x_1 + 100x_2 \end{aligned}$$

$$\begin{aligned}\min z_3 &= x_2 \\ 8x_1 + 10x_2 &\leq 40 \\ x_2 &\geq 6 \\ x_1, x_2 &\geq 0\end{aligned}$$

编写目标函数的 M 文件，保存为 goal.m，返回目标计算值：

```
function f=myfun(x)
f(1)=8*x(1)+10*x(2);
f(2)=-100*x(1)-100*x(2);
f(3)=-x(2);
```

给定目标，权重按目标比例确定，给出初始值：

```
goal=[400 -800 -6];
weight=[400 -800 -6];
x0=[2 2];
```

给出约束条件的系数：

```
A=[8 10;0 -1];
b=[40 -6];
lb=zeros(2,1);
options=optimset('MaxFunEvals',5000); %函数将最大次数设置为 5000 次
[x,fval,attainfactor,exitflag]=...
fgoalattain(@goal1,x0,goal,weight,A,b,[],[],LB,[],[],options);
```

命令窗口中的输出结果如下所示：

```
x=
    2.0429    1.9458

fval=
    35.8007   -398.8648   -1.9458

attainfactor=
   -0.0646

exitflag=
    0
```

由结果可知，经过 5000 次迭代以后，生产 A、B 涂料的数量分别为 204.29kg 和 194.58kg。

第 9 章 S-函数

Simulink 为用户提供了许多内置的基本库模块,通过这些模块进行连接而构成系统的模型。对于那些经常使用的模块进行组合并封装可以构建出重复使用的新模块,但它依然基于 Simulink 原来提供的内置模块。Simulink S-functions 是一种强大的对模块库进行扩展的工具。

本章主要介绍 S-函数 (S-functions) 的基本概念和工作原理, Level-1 M 文件型、Level-2 M 文件型和 C MEX 文件型 S-函数的编写,以及使用 S-函数创建器进行 S-函数的编写。

9.1 基本概念

S-函数是 Simulink 模块的计算机语言描述,可以用来表示连续、离散或者混合系统,还可以实现硬件驱动等功能。S-函数可以由 M 文件或 C/C++、Fortran 等编程语言以 MEX 文件的形式编写。它提供强大的机制拓展了 Simulink 的应用,而且调用方便。S-函数提供了扩展 Simulink 模块库的有力工具,它采用一种特定的调用语法,使函数和 Simulink 解法器进行交互。S-函数最广泛的用途是定制用户自己的 Simulink 模块。

S-函数与 Simulink 交互的三个模块如图 9-1 所示,其中 Level-1 M 文件型和 C MEX 文件型 S-函数通过左边模块交互, Level-2 M 文件型 S-函数通过中间模块交互,使用 S-函数创建器的 S-函数通过右边模块交互,它们都位于\Simulink\User-Defined Functions 目录下。



图 9-1 S-函数与 Simulink 交互的三个模块

第一种交互方式的基本步骤如下：

(1) 按照规定格式编写 S-函数,如函数 my_sfunction,并保存为指定类型文件,如 my_sfunction.m 或 my_sfunction.c。

(2) 在 Simulink 的模型中增加 S-Function 模块,双击后将 S-function name 属性值设为 my_sfunction,如图 9-2 所示,同时可以设置参数等信息。

第二种交互方式的基本步骤如下：

(1) 按照规定格式编写 S-函数,如函数 my_sfunction,并保存为指定类型文件,如 my_sfunction.m。

(2) 在 Simulink 的模型中增加 Level-2 M-file S-Function 模块,双击后将 M-file name 属性值设为 my_sfunction,如图 9-3 所示,同时可以设置参数等信息。

第三种交互方式的基本步骤将在 9.5 节中详细介绍。

通过上面介绍的三种交互方式,可以将 S-函数看做 Simulink 自带的用户可自定义的模块。下面通过一个例子说明 S-函数的功能

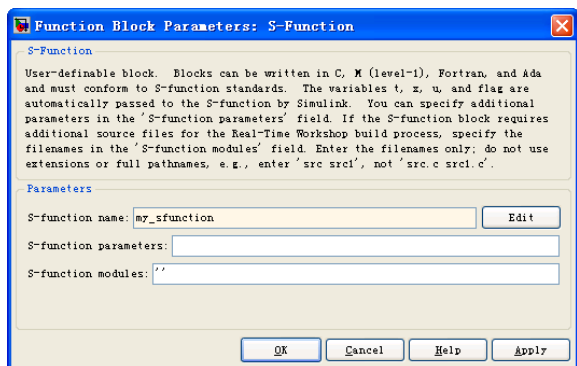


图 9-2 S-Function 模块的交互

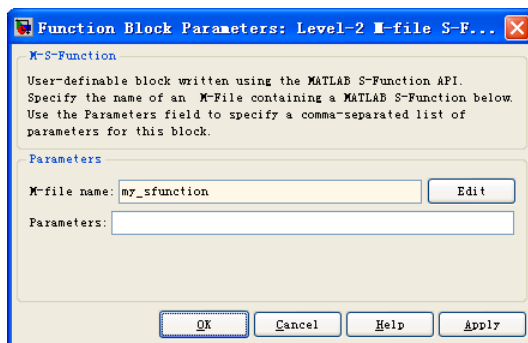


图 9-3 Level-2 M-file S-Function 模块的交互

【例 9-1】用 Level-1 M 文件型 S-函数（MySfunction1.m）描述方程
$$\begin{cases} \dot{x} = x \cos(tx) + u \\ x(0) = 1 \\ y = 3x \end{cases}, \text{ 其}$$

内容如下：

```
function [sys,x0,str,ts] = MySfunction1(t,x,u,flag)

switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case {2, 4, 9}
        sys = [];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates    = 1;           %表示仅一个连续状态 x
sizes.NumDiscStates    = 0;
sizes.NumOutputs       = 1;           %表示仅一个输出 x
sizes.NumInputs        = 1;           %表示仅一个输入 u
sizes.DirFeedthrough   = 0;
sizes.NumSampleTimes   = 0;
sys = simsizes(sizes);
x0  = 1;                             %表示 x(0)=1
```

```

str = [];
ts = [];

function sys=mdlDerivatives(t,x,u)
sys=x*cos(t*x)+u;           %表示 x 导数的表达式

function sys=mdlOutputs(t,x,u)
sys = 3*x;                  %表示输出 y=3*x

```

在 Simulink 中, 使用该 S-函数的实例如图 9-4 所示, 即对于给定输入 u , 计算状态 $x(t)$ 的轨迹并输出, 运行后双击 Scope 模块可得到如图 9-5 所示的结果。

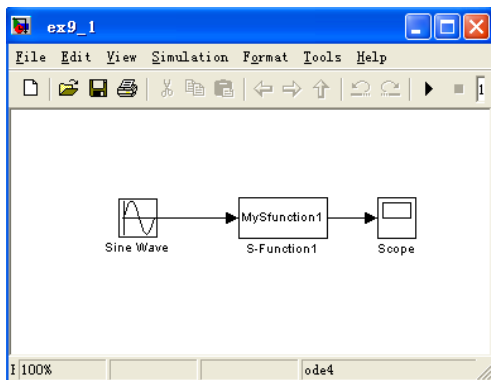


图 9-4 S-函数应用实例

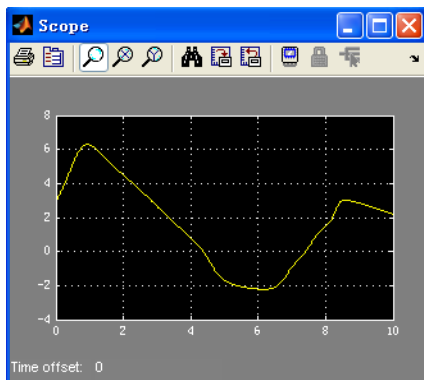


图 9-5 S-函数应用实例的运行结果

由本例的运行结果可以看出, S-函数提供了强大的数学描述功能, 将 Simulink 中复杂的建模过程转变为简单的代码编制。

9.2 工作原理

本节将从 S-函数的优点、数学描述、工作流程和回调函数几个方面介绍 S-函数的工作原理。

1. S-函数的优点

使用 S-函数, 可通过它创建通用性模块, 反复多次调用并可以每次使用不同的参数值, 具体来说用户可以实现以下操作:

- 可通过多种语言来创建新的通用性的 Simulink 模块。
- 只需遵循一系列简单的规则, 用户即可将自己的算法用 S-函数实现。当写好 S-函数后, 可在 User-Defined Functions 模块库的 S-function 模块中通过名称来调用已编写的 S-函数, 并且进行封装。
- 可通过 S-函数将一个系统描述成一个数学方程。
- 便于使用图形化仿真。
- 可创建代表硬件驱动模块。

2. S-函数的数学描述

一个 S-函数表示输入和输出间的函数映射关系, 其中包含状态的描述, 如图 9-6 所示。

描述一个 S-函数包含三个显性的元素，即输入 u 、状态 x 和输出 y ，以及两个隐性的元素，即时间 t 和采样时间 T （可以由 Simulink 仿真环境设置），同时输入 u 、状态 x 和输出 y 都可以是时间 t 和采样时间 T 的函数。S-函数可以方便地实现如下操作：

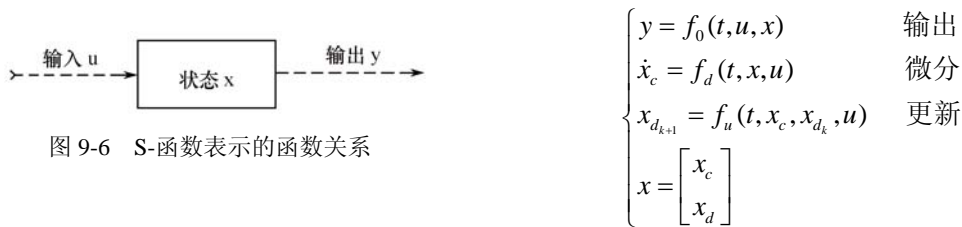


图 9-6 S-函数表示的函数关系

3. S-函数的工作流程

Simulink 将 S-函数的上述操作划分为不同的仿真步骤，分别是计算模块的输出，计算连续状态的微分，计算离散状态的更新。同时 Simulink 仿真的开始和结束，还包括初始化和结束处理两个步骤。

S-函数按照图 9-7 所示的流程执行。首先是初始化阶段，在这一阶段 Simulink 将库模块集合到模型，设置模块端口的数据宽度、数据类型和采样时间，评估模块参数，确定模块执行顺序，分配内存；其次进入仿真阶段，此时 Simulink 进入仿真循环，每一次仿真可以称为一个仿真步，在每一个仿真步中，Simulink 按初始化阶段确定的顺序执行各个模块，对每个模块，Simulink 计算模块在当前采样时间的状态、微分和输出，直至仿真结束。

4. S-函数的回调函数

一个 S-函数由实现指定操作的一系列 S-函数的回调函数组成，也可以看做由一系列子函数组成。在运行不同仿真步骤时，Simulink 调用相应的 S-函数回调函数来执行每个仿真步骤所需的任务。S-函数回调函数的名称将在后面小节中详细解释，这里首先介绍仿真步骤的含义。

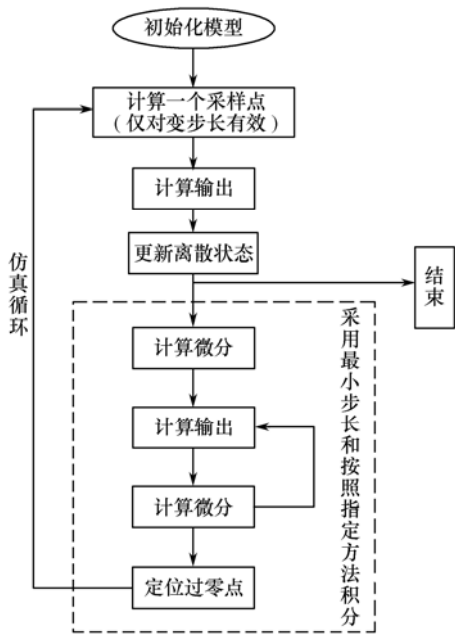


图 9-7 S-函数执行流程图

（1）初始化：在进入仿真循环前，Simulink 需要初始化 S-函数。在此阶段，Simulink 主要完成以下任务：

- 初始化 SimStruct，这是一种 Simulink 结构，包含了 S-函数的信息。
- 确定输入/输出端口的数目和大小。
- 确定模块的采样时间。
- 分配内存和数组长度。

（2）计算下一个采样点：如果模型使用变步长解法器，则需要确定下一个采样点的时刻，即下一个仿真步的大小；如果模型使用定步长解法器，此步将忽略，直接进入下一步骤。

（3）计算当前主仿真步的输出：计算当前主仿真步的模块所有输出端口的值，如仿真结束将其输出。

（4）更新当前主仿真步的离散状态：更新所有模块当前主仿真步的离散状态。

(5) 积分：只有当模型具有连续状态或者非采样过零点时，Simulink 才会有这一仿真步骤，同时 Simulink 按最小时间步长来调用 S-函数输出和微分的回调函数以及过零点部分。

9.3 Level-1 M 文件型

用 M 语言编写的 S-函数称为 M 文件 S-函数，根据 API 版本不同，分为 Level-1 M 文件型 S-函数和 Level-2 M 文件型 S-函数。

本节将通过 Level-1 M 文件型 S-函数的概述、编写方法以及大量的实例来介绍该类型的 S-函数，以满足用户不同的需求。

9.3.1 概述

在 MATLAB 命令窗口中输入命令 `sfundemos`，可以查看 S-函数示例，如图 9-8 所示，其中列出了用多种方式书写的 S-函数的实例。

双击“M-files”，可以看到如图 9-9 所示的界面，其中列出了用两种方式书写的 M 文件型 S-函数的实例。

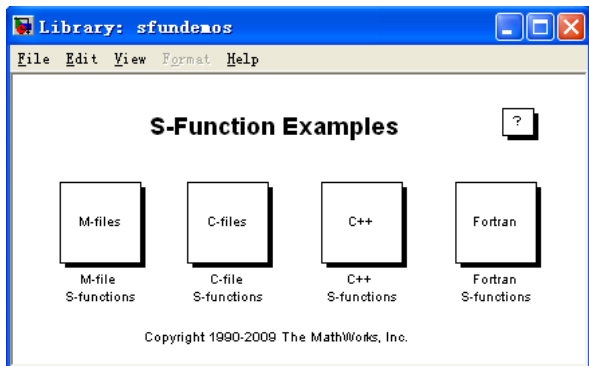


图 9-8 S-函数实例演示模块

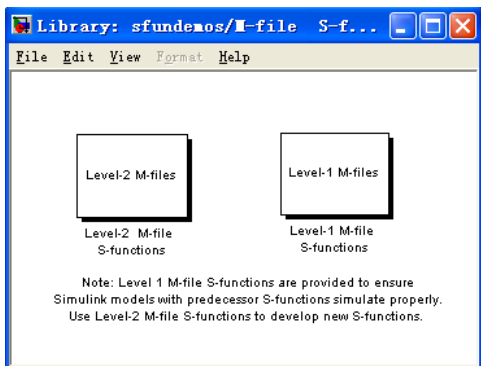


图 9-9 M 文件 S-函数实例演示模块

继续双击“Level-1 M-files”，可以看到如图 9-10 所示的界面，其中列出了 Level-1 M 文件型 S-函数的大量实例。

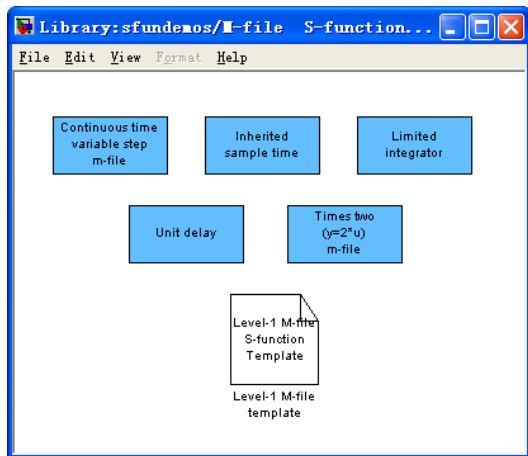


图 9-10 Level-1 M 文件型 S-函数实例演示模块

图 9-10 中包含一个 Level-1 M-file S-function Template 示例，它提供了书写该类型 S-函数的模板，删除注释后的具体内容如下：

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)

switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 2,
        sys=mdlUpdate(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys=mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 0;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0 = [];
str = [];
ts = [0 0];

function sys=mdlDerivatives(t,x,u)
```

```
sys = [];  
  
function sys=mdlUpdate(t,x,u)  
sys = [];  
  
function sys=mdlOutputs(t,x,u)  
sys = [];  
  
function sys=mdlGetTimeOfNextVarHit(t,x,u)  
sampleTime = 1;  
sys = t + sampleTime;  
  
function sys=mdlTerminate(t,x,u)  
sys = [];
```

9.3.2 编写方法

本小节主要针对 9.3.1 节中模板的各部分加以详细解释。编写 S-函数就是根据需求，用相应的代码去代替模板中对应部分的代码。

(1) 函数声明

第一行“function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)”是函数的声明，其中各参数的含义如表 9-1 所示。

表 9-1 各参数的含义

参数名	参数含义
sfuntmpl	S-函数的名称，用户可换成期望的函数名
t	当前仿真时间
x	状态向量
u	输入向量
flag	用以标示 S-函数当前所处的仿真步骤，以便执行相应的回调函数，取值为 0、1、2、3、4 或 9，同时 5 保留以便扩展
sys	不同的 flag 值，sys 返回值的含义不同： <ul style="list-style-type: none">• flag=0，sys 返回系统描述• flag=1，sys 返回微分结果• flag=2，sys 返回更新结果• flag=3，sys 返回输出结果• flag=4，sys 返回下一个采样点• flag=9，sys 返回空值
x0	只有在 flag=0 时返回初始状态值
str	只有在 flag=0 时返回状态字符串
ts	只有在 flag=0 时返回采样时间

同时完整的函数声明如下所示：


```
[sys,x0,str,ts] = sfunc(t,x,u,flag,p1,...,pn)
```

其中 $p1,...,pn$ 为 S-函数模块的参数。

（2）回调函数调用

在仿真过程中，Simulink 重复地调用 S-函数，并根据不同仿真步骤为参数 $flag$ 赋予不同的值，以便调用指定的回调函数。表 9-2 列出了参数 $flag$ 与 S-函数回调函数间的对应关系，以及各回调函数的说明。

表 9-2 参数 $flag$ 与 S-函数回调函数的关系

flag 取值	S-函数回调函数	说明
flag = 0	mdlInitializeSizes	初始化
flag = 1	mdlDerivatives	计算微分
flag = 2	mdlUpdate	更新离散状态
flag = 3	mdlOutputs	计算输出
flag = 4	mdlGetTimeOfNextVarHit	计算下一个采样时间
flag = 9	mdlTerminate	结束仿真

语句 “`DASudio.error('Simulink:blocks:unhandledFlag', num2str(flag));`” 用于在 S-函数运行出错时显示报错信息。

（3）回调函数 mdlInitializeSizes

回调函数 `mdlInitializeSizes` 中的 `Sizes` 是一个结构体，其各个字段的含义如表 9-3 所示。

其中直接馈入是指系统的输出或可变采样时间是否受到输入的控制，如 $y=ku$ （ u 是输入， k 是放大因子， y 是输出）是直接馈入的， $y=kx$ （ x 是状态， k 是放大因子， y 是输出）不是直接馈入的。

表 9-3 Sizes 结构体的各字段含义

字段名	含义
sizes.NumContStates	连续状态的个数
sizes.NumDiscStates	离散状态的个数
sizes.NumOutputs	输出的数目（所有输出向量的宽度之和）
sizes.NumInputs	输入的数目（所有输入向量的宽度之和）
sizes.DirFeedthrough	有无直接馈入，等于 0 表示没有，等于 1 表示有
sizes.NumSampleTimes	采样时间的个数，至少是一个采样时间

下述语句的含义是， $x0$ 表示状态的初始值， str 始终是空矩阵， ts 为 $n \times 2$ 的矩阵，其中 n 为采样时间的个数，每个采样时间的设置如表 9-4 所示。

```
x0 = [];  
str = [];  
ts = [0 0];
```

表 9-4 ts 每个采样时间的设置

取值	含义
[0 0]	连续采样时间
[0 1]	连续采样时间，以最小步长运算
[period offset]	定步长离散采样时间，period 为周期，offset 为偏置量，且 $0 < \text{offset} < \text{period}$ 此时第 m 步的采样时间 $T_m = m * \text{period} + \text{offset}$ 。
[-2 0]	变步长离散采样时间，采样时间由 flag=4 的回调函数设置
[-1 0]	在最小步长内改变的函数并且采用驱动模块的采样时间
[-1 1]	在最小步长内不改变的函数并且采用驱动模块的采样时间

(4) 回调函数 mdlDerivatives

下述语句的含义是：sys 表示状态的导数 \dot{x} ，如 $x = (x_1 \ x_2 \ \cdots \ x_n)^T$ ，则 $\text{sys} = (\text{sys}(1) \ \text{sys}(2) \ \cdots \ \text{sys}(n))^T$ ，其中 $\text{sys}(i) = \dot{x}_i$ ($i=1, 2, \cdots, n$)；等号右边的“[]”表示导数的表达式。

```
sys = [];
```

(5) 回调函数 mdlUpdate

下述语句的含义是：sys 表示状态的下一步取值 $x(n+1)$ ，同理 sys 可以为向量；等号右边的“[]”表示下一步状态值的表达式。

```
sys = [];
```

(6) 回调函数 mdlOutputs

下述语句的含义是，sys 表示输出 y ，同理 sys 可以为向量；等号右边的“[]”表示输出的表达式。

```
sys = [];
```

(7) 回调函数 mdlGetTimeOfNextVarHit

下述语句的含义是，sampleTime 表示采样步长，sys 表示下一步的采样时间， $t + \text{sampleTime}$ 的位置为下一步采样时间的表达式，这里表示下一步的采样时间为在当前采样时间的基础上增加 1 秒。

```
sampleTime = 1;
sys = t + sampleTime;
```

(8) 回调函数 mdlTerminate

下述语句表示 sys 输出为空值。

```
sys = [];
```

9.3.3 实例

例 9-1 给出了 Level-1 M 文件型 S-函数的简单实例，下面通过不同类型的实例进一步说明 Level-1 M 文件型 S-函数的编写方法和编写技巧。

1. 状态、输入和输出多维的情况

例 9-1 是状态、输入和输出都是一维的例子，例 9-2 给出一个状态、输入和输出都是多维的例子。

【例 9-2】用 Level-1 M 文件型 S-函数（MySfunction2.m）描述方程

$$\begin{cases} \dot{x}_1 = 2 * x_2 \\ \dot{x}_2 = 3 * x_3 + u_1 \\ \dot{x}_3 = x_1^2 \cos(tx_2) x_3 u_2 \end{cases}, \text{ 其内容如下:}$$

$$\begin{cases} x_1(0) = 1, x_2(0) = 1, x_3(0) = 2 \\ y = [x_1 \quad 2 * x_2 \quad x_1 + x_3 \quad x_1 + 2 * x_2] \end{cases}$$

```
function [sys,x0,str,ts] = MySfunction2(t,x,u,flag)

switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 2,
        sys=mdlUpdate(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys=mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates = 3;    %3 个状态
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;    %4 个输出
sizes.NumInputs = 2;    %2 个输入
sizes.DirFeedthrough = 0;    %无直接馈入
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0 = [1;1;2];                %初始条件[1 1 2]
```

```

str = [];
ts = [0 0]; %连续采样时间

function sys=mdlDerivatives(t,x,u) %描述微分方程
sys(1)=2*x(2); %等价于  $dx_1=2*x_2$ 
sys(2)=3*x(3)+u(1); %等价于  $dx_2=3*x_3+u_1$ 
sys(3)=x(1)^2*cos(t*x(2))*x(3)*u(2); %等价于  $dx_3=x_1^2*cos(t*x_2)*x_3*u_2$ 

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u) %描述输出方程
sys(1)=x(1); %等价于  $y_1=x_1$ 
sys(2)=2*x(2); %等价于  $y_2=2*x_2$ 
sys(3)=x(1)+x(3); %等价于  $y_3=x_1+x_3$ 
sys(4)=x(1)+2*x(2); %等价于  $y_4=x_1+2*x_2$ 
function sys=mdlGetTimeOfNextVarHit(t,x,u) %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];

```

在 Simulink 中,使用该 S-函数的实例如图 9-11 所示,即对于给定输入 u 计算输出 y ,运行后双击 Scope 模块可得到如图 9-12 所示的结果。

由于输入是二维的,在图 9-11 中使用了一个 Mux 模块将两个一维输入联合后作为 S-函数的输入;如果信号本身是二维的,可以直接连接并作为 S-函数的输入;如果信号本身是大于二维的,必须经过适当的处理,保证 S-函数的输入是且必须是二维的。

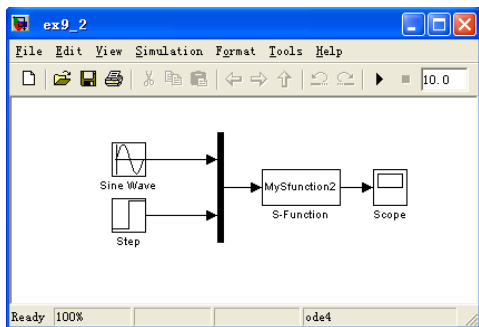


图 9-11 Simulink 仿真框图

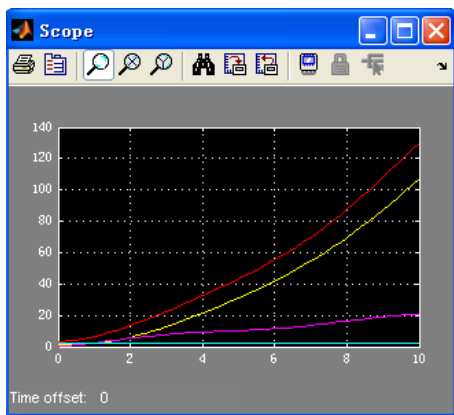


图 9-12 包含 4 个输出的仿真曲线

由于输出是 4 维的,所以可得到如图 9-12 所示的 4 条输出曲线。

需要说明的是, S-函数保存的文件名和函数名要一致,保存 M-文件 S-函数的目录要包含

在 MATLAB 7.10 的搜索路径文件之中。

2. 采用直接馈入的方式

例 9-3 实现状态空间的表示，并利用了直接馈入的方式。

【例 9-3】用 Level-1 M 文件型 S-函数（MySfunction3.m）描述方程
$$\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases} \quad (\text{其中})$$

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \text{它等价于} \begin{cases} \dot{x}_1 = x_1 \\ \dot{x}_2 = x_2 \\ \dot{x}_3 = -x_1 + x_2 + 4x_3 + u \\ x_1(0) = 0, \quad x_2(0) = 0, \quad x_3(0) = 0 \\ y_1 = 2x_1 + u \\ y_2 = x_2 + u \end{cases},$$

其内容如下：

```
function [sys,x0,str,ts] = MySfunction3(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 2,
    sys=mdlUpdate(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
case 9,
    sys=mdlTerminate(t,x,u);
otherwise
    DASTudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates = 3;      %3 个状态
sizes.NumDiscStates = 0;
```

```

sizes.NumOutputs    = 2;      %2 个输出
sizes.NumInputs     = 1;      %1 个输入
sizes.DirFeedthrough = 1;      %有直接馈入
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0  = [0;0;0];                %初始条件[0 0 0]
str = [];
ts   = [0 0];                  %连续采样时间

function sys=mdlDerivatives(t,x,u) %描述微分方程

%紧凑描述
A=[1 0 0;0 1 0;-1 1 4];
B=[0;0;1];
sys=A*x+B*u;                   %等价于  $\dot{X}=AX+Bu$ 

%%展开描述
%sys(1)=x(1);                  %等价于  $\dot{x}_1=x_1$ 
%sys(2)=x(2);                  %等价于  $\dot{x}_2=x_2$ 
%sys(3)=-x(1)+x(2)+4*x(3)+u;   %等价于  $\dot{x}_3=-x_1+x_2+4*x_3+u$ 

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u) %描述输出方程

%紧凑描述
C=[2 0 0;0 1 0];
D=[1;1];
sys=C*x+D*u;                   %等价于  $Y=CX+Du$ 

%%展开描述
%sys(1)=2*x(1)+u;              %等价于  $y_1=2*x_1+u$ 
%sys(2)=x(2)+u;                %等价于  $y_2=x_2+u$ 

function sys=mdlGetTimeOfNextVarHit(t,x,u) %将输出置空
sys = [];

```

```
function sys=mdlTerminate(t,x,u)
```

```
sys = [];
```

在 Simulink 中, 使用该 S-函数的实例如图 9-13 所示, 运行后双击 Scope 模块可得到如图 9-14 所示的结果。

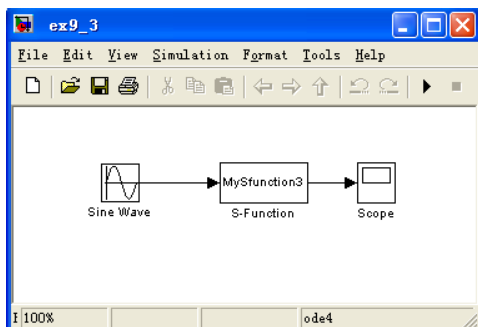


图 9-13 Simulink 仿真框图

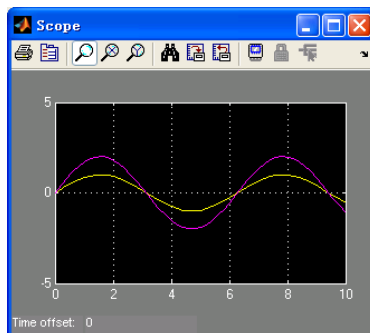


图 9-14 Simulink 仿真运行结果

本例中的语句 `sizes.DirFeedthrough = 1`; 若改为 `sizes.DirFeedthrough = 0`;, 系统则报错, 因为回调函数 `mdlOutputs` 中利用了输出的信息, 而 `sizes.DirFeedthrough = 0`; 表示无直接馈入。

同时本例中给出了方程的两种表示方式, 这两种表示方式是等价的, 不仅是数学描述上等价, 运行结果也是相同的, 读者可以自行验证。

3. 参数传递

前面的例子都没有涉及函数参数传递的问题, 包括 S-函数本身的参数传递, 以及 S-函数与其回调函数间的参数传递。下面的例子说明了这两类函数参数的传递方法。

【例 9-4】用 Level-1 M 文件型 S-函数 (MySfunction4.m) 描述方程
$$\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases} \quad (\text{其中})$$

$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ w_1 & w_2 & w_3 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $D = \begin{pmatrix} 1 \\ q \end{pmatrix}$, $W = (w_1 \ w_2 \ w_3)$, 其内容如下:

```
function [sys,x0,str,ts] = MySfunction4(t,x,u,flag,W,q)
```

```
A=[0 1 0;0 1 0;W(1) W(2) W(3)]; %声明在本函数及其回调函数中有效的变量, 其中使用到外部参数向量 W
```

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

```
C=[1 0 0;0 1 0];
```

```
D=[1;q]; %其中使用到外部参数 q
```

```
switch flag,
```

```
case 0,
```

```
    [sys,x0,str,ts]=mdlInitializeSizes(P,q); %附加外部参数, 即使不使用也需要附加
```

```
case 1,
```

```
    sys=mdlDerivatives(t,x,u,A,B); %附加使用到的参数
```

```

case 2,
    sys=mdlUpdate(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u,C,D); %附加使用到的参数
case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
case 9,
    sys=mdlTerminate(t,x,u);
otherwise
    DASudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes(P,q); %附加外部参数

sizes = simsizes;

sizes.NumContStates = 3;    %3 个状态
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;    %2 个输出
sizes.NumInputs = 1;    %1 个输入
sizes.DirFeedthrough = 1;    %有直接馈入
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0 = [0;0;0];                %初始条件[0 0 0]
str = [];
ts = [0 0];                %连续采样时间

function sys=mdlDerivatives(t,x,u,A,B) %附加使用到的参数，描述微分方程
    sys=A*x+B*u;                %等价于  $\dot{X}=AX+Bu$ 

function sys=mdlUpdate(t,x,u)
    sys = [];

function sys=mdlOutputs(t,x,u,C,D)    %附加使用到的参数，描述输出方程
    sys=C*x+D*u;                %等价于  $Y=CX+Du$ 

function sys=mdlGetTimeOfNextVarHit(t,x,u) %将输出置空

```



```
sys = [];  
  
function sys=mdlTerminate(t,x,u)  
  
sys = [];
```

在 Simulink 中, 使用该 S-函数的实例如图 9-15 所示。当 $W=(w1\ w2\ w3)=(-1\ -2\ -3)$ 和 $q=1$ 时, 双击 S-Function 模块, 如图 9-16 所示设置 S-function parameters 属性即可, 运行后双击 Scope 模块可得到如图 9-17 所示的结果。

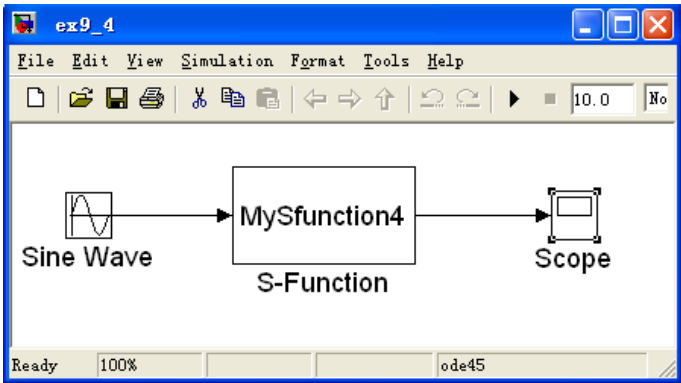


图 9-15 Simulink 仿真框图

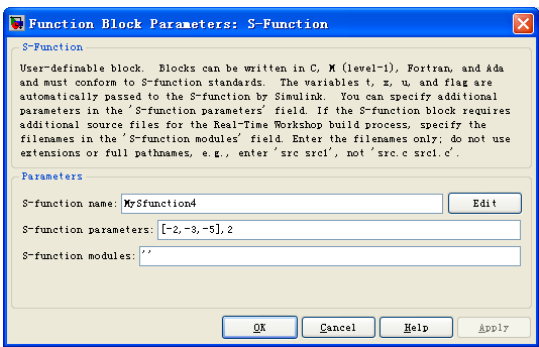


图 9-16 S-函数参数设置

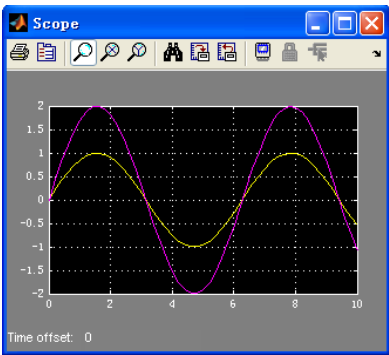


图 9-17 Simulink 仿真运行结果

这里需要说明如下两点:

- (1) 对于外部参数传递, 多个参数的写法是 $w1, w2, w3, \dots, wn$; 若其中包含参数向量或者矩阵, 对应参数可以通过中括号对表示成相应格式, 如本例。
- (2) 对于内部参数传递, 回调函数的声明和书写中都要附加被传递的参数名。

4. 离散系统

前面的例子都是关于连续系统的, 下面给出一个离散系统的例子。

【例 9-5】用 Level-1 M 文件型 S-函数 (MySfunction5.m) 描述方程

$$\begin{cases} X(n+1) = AX(n) + Bu(n) \\ X(0) = 0 \\ Y(n) = CX(n) + Du(n) \end{cases}$$

(其中 $A = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 3 & 3 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $D = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$), 其内容如下:

```
function [sys,x0,str,ts] = MySfunction5(t,x,u,flag)

switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 2,
        sys=mdlUpdate(t,x,u);
    case 3,
        sys=mdlOutputs(t,x,u);
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys=mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates = 0;
sizes.NumDiscStates = 3;    %3 个离散状态
sizes.NumOutputs = 2;    %2 个输出
sizes.NumInputs = 1;    %1 个输入
sizes.DirFeedthrough = 1;    %有直接馈入
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

x0 = [0;0;0];                %初始条件[0 0 0]
str = [];
ts = [0.1 0];                %定步长离散采样时间，采样周期为 0.1，偏置量为 0
```

```
function sys=mdlDerivatives(t,x,u)
sys=[];

function sys=mdlUpdate(t,x,u)           %描述差分方程

A=[0 -1 0;0 0 -1; 1 3 3];
B=[0;0;1] ;
sys=A*x+B*u;                           %等价于 X(n+1)=AX(n)+Bu(n)

function sys=mdlOutputs(t,x,u)         %描述输出方程

C=[1 0 0;0 1 0];
D=[1;2];
sys=C*x+D*u;                           %等价于 Y(n)=CX(n)+Du(n)

function sys=mdlGetTimeOfNextVarHit(t,x,u) %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-18 所示，运行后双击 Scope 模块可得到如图 9-19 所示的结果。

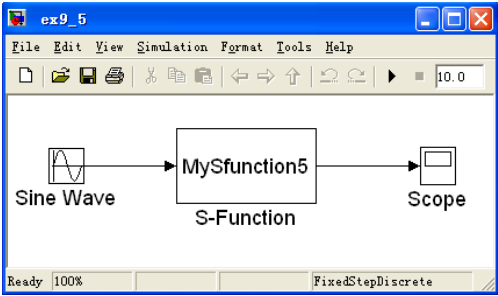


图 9-18 Simulink 仿真框图

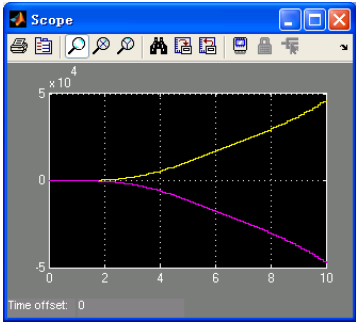


图 9-19 Simulink 仿真运行结果

本例给定了采样周期和偏置量，仿真结果也显示采样周期为 0.1s，偏置量为 0。

5. 混合系统

混合系统指既有连续又有离散状态的系统，下面给出一个混合系统的例子，其中还涉及到多个仿真时间的处理。

【例 9-6】用 Level-1 M 文件型 S-函数（MySfunction6.m）描述方程

$$\begin{cases} \dot{x}_1 = 7 * u \\ x_2(n+1) = x_2(n) + x_1 \quad (T=1) \\ x_1(0) = 0, x_2(0) = 0 \\ y = [x_1 \quad x_2]^T \end{cases}, \text{ 其内容如下:}$$

```
function [sys,x0,str,ts] = MySfunction6(t,x,u,flag)

dperiod = 1;           %离散状态的采样周期
doffset = 0;           %离散状态的采样时间偏置量

switch flag,
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset); %附加使用到的参数
    case 1,
        sys=mdlDerivatives(t,x,u);
    case 2,
        sys=mdlUpdate(t,x,u,dperiod,doffset); %附加使用到的参数
    case 3,
        sys=mdlOutputs(t,x,u);
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys=mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset) %附加使用到的参数

sizes = simsizes;

sizes.NumContStates = 1; %1 个连续状态
sizes.NumDiscStates = 1; %1 个离散状态
sizes.NumOutputs = 2; %2 个输出
sizes.NumInputs = 1; %1 个输入
sizes.DirFeedthrough = 0; %无直接馈入
sizes.NumSampleTimes = 2; %2 个采样时间
```

```

sys = simsizes(sizes);

x0 = [0;0];           %初始条件[0 0]
str = [];
ts = [0 0;           %两个采样时间，一个是连续采样时间
      dperiod,doffset]; %一个是采样周期为 1，偏置量为 0

function sys=mdlDerivatives(t,x,u)           %描述微分方程
sys=7*u;                                     %等价于  $dx=2u$ 

function sys=mdlUpdate(t,x,u,dperiod,doffset) %附加使用到的参数，描述差分方程

t_times=(t - doffset)/dperiod;               %计算当前时间扣除偏置量后相对采样周期的倍数
t_err= abs(round(t_times)-t_times)           %比较与正倍数的距离
if t_err < 1e-8                               %距离足够小则认为时刻位于采样点上，此时修正 x(n+1)，否则 x(n+1)值不变
    sys =x(2)+ x(1);                         %等价于  $x(n+1)=x(n)+x1$ 
else
    sys = [];
end

function sys=mdlOutputs(t,x,u)               %描述输出方程
sys=x;                                       %等价于  $y=[x1;x2]$ 

function sys=mdlGetTimeOfNextVarHit(t,x,u)   %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];

```

在 Simulink 中，使用该 S-函数的实例如图 9-20 所示，运行后双击 Scope 模块可得到如图 9-21 所示的结果。

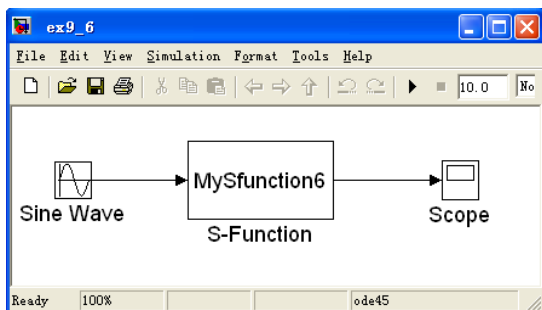


图 9-20 Simulink 仿真框图

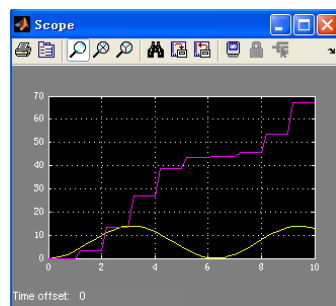


图 9-21 Simulink 仿真运行结果

由本例可以看出,处理混合系统时,需要对离散状态的更新做特殊处理,令其在指定时间点上操作,其他时刻不更新。

9.4 Level-2M 文件型

上一节从多个角度详细介绍了 Level-1 M 文件型 S-函数的编写方法,因为这是当前应用最广、用户使用最熟练的方式。目前, MATLAB 在保留 Level-1 M 文件型 S-函数的基础上,力推 Level-2 M 文件型 S-函数,并提供了由 Level-1 M 文件型到 Level-2 M 文件型 S-函数的转换方法。本节将着重介绍 Level-2 M 文件型 S-函数。

9.4.1 概述

与查看 Level-1 M 文件型 S-函数演示程序的过程类似,可以通过操作看到如图 9-9 所示的界面,继续双击“Level-2 M-files”,可以看到如图 9-22 所示的界面,其中列出了 Level-2 M 文件型 S-函数的大量实例。

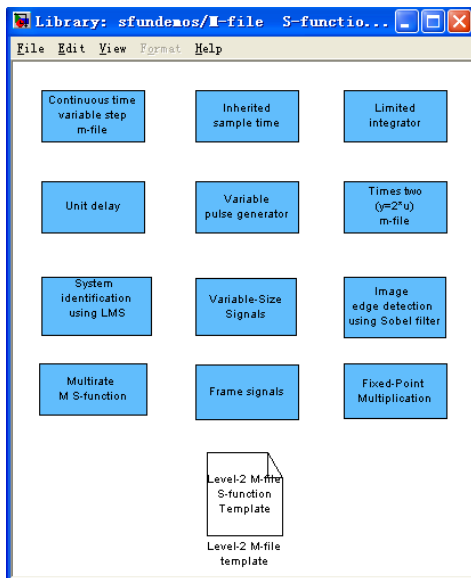


图 9-22 Level-2 M 文件型 S-函数实例演示模块

图 9-22 中包含一个 Level-2 M-file S-function Template 示例,它提供了书写该类型 S-函数的模板,删除注释后的具体内容如下:

```
function msfuntmpl(block)

setup(block);

function setup(block)
    block.NumInputPorts = 1;
    block.NumOutputPorts = 1;
```

```

block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

block.InputPort(1).DatatypeID = 0;
block.InputPort(1).Complexity = 'Real';

block.OutputPort(1).DatatypeID = 0; % double
block.OutputPort(1).Complexity = 'Real';

block.NumDialogPrms = 3;
block.DialogPrmsTunable = {'Tunable','Nontunable','SimOnlyTunable'};

block.SampleTimes = [0 0];

block.SetAccelRunOnTLC(false);

block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
block.RegBlockMethod('SetInputPortDataType', @SetInpPortDataType);
block.RegBlockMethod('SetOutputPortDataType', @SetOutPortDataType);
block.RegBlockMethod('SetInputPortComplexSignal', @SetInpPortComplexSig);
block.RegBlockMethod('SetOutputPortComplexSignal', @SetOutPortComplexSig);
block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('ProcessParameters', @ProcessPrms);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs);
block.RegBlockMethod('Update', @Update);
block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Projection', @Projection);
block.RegBlockMethod('SimStatusChange', @SimStatusChange);
block.RegBlockMethod('Terminate', @Terminate);
block.RegBlockMethod('WriteRTW', @WriteRTW);

```

```

function CheckPrms(block)
a = block.DialogPrm(1).Data;
if ~strcmp(class(a), 'double')
    DAStudio.error('Simulink:block:invalidParameter');

```

```
end

function ProcessPrms(block)
    block.AutoUpdateRuntimePrms;

function SetInpPortFrameData(block, idx, fd)
    block.InputPort(idx).SamplingMode = fd;
    block.OutputPort(1).SamplingMode = fd;

function SetInpPortDims(block, idx, di)
    block.InputPort(idx).Dimensions = di;
    block.OutputPort(1).Dimensions = di;

function SetOutPortDims(block, idx, di)
    block.OutputPort(idx).Dimensions = di;
    block.InputPort(1).Dimensions = di;

function SetInpPortDataType(block, idx, dt)
    block.InputPort(idx).DataTypeID = dt;
    block.OutputPort(1).DataTypeID = dt;

function SetOutPortDataType(block, idx, dt)
    block.OutputPort(idx).DataTypeID = dt;
    block.InputPort(1).DataTypeID = dt;

function SetInpPortComplexSig(block, idx, c)
    block.InputPort(idx).Complexity = c;
    block.OutputPort(1).Complexity = c;

function SetOutPortComplexSig(block, idx, c)
    block.OutputPort(idx).Complexity = c;
    block.InputPort(1).Complexity = c;

function DoPostPropSetup(block)
    block.NumDworks = 1;

    block.Dwork(1).Name = 'x1';
    block.Dwork(1).Dimensions = 1;
    block.Dwork(1).DatatypeID = 0; % double
    block.Dwork(1).Complexity = 'Real'; % real
```



```
block.Dwork(1).UsedAsDiscState = true;

block.AutoRegRuntimePrms;

function InitializeConditions(block)

function Start(block)
    block.Dwork(1).Data = 0;

function WriteRTW(block)
    block.WriteRTWParam('matrix', 'M', [1 2; 3 4]);
    block.WriteRTWParam('string', 'Mode', 'Auto');

function Outputs(block)
    block.OutputPort(1).Data = block.Dwork(1).Data + block.InputPort(1).Data;

function Update(block)
    block.Dwork(1).Data = block.InputPort(1).Data;

function Derivatives(block)

function Projection(block)

function SimStatusChange(block, s)
    if s == 0
        disp('Pause has been called');
    elseif s == 1
        disp('Continue has been called');
    end

function Terminate(block)
```

9.4.2 编写方法

本小节主要针对9.4.1节中模板的各部分加以详细解释。编写S-函数就是根据需求，用相应的代码去代替模板中对应部分的代码。

1. 两种 M 文件型 S-函数的关系

前面的 9.3.2 节详细讲解了 Level-1 M 文件型 S-函数模板中的各部分，下面先介绍如何由 Level-1 M 文件型转换为 Level-2 M 文件型 S-函数。

首先，以 Level-1 M 文件型 S-函数的参数 flag 为基准，建立 Level-1 M 文件型和 Level-2 M 文件型 S-函数两个模板间的对应关系，如表 9-5 所示。

表 9-5 两种 M 文件型 S-函数模板间基于参数 flag 的对应关系

Level-1 M 文件型	Level-2 M 文件型
function [sys,x0,str,ts]= MySfunction (t,x,u,flag)	function MySfunction(block)
case 0, [sys,x0,str,ts] = mdlInitializeSizes;	setup(block)
case 1, sys=mdlDerivatives(t,x,u);	block.RegBlockMethod('Derivatives', @Derivatives);
case 2, sys = mdlUpdate(t,x,u);	block.RegBlockMethod('Update',@Update);
case 3, sys = mdlOutputs(t,x,u);	block.RegBlockMethod('Outputs' ,@Output);
case 4, sys=mdlGetTimeOfNextVarHit(t,x,u);	模板中无对应语句设置，可通过如下回调函数实现： block.RegBlockMethod('Outputs' ,@Output)
case 9, sys=mdlTerminate(t,x,u);	block.RegBlockMethod('Terminate', @Terminate)

这里需要说明如下两点：

- Level-2 M 文件型 S-函数不支持过零检验。
- function [sys,x0,str,ts]= MySfunction (t,x,u,flag)中的参数 flag 在 Level-2 M 文件型 S-函数中已经不存在，参数 t 对应 block.CurrentTime，参数 x 对应 block.Dwork，参数 u 对应 block.InputPort.Data，参数 ts 对应 block.SampleTimes，参数 str 已经不存在，参数 x0 的内容将在后面讲解，参数 sys 在系统输出时对应 block.OutputPort.Data。

其次，以Level-1 M文件型S-函数的回调函数mdlInitializeSizes为基准，建立Level-1 M文件型和Level-2 M文件型S-函数两个模板间的对应关系，如表9-6所示。

表 9-6 两种 M 文件型 S-函数模板间基于回调函数 mdlInitializeSizes 的对应关系

Level-1 M 文件型	Level-2 M 文件型
sizes.NumContStates = 0;	function DoPostPropSetup(block) block.NumContStates = 1; %声明连续状态数 block.AutoRegRuntimePrms; %将所有可调参数注册为运行参数
sizes.NumDiscStates = 0;	function DoPostPropSetup(block) block.NumDworks = 1; %声明离散状态数 block.Dwork(1).Name= 'x1'; %状态名 block.Dwork(1).Dimensions= 1; %状态维数 block.Dwork(1).DatatypeID= 0; %状态数据类型，0 为 double 型 block.Dwork(1).Complexity = 'Real'; %状态数据类型，'Real'为实数 block.Dwork(1).UsedAsDiscState = true; %状态类型 %true 为离散，false 为连续 block.AutoRegRuntimePrms;%将所有可调参数注册为运行参数
sizes.NumOutputs = 0;	function setup(block)
sizes.NumInputs = 0;	block.NumInputPorts = 1;%设置输入端口数，每个端口可包含多个输入 block.NumOutputPorts = 1; %设置输出端口数，每个端口可包含多个输出 block.InputPort(1).DatatypeID = 0; %第一个输入端口数据类型，double 型 block.InputPort(1).Complexity = 'Real'; %数据类型，实数型 block.OutputPort(1).DatatypeID=0;%第一个输出端口数据类型，double 型 block.OutputPort(1).Complexity = 'Real'; %数据类型，实数型 模板中无对应语句设置端口的输入数目，可以在 function setup(block)中如下设置： block.InputPort(1).Dimensions = 2; %第一个输入端口包含 2 个输入

续表

Level-1 M 文件型	Level-2 M 文件型
sizes.DirFeedthrough = 1;	模板中无对应语句设置，可以在 function setup(block)中如下设置： block.InputPort(1).DirectFeedthrough = true;
sizes.NumSampleTimes = 1;	模板中无对应语句设置，在 block.SampleTimes 中隐含
x0 = [];	在 function InitializeConditions(block)中设置
str = [];	模板中无对应语句设置
ts = [0 0];	function setup(block) block.SampleTimes = [0 0];

最后，以Level-1 M文件型S-函数的回调函数为基准，建立Level-1 M文件型和Level-2 M文件型S-函数两个模板间的对应关系，如表9-7所示。

表 9-7 两种 M 文件型 S-函数模板间基于回调函数的对应关系

Level-1 M 文件型	Level-2 M 文件型
function sys=mdlDerivatives(t,x,u)	function Derivatives (block)
sys=[];	block.Derivatives.Data=[]
function sys=mdlUpdate(t,x,u)	function Update(block)
sys=[];	block.Dwork.Data =[];
function sys=mdlOutputs(t,x,u)	function Outputs(block)
sys=[];	block.OutputPort.Data =[];
function sys=mdlGetTimeOfNextVarHit(t,x,u)	功能包含在 function Outputs(block)中
function sys=mdlTerminate(t,x,u)	function Terminate(block)

综上，按照Level-1 M文件型书写模板，给出了其中语句及参数与Level-2 M文件型S-函数的对应关系。这为Level-1 M文件型S-函数的移植提供了方法，同时也介绍了Level-2 M文件型S-函数模板的有关内容。

2. Level-2 M 文件型 S-函数的特性

Level-2 M文件型S-函数模板中除了具有前面介绍的与Level-1 M文件型相同的属性外，还有其特有的属性。

首先，介绍回调函数setup中的新增语句，如表9-8所示。

表 9-8 回调函数 setup 中的新增语句

语句	含义
block.SetPreCompInpPortInfoToDynamic; block.SetPreCompOutPortInfoToDynamic;	设置输入/输出端口的属性为继承性或动态
block.NumDialogPrms = 3; block.DialogPrmsTunable = { 'Tunable','Nontunable','SimOnlyTunable'};	设置外部参数和参数类型（可调、不可调或仅仿真可调）
block.SetAccelRunOnTLC(false);	指定是否使用TLC文件生成仿真目标文件

续表

语句	含义
block.RegBlockMethod('CheckParameters', @CheckPrms);	新增回调函数声明
block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);	
block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);	
block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);	
block.RegBlockMethod('SetInputPortDataType', @SetInpPortDataType);	
block.RegBlockMethod('SetOutputPortDataType', @SetOutPortDataType);	
block.RegBlockMethod('SetInputPortComplexSignal', @SetInpPortComplexSig);	
block.RegBlockMethod('SetOutputPortComplexSignal', @SetOutPortComplexSig);	
block.RegBlockMethod('ProcessParameters', @ProcessPrms);	
block.RegBlockMethod('Start', @Start);	
block.RegBlockMethod('Projection', @Projection);	
block.RegBlockMethod('SimStatusChange', @SimStatusChange);	
block.RegBlockMethod('WriteRTW', @WriteRTW);	

其次，介绍新增的回调函数，如表9-9所示。

表 9-9 新增的回调函数

名称	含义
function CheckPrms(block)	检查参数的合法性
function ProcessPrms(block)	更改运行参数值
function SetInpPortFrameData(block, idx, fd)	检查和设置输入端口属性
function SetInpPortDims(block, idx, di)	检查和设置输入端口维数
function SetOutPortDims(block, idx, di)	检查和设置输出端口维数
function SetInpPortDataType(block, idx, dt)	检查和设置输入端口数据类型
function SetOutPortDataType(block, idx, dt)	检查和设置输出端口数据类型
function SetInpPortComplexSig(block, idx, c)	检查和设置输入端口是否为复数属性
function SetOutPortComplexSig(block, idx, c)	检查和设置输出端口是否为复数属性
function Start(block)	初始化状态和工作空间变量值
function WriteRTW(block)	向 RTW 文件写入指定信息
function Projection(block)	仿真步中更新预测值
function SimStatusChange(block, s)	仿真暂停时（s=0）和重启时（s=1）调用

综上，详细介绍了Level-2 M文件型S-函数模板的内容，这为Level-2 M文件型S-函数的书写提供了便利。

9.4.3 实例

下面通过不同类型的实例进一步说明 Level-2 M 文件型 S-函数的编写方法和编写技巧。

1. 连续系统

这里给出一个连续系统的例子。

【例 9-7】用 Level-2 M 文件型 S-函数（MySfunction7.m）描述方程

$$\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases} \quad (\text{其中})$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -5 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, D = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \text{ 其内容如下:}$$

```
function MySfunction7 (block)

setup(block);

function setup(block)
    block.NumInputPorts = 1;
    block.NumOutputPorts = 1;

    block.SetPreCompInpPortInfoToDynamic;
    block.SetPreCompOutPortInfoToDynamic;

    block.InputPort(1).DatatypeID = 0; % double
    block.InputPort(1).Complexity = 'Real';
    block.InputPort(1).Dimensions = 1;
    block.InputPort(1).DirectFeedthrough = true; %有直接馈入

    block.OutputPort(1).DatatypeID = 0; % double
    block.OutputPort(1).Complexity = 'Real';
    block.OutputPort(1).Dimensions = 2; %一个输出端口包含两个输出

    block.SampleTimes = [0 0];

    block.SetAccelRunOnTLC(false);

    %使用到的回调函数
    block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
    block.RegBlockMethod('InitializeConditions', @InitializeConditions);
    block.RegBlockMethod('Outputs', @Outputs);
    block.RegBlockMethod('Derivatives', @Derivatives);

    function DoPostPropSetup(block)
        block.NumContStates = 3; %3个连续状态

        block.AutoRegRuntimePrms;

    function InitializeConditions(block)
        block.ContStates.Data = [0;0;0];
```

```

function Outputs(block)
C=[1 0 0;0 1 0];
D=[1;2];
block.OutputPort(1).Data =C* block.ContStates.Data + D*block.InputPort(1).Data;

function Derivatives(block)
A=[0 1 0;0 1 0;-2 -3 -5];
B=[0;0;1];
block.Derivatives.Data =A* block.ContStates.Data +B* block.InputPort(1).Data;

```

在Simulink中，使用该S-函数的实例如图9-23所示。

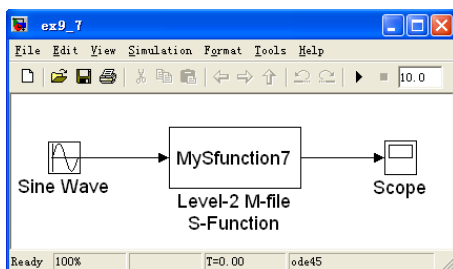


图 9-23 Simulink 仿真框图

本例给出的是针对连续状态的仿真，而模板给出的是针对离散状态的仿真。这里需要说明的是，离散状态与连续状态在表述上存在较大差异，在应用时要特别注意。

2. 离散系统

这里给出一个离散系统的例子，同时将外部参数传入 S-函数。

【例 9-8】用 Level-2 M 文件型 S-函数(MySfunction8.m)描述方程

$$\begin{cases} X(n+1) = AX(n) + Bu(n) \\ X(0) = 0 \\ Y(n) = CX(n) + Du(n) \end{cases}$$

(其中 $A = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ e1 & e2 & e3 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $D = \begin{pmatrix} 1 \\ q \end{pmatrix}$, $E = (e1 \ e2 \ e3)$), 其内容如下:

```

function MySfunction8(block)

setup(block);

function setup(block)
    block.NumInputPorts = 1;
    block.NumOutputPorts = 1;

    block.SetPreCompInpPortInfoToDynamic;
    block.SetPreCompOutPortInfoToDynamic;

```

```
block.InputPort(1).DatatypeID = 0;
block.InputPort(1).Complexity = 'Real';
block.InputPort(1).Dimensions = 1;
block.InputPort(1).DirectFeedthrough = true; %有直接馈入

block.OutputPort(1).DatatypeID = 0; % double
block.OutputPort(1).Complexity = 'Real';
block.OutputPort(1).Dimensions = 2; %一个输出端口包含两个输出

block.NumDialogPrms = 2; %外部传入两个参数 E、q
block.DialogPrmsTunable = {'Tunable', 'Tunable'};

block.SampleTimes = [0.1 0]; %采样周期为 0.1，偏置量为 0

block.SetAccelRunOnTLC(false);

%使用到的回调函数
block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('Outputs', @Outputs);
block.RegBlockMethod('Update', @Update);

function DoPostPropSetup(block)
    block.NumDworks = 3; %3 个离散状态

    block.Dwork(1).Name = 'x1'; %第 1 个离散状态
    block.Dwork(1).Dimensions = 1;
    block.Dwork(1).DatatypeID = 0; % double
    block.Dwork(1).Complexity = 'Real'; % real
    block.Dwork(1).UsedAsDiscState = true;

    block.Dwork(2).Name = 'x2'; %第 2 个离散状态
    block.Dwork(2).Dimensions = 1;
    block.Dwork(2).DatatypeID = 0; % double
    block.Dwork(2).Complexity = 'Real'; % real
    block.Dwork(2).UsedAsDiscState = true;

    block.Dwork(3).Name = 'x3'; %第 3 个离散状态
    block.Dwork(3).Dimensions = 1;
    block.Dwork(3).DatatypeID = 0; % double
    block.Dwork(3).Complexity = 'Real'; % real
    block.Dwork(3).UsedAsDiscState = true;
```

```
block.AutoRegRuntimePrms;
```

```
function InitializeConditions(block)
```

```
block.Dwork(1).Data = 0; %为第 1 个离散状态赋值
```

```
block.Dwork(2).Data = 0; %为第 2 个离散状态赋值
```

```
block.Dwork(3).Data = 0; %为第 3 个离散状态赋值
```

```
function Outputs(block)
```

```
block.OutputPort(1).Data(1)=block.Dwork(1).Data + block.InputPort(1).Data;
```

```
block.OutputPort(1).Data(2)=block.Dwork(2).Data + block.DialogPrm(2).Data*block.InputPort(1).Data;
```

```
% block.DialogPrm(2).Data 表示第 2 个参数 q 的值
```

```
function Update(block)
```

```
block.Dwork(1).Data =-block.Dwork(2).Data;
```

```
block.Dwork(2).Data =-block.Dwork(3).Data;
```

```
block.Dwork(3).Data = block.DialogPrm(1).Data(1)* block.Dwork(1).Data+ ...
```

```
block.DialogPrm(1).Data(2)* block.Dwork(2).Data+ ...
```

```
block.DialogPrm(1).Data(3)* block.Dwork(3).Data+ block.InputPort(1).Data;
```

```
% block.DialogPrm(1).Data(1)表示第 1 个参数的第一个分量 p1 的值
```

在 Simulink 中,使用该 S-函数的实例如图 9-24 所示。当 $E = (e1 \ e2 \ e3) = (1 \ 3 \ 3)$ 和 $q = 2$ 时,双击 S-Function 模块,如图 9-25 所示设置 S-function parameters 属性即可,且运行结果如图 9-26 所示。

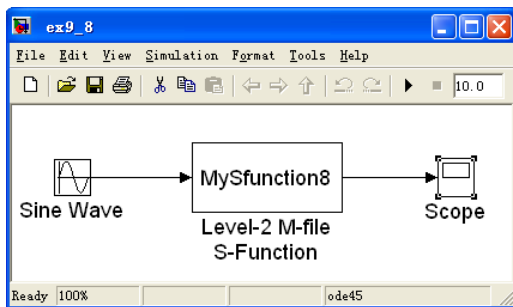


图 9-24 Simulink 仿真框图

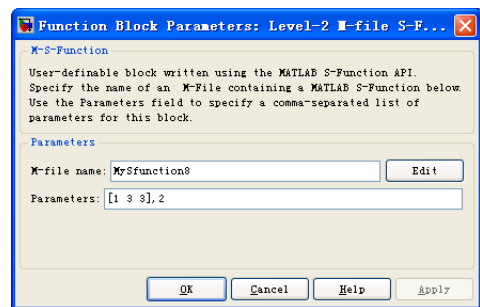


图 9-25 S-函数参数设置

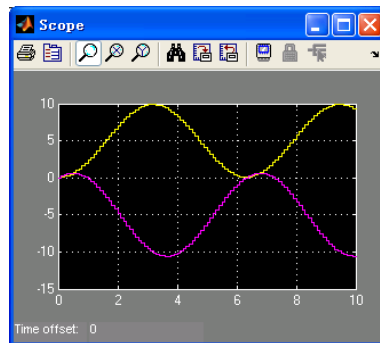


图 9-26 Simulink 仿真运行结果

本例给定了采样周期和偏置量，仿真结果也显示采样周期为 0.1s，偏置量为 0。

9.5 C MEX 文件型

前面两节详细介绍了两类 M 文件型 S-函数的编写方法，本节着重介绍 C MEX 文件型 S-函数的编写方法。除了 C MEX 文件型，还有 C++ MEX、Fortran MEX 文件型 S-函数。在这些 MEX 文件型 S-函数中，C MEX 文件型是最常用的。

9.5.1 概述

1. MEX 文件简介

在 MATLAB 中，可调用的 C 或 Fortran 语言程序称为 MEX 文件，MATLAB 可以直接把 MEX 文件视为它的内建函数进行调用。MEX 文件是动态链接的例程，MATLAB 解释器可以自动载入并执行它。MEX 文件主要有以下几方面的应用：

- 在 MATLAB 中，M 文件的计算速度特别是循环迭代的速度远比 C 语言慢，因此可以把要求大量循环迭代的部分用 C 语言编写为 MEX 文件，以提高计算速度。
- 对于已经开发的 C 语言程序，则不必将其转化为 M 文件而重复劳动，通过添加入口程序 mexFunction，即可由 MATLAB 调用。
- 直接控制硬件，如 A/D 采集卡、D/A 输出卡等，以用于数据采集或控制应用。

2. C MEX 文件简介

C MEX 文件，就是基于 C 语言编写的 MEX 文件，是 MATLAB 应用程序接口的一个重要组成部分。通过它不但可以将现有的使用 C 语言编写的函数轻松地引入 MATLAB 环境中使用，避免了重复的程序设计，而且可以使用 C 语言为 MATLAB 定制用于特定目的的函数，以完成在 MATLAB 中不易实现的任务，同时还可以使用 C 语言提高 MATLAB 环境中数据的处理效率。

C 语言的 MEX 文件的源程序由两个非常明显的部分组成：

- 计算程序，即在 MEX 文件中完成计算功能的程序代码。计算可以是普通的 C 语言程序，按照 C 语言规则编写即可。
- 入口程序，将计算程序与 MATLAB 连接的入口函数 mexFunction。

需要注意的是，MEX 文件虽然具有较强大的功能，但并不是对所有的应用都恰当。MATLAB 是一个高效率的编程系统，特别适合于工程计算、系统仿真等应用，它的最大优点就是将人们从繁杂的程序中解放出来。因此，能够用 M 文件完成的程序，应尽量使用 MATLAB 编写，除非遇到必须使用 MEX 文件的情况。

与查看 Level-1 和 Level-2 M 文件型 S-函数演示程序的过程类似，可以通过操作看到如图 9-8 所示的界面，继续双击“C-files”，可以看到如图 9-27 所示的界面，其中列出了 C MEX 文件型 S-函数的大量实例。

图 9-27 中包含一个 Basic C-MEX Template 和一个 Detailed C-MEX Template 示例，一个是为了基本应用，另一个是为了高级应用。它们提供了书写该类型 S-函数的模板，删除注释后的具体内容如下：

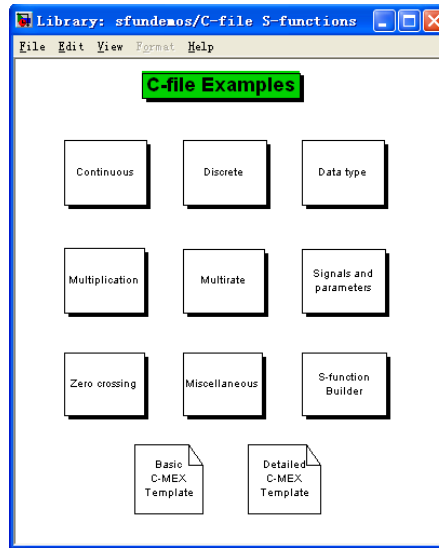


图 9-27 C MEX 文件型 S-函数实例演示模块

(1) Basic C-MEX Template

```
#define S_FUNCTION_NAME  sfuntmpl_basic
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true); /*direct input signal access*/

    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
}
```

```

        ssSetNumSampleTimes(S, 1);
        ssSetNumRWork(S, 0);
        ssSetNumIWork(S, 0);
        ssSetNumPWork(S, 0);
        ssSetNumModes(S, 0);
        ssSetNumNonsampledZCs(S, 0);

        ssSetOptions(S, 0);
    }

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS    /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)

    static void mdlInitializeConditions(SimStruct *S)
    {
    }
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START    /* Change to #undef to remove function */
#if defined(MDL_START)

    static void mdlStart(SimStruct *S)
    {
    }
#endif /* MDL_START */

static void mdlOutputs(SimStruct *S, int_T tid)
{
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T      *y = ssGetOutputPortSignal(S,0);
    y[0] = u[0];
}

```

```

#define MDL_UPDATE /* Change to #undef to remove function */
#if defined(MDL_UPDATE)

    static void mdlUpdate(SimStruct *S, int_T tid)
    {
    }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)

    static void mdlDerivatives(SimStruct *S)
    {
    }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif

```

(2) Detailed C-MEX Template

```

#define S_FUNCTION_NAME your_sfunction_name_here
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

#define MDL_CHECK_PARAMETERS /* Change to #undef to remove function */
#if defined(MDL_CHECK_PARAMETERS) && defined(MATLAB_MEX_FILE)
    static void mdlCheckParameters(SimStruct *S)
    {
    }
#endif /* MDL_CHECK_PARAMETERS */

#define MDL_PROCESS_PARAMETERS /* Change to #undef to remove function */

```

```

#if defined(MDL_PROCESS_PARAMETERS) && defined(MATLAB_MEX_FILE)
    static void mdlProcessParameters(SimStruct *S)
    {
    }
#endif /* MDL_PROCESS_PARAMETERS */

static void mdlInitializeSizes(SimStruct *S)
{
    int_T nInputPorts = 1; /* number of input ports */
    int_T nOutputPorts = 1; /* number of output ports */
    int_T needsInput = 1; /* direct feed through */

    int_T inputPortIdx = 0;
    int_T outputPortIdx = 0;

    ssSetNumSFcnParams(S, 0); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    ssSetNumContStates(S, 0); /* number of continuous states */
    ssSetNumDiscStates(S, 0); /* number of discrete states */

    if (!ssSetNumInputPorts(S, nInputPorts)) return;
    if (!ssSetInputPortDimensionInfo(S, inputPortIdx, DYNAMIC_DIMENSION)) return;
    ssSetInputPortDirectFeedThrough(S, inputPortIdx, needsInput);
    if (!ssSetNumOutputPorts(S, nOutputPorts)) return;
    if (!ssSetOutputPortDimensionInfo(S, outputPortIdx, DYNAMIC_DIMENSION)) return;

    ssSetNumSampleTimes(S, 1); /* number of sample times */

    ssSetNumRWork(S, 0); /* number of real work vector elements */
    ssSetNumIWork(S, 0); /* number of integer work vector elements */
    ssSetNumPWork(S, 0); /* number of pointer work vector elements */
    ssSetNumModes(S, 0); /* number of mode work vector elements */
    ssSetNumNonsampledZCs(S, 0); /* number of nonsampled zero crossings */

    ssSetOptions(S, 0); /* general options (SS_OPTION_xx) */

} /* end mdlInitializeSizes */

```

```
#define MDL_SET_INPUT_PORT_FRAME_DATA /* Change to #undef to remove function */
#if defined(MDL_SET_INPUT_PORT_FRAME_DATA) && defined(MATLAB_MEX_FILE)
    static void mdlSetInputPortFrameData(SimStruct *S,
                                         int portIndex,
                                         Frame_T frameData)
    {
    } /* end mdlSetInputPortFrameData */
#endif /* MDL_SET_INPUT_PORT_FRAME_DATA */

#define MDL_SET_INPUT_PORT_WIDTH /* Change to #undef to remove function */
#if defined(MDL_SET_INPUT_PORT_WIDTH) && defined(MATLAB_MEX_FILE)
    static void mdlSetInputPortWidth(SimStruct *S, int portIndex, int width)
    {
    } /* end mdlSetInputPortWidth */
#endif /* MDL_SET_INPUT_PORT_WIDTH */

#define MDL_SET_OUTPUT_PORT_WIDTH /* Change to #undef to remove function */
#if defined(MDL_SET_OUTPUT_PORT_WIDTH) && defined(MATLAB_MEX_FILE)
    static void mdlSetOutputPortWidth(SimStruct *S, int portIndex, int width)
    {
    } /* end mdlSetOutputPortWidth */
#endif /* MDL_SET_OUTPUT_PORT_WIDTH */

#define MDL_SET_INPUT_PORT_DIMENSION_INFO /* Change to #define to add function */
#if defined(MDL_SET_INPUT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
    static void mdlSetInputPortDimensionInfo(SimStruct *S,
                                              int_T portIndex,
                                              const DimsInfo_T *dimsInfo)
    {
    } /* mdlSetInputPortDimensionInfo */
#endif /* MDL_SET_INPUT_PORT_DIMENSION_INFO */

#define MDL_SET_OUTPUT_PORT_DIMENSION_INFO /* Change to #define to add function */
#if defined(MDL_SET_OUTPUT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
    static void mdlSetOutputPortDimensionInfo(SimStruct *S,
                                              int_T portIndex,
                                              const DimsInfo_T *dimsInfo)
    {
    }
#endif
```

```

    } /* mdlSetOutputPortDimensionInfo */
#endif /* MDL_SET_OUTPUT_PORT_DIMENSION_INFO */

#undef MDL_SET_DEFAULT_PORT_DIMENSION_INFO /* Change to #define to add fcn */
#if defined(MDL_SET_DEFAULT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
    static void mdlSetDefaultPortDimensionInfo(SimStruct *S)
    {
        } /* mdlSetDefaultPortDimensionInfo */
#endif /* MDL_SET_DEFAULT_PORT_DIMENSION_INFO */

#define MDL_SET_INPUT_PORT_SAMPLE_TIME
#if defined(MDL_SET_INPUT_PORT_SAMPLE_TIME) && defined(MATLAB_MEX_FILE)
    static void mdlSetInputPortSampleTime(SimStruct *S,
                                           int_T    portIdx,
                                           real_T    sampleTime,
                                           real_T    offsetTime)
    {
        } /* end mdlSetInputPortSampleTime */
#endif /* MDL_SET_INPUT_PORT_SAMPLE_TIME */

#define MDL_SET_OUTPUT_PORT_SAMPLE_TIME
#if defined(MDL_SET_OUTPUT_PORT_SAMPLE_TIME) && defined(MATLAB_MEX_FILE)
    static void mdlSetOutputPortSampleTime(SimStruct *S,
                                           int_T    portIdx,
                                           real_T    sampleTime,
                                           real_T    offsetTime)
    {
        } /* end mdlSetOutputPortSampleTime */
#endif /* MDL_SET_OUTPUT_PORT_SAMPLE_TIME */

static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* Register one pair for each sample time */
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);

} /* end mdlInitializeSampleTimes */

#define MDL_SET_INPUT_PORT_DATA_TYPE /* Change to #undef to remove function */
#if defined(MDL_SET_INPUT_PORT_DATA_TYPE) && defined(MATLAB_MEX_FILE)

```

```

static void mdlSetInputPortDataType(SimStruct *S, int portIndex,DTypeId dType)
{
} /* mdlSetInputPortDataType */
#endif /* MDL_SET_INPUT_PORT_DATA_TYPE */

#define MDL_SET_OUTPUT_PORT_DATA_TYPE /* Change to #undef to remove function */
#if defined(MDL_SET_OUTPUT_PORT_DATA_TYPE) && defined(MATLAB_MEX_FILE)
static void mdlSetOutputPortDataType(SimStruct *S,int portIndex,DTypeId dType)
{
} /* mdlSetOutputPortDataType */
#endif /* MDL_SET_OUTPUT_PORT_DATA_TYPE */

#define MDL_SET_DEFAULT_PORT_DATA_TYPES /* Change to #undef to remove function*/
#if defined(MDL_SET_DEFAULT_PORT_DATA_TYPES) && defined(MATLAB_MEX_FILE)
static void mdlSetDefaultPortDataTypes(SimStruct *S)
{
} /* mdlSetDefaultPortDataTypes */
#endif /* MDL_SET_DEFAULT_PORT_DATA_TYPES */

#define MDL_SET_INPUT_PORT_COMPLEX_SIGNAL /* Change to #undef to remove */
#if defined(MDL_SET_INPUT_PORT_COMPLEX_SIGNAL) && defined(MATLAB_MEX_FILE)
static void mdlSetInputPortComplexSignal(SimStruct *S,
                                         int      portIndex,
                                         CSignal_T cSignalSetting)
{
} /* mdlSetInputPortComplexSignal */
#endif /* MDL_SET_INPUT_PORT_COMPLEX_SIGNAL */

#define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL /* Change to #undef to remove */
#if defined(MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL) && defined(MATLAB_MEX_FILE)
static void mdlSetOutputPortComplexSignal(SimStruct *S,
                                         int      portIndex,
                                         CSignal_T cSignalSetting)
{
} /* mdlSetOutputPortComplexSignal */
#endif /* MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL */

#define MDL_SET_DEFAULT_PORT_COMPLEX_SIGNALS /* Change to #undef to remove */
#if defined(MDL_SET_DEFAULT_PORT_COMPLEX_SIGNALS) && defined(MATLAB_MEX_FILE)
static void mdlSetDefaultPortComplexSignals(SimStruct *S)

```



```

{
} /* mdlSetDefaultPortComplexSignals */
#endif /* MDL_SET_DEFAULT_PORT_COMPLEX_SIGNALS */

#define MDL_SET_WORK_WIDTHS    /* Change to #undef to remove function */
#if defined(MDL_SET_WORK_WIDTHS) && defined(MATLAB_MEX_FILE)
    static void mdlSetWorkWidths(SimStruct *S)
    {
    }
#endif /* MDL_SET_WORK_WIDTHS */

#define MDL_INITIALIZE_CONDITIONS    /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
    static void mdlInitializeConditions(SimStruct *S)
    {
    }
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START    /* Change to #undef to remove function */
#if defined(MDL_START)
    static void mdlStart(SimStruct *S)
    {
    }
#endif /* MDL_START */

#define MDL_GET_TIME_OF_NEXT_VAR_HIT    /* Change to #undef to remove function */
#if defined(MDL_GET_TIME_OF_NEXT_VAR_HIT) && (defined(MATLAB_MEX_FILE) || \
                                              defined(NRT))

    static void mdlGetTimeOfNextVarHit(SimStruct *S)
    {
        time_T timeOfNextHit = ssGetT(S) /* + offset */ ;
        ssSetTNext(S, timeOfNextHit);
    }
#endif /* MDL_GET_TIME_OF_NEXT_VAR_HIT */

#define MDL_ZERO_CROSSINGS    /* Change to #undef to remove function */
#if defined(MDL_ZERO_CROSSINGS) && (defined(MATLAB_MEX_FILE) || defined(NRT))
    static void mdlZeroCrossings(SimStruct *S)
    {
    }

```

```

#endif /* MDL_ZERO_CROSSINGS */

static void mdlOutputs(SimStruct *S, int_T tid)
{
} /* end mdlOutputs */

#define MDL_UPDATE /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
    static void mdlUpdate(SimStruct *S, int_T tid)
    {
    }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
    static void mdlDerivatives(SimStruct *S)
    {
    }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}

#define MDL_RTW /* Change to #undef to remove function */
#if defined(MDL_RTW) && defined(MATLAB_MEX_FILE)
    static void mdlRTW(SimStruct *S)
    {
    }
#endif /* MDL_RTW */

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfuns.h" /* Code generation registration function */
#endif

```

9.5.2 编写方法

本小节主要针对 9.5.1 小节 Basic C-MEX 模板中的各部分加以详细解释。编写 S-函数就是根据需求，用相应的代码去代替模板中对应部分的代码。

1. Level-2 M 文件型和 C MEX 文件型 S-函数的关系

Level-2 M 文件型与 C MEX 文件型 S-函数在书写上非常近似，表 9-10 列出了两者在回调函数上的等价关系。

表 9-10 Level-2 M 文件型与 C MEX 文件型 S-函数回调函数的等价关系

Level-2 M 文件型	C MEX 文件型
setup	mdlInitializeSizes
CheckParameters	mdlCheckParameters
Derivatives	mdlDerivatives
Disable	mdlDisable
Enable	mdlEnable
InitializeCondition	mdlInitializeConditions
Outputs	mdlOutputs
PostPropagationSetup	mdlSetWorkWidths
ProcessParameters	mdlProcessParameters
Projection	mdlProjection
SetInputPortComplexSignal	mdlSetInputPortComplexSignal
SetInputPortDataType	mdlSetInputPortDataType
SetInputPortDimensions	mdlSetInputPortDimensionInfo
SetInputPortSampleTime	mdlSetInputPortSampleTime
SetInputPortSamplingMode	mdlSetInputPortFrameData
SetOutputPortComplexSignal	mdlSetOutputPortComplexSignal
SetOutputPortDataType	mdlSetOutputPortDataType
SetOutputPortDimensions	mdlSetOutputPortDimensionInfo
SetOutputPortSampleTime	mdlSetOutputPortSampleTime
SimStatusChange	mdlSimStatusChange
Start	mdlStart
Terminate	mdlTerminate
Update	mdlUpdate
WriteRTW	mdlRTW

2. C MEX 文件型 S-函数的特性

C MEX 文件型 S-函数的编写满足 C 语言的规范，这是与前面 M 文件型 S-函数的最大区别。下面具体解释各段代码的含义。

(1) S-函数声明

```
#define S_FUNCTION_NAME    MySfunction /*命名函数名 MySfunction*/
#define S_FUNCTION_LEVEL 2 /*指定 LEVEL 2 类型*/
```

(2) 导入支持 SimStruct 的库文件

```
#include "simstruc.h"
```

(3) 模块初始化回调函数

```
static void mdlInitializeSizes(SimStruct *S) /*S 相当于 Level-2 M 文件型 S-函数中的 block*/
```

```

{
    ssSetNumSFcnParams(S, 0); /*设置外部参数个数*/
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /*期望参数个数不等于实际参数个数的处理方法*/
        return;
    }

    ssSetNumContStates(S, 0); /* 设置连续状态个数*/
    ssSetNumDiscStates(S, 0); /* 设置离散状态个数*/

    if (!ssSetNumInputPorts(S, 1)) return; /* ssSetNumInputPorts(S, 1)设置输入端口个数为 1*/
    ssSetInputPortWidth(S, 0, 1); /*设置第一个输入端口的输入个数, C 语言中第一个元素的下标为 0*/
    ssSetInputPortRequiredContiguous(S, 0, true); /*设置第一个输入端口始终输入*/

    ssSetInputPortDirectFeedThrough(S, 0, 1); /*设置第一个输入端口有直接馈入*/

    if (!ssSetNumOutputPorts(S, 1)) return; /* ssSetNumOutputPorts(S, 1)设置输出端口个数为 1*/
    ssSetOutputPortWidth(S, 0, 1); /*设置第一个输出端口的输入个数*/

    ssSetNumSampleTimes(S, 1); /*设置采样时间个数为 1*/
    ssSetNumRWork(S, 0); /* 设置实数元素个数*/
    ssSetNumIWork(S, 0); /* 设置整数元素个数*/
    ssSetNumPWork(S, 0); /* 设置指针元素个数*/
    ssSetNumModes(S, 0); /* 设置模式元素个数*/
    ssSetNumNonsampledZCs(S, 0); /* 设置非采样过零个数*/

    ssSetOptions(S, 0); /*具体选项设置见 simstruc.h*/
}

```

(4) 采样时间初始化回调函数

```

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME); /*为第一个采样时间设置采样周期*/
    ssSetOffsetTime(S, 0, 0.0); /*为第一个采样时间设置偏置量*/
}

```

(5) 状态初始值设置回调函数

```

#define MDL_INITIALIZE_CONDITIONS /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
    static void mdlInitializeConditions(SimStruct *S)
    {

```

/*模板中没有如何设置状态初始值的示例语句，可以采用如下方法设置：

```
*real_T *xc0 = ssGetContStates(S);    *设定连续状态初始值的指针，进而赋值
*real_T *xd0 = ssGetRealDiscStates(S); *设定离散状态初始值的指针，进而赋值
*/
}
```

```
#endif /* MDL_INITIALIZE_CONDITIONS */
```

(6) 模块启动设置回调函数

```
#define MDL_START /* Change to #undef to remove function */
```

```
#if defined(MDL_START)
```

```
static void mdlStart(SimStruct *S)
```

```
{
}
```

```
#endif /* MDL_START */
```

(7) 输出回调函数

```
static void mdlOutputs(SimStruct *S, int_T tid)
```

```
{
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0); /*将第一个输入端口的值赋给变量*/
    real_T        *y = ssGetOutputPortSignal(S,0); /*声明第一个输出端口值所对应的指针*/
    y[0] = u[0];    /*将待输出的值赋给第一个输出端口*/
}
```

(8) 更新回调函数

```
#define MDL_UPDATE /* Change to #undef to remove function */
```

```
#if defined(MDL_UPDATE)
```

```
static void mdlUpdate(SimStruct *S, int_T tid)
```

```
{
```

/*模板中没有如何使用的示例语句，下面给出一个演示：

```
*real_T *x = ssGetRealDiscStates(S); *将连续状态的值赋给指定变量
```

```
*const real_T *u = (const real_T*) ssGetInputPortSignal(S,0); *将第一个输入端口的值赋给变量
```

```
*进而对 x 进行赋值
```

```
*/
```

```
}
```

```
#endif /* MDL_UPDATE */
```

(9) 微分回调函数

```
#define MDL_DERIVATIVES /* Change to #undef to remove function */
```

```
#if defined(MDL_DERIVATIVES)
```

```
static void mdlDerivatives(SimStruct *S)
```

```
{
```

/*模板中没有如何使用的示例语句，下面给出一个演示：

```
*real_T *dx = ssGetdX(S); *将连续状态的导数值赋给指定变量
```

```
*real_T *x = ssGetContStates(S); *将连续状态的值赋给指定变量
```

```

    *const real_T *u = (const real_T*) ssGetInputPortSignal(S,0); *将第一个输入端口的值赋给变量
    *进而对 dx 进行赋值
    */
}
#endif /* MDL_DERIVATIVES */

```

(10) 终止回调函数

```

static void mdlTerminate(SimStruct *S)
{
}

```

(11) S-函数结尾

```

#ifdef MATLAB_MEX_FILE /*是否被编译为 MEX 文件 */
#include "simulink.c" /*导入 simulink.c */
#else
#include "cg_sfun.h" /*导入 simulink.cg_sfun.h*/
#endif

```

综上，详细介绍了 C MEX 文件型 S-函数模板的内容，这为 C MEX 文件型 S-函数的书写提供了便利。

9.5.3 实例

前面介绍了 C MEX 文件型 S-函数的写法，下面通过不同类型的实例进一步说明 C MEX 文件型 S-函数的编写方法和编写技巧。

1. 连续系统

下面给出一个连续系统的例子。

【例 9-9】用 C MEX 文件型 S-函数（MySfunction9.c）描述方程
$$\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases} \quad (\text{其中},$$

$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $D = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$), 其内容如下:

```

#define S_FUNCTION_NAME MySfunction9
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(S, 3);

    if (!ssSetNumInputPorts(S, 1)) return;

```

```

    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 2);

    ssSetNumSampleTimes(S, 1);

    ssSetOptions(S, 0);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetContStates(S);
    int_T i;

    //X0=0
    for (i=0; i<3; i++) {
        x0[i] = 0;
    }
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    real_T *x = ssGetContStates(S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    //Y=CX+Du
    y[0] = x[0] + u[0];
    y[1] = x[1] + 2*u[0];
}

#define MDL_DERIVATIVES

```

```

static void mdlDerivatives(SimStruct *S)
{
    real_T      *dx    = ssGetdX(S);
    real_T      *x      = ssGetContStates(S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    //dX=AX+Bu
    dx[0] =  x[1];
    dx[1] =  x[2];
    dx[2] = -x[0]-2*x[1]-3*x[2]+u[0];
}

static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif

```

在仿真前，需要首先使用如下指令编译MySfunction9.c:

```
mex MySfunction9.c
```

此时，MySfunction9.c所在目录下增加了一个文件MySfunction9.mexw32。

在Simulink中，使用该S-函数的实例如图9-28所示，运行后双击Scope模块可得到如图9-29所示的结果。

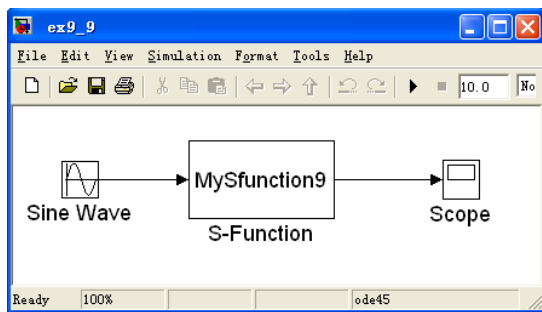


图 9-28 Simulink 仿真框图

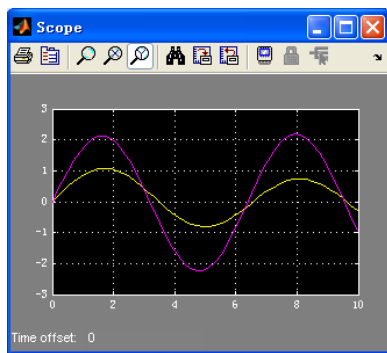


图 9-29 Simulink 仿真运行结果

本例需要说明的是，上述代码不支持中文注释。

2. 离散系统

下面给出一个离散系统的例子，同时将外部参数传入 S-函数。

【例 9-10】用 C MEX 文件型 S-函数 (MySfunction10.c) 描述方程

$$\begin{cases} X(n+1) = AX(n) + Bu(n) \\ X(0) = 0 \\ Y(n) = CX(n) + Du(n) \end{cases}$$

(其中 $A = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ d1 & d2 & d3 \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, $D = \begin{pmatrix} 1 \\ q \end{pmatrix}$, $D = (d1 \ d2 \ d3)$), 其内容如

下:

```
#define S_FUNCTION_NAME MySfunction10
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 2); /* 2 parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    ssSetNumDiscStates(S, 3);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 2);

    ssSetNumSampleTimes(S, 1);

    ssSetOptions(S, 0);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, 0.1);
    ssSetOffsetTime(S, 0, 0.0);
}
```

```

}

#define MDL_INITIALIZE_CONDITIONS    /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
    static void mdlInitializeConditions(SimStruct *S)
    {
        real_T *x0 = ssGetRealDiscStates (S);
        int_T i;

        //X0=0
        for (i=0; i<3; i++) {
            x0[i] = 0;
        }
    }
#endif /* MDL_INITIALIZE_CONDITIONS */

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T      *y      = ssGetOutputPortRealSignal(S,0);
    real_T      *x      = ssGetRealDiscStates (S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T *p2= mxGetPr(ssGetSFcnParam(S,1));
    //Y=CX+Du
    y[0] = x[0] + u[0];
    y[1] = x[1] + p2[0]* u[0];
}

#define MDL_UPDATE    /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
    static void mdlUpdate(SimStruct *S, int_T tid)
    {
        real_T      *y      = ssGetOutputPortRealSignal(S,0);
        real_T      *x      = ssGetRealDiscStates (S);
        real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
        real_T *p1= mxGetPr(ssGetSFcnParam(S,0));
        real_T      tmpx[3]={ 0,0,0};
        //X(k+1)=AX(k)+Bu(k)
        tmpx[0]=-x[1];
        tmpx[1]=-x[2];

```

```
tmpx[2]=p1[0]*x[0]+p1[1]*x[1]+ p1[2]*x[2]+u[0];
x[0]=tmpx[0];
x[1]=tmpx[1];
x[2]=tmpx[2];
}

#endif /* MDL_UPDATE */

static void mdlTerminate(SimStruct *S)
{
}
```

```
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

在仿真前，需要首先使用如下指令编译MySfunction10.c:

```
mex MySfunction10.c
```

此时，MySfunction10.c所在目录下增加了一个文件MySfunction10.mexw32。

在 Simulink 中，使用该 S-函数的实例如图 9-29 所示。当 $D=(d1\ d2\ d3)=(1\ 3\ 3)$ 和 $q=2$ 时，双击 S-Function 模块，如图 9-30 所示设置 S-function parameters 属性即可，且运行结果如图 9-31 所示。

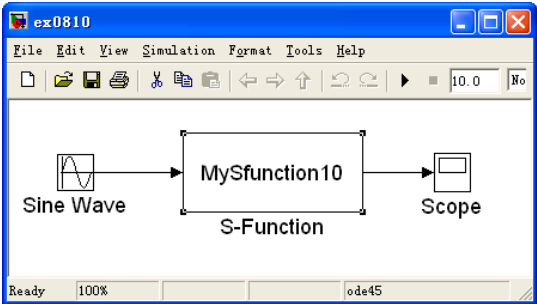


图 9-29 Simulink 仿真框图

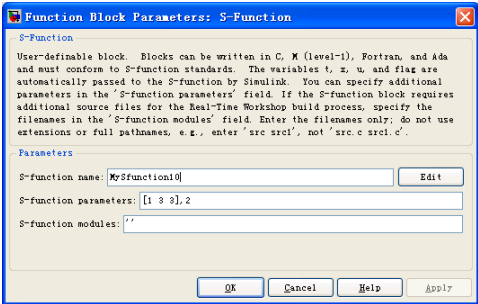


图 9-30 S-函数参数设置

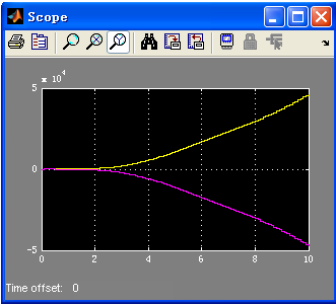


图 9-31 Simulink 仿真运行结果

若将函数声明和回调函数 mdlUpdate 分别做如下替换, 并保存为 MySfunction102.c, 编译和设置参数后运行结果如图 9-25 所示。

```
#define S_FUNCTION_NAME MySfunction102
static void mdlUpdate(SimStruct *S, int_T tid)
{
    real_T      *y      = ssGetOutputPortRealSignal(S,0);
    real_T      *x      = ssGetRealDiscStates (S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T *p1= mxGetPr(ssGetSFcnParam(S,0));
    //X(k+1)=AX(k)+Bu(k)
    x[0]=-x[1];
    x[1]=-x[2];
    x[2]=p1[0]*x[0]+p1[1]*x[1]+ p1[2]*x[2]+u[0];
}
```

这里需要注意的是, 本例对于相同的方程, 却有两个不同的结果。S-函数 MySfunction10.c 和 MySfunction5.m 采用的是同步解算差分方程的方式, 即在计算新状态值时是同步更新的; 而 S-函数 MySfunction102.c 和 MySfunction8.m 采用的是异步解算差分方程的方式, 即在计算新状态值时按照指定次序更新, 且已更新状态的结果用于后面状态的更新; 对于异步方式, 通过改变更新次序, 也将获得不同的仿真结果。

对于连续系统而言, 不存在上述的差异。换句话说, 微分方程的解算都是采用同步方式。

9.6 使用 S-函数创建器编写 C MEX 文件型

前面介绍了如何使用各类模板生成 S-函数, MATLAB 还提供了 S-函数创建器 (S-functions Builder), 以便通过界面生成 C MEX 文件型 S-函数。

下面通过一个实例说明基于 S-函数创建器生成 C MEX 文件型 S-函数的方法。

【例 9-11】用 S-函数创建器生成 S-函数 (MySfunction11.c) 描述方程

$$\begin{cases} X(n+1) = AX(n) + Bu(n) \\ X(0) = 0 \\ Y(n) = CX(n) + Du(n) \end{cases} \quad \left(\text{其中 } A = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ p1 & p2 & p3 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, D = \begin{pmatrix} 1 \\ q \end{pmatrix}, \right.$$

$P = (p1 \ p2 \ p3)$)。具体步骤如下:

- (1) 创建如图 9-31 所示的 Simulink 仿真初始框图。
- (2) 双击“system”模块, 可以看到如图 9-32 所示的 S-函数设计界面。
- (3) 在如图 9-32 的 S-函数设计界面上填写如下内容:
 - 在“S-function name (文件名)”处填写“MySfunction11”。
 - 在“Initialization (初始化)”选项卡下“Number of discrete states (离散状态个数)”处填写“3”, “Discrete states IC (初始状态值)”处填写“[0 0 0]”, “Sample mode (采样时间类型)”处选择“Discrete”, “Sample time value (采样时间值)”处填写“0.1”, 目前不支持多采样时间和偏置量的设置。

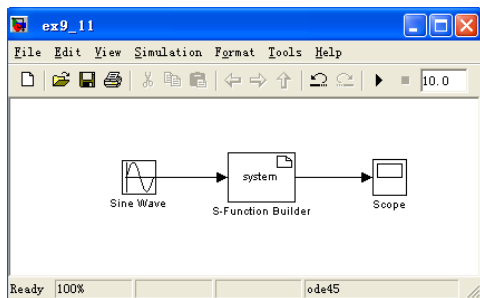


图 9-31 Simulink 仿真初始框图

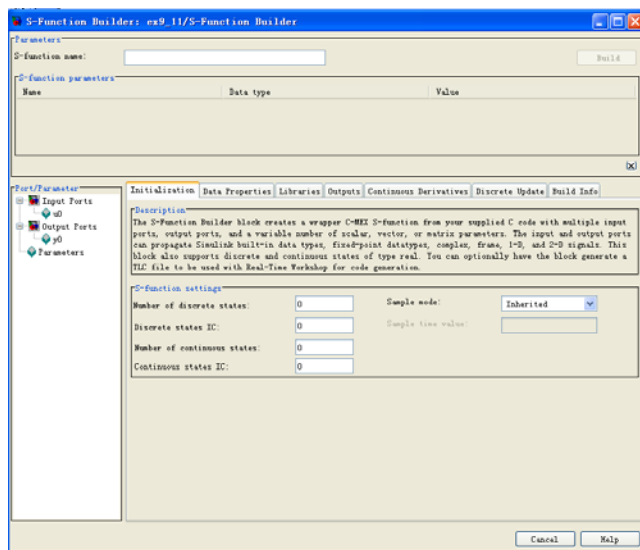




图 9-32 S-函数设计界面

- 在“Data Properties（数据属性）”选项卡的“Output ports”子选项卡下，选择 y0（端口 1）的“Rows（输出个数）”为 2。
- 在“Data Properties（数据属性）”选项卡的“Parameters”子选项卡下，先单击  图标（增加参数），再将新增参数的“Parameter name（参数名称）”设置为 p1。
- 在“Data Properties（数据属性）”选项卡的“Parameters”子选项卡下，先单击  图标（增加参数），再将新增参数的“Parameter name（参数名称）”设置为 p2。
- 在“Outputs（输出回调函数）”选项卡中填写如下内容：

```
y0[0]=xD[0]+u0[0];
y0[1]=xD[1]+p2[0]*u0[0];
```

其中，xD[0]表示第一个离散状态，u0[0]表示第一个输入端口的第一个输入，y0[0]表示第一个输出端口的第一个输出，p2[0]表示前面设置的参数 p2。同时，用 xC[0]表示第一个连续状态，dx[0]表示第一个连续状态的导数，若参数 P 是 $m \times n$ 的矩阵，则 $P(i,j)$ 应表示成 $P[i+(j-1)*m]$ ，即将数据按列拉直后排序。

- 在“S-function parameters（参数列表）”下参数 p1 的“Value（值）”处填写“[1 3 3]”，参数 p2 的“Value（值）”处填写“2”。
- 在“Discrete Update（离散状态更新回调函数）”选项卡中填写如下内容：

```
xD[0]=-xD[1];
xD[1]=-xD[2];
xD[2]=p1[0]*xD[0]+p1[1]*xD[1]+ p1[2]*xD[2]+u0[0];
```

（4）单击主界面上的“Build”按钮，则自动生成文件 MySfunction11.c、MySfunction11_wrapper.c、MySfunction11.tlc 和 MySfunction11.mexw32，且得到如图 9-33 所示的编译结果。

最后得到如图 9-34 所示的 Simulink 仿真框图，运行结果如图 9-35 所示。

通过 S-函数创建器生成 C MEX 文件型 S-函数比较方便，但功能受到一定的限制，读者可以根据实际需要选择生成的方法。

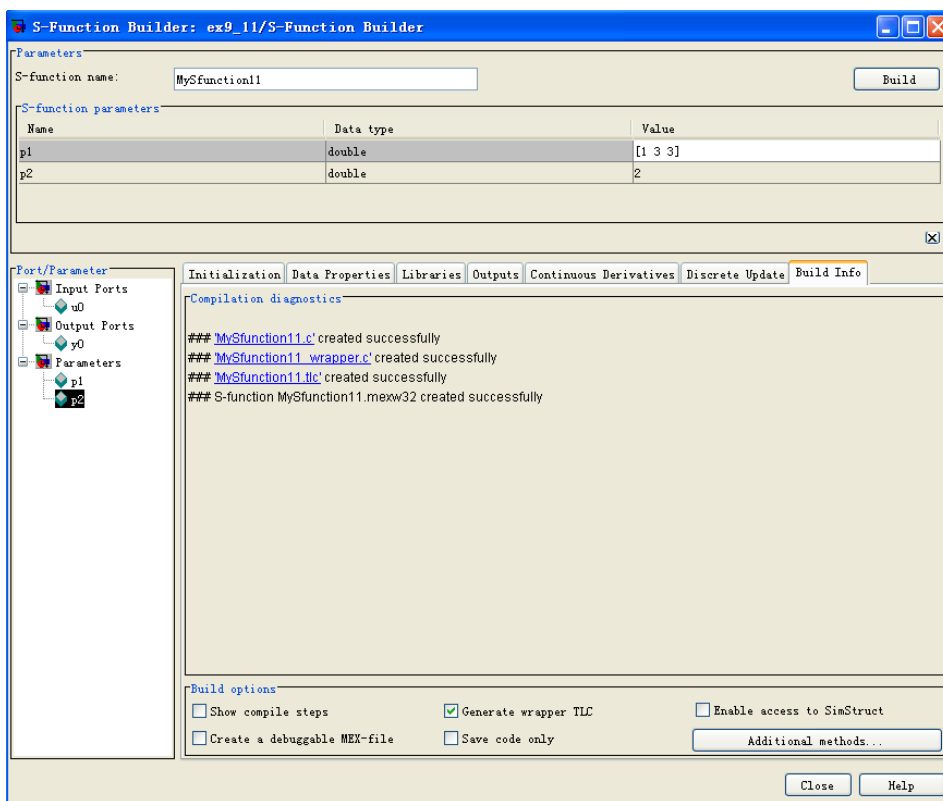


图 9-33 S-函数创建器编译结果

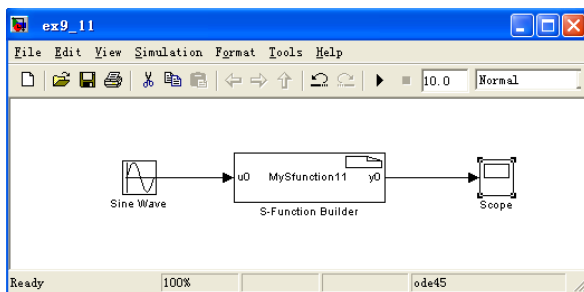


图 9-34 Simulink 仿真框图

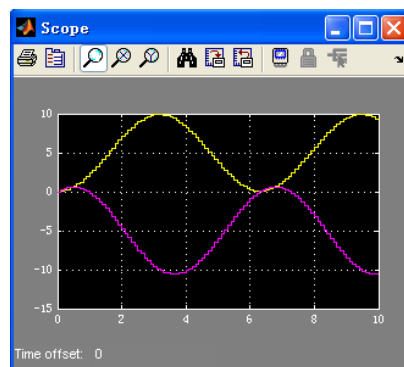


图 9-35 Simulink 运行结果

第 10 章 物理系统的建模和仿真

Simscape 是在 Simulink 基础上扩展的工具模块，用来搭建不同领域物理系统的模型，并进行仿真，例如由机械、传动、液压和电气元件组合而成的系统。Simscape 可以广泛应用于汽车业、航空业、国防和工业装备制造业。本章主要介绍如何利用 Simscape 进行不同类型（多学科）物理系统混合建模和仿真，从而实现搭建用户自定义的物理元件、库等。

在 Simscape 环境中，用户的建模过程如同装配真实的物理系统。Simscape 采用物理拓扑网络方式构建模型：每一个建模模块都对应一个实际的物理元器件，例如油泵、马达或者运算放大器，这种建模方式能直观地表现出物理系统的组成结构，而不是用晦涩的数学方程来表达；根据模型所表达的系统组成关系，自动构造出可以计算系统动态特性的数学方程，这些方程可同其他 Simulink 模型一起结合运算；在统一环境中实现多种类型物理系统建模和仿真，包括机械，电气和液压系统；使用基本物理建模单元构造模型，并提供了建模所需的模块库和相关简单数学运算单元。

10.1 物理元件库

Simscape 的建模库包含 6 个元件库，有基本元件库、传动系统元件库、电气系统元件库、液压系统元件库、机械系统元件库以及应用元件库。

其中提供了基本的传动建模单元、超过 24 个电气建模单元、15 个液压建模单元和 23 个机械建模单元，这些单元之间可以互相连接，联合建模。这些基本的单元也可以组合起来，构造更加复杂的器件模型。

Simscape 包含的元件库如图 10-1 所示。

基本元件库如图 10-2 所示。

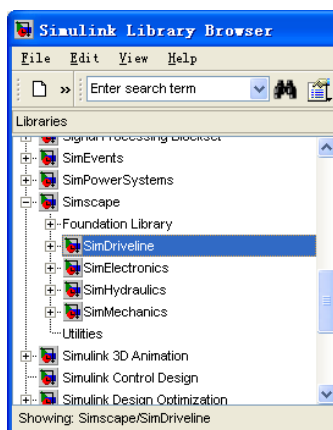


图 10-1 Simscape 下的元件库

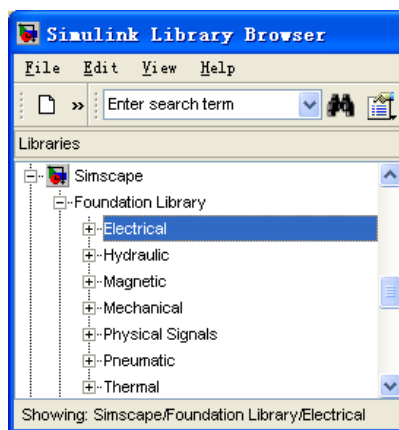


图 10-2 基本元件库

基本元件库主要包括：Electrical（有关电的）、Hydraulic（有关液压的）、Magnetic（有关

磁的)、Physical Signals (物理信号的)、Pneumatic (气动的) 以及 Thermal (有关热量的)。
Electrical 元件库如图 10-3 所示。

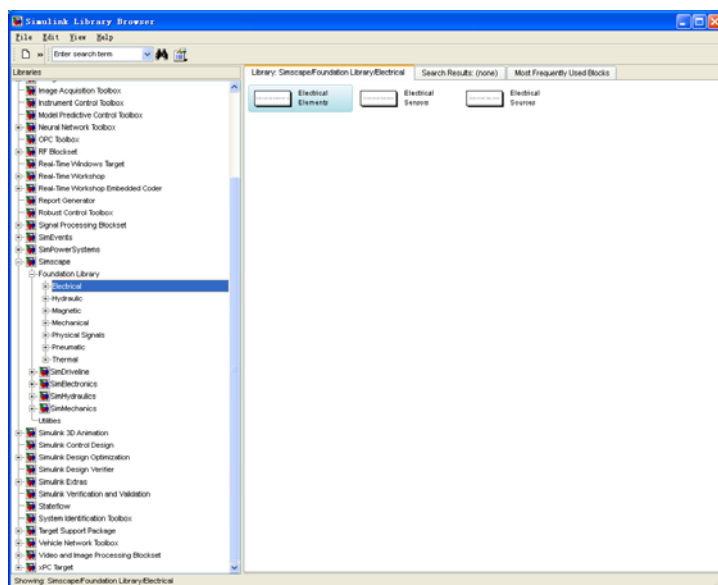


图 10-3 Electrical 元件库

单击鼠标右键，选中“Electrical Elements”，列出其中包含的元器件，如图 10-4 所示。

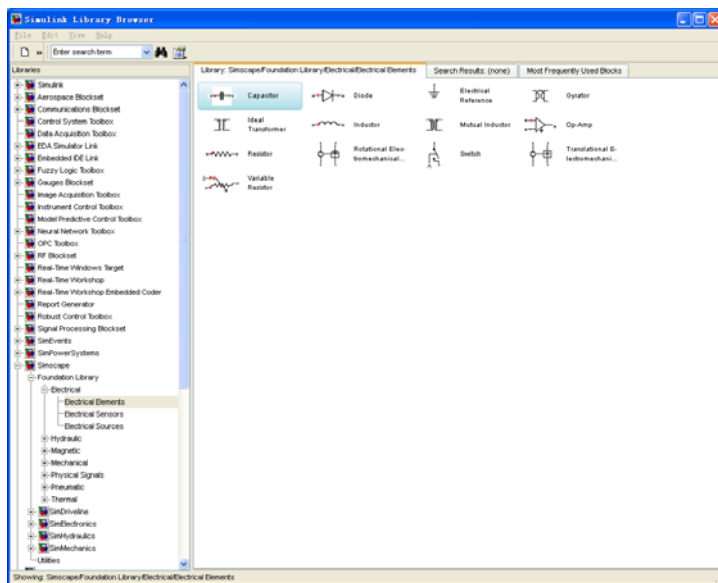


图 10-4 电器元件

Simscape 模型中的 Sensor 模块用来测量机械量（力/力矩，速度）、液压量（压力，流量）或电气量（电压，电流），信号量可以输出给标准的 Simulink 模块处理。而 Source 模块能够将标准的 Simulink 信号转换成同等量值的上述物理信号。Sensor 和 Source 模块的使用将 Simulink 控制算法模型同 Simscape 物理网络拓扑模型有机地结合起来，可实现闭环控制算法开发。图 10-5 为电气系统中的 Sensor 模块。

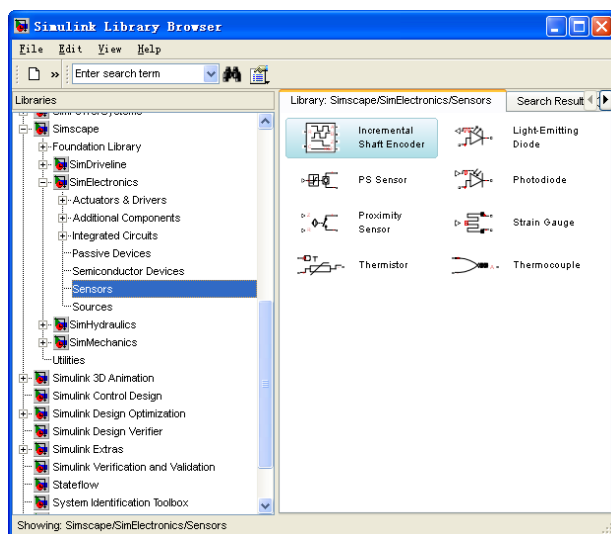


图 10-5 电气系统中的 Sensor 模块

10.2 机械系统

SimMechanics 集成于 Simulink 之中，是进行控制器和对象系统跨领域/学科的研究分析模块集。SimMechanics 为多体动力机械系统及其控制系统提供了直观、有效的建模分析手段，一切工作均在 Simulink 环境中完成。它提供了大量对应实际系统的元件，如刚体、铰链、约束、坐标系统、作动器和传感器等。使用这些模块可以方便地建立复杂图形化机械系统模型，进行机械系统的单独分析与任何 Simulink 设计的控制器及其他动态系统相连进行综合仿真。

SimMechanics 是 Simulink 物理建模产品家族的一员，该产品系列扩展了 Simulink 的建模能力，利用它们制作的模型仍能与传统 Simulink 模块所建立的模型相融合。

1. 特点

- 提供了三维刚体机械系统的建模环境。
- 包含了一系列分析机械运动和设计机械元件尺寸的仿真技术。
- 完整的建模层次，允许机械模型模块与其他类型模块结合使用。
- 可在 Simulink 中建立高精度、非线性的模型以支持控制系统的开发和测试。
- SolidWorks 转换器可以通过 CAD 工具定义机械模型。
- 包括各种铰链和约束形式。
- 可对平移运动和旋转运动，力和力矩进行建模、分析。
- 提供平衡点和线性化工具以支持控制系统设计。
- 使用 Virtual Reality Toolbox 或 MATLAB®图形（Handle Graphics®）支持机械系统可视化及动画显示。
- 可进行系统的运动学和正向、逆向动力学分析。
- 使用 O(n)递归求解多体动力学系统运动方程。
- 为模型定义提供多种本地坐标系统。

2. 功能

- 使用 Simulink 集成化的图形界面建立机械多体动力学系统的模型并进行仿真。

SimMechanics 使得用户可以方便地修改系统中的物理参数, 包括位置、方位角和机械元件运动参数等。

- 使用 Simulink 变步长积分法可以得到较高的计算精度。
- Simulink 的过零检测功能以双精度数据水平判定和求解不连续过程, 对于机械系统中存在的静摩擦和机械硬限位等情况建模具有重要的意义。
- SimMechanics 模型可与 Simulink 的控制系统模型方便地结合, 在同一个环境中对控制器和被控对象建模。

10.2.1 主要的机械元件

SimMechanics 系统包含的子系统:

- 使用 Simulink 查表模块和 SimMechanics 传感器和作动器定义的非线性的弹性单元。
- 用来定义航空器件压力分布的空气动力学拖曳模块, 例如副翼和方向舵。
- 主动车辆悬架系统, 例如防侧翻机械装置和控制器。
- 为飞行器和地面车辆设计的轮胎。

SimMechanics 系统包含如下模块:

- 具有质量的实体单元。
- 平移和旋转联接铰链单元。
- 向机械系统提供力和力矩作用的作动器单元, 可接受 Simulink 模型的信号。
- 测量机械系统运动物理量的传感器单元, 可向 Simulink 模型输出信号。

(1) 实体、铰链、约束和坐标系统

SimMechanics 支持任意数量的实体。实体通过质量属性、坐标系统定义, 以及铰链与其他实体相连。

可以在系统的运动实体上添加相应的运动约束。约束通过使用 Simulink 信号限定实体, 并以时间函数的形式驱动实体运动来实现。

SimMechanics 界面为定义坐标系统、约束和驱动条件以及力/力矩提供了多种方式, 包括:

- 在实体上连接多个本地坐标, 用于施加作用条件和测量物理量。
- 通过添加自己定制的模块来定制扩展铰链库。
- 在 SimMechanics 模块中使用 MATLAB 表达式和 Simulink 工具。

(2) 作动器和传感器

Simulink 和 SimMechanics 模块之间的联系通过作动器和传感器模块来完成。

作动器使用 Simulink 信号来指定实体或铰链上的力和运动, 包括:

- 指定实体或铰链的运动参数, 如按某种时间函数变化的位移、速度或加速度。
- 用 Simulink 信号 (包括系统中传感器的反馈信号) 指定力和力矩并施加在实体或铰链上。
- 检测由不连续摩擦力引起的离散事件。
- 计算系统的初始状态 (位移和速度), 用于动力学仿真。

传感器模块用来检测实体和铰链的运动参数, 并输出为 Simulink 信号。包括:

- 在 Simulink 示波器模块中显示系统的位移、速度和加速度。
- 监视系统中的作用力。

(3) 机械运动的仿真和分析

SimMechanics 为机械系统提供了如下仿真/分析方式:

- 正向动力学分析——求解机械系统在给定激励下的响应。
- 逆向动力学分析——求解机械系统在给定运动结果时所需的力和力矩。
- 运动学分析——在约束条件下求解系统中的位移、速度和加速度，并进行一致性检查。
- 线性化分析——可求得系统在指定小扰动或初始状态下的线性化模型，以分析系统响应性能。
- 平衡点分析——可以确定稳态平衡点，供系统分析和线性化使用。

10.2.2 建模的基本要点及步骤

1. 建模的基本要点

- 每一个 Simscape 网络都必须包含一个 Solver Configuration 模块。
- 如果模型中有液压元件，则每一个液压回路都必须包含一个 Custom Hydraulic Fluid 模块或者 Hydraulic Fluid 模块。
- 为了与通用 Simulink 模块相连，比如 Sources 和 Scopes 模块，使用 Connector 模块（Sensor、Converter 等）。
- 采取逐步建模的方法。从一个简单模型开始，运行并调试好后，再添加其余功能。比如，可以先使用 Foundation 库里的 Resistive Tube 模块来建模，它只计算摩擦损失。在随后的系统设计中，可能要考虑到流体的可压缩性，这时就利用 Hydraulic Pipeline 模块来取代它，或者是使用 Segmented Pipeline 模块来考虑流体惯性力，取决于具体的应用场合。这些不同的数学模型，它们的元件设置界面（端口的数目和类型以及相应的通量和跨量）可能还是一样的，这表示可以在不改变模型网络的情况下，使用不同的模块来适应不同的精度要求。
- Simscape 模块有两种接口：Conserving Ports■（小方块）和 Physical Signal Inports and Outports△（小三角）。
- Simscape 里有不同类型的 Physical Conserving Ports，比如液压的、电的、热的、机械平移和机械转动的。每一种类型都有与之相关的 Through 和 Across 变量。
- 只有同种类型的端口才能相连。端口间的物理连线都是双向的，传送物理变量（如之前所说的 Through 和 Across 变量）。物理端口不能直接与 Simulink 端口相连。两种直接相连的 Conserving Ports 端口的所有 Across 变量必须相同（比如电压或者角速度）。物理连接线可以分支。与别的组件直接相连的组件拥有相同的 Across 变量。任何在物理连接链上传递的 Through 变量（比如电流或者扭矩）分布在所有相连的组件上，至于分布的形式取决于系统动力学。对于所有的 Through 变量，所有传入某个子域的量等于所有传出的量（类似于电学的节点电流）。
- Physical Signal Ports 之间可以互连，就像一般的 Simulink 信号那样连接。Physical Signal Ports 与 Simulink 端口之间的连接，通过转换模块 Simulink-PS (PS-Simulink) Converter Block。

2. 建模的基本步骤

下面就以如图 10-6 所示物理模型为例，介绍建模的基本步骤。

（1）从库里找到相应的模块，连接基本物理网络，如图 10-7 所示。

（2）加入外力，各个原件在 Foundation 下的 Sensors and Sources 库以及 Simscape 下的 Utilities 库。各个模块的功能基本上从其名字可以看出。建好的模型如图 10-8 所示。

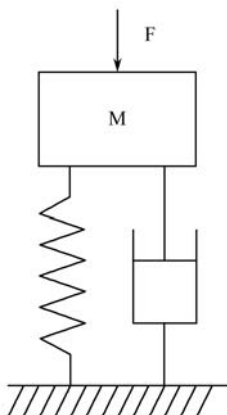


图 10-6 简单的物理模型

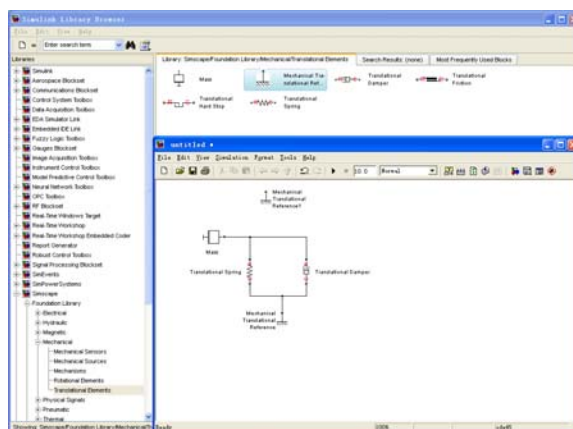


图 10-7 搭接的物理网络图

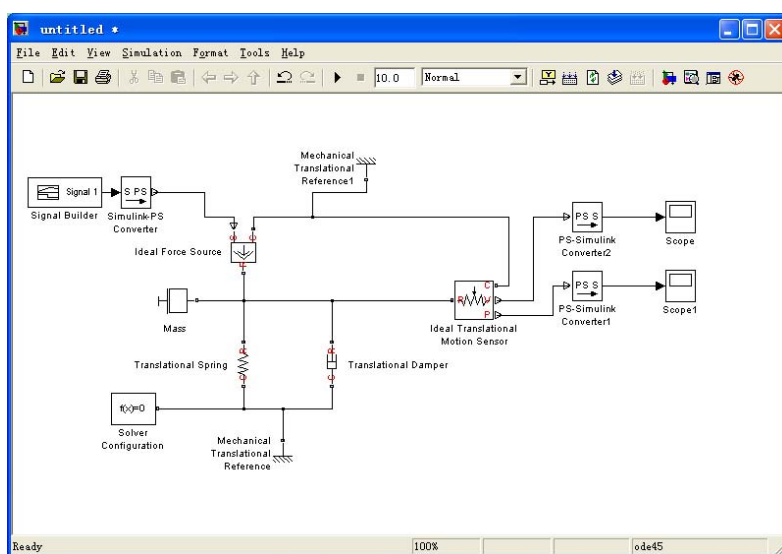


图 10-8 建好的物理模型

(3) 配置求解环境。选择“Simulation”→“Configuration Parameters”菜单命令，出现如图 10-9 所示界面。

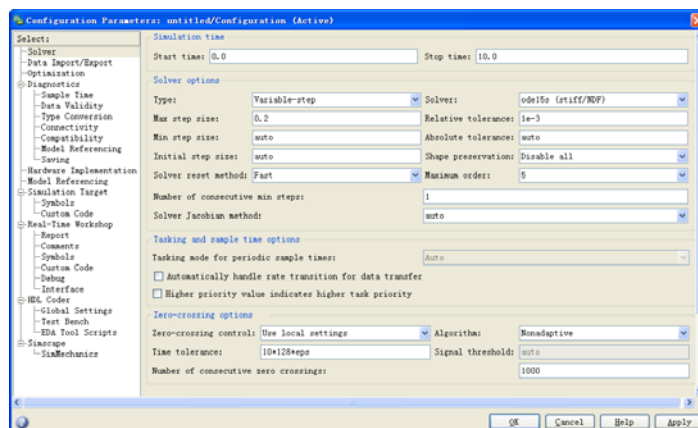


图 10-9 参数配置界面

通过配置参数,可以设置仿真的起始时间为 0.0,结束时间为 10.0,求解类型为 Variable-step (即变步长),求解方法为 ode15s (即 NDF 算法)等一系列相关的参数。

(4) 设置好参数之后,进行仿真。双击信号构造模块,设置输入信号,并单击“仿真”按钮运行仿真。分别出现如图 10-10 和图 10-11 所示界面。

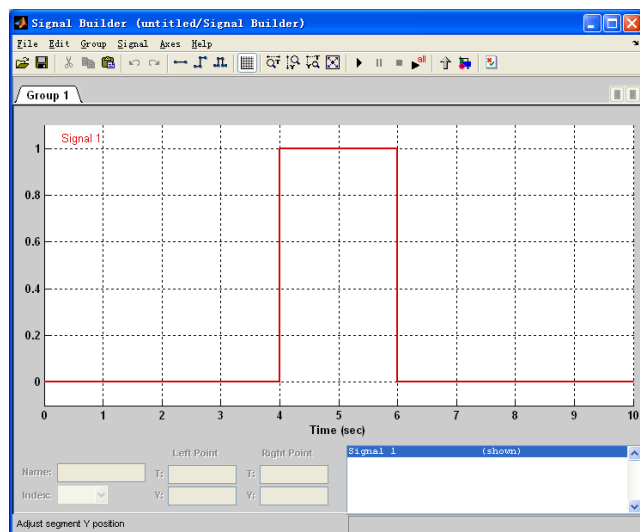


图 10-10 输入信号设置

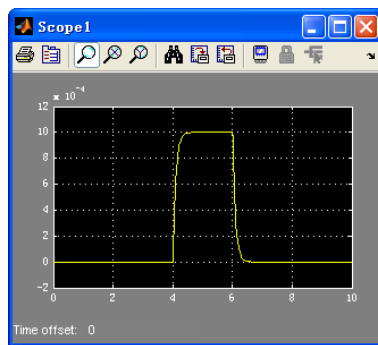
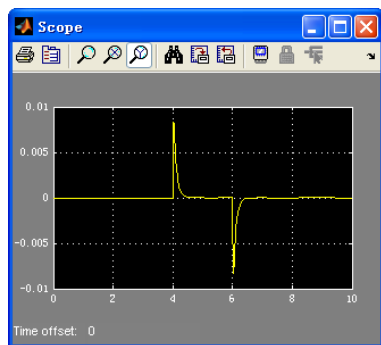


图 10-11 仿真结果

(5) 最后,还可以调整参数:

- 改变力,修改力信号模块。
- 改变模型参数,比如质量和刚度,双击质量体和弹簧元件。
- 改变质量体位置输出单位,双击 Converter 模块。

10.2.3 常用的机械系统

下面为帮助中典型机械系统的 Simscape 仿真,如图 10-12 所示。

此模型是由旋转和平移两个机械块组成的。

可以设置各个元器件的参数,下面仅做简要介绍。

轮轴 (Wheel and Axle) 配置参数如图 10-13 所示。

通过此设置,可以改变轮轴的半径等。

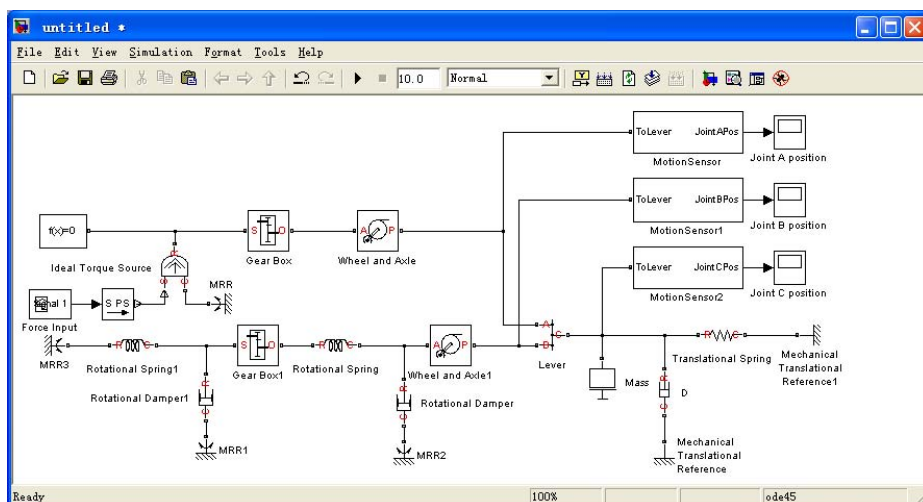


图 10-12 简单的机械系统模型

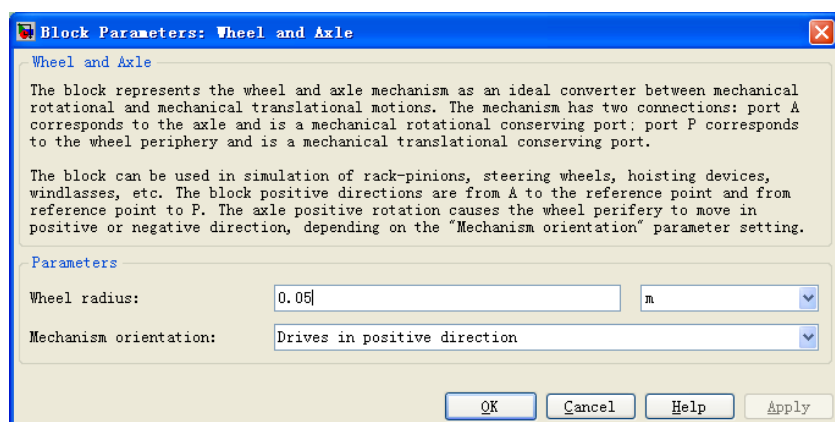


图 10-13 轮轴参数设置

对于弹簧，可以设置弹簧弹力系数，如图 10-14 所示。

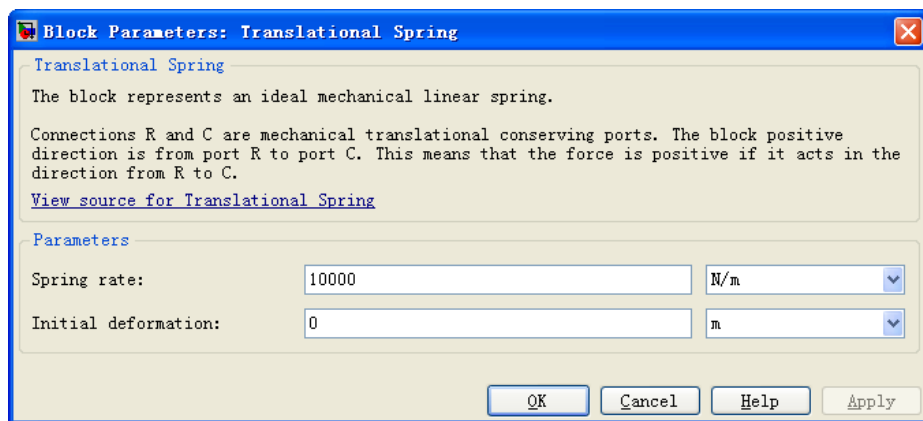


图 10-14 弹力系数设置

依次对各个模块设置相应的参数值，这里不再赘述。

图 10-15 所示为系统的仿真图。

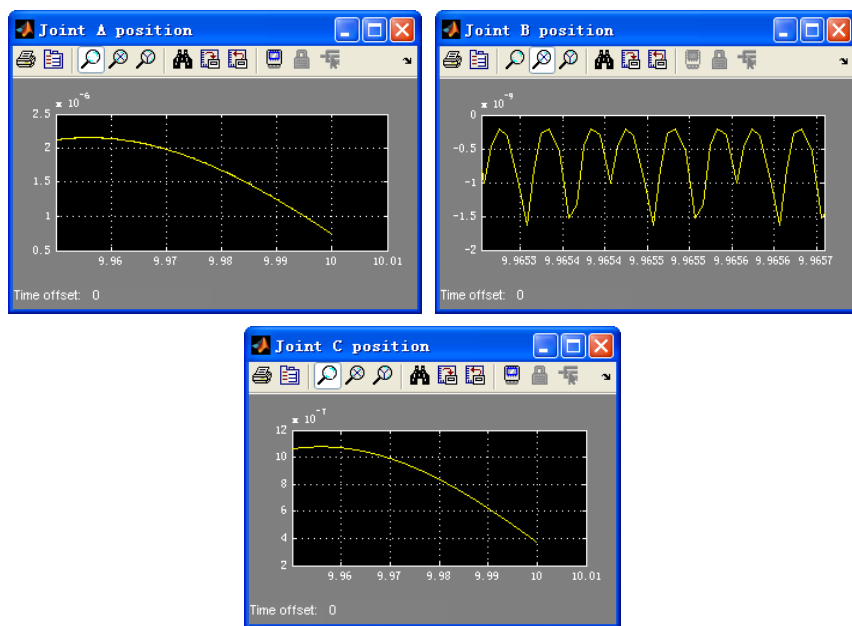


图 10-15 系统仿真图

10.3 电气系统

SimElectronics 为电子和机电系统的建模与仿真提供工具扩展了 Simscape 的功能。SimElectronics 使得电子和机电系统部件如物理网络一样进行多领域系统建模成为可能，它提供了半导体、电机、驱动、传感器和作动器部件，以及可定制的子系统模块。

10.3.1 主要的电气元件

SimElectronics 拥有多个模块，分别位于各个子模块库中。用户可以同标准的 Simulink 模块一起使用建立包含电气系统和控制回路的模型。

SimElectronics 模块库包含下列子模块库：Actuators & Drivers——执行机构及传动器，Additional components 额外元件；Integrated Circuits——集成电路；Passive Devices——被动元件；Semiconductor Devices——半导体器件；Sensors——传感器；Sources——电源。

下面只针对“执行机构及传动器”子模块库做简要介绍。

Actuators & Drivers 下包含三项，分别为：Drivers（传动器）、Rotational Actuators（转动传动器）及 Translation Actuators（平移传动器）。图 10-16~图 10-18 分别列出了各项中的元件。

(1) 传动器

“传动器”项中包含的元件有：受控脉宽调制电压、半桥、步进电机驱动。

(2) 转动传动器

“转动传动器”项中包含的元件有：直流电机、并励电机、感应电机、伺服电机、步进电机、交直流两用电机等。

(3) 平移传动器

“平移传动器”项中包含的元件有：通用的线性驱动器、piezo 线性感应电动机、piezo 堆、

螺旋管等。

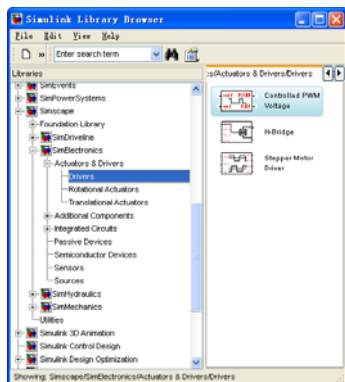


图 10-16 传动器

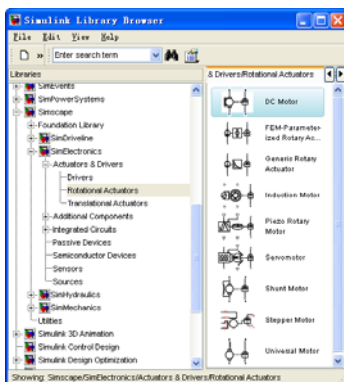


图 10-17 转动传动器

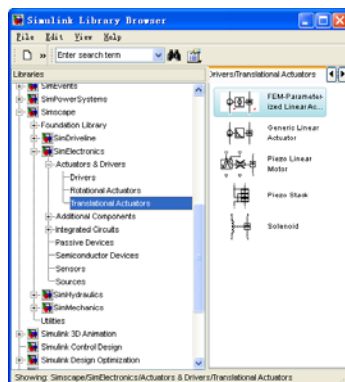


图 10-18 平移传动器

10.3.2 建模的基本步骤

下面就如图 10-19 所示电路模型，介绍建模的基本步骤。

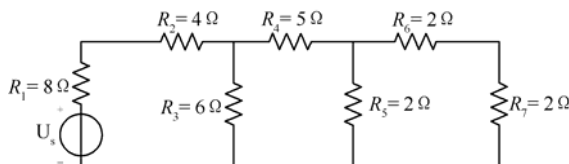


图 10-19 简单的电路模型

(1) 从库中找到相应的模块，连接基本电器件网络，如图 10-20 所示。

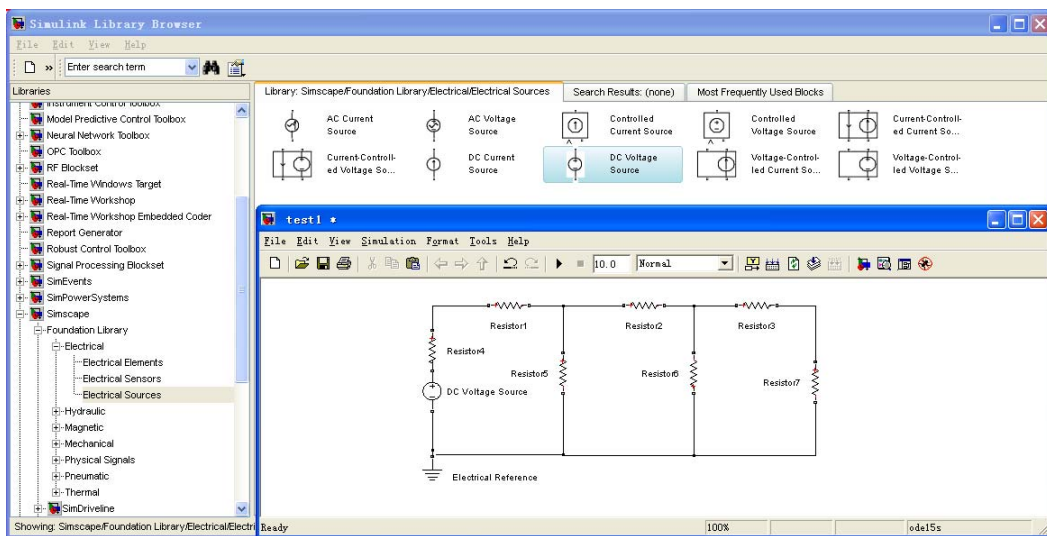


图 10-20 搭接的电气网络图

(2) 完善电气图，建好的模型如图 10-21 所示。

(3) 配置求解环境。选择“Simulation”→“Configuration Parameters”菜单命令，出现如图 10-22 所示界面。

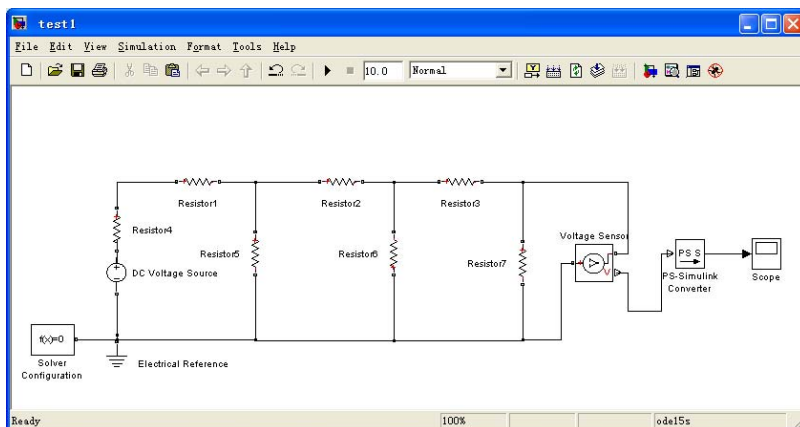


图 10-21 建好的电气模型

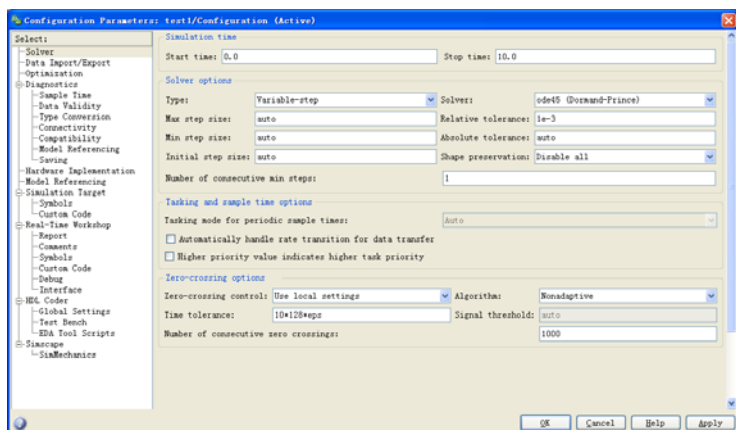


图 10-22 参数配置界面

可以设置仿真的起始时间为 0.0，结束时间为 10.0，求解类型为 Variable-step（即变步长），求解方法为 ode45（即龙格库塔法）一系列相关的参数。

对于电阻器件，双击即可设置大小，本例中设置为 2，如图 10-23 所示。

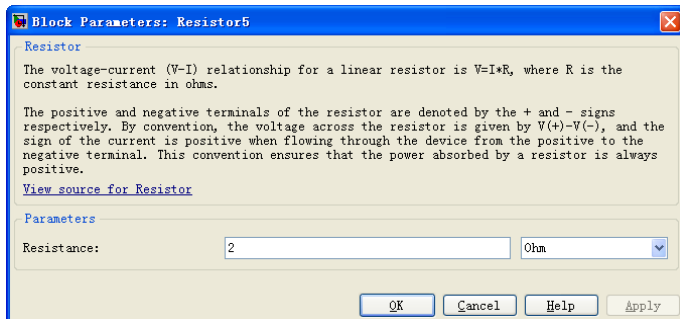


图 10-23 电阻值设置

对于 DC Voltage Source 电压源器件，设置电压值，本例中设置为 24V，如图 10-24 所示。

对于 Solver Configuration（解算器），可设置是否从稳定状态开始仿真以及一贯允许误差值等，本例中选用默认值，如图 10-25 所示。

对于转换模块 PS-Simulink，可以选择输出信号单位，本例中选用电压信号，如图 10-26

所示。

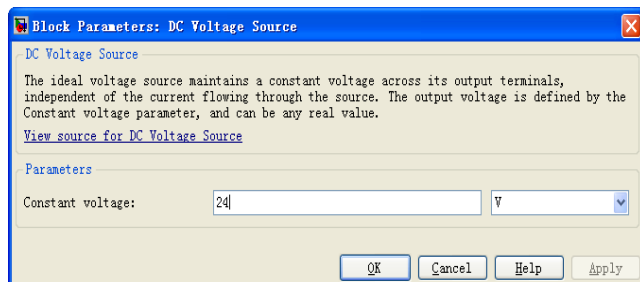


图 10-24 电压值设置

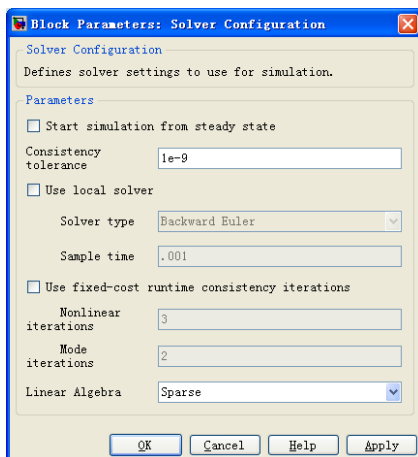


图 10-25 解算器设置

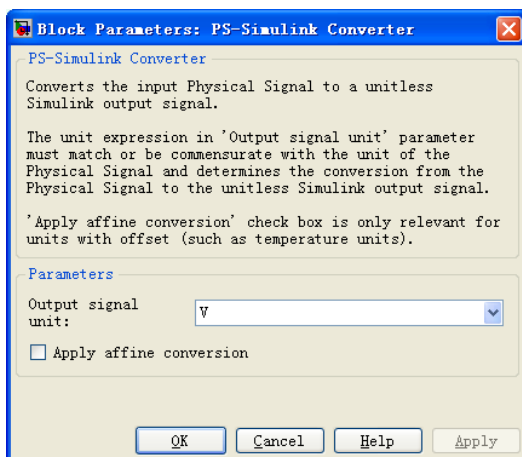


图 10-26 输出信号单位选择

(4) 设置好参数之后, 进行仿真, 并单击“仿真”按钮运行仿真。单击示波器出现图 10-27 所示界面。

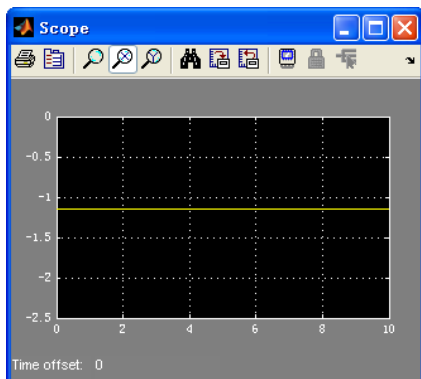


图 10-27 仿真结果

10.3.3 常用的电气系统

下面简要介绍两个常用电气系统的建模和仿真。

(1) 典型的积分运算电路(如图 10-28 所示)

搭接的 Simulink 图形如图 10-29 所示。

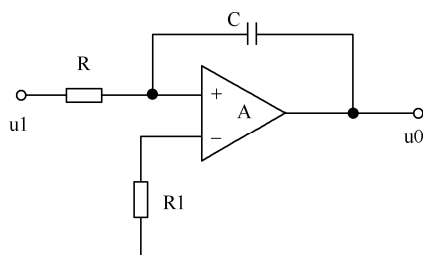


图 10-28 积分运算电路

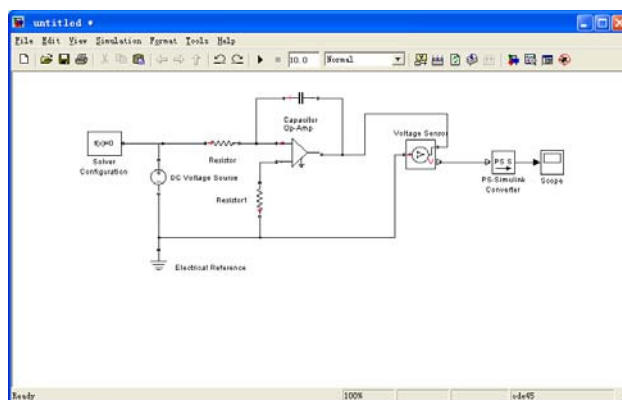


图 10-29 积分运算电路的 Simulink 图

参数设置：设置电源电压为直流 5V，电阻值设为 2V，则仿真图如 10-30 所示。

(2) 典型的微分运算电路 (如图 10-31 所示)

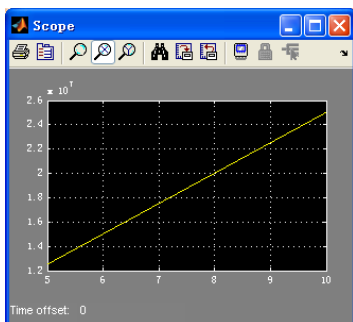


图 10-30 积分运算电路输出值

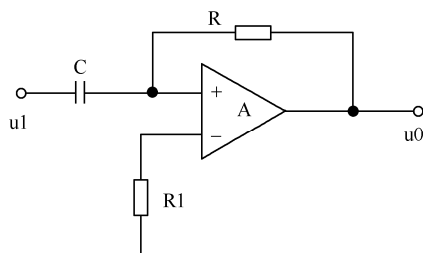


图 10-31 微分运算电路

搭接的 Simulink 图形如图 10-32 所示。

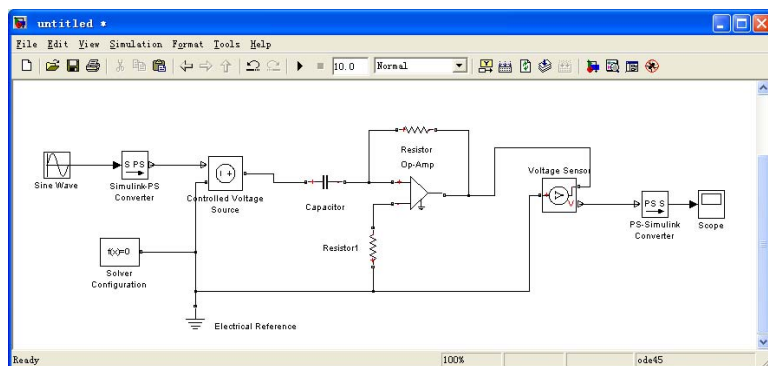


图 10-32 微分运算电路的 Simulink 图

参数设置：设置电源电压为标准正弦波，电阻值设为 2V，则仿真图如 11-33 所示。

(3) 两级阻容耦合放大电路 (如图 10-34 所示)

搭接的 Simulink 图形如图 10-35 所示。

参数设置：设置电源为 12V，输入电压为交流电压，电阻值设为 2V，则仿真图如 10-36 所示。

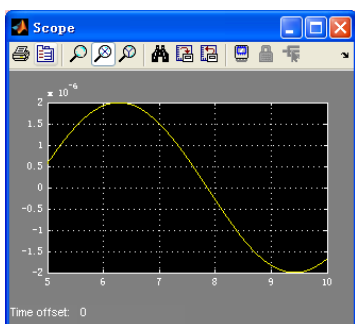


图 10-33 微分运算电路输出值

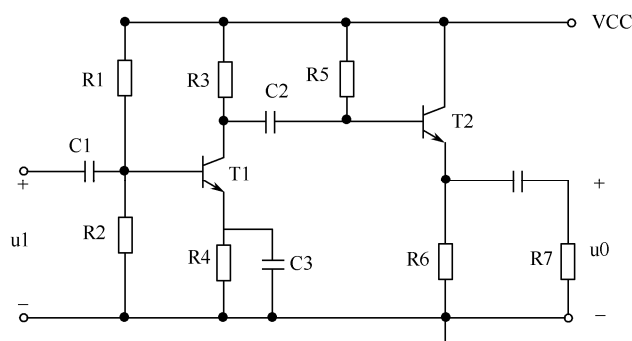


图 10-34 两级阻容耦合放大电路

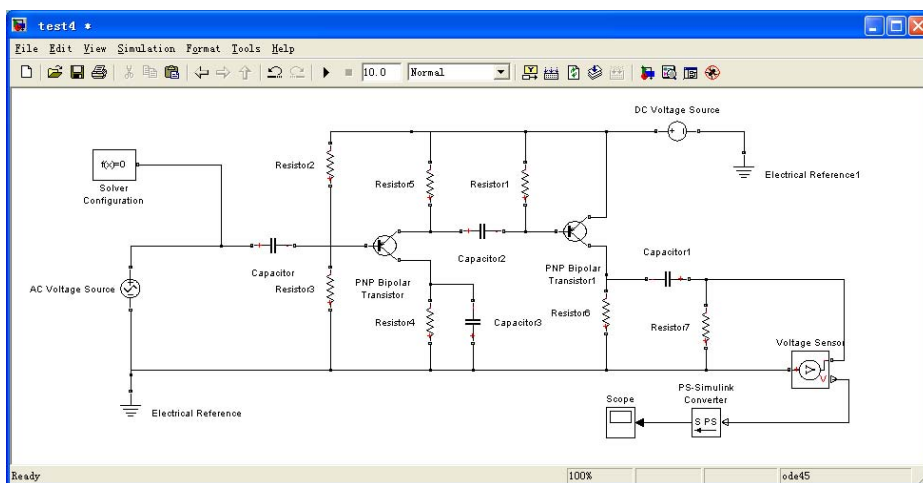


图 10-35 两级阻容耦合放大电路的 simulink 图

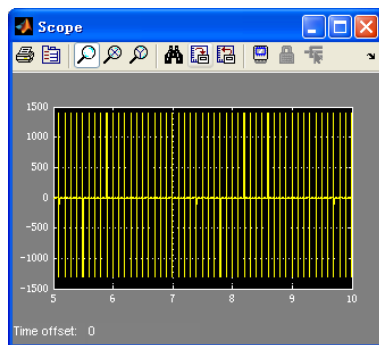


图 10-36 两级阻容耦合放大电路输出值

10.4 多域物理系统实例

除了上述的机械系统和电气系统之外, Simscape 中还具有其他领域物理系统建模与仿真的功能, 如机械传动系统建模和仿真工具以及液压系统建模和仿真工具等, 下面仅对液压系统做简要介绍。

1. 液压系统

SimHydraulics 是液压传动和控制系统的建模和仿真工具, 扩展了 Simulink 的功能。使用

这个工具可以建立含有液压和机械元件的物理网络模型，用于跨专业领域系统的建模。

SimHydraulics 提供了构成液压系统的元器件模块库，库中包括了用于构造其他元件的基本元素模块。SimHydraulics 适用于汽车、航空、国防和工业装备等领域的各种应用，例如自动变速器、舵面操纵系统和重载驱动装置的建模分析。

SimHydraulics 同 SimMechanics、SimDriveline 和 SimPowerSystems 一同使用，能够支持对复杂机液系统和电液系统的建模，以分析它们相互交联的影响。

图 10-37 所示是 demo 中的一个示例。

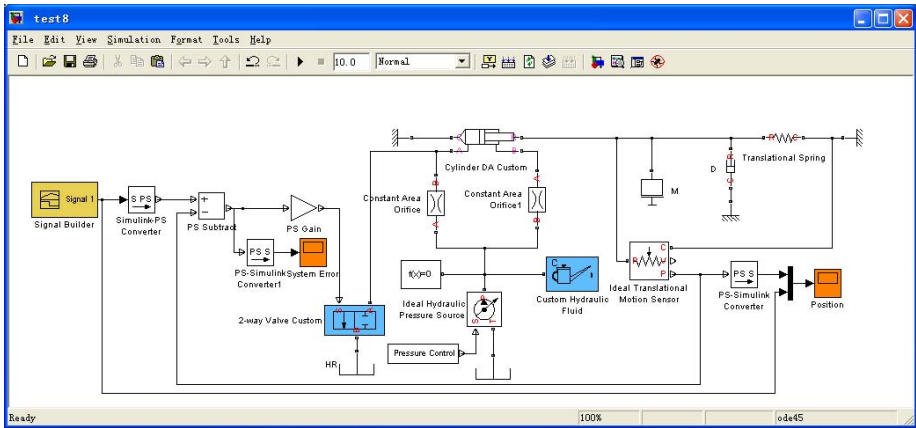


图 10-37 双向阀门在闭环控制回路中的应用

仿真截止时间设置为 0.5S，则仿真结果如图 10-38 所示。

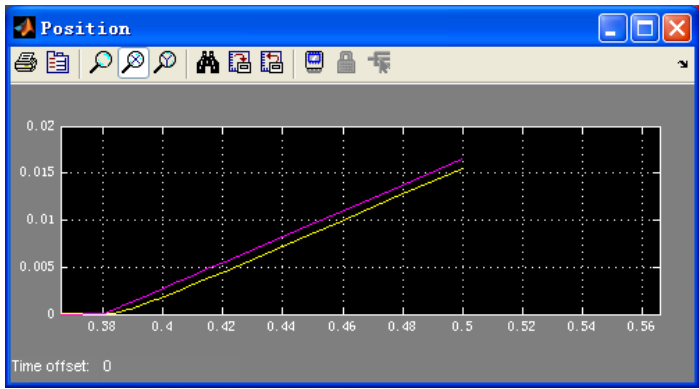


图 10-38 仿真结果

SimHydraulics 可在 Simulink 下建立液压系统回路的网络模型，并且建立的模型可以同机械和控制器模型相结合。

2. 电气系统与机械系统相结合

下面就以直流电机的例子来介绍一下电气系统和机械系统的简单搭接，如图 10-39 所示。

设定参数进行仿真：设定直流电压为 12V，电阻为 2Ω。最终转速值如图 10-40 所示。

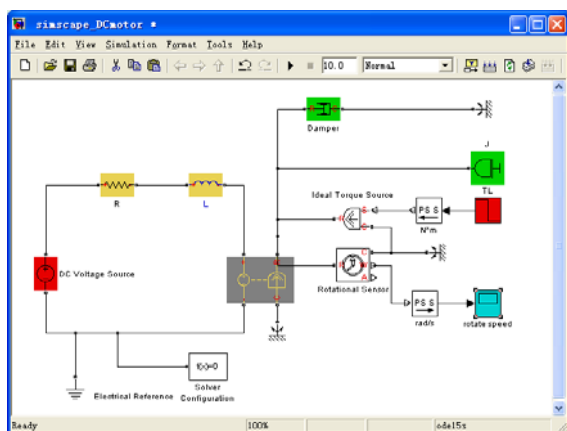


图 10-39 直流电机 Simscape 图

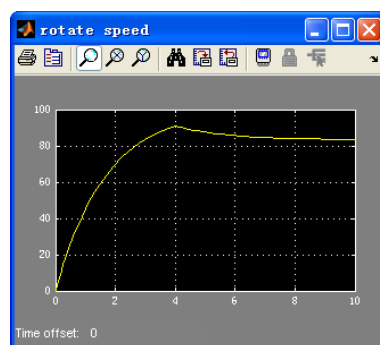


图 10-40 直流电机转速值

第 11 章 MATLAB 外部接口

MATLAB 提供与外部交互的强大功能，MATLAB 的外部接口使得 MATLAB 可以与外部设备和程序实现数据交互和程序移植，可以扩充 MATLAB 强大的数值计算和图形显示功能，从而弥补其执行效率较低的缺点，同时增强其他应用程序进行软件开发的功能，改善了软件开发效率。通过 MATLAB 接口编程，可以充分利用现有资源，能更容易地编写出功能强大、结构简洁的应用程序。

MATLAB 具有直接对磁盘文件进行访问的功能，提供很多文件输入和输出的内建函数，它们可对二进制文件或 ASCII 文件进行方便的打开、关闭和存储等操作。

本章将简要介绍 MATLAB 对磁盘文件访问的功能和 MATLAB 平台与其他平台间的外部接口，包括通过编译器生成可独立执行的代码，在 Word 和 Excel 中使用 MATLAB 资源，在 C 语言中调用 MATLAB 资源，调用外部设备以及互联网资源等内容。

11.1 文本文件

MATLAB 提供的文件访问能力，可以实现数据的导入、导出，增强了其程序设计的灵活性。MATLAB 支持的文件类型包括 MATLAB 自带文件、扩展标记文件、科学数据文件、文本文件、音频文件、视频文件、表单文件和图片文件。其中，文本文件是使用最广泛、最便于阅读的文件类型，下面介绍有关文本文件的访问。

11.1.1 打开/关闭文件

1. 打开文件

根据操作系统的要求，在程序中要使用或者创建一个磁盘文件时，必须向操作系统发出打开文件的命令，使用完毕后，还必须关闭这个文件。

在 MATLAB 中，使用与 C 语言同名的 `fopen` 函数打开二进制形式的文件，其具体用法如下：

```
fid=fopen(filename, permission)
```

```
[fid,message]=fopen(filename, permission)
```

`filename` 参数用来表示要打开的文件名（包含后缀）。

`permission` 参数用来表示文件处理方式，其选项如下所示：

- 'r'：以只读文件方式处理。
- 'w'：以更新文件方式处理，如果文件名不存在，则生成新文件，如果文件名存在，则覆盖文件原有内容。
- 'a'：以修改文件方式处理，如果文件名不存在，则生成新文件，如果文件名存在，则在文件原有内容末尾增加新内容。
- 'r+'：以读写文件方式处理读写文件，但不生成文件。

- 'w+': 如果文件名不存在, 则生成新文件, 并可进行读写操作, 如果文件名存在, 则覆盖文件原有内容, 并可进行读写操作。
- 'a+': 如果文件名不存在, 则生成新文件, 并可进行读写操作, 如果文件名存在, 则在文件原有内容末尾增加新内容, 并可进行读写操作。
- 'W': 以更新文件方式处理时没有自动格式。
- 'A': 以修改文件方式处理时没有自动格式。

当文件以文本形式打开时, 需要在上述指定的 `permission` 字符(串)后加字符 `t`, 如 `rt`、`w+t` 等。

`fid` 参数是由操作系统设定的一个整数, 用来表示文件操作的状态及标识已打开的文件。如果返回值为 `-1`, 则表示 `fopen` 无法打开该文件, 如以只读的形式打开不存在的文件; 如果返回值为非负数, 则表示文件标识。

`message` 参数用来表示文件操作的相关信息。

【例 11-1】以只读方式依次打开 `tan` 函数、`atan` 函数、`sin` 函数、`cos` 函数以及不存在的 `sincos` 函数对应文件。

命令窗口中输入如下语句:

```
[fid1,message1]=fopen('tan.m','r')
[fid2,message2]=fopen('atan.m','r')
[fid3,message3]=fopen('sin.m','r')
[fid4,message4]=fopen('cos.m','r')
[fid5,message5]=fopen('sincos.m','r')
```

命令窗口中的输出结果如下所示:

```
fid1 =
     3

message1 =
''

fid2 =
     4

Message2 =
''

fid3 =
     5

Message3 =
''

fid4 =
```



```
6

Message4 =
    ''

fid5 =
    -1

Message5 =
    'No such file or directory'
```

需要说明的是，前几条语句为已存在的文件分别给出文件标识 3、4、5 和 6，这四个数字仅仅是一个标识，不同情况下运行可能数值不同。

为了后续操作的顺利进行，程序设计中每次打开文件，都要进行该操作是否正确的判断，具体语句如下所示：

```
[fid,message]=fopen(filename,'r');
if fid==-1
    disp(message);
end
```

【例 11-2】如果某文件不存在，用函数 `fopen` 按只读方式打开文件。

在命令窗口中输入如下语句：

```
[fid,message]=fopen('x1.m','r')
```

命令窗口中的输出结果如下所示：

```
fid =
    -1

message =
    'No such file or directory'
```

2. 关闭文件

在打开文件后，如果完成了对应的读写工作，应该关闭文件。这样做一是为了提高程序的可靠性，二是可以避免系统资源的浪费。在 MATLAB 中，使用与 C 语言同名的 `fclose` 函数关闭文件，其具体用法如下：

```
status=fclose(fid)
```

其中，`fid` 参数即为要关闭文件的文件标识，它也是打开该文件时的返回值，如果关闭成功 `status` 返回值为 0，否则返回值为 -1。

【例 11-3】关闭已打开的文件。

命令窗口中输入如下语句：

```
fid=fopen('cos.m','r')
status=fclose(fid)
```

命令窗口中的输出结果如下所示：

```
fid =
    11

status =
```

0

如果需要一次关闭目前所有打开的文件，则可以执行如下语句：

```
status=fclose('all')
```

11.1.2 二进制形式访问

对于 MATLAB 而言，二进制文件是相对比较容易处理的，这些文件比较容易和 MATLAB 进行交互。常见的二进制文件包括.m、.dat、.txt 等。

本小节将着重介绍以二进制的形式访问文本文件。

1. 读取文件

在 MATLAB 中使用 fread 函数实现从文件中读取二进制数据，并将文本内容看成一个整数序列，结果最后以矩阵的形式返回，其具体使用方法如下：

```
a=fread(fid)
a=fread(fid,size)
a=fread(fid,size,precision)
```

fid 参数是打开文件时得到的文件标识。

size 参数表示读取整数的个数，它可能的形式如下所示：

- n: 读取后 n 个整数，并写入一个列向量中。
- inf: 读取至文件末尾的整数。
- [m,n]: 读取后 $m \times n$ 个整数，按列排序存放到 $m \times n$ 的矩阵中。

precision 参数表示读取的数据类型，默认情况是 uchar（即 8 位字符型）。常用的数据类型如表 11-1 所示，其中还给出了与 C 和 Fortran 语言的对比。

表 11-1 数据类型

MATLAB	C 或 Fortran	描述
'uchar'	'unsigned char'	无符号字符型
'schar'	'signed char'	带符号字符型（8 位）
'int8'	'integer*1'	整型（8 位）
'int16'	'integer*2'	整型（16 位）
'int32'	'integer*4'	整型（32 位）
'int64'	'integer*8'	整型（64 位）
'uint8'	'integer*1'	无符号整型（8 位）
'uint16'	'integer*2'	无符号整型（16 位）
'uint32'	'integer*4'	无符号整型（32 位）
'uint64'	'integer*8'	无符号整型（64 位）
'single'	'real*4'	浮点数（32 位）
'float32'	'real*4'	浮点数（32 位）
'double'	'real*8'	浮点数（64 位）
'float64'	'real*8'	浮点数（64 位）

还有一些类型是与平台有关的，平台不同可能位数不同，如表 11-2 所示。

【例 11-4】以二进制的方式读取文件 ex12_4.m，其内容如下所示：

```
function y=ex12_4(n)
y=rand(n);
```

表 11-2 与平台有关的数据类型

MATLAB	C 或 Fortran	描述
'char'	'char*1'	字符型
'short'	'short'	整型（16 位）
'int'	'int'	整型（32 位）
'long'	'long'	整型（32 位或 64 位）
'ushort'	'unsigned short'	无符号整型（16 位）
'uint'	'unsigned int'	无符号整型（32 位）
'ulong'	'unsigned long'	无符号整型（32 位或 64 位）
'float'	'float'	浮点数（32 位）

在命令窗口中输入如下语句：

```
fclose('all');
clear
clc
fid=fopen('ex12_4.m', 'r');
a=fread(fid);
a1=a'
a=fread(fid,8);
a2=a'
fclose(fid);
```

命令窗口中的输出结果如下所示：

```
a1 =
Columns 1 through 18
102    117    110    99    116    105    111    110    32    121    61    101    120    49
50    48    51    40
Columns 19 through 36
110    41    13    10    121    61    114    97    110    100    40    110    41    59
13    10    13    10
Columns 37 through 38
13    10

a2 =

[]
```

从本例可以看出，a2 并没有得到预期结果，而是得到空矩阵。这是因为读语句执行后会影响文件内的控制位置，详细内容见 12.1.4 小节。语句“a=fread(fid)”执行后，文件内的控制位置位于文件末尾，所以无法向下读取 8 个字节。每次文件打开，都将文件内的控制位置置于开始处。

【例 11-5】以二进制的方式读取上述文件 ex12_4.m。

在命令窗口中输入如下语句：

```

fclose('all');
clear
clc
fid=fopen('ex12_4.m', 'r');
a1=fread(fid,[3,6])
a=fread(fid,inf);
a2=a'
fclose(fid);

```

命令窗口中的输出结果如下所示：

```

a1 =
    102    99   111   121   120    48
    117   116   110    61    49    51
    110   105    32   101    50    40

a2 =
Columns 1 through 18
    110    41    13    10   121    61   114    97   110   100    40   110    41    59
    13    10    13    10
Columns 19 through 20
    13    10

```

从本例也可以看出，执行语句“a=fread(fid,[3,6])”影响了文件内的控制位置，因此再执行语句“fread(fid,inf)”的结果维数不是例 11-4 中的 38，而是 20。

【例 11-6】以指定数据类型的二进制方式读取上述文件 ex12_4.m。

在命令窗口中输入如下语句：

```

fclose('all');
clear
clc
fid=fopen('ex12_4.m', 'r');
a=fread(fid,inf, 'int8') ;
a1=a'
fclose(fid);
fid=fopen('ex12_4.m', 'r');
a=fread(fid,inf, 'int16') ;
a2=a'
fclose(fid);

```

命令窗口中的输出结果如下所示：

```

a1 =
Columns 1 through 18
    102   117   110    99   116   105   111   110    32   121    61   101   120    49
    50    48    51    40

```

Columns 19 through 36

```
110    41    13    10   121    61   114    97   110   100    40   110    41    59
    13    10    13    10
```

Columns 37 through 38

```
13    10
```

a2 =

Columns 1 through 9

```
30054    25454    26996    28271    31008    25917    12664
12338    10291
```

Columns 10 through 18

```
10606    2573    15737    24946    25710    28200    15145
2573    2573
```

Column 19

```
2573
```

从本例可以看出，采用不同的精度得到的结果不同。

2. 写入文件

在 MATLAB 中使用 `fwrite` 函数实现将二进制数据写入已打开的文件，其具体使用方法如下：

```
count=fwrite(fid,a,precision)
```

`fid` 参数是打开文件时得到的文件标识，`a` 参数是待写入的矩阵，`precision` 参数的含义同前，`count` 为成功写入的元素个数。

【例 11-7】将矩阵写入文件 `ex12_7.txt`。

在命令窗口中输入如下语句：

```
clear
clc
A=[1 2 3;4 5 6;7 8 9];
fid=fopen('ex12_7.txt','w');
count=fwrite(fid,A,'int32')
closestatus=fclose(fid)
```

运行结果如下：

```
count =
     9
closestatus =
     0
```

再输入如下代码：

```
clear
clc
fid=fopen('ex12_7.txt','r');
A=fread(fid,[3 3],'int32');
```

```

closestatus=fclose(fid);
B=magic(3);
C=A*B

```

运行结果如下：

```

C =
    26    38    26
    71    83    71
   116   128   116

```

需要说明的是，尽管 ex12_7.txt 文件的扩展名是.txt，但当打开该文件时无法看到数据。

11.1.3 普通形式访问

本小节将着重介绍以普通的形式访问文本文件。

1. 读取文件

在 MATLAB 中使用 fgetl 函数和 fgets 函数实现将文本文件中的某一行读出，并将该行的内容以字符串形式返回，区别在于 fgetl 函数会舍弃换行符，而 fgets 函数会保留换行符。它们的具体使用方法如下：

```

tline=fgetl(fid)
tline=fgets(fid)

```

【例 11-8】用函数 fgetl 实现命令 type 的功能。

具体代码序列如下：

```

fid=fopen('sinc.m');
while 1
    tline = fgetl(fid);
    if ~ischar(tline)
        break;
    else
        disp(tline)
    end
end
fclose(fid);

```

运行结果如下：

```

function y=sinc(x)
%SINC Sin(pi*x)/(pi*x) function.
%   SINC(X) returns a matrix whose elements are the sinc of the elements
%   of X, i.e.
%       y = sin(pi*x)/(pi*x)    if x ~= 0
%       = 1                    if x == 0
%   where x is an element of the input matrix and y is the resultant
%   output element.

```

```
%
% See also SQUARE, SIN, COS, CHIRP, DIRIC, GAUSPULS, PULSTRAN, RECTPULS,
% and TRIPULS.
% Author(s): T. Krauss, 1-14-93
% Copyright 1988-2002 The MathWorks, Inc.
% $Revision: 1.7 $ $Date: 2002/04/15 01:13:58 $
y=ones(size(x));
i=find(x);
y(i)=sin(pi*x(i))./(pi*x(i));
```

如果已知写入时的格式，可以按此格式完整地读出。在 MATLAB 中使用 `fscanf` 函数实现已知格式文件的读取，即当确定文件的 ASCII 码格式时，可用函数 `fscanf` 进行更精确的读取，其具体使用方法如下：

```
a=fscanf(fid,format)
a=fscanf(fid,format,size)
[a,count]=fscanf(fid,format,size)
```

`fid` 参数是打开文件时得到的文件标识。`size` 参数与 11.1.2 节中的类似，`a` 参数是读取数据构成的向量（如 `size` 参数形如 $m \times n$ 则返回矩阵，否则返回列向量），`count` 参数是读取数据的个数，`format` 参数用于指定读入数据的格式，常用的选项如下所示：

- **%s**：按字符串进行输入转换。
- **%d**：按十进制数据进行转换。
- **%f**：按浮点数进行转换。

另外还有其他的格式，它们与 C 语言中的 `fprintf` 中参数的用法是相同的。

在格式说明中，除了单个的空格字符可以匹配任意个数的空格字符外，通常字符在输入转换时将一一匹配，函数 `fscanf` 将输入的文件看做一个输入流，MATLAB 根据格式来匹配输入流，并将匹配后的数据读入到 MATLAB 中。

【例 11-9】读取文本文件（ex12_9.txt）中的数据，这些数据由 `y=rand(5,6)` 产生。

具体代码序列如下：

0.7577	0.7060	0.8235	0.4387	0.4898	0.2760
0.7431	0.0318	0.6948	0.3816	0.4456	0.6797
0.3922	0.2769	0.3171	0.7655	0.6463	0.6551
0.6555	0.0462	0.9502	0.7952	0.7094	0.1626
0.1712	0.0971	0.0344	0.1869	0.7547	0.1190

在命令窗口中输入如下语句：

```
clear
clc
fid=fopen('ex12_9.txt','r');
d1=fscanf(fid,'%s',[5 6])
fclose(fid);
fid=fopen('ex12_9.txt','r');
d2=fscanf(fid,'%f',[5 6])
```

```
fclose(fid);
fid=fopen('ex12_9.txt','r');
d=fscanf(fid,'%f');
d3=d'
fclose(fid);
```

命令窗口中的输出结果如下所示：

```
d1 =
04158.07955 0.06045 0.03185 0.03315 0.09724.
.03943 0.07542 0.08001 0.01586 0.05642 0.05979
4.03053 0.01751 0.04638 0.03591 0.00928 0.0331
97.07059 0.02498 0.00812 0.03942 0.07392 0.037
859.00152 0.03633 0.06455 0.01946 0.08059 0.02

d2 =
    0.4984    0.3517    0.1966    0.8143    0.1493    0.8909
    0.7513    0.9597    0.8308    0.2511    0.2435    0.2575
    0.9593    0.2551    0.3404    0.5853    0.6160    0.9293
    0.8407    0.5472    0.5060    0.5853    0.5497    0.4733
    0.3500    0.2543    0.1386    0.6991    0.2238    0.9172

d3 =
Columns 1 through 11
    0.4984    0.7513    0.9593    0.8407    0.3500    0.3517    0.9597    0.2551    0.5472
0.2543    0.1966
Columns 12 through 22
    0.8308    0.3404    0.5060    0.1386    0.8143    0.2511    0.5853    0.5853    0.6991
0.1493    0.2435
Columns 23 through 30
    0.6160    0.5497    0.2238    0.8909    0.2575    0.9293    0.4733    0.9172
```

由本例可以看出，按格式读取时，必须选择正确的格式（包括数据类型和维数）才能正确地复现数据。

2. 写入文件

在 MATLAB 中使用 `fprintf` 函数实现将数据按给定格式写入文件，其具体使用方法如下：

```
count=fprintf(fid,format,y)
```

`fid` 参数是打开文件时得到的文件标识，`y` 参数用于指定要写入的数据，`count` 参数用于返回成功写入的字节数，`format` 参数用于指定写入文件的数据格式，常用的格式如下所示：

- `%e`：科学计数形式，即数值表示成 $a \times 10^b$ 形式。
- `%f`：固定小数点位置的数据形式。
- `%g`：在上述两种格式中自动选取较短的格式。

此外，还可以包括数据占用的最小宽度和数据精度的说明。可以同时使用 `\n`、`\r`、`\t`、`\b`、

\f 等分别代表换行、回车、Tab、退格、走纸等字符，可以用\\代表反斜线\，可以用%%代表百分号%。

【例 11-10】向文件 ex12_10.dat 中写入数据。

在命令窗口中输入如下语句：

```
clear
clc
y=rand(5)
fid=fopen('ex12_10.dat','w');
fprintf(fid,'%6.3f',y);
fclose(fid);
clear
fid=fopen('ex12_10.dat','r');
ey=fscanf(fid,'%f');
ey1=ey'
fclose(fid);
fid=fopen('ex12_10.dat','r');
ey2=fscanf(fid,'%f',[4 4])
fclose(fid);
```

命令窗口中的输出结果如下所示：

```
y =
    0.1361    0.8530    0.0760    0.4173    0.4893
    0.8693    0.6221    0.2399    0.0497    0.3377
    0.5797    0.3510    0.1233    0.9027    0.9001
    0.5499    0.5132    0.1839    0.9448    0.3692
    0.1450    0.4018    0.2400    0.4909    0.1112

ey1 =
Columns 1 through 11
    0.1360    0.8690    0.5800    0.5500    0.1450    0.8530    0.6220    0.3510    0.5130
    0.4020    0.0760
Columns 12 through 22
    0.2400    0.1230    0.1840    0.2400    0.4170    0.0500    0.9030    0.9450    0.4910
    0.4890    0.3380
Columns 23 through 25
    0.9000    0.3690    0.1110

ey2 =
    0.1360    0.1450    0.5130    0.1230
    0.8690    0.8530    0.4020    0.1840
    0.5800    0.6220    0.0760    0.2400
```

0.5500 0.3510 0.2400 0.4170

本例需要说明的是，'%6.3f'表示占 6 个字符位，小数点后的精度是 3 位。MATLAB 的默认设置是小数点后 4 位（数值格式 short，数值显示 loose），所以 y 是小数点后 4 位；而指定写入文件是小数点后 3 位，所以写入时进行了四舍五入运算；读出时是写入内容的再现。

11.1.4 文件内的位置控制

打开文件读写数据时，需要判断和控制文件的读写位置，如数据是否读完，需要读写指定位置上的数据等。在读写文件时，MATLAB 会自动创建一个文件位置指针来管理和维护文件读写数据的起始位置。

读写数据时操作系统默认的方式是，从文件头开始顺序向后读写数据直至文件尾。操作系统通过文件指针来指示当前的文件位置，通过控制指针实现文件内的位置控制。在 MATLAB 中提供如表 11-3 所示的位置控制函数。

表 11-3 文件位置的控制函数

函数	功能
feof	判断指针是否在文件尾位置
fseek	设定文件指针位置
ftell	获取文件指针位置
frewind	设置指针至文件头位置

下面分别介绍表 11-3 中的函数。

feof 函数的具体用法如下：

status = feof(fid)

fid 参数是打开文件时得到的文件标识。status 参数为 1 表示指针在文件尾，否则为 0。

fseek 函数的具体用法如下：

status = fseek(fid, offset, origin)

fid 参数是打开文件时得到的文件标识。offset 参数是偏移量，以字节数为单位（正整数表示往文件尾方向移动指针，0 表示不移动指针，负整数表示往文件头方向移动指针）。origin 参数是基准点（'bof'和-1 表示文件头位置，'cof'和 0 表示目前位置，'eof'和 1 表示文件尾位置）。status 参数为 0 表示操作成功，否则为 1。

ftell 函数的具体用法如下：

position = ftell(fid)

fid 参数是打开文件时得到的文件标识。position 参数表示距离文件头位置的字节数，如果为-1 表示操作失败。

frewind 函数的具体用法如下：

frewind(fid)

fid 参数是打开文件时得到的文件标识。

【例 11-11】利用文件内的位置控制读取文件。

在命令窗口中输入如下语句：

clear

```
clc
fid=fopen('magic.m','r');
p1=ftell(fid)
a1=fread(fid,[4 4])
status=fseek(fid,10,'cof');
p2= ftell(fid)
a2= fread (fid,[4 4])
frewind(fid);
p3= ftell(fid)
a3= fread(fid,[4 4])
status=fseek(fid,0,'eof');
p4= ftell(fid)
d=feof(fid)
fclose(fid);
```

命令窗口中的输出结果如下所示:

```
p1 =
    0

a1 =
    102    116    32    32
    117    105    77    109
    110    111    32    97
    99    110    61    103

p2 =
    26

a2 =
    73    77    99    117
    67    97    32    97
    32   103   115   114
    32   105   113   101

p3 =
    0

a3 =
    102    116    32    32
    117    105    77    109
    110    111    32    97
```

```

    99    110    61    103

p4 =
    1043

d =
     0

```

由本例结果可以看出，文件头位置是 0，文件经过读取指针会相应地移动，文件指针受到指定函数的控制。

【例 11-12】通过演示数据文件指定位置数据读取的方法。

在命令窗口中输入如下语句：

```

A=magic(4);
fid=fopen('c:\data.txt','w');           %打开文件
fprintf(fid,'%d\n','int8',A);           %把 A 写入文件
fclose(fid);
fid=fopen('c:\data.txt','r');
frewind(fid);                           %把指针放在文件开头
if feof(fid)==0                          %如果没有到文件结尾，读取数据
    [B,COUNT]=fscanf(fid,'%d\n')         %把数据放入 B 中
    position=ftell(fid)                  %得到当前指针位置
end
if feof(fid)==1                          %如果指针已在文件结尾，重新设置指针
    status=fseek(fid,-4,'cof')           %设置指针从当前位置向文件头移动 4 个位置
    [C,COUNT1]=fscanf(fid,'%d\n')        %把读取到的数据放入 C 中
end
fclose(fid);                             %关闭数据

```

命令窗口中的输出结果如下所示：

```

B =
    105
    110
    116
    56
    16
     5
     9
     4
     2
    11
     7
    14

```

```
3
10
6
15
13
8
12
1
COUNT =
20
position =
54
status =
0
C =
2
1
COUNT1 =
2
```

11.2 MATLAB 与 Word 混合使用

MATLAB 提供了 Notebook 将 Microsoft Word 和 MATLAB 完美结合，即在编辑 Word 文档时利用 MATLAB 资源，包括文字处理、科学计算和绘图功能，这样的文件也称为 M-book 文档，它不仅拥有 Microsoft Word 的全部文字处理功能，而且具备 MATLAB 无与伦比的数学计算功能和灵活自如的计算结果可视化功能。它的工作原理是，首先在 Word 文档中创建命令，其次传递给 MATLAB 后台处理，最后将后台处理结果回传到 Word 中。

11.2.1 Notebook 的安装和使用

在命令窗口中输入如下语句：

```
notebook -setup
```

按回车键，命令窗口中的输出结果如下所示：

```
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
```

```
Setup complete
```

在命令窗口中输入如下语句：

```
notebook
```

执行后可以得到如图 11-1 所示的界面。

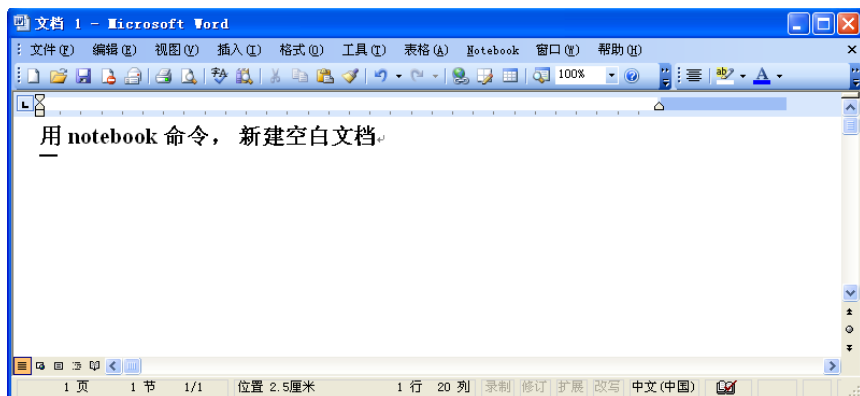


图 11-1 M-book 新文档

通过下面的方式可以打开一个已有的 M-book 文档：

```
notebook path\filename.doc
```

11.2.2 Notebook 的实际应用

在 M-book 文档中包括普通的文字，还包括需要和 MATLAB 交互的文字。Notebook 将与 MATLAB 交互文字的基本单位称做细胞，细胞可以是表达式，也可以是程序。Notebook 采用输入细胞（Input Cell）来定义 MATLAB 的代码，具体操作步骤如下：

- （1）采用文本格式输入代码，末尾不要加回车和空格。
- （2）通过 Notebook 菜单中的“Define Input Cell”选项定义输入细胞，其中输入细胞都显示为黑方括号包括绿色字符的形式。
- （3）通过 Notebook 菜单中的“Evaluate Cell”选项或者按“Ctrl+Enter”键，运行输入细胞内的代码，并得到黑方括号包括蓝色字符形式的输出细胞。如果出现错误，黑方括号内将包括红色字符。

Notebook 提供了选择细胞和运行细胞的功能，并将 MATLAB 运算结果以细胞保存。下面通过一个实例说明 Notebook 的应用。

【例 11-13】 Notebook 的简单应用，步骤如下：

- （1）新建一个 M-book。
- （2）输入“n=rand(4)”，选中 Notebook 菜单中的“Define Input Cell”选项将其定义为输入细胞，再按“Ctrl+Enter”键运行它，或者通过 Notebook 菜单中的“Evaluate Cell”选项运行，此时在 M-book 中看到如下内容：

```
n=rand(4)
n =
    0.7803    0.0965    0.5752    0.8212
    0.3897    0.1320    0.0598    0.0154
    0.2417    0.9421    0.2348    0.0430
    0.4039    0.9561    0.3532    0.1690
```

- （3）继续输入“m=sqrt(n)”，并定义为输入细胞且运行，可以在 M-book 中看到如下内容：

```
n=rand(4)
n =
```

0.7803	0.0965	0.5752	0.8212
0.3897	0.1320	0.0598	0.0154
0.2417	0.9421	0.2348	0.0430
0.4039	0.9561	0.3532	0.1690

```
m=sqrt(n)
```

```
m =
```

0.8833	0.3106	0.7584	0.9062
0.6243	0.3633	0.2445	0.1241
0.4916	0.9706	0.4845	0.2074
0.6355	0.9778	0.5943	0.4111

可以将 M-book 文档如同 Word 文件一样保存，以便今后查看和修改。

【例 11-14】利用 Notebook 绘图，具体步骤如下：

- (1) 新建一个 M-book，输入“绘图”并按回车键。
- (2) 再输入代码序列。
- (3) 然后输入“结束绘图”。
- (4) 保存文档。

按照以上的操作步骤，最后的运行结果如下：

绘图

```
[x,y]=meshgrid(-2:0.3:2);
```

```
z=peaks(x,y);
```

```
figure
```

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

```
mesh(x,y,z)
```

```
title('mesh')
```

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

```
meshc(x,y,z)
```

```
title('meshc')
```

```
subplot(2,2,3)
```

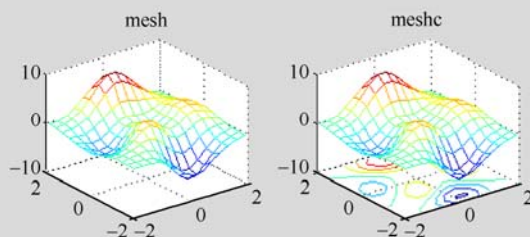
```
meshz(x,y,z)
```

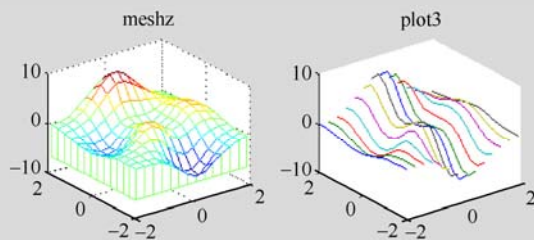
```
title('meshz')
```

```
subplot(2,2,4)
```

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

```
title('plot3')
```





结束绘图

在使用 Notebook 时需要注意以下问题:

- clear、save 等命令是可以执行的, 且含义同 MATLAB 中的一致, 但有些命令是不可以执行的, 如 clc 命令不会将前面的文字和细胞清空。
- M-book 文档细胞中的标点符号和 MATLAB 中的一样都必须在英文状态下输入。
- M-book 文档中细胞运行的速度比在 MATLAB 中要慢很多。
- 带鼠标操作交互的代码最好不在 M-book 文档中运行。
- Windows 是一种多任务操作系统。但在运行 M-book 文档时, 最好不运行其他程序, 不执行其他任务, 以免影响 M-book 文档中程序的正确执行。
- 当编辑科技论文或其他文档时, 最后可将细胞转换为普通文本。
- 可使用 Notebook 菜单中的“Bring MATLAB to Font”选项或者按组合键“Alt+M”把 MATLAB 的命令窗口调到前台。
- 可使用 Notebook 菜单中的“Toogle Graph Output for Cell”选项控制是否显示输入细胞或输出细胞的输出图形。

11.3 MATLAB 与 Excel 混合使用

本节主要介绍如何在 Excel 平台下使用 MATLAB 的资源。

MATLAB 作为功能强大的数学软件, 数据计算和图像显示是优势, 而微软的 Excel 同样具有较强的数据统计和显示能力。Excel 在一些较为简单的图像显示上方便易用, 对一些复杂的图像显示就差强人意了。MATLAB 提供了两种方法实现了与 Excel 的数据共享和功能交互, 使得这两大软件有机地结合起来。

MATLAB 提供了 Spreadsheet Link 将 Microsoft Excel 和 MATLAB 完美结合, 即在 Excel 表单中利用 MATLAB 资源, 包括科学计算和绘图功能。它的工作原理是, 首先在 Excel 表单中创建命令, 其次传递给 MATLAB 进行后台处理, 最后将后台处理结果回传到 Excel 表单中。

Excel Link 的运行机制如图 11-2 所示。

11.3.1 Spreadsheet Link 的安装

Excel Link 是一个插件软件, 在基于 Windows 的环境中集成了 Excel 和 MATLAB。Excel 和 MATLAB 的协作可以在 Excel 工作表和宏工具中使用 MATLAB 的数值分析、数学计算和绘图功能。Excel Link 保持了两两者之间数据的同步交换, 它的工作环境必须是微软的 Windows, 如

Windows XP 等,系统必须安装 Excel 和 MATLAB,而且安装 MATLAB 时选择安装 Excel Link。

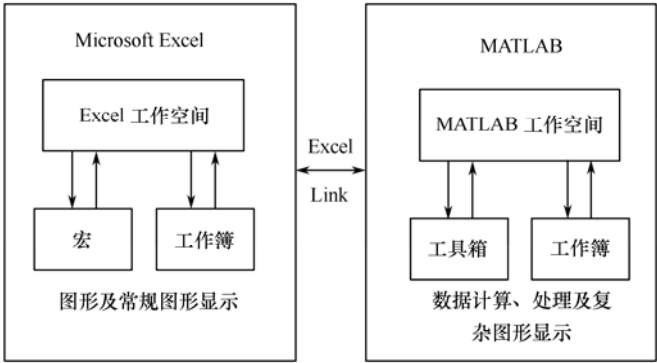


图 11-2 Excel Link 的运行机制

上述系统要求具备后,还需对 Excel Link 进行配置,以实现 MATLAB 与 Excel 的连接。在 MATLAB 中使用 Spreadsheet Link 的前提是已经安装 Microsoft Excel 系列产品中的一个,下面给出 Spreadsheet Link 的安装步骤。

(1) 启动 Microsoft Excel, 选择“工具”菜单中的“加载宏”选项, 如图 11-3 所示。

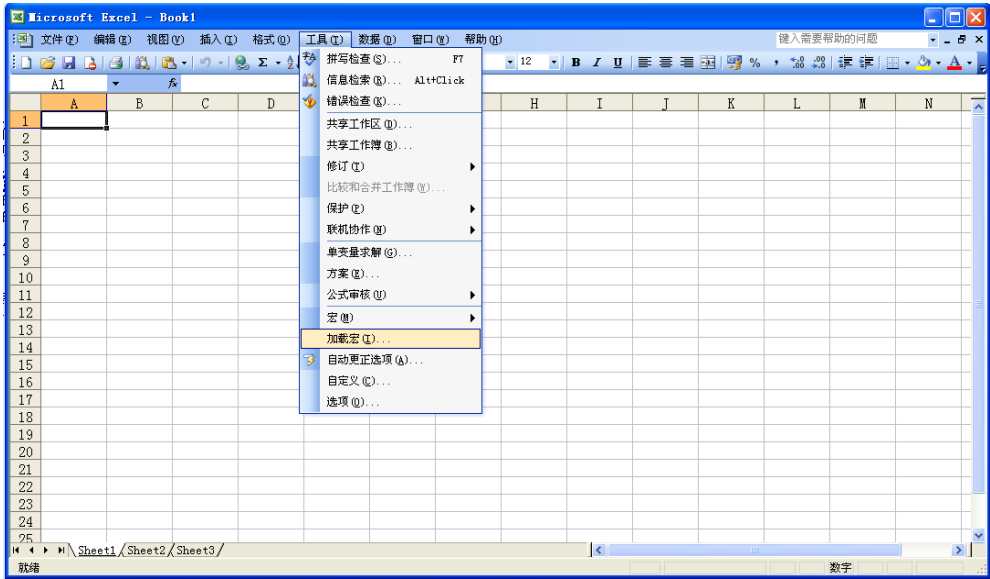


图 11-3 “加载宏”选项

(2) 在“加载宏”对话框中单击“浏览”按钮, 选择 MATLAB 路径下的\toolbox\exlink 子目录中的 exclink.xla 文件, 然后单击“确定”按钮, 如图 11-4 所示。

(3) 返回“加载宏”对话框, 此时已经填加并选中“Spreadsheet Link EX 3.1.1 for use with MATLAB”选项, 如图 11-5 所示。单击“确定”按钮即可加载 MATLAB, 并出现如图 11-6 所示的 Excel 窗口。

其中增加了 Spreadsheet Link 的工具条, 如图 11-7 所示。

Spreadsheet Link 的工具条中出现了 7 个执行 MATLAB 的命令按钮, 它们的含义如下:

- startmatlab: 启动 MATLAB。



图 11-4 加载宏文件

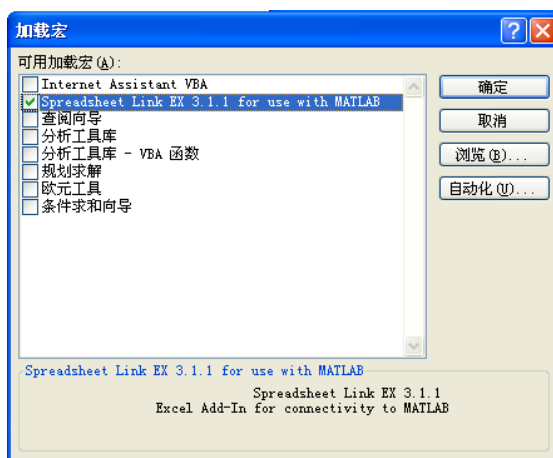


图 11-5 已经选中“Spreadsheet Link EX 3.1.1 for use with MATLAB”选项

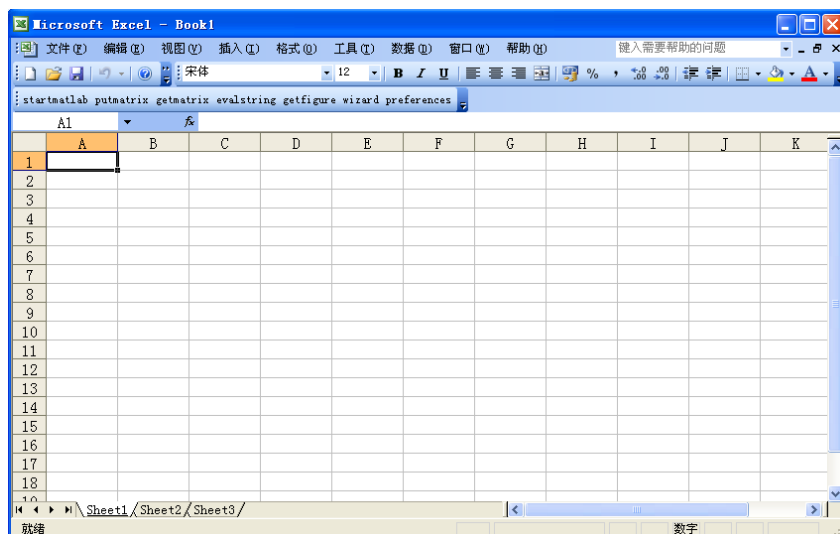


图 11-6 Excel 窗口

```
startmatlab putmatrix getmatrix evalstring getfigure wizard preferences
```

图 11-7 Spreadsheet Link 工具条

- putmatrix: 把数据传给 MATLAB。
- getmatrix: 从 MATLAB 提取数据。
- evalstring: 执行 MATLAB 命令。
- getfigure: 获取当前的 MATLAB 图形。
- wizard: 打开 MATLAB 函数向导。
- preferences: 打开 MATLAB/Spreadsheet Link EX 参数设置对话框。

• Spreadsheet Link 工具条在不需要时, 可以隐藏起来, 方法是: 在工具条上单击鼠标右键, 在弹出的菜单中取消选择 “Spreadsheet Link EX” 选项。

11.3.2 Spreadsheet Link 的启动和退出

按照上述步骤安装 Spreadsheet Link 后, 将在每次启动 Excel 时自动启动 Spreadsheet Link 和 MATLAB。

如果希望改变此种启动方式, 可以在 Excel 编辑框中输入 “=MLAutoStart(“no”)” 语句, 执行后即可改变设置, 如图 11-8 所示; 当然如果希望恢复原设置, 可以输入 “=MLAutoStart(“yes”)” 语句。

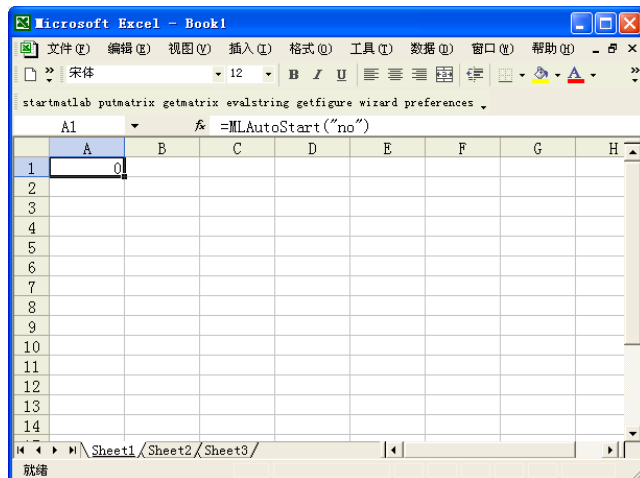


图 11-8 输入命令

对于 Spreadsheet Link 和 MATLAB 没有自动启动的情况, 可以在 Excel 平台中手动启动。首先在 “工具” 菜单中选择 “宏” 选项, 如图 11-9 所示。

打开如图 11-10 所示的 “宏” 对话框, 输入 “MATLABinit”, 单击 “执行” 按钮后即可启动 Spreadsheet Link 和 MATLAB。

当退出 Excel 时将自动退出 Spreadsheet Link 和 MATLAB。如果希望在 Excel 平台中退出 Spreadsheet Link 和 MATLAB, 只需在编辑框中输入 “=MLClose()” 语句即可, 如图 11-11 所示。

11.3.3 Spreadsheet Link 的实际应用

在应用 Spreadsheet Link 时, 主要是实现 Excel 数据的读入, MATLAB 对数据的处理和显示, 以及将处理结果显示在 Excel 中。下面通过一个实例说明 Spreadsheet Link 的应用。

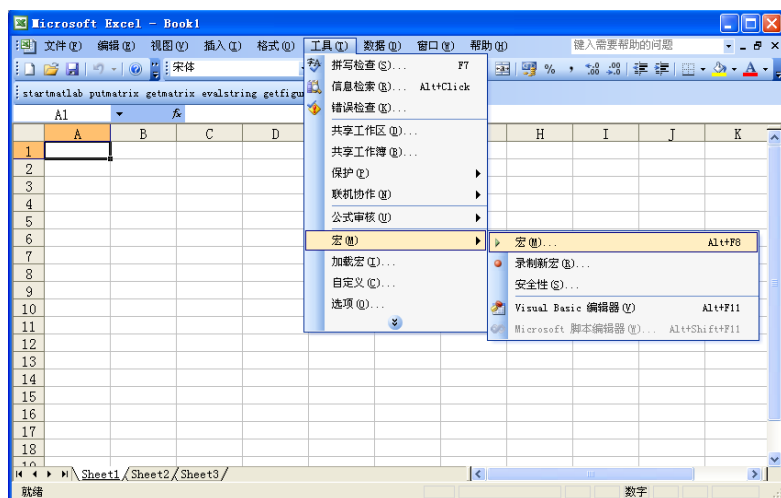


图 11-9 选择“宏”选项

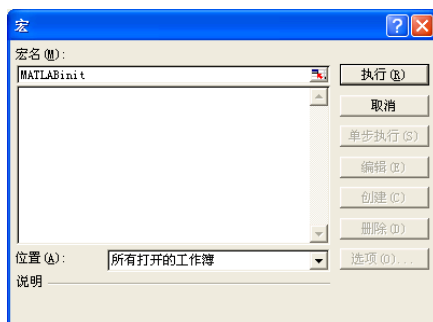


图 11-10 手动启动 Spreadsheet Link 和 MATLAB

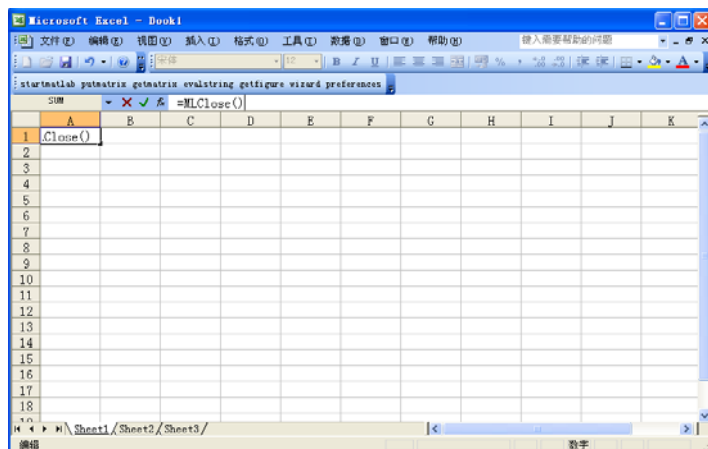


图 11-11 手动退出 Spreadsheet Link 和 MATLAB

【例 11-15】一个 Spreadsheet Link 的实例。

本例用数据表执行方式运行，使用 Excel 工作表组织和显示数据。该实例 ExliSamp.xls 位于 MATLAB 安装路径下的\toolbox\exlink 子目录中。

启动 Excel、Excel link 和 MATLAB，打开示例文件 ExliSamp.xls。

单击 ExliSamp.xls 中的 Sheet1 标签，可以看到数据表中包含一个被命名为 DATA 的数据区

A4:C28，如图 11-12 所示。

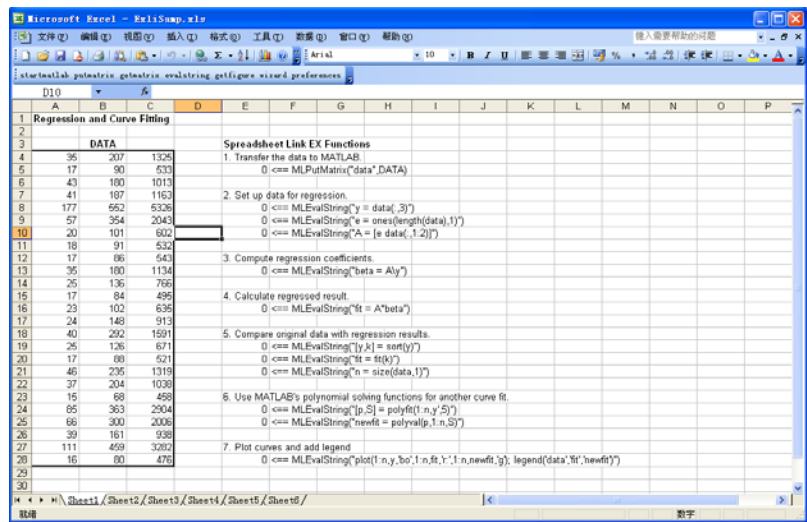


图 11-12 ExliSamp.xls 中的 Sheet1 数据页

- 该数据区包含了本例的数据集，按照下面的步骤进行操作：
- （1）选中 Sheet1 页的 E5 单元格，按“F2”键，然后按回车键执行 Excel Link 函数 MLPutMatrix ("data",DATA)，将 DATA 复制到 MATLAB 的 data 变量中，data 包含 3 个变量的 25 次观测值。
 - （2）选中 E28 单元格，按“F2”键，然后按回车键，对 E9 和 E10 单元格重复上面的操作。这些 Excel Link 函数让 MATLAB 以第 1 列和第 2 列的数据对应的变量为自变量，以第 3 列数据对应的变量为因变量进行回归。
 - （3）运行 E13 中的函数。使用 MATLAB 的“\”操作符计算回归系数，以便求解线性方程 $A \cdot \text{beta} = y$ 。
 - （4）运行 E16 中的函数。MATLAB 使用矩阵-向量乘法生成回归结果（fit）。
 - （5）运行 E19、E20、E21 中的函数。这些函数对原始数据与 fit 进行比较，将数据按升序排列，对 fit 采用相同的 permutationg，并创建一个表示观测数据的标量。
 - （6）运行 E24、E25 中的函数。用于拟合多项式，plotfit 函数生成一个 5 阶多项式 polyval，其次计算每个数据点上多项式的结果，确定拟合精度（newfit）；
 - （7）运行 E28 中的函数。添加图列得到用 MATLAB 的 plot 函数画出的图形，其中“○”代表原始数据，“.....”代表回归结果，“——”代表多项式结果。可得到如图 11-13 所示的结果。

对图 11-13 中的 data、fit 和 newfit 三条曲线进行比较，可以发现数据具有很强的相关性，不是线性独立的，拟合曲线和原始数据并不是十分吻合，而 5 阶多项式拟合则显示了更加精确的数学模型。

- 这里还需要说明以下几点：
- Excel 本身也具有强大的数据显示功能，在 MATLAB 与 Excel 混合使用时可以充分利用它的这一优势。
 - Spreadsheet Link 函数名对字母的大小写不作区分，如 MLPutMatrix 与 mlptmatrix 等价，而 MATLAB 函数名是区分大小写的。
 - Excel 表单的执行语句一般加等号，如“=MLGetMatrix("y","C1:L10")”，并且函数的参数用圆括号括起来。在宏中，函数名和第一个参数之间隔一个空格，不能使用圆括号。

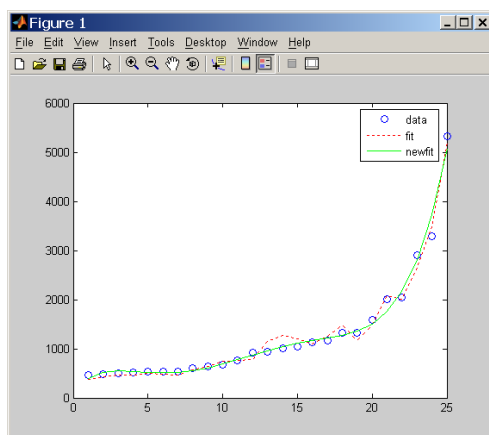


图 11-13 回归和曲线拟合结果显示

- 单击执行语句的单元显示对应的语句，执行完毕后单元显示值为 0。
- 执行语句中的标点符号和 MATLAB 中的一样，都必须在英文状态下输入。
- 在 Excel 的电子表格中直接输入函数即可，不要使用 Excel 的函数向导，否则会产生不可预料的结果。
- Excel Link 只处理 MATLAB 二维的数值数组和一维的字符数组（字符串）以及二维的细胞数组，而不能操作 MATLAB 的多维数组和结构体。

11.4 编译器

用户可能希望 MATLAB 能够更快地运行程序代码，或者希望获得可摆脱 MATLAB 运行环境而独立运行的可执行文件。为满足用户这方面的需要，出现了 MATLAB 编译器。

MATLAB 编译器可以编译 M 文件、MEX 文件以及其他的 MATLAB 代码，支持 MATLAB 所有的特性，包括对象、私有函数和方法。

编译器可以产生以下几种应用程序：

- 独立运行的程序：它们可以在没有安装 MATLAB 7.10 的机器上运行。
- C 和 C++ 共享库（在 Microsoft Windows 操作系统中为动态链接库 DLL）：这些共享库可以在没有安装 MATLAB 7.10 的用户机器上运行。

前面介绍的 MATLAB 程序都是运行在 MATLAB 平台上的，本节将介绍如何得到由 MATLAB 平台开发，经编译器编译后可以独立于 MATLAB 平台运行的程序。

11.4.1 编译器的安装和配置

在使用编译器前，需要进行安装和配置，在命令窗口输入如下语句：

```
mbuild -setup
```

命令窗口中的输出结果如下所示：

```
Please choose your compiler for building standalone MATLAB applications:
```

```
Would you like mbuild to locate installed compilers [y]/n?
```

其含义是“请选择独立运行程序的编译器，是否需要 mbuild 查找已安装的编译器”，选择

n 后，命令窗口中的输出结果如下所示：

```
Select a compiler:
[1] Lcc-win32 C 2.4.1
[2] Microsoft Visual C++ 6.0
[3] Microsoft Visual C++ 2005 SP1
[4] Microsoft Visual C++ 2008 Express
[5] Microsoft Visual C++ 2008 SP1

[0] None
```

Compiler:

这里列出了 MATLAB 支持的通用编译器。选择 0 后，命令窗口中的输出结果如下所示：

```
mbuild: No compiler selected. No action taken.
```

再次在命令窗口输入如下语句：

```
mbuild -setup
```

命令窗口中的输出结果如下所示：

```
Please choose your compiler for building standalone MATLAB applications:
```

```
Would you like mbuild to locate installed compilers [y]/n?
```

选择 y 后，命令窗口中的输出结果如下所示：

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in F:\MATLAB\sys\lcc
[2] Microsoft Visual C++ 6.0 in C:\Program Files\Microsoft Visual Studio
[2] Microsoft Visual C++ 6.0 in D:\新建文件夹
[0] None
```

Compiler:

此时显示系统中已经安装的编译器，并且列出安装的目录。Lcc-win32 C 2.4.1 是 MATLAB 自带的 C 编译器，不能用来编译 C++。选择 1 后，命令窗口中的输出结果如下所示：

```
Please verify your choices:
```

```
Compiler: Lcc-win32 C 2.4.1
```

```
Location: E:\MATLAB\sys\lcc
```

```
Are these correct [y]/n?
```

选择 y 后，命令窗口中的输出结果如下所示：

```
Trying to update options file: C:\Documents and Settings\Administrator\Application Data\MathWorks\
MATLAB\R2010a\compopts.bat
```

```
From template: E:\MATLAB\bin\win32\mbuildopts\lcccompp.bat
```

```
Done . . .
```

至此，完成了编译器的安装和配置。

另外，为了能够使用 MATLAB 编译器生成的组件，必须安装 MCR。可以先将 MATLAB 安装路径下的\toolbox\compiler\deploy\win32 子目录中的文件 MCRInstaller.exe 复制到另一路径，然后双击进行安装，直到提示安装结束。

11.4.2 编译命令

在 MATLAB 中使用 mcc 命令对 MATLAB 各类代码进行编译，它的具体用法如下所示：

MCC [-options] fun [fun2 ...]

其中 options 为选项，fun 和 fun2 为 MATLAB 代码文件，最常用的几种形式如下：

- `mcc -m myfun`：将 M 文件生成独立运行的同名 exe 文件。
- `mcc -m myfun1 myfun2`：将 M 文件主函数生成可独立运行的同名 exe 文件。
- `mcc -W lib:liba -T link:lib a0 a1`：将两个 M 文件生成名为 liba 的 C 共享库。
- `mcc -W cpplib:liba -T link:lib a0 a1`：将两个 M 文件生成名为 liba 的 C++ 共享库。

下面通过几个实例具体介绍编译命令的使用方法。

【例 11-16】编译函数求矩阵的特征值，具体的函数指令如下所示：

```
function eigA=myeig(u1,u2,u3,u4)
A=[eval(u1) eval(u2); eval(u3) eval(u4)] %数据类型强制转换
eigA=eig(A)
```

在命令窗口中输入如下语句：

```
mcc -m myeig
```

执行后，myeig.m 所在目录下增加了 myeig.exe、myeig.prj、myeig_main.c、myeig_delay_load.c 和 myeig_mcc_component_data.c 文件。

在命令窗口中输入如下语句：

```
type myeig_main.c
```

命令窗口中的输出结果请读者自行查阅。

由上可见编译器的版本为 V4.13。然后进入命令提示符模式（DOS 窗口），并将 myeig.m 所在目录设置为当前目录，在 DOS 窗口中输入如下语句：

```
myeig 1.2 2.4 3.6 4.8
```

DOS 窗口中的运行结果如图 11-14 所示。

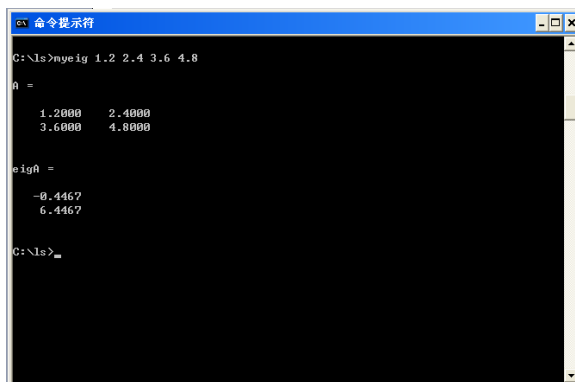


图 11-14 在 DOS 窗口得到的结果

这里需要说明的是，语句“`A=[eval(u1) eval(u2); eval(u3), eval(u4)]`”中的强制转换对于编译来说非常重要，否则系统会将输入按照字符类型进行处理。


11.4.3 项目开发工具

下面通过一个实例介绍项目开发工具的应用。

【例 11-17】利用项目开发工具构建 C 共享库，具体步骤和操作如下：

在 MATLAB 主窗口中选择“File”→“New”→“Deployment Project”菜单命令，打开如图 11-15 所示的界面。

在“Name”框中填写项目名称（这里为 `myfunction.prj`），在“Location”框中选择保存的路径，然后单击“OK”按钮，出现如图 11-16 所示的界面。

选中[add main file]可以打开文件选择对话框。选择如下内容的函数文件并单击图标，可以打开如图 11-17 所示的界面。

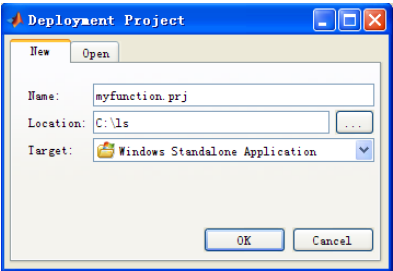


图 11-15 项目开发工具开始界面

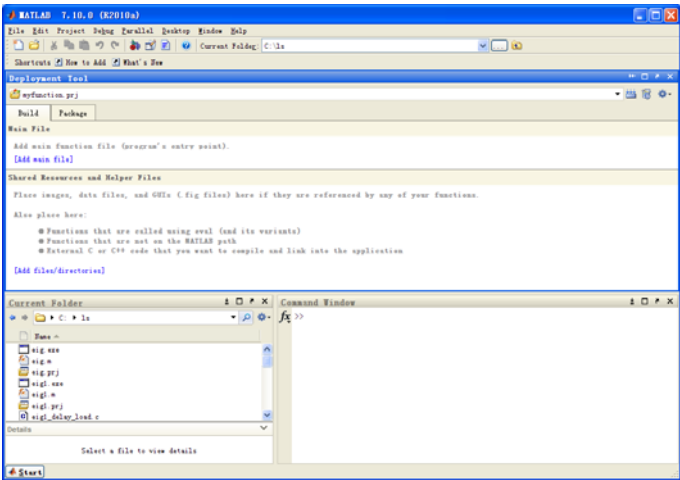


图 11-16 项目开发工具界面

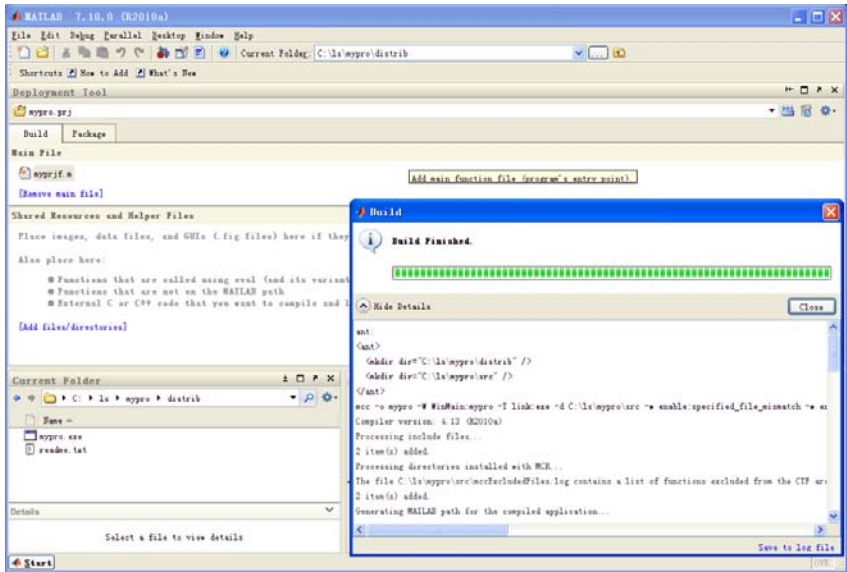


图 11-17 编译后的结果

```
function y=myprjf(x)
y=sin(eval(x))
```

编译后在指定目录下增加了 mypro.prj 文件和子目录 mypro，目录 mypro 下还包含两个下一级目录 distrib 和 src。

最后进入 DOS 窗口，并将当前目录设置为 myprjf.m 所在目录下的\mypro\distrib 目录（或\mypro\src 目录），再如图 11-18 所示输入如下语句：

```
mypro pi
```

在 DOS 窗口中的运行结果如图 11-19 所示。

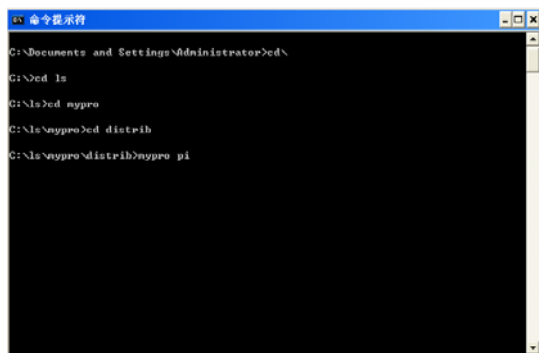


图 11-18 在 DOS 窗口输入命令

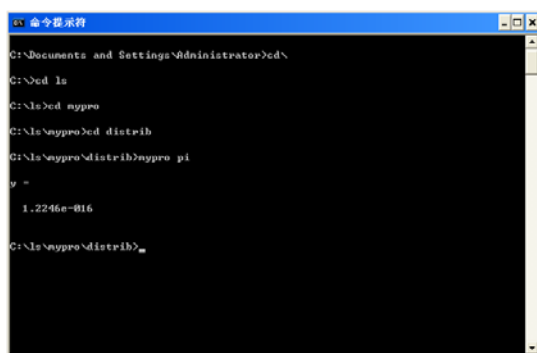


图 11-19 在 DOS 窗口得到的结果

11.5 MATLAB 与 C 语言混合使用

MATLAB 与 C 语言的混合使用包括两个方面，即在 MATLAB 平台上调用 C 语言资源，以及在 C 语言平台上调用 MATLAB 资源。

在 MATLAB 中使用 mex 命令将 C 语言文件编译成 MEX 文件形式的共享库，以便 MATLAB 调用。尽管在 MATLAB 中调用 MEX 文件会比较简单，但是一般格式编写的 C 语言程序代码并不能直接编译成可以被 MATLAB 调用的 MEX 文件，只有符合某种特殊格式的 C 程序代码才能编译成为 MEX 文件。

【例 11-18】编写 C 语言 MEX 程序代码，实现将输入量的 2 倍进行输出。

首先编写 C 语言程序代码，如下所示：

```
#include<math.h>
void timestwo(double y[],double x[])
{
y[0]=2.0*x[0]
return;
}
```

然后编写对应的 C 语言 MEX 程序代码，如下所示：

```
#include"mex.h"
void timestwo(double y[],double x[])
{
y[0]=2.0*x[0];
```

```

}
void mexFunction(int nlhs,mxArray*plhs[],int nrhs,const mxArray*prhs[])
{
double *x,*y;
int mrows,ncols;                                /*检测输入参数个数*/

if (nrhs!=1)
{
mexErrMsgTxt("One input required.");
}
else if(nlhs>1)
{
mexErrMsgTxt("Too many output arguments");          /*确保输入参数是标量，并且是正值*/
}
mrows=mxGetM(prhs[0]);
ncols=mxGetN(prhs[0]);
if
(!mxIsDouble(prhs[0])||mxIsComplex(prhs[0])||(mrows==1&&ncols==1))
{mexErrMsgTxt("Input must be a noncomplex scalar double.");    /*创建返回参数的属性*/
}
plhs[0]=mxCreateDoubleMatrix(mrows,ncols,mxREAL);              /*为输入参数和输出参数分配指针*/
x=mxGetPr(prhs[0]);
y=mxGetPr(plhs[0]);
timwstwo(y,x);                                                  /*调用子函数*/
}

```

将上述文件保存为 `timstwo.c`, 在 MATLAB 环境中进行编译。在命令窗口中输入如下语句:

```

mex timstwo.c          %编译代码
which timstwo.dll      %定位编译后的文件

```

最后得到编译后的文件信息为:

D:\softWare\MATLAB 7.10\work\timstwo.dll

命令窗口中的输出结果如下所示:

```

x=2;
y=timstwo(x)
y=
4

```

11.6 MATLAB 与外部设备和互联网交互

MATLAB 之所以具备广泛的应用领域, 具有十分强大的功能, 一方面是由于 MATLAB 中具有能实现与外部设备进行通信的大量函数, 包括串口、网络以及即插即用等设备, 另一方面

还具备能够实现设备驱动程序编写的大量函数。

现在，通过两个简单的实例来了解 MATLAB 这方面的强大功能。

【例 11-19】举例说明 USB 接口设备拍摄的图像的读取。

在命令窗口中输入如下语句：

```
winvideoinfo = imaqhwinfo('winvideo')
device1 = winvideoinfo.DeviceInfo(1)
device1.DefaultFormat
```

命令窗口中的输出结果如下所示：

```
winvideoinfo =
    AdaptorDllName: 'F:\MATLAB\toolbox\imaq\imaqadaptors\win32\mwwinvideoimaq.dll'
    AdaptorDllVersion: '3.5 (R2010a)'
    AdaptorName: 'winvideo'
    DeviceIDs: {[1]}
    DeviceInfo: [1x1 struct]

device1 =
    DefaultFormat: 'YUY2_1280x1024'
    DeviceFileSupported: 0
    DeviceName: 'USB 视频设备'
    DeviceID: 1
    ObjectConstructor: 'videoinput('winvideo', 1)'
    SupportedFormats: {1x7 cell}

ans =
YUY2_1280x1024
```

上述结果显示，系统安装一个 winvideo 设备，该设备是 USB 视频设备，默认图像格式为 YUY2_1280x1024。

其次在命令窗口中输入如下语句：

```
clear
clc
obj=videoinput('winvideo',1);
preview(obj);
```

在屏幕上则出现如图 11-20 所示的摄像头的连续图像。

【例 11-20】举例说明 MATLAB 与互联网的交互功能。

输入如下所示的代码并保存：

```
t = tcpip('www.baidu.com',80);      % 设置连接网站和连接端口
set(t, 'InputBufferSize', 5000);    % 设置连接的缓冲区
fopen(t)                             % 打开连接
fprintf(t, 'GET /');                 % 得到首页内容
get(t, 'BytesAvailable')             % 显示可用字节数
```

```
A=[]; %设置初值
while (get(t, 'BytesAvailable') > 0) %如果剩余字节数不为 0
    A = [A, fscanf(t, 13); %fscanf(t)按行读取数据, 并且文件指针移动
    %13 为回车键对应的 ASCII 码
end
if isempty(A) %判断是否为空
    disp('No data');
else
    A=A %显示读取结果
end
fclose(t); %关闭连接
delete(t); %删除连接
clear t %清除连接
```

执行结果等效于打开指定网站主页后, 单击页面菜单中“查看源代码”选项的结果。

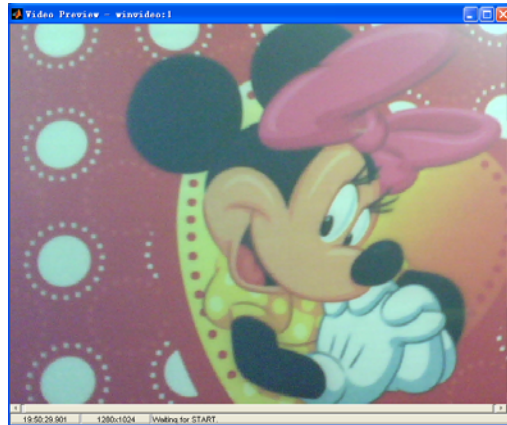


图 11-20 截屏结果

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036

经典 实用 权威

MATLAB 2010从入门到精通

本书主要从实际应用角度和快速入门角度对MATLAB 2010进行通用性介绍，没有局限于某些具体领域介绍某个或某几个工具箱，而着重于讲清和讲透通用内容，为具体应用打下坚实的基础。

本书对MATLAB 2010进行了详细的介绍和讲解，以实际应用为导向，力求做到由简入繁，并达到快速入门和迅速提高的目的。

本书条理明晰、深入浅出，并配有大量实用的例子，适合作为学习或使用MATLAB这一重要工具的本科生、研究生、教师以及广大科技工作者的参考书。

MATLAB 2010从入门到精通

软件测试从入门到精通

设计模式从入门到精通

C语言从入门到精通

Visual C# 2010从入门到精通

Visual Basic 2010中文版从入门到精通

UML与Rational Rose 2003从入门到精通

Java SE 6从入门到精通

Oracle 11g从入门到精通

SQL从入门到精通

SQL Server 2008中文版从入门到精通

Windows 7中文版从入门到精通

Windows Server 2008从入门到精通

Project 2010中文版从入门到精通

Word 2010中文版从入门到精通

Excel 2010中文版从入门到精通

PowerPoint 2010中文版从入门到精通

Crystal Reports 2008水晶报表从入门到精通



责任编辑：李红玉

封面设计：李娜

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书



定价：64.00元