

MATLAB

开发应用案例解密



MATLAB

N个实用技巧

——MATLAB中文论坛精华总结

北京航空航天大学出版社

刘焕进 王 辉 李 鹏 刘衍琦 编著



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

MATLAB 开发实例系列图书

MATLAB N个实用技巧

——MATLAB中文论坛精华总结

刘焕进 王 辉 李 鹏 刘衍琦 编著

北京航空航天大学出版社

北京航空航天大学出版社

内 容 简 介

本书是诸位作者多年使用 MATLAB/Simulink 并帮助论坛网友解决实际问题经验的总结,也是 MATLAB/Simulink 应用实战技巧的真诚分享。希望通过本书给读者提供最便捷、最直接的支持,解决最具体、最实际的难题。

本书共 8 章,提供了 99 个实用技巧,涵盖了 MATLAB/Simulink 安装、启动与配置;基础知识;绘图操作;文件操作;论文发表;程序自动化运行;GUI 高级使用;MATLAB/Simulink 和其他语言混合编程等相关知识及其应用。随着各方面反馈信息的不断汇总和更新,作者还将不断提供新的技巧,并尝试通过网络互动平台协助本书读者解决实际应用中遇到的第 N 个问题。

本书适合 MATLAB/Simulink 初学者作为学习资料和答疑手册使用;对有一定 MATLAB/Simulink 应用基础的读者,可将本书作为技巧交流的平台;还可作为相关专业的高校师生及科研人员的参考书。

图书在版编目(CIP)数据

MATLAB N 个实用技巧 / 刘焕进等编著. --北京 :
北京航空航天大学出版社, 2011. 4

ISBN 978 - 7 - 5124 - 0324 - 6

I. ①M… II. ①刘… III. ①计算机辅助计算—软件
包, MATLAB IV. ①TP391.75

中国版本图书馆 CIP 数据核字(2011)第 007285 号

版权所有,侵权必究。

MATLAB N 个实用技巧

——MATLAB 中文论坛精华总结

刘焕进 王 辉 李 鹏 刘衍琦 编著

责任编辑 史 东

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) <http://www.buaapress.com.cn>

发行部电话:(010)82317024 传真:(010)82328026

读者信箱: bhpress@263.net 邮购电话:(010)82316936

有限公司印装 各地书店经销

*

开本:787×1092 1/16 印张:22.75 字数:582 千字

2011 年 4 月第 1 版 2011 年 4 月第 1 次印刷 印数:5 000 册

ISBN 978 - 7 - 5124 - 0324 - 6 定价:42.00 元

序

MATLAB 中文论坛(www.ilovematlab.cn)自创办至今,在短短的 3 年时间里,我们见证了 MATLAB®/Simulink®^① 在中国的普及。毫不夸张地说,仅在中国大陆,使用过 MATLAB/Simulink 的人数已超过 100 万,正在使用的人数超过 30 万。MATLAB/Simulink 提供的上百种工具箱几乎已经涵盖了所有的工程与科学领域的研究和应用。我们也坚信,MATLAB/Simulink 将会以其卓越性能和优秀品质吸引更多的用户!

在为此书作序的时候,MATLAB 中文论坛的主题已经达到 10 万,帖子超过 100 万,有效会员 30 万。MATLAB 的基础技巧已经在论坛里得到了充分的展示与探讨,大家常见的问题大部分获得了解答!是时候做一个小小的总结,把 MATLAB/Simulink 里常见的一些小问题、技巧集中讨论一下了。相信这将有助于 MATLAB 使用者快速入门,轻松解决一些常见的问题。

我们认为,基础知识主要包括:绘图操作技巧;文件读写技巧;论文图片使用技巧;程序自动化运行技巧;GUI 高级编程技巧;MATLAB 与其他程序混合编程以及 MATLAB 安装、启动过程中遇到的一些常见问题。

试想一下,你在发表 SCI 论文的时候,准备的图片总是不尽人意——legend 总是覆盖曲线,bar 图上无法加上 errorbar,坐标系里无法添加子坐标系等。如果你能掌握书中所讲到的相关技巧,你的论文图片必将更加专业化。

你是否曾经遇到过,一个大型优化的程序要运行好几天,自己得经常在电脑前查看运行过程。其实完全没有这个必要,你可以让你的程序定时给你发送邮件或手机短信,让你对程序的运行状态了如指掌。这本书里就介绍了这方面的相关技巧。

诸如此类的小技巧可能不会对你的研究产生质的改变,然而却能在很大程度上让你的研究生活变的很惬意。我们坚信,此书能让你在 MATLAB/Simulink 的使用过程中,有更好的体验!

熟悉 MATLAB 中文论坛的读者都了解,论坛喜欢把一切问题都案例话,这样非常方便用户模仿。我们坚信,学习任何一种语言,模仿都是第一步。在我们提供的技巧里,每一个技巧都有配套的案例,非常方便学习。同样,别忘了此书的作者为你提供“有问必答”服务。阅读此书过程中,有任何疑问,随时可以在该书位于 MATLAB 中文论坛的在线交流版块(<http://www.iLoveMatlab.cn/forum-178-1.html>)向作者提问!

^① MATLAB®、Simulink® 是 MathWorks 公司的注册商标,本书其余各处均用 MATLAB 代替 MATLAB®,用 Simulink 代替 Simulink®。

北京航空航天大学出版社“MATLAB 开发实例系列图书”的出版还得到了美国 MathWorks 公司 Book Program 的支持,在此表示谢意,并特别感谢 Lauren Tabolinsky 女士为此提供的各种帮助。国内读者如果想购买 MATLAB 软件及系列产品,请直接联系:

MathWorks 中国

电话: +86 - 10 - 59827000

E-mail: info@mathworks.cn

张延亮(math)

(MATLAB 中文论坛创始人)

2010 年 12 月,多伦多

北京航空航天大学出版社

前 言

编写目的

MATLAB 作为当今世界上应用最为广泛的高性能计算和可视化软件,具有非常强大的科学计算、数值分析、图形显示、系统分析和建模等功能,在信号处理、图像处理、通信工程、自动控制等领域得到了广泛应用。

目前,市面上已有不少介绍 MATLAB 的书籍。这些书籍有的侧重于讲述 MATLAB 的总体功能,如《MATLAB 从入门到精通》、《MATLAB 宝典》,等等;有的侧重于讲述某一个功能模块的知识,如《MATLAB 与 C/C++ 混合编程》、《MATLAB/Simulink 建模与仿真》,等等;有的侧重于讲述 MATLAB 在某一专业领域的应用,如《MATLAB 时频分析技术及其应用》、《MATLAB 2007 图像处理技术与应用》,等等。

但是,读者在使用 MATLAB 解决实际问题时,会遇到各种各样的问题,这些问题可能涉及 MATLAB 的方方面面,如 MATLAB 基本语法、数据可视化、图形用户界面设计、文件输入/输出、MATLAB 工具箱、Simulink 仿真、MATLAB 与其他编程语言混合编程等。读者最希望能在最短的时间内、以最好的效果来解决所遇到的问题,而不是再在众多的 MATLAB 资料中查阅相关知识点,并调试所编写的代码。本书正是基于此目的来编写的。

本书特点

书中所有技巧均取材于 MATLAB 中文论坛,是对论坛会员所提常见问题的提炼、汇总。本书在编写时力求所选技巧的一般性和通用性,尽量不涉及较深的专业知识(如图像处理、信号处理、神经网络等)。因此,本书适合于各个专业的读者使用。

学习任何一门编程语言,模仿都是非常重要的一步。对于本书所选的所有技巧,书中都详细地给出了技巧的用途以及实现该技巧所用到的 MATLAB 知识点,并详细介绍了技巧的实现步骤,非常方便读者学习和模仿。此外,书中每一个案例都给出了完整的代码,所给出的代码简洁、高效,便于用户直接重用这些代码来解决自己的问题。

此外,本书的作者将会对读者提供“有问必答”服务。读者在阅读此书的过程中,有任何疑问,都可以随时在该书的在线交流版块向作者提问,本书作者会在第一时间回答读者的提问。

本书主要内容

本书在编写过程中并不是仅仅将所有技巧简单地罗列,而是将所有技巧分门别类,并遵循由浅入深的原则。

本书的主要内容安排如下:

第 1 章 安装、启动和配置。本章主要介绍 MATLAB 软件的相关操作,包括 MATLAB 软件的安装方法和步骤、MATLAB 软件的快速启动、内存的优化配置、工具箱的添加、中文字体的设置与显示、工作路径的设置与修改、编译器的安装与配置、中文 Simulink 模型的打开等相关问题和技巧。

第 2 章 基础知识。本章主要介绍 MATLAB 编程的相关基础知识,包括 MATLAB 基

础语法、图形窗口及控件的操作方法、数组和矩阵的操作、字符串的操作、判断函数的使用、全局变量的使用、动画的制作和保存、典型控件的使用、GUI 开发基本方法、MATLAB 程序的调试和编译等相关问题和技巧。

第 3 章 绘图操作技巧。本章主要介绍 MATLAB 绘图和可视化操作技巧。包括绘图操作基本方法、坐标轴对象的操作、色图矩阵的控制、隐函数的绘图等相关问题和技巧。

第 4 章 文件操作技巧。本章主要介绍在 MATLAB 程序中读写其他文件的技巧。包括创建和删除文件或文件夹,在 MATLAB 程序中创建 Microsoft Word、Microsoft Excel 文档,读写 MAT 文件、Excel 文件、文本文件,向同一文件中追加存储数据等相关问题和技巧。

第 5 章 论文发表专用技巧。本章主要介绍读者在发表论文时所遇到的利用 MATLAB 进行科学计算以及数据可视化方面的技巧。包括将运行结果导出为高质量的图片、在界面上显示数学公式及特殊符号、为图形添加图例说明、控制数据的显示精度及运算精度等相关问题和技巧。

第 6 章 程序自动化运行技巧。本章主要介绍 MATLAB 程序自动化运行方面的技巧。包括定时器的使用、定时发送邮件和短信、监控拍照以及程序暂停、终止等相关问题和技巧。

第 7 章 GUI 高级技巧。本章主要介绍利用 MATLAB 进行图形用户界面开发的高级技巧。包括句柄结构的使用、同一 MATLAB 程序内或不同 MATLAB 程序之间的数据传递、控件的动态创建、图像的放大和裁剪、标签页的制作、不同坐标轴中的点的坐标变换、在 GUI 中控制 Simulink 仿真过程等相关问题和技巧。

第 8 章 MATLAB 与其他语言混合编程。本章主要介绍 MATLAB 与其他编程语言之间的混合编程技巧。包括 MATLAB 与 VB、C++、C#、LabVIEW 等的混合编程;MATLAB 与 Access、MySQL 数据库的混合编程;将常用的 CAD 模型导入 MATLAB 进行仿真等相关问题和技巧。

读者对象

本书所选的技巧既涉及 MATLAB 的基础知识,也涉及 MATLAB 的高级应用。本书在写作过程中力求通俗易懂,所选案例具有很强的代表性和通用性。因此,无论对于 MATLAB 的初学者还是具有一定基础的高级用户,本书都是一本难得的参考用书。同时,本书也适合作为广大高校师生和科研工作人员的参考用书。

致 谢

本书由刘焕进、王辉、李鹏、刘衍琦负责编写。本书在编写过程中,得到了合肥工业大学、山东大学、河海大学以及大连理工大学有关师生的热心帮助和大力支持;MATLAB 中文论坛创始人张延亮(math)博士对本书的编写进行了全程指导;MATLAB 中文论坛的会员也对本书的编写表示了极大的关注和支持。此外,北京航空航天大学出版社的编辑们在本书的编写和校对过程中付出了辛勤的劳动,并提出了大量宝贵的意见,在此一并表示衷心的感谢。

由于编写时间仓促,加之作者学识所限,书中如有错误和疏漏之处,恳请广大读者和各位专家的批评指正。本书勘误网址:<http://www.iLoveMatlab.cn/thread-114467-1-1.html>。

编著者

2010 年 2 月

目 录

第 1 章 安装、启动和配置	1
1.1 技巧 1: MATLAB 的安装	1
1.2 技巧 2: MATLAB 的启动	8
1.3 技巧 3: 内存的优化配置	12
1.4 技巧 4: 工具箱的添加	16
1.5 技巧 5: 中文字体的设置与显示	18
1.6 技巧 6: 工作路径的设置与修改	22
1.7 技巧 7: MATLAB 自带的 MEX 和 VR 编译器的安装和配置	26
1.8 技巧 8: 解决 Simulink 模型打不开的问题	28
第 2 章 基础知识	31
2.1 技巧 9: 操作图形窗口及其控件的通用方法——set 和 get 命令	31
2.2 技巧 10: 定义回调函数需遵循的语法规则	33
2.3 技巧 11: 元胞数组(Cell Array)的使用方法	35
2.4 技巧 12: 结构数组(struct array)的使用方法	38
2.5 技巧 13: 矩阵(Matrix)的常用操作方法	42
2.6 技巧 14: 字符串的操作方法	46
2.7 技巧 15: 判断函数的使用方法	51
2.8 技巧 16: varargin、varargout、nargin 和 nargout 的使用方法	56
2.9 技巧 17: 执行字符串中包含的 MATLAB 表达式	59
2.10 技巧 18: 实现函数 M 文件和基本工作空间中变量的相互调用	62
2.11 技巧 19: 调用外部程序打开指定文件	65
2.12 技巧 20: 定义和使用全局变量	68
2.13 技巧 21: 计算程序运行所需的时间	70
2.14 技巧 22: 动画的制作和保存	72
2.15 技巧 23: 根据离散点拟合椭圆方程	76
2.16 技巧 24: MATLAB 中类的定义及使用	78
2.17 技巧 25: 给控件、菜单、工具条定义快捷键	80
2.18 技巧 26: MATLAB 程序的调试(Debug)	87
2.19 技巧 27: 在 MATLAB 程序中使用提示音	91
2.20 技巧 28: 将 MATLAB 程序编译成可执行文件	95
2.21 技巧 29: Pop-up Menu 和 Listbox 控件的使用方法	101
2.22 技巧 30: Button Group 和 Panel 控件的使用方法	104
2.23 技巧 31: 使用 Static Text、Edit Text 和 Listbox 控件实现多行显示	110
2.24 技巧 32: Uitable 控件的使用方法	113
2.25 技巧 33: 滑动条 Slider 的使用方法	117

2.26	技巧 34: 进度条 Waitbar 的使用方法	119
2.27	技巧 35: 在 MATLAB 程序中响应鼠标的操作	124
2.28	技巧 36: 在 MATLAB 程序中响应键盘的操作	127
2.29	技巧 37: MATLAB 图形用户界面开发基本方法	128
2.30	技巧 38: MATLAB Notebook 的使用方法	133
2.31	技巧 39: 符号函数、内联函数及匿名函数的操作方法	137
2.32	技巧 40: 在 MATLAB 程序中操作系统剪贴板	141
第 3 章	绘图操作技巧	144
3.1	技巧 41: 绘图操作基本方法	144
3.2	技巧 42: 利用 annotation 命令实现图形的标注	150
3.3	技巧 43: 坐标轴对象的 ButtonDownFcn 回调函数的调用	152
3.4	技巧 44: 坐标轴对象使用 subplot 后句柄失效的解决方法	154
3.5	技巧 45: 高维(四维)数据可视化方法	157
3.6	技巧 46: 色图矩阵(colormap)的控制	161
3.7	技巧 47: 以图片为背景建立坐标轴绘图	164
3.8	技巧 48: MATLAB 中隐函数的绘图方法	166
第 4 章	文件操作技巧	169
4.1	技巧 49: 通过 MATLAB 程序创建和删除文件或文件夹	169
4.2	技巧 50: 对文件的路径名、扩展名等各部分信息的操作	171
4.3	技巧 51: 取得指定文件夹下的所有文件	172
4.4	技巧 52: 通过 MATLAB 程序复制或移动文件/文件夹	174
4.5	技巧 53: 向同一个数据文件(.txt 或 .mat)中追加存储数据	177
4.6	技巧 54: 读/写 Microsoft Excel 文件	179
4.7	技巧 55: 在 MATLAB 程序中创建 Microsoft Excel 文档	182
4.8	技巧 56: 在 MATLAB 程序中创建 Microsoft Word 文档	186
4.9	技巧 57: MAT 文件的操作方法	188
4.10	技巧 58: 在 MATLAB 中读/写文本文件(.txt 文件)	191
4.11	技巧 59: 打开/保存文件对话框的使用方法	196
4.12	技巧 60: 修改 GIF 文件的内容	200
4.13	技巧 61: 在 MATLAB 中制作电子相册	203
第 5 章	论文发表专用技巧	206
5.1	技巧 62: 导出 figure 为 jpg、tiff 等适合论文使用的图片	206
5.2	技巧 63: 在界面上显示数学公式和特殊字符	210
5.3	技巧 64: 导出运行矩阵为 Latex 表格	212
5.4	技巧 65: 控制数据的显示精度和参与运算的精度	215
5.5	技巧 66: 为绘制的图形添加图例(legend)	217
第 6 章	程序自动化运行技巧	224
6.1	技巧 67: 在 MATLAB 程序中使用定时器	224
6.2	技巧 68: 利用 MATLAB 程序定时发送邮件、短信	227
6.3	技巧 69: 利用 MATLAB 程序定时使用摄像头拍照	229

6.4	技巧 70: 实现程序的暂停、继续、终止功能	234
第 7 章	GUI 高级技巧	240
7.1	技巧 71: 在 MATLAB 程序中使用句柄结构	240
7.2	技巧 72: 同一 MATLAB 程序内不同控件或函数之间的数据传递	243
7.3	技巧 73: 不同 MATLAB 程序之间的数据传递	247
7.4	技巧 74: 多个 MATLAB 程序之间数据的双向传递	250
7.5	技巧 75: 在一个程序中操作另一个程序中的控件或对象	252
7.6	技巧 76: 在界面上动态创建控件	254
7.7	技巧 77: 屏幕上的点在不同坐标轴中的坐标变换	256
7.8	技巧 78: 给放大的图像加上滚动条以方便浏览	260
7.9	技巧 79: 图像的定点放大和按任意形状裁剪	261
7.10	技巧 80: 取得 Data Cursor 指示的数值以及改变其显示格式	266
7.11	技巧 81: 改变界面窗口左上角的 logo 的方法	269
7.12	技巧 82: GUI 工具按钮与下拉菜单的组合	270
7.13	技巧 83: 在 GUI 中制作标签页	274
7.14	技巧 84: 在界面上实现树形浏览文件的功能	280
7.15	技巧 85: 实现 GUI 控件的双击和单击	285
7.16	技巧 86: 使用鼠标拖放来改变坐标轴中的图形大小	289
7.17	技巧 87: 修改菜单、列表框或弹出菜单等各条目的字体和颜色	292
7.18	技巧 88: 在 GUI 中控制 Simulink 仿真过程及结果显示	294
7.19	技巧 89: 在 GUI 中启动和停止 Simulink 仿真	297
7.20	技巧 90: 编程实现图像的缩放和移动功能	300
第 8 章	MATLAB 与其他语言混合编程	303
8.1	技巧 91: 在 MATLAB 中制作 COM 组件	303
8.2	技巧 92: MATLAB 与 VB 混合编程	306
8.3	技巧 93: MATLAB 与 C++ 混合编程	310
8.4	技巧 94: 在 MATLAB 程序中使用动态链接库文件	317
8.5	技巧 95: MATLAB 与 Access 数据库混合编程	321
8.6	技巧 96: MATLAB 与 MySQL 数据库混合编程	333
8.7	技巧 97: MATLAB 与 LabVIEW 混合编程	338
8.8	技巧 98: MATLAB 与 C# 混合编程	343
8.9	技巧 99: 将常用 CAD 模型导入 MATLAB 中进行仿真	348
参考文献		354

1.1 技巧 1: MATLAB 的安装

1.1.1 技巧用途

MATLAB 作为当今世界上应用最为广泛的数学软件,具有非常强大的数值计算、数据分析处理、系统分析、图形显示、符号运算等功能,已在生物工程、图像处理、语音处理、控制等领域得到广泛应用。

本节以 MATLAB R2008b 为例,详细介绍安装步骤。其他版本的 MATLAB 软件的安装可参考相应的随机文档。

1.1.2 技巧实现

1. 安装步骤

在安装 MATLAB 某一个版本的软件之前,用户必须首先取得 MathWorks 公司提供的安装许可文件(license file)和文件安装码(file installation key)。文件安装码用来安装 MATLAB 软件,安装许可文件用来激活 MATLAB 软件。已经安装的 MATLAB 软件必须激活后才能使用。

第 1 步:使用光盘自启动安装程序或者双击安装文件中的 setup.exe 文件启动安装程序,弹出欢迎对话框,如图 1.1-1 所示。该对话框有两个选项:“Install automatically using the

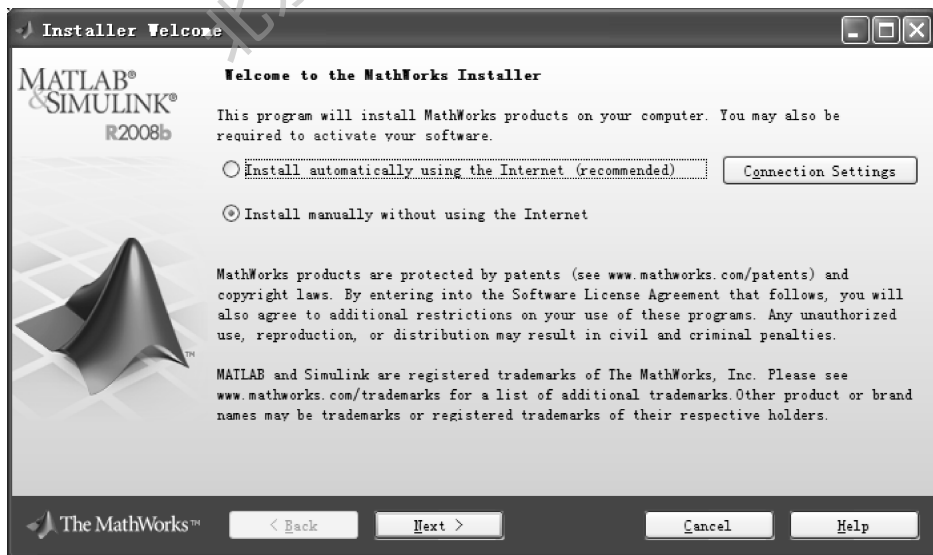


图 1.1-1 欢迎对话框

Internet (recommended)”和“Install manually without using the Internet”。第 1 个选项为推荐的方式,通过 Internet 自动安装;第 2 个选项为手动安装,不使用 Internet。这里选择手动安装的方式。选好之后,单击 Next 按钮继续安装。

第 2 步:弹出许可协议对话框,如图 1.1-2 所示。用户需仔细查看软件的许可协议,选择 Yes 表示接受许可协议条款。然后,单击 Next 按钮继续安装。

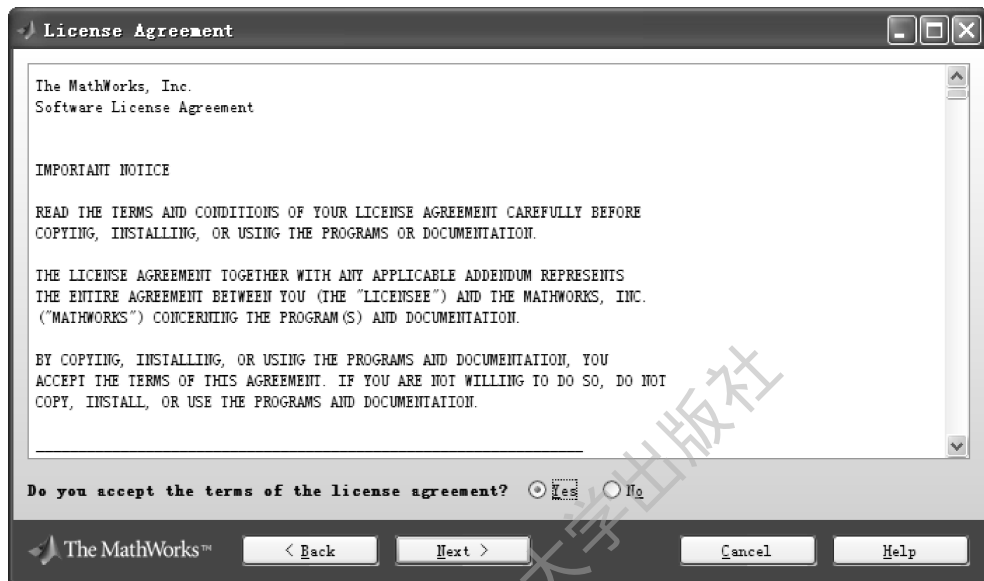


图 1.1-2 许可协议对话框

第 3 步:弹出软件安装许可密钥对话框,如图 1.1-3 所示。选中“I have the File Installation Key for my license”单选按钮,在编辑框中输入安装许可密钥,然后单击 Next 按钮继续安装。



图 1.1-3 软件安装许可密钥对话框

第 4 步：弹出安装类型对话框，如图 1.1-4 所示。对话框有 Typical(典型安装)和 Custom(自定义安装)两个选项。如果用户对 MATLAB 产品比较熟悉，可以选择自定义安装，之后再选择要安装的 MATLAB 组件及其工具箱等组件。这里选择 Custom，然后单击 Next 按钮继续安装。

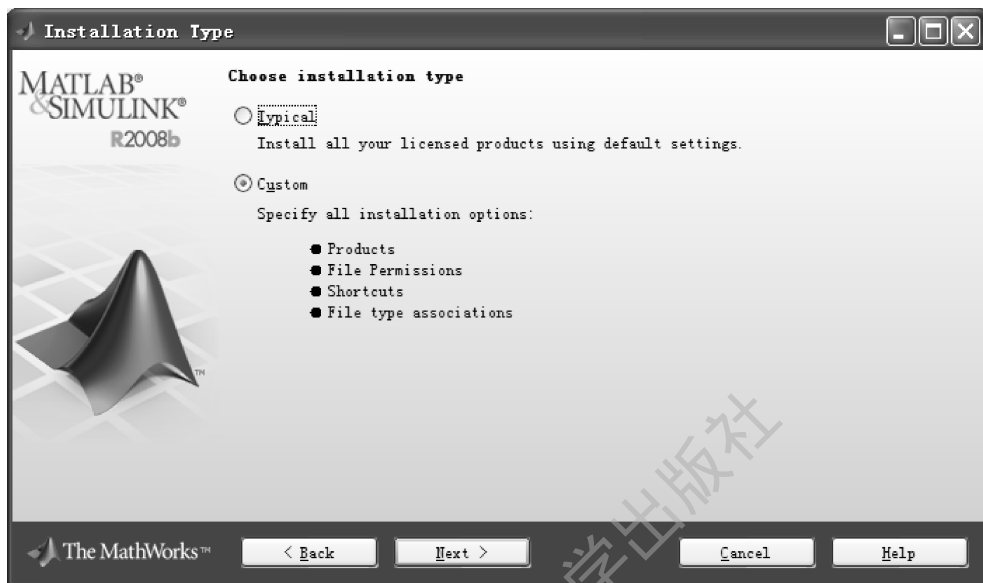


图 1.1-4 安装类型对话框

第 5 步：弹出目录选择对话框，如图 1.1-5 所示，选择 MATLAB 软件的安装文件夹。用户可以选择一个安装文件夹，如“D:\Program Files\MATLAB\R2008b\”。单击 Restore Default Folder 按钮，可以将安装文件夹重置为默认的安装文件夹。MATLAB 的默认安装文件夹为“C:\Program Files\MATLAB\R2008b\”。设置完安装文件夹，单击 Next 按钮继续

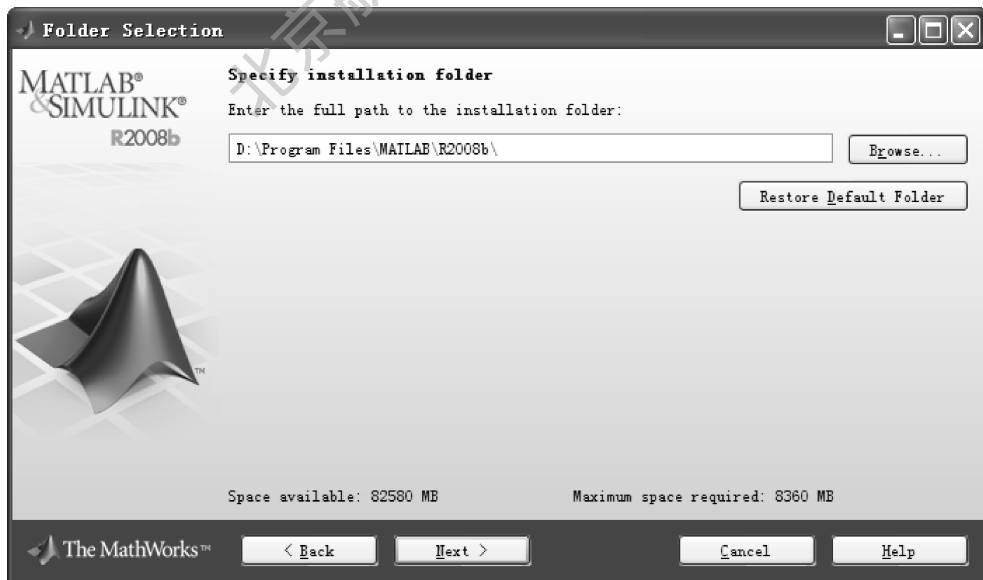


图 1.1-5 目录选择对话框

安装。

第 6 步:弹出产品选择(Product Selection)对话框,如图 1.1-6 所示。用户可以在列表框中选择要安装的组件,如 MATLAB、Simulink、Compiler 以及所需要的工具箱。

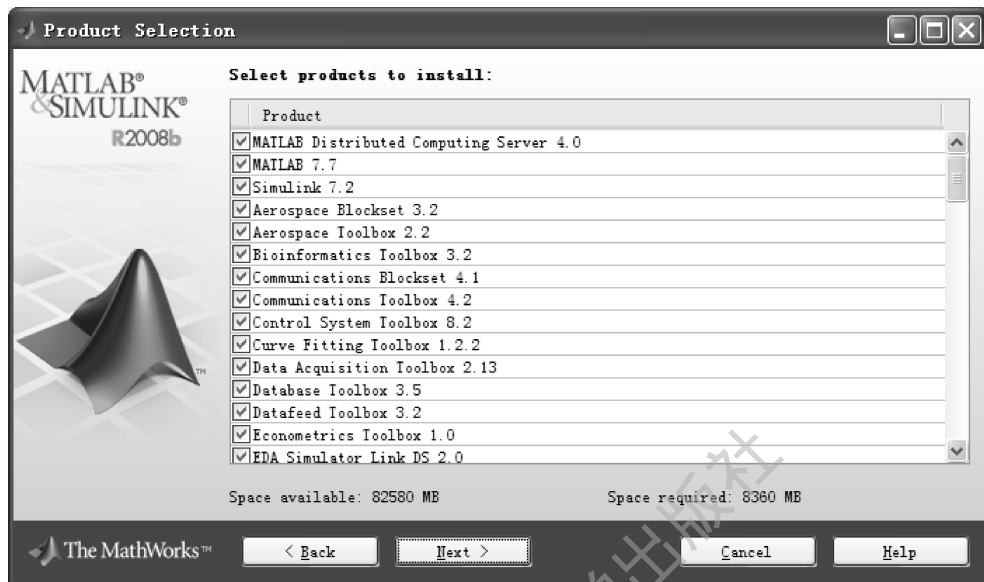


图 1.1-6 产品选择对话框

在对话框的下方给出了安装目录所在磁盘的可用空间(Space available)以及安装 MATLAB 系列软件所需的磁盘空间(Space required)。选择完要安装的产品后,单击 Next 按钮继续安装。

第 7 步:弹出安装选项(Installation Options)对话框,如图 1.1-7 所示。用户可以设置安装文件的属性为“只读”,也可以在桌面或开始菜单中添加 MATLAB 快捷方式,还可以将

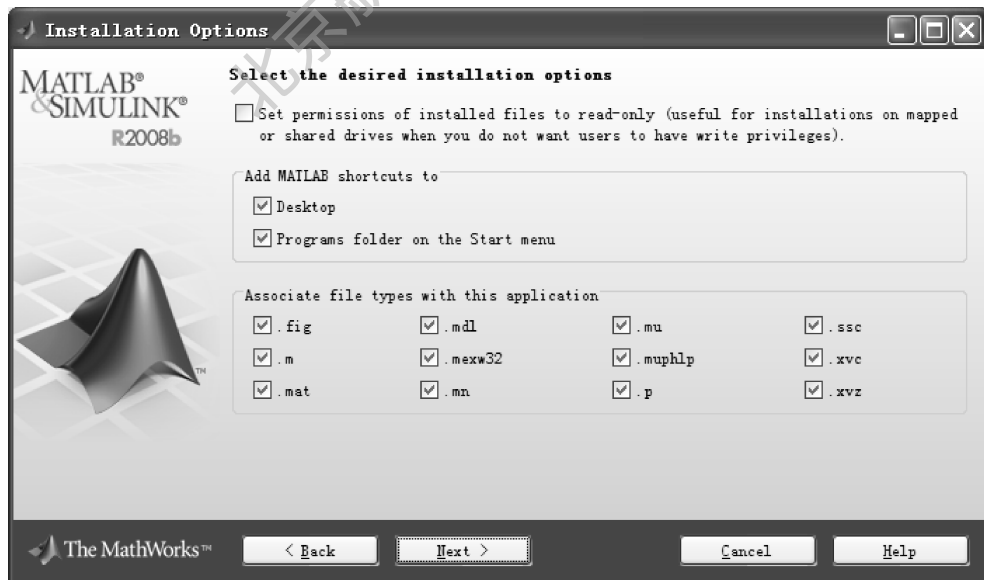


图 1.1-7 安装选项对话框

.fig、.m、.mat 等文件类型与 MATLAB 应用程序相联系。这些类型的文件在打开时默认是调用 MATLAB 程序打开的。

设置完成后,单击 Next 按钮继续安装。

第 8 步:弹出确认(Confirmation)对话框,确定用户的安装设置,如图 1.1-8 所示。如果列表中所列内容为用户预期的安装内容,则单击 Install 按钮开始安装软件;否则,用户可以单击 Back 按钮返回先前的安装步骤来重新设置。

在确认所有的安装信息都正确后,单击 Install 按钮开始安装。

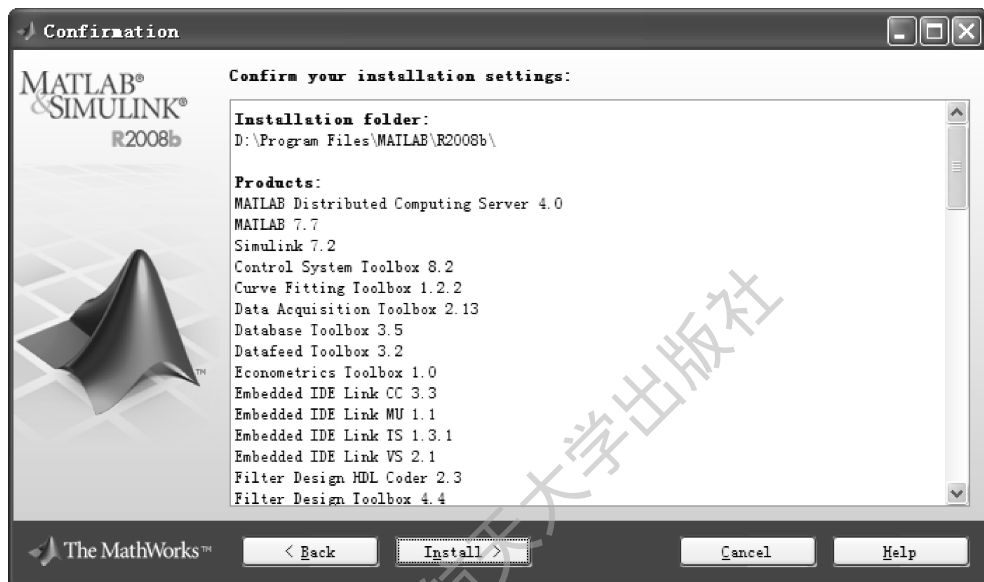


图 1.1-8 确认对话框

第 9 步:开始安装并弹出进度对话框,提示软件的安装进度,直至安装结束,如图 1.1-9 所示。



图 1.1-9 安装进度对话框

第 10 步：安装完成后，弹出产品配置提示对话框，如图 1.1 - 10 所示。单击 Next 按钮，继续安装。

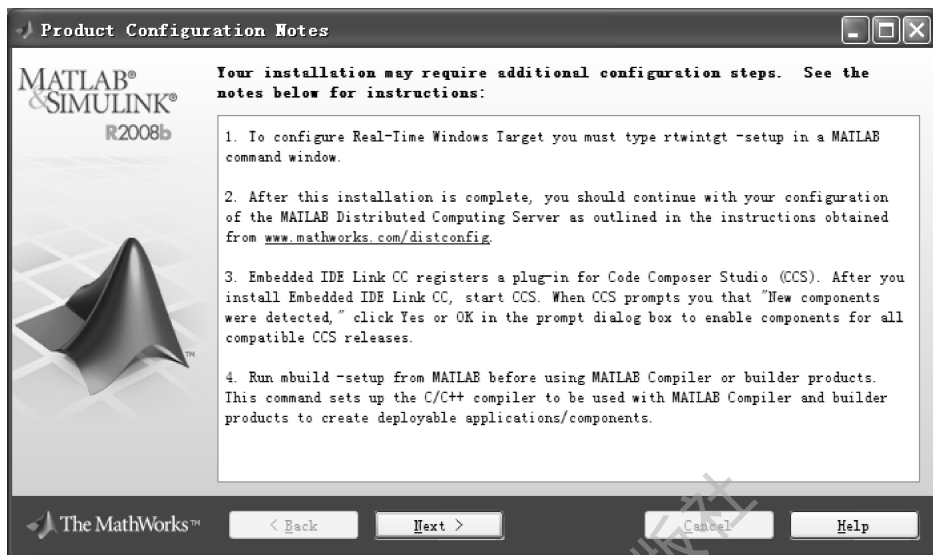


图 1.1 - 10 产品配置提示对话框

第 11 步：MATLAB 软件安装完成后，必须首先激活 MATLAB 软件后才能使用。因此在安装完成对话框中，如图 1.1 - 11 所示，用户需要选择 Activate MATLAB(激活 MATLAB)，然后单击 Next 按钮来激活 MATLAB 软件。



图 1.1 - 11 安装完成对话框

第 12 步：在软件激活对话框内，如图 1.1 - 12 所示，可以选择通过 Internet 来激活，也可以选择手工激活（如果用户的计算机内有软件的许可文件）。

如果用户手头有 Internet 链接，已经注册了 MathWorks 公司的账户，并且有随购买的软件而带的激活码，则建议用户选择“Activate automatically using the Internet (recommen-

ded) ”来自动激活软件。

如果用户手头没有 Internet 链接,但有随购买软件而带的授权许可文件,则可以选择 Activate manually without the Internet 来手工激活 MATLAB 安装程序。

这里选择手工激活方式,单击 Next 按钮继续激活程序。

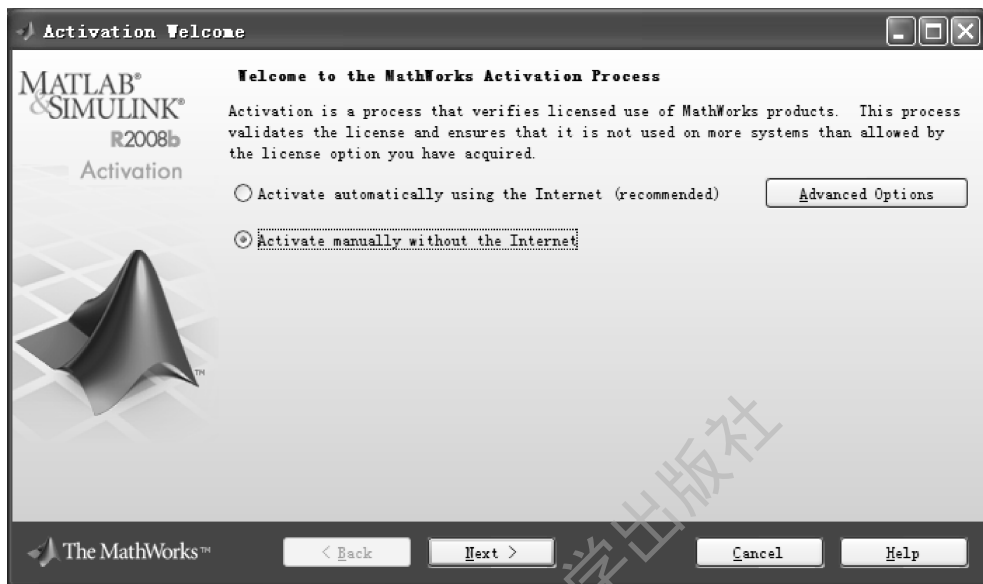


图 1.1 - 12 软件激活对话框

第 13 步:弹出离线激活对话框,如图 1.1 - 13 所示。选中 Enter the path to the license file 单选按钮,并单击 Browse 按钮来查找授权许可文件,然后单击 Next 按钮继续激活程序。

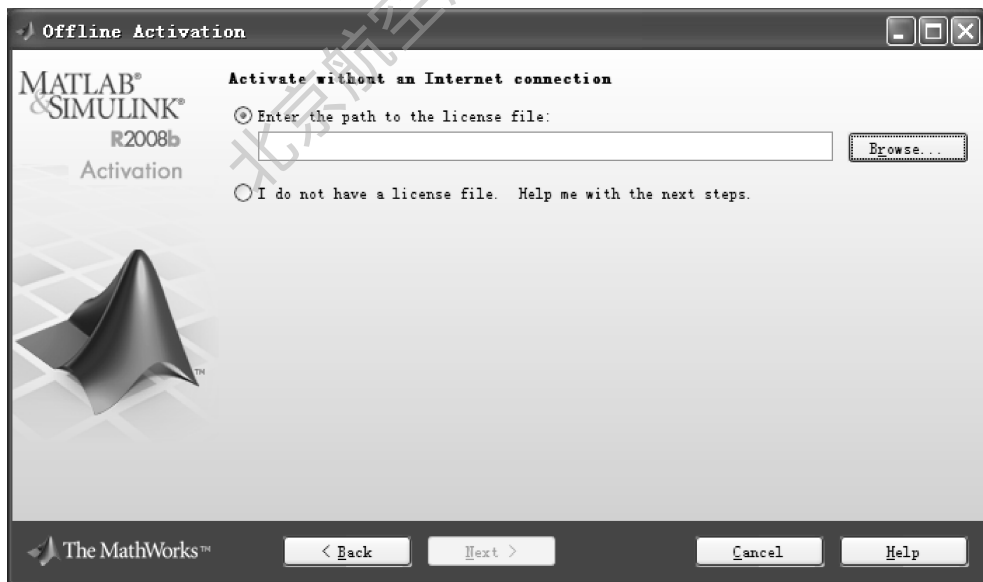


图 1.1 - 13 离线激活对话框

第 14 步:完成软件激活,选中图 1.1 - 14 中的 Start MATLAB 复选框,单击 Finish 按钮, MATLAB 软件即安装完成,同时启动 MATLAB 软件。



图 1.1 - 14 激活完成对话框

2. 其他问题

用户在安装完 MATLAB 软件后,可以调用 MATLAB 命令来获取软件安装的相关信息。

(1) 查看 MATLAB 软件的安装路径

用户可以调用 `matlabroot` 命令来取得 MATLAB 软件的安装路径。

```
% 调用 matlabroot 命令返回 MATLAB 的安装路径
```

```
>> rd = matlabroot
```

```
rd =
```

```
D:\Program Files\MATLAB\R2008b
```

(2) 利用 `matlabroot` 来表示 MATLAB 安装路径

```
% 调用 fullfile 来产生完整路径
```

```
>> fullfile(matlabroot,'toolbox','matlab','general')
```

```
ans =
```

```
D:\Program Files\MATLAB\R2008b\toolbox\matlab\general
```

(3) 将 MATLAB 的安装目录设置为当前目录

```
% 将当前目录切换到 MATLAB 安装目录上
```

```
cd(matlabroot);
```

(4) 在 MATLAB 搜索路径上添加文件夹

```
addpath([matlabroot '/toolbox/local/myfiles'])
```

1.2 技巧 2: MATLAB 的启动

1.2.1 技巧用途

MATLAB 软件由一系列工具组成,这些工具能方便用户使用 MATLAB 提供的函数和

文件。其中许多工具采用的是图形用户界面,包括 MATLAB 桌面和命令窗口、历史命令窗口、编辑器和调试器、路径搜索和用于用户浏览帮助文件的浏览器等。随着 MATLAB 的商业化以及软件本身的不断升级, MATLAB 的用户界面也越来越精致,更加接近 Windows 的标准界面,人机交互性更强,操作更简单。

但是 MATLAB 软件本身含有很多工具箱,而且在运行过程中占用的内存量也很可观。为了节省资源,可以适当地屏蔽掉一些功能,使其不随 MATLAB 一起启动,从而达到节约资源、加速启动的目的。此外,用户也可以定义一些默认设置(如默认的窗口颜色、字体大小等),在 MATLAB 启动时自动加载这些设置,免去用户每次都需要手动设置的麻烦。

1.2.2 技巧实现

1. 启动时加载用户自定义设置

当 MATLAB 启动时,会自动执行 matlabrc.m 文件。matlabrc.m 文件一般在{matlab 根目录}\toolbox\local 目录下,用户可以在命令窗口中输入“which matlabrc.m”来查看其所在的目录。

```
>> which matlabrc.m
D:\Program Files\MATLAB\R2008a\toolbox\local\matlabrc.m
```

如果 MATLAB 检测到在当前目录下存在 startup.m 文件,则 MATLAB 会自动调用、运行该文件。startup.m 是用户自定义的命令文件,文件中的代码由用户根据需要自己添加。在这个文件中,用户可以添加自己预先定义的内容和设置。

打开 matlabrc.m 文件,可以看到如下的语句,用来检测 startup.m 文件是否存在。如果该文件存在,则调用该文件运行。

```
% 如果 startup.m 文件存在,则执行该文件
if (exist('startup','file') == 2) || ...
    (exist('startup','file') == 6)
    startup
end
```

【例 1.2-1】 启动 MATLAB,显示欢迎信息,去除 figure 窗口上默认的工具条。

创建 M 文件,命名为 startup.m,在其中添加下列语句,并保存到 MATLAB 工作目录中(如 D:\work)。

```
% 显示欢迎信息
disp('Welcome to MATLAB! ');
% 关闭 figure 窗口上默认的 MATLAB 标准工具条
set(0,'defaultfiguretoolbar','none')
```

启动 MATLAB,则在命令窗口中显示“Welcome to MATLAB!”信息,如图 1.2-1 所示。再输入 gcf 命令,可见弹出的 figure 窗口中不带 MATLAB 标准工具条。

2. MATLAB 的快速启动

MATLAB 的快速启动方法有如下 3 种:

(1) 在运行命令框中输入启动命令

单击计算机任务栏中的“开始”→“运行”,弹出“运行”对话框,在其中输入“matlab.exe -

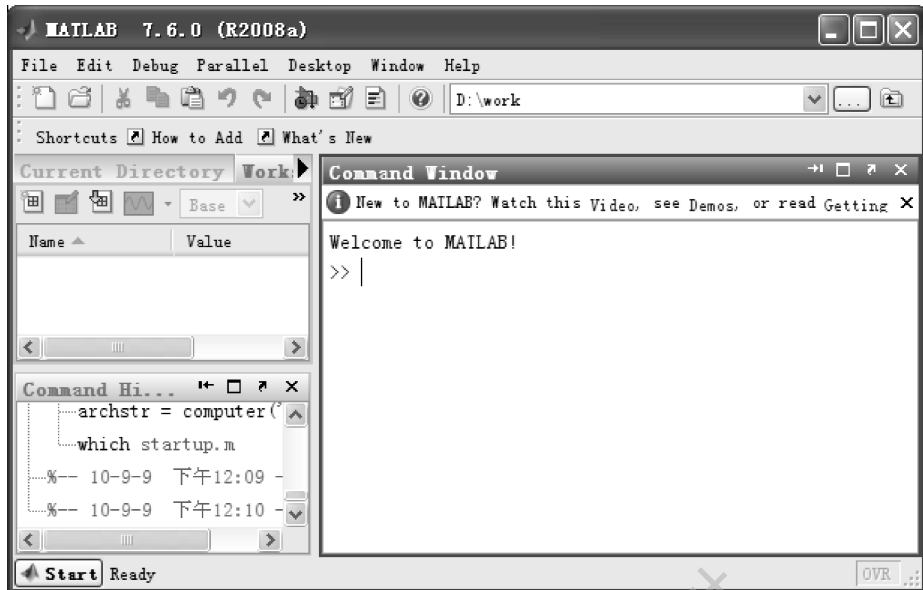


图 1.2-1 启动界面

nojvm”,如图 1.2-2 所示。在启动 MATLAB 时将禁用 java 虚拟机,启动后的 MATLAB 界面如图 1.2-3 所示。



图 1.2-2 MATLAB 快速启动命令输入对话框

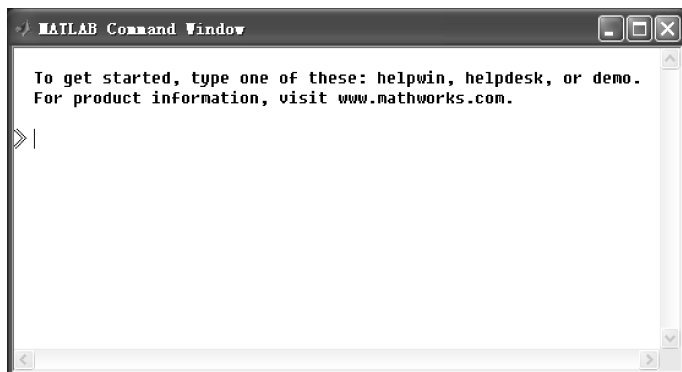


图 1.2-3 不启动 JVM 的 MATLAB 界面

(2) 在 DOS 窗口中输入启动命令

单击计算机任务栏中的“开始”→“运行”，弹出“运行”对话框。在其中输入 cmd 后进入 MS-DOS 窗口，然后输入命令“matlab.exe -nojvm”，如图 1.2-4 所示。启动 MATLAB 时将不启用 Java 虚拟机。

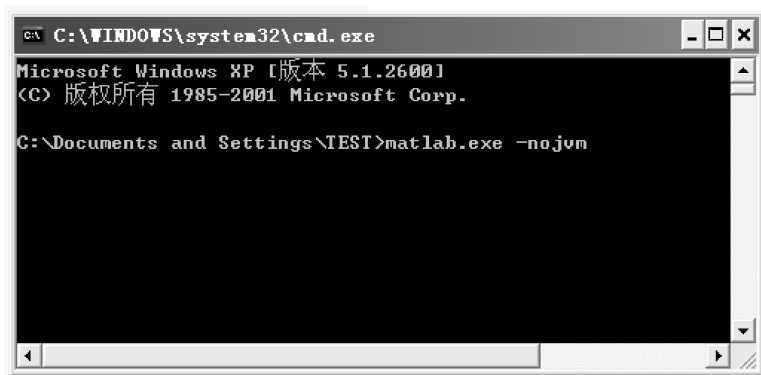


图 1.2-4 MATLAB 快速启动 DOS 命令输入界面

(3) 在 MATLAB 软件的快捷方式中加入启动命令

- 右击 MATLAB 的快捷方式，在快捷菜单中选择“属性”；
- 在“快捷方式”标签页中的“目标”栏中添加 -nojvm；
- 单击“确定”或“应用”按钮。

如图 1.2-5 所示。



图 1.2-5 MATLAB 快速启动快捷方式设置对话框

1.2.3 技巧扩展

如果用户只是用到 MATLAB 的部分工具箱,则可以通过修改 `pathdef.m` 文件将用不到的工具箱注释掉。这样既可加速启动,又可缩短运行 M 脚本时自动搜索文件所用的时间,切实提高 MATLAB 效率,降低内存消耗,减少处理海量数据时产生的内存溢出等问题。

1.3 技巧 3: 内存的优化配置

1.3.1 技巧用途

在使用 MATLAB 处理大量数据、图像时, MATLAB 有时会给出 Out of Memory 的提示信息。因为系统分配给 MATLAB 的内存已经使用完毕,用户在处理过程中的中间变量和结果没有内存位置可以暂存和存储,这时 MATLAB 便会给出这类提示。究其原因,主要包括两点:一是用户所编写的程序算法本身的弊病,如有时在设计算法时,为了提高效率,可能会通过牺牲内存来实现,如果系统内存较小,这一类算法编写的程序很容易出现内存不足的情况;另外一点就是系统配置方面的问题,如果系统配置不佳,则可以通过增加算法的运算时间来减少内存开销,但有时即使这样也不能收到较好的效果,需要用户再进行额外的操作。

本节将介绍用户如何手工配置内存,以提高 MATLAB 软件的运行速度和效率。

1.3.2 技巧实现

1. memory 函数以及内存配置方法

memory 函数用于显示内存信息,只对 Windows 系统有效。其调用格式如下:

- **memory**
- **userview = memory**
- **[userview, sysview] = memory**

其中,直接使用 memory 函数,返回如下 4 项内容:

- ① 最大矩阵大小;
- ② 所有矩阵的最大内存消耗;
- ③ MATLAB 进程所用的内存数;
- ④ 物理内存总数。

上述第 2 种调用方法只给出以上 4 项中的前 3 项内容;第 3 种调用方式,将用户信息和系统信息分开给出,分别为:

- ① 用户信息:以上 4 项中的前 3 项。
- ② 系统信息:虚拟地址空间,系统内存和物理内存。

在这里,我们只关心如下两项内容:

- ① 最大矩阵大小;
- ② 所有矩阵的最大内存消耗。

它们直接关系到用户在使用 MATLAB 时所能创建的单个矩阵大小的最大值或者用户所有数据所能拥有的最大内存数量。

如果在命令窗口输入 memory,将会出现如下运行结果(针对 32 位系统):

```
Maximum possible array:          600 MB (6.290e+ 008 bytes) *
Memory available for all arrays:  1394 MB (1.461e+ 009 bytes) * *
Memory used by MATLAB:           403 MB (4.230e+ 008 bytes)
Physical Memory (RAM):           1978 MB (2.074e+ 009 bytes)
```

* Limited by contiguous virtual address space available.

** Limited by virtual address space available.

说明系统物理内存还有足够的空间可供 MATLAB 进程虚拟内存地址映射使用。增加 MATLAB 进程虚拟内存,可以有效地增大 MATLAB 所能建立的最大矩阵或者 MATLAB 工作空间中同时存在的数据的总大小。这样便可以有效地解决在处理过大的数据时出现 Out of Memory 的情况。那么,如何增加 MATLAB 进程所拥有的虚拟内存数呢? 这里列举以下常用方法供读者选用:

(1) 禁止 Java 虚拟机启动

在 1.2 节中介绍过,MATLAB 启动时,部分虚拟内存地址被保留起来用于 Java 虚拟机使用。这部分地址不能用来保存数据,因此会减少数据能够使用的内存空间。如果不进行调试或者一些界面的编程工作,可以禁止 Java 虚拟机启动,这样可以有效地节省一部分内存空间。

在桌面 MATLAB 快捷方式上右击,选择“属性”。在弹出的 MATLAB 属性对话框中,将“目标”编辑框中的内容修改为:

```
"MATLABROOT\bin\matlab.exe" -nojvm
```

还有其他的禁止 Java 虚拟机自启动方法,在 1.2 节已经有很详细的讲解,这里不再赘述。

(2) 增加虚拟内存数量

右击桌面上的“我的电脑”,在弹出的菜单中选择“属性”。在“系统属性”对话框中选择“高级”选项卡,在其中的“性能”区域单击“设置”按钮,出现“性能选项”对话框。选择其中的“高级”选项卡,在“虚拟内存”区域单击“更改”按钮即可更改系统的虚拟内存数量,如图 1.3-1 和图 1.3-2 所示。

(3) 打开系统 3 GB 开关(对于 32 位系统)

操作系统根据许多不同的规则和设置来管理系统资源,以最大限度地提高资源利用率,并确保所有应用/进程都能平等地访问这些资源。例如,操作系统可以控制单个应用/进程所占用的总内存量。又例如,在 Windows XP 系统中,总内存的默认值设置为 2 GB。如果某个应用/进程占用的总内存超过 2 GB,那么操作系统将会立即终止该应用/进程。

由于总内存只是一种系统设置,因此用户可对其进行修改。在 Windows XP 中,用户可以“反转”3 GB 切换开关,将某个应用/进程可用的内存总量增加到 3 GB。3 GB 切换开关对于内存密集型程序特别有用。

① 右击桌面上“我的电脑”,在弹出的菜单中选择“属性”,将出现“系统属性”对话框,如图 1.3-1 所示。

② 单击“高级”选项卡,在“启动和故障恢复”区域中单击“设置”,将出现“启动和故障恢复”对话框,如图 1.3-3 所示。

③ 在“系统启动”区域中,单击“编辑”按钮,在记事本中打开 Windows 系统的 boot.ini 文件。

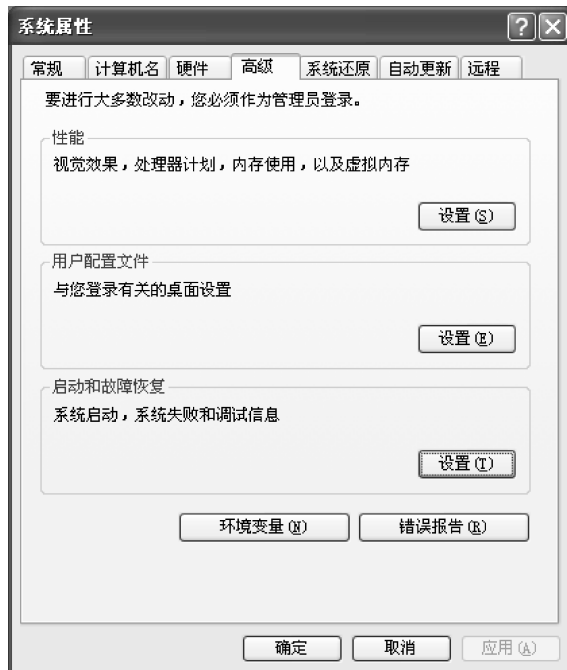


图 1.3-1 “系统属性”对话框

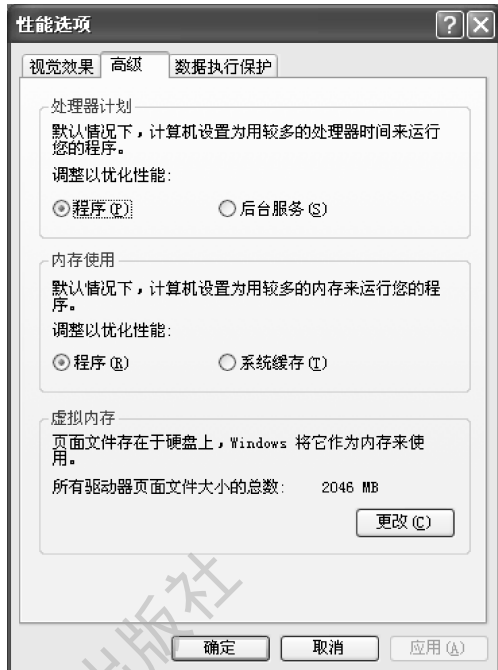


图 1.3-2 “性能选项”对话框



图 1.3-3 “启动和故障恢复”对话框

在 [Operating Systems] 部分中, 将以下开关添加到包含 /fastdetect 开关的启动命令行结尾: /3 GB。boot.ini 的内容如下:


```
[boot loader]
timeout = 5
default = multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS = "Microsoft Windows XP Professional" /noexecute =
optin /fastdetect/3GB
C:\gglldr.mbr = GGhost
```

④ 保存更改, 关闭记事本, 单击“确定”按钮关闭打开的对话框, 然后重新启动计算机以使更改生效。

2. pack 函数以及 MATLAB 的内存整理

pack 函数可以合并 MATLAB 新建变量时开辟的内存空间, 避免因存储空间的断续造成连续的内存地址不足, 以存储比较大的矩阵, 从而引起 Out of Memory 的情况。

pack 函数的调用格式如下:

- **pack**
- **pack filename**
- **pack('filename')**

其中, pack('filename') 的使用方式为 pack filename 命令方式的函数用法。直接使用 pack 命令, MATLAB 将自动整理工作空间中的变量。其整理方式如下:

- ① 保存 base 空间以及全局变量空间中驻留的变量至一个 .mat 文件;
- ② 清空工作空间中的所有变量;
- ③ 重新载入所保存 .mat 文件中的 base 空间以及全局变量空间中驻留的变量, 并删除所保存的 .mat 文件。

如果使用 filename 参数, 那么 MATLAB 将工作空间中的变量保存到 filename.mat 文件中。该文件保存在当前工作路径中。

使用 pack 函数可以有效地整理 MATLAB 分配的断续内存空间, 为比较大的变量腾出连续内存地址, 但是并不能提高 MATLAB 所拥有的内存数目。也就是说, 使用 pack 函数之后, 再次使用 memory 函数, 所给出的最大矩阵大小以及所有变量的最大内存消耗并不能增加。

3. 解决 Out of Memory 的其他方式

除了上面介绍的通过设置系统虚拟内存或者使用 pack 函数整理内存之外, 编程时如果能够注意一些事项, 也可以有效地提高内存使用率, 避免 Out of Memory 的出现。具体来讲, 下面几种方法在编程中会经常用到:

- ① 不用的变量及时清除掉, 避免占用内存。
- ② 允许条件下, 可以重复使用同一个内存空间, 也就是说, 对不用的变量重新赋值, 作为新的变量来用。
- ③ 循环体中需要赋值的变量, 在循环之前预分配空间, 这样不但可以给变量分配连续的内存, 还可以有效地提高循环速度。
- ④ 牺牲效率, 减少内存消耗。在很多情况下, 一些算法提高计算效率的代价是内存的消耗, 如果内存资源不足的话, 应尽量避免这一类算法的使用。

1.4 技巧 4：工具箱的添加

1.4.1 技巧用途

MATLAB 工具箱是 MATLAB 功能的进一步扩展,是 Mathworks 公司与第三方在 MATLAB 主程序包提供的强大的数值运算的基础上,针对具体的工程问题提供的特殊的函数集合。使用特定的工具箱可以帮助用户解决特殊的工程问题,使用户可以方便、快捷地使用复杂的计算公式,避免了用户自己编写复杂的算法程序。

MATLAB 工具箱实际上就是基于 MATLAB 软件而开发的一组实用的函数 M 文件或 Simulink 仿真模型。随着 MATLAB 功能的不断扩展,工具箱将会越来越多。因此,用户很有必要了解如何向 MATLAB 中添加新的工具箱。

1.4.2 技巧实现

对于新的工具箱的添加,分为以下几种情况:

1. 添加 MATLAB 安装软件中所带的工具箱

如果工具箱文件存在于 MATLAB 安装软件中,用户只需要重新执行安装程序,并勾选需要安装的工具箱,重新安装即可。以 MATLAB R2008a 为例,重新启动 MATLAB 安装程序,在 Product Selection 对话框中,重新勾选“Aerospace Blockset 3.1”和“Aerospace Toolbox 2.1”工具箱,然后单击 Next 按钮重新添加即可,如图 1.4-1 所示。

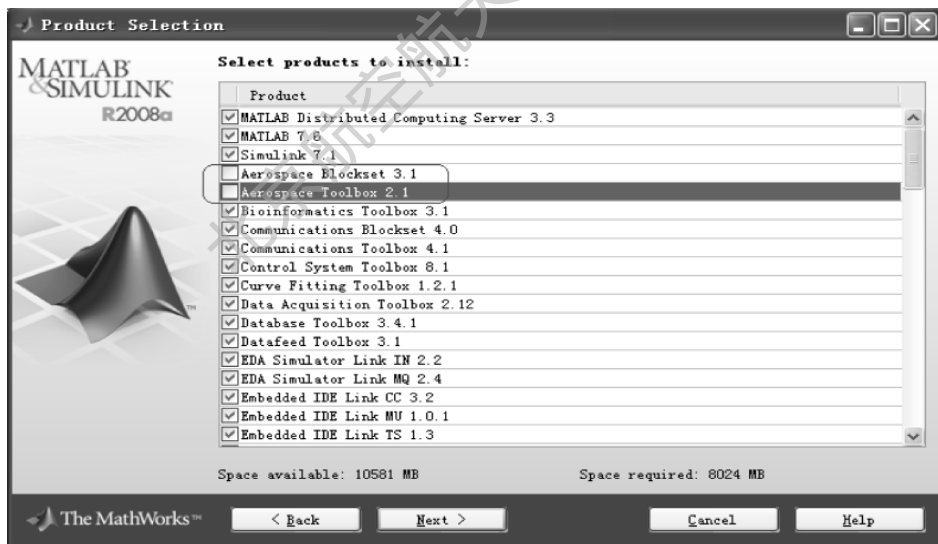


图 1.4-1 重新选中工具箱并安装

2. 添加单独下载的工具箱

用户可以通过网络下载自己需要的 MATLAB 工具箱,然后把这些工具箱添加到本机的 MATLAB 环境中,就可以利用这些工具箱提供的函数来解决自己的问题了。

以下以从网站上下载的 MATLAB 的 dfp(digital filter package)工具箱为例来说明工具箱的安装步骤:

(1) 将 dfp 工具箱复制到本地计算机中的某个目录中

例如:将 dfp 工具箱复制到 MATLAB 安装文件夹下的目录“D:\Program Files\MATLAB\R2008a\toolbox\”中。

【注】要添加的 dfp 工具箱包含一个文件夹“dfp”,其中包含有子文件夹 codgen、doc、uitools 等,同时还包含很多 M 文件 arrow.m、d_append.m、d_codgen.m 等,用户需要把整个文件夹 dfp 都复制到 D:\Program Files\MATLAB\R2008a\toolbox\目录下。

(2) 将工具箱所在的文件夹及其子文件夹添加到 MATLAB 的搜索路径中

可以使用命令行方式和图形用户界面方式,将工具箱文件夹添加到 MATLAB 的搜索路径中。

① 命令行方式。首先用户需要调用 genpath 函数来取得包含工具箱所在的文件夹及其子文件夹名称的字符串,然后调用 addpath 命令来添加。例如:

```
% 取得工具箱所在的完整的路径
>> rd = toolboxdir('dfp')
rd =
d:\program files\matlab\r2008a\toolbox\dfp
% 产生包含工具箱所在的文件夹及其子文件夹的名称的字符串
>> p = genpath(rd);
% 将工具箱所在的文件夹及其子文件夹添加到 MATLAB 的搜索路径中
>> addpath(p)
```

如果要添加的工具箱中存在与 MATLAB 已有文件同名的文件,则 MATLAB 会给出如下警告信息:

```
Warning: Function d:\program files\matlab\r2008a\toolbox\dfp\uitools\ishandle.m has the same
name as a MATLAB builtin. We suggest you rename the function to avoid a potential name conflict.
```

用户可以修改工具箱中的 M 文件名称,以避免重名的情况出现。

② 图形用户界面方式。在 MATLAB 主窗口选择 File→Set Path 菜单项,弹出图 1.4-2 所示的对话框。



图 1.4-2 设置搜索路径

单击 Add with Subfolder 按钮,弹出“浏览文件夹”对话框,找到工具箱所在的文件夹,单击“确定”按钮,则工具箱所在的文件夹及子文件夹出现在 MATLAB search path 的最上端。单击 Save 按钮保存搜索路径的设置,然后单击 Close 按钮关闭对话框。

(3) 更新工具箱路径的缓存和缓存文件

可以采用如下两种方式来更新缓存和缓存文件:

① 调用命令 rehash。

```
rehash toolboxcache
```

② 选择 File→Preferences 菜单项,弹出 Preferences 对话框,如图 1.4-3 所示。选中左侧的 General 选项,勾选右侧的 Enable toolbox path cache 复选框,单击 Update Toolbox Path Caching 按钮,将工具箱路径的缓存更新。

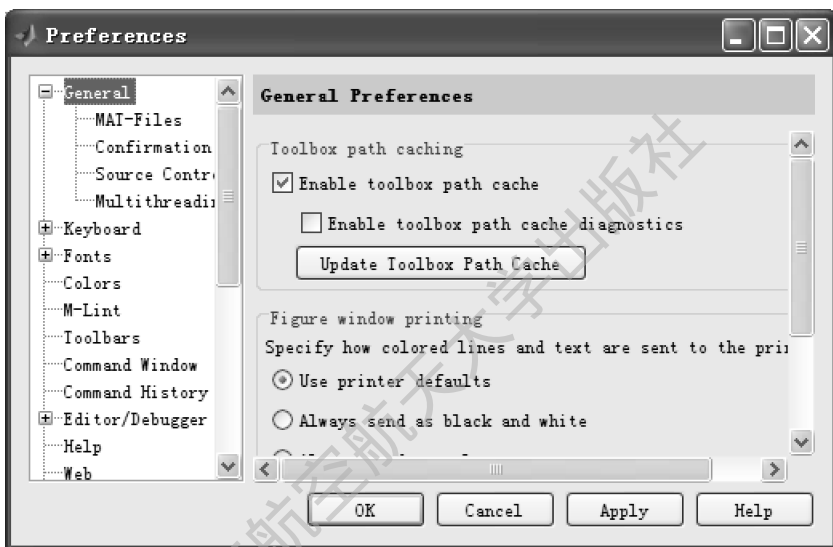


图 1.4-3 更新工具箱路径缓存

(4) 验证工具箱是否安装成功

通过调用 which 命令来查看工具箱是否安装成功。which 命令能定位文件或函数,如果 which 命令能返回工具箱中的某一 M 文件的完整路径,则表明工具箱安装成功,该工具箱就可以使用了。

```
% 定位 dfp 工具箱中的 d_disply 文件
>> which d_disply.m
D:\Program Files\MATLAB\R2008a\toolbox\dfp\d_disply.m
```

1.5 技巧 5: 中文字体的设置与显示

1.5.1 技巧用途

MATLAB 中有很多窗口,像命令行窗口、历史命令窗口、工作空间浏览器、当前目录浏览器等窗口。每个窗口都可以设置特定的字体,所以用户可以对每个窗口自定义自己喜欢的字

体。因为 MATLAB 各窗口默认的字体是与系统有关系的,而系统的字体跟当前设定的系统主题有关系,所以当字体设置不当或者是修改系统主题时,MATLAB 某些窗口可能会出现中文乱码,这时可以通过修改对应窗口的字体,或者更改系统主题来解决。

1.5.2 技巧实现

1. 字体的设置

以 Windows XP 系统下的 MATLAB R2008a (MATLAB 7.6.0.324) 为例来说明。MATLAB 字体的设置是在 Preferences(首选项)对话框中进行的。打开 Preferences 对话框的方法有多种:

- ① 选择菜单 File→Preferences;
- ② 在 Command Window 窗口中输入命令 Preferences 并回车,弹出首选项对话框;
- ③ 直接单击 MATLAB 工具栏中的 Preferences 图标。如果图标不存在,在 MATLAB 工具栏上右击任意图标,弹出右键菜单,选择其中的 Customize 项,再选中 Controls 框里 Preferences 前面的复选框,回到 MATLAB 主界面就能看到 Preferences 图标出现在工具条中了,如图 1.5-1 所示。

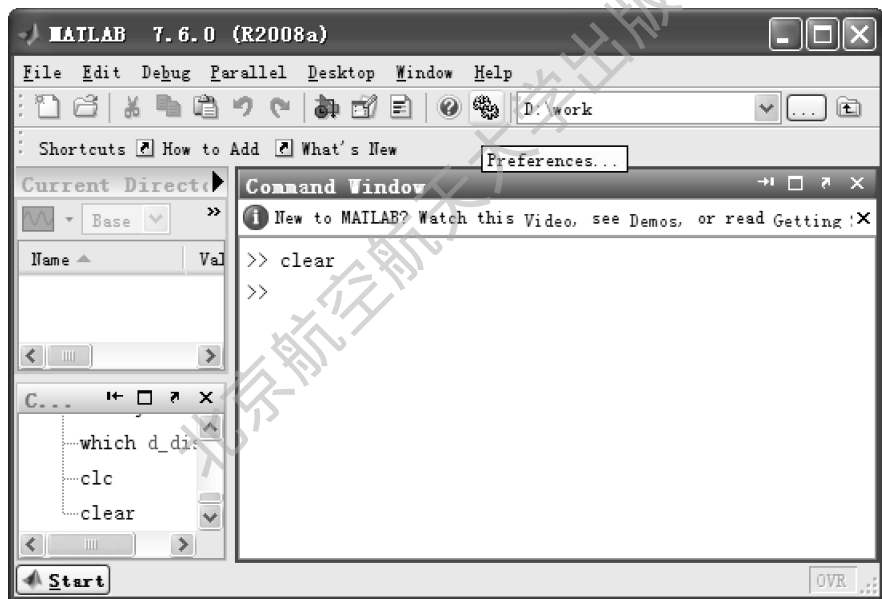


图 1.5-1 Preferences 图标

以上 3 种方法都可以打开 Preferences 对话框。选择对话框左边列表框中的 Fonts 选项,如图 1.5-2 所示,在该对话框中可以对 MATLAB 各窗口字体进行设置。

如图 1.5-2 所示,Fonts 的 Preferences 对话框中有两个选项:Desktop code font(桌面代码字体)、Desktop text font(桌面文本字体)。

① 桌面代码字体是用来设置 Command Window(命令行窗口)、Command History(历史命令窗口)和 Editor(编辑器)、Current Directory(当前目录)、Workspace(工作空间)等字体的,由于各窗口内容都是代码,所以称之为桌面代码字体。窗口代码字体的默认设置是:Monospaced,Plain,10。

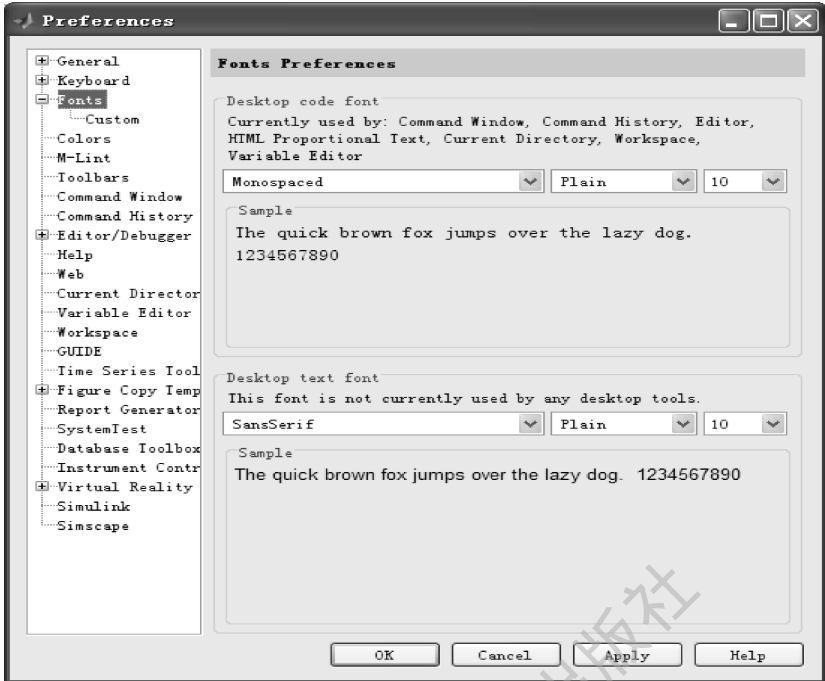


图 1.5-2 Fonts Preferences 设置

其中,字体下拉列表框里显示的字体来自操作系统的字体库,如果一个字体存在但是 MATLAB 却不能显示它,那么 MATLAB 会将其从列表框中删掉。字形有 4 种,分别是 Plain(常规)、Bold(加粗)、Italic(倾斜)、Bold Italic(加粗、倾斜);字号有 8、9、10、12 等几种选择,但是可以直接输入需要的其他大小的字号。

② 桌面文本字体用来设置任何桌面工具当前所用的字体。窗口文本的默认设置是使用系统字体,它是与系统字体有关系的。窗口文本字体默认设置是:SansSerif,Plain,10。

此外,用户单击左侧树形结构中的 Fonts 下的 Custom,可以分别对各个桌面工具进行自定义字体设置,分别定义字体、字形和字号,如图 1.5-3 所示。

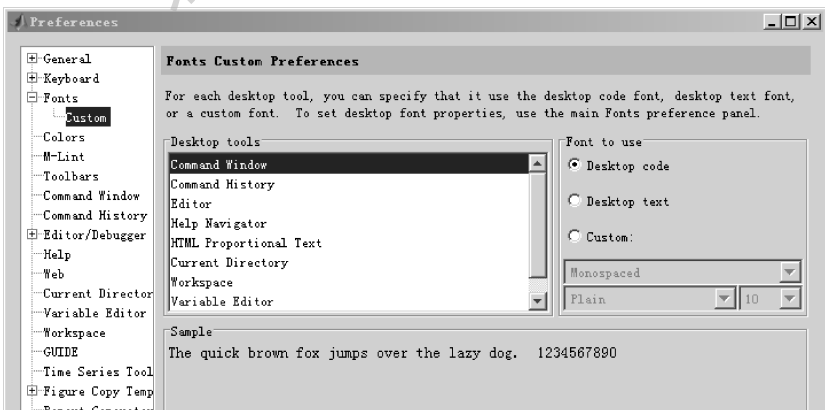


图 1.5-3 自定义字体对话框

从图 1.5-3 中的 Desktop tools 列表框中选择一个窗口,这个窗口当前使用的字体类型就会显示在 Font to use 下方。改变被选窗口的字体类型,可以在 Font to use 里选择 Desktop

code 或者是 Desktop text,也可以选择 Custom,然后指定字体类型。

【例 1.5-1】 改变窗口代码字体的默认设置,将 Command History 的字体使用窗口文本字体取代默认的窗口代码字体,且为 Current Directory 浏览器指定一个自定义字体。

具体修改如下:

① 改变窗口代码字体。在 Fonts Preferences 面板上,将 Desktop code font 设置成 Times New Roman, Plain, 14。

② 窗口文本字体使用系统默认字体。在 Fonts Preferences 面板上的 Desktop text font 下选择 Use system font。

③ 单击 Apply。

④ 设置 Command History 窗口使用 Desktop text 字体:

- a. 单击 Custom Fonts 链接;
- b. 从 Desktop tools 中选择 Command History;
- c. 选择 Desktop text 单选框。

⑤ 为 Current Directory 浏览器应用自定义字体:

- a. 从 Desktop tools 中选择 Current Directory;
- b. 选择自定义单选框;
- c. 选择 Arial Narrow, Plain, 11;
- d. 单击 OK。

表 1.5-1 详细列出了字体改变的结果。

表 1.5-1 字体改变结果

工具窗口	字体类型	字体特征
Command Window	Desktop code	Monotype 改成 Times New Roman font, Plain, 14 point
Command History	Desktop text	与当前系统字体相同,当前系统的字体显示在 Use system font 复选框下面的灰色区域
Editor	Desktop code	Times New Roman 字体, Plain, 14 point
Help Navigator	Desktop text	与当前系统字体相同,当前系统的字体显示在 Use system font 复选框下面的灰色区域
HTML Proportional Text	Custom	SansSerif 字体, Plain, 10 point
Current Directory browser	Custom	Narrow font, Plain, 11 point
Workspace browser	Desktop text	与当前系统字体相同,当前系统的字体显示在 Use system font 复选框下面的灰色区域
Variable Editor	Desktop text	与当前系统字体相同,当前系统的字体显示在 Use system font 复选框下面的灰色区域

表 1.5-2 列出了出厂默认字体,以及使用对应字体的窗口。

如果想回到出厂设置,将窗口代码字体和文本字体按照表格中的出厂参数设置即可。

2. 中文字体设置和显示

以下以 Command Window 窗口字体显示为例来说明中文字体设置和显示。

表 1.5-2 字体出厂设置

字 体	出厂默认字体和举例	使用出厂默认字体的窗口
Desktop code font	Monospaced, Plain, 10 pointSample text font	* Command History * Command Window * Editor(也适用于 the Shortcuts Editor)
Desktop text font	Your system's current font	* Current Directory browser (也适用于路径浏览器) * 在命令窗口和编辑器中的 Function Browser * Help Navigator * Workspace browser * Variable Editor

在默认字体 Desktop code 情况下(见图 1.5-3),中文字体可以正常显示。默认字体是: Monospaced,Plain, 10。

当字体改成 Custom 时,如图 1.5-4 所示,这时的字体是:Tahoma,plain,8。Command Window 窗口中文显示乱码。

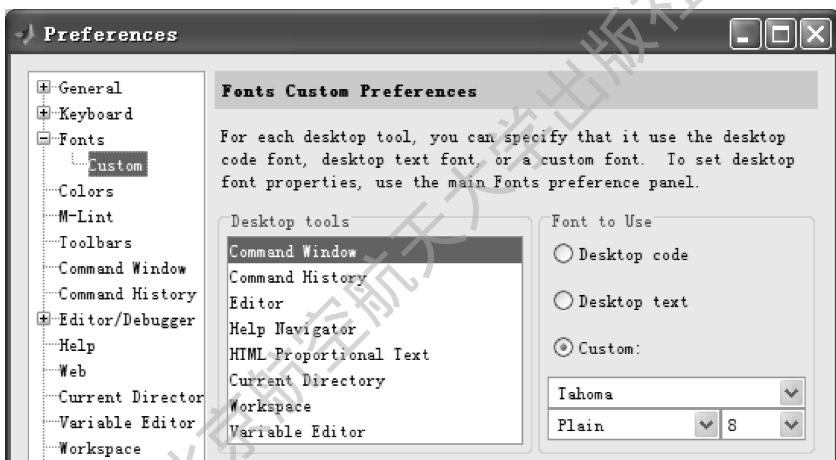


图 1.5-4 命令窗口字体设置成 Desktop text

当自定义成宋体、楷体、隶书等中文字体时,Command Window 窗口中的中文显示正常;当自定义成其他字体时就会出现中文乱码。

其他窗口的中文字体的设置与显示均与 Command Window 窗口类似。

【注】当将系统主题设为某些第 3 方主题时,Current Directory 窗口路径中的中文会显示乱码,这时将主题还原成系统默认主题或者是经典主题即可。

1.6 技巧 6: 工作路径的设置与修改

1.6.1 技巧用途

MATLAB 系统包含了数目繁多的文件,另外随着用户程序的开发,又会增加很多的用户程序文件。如何对这些文件进行合理的组织和管理,对于 MATLAB 编程环境非常重要。MATLAB 系统文件有严谨的目录结构,对不同的文件使用不同的存放位置,通过搜索路径来

实现文件的搜寻功能。

用户也有很多的文件需要分类管理,那么,是否可以将用户自定义目录加入 MATLAB 的搜索路径中?此外,用户希望将自己习惯使用的目录作为 MATLAB 启动后默认的目录,也就是 MATLAB 的启动目录,这就涉及启动目录的设置。MATLAB 安装后,即自动设置了默认的启动目录,使用 MATLAB 的默认启动目录,会比用户定义其他目录作为启动目录有更多便利之处。

1.6.2 技巧实现

1. 当前目录和搜索路径设置技巧

(1) 当前目录

当前目录是指 MATLAB 运行时的工作目录,它是一个相对位置。图 1.6-1 是一个浮动的当前目录窗口,它可以嵌入到 MATLAB 的主窗口上。另外还有一个地址栏形式的当前目录显示,如图 1.6-2 所示。

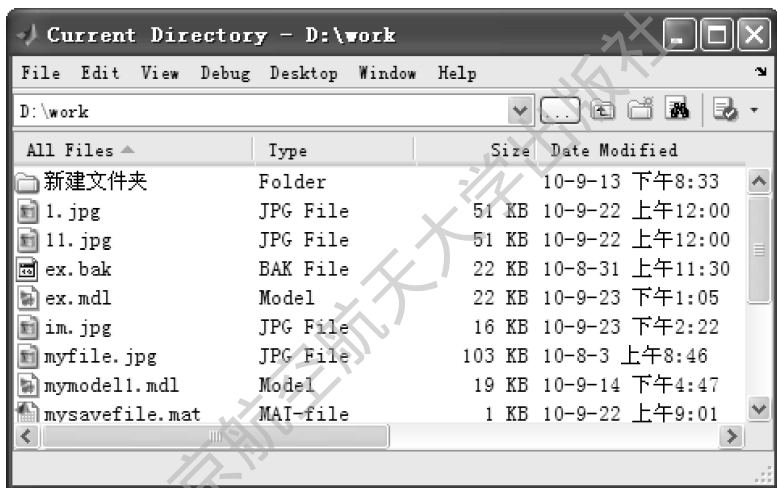



图 1.6-1 浮动的当前目录窗口



图 1.6-2 地址栏形式显示的当前目录

MATLAB 中,在当前目录下的文件、函数可以直接被 MATLAB 运行或者调用。当前目录的设置方法如下:

- ① 通过图 1.6-2 中工具栏上的  来设置。
- ② 使用 cd 函数来设置。其中,cd 函数的使用方法为:

```
% 返回当前目录
cd;
% 设置当前目录
cd('d:\Program Files\MATLAB\R2009a\work');
% 将当前目录的上一层目录设置为当前目录
cd('..');
```

(2) 搜索路径

MATLAB 通过搜索路径方便地组织和管理 MATLAB 文件。如果想调用不在 MATLAB 当前目录下的文件,则需要将这些文件保存到 MATLAB 的搜索路径下,这样 MATLAB 就可以运行或者调用这些文件了。

可以通过 path 函数或者使用 Set Path 对话框查看 MATLAB 的搜索路径内容:

```
% 返回搜索路径内容
>> path

MATLABPATH

d:\program files\matlab\r2008a\toolbox\dfp
d:\program files\matlab\r2008a\toolbox\dfp\codegen
d:\program files\matlab\r2008a\toolbox\dfp\codegen\mot56k
.....
% 从 Set Path 对话框中查看搜索路径内容
>> pathtool;
```

Set Path 对话框如图 1.6-3 所示。

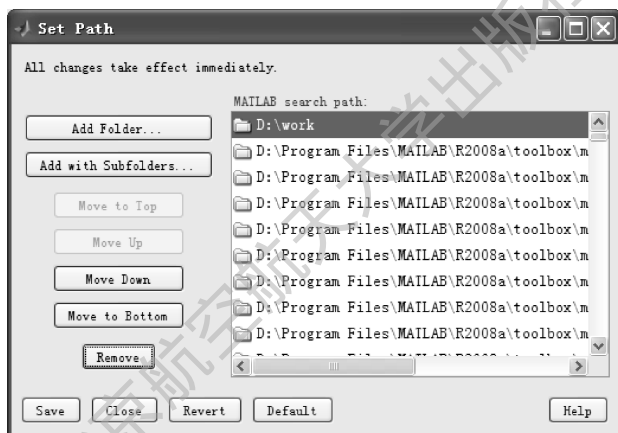


图 1.6-3 Set Path 对话框

为了合理地管理用户自定义目录和文件,可以将用户目录纳入搜索路径之下。搜索路径的设置方式有以下几种:

① 在当前目录窗口中右击需要设置为搜索路径的目录,选择 Add to Path,在弹出菜单中选择 Selected Folders 或者 Selected Folders and Subfolders,如图 1.6-4 所示。前者将所选路径添加到搜索路径,后者将所选路径以及该路径下的子目录都添加到搜索路径中。

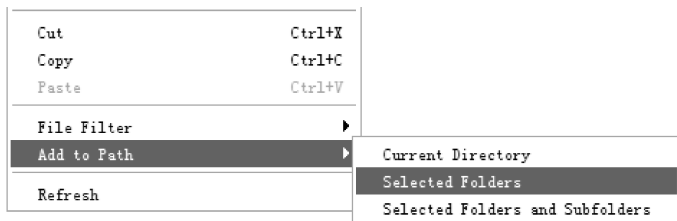


图 1.6-4 从当前目录下设置搜索路径

② 使用 path 函数,方法如下:

```
% 将指定的路径添加到搜索路径的最后位置
path(path, 'd:\Program Files\MATLAB\R2008a\work\MATLAB_100');
% 将指定的路径添加到搜索路径的最前位置
path('d:\Program Files\MATLAB\R2008a\work\MATLAB_100', path);
```

③ 使用 Set Path 对话框,如图 1.6-3 所示。单击 Add Folder 按钮,会出现“浏览文件夹”对话框,如图 1.6-5 所示。

选择需要添加到搜索路径中的目录,选择“确定”。或者选择 Add with Subfolders,这样可以将所选的目录及其子目录都添加到搜索路径中。最后按 Save 按钮保存搜索路径内容。如果要将搜索路径设置为 MATLAB 的默认值,那么选择 Default 按钮,然后按下 Save 按钮更新。

④ 直接修改 pathdef.m 文件修改搜索路径。

MATLAB 正是使用 pathdef.m 文件存放搜索路径信息的。使用前面的各种方法,系统会将搜索路径信息保存到该文件中,因此直接修改也可以达到修改目的。这种方法一般不使用,因为路径的输入比较麻烦而且容易出错。

2. 启动目录的设置技巧

启动目录是指 MATLAB 启动时的当前目录。一般将启动目录设置成用户习惯使用的目录。MATLAB 默认的启动目录为 C 盘用户文档中的 MATLAB 文件夹,例如 C:\Documents and Settings\Administrator\My Documents\MATLAB。使用默认的启动目录会有一些好处。比如,当更新 MATLAB 版本时,新版本 MATLAB 会直接将该目录作为启动目录,用户可以在不做任何更改的情况下继续正常运行用户文件。另外,由于系统对使用该计算机的所有用户都提供一个 My Documents 文件夹,彼此之间不能互相访问,具有良好的保密性。

启动目录的设置方法有:

(1) 使用 MATLAB 属性设置

右击 MATLAB 软件的桌面快捷方式,在弹出的菜单中,选择“属性”,在属性对话框的“起始位置”栏输入用户习惯使用的启动目录,如图 1.6-6 所示。



图 1.6-5 “浏览文件夹”对话框



图 1.6-6 在快捷方式中设置启动目录

(2) 使用 userpath 函数

userpath 函数返回 MATLAB 搜索路径的最顶层目录,如果没有设置图 1.6-6 中“起始位置”处的内容,那么 userpath 返回内容就是启动目录。因此可以通过 userpath 的设置来修改启动目录。方法如下:

```
% 设置该目录为启动目录
userpath('D:\Program Files\MATLAB\R2008a\work');
```

如果此时使用 path 或者从 Set Path 对话框中查看 MATLAB 搜索路径,会发现搜索路径的最上面一个已经变为所设置的 userpath 目录,而不再是默认的启动目录位置。这是和(1)中方法设置启动目录的区别。在(1)中设置好启动目录之后,搜索路径的顶层位置仍然是默认的启动目录位置。

如果要将启动目录返回默认位置,可以使用:

```
% 设置启动目录为默认位置
userpath('reset');
```

1.7 技巧 7: MATLAB 自带的 MEX 和 VR 编译器的安装和配置

1.7.1 技巧用途

MATLAB 作为当今世界上应用最为广泛的数学软件,具有数值运算功能强大、图形处理能力强大、程序环境高级但简单、工具箱丰富几个特点。高效利用 MATLAB 强大的工具箱对于解决实际问题具有重要意义,其中 MEX、VR 等有广泛的应用。本节将介绍如何准确快速地安装这些编译器。

1.7.2 技巧实现

1. MATLAB 自带的 MEX 编译器安装与配置

MEX 代表 MATLAB Executable,在 MATLAB 中可调用的 C 或 Fortran 语言程序称为 MEX 文件。MEX 文件是一种特殊的动态链接库函数,它能够在 MATLAB 里像一般的 M 函数那样来执行。该方法适用于 C/C++ 和 Fortran 语言。

MATLAB 自带的 MEX 编译器安装与配置操作如下:

(1) 设置合适的编译器

在 MATLAB 命令窗口中输入 mex -setup,完成编译器的设置。

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc - win32 C 2.4.1 in D:\PROGRA~1\MATLAB\R2008a\sys\lcc\bin
[2] Microsoft Visual C++ 6.0 in D:\Program Files\Microsoft Visual Studio
[0] None
Compiler: 1
Please verify your choices:
Compiler: Lcc - win32 C 2.4.1
```

```

Location: D:\PROGRA~1\MATLAB\R2008a\sys\lcc\bin
Are these correct [y]/n? y
Trying to update options file: C:\Documents and Settings\Administrator\Application Data\Math-
Works\MATLAB\R2008a\mexopts.bat From template: D:\PROGRA~1\MATLAB\R2008a\bin\win32\mexopts\
lccopts.bat
Done . . .
*****
Warning: The MATLAB C and Fortran API has changed to support MATLAB variables with more than 2^32 -
1 elements. In the near future you will be required to update your code to utilize the new API. You can
find more information about this at:
http://www.mathworks.com/support/solutions/data/1-5C27B9.html? solution=1-5C27B9
Building with the -largeArrayDims option enables the new API.
*****

```

(2) 设置系统路径

右击桌面“我的电脑”，然后单击“属性”→“高级”→“环境变量”→“系统变量”→Path 选项，出现“编辑系统变量”对话框，如图 1.7-1 所示。



图 1.7-1 “编辑系统变量”对话框

在“变量值”中增加以下路径：

头文件：C:\MATLAB7\extern\include

库文件：C:\MATLAB7\extern\lib\win32\microsoft\msvc60

DLL 文件：C:\MATLAB7\bin\win32

2. MATLAB 的 VR 工具箱安装与配置

虚拟现实(Virtual Reality)是指采用各种技术，来营造一个能使人有置身于现实世界中的感觉的环境。也就是要能使人产生和置身于现实世界中相同的视觉、听觉、触觉、嗅觉、味觉等。

MATLAB 的 Virtual Reality(虚拟现实)工具箱配置：安装三维实体的编辑器和浏览器。MATLAB 自带了三维实体编辑器和浏览器，安装步骤如下：

(1) 安装浏览器

在 Command Window 中输入 `vrinstall -install viewer`，会出现如下信息：

```

>> vrinstall -install viewer
Installing blaxxun Contact viewer ...
Do you want to use OpenGL or Direct3D acceleration? (o/d)

```

这条信息提示用户是选择 OpenGL 加速还是 Direct3D 加速，输入 o 或 d 选择相应的浏览器安装。输入 d 并回车后，会出现 blaxxun Contact viewer 的安装程序，如图 1.7-2 所示，用户按照提示一步步安装即可。

安装开始后，将会给出提示信息：

```

Starting viewer installation ...
Done.

```

(2) 安装编辑器

在 Command Window 中输入 `vrinstall -install editor`，按提示安装即可。

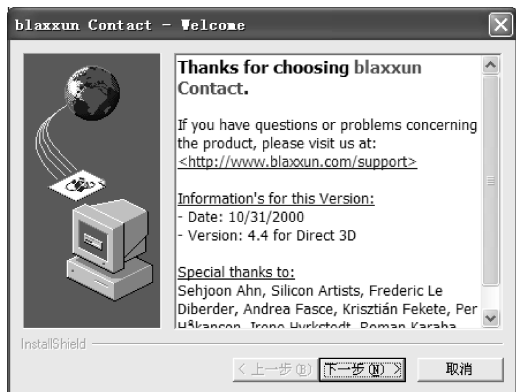


图 1.7-2 blaxxun Contact 安装对话框

```
>> vinstall - install editor
Starting editor installation ...
Done.
```

此外,用户使用 `vinstall-install` 命令会同时安装浏览器和编辑器。
用命令 `vinstall-check` 检查是否安装成功。

```
>> vinstall - check
External VRML viewer:    installed
VRML editor:            installed
```

1.8 技巧 8: 解决 Simulink 模型打不开的问题

1.8.1 技巧用途

Simulink 作为 MATLAB 的工具箱之一,是交互式动态系统建模、仿真和分析的图形系统,用户常常需要在 Simulink 环境下建立模型,进行系统的仿真和分析工作。但常常出现这样的问题:用户打开含有中文字符的模型,或者打开使用不同 MATLAB 版本软件制作的模型时,会出现错误提示,模型文件打不开。出现这样的问题该如何解决呢?本节给出几个常用的方法。

1.8.2 技巧实现

出现 Simulink 模型打不开的问题可能有如下几个原因:

① 模型的创建和打开所使用的 MATLAB 的版本不同。由于 MATLAB 不同版本之间的兼容性问题,使用不同版本的软件来打开模型时,有可能出现模型打不开的问题。因此,建议最好选用相应版本的软件来打开模型文件,这样可以避免出现上述情况,有时还能避免由于系统模型的转换带来的错误。

② 模型中含有不能识别的字符,比如中文字符等。由于 MATLAB 软件对中文字符支持方面的缺陷,在使用时可能会产生一些问题。

【例 1.8-1】 建立如图 1.8-1 所示的模型,保存为 `ex.mdl`。将模型中 Gain 模块的 Description 中加入中文字符“放大器”,如图 1.8-2 所示。然后单击 OK 按钮,会出现如

图 1.8-3 所示的错误提示。

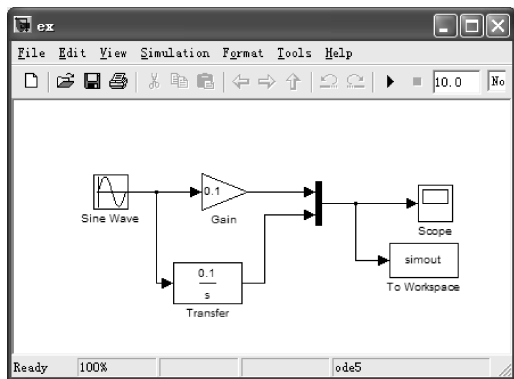


图 1.8-1 模型

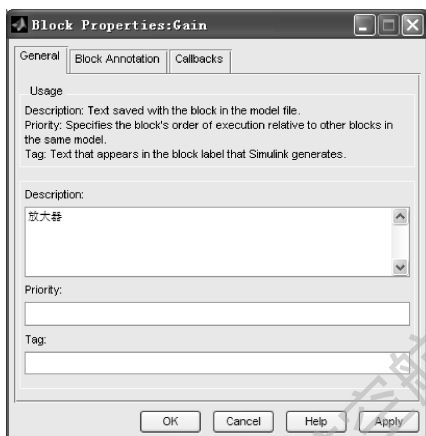


图 1.8-2 模块中加入中文注释

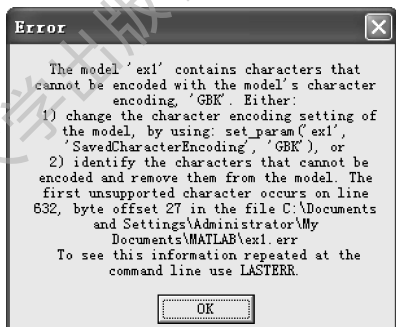


图 1.8-3 保存模块时的错误提示

出现这种错误提示是因为在模型中的中文字符是非 GBK 字符。这时在命令窗口中输入如下命令：

```
set_param('ex', 'SavedCharacterEncoding', 'ISO-8859-1')
```

第 1 个参数为模型名称，第 2 个参数为模型的“SavedCharacterEncoding”属性名称，第 3 个参数为属性值。

命令执行后，就可以保存模型了。但在命令窗口中会给出如下警告信息：

```
Warning: Model 'ex' is using character encoding 'ISO-8859-1', but is now being saved from a MATLAB session using the character encoding setting 'GBK'. String edits performed in this session may contain characters that will be misrepresented when the model is reloaded.
```

这是由于模型的字符集被用户设置为“ISO-8859-1”，而 Simulink 进程所默认使用的字符集为“GBK”，两者不一致所引起的。所以，当打开先前保存的模型时，会出现如图 1.8-4 所示错误提示。

要想打开带中文字符的 Simulink 模型，可按照如下方法进行：

① 在命令窗口中输入下列命令，关闭所有仿真窗口。

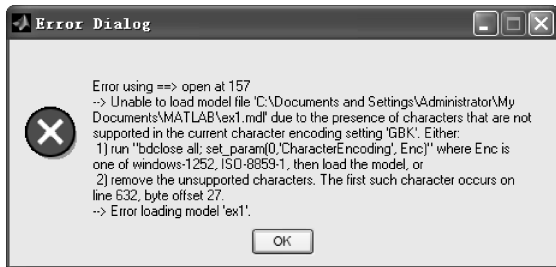


图 1.8-4 打开含有中文字符模块的错误提示

```
bdclose all;
```

② 设置 Simulink 进程的 CharacterEncoding 属性。

```
set_param(0, 'CharacterEncoding', 'ISO-8859-1');
```

运行完以上两条命令后,就可以顺利地打开 Simulink 模型了。

打开模型后,可以看到原先的中文字符变成了乱码,如图 1.8-5 所示。把中文字符去掉或改成英文字符,单击 OK,会弹出如图 1.8-6 所示对话框,选择 Save With New Name,在弹出的对话框中输入新的模型名称,重新保存模型文件。

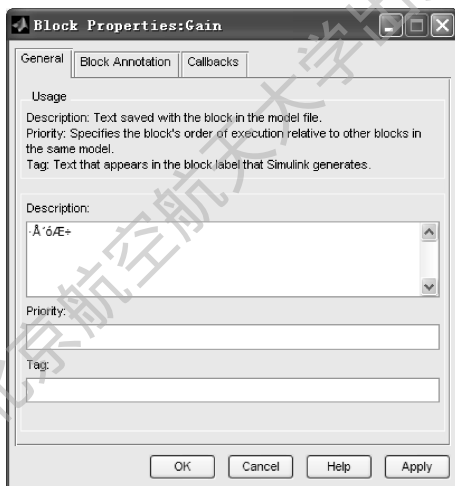


图 1.8-5 中文字符被修改后的界面

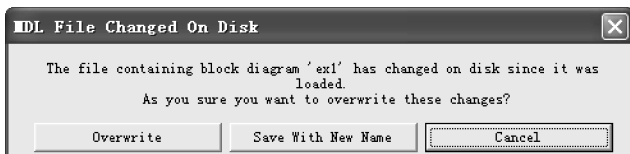


图 1.8-6 重新保存模型时的提示信息

【注】 调用“set_param(0, 'CharacterEncoding', 'ISO-8859-1')”命令后, MATLAB 窗口的 Current Directory 列表框中的中文路径将变为乱码,并且 MATLAB 也不再识别中文路径。因此,最好在打开模型后,把其中的中文字符修改为英文字符,然后调用“set_param(0, 'CharacterEncoding', 'GBK')”重新设置字符集。

2.1 技巧 9：操作图形窗口及其控件的通用方法

——set 和 get 命令

2.1.1 技巧用途

MATLAB 语言不仅提供了面向过程的编程方法,也提供了面向对象的编程方法。一个使用 MATLAB 语言开发的图形用户界面是由多个图形对象(graphical objects)组成的。

MATLAB 中的图形对象包括电脑屏幕(又称为根对象或 root 对象)、界面的窗口对象(figure)、用户菜单对象(uimenu)、用户工具条对象(uitoolbar)、坐标轴对象(axes)、用户界面控件对象(uicontrol,如 edit、pushbutton 等),还包括在坐标轴中绘制的各种线条对象,等等。

MATLAB 为所创建的图形对象分配一个唯一的句柄(handle),利用句柄来标识每一个图形对象。同时,每一个图形对象都有一组固定的属性值,用来标识对象的外观和行为,如界面窗口的位置(position)、编辑控件的背景颜色等。用户可以通过 MATLAB 提供的操作函数(方法),通过对象的句柄来访问或设置对象的属性值,如改变界面窗口的大小、改变图形的颜色、取得编辑框的输入等。

MATLAB 提供给用户操作对象的方法中有两个通用的方法,这就是 set 和 get 两个函数。MATLAB 把对象的所有特征都纳入对象的属性中,用户只要通过 set 和 get 函数,就可以非常方便地修改或设置对象的属性值,从而可以控制对象的外观和行为。

2.1.2 技巧实现

1. 设置对象的属性值: set 函数

set 函数常用的调用格式如下:

● **set(H,'PropertyName',PropertyValue,...)**

其中,H 为对象的句柄或句柄数组,PropertyName 为对象的属性名称,PropertyValue 为对应的属性值。如果一组对象都有相同的属性名称,用户若想把对象的属性设置为相同的数值,可以把对象的句柄全部包含到 H 数组中。

例如,设置界面上 Edit 中显示的内容为“my value!”,可以使用如下的方法:

```
% 设置 Edit 控件的 String 属性,edit_handle 为 Edit 控件的句柄
set(edit_handle,'String','my value!');
```

● **set(H,a)**

设置句柄 H 指向的对象属性的属性值。其中,a 是一个结构体(struct),结构体中的域名为对象的属性名称,对应的域值为相应的属性值。

【例 2.1-1】 在程序界面上创建一个坐标轴,修改坐标轴的字体大小为 12,背景颜色为

蓝色,代码如下:

```
% 创建 figure 对象
hfig = figure(1);
% 创建坐标轴对象,指定其父对象为 figure 1
haxes1 = axes('parent',hfig);
% 设置坐标轴的 Color 和 FontSize 属性
prop.Color = 'b';
prop.FontSize = 12;
set(haxes1,prop);
```

2. 取得对象的属性值: get 函数

get 函数常用的调用格式如下:

● get(h)

取得句柄 h 指向的对象的当前所有的属性值。

● get(h,'PropertyName')

取得句柄 h 指向的对象的 PropertyName 属性的属性值。

【例 2.1-2】 创建一个界面窗口,并查询标识其位置和大小度量单位 Units 的属性值。

```
% 创建界面窗口
hfig = figure(1);
% 查询其 Units 属性值
get(hfig,'units')
% 其 Units 属性值为 pixels(像素)
ans =
pixels
```

【例 2.1-3】 查询在界面窗口中可以设置哪些形状的鼠标指针。

```
% figure 的 Pointer 属性标识了鼠标指针的形状
set(gcf,'pointer')
    返回值为:[ crosshair | fullcrosshair | {arrow} | ibeam | watch | topl | topr | botl | botr | left
| top | right | bottom | circle | cross | fleur | custom | hand ]
```

set 语句查询 figure 对象的 Pointer 属性所有可能的属性值。其中,“{arrow}”属性值表示 figure 的 Pointer 属性的默认值为 arrow,即鼠标指针的形状为箭头形状。要查询 figure 窗口当前所使用的鼠标指针的形状,调用 get 命令:

```
>> get(gcf,'pointer')
ans =
arrow
```

【例 2.1-4】 查询计算机屏幕的尺寸。

```
% 首先取得标识计算机屏幕大小的度量单位
>> get(0,'units')
ans =
pixels
% 取得屏幕的尺寸
>> get(0,'screensize')
```

ans =

1

1

1280

800

即屏幕的尺寸为 1280×800 像素点。

如果在 GUIDE 编程环境中,用户还可以利用“属性查看器”(Property Inspector)这一交互式工具来设置或查看图形对象的属性值。“属性查看器”可以通过双击图形对象打开,也可以通过在命令窗口中输入命令 inspect 来打开。

2.2 技巧 10: 定义回调函数需遵循的语法规则

2.2.1 技巧用途

在 MATLAB 中,回调函数(Callback)是一个重要的概念,它是用来响应图形对象的事件的。当用户或程序触发了事件后,如用户按下按钮(pushbutton)、单击鼠标、按下键盘的按键、定时器的定时时间到达等,都会触发 MATLAB 自动调用这些事件的回调函数来执行相应的操作。

在 MATLAB 编程中,不同的图形用户界面(GUI)程序共享同一个事件队列。同时,不同的 MATLAB 控件,也有不同的回调函数属性。例如:figure 的 WindowButtonDownFcn、WindowButtonUpFcn 和 WindowButtonMotionFcn 用来处理鼠标事件;按钮的 Callback 用来处理按钮按下这一事件,等等。

在使用 MATLAB 的 GUIDE 工具来开发图形用户界面程序时,可以由 GUIDE 自动生成回调函数的框架,用户只需在函数体内添加事件的处理代码即可。如果用户直接编写 M 文件来开发 GUI 程序,则需要手工编写回调函数的框架及其执行代码,这就需要用户了解定义回调函数需要遵循的语法规则。

2.2.2 技巧实现

在 MATLAB 程序中,回调函数有如下几种编写方式:

① 如果回调函数要执行的语句比较少,可以把这些代码写成字符数组的形式直接赋值给对象的回调函数属性。

【例 2.2-1】 定义按钮 pushbutton1 的 Callback 属性,回调函数要执行的语句使用“[]”和“'”符号括起来。

% 定义 M 文件的主函数名称为 DefineCallback,不带输入和输出参数

```
function DefineCallback
```

```
% 创建界面窗口
```

```
hFig= figure('units','normalize',...
```

```
    'position',[0.4 0.4 0.3 0.2]);
```

```
% 在窗口中创建按钮控件,并设置其 Callback 属性值为字符数组
```

```
uicontrol('parent',hFig,...
```

```
    'style','pushbutton',...
```

```
    'String','Execute Callback',...
```

```
    'units','normalize',...
```

```
    'position',[0.4 0.4 0.3 0.2],...
```

```
    'callback','figure;',...
```

```
'x=0:pi/20:2*pi;','...
'y=sin(x);','...
'plot(x,y);']];
```

程序运行时,单击 Execute Callback 按钮,程序会逐句执行[]内的命令。[]内的每条命令必须用两个单引号“'”括起来,每条语句之间用逗号“,”隔开。程序运行的界面如图 2.2-1 所示。

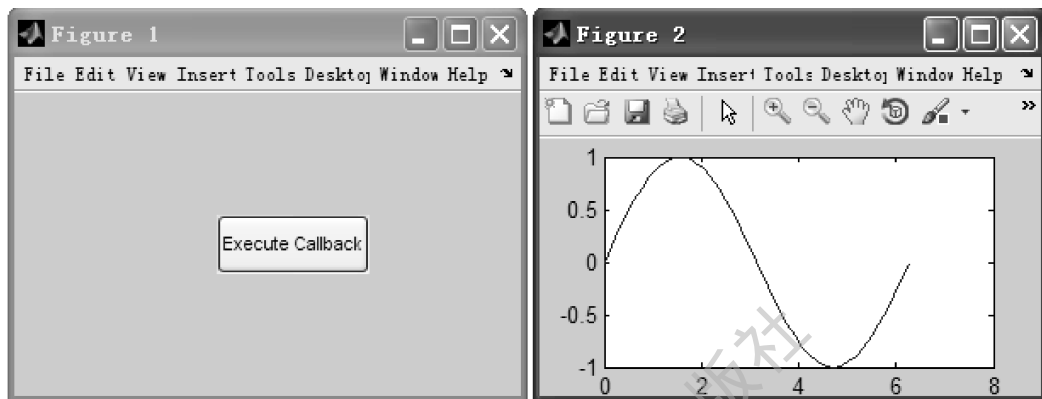


图 2.2-1 运行结果

② 如果回调函数要执行的语句比较多,或者为了简便起见,回调函数可以编写为单独的函数 M 文件或函数体。

在这种情况下,MATLAB 对定义回调函数有严格的语法规则,用户必须按照这些规则来定义回调函数。回调函数的语法规则如表 2.2-1 所列,表中内容以定义下压按钮控件的 Callback 回调函数为例。

表 2.2-1 定义回调函数的语法规则

设置对象的 Callback 属性	定义回调函数框架代码
set(hObject,'Callback','myfile')	function myfile
set(hObject,'Callback',@myfile)	function myfile(obj, event)
set(hObject,'Callback',{'myfile',5,6})	function myfile(obj,event,arg1,arg2)
set(hObject,'Callback',{'@myfile',5,6})	function myfile(obj,event,arg1,arg2)

在第 1 种情况下,回调函数没有输入参数,回调函数必须保存成单独的 M 文件。

在第 2 种情况下,对象 hObject 的 Callback 属性设置为函数句柄的形式。这种情况下,回调函数 myfile 必须带两个参数:obj 表示调用该回调函数的对象的句柄,如 pushbutton 的句柄;event 是个结构体,其中包含了事件的信息。这时的回调函数可以是单独的函数 M 文件,也可以写在主函数 M 文件内。

在第 3 种情况下,对象 hObject 的 Callback 属性设置为{'myfile',5,6},回调函数不仅必须带 obj 和 event 两个参数,而且还包含了用户需要传递的其他参数。其中,用户传递的参数的个数不受限制。这时,回调函数也必须保存成单独的 M 文件。

在第 4 种情况下,对象 hObject 的 Callback 属性设置为{@myfile,5,6},回调函数不仅必须带 obj 和 event 两个参数,而且还包含了用户需要传递的其他参数。其中,用户传递的参数

的个数不受限制。这时的回调函数可以是单独的函数 M 文件,也可以写在主函数 M 文件内。

【例 2.2-2】 将例 2.2-1 中的回调函数定义为单独的子函数。

```
% 定义 M 文件的主函数名称为 DefineCallback, 不带输入和输出参数
function DefineCallback
% 创建界面窗口
hFig= figure('units','normalize',...
    'position',[0.4 0.4 0.3 0.2]);
% 在窗口中创建按钮控件
hpush= uicontrol('parent',hFig,...
    'style','pushbutton',...
    'String','Execute Callback',...
    'units','normalize',...
    'position',[0.4 0.4 0.3 0.2]);
% 设置按钮的 Callback 属性
set(hpush,'callback',@mycallback);

% 定义回调函数为子函数
function mycallback(hobj,event)
figure;
x=0:pi/20:2*pi;
y=sin(x);
plot(x,y);
```

2.3 技巧 11: 元胞数组 (Cell Array) 的使用方法

2.3.1 技巧用途

元胞数组 (Cell Array) 是 MATLAB 语言中比较特殊的数据类型,它是由一系列元胞 (Cell) 构成的数组。

每一个元胞可以存放不同类型的数据,每个元胞中的数据可以是数字、字符或字符串、数字数组或字符串数组,也可能是元胞数组或结构数组,等等。

由于元胞数组对元胞内的数据类型没有限制,可以为不同的数据类型,因此,使用元胞数组可以存储不同类型的数据,这对用户的编程有很大的方便性和灵活性。

2.3.2 技巧实现

1. 元胞数组的创建

创建元胞数组常用的方法有 3 种:

① 使用语句直接生成元胞数组。使用英文的大括号“{ }”来创建元胞数组,就像使用中括号“[]”来创建数字数组或字符串数组一样。

【例 2.3-1】 直接使用语句来创建元胞数组 a。

```
% 在英文输入法下输入{}和[]
>> a = {'hello' [1 2 3;4 5 6];1 {'1' '2'}}
a =
    'hello'      [2x3 double]
    [ 1]        {1x2 cell }
```

② 对元胞数组中的各元胞逐一赋值,从而创建元胞数组。

【例 2.3-2】 将元胞数组 a 中的元胞逐一赋值。

```
>> a{1,1} = 'hello';a{1,2} = [1 2 3;4 5 6];a{2,1} = 1;a{2,2} = {'1' '2'};
>> a
a =
    'hello'      [2x3 double]
    [ 1]        {1x2 cell }
```

③ 使用 cell 函数来生成一个空的元胞数组,然后对每一个元胞赋值。

【例 2.3-3】 使用 cell 函数来创建元胞数组。

```
% 生成 2x3 的、元素为空的元胞数组
>> a = cell(2,3)
a =
    []      []      []
    []      []      []
```

2. 元胞数组的使用和操作

可以采用“{}”和“()”两种方式来访问元胞数组的元素。两种方式返回的结果是不同的:用 cellname{m,n} 返回的是元胞数组在(m,n)位置上的元胞中的数据,用 cellname(m,n) 返回的是元胞数组在(m,n)位置上的元胞。

以下是几个常用的对元胞数组进行操作的函数:

(1) iscell 函数

该函数用来判断某一数组是否为元胞数组。如果数组是元胞数组,则函数返回 1;如果数组不是元胞数组,则函数返回 0。

【例 2.3-4】 判断数组 A 是否为元胞数组。

```
% 定义一个元胞数组 A
>> A = {1 2 3};
% 判断 A 是否为元胞数组,如果为元胞数组,则函数返回值为 1
>> tf = iscell(A)
tf =
    1
```

(2) celldisp 函数

该函数用来显示元胞数组中的内容。常用的调用格式如下:

● celldisp(C)

直接显示元胞数组 C 中的内容。

● celldisp(C,name)

显示元胞数组 C 中的内容,数组名称 C 用 name 代替。

【例 2.3-5】 显示元胞数组 C 中的内容。

```
>> clear
>> C = {'Smith' [1 2;3 4] [12]};
% 直接显示元胞数组 C 中的内容
>> celldisp(C)
C{1} =
Smith
C{2} =
     1     2
     3     4
C{3} =
    12
% 显示元胞数组 C 中的内容,数组的名称用 cellcontent 代替
>> celldisp(C,'cellcontent')
cellcontent{1} =
Smith
cellcontent{2} =
     1     2
     3     4
cellcontent{3} =
    12
```

(3) cellstr 函数

C = cellstr(S)

该函数把字符串数组转换为元胞数组。其中,S 是一个 $m \times 1$ 的字符串数组,cellstr 把 S 的每一行字符串转换为元胞,并去除字符串结尾处的空格。

【例 2.3-6】 将字符串数组转换为元胞数组。

```
>> S = ['abc '; 'defg'; 'hi m'];
>> cellstr(S)
ans =
    'abc'      % 原先 abc 后面的空格被清除
    'defg'
    'hi m'     % i 和 m 之间的空格仍然保留
```

(4) cellplot 函数

该函数以图形化的方式显示元胞数组中的内容。

【例 2.3-7】 显示元胞数组 S 中的内容(包括空格和字符)。

```
>> S = {'abc ', 'defg', 'hi m'};
>> cellplot(S)
```

运行结果如图 2.3-1 所示。可以看出,元胞数组中每个元胞中的内容都很直观地标识在图中,灰色方格标识的是元胞中的字符,白色方格标识的是元胞中的空格。

(5) num2cell 函数

该函数把数字数组转换为元胞数组,调用格式如下:

C = num2cell(A, dims)

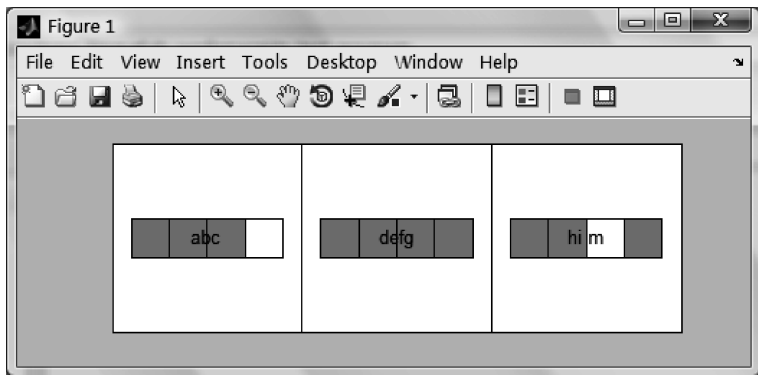


图 2.3-1 以图形方式显示元胞数组的内容

其中, A 是要转换的数字数组; dim 表示按行或按列转换, $\text{dim}=1$ 表示把 A 的每一列转换为一个元胞, $\text{dim}=2$ 表示把 A 的每一行转换为一个元胞。

【例 2.3-8】 将数字数组 A 按行或按列转换为元胞数组。

```
% A 是 4x3 的数组
>> A = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
% 把 A 的每一列转换为一个元胞, 得到的 C 是 1x3 的元胞数组
>> C = num2cell(A,1)
C =
    [4x1 double]    [4x1 double]    [4x1 double]
% 把 A 的每一行转换为一个元胞, 得到的 C 是 4x1 的元胞数组
>> C = num2cell(A,2)
C =
    [1x3 double]
    [1x3 double]
    [1x3 double]
    [1x3 double]
```

(6) `struct2cell` 和 `cell2struct` 函数

见“技巧 12: 结构数组(struct array)的使用方法”。

2.4 技巧 12: 结构数组(struct array)的使用方法

2.4.1 技巧用途

结构数组(struct array)是 MATLAB 中的一种重要的数据类型。同元胞数组类似, 结构数组也可以存放不同类型的数据; 但结构数组的内容更加丰富、应用更加广泛, 很多复杂的编程问题使用结构数组就会变得简单方便。例如, MATLAB 中的句柄结构就是结构数组的一个很好的例子, 因此, 使用好结构数组会给用户的编程带来很大的便利。

2.4.2 技巧实现

结构数组是由结构(struct)组成的, 每一个结构都包含多个结构域(fields)。例如, 多个图

形对象构成一个结构数组,每个图形对象就是一个结构,对象的一个属性就对应着一个结构域。数据不能直接存储在结构中,只能存放在结构域中。结构域可以存放任何类型和任何大小的数据。

在 MATLAB 中,可以使用如下方法来操作结构数组。

1. 结构数组的创建

常用的创建结构数组的方法有两种:

(1) 用直接法创建

【例 2.4-1】 使用直接法来创建结构数组。

```
>> A(1).name = 'Pat';
A(1).number = 176554;
A(2).name = 'Tony';
A(2).number = 901325;
>> A

A =

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

(2) 通过 struct 函数创建

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

其中,s 为结构数组的名称,field1 和 field2 为结构域的名称,values1 和 values2 为结构域的数据。

【例 2.4-2】 利用 struct 函数来创建结构数组。

```
>> A(1) = struct('name','Pat','number',176554);
A(2) = struct('name','Tony','number',901325);
>> A

A =

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

2. 结构数组的操作

(1) deal 函数

该函数用来得到结构数组中的指定结构域的值。用户可以使用形如“mystruct.name1”的命令形式来直接取得结构体中对应结构域的值,也可以使用 deal 函数来一次得到多个结构域的值。

【例 2.4-3】 使用 deal 函数来得到结构体中各结构域的值。

```
% 定义结构数组 A
>> A.name = 'Pat'; A.number = 176554; A(2).name = 'Tony'; A(2).number = 901325;
% 得到结构数组中所有 name 结构域的数据
>> [name1,name2] = deal(A(:,).name)
```

```
name1 =
    Pat
name2 =
    Tony
```

(2) getfield 函数

该函数用来取得结构数组中指定结构域的值。函数的调用格式如下：

● **f = getfield(s, 'field')**

当 s 是 1×1 的结构数组时,得到 s 中 field 结构域的值。

● **f = getfield(s, {i,j}, 'field', {k})**

当 s 是 $m \times n$ 的结构数组时,返回结构数组的指定元素 s(i,j)的指定结构域 field 的指定值,相当于 $f = s(i,j).field(k)$ 。

【例 2.4 - 4】 使用 getfield 函数来取得结构体中结构域的值。

```
% 定义 mystr 结构数组
>> mystr(1,1).name = 'alice';mystr(1,1).ID = 0;mystr(2,1).name = 'gertrude';
mystr(2,1).ID = 1;
% 取得 mystr(2,1)的结构域 name 的值
>> f = getfield(mystr, {2,1}, 'name')
f =
gertrude
```

【注】 在使用 getfield 函数时,结构数组元素的下标和结构域的下标要用 {} 括起来。

(3) setfield 函数

该函数用来设置结构数组中结构域的值。函数的调用格式如下：

● **s = setfield(s, 'field', v)**

● **s = setfield(s, {i,j}, 'field', {k}, v)**

其中, v 是结构域的新值。该函数的使用方法与 getfield 类似。

(4) rmfield 函数

该函数用来删除结构数组中的结构域。函数的调用格式如下：

● **s = rmfield(s, 'fieldname')**

删除结构域 fieldname。

● **s = rmfield(s, fields)**

删除 fields 中指定的结构域。

【例 2.4 - 5】 删除结构数组中的指定结构域。

```
% 定义结构数组 s
>> s.field1 = [1 2 3];s.field2 = 'string';s.field3 = {1 2 3;4 5 6};
% 删除结构域 field1
>> s = rmfield(s, 'field1')
s =
    field2: 'string'
    field3: {2x3 cell}
% 删除结构域 'field2', 'field3'
>> s = rmfield(s, {'field2', 'field3'})
```

```
s =
    field1: [1 2 3]
```

【注】必须用“s=rmfield(s, …);”才能删除结构数组中的结构域,即必须把 rmfield 函数的返回值赋值给原结构数组,才能保证删除结果生效。

(5) 判断函数

● tf = isfield(S, 'fieldname')

判断 S 中是否存在 fieldname 结构域。

● tf = isfield(S, C)

判断 S 中是否存在 C 中所含的结构域,C 是 cell 数组。

● tf = isstruct(A)

判断 A 是否是结构数组。

(6) struct2cell 和 cell2struct 函数

这两个函数实现 struct 数组和 cell 数组的相互转换。

● c = struct2cell(s)

将 struct 数组转换为 cell 数组。

【例 2.4-6】 将结构数组转换为元胞数组。

```
% 定义结构数组 s
```

```
>> s.field1 = [1 2 3]; s.field2 = 'string'; s.field3 = {1 2 3; 4 5 6};
```

```
>> s
```

```
s =
    field1: [1 2 3]
    field2: 'string'
    field3: {2x3 cell}
```

```
% 将结构数组转换为元胞数组
```

```
>> c = struct2cell(s)
```

```
c =
    [1x3 double]
    'string'
    {2x3 cell }
```

● s = cell2struct(c, fields, dim)

将 cell 数组转换为 struct 数组,结构域名由 fields 指定。

【例 2.4-7】 将元胞数组转换为结构数组。

```
>> c = {'birch', 'betula', 65; 'maple', 'acer', 50}
```

```
c =
```

```
    'birch'    'betula'    [65]
    'maple'    'acer'      [50]
```

```
% fields 包含 struct 中的结构域名
```

```
>> fields = {'name', 'genus', 'height'};
```

```
% dim = 2 表示把 c 中的各行转换为 struct 数组
```

```
>> s = cell2struct(c, fields, 2);
```

```
s =
```

```
2x1 struct array with fields:
```

```

name
genus
height
>> s(1)
ans =
name: 'birch'
genus: 'betula'
height: 65
>> s(2)
ans =
name: 'maple'
genus: 'acer'
height: 50
>> fields = {'field1', 'field2'};
% dim = 1 表示把 c 中的各列转换为 struct 数组
>> s = cell2struct(c, fields, 1);
>> s(1)
ans =
field1: 'birch'
field2: 'maple'
>> s(2)
ans =
field1: 'betula'
field2: 'acer'
>> s(3)
ans =
field1: 65
field2: 50

```

2.5 技巧 13: 矩阵(Matrix)的常用操作方法

2.5.1 技巧用途

MATLAB 中数据的基本格式是矩阵。MATLAB 中的矩阵是一个广义的概念,行向量、列向量和标量都是矩阵的特例。矩阵可以是二维的,也可以是多维的。掌握矩阵的常用基本操作,对用户利用矩阵来解决自己的问题很有帮助。

2.5.2 技巧实现

1. 查找矩阵中的元素

(1) find 函数

在 MATLAB 中,可以调用 find 函数在矩阵中查找满足一定条件的元素。find 函数常用的调用格式为:

● **ind = find(X)**

● $[m\ n] = \text{find}(X)$

其中, X 为要查找的矩阵, ind 保存元素在矩阵 X 中的线性索引值。因为在 MATLAB 中, 矩阵是按列存储的, ind 的值表示元素在矩阵中按列存储时的位置。 m 和 n 是列向量, 分别保存元素在矩阵中的位置的行下标和列下标。

【例 2.5 - 1】 find 函数的使用方法。

```
% 定义矩阵 A
>> A = [1 2 3 4; 5 6 7 8];
% 查找 A 中所有大于 3 的元素, 返回元素的线性索引值
>> ind = find(A > 3)
ind =
     2
     4
     6
     7
     8
% 查找 A 中所有大于 3 的元素, 返回元素的行下标和列下标
>> [m n] = find(A > 3)
m =
     2
     2
     2
     1
     2
n =
     1
     2
     3
     4
     4
```

(2) ind2sub 和 sub2ind 函数

这两个函数实现线性索引值和行、列下标之间的转换。函数的调用格式如下:

● $[I, J] = \text{ind2sub}(\text{siz}, \text{IND})$

● $\text{IND} = \text{sub2ind}(\text{siz}, I, J)$

其中, siz 表示矩阵的大小, IND 保存元素的线性索引值, I 和 J 保存元素的行下标和列下标。

【例 2.5 - 2】 矩阵元素的线性索引与行列下标之间的转换。

```
>> A = [1 2 3 4; 5 6 7 8];
>> ind = find(A > 3);
>> [m n] = find(A > 3);
>> [I J] = ind2sub(size(A), ind)
>> IND = sub2ind(size(A), I, J)
I =
     2
     2
     2
```

```

1
2
J =
1
2
3
4
4
IND =
2
4
6
7
8

```

2. 删除矩阵中的指定元素

若想删除矩阵中的指定元素,只需将这些元素赋值为空(“[]”)即可。例如,假设 A 是 $m \times n$ 维的矩阵,可以使用如下命令删除矩阵中的指定元素:

① **A(sub2ind(size(A),i,j)) = []** 删除 A 的第 i 行、第 j 列的元素。

② **A(i,:) = []** 删除 A 的第 i 行的数据。

③ **A(i:j,:) = []** 删除 A 的第 i 行到第 j 行的数据。

④ **A(:,j) = []** 删除 A 的第 j 列的所有元素。

⑤ **A(:,i:j) = []** 删除 A 的第 i 列到第 j 列的数据。

【例 2.5-3】 删除矩阵中的指定元素。

```

>> A = [1 2 3 4;5 6 7 8];
% 删除 A 的第 1 行的数据
>> A(1,:) = []
A =
     5     6     7     8
>> A = [1 2 3 4;5 6 7 8];
% 删除 A 的第 1 列的数据
>> A(:,1) = []
A =
     2     3     4
     6     7     8

```

【注】 对于矩阵中单个元素的删除,MATLAB 只允许使用线性索引值来指定该元素。所以在①中调用 sub2ind(size(A),i,j)来转换。如果使用“A(i,j)=[];”命令,则 MATLAB 会提示错误“??? Subscripted assignment dimension mismatch.”。

3. 取得矩阵中的指定元素

用户可以使用如下方法来取得矩阵的某一(些)行或列的元素:

- **X=A(i,:)** 取得 A 的第 i 行数据,并赋值给变量 X。
- **X=A(i:j,:)** 取得 A 的第 i 行到第 j 行数据,并赋值给变量 X。
- **Y=A(:,j)** 取得 A 的第 j 列的数据。

- **Y=A(:,i:j)** 取得 A 的第 i 列到第 j 列的数据。
- **Z=A(i:j,n:m)** 取得矩阵第 i 行到第 j 行以及第 n 列到第 m 列之间的数据。

4. 查询矩阵的大小

设 A 是一个 $m \times n$ 的矩阵, 可以调用以下函数来得到矩阵 A 的大小, 即矩阵的行数和列数等信息。

- **num = size(A)** 返回矩阵的行数和列数, num 是个 1×2 的数组, 第 1 个数值是矩阵的行数, 第 2 个数值是矩阵的列数。
- **num = length(A)** 返回 A 的行数和列数的最大值, 相当于 $\max(\text{size}(A))$ 。
- **num = size(A,1)** 返回矩阵 A 的行数。
- **num = size(A,2)** 返回矩阵 A 的列数。

【例 2.5 - 4】 查询矩阵的大小。

% 由 rand 命令生成的一个 $4 \times 3 \times 2$ 的三维矩阵

```
>> A = rand(4,3,2);
```

% 得到矩阵的大小信息

```
>> num = size(A)
```

```
num =
```

```
4      3      2
```

```
>> num = length(A)
```

```
num =
```

```
4
```

```
>> num = size(A,1)
```

```
num =
```

```
4
```

```
>> num = size(A,2)
```

```
num =
```

```
3
```

```
>> num = size(A,3)
```

```
num =
```

```
2
```

5. 取得矩阵中元素的最大值和最小值: max 和 min 函数

- **C = max(A)** 取得矩阵 A 中每一列的最大值, 组成行向量返回给 C。
- **C = max(A,B)** 取得矩阵 A 和 B 对应元素的最大值。
- **C = max(A,[],dim)** 取得矩阵每行或每列的最大值, dim=1 表示每列的最大值组成行向量, dim=2 表示每行的最大值组成列向量。
- **C = min(A)** 取得矩阵 A 中每一列的最小值, 组成行向量返回给 C。
- **C = min(A,B)** 取得矩阵 A 和 B 对应元素的最小值。
- **C = min(A,[],dim)** 取得矩阵每行或每列的最小值, dim=1 表示每列的最小值组成行向量, dim=2 表示每行的最小值组成列向量。

【例 2.5 - 5】 求矩阵的最大和最小值。

```
>> clear
```

```
>> a = [2 3; 3 6; 4 9]
```

```
a =
```

```

2      3
3      6
4      9
>> b = [1 4;4 5;5 8]
b =
     1     4
     4     5
     5     8
>> max(a)
ans =
     4     9
>> min(a)
ans =
     2     3
>> max(a,b)
ans =
     2     4
     4     6
     5     9
>> max(a,[],2)
ans =
     3
     6
     9
>> max(a,[],1)
ans =
     4     9

```

2.6 技巧 14：字符串的操作方法

2.6.1 技巧用途

在使用 MATLAB 时经常遇到对字符和字符串的操作。字符串能够在计算机屏幕上显示,也可以用来组成一些 MATLAB 命令。

一个字符串是存储在一个行向量中的文本,这个行向量中的每一个元素代表一个字符。字符串可以由零或多个字符组成,一般记为 $s = 'a_1 a_2 \cdots a_n'$ ($n \geq 0$), $a_1 \sim a_n$ 可以是字母、数字或特殊字符,每个字符占 1 位。

实际上,字符向量中的元素存放的是字符的 ASCII 码值,当在屏幕上显示字符变量的值时,显示的是文本,而不是其 ASCII 码值。由于字符串是以向量的形式来存储的,所以可以通过其下标来访问字符串中的元素。

多个字符串也可以构成字符矩阵,但是矩阵的每行字符数必须相同。

MATLAB 提供了操作字符串的方法和一些函数,熟练掌握这些方法和函数有助于用户利用 MATLAB 对各种字符串问题进行处理。

2.6.2 技巧实现

1. 字符串的创建

MATLAB 中创建字符串非常简单,将字符串中的字符放到一对单引号之间即可。该对单引号必须在英文状态下输入。

【例 2.6-1】 创建字符串 string。

```
% 创建普通字符串
>> string = 'To study MATLAB! '
string =
To study MATLAB!
% 创建带单引号的字符串,在出现单引号的地方用两个单引号代替(')
>> string = 'We're going to study MATLAB! '
string =
We're going to study MATLAB!
>>
```

2. 字符串中元素的访问和操作

字符串是以向量的形式存储的,可以通过其下标访问其中的元素。

【例 2.6-2】 将上述 string 字符串中的 study 替换为 learn。

```
% 创建字符串 string
>> string = 'We're going to study MATLAB! '
string =
We're going to study MATLAB!
% 将其中的 study 替换为 learn
>> string(16:20) = 'learn'
string =
We're going to learn MATLAB!
```

【例 2.6-3】 取出 string 中的子串 learn。

```
>> subString = string(16:20)
subString =
learn
```

【例 2.6-4】 将上述 string 字符串倒排。

```
>> newString = string(end:-1:1)
newString =
! BALTAM nrael ot gniog er'eW
```

冒号表达式的使用和在数值数组中的使用相同。

【例 2.6-5】 计算字符串中字符的个数。

```
>> [r c] = size(string)
r =
1
c =
28
```

其中, r 代表行数; c 代表列数, 即字符数。

3. 字符串中字符的 ASCII 码值

字符串中的字符是以其对应的 ASCII 码值来存储的。abs 和 double 命令都可以用来获取字符串对应的 ASCII 码数值数组。char 命令则可以把 ASCII 码数值数组转换为字符串。

【例 2.6-6】字符串的 ASCII 码值与字符串的转换。

```
% 取得 ASCII 码值
>> ascii_string = double(string)
ascii_string =
    Columns 1 through 21
    87    101    39    114    101    32    103    111    105    110    103    32    116    111
32    108    101    97    114    110    32
    Columns 22 through 28
    77    65    84    76    65    66    33
% 转换为字符串
>> string = char(ascii_string)
string =
We're going to learn MATLAB!
```

4. 多个字符串的连接和比较

MATLAB 提供了两个命令用于字符串的连接: strcat 和 strvcat。比较字符串的内容可以使用 strcmp 和 strcmp。

- **strcat(str1, str2, ...)** 将字符串 str1、str2……连接成行向量。
- **strvcat(str1, str2, ...)** 将字符串 str1、str2……连接成列向量, 各字符串必须有相同的字符个数。
- **strmatch(key, str)** 检查 str 中的各行, 返回一个列向量, 包含了各行以字符串 key 开头的行号。
- **strncmp(str1, str2, n)** 比较字符串 str1 和 str2 的前 n 个字符(区分大小写), 如果相同则返回 1, 不相同返回 0。
- **strncmpi(str1, str2, n)** 比较字符串 str1 和 str2 的前 n 个字符(不区分大小写), 如果相同则返回 1, 不相同返回 0。

【例 2.6-7】字符串的比较和连接。

```
>> str1 = 'abcdefg'; str2 = 'hijklmn'; str3 = 'abckjhl'; str4 = 'ABCKjhl';
% 连接 str1 和 str2 为行向量
>> newstr1 = strcat(str1, str2)
newstr1 =
abcdefg h i j k l m n
% 连接 str1、str2 和 str3 为列向量
>> newstr2 = strvcat(str1, str2, str3)
newstr2 =
abcdefg
hijklmn
abckjhl
% 在 newstr2 中查找以 'abc' 开头的各行
>> strmatch('abc', newstr2)
```

```
ans =
    1
    3
% 比较字符串 str1 和 str4 的前 3 个字符,区分字符的大小写
>> strncmp(str1,str4,3)
ans =
    0
% 比较字符串 str1 和 str4 的前 3 个字符,不区分字符的大小写
>> strncmpi(str1,str4,3)
ans =
    1
```

5. 数字数组和字符串的转换函数

- **num2str(A)** 将数字或数组 A 转换为字符串(数组)。
- **str2num(str)** 将字符串 str 转换为数字或数组。
- **mat2str(A)** 将数字数组 A 转换成字符串(行向量)。
- **int2str(A)** 把整数数值或数组转换为整数数字组成的字符串。

【例 2.6-8】 数字数组和字符串的转换。

% 创建数字数组 A

```
>> A = [1 2 3;4 5 6;78 89 10]
```

A =

```
    1     2     3
    4     5     6
   78    89    10
```

% 将数字数组 A 转换为字符数组

```
>> str = num2str(A)
```

str =

```
    1     2     3
    4     5     6
   78    89    10
```

```
>> whos str
```

Name	Size	Bytes	Class	Attributes
str	3x10	60	char	

% 将字符数组 str 转换为数字数组

```
>> num = str2num(str)
```

num =

```
    1     2     3
    4     5     6
   78    89    10
```

```
>> whos num
```

Name	Size	Bytes	Class	Attributes
num	3x3	72	double	

% 将数字数组 A 转换为字符串(行向量),向量中的元素包括“[”、“]”、“;”等字符。

```
>> str2 = mat2str(A)
```

str2 =

```
[1 2 3;4 5 6;78 89 10]
```

```
>> whos str2
      Name      Size      Bytes  Class  Attributes
      str2      1x22          44   char
```

6. 其他常用字符串操作函数

- **blanks(n)** 返回由 n 个空格组成的字符串。
- **deblank(str)** 去掉字符串 str 结尾处的空格。
- **strtrim(str)** 去掉 str 的开头和结尾的空格、制表符、换行符。
- **strread(str)** 从字符串中读取格式化的数据
- **lower(str)** 将 str 中的大写字母转换成小写字母。
- **upper(str)** 将 str 中的所有字母转换成大写字母。
- **isletter(str(i))** 如果 str 中的第 i 个字符是字母,则返回 1,否则返回 0。
- **isspace(str)** 返回一个和 str 大小相同的向量。如果在 str 中的某个位置为空格、制表符或换行符,则向量的相应位置元素为 1,否则为 0。
- **strcmp(str1,str2)** 比较字符串 str1 和 str2,若相等则返回 1,否则返回 0。区分大小写。
- **strcmp(str1,str2)** 比较字符串 str1 和 str2,若相等则返回 1,否则返回 0。不区分大小写。
- **findstr(str1,str2)** 返回一个向量,包含 str1 中出现子串 str2 的起始位置。
- **strfind(str,patten)** 查找 str 中是否有字符串 pattern,返回字符串出现的位置。
- **strrep(str1,str2,str3)** 把 str1 中含有 str2 的所有位置用 str3 来代替。
- **lasterr** 返回上一个错误信息的字符串。
- **lastwarn** 返回上一个警告信息的字符串。

【例 2.6-9】 字符串中字符的大小写的转换。

```
>> str = 'matlab'
str =
matlab
>> str1 = upper(str)
str1 =
MATLAB
>> str = lower(str1)
str =
matlab
```

【例 2.6-10】 查找在 str1 中出现 str2 的位置。

```
>> str1 = 'abcdefg';str2 = 'cdf';
>> findstr(str1,str2)
ans =
     []
>> str1 = 'abcdefg';str2 = 'cd';
>> findstr(str1,str2)
ans =
```

2.7 技巧 15：判断函数的使用方法

2.7.1 技巧用途

MATLAB 提供了众多的判断函数,用来判断某一变量或某一对象是否满足某些条件,然后根据这些条件分别对变量或对象进行相应的操作。

下面详细介绍这些判断函数的使用方法。

2.7.2 技巧实现

1. isappdata(h,name)函数

判断句柄 h 所指定的对象中是否存在名称为 name 的应用程序数据。h 为对象的句柄, name 是与该对象相关联的应用程序数据的名称。如果存在以 name 为名称的应用程序数据,则返回 1;否则,返回 0。

【例 2.7-1】 判断输入变量是否为应用程序数据。

```
% 创建 figure 对象
h = gcf;
value = rand(3);
% 设置应用程序数据,名称为 mydata
setappdata(h,'myappdata',value);
% 判断 myappdata 是否是应用程序数据
tf = isappdata(h,'myappdata')
tf =
    1
```

tf 的返回值为 1,表示 myappdata 是与当前窗口相关联的应用程序数据。

2. iscell(A)函数

判断输入变量 A 是否为元胞数组。如果是,则返回 1;否则,返回 0。

【例 2.7-2】 判断输入变量是否为元胞数组。

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

3. iscellstr(A)函数

判断输入变量 A 是否为包含字符串的元胞数组。如果是,则返回 1;否则,返回 0。

【例 2.7-3】 判断输入变量是否为包含字符串的元胞数组。

```
A{1,1} = 'Thomas Lee';
A{1,2} = 'Marketing';
A{2,1} = 'Allison Jones';
```

```
A{2,2} = 'Development';
iscellstr(A)
ans =
    1
```

4. ischar(A) 函数

判断输入变量 A 是否为字符数组。如果是,则返回 1;否则,返回 0。

【例 2.7-4】 判断输入变量是否为字符数组。

```
% 双精度数字数组
C{1,1} = magic(3);
% 字符数组
C{1,2} = 'John Doe';
% 复数数组
C{1,3} = 2 + 4i;
>> C
C =
[3x3 double]    'John Doe'    [2.0000 + 4.0000i]
% ischar 函数返回值表明只有 C{1,2} 是字符数组
for k = 1:3
    x(k) = ischar(C{1,k});
end
x
x =
    0     1     0
```

5. isdir(A) 函数

判断输入值是否是文件夹。如果是,则返回 1;否则,返回 0。

【例 2.7-5】 判断输入变量是否为文件夹名称。

```
>> tf = isdir('D:\work')
tf =
    1
```

6. isempty(A) 函数

判断输入数组 A 是否为空。如果是,则返回 1;否则,返回 0。

【例 2.7-6】 判断输入数组是否为空。

```
A = rand(2,3);
B = [];
tf1 = isempty(A)
tf2 = isempty(B)
tf1 =
    0
tf2 =
    1
```

7. isequal(A, B, ...) 函数

判断输入数组 A、B... 的值是否相等。A、B... 必须是同类型的大小相同的数组。

【例 2.7-7】 判断输入的两个数组的值是否相等。

```
A = [1 2 3;4 5 6];
B = [7 8 9;10 11 12];
tf1 = isequal(A,B)
tf1 =
    0
```

8. isfield(S, 'fieldname') 函数

判断结构体 S 中是否包含 fieldname 所表示的结构域。如果包含,则返回 1;否则,返回 0。

【例 2.7-8】 判断结构体中是否包含指定的结构域。

```
patient.name = 'John Doe';
patient.billing = 127.00;
isfield(patient,'billing')
ans =
    1
```

9. tf=isfinite(A) 函数

返回与数组 A 相同大小的数组。如果数组元素为有限值,则 tf 相应位置的值为 1;若数组元素的值为无穷大或 NaN,则 tf 相应位置的值为 0。

【例 2.7-9】 判断数组的元素是否为无穷大或 NaN。

```
a = [-2 -1 0 1 2];
isfinite(1./a)
ans =
     1     1     0     1     1
isfinite(0./a)
ans =
     1     1     0     1     1
```

10. ishandle(H) 函数

判断输入数组的各元素是否是有效的图形对象的句柄。如果是,则其值为 1;否则,其值为 0。

【例 2.7-10】 判断输入变量是否为有效的图形对象的句柄。

```
h = gcf;
tf1 = ishandle(h)
close all
tf2 = ishandle(h)
tf1 =
    1
tf2 =
    0
```

11. isglobal(A) 函数

判断输入变量 A 是否为全局变量。如果是,则返回 1;否则,返回 0。

【例 2.7-11】 判断输入变量是否为全局变量。

```
global A
A = 100;
isglobal(A)
ans =
    1
```

12. iskeyword('str') 函数

判断 str 是否为 MATLAB 的关键字。要得到 MATLAB 中的关键字,在命令窗口中输入 iskeyword 命令,得到 MATLAB 中的关键字有:

break、case、catch、classdef、continue、else、elseif、end、for、function、global、if、otherwise、parfor、persistent、return、switch、try、while。

13. isstruct(A) 函数

判断输入数组 A 是否为结构数组。

【例 2.7 - 12】 判断输入变量是否为结构数组。

```
patient.name = 'John Doe';
patient.billing = 127.00;
isstruct(patient)
ans =
    1
```

14. isnumeric(A) 函数

判断输入数组 A 是否为数字数组。

【例 2.7 - 13】 判断输入变量是否为数字数组。

```
C{1,1} = pi;
C{1,2} = 'John Doe';
tf1 = isnumeric(C{1,1})
tf2 = isnumeric(C{1,2})
tf1 =
    1
tf2 =
    0
```

15. islogical(A) 函数

判断输入数组 A 是否为逻辑数组。

【例 2.7 - 14】 判断输入变量是否为逻辑数组。

```
C{1,1} = ispc;
C{1,2} = 'John Doe';
tf1 = islogical(C{1,1})
tf2 = islogical(C{1,2})
tf1 =
    1
tf2 =
    0
```


16. ismember(A,S)函数

判断数组 A 中的元素是否存在于数组 S 中。

【例 2.7 - 15】 判断一个数组中的元素是否存在于另一个数组中。

```
set = [0 2 4 6 8 10 12 14 16 18 20];
A = [1;2;3;4;5];
ismember(A, set)
ans =
    0
    1
    0
    1
    0
```

17. isnan(A)函数

判断数组 A 中的元素是否为 NaN。

【例 2.7 - 16】 判断输入数组中的元素是否为 NaN。

```
a = [-2 -1 0 1 2];
isnan(0./a)
ans =
    0    0    1    0    0
```

18. isreal(A)函数

判断输入数组 A 是否为实数数组。

【例 2.7 - 17】 判断输入变量是否为实数。

```
x = magic(3);
y = complex(x);
isreal(x)
ans =
    1
isreal(y)
ans =
    0
```

19. isscalar(A)函数

判断输入 A 是否为标量。

【例 2.7 - 18】 判断输入变量是否为标量。

```
A = rand(5);
isscalar(A)
ans =
    0
isscalar(A(3,2))
ans =
    1
```

20. issorted(A) 函数

判断输入数组元素是否是有序(升序或降序)排列。

【例 2.7 - 19】 判断输入数组中的元素是否为有序排列。

```
A = [5 12 33 39 78 90 95 107 128 131];
issorted(A)
ans =
    1
```

21. isvalid(serial) 函数

判断串行端口对象是否有效。

【例 2.7 - 20】 判断串口对象是否有效。

```
s1 = serial('COM1');
s2 = serial('COM1');
delete(s2)
sarray = [s1 s2];
isvalid(sarray)
ans =
    1     0
```

2.8 技巧 16: varargin、varargout、nargin 和 nargout 的使用方法

2.8.1 技巧用途

函数是 MATLAB 中的一个重要的编程要素。在 MATLAB 中,用户可以根据需要编写自定义的函数。自定义函数时用户可以选择带或不带输入和输出参数(或返回参数)。一般情况下,为了便于与其他程序进行数据交换和共享,自定义函数最好带输入和输出参数。

如果函数的输入和输出参数的个数是确定的,这种情况很好处理。但是如果函数的输入和输出参数个数是不确定的,应该如何定义函数呢? MATLAB 语言为满足用户的这种需求,提供了 varargin、varargout、nargin 和 nargout 预定义的参数。利用这些预定义的参数,用户就能非常容易地处理函数的输入和输出参数不确定的情况。

2.8.2 技巧实现

1. 函数的输入和输出参数个数确定的情况

在这种情况下,用户可以按照如下方式来定义函数:

```
function [out1 out2 out3] = myfun(in1, in2)
out2 = in1 * in2;
out3 = 10;
out1 = in1.^2;
```

函数带有 2 个输入参数和 3 个输出参数,虽然在函数体内,变量的计算顺序是 out2、out3 和 out1,但函数的返回顺序是 out1、out2 和 out3,与返回参数在函数体内的计算顺序无关。

此外,实际调用函数时,其返回参数的个数可以少于函数定义的返回参数的个数。例如:

函数 myfun 可以返回 1 个参数、2 个参数或 3 个参数,下面举例说明。

【例 2.8-1】 有关函数 myfun 的返回值问题。

```
% 调用时无返回参数,这时函数默认返回第 1 个输出参数的值
>> myfun(20,30)
ans =
    400
% 调用时带 1 个返回参数,返回第 1 个输出参数的值
>> [value1] = myfun(20,30)
value1 =
    400
% 调用时带 2 个返回参数,返回前 2 个输出参数的值
>> [value1 value2] = myfun(20,30)
value1 =
    400
value2 =
    600
% 调用时带 3 个返回参数,返回全部输出参数的值
>> [value1 value2 value3] = myfun(20,30)
value1 =
    400
value2 =
    600
value3 =
    10
```

2. 函数的输入和输出参数个数不确定的情况

在这种情况下,MATLAB 语言可以使用 varargin 和 varargout 结构来定义函数的输入和输出参数,并使用 nargin 和 nargsout 来获取函数调用时输入参数和输出参数的个数。

(1) varargin 和 varargout

利用 varargin 和 varargout 可以传递任意数目的输入参数和输出参数。

varargin(variable length input argument list)和 varargout(variable length output argument list)都是可变长度的元胞数组,是 MATLAB 预定义的专用参数,可以分别用来存储函数的输入参数和输出参数。它们是在函数的输入和输出参数个数不确定的情况下使用,也就是可以应用到可变输入输出参数的函数中。

使用 varargin 有两种情况: function myfun(varargin)、function myfun(in1,in2,varargin)。

在第 1 种情况下,函数被调用时,MATLAB 使用 varargin{1}来接收函数的第 1 个输入值,varargin{2}来接收函数的第 2 个输入值,依此类推。

在第 2 种情况下,函数的第 1 个和第 2 个输入值由 in1 和 in2 来接收,varargin{1}接收函数的第 3 个输入值,varargin{2}来接收函数的第 4 个输入值,依此类推。

同样,使用 varargout 也有两种情况: function varargout = myfun(in1,in2,...)、function [out1 out2 varargout] = myfun(in1,in2,...)。

在第 1 种情况下,函数被调用时,MATLAB 使用 varargout{1}来接收第 1 个返回值,varargout{2}来接收第 2 个返回值,依此类推。

在第 2 种情况下,函数被调用时,MATLAB 使用 out1 和 out2 来接收第 1 个和第 2 个返回值,varargout{1}用来接收第 3 个返回值,varargout{2}用来接收第 4 个返回值,依此类推。

【注】 varargin 和 varargout 只用于包含可选参数的函数当中,在声明时,varargin 和 varargout 必须作为最后一个参数声明,在声明中必须小写。正确的使用格式如下:

```
function [out1 out2 varargout]=myfunc(in1,in2,varargin)
```

要取得 varargin 和 varargout 中的参数,可以使用 varargin{1}、varargin{2}、varargout{1}、varargout{2}等。

(2) nargin 和 nargout

在一个函数 M 文件内,调用 nargin 或 nargout 可以确定用户调用函数时设置了多少个输入参数和输出参数。

nargin 和 nargout 可以带有一个输入参数。该输入参数是一个函数名,用来确定该函数的输入和输出参数的个数,如 nargin(fun)和 nargout(fun)。

【例 2.8-2】 nargin 和 varargin 的使用方法。

```
% 定义函数 vartest,带 2 个独立输入参数和 1 个 varargin 数组
function vartest(argA, argB, varargin)
% 取得 varargin 中元素(输入值)的个数
optargin = size(varargin,2);
% 由 nargin 得到函数总的输入值的个数,从而求得在独立参数中的输入值的个数
stdargin = nargin - optargin;
fprintf(' 函数的输入值的个数为: %d\n', nargin)
fprintf(' 独立参数中的输入值(%d):\n', stdargin)
if stdargin >= 1
    fprintf('          %d\n', argA)
end
if stdargin == 2
    fprintf('          %d\n', argB)
end
fprintf('varargin 中包含的输入值(%d):\n', optargin)
for k = 1 : size(varargin,2)
    fprintf('          %d\n', varargin{k})
end
```

将上述代码保存为函数 M 文件,文件名为 vartest.m,然后在命令窗口中调用,结果如下:

```
>> vartest(10,20,30,40,50,60,70)
函数的输入值的个数为: 7
独立参数中的输入值(2):
    10
    20
varargin 中包含的输入值(5):
    30
    40
    50
    60
    70
```

【例 2.8-3】 nargin 和 varargout 的使用方法。

定义函数 mysize, 该函数用来求得输入数组的大小, 并返回这些信息。

```
function [s,varargout] = mysize(x)
% 调用 nargin 命令取得调用函数时返回参数的个数
out1 = nargin;
% 确定 varargout 中元素的个数
nout = max(out1,1) - 1;
% 将输入数组的行数和列数组成的数组赋值给输出参数 s
s = size(x);
% 分别将输入数组的行数和列数保存到 varargout 中
for k = 1:nout
    varargout(k) = {s(k)};
end
```

在命令窗口中调用 mysize 函数, 结果如下:

```
>> [s,rows,cols] = mysize([1 2 3;3 4 5])
s =
     2     3
rows =
     2
cols =
     3
```

【例 2.8-4】 函数名作为 nargin 和 nargs 的输入参数。

如果将函数名作为 nargin 和 nargs 的输入参数, 则返回的是函数定义的输入参数和输出参数的个数, 即函数的形参的个数。

定义函数 myfunc, 保存为 myfunc.m 文件:

```
function out = myfunc(in1,in2,in3)
num1 = nargin;
num2 = nargs;
```

在命令窗口中调用 nargin 和 nargs, 将函数名作为它们的输入参数, 结果如下:

```
>> num1 = nargin('myfunc')
num1 =
     3
>> num2 = nargs('myfunc')
num2 =
     1
```

2.9 技巧 17: 执行字符串中包含的 MATLAB 表达式

2.9.1 技巧用途

在命令窗口或 M 文件中, 用户可以直接输入或编写包含 MATLAB 函数、变量、数值和操作符的表达式, MATLAB 会直接执行这些表达式; 但有时, 为了提高程序的自动化运行水平,

MATLAB 表达式可能包含在字符串中。例如:用户在程序的界面窗口的编辑框中输入 MATLAB 命令来控制程序的执行;自动生成了包含 load 命令的字符串,用来批量读取 MAT 文件的内容,等等。

本节将介绍在 MATLAB 中如何执行这些包含在字符串中的表达式。

2.9.2 技巧实现

MATLAB 提供了 eval、evalc 和 evalin 函数,可以执行字符串中包含的 MATLAB 表达式。

1. eval 函数

eval 函数的调用格式为:

- **eval(expression)**
- **[a1, a2, a3, ...] = eval('function(var)')**

其中,expression 为包含 MATLAB 表达式的字符串;function 为函数名,var 为函数的输入参数,a1、a2...为函数的输出参数。

2. evalc 函数

evalc 函数的调用格式为:

- **T = evalc(expression)**
- **[T, a1, a2, a3, ...] = evalc(expression)**

evalc 函数与 eval 函数功能基本相同,只是 evalc 函数会把输出到命令窗口中的内容捕获到输出参数 T 中(错误信息除外)。

3. evalin 函数

evalin 函数的功能是在指定的工作空间中执行 MATLAB 表达式。其调用格式为:

- **evalin(ws, expression)**
- **[a1, a2, a3, ...] = evalin(ws, expression)**

其中,ws 为指定工作空间的字符串,其值可以为 'base'(基本工作空间)或 'caller'(函数工作空间);expression 为包含 MATLAB 表达式的字符串。

evalin 函数的用法见“技巧 18: 实现函数 M 文件和基本工作空间中变量的相互调用”。

【例 2.9-1】 eval 和 evalc 函数的使用方法。

下面的代码中调用了 eval 函数,用来在窗口中绘制正弦曲线并添加图例说明。

```
x = 0:pi/50:2 * pi;
str = 'y = sin(x); plot(x,y); legend('sin(x)')';
eval(str);
```

程序运行结果如图 2.9-1 所示。

字符串中包含多条语句时,语句与语句之间要用逗号“,”或分号“;”来隔开。

若调用 evalc 函数,结果是一致的:

```
x = 0:pi/50:2 * pi;
str = 'y = sin(x), plot(x,y), legend('sin(x)');';
evalc(str);
```

【例 2.9-2】 在界面的 Edit 控件中输入函数表达式以及自变量的取值后,在窗口的坐标轴中绘制相应的图形。

① 利用 GUIDE 创建程序的界面窗口,在其中创建 Edit1 控件用于输入函数表达式,

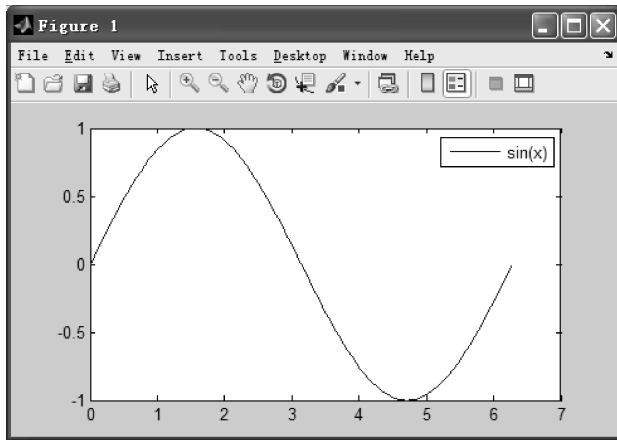


图 2.9-1 eval 执行后的程序界面

Edit2 控件用于输入自变量的数值, Axes 用于绘制图形, Pushbutton 控件用于执行绘图命令。

② 在 plot 按钮的 Callback 添加代码, 调用 eval 函数进行相应的处理:

```
% 取得 Edit1 的输入, 即函数表达式
func = get(handles.edit1, 'string');
% 取得 Edit2 的输入, 即自变量的值
var = get(handles.edit2, 'string');
% 执行 var 的字符串的内容, 得到变量的数值
varx = eval(var);
% 用 varx 的值替代 func 中的 x, 并计算 func 的数值
a = subs(func, 'x', varx);
% 绘图
axes(handles.axes1);
plot(varx, a, 'b');
```

程序运行结果如图 2.9-2 所示。

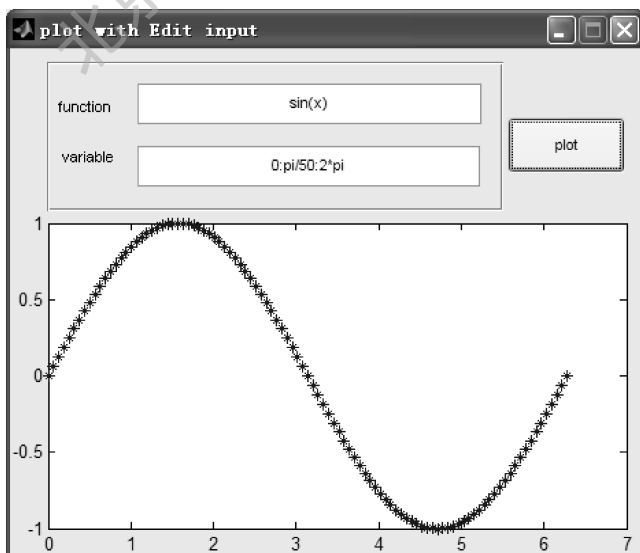


图 2.9-2 例 2.9-2 程序的运行效果图

2.10 技巧 18: 实现函数 M 文件和基本工作空间中变量的相互调用

2.10.1 技巧用途

在 MATLAB 中,存在两类工作空间:一类是基本工作空间(base workspace),一类是函数工作空间(caller workspace)。当运行纯脚本的 M 文件时,其运行过程中产生的变量都保存在基本工作空间中,用 save 和 load 命令就可以保存和加载这些变量。而对于函数 M 文件,运行过程中的变量是存储在函数工作空间中的,用户想查看其中的信息,只有在调试模式下才能看到,这两类工作空间中的变量是不能直接相互访问的。

在实际的应用中,有时需要在一个工作空间中访问另一个工作空间中的变量。例如:在使用 simulink 仿真的过程中,仿真的结果输出到了基本工作空间中,用户想通过图形用户界面来实时显示仿真的结果;在运行完图形用户界面程序后,把程序运行的结果输出到基本工作空间中,以供其他程序使用。

2.10.2 技巧实现

MATLAB 提供了 evalin 和 assignin 两个函数来实现函数工作空间和基本工作空间中的变量的相互使用。

1. evalin 函数

evalin 是在指定的工作空间中执行 MATLAB 命令。利用 evalin 函数可以在函数工作空间中访问基本工作空间中的变量。evalin 函数的调用格式为:

- evalin(ws, expression)
- [a1, a2, ...] = evalin(ws, expression)

其中,ws 为标识工作空间的字符串,其值可以为 'base'(基本工作空间)或 'caller'(函数工作空间);expression 为包含 MATLAB 表达式的字符串;a1、a2...为返回参数。

【例 2.10-1】 evalin 函数的使用方法。

① 在函数 M 文件中把基本工作空间中变量的值赋值给局部变量 v:

```
function use_evalin
% 假设 var 变量已经存在于 base 中,使用 evalin 得到 var 的值
v = evalin('base', 'var');
% 在命令窗口中显示变量 v 的值
disp(v);
```

② 查询基本工作空间中的所有变量名称,并把变量名称保存到 v 数组中:

```
function use_evalin
v = evalin('base', 'who');
disp(v);
```

运行结果如图 2.10-1 所示。

【例 2.10-2】 调用基本工作空间中的变量在图形窗口中绘制图形。

① 在基本工作空间中产生 x 和 y 两个变量:

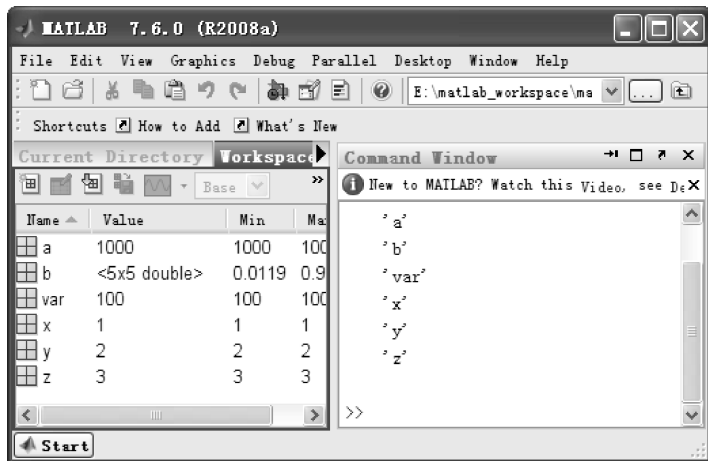


图 2.10-1 evalin 函数的执行结果

```
>> x=0:pi/50:2*pi;
>> y=sin(x);
```

② 在函数 M 文件中调用这些数据来在界面上绘制曲线：

```
function myfunction
% 创建界面窗口
hf = figure('units','normalized',...
    'name','自 base 中调用数据绘图',...
    'position',[0.4 0.3 0.4 0.3]);
% 创建坐标轴
haxes = axes('parent',hf,...
    'units','normalized',...
    'position',[0.1 0.1 0.8 0.8]);
% 取得 base 中的 x 和 y 变量的值,保存到 xdata 和 ydata 中
xdata1 = evalin('base','x');
ydata1 = evalin('base','y');
% 在指定的坐标轴中绘图
axes(haxes);
plot(xdata1,ydata1);
```

程序运行的结果如图 2.10-2 所示。

2. assignin 函数

assignin 将函数 M 文件中的变量的值“分配给”指定工作空间中的变量。函数的调用格式如下：

assignin(ws, 'var', val)

其中,ws 为标识工作空间的字符串,其值可以为 'base' 或 'caller';val 为函数 M 文件中的局部变量,'var' 为指定工作空间 ws 中的变量,若变量 'var' 在指定工作空间中不存在,则 MATLAB 自动创建该变量。

【例 2.10-3】 将函数中变量的值传递到基本工作空间。

在例 2.10-2 中的 myfunction 函数的末尾添加如下语句,即可以在基本工作空间中产生