

MATLAB应用与提高系列

MATLAB

与外部程序接口

苏金明 黄国明 刘 波 编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

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

MATLAB 应用与提高系列

MATLAB 与外部程序接口

苏金明 黄国明 刘 波 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书系统地介绍了 MATLAB 与外部程序的接口方法和技巧。全书共分 13 章。第 1 至第 3 章介绍了 MATLAB 与 DOS 程序的接口,其内容包括数据输入和输出的方法、MATLAB 编译器和 MATLAB 与 FORTRAN、C 的接口。第 4 章至第 9 章介绍了 MATLAB 与 Windows 程序的接口,其内容包括 MATLAB 与 Visual Basic、Visual C++、Excel 和 SPSS 的接口,以及最新推出的 COM 生成器和 Excel 生成器。第 10 章介绍了 MATLAB 与硬件接口。第 11 章至第 13 章介绍了运行时服务器、报表生成器以及提高代码运行效率的若干手段。书中列举了大量的实例,以便于读者理解和运用。

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 与外部程序接口/苏金明等编著. —北京:电子工业出版社, 2004.1

(MATLAB 应用与提高系列)

ISBN 7-5053-9296-4

I. M… II. 苏… III. 计算机辅助计算—软件包, MATLAB IV. TP391.75

中国版本图书馆 CIP 数据核字(2003)第 100105 号

责任编辑:王昌铭

印 刷:北京市增富印刷有限责任公司

出版发行:电子工业出版社 <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:19 字数:480 千字

版 次:2004 年 1 月第 1 版 2004 年 1 月第 1 次印刷

印 数:5000 册 定价:29.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话:(010) 68279077。质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前言

许多人很喜欢 MATLAB, 觉得它是一个很不错的软件, 能够给从事科学计算的同志带来更多的便利和可能性。

MATLAB 好, 首先表现在它的不断创新。MATLAB 的每次更新都能给人以惊喜, 要么是原有的功能得到扩充或提高, 要么是出现新的工具箱或实用工具, 要么是整体性能得到改进。DDE、OLE、ActiveX、COM 这样一些流行或曾经流行的标准和技术, 在 MATLAB 中都能得到及时的响应。其次, 它能满足个性化的需求。MATLAB 提供了几十个工具箱。利用这些工具箱, 可以解决不同领域的数学问题。而且由于 MATLAB 的可扩展性, 用户还可以编写自己领域的工具箱, 提高工作效率。除了工具箱以外, MATLAB 还提供了琳琅满目的实用工具。利用它们, 可以实现不同的功能。比如, 你用 MATLAB 开发了一套新算法, 是 M 文件, 但不想让别人看到源代码, 想保密, 于是考虑做成独立应用程序, 用 mcc 来做。如果 mcc 解决不了, 就用运行时服务器。如果想把该算法集成到 VB、VC 中去, 但不想重写代码, 可以用 COM 生成器把对应的 M 文件做成 COM 组件, 然后集成。所以, 只要你需要, 总有一款适合你。

MATLAB 是解释型语言, 运行速度比较慢。但从 MATLAB 6.5 开始, 它比较全面地提速了, 提速后的运行速度与向量化后的效果相当。虽然在某些情况下, 仍然需要通过循环向量化或预分配数组内存空间等技巧来加速运行, 但我们仍然能看到 MATLAB 所做的努力。MATLAB 提供了多种方法来加速运行。通过 Profiler 工具或 profile 函数, 可以获取每行代码的运行情况, 包括运行时间和调用次数等, 因而知道哪些语句行花费的时间最多, 可以集中精力进行改进。

作为一个专业的科学计算软件, MATLAB 的功能首先在于应用, 即应用现有函数和工具箱解决具体问题。在用的过程中, 用户会发现问题, 并逐渐有更高的要求。比如想开发自己的算法, 开发速度更快的应用, 或者想用 VC、VB 等开发更美观的界面等。所以, 用而优则开发, 这是很自然的追求, 也是大多数 MATLAB 学习者要走的路。

整套书共分 3 册, 分别偏重于入门、工具箱应用和接口。第一册分计算、绘图和编程 3 部份, 介绍了 MATLAB 的入门知识和技巧。第二册主要介绍我们所熟悉的统计、优化、偏微分方程数值解、样条、信号处理和曲线拟合等 6 个工具箱的最新版本。第三册介绍 MATLAB 与外部程序的接口, 包括 MATLAB 与 FORTRAN、C、Visual Basic、Visual C++、Excel、SPSS 以及硬件等的接口技术, 其中还介绍了 MATLAB 编译器、COM 生成器、Excel 生成器、运行时服务器、报表生成器、Excel Link、ImportWizard、Profiler 等工具的用法。

应该说, 除了受专业限制, 有一些工具箱没有介绍以外, MATLAB 所提供的大部分功能在这 3 本书中都有不同程度的阐述, 只要认真阅读, 终会有所收获。当你在学习的过程中, 感觉自己一天天变得更加充实, 因而内心充满喜悦的时候, 我们为你高兴!

关于这本书

本书主要介绍 MATLAB 与外部程序的接口方法和技巧,包括 MATLAB 与 DOS 程序的接口、MATLAB 与 Windows 程序的接口、MATLAB 与硬件的接口以及辅助工具等内容。适合于 MATLAB 中、高级读者阅读和参考。

第 1 章至第 3 章介绍 MATLAB 与 DOS 程序的接口。主要介绍了 MATLAB 数据输入、输出的方法、MATLAB 编译器和 MATLAB 与 FORTRAN、C 的接口。用 Import Wizard 工具导入数据是比较简便快捷的方法,另外还介绍了 MAT 数据的读写方法。MATLAB 编译器的新版本是 3.0,利用它,可以创建 MEX 文件和独立的应用程序,结合 COM 生成器和 Excel 生成器,还可以创建 COM 组件和插件。MATLAB 与 FORTRAN 和 C 的接口主要通过 MEX 文件和引擎函数库来实现。

第 4 章至第 9 章介绍 MATLAB 与 Windows 程序的接口,包括 MATLAB 与 Visual Basic、Visual C++、Excel 和 SPSS 的接口,以及两个实用工具 COM 生成器和 Excel 生成器。COM 生成器和 Excel 生成器是 MATLAB 的新工具,分别可以把 M 文件、MEX 文件做成 COM 组件和 Excel 插件。COM 组件可以在 VB、VC 等支持它的语言环境中被引用,从而可以实现进程内的集成。这部分还重点介绍了基于 ActiveX 自动化的接口技术,并列举了若干实例。在学习这些内容时,读者(特别是对那些花了很多精力来编写和调试数值算法的读者)可以从另一个层面上理解 MATLAB 的强大。

第 10 章结合实例介绍了 MATLAB 与硬件接口的方法。

第 11 章至第 13 章介绍了 MATLAB 的两个实用工具,即运行时服务器和报表生成器。另外还介绍了提高 MATLAB 运行效率的各种方法。运行时服务器弥补了 MATLAB 编译器的不足,可以在更宽松的条件下生成可执行程序。使用报表生成器,可以根据需要,将操作代码和图表结果等以报表的形式输出。计算速度慢是 MATLAB 的弱点, MATLAB 为此想了很多办法。本书最后一章介绍了提高代码运行效率的若干手段。其中重点介绍了可以监测代码运行状况的 Profiler 工具和 profile 函数。

本书的第 7 章由黄国明编写,第 10 章由刘波编写,其余各章由苏金明编写。刘宏、李攀峰等提供了帮助,苏华惠和刘玉珊做了大量的录入工作,一并表示感谢!

由于能力有限,书中错误和不足之处在所难免,谨请读者批评指正!有任何问题,请通过电子邮件与我们联系。

苏金明 s_jm@263.net.cn

黄国明 elara@163.net

刘波 sclibo@mail.sc.cninfo.net

编著者

2003 年 6 月

目 录

第 1 章 数据输入和输出	(1)
1.1 MATLAB 数据输入和输出的方法	(1)
1.1.1 向 MATLAB 输入数据	(1)
1.1.2 从 MATLAB 提取数据	(1)
1.2 读写 MAT 文件	(2)
1.2.1 MAT 文件接口库	(2)
1.2.2 创建 C MAT 文件示例	(3)
1.2.3 读取 C MAT 文件示例	(7)
1.2.4 创建 FORTRAN MAT 文件示例	(11)
1.2.5 读取 FORTRAN MAT 文件示例	(15)
1.2.6 编译和链接 MAT 文件	(17)
1.3 Import Wizard	(18)
1.3.1 输入 MAT 数据	(18)
1.3.2 输入 Excel 数据	(20)
1.3.3 输入文本数据	(21)
1.3.4 输入图像数据	(21)
第 2 章 编译器	(23)
2.1 概述	(23)
2.1.1 使用编译器	(23)
2.1.2 MATLAB 编译器族	(24)
2.2 安装和注册	(26)
2.2.1 系统需求	(26)
2.2.2 编译器选项文件	(27)
2.2.3 MATLAB 编译器	(27)
2.2.4 MEX 确认	(28)
2.2.5 MATLAB 编译器确认	(30)
2.3 独立应用	(31)
2.3.1 MEX 文件与独立应用之间的区别	(31)
2.3.2 创建独立的 C/C++ 应用	(32)
2.3.3 在 PC 上生成独立应用	(32)
2.3.4 发布独立应用程序	(37)
2.3.5 生成共享库	(37)
2.3.6 生成 COM 对象	(38)

2.3.7	创建 Excel 插件	(38)
2.4	控制代码的生成	(39)
2.4.1	概述	(39)
2.4.2	编译私有的方法函数	(41)
2.4.3	生成的头文件	(42)
2.4.4	内部接口函数	(44)
第 3 章	MATLAB 与 C 和 FORTRAN 接口	(47)
3.1	MEX 文件	(47)
3.1.1	关于 MEX 文件	(47)
3.1.2	使用数据类型	(48)
3.1.3	MEX 文件的组成	(49)
3.1.4	MEX 文件的参数	(50)
3.1.5	自动生成 MEX 文件	(50)
3.1.6	手工生成 MEX 文件	(51)
3.1.7	定制 MEX 文件	(55)
3.2	引擎函数	(57)
3.2.1	MATLAB 引擎库	(57)
3.2.2	调用引擎函数示例	(57)
3.3	其他混合编程方法	(64)
3.3.1	一个简单的例子	(64)
3.3.2	C 调用经过编译的 M 文件——高级示例	(67)
3.3.3	从 C MEX 文件中调用 MATLAB 函数	(70)
第 4 章	COM 生成器 (COM Builder)	(72)
4.1	创建 COM 生成器组件	(72)
4.1.1	创建工程	(72)
4.1.2	管理 M 文件和 MEX 文件	(73)
4.1.3	生成组件	(74)
4.1.4	打包和分发组件	(74)
4.1.5	组件生成的内部过程	(75)
4.1.6	数据转换	(75)
4.1.7	调用约定	(76)
4.1.8	COM 生成器组件的兼容性	(76)
4.2	利用 COM 生成器组件编程	(76)
4.2.1	给 COM 生成器对象添加方法和属性	(76)
4.2.2	给 COM 生成器对象添加事件	(78)
4.2.3	创建类实例	(80)
4.2.4	调用类实例的方法	(82)
4.2.5	处理 varargin 和 varargout 变量	(82)
4.2.6	在调用方法的过程中控制错误	(82)

4.2.7	修改标记	(83)
4.3	应用举例	(84)
4.3.1	创建 M 文件	(84)
4.3.2	创建工程	(84)
4.3.3	生成工程	(85)
4.3.4	创建 Visual Basic 工程	(85)
4.3.5	创建用户界面	(85)
4.3.6	测试应用	(88)
4.3.7	组件打包	(88)
第 5 章	Excel 生成器 (Excel Builder)	(90)
5.1	创建 Excel 生成器插件	(90)
5.1.1	创建工程	(90)
5.1.2	管理 M 文件和 MEX 文件	(91)
5.1.3	生成组件	(92)
5.1.4	测试 VBA 模块	(92)
5.1.5	打包和发布组件	(93)
5.2	用 Excel 生成器组件编程	(93)
5.2.1	用 Excel 初始化生成器库	(94)
5.2.2	创建类的实例	(94)
5.2.3	调用类实例的方法	(96)
5.2.4	处理 varargin 和 varargout 变量	(97)
5.2.5	在调用方法的过程中控制错误	(98)
5.2.6	修改标记	(98)
5.3	魔方示例	(101)
5.3.1	一个输入的情况	(101)
5.3.2	使用多个文件和变量	(103)
5.4	谱分析示例	(107)
5.4.1	创建组件	(108)
5.4.2	将组件集成到 VBA 中	(109)
5.4.3	创建图形用户界面	(111)
5.4.4	保存和测试插件	(116)
5.4.5	打包组件	(117)
5.5	工具箱	(118)
5.5.1	MWUtil 类	(118)
5.5.2	MWFlags 类	(122)
5.5.3	MWStruct 类	(124)
5.5.4	MWField 类	(127)
5.5.5	MWComplex 类	(127)
5.5.6	MWSparse 类	(128)

5.5.7	MWArg 类	(130)
5.5.8	3 个枚举类型	(130)
第 6 章	MATLAB 与 Visual Basic 接口	(132)
6.1	DDE (动态数据交换) 编程	(132)
6.1.1	DDE 的概念和技巧	(132)
6.1.2	MATLAB 作为服务器端	(133)
6.1.3	MATLAB 作为客户端	(135)
6.2	MATLAB 调用 VB 组件	(136)
6.2.1	在 MATLAB 中创建 COM 自动化控件	(136)
6.2.2	对象属性	(136)
6.2.3	操作对象的方法	(142)
6.2.4	对象事件	(144)
6.2.5	确认对象	(148)
6.2.6	保存和删除工作	(149)
6.2.7	MATLAB 作为自动化客户端示例	(150)
6.2.8	使用 COM 集合	(154)
6.2.9	转换数据	(155)
6.3	VB 调用 MATLAB (组件)	(156)
6.3.1	MATLAB COM 自动化方法和属性	(156)
6.3.2	MATLAB 作为自动化服务器端示例	(157)
第 7 章	MATLAB 与 Visual C++ 接口	(162)
7.1	Visual C++ 调用 MATLAB 引擎	(162)
7.1.1	引擎库函数	(163)
7.1.2	阵列的创建与访问	(165)
7.1.3	在 Visual C++ 中调用 MATLAB 引擎	(168)
7.2	MATLAB 可执行程序	(169)
7.2.1	接口函数 mexFunction	(170)
7.2.2	在 Visual C++ 中实现 MATLAB 可执行程序	(170)
7.3	VC 调用 MATLAB 数学库	(174)
7.3.1	MATLAB C++ 数学函数库	(174)
7.3.2	使用 MATLAB 数学函数库的环境设置	(175)
7.3.3	在 Visual C++ 中调用 MATLAB 数学函数库	(175)
7.3.4	VC++ 环境下的 MATLAB 开发	(177)
第 8 章	MATLAB 与 Excel 接口	(179)
8.1	自动化链接	(179)
8.1.1	MATLAB 作为自动化客户端	(179)
8.1.2	MATLAB 作为自动化服务器端	(180)
8.2	Excel Link 插件	(181)
8.2.1	概述	(181)

8.2.2	安装和操作 Excel Link 插件	(182)
8.2.3	Excel Link 的函数	(185)
8.2.4	技巧和提示	(186)
8.2.5	Excel Link 使用实例	(188)
第 9 章	MATLAB 与 SPSS 接口	(191)
9.1	SPSS 软件	(191)
9.2	SPSS 中的对象	(191)
9.3	MATLAB 调用 SPSS	(193)
9.4	SPSS 调用 MATLAB	(195)
第 10 章	MATLAB 与硬件接口	(199)
10.1	MATLAB 串行接口介绍	(199)
10.2	利用串行口进行通信	(199)
10.2.1	一个简单的例子	(200)
10.2.2	通信步骤及有关函数介绍	(201)
10.3	应用实例	(205)
第 11 章	运行时服务器 (Runtime Server)	(208)
11.1	概述	(208)
11.1.1	编译器的局限和约束	(208)
11.1.2	运行时服务器	(209)
11.1.3	MATLAB 运行时服务器的特点	(210)
11.2	安装运行时服务器	(210)
11.3	开发运行时应用程序应注意的问题	(210)
11.3.1	防止在命令窗口中输入和输出	(210)
11.3.2	有选择地使默认菜单选项不可用	(211)
11.3.3	提供一个退出应用程序的方法	(213)
11.3.4	捕捉错误	(214)
11.4	开发一个 MATLAB 运行时 GUI 应用程序	(216)
11.4.1	组织文件、管理启动任务	(216)
11.4.2	编译 GUI 应用程序	(218)
11.4.3	测试和调试应用程序	(220)
11.4.4	运行时 GUI 应用程序示例	(222)
11.5	开发 MATLAB 运行时引擎应用程序	(227)
11.5.1	组织文件和管理启动任务	(227)
11.5.2	编译应用程序	(229)
11.5.3	测试和调试应用程序	(229)
11.5.4	ActiveX 自动化示例	(230)
11.5.5	引擎 API 示例	(236)
11.6	发布 MATLAB 运行时应用程序	(240)
11.6.1	创建启动窗口	(240)

11.6.2	组织文件	(240)
11.6.3	自动打包	(240)
11.6.4	手工打包	(241)
11.6.5	自动生成安装器	(241)
11.6.6	手工创建安装器	(241)
第 12 章	报表生成器 (Report Generator)	(243)
12.1	概述	(243)
12.1.1	什么是报表生成器	(243)
12.1.2	报表生成器的组成	(243)
12.1.3	演示——生成一个报表	(244)
12.2	生成报表	(246)
12.2.1	使用命令行生成报表	(247)
12.2.2	使用安装文件列表创建报表	(247)
12.2.3	使用安装文件编辑器创建报表	(247)
12.3	编辑组件的属性	(254)
12.3.1	概述	(254)
12.3.2	图像“For”循环组件	(255)
12.4	生成和编辑安装文件	(259)
12.4.1	组件	(259)
12.4.2	安装文件编辑器	(259)
12.4.3	安装文件大纲	(259)
12.4.4	添加组件	(259)
12.4.5	激活组件	(260)
12.4.6	移动组件	(260)
12.4.7	剪切、复制和粘贴组件	(261)
12.5	创建自己的组件	(261)
12.5.1	启动组件创建大师	(262)
12.5.2	输入组件分类信息	(262)
12.5.3	创建组件名	(263)
12.5.4	创建组件属性	(264)
12.5.5	创建组件的方法	(266)
12.5.6	回顾所有的组件信息	(268)
12.5.7	创建和确认组件	(268)
第 13 章	改善 MATLAB 的运行效率	(269)
13.1	改善运行的技巧	(269)
13.1.1	分析程序的运行状况	(269)
13.1.2	循环向量化	(270)
13.1.3	数组的内存预分配	(271)
13.1.4	加速运行的其他方法	(272)

13.2	MATLAB 6.5 以上版本对运行效率的改进	(272)
13.2.1	MATLAB 中能加速和不能加速的元素	(272)
13.2.2	运行 MATLAB 时应该避免的问题	(274)
13.2.3	加速运行演示	(275)
13.3	程序运行情况监测——Profiler	(276)
13.3.1	Profiler 的运行环境	(276)
13.3.2	使用 Profiler	(277)
13.3.3	监测一个图形用户界面的运行情况	(277)
13.3.4	从命令窗口监测语句	(278)
13.3.5	监测综述报表	(278)
13.3.6	监测详细报表	(278)
13.3.7	利用 Profiler 报表中的信息	(282)
13.3.8	改变 Profiler 的字体	(282)
13.4	使用 Profile 函数	(283)
13.4.1	Profile 函数语法和使用步骤	(283)
13.4.2	Profile 函数使用演示	(283)
13.4.3	使用 Profiler 结果结构示例	(286)
13.5	有效使用内存	(288)
13.5.1	内存管理函数	(288)
13.5.2	驻留内存的方法	(288)

第 1 章 数据输入和输出

MATLAB 中,数据的输入和输出可以有多种方法。本章先简要介绍这些方法,然后重点介绍 MAT 文件的读写和 Import Wizard 工具的使用。

1.1 MATLAB 数据输入和输出的方法

1.1.1 向 MATLAB 输入数据

从其他程序向 MATLAB 输入数据可以用几种方法。输入数据的最好办法与数据量、数据模式等有关,可以根据需要从下面选择一种。

(1) 以列表的形式输入数据。这种方法适合于数据量很少(少于 10~15 个)的情况,数据可以用中括号“[]”括起来。

(2) 以 M 文件的形式创建数据。使用文本编辑器创建一个 M 文件,它以元素列表的形式输入数据。这种方法在数据不以计算机可读的形式出现,而必须手工键入时有用。该方法本质上与上一种方法相同,其好处是可以用编辑器改变和修正数据。

(3) 从 ASCII 文本文件中装入数据。文本文件以 ASCII 形式保存,可以用一般的文本编辑器编辑 ASCII 文本文件。文本文件可以用 load 命令直接读入到 MATLAB 中,结果是创建一个与文件名相同的变量名。

(4) 用 MATLAB I/O 函数读取数据。可以用 fopen, fread 和 MATLAB 的其他低级 I/O 函数读取数据。本法对于从具有自己数据格式的应用程序中载入数据很有用。

(5) 编写 MEX 文件来读取数据。如果过程已经可以用于从其他应用程序中读取数据文件,则可以考虑选用本法。

(6) 编写程序,转换数据。可以通过编写 C 或 FORTRAN 程序来把数据转换为 MAT 文件格式,然后用 load 命令把 MAT 文件读入 MATLAB。

1.1.2 从 MATLAB 提取数据

从 MATLAB 提取数据有以下一些方法。

(1) 创建一个日记文件(diary file)。对于小矩阵,使用 diary 命令创建日记文件并显示变量,把它们反馈到该文件,以后可以用文本编辑器操作这个日记文件,其输出包括用到的 MATLAB 命令。

(2) 使用 save 命令。

(3) 用带-ascii 选项的 save 命令把数据保存为 ASCII 形式,如

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

```
save temp.dat A -ascii
```

创建一个 ASCII 文件 temp.dat, 内容为:

1.3889088e-001 2.7218792e-001 4.4509643e-001
2.0276522e-001 1.9881427e-001 9.3181458e-001
1.9872174e-001 1.5273927e-002 4.6599434e-001
6.0379248e-001 7.4678568e-001 4.1864947e-001

-ascii 选项只支持二维 double 数组和字符数组，不支持多维数组、单元数组和结构。

(4) 使用 MATLAB I/O 函数。用 fopen, fwrite 和其他低级 I/O 函数将数据编写为特殊格式。本法适用于将数据文件编写为其他应用程序需要的格式。

(5) 开发一个 MEX 文件来写入数据。该方法适用于已有可用的程序，并利用它可以将数据写为其他程序需要的格式的情况。

(6) 把数据转换为 MAT 文件。可用 save 命令将数据写入 MAT 文件，然后通过编写 C 或 FORTRAN 程序将 MAT 文件转换为特殊的格式。

1.2 读写 MAT 文件

MATLAB 中的 save 命令将当前内存中的 MATLAB 数组保存为一种称为 MAT 文件的二进制文件。之所以称为 MAT 文件，是因为这些文件的扩展名是.mat。load 命令进行 save 命令的相反操作。它把磁盘上 MAT 文件中的 MATLAB 数组读入到 MATLAB 工作空间。MAT 文件中可以包含一个或多个 MATLAB 5 以上版本所支持的数据类型，包括字符串、矩阵、多维数组、结构和单元数组等。MATLAB 把这些数据作为连续的字节流写入磁盘。

1.2.1 MAT 文件接口库

MAT 文件接口库包含一系列读写 MAT 文件的过程。可以从自己的 C 和 FORTRAN 程序中调用这些过程，这些过程的名称都有前缀 mat，表 1-1 和表 1-2 分别列出了 C 和 FORTRAN 可用的 MAT 函数和它们的功能。

表 1-1 C MAT 函数

MAT 函数	功 能
matOpen	打开一个 MAT 文件
matClose	关闭一个 MAT 文件
matGetDir	从 MAT 文件获取一个 MATLAB 数组的列表
matGetFp	获取一个 ANSI C 文件指针给 MAT 文件
matGetVariable	从 MAT 文件获取一个 MATLAB 数组
matPutVariable	写一个 MATLAB 数组到 MAT 文件
matGetNextVariable	从 MAT 文件中获取下一个 MATLAB 数组
matDeleteVariable	从 MAT 文件中删除一个 MATLAB 数组
matPutVariableAsGlobal	将一个 MATLAB 数组放到一个 MAT 文件中，这样，使用 load 命令可以将它放到全局工作空间中
matGetVariableInfo	从 MAT 文件中获取一个 MATLAB 数组头的信息
matGetNextVariableInfo	从 MAT 文件中获取下一个数组头的信息

表 1-2 FORTRAN MAT 函数

MAT 函数	功 能
matOpen	打开一个 MAT 文件
matClose	关闭一个 MAT 文件
matGetDir	从 MAT 文件获取一个 MATLAB 数组的列表
matGetVariable	从 MAT 文件获取一个命名的 MATLAB 数组
matGetVariableInfo	从 MAT 文件中获取命名 MATLAB 数组头的信息
matPutVariable	将 MATLAB 放入一个 MAT 文件
matPutVariableAsGlobal	将 MATLAB 数组放入一个 MAT 文件
matGetNextVariable	从 MAT 文件中获取下一个序列 MATLAB 数组
matGetNextVariableInfo	从 MAT 文件中获取下一个序列 MATLAB 数组的头的信息
matDeleteVariable	从 MAT 文件中删除一个 MATLAB 数组

1.2.2 创建 C MAT 文件示例

本例是 MATLAB 自带的示例程序，演示如何用库过程创建一个 MAT 文件，它可以载入到 MATLAB 中。本程序还演示如何核对读写失败时调用的 MAT 函数的返回值。该文件位于 <matlab>\extern\eng_mat 目录下。

```

/*
 * 调用语法:
 *   matcreat
 * 创建一个可以在 MATLAB 中装载的 MAT 文件
 * 本程序用到下面一些 MAT 函数:
 *   matClose
 *   matGetVariable
 *   matOpen
 *   matPutVariable
 *   matPutVariableAsGlobal
 *
 * Copyright 1984-2000 The MathWorks, Inc.
 * $Revision: 1.9 $
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "mat.h"

#define BUFSIZE 256

int main() {
    MATFile *pmat;

```

```

mxArray *pa1, *pa2, *pa3;
double data[9] = { 1.0, 4.0, 7.0, 2.0, 5.0, 8.0, 3.0, 6.0, 9.0 };
const char *file = "mattest.mat";
char str[BUFSIZE];
int status;

printf("Creating file %s...\n\n", file);
pmat = matOpen(file, "w");
if (pmat == NULL) {
    printf("Error creating file %s\n", file);
    printf("(Do you have write permission in this directory?)\n");
    return(EXIT_FAILURE);
}

pa1 = mxCreateDoubleMatrix(3,3,mxREAL);
if (pa1 == NULL) {
    printf("%s : Out of memory on line %d\n", __FILE__,
        __LINE__);
    printf("Unable to create mxArray.\n");
    return(EXIT_FAILURE);
}

pa2 = mxCreateDoubleMatrix(3,3,mxREAL);
if (pa2 == NULL) {
    printf("%s : Out of memory on line %d\n", __FILE__,
        __LINE__);
    printf("Unable to create mxArray.\n");
    return(EXIT_FAILURE);
}

memcpy((void *)(mxGetPr(pa2)), (void *)data, sizeof(data));

pa3 = mxCreateString("MATLAB: the language of technical
                    computing");
if (pa3 == NULL) {
    printf("%s : Out of memory on line %d\n", __FILE__,
        __LINE__);
    printf("Unable to create string mxArray.\n");
    return(EXIT_FAILURE);
}

status = matPutVariable(pmat, "LocalDouble", pa1);
if (status != 0) {
    printf("%s : Error using matPutVariable on line %d\n",

```

```

        __FILE__, __LINE__);
    return(EXIT_FAILURE);
}

status = matPutVariableAsGlobal(pmat, "GlobalDouble", pa2);
if (status != 0) {
    printf("Error using matPutVariableAsGlobal\n");
    return(EXIT_FAILURE);
}

status = matPutVariable(pmat, "LocalString", pa3);
if (status != 0) {
    printf("%s : Error using matPutVariable on line %d\n",
        __FILE__, __LINE__);
    return(EXIT_FAILURE);
}

/*
 * 用 matPutVariable 函数覆盖一个已经存在于 MAT 文件中的数组
 */
memcpy((void*)(mxGetPr(pa1)), (void*)data, sizeof(data));
status = matPutVariable(pmat, "LocalDouble", pa1);
if (status != 0) {
    printf("%s : Error using matPutVariable on line %d\n",
        __FILE__, __LINE__);
    return(EXIT_FAILURE);
}

/*清除*/
mxDestroyArray(pa1);
mxDestroyArray(pa2);
mxDestroyArray(pa3);

if (matClose(pmat) != 0) {
    printf("Error closing file %s\n", file);
    return(EXIT_FAILURE);
}

/*重新打开文件并用 matGetVariable 函数确认它的内容*/
pmat = matOpen(file, "r");
if (pmat == NULL) {
    printf("Error reopening file %s\n", file);
    return(EXIT_FAILURE);
}

```