薛定宇教授大讲堂（卷Ⅰ）

MATLAB 程序设计

薛定宇◎著
Xue Dingyu

Professor Xue Dingyu's Lecture Hall（Volume Ⅰ）
MATLAB Programming

清華大學出版社
北京

内 容 简 介

MATLAB 语言是进行科学计算的利器。本书系统地论述了 MATLAB 的功能及使用 MATLAB 语言编程的方法。本书内容包括 MATLAB 语言的常用数据结构和语句结构、矩阵的代数运算、超越函数的计算方法与数据处理的方法、MATLAB 语言的流程控制结构与应用、MATLAB 函数编写与调试，以及 MATLAB 的科学可视化方法。此外，本书还介绍了 MATLAB 语言的接口设计、面向对象的程序设计方法与图形用户界面设计方法等。

本书可作为一般读者学习和掌握 MATLAB 语言的工具书，也可作为高等学校理工科各类专业本科生与研究生学习计算机数学语言（MATLAB）的教材。

图书在版编目（CIP）数据

薛定宇教授大讲堂. 卷Ⅰ, MATLAB 程序设计/薛定宇著. —北京：清华大学出版社，2019
ISBN 978-7-302-51868-6

Ⅰ. ①薛… Ⅱ. ①薛… Ⅲ. ①Matlab 软件-程序设计-高等学校-教学参考资料 Ⅳ. ①TP317

中国版本图书馆 CIP 数据核字(2018)第 285091 号

责任编辑：盛东亮
封面设计：李召霞
责任校对：梁　毅
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：186mm×240mm　　**印　张**：16.5　　**字　　数**：315 千字
版　　次：2019 年 7 月第 1 版　　**印　　次**：2019 年 7 月第 1 次印刷
定　　价：69.00 元

产品编号：078635-01

前　言

PREFACE

科学运算问题是每个理工科学生和科技工作者在课程学习、科学研究与工程实践中常常会遇到的问题，不容回避。对于非纯数学专业的学生和研究者而言，从底层全面学习相关数学问题的求解方法并非一件简单的事情，也不易得出复杂问题的解。所以，利用当前最先进的计算机工具，高效、准确、创造性地求解科学运算问题是一种行之有效的方法，尤其能够满足理工科人士的需求。

作者曾试图在同一部著作中叙述各个数学分支典型问题的直接求解方法，通过清华大学出版社出版了《高等应用数学问题的MATLAB求解》。该书从2004年出版之后多次重印再版，并于2018年出版了第4版，还配套发布了全新的MOOC课程[①]，一直受到广泛的关注与欢迎。首次MOOC开课的选课人数接近14000人，教材内容也被数万篇期刊文章和学位论文引用。

从作者首次使用MATLAB语言算起，已经有30年的时间了，通过相关领域的研究、思考与一线教学实践，积累了大量的实践经验资料。这些不可能在一部著作中全部介绍，所以与清华大学出版社策划与编写了这套“薛定宇教授大讲堂”系列著作，系统深入地介绍基于MATLAB语言与工具的科学运算问题的求解方法。

本系列著作不是原来版本的简单改版，通过十余年的经验和资料积累，全面贯穿“再认识”的思想写作此书，深度融合科学运算数学知识与基于MATLAB的直接求解方法与技巧，力图更好地诠释计算机工具在每个数学分支的作用，帮助读者以不同的思维与视角了解工程数学问题的求解方法，创造性地得出问题的解。

本系列著作卷I可以作为学习MATLAB入门知识的教材与参考书，也为读者深入学习与熟练掌握MATLAB语言编程技巧，深度理解科学运算领域MATLAB的应用奠定一个坚实的基础。后续每一卷试图对应一个数学专题或一门数学课程进行展开。整套系列著作的写作贯穿“计算思维”的思想，深度探讨该数学专题的问题求解方法。本系列著作既适合于学完相应的数学课程之后，深入学习利用计算

① MOOC网址：https://www.icourse163.org/learn/NEU-1002660001

机工具的科学运算问题求解方法与技巧，也可作为相应数学课程同步学习的伴侣，在学习相应课程理论知识的同时，侧重学习基于计算机的数学问题求解方法，从另一个角度观察、审视数学课程所学的内容，扩大知识面，更好地学习、理解并实践相应的数学课程。

本书是系列著作的卷Ⅰ。本书系统介绍MATLAB语言编程方法，首先介绍MATLAB语言的常用数据结构和语句结构，然后介绍矩阵的代数运算、超越函数的计算方法与数据处理的方法，并介绍MATLAB的流程控制结构与应用、MATLAB函数编写与调试等编程技巧、MATLAB的科学可视化方法。本书还介绍MATLAB语言的接口设计、面向对象的程序设计方法与图形用户界面设计方法等，旨在为读者继续学习科学运算或其他领域的知识奠定较好的基础。

值此系列著作付梓之际，衷心感谢相濡以沫的妻子杨军教授，她数十年如一日的无私关怀是我坚持研究、教学与写作工作的巨大动力。

薛定宇

2019年5月

目　录

CONTENTS

第1章　计算机数学语言概述 …… 1
1.1　数学问题计算机求解概述 …… 1
1.1.1　为什么要学习计算机数学语言 …… 1
1.1.2　数学问题的解析解与数值解 …… 4
1.1.3　数学运算问题软件包发展概述 …… 5
1.1.4　常规计算机语言的局限性 …… 7
1.2　计算机数学语言简介 …… 8
1.2.1　计算机数学语言的出现 …… 8
1.2.2　有代表性的计算机数学语言 …… 9
1.3　科学运算问题的三步求解方法 …… 10
本章习题 …… 12
第2章　MATLAB语言程序设计基础 …… 13
2.1　MATLAB命令窗口与基本命令 …… 14
2.1.1　变量名命名规则 …… 14
2.1.2　保留的常数 …… 15
2.1.3　显示格式的设置 …… 16
2.1.4　底层操作系统命令 …… 16
2.1.5　MATLAB的工作环境设置 …… 17
2.1.6　MATLAB的工作空间与管理 …… 18
2.1.7　MATLAB的其他辅助工具 …… 18
2.2　常用数据结构 …… 19
2.2.1　数值型数据 …… 19
2.2.2　符号型数据 …… 20
2.2.3　任意符号型矩阵的生成 …… 22
2.2.4　符号型函数 …… 22
2.2.5　整型变量与逻辑变量 …… 22
2.2.6　数据结构类型的识别 …… 23

2.2.7　矩阵的维数与长度 …… 23
2.3　字符串数据结构 …… 24
2.3.1　一般字符串的表示 …… 24
2.3.2　字符串的处理方法 …… 24
2.3.3　字符串的转换与读写方法 …… 26
2.3.4　字符串命令的执行 …… 27
2.3.5　MuPAD接口函数的编写 …… 27
2.4　其他常用数据结构 …… 28
2.4.1　多维数组 …… 28
2.4.2　单元数组 …… 29
2.4.3　表格数据 …… 30
2.4.4　结构体 …… 32
2.4.5　其他数据结构 …… 33
2.5　MATLAB的基本语句结构 …… 33
2.5.1　直接赋值语句 …… 33
2.5.2　函数调用语句 …… 34
2.5.3　多样的函数调用机制 …… 34
2.5.4　冒号表达式 …… 34
2.5.5　子矩阵的提取 …… 35
2.5.6　等间距行向量的生成 …… 36
2.6　数据文件的读取与存储 …… 36
2.6.1　数据文件的读取与存储命令 …… 36
2.6.2　文件读写的底层方法 …… 37
2.6.3　Excel文件的读取与存储 …… 38
本章习题 …… 39
第3章　基本数学运算 …… 42
3.1　矩阵的代数运算 …… 42
3.1.1　矩阵的转置、翻转与旋转 …… 42
3.1.2　矩阵的加减乘除运算 …… 44
3.1.3　复数矩阵及其变换 …… 45
3.1.4　矩阵的乘方与开方 …… 45
3.1.5　矩阵的点运算 …… 47
3.2　矩阵的逻辑运算与比较运算 …… 47
3.2.1　矩阵的逻辑运算 …… 47
3.2.2　矩阵的比较运算 …… 48

3.2.3 矩阵元素的查询命令 …… 48
3.2.4 属性判定语句 …… 49
3.3 超越函数的计算 …… 49
3.3.1 指数与对数函数的计算 …… 50
3.3.2 三角函数的计算 …… 50
3.3.3 反三角函数的计算 …… 52
3.3.4 矩阵的超越函数 …… 52
3.4 符号表达式的化简与变换 …… 54
3.4.1 多项式的运算 …… 54
3.4.2 三角函数的变换与化简 …… 55
3.4.3 符号表达式的化简 …… 55
3.4.4 符号表达式的变量替换 …… 56
3.4.5 符号运算结果的转换 …… 56
3.5 基本数据运算 …… 57
3.5.1 数据的取整与有理化运算 …… 57
3.5.2 向量的排序、最大值与最小值 …… 58
3.5.3 数据的均值、方差与标准差 …… 59
3.5.4 质因数与质因式 …… 60
3.5.5 排列与组合 …… 61
本章习题 …… 62

第4章 MATLAB语言的流程结构 …… 64
4.1 循环结构 …… 64
4.1.1 for循环结构 …… 64
4.1.2 while循环结构 …… 66
4.1.3 迭代方法的循环实现 …… 67
4.1.4 循环结构的辅助语句 …… 69
4.1.5 向量化编程实现 …… 69
4.2 条件转移结构 …… 71
4.2.1 简单的条件转移结构 …… 71
4.2.2 条件转移结构的一般形式 …… 72
4.2.3 分段函数的向量化表示 …… 74
4.3 开关结构 …… 75
4.4 试探结构 …… 77
本章习题 …… 78

第5章　函数编写与调试 ····· 80
5.1　MATLAB的脚本程序 ····· 80
5.2　MATLAB语言函数的基本结构 ····· 81
5.2.1　函数的基本结构 ····· 81
5.2.2　函数名的命令规则 ····· 83
5.2.3　函数编写举例 ····· 83
5.3　函数编写的技巧 ····· 86
5.3.1　递归调用 ····· 86
5.3.2　可变输入输出个数的处理 ····· 87
5.3.3　输入变元的容错处理 ····· 89
5.3.4　全局变量 ····· 89
5.3.5　存取MATLAB工作空间中的变量 ····· 90
5.3.6　匿名函数与inline函数 ····· 91
5.3.7　子函数与私有函数 ····· 93
5.4　MATLAB程序的调试 ····· 93
5.4.1　MATLAB程序的跟踪调试 ····· 93
5.4.2　伪代码与代码保密处理 ····· 96
5.5　MATLAB实时编辑器 ····· 96
5.5.1　实时文档编辑界面 ····· 97
5.5.2　建立一个简单的文档 ····· 97
5.5.3　嵌入代码的运行 ····· 98
5.5.4　在实时编辑器中嵌入其他对象 ····· 99
5.5.5　实时编辑文档的输出 ····· 101
本章习题 ····· 101
第6章　二维图形绘制 ····· 105
6.1　二维曲线的绘制 ····· 105
6.1.1　二元数据的曲线绘制 ····· 105
6.1.2　数学函数的曲线绘制 ····· 108
6.1.3　分段函数的曲线绘制 ····· 108
6.1.4　二维图形的标题处理 ····· 109
6.1.5　多纵轴曲线的绘制 ····· 111
6.2　图形修饰 ····· 112
6.2.1　利用界面工具的修饰 ····· 113
6.2.2　LaTeX支持的修饰命令 ····· 115
6.2.3　数学公式叠印与宏包设计 ····· 116

6.3 其他二维图形绘制语句 ····· 117
6.3.1 极坐标曲线的绘制 ····· 117
6.3.2 离散信号的图形表示 ····· 118
6.3.3 直方图与饼图 ····· 120
6.3.4 填充图 ····· 122
6.3.5 对数坐标图 ····· 123
6.3.6 误差限图 ····· 124
6.3.7 动态轨迹显示 ····· 124
6.3.8 二维动画的显示 ····· 124
6.4 图形窗口的分割 ····· 125
6.4.1 规范分割 ····· 125
6.4.2 任意分割 ····· 126
6.5 隐函数绘制及应用 ····· 128
6.6 图像的显示与简单处理 ····· 130
6.6.1 图像的输入 ····· 130
6.6.2 图像的编辑与显示 ····· 131
6.6.3 颜色空间转换 ····· 132
6.6.4 边缘检测 ····· 132
6.6.5 直方图均衡化 ····· 133
6.7 MATLAB 图形的输出方法 ····· 134
6.7.1 图形输出菜单与应用 ····· 134
6.7.2 图形输出命令 ····· 135
本章习题 ····· 136
第 7 章 三维图形表示 ····· 138
7.1 三维曲线绘制 ····· 138
7.1.1 三维曲线绘制命令 ····· 138
7.1.2 已知数学函数的三维曲线绘制 ····· 139
7.1.3 三维填充图 ····· 140
7.1.4 三维直方图与饼图 ····· 140
7.1.5 条带图 ····· 142
7.2 三维曲面绘制 ····· 144
7.2.1 网格图与表面图 ····· 144
7.2.2 表面图的阴影与光照 ····· 147
7.2.3 图像文件的三维表面图 ····· 149
7.2.4 已知函数的表面图 ····· 150

7.2.5　散点数据的表面图绘制 …… 151
7.3　三维图形视角设置 …… 152
7.3.1　视角的定义 …… 152
7.3.2　三视图的设置 …… 153
7.3.3　任意视角的设置 …… 153
7.4　其他三维绘图 …… 154
7.4.1　等高线 …… 154
7.4.2　矢量图 …… 155
7.4.3　三元隐函数的绘图 …… 156
7.4.4　参数方程的表面图 …… 158
7.4.5　复变函数的三维表面图 …… 158
7.4.6　球面与柱面 …… 159
7.4.7　Voronoi图与Delaunay剖分 …… 161
7.5　三维图形的特殊处理 …… 163
7.5.1　三维曲面的旋转 …… 163
7.5.2　坐标轴变换的三维曲面 …… 164
7.5.3　三维图形的剪切 …… 165
7.5.4　三维表面图贴面处理 …… 166
7.6　四维图形绘制 …… 167
7.6.1　切片图 …… 167
7.6.2　体视化界面 …… 168
7.6.3　三维动画的制作与播放 …… 169
本章习题 …… 171
第8章　MATLAB语言与其他语言的接口 …… 173
8.1　C语言环境下提供的MATLAB变量格式及函数概述 …… 174
8.1.1　编译程序的环境设置 …… 174
8.1.2　Mex下的数据结构 …… 175
8.1.3　Mex文件的结构 …… 176
8.1.4　Mex文件的编写方法与步骤 …… 179
8.2　不同数据结构的Mex处理 …… 180
8.2.1　不同类型输入输出变元的处理 …… 181
8.2.2　字符串变量的读写 …… 181
8.2.3　多维数组的处理 …… 183
8.2.4　单元数组的处理 …… 184
8.2.5　MAT文件的读写方法 …… 185

8.3 C程序中直接调用MATLAB函数 ······ 187
8.4 MATLAB函数的独立程序转换 ······ 191
本章习题 ······ 192
第9章 面向对象程序设计基础 ······ 193
9.1 面向对象编程的基本概念 ······ 193
9.1.1 类与对象 ······ 193
9.1.2 类与对象数据结构 ······ 194
9.2 类的设计 ······ 195
9.2.1 类的设计方法 ······ 195
9.2.2 类的定义与输入 ······ 196
9.2.3 类的显示 ······ 197
9.3 重载函数的编写 ······ 198
9.3.1 加法的重载函数编写 ······ 198
9.3.2 合并同类项的化简函数 ······ 199
9.3.3 减法重载函数 ······ 200
9.3.4 乘法重载函数 ······ 200
9.3.5 乘方运算重载函数 ······ 202
9.3.6 域的赋值与提取 ······ 203
9.4 类的继承与扩展 ······ 203
9.4.1 扩展类的定义与显示 ······ 204
9.4.2 ftf对象的连接重载函数 ······ 205
9.4.3 分数阶传递函数的频域分析 ······ 207
本章习题 ······ 208
第10章 MATLAB的图形用户界面设计技术 ······ 209
10.1 MATLAB语言图形界面编程基础 ······ 209
10.1.1 MATLAB图形界面中各对象的关系 ······ 209
10.1.2 窗口对象及属性设置 ······ 210
10.1.3 窗口的常用属性 ······ 211
10.1.4 对象属性的读取与修改 ······ 213
10.1.5 简易对话框 ······ 215
10.1.6 标准对话框及其调用 ······ 216
10.2 MATLAB图形界面设计基本控件 ······ 219
10.2.1 MATLAB支持的基本控件 ······ 219
10.2.2 控件的常用属性 ······ 221
10.2.3 控件句柄的获取 ······ 221

10.3　图形用户界面设计工具Guide ····· 222
10.4　图形用户界面的高级技术 ····· 231
10.4.1　菜单系统的设计 ····· 231
10.4.2　工具栏设计 ····· 232
10.4.3　ActiveX控件的嵌入与编程 ····· 234
10.5　工具箱的集成与发布 ····· 235
本章习题 ····· 235
参考文献 ····· 237
MATLAB函数名索引 ····· 239
术语索引 ····· 245

第1章

计算机数学语言概述

1.1 数学问题计算机求解概述

数学问题是科学研究中不可避免的问题。研究者通常将自己研究的问题用数学建模的方法建立数学模型，然后通过求解数学模型的方法获得所研究问题的解。建立数学模型需要所研究领域的专业知识，而有了数学模型则可以采用本书介绍的通用数值方法或解析方法去直接求解。本章将首先对计算机数学语言进行简单介绍，通过实例介绍为什么需要学习计算机数学语言，然后介绍计算机数学语言和数学工具的发展简况。

1.1.1 为什么要学习计算机数学语言

求解科学运算问题时手工推导当然是有用的，但并不是所有的问题都是能手工推导的，故需要由计算机完成相应的任务。用计算机求解的方式有两种，其一是用成型的数值分析算法、数值软件包与手工编程相结合的求解方法，其二是采用国际上有影响力的专门的计算机语言求解问题，这类语言包括MATLAB、Mathematica[1]、Maple[2]等，本书统称之为“计算机数学语言”。顾名思义，用数值方法只能求解数值计算的问题，至于像公式推导等数学问题，例如求解$x^3+bx+c=0$方程的解，在b、c不是给定数值时，数值分析的方式是没有用的，必须使用计算机数学语言求解。

本书将涉及问题的求解方法称为“数学运算”，以区别于传统意义下的“数学计算”，因为后者往往对应于数学问题的数值求解方法。本书介绍的内容还尽可能地包括解析求解方法，如果解析解不存在则将介绍数值解方法。

在系统介绍本书的内容之前，先介绍几个例子，读者可以思考其中提出的问题，从中体会学习本书的必要性。相应的MATLAB语句后面还将详细介绍。

例1-1 考虑一个“奥数”类题目：2017^{2017}的最后一位数是什么？

解 如果不借助计算机工具，数学家用纸笔能计算出来的只有这个数的个位了。事

实上，这样的解在现实生活中没有任何意义和价值，因为一个很昂贵的物品人们不会纠结其售价的个位数是1还是9，还是其他的什么数，人们更感兴趣的是这个数有多少位，其最高位是几，每位数是什么等。而对这些问题的求解，数学家是无能为力的，只能借助于专用的计算机工具求解。借助计算机数学语言，可以直接得出该数的精确值是$390657\cdots8177$，共有6666位数，该数可以充满本书的两页多。

例1-2　大学的高等数学课程介绍了微分与积分的概念和数学推导方法，实际应用中也可能遇到高阶导数的问题。已知$f(x)=\sin x/(x^2+4x+3)$这样的简单函数，如何求解出$\mathrm{d}^4f(x)/\mathrm{d}x^4$？

解　这样的问题用手工推导是可行的，由高等数学的知识可以先得出函数的一阶导数$\mathrm{d}f(t)/\mathrm{d}x$，对结果求导得出函数的二阶导数，对结果再求导得出三阶导数，继续进一步求导就能求出所需的$\mathrm{d}^4f(x)/\mathrm{d}x^4$，重复此方法还能求出更高阶的导数。这个过程比较机械，适合用计算机实现，用现有的计算机数学语言可以由一行语句求解问题：

```
>> syms x; f=sin(x)/(x^2+4*x+3); y=diff(f,x,4) %描述原函数并直接求导
```

上述语句得出的结果为

$$\begin{aligned}\frac{\mathrm{d}^4f(t)}{\mathrm{d}x^4}=&\frac{\sin x}{x^2+4x+3}+4\frac{(2x+4)\cos x}{(x^2+4x+3)^2}-12\frac{(2x+4)^2\sin x}{(x^2+4x+3)^3}\\&+12\frac{\sin x}{(x^2+4x+3)^2}-24\frac{(2x+4)^3\cos x}{(x^2+4x+3)^4}+48\frac{(2x+4)\cos x}{(x^2+4x+3)^3}\\&+24\frac{(2x+4)^4\sin x}{(x^2+4x+3)^5}-72\frac{(2x+4)^2\sin x}{(x^2+4x+3)^4}+24\frac{\sin x}{(x^2+4x+3)^3}\end{aligned}$$

显然，若依赖手工推导，得出这样的结果需要很繁杂、细致的工作，稍有不慎就可能得出错误的结果，所以应该将这样的问题推给计算机去求解。实践表明，利用著名的MATLAB语言，在4 s内就可以精确地求出$\mathrm{d}^{100}f(x)/\mathrm{d}x^{100}$。

例1-3　还记得线性代数课程中介绍的求高阶矩阵的行列式的方法吗？

解　线性代数课程介绍的通用方法是代数余子式的方法，可以将一个n阶矩阵的行列式问题化简成n个$n-1$阶行列式问题，而$n-1$阶又可以化简为$n-2$阶的问题，这样用递归的方法可以最终化简成一阶矩阵的行列式求解问题，而该问题是有解析解的，就是该一阶矩阵本身，所以数学家可以得出结论，任意阶矩阵的行列式都可以直接求解出解析解。

事实上，这样的结论忽略了计算复杂度问题，这样的算法计算量很大，高达$(n-1)(n+1)!+n$，例如$n=25$时，运算次数为9.679×10^{27}，相当于在每秒12.54亿亿次的神威太湖之光（2017年世界上最快的超级计算机）上计算204年，虽然用代数余子式的方法可以求解，但求解是不现实的。其实在某些领域中甚至需要求解成百上千阶矩阵的问题，所以用代数余子式的方法是不可行的。

数值分析中提供了求解行列式问题的各种算法，但有时传统的方法对某些矩阵会

得出错误的结果，特别是接近奇异的矩阵。考虑Hilbert矩阵

$$\boldsymbol{H}=\begin{bmatrix} 1 & 1/2 & 1/3 & \cdots & 1/n \\ 1/2 & 1/3 & 1/4 & \cdots & 1/(n+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/(n+1) & 1/(n+2) & \cdots & 1/(2n-1) \end{bmatrix}$$

并假设$n=80$，用数值分析方法或软件很容易得出$\det(\boldsymbol{H})=0$的不精确结果，从而导致矩阵奇异这样的错误结论。事实上，用计算机数学语言MATLAB很容易在1.79 s内得出该行列式的精确解为

$$\det(\boldsymbol{H})=\frac{1}{\underbrace{99030101466993477878867678410192 51\cdots 00000}_{\text{全部3789位，因排版的限制省略了中间的数字}}}\approx 1.00979\times 10^{-3790}$$

求解一般高阶矩阵求逆问题需要计算机数学语言，对特殊的矩阵问题更需要这样的语言，以免得出错误的结果。本例采用的MATLAB语句为

$\boldsymbol{H}$=`sym(hilb(80)); det(`$\boldsymbol{H}$`)`

例1-4 你会求解下面两个方程吗？

$$\begin{cases} x+y=35 \\ 2x+4y=94 \end{cases} \qquad \begin{cases} x+3y^3+2z^2=1/2 \\ x^2+3y+z^3=2 \\ x^3+2z+2y^2=2/4 \end{cases}$$

解 第一个方程是鸡兔同笼问题，即使不使用计算机工具人们也可以直接求解。如果使用MATLAB语言，可以用下面的命令直接求解该方程：

```
>> syms x y; [x0,y0]=vpasolve(x+y==35,2*x+4*y==94)
```

有了MATLAB这样的高水平计算机语言，求解第二个方程与鸡兔同笼问题一样简单，只须将方程用符号表达式表示出来，就可以由vpasolve()函数直接求解，得出方程的全部27个根，将根代入方程，则误差范数达到10^{-34}级。第二个方程的代码如下：

```
>> syms x y z;  %用符号表达式表示方程，更利于检验
   f1(x,y,z)=x+3*y^3+2*z^2-1/2; f2(x,y,z)=x^2+3*y+z^3-2; %描述方程
   f3(x,y,z)=x^3+2*z+2*y^2-2/4; [x0,y0,z0]=vpasolve(f1,f2,f3)
   size(x0), norm([f1(x0,y0,z0) f2(x0,y0,z0) f3(x0,y0,z0)])
```

例1-5 试求解下面的线性规划问题：

$$\min \qquad (-2x_1-x_2-4x_3-3x_4-x_5)$$

$$\boldsymbol{x}\ \text{s.t.}\begin{cases} 2x_2+x_3+4x_4+2x_5\leqslant 54 \\ 3x_1+4x_2+5x_3-x_4-x_5\leqslant 62 \\ x_1,x_2\geqslant 0, x_3\geqslant 3.32, x_4\geqslant 0.678, x_5\geqslant 2.57 \end{cases}$$

解 求解线性规划问题需要最优化类课程的基础知识。因为上述问题是有约束最优化问题，不能用高等数学中令目标函数导数为0、得出若干方程再用求解方程的方式求解最优化问题，而必须用线性规划中介绍的算法求解，例如使用如下代码：

```
>> clear; P.f=[-2 -1 -4 -3 -1]; P.Aineq=[0 2 1 4 2; 3 4 5 -1 -1];
   P.Bineq=[54 62]; P.lb=[0;0;3.32;0.678;2.57]; P.solver='linprog';
   P.options=optimset; x=linprog(P) %描述线性规划问题并求解
```

得出所需的最优解$x_1=19.7850, x_2=0, x_3=3.3200, x_4=11.3850, x_5=2.5700$。

这样的求解借助于数值分析或最优化方法等课程介绍的数值算法就可以容易地实现。但如果再添加约束,例如需要得出该最优化问题的整数解,原来的问题就变成了整数规划问题。很少有相关书籍、软件能直接求解这样的问题。而利用计算机数学语言可以求出该整数规划问题的解为$x_1=19, x_2=0, x_3=4, x_4=10, x_5=5$。

例1-6 许多课程要用到的应用数学分支,如积分变换、复变函数、微分方程、数据插值与拟合、概率论与数理统计、数值分析等,课程考试之后还记得其中问题的求解方法吗?

例1-7 现代科学技术在其发展过程中,催生了若干新的数学分支,如模糊集合与粗糙集合、人工神经网络等,如果不借助于计算机工具,利用其中任何一个分支解决实际问题都是个耗时并困难的任务。因为首先要了解相关领域的来龙去脉,弄清算法并将算法用计算机语言正确地实现。然而,利用这些新分支的数学工具解决某些特定的数学问题却是较容易的,因为可以借助前人已经开发好的工具和框架。

很多专门的课程,如电路、电子技术、电力电子技术、电机与拖动、自动控制原理等,在介绍原理与方法时一般采用简单的例子,刻意回避高阶的或复杂的例子。究其原因,是当时缺少高水平计算机数学语言甚至是数值分析技术的支持,所以在这些课程中很多方法不一定适合于复杂的问题求解。在实际研究中遇到稍复杂一点的问题时,只靠手工推导的方法是得不出精确结果的,所以需要特殊的专业软件或语言来解决问题,而计算机数学语言,如MATLAB语言,通常可以较好地解决相关问题,并对研究的问题提供一个全新的视角。

从上面的例子可以看出,解决数学问题用手工推导的方法虽然有时可行,但对很多复杂问题是不现实或不可靠的,用传统数值分析课程甚至成型的软件包得出的结果有时也是错误的,故需要学习计算机数学语言,以便更好地解决以后学习和研究中遇到的问题。

1.1.2 数学问题的解析解与数值解

现代科学与工程的发展离不开数学。数学家们感兴趣的问题和其他科学家、工程技术人员所关注的问题是不同的。数学家往往对数学问题的解析解,或称闭式解(closed-form solution)和解的存在性、唯一性的严格证明感兴趣,而工程技术人员一般对解是多少、有多少解等问题更关心。换句话说,能用某种方法获得问题的解则是工程技术人员更关心的问题。而获得这样解的最直接方法就是数值解法。

数学问题解析解不存在的情况是非常常见的。例如，定积分 $\frac{2}{\sqrt{\pi}}\int_0^a \mathrm{e}^{-x^2}\mathrm{d}x$ 在上限为有穷时就没有解析解。数学家可以发明新的函数 $\mathrm{erf}(a)$ 定义这样的解，但解的值到底多大却不是一目了然的。所以在这样的情况下，要想获得积分的值，就必须采用数值解技术。

再如，圆周率 π 的值本身就没有解析解，中国古代的数学家、天文学家祖冲之（公元 429– 公元 500 年）早在公元 480 年就算定了该值在 3.1415926 和 3.1415927 之间。在一般科学与工程应用中，取这样的值就能保证较高的精度，而对于粗略估算来说，使用公元前 20 世纪古埃及人的 3.16045 或公元前 250 年左右阿基米德（公元前 287– 公元前 212 年）的 3.1418 都未尝不可，而没有必要非去追求不存在的解析解。所以在这样的问题上，数值解法的优势就显示出来了。

数学问题的数值解法已经成功地应用于各个领域。例如，在力学领域，常用有限元法求解偏微分方程；在航空、航天与自动控制领域，经常用到数值线性代数与常微分方程的数值解法等解决实际问题；在工程与非工程系统的计算机仿真中，核心问题的求解也需要用到各种差分方程、常微分方程的数值解法；在高科技的数字信号处理领域，离散的快速 Fourier 变换（FFT）已经成为其不可或缺的工具。在科学工程研究中能掌握一个或多个实用的计算工具，无疑会为研究者提供解决实际问题的强有力手段。

1.1.3 数学运算问题软件包发展概述

数字计算机的出现给数值计算技术的研究注入了新的活力。在数值计算技术的早期发展中，出现了一些著名的数学软件包，如美国的基于特征值的软件包 EISPACK [3,4] 和线性代数软件包 LINPACK [5]，英国牛津数值算法研究组（Numerical Algorithm Group，NAG）开发的 NAG 软件包 [6] 及享有盛誉的著作（文献 [7]）中给出的程序集（这里称 Numerical Recipes 软件包）等，这些都是在国际上广泛流行的、有着较高声望的软件包。

美国的 EISPACK 和 LINPACK 是基于矩阵特征值和奇异值解决线性代数问题的专用软件包。限于当时的计算机发展状况，这些软件包大都是由 Fortran 语言编写的源程序组成的。例如，若想求出 N 阶实矩阵 $\boldsymbol{A}$ 的全部特征值（用 $\boldsymbol{W}_{\mathrm{R}}$、$\boldsymbol{W}_{\mathrm{I}}$ 数组分别表示其实部和虚部）和对应的特征向量矩阵 $\boldsymbol{Z}$，则 EISPACK 软件包给出的子程序建议调用路径为

```
CALL BALANC(NM,N,A,IS1,IS2,FV1)
CALL ELMHES(NM,N,IS1,IS2,A,IV1)
CALL ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
```

```
CALL HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
IF (IERR.EQ.0) GOTO 99999
CALL BALBAK(NM,N,IS1,IS2,FV1,N,Z)
```

由上面的叙述可以看出，要求矩阵的特征值和特征向量，首先要对一些数组和变量依据EISPACK的格式作出定义和赋值，并编写出主程序，再经过编译和连接过程，形成可执行文件，最后才能得出所需的结果。

NAG软件包和Numerical Recipes软件包则包括了各种各样数学问题的数值解法，二者中NAG的功能尤其强大。NAG的子程序都是以字母加数字编号的形式命名的，非专业人员很难找到适合自己问题的子程序，更不用说能保证以正确的格式去调用这些子程序了。这些程序包使用起来极其复杂，每个函数有很多变元，很难保证使用者不出错。

Numerical Recipes软件包是一个在国际上广泛应用的软件包，子程序有C、Fortran和Pascal等版本，适合于科学研究者和工程技术人员直接应用。该书的程序包由200多个高效、实用的子程序构成，这些子程序一般都有较好的数值特性，比较可靠，为各国的研究者所信赖。

具有Fortran和C等高级计算机语言知识的读者可能已经注意到，如果用它们进行程序设计，尤其当涉及矩阵运算或画图时，编程会很麻烦。例如，若想求解一个线性代数方程，用户得首先编写一个主程序，然后编写一个子程序读入各个矩阵的元素，之后再编写一个子程序，求解相应的方程（如使用Gauss消去法），最后输出计算结果。如果选择的计算子程序不是很可靠，则所得的计算结果往往会出现问题。如果没有标准的子程序可以调用，则用户往往要将自己编好的子程序逐条地输入计算机，然后进行调试，最后进行计算。这样一个简单的问题往往需要用户编写100条左右的源程序，输入与调试程序也是很费事的，并无法保证所输入的程序完全可靠。求解线性方程组这样一个简单的功能需要100条源程序，其他复杂的功能往往要求有更多条语句，如采用双步QR法求取矩阵特征值的子程序则需要500多条源程序，其中任何一条语句有问题或调用不当（如数组维数不匹配）都可能导致错误结果的出现。

尽管如此，数学软件包仍在继续发展，其发展方向是采用国际上最先进的数值算法，提供更高效、更稳定、更快速、更可靠的数学软件包。例如，在线性代数计算领域，LAPACK[8]已经成为当前最有影响的软件包，但它们的目的似乎已经不再是为一般用户提供解决问题的方法，而是为数学软件提供底层的支持。新版的MATLAB语言以及自由软件Scilab[9]等著名的计算机数学语言已经放弃了一直使用的LINPACK和EISPACK，而采用LAPACK为其底层支持软件包。

一些数学的专门分支也出现了相关的数学程序库，支持Fortran、C++等语言直接调用与编程，MATLAB可以通过特殊接口的形式直接调用这些程序。在互联网上同样有大量的MATLAB语言和其他计算机数学语言的数学工具箱，所以遇到典型问题的数学求解时，可以直接利用相关的工具箱求解，因为其中大部分工具箱毕竟还是在相应领域有影响的专家编写的，得出的结果往往比外行自己查阅书籍、论文编写底层程序的可信度要高得多。

1.1.4 常规计算机语言的局限性

人们有时习惯用其他计算机语言（如C和Fortran）解决科学计算问题。毋庸置疑，这些计算机语言在数学与工程问题的求解中起过很大的作用，而且它们曾经是实现MATLAB这类高级语言的底层计算机语言。然而，对于一般科学研究者来说，利用C这类语言去求解数学问题是远远不够的。首先，一般程序设计者无法编写出符号运算和公式推导类程序，只能编写数值计算程序；其次，数值分析类教科书中介绍的数值算法往往不是求解实际数学问题的最好方法；除了上述局限性外，若采用底层计算机语言编程，由于程序冗长难以验证，即使得出结果也不敢相信该结果。所以应该采用更可靠、更简洁的专门计算机数学语言来进行科学研究，因为这样可以将研究者从烦琐的底层编程中解放出来，更好地把握求解的问题，避免“只见树木、不见森林”的认识偏差，这无疑是受到更多研究者认可的解决问题方式。

例 1-8 已知Fibonacci序列的前两个元素为$a_1 = a_2 = 1$，随后的元素可以由$a_k = a_{k-1} + a_{k-2}, k = 3, 4, \cdots$递推地计算出来。试用计算机列出该序列的前100项。

解 C语言在编写程序之前需要首先给变量选择数据类型，因为此问题需要的是整数，所以很自然地会选择int或long表示序列的元素，若选择数据类型为int，则可以编写出如下C程序：

```
#include <stdio.h>
main()
{ int a1, a2, a3, i;
  a1=1; a2=1; printf("%d  %d  ",a1,a2);
  for (i=3; i<=100; i++)
  { a3=a1+a2; printf("%d  ",a3); a1=a2; a2=a3;
}}
```

只用了上面几条语句，问题就看似轻易地被解决了。然而该程序是错误的！运行该程序会发现，该序列显示到第24项突然会出现负数，而再显示下几项会发现时正时负。显然，上面的程序出了问题。问题出在int整型变量的选择上，因为该数据类型能表示数值的范围为$(-32767, 32767)$，超出此范围则会导致错误的结果。即使采用long整型数据定义，也只能保留31位二进制数值，即保留9位十进制有效数字，超过这个数仍然

返回负值。可见，采用C语言，如果某些细节考虑不周，则可能得出完全错误的结论。所以说C这类语言得出的结果有时不大令人信服。用MATLAB语言则不必考虑这些烦琐的问题，可以直接编写下面的底层程序：

```
>> a=[1 1]; for i=3:100, a(i)=a(i-1)+a(i-2); end; a(end) %循环计算
```

另外，由于long整型数据只能保持9位有效数字，而double型只能保留15位有效数字，如果得出的结果超出此范围，则精度将存在局限性。采用MATLAB的符号运算则可以避免这类问题，只需将第1个语句修改成a=sym([1,1])就可以得出a_{100}的值为354224848179261915075，甚至用类似的语句都能在24 s内得出a_{5000}的全部1045位有效数字，该结果是采用任何数值计算语言都无法得出的。

例1-9　试编写出两个矩阵$\boldsymbol{A}$和$\boldsymbol{B}$相乘的C语言通用程序。

解　如果$\boldsymbol{A}$为$n\times p$矩阵，$\boldsymbol{B}$为$p\times m$矩阵，则由线性代数理论，可以得出$\boldsymbol{C}$矩阵，其矩阵元素为

$$c_{ij}=\sum_{k=1}^{p}a_{ik}b_{kj},\quad i=1,2,\cdots,n,\ j=1,2,\cdots,m$$

分析上面的算法，容易编写出C语言程序，其核心部分为三重循环结构：

```
for (i=0; i<n; i++){for (j=0; j<m; j++){
  c[i][j]=0; for (k=0; k<p; k++) c[i][j]+=a[i][k]*b[k][j];}}
```

看起来这样一个通用程序通过这几条语句就解决了。事实不然，这个程序有个致命的漏洞，就是没考虑两个矩阵是不是可乘。如果$\boldsymbol{A}$矩阵的列数等于$\boldsymbol{B}$矩阵的行数，则两个矩阵可乘，所以很自然地想到应该加一个判定语句：

if $\boldsymbol{A}$的列数不等于$\boldsymbol{B}$的行数，给出错误信息

其实这样的判定可能引入新的漏洞，因为若$\boldsymbol{A}$或$\boldsymbol{B}$为标量，则$\boldsymbol{A}$和$\boldsymbol{B}$无条件可乘，而增加上面的if语句反而会给出错误信息。这样在原来的基础上还应该增加判定$\boldsymbol{A}$或$\boldsymbol{B}$是否为标量的语句。

其实即使考虑了上面所有的内容，程序还不是通用的程序，因为并未考虑矩阵为复数矩阵的情况。这也需要特殊的语句处理。

从这个例子可见，用C这类语言处理某类标准问题时需要特别细心，否则难免会有漏洞，致使程序出现错误，或其通用性受到限制，甚至可能得出有误导性的结果。在MATLAB语言中则没有必要考虑这样的琐碎问题，因为$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的积由$\boldsymbol{C}$=$\boldsymbol{A}$*$\boldsymbol{B}$直接求取，若可乘则得出正确结果，如不可乘则给出出现问题的原因。

1.2　计算机数学语言简介

1.2.1　计算机数学语言的出现

MATLAB语言为数学问题的计算机求解，特别是控制系统的仿真领域起到了巨大的推动作用。1978年美国New Mexico大学计算机科学系的主任Cleve Moler

教授认为用当时最先进的EISPACK和LINPACK软件包求解线性代数问题过程过于烦琐，所以构思一个名为MATLAB（Matrix Laboratory，矩阵实验室）的交互式计算机语言。该语言一开始为免费版本，1984年The MathWorks公司（现名MathWorks公司）成立，并推出了1.0版。该语言的出现正赶上控制界基于状态空间的控制理论蓬勃发展的阶段，所以很快就引起了控制界学者的关注，出现了用MATLAB语言编写的控制系统工具箱，在控制界产生了巨大的影响，成为控制界的标准计算机语言。后来由于控制界及相关领域提出的各种各样要求，MATLAB语言得到了持续发展，使得其功能越来越强大。可以说，MATLAB语言是由计算数学专家首创的，但是由控制界学者“捧红”的新型计算机语言。目前大部分工具箱都是面向自动控制和相关学科的，但随着MATLAB语言的不断发展，目前也在其他领域广泛使用。稍后出现的Mathematica及Maple等语言也是当前应用广泛的计算机数学语言。

此外，法国计算机科学与控制研究院INRIA开发的自由软件Scilab也可以解决部分常用的数学问题，其最显著的特色是完全免费且源代码全部公开，但在求解数学问题的功能上尚无法和MATLAB等计算机数学语言媲美。

1.2.2 有代表性的计算机数学语言

目前在国际上有三种计算机数学语言最有影响力：MathWorks公司的MATLAB语言、Wolfram Research公司的Mathematica语言和Waterloo Maplesoft公司的Maple语言。这三种语言各有特色，其中MATLAB长于数值运算，其程序结构类似于其他计算机语言，因而编程很方便。Mathematica和Maple有强大的解析运算和数学公式推导、定理证明的功能，相应的数值计算能力比MATLAB要弱，这两种语言更适合于纯数学领域的计算机求解。此外，德国的MuPAD[10]也是较好的计算机数学语言。

与Mathematica及Maple相比，MATLAB语言的数值运算功能是很出色的。除此之外，更有一个另两种语言不可替代的优势，就是MATLAB语言在各种各样的领域均有领域专家编写的工具箱，可以高效、可靠地解决各种各样的问题。MATLAB符号运算工具箱采用MuPAD为其符号运算引擎，有些符号运算的能力较以前版本有所改善，也有很多功能不如早期版本。从整体水平来看，MATLAB的符号运算能力在纯数学问题求解中不如Mathematica与Maple，从一般应用数学运算领域看，差距则不是太大，可以利用MATLAB的符号运算功能很好地解决一般的科学运算问题。

本丛书采用MATLAB语言为主要计算机数学语言，系统地介绍其在数学及一

般科学运算问题求解中的应用。掌握了该语言将提高读者求解数学问题的能力，提高数学水平，拓广知识面，使得原来看似高深的应用数学问题的实际求解变得轻而易举。

本书是丛书的第一卷，将系统地介绍MATLAB语言的编程方法与技巧。丛书的其他卷次将利用MATLAB语言为主要工具，深入、系统地介绍每个数学分支的相关内容，可以从不同的角度重新学习与实践各个数学分支问题的直接求解，还可以利用MATLAB这样的工具对相应的数学分支开展创造性的研究，得出别人没有观测到的结果。

1.3　科学运算问题的三步求解方法

本书倡导一种科学运算问题的三步求解方法[11]，这三个步骤分别是“是什么”“如何描述”和“求解”。在“是什么”步骤中，侧重于数学问题的物理解释和含义。即使学生没有学习过相关的数学分支，也可能通过简单的语言叙述大致理解问题的物理含义。在“如何描述”步骤中，用户应该知道如何将数学问题用MATLAB描述出来。在“求解”步骤中，用户应该知道调用哪个MATLAB函数将原始数学问题直接求解出来。如果有现成的MATLAB函数，则应该调用相应函数直接求解出问题；如果没有现成函数，则编写出通用程序得出问题的解。

例1-10　用例1-5中的线性规划问题的求解演示三步求解方法。

$$\boldsymbol{x}\ \text{s.t.}\ \begin{cases}\min & -2x_1-x_2-4x_3-3x_4-x_5\\ & 2x_2+x_3+4x_4+2x_5\leqslant 54\\ & 3x_1+4x_2+5x_3-x_4-x_5\leqslant 62\\ & x_1,x_2\geqslant 0,\ x_3\geqslant 3.32,\ x_4\geqslant 0.678,\ x_5\geqslant 2.57\end{cases}$$

解　有的读者很可能没有系统地学习过最优化等相关的课程。不过不要紧，即使没有学习过相关的理论知识，也可以通过下面的三步求解方法得出问题的解。

（1）“**是什么**”。本步先理解每个数学问题的物理含义。在这个具体问题中，读者可以将原始问题从字面上理解为：在满足下面联立不等式约束

$$\begin{cases}2x_2+x_3+4x_4+2x_5\leqslant 54\\ 3x_1+4x_2+5x_3-x_4-x_5\leqslant 62\\ x_1,x_2\geqslant 0,\ x_3\geqslant 3.32,\ x_4\geqslant 0.678,\ x_5\geqslant 2.57\end{cases}$$

的前提下，怎么发现一组决策变量x_i的值，能使得目标函数$f(\boldsymbol{x})=-2x_1-x_2-4x_3-3x_4-x_5$的值为最小。所以，即使没有学习过最优化课程的读者也不难从字面上理解该问题的数学公式。

（2）“**如何描述**”。读者将学会如何将数学问题用MATLAB函数描述出来，在例1-5的代码中，用下面的方法建立一个变量P描述整个数学问题：

```
>> clear; P.f=[-2 -1 -4 -3 -1];                    %目标函数
```

```
P.Aineq=[0 2 1 4 2; 3 4 5 -1 -1]; P.Bineq=[54 62]; %约束条件
P.solver='linprog'; P.lb=[0;0;3.32;0.678;2.57];     %下边界
P.options=optimset; %将整个线性规划问题用结构体变量P描述出来
```

(3) “求解”。调用线性规划求解函数 linprog() 直接求解问题,得出问题的解。

```
>> x=linprog(P) %调用linprog()函数求解数学问题
```

例 1-11 人工神经网络是近年来应用较广泛的智能类数学工具,擅长于数据拟合与分类等运算。假设由下面的语句生成样本点数据:

```
>> x=0:0.1:pi; y=exp(-x).*sin(2*x+2);
```

试利用样本点建立人工神经网络模型,并绘制函数曲线。

解 如果不想花时间或没有时间去学习人工神经网络的系统理论,只想使用神经网络解决本例的数据拟合问题,则可以考虑利用前面介绍的三步求解方法,花几分钟了解神经网络基本概念与使用方法,就能利用人工神经网络求解数据拟合问题。回到前面提及的三步求解方法:

(1) 什么是人工神经网络。没有必要去了解人工神经网络的技术细节,只须将人工神经网络看作一个信息处理单元,它接受若干路信号进行处理,得出输出信号。

(2) 将神经网络的数学模型建立起来,选择 fitnet() 函数建立空白神经网络模型,用 train() 训练神经网络,得出可用的人工神经网络模型,如图 1-1 所示。

```
>> net=fitnet(5); net=train(net,x,y), view(net)
```

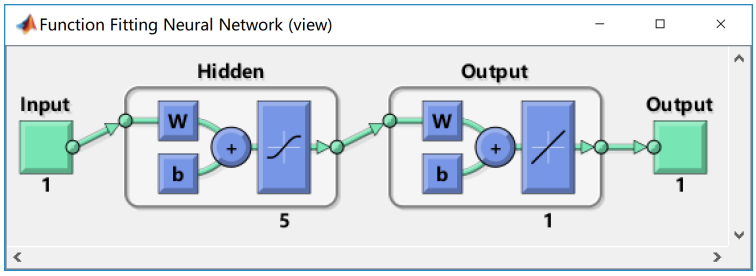


图 1-1 人工神经网络的结构

(3) 使用神经网络绘制曲线,并与理论值比较,如图 1-2 所示。可见,即使不系统学习人工神经网络,也可以直接利用人工神经网络解决实际问题。用户还可以调整神经网络的结构参数,如修改节点个数等,通过实践观察和比较不同参数下曲线拟合的效果。

```
>> t0=0:0.01:pi; y1=net(t0); y0=exp(-t0).*sin(2*t0+2);
   plot(t0,y0,t0,y1)
```

从表面上看,本套系列著作涉及大量的数学公式,有些看起来很深奥。但是,即使读者的数学基础不是很好,也不要害怕,因为本套系列著作的目标不是讲解数学问题的底层细节,而是帮助读者在大概理解该问题物理含义的前提下,绕开底层烦

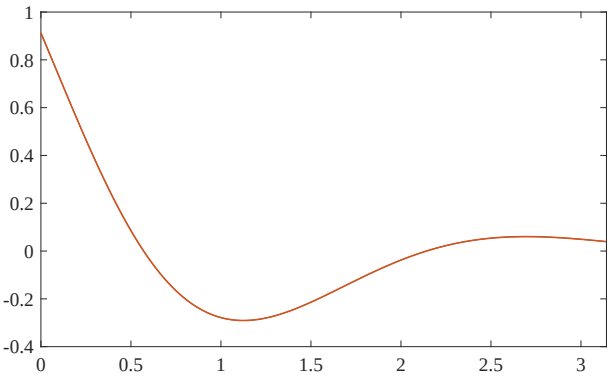


图 1-2　人工神经网络的数据拟合效果

琐的数学求解方法，将问题用计算机能理解的格式推给计算机，直接得出问题可靠的解。借助计算机能提供的强大求解工具，读者求解实际应用数学问题的能力完全可以远超过不会或不擅用计算机工具的一流数学家。通过学习本书的内容，读者能显著地提高应用数学问题的实际求解水平。

本章习题

1.1 在机器上安装MATLAB语言环境，并输入`demo`命令，由给出的菜单系统和对话框运行演示程序，领略MATLAB语言在求解数学问题方面的能力与方法。

1.2 考虑“2017^{2017}”的例题。用C语言常用的数据结构有可能表示该数据吗？如果不能，试利用MATLAB计算该结果（提示：应该用`sym(2017)`表示2017）。

1.3 人们到底能记住圆周率π的前多少位？试试`vpa(pi,50)`命令，让计算机帮助“记忆”，将50再换成更大的数试试。

1.4 科学运算问题靠记忆是不靠谱的，即使记得住π，还能记得住$\sqrt{\pi}$、$\sqrt[3]{\pi}$吗？可以再试试`vpa(sym(pi)^(1/15),500)`，读者能猜出来该命令计算的是什么吗？

1.5 假设已知广义Lyapunov方程如下：

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \boldsymbol{X} + \boldsymbol{X} \begin{bmatrix} 16 & 4 & 1 \\ 9 & 3 & 1 \\ 4 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

试利用`lookfor lyapunov`命令查询与关键词`lyapunov`有关的函数名，并用`doc`或`help`命令获得相关函数的进一步调用信息，观察是否能得出该方程的解，并检验得出解的精度。

1.6 利用联机帮助命令`help sym/diff`查询符号运算工具箱中的求导函数`diff()`，在了解所提供的帮助信息的基础上试着求解例1-2中给出的问题，并比较两个结果。另外，试通过积分运算还原回原函数。

第2章

MATLAB语言程序设计基础

MATLAB语言是当前国际上自动控制领域的首选计算机语言，也是很多理工科专业最适合的计算机数学语言。本书以MATLAB语言为主要计算机语言，系统、全面地介绍在数学运算问题中MATLAB语言的应用。掌握该语言不但有助于更深入地理解和掌握数学问题的求解思路，提高求解数学问题的能力，而且还可以充分利用该语言进行编程，在其他专业课程的学习中得到积极的帮助。

与其他程序设计语言相比，MATLAB语言有如下的优势：

（1）**简洁高效性**。MATLAB程序设计语言集成度高，语句简洁。一般用C或C++等程序设计语言编写的数百条语句，用MATLAB语言一条语句就能解决问题。其程序可靠性高、易于维护，可以大大提高解决问题的效率和水平。

（2）**科学运算功能**。MATLAB语言以矩阵为基本单元，可以直接用于矩阵运算。另外，最优化问题、数值微积分问题、微分方程数值解问题、数据处理问题等都能直接用MATLAB求解。

（3）**绘图功能**。MATLAB语言可以用最直观的语句将实验数据或计算结果用图形的方式显示出来，并可以将难以显示出来的隐函数直接用曲线绘制出来。MATLAB语言还允许用户用可视的方式编写图形用户界面，其难易程度和Visual Basic相仿，这使得用户可以很容易地利用该语言编写通用程序。

（4）**庞大的工具箱与模块集**。MATLAB是被控制界的学者“捧红”的，是控制界通用的计算机语言，在应用数学及控制领域等几乎所有的研究方向均有自己的工具箱，而且由专业领域内知名专家编写，可信度比较高。随着MATLAB的日益普及，在其他工程领域也出现了工具箱，这也大大促进了MATLAB语言在诸多领域的应用。

（5）**强大的动态系统仿真功能**。Simulink提供的面向框图的仿真及概念性仿真功能，使得用户能容易地建立复杂系统模型，准确地对其进行仿真分析。Simulink的概念性仿真模块集允许用户在一个框架下对含有控制环节、机械环节和电子、电

机环节的机电一体化系统进行建模与仿真,这是目前其他计算机语言无法做到的。

本章2.1节将介绍MATLAB语言编程的最基本内容,包括变量名命名规则、MATLAB保留的常数、显示格式的设定、工作环境的设置与工作空间管理等内容。2.2~2.4节将介绍MATLAB下支持的数据结构,包括常用的双精度变量与符号型变量、字符串、多维数组、单元数组、表格、结构体等。2.5节将介绍MATLAB的基本语句结构,包括简单赋值语句与函数调用语句等,还将介绍冒号表达式与子矩阵提取方法。2.6节将介绍MATLAB变量与文件的交互方法,包括普通数据文件,纯文本文件与Microsoft Excel文件等。

掌握了MATLAB语言的基本功能与程序设计方法,就能更好地理解和使用该语言研究科学运算问题求解的内容,还可以为其他相关后续课程的学习打下良好的基础,因为学会使用这样强有力的工具,就多了一个观察问题的视角,可能会发现传统课程或学科分支中很少有人注意到的新内容。

2.1　MATLAB命令窗口与基本命令

本节将介绍MATLAB最基础的入门知识:包括变量的命名规则;MATLAB保留的常数量;MATLAB命令窗口的基础命令、显示环境与工作环境的设置;MATLAB工作空间的管理等。

2.1.1　变量名命名规则

MATLAB语言变量名应该由一个字母引导,后面可以跟字母、数字、下画线等。例如,`MYvar12`、`MY_Var12`和`MyVar12_`均为有效的变量名,而`12MyVar`和`_MyVar12`为无效的变量名。在MATLAB中变量名是区分大小写的,也就是说,`Abc`和`ABc`两个变量名表达的是不同的变量,在使用MATLAB语言编程时一定要注意。

另外一点值得注意的是,如果不小心使用了一个与MATLAB已有函数同名的变量名,则有可能屏蔽掉原来的函数,导致错误的结果,所以在使用变量名前应该避开已有的变量名,例如,先用`which`命令查一下有没有这样的函数名。

另一种测试某名字是否被占用的方法是使用`key=exist('name')`函数,其中,要测试的名字为`name`,`key`为检测结果,若`key`为1表示MATLAB当前工作空间中存在一个名为`name`的变量名;为2则表示MATLAB路径下存在`name.m`文件;为3则表示在MATLAB的路径下存在一个`name.dll`文件;为4则表示存在一个Simulink文件;为5则表示存在一个内核的MATLAB函数`name()`;为6则表示在MATLAB路径下存在伪代码文件`name.p`;为7则表示在MATLAB路径下存在一个`name`文件夹,所以变量命名时这些都是需要避开的。

例2-1 exp(x)函数可以用于求变量的e^x指数，但如果不小心将其设置为变量名，则会屏蔽掉原来的函数。可以尝试一下下面的语句：

```
>> exp(1), exp(5), exp=3.1; exp(1), exp(5)
```

在执行exp=3.1命令之前，该函数可以正常运行，而执行之后，exp()函数就被屏蔽掉了，所以可能得出错误的结果。如果想恢复到正常状态，则应该将clear exp命令的exp变量删除。

2.1.2 保留的常数

在MATLAB语言中还为特定常数保留了一些名称，虽然这些常量都可以重新赋值，但建议在编程时应尽量避免对其重新赋值。

（1）eps。机器的浮点运算误差限。PC上eps的默认值为2.2204×10^{-16}，若某个量的绝对值小于eps，则可以认为这个量为0。

（2）i和j。若常量i或j未被改写，则它们都表示纯虚数量j $=\sqrt{-1}$。但在MATLAB程序编写过程中经常可能改写这两个变量，如在循环过程中常用它们表示循环变量，则应确认使用这两个变量时没有被改写。如果想恢复该常量，则可以用语句i=sqrt(−1)或i=1i重新设置。

（3）Inf。无穷大量$+\infty$的MATLAB表示，也可以写成inf。同样地，$-\infty$可以表示为-Inf。在MATLAB程序执行时，即使遇到了以0为除数的运算，也不会终止程序的运行，而只给出一个“除0”警告，并将结果赋成Inf，这样的定义方式符合IEEE的标准。从数值运算编程角度看，这样的实现形式明显优于C语言这样的非专业计算机语言。

（4）NaN。不定式（not a number），通常由0/0运算、Inf/Inf、0*Inf及其他可能的运算得出。NaN是一个很奇特的量，如NaN与Inf的乘积仍为NaN。

（5）pi。圆周率π的双精度浮点表示，保留数位为3.141592653589793。

（6）true或false。逻辑变量，表示“真”或“伪”，又表示为逻辑1或逻辑0。

（7）lasterr和lastwarn。存放最新一次的错误信息或警告信息。此变量为字符串型，如果在本次执行过程中没出现过错误或警告，则此变量为空字符串。

例2-2 如果某个圆的半径$r=5$，试求其周长与面积。

解 由于圆周长与面积的计算公式分别为$L=2\pi r$，$S=\pi r^2$，可以利用常数pi，并由下面的公式得出该圆的周长与面积分别为$L=31.4159$，$S=78.5398$。

```
>> r=5; L=2*pi*r, S=pi*r^2 %计算圆的周长和面积
```

其中>>为MATLAB的提示符，由机器自动给出，在提示符下可以输入各种的MATLAB命令。

MATLAB的语句之间可以由逗号或分号分隔，也可以换行开始下一条语句。若语句末尾有分号时，语句的执行结果不显示出来，末尾没有分号的语句将在语句执行后直接显示结果。例如，前面的语句中，$r=5$语句后面有分号，所以结果不显示出来，而后面两条语句由于不是分号结尾的，所以结果会自动显示出来。

2.1.3 显示格式的设置

在默认的格式下，MATLAB显示采用的是`short`（短型）格式，通常显示到小数点后4位有效数字，如果想显示更多位有效数字，则可以采用`long`（长型）格式，通常可以显示小数点后15位有效数字，设置语句是`format long`，若想恢复成短型格式，则给出`format short`命令。

除了这两种最常用的显示格式外，`format`命令可以带的选项还包括`compact`（紧凑型）、`loose`（宽松型，默认的）、`rat`（有理型）等。

值得指出的是，`format`命令并不能改变计算的结果，它改变的只有显示的格式，一般情况下用户可以根据需要自行选择有利的显示格式。

例2-3　试显示例2-2中圆周长、面积的更精确的结果，并求出其有理近似。

解　由下面语句可以重新计算圆周长与面积，并显示出更精确的结果。

```
>> format long, r=5; L=2*pi*r, S=pi*r^2
```

得出的结果为$L=31.415926535897931$，$S=78.539816339744831$。

如果想得出其有理近似，则可以将显示形式设置为`rat`：

```
>> format rat, L, S, format short
   e1=3550/113-2*pi*r, e2=8875/113-pi*r^2
```

这样可以得出$L=3550/113$，$S=8875/113$，还可以得出周长与面积的有理近似误差分别为$e_1=2.6676\times10^{-6}$，$e_2=6.6691\times10^{-6}$。

还可以使用`get(0,'Format')`命令读取当前的显示形式。

如果已知变量a，还可以使用`disp(a)`将变量a在MATLAB命令窗口中直接显示出来，其中a为MATLAB支持的任何数据结构。

MATLAB还提供了`type file_name`命令显示纯文本文件`file_name`，也可以使用`edit`命令打开纯文本文件进行编辑。

2.1.4 底层操作系统命令

在MATLAB下可以直接执行操作系统命令如下：

[执行状态，结果]=dos(命令名，参数)

[执行状态，结果]=unix(命令名，参数)

如果“执行状态”为0则执行成功，执行结果将在“结果”中返回。用户可以尝

试 [s,a]=dos('dir','-echo') 命令，观察得出的结果。

除了 dos()、unix() 这些函数之外，还可以使用 cd（改变路径）、pwd（显示当前路径）、delete（删除文件）、recycle（将被删除的文件转存）等函数或命令。还可以直接在 MATLAB 下执行某个可执行文件，例如若在当前路径下有一个文件 mytest.exe（或 *.com 文件），则可以给出 !mytest 命令直接执行这个文件。

2.1.5 MATLAB 的工作环境设置

启动 MATLAB 之后，在标准 MATLAB 界面上会有一个如图 2-1 所示的工具栏，其中允许用户实现文件的读写、变量的设置等操作。直接单击工具就可以完成各种各样的简单任务。

图 2-1 MATLAB 工具栏

单击“布局”按钮可以设置 MATLAB 窗口的显示形式，单击“预设”图标可以实现默认的初始设置，如字体与颜色设置等。

在实际应用中有时读者可能用到自己编写的或下载的其他工具箱，此时需要扩展 MATLAB 的工作路径，将新工具箱的路径包含在内。单击 MATLAB 命令窗口工具栏中的“设置路径”图标，则可以得出如图 2-2 所示的对话框。根据需要选择“添加文件夹”或“添加并包含子文件夹”，进一步设置路径，设置之后单击“保存”按钮，则下次启动 MATLAB 时也会自动完成路径设置。

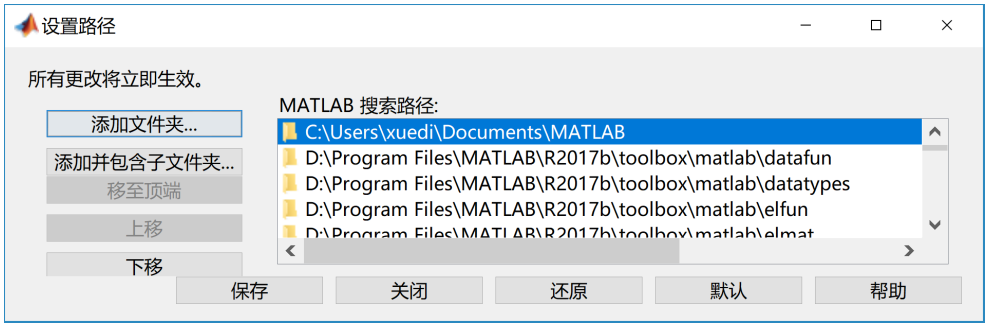


图 2-2 “路径设置”对话框

MATLAB 默认的工作文件夹是 MATLAB 根目录下的 bin 文件夹，该路径是只读的，有的时候使用起来不是特别方便，建议将其设置在计算机“我的文档”下的 MATLAB 文件夹。可以建立一个 startup.m 文件，参考下面的内容：

```
cd('C:\Users\xuedi\Documents\MATLAB') %根据实际情况修改此路径
format compact                        %设置紧凑型显示格式
```

将该文件移动到MATLAB根目录下的bin文件夹，下次启动MATLAB后会自动执行该文件，所以要设置好工作环境。

2.1.6　MATLAB的工作空间与管理

MATLAB的工作空间（workspace）是MATLAB存储变量的场所，用who命令将显示出当前工作空间中所有的变量名，而whos命令将显示出所有的变量名及其数据结构、占用空间等信息。

如果想清除工作空间的所有变量，则可以给出clear命令。如果想清除工作空间中的某几个变量，在clear命令后列出这些变量名即可，注意这些变量名之间用空格分隔。与之相近，clearvars命令允许清除若干个变量，该命令还允许使用-except选项列出需要保留的变量名，该命令将清除这些变量以外的所有变量。

用户可以使用save和load这一对命令存储或读入某些变量或全部变量，正常情况下对应的文件名是以mat为后缀名的，存储文件的格式也是二进制格式的。

还可以直接给出workspace命令，直接打开工作空间管理器界面，用户可以从中选择并处理相应的变量。

2.1.7　MATLAB的其他辅助工具

本节将介绍MATLAB编程与命令窗口使用中的一些技巧，包括耗时检测、历史命令查询与代码分析等，以便读者能更高效地使用MATLAB语言，解决自己的问题。

（1）**箭头键的使用**。按上箭头键可以回滚给出以前的命令，所以查找以前的命令可以使用上箭头；如果想找出一条以a开始的命令，则输入a再按上箭头回滚寻找以前的命令。

（2）**命令历史信息窗口**。单击图2-1工具栏的“布局”按钮，则可以选择“命令历史信息”，打开历史信息窗口，从中选择以前给出的命令，双击则可以再次执行这些命令。这样的方式有时比箭头方式更实用。

（3）**测耗时**。MATLAB提供了两套程序耗时计时方法，一套是采用tic、toc命令对，在程序执行前调用tic命令启动秒表，程序执行后，调用toc命令读耗时；另一套命令是采用读取CPU时间的方法实现的，程序执行前调用t_0=cputime存储当前的CPU时间，程序执行后，由cputime$-t_0$命令测得执行时间。这两种计时方法各有特点，计时结果相差不大。

（4）**代码分析器**。单击工具栏“代码分析器”按钮，则将打开如图2-3所示的窗

口，对当前路径下的MATLAB程序进行检验，给出修改或优化建议。例如该窗口对bk_prt.m函数提出三条建议，用户可以自己选择是否依照建议修改自己的程序。

图 2-3 代码分析器界面

2.2 常用数据结构

程序设计中很关键的内容是数据结构。本节将介绍两类科学运算中常用的数据结构——数值型数据结构与符号型数据结构。2.3、2.4 节将介绍其他的数据结构，为程序设计的介绍奠定一个坚实的基础。

2.2.1 数值型数据

强大方便的数值运算功能是MATLAB语言最显著的特色。为保证较高的计算精度，MATLAB语言中最常用的数值量为双精度浮点数，占8字节（64位），遵从IEEE记数法，有11个指数位、52位尾数及1个符号位，值域的近似范围为 $-1.7\times10^{308}\sim1.7\times10^{308}$，其MATLAB表示为`double()`。

在极个别的场合，可能会使用到单精度数据结构。该数据结构为32位二进制浮点数，一般能保留小数点后 7位有效数字，其MATLAB转换命令为`single()`。

MATLAB中最基本的数据结构是双精度复数矩阵，这里将通过例子演示一般矩阵的输入方法。

例2-4 试在MATLAB工作空间中输入矩阵

$$\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 0\end{bmatrix}$$

解 在MATLAB语言中表示一个矩阵是很容易的事，可以由下面的MATLAB语

句将该矩阵直接输入到工作空间中:

```
>> A=[1,2,3; 4 5,6; 7,8 0]     %矩阵的直接输入语句
```

该语句将矩阵赋给变量$\boldsymbol{A}$,同时,在命令窗口中按照下面的格式显示该矩阵:

```
A =
     1     2     3
     4     5     6
     7     8     0
```

为阅读方便,本书后续内容将不再给出MATLAB格式的显示,而直接给出数学格式的显示。矩阵的内容由方括号括起来的部分表示,在方括号中的分号或回车符号表示矩阵的换行,逗号或空格表示同一行矩阵元素间的分隔。给出了上面的命令,就可以在MATLAB 的工作空间中建立一个$\boldsymbol{A}$变量了。

还可以在已知$\boldsymbol{A}$矩阵的基础上,按照MATLAB允许的行列格式规则,给出下面的语句,动态地调整矩阵$\boldsymbol{A}$的维数。

```
>> A=[[A; [1 2 3]], [1;2;3;4]] %矩阵维数动态变化
```

例2-5　试在MATLAB环境中输入复数矩阵

$$\boldsymbol{B}=\begin{bmatrix}1+9\mathrm{j} & 2+8\mathrm{j} & 3+7\mathrm{j}\\ 4+6\mathrm{j} & 5+5\mathrm{j} & 6+4\mathrm{j}\\ 7+3\mathrm{j} & 8+2\mathrm{j} & 0+\mathrm{j}\end{bmatrix}$$

解　复数矩阵的输入同样也是很简单的,在MATLAB环境中定义了两个记号i和j,可以用来直接输入复数矩阵。这样可以通过下面的MATLAB语句对复数矩阵直接进行赋值,其中,在单独表示j时,建议使用1i或sqrt(−1),不建议使用i或j。

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i]
```

2.2.2　符号型数据

MATLAB还定义了符号型(symbolic)变量,以区别于常规的数值型变量,可以用于公式推导和数学问题的解析解法。进行解析运算前需要首先将采用的变量声明为符号变量,这需要用syms命令实现。该语句具体的用法为

```
syms 变量名列表　变量集合
```

其中“变量名列表”给出需要声明的变量列表,可以同时声明多个变量,中间只能用空格分隔,而不能用逗号等其他符号分隔。

如果需要,还可以进一步声明变量的“变量集合”,可以使用的集合为positive(正数)、integer(整数)、real(实数)、rational(有理数)等。如果需要将a、b均定义为符号变量,则可以用syms a b语句声明,该命令还支持对符号变量具体形式的设定,如syms a real可以将变量a设置为实数。如果将“变量集合”设置为clear,则将清除变量的集合设定,将其还原为一般符号变量。

符号变量的类型可以由assumptions()函数读出,例如,若用syms a real语

句声明变量a,则assumptions(a)将返回in(a,'real')。

MATLAB符号运算工具箱还提供了$\boldsymbol{x}$=symvar(f)函数,可以从符号表达式f中提取符号变量列表$\boldsymbol{x}$。

符号型数值可以通过变精度算法函数vpa()以任意指定的精度显示出来。该函数的调用格式为vpa(A)或vpa(A,n),其中A为需要显示的表达式或矩阵,n为指定的有效数字位数,前者以默认的32位十进制位数显示结果。

例2-6 如何用计算机描述$1/3$和简单的算数运算$1/3\times 0.3=?$

解 常规计算机语言采用双精度数据结构,计算机数学语言则支持符号型数据结构。在表示数值上符号型数值与双精度数值有什么区别呢?双精度数据结构是不能存储$1/3$的,只能存储成0.333333333333333,后面的各位都被截断了,而符号型的sym(1/3)全程存储和参与运算的都是$1/3$,没有误差。

很显然,在数学上$1/3\times 0.3=0.1$,在符号数据结构下也确实如此。现在看看在双精度数据结构下会出现什么现象:可以给出下面的语句,从得出的结果可见,二者之间是有误差的,误差为-1.3878×10^{-17}。

```
>> 1/3*0.3-0.1
```

例2-7 试显示出圆周率π的前105位有效数字。

解 使用符号运算工具箱中提供的vpa()函数可以按任意精度显示符号变量的值,故题中要求的结果可以用下面语句实现:

```
>> vpa(pi,105)  %显示圆周率π的前105位,还可以选择更多的位数
```

这样可以显示出π的值为3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798215。若不指定位数n,则vpa(pi)命令将得出结果为$\pi=3.14159265358979323846264338327 95$。

值得指出的是,由这里给出的命令行显示的格式最多只能显示32766个字符,如果想显示更多位,则可以参考后面例4-14给出的方法分行显示。

例2-8 试显示无理数e的前50位数字。

解 如果想得出e的前50位数,可以尝试vpa(exp(1),50)命令,不过该命令会先在双精度框架下得出e,再显示其前50位,所以显示的结果是不精确的,正确的方法是应该在符号型运算的框架下计算e,使用的语句应该为vpa(exp(sym(1)),50),得出的结果为2.7182818284590452353602874713526624977572470937。

符号变量的属性还可以由assume()与assumeAlso()函数进一步设置。例如,若x为实数,且$-1\leqslant x<5$,则可以用下面的MATLAB语句直接设定:

```
>> syms x real; assume(x>=-1); assumeAlso(x<5); %设定-1≤x<5
```

调用assumptions(x)函数,则将显示出符号变量x为[x<5,-1<=x]。

例2-9　试声明一个不超过3000的正整数型符号k,使其为13的倍数。

解　可以计算出$\lfloor 3000/13 \rfloor$,这样可以给出下面的MATLAB命令声明正整数k:

```
>> syms k1; assume(k1,'integer');
   assumeAlso(k1<=floor(3000/13)); %计算最大允许的整数
   assumeAlso(k1>0); k=13*k1 %声明变量的下界,则整数变量k为13的倍数
```

如果在MATLAB工作空间中已有a变量,则原则上可以通过A=sym(a)将其转换成符号变量,不过有时应该做特殊的处理,这里将通过下面的例子做出演示。

例2-10　试用符号型数据结构表示数值12345678901234567890。

解　这个问题看似很简单,可以由命令A=sym(12345678901234567890)直接输入,不过读者可能对得出的结果感到困惑不解,因为得到的是$A = 12345678901234567168$,显然这不是正确的。从MATLAB的执行机制看,该语句首先将数据转换成双精度结构,然后再转换成符号变量,从而出现偏差,所以,在数据类型转换时应该格外注意。正确的解决方法是用字符串表示多位的数字,然后再用sym()函数转换。下面的语句可以原封不动地输入50位整数。

```
>> B=sym('12345678901234567890123456789012345678901234567890')
```

例2-11　试将例2-5中的复数矩阵转换成符号型的数据结构。

解　仍使用例2-5中的命令输入该矩阵,再调用sym()函数则可以将其直接转换为符号型矩阵,显示其结果比较两种数据结构显示格式上的差异。

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i], B=sym(B)
```

2.2.3　任意符号型矩阵的生成

MATLAB提供的sym()函数还可以用于任意矩阵的生成,例如,由命令

$\boldsymbol{A}$=sym('a',[n,m]), $\boldsymbol{B}$=sym('b%d%d',[n,m])

可以生成任意矩阵$\boldsymbol{A}$与$\boldsymbol{B}$,不过其格式略有不同,$\boldsymbol{A}$矩阵的元素为ai_j,矩阵$\boldsymbol{B}$的元素为b_{ij},可见前者命令有问题。

2.2.4　符号型函数

在符号型数据结构的基础上还可以定义出符号型函数。符号型函数同样可以由syms命令直接声明,下面通过例子演示声明方法。

例2-12　试声明符号函数$F(x)$、$G(x,y,z,u)$。

解　应该先将自变量声明为符号变量,然后再声明函数$F(x)$与$G(x,y,z,u)$。

```
>> syms x y z u F(x) G(x,y,z,u)
```

2.2.5　整型变量与逻辑变量

考虑到一些特殊的应用,例如图像处理,MATLAB语言还引入了无符号的8位整形数据类型,其MATLAB表示为uint8(),其值域为$0 \sim 255$,这样可以极大地

节省MATLAB的存储空间，提高处理速度。此外，在MATLAB中还可以使用其他整型数据类型，如int8()、int16()、int32()、uint16()、uint32()等，每一个类型后面的数字表示其位数，其含义不难理解。

MATLAB还提供了逻辑型的数据结构，只能取值0和1，其对应的转换函数为logical()。双精度或其他数据结构有时也可以起逻辑变量的作用，如果某双精度变量的值为0，则可以认为它为逻辑0，否则可以认为是逻辑1。

2.2.6 数据结构类型的识别

可以用key=class(a)函数直接识别出a变量的数据结构类型key，其中，若a为符号变量则返回'sym'，符号函数则返回'symfun'。其他支持的数据结构类型为'double'（双精度）、'single'（单精度）、'int*'（整型，*为位数，如16位）、'uint*'（无符号整型）、'char'（字符型、字符串）、'logical'（逻辑型）、'struct'（结构体）、'cell'（单元数组）、'function_handle'（函数句柄）等。

除了用class()函数之外，MATLAB还提供了一系列以is开头的函数来判定某个变量a的数据结构，如key=isdouble(a)可判定a是否为双精度变量，如果是则返回的key为逻辑1，否则为逻辑0。

这类函数还有很多，例如ischar(a)（判定a是否为字符串）、isnumeric(a)（判定a是否为数字）等，这类函数名的含义很明确，在这里不过多地解释了。与这类命令类似，还可以使用isa()函数具体判定数据结构，例如，key=isa(a,'double')可以用来判定a是不是双精度数据结构，其作用基本上与isdouble()函数等效。

2.2.7 矩阵的维数与长度

可以由size()函数与length()函数测取矩阵或向量的长度，具体的调用格式是比较好理解的，这里就不多解释了。

$\boldsymbol{k}$=size($\boldsymbol{A}$)，[n,m]=size($\boldsymbol{A}$)，n=length($\boldsymbol{v}$)

函数length($\boldsymbol{A}$)还可以读取矩阵$\boldsymbol{A}$的长度，即$\boldsymbol{A}$矩阵行数与列数的最大值，还可以分别由size($\boldsymbol{A}$,1)与size($\boldsymbol{A}$,2)提取$\boldsymbol{A}$矩阵的行数和列数。

MATLAB还提供了reshape()函数重新调整矩阵的维数，其调用格式为

$\boldsymbol{B}$=reshape($\boldsymbol{A}$,n_1,m_1)，$\boldsymbol{B}$=reshape($\boldsymbol{A}$,[n_1,m_1])

它可以将总共$m \times n$个元素的$\boldsymbol{A}$矩阵转换成n_1行m_1列的新矩阵$\boldsymbol{B}$，其中，$nm = n_1 m_1$。具体方法是将$\boldsymbol{A}$矩阵先按列展开生成列向量，将其截为长度为n_1的列向量，总共可以截出m_1段，这样就可以将这些列向量依次排列构造出新的$\boldsymbol{B}$矩阵了。

MATLAB的openvar(var)函数会打开一个数据编辑图形用户界面，允许用户通过可视的方式编辑变量，其中var为变量名字符串，后面将演示该函数的应用。

2.3　字符串数据结构

字符串是程序设计中的常用数据结构，很多输入输出应用都需要借助于字符串实现。本节将介绍字符串的表示方法，并将介绍字符串查找、字符串比较等一般处理方法，最后将介绍字符串的读写与转换方法。

2.3.1　一般字符串的表示

MATLAB支持字符串变量，可以用它存储相关的信息。其实，例2-10已经介绍了字符串的一种应用场合。

与C语言等程序设计语言不同，MATLAB字符串是用单引号引起来的，而不是用双引号。例如，由下面的语句可以直接将一个字符串输入给计算机。

```
>> strA='Hello World!'
```

例2-13　多个字符串的串接与其他处理方法演示。

解　字符串中可以使用中文。如果有多个字符串，可以按照向量构造的形式把它们串接起来，形成一个更长的字符串，例如

```
>> strA='Hello World!'
   strB=' 三个字符串串联  '; strC=[strA, strB, strA]
```

该语句得出一个更长的字符串，是由这三个字符串串接而成，结果如下：

```
'Hello World! 三个字符串串联  Hello World!'
```

MATLAB还可以将若干个不同长度的字符串处理成字符串“列向量”，可以采用MATLAB中提供的str2mat()函数实现。

```
>> strD=str2mat(strA,strB,strA)
```

前面语句生成的是3×12的字符串数组，结果如下：

```
'Hello World!'
' 三个字符串串联  '
'Hello World!'
```

如果想提取出第1行的字符串，需要使用strD(1,:)命令提取整行字符串，而不能使用strD(1)命令，否则只能提取第一行的第一个字符。函数strvcat()的作用与str2mat()函数是完全一致的。

既然字符串变量是由单引号括起来的，那么如何在一个字符串中表示单引号呢？字符串中的单引号应该由两个接连的单引号表示。这样就不难理解由下面的字符串赋值语句得出的结果了。

```
>> strE='In this string, single quote '' is defined.'
```

2.3.2　字符串的处理方法

本节的函数可以进行字符串的对比、查找和替换，同时，下面还将介绍与字符串有关的其他处理方法。

（1）**字符串比较**。可以由strcmp()函数完成两个字符串的比较，其调用格式为k=strcmp(str$_1$,str$_2$)，其中str$_1$和str$_2$为两个被比较的字符串，若两个字符串完全相同，则返回的k为1，否则为0。

注意，在MATLAB编程时一定不能用str$_1$==str$_2$这样的关系运算式判定两个字符串是否相同，否则当两个字符串长度不同时会导致错误。

（2）**字符串查找**。函数findstr()可以用来找出一个子字符串在另一个字符串中出现处的下标，其调用格式为$\boldsymbol{k}$=findstr(str$_1$,str$_2$)，其中str$_1$和str$_2$为两个字符串，此函数将返回较短的一个字符串在另一个字符串中出现的下标位置，如果该字符串不在另一个字符串中出现，则返回一个空矩阵。

例2-14 如果strA变量存储了字符串'Hello World!'，试找出字母o所在的位置。该字母出现几次？

解 如果想找出其中的'o'字符，则需给出下面的命令：

```
>> strA='Hello World!'; k=findstr(strA,'o'), length(k)
```

得出的结果为向量$\boldsymbol{k}=[5,8]$，说明字符串的第5个字符和第8个字符为字母o，该字母出现了两次。如果将findstr()函数的两个变元变换次序，得出的$\boldsymbol{k}$是完全一致的。

（3）**字符串替换**。可以用strrep()函数进行字符串的替换，该函数的调用格式为str=strrep(str$_1$,str$_2$,str$_3$)，其中str$_1$为原字符串，str$_2$为要替换掉的子字符串，而str$_3$为要替换成的子字符串，替换后的最终结果在str字符串中返回。例如下面的语句将得出一个新字符串——str='HellLA WLArld!'。

```
>> strF=strrep(strA,'o','LA')
```

（4）**获得字符串的长度**。可以用length()函数测出字符串的字符个数，例如，语句k=length(strA)可以得出字符串的字符个数为$k=12$。

（5）**删除字符串尾部的空格**。可以采用deblank(strA)函数完成这样的工作，若想删除字符串strA中全部的空格，则可以使用下面的语句：

```
str=strA(find(strA~=' ')), 或 str=strA(strA~=' ')
```

例2-15 考虑例2-13中的串联字符串，试测出其字符个数，删除空格后再测出结果字符串中字符的个数。

解 可以用例2-13中的方法构造串联字符串，并得出其字符个数为34。

```
>> strA='Hello World!'; strB=' 三个字符串串联 ';
   strC=[strA,strB,strA]; length(strC) %串联字符串的字符个数
   s1=strC(strC~=' '), length(s1)      %删除空格后的字符串及字符个数
```

删除空格后的字符串为s$_1$='HelloWorld!三个字符串串联HelloWorld!'，其字符个数为29。可见，每个中文文字也记作一个字符。

2.3.3　字符串的转换与读写方法

（1）**字符串与双精度数的相互转换**。double()函数可以获得字符串中各个字符的ASCII码构成的双精度型向量，char()函数可以转换回原来的字符串。

例2-16　试将'Hello World!'字符串转换成ASCII码形式。

解　先将字符串输入到MATLAB工作空间，然后调用double()函数，代码如下：

```
>> strA='Hello World!'; v=double(strA), s1=char(v)
```

得出对应ASCII码为$\boldsymbol{v}=[72,101,108,108,111,32,87,111,114,108,100]$，每一个数字对应相应的字符。如果对结果运行char()函数，则将还原回原来的字符串。

（2）**符号表达式的字符串转换**。如果给出了符号表达式a，则可以由char()函数将其转换为字符串的形式，调用格式为str=char(a)。

（3）**MATLAB变量转换为字符串**。MATLAB还提供了很多字符串转换函数，如果$\boldsymbol{v}$是行向量，则可以由str=num2str($\boldsymbol{v}$)或str=num2str($\boldsymbol{v}$,n)命令将其转换成字符串，默认的格式是保留小数点后四位有效数字。此外，还允许用户指定n选择有效数字位数。

例2-17　试观察双精度下的1/3到底在MATLAB下表示成什么值。

解　如果想观察该值的精确结果，则可以将其转换成字符串，多显示几位。由下面的语句可见，1/3在双精度数据结构下存储为0.333333333333333314829616256247，其小数点后前15位是准确的，15位后的其余数字是MATLAB双精度数据结构由某种规则自动生成的不可靠的数字。

```
>> a=1/3; num2str(a,30)
```

如果$\boldsymbol{v}$是矩阵，num2str()函数将逐行转换矩阵，生成一个字符串矩阵。除了这个函数之外，MATLAB还提供了int2str()函数，将整数向量转换成字符串。如果$\boldsymbol{v}$不是整数向量，则会自动对其取整再转换成字符串。

（4）**带有格式的字符串生成方法**。MATLAB还提供了底层函数sprintf()，将得出的结果以指定的格式写入字符串，该函数与其原型的C语言同名函数调用格式是很接近的。

str=sprintf(格式,a_1,a_2,$\cdots$,a_m)

其中“格式”为读写格式控制字符串，'%d'表示输出整数，'%f'表示输出浮点数，'%s'表示输出字符串。这样，a_i变量将按照指定的格式写入字符串str。

例2-18　试用更可读的格式显示例2-2得出的圆周长与面积。

解　例2-2使用直接显示变量的方法显示了得出的结果，这里考虑用带有格式的方法显示得出的结果，增加其可读性。可以给出下面的语句：

```
>> r=5; L=2*pi*r; S=pi*r^2; %计算周长与面积,但不显示
   str=sprintf('圆周长为%f,圆面积为%f',L,S), disp(str)
```

这样得出的字符串 str 经过 disp() 函数处理,显示的结果为

圆周长为 31.415927,圆面积为 78.539816

2.3.4 字符串命令的执行

MATLAB 提供了可以执行字符串命令的函数 eval()。实际编程中有时可以将命令生成字符串 str 的形式,然后调用 eval(str)函数执行字符串,得出所需的执行结果。下面将通过例子演示这样的命令格式。

例 2-19 假设有一个行向量 $\boldsymbol{b}$,其分量个数 n 可以由 length() 函数测出,试给出 MATLAB 语句,将这个行向量的每个元素依次赋给变量 a1,a2,$\cdots$,an。

解 先随意写出一个 $\boldsymbol{b}$ 行向量,然后用循环语句(后面将专门介绍)对每一个分量单独处理。这些内容用常规方法是没有问题的,难点是怎么生成一个变量名为 a* 的变量。可以用字符串的方式实现,生成字符串 s 并调用 eval() 函数执行,则可以解决问题。

```
>> b=[1 3 7 5 4 2 8 9 6 4 3 2 6]; n=length(b); %随意生成一个行向量
   for i=1:n, s=['a' int2str(i) '=b(i);']; eval(s); end
```

对这组选择的向量 $\boldsymbol{b}$,可见,$n=13$,利用 who命令可以发现这组新生成的变量 a1,a2,$\cdots$,a13,观察这些变量的值可以发现完成了预定任务。

还可以利用 MATLAB 提供的 feval() 函数去调用一个已知的 MATLAB 函数,求出该函数的值 feval(fun,p_1,p_2,$\cdots$,p_n),其中 fun 对应一个已知的函数名,从效果上看该语句等效于 fun(p_1,p_2,$\cdots$,p_n)。

2.3.5 MuPAD 接口函数的编写

MATLAB 的符号运算是借助 MuPAD 实现的。MuPAD 提供了各种各样的底层函数,这些函数是不能由 MATLAB 直接调用的,所以符号运算工具箱为常用的符号运算功能提供了专门的 MATLAB 接口函数,通过这些接口函数将 MATLAB 命令传送给 MuPAD 去执行。这种接口的核心命令是

f=feval(symengine,MuPAD 函数名,变量列表)

其中“MuPAD 函数名”是指 MuPAD 底层函数的名字,“变量列表”是 MuPAD 能够接受的输入变量列表。编写接口函数时,应该将 MuPAD 期望的输入变量直接传递给 MuPAD,具体的传递方法将在下面的例子中给出。

例 2-20 假设给定一个符号函数 $f(x)$,该函数可以通过 Padé 近似技术得出一个有理近似函数 $f(x)\approx N(x)/D(x)$,其中 $N(x)$ 与 $D(x)$ 为多项式。MATLAB 并未提供 Padé 近似的符号运算函数,MuPAD 提供的底层函数 pade() 虽然可以求取近似,但 MATLAB 并不能直接调用该函数,需要用户自己编写接口。已知 MuPAD 下 pade() 函

数的调用格式为 F=pade(f,x,[m,n]),试编写出这样的通用接口。

解　由于这里需要函数编写的基础知识，其内容将在第5章中介绍，所以用户现在不必读懂整个函数，阅读并理解后面两条语句就可以了。分子与分母阶次需要由int2str()函数转换成字符串，调用feval()函数将这些参数以固定的格式传给MuPAD,再通过symeigine启动MuPAD的符号运算引擎就可以了。

```
function p=padefrac(f,varargin)
[x,n,m]=default_vals({symvar(f),2,2},varargin{:});
orders=['[' int2str(n) ',' int2str(m) ']'];
p=feval(symengine,'pade',f,x,orders);
```

学会并仿照这样的思想，用户就可以编写出一些实用的MuPAD接口函数，进一步扩展MATLAB下的符号运算功能了。

2.4　其他常用数据结构

MATLAB是一种通用的程序设计语言，所以除了用于计算的数值型、符号型、字符串数据结构之外，还支持其他数据结构，如结构体型数据结构、多维数组型数据结构、单元数组型数据结构、表格型数据结构，此外还支持类与对象的数据结构。本节将系统介绍这些数据结构的定义与使用方法。

2.4.1　多维数组

三维数组是一般矩阵的直接拓展，可以这样理解，三维数组可以直接用于彩色数字图像的描述，在控制系统的分析中也可以直接用于多变量系统的频域响应表示。在实际编程中还可以使用维数更高的数组。

除了标准的二维矩阵之外，MATLAB定义了三维或多维数组。三维数组很好理解，假设有若干个维数相同的矩阵 $\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_m$，那么把这若干个矩阵一页一页地叠起来，就可以构成一个三维数组。三维数组的示意图如图2-4所示。

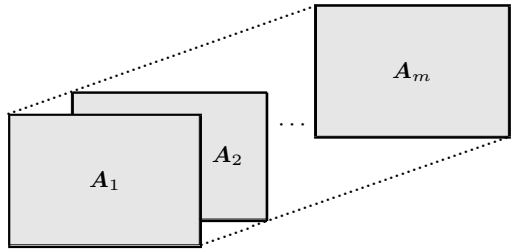


图 2-4　三维数组的示意图

在图像处理与计算机视觉领域，可以将图像用矩阵表示，矩阵每个元素表示图像像素的灰度值。如果仿照图2-4给出的方式，用摞起来的三个矩阵分别表示图像

的红、绿、蓝色像素分量，则这样摞起来的三个矩阵就构成了三维数组，可以用来表示彩色图像。

三维以上的数组是不能用示意图来表示的，但通过三维数组的定义就不难理解多维数组的意义。构造m维数组时，可使用如下的cat()函数构造m维数组：

$\boldsymbol{A}$=cat(m,$\boldsymbol{A}_1$,$\boldsymbol{A}_2$,$\cdots$,$\boldsymbol{A}_m$)

其中$\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_m$数组均应该为$m-1$维数组。

size()函数可以用于多维数组的维数检测，size($\boldsymbol{A}$,k)命令可以读取第k维的元素个数。如果$\boldsymbol{A}$是矩阵，则$k=1$提取矩阵行数，$k=2$提取矩阵列数。

对多维数组$\boldsymbol{A}$来说，$\boldsymbol{A}$(:)命令将得出原多维数组所有元素按列排列的列向量，而reshape()函数也可以用于多维数组的重新定维。

2.4.2 单元数组

单元数组是矩阵的直接扩展，其存储格式类似于普通的矩阵，而矩阵的每个元素不是数值，可以认为能存储任意类型的信息，这样每个元素称为“单元”(cell)，例如，$\boldsymbol{A}\{i,j\}$可以表示单元数组$\boldsymbol{A}$的第i行、第j列的内容。

例2-21 将4个完全不同数据结构的变量构造成一个2×2的单元数组。

解 可以描述如下4个变量，它们属于各不相同的数据结构，用矩阵输入的方法显然不能将这样4个互不相干的变量用一个变量描述出来，必须借助单元数组的结构，将它们安排到不同的单元内。单元数组的输入方法与矩阵很接近，所不同的是，不能用方括号，必须使用花括号。可以给出如下的命令：

```
>> A=[1 2 3; 4 5 6; 7 8 0]; strA='Hello World!';
   syms x; clear F; F(x)=x^2*sin(x); B={A,strA; x F}
```

该单元数组的显示结果为

```
  2×2 cell 数组
    {3×3 double}    {'Hello World!'}
    {1×1 sym   }    {1×1 symfun    }
```

单元数组的元素可以用$\boldsymbol{B}\{i,j\}$命令提取出来，例如$\boldsymbol{B}\{2,2\}$可以提取右下角单元的内容。

如果一个单元数组各个单元的数据结构都相同，且维数相容，则可以使用函数cell2mat()将其转换为矩阵的形式，调用格式为$\boldsymbol{A}$=cell2mat(C)。矩阵$\boldsymbol{A}$也可以通过C=mat2cell($\boldsymbol{A}$,$\boldsymbol{v}_1$,$\boldsymbol{v}_2$)命令将矩阵拆分成子矩阵，构造单元数组C，拆分的方式由$\boldsymbol{v}_1$、$\boldsymbol{v}_2$确定。

例2-22 可以考虑将下面的分块矩阵存储成单元数组，然后由单元数组将整个矩

阵提取出来。

$$\boldsymbol{A}=\left[\begin{array}{cc:cccc} 2 & 3 & 1 & 2 & 1 & 1 \\ \hdashline 1 & 2 & 1 & 2 & 3 & 2 \\ 3 & 1 & 2 & 3 & 3 & 1 \end{array}\right]$$

解　由下面的命令先输入4个子矩阵，然后存成单元数组的形式，再调用转换函数cell2mat()就可以将整个矩阵提取出来。

```
>> a11=[2 3]; a12=[1 2 1 1]; a21=[1 2; 3 1];
   a22=[1 2 3 2; 2 3 3 1]; C={a11 a12; a21 a22} %表示成单元数组
   A=cell2mat(C)                                %提取整个矩阵
```

如果想由单元数组C提取$\boldsymbol{A}$矩阵右下角的子矩阵，则可以使用B=C{2,2}命令提取，不过得出的结果是单元数组，需要将其转换成双精度矩阵，$\boldsymbol{A}_1$=double(B)，或更简洁的，$\boldsymbol{A}_1$=double(C{2,2})。

用下面的命令还可以将整个矩阵按照给出的格式拆分，存储成单元数组，拆分中的$\boldsymbol{v}_1$可以取作$\boldsymbol{v}_1=[1,2]$，表示第一个单元行数为1，第二个单元行数为2。相应地，按照给出的分块矩阵，$\boldsymbol{v}_2=[2,4]$，这样可以给出下面的命令：

```
>> C1=mat2cell(A,[1 2],[2 4]) %将矩阵拆分成分块矩阵，构造单元数组
```

2.4.3　表格数据

表格数据的数据类型为table，专门用于处理表格或数据库的存储与处理。下面将通过例子演示表格数据的使用方法。

例2-23　八大行星的一些参数在表2-1中给出，其中相对参数都是由地球参数换算的，半长轴的单位为AU（astronomical unit，天文单位，为149597870700 m$\approx 1.5\times10^{11}$ m），自转周期的单位为天。试用MATLAB表示这些行星参数。

表2-1　八大行星的一些参数

名称	相对直径	相对质量	半长轴	相对轨道周期	离心率	自转周期	卫星个数	行星环
水星	0.382	0.06	0.39	0.24	0.206	58.64	0	无
金星	0.949	0.82	0.72	0.62	0.007	−243.02	0	无
地球	1	1	1	1	0.017	1	1	无
火星	0.532	0.11	1.52	1.88	0.093	1.03	2	无
木星	11.209	317.8	5.20	11.86	0.048	0.41	69	有
土星	9.449	95.2	9.54	29.46	0.054	0.43	62	有
天王星	4.007	14.6	19.22	84.01	0.047	−0.72	27	有
海王星	3.883	17.2	30.06	164.8	0.009	0.67	14	有

解　如果用一个变量表示整个表格，当然可以考虑使用单元数组，不过采用MATLAB的表格结构将更规范、方便。若想采用表格数据结构，需要设计一个表头，以便更好地处理表格。

表头必须用英文单词或字母表示，可以将每一列分别用字符串表示成

```
name,diameter,mass,axis,period,eccentricity,rotation,moon,ring
```

用户可以将表格的每一列用单元数组或列向量的形式先输入给计算机，然后调用 table()函数构造单元数组。由下面的语句就可以输入该表格。由于后面例子还可能用到这个表格数据，所以可以将其存入 c2dtab.mat 文件。

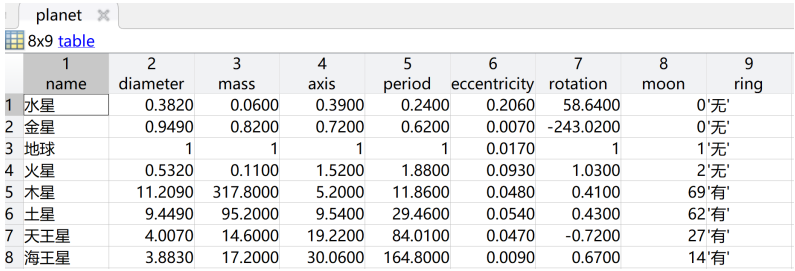
```
>> name=str2mat('水星','金星','地球','火星','木星','土星',...
                '天王星','海王星');
   diameter=[0.382;0.949;1;0.532;11.209;9.449;4.007;3.883];
   mass=[0.06; 0.82; 1; 0.11; 317.8; 95.2; 14.6; 17.2];
   axis=[0.39; 0.72; 1; 1.52; 5.2; 9.54; 19.22; 30.06];
   period=[0.24; 0.62; 1; 1.88; 11.86; 29.46; 84.01; 164.8];
   eccentricity=[0.206; 0.007; 0.017; 0.093; 0.048; ...
        0.054; 0.047; 0.009];
   rotation=[58.64;-243.02;1;1.03;0.41;0.43;-0.72;0.67];
   moon=[0; 0; 1; 2; 69; 62; 27; 14];
   ring={'无';'无';'无';'无';'有';'有';'有';'有'};
   planet=table(name,diameter,mass,axis,period,eccentricity,...
                rotation,moon,ring)
   save c2dtab planet
```

这样，表格变量 planet 就在 MATLAB 工作空间中建立起来了。由于最后一条语句后面没有分号，所以整个变量会以表格的形式直接显示出来，其格式是清晰明了的。如果想提取表格某一列，例如第 5 列，由于其表头名称为 period，所以用下面语句就可以将其提取出来：

```
>> planet.period
```

前面介绍的 openvar()函数可以用来打开表格变量的图形用户界面，如图 2-5 所示，可以利用界面可视地编辑该变量。

```
>> openvar('planet')
```



	1 name	2 diameter	3 mass	4 axis	5 period	6 eccentricity	7 rotation	8 moon	9 ring
1	水星	0.3820	0.0600	0.3900	0.2400	0.2060	58.6400	0	'无'
2	金星	0.9490	0.8200	0.7200	0.6200	0.0070	-243.0200	0	'无'
3	地球	1	1	1	1	0.0170	1	1	'无'
4	火星	0.5320	0.1100	1.5200	1.8800	0.0930	1.0300	2	'无'
5	木星	11.2090	317.8000	5.2000	11.8600	0.0480	0.4100	69	'有'
6	土星	9.4490	95.2000	9.5400	29.4600	0.0540	0.4300	62	'有'
7	天王星	4.0070	14.6000	19.2200	84.0100	0.0470	-0.7200	27	'有'
8	海王星	3.8830	17.2000	30.0600	164.8000	0.0090	0.6700	14	'有'

图 2-5 openvar()函数的图形用户界面

例 2-24 已知地球的质量为 5.965×10^{24} kg，试求出木星的质量。

解　有了例2-23中的表格变量planet,并已知木星数据在第5行,可以立即由下面语句求出木星的质量为1.8957×10^{27} kg:

```
>> load c2dtab; M=5.965e24*planet.mass(5) %读入表格并计算
```

上述语句如果不给出(5),则可以一次性求出所有八大行星的质量。

MATLAB提供了一系列表格数据结构的转换函数,其中函数table2cell()与table2struct()比较实用，可以将表格数据直接转换为单元数组与结构体变量。此外，有时还可以使用table2array()函数，不过对planet这样既包含数字又包含字符串的表格则不能进行转换。

2.4.4　结构体

结构体数据结构struct也适合描述表格型或数据库型信息。结构体可以包含下级的信息,称为域(field)或成员变量(membership variable)。与表格数据结果的表示方式不同,假设某结构体名为T,其中一个域为a,则结构体数据结构通过T.a命令读取或赋值该域。下面通过例子演示结构体变量的处理方法。

例2-25　重新考虑例2-23中的问题，试用结构体数据结构将其输入到MATLAB环境。

解　结构体数据结构的输入与前面计算的表格数据不同,可以通过下面语句将各个域直接输入,也可以通过struct()函数直接输入。

```
>> P.name=str2mat('水星','金星','地球','火星','木星','土星',...
                  '天王星','海王星');
   P.diameter=[0.382;0.949;1;0.532;11.209;9.449;4.007;3.883];
   P.mass=[0.06; 0.82; 1; 0.11; 317.8; 95.2; 14.6; 17.2];
   P.axis=[0.39; 0.72; 1; 1.52; 5.2; 9.54; 19.22; 30.06];
   P.period=[0.24; 0.62; 1; 1.88; 11.86; 29.46; 84.01; 164.8];
   P.eccentricity=[0.206; 0.007; 0.017; 0.093; 0.048; ...
        0.054; 0.047; 0.009];
   P.rotation=[58.64;-243.02;1;1.03;0.41;0.43;-0.72;0.67];
   P.moon=[0; 0; 1; 2; 69; 62; 27; 14];
   P.ring={'无';'无';'无';'无';'有';'有';'有';'有'};
```

该结构体的显示格式为

```
  包含以下字段的struct:
           name: [8×3 char]
       diameter: [8×1 double]
           mass: [8×1 double]
           axis: [8×1 double]
         period: [8×1 double]
```

```
    eccentricity: [8×1 double]
        rotation: [8×1 double]
            moon: [8×1 double]
            ring: 8×1 cell
```

有了结构体变量P，则可以由下面语句直接求解例2-24，得出完全一致的结果，且复杂程度比较接近，M 为各个行星的质量向量。

```
>> M=5.965e24*P.mass; M(5) %重新计算木星质量
```

MATLAB也提供了一系列结构体数据结构的转换函数，如struct2cell()与struct2table()，可以将结构体数据直接转换为单元数组与表格变量。此外，有时还可以使用table2array()函数，不过对P这样既包含数字又包含字符串的结构体数据结构则不能进行转换。

2.4.5 其他数据结构

类（class）是MATLAB面向对象编程的重要数据结构。MATLAB允许用户自己编写包含各种复杂信息的变量，称为类变量，类变量可以包含各种下级的信息，也称为域或成员变量，还可以重新对类进行各种底层运算，称为重载函数（overload function）。每一个类的实例又称为对象（object）。类与对象编程在很多领域都特别有用，第9章将通过例子专门介绍相关的编程方法。

2.5 MATLAB的基本语句结构

MATLAB的语句有两种最基本的结构——直接赋值结构和函数调用结构。除此之外，还将介绍冒号表达式与子矩阵提取等方面的内容。

2.5.1 直接赋值语句

直接赋值语句的基本结构为赋值变量＝表达式，这一过程把等号右边的“表达式”直接赋给左边的“赋值变量”，并返回到MATLAB的工作空间。如果赋值表达式后面没有分号，则将在MATLAB命令窗口中显示表达式的运算结果。若不想显示运算结果，则应该在赋值语句的末尾加一个分号。如果省略了赋值变量和等号，则表达式运算的结果将赋给保留变量ans。所以说，保留变量ans将永远存放最近一次无赋值变量语句的运算结果。

其实，前面已经有大量的例子演示了直接赋值语句。这里侧重于利用直接赋值语句处理符号函数及符号表达式的输入与处理方法。

例2-26 若 $f(x)=x^2-x-1$，试求 $f(f(f(f(f(f(f(f(f(f(x)))))))))$。如果结果是多项式，多项式的最高阶次是多少？

解　最简单的方式是由符号函数格式描述$f(x)$，这样，看起来比较复杂的复合函数可以由下面的直接嵌套方法求出来。得出的多项式可以由expand()函数展开：

```
>> syms x; f(x)=x^2-x-1;
   F(x)=f(f(f(f(f(f(f(f(f(x)))))))))，F1=expand(F)
```

展开的多项式如下，可见该多项式是1024次多项式。

$$F_1(x)=x^{1024}-512x^{1023}+130048x^{1022}-21846272x^{1021}+\cdots$$

2.5.2　函数调用语句

另一种常用的赋值语句格式为函数的格式，函数是MATLAB的主流编程方式。函数调用的基本结构为

[返回变元列表]=fun_name(输入变元列表)

其中fun_name为函数名，其命名的要求和变量名的要求是一致的，一般函数名应该对应在MATLAB路径下的一个文件。例如，函数名my_fun应该对应于my_fun.m文件。当然，还有一些函数名需对应于MATLAB内核中的内核函数（built-in function），如inv()函数等。

“返回变元列表”和“输入变元列表”均可以由若干个变量名组成，它们之间应该分别用逗号分隔。返回变元还允许用空格分隔，例如[$\boldsymbol{U}$ $\boldsymbol{S}$ $\boldsymbol{V}$]=svd($\boldsymbol{X}$)，该函数对给定的$\boldsymbol{X}$矩阵进行奇异值分解，所得的结果由$\boldsymbol{U}$、$\boldsymbol{S}$、$\boldsymbol{V}$这三个变量返回。

2.5.3　多样的函数调用机制

MATLAB提供了灵活的执行机制，允许用户用不同的格式调用相同的函数，比如，MATLAB提供了内核函数eig($\boldsymbol{A}$)，可以直接计算给定矩阵的特征值；如果使用调用格式[$\boldsymbol{V}$,$\boldsymbol{D}$]=eig($\boldsymbol{A}$)，则除了返回特征值$\boldsymbol{D}$之外，还将返回特征向量矩阵$\boldsymbol{V}$；如果采用调用格式eig($\boldsymbol{A}$,$\boldsymbol{B}$)，则将求解广义特征值问题。

除此之外，MATLAB在不同的工具箱中提供了同名的eig()函数，比如符号运算工具箱提供的eig()函数可以求取符号矩阵特征值的解析解，控制系统工具箱提供的eig()函数可以求出线性系统的极点。MATLAB语言有比较好的执行机制，在调用这些同名函数时不会出现混淆。在此执行机制下，先识别输入变元是什么数据类型，然后调用相应数据类型下的eig()函数，得出对应的结果。

MATLAB函数的格式、编写方法、技巧与调试方法将在第5章中详细介绍。

2.5.4　冒号表达式

冒号表达式是MATLAB中很有用的表达式，在向量生成、子矩阵提取等很多方面都是特别重要的。冒号表达式的格式为$\boldsymbol{v}$=s_1:s_2:s_3，该函数将生成一个行向

量$\boldsymbol{v}$，其中s_1为向量的起始值，s_2为步距，该向量将从s_1出发，每隔步距s_2取一个点，直至不超过s_3的最大值就可以构成一个向量。若省略s_2，则步距取默认值1。

例2-27 试探不同的步距，从$t\in[0,\pi]$区间取出一些点构成向量。

解 先试一下步距0.2，这样可以用下面的语句生成一个向量：

```
>> v1=0:0.2:pi  %注意，最终取值为3而不是π，因为下一个点3.2大于π
```

该语句将生成行向量$\boldsymbol{v}_1=[0,0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8,\ 2,2.2,2.4,2.6,2.8,3]$。

下面还将尝试冒号表达式不同的写法，并得出如下的结果：

```
>> v2=0:-0.1:pi, v3=0:pi, v4=pi:-1:0 %对照结果理解不同的冒号表达式
```

产生的$\boldsymbol{v}_2$向量为1×0空矩阵，$\boldsymbol{v}_3=[0,1,2,3]$，$\boldsymbol{v}_4=[3.1416,2.1416,1.1416,0.1416]$。

例2-28 试找出1～1000内所有能被13整除的整数。

解 如果逐个数去判定每个数是不是能被13整除需要循环运算，比较麻烦。解决这样的问题还可以换一个思路：第一个能被13整除的是多少呢？显然是$a_1=13$，第二个呢？$a_2=a_1+13$。以后的各个数分别是$a_3=a_2+13$，$a_4=a_3+13$，$\cdots$，显然，这些数是从13开始，以13为步距生成的一组数，由冒号表达式可以直接生成这些数据，所以可以用下面的MATLAB命令直接解决问题：

```
>> A=13: 13: 1000
```

2.5.5 子矩阵的提取

提取子矩阵的具体方法是$\boldsymbol{B}$=$\boldsymbol{A}$($\boldsymbol{v}_1$,$\boldsymbol{v}_2$)，其中$\boldsymbol{v}_1$向量表示子矩阵要保留的行号构成的向量，$\boldsymbol{v}_2$表示要保留的列号构成的向量，这样从$\boldsymbol{A}$矩阵中提取有关的行和列，就可以构成子矩阵$\boldsymbol{B}$了。若$\boldsymbol{v}_1$为:，则表示要提取所有的行，$\boldsymbol{v}_2$亦有相应的处理结果。关键词end表示最后一行（或列，取决于其位置）。

如果想删除矩阵的第i行元素，可以给出简单的命令$\boldsymbol{A}(i,:)$=[]。

例2-29 下面将列出若干命令，并加以解释，读者可以自己由测试矩阵体会这些子矩阵提取语句。

```
>> A=[1,2,3; 4 5,6; 7,8 0]; %矩阵输入。由于语句末尾有分号，矩阵不显示
   B1=A(1:2:end,:)          %提取A矩阵全部奇数行、所有列
   B2=A([3,2,1],[1,1,1])    %提取A矩阵3、2、1行，由首列构成矩阵
   B3=A(:,end:-1:1)         %将A矩阵左右翻转，即最后一列排在最前面
   A(2,:)=[]; A(:,3)=[]     %删除A矩阵的第2行第3列
```

上述的语句将生成下面的各个矩阵：

$$\boldsymbol{B}_1=\begin{bmatrix}1&2&3\\7&8&0\end{bmatrix},\quad \boldsymbol{B}_2=\begin{bmatrix}7&7&7\\4&4&4\\1&1&1\end{bmatrix},\quad \boldsymbol{B}_3=\begin{bmatrix}3&2&1\\6&5&4\\0&8&7\end{bmatrix},\quad \boldsymbol{A}=\begin{bmatrix}1&2\\7&8\end{bmatrix}$$

例2-30 在线性代数中，矩阵$\boldsymbol{A}$的第(i,j)代数余子式定义为矩阵删除第i行第j

列后剩余部分的矩阵行列式乘以$(-1)^{i+j}$，试用MATLAB实现代数余子式计算。

解 学会了矩阵行、列的删除方法，不难给出下面的命令求取代数余子式：

```
>> B=A; B(i,:)=[]; B(:,j)=[]; d=(-1)^(i+j)*det(B)
```

2.5.6 等间距行向量的生成

如果给定了间距，则可以通过冒号表达式生成一组等间距的数据点，但这样的生成方法有时可能漏掉重要的点。例如，如果想在$[0,\pi]$区间内生成一组等间距点，并选择0.1为步距，则用户会很自然地给出命令$\boldsymbol{v}$=0: 0.1: pi，不过观察结果会发现，向量最后一个点为3.1，而不是期望的π，这是因为利用现有的规则，下一个点应该为3.2，但这个数大于π，所以这样一个点就被自动略去了。如何保证这样一个点呢？可以考虑将步距选择为$\pi/30$，再调用冒号表达式生成等间距样本点。

MATLAB还提供了另外两个函数生成不同的等间距行向量：

（1）**线性等间距点的生成**。可以使用MATLAB函数$\boldsymbol{v}$=linspace(n_1,n_2,N)，其中，向量的点数为N（默认值为50），起始值为n_1，终止值为n_2。这样的向量又称为线性等间距向量。

（2）**对数等间距点的生成**。在某些特定的领域，要求以对数等间距生成行向量，则需调用logspace()函数生成行向量了，其调用格式为$\boldsymbol{w}$=logspace(n_1,n_2,N)，该语句将生成N个数据点的行向量，其第一个值为$\lg n_1$，最后一个值为$\lg n_2$。若想在$[10^{-3},10^4]$频率段内生成30个点，则需要给出命令$\boldsymbol{w}$=logspace(-3,4,30)。

2.6 数据文件的读取与存储

在科学研究中难免会有大量结果需要存成文件保留起来，还可能需要通过文件传递给其他软件，也需要从存储的文件或其他软件生成的文件读取数据。所以，本节将介绍一般数据文件与Excel文件的读写方法。

2.6.1 数据文件的读取与存储命令

前面曾经提及，用load与save命令可以进行数据文件的读写，这里将更详细地探讨这两个函数的使用方法。

save命令的调用格式为save 文件名 变量列表，其中“文件名”为字符串，文件的存储地点为当前文件夹。“变量列表”中的变量名用空格分隔，不能用逗号或其他符号分隔。如果要把当前工作空间中的全部变量都存起来，则可以不给出“变量列表”。另外，如果不给出“文件名”，则默认存入文件matlab.mat。

如果想用ASCII码格式存储文件，则需给出-ascii选项如下：

```
save 文件名 -ascii 变量列表
```

如果文件名或路径名中含有空格，直接使用load与save命令会导致错误，必须使用load()与save()函数。

save(文件名,'a_1','a_2',$\cdots$,'a_m')，a=load(文件名)

其中“文件名”可以是文件名，也可以是带有路径信息的绝对文件名，是用字符串表示的，允许带有空格。注意，在调用这个函数时，需要存储的变量名一定要以字符串的形式给出。

例2-31 试将例2-23中的表格数据存入my data.dat文件中。

解 如果想将结果存入my data.dat文件（该文件名有意保留了一个空格），用save命令可能引起歧义，所以只能使用save()函数。

```
>> load c2dtab; save('my data.dat','planet')
```

2.6.2 文件读写的底层方法

类似C语言，MATLAB提供了一整套文件读写的底层函数，归纳如下：

（1）fopen()函数可以打开一个文件，其调用格式为

[fid,错误信息]=fopen(文件名,选项)

其中“选项”的可选值为'r'（作为只读文件打开，为默认选项）、'w'（建立可写的空白文件，如果该文件非空，则清空该文件）等，还有其他选项，具体见该函数的帮助信息。返回的fid为文件句柄，若其值为-1表示操作文件不成功，“错误信息”返回错误信息字符串。该函数还可以带有其他的输入变元，如编码格式等，具体请参见该函数的联机帮助信息。

（2）判断文件是否结束，可以给出key=feof(fid)，如果其值为1表示前面一条文件操作语句已到了文件末尾，否则表示未到文件末尾。

（3）以字符串形式读入一行内容str=fgetl(fid)。

（4）文件读写的底层函数——fscanf()与fprintf()，这两个函数与它们的C语言原型函数的调用格式很接近，调用格式如下：

a=fscanf(fid,格式)，fprintf(fid,格式,a_1,a_2,$\cdots$,a_n)

其中“格式”为读写格式控制字符串，与前面介绍的sprintf()函数一致。

（5）可以用fclose(fid)命令关闭文件，如果关闭文件不成功，返回-1。

例2-32 试用底层命令编写一个能显示ASCII码文本文件的小程序。

解 用MATLAB的type命令可以完全实现，但该命令不能进一步控制格式，比如说在每行代码后面显示一个空行。本例子仅用于演示文件读写的底层函数使用方法。用下面的第一条语句可以打开纯文本文件magic.m，然后用循环结构逐条显示该文本文件，直至文件结束（即feof()函数返回1），再关闭该文件。关于循环语句与循环结构

相关的内容后面将专门介绍。

```
>> f=fopen('magic.m') %打开一个文件,获得文件句柄f
   while feof(f)~=1, disp(fgetl(f)); disp(' '), end,  fclose(f)
```

2.6.3　Excel文件的读取与存储

MATLAB提供的xlsread()函数可以直接从Microsoft Excel文件中提取数据,xlsread()函数的调用格式如下:

[$\boldsymbol{N}$,TXT,RAW]=xlsread(文件名,表单序号,范围)

其中"表单序号"为Excel工作表序号,"范围"为字符串,给出的是Excel格式的范围表示,比如,如果想将Excel文件的第B列到第F列、第3行到第20行范围内的数据读入MATLAB工作空间,"范围"可以设置为'B3:F20'。这时,该范围内的数值数据将由$\boldsymbol{N}$矩阵返回,Excel各个单元格的文本表示将由TXT返回,而Excel文件的原始信息将由RAW返回。

可以调用xlrwrite()命令将MATLAB工作空间中的变量直接写到Excel文件中,其调用格式为

xlswrite(文件名,变量名,表单序号,范围)

其中"变量名"为要写入的变量名,它可以是矩阵,也可以是二维常规的单元数组,"表单序号"与"范围"选项的定义与前面一致。另外,如果将其选作'A1'表示从左上角写入,如果选作C2则表示从Excel工作表的第C列、第二行开始写入。如果从左上角写起,则在函数调用时可以略去后两个变元。

值得指出的是,如果要写入的文件处于打开状态,则xlswrite()函数调用将失败,并给出相应的错误信息。需要先关闭文件再写入。

例2-33　考虑例2-23中的表格数据,试将其存入Excel文件。

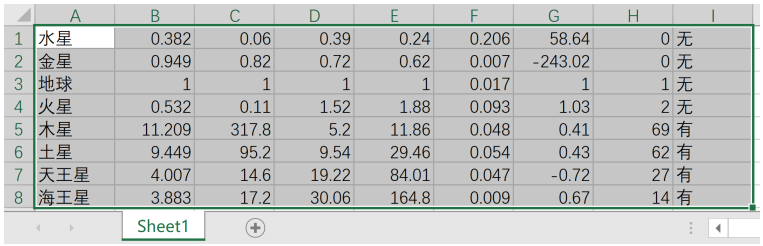
解　遗憾的是,xlswrite()函数不支持表格数据的处理,需要先将其转换成单元数组的形式。所以可以给出下面语句,从Excel文件左上角开始写信息。自动生成的Excel文件test.xls截图如图2-6所示,和所期望的形式完全一致。不过与图2-5给出的界面相比,这样的文件是没有表头的。

```
>> load c2dtab, C=table2cell(planet); %读入表格并转换成单元数组格式
   xlswrite('test',C,1,'A1')          %这里,后两个输入变元可以省略
```

关闭这个Excel文件,再给出下面的命令,从第C列第二行写起,则得出如图2-7所示的更新Excel文件。这两次写入的重叠部分以新写入的为准。

```
>> xlswrite('test',C,1,'C2')  %从第C列第2行位置开始写入
```

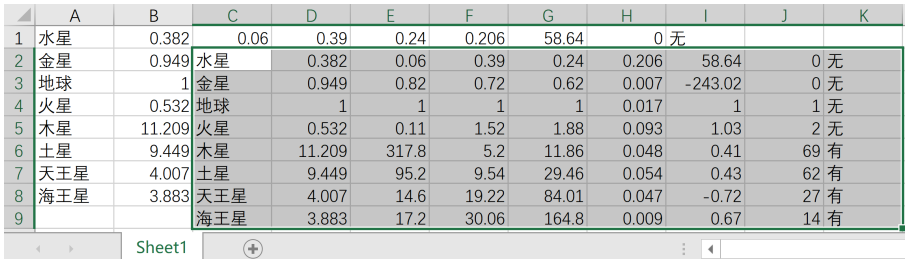
相比之下,例2-25给出的结构体数据若由struct2cell()函数转换后,则不能写入Excel文件,可以考虑变通方法,先将其转换成表格数据,再转换成单元数组,再重复



	A	B	C	D	E	F	G	H	I
1	水星	0.382	0.06	0.39	0.24	0.206	58.64	0	无
2	金星	0.949	0.82	0.72	0.62	0.007	-243.02	0	无
3	地球	1	1	1	1	0.017	1	1	无
4	火星	0.532	0.11	1.52	1.88	0.093	1.03	2	无
5	木星	11.209	317.8	5.2	11.86	0.048	0.41	69	有
6	土星	9.449	95.2	9.54	29.46	0.054	0.43	62	有
7	天王星	4.007	14.6	19.22	84.01	0.047	-0.72	27	有
8	海王星	3.883	17.2	30.06	164.8	0.009	0.67	14	有

Sheet1

图 2-6 生成的 Excel 文件截图



	A	B	C	D	E	F	G	H	I	J	K
1	水星	0.382	0.06	0.39	0.24	0.206	58.64	0	无		
2	金星	0.949	水星	0.382	0.06	0.39	0.24	0.206	58.64	0	无
3	地球	1	金星	0.949	0.82	0.72	0.62	0.007	-243.02	0	无
4	火星	0.532	地球	1	1	1	1	0.017	1	1	无
5	木星	11.209	火星	0.532	0.11	1.52	1.88	0.093	1.03	2	无
6	土星	9.449	木星	11.209	317.8	5.2	11.86	0.048	0.41	69	有
7	天王星	4.007	土星	9.449	95.2	9.54	29.46	0.054	0.43	62	有
8	海王星	3.883	天王星	4.007	14.6	19.22	84.01	0.047	-0.72	27	有
9			海王星	3.883	17.2	30.06	164.8	0.009	0.67	14	有

Sheet1

图 2-7 更新后的 Excel 文件截图

上述过程则可以存入 Excel 文件。

现在考虑最新生成的 Excel 文件 `test.xls`，如果调用 `xlsread()` 函数

```
>> [N,txt,raw]=xlsread('test.xls')
```

则可以得出如下的数值矩阵，可见，在数值区域内所有的数值由矩阵返回，如果对应的矩阵元素不是数值或为空，则自动填写 NaN。

$$\boldsymbol{N}=\begin{bmatrix} 0.382 & 0.06 & 0.39 & 0.24 & 0.206 & 58.64 & 0 & \text{NaN} & \text{NaN} \\ 0.949 & \text{NaN} & 0.382 & 0.06 & 0.39 & 0.24 & 0.206 & 58.64 & 0 \\ 1 & \text{NaN} & 0.949 & 0.82 & 0.72 & 0.62 & 0.007 & -243.02 & 0 \\ 0.532 & \text{NaN} & 1 & 1 & 1 & 1 & 0.017 & 1 & 1 \\ 11.209 & \text{NaN} & 0.532 & 0.11 & 1.52 & 1.88 & 0.093 & 1.03 & 2 \\ 9.449 & \text{NaN} & 11.209 & 317.8 & 5.2 & 11.86 & 0.048 & 0.41 & 69 \\ 4.007 & \text{NaN} & 9.449 & 95.2 & 9.54 & 29.46 & 0.054 & 0.43 & 62 \\ 3.883 & \text{NaN} & 4.007 & 14.6 & 19.22 & 84.01 & 0.047 & -0.72 & 27 \\ \text{NaN} & \text{NaN} & 3.883 & 17.2 & 30.06 & 164.8 & 0.009 & 0.67 & 14 \end{bmatrix}$$

其他全部信息在 `raw` 变量返回，这里显示从略。

本章习题

2.1 启动 MATLAB 环境，并给出语句

```
>> tic, A=rand(500); B=inv(A); norm(A*B-eye(500)), toc
```

试运行该语句，观察得出的结果，并利用 `help` 或 `doc` 命令对不熟悉的语句进行帮助信息查询，逐条给出上述程序段与结果的解释。

2.2 试对任意整数 k 化简表达式 $\sin(k\pi+\pi/6)$。

2.3 试判定 a=5; key=isinteger(a)语句的执行结果。key 是什么？

2.4 试求出无理数 $\sqrt{2}$、$\sqrt[6]{11}$、$\sin 1^\circ$、e^2、$\ln(21)$、$\log_2(\mathrm{e})$ 的前 200 位有效数字。

2.5 如果想精确地求出 lg(12345678)，试判断下面哪个命令是正确的。

（1）vpa(log10(sym(12345678)))

（2）vpa(sym(log10(12345678)))

2.6 试证明恒等式

（1）$\mathrm{e}^{\mathrm{j}\pi}+1=0$

（2）$\dfrac{1-2\sin\alpha\cos\alpha}{\cos^2\alpha-\sin^2\alpha}=\dfrac{1-\tan\alpha}{1+\tan\alpha}$

2.7 可以由 $\boldsymbol{A}$=rand(3,4,5,6,7,8,9,10,11)命令生成一个多维伪随机数数组。试判定一共生成了多少个随机数，这些随机数的均值是多少。

2.8 用 MATLAB 语句输入矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，

$$\boldsymbol{A}=\begin{bmatrix}1&2&3&4\\4&3&2&1\\2&3&4&1\\3&2&4&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1+4\mathrm{j}&2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}\\4+1\mathrm{j}&3+2\mathrm{j}&2+3\mathrm{j}&1+4\mathrm{j}\\2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\\3+2\mathrm{j}&2+3\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\end{bmatrix}$$

前面给出的是 4×4 矩阵，如果给出 $\boldsymbol{A}(5,6)=5$ 的命令将得出什么结果？

2.9 试将下面的分块矩阵拆分成 3×3 的单元数组。

$$\boldsymbol{A}=\left[\begin{array}{cc:cccc:cc}3&2&1&3&2&2&3&3\\\hdashline1&3&1&2&1&2&2&2\\2&1&3&3&2&3&3&1\\3&1&3&1&2&2&1&1\\\hdashline1&3&2&2&2&3&1&2\end{array}\right]$$

2.10 已知数学函数 $f(x)=\dfrac{x\sin x}{\sqrt{x^2+2}(x+5)}$，$g(x)=\tan x$，试求出复合函数 $f(g(x))$ 和 $g(f(x))$。

2.11 由于双精度数据结构有一定的位数限制，大数的阶乘很难保留足够的精度。试用数值方法和符号运算的方法计算并比较 C_{50}^{10}，其中 $\mathrm{C}_m^n=m!/\big(n!(m-n)!\big)$。符号运算工具箱还提供了函数 nchoosek() 专门的计算组合问题，其调用的格式为 nchoosek(sym(m),n)。

2.12 试列出大于 -100 的所有可以被 11 整除的负整数，并找出 [3000,5000] 区间内所有可以被 11 整除的正整数。

2.13 假设已知矩阵 $\boldsymbol{A}$，试给出相应的 MATLAB 命令，将其全部偶数行提取出来，赋给 $\boldsymbol{B}$ 矩阵，用 $\boldsymbol{A}$=magic(8) 命令生成 $\boldsymbol{A}$ 矩阵，用上述命令检验一下结果是否正确。

2.14 试将 100×100 的魔方矩阵的第 2～33 列存入 Excel 文件。

2.15 试将字符串'Do you speak MATLAB?'中的'a'和'A'的位置都找出来，并改变其大小写。

2.16 试用变量编辑界面可视地编辑工作空间中的矩阵变量，具体的方法：单击“打开变量”右侧的黑三角符号，从列表中选择想编辑的变量名，并打开编辑界面。