第 3 章　MATLAB 基础

本章内容

- 什么是 MATLAB？为什么它被选作本书的工具？
- MATLAB 提供了什么样的编程环境？
- 什么是 M-文件？
- MATLAB 脚本和函数间的差别是什么？
- 如何能启动 MATLAB？

3.1　MATLAB 介绍

MATLAB（MATrix LABoratory）是一个数据分析、原型设计和可视化的工具，它内置对矩阵和矩阵操作、优秀的图形处理能力、高层编码语言和开发环境的支持。

MATLAB 已在工程师、科学家以及工业界和学术界研究人员中非常流行，这源于多个因素，如它具有丰富的特殊函数（嵌入在工具箱中）并适用于很多重要领域，从神经网络到金融到图像处理（这是本书主要关注点，将在第 4 章讨论）[1]。

MATLAB 有大量内置的文档。它包含对 MATLAB 主函数的描述，示例程序码，相关的演示，以及通用的帮助页。MATLAB 文档可以用许多不同的方式访问，从仅文字的命令行帮助到超链接的 HTML 网页（可在 MathWorks 网站：http://www.mathworks.com 找到它们的在线版本）。

MATLAB 的基本数据种类是矩阵（或**数组**）。MATLAB 不需要标注尺寸，即不需要在实际使用前分配存储器。所有数据都被看作某种形式的矩阵。单个数值被 MATLAB 看作是 1×1 的矩阵。

MATLAB 的算法开发环境提供一个命令行界面，一个 MATLAB 程序语言翻译器，一组丰富的数字和字符操作函数，2-D 和 3-D 绘图函数，以及构建图形用户界面（GUI）的能力。MATLAB 程序语言翻译命令，这通过消除编译而缩短了编程的时间。

如果你想以正确的方式开始 MATLAB，现在可参见 3.5 节。

3.2　MATLAB 的基本元素

在本节中，将介绍 MATLAB 的**基本元素**，它的环境，数据种类，以及命令行操作。

1　在 2003 年，MathWorks 发行了图像采集工具箱，它包含一系列图像采集函数。该工具箱将在 5.3.2 小节概括介绍，更多细节将不在本书中展开。

3.2.1 工作环境

MATLAB 的**工作环境**包括：

- MATLAB 桌面：它一般包括 5 个子窗：命令窗，工作区域浏览器，当前目录窗，命令历史窗，以及一个或多个图窗（当用户显示一个图形，如图表或图像时可见）。
- MATLAB 编辑器：它用来生成和编辑 M-文件。它包含一系列用来存储、观看、调试 M-文件的函数。
- 帮助系统：它主要包括帮助浏览器，可显示 HTML 文档，并有许多搜索和显示的选项。

在 3.5 节中，你将会亲手参加到对 MATLAB 环境的介绍中。

3.2.2 数据种类

MATLAB 支持的**数据种类**（或**数据类别**）列在表 3.1 中。

表 3.1 MATLAB 数据类别

数据类	描 述
uint8	8-比特无符号整数（每个元素 1 个字节）
uint16	16-比特无符号整数（每个元素 2 个字节）
uint32	32-比特无符号整数（每个元素 4 个字节）
int8	8-比特有符号整数（每个元素 1 个字节）
int16	16-比特有符号整数（每个元素 2 个字节）
int32	32-比特有符号整数（每个元素 4 个字节）
single	单精度浮点数（每个元素 4 个字节）
double	双精度浮点数（每个元素 8 个字节）
logical	值为 0（假）或 1（真）（每个元素 1 个字节）
char	字符（每个元素 2 个字节）

表中列出的前 8 种数据类型称为数字类，MATLAB 中的一个数字值除非另外指出都属于类 double。数据类间的转换（也称类型转换）是可能的（且常常需要）。MATLAB 中的一个字符串就是一个简单的 $1 \times n$ 字符数组。

3.2.3 MATLAB 中的数组和矩阵索引

MATLAB **数组**（矢量和矩阵）使用基于 1 的约定来索引。因此，a(1) 根据句法指一个 1-D 数组 a 的第 1 个元素，f(1,1) 根据句法指一个 2-D 数组第 1 个元素，如在一幅单色图像 f 中左上角的像素。冒号（:）操作符提供了强有力的索引能力，如你将在教程 3.2 中所见。

MATLAB 并不限制一个数组的维数，但对一个数组允许具有的最大数量的元素有一个上限。如果你键入

```
[c, maxsize] = computer
```

变量 maxsize 将包含你使用的计算机和 MATLAB 版本所允许的一个数组具有的最大元素数量。

3.2.4　标准数组

MATLAB 有许多有用的标准**内置数组**：

- zeros(m, n)：生成一个 $m \times n$ 的 0 数组。
- ones(m, n)：生成一个 $m \times n$ 的 1 数组。
- true(m, n)：生成一个 $m \times n$ 的逻辑 1 数组。
- false(m, n)：生成一个 $m \times n$ 的逻辑 0 数组。
- eye(n)：返回一个 $n \times n$ 的单位矩阵。
- magic(m)：返回一个阶为 m 的魔术平方[1]。
- read(m, n)：生成一个 $m \times n$ 的矩阵，它的元素是在[0，1]范围内均匀分布的伪随机数。
- readn(m, n)：生成一个 $m \times n$ 的矩阵，它的元素是满足均值为 0，方差为 1 的高斯分布的伪随机数。

3.2.5　命令行操作

MATLAB 中的许多功能可借助命令行来访问（可在 3.6 节的第 1 步看到）。一个**命令行操作**等价于执行一行代码，通常是在提示符（>>）后键入一个内置函数的名称和参数来调用这个函数。如果这个行没有包括结果需要赋予的变量，结果将赋给一个内置的变量 ans（如在 answer 中）。如果这个行不是以一个分号结束，结果将返回给命令窗。

考虑到在 MATLAB 中有大量可用的内置函数（以及它们可能使用的参数），命令行界面是一种非常有效地访问这些函数而不需要求助于一系列复杂的菜单和对话框的方式。用来学习有用函数的名称、句法，以及参数的时间是值得的，因为在命令行上（一次一个命令）的任何工作形式将也是一个大程序或用户定义函数的一部分。另外，MATLAB 还提供了更多的方式以使命令行的交互更有效，如很容易访问（用方向键）先前的操作（并节省键入时间）。成功的命令行操作也可以从命令历史窗中选择并结合进一个脚本。

3.3　编程工具：脚本和函数

MATLAB 开发环境翻译用 MATLAB 程序语言编写的命令，这当主要目标是快速地写出一个样本算法时非常方便。一旦一个程序稳定了，它可用 MATLAB 编译器（作为主产品的附件提供）来编译以更快地执行，这对大数据集是特别重要的。MATLAB 编译器 MATCOM 将 MATLAB 原生代码转换成 C++码，编译 C ++码，并将它与 MATLAB 库链接。

1　一个魔术平方是一个方数组，其中沿任何行、列、或主对角线的和都相同。

编译的代码可以比它们解释的代码快 10 倍，但加速因子依赖于原始的码是如何矢量化的；非常优化的矢量化码有可能无法再加速。

为了较快地计算，程序员可以通过 MEX 动态地将 C 例程按 MATLAB 函数链接。

3.3.1 M-文件

MATLAB 中的 **M-文件**可以是执行一系列 MATLAB 命令（或声明）的**脚本**，也可以是接受变量（参数）并产生一个或多个输出值的函数。用户生成的函数扩展了 MATLAB 和它的工具箱满足特定需求和应用的能力。

一个 M-文件包含一个由一系列需要解释和执行的命令构成的脚本。除调用内置函数外，脚本还可以包含变量声明，调用用户定义的函数（这可存在分离的文件中），判断语句，以及重复循环。脚本常用一个文字编辑器（如 MATLAB 内置编辑器）来生成，并用一个有意义的名称和.m 扩展来存储。一旦一个脚本生成和存储后，它就可通过简单地键入它的名称而由命令行来调用。

一个包含一个**函数**的 M-文件具有如下组元。

- 函数定义行：它的形式为

  ```
  function[output] = function_name(inputs)
  ```

  需要有一个关键词 function。输出变量用方括号括住，而输入变量用圆括号括住。如果函数不返回任何值，只需要使用词 function 且不需要方括号或等号。函数名称必须用一个字母开头，不能包含空格，且长度要限制在 63 个字符内。
- H1 行：它是一个单独的命令行，接在函数定义行后。在 H1 行和函数定义行之间不能有空行或前导空格。H1 行是用户键入下列命令时首先在命令窗中出现的文字：

  ```
  >>help function_name
  ```
  [1]

  因为这个行提供了有关 M-文件的内容和目的的重要摘要信息，它应该尽可能具有描述性。
- 帮助文字：它是接着 H1 行的一个文字块，两者之间没有空行。当用户键入下列命令时：

  ```
  >>help function_name
  ```

  帮助文字将会显示在 H1 行后。
- 函数体：它包含执行计算和对输出参数[2]赋值的所有 MATLAB 代码。
- 注释：它在 MATLAB 中由符号%引导。

下面是一个将用在教程 3.3 中的简单函数（raise_to_power）的示例：

```
function z = raise_to_power(val, exp)
```

1 注意在 MATLAB 的教学版本中提示符是不同的（EDU>>）。

2 尽管 MATLAB 程序语言包含“return”命令，它并没有任何变量。这与其他程序语言不同，其他语言可使用“return”后接一个参数来返回单个数值或一个指向存储器中多个值的指针。

```
%RAISE_TO_POWER Calculate power of a value
% z = raise_to_power(val, exp) raise val to a power with value of exp
% and store it in z.
z = val ^ exp;
```

3.3.2　操作符

MATLAB 操作符可归成如下 3 类。

- 算术操作符：对矩阵执行数值计算。
- 关系操作符：比较操作数。
- 逻辑操作符：执行标准的逻辑函数（如，AND，NOT，以及 OR）。

因为 MATLAB 使用矩阵作为它标准的内置数据类型，MATLAB 中有关数组和矩阵的操作符数量远远超过在常规程序语言中传统的操作符数量。表 3.2 包含了对它们的一个总结。所有的操作数都可以是实数或复数。

表 3.2　MATLAB 数组和矩阵算术操作

操作符	名　　称	MATLAB 函数
+	数组和矩阵加法	plus(a, b)
-	数组和矩阵减法	minus(a, b)
.*	逐元素数组乘法	times(a, b)
*	矩阵乘法	mtimes(a, b)
./	数组右除	rdivide(a, b)
.\	数组左除	ldivide(a, b)
/	矩阵右除	mrdivide(a, b)
\	矩阵左除	mldivide(a, b)
.^	数组幂	power(a, b)
.^	矩阵幂	mpower(a, b)
.'	矢量和矩阵转置	transpose(a)
'	矢量和矩阵复数共轭转置	ctranspose(a)
+	一元加法	uplus(a)
-	一元减法	uminus(a)

表 3.3 给出一些最有用的，也可方便地在 MATLAB 中执行的特殊矩阵操作。

表 3.3　MATLAB 特殊的矩阵操作示例

名　　称	MATLAB 操作或函数	名　　称	MATLAB 操作或函数
矩阵转置	撇号（'）操作符	左右翻转	fliplr 函数
逆	inv 函数	矩阵旋转	rot90 函数
矩阵行列式	det 函数	矩阵重整	reshape 函数
上下翻转	flipud 函数	对角元素之和	trace 函数

因为单色图像本质上是 2-D 数组，即矩阵，所有表 3.2 和表 3.3 的操作数对图像都适用。

但是 MATLAB 的图像处理工具包（IPT）还包括涉及图像的主要算术操作的特殊版本，其主要优点是支持整数数据类型（MATLAB 数学操作符要求类 double 的输入）。这些函数列在表 3.4 中，并将在第 6 章中详细讨论。

表 3.4 IPT 支持的特殊算术函数

函 数	描 述
imadd	将两幅图像相加或将一个常数加到一幅图像上
imsubtract	将两幅图像相减或从一幅图像中减去一个常数
immultiply	将两幅图像相乘（逐点）或用一个常数去乘一幅图像
imdivide	将两幅图像相除（逐点）或用一个常数去除一幅图像
imabsdiff	计算两幅图像之间的绝对差值
imcomplement	对一幅图像取反
imlincomb	计算两幅或多幅图像的线性组合

在 MATLAB 中的关系操作符与任何程序语言中的关系操作符类似，它们列在表 3.5 中。

表 3.5 关系操作符

操作符	名 称	操作符	名 称
<	小于	>=	大于或等于
<=	小于或等于	==	等于
>	大于	~=	不等于

MATLAB 包括一组标准的逻辑操作，它们列在表 3.6 中。MATLAB 也支持列在表 3.7 中的逻辑函数，以及大量返回逻辑 1（真）或逻辑 0（假）的函数，这依赖于它们变量的值或条件是 true 或 false，如 isempty(a)，isequal(a)，isnumeric(a)，以及许多其他函数（参见 MATLAB 在线文档）。

表 3.6 逻辑操作符

操作符	名 称	操作符	名 称
&	AND	~	NOT
|	OR		

表 3.7 逻辑函数

函 数	描 述
XOR	在两个操作数间执行异或（XOR）
all	如果一个矢量的所有元素都非零返回 1，否则返回 0。对矩阵逐行进行
any	如果一个矢量的任一元素为非零返回 1，否则返回 0。对矩阵逐行进行

3.3.3 重要变量和常量

MATLAB 有很多**内置变量**和**内置常量**，部分列在表 3.8 中。

表 3.8　选出来的内置变量和内置常量

名　称	返回值	名　称	返回值
ans	最近的回答	NaN（或 nan）	非数值（如 0/0 的结果）
eps	浮点相对精度	Inf	无穷（用 0 除的结果）
I（或 j）	虚部单位（$\sqrt{-1}$）		

3.3.4　数字表示

MATLAB 可用传统的十进制记数法（具有可选的小数点和前置正负号）以及科学计数法（使用字母 e 表示指数为 10 的幂）来表示数字。复数可使用 i 或 j 作为虚部的前缀来表达。

所有数字都在内部用 IEEE 的浮点标准存储，所得到的范围大约在 10^{-308}～10^{+308}。

3.3.5　流程控制

MATLAB 程序语言支持在大多数其他现代高层程序语言中常用的**流程控制**语句：if（选项为 else 和 elseif）和 switch 判断语句，for 和 while 循环和关联语句（break 和 continue），以及用于出错处理的 try…catch 块。对特定的句法细节，请参考在线文档。

3.3.6　代码优化

作为本质上面向矩阵的结果，MATLAB 程序语言是一种矢量化的语言，这表示它可对结合成矢量或矩阵的数字执行很多操作而不需要明确的循环语句。矢量化的代码更加紧凑，更加有效，且可并行化[1]。除了使用矢量化的循环，MATLAB 程序员还被鼓励使用其他优化技巧，如预分配数组使用的内存。这些想法以及它们对 MATLAB 代码执行速度的影响将在教程 3.3 中讨论。

3.3.7　输入和输出

基本的输入和输出功能可使用函数 input（要求用户输入并从键盘读取数据）和 disp（在屏幕上显示文字或数组）。MATLAB 还包括很多支持从文件读入和写出到文件的函数。

3.4　图形和可视化

MATLAB 具有丰富的图元以绘制 2-D 和 3-D 图形。教程 9.1 将结合绘制图像直方图和

1　变得习惯于用矢量化方式写代码而不是像在传统的程序语言中使用等价的嵌套来循环并不是一个简单的过程，需要时间来掌握它。

变换函数挖掘一些 2-D 绘图功能，而第 11 章的教程将结合在频率域设计图像处理滤波器给出一些 3-D 绘图示例。

MATLAB 也包括许多显示图像（和检查其内容）的内置函数，其中一些将在本书中多次使用（在教程 4.2 中讨论更多的细节）。

3.5 教程 3.1：MATLAB——导览

目的

这个教程的目的是给出对 MATLAB 环境的一个简短的概述。

目标

- 熟悉 MATLAB 的工作环境。
- 学习如何使用工作目录和设置通路。
- 熟悉 MATLAB 的帮助系统。
- 探究揭示系统信息和其他有用内置功能的函数。

规程

MATLAB 的环境有一个简单的布局，包括若干个关键的区域（图 3.1）[1]。下面对每一个进行描述：

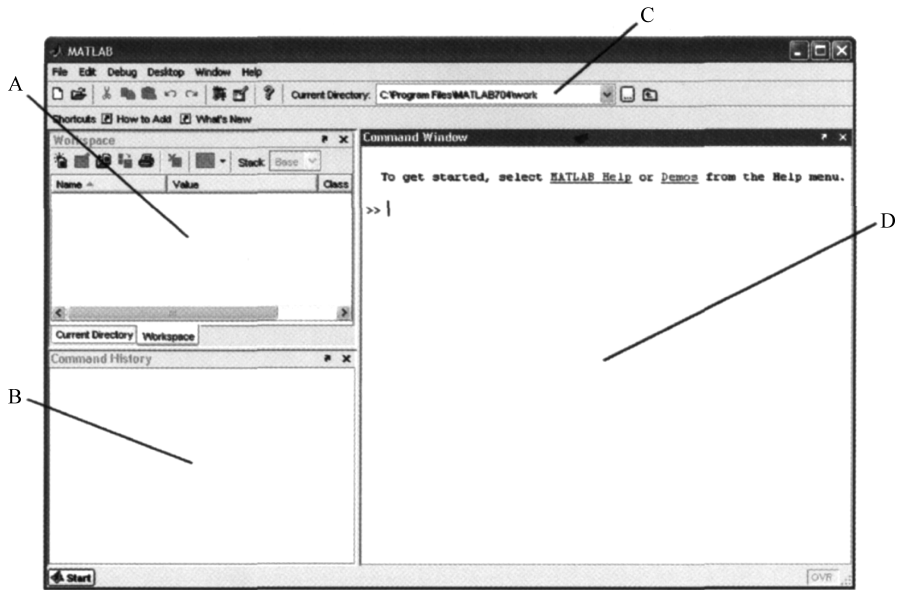

图 3.1 MATLAB 环境

- A：这个窗由两个标签区域构成，一个显示当前工作目录中的所有文件，另一个显示工作空间。工作空间列出所有当前使用的变量。

1 你的 MATLAB 窗显示的实际方式可能与图 3.1 不同，这依赖于操作系统、MATLAB 版本，以及在桌面菜单里需要显示的所选窗和工作区域。

- B：这个窗给出命令历史。
- C：在这个窗中可以变动当前的工作目录。为改变**当前目录**，可以或者在文字框中直接键入或者单击目录的按钮。还可以使用 path 命令改变工作目录。更多信息见帮助文档。
- D：这是命令窗。在这里通过键入命令来控制 MATLAB。

为使用 M-文件和图像文件，MATLAB 需要知道这些文件在什么位置。这可用两种方式实现：将当前目录设置在一个特定的位置，或者将位置加到 MATLAB 知道的**通路**列表上。当前目录只用于临时位置或当你仅需要访问一个目录一次时。该目录在每次 MATLAB 重新启动时被重新设置。为改变当前目录，参考上面对 C 的描述。设置一条通路是一个告诉 MATLAB 文件在什么位置的永久方式，这个位置将保留在 MATLAB 关闭时的设置中。下面的步骤将演示如何在 MATLAB 中设置一个通路：

（1）从 **File** 菜单，选择 **Set Path**…

通路将在列表框中按优先级递减的顺序排列。

（2）如果希望加入的目录中还有子目录，那么使用 **Add with Subfolders** 按钮。如果目录中只有文件，可以使用 **Add Folder** 按钮。为改变目录的优先级，使用 **Move** 按钮

（3）**Save** 改变，并关闭 **Set Path** 窗。

MATLAB 中的帮助系统很有用。它提供关于函数的信息，如句法、描述、需要的和可选的输入参数以及输出参数。它还包括许多演示。下面的步骤将展示如何访问帮助文档并在帮助窗中漫游。

（4）为访问帮助系统，在目录窗中键入 doc。现在打开帮助系统。

帮助窗的左标签区域显示一棵帮助信息树。帮助窗的右标签区域显示所选择的帮助文档。如果准确知道哪个函数需要帮助，可以使用 doc，helpwin 或 help 命令以直接进入帮助信息。

（5）在 MATLAB 的命令窗中，执行下面的命令以观看函数 computer 的文档。

```
help computer
helpwin computer
doc computer
```

问题 1：命令 help，helpwin 和 doc 有什么区别？

有若干命令可以用来获得有关系统的信息。在这个步骤发掘它们。

（6）执行下面的命令，一次一个，看看它们的功能。

```
realmin
realmax
bitmax
computer
ver
version
hostid
license
```

问题 2：命令 ver 和 version 有什么区别？

（7）对一些 MATLAB 函数，重复键入命令 why。

3.6 教程 3.2：MATLAB 数据结构

目的

这个教程的目的是学习如何生成、初始化和访问一些 MATLAB 中最有用的**数据结构**。

目标

- 学习如何使用 MATLAB 来进行包含变量、数组、矩阵和矢量的基本计算。
- 探究多维数组和元胞数组。
- 复习矩阵操作。
- 学习如何使用 MATLAB 结构。
- 发掘可与 MATLAB 数据结构共同使用的函数。

规程

（1）在命令窗中执行下列代码行，一次一行，看如何将 MATLAB 用作一个计算器。

```
2 + 3
2*3 + 4*5 +6*7
```

问题 1：变量 ans 用来做什么？

问题 2：在语句末尾使用分号的用意是什么？

（2）使用变量执行计算。

```
Fruit_per_box = 20; num_of_boxes = 5;
Total_num_of_fruit = fruit_per_box * num_of_boxes
```

问题 3：尝试生成你自己的变量。变量要区分大小写吗？

问题 4：变量 pi，eps，inf，i 的值或用途是什么？有可能重写它们其中之一吗？如果能，如何能撤销它？

（3）执行命令 who 和 whos，一次一个，以观看它们的功能以及它们之间的差别。

有若干个命令可以保持 MATLAB 环境清洁。当你感觉命令窗或工作空间由于语句和变量而杂乱时使用它们。

（4）在工作空间中清除一个变量。在命令执行后，注意变量是如何从工作空间消失的。

```
clear fruit_per_box
```

（5）使用下列代码行清除命令窗和所有变量（一次一个以单独观察它们的效果）。

```
clc
clear all
```

（6）执行下列代码行以生成一个 3 × 3 的矩阵。

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

问题 5：在这个语句中，分号有什么用？

冒号操作符

（7）MATLAB 中一个非常有用的操作符是**冒号操作符**（:）。它可用来生成一个数字矢量。

```
1:5
```

（8）第 3 个参数确定如何在起始和结束数字间进行计数。它被指定在起始值和结束值之间。

```
1:1:5
5:-1:1
1:2:9
9:-2:1
```

问题 6：写出一个语句，它能生成一个值从 0 到 π，每次增加 π/4 的矢量。

（9）冒号操作符也可用来返回一个矩阵的一整行或一整列。

```
A = [1 2 3; 4 5 6; 7 8 9]
A(:,1)
A(1,:)
```

问题 7：写出一行代码，它能生成与上面变量 A 一样的 3 × 3 的矩阵，但使用冒号操作符来生成每一行的数字序列而不显式地写它们。

（10）冒号操作符也可用函数 colon 代替，该函数执行相同的操作。

```
colon(1,5)
```

从上面的步骤可见，要生成一个具有均匀分布数字的矢量，如果知道矢量从哪里开始和到哪里结束，以及各个值之间的距离有多大，那么很容易用冒号（:）操作符来实现。在有些情况下，可能会希望生成一个范围在两个数字间的数字矢量，但仅知道需要的值的数量（例如，生成一个包含 4 个 π/4 和 π 之间值的矢量）。为此，使用 linspace 函数。

（11）执行下面命令，看函数 linspace 如何工作。

```
linspace(pi/4,pi,4)
```

（12）比较上述步骤的结果与下面的值。

```
pi/4
pi/2
3*pi/4
pi
```

特殊的内置矩阵

MATLAB 有若干个内置函数，它们可自动生成经常用到的矩阵。

（13）执行下面的代码行，每次一行。

```
zeros(3,4)
ones (3,4)
ones (3,4) * 10
read (3,4)
readn (3,4)
```

问题 8：函数 rand(M,N)和 randn(M,N)之间有什么差别？

矩阵级联

矩阵级联可用方括号（[]）或用函数 cat 实现。例如，考虑语句

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

方括号是 3 行的组合。它们可以分别定义为矢量并用方括号结合进一个矩阵而不是显式地每次定义一行。

（14）将 3 个单独的矢量结合进一个 3 × 3 的矩阵。

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

类似地，方括号也可用来删除一个矩阵的一行。

（15）删除矩阵 A 的最后一行（第 3 行）。注意如何使用冒号操作符以指定整个行。

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

一个有 *N* 个元素的矢量是一个有一行和 *N* 列的数组。一个矢量的元素可以通过元素的数目来访问，如在 X(5)中，所访问的是矢量 X 的第 5 个元素。一个 2-D 矩阵的元素可通过先指定它的行然后它的列来访问，如在 X(2,5)中，所返回的是第 2 行第 5 列的元素。维数高于 2 的矩阵可用类似的方式访问。还需要注意的是在 MATLAB 中数组是从 1 开始计数的——一个数组的第 1 个元素赋为 1 或使用 1 来访问，这与许多程序语言从 0 开始计数不同。

（16）使用函数 ones 和 rand 来生成多维数组。

```
A = ones(4,3,2);
B = rand(5,2,3);
size(A)
size(B)
disp(A)
disp(B)
```

问题 9：函数 size 做了什么？

问题 10：函数 disp 做了什么？

包含矩阵的操作

对矩阵执行算术操作可通过操作符+，–，*，/来实现。对相乘（*）和相除（/）操作

符，缺省的是矩阵乘法和矩阵除法。要对一个矩阵的单个元素执行操作，要在操作符前加一个点（.）。

（17）使用函数 ones 和 rand 来生成多维数组。

```
X = [1 2 -2; 0 -3 4; 7 3 0]
Y = [1 0 -1; 2 3 -5; 1 3 5]
X * Y
```

（18）执行逐元素的乘法。

```
X .* Y
```

（19）执行另一个对两个矩阵的矩阵乘法。

```
X = eye(3,4)
Y = rand(4,2)
X * Y
Y * X
```

问题 11：为什么最后一个操作不成功？

（20）使用函数和执行对一个矩阵对角线的操作。

```
Y = rand(3,3)*4
Y_diag = diag(Y)
Y_trace = trace(Y)
```

问题 12：函数 diag 做了什么？

问题 13：函数 trace 做了什么？

问题 14：另写一个语句，它能产生与函数 trace 相同的结果？

（21）计算一个矩阵的转置。

```
Y
Y-t = Y'
```

（22）计算一个矩阵的逆并证明 $\boldsymbol{Y}\boldsymbol{Y}^{-1} = \boldsymbol{Y}^{-1}\boldsymbol{Y} = \boldsymbol{I}$，其中 $\boldsymbol{I}$ 是单位矩阵。

```
Y_inv = inv(Y)
Y * Y_inv
Y_inv * Y
```

（23）计算一个矩阵的行列式。

```
Y_det = det(Y)
```

元胞数组

如前所述，矩阵是 MATLAB 中的基本数据类型。它与一个数组是一种同质数据结构的传统定义很相似，即它的所有元素都是相同的类型。另一方面，**元胞数组**是另一种数组，其中的每个元胞可以是 MATLAB 允许的任何数据类型。每个元胞都与其他元胞无关，所

以，它可包含 MATLAB 支持的任何数据类型。当使用元胞数组时，在对一个元胞访问或赋值时需要很小心，需要使用花括号（{}）而不是括号。

（24）执行下面的代码行（一次一行），观察 MATLAB 如何处置元胞数组。

```
X{1} = [1 2 3; 4 5 6; 7 8 9];          %元胞 1 是一个矩阵
X{2} = 2+3i;                           %元胞 2 是一个复数
X{3} = 'String';                       %元胞 3 是一个字符串
X{4} = 1:2:9;                          %元胞 4 是一个矢量
X
celldisp(X)
X(1)
X{1}
```

问题 15：函数 celldisp 做了什么？

问题 16：百分数（%）字符做了什么？

问题 17：上面最后两行代码间有什么差别（X(1)相对于 X{1}）？

还有另一种将数值赋予元胞数组的方式，虽然在语法上不同，但也能给出相同结果。注意下一个步骤中，元胞指针被普通的括号（()）括住，但被元胞保存下来的数据用花括号（{}）所封装。

（25）执行下面的代码行，观察另一种给元胞数组赋值的方式。

```
X(1) = {[1 2 3; 4 5 6; 7 8 9]};
```

（26）下面几行代码将展示当面对字符串时正确和不正确的给元胞数组赋值的方式。

```
X(3) = 'This produces an error'
X(3) = {'This is okay'}
X{3} = 'This is okay too'
```

结构

结构是另一种在 MATLAB 中存储数据的方式。结构的句法与其他程序语言中的类似。可使用点（.）操作符来指示一个结构中的不同域。具有相同布局（域的数目、它们的名称、尺寸和含义）的结构可以结合进一个（结构的）数组中。

（27）生成一个表达两幅图像和它们尺寸的有两个结构的数组。

```
my_images(1).imagename = 'Image 1';
my_images(1).width = 256;
my_images(1).height = 256;
my_images(2).imagename = 'Image 2';
my_images(2).width = 128;
my_images(2).height = 128;
```

（28）观察结构的细节并显示一个域的内容。

```
my_images(1)
my_images(2).imagename
```

（29）显示有关结构的信息。

```
num_of_images = prod(size(my_images))
fieldnames(my_images)
class(my_images)
isstruct(my_images)
isstruct(num_of_images)
```

问题 18：函数 fieldnames 做了什么？

问题 19：当从函数 isstruct 得到的结果为 1 时是什么含义？当结果为 0 时是什么含义？

问题 20：使用帮助系统确定哪个函数可用来从一个结构中删除一个域。

3.7　教程 3.3：MATLAB 编程

目的

这个教程的目的是使用脚本和函数来探究编程概念。

目标

- 学习如何编写和执行脚本。
- 探究 MATLAB 编辑器。
- 探究函数以及如何编写函数。
- 学习循环矢量化的基础。

将需要

- script3_3_1.m
- script3_3_2.m
- raise_to_power.m

规程

尽管命令窗很简单，使用也方便，但它并没有提供保存和编辑代码的方法。不过，存储在命令历史窗中的命令可以很方便地写进一个脚本文件。一个脚本是一个具有扩展.m 的文字文件，可用来存储 MATLAB 代码。脚本可用内置编辑器生成和修改。

（1）要开始一个新脚本，导航到 **File → New → M-File**。

MATLAB 编辑器可以依赖于环境如何事先设置而打开一个分离的窗或停驻在 MATLAB 环境中。

（2）打开名为 script3_3_1.m 的文件。如果 M-文件在当前目录中，就可以通过双击文件名而从当前目录列表打开它。

M-文件根据句法彩色编码以帮助读取。你可能已注意到，命令可使用两种方法加到任何脚本里：百分数（%）符号或用%{和%}括住。第 2 种方法用于脚本中的头信息。

问题 1：基于文件 script3_3_1.m 中的脚本，使用百分号（%）与使用%{和%}有什么

不同?

为执行脚本中的一个或多个代码行，高亮显示代码并按 F9 键[1]。

（3）高亮显示 script3_3_1.m 中的所有代码并按 F9 键以执行脚本。

元胞模式

元胞模式是一种新的编辑模式（MATLAB 7.0 及更新版本），它允许对变量进行小的调整，并能方便地重新执行一段代码。

（4）打开文件 script3_3_2.m。

（5）在 MATLAB 环境中通过导航到 **Cell → Enable Cell Mode** 来启用元胞模式。

启用元胞模式后，可注意到前面有两个百分号（%%）的注解比其他注解稍微粗一些。这对应块标题。为生成一个元胞块，需要做的只是给它一个块标题。元胞块允许你一次修改和执行一块代码。在单个脚本中可以存在多于一个元胞块。

（6）执行脚本中的所有代码。

这个脚本是一个示例，要将一幅单色图像的直方图通过函数 imadjust 拉伸到整个允许值的动态范围。该函数也允许对图像进行伽玛校正。此时，直方图拉伸和伽玛校正的概念并不重要；它们将在第 10 章详细讨论。要关注的是元胞模式可能在后面有用。通过对伽玛值进行小的调整，可以看到其对调整图像的影响。元胞模式使这个工作很容易。

（7）确定变量伽玛被赋予值 1 的代码行。仅高亮显示值 1。

当元胞模式工作时，一个新的菜单条出现在编辑器中。在所有图标中，有两个文字框允许对刚在代码中选择的值进行调整。第 1 个文字框允许加和减，另一个允许乘和除在代码中高亮的值。

（8）在加/减文字框中键入值 0.1 并按右边的加号（+）符号钮。

问题 2：代码中变量 gamma 的值发生了什么变化？

问题 3：除了代码的改变，当按加号钮时还发生了什么变化？如果再次按会发生什么变化？

问题 4：除了在本教程中描述的以外，当启用元胞模式时还有哪些特征可用？

函数

函数也是包含 MATLAB 代码的.m 文件，但还增加了可从另一个脚本或函数调用以及可以接受参数的能力。打开函数 raise_to_power 看看文件是如何构建的。

（9）在编辑器中打开 raise_to_power。

一个函数的 M-文件包括调用函数，执行它，以及显示它的帮助文档中所有需要的信息。文件中的第 1 行定义函数以及它的输入和输出参数。在函数 raise_to_power 的情况里，它获取两个参数并返回一个。紧接在函数定义后的注解都看作帮助文档。

（10）确认 raise_to_power.m 文件在这样一个目录中，该目录或者是所设置的通路的一部分或者是当前目录。

（11）在命令窗中键入下面的语句以观看函数的帮助信息：

1 快捷键有可能随 MATLAB 的版本和操作系统的不同而不同。列在本书中的快捷键用于 MATLAB 的 PC 版本。

```
help raise_to_power
```

问题 5：比较帮助文档与 M-文件中的注解。MATLAB 是如何确定哪个注解显示为帮助哪个注解只是函数代码的注解？

因为要使用 MATLAB 进行图像操作，所以需要确认利用 CPU 和高效内存编码的优点——特别是循环矢量化。举个例子，下面的伪码块可以很容易地在任何程序语言中实现。NB：依赖于计算机的速度，可以用下面的代码片段改变 MAX_CNT 的值。

```
MAX_CNT = 10000
for i = 1 to MAX_CNT
    do x(i) = i^2
```

这里填充了一个数组，它的每个元素都是那个元素平方的指针。下面的 MATLAB 代码用典型的程序语言实现上面的伪码。在这些例子中，命令 tic 和 toc 用来计算执行的时间。

（12）用下面的语句实现上面的伪码。

```
tic
MAX_CNT = 10000
for i = 1:MAX_CNT
    x(i) = i^2;
end
toc
```

通过简单地预分配数组使用的内存有可能极大地影响这个循环的速度。这是很有效的，因为每次加数据到矩阵中时，如果没有空间了，就需要分配新的内存以保持较大的变量。如果已分配了正确数量的内容，则 MATLAB 只需要改变每个元胞中的数据。

（13）利用预分配，再次实现上述的循环。

```
tic
MAX_CNT = 10000
X = zeros(1,MAX_CNT);
for i = 1:MAX_CNT
    x(i) = i^2;
end
toc
```

问题 6：通过预分配数组 x 使性能提高了多少倍？

在 MATLAB 中，这仍被认为是差的编程技术。MATLAB 在你写的代码和计算机硬件间起翻译的功能。这表示每次遇到一个语句，它将其翻译成能被硬件理解的底层语言。因此，上面的循环要使 MATLAB 在循环中每次都翻译语句——在上例中是 10000 次。相对于计算机硬件，翻译的速度很慢。因为 MATLAB 本质上对矩阵进行操作，所以可使用循环矢量化避免循环来执行相同的操作。

（14）实现对循环更有效的版本。

```
tic
```

```
MAX_CNT = 10000
i = 1:MAX_CNT;
x = i.^2;
end
toc
```

问题 7：在上面的代码中，为什么使用.^代替^？

问题 8：相比前两个方法循环矢量化比它们快了多少？

这里并不需要告诉 MATLAB 对每个元素进行操作，因为那是 MATLAB 环境的本质。循环矢量化还考虑了在第 1 个方法中的预分配问题。

问题 9：考虑下列伪代码，写出它们的矢量化等价码。

```
i = 0
for t = 0 to 2*pi in steps of pi/4
    do i = i + 1
        x(i) = sin(t)
```

前面已说过，一个脚本可从命令历史窗获得。现在已加入了一些命令，可看看如何来实现这点。

（15）要从命令历史窗生成一个脚本，确定所加入的最后 4 个语句。为选择所有这 4 个语句，按住 **ctrl** 键并选择各个语句，然后右击高亮的语句，并选择 **Create M-File**。

学到了什么？

- MATLAB（MATrix LABoratory）是一个数据分析、原型设计以及可视化的工具，它对矩阵和矩阵操作有内嵌的支持，还有优秀的图形能力，以及高层的程序语言和开发环境。它被选中作为本书的工具是因为它使用方便和具有对图像处理操作（封装在图像处理工具箱中）的大量内嵌支持。
- MATLAB 工作环境包括 MATLAB 桌面，MATLAB 编辑器和帮助系统。
- MATLAB 中的 M-文件可以是脚本，它简单执行一系列 MATLAB 命令（或语句）；也可以是函数，它接受变量（参数）并生成一个或多个输出值。用户生成的函数扩展了 MATLAB 和其工具箱满足特定需求或应用的能力。
- 一个包含由一系列需要翻译和执行的命令所组成脚本的 M-文件很像一个批处理文件。另一方面，一个 M-文件也可包含一个函数，由一段执行特定任务的代码组成，具有意义的名称（其他函数可用来调用它），并（可选地）接受参数和返回结果。
- 开始使用 MATLAB 的最好方法是发掘它优良的产品文档和在 MathWork 网站的在线资源。

了解更多

有许多关于 MATLAB 的书籍，例如：

- Hanselman, D. and Littlefield, B., Mastering MATLAB 7, Upper Saddle River, NJ: Prentice Hall, 2005. 优秀的参考书。
- Pratap, R., Getting Started with MATLAB, New York, NY: Oxford University Press, 2002. 初学者非常值得一读。

在网上

有许多在线的免费 MATLAB 教程。与本书配套网站（http://www.ogemarques.com/）上包括与它们中一些的链接。

3.8 练　习　题

3.1 使用 MATLAB 作为计算器，执行下面的计算：

（a）24.4×365

（b）$\cos(\pi/4)$

（c）$\sqrt[3]{45}$

（d）$e^{-0.6}$

（e）4.6^5

（f）$y = \sum_{i=1}^{6} (i^2 - 3)$

3.2 使用函数 `format` 回答下列问题：

（a）MATLAB 用于浮点数计算的精确度是多少？

（b）在命令窗中显示浮点数计算结果的默认的小数位数是多少？

（c）如何能改变成不同的小数位数？

3.3 初始化下列变量：$x = 345.88$；$y = \lg(45.8)$；$z = \sin(3\pi/4)$。注意 MATLAB 函数 `log` 是自然对数；所以需要使用 `log10` 来计算一个数的常用（基为 10）对数。

使用它们计算下列表达式：

（a）$e^{(z^3 - y^{1.2})}$

（b）$x^2 - \sqrt{(4y + z)}$

（c）$\dfrac{x - y}{z + 5y}$

3.4 初始化下列矩阵：

$$\boldsymbol{X} = \begin{bmatrix} 4 & 5 & 1 \\ 0 & 2 & 4 \\ 3 & 4 & 1 \end{bmatrix}$$

$$\boldsymbol{Y} = \begin{bmatrix} -7 & 6 & -1 \\ 3 & -2 & 0 \\ 13 & -4 & 1 \end{bmatrix}$$

$$\boldsymbol{X} = \begin{bmatrix} 0 & 7 & 1 \\ 0 & 2 & -5 \\ 3 & 3 & 1 \end{bmatrix}$$

使用它们计算下列表达式：

（a）$3\boldsymbol{X} + 2\boldsymbol{Y} - \boldsymbol{Z}$

（b）$\boldsymbol{X}^2 - \boldsymbol{Y}^3$

（c）$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}^{\mathrm{T}}$

（d）$\boldsymbol{X}\boldsymbol{Y}^{-1}$

3.5 下列 MATLAB 函数有什么不同：

（a）`log` 和 `log10`

（b）`rand` 和 `randn`

（c）`power` 和 `mpower`

（d）`uminus` 和 `minus`

3.6 函数 `lookfor` 的用途是什么？举一个使用它的例子。

3.7 函数 `which` 的用途是什么？举一个使用它的例子。