

瀚哥的 MATLAB 手册

(中英注释版)

数模小朋友的好帮手，英语爱好者的灵魂鸡汤，程序员眼中
欲罢不能的手册！瀚哥就是要数猫倾情投入无数日夜的念道
德经，造就的 MATLAB 手册！

联系骚年：QQ：569929309

微博：瀚哥就是要数猫(现在更名“一只尼玛 mark”)

2014-2016 年

前言

MATLAB 语言是一种工程语言，语法很像 VB 和 C，比 R 语言容易学，你知道 R 语言的语法有多糟糕么。同样，相对于 Python，MATLAB 的优点是天生为了算而算，数与数之间的运算就是矩阵与矩阵之间的运算，在运算方面可能容易学一点。但是，MATLAB 是收费的，更多信息请参考其官方网站。

本人是一名数学转计算机专业的学生，参加过两次美国大学生数学建模，全部获得没什么水准的二等奖，一次全国大学生数学建模一等奖。我深知队友的重要性，而我担任编程和一些建模工作，有一次比赛大部分写作是我负责的，其实美赛中英语这个梗不重要。

编程在数模竞赛的重要性有多大？首先在国赛，直接看你的代码量来决定你的排位。美赛中虽不要源码，可是一个模型建立后，通过人工根本无法进行运算，当然可以用 Excel，我也就不说什么了。编码运算模型，可视化结果，这就是一个数模团队多么需要程序员的原因。

编程很容易，可是资源很少，或者是翻译得有问题，相对于 MATLAB，很少有人写这方面的书，大部分是直接写一堆算法，MATLAB 部分就作为第一二章简单带过，然后取了个名字，书名带着 MATLAB，我也就不说什么了。其实大家不知道，每一门语言都有强大的官方文档，而且是随着安装环境一起被下载下来的，只不过是英文，那是最全最详尽的文档，大部分翻译都是从上面摘下来的。

我翻译了 MATLAB 快速入门那一部分，中英结合，为什么我不直接砍掉英文，因为英文才是作者最原始的意思，无法取代，比如张三对李四说的话，到李四到老王说的话，意思会差很远。我在大学过了英语四六级，而且是裸考的，语言其实只是一门工具。下面为我翻译的手册，大家随手看看吧。有问题的话请查有道字典什么的，然后作为一个玩过很多编程语言的人，我是理解了意思再翻译的。

（数组是一种数据类型，矩阵是线性代数术语，矢量是解析几何术语，在 MATLAB 语言中数组是存放东西的容器，所有的变量都是数组，存放的东西根据现实世界实际情况可以是矩阵或矢量，数组=矩阵=矢量，这样理解起来就不难了。向量是物理术语，矢量是数学术语。）

有问题联系 QQ: 569929309, 微博: 一只尼玛_mark
时过境迁，微博名已经改了，不要问我为什么。

时间：2015/12/30
广财宿舍 生病中

今天终于把这个东西翻完了，总共就翻译了四五次，每次几个小时。这个英文单词统计了一下就 6600 多个词，相比在美赛 38 页全是英文的，明显有点弱。英文读起来还是很好的，MATLAB 帮助文档简直是友好，友善，易懂，还有视频练听力，大家看了这个学点皮毛，然后写一些可以应用的程序吧。

时间：2016/1/1
广财宿舍 翻译完

Contents

目录

Part 1:Quick Start.....	1
I. Product Description.....	1
产品描述.....	1
I. 1. The Language of Technical Computing.....	1
工程计算语言.....	1
I. 2. Desktop.....	1
桌面.....	1
II. Matrices and Arrays.....	4
矩阵和数组.....	4
II. 1. Array Creation.....	4
数组创建.....	4
II. 2. Matrix and Array Operations.....	5
矩阵和数组运算.....	5
II. 3. Concatenation.....	7
串联.....	7
II. 4. Complex Numbers.....	8
复数还记得吗?	8
III. Array Indexing.....	9
数组索引.....	9
IV. Workspace.....	11
工作站.....	11
V. Character Strings.....	13
字符串.....	13
VI. Functions.....	14
函数.....	14
VII. Plots.....	15
绘图.....	15
VII. 1. Line Plots.....	15
线性绘图（二维）.....	15
VII. 2. 3-D Plots.....	17
三维绘图.....	17
VII. 3. Subplots.....	18
子图.....	18
VIII. Scripts.....	19
脚本.....	19
VIII. 1. Sample Script.....	19
简单脚本.....	19
VIII. 2. Loops and Conditional Statements.....	21
循环和条件语句.....	21

VIII. 3. Script Locations.....	23
脚本位置.....	23
IX. Help.....	24
帮助.....	24
Part 2:Programming.....	25
X. Programming: Control Flow.....	25
编程指南：控制流.....	25
X. 1. Conditional Control — if, else, switch.....	25
条件控制— if, else, switch.....	25
X. 2. Array Comparisons in Conditional Statements.....	27
条件语句中的数组比较.....	27
X. 3. Loop Control — for, while, continue, break.....	28
循环控制— for, while, continue, break.....	28
X. 4. Program Termination — return.....	31
程序终结者— return.....	31
X. 5. Vectorization.....	31
矢量化.....	31
X. 6. Preallocation.....	32
预先配置.....	32
XI. Programming: Scripts and Functions.....	33
编程指南：脚本和函数.....	33
XI. 1. Overview.....	33
概况.....	33
XI. 2. Scripts.....	34
脚本.....	34
XI. 3. Functions.....	35
函数.....	35
XI. 4. Types of Functions.....	37
函数类型.....	37
XI. 5. Global Variables.....	41
全局变量.....	41
XI. 6. Command vs. Function Syntax.....	41
命令模式 VS 函数格式.....	41

Part 1:Quick Start

I. Product Description

产品描述

I. 1. The Language of Technical Computing

工程计算语言

MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

MATLAB 是一个高性能的语言和交互式环境，能让你更快地演算集中式的计算任务，而不是依靠传统的 C,C++, Fortran 等语言。好牛！

Key Features

主要特征

1.High-level language for technical computing

工程计算的高性能语言

2.Development environment for managing code, files, and data

管理代码、文件和数据的开发环境

3.Interactive tools for iterative exploration, design, and problem solving

交互式探索，设计，问题解决的交互式工具

4.Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration

数学函数，用于线性代数，统计，傅里叶分析，过滤，最优化，数值综合

5.2-D and 3-D graphics functions for visualizing data

二维，三维图形函数来可视化数据

6.Tools for building custom graphical user interfaces

构建一般图形用户界面的工具

7.Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft® Excel

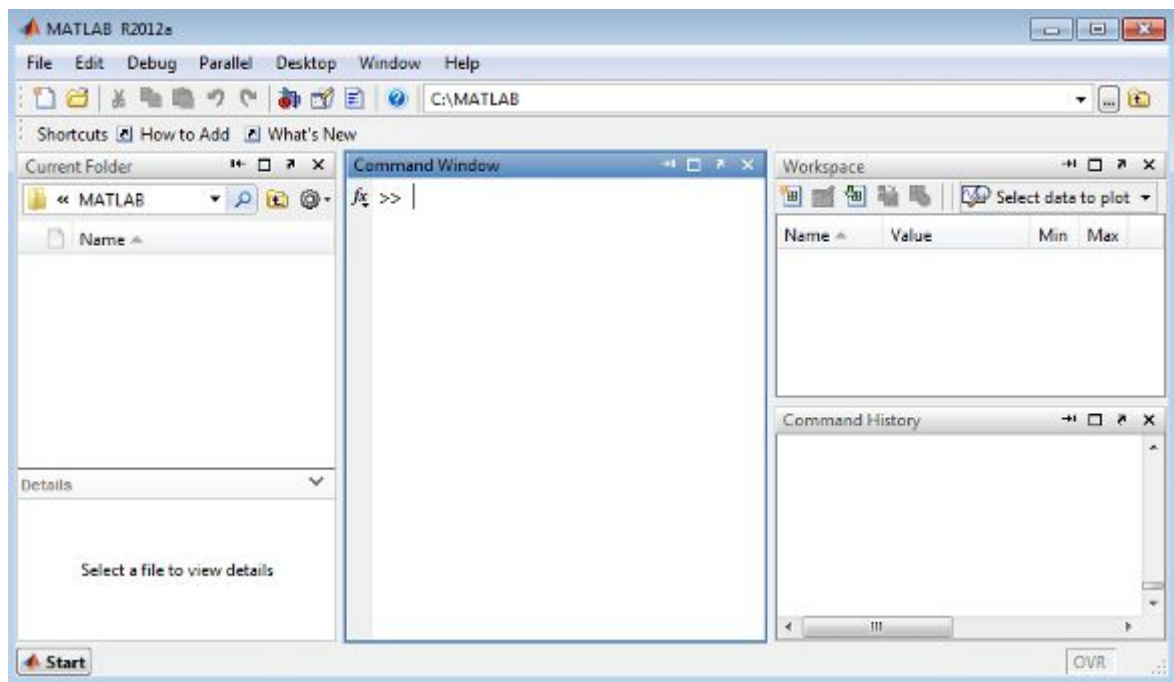
MATLAB 集成函数依靠的算法来自外部应用程序及语言，例如 C，C++,Fortran,JAVA,COM 和 excel

I. 2. Desktop

桌面

When you start MATLAB, the desktop appears in its default layout.

当你打开 MATLAB，桌面将显示其默认布局。多像 VC++，比它强多了！



The desktop includes these panels:

桌面包括这些部分：

1.Current Folder — Access your files.

当前目录——访问你的文件

2.Command Window — Enter commands at the command line, indicated by the prompt (>>).

命令行窗口——在命令行输入命令，指示通过 DOS 系统提示符的风格 (>>)

3.Workspace — Explore data that you create or import from files.

工作站——查看你创建的或从文件导入的数据

4.Command History — View or rerun commands that you entered at the command line.

命令历史——查看或重新运行你在命令行输入的命令，多人人性化。

As you work in MATLAB, you issue commands that create variables and call functions. For example, create a variable named `a` by typing this statement at the command line:

当你在 MATLAB 下工作的时候，你发布的命令将会创建变量和调用函数。例如，创建一个变量，名字是 `a`，通过在命令行打这个东东：

```
>>a = 1
```

MATLAB adds variable `a` to the workspace and displays the result in the Command Window
它就会在工作站加上一个变量，然后在命令窗口显示结果。类型都不用定义，多省心！

```
a =
```

```
1
```

Create a few more variables.

创建更多的变量

```
>>b = 2
```

```
b =
```

```
2
```

```
>>c = a + b
```

```
c =
```

```
3
```

```
>>d = cos(a)
```

```
d =
```

```
0.5403
```

三角函数都可以求！

When you do not specify an output variable, MATLAB uses the variable `ans`, short for answer, to store the results of your calculation.

当你没有定义一个输出的变量，它就会用这个东东 `ans` 来代替，就是 short for answer 答案缩写，存储你的计算结果。多有人文关怀，赞一个！

```
>>sin(a)
```

```
ans =
```

```
0.8415
```

If you end a statement with a semicolon, MATLAB performs the computation, but suppresses the display of output in the Command Window.

如果你以分号的状态结尾，MATLAB 会开展计算，但会镇压命令窗口输出的显示，镇压就是不让结果出来！多好的词！

```
>>c = a*b;
```

You can recall previous commands by pressing the up- and down-arrow keys, \uparrow and \downarrow . Press the arrow keys either at an empty command line or after you type the first few characters of a command. For example, to recall the command `b = 2`, type `b`, and then press the up-arrow key.

你能重新呼唤先前的命令，只要你按 键盘上↑ 和 ↓。按这个箭头键不管你的命令是空的还是只打入了命令开头的几个字符，没问题，为了呼唤命令 `b=2`，打 `b`，然后按上下箭头键！

II. Matrices and Arrays

矩阵和数组

不要跟我说不知道矩阵，去看线代！，

MATLAB is an abbreviation for "matrix laboratory." While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and arrays.

重点来了，MATLAB 就是 "matrix laboratory."（“矩阵实验室”）的简称！当其他程序语言通常还在一次一个数字一个数字地运行，MATLAB 就被设计来运算整个矩阵和数组，主要功能就是这样，所以才叫矩阵实验室！

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

所有的 MATLAB 变量都是多维数组，不管数据是什么类型。一个矩阵就是一个二维数组！它总是在线性代数里面使用。多温馨的说明书，还不快去复习！

II. 1. Array Creation

数组创建

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

为了创建一个只有一行的四元素数组，你只要用逗号或者空格分开这些元素即可。

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

returns

返回

```
a =
```

```
1      2      3      4
```

This type of array is a row vector.

这种类型的数组叫行向量。连线代都教会你，多强大的手册！

To create a matrix that has multiple rows, separate the rows with semicolons.

为了创建多行矩阵怎么办？那就用分号分开那些行。

```
a = [1 2 3; 4 5 6; 7 8 10]
```



```
a =
```

```

1     2     3
4     5     6
7     8    10
```

Another way to create a matrix is to use a function, such as `ones`, `zeros`, or `rand`. For example, create a 5-by-1 column vector of zeros.

其他创建矩阵的方式就是利用函数。例如 `ones()`, `zero()`, `rand()`。举个例子，创建一个五行一列的列向量，元素都是零！

```
>>z = zeros(5,1)
```

```
z =
```

```

0
0
0
0
0
```

II. 2. Matrix and Array Operations

矩阵和数组运算

MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

MATLAB 允许你在一个矩阵中处理所有的值只要使用一个算数操作符或者函数。

```
>>a + 10
```

```
ans =
```

```

11     12     13
14     15     16
17     18     20
```

```
>>sin(a)
```

```
ans =
```

```

0.8415    0.9093    0.1411
-0.7568   -0.9589   -0.2794
0.6570    0.9894   -0.5440
```

矩阵运算不会么，去看线性代数，骚年！

To transpose a matrix, use a single quote ('):

转置一个矩阵，即行列呼换，使用一个引用号！

```
>>a'
```

```
ans =
```

```
1     4     7
2     5     8
3     6    10
```

You can perform standard matrix multiplication, which computes the inner products between rows and columns, using the `*` operator. For example, confirm that a matrix times its inverse returns the identity matrix:

你能使用标准的矩阵乘法，可以在行与列之中计算内积，又称点积，利用`*`操作符。例如，证明一个矩阵乘以它的逆，会返回特征矩阵即单位矩阵： $A * A^{-1} = E$

```
>>p = a*inv(a)
```

```
p =
```

```
1.0000         0 -0.0000
         0 1.0000         0
         0         0 1.0000
```

Notice that `p` is not a matrix of integer values. MATLAB stores numbers as floating-point values, and arithmetic operations are sensitive to small differences between the actual value and its floating-point representation. You can display more decimal digits using the `format` command:

注意到下面的 `p` 不是矩阵的整数值，MATLAB 将数字存储为浮点值，算术操作对于实际的数值和浮点值表示这些微小的差异是敏感的。你能够显示更多的小数数字利用格式命令：

```
>>format long
```

```
>>p = a*inv(a)
```

```
p =
```

```
1.000000000000000         0 -0.000000000000000
         0 1.000000000000000         0
         0         0 0.999999999999998
```

Reset the display to the shorter format using

重置显示到短格式使用

```
>>format short
```

```
>>p=a*inv(a)
```

p =

```
1.0000    0 -0.0000
    0 1.0000    0
    0    0 0.1000
```

format affects only the display of numbers, not the way MATLAB computes or saves them.

格式化仅仅影响数字的显示，而不是 MATLAB 计算和保存的方式

To perform element-wise multiplication rather than matrix multiplication, use the .* operator:

为了显示元素的智能乘法而不是矩阵乘法，使用一点加乘号！！

```
>>p = a.*a
```

p =

```
1    4    9
16   25   36
49   64  100
```

The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise. For example, raise each element of a to the third power:

矩阵对于乘法，除法，乘方，每一个都有相应的数组操作会使用元素智能。例如，提高每一个元素到它的三次幂。

```
>>a.^3
```

ans =

```
1    8   27
64  125  216
343  512 1000
```

II. 3. Concatenation

串联

累了没，看看串联。没错，你不是在上物理，就是串联！

Concatenation is the process of joining arrays to make larger ones. In fact, you made your first array by

concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

串联是一个组装数组使其更大的过程。事实上，你弄的第一个数组是通过一个一个独立元素串联起来的。一对方括号就是串联符号。

```
>>A = [a,a]
```

```
A =
```

```

1     2     3     1     2     3
4     5     6     4     5     6
7     8    10     7     8    10
```

Concatenating arrays next to one another using commas is called horizontal concatenation. Each array must have the same number of rows. Similarly, when the arrays have the same number of columns, you can concatenate vertically using semicolons.

利用逗号串数组到下一个叫水平串联，每一个数组必须有相同数量的行。同样的，当数组有同样数量的列，你可以用分号将其垂直串联！想象得出吧，我还以为你看手册看到要吐了。

```
>>A = [a; a]
```

```
A =
```

```

1     2     3
4     5     6
7     8    10
1     2     3
4     5     6
7     8    10
```

II. 4. Complex Numbers

复数还记得吗？

Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of -1 .

复数有实部和虚部。虚部这个部分呢，恩，是个问题，它就是-1 开平方根！

```
>>sqrt(-1)
```

```
ans =
```

```
0 + 1.0000i
```

To represent the imaginary part of complex numbers, use either i or j .

为了展示一个复数的虚部，利用 i 或 j ！

```
>>c = [3+4i, 4+3j, -i, 10j]
```

```
c =
```

```
3.0000 + 4.0000i    4.0000 + 3.0000i    0 - 1.0000i    0 + 10.0000i
```

III. Array Indexing

数组索引

有没有越来越觉得在上程序设计？MATLAB 就是一门高水平语言嘛！

Every variable in MATLAB is an array that can hold many numbers. When you want to access selected elements of an array, use indexing.

每一个在 MATLAB 的变量都是一个可以装很多数的数组。可是你要数数怎么办，想使用那个被选择的元素，可它却在数组里。众里寻他千百度，用索引吧！

For example, consider the 4-by-4 magic square A:

例如，考虑一下一个 4 叉 4 的有魔力的方阵 A，这个魔力什么意思呢？

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

```
A =
```

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

哦，其实就是没有一个数字是相同的，你还发现了什么？

There are two ways to refer to a particular element in an array. The most common way is to specify row and column subscripts, such as

有两种方式引用数组中的特定元素。最常用的方法就是定义行列下标。如：

```
>>A(4,2)
```

```
ans =
```

```
14
```

Less common, but sometimes useful, is to use a single subscript that traverses down each column in order:

较少的常用，就是不常用。可是他说很好用哦，就是用一个下标，在每一列按顺序从上到下地移动。

```
>>A(8)
```

```
ans =
```

```
14
```

Using a single subscript to refer to a particular element in an array is called linear indexing.

使用一个下标去引用在数组中的特定元素叫做线性索引。

If you try to refer to elements outside an array on the right side of an assignment statement, MATLAB throws an error.

如果你尝试在一个可分配状态的右边去引用超出数组的元素，MATLAB 会告诉你：出错了！你肯定不信，你在试吗？右边是神马意思？

```
>>test = A(4,5)
```

Attempted to access A(4,5); index out of bounds because size(A)=[4,4].

尝试去访问 A (4,5)，索引超出界限。因为 A 只有 4*4 那么大！

However, on the left side of an assignment statement, you can specify elements outside the current dimensions. The size of the array increases to accommodate the newcomers.

可是，在可分配状态的左边。Just do it!你可以定义一个超出目前维数的元素。数组的大小将会增加，因为它要适应新来者，新来者，吼吼！！

```
>>A(4,5) = 17
```

```
A =
```

```
16     2     3    13     0
  5    11    10     8     0
  9     7     6    12     0
  4    14    15     1    17
```

To refer to multiple elements of an array, use the colon operator, which allows you to specify a range of the form start:end. For example, list the elements in the first three rows and the second column of A:

为引用一个数组的多个元素，MULTIPLE 有并联意思哦，使用 colon 符号，没错，就是结肠，NO？哥斯达黎加货币符号？巴拿马的科隆市？不是啦，是用冒号：能够使你定义一系列的组成，开始：结束。例如，列出矩阵 A 在前三行且在第二列的元素：

```
>>A(1:3,2)
```

```
ans =
```

```
2
11
7
```

从上到下数到 3 行，再第二列。数数方法就是不一样！

The colon alone, without start or end values, specifies all of the elements in that dimension. For example, select all the columns in the third row of A:

冒号什么都没有，孤零零？那就指定全部元素在那一维里，例如，选择 A 在第三行的所有元素

```
>>A(3,:)
```

```
ans =
```

```
9    7    6   12    0
```

有没有凌乱了？记住括号里第一个是行，第二个是列就行了

The colon operator also allows you to create an equally spaced vector of values using the more general form start:step:end.

冒号也允许你创建一个以相同距离排列的向量，使用更常规的形式 开始：步长：结束

```
>>B = 0:10:100
```

```
B =
```

```
0    10    20    30    40    50    60    70    80    90   100
```

If you omit the middle step, as in start:end, MATLAB uses the default step value of 1.

如果你省略或者说你忘记中间的步长，那么 MATLAB 就使用默认步长 1！

有没有觉得好像很弱智的感觉？

第一次翻译 2014/1/11/23:47 广财宿舍

IV. Workspace

工作站

The workspace contains variables that you create within or import into MATLAB from data files or other programs. For example, these statements create variables A and B in the workspace.

工作站会包含你从命令行创建的变量、以及从数据文件或程序里导入到 MATLAB 的变量。例如：这些语句在工作站里创建变量 A 和变量 B。

```
>> A = magic(4);
```

```
>> B = rand(3,5,2);
```

You can view the contents of the workspace using whos.

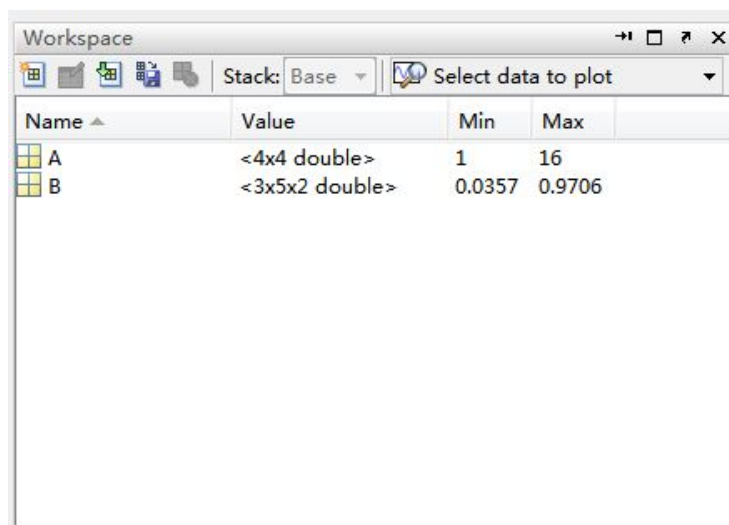
你能够查看工作站里的内容使用命令 whos

```
>> whos
```

Name	Size	Bytes	Class	Attributes
A	4x4	128	double	
B	3x5x2	240	double	

The variables also appear in the Workspace pane on the desktop.

这些变量也会出现在桌面上的工作站面板



Workspace variables do not persist after you exit MATLAB. Save your data for later use with the save command,

如果你关闭 MATLAB，工作站的变量不会保存。保存你的数据以便再用请使用保存命令。

```
>> save myfile.mat
```

Saving preserves the workspace in your current working folder in a compressed file with a .mat extension, called a MAT-file.

数据将保存在你的当前工作目录，一个有着.mat 扩展名的压缩文件，称为 MAT-文件。

To clear all the variables from the workspace, use the clear command.

清除工作站的所有变量，使用清除命令。

```
>> clear
```

Restore data from a MAT-file into the workspace using load.

使用加载命令从 MAT-文件中加载数据到工作站。


```
>> load myfile.mat
```

V. Character Strings

字符串

A character string is a sequence of any number of characters enclosed in single quotes. You can assign a string to a variable.

一个字符串是一个被单引用号围住的字符序列，字符数量是任意的（不过太多不太好）。你可以将一个字符串分配给一个变量。

```
>> myText = 'Hello, world';
```

If the text includes a single quote, use two single quotes within the definition.

如果文本包含了单引号，那使用双引号来定义。

```
>> otherText = 'You"re right'
```

```
otherText =
```

```
You're right
```

myText and otherText are arrays, like all MATLAB variables. Their class or data type is char, which is short for character.

变量 myText 和 otherText 是数组，像所有的 MATLAB 变量一样。它们的类或者说数据类型是 char（字符），character 的简写。

```
>> whos myText
```

Name	Size	Bytes	Class	Attributes
myText	1x12	24	char	

You can concatenate strings with square brackets, just as you concatenate numeric arrays.

你可以串联字符串使用方括号 []，就像你串联数值数组一样。

```
>> longText = [myText, ' - ', otherText]
```

```
longText =
```

```
Hello, world - You're right
```

To convert numeric values to strings, use functions, such as num2str or int2str.

为了将数字值转换成字符串，使用函数如 num2str 或 int2str。

```
>> f = 71;  
c = (f-32)/1.8;  
tempText = ['Temperature is ',num2str(c),'C']
```

```
tempText =
```

```
Temperature is 21.6667C
```

VI. Functions

函数

MATLAB provides a large number of functions that perform computational tasks. Functions are equivalent to subroutines or methods in other programming languages.

MATLAB 提供数量庞大的函数来演算计算任务。Functions（函数）等价于其他编程语言里的 subroutines（子程序）或者 method（方法）。

Suppose that your workspace includes variables A and B, such as
假设你的工作站包含变量 A 和 B，如：

```
>> A = [1 3 5];  
>> B = [10 6 4];
```

To call a function, enclose its input arguments in parentheses:
为调用函数，将输入参数包含在圆括号里：

```
>> max(A);
```

If there are multiple input arguments, separate them with commas:
如果有多个输入参数，使用逗号,将它们分开：

```
>> max(A,B);
```

Return output from a function by assigning it to a variable:
从函数中返回输出，将其分配给一个变量：

```
>> maxA = max(A);
```

When there are multiple output arguments, enclose them in square brackets:
当有多个输出参数时，用方括号[]包住它们：

```
>> [maxA,location] = max(A);
```

Enclose any character string inputs in single quotes:

输入为字符串时加上单引号：

```
>> disp('hello world');  
hello world
```

To call a function that does not require any inputs and does not return any outputs, type only the function name, The `clc` function clears the Command Window:

调用函数有时不需要任意输入参数，也可能不返回任何输出，只要打一下函数名就行：

```
>> clc （清空命令行界面，最重要的就是函数，在实践中熟练函数吧！）
```

VII. Plots

绘图

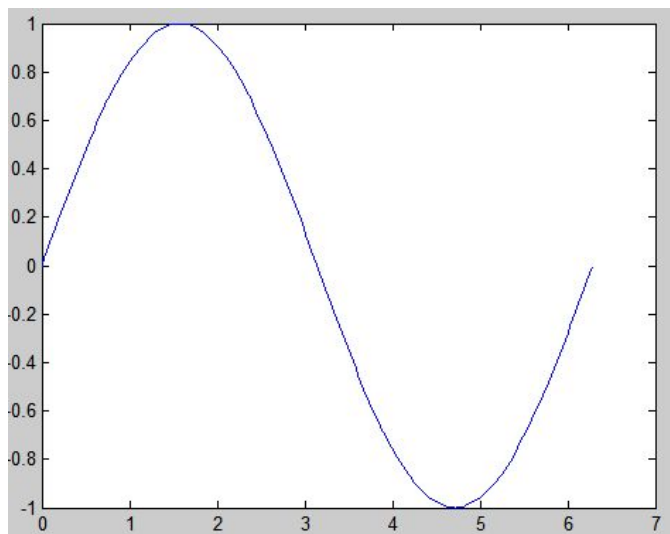
VII. 1. Line Plots

线性绘图（二维）

To create two-dimensional line plots, use the `plot` function. For example, plot the value of the sine function from 0 to 2π :

创建二维线图，使用 `plot` 函数。例如绘出正弦函数从 0 到 2π 的值：

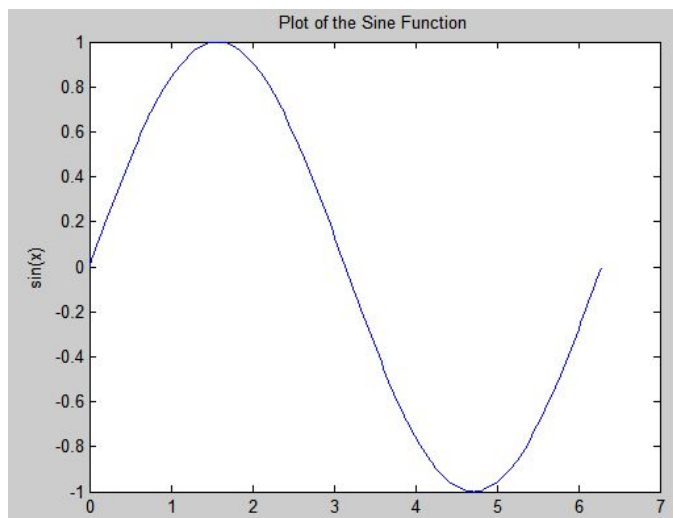
```
>> x = 0:pi/100:2*pi;  
y = sin(x);  
plot(x,y)
```



You can label the axes and add a title.

你能够为坐标轴加上标签，也可以加上标题。

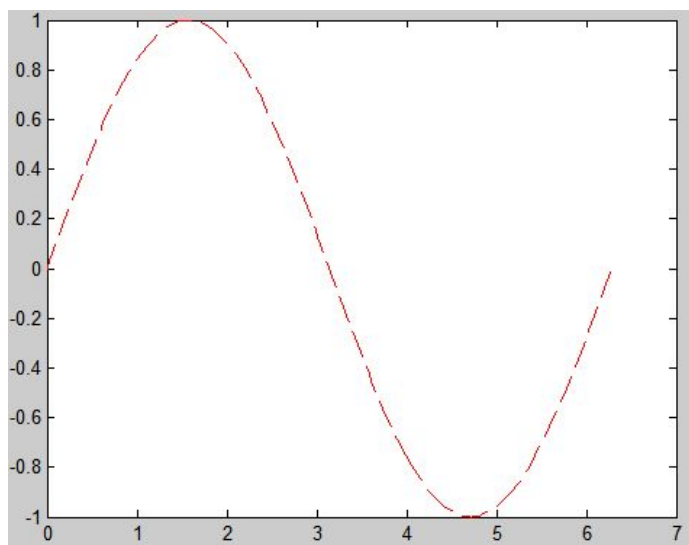
```
>> xlabel('x')  
ylabel('sin(x)')  
title('Plot of the Sine Function')
```



By adding a third input argument to the plot function, you can plot the same variables using a red dashed line.

在 plot 函数加上第三个输入参数，你可以使用红色虚线绘出同样的变量。

```
>> plot(x,y,'r--')
```



The 'r--' string is a line specification. Each specification can include characters for the line color, style, and marker. A marker is a symbol that appears at each plotted data point, such as a +, o, or *. For example, 'g:*' requests a dotted green line with * markers.

'r--'字符串是线的详述。每一个详述可以包含代表线颜色，类型和标记的字符。一个标记是一个符号，出现在绘制的每个数据点上，如一个+, o, 或者 *。例如'g:*' 要求绿色的点线有着*标记。

Notice that the titles and labels that you defined for the first plot are no longer in the current figure

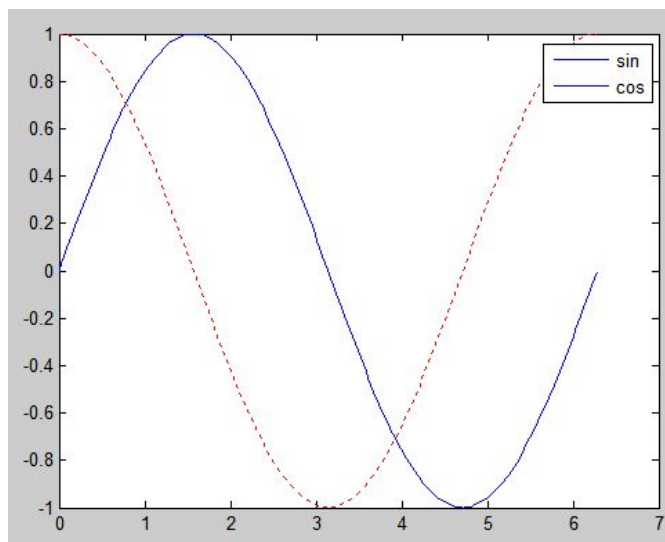
window. By default, MATLAB clears the figure each time you call a plotting function, resetting the axes and other elements to prepare the new plot.

注意：你第一次绘制定义标题和标签不会再出现在当前的图像窗口。默认情况下，MATLAB 在你每一次调用绘图函数时清空图像，重新设置坐标轴和其他元素准备新的绘制。

To add plots to an existing figure, use hold.

为了在原有图像中继续绘图，使用 hold 命令。

```
>> x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
hold on
y2 = cos(x);
plot(x,y2,'r:')
legend('sin','cos')
>>
```



Until you use hold off or close the window, all plots appear in the current figure window.

除非你使用 hold off 命令或关闭窗口，否则所有的绘图都会在当前图像窗口显示。

VII. 2. 3-D Plots

三维绘图

Three-dimensional plots typically display a surface defined by a function in two variables, $z = f(x,y)$.

三维绘图一般显示一个平面，由一个带两个输入参数的函数 $z = f(x,y)$ 定义。

To evaluate z , first create a set of (x,y) points over the domain of the function using meshgrid.

为了计算 z ，首先使用 meshgrid 在函数域上创建一组点 (x,y)

Example: Evaluate the function $x \cdot \exp(-x^2 - y^2)$

over the range $-2 < x < 2$, $-4 < y < 4$,

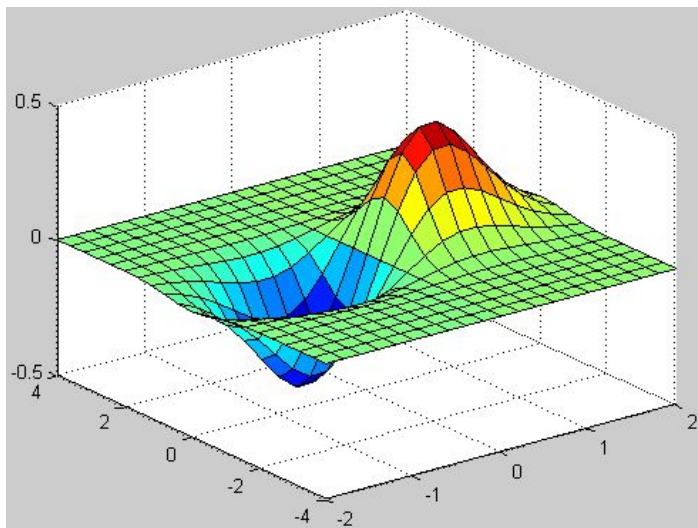
例子：计算函数 $x \cdot \exp(-x^2 - y^2)$ 在函数域 $-2 < x < 2$, $-4 < y < 4$,

```
>> [X,Y] = meshgrid(-2:2:2, -4:4:4);
      Z = X .* exp(-X.^2 - Y.^2);
```

Then, create a surface plot.

然后，绘制平面图。

```
>> surf(X,Y,Z)
```



Both the surf function and its companion mesh display surfaces in three dimensions. surf displays both the connecting lines and the faces of the surface in color. mesh produces wireframe surfaces that color only the lines connecting the defining points.

surf 函数和对应的 mesh 函数会在三维上显示平面。surf 函数显示有颜色的连接线和平面表面，mesh 函数生成平面的线框，就是那些连接定义点的有颜色的线。

VII. 3. Subplots

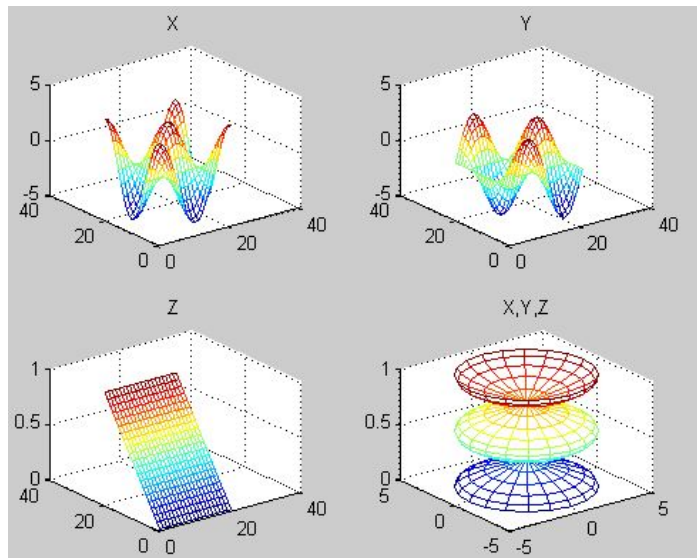
子图

You can display multiple plots in different subregions of the same window using the subplot function. 你可以在同一个窗口显示多个图在不同的子区域里，使用 subplot 函数。

For example, create four plots in a 2-by-2 grid within a figure window. 例如，在图像窗口创建 4 幅图在 2 行 2 列网格里。

```
>> t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
```

```
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
subplot(2,2,4); mesh(X,Y,Z); title('X,Y,Z');
```



The first two inputs to the subplot function indicate the number of plots in each row and column. The third input specifies which plot is.

subplot 函数的前两个输出参数指明了在每行和每列图的数量，第三个参数确定了是哪个图。

第二次翻译 2015/8/10/23:00 潮州家里

VIII. Scripts

脚本

The simplest type of MATLAB program is called a script. A script is a file with a .m extension that contains multiple sequential lines of MATLAB commands and function calls. You can run a script by typing its name at the command line.

类型最简单的 MATLAB 程序被称为脚本。一个脚本是一个有着.m 扩展名的文件，里面包含多行 MATLAB 命令和函数调用。你可以在命令行输入其名字来运行脚本。

VIII. 1. Sample Script

简单脚本

To create a script, use the edit command,
创建脚本，使用 edit 命令，

```
>> edit plotrand
```

This opens a blank file named `plotrand.m`. Enter some code that plots a vector of random data:

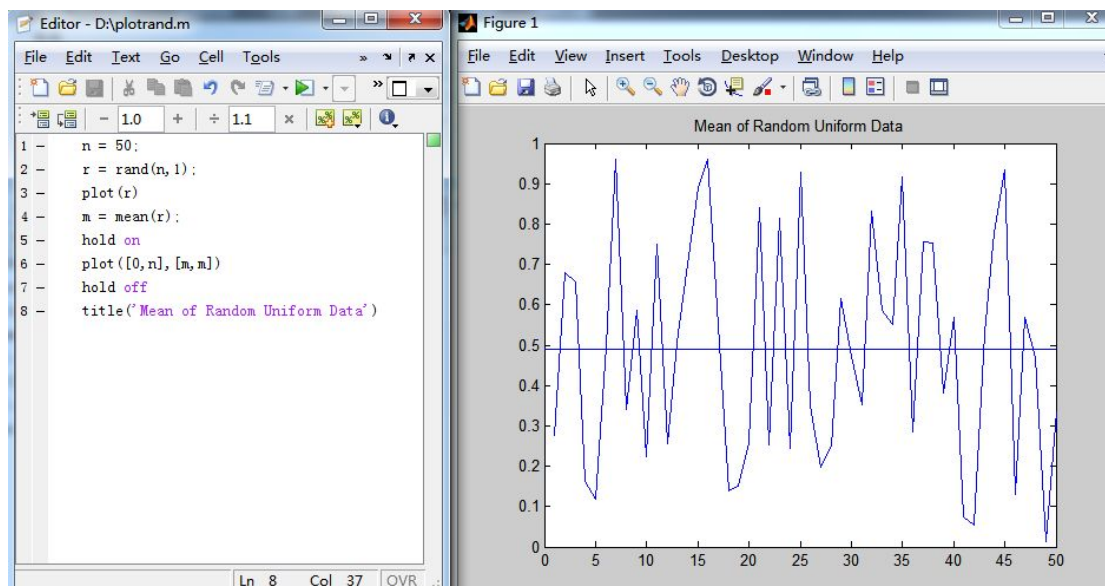
这个打开了一个空白的文件，命名为 `plotrand.m`。输入一些代码绘出一组随机数：

```
n = 50;
r = rand(n,1);
plot(r)
```

Next, add code that draws a horizontal line on the plot at the mean:

接下来，加上能够在图上画出平均数水平线的代码：

```
m = mean(r);
hold on
plot([0,n],[m,m])
hold off
title('Mean of Random Uniform Data')
```



Whenever you write code, it is a good practice to add comments that describe the code. Comments allow others to understand your code, and can refresh your memory when you return to it later. Add comments using the percent (%) symbol.

不管你什么时候写代码，加上注释去描述代码是个好习惯。注释可以让其他人明白你的代码，并且可以让你更新记忆，比如当你离开很久后再去看。加上注释使用百分号%符号。

```
% Generate random data from a uniform distribution
% and calculate the mean. Plot the data and the mean.
```

```
n = 50;           % 50 data points
r = rand(n,1);
plot(r)
```



```
% Draw a line from (0,m) to (n,m)
m = mean(r);
hold on
plot([0,n],[m,m])
hold off
title('Mean of Random Uniform Data')
```

Save the file in the current folder. To run the script, type its name at the command line:

在当前目录保存文件。为了运行脚本，在命令行输入文件名称：

```
>> plotrand
```

You can also run scripts from the Editor by pressing the Run button.

你也可以在编辑器上运行脚本，点击运行 Run 按钮（或按 F5 哦）。

VIII. 2. Loops and Conditional Statements

循环和条件语句

Within a script, you can loop over sections of code and conditionally execute sections using the keywords for, while, if, and switch.

在脚本里，你可以循环和条件执行一部分代码，使用关键字 for, while, if 和 switch。

For example, create a script named calcmean.m that uses a for loop to calculate the mean of five random samples and the overall mean.

例如，创建一个名为 calcmean.m 的脚本，使用 for 循环计算 5 个随机样本的平均值，再求总的平均值。

```
nsamples = 5;
npoints = 50;

for k = 1:nsamples
    currentData = rand(npoints,1);
    sampleMean(k) = mean(currentData);
end
overallMean = mean(sampleMean)
```

Now, modify the for loop so that you can view the results at each iteration. Display text in the Command Window that includes the current iteration number, and remove the semicolon from the assignment to sampleMean.

现在，修改循环语句方便你查看每次迭代的结果。在命令行显示包含当前迭代数字的文本，只要去掉分配变量 sampleMean 处的分号。

```

for k = 1:nsamples
    iterationString = ['Iteration #',int2str(k)];
    disp(iterationString)
    currentData = rand(npoints,1);
    sampleMean(k) = mean(currentData)
end
overallMean = mean(sampleMean)

```

When you run the script, it displays the intermediate results, and then calculates the overall mean.
当你运行脚本,它会显示中间的结果, 然后计算总的平均数。

```
calcmean
```

```
Iteration #1
```

```
sampleMean =
```

```
    0.5001
```

```
Iteration #2
```

```
sampleMean =
```

```
    0.5001    0.5185
```

```
Iteration #3
```

```
sampleMean =
```

```
    0.5001    0.5185    0.4927
```

```
Iteration #4
```

```
sampleMean =
```

```
    0.5001    0.5185    0.4927    0.5704
```

```
Iteration #5
```

```
sampleMean =
```

```
    0.5001    0.5185    0.4927    0.5704    0.4970
```

```
overallMean =
```

```
0.5158
```

In the Editor, add conditional statements to the end of `calcmean.m` that display a different message depending on the value of `overallMean`.

在编辑器, 在 `calcmean.m` 文件的最后加上条件语句, 根据变量 `overallMean` 的值显示不同的信息。

```
if overallMean < .49
    disp('Mean is less than expected')
elseif overallMean > .51
    disp('Mean is greater than expected')
else
    disp('Mean is within the expected range')
end
```

Run `calcmean` and verify that the correct message displays for the calculated `overallMean`. For example:

运行 `calcmean`, 核实计算的 `overallMeans` 是否显示正确的信息, 例如:

```
overallMean =
```

```
0.5178
```

```
Mean is greater than expected
```

VIII. 3. Script Locations

脚本位置

MATLAB looks for scripts and other files in certain places. To run a script, the file must be in the current folder or in a folder on the search path.

MATLAB 在特定的位置查找脚本和其他文件, 文件必须放在当前目录或者放在搜索路径中 (默认位置)。

By default, the MATLAB folder that the MATLAB Installer creates is on the search path. If you want to store and run programs in another folder, add it to the search path. Select the folder in the Current Folder browser, right-click, and then select Add to Path.

默认情况下, MATLAB 安装时创建的 MATLAB 目录都在搜索路径上。如果你想在其他目录里保存和运行程序, 把它加入搜索路径。在当前目录的浏览窗口选择目录, 右击, 然后选择加入路径。

IX. Help

帮助

All MATLAB functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax. There are several ways to access this information from the command line:

所有的 MATLAB 函数有支持文档，包含例子、函数输入输出、调用格式的详细说明。在命令行上，有几个方式可以获取这些信息：

Open the function documentation in a separate window using the `doc` command.

在另外的窗口打开函数文档，使用 `doc` 命令。

```
>> doc mean
```

Display function hints (the syntax portion of the function documentation) in the Command Window by pausing after you type the open parentheses for the function input arguments.

当你敲入左括号后暂停输入参数，在窗口上将显示函数暗示（函数文档的格式部分）。

```
>> mean(
```



View an abbreviated text version of the function documentation in the Command Window using the `help` command.


在命令窗口查看简略的函数文档版本，使用 `help` 命令

```
>> help mean
```

```
mean    Average or mean value.
```

```
For vectors, mean(X) is the mean value of the elements in X. For
matrices, mean(X) is a row vector containing the mean value of
each column. For N-D arrays, mean(X) is the mean value of the
```

Access the complete product documentation by clicking the help icon.

查看完整的产品文档，点击帮助图标 。

Part 2: Programming

X. Programming: Control Flow

编程指南：控制流

X. 1. Conditional Control — if, else, switch

条件控制— if, else, switch

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement. For example:

条件语句使你在运行时能够选择执行哪一块代码。最简单的条件语句是 if 语句。例如：

```
% Generate a random number
a = randi(100, 1);
```

```
% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

if statements can include alternate choices, using the optional keywords elseif or else. For example:

如果语句可以包括可选的选项，使用可选的关键字 elseif 或者 else。例如：

```
a = randi(100, 1);

if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
```

```
>> help randi
randi Pseudorandom integers from a uniform discrete distribution.
生成均匀分布的伪随机数。
```

Alternatively, when you want to test for equality against a set of known values, use a switch statement. For example:

或者，当你想测试一组已知值的相等性，使用 switch 语句。
例如：

```
[dayNum, dayString] = weekday(date, 'long', 'en_US');
```

```
switch dayString
    case 'Monday'
        disp('Start of the work week')
    case 'Tuesday'
        disp('Day 2')
    case 'Wednesday'
        disp('Day 3')
    case 'Thursday'
        disp('Day 4')
    case 'Friday'
        disp('Last day of the work week')
    otherwise
        disp('Weekend!')
end
```

For both if and switch, MATLAB executes the code corresponding to the first true condition, and then exits the code block. Each conditional statement requires the end keyword.

if 和 switch 只执行对应第一个正确条件的代码，然后终止代码块。每个条件语句需要 end 的关键词。

In general, when you have many possible discrete, known values, switch statements are easier to read than if statements. However, you cannot test for inequality between switch and case values. For example, you cannot implement this type of condition with a switch:

通常，当你有许多可能不连续的已知值，switch 语句比 if 语句更容易阅读。可是，你对 switch 和 case 中的值不能进行不等式测试。例如，你不能使用 switch 实现这种类型的条件：

```
yourNumber = input('Enter a number: ');
```

```
if yourNumber < 0
    disp('Negative')
elseif yourNumber > 0
    disp('Positive')
else
    disp('Zero')
end
```

X. 2. Array Comparisons in Conditional Statements

条件语句中的数组比较

It is important to understand how relational operators and if statements work with matrices. When you want to check for equality between two variables, you might use

```
if A == B, ...
```

明白关系操作和 if 语句在矩阵中怎样作用是很重要的。当你想在两个变量中检查相等性。你可以使用

```
if A == B, ...
```

This is valid MATLAB code, and does what you expect when A and B are scalars. But when A and B are matrices, `A == B` does not test if they are equal, it tests where they are equal; the result is another matrix of 0s and 1s showing element-by-element equality. (In fact, if A and B are not the same size, then `A == B` is an error.)

这是合法的 MATLAB 代码，当 A 和 B 都是标量它运行得好好。但当 A 和 B 是矩阵，`A == B` 不测试它们是否相等，它测试在哪个部分它们相等；结果是另外的 0-1 矩阵，一个元素一个元素地显示相等性。（事实上，当 A 和 B 的规格不同，`A==B` 会出错。）

```
>> A = magic(4);    B = A;    B(1,1) = 0;
A == B
```

```
ans =
```

```

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

The proper way to check for equality between two variables is to use the `isequal` function:

```
if isequal(A, B), ...
```

更合适的方式检查两个变量的相等性是使用 `isequal` 函数：

```
if isequal(A, B), ...
```

`isequal` returns a scalar logical value of 1 (representing true) or 0 (false), instead of a matrix, as the expression to be evaluated by the if function. Using the A and B matrices from above, you get

当这个表达式被 if 函数计算时，`isequal` 返回标量的逻辑值 1（表示真）或 0（假），而不是矩阵。利用上述的矩阵 A 和 B，你可以得到

```
>> isequal(A, B)
```

```
ans =
```

```
0
```

Here is another example to emphasize this point. If A and B are scalars, the following program will never reach the "unexpected situation". But for most pairs of matrices, including our magic squares with interchanged columns, none of the matrix conditions $A > B$, $A < B$, or $A == B$ is true for all elements and so the else clause is executed:

这里有其他的例子去强调这个点。如果 A 和 B 是标量，下面的程序将永远不会到达"unexpected situation"。但是对于大多数矩阵对，包括我们行列交换的魔术方阵，对于所有的元素没有一个矩阵条件 $A > B$, $A < B$, or $A == B$ 是真的，所以 else 从句会被执行：

```
if A > B
    'greater'
elseif A < B
    'less'
elseif A == B
    'equal'
else
    error('Unexpected situation')
end
```

Several functions are helpful for reducing the results of matrix comparisons to scalar conditions for use with if, including

几个很好用的函数在使用 if 条件语句时可以简化矩阵比较的结果，包括

```
isequal
isempty
all
any
```

请使用 help 命令查看函数用法！！

X. 3. Loop Control — for, while, continue, break

循环控制— for, while, continue, break

This section covers those MATLAB functions that provide control over program loops.

这一节涵盖了在程序循环中提供控制的 MATLAB 函数。

for

The for loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements:

for 循环在固定预定的次数内重复一组语句。一个相匹配的 end 描述了这个语句：

```
>> for n = 3:32
    r(n) = rank(magic(n));
end
```



```

r
>> help rank
rank    Matrix rank.

```

The semicolon terminating the inner statement suppresses repeated printing, and the `r` after the loop displays the final result.

分号终结内部语句并压住重复的输出，在循环之后 `r` 显示了最后的结果。

It is a good idea to indent the loops for readability, especially when they are nested:

为了可读性缩排循环是个好主意，特别当它们嵌套在一起：

```

for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
end

```

while

The while loop repeats a group of statements an indefinite number of times under control of a logical condition. A matching `end` delineates the statements.

在逻辑条件控制，while 循环在不确定次数下重复一组语句。一个相匹配的 `end` 描述了这个语句。

Here is a complete program, illustrating while, if, else, and end, that uses interval bisection to find a zero of a polynomial:

这是一个完整的程序，强调 while,if,else 和 end，使用二分区间去寻找多项式的零点。

```

a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x

```

The result is a root of the polynomial $x^3 - 2x - 5$, namely

结果是多项式 $x^3 - 2x - 5$ 的根，即是

```

x =
    2.09455148154233

```

The cautions involving matrix comparisons that are discussed in the section on the if statement also apply to the while statement.

在 if 一节讨论的复杂矩阵比较也同样应用于 while 语句。

第三次翻译 2015/8/11 21:12 潮州家中

continue

The continue statement passes control to the next iteration of the for loop or while loop in which it appears, skipping any remaining statements in the body of the loop. The same holds true for continue statements in nested loops. That is, execution continues at the beginning of the loop in which the continue statement was encountered.

当 continue 语句在 for 和 while 循环中出现时，会跳过控制流到下一个迭代，即跳过循环体中剩下的所有语句。在复杂的循环中 continue 也是同样表现的。也就是说，当遇到 continue 语句时，会重新从循环头部开始执行。

The example below shows a continue loop that counts the lines of code in the file magic.m, skipping all blank lines and comments. A continue statement is used to advance to the next line in magic.m without incrementing the count whenever a blank line or comment line is encountered:

下面的例子展示了一个带有 continue 的循环，统计 magic.m 文件中代码的行数，跳过所有空格行和注释。一个 continue 语句被用来提前到达 magic.m 文件的下一行而不增加计数，当它遇到空格行或者注释行的时候。

```
fid = fopen('magic.m','r'); % 以只读形式打开这个文件
count = 0;
while ~feof(fid) % 当文件指示符 fid 没有到达末尾
    line = fgetl(fid); % 得到文件的一行
    if isempty(line) || strncmp(line,'% ',1) || ~ischar(line) % 行是空的或者以%开头 或者不是字符
        continue
    end
    count = count + 1;
end
fprintf('%d lines\n',count); % 打印计数
fclose(fid); % 关闭文件指针
```

break

The break statement lets you exit early from a for loop or while loop. In nested loops, break exits from the innermost loop only.

Break 语句使你能够提前结束 for 和 while 循环。在复杂的循环中，break 只能结束循环的最里层，想想看几个循环嵌套。

Here is an improvement on the example from the previous section. Why is this use of break a good idea?

下面是之前章节 `example` 的一个改进版。为什么说使用 `break` 是一个好主意？

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b % 精确值？请百度
    x = (a+b)/2;
    fx = x^3-2*x-5; % 二分法求函数零点？
    if fx == 0 % 找到函数零点？
        break
    elseif sign(fx) == sign(fa) % 符号一致，都是正数或负数
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

X. 4. Program Termination — `return`

程序终结者 — `return`

This section covers the MATLAB `return` function that enables you to terminate your program before it runs to completion.

这一节涉及到 MATLAB 函数的返回，使你能够在函数还没跑完之前就终结你的程序。

`return`

`return` terminates the current sequence of commands and returns control to the invoking function or to the keyboard. `return` is also used to terminate keyboard mode. A called function normally transfers control to the function that invoked it when it reaches the end of the function. You can insert a `return` statement within the called function to force an early termination and to transfer control to the invoking function.

`return` 终结当前一系列的命令并返回控制到调用它的函数或者键盘输入的地方。`return` 也被用到键盘模式下(即命令行格式下)。一个被调用的函数，当它执行到最后时，正常情况下会转移控制到调用它的函数。你可以在被调用的函数中插入一个 `return` 语句，提前强制终止并转移控制到调用函数。

X. 5. Vectorization

矢量化

One way to make your MATLAB programs run faster is to vectorize the algorithms you use in constructing the programs. Where other programming languages might use for loops or DO loops, MATLAB can use vector or matrix operations. A simple example involves creating a table of logarithms:

使你的 MATLAB 程序跑得更快的一个方法是矢量化你在构建程序时使用的算法。其他编程语言可能使用 for 循环或 do-while 循环，但 MATLAB 能使用矢量或矩阵运算。一个简单的例子，涉及到创建一个使用对数算法生成的表。

```
x = .01;
for k = 1:1001
    y(k) = log10(x);
    x = x + .01;
end
```

A vectorized version of the same code is
一个矢量化版本，同样的代码是

```
x = .01:.01:10; % 起始：步长：终止
y = log10(x);
```

For more complicated code, vectorization options are not always so obvious.
对于一些复杂的代码，矢量化选择不总是很明显，就是说用到了可是你不知道。

X.6. Preallocation

预先配置

If you cannot vectorize a piece of code, you can make your for loops go faster by preallocating any vectors or arrays in which output results are stored. For example, this code uses the function zeros to preallocate the vector created in the for loop. This makes the for loop execute significantly faster:

如果你不能矢量化代码块，通过预先配置一些存放输出结果的矢量或数组，能够使你的循环运行得更快。例如：下面这个代码使用 zeros 函数预先配置在循环中创建的矢量，使得 for 循环执行起来显著性地快。

```
r = zeros(32,1); % 三十二行一列？全是零？
for n = 1:32
    r(n) = rank(magic(n)); % 请 help rank
end
```

Without the preallocation in the previous example, the MATLAB interpreter enlarges the r vector by one element each time through the loop. Vector preallocation eliminates this step and results in faster execution.

在上面的例子中，没有预先配置的话，MATLAB 解释器在循环过程中，每一次都会因为一个元素扩大 r 矢量的长度。

XI. Programming: Scripts and Functions

编程指南：脚本和函数

XI. 1. Overview

概况

The MATLAB product provides a powerful programming language, as well as an interactive computational environment. You can enter commands from the language one at a time at the MATLAB command line, or you can write a series of commands to a file that you then execute as you would any MATLAB function. Use the MATLAB Editor or any other text editor to create your own function files. Call these functions as you would any other MATLAB function or command.

MATLAB 产品提供了一个强大的编程语言，同时也是一个交互式计算环境。你可以马上在 MATLAB 命令行输入该语言的命令，或者写下一连串的命令到文件中，然后像其他 MATLAB 函数一样执行。使用 MATLAB 编辑器或者其他的文本编辑器去创建你自己的函数文件，只要你想，在其他的 MATLAB 函数或命令行中都可以调用这些函数。

There are two kinds of program files:

有两类程序文件

- Scripts, which do not accept input arguments or return output arguments. They operate on data in the workspace.
- 脚本：不接受输入参数和不返回输出参数，它们在工作区间 workspace 进行数据运算。
- Functions, which can accept input arguments and return output arguments. Internal variables are local to the function.
- 函数：接受输入参数和返回输出参数。内部变量只是存在于函数中。

If you are a new MATLAB programmer, just create the program files that you want to try out in the current folder. As you develop more of your own files, you will want to organize them into other folders and personal toolboxes that you can add to your MATLAB search path.

如果你是一个新的 MATLAB 编程人员，那么只需要在当前目录中创建你想要解决问题的程序文件。当你开发了越来越多自己的文件，你将会想要组织它们到其他的文件夹中或者个人工具箱，解决方法是，你可以将这些文件放在 MATLAB 搜索路径中。

If you duplicate function names, MATLAB executes the one that occurs first in the search path.

如果你重复了函数名，MATLAB 将会执行在搜索路径中出现的第一个文件。

To view the contents of a program file, for example, myfunction.m, use

为了查看程序文件的内容，例如：myfunction.m，使用以下命令

`type myfunction`

XI. 2. Scripts

脚本

When you invoke a script, MATLAB simply executes the commands found in the file. Scripts can operate on existing data in the workspace, or they can create new data on which to operate. Although scripts do not return output arguments, any variables that they create remain in the workspace, to be used in subsequent computations. In addition, scripts can produce graphical output using functions like `plot`.

当你引用脚本，MATLAB 简单地执行在文件中找到的命令。脚本能够使用在工作站存在的数据进行运算，或者它们在运算中可以产生新的数据。尽管脚本不会返回输出参数，但它们创建的变量仍然留在工作站中，可以用来进行后继的运算。另外，脚本能够生成图形的输出，只要使用函数，如 `plot`。

For example, create a file called `magicrank.m` that contains these MATLAB commands:

例如，创建一个名为 `magicrank.m` 的文件，包含以下 MATLAB 命令：

```
% Investigate the rank of magic squares 研究魔术方阵的秩？
r = zeros(1,32);
for n = 3:32
    r(n) = rank(magic(n));
end
r
bar(r)
```

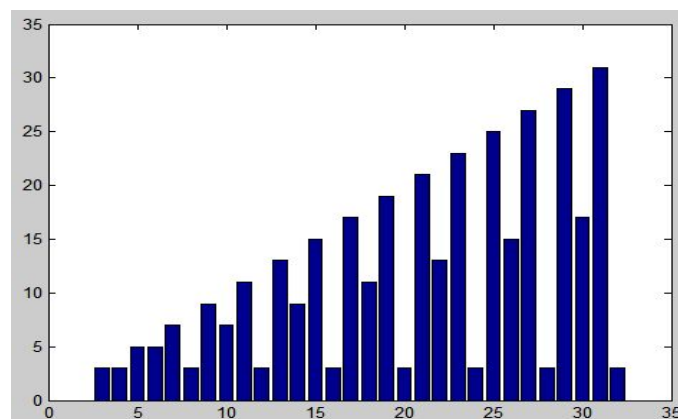
Typing the statement

敲入以下语句

`magicrank`

causes MATLAB to execute the commands, compute the rank of the first 30 magic squares, and plot a bar graph of the result. After execution of the file is complete, the variables `n` and `r` remain in the workspace.

使得 MATLAB 执行命令，计算前三十个魔术方阵的秩，然后绘出结果的条形图。执行文件结束以后，变量 `n` 和 `r` 仍然留在工作站。



XI. 3. Functions

函数

Functions are files that can accept input arguments and return output arguments. The names of the file and of the function should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt.

函数是一个能够接受输入参数和返回输出参数的文件。文件名应该和函数名一样。函数在它们自己的工作区间操作变量，与你在 MATLAB 命令行提示模式下的工作区间是相互分隔的。

A good example is provided by rank. The file rank.m is available in the folder : toolbox/matlab/matfun
rank 函数提供了一个好例子，这个 rank.m 文件在这个目录下是可用的：toolbox/matlab/matfun

You can see the file with

你可以查看这个文件，使用

`type rank`

Here is the file:

```
function r = rank(A,tol)
%   RANK Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

s = svd(A); % 请 help svd
if nargin==1 % 输入参数为一个
    tol = max(size(A')) * max(s) * eps;
end
r = sum(s > tol);
```

The first line of a function starts with the keyword function. It gives the function name and order of arguments. In this case, there are up to two input arguments and one output argument.

函数的第一行以关键字 `function` 开始。它点明了函数的名字和参数的顺序。在这个例子中，它们有两个输入参数和一个输出参数。

The next several lines, up to the first blank or executable line, are comment lines that provide the help text. These lines are printed when you type

下面几行，在第一个空白行或可执行行之上，是注释行，提供了帮助文本。这些行将会被打印出来，如果你敲以下命令。

`>> help rank`

rank Matrix rank.

`rank(A)` provides an estimate of the number of linearly independent rows or columns of a matrix `A`.

`rank(A,tol)` is the number of singular values of `A` that are larger than `tol`.

`rank(A)` uses the default `tol = max(size(A)) * eps(norm(A))`.

Class support for input `A`:

float: double, single

Overloaded methods:

`gf/rank`

`xregdesign/rank`

Reference page in Help browser

`doc rank`

The first line of the help text is the H1 line, which MATLAB displays when you use the `lookfor` command or request help on a folder.

帮助文本的第一行是大标题行，就是当你使用搜索命令或者在某一文件夹中请求 `help` 时 MATLAB 显示的东西。

The rest of the file is the executable MATLAB code defining the function. The variable `s` introduced in the body of the function, as well as the variables on the first line, `r`, `A` and `tol`, are all local to the function; they are separate from any variables in the MATLAB workspace.

文件的其他部分是定义在函数中的可执行 MATLAB 代码。出现在函数体中的变量 `s` 和出现在第一行的变量 `r`, `A`, `tol`，全部是函数中的局部变量；它们与 MATLAB 工作区间的变量是相互分隔的。

This example illustrates one aspect of MATLAB functions that is not ordinarily found in other programming languages—a variable number of arguments. The `rank` function can be used in several different ways:

这个例子说明了 MATLAB 函数的一个特征，在其他编程语言一般不会出现——参数数量是可变的。可以使用几种不同的方式调用 `rank` 函数。

`rank(A)`

`r = rank(A)`

`r = rank(A,1.e-6)`

Many functions work this way. If no output argument is supplied, the result is stored in `ans`. If the second input argument is not supplied, the function computes a default value. Within the body of the function, two quantities named `nargin` and `nargout` are available that tell you the number of input and output arguments involved in each particular use of the function. The `rank` function uses `nargin`, but does not need to use `nargout`.

很多函数都是这样用的。如果没有提供输出参数，结果会保存在变量 `ans` 中，如果没有提供第二

个输入参数，函数使用默认值运算。在函数体内，有两个名为 `nargin` 和 `narout` 的隐藏变量是可用的，告诉你每一个特定函数使用时，输入和输出参数的数量。`rank` 函数使用了 `nargin`，但是并不需要使用 `nargout`。

XI. 4. Types of Functions

函数类型

MATLAB offers several different types of functions to use in your programming.
MATLAB 提供了几种不同的函数类型，方便你在编程中使用。

Anonymous Functions

匿名函数

An anonymous function is a simple form of the MATLAB function that is defined within a single MATLAB statement. It consists of a single MATLAB expression and any number of input and output arguments. You can define an anonymous function right at the MATLAB command line, or within a function or script. This gives you a quick means of creating simple functions without having to create a file for them each time.

一个匿名函数是一个 MATLAB 函数的简单组成，使用单一的 MATLAB 语句来定义。它包含单一的 MATLAB 表达式或任意数量的输入输出参数。你可以马上在 MATLAB 命令行定义匿名函数，或者在一个函数或脚本文件中。这样提供了一个创建简单函数的快速方式，而不是每次都需要为它们创建一个文件。

The syntax for creating an anonymous function from an expression is
从一个表达式创建一个匿名函数的格式如下

```
f = @(arglist)expression
```

The statement below creates an anonymous function that finds the square of a number. When you call this function, MATLAB assigns the value you pass in to variable `x`, and then uses `x` in the equation `x.^2`:

下面的语句创建了一个匿名函数，计算数字的平方。当你调用这个函数，MATLAB 把你输入的值指派给变量 `x`，然后使用 `x` 计算方程式 `x.^2`：

```
sqr = @(x) x.^2;
```

To execute the `sqr` function defined above, type
执行上面的 `sqr` 函数，敲入

```
a = sqr(5)
a =
    25
```

Primary and Subfunctions

主函数和子函数

Any function that is not anonymous must be defined within a file. Each such function file contains a required primary function that appears first, and any number of subfunctions that can follow the primary. Primary functions have a wider scope than subfunctions. That is, primary functions can be called from outside of the file that defines them (for example, from the MATLAB command line or from functions in other files) while subfunctions cannot. Subfunctions are visible only to the primary function and other subfunctions within their own file.

任何非函数必须定义在文件中。每一个这样的函数文件必须包含一个主函数，第一个出现的那个，任意数量的子函数可以出现在主函数后面。主函数比子函数有更大的作用域。也就是，主函数可以在文件外被调用（例如，从 MATLAB 命令行或者其他文件中的函数），而子函数并不能，子函数只对主函数或者属于同一文件的子函数可见。如下：

以上子函数翻译为劣函数比较好，因为是并列关系而不是包含关系，且 Sub 前缀是替代的意思。

```
function out1= f1( in1 )
out1=in1;
f2(out1);
end
function out2= f2( in2 )
out2=in2;
end
```

在同一文件定义了两个函数

```
>> f1(3)
```

```
ans =
```

```
3
```

```
>> f2(3)
```

```
Undefined function 'f2' for input arguments of type 'double'.
```

但是只有第一个出现的函数，即主函数可以在外部调用。

The rank function shown in the section on Functions is an example of a primary function.

出现在 Functions 一节的 rank 函数是一个主函数的例子。

Private Functions

私有函数

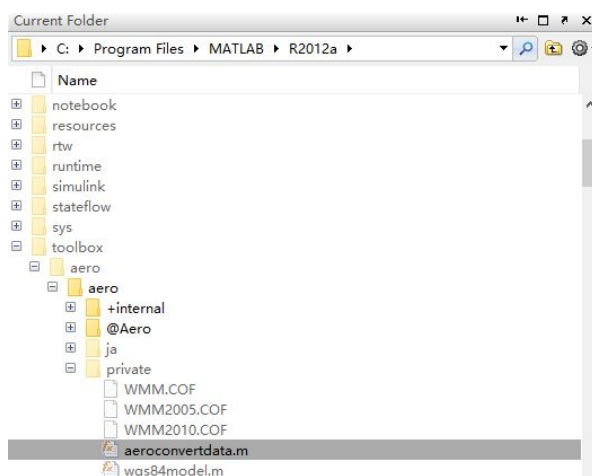
A private function is a type of primary function. Its unique characteristic is that it is visible only to a

limited group of other functions. This type of function can be useful if you want to limit access to a function, or when you choose not to expose the implementation of a function.

私有函数是主函数的一种类型。它独特的特征是只对有限函数组可见。这类函数可以使用的场合是，如果你想禁止一个函数的使用，或者你想选择性地不暴露函数的实现。

Private functions reside in subfolders with the special name `private`. They are visible only to functions in the parent folder. For example, assume the folder `newmath` is on the MATLAB search path. A subfolder of `newmath` called `private` can contain functions that only the functions in `newmath` can call.

私有函数存放在带有特殊名字 `private` 的子文件夹中。它们只对父目录的函数可见。例如，假设一个在 MATLAB 搜索路径中的 `newmath` 的文件夹，它有一个子目录名为 `private`，里面包含各种函数，那它们只能被 `newmath` 中的函数调用。如下图：



Because private functions are invisible outside the parent folder, they can use the same names as functions in other folders. This is useful if you want to create your own version of a particular function while retaining the original in another folder. Because MATLAB looks for private functions before standard functions, it will find a private function named `test.m` before a nonprivate file named `test.m`.

因为私有函数在父目录外不可见，所以在其他文件夹中的函数可以使用一样的名字。这是非常有用的，如果你想创建属于自己版本的特定函数，同时想保留其他文件夹中的原始函数。因为在寻找标准函数之前会寻找私有函数，例如寻找非私有函数 `test.m` 之前，它将先找私有函数 `test.m`。

越翻译越拗口，歪果仁怎么废话那么多，各种定语从句，还带一堆补语从句。相比之下，我们中国人显得不是很有人情，直接各种证明，不带感情色彩，而老外各种温馨提示。

Nested Functions

嵌套函数

You can define functions within the body of another function. These are said to be nested within the outer function. A nested function contains any or all of the components of any other function. In this example, function B is nested in function A:

你可以在定义函数在另一个函数的体内。它们也被称为被嵌套在外部函数中（好拗口啊，我感觉我的大脑快死了。）一个嵌套函数包含任意其他函数的一部分或全部组成部分。在下面例子中，

函数 B 嵌套在函数 A 中

```
function x = A(p1, p2)
...
B(p2)
    function y = B(p3)
    ...
    end
...
end
```

Like other functions, a nested function has its own workspace where variables used by the function are stored. But it also has access to the workspaces of all functions in which it is nested. So, for example, a variable that has a value assigned to it by the primary function can be read or overwritten by a function nested at any level within the primary. Similarly, a variable that is assigned in a nested function can be read or overwritten by any of the functions containing that function.

和其他函数一样，一个嵌套函数有它自己的工作区间，在函数中被使用的变量都会被保存。另外，它也有权限使用所有嵌套它的函数的工作区。那么，举一个例子，一个被主函数赋值的变量，它能够被嵌套在主函数的任意层次的子函数读或重写。同样的，一个在嵌套函数中被赋值的变量，可以被包含它的任意函数读和重新。

上面其实就是作用域。主函数中的变量是全局变量，子函数中的是局部变量。全局变量到处可被调用，局部变量只能在指定的函数中被调用。但是，MATLAB 里面有自己全局变量的说明，见下一部分。所以只能说变量在函数链中只能向下传递，不能向上。而下文的真正定义的全局变量却可以。

Function Overloading

函数重载

Overloaded functions act the same way as overloaded functions in most computer languages. Overloaded functions are useful when you need to create a function that responds to different types of inputs accordingly. For instance, you might want one of your functions to accept both double-precision and integer input, but to handle each type somewhat differently. You can make this difference invisible to the user by creating two separate functions having the same name, and designating one to handle double types and one to handle integers. When you call the function, MATLAB chooses which file to dispatch to based on the type of the input arguments.

重载函数以大多数计算机语言的重载方式一样运行。重载函数非常有用，当你需要创建一个函数，而它要对相应的不同类型的输入作出反应。例如，你可能想你的一个函数既能接受 `double` 浮点数的输入，又能接受 `int` 整数的输入，但是处理每一种类型的方式是不同的。你可以使这些不同对用户隐藏，只要创建两个独立的函数，有同样的名字，但设计一个处理浮点数，一个处理整数。当你调用函数时，MATLAB 根据输入参数的类型选择哪一个文件进行发送。

XI. 5. Global Variables

全局变量

If you want more than one function to share a single copy of a variable, simply declare the variable as global in all the functions. Do the same thing at the command line if you want the base workspace to access the variable. The global declaration must occur before the variable is actually used in a function. Although it is not required, using capital letters for the names of global variables helps distinguish them from other variables. For example, create a new function in a file called falling.m:

如果你想更多的函数分享同一个变量,简单的使用 `global` 相似所有函数中定义变量。在命令行中,也做同样的事,如果你想通过工作站去使用那个变量。全局声明必须在变量使用前出现。没必要,但是全局变量还是要以大写字母命名,方便它们与其他变量区分。例如,创建一个新的函数,文件名叫 `falling.m`。

```
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

Then interactively enter the statements

交互式输入以下语句（交互式！！）

```
global GRAVITY
GRAVITY = 32;
y = falling((0:1:5));
```

The two global statements make the value assigned to GRAVITY at the command prompt available inside the function. You can then modify GRAVITY interactively and obtain new solutions without editing any files.

两个全局语句使得在命令模式下赋值的 GRAVITY 也在函数中可用。你可以交互地修改 GRAVITY。不用修改任何文件就可以得到新的结果。

XI. 6. Command vs. Function Syntax

命令模式 VS 函数格式

You can write MATLAB functions that accept string arguments without the parentheses and quotes.

你可以写 MATLAB 函数,只接受字符串,而没有圆括号和引号

```
foo a b c
```

That is, MATLAB interprets as

MATLAB 解释为:

```
foo('a','b','c')
```

However, when you use the unquoted command form, MATLAB cannot return output arguments. For

example.

可是，当你使用不带引号的命令形式，MATLAB 不会返回输出参数，例如：

`legend apples oranges`

creates a legend on a plot using the strings apples and oranges as labels. If you want the legend command to return its output arguments, then you must use the quoted form:

在图上创建图例，使用字符串苹果和橘子作为标记。如果你想使 legend 命令返回输出参数，你必须使用引号形式：

`[leg,h,objh] = legend('apples','oranges');`

In addition, you must use the quoted form if any of the arguments is not a string.

另外，如果其他的参数不是字符串形式，你必须使用引号形式。

Caution: While the unquoted command syntax is convenient, in some cases it can be used incorrectly without causing MATLAB to generate an error.

注意：虽然非引号命令格式很方便，但在一些情况下会被不正确地使用，然而并不会使 MATLAB 产生错误。

Constructing String Arguments in Code

在代码中构造字符串参数

The quoted function form enables you to construct string arguments within the code. The following example processes multiple data files, August1.dat, August2.dat, and so on. It uses the function int2str, which converts an integer to a character, to build the file name:

引号形式可以使你在代码中构造字符串参数。下面的例子处理多个数据文件，August1.dat，August2.dat 等。它使用函数 int2str，将数字转成字符，然后构成文件名。

```
for d = 1:31
    s = ['August' int2str(d) '.dat'];
    load(s)
    % Code to process the contents of the d-th file
end
```