仲恺農業工程學院

ZHONGKAI UNIVERSITY OF AGRICULTURE AND ENGINEERING

实验参考资料——MATLAB 编程入门教程 V1

计算科学学院　信息与计算科学系

目　　录

第 1 部分 MATLAB 简介

MATLAB 是 matrix & laboratory 两个词的组合，即矩阵实验室，是由美国 MathWorks 公司发布的，用于算法开发、数据可视化、数据分析以及数值计算的高级计算技术语言和交互式环境。具体来说，MATLAB 的基本数据单位是矩阵，可以进行各种矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序（如 C/C++、Java 等）等，主要应用于工程计算、控制设计、信号处理与通讯、图像处理、信号检测、金融建模设计与分析等领域。

在本参考教程中，将介绍与 MATLAB 编程相关的基础知识，主要包括 MATLAB 界面环境使用、矩阵与数组运算、程序设计的控制流语句、函数与 M 文件操作，MATALB 绘图等。在案例编程实现中，以大家熟悉的数学方法去描述问题和求解。此处使用的版本是 MATLAB R2014b，其运行主界面如图 1-1 所示。

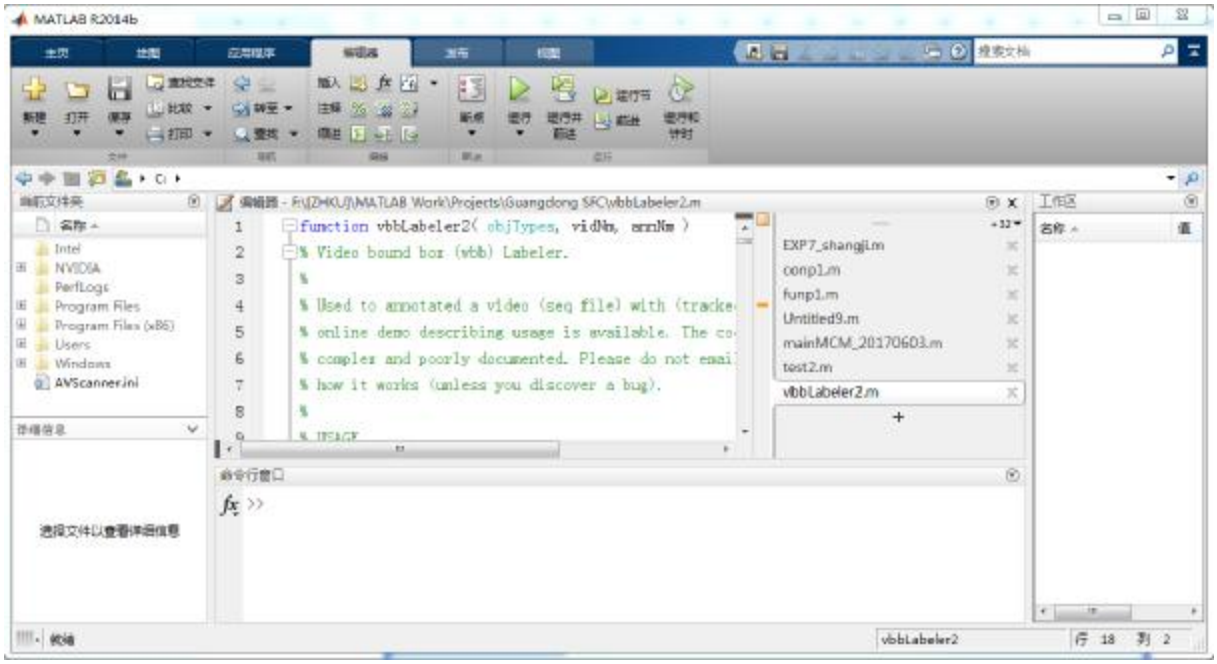


图 1-1 MATLAB R2014b 主界面

主界面中，左上角的窗口是当前文件夹窗口 Current Folder，用来显示当前工作目录下的所有文件，通常函数文件或数据文件都默认保存在当前工作目录，可以更改此目录以保存文件，也可以将这个窗口关闭，只要点击该窗口上的关闭按钮则可；左下角则显示在当前文件夹窗口中被选中的文本文件的基本信息 Details。

中间窗口是编辑器 Editor（上）和命令行窗口 Command Window（下），它是 MATLAB 界面窗口的重要组成部分，编辑器用于编辑和保存脚本文件；而部分人交互操作可直接通过命令行窗口完成，在该窗口中输入命令，然后回车，即可执行命令。

右侧是工作空间窗口 Workspace，用于查看保存在工作空间的各变量，可以方便地查看各变量的相关信息如变量名、行列数、各元素中的最大值与小值，以及该变量在内

存分配的字节数等。

启动 MATLAB 后，在命令行窗口中会出现“>>”，该符号表示 MATLAB 环境已准备就绪，正等待命令的输入。

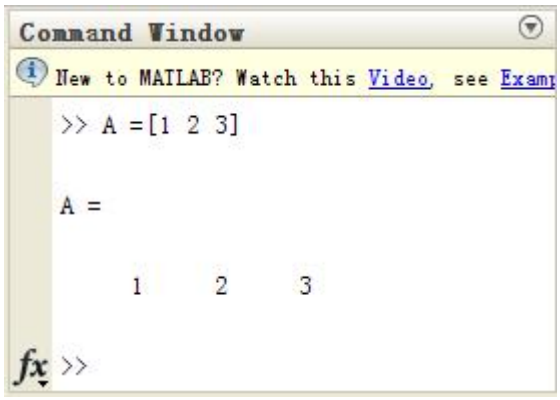


图 1-2 直接在命令窗口中输入并执行指令

如图 1-2 所示，现输入指令：

```
>> A = [1 2 3]
```

然后回车，则出现运算结果：

```
A =
    1    2    3
```

上述指令表示输入一个一行三列的矩阵（即行向量），并将此矩阵（向量）保存在变量 A 中。关于矩阵及向量，将在第二部分中做详细解释。此时，Workspace 窗口中会出现一条信息，如图 1-3 所示，其显示了变量 A 和它的值、最小元素、最大元素及类型等信息。

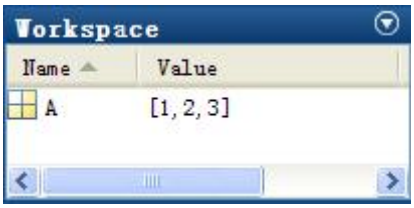


图 1-3 工作空间窗口

由于 MATLAB 功能强大且应用广泛，限于本参考资料篇幅原因，此处所介绍的内容非常有限，要深入学习 MATLAB，还需要参看其它相关参考资料。除此之外，MATLAB 自身也提供了强大的帮助功能，其内容丰富，形式多样，进入 MATLAB 帮助文档界面后，点击相应的连接即可。

第 2 部分 矩阵与数组运算

一、矩阵与数组的概念

在 MATLAB 中，数组和矩阵本身是没有区别的，在内存中是一样的。只是针对不同的运算方式，将其称为数组运算或矩阵运算。如果运算是按元素对应进行的，则称为数组运算；如果按线性代数学中的方式运算，则称为矩阵运算。

MATLAB 中数据是以数组形式来组织的，这将大大提高计算速度根据需要。根据需要，可分别创建一维数组和二维数或多维数组。下面将举例说明数组的运算方式和矩阵的运算方式。

例 2-1 计算下列两个矩阵的相乘和相加运算：

$$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}, B=\begin{pmatrix}-1 & 1\\ 0 & 2\end{pmatrix}$$

在 MATALB 界面中的 Command Window 窗口中输入如下指令，然后回车：

```
>> A=[1 2;3 4]          % 输入的矩阵保存在变量 A 中
A =
      1     2
      3     4
>> B=[-1 1;0 2]         % 输入的矩阵保存在变量 B 中
B =
      -1     1
       0     2
>> C = A*B              % 矩阵方式运算，即按线性代数中的矩阵相乘方式计算
C =
     -1     5
     -3     11
>> A.*B                 % 数组方式运算，即逐元素对应相乘
ans =
    -1     2
     0     8
```

说明：

1、MATLAB 中，矩阵或向量按元素逐个输入的方法，就是将所有元素放在一对方括号内，行与行之间以分号“;”隔开，每一行中各元素之间以空格或逗号隔开。

2、MATLAB 指令输入时，必需在英文状态下输入，否则会提示语法错误；

3、如果一条指令以分号“;”作为结束符，则运算结果并不显示在 Command Window 窗口中，但会保存在 Workspace 中，可通过 Workspace 窗口进行查看。如果没有分号，运算结果不仅会保存在 Workspace 中，还将显示在 Command Window 中；

4、百分号“%”表示注释，从%开始到行未为方便人阅读所加的注释，不是指令中

的一部分；

5、本资料中所有的 MATLAB 指令都用加粗字体；

6、“＝”称为赋值符号，如果表达式的值没有赋值给某个指定的变量名，则将表达式的值赋值给的变量名 ans，当下一行指令也没有指定将表达式的值赋值给某个指定变量时，将用下一行指令的执行结果覆盖 ans 的值。

7、与 C/C++、Java 等语言不同，MATLAB 中的变量无需要声明就可直接使用。

8、矩阵运算方式下运算符左侧没有小黑点：“*”，数组运算方式下运算符左侧有小黑点：“.*”,其它运算符也是如此，只要运算符左侧有小黑点的就是数组方式运算，否则就矩阵方式运算。

二、矩阵与数组的创建

(1) 逐个元素输入法

例 2-2 逐一输入元素生成矩阵或数组。

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

(2) 步长生成法获得一组数组

使用冒号运算符“:”,如“m:n”,表示在m和 n 之间以步长为 1 生成一个一维数组；

例 2-3 指定步长生成数组。

```
>> A = 2:6
A =
    2    3    4    5    6
```

如果指定步长，则应采用形式“m:d:n”，其中 d 代表步长，最后一个元素是 m 加上步长的整数倍，不一定是 n。

例 2-4 指定步长生成数组。

```
>> A = 2:3:7
A =
    2    5
>> A = 2:3:9
A =
    2    5    8
```

(3) 定数线性采样法生成一维数组

A = linspace(a,b,n)

在 a 和 b 之间按等间隔采样生成元素为 n 的一维数组，包括端点。

例 2-5 指定区间和元素个数生成数组。

```
>> linspace(1,5,4)
ans =
   1.0000    2.3333    3.6667    5.0000
```

(4) 特殊矩阵（数组）的生成方法

1、ones(m,n)：生成 m 行 n 列，且元素全是 1 的矩阵。

2、zeros(m,n)：生成 m 行 n 列，且元素全是 0 的矩阵。

3、eye(m,n)：生成 m 行 n 列的矩阵，当 m 和 n 不相等时，左上角方阵为单位矩阵，m 和 n 相等时，可用 eye(n)生成一个 n 阶的单位矩阵。

例 2-6 生成单位矩阵。

```
>> eye(3,5)
ans =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
>> eye(5,3)
ans =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
     0     0     0
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

4、随机矩阵的创建。

例 2-7 生成随机矩阵

```
>> rand('state',0)      % 把均匀分布伪随机发生器置为 0 状态
>> v = rand(2,3)        % 产生一个 2×3 的随机矩阵
v =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
```

5、魔方矩阵（行和、列和与对角和都相等，为小于等于元素个数的互不相同的整数）。

例 2-8 生成魔方矩阵。

```
>> m = magic(3)          % 产生一个 3 阶魔方矩阵
m =
     8     1     6
     3     5     7
     4     9     2
```

三、矩阵元素的访问操作

以下以举例的方式说明矩阵的访问操作如下：

例 2-9 访问和操作元素。

```
>> v =  [1 2 3 4 5 6 7];   % 生成一个行向量
>> v(3)                    % 查询第三个元素的值
ans =
     3
>> v(3)=23                 % 将第三个元素的值设为 23
v =
     1     2    23     4     5     6     7
>> v([1 2 6])=[11 12 16]  % 将下标为 1、2、6 的三元素值设为 11、12、16
v =
    11    12    23     4     5    16     7
>> v(4:end)                % 查询第 4 至最后元素之间的所有元素
ans =
     4     5    16     7
>> v(1:5)                  % 查询第 1 至 5 个元素
ans =
    11    12    23     4     5
>> m=[1 2 3;4 5 6]  % 产生一个新矩阵 m
m =
     1     2     3
     4     5     6
>> m(2,3)             % 查询第 2 行第 3 列位置上的元素
ans =
     6
>> m(:,2)             % 查询第 2 列元素上所有行的元素
ans =
     2
     5
>> m(2,:)             % 查询第 2 行上所有列的元素
ans =
     4     5     6
>> m(2,[1 2])         % 查询第 2 行上的第 1、2 列位置上的元素
ans =
     4     5
>> m(1,[2 3])=[88 99]  % 将第 1 行上的第 2、3 列上的元素分别设为 88 和 99
m =
     1    88    99
     4     5     6
>> m(2,2)=518           % 将第 2 行 2 列位置上的元素设为 518
m =
     1    88    99
```

```
    4   518     6
```

四、常用的运算符和运算函数

在该部分，列出一些比较常用的，稍作说明，其它的以列表形式给出，以便查询。其它更多的运算函数请参看帮助系统。

(1) 数组（矩阵）运算

表 2-1 运算符与运算函数

运算函数名	实现功能	运算函数名	备注
plus 或+	加	minus 或-	减
times 或.*	数组乘法	mtimes 或*	矩阵乘法
rdivide 或./	数组右除	mrdivide 或/	矩阵右除
ldivide 或.\	数组左除	mldivide 或\	矩阵左除
power 或.^	数组求幂	mpower 或^	矩阵求幂
cumprod	累计求乘积	cumsum	累计求和
diff	计算差分	prod	计算数组乘积
sum	计算数组元素的和	fix	向零方向圆整
ceil	向数轴正向圆整	floor	向数轴负向圆整
round	向最近的整数圆整	mod(a,b)	a 除以 b 的余数
dot	计算点积	cross	计算叉积

1、加法：c = a+b 或 c = plus(a,b)

参数 a 和 b 要求有相同的行列数。

2、减法：c = a-b 或 c = minus(a,b)

参数 a 和 b 要求有相同的行列数。

3、数组方式的乘法（Array multiply）：c = a.*b 或 c = times(a,b)

逐元素相乘，且参数 a 和 b 要求有相同的行列数。

4、矩阵方式的乘法（Matrix multiplication）：C = A*B

按线性代数中的矩阵乘法进行运算 A 的列数必须等于 B 的行数。

5、数组方式右除（Right array division）：x = A./B 或 x = rdivide(A,B)

逐元素相除，要求参数 A， B 的行列数相同。

例 2-10 组数方式右除矩阵。

```
>> A = [2 4 6 8;3 5 7 9];
>> B = 10*ones(2,4);   % ones(2,4)生成 2 行 4 列且元素全是 1 的矩阵，
>> x = A./B
x =

    0.2000    0.4000    0.6000    0.8000
    0.3000    0.5000    0.7000    0.9000
```

6、数组方式左除（Left array division）：x = B.\A 或 x = ldivide(B,A)

逐元素相除，要求参数 A， B 的行列数相同。

例 2-11 组数法左除矩阵。

```
>> A = ones(2, 3);
>> B = [1 2 3;4 5 6];
>> x = B.\A
x =
   1.0000    0.5000    0.3333
   0.2500    0.2000    0.1667

>> C = 2;
>> D = [1 2 3;4 5 6];
>> x = D.\C
x =

   2.0000    1.0000    0.6667
   0.5000    0.4000    0.3333
```

7、矩阵方式右除（Solve systems of linear equations xA = B for x）：x = B/A 或 x = mrdivide(B,A)

当 A 是标量时，则是逐元素相除以该标题值；

当 A 是方阵时，且方程的解存在，则求得的结果为方程的解；

当 A 不是方阵时，则按最小均方差给出一个近似解。

例 2-12 求解方程有唯一解的方程，x*A = B：

```
>> A = [1 1 3; 2 0 4; -1 6 -1];
>> B = [2 19 8];
>> x = B/A
x =
   1.0000    2.0000    3.0000
```

以下计算不确定解的方程 x*C= D，按最小均方差给出方程的近似解：

```
>> C = [1 0; 2 0; 1 0];
>> D = [1 2];
>> x = D/C
Warning: Rank deficient, rank = 1, tol =  6.280370e-16.
x =
        0    0.5000         0
```

8、矩阵方式左除（Solve systems of linear equations Ax = B for x）：x = A\B 或 x = mldivide(A,B)

此函数的用法类似于右除。

9、数组方式计算幂（Array power）：Z = X.^Y

此函数的作法类似于右除。

10、矩阵方式计算幂（Matrix power ）：c = a^b 或 c = mpower(a,b)

11、累计求积：B = cumprod(A) 或 B = cumprod(A,dim)

例 2-13 连续累计求积。

```
>> A = 1:5
A =
    1    2    3    4    5
>> cumprod(A)
ans =
    1    2    6   24  120
>> A = [1 2 3; 4 5 6];
>> cumprod(A,1)
ans =
    1    2    3
    4   10   18
>> cumprod(A,2)
ans =
    1    2    6
    4   20  120
```

12、累计求和（Cumulative sum）：B = cumsum(A) 或 B = cumsum(A,dim)

例 2-14 连续累计求和。

```
>> A = 1:5
A =
    1    2    3    4    5
>> cumsum(A)
ans =
    1    3    6   10   15
>> A = [1 2 3; 4 5 6];
>> cumsum(A,1)
ans =
     1    2    3
     5    7    9
>> cumsum(A,2)
ans =
     1    3    6
     4    9   15
```

(2) 逻辑运算

表 2-2 常用逻辑运算符

运算函数名	实现功能	运算函数名	备注
and 或&	逻辑与	not 或~	逻辑非
or 或丨	逻辑或	xor	异或
all	指定维数的全部元素逻辑与	any	指定维数的全部元素逻辑或
find	确定非零元素的位置		
islogical	检测是否为逻辑类型	logical	将数值转换为逻辑类型

例 2-15 逻辑运算。

```
>>  A = [1,3,1;-3,3,-2],B = [0,2,0;1,3,0]
A =
     1     3     1
    -3     3    -2
B =
     0     2     0
     1     3     0
>> C = and(A,B)      % 逻辑与，输入为 1 的位置， A,B 中对应位置上的元素都为非零值
C =
     0     1     0
     1     1     0
>> C = ~B            % 逻辑非，B 中非零值的位置输出 0，零值的位置输出 1.
C =
     1     0     1
     0     0     1
>> C = or(A,B)       % 逻辑或，A 与 B 对应位置至少有一个不是零，则输出 1，否则输出零
C =
     1     1     1
     1     1     1
>> C = xor(A,B)      % 逻辑异或，A 与 B 对应位置一个为零，另一个为非零，则在对应位置，则输出 1，否则输出零
C =
     1     0     1
     0     0     1
>> C = all(B)        % B 的每一列的全部元素逻辑与
C =
     0     1     0
```

```
>> C = all(B,2)      % B 的第二维即每一行的全部元素逻辑与
C =
     0
     0
>> C = any(B,1)      % B 的第一维即每一列的全部元素逻辑或
C =
     1     1     0
>> C = any(B,2)      % B 的第二维即每一行的全部元素逻辑或
C =
     1
     1
>> A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B = A>10          % A 中元素值>10，则在该位置输出 1，否则输出 0
B =
     1     0     0     1
     0     1     0     0
     0     0     0     1
     0     1     1     0
>> [ r,c] = find(B);   % 找到 B 中值为 true(1)的位置，依次将行号和列号分别保存在 r 和 c 中。
>> r',c'
ans =
     1     2     4     4     1     3
ans =
     1     2     2     3     4     4
>> C = find(B);   C'
ans =
     1     6     8    12    13    15
```

最后一行指令，找到 B 中值为 true(1)的位置，并按列串联起来组成一个列向量，输出值为 true(1)的位置，比如按列串联起来共有 16 个元素。在 B 中第三行第四列位置上的值是 1，则按列串联后，它排在在 15 个位置，所以该位置就输出 15.

(3) 关系运算

表 2-3 关系运算符

运算函数名	实现功能	运算函数名	备注
eq 或= =	检测相等	ge 或>=	检测大于等于
gt 或>	检测大于	le 或<=	检测小于等于
lt 或<	检测小于	ne 或~=	检测不相等
isequal	检测不相等		

因为关系运算的使用比较类似，其它可参看帮助。

例 2-16 关系运算。

```
>> A = magic(2)
A =
     1     3
     4     2
>> B = repmat(A,2,2)     %  repmat 将 4 个矩阵 A 为排列成 2 行 2 列
B =
     1     3     1     3
     4     2     4     2
     1     3     1     3
     4     2     4     2
>> C = magic(4)
C =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B==C
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     1     0     0     0
```

输出结果为一个逻辑矩阵，值为 1 位置，对应在 B 和 C 的相同位置上的两个元素相等，为 0 则不相等。

五、矩阵的基本特征参数计算

(1) 矩阵信息的基本参数

包括元素个数函数 numel(A)、行列数 size(A)、行列数中的最大者 length(A)、及其最大元素 max(A)、最小元素 min(A)。

例 2-17 矩阵信息的基本参数。

```
>> A = [1 2 3; 4 5 6];  %以下采用同一个示例矩阵
A =
     1     2     3
     4     5     6
>> numel(A)              %统计矩阵的元素个数
ans =
     6
>> size(A)               %计算矩阵的行列数
ans =
     2     3
>> length(A)             %计算行数与列数中的最大者
ans =
     3
>> max(A(:))             %求出矩阵中所有元素中的最大者
ans =
     6
>> min(A(:))             %求出矩阵中所有元素中的最小者
ans =
     1
```

(2) 与矩阵相关的基本特征参数

基本特征参数包括矩阵的行列式(det)、秩(rank)、逆矩阵(inv)、条件数 cond、特征值与特征向量(eig)及对角化。

1、计算行列式：

例 2-18 计算行列式。

```
>> A = [2 3 4;1 1 9;1 2 -6]
>> det(A)                % 求 A 的行列式
ans =
    1
```

2、计算矩阵的秩：

例 2-19 矩阵的秩。

```
>> rank(A)
ans =
     3
```

3、逆矩阵：

例 2-20 逆矩阵。

```
>> inv(A)
ans =
   -24    26    23
    15   -16   -14
     1    -1    -1
```

4、条件数，反映 AX = b 中，如果 A 或 b 发生细微变化，解变化的剧烈程度。如果条件数很大说明是病态方程方程，不稳定方程。

例 2-21 条件数。

```
>> cond(A)
ans =
  575.8240
```

5、特征值与特征向量。

如果 eig(A)只有一个返回参数，则它是一个向量数组，元素为 A 的各个特征值。

例 2-22 特征值与特征向量。

```
>> eig(A)
ans =
    5.0914
   -0.0243
   -8.0670
```

当 eig(A)有两个返回参数时，两个参数皆为矩阵，第一个参数的列向量对应 A 的不相关单位特征向量。第二个参数矩阵是一个对角矩阵，对角线元素为 A 的特征值。

```
>> [V,D] = eig(A)
V =
    0.7936    0.8491   -0.0747
    0.5823   -0.5272   -0.6983
    0.1766   -0.0343    0.7118
D =
    5.0914         0         0
         0   -0.0243         0
         0         0   -8.0670
```

对下演示通过矩阵 V，对 A 进行对角化处理。

```
>> inv(V)*A*V
ans =
    5.0914   -0.0000    0.0000
    0.0000   -0.0243   -0.0000
   -0.0000   -0.0000   -8.0670
```

第 3 部分 程序设计的控制流语句

一、循环控制语句

1、for 语句

通常用来执行循环次数已知的情况，可按指定次数来重复执行循环体中的内容。基本语法格式为：

```
for m = a : b : c
    循环体语句;
end
```

首先表达式 1 是索引变量的初值，b 的值为步长，c 的值为循环索引变量的终值。当步长为 1 时，表达式 2 可以省略。

例 3-1 循环语句。

```
for m = 1:5
    x(m) = m.^2;
end
x
```

则输出如下：

```
x =
     1     4     9    16    25
```

例 3-2 计算 $\sum_{1}^{10}\frac{1}{n^2}$。

```
y = 0;
m = 10;
for n= 1:m
   y = y + 1/(n.^2);
end
y
```

输出为：

```
y =
   1.5498
```

MATLAB 中，过多的循环语句会降低程序执行速度，一般情况下，尽量使用数组方式来处理。

例如上面的循环结构，可用如下指令代替。

```
y = 0;
n = 1:10;
y = sum(1./(n.^2)),
```

for 循环可以嵌套使用，例如：

```
for m =1:4
    for n = 1:4
        A(m,n) = m*n;
    end
end
A
A =
     1     2     3     4
     2     4     6     8
     3     6     9    12
     4     8    12    16
```

2、while 语句

while 语句也是一种循环语句，一般用于事先不能确定循环次数的情况。调用语法如下：

```
while（条件表达式）
      循环语句
end
```

其执行过程：若条件表达式成立，则执行循环体语句，执行后再判断条件是否成立，如果成立，则重复执行循环体语句，否则跳出循环。

以上两个循环语句可以多重嵌套。

例 3-3 while 循环语句。

```
m = 2;
while (m<10)
    m = m*2;
end
m
m =
    16
```

二、条件转移语句

条件转移语句的语法如下：

```
if (表达式 1)
    语句部分 1;
elseif (表达式 2)
    语句部分 2;
else
    语句部分 3;
end
```

条件转移语句可以和循环语句嵌套使用。

例 3-4 条件转移语句。

```
nrows = 5;
ncols = 5;
myData = ones(nrows, ncols);
for r = 1:nrows
   for c = 1:ncols
      if (r == c)
         myData(r,c) = 2;
      elseif (abs(r - c) == 1)
         myData(r,c) = -1;
      else
         myData(r,c) = 0;
      end
   end
end
myData
```

则输出为：

```
myData =
     2    -1     0     0     0
    -1     2    -1     0     0
     0    -1     2    -1     0
     0     0    -1     2    -1
     0     0     0    -1     2
```

三、开关控制语句

switch 分支条件（数值或字符串）
case 数值（或字符串）条件一
　　对应的执行语句;
case 数值（或字符串）条件二
　　对应的执行语句;
……
otherwise
　　各个 case 语句均不符合时，执行的语句;
end

例 3-5 开关控制语句。

```
mynumber = input('Enter a number:');
switch mynumber
    case -1
        disp('negative one');  % disp 只用来将参数的值显示在命令窗口中
    case 0
        disp('zero');
    case 1
```

```
        disp('positive one');
    otherwise
        disp('other value');
end
```

注意，只有第一个匹配的分支执行，例如：

```
result = 52;
switch(result)
   case 52
      disp('result is 52')
   case {52, 78}
      disp('result is 52 or 78')
end
result is 52
```

三、break 与 continue 语句

两条语句配合 for 或 while 循环使用，

break 用于终止循环的执行。当在循环体内执行到该语句时，程序将跳出循环，继续执行循环语句的下一语句。

continue 语句用于中止单次循环中出现在 continue 语句后面的部分，继续执行下一次循环。

例 3-6 求 100-200 之间第一个能被 21 整除的整数。

```
for n = 100:200
    if rem(n,21)~=0        % rem 计算 n 除以 21 后所得余数
        continue;
    end
    break
end
n
n =
   105
```

第 4 部分 文件与编程

在 MATLAB 环境下，有三类重要的文件。第一类是扩展名以.mat 结尾、用于存贮数据的文件，数据以 mat 文件形式存储在硬盘中。第二类是用于存放可独立运行的脚本文件，称为 M 命令文件。如果将一系列 MATLAB 指令程序存放在一个 M 文件中，以 M 文件形式组织执行指令程序，相对于直接在 Command Window 而言，会更方便编辑、调试和运行程序，使用起来也更加简洁。第三类就是函数文件，函数文件具有参数传递功能，有参数和返回值。第二和第三类文件均以.m 作为其扩展名，下面将逐一介绍其使用方法。

一、mat 数据文件与数据导入

新一次启动 MATLAB 后，之前在 Workspace（Workspace 中的数据直接存放于内存中）里使用的各种变量和数据都不复存在，许多时候我们希望数据还能再出现，而不用繁琐的手工输入。MATLAB 提供了保存工作区中的各种数据的功能，并将数据保存在 mat 数据文件中。

mat 数据文件是 MATLAB 数据存储的标准格式，文件是标准的二进制文件，还可以 ASCII 码形式保存和加载，下面将以实例说明如何将数据保存到 mat 文件中。

几个相关函数名：

(1) save：用于将数据保存在 mat 文件中

其语法形式如下：

1、save(filename)：将工作区中所有的变量数据保存到文件名为 filename 指定的 mat 文件中。

2、save(filename, variables)：同上，区别在于该调用形式只保存指定的几个变量。

3、save(filename, '-struct', structName, fieldNames)：如果不带第四个参数，则保存结构体 structName 中的全部域，如果带上第四个参数，则保存结构体中的指定域。

4、save(filename, ..., '-append')：在已经存在的文件中增加变量数据。

5、save(filename, ..., format)：指定文件格式。

6、save(filename, ..., version)：用于指定 mat 文件的版本号。

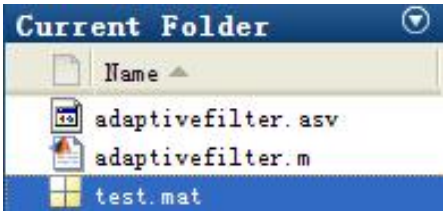
7、save filename ...：在命令行中使用，简洁方便，无需单引号括起来。

例 4-1 生成一个 4 阶魔方矩阵和一个 3 阶魔方矩阵，并全部保存在 test.mat 文件中。

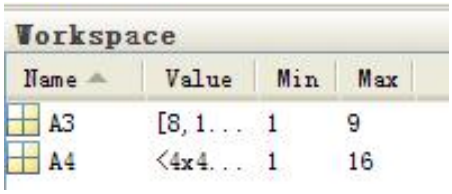
```
>> A4 = magic(4)
A4 =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
>> A3 = magic(3)
A3 =
     8     1     6
     3     5     7
     4     9     2
>> save test.mat
```

如图 4-1 所示，当指令执行完毕后，在 Current Folder 窗口（图 4-1(a)）中多出了一个文件 test.mat，同时在 Workspace 窗口（图 4-1(b)）中多出了两个变量。



(a)



(b)

图 4-1 数据保存为 mat 文件后的工作空间和当前文件夹状态

然后在命令行中输入：

>> clear

则清除 Workspace 中的所有变量，窗口中显示的内容被清空。

(2) load：用于打开 mat 文件。

在例 1 中，继续输入如下指令：

>> load test

当指令执行完毕时，Workspace 窗口中会出现如图 4-1(b)所示的变量。

例 4-2 保存变量到文件。

```
>> a = magic(4);
>> b = ones(2, 4) * -5.7;
>> c = [8 6 4 2];          % 以上三行生成三个变量 a,b,c
>> save mydata a b c       % 将变量 a,b,c 保存到文件 mydata.mat 中
>> clear                   % 清除三个变量同时 Workspace 窗口中显示的三个变量消失
>> load mydata             % 从 mydata.mat 文件中读取三个变量，并显示在 Workspace 窗口中。
```

二、其它格式文件数据导入

使用函数 importdtata 可以导入已保存的 mat 数据文件，也可导入其它格式的数据文件。该函数的语法形式如下：

- importdata('filename')，将 filename 中的数据导入到工作区中。
- A = importdata('filename')，将 filename 中的数据导入到工作区中，并保存为变量 A。
- importdata('filename','delimiter')，将 filename 中的数据导入到工作区中，以 delimiter 指定的符号作为分隔符。

下面以 2011 年高教社杯全国大学生数学建模竞赛的 A 题中的污染数据为例说明如何导入 excel 中的数据。

竞赛题中给出的数据表中，为了方便处理只将其中的数据部分复制到新的 excel 文件中，当然如果不这样做，也可以处理，相对要使用更多的程序代码。

例 4-3 导入数据文件。

将数据部分复制到 excel 文件中，假设其文件名为 wurang.xls，并将其复制到 MATLAB 的当前工作目录下，然后输入如下指令：

```
>> D = importdata('wurang.xls');
D =
    Sheet1: [319x5 double]
>> mydata = D.Sheet1;
```

上述指令执行后，Workspace 窗口中出现相应的存放数据的变量，如图 4-2 所示。

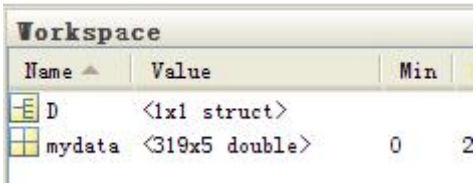


图 4-2 执行 importdata 指令后的工作空间状态

importdata 除了导入 excel 文件数据外，也可以读取文本文件中的数值型数据（各数值之间是以空隔或逗号分隔的）。

三、M 指令文件（脚本文件）

由于脚本文件编辑器（Editor）中编编辑、运行和调试方面，要比 Commond 窗口中方便。MATLAB 编程求解都是在 Editor 窗口中进行。但有时也要在 Commond 窗口查看相关信息，比如当程序语法错误或执行不通过时，会将错误提示显示在 Commond 窗口中，或都可以在 Commond 窗口中观察变量的值。

如图 4-3 所示，点击 HOME 页上的按钮“New Script”，可打开 M 文件编辑器，然后在文本编辑框中输入需要指令（脚本代码），然后点击“Run”按钮，则会弹出要求保存 M 文件的对话框，选择当前目录进行保存后，则会运行 M 脚本编辑器中的所有指令。

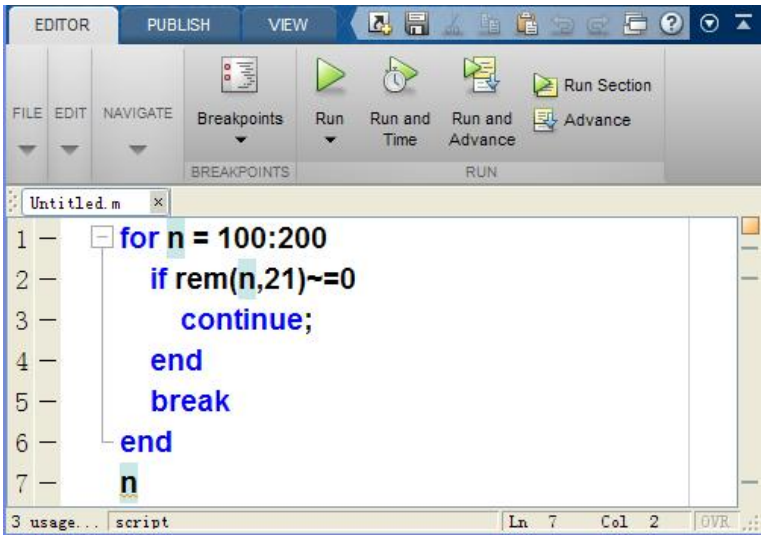


图 4-3 在.m 文件中编辑、保存并运行脚本代码

有时程序本身并没有语法错误，在编译时能通过并且能运行，但有时可能是程序本身的设计有错或出现意外的错误结果，这就需要对程序进行调试，对涉及的变量进行跟踪，以便找出问题所在，并修正。

对程序进行调试时，可以在左侧行号处通过单击设置断点，以观察各个变量的值在程序运行过程中的变化情况。断点会以红色圆圈来表示，如果再次单击断点，则会取消断点。如果程序中有断点，点击 Run 按钮后，则程序执行到断点处后进行等待，此时程序是以调试模式运行的。如果将鼠标移到某个变量处，则会实时显示该变量的值，如图 4-4 所示。

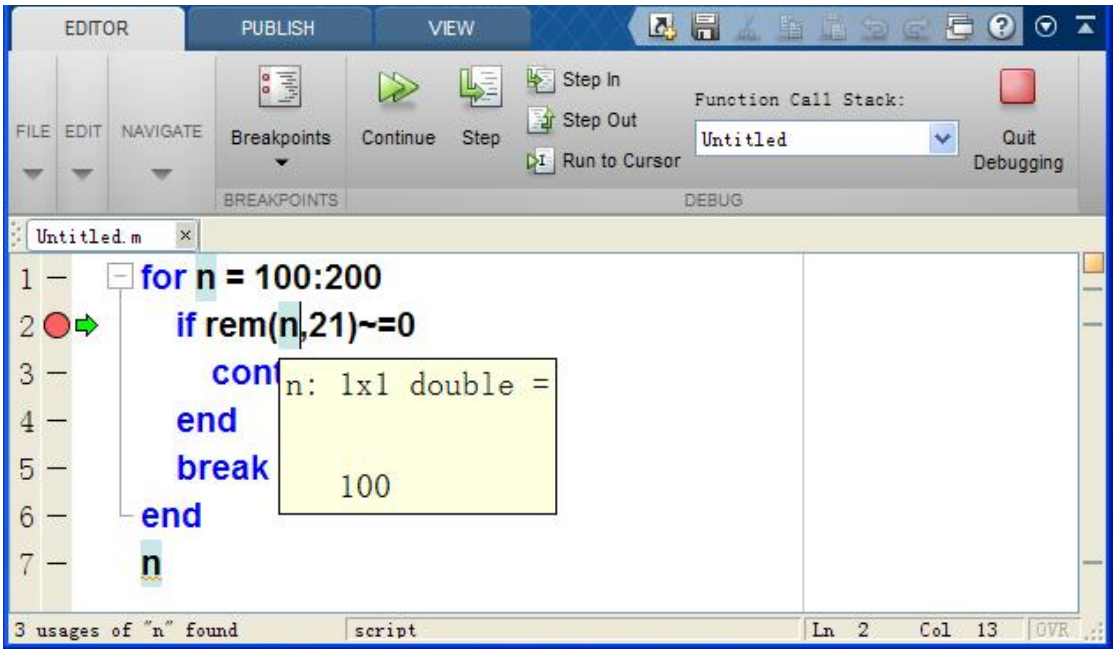


图 4-4 在.m 文件编辑器中调试程序

如果选择工具栏上的 Step 或按下 F10 键，则会继续逐条单步执行指令；如果选择 Step In 或按下 F11 键，则是进入被调函数的函数体中；如果选择 Step Out 或按下组合键 Shift+F11 时，则会从函数体中返回到主调函数所在的下一条指令中。

注：有关函数的概念及其使用，将在下一部分进行解释。

四、M 函数文件

M 函数文件的格式有严格规定，它必须以关键字“function”开头，函数声明的语法形式如下：

function [y1, ..., yn] = myfun(x1, ..., xm)

function 是关键字，myfun 是自定义的函数名，必须是合法标识任。其中 x1, ..., xm 是输入参数列表，y1, ..., yn 是返回值列表，该行必须放在 M 文件的首行。

第二行开始就是函数体，一个函数文件中可以包括多个函数，每一个函数体的最后有一个关键字 end（也可以省略），且除第一个函数之外，其余函数均作为子函数，可以相互调用。

保存函数文件时，M 函数文件的扩展名必须是.m，且文件名应和函数名相同。

函数文件与命令文件的主要区别在于：

1、函数文件一般都要带参数，一般要有返回结果，而命令文件没有参数与返回结果。

2、函数文件的变量是局部变量，运行期间有效（如果想要在函数运行完毕后变量还有效，需要将其声明为全局变量），运行完毕就自动被清除，而命令文件的变量是全局变量，执行完毕后仍被保存在内存中。

3、函数文件要定义函数名，且保存该函数文件的文件名必须是函数名.m。M 函数文件可以有多个自变量和多个因变量，当有多个自变量和/或因变量时用“[ ]”括起来。

例 4-4 编写一个单输出的函数，用于计算输入数组的平均值。

新建一个 M 文件，并在编辑窗口中输入如下代码，然后保存，文件名为 average，

```
function  y =  average (x)
if ~isvector(x)
   error('Input must be a vector'); % 输入必须为一维数组
end
y = sum(x)/length(x);
```

其中 x 是输入参数（本例中要求必须为一维数组），因为只有一个输入参数，所以无需方括号，返回值 y 也只有一个，也无需方括号。average 是函数名。

这时，就可以在 Commond 窗口或其他.m 文件中像调用 MATLAB 自带函数一样调用 average 函数了。

例如在 Commond 窗口的命令行中输入如下指令：

```
>> z = 1:99;
>> average(z)
```

则会输出如下结果：

```
ans =
   50
```

例 4-5 编写一个多输出的函数，用于计算输入数组的平均值和方差，函数名为 stat，

且 M 文件名也必须是 stat。

新建一个 M 文件，并在编辑窗口中输入如下代码，然后保存，文件名为 stat.m，

```
function [m,s] = stat(x)
n = length(x);
m = sum(x)/n;
s = sqrt(sum((x-m).^2/n));
```

例如在 Commond 窗口的命令行中输入如下指令：

```
>> values = [12.7, 45.4, 98.9, 26.6, 53.1];
>> [ave, stdev] = stat(values)
ave =
   47.3400
stdev =
   29.4124
```

例 4-6 M 函数文件中有多个函数，且第一个函数的函数名为 stat2，且多个返回值，用于计算平均值和方。

```
function [m,s] = stat2(x)
n = length(x);
m = avg(x,n);
s = sqrt(sum((x-m).^2/n));

function m = avg(x,n)
m = sum(x)/n;
```

第一个函数 stat2 是主调函数，第二个函数 avg 是子函数。

在 Commond 窗口的命令行中输入如下指令后，可得出相应的运算结果：

```
>> values = [12.7, 45.4, 98.9, 26.6, 53.1];
>> [ave,stdev] = stat(values)
ave =
   47.3400
stdev =
   29.4124
```

第 5 部分 数据与函数图形的可视化

视觉是人们感受世界、认识自然的最重要依靠。数据可视化的目的在于：通过图形，从一堆杂乱的离散数据中观察数据间的内在关系，感受由图形所传递的数据的内在本质。MATLAB 环境中一向注重数据的图形表示，并不断地采用新技术改进和完善其可视化功能。

本部分将系统地阐述：离散数据表示成图形的基本机理；曲线、曲面绘制的基本技法和指令；涉及的关键步骤是准备数据与显示数据。

一、二维图形的绘制

常用的二维绘图函数有 plot、quiver、contour。除此之外还有不太常用的绘图函数，如 polar、loglog、semilogx、semilogy、bar、barh、rose、stairs、area、pie、scatter、hist、erroerbar、stem、feather、commet、compass 等，详情可以参看 MATLAB 帮助文件或查阅其他有关 MATLAB 编程的教材，这里只介绍 plot、quiver 和 contour 三个常用绘图函数的调用。

1、plot 绘图函数简介

plot 的语法形式如下：

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
plot(...,'PropertyName',PropertyValue,...)
plot(axes_handle,...)
h = plot(...)
```

函数说明：

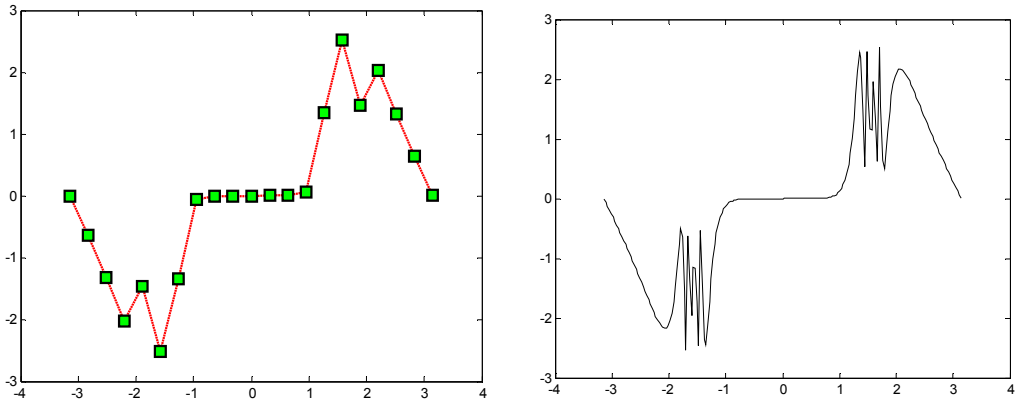
(1) 当输入参数只有一个时，如 plot(Y)，Y 是输入向量，以 Y 的下标作为横坐标，以 Y 的值作为纵坐标绘图，默认的方式是各离散点之间使用线段连接起来，当点取得足够密时，看起来就是一条光滑的曲线。

(2) plot(X1,Y1,...,Xn,Yn) 针对每一对坐标数据参数(Xi,Yi)绘制一条曲线，Xi,Yi 可以是一个向量（一维数组）。每一对坐标数据参数后可紧跟一个控制线型和/或颜色的的参数，也可以没有该参数。如果没有线型或颜色的参数，则采用默认的线型和颜色绘制曲线。注意，Xi,Yi 也可以是矩阵，情况略复杂，成对的矩阵参数必需是同型矩阵（行列数都相同），以 Xi 的某列为横坐标向量，同 Yi 中相同列号的列为纵坐标向量绘制曲线，具体请看实例演示。

例 5-1 绘制函数 $f(x)=\tan(\sin(x))-\sin(\tan(x))$ 在 $[-\pi,\pi]$ 上的不同参数下的图像。

```
x = -pi:pi/10:pi;                    %第一步准备数据：自变量数组
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...    %三个圆点“...”是续行符号
    'MarkerEdgeColor','k',...        %各点连线的颜色，
     'MarkerFaceColor','g',...       %坐标点的颜色
     'MarkerSize',10)                %'--rs'中的--表示连线线型，rs 为点形状
```

上述代码运行后，得到如图 5-1(a)所示的图像。



(a) (b)

图 5-1 例 5-1 程序的运行结果

以上只是为了说明 plot 的参数使用，通常绘图都不会使用那么多的参数，一般使用简单的实线即可，比如：

```
x = -pi:pi/100:pi;       % 采样更细，则会显示曲线的更多细节。
y = tan(sin(x)) - sin(tan(x));
plot(x,y, 'k')
```

此代码段产生如图 5-1(b)所示的图形。

注：常用的颜色表示有，b 蓝；g 绿；r 红；c 青；m 品红；y 黄；k 黑；w 白。

例 5-2 绘制平移正弦曲线簇。

```
t = 0:pi/100:2*pi;
y = sin(t);
y2 = sin(t-0.25);          %向右平衡 0.25 单位，绘图区间不变
y3 = sin(t-0.5);           %向右平衡 0.5 单位，绘图区间不变
plot(t,y,t,y2,t,y3);
hold on;                   %保留之前所画图，之后的画图时不清除以前画的图
plot(t,y-1,t,y2+1,t,y3); %上、下各平衡 1 个单位
```

上述代码运行后，可得到如图 5-2 所示的图形。

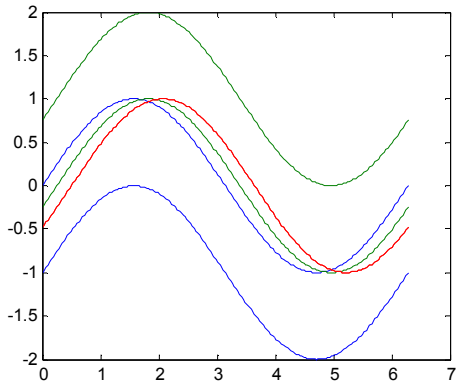


图 5-2 例 5-2 程序的运行结果

例 5-3 一组椭圆的绘制：$\frac{x^2}{a^2}+\frac{y^2}{25-a^2}=1$。

```
th = [0:pi/50:2*pi]';          % 参数值数组 101×1
a = [0.5:.5:4.5];              % 轴长数组 1×9
X = cos(th)*a;                 % 水平坐标值数组 101×9
Y = sin(th)*sqrt(25-a.^2);     % 纵坐标值数组 101×9
plot(X,Y)                      % 绘图，以 X,Y 中对应的列分别的曲线的横、
                               % 纵坐标绘曲线。因 X,Y 共 9 列，所以共绘
                               % 制 9 条曲线。
```

上述代码运行后，可得到如图 5-3 所示的图形。

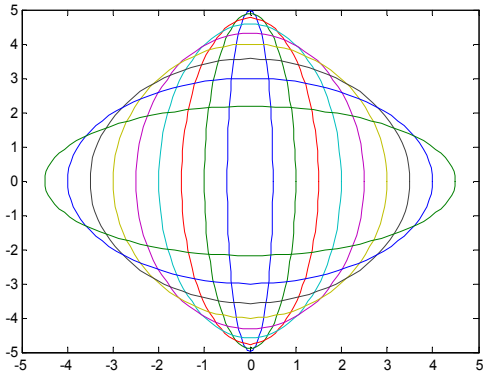


图 5-3 例 5-3 程序的运行结果

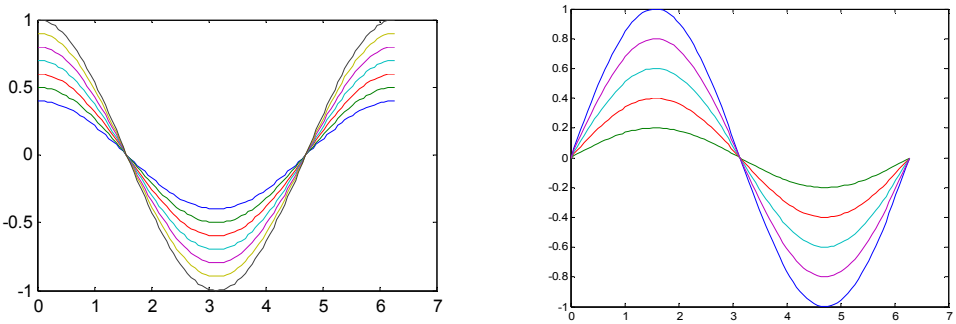
例 5-4 一组余弦曲线的绘制。

```
t=(0:pi/50:2*pi)';          %自变量数据准备 101×1，注意“'”表示取转置
k=0.4:0.1:1;                %缩放系数数组 1×7
Y=cos(t)*k;                 %函数的相关数据 101×7
plot(t,Y)                   %绘图
```

上述代码运行后，可得到如图 5-4(a)所示的图形。

注：相当于：plot(t,0.5*cos(t) ,t,0.6*cos(t) ,t,0.7*cos(t) ,t,0.8*cos(t) ,t,0.9*cos(t) ,t, cos(t))

如果将上例中的余弦换成正弦，如 y =sinx, y=0.2sinx, y = 0.4sinx, y = 0.6sinx, y = 0.8sinx，则可得到如图 5-4 (b)所示的图形。



(a) (b)

图 5-4 例 5-4 程序的运行结果

2、contour 绘图函数简介

contour 用于绘制三维曲面的二维等高线，并带有标注。它的常用的基本语法如下：

contour(X,Y,Z)

contour(X,Y,Z,n)

contour(X,Y,Z,v)

函数说明：

X,Y,Z 是同型矩阵（行列数相同），分别表示记录空间曲面网格点的,x,y,z 坐标矩阵，n 表示等高线的条数，是一个标量。v 是一个向量，其元素值表示高度值（即 z 坐标值），在 v 指定的 z 坐标高度画等高线。

当然还有其它语法形式，具体参看帮助文件。

由于空间曲面的描述需要网格点，而产生网格数据的函数是 meshgrid，具体看帮助系统，此处仅举例说明。

例 5-5 绘制曲面 $z = xe^{(-x^2-y^2)}$ 的等高线，其中 -2<x <2, -2<y<2。

```
[X,Y] = meshgrid(-2:.2:2,-2:.2:3);   % X、Y 分别是记录 x 和 y 坐标的矩阵
Z = X.*exp(-X.^2-Y.^2);              % Z 分别是记录 z 坐标的矩阵
[C,h] = contour(X,Y,Z);              % 绘制等高线，C 记录各等高线数据
                                     % 的矩阵，h 是等高线图形对象
colormap cool                        % 修改颜色风格
set(h,'ShowText','on');              % 显示条等高线的标注
```

上述代码可得到如图 5-5 所示的图形。注：可以在 Workspace 中查看变量 C 和 h。

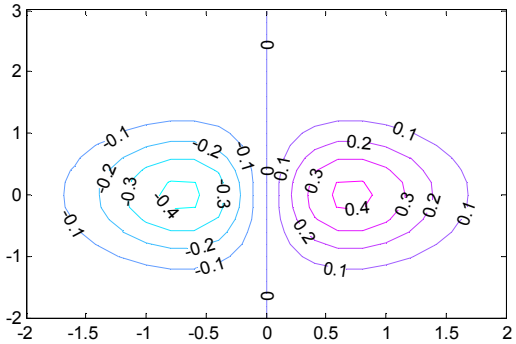


图 5-5 例 5-5 程序的运行结果

3、gradient 与 quiver 绘制二维梯度场

gradient 用于绘制二元函数的梯度场，其运算是基于离散网格数据点的，梯度使用差分来近似表示梯度的理论值。其常用的语法形式如下：

FX = gradient(F)

[FX,FY] = gradient(F)

[FX,FY,FZ,...] = gradient(F)

[...] = gradient(F,a,b)

函数说明：F 表示二元函数的函数值矩阵，即 Z = f(x,y)中的 z 坐标值的矩阵。FX 是水平方向的偏导数值矩阵，即水平方向的差分，FY 表示垂直方向的差分值矩阵。如是三元函数，则 FZ 表示 z 轴方向的差分值矩阵。a,b 用于指定计算差分时两个方向的步长。如果参数只两个，一个是 F，另一个是标量参数，则两个方向步长相同。

quiver 函数是根据 gradient 函数得到的水平和垂直梯度值来绘制梯度场，有时为了方便，将梯度场和等高线绘制在同一幅图中。quiver 的常用语法如下：

quiver(x,y,u,v)

h = quiver(...)

x,y,u 与 v 是同型矩阵，x 与 y 记录定义域中网格点的横坐标值与纵坐标值，u 与 v 记录网格点的水平与纵向梯度值，h 表示所绘梯度场对象。

例 5-6 绘制曲面 $z = xe^{(-x^2-y^2)}$ 的梯度场及带有等高线的梯度场，其中-2<x <2, -2<y<2。

```
x = -2:.2:2;                    % 定义域中 x 方向的定义域
y = -1:.2:1;                    % 定义域中 y 方向的定义域
[xx,yy] = meshgrid(x,y);        % 产生网格矩阵以记录网格点的 x 坐标和
                                %  y 坐标的值
zz = xx.*exp(-xx.^2-yy.^2);     % 记录网格点的 z 坐标
[px,py] = gradient(zz,.2,.2);   % 得到 x,y 方向梯度
quiver(x,y,px,py,2);    %绘制梯度场，第 5 个参数表示箭头的长度。
```

上述代码运行后，可得到如图 5-6 所示的图形。

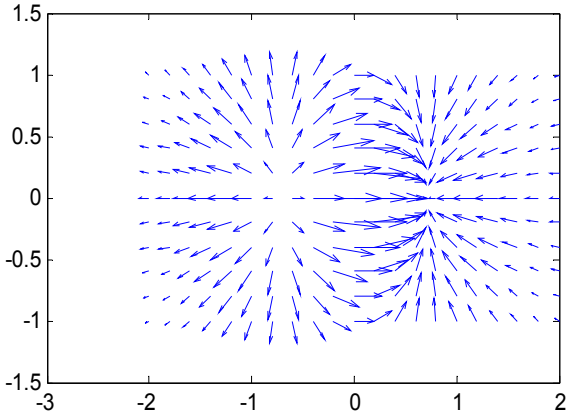


图 5-6 例 5-6 程序的运行结果

例 5-7 绘制一个带有等高线的梯度场。

```
v = -2:0.2:2;
[x,y] = meshgrid(v);          %二维定义域中的网格点坐标值的矩阵
z = x .* exp(-x.^2 - y.^2); %二维网格点上对应的 z 坐标，即高度坐标
[px,py] = gradient(z,.2,.2);%由计算梯度，是以差分来代替理论梯度值。
contour(v,v,z),              %绘制轮廓线
hold on,  %保留原图
quiver(v,v,px,py,2),         %绘制向量场。
hold off
```

上述代码运行后，可得到如图 5-7 所示的图形。

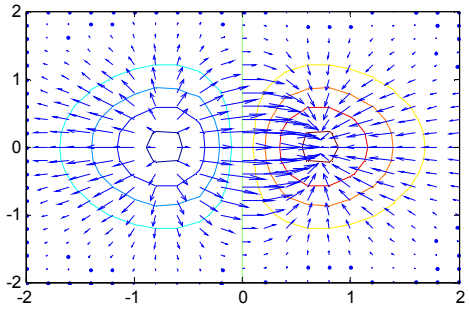


图 5-7 例 5-7 程序的运行结果

对于三维情形，可以使用 quiver3 函数来绘制三维的向量场。

二、三维图形的绘制

1、plot3 绘图绘制三维曲线

plot3 用于绘制三维函数曲线，功能和语法类似于二维函数曲线的绘制。其常用语法如下：

plot3(X1,Y1,Z1,...)

plot3(X1,Y1,Z1,LineSpec,...)

h = plot3(...)

函数说明：

X1、Y1、Z1 分别表示曲线的三个坐标数组，LineSpec 表示曲线属性。h 是表示曲线对象。

例 5-8 绘制螺旋曲线，其中 0<x <10*pi。

```
t = 0:pi/50:10*pi;
X=sin(t);                 % 曲线上各点的 x 坐标
Y=cos(t);                 % 曲线上各点的 y 坐标
Z=t;                      % 曲线上各点的 z 坐标
plot3(X,Y,Z);             % 绘制三维曲线，默认各点之间用线段连接
grid on;                  % 绘制网格线，以方便估计曲点上值
axis square               % 三个坐标方向按相同比例显示
```

上述代码运行后，可得到如图 5-8 所示的图形。

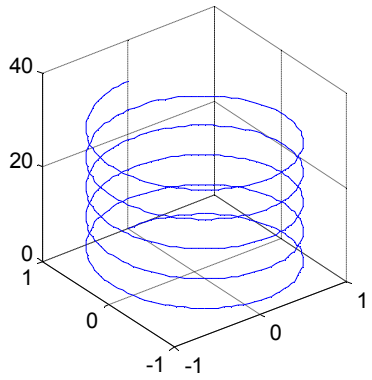


图 5-8 例 5-8 程序的运行结果

例 5-9 参矩阵为参数绘制空间曲线簇，其中 0<x <10*pi。

```
[X,Y] = meshgrid([-2:0.1:2]);     % x,y 都是 41×41 矩阵
Z = X.*exp(-X.^2-Y.^2);           %  z 是 41×41 矩阵
plot3(X,Y,Z)                      % 参数为矩阵，则依三个参数矩阵的对应
                                  % 的列绘制各条曲线
grid on
```

上述代码运行后，可得到如图 5-9 所示的图形。

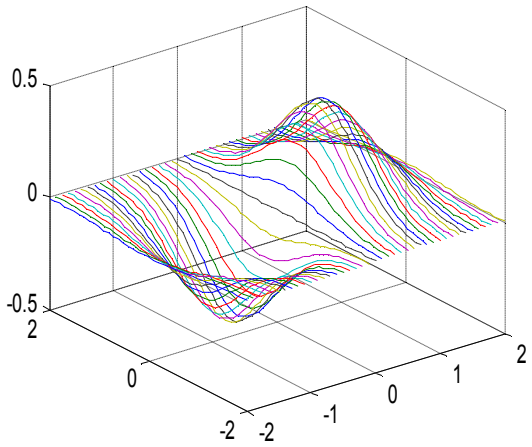


图 5-9 例 5-9 程序的运行结果

2、网格曲面绘制

mesh 函数用于绘制空间网格曲面。默认用 z 轴坐标值，即高度值用来控制颜色。它的常用语法如下：

mesh(X,Y,Z)

mesh(...,C)

meshc(...)

函数说明：

X,Y,Z 是网格点数据的三个坐标数据，C 表示控制颜色的矩阵，与 X,Y,Z 同型。meshc 在绘制网格曲面时，再在下方表面绘制等高线。

例 5-10 网格线曲面绘制演示，绘制如图 5-10 所示的图形。

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);                    % 绘制带有等高线的图
axis([-3 3 -3 3 -10 5])          % 三个坐标方向显示范围
```

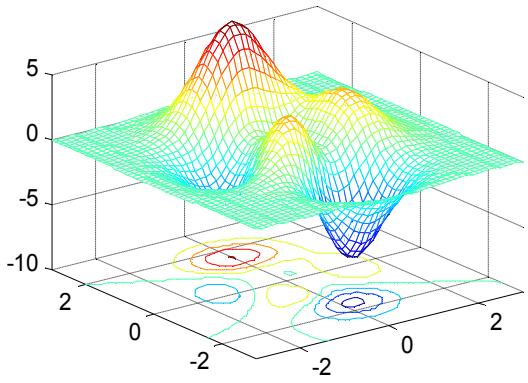


图 5-10 例 5-10 程序的运行结果

surf 函数用于绘制曲面，它的常用语法为：

surf(X,Y,Z)

surf(X,Y,Z,C)

surfc(...)

它的参数和 mesh 是类似的，不再重复。

例 5-11 曲面绘制演示，绘制如图 5-11 所示的图形。

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)                     % 采用渲染方法绘制曲面，同时绘制等高线
colormap hsv
axis([-3 3 -3 3 -10 5])
```

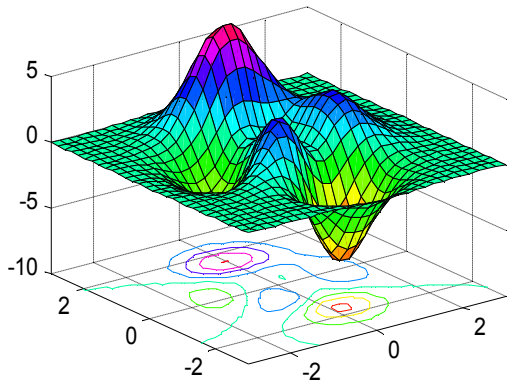


图 5-11 例 5-11 程序的运行结果

3、三维等高线绘制

三维等高线，二维等高线绘制一样，具体看实例演示或帮助文件。

contour3(X,Y,Z)

contour3(X,Y,Z,n)

contour3(X,Y,Z,v)

[C,h] = contour3(...)

例 5-12 三维等高线绘制实例演示。

```
[X,Y,Z] = peaks;
[C,h] =contour3(X,Y,Z,10);     % 绘制三维等高线，颜色自动
set(h,'LineWidth',2);          % 设置线宽
title('Twenty Contours of the peaks Function');
```

上述代码运行后，可得到如图 5-12 所示的图形。如果再加上“clabel(C,h);”，则会产生各条线的高度标准。

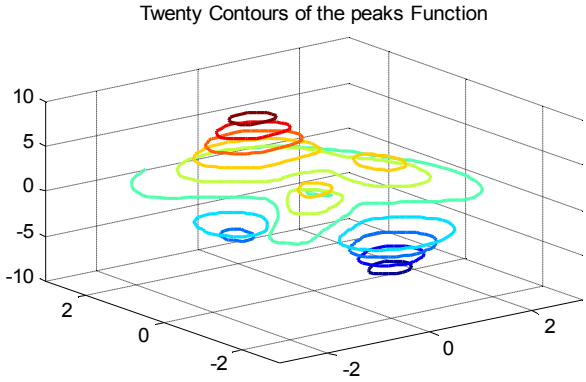


图 5-12 例 5-12 程序的运行结果

三、简捷绘图指令

针对上述的各种绘图函数，MATLAB 给出了简捷的绘图指令，只要以字符串的形式给出函数表达式，即可绘制出函数图形，也可适当设定其它参数，如定义域、颜色等。函数名类似，只不过是字母“e”打头，具体请参看帮助，常用简捷绘图指令有 ezcontour 、ezcontourf 、ezmesh 、ezmeshc 、ezplot 、ezplot3、ezpolar、ezsurf、ezsurfc 和 fplot。

例 5-13 在圆域上绘制 $z = xy$ 的图形，运行结果如图 5-13 所示。

```
ezsurf('x*y','circ');     % 在指定的圆形定义域范围内画出函图形
shading flat;             % 网格线和网格面采用相同的的颜色映射方法
view([-18,28])            % 设定视角，也可以在绘图窗口中以三维方式拉动旋转
```

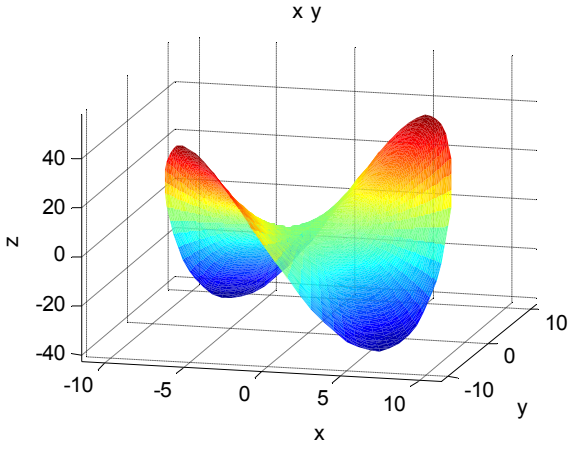


图 5-13 例 5-13 程序的运行结果

例 5-14 在圆域上绘制$z = x^2 + y^2$的图形，运行结果如图 5-14 所示。

```
ezsurf('x^2 + y^2','circ')      % 在指定的圆形定义域范围内画出函图形
shading flat;                   % 网格线和网格面采用相同的的颜色映射方法
```

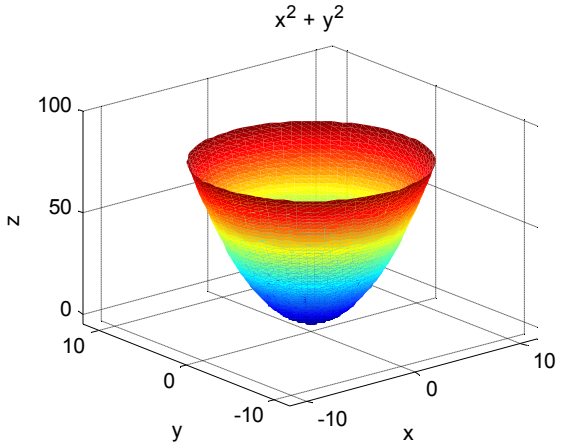


图 5-14 例 5-14 程序的运行结果

第 6 部分 MATLAB 数字图像处理初步

MATLAB 环境中自带了一个图像处理工具箱，其功能非常强大，包含了绝大部分经典/主流和最新的图像处理算法。然而，在本学期《数字图像处理》的课程学习过程以及课程实验中，并不建议一上来就立即仅仅满足于调用各种图像处理算法对应的接口函数，最好能按照课程实验的要求，选择合适的图像处理算法，根据其原理尝试独自编写算法代码，以完成相应的实验内容。所以，本部分仅初步介绍一些常用的、与课程实验中涉及到的图像算法关联度不大、但却必不可少的工具箱函数。

注：可在课程实验中调用 MATLAB 图像处理工具箱中的函数，用于对比自己编写的函数的处理效果是否准确。

一、图像的读取、显示与保存（写入磁盘）

MATLAB 图像处理工具箱提供了图像的读取、显示和保存图像的功能函数，这些函数可以在本学期的课程实验中调用（因为本课程的重点在于理解并掌握各类基本的图像处理算法及其实现，暂不要求考虑图片数据的读取与显示问题），下面对这三种常用函数进行简要的介绍。

(1) 图片的读入

imread()函数用于读入各种图像文件，其常用的语法形式（其他调用的语法形式请查阅 MATLAB 的帮助文档）为：

```
img = imread('filename.fmt');
```

上述两条指令默认要求图片文件为当前 MATLAB 的工作目录；filename 为图片文件的文件名，参数 fmt 指定了该图片文件的扩展名。

例 6-1 读入当前工作目录中的图片文件“DIP3.JPG”，并观察工作空间的更新状态。

输入如下指令并执行：

```
img = imread('DIP3.JPG');
```

上述指令执行后，可以在工作空间中观察到新增了一个变量 img，如图 6-1 所示。

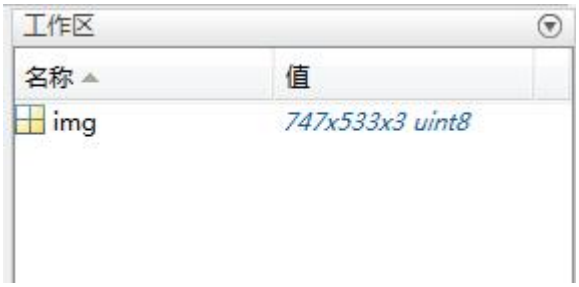


图 6-1 读取输入图片数据后的工作空间

可以看到，通过 imread()函数得到的返回结果 img，其数据类型是无符号整型（uint8），img 是一个三维数组，表示输入的是一副 3 通道图像，该图像的高度为 747 像素、宽度为 533 像素。

(2) 图片的显示

MATLAB 中有几个显示图片的函数，但最为常用的函数是 imshow()，其常用的语法形式（其他调用的语法形式请查阅 MATLAB 的帮助文档）有：

```
imshow(img);
imshow(img, level);
imshow(img, [low high]);
```

imshow(img, level)和 imshow(img, [low high])用于显示灰度图像，level 表示灰度级层数，若无该参数，则其默认值为 256；[low high]为显示图像的灰度级取值范围。

需要注意的是：当调用 imshow()函数显示图像时，最好保证变量 img 的数据类型为 uint8，如果 img 的数据类型是 double，可以在调用 imshow()前转换其数据类型，具体的转换方法将在下一小节介绍。当然，即使 img 为 double 类型，也是能够调用 imshow()函数，但这时要能正确显示图像内容需要对 img 的取值范围有特殊要求，这里要求大家自行设计相关的实验来进行验证分析和总结。

例 6-2 显示保存在工作空间中的图片数据 img（由例 6-1 读入）。

```
imshow(img);
```

上述指令执行后，显示效果如图 6-2 所示。

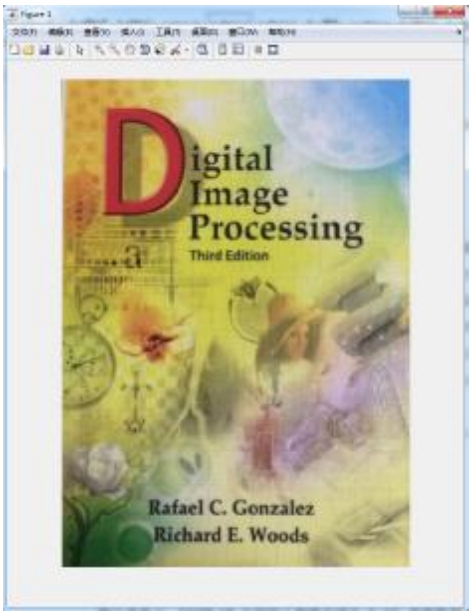


图 6-2 调用 imshow 函数显示工作空间中 img

(3) 图片的保存

假设经过某类算法处理完输入图像后，拟将新得到的图像 img2 保存到磁盘中时，可通过调用 imwrite()函数来实现，其常用的调用语法形式如下：

imwrite(img2,'filename.fmt','fmt');

其中“filename”为保存后图片的文件名，“fmt”为其扩展名。例如，考虑如下指令：

imwrite(img2,'newPicture.tif','tif);

该指令执行完毕后，系统会在当前 MATLAB 的工作目录下创建一个名为“newPicture”、扩展名为“tif””的图片文件。

二、图像和图像数据

如第一小节所述，通过 imread()函数读取的图像数据，其类型为 uint8，MATLAB 中用该类型存储的图像即为 8 位图，相对于默认条件下 double 类型来说，采用这种数据类型的好处是比较节省数据存储空间。值得指出的是，虽然调用 imread 函数所得的图像数据类型为 uint8，但是在图像算法处理过程中，各类运算使用的数据类型却是 double。这是因为直接对 uint8 类型的数据进行运算时可能会产生溢出（如 1 个字节的 uint8 类型最大只能存储的数据是 255），从而导致错误的运算结果；而另一方面，采用 double 类型的数据也是为了保证数据处理的精度。

所以，在调用 MATLAB 数字图像处理工具箱函数或者编写自定义图像处理函数时，必须注意被调函数所要求的输入参数类型，在必要时应将 double 与 uint8 两种类型之间

做一个转换。

先来看如下几行指令：

```
img = imread('DIP3.JPG');
img2 = double(img);
img3 = im2double(img);
img4 = double(img)/255;
```

上述四条指令，第一条读入图像数据，结果暂存于 uint8 类型的变量 img 中，为了保证后续运算的正确性，需要先将 img 的数据类型转换为 double 类型。后面三条指令都是实现 uint8 向 double 类型转换的功能，这里详细说明一下它们之间的异同。第二条指令中的 double()函数，即是简单地将 uint8 类型的数据 img 转换为 double 类型，但是数据的数值大小并没有变化，如原来的 img 数据范围是 0~255，那么经过 double()函数转化后，其数据范围仍是 0~255。第三条指令中的 im2double()函数，在将 uint8 类型的数据转换为 double 类型的同时，把数据范围由原来 img 的 0~255 映射到了 0~1 范围内，可以看成是数据的一种归一化操作。容易理解，第四条指令在实现效果上与第三条指令相同。至于采用哪一种转换方式，可根据自己数据处理的目的和习惯来定，两者的处理效果是等同的。

在 MATLAB 环境汇总处理完图像数据，显示该数据或者调用 imwrite()函数将其写入图片文件中的时候，需要注意一个问题：如果直接对 double 类型的数据矩阵 img5 执行指令“imshow(img5)”，会发现有时候显示的结果是一副全白的图像（这就是上文中要求大家去实验验证和分析的问题，请问你发现并理解为什么了吗？）。原因在于 imshow()函数在显示图像时对 double 类型的取值范围规定在 0～1 之间，即这时在数值上大于等于 1 的像素点都显示为白色；而 imshow()函数对 uint8 类型的取值范围则规定在 0～255 之间。因此，为了能正确显示一幅处理后的图像，应该做到：显示 double 类型的图像时，要么将图像数值归一化到 0～1 之间，要么将 double 类型的数据转换成 uint8 类型。解决上述问题的指令有：

```
inshow(uint8(img5));       % 方法 1：转换成 uint8 类型后在显示
imshow(img5/255);          % 方法 2：将 double 类型的图像矩阵数据调整到 0～1 之间
imshow(img5, []);          % 方法 3：“[]”表示自动调整数据的范围以便正确显示
```

上面第一条指令中的表达式“uint8(img5)”，其作用是将 double 类型的变量 img5 转换成 uint8 类型。