Matlab (I)

张晓平

武汉大学

2018 年 9 月 10 日

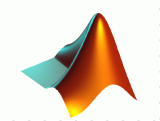
目录 I

目录 II

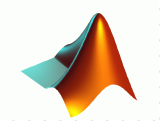
1. 运行 matlab

1.1 启动 matlab

启动 matlab

- 双击图标  .
- 进入命令窗口，可用来输入命令或显示纯文本结果。

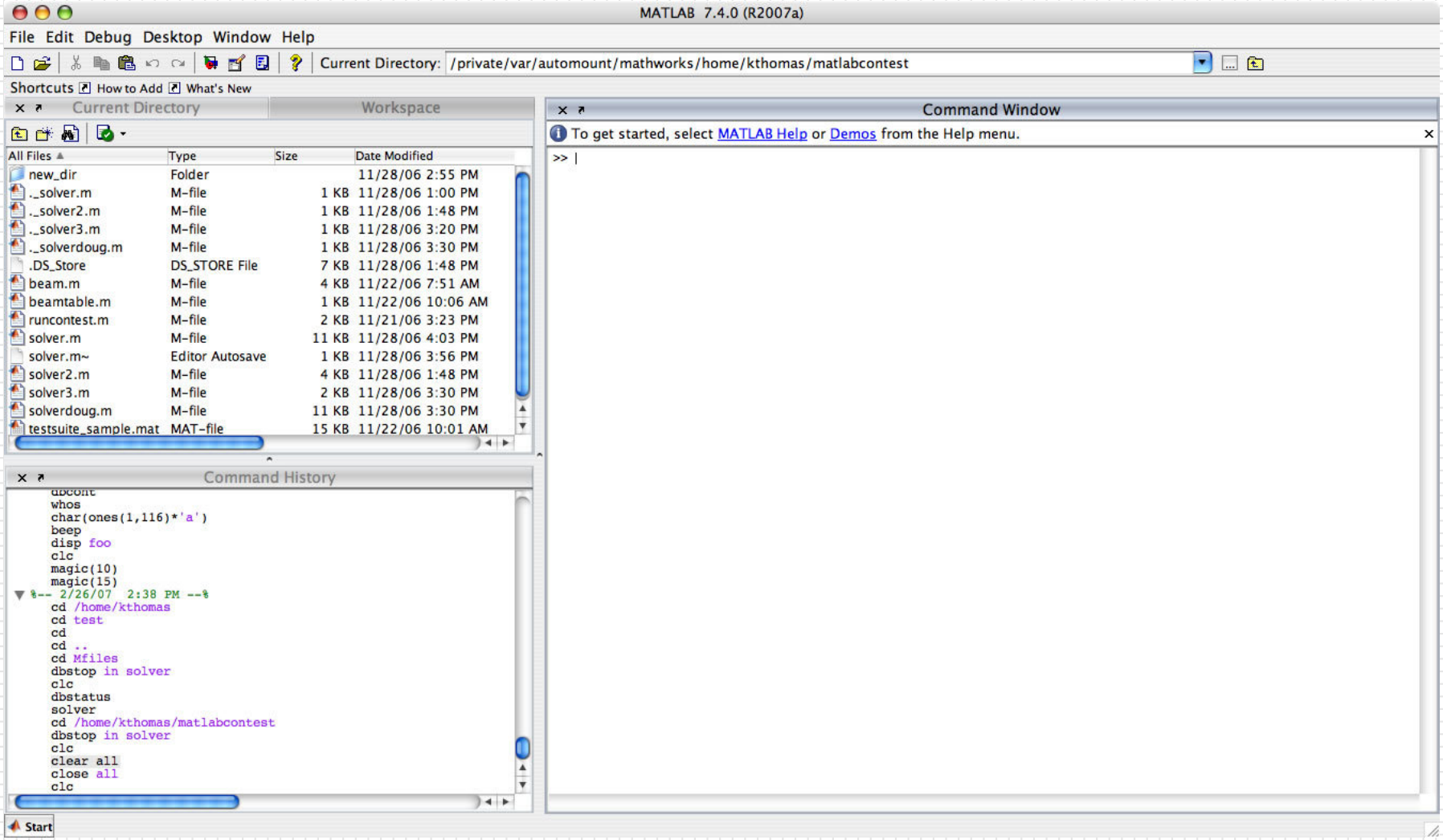
启动 matlab

- 双击图标  .
- 进入命令窗口，可用来输入命令或显示纯文本结果。

学习 matlab 的一个有效方法就是

边学边用

启动 matlab


MATLAB 7.4.0 (R2007a)
File Edit Debug Desktop Window Help
Current Directory: /private/var/automount/mathworks/home/kthomas/matlabcontest
Shortcuts How to Add What's New
Current Directory
Workspace
All Files Type Size Date Modified
new_dir Folder 11/28/06 2:55 PM
._solver.m M-file 1 KB 11/28/06 1:00 PM
._solver2.m M-file 1 KB 11/28/06 1:48 PM
._solver3.m M-file 1 KB 11/28/06 3:20 PM
._solverdoug.m M-file 1 KB 11/28/06 3:30 PM
.DS_Store DS_STORE File 7 KB 11/28/06 1:48 PM
beam.m M-file 4 KB 11/22/06 7:51 AM
beamtable.m M-file 1 KB 11/22/06 10:06 AM
runcontest.m M-file 2 KB 11/21/06 3:23 PM
solver.m M-file 11 KB 11/28/06 4:03 PM
solver.m~ Editor Autosave 1 KB 11/28/06 3:56 PM
solver2.m M-file 4 KB 11/28/06 1:48 PM
solver3.m M-file 2 KB 11/28/06 3:30 PM
solverdoug.m M-file 11 KB 11/28/06 3:30 PM
testsuite_sample.mat MAT-file 15 KB 11/22/06 10:01 AM
Command History
dbcont
whos
char(ones(1,116)*'a')
beep
disp foo
clc
magic(10)
magic(15)
%-- 2/26/07 2:38 PM --%
cd /home/kthomas
cd test
cd
cd ..
cd Mfiles
dbstop in solver
clc
dbstatus
solver
cd /home/kthomas/matlabcontest
dbstop in solver
clc
clear all
close all
clc
Command Window
To get started, select MATLAB Help or Demos from the Help menu.
>>
Start

matlab 桌面

- matlab 桌面给用户提供了多种方法来进行人机交互:
  - 在命令窗口中输入命令
  - 查看变量的值
  - 编辑函数与脚本
  - 图形的创建与标注
- 键入

```
playbackdemo('desktop')
```

可观看 matlab 桌面的动画演示。

1.2 matlab 用于表达式计算

matlab 充当计算器

```
>> 2 + 6 - 4
ans =
     4

>> ans/2
ans =
     2
```

ans为自动变量

赋值

```
>> a = 5
a =
     5
>> b = 6
b =
     6
>> c = b/a
c =
    1.2000
```

1.3 函数与变量

预定义变量和内置函数

```
>> pi
ans =
    3.1416
>> sin(ans/4)
ans =
    0.7071
```

函数的参数紧跟在函数名后的括号中。

预定义变量和内置函数

```
>> pi
ans =
    3.1416
>> sin(ans/4)
ans =
    0.7071
```

函数的参数紧跟在函数名后的括号中。注意：角度采用弧度制!

内置函数

```
>> log(256)
ans =
    5.5452
>> log10(256)
ans =
    2.4082
>> log2(256)
ans =
    8
```

在线帮助

- 使用在线帮助查询指定函数的信息：

```
>> help sqrt
```

在线帮助

- 使用在线帮助查询指定函数的信息：

```
>> help sqrt
```

- 使用 doc 函数可打开使用手册的联机版本，这对查询更复杂的命令非常有用：

```
>> doc plot
```

在线帮助

- 使用在线帮助查询指定函数的信息：

```
>> help sqrt
```

- 使用 doc 函数可打开使用手册的联机版本，这对查询更复杂的命令非常有用：

```
>> doc plot
```

- 使用 lookfor 可用来查找某个特殊主题的相关函数。当函数名未知时，用 lookfor 查找函数可知 matlab 是否提供所需操作的相关函数：

```
>> lookfor functionName
```

联机帮助

- 语法

```
>> help functionName
```

联机帮助

- 语法

```
>> help functionName
```

- 举例

```
>> help log
```

联机帮助

- 语法

```
>> help functionName
```

- 举例

```
>> help log
```

- 结果

```
LOG    Natural logarithm.
LOG(X) is the natural logarithm of the
elements of X.
Complex results are produced if X is not
positive.
See also log1p, log2, log10, exp, logm,
reallog.
...
```

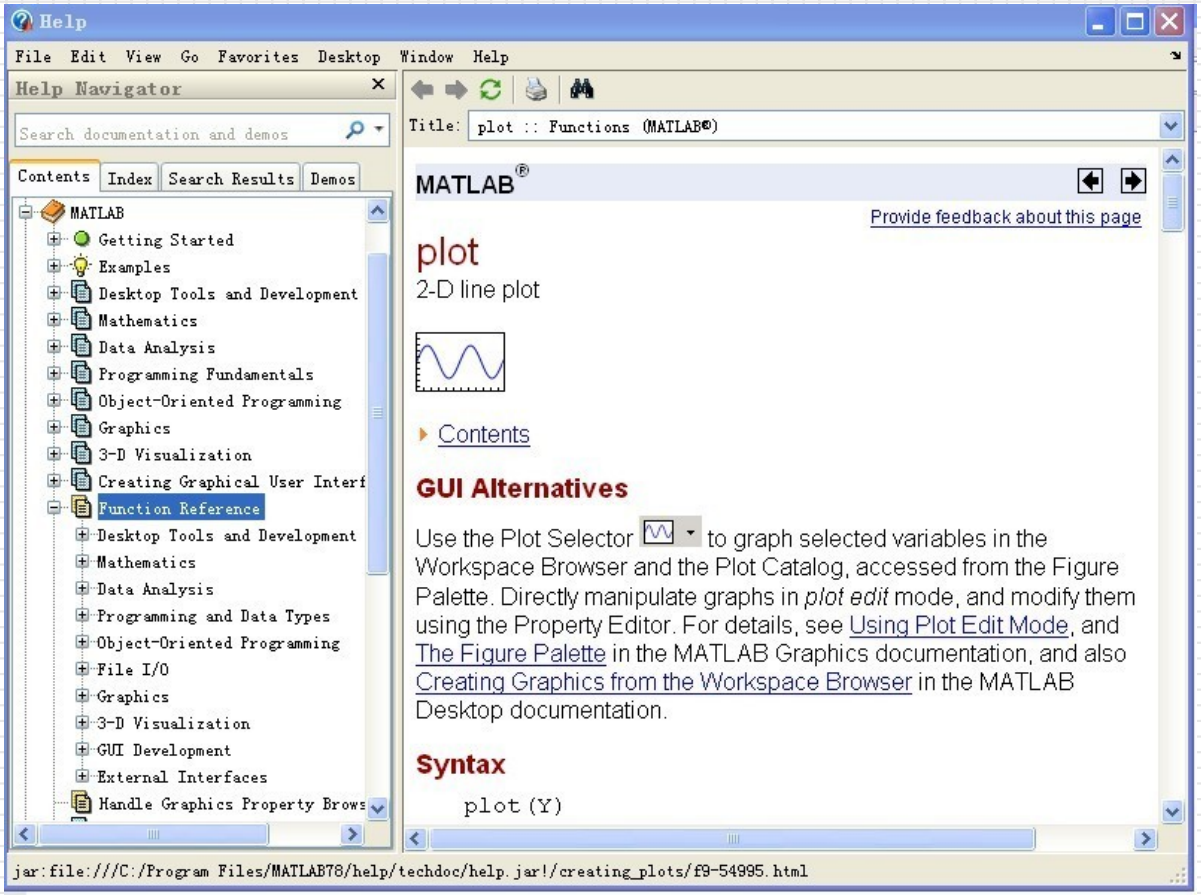
联机帮助

help函数提供了一份如何使用命令的紧凑说明，要得到某个命令更为详尽的信息，可以使用doc函数.

联机帮助

键入doc命令将打开help浏览器：

```
>> doc plot
```



查找函数

- 语法

```
>> lookfor string
```

查找函数

- 语法

```
>> lookfor string
```

- 举例

```
>> lookfor cosine
```

查找函数

- 语法

```
>> lookfor string
```

- 举例

```
>> lookfor cosine
```

- 结果

```
cos    - Cosine of argument in radians.
cosh   - Hyperbolic cosine.
cosd   - Cosine of argument in degrees.
acos   - Inverse cosine, result in radians.
acosd  - Inverse cosine, result in degrees.
...
```

利用分号禁止输出

给 x，y，z 赋值，但只输出 z 的值:

```
>> x = 5;
>> y = sqrt(59);
>> z = log(y) + x^0.25
z =
    3.5341
```

函数与变量

键入变量名，同时省略分号可以打印该变量的值

```
>> x = 5;
>> y = sqrt(59);
>> z = log(y) + x^0.25
z =
    3.5341
>> y
y =
    7.6811
```

函数与变量

一行输入多个表达式，用逗号或分号把每个表达式分开

```
>> a = 5; b = sin(a), c = cosh(a)
b =
    -0.9589
c =
    74.2099
```

函数与变量

一行输入多个表达式，用逗号或分号把每个表达式分开

```
>> a = 5; b = sin(a), c = cosh(a)
b =
    -0.9589
c =
    74.2099
```

分号限制输出，逗号允许输出。

matlab 变量

- 变量要先定义，再使用。
- 当一个变量未被定义时，在给它赋值时会被创建，并且其值可通过后面的赋值语句来改变:

```
>> t = 5;
>> t = t + 2
t =
     7
```

- 等号右边的任何变量必须在前面被定义过才能使用。

```
>> x = 2*z
??? Undefined function or variable 'z'.
```

matlab 变量

- 命名规则:
  - 以a-z或者A-Z开头
  - 其余字符只能为a-z，A-Z，0-9 或者下划线 _
  - 不能超过 31 个字符
  - 不能与内置变量，内置函数或者自定义函数重名
  - 区分大小写

matlab 变量

- 命名规则:
  - 以a-z或者A-Z开头
  - 其余字符只能为a-z，A-Z，0-9 或者下划线 _
  - 不能超过 31 个字符
  - 不能与内置变量，内置函数或者自定义函数重名
  - 区分大小写
- 举例

```
xxxxxxx,  pipeRadius, widgets_per_box
mysum, mySum
```

函数与变量

名称	含义
ans	当表达式的值未赋给某个变量时，系统自动将它赋给 ans
eps	浮点精度 (机器精度)
i, j	单位虚数
pi	π
realmax	最大正浮点数 (1.7977e+308)
realmin	最小正浮点数 (2.2251e-308)
Inf	∞, 比最大正浮点数realmax大，是1/0的值
NaN	不是一个数，如 0/0的值

函数与变量

- 规则:

  内置变量由内置函数使用，所以用户最好不要给内置变量重新赋值

函数与变量

- 规则:

  内置变量由内置函数使用，所以用户最好不要给内置变量重新赋值

- 例外:

  i 和 j 被预置为 $\sqrt{-1}$，但在矩阵下标中常用到它们。

2. 矩阵与向量

矩阵与向量

谨记 matlab 中的所有变量都是数组。

通过变量名访问数组，通过下标 (index) 访问数组中的元素。

矩阵与向量

谨记 matlab 中的所有变量都是数组。

通过变量名访问数组，通过下标 (index) 访问数组中的元素。

- 向量：只有一行或者只有一列的矩阵
- 标量：只有一行且只有一列的矩阵
- 字符串：字符型数组

2.1 创建矩阵和向量

创建 matlab 变量

matlab 变量通过赋值语句创建

```
>> x = expression
```

expression可以是:

- 手动输入
- 用数学表达式得到矩阵
- 使用内置函数，函数的返回值是矩阵
- 用户编写函数，函数的返回值是矩阵
- 从磁盘文件中导入矩阵数据

矩阵和向量的逐元创建

$$A = \begin{pmatrix} 3 & 2 \\ 3 & 1 \\ 1 & 4 \end{pmatrix}, x = \begin{pmatrix} 5 \\ 7 \\ 9 \\ 2 \end{pmatrix}, v = \begin{pmatrix} 9 & -3 & 4 & 1 \end{pmatrix}$$

矩阵和向量的逐元创建

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

矩阵和向量的逐元创建

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

```
>> x = [5; 7; 9; 2]
x =
     5
     7
     9
     2
>> v = [9 -3 4 1]
v =
     9    -3     4     1
```

矩阵和向量的逐元创建

手动输入向量时，各元素用方括号括起来。

矩阵和向量的逐元创建

手动输入向量时，各元素用方括号括起来。

创建行向量时，用空格分隔元素

```
>> v = [7 3 9]
v =
     7     3     9
```

矩阵和向量的逐元创建

手动输入向量时，各元素用方括号括起来。

创建行向量时，用空格分隔元素

```
>> v = [7 3 9]
v =
     7     3     9
```

创建列向量时，用分号断行

```
>> v = [2; 6; 1]
v =
     2
     6
     1
```

矩阵和向量的逐元创建

矩阵赋值时，行元素以空格分隔，列以分号分隔

```
>> A = [1 2 3; 5 7 11; 13 17 19]
A =
     1     2     3
     5     7    11
    13    17    19
```

转置算子

矩阵一旦创建，可通过其他运算符进行转换。

转置算子

矩阵一旦创建，可通过其他运算符进行转换。

转置操作符将行向量转化为列向量（反之亦然），将矩阵的行转化为列。

```
>> v = [2 4 1 6]
v =
     2     4     1     6
>> w = v'
w =
     2
     4
     1
     6
```

转置算子

变量被创建后，可被重新赋值。

```
>> x = 2;
>> x = x + 2
x =
     4
```

```
>> y = [1 2 3 4]
y =
     1     2     3     4
>> y = y'
y =
     1
     2
     3
     4
```

创建矩阵和向量

- 用内置函数创建向量:
  `linspace`和 `logspace`
- 用内置函数创建矩阵:
  `ones, zeros, eye, diag, ...`

注意: `ones`和`zeros`也能用于创建向量。

创建矩阵和向量

`linspace`函数创建一个行向量，其元素值之间的差值相同。

创建矩阵和向量

linspace函数创建一个行向量，其元素值之间的差值相同。

```
x = linspace(startValue, endValue)
                              (nelements = 100)
x = linspace(startValue, endValue, nelements)
```

创建矩阵和向量

linspace函数创建一个行向量，其元素值之间的差值相同。

```
x = linspace(startValue, endValue)
                              (nelements = 100)
x = linspace(startValue, endValue, nelements)
```

```
>> u = linspace(0.0, 0.25, 5)
u =
    0    0.0625    0.1250    0.1875    0.2500
>> u = linspace(0.0, 0.25);
```

创建矩阵和向量

linspace函数创建一个行向量，其元素值之间的差值相同。

```
x = linspace(startValue, endValue)
                                (nelements = 100)
x = linspace(startValue, endValue, nelements)
```

```
>> u = linspace(0.0, 0.25, 5)
u =
    0    0.0625    0.1250    0.1875    0.2500
>> u = linspace(0.0, 0.25);
```

谨记：语句以冒号结束会禁止输出。

用linspace创建向量

要创建列向量，只需加上转置操作符。

```
>> v = linspace(0, 9, 4)'
v =
     0
     3
     6
     9
```

举例：三角函数的使用

```
>> x = linspace(0, 2*pi, 6)';
>> y = sin(x);
>> z = cos(x);
>> [x y z]
ans =
         0         0    1.0000
    1.2566    0.9511    0.3090
    2.5133    0.5878   -0.8090
    3.7699   -0.5878   -0.8090
    5.0265   -0.9511    0.3090
    6.2832   -0.0000    1.0000
```

举例：三角函数的使用

- 表达式`y=sin(x)`和`z=cos(x)`使用向量化操作。

举例：三角函数的使用

- 表达式`y=sin(x)`和`z=cos(x)`使用向量化操作。
- 如果向量函数的输入为向量或矩阵，则输出通常为相同尺寸的向量或矩阵。

用logspace创建向量

同linspace类似，logspace函数创建一个行向量，其元素之间间距相等，且为对数步长。

用logspace创建向量

同linspace类似，logspace函数创建一个行向量，其元素之间间距相等，且为对数步长。

```
x = logspace(startValue, endValue)
x = logspace(startValue, endValue, nelements)
```

创建位于 $10^{\text{startValue}}$ 和 10^{endValue} 间的 nelements 个元素(缺省值为 100)。

用logspace创建向量

同linspace类似，logspace函数创建一个行向量，其元素之间间距相等，且为对数步长。

```
x = logspace(startValue, endValue)
x = logspace(startValue, endValue, nelements)
```

创建位于 $10^{\text{startValue}}$ 和 10^{endValue} 间的 nelements 个元素(缺省值为 100)。

```
>> u = logspace(1, 4, 4)
u =
    10    100    1000    10000
```

用函数创建矩阵

名称	含义
`diag`	创建含指定主对角元的矩阵，或提取矩阵的主对角元
`eye`	创建单位阵
`ones`	创建所有元素为 1 的矩阵
`rand`	创建元素为随机数的矩阵
`zeros`	创建所有元素皆为 0 的矩阵
`linspace`	创建等间距元素的行向量
`logspace`	创建对数间距的行向量

用函数创建矩阵

用`ones`和`zeros`设置矩阵或向量的初值。

用函数创建矩阵

用ones和zeros设置矩阵或向量的初值。

```
A = ones(nrows, ncols)
A = zeros(nrows, ncols)
```

用函数创建矩阵

用ones和zeros设置矩阵或向量的初值。

```
A = ones(nrows, ncols)
A = zeros(nrows, ncols)
```

```
>> D = ones(3, 3)
D =
     1     1     1
     1     1     1
     1     1     1
>> E = ones(2, 4)
E =
     1     1     1     1
     1     1     1     1
```

用函数创建矩阵

ones 和 zeros 也可用于创建向量。

```
v = ones(1, ncols)    创建元素均为 1 的列向量
v = ones(nrows, 1)    创建元素均为 1 的行向量
v = zeros(1, ncols)   创建元素均为 0 的列向量
v = zeros(nrows, 1)   创建元素均为 0 的行向量
```

用函数创建矩阵

```
>> s = ones(1, 4)
s =
     1     1     1     1
>> t = ones(3, 1)
t =
     0
     0
     0
```

用函数创建矩阵 (4)

eye 用于创建指定尺寸的单位阵，也可用于创建主对角元为 1 的非方阵。

```
A = eye(n)
A = eye(nrows, ncols)
```

用函数创建矩阵 (4)

eye 用于创建指定尺寸的单位阵，也可用于创建主对角元为 1 的非方阵。

```
A = eye(n)
A = eye(nrows, ncols)
```

```
>> C = eye(4)
C =
     1   0   0   0
     0   1   0   0
     0   0   1   0
     0   0   0   1
```

```
>> D = eye(3, 4)
     1   0   0   0
     0   1   0   0
     0   0   1   0
```

用函数创建矩阵

diag 既可用于创建指定主对角元的矩阵，也可用于提取矩阵的主对角元。

```
A = diag(v)
v = diag(A)
```

用函数创建矩阵

```
>> v= [1 2 3];
>> A = diag(v)
A =
     1     0     0
     0     2     0
     0     0     3
```

```
>> B = [1:4; 5:8; 9:12]
   B =
        1     2     3     4
        5     6     7     8
        9    10    11    12
>> w = diag(B)
    w =
        1
        6
       11
```

用函数创建矩阵

diag 函数的功能取决于输入的特征。

```
>> A = diag([3 2 1])
A =
     3     0     0
     0     2     0
     0     0     1
>> B = [4 2 2; 3 6 9; 1 1 7];
>> v = diag(B)
v =
     4
     6
     7
```

2.2 下标操作

下标操作

1、下标操作可用于访问矩阵元素。

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

下标操作

2、下标操作也可用于矩阵元素赋值或修改元素的值。

```
>> A(1, 1) = c/b;
A =
    0.2500    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    9.0000
```

下标操作

2、下标操作也可用于矩阵元素赋值或修改元素的值。

```
>> A(1, 1) = c/b;
A =
    0.2500    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    9.0000
```

超过矩阵维数的元素访问会导致错误。

```
>> A(1,4)
??? Attempted to access A(1,4); index out of
bounds because size(A)=[3,3].
```

下标操作

3、超过矩阵维数的元素赋值会调整矩阵的尺寸，换句话说，matlab 会自动在运行中调整矩阵尺寸。

```
>> A = [1 2; 3 4];
A =
     1     2
     3     4
>> A(3, 3) = 11
A =
     1     2     0
     3     4     0
     0     0    11
```

2.3 冒号操作

冒号操作

冒号操作对高效使用 matlab 非常重要。冒号既可用作操作符也可用作通配符。用冒号操作可

- 创建矩阵
- 访问或提取矩阵元素

冒号操作

语法:

```
vector = startValue:endValue
vector = startValue:increment:endValue
```

冒号操作

语法:

```
vector = startValue:endValue
vector = startValue:increment:endValue
```

注意: startValue, increment 和 endValue 不一定是整数。

冒号操作

1、创建行向量。

```
>> s = 1:4
s =
     1     2     3     4
>> t = 0:0.1:0.4
t =
         0    0.1000    0.2000    0.3000    0.4000
```

冒号操作

2、创建列向量。

```
>> u = (1:4)'
u =
     1
     2
     3
     4
>> v = 1:5'
v =
     1     2     3     4     5
```

冒号操作

2、创建列向量。

```
>> u = (1:4)'
u =
     1
     2
     3
     4
>> v = 1:5'
v =
     1     2     3     4     5
```

v 是行向量，因为转置操作符的作用范围为5，即5'还是5.

冒号操作

3、作为通配符访问整行或整列。

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

```
>> A(2, :)
ans =
     4     5     6
```

冒号操作

4、访问行或列的子集。

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

```
>> A(1:2, 2:3)
ans =
     2     3
     5     6
```

冒号操作

5、用作一些复杂操作的紧凑表达。

```
>> A = ones(8, 8);
>> A(3:6,3:6)=zeros(4,4)
A =
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
```

冒号操作

6、将任何向量转换成列向量。

```
>> x = 1:4;
>> y = x(:)
y =
     1
     2
     3
     4
```

冒号操作

7、逐列添加将矩阵转换成列向量。

```
>> A = rand(2, 2);
A =
    0.8147    0.1270
    0.9058    0.9134
>> v = A(:)
v =
    0.8147
    0.9058
    0.1270
    0.9134
```

3. 变量的其他类型

变量的其他类型

在 matlab 的计算中，经常用到数值型变量和字符型变量。数值型变量可以是实数或复数。字符型变量主要用在图形的标注或用户定义的函数名中。

以下讨论几种数值或者字符矩阵的简单变种：

- 复数
- 字符串
- 多项式

3.1 复数

复数

- 复数计算已经完全集成到 matlab 中，matlab 将所有变量都看成复数。
- 虚数单位被系统预定义为 i = j = $\sqrt{-1}$。

复数

```
>> sqrt(-4)
ans =
     0 + 2.0000i
>> x = 1 + 2*i
x =
     1.0000 + 2.0000i
>> y = 1 - 2*i
y =
     1.0000 - 2.0000i
>> z = x*y
z =
     5
```

虚数单位

i和j是缺省值为 $\sqrt{-1}$ 的 matlab 变量

```
>> i^2
ans =
    -1
```

虚数单位

i和j均可重新赋值

```
>> i = 5;
>> t = 8;
>> u = sqrt(i-t);
u =
    0 + 1.7321i
>> u * u
ans =
   -3.0000
```

虚数单位

- 给复数赋值时，常数与i或j做乘积时 * 号可用可不用。

```
>> x = 1 + 2 * i;
>> x = 1 + 2i;          %两种方式等价
```

虚数单位

- 给复数赋值时，常数与i或j做乘积时 * 号可用可不用。

```
>> x = 1 + 2 * i;
>> x = 1 + 2i;          %两种方式等价
```

- 只要单位虚数i和j处于表达式的尾部，可省略 * 号。

```
>> x = 1 + i * 2;
>> x = 1 + i2;          %错误
```

虚数单位

- 给复数赋值时，常数与i或j做乘积时 * 号可用可不用。

```
>> x = 1 + 2 * i;
>> x = 1 + 2i;          %两种方式等价
```

- 只要单位虚数i和j处于表达式的尾部，可省略 * 号。

```
>> x = 1 + i * 2;
>> x = 1 + i2;          %错误
```

- i或j不能和变量名一起用，否则会引起混乱：

```
>> w = 2;
>> x = 1 + wi
?? Undifined function or variable wi.
```

虚数单位

i和j通常用作数组下标

```
>> A = [1 2; 3 4];
>> i = 2;
>> A(i, i) = 1
A =
^^I1    2
^^I3    1
>> x = A(2, j)
??? Subscript indices must either be real
positive integers or logicals.
```

虚数单位

i和j通常用作数组下标

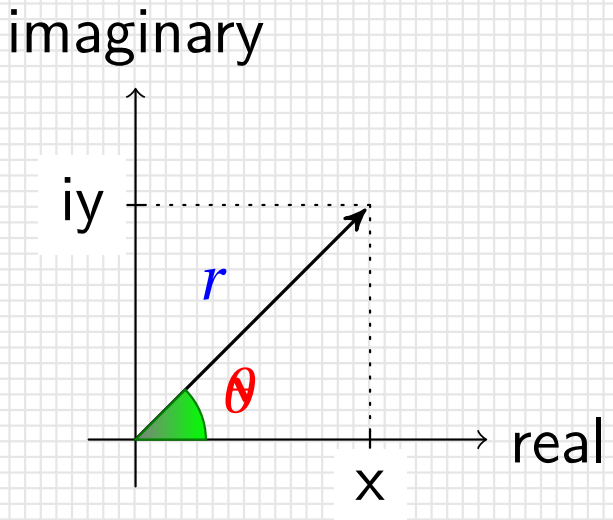
```
>> A = [1 2; 3 4];
>> i = 2;
>> A(i, i) = 1
A =
^^I1    2
^^I3    1
>> x = A(2, j)
??? Subscript indices must either be real
positive integers or logicals.
```

注意: 在进行复数运算时，最好保持i或j 的预赋值为 $\sqrt{-1}$。

复数

Euler 公式表示一个复数：

$$\begin{cases} z = \zeta e^{i\theta} \\ x = \mathrm{Re}(z) = |z|\cos(\theta) = r\cos(\theta) \\ x = i\mathrm{Im}(z) = i|z|\sin(\theta) = ir\sin(\theta) \end{cases}$$



复数

名称	含义
`abs`	复数的模
	`abs(z)` 等价于 `sqrt(real(z)^2 + imag(z)^2)`
`angle`	复数的辐角
`exp`	若 x 为实数，则 $\exp(x) = e^x$
	若 z 为实数，则 $\exp(z) = e^{\mathrm{Re}(z)}(\cos \mathrm{Im}(z) + i \sin \mathrm{Im}(z))$
`conj`	复数的共轭
`imag`	复数的虚部
`real`	复数的实部

复数

```
>> zeta = 5;
>> theta = pi/3;
>> z = zeta * exp(i*theta)
z =
    2.5000 + 4.3301i
>> abs(z)
ans =
    5
>> sqrt(z*conj(z))
ans =
    5
```

复数

```
>> zeta = 5;
>> theta = pi/3;
>> z = zeta * exp(i*theta)
z =
    2.5000 + 4.3301i
>> abs(z)
ans =
     5
>> sqrt(z*conj(z))
ans =
     5
```

```
>> x = real(z)
x =
    2.5000
>> y = imag(z)
ans =
    4.3301
>> angle(z)*180/pi
ans =
   60.0000
```

复数

```
>> zeta = 5;
>> theta = pi/3;
>> z = zeta * exp(i*theta)
z =
    2.5000 + 4.3301i
>> abs(z)
ans =
     5
>> sqrt(z*conj(z))
ans =
     5
```

```
>> x = real(z)
x =
    2.5000
>> y = imag(z)
ans =
    4.3301
>> angle(z)*180/pi
ans =
   60.0000
```

谨记：所有角度均采用弧度制。

3.2 字符串

字符串

- 字符串是元素为字符的矩阵
- 字符串常量用单引号表示
- 冒号操作和下标操作仍适用

字符串

```
>> first = 'Black';
>> last  = 'Smith';
>> name  = [first, ' ', last]
name =
Black Smith
>> length(name)
ans =
    11
>> name(8:11)
ans =
mith
```

字符串操作

名称	含义
`char`	利用 ASCII 码将整数转换为字符，或字符矩阵
`findstr`	从一个字符串中寻找指定字符串
`length`	返回字符串的字符个数
`num2str`	将数字转换成字符串
`str2num`	将字符串转换成数字
`strcmp`	比较两个字符串
`strncmp`	比较两个字符串的前 n 个元素
`sprintf`	用作格式化输出

字符串操作

num2str：将数字转换成字符串

```
>> msg1 = ['There␣are␣', num2str(100/2.54), ...
    '␣inches␣in␣a␣meter']
msg1 =
There are 39.3701 inches in a meter
```

字符串操作

num2str: 将数字转换成字符串

```
>> msg1 = ['There␣are␣', num2str(100/2.54), ...
    '␣inches␣in␣a␣meter']
msg1 =
There are 39.3701 inches in a meter
```

sprintf: 用作格式化输出

```
>> msg2 = sprintf('There␣are␣%5.2f␣cubic␣inches␣
in␣a␣liter', 1000/2.54^3)
msg2 =
There are 61.02 cubic inches in a liter
```

字符串操作

`char`：可用于连接字符串

```
>> both = char(msg1, msg2)
both =
There are 39.3701 inches in a meter
There are 61.02 cubic inches in a liter
```

字符串操作

`char`：也可将 ASCII 码转换成字符

```
>> char(49)
ans =
1
>> char([77 65 84 76 65 66])
ans =
matlab
```

字符串操作

strcmp：判断两个字符串是否相等

```
>> msg1 = 'Hello C!';
>> msg2 = 'Hello matlab!';
>> strcmp(msg1, msg2)
ans =
     0
```

字符串操作

strncmp：判断两个字符串的前 n 个字符是否相等

```
>> strncmp(msg1, msg2, 6)
ans =
     1
>> strncmp(msg1, msg2, 7)
ans =
     0
```

字符串操作

findstr：返回指定字符串在另一个字符串中的出现位置

```
>> findstr('l', msg1)
ans =
     3     4
>> msg1(3:4)
ans =
ll
```

3.3 多项式

多项式

matlab 多项式以系数向量的方式存储。多项式系数以 x 的降幂形式存储：

$$P_n(x) = c_1 x^n + c_2 x^{n-1} + \cdots + c_n x + c_{n+1}.$$

多项式

例　计算 $x^3-2x+12$ 在 $x=1.5$ 的值

多项式

例　计算 $x^3-2x+12$ 在 $x=1.5$ 的值

将多项式系数存储在向量 c 中

```
>> c = [1 0 -2 12];
```

多项式

例　计算 $x^3-2x+12$ 在 $x=1.5$ 的值

将多项式系数存储在向量 c 中

```
>> c = [1 0 -2 12];
```

使用内置函数polyval计算多项式的值

```
>> polyval(c, 1.5)
ans =
    12.3750
```

多项式

名称	含义
conv	两个多项式的乘积
deconv	两个多项式的商
poly	创建指定根的多项式
polyder	对多项式求导数
polyval	计算多项式的值
polyfit	多项式拟合
roots	求多项式的根

4. 矩阵向量操作

matlab 中的线性代数

matlab 是matrix laboratory 的缩写。

matlab 的数据结构和语法使得线性代数的标准运算，如矩阵向量的加、减和乘积变得非常容易。

这里我们介绍几种常见的运算：

- 向量的加减
- 内积和外积
- 向量化
- 数组操作

4.1 基本的矩阵向量操作

向量的加减

向量的加减法是逐元运算。

```
>> u = [10 9 8];
>> v = [1 2 3];
>> u + v
ans =
    11    11    11

>> u - v
ans =
     9     7     8
```

向量的内积和外积

向量的内积使两个向量构成一个标量：

$$\sigma = \boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}\boldsymbol{v}^T \Longleftrightarrow \sigma = \sum u_i v_i$$

向量的外积使两个向量构成一个矩阵：

$$A = \boldsymbol{u}^T \boldsymbol{v} \Longleftrightarrow a_{i,j} = \sum u_i v_j$$

向量的内积和外积

```
>> u = [10 9 8];
>> v = [1 2 3];
>> u * v'
ans =
52

>> u' * v
ans =
    10    20    30
     9    18    27
     8    16    24
```

4.2 向量化

向量化

- 向量化，即使用简单、紧凑的表达式，而不使用显示的循环，来对向量的元素进行操作。
- 向量化使得计算得以简洁地表示，使程序员可以专注于正在执行的操作。
- 向量化对高效使用 `matlab` 非常重要。

使用内置函数向量化

- 大部分内置函数支持向量化操作。
- 如果输入是标量，则结果也为标量。
- 如果输入是向量或者矩阵，则输出为与输入相同尺寸的向量或矩阵。

使用内置函数向量化

```
>> x = 0:pi/4:pi
x =
    0    0.7854    1.5708    2.3562    3.1416
>> y = cos(x)
y =
    1.0000    0.7071    0.0000   -0.7071
   -1.0000
```

与 C 的不同

matlab 语句

```
x = 0:pi/4:pi;
y = cos(x);
```

与 C 的不同

matlab 语句

```
x = 0:pi/4:pi;
y = cos(x);
```

C 语句

```
double x[5], y[5];
double pi = 3.1515926;
double dx = pi/4.d0;
int i;
for (i = 0; i < 5; i++)
{
    x[i] = i * dx;
    y[i] = sin(x[i]);
}
```

与 C 的不同

matlab 语句

```
x = 0:pi/4:pi;
y = cos(x);
```

C 语句

```
double x[5], y[5];
double pi = 3.1515926;
double dx = pi/4.d0;
int i;
for (i = 0; i < 5; i++)
{
    x[i] = i * dx;
    y[i] = sin(x[i]);
}
```

matlab 语句中没有出现显式的循环。

向量化

```
>> A = pi * [1 2; 3 4]
A =
    3.1416    6.2832
    9.4248   12.5664
>> S = sin(A)
S =
1.0e-15 *
    0.1225   -0.2449
    0.3674   -0.4899
```

向量化

```
>> A = pi * [1 2; 3 4]
A =
    3.1416    6.2832
    9.4248   12.5664
>> S = sin(A)
S =
1.0e-15 *
    0.1225   -0.2449
    0.3674   -0.4899
```

```
>> B = A/2
B =
    1.5708    3.1416
    4.7124    6.2832
>> T = sin(B)
T =
    1.0000    0.0000
   -1.0000
   -0.0000
```

****

4.3 数组运算符

数组运算符

- 数组运算符支持逐元操作，这与线性代数中的运算规则不一样；
- 数组运算符与标准运算相比，前面多了一个点；
- 数组运算符是编写向量化程序的一个重要工具。

数组运算符

- 数组运算符支持逐元操作，这与线性代数中的运算规则不一样；
- 数组运算符与标准运算相比，前面多了一个点；
- 数组运算符是编写向量化程序的一个重要工具。

名称	含义
.*	逐元乘积
./	逐元“右”除
.\	逐元“左”除
.^	逐元求幂

逐元乘法和除法

```
>> u = [1 2 3];
>> v = [4 5 6];
>> w = u.*v
w =
     4    10    18

>> x= u./v
x =
    0.2500    0.4000    0.5000
```

逐元乘法和除法

```
>> u = [1 2 3];
>> v = [4 5 6];
>> y = sin(pi*u/2) .* cos(pi*v/2)
y =
    1.0000    0.0000    1.0000

>> z = sin(pi*u/2) ./ cos(pi*v/2)
z =
    1.0000    0.4000    1.0000
```

逐元乘法和除法应用于矩阵

```
>> A = [1 2 3 4; 5 6 7 8];
>> B = [8 7 6 5; 4 3 2 1];
>> A .* B
ans =
     8    14    18    20
    20    18    14     8
>> A * B
??? Error using ==> * ...
>> A * B'
ans =
    60    20
   164    60
```

用于矩阵求幂

```
>> A = [1 2 3 4; 5 6 7 8];
>> A.^2
ans =
     1     4     9    16
    25    36    49    64
```

5. matlab 的 workspace

matlab 的 workspace

所有变量都存在于 matlab 的 workspace。熟悉了 workspace 的概念，你就可以

- 创建、赋值、删除变量
- 从外部文件中载入数据
- 操作 matlab 的路径

matlab 的 workspace

- clear：用于删除 workspace 中的所有变量。
- who：用于显示 workspace 中的变量名。

```
>> clear
>> who
>> a = 5; b = 2; c = 1;
>> d(1) = sqrt(b^2 - 4*a*c);
>> d(2) = -d(1);
>> who
Your variables are:
    a    b    c    d
```

matlab 的 workspace

whos：显示 workspace 中每个变量的名称、大小、内存分配以及变量类型。

```
>> whos
Name        Size    Bytes  Class     Attributes

a           1x1         8  double
b           1x1         8  double
c           1x1         8  double
d           1x2        32  double    complex
```

matlab 的 workspace

内置变量的类型有double, char, sparse, struct, cell。一个变量的类型决定了数据的存储类型，我们只需要用到double型的数值数据以及char型的字符串数据。

外部文件

将数据写入文件

```
save fileName
save fileName variable1 variable2 ...
save fileName variable1 variable2 ... -ascii
```

读数据保存在矩阵中

```
load fileName
load fileName matrixVariable
```

从外部文件中载入数据

例：从一个文件中读取数据，然后画图。

```
>> load file.dat
>> xdata = file(:, 1);
>> ydata = file(:, 2);
>> plot(xdata, ydata)
```

从外部文件中载入数据

matlab 只能使用其路径下的函数和数据文件。

- 要将D:\matlab\code\加入路径，可键入

```
>> p = path;
>> path(p, 'D:\matlab\code\');
```

- 路径的表示方法与操作系统有关，如Unix/Linux应写成

```
>> p = path;
>> path(p, '~/matlab/code/');
```

6. 绘图

绘图

- 简单绘图
- 坐标轴标度及注释
- 2d 和 3d 绘图

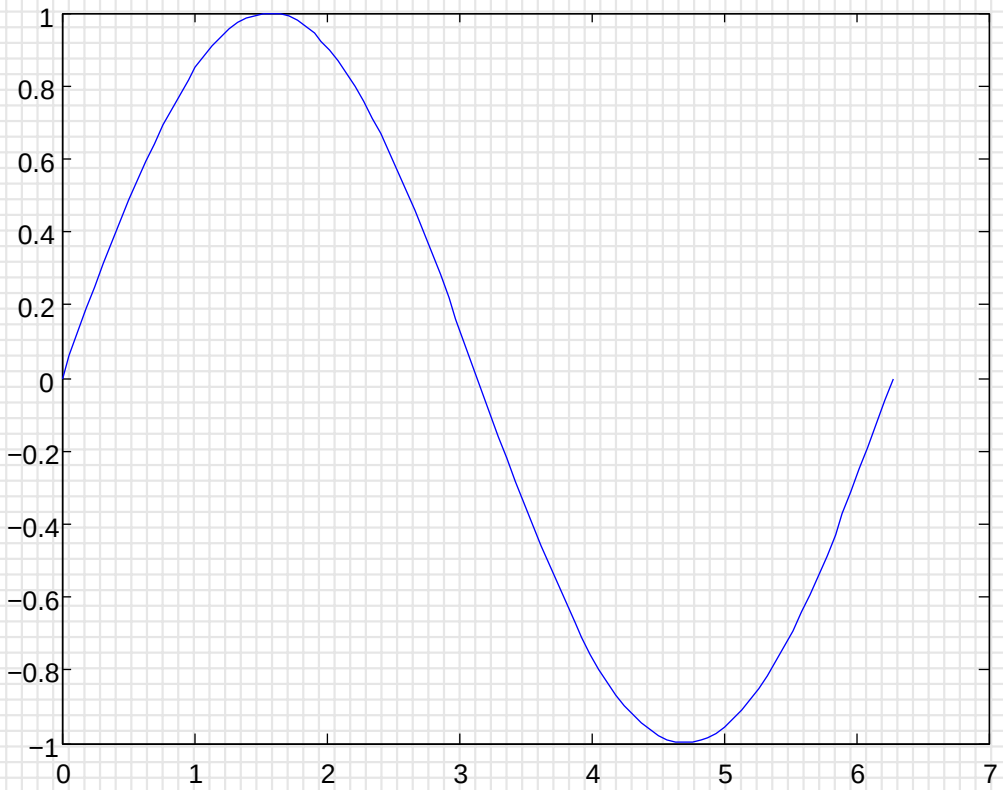
简单绘图

```
plot(x, y)
plot(xdata, ydata, symbol)
plot(x1, y1, x2, y2, ...)
plot(x1, y1, symbol1, x2, y2, symbol2, ...)
```

注意：x 和 y 必须有相同的尺寸，x1 和 y1 必须有相同的尺寸，x2 和 y2 必须有相同的尺寸，...

简单绘图

```
>> x = linspace(0, 2*pi);
>> y = sin(x);
>> plot(x, y);
```



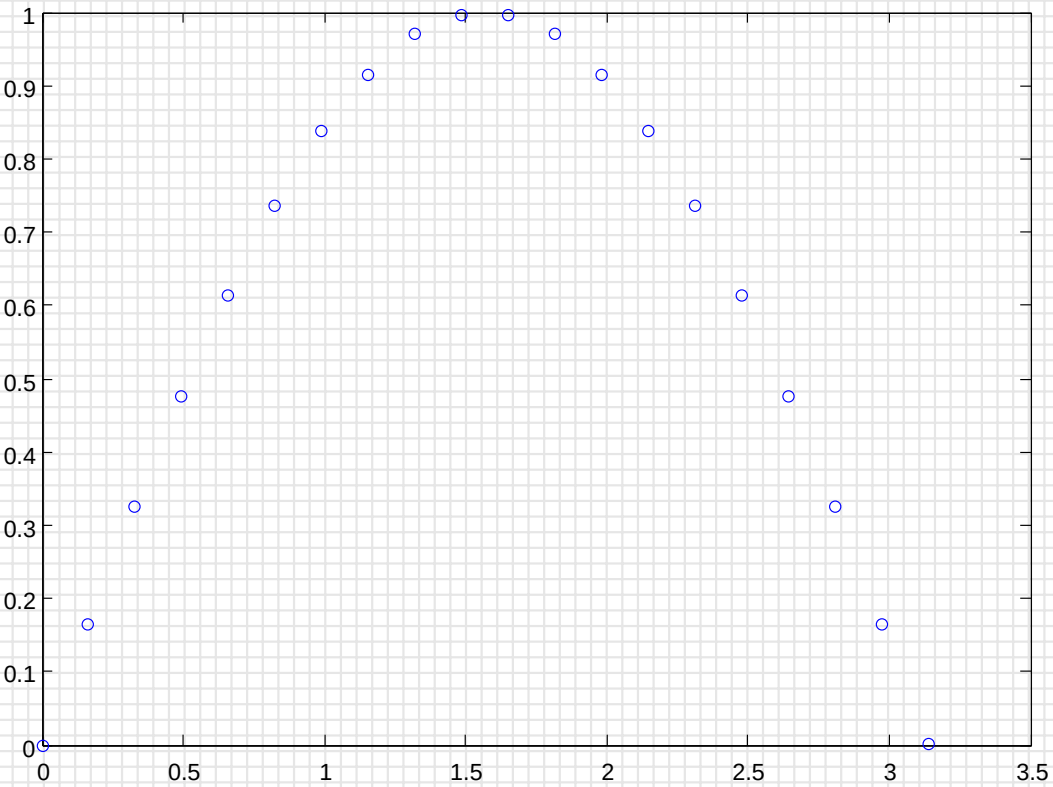
线条与符号的类型

颜色		符号				线条	
y	yellow 黄	.	point	^	triangle(up)	-	solid
m	magenta 紫红	o	circle	<	triangle(left)	:	dotted
c	cyan 蓝绿	x	x-mark	>	triangle(right)	-.	dashdot
r	red 红	+	plus	p	pentagram 五角星	–	dahsed
g	green 绿	*	star	h	hexagram 六角星		
b	blue 蓝	s	square				
w	white 白	d	diamond				
k	black 黑	v	triangle(down)				

要选择颜色/符号/线条类型，可从每一列中选一个元素。

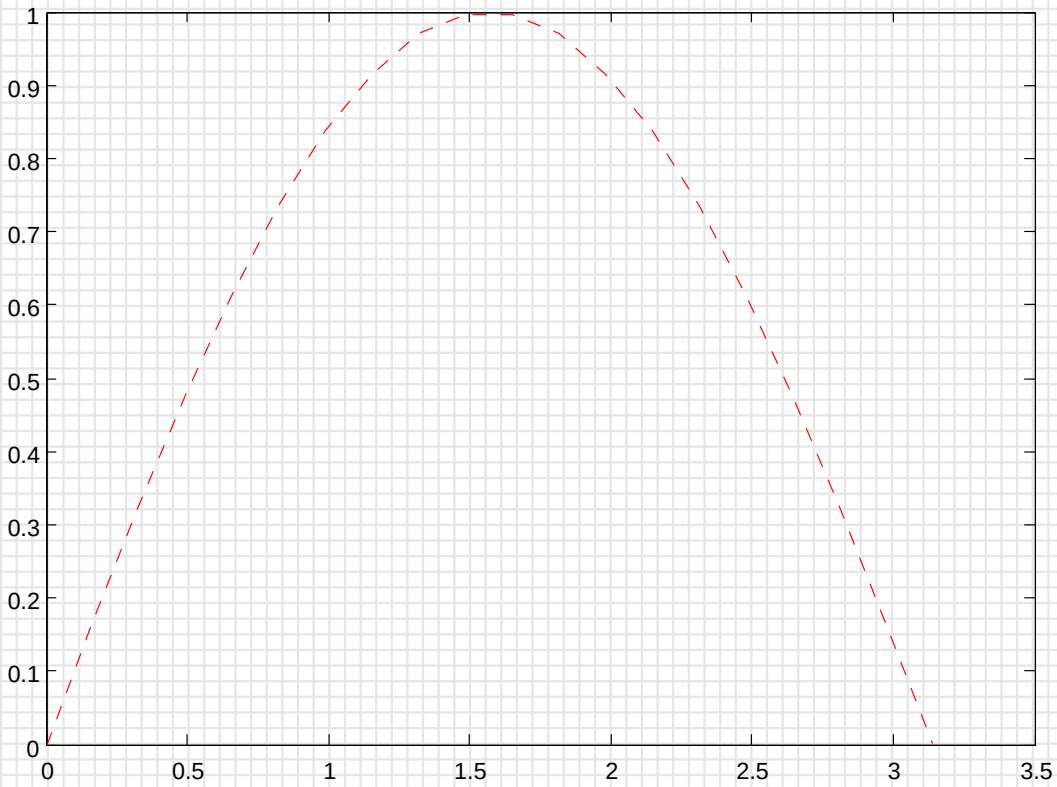
线条与符号的类型

```
>> x = linspace(0, pi, 10);
>> y = sin(x);
>> plot(x, y, 'bo');
```



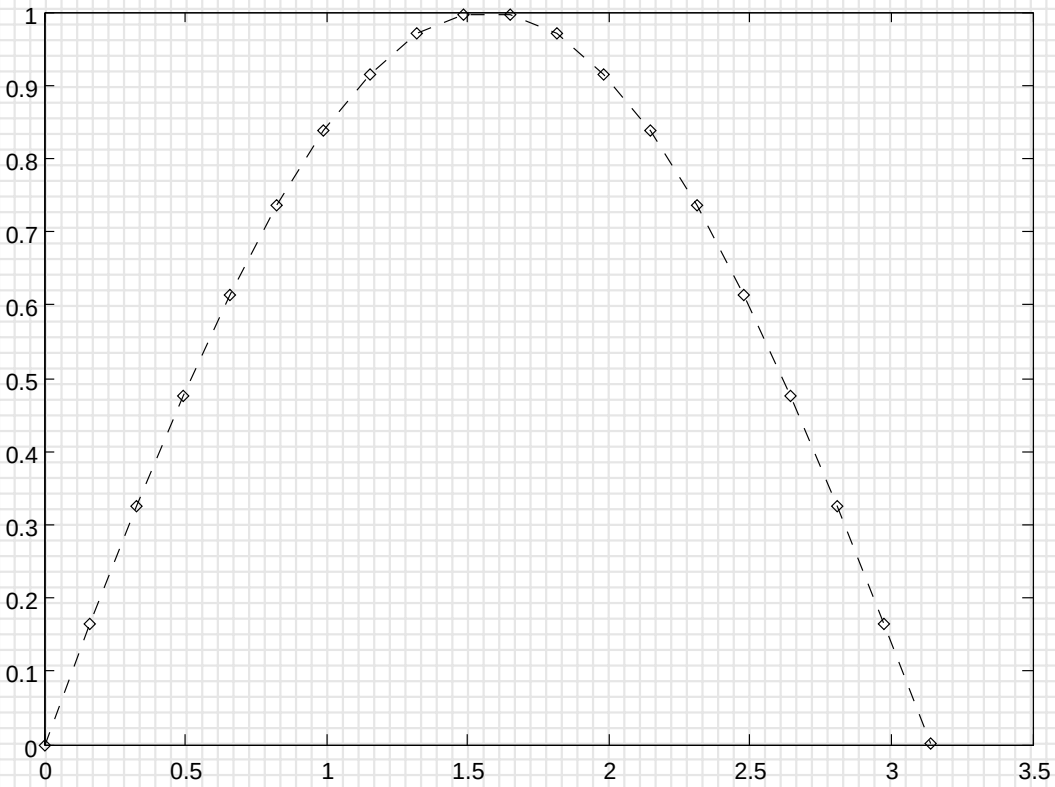
线条与符号的类型

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



线条与符号的类型

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



可选的轴标注

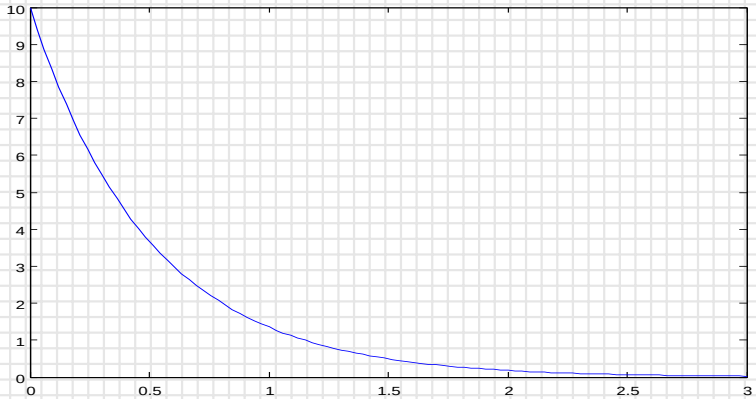
- 在很多问题中，通过对数据进行对数转换可以更清晰地看出数据的某些特征，在对数坐标系中描绘数据点的曲线，可以直接地表现对数转换。
- 对数转换有双对数坐标转换和单轴对数坐标转换两种。
- 用`loglog`函数可以实现双对数坐标转换，用`semilogx`和`semilogy`函数可以实现单轴对数坐标转换。

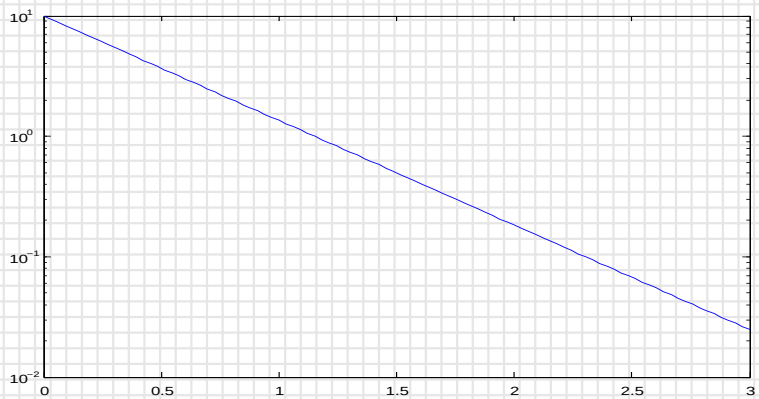
可选的轴标注

名称	轴标注
`loglog`	表示 x、y 坐标都是对数坐标系
`semilogx`	表示 x 坐标轴是对数坐标系
`semilogy`	表示 y 坐标轴是对数坐标系
`plotyy`	有两个坐标轴，一个在左边，一个在右边

可选的轴标注

```
>> x = linspace(0, 3);
>> y = 10*exp(-2*x);
>> plot(x, y);
>> semilogy(x, y);
```





subplot

- 功能：用于分割 figure，创建子坐标系
- 语法：

```
subplot(nrows, ncols, thisplot)
```

- 描述：将 figure 换份成 $m \times n$ 块，在第thisplot块中创建坐标系。

subplot

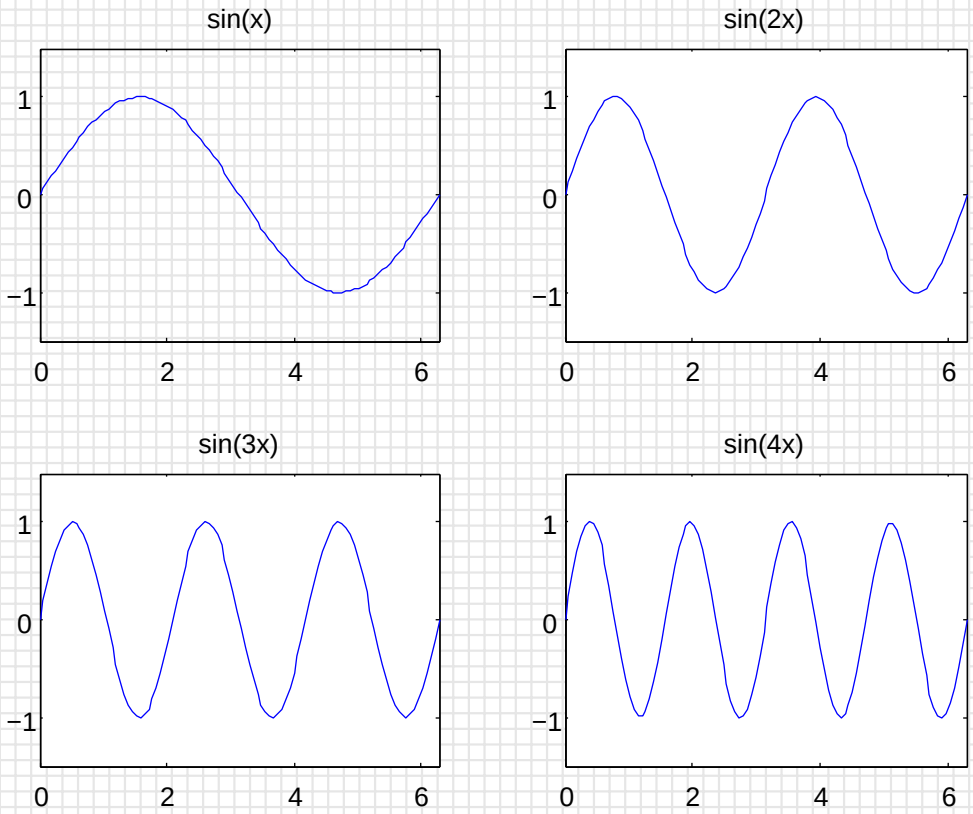
```
>> x = linspace(0, 2*pi);
>> subplot(2,2,1);
>> plot(x, sin(x)); axis([0 2*pi -1.5 1.5]);
title('sin(x)');

>> subplot(2,2,2);
>> plot(x, sin(2*x)); axis([0 2*pi -1.5 1.5]);
title('sin(2x)');

>> subplot(2,2,3);
>> plot(x, sin(3*x)); axis([0 2*pi -1.5 1.5]);
title('sin(3x)');

>> subplot(2,2,4);
>> plot(x, sin(4*x)); axis([0 2*pi -1.5 1.5]);
title('sin(4x)');
```

subplot



plot 的标注参数

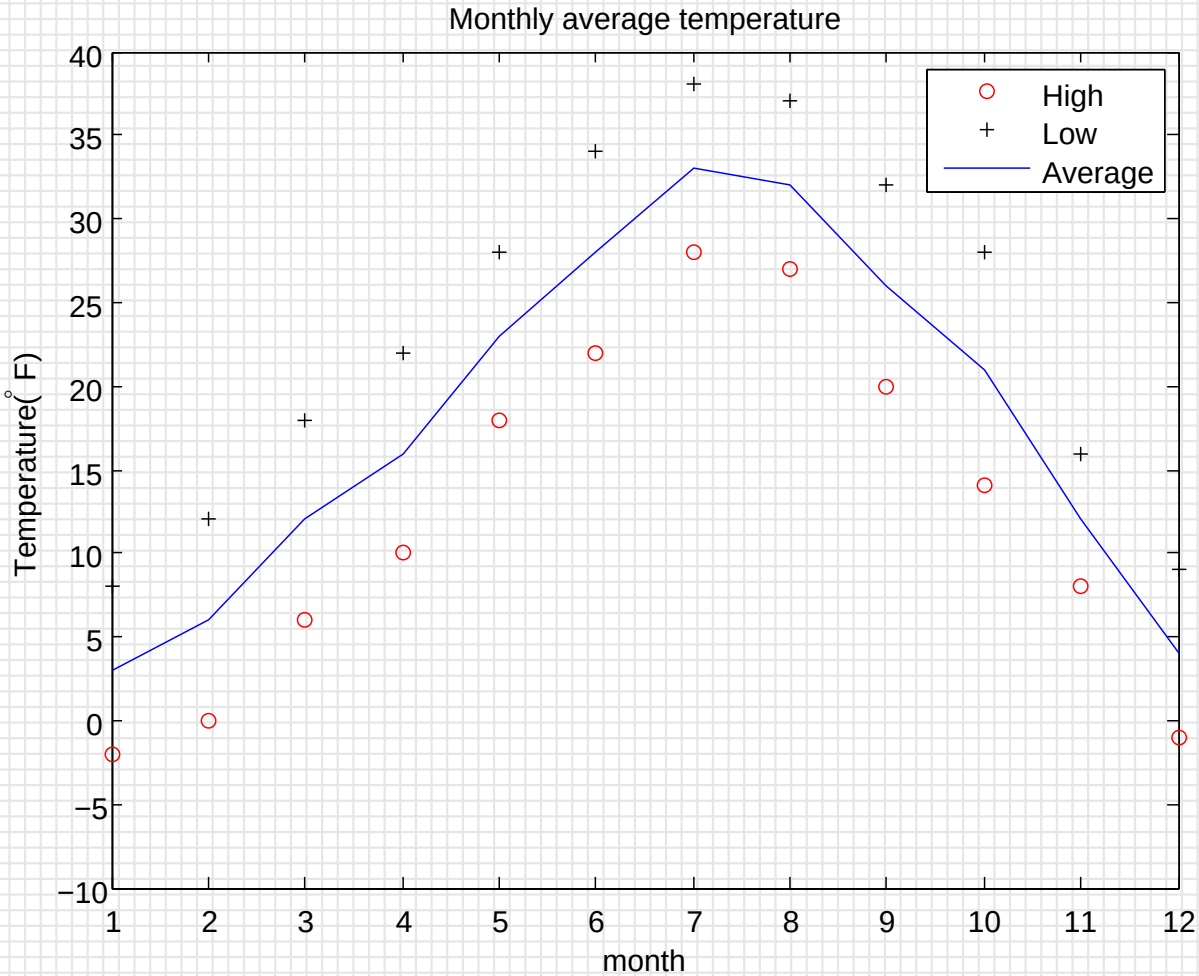
名称	轴标注
axis	设置坐标轴参数
grid	绘制 x、y 轴主刻度上的网格线
gtext	在鼠标点击处添加文字
legend	创建图例说明
text	在指定的坐标点处添加文字
xlabel	标注 x 轴
ylabel	标注 y 轴
title	在图示上方添加标题

表: plot 的标注参数

plot 标注范例

```
>> load pdxTemp.dat;
>> m = pdxTemp(:,1);
>> T = pdxTemp(:,2:4);
>> plot(m,T(:,1),'ro',m,T(:,2),'k+',m,T(:,3),'b-
');
>> xlabel('Month');
>> ylabel('Temperature␣({}^\circ␣F)');
>> title('Monthly␣average␣temperature␣for␣PDX');
>> axis([1 12 20 100]);
>> legend('High','Low','Average',1);
```

plot 标注范例


Monthly average temperature
High
Low
Average
Temperature(°F)
month
40
35
30
25
20
15
10
5
0
−5
−10
1
2
3
4
5
6
7
8
9
10
11
12