

入门

1

MATLAB 产品说明	1-2
主要功能	1-2
桌面基础知识	1-3
矩阵和数组	1-5
数组索引	1-8
工作区变量	1-10
文本和字符	1-11
字符串数组中的文本	1-11
字符数组中的数据	1-11
调用函数	1-13
二维图和三维图	1-14
编程和脚本	1-20
脚本	1-20
实时脚本	1-20
循环及条件语句	1-21
脚本位置	1-22
帮助和文档	1-23

语言基础知识

2

矩阵和幻方矩阵	2-2
关于矩阵	2-2
输入矩阵	2-3
矩阵求和、转置和对角矩阵	2-3
magic 函数	2-5
生成矩阵	2-5
表达式	2-7
变量	2-7
数字	2-7

矩阵运算符	2-8
数组运算符	2-8
函数	2-10
表达式示例	2-11
输入命令	2-12
format 函数	2-12
取消输出	2-13
输入长语句	2-13
命令行编辑	2-13
索引	2-14
下标	2-14
冒号运算符	2-14
串联	2-15
删除行和列	2-16
标量扩展	2-16
逻辑下标	2-17
find 函数	2-18
数组类型	2-19
多维数组	2-19
元胞数组	2-20
字符与文本	2-22
结构体	2-23

数学

3

线性代数	3-2
MATLAB 环境中的矩阵	3-2
线性方程组	3-9
分解	3-17
幂和指数	3-22
特征值	3-25
奇异值	3-28
非线性函数的运算	3-31
函数句柄	3-31
功能函数	3-31
多变量数据	3-34
数据分析	3-35
简介	3-35
数据的预处理	3-35
汇总数据	3-39
可视化数据	3-42
数据建模	3-50

基本绘图函数	4-2
创建绘图	4-2
在一幅图形中绘制多个数据集	4-4
指定线型和颜色	4-6
绘制线条和标记	4-7
绘制虚数和复数数据	4-8
将绘图添加到现有图形中	4-9
图窗窗口	4-10
在一幅图窗中显示多个绘图	4-11
控制轴	4-11
添加轴标签和标题	4-13
保存图窗	4-14
保存工作区数据	4-14
创建网格图和曲面图	4-15
关于网格图和曲面图	4-15
可视化包含两个变量的函数	4-15
显示图像	4-20
图像数据	4-20
读取和写入图像	4-21
打印图形	4-23
打印概述	4-23
从“文件”菜单打印	4-23
将图窗导出到图形文件	4-23
使用 Print 命令	4-23
处理图形对象	4-25
图形对象	4-25
设置对象属性	4-26
用于处理对象的函数	4-28
传递参数	4-29
查找现有对象的句柄	4-30

控制流	5-2
条件控制 - if、else、switch	5-2
循环控制 - for、while、continue、break	5-4
程序终止 - return	5-5
向量化	5-6
预分配	5-6
脚本和函数	5-7
概述	5-7
脚本	5-7

函数	5-8
函数类型	5-9
全局变量	5-10
命令与函数语法	5-11

入门

- “MATLAB 产品说明” (第 1-2 页)
- “桌面基础知识” (第 1-3 页)
- “矩阵和数组” (第 1-5 页)
- “数组索引” (第 1-8 页)
- “工作区变量” (第 1-10 页)
- “文本和字符” (第 1-11 页)
- “调用函数” (第 1-13 页)
- “二维图和三维图” (第 1-14 页)
- “编程和脚本” (第 1-20 页)
- “帮助和文档” (第 1-23 页)

MATLAB 产品说明

全世界数以百万计的工程师和科学家都在使用 MATLAB 分析和设计为我们的世界带来巨变的系统和产品。MATLAB 广泛应用于汽车主动安全系统、星际宇宙飞船、健康监控设备、智能电网和 LTE 蜂窝网络。它用于机器学习、信号处理、图像处理、计算机视觉、通信、计算金融、控制设计、机器人等。

数学、图形、编程。

MATLAB 平台进行了优化，可以更好地解决工程和科学问题。基于矩阵的 MATLAB 语言是世界上表示计算数学最自然的方式。可以使用内置图形轻松可视化数据和深入了解数据。您可以通过众多的预置工具箱，立即开始使用对您的领域而言非常重要的算法。欢迎您使用桌面环境进行试验、探索 and 发现。这些 MATLAB 工具和功能全部进行了严格测试，可彼此配合工作。

扩展、集成部署。

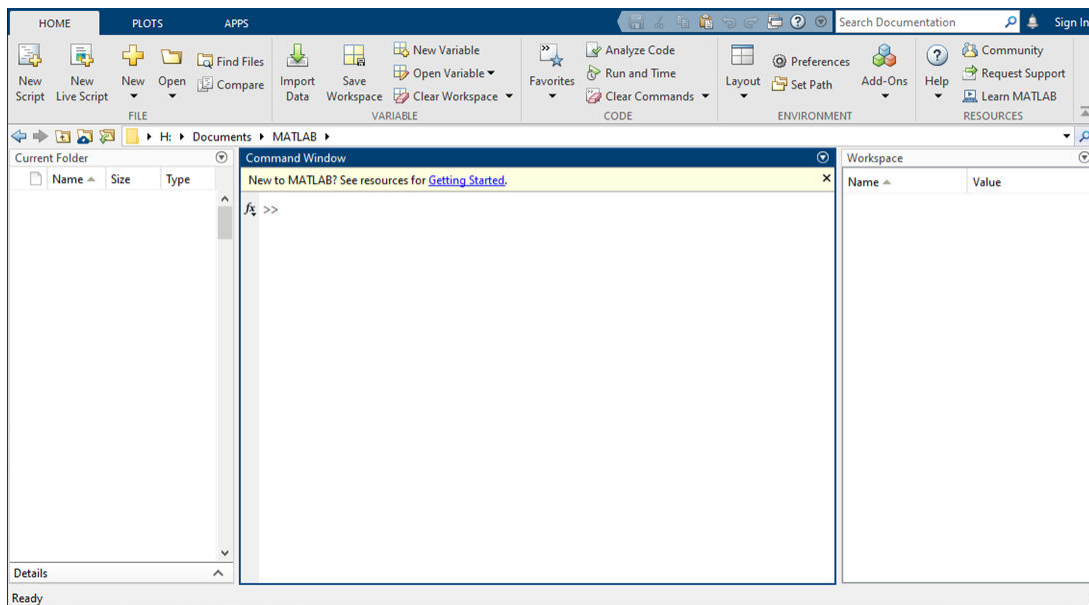
MATLAB 可帮助您不仅仅将自己的创意停留在桌面。您可以对大型数据集运行分析，并扩展到群集和云。MATLAB 代码可以与其他语言集成，使您能够在 Web、企业 and 生产系统中部署算法和应用程序。

主要功能

- 用于科学和工程计算的高级语言
- 为迭代探查、设计和问题求解而设计的桌面环境
- 用于可视化数据的图形和用于创建自定义绘图的工具
- 用于曲线拟合、数据分类、信号分析、控制系统优化和许多其他任务的 App
- 用于各种工程和科学应用程序的附加功能工具箱
- 用于构建包含自定义用户界面的应用程序的工具
- 用于 C/C++、Java®、.NET、Python、SQL、Hadoop 和 Microsoft® Excel® 的接口
- 用于将 MATLAB 程序与最终用户共享的无版权费部署选项

桌面基础知识

启动 MATLAB 时，桌面会以默认布局显示。



桌面包括下列面板：

- **当前文件夹** - 访问您的文件。
- **命令行窗口** - 在命令行中输入命令（由提示符 (>>) 表示）。
- **工作区** - 浏览您创建或从文件导入的数据。

使用 MATLAB 时，可发出创建变量和调用函数的命令。例如，通过在命令行中键入以下语句来创建名为 **a** 的变量：

```
a = 1
```

MATLAB 将变量 **a** 添加到工作区，并在命令行窗口中显示结果。

```
a =
```

```
1
```

创建更多变量。

```
b = 2
```

```
b =
```

```
2
```

```
c = a + b
```

```
c =
```

```
3
```

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

```
d =
```

```
0.5403
```

如果未指定输出变量，MATLAB 将使用变量 **ans**（answer 的缩略形式）来存储计算结果。

```
sin(a)
```

```
ans =
```

```
0.8415
```

如果语句以分号结束，MATLAB 会执行计算，但不在命令行窗口中显示输出。

```
e = a*b;
```

按向上 (↑) 和向下箭头键 (↓) 可以重新调用以前的命令。在空白命令行中或在键入命令的前几个字符之后按箭头键。例如，要重新调用命令 **b = 2**，请键入 **b**，然后按向上箭头键。

矩阵和数组

MATLAB 是 “matrix laboratory” 的缩写形式。MATLAB® 主要用于处理整个的矩阵和数组，而其他编程语言大多逐个处理数值。

所有 MATLAB 变量都是多维数组，与数据类型无关。矩阵是指通常用来进行线性代数运算的二维数组。

数组创建

要创建每行包含四个元素的数组，请使用逗号 (,) 或空格分隔各元素。

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

```
a = 1×4
```

```
1    2    3    4
```

这种数组为行向量。

要创建包含多行的矩阵，请使用分号分隔各行。

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

```
a = 3×3
```

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

创建矩阵的另一种方法是使用 `ones`、`zeros` 或 `rand` 等函数。例如，创建一个由零组成的 5×1 列向量。

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

```
z = 5×1
```

```
0
0
0
0
0
```

矩阵和数组运算

MATLAB 允许您使用单一的算术运算符或函数来处理矩阵中的所有值。

```
a + 10
```

```
ans = 3×3
```

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

```
sin(a)
```

```
ans = 3×3
```

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

要转置矩阵，请使用单引号 ('):

```
a'
```

```
ans = 3×3
```

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

您可以使用 `*` 运算符执行标准矩阵乘法，这将计算行与列之间的内积。例如，确认矩阵乘以其逆矩阵可返回单位矩阵：

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

```
p = 3×3
```

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

请注意，`p` 不是整数值矩阵。MATLAB 将数字存储为浮点值，算术运算可以区分实际值与其浮点表示之间的细微差别。使用 `format` 命令可以显示更多小数位数：

```
format long
```

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

```
p = 3×3
```

```
    1.0000000000000000         0         0
    0.0000000000000002    1.0000000000000000         0
    0.0000000000000002   -0.0000000000000004    1.0000000000000000
```

使用以下命令将显示内容重置为更短格式

```
format short
```

`format` 仅影响数字显示，而不影响 MATLAB 对数字的计算或保存方式。

要执行元素级乘法（而非矩阵乘法），请使用 `.*` 运算符：

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

```
p = 3×3
```

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

乘法、除法和幂的矩阵运算符分别具有执行元素级运算的对应数组运算符。例如，计算 **a** 的各个元素的三次方：

```
a.^3
```

```
ans = 3×3
```

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

串联

串联是连接数组以便形成更大数组的过程。实际上，第一个数组是通过将其各个元素串联起来而构成的。成对的方括号 `[]` 即为串联运算符。

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

```
A = 3×6
```

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

使用逗号将彼此相邻的数组串联起来称为水平串联。每个数组必须具有相同的行数。同样，如果各数组具有相同的列数，则可以使用分号垂直串联。

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

```
A = 6×3
```

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

复数

复数包含实部和虚部，虚数单位是 -1 的平方根。

```
sqrt(-1)
```

```
ans = 0.0000 + 1.0000i
```

要表示复数的虚部，请使用 **i** 或 **j**。

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

```
c = 2×2 complex
```

```
 3.0000 + 4.0000i  4.0000 + 3.0000i
 0.0000 - 1.0000i  0.0000 +10.0000i
```

数组索引

MATLAB® 中的每个变量都是一个可包含许多数字的数组。如果要访问数组的选定元素，请使用索引。

以 4×4 幻方矩阵 A 为例：

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

```
A = 4×4
```

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

引用数组中的特定元素有两种方法。最常见的方法是指定行和列下标，例如

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

```
ans = 14
```

另一种方法不太常用，但有时非常有用，即使用单一下标按顺序向下遍历每一列：

```
A(8)
```

```
ans = 14
```

使用单一下标引用数组中特定元素的方法称为线性索引。

如果尝试在赋值语句右侧引用数组外部元素，MATLAB 会引发错误。

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

Index exceeds matrix dimensions.

不过，您可以在赋值语句左侧指定当前维外部的元素。数组大小会增大以便容纳新元素。

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

```
A = 4×5
```

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

要引用多个数组元素，请使用冒号运算符，这使您可以指定一个格式为 **start:end** 的范围。例如，列出 A 前三行及第二列中的元素：

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

```
ans = 3×1
```

```
2
11
7
```

单独的冒号（没有起始值或结束值）指定该维中的所有元素。例如，选择 A 第三行中的所有列：

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

```
ans = 1×5
```

```
    9    7    6   12    0
```

此外，冒号运算符还允许您使用较通用的格式 `start:step:end` 创建等距向量值。

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

```
B = 1×11
```

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

如果省略中间的步骤（如 `start:end` 中），MATLAB 会使用默认步长值 1。

工作区变量

工作区包含在 MATLAB 中创建或从数据文件或其他程序导入的变量。例如，下列语句在工作区中创建变量 A 和 B。

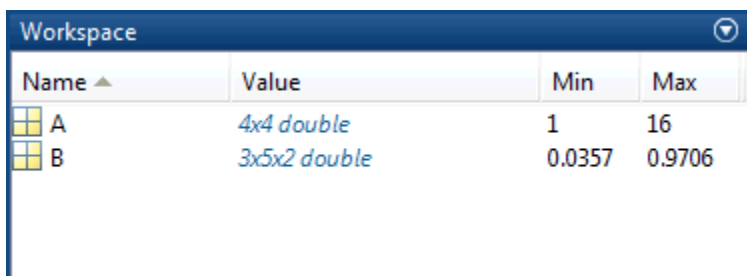
```
A = magic(4);  
B = rand(3,5,2);
```

使用 `whos` 可以查看工作区的内容。

```
whos
```

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

此外，桌面上的“工作区”窗格也会显示变量。



退出 MATLAB 后，工作区变量不会保留。使用 `save` 命令保存数据以供将来使用，

```
save myfile.mat
```

通过保存，系统会使用 `.mat` 扩展名将工作区保存在当前工作文件夹中一个名为 MAT 文件的压缩文件中。

要清除工作区中的所有变量，请使用 `clear` 命令。

使用 `load` 将 MAT 文件中的数据还原到工作区。

```
load myfile.mat
```

文本和字符

字符串数组中的文本

当您处理文本时，将字符序列括在双引号中。可以将文本赋给变量。

```
t = "Hello, world";
```

如果文本包含双引号，请在定义中使用两个双引号。

```
q = "Something "quoted" and something else."
```

```
q =
```

```
"Something "quoted" and something else."
```

与所有 MATLAB 变量一样，**t** 和 **q** 为数组。它们的类或数据类型是 **string**。

```
whos t
```

Name	Size	Bytes	Class	Attributes
t	1x1	174	string	

注意 使用双引号创建字符串数组是在 R2017a 中引入的。如果您使用的是更早期的版本，请创建字符数组。有关详细信息，请参阅“字符数组中的数据”（第 1-11 页）。

要将文本添加到字符串的末尾，请使用加号运算符 **+**。

```
f = 71;
c = (f-32)/1.8;
tempText = "Temperature is " + c + "C"
```

```
tempText =
"Temperature is 21.6667C"
```

与数值数组类似，字符串数组可以有多个元素。使用 **strlength** 函数求数组中每个字符串的长度。

```
A = ["a","bb","ccc","dddd","eeeeee","ffffff"]
```

```
A =
2x3 string array
"a"    "bb"    "ccc"
"dddd" "eeeeee" "ffffff"
```

```
strlength(A)
```

```
ans =
```

```
1 2 3
4 6 7
```

字符数组中的数据

有时，字符表示的数据并不对应到文本，例如 DNA 序列。您可以将此类数据存储在数据类型为 **char** 的字符数组中。字符数组使用单引号。

```
seq = 'GCTAGAATCC';  
whos seq
```

Name	Size	Bytes	Class	Attributes
seq	1x10	20	char	

数组的每个元素都包含单个字符。

```
seq(4)
```

```
ans =  
'A'
```

使用方括号串联字符数组，就像串联数值数组一样。

```
seq2 = [seq 'ATTAGAAACC']
```

```
seq2 =  
'GCTAGAATCCATTAGAAACC'
```

在字符串数组引入之前编写的程序中，字符数组很常见。接受 **string** 数据的所有 MATLAB 函数都能接受 **char** 数据，反之亦然。

调用函数

MATLAB® 提供了大量执行计算任务的函数。在其他编程语言中，函数等同于子例程或方法。

要调用函数，例如 `max`，请将其输入参数括在圆括号中：

```
A = [1 3 5];  
max(A)
```

```
ans = 5
```

如果存在多个输入参数，请使用逗号加以分隔：

```
B = [10 6 4];  
max(A,B)
```

```
ans = 1×3
```

```
10    6    5
```

通过将函数赋值给变量，返回该函数的输出：

```
maxA = max(A)
```

```
maxA = 5
```

如果存在多个输出参数，请将其括在方括号中：

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

```
maxA = 5
```

```
location = 3
```

将任何字符输入括在单引号中：

```
disp('hello world')
```

```
hello world
```

要调用不需要任何输入且不会返回任何输出的函数，请只键入函数名称：

```
clc
```

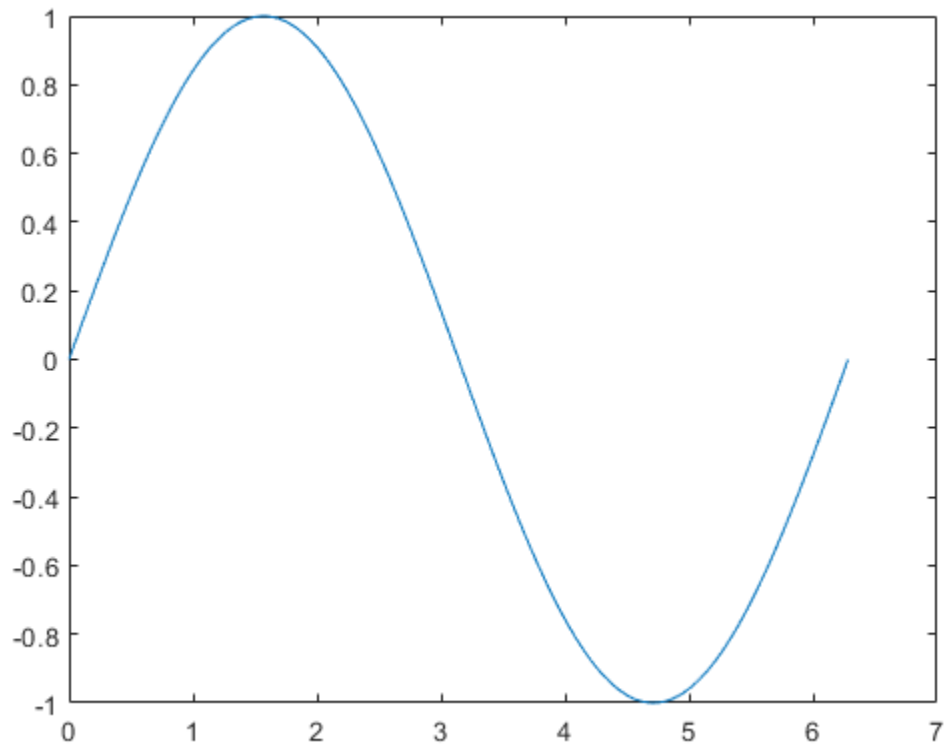
`clc` 函数清空命令行窗口。

二维图和三维图

线图

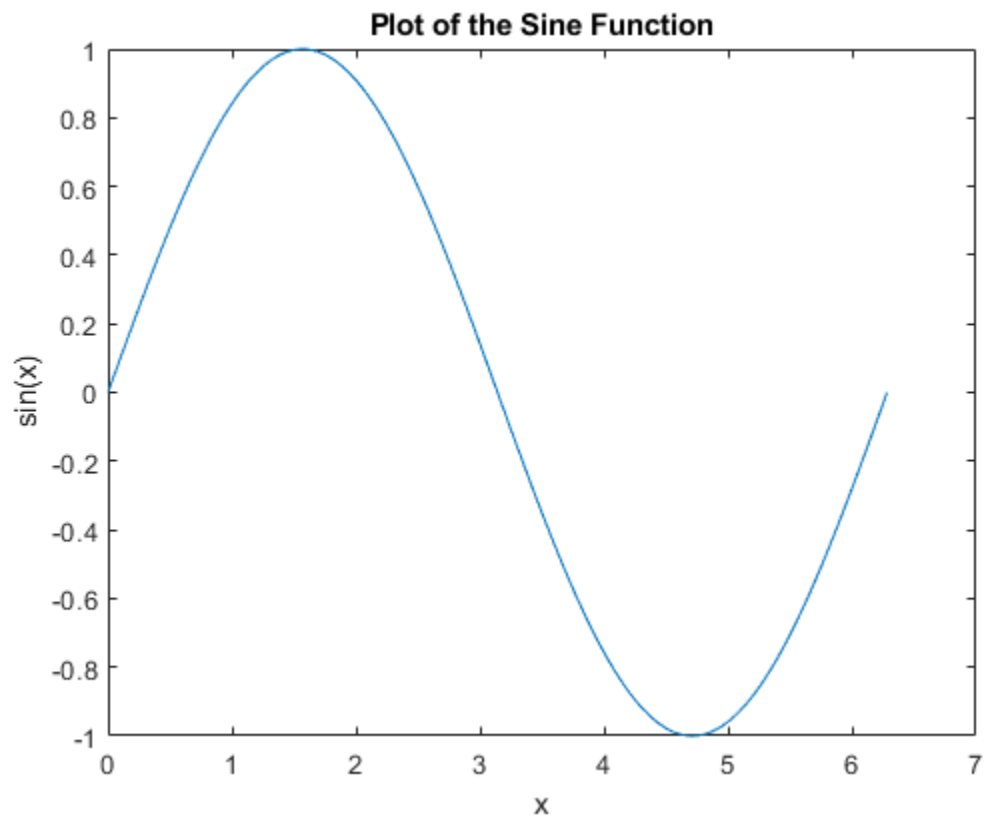
要创建二维线图，请使用 `plot` 函数。例如，绘制从 0 到 2π 之间的正弦函数值：

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



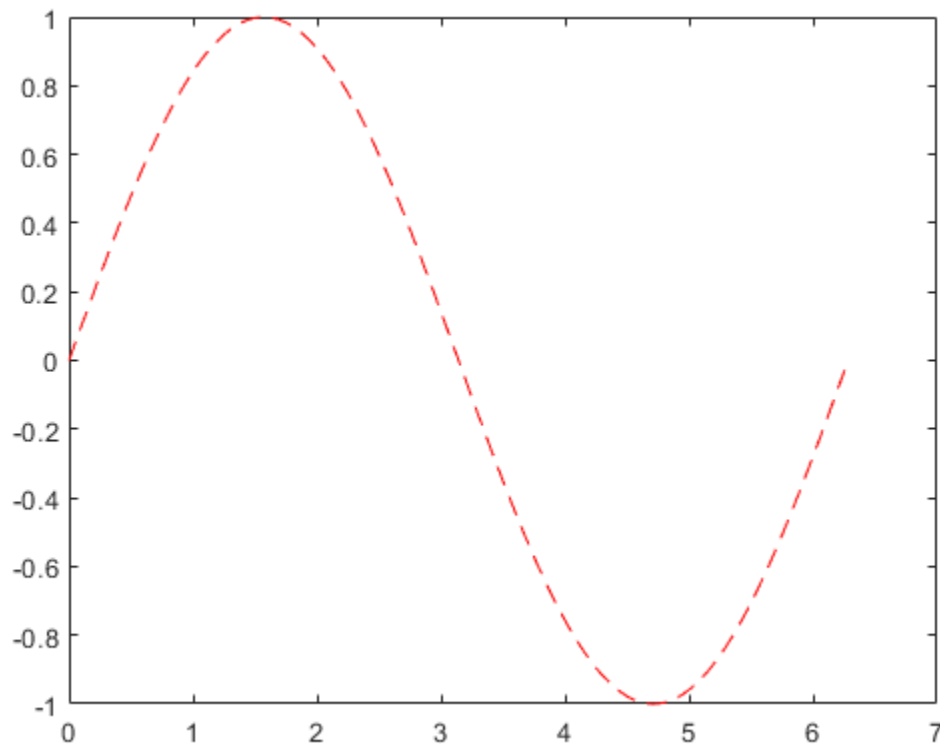
可以标记轴并添加标题。

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



通过向 **plot** 函数添加第三个输入参数，您可以使用红色虚线绘制相同的变量。

plot(x,y,'r-')



'r--' 为线条设定。每个设定可包含表示线条颜色、样式和标记的字符。标记是在绘制的每个数据点上显示的符号，例如，+、o 或 *。例如，'g:*' 请求绘制使用 * 标记的绿色点线。

请注意，为第一幅绘图定义的标题和标签不再被用于当前的图窗窗口中。默认情况下，每次调用绘图函数、重置坐标区及其他元素以准备新绘图时，MATLAB® 都会清空图窗。

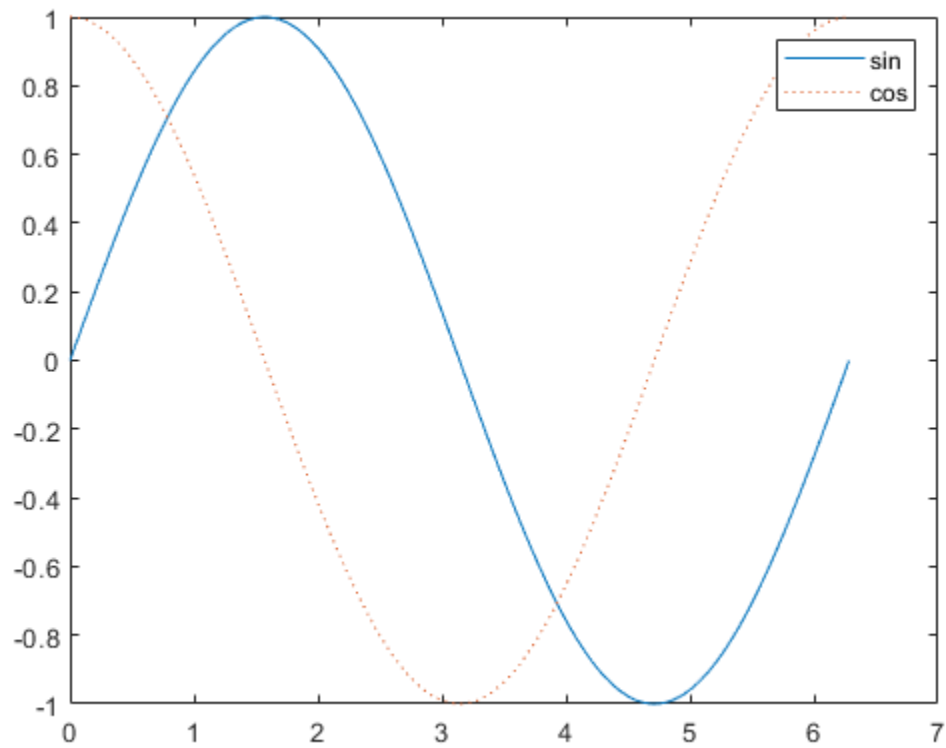
要将绘图添加到现有图窗中，请使用 **hold on**。在使用 **hold off** 或关闭窗口之前，当前图窗窗口中会显示所有绘图。

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

hold on

```
y2 = cos(x);  
plot(x,y2,':')  
legend('sin','cos')
```

hold off



三维绘图

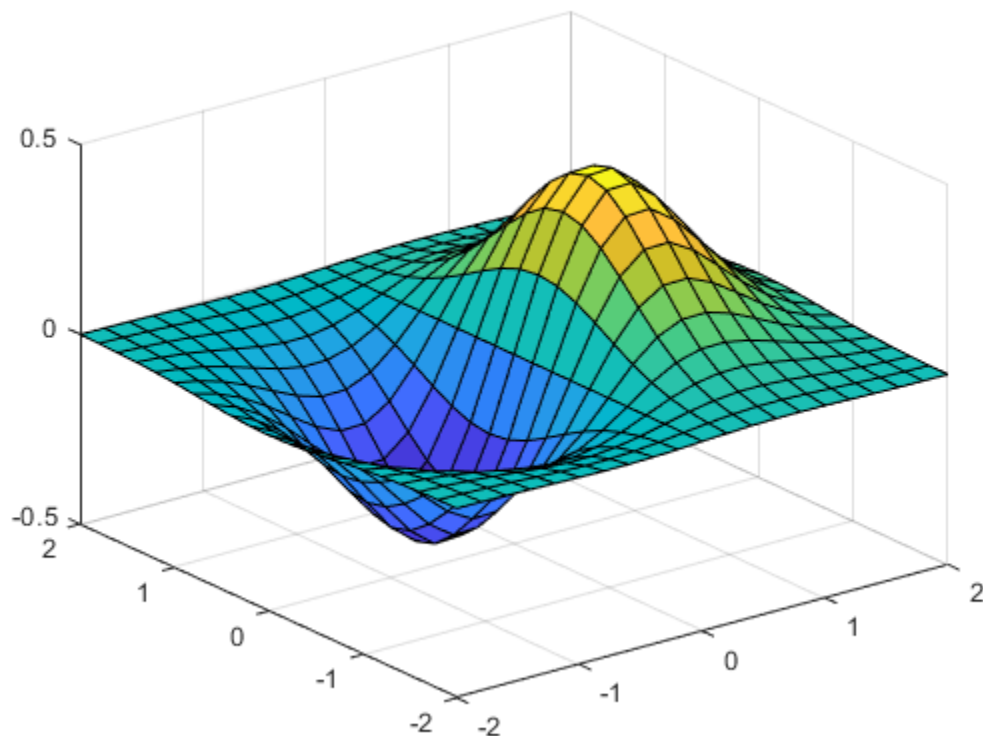
三维图通常显示一个由带两个变量的函数（即 $z = f(x,y)$ ）定义的曲面图。

要计算 z ，请首先使用 `meshgrid` 在此函数的域中创建一组 (x,y) 点。

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

然后，创建曲面图。

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



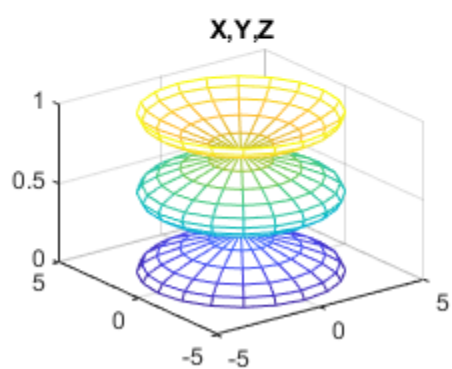
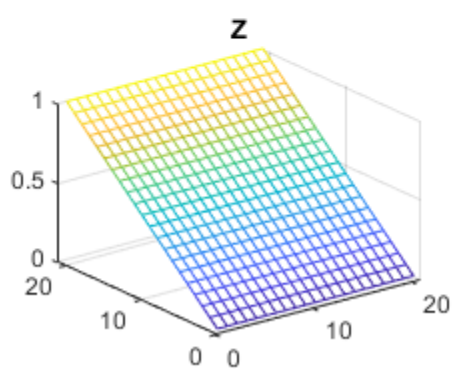
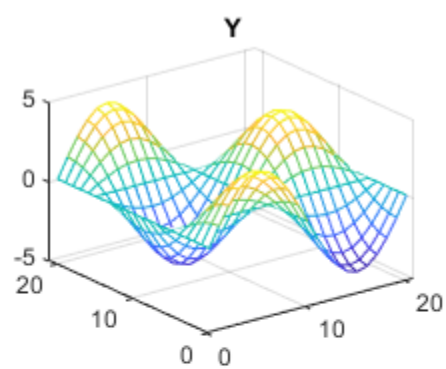
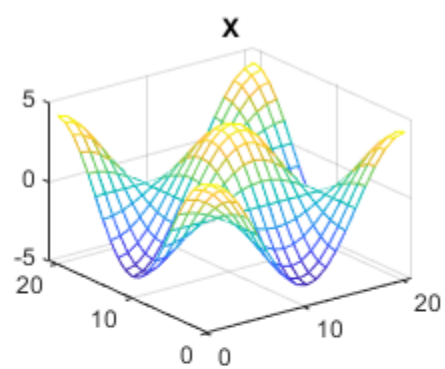
surf 函数及其伴随函数 **mesh** 以三维形式显示曲面图。**surf** 使用颜色显示曲面图的连接线和面。**mesh** 生成仅以颜色标记连接定义点的线条的线框曲面图。

子图

使用 **subplot** 函数可以在同一窗口的不同子区域显示多个绘图。

subplot 的前两个输入表示每行和每列中的绘图数。第三个输入指定绘图是否处于活动状态。例如，在图窗窗口的 2×2 网格中创建四个绘图。

```
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');
```



编程和脚本

本节内容
“脚本” (第 1-20 页)
“实时脚本” (第 1-20 页)
“循环及条件语句” (第 1-21 页)
“脚本位置” (第 1-22 页)

脚本是最简单的一种 MATLAB 程序。脚本是一个包含多行连续的 MATLAB 命令和函数调用的文件。在命令行中键入脚本名称即可运行该脚本。

脚本

要创建脚本，请使用 `edit` 命令。

`edit mysphere`

该命令会打开一个名为 `mysphere.m` 的空白文件。输入代码，以创建一个单位球、将半径加倍并绘制结果图：

```
[x,y,z] = sphere;  
r = 2;  
surf(x*r,y*r,z*r)  
axis equal
```

接下来，添加代码以计算球的表面积和体积：

```
A = 4*pi*r^2;  
V = (4/3)*pi*r^3;
```

编写代码时，最好添加描述代码的注释。注释能够让其他人员理解您的代码，并且有助于您在稍后返回代码时再度记起。使用百分比 (%) 符号添加注释。

```
% Create and plot a sphere with radius r.  
[x,y,z] = sphere;    % Create a unit sphere.  
r = 2;  
surf(x*r,y*r,z*r)    % Adjust each dimension and plot.  
axis equal           % Use the same scale for each axis.  
  
% Find the surface area and volume.  
A = 4*pi*r^2;  
V = (4/3)*pi*r^3;
```

将文件保存在当前文件夹中。要运行脚本，请在命令行中键入脚本名称：


`mysphere`

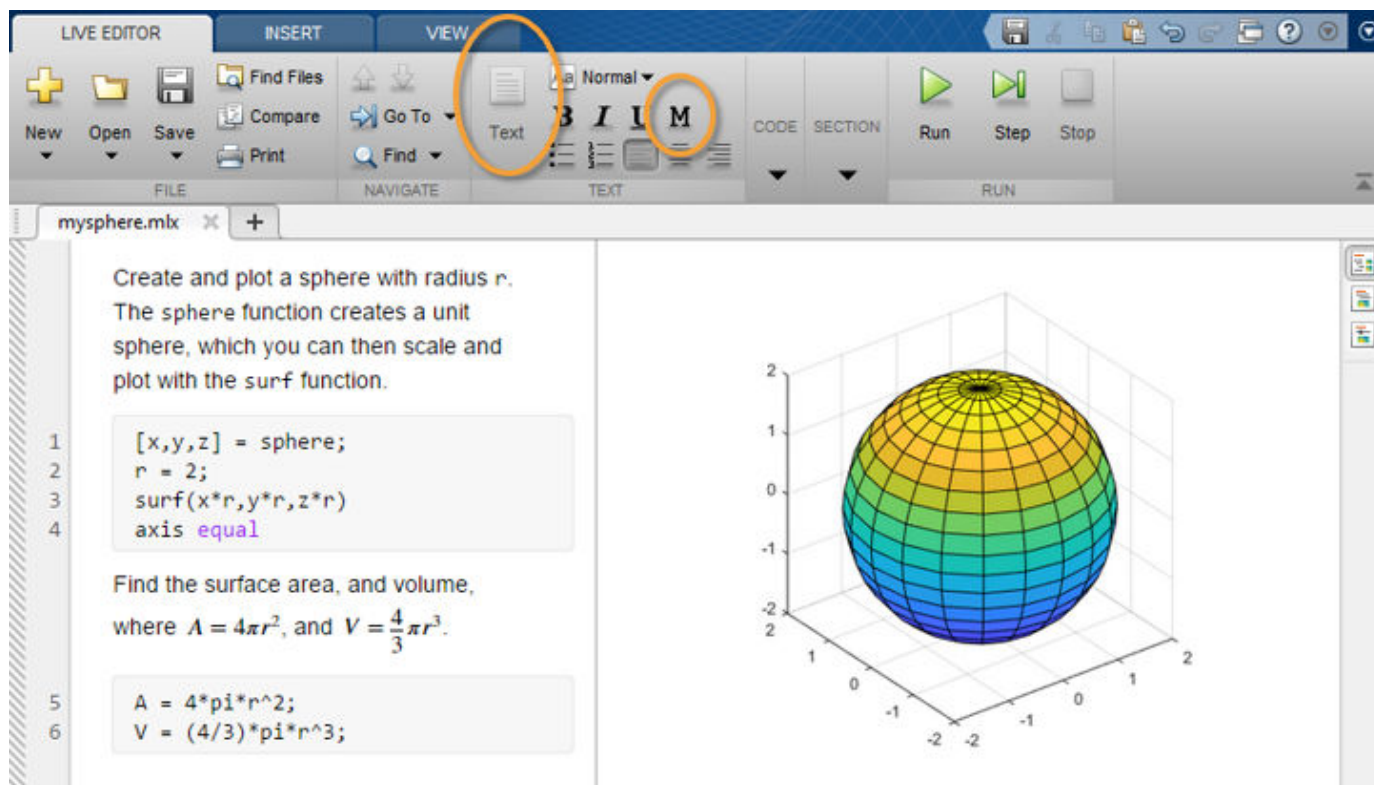
还可以从编辑器使用运行按钮  运行脚本。

实时脚本

您可以使用实时脚本中的格式设置选项来增强代码，而不是以纯文本编写代码和注释。实时脚本有助于您查看代码和输出并与之交互，还可以包含格式化文本、方程和图像。

例如，通过选择**另存为**并将文件类型更改为 MATLAB 实时代码文件 (*.mlx)，将 `mysphere` 转换为实时脚本。然后，用格式化文本替换代码注释。例如：

- 将注释行转换为文本。选择以百分比符号开头的每一行，然后选择**文本**、。删除百分比符号。
- 重写文本以替换代码行末尾的注释。要将等宽字体应用于文本中的函数名，请选择 **M**。要添加方程，请在**插入**选项卡上选择**方程**。



要使用 `edit` 命令创建新的实时脚本，请在文件名中包含 `.mlx` 扩展名：

`edit newfile.mlx`

循环及条件语句

在任何脚本中，您都可以定义按循环重复执行或按条件执行的代码段。循环使用 `for` 或 `while` 关键字，条件语句使用 `if` 或 `switch`。

循环在创建序列时很有用。例如，创建一个名为 `fibseq` 的脚本，该脚本使用 `for` 循环来计算斐波那契数列的前 100 个数。在这个序列中，最开始的两个数是 1，随后的每个数是前面两个数的和，即 $F_n = F_{n-1} + F_{n-2}$ 。

```
N = 100;
f(1) = 1;
f(2) = 1;
```

```
for n = 3:N
    f(n) = f(n-1) + f(n-2);
```

```
end  
f(1:10)
```

运行该脚本时，`for` 语句定义一个名为 `n` 的计数器，该计数器从 3 开始。然后，该循环重复为 `f(n)` 赋值，`n` 在每次执行中递增，直至达到 100。脚本中的最后一条命令 `f(1:10)` 显示 `f` 的前 10 个元素。

```
ans =  
    1    1    2    3    5    8   13   21   34   55
```

条件语句仅在给定表达式为 `true` 时执行。例如，根据随机数的大小为变量赋值：`'low'`、`'medium'` 或 `'high'`。在本例中，随机数是在 1 和 100 之间的一个整数。

```
num = randi(100)  
if num < 34  
    sz = 'low'  
elseif num < 67  
    sz = 'medium'  
else  
    sz = 'high'  
end
```

语句 `sz = 'high'` 仅在 `num` 大于或等于 67 时执行。

脚本位置

MATLAB 在特定位置中查找脚本及其他文件。要运行脚本，该文件必须位于当前文件夹或搜索路径中的某个文件夹内。

默认情况下，MATLAB 安装程序创建的 **MATLAB** 文件夹位于此搜索路径中。如果要将程序存储在其他文件夹，或者要运行其他文件夹中的程序，请将其添加到此搜索路径。在当前文件夹浏览器中选中相应的文件夹，右键点击，然后选择**添加到路径**。

帮助和文档

所有 MATLAB 函数都有辅助文档，这些文档包含一些示例，并介绍函数输入、输出和调用语法。从命令行访问此信息有多种方法：

- 使用 **doc** 命令在单独的窗口中打开函数文档。

doc mean

- 在键入函数输入参数的左括号之后暂停，此时命令行窗口中会显示相应函数的提示（函数文档的语法部分）。

mean(

- 使用 **help** 命令可在命令行窗口中查看相应函数的简明文档。

help mean

点击帮助图标  即可访问完整的产品文档。

语言基础知识

- “矩阵和幻方矩阵” (第 2-2 页)
- “表达式” (第 2-7 页)
- “输入命令” (第 2-12 页)
- “索引” (第 2-14 页)
- “数组类型” (第 2-19 页)

矩阵和幻方矩阵

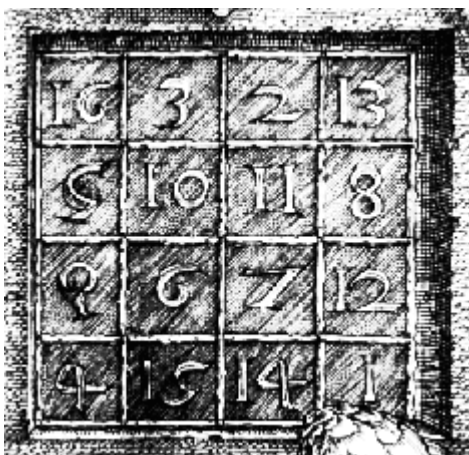
本节内容
“关于矩阵” (第 2-2 页)
“输入矩阵” (第 2-3 页)
“矩阵求和、转置和对角矩阵” (第 2-3 页)
“magic 函数” (第 2-5 页)
“生成矩阵” (第 2-5 页)

关于矩阵

在 MATLAB 环境中，矩阵是由数字组成的矩形数组。有时， 1×1 矩阵（即标量）和只包含一行或一列的矩阵（即向量）会附加特殊含义。MATLAB 采用其他方法来存储数值数据和非数值数据，但刚开始时，通常最好将一切内容都视为矩阵。MATLAB 旨在尽可能简化运算。其他编程语言一次只能处理一个数字，而 MATLAB 允许您轻松快捷地处理整个矩阵。本手册中使用的有效示例矩阵摘自德国艺术家和业余数学家 Albrecht Dürer 在文艺复兴时期的雕刻 Melencolia I。



这幅图布满了数学符号，通过仔细观察，您会发现右上角有一个矩阵。此矩阵称为幻方矩阵，在 Dürer 所处的时代，此幻方矩阵被视为富有真正的神秘性质。它具有某些值得让人深究的迷人特征。



输入矩阵

开始学习 MATLAB 的最佳方法是了解如何处理矩阵。启动 MATLAB 并按照每个示例操作。

您可以采用多种不同方法在 MATLAB 中输入矩阵：

- 输入元素的明确列表。
- 从外部数据文件加载矩阵。
- 使用内置函数生成矩阵。
- 使用您自己的函数创建矩阵，并将其保存在文件中。

首先，以元素列表的形式输入丢勒的矩阵。您只需遵循一些基本约定：

- 使用空格或逗号分隔行的元素。
- 使用分号 ; 表示每行末尾。
- 使用方括号 [] 将整个元素列表括起来。

要输入丢勒矩阵，只需在命令行窗口中键入即可

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB 显示刚才您输入的矩阵：

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

此矩阵与雕刻中的数字一致。输入矩阵之后，MATLAB 工作区会自动记住此矩阵。您可以将其简称为 A。现在，您已经在工作区中输入 A，让我们看看它为什么如此有趣吧。它有什么神奇的地方呢？

矩阵求和、转置和对角矩阵

您可能已经注意到，幻方矩阵的特殊属性与元素的不同求和方法相关。如果沿任何行或列求和，或者沿两条主对角线中的任意一条求和，您将始终得到相同数字。让我们使用 MATLAB 来验证这一点。尝试的第一个语句是

```
sum(A)
```

MATLAB 返回的结果为

```
ans =  
    34    34    34    34
```

如果未指定输出变量，MATLAB 将使用变量 **ans**（answer 的缩略形式）来存储计算结果。您已经计算包含 A 的列总和的行向量。每个列的总和都相同，即幻数和 34。

行总和如何处理？MATLAB 会优先处理矩阵的列，因此获取行总和的一种方法是转置矩阵，计算转置的列总和，然后转置结果。

MATLAB 具有两个转置运算符。撇号运算符（例如，A'）执行复共轭转置。它会围绕主对角线翻转矩阵，并且还会更改矩阵的任何复数元素的虚部符号。点撇号运算符 (A.') 转置矩阵，但不会影响复数元素的符号。对于包含所有实数元素的矩阵，这两个运算符返回相同结果。

因此

```
A'
```

生成

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

而

```
sum(A)'
```

生成包含行总和的列向量

```
ans =  
    34  
    34  
    34  
    34
```

有关避免双重转置的其他方法，请在 **sum** 函数中使用维度参数：

```
sum(A,2)
```

生成

```
ans =  
    34  
    34  
    34  
    34
```

使用 **sum** 和 **diag** 函数可以获取主对角线上的元素的总和：

```
diag(A)
```

生成


```
ans =
    16
    10
     7
     1
```

而

```
sum(diag(A))
```

生成

```
ans =
    34
```

从数学上讲，另一条对角线（即所谓的反对角线）并不是十分重要，因此 MATLAB 没有对此提供现成的函数。但原本用于图形的函数 **fliplr** 可以从左往右地翻转矩阵：

```
sum(diag(fliplr(A)))
ans =
    34
```

您已经验证丢勒雕刻中的矩阵确实是一个幻方矩阵，同时在验证过程中，您已经尝试了几个 MATLAB 矩阵运算。下面各部分继续使用此矩阵来演示 MATLAB 的其他功能。

magic 函数

MATLAB 实际包含一个内置函数，该函数可创建几乎任意大小的幻方矩阵。此函数称为 **magic** 也就不足为奇了：

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

此矩阵几乎与丢勒雕刻中的矩阵相同，并且具有所有相同的“神奇”性质；唯一区别在于交换了中间两列。

您可以交换 **B** 的中间两列，使其看起来像丢勒 **A**。针对 **B** 中的每一行，按照指定顺序（1、3、2、4）对列进行重新排列：

```
A = B(:,[1 3 2 4])
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

生成矩阵

MATLAB 软件提供了四个用于生成基本矩阵的函数。

zeros 全部为零

ones	全部为 1
rand	均匀分布的随机元素
randn	正态分布的随机元素

下面给出了一些示例：

```
Z = zeros(2,4)
```

```
Z =  
    0    0    0    0  
    0    0    0    0
```

```
F = 5*ones(3,3)
```

```
F =  
    5    5    5  
    5    5    5  
    5    5    5
```

```
N = fix(10*rand(1,10))
```

```
N =  
    9    2    6    4    8    7    4    0    8    4
```

```
R = randn(4,4)
```

```
R =  
    0.6353    0.0860   -0.3210   -1.2316  
   -0.6014   -2.0046    1.2366    1.0556  
    0.5512   -0.4931   -0.6313   -0.1132  
   -1.0998    0.4620   -2.3252    0.3792
```

表达式

本节内容
“变量” (第 2-7 页)
“数字” (第 2-7 页)
“矩阵运算符” (第 2-8 页)
“数组运算符” (第 2-8 页)
“函数” (第 2-10 页)
“表达式示例” (第 2-11 页)

变量

与大多数其他编程语言一样，MATLAB 语言提供数学表达式，但与大多数编程语言不同的是，这些表达式涉及整个矩阵。

MATLAB 不需要任何类型声明或维度说明。当 MATLAB 遇到新的变量名称时，它会自动创建变量，并分配适当大小的存储。如果此变量已存在，MATLAB 会更改其内容，并根据需要分配新存储。例如，

```
num_students = 25
```

创建一个名为 `num_students` 的 1×1 矩阵，并将值 25 存储在该矩阵的单一元素中。要查看分配给任何变量的矩阵，只需输入变量名称即可。

变量名称包括一个字母，后面可以跟随任意数目的字母、数字或下划线。MATLAB 区分大小写；它可以区分大写和小写字母。`A` 和 `a` 不是相同变量。

尽管变量名称可以为任意长度，MATLAB 仅使用名称的前 N 个字符（其中 N 是函数 `namelengthmax` 返回的数字），并忽略其余字符。因此，很重要的一点是，应使每个变量名称的前 N 个字符保持唯一，以便 MATLAB 能够区分变量。

```
N = namelengthmax
N =
    63
```

数字

MATLAB 使用传统的十进制记数法以及可选的小数点和前导加号或减号来表示数字。科学记数法使用字母 `e` 来指定 10 次方的缩放因子。虚数使用 `i` 或 `j` 作为后缀。下面给出了合法数字的一些示例：

```
3          -99          0.0001
9.6397238  1.60210e-20  6.02252e23
1i         -3.14159j    3e5i
```

MATLAB 使用 IEEE® 浮点标准规定的 `long` 格式在内部存储所有数字。浮点数的有限精度约为 16 位有效小数位数，有限范围约为 10⁻³⁰⁸ 至 10⁺³⁰⁸。

以双精度格式表示的数字的最大精度为 52 位。任何需要 52 位以上的双精度数字都会损失一定精度。例如，下面的代码因截断而将两个不相等的值显示为相等：

```
x = 36028797018963968;
y = 36028797018963972;
```

```
x == y
ans =
    1
```

整数的可用精度为 8 位、16 位、32 位和 64 位。将相同数字存储为 64 位整数会保留精度：

```
x = uint64(36028797018963968);
y = uint64(36028797018963972);
x == y
ans =
    0
```

MATLAB 软件存储复数的实部和虚部。该软件根据上下文采用不同方法来处理各个部分的量值。例如，`sort` 函数根据量值进行排序，如果量值相等，则根据相位角度排序。

```
sort([3+4i, 4+3i])
ans =
    4.0000 + 3.0000i    3.0000 + 4.0000i
```

这是由相位角度所致：

```
angle(3+4i)
ans =
    0.9273
angle(4+3i)
ans =
    0.6435
```

“等于”关系运算符 `==` 要求实部和虚部相等。其他二进制关系运算符 `>`、`<`、`>=` 和 `<=` 忽略数字的虚部，而仅考虑实部。

矩阵运算符

表达式使用大家熟悉的算术运算符和优先法则。

+	加法
-	减法
*	乘法
/	除法
\	左除
^	幂
'	复共轭转置
()	指定计算顺序

数组运算符

如果矩阵不用于线性代数运算，则成为二维数值数组。数组的算术运算按元素执行。这意味着，加法和减法运算对数组和矩阵都是相同的，但乘法运算不相同。MATLAB 的乘法数组运算表示法中包含点，也就是小数点。

运算符列表包括

+	加法
-	减法
.*	逐元素乘法
./	逐元素除法
.\	逐元素左除
.^	逐元素幂
.'	非共轭数组转置

如果使用数组乘法将丢勒的幻方矩阵自乘

```
A.*A
```

则会生成一个数组，该数组包含介于 1 至 16 之间的整数的平方，并且以不常见的顺序排列：

```
ans =
    256     9     4   169
     25   100   121    64
     81    36    49   144
     16   225   196     1
```

构建表

数组运算对构建表非常有用。假定 **n** 为列向量

```
n = (0:9)';
```

然后，

```
pows = [n n.^2 2.^n]
```

构建一个平方和 2 次幂的表：

```
pows =
     0     0     1
     1     1     2
     2     4     4
     3     9     8
     4    16    16
     5    25    32
     6    36    64
     7    49   128
     8    64   256
     9    81   512
```

初等数学函数逐元素处理数组元素。因此

```
format short g
x = (1:0.1:2)';
logs = [x log10(x)]
```

构建一个对数表。

```
logs =
     1.0         0
     1.1    0.04139
```

1.2	0.07918
1.3	0.11394
1.4	0.14613
1.5	0.17609
1.6	0.20412
1.7	0.23045
1.8	0.25527
1.9	0.27875
2.0	0.30103

函数

MATLAB 提供了大量标准初等数学函数，包括 **abs**、**sqrt**、**exp** 和 **sin**。生成负数的平方根或对数不会导致错误；系统会自动生成相应的复数结果。MATLAB 还提供了许多其他高等数学函数，包括贝塞尔函数和 gamma 函数。其中的大多数函数都接受复数参数。有关初等数学函数的列表，请键入

help elfun

有关更多高等数学函数和矩阵函数的列表，请键入

help specfun

help elmat

某些函数（例如，**sqrt** 和 **sin**）是内置函数。内置函数是 MATLAB 核心的一部分，因此这些函数非常高效，但计算详细信息是不可访问的。其他函数使用 MATLAB 编程语言实现，因此可以访问其计算详细信息。

内置函数与其他函数之间存在一些差异。例如，对于内置函数，您看不到代码。对于其他函数，您可以看到代码，甚至可以根据需要修改代码。

一些特殊函数提供了有用的常量值。

pi	3.14159265...
i	虚数单位 $\sqrt{-1}$
j	与 i 相同
eps	浮点相对精度 $\varepsilon = 2^{-52}$
realmin	最小浮点数 2^{-1022}
realmax	最大浮点数 $(2 - \varepsilon)2^{1023}$
Inf	无穷大
NaN	非数字

通过将非零值除以零或计算明确定义的溢出（即超过 **realmax**）的数学表达式，会生成无穷大。通过尝试计算 **0/0** 或 **Inf-Inf** 等没有明确定义的数值的表达式，会生成非数字。

函数名称不会保留。您可以使用如下新变量覆盖任何函数名称

eps = 1.e-6

并在后续计算中使用该值。可以使用以下命令恢复原始函数

clear eps

表达式示例

您已经学习了 MATLAB 表达式的几个示例。下面是一些其他示例及生成的值：

```
rho = (1+sqrt(5))/2  
rho =  
    1.6180
```

```
a = abs(3+4i)  
a =  
    5
```

```
z = sqrt(besselk(4/3,rho-i))  
z =  
    0.3730+ 0.3214i
```

```
huge = exp(log(realmax))  
huge =  
    1.7977e+308
```

```
toobig = pi*huge  
toobig =  
    Inf
```

输入命令

本节内容
“format 函数” (第 2-12 页)
“取消输出” (第 2-13 页)
“输入长语句” (第 2-13 页)
“命令行编辑” (第 2-13 页)

format 函数

`format` 函数控制所显示的值的数值格式。此函数仅影响数字显示方式，而不会影响 MATLAB 软件如何计算或保存数字。下面提供了不同格式及由向量 `x` 生成的最终输出，该向量的各个分量具有不同的量值。

注意 为了确保适当的间隔，请使用等宽字体，例如 Courier。

```
x = [4/3 1.2345e-6]

format short

1.3333 0.0000

format short e

1.3333e+000 1.2345e-006

format short g

1.3333 1.2345e-006

format long

1.33333333333333 0.00000123450000

format long e

1.33333333333333e+000 1.23450000000000e-006

format long g

1.33333333333333 1.2345e-006

format bank

1.33 0.00

format rat

4/3 1/810045

format hex

3ff5555555555555 3eb4b6231abfd271
```


如果矩阵的最大元素大于 10^3 或小于 10^{-3} ，MATLAB 会对短格式和长格式应用常用缩放因子。

除了上面显示的 `format` 函数，

`format compact`

会不显示在输出中出现的多个空行。这样，您可以在屏幕或窗口中查看更多信息。如果要进一步控制输出格式，请使用 `sprintf` 和 `fprintf` 函数。

取消输出

如果您在仅键入语句后按 **Return** 或 **Enter**，MATLAB 会在屏幕上自动显示结果。但是，如果使用分号结束行，MATLAB 会执行计算，但不会显示任何输出。当生成大型矩阵时，此功能尤其有用。例如，

```
A = magic(100);
```

输入长语句

如果语句无法容纳在一行中，请使用省略号（三个句点）`...`，后跟 **Return** 或 **Enter** 以指示该语句在下一行继续。例如，

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...  
    - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

=、+ 和 - 符号周围的空白是可选的，但可提高可读性。

命令行编辑

使用键盘上的各个箭头键和控制键可以重新调用、编辑和重用先前键入的语句。例如，假定您错误地输入了

```
rho = (1 + sqrt(5))/2
```

`sqrt` 的拼写不正确。MATLAB 会给出以下错误信息

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

您只需按 `↑` 键，而不必重新键入整行。系统将重新显示键入的语句。使用 `←` 键移动光标并插入缺少的 `r`。反复使用 `↑` 键可重新调用前面的行。键入几个字符并按 `↑` 键可查找前文中以这些字符开头行。还可以从命令历史记录中复制以前执行的语句。

索引

本节内容
“下标” (第 2-14 页)
“冒号运算符” (第 2-14 页)
“串联” (第 2-15 页)
“删除行和列” (第 2-16 页)
“标量扩展” (第 2-16 页)
“逻辑下标” (第 2-17 页)
“find 函数” (第 2-18 页)

下标

A 的行 i 和列 j 中的元素通过 A(i,j) 表示。例如，A(4,2) 表示第四行和第二列中的数字。在幻方矩阵中，A(4,2) 为 15。因此，要计算 A 第四列中的元素的总和，请键入

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

此下标生成

```
ans =  
    34
```

但这不是计算某列总和的最佳方法。

此外，还可以使用单一下标 A(k) 引用矩阵的元素。单一下标是引用行和列向量的常见方法。但是，也可以对满二维矩阵应用单一下标。在这种情况下，数组被视为一个由原始矩阵的列构成的长列向量。因此，在幻方矩阵中，A(8) 是另一种引用存储在 A(4,2) 中的值 15 的方法。

如果尝试使用矩阵外部元素的值，则会生成错误：

```
t = A(4,5)
```

索引超出矩阵维度。

相反，如果将值存储在矩阵外部元素中，则会增大大小以便容纳新元素：

```
X = A;  
X(4,5) = 17  
  
X =  
    16     3     2    13     0  
     5    10    11     8     0  
     9     6     7    12     0  
     4    15    14     1    17
```

冒号运算符

冒号 : 是最重要的 MATLAB 运算符之一。它以多种不同形式出现。表达式

```
1:10
```

是包含从 1 到 10 之间的整数的行向量：

```
1 2 3 4 5 6 7 8 9 10
```

要获取非单位间距，请指定增量。例如，

```
100:-7:50
```

为

```
100 93 86 79 72 65 58 51
```

而

```
0:pi/4:pi
```

为

```
0 0.7854 1.5708 2.3562 3.1416
```

包含冒号的下标表达式引用部分矩阵：

```
A(1:k,j)
```

表示 A 第 j 列中的前 k 个元素。因此，

```
sum(A(1:4,4))
```

计算第四列的总和。但是，执行此计算有一种更好的方法。冒号本身引用矩阵行或列中的所有元素，而关键字 **end** 引用最后一个行或列。因此，

```
sum(A(:,end))
```

计算 A 最后一列中的元素的总和：

```
ans =  
    34
```

为什么 4×4 幻方矩阵的幻数和等于 34？如果将介于 1 到 16 之间的整数分为四个总和相等的组，该总和必须为

```
sum(1:16)/4
```

当然，也即

```
ans =  
    34
```

串联

串联是连接小矩阵以便形成更大矩阵的过程。实际上，第一个矩阵是通过将其各个元素串联起来而构成的。成对的方括号 **[]** 即为串联运算符。例如，从 4×4 幻方矩阵 A 开始，组成

```
B = [A A+32; A+48 A+16]
```

结果会生成一个 8×8 矩阵，这是通过连接四个子矩阵获得的：

```
B =
```

```
16  3  2 13 48 35 34 45
 5 10 11  8 37 42 43 40
 9  6  7 12 41 38 39 44
 4 15 14  1 36 47 46 33
64 51 50 61 32 19 18 29
53 58 59 56 21 26 27 24
57 54 55 60 25 22 23 28
52 63 62 49 20 31 30 17
```

此矩阵是一个接近于幻方矩阵的矩阵。此矩阵的元素是经过重新排列的整数 1:64。此矩阵的列总和符合 8×8 幻方矩阵的要求：

```
sum(B)
```

```
ans =
    260    260    260    260    260    260    260    260
```

但是其行总和 `sum(B)'` 并不完全相同。要使其成为有效的 8×8 幻方矩阵，需要进行进一步操作。

删除行和列

只需使用一对方括号即可从矩阵中删除行和列。首先

```
X = A;
```

然后，要删除 X 的第二列，请使用

```
X(:,2) = []
```

这会将 X 更改为

```
X =
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

如果您删除矩阵中的单个元素，结果将不再是矩阵。因此，以下类似表达式

```
X(1,2) = []
```

将会导致错误。但是，使用单一下标可以删除一个元素或元素序列，并将其余元素重构为一个行向量。因此

```
X(2:2:10) = []
```

生成

```
X =
    16     9     2     7    13    12     1
```

标量扩展

可以采用多种不同方法将矩阵和标量合并在一起。例如，通过从每个元素中减去标量而将其从矩阵中减去。幻方矩阵的元素平均值为 8.5，因此

```
B = A - 8.5
```

形成一个列总和为零的矩阵：

```
B =
    7.5   -5.5   -6.5    4.5
   -3.5    1.5    2.5   -0.5
    0.5   -2.5   -1.5    3.5
   -4.5    6.5    5.5   -7.5
```

```
sum(B)
```

```
ans =
    0    0    0    0
```

通过标量扩展，MATLAB 会为范围中的所有索引分配一个指定标量。例如，

```
B(1:2,2:3) = 0
```

将 B 的某个部分清零：

```
B =
    7.5    0    0    4.5
   -3.5    0    0   -0.5
    0.5   -2.5   -1.5    3.5
   -4.5    6.5    5.5   -7.5
```

逻辑下标

根据逻辑和关系运算创建的逻辑向量可用于引用子数组。假定 X 是一个普通矩阵，L 是一个由某个逻辑运算生成的同等大小的矩阵。那么，X(L) 指定 X 的元素，其中 L 的元素为非零。

通过将逻辑运算指定为下标表达式，可以在一个步骤中完成这种下标。假定您具有以下数据集：

```
x = [2.1 1.7 1.6 1.5 NaN 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8];
```

NaN 是用于缺少的观测值的标记，例如，无法响应问卷中的某个项。要使用逻辑索引删除缺少的数据，请使用 `isfinite(x)`，对于所有有限数值，该函数为 true；对于 NaN 和 Inf，该函数为 false：

```
x = x(isfinite(x))
x =
    2.1 1.7 1.6 1.5 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8
```

现在，存在一个似乎与其他项很不一样的观测值，即 5.1。这是一个离群值。下面的语句可删除离群值，在本示例中，即比均值大三倍标准差的元素：

```
x = x(abs(x-mean(x)) <= 3*std(x))
x =
    2.1 1.7 1.6 1.5 1.9 1.8 1.5 1.8 1.4 2.2 1.6 1.8
```

标量扩展对于另一示例，请使用逻辑索引和标量扩展将非质数设置为 0，以便高亮显示丢勒幻方矩阵中的质数的位置。（请参阅“magic 函数”（第 2-5 页）。）

```
A(~isprime(A)) = 0
```

```
A =
    0    3    2   13
    5    0   11    0
    0    0    7    0
    0    0    0    0
```

find 函数

find 函数可用于确定与指定逻辑条件相符的数组元素的索引。**find** 以最简单的形式返回索引的列向量。转置该向量以便获取索引的行向量。例如，再次从丢勒的幻方矩阵开始。（请参阅“magic 函数”（第 2-5 页）。）

```
k = find(isprime(A))'
```

使用一维索引选取幻方矩阵中的质数的位置：

```
k =  
    2    5    9   10   11   13
```

使用以下命令按 **k** 确定的顺序将这些质数显示为行向量

```
A(k)
```

```
ans =  
    5    3    2   11    7   13
```

将 **k** 用作赋值语句的左侧索引时，会保留矩阵结构：

```
A(k) = NaN
```

```
A =  
    16 NaN NaN NaN  
NaN    10 NaN    8  
    9    6 NaN   12  
    4   15   14    1
```

数组类型

本节内容
“多维数组” (第 2-19 页)
“元胞数组” (第 2-20 页)
“字符与文本” (第 2-22 页)
“结构体” (第 2-23 页)

多维数组

MATLAB 环境中的多维数组是具有多个下标的数组。创建多维数组的一种方法是调用具有多个参数的 `zeros`、`ones`、`rand` 或 `randn`。例如，

```
R = randn(3,4,5);
```

创建一个 3×4×5 数组，共包含 $3 \times 4 \times 5 = 60$ 个正态分布的随机元素。

三维数组可表示在矩形网格中采样的三维物理数据，例如室内温度。或者也可以表示矩阵序列 $A^{(k)}$ 或与时间相关的矩阵示例 $A(t)$ 。在下面的示例中，第 k 个或第 t_k 个矩阵的第 (i, j) 个元素由 $A(i,j,k)$ 表示。

MATLAB 与丢勒的 4 阶幻方矩阵版本的区别在于交换了两个列。通过交换列，可以生成许多不同的幻方矩阵。语句

```
p = perms(1:4);
```

生成 $4! = 24$ 置换 1:4。第 k 个置换为行向量 $p(k,:)$ 。然后，

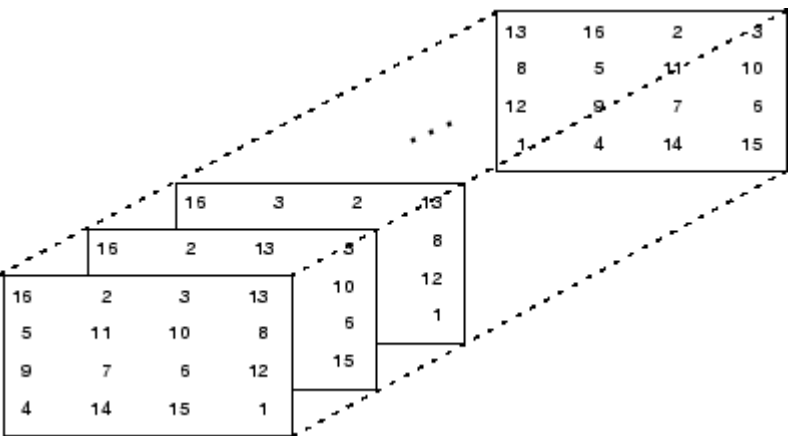
```
A = magic(4);
M = zeros(4,4,24);

for k = 1:24
    M(:, :, k) = A(:, p(k,:));
end
```

将含有 24 个幻方矩阵的序列存储在三维数组 M 中。 M 大小为

```
size(M)
```

```
ans =
     4     4    24
```



注意 此插图中显示的矩阵顺序可能不同于您的结果。`perms` 函数始终返回输入向量的所有置换，但置换顺序可能因不同 MATLAB 版本而异。

语句

```
sum(M,d)
```

通过改变第 `d` 个下标来计算总和。因此

```
sum(M,1)
```

是一个含有 24 个行向量副本的 $1 \times 4 \times 24$ 数组

```
34  34  34  34
```

而

```
sum(M,2)
```

是一个含有 24 个列向量副本的 $4 \times 1 \times 24$ 数组

```
34
34
34
34
```

最后，

```
S = sum(M,3)
```

在序列中添加 24 个矩阵。结果的大小为 $4 \times 4 \times 1$ ，因此它看似是 4×4 数组：

```
S =
    204    204    204    204
    204    204    204    204
    204    204    204    204
    204    204    204    204
```

元胞数组

MATLAB 中的元胞数组是以其他数组的副本为元素的多维数组。使用 `cell` 函数可以创建空矩阵的元胞数组。但是，更普遍的做法是，通过将其他内容的集合括入花括号 `{}` 中来创建元胞数组。花括号还可以与下标配合使用，以便访问各个元胞的内容。例如，

```
C = {A sum(A) prod(prod(A))}
```

生成一个 1×3 元胞数组。这三个元胞包含幻方矩阵、列总和的行向量及其所有元素的乘积。当显示 `C` 时，您会看到

```
C =
    [4x4 double]    [1x4 double]    [20922789888000]
```

这是因为前两个元胞太大，无法在此有限空间中输出，但第三个元胞仅包含 $16!$ 一个数字，因此有空间可以输出此数字。

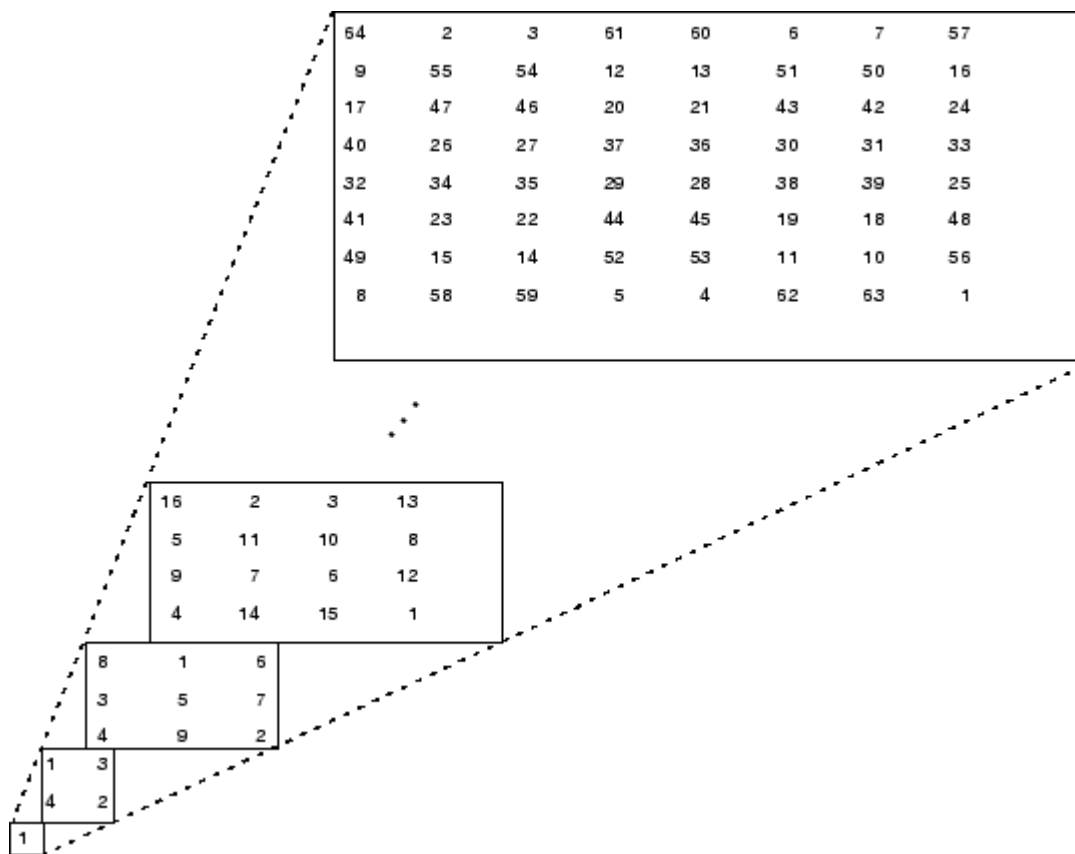
请牢记以下两大要点。第一，要检索某个元胞的内容，请在花括号中使用下标。例如，`C{1}` 检索幻方矩阵，`C{3}` 为 16!。第二，元胞数组包含其他数组的副本，而不包含指向这些数组的指针。如果随后更改 `A`，`C` 不会发生变化。

使用三维数组可以存储相同大小的矩阵序列。元胞数组可用于存储不同大小的矩阵序列。例如，

```
M = cell(8,1);
for n = 1:8
    M{n} = magic(n);
end
M
```

生成具有不同顺序的幻方矩阵序列：

```
M =
[      1]
[ 2x2 double]
[ 3x3 double]
[ 4x4 double]
[ 5x5 double]
[ 6x6 double]
[ 7x7 double]
[ 8x8 double]
```



使用以下命令可以检索 4×4 幻方矩阵

```
M{4}
```

字符与文本

使用单引号在 MATLAB 中输入文本。例如，

```
s = 'Hello'
```

其结果与您目前已学过的数值矩阵或数组不属于同一类型，而是一个 1×5 字符数组。

字符在内部作为数字存储，而不会采用浮点格式存储。语句

```
a = double(s)
```

将字符数组转换为数值矩阵，该矩阵包含每个字符的 ASCII 代码的浮点表示。结果为

```
a =  
    72    101    108    108    111
```

语句

```
s = char(a)
```

是刚才转换的逆转换。

将数字转换为字符可以调查计算机上的各种可用字体。基本 ASCII 字符集中的可打印字符由整数 32:127 表示。（小于 32 的整数表示不可打印的控制字符。）这些整数使用以下语句按相应的 6×16 数组的形式排列

```
F = reshape(32:127,16,6)';
```

扩展 ASCII 字符集中的可打印字符由 F+128 表示。将这些整数解释为字符时，结果取决于当前使用的字体。键入语句

```
char(F)  
char(F+128)
```

然后改变命令行窗口所用的字体。要更改字体，请在**主页**选项卡上的**环境**部分中，点击**预设 > 字体**。如果代码行中包含制表符，请使用等宽字体（例如，**Monospaced**）以便在不同行中对齐制表符位置。

使用方括号进行串联可将文本变量联接到一起。语句

```
h = [s, ' world']
```

水平联接字符并生成

```
h =  
Hello world
```

语句

```
v = [s; 'world']
```

垂直联接字符并生成

```
v =  
Hello  
world
```

请注意，必须在 h 中的 'w' 前插入一个空格，并且 v 中的两个单词的长度必须相同。生成的数组均为字符数组；h 为 1×11，v 为 2×5。

要操作包含不同长度的行的文本主体，您有两种选择，即使用填充的字符数组或使用字符向量元胞数组。创建字符数组时，数组各行的长度必须相同。（使用空格填充较短行的末尾。）`char` 函数可执行这种填充操作。例如，

```
S = char('A','rolling','stone','gathers','momentum.')
```

生成一个 5×9 字符数组：

```
S =
A
rolling
stone
gathers
momentum.
```

再者，您也可以将文本存储在元胞数组中。例如，

```
C = {'A';'rolling';'stone';'gathers';'momentum.'}
```

创建一个不需要任何填充的 5×1 元胞数组，因为该数组的各行可以具有不同的长度：

```
C =
'A'
'rolling'
'stone'
'gathers'
'momentum.'
```

使用以下语句可以将填充后的字符数组转换为字符向量元胞数组：

```
C = cellstr(S)
```

使用以下语句可以逆转此过程

```
S = char(C)
```

结构体

结构体是多维 MATLAB 数组，包含可按文本字段标志符访问的元素。例如，

```
S.name = 'Ed Plum';
S.score = 83;
S.grade = 'B+'
```

创建一个具有三个字段的标量结构体：

```
S =
    name: 'Ed Plum'
    score: 83
    grade: 'B+'
```

与 MATLAB 环境中的所有其他内容一样，结构体也为数组，因此可以插入其他元素。在本示例中，数组的每个元素都是一个具有若干字段的结构体。可以一次添加一个字段，

```
S(2).name = 'Toni Miller';
S(2).score = 91;
S(2).grade = 'A-';
```

也可以使用一个语句添加整个元素：

```
S(3) = struct('name','Jerry Garcia',...  
            'score',70,'grade','C')
```

现在，结构体非常大以致仅输出摘要：

```
S =  
1x3 struct array with fields:  
    name  
    score  
    grade
```

将不同字段重新组合为其他 MATLAB 数组的方法有许多种。这些方法大多基于逗号分隔列表的表示法。键入

```
S.score
```

与键入

```
S(1).score, S(2).score, S(3).score
```

相同，这是一个逗号分隔列表。

如果将生成此类列表的表达式括在方括号中，MATLAB 会将该列表中的每一项都存储在数组中。在本示例中，MATLAB 创建一个数值行向量，该向量包含结构体数组 S 的每个元素的 **score** 字段：

```
scores = [S.score]  
scores =  
    83    91    70  
  
avg_score = sum(scores)/length(scores)  
avg_score =  
    81.3333
```

要根据某个文本字段（例如，**name**）创建字符数组，请对 **S.name** 生成的逗号分隔列表调用 **char** 函数：

```
names = char(S.name)  
names =  
    Ed Plum  
    Toni Miller  
    Jerry Garcia
```

同样，通过将生成列表的表达式括入花括号中，可以根据 **name** 字段创建元胞数组：

```
names = {S.name}  
names =  
    'Ed Plum'    'Toni Miller'    'Jerry Garcia'
```

要将结构体数组的每个元素的字段赋值给结构体外部的单独变量，请指定等号左侧的每个输出，并将其全部括在方括号中：

```
[N1 N2 N3] = S.name  
N1 =  
    Ed Plum  
N2 =  
    Toni Miller
```

```
N3 =
    Jerry Garcia
```

动态字段名称

访问结构体中的数据的最常用方法是指定要引用的字段的名称。访问结构体数据的另一种方法是使用动态字段名称。这些名称将字段表示为变量表达式，MATLAB 会在运行时计算这些表达式。此处显示的点-括号语法将 **expression** 作为动态字段名称：

```
structName.(expression)
```

使用标准 MATLAB 索引语法创建此字段的索引。例如，要在字段名称中计算 **expression**，并在行 7 中的 1 至 25 列内获取该字段的值，请使用

```
structName.(expression)(7,1:25)
```

动态字段名称示例

下面显示的 **avgscore** 函数可用于计算考试的平均分数，并使用动态字段名称检索 **testscores** 结构体中的信息：

```
function avg = avgscore(testscores, student, first, last)
for k = first:last
    scores(k) = testscores.(student).week(k);
end
avg = sum(scores)/(last - first + 1);
```

您可以运行此函数，并对动态字段 **student** 使用不同值。首先，对包含 25 周内的分数的结构体进行初始化：

```
testscores.Ann_Lane.week(1:25) = ...
    [95 89 76 82 79 92 94 92 89 81 75 93 ...
     85 84 83 86 85 90 82 82 84 79 96 88 98];

testscores.William_King.week(1:25) = ...
    [87 80 91 84 99 87 93 87 97 87 82 89 ...
     86 82 90 98 75 79 92 84 90 93 84 78 81];
```

现在，运行 **avgscore**，并在运行时使用动态字段名称为 **testscores** 结构体提供学生姓名字段：

```
avgscore(testscores, 'Ann_Lane', 7, 22)
ans =
    85.2500

avgscore(testscores, 'William_King', 7, 22)
ans =
    87.7500
```


数学

- “线性代数” (第 3-2 页)
- “非线性函数的运算” (第 3-31 页)
- “多变量数据” (第 3-34 页)
- “数据分析” (第 3-35 页)

线性代数

本节内容
“MATLAB 环境中的矩阵” (第 3-2 页)
“线性方程组” (第 3-9 页)
“分解” (第 3-17 页)
“幂和指数” (第 3-22 页)
“特征值” (第 3-25 页)
“奇异值” (第 3-28 页)

MATLAB 环境中的矩阵

本主题介绍如何在 MATLAB 中创建矩阵和执行基本矩阵计算。

MATLAB 环境使用术语矩阵来表示包含以二维网格排列的实数或复数的变量。更广泛而言，数组为向量、矩阵或更高维度的数值网格。MATLAB 中的所有数组都是矩形，在这种意义上沿任何维度的分量向量的长度均相同。矩阵中定义的数学运算是线性代数的主题。

创建矩阵

MATLAB 提供了许多函数，用于创建各种类型的矩阵。例如，您可以使用基于帕斯卡三角形的项创建一个对称矩阵：

```
A = pascal(3)

A =
     1     1     1
     1     2     3
     1     3     6
```

您也可以创建一个非对称幻方矩阵，它的行总和与列总和相等：

```
B = magic(3)

B =
     8     1     6
     3     5     7
     4     9     2
```

另一个示例是由随机整数构成的 3×2 矩形矩阵：在这种情况下，randi 的第一个输入描述整数可能值的范围，后面两个输入描述行和列的数量。

```
C = randi(10,3,2)

C =
     9    10
    10     7
     2     1
```

列向量为 m×1 矩阵，行向量为 1×n 矩阵，标量为 1×1 矩阵。要手动定义矩阵，请使用方括号 [] 来表示数组的开始和结束。在括号内，使用分号 ; 表示行的结尾。在标量（1×1 矩阵）的情况下，括号不是必需的。例如，以下语句生成一个列向量、一个行向量和一个标量：


```
u = [3; 1; 4]
```

```
v = [2 0 -1]
```

```
s = 7
```

```
u =
    3
    1
    4
```

```
v =
    2    0   -1
```

```
s =
    7
```

有关创建和处理矩阵的详细信息，请参阅“创建、串联和扩展矩阵”。

矩阵的加法和减法

矩阵和数组的加减法是逐个元素执行的，或者说是按元素执行的。例如，A 加 B 之后再减去 A 又可以得到 B：

```
X = A + B
```

```
X =
    9    2    7
    4    7   10
    5   12    8
```

```
Y = X - A
```

```
Y =
    8    1    6
    3    5    7
    4    9    2
```

加法和减法要求两个矩阵具有兼容的维度。如果维度不兼容，将会导致错误：

```
X = A + C
```

```
Error using +
Matrix dimensions must agree.
```

有关详细信息，请参阅“数组与矩阵运算”。

向量乘积和转置

长度相同的行向量和列向量可以按任一顺序相乘。其结果是一个标量（称为内积）或一个矩阵（称为外积）：

```
u = [3; 1; 4];
v = [2 0 -1];
x = v*u
```

```
x =
    2
```

```
X = u*v
```

```
X =
```

```

6   0  -3
2   0  -1
8   0  -4
```

对于实矩阵，转置运算对 a_{ij} 和 a_{ji} 进行交换。对于复矩阵，还要考虑是否用数组中复数项的复共轭来形成复共轭转置。MATLAB 使用撇号运算符 (') 执行复共轭转置，使用点撇号运算符 (.') 执行无共轭的转置。对于包含所有实数元素的矩阵，这两个运算符返回相同结果。

示例矩阵 **A = pascal(3)** 是对称的，因此 **A'** 等于 **A**。然而，**B = magic(3)** 不是对称的，因此 **B'** 的元素是 **B** 的元素沿主对角线反转之后的结果：

```
B = magic(3)
```

```
B =
```

```

8   1   6
3   5   7
4   9   2
```

```
X = B'
```

```
X =
```

```

8   3   4
1   5   9
6   7   2
```

对于向量，转置会将行向量变为列向量（反之亦然）：

```
x = v'
```

```
x =
```

```

2
0
-1
```

如果 **x** 和 **y** 均为实数列向量，则乘积 **x*y** 不确定，但以下两个乘积

```
x'*y
```

和

```
y'*x
```

产生相同的标量结果。此参数使用很频繁，它有三个不同的名称内积、标量积或点积。甚至还有一个专门的点积函数，称为 **dot**。

对于复数向量或矩阵 **z**，参量 **z'** 不仅可转置该向量或矩阵，而且可将每个复数元素转换为其复共轭数。也就是说，每个复数元素的虚部的正负号将会发生更改。以如下复矩阵为例：

```
z = [1+2i 7-3i 3+4i; 6-2i 9i 4+7i]
```

```
z =
```

```
1.0000 + 2.0000i  7.0000 - 3.0000i  3.0000 + 4.0000i
6.0000 - 2.0000i  0.0000 + 9.0000i  4.0000 + 7.0000i
```

z 的复共轭转置为：

```
z'
```

```
ans =
```

```
1.0000 - 2.0000i  6.0000 + 2.0000i
7.0000 + 3.0000i  0.0000 - 9.0000i
3.0000 - 4.0000i  4.0000 - 7.0000i
```

非共轭复数转置（其中每个元素的复数部分保留其符号）表示为 $z.'$ ：

```
z.'
```

```
ans =
```

```
1.0000 + 2.0000i  6.0000 - 2.0000i
7.0000 - 3.0000i  0.0000 + 9.0000i
3.0000 + 4.0000i  4.0000 + 7.0000i
```

对于复数向量，两个标量积 $x'*y$ 和 $y'*x$ 互为复共轭数，而复数向量与其自身的标量积 $x'*x$ 为实数。

矩阵乘法

矩阵乘法是以这样一种方式定义的：反映底层线性变换的构成，并允许紧凑表示联立线性方程组。如果 A 的列维度等于 B 的行维度，或者其中一个矩阵为标量，则可定义矩阵乘积 $C = AB$ 。如果 A 为 $m \times p$ 且 B 为 $p \times n$ ，则二者的乘积 C 为 $m \times n$ 。该乘积实际上可以使用 MATLAB `for` 循环、`colon` 表示法和向量点积进行定义：

```
A = pascal(3);
B = magic(3);
m = 3;
n = 3;
for i = 1:m
    for j = 1:n
        C(i,j) = A(i,:)*B(:,j);
    end
end
```

MATLAB 使用星号表示矩阵乘法，如 $C = A*B$ 中所示。矩阵乘法不适用交换律；即 $A*B$ 通常不等于 $B*A$ ：

```
X = A*B
```

```
X =
```

```
15  15  15
26  38  26
41  70  39
```

```
Y = B*A
```

```
Y =
```

```
15  28  47
15  34  60
15  28  43
```

矩阵可以在右侧乘以列向量，在左侧乘以行向量：

```
u = [3; 1; 4];
x = A*u
```

```
x =
```

```
8
17
30
```

```
v = [2 0 -1];
y = v*B
```

```
y =
```

```
12 -7 10
```

矩形矩阵乘法必须满足维度兼容性条件：由于 A 是 3×3 矩阵， C 是 3×2 矩阵，因此可将二者相乘得到 3×2 结果（共同的内部维度会消去）：

```
X = A*C
```

```
X =
```

```
24 17
47 42
79 77
```

但是，乘法不能以相反的顺序执行：

```
Y = C*A
```

```
Error using *
Incorrect dimensions for matrix multiplication. Check that the number of columns
in the first matrix matches the number of rows in the second matrix. To perform
elementwise multiplication, use '.*'.
```

您可以将任何内容与标量相乘：

```
s = 10;
w = s*y
```

```
w =
```

```
120 -70 100
```

当您将数组与标量相乘时，标量将隐式扩展为与另一输入相同的大小。这通常称为标量扩展。

单位矩阵

普遍接受的数学表示法使用大写字母 I 来表示单位矩阵，即主对角线元素为 1 且其他位置元素为 0 的各种大小的矩阵。这些矩阵具有以下属性：无论维度是否兼容， $AI = A$ 和 $IA = A$ 。

原始版本的 MATLAB 不能将 I 用于此用途，因为它不会区分大小字母和小写字母，并且 i 已用作下标和复数单位。因此，引入了英语双关语。函数

```
eye(m,n)
```

返回 $m \times n$ 矩形单位矩阵，`eye(n)` 返回 $n \times n$ 单位方阵。

矩阵求逆

如果矩阵 A 为非奇异方阵（非零行列式），则方程 $AX = I$ 和 $XA = I$ 具有相同的解 X 。此解称为 A 的逆矩阵，表示为 A^{-1} 。`inv` 函数和表达式 A^{-1} 均可对矩阵求逆。

```
A = pascal(3)
```

```
A =
     1     1     1
     1     2     3
     1     3     6
```

```
X = inv(A)
```

```
X =
     3.0000    -3.0000     1.0000
    -3.0000     5.0000    -2.0000
     1.0000    -2.0000     1.0000
```

```
A*X
```

```
ans =
     1.0000         0         0
     0.0000     1.0000    -0.0000
    -0.0000     0.0000     1.0000
```

通过 `det` 计算的行列式表示由矩阵描述的线性变换的缩放因子。当行列式正好为零时，矩阵为奇异矩阵，因此不存在逆矩阵。

```
d = det(A)
```

```
d =
```

```
1
```

有些矩阵接近奇异矩阵，虽然存在逆矩阵，但计算容易出现数值误差。`cond` 函数计算逆运算的条件数，它指示矩阵求逆结果的精度。条件数的范围是从 1（数值稳定的矩阵）到 `Inf`（奇异矩阵）。

```
c = cond(A)
```

```
c =
```

```
61.9839
```

很少需要为某个矩阵构造显式逆矩阵。求解线性方程组 $Ax = b$ 时，常常会误用 `inv`。从执行时间和数值精度方面而言，求解此方程的最佳方法是使用矩阵反斜杠运算符，即 $x = A \backslash b$ 。有关详细信息，请参阅 `mldivide`。

Kronecker 张量积

两个矩阵的 Kronecker 乘积 `kron(X,Y)` 为 X 的元素与 Y 的元素的所有可能乘积构成的较大矩阵。如果 X 为 $m \times n$ 且 Y 为 $p \times q$ ，则 `kron(X,Y)` 为 $mp \times nq$ 。元素以特定方式排列，呈现 X 的每个元素分别与整个矩阵 Y 相乘的结果。

```
[X(1,1)*Y X(1,2)*Y ... X(1,n)*Y
 ...
 X(m,1)*Y X(m,2)*Y ... X(m,n)*Y]
```

Kronecker 乘积通常与元素为 0 和 1 的矩阵配合使用，以构建小型矩阵的重复副本。例如，如果 X 为 2×2 矩阵

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

且 $I = \text{eye}(2,2)$ 为 2×2 单位矩阵，则：

`kron(X,I)`

`ans =`

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$$

并且

`kron(I,X)`

`ans =`

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

除了 `kron` 之外，对复制数组有用的其他函数还有 `repmat`、`repelem` 和 `blkdiag`。

向量范数和矩阵范数

向量 x 的 p -范数，

$$\|x\|_p = \left(\sum |x_i|^p \right)^{1/p},$$

使用 `norm(x,p)` 进行计算。此运算是为 $p > 1$ 的任意值定义的，但最常见的 p 值为 1、2 和 ∞ 。默认值为 $p = 2$ ，这与欧几里德长度或向量幅值对应：

`v = [2 0 -1];`

`[norm(v,1) norm(v) norm(v,inf)]`

`ans =`

3.0000 2.2361 2.0000

矩阵 A 的 p -范数，

$$\|A\|_p = \max_x \frac{\|Ax\|_p}{\|x\|_p},$$

可以针对 $p = 1$ 、2 和 ∞ 通过 `norm(A,p)` 进行计算。同样，默认值也为 $p = 2$ ：

`A = pascal(3);`

`[norm(A,1) norm(A) norm(A,inf)]`

```
ans =  
10.0000  7.8730 10.0000
```

如果要计算矩阵的每行或每列的范数，可以使用 `vecnorm`：

```
vecnorm(A)  
ans =  
1.7321  3.7417  6.7823
```

使用线性代数方程函数的多线程计算

对于许多线性代数函数和按元素的数值函数，MATLAB 软件支持多线程计算。这些函数将自动在多个线程上执行。要使函数或表达式在多个 CPU 上更快地执行，必须满足许多条件：

- 1 函数执行的运算可轻松划分为并发执行的多个部分。这些部分必须能够在进程之间几乎不通信的情况下执行。它们应需要很少的序列运算。
- 2 数据大小足以使并发执行的任何优势在重要性方面超过对数据分区和管理各个执行线程所需的时间。例如，仅当数组包含数千个或以上的元素时，大多数函数才会加速。
- 3 运算未与内存绑定；处理时间不受内存访问时间控制。一般而言，复杂函数比简单函数速度更快。

对于大型双精度数组（约 10,000 个元素），矩阵乘法 ($\mathbf{X}*\mathbf{Y}$) 和矩阵乘幂 ($\mathbf{X}^{\mathbf{p}}$) 运算符会大幅增加速度。矩阵分析函数 `det`、`rcond`、`hess` 和 `expm` 也会对大型双精度数组大幅增加速度。

线性方程组

- “计算注意事项”（第 3-9 页）
- “通解”（第 3-10 页）
- “方阵方程组”（第 3-10 页）
- “超定方程组”（第 3-12 页）
- “欠定方程组”（第 3-14 页）
- “多右端线性方程组的求解”（第 3-16 页）
- “迭代法”（第 3-17 页）
- “多线程计算”（第 3-17 页）

计算注意事项

进行科学计算时，最重要的一个问题是对联立线性方程组求解。

在矩阵表示法中，常见问题采用以下形式：给定两个矩阵 \mathbf{A} 和 \mathbf{b} ，是否存在一个唯一矩阵 \mathbf{x} 使 $\mathbf{Ax} = \mathbf{b}$ 或 $\mathbf{x}\mathbf{A} = \mathbf{b}$ ？

考虑 1×1 示例具有指导意义。例如，方程

$$7x = 21$$

是否具有唯一解？

答案当然是肯定的。方程有唯一解 $x = 3$ 。通过除法很容易求得该解：

$$x = \frac{21}{7} = 3。$$

该解通常不是通过计算 7 的倒数求得的，即先计算 $7^{-1} = 0.142857\dots$ ，然后将 7^{-1} 乘以 21。这将需要更多的工作，而且如果 7^{-1} 以有限位数表示时，准确性会较低。类似注意事项也适用于多个未知数的线性方程组；MATLAB 在解此类方程时不会计算矩阵的逆。

尽管这不是标准的数学表示法，但 MATLAB 使用标量示例中常见的除法术语来描述常规联立方程组的解。斜杠 / 和反斜杠 \ 这两个除号分别对应 MATLAB 函数 `mrdivide` 和 `ldivide`。两种运算符分别用于未知矩阵出现在系数矩阵左侧或右侧的情况：

$x = b/A$	表示使用 <code>mrdivide</code> 获得的矩阵方程 $xA = b$ 的解。
$x = A \backslash b$	表示使用 <code>ldivide</code> 获得的矩阵方程 $Ax = b$ 的解。

考虑将方程 $Ax = b$ 或 $xA = b$ 的两端“除以” A 。系数矩阵 A 始终位于“分母”中。

$x = A \backslash b$ 的维度兼容性条件要求两个矩阵 A 和 b 的行数相同。这样，解 x 的列数便与 b 的列数相同，并且其行维度等于 A 的列维度。对于 $x = b/A$ ，行和列的角色将会互换。

实际上， $Ax=b$ 形式的线性方程组比 $xA=b$ 形式的线性方程组更常见。因此，反斜杠的使用频率要远高于斜杠的使用频率。本节其余部分将重点介绍反斜杠运算符；斜杠运算符的对应属性可以从以下恒等式推知：

$$(b/A)' = (A' \backslash b').$$

系数矩阵 A 不需要是方阵。如果 A 的大小为 $m \times n$ ，则有三种情况：

$m = n$	方阵方程组。求精确解。
$m > n$	超定方程组，即方程个数多于未知数个数。求最小二乘解。
$m < n$	欠定方程组，即方程个数少于未知数个数。使用最多 m 个非零分量求基本解。

ldivide 算法

`ldivide` 运算符使用不同的求解器来处理不同类型的系数矩阵。通过检查系数矩阵自动诊断各种情况。有关详细信息，请参阅 `ldivide` 参考页的“算法”小节。

通解

线性方程组 $Ax = b$ 的通解描述了所有可能的解。可以通过以下方法求通解：

- 1 求对应的齐次方程组 $Ax = 0$ 的解。使用 `null` 命令通过键入 `null(A)` 来执行此操作。这会将解空间的基向量恢复为 $Ax = 0$ 。任何解都是基向量的线性组合。
- 2 求非齐次方程组 $Ax = b$ 的特定解。

然后，可将 $Ax = b$ 的任何解写成第 2 步中的 $Ax = b$ 的特定解加上第 1 步中的基向量的线性组合之和。

本节其余部分将介绍如何使用 MATLAB 求 $Ax = b$ 的特定解，如第 2 步中所述。

方阵方程组

最常见的情况涉及到一个方阵系数矩阵 A 和一个右侧单列向量 b 。

非奇异系数矩阵

如果矩阵 A 是非奇异矩阵，则解 $x = A \backslash b$ 的大小与 b 的大小相同。例如：

```
A = pascal(3);
u = [3; 1; 4];
```



```
x = A\u
```

```
x =
    10
   -12
     5
```

可以确认 $A*x$ 恰好等于 u 。

如果 A 和 b 为方阵并且大小相同，则 $x = A \setminus b$ 也具有相同大小：

```
b = magic(3);
X = A\b
```

```
X =
    19    -3    -1
   -17     4    13
     6     0    -6
```

可以确认 $A*x$ 恰好等于 b 。

以上两个示例具有确切的整数解。这是因为系数矩阵选为 `pascal(3)`，这是满秩矩阵（非奇异的）。

奇异系数矩阵

如果方阵 A 不包含线性无关的列，则该矩阵为奇异矩阵。如果 A 为奇异矩阵，则 $Ax = b$ 的解将不存在或不唯一。如果 A 接近奇异或检测到完全奇异性，则反斜杠运算符 $A \setminus b$ 会发出警告。

如果 A 为奇异矩阵并且 $Ax = b$ 具有解，可以通过键入以下内容求不是唯一的特定解

```
P = pinv(A)*b
```

`pinv(A)` 是 A 的伪逆。如果 $Ax = b$ 没有精确解，则 `pinv(A)` 将返回最小二乘解。

例如：

```
A = [ 1   3   7
      -1   4   4
       1  10  18]
```

为奇异矩阵，可以通过键入以下内容进行验证：

```
rank(A)
```

```
ans =
```

```
2
```

由于 A 不是满秩，它有一些等于零的奇异值。

精确解。对于 $b = [5; 2; 12]$ ，方程 $Ax = b$ 具有精确解，给定

```
pinv(A)*b
```

```
ans =
    0.3850
   -0.1103
    0.7066
```

通过键入以下内容验证 $\text{pinv}(A)*b$ 是否为精确解

```
A*pinv(A)*b
```

```
ans =
    5.0000
    2.0000
   12.0000
```

最小二乘解。但是，如果 $b = [3;6;0]$ ，则 $Ax = b$ 没有精确解。在这种情况下， $\text{pinv}(A)*b$ 会返回最小二乘解。键入

```
A*pinv(A)*b
```

```
ans =
   -1.0000
    4.0000
    2.0000
```

则不会返回原始向量 b 。

通过得到增广矩阵 $[A \ b]$ 的简化行阶梯形式，可以确定 $Ax = b$ 是否具有精确解。为此，对于此示例请输入

```
rref([A b])
ans =
    1.0000     0    2.2857     0
         0    1.0000    1.5714     0
         0     0     0  1.0000
```

由于最下面一行全部为零（最后一项除外），因此该方程无解。在这种情况下， $\text{pinv}(A)$ 会返回最小二乘解。

超定方程组

此示例说明在对试验数据的各种曲线拟合中通常会如何遇到超定方程组。

在多个不同的时间值 t 对数量 y 进行测量以生成以下观测值。可以使用以下语句输入数据并在表中查看该数据。

```
t = [0 .3 .8 1.1 1.6 2.3]';
y = [.82 .72 .63 .60 .55 .50]';
B = table(t,y)
```

```
B=6×2 table
```

t	y
0	0.82
0.3	0.72
0.8	0.63
1.1	0.6
1.6	0.55
2.3	0.5

尝试使用指数衰减函数对数据进行建模

$$y(t) = c_1 + c_2 e^{-t}.$$

上一方程表明，向量 y 应由两个其他向量的线性组合来逼近。一个是元素全部为 1 的常向量，另一个是带有分量 $\exp(-t)$ 的向量。未知系数 c_1 和 c_2 可以通过执行最小二乘拟合来计算，这样会最大限度地减小数据与模型偏差的平方和。在两个未知系数的情况下有六个方程，用 6×2 矩阵表示。

```
E = [ones(size(t)) exp(-t)]
```

```
E = 6×2
```

```
1.0000 1.0000
1.0000 0.7408
1.0000 0.4493
1.0000 0.3329
1.0000 0.2019
1.0000 0.1003
```

使用反斜杠运算符获取最小二乘解。

```
c = E \ y
```

```
c = 2×1
```

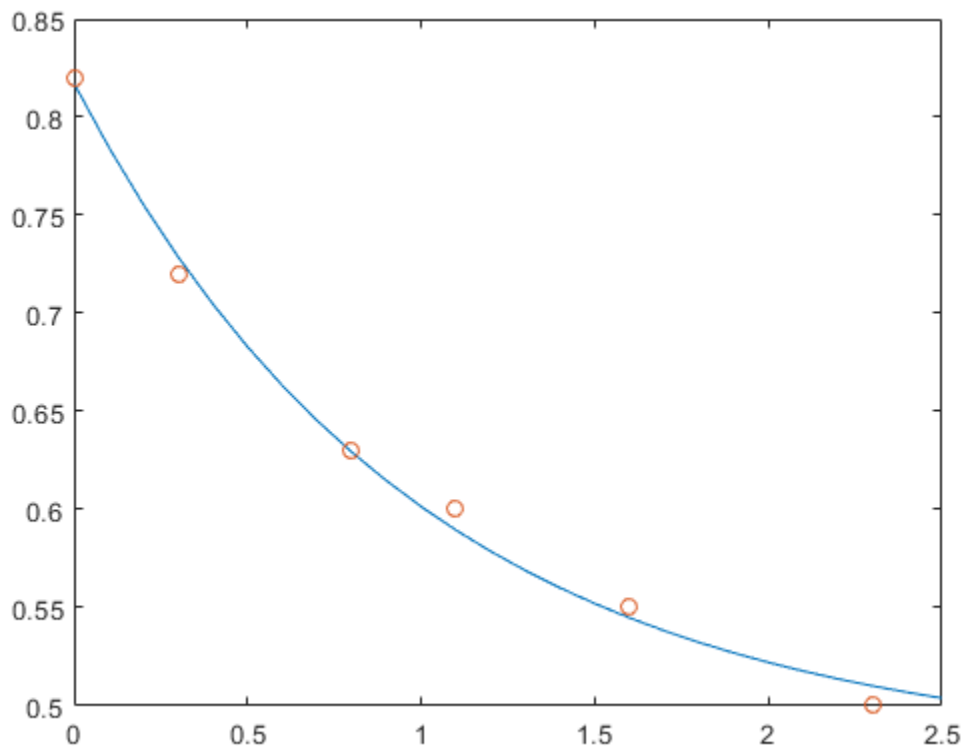
```
0.4760
0.3413
```

也就是说，对数据的最小二乘拟合为

$$y(t) = 0.4760 + 0.3413e^{-t}.$$

以下语句按固定间隔的 t 增量为模型求值，然后与原始数据一同绘制结果：

```
T = (0:0.1:2.5)';
Y = [ones(size(T)) exp(-T)]*c;
plot(T,Y,'-t,y','o')
```



$E \cdot c$ 与 y 不完全相等，但差值可能远小于原始数据中的测量误差。

如果矩形矩阵 A 没有线性无关的列，则该矩阵秩亏。如果 A 秩亏，则 $AX = B$ 的最小二乘解不唯一。如果 A 秩亏，则 $A \setminus B$ 会发出警告，并生成一个最小二乘解。您可以使用 `lsqminnorm` 求在所有解中具有最小范数的解 X 。

欠定方程组

本例演示了欠定方程组的解不唯一的情况。欠定线性方程组包含的未知数比方程多。MATLAB 矩阵左除运算求基本最小二乘解，对于 $m \times n$ 系数矩阵，它最多有 m 个非零分量。

以下是一个简单的随机示例：

```
R = [6 8 7 3; 3 5 4 1]
```

```
rng(0);
```

```
b = randi(8,2,1)
```

R =

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

b =

```
7
8
```

线性方程组 $\mathbf{R}\mathbf{p} = \mathbf{b}$ 有两个方程，四个未知数。由于系数矩阵包含较小的整数，因此适合使用 `format` 命令以有理格式显示解。通过以下命令可获取特定解

```
format rat
p = R\b
```

```
p =
    0
   17/7
    0
  -29/7
```

其中一个非零分量为 $\mathbf{p}(2)$ ，因为 $\mathbf{R}(:,2)$ 是具有最大范数的 \mathbf{R} 的列。另一个非零分量为 $\mathbf{p}(4)$ ，因为 $\mathbf{R}(:,4)$ 在消除 $\mathbf{R}(:,2)$ 后起控制作用。

欠定方程组的完全通解可以通过 \mathbf{p} 加上任意零空间向量线性组合来表示，可以使用 `null` 函数（使用请求有理基的选项）计算该空间向量。

```
Z = null(R,'r')
```

```
Z =
   -1/2    -7/6
   -1/2     1/2
    1         0
    0         1
```

可以确认 $\mathbf{R}*\mathbf{Z}$ 为零，并且残差 $\mathbf{R}*\mathbf{x} - \mathbf{b}$ 远远小于任一向量 \mathbf{x} （其中

```
 $\mathbf{x} = \mathbf{p} + \mathbf{Z}*\mathbf{q}$ 
```

由于 \mathbf{Z} 的列是零空间向量，因此 $\mathbf{Z}*\mathbf{q}$ 是以下向量的线性组合：

$$\mathbf{Z}\mathbf{q} = \begin{pmatrix} \vec{x}_1 & \vec{x}_2 \end{pmatrix} \begin{pmatrix} u \\ w \end{pmatrix} = u\vec{x}_1 + w\vec{x}_2 .$$

为了说明这一点，选择任意 \mathbf{q} 并构造 \mathbf{x} 。

```
q = [-2; 1];
x = p + Z*q;
```

计算残差的范数。

```
format short
norm(R*x - b)
```

```
ans =
    2.6645e-15
```

如果有无限多个解，则最小范数解具有特别意义。您可以使用 `lsqminnorm` 计算最小范数最小二乘解。该解具有 `norm(p)` 的最小可能值。

```
p = lsqminnorm(R,b)
```

```
p =
```

```
-207/137
365/137
79/137
-424/137
```

多右端线性方程组的求解

某些问题涉及求解具有相同系数矩阵 A 但具有不同右端 b 的线性方程组。如果可以同时使用不同的 b 值，则可以将 b 构造为多列矩阵，并使用单个反斜杠命令求解所有方程组： $X = A \setminus [b_1 \ b_2 \ b_3 \ \dots]$ 。

但是，有时不同的 b 值并非全部同时可用，也就是说，您需要连续求解若干方程组。如果使用斜杠 (/) 或反斜杠 (\) 求解其中一个方程组，则该运算符会对系数矩阵 A 进行分解，并使用此矩阵分解来求解。然而，随后每次使用不同的 b 求解类似方程组时，运算符都会对 A 进行同样的分解，而这是一次冗余计算。

此问题的求解是预先计算 A 的分解，然后重新使用因子对 b 的不同值求解。但是，实际上，以这种方式预先计算分解可能很困难，因为需要知道要计算的分解 (LU、LDL、Cholesky 等) 以及如何乘以因子才能对问题求解。例如，使用 LU 分解，您需要求解两个线性方程组才能求解原始方程组 $Ax = b$ ：

```
[L,U] = lu(A);
x = U \ (L \ b);
```

对于具有若干连续右端的线性方程组，建议使用 `decomposition` 对象求解。借助这些对象，您可利用预先计算矩阵分解带来的性能优势，而不必了解如何使用矩阵因子。您可以将先前的 LU 分解替换为：

```
dA = decomposition(A,'lu');
x = dA \ b;
```

如果您不确定要使用哪种分解，`decomposition(A)` 会根据 A 的属性选择正确的类型，类似于反斜杠的功能。

以下简单测试验证了此方法可能带来的性能优势。该测试分别使用反斜杠 (\) 和 `decomposition` 对同一稀疏线性方程组求解 100 次。

```
n = 1e3;
A = sprand(n,n,0.2) + speye(n);
b = ones(n,1);
```

```
% Backslash solution
```

```
tic
for k = 1:100
    x = A \ b;
end
toc
```

Elapsed time is 9.006156 seconds.

```
% decomposition solution
```

```
tic
dA = decomposition(A);
for k = 1:100
    x = dA \ b;
end
toc
```

Elapsed time is 0.374347 seconds.

对于这个问题，`decomposition` 求解比单独使用反斜杠要快得多，而语法仍然很简单。

迭代法

如果系数矩阵 A 很大并且是稀疏矩阵，分解方法一般情况下将不会有效。迭代方法可生成一系列近似解。MATLAB 提供了多个迭代方法来处理大型的稀疏输入矩阵。

函数	说明
<code>pcg</code>	预处理共轭梯度法。此方法适用于 Hermitian 正定系数矩阵 A 。
<code>bicg</code>	双共轭梯度法
<code>bicgstab</code>	双共轭梯度稳定法
<code>bicgstabl</code>	双共轭梯度稳定法(l)
<code>cgs</code>	共轭梯度二乘法
<code>gmres</code>	广义最小残差法
<code>lsqr</code>	LSQR 方法
<code>minres</code>	最小残差法。此方法适用于 Hermitian 系数矩阵 A 。
<code>qmr</code>	拟最小残差法
<code>symmlq</code>	对称的 LQ 方法
<code>tfqmr</code>	无转置 QMR 方法

多线程计算

对于许多线性代数函数和按元素的数值函数，MATLAB 软件支持多线程计算。这些函数将自动在多个线程上执行。要使函数或表达式在多个 CPU 上更快地执行，必须满足许多条件：

- 1 函数执行的运算可轻松划分为并发执行的多个部分。这些部分必须能够在进程之间几乎不通信的情况下执行。它们应需要很少的序列运算。
- 2 数据大小足以使并发执行的任何优势在重要性方面超过对数据分区和管理各个执行线程所需的时间。例如，仅当数组包含数千个或以上的元素时，大多数函数才会加速。
- 3 运算未与内存绑定；处理时间不受内存访问时间控制。一般而言，复杂函数比简单函数速度更快。

如果启用多线程，`inv`、`lscov`、`linsolve` 和 `mldivide` 将会对大型双精度数组（约 10,000 个元素或更多）大幅增加速度。

分解

- “简介”（第 3-17 页）
- “Cholesky 分解”（第 3-17 页）
- “LU 分解”（第 3-18 页）
- “QR 分解”（第 3-19 页）
- “对分解使用多线程计算”（第 3-22 页）

简介

本节中讨论的所有三种矩阵分解利用了三角形矩阵，其中对角线上下的所有元素都为零。涉及三角矩阵的线性方程组可以使用前代或回代方法轻松快捷地求解。

Cholesky 分解

Cholesky 分解将对称矩阵表示为三角矩阵与其转置的积

$$A = R' R,$$

其中，R 是上三角矩阵。

并非所有对称矩阵都可以通过这种方式进行分解；采用此类分解的矩阵被视为正定矩阵。这表明，A 的所有对角线元素都是正数，并且非对角线元素“不太大”。帕斯卡矩阵提供了有趣的示例。在本章中，示例矩阵 A 为 3×3 帕斯卡矩阵。暂时转换为 6×6：

```
A = pascal(6)
```

```
A =
    1    1    1    1    1    1
    1    2    3    4    5    6
    1    3    6   10   15   21
    1    4   10   20   35   56
    1    5   15   35   70  126
    1    6   21   56  126  252
```

A 的元素为二项式系数。每个元素都是其北向和西向邻点之和。Cholesky 分解为

```
R = chol(A)
```

```
R =
    1    1    1    1    1    1
    0    1    2    3    4    5
    0    0    1    3    6   10
    0    0    0    1    4   10
    0    0    0    0    1    5
    0    0    0    0    0    1
```

这些元素同样为二项式系数。R'*R 等于 A 的情况说明了涉及二项式系数的积之和的单位矩阵。

注意 Cholesky 分解也适用于复矩阵。采用 Cholesky 分解的任何复矩阵都满足

$$A = R' R,$$

，并且被视为 Hermitian 正定矩阵。

通过 Cholesky 分解，可以将线性方程组

$$Ax = b$$

替换为

$$R' Rx = b.$$

由于反斜杠运算符能识别三角形方程组，因此这可以在 MATLAB 环境中通过以下表达式快速进行求解

$$x = R \setminus (R' \setminus b)$$

如果 A 为 n×n，则 chol(A) 的计算复杂度为 O(n³)，但后续的反斜杠解的复杂度仅为 O(n²)。

LU 分解

LU 分解（或高斯消去法）将任何方阵 A 都表示为下三角矩阵和上三角矩阵的置换之积

$$A = LU,$$

其中, L 是对角线元素为 1 的下三角形矩阵的置换, U 是上三角形矩阵。

出于理论和计算原因, 必须进行置换。矩阵

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

在不交换其两行的情况下不能表示为三角矩阵的积。尽管矩阵

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix}$$

可以表示为三角矩阵之积, 但当 ε 很小时, 因子中的元素也会很大并且会增大误差, 因此即使置换并非完全必要, 它们也是所希望的。部分主元消元法可确保 L 的元素的模以 1 为限, 并且 U 的元素并不大于 A 的元素。

例如:

$$[L, U] = \text{lu}(B)$$

$$L = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0.3750 & 0.5441 & 1.0000 \\ 0.5000 & 1.0000 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 8.0000 & 1.0000 & 6.0000 \\ 0 & 8.5000 & -1.0000 \\ 0 & 0 & 5.2941 \end{bmatrix}$$

通过对 A 执行 LU 分解, 可以

$$A*x = b$$

使用以下表达式快速对线性方程组求解

$$x = U \setminus (L \setminus b)$$

行列式和逆矩阵是通过 LU 分解使用以下表达式进行计算的

$$\det(A) = \det(L) * \det(U)$$

和

$$\text{inv}(A) = \text{inv}(U) * \text{inv}(L)$$

也可以使用 $\det(A) = \text{prod}(\text{diag}(U))$ 计算行列式, 但行列式的符号可能会相反。

QR 分解

正交矩阵或包含正交列的矩阵为实矩阵, 其列全部具有单位长度并且相互垂直。如果 Q 为正交矩阵, 则

$$Q'Q = QQ' = I.$$

最简单的正交矩阵为二维坐标旋转:

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}.$$

对于复矩阵，对应的术语为单位。由于正交矩阵和单位矩阵会保留长度、保留角度并且不会增大误差，因此适用于数值计算。

正交或 QR 分解将任何矩形矩阵表示为正交或酉矩阵与上三角矩阵的积。此外，也可能涉及列置换。

$$A = QR$$

或

$$AP = QR,$$

其中，Q 为正交或单位矩阵，R 为上三角矩阵，P 为置换向量。

QR 分解有四种变化形式 - 完全大小或合适大小，以及使用列置换或不使用列置换。

超定线性方程组涉及行数超过列数的矩形矩阵，也即 $m \times n$ 并且 $m > n$ 。完全大小的 QR 分解可生成一个方阵 ($m \times m$ 正交矩阵 Q) 和一个矩形 $m \times n$ 上三角矩阵 R:

```
C=gallery('uniformdata',[5 4], 0);
[Q,R] = qr(C)
```

Q =

```
0.6191  0.1406 -0.1899 -0.5058  0.5522
0.1506  0.4084  0.5034  0.5974  0.4475
0.3954 -0.5564  0.6869 -0.1478 -0.2008
0.3167  0.6676  0.1351 -0.1729 -0.6370
0.5808 -0.2410 -0.4695  0.5792 -0.2207
```

R =

```
1.5346  1.0663  1.2010  1.4036
0  0.7245  0.3474 -0.0126
0  0  0.9320  0.6596
0  0  0  0.6648
0  0  0  0
```

在许多情况下，Q 的最后 $m - n$ 列是不需要的，因为这些列会与 R 底部的零相乘。因此，精简 QR 分解可生成一个矩形矩阵（具有正交列的 $m \times n$ Q）以及一个方阵 $n \times n$ 上三角形矩阵 R。对于 5×4 示例，不会节省太多内存，但是对于更大的大量矩形矩阵，在时间和内存方面的节省可能会很重要：

```
[Q,R] = qr(C,0)
```

Q =

```
0.6191  0.1406 -0.1899 -0.5058
0.1506  0.4084  0.5034  0.5974
0.3954 -0.5564  0.6869 -0.1478
0.3167  0.6676  0.1351 -0.1729
0.5808 -0.2410 -0.4695  0.5792
```

R =

```

1.5346  1.0663  1.2010  1.4036
  0  0.7245  0.3474 -0.0126
  0    0  0.9320  0.6596
  0    0    0  0.6648

```

与 LU 分解相比, QR 分解不需要进行任何消元或置换。但是, 可选的列置换 (因存在第三个输出参数而触发) 对检测奇异性或秩亏是很有帮助的。在分解的每一步, 未分解的剩余矩阵的列 (范数最大) 将用作该步骤的基础。这可以确保 R 的对角线元素以降序排列, 并且各列之间的任何线性相关性肯定能够通过检查这些元素来显示。对于此处提供的小示例, C 的第二列的范数大于第一列的范数, 因此这两列被交换:

`[Q,R,P] = qr(C)`

```

Q =
-0.3522  0.8398 -0.4131
-0.7044 -0.5285 -0.4739
-0.6163  0.1241  0.7777

```

```

R =
-11.3578 -8.2762
  0  7.2460
  0    0

```

```

P =
  0  1
  1  0

```

组合了合适大小和列置换后, 第三个输出参数为置换向量而不是置换矩阵:

`[Q,R,p] = qr(C,0)`

```

Q =
-0.3522  0.8398
-0.7044 -0.5285
-0.6163  0.1241

```

```

R =
-11.3578 -8.2762
  0  7.2460

```

```

p =
  2  1

```

QR 分解可将超定线性方程组转换为等效的三角形方程组。表达式

`norm(A*x - b)`

等于

`norm(Q*R*x - b)`

与正交矩阵相乘可保留欧几里德范数, 因此该表达式也等于

`norm(R*x - y)`

其中 $y = Q^*b$ 。由于 R 的最后 $m-n$ 行为零, 因此该表达式将分为两部分:

`norm(R(1:n,1:n)*x - y(1:n))`

并且

```
norm(y(n+1:m))
```

如果 A 具有满秩，则可以对 x 求解，使这些表达式中的第一个表达式为零。然后，第二个表达式便可以提供残差范数。如果 A 没有满秩，则可以通过 R 的三角形结构体对最小二乘问题求基本解。

对分解使用多线程计算

对于许多线性代数函数和按元素的数值函数，MATLAB 软件支持多线程计算。这些函数将自动在多个线程上执行。要使函数或表达式在多个 CPU 上更快地执行，必须满足许多条件：

- 1 函数执行的运算可轻松划分为并发执行的多个部分。这些部分必须能够在进程之间几乎不通信的情况下执行。它们应需要很少的序列运算。
- 2 数据大小足以使并发执行的任何优势在重要性方面超过对数据分区和管理各个执行线程所需的时间。例如，仅当数组包含数千个或以上的元素时，大多数函数才会加速。
- 3 运算未与内存绑定；处理时间不受内存访问时间控制。一般而言，复杂函数比简单函数速度更快。

对于大型双精度数组（约 10,000 个元素），`lu` 和 `qr` 会大幅增加速度。

幂和指数

本主题说明如何使用各种方法计算矩阵幂和指数。

正整数幂

如果 A 为方阵并且 p 为正整数，则 A^p 实际上是将 A 乘以其自身 $p-1$ 次。例如：

```
A = [1 1 1
      1 2 3
      1 3 6];
A^2

ans = 3×3

      3      6     10
      6     14     25
     10     25     46
```

逆幂和分数幂

如果 A 为方阵并且是非奇异的，则 $A^{(-p)}$ 实际上是将 `inv(A)` 乘以其自身 $p-1$ 次。

```
A^(-3)

ans = 3×3

    145.0000 -207.0000  81.0000
   -207.0000  298.0000 -117.0000
    81.0000 -117.0000  46.0000
```

MATLAB® 用相同的算法计算 `inv(A)` 和 $A^{(-1)}$ ，因此结果完全相同。如果矩阵接近奇异，`inv(A)` 和 $A^{(-1)}$ 都会发出警告。

```
isequal(inv(A),A^(-1))
```

```
ans = logical
1
```

还允许小数幂，例如 $A^{(2/3)}$ 。使用小数幂的结果取决于矩阵特征值的分布。

```
A^(2/3)
```

```
ans = 3×3
```

```
0.8901 0.5882 0.3684
0.5882 1.2035 1.3799
0.3684 1.3799 3.1167
```

逐元素幂

\wedge 运算符计算逐元素幂。例如，要对矩阵中的每个元素求平方，可以使用 $A.^2$ 。

```
A.^2
```

```
ans = 3×3
```

```
1 1 1
1 4 9
1 9 36
```

平方根

使用 `sqrt` 函数可以方便地计算矩阵中每个元素的平方根。另一种方法是 $A^{(1/2)}$ 。

```
sqrt(A)
```

```
ans = 3×3
```

```
1.0000 1.0000 1.0000
1.0000 1.4142 1.7321
1.0000 1.7321 2.4495
```

对于其他根，您可以使用 `nthroot`。例如，计算 $A^{(1/3)}$ 。

```
nthroot(A,3)
```

```
ans = 3×3
```

```
1.0000 1.0000 1.0000
1.0000 1.2599 1.4422
1.0000 1.4422 1.8171
```

这些按元素计算的根不同于矩阵平方根，后者计算得到的是另一个矩阵 B 以满足 $A = BB$ 。函数 `sqrtnm(A)` 采用更精确的算法计算 $A^{(1/2)}$ 。`sqrtnm` 中的 `m` 将此函数与 `sqrt(A)` 区分开来，后者与 $A^{(1/2)}$ 一样，以逐元素方式工作。

```
B = sqrtnm(A)
```

```
B = 3×3
```

```

0.8775    0.4387    0.1937
0.4387    1.0099    0.8874
0.1937    0.8874    2.2749

```

B^2

```
ans = 3×3
```

```

1.0000    1.0000    1.0000
1.0000    2.0000    3.0000
1.0000    3.0000    6.0000

```

标量底

除了对矩阵求幂以外，您还可以以矩阵为次数对标量求幂。

2^A

```
ans = 3×3
```

```

10.4630    21.6602    38.5862
21.6602    53.2807    94.6010
38.5862    94.6010    173.7734

```

当您以矩阵为次数对标量求幂时，MATLAB 使用矩阵的特征值和特征向量来计算矩阵幂。如果 $[V,D] = \text{eig}(A)$ ，则 $2^A = V 2^D V^{-1}$ 。

```

[V,D] = eig(A);
V*2^D*V^(-1)

```

```
ans = 3×3
```

```

10.4630    21.6602    38.5862
21.6602    53.2807    94.6010
38.5862    94.6010    173.7734

```

矩阵指数

矩阵指数是以矩阵为次数对标量求幂的特殊情况。矩阵指数的底是欧拉数 $e = \exp(1)$ 。

```

e = exp(1);
e^A

```

```
ans = 3×3
103 ×
```

```

0.1008    0.2407    0.4368
0.2407    0.5867    1.0654
0.4368    1.0654    1.9418

```

expm 函数是计算矩阵指数的一种更方便的方法。

expm(A)

```
ans = 3×3
103 ×

    0.1008    0.2407    0.4368
    0.2407    0.5867    1.0654
    0.4368    1.0654    1.9418
```

矩阵指数可以用多种方法来计算。有关详细信息，请参阅“矩阵指数”。

处理较小的数字

对于非常小的值 x ，MATLAB 函数 **log1p** 和 **expm1** 可以精确计算 $\log(1 + x)$ 和 $e^x - 1$ 。例如，如果您尝试将小于计算机精度的一个数与 1 相加，则结果会舍入到 1。

```
log(1+eps/2)
```

```
ans = 0
```

但是，**log1p** 能够返回更准确的答案。

```
log1p(eps/2)
```

```
ans = 1.1102e-16
```

同样，对于 $e^x - 1$ ，如果 x 非常小，则会将它舍入为零。

```
exp(eps/2)-1
```

```
ans = 0
```

同样，**expm1** 能够返回更准确的答案。

```
expm1(eps/2)
```

```
ans = 1.1102e-16
```

特征值

- “特征值的分解” (第 3-25 页)
- “多重特征值” (第 3-26 页)
- “Schur 分解” (第 3-27 页)

特征值的分解

方阵 A 的特征值和特征向量分别为满足以下条件的标量 λ 和非零向量 u

$$Au = \lambda u。$$

对于对角矩阵的对角线上的特征值 Λ 以及构成矩阵列的对应特征向量 V ，公式为

$$AV = V\Lambda。$$

如果 V 是非奇异的，这将变为特征值分解。

$$A = V\Lambda V^{-1}。$$

微分方程 $dx/dt = Ax$ 的系数矩阵就是一个很好的示例：

```
A =
    0  -6  -1
    6   2 -16
   -5  20 -10
```

此方程的解用矩阵指数 $x(t) = e^{tA}x(0)$ 表示。语句

```
lambda = eig(A)
```

生成包含 A 的特征值的列向量。对于该矩阵，这些特征值为复数：

```
lambda =
   -3.0710
  -2.4645+17.6008i
  -2.4645-17.6008i
```

每个特征值的实部都为负数，因此随着 t 的增加， $e^{\lambda t}$ 将会接近零。两个特征值 $\pm\omega$ 的非零虚部为微分方程的解提供了振动分量 $\sin(\omega t)$ 。

使用这两个输出参数，**eig** 便可以计算特征向量并将特征值存储在对角矩阵中：

```
[V,D] = eig(A)
```

```
V =
   -0.8326    0.2003 - 0.1394i    0.2003 + 0.1394i
   -0.3553   -0.2110 - 0.6447i   -0.2110 + 0.6447i
   -0.4248   -0.6930    -0.6930
```

```
D =
   -3.0710         0         0
         0   -2.4645+17.6008i         0
         0         0   -2.4645-17.6008i
```

第一个特征向量为实数，另外两个向量互为共轭复数。所有三个向量都归一化为具有等于 1 的欧几里德长度 **norm(v,2)**。

矩阵 $V*D*inv(V)$ （可更简洁地写为 $V*D/V$ ）位于 A 的舍入误差界限内。 $inv(V)*A*V$ 或 $V\backslash A*V$ 都在 D 的舍入误差界限内。

多重特征值

某些矩阵没有特征向量分解。这些矩阵是不可对角化的。例如：

```
A = [ 1  -2  1
      0   1  4
      0   0  3]
```

对于此矩阵

```
[V,D] = eig(A)
```

生成

```
V =
```



```

1.0000  1.0000 -0.5571
  0    0.0000  0.7428
  0     0    0.3714

```

D =

```

 1  0  0
 0  1  0
 0  0  3

```

$\lambda = 1$ 时有一个双精度特征值。V 的第一列和第二列相同。对于此矩阵，并不存在一组完整的线性无关特征向量。

Schur 分解

许多高级矩阵计算不需要进行特征值分解。而是使用 Schur 分解。

$$A = USU^H$$

其中，U 是正交矩阵，S 是对角线上为 1×1 和 2×2 块的块上三角矩阵。特征值是通过 S 的对角元素和块显示的，而 U 的列提供了正交基，它的数值属性要远远优于一组特征向量。

例如，比较下面的亏损矩阵的特征值和 Schur 分解：

```

A = [ 6  12  19
      -9 -20 -33
       4   9  15 ];

```

[V,D] = eig(A)

V =

```

-0.4741 + 0.0000i -0.4082 - 0.0000i -0.4082 + 0.0000i
 0.8127 + 0.0000i  0.8165 + 0.0000i  0.8165 + 0.0000i
-0.3386 + 0.0000i -0.4082 + 0.0000i -0.4082 - 0.0000i

```

D =

```

-1.0000 + 0.0000i  0.0000 + 0.0000i  0.0000 + 0.0000i
 0.0000 + 0.0000i  1.0000 + 0.0000i  0.0000 + 0.0000i
 0.0000 + 0.0000i  0.0000 + 0.0000i  1.0000 - 0.0000i

```

[U,S] = schur(A)

U =

```

-0.4741  0.6648  0.5774
 0.8127  0.0782  0.5774
-0.3386 -0.7430  0.5774

```

S =

```

-1.0000  20.7846 -44.6948
  0    1.0000 -0.6096
  0    0.0000  1.0000

```

矩阵 A 为亏损矩阵，因为它不具备一组完整的线性无关特征向量（ V 的第二列和第三列相同）。由于 V 的列并非全部是线性无关的，因此它有一个很大的条件数，大约为 $1e8$ 。但 `schur` 可以计算 U 中的三个不同基向量。由于 U 是正交矩阵，因此 $\text{cond}(U) = 1$ 。

矩阵 S 的实数特征值作为对角线上的第一个条目，并通过右下方的 2×2 块表示重复的特征值。 2×2 块的特征值也是 A 的特征值：

```
eig(S(2:3,2:3))
ans =
    1.0000 + 0.0000i
    1.0000 - 0.0000i
```

奇异值

矩形矩阵 A 的奇异值和对应的奇异向量分别为满足以下条件的标量 σ 以及一对向量 u 和 v

$$\begin{aligned} Av &= \sigma u \\ A^H u &= \sigma v, \end{aligned}$$

其中 A^H 是 A 的 Hermitian 转置。奇异向量 u 和 v 通常缩放至范数为 1。此外，如果 u 和 v 均为 A 的奇异向量，则 $-u$ 和 $-v$ 也为 A 的奇异向量。

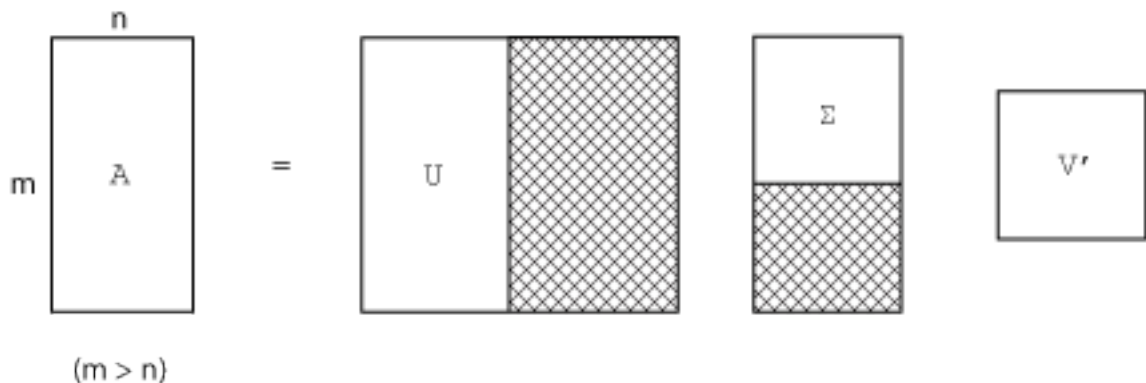
奇异值 σ 始终为非负实数，即使 A 为复数也是如此。对于对角矩阵 Σ 的对角线上的奇异值以及构成两个正交矩阵 U 和 V 的列的对应奇异向量，方程为

$$\begin{aligned} AV &= U\Sigma \\ A^H U &= V\Sigma. \end{aligned}$$

由于 U 和 V 均为单位矩阵，因此将第一个方程的右侧乘以 V^H 会生成奇异值分解方程

$$A = U\Sigma V^H.$$

$m \times n$ 矩阵的完整奇异值分解涉及 $m \times m$ U 、 $m \times n$ Σ 以及 $n \times n$ V 。换句话说， U 和 V 均为方阵， Σ 与 A 的大小相同。如果 A 的行数远多于列数 ($m > n$)，则得到的 $m \times m$ 矩阵 U 为大型矩阵。但是， U 中的大多数列与 Σ 中的零相乘。在这种情况下，精简分解可通过生成一个 $m \times n$ U 、一个 $n \times n$ Σ 以及相同的 V 来同时节省时间和存储空间：



特征值分解是分析矩阵（当矩形表示从向量空间到其自身的映射时）的合适工具，就像分析常微分方程一样。但是，奇异值分解是分析从一个向量空间到另一个向量空间（可能具有不同的维度）的映射的合适工具。大多数联立线性方程组都属于这第二类。

如果 A 是方形的对称正定矩阵，则其特征值分解和奇异值分解相同。但是，当 A 偏离对称性和正定性时，这两种分解之间的差异就会增加。特别是，实矩阵的奇异值分解始终为实数，但非对称实矩阵的特征值分解可能为复数。

对于示例矩阵

$$A = \begin{bmatrix} 9 & 4 \\ 6 & 8 \\ 2 & 7 \end{bmatrix}$$

完整的奇异值分解为

$$[U, S, V] = \text{svd}(A)$$

$$U =$$

$$\begin{bmatrix} 0.6105 & -0.7174 & 0.3355 \\ 0.6646 & 0.2336 & -0.7098 \\ 0.4308 & 0.6563 & 0.6194 \end{bmatrix}$$

$$S =$$

$$\begin{bmatrix} 14.9359 & & 0 \\ & 0 & 5.1883 \\ & 0 & 0 \end{bmatrix}$$

$$V =$$

$$\begin{bmatrix} 0.6925 & -0.7214 \\ 0.7214 & 0.6925 \end{bmatrix}$$

可以验证 $U \cdot S \cdot V'$ 在舍入误差界限内是否等于 A 。对于此类小问题，精简分解只是略小一些。

$$[U, S, V] = \text{svd}(A, 0)$$

$$U =$$

$$\begin{bmatrix} 0.6105 & -0.7174 \\ 0.6646 & 0.2336 \\ 0.4308 & 0.6563 \end{bmatrix}$$

$$S =$$

$$\begin{bmatrix} 14.9359 & & 0 \\ & 0 & 5.1883 \end{bmatrix}$$

$$V =$$

```
0.6925 -0.7214
0.7214 0.6925
```

同样， $U*S*V'$ 在舍入误差界限内等于 A 。

如果矩阵 A 很大并且是稀疏矩阵，则使用 `svd` 来计算所有奇异值和向量在某些情况下可能会不太切合实际。例如，如果您只需了解几个最大的奇异值，则计算一个 5000×5000 稀疏矩阵的所有奇异值会带来大量额外工作。在只需要一部分奇异值和向量的情况下，`svds` 函数优先于 `svd`。

对于一个密度约为 30% 的 1000×1000 随机稀疏矩阵，

```
n = 1000;
A = sprand(n,n,0.3);
```

最大的六个奇异值为

```
S = svds(A)
```

```
S =
```

```
130.2184
16.4358
16.4119
16.3688
16.3242
16.2838
```

此外，最小的六个奇异值为

```
S = svds(A,6,'smallest')
```

```
S =
```

```
0.0740
0.0574
0.0388
0.0282
0.0131
0.0066
```

对于可作为满矩阵 `full(A)` 载入内存的较小矩阵，使用 `svd(full(A))` 的速度可能仍旧快于使用 `svds`。但对于确实很大的稀疏矩阵，就有必要使用 `svds`。

非线性函数的运算

本节内容
“函数句柄” (第 3-31 页)
“功能函数” (第 3-31 页)

函数句柄

可以创建任何 MATLAB 函数的句柄，并将该句柄用作引用该函数的一种方式。函数句柄通常在参数列表中传递给其他函数，然后，其他函数可以使用该句柄执行或计算相应函数。

在 MATLAB 中，使用 at 符号 @ 在函数名称前面构造函数句柄。下面的示例为 sin 函数创建一个函数句柄，并将其赋值给变量 fhandle：

```
fhandle = @sin;
```

您可以按照使用函数名称调用函数的相同方式，通过函数句柄来调用函数。语法为

```
fhandle(arg1, arg2, ...);
```

下面显示的函数 plot_fhandle 接收函数句柄和数据，使用函数句柄生成 y 轴数据，并对数据绘图：

```
function plot_fhandle(fhandle, data)
plot(data, fhandle(data))
```

当调用带有如下 sin 函数的句柄和参数的 plot_fhandle 时，得到的计算结果会生成正弦波图：

```
plot_fhandle(@sin, -pi:0.01:pi)
```

功能函数

名为“接受函数句柄的函数”的函数类与标量变量的非线性函数配合使用。也就是说，某个函数基于另一个函数运行。接受函数句柄的函数包括

- 找零
- 优化
- 求积
- 常微分方程

MATLAB 通过定义非线性函数的文件来表示非线性函数。例如，以下是 matlab/demos 文件夹中的 humps 函数的简化版本：

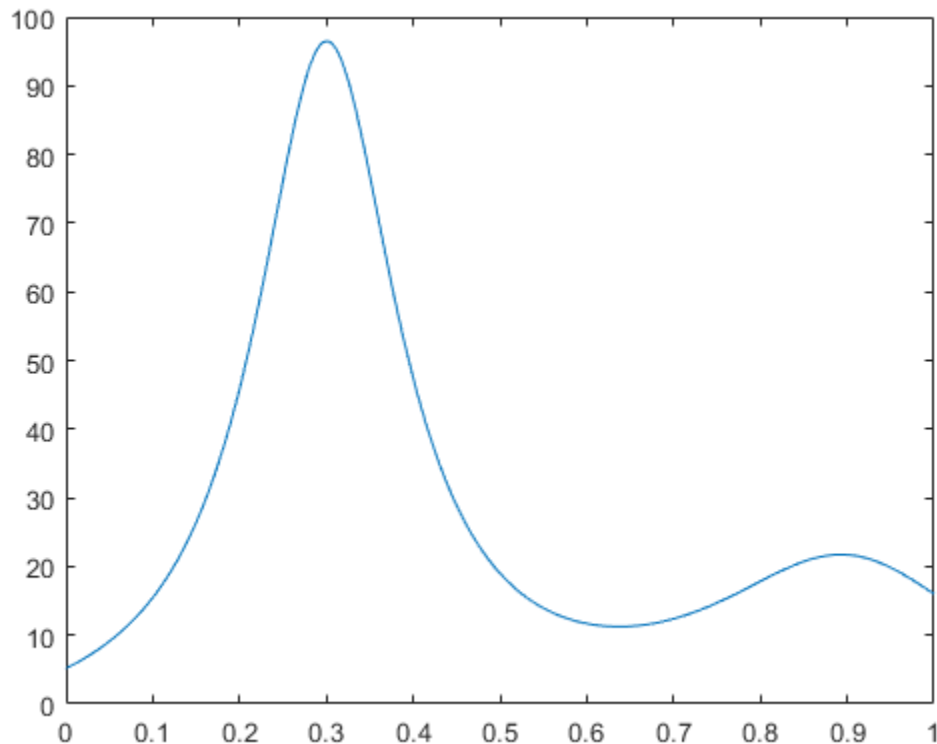
```
function y = humps(x)
y = 1./((x-3).^2 + .01) + 1./((x-9).^2 + .04) - 6;
```

使用以下命令在区间 $0 \leq x \leq 1$ 中的一组点处计算此函数

```
x = 0:.002:1;
y = humps(x);
```

然后，使用以下命令绘制该函数

```
plot(x,y)
```



此图形表明，该函数在 $x = 0.6$ 附近具有局部最小值。函数 `fminsearch` 会求最小值，即此函数采用此最小值时的 x 值。`fminsearch` 的第一个参数是此函数达到最小值时的函数句柄，第二个参数是猜测的最小值的大致位置：

```
p = fminsearch(@humps,.5)
p =
    0.6370
```

要在极小值处计算此函数，

```
humps(p)
```

```
ans =
    11.2528
```

数值分析采用术语求积和积分来区别定积分的近似数值和常微分方程的数值积分。MATLAB 正交例程包括 `quad` 和 `quadl`。语句

```
Q = quadl(@humps,0,1)
```

计算此图形中曲线下方的面积，并生成

```
Q =
    29.8583
```

最后，此图形表明该函数在此区间中永远不会为零。这样，如果使用

```
z = fzero(@humps,.5)
```

搜索零，则会在此区间外部找到一个零：

$$z = -0.1316$$

多变量数据

MATLAB 对多变量统计数据使用列向分析。数据集中的每一列都代表一个变量，每一行都代表一个观测值。第 (i,j) 个元素是第 j 个变量的第 i 个观测值。

例如，设想一个具有三个变量的数据集：

- 心率
- 体重
- 每周锻炼小时数

对于五个观测值，最终数组可能如下所示

```
D = [ 72      134      3.2
      81      201      3.5
      69      156      7.1
      82      148      2.4
      75      170      1.2 ]
```

第一行包含患者 1 的心率、体重和锻炼小时数，第二行包含患者 2 的数据，依此类推。现在，您可以对此数据集应用多个 MATLAB 数据分析函数：例如，要获取每一列的平均差和标准差，请使用

```
mu = mean(D), sigma = std(D)
```

```
mu =
    75.8    161.8     3.48
```

```
sigma =
    5.6303    25.499     2.2107
```

有关 MATLAB 中提供的数据分析函数的列表，请键入

```
help datafun
```

如果您可以使用 Statistics and Machine Learning Toolbox™ 软件，请键入

```
help stats
```


数据分析

简介

每个数据分析都包含一些标准的活动：

- 预处理 - 考虑离群值以及缺失值，并对数据进行平滑处理以便确定可能的模型。
- 汇总 - 计算基本的统计信息以描述数据的总体位置、规模及形状。
- 可视化 - 绘制数据以便确定模式和趋势。
- 建模 - 更全面地描述数据趋势，以便预测新数据值。

数据分析通过这些活动，以实现两个基本目标：

- 1 使用简单模型来描述数据中的模式，以便实现正确预测。
- 2 了解变量之间的关系，以便构建模型。

此部分说明如何在 MATLAB 环境中执行基本数据分析。

数据的预处理

此示例显示如何预处理分析用的数据。

概述

通过将数据加载到合适的 MATLAB® 容器变量并区分“正确”数据和“错误”数据，开始数据分析。这是初级步骤，可确保在后续的分析过程中得出有意义的结论。

加载数据

首先加载 `count.dat` 中的数据：

```
load count.dat
```

这个 24×3 数组 `count` 包含三个十字路口（列）在一天中的每小时流量统计（行）。

缺失数据

MATLAB NaN（非数字）值通常用于表示缺失数据。通过 NaN 值，缺失数据的变量可以维护其结构体 - 在本示例中，即在所有三个十字路口中的索引都是一致的 24×1 向量。

使用 `isnan` 函数检查第三个十字路口的数据是否存在 NaN 值：

```
c3 = count(:,3); % Data at intersection 3  
c3NaNCount = sum(isnan(c3))
```

```
c3NaNCount = 0
```

`isnan` 返回一个大小与 `c3` 相同的逻辑向量，并且通过相应条目指明数据中 24 个元素内的每个元素是存在 (1) 还是缺少 (0) NaN 值。在本示例中，逻辑值总和为 0，因此数据中没有 NaN 值。

离群值部分的数据中引入了 NaN 值。

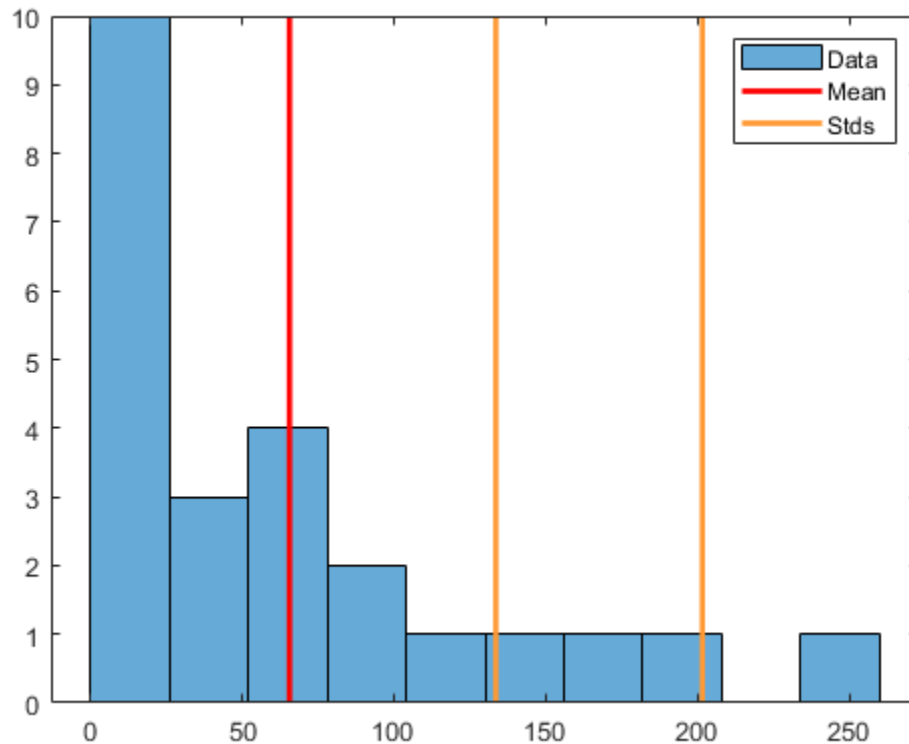
离群值

离群值是与其余数据中的模式明显不同的数据值。离群值可能由计算错误所致，也可能表示数据的重要特点。根据对数据及数据源的了解，确定离群值并决定其处理方法。

确定离群值的一种常用方法是查找与均值 μ 的标准差 σ 大于某个数字的值。下面的代码绘制第三个十字路口的数据直方图以及 μ 和 $\mu + \eta\sigma$ ($\eta = 1, 2$) 处的直线：

```
h = histogram(c3,10); % Histogram
N = max(h.Values); % Maximum bin count
mu3 = mean(c3); % Data mean
sigma3 = std(c3); % Data standard deviation

hold on
plot([mu3 mu3],[0 N],'r','LineWidth',2) % Mean
X = repmat(mu3+(1:2)*sigma3,2,1);
Y = repmat([0;N],1,2);
plot(X,Y,'Color',[255 153 51]./255,'LineWidth',2) % Standard deviations
legend('Data','Mean','Stds')
hold off
```



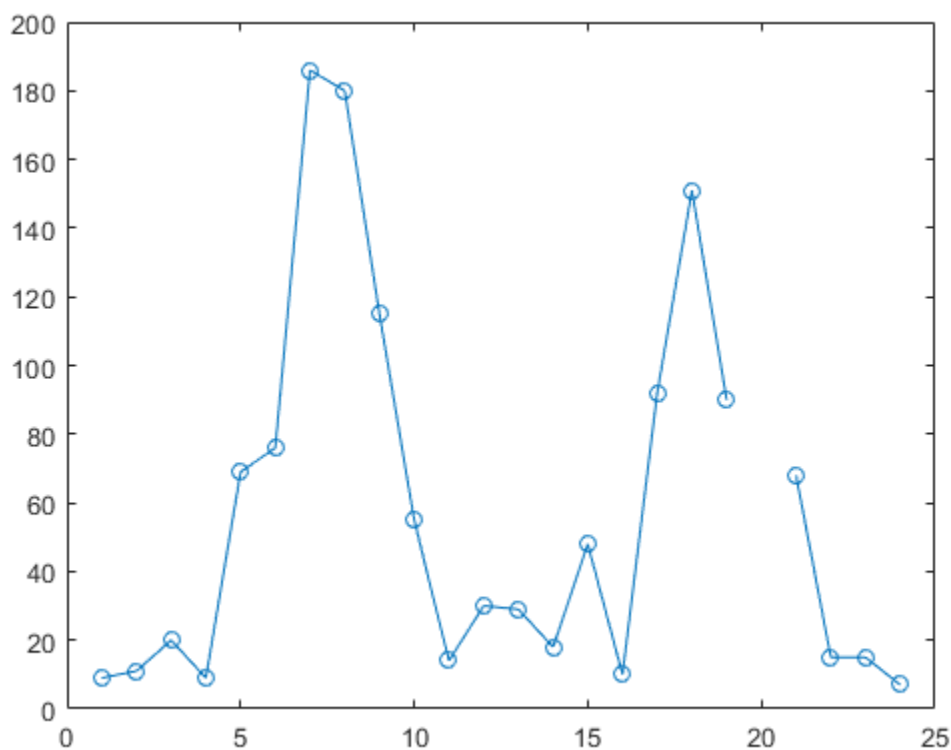
此绘图表明某些数据比均值大两个标准差以上。如果将这些数据标识为错误（而非特点），请将其替换为 NaN 值，如下所示：

```
outliers = (c3 - mu3) > 2*sigma3;
c3m = c3; % Copy c3 to c3m
c3m(outliers) = NaN; % Add NaN values
```

平滑和筛选

第三个十字路口的数据时序图（已在离群值中删除该离群值）生成以下绘图：

```
plot(c3m,'o-')
hold on
```



在绘图中，第 20 个小时的 NaN 值出现间隔。这种对 NaN 值的处理方式是 MATLAB 绘图函数所特有的。

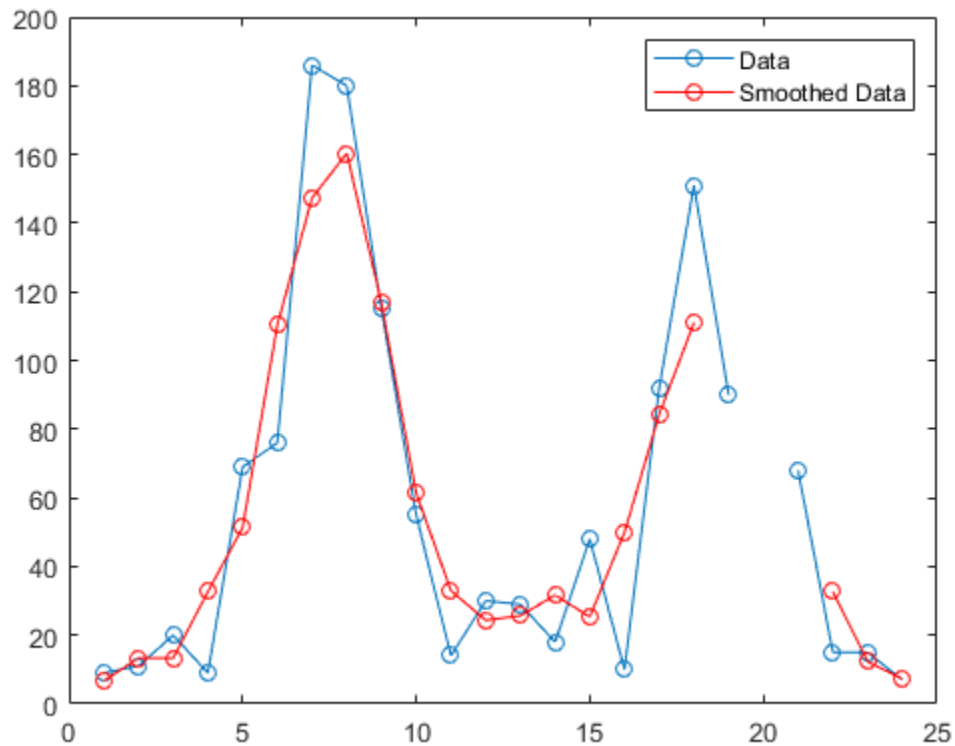
噪音数据围绕预期值显示随机变化。您可能希望在构建模型之前对数据进行平滑处理，以便显示其主要特点。平滑处理应当以下面两个基本假定为基础：

- 预测变量（时间）和响应（流量）之间的关系平稳。
- 由于已减少噪音，因此平滑算法生成比预期值更好的估计值。

使用 MATLAB `convn` 函数对数据应用简单移动平均平滑法：

```
span = 3; % Size of the averaging window
window = ones(span,1)/span;
smoothed_c3m = convn(c3m>window,'same');
```

```
h = plot(smoothed_c3m,'ro-');
legend('Data','Smoothed Data')
```



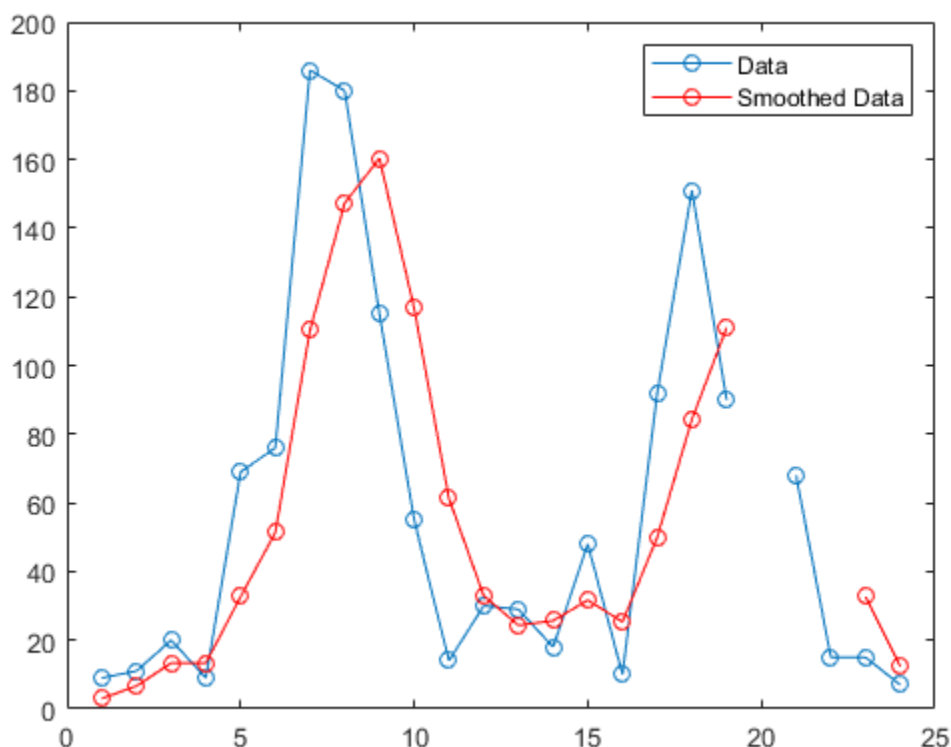
使用变量 `span` 控制平滑范围。当平滑窗口在数据中包含 `NaN` 值时，平均值计算返回 `NaN` 值，从而增大平滑数据中的间隔大小。

此外，还可以对平滑数据使用 `filter` 函数：

```
smoothed2_c3m = filter(window,1,c3m);
```

```
delete(h)
```

```
plot(smoothed2_c3m,'ro-','DisplayName','Smoothed Data');
```



平滑数据在以上绘图的基础上发生了偏移。带有 'same' 参数的 `convn` 返回卷积的中间部分，其长度与数据相同。`filter` 返回卷积的开头，其长度与数据相同。否则算法相同。

平滑处理可估计预测变量的每个值的响应值分布的中心。它使许多拟合算法的基本假定无效，即预测器的每个值的错误彼此独立。相应地，您可以使用平滑数据确定模型，但应避免使用平滑数据拟合模型。

汇总数据

此示例显示如何汇总数据。

概述

许多 MATLAB® 函数都可以用于汇总数据样本的总体位置、规模和形状。

使用 MATLAB® 的一大优点是：函数处理整个数据数组，而不是仅处理单一标量值。这些函数称为向量化函数。通过向量化可以进行有效的问题公式化（使用基于数组的数据）和有效计算（使用向量化统计函数）。

位置度量

通过定义“典型”值来汇总数据示例的位置。使用函数 `mean`、`median` 和 `mode` 计算常见位置度量或“集中趋势”：

```
load count.dat
x1 = mean(count)
```

```
x1 = 1×3
    32.0000  46.5417  65.5833
```

```
x2 = median(count)
x2 = 1×3
    23.5000  36.0000  39.0000
```

```
x3 = mode(count)
x3 = 1×3
    11     9     9
```

与所有统计函数一样，上述 MATLAB® 函数汇总多个观测（行）中的数据，并保留变量（列）。这些函数在一次调用中计算三个十字路口中的每个十字路口的数据位置。

规模度量

有多种方法可以度量数据样本的规模或“分散程度”。MATLAB® 函数 `max`、`min`、`std` 和 `var` 计算某些常见度量：

```
dx1 = max(count)-min(count)
dx1 = 1×3
    107   136   250
```

```
dx2 = std(count)
dx2 = 1×3
    25.3703  41.4057  68.0281
```

```
dx3 = var(count)
dx3 = 1×3
    103 ×
    0.6437   1.7144   4.6278
```

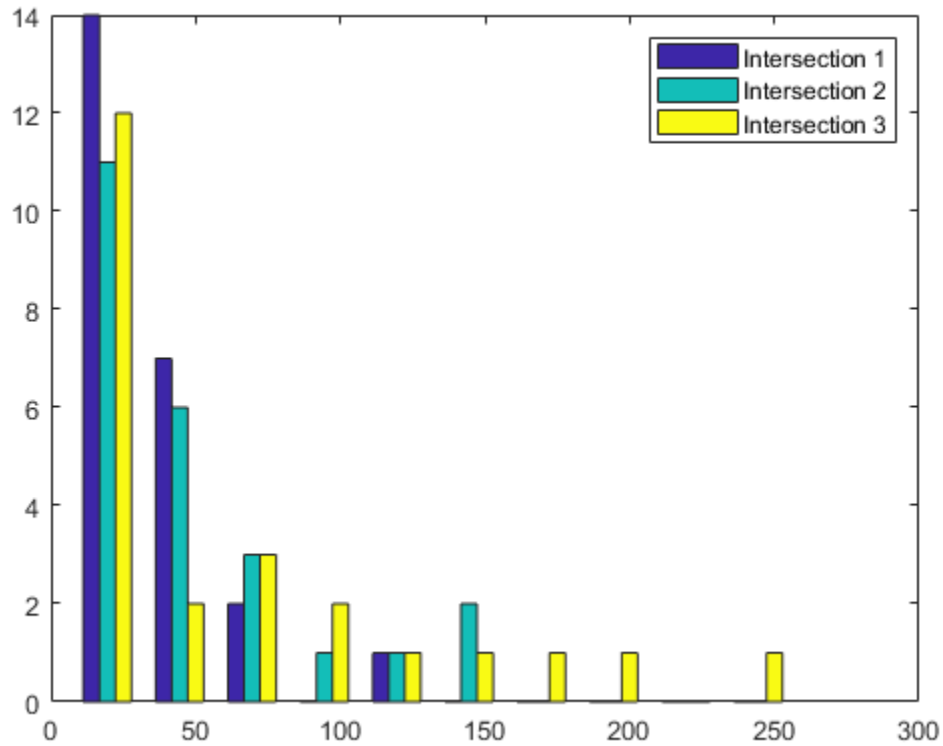
与所有统计函数一样，上述 MATLAB® 函数汇总多个观测（行）中的数据，并保留变量（列）。这些函数在一次调用中计算三个十字路口中的每个十字路口的数据规模。

分布形状

汇总分布的形状比汇总分布的位置或规模更难。MATLAB® `hist` 函数绘制直方图，可视化显示汇总数据：

```
figure
hist(count)
legend('Intersection 1',...
```

'Intersection 2',...
'Intersection 3')



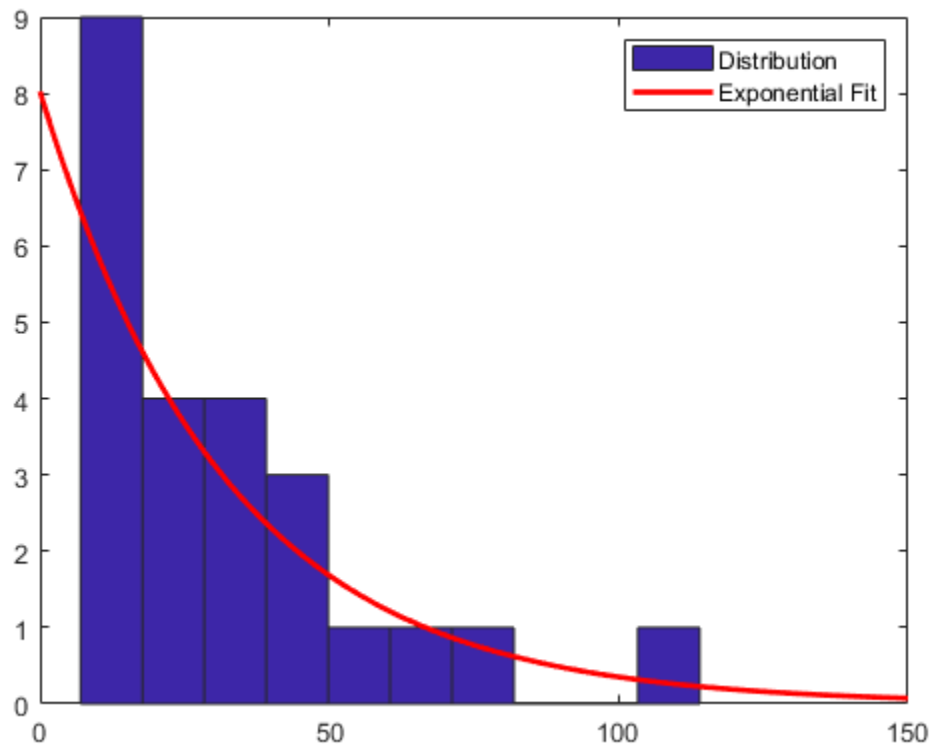
参数模型提供分布形状的汇总分析。指数分布和数据均值指定的参数 **mu** 非常适用于流量数据：

```
c1 = count(:,1); % Data at intersection 1
[bin_counts,bin_locations] = hist(c1);
bin_width = bin_locations(2) - bin_locations(1);
hist_area = (bin_width)*(sum(bin_counts));

figure
hist(c1)
hold on

mu1 = mean(c1);
exp_pdf = @(t)(1/mu1)*exp(-t/mu1); % Integrates
                                     % to 1

t = 0:150;
y = exp_pdf(t);
plot(t,(hist_area)*y,'r','LineWidth',2)
legend('Distribution','Exponential Fit')
```



将常规参数模型与数据分布拟合的方法不在此部分的论述范围内。Statistics and Machine Learning Toolbox™ 软件提供用于计算分布参数的最大似然估计的函数。

可视化数据

- “概述” (第 3-42 页)
- “二维散点图” (第 3-42 页)
- “三维散点图” (第 3-44 页)
- “散点图数组” (第 3-45 页)
- “浏览图形中的数据” (第 3-46 页)

概述

您可以使用多种 MATLAB 图形来可视化数据模式和趋势。此部分介绍的散点图有助于可视化不同十字路口的流量数据之间的关系。数据浏览工具用于在图形上查询各个数据点，并与数据点进行交互。

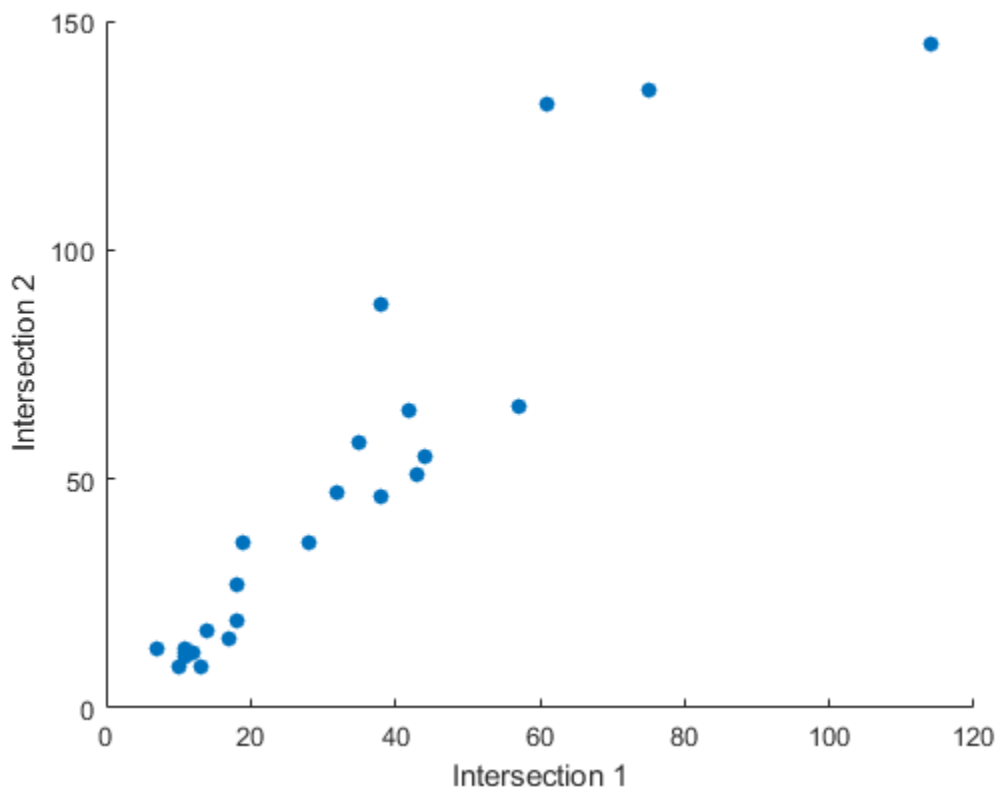
注意 此部分继续执行“汇总数据” (第 3-39 页) 中的数据分析。

二维散点图

二维散点图使用 `scatter` 函数创建，用于显示前两个十字路口的流量之间的关系：


```
load count.dat
c1 = count(:,1); % Data at intersection 1
c2 = count(:,2); % Data at intersection 2
```

```
figure
scatter(c1,c2,'filled')
xlabel('Intersection 1')
ylabel('Intersection 2')
```



使用 `cov` 函数计算的协方差计算两个变量之间的线性关系强度（数据在散点图中沿着最小二乘直线排列的松紧度）：

```
C12 = cov([c1 c2])
```

```
C12 = 2×2
103 ×
```

```
0.6437  0.9802
0.9802  1.7144
```

结果以对称的方阵形式显示，并在第 (i, j) 个位置中显示第 i 个和第 j 个变量的协方差。第 i 个对角线元素是第 i 个变量的方差。

协方差的缺点是：取决于度量各个变量所使用的单位。您可以将变量的协方差除以标准差，以将值归一化为介于 +1 和 -1 之间。`corrcoef` 函数计算相关系数：

```
R12 = corrcoef([c1 c2])
```

```
R12 = 2×2
```

```
1.0000 0.9331
0.9331 1.0000
```

```
r12 = R12(1,2) % Correlation coefficient
```

```
r12 = 0.9331
```

```
r12sq = r12^2 % Coefficient of determination
```

```
r12sq = 0.8707
```

由于已经过归一化，因此相关系数的值可以方便地与其他成对的十字路口的值相比较。相关系数的平方（即决定系数）是最小二乘直线的方差除以均值方差的结果。因此，它与响应（在本示例中为第 2 个十字路口的流量）中的方差成比例，在散点图中，该方差已被清除，或者用最小平方直线以统计方式说明。

三维散点图

三维散点图使用 `scatter3` 函数创建，用于显示所有三个十字路口的流量之间的关系：使用在上述步骤中创建的变量 `c1`、`c2` 和 `c3`：

```
figure
```

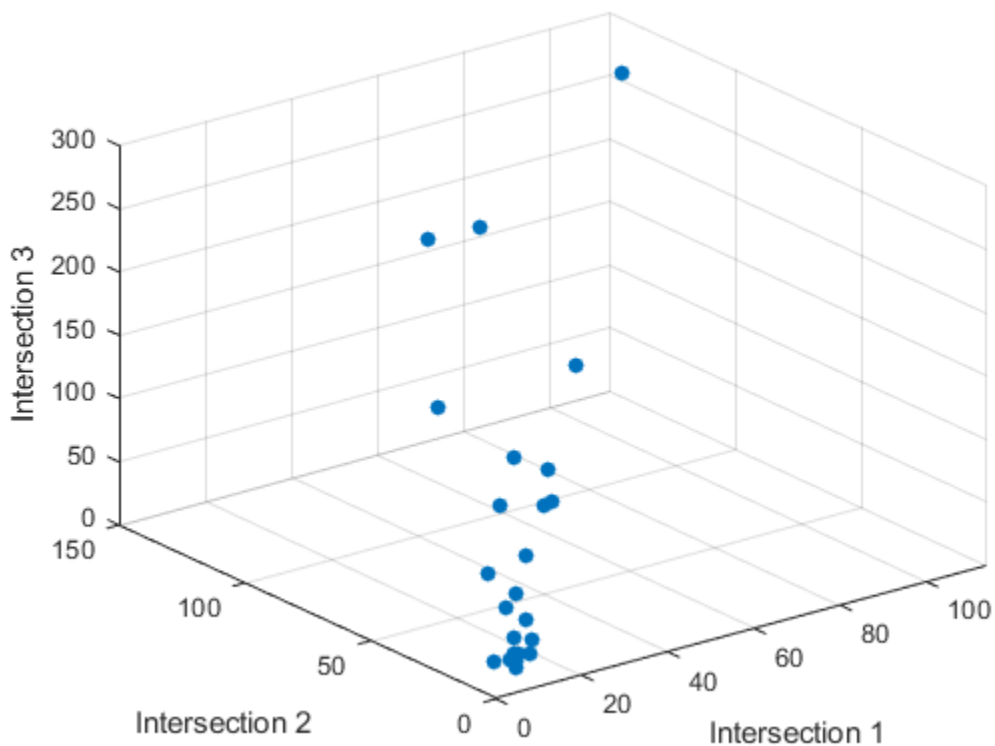
```
c3 = count(:,3); % Data at intersection 3
```

```
scatter3(c1,c2,c3,'filled')
```

```
xlabel('Intersection 1')
```

```
ylabel('Intersection 2')
```

```
zlabel('Intersection 3')
```



通过使用 **eig** 函数计算协方差矩阵的特征值来度量三维散点图中的变量之间的线性关系强度。

```
vars = eig(cov([c1 c2 c3]))
```

```
vars = 3×1  
103 ×
```

```
0.0442
```

```
0.1118
```

```
6.8300
```

```
explained = max(vars)/sum(vars)
```

```
explained = 0.9777
```

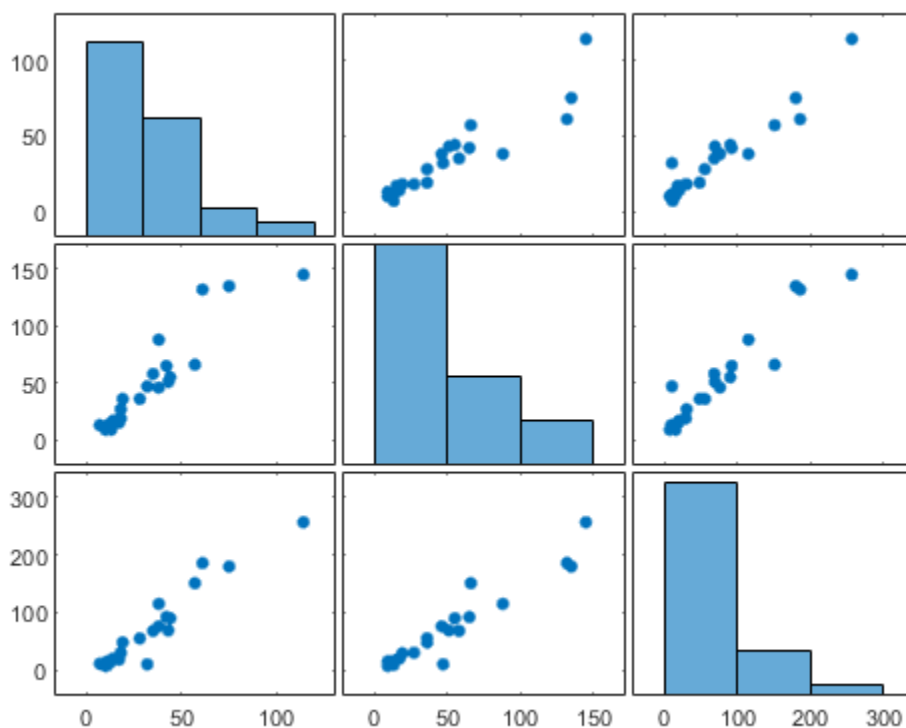
特征值是基于数据的主分量的方差。变量 **explained** 度量数据轴上第一个主分量说明的方差的比例。与二维散点图的决定系数不同的是，此度量会区分预测器和响应变量。

散点图数组

使用 **plotmatrix** 函数比较多对十字路口之间的关系：

```
figure
```



```
plotmatrix(count)
```



位于数组第 (i, j) 个位置的绘图是一个散点图，第 i 个变量位于纵轴上，第 j 个变量位于横轴上。第 i 个对角线位置的绘图是第 i 个变量的直方图。

浏览图形中的数据


使用图窗工具栏中的两个工具，您可以通过鼠标选取几乎任何 MATLAB 图形中的观测值：

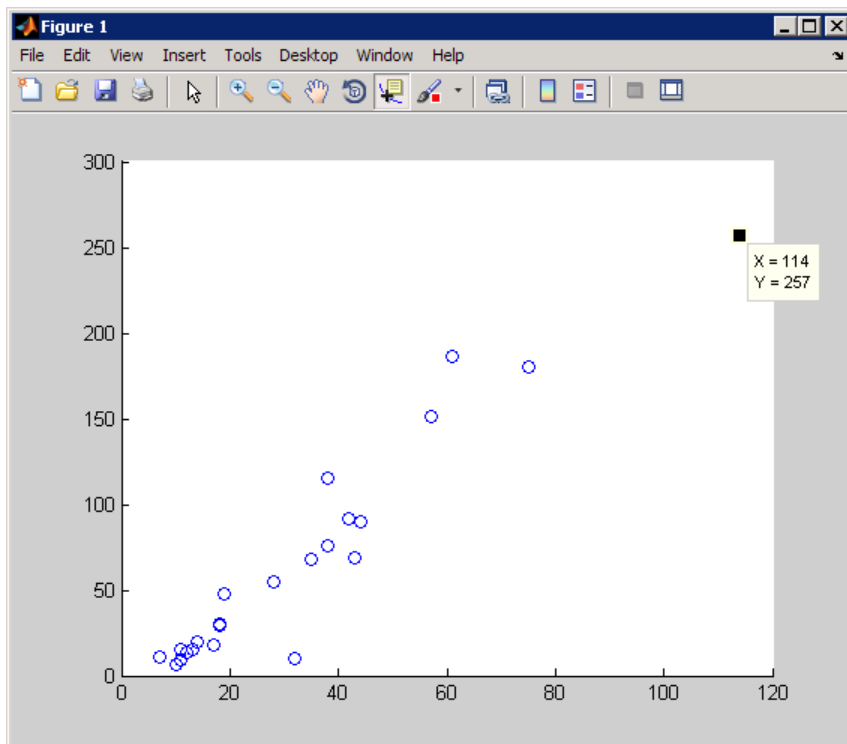
- 数据游标 
- 数据刷亮 

上述每个工具都会使您置于探索模式，在此模式下，您可以在图形上选择数据点以便确定其值，并创建工作区变量来包含特定观测值。当使用数据刷亮时，您还可以复制、删除或替换所选观测值。


例如，绘制 `count` 的第一列和第三列的散点图：

```
load count.dat
scatter(count(:,1),count(:,3))
```

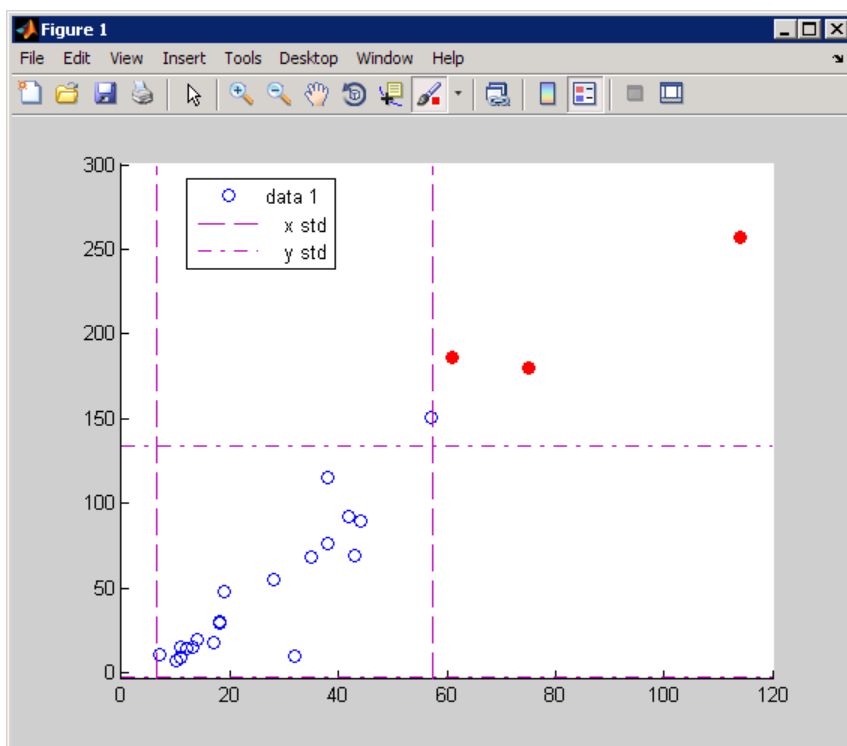
选择数据游标工具 ，然后点击最右侧的数据点。此时将在该处显示说明该点的 x 和 y 值的数据提示。



默认情况下，数据提示显示 x 、 y 和 z （适用于三维绘图）坐标。您可以将数据提示从一个数据点拖至另一个数据点以便查看新值，或者通过右键点击数据提示并使用上下文菜单来添加其他数据提示。此外，还可以使用 MATLAB 代码自定义数据提示显示的文本。

数据刷亮是一个相关功能，用于通过点击或拖动在图形上高亮显示一个或多个观测值。要进入数据刷亮模式，请在图窗工具栏上点击“数据刷亮”工具  左侧。点击此工具图标右侧的箭头会显示一个调色板，用于选择刷亮观测值所使用的颜色。此图窗显示的散点图与上一图窗相同，但超过均值一个标准差的所有观测值（使用 **工具 > 数据统计信息** GUI 确定）刷亮为红色。

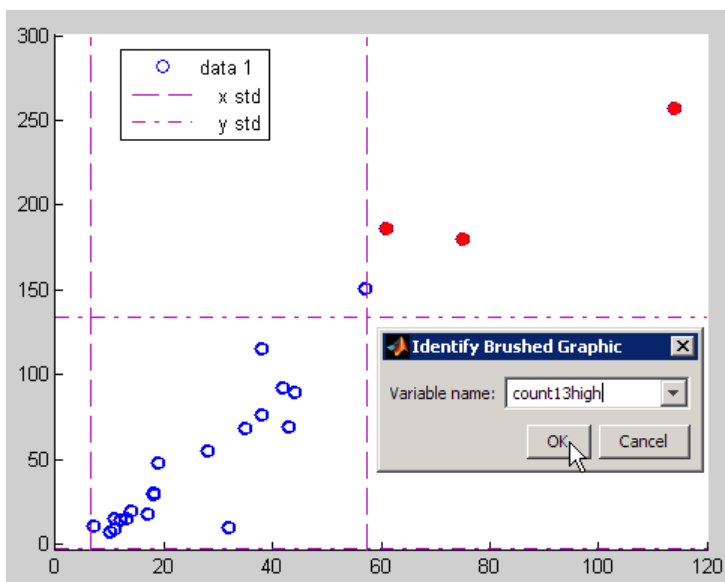
```
scatter(count(:,1),count(:,3))
```



刷亮数据观测值之后，可以对数据观测值执行下列运算：

- 删除数据观测值。
- 将数据观测值替换为常量值。
- 将数据观测值替换为 NaN 值。
- 将数据观测值拖动、复制并粘贴到命令行窗口。
- 将数据观测值另存为工作区变量。

例如，使用“数据刷亮”上下文菜单或**工具 > 刷亮 > 新建变量**选项创建一个名为 `count13high` 的新变量。



此时将在工作区中生成一个新变量：


count13high

count13high =

```
61 186
75 180
114 257
```

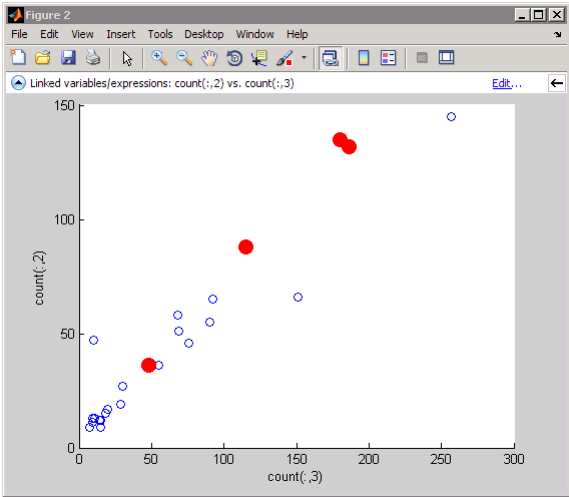
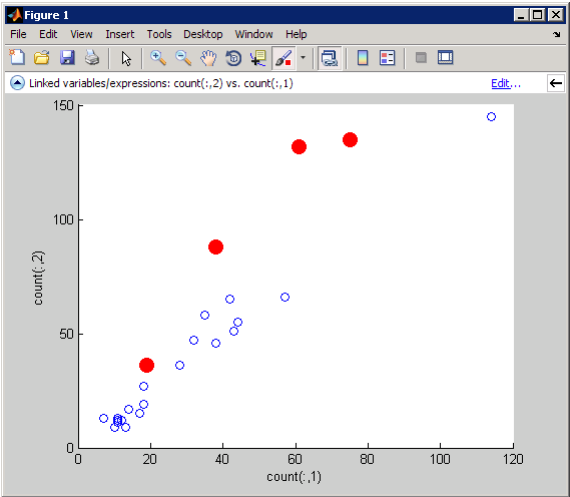
链接图或数据链接是一个与数据刷亮紧密相关的功能。如果绘图实时连接到所描绘的工作区数据，则此绘图已链接。对于存储在对象的 **XData**、**YData**（以及 **ZData**，如果适用的话）中的变量副本，当所链接的工作区变量发生更改或被删除时，这些变量副本会自动更新。这会导致显示这些副本的图形自动更新。

通过将绘图链接到变量，您可以通过观测值的不同表示形式来跟踪特定观测值。当刷亮链接图中的数据点时，刷亮一幅图形会高亮显示链接到相同变量的所有图形中的相同观测值。

数据链接建立图窗和工作区变量之间的即时双向通信，通信方式与“变量编辑器”和工作区变量之间的通信方式相同。通过在图窗的工具栏上激活“数据链接”工具  可以创建链接。激活此工具后，绘图顶部会显示下图中显示的“链接图”信息栏（可能会遮住绘图标题）。您可以消除此信息栏（如下图所示），而不必取消链接绘图；此信息栏不会被打印，并且不会随图窗一起保存。

下面两幅图形描绘在刷亮左侧图形中的某些观测值之后链接数据的散点图显示。常见变量 **count** 将刷亮标记携带到右图。即使右图未处于数据刷亮模式，它也会显示刷亮标记，因为它链接到其变量。

```
figure
scatter(count(:,1),count(:,2))
xlabel('count(:,1)')
ylabel('count(:,2)')
figure
scatter(count(:,3),count(:,2))
xlabel('count(:,3)')
ylabel('count(:,2)')
```



同左图相比，右图表明刷亮的观测值线性关系更加密切。


当在“变量编辑器”中显示这些变量时，刷亮数据观测值以刷亮颜色高亮显示，如此处所示：

openvar count

Variable Editor - count

count <24x3 double>

	1	2	3	4
1	11	11	9	
2	7	13	11	
3	14	17	20	
4	11	13	9	
5	43	51	69	
6	38	46	76	
7	61	132	186	
8	75	135	180	
9	38	88	115	
10	28	36	55	
11	12	12	14	
12	18	27	30	
13	18	19	29	
14	17	15	18	
15	19	36	48	
16	32	47	10	
17	42	65	92	
18	57	66	151	
19	44	55	90	
20	114	145	257	
21	35	58	68	
22	11	12	15	
23	13	9	15	
24	10	9	7	

在“变量编辑器”中，可以更改链接的绘图数据的任何值，并且图形将反映所做的编辑。要刷亮“变量编辑器”中的数据观测值，请点击其“刷亮工具”按钮。如果链接图当前描绘了刷亮的变量，刷亮的观测值将在链接图和“变量编辑器”中高亮显示。当刷亮作为矩阵中的某个列的变量时，还会刷亮该行中的其他列。也即，可以刷亮行或列向量中的各观测值，但矩阵中的所有列都会在刷亮的任何行中高亮显示，而不是仅高亮显示您点击的观测值。

数据建模

- “概述”（第 3-50 页）
- “多项式回归”（第 3-50 页）
- “一般线性回归”（第 3-51 页）

概述

参数模型将对数据关系的理解转换为具有预测能力的分析工具。对于流量数据的上升和下降趋势，多项式模型和正弦模型是理想选择。

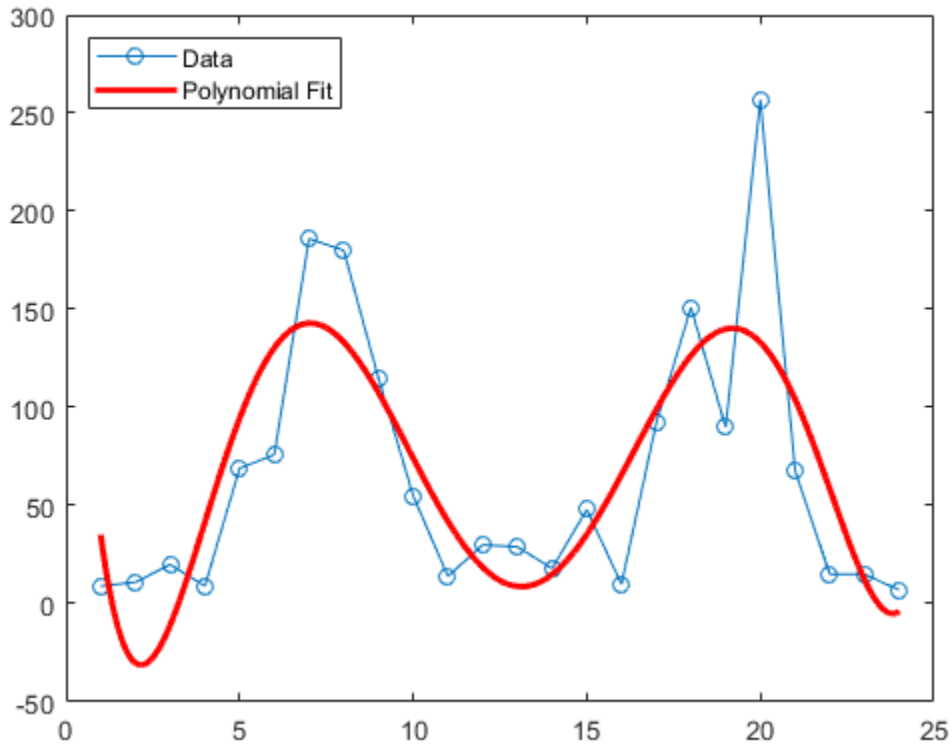
多项式回归

使用 `polyfit` 函数估计多项式模型的系数，然后使用 `polyval` 函数根据预测变量的任意值评估模型。

以下代码使用 6 次多项式模型拟合第三个十字路口的流量数据：

```
load count.dat
c3 = count(:,3); % Data at intersection 3
tdata = (1:24)';
p_coeffs = polyfit(tdata,c3,6);

figure
plot(c3,'o-')
hold on
tfit = (1:0.01:24)';
yfit = polyval(p_coeffs,tfit);
plot(tfit,yfit,'r-','LineWidth',2)
legend('Data','Polynomial Fit','Location','NW')
```

此模型的优点是可以非常简单地跟踪升降趋势。但是，此模型的预测能力可能有欠准确性，特别是在数据两端。

一般线性回归

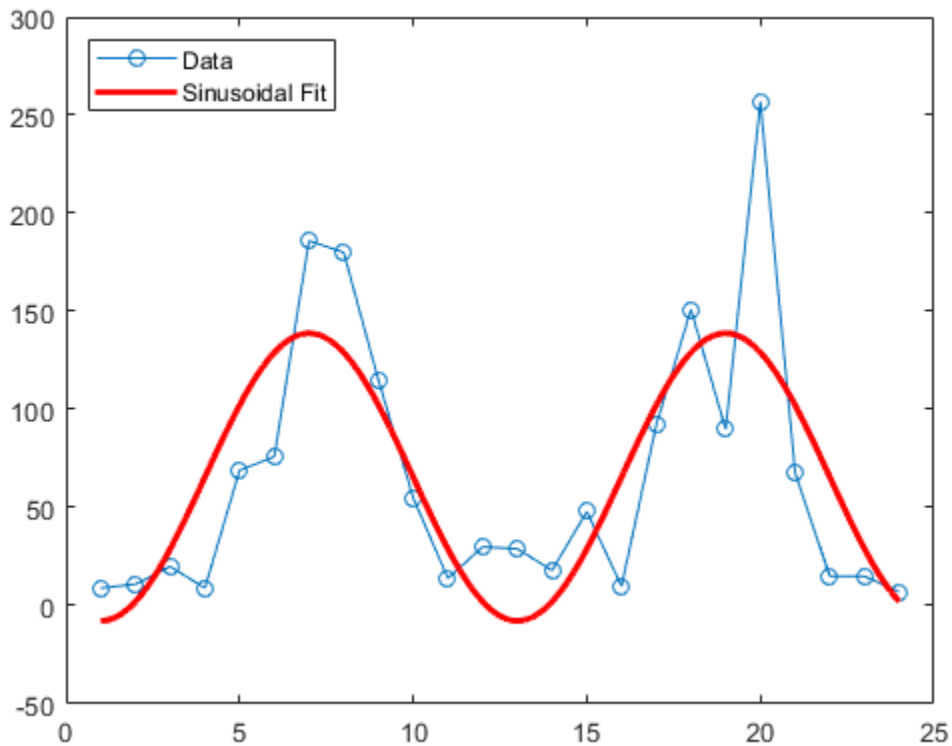
假定数据是周期为 12 个小时的周期性数据，并且峰值出现在第 7 个小时左右，拟合以下形式的正弦模型是合理的：

$$y = a + b\cos((2\pi/12)(t - 7))$$

系数 a 和 b 呈线性关系。使用 MATLAB® `mldivide`（反斜杠）运算符拟合一般线性模型：

```
load count.dat
c3 = count(:,3); % Data at intersection 3
tdata = (1:24)';
X = [ones(size(tdata)) cos((2*pi/12)*(tdata-7))];
s_coeffs = X\c3;

figure
plot(c3,'o-')
hold on
tfit = (1:0.01:24)';
yfit = [ones(size(tfit)) cos((2*pi/12)*(tfit-7))]*s_coeffs;
plot(tfit,yfit,'r-','LineWidth',2)
legend('Data','Sinusoidal Fit','Location','NW')
```



使用 `lscov` 函数计算拟合时的统计信息，例如系数的估计标准误差和均方误差：

```
[s_coeffs,stdx,mse] = lscov(X,c3)
```

```
s_coeffs = 2×1
```

```
65.5833  
73.2819
```

```
stdx = 2×1
```

```
8.9185  
12.6127
```

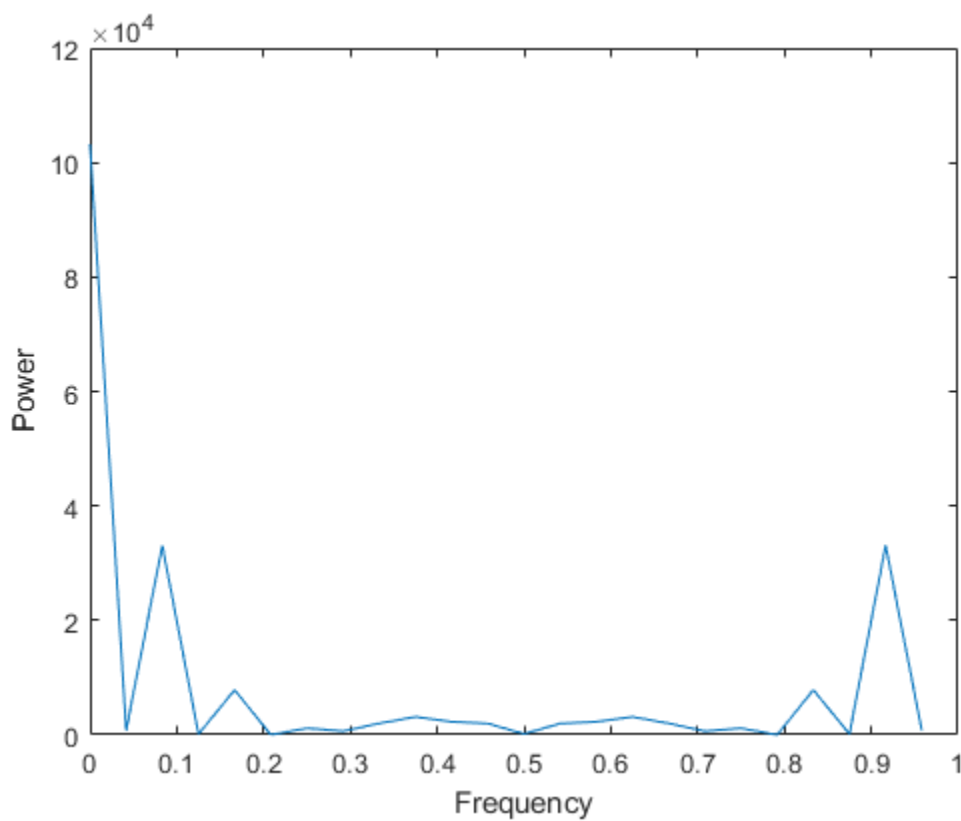
```
mse = 1.9090e+03
```

使用周期图（用 `fft` 函数计算）检查数据周期是否假定为 12 小时：

```
Fs = 1; % Sample frequency (per hour)
n = length(c3); % Window length
Y = fft(c3); % DFT of data
f = (0:n-1)*(Fs/n); % Frequency range
P = Y.*conj(Y)/n; % Power of the DFT
```

```
figure
plot(f,P)
```

```
xlabel('Frequency')  
ylabel('Power')
```



```
predicted_f = 1/12
```

```
predicted_f = 0.0833
```

0.0833 附近的峰值证明此假定是正确的，虽然其出现频率稍微高一点。可以依此相应调整此模型。

另请参阅

`convn` | `corrcoef` | `cov` | `eig` | `fft` | `filter` | `histogram` | `isnan` | `lscov` | `max` | `mean` | `median` | `min` | `mldivide` | `mode` | `plotmatrix` | `polyfit` | `polyval` | `scatter` | `scatter3` | `std` | `var`

图形

- “基本绘图函数” (第 4-2 页)
- “创建网格图和曲面图” (第 4-15 页)
- “显示图像” (第 4-20 页)
- “打印图形” (第 4-23 页)
- “处理图形对象” (第 4-25 页)

基本绘图函数

本节内容
“创建绘图” (第 4-2 页)
“在一幅图形中绘制多个数据集” (第 4-4 页)
“指定线型和颜色” (第 4-6 页)
“绘制线条和标记” (第 4-7 页)
“绘制虚数和复数数据” (第 4-8 页)
“将绘图添加到现有图形中” (第 4-9 页)
“图窗窗口” (第 4-10 页)
“在一幅图窗中显示多个绘图” (第 4-11 页)
“控制轴” (第 4-11 页)
“添加轴标签和标题” (第 4-13 页)
“保存图窗” (第 4-14 页)
“保存工作区数据” (第 4-14 页)

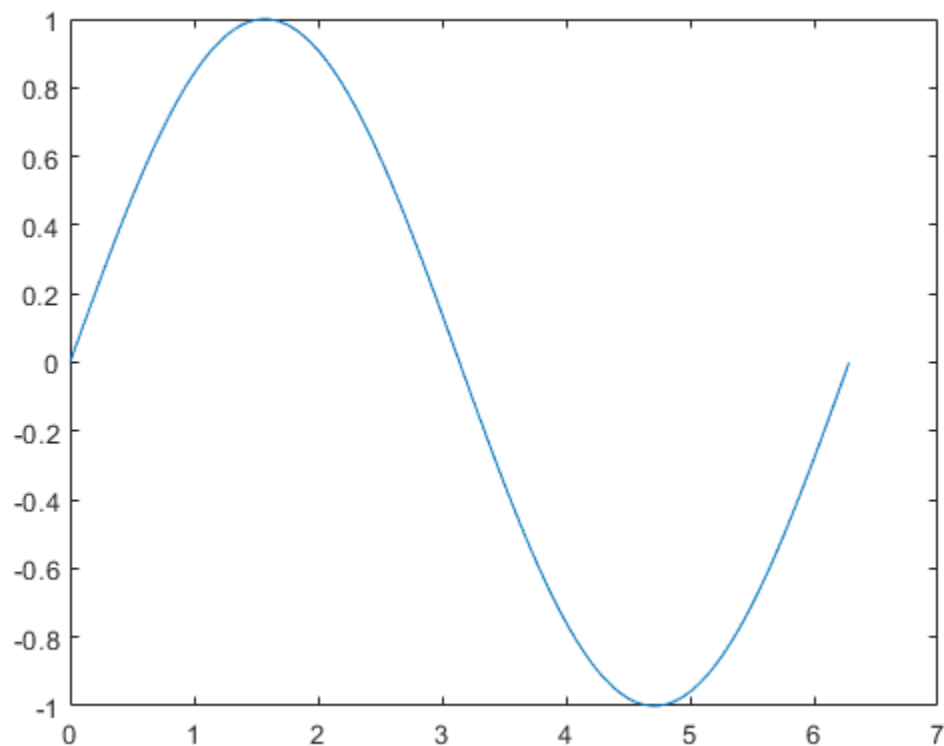
创建绘图

`plot` 函数具有不同的形式，具体取决于输入参数。

- 如果 `y` 是向量，`plot(y)` 会生成 `y` 元素与 `y` 元素索引的分段线图。
- 如果有两个向量被指定为参数，`plot(x,y)` 会生成 `y` 对 `x` 的图形。

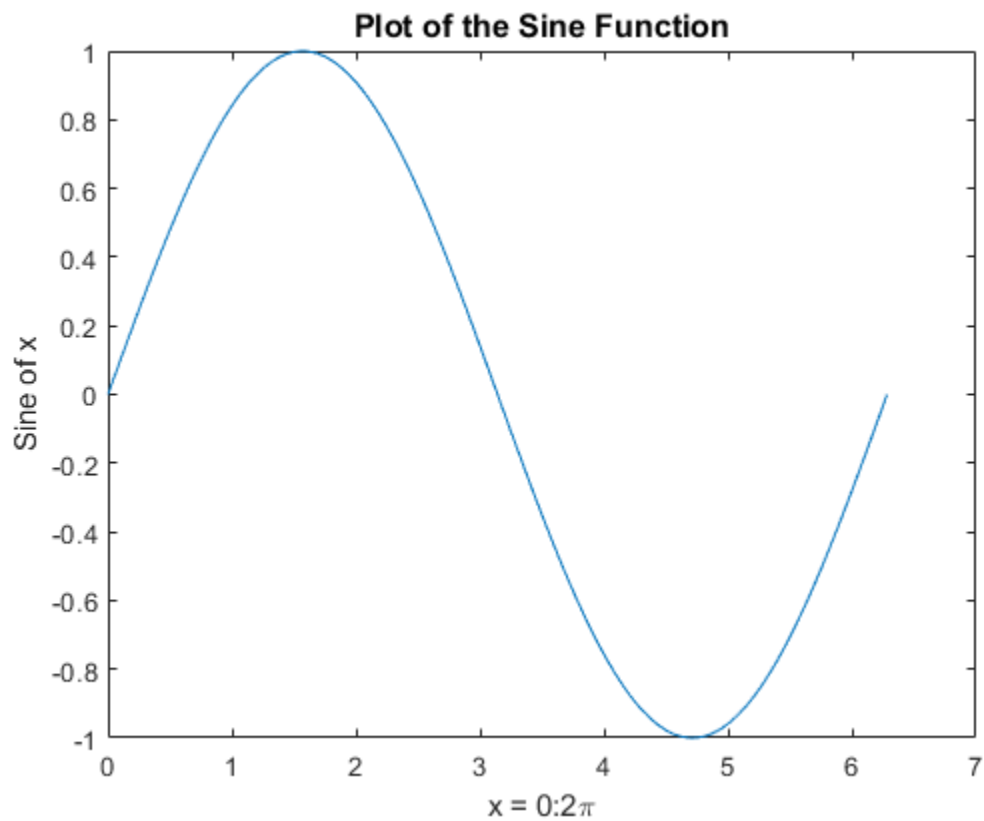
使用冒号运算符创建从 0 至 2π 的 `x` 值向量，计算这些值的正弦，并绘制结果。

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



添加轴标签和标题。`xlabel` 函数中的字符 `\pi` 用于创建符号 π 。`title` 函数中的 `FontSize` 属性用于增大标题所用的文本大小。

```
xlabel('x = 0:2\pi')  
ylabel('Sine of x')  
title('Plot of the Sine Function','FontSize',12)
```

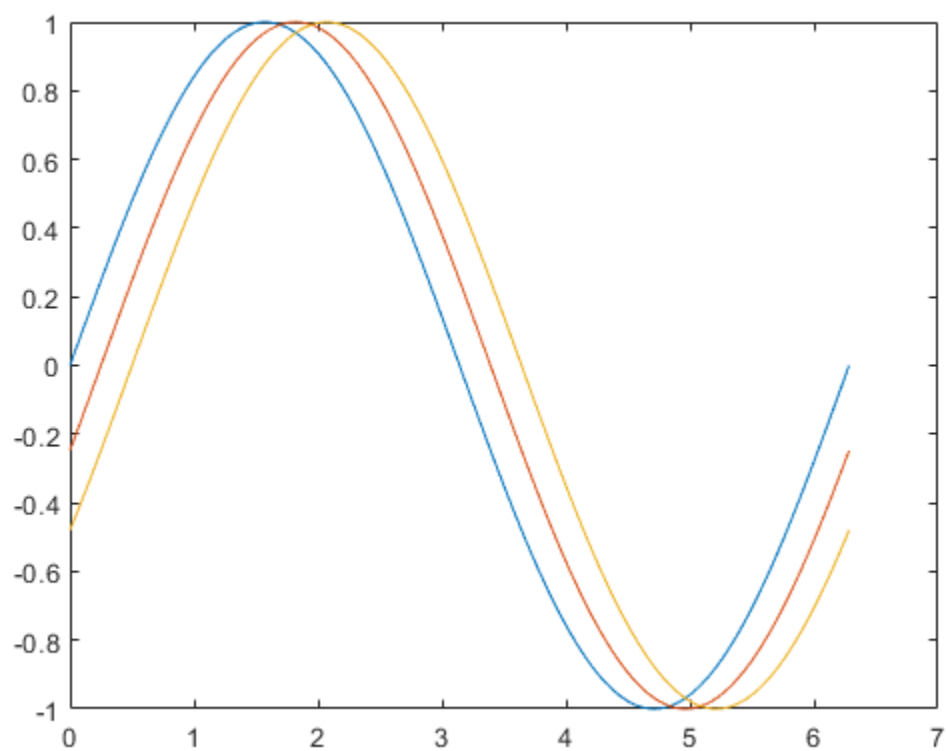


在一幅图形中绘制多个数据集

通过调用一次 `plot`，多个 `x-y` 对组参数会创建多幅图形。MATLAB® 对每条线使用不同的颜色。

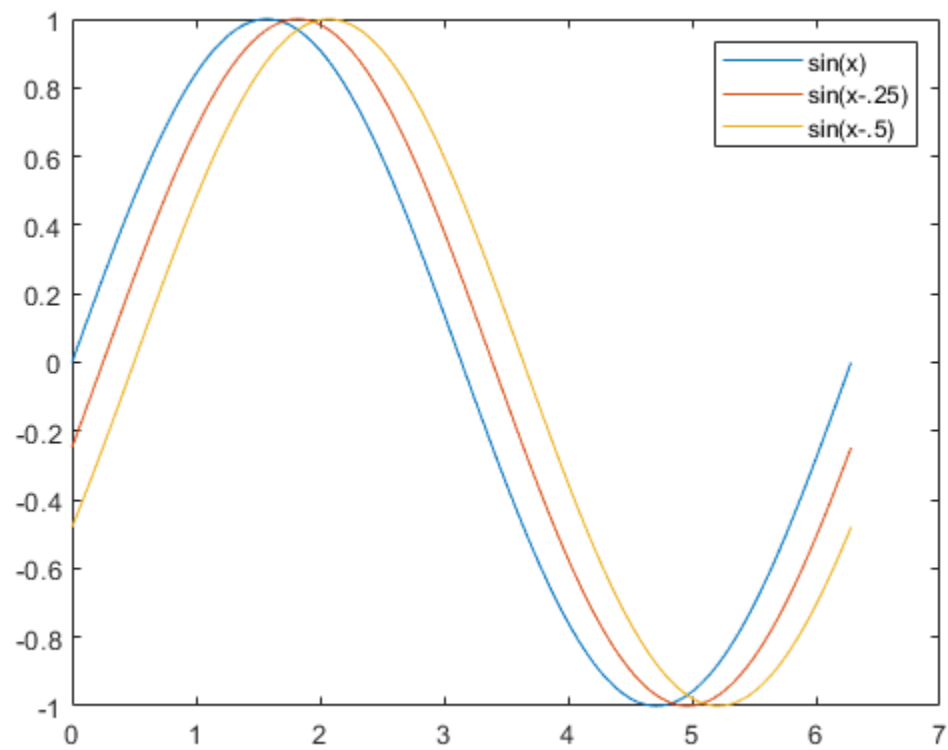
例如，下列语句绘制 `x` 的三个相关函数：

```
x = 0:pi/100:2*pi;  
y = sin(x);  
y2 = sin(x-.25);  
y3 = sin(x-.5);  
plot(x,y,x,y2,x,y3)
```

legend 函数提供了一种标识各条线的简单方法：

```
legend('sin(x)', 'sin(x-.25)', 'sin(x-.5)')
```



指定线型和颜色

使用 `plot` 命令绘制数据时，可以指定颜色、线型和标记（例如加号或圆圈）：

```
plot(x,y,'color_style_marker')
```

`color_style_marker` 包含一至四个字符（包括在单引号中），这些字符根据颜色、线型和标记类型构造而成。例如，

```
plot(x,y,'r:+')
```

使用红色点线绘制数据，并在每个数据点处放置一个 + 标记。

`color_style_marker` 由下列元素的组合形式构成。

类型	值	含义
颜色	'c'	青蓝
	'm'	品红
	'y'	黄
	'r'	红
	'g'	绿
	'b'	蓝
	'w'	白
	'k'	黑

类型	值	含义
线型	'-'	实线
	'--'	虚线
	'.'	点线
	'-.'	点划线
	无字符	没有线条
标记类型	'+'	加号
	'o'	空心圆
	'*'	星号
	'x'	字母 x
	's'	空心正方形
	'd'	空心菱形
	'^'	空心上三角
	'v'	空心下三角
	'>'	空心右三角
	'<'	空心左三角
	'p'	空心五角形
	'h'	空心六角形
	无字符	无标记

绘制线条和标记

如果指定标记类型，但未指定线型，MATLAB 仅使用标记创建图形，而不会创建线条。例如，

```
plot(x,y,'ks')
```

在每个数据点绘制黑色正方形，但不会使用线条连接标记。

语句

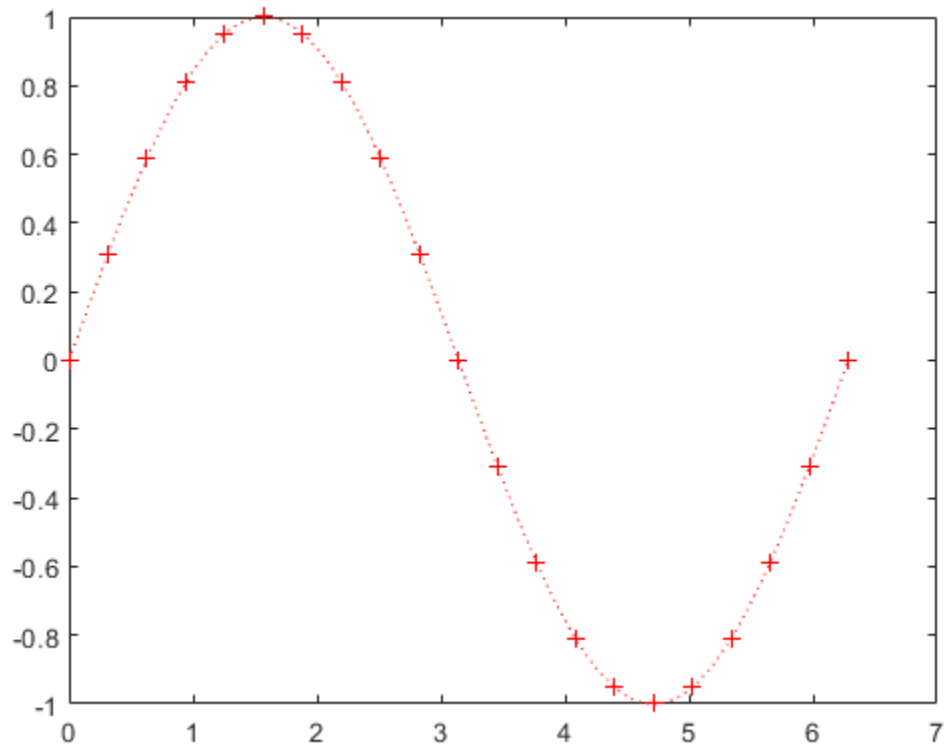
```
plot(x,y,'r:')
```

绘制红色点线，并在每个数据点处放置加号标记。

在每十个数据点处放置标记

此示例展示如何使用比绘制线条所用的数据点更少的数据点来绘制标记。它使用点线图和标记图（分别采用不同数目的数据点）绘制两次数据图：

```
x1 = 0:pi/100:2*pi;
x2 = 0:pi/10:2*pi;
plot(x1,sin(x1),'r:',x2,sin(x2),'r+')
```



绘制虚数和复数数据

将多个复数值作为参数传递给 `plot` 时，MATLAB 会忽略虚部，但传递一个复数参数时除外。对于这一特殊情况，该命令是绘制虚部对实部的图的一种快捷方式。因此，

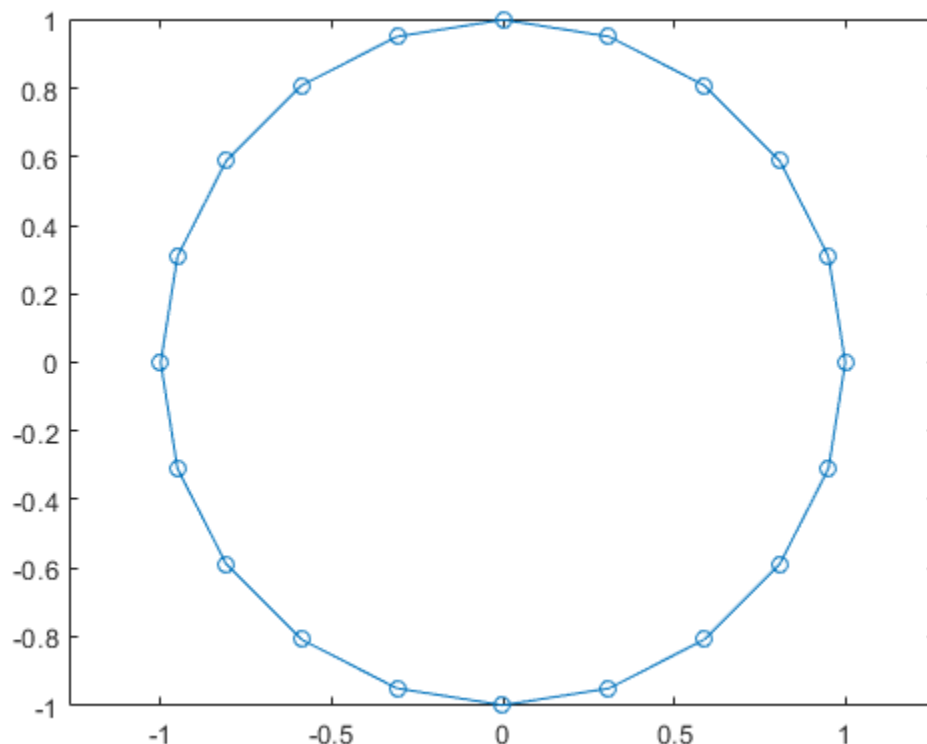
`plot(Z)`

其中 Z 是复数向量或矩阵，等效于

`plot(real(Z),imag(Z))`

下列语句将绘制一个具有 20 条边的多边形，并在各顶点处绘制一个小圆圈。

```
t = 0:pi/10:2*pi;  
plot(exp(1i*t),'-o')  
axis equal
```



`axis equal` 命令使 x 和 y 轴上的各刻度线增量的长度相同，这会使此绘图看起来更加圆润。

将绘图添加到现有图形中

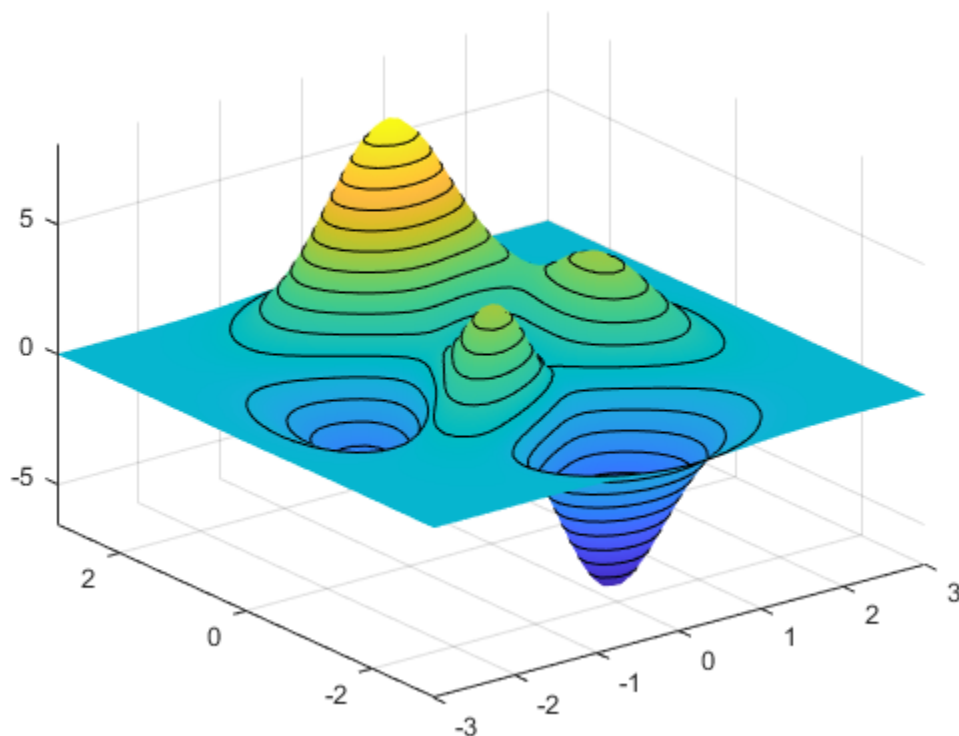
`hold` 命令用于将绘图添加到现有图形中。当键入

`hold on`

时，MATLAB 不会在您发出其他绘图命令时替换现有图形。MATLAB 而会将新图形与当前图形合并在一起。

例如，下列语句首先创建 `peaks` 函数的曲面图，然后叠加同一函数的等高线图：

```
[x,y,z] = peaks;
% Create surface plot
surf(x,y,z)
% Remove edge lines a smooth colors
shading interp
% Hold the current graph
hold on
% Add the contour graph to the pcolor graph
contour3(x,y,z,20,'k')
% Return to default
hold off
```



图窗窗口

如果尚未创建图窗窗口，绘图函数会自动打开一个新的图窗窗口。如果打开了多个图窗窗口，MATLAB 将使用指定为“当前图窗”（通常为上次使用的图窗）的图窗窗口。

要将现有图窗窗口设置为当前的图窗，请将指针放置在该窗口中并点击鼠标，或者也可以键入

figure(n)

其中 **n** 是图窗标题栏中的编号。

要打开新的图窗窗口并将其作为当前图窗，请键入

figure

清空图窗以便创建新绘图

如果某图窗已存在，大多数绘图命令会清除轴并使用此图窗创建新绘图。但是，这些命令不会重置图窗属性，例如，背景色或颜色图。如果已在以前的绘图中设置图窗属性，您可以先使用带有 **reset** 选项的 **clf** 命令。

clf reset

然后创建新绘图，以便将此图窗的属性恢复为其默认值。

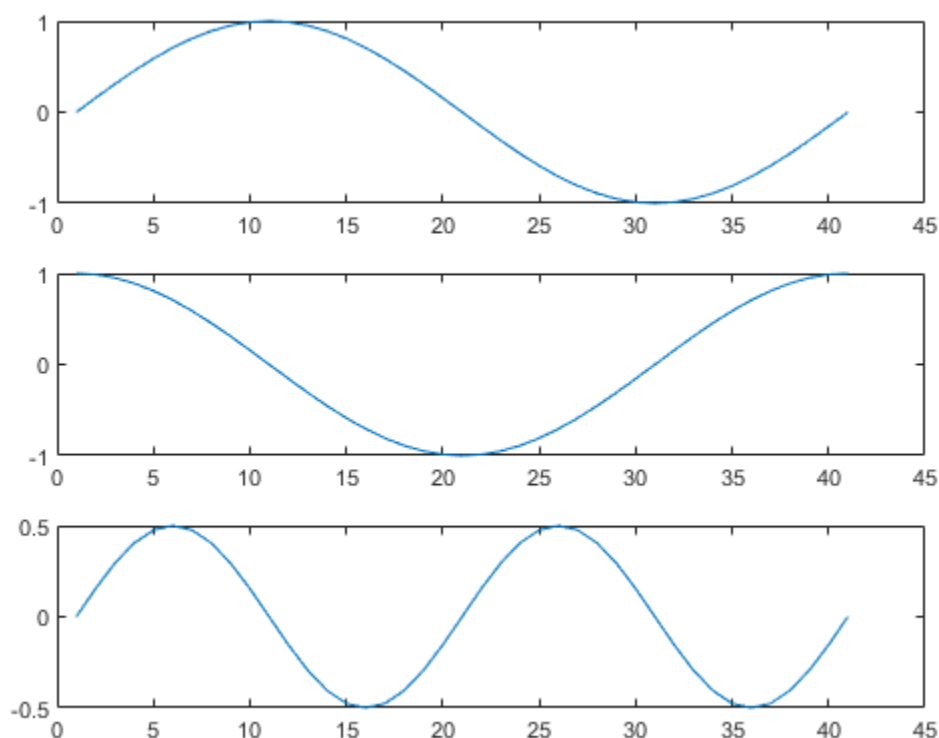
在一幅图窗中显示多个绘图

`subplot` 命令用于在同一窗口中显示多个绘图，或者在同一张纸上打印这些绘图。键入以下命令

```
subplot(m,n,p)
```

会将图窗窗口划分为由多个小子图组成的 $m \times n$ 矩阵，并选择第 p 个子图作为当前绘图。这些绘图沿图窗窗口的第一行进行编号，然后沿第二行进行编号，依此类推。例如，下列语句在图窗窗口的三个子区域中绘制数据：

```
x = 0:pi/20:2*pi;
subplot(3,1,1); plot(sin(x))
subplot(3,1,2); plot(cos(x))
subplot(3,1,3); plot(sin(x).*cos(x))
```



控制轴

`axis` 命令提供了许多用于设置图形的比例、方向和纵横比的选项。

自动改变坐标轴范围和刻度线

默认情况下，MATLAB 查找数据的最大值和最小值，并选择坐标轴范围来覆盖此范围。MATLAB 选择范围和轴刻度线值，以便生成可清楚地显示数据的图形。但是，您可以使用 `axis` 或 `xlim`、`ylim` 与 `zlim` 函数来设置您自己的范围。

注意 更改某根轴的极限会导致其他极限也发生更改，以便更好地表示数据。要禁用自动极限设置，请输入 `axis manual` 命令。

设置坐标轴范围

`axis` 命令用于指定您自己的极限：

```
axis([xmin xmax ymin ymax])
```

或者对于三维图形，

```
axis([xmin xmax ymin ymax zmin zmax])
```

请使用命令

```
axis auto
```

重新启用自动极限选择。

设置轴纵横比

`axis` 命令还可用于指定多种预定义模式。例如，

```
axis square
```

使 x 轴和 y 轴的长度相同。

```
axis equal
```

使 x 轴和 y 轴上的各个刻度线增量的长度相同。这意味着

```
plot(exp(1i*(0:pi/10:2*pi)))
```

(后跟 `axis square` 或 `axis equal`) 会将椭圆形转变为正圆：

```
axis auto normal
```

将轴比例恢复为其默认的自动模式。

设置轴可见性

使用 `axis` 命令可以显示或隐藏轴。

```
axis on
```

显示轴。这是默认设置。

```
axis off
```

隐藏轴。

设置网格线

`grid` 命令启用和禁用网格线。语句

```
grid on
```

启用网格线，而

`grid off`

再次禁用网格线。

添加轴标签和标题

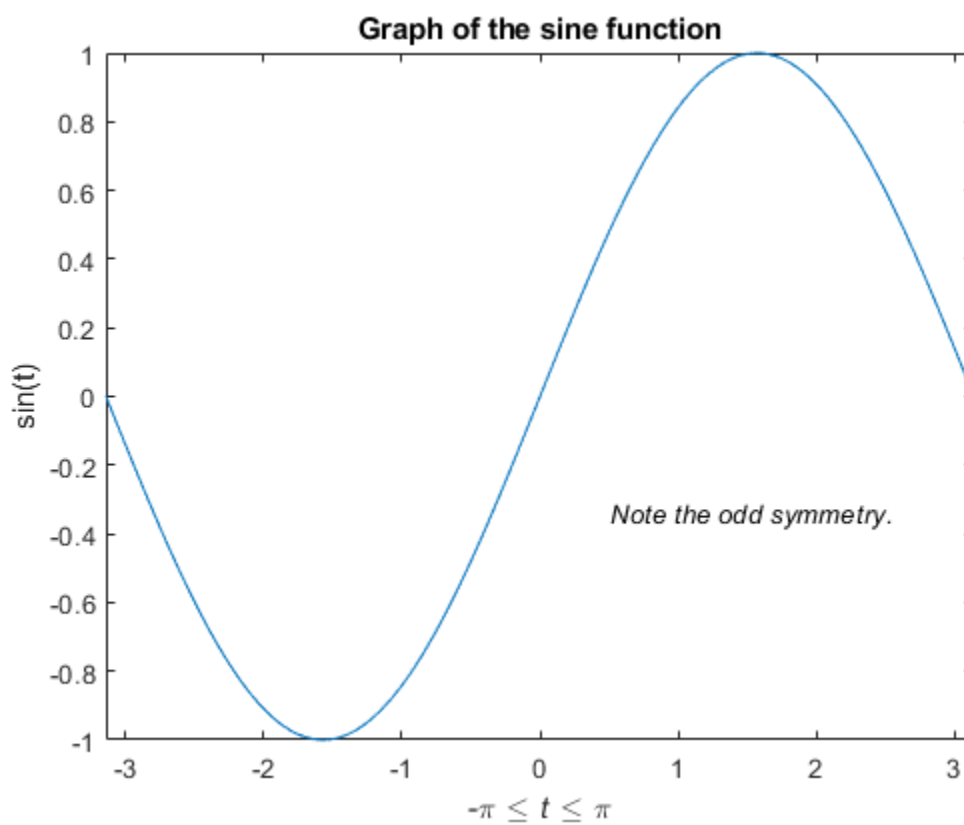
此示例展示如何创建图形并增强其显示：

- 定义 x 和 y 轴的范围 (`axis`)
- 对 x 和 y 轴添加标签 (`xlabel`、`ylabel`)
- 添加标题 (`title`)
- 在图形中添加文本附注 (`text`)

使用 LaTeX 表示法生成数学符号。

```
t = -pi:pi/100:pi;
y = sin(t);
plot(t,y)

axis([-pi pi -1 1])
xlabel('\pi \leq t \leq \pi')
ylabel('sin(t)')
title('Graph of the sine function')
text(0.5,-1/3,'\itNote the odd symmetry.')
```



如需关于在图形中放置箭头、方框和圆圈的信息，请参阅 `annotation` 函数。

保存图窗

通过从**文件**菜单中选择**保存**来保存图窗。这会将图窗写入到文件，包括属性数据、图窗菜单、uicontrol 和所有注释（即整个窗口）。如果这个图窗以前未被保存过，**另存为**对话框则会出现。此对话框提供用于将图窗另存为 .fig 文件或将其导出为图形格式的选项。

如果以前保存过这个图窗，再次使用**保存**会以“静默”方式保存图窗，而**另存为**对话框不会出现。

要使用标准图形格式（例如，TIFF 或 JPG）保存图窗以便用于其他应用程序，请从**文件**菜单中选择**另存为**（如果需要其他控件，则选择**导出设置**）。

注意 当指定保存图窗的格式时，下次保存该图窗或新图窗时，将再次使用该文件格式。如果您不希望按以前使用的格式保存，请使用**另存为**，并确保将**保存类型**下拉菜单设置为要写入的文件类型。

也可通过以下命令行进行保存：

- 使用 `savefig` 函数将图窗及其包含的图形对象保存为 .fig 文件。
- 使用包含任意选项的 `saveas` 命令，以各种格式保存图窗。

加载图窗

您可以使用以下函数将图窗加载到 MATLAB：

- 使用 `openfig` 函数加载保存为 .fig 文件的图窗。
- 使用 `imread` 函数将标准图形文件（包括保存图窗）读入到 MATLAB 中。

生成 MATLAB 代码以便重建图窗

通过从图窗**文件**菜单中选择**生成代码**，可以生成用于重建图窗及其所包含的图形的 MATLAB 代码。如果您已使用绘图工具创建图形，并且希望使用相同或不同数据创建类似图形，此选项尤其有用。

保存工作区数据

通过从图窗**文件**菜单中选择**将工作区另存为**，可以保存工作区中的变量。使用图窗**文件**菜单中的**导入数据**项可以重新加载保存的数据。MATLAB 支持多种数据文件格式，包括 MATLAB 数据文件，该数据文件的扩展名为 .mat。

创建网格图和曲面图

本节内容
“关于网格图和曲面图” （第 4-15 页）
“可视化包含两个变量的函数” （第 4-15 页）

关于网格图和曲面图

MATLAB 在 x-y 平面中的网格上方使用点的 z 坐标来定义曲面图，并使用直线连接相邻的点。`mesh` 和 `surf` 函数以三维形式显示曲面图。

- `mesh` 生成仅使用颜色来标记连接定义点的线条的线框曲面图。
- `surf` 使用颜色显示曲面图的连接线和面。

MATLAB 通过将索引的 z 数据值映射到图窗颜色图来标记曲面图颜色。

可视化包含两个变量的函数

要显示包含两个变量的函数 $z = f(x,y)$,

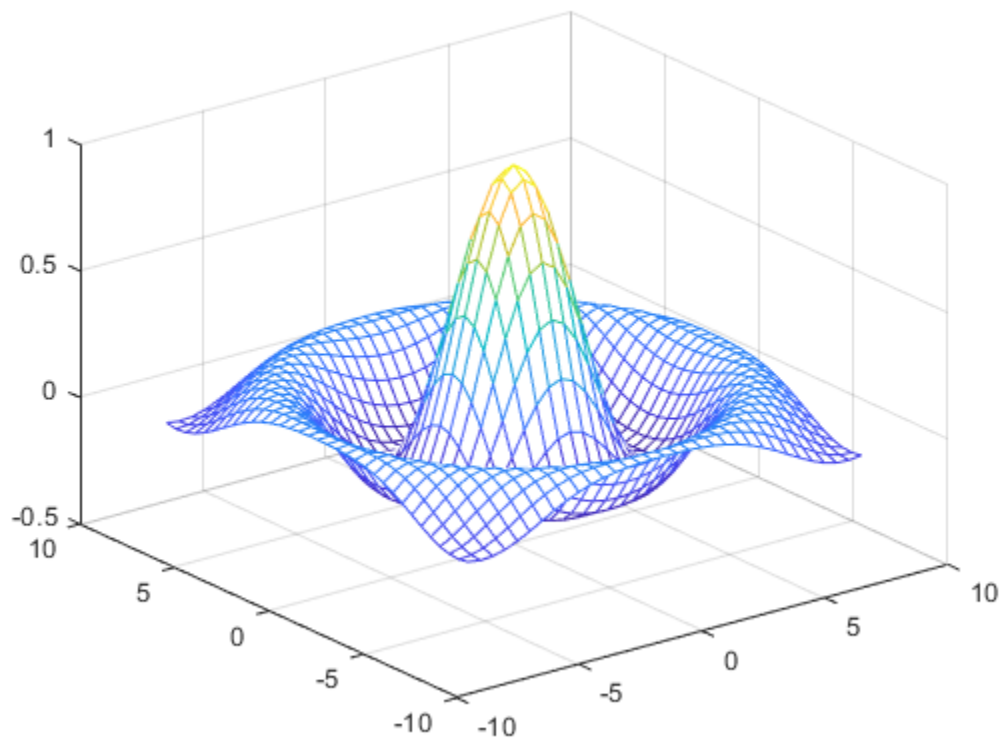
- 1 在此函数的域中，生成分别由重复行和重复列组成的 `X` 和 `Y` 矩阵。
- 2 使用 `X` 和 `Y` 计算此函数并绘制其图形。

`meshgrid` 函数将一个向量或两个向量（即 `x` 和 `y`）指定的域转换为矩阵 `X` 和 `Y`，以便用于计算包含两个变量的函数。`X` 的行是向量 `x` 的副本，`Y` 的列是向量 `y` 的副本。

绘制正弦函数

此示例说明如何计算和绘制 `x` 和 `y` 方向之间的二维 `sinc` 函数 $\sin(R)/R$ 。`R` 是距原点的距离，原点在矩阵的中心。添加 `eps`（非常小的值）可防止网格图在 `R = 0` 处的点出现孔洞。

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```



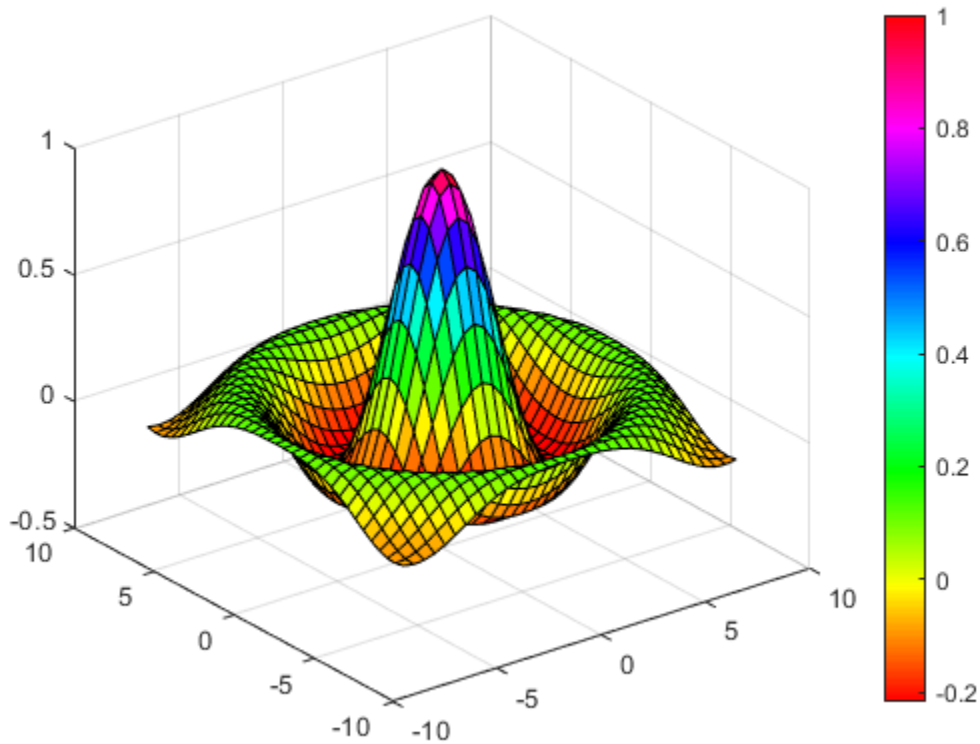
默认情况下，MATLAB 使用当前颜色图来标记网格图颜色。

彩色曲面图

此示例说明如何将 `sinc` 函数绘制为曲面图，指定颜色图并添加颜色栏以便显示数据与颜色的映射。

曲面图与网格图相似，但曲面图的矩形面是彩色的。各个面的颜色由 Z 的值及颜色图（即颜色的有序列表）决定。

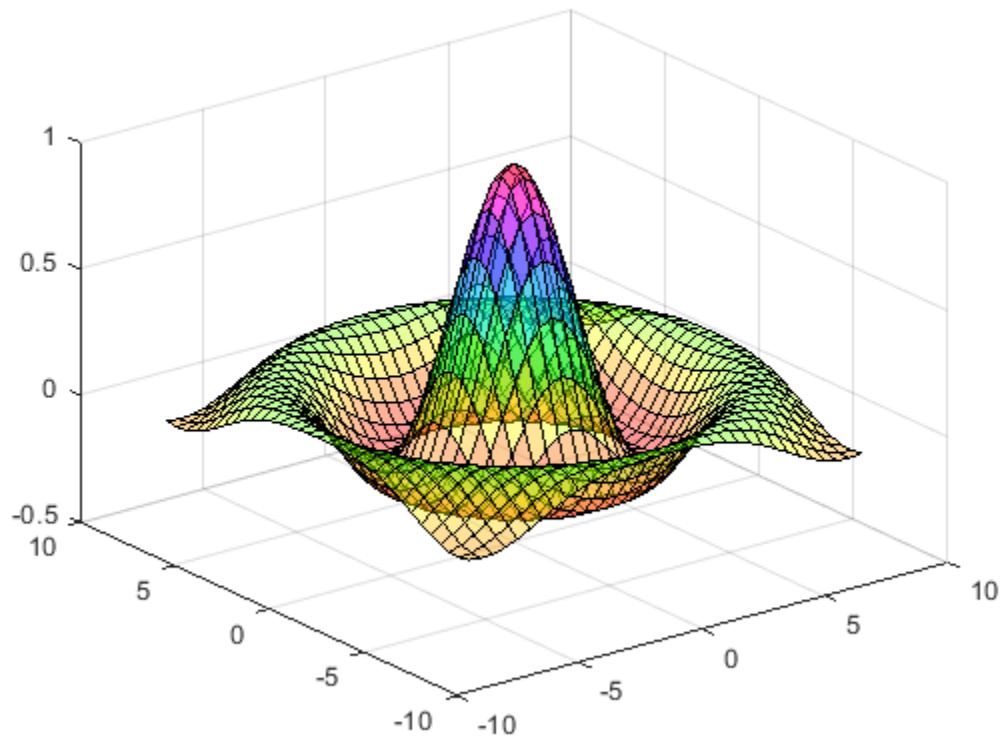
```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
surf(X,Y,Z)  
colormap hsv  
colorbar
```



将曲面图设置为透明

此示例展示如何将曲面图的面设置为不同透明度。透明度（称为 alpha 值）可以针对整个对象进行指定，也可以基于 **alphamap**（其作用方式与颜色图类似）进行指定。

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
surf(X,Y,Z)  
colormap hsv  
alpha(.4)
```

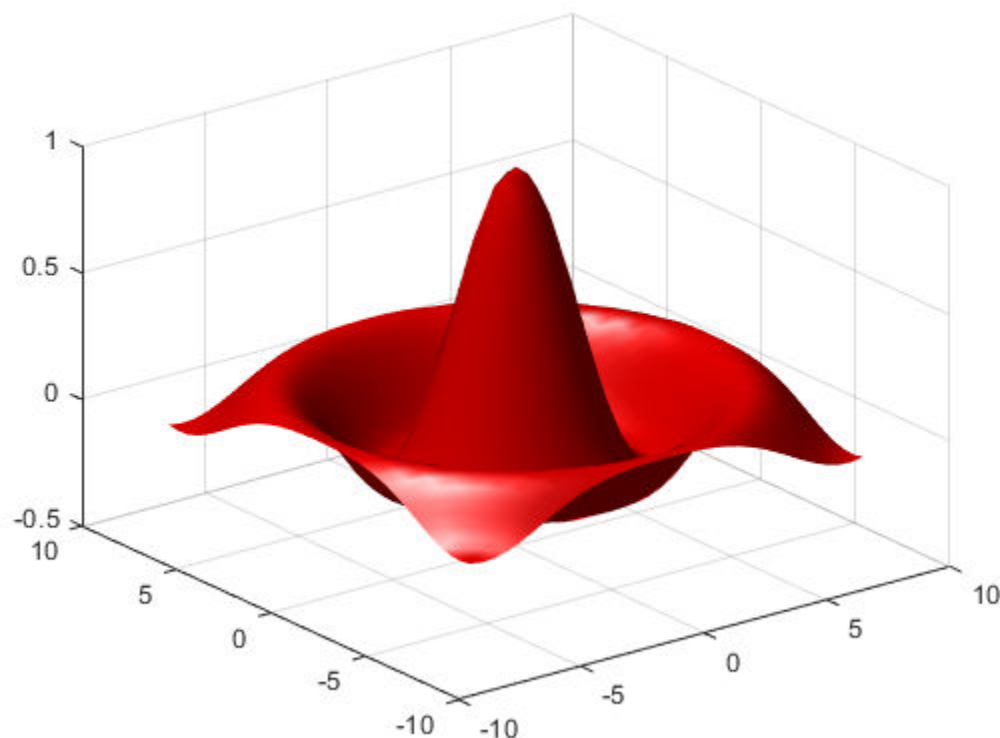


MATLAB 显示一个面 alpha 值为 0.4 的曲面。Alpha 值的范围介于 0（完全透明）和 1（不透明）之间。

使用灯光照亮曲面图

本示例展示的曲面图与前面示例所用的曲面图相同，但将其设置为红色，并删除了网格线。然后会在“照相机”（照相机即为您观察曲面图所处的空间位置）左侧添加一个灯光对象：

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
surf(X,Y,Z,'FaceColor','red','EdgeColor','none')  
camlight left;  
lighting phong
```

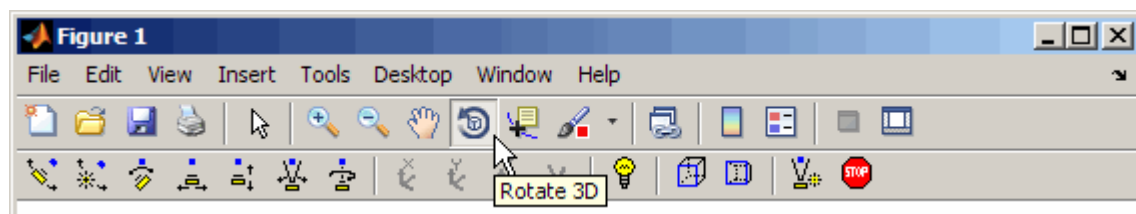


光照是使用定向光源照亮对象的技术。在特定情况下，此技术能够更清楚地显示曲面图形状的细微差异。光照还可用于增添三维图形的真实感。

操作曲面图

图窗工具栏和照相机工具栏提供了以交互方式浏览三维图形的方法。通过从图窗**视图**菜单中选择**照相机工具栏**可以显示照相机工具栏。

下图显示选择了**三维旋转**工具的两个工具栏。



这些工具使您能围绕曲面图对象移动照相机、缩放、添加光照并执行其他查看操作，而不需执行其他命令。

显示图像

本节内容
“图像数据” (第 4-20 页)
“读取和写入图像” (第 4-21 页)

图像数据

您可以将二维数值数组显示为图像。在图像中，数组元素决定了图像的亮度或颜色。例如，加载一个图像数组及其颜色图：

```
load durer
whos
Name      Size      Bytes Class
X         648x509    2638656 double array
caption    2x28         112 char array
map        128x3      3072 double array
```

加载文件 `durer.mat`，向工作区添加三个变量。数组 `X` 是一个 648×509 矩阵，`map` 是作为此图像的颜色图的 128×3 数组。

MAT 文件（例如 `durer.mat`）是用于提供保存 MATLAB 变量的方法的二进制文件。

`X` 的元素是介于 1 和 128 之间的整数，用作颜色图 `map` 的索引。要显示图像，请使用 `imshow` 函数：

```
imshow(X,map)
```

重新生成阿尔布雷特•丢勒的蚀刻板画。



读取和写入图像

使用 `imread` 函数可以读取标准图像文件（TIFF、JPEG、PNG 等）。`imread` 返回的数据类型取决于读取的图像类型。

使用 `imwrite` 函数可以将 MATLAB 数据写入到各种标准图像格式。

打印图形

本节内容
“打印概述” (第 4-23 页)
“从 “文件” 菜单打印” (第 4-23 页)
“将图窗导出到图形文件” (第 4-23 页)
“使用 Print 命令” (第 4-23 页)

打印概述

可以在与计算机连接的打印机上直接打印 MATLAB 图窗，也可以将图窗导出到 MATLAB 支持的某种标准图形文件格式。打印和导出图窗有两种方法：

- 使用**文件**菜单下的**打印**、**打印预览**或**导出设置** GUI 选项。
- 使用 `print` 命令即可通过命令行来打印或导出图窗。

使用 `print` 命令可以更好地控制驱动程序和文件格式。使用“打印预览”对话框可以更好地控制图窗大小、比例、位置和页面标题。

从 “文件” 菜单打印

文件菜单下包含两个与打印相关的菜单选项：

- **打印预览**选项可显示一个对话框，用于在预览输出页面时设置要打印的图窗的布局和图窗样式，您可以从此对话框打印图窗。此对话框包含以前包含在“页面设置”对话框中的选项。
- **打印**选项可显示一个对话框，用于选择打印机、选择标准打印选项和打印图窗。

使用**打印预览**可以确定打印输出是否符合您的要求。点击“打印预览”对话框上的**帮助**按钮将显示如何设置页面的信息。

将图窗导出到图形文件

文件菜单中的**导出设置**选项可打开一个 GUI，用于设置要保存为图形文件的图窗的文本大小、字体和样式等图形特征。“导出设置”对话框用于定义和应用模板以便自定义和标准化输出。设置之后，您可以将图窗导出为多种标准图形文件格式，例如 EPS、PNG 和 TIFF。

使用 Print 命令

`print` 命令为发送给打印机的输出类型提供了更大灵活性，并且允许您通过函数和脚本文件来控制打印。结果可以直接发送到默认打印机，也可以存储在特定的输出文件中。可以使用多种输出格式，包括 TIFF、JPEG 和 PNG。

例如，此语句将当前图窗窗口的内容作为 PNG 图形存储在名为 `magicsquare.png` 的文件中。

```
print -dpng magicsquare.png
```

要以屏幕上的图窗的同等大小保存图窗，请使用下列语句：

```
set(gcf,'PaperPositionMode','auto')
print -dpng -r0 magicsquare.png
```

要将同一图窗存储为 TIFF 文件（分辨率为 200 dpi），请使用以下命令：

```
print -dtiff -r200 magic-square.tif
```

如果在命令行中键入 **print**

```
print
```

将在默认打印机上打印当前图窗。

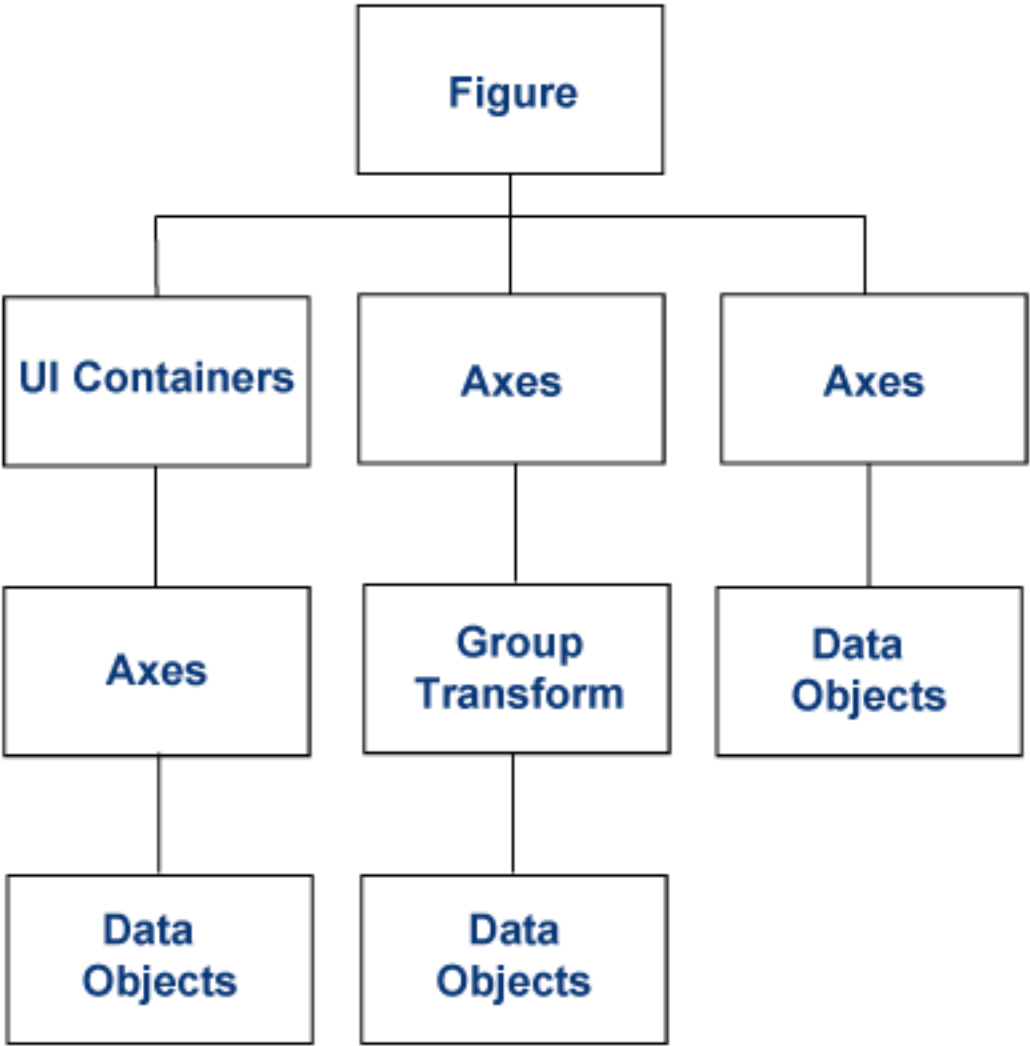
处理图形对象

本节内容

- “图形对象”（第 4-25 页）
- “设置对象属性”（第 4-26 页）
- “用于处理对象的函数”（第 4-28 页）
- “传递参数”（第 4-29 页）
- “查找现有对象的句柄”（第 4-30 页）

图形对象

图形对象是用于显示图形的基本元素。这些对象按层次结构组织，如下图所示。



当调用绘图函数时，MATLAB 使用各种图形对象（例如，图窗窗口、轴、线条、文本等）创建图形。每个对象都具有一组固定的属性，您可以使用这些属性控制图形的行为和外观。

例如，以下语句使用变量 `y` 中的数据创建条形图，并设置关于条形显示方式的属性：

```
y = [75 91 105 123.5 131 150 179 203 226 249 281.5];  
bar(y,'FaceColor','green','EdgeColor','black','LineWidth',1.5)
```

常见图形对象

当调用函数以便创建图形时，MATLAB 会创建图形对象的层次结构。例如，调用 `plot` 函数会创建下列图形对象：

- 图窗 - 包含轴、工具栏、菜单等的窗口。
- 轴 - 包含表示数据的对象的坐标系
- 线条 - 代表传递至 `plot` 函数的数据值的线条。
- 文本 - 用于轴刻度线和可选标题及注释的标签。

不同类型的图形使用不同对象来表示数据。由于存在许多种图形，因此也存在许多数据对象类型。其中一些用于一般用途，例如线条和矩形，还有一些是用于高度专业的用途，例如误差条、颜色栏和图例。

访问对象属性

绘图函数可返回用于创建图形的对象。例如，以下语句将创建一个图形并返回由 `plot` 函数创建的线条对象：

```
x = 1:10;  
y = x.^3;  
h = plot(x,y);
```

使用 `h` 来设置线条对象的属性。例如，设置它的 `Color` 属性。

```
h.Color = 'red';
```

此外，也可以在调用绘图函数时指定线条属性。

```
h = plot(x,y,'Color','red');
```

可以查询线条属性以便查看当前值：

```
h.LineWidth
```

```
ans =  
    0.5000
```

查找对象的属性

要查看对象的属性，请输入：

```
get(h)
```

MATLAB 将返回包含对象属性及当前值的列表。

要查看对象属性及可能的值信息，请输入：

```
set(h)
```

设置对象属性

可使用 `set` 函数一次设置多个属性。

设置现有对象的属性

要对多个对象的同一属性设置相同值，请使用 `set` 函数。

例如，下面的语句绘制一个 5×5 矩阵（创建五个线条对象，每列各一个），然后将 `Marker` 属性设置为正方形，并将 `MarkerFaceColor` 属性设置为绿色。

```
y = magic(5);
h = plot(y);
set(h,'Marker','s','MarkerFaceColor','g')
```

在本示例中，`h` 是一个包含五个句柄的向量，图形中的每个线条（共五个）各一个句柄。`set` 语句将所有线条的 `Marker` 和 `MarkerFaceColor` 属性设置为相同值。

要对一个对象设置属性值，请对句柄数组建立索引：

```
h(1).LineWidth = 2;
```

设置多个属性值

如果要将每个线条的属性设置为不同值，您可以使用元胞数组存储所有数据，并将其传递给 `set` 命令。例如，创建绘图并保存线条句柄：

```
figure
y = magic(5);
h = plot(y);
```

假定您要为每个线条添加不同标记，并使标记的面颜色与线条的颜色相同。您需要定义两个元胞数组，一个包含属性名，另一个包含属性所需的值。

`prop_name` 元胞数组包含两个元素：

```
prop_name(1) = {'Marker'};
prop_name(2) = {'MarkerFaceColor'};
```

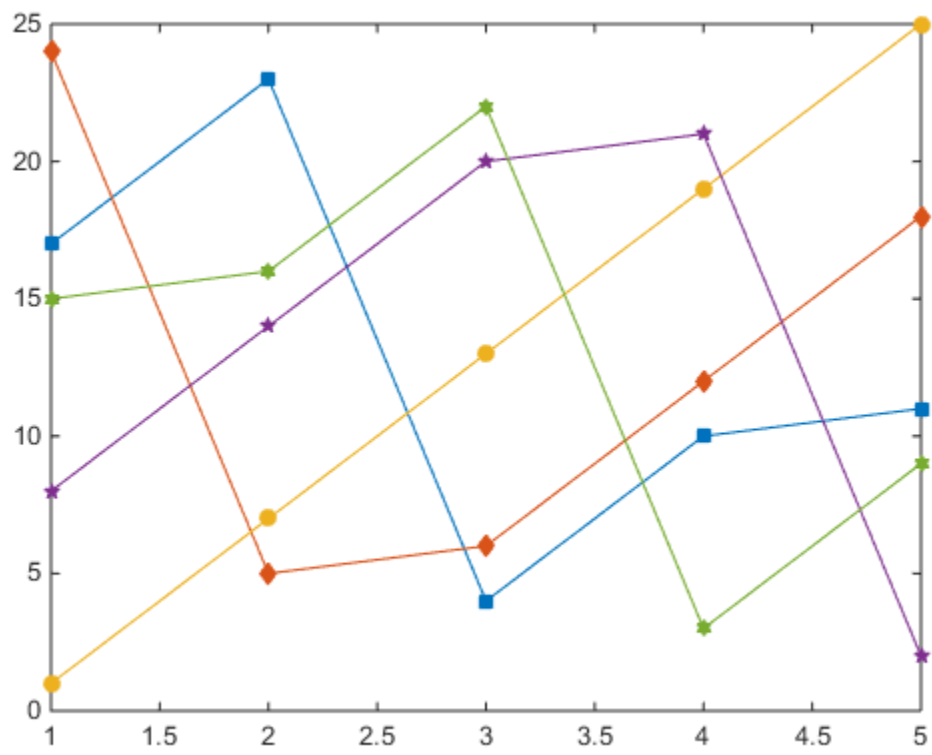
`prop_values` 元胞数组包含 10 个值：`Marker` 属性有 5 个值，`MarkerFaceColor` 属性有 5 个值。请注意，`prop_values` 是一个二维元胞数组。第一个维表示值应用于 `h` 中的哪个句柄，第二个维表示值分配给哪个属性：

```
prop_values(1,1) = {'s'};
prop_values(1,2) = {h(1).Color};
prop_values(2,1) = {'d'};
prop_values(2,2) = {h(2).Color};
prop_values(3,1) = {'o'};
prop_values(3,2) = {h(3).Color};
prop_values(4,1) = {'p'};
prop_values(4,2) = {h(4).Color};
prop_values(5,1) = {'h'};
prop_values(5,2) = {h(5).Color};
```

`MarkerFaceColor` 始终分配到相应线条的颜色的值（通过获取线条 `Color` 属性获得）。

定义元胞数组之后，调用 `set` 以便指定新属性值：

```
set(h,prop_name,prop_values)
```



用于处理对象的函数

此表列出了处理对象时常用的函数。

函数	用途
allchild	查找指定对象的所有子级。
ancestor	查找图形对象的父级。
copyobj	复制图形对象。
delete	删除对象。
findall	查找所有图形对象（包括隐藏句柄）。
findobj	查找具有指定属性值的对象的句柄。
gca	返回当前轴的句柄。
gcf	返回当前图窗的句柄。
gco	返回当前对象的句柄。
get	查询对象的属性的值。
ishandle	如果值是有效对象句柄，则为 True。
set	设置对象的属性的值。

传递参数

可定义专用绘图函数以简化自定义图形的创建过程。通过定义函数，可以像 MATLAB 绘图函数一样传递参数。

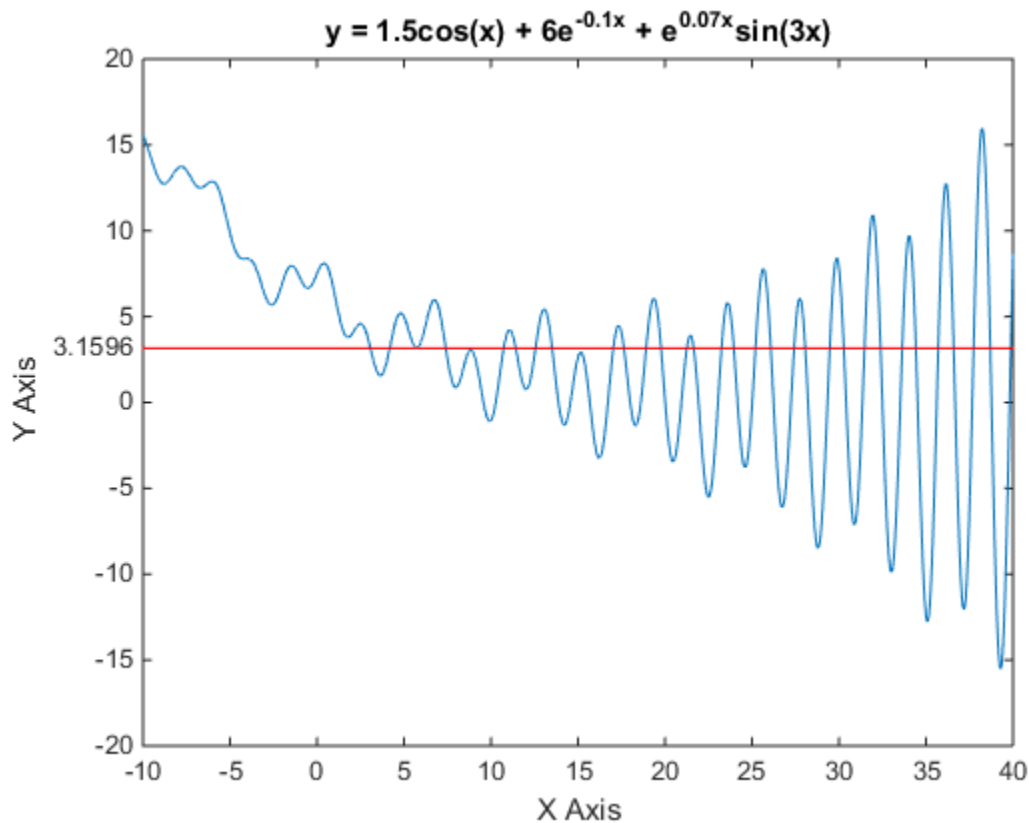
下面的示例显示了一个 MATLAB 函数，该函数在输入参数 x 指定的范围中计算数学表达式，然后绘制结果。第二次调用 `plot` 函数会将结果的 `mean` 值绘制为红线。

该函数基于计算值修改 y 轴刻度。加入轴标签和标题后，即完成了图形自定义。

```
function plotFunc(x)
    y = 1.5*cos(x) + 6*exp(-.1*x) + exp(.07*x).*sin(3*x);
    ym = mean(y);
    hfig = figure('Name','Function and Mean');
    hax = axes('Parent',hfig);
    plot(hax,x,y)
    hold on
    plot(hax,[min(x) max(x)],[ym ym],'Color','red')
    hold off
    ylab = hax.YTick;
    new_ylab = sort([ylab, ym]);
    hax.YTick = new_ylab;
    title('y = 1.5cos(x) + 6e^{-0.1x} + e^{0.07x}sin(3x)')
    xlabel('X Axis')
    ylabel('Y Axis')
end
```

定义输入参数的值，并调用此函数。

```
x = -10:.005:40;
plotFunc(x)
```



查找现有对象的句柄

`findobj` 函数可用于通过搜索具有特定属性值的对象来获取图形对象的句柄。使用 `findobj`，您可以指定任何属性组合的值，这样可以方便地从多个对象中选取一个对象。此外，`findobj` 还可识别正则表达式。

查找特定类型的所有对象

由于所有对象都具有标识对象类型的 `Type` 属性，因此可以查找特定对象类型的所有匹配项的句柄。例如，

```
h = findobj('Type','patch');
```

查找所有补片对象的句柄。

查找具有特定属性的对象

您可以指定多个属性以便缩小搜索范围。例如，

```
plot(rand(5),'r:');  
h = findobj('Type','line','Color','r','LineStyle',':');
```

查找所有红色点线的句柄。

```
h =
```

```
5x1 Line array:
```

```
Line  
Line  
Line  
Line  
Line
```

限制搜索范围

通过将起始图窗或轴的句柄作为传递的第一个参数，您可以在对象层次结构中指定起点。例如，

```
h = findobj(gca,'Type','text','String','\pi/2');
```

仅在当前坐标区中查找 $\pi/2$ 。

编程

- “控制流” (第 5-2 页)
- “脚本和函数” (第 5-7 页)

控制流

本节内容

“条件控制 - if、else、switch” (第 5-2 页)
“循环控制 - for、while、continue、break” (第 5-4 页)
“程序终止 - return” (第 5-5 页)
“向量化” (第 5-6 页)
“预分配” (第 5-6 页)

条件控制 - if、else、switch

条件语句可用于在运行时选择要执行的代码块。最简单的条件语句为 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
```

通过使用可选关键字 elseif 或 else，if 语句可以包含备用选项。例如：

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

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

再者，当您希望针对一组已知值测试相等性时，请使用 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
```

对于 **if** 和 **switch**，MATLAB 执行与第一个 **true** 条件相对应的代码，然后退出该代码块。每个条件语句都需要 **end** 关键字。

一般而言，如果您具有多个可能的离散已知值，读取 **switch** 语句比读取 **if** 语句更容易。但是，无法测试 **switch** 和 **case** 值之间的不相等性。例如，无法使用 **switch** 实现以下类型的条件：

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

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

条件语句中的数组比较

了解如何将关系运算符和 **if** 语句用于矩阵非常重要。如果您希望检查两个变量之间的相等性，您可以使用

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

这是有效的 MATLAB 代码，并且当 **A** 和 **B** 为标量时，此代码会如期运行。但是，当 **A** 和 **B** 为矩阵时，用 **A == B** 不会测试二者是否相等，而会测试二者相等的位置；结果会生成另一个由 0 和 1 构成的矩阵，并显示元素与元素的相等性。

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

```
A == B
```

```
ans =
```

```
4×4 logical array
```

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

检查两个变量之间的相等性的正确方法是使用 **isequal** 函数：

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

isequal 返回 1（表示 **true**）或 0（表示 **false**）的标量逻辑值，而不会返回矩阵，因此能被用于 **if** 函数计算表达式。通过使用上面的 **A** 和 **B** 矩阵，您可以获得

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

```
ans =
```

```
logical
```

```
0
```

下面给出另一示例来重点介绍这一点。如果 A 和 B 为标量，下面的程序永远不会出现“意外状态”。但是对于大多数矩阵对（包括交换列的幻方矩阵），所有元素均不满足任何矩阵条件 $A > B$ 、 $A < B$ 或 $A == B$ ，因此将执行 `else` 子句：

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

有几个函数对减少标量条件的矩阵比较结果以便用于 `if` 非常有用，这些函数包括

```
isequal
isempty
all
any
```

循环控制 - `for`、`while`、`continue`、`break`

此部分涵盖为程序循环提供控制的 MATLAB 函数。

`for`

`for` 循环按预先确定的固定次数重复一组语句。匹配的 `end` 用于界定语句结尾：

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

内部语句的终止分号禁止了循环中的重复输出，循环后的 `r` 显示最终结果。

最好对循环进行缩进处理以便于阅读，特别是使用嵌套循环时：

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

`while`

`while` 在逻辑条件的控制下将一组语句重复无限次。匹配的 `end` 用于界定语句结尾。

下面是一个完整的程序，用于演示如何使用 `while`、`if`、`else` 和 `end` 来寻找区间二分法求多项式的零。

```
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
```



```

    else
        b = x; fb = fx;
    end
end
x

```

结果生成多项式 $x^3 - 2x - 5$ 的根，即

```

x =
    2.09455148154233

```

在 if 语句部分中讨论的与矩阵比较相关的注意事项同样适用于 while 语句。

continue

continue 语句将控制权传递给它所在的 **for** 循环或 **while** 循环的下一迭代，并跳过循环体中的任何其余语句。此道理同样适用于嵌套循环中的 **continue** 语句。也就是说，执行会从遇到 **continue** 语句的循环开头继续。

下面的示例演示的 **magic.m** 循环计算文件中的代码行数目的 **continue** 循环，并跳过所有空行和注释。**continue** 语句用于前进到 **magic.m** 中的下一行，而不会在遇到空行或注释行时增加行计数：

```

fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) || strncmp(line,'% ',1) || ~ischar(line)
        continue
    end
    count = count + 1;
end
fprintf('%d lines\n',count);
fclose(fid);

```

break

break 语句用于提前从 **for** 循环或 **while** 循环中退出。在嵌套循环中，**break** 仅从最里面的循环退出。

下面对前述部分中的示例进行了改进。使用此 **break** 的优点是什么？

```

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

```

程序终止 - return

此部分包含用于在程序运行完成之前终止程序的 MATLAB **return** 函数。

return

return 终止当前命令序列，并将控制权返回给调用函数或键盘。此外，**return** 还用于终止 **keyboard** 模式。被调用的函数通常在到达函数末尾时将控制权转交给调用它的函数。您可以在被调用的函数中插入一个 **return** 语句，以便强制提前终止并将控制权转交给调用函数。

向量化

提高 MATLAB 程序的运行速度的一种方法是向量化构造程序时所使用的算法。其他编程语言可使用 **for** 循环或 **DO** 循环，而 MATLAB 可使用向量或矩阵运算。下面提供了一个与创建算法表相关的简单示例：

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

相同代码段的向量化版本为

```
x = .01:.01:10;
y = log10(x);
```

对于更复杂的代码，向量化选项并不总是这么明显。

预分配

如果无法向量化某段代码，可以通过预分配存储输出结果的任何向量或数组来提高 **for** 循环的运行速度。例如，此代码使用函数 **zeros** 来预分配在 **for** 循环中创建的向量。这显著提高了 **for** 循环的执行速度：

```
r = zeros(32,1);
for n = 1:32
    r(n) = rank(magic(n));
end
```

如果未经过上述示例中的预分配，MATLAB 解释器会在每次遍历循环时将 **r** 向量增大一个元素。向量预分配避免了此步骤，并提高了执行速度。

脚本和函数

本节内容
“概述” (第 5-7 页)
“脚本” (第 5-7 页)
“函数” (第 5-8 页)
“函数类型” (第 5-9 页)
“全局变量” (第 5-10 页)
“命令与函数语法” (第 5-11 页)

概述

MATLAB 提供了一个强大的编程语言和交互式计算环境。您可以使用此语言在 MATLAB 命令行中一次输入一个命令，也可以向某个文件写入一系列命令，按照执行任何 MATLAB 函数的相同方式来执行这些命令。使用 MATLAB 编辑器或任何其他文件编辑器可以创建您自己的函数文件。按照调用任何其他 MATLAB 函数或命令的相同方式来调用这些函数。

两种程序文件：

- 脚本，不接受输入参数或返回输出参数。它们处理工作区中的数据。
- 函数，可接受输入参数，并返回输出参数。内部变量是函数的局部变量。

如果您是新 MATLAB 程序员，您只需在当前文件夹中创建您希望尝试的程序文件。当您创建的文件越来越多时，您可能希望将这些文件组织到其他文件夹和个人工具箱，以便将其添加到您的 MATLAB 搜索路径中。

如果您复制多个函数名称，MATLAB 会执行在搜索路径中显示的第一个函数。

要查看程序文件（例如，`myfunction.m`）的内容，请使用

`type myfunction`

脚本

当调用脚本时，MATLAB 仅执行在文件中找到的命令。脚本可以处理工作区中的现有数据，也可以创建要在其中运行脚本的新数据。尽管脚本不会返回输出参数，其创建的任何变量都会保留在工作区中，以便在后续计算中使用。此外，脚本可以使用 `plot` 等函数生成图形输出。

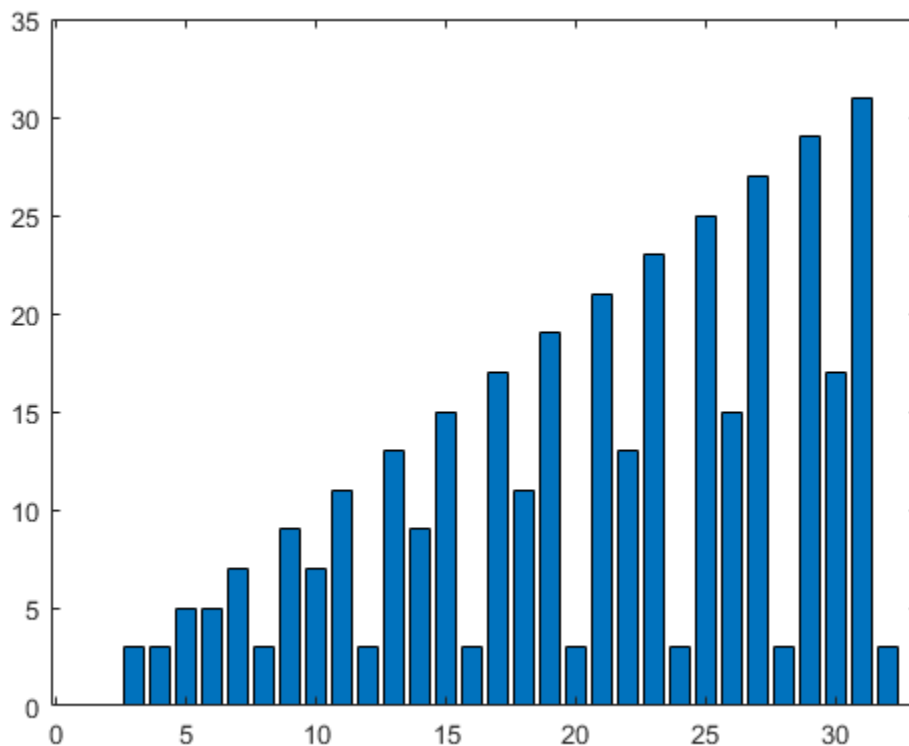
例如，创建一个名为 `magicrank.m` 的文件，该文件包含下列 MATLAB 命令：

```
% Investigate the rank of magic squares
r = zeros(1,32);
for n = 3:32
    r(n) = rank(magic(n));
end
bar(r)
```

键入语句

`magicrank`

使 MATLAB 执行命令、计算前 30 个幻方矩阵的秩，并绘制结果的条形图。执行完文件之后，变量 **n** 和 **r** 将保留在工作区中。



函数

函数是可接受输入参数并返回输出参数的文件。文件名和函数名称应当相同。函数处理其自己的工作区中的变量，此工作区不同于您在 MATLAB 命令提示符下访问的工作区。

rank 提供了一个很好的示例。文件 **rank.m** 位于文件夹

toolbox/matlab/matfun

您可以使用以下命令查看文件

type rank

下面列出了此文件：

```
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);
```

```
if nargin==1
    tol = max(size(A)) * max(s) * eps;
end
r = sum(s > tol);
```

函数的第一行以关键字 **function** 开头。它提供函数名称和参数顺序。本示例中具有两个输入参数和一个输出参数。

第一个空行或可执行代码行前面的后续几个行是提供帮助文本的注释行。当键入以下命令时，会输出这些行

```
help rank
```

帮助文本的第一行是 H1 行，当对文件夹使用 **lookfor** 命令或请求 **help** 时，MATLAB 会显示此行。

文件的其余部分是用于定义函数的可执行 MATLAB 代码。函数体中引入的变量 **s** 以及第一行中的变量（即 **r**、**A** 和 **tol**）均为函数的局部变量；他们不同于 MATLAB 工作区中的任何变量。

本示例演示了 MATLAB 函数不同于其他编程语言函数的一个方面，即可变数目的参数。可以采用多种不同方法使用 **rank** 函数：

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

许多函数都按此方式运行。如果未提供输出参数，结果会存储在 **ans** 中。如果未提供第二个输入参数，此函数会运用默认值进行计算。函数体中提供了两个名为 **nargin** 和 **nargout** 的数量，用于告知与函数的每次特定使用相关的输入和输出参数的数目。**rank** 函数使用 **nargin**，而不需要使用 **nargout**。

函数类型

MATLAB 提供了多种不同函数用于编程。

匿名函数

匿名函数是一种简单形式的 MATLAB 函数，该函数在一个 MATLAB 语句中定义。它包含一个 MATLAB 表达式和任意数目的输入和输出参数。您可以直接在 MATLAB 命令行中定义匿名函数，也可以在函数或脚本中定义匿名函数。这样，您可以快速创建简单函数，而不必每次为函数创建文件。

根据表达式创建匿名函数的语法为

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

下面的语句创建一个求某个数字的平方的匿名函数。当调用此函数时，MATLAB 会将您传入的值赋值给变量 **x**，然后在方程 **x.^2** 中使用 **x**：

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

要执行 **sqr** 函数，请键入

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

主函数和局部函数

任何非匿名函数必须在文件中定义。每个此类函数文件都包含一个必需的主函数（最先显示）和任意数目的局部函数（位于主函数后面）。主函数的作用域比局部函数更广。因此，主函数可以从定义这些函数的

文件外（例如，从 MATLAB 命令行或从其他文件的函数中）调用，而局部函数则没有此功能。局部函数仅对其自己的文件中的主函数和其他局部函数可见。

“函数”（第 5-8 页）部分中显示的 `rank` 函数就是一个主函数的示例。

私有函数

私有函数是一种主函数。其特有的特征是：仅对一组有限的其他函数可见。如果您希望限制对某个函数的访问，或者当您选择不公开某个函数的实现时，此种函数非常有用。

私有函数位于带专有名称 `private` 的子文件夹中。它们是仅可在母文件夹中可见的函数。例如，假定文件夹 `newmath` 位于 MATLAB 搜索路径中。`newmath` 的名为 `private` 子文件夹可包含只能供 `newmath` 中的函数调用的特定函数。

由于私有函数在父文件夹外部不可见，因此可以使用与其他文件夹中的函数相同的名称。如果您希望创建您自己的特定函数的版本，并在其他文件夹中保留原始函数，此功能非常有用。由于 MATLAB 在标准函数之前搜索私有函数，因此在查找名为 `test.m` 的非私有文件之前，它将查找名为 `test.m` 的私有函数。

嵌套函数

您可以在函数体中定义其他函数。这些函数称为外部函数中的嵌套函数。嵌套函数包含任何其他函数的任何或所有组成部分。在本示例中，函数 `B` 嵌套在函数 `A` 中：

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

与其他函数一样，嵌套函数具有其自己的工作区，可用于存储函数所使用的变量。但是，它还可以访问其嵌套在的所有函数的工作区。因此，举例来说，主函数赋值的变量可以由嵌套在主函数中的任意级别的函数读取或覆盖。类似地，嵌套函数中赋值的变量可以由包含该函数的任何函数读取或被覆盖。

全局变量

如果您想要多个函数共享一个变量副本，只需在所有函数中将此变量声明为 `global`。如果您想要基础工作区访问此变量，请在命令行中执行相同操作。全局声明必须在函数中实际使用变量之前进行。全局变量名称使用大写字母有助于将其与其他变量区分开来，但这不是必需的。例如，在名为 `falling.m` 的文件创建一个新函数：

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

然后，以交互方式输入语句

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

通过上述两条全局语句，可以在函数内使用在命令提示符下赋值给 `GRAVITY` 的值。然后，您可以按交互方式修改 `GRAVITY` 并获取新解，而不必编辑任何文件。

命令与函数语法

您可以编写接受字符参数的 MATLAB 函数，而不必使用括号和引号。也就是说，MATLAB 将

```
foo a b c
```

解释为

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

但是，当使用不带引号的命令格式时，MATLAB 无法返回输出参数。例如，

```
legend apples oranges
```

使用 **apples** 和 **oranges** 作为标签在绘图上创建图例。如果您想要 **legend** 命令返回其输出参数，必须使用带引号的格式：

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

此外，如果其中任一参数不是字符向量，必须使用带引号的格式。

小心 虽然不带引号的命令语法非常方便，但在某些情况下可能会出现使用不当的情形，而 MATLAB 并不会产生错误信息。

在代码中构造字符参数

带引号的函数格式可用于在代码中构造字符参数。下面的示例处理多个数据文件，即 **August1.dat**、**August2.dat** 等。它使用函数 **int2str**，该函数将整数转换为字符以便生成文件名：

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

