Study on MatLab

# <https://www.mathworks.com/help/matlab/getting-started-with-matlab.html>

# Getting Started with MATLAB

* [MATLAB Product Description](https://www.mathworks.com/help/matlab/learn_matlab/product-description.html)
* [System Requirements](http://www.mathworks.com/products/availability/index.html#ML)

## Tutorials

[**Desktop Basics**](https://www.mathworks.com/help/matlab/learn_matlab/desktop.html)

Enter statements at the command line and view results.

[**Matrices and Arrays**](https://www.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html)

MATLAB® operates primarily on arrays and matrices, both in whole and in part. A matrix is a two-dimensional array often used for linear algebra.

[**Array Indexing**](https://www.mathworks.com/help/matlab/learn_matlab/array-indexing.html)

Variables in MATLAB are typically arrays that can hold many numbers. When you want to access selected elements of an array, use indexing.

[**Workspace Variables**](https://www.mathworks.com/help/matlab/learn_matlab/workspace.html)

The workspace contains variables that you create within or import into MATLAB from data files or other programs.

[**Text and Characters**](https://www.mathworks.com/help/matlab/learn_matlab/character-strings.html)

When you are working with text, enclose sequences of characters in single quotes.

[**Calling Functions**](https://www.mathworks.com/help/matlab/learn_matlab/calling-functions.html)

MATLAB provides a large number of functions that perform computational tasks. To call a function, enclose its input arguments in parentheses.

[**2-D and 3-D Plots**](https://www.mathworks.com/help/matlab/learn_matlab/plots.html)

Graphics functions include 2-D and 3-D plotting functions to visualize data and communicate results.

[**Programming and Scripts**](https://www.mathworks.com/help/matlab/learn_matlab/scripts.html)

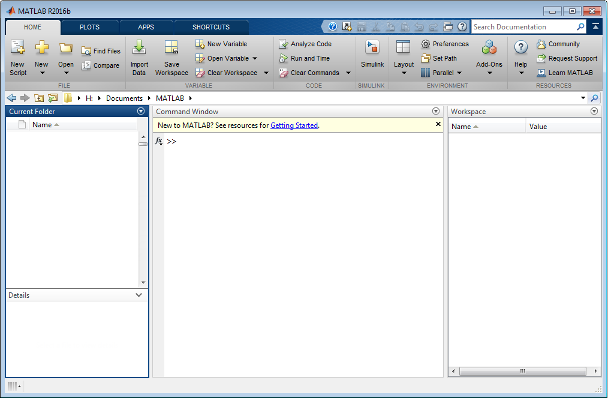
The simplest type of MATLAB program is called a script. A script contains a sequence of commands and function calls.

[**Help and Documentation**](https://www.mathworks.com/help/matlab/learn_matlab/help.html)

All functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax.

1. Desktop Basics

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



The desktop includes these panels:

* **Current Folder** — Access your files.
* **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
* **Workspace** — Explore data that you create or import from files.

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

a = 1

MATLAB adds variable a to the workspace and displays the result in the Command Window.

a =

1

Create a few more variables.

b = 2

b =

2

c = a + b

c =

3

d = cos(a)

d =

0.5403

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

sin(a)

ans =

0.8415

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

e = a\*b;

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

**Next in Getting Started with MATLAB**

[Matrices and Arrays](https://www.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html)

## Matrices and Arrays

[Open Script](matlab:openExample('matlab/MatricesAndArraysGSExample'))

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

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

### Array Creation

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

a = [1 2 3 4]

a =

1 2 3 4

This type of array is a row vector.

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

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

a =

1 2 3

4 5 6

7 8 10

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

z = zeros(5,1)

z =

0

0

0

0

0

### Matrix and Array Operations

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

a + 10

ans =

11 12 13

14 15 16

17 18 20

sin(a)

ans =

0.8415 0.9093 0.1411

-0.7568 -0.9589 -0.2794

0.6570 0.9894 -0.5440

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

a'

ans =

1 4 7

2 5 8

3 6 10

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

p = a\*inv(a) (求矩阵的逆，方阵)

p =

1.0000 0 -0.0000

0 1.0000 0

0 0 1.0000

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

format long

p = a\*inv(a)

p =

1.000000000000000 0 -0.000000000000000

0 1.000000000000000 0

0 0 0.999999999999998

Reset the display to the shorter format using

format short

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

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

p = a.\*a

p =

1 4 9

16 25 36

49 64 100

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

a.^3 （幂）

ans =

1 8 27

64 125 216

343 512 1000

### Concatenation（连接）

Concatenation is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

A = [a,a] 连接

A =

1 2 3 1 2 3

4 5 6 4 5 6

7 8 10 7 8 10

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

A = [a; a]

A =

1 2 3

4 5 6

7 8 10

1 2 3

4 5 6

7 8 10

### Complex Numbers(复数)

Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of -1. （i.^2= -1）

sqrt(-1) --求平方根

ans =

0.0000 + 1.0000i

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

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

c =

3.0000 + 4.0000i 4.0000 + 3.0000i

0.0000 - 1.0000i 0.0000 +10.0000i

## Next in Getting Started with MATLAB

[Array Indexing](https://www.mathworks.com/help/matlab/learn_matlab/array-indexing.html)

## 3. Array Indexing

[Open Script](matlab:openExample('matlab/ArrayIndexingGSExample'))

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

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

A = magic(4)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

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

A(4,2) -小括号取元素

ans =

14

Less common, but sometimes useful, is to use a single subscript that traverses down each column in order:（一列一列累加起来）

A(8)

ans =

14

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

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

test = A(4,5) –取元素越界会报错

Index exceeds matrix dimensions.

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

A(4,5) = 17 –越界定义元素，可以

A =

16 2 3 13 0

5 11 10 8 0

9 7 6 12 0

4 14 15 1 17

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

A(1:3,2) （行，列）

ans =

2

11

7

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

A(3,:)

ans =

9 7 6 12 0

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

B = 0:10:100

B =

0 10 20 30 40 50 60 70 80 90 100

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

## Next in Getting Started with MATLAB

[Workspace Variables](https://www.mathworks.com/help/matlab/learn_matlab/workspace.html)

## 4. Workspace Variables

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

A = magic(4);

B = rand(3,5,2);

You can view the contents of the workspace using whos.

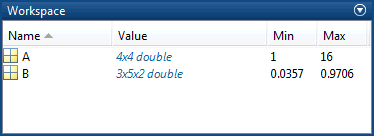
whos

Name Size Bytes Class Attributes

A 4x4 128 double

B 3x5x2 240 double

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



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

save myfile.mat

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

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

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

load myfile.mat

## Next in Getting Started with MATLAB

[Text and Characters](https://www.mathworks.com/help/matlab/learn_matlab/character-strings.html)

## 5. Text and Characters

[Open Script](matlab:openExample('matlab/CharacterStringsGSExample'))

When you are working with text, enclose sequences of characters in single quotes. You can assign text to a variable.

myText = 'Hello, world';

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

otherText = 'You**''**re right'

otherText =

You're right

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

whos myText

Name Size Bytes Class Attributes

myText 1x12 24 char

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

longText = [myText,' - ',otherText]

longText =

Hello, world - You're right

To convert numeric values to characters, use functions, such as num2str or int2str.（数字与字符串转换）

f = 71;

c = (f-32)/1.8;

tempText = ['Temperature is ',num2str(c),'C']

tempText =

Temperature is 21.6667C

## Next in Getting Started with MATLAB

[Calling Functions](https://www.mathworks.com/help/matlab/learn_matlab/calling-functions.html)

## 6. Calling Functions

[Open Script](matlab:openExample('matlab/CallingFunctionsGSExample'))

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

To call a function, such as max, enclose its input arguments in parentheses:

A = [1 3 5];

max(A)

ans =

5

If there are multiple input arguments, separate them with commas:

B = [10 6 4];

max(A,B) – 对应列相比较，每列返回一个最大值

ans =

10 6 5

Return output from a function by assigning it to a variable:

maxA = max(A)

maxA =

5

When there are multiple output arguments, enclose them in square brackets:

[maxA,location] = max(A) –一个最大值及其位置

maxA =

5

location =

3

Enclose any character inputs in single quotes:

disp('hello world')

hello world

To call a function that does not require any inputs and does not return any outputs, type only the function name:

clc

The clc function clears the Command Window.

## Next in Getting Started with MATLAB

[2-D and 3-D Plots](https://www.mathworks.com/help/matlab/learn_matlab/plots.html)

## 7. 2-D and 3-D Plots

### Line Plots

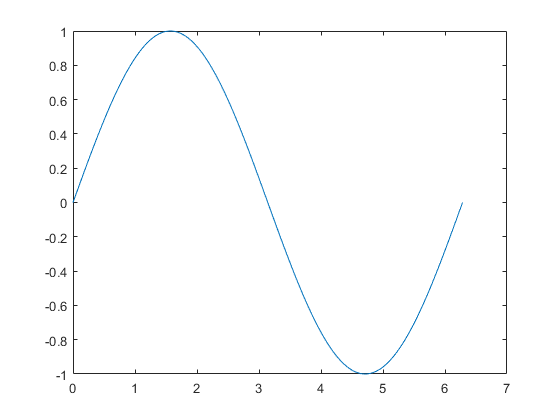
[Open Script](matlab:openExample('matlab/LinePlotsGSExample'))

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

x = 0:pi/100:2\*pi;

y = sin(x);

plot(x,y)

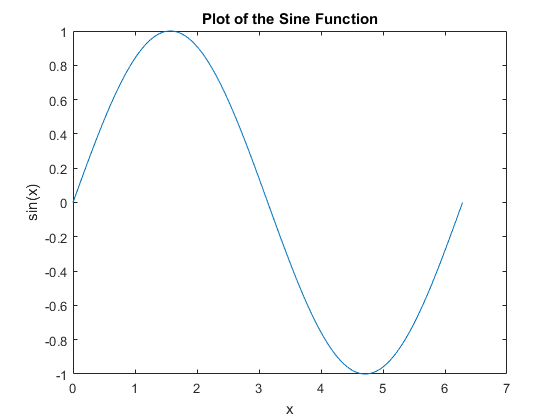


You can label the axes and add a title.

xlabel('x')

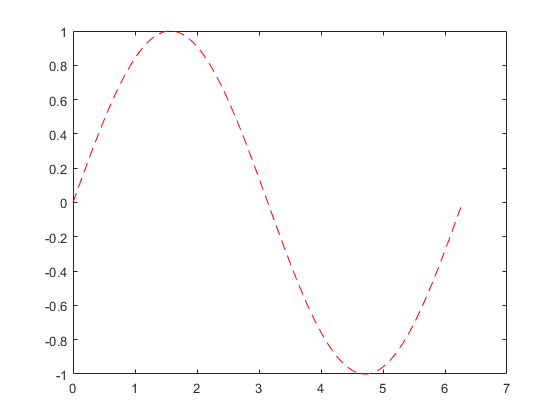
ylabel('sin(x)')

title('Plot of the Sine Function')



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

plot(x,y,'r--') ‘r\*’ ‘r’ ‘r+’ ‘ro’ ‘r:’ r-red/ g-green/ b-blue/ y-yellow



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

Notice that the titles and labels that you defined for the first plot are no longer in the current figure window. By default, MATLAB® clears the figure each time you call a plotting function, resetting the axes and other elements to prepare the new plot.

To add plots to an existing figure, use hold.

x = 0:pi/100:2\*pi;

y = sin(x);

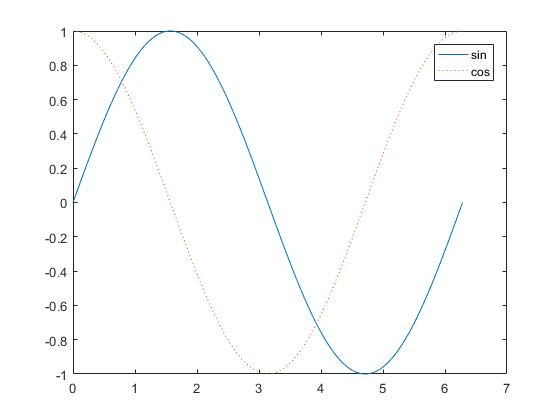
plot(x,y)

hold on

y2 = cos(x);

plot(x,y2,':')

legend('sin','cos')



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

### 3-D Plots

[Open Script](matlab:openExample('matlab/ThreeDPlotsGSExample'))

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

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

x=-2:.2:2

y=-2:.2:2

[x,y]=meshgrid(x,y)

[X,Y] = meshgrid(-2:.2:2); //x,y一样

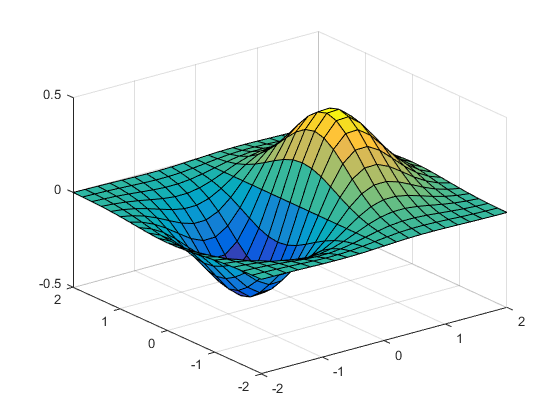
Z = X .\* exp(-X.^2 - Y.^2);

Then, create a surface plot.

surf(X,Y,Z)

% 三维曲线  
plot3(X, Y, Z)

% 三维曲面  
figure  
surf(X, Y, Z)



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

### Subplots

[Open Script](matlab:openExample('matlab/SubplotsGSExample'))

You can display multiple plots in different subregions of the same window using the subplot function.

The first two inputs to subplot indicate the number of plots in each row and column. The third input specifies which plot is active. For example, create four plots in a 2-by-2 grid within a figure window.

t = 0:pi/10:2\*pi;

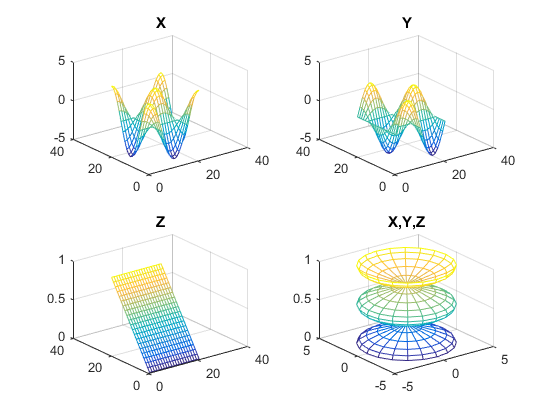
[X,Y,Z] = cylinder(4\*cos(t));

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');



## Next in Getting Started with MATLAB

[Programming and Scripts](https://www.mathworks.com/help/matlab/learn_matlab/scripts.html)

## 8. Programming and Scripts

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

### Sample Script

To create a script, use the edit command,

edit plotrand

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

n = 50;

r = rand(n,1);

plot(r)

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

m = mean(r);

hold on

plot([0,n],[m,m])

hold off

title('Mean of Random Uniform Data')

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

% Generate random data from a uniform distribution

% and calculate the mean. Plot the data and the mean.

n = 50; % 50 data points

r = rand(n,1);

plot(r)

% Draw a line from (0,m) to (n,m)

m = mean(r);

hold on

plot([0,n],[m,m])

hold off

title('Mean of Random Uniform Data')

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

plotrand

You can also run scripts from the Editor by pressing the **Run** button, https://www.mathworks.com/help/matlab/learn_matlab/run_ts_16.png.

### Loops and Conditional Statements

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

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

nsamples = 5;

npoints = 50;

for k = 1:nsamples

currentData = rand(npoints,1);

sampleMean(k) = mean(currentData);

end

overallMean = mean(sampleMean)

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

for k = 1:nsamples

iterationString = ['Iteration #',int2str(k)];

disp(iterationString)

currentData = rand(npoints,1);

sampleMean(k) = mean(currentData)

end

overallMean = mean(sampleMean)

When you run the script, it displays the intermediate results, and then calculates the overall mean.

calcmean

Iteration #1

sampleMean =

0.3988

Iteration #2

sampleMean =

0.3988 0.4950

Iteration #3

sampleMean =

0.3988 0.4950 0.5365

Iteration #4

sampleMean =

0.3988 0.4950 0.5365 0.4870

Iteration #5

sampleMean =

0.3988 0.4950 0.5365 0.4870 0.5501

overallMean =

0.4935

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

if overallMean < .49

disp('Mean is less than expected')

elseif overallMean > .51

disp('Mean is greater than expected')

else

disp('Mean is within the expected range')

end

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

overallMean =

0.5178

Mean is greater than expected

### Script Locations

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

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

## Next in Getting Started with MATLAB

[Help and Documentation](https://www.mathworks.com/help/matlab/learn_matlab/help.html)

## 9. Help and Documentation

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

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

doc mean

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

mean(

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

help mean

Access the complete product documentation by clicking the help icon https://www.mathworks.com/help/matlab/learn_matlab/helpicon.png.