MATLAB
Built-in functions and plotting

Edited by Péter Vass

Built-in fuctions

A function is a suitably ordered and structured group of statements (based on an algorithm) which performs a specific task.
A built-in function is a function which is offered by the MATLAB environment.
Actually, it is ready for the users to utilize it as a computational aid.
In order to call a built-in function, we must know the rules of its call.

The general form of calling a function:

[out1,out2, ..., outN] = funcname(inp1,inp2,inp3, ..., inpM)

It contains:

- the names of the output variables between square brackets,
- equals sign,
- the name of the function,
- and the names of the input variables between round brackets.

Built-in fuctions

Output variables

The output variables receive and store the results of the computation.
Their order is fixed.
Most of the built-in functions can be called with different number of output variables.
If we call the function with a single output variable, the use of square brackets are not necessary.

Input variables

The input variables provide the required data for the computation.
Their order is also fixed.
Most of the built-in functions can be called with different number of input variables.
The round brackets may only be left out when no input variable is necessary for the given function call.

Built-in fuctions

An example for the simplest form of a function call without any output and input variables:

```
>>clc
```

The detailed information about the calling rules of a given function can be found in the MATLAB help system.

```
>>doc
```

Help window appears.

At first, select MATLAB item then MATLAB functions item.

If you scroll down the track bar, you fill find the list of all built-in functions grouped in different categories.

After the name of a function, a short information is displayed about its role to help the user in finding the right function.

For the detailed information, click the name of the function.

Built-in fuctions

The most frequently used mathematical functions can be found in the following categories:

- arithmetic,
- trigonometry (e.g. sin, cos, tan, asin, acos, atan, their arguments should be specified in radians),
- exponents and logarithms (e.g. exp, log, log2, log10).

Some useful functions for performing data analysis:

max	maximum
min	minimum
find	find indices of nonzero elements
mean	average or mean
median	median
std	standard deviation
sort	sort in ascending order
sortrows	sort rows in ascending order
sum	sum of elements

Built-in fuctions

prod	product of elements
diff	difference between elements
trapz	trapezoidal integration
cumsum	cumulative sum
cumprod	cumulative product
cumtrapz	cumulative trapezoidal integration

Built-in fuctions

prod	product of elements
diff	difference between elements
trapz	trapezoidal integration
cumsum	cumulative sum
cumprod	cumulative product
cumtrapz	cumulative trapezoidal integration

Plotting 2D graphs

The steps of displaying a function of one variable in the form of a 2D graph:

specifying the range of values for the independent variable,

defining the values of the dependent variable,

calling the plot command with the two input variables.

Example:

```
>> x = 0:pi/20:2*pi;
>> y = sin(x);
>> plot(x,y)
```

There are many options for changing the appearance of a plot.

For instance, it is possible to specify colour, line styles, and markers.

The general format of specifying these properties in a plot command:

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

where *colour_style_marker* is a string containing from one to four characters (enclosed in single quotation marks).

Plotting 2D graphs

The 'colour style marker' string may contain the characters below:

colour		line style	
'c'	cyan	'-'	solid (default)
'm'	magenta	'--'	dashed
'y'	yellow	':'	dotted
'r'	red	'.-'	dash-dot
'g'	green	'.'	a dot for each value
'b'	blue (default)		
'w'	white		
'k'	black		

Plotting 2D graphs

Marker type

'+'	plus mark
'o'	unfilled circle
'*'	asterisk
'x'	x mark
's'	filled square
'd'	filled diamond
'^'	filled upward triangle
'v'	filled downward triangle
'>'	filled rightward triangle
'<'	filled leftward triangle
'p'	filled pentagram
'h'	filled hexagram

Colours, line styles and markers may be combined arbitrarily.

Plotting 2D graphs

Examples

```
>>plot(x, y, 'r-.')
>>plot(x, y, 'bx')
>>plot(x, y, 'b:x')
```

We can add title, labels, grid lines, and scaling to the graph:

- xlabel and ylabel commands generate labels along x-axis and y-axis,
- title command puts a title on the top of the graph,
- grid on command displays the grid lines on the graph (grid off will switch off the display of the grid lines),
- axis equal command generates the plot with axes of the same scale factor,
- axis square command produces a plot with axes of equal length,

Plotting 2D graphs

Example

```
>> xlabel('x-axis')
>> ylabel('y-axis')
>> title('The graph of sin(x)')
>> grid on
>> axis equal
```

Displaying more curves

We can display more functions or curves in the same graph by means of the plot command.

General format

plot(xdata1,ydata1, 'colour_style_marker1',xdata2,ydata2, 'colour_style_marker2', …)

Plotting 2D graphs

Example

```
>> x=-3:0.1:3;
>> y1=x.^2;
>> y2=x.^3;
>> plot(x, y1, 'k', x, y2, 'r')
>>grid on
```

If we want to add a new curve to an existing plot, we have to apply the hold on command before plotting the new curve for keeping the previously created content of the plot.
Without using the hold on command, the original content of the plot will be erased.

Plotting 2D graphs

Example

```
>> y3=x.^4;
>> hold on
>> plot(x, y3, 'g')
>> grid on
```

To remove the last plotted curve, type the undo command.

To switch off the effect of the hold on command, type hold off.

We can also create a legend for the graph by means of the legend command (see doc legend for the details)

Example

```
>>legend('x^2', 'x^3', 'x^4', 'Location', 'north')
```

The clf (clear figure) command can be used for erasing the plot, and the close command closes the plot window (called figure in MATLAB).

Plotting 2D graphs

Displaying more graphs in the same plot window (figure)
By means of the subplot command, the whole area of the plot window can be divided into partial areas of graphs arranged in columns and rows.
The serial number of a partial graph is always counted from left to right in a row then the numbering continues in the next row.
For instance, the 3rd graph in a set of graphs arranged in two rows and two columns is the first graph of the second row.

Example:

```
>>t = 0:0.1:2*pi;
>>subplot(2,2,1)
>>plot(cos(t),sin(t))
>>subplot(2,2,2)
>>plot(cos(t),sin(2*t))
>>subplot(2,2,3)
>>plot(cos(t),sin(3*t))
```

Plotting 2D graphs

```
>>subplot(2,2,4)
>>plot(cos(t),sin(4*t))
```

By means of the cla command, any graph previously selected by the subplot command can be deleted form a set of graphs.

Example:

```
>>subplot(2,2,2)
>>cla
```

Displaying text on a graph

text(x, y, 'text')   places text at position x, y

Example:

```
>> t=0:0.1:2*pi;
>> plot(t,sin(t));
>> text(2.7, 0.6, 'sin x');
```

Plotting 2D graphs

Other types of graphs

bar chart

```
>>planets=[1,2,3,4,5,6,7,8];
>> radius=[0.38,0.95,1,0.53,11.19,9.4,3.85,3.67];
>> bar(planets, radius)
>> xlabel('Planets: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune');
>> ylabel('Radius related to the Earth');
>> grid
```

histogram

```
>> x = randn(10000,1);
>> hist(x, 100)
```

See plot, bar, hist, pie, area, rose, pareto in the help.

Plotting 2D graphs

Contour lines

We can visualize a mathematical function of two variables by means of contour lines.

At first, we have to create a two-dimensional data set of the independent variables by calling the meshgrid command.

Example:

```
>> [x,y] = meshgrid(-5:0.1:5, -3:0.1:3);
>> z=x.^2/4 - y.^2/4;
>> contour(x, y, z)
>> contourf(x,y,z)
```

The color scale may be modified by the colormap command.

Plotting 3D graphs

Displaying curves in 3D

The plot command has a 3D version called plot3.

Example:

```
>> t = 0:0.1:2*pi;
>> plot3(cos(3*t),sin(3*t),t);
```

Displaying surfaces in 3D

Example:

```
>> [x,y] = meshgrid(-5:0.1:5, -3:0.1:3);
>> z=x.^2/4 - y.^2/4;
>>mesh(x,y,z);
>> surf(x,y,z)
>> surfc(x,y,z)
```