الكلية	العلوم
القسم	الرياضيات
المادة باللغة الانجليزية	Programming (MATLAB)
المادة باللغة العربية	البرمجة بلغة (ماتلاب)
المرحلة الدراسية	الثانية
اسم التدريسي	صفوت عبدالقادر حمد
عنوان المحاضرة باللغة الانجليزية	Plot Function
عنوان المحاضرة باللغة العربية	دوال الرسم
رقم المحاضرة	العاشرة
المصادر والمراجع	MATLAB A Practical Introduction to Programming and Problem Solving
	MATLAB The Language of Technical Computing
	MATLAB numerical computing

University Of Anbar

College Of Science

Math Department

Programing Language (MATLAB)

Lecture 10

Plot Function

By:

Safwat A. Hamad

For the 2nd stage Math Department

1. The Plot Function

For now, we'll start with a very simple graph of one point using the plot function. The following script, ***plotonepoint***, plots one point. To do this, first values are given for the x and y coordinates of the point in separate variables. The point is plotted using a red star ('*'). The plot is then customized by specifying the minimum and maximum values on first the x and then y-axes. Labels are then put on the x-axis, the y-axis, and the graph itself using the functions ***xlabel***, ***ylabel***, and ***title***. (Note: there are no default labels for the axes.)

All of this can be done from the Command Window, but it is much easier to use a script. The following shows the contents of the script ***plotonepoint*** that accomplishes this. The ***x*** coordinate represents the time of day (e.g., 11 a.m.) and the ***y*** coordinate represents the temperature (e.g., in degrees Fahrenheit) at that time.

plotonepoint.m

```
% This is a really simple plot of just one point!
% Create coordinate variables and plot a red '*'
x = 11;
y = 48;
plot(x,y,'r*')
% Change the axes and label them
axis([9 12 35 55])
xlabel('Time')
ylabel('Temperature')
% Put a title on the plot
title('Time and Temp')
```

In the call to the axis function, one vector is passed. The first two values are the minimum and maximum for the x-axis, and the last two are the minimum and maximum for the y-axis. Executing this script brings up a Figure Window with the plot (see Figure 1).

To be more general, the script could prompt the user for the time and temperature, rather than just assigning values. Then, the axis function could be used based on whatever the values of x and y are, as in the following example:

```
axis([x-2 x+2 y-10 y+10])
```

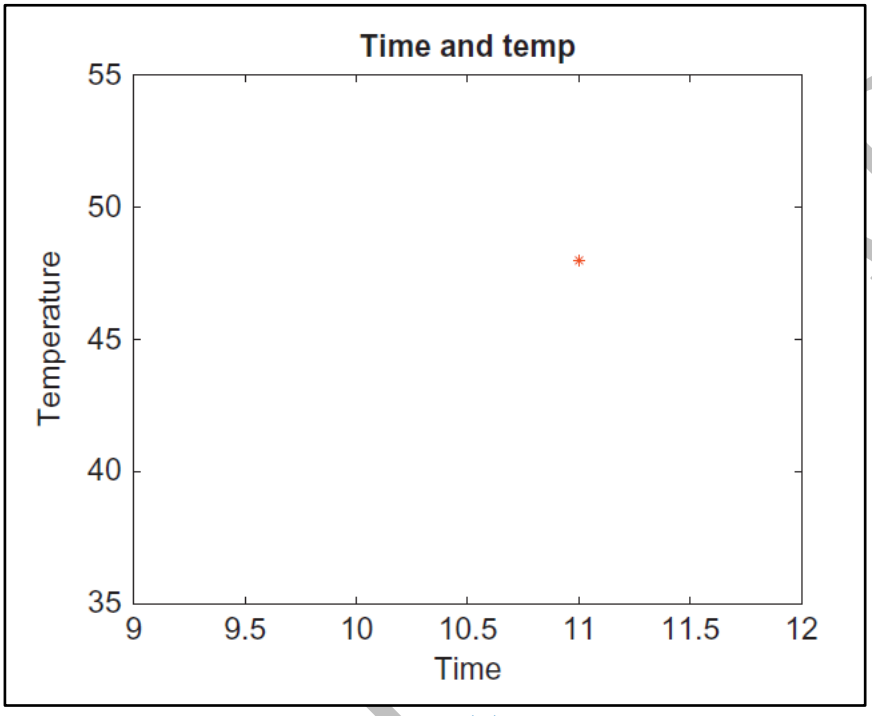

Figure (1)

In addition, although they are the x and y coordinates of a point, variables named time and temp might be more mnemonic than x and y.

To plot more than one point, x and y vectors are created to store the values of the (x,y) points. For example, to plot the points

```
(1,1)
(2,5)
(3,3)
(4,9)
(5,11)
(6,8)
```

first an x vector is created that has the x values (as they range from1 to 6 in steps of 1, the colon operator can be used) and then a y vector is created with the y values.

The following will create (in the Command Window) x and y vectors and then plot them (see Figure 2).

```
>> x = 1:6;
>> y = [1 5 3 9 11 8];
>> plot(x,y)
```

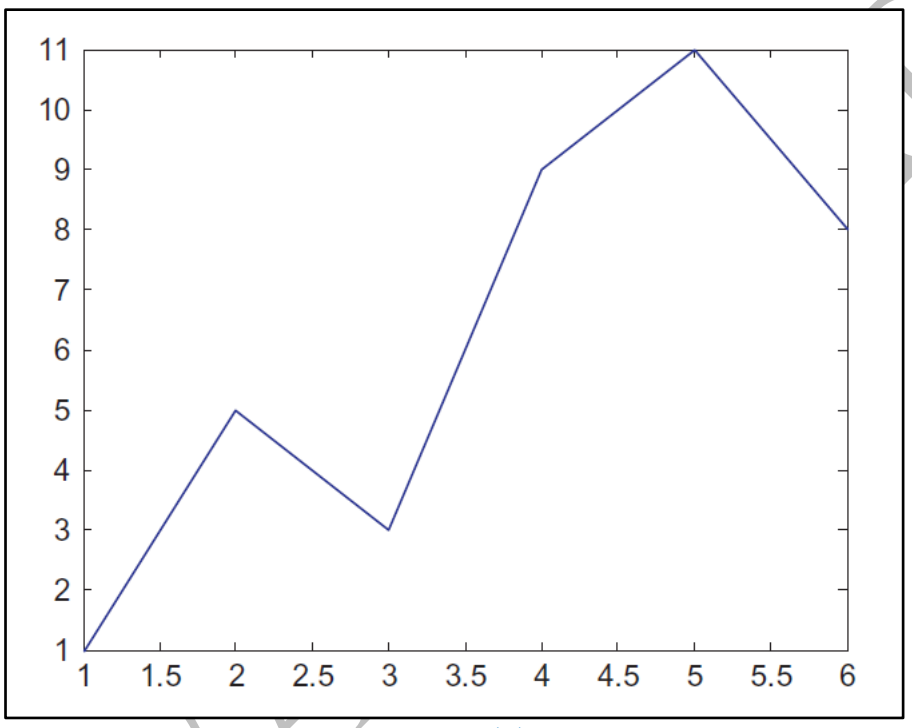

Figure (2)

Note that the points are plotted with straight lines drawn in between. Note also that the default color is a shade of blue. Also, the axes are set up according to the data; for example, the x values range from 1 to 6 and the y values from 1 to 11, so that is how the axes are set up. There are many options for the axis function; for example, just calling it with no arguments returns the values used for the x and y-axes ranges.

```
>> arang = axis
arang =
      1  6  1  11
```

Axes can also be turned on and off, and they can be made square or equal to each other. A subset of the data can also be shown by limiting the extent of the axes. Also,

note that in this case the x values are the indices of the y vector (the y vector has six values in it, so the indices iterate from 1 to 6).When this is the case, it is not necessary to create the x vector. For example,

```
>> plot(y)
```

***** will plot exactly the same figure without using an x vector.

2. Customizing a Plot: Color, Line Types, Marker Types

Plots can be done in the Command Window, as shown here, if they are really simple. However, many times it is desired to customize the plot with labels, titles, and so on, so it makes more sense to do this in a script. Using the help function for plot will show the many options such as the line types and colors. In the previous script ***plotonepoint***, the character vector 'r*' specified a red star for the point type.

The ***LineSpec***, or line specification, can specify up to three different properties in a character vector or string, including the color, line type, and the symbol or marker used for the data points.

The possible colors are:

```
b blue
g green
r red
c cyan
m magenta
y yellow
k black
w white
```

Either the single character listed above or the full name of the color can be used in the string to specify the color. The ***plot symbols***, or ***markers***, that can be used are:

```
. point
o circle
```

```
x x-mark
+ plus
* star
s square
d diamond
v down triangle
^ up triangle
< left triangle
> right triangle
p pentagram
h hexagram
```

Line types can also be specified by the following:

```
- solid
: dotted
-. dash dot
– dashed
(none) no line
```

If no line type is specified and no marker type is specified, a solid line is drawn between the points, as seen in the last example.

3. Simple Related Plot Functions

Other functions that are useful in customizing plots include clf, figure, hold, legend, and grid. Brief descriptions of these functions are given here; use help to find out more about them:

- clf: clears the Figure Window by removing everything from it.
- figure: creates a new, empty Figure Window when called without any arguments. Calling it as figure(n)where n is an integer is a way of creating and maintaining multiple Figure Windows and of referring to each individually.

- hold: is a toggle that freezes the current graph in the Figure Window, so that new plots will be superimposed on the current one.
- Just hold by itself is a toggle, so calling this function once turns the hold on, and then the next time turns it off. Alternatively, the commands hold on and hold off can be used.
- legend: displays strings or character vectors passed to it in a legend box in the Figure Window, in order of the plots in the Figure Window.
- grid: displays grid lines on a graph. Called by itself, it is a toggle that turns the grid lines on and off. Alternatively, the commands grid on and grid off can be used.

For example, the following script creates two separate Figure Windows. First, it clears the Figure Window. Then, it creates an x vector and two different y vectors (y1 and y2). In the first Figure Window, it plots the y1 values using a bar chart. In the second Figure Window, it plots the y1 values as black lines, puts hold on so that the next graph will be superimposed, and plots the y2 values as black circles. It also puts a legend on this graph and uses a grid. Labels and titles are mitted in this case, as it is generic data.

plot2figs.m

```
% This creates 2 different plots, in 2 different
% Figure Windows, to demonstrate some plot features
clf
x = 1:5; % Not necessary
y1 = [2 11 6 9 3];
y2 = [4 5 8 6 2];
% Put a bar chart in Figure 1
figure(1)
bar(x,y1)
% Put plots using different y values on one plot
```

```
% with a legend
figure(2)
plot(x,y1,'k')
hold on
plot(x,y2,'ko')
grid on
legend('y1','y2')
```

Running this script will produce two separate Figure Windows. If there are no other active Figure Windows, the first, which is the bar chart, will be in the one titled "Figure A" in MATLAB. The second will be in "Figure B".

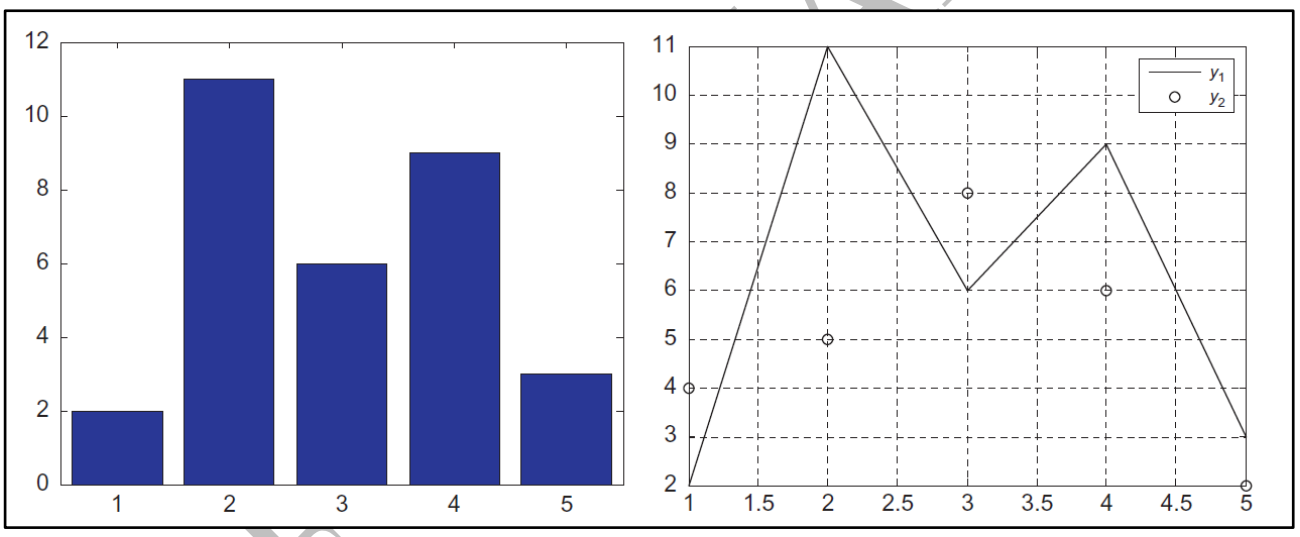

Figure (A) Figure (B)

Note that the first and last points are on the axes, which makes them difficult to see. That is why the axis function is used frequently, as it creates space around the points so that they are all visible.

The ability to pass a vector to a function and have the function evaluate every element of the vector can be very useful in creating plots. For example, the following script graphically displays the difference between the (sin) and (cos) functions:

Sin&ncos.m

```
% This script plots sin(x) and cos(x) in the same Figure Window
% for values of x ranging from 0 to 2*pi
clf
x = 0: 2*pi/40: 2*pi;
y = sin(x);
plot(x,y,'ro')
hold on
y = cos(x);
plot(x,y,'b+')
legend('sin', 'cos')
xlabel('x')
ylabel('sin(x) or cos(x)')
title('sin and cos on one graph')
```

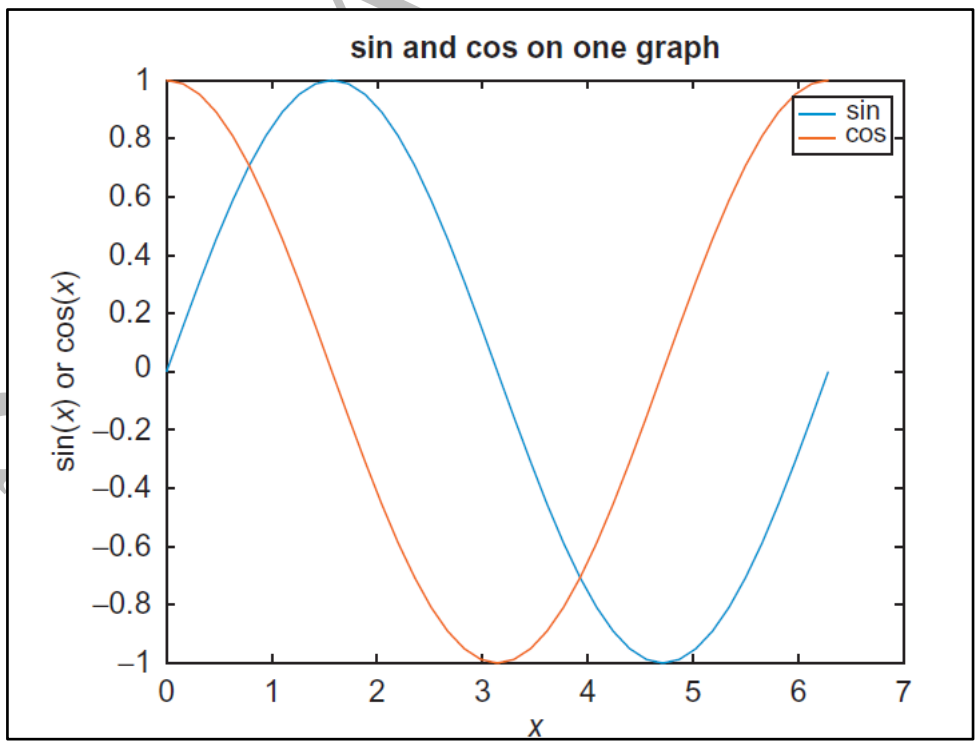

Figure (3)

The script creates an x vector; iterating through all of the values from 0 to 2*π in steps of 2*π/40 gives enough points to get a good graph. It then finds the sine of each x value and plots these. The command hold on freezes this in the Figure Window, so the next plot will be superimposed. Next, it finds the cosine of each x value and plots these points. The legend function creates a legend; the first character vector is paired with the first plot, and the second character vector with the second plot. Running this script produces the plot seen in Figure 3.5.

Beginning with Version R2014b, when hold on is used, MATLAB uses a sequence of colors for the plots, rather than using the default color for each. Of course, colors can also be specified as was done in this script. Note that instead of using hold on, both functions could have been plotted using one call to the plot function:

```
plot(x,sin(x),x,cos(x))
```
