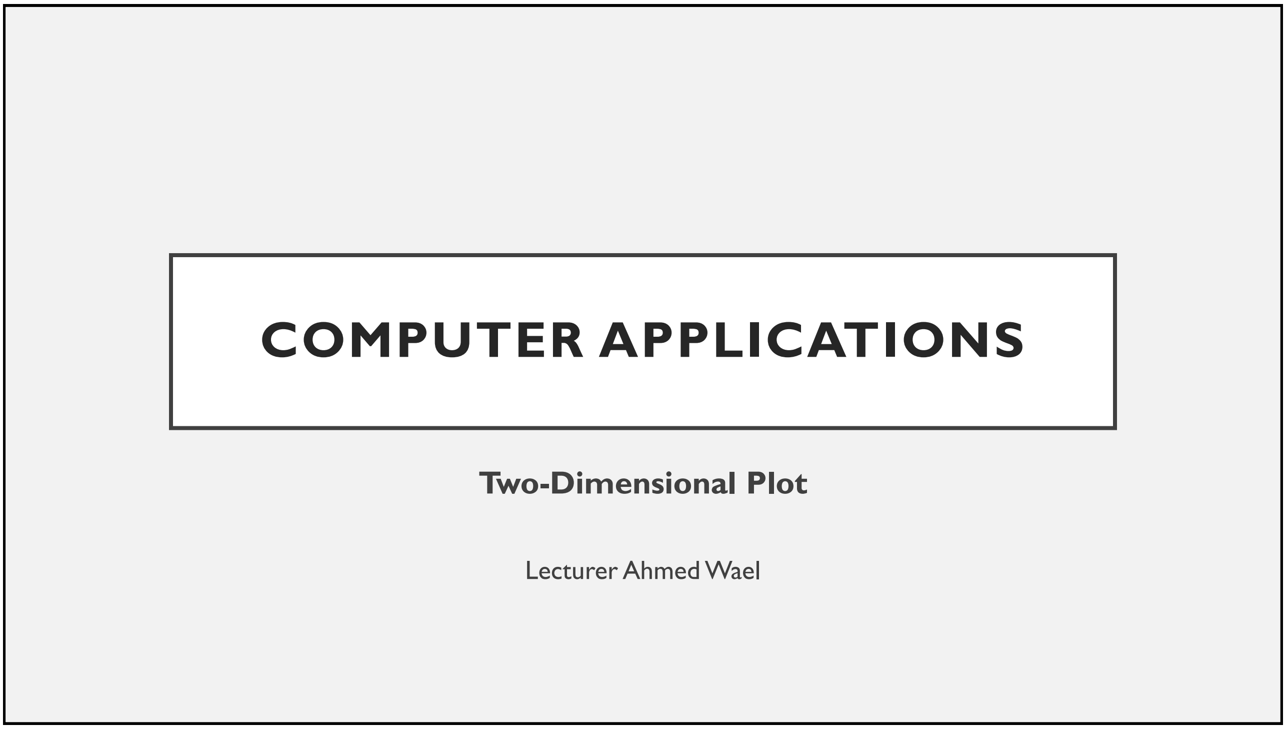
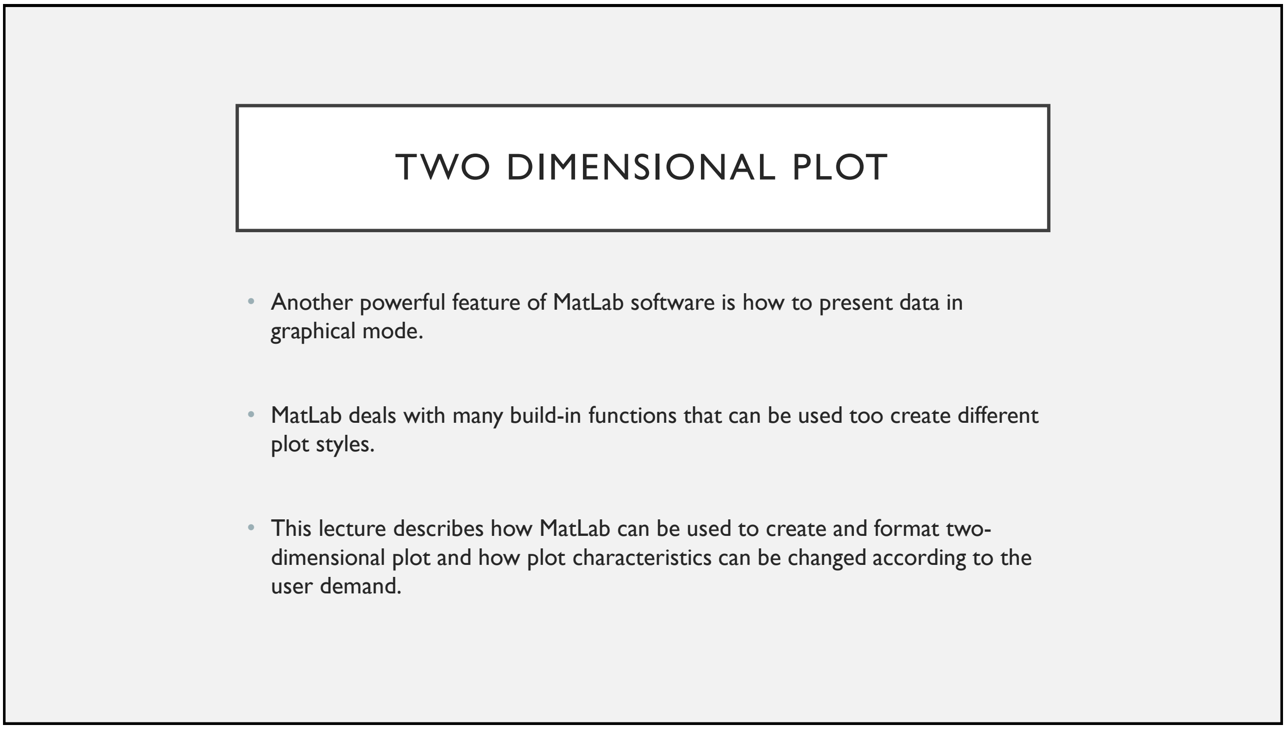


COMPUTER APPLICATIONS

Two-Dimensional Plot

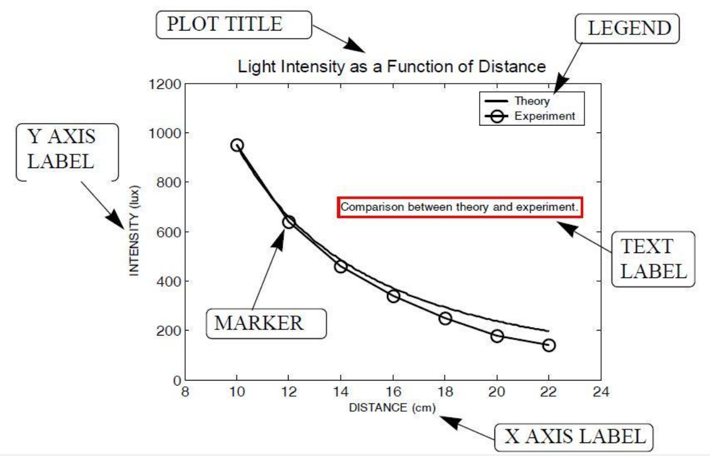
Lecturer Ahmed Wael



TWO DIMENSIONAL PLOT

- Another powerful feature of MatLab software is how to present data in graphical mode.
- MatLab deals with many build-in functions that can be used too create different plot styles.
- This lecture describes how MatLab can be used to create and format two-dimensional plot and how plot characteristics can be changed according to the user demand.

TWO-DIMENSIONAL PLOT



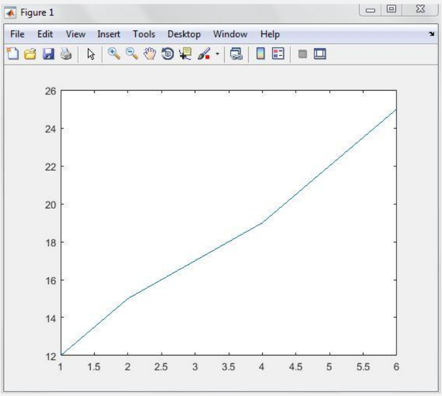
THE *plot* COMMAND

- To create two dimensional plot:

$$plot(x, y)$$

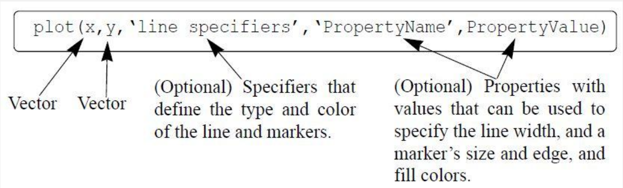
- x, y are one-dimensional vectors. The x represents the horizontal axis and y is the vertical axis.
- ***Example***

```
>> x = [1 2 3 4 5 6];
>> y = [ 12 15 17 19 22 25];
>> plot(x,y)
```



plot SPECIFICATIONS

- The *plot* command has additional options that used to specify the line properties.
- Color, line style marker, axis labels, title are options those can be configured for plotting figures.
- To make *plot* command works with such changes, we use:

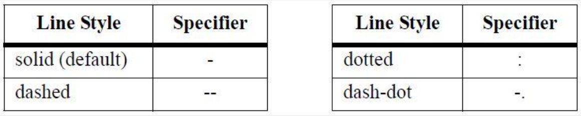


```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
```

Vector
Vector
(Optional) Specifiers that define the type and color of the line and markers.
(Optional) Properties with values that can be used to specify the line width, and a marker's size and edge, and fill colors.

LINE SPECIFIER

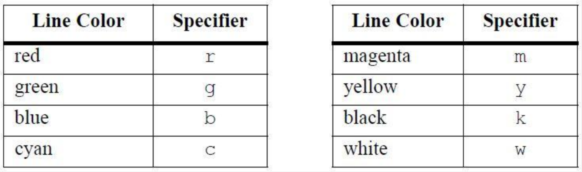
- To change the line type:



Line Style	Specifier
solid (default)	-
dashed	--

Line Style	Specifier
dotted	:
dash-dot	-.

- To change the line color:

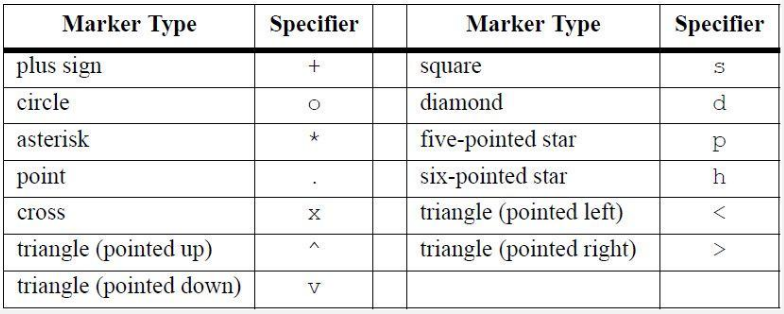


Line Color	Specifier
red	r
green	g
blue	b
cyan	c

Line Color	Specifier
magenta	m
yellow	y
black	k
white	w

LINE SPECIFIER

- To introduce or change the line marker:



Marker Type	Specifier		Marker Type	Specifier
plus sign	+		square	s
circle	o		diamond	d
asterisk	*		five-pointed star	p
point	.		six-pointed star	h
cross	x		triangle (pointed left)	<
triangle (pointed up)	^		triangle (pointed right)	>
triangle (pointed down)	v			

NOTES TO BE CONSIDERED

- To import specifiers inside the *plot* command, they have to be introduced as a string command.
- The specifiers can be defined in any order
- The number of specifier is optional, you can make one, two or more depending on what you need to define.
- ***Example:***

  >> *plot (x, y)* *plot without specifier*

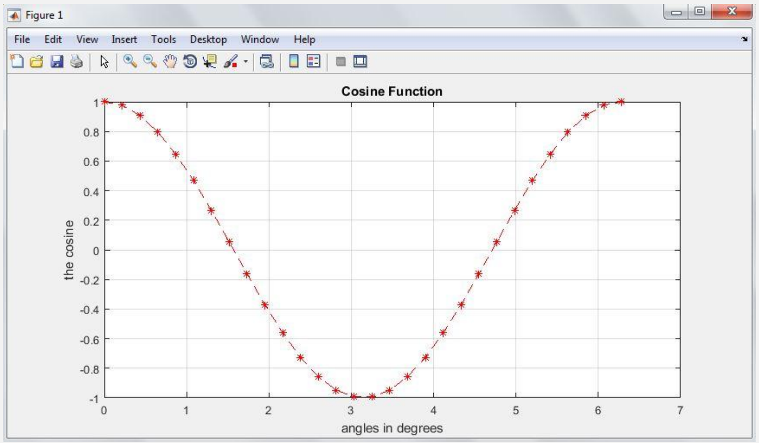
  >> *plot (x, y,’r’)* *plot with red color specifier*

  >> *plot (x, y,‘- - r’, ‘^’)* *plot with red, dashed-line, and marker specifier*

EXAMPLE #1

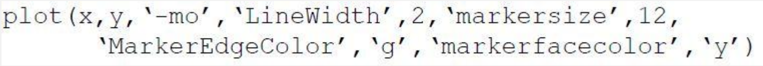
```
>> x = linspace (0, 2*pi, 30);        % a vector of 30 angles between 0 and 2pi
>> y = cos (x);                       % a vector of cosine function for each angle defined by x
>> plot ( x, y, ‘r--*‘ )          % two-dimensional plot with red, dashed-line type and * marker specifier
>> grid on;                           % opens grid on the graphed background
>> xlabel (”angles in degrees”)       % gives a title for the x-axis
>> ylabel (“the cosine”)              % gives and title for the y-axis
>> title (”Cosine Function”)          % gives a title for the whole plot.
```

EXAMPLE #1



PROPERTY NAME AND VALUE

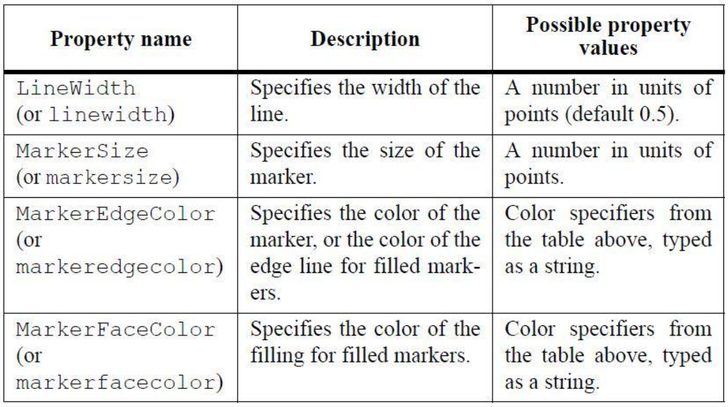
- To specify the thickness of the line, the size of the marker, and the color of the marker's edge and fill we do the follow:



```
plot(x,y,'-mo','LineWidth',2,'markersize',12,
         'MarkerEdgeColor','g','markerfacecolor','y')
```

- All properties of the name and value have to be inside the *plot* command.

PROPERTY NAME AND VALUE

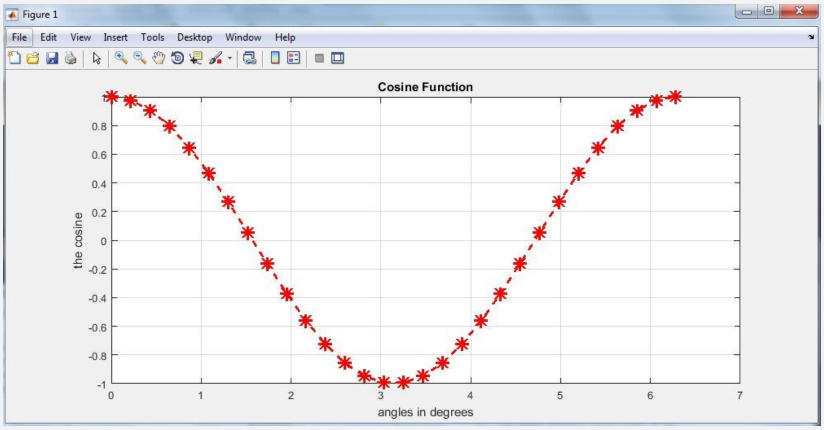


Property name	Description	Possible property values
`LineWidth` (or `linewidth`)	Specifies the width of the line.	A number in units of points (default 0.5).
`MarkerSize` (or `markersize`)	Specifies the size of the marker.	A number in units of points.
`MarkerEdgeColor` (or `markeredgecolor`)	Specifies the color of the marker, or the color of the edge line for filled markers.	Color specifiers from the table above, typed as a string.
`MarkerFaceColor` (or `markerfacecolor`)	Specifies the color of the filling for filled markers.	Color specifiers from the table above, typed as a string.

EXAMPL #2

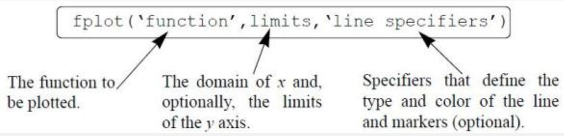
```
>> x = linspace (0, 2*pi, 30);                              % a vector of 30 angles between 0 and 2pi
>> y = cos (x);                          % a vector of cosine function for each angle defined by x
>> plot ( x, y, ‘r--*‘, ‘linewidth’, 2, ‘markersize’, 12 )  % two-dimensional plot with red, dashed-line type
                                                            and * marker specifier, and the line width = 2 and
                                                            the marker size = 12
>> grid on;                                                 % opens grid on the graphed background
>> xlabel (”angles in degrees”)                             % gives a title for the x-axis
>> ylabel (”the cosine”)                                    % gives and title for the y-axis
>> title (”Cosine Function”)                                % gives a title for the whole plot.
```

EXAMPLE #2



THE *fplot* COMMAND

- Another plot buid-in function provides more functionality in MatLab software which plots a function with the form $y = f(x)$.
- The *fplot* command form:



```
fplot('function',limits,'line specifiers')
```

The function to be plotted.

The domain of x and, optionally, the limits of the y axis.

Specifiers that define the type and color of the line and markers (optional).

- *'function'*: it can be applied directly inside *fplot* command as a string
- *Limits*: the limit argument is a vector with two elements that specify the minimum and maximum values of x-axis and y-axis (e.g. [xmin, xmax, ymin, ymax].
- *Line specifiers*: similar to *plot* command

EXAMPLE #3

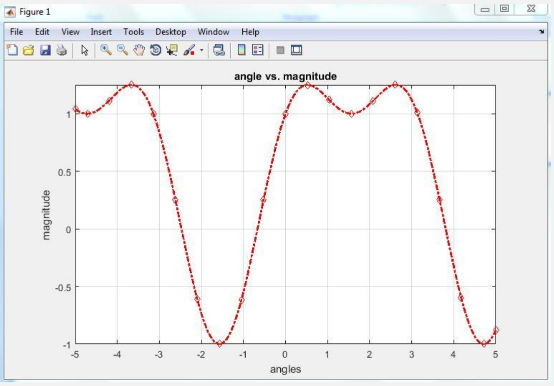
```
>> fplot('cos(x).^2+sin(x)', [-5 5], 'r-.d', 'linewidth', 2)

>> grid on

>> xlabel("angles")

>> ylabel("magnitude")

>> title("angle vs. magnitude")
```



EXAMPLE #4 M-PLOT

- Consider the following function to be plotted using MatLab:

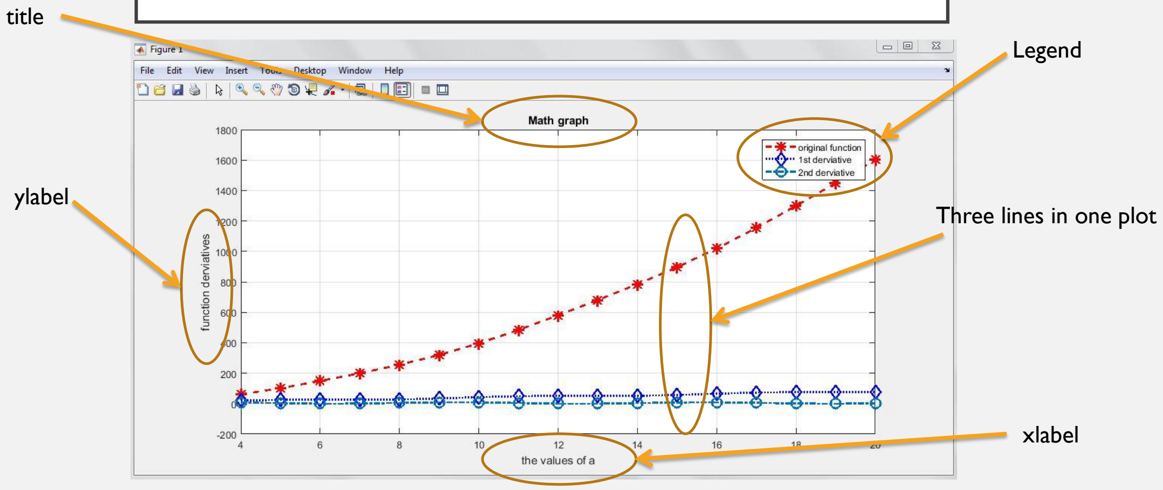
$$x = \ 2a^2 + 5\cos(a)$$

- we would to plot the function $x(a)$ along with its 1st and 2nd derivatives in single plot window.
- Take the value of a between 4 to 20.
- The first derivatives: $x1 = 4a - 5\sin(a)$
- The second derivatives: $x2 = 4 \ - 5\cos(a)$

EXAMPLE #4 M-PLOT

```
>> a = [4:20];                                                            % A vector “a” is defined
>> x = 4.*a.^2 + 5.*cos(a);                                               % Function “x”
>> x_1 = 4.*a - 5.*sin(a);                                                % First derivative
>> x_2 = 4 - 5.*cos(a);                                                   % Second derivative
>> plot(a,x,'r--*',a,x_1,'b:d',a,x_2,'-.o','linewidth',2,'markersize',10)  % Plot function with multiple plot option
>> grid on;                                                               % Grid is activated
>> xlabel('the values of a');                                             % Label for the horizontal axis
>> ylabel('function derivatives');                                        % Label for the vertical axis
>> title("Math graph")                                                    % A title defined
>> legend('original function','1st derviative','2nd derivative')         % The legend shows a sample of the line
                                                                          type of each graph that is plotted, and places a
                                                                          label, specified by the user, beside the line sample
```

EXAMPLE #4 M-PLOT



title

ylabel

Legend

Three lines in one plot

xlabel

hold on, hold off COMMANDS

- Example #4 can be formatted using the *"hold on"* and *"hold off"* commands.

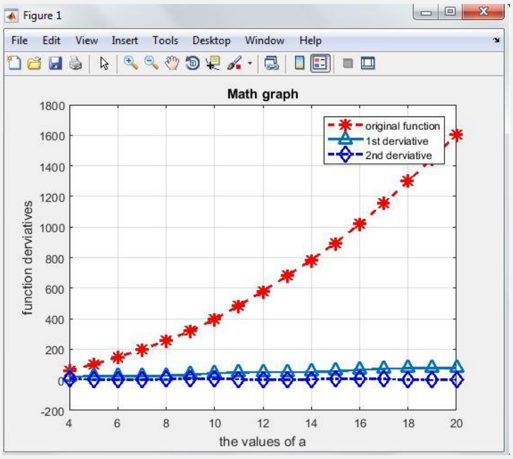
- The *"hold on"* function is to freeze your first plot to add other functions can be added to your main plot window.

- When all functions are inserted, *"hold off"* commands typed to activate you plot window.

- Lets examine these two commands on example #4

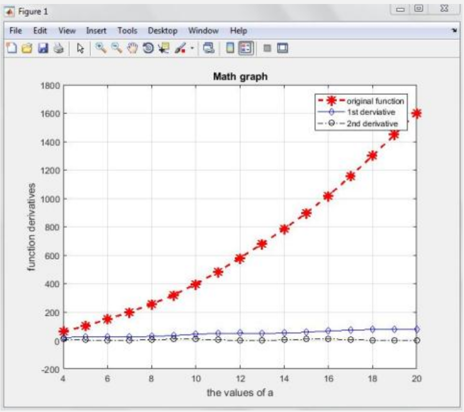
EXAMPLE #5

```
>> a = [4:20];
>> x = 4.*a.^2 + 5.*cos(a);
>> x_1 = 4.*a - 5.*sin(a);
>> x_2 = 4 - 5.*cos(a);
>> plot(a,x,'r--*','linewidth',2,'markersize',10)
>> hold on;
>> plot(a,x_1,'-^','linewidth',2,'markersize',10)
>> plot(a,x_2,'b-.d','linewidth',2,'markersize',10)
>> hold off;
>> grid on;
>> xlabel('the values of a');
>> ylabel('function derivatives');
>> title("Math graph")
>> legend('original function','1st derviative','2nd derivative')
```



Line-command vs multi-polt

```
>> a = [4:20];
>> x = 4.*a.^2 + 5.*cos(a);
>> x_1 = 4.*a - 5.*sin(a);
>> x_2 = 4 - 5.*cos(a);
>> plot(a,x,'r--*','linewidth',2,'markersize',10)
>> line(a,x_1,'linestyle','-','color','b','marker','d')
>> line(a,x_2,'linestyle','-.','color', 'k','marker', 'o')
>> grid on;
>> xlabel('the values of a');
>> ylabel('function derivatives');
>> title("Math graph")
>> legend('original function','1st derviative','2nd derivative')
```



Axis- COMMAND

- The *axis-command* defines the horizontal and vertical axis limits.
- In order to define axis limits, we use the following :

```
axis([xmin    xmax    ymin    ymax])
```

Where:

xmin: *the minimum value of the horizontal axis*

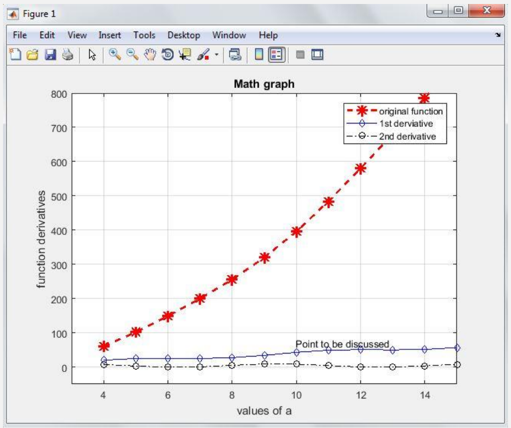
ymin: *the minimum value of the vertical axis*

xmax: *the maximum value of the horizontal axis*

ymax: *the maximum value of the vertical axis*

EXAMLE #6

```
>> a = [4:20];
>> x = 4.*a.^2 + 5.*cos(a);
>> x_1 = 4.*a - 5.*sin(a);
>> x_2 = 4 - 5.*cos(a);
>> plot(a,x,'r--*','linewidth',2,'markersize',10)
>> line(a,x_1,'linestyle','-','color','b','marker','d')
>> line(a,x_2,'linestyle','-.','color', 'k','marker', 'o')
>> grid on;
>> axis([3 15 -50 800]);
>> gtext('Point to be discussed');
>> xlabel('values of a');
>> ylabel('function derivatives');
>> title("Math graph")
>> legend('original function','1st derviative','2nd derivative')
```



LOGARITHMIC SCALE

- Over a wide range of values, it is preferable to plot data in logarithmic scale as it give a mean for those values.
- MatLab provides build-in functions to provide log scale as follow:

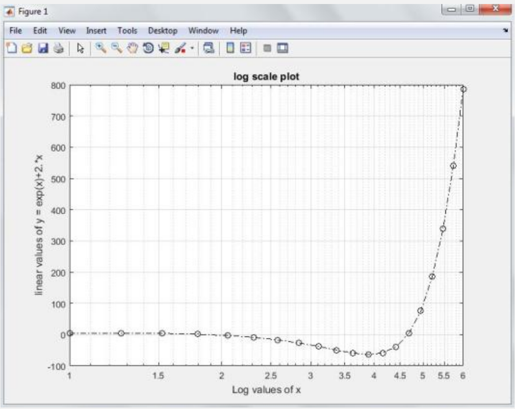
semilogy(x,y): *plots y versus x with a log scale (base 10) of y and linear scale of x*

semilogx(x,y): *plots x versus y with a log scale (base 10) of x and linear scale of y*

Loglog(x,y): *plots x versus y with log scales of x and y.*

EXAMPLE #7

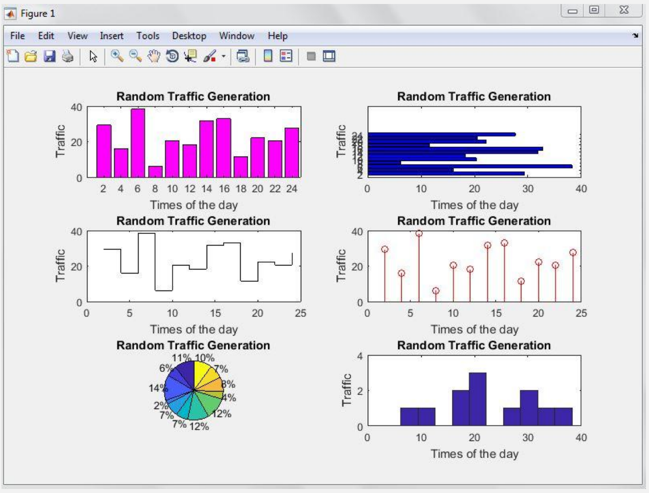
```
>> x = linspace(1,6,20);
>> y = 2.*cos(x).*exp(x)+2.*x;
>> semilogx(x,y,'k-.o');
>> grid on;
>> xlabel('Log values of x');
>> ylabel('linear values of y = exp(x)+2.*x');
>> title('log scale plot')
```



SPECIAL GRAPHS

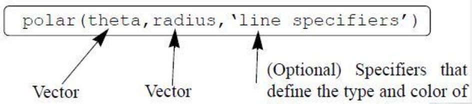
```
>> x = [02 04 06 08 10 12 14 16 18 20 22 24]; y = (40-5)*rand(1,12)+5;
>> subplot(3,2,1); bar(x,y,'m');
>> xlabel('Times of the day');ylabel('Traffic'); title('Random Traffic Generation')
>> subplot(3,2,2); barh(x,y,'b');
>> xlabel('Times of the day'); ylabel('Traffic'); title('Random Traffic Generation')
>> subplot(3,2,3); stairs(x,y,'k');
>> xlabel('Times of the day'); ylabel('Traffic'); title('Random Traffic Generation')
>> subplot(3,2,4); stem(x,y,'r');
>> xlabel('Times of the day'); ylabel('Traffic'); title('Random Traffic Generation')
>> subplot(3,2,5); pie(y);
>> xlabel('Times of the day'); ylabel('Traffic'); title('Random Traffic Generation')
>> subplot(3,2,6); hist(y);
>> xlabel('Times of the day'); ylabel('Traffic'); title('Random Traffic Generation')
```

SPECIAL GRAPHS



POLAR PLOT

- *polar* plot used to plot a point identified by its value and the rotating angle as follow:



```
polar(theta,radius,'line specifiers')
```

(Optional) Specifiers that define the type and color of the line and markers.

- **<u>Example:</u>**

```
>> x = linspace(pi,4*pi,100);
>> y = 3.*sin(0.5*x)+3.*x.^3;
>> polar(x,y)
```

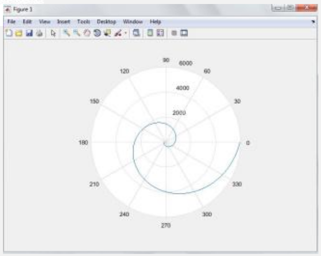


Figure 1
File Edit View Insert Tools Desktop Window Help
90
60
30
0
330
300
270
240
210
180
150
120
6000
4000
2000