AYDIN
ADNAN MENDERES UNIVERSITY
FACULTY OF ENGINEERING
Department of Electrical and Electronics Engineering

EE214 – Computer Aided Engineering Applications

Prof. Dr. Yılmaz KALKAN

EE214 – Computer Aided Engineering Applications

2D GRAPHICS

Objective

- **Present the implementation of a wide selection of two-dimensional plotting capabilities.**

Topics

Introduction: Graphics Management

Basic 2D Plotting Commands

Changing a Graph's Overall Appearance

Graph Annotations and Enhancement

Purpose of a Graphic*

A graph should communicate complex ideas with clarity, precision, and efficiency.

Graphical excellence is attained when the viewer obtains the greatest number of ideas in the shortest time with the least amount of ink.

* E. Tufte, *Visual Display of Quantitative information*, Graphics Press, Cheshire, CT, 1997

The Functions Whose Usage We Shall Illustrate Are -

Management	Generation	
figure	2D	2D
hold	axes	axis, axis equal,
subplot	bar	axis off,
zoom	convhull	axis image
	delauney	box
	fill	clabel
	image	grid
	loglog	legend
	movie	set
	patch	text
	pie	title
	plot	xlabel
	plotyy	xlim
	polar	ylabel
	semilogx	ylim
	semilogy	
	stairs	
	stem	
	voronoi	

Basic 2D Plotting Commands

The basic 2D plotting command is

`plot`**(u, v, c)**

where

***u* and *v* are the *x* and *y* coordinates, respectively, of a point or a series of points.**

These points are either a pair of numbers, vectors of the same length, matrices of the same order, or expressions that, when evaluated, result in one of these three quantities.

The quantity *c* is a string of characters.

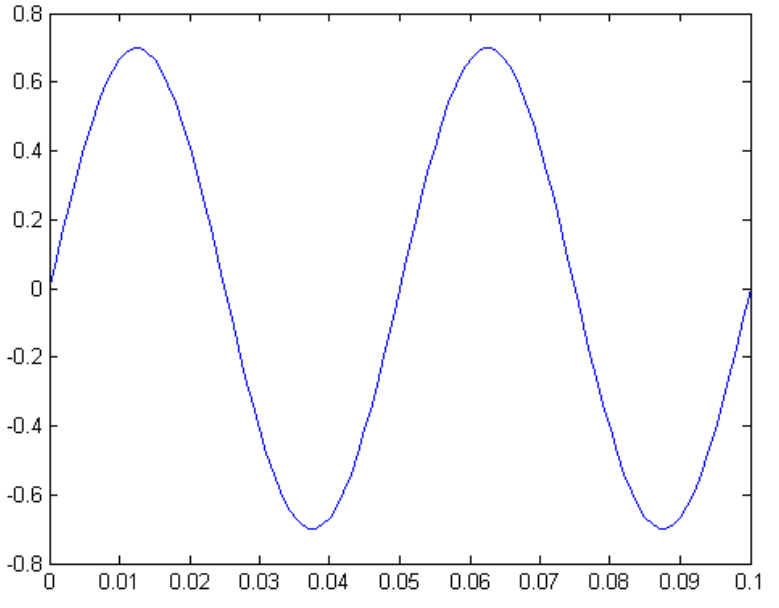
- **One character specifies the line/point color.**
- **One character specifies the point type, if points are to be plotted.**
- **One or two characters are used to specify the line characteristics.**

The order of the three sets of characters within the single quotes is not important.

If *c* is omitted, then the system's default values are used.

Example–

```
t=0:0.001:0.1;
f = 20; %frequency =20Hz
A = 0.7; % amplitude
y = A * sin( 2* pi* f * t); % sinusoidal signal
plot(t,y)
```



Line and Point Characteristics

Line type		Line or point color		Point type	
Symbol	Description	Symbol	Description	Symbol	Description
-	Solid	r	Red	+	Plus sign
--	Dashed	g	Green	o	Circle
:	Dotted	b	Blue	*	Asterisk
-.	Dashed-dot	c	Cyan	.	Point
		m	Magenta	x	Cross
		y	Yellow	s	Square
		k	Black	d	Diamond
		w	White	^	Upward-pointing triangle
				v	Downward-pointing
				>	triangle
				<	Right-pointing triangle
				p	Left-pointing triangle
				h	Pentagram
					Hexagram

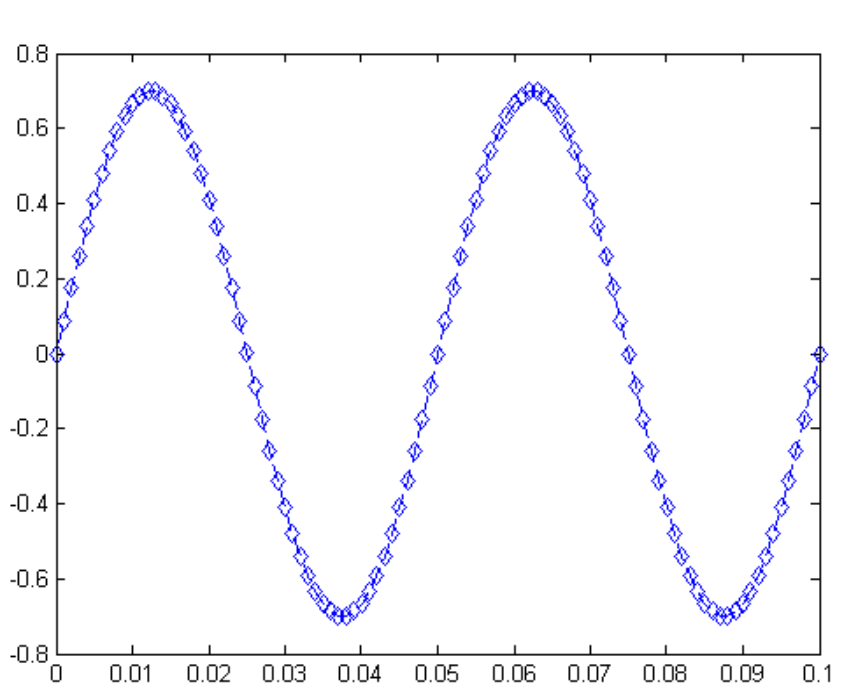
Example –

To plot blue dashed lines connecting blue diamonds,

c = 'b--d'

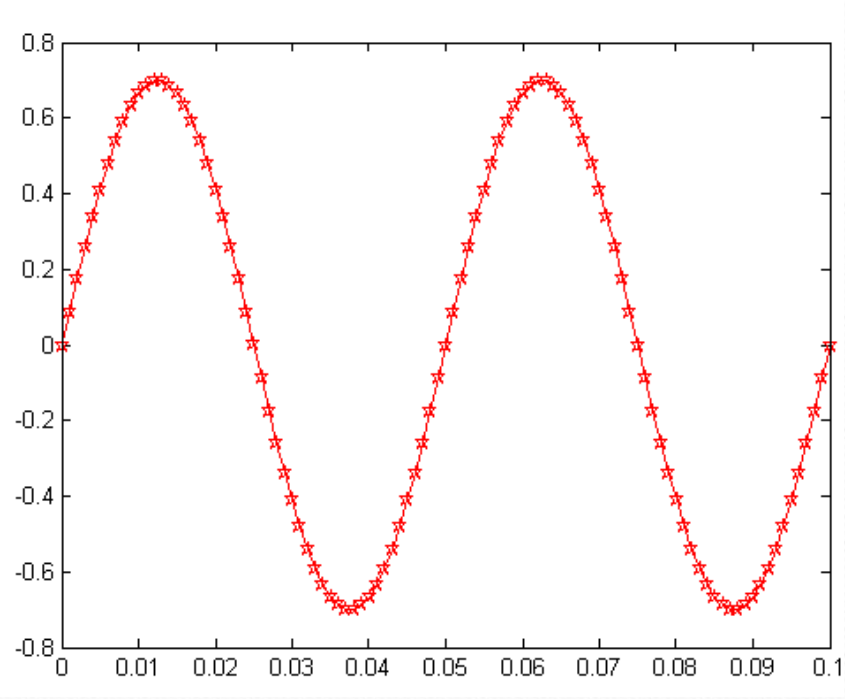
Example –

```
t=0:0.001:0.1;
f= 20; %frequency = 20Hz
A = 0.7; % amplitude
y = A * sin( 2* pi * f * t); % sinusoidal signal
plot(t , y , 'b--d')
```



Examples – Contunied

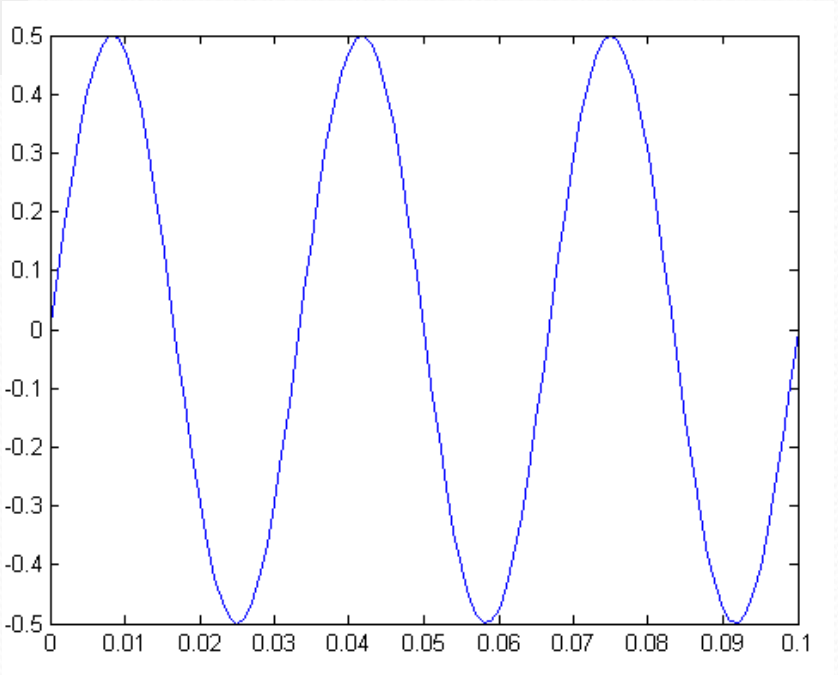
```
t=0:0.001:0.1;
f= 20; %frequency = 20Hz
A = 0.7; % amplitude
y = A * sin( 2* pi * f * t); % sinusoidal signal
plot(t , y , 'r-p')
```



Example –

```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

plot(t,y1)
plot(t,y2)
```



New Figure

- **A graph is created in a figure window.**
- **Any previously created window is removed prior to creating the new figure window.**
- **To retain each new graph in its own figure window, one must use**

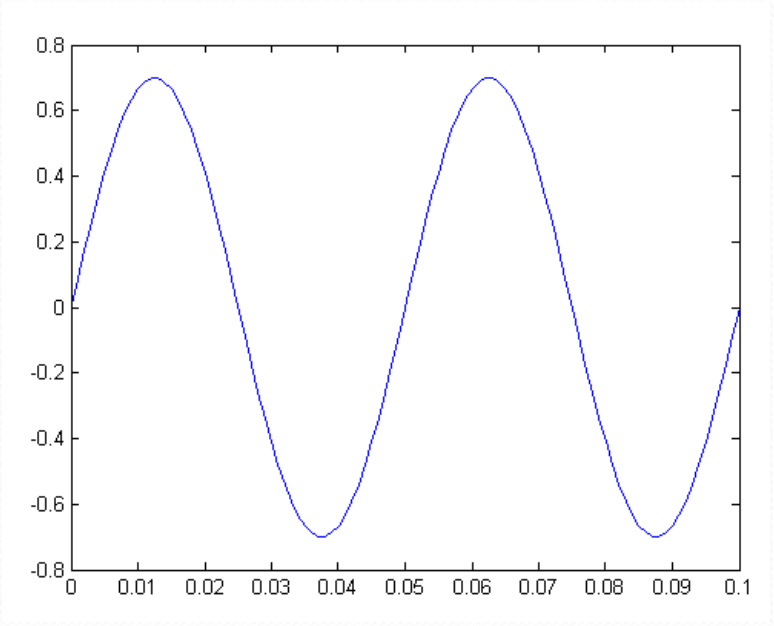
  `figure`(n)

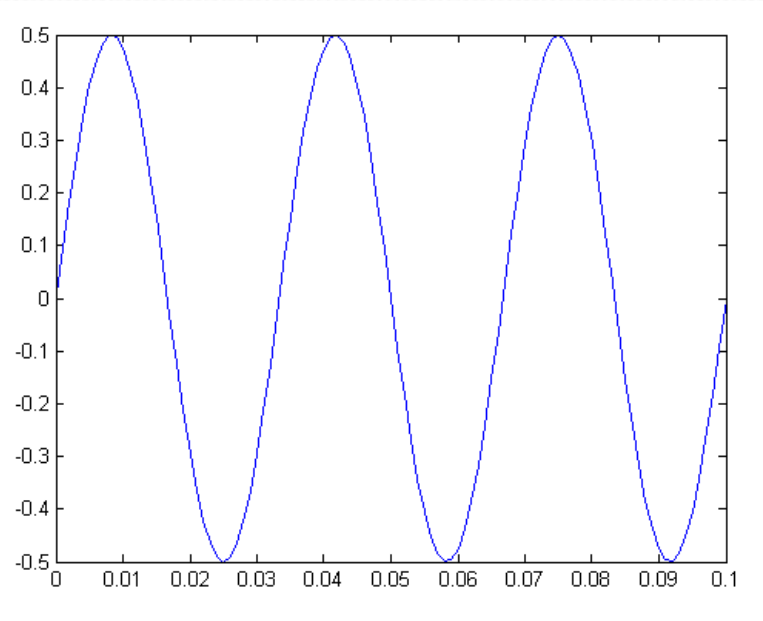
  where *n* is an integer.

  If the argument of `figure` is omitted, then MATLAB gives it the next integer value.

```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

plot(t,y1)
figure
plot(t,y2)
```





Same Figure

Since each graph-generation function creates a new figure window, in order to draw more than one curve, surface, or line (or combination of these) on a given graph, one must use

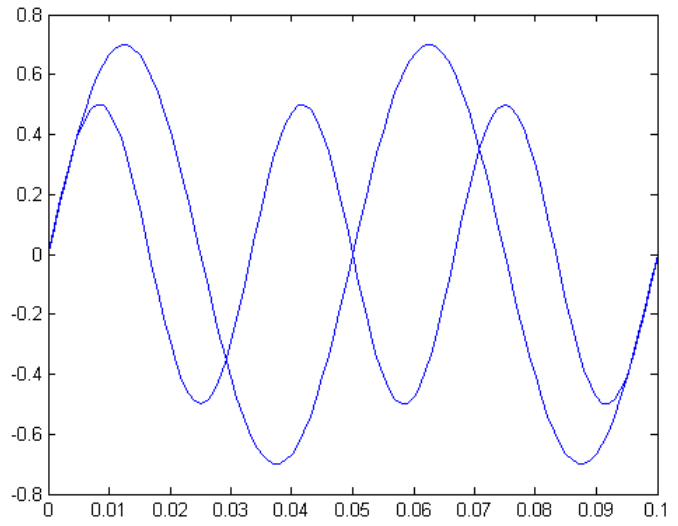
```
hold on
```

which holds the current window (or subplot sector) active.

Example –

```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

plot(t,y1)
hold on
plot(t,y2)
```



When both lines and points are to be plotted, but the points defining the line (*u*1, *v*1) are different from the points that are to be plotted (*u*2, *v*2), we use either

```
plot(u1, v1, c1, u2, v2, c2)
```

or

```
plot(u1, v1, c1)
hold on
plot(u2, v2, c2)
```

where

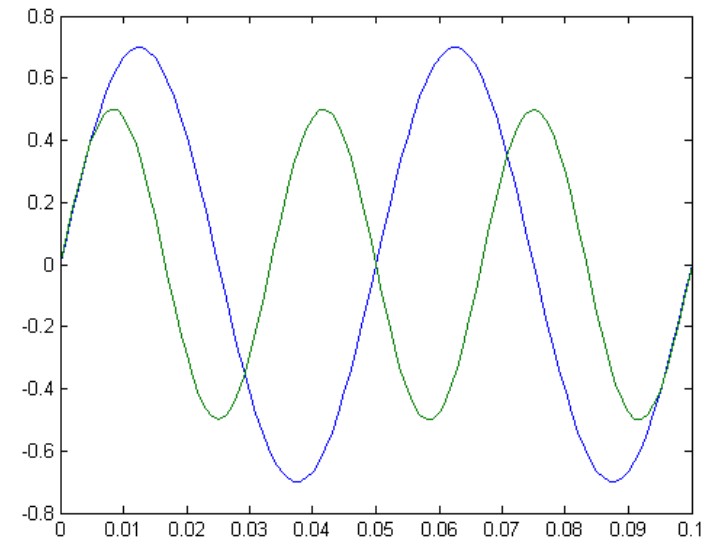
***c*1 contains the symbols for the line type and color.**

***c*2 contains the symbols for the point type and color.**

Example –

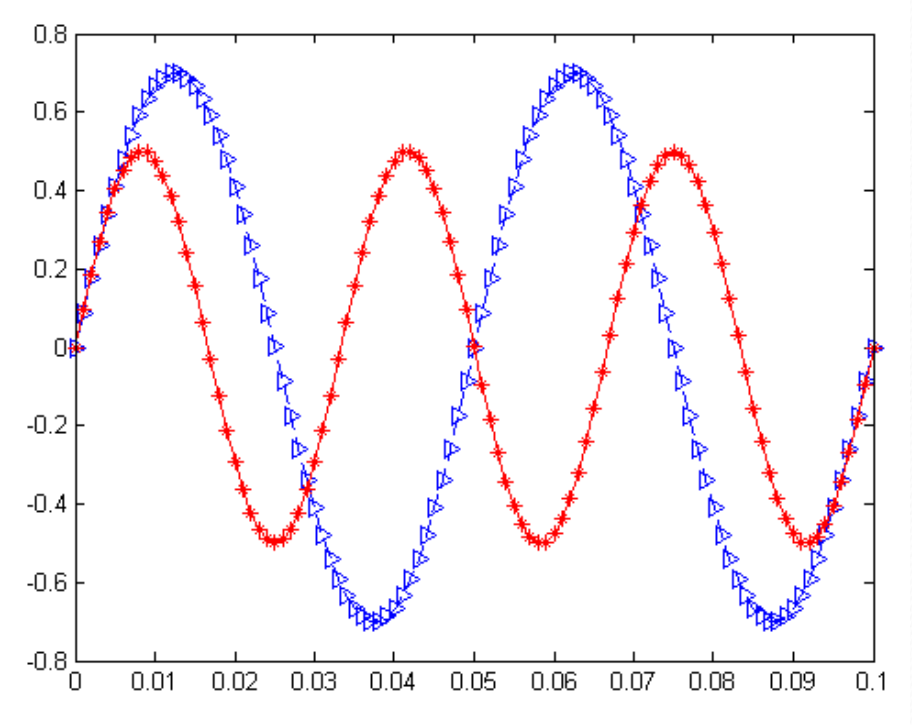
```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

plot(t , y1 , t , y2)
```



```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

plot(t , y1 , 'b-->' , t , y2, 'r-*')
```



- **One can also place several independently created graphs in one figure window with**

  `subplot`(i, j, k)

  where

  ***i*, *j* divide the window into sectors (rows and columns)**

  ***k* indicates in which sector a graph is to be placed**

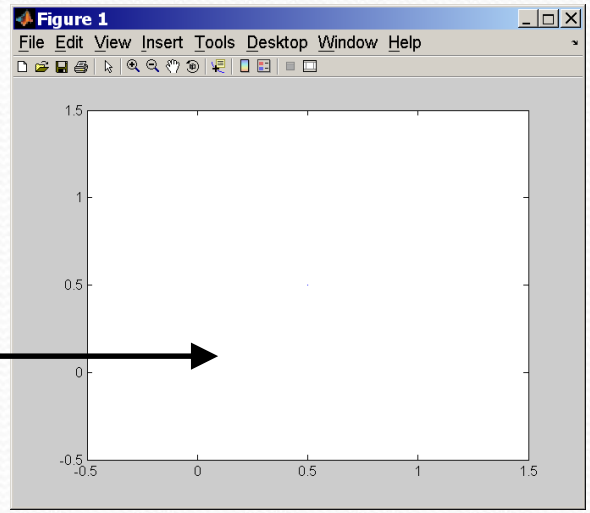
  A value of 1 for this argument indicates the upper left corner, and the product of the number of rows and columns indicates the lower right corner.

  As the numbers increase, they indicate the sectors from left to right, starting at the top row.

Examples –

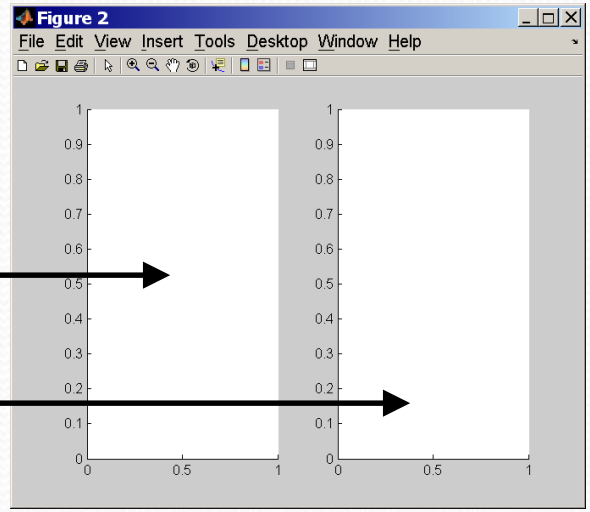
Script or Function

```
figure(1)
plotting expressions
```



Script or Function

```
figure(2)
subplot(1, 2, 1)
plotting expressions
subplot(1, 2, 2)
plotting expressions
```

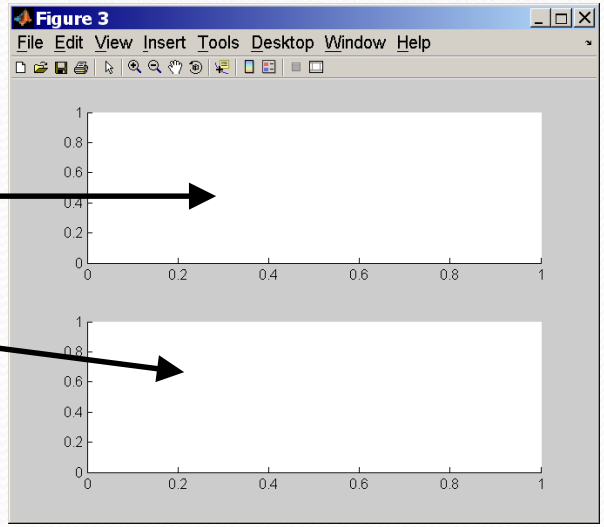


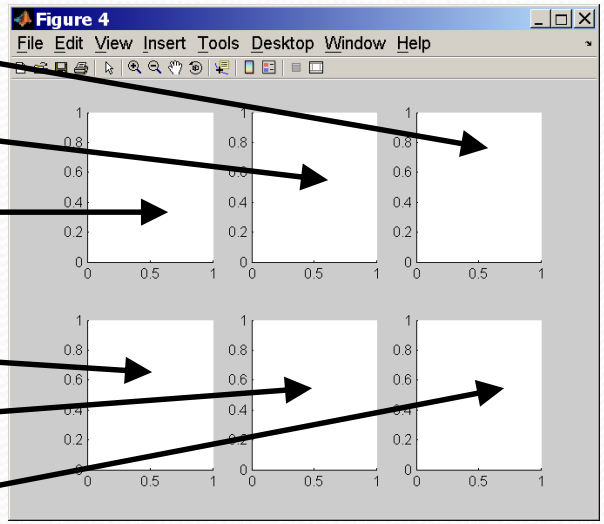
Script or Function

```
figure(3)
subplot(2, 1, 1)
plotting expressions
subplot(2, 1, 2)
plotting expressions
```

Script or Function

```
figure(4)
subplot(2, 3, 3)
plotting expressions
subplot(2, 3, 2)
plotting expressions
subplot(2, 3, 1)
plotting expressions
subplot(2, 3, 4)
plotting expressions
subplot(2, 3, 5)
plotting expressions
subplot(2, 3, 6)
plotting expressions
```

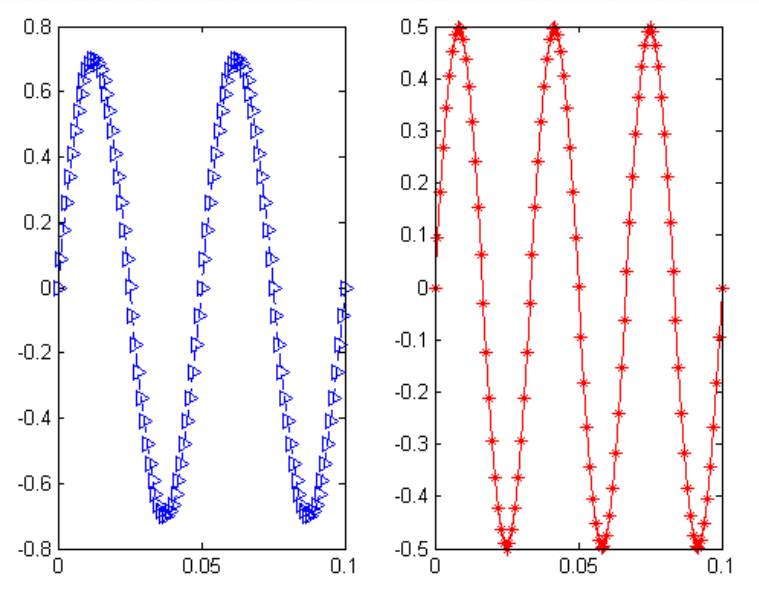




Examples–

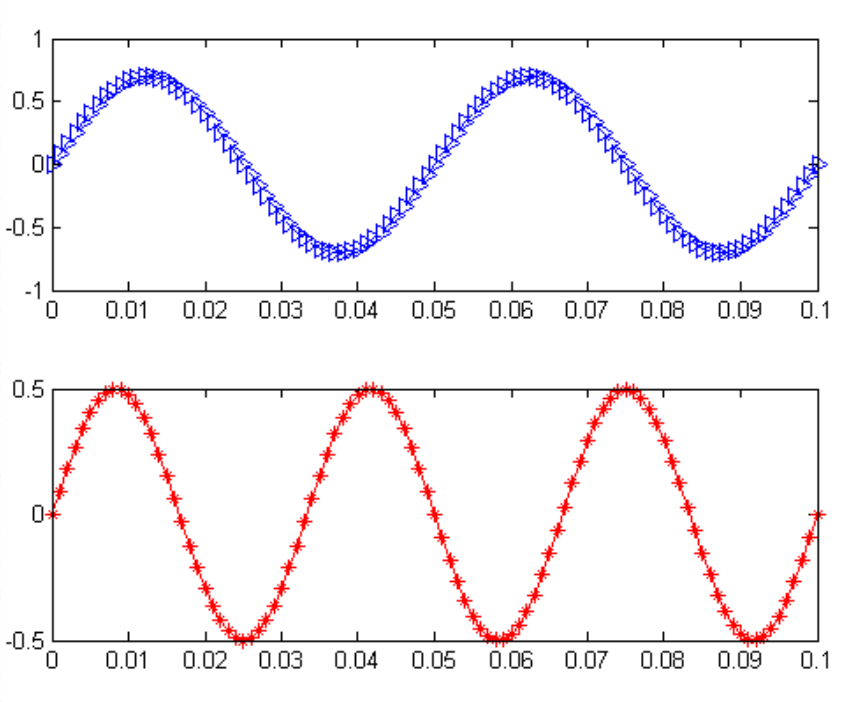
```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

subplot(1,2,1)
plot(t , y1 , 'b-->')
subplot(1,2,2)
plot(t , y2, 'r-*')
```



```
t=0:0.001:0.1;
f1 = 20; %frequency#1 = 20Hz
f2 = 30; %frequency#2 = 30Hz
A1 = 0.7; % amplitude#1
A2 = 0.5; %Amplitude#2
y1 = A1 * sin( 2* pi * f1 * t); % sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2

subplot(2,1,1)
plot(t , y1 , 'b-->')
subplot(2,1,2)
plot(t , y2, 'r-*')
```



```
t=0:0.001:0.1;

f1 = 30; %frequency#1 = 30Hz
f2 = 20; %frequency#2 = 20Hz
f3 = 10; %frequency#3 = 10Hz
f4 = 5;  %frequency#3 = 5Hz

A1 = 0.7; % amplitude#1
A2 = 0.6; %Amplitude#2
A3 = 0.5; %Amplitude#3
A4 = 0.4; %Amplitude#4

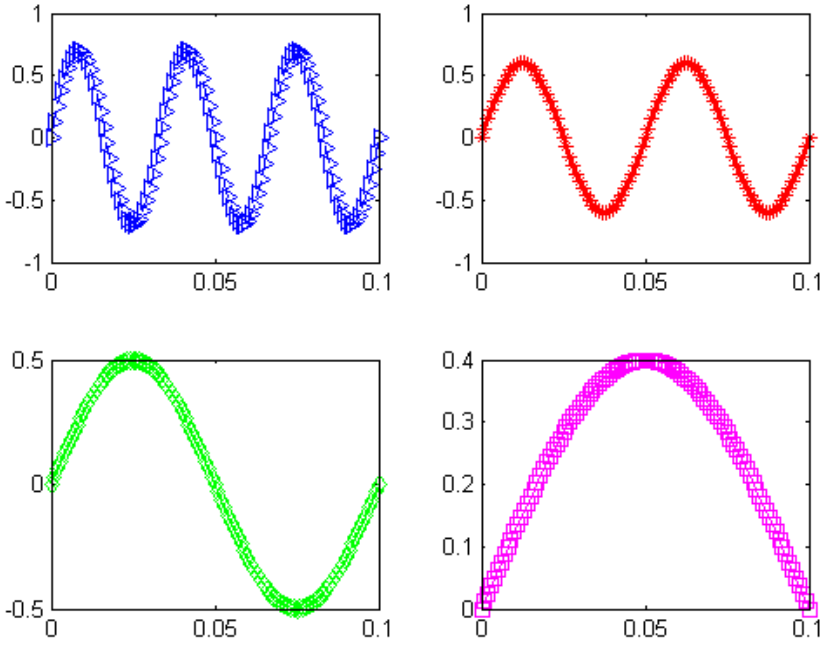
y1 = A1 * sin( 2* pi * f1 * t); %sinusoidal signal#1
y2 = A2 * sin( 2* pi * f2 * t); %sinusoidal signal#2
y3 = A3 * sin( 2* pi * f3 * t); %sinusoidal signal#3
y4 = A4 * sin( 2* pi * f4 * t); %sinusoidal signal#4

subplot(2,2,1)
plot(t , y1 , 'b-->')

subplot(2,2,2)
plot(t , y2, 'r-*')

subplot(2,2,3)
plot(t , y3, 'g-d')

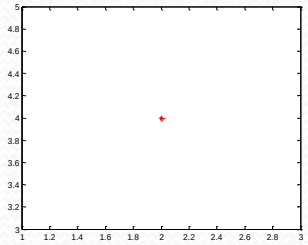
subplot(2,2,4)
plot(t , y4, 'm-s')
```



Points

To place a red asterisk at the location (2,4), we use

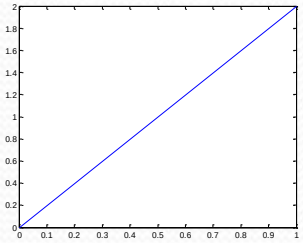
```
plot(2, 4, 'r*')
```



Lines

To draw a straight line that goes from (0,0) to (1,2) using the default line type (solid) and the default color (blue), we use

```
plot([0, 1], [0, 2])
```

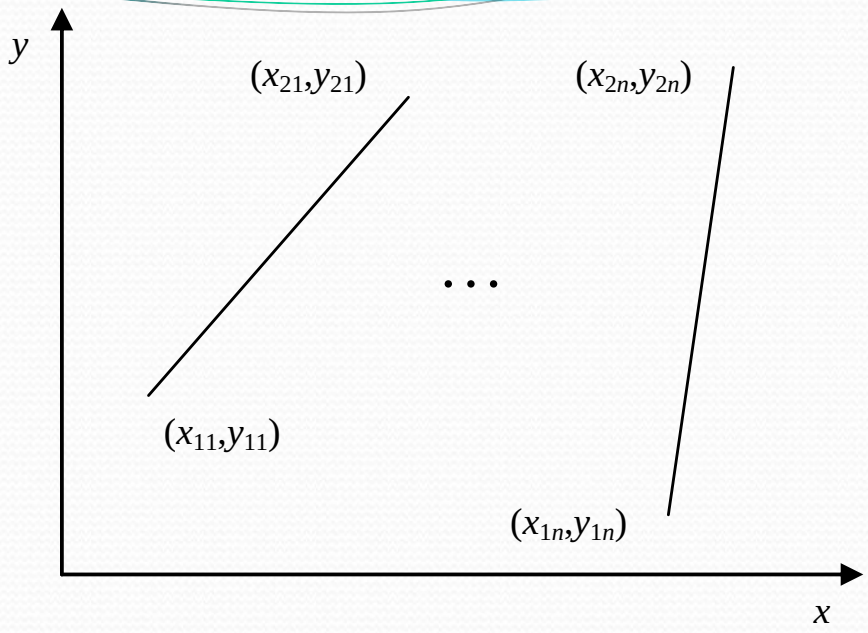


Notice that the first two-element vector [0, 1] represents the values of the *x* coordinates, and the second two-element vector [0, 2] represents the values of the *y* coordinates.

To draw a set of n unconnected lines whose end points are (x_{1k},y_{1k}) and (x_{2k},y_{2k}), $k = 1,2,\ldots,n$, we create four vectors

$$x_j = \begin{bmatrix} x_{j1} & x_{j2} & \ldots & x_{jn} \end{bmatrix}$$

$$y_j = \begin{bmatrix} y_{j1} & y_{j2} & \ldots & y_{jn} \end{bmatrix} \quad j = 1,2$$



Then, the plot command is

```
x1 = [...];   x2 = [...];
y1 = [...];   y2 = [...];
plot([x1; x2], [y1; y2])
```

where [x1; x2] and [y1; y2] are each (2×n) arrays.

This plot expression is a generalization of the plotting function `stem`, which assumes that $x1 = x2$ and that $y1 = 0$.

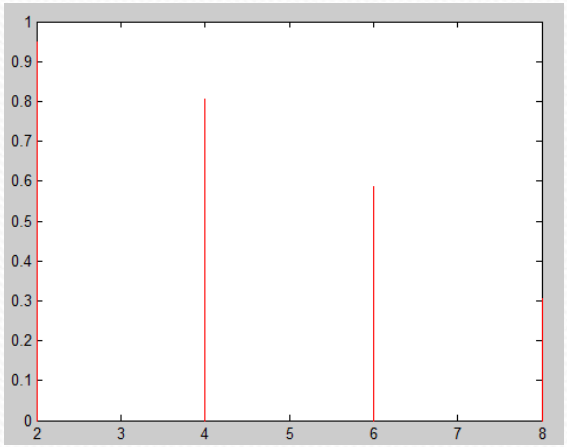
Example –

Let us draw four black vertical lines from $y = 0$ to $y = \cos(\pi x/20)$ when $x = 2, 4, 6,$ and 8.

The script is

```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x], y, 'r')
```

where we have used the fact that $x_{1k} = x_{2k}$, $k = 1, \ldots, n$.

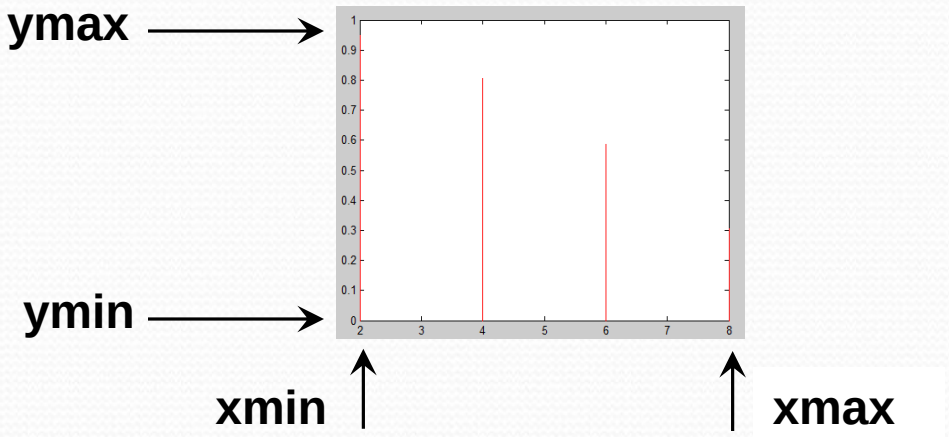


Unfortunately, because of automatic scaling of the axes, the first and last lines are coincident with the box around the figure.

Therefore, to adjust the axes so that these lines are visible, we use

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

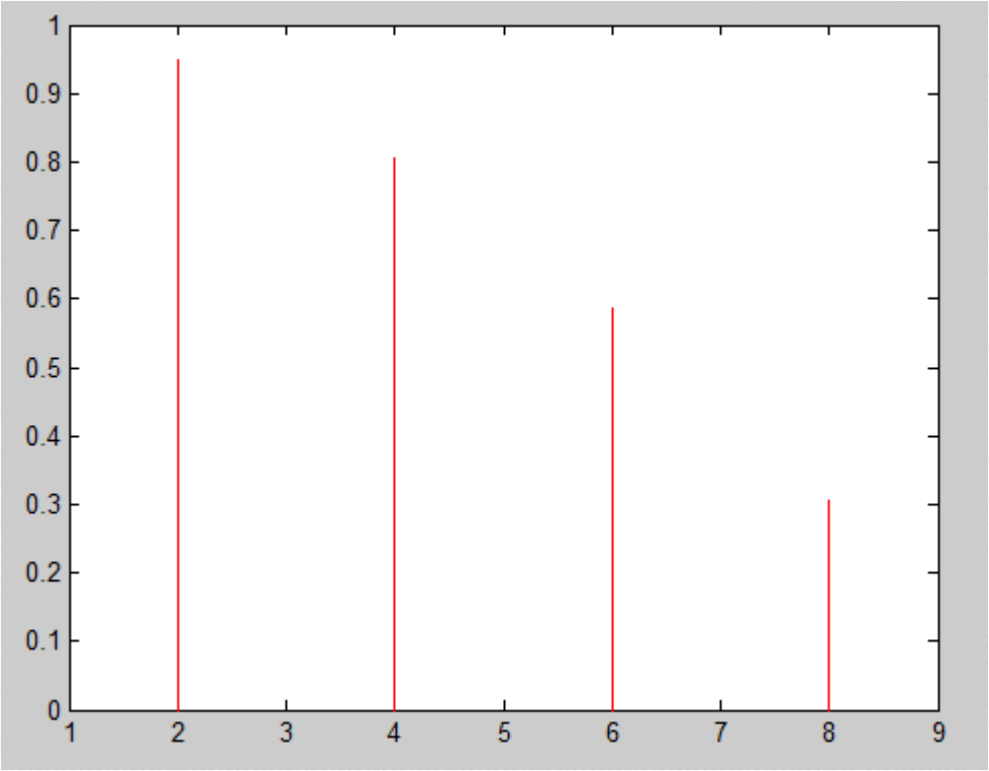
where *xmin*, *xmax*, *ymin*, and *ymax* are the minimum and maximum values of the *x* and *y* axes, respectively.



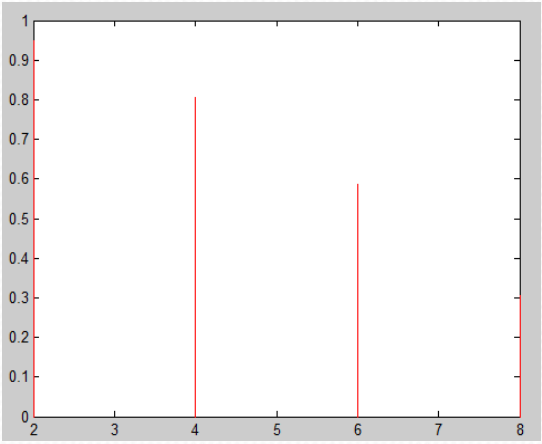
Example – continued

The revised script is

```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x], y, 'r')
axis([1, 9, 0, 1])
```



vs.

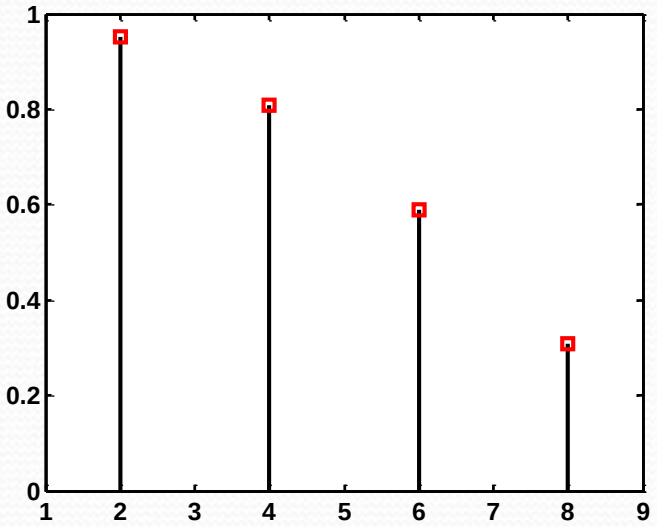


Example – continued

To place a red square at the end of each vertical line that is off the *x*-axis, we have to add another triplet of instructions in the plot command

Thus, the script becomes

```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x], y, 'k', x, cos(pi*x/20), 'rs')
axis([1, 9, 0, 1])
```

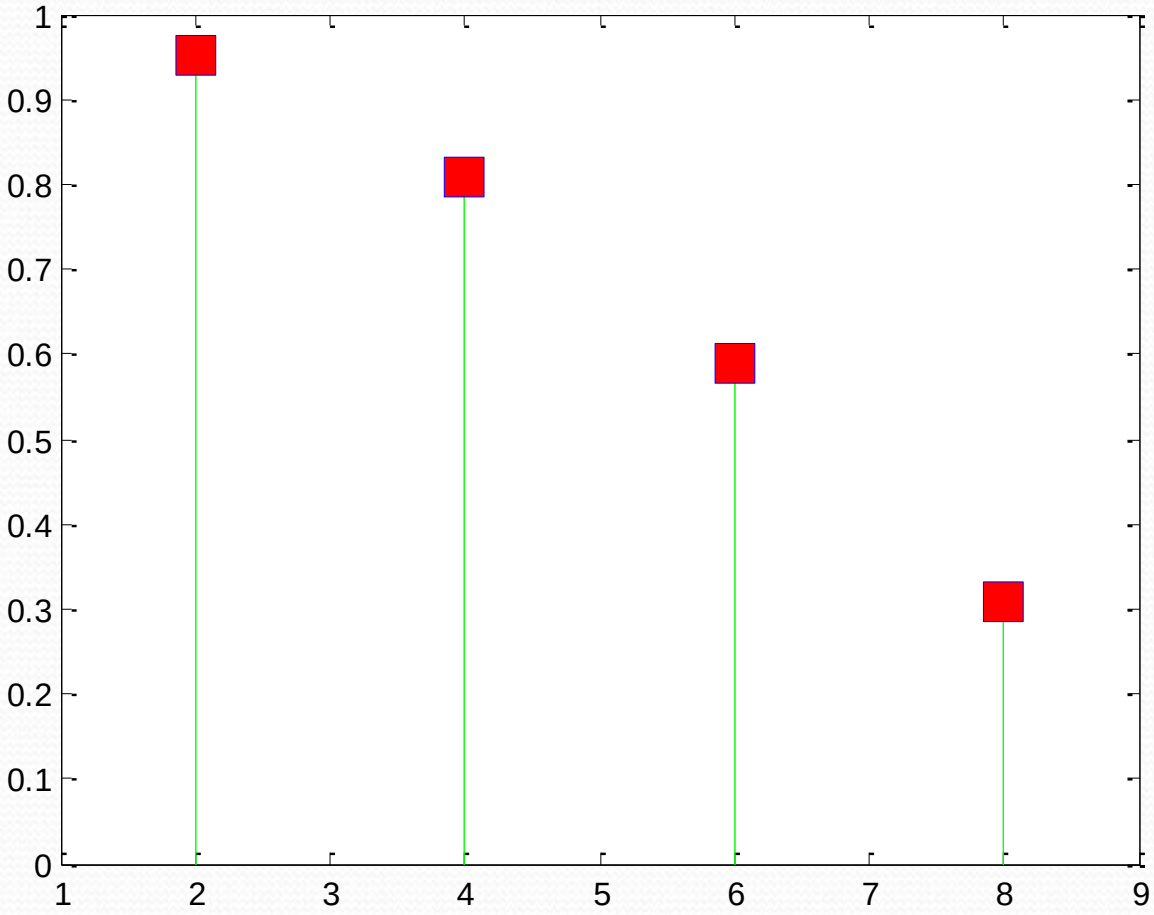


Example – continued

To change the square with red edges to a solid red square with a blue edges that is larger than the default size:

The script becomes

```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x], y, 'g')
hold on
plot(x, cos(pi*x/20), 's', 'MarkerEdgeColor', 'b', ...
    'MarkerFaceColor', 'r', 'MarkerSize', 14)
axis([1, 9, 0, 1])
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9

Example – continued

```
x = linspace(0, 3.5);
plot(x, [0.1*x.^2; cos(x).^2; exp(-0.3*x)], 'k')
```

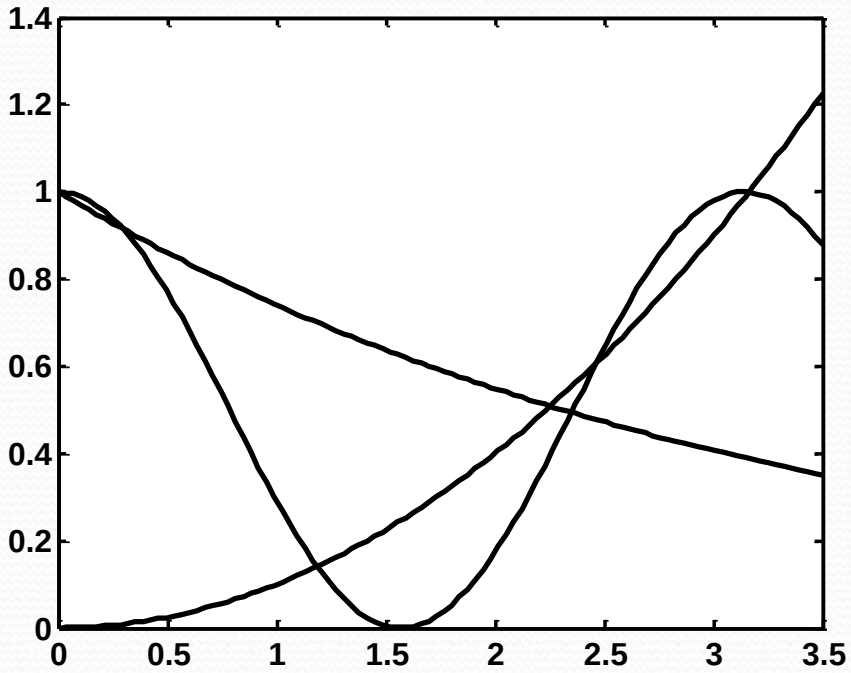
or

```
x = linspace(0, 3.5);
plot(x, 0.1*x.^2, 'k', x, cos(x).^2, 'k', x, exp(-0.3*x), 'k')
```

or

```
x = linspace(0, 3.5);
plot(x, 0.1*x.^2, 'k')
hold on
plot(x, cos(x).^2, 'k')
plot(x, exp(-0.3*x), 'k')
```

Note: Color specification is required.

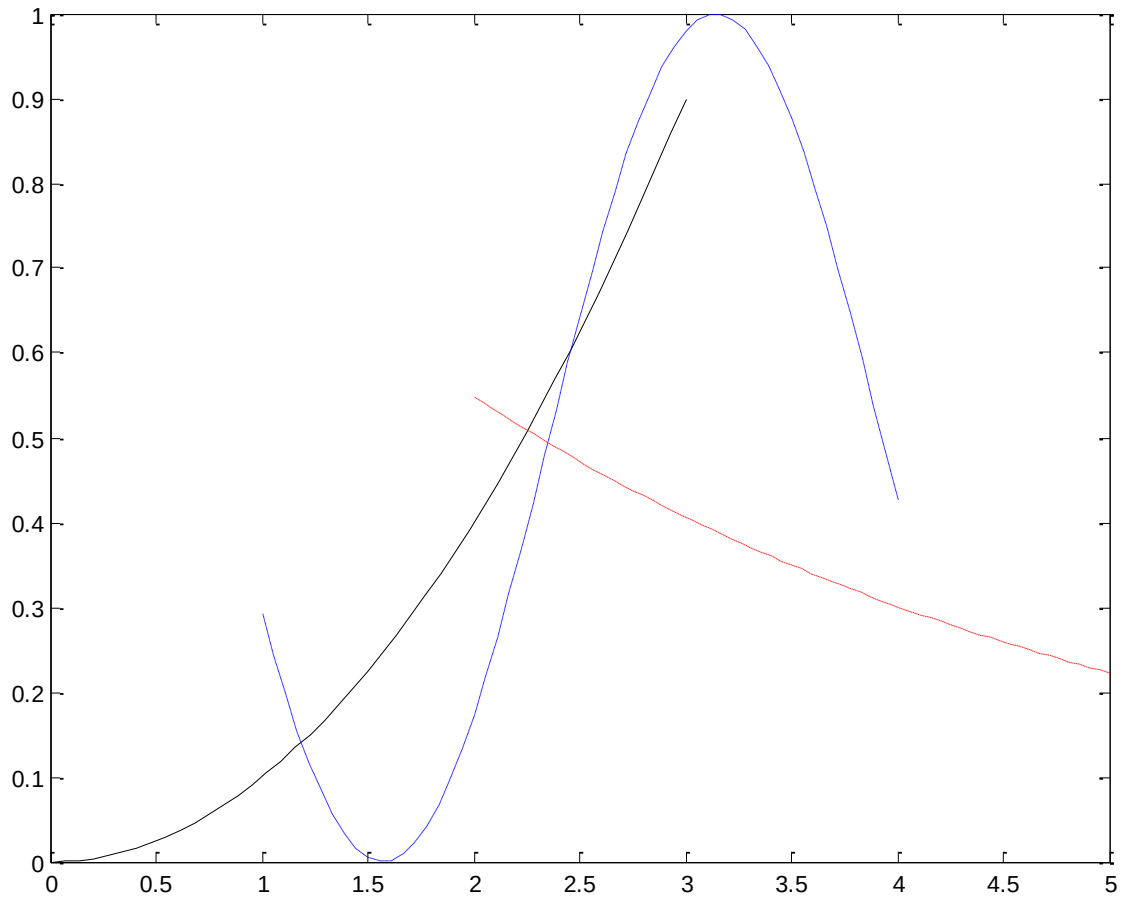


Example – continued

If the range of the independent variable for each of these functions is different, then only the second and third scripts can be used.

For example, if $0 \le x \le 3$, $1 \le y \le 4$, and $2 \le z \le 5$, then

```
x = linspace(0, 3, 45);
y = linspace(1, 4, 55);
z = linspace(2, 5, 65);
plot(x, 0.1*x.^2, 'k-', y, cos(y).^2, 'b--', z, exp(-0.3*z), 'r-.')
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Changing Graph Appearance

box, grid, and axis

Several functions can be used to change the appearance of a graph -

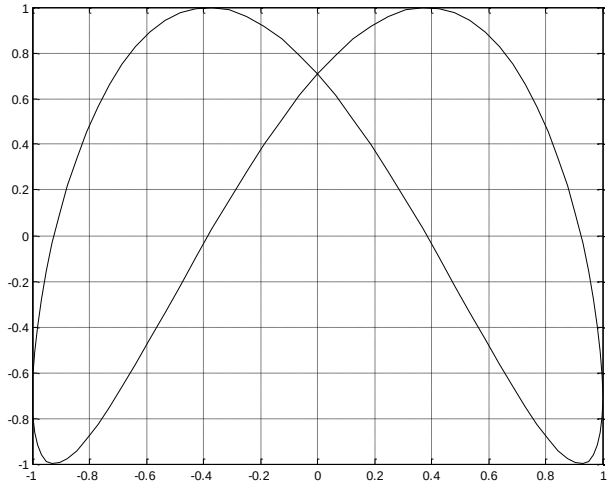
`axis on` or `axis off` [default – on]

`box on` or `box off` [default – on]

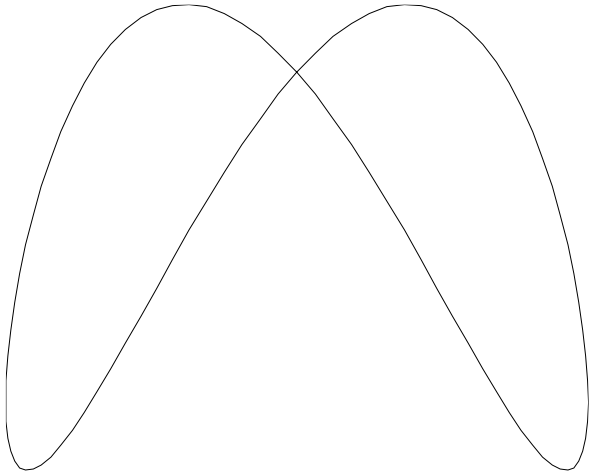
`grid on` or `grid off` [default – off]

The function `box on` only works when `axis on` has been selected.

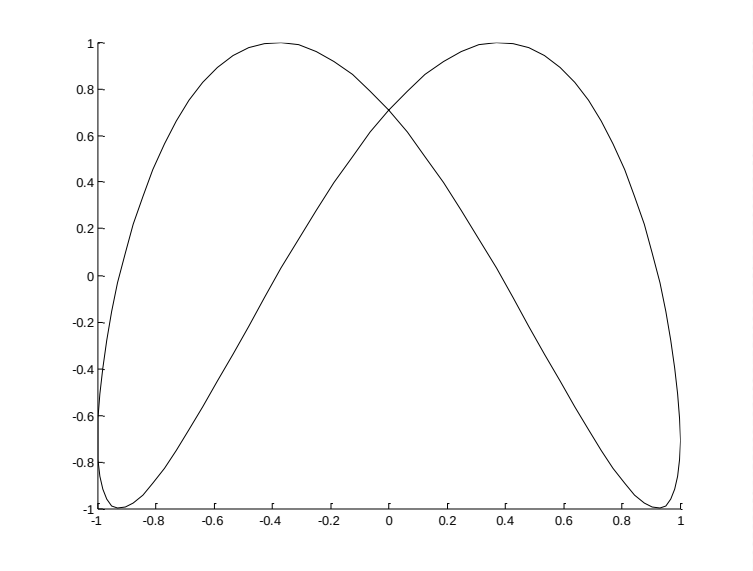
```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box on
grid on
```




```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box off
grid off
axis off
```



```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box off
grid off
```



Graph Annotations and Visual Enhancement

We shall illustrate by example how to enhance a graph -

- **With axis labels, figure titles, labeled curves, legends, filled areas, and placement of text.**
- **By altering the attributes of the axes, curve lines, and text.**
- **By using Greek letters, mathematical symbols, and subscripts and superscripts.**
- **By positioning one figure inside another figure.**
- **By using the interactive plotting tools.**
- **By using animation.**

Axes and Curve Labels, Figure Titles, Legends, and Text Placement

The functions that are used to label the *x* and *y* axes and to place a title above the graph are, respectively,

```
xlabel('.....')
ylabel('.....')
title('.....')
```

where *s* is a string.

The function that places text anywhere in the figure window is

```
text(x, y, '.....')
```

where *x* and *y* are the coordinates of where the text given by the string *s* will be placed.

Example –

Let us draw, label, title, and annotate the relationship of two intersecting curves, cos(*x*) and 1/cosh(*x*), over the range $0 \leq x \leq 6$.

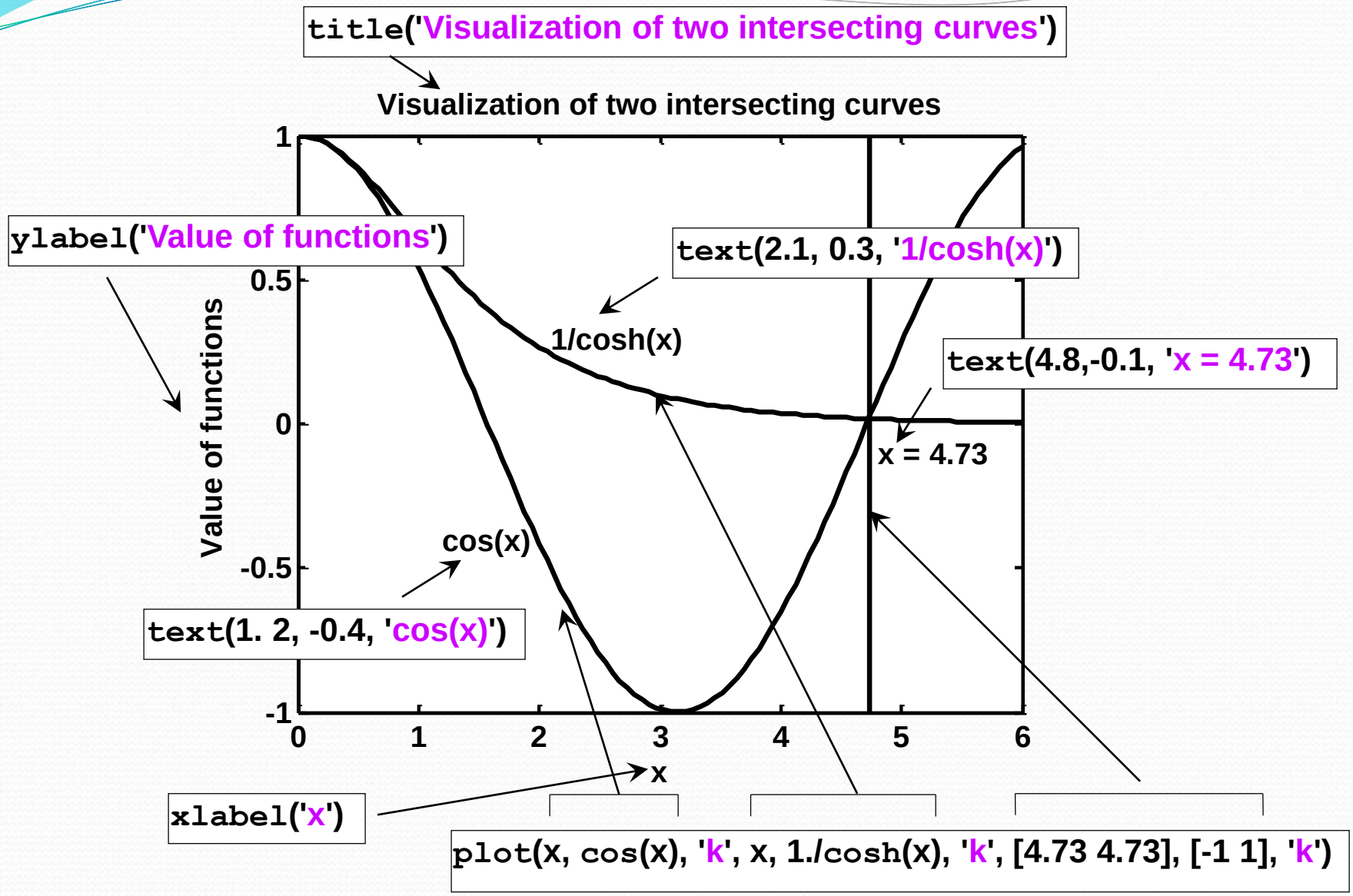
In this range, these two curves intersect at $x = 0$ and $x = 4.73$.

We shall also draw a vertical line through the intersecting point $x = 4.73$ and denote the value of *x* near this intersection.

The script to perform these operations is

```
x = linspace(0, 6, 100);
plot(x, cos(x), 'k', x, 1./cosh(x), 'k', [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Value of functions')
title('Visualization of two intersecting curves')
text(4.8, -0.1, 'x = 4.73')
text(2.1, 0.3, '1/cosh(x)')
text(1.2, -0.4, 'cos(x)')
```

Note – coordinates for the text placements have been found interactively (i.e., with a little trial and error).


title('Visualization of two intersecting curves')
Visualization of two intersecting curves
ylabel('Value of functions')
Value of functions
text(2.1, 0.3, '1/cosh(x)')
1/cosh(x)
text(4.8,-0.1, 'x = 4.73')
x = 4.73
cos(x)
text(1. 2, -0.4, 'cos(x)')
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
x
xlabel('x')
plot(x, cos(x), 'k', x, 1./cosh(x), 'k', [4.73 4.73], [-1 1], 'k')

Another way to identify the curves is with

`legend`(s1, s2, …, sn, 'Location', 'Keyword')

where

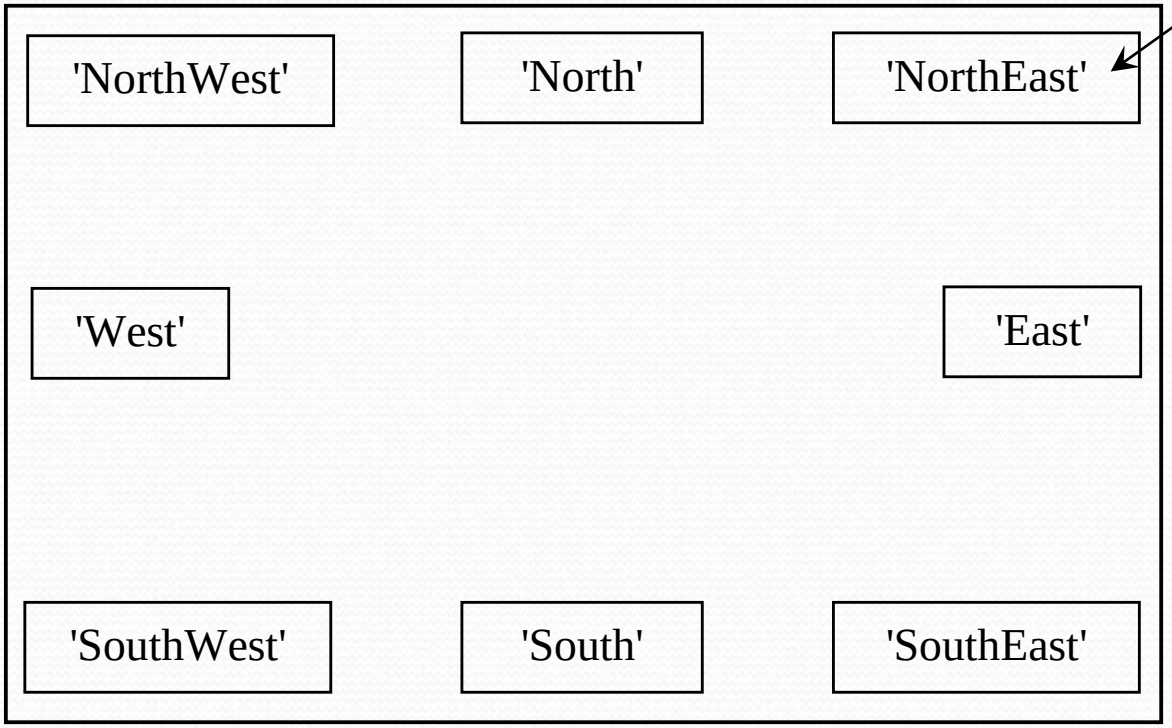
s1, etc, are the strings containing the alphanumeric identifier for each line that will appear in the legend box in the order that they are drawn.

***Keyword* is an optional string that places the legend in one of the 8 predetermined locations on the graph.**

The number of arguments of legend is less than or equal to the number of different lines being drawn by one or more plot functions.

The legend has to be placed after all the plot functions have been executed.

'Keyword' =



Default location

The box around the legend can be toggled on and off with

```
legend('boxon')   % Default
```

and

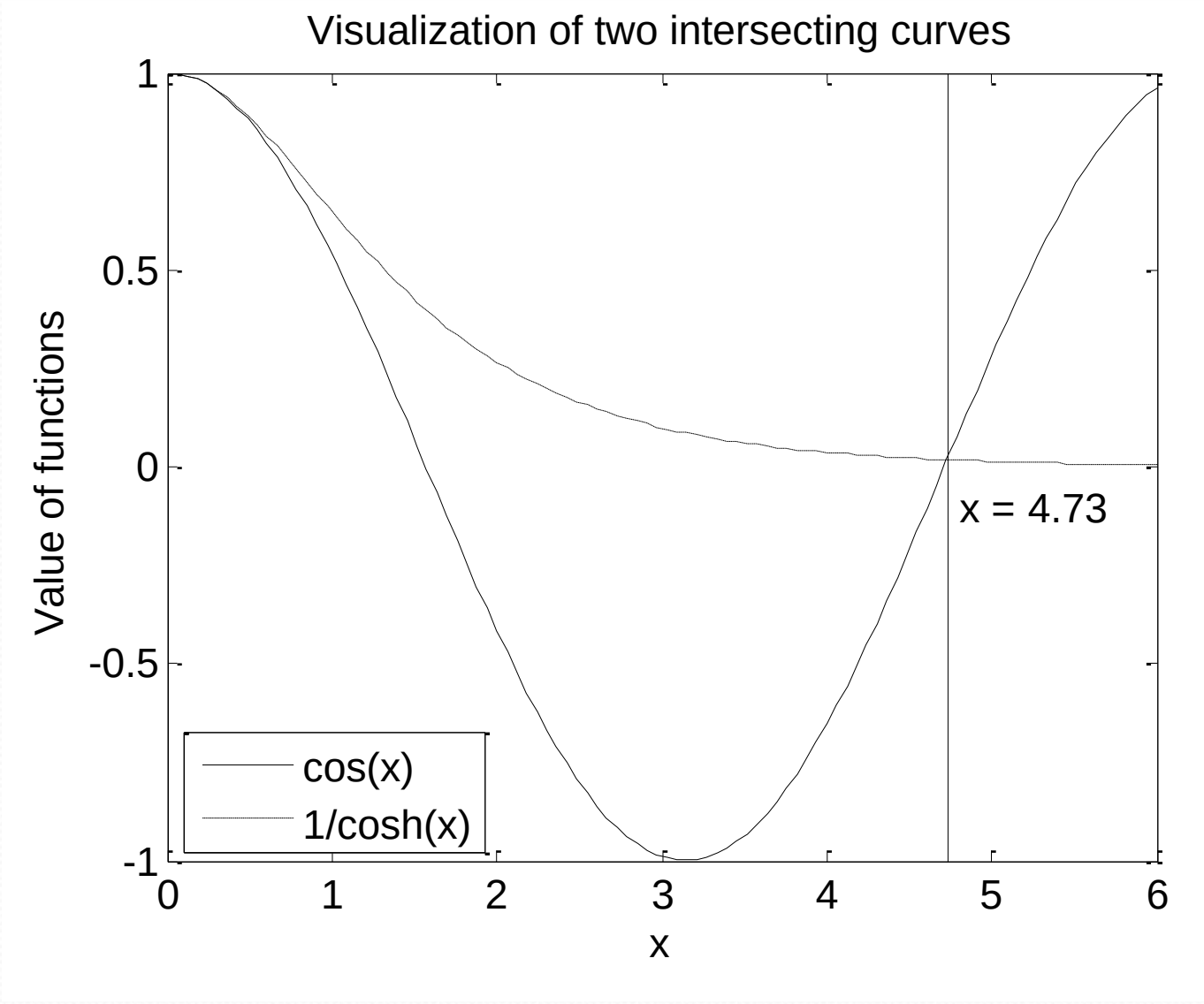
```
legend('boxoff')
```

`legend` differs from `text` in that `text` can be used as many times as practical, whereas `legend` can only be used once.

Also, `legend` places all the text within a box.

Example –

```
x = linspace(0,6, 100);
plot(x, cos(x), 'k-', x, 1./cosh(x), 'k--',
    [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Value of functions')
title('Visualization of two intersecting curves')
text(4.8, -0.1, 'x = 4.73')
legend('cos(x) ', '1/cosh(x) ', 'Location', 'Southwest')
```


Visualization of two intersecting curves
Value of functions
x
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
x = 4.73
cos(x)
1/cosh(x)

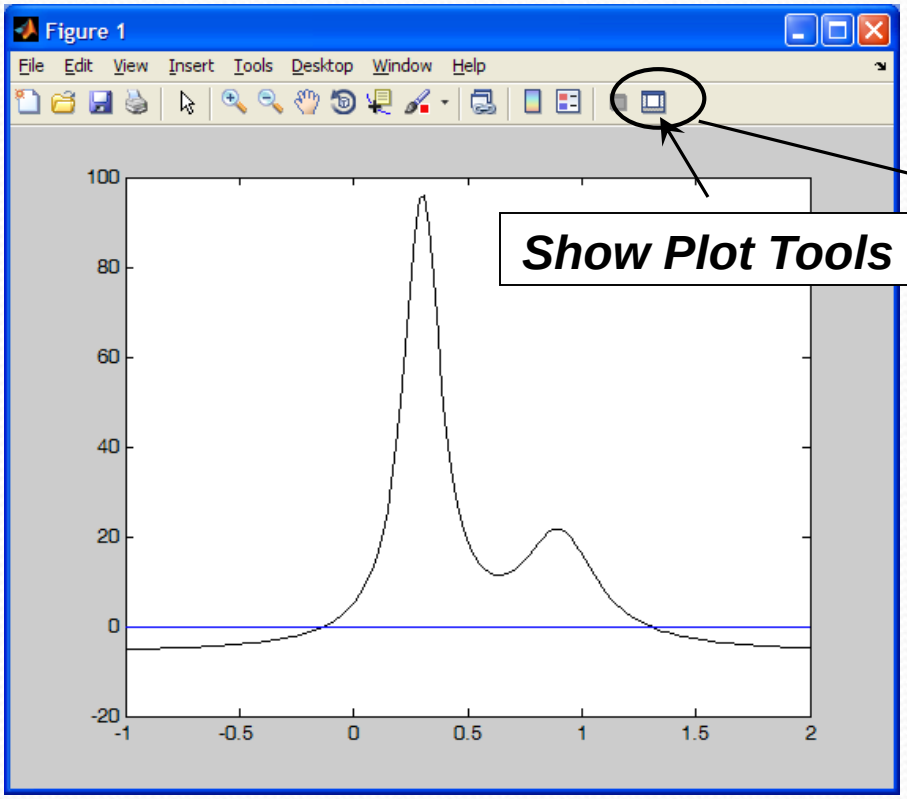
Interactive Plotting Tools

Changes to the attributes of the text, curves, and axes can be made interactively in the figure window by selecting the appropriate operation from its *Tools* menu.

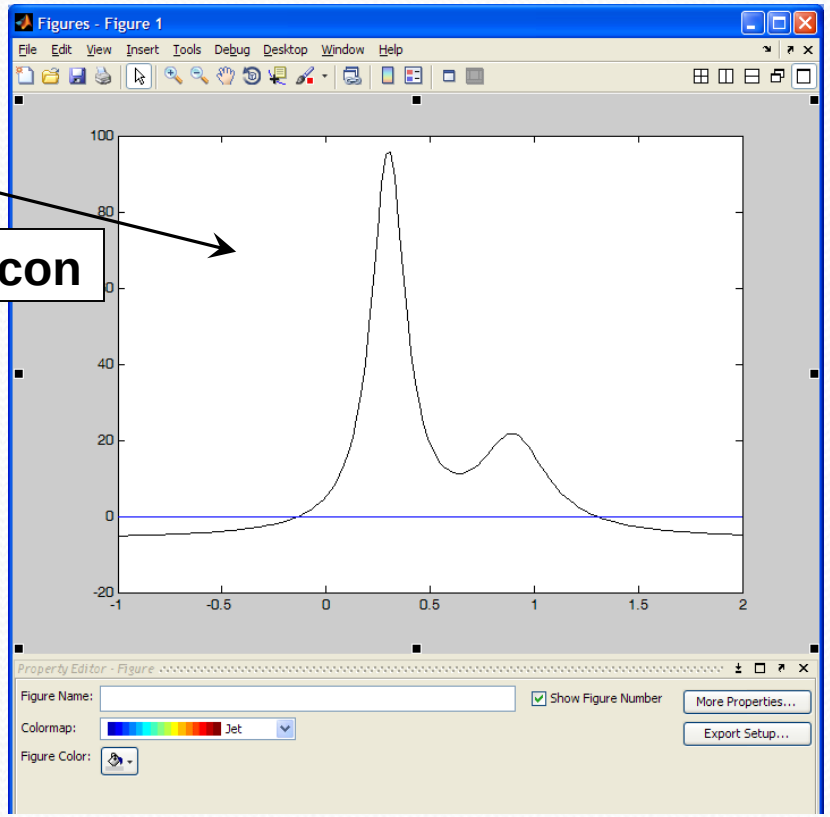
After the changes have been made, the figure can be saved as a function M file.

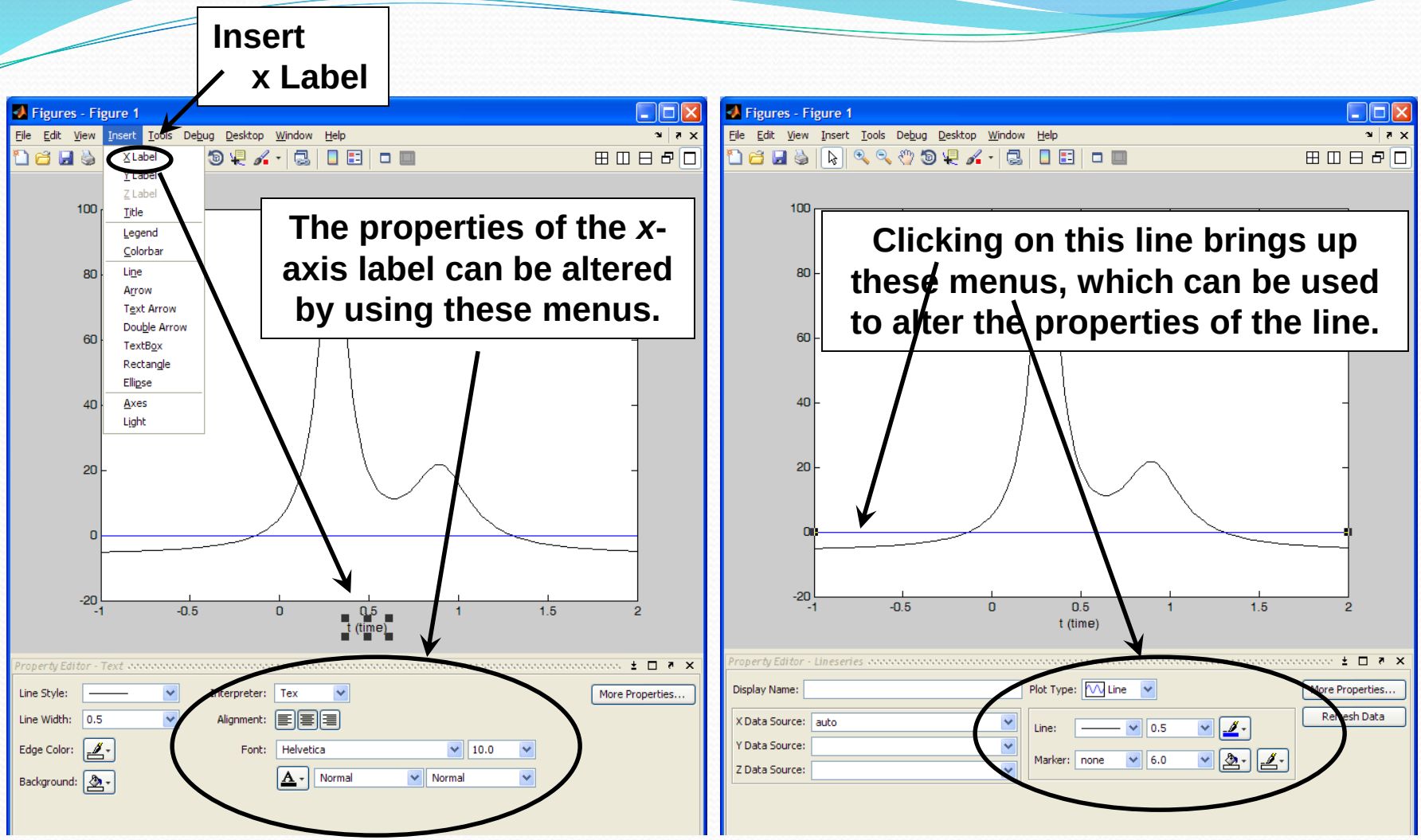
We shall illustrate how to use these interactive tools by using them to make a few modifications to the following program.

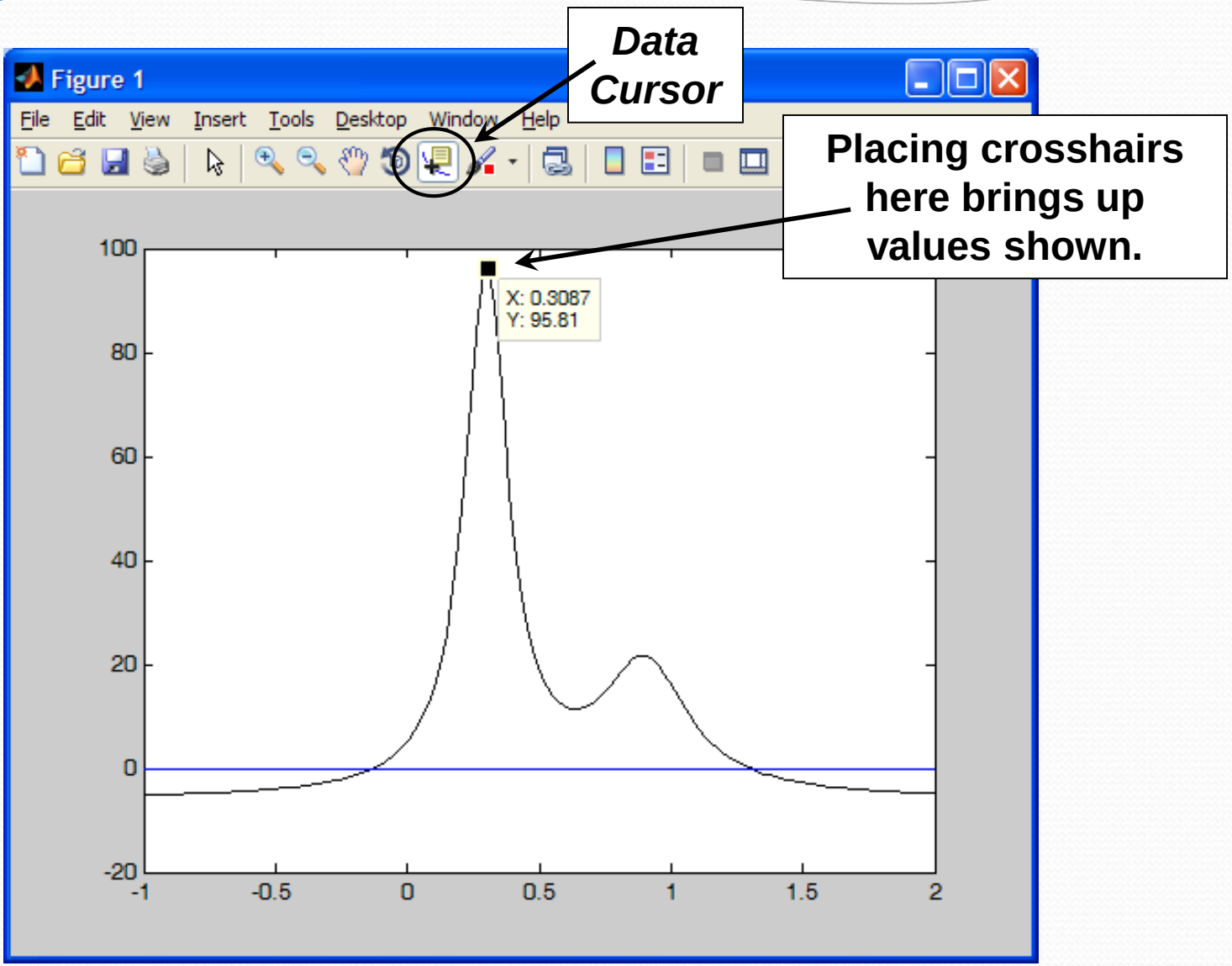
```
x = linspace(-1, 2, 150);
plot(x, humps(x), 'k-', [-1, 2], [0, 0], 'b-')
```

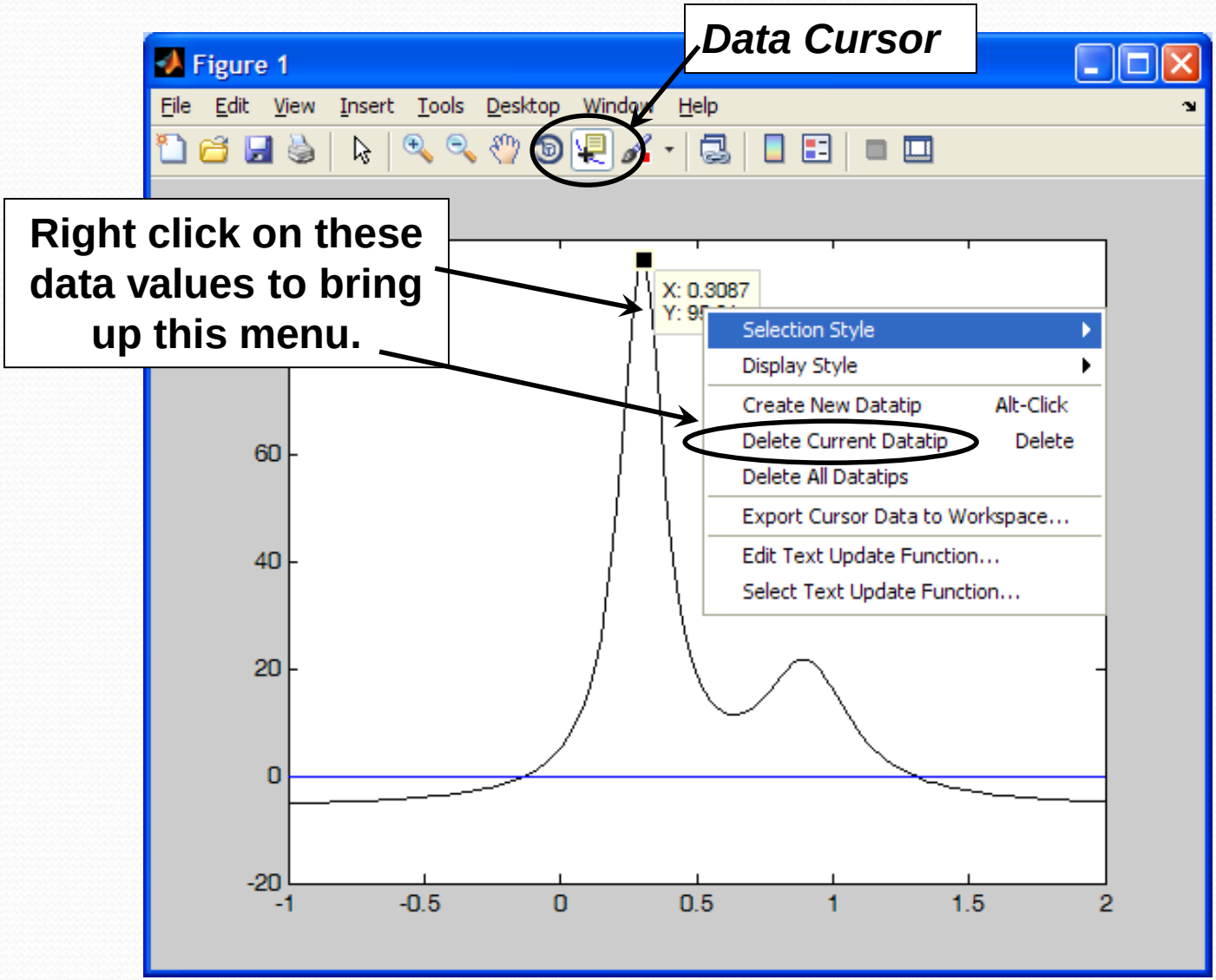


***Show Plot Tools* icon**




Insert
x Label
Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
X Label
Y Label
Z Label
Title
Legend
Colorbar
Line
Arrow
Text Arrow
Double Arrow
TextBox
Rectangle
Ellipse
Axes
Light
The properties of the x-axis label can be altered by using these menus.
t (time)
Property Editor - Text
Line Style:
Line Width:
0.5
Edge Color:
Background:
Interpreter:
Tex
Alignment:
Font:
Helvetica
10.0
Normal
Normal
More Properties...
Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Clicking on this line brings up these menus, which can be used to alter the properties of the line.
t (time)
Property Editor - Lineseries
Display Name:
Plot Type:
Line
More Properties...
Refresh Data
X Data Source:
auto
Y Data Source:
Z Data Source:
Line:
0.5
Marker:
none
6.0


Figure 1
File Edit View Insert Tools Desktop Window Help
Data Cursor
Placing crosshairs here brings up values shown.
X: 0.3087
Y: 95.81
100
80
60
40
20
0
-20
-1
-0.5
0
0.5
1
1.5
2


Data Cursor
Figure 1
File Edit View Insert Tools Desktop Window Help
Right click on these data values to bring up this menu.
X: 0.3087
Selection Style
Display Style
Create New Datatip Alt-Click
Delete Current Datatip Delete
Delete All Datatips
Export Cursor Data to Workspace...
Edit Text Update Function...
Select Text Update Function...
60
40
20
0
-20
-1
-0.5
0
0.5
1
1.5
2