

MATLAB

Two Dimensional Plots

Topics Covered:

1. Plotting basic 2-D plots.

The **plot** command.

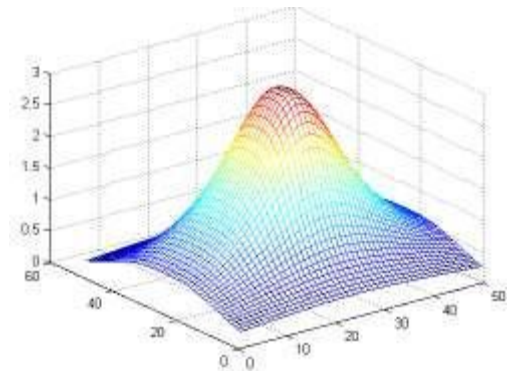
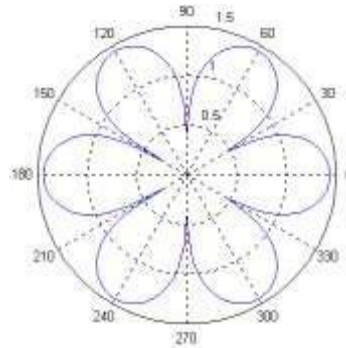
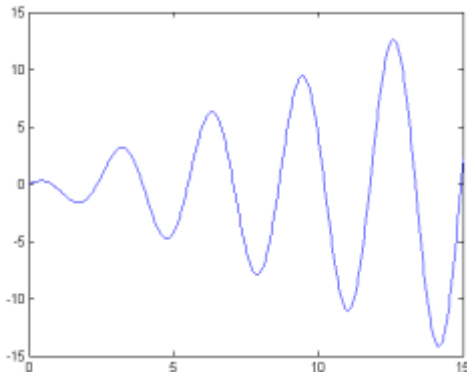
The **fplot** command.

Plotting multiple graphs in the same plot.

Formatting plots.

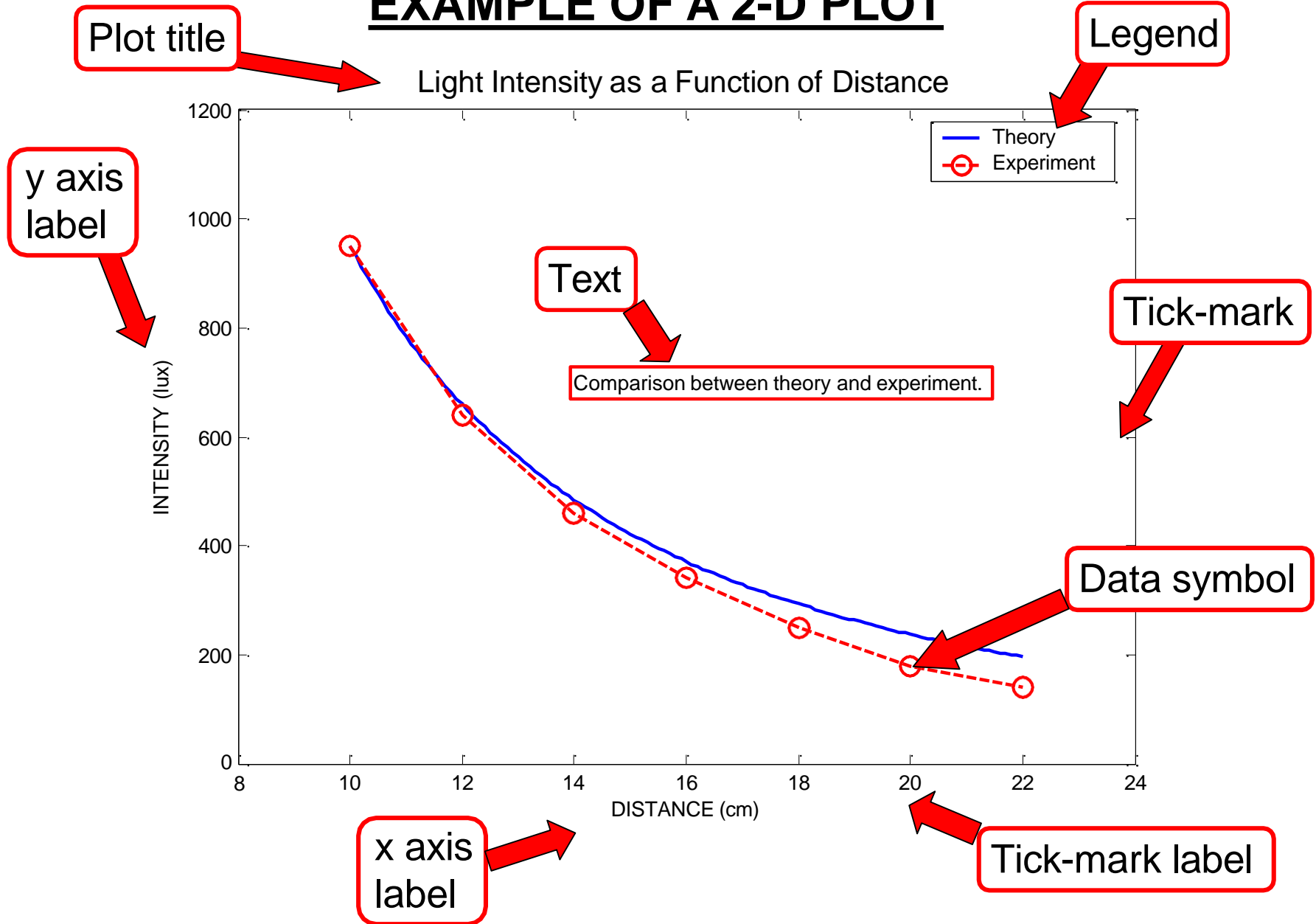
MAKING X-Y PLOTS

MATLAB has many functions and commands that can be used to create various types of plots.



In our class we will only create two dimensional x – y plots.

EXAMPLE OF A 2-D PLOT



PLOT OF GIVEN DATA

Given data:

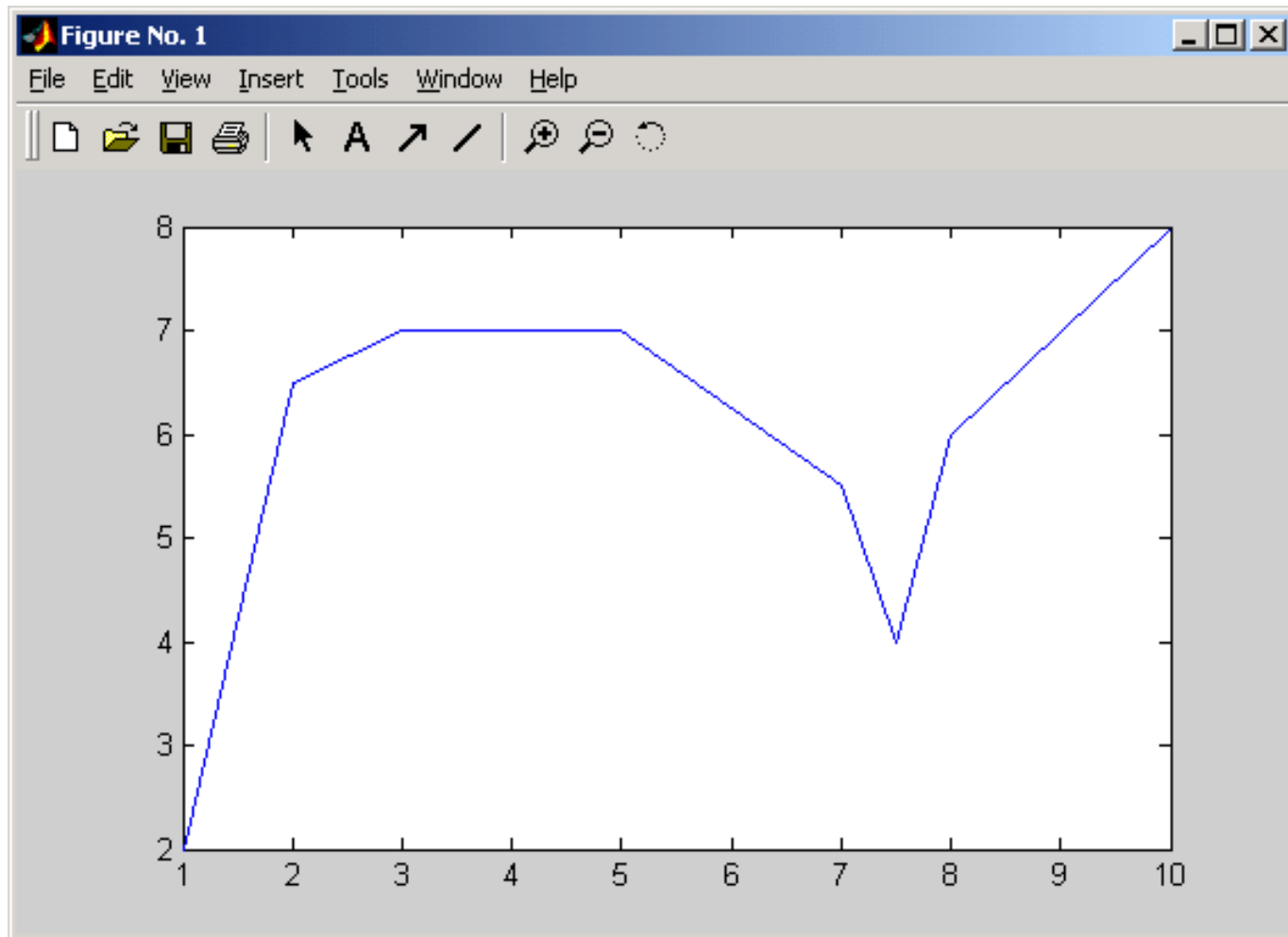
x	1	2	3	5	7	7.5	8	10
y	2	6.5	7	7	5.5	4	6	8

A plot can be created by the commands shown below. This can be done in the Command Window, or by writing and then running a script file.

```
>> x=[1 2 3 5 7 7.5 8 10];  
>> y=[2 6.5 7 7 5.5 4 6 8];  
>> plot(x,y)
```

Once the plot command is executed, the Figure Window opens with the following plot.

PLOT OF GIVEN DATA



LINE SPECIFIERS IN THE `plot()` COMMAND

Line specifiers can be added in the **`plot`** command to:

- Specify the style of the line.
- Specify the color of the line.
- Specify the type of the markers (if markers are desired).

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

LINE SPECIFIERS IN THE `plot()` COMMAND

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

Line Style	Specifier	Line Color	Specifier	Marker Type	Specifier
Solid	-	red	r	plus sign	+
dotted	:	green	g	circle	o
dashed	--	blue	b	asterisk	*
dash-dot	-.	Cyan	c	point	.
		magenta	m	square	s
		yellow	y	diamond	d
		black	k		

LINE SPECIFIERS IN THE `plot()` COMMAND

- The specifiers are typed inside the `plot()` command as strings.
- Within the string the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all the three can be included in a command.

EXAMPLES:

`plot(x,y)`

A solid blue line connects the points with no markers.

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

A solid red line connects the points with no markers.

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

A yellow dashed line connects the points.

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

The points are marked with * (no line between the points.)

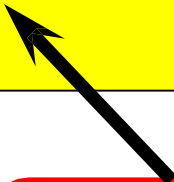
`plot(x,y,'g:d')`

A green dotted line connects the points which are marked with diamond markers.

PLOT OF GIVEN DATA USING LINE SPECIFIERS IN THE `plot()` COMMAND

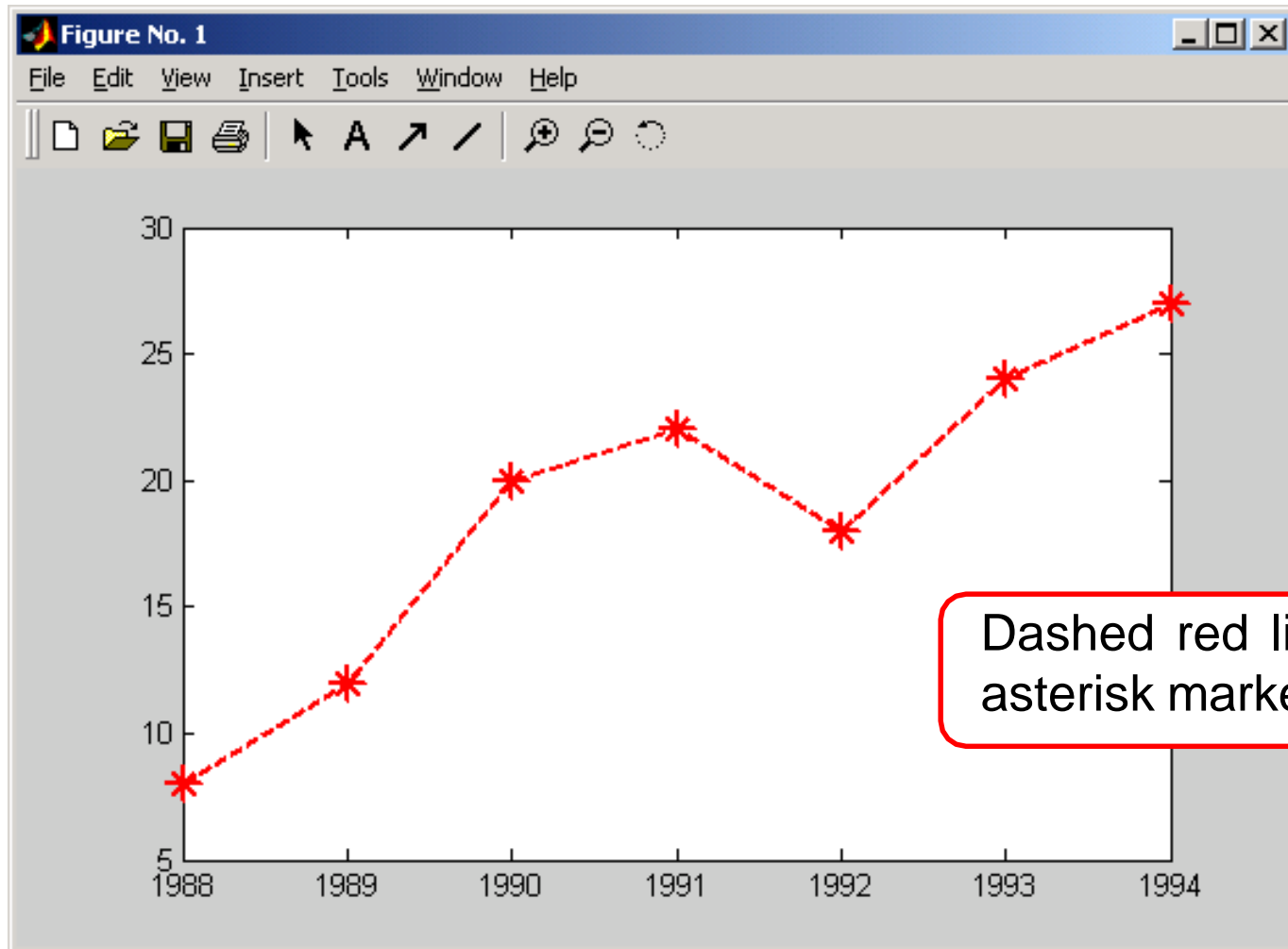
Year	1988	1989	1990	1991	1992	1993	1994
Sales (M)	127	130	136	145	158	178	211

```
>> year = [1988:1:1994];  
>> sales = [127, 130, 136, 145, 158, 178, 211];  
>> plot(year,sales,'--r*')
```



Line Specifiers:
dashed red line and
asterisk markers.

PLOT OF GIVEN DATA USING LINE SPECIFIERS IN THE `plot()` COMMAND



CREATING A PLOT OF A FUNCTION

Consider: $y = 3.5^{-0.5x} \cos(6x)$ for $-2 \leq x \leq 4$

A script file for plotting the function is:

% A script file that creates a plot of

% the function: $3.5^{(-0.5x)} \cos(6x)$

`x = [-2:0.01:4];`



Creating a vector with spacing of 0.01.

`y = 3.5.^(-0.5*x).*cos(6*x);`



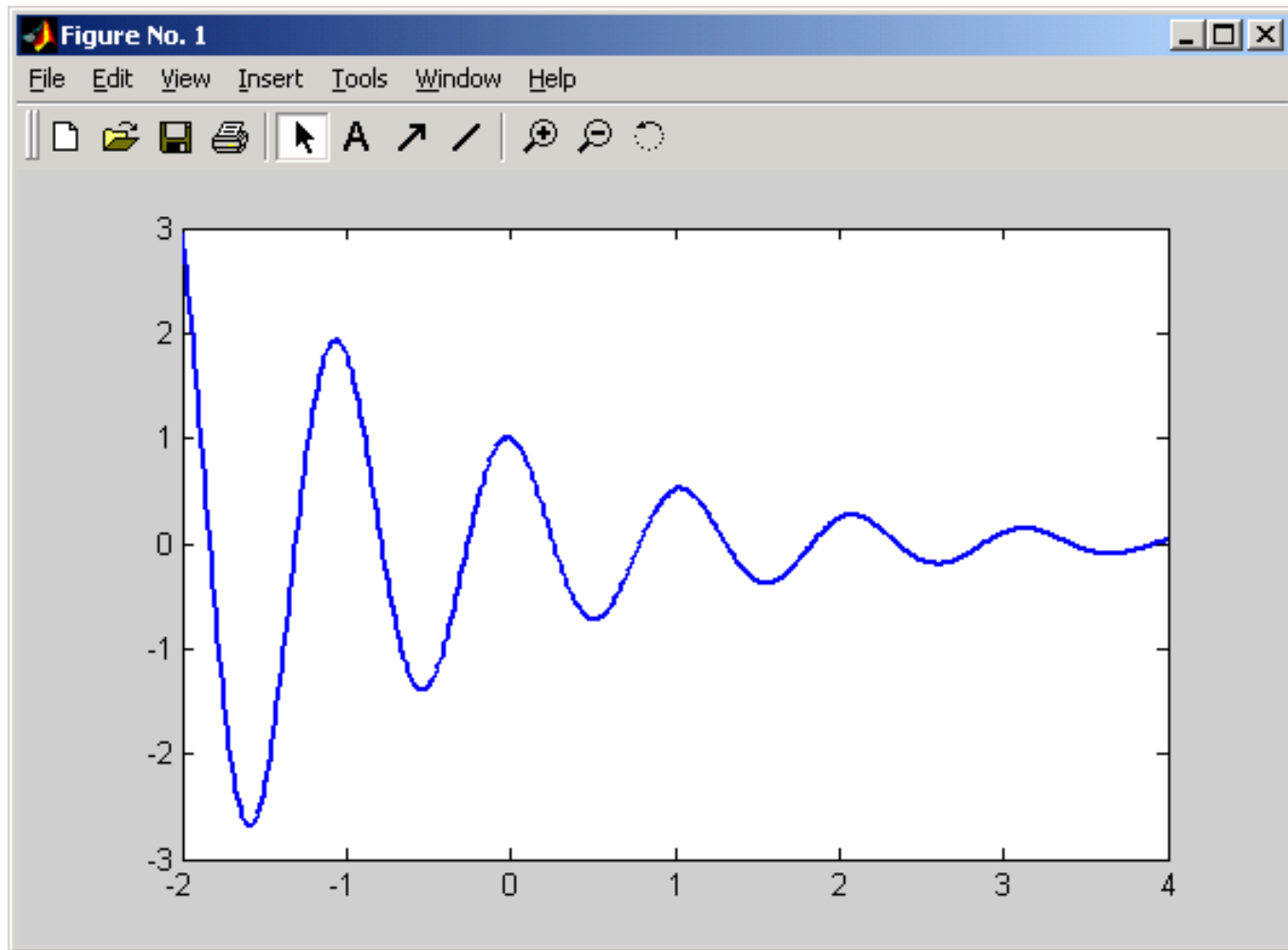
Calculating a value of **y**
for each **x**.

`plot(x,y)`

Once the plot command is executed, the Figure Window opens with the following plot.

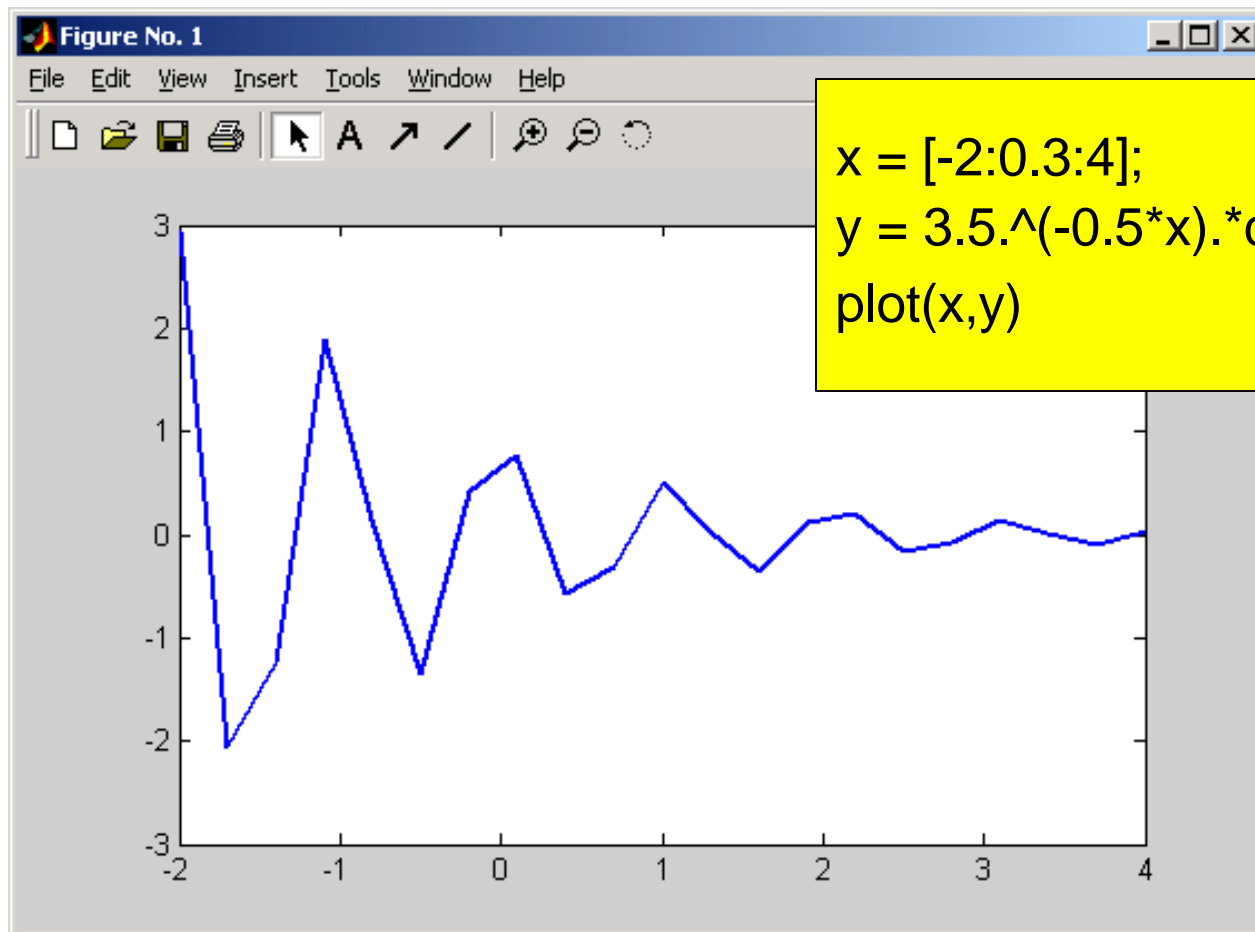
A PLOT OF A FUNCTION

$$y = 3.5^{-0.5x} \cos(6x) \quad \text{for } -2 \leq x \leq 4$$



CREATING A PLOT OF A FUNCTION

If the vector **x** is created with large spacing, the graph is not accurate.
Below is the previous plot with spacing of 0.3.



THE `fplot` COMMAND

The **`fplot`** command can be used to plot a function with the form: $y = f(x)$

`fplot('function', limits)`

- The function is typed in as a string.
- The limits is a vector with the domain of x , and optionally with limits of the y axis:

`[xmin, xmax]`

or

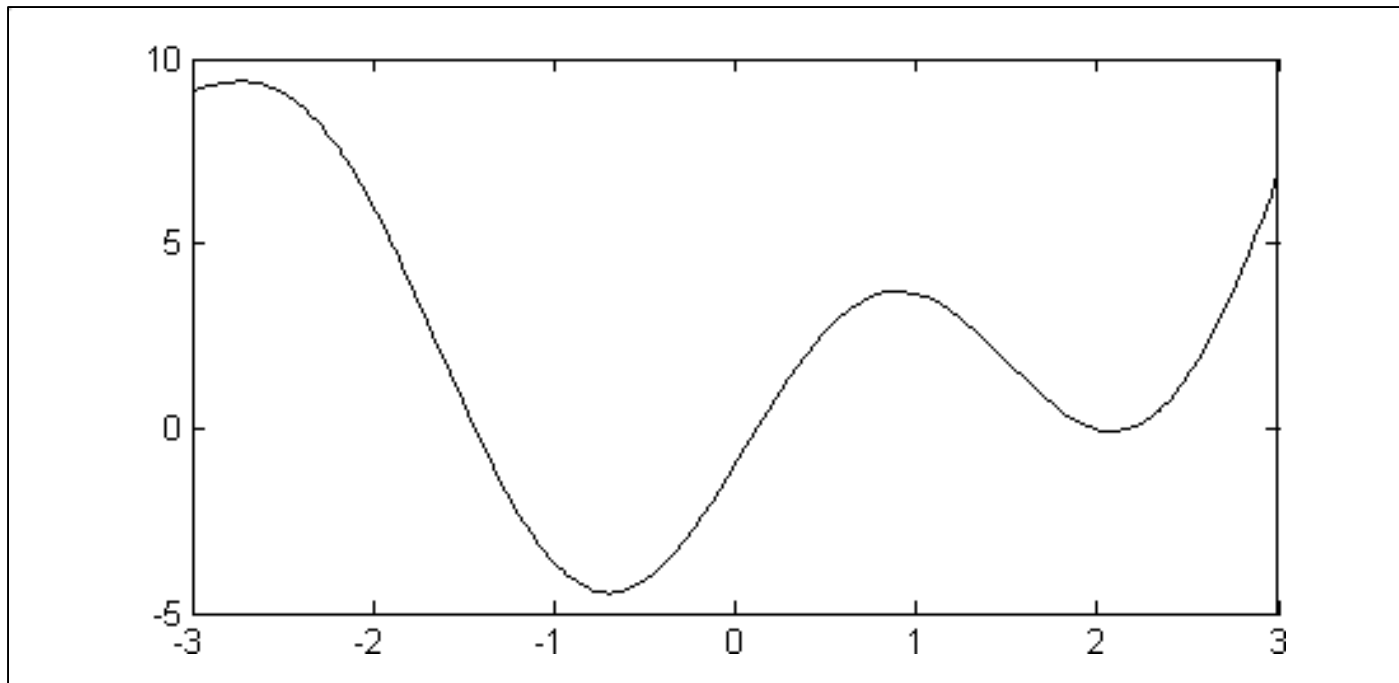
`[xmin, xmax, ymin, ymax]`

- Line specifiers can be added.

PLOT OF A FUNCTION WITH THE `fplot()` COMMAND

A plot of: $y = x^2 + 4\sin(2x) - 1$ for $-3 \leq x \leq 3$

```
>> fplot('x^2 + 4 * sin(2*x) - 1')
```



USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

```
plot(x,y,u,v,t,h)
```

Plots three graphs in the same plot:

y versus **x**, **v** versus **u**, and **h** versus **t**.

- By default, MATLAB makes the curves in different colors.
- Additional curves can be added.
- The curves can have a specific style by adding specifiers after each pair, for example:

```
plot(x,y,'-b',u,v,'-r',t,h,'g:')
```


USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

Plot of the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \leq x \leq 4$, all in the same plot.

```
x = [-2:0.01:4];
```

← vector **x** with the domain of the function.

```
y = 3*x.^3-26*x+6;
```

← Vector **y** with the function value at each **x**.

```
yd = 9*x.^2-26;
```

← Vector **yd** with values of the first derivative.

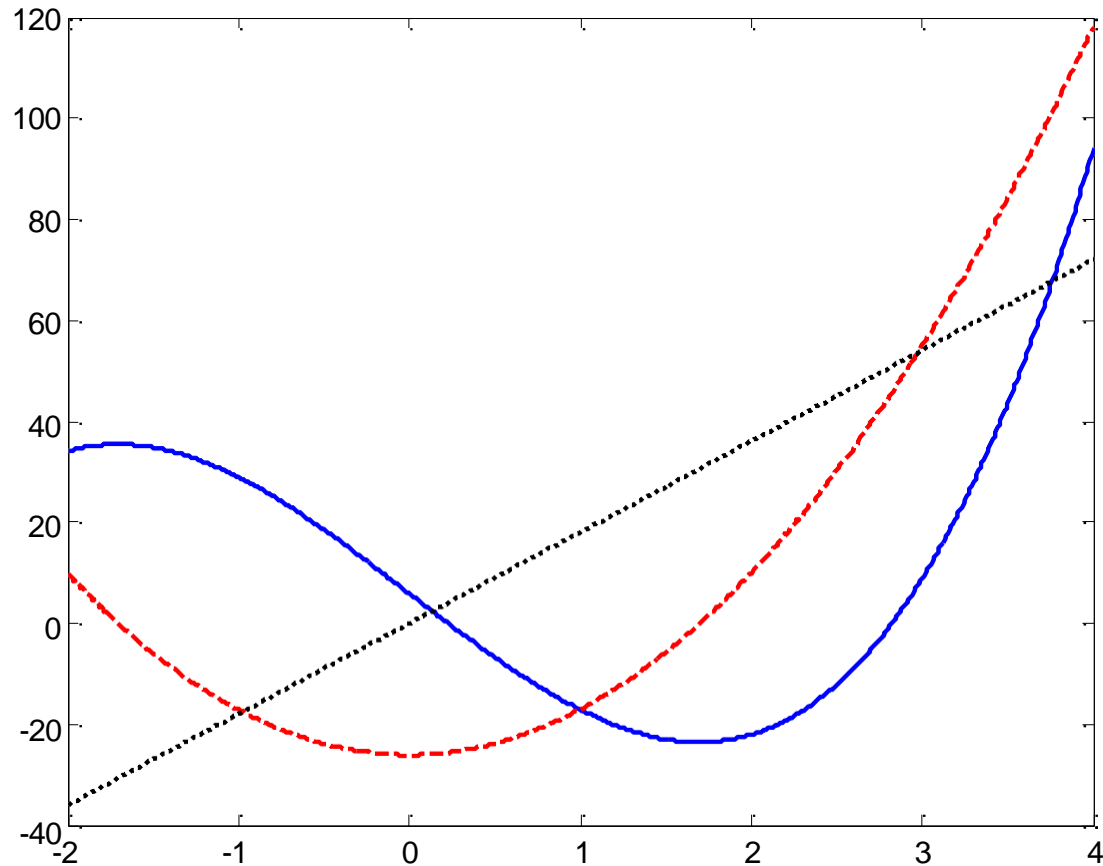
```
ydd = 18*x;
```

← Vector **ydd** with values of the second derivative.

```
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

← Create three graphs, **y** vs. **x** (solid blue line), **yd** vs. **x** (dashed red line), and **ydd** vs. **x** (dotted black line) in the same figure.

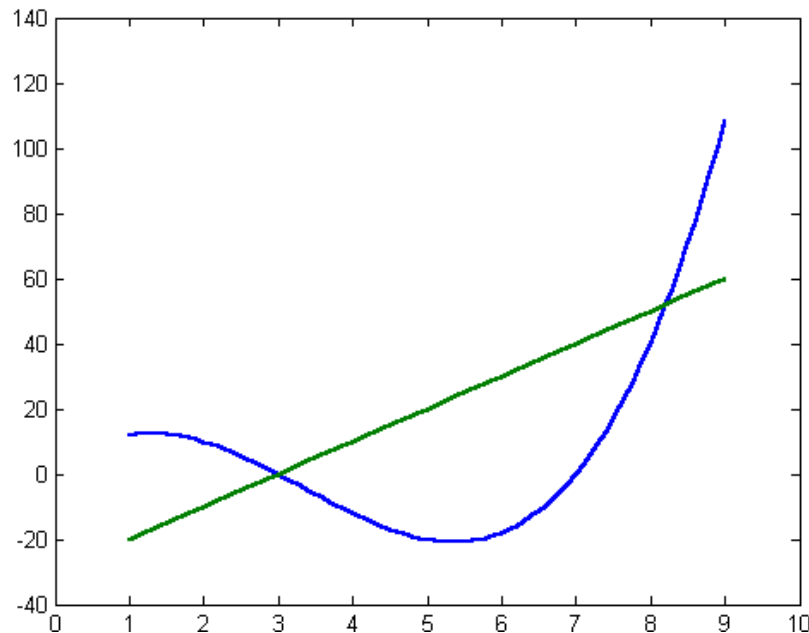
USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT



PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT

Two methods:

1. Using the **plot** command.
2. Using the **hold on**, **hold off** commands.



USING THE `hold on`, `hold off`, COMMANDS TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

`hold on`

Holds the current plot and all axis properties so that subsequent plot commands add to the existing plot.

`hold off`

Returns to the default mode whereby plot commands erase the previous plots and reset all axis properties before drawing new plots.

This method is useful when all the information (vectors) used for the plotting is not available at the same time.

USING THE hold on, hold off, COMMANDS TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

Plot of the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \leq x \leq 4$ all in the same plot.

```
x = [-2:0.01:4];
```

```
y = 3*x.^3-26*x+6;
```

```
yd = 9*x.^2-26;
```

```
ydd = 18*x;
```

```
plot(x,y,'-b')
```

← First graph is created.

```
hold on
```

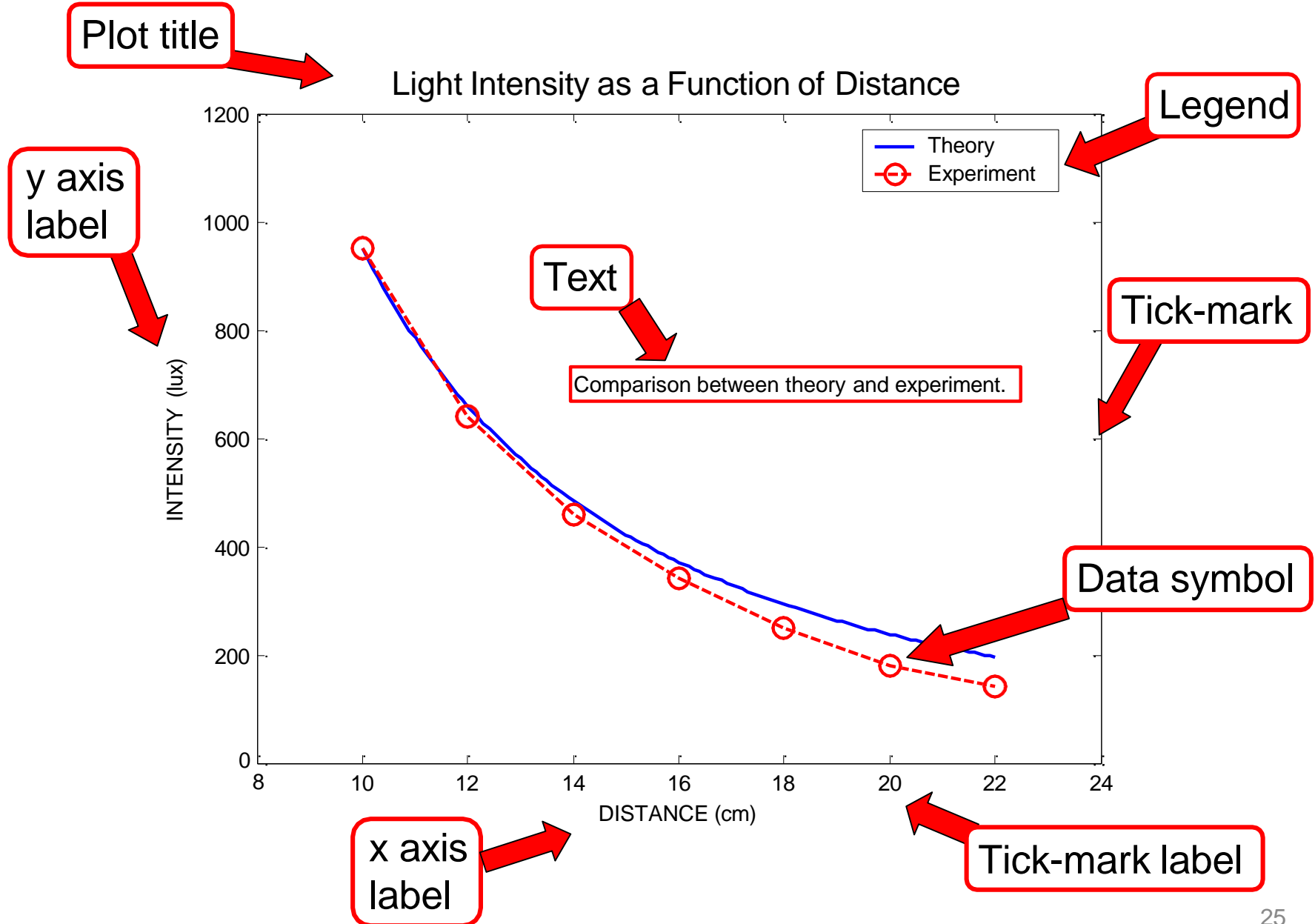
```
plot(x,yd,'--r')
```

```
plot(x,ydd,':k')
```

← Two more graphs are created.

```
hold off
```

EXAMPLE OF A FORMATTED 2-D PLOT



FORMATTING PLOTS

A plot can be formatted to have a required appearance.

With formatting you can:

- Add title to the plot.
- Add labels to axes.
- Change range of the axes.
- Add legend.
- Add text blocks.
- Add grid.

FORMATTING PLOTS

There are two methods to format a plot:

1. Formatting commands.

In this method commands, that make changes or additions to the plot, are entered after the `plot()` command. This can be done in the Command Window, or as part of a program in a script file.

2. Formatting the plot interactively in the Figure Window.

In this method the plot is formatted by clicking on the plot and using the menu to make changes or add details.

FORMATTING COMMANDS

```
title('fuctions')
```

Adds the string as a title at the top of the plot.

```
xlabel('string')
```

Adds the string as a label to the x -axis.

```
ylabel('string')
```

Adds the string as a label to the y -axis.

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

Sets the minimum and maximum limits of the x - and y -axes.

FORMATTING COMMANDS

```
legend( 'string1' , 'string2' , 'string3' )
```

Creates a legend using the strings to label various curves (when several curves are in one plot). The location of the legend is specified by the mouse.

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

Places the string (text) on the plot at coordinate x,y relative to the plot axes.

```
gtext( 'string' )
```

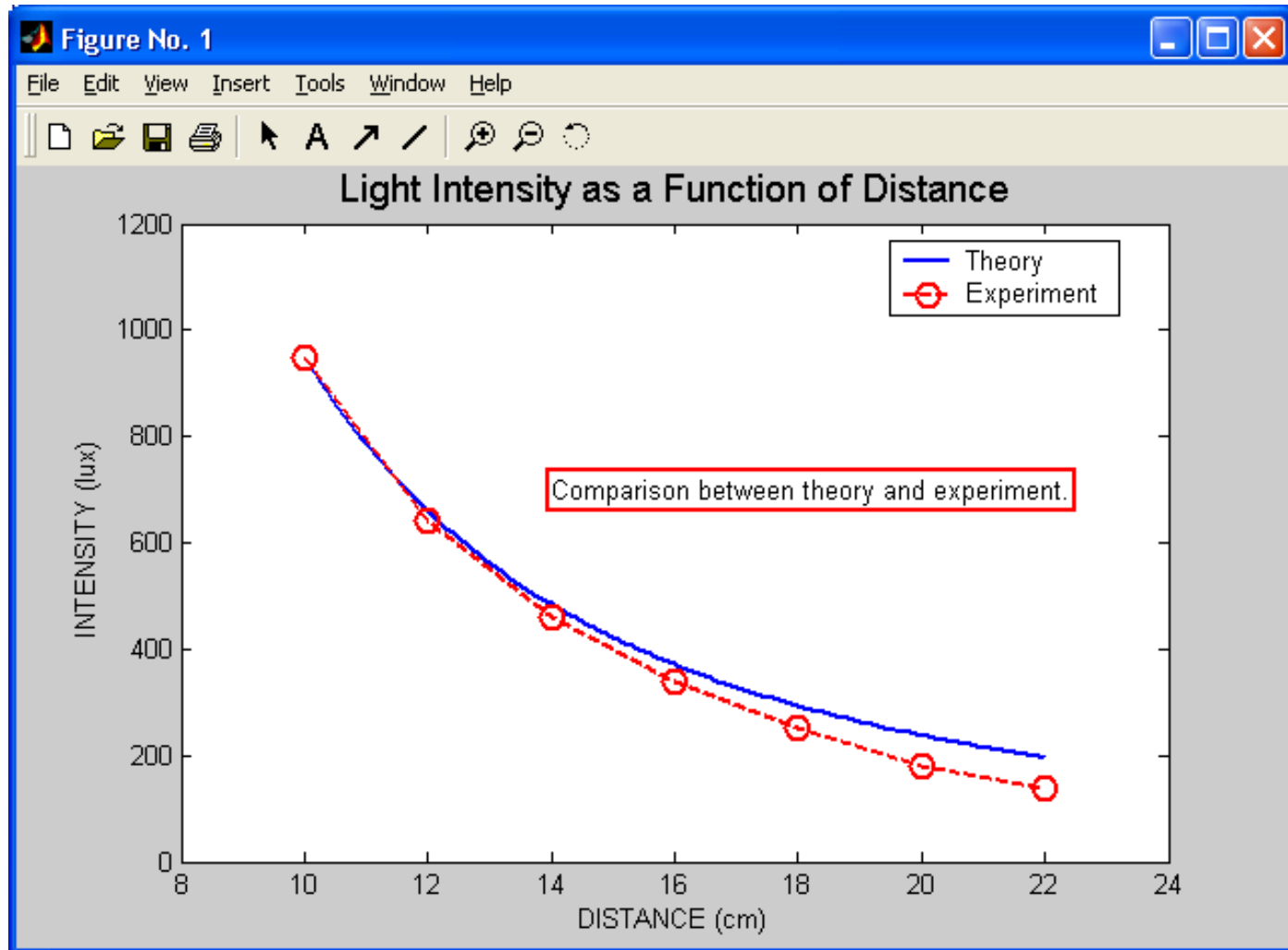
Places the string (text) on the plot. When the command executes the figure window pops and the text location is clicked with the mouse.

FORMATTING Example

```
x = [10: 0.1: 22];  
y = 95000 ./ x .^ 2;  
x_data = [10: 2: 22];  
y_data = [950 640 460 340 250 180 140];  
plot(x, y, '-', x_data, y_data, 'ro--')  
xlabel('DISTANCE (cm)') ← Labels for the axes  
ylabel('INTENSITY (lux)')  
title('Light Intensity as a Function of Distance') ← Title of the plot  
axis( [8 24 0 1200] ) ← Set limits of the axes  
legend('Theory', 'Experiment') ← Create legend
```

The plot that is obtained is shown again in the next slide.

EXAMPLE OF A FORMATTED PLOT



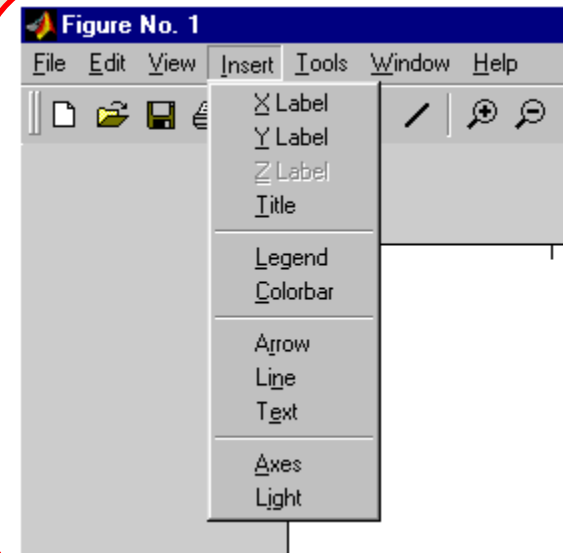
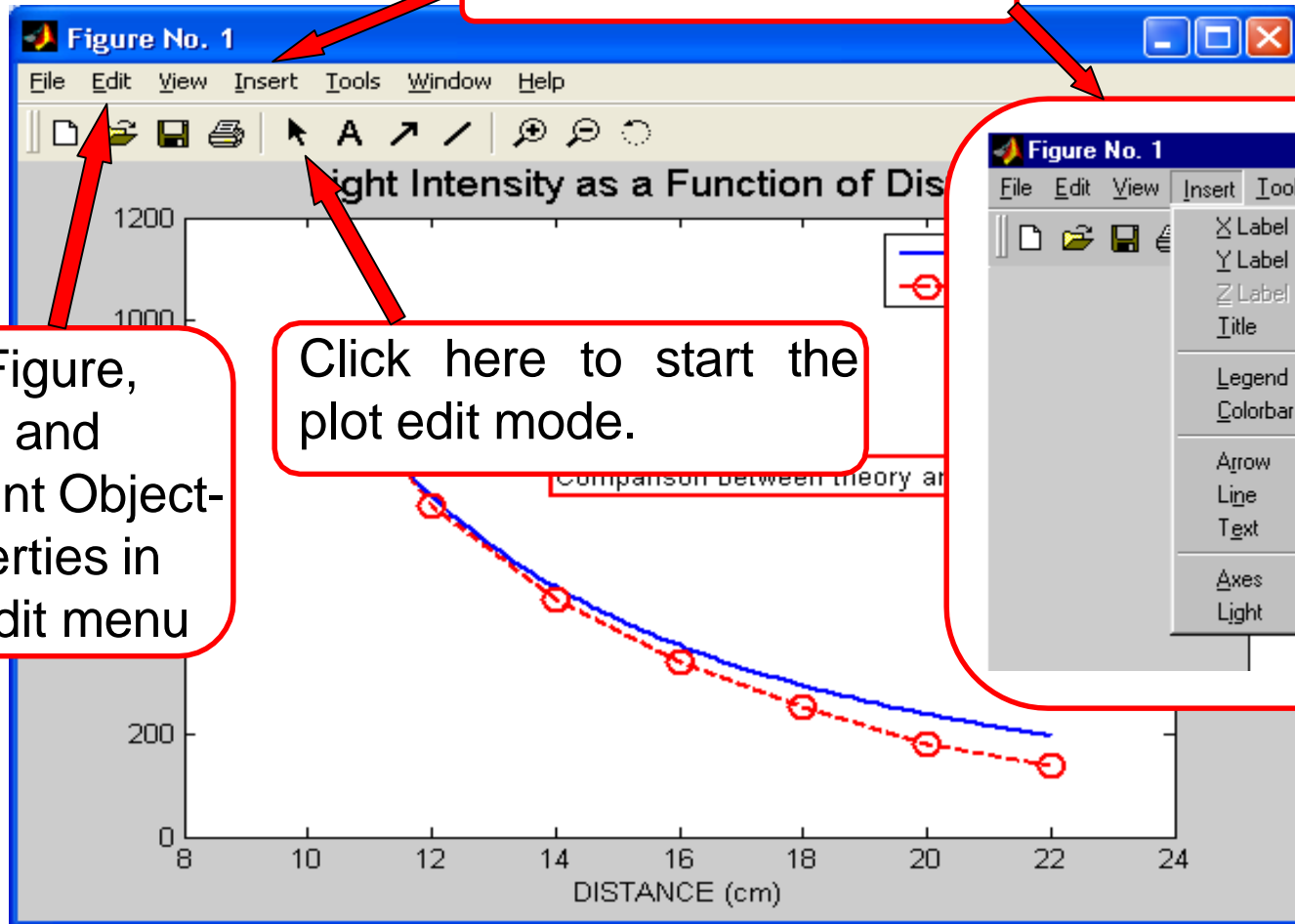
FORMATTING A PLOT IN THE FIGURE WINDOW

Once a figure window is open, the figure can be formatted interactively.

Use the insert menu to

Use Figure, Axes, and Current Object-Properties in the Edit menu

Click here to start the plot edit mode.



PLOTTING MULTIPLE PLOTS ON ONE PAGE

Several plots on one page can be created with the subplot command.

`subplot(m,n,p)` This command creates $m \times n$ plots in the Figure Window. The plots are arranged in m rows and n columns. The variable p defines which plot is active. The plots are numbered from 1 to $m \times n$. The upper left plot is 1 and the lower right plot is $m \times n$. The numbers increase from left to right within a row, from the first row to the last.

PLOTTING MULTIPLE PLOTS ON ONE PAGE

For example, the command:

`subplot(3,2,p)`

Creates 6 plots arranged in 3 rows and 2 columns.

`subplot(3,2,1)`

`subplot(3,2,2)`

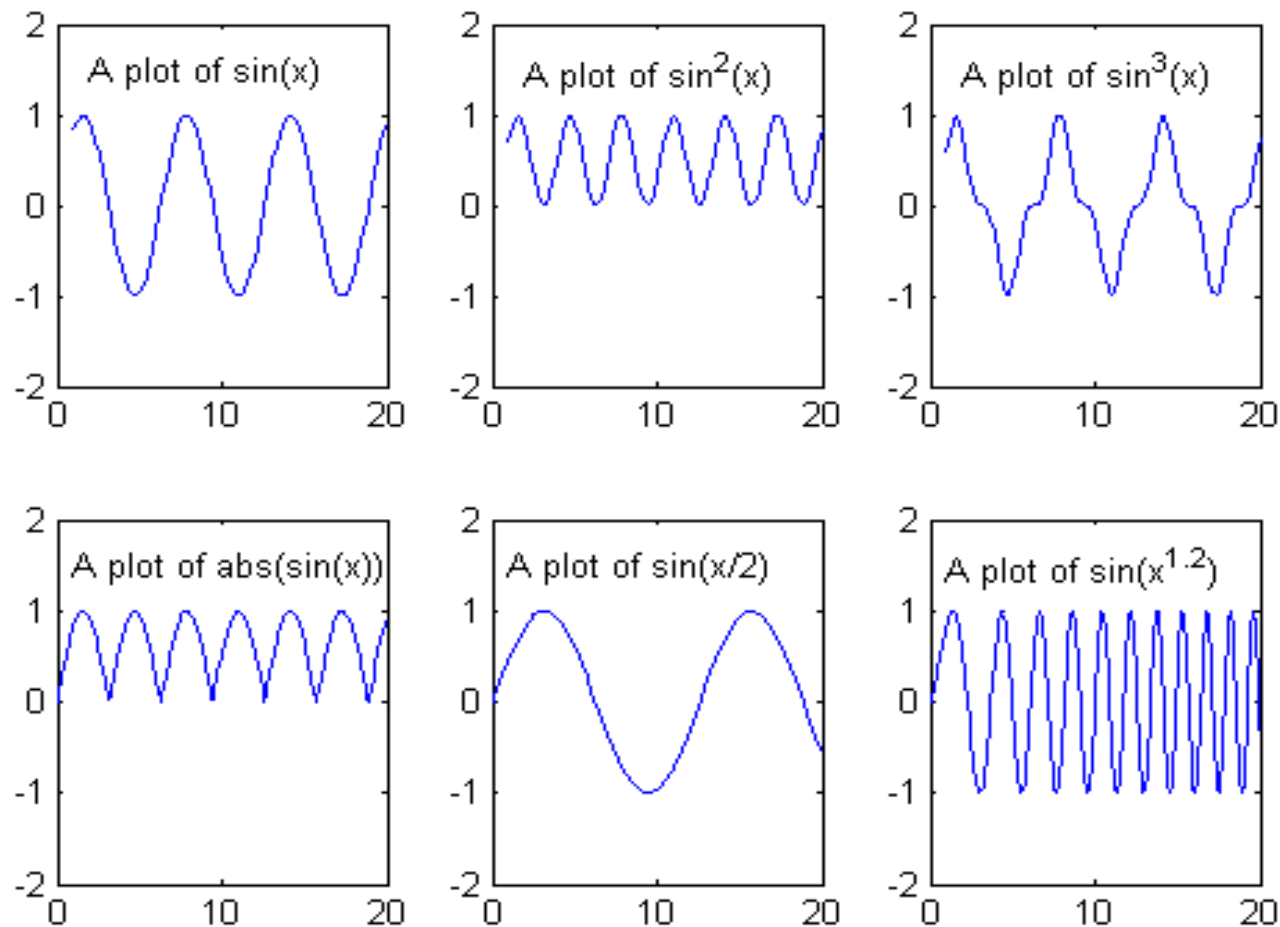
`subplot(3,2,3)`

`subplot(3,2,4)`

`subplot(3,2,5)`

`subplot(3,2,6)`

EXAMPLE OF MULTIPLE PLOTS ON ONE PAGE



The script file of the figure above is shown in the next slide.

SCRIPT FILE OF MULTIPLE PLOTS ON ONE PAGE

```
% Example of using the subplot command.  
x1=20:100;      % Creating a vector x1  
y1=sin(x1);      % Calculating y1  
subplot(2,3,1)   % Creating the first plot  
plot(x1,y1)      % Plotting the first plot  
axis([0 20 -2 2]) % Formatting the first plot  
text(2,1.5,'A plot of sin(x)')  
y2=sin(x1).^2;   % Calculating y2  
subplot(2,3,2)   % Creating the second plot  
plot(x1,y2)      % Plotting the second plot  
axis([0 20 -2 2]) % Formatting the second plot  
text(2,1.5,'A plot of sin^2(x)')  
y3=sin(x1).^3;   % Calculating y2
```

(The file continues on the next slide)

SCRIPT FILE OF MULTIPLE PLOTS ON ONE PAGE

(CONT,)

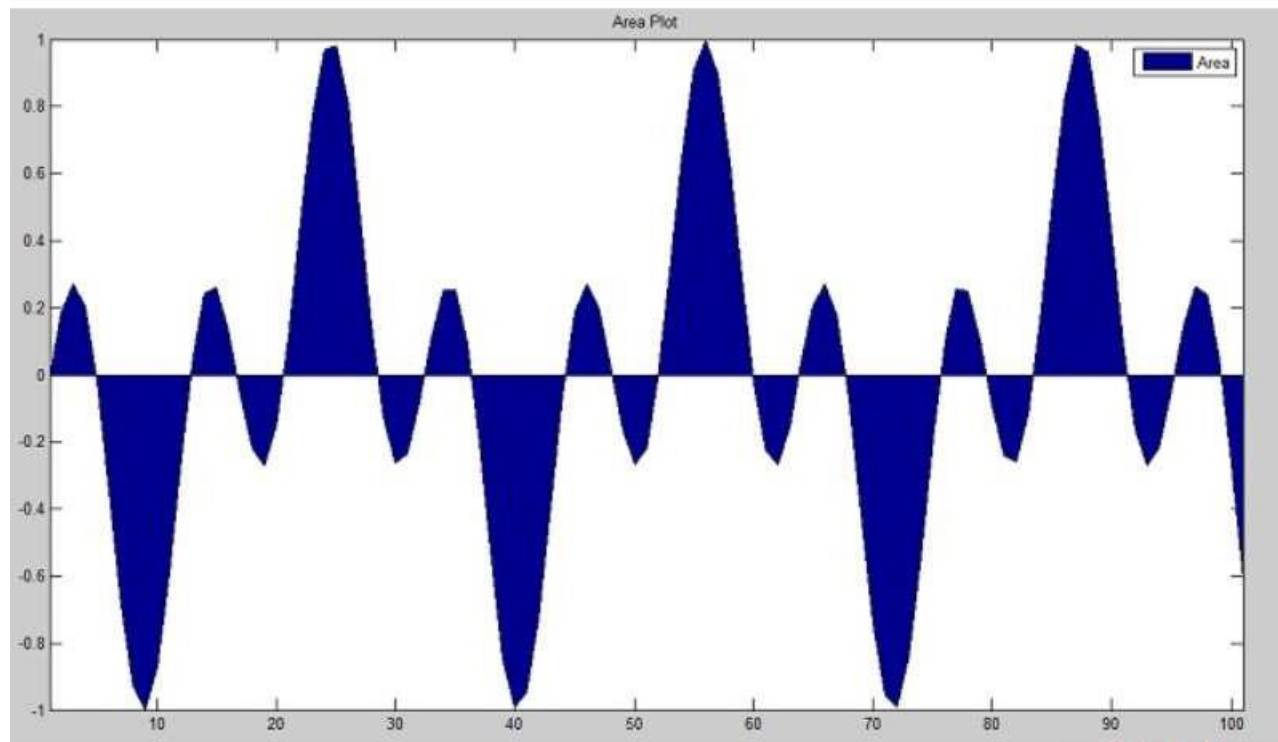
subplot(2,3,3)	% Creating the third plot
plot(x1,y3)	% Plotting the third plot
axis([0 20 -2 2])	% Formatting the third plot
text(2,1.5,'A plot of $\sin^3(x)$ ')	
subplot(2,3,4)	% Creating the fourth plot
fplot('abs(sin(x))',[0 20 -2 2])	% Plotting the fourth plot
text(1,1.5,'A plot of $\text{abs}(\sin(x))$ ')	% Formatting the fourth plot
subplot(2,3,5)	% Creating the fifth plot
fplot('sin(x/2)',[0 20 -2 2])	% Plotting the fifth plot
text(1,1.5,'A plot of $\sin(x/2)$ ')	% Formatting the fifth plot
subplot(2,3,6)	% Creating the sixth plot
fplot('sin(x.^1.4)',[0 20 -2 2])	% Plotting the fifth plot
text(1,1.5,'A plot of $\sin(x^{1.^2})$ ')	% Formatting the sixth plot

Area Plot

- In the Area plotting graph, you can use basic functions. It is a very easy draw.
- In the MATLAB plotting, there is a function `area()` to plot Area.

% To create the area plot for the given equation $\sin(t)\cos(2t)$. %
Enter the value of range of variable 't'.

```
t=[0:0.2:20];  
a=[sin(t).*cos(2.*t)];  
area(a)  
title('Area Plot')
```



Stem Plot

- In Stem plot, the discrete sequence data and variables are used. This plot is created by using the stem() function.

% Consider the variable range of 'x' and 'y',

```
x=[3 1 6 7 10 9 11 13 15 17];
```

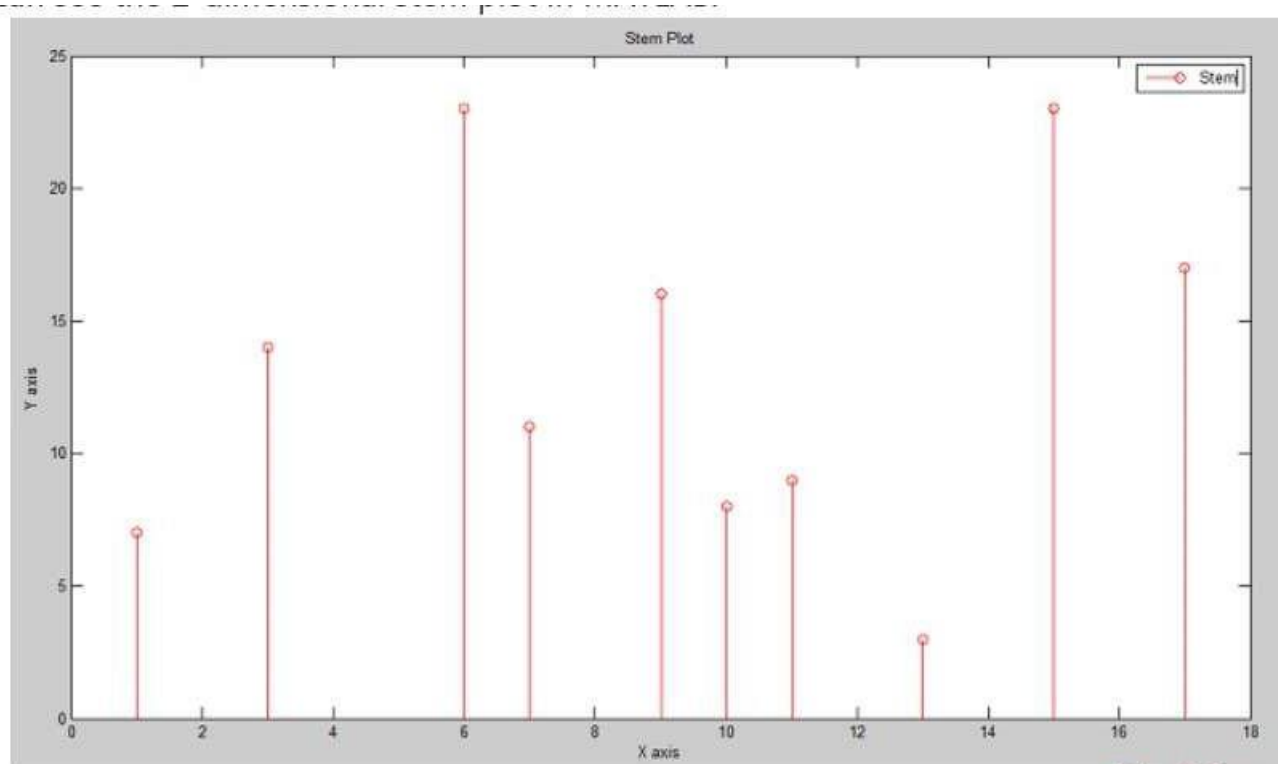
```
y=[14 7 23 11 8 16 9 3 23 17];
```

```
stem(x,y,'r')
```

```
title('Stem Plot')
```

```
xlabel('X axis')
```

```
ylabel('Y axis')
```



Bar Plot

- A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.

```
x=[1 3 5 7 10 13 15];
```

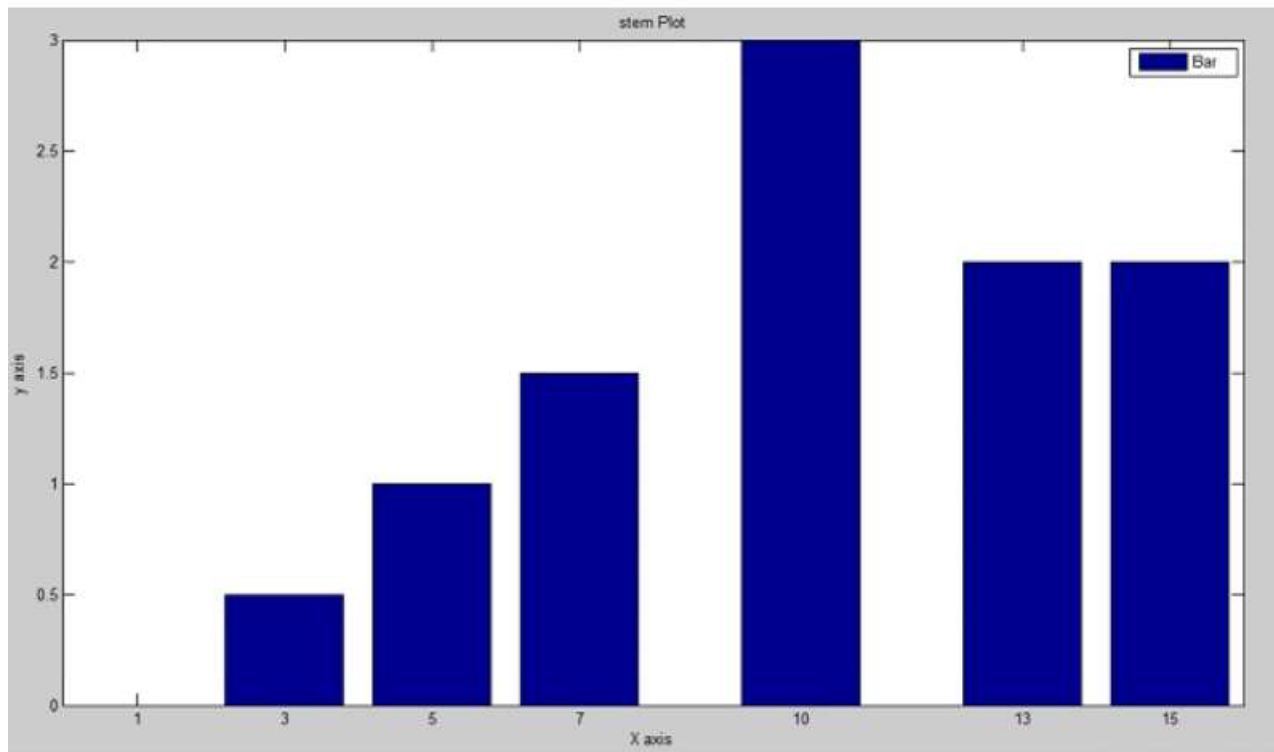
```
y=[0 0.5 1 1.5 3 2 2];
```

```
bar(x,y)
```

```
title('Bar Plot')
```

```
xlabel('X axis')
```

```
ylabel('y axis')
```



Barh Plot

- Barh plot is short abbreviations of Horizontal bar. Here I am using the Barh function for the horizontal plane.

```
x=[1 3 5 7 10 13 15];
```

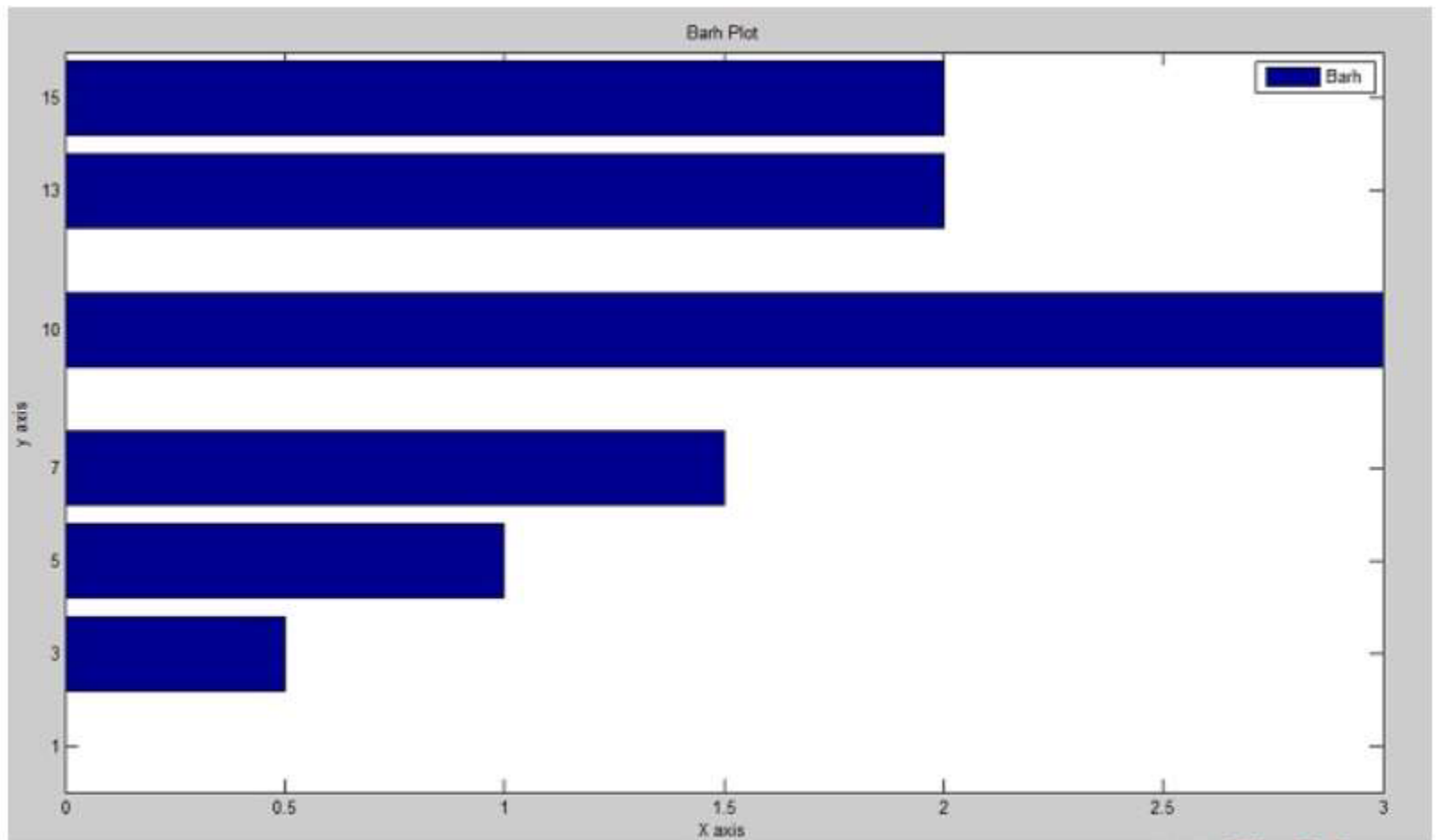
```
y=[0 0.5 1 1.5 3 2 2];
```

```
barh(x,y)
```

```
title('Barh Plot')
```

```
xlabel('X axis')
```

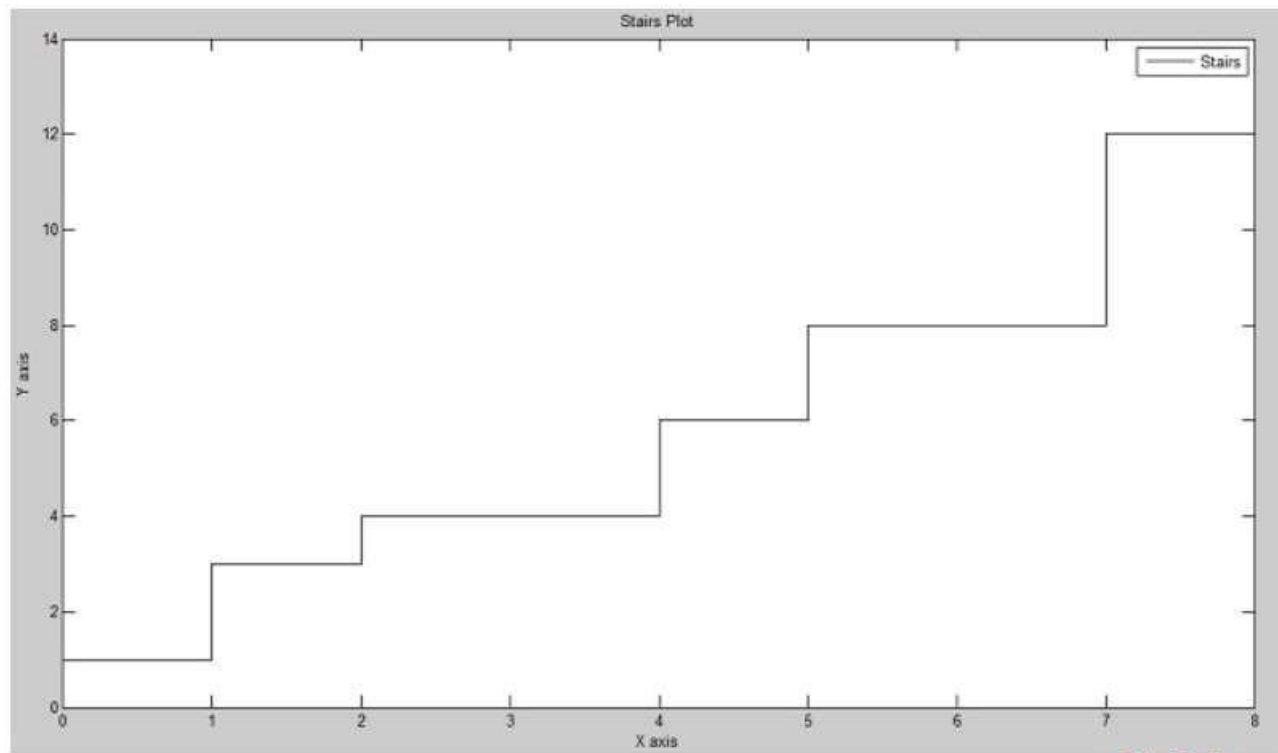
```
ylabel('y axis')
```



Stairs Plot

- This is again one of the MATLAB 2D plots that look more like stairs.

```
x=[0 1 2 4 5 7 8];  
y=[1 3 4 6 8 12 13];  
stairs(x,y)  
title('Stairs Plot')  
xlabel('X axis')  
ylabel('Y axis')
```



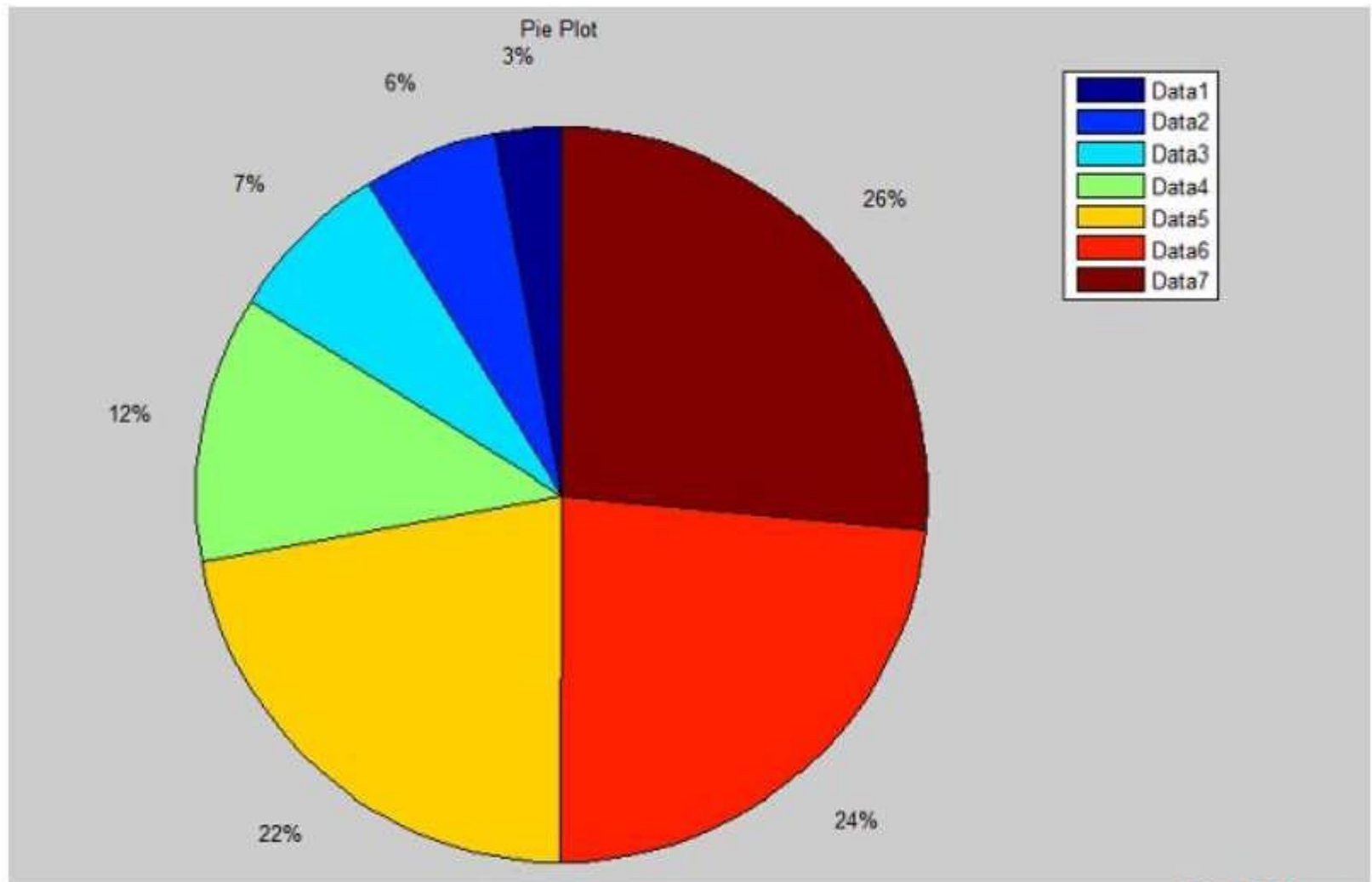
Pie Plot

- In mathematics, the pie chart is used to indicate data in percentage (%) form.

```
x=[10 20 25 40 75 80 90];
```

```
pie(x)
```

```
title('Pie Plot')
```



Scatter Plot

- A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

```
x=[1 2 3 5 7 9 11 13 15];  
y=[1.2 3 4 2.5 3 5.5 4 6 7];  
scatter(x,y,'g')  
title('Scatter Plot')  
xlabel('X axis')  
ylabel('Y axis')
```

Solving Basic Algebraic Equations in MATLAB

- The solve function is used for solving algebraic equations. In its simplest form, the solve function takes the equation enclosed in quotes as an argument.
- For example, let us solve for x in the equation $x-5 = 0$

`solve('x-5=0')` or
`y = solve('x-5 = 0')` or
`solve('x-5')`

Solving Quadratic Equations in MATLAB

- The solve function can also solve higher order equations. It is often used to solve quadratic equations. The function returns the roots of the equation in an array.
- The following example solves the quadratic equation $x^2 - 7x + 12 = 0$.

```
eq = 'x^2 -7*x + 12 = 0';  
s = solve(eq);  
disp('The first root is: ');  
disp(s(1));  
disp('The second root is: '),  
disp(s(2));
```

Solving System of Equations in MATLAB

- The **solve** function can also be used to generate solutions of systems of equations involving more than one variables.

- Let us solve the equations – $5x + 9y = 5$

$$3x - 6y = 4$$

- Create a script file and type the following code –

```
s = solve('5*x + 9*y = 5','3*x - 6*y = 4'); s.x
```

```
s.y
```

When you run the file, it displays the following result – ans = 22/19

```
ans = -5/57
```

Polynomials

- MATLAB represents polynomials as row vectors containing coefficients ordered by descending powers. For example, the equation $P(x) = x^4 + 7x^3 - 5x + 9$ could be represented as –
 $p = [1 \ 7 \ 0 \ -5 \ 9]$

Evaluating Polynomials

- The **polyval** function is used for evaluating a polynomial at a specified value. For example, to evaluate our previous polynomial **p**, at $x = 25$, type –
 $p = [1 \ 7 \ 0 \ -5 \ 9];$
 $\text{polyval}(p, 25)$

- MATLAB also provides the **polyvalm** function for evaluating a matrix polynomial. A matrix polynomial is a **polynomial** with matrices as variables.
- For example, let us create a square matrix X and evaluate the polynomial p

```
p = [1 7 0 -5 9];
```

```
X = [1 2 -3 4; 2 -5 6 3; 3 1 0 2; 5 -7 3 8];
```

```
polyvalm(p, X)
```

Finding the Roots of Polynomials

- The **roots** function calculates the roots of a polynomial. For example, to calculate the roots of our polynomial `p`, type –

```
p = [1 7 0 -5 9];
```

```
r = roots(p)
```

MATLAB - Transforms

- MATLAB provides command for working with transforms, such as the Laplace and Fourier transforms.
- Transforms are used in science and engineering as a tool for simplifying analysis and look at data from another angle.
- The Fourier transform allows us to convert a signal represented as a function of time to a function of frequency. Laplace transform allows us to convert a differential equation to an algebraic equation.

laplace, fourier and fft

- In this example, we will compute the Laplace transform of some commonly used functions.
- Create a script file and type the following code

```
syms s t a b w
```

```
laplace(a)
```

```
laplace(t^2)
```

```
laplace(t^9)
```

```
laplace(exp(-b*t))
```

```
laplace(sin(w*t))
```

```
laplace(cos(w*t))
```


The Inverse Laplace Transform

- MATLAB allows us to compute the inverse Laplace transform using the command `ilaplace`.
- For example,

`ilaplace(1/s^3)`

Fourier Transform

- Fourier transforms commonly transforms a mathematical function of time, $f(t)$, into a new function, sometimes denoted by F , whose argument is frequency with units of cycles/s (hertz) or radians per second. The new function is then known as the Fourier transform and/or the frequency spectrum of the function f .

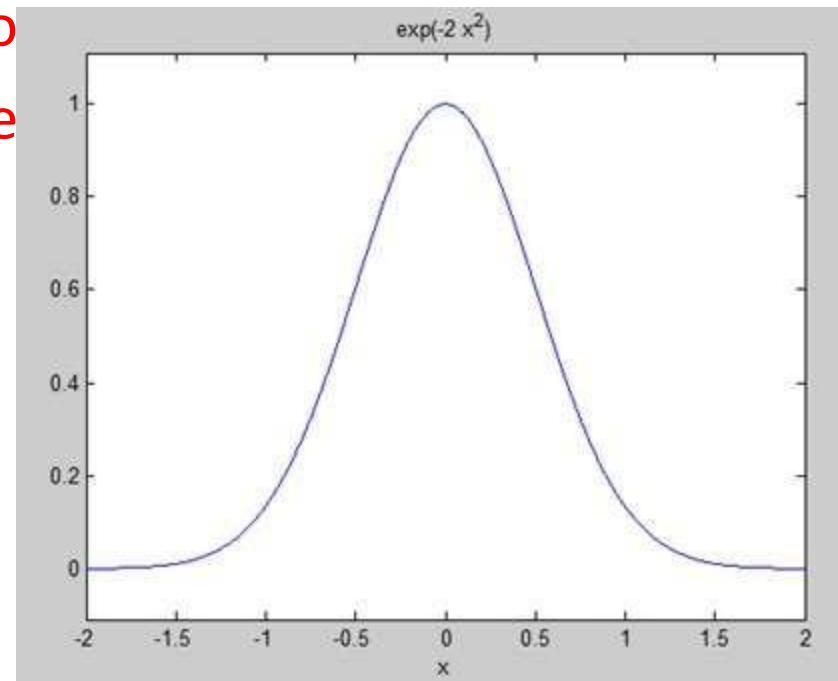
- create a script file and type the following code in it –

`syms x`

`f = exp(-2*x^2); %our function`

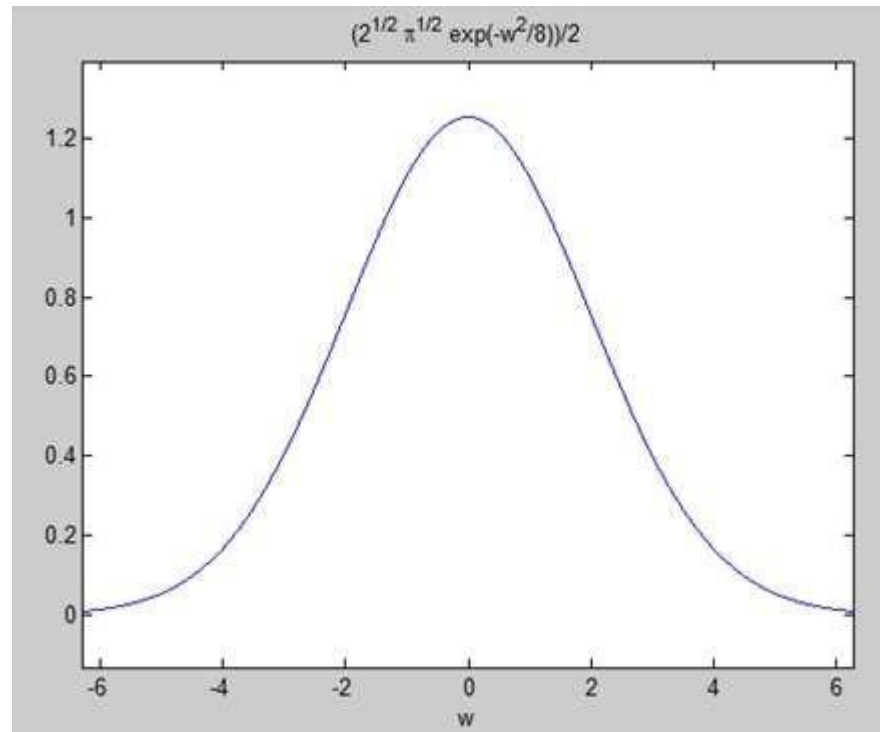
`ezplot(f,[-2,2]) % plot of our function`

`frt= fourier(exp(-2*x^2)) % Fourier`



- The following result is displayed –
$$\text{ftr} = (2^{1/2} \pi^{1/2} \exp(-w^2/8))/2$$

`ezplot(ftr)`



Inverse Fourier Transforms

- MATLAB provides the `ifourier` command for computing the inverse Fourier transform of a function. For example,

$$f = \text{ifourier}(-2 * \exp(-\text{abs}(w)))$$

- MATLAB will execute the above statement and display the result –

$$f = -2/(\pi * (x^2 + 1))$$

MATLAB - Differential

- MATLAB provides the **diff** command for computing symbolic derivatives. In its simplest form, you pass the function you want to differentiate to diff command as an argument.
- compute the derivative of the function $f(t) = 3t^2 + 2t^{-2}$

```
syms t
```

```
f = 3*t^2 + 2*t^(-2);
```

```
diff(f)
```

```

syms x
syms t
f = (x + 2)*(x^2 + 3)
der1 = diff(f)
f = (t^2 + 3)*(sqrt(t) + t^3)
der2 = diff(f)
f = (x^2 - 2*x + 1)*(3*x^3 - 5*x^2 + 2)
der3 = diff(f)
f = (2*x^2 + 3*x)/(x^3 + 1)
der4 = diff(f)
f = (x^2 + 1)^17
der5 = diff(f)
f = (t^3 + 3*t^2 + 5*t - 9)^(-6)
der6 = diff(f)

```

Computing Higher Order Derivatives

- To compute higher derivatives of a function f , we use the syntax **diff(f,n)**.
- Let us compute the second derivative of the function $y = f(x) = x \cdot e^{-3x}$

```
f = x*exp(-3*x);  
diff(f, 2)
```


- let us solve a problem. Given that a function $y = f(x) = 3 \sin(x) + 7 \cos(5x)$. We will have to find out whether the equation $f'' + f = -5\cos(2x)$ holds true.

```
syms x
```

```
y = 3*sin(x)+7*cos(5*x);    % defining the function
```

```
lhs = diff(y,2)+y; %evaluting the lhs of the equation
```

```
rhs = -5*cos(2*x);    %rhs of the equation
```

```
if(isequal(lhs,rhs))
```

```
    disp('Yes, the equation holds true');
```

```
    else disp('No, the equation does not hold true');
```

```
End
```

```
    disp('Value of LHS is: ');
```

```
    disp(lhs);
```

Might be helpful for the Assignment

```
s1 = 'Good'; s2 = 'morning';
```

```
s = strcat(s1,s2)
```

Colors	Numbers
	Black= 0
	Brown= 1
	Red= 2
	Orange= 3
	Yellow= 4
	Green= 5
	Blue= 6
	Violet= 7
	Grey= 8
	White= 9
	Gold= 5%
	Silver= 10%