CS 100

Two Dimensional Plots

Topics Covered:

1. Plotting basic 2-D plots.

   The **`plot`** command.

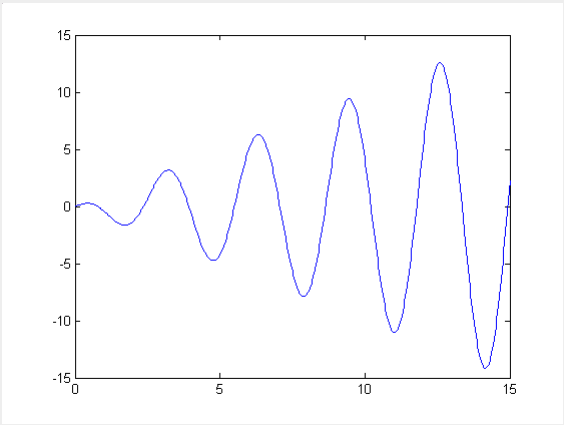
   The **`fplot`** command.

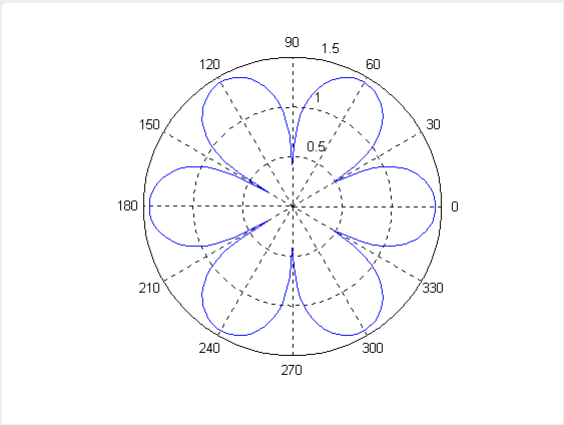
   Plotting multiple graphs in the same plot.

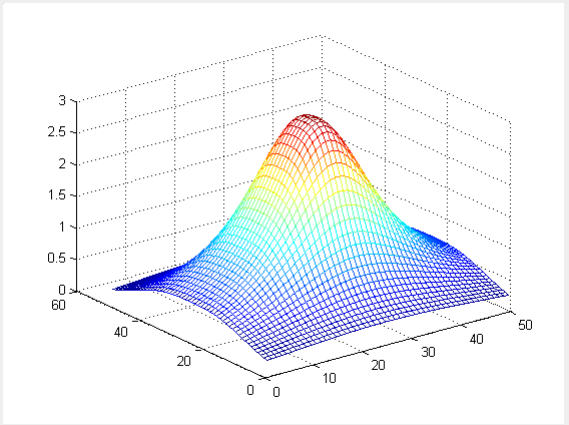
   Formatting plots.

MAKING X-Y PLOTS

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

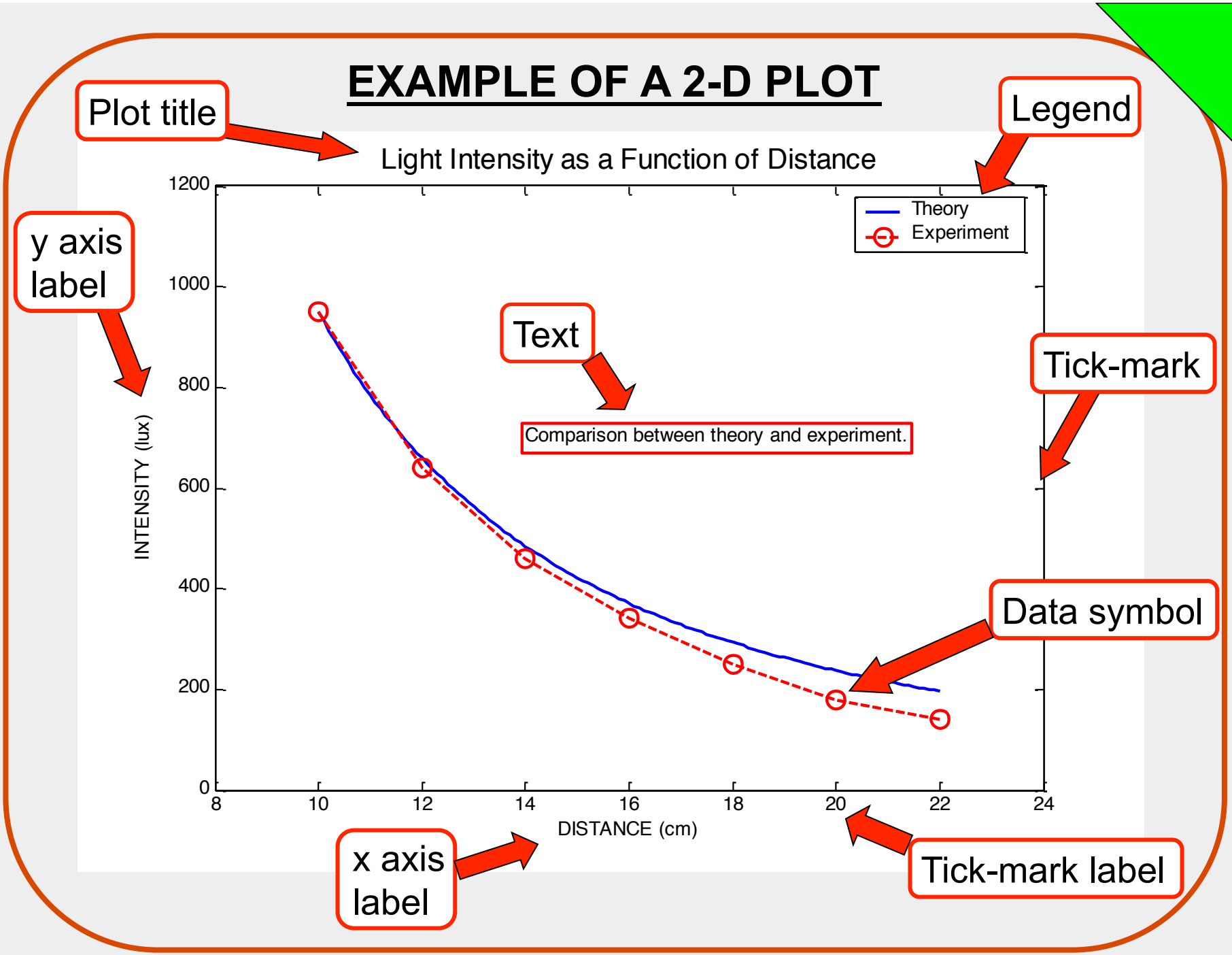






In our class we will only create two dimensional x – y plots.

EXAMPLE OF A 2-D PLOT



TWO-DIMENSIONAL `plot()` COMMAND

The basic 2-D plot command is:

```
plot(x,y)
```

where **x** is a vector (one dimensional array), and **y** is a vector. Both vectors **must** have the same number of elements.

- The plot command creates a single curve with the **x** values on the abscissa (horizontal axis) and the **y** values on the ordinate (vertical axis).
- The curve is made from segments of lines that connect the points that are defined by the **x** and **y** coordinates of the elements in the two vectors.

CREATING THE X AND Y VECTORS

- If data is given, the information is entered as the elements of the vectors **x** and **y**.
- If the values of **y** are determined by a function from the values of **x**, than a vector **x** is created first, and then the values of **y** are calculated for each value of **x**. The spacing (difference) between the elements of **x** must be such that the plotted curve will show the details of the function.

PLOT OF GIVEN DATA

Given data:

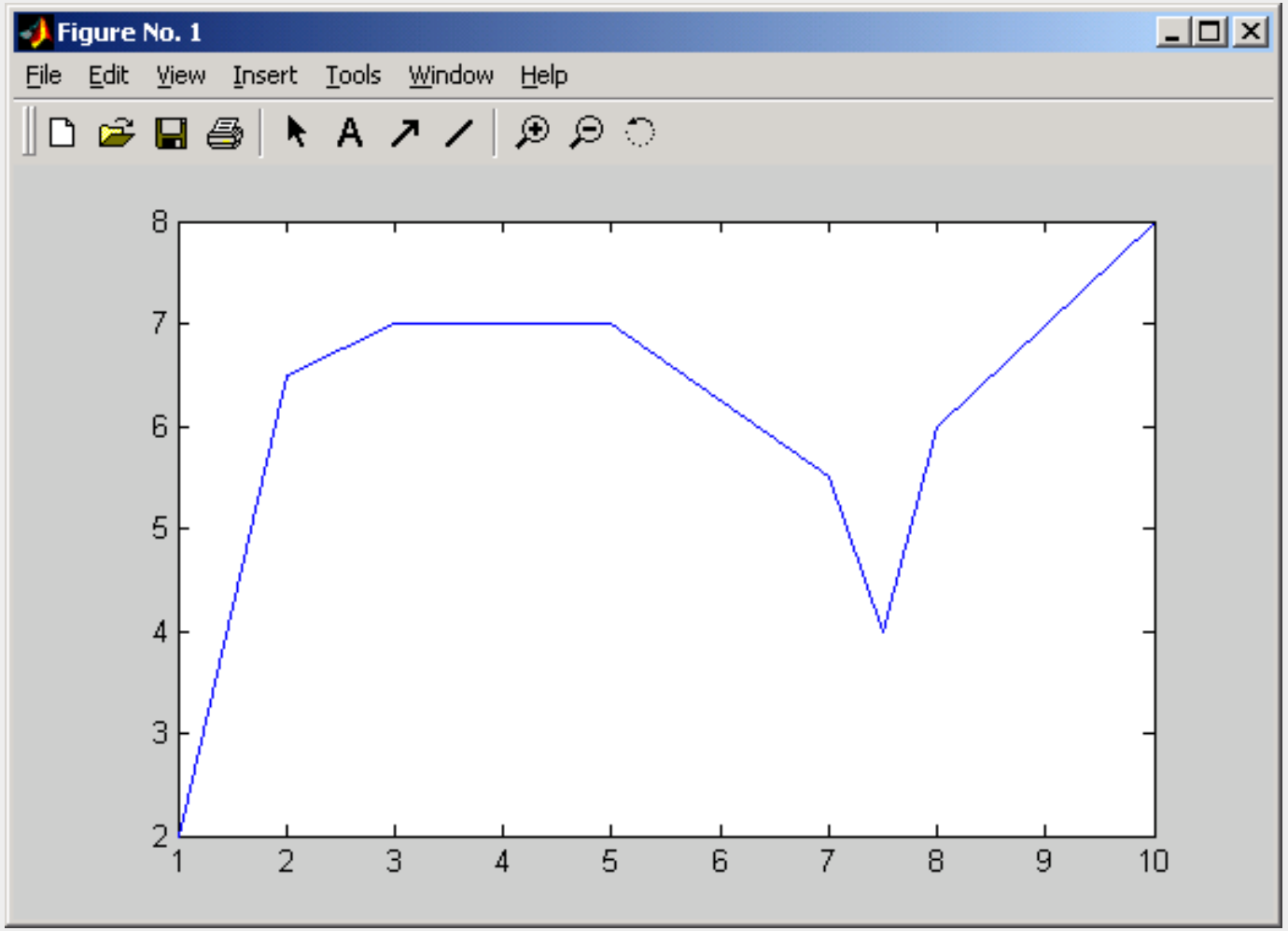
x	1	2	3	5	7	7.5	8	10
y	2	6.5	7	7	5.5	4	6	8

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.

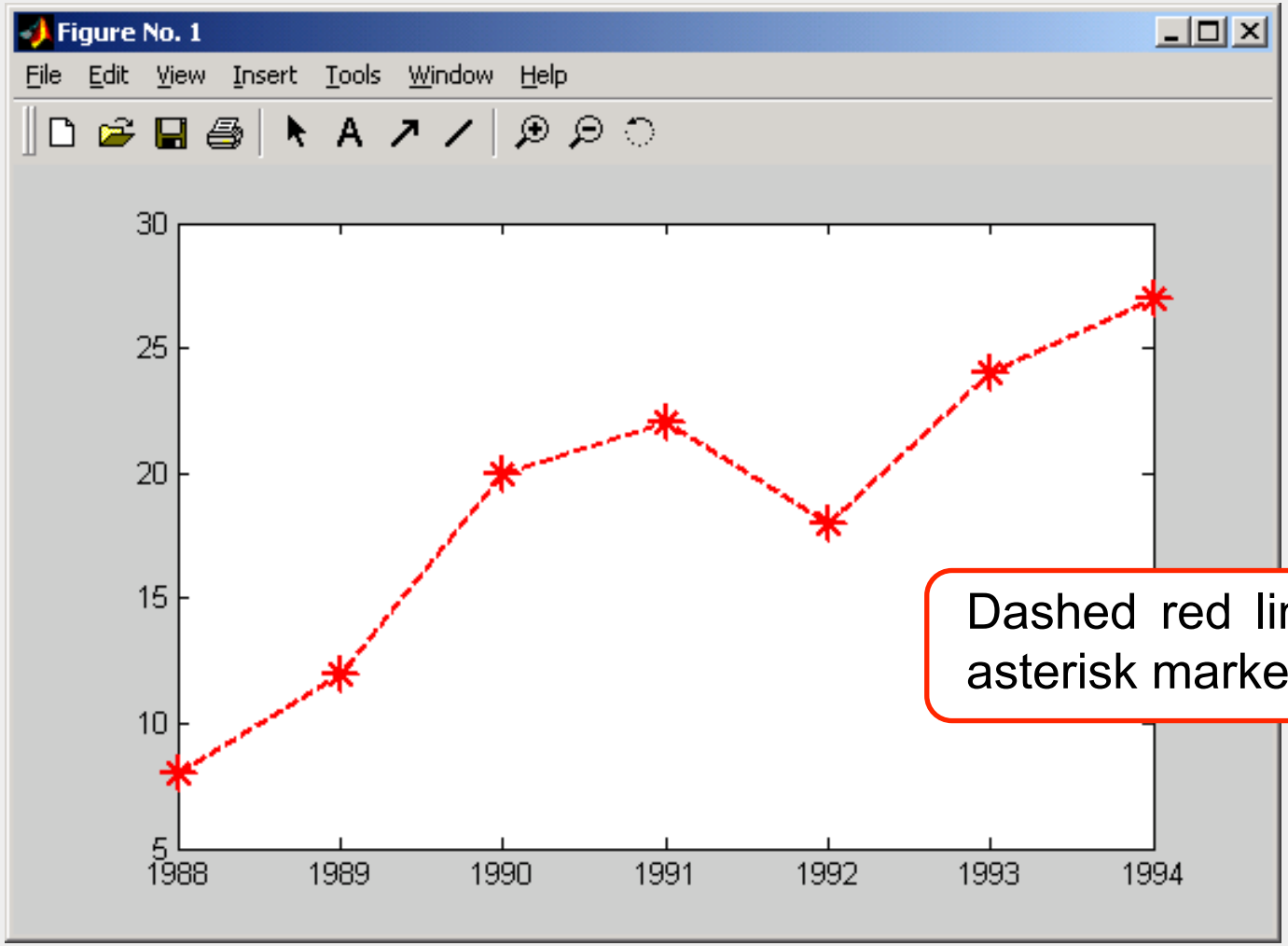
PLOT OF GIVEN DATA USING LINE SPECIFIERS IN THE `plot()` COMMAND

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 $-2 \le x \le 4$

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

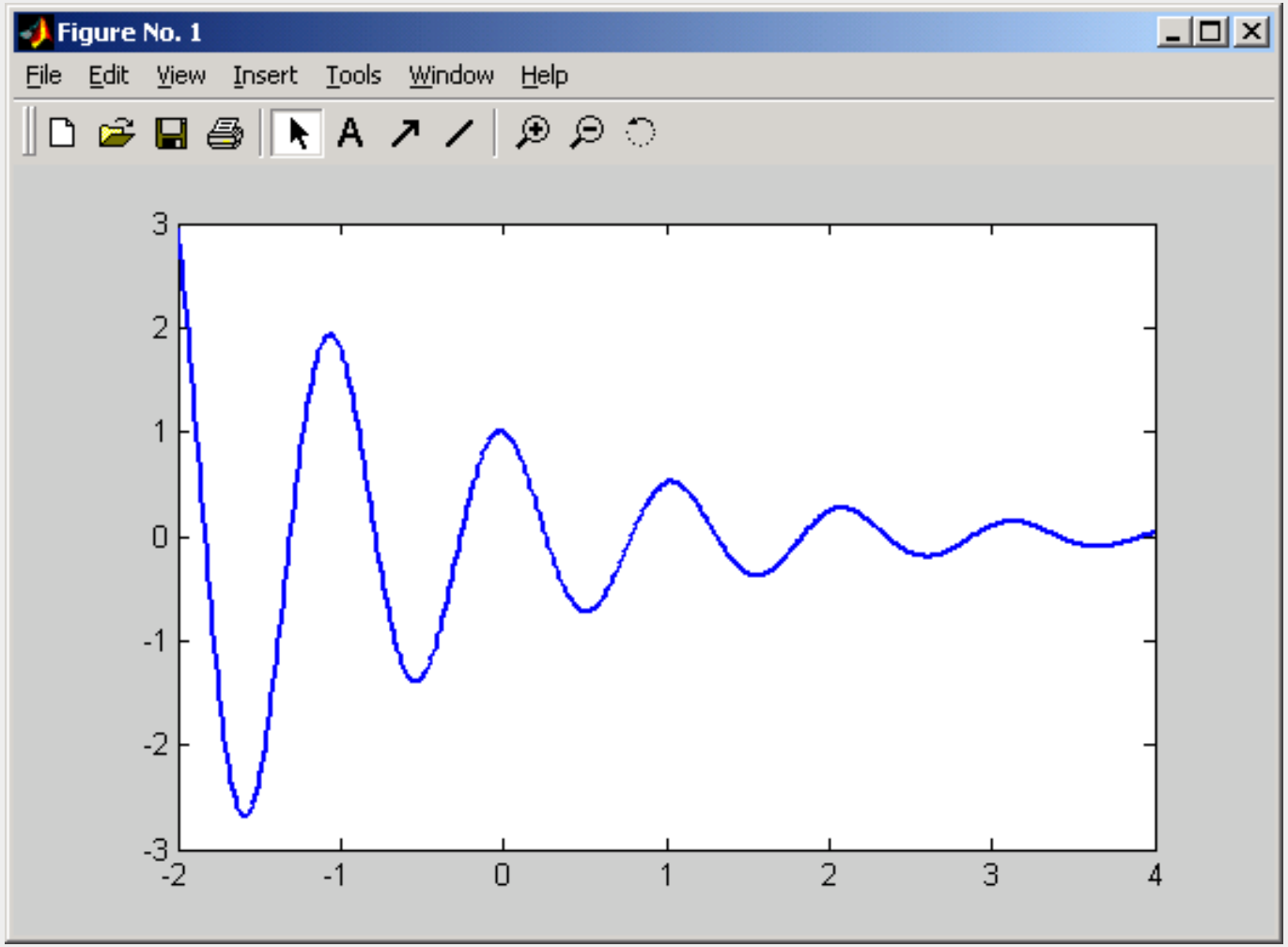
Creating a vector with spacing of 0.01.

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

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

A PLOT OF A FUNCTION

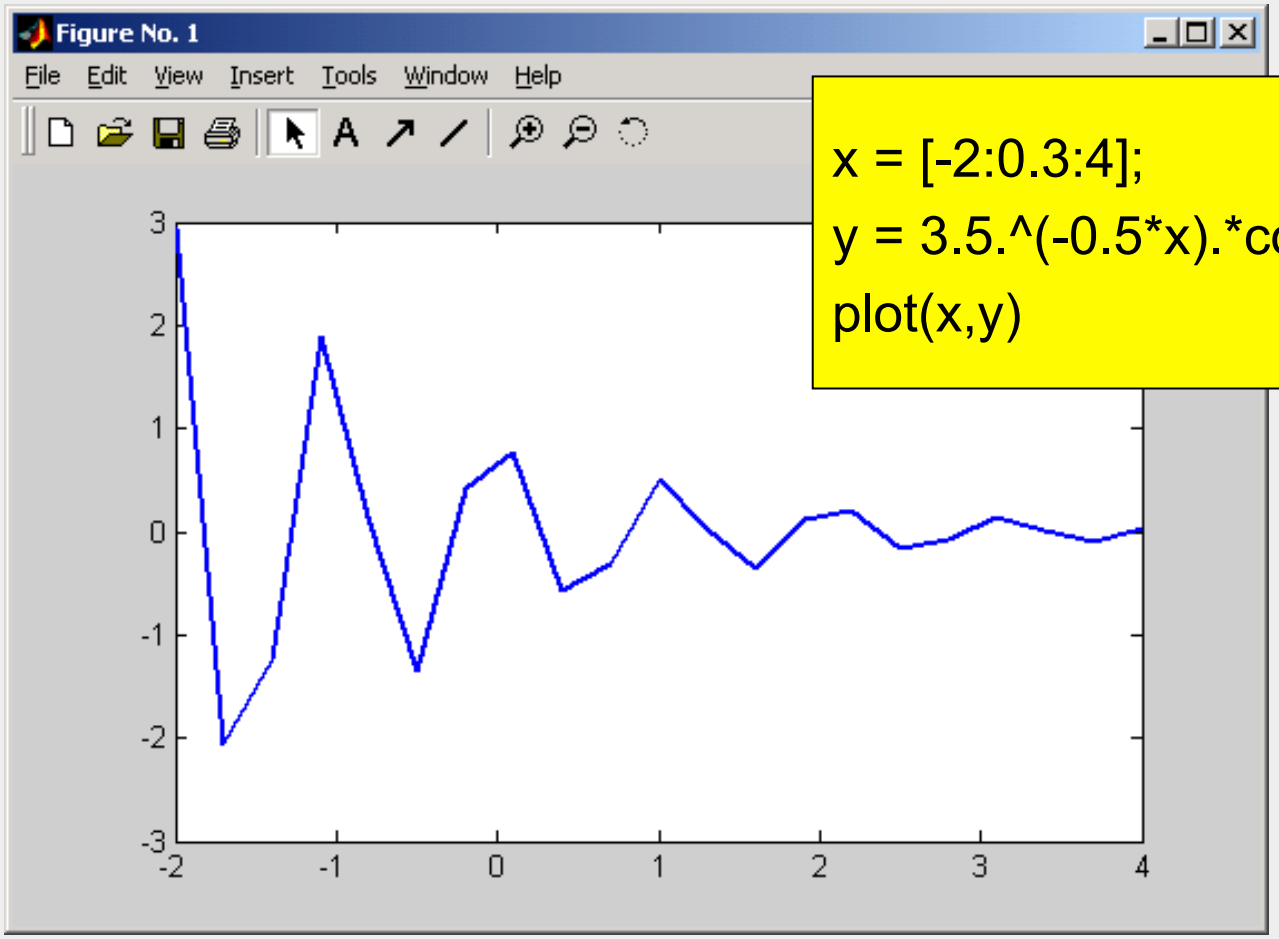
$$y = 3.5^{-0.5x}\cos(6x) \qquad \text{for} \quad -2 \le x \le 4$$



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:

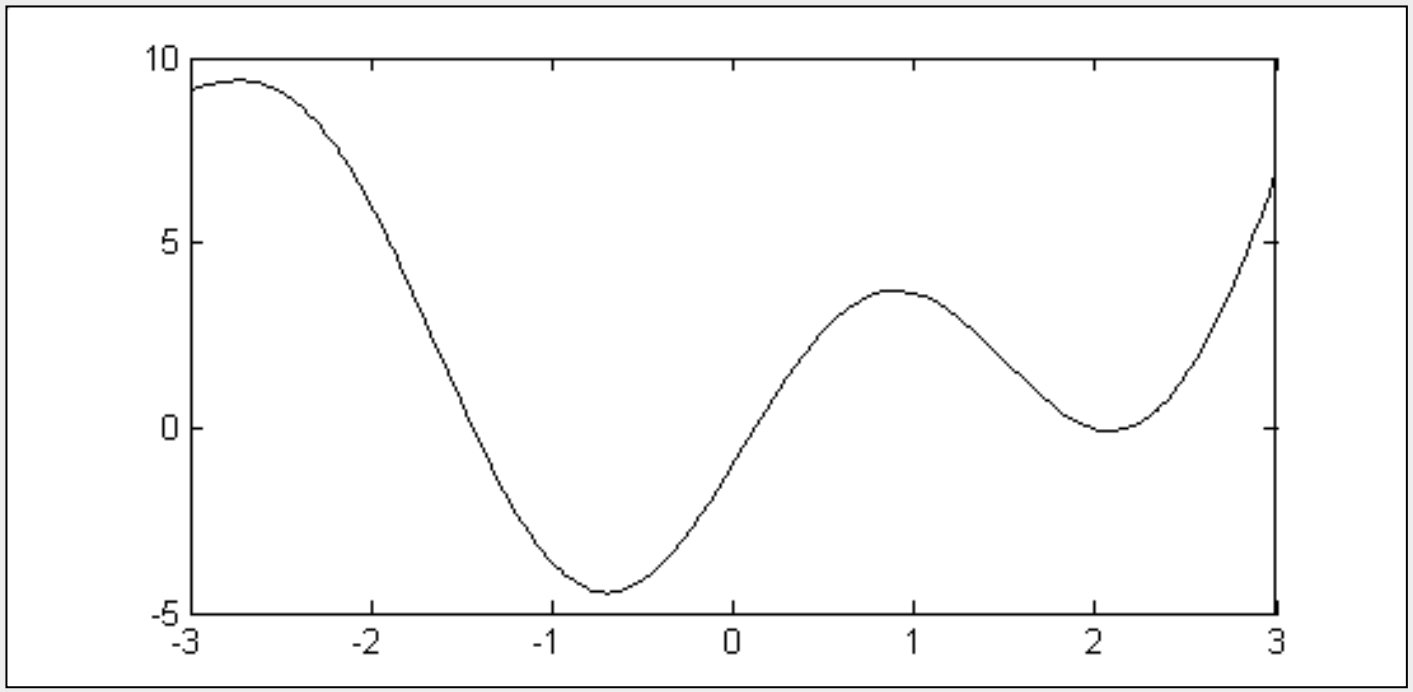
`[xmin,xmax]` or `[xmin,xmax,ymin,ymax]`

- Line specifiers can be added.

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

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

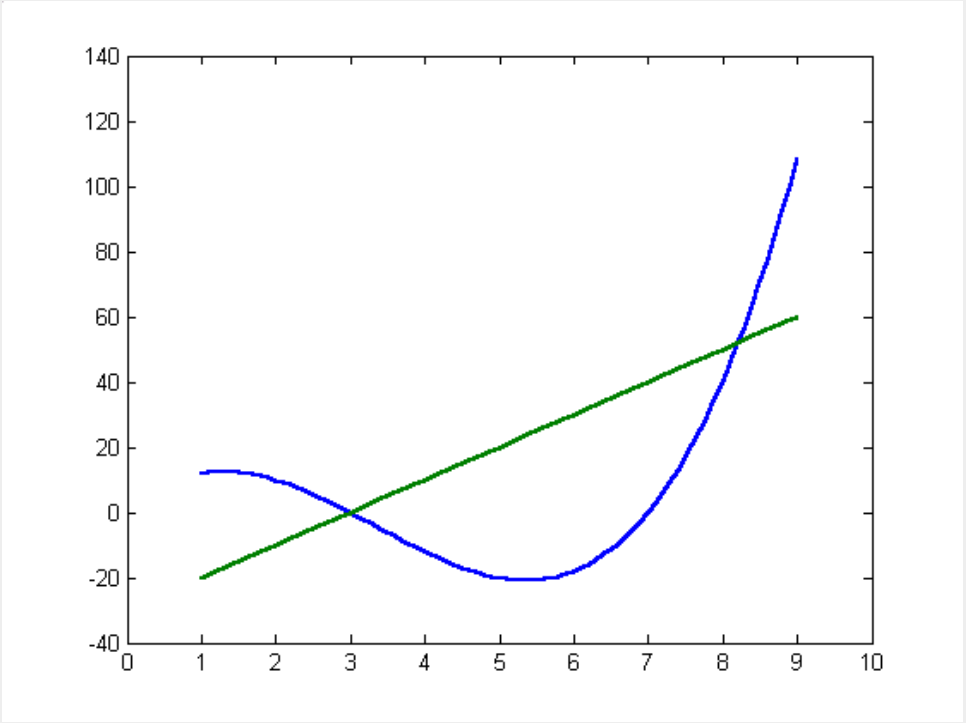
```
>> fplot('x^2 + 4 * sin(2*x) - 1', [-3 3])
```



PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT

Plotting two (or more) graphs in one plot:

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



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 $-2 \le x \le 4$, 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',x,yd,'--r',x,ydd,':k')
```

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

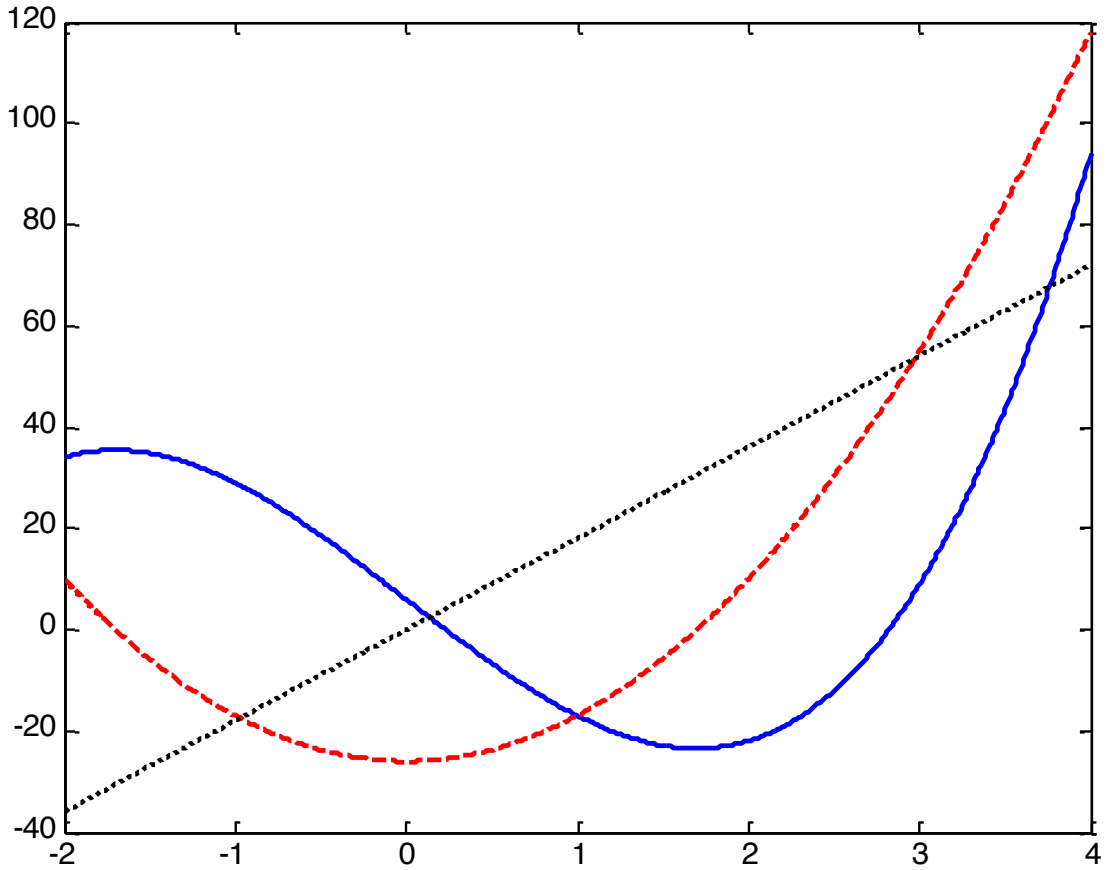
Vector **y** with the function value at each **x**.

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

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

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



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

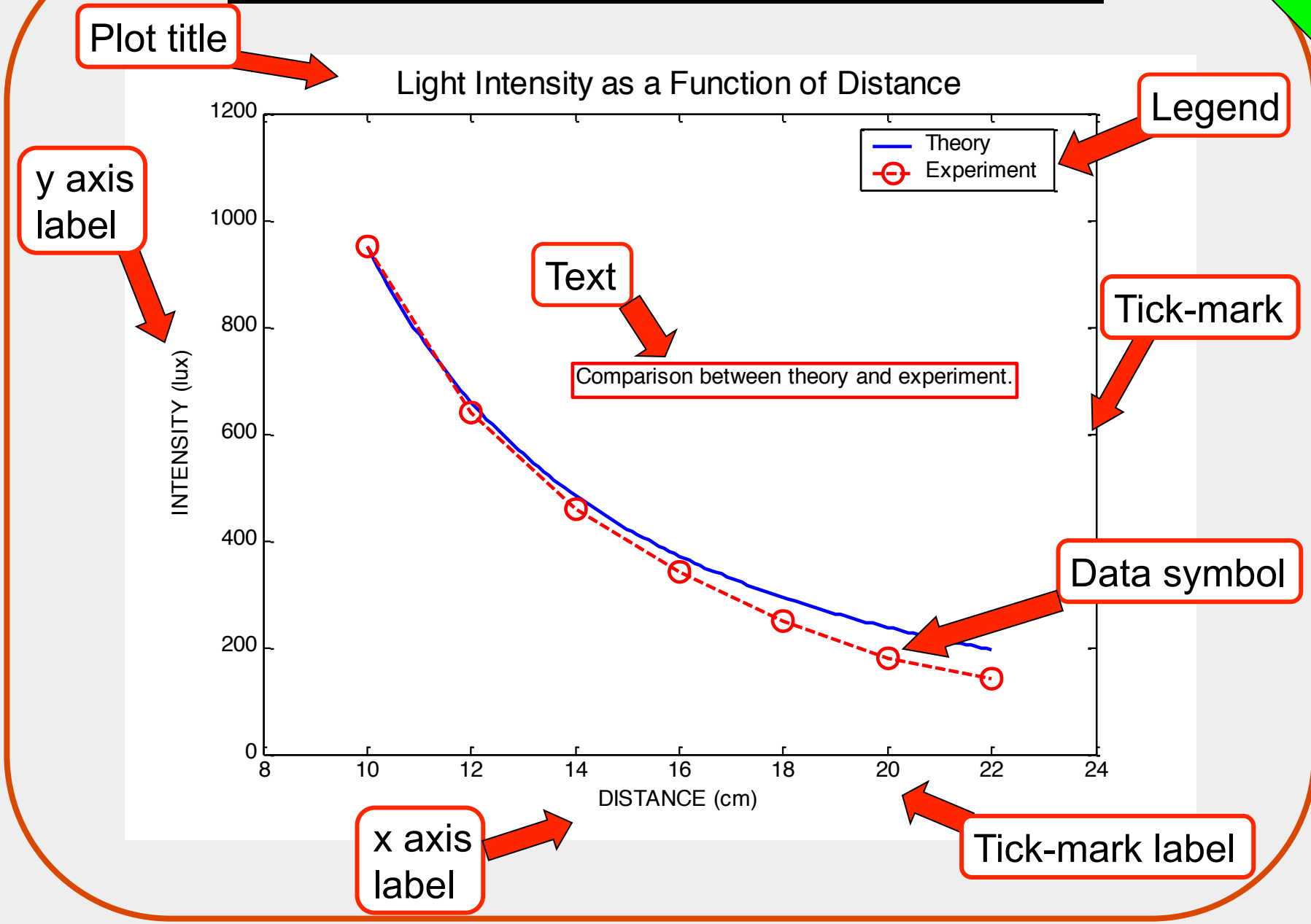
Plot of the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \le x \le 4$ all in the same plot.

```
x = [-2:0.01:4];
y = 3*x.^3-26*x+6;
yd = 9*x.^2-26;
ydd = 18*x;
plot(x,y,'-b')
hold on
plot(x,yd,'--r')
plot(x,ydd,':k')
hold off
```

First graph is created.

Two more graphs are created.

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('string')
```

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.

EXAMPLE OF A FORMATTED PLOT

Below is a script file of the formatted light intensity plot (2nd slide).

```
x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
plot(x,y,'-','LineWidth',1.0)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',10)
hold off
```

Creating vector **x** for plotting the theoretical curve.

Creating vector **y** for plotting the theoretical curve.

Creating a vector with coordinates of data points.

Creating a vector with light intensity from data.

EXAMPLE OF A FORMATTED PLOT

Formatting of the light intensity plot (cont.)

```
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('\fontname{Arial}Light Intensity as a Function of
Distance','FontSize',14)
axis([8 24 0 1200])
text(14,700,'Comparison between theory and
experiment.','EdgeColor','r','LineWidth',2)
legend('Theory','Experiment',0)
```

Labels for the axes.

Title for the plot.

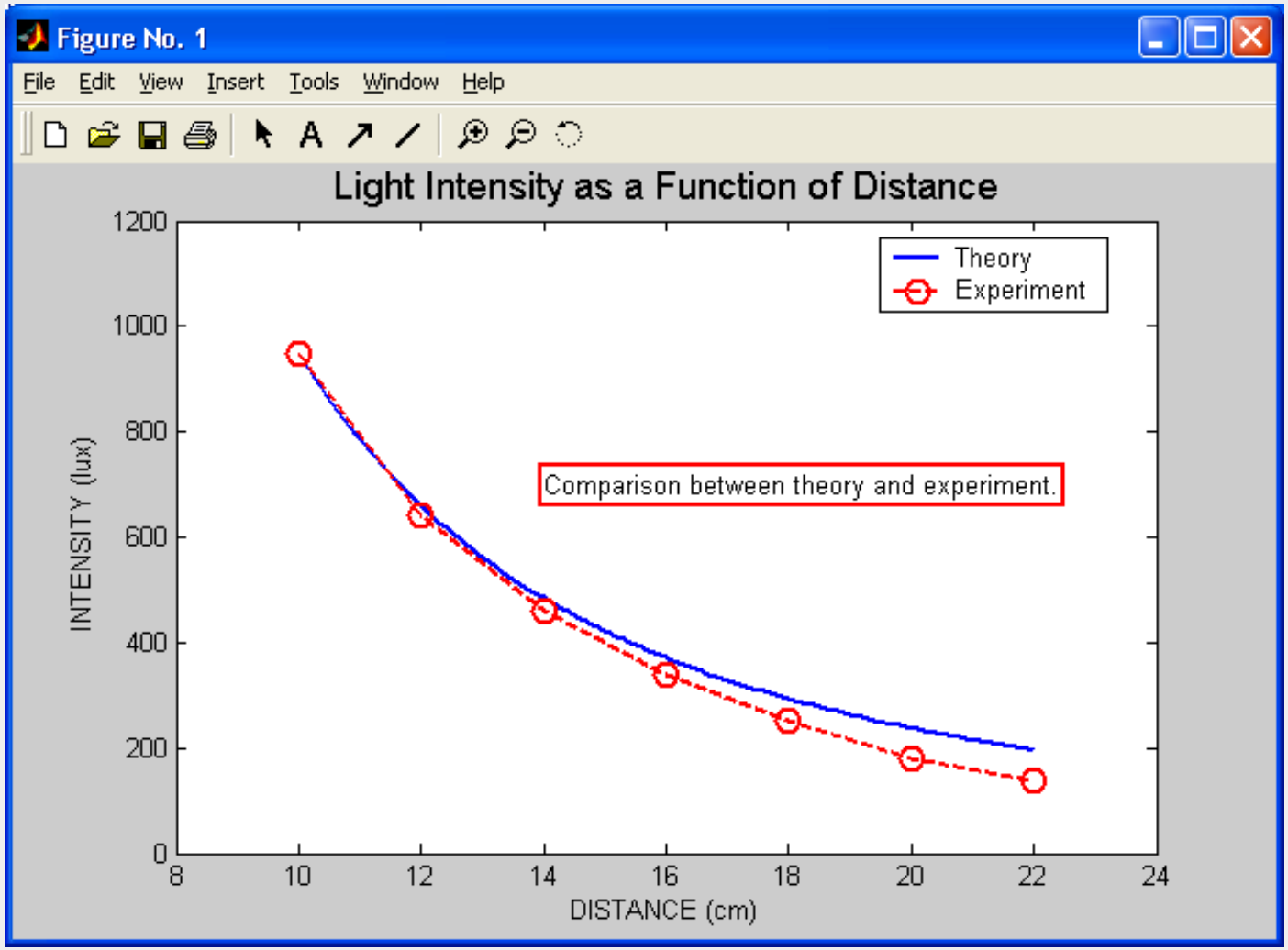
Setting limits of the axes.

Creating text.

Creating a 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.