Chapter 5

Two-Dimensional Plots

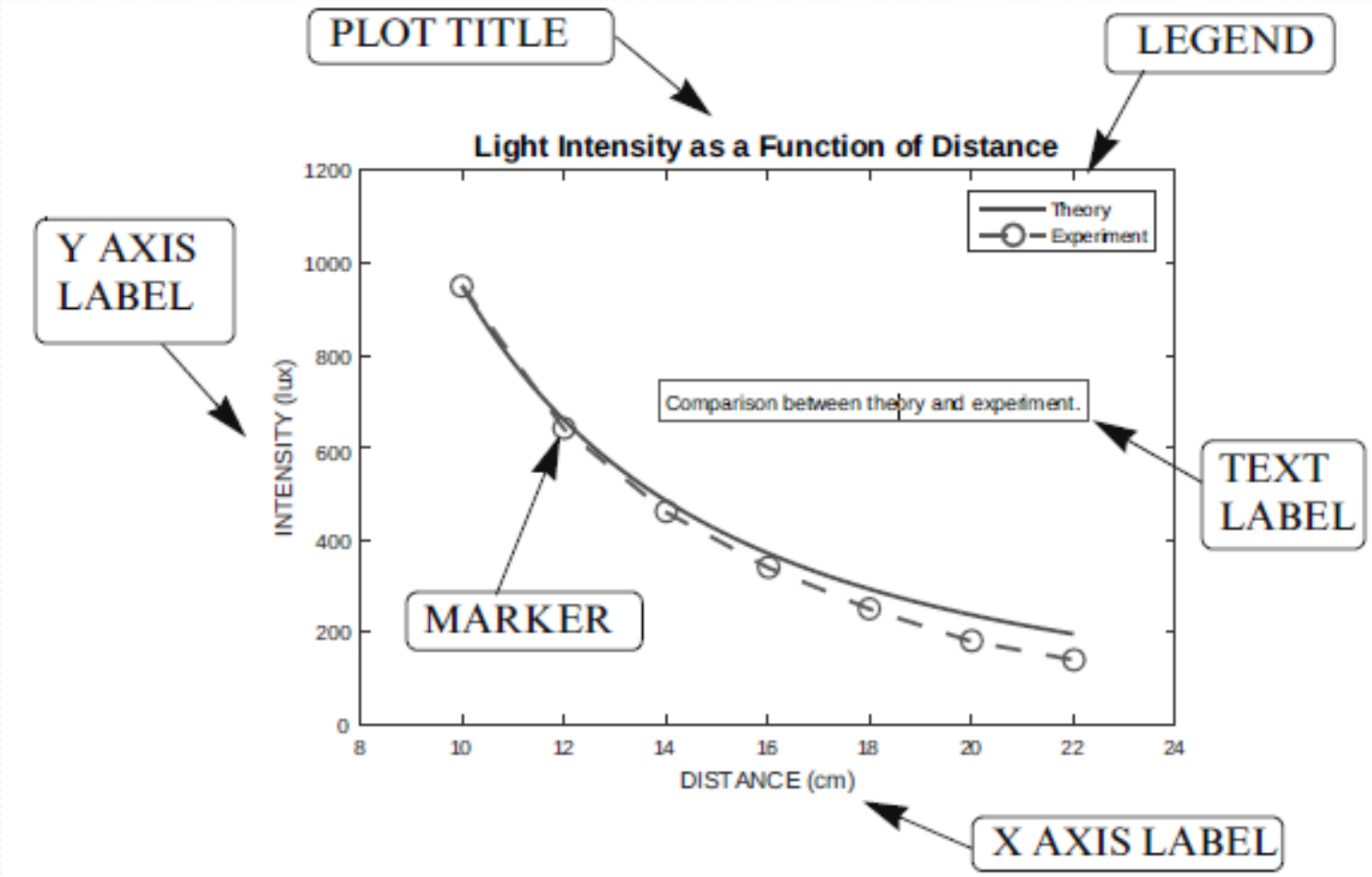
MATLAB An Introduction With Applications, 6th Edition
Dr. Amos Gilat
The Ohio State University

Slide deck by
Dr. Greg Reese
Miami University

This chapter will cover 2D (two-dimensional) plots. Many options:

- Linear, semi-logarithmic, logarithmic axes
- Line type, color, thickness
- Lots of different data-point markers
- Grid lines, titles, text comments, legends
- Subplots
- Bar, stair, polar plots

Important parts of a 2D plot



`plot` command used to make basic 2D plots. Simplest form is

```
plot(y)
```

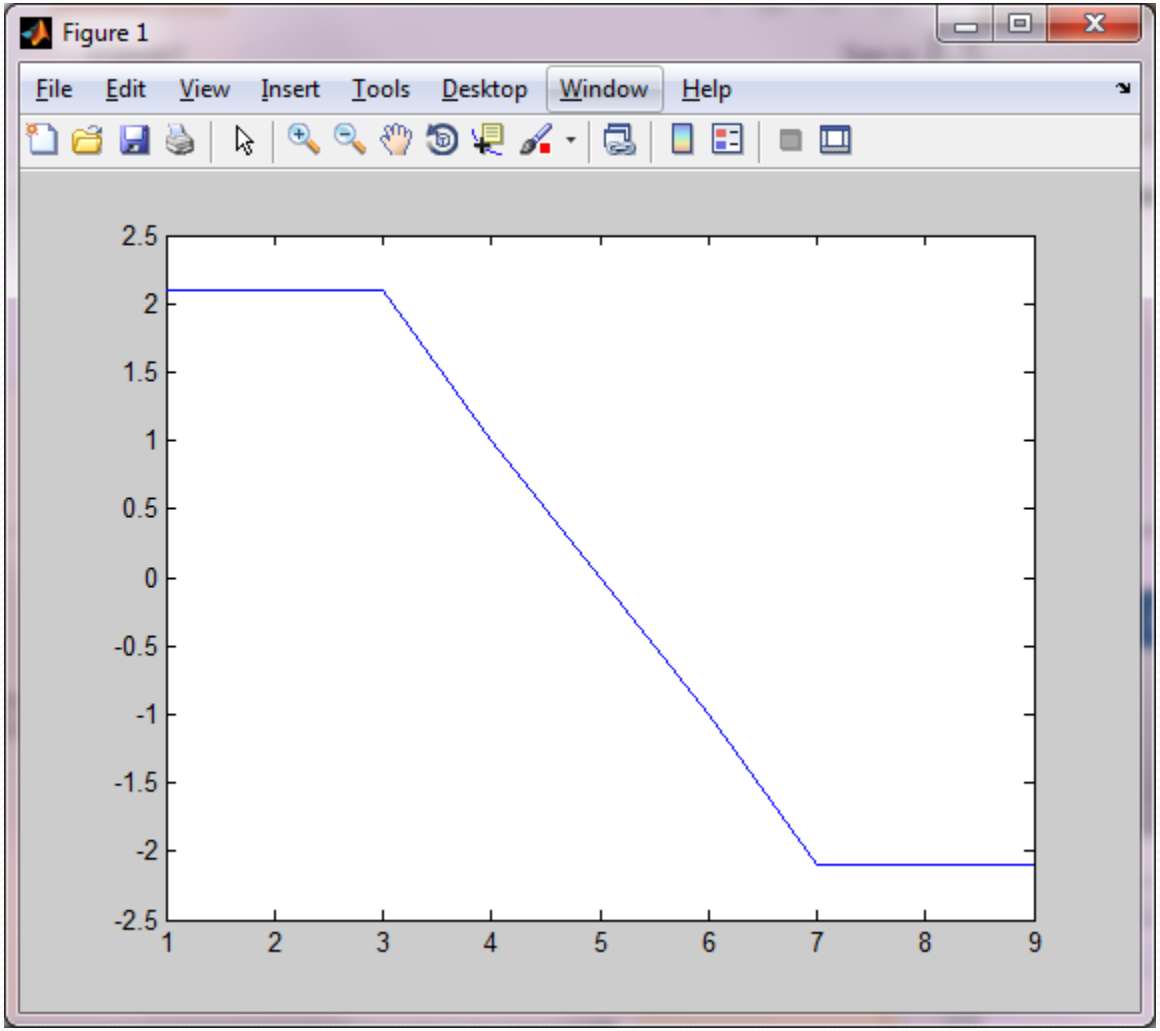
- Plots vector `y` on vertical axis, numbers 1 through N on horizontal axis (N = number of points in `y`)
- If there's a Figure Window, draws in it. Otherwise, creates a new Figure Window and draws in that

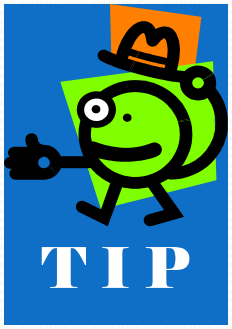
plot(y) default values

- Both axes linear
  - MATLAB chooses axis ranges so that end values are nice
- Points connected by straight lines
- No point markers
- Points and lines in blue

Example

```
>> y = [ 2.1 2.1 2.1 1 0 -1 -2.1 -2.1 -2.1 ];
>> plot( y )
```





If after issuing plot command, no Figure Window appears, window is buried. Click on Figure Window icon on task bar to make window appear

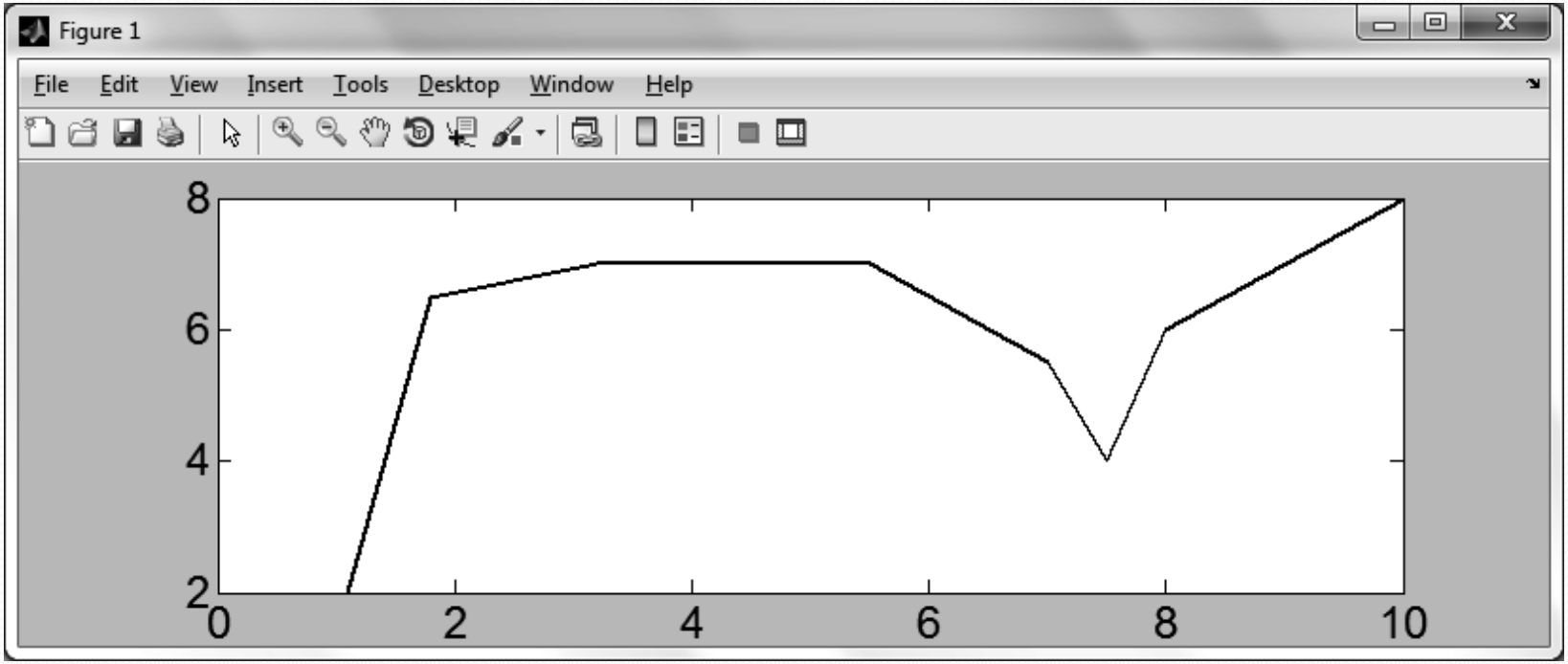
Second simplest form is

`plot(x,y)`

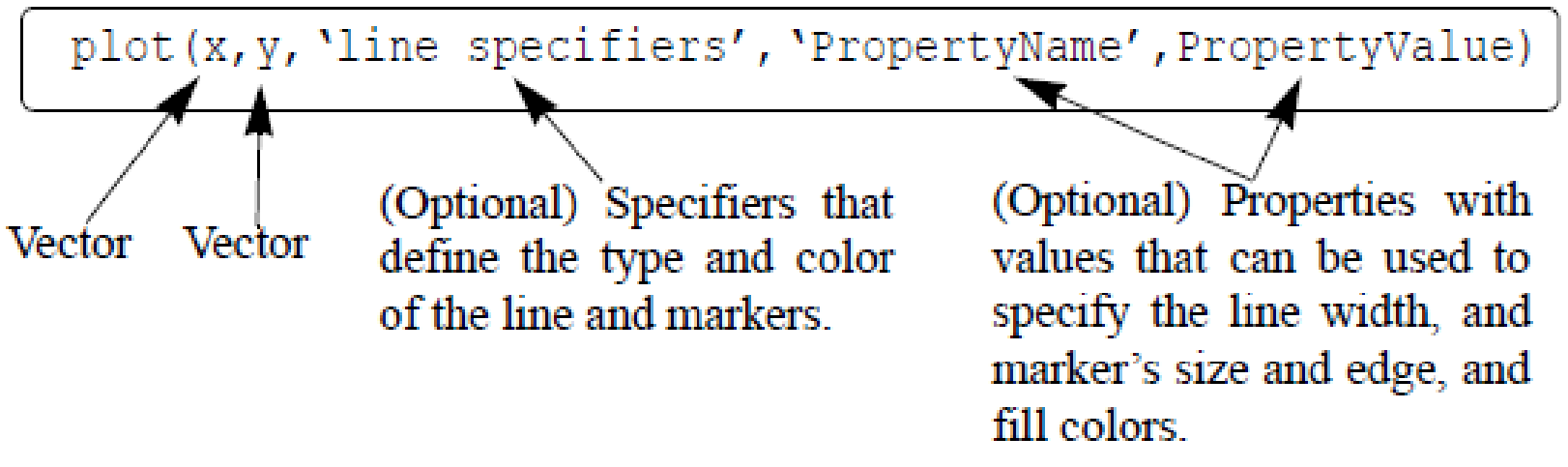
- `x` and `y` are vectors that have same size (number of elements) but any dimension
- `x`-values on horizontal axis, `y`-values on vertical axis
- Default values same as for `plot(x)`

Example

```
>> x=[1.1  1.8  3.2  5.5  7  7.5  8  10];
>> y=[2  6.5  7  7  5.5  4  6  8];
>> plot(x,y)
```



To use values other than defaults,



Line specifiers define style and color of lines, and marker types

Line Style	Specifier
solid (default)	-
dashed	--

Line Style	Specifier
dotted	:
dash-dot	-.

Line Color	Specifier
red	r
green	g
blue	b
cyan	c

Line Color	Specifier
magenta	m
yellow	y
black	k
white	w

Marker Types

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 about using the specifiers:

- 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.

Some examples:

`plot(x,y)`	A blue solid line connects the points with no markers (default).
`plot(x,y,'r')`	A red solid line connects the points.
`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 that are marked with diamond markers.

Property Name and Property Value:

- In `plot` command, type property name in quote marks, then comma, then 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.

For example, the command:

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

creates a plot that connects the points with a magenta solid line and circles as markers at the points. The line width is two points and the size of the circle markers is 12 points. The markers have a green edge line and yellow filling.

A note about line specifiers and properties:

The three line specifiers, which indicate the style and color of the line, and the type of the marker can also be assigned with a `PropertyName` argument followed by a `PropertyValue` argument. The Property Names for the line specifiers are:

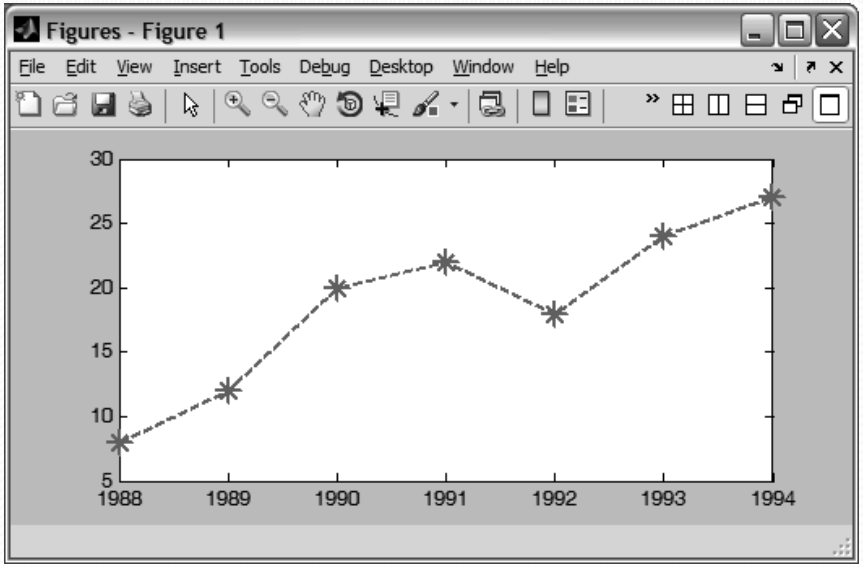
Specifier	Property Name	Possible property values
Line style	`linestyle` (or `LineStyle`)	Line style specifier from the table above, typed as a string.
Line color	`color` (or `Color`)	Color specifier from the table above, typed as a string.
Marker	`marker` (or `Marker`)	Marker specifier from the table above, typed as a string.

YEAR	1988	1989	1990	1991	1992	1993	1994
SALES (millions)	8	12	20	22	18	24	27

```
>> yr=[1988:1:1994];
>> sle=[8  12  20  22  18  24  27];
>> plot(yr,sle,'--r*','linewidth',2,'markersize',12)
>>
```

Line Specifiers: dashed red line and asterisk marker.

Property Name and Property Value: the line width is 2 points and the marker size is 12 points.



One way to plot a function of an independent variable:

1. Create a vector of values of the independent variable
2. Create a vector of value of function at every element of above vector
3. Plot using `plot(x,y)`

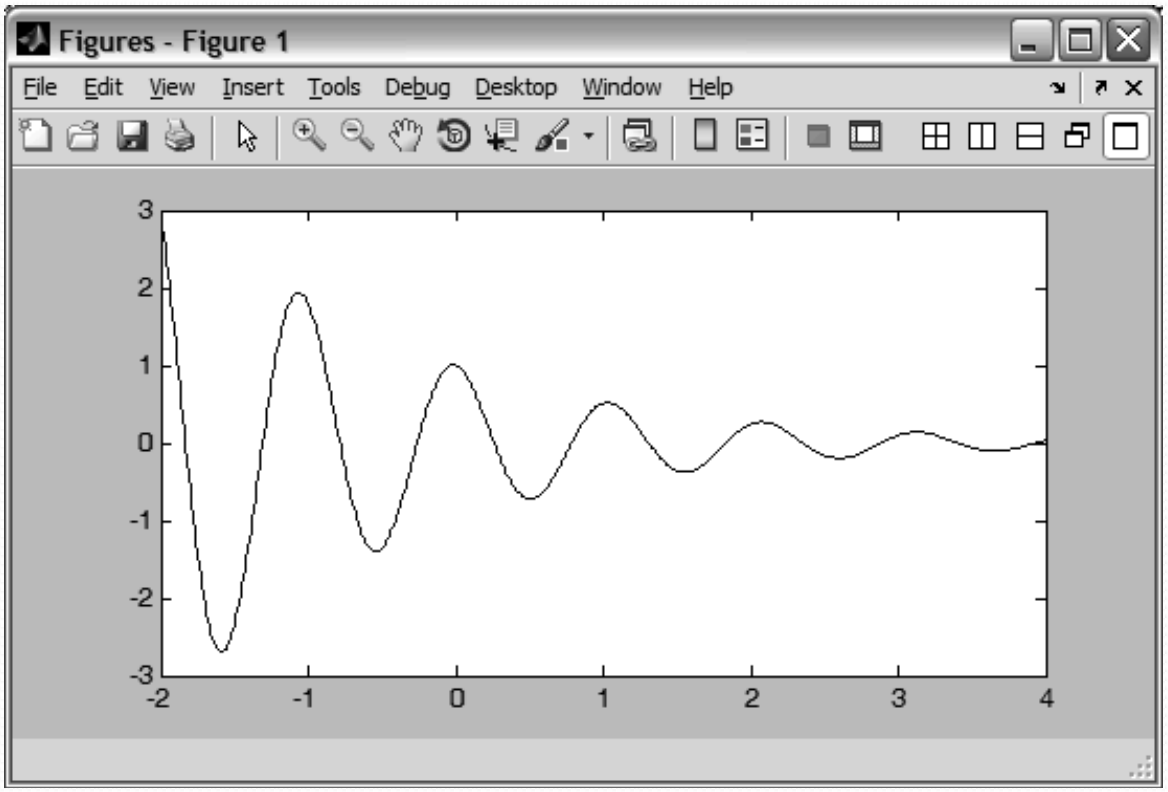
```
%  A script file that creates a plot of
%  the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];
y=3.5.^(-0.5*x).*cos(6*x);

plot(x,y)
```

Create vector x with the domain of the function.

Create vector y with the function value at each x.

Plot y as a function of x.



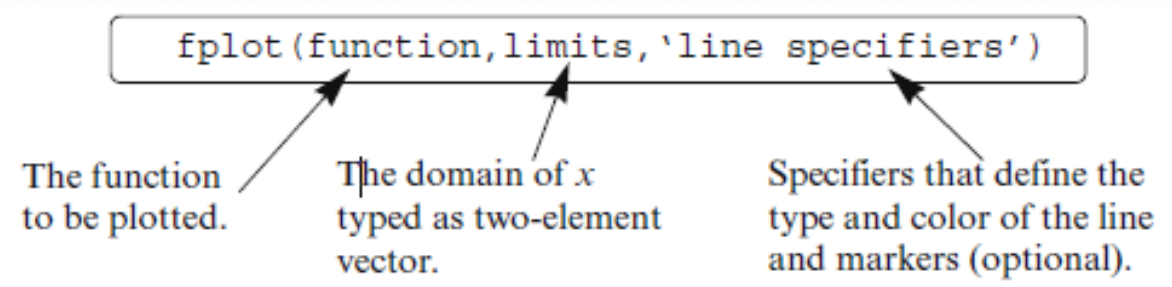
To copy entire Figure Window into another program, e.g., Word, PowerPoint

1. Click on Figure Window to make it current window
2. Select ALT+PRNTSCRN
3. Paste into other application

To copy just plot area of Figure Window into another program, e.g., Word, PowerPoint

1. In Figure Window, select Edit, then Copy Figure
2. Paste into other application

fplot plots a function $y = f(x)$ between given limits



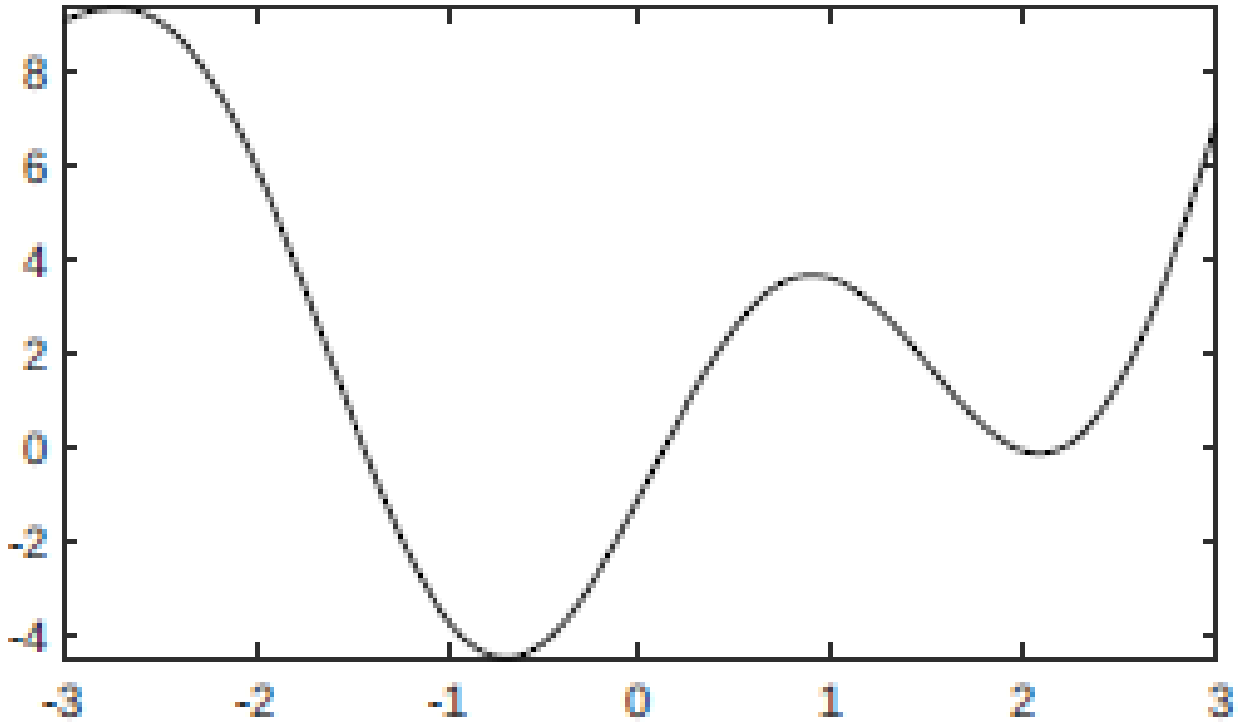
- Specify function in form of anonymous function (see Section 7.8)
  - For function f(x), anonymous form is `@(x) f(x)` e.g., for $f(x) = 8x^2 + 5\cos x$, anonymous function is `@(x) 8*x.^2+5*cos(x)`
- Function can include MATLAB's functions or ones that you write

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

- Independent variable in function can be any letter, e.g.,
  - `8*x^2+5*cos(x)` or `8*t^2+5*cos(t)`
- `limits` – two- or four-element vector that specifies plot axes limits
  - `[xmin xmax]` – limits on x-axis
  - `[xmin xmax ymin ymax]` – limits on both axes
- `line specifiers` – same as in `plot`

Example

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



Often want to graph more than one set of data on the same plot

MATLAB can do this three different ways

Plot two more graphs on same plot as follows (example for three graphs)

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

- Plots y vs. x, v vs. u, h vs. t
- Vectors of each pair must be same size
  - Can be different than sizes in other pairs
- Can use line specifiers by putting in triplets (x-data, y-data, specifier), e.g.,

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

```
x=[-2:0.01:4];
```
Create vector x with the domain of the function.

```
y=3*x.^3-26*x+6;
```
Create vector y with the function value at each x.

```
yd=9*x.^2-26;
```
Create vector yd with values of the first derivative.

```
ydd=18*x;
```
Create vector ydd with values of the second derivative.

```
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```
Create three graphs, y vs. x, yd vs. x, and ydd vs. x, in the same figure.

The plot that is created is shown in Figure 5-7.

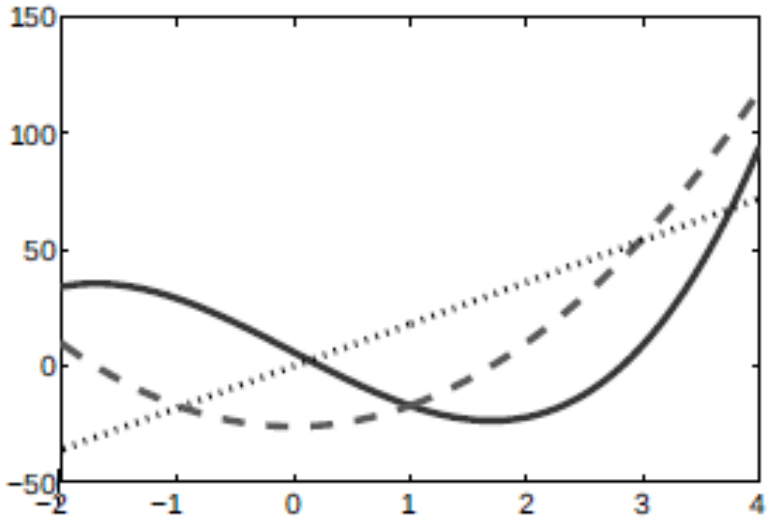


Figure 5-7: A plot of the function $y = 3x^3 - 26x + 10$ and its first and second derivatives.

Normally, each time you execute `plot` it erases previous plot and draws new one. To change this behavior:

- Draw the first graph with `plot`
- Issue the command `hold on`
- Call `plot` for each of the remaining graphs
- Issue the command `hold off`

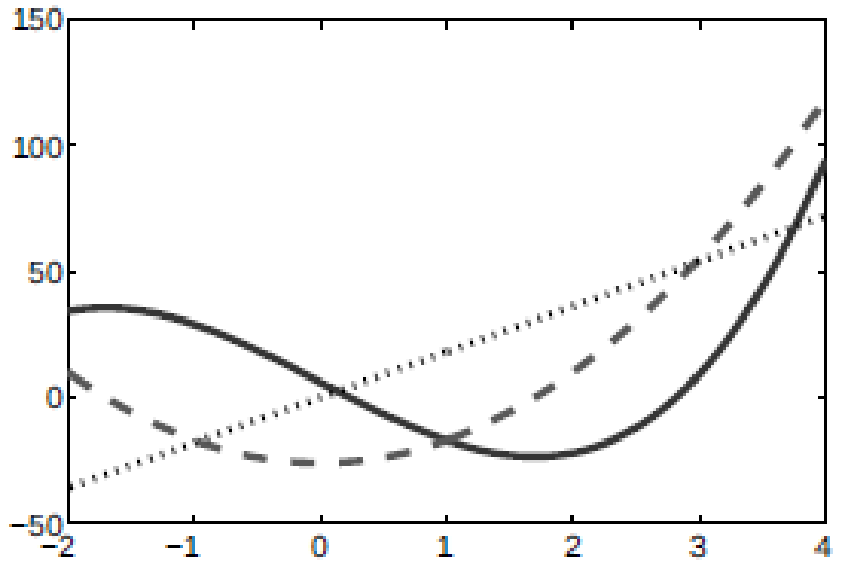
Graphs drawn after `hold on` are added to plot. Graphs drawn after `hold off` erase plot

5.3.2 Using the `hold on` and `hold off` Commands

```
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
```

The first graph is created.

Two more graphs are added to the figure.



`line` command adds additional graphs to an existing plot

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

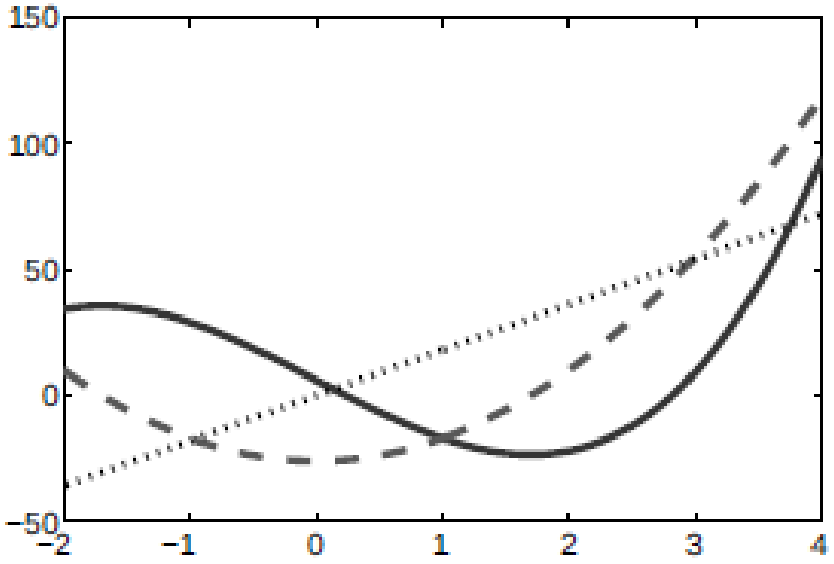
Example

```
line(x,y,'linestyle','--','color','r','marker','o')
```

adds graph drawn with dashed red line and circular markers to current plot

5.3.3 Using the `line` Command

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



Will learn how to spruce up a plot by adding

- Axis labels
- Title
- Legend
- Text
- Grid
- Custom axis ranges

`plot` or `fplot` make basic plot. After issuing that command, can use

- `xlabel('some text')` writes label below horizontal axis
  - Example: `xlabel('Time (sec)')`
- `ylabel('some text')` writes label to left of vertical axis
  - Example: `ylabel('Current (mA)')`

- `title('Some text')` writes title above plot
  - Example: `title('Diode Current')`
- `text(x,y,'Some text')` places text in figure with first character at `(x,y)`
  - Example:
    ```
    text(x,y,'Peak 3.5 sec after first')
    ```
- `gtext('Some text')` – figure window opens, user clicks on graph where she wants text to appear

`legend('text1','text2',...,pos)` writes legend

- For each graph (data set) displays short line in same style as graph line and adds specified text
  - First string goes with first graph plotted, second string goes with second graph plotted, etc.
- Most useful for plots having at least two graphs
- `pos` values in book are obsolete as of MATLAB 7.0 (R14). Type `help legend` or see documentation for new values

Formatting the text within the `xlabel, ylabel, title, text` and `legend` commands:

Can format text displayed by above commands

- Can set font, size, character color, background color, sub/superscript, style (bold, italic, etc.)
- Can display Greek letters
- Can format using modifiers within text string or by adding property names and values to command

Text modifiers are placed inside text string and affect appearance of text

- All text following modifier gets altered
- To only modify some text put open brace ({), modifier, text-to-be-modified, close brace (})

Example titles

```
title('\it What You Should Never See')
```

makes

What You Should Never See

```
title('What You Should{\it Never}
See')
```

makes

What You Should *Never* See

Example titles

```
title('\fontname{Old English Text MT}...
My Name is Robin Hood')
```

makes

My Name is Robin Hood

```
title(...
'{\fontname{Old English Text MT}Robin
Hood}...
was here')
```

makes

Robin Hood was here

Some common modifiers

- `\bf` – **bold face**
- `\it` – *italic*
- `\rm` – normal font
- `\fontname{fontname}` – font name
- `\fontsize{fontsize}` – font size

Subscript and superscript:

To make single character

- Subscript – precede it by underscore (_)
- Superscript – precede it by caret(^)

For multiple characters, same as above but enclose characters in (curly) braces

- `xlabel('H_2O')` makes H_2O
- `ylabel('e^{-k*sin(x)}')` makes $e^{-k*sin(x)}$

Greek characters:

To make a Greek letter, follow backslash with letter name (in English!)

- Name in lowercase makes lowercase Greek letter
- Name with capitalized first letter makes uppercase Greek letter

```
ylabel('Standard deviation (\sigma) of resistance in M\Omega')
```

makes

Standard deviation (σ) of resistance in MΩ

Some Greek characters

Characters in the string	Greek Letter
\alpha	α
\beta	β
\gamma	γ
\theta	θ
\pi	π
\sigma	σ

Characters in the string	Greek Letter
\Phi	Φ
\Delta	Δ
\Gamma	Γ
\Lambda	Λ
\Omega	Ω
\Sigma	Σ

For `xlabel, ylabel, title, text,` can also change display of entire text string by using property name – property value pairs, e.g.,

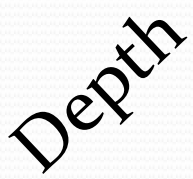
```
text(x,y,'Some text',PropertyName,PropertyValue)
```

- `PropertyName` is text string
- `PropertyValue` is number if value is number or text string if value is letter or word

Example

```
text(x,y,'Depth','Rotation',45)
```

makes



Some property-name property-value pairs

Property name	Description	Possible property values
`Rotation`	Specifies the orientation of the text.	Scalar (degrees) Default: 0
`FontAngle`	Specifies italic or normal style characters.	`normal`, `italic` Default: `normal`
`FontName`	Specifies the font for the text.	Font name that is available in the system.
`FontSize`	Specifies the size of the font.	Scalar (points) Default: 10
`FontWeight`	Specifies the weight of the characters.	`light`, `normal`, `bold` Default: `normal`
`Color`	Specifies the color of the text.	Color specifiers (see Section 5.1).
`Background-Color`	Specifies the background color (rectangular area).	Color specifiers (see Section 5.1).
`EdgeColor`	Specifies the color of the edge of a rectangular box around the text.	Color specifiers (see Section 5.1). Default: none.
`LineWidth`	Specifies the width of the edge of a rectangular box around the text.	Scalar (points) Default: 0.5

The `axis` command:

MATLAB makes axes limits in `plot` command so that all data appears and limits are nice numbers. Can change that with `axis` command

Common `axis` variations are:

`axis([xmin xmax ymin ymax])`

- Sets limits of both axes

`axis equal`

- Sets same scale for both axes

`axis square`

- Sets axis region to be square

`axis tight`

- Sets axes limits to range of data (not usually nice numbers!)

The `grid` command:

`grid on`

- Adds grid lines to plot

`grid off`

- Removes grid lines from plot

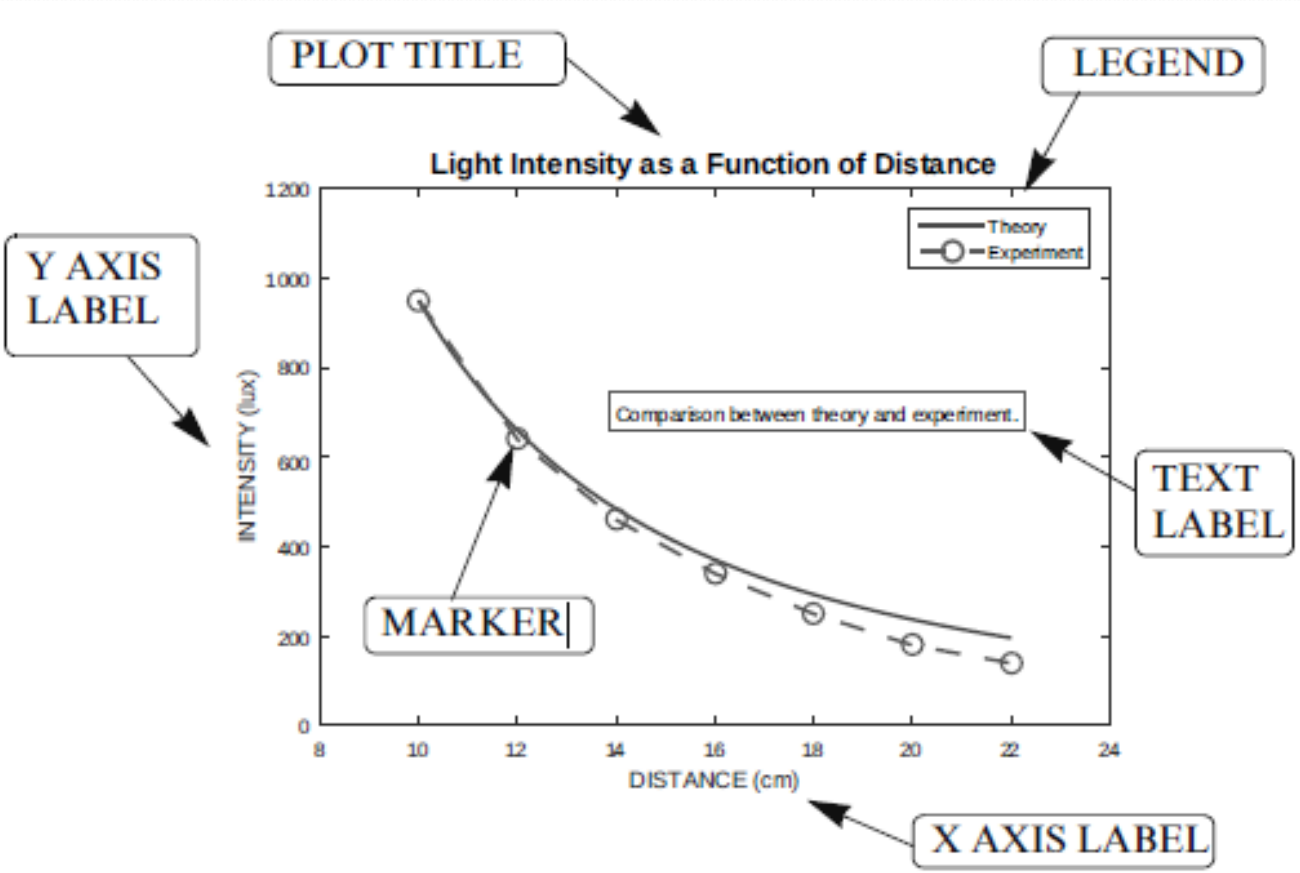
This script

```
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)
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.','Edge-
Color','r','LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize'
legend('Theory','Experiment',0)
hold off
```

Formatting text inside the `title` command.

Formatting text inside the `text` command.

Makes this plot

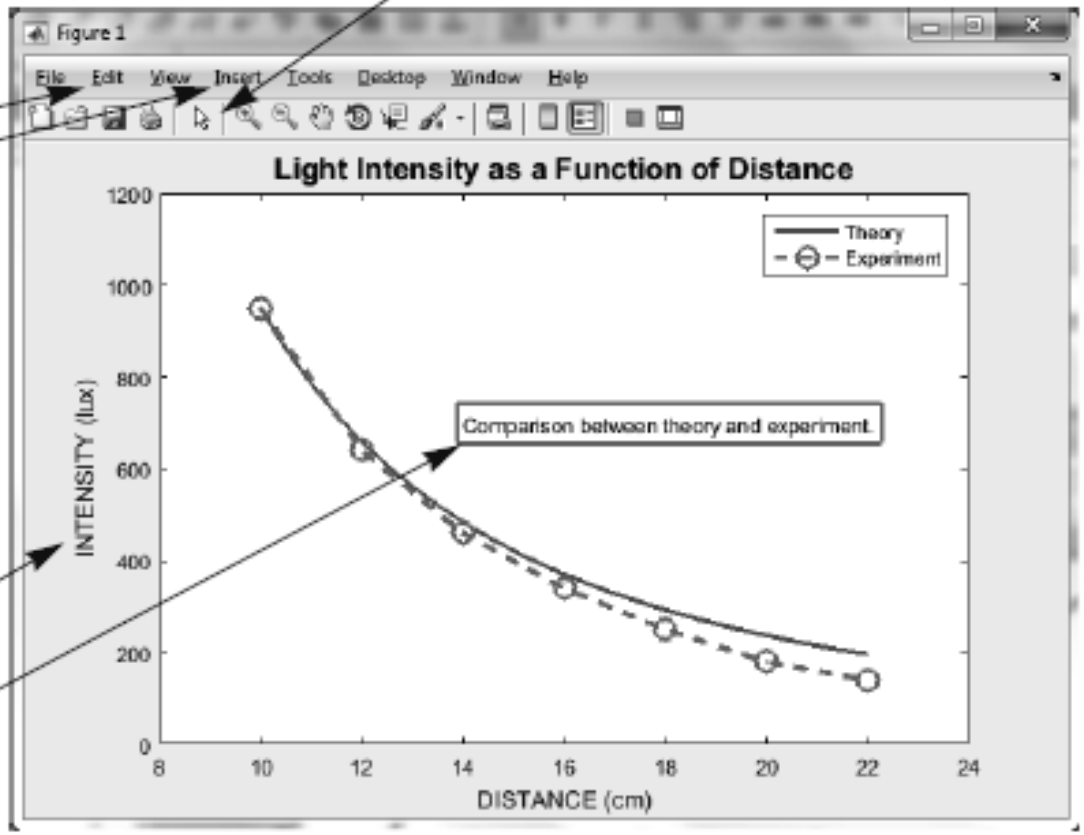


Can interactively format plot from Figure Window

Click the arrow button to start the plot edit mode. Then click on an item. A window with formatting tool for the item opens.

Use the **Edit** and **Insert** menus to add formatting objects, or to edit existing objects.

Change position of a label, legend, or other object by clicking on the object and dragging.



Often use plot with one or both axes being logarithmic (log)

- Used to display data with large range of values
- Used to make some functional relationships more apparent
  - For example, $y = 10^{(2x+3)}$ is a straight line on a semilog plot

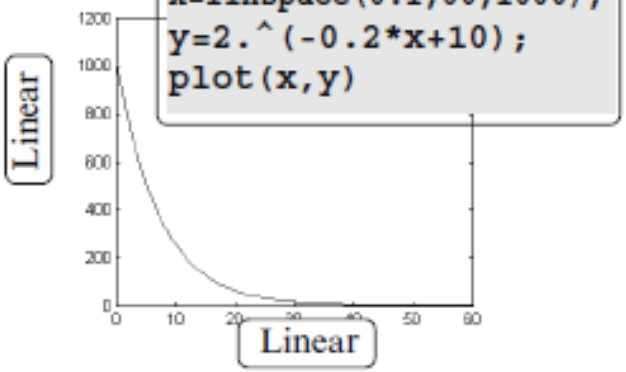
MATLAB commands for log plots

`semilogy(x,y)`	Plots y versus x with a log (base 10) scale for the y axis and linear scale for the x axis.
`semilogx(x,y)`	Plots y versus x with a log (base 10) scale for the x axis and linear scale for the y axis.
`loglog(x,y)`	Plots y versus x with a log (base 10) scale for both axes.

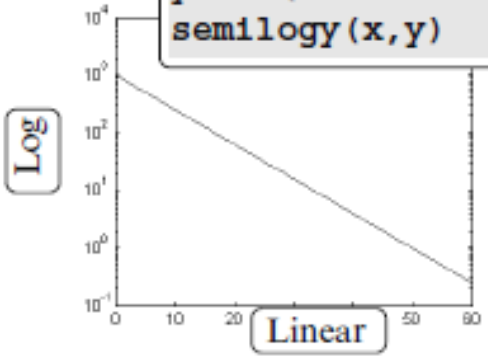
- Can use line specifiers and property-name property-value pairs as in `plot`
- On logarithmic axis, make sure all data is > 0 because otherwise log is undefined

Examples

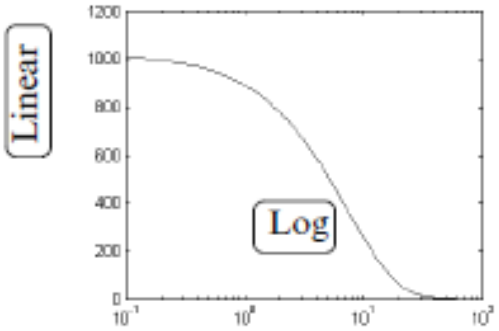
```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
plot(x,y)
```



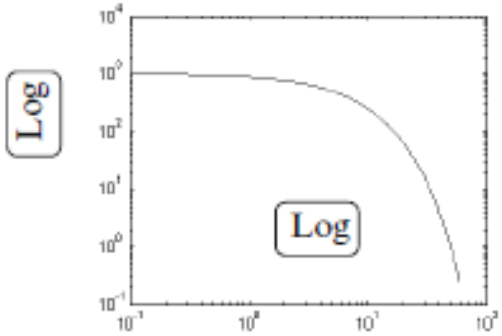

```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
semilogy(x,y)
```




```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
semilogx(x,y)
```




```
x=linspace(0.1,60,1000);
y=2.^(-0.2*x+10);
loglog(x,y)
```



Experimental data that is plotted usually also shows some measure of the uncertainty in the measurements

- Often shown by *error bars*, (usually small) vertical lines above and below data points. Their size is the size of the uncertainty
- Uncertainty measure is often the *standard error*, which is approximately the standard deviation of the samples used to compute a data point

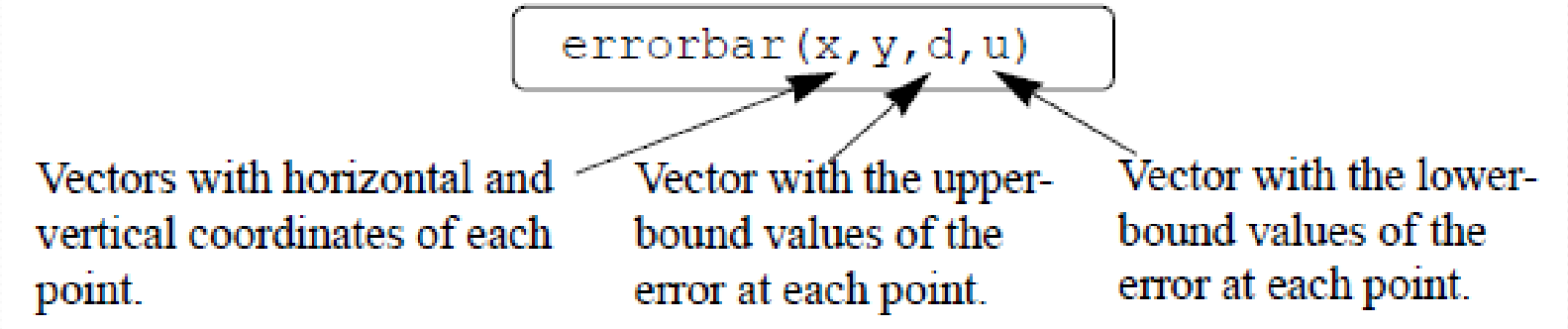
`errorbar(x,y,e)`

- All vectors in the command must be same size
- `x` and `y` are horizontal- and vertical-axis data
- `e` is error measurement at each point
  - At each `y(i)`, MATLAB draws vertical error bar from `y(i)-e(i)` to `y(i)+e(i)`

`errorbar(x,y,d,u)`

- All vectors in the command must be same size
- `x` and `y` are horizontal- and vertical-axis data
- `u,d` are error measurements at each point
  - At each `y(i)`, MATLAB draws vertical error bar from `y(i)-d(i)` to `y(i)+u(i)`

NOTE: - descriptions for second and third arrows at bottom of page should be switched



MATLAB has lots of special types of plots, e.g., bar, stairs, stem, pie

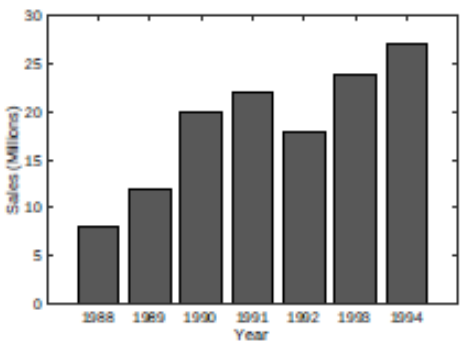
- For more information on types of plots, click on the Help icon, then on MATLAB, then scroll down to the Graphics section and click on 2-D and 3-D plots

Examples of specialized plots

Vertical Bar Plot

Function format:

`bar(x,y)`



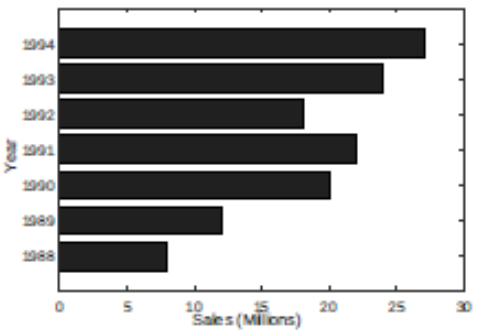

```
yr=[1988:1994];
sle=[8 12 20 22 18 24
27];
bar(yr,sle,'r')
xlabel('Year')
ylabel('Sales (Mil-
lions)')
```

The bars are in red.

Horizontal Bar Plot

Function format:

`barh(x,y)`

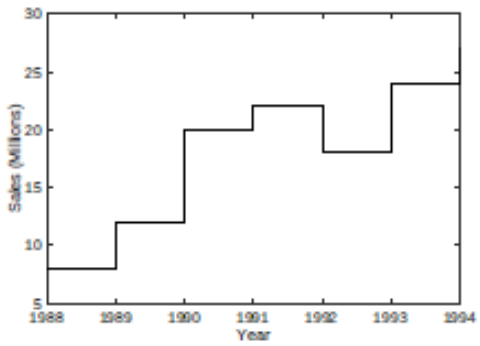



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
barh(yr,sle)
xlabel('Sales (Millions)')
ylabel('Year')
```

Stairs Plot

Function format:

`stairs(x,y)`

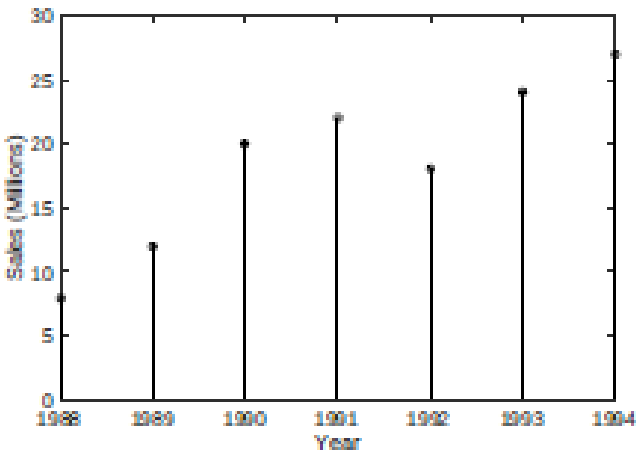



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
stairs(yr,sle)
xlabel('Year')
ylabel('Sales (Mil-
lions)')
```

Stem Plot

Function Format

```
stem(x,y)
```

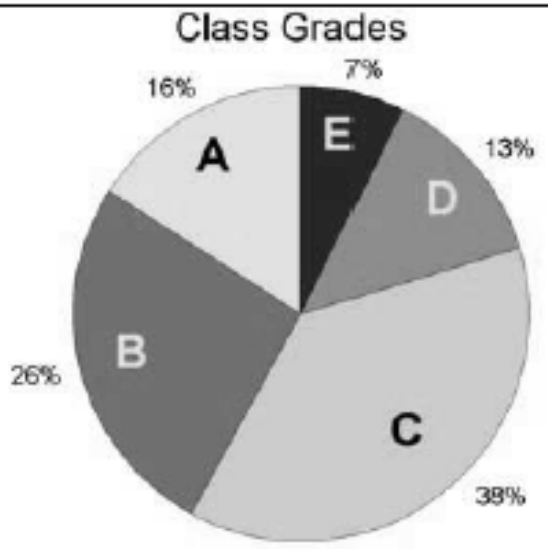



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
stem(yr,sle)
xlabel('Year')
ylabel('Sales (Mil-
lions)')
```

Pie Plot

Function format:

```
pie(x)
```




```
grd=[11 18 26 9 5];
pie(grd)
title('Class Grades')
```

MATLAB draws the sections in different colors. The letters (grades) were added using the Plot Editor.

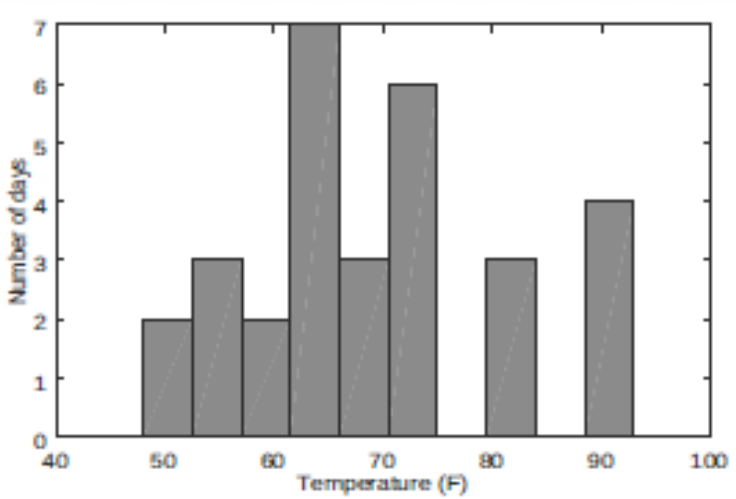
A *histogram* is a plot of the distribution of data. Entire range of data broken into consecutive subranges or *bins*.

In histogram plot

- Each bin represented by vertical bar
- Left and right of vertical bar show range of data in bin
- Height of vertical bar is number of data points in bin

MATLAB command `hist` makes histogram. Simples form is `hist(y)`

- `y` is vector of data points
- `hist` divides range into ten equal bins, then plots result




```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89
91 80 59 69 56 64 63 66 64 74 63 69];
>> hist(y)
```

Additional forms of `hist`:

`hist(y,nbins)`

- MATLAB divides range into `nbins` (scalar) bins of equal size

`hist(y,x)`

- `x` (vector) specifies midpoint of each bin
  - Spacing between consecutive elements of `x` can be different
  - Bin edges are midpoints of consecutive bins

Can get histogram heights if desired

```
n=hist(y) n=hist(y,nbins)
n=hist(y,x)
```

- Output `n` is a vector
  - Size of `n` is number of bins
  - Value of element of `n` is number of data points in corresponding bin

For two forms, can also get bin centers

`[n xout]=hist(y)` or `[n xout]=hist(y,nbins)`

- `n` same as before
- `xout(i)` is center of *i*th bin

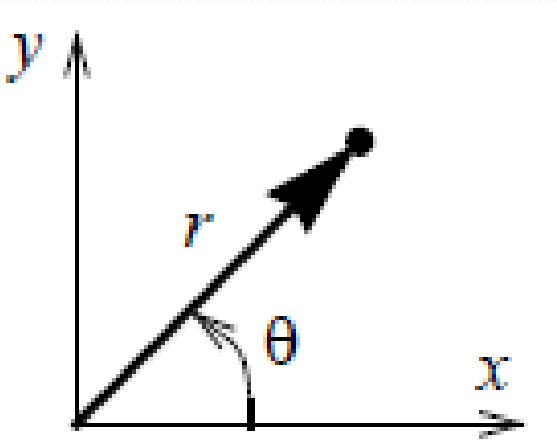
```
>> [n xout]=hist(y)

n =
   2   3   2   7   3   6   0   3   0   4

xout =
    50.2500   54.7500   59.2500   63.7500   68.2500   72.7500
77.2500   81.7500   86.2500   90.7500
```

In *polar coordinates*, points in plane specified by (r,θ)

- r is distance from origin
- θ is angle from positive, horizontal axis. θ is positive in counterclockwise direction



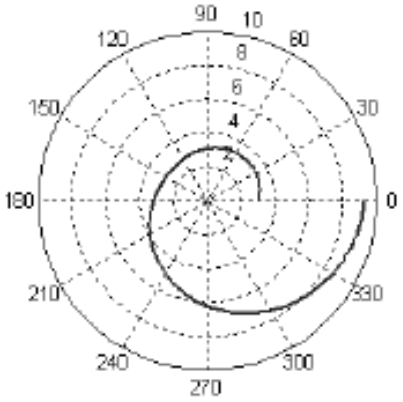
To make polar plot in MATLAB, use

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

- `theta` in radians and as defined previously
- `radius` as defined in previously
- Both must be vectors and of same size
- `line specifiers` same as in `plot`

For example, a plot of the function $r = 3\cos^2(0.5\theta) + \theta$ for $0 \le \theta \le 2\pi$ is shown below.

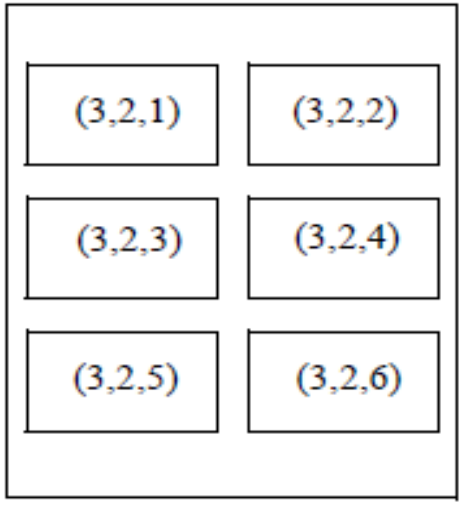
```
t=linspace(0,2*pi,200);
r=3*cos(0.5*t).^2+t;
polar(t,r)
```



`subplot(m,n,p)`

divides Figure Window into `m` rows and `n` columns of subplots

- Subplots numbered from left to right and top to bottom, with upper left being subplot `1` and lower right subplot `m*n`. `p` in subplot command refers to this numbering



Subplot numbers for 3x2 set of subplots

`subplot(m,n,p)`

- If subplots don't exist, `subplot` creates them and makes subplot `p` the current subplot
- If subplots exist, `subplot` makes subplot `p` the current one
- When `subplot` defines current subplot, next `plot` and formatting commands draw in current subplot

See Sample Problem 5-2 for example of `subplot` use

On execution, any plotting command

1. Creates a Figure Window (if none exists)
2. Erases any plot in that window
3. Draws new plot

Can be useful though to have plots in multiple windows. `figure` command lets you do this

`figure`

1. Creates a new Figure Window
2. Labels the window "Figure n"
   - *n* such that first window is Figure 1, second is Figure 2, etc.
3. Makes new window the *active* Figure Window
4. Brings window to front of the screen

Subsequent plotting commands draw in the active Figure Window

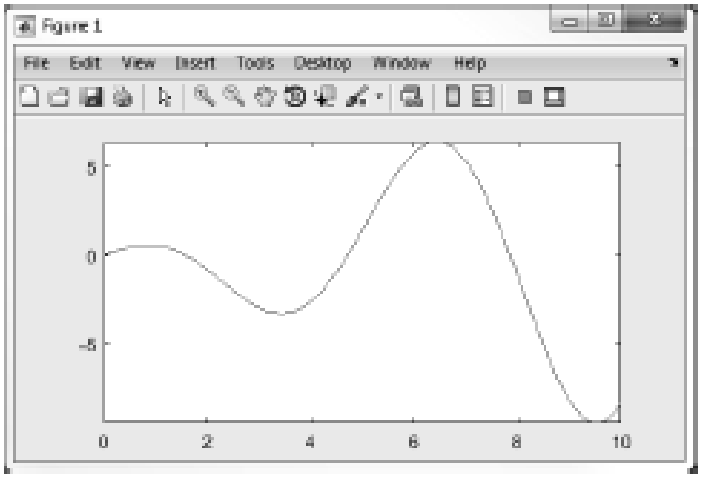
Example

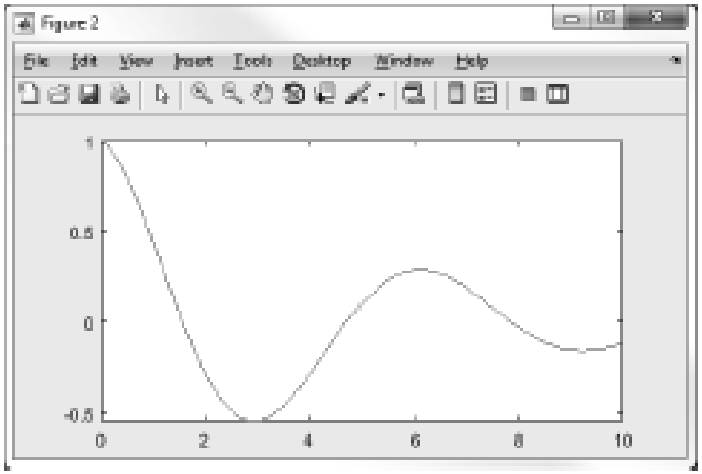
```
>> fplot(@ (x) x.*cos(x),[0,10])
>> figure
>> fplot(@ (x) exp(-0.2*x).*cos(x),[0,10])
```

Plot displayed in Figure 1 window.

Figure 2 window opens.

Plot displayed in Figure 2 window.





`figure(n)`

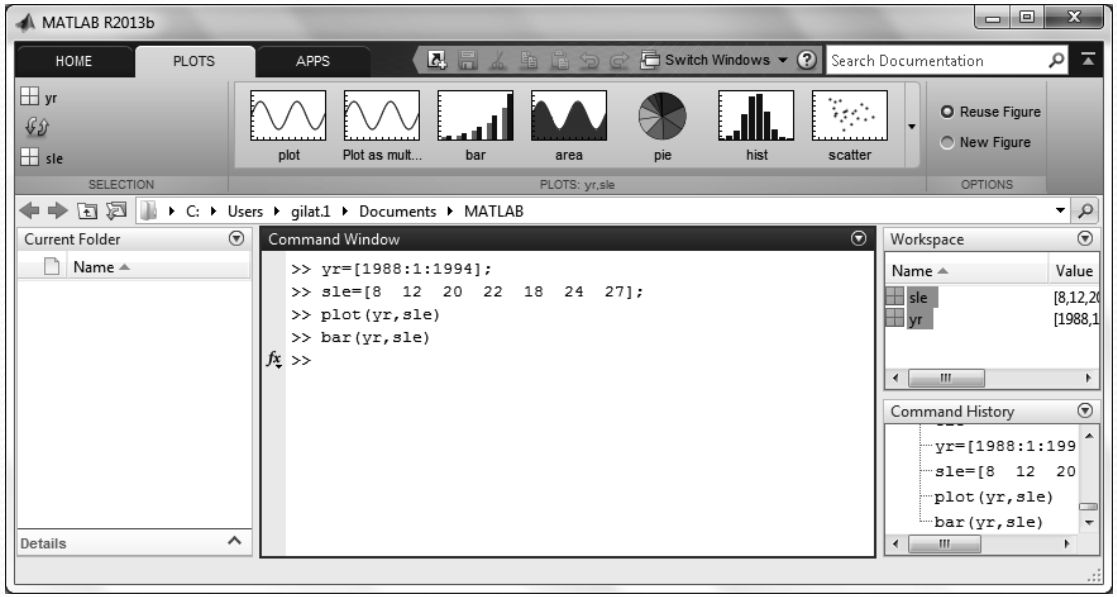
- If Figure Window `n` exists, makes it the active window
- If Figure Window `n` doesn't exist, creates it and makes it the active window
- `figure(n)` useful in scripts, e.g., scripts in which data set 1 is displayed in Figure 1, data set 2 is displayed in Figure 2, etc.

Use `close` command to close figure windows

- `close` closes active Figure Window
- `close(n)` closes Figure Window n
- `close all` closes all open Figure Windows

Can make plots interactively by using PLOTS Toolstrip in Command Window

1. If Workspace Window not displayed, click on Layout icon, then on Workspace
2. Make sure any variables you want to use in plotting have values assigned to them
3. Click on PLOT tab

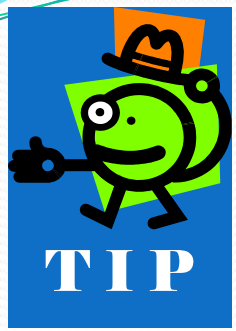


To make a 2-D plot

1. In Workspace Window, click on vector you want to use for horizontal axis
2. While holding down CTRL key, click on other vector(s) you want to plot on vertical axis
   - If you only select one vector, MATLAB plots element numbers on horizontal axis and vector on vertical axis
3. Plot types you can use will be visible in toolstrip. Click on desired plot type

After clicking on plot type, MATLAB opens new Figure Window with data plotted

- To replot same data with different plot type, select Reuse Figure on right side of toolstrip, then click on new plot type. MATLAB erases previous plot and makes new plot in same window
- To replot same data with additional plot type, select New Figure on right side of toolstrip, then click on new plot type. MATLAB opens new Figure Window and makes plot with new type in it



After plotting, MATLAB displays in Command Window the command it used to make plot. You can copy the command and paste it into a script to enable the script to create the same plot.