THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

ENG 101 Module #4
MATLAB Plots

Dr. Emil Jovanov
Electrical and Computer Engineering
University of Alabama in Huntsville
emil.jovanov@uah.edu
http://www.ece.uah.edu/~jovanov

This presentation 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

5.0

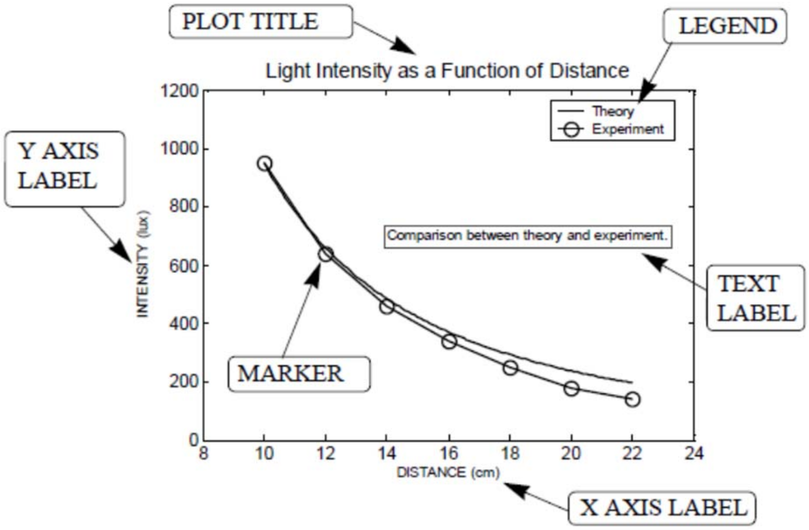


Figure 5-1: Example of a formatted two-dimensional plot.

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

`plot(y)`

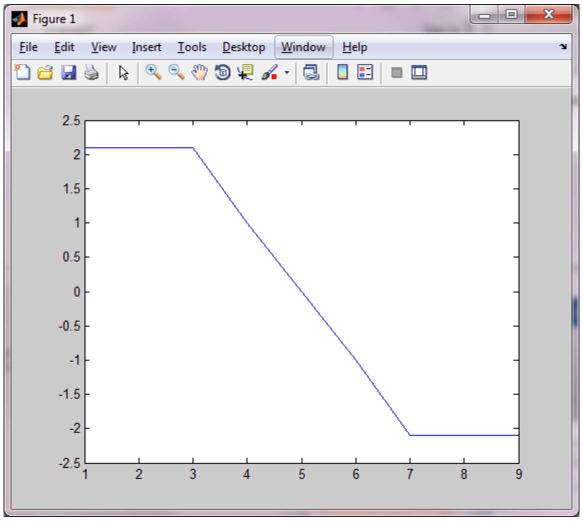
- Plots vector `y` on vertical axis, numbers 1 through N on horizontal axis (N = number of points in `y`)
- If there's a Figure Window, draws in it. Otherwise, creates a new Figure Window and draws in that

`plot(y)` default values

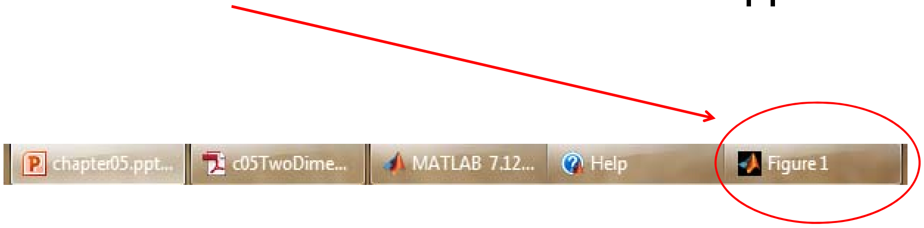
- Both axes linear
  - MATLAB chooses axis ranges so that end values are nice
- Points connected by straight lines
- No point markers
- Points and lines in blue

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)
```

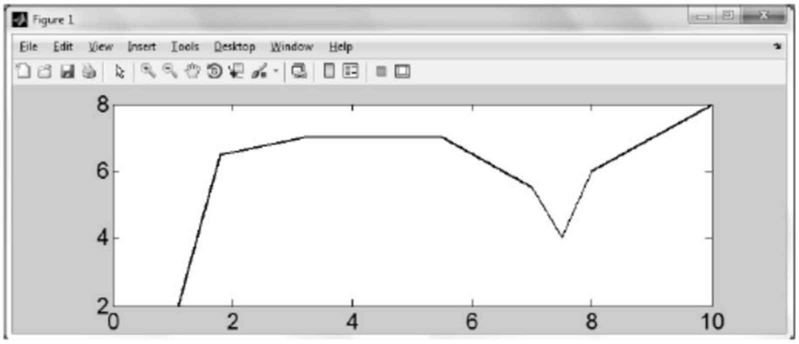
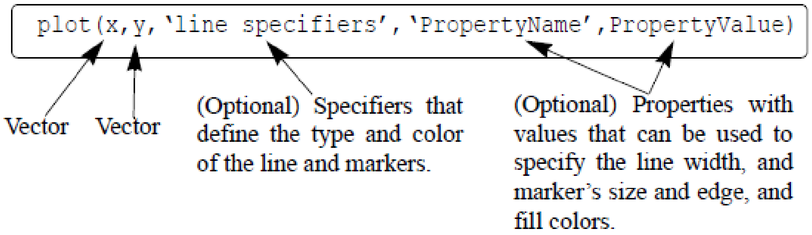


Figure 5-2: The Figure Window with a simple plot.

To use values other than defaults,



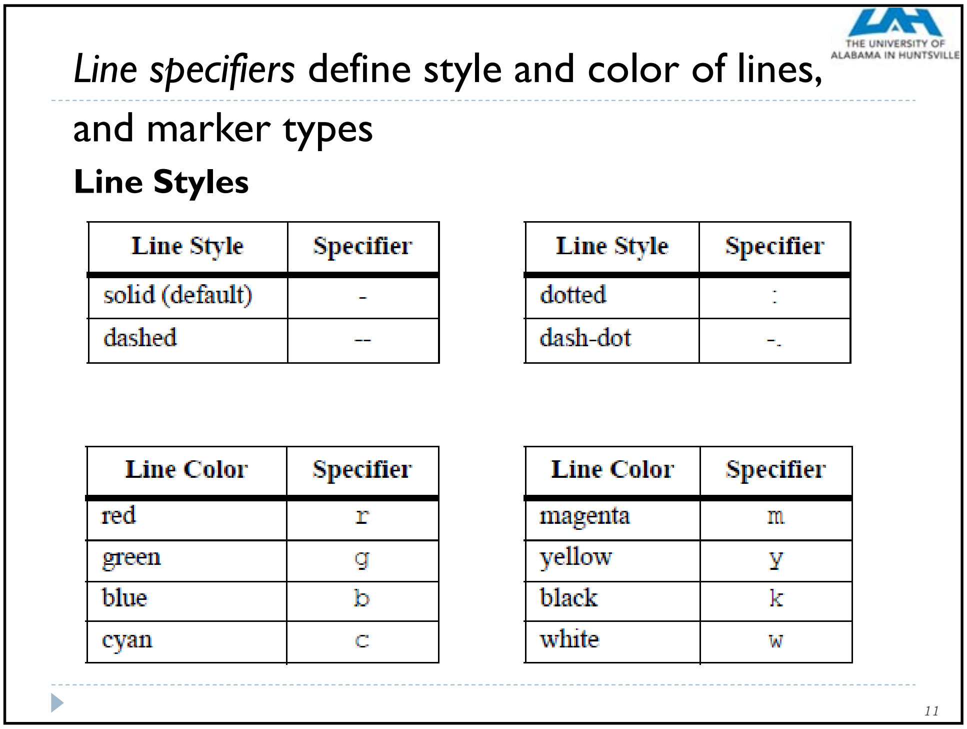
```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
```

Vector

Vector

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

(Optional) Properties with values that can be used to specify the line width, and marker's size and edge, and fill colors.



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

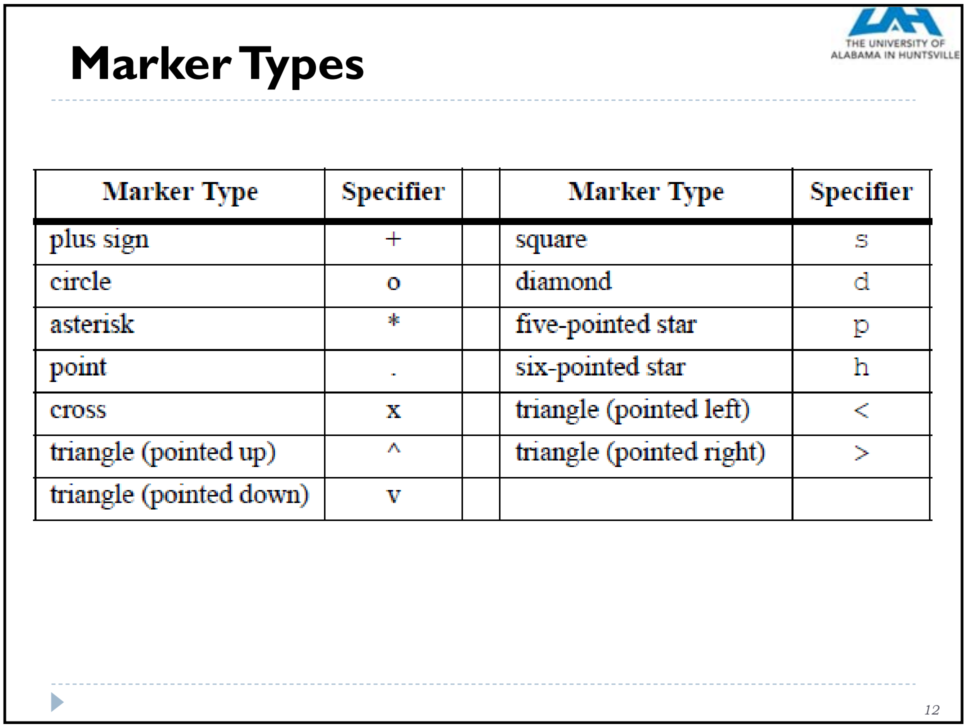
Line Styles

Line Style	Specifier
solid (default)	-
dashed	--

Line Style	Specifier
dotted	:
dash-dot	-.

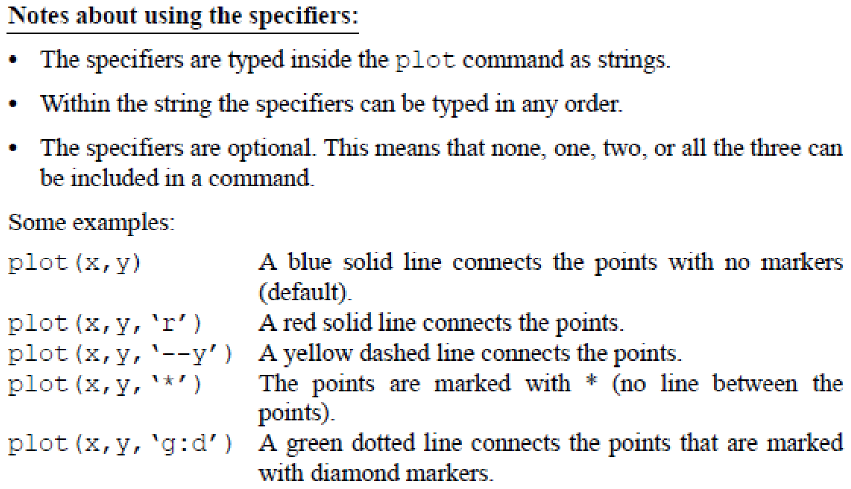
Line Color	Specifier
red	r
green	g
blue	b
cyan	c

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:

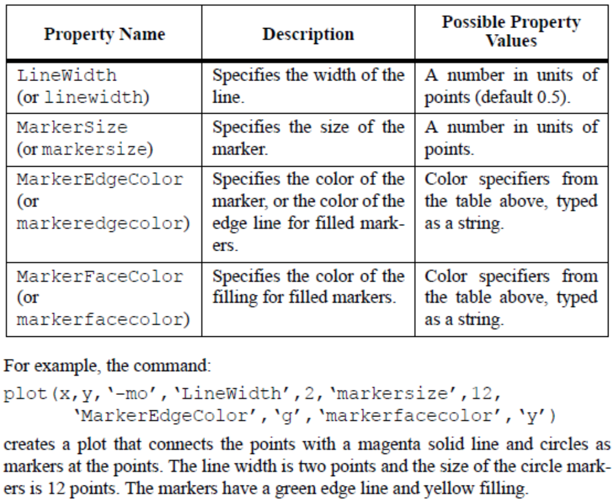
- The specifiers are typed inside the `plot` command as strings.
- Within the string the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all the three can be included in a command.

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

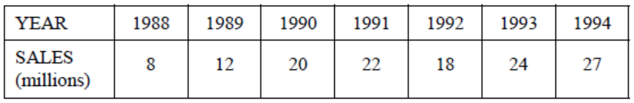


Property Name	Description	Possible Property Values
`LineWidth` (or `linewidth`)	Specifies the width of the line.	A number in units of points (default 0.5).
`MarkerSize` (or `markersize`)	Specifies the size of the marker.	A number in units of points.
`MarkerEdgeColor` (or `markeredgecolor`)	Specifies 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.

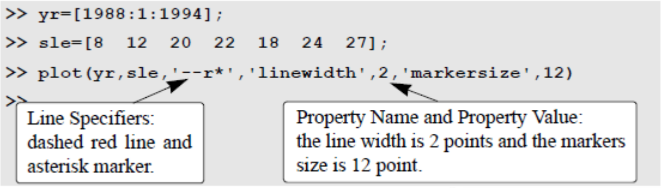
For example, the command:

```
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.



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 markers size is 12 point.

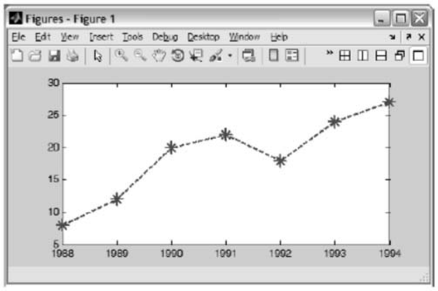


Figure 5-3: The Figure Window with a plot of the sales data.

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.

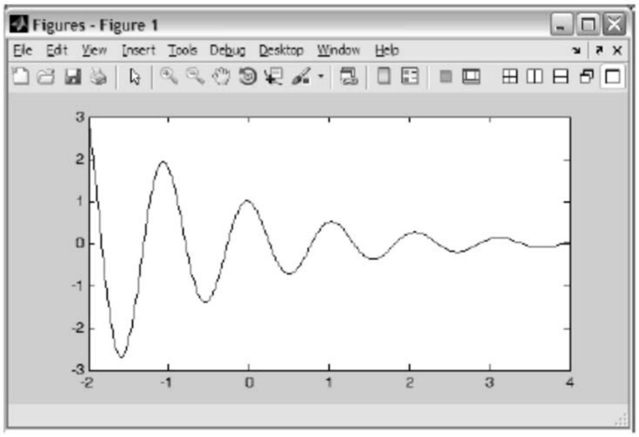


Figure 5-4: The Figure Window with a plot of the function $y = 3.5^{-0.5x}\cos(6x)$.

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

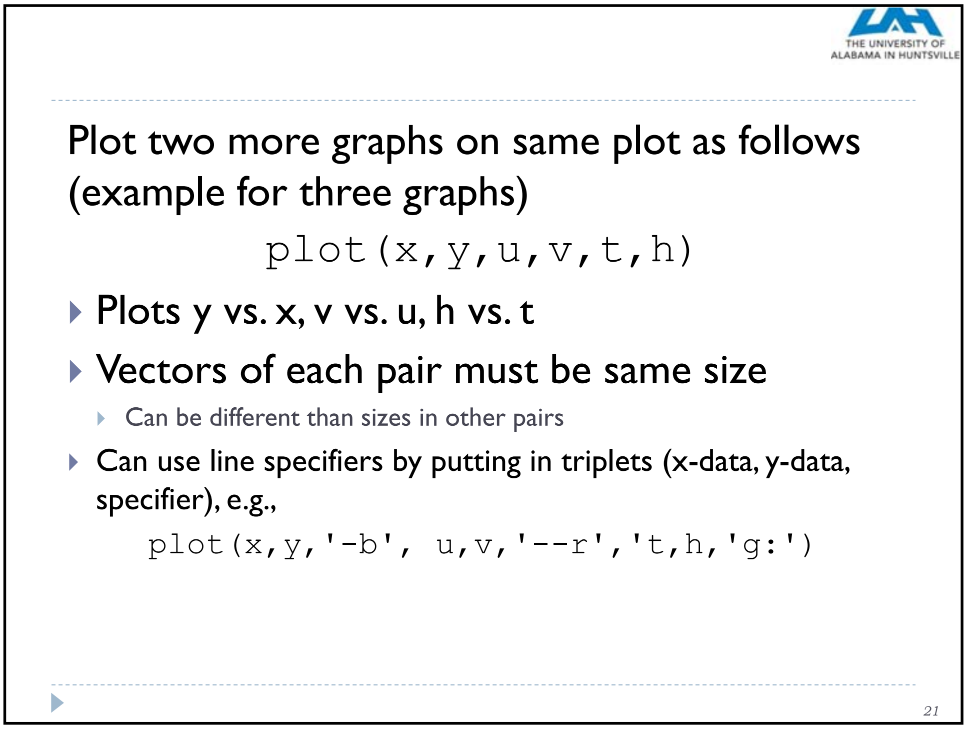
1. Click on Figure Window to make it current window
2. Select ALT+PRNTSCRN
3. Paste into other application

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

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

MATLAB can do this three different ways

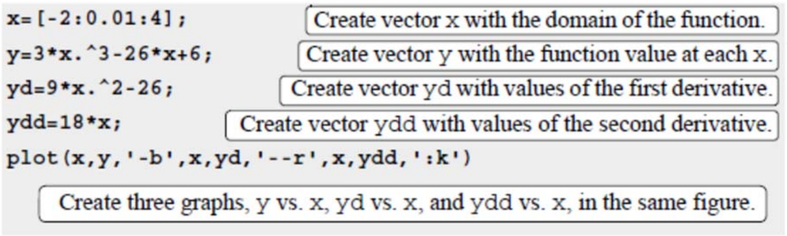


Plot two more graphs on same plot as follows (example for three graphs)

```
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];
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')
```

Create vector x with the domain of the function.

Create vector y with the function value at each x.

Create vector yd with values of the first derivative.

Create vector ydd with values of the second derivative.

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.

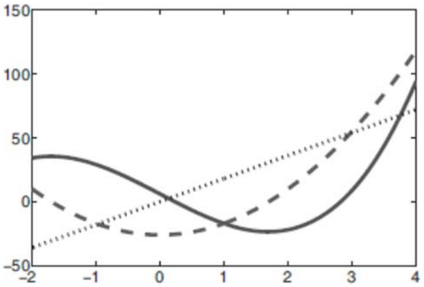


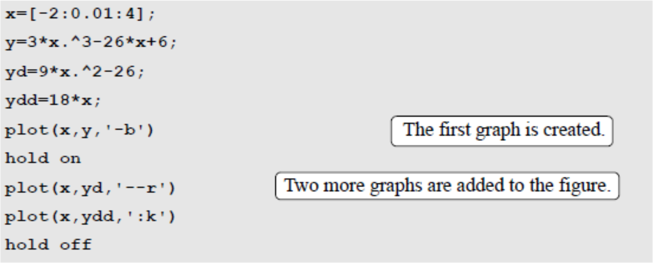
Figure 5-7: A plot of the function $y = 3x^3 - 26x + 10$ and its first and second derivatives.

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



```
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.

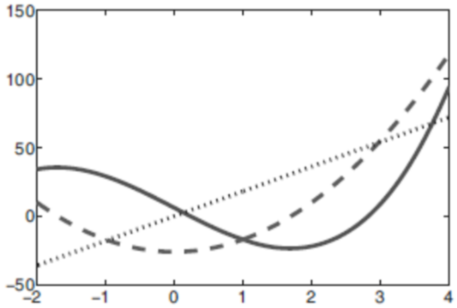


Figure 5-7: A plot of the function $y = 3x^3 - 26x + 10$ and its first and second derivatives.

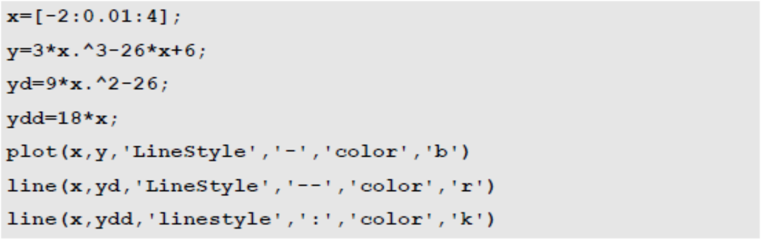
`line` command adds additional graphs to an existing plot

```
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



```
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')
```

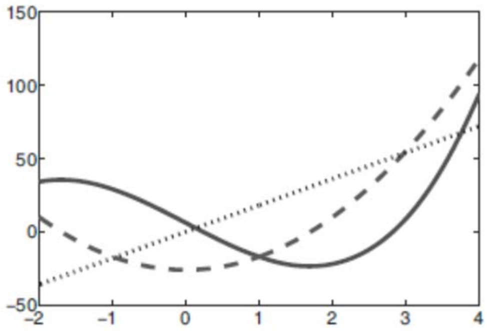
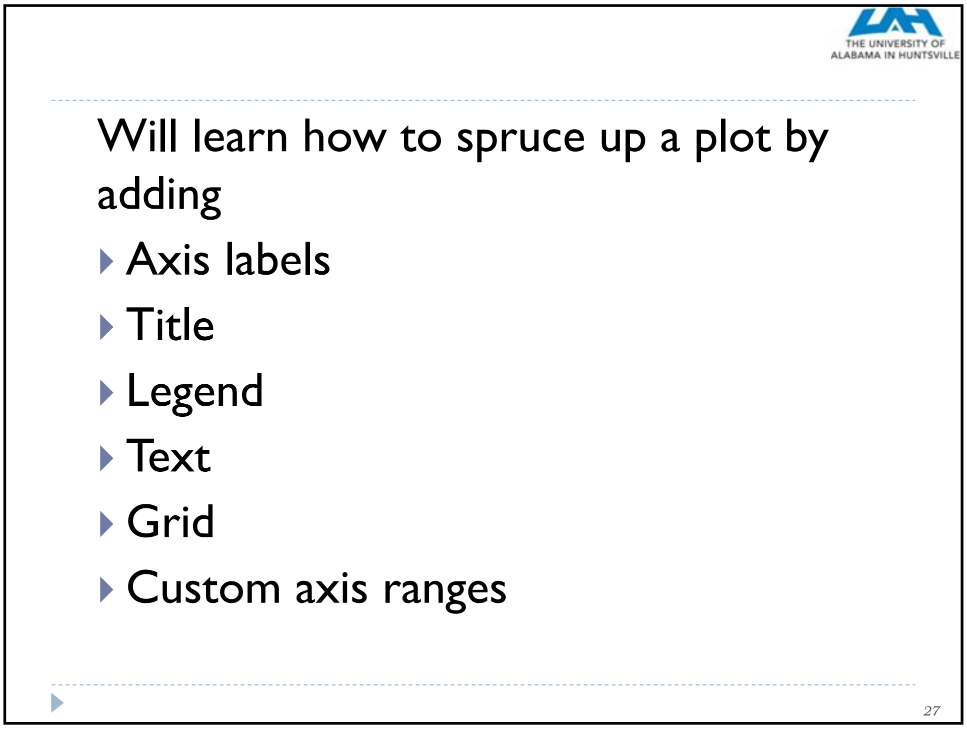
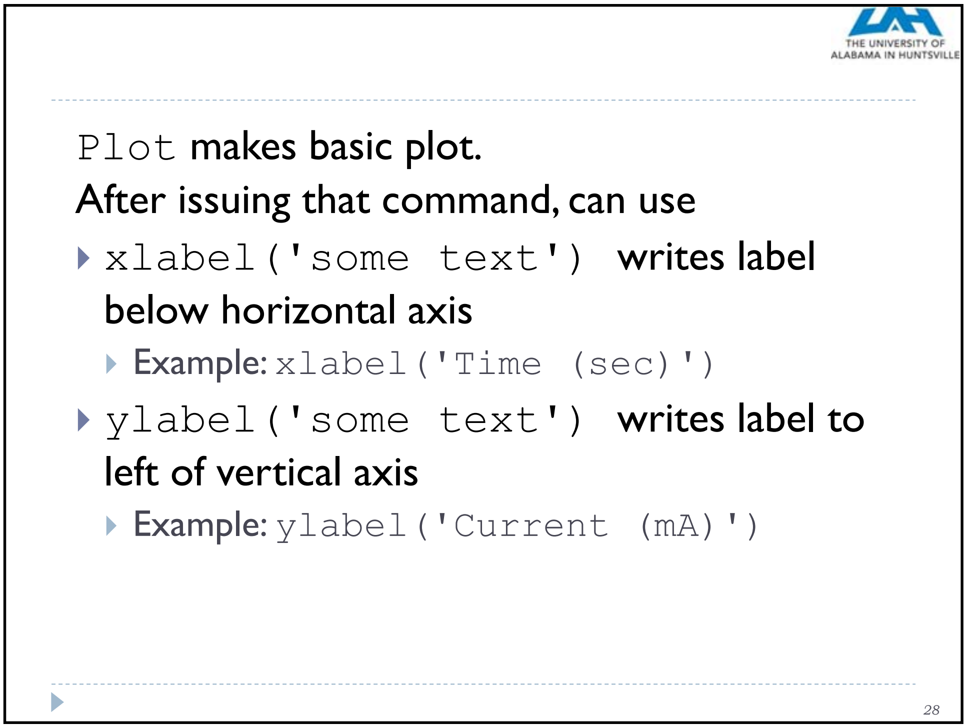


Figure 5-7: A plot of the function $y = 3x^3 - 26x + 10$ and its first and second derivatives.



Will learn how to spruce up a plot by adding

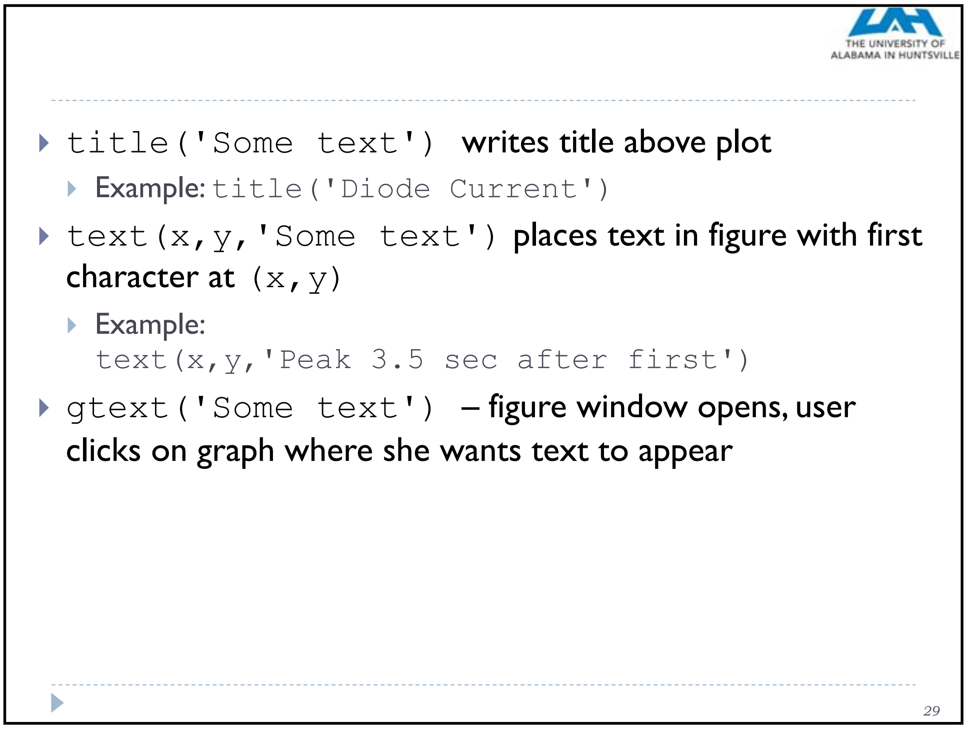
- Axis labels
- Title
- Legend
- Text
- Grid
- Custom axis ranges



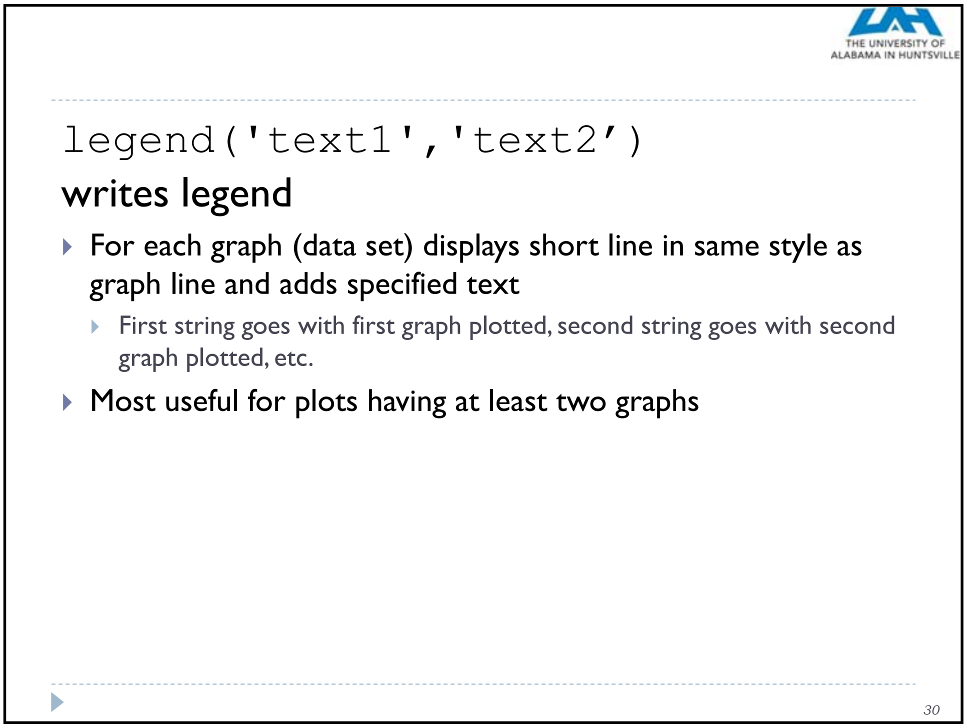
`Plot` makes basic plot.

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’)`

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

Formatting the text

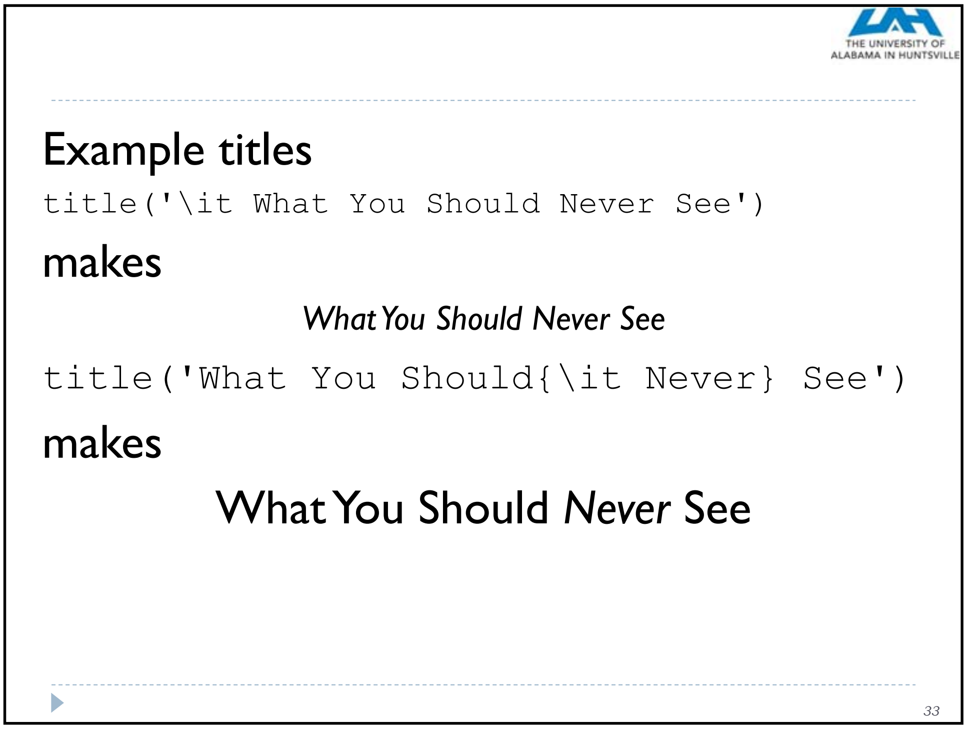
`xlabel, ylabel, title, text` **and** `legend` **commands:**

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



Example titles

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

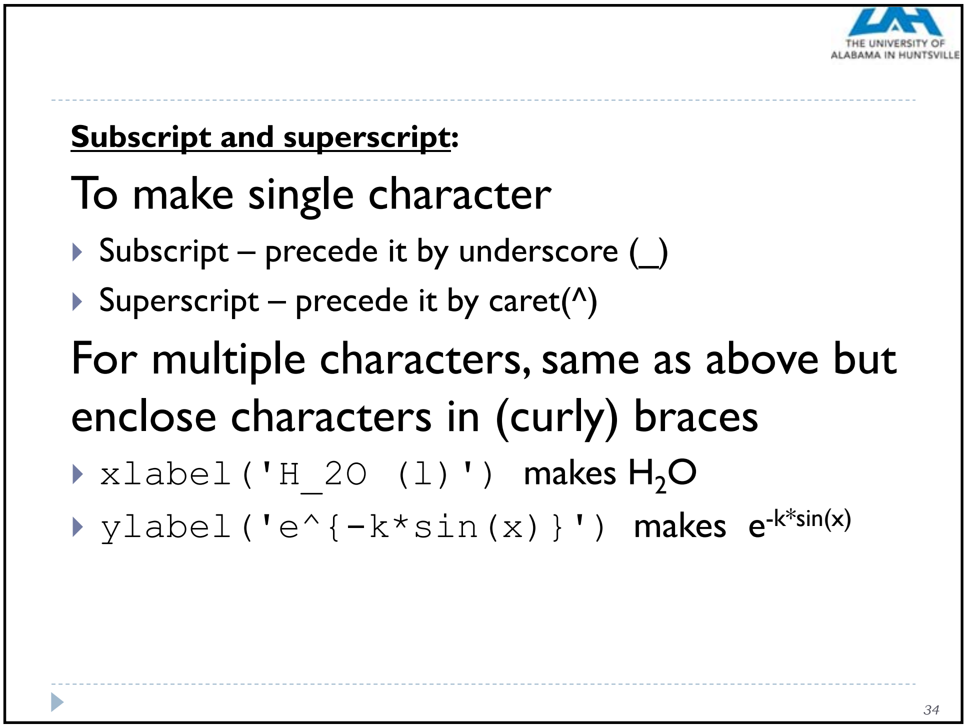
makes

What You Should Never See

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

makes

What You Should *Never* See



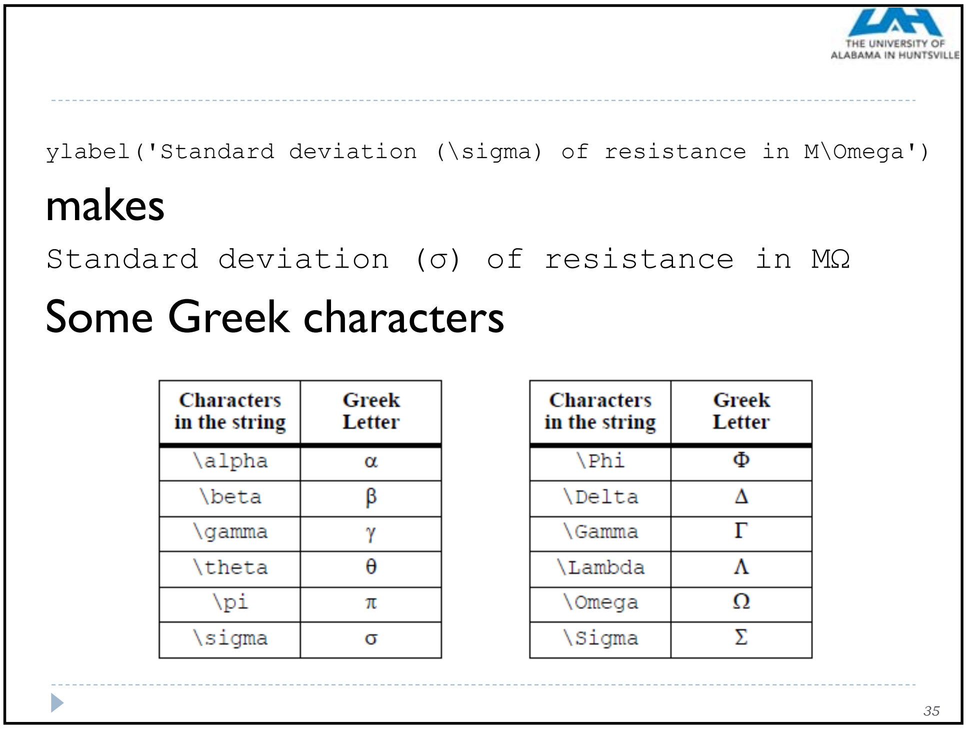
Subscript and superscript:

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



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

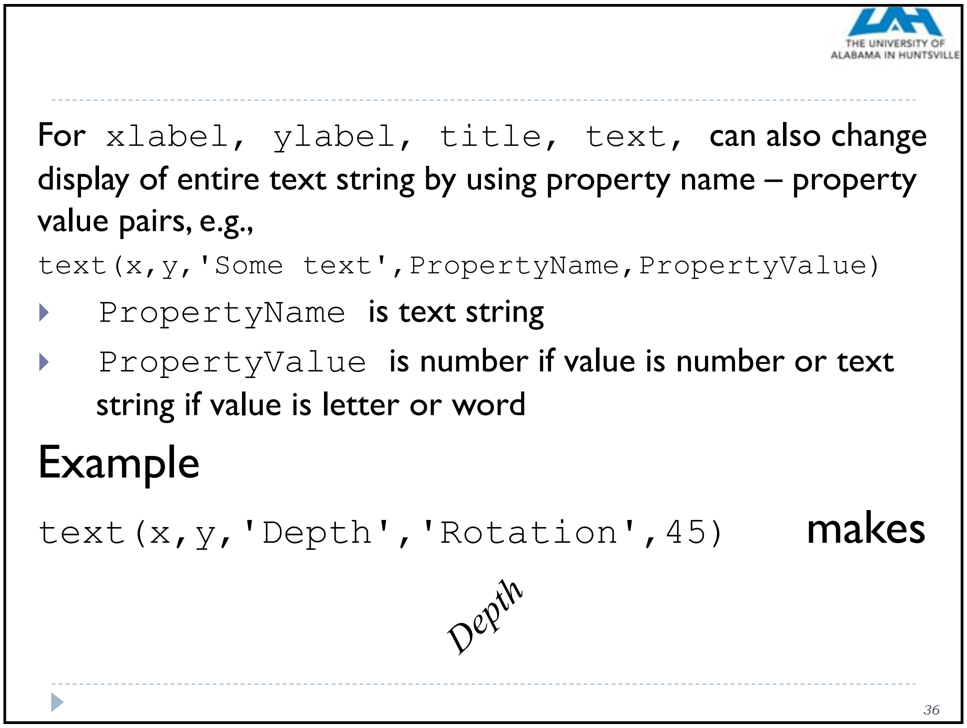
makes

`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)`

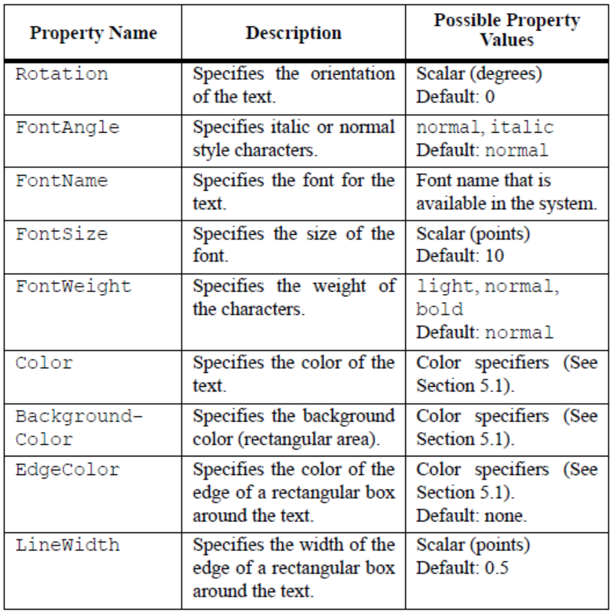
- `PropertyName` is text string
- `PropertyValue` is number if value is number or text string if value is letter or word

Example

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

Depth

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

Common `axis` variations are:

`axis([xmin xmax ymin ymax])`

- Sets limits of both axes

`axis equal`

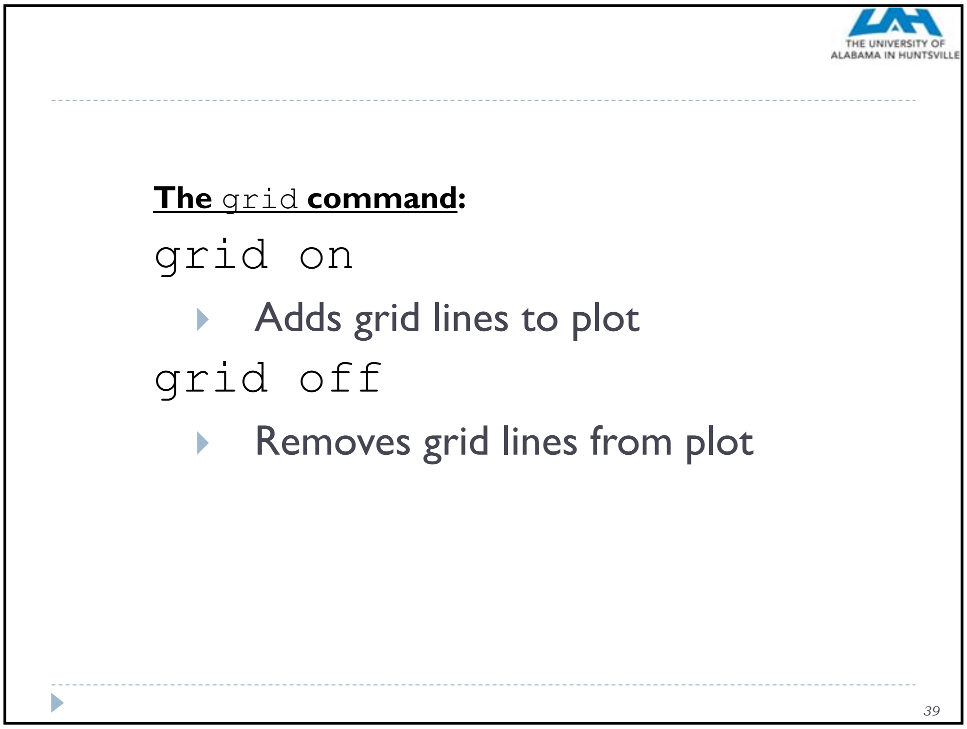
- Sets same scale for both axes

`axis square`

- Sets axis region to be square

`axis tight`

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



The `grid` command:

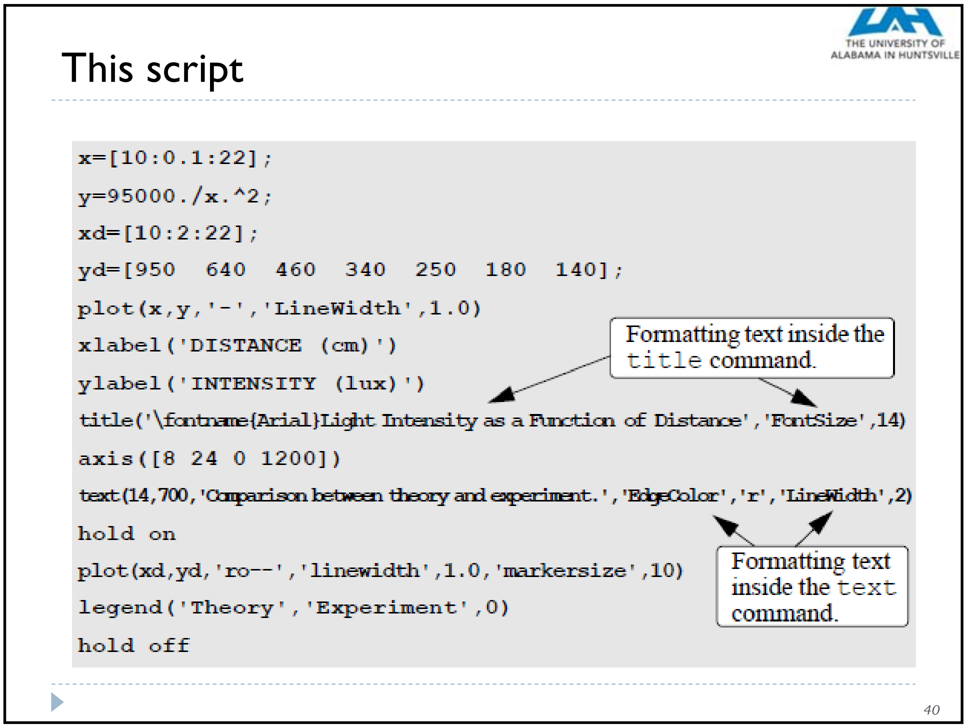
`grid on`

- Adds grid lines to plot

`grid off`

- Removes grid lines from plot

39



This script

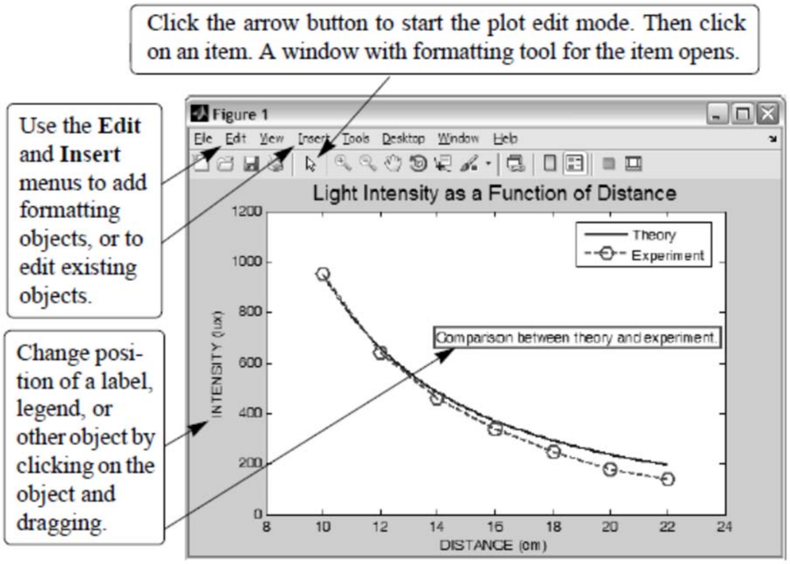
```
x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950  640  460  340  250  180  140];
plot(x,y,'-','LineWidth',1.0)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('\fontname{Arial}Light Intensity as a Function of Distance','FontSize',14)
axis([8 24 0 1200])
text(14,700,'Comparison between theory and experiment.','EdgeColor','r','LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',10)
legend('Theory','Experiment',0)
hold off
```

Formatting text inside the `title` command.

Formatting text inside the `text` command.

40

Makes this plot



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.

Figure 1
File Edit View Insert Tools Desktop Window Help
Light Intensity as a Function of Distance
Theory
Experiment
Comparison between theory and experiment.
INTENSITY (lux)
DISTANCE (cm)
0
200
400
600
800
1000
1200
8
10
12
14
16
18
20
22
24

Figure 5-8: Formatting a plot using the Plot Editor.

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

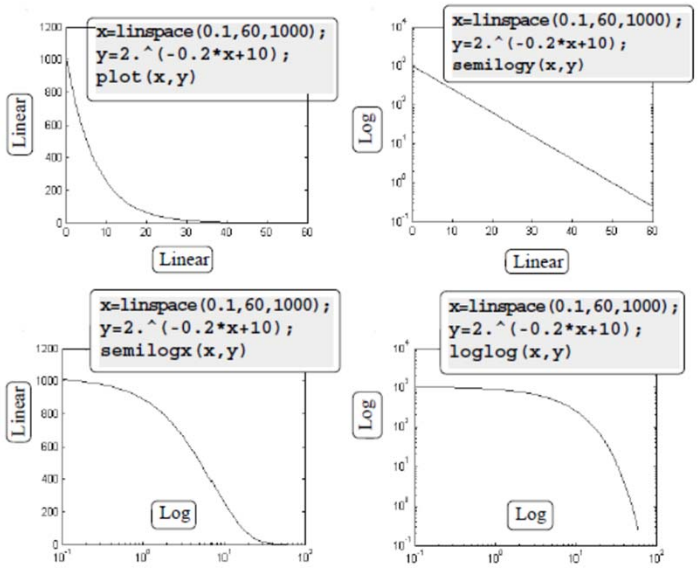
`semilogy(x,y)` Plots y versus x with a log (base 10) scale for the y axis and linear scale for the x axis.

`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

Example



```
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)
```

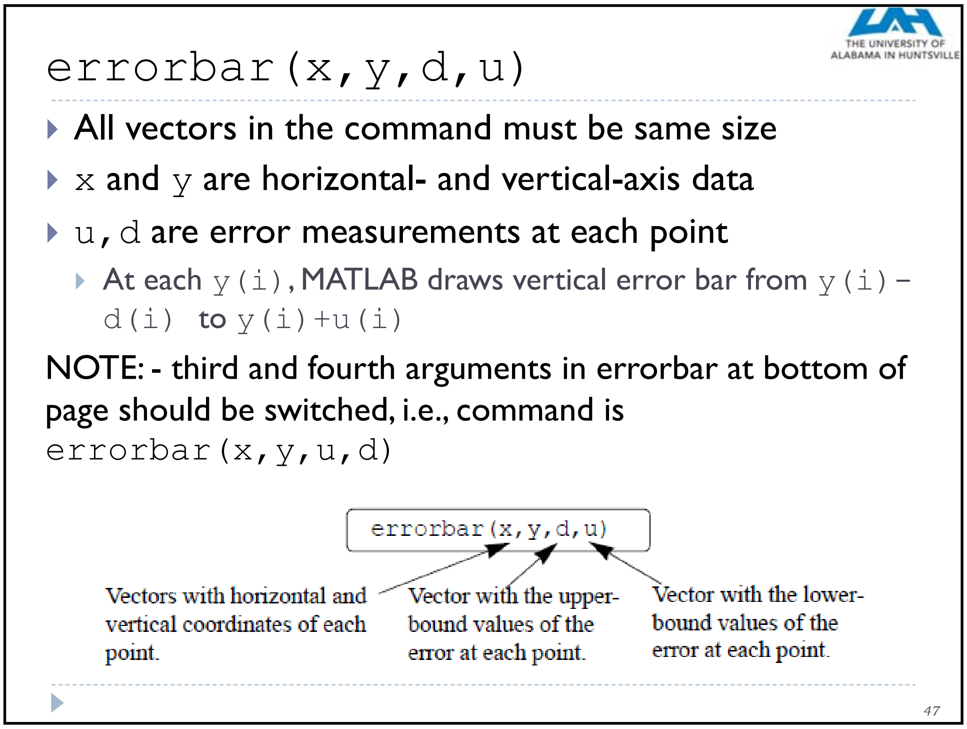
Figure 5-9: Plots of $y = 2^{(-0.2x+10)}$ with linear, semilog, and log-log scales.

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)`

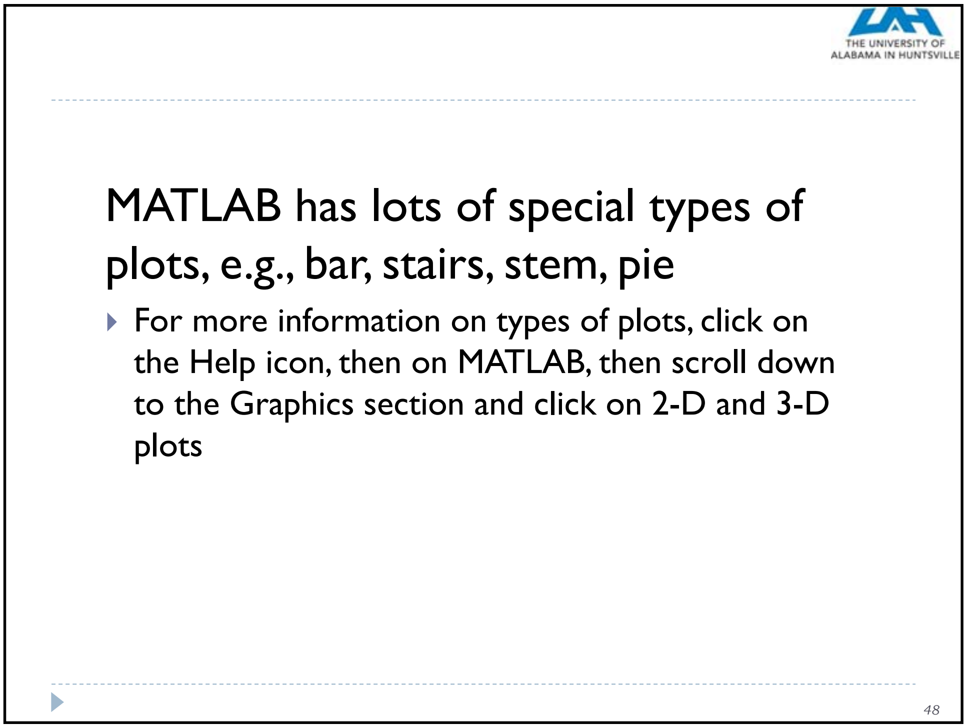
NOTE: - third and fourth arguments in errorbar at bottom of page should be switched, i.e., command is
`errorbar(x,y,u,d)`

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

Vectors with horizontal and vertical coordinates of each point.

Vector with the upper-bound values of the error at each point.

Vector with the lower-bound values of the error at each point.

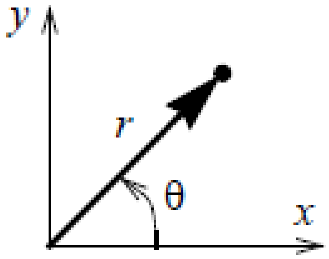


MATLAB has lots of special types of plots, e.g., bar, stairs, stem, pie

- 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

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

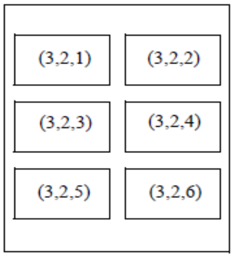
- *r* is distance from origin
- *ϑ* is angle from positive, horizontal axis. *ϑ* is positive in counterclockwise direction



`subplot(m,n,p)`

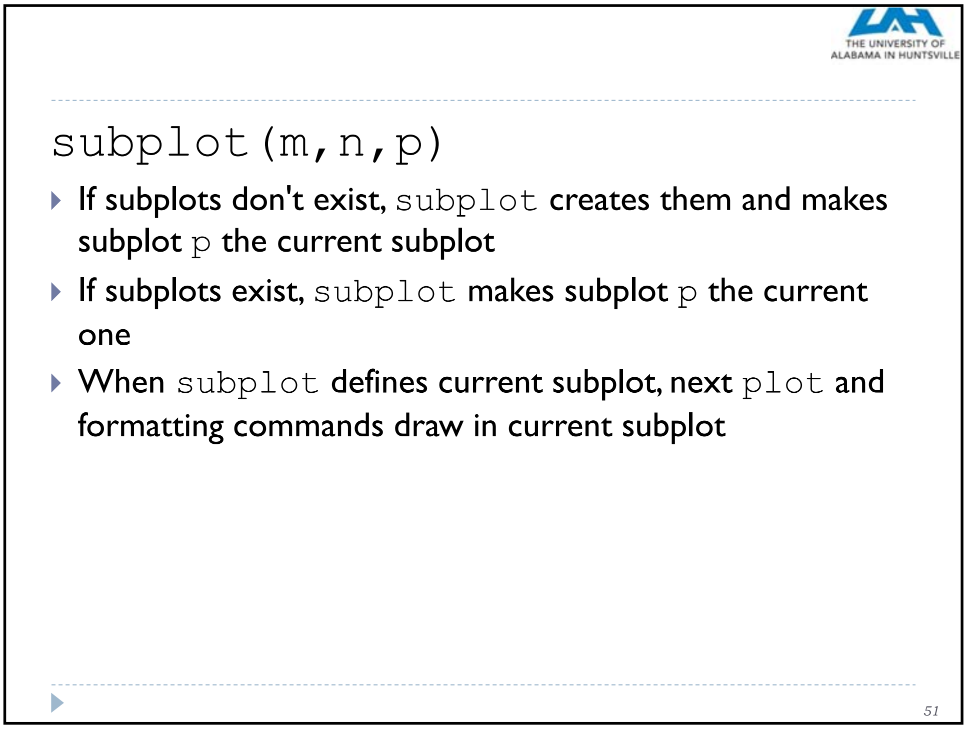
divides Figure Window into `m` rows and `n` columns of subplots

- 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



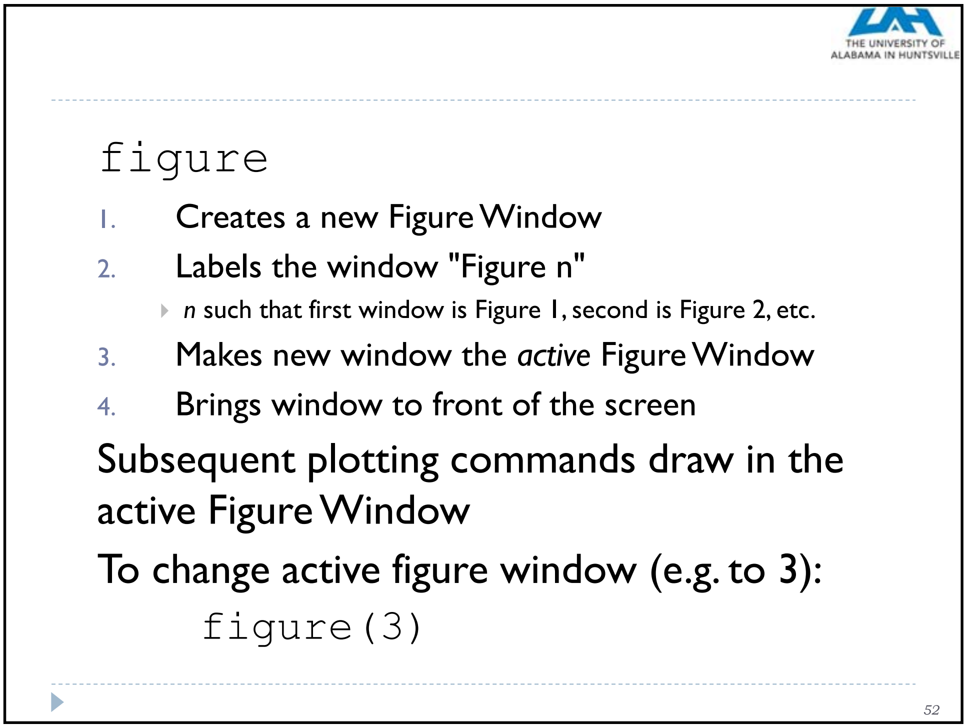
(3,2,1)	(3,2,2)
(3,2,3)	(3,2,4)
(3,2,5)	(3,2,6)

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



`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

To change active figure window (e.g. to 3):

```
figure(3)
```