MATLAB Workshop 14 - Plotting Data in MATLAB

Objectives: Learn the basics of displaying a data plot in MATLAB.

MATLAB Features:

graphics commands

Command	Action
`plot(x,y,symbol)`	creates a pop up window that displays the (*x*,*y*) data points specified on linearly-scaled axes with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. MATLAB automatically scales the axes to fit the data.
`semilogy(x,y,symbol)`	creates a pop up window that displays the (*x*,*y*) data points specified on a graph with the *y*-axis scaled in powers of 10 and the *x*-axis scaled linearly with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. MATLAB automatically scales the axes to fit the data.
`loglog(x,y,symbol)`	creates a pop up window that displays the (*x*,*y*) data points specified on a graph with both the x- and *y*-axes scaled in powers of 10 with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. MATLAB automatically scales the axes to fit the data.
`xlabel(xname)`	adds the text in the *string variable* **`xname`** below the *x*-axis.
`ylabel(yname)`	adds the text in the *string variable* **`yname`** below the *y*-axis.
`title(graphname)`	adds the text in the *string variable* **`graphname`** above the plot.
`axis('equal')`	forces equal-scaling on the *x*- and *y*-axes

graph symbol options

Symbol Options			
Symbol Color		Symbol	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green blue	*	star
b	blue	s	square
w	white	d	diamond
k	black	v	triangle (down)
		^	triangle (up)
		<	triangle (left)
		>	triangle (right)
		p	pentagram
		h	hexagram

- **Data display**

One of the most common activities that engage engineers is the display, analysis, and interpretation of data. How we choose to analyze and interpret data often depends upon the visual characteristics that we note from the initial display.

MATLAB has numerous graphics techniques that allow simple, easy visualization of data and functions. Basic 2D plotting capabilities are addressed in this workshop. If you are interested in more information than is provided here, you can obtain it directly from the MATLAB help facility by typing

`help graphics`	General description of graphics functions
`help graph2d`	More specialized description of 2D graphics functions
`help graph3d`	More specialized description of 3D graphics functions
`help specgraph`	Description of special graphic styles and displays
`help winfun`	Description of Windows^{TM} interface functions

MATLAB actually contains a complete *graphics toolbox* called *Handle Graphics* that allows one to have complete control over the look and feel of any graphics display. This includes font type and font size for text, location of text on the screen, embedding graphs within other graphs, etc, that the advanced graphics designer might want to control. If you are not satisfied with the basic graphics capabilities described in these workshops, you might want to learn more about *Handle Graphics*.

- **An Environmental Engineering problem**

Wastewater treatment to assure that water returned to the environment after use is as clean as or cleaner than the water source falls within the province of environmental and civil engineers. A common method to remove small diameter particulate matter from wastewater is to add a small amount of an agglomeration agent that causes the particles to cling together and form larger particles. Larger diameter particles will *settle* (fall to the bottom of a tank or pond) faster than small diameter particles.

An environmental engineer has obtained the laboratory measurements shown in the table below for settling of solids in a holding pond after adding an agglomeration agent. The experiment measured the mass of solids settled to the bottom of the pond as a function of time. The experiment was run in duplicate, i.e., twice, so there are two listings of (time,mass) data in the table.

Solids settling: agglomeration as a function of time										
Time, min	1	5	10	15	20	1	5	10	15	20
Mass, kg	0.12	4	16	37	61	0.19	3.2	14	33	58

- **Plotting data**

Ultimately, the engineer would like to develop an equation that describes the settled mass as a function of time based upon the data. As a first step in analyzing the information, she would like to display the data in a simple graph. Because she took this introductory course to MATLAB, the engineer has available a script to help with the experimental data analysis. Running the script produces a pop-up window with the graphical display at the right.

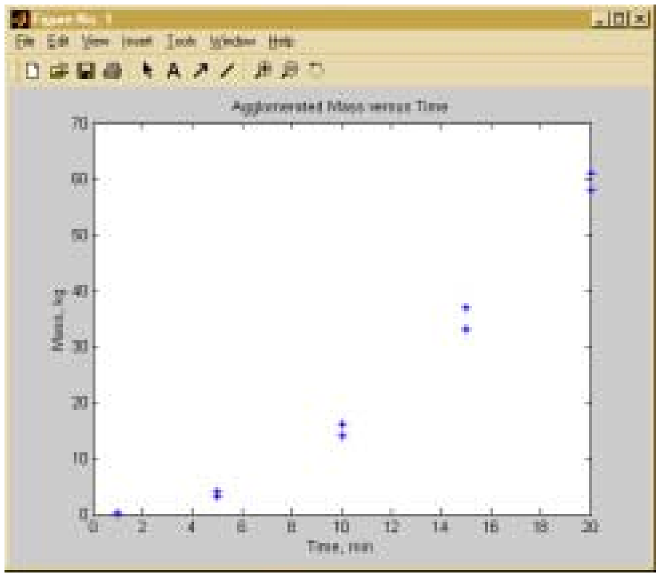


Figure No. 1
File Edit View Insert Tools Window Help
Agglomerated Mass versus Time
Mass, kg
Time, min

The graphical display has several elements that are common to all graphical data displays:

- Data shown as discrete points
- Symbol (and color) used to display data
- Appropriate range and domain for data display
- *x*-axis label (with units)
- *y*-axis label (with units)
- Graph title

Note that there is no line "connecting the dots."

(1) MATLAB commands to display data on a graph.

Create the graph of agglomeration mass versus time directly in the Command Window by entering the following commands:

```
» % create (x,y) data points in separate vectors
» xdat = [1,5,10,15,20,1,5,10,15,20];
» ydat = [0.12 4 16 33 61 0.19 3.2 14 37 58];
»
» % display data on linear graph
» plot(xdat,ydat,'b*')
    a pop-up window should appear with the data points plotted
» xlabel('Time, min')
» ylabel('Mass, kg')
» title('Agglomerated Mass versus Time')
    the axis labels and title should also now appear
```

After creating a set of (*x*,*y*) data points as separate *x*- and *y*-vectors, the data can be plotted in an annotated graph using the **`plot`**, **`xlabel`**, **`ylabel`**, and **`title`** MATLAB commands. Focusing on the graphical display commands:

`plot(x,y,symbol)` - creates a pop up window that displays the (*x*,*y*) data points specified on linearly-scaled axes with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. Note that MATLAB automatically scales the axes to fit the data.

`xlabel(xname)` - adds the text in the *string variable* **`xname`** below the *x*-axis.

`ylabel(yname)` - adds the text in the *string variable* **`yname`** beside the *y*-axis.

`title(graphname)` - adds the text in the *string variable* **`graphname`** above the plot.

(2) Create a function to display data on a linear graph.

Data plotting occurs frequently. Rather than typing the commands repetitively to create the display over and over to display such graphs, having a function that can display the graph with associated annotation (title and axis labels) would save time and effort.

Following the design algorithm for functions:

`function linearplot`

(1) Function to plot a set of (*x*,*y*) data with specified symbol and annotate plot with *x*-axis label, *y*-axis label, and title.

(2) Produces a pop-up window with plot but returns no values.

(3) Requires

`xdata`	vector with *x*-data points
`ydata`	vector with *y*-data points corresponding to **`xdata`** points
`symbol`	string value specifying symbol for plot
`xname`	string value specifying text for *x*-axis label
`yname`	string value specifying text for *y*-axis label
`graphname`	string value specifying text for graph title

(4) Algorithm

`plot`, **`xlabel`**, **`ylabel`**, and **`title`** commands - in order

Create the function **`linearplot`** and save it in your current workspace. Be sure to have appropriate header comments and a variable dictionary.

(3) Test **`linearplot`** using *hardwired* values.

If your function is working properly, it should be able to reproduce the graph currently in the pop-up window if it is given the same values. Close the pop-up window by clicking on the **X** in the upper right hand corner. Since the **`xdat`** and **`ydat`** vectors are already available, test your function by entering

```
»
» % test linear plot function
» linearplot(xdat,ydat,'b*','Time, min', 'Mass, kg', ...
'Agglomerated Mass versus Time')
```

The preceding command assumes that you have designed your function with the input variables in the same order as shown here. If not, change the command to reflect the order in which you listed the variables. The figure should again appear, completely annotated.

(4) Test **`linearplot`** using variable values.

The previous test used "hardwired" values. This is fine if you are working from the Command Window. However, if you are developing a script to present and analyze data (does this provide a hint as to where we are going?), you will not be using the Command Window to call the function - it will be called from the script using variable names that are holding the desired information.

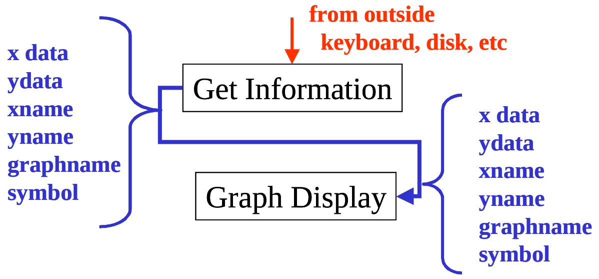
To test this, close the pop-up window and enter the following in the Command Window

```
»
» % test linear plot function with variable parameters
» symbol = 'b*';
» xname = 'Time, min';
» yname = 'Mass, kg';
» graphname = 'Agglomerated Mass versus Time';
» linearplot(xdat,ydat,symbol,xname,yname,graphname)
```

The figure should again appear, completely annotated.

(5) Create a function to ask the user for graph display information.

The information flow in a script to display data looks something like the diagram at the right. We already know that the *Graph Display* function requires the values shown at the right function box. We have supplied those values through the Command Window. However, a more useful script needs to get the values from the user. In particular, the information needed from the user includes:



- The (*x*,*y*) data set. This is generally contained in a data file!
- The graph labels: *x*-axis, *y*-axis, and title.
- The symbol to use for display of the data.

Use the function design methodology to design a function to get the necessary information to provide to linearplot. Hint: have you already created a function that extracts values from a data file? Rhetorical question - the answer is yes! Can you adapt that function to this need? What will you name your function?

The only part of this function that would be new is the appropriate symbol (type and color) to use for the data display. The two parts of the symbol are defined individually according to the table at the right. Note that, according to this table, we have been specifying a blue star as the symbol for the agglomeration versus time graphs. The order in which they are specified to the **`plot`** function does not matter. That is, **`'b*'`** is the same as **`'*b'`**.

Symbol Options			
Symbol Color		Symbol	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green blue	*	star
b	blue	s	square
w	white	d	diamond
k	black	v	triangle (down)
		^	triangle (up)
		<	triangle (left)
		>	triangle (right)
		p	pentagram
		h	hexagram

Given the large number of choices, your function should display a *menu* to ask for the color choice and, depending upon the response, use an **`if/elseif/else`** or **`switch/case`** structure to assign the appropriate code, e.g., **`'b'`**, to the variable **`symbolcolor`**. A similar process can be used to assign the symbol type code to the variable **`symboltype`**. Finally, the two can be combined (recall Workshop 3 on vectors) by the assignment statement

```
symbol = [symbolcolor, symboltype];
```

Design and code a data input function for the graph display script. How can you test your function? Hint: Create a data file that has the agglomeration data as (*x*,*y*) pairs and see whether you can reproduce the pop-up window display with successive calls to your information input and linear graph display functions.

- **Logarithmically-scaled plots**

The ultimate goal of plotting data is to find a relationship (equation) that relates the dependent variable to the independent variable. Workshop 15 presents techniques to find such relationships. Three functional forms bear special mention here. The first has already been used. A linearly-scaled graph, such as produced by **`plot`**, will suggest a straight-line fit, i.e.,

$$y = a_1 x + a_0$$

if all of the data points fall on a relatively straight line. a_1 is the slope of the line and a_0 is the y-intercept for the line. The agglomeration plot versus time has obvious curvature when plotted with linear scales - suggesting that a different functional form to relate the dependent and independent variables might be better.

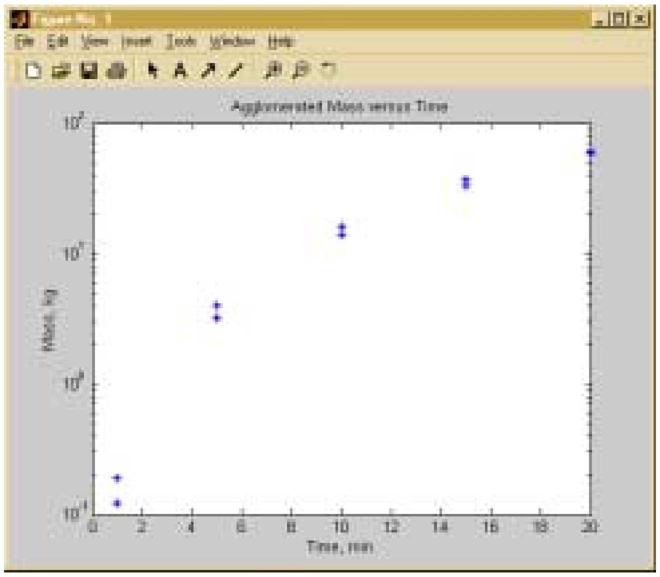
The first alternative that engineers usually consider is an exponential relationship, given by

$$y = ae^{bx}$$

where *a* and *b* are constants. This relation is called exponential because the independent variable appears as an exponent in the equation. Note that small changes in *x* can cause large changes in *y* because of the exponential relationship. The hypothesis that the best relation is exponential can be easily tested in MATLAB by plotting the data on a semi-log plot. The phrase semi-log comes from taking (natural) logarithms of both side of the exponential relationship, yielding

$$\ln(y) = bx + \ln(a)$$

Plotting ln(*y*) versus *x* should produce a straight-line if this equation holds. Alternatively, we could plot the (*x*,*y*) data directly on a graph that has the *y*-axis scaled in powers of ten (why?). If we do this for the agglomeration data, we get the pop-up window shown at the right. Note the scaling for the *y*-axis goes in powers of 10 while the *x*-axis is linear. When looking at graphical data, always check the axis scaling!!!



A graph with the *y*-axis scaled in powers of ten and the *x*-axis scaled linearly is called a ***semi-log graph***.

(6) Create a function to display data on a *semi-log graph*.

MATLAB has a simple command that will plot data on a semi-log graph:

`semilogy(x,y,symbol)` - creates a pop up window that displays the (*x*,*y*) data points specified on a graph with the *y*-axis scaled in powers of 10 and the *x*-axis scaled linearly with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. Note that MATLAB automatically scales the axes to fit the data.

Annotations appear with the **`xlabel`**, **`ylabel`**, and **`title`** commands as before. Create a function, **`semilogplot`**, that will create an annotated semi-log plot of data. Test your function using the agglomeration data to produce the semi-log plot above.

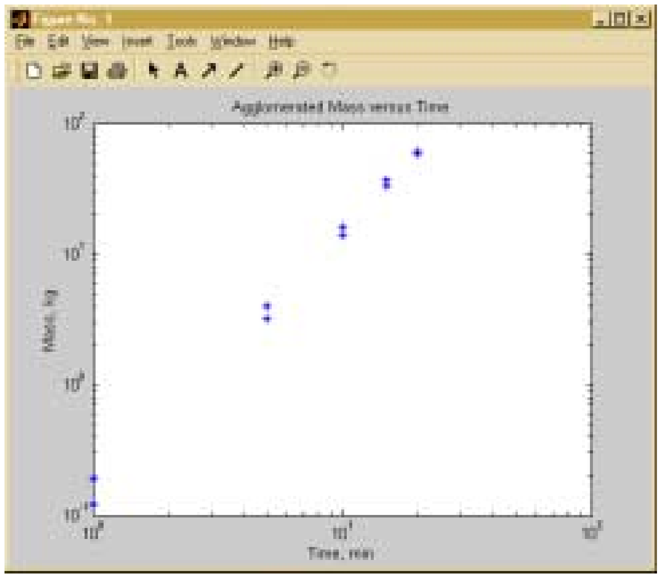
The second alternative that engineers usually consider is a power-law relationship, given by

$$y = ax^b$$

where a and b are constants. This relation is called power-law because the independent variable is raised to a power in the equation. The hypothesis that the best relation for experimental data is power-law can be easily tested in MATLAB by plotting the data on a log-log plot. The phrase log-log comes from taking (natural) logarithms of both side of the power-law relationship, yielding

$$\ln(y) = b\ln(x) + \ln(a)$$

Plotting ln(y) versus ln(x) should produce a straight-line if this equation holds. Alternatively, we could plot the (x,y) data directly on a graph that has both the x- and y-axes scaled in powers of ten (why?). If we do this for the agglomeration data, we get the pop-up window shown at the right. Note the scaling for both the x- and y-axes goes in powers of 10. Again, when looking at graphical data, always check the axis scaling!!!



A graph with both the x- and y-axes scaled in powers of ten is called a ***log-log graph***.

<u>(7) Create a function to display data on a *log-log graph*.</u>

MATLAB has a simple command that will plot data on a semi-log graph:

`loglog(x,y,symbol)` - creates a pop up window that displays the (x,y) data points specified on a graph with both the x- and y-axes scaled in powers of 10 with the symbol (and color) specified in the *string variable* **`symbol`**. The data points are supplied as separate **`x`** and **`y`** vectors. Note that MATLAB automatically scales the axes to fit the data.

Annotations appear with the **`xlabel`**, **`ylabel`**, and **`title`** commands as before. Create a function, <u>**`loglogplot`**</u>, that will create an annotated log-log plot of data. Test your function using the agglomeration data to produce the log-log plot above.

<u>(8) Which functional form best fits the agglomeration data? Why?</u>

Exercises:

1. Create script that will perform the following
   - a Display a header message when first run;
   - b Get plot information (data, annotation, symbol) ;
   - c Ask the user to select what type of plot (linear, semi-log, log-log);
   - d Display the requested plot;
   - e Ask the user whether the plot is ok and if not, cycle back to (c) for another plot;
   - f Repeat the cycle (c,d,e) until the user is satisfied ;

g Ask the user whether to plot another data set and if yes, cycle back to (b) ;

2. *Parametric equation for a circle*. The parametric equation for a circle is $x = r\cos(\theta)$ and $y = r\sin(\theta)$ where r is the radius and θ is the angle of rotation counter-clockwise from the positive x-axis. Use **`linspace`** to create an angle vector, **theta**, with 200 equally spaced values in the range (0, 2π). Compute the corresponding x- and y-vectors for $r = 2$. Use **`linearplot`** to display the resulting (x,y) plot.

   Doesn't look much like a circle does it? Why? (Hint: look at the scales on the x & y axes.) Correct the problem by typing the command

```
axis('equal')
```

   This forces the scales on the x and y axes to be the same.

3. Plot each of the following relations first with **`linearplot`** and then with the appropriate plot to obtain a straight-line result. Use linspace to create the x-vector and the equation to create the y-vector.

   a. $y = 1.25e^{-x/25}$ $(0 \le x \le 100)$

   b. $x = 1.25e^{2.5y}$ $(10 \le x \le 50)$

   c. $y = 2x^{1.5}$ $(0 < x \le 100)$

   d. $y = 3x^2 - 2x + 1$ $(-10 \le x \le 10)$

   e. $y = 2\sin(\pi x/2)e^{-2x}$ $(0 \le x \le 10)$

4. As part of a mechanical engineering laboratory, you obtained the following stress-strain data on a new material. Use MATLAB to plot the data with red triangles.

Strain, (in/in)	Load, lb_f	Strain, (in/in)	Load, lb_f	Strain, (in/in)	Load, lb_f
0.0002	900	0.0024	7350	0.020	9150
0.0004	1600	0.0028	7850	0.040	8950
0.0006	2350	0.0032	8200	0.060	8100
0.0008	3150	0.0036	8400	0.080	7250
0.0010	3900	0.0040	8600	0.10	6300
0.0012	4650	0.0044	8700	0.12	5150
0.0014	5200	0.0048	8800	0.14	3950
0.0016	5750	0.0052	8900	0.16	2100
0.0018	6300	0.0070	9100	0.17	500
0.0020	6700	0.0100	9200	Fracture	

Recap: You should have learned

- How to create simple linear, semi-log, and log-log plots of data using MATLAB.