# MATLAB GRAPHING

# Using MATLAB 6 (Release 12)

As with paper and pencil graphing, it is often necessary to plot data in a variety of ways in order to see if a common type of graphing scheme will cause the data to lie on a straight line (as opposed to a curve). When this occurs, the form of an empirical function that describes the data can be determined. Data that can be represented by a straight line on a rectilinear plot are said to be *linear*, and are described by an empirical function of the y = mx + b form. Data that produce a straight line on a log-log plot are represented by a *power* function of the  $y = b x^m$  form. Data that are best fit by a straight line on a semilog plot are said to be *exponential*, and are described by an empirical function of the form  $y = b e^{mx}$  when the y-axis is logarithmic. Once the form of the empirical function is known, the next order of business is to determine the constants *m* and *b* for the equation of the line that represents the data. Here we will outline the basic steps used to produce a plot and determine the equation constants using MATLAB. The general procedure is given below, and then illustrated with a guide for the three types of plots (rectilinear, loglog, and semilog) mentioned above. This style of presentation is somewhat repetitive, but gives an orderly account of the syntax and output interpretation required for MATLAB use.

General Graphing Procedure Steps

- 1. Enter the data.
- 2. Plot the data as points.
- 3. Adjust axes limits, if desired.
- 4. Insert the grid lines.
- 5. Label the axes.
- 6. Determine the equation of the best-fit line.
- 7. Plot the best-fit line.
- 8. Copy the figure to a Word document
- 9. Add the title in Word, placing it below the *x*-axis label.

To obtain maximum benefit from the time you invest in reading this document, open MATLAB and enter the indicated commands (designated by ">>") in the command window as you read.

#### **ENTERING THE DATA**

Suppose you had run an experiment and acquired some "time" and "velocity" data.

Time t, s :	0.0	10.0	20.0	30.0	40.0	50.0	60.0
Velocity V, ft/s:	0.0	41.3	77.4	125	165	197	248

This data can be entered in the MATLAB command window as two separate vectors (here called Time and Velocity),

» Time = [ 0, 10, 20, 30, 40, 50, 60 ]; » » Velocity = [ 0.0, 41.3, 77.4, 125, 165, 197, 248 ];

or as a single matrix with each row or column of the matrix containing the time and velocity data (here the matrix called MyData has time values stored as row 1, and velocity values stored as row 2).

» MyData = [0, 10, 20, 30, 40, 50, 60; 0.0, 41.3, 77.4, 125, 165, 197, 248]

MyData =

0	10.0000	20.0000	30.0000	40.0000	50.0000	60.0000
0	41.3000	77.4000	125.0000	165.0000	197.0000	248.0000

When entering data at the command window, remember that commas or spaces can be used to delimit (separate) the elements within a row of a MATLAB matrix, and semicolons are used to delimit the rows.

#### RECTILINEAR PLOTS

MATLAB syntax for a rectilinear plot of points is

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

where x contains the independent variable values, y contains the dependent variable values, and the 'o' designates that the plotted values be represented by lower case "o" on the plot. For the example at hand, entering

» plot(Time,Velocity,'o')
»

will produce the plot that is shown below in the MATLAB figure window. If this is the first time that you have entered a plotting command in the current workspace in the command window, the figure window will open automatically, and your screen view will be transferred to the now operative figure window. After you move back to the command window, upon entering subsequent plotting or plot editing commands there, the command window will remain the operative window. Then, in order to see what has been done to your plot in the figure window with an entered command, you can select "Figure Window" from the "Window" pull-down menu. Of course, you can also move between the windows by just clicking on one when they are both visible on your screen. When you are ready to return to the command window, just select "Command Window" from the "Window" pull-down menu or click on the command window if it is in view.



Once you have this plot in the figure window, you can use menu options to edit the plot. The first step is to enable the plot editing features by selecting the "Edit Plot" option from the "Tools" pull-down menu, or by clicking on the upward arrow that points to the left in the toolbar. If your figure window toolbar is not in view, select "Figure Toolbar" from the "View" pull-down menu.

Now, in the current plot, MATLAB has set the axes limits by default, but these can be changed. In the figure window, highlight the axes of the plot and select "Axes Properties" from the "Edit" pull-down menu. In the dialogue box that appears with the "Scale" tab open, there are options to 1) adjust the limits of both axes; 2) adjust tick marks and labels for calibration of the axes; 3) change the axes scales between linear or log and normal or reverse; and 4) insert grid lines. Under the "Style" tab, you have options to change the calibration and labeling fonts and properties. The "Labels" tab offers axes labels and the figure title, but these options are also available under the "Insert" pull-down menu of the figure window. Options under the other tabs in "Axes Properties" will not be needed for EF1015 work, but may serve you well in upper classes. With regard to placing a title on the plot, be aware that MATLAB will put the title at the top of the figure by default. This is *not* acceptable format, but moving the title may prove troublesome. The easiest fix is to copy your figure to a Word document, and add the title there. Before copying the figure, it is useful to check the "Copying Options"

(under "Edit" in the figure window); use the "Figure background color" options to remove the gray background that often accompanies plots copied from MATLAB to Word. There are several other editing features that the figure window provides, including line styles, marker styles (e.g., the previous 'o' choice can be changed), and font styles for text. In editing your plots, remember to avoid the use of color as a distinction between data sets because that type of distinction is lost with photocopying.

Figure editing can also be achieved through commands entered in the command window. These editing line commands, along with the MATLAB line commands to plot, can be used within programming files when graphical output is needed from a MATLAB computer program. Some of the editing commands are listed and explained below, and a more complete list of line commands for two-dimensional graphing can be obtained by typing "help graph2d" at the prompt in the command window.

» hold on » axis([0 70 0 300]) » grid » xlabel('Time t, s') » ylabel('Velocity V, ft / s') » title('THE INFLUENCE OF TIME ON VELOCITY')

The "hold on" command retains the current rectilinear plot of the data as points (until you disable it by typing "hold off"). The "axis" command adjusts the *x*-axis limits to 0 and 70, and the *y*-axis limits to 0 and 300. The "grid" command inserts grid lines along both axes. The "xlabel" statement labels the *x*-axis with the text enclosed within the single quotes, and the "ylabel" statement labels the *y*-axis with the text enclosed within single quotes. The "title" command places the text enclosed in single quotes at the top of the plot; this should be moved to a location beneath the *x*-axis label. Whichever way you choose to edit the plot, you should eventually achieve something similar to the figure shown below in the figure window.



THE INFLUENCE OF TIME ON VELOCITY

Now, the best-fit line should be plotted on the same figure with the data points. To do this, let MATLAB determine the equation constants for a linear fit of the time-velocity data.

Use MATLAB's "polyfit" command to obtain a first-order fit of the time-velocity data with time as the independent variable. A first-order (i.e., linear) fit will produce two equation constants. The first is the coefficient for the independent variable raised to the first power, and the second is the coefficient for the independent variable raised to the zeroeth power (which is 1 for any value of the variable). You know these equation constants better as m and b.

Enter this command.

» EQ = polyfit(Time,Velocity,1)

EQ =

4.0821 -0.5071

In entering the command above, you have told MATLAB that Time contains the independent variable values, that Velocity contains the dependent variable values, that a first order fit is required, and that the two equation constants are to be stored in the matrix called "EQ". At this point,

EQ(1) = m and EQ(2) = b.

To produce the best-fit line, use these equation constants and the Time values to calculate new velocity values called Vcalc.

w Vcalc = EQ(1).\* Time + EQ(2)

Vcalc =

 $-0.5071 \quad 40.3143 \quad 81.1357 \quad 121.9571 \quad 162.7786 \quad 203.6000 \quad 244.4214$ 

You should notice that the value assignment statement you used to generate Vcalc has the y = mx + b form, and also that the multiplication of EQ(1) and Time is an array operation (because you want the multiplication to occur on an element–by-element basis for the vector Time).

Now plot the Vcalc values (i.e., the calculated values of velocity) against Time, as a line.

» plot(Time,Vcalc,'-')

In the figure window, you see this.



The best-fit line is now plotted.

Two things should be noted at this point. First, we calculated new dependent (*y*-axis) values, but kept our independent (*x*-axis) values to plot the best-fit line. Why? Because we made the assumption that there is less uncertainty in our independent variable values, and they should be kept. After all, we control the independent variable values, while the dependent values we measure may be subject to a wide variety of uncertainty sources. This "less uncertainty in the independent values" assumption is likely to be true, but there are no guarantees. Experimentalists, for example, might misread dials when adjusting the independent variable value.

The second thing to note is that MATLAB's figure window offers "Basic Fitting" under the "Tools" menu, and a linear fit is available there. You might use this feature to check your work, but always provide your instructor with the polyfit work you do to obtain m and b, and to plot the best-fit line.

## LOG-LOG PLOTS

MATLAB syntax for a loglog plot of points is

loglog(x, y, 'o')

where x contains the independent variable values, y contains the dependent variable values, and 'o' designates that the plotted values be marked on the plot by lower case "o". To put the time-velocity data on a log-log plot, enter the following commands.

» hold off » loglog(Time,Velocity,'o')

The "hold off" command allows MATLAB to overwrite any existing plot in the figure window when a new plotting command is executed. The "loglog" command produces a plot of points with both the x and y axes having logarithmic scales. After editing to label the axes, insert a grid, and add a title, the figure window holds the plot shown below.



THE INFLUENCE OF TIME ON VELOCITY

You should notice that MATLAB has plotted only six of the seven coordinate pairs; the (0.0, 0.0) set was eliminated because the quantity "log 0.0" cannot be handled. Now, the best–fit line should be plotted. Again, MATLAB's polyfit command is used, but there are some twists and turns that must be worked through.

The first thing to realize is that a power function of the form  $y = b x^m$  has a linear form that can be obtained by taking the log of both sides of the equation. If this is done, the resulting expression looks like this.

log y = m log x + log b

This form indicates (by comparison with y = mx + b) that a linear fit between log x and log y data will yield the values *m* and log b. First the (0.0, 0.0) data pair must be eliminated; the data is re-entered without the zeros, as "t" and "V".

» Time

Time =

0 10 20 30 40 50 60 » t = [10 20 30 40 50 60]; » Velocity

Velocity =

0 41.3000 77.4000 125.0000 165.0000 197.0000 248.0000

» V = [41.3 77.4 125 165 197 248];

Now the "polyfit" command can be used as shown below, where "log10" designates base 10 log values.

 $\Rightarrow$  EQLL = polyfit(log10(t), log10(V), 1)

EQLL =

1.0010 0.6069

**»** 

The numbers MATLAB returns must be interpreted properly. The first, EQLL(1), is m, and the second, EQLL(2), is log b. This is surmised by comparing

log y = m log x + log b

with the "polyfit" command that was entered. Calculated values for velocity can be obtained with these numbers and the time values stored in "t".

EQLL(1) = m and EQLL(2) = log b and b = 10 <sup>EQLL(2)</sup> » Vc = 10 ^ EQLL(2) .\* t .^ EQLL(1) Vc = 40.5458 81.1505 121.7776 162.4191 203.0714 243.7322

The calculated velocity values (Vc) are plotted against "t" values with the "loglog" command to produce the best-fit line on the plot.

```
» hold on
» loglog(t,Vc,'-')
»
```

```
»
```



THE INFLUENCE OF TIME ON VELOCITY

## SEMILOG PLOTS

MATLAB syntax for semilog plotting is

semilogy(x, y, 'o')

where a logarithmic scale has been specified for the *y*-axis ("semilogx" is used for a logarithmic *x*-axis). To avoid difficulties with zeros, "t" and "V" are used here as well.

» hold off
» semilogy(t,V,'o')



In order to plot the best-fit line, the equation constants will be determined with MATLAB's "polyfit" command, but, again, there are some twists.

Exponential functions of the  $y = b e^{mx}$  form are put into linear form by taking the natural log of both sides of the equation. This yields the expression

ln y = m x + ln b

which indicates that a linear fit between x and ln y will generate m and ln b values. You should note that MATLAB understands "log" to be the natural logarithm.

= polyfit(t, log(V), 1)

EQSL =

0.0344 3.5958

So, in this case, EQSL(1) = m and EQSL(2) = ln b. These values and "t" are used to calculate velocity values for the semilog plot.

 $EQSL(1) = m \quad and \quad EQSL(2) = \ln b \qquad b = e^{EQSL(2)} \quad where \quad e = exp(1)$  $> Vcs = (exp(1)) \land EQSL(2) .* exp(EQSL(1) .* t)$ 

Vcs =

51.4135 72.5293 102.3173 144.3395 203.6203 287.2480

These calculated velocity values, Vcs, are plotted against time using the "semilogy" command to produce the best-fit line on the plot.





THE INFLUENCE OF TIME ON VELOCITY