Curve Fitting, Loglog Plots, and Semilog Plots [1]

In this MATLAB exercise, you will learn how to plot data and how to fit lines to your data.

Suppose you are measuring the height h of a seedling as it grows. The height (measured in centimeters) will be a function of time t (measured in days). Suppose you measure the height once a day and you obtain the following data:

t (days)	1	2	3	4	5
h (cm)	5.2	6.6	7.3	8.6	10.7

Let us plot these data in MATLAB. For this, you have to represent your data as one-dimensional arrays, which are also called *vectors*. You can define vectors in MATLAB as follows:

```
>> t = [1 2 3 4 5]
>> h = [5.2 6.6 7.3 8.6 10.7]
```

Don't forget the spaces that separate the numbers in your vectors. Once you have entered your data, you can use the `plot` command to display your data:

```
>> plot(t, h, 'ro')
```

The `'ro'` part of the `plot` command tells MATLAB to display your data points as red circles. This part of the `plot` command is optional. If you omit it, MATLAB will join your data points by straight lines. Try:

```
>> plot(t, h)
```

We really don't want lines joining each pair of consecutive data points in this exercise; and circles are a bit vague as markers for data points, so let us try this:

```
>> plot(t, h, 'k+')
```

Now the first and last data points have become invisible; let us make them reappear by changing the axes on our plot. You can force MATLAB to pick the interval $[0, 6]$ for the x-axis and the interval $[0, 15]$ for the y-axis by entering:

```
>> axis([0 6 0 15])
```

Now look at the resulting graph. Does it appear that $h(t)$ is a linear function? Certainly, there is no straight line that *exactly* fits all data points, but it appears that the growth of your seedling can be *approximately* described by a linear function.

Which linear function would best fit your data points? This question is your first example of a *curve fitting problem*. In this case, we want to know which straight line best fits all the data points you have plotted in this graph. MATLAB has an easy command for finding this straight line. Enter:

```
>> polyfit(t, h, 1)
```

[1]Copyright ©2003 Winfried Just, Department of Mathematics, Ohio University. All rights reserved.

MATLAB returns two numbers. The first of these numbers, 1.3, is the slope of your line. The second number, 3.78 is its y-intercept. Let us plot this line on the same graph as the data points. First we must make sure that MATLAB does not erase the previous graph when you add the line. For this, enter:

```
>> hold on
```

Now we want to use the `plot` command to draw the line. We have seen that this command plots data points and may connect them by straight lines. So we need to define vectors of (lots of) data points that lie on this straight line. First we define an x-vector.

```
>> x = [0:0.5:6]
```

MATLAB gives you lots of numbers because this command defines a vector whose first entry is 0, the last entry is 6, and the other entries are obtained in increments of 0.5. If you define long vectors this way, you do not want to see all these numbers on the screen. You can instruct MATLAB to suppress the output of a command by putting a semicolon at the end. Now let us define a y-vector that contains the function values of the x-vector for the function $y(x) = 1.3x + 3.78$:

```
>> y = 1.3*x + 3.78
```

Although the definition of y looks like the definition of a number, y is really an array of numbers, since x is. Now let us plot the graph of the straight line that supposedly best fits our data points. This time we *do* want the `plot` command to connect points by straight lines!

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

Look at your graph. The straight line we found does seem to approximate our data points rather well, doesn't it?

Before you print your graph,you want to properly label your axes and add a title to it. You can label the horizontal axis as follows:

```
>> xlabel('time in days')
```

And the vertical axis:

```
>> ylabel('seedling height in cm')
```

When labelling a graph, you always should indicate the units of your variables.
Now you can add your name as a title to the graph, like:

```
>> title('My name')
```

If this MATLAB exercise is being counted in your grade, then print this figure for submission.

Next I want to show you how to use semilog and loglog plots to fit curves to data that appear to be governed by an exponential or power relationship. Since we will produce a fresh plot, let us enter:

```
>> hold off
```

Now suppose that you recorded, once every week, the length L (in centimeters) of a growing rodent pup and that you also recorded its mass m (in grams). You got the following measurements:

t (weeks)	1	2	3	4	5
L (cm)	1.0	1.6	3.0	6.2	12.8
m (g)	0.1	0.3	2.1	19.0	168.7

Let us define new vectors:

```
>> L = [1.0 1.6 3.0 6.2 12.8]
>> m = [0.1 0.3 2.1 19.0 168.7]
```

Let us try to plot L as a function of t:

```
>> plot(t, L, 'ro')
```

The data points, shown as red circles, do not appear to sit approximately on a straight line. How can we approximate them? To get an idea, we can try to display the same data points in a semilog plot. Enter:

```
>> semilogy(t, L, 'ro')
```

Now the data points do appear to sit on a straight line! Note that while the scale on the horizontal axis (corresponding to the variable t) is still linear, the vertical axis (which depicts L) is now drawn on a logarithmic scale. We still can use our `polyfit` function to find an equation for this line, but now we need to remember that the line will depict a relationship between t and $\log_{10} L$. In MATLAB, $\log_{10} L$ can be entered as `log10(L)`. Thus we enter:

```
>> polyfit(t, log10(L), 1)
```

MATLAB tells us that a line of slope 0.2803 and y-intercept -0.3246 appears to fit the semilog plot best. In mathematical terms, this means that the data appear to approximately be related as follows:

$$log_{10} L = 0.2803t - 0.3246 \tag{1}$$

Raising 10 to the power equal to both sides of equation (1) we obtain:

$$10^{\log_{10} L} = 10^{0.2803t - 0.3246} \tag{2}$$

or equivalently,

$$L = 10^{0.2803t - 0.3246} \tag{3}$$

It follows that the length of the rodents appears to grow exponentially as a function of time.

Let us check whether our predicted curve really fits the data. Enter:

```
>> plot(t, L, 'ro')
>> hold on
```

To plot the graph of the exponential function, we define vectors x and y as before. This time you do want to suppress output by putting semicolons at the end. Note the little period in front of the exponentiation symbol in the definition of y. It must be there in order to indicate that we raise every element of the array y to the stated power, as opposed to raising the whole array to a power, which is a different operation that is of no interest here.

```
>> x = [1:0.01:5];
>> y = [10.^(0.2803*x - 0.3246)];
```

Now enter:

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

and look at the figure. The exponential function does not fit the data perfectly, but reasonably well; doesn't it? Now add labels to the axes and your name as a title to the figure, and print it if submission is required.

Finally, let us explore the relationship between the length L of your animal and its mass m.
Since we will produce a fresh plot, let us enter:

```
>> hold off
```

Let us try to plot m as a function of L:

```
>> plot(L, m, 'ro')
```

The data points, shown as red circles, do not appear to sit approximately on a straight line. How can we approximate them? To get an idea, we can try a semilog plot again. Enter:

```
>> semilogy(L, m, 'ro')
```

The points on this graph do not appear to sit on a straight line, do they? So let us try to display the same data points on a log-log plot for a change. In curve fitting, you often have to experiment with different approaches before you get a good fit. Enter:

```
>> loglog(L, m, 'ro')
```

Now the data points do appear to sit on a straight line! Note that the scales on both axes are now logarithmic. We still can use our `polyfit` function to find an equation for this line, but now we need to remember that the line will depict a relationship between $\log_{10} L$ and $\log_{10} m$. Thus we enter:

```
>> polyfit(log10(L), log10(m), 1)
```

MATLAB tells us that a line of slope 2.9537 and y-intercept -1.0636 appears to fit the log-log plot best. In mathematical terms, this means that the data appear to approximately be related as follows:

$$\log_{10} m = 2.9537 \log_{10} L - 1.0636 \tag{4}$$

Raising 10 to the power equal to both sides of equation (4) and using laws of exponents we obtain:

$$10^{\log_{10} m} = 10^{2.9537 \log_{10} L - 1.0636} = (10^{\log_{10} L})^{2.9537} \cdot 10^{-1.0636} \tag{5}$$

Since $L^{\log_{10} L} = L$ and $10^{-1.0636} = 0.0864$, we obtained the following power relationship for m as a function of L:

$$m = 0.0864L^{2.9537} \tag{6}$$

This makes sense, since the mass of an animal should be approximately proportional to the third power of its length. Let us check whether our predicted curve really fits the data. Enter:

```
>> plot(L, m, 'ro')
>> hold on
```

To plot the graph of the exponential function, we define vectors x and y as before. Again, you do want to suppress output by putting semicolons at the end. The vector x should cover the whole horizontal axis of our plot, so it should range from 0 to 14.

```
>> x = [0:0.01:14];
>> y = [0.0864*x.^2.9537];
```

Now enter:

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

and look at the figure. The power function does not fit the data perfectly, but reasonably well; doesn't it? Now add labels to the axes of your graph and your name as a title to the figure, and print it if submission is required.