Data Plotting and Curve Fitting in MATLAB

Curve Fitting

Get the file pwl.dat from the class web page. This is an ASCII text file containing two columns of numbers representing the *x* and *y* coordinates of a dataset.

From MATLAB, type `load pwl.dat` to load the file into a matrix named `pwl`. Type `pwl` to display the 100×2 matrix in text form. In order to view the data graphically, type

```
plot(pwl(:,1), pwl(:,2), 'o')
```

This plots the second column vs. the first column using circles for each data point. This is the appropriate way to display *measured* data from an experiment. You should not connect the points with lines because no measurements were made between the points.

Incorrect way to plot measured data: `plot(pwl(:,1) ,pwl(:,2))`

Now that we've got the data plotted as individual points, lets label our axes:

```
xlabel('Input Voltage [V]')
ylabel('Output Current [mA]')
title('Curve Fitting Exercise')
```

It is *very* important to *always* label your axes *with units!* This is engineering, not mathematics – we deal with physical quantities.

Now let's fit this (obviously piece-wise linear) data with two straight lines. MATLAB has a handy function called `polyfit` that fits an *n*-th order polynomial to a set of data points. A line is just a 1st-order polynomial. Type `help polyfit` to learn more about this function.

First, we need to isolate the first linear section of the data. Let's plot a subsection of the data and try to get all of the first region and none of the second region:

```
plot(pwl(1:60,1), pwl(1:60,2), 'o')
```

Oops, it looks like we plotted a bit too much of the curve. Try this again until you isolate the first region. Now we give this data to `polyfit`:

```
fit1 = polyfit(pwl(1:50,1), pwl(1:50,2), 1)
```

Look at the variable `fit1`. As you learned from the `polyfit` help file, `fit1` is a vector representing the coefficients of the fitting polynomial. We can plot the line represented by these parameters by entering:

```
plot(pwl(:,1), fit1(1)*pwl(:,1)+fit1(2))
```

Notice that we leave out the `'o'` parameter so the our fit is plotted as a continuous line. This is appropriate for a theoretical fit. If we want to plot the data and the fit and the same time, we can enter:

```
plot(pwl(:,1), pwl(:,2), 'o', pwl(:,1), fit1(1)*pwl(:,1)+fit1(2))
```

If you're getting tired of typing `pwl(:,1)`, etc., create new variables with shorter names:

```
x = pwl(:,1);
y = pwl(:,2);
```

Now find fit parameters for the second half of the dataset, and plot the data with both fits together:

```
plot(x, y, 'o', x, fit1(1)*x+fit1(2), x, fit2(1)*x+fit2(2))
```

You might want to adjust the axes to zoom in on the data:

```
axis([0 20 -20 50])
```

Finally, you'll need to relabel the axes of the graph and provide a title.

Plotting Exponential Curves

Now load the second sample dataset: edata.dat. Like `pwl`, `edata` is a 100×2 matrix of x and y coordinates. Plot the data:

```
plot(edata(:,1), edata(:,2), 'o')
```

The data seems to follow an exponential curve. Let's plot x vs. log(y):

```
plot(edata(:,1), log10(edata(:,2)), 'o')
```

[In MATLAB, `log(x)` = ln(x) and `log10(x)` = $\log_{10}(x)$.] It's pretty clear that this data is exponential. This is a pretty poor way to represent the y-axis, however, because it only shows the exponent of the data. What would the units be? Here's the way you should plot exponential data:

```
semilogy(edata(:,1), edata(:,2), 'o')
```

The command `semilogy` creates a semi-logarithmic plot with nicely-labeled axes (see also `semilogx` and `loglog`). Now we can label our axes, and the y-axis units make sense:

```
xlabel('V_{GS} [V]')
ylabel('I_{D} [A]')
title('Subthreshold MOSFET Behavior')
```

Fitting Exponential Curves

MATLAB has no command for fitting an exponential function to data. However, we can take the logarithm of the y data and fit the data with a straight line. First, let's relabel our data:

```
x = edata(:,1);
y = edata(:,2);
```

Now, we can generate a straight-line fit to ln(y):

```
fit = polyfit(x, log(y), 1)
```

Now in order to reconstruct an exponential function, we have to exponentiate the fitting line $y = ax+b$, where `fit(1)` = a and `fit(2)` = b:

$$e^{ax+b} = e^b e^{ax}$$

So we can plot the data with the exponential fit as:

```
semilogy(x, y, 'o', x, exp(fit(2)).*exp(fit(1)*x))
```

The “`.*`” operator means item-by-item multiplication of two vectors. (The “`*`” operator can only be used when multiplying a scalar by a scalar or a vector by a scalar.) The “exponential slope” is given by `fit(1)`.

Similar techniques can be used to fit square-law data.

Other Useful MATLAB Functions

Use the `help` function to read about these potentially useful commands:

```
grid
hold
axis
gtext
figure
plot
semilogy
semilogx
loglog
polyfit
subplot
sqrt
diff
```