Basic MATLAB: for Physics majors!  Vinnie Zoutenbier

If you haven't ever taken a computer class MATLAB can be a little hard to understand. Here, I'll do my best to break it down to the bare minimum and get you to be able to do a few basic operations that I think are important for physicists.

When opening MATLAB for the first time, it can be pretty daunting. It has lots of boxes all around a main window with two greater-than symbols (the large text box is your command prompt.) This is where you type calls to functions that compactly carry out whatever calculations or code may be behind them. MATLAB, as most other programming languages, will let you assign numbers to different variables (can be 1-30 characters starting with a letter and containing only letters, numbers and underscores.) It's important to realize that MATLAB makes all it's variables matrices; a constant is recognized as a 1x1 matrix (see code, line 1; the semicolon at the end of an assignment stops MATLAB from showing the result of the assignment) Okay, but where's the fun in that?

Let's suppose you have some data and you want to be able to graph it. Just input it as a vector (matrix row) as shown in line 2 of the code below (to delineate each member put a space or a coma between them.) From here there's a lot that can be done with the data. You can multiply it by a constant (line 3,) add a constant to it (line 4), be careful with multiply by other data sets because then you move into the realm of linear algebra and matrix dimensions will need to match. For the purpose of getting it done, I'm just going to give the code:

```
for i=1:size(xdata,2)
    xdata(i)=(xdata(i)*xdata(i));
end
```

which will square all the terms. This is the code for simple loop and is very versatile for a large number of calculations. Note: this, as does the code in lines 2 and 3, overwrites the old xdata term; if you don't want this to happen, just change the name of the left side of the equality to any unused variable name. In the for loop (the second line above) just make a call to xdata(i) when you want to use the data point and manipulate your data however you want using +, -, *, /, sqrt( ), etc. For example, let's say you measured some displacements and want to know the potential energy of a spring (k=15) given by each:

```
for i=1:size(xdata,2)
    xdata(i)=.5*15*(xdata(i)*xdata(i));
end
```

MATLAB follows the order of operation, but it's a good idea to use parenthesis if something isn't working out right.

Finally, let's say you want to plot your results. Just input your data into two vectors of the same size and call the plot function (line 7.) If you're looking to get fancy, call p=polyfit(xdata,ydata,n), which will return a vector whose members are the coefficients of the polynomial that best fits your data points of degree n. This means that the function \( x^3-2x^2+3x+17 \) will return \( p=[1 \ -2 \ 3 \ 17] \). To plot this function, figure out which x values you want it evaluated at and call it xfit, (usually you want a large number of evaluation point for a smooth line.) Call yfit=polyval(p,xfit) and KABAM! You got a complete dataset for the best fit. Plot it like we did the regular data points leaving off the 'X' (line 8, this connects the dots with a line rather than just a scatter plot of data.) To graph the old plots with the new best fit just put the keyword hold between the plot calls (line 9.)

There. We did it. That wasn't so hard. Now you can't say you don't know how to use MATLAB because you can. You can manipulate data and even graph it with a best fit curve! Make sure you play around with this and if you need help, don't be afraid to google for a more in-depth tutorial.
(1) >>x=17;
(2) >>xdata=[0.0 9.6 19.5 29.0 38.5];
(3) >>xdata=xdata*4;
(4) >>xdata=xdata+2;
(5) >>xdata=[0.0 9.6 19.5 29.0 38.5];
(6) >>ydata=[0,1,2,3,4];
(7) >>plot(xdata,ydata,'X')
(8) >>plot(xfit,yfit)
(9) >>plot(xdata,ydata,'X'),hold, plot(xfit,yfit)