A.J. Melhus
3/21/10

Introductory MATLAB exercises, part 2

When you are stuck working with Matlab and you need to turn somewhere for help, there are 4 places to go:

IN MATLAB

1. Use the <help> **command**. This brings up a basic list of features about the function or script you are interested in, in the **command window**:

`>> help cosh` - this brings up a list of features for the cosh function.

2. Use the **HELP window**. Click on the characteristic blue '?' box to bring up Matlab's **help and demo window**. This window allows you to search for terms and generally produces more results than the help command line action. In addition to basic information, the sections often provide useful demos for different features of each function or script, so that you can see how to use it effectively.

OUTSIDE MATLAB

3. Use the **internet**! While you should be more wary of this method, this is sometimes the best tool for quick, bare-bones information. You can simply Google nearly any Matlab help topic or general problem, and someone out there has likely made an HTML page with code to follow along (many pages are .edu so they are probably reliable).

ACTUAL PERSON

4. If you are on the **University of Maryland campus**, there are **Matlab tutors** in various WAM computer labs (usually at the frequented labs like the PG-2 lab under Regents Drive Garage). These people are pros and can tackle nearly any Matlab problem you ask them, if you know how to ask it.

Walk over to the OIT building or check the OIT website for the tutors' locations and hours (they generally help during normal business week hours, about 12-4 pm).

Plotting in Matlab:

The easiest way to get a quick plot of some function is to use the <ezplot> command. This allows you to plot symbolic, vector or string entries.

`>> ezplot(f)` plots the expression $f = f(x)$ over the default domain: $-2\pi < x < 2\pi$

`>> ezplot(f,[min,max])` plots $f = f(x)$ over the domain: min < x < max.

```
>> f = 'sin(x)';
>> ezplot(f, [0,2*pi])
```

- this plots sin(x) from $0 < x < 2\pi$. Or, even more simply:

`>> ezplot('cos(x)')` - this plots cos(x) from $-2\pi < x < 2\pi$

From this simple plotting command, we can work our way into the more powerful and useful 2-D plotting command, `<plot>`.

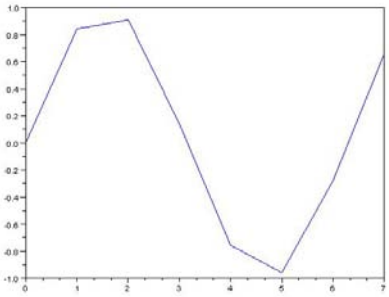
From the web: "Two-dimensional line and symbol plots are created with the `plot` command. In its simplest form `plot` takes two arguments

```
>> plot(xdata,ydata)
```

where `xdata` and `ydata` are vectors containing the data. Note that `xdata` and `ydata` must be the same length and both must be the same type, i.e., both must be either row or column vectors."

Unlike `ezplot`, `plot` will only plot the number of x-data you set. As shown in the following example:

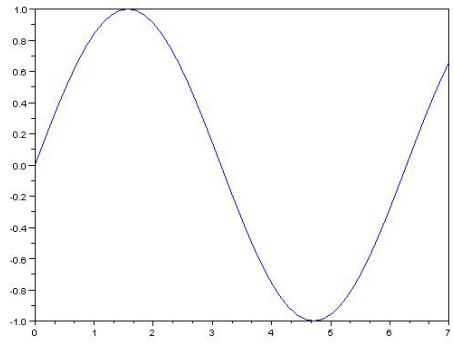
```
>> x = 0:2*pi;
>> y = sin(x);
>> plot(x,y)
```



This plot is bad because I only plotted about 8 points over [0, 2π]. There are a number of ways to work around this problem. The easiest is to use the `<linspace>` command. This command allows you to tell Matlab how for many points you want it to evaluate the function or command.

```
>> x = linspace(0,2*pi, 50);   %I told Matlab to make an x vector with 50
                               %evenly space points within [0, 2*pi]

>> y = sin(x);
>> plot(x,y)
```



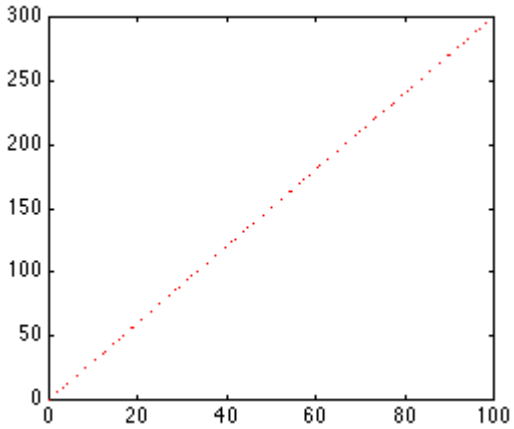
This plot is good because I plotted many more points within [0, 2π].

Plot Aesthetics: (taken/edited from http://www.engin.umich.edu/group/ctm/extras/plot.html)

The color and point marker can be changed on a plot by adding a third parameter (in single quotes) to the `plot` command. For example, to plot the above function as a red, dotted line, the m-file should be changed to:

```
x = 0:0.1:100;
y = 3*x;
plot(x,y,'r:')
```

The plot now looks like:



The third input consists of one to three characters which specify a color and/or a point marker type. The list of colors and point markers is as follows:

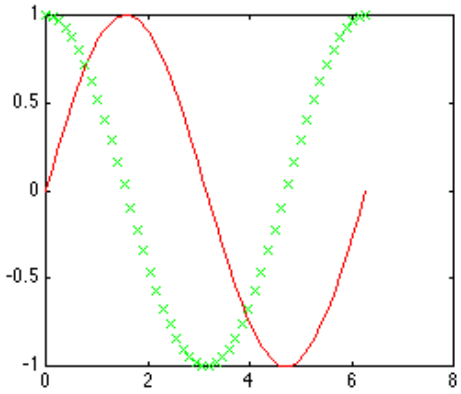
```
y     yellow          .     point
m     magenta         o     circle
c     cyan            x     x-mark
r     red             +     plus
g     green           -     solid
b     blue            *     star
w     white           :     dotted
k     black           -.    dashdot
                        --    dashed
```

Multiple Plots

You can plot more than one function on the same figure. Let's say you want to plot a sine wave and cosine wave on the same set of axes, using a different color and point marker for each. The following m-file could be used to do this:

```
x = linspace(0,2*pi,50);
y = sin(x);
z = cos(x);
plot(x,y,'r', x,z,'gx')
```

You will get the following plot of a sine wave and cosine wave, with the sine wave in a solid red line and the cosine wave in a green line made up of x's:



By adding more sets of parameters to plot, you can plot as many different functions on the same figure as you want. When plotting many things on the same graph it is useful to differentiate the different functions based on color and point marker. This same effect can also be achieved using the `hold on` and `hold off` commands. The same plot shown above could be generated using the following m-file:

```
x = linspace(0,2*pi,50);
y = sin(x);
plot(x,y,'r')
z = cos(x);
hold on
plot(x,z,'gx')
hold off
```

Always remember that if you use the `hold on` command, all plots from then on will be generated on one set of axes, without erasing the previous plot, until the `hold off` command is issued.

2 ways to plot multiple objects in same graph figure:

1. `plot(x,y1, x,y2, x,y3, x,y4,'aesthetic',...., x,y_n)`

-- Use multiple arguments within one `plot` command.

2.
```
plot(x,y1)
hold on
plot(x2,y2, 'aesthetic')
plot(x3,y3, 'aestethic')
hold off
```

-- Use `hold on/off` to make adjustments to the current figure without erasing any objects or information.