Basic Plotting in Matlab®

***Abstract*-This document contains detailed instructions for making basic Matlab® plots and adding labels to them.**

I. Your Computer Account

Your first step is to login on the computer in the EE lab. The computers are now linked to the Engman Lab, so you need an Engman account. If try logging in and it doesn't work, you may need to create an Engman Account. If you only have a CADE account, go to the CADE website (cade.utah.edu) and find the link for resetting your password. I found that I had to use my CIS password and my CADE username on the Engman machine.

II. Logging In

Wake up the computer by pressing the shift key (or if that fails, press the power button on the front of the computer [not the monitor]). As instructed on the screen, press CTRL + ALT + DEL (Del key is on the bottom of the keypad on the right side.)

At the login screen, click on switch user. A new screen will appear with two icons. Click on the "Other User" square. On the next screen, type in your username and password for the Engman Lab and click on the right arrow. If the machine says "welcome" you will see your desktop after a few seconds.

III. Activating MATLAB®

To activate MATLAB®, you must find the MATLAB® program and move it to your desktop. To do so, click on the Windows four-colored icon on the bottom left of the screen and then click on "All Programs" near the bottom left of the popup window. In the list of programs that appears, find the MATLAB® folder and click on it. Click on the R2011a folder to open it. You should see a file called "MATLAB® R2011a" with an orange and blue cone-shaped icon. Drag this icon to your desktop.

IV. Launching MATLAB®

Launch MATLAB® by double-clicking on its icon.

The MATLAB® window has four sub-windows. The window in which you type commands is in the middle and has a >> prompt. You can type commands at this prompt much like you would in a calculator. Indeed, MATLAB® may be thought of as a powerful calculator.

Click your mouse to the right of the >> prompt and type in a mathematical expression such as 2+2 then hit the Return (or Enter) key. Your screen will appear as follows:

```
>> 2+2

ans =

     4

>>
```

You will notice that there is a good deal of wasted space in the output. On the menu at the top of the MATLAB® window, click on File and then Preferences. On the popup Preferences window, look for “Numeric display:” to the right side of the list of files. It will show “loose” as the format. Use the drop down menu to select “compact” to eliminate some space in printouts. Click on OK to close the preferences window and try entering 2+2 again. You will see the following on the screen:

```
>> 2+2
ans =
           4
```

V. ENTERING DATA INTO MATLAB®

Suppose you have measured some data points that have x and y coordinates. For example, the x value might be a voltage across a component in a circuit, and the corresponding y value might be the current through that component for that value of x (i.e., voltage) . Table I, below, shows four such measurements that we wish to enter into MATLAB®.

x (voltage in Volts)	y (current in mA)
0.10	6.01
1.02	5.08
3.95	1.75
5.83	0.42

Although the data are really ordered pairs such as (0.10, 6.01), (1.02, 5.08), etc., we enter all the x values into MATLAB® at once, creating what is called an array. Be sure to use square brackets, [and], when entering the data, not parentheses, (and). For our example, we use the following:

```
>>x = [0.10, 1.02, 3.95, 5.83]
```

The line ends as always when we press the Return (or Enter) key, and MATLAB® will print out our data to let us know it has stored it.

Note: You can tell MATLAB® not to print answers by placing a semicolon at the end of a command:

```
>>x = [0.10, 1.02, 3.95, 5.83];
```

This is helpful later on when you want to control the appearance of your results.

Note: We have created an array called x that we can now use in place of typing all our x values. If we now type x at the >> prompt, our array prints out again.

Now we enter the y values, being careful that we enter the first value first, and so on:

```
>>y = [6.01, 5.08, 1.75, 0.42]
```

Note: If enter an incorrect value in your array, just type the entire line over again with the correct value.

Note: You can bring back previous commands (and then edit them) by hitting the up arrow key on the keyboard. This saves lots of time when typing in long commands in which you want to make small changes.

VI. Plotting Data:

To plot our data, we use one simple plot command. (The ease with which we can make a plot in MATLAB® is one of the reasons why MATLAB® is such a powerful tool.) Our plot command tells MATLAB® three things:

1) What variable contains values in the horizontal direction,
2) What variable contains values along the vertical direction, and
3) What plot symbol to use for the data points.

Here, our plot symbol will be a blue + for each data point:

```
>> plot(x, y, 'b+')
```

Note: Now you must use parentheses, (and), and not square brackets. You must also not leave a space between the word plot and the left parenthesis, (. A space between the b and + would also cause a problem. MATLAB® will print an error message in red if you make such a mistake. The error messages may be cryptic, but you can correct errors by simply typing in the command again with the correct format. Our plot appears as shown in Fig. 1.

Note: The single quote marks around the b+ tell MATLAB® that the b+ is a string of characters. More will be said about that when you learn MATLAB® in detail. For now, just be sure you use the single quotes. Double quotes do not work, by the way.

Note: You can read all about the plot command and available plotting symbols by typing "help plot" at the >> prompt:

```
>> help plot
```

If you know the name of any MATLAB® command, you can type "help" followed by the command name to get information about it. The problem in MATLAB® is figuring what command you need! For that, you can try "help help" or you can probably find it using Google on the web.

Note: The plot window may be in back of the main MATLAB® window. To find the plot, click on MATLAB®'s Window menu and look for an entry such as "A Figure 1".

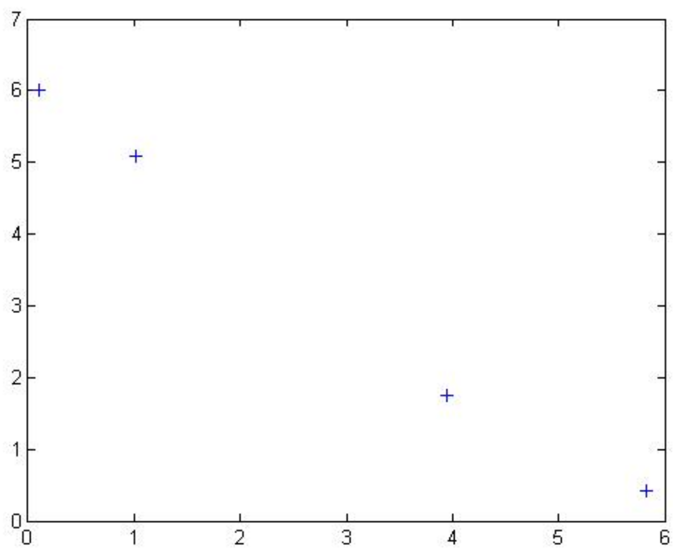

Figure 1. Four data points plotted as + signs.

VII. Adding Features to the Data Plot:

To add labels to the axes and a title to the plot, we use commands like the following:

```
>> xlabel('Voltage (V)')
>> ylabel('Current (mA)')
>> title('Power Supply Current versus Voltage')
```

We may want to control the starting and ending points of the axes instead of letting MATLAB® do it automatically. We can, for example, force MATLAB® to include negative values on the axes and show values up to 10 on both axes by using the following command:

```
>>axis([-1, 10, -5, 10])
```

Note: The sequence of the numbers in the axis command is as follows:

```
[xmin, xmax, ymin, ymax]
```

Note: Both parentheses and square brackets are necessary here. The square brackets create an array from the four numbers, -1, 10, -5, and 10. The parentheses indicate that the numbers in the array are an argument of the function called "axis". That is, the array is being handed into the axis command as information the axis command needs in order to complete its task of drawing axes.

Note: The command is "axis", not "axes". MATLAB® has many such quirks.

VIII. Adding a Straight-Line Fit to a Plot:

The four data points on our plot seem to lie on a line. We could use a ruler to draw a straight line passing through (almost through) the data points, or we can have MATLAB® find the best straight-line fit for our data:

```
>> linefit = polyfit(x, y, 1)
linefit =
    -1.0340 6.1127
```

Note: Our results are now stored in an array called "linefit". When naming an array, be careful to avoid using words that have special meanings in MATLAB®. For example, the word "line" is a command in MATLAB®. If we call something "line", the MATLAB® command for line is no longer available, and we might get confused later on. To check whether something is a MATLAB® command, use, e.g., "help line". If nothing is found, you may safely use the variable name.

This command returns the values of slope, a, and y-intercept, b, for our line:

$y = ax + b$

We now want to create two data points (x and y values) on our line by plugging in two values of x at the ends of the line. Given our axis command, we will use x = -1 and x = 10:

```
>> xvals = [-1, 10];
>> yline = polyval(linefit, xvals)
```

Our x values for the line are now in "linefit", and our y values for the line are now in "yline".

IX. PLOTTING THE STRAIGHT LINE:

We want to superimpose the straight line plot on the existing data plot in MATLAB®. To do so, we must tell MATLAB® to "hold on" to the existing plot and add our line plot to it.

```
>> hold on
```

Note: We must now use a "hold off" command if we ever want to create a new plot. Otherwise, all our plots we continue to be superimposed on all the previous plots. Another way to start a new plot is to close the Figure 1 window containing the plot.

Now we plot the straight-line fit as a red line:

```
>> plot(xvals, yline, 'r-')
```

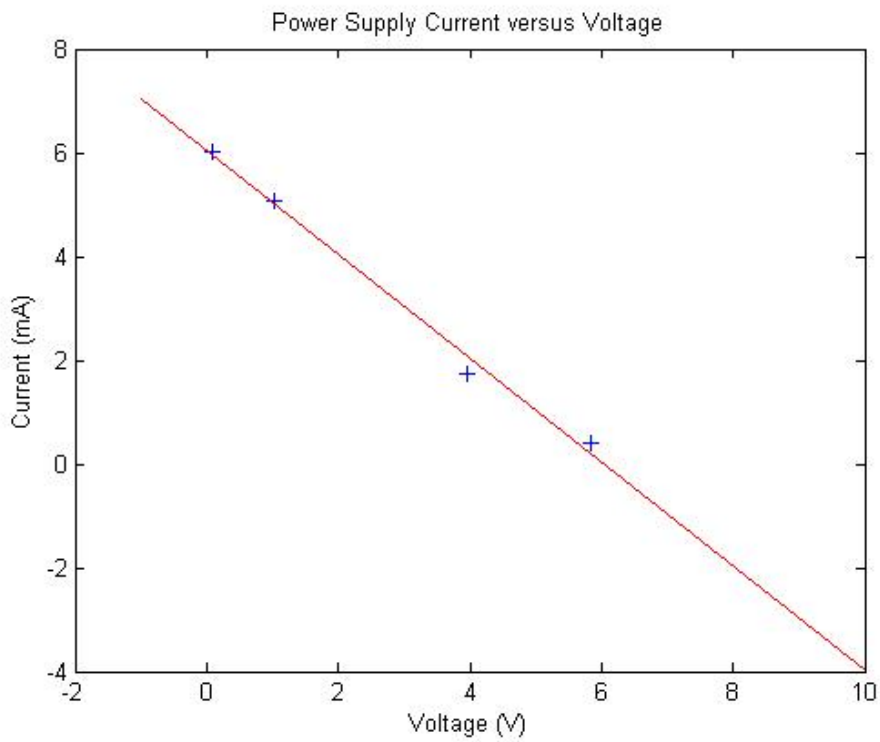

Figure 2. Result of using polyfit() to find straight line fit of four data points

X. PRINTING THE PLOT:

Make the Figure 1 window active (using the Window menu, if necessary). Under the File menu, click on Print. The printer name will something like "digiprint5". If your plot doesn't print immediately, the printer may need paper or may need to warm up, or might even be turned off. To save paper, investigate the problem rather than printing multiple times.

XI. SHUTTING DOWN MATLAB®:

You can exit MATLAB® by typing "exit" at the >> prompt or by clicking on Quit under the File menu.

XII. Logging out:

Click on the Windows four-colored icon in the lower left corner of the screen and then click on the small arrow to the right of the "Shut down" button in the lower right corner of the window that popped up. On the new popup menu, click on "Log off".