6.002 Lab 3
MATLAB Information

Lance Bourque

April 23, 2006

1 Introduction

The purpose of this document is to provide useful information about how to get MATLAB to produce the plots in 6.002 Lab 3 Pre-Lab exercise (3-2) and Pre-Lab exercise (3-5). You may also wish during the Post-Lab to use MATLAB to plot data collected during the In-Lab exercises. This issue is also addressed. Note that some MATLAB functions, e.g. printing, *will not work* at all in the 6.002 lab! You should go to a regular ATHENA cluster to work on MATLAB.

2 Starting MATLAB on ATHENA

First, you will want to log in to ATHENA and start MATLAB. The easiest way to do this is to type `add matlab` followed by the `ENTER` key at the `athena%` prompt. Then type `matlab&` at the new `athena%` prompt. After some time a small window called "Matlab 7.1 (R14SP3)" or something very similar will appear. At the `>>` prompt in the new window type `desktop` followed by the `ENTER` key as suggested in the MATLAB window. After some more time, a new window called "MATLAB", the MATLAB GUI, will appear. You will be working in the "Command Window" part of the new MATLAB window from now on. If there is no `>>` prompt visible in the Command Window, left-click in the Command Window, and then hit `ENTER`. A `>>` prompt will appear.

3 Preparing Equations for Plotting: Pre-Lab (3-2)

Plotting on MATLAB is quite easy once you have set up the appropriate x-axis and y-axis variables. For example, if we already had y as a function of x, we would simply type

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

and a new plot window would pop up containing our desired plot. The purpose of this section is to set up our x-axis and y-axis variables.

For plotting Pre-Lab Exercise (3-2), we know from Pre-Lab Exercise (3-1) that the form of $v_{OUT}(t)$ is

$$v_{OUT}(t) = V_{TO}\sin(\omega_T t + \phi_T)e^{-\alpha_T t}$$

where the parameters V_{TO}, ω_T, ϕ_T, and α_T are some functions of L, C, R, R_{IN} and V_{TI}. First, calculate the numerical values of the parameters. Enter *your results* into MATLAB like this:

`>> V_TO =` *the value of* V_{TO} *you calculated*
`>> w_T =` *the value of* ω_T *you calculated*
...

for all of V_{TO}, ω_T, ϕ_T, and α_T.

Next, we must generate the x-axis variable for plotting. In MATLAB this will be a vector containing all the x-axis points for which you would like to calculate and plot your y-axis variable. Create such a vector like this:

```
>> t = linspace(0,0.3e-3,1000);
```

This command will generate a t vector with 1000 evenly-spaced points starting at $t = 0$ and ending at $t = 0.3 \times 10^{-3}$. (For t measured in seconds, we will have $0 \leq t \leq 0.3$msec). This is the x-axis variable. For more information on `linspace()` (or on any MATLAB command) type `>> help linspace` (or `>> help` followed by the name of the command you would like help on) at the prompt. If you forget the `;` above, MATLAB will echo the entire t vector back to you, and it will be messy; remember the `;` above.

Now, all of our variable names are defined, and we can use the definitions to generate the y-axis variable defined by our function

$$v_{OUT}(t) = V_{TO}\sin(\omega_T t + \phi_T)e^{-\alpha_T t}$$

above. Here's how to do it in MATLAB. Type the following at the prompt:

```
>> v_OUT = V_TO * sin(w_T*t + phi_T) .* exp(-alpha_T*t);
```

assuming you have used these variable names in your parameter definitions. Do not use `*` in place of `.*` or vice-versa above. (FYI, `*` multiplies a number times a vector to produce a vector. `.*` performs an elementwise multiplication on two vectors to produce a vector. `*` on two vectors will perform a vector dot product and return a number). Also, you should use the `;` unless you wish to view all 1000 entries in the v_{OUT} vector (you don't).

Our x-axis variable, `t`, and our y-axis variable, `v_OUT` are now ready to plot.

4 Plotting the Time Response: Pre-Lab (3-2)

Now that the variables have been set up, plotting is easy: at the prompt type

```
>> plot(t,v_OUT)
```

A plot window will pop up containing the plot of v_{OUT} versus t. You may, at this point, wish to save your plot by using the plot window's menu commands.

As an alternative to the above, for Pre-Lab (3-2) you could use the MATLAB function `step()` on the system's transfer function (obtained from impedance in Pre-Lab (3-4)) to plot the time response to a step input. This might be easier, or it might be less intuitive since we have not spent much time discussing system functions in detail in the time domain. Type `>> help step` at the MATLAB prompt for more information. We will, however, use the system function to plot the frequency response in Pre-Lab (3-5) later.

5 Printing Your Plot

You should be looking at a plot in a window entitled "Figure 1". Go to the `File` menu and select `Print...` If necessary, select the desired printer and hit `OK` or `Print` to send your plot to the ATHENA printer. Reminder, this *will not work* from the 6.002 Lab! You must print from a regular ATHENA cluster.

6 Adding Your Data to an Existing Plot

In the Post-Lab you will be asked to plot your In-Lab data on the plot of your Pre-Lab prediction. You can add your data to the existing plot in the following way:

First, you must again set up x-axis and y-axis variables containing *your In-Lab data*. For example, for the following measured (t, v_{OUT}) pairs (these are made up for illustration purposes only) (1,2), (3,4), (5,6), you should type

```
>> myT = [1 3 5];
>> myVOUT = [2 4 6];
```

With your data set up in this way, you can add it to the plot of your prediction. First, view the existing plot for which you would like to add your data (that is to say, make the plot the active window). Next, return *directly* to the Command Window and type

```
>> hold on
>> plot(myT, myVOUT, 'x')
```

This will add your data points to the graph as points with no connecting line between them. You may wish to add a legend to the plot. This can be accomplished by typing

```
>> legend('Prediction','Measured Data')
```

for example. Feel free to change the legend names as appropriate for your work.

When you are done adding data, you may wish to type

```
>> hold off
```

to avoid adding anything else to the plot. You may choose to save your plot at any time.

7 Plotting Pre-Lab (3-5)

The easiest way to use MATLAB to plot the frequency response in Pre-Lab Exercise (3-5) is to take advantage of the built-in function `bode()`. Here's how to do it.

As a result of your calculations in Pre-Lab (3-4) you will have a function, $H_S(\omega_S)$, which is likely a ratio of frequency-dependent complex numbers. If we let s fill in for the complex frequency variable $j\omega_S$, then we might have a function of the form

$$H_s(s) = \frac{As^2 + Bs}{Cs^2 + Ds + E}$$

where A, B, C, D, and E are *real numbers*. Note that the function above is for illustration only; you might have a slightly different form. We can get MATLAB to plot the desired (Pre-Lab (3-5)) plots directly if we let MATLAB know the form of our system function. We also need to let MATLAB know over what frequency range to plot the data. We can set up an appropriate frequency vector (x-axis variable) by typing

```
>> W = 2*pi*logspace(3,5,1000)
```

This frequency vector will contain 1000 logarithmically spaced points from 10^3 Hz to 10^5 Hz (remember $\omega = 2\pi f$).

We also need to let MATLAB know what the coefficients of s are in our transfer function. We do that by entering the numerator *coefficients* as one vector and the denominator *coefficients* as another, and then telling MATLAB to generate a system function with the above form. Once MATLAB knows the system function, it can plot the frequency response directly. Thus, for the above example, we should enter

```
>> num = [A B 0]
>> den = [C D E]
>> HS=tf(num,den)
>> [MAG, PHASE] = bode(HS,W);
```

`HS` will be known to MATLAB as the system function, $H_S(s)$, above. Please note the 0 in the definition of `num` above corresponding to the coefficient of s^0. The 0 is absolutely necessary in this example. The magnitude and phase information have been stored in the vector arrays `MAG` and `PHASE` respectively.

For the magnitude plot requested in Pre-Lab (3-5) you should type

```
>> loglog(W/(2*pi*10e3), MAG(:))
```

where the x-axis variable, `W`, has been normalized as requested in the Lab instructions, and the plot has logarithmic x and y axes.

Similarly, you can create a new figure and plot the phase by typing

```
>> figure(2)
>> semilogx(W/(2*pi*10e3), PHASE(:))
```

This plot has a logarithmic x-axis and a linear y-axis as requested.

To add In-Lab data and legends to these plots, follow the instructions in section 6.

8 Formatting MATLAB Plots

It is easy to add additional formatting to MATLAB plots. For example, to add labels to the x- and y-axes and to add a title to your plot, first make the plot the active window (look at it), then return directly to the MATLAB Command Window and type

```
>> xlabel('My x-axis label')
>> ylabel('My y-axis label')
>> title('My title')
```

where the labels and title above have been replaced with some labels and title that you find appropriate.