ECE 332 Lab 4m

Matlab for RLC Circuit

Computers and circuits work well together as you have seen with Multisim and myDAQ. Now we add a third computer application, Matlab. Its name is derived from MATrix LABoratory because it is designed to work primarily with matrices. In fact, its name is a caution to the user–if you forget Matlab thinks matrices first, you get unexpected results as we'll see later in this lab.

In the previous lab, you simulated the time-domain operation of a RC circuit. All elements were ideal. The responses that you observed were perfect , unhindered by imperfect, non-ideal resistance, capacitance, and inductance. Now we leave that ideal world to see what happens with real, non-ideal circuit elements.

Introduction

Matlab as we will use it is essentially a command-line language. You type a command and it executes it. You can't edit your commands once you've executed them. You can use the up-arrow to go back up to a command and make it the current command, perhaps editing it before executing it again.

Matlab is particularly good at graphing and plotting, with editing features for making first-class plots. These are much more advanced than the rather archaic command-line structure. You can also write files–m files– that are essentially Matlab programs, then execute them instead of having to type commands.

Goals

In this lab, we are going to learn to do two fairly simple things in Matlab. Since all of you have taken ECE 210, this will likely be a good review of Matlab's syntax and capabilities.

- Use Matlab's graphing capability to plot the output of an RLC circuit excited by a step function.
- Improve that plot to a professional level.

Theory

Lab 5s contains a good discussion of the second-order differential equation for an RLC circuit (we will do this lab next lesson). The general solution to the equation in that lab is

$$v_o(t) = K_1 e^{-\alpha t}\cos(\beta t) + K_2 e^{-\alpha t}\sin(\beta t) + A$$

where the circuit is driven by a step function of height A.

If the circuit is initially dead i.e. initial conditions are zero, then this simplifies to:

$$v_o(t) = A - Ae^{-\alpha t}\left[\cos(\beta t) + \frac{\alpha}{\beta}\sin(\beta t)\right]$$

We will write Matlab code to plot this outcome.

Procedure

This lab has three parts, all working with Matlab:

- Run a practice sample to plot a result.
- Improve the plot.
- Write code to plot the Lab 5v result.

A. Practice Plot

Suppose you need to design a series RLC circuit that has an undamped natural frequency f_o of 4 kHz and a damping factor ζ of 0.25. You plan to use Matlab to plot the response of this circuit to a step of 10 volts.

We first need to calculate the values of α and β for the equation above:

$$\alpha = \zeta\omega_o = \zeta(2\pi f_o)$$

$$\beta = \omega_d = \omega_o\sqrt{1-\zeta^2} = 2\pi f_o\sqrt{1-\zeta^2}$$

In the Matlab code that follows are numerous comments. There are lines starting with the % sign.

```
clear all

% It is good practice to clear everything
before starting new code since variables
retain old values

zeta = 0.25

f = 4e3

A = 10

% Label constants so they can be easily
changed later

alpha = zeta*2*pi*f

beta = 2*pi*f*sqrt(1 - zeta^2)

% Calculate values needed for equation

t = 0:10e-6:500e-6;

% This establishes a variable t as a
vector with 51 values from 0 to 500
micro second.  If you don't put the
semicolon at the end of the command,
your screen will fill with all 51 values.

v = A - A*exp(-alpha*t).* (cos(beta*t) +
alpha/beta*sin(beta*t));

%  There are two types of multiplication
here: the * is multiplying by a constant;
the .* (note the dot) is the element-by-
element product of the two functions,
each of which is a vector with 51
elements.  Youwill get a somewhat cryptic
error message if you omit the dot.  Try
omitting it to see what happens.

plot(t,v)

%  The plot will be in a different window
and may be  hiding behind your main
window

xlabel('t seconds') ylabel('v volts')

% These label the axes.  Note the
 single quotes

grid on

title('Lab 4m')
```

B. Improve the Plot

Your plot isn't a bad one, but it can be sharpened up to provide a clearer plot with good calibrations. We'll rerun the plot command and then edit the plot. First, use the up arrow to work backward through your commands to get to the plot command. Edit it to become:

```
plot(1000*t,v)
```

Then, edit the *xlabel* command to become

```
xlabel(t,milliseconds')
```

Notice the time axis is easier to read and doesn't require the reader to mentally do the ten-to-the calculation.

We can do more:

- From within the plot window, **Edit⇒Axes Properties**.

- In the box at the bottom, select **Y Axis**.
- Select **Ticks...**
- Check **Step by** and insert **1**.
- Apply and OK.
- Click **Grid** and check X and Y.
- Add a title in **Title** and a y-axis label in **Y Label:**.
- the exit button in the upper right corner of the editing box will close editing and leave your plot improved.

Now you can resize your graph and change its proportions. You can also save it in a number of formats such as *jpg* for use in Powerpoint presentations.

The End

This lab has served as a quick introduction to Matlab and demonstrated how one can make professional-appearing plots for use in papers and presentations.

In addition to the Matlab instructions that are linked in the 332 web site, there are numerous help files on line. One from MathWorks that is very complete is at

`http://www.mathworks.com/help/techdoc/learn_ matlab/bqr_2pl.html`

Its search command is very useful for looking up commands quickly.