Laboratory 1

Electronics Engineering 3210

Introduction to MATLAB

Purpose:

In this lab, students will be introduced to the some of the capabilities of MATLAB. As an exercise, students will convolve two discrete time signals, first using nested loops, then using the built-in MATLAB functions. Students are encouraged, however, to try different things and analyze the outcomes to expand their learning beyond the basic requirements of this laboratory exercise.

Preliminary:

Matlab has three user interfaces that can be used:

1. **The Command Line** is similar to a calculator in which commands may be entered. All commands will display the answer unless the "echo" feature is disabled by concluding the command with a semi-colon ";"
2. **The Editor** can be opened by typing **edit** at the command line. Using the editor, a sequence of commands may be entered and saved in a file and then run as a program (or script) in the command line. The file name should be 7 characters or less, not contain any special characters, and end with a ".m" extension.
3. **Simulink** is a graphical interface in which information (signals) are passed through commands (blocks) to yield the desired results.

In this lab, we will use only the command line and the editor. (It is recommended that you use the editor for this laboratory exercise – it will save a lot of typing).

It should be noted that MATLAB (like other computational software packages) is extremely powerful and provides many language features, specialized functions and subroutines. In MATLAB if you are unsure what a command does, type **`>> help {command}`** at the command line. If you are unsure what command to use type **`>> help`** and a list of command groups will appear. Enter **`>> help {command group}`** and a list of commands that behave similarly will appear. There is also a graphic user interface for this help environment that can be accessed via the icon in the lower left corner of the MATLAB window.

Also keep in mind as you write MATLAB scripts that the more complex your problem becomes the more important comments and documentation become. Comments can be added to MATLAB commands by typing "%" and following it with any comment text you wish.

Remember that MATLAB IS NOT A THINKING ENTITY. IT WILL ONLY DO EXACTLY WHAT YOU TELL IT, NOT WHAT YOU WANT IT TO DO, which in many cases are two different things.

MATLAB is also designed to echo every answer to the command line. Users may turn the echo feature off by following the command with a semi-colon.

Procedure:

1. Define a vector t that includes values from 0 to 4π in 0.1 increments.

   ```
   >> t = [0:0.1:4*pi]
   ```

   Individual values of a vector can be accessed with parentheses, i.e. **`t(1)`**, **`t(2)`**, … In MATLAB, all matrix and vector indices start with 1.

2. Use a **`for`** loop to individually calculate x(t) = sin (0.5 t) and store the result in a vector. (Type "**`>>help for`**" in the command line for more information on the **`for`** loop.)

   NOTE: MATLAB statements act on vectors or matrices; scalars are just 1x1 matrices. So this step could have been accomplished simply by taking the sine of a scaled vector, i.e. **`x = sin(0.5*t)`**. But write the code using a loop anyway.

3. Using **`plot`**, graph x(t) and label all axis and attach a title. (Try **`plot(t,x)`**, then use title(), xlabel() and ylabel(). Use the help command to learn how to use these MATLAB functions. Try, for example, to plot the function in a color other than blue.) Also note that MATLAB accepts TEX type set commands for displaying symbols, e.g. \pi, \epsilon, \mu)

4. Consider a system whose impulse response is h(t) = u(t-1) – u(t-3). Write MATLAB code to create a vector for h(t) using the same time scale (0.1) as you used for x. The data point corresponding to 1.0 and 3.0 should be 0.5 and the data points in between should be 1.0. Hint: a vector can be defined with square brackets surrounding comma-separated elements or vectors, and the functions zeros() and ones() create vectors. As an example, see what you get if you type:

   ```
   >>y = [zeros(1,9),0.5,1.0,0.5]
   ```

5. Write a MATLAB program that will convolve y(t) = h(t) * x(t) and graph the results.

6. Use the built-in MATLAB function **`conv`** to repeat part 5.

7. Write a title and short description in your lab book. Attach your MATLAB script and the plotted results x(t), h(t) and h(t)*x(t).

8. Write a summary in your lab book. Compare the results of your convolution program with the MATLAB conv subroutine. Also include any additional lessons learned and any topics/commands that may be helpful in future MATLAB based labs.