

# **ECE 146B: Digital Communication Theory and Techniques**

## **Lab 1: Review of Matlab, Simulink, and the Communication Toolbox**

### **Pulse Amplitude Modulation and Signal Reconstruction**

**Lab Report Due: 5:00 p.m., Friday, April 17, 2009**

**(Place in the ECE 146B Homework Box on the 3rd Floor of Harold Frank Hall)**

## **1 Objective**

The objective of this lab is to review how to access Matlab, Simulink, and the Communications Toolbox, and to become familiar with the basic operations of these applications. This part of the lab will not be part of your report; it is provided for those students who did not take ECE 146A. Most students should skip to the sections on pulse amplitude modulation (PAM).

### **1.1 Matlab**

Using Matlab, you will review numerical calculations with complex numbers, and implement vector operations such as vector import and export procedures. You will also familiarize yourself with Matlab's graphical plotting capabilities.

### **1.2 Simulink**

Using Simulink, you will simulate a simple dynamic system that includes a signal generator and a scope. You will also demonstrate how a periodic waveform can be generated by summing together many sinusoids with different frequencies.

### **1.3 Communication Toolbox**

Using the Communication Toolbox, you will become familiar with some applications that will be needed later in the other software labs.

### **1.4 Pulse Amplitude Modulation**

The primary objective of this lab is to learn about two types of sampling, the effect of undersampling (aliasing), and methods of signal reconstruction.

## **2 Equipment**

Matlab, Simulink, and the Communication Toolbox software are available on the ECI workstations.

## 3 Matlab

### 3.1 Introduction

Matlab is a computing environment specially designed for matrix computations. It is widely used for the study of a variety of applications, including circuits, signal processing, control systems, communications, image processing, symbolic mathematics, statistics, neural networks, wavelets, and system identification. Its large library of built-in functions and toolboxes, as well as its graphical capabilities, make it a valuable tool for electrical engineering education and research.

Matlab has an interactive mode in which user commands are interpreted immediately as they are typed. Alternatively, a program (called a script) can be written in advance using a text editor, saved as a file, and then executed in Matlab.

### 3.2 Matrix Manipulation

The basic objects manipulated by Matlab are two-dimensional matrices (though recent versions can process multidimensional matrices). Recall that a vector is a special case of a matrix that has only one row or one column. In this course, we will define a vector as a column vector, which corresponds to a single column of a matrix, e.g., an  $N \times 1$  matrix with  $N$  rows and one column. A row vector is obtained from a column vector by using the transpose operator.

You will find that Matlab is extremely powerful when performing matrix manipulations because many scalar operations operate in parallel for all elements of a matrix. This almost eliminates the need for iterative loops employed in most conventional programming languages. For example, in order to generate  $s(n) = \sin(2\pi n/1000)$  for  $n = 1, \dots, 1000$ , we can write the following program in C.

```
for (n = 1; n <= 1000; n++){
    s(n) = sin(2 * pi * n/1000)
}
```

In Matlab, we could use a **for** loop as follows.

```
for n = 1 : 1000,
    s(n) = sin(2 * pi * n/1000);
end
```

However, it is much simpler to write

```
s = sin(2 * pi * (1 : 1000)/1000);
```

Since Matlab programs are interpreted (not compiled), **for** loops and **while** loops are inefficient. They should be avoided whenever possible.

### 3.3 Data Import and Export

Data generated by other programs can be imported into Matlab with the **load** command. The raw data file can be in ASCII format. Matlab can also generate data to be processed by other programs using the **save** command.

In order to export vector  $x$  (an  $N \times 1$  matrix) to a file called x.dat in ASCII, use the command

```
>>save x.dat x -ascii
```

It is considered good practice to assign the variable name to the file name, as in this example, although both names can be arbitrary.

In order to import data from a file called x.dat, use the following.

```
>>load x.dat -ascii
```

The contents of the file will be loaded to the variable  $x$ . If the file contains  $N$  rows, and each of the rows contains exactly one number,  $x$  will be an  $N \times 1$  matrix (i.e., a column vector).

### 3.4 Graphical Plotting

Matlab supports graphical plotting on the computer screen and to a printer. The command for plotting on the screen is **plot**, which can be used in several formats, as follows.

```
>>plot(y) % Plots vector y versus the index.  
>>plot(x,y) % Plots vector y versus vector x.  
>>plot(x,y '<line type>') % Plots vector y versus vector x with the specified <line type>.
```

Possible line types include line, point, and color specifications. The command **plot** can also take on other forms depending on its argument list. Please refer to the on-line help for a detailed discussion of this command.

A number of commands are used to help generate the desired plot. These include: **axis**, **hold**, **title**, **xlabel**, **ylabel**, **text**, **gtext**, etc. Some of these will be needed for this lab – please refer to the on-line help for details on these commands. Use the **print** command to obtain a high-quality hard copy of the current screen plot.

### 3.5 Obtaining Help

There are two main ways of obtaining help. The first way is by invoking the **help** command on the Matlab command window to provide you with a list of topics for which help is available. By using **help** *<topic>* where *topic* is the name of a toolbox or function, you will receive either a list of functions available in the toolbox, or an explanation of the function itself, respectively.

The second method of obtaining on-line help is by reading the manuals that are installed on the system in PDF format. You can use a web browser with the Adobe Acrobat Reader plug-in to view these manuals.

## 4 Simulink

Simulink is a program for simulating signals and dynamic systems. As an extension of Matlab, Simulink adds many features specific to the simulation of dynamic systems while retaining all of Matlab's general purpose functionality.

Simulink has two phases of use: *model definition* and *model analysis*. A typical session starts by either defining a new model or by recalling a previously defined model, and then proceeds to analyze that model. In order to facilitate the model definition, Simulink has a large class of windows

called *block diagram* windows. In these windows, models are created and edited principally by mouse-driven commands. An important part of mastering Simulink is to become familiar with manipulations of various model components in these windows.

After you define a model, you can analyze it either by choosing options from the Simulink menus or by entering commands in the Matlab command window. The progress of an ongoing simulation can be viewed while it is running, and the final results can be made available in the Matlab workspace when the simulation is complete.

## 5 Preparation

### 5.1 Matlab

Please read about the following topics using the on-line help documents (or a book on Matlab).

- Command mode operation.
- On-line help.
- Script editing and execution.
- Command language: constants, expressions, assignments, m-files, function calls, function definitions, and comments.
- Matlab commands: **diary**, **echo**, **type**, **!**, **pause**, **quit**, **who**, and **whos**.
- Predefined variables: **ans**, **i**, **j**, and **pi**.
- Built-in operators for vector manipulation: **:** (range selector), **'** (transpose), **+**, **-**, **\***, **/**, **^**, **.\***, **./**, and **.^**.
- Language constructs: **for**, **while**, **end**, **if**, and **break**.
- Commands to generate graphical plots: **plot**, **bar**, **stairs**, **title**, **xlabel**, **ylabel**, **text**, **gtext**, **hold**, **axis**, **grid**, **clg**, and **print**.
- Built-in functions: **abs**, **angle**, **clear**, **conj**, **cos**, **exp**, **imag**, **real**, **sin**, **rem**, **round**, **ceiling**, **floor**, **fix**, **fliplr**, **flipud**.

### 5.2 Simulink

A periodic waveform can be simulated by summing the output of several sine wave generators where each generator produces one harmonic of the periodic waveform. The magnitude and phase of each generator must be controlled separately. Such a system model is called a Fourier Synthesizer and will be constructed to simulate several periodic waveforms in this lab. The synthesizer generates the fundamental frequency through the  $n$ th harmonic with separate magnitude and phase controls on each. The sum of all the harmonics is available as the output.

The Sine Wave block in Simulink provides time-varying sinusoids. The amplitude, frequency, and phase of the signal are determined by parameters selected by the user. We will use a Sine Wave block as a harmonic generator in the Fourier Synthesizer. Note that the parameters in the Sine

Wave block can be set to constants or variables. Any variables on which a parameter is dependent must be defined in the workspace when a simulation is started, or else Simulink will indicate an error for the block.

The Sum block in Simulink adds the value of each input to produce a scalar output. The list of sign parameters may be represented with a constant or a combination of the symbols + or -. A combination of pluses and minuses describes the polarity of each individual port, where the number of ports is equal to the number of symbols used. The length of the string is used to determine the number of ports drawn; all the characters other than the plus sign are interpreted as minus signs, including spaces. Thus, it is important not to leave spaces between plus and minus signs in the parameter string, or they will cause unwanted negative polarity input ports to appear. This block will serve as a crucial part in the synthesizer, superimposing all the input harmonics to form the desired periodic waveform.

The Scope block in Simulink will be used to display the input harmonics and the final output waveform. The Averaging Spectrum Analyzer block will be used to analyze the spectrum of the input waveform and the frequency response of the two filters.

- Write the Fourier Series for a square wave with odd symmetry, a peak value of 0.3 V, an average value of zero, and a fundamental frequency of 1 Hz. Write the series in trigonometric form up to the ninth harmonic.
- Write the Fourier series for a triangular wave with even symmetry, a peak value of 1 V, an average value of zero, and a fundamental frequency of 1 Hz. Write the series in trigonometric form up to the 9th harmonic.
- Draw a block diagram of the Fourier Synthesizer using the blocks available in Simulink, i.e., Sine Wave, Sum, and Scope, where the Scope block is used for displaying the synthesized periodic waveform.

## 6 Problems

### 6.1 Matlab

1. Express the following in polar form and rectangular form using Matlab.

(a)  $(1 - j\sqrt{3})^3 + j(1 + j)e^{j\pi/6}$

(b)  $\frac{2-j(6/\sqrt{3})}{2+j(6/\sqrt{3})} \frac{1+j\sqrt{3}}{e^{j\pi/3}-1}$

(c)  $[(\sqrt{3} + j)2\sqrt{2}e^{j\pi/4}]^{0.6}$

2. Find the roots of the following expressions using Matlab.

(a)  $3.72x^2 + 2.1x + 8.25 = 0$

(b)  $4x^2 + 25x - 56 = 0$

3. A complex number  $y = a + jb$  is perpendicular to  $x = c + jd$  if  $a = -d$  and  $b = c$ . Define a function  $y = \mathbf{perp}(x)$  which takes a complex number  $x$  as input and returns a complex number  $y$  which is perpendicular to  $x$ .

4. Using function **perp** defined above, find complex numbers perpendicular to the following.
  - (a)  $3 - j4$
  - (b)  $2.3e^{-j0.4}$
  - (c)  $-2.08(\cos(\pi/3) + j \sin(\pi/3))$
5. Let  $x(n) = n$  for  $n = 1, \dots, 8$ . Calculate and plot  $c(n) = \lfloor \sqrt{x(n)} \rfloor + \lfloor e^n \rfloor$ , for  $n = 1, \dots, 8$ .
6. Compute and plot  $x(n) = e^{-0.003n} \cos(2\pi n/100 + 3)$ , for  $n = 1, \dots, 1000$ . On the same graph, plot  $e^{-0.003n}$  and  $-e^{-0.003n}$  with different colors and line types. Label the curves to identify each one.
7. Given  $x(n) = \sin(n/100)$ , generate and plot the following vectors for  $-628 \leq n \leq 628$ .
  - (a) Plot  $x(n)$ ,  $y(n) = 0.5x(n)$ , and  $z(n) = 0.2x(5n)$  on the same graph.
  - (b) Plot  $x(n)$ ,  $v(n) = x(n - 100)$ , and  $w(n) = x(n + 150)$  on the same graph.
8. Generate a vector for  $s(n) = (-1)^{n \text{div} 8} (n \text{mod} 8)$  for  $n = 0, \dots, 63$  where `mod` yields the residual and `div` yields the quotient of the division of  $n$  by 8. Export the vector to the operating system in ASCII format such that each element of the vector is a row in the output file. Plot this vector using the Matlab commands **plot**, **bar**, and **stairs**, respectively.
9. Vector  $y$  is the reverse of vector  $x$  if the elements of  $y$  are in reverse order relative to those in  $x$ . For example, if  $x = [1; 2; 3]$  and  $y$  is the reverse of  $x$ , then  $y = [3; 2; 1]$ . Use Matlab functions **fliplr** and **flipud** to compute the reverse of  $s(n)$  created above.

## 6.2 Matlab Part of the Lab Report

Please use the following format when preparing your lab report.

1. A script (.m) file should be created for each of the previous problems. They should be named **prob\_1\_a.m** for problem 1(a), etc. Script files for user functions should have proper names based on their functionality.
2. Make sure that you show only important intermediate results. The responses from Matlab should be suppressed for obvious or tedious operations (e.g., assigning a constant to a variable).
3. All script files should contain comments with enough details in order to understand your program.
4. A session *diary* that records the executions of all script files should be created. The following is an example session diary.

```
>>prob_1_a
(execution code for Problem 1(a))
>>prob_1_b
(execution code for Problem 1(b))
etc.
```

5. You should turn in all .m files, all graphical plots, and the diaries for problems 1-4 and 7. Please label these clearly and remember to comment your m-files as well.
6. Write a paragraph about questions and confusions that you have in the Matlab part of the lab.

## 6.3 Simulink

### 6.3.1 Simple Multiplier

Let's design and analyze a multiplier system with a square-wave input.

1. Once Matlab is loaded, type **simulink** at the Matlab prompt. The Simulink library window and an untitled new window will appear. You can name your session and save it.
2. In the library window, double-click on the block named *sources*. A new window of Simulink/sources will appear.
3. Copy a block by dragging it from the Simulink/sources window to your new window. (Any of these blocks can be copied.)
4. In your window, using the left mouse button, double-click on the pulse generator block. A new window appears that displays all the properties of your selected block. Specify the model fields as follows.

**Period:** 1

**Duty cycle:** 50%

**Amplitude:** 0.7

5. Next, we need to multiply the output of the signal generator by a gain. You can use a gain block from the math library by placing it in the design window. Double-click on the *gain* block to open the property window and set the gain to 0.9.
6. To connect the blocks, you must connect the output port ( $>$ ) to the input port ( $>$ ). To connect the generator block to the gain block, use the left mouse button to click the output port of the signal generator block and drag it to the input port of the gain block. A connection line will appear.
7. To analyze the output, drag the Scope block from Sinks of the Simulink block library. Double-click the Scope block to open the Scope window. Specify the Scope axes as follows.

**Ymax:** 1.5

**Xmax:** -1.5

**Time range:** 10

8. Select the Simulation menu and then parameters to open the Simulation control window. Set the simulation parameters as follows.

**Start time:** 0.0  
**Stop time:** 10.0  
**Solver options Type:** Fixed step discrete (no continuous states)  
**Fixed step size:** 0.01

9. Select the Simulation menu and then run the simulation by clicking Start.
10. Save your model file to disk as a Matlab mdl-file by selecting Save from the file menu. This file can be redrawn and simulated on the screen for further editing.

In order to print your plots, you use the following steps.

1. Run the simulation.
2. After finishing the simulation, when you use the default variable name, you can access the ScopeData variable. The first column of ScopeData is the time axis for the simulation, and the second column is the first input to the scope. Use the following Matlab commands.

```
plot(ScopeData(:,1),ScopeData(:,2))  
axis([0 10 -1.5 1.5]);
```

3. Print the plot from Matlab as follows.

```
print  
print -dps my.ps % Save the plot as a postscript file.
```

You can print my.ps from the Unix prompt by typing `lpr -Pprintername my.ps`.

### 6.3.2 Fourier Synthesizer

1. Use Simulink blocks Sine Wave, Sum, and Power Spectral Density to construct a Fourier Synthesizer model for the periodic square wave discussed in the Preparation section. Based on your calculations, set the magnitude, phase, and frequency for these two inputs to the synthesizer, i.e., set the parameters in the Sine blocks which are used to generate the input signals of the synthesizer. Before you add these inputs to the synthesizer, observe the signal on the Scope, making sure that the display is consistent with the parameters you set. Start the simulation and observe the output waveform in the Scope window and its spectrum in the Power Spectral Density window. Note that the path for the Power Spectral Density block is Blocksets&Toolboxes→Simulink Extras→Additional Sinks→Power Spectral Density.
2. Add second, third, . . . , and ninth harmonics to the system – one at a time. In order to do this, you first need to add a “+” in the parameter field of the Sum block, then insert another Sine Wave Generator into the system; of course, you need to set the proper parameters for the Sine Wave based on your previous calculations. Again, you need to observe the individual harmonics using the Scope, and observe the change of the system output waveform in the Scope window when the higher-order harmonics are added to the system. Generate a square wave with the specifications in the Preparation section directly from the Wave Generator block. Compare your simulated result with this square wave by superimposing the two waveforms. Are they consistent? Can you determine how many harmonics are needed to

achieve this accuracy? Is your discovery consistent with your estimate from the Preparation section? Is it consistent with the spectrum you observed?

3. Print the simulated square wave for your report.
4. Save this working model to disk as a Matlab .mdl file.
5. Change the phase and magnitude of each harmonic and note that as far as the waveform shape is concerned, the phase of a harmonic is much more important than the magnitude. Try this step a few times, and print out a waveform that supports this conclusion based on your observations.
6. Repeat these steps for the triangular wave in the Preparations section.

#### **6.4 Simulink Part of the Lab Report**

Please include the following items in your lab report.

1. Print a copy of your design.
2. Print copies of the Scope input and output.
3. Answer all questions regarding your design.

#### **6.5 Communication Toolbox**

Open the Communication Toolbox located in Blocksets&Toolboxes→Comm Tbx Library. This shows all of the elements available to a communication system. Open several of the blocks in order to see what functions are available; for example, look at Modulation and Analog Modem. Within Analog Modem, there are several modulation techniques that were studied in ECE 146A. Although many topics will not be covered in this course, try experimenting with some of the blocks to familiarize yourself with what is available. If you need further help, there is an on-line tutorial in PDF form for the Communications Toolbox which provides excellent descriptions of the available functions and blocks.

#### **6.6 Communication Toolbox Part of the Lab Report**

Provide a discussion of one of the demonstrations that you studied.

### **7 Pulse Amplitude Modulation**

Uniform sampling is implemented using a carrier pulse train with the appropriate frequency. In order to produce natural samples, the Matlab sampling frequency should be on the order of 50-100 times that of the carrier pulse train, even with a 10% duty cycle. The system sampling rate should be at least twice the highest frequency of the message signal in order to satisfy the Nyquist rate. Use the pulse generator to implement the carrier pulse train.

## 7.1 Natural Sampling

Natural sampling is achieved when the carrier pulse train is multiplied by the modulating signal. In this case, the pulses take on the shape of the message signal.

## 7.2 Aliasing

To observe the effects of aliasing, select a sinusoidal waveform for the message signal and use natural sampling with the parameters: 1000 total samples, 1000 Hz pulse train with a 50% duty cycle, and a 100 Hz message signal. A Simulink sampling time of 0.00001 will ensure that these signals are sufficiently represented. Increase the signal frequency by factors of two to 200 Hz, 400 Hz, and 800 Hz. In the spectrum, verify the observed frequency for each case.

(1) Explain your observations. What change in the spectrum do you observe with the last value?

Now change the message signal to a 100 Hz triangular wave, and use natural sampling with the parameters: 1000 total samples and a 1100 Hz pulse train with a 25% duty cycle. This should have a broad spectrum with lines at odd multiples of 100 Hz. However, with the sampling frequency at 1100 Hz, the image spectrum (centered at 1100 Hz) will have certain harmonics which are aliased down to the baseband.

(2) Determine from the spectrum which are the desired harmonics and which are the aliased components.

Finally, apply a low-pass filter with a sharp cutoff at 500 Hz to the triangular waveform before it modulates the pulse train. Although the filtered triangular wave will be smoother, we are interested in the spectrum of the filtered waveform before and after the PAM operation. Use a couple of different frequencies for the triangle waveform message signal.

(3) Is there any difference in the effect of the PAM operation on the spectrum compared to the unfiltered case? Explain.

## 7.3 Reconstruction and Imaging

The sampled signal can be reconstructed by extracting any of the sidebands or the baseband itself. The former requires frequency shifting followed by low-pass filtering, or alternatively bandpass filtering followed by frequency shifting. The latter only requires low-pass filtering. Starting with the 50 Hz triangular waveform, apply a low-pass filter with 500 Hz cutoff, process with a PAM circuit (using natural sampling), and low-pass filter again with a 500 Hz cutoff.

(4) Do you obtain the original modulating waveform after all this processing? Explain your result.

Next, examine the effect of imaging caused by the inclusion of image frequencies in the reconstructed spectrum. If the demodulator low-pass filter has a cutoff frequency higher than half the sampling frequency, some of the energy in the first spectral image is allowed to pass. To observe this effect, make your second low-pass filter cutoff at 750 Hz instead of 500 Hz.

(5) How do the waveforms and the spectra here compare to those of the previous case?

## 7.4 Flat-Top Sampling

Flat-top sampling is achieved by designing a sample-and-hold circuit that is incorporated into the modulator. The pulse train has a constant amplitude for each sample (hence the term flat-top). One way of implementing flat-top sampling is to first create a staircase approximation to the message signal by multiplying it with a 100% duty cycle carrier pulse train. Natural sampling is then applied to this staircase approximation.

Perform flat-top sampling on the triangular waveform, first using only the staircase approximation and then sampling it using a 50% duty cycle. Thus, you will have two flat-top sampled signals: one with a 100% duty cycle and one with a 50% duty cycle.

- (6) How do the spectra of the two flat-top sampled signals differ from each other and with respect to the spectrum of the naturally sampled signal?

Finally, design a reconstruction filter for processing the flat-top sampled waveform.

- (7) How does the reconstructed signal compare to the original waveform and to its counterpart under natural sampling?

## 7.5 Lab Report

Please include the following in your lab report.

1. Printouts of all Simulink models designed for the PAM section of this lab.
2. Answers to all questions, as well as supporting comments regarding the theory versus your simulation results.
3. A paragraph containing questions and confusions you experienced in this lab.