GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2026 Fall 2013
Lab #1: Waveform Plotting, and Reading, Recording and Writing Signals

Date: 26 – 29 August 2013

All labs are held in room 2440 of the Klaus building. Your GT login will work, as long as you specify the "Windows domain" to be AD. If you have difficulty logging in, send an E-mail to

help@ece.gatech.edu,
or visit the web site: **www.ece-help.gatech.edu**

If you have MATLAB installed on your laptop, you are welcome to use your laptop in lab instead of the lab desktop computers if you wish.

General Lab Instruction

Pre-Lab: You should read the Pre-Lab section of the lab and do all the steps in the Pre-Lab section ***before your assigned lab time.*** For this lab in particular, you are advised to read through the exercise as well before coming to the lab so that you know what are involved.

Exercise and Verification: The Exercise section must be completed **during your assigned Lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by signing on the **Instructor Verification** line. When you have completed a step that requires verification, raise your hand and demonstrate the step to the TA or instructor.

Lab Questions: *Instructor Verification* is part of the Lab Sheet which has another component with a few lab related questions that need to be answered at your own pace. The completed lab sheet is, unless noted otherwise, due at the beginning of the next lab.

The predecessor to the three-credit-hour class ECE2026, which was a four-credit-hour class called ECE2025, consisted of labs that require a substantial amount of work to be done outside of lab and an elaborate lab report. To change the workload commensurate with losing one credit hour, we have reworked the labs such that they can be completed in individual lab sessions, without requiring a separate lab report. To make this work well, you will need to ***come into lab prepared***.

Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students and you are allowed to consult old lab reports, but you cannot give or receive written material or electronic files. Your submitted work should be original and it should be your own work.

PRINTING BUDGET: For the printers in the ECE labs, you have a quota. Please limit your printing to essential items for the labs. If you need to print lecture slides and other large documents, use the central (OIT) printing facilities.

---

1. Introduction

The objective of this lab is to learn to "acquire" signals into MATLAB and to display them for visualization. Waveform visualization is an important tool as well as a skill in signal analysis. Signals we learn to process in this course may be generated or recorder internally as a data array within MATLAB, or acquired from external sources such as a **`wav`** file. Furthermore, with a proper format converter, you may also record your own voice or music with your own computer and then load the recorded results, after converting them to the supported formats, into MATLAB for processing to produce interesting effects.

2. Pre-Lab

Please read through the information below prior to attending your lab.

2.1 Overview

In Lab00, you have gained or regained familiarity about MATLAB, particularly in terms of arithmetic operations. Using simple examples, we learned to assign data values to a variable such as:

```
x = 10;
y = [1 2 3 4 5];
yy = [1 2 3; 4 5 6];
```

A slightly more sophisticated example in Lab00 was:

```
xx = -0.1:0.01:0.5 ;
```

which will create a row vector of dimension 61, with values that span between -0.1 and 0.5 (inclusive) at an increment of 0.01. In this lab, we'll learn to use these ready expressions to generate some "signals."

Once a signal is generated, an immediate tool we use very often is to plot it for visualization and inspection. You'll learn to use common plot routines to show, in a number of ways, the signal you have generated or acquired (see below).

Often times, we need to process a signal that is obtained by some other means, instead of being generated "internally" with MATLAB commands. For example, you may have digital music on your computer, stored as files. Or, you may use your own computer to record a sound or your speech, which may also be stored as a file electronically. In this lab, we'll also learn to read or load external data into MATLAB for processing. We'll also learn to "write" the processed data or signal into a file for future consumption.

2.2 Microphone and Headset/Loudspeaker

For this lab and many future labs, you'll need a microphone and a headphone. Many laptops already come with these devices. In case you need to acquire one set of these devices, there are many cheap selections over the Internet. You do not need to become an audiophile and spend big money on these; keep it cheap and simple. (Of course, if you have a passion towards audio and music, you may have already owned some really nice headphones and microphones. Use them.)

2.3 Getting Prepared

The following is a set of commands that you will learn and use in this lab. Read MATLAB documentations with regard to these commands first, before you come to the lab:

`plot; subplot; figure; hold; input; wavread; wavwrite; audiorecorder; record; play; getaudiodata`

In case you have an early version of MATLAB that does not support **`audiorecorder`** and related commands, use the provided **`wav`** file. You may try to use these commands before coming to the lab.

3. Exercise

3.1 Interactive Input in MATLAB

MATLAB allows the user to provide input through interactive commands. This may be handy, particularly when you want the user to be able to specify some data values or strings in a function (**.m** file) during the so-called run-time. Here is a function example (cubic root of) that involves a user input:

```
function a = crootof
x = input('what: ');
if x >= 0
    a=(x)^(1/3);
else
    'non-negative number only'
end
```

Then, in MATLAB, the following shows one application of the command **crootof**:

```
>> crootof
what: 287318
ans =
   65.9864
```

where the number 287318 was given by the user.

A user can use the **input** command to provide text strings to MATLAB as well. This is done by specifying a **'s'** argument in the command line:

```
>> ss= input('Text input: ','s')
Text input: ECE2026 signal processing
ss =
ECE2026 signal processing
>> ss(1)
ans =
E
```

The last three lines above show that the first character of the string **ss** is an **E**. Now, write a MATLAB code that asks the user to input a string (say **Mary has a little lamb**), and then prints out the characters at odd locations of the input string (in this example, **Mr_a__itelm**). Try other examples as well.

Instructor's Verification

3.2 Generating Sinusoids

An example for generating a sinusoid was included in Lab00. Here, we will again use the built-in MATLAB editor to create a script file called **asinusoid.m** containing the following lines for future use:

```
function xs = asinusoid(amp, freq, pha, fs, tsta, tend)
% amp  = amplitude
% freq = frequency in cycle per second
% pha  = phase, time offset for the first peak
% fs   = number of sample values per second
% tsta  = starting time in sec
% tend  = ending time in sec
tt = tsta : 1/fs : tend; % time indices for all the values
xs = amp * cos( freq*2*pi*tt + pha );
end
```

This function **asinusoid** can be called once the argument values are specified. For example,

```
amp  = 2.2;
freq = 100;
pha  = pi/3;
fs   = 8000;
tsta = 0;
```

```
tend = 3;  %a 3-sec long signal
xs = asinusoid(amp, freq, pha, fs, tsta, tend);
%<--- plot first three cycles of the generated sinusoid
ts = tsta:1/fs:tsta+3/freq;
le = length(ts);
plot( ts, xs(1:le), 'b-', ts, xs(1:le), 'r--' ), grid on
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

You may want to try other numbers to appreciate the use of the function. In lecture, you will learn or have learned more about a sinusoid, what these parameters mean, and many properties thereof. Here we are most interested in the plotting tool; so you may want to read its documentation carefully.

Now, change all relevant numbers in the above example so as to generate and plot two other sinusoids **xs1** and **xs2**, each 3 seconds long, one with a frequency of 400 cycles/s, an amplitude of 1.6, and a phase of 1.2, and the other with frequency 410, amplitude 2.1, and phase 0. Then, add these two sinusoids together to form **xss = xs1 + xs2**. Plot **xss** between time 0.7s and 1.0s. Show the plotting result to the instructor.

Instructor's Verification

3.3 Reading WAV File into MATLAB and Playing an Array

Many sound files are stored in **.wav** format. For information about this particular format, see http://en.wikipedia.org/wiki/WAV. Other formats are available, such as **mp3**; we'll focus on **.wav** in this lab. The MATLAB command **wavread** is the one to use to read data in **wav** files into MATLAB data arrays. For example:

```
xx = wavread('myvoice.wav');
```

will load the data in the file **myvoice.wav** into the array **xx**. Another command **length(xx)** will tell you how many values have been read into **xx**. In many applications, you may need to know what is called the sampling rate, which is the number of sample values per second at which the data was acquired into the file. These parameters can be retrieved as part of the **wavread** result:

```
[xx, fs, nbit] = wavread('myvoice.wav');
```

where **xx** is the data array, **fs** is the sampling rate, and **nbit** is the number of bits used to represent each value in the file **myvoice.wav.**

For this exercise, you need to access some **wav** files. If you have your own inventory of **wav** files, you are welcome to use them here. (Make sure the file is long enough for the exercise; see below). If not, a file called **ece2026lab1.wav** can be downloaded from **t-square** for your use. Learn to use the **wavread** command to read the data into an array, say **xx**. You can find out how long the signal is in seconds by reading the length of the array via **length(xx)** which gives you the total number of samples and then divide it by **fs** , the sampling rate in samples per second. Then, plot the sound wave **xx**, from t = 0.25 to 0.5. You need to know how to translate time into the index of the array; an example has already been shown in Section 3.2. Show the plotted result to the instructor.

Instructor's Verification

In MATLAB, an array of data can be "played" to produce audible sound. To do so, you can use:

```
soundsc(xx, fs);
% what is xx and fs??? Make sure you know what these are
```

or

```
wavplay(xx, fs);
```

Try these commands yourself and listen to the data you just read in from the file **ece2026lab1.wav.**

3.4 Processing the Data and Writing the Result into a `wav` File

3.4.1 Time reversal

Following the previous section, you have an array **`xx`** that contains some sound. Let's perform some simple processing here. First, let's attenuate the signal to half,

```
xh = xx * 0.5;
```

where the new array **`xh`** has all the sample values halved from the original signal. Then, we want to reverse the time and write the signal out to a **`wav`** file:

```
le = length(xh);
xhr = xh(??:??:??);   % figure out how to fill in those ??s
wavwrite(xhr , fs , 'ECE2026lab1outa.wav');
```

Note that the argument **`fs`** can be copied from the **`wavread`** result. But you can experiment with your own number, such as reducing it to half or doubling it to twice the original value. Using **`soundsc`** or **`wavplay`**, play the output **`wav`** file **`ECE2026lab1outa.wav`** to the instructor for verification.

Instructor's Verification

3.4.2 Spectrum Reversal

Now you need to figure out a way to achieve the following: alternately change the sign of the values in the signal array. That is,

$$y[n] = \begin{cases} x[n], & n \text{ odd} \\ -x[n], & n \text{ even} \end{cases}$$

For example, if $x[0] = 1.1, x[1] = 1.2, x[2] = -0.3, x[3] = -1.8, \cdots$, you want to produce a signal that is $y[0] = 1.1, y[1] = -1.2, y[2] = -0.3, y[3] = 1.8, \cdots$. (Note that in a vector the index starts from 1 but in DSP we start from 0. This discrepancy in indexing is going to stay for quite some time, unfortunately.)

Once you manage to accomplish the above, write the output to a **`wav`** file **`ECE2026lab1outb.wav.`** Show the code and play the result to the instructor for verification.

Instructor's Verification

3.5 Recording and Playing Sounds in MATLAB (Optional after Lab session)

You can also record your own sound with MATLAB. To record data from an audio input device (such as a microphone connected to your system) for processing in MATLAB, follow the sequence below:

- Create an **`audiorecorder`** object:

```
>> audiobject = audiorecorder(8000, 16, 1);
% an object here is like a device or channel
% fs = 8000; use 8000 values for each second of sound
% nbit = 16; use 2 bytes to represent a value
% nchannel = 1; use 2 for stereo recording...
% (need stereo microphone)
% you may try other numbers
```

- Call **`record`** or **`recordblocking`**:

```
>> record(audiobject)
>> stop(audiobject)
```

```
% record opens a channel that links to audiobject
% stop closes the channel; if you do not close the channel..
% recording continues and audiobject cannot be accessed
```

or

```
>> recordblocking(audiobject, 5);
% record a fixed duration of 5 seconds into audiobject
```

- Create a numeric array corresponding to the signal data using the **getaudiodata**:

```
>> xx = getaudiodata(audiobject);
```

For example, connect a microphone to your system and record your voice for 5 seconds. Capture the numeric signal data and create a plot (taken from Mathworks):

```
% Record your voice for 5 seconds.
recObj = audiorecorder(8000, 16, 1);
disp('Start speaking.')
recordblocking(recObj, 5);
disp('End of Recording.');

% Play back the recording.
play(recObj);

% Store data in double-precision array.
myRecording = getaudiodata(recObj);

% Plot the samples.
plot(myRecording);
```

In the above, the array **myRecording** contains the voice data, which can be processed, e.g., as in 3.4.1 and 3.4.2 above. The results can be written into a **wav** file using the aforementioned **wavwrite** command. If you choose to do this, mark the box in the Lab sheet and send the **wav** files to your TA for comment.

Lab #1
ECE-2026 Fall-2013
LAB SHEET

Turn this page in to your grading TA at the beginning of the next lab

Name: _____ Date of Lab: _____

Part 3.1 Interactive input
Write out the string output below and show it to the instructor. Verified: _____

Part 3.2 Generating sinusoids
Show the specified plot of sum of two sinusoids to the instructor. Verified: _____

Part 3.3 Reading data from a **`wav`** file
Show the specified plot of the sound wave read from the file. Verified: _____

Part 3.4.1 Processing the data and writing the result to a file
Play the output **`wav`** file to the instructor. Verified: _____

Part 3.4.2 Processing the data and writing the result to a file
Play the output **`wav`** file to the instructor. Verified: _____

Part 3.5 I have done the optional recording exercise. Audio files have been sent to the grading TA. _____

Questions:

Question 2-1: What are the sound file formats that can be read into MATLAB data arrays?

Question 2-2: A tone signal is a single sinusoid that can be heard by our ears. Suppose a tone signal of frequency 500Hz and of 3 seconds in duration in the file **`myvoice.wav`** is read into an array by the following code:

```
[xx, fs, nbit] = wavread('myvoice.wav');
```

Further suppose it is played with the following line of code:

```
wavplay(xx,fs/2)
```

How long will the signal last in seconds? What will be the frequency of the perceived tone? (It is okay for you to guess here as long as you can provide reasoning to justify your guess.)

Output duration = s	Perceived frequency = Hz