GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2026 Fall 2013

Lab #3: Synthesizing Signals: AM, FM and Chirps

Date: 16-19 September 2013

Each Lab assignment in ECE2026 consists of three parts: Pre-Lab, In-lab Tasks, and Take-home Questions. It requires you to come into lab prepared. Be sure to read the entire lab carefully before arriving.

Pre-Lab: You should read the Pre-Lab section of the lab and go over all exercises in this section before going to your assigned lab session. Although you do not need to turn in results from the Pre-Lab, doing the exercises therein will help make your in-lab experience more rewarding and go more smoothly with less frustration and panicking.

In-lab Tasks and Verification: There are a number of designated tasks for each student to accomplish during the lab session. Students are encouraged to read, prepare for and even attempt at these tasks beforehand. These tasks 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 in¬structors must verify the appropriate steps by signing on the Instructor Verification line. When you have completed a step that requires verification, simply put a plastic cup on top of your PC and demonstrate the step to one of the TAs or the professor. (You can also use the plastic cups to indicate if you have a more general question, i.e. you can use it to get our attention even if you don't have an Instructor Verification ready.)

Take-home Questions: At the end of each lab sheet below all the verification steps, several questions are to be answered by the student, who can choose to complete the answers while in the lab or after leaving the lab.

The lab sheet with all verification signatures and answers to the questions needs to be turned in to the Lab-grading TA at the beginning of the next lab session.

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 & Overview

The objective of this lab is to introduce more complicated signals that are related to the basic sinusoid. These signals which implement frequency modulation (FM) and amplitude modulation (AM) are widely used in communication systems such as radio and television, but they also can be used to create interesting sounds that mimic musical instruments or have some peculiar perceptual effects. There are a number of demonstrations on the CD-ROM that provide examples of these signals for many different conditions. In this lab, we'll only try to synthesize the simple ones; we'll save more sophisticated ones for later.

CD-ROM

SP-First MATLAB *Toolbox*

2 Pre-Lab

We have learned about the properties of sinusoidal waveforms of the form:

$$x(t) = A\cos(2\pi f_0 t + \varphi) = \mathrm{Re}\left\{Ae^{j\varphi}e^{j2\pi f_0 t}\right\} \quad (1)$$

(1) is in fact a special, albeit fundamental, case of the following signal

$$x(t) = \mathrm{Re}\left\{A(t)e^{j\theta(t)}\right\} \quad (2)$$

where $A(t)$ is the amplitude function and $\theta(t)$ is the angle function, both of which can vary with time. The sinusoid signal of (1) simply has an amplitude function that is constant and an angle function that is a first-order function of time. In this lab, we will extend our treatment of sinusoidal waveforms to more complicated signals in which the amplitude and the angle function may no longer be as simple as the sinusoidal case.

2.1 Amplitude Modulation

If we multiply two sinusoids together, we can express the result as:

$$x(t) = A_1 \cos(2\pi f_1 t + \varphi_1) \bullet A_2 \cos(2\pi f_2 t + \varphi_2) \tag{3}$$

where $\{A_k, f_k, \varphi_k\}$, $k = 1, 2$, are the parameters for the two sinusoids, respectively. In reference to (2), one of the two sinusoids can be considered as the amplitude function, which changes with time. This is the simplest case of an amplitude-modulated signal; i.e., the amplitude of a sinusoid is "modulated" by another sinusoid. Using Euler's formula, we can rewrite the expression of (3) as

$$\begin{aligned} x(t) &= A_1 \cos(\omega_1 t + \varphi_1) \bullet A_2 \cos(\omega_2 t + \varphi_2) \\ &= A_1 \frac{e^{j(\omega_1 t + \varphi_1)} + e^{-j(\omega_1 t + \varphi_1)}}{2} \bullet A_2 \frac{e^{j(\omega_2 t + \varphi_2)} + e^{-j(\omega_2 t + \varphi_2)}}{2} \\ &= \frac{A_1 A_2}{4} \left\{ \left[e^{j[(\omega_1 + \omega_2)t + (\varphi_1 + \varphi_2)]} + e^{-j[(\omega_1 + \omega_2)t + (\varphi_1 + \varphi_2)]} \right] + \left[e^{j[(\omega_1 - \omega_2)t + (\varphi_1 - \varphi_2)]} + e^{-j[(\omega_1 - \omega_2)t + (\varphi_1 - \varphi_2)]} \right] \right\} \\ &= A_1 A_2 \left\{ \cos[(\omega_1 + \omega_2)t + (\varphi_1 + \varphi_2)] + \cos[(\omega_1 - \omega_2)t + (\varphi_1 - \varphi_2)] \right\} \end{aligned} \tag{4}$$

which turns out to be sum of two sinusoids of equal amplitude. The "Beat Control GUI" **`beatcon`** is part of the *SP First* toolbox for your exploration of simple AM signals.

In AM broadcasting, the radio signal takes the form of:

$$x(t) = (1 + \alpha s(t)) \cos(\omega_c t + \varphi)$$

where ω_c is the so-called carrier frequency, usually in the range of 531–1,611kHz (medium wave, see your AM radio frequency dial), to allow electromagnetic wave propagation, and $s(t)$ is the signal, voice or music, that you will hear from loudspeaker. Parameter α is the modulation index, detail about which is not the focus of our class.

2.2 Frequency Modulation (FM)

We will also look at signals in which the *frequency* may vary as a function of time. We define (instantaneous) frequency as the time-derivative of the angle function

$$\omega_i(t) = \frac{d}{dt}\theta(t); \quad \text{or,} \quad f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\theta(t) \quad \text{in Hz} \tag{5}$$

For the case of (1), $\theta(t) = \omega_0 t + \varphi$

$$\omega_i(t) = \frac{d}{dt}[\omega_0 t + \varphi] = \omega_0, \quad \text{a constant}$$

which results in a simple sinusoid. The angle function can be of a general time-varying function, and it is called an FM signal, particularly when the amplitude is held constant. In FM radio, the signal transmitted over the airwave would take the form:

$$x(t) = A\cos\{[\omega_c + \tfrac{\Delta\omega_0}{B} s(t)]t\}$$

where $s(t)$, like in the AM case, may be a music or speech signal, and B is some constant, in conjunction with $\Delta\omega_0$, to control the amount of frequency deviation around the carrier frequency, ω_c, relative to the input signal $s(t)$. The time-varying frequency of such a signal has the following form:

$$\omega_i(t) = \tfrac{d}{dt}\left\{[\omega_c + \tfrac{\Delta\omega_0}{B} s(t)]t\right\} = \omega_c + \tfrac{\Delta\omega_0}{B} s(t) + t\,\tfrac{\Delta\omega_0}{B}\left[\tfrac{d}{dt} s(t)\right]$$

In this lab, we will explore some simple cases of FM signal, in which the instantaneous frequency changes with time according to a well-defined function.

2.3 Chirp Signals

A linear-FM chirp signal is a sinusoid whose frequency changes linearly from a starting value to an ending one. The formula for such a signal can be defined by creating a complex exponential signal with a quadratic angle function by defining $\theta(t)$ in (2) as

$$\theta(t) = 2\pi\mu t^2 + 2\pi f_0 t + \varphi \tag{6}$$

The derivative of $\theta(t)$ yields an instantaneous frequency (5) that changes linearly versus time:

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\theta(t) = 2\mu t + f_0$$

The slope of $f_i(t)$ is equal to 2μ and its intercept at $t = 0$ is equal to f_0. If the signal starts at time $t = 0$ s, then f_0 is also the starting frequency. Since the linear variation of the frequency can produce an audible sound similar to a siren or a chirp, the linear-FM signals are also called chirps. More elaborate siren sounds can be generated by choosing a proper $\theta(t)$.

The following subsection will help you get started with synthesizing a chirp signal.

2.3.1 MATLAB Synthesis of Chirp Signals

The following MATLAB code will synthesize a chirp:

```
fsamp = 10000;
dur = 1.5;
tt = 0 : 1/fsamp : dur;
f1 = 400;
mu = 1000;
psi = 2*pi*(0.1 + f1*tt + mu*tt.*tt);
xx = real( 10*exp(j*psi) );
soundsc( xx, fsamp );
```

Do the following before going to the lab session:

(a) Determine the total duration of the synthesized signal in seconds, and also the length of the **`tt`** vector.

(b) In MATLAB, signals can only be synthesized by evaluating the signal's defining formula at discrete instants of time. These are called *samples* of the signal. For the chirp we do the following:

$$x(t_n) = A\cos(2\pi\mu t_n^2 + 2\pi f_0 t_n + \varphi)$$

where t_n is the n^{th} time sample. In the MATLAB code above, identify the values of A, μ, f_0 and φ.

(c) Determine the range of frequencies (in Hertz) that will be synthesized by the MATLAB script above, i.e., determine the minimum and maximum frequencies (in Hz) that will be heard. This will require that you relate the defining parameters to the minimum and maximum frequencies. Make a sketch by hand of the instantaneous frequency versus time; make sure your labels are correct.

(d) Use **`soundsc()`** to listen to the signal in order to determine whether the signal's frequency content is increasing or decreasing. Notice that **`soundsc()`** needs to know two things: the vector containing the signal samples, and the rate at which the signal samples were created (**`fsamp`** in the code above). If you do not include an explicit number for the sampling rate, the program default is 8000 in **`soundsc`**. For more information do **`help sound`** and **`help soundsc`** in MATLAB.

(e) Now, change the duration to 4 s (**`dur = 4`**) and repeat (a)—(d) above. Describe and try to explain what you hear.

3 Exercise

3.1 Amplitude Modulation

3.1.1 Beat Notes

In (4), we see an equality that relates sum of sinusoids and product of sinusoids. Most of the time, we hear "multiple tones" when listening to a sound that consists of two (or more) sine waves; that is, we perceive the sound as "sum" of sinusoids. However, when the frequencies of the two sinusoidal components are close to each other, we hear what is commonly described as "beats" or "beat notes."

To assist you in your experiments with beat notes and AM signals, the MATLAB GUI tool called **`beatcon`** has been created. This *user interface controller* will exhibit the basic signal shapes for beat signals and play the signals. A small control panel will appear on the screen with *buttons* and *sliders* that vary the different parameters for the beat signals. Experiment with the **`beatcon`** control panel and use it to produce a beat signal with two frequency components: one at 790 Hz and the other at 810 Hz. Use a longer duration than the default to hear the *beat frequency* sound. Determine the beat frequency.

Instructor Verification (separate page)

3.1.2 Amplitude Modulated Speech

One may be curious about how a speech signal would sound like when used to amplitude-modulate a sinusoid. Perform the following:

a. Read in a speech signal, $s(t)$, from the file **`Lab3voice.wav`** (available from **`t-square`**);

b. Generate a sinusoid with carrier frequency ω_c (see Lab02 Section 3.2);

c. Create a signal $x(t) = s(t)\cos(\omega_c t)$; then play and listen to $x(t)$.

Perform the above with three carrier frequencies, ω_c =25, 400, and 1000 Hz respectively, save the output signals in three files (**`Lab3out`***x*`.wav`, *x*=1, 2, 3), and describe what you hear on the verification sheet. You may be randomly asked to mail one of the output files to the grading TA. Once the three signals have been saved, ask for instructor verification.

Instructor Verification (separate page)

3.2 Function for a Chirp

Use the code provided in the pre-Lab section as a starting point in order to write a MATLAB function that will synthesize a "chirp" signal according to the following template. This will require that you relate the chirp parameters μ, f_0, and φ to the starting and ending frequencies. Fill in code where you see **`???`**.

```
function [xx,tt] = make_chirp( f1, f2, dur, fsamp )
%MAKE_CHIRP generate a linear-FM chirp signal
%
% usage: xx = make_chirp( f1, f2, dur, fsamp )
%
% f1 = starting frequency
% f2 = ending frequency
% dur = total time duration
% fsamp = sampling frequency
% this template code does not consider the initial phase, phi
%
% xx = (vector of) samples of the chirp signal
% tt = vector of time instants for t=0 to t=dur
%
if( nargin < 4 ) %-- Allow optional input argument
fsamp = 10000;
```

```
end
tt = ???
psi = 2*pi*( ???*tt + ???*tt.*tt);
xx = real( exp(j*psi) );
```

As a test case, generate a chirp sound whose frequency starts at 2200 Hz and ends at 500 Hz; its duration should be 1 sec and the sampling rate should be f_s = 10000 samples/sec. Listen to the chirp using the **`soundsc`** function and play it to the instructor for verification. Give the exact calling sequence for **`make_chirp.m`** in order to produce the test case.

Instructor Verification (separate page)

3.3 Spectrograms

It is often useful to think of a signal in terms of its spectrum. A signal's spectrum is a representation of the frequencies present in the signal, each having its own complex amplitude. For a constant frequency sinusoid as in (1) the spectrum consists of two spikes, one at $\omega = 2\pi f_0$, the other at $\omega = -2\pi f_0$. For a more complicated signal the spectrum may be very interesting, e.g., the case of FM, where the spectrum components are time-varying. One way to represent the time-varying spectrum of a signal is the *spectrogram* (see Chapter 3 in the text). A spectrogram is produced by estimating the frequency content in short sections of the signal. The magnitude of the spectrum over individual sections is plotted as intensity or color on a two-dimensional plot versus frequency and time. (We'll come back and learn more about the math behind spectrogram in the future.)

CD-ROM

Spectrogram & Sounds

When unsure about a command, use **`help`**.

There are a few important things to know about spectrograms:

a. In MATLAB the function **`spectrogram`** will compute the spectrogram. Type **`help spectrogram`** to learn more about this function and its arguments. If you have an older version of MATLAB, the function is called **`specgram`**.

b. If you are working at home, you might not have the **`spectrogram()`** function as it is part of the *Signal Processing Toolbox*. In that case, use the function **`plotspec(xx,fs)`** which is part of the *SP-First Toolbox* which can be downloaded from

http://users.ece.gatech.edu/mcclella/SPFirst/Updates/SPFirstMATLAB.html

• Note: The argument list for **`plotspec()`** has a different order from **`spectrogram`**, because **`plotspec()`** uses an optional third argument for the *window length* (default value is 256). In addition, **`plotspec()`** does not use color for the spectrogram; instead, darker shades of gray indicate larger values with black being the largest.

c. Spectrograms are numerical calculations and provide only an estimate of the time-varying frequency content of a signal. There are theoretical limits on how well they can actually represent the frequency content of a signal. Another lab on the *SP-First* CD-ROM treats this problem by using the spectrogram to extract the frequencies of piano notes.

d. A common call to the function is **`spectrogram(xx,512,384,512,fs,'yaxis')`**. The second and the third arguments[1] are the *window length* and *overlap length*, respectively, which could be varied to get different looking spectrograms. In the example above, successive chunks of 512 data points with an overlap of 384 points are used to generate the spectrogram. The fourth argument is the size of Fast Fourier Transform (FFT) and **`fs`** is the sampling rate. You use the string argument **`'yaxis'`** to specify

[1]If the second argument of **`spectrogram`** is made equal to the "empty matrix" then the default value of 256 is used.

the orientation of the plot, i.e., with the y-axis signifying the frequency and x-axis the time. The spectrogram is able to better "see" the separate spectrum lines with a longer window length[2], e.g., 1024.

e. **Frequency Range:** Normally the spectrogram image contains only positive frequencies. However, you can produce a spectrogram image containing negative frequencies if you use the function **`plotspec`** and if you make the input signal complex. Even if your signal is real, you can add a very tiny imaginary part, e.g., **`xx = xx + j*1e-14`**, to make it seem to be complex-valued.

Warning: This trick works nicely with the *SP-First* function called **`plotspec`**. However, when used with **`spectrogram`** it produces an image that does not have the negative frequency region in the proper location.

In order to see a typical spectrogram, run the following code:

```
fs=10000;   xx=cos(2000*pi*(0:1/fs:0.5));
spectrogram(xx,512,384,512,fs,'yaxis');   colorbar
```

or, if you are using **`plotspec(xx,fs):`**

```
fs=10000; xx=cos(2000*pi*(0:1/fs:0.5)); plotspec(xx,fs,1024); colorbar
```

Notice that the spectrogram image contains one horizontal line at the correct frequency of the sinusoid. For a spectrogram with negative frequencies, try the following

```
xx = cos(2000*pi*(0:1/fs:0.5)); plotspec(xx+j*1e-9,fs,1024); colorbar
```

Now, you will display the spectrogram of the chirp synthesized in the previous section. You can use command **`spectrogram`** to plot the spectrogram of a signal. For example:

```
spectrogram(xx,512,384,1024,fsamp,'yaxis')     % plot a 1-sided spectrogram
```

Or, if you are interested in tracking the negative side of the frequency axis, the following example will show a 2-sided spectrogram:

```
fres = 10;                        % define frequency resolution
FF = -fsamp/2:fres:fsamp/2        % define freq range in spectrogram plot
spectrogram(xx,512,384,FF,fsamp,'yaxis')      % plot 2-sided spectrogram
```

Now show the spectrogram, both 1-sided and 2-sided, of the chirp produced in the previous part, Section 3.2.

Instructor Verification (separate page)

[2] Usually the window length is chosen to be a power of two, because a special algorithm called the FFT is used in the computation. The fastest FFT programs are those where the signal length is a power of 2.

Lab #3
ECE-2026 Fall-2013
LAB SHEET

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

Name: _____ Date of Lab:_____

Part 3.1.1 Jot down the beat frequency here _____ and play the sound to the instructor:

Verified: _____ Date/Time:_____

Part 3.1.2 Qualitatively write below what you hear in the three results. Instructor specify which output file to submit _____.
Your observations:

Verified: _____ Date/Time:_____

Part 3.2 Write the calling sequence below and play the synthesized chirp to the instructor.

`Make-chirp` call sequence:

Verified: _____ Date/Time:_____

Part. 3.3 show the spectrogram, both 1-sided and 2-sided, of the chirp produced in 3.2 to the instructor for verification.

Verified: _____ Date/Time:_____

Questions:

Question 3.1: A chirp is produced by the following code. (Try it yourself.)

```
tt=0:1/8000:3; theta=2*pi*(600*tt+6*cos(2*pi*5*tt)); xx=cos(theta);
```

What are the maximum and the minimum instantaneous frequencies in the signal?

Max instantaneous freq:_____; Minimum instantaneous freq:_____

Question 3.2: Similar to Q3.1, now the code is changed to

```
tt=0:1/8000:3; theta=2*pi*tt.*(600+6*cos(2*pi*5*tt)); xx=cos(theta);
```

What are the maximum and the minimum instantaneous frequencies in the signal? What is the audible difference between the two chirp signals (Q3.1 and Q3.2)? Why?

Max instantaneous freq:_____; Minimum instantaneous freq:_____
Explanation: