GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

EE 2025 Spring 2005
Lab #3: AM and FM Sinusoidal Signals

Date: 1 – 7-Feb 2005

You should read the Pre-Lab section of the lab and do all the exercises in the Pre-Lab section ***before your assigned lab time.***

The Warm-up section of each lab 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, simply raise your hand and demonstrate the step to the TA or instructor. After completing the warm-up section, turn in the verification sheet to your TA.

The exercises in Section 4 should written up in this week's lab report. More information on the lab report format can be found on Web-CT under the "Information" link. You should ***label*** the axes of your plots and include a title and Figure number for every plot. Every plot should be referenced by Figure number in your text discussion. In order to make it easy to find all the plots, include each plot *inlined* within your report. This can be done easily with MATLAB's `notebook` capability.

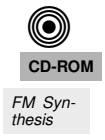
Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students and you are allowed to consult old lab reports, but you cannot give or receive written material or electronic files.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

The report will be **due during the period 8 to 14-Feb. at the start of your lab.**

---

1 Introduction

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. There are a number of demonstrations on the CD-ROM that provide examples of these signals for many different conditions.



2 Pre-Lab

We have spent a lot of time learning about the properties of sinusoidal waveforms of the form:

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

In this lab, we will extend our treatment of sinusoidal waveforms to more complicated signals composed of sums of sinusoidal signals, or sinusoids with changing frequency, i.e., frequency-modulated sinusoids.

2.1 Amplitude Modulation

If we add several sinusoids, each with a different frequency (f_k) we can express the result as:

$$x(t) = \sum_{k=1}^{N} A_k \cos(2\pi f_k t + \varphi_k) = \Re e \left\{ \sum_{k=1}^{N} \left(A_k e^{j\varphi_k} \right) e^{j2\pi f_k t} \right\} \tag{2}$$

where $A_k e^{j\varphi_k}$ is the complex amplitude of the k^{th} complex exponential term. The choice of f_k will determine the nature of the signal—for amplitude modulation or beat signals we pick two or three frequencies that are very close together, see Chapter 3.

2.2 Frequency Modulated Signals

We will also look at signals in which the frequency varies as a function of time. In the constant-frequency sinusoid (1) the argument of the cosine is $(2\pi f_0 t + \varphi)$ which is also the exponent of the complex exponential. We will refer to the argument of the cosine as the ***angle function***. In equation (1), the *angle function* changes *linearly* versus time, and its time derivative is $2\pi f_0$ which equals the constant frequency of the cosine.

A generalization is available if we adopt the following notation for the class of signals with time-varying angle functions:

FM Synthesis

$$x(t) = A\cos(\psi(t)) = \Re e\{A e^{j\psi(t)}\} \tag{3}$$

The time derivative of the angle function $\psi(t)$ in (3) gives a frequency

$$\omega_i(t) = \frac{d}{dt}\psi(t) \qquad \text{(rad/sec)}$$

but if we prefer units of hertz, then we divide by 2π to define the *instantaneous frequency:*

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\psi(t) \qquad \text{(Hz)} \tag{4}$$

2.3 Chirp, or Linearly Swept Frequency

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 quadratic angle function by defining $\psi(t)$ in (3) as

Spectrograms & Sounds: Wide-band FM

$$\psi(t) = 2\pi\mu t^2 + 2\pi f_0 t + \varphi$$

The derivative of $\psi(t)$ yields an instantaneous frequency (4) that changes *linearly* versus time.

$$f_i(t) = 2\mu t + f_0$$

The slope of $f_i(t)$ is equal to 2μ and its intercept is equal to f_0. If the signal starts at time $t = 0$ secs., then f_0 is also the starting frequency. The frequency variation produced by the time-varying angle function is called *frequency modulation,* and this class of signals is called FM signals. Finally, 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.*

2.4 MATLAB Synthesis of Chirp Signals

The following MATLAB code will synthesize a chirp:

```
fsamp = 8000;
dt = 1/fsamp;
dur = 1.1;
tt = 0 : dt : dur;
f1 = 400;
psi = 2*pi*(100 + f1*tt + 500*tt.*tt);
xx = real( 7.7*exp(j*psi) );
soundsc( xx, fsamp );
```

(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. Make a sketch by hand of the instantaneous frequency versus time. Determine the minimum and maximum frequencies (in Hz) that will be heard.

(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` above).
For more information do `help sound` and `help soundsc` in MATLAB.

3 Warm-up

The instructor verification sheet may be found at the end of this lab. The "Beat Control GUI" is part of the *SP First* toolbox, and it should be on the MATLAB path already installed on the ECE computers.

3.1 Beat Control GUI

To assist you in your experiments with beat notes and AM signals, the 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. It can also call a user-written function called `beat.m`. Experiment with the **`beatcon`** control panel and use it to produce a beat signal with two frequency components: one at 648 Hz and the other at 672 Hz. Use a longer duration than the default to hear the "beat." Demonstrate the plot and sound to your TA.

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. Fill in code where you see `???`.

```
function   [xx,tt] = syn_chirp( f1, f2, dur, fsamp )
%SYNCHIRP      generate a linear-FM chirp signal
%
%  usage:   xx = syn_chirp( f1, f2, dur, fsamp )
%
%      f1 = starting frequency
%      f2 = ending frequency
%     dur = total time duration
%   fsamp = sampling frequency  (OPTIONAL: default is 8000)
%
%      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 = 8000;
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 3000 Hz and ends at 200 Hz; its duration should be 0.8 sec and the sampling rate should be $f_s = 8000$ samples/sec. Listen to the chirp using the `soundsc` function. Give the exact calling sequence for `syn_chirp.m` in order to produce the test case.

Instructor Verification (separate page)

3.3 Advanced Topic: 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. 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, as in the case of FM, where the spectrum is considered to be 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 found 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.

Sounds & Spectrograms

When unsure about a command, use `help`.

There are a few important things to know about spectrograms:

1. In MATLAB the function `specgram` will compute the spectrogram. Type `help specgram` to learn more about this function and its arguments.

2. If you are working at home, you might not have the `specgram()` function because 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 WebCT.

   • Note: The argument list for `plotspec()` has a different order from `specgram`, 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.

3. 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 CD-ROM that accompanies the text treats this problem by using the spectrogram to extract the frequencies of piano notes.

4. A common call to the function is `specgram(xx,1024,fs)`. The second argument[1] is the *window length* which could be varied to get different looking spectrograms. The spectrogram is able to "see" the separate spectrum lines with a longer window length, e.g., 1024 or 2048.[2]

5. **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 `specgram` it produces an image that does not have the negative frequency region in the proper location.

In order to see what the spectrogram produces, run the following code:

```
fs=8000; xx = cos(2000*pi*(0:1/fs:0.5)); specgram(xx,1024,fs); colorbar
```

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

```
fs=8000; 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 show the spectrogram of the chirp produced in the previous part, Section 3.2.

Instructor Verification (separate page)

4 Lab: Chirps and Beats

For the lab exercise and lab report, you will synthesize some AM and FM signals. In order to verify that these signals have the correct frequency content, you will use the spectrogram. Your lab report should discuss the connection between the *time-domain* definition of the signal and its *frequency-domain* content.

4.1 Beat Notes

In the section on beat notes in Chapter 3 of the text, we analyzed the situation in which we had two sinusoidal signals of slightly different frequencies; i.e.,

$$x(t) = A\cos(2\pi(f_c - f_\Delta)t) + B\cos(2\pi(f_c + f_\Delta)t) \tag{5}$$

In this part, we will compute samples of such a signal and listen to the result.

(a) Write an M-file called `beat.m` that implements (5) and has the following as its first lines:

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

[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.

```
function     [xx, tt] = beat(A, B, fc, delf, fsamp, dur)
%BEAT    compute samples of the sum of two cosine waves
%  usage:
%       [xx, tt] = beat(A, B, fc, delf, fsamp, dur)
%
%              A = amplitude of lower frequency cosine
%              B = amplitude of higher frequency cosine
%             fc = center frequency
%           delf = frequency difference
%          fsamp = sampling rate
%            dur = total time duration in seconds
%             xx = output vector of samples
%--Second Output:
%             tt = time vector corresponding to xx
```

Include a copy of your M-file in your lab report. You might want to call the `add_cmplx_exp()` function written in Lab #2 to do the calculation. The function should also generate its own time vector, because that vector can be used to define the horizontal axis when plotting.

CD-ROM

`beatcon.m`

(b) To assist you in your experiments with beat notes a tool called **`beatcon`** has been created. This *user interface controller* is able to call your function `beat.m`, if you check the box Use External beat() in the lower left-hand corner of the GUI. Therefore, before you invoke **`beatcon`** you should be sure your M-file is free of errors. Also, make sure that your `beat.m` function is on the MATLAB path.

Test the M-file written in part (a) via **`beatcon`** by using the values `A=2`, `B=3`, `fc=512`, `delf=8`, `fsamp=8000`, and `dur=1.2` secs. Plot the first 0.2 seconds of the resulting signal. Describe the waveform and explain its properties. Hand in a copy of your plot with measurements of the period of the "envelope" and period of the high frequency signal underneath the envelope. The outline of the waveform is called the *envelope*; derive a mathematical formula for the envelope. When `A` and `B` are different, the beat signal would have to be expressed as the product of an envelope $e(t)$ and a sinusoid with a time-varying phase $\psi(t)$:

$$e(t)\cos(\omega_1 t + \psi(t))$$

Note: the frequency ω_1 is not the center frequency; and $\psi(t)$ causes frequency modulation.

4.2 Beat Note Spectrograms

Beat notes provide an interesting way to investigate the time-frequency characteristics of spectrograms. Although some of the mathematical details are beyond the reach of this course, it is not difficult to appreciate the following issue: there is a fundamental trade-off between knowing which frequencies are present in a signal (or its spectrum) and knowing how those frequencies vary with time. As mentioned previously in Section 3.3, a spectrogram estimates the frequency content over short sections of the signal. If we make the section length very short we can track rapid changes in the frequency. However, shorter sections lack the ability to do accurate frequency measurement because the amount of input data is limited. On the other hand, long sections allow the spectrogram to perform excellent frequency measurements, but it fails to track sudden frequency changes well. For example, if a signal is the sum of two sinusoids whose frequencies are nearly the same, a long section length is needed to "resolve" the two sinusoidal components. This trade-off between the section length (in time) and frequency resolution is equivalent to Heisenburg's Uncertainty Principle in physics. More discussion of the spectrogram will be undertaken in the last chapter of *SP-First*.

A beat note signal may be viewed as a single frequency signal whose amplitude varies with time, *or* as the sum of two signals with different constant frequencies. Both views will be needed to explain the effect of window length when finding the spectrogram of a beat signal.

(a) Create and plot a beat signal defined via: $x(t) = \cos(2\pi(16)t)\cos(2\pi(678)t - \pi/2)$, with a duration of 1.5 secs. Use a sampling rate of $f_s = 8000$ samples/sec to produce the signal in MATLAB. Use `xt` as the name of the MATLAB vector for the signal.

(b) Derive (mathematically) the spectrum of the signal defined in part (a). Make a sketch (by hand) of the spectrum with the correct frequencies and complex amplitudes.

(c) The signal defined in part (a) is periodic; determine its period from the mathematical theory of harmonic spectra.

(d) Plot the spectrogram of $x(t)$ using a window length of 1024 using the commands[3]:

```
plotspec(xt+j*1e-12,fs,1024); grid on
```

Comment on what you see. Are the correct frequencies present in the spectrogram? If necessary, use the zoom tool (in the MATLAB figure window) to examine the important regions of the spectrogram.

(e) Plot the spectrogram of $x(t)$ using a window length of 128 using the commands:

```
plotspec(xt+j*1e-12,fs,128); grid on
```

Comment on the dark and light bands that you see, and determine what property of $x(t)$ is causing them. In addition, compare to the previous spectrogram.

(f) Which spectrogram is correct in the sense that it gives the same result as we would expect for the signal's spectrum derived from the mathematical formula for the signal.

4.3 Spectrogram of an FM Signal

Synthesize an FM signal whose instantaneous frequency is

$$\omega_i(t) = 3000\pi + 800\pi\cos(5\pi t) \quad \text{radians/sec}$$

over a duration of 1.0 secs. Use a sampling rate of 8000 samples/sec.

Listen to the signal. What comments can you make regarding the sound of the FM signal (e.g., is the frequency movement linear)? Does it chirp down, or chirp up, or both?
Create a spectrogram of this chirp signal, and use it to verify that you have the correct instantaneous frequencies. In addition, give the mathematical formula for the chirp.

4.4 A Spectrogram Puzzle

Synthesize a second FM signal (for your lab report) according to the following formula:

$$x(t) = \cos(1600\pi t + 1100\cos(4\pi t)) \qquad \text{for } 0 \le t \le 1.2$$

and use a sampling rate of 8000 samples/sec.

Listen to the FM signal. Does it chirp down, or chirp up, or both? Give the mathematical formula for the instantaneous frequency of the FM signal. In addition, create a spectrogram of this second FM signal. Use the theory of the spectrum (with its positive and negative frequency components) to help explain the connection between what you hear and what you see in the spectrogram. Observe the changing instantaneous frequency in the spectrogram which implies that the frequency components in the spectrum are moving.
Hint: In order to create a **spectrogram with negative frequencies**, see the instructions in Section 3.3.

[3] Use `plotspec` instead of `specgram` in order to get a linear amplitude scale rather than logarithmic.

Lab #3
ECE-2025
Spring-2005
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your TA before the end of your lab period.

Name: _____ Date of Lab: _____

Part 3.1 Demonstrate usage of the Beat Control GUI.

Verified:_____ Date/Time:_____

Part 3.2 Demonstrate the `syn_chirp.m` function. In the space below write how you would call the function with a correct set of arguments.

Verified:_____ Date/Time:_____

Part 3.3 Demonstrate the ***two-sided*** spectrogram of a sinusoid and a chirp, i.e., include the negative frequencies by using the MATLAB function `plotspec`.

Verified:_____ Date/Time:_____