Lab P-4: AM and FM Sinusoidal Signals

Pre-Lab and Warm-Up: You should read at least the Pre-Lab and Warm-up sections of this lab assignment and go over all exercises in the Pre-Lab section before going to your assigned lab session.

Verification: 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 demonstrate the step to the instructor. Turn in the completed verification sheet to your instructor when you leave the lab.

Lab Report: Write a lab report on Section 4 with graphs and explanations. Please **label** the axes of your plots and include a title for every plot. In order to keep track of plots, include each plot *inlined* within your report. If you are unsure about what is expected, ask the instructor who will grade your report.

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 companion website that provide examples of these signals for many different conditions, e.g., the demo of *FM Synthesis* in Chapter 3.

2 Pre-Lab

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

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

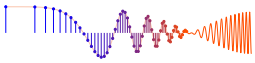
In this lab, we will extend our treatment of sinusoidal waveforms to more complicated signals composed of sums of sinusoidal signals, or sinusoids with changing frequency.

2.1 Amplitude Modulation

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

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

where $A_k e^{j\varphi_k}$ is the complex amplitudeof 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 very close together. See Chapter 3 for a more detailed discussion of beat signals.

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 also the exponent of the complex exponential, so the angle of this signal is the exponent $(2\pi f_0 t + \varphi)$. This angle function changes *linearly* versus time, and its time derivative is $2\pi f_0$ which equals the constant frequency of the cosine in rad/s.

A generalization is available if we adopt the following notation for the class of signals represented by a cosine function with a time-varying angle:

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

The time derivative of the angle from (3) gives a frequency in rad/s

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

but we prefer units of hertz, so 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 *chirp* signal is a sinusoid whose frequency changes linearly from a starting value to an ending one.[1] The formula for such a signal can be defined by creating a complex exponential signal with quadratic angle by defining $\psi(t)$ in (3) as

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

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

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

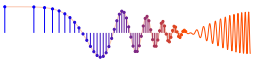
The slope of $f_i(t)$ is equal to 2μ and its intercept is equal to f_0. If the signal starts at time $t = 0$ secs., then f_0 is also the starting frequency. The frequency variation produced by such a time-varying angle is called *frequency modulation.* This kind of signal is an example of an frequency modulated (FM) signal. More generally, we often consider them to be part of a larger class called *angle modulation* 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 = 11025;
dt = 1/fsamp;
dur = 1.8;
tt = 0 : dt : dur;
psi = 2*pi*(100 + 200*tt + 500*tt.*tt);
xx = real( 7.7*exp(j*psi) );
soundsc( xx, fsamp );
```

[1] There is a demo on the companion website called *Spectrograms & Sounds: Wideband FM*.

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

(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 $t_n = nT_s$ represents discrete time instants. In the MATLAB code above, what is the value for t_n? What are 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. What are the minimum and maximum frequencies that will be heard?

(d) Listen to the signal to determine whether the signal's frequency content is increasing or decreasing (use `soundsc()`). Notice that `soundsc()` needs to know the sampling rate at which the signal samples were created. For more information do `help sound` and `help sound()`.

3 Warm-up

The instructor verification sheet may be found at the end of this lab.

3.1 Beat Control GUI

To assist you in your experiments with beat notes and AM signals, the tool called `beatcon` has been created.[2] 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 at 850 Hz and 870 Hz. Demonstrate the plot and sound to your instructor.

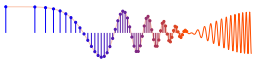
Instructor Verification (separate page)

3.2 Function for a Chirp

Use the code provided in the warm-up as a starting point in order to write a MATLAB function that will synthesize a "chirp" signal according to the following comments:

```
function   [xx,tt] = mychirp( f1, f2, dur, fsamp )
%MYCHIRP      generate a linear-FM chirp signal
%
%  usage:   xx = mychirp( f1, f2, dur, fsamp )
%
%      f1 = starting frequency
%      f2 = ending frequency
%     dur = total time duration
%   fsamp = sampling frequency  (OPTIONAL: default is 11025)
%
```

[2] The MATLAB M-file *beatcon.m* is part of the *DSP First* MATLAB *Toolbox*.

```
%      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 = 11025;
end
```

As a test case, generate a chirp sound whose frequency starts at 2500 Hz and ends at 500 Hz; its duration should be 1.5 s. Listen to the chirp using the `soundsc` function. Include a listing of the `mychirp.m` function that you wrote.

Instructor Verification (separate page)

3.3 Advanced Topic: Spectrograms

It is often useful to think of signals in terms of their spectra. 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 components, one at $2\pi f_0$, the other at $-2\pi f_0$. For more complicated signals, the spectrum may be very interesting and, in the case of FM, 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.[3]

When unsure about a command, use `help`.

There are a few important things to know about spectrograms:

1. In MATLAB the function spectrogram will compute the spectrogram, as already explained in a recent lecture. Type `help spectrogram` to learn more about this function and its arguments.

2. Spectrograms are numerically calculated and only provide an estimate of the time-varying frequency content of a signal. There are theoretical limits on how well spectrograms can actually represent the frequency content of a signal. A later lab will treat this problem when we use the spectrogram to extract the frequencies of piano notes.

3. A common call to the function is `spectrogram(xx,1024,[],1024,fs,'yaxis')`. The second argument[4] is the *window length* which is varied to get different looking spectrograms. The spectrogram is able to "distinguish" the separate spectrum lines with a longer window length, e.g., 1024 or 2048.[5]

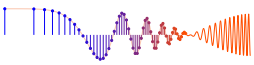
4. An alternative to `spectrogram()` is the function `plotspec(xx,fs,<windowLength>)` which is loaded if you have installed the DSP First toolbox.

   • Notes: 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 display; instead, darker shades of gray indicate larger values with black being the largest. Finally, the amplitude of the display in linear for `plotspec`, but logarithmic for `spectrogram`.

[3] In Chapter 8, there is a Spectrogram GUI called `specgramdemo`; it is installed as part of the *DSP First* MATLAB *Toolbox*.

[4] The second and fourth arguments specify the window length and FFT length which should be equal for this lab; the third argument is set equal to the "empty matrix" so that the default value for overlap will be used.

[5] 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.

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

```
fs=8000; xx = cos(3000*pi*(0:1/fs:0.5)); specgram(xx,1024,fs); colorbar
```

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

```
fs=8000; xx = cos(3000*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.

4 Lab Exercise: Chirps and Beats

For the lab exercise and lab report, you will synthesize some AM and FM signals. In order to verify that they 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.,

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

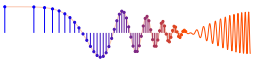
In this part, we will compute samples of such a signal and listen to the result.

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

```
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 `syn_sin()` 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.

(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=10`, `B=10`, `fc=1000`, `delf=10`, `fsamp=11025`, and `dur=1` s. 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.

(c) (Optional)[6] Experiment with different values of the frequency difference f_Δ. Listen to the sounds (there is a *button* on `beatcon` that will do this for you automatically) and think about the relationship between the sound and waveform.

4.2 More on 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 can give excellent frequency measurements, but fail to track 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.

When $A = B$ in (5), the beat signal can be expressed as

$$x(t) = A\cos(2\pi(f_c - f_\Delta)t) + A\cos(2\pi(f_c + f_\Delta)t) = A[\cos(2\pi f_\Delta t)]\cos(2\pi f_c t)$$

Therefore, a beat note signal may be viewed as two signals with different constant frequencies, *or* as a single frequency signal whose amplitude varies with time. Both views will be useful in evaluating the effect of window length when finding the spectrogram of a beat signal.

(a) Create and plot a beat signal with

(i) $f_\Delta = 32$ Hz

(ii) $T_{\text{dur}} = 0.26$ s

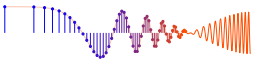
(iii) $f_s = 11025$ Hz

(iv) $f_c = 2000$ Hz

(b) Find the spectrogram using a window length of 2048 using the commands:
`spectrogram(x,2048,[],2048,fsamp); colormap(1-gray(256)).`
Comment on what you see. Are the correct frequencies present in the spectrogram? If necessary, use the zoom tool to examine the important region of the spectrogram.

(c) Find the spectrogram using a window length of 16 using the commands:
`spectrogram(x,16,[],16,fsamp); colormap(1-gray(256)).`
Comment on what you see, and compare to the previous spectrogram.

[6] Optional means that you do not have to include this in your lab report.

4.3 Spectrogram of a Chirp

Use the `mychirp` function (written during the Warm-up) to synthesize a chirp signal for your lab report. Use the following parameters:

1. A total time duration of 3 s, with a D/A conversion rate of $f_s = 11025$ Hz.
2. The instantaneous frequency starts at 5,000 Hz and ends at 300 Hz.

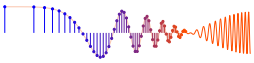
Listen to the signal. What comments can you make regarding the sound of the chirp (e.g., is the frequency movement linear)? Does it chirp down, or chirp up?
Create a spectrogram of this chirp signal, and use it to verify that you have the correct instantaneous frequencies.

4.4 A Chirp Puzzle

Synthesize a second "chirp" signal (for your lab report) with the following parameters:

1. A total time duration of 3 s, with a sampling rate of $f_s = 11025$ Hz.
2. The instantaneous frequency starts at 3,000 Hz and ends at $-2,000$ Hz (negative frequency).

Listen to the signal. Does it chirp down, or chirp up, or both?
Create a spectrogram of this second chirp signal.
Use the theory of the spectrum (with its positive and negative frequency components) to help explain what you hear and what you see in the spectrogram. In other words, the changing instantaneous frequency implies that the frequency components in the spectrum are moving.



Lab: AM and FM Sinusoidal Signals
INSTRUCTOR VERIFICATION SHEET

For each verification, be prepared to explain your answer and respond to other related questions that your instructor might ask. Turn this page in at 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 `mychirp.m` function. In the space below write how you would call the function with a correct set of arguments.

Verified:_____ Date/Time:_____