DSP First

Laboratory Exercise #4

AM and FM Sinusoidal Signals

The objective of this lab is to introduce more complicated signals that are related to the basic sinusoid. These are 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.

FM Synthesis

1 Overview

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

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

In this lab, we will continue to investigate sinusoidal waveforms, but for more complicated signals composed of sums of sinusoidal signals, or sinusoids with changing frequency.

1.1 Amplitude Modulation

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

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

where $Z_k = A_k e^{j\phi_k}$ is the complex exponential amplitude. The choice of f_k will determine the nature of the signal—for amplitude modulation we pick two or three frequencies very close together, see Chapter 3.

1.2 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 phase of this signal is the exponent $(2\pi f_0 t + \phi)$. This phase 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 phase:

FM Synthesis

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

The time derivative of the phase from (3) gives a frequency

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

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

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

1.3 Chirp, or Linearly Swept Frequency

Spectrograms & Sounds: Wide-band FM

A *chirp* signal is a sinusoid whose frequency changes linearly from some low value to a high one. The formula for such a signal can be defined by creating a complex exponential signal with quadratic phase by defining $\psi(t)$ in (3) as

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

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 $t = 0$, then f_0 is also the starting frequency. The frequency variation produced by the time-varying phase 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."

1.4 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 spikes, one at $2\pi f_0$, the other at $-2\pi f_0$. For more complicated signals the spectra 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.

Sounds & Spectrograms

There are a few important things to know about spectrograms:

1. In MATLAB the function `specgram` will compute the spectrogram, as already explained in Lab 3. Type `help specgram` 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 they can actually represent the frequency content of a signal. Lab 11 will treat this problem when we use the spectrogram to extract the frequencies of piano notes.

2 Warm-up

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

2.1 MATLAB Synthesis of Chirp Signals

(a) The following MATLAB code will synthesize a chirp:

```
fsamp = 8000;
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) );
sound( xx, fsamp );
```

Determine the range of frequencies (in hertz) that will be synthesized by this MATLAB script. Make a sketch by hand of the instantaneous frequency versus time. What are the minimum and maximum frequencies that will be heard? Listen to the signal to verify that it has the expected frequency content.

Instructor Verification (separate page)

(b) Use the code provided in part (a) to help you write a MATLAB function that will synthesize a "chirp" signal according to the following comments:

```
function   xx = 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 8000)
%
if( nargin < 4 )   %-- Allow optional input argument
   fsamp = 8000;
end
```

When unsure about a command, use `help`.

Generate a chirp sound to match the frequency range of the chirp in part (a). Listen to the chirp using the `sound` function. Also, compute the spectrogram of your chirp using the MATLAB function: `specgram(xx,[],fsamp)`.

Instructor Verification (separate page)

3 Lab A: Chirps and Beats

3.1 Synthesize a Chirp

Use your MATLAB function `mychirp` to synthesize a "chirp" signal for your lab report. Use the following parameters:

1. A total time duration of 3 secs. with a D/A conversion rate of $f_s = 8000$ Hz.
2. The instantaneous frequency starts at 15,000 Hz and ends at 300 Hz.

Listen to the signal. What comments can you make regarding the sound of the chirp (e.g. is it linear)? Does it chirp down, or chirp up, or both? Create a spectrogram of your chirp signal. Use the sampling theorem (from Chapter 4 in the text) to help explain what you hear and see.

3.2 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, f, 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
%--OPTIONAL Output:
%             tt = time vector corresponding to xx
```

Hand in a copy of your M-file. You might want to call the `sumcos` written in Lab 2 to do the calculation. The function could also generate its own time vector. You may elect to not implement the second output vector `tt`, but it is quite convenient for plotting. To assist you in your experiments with beat notes a new tool called `beatcon` has been created. This *user interface controller* actually calls your function `beat.m`. Therefore, before you invoke `beatcon` you should be sure your M-file is free of errors. Once you have the function `beat.m` working properly invoke the demo/tool by typing `beatcon` at the MATLAB prompt A small control panel will appear on the screen with *buttons* and *sliders* that vary the different parameters for these exercises. Experiment with the `beatcon` control panel and use it to complete the remainder of exercises in this section.

(b) Test the M-file written in part (a) via `beatcon` by using the values `A=10`, `B=10`, `fc=1000`, `delf=10`, `fsamp=8000`, and `dur=1` 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.

(c) For this part, set `delf` to 10 Hz. Send the resulting signal to the D-to-A converter and listen to the sound (there is a *button* on `beatcon` that will do this for you automatically). Explain the nature of the sound based on the waveform plotted in part (b) and on the theory developed in Chapter 3.

(d) Experiment with different values of the frequency difference f_Δ.

3.3 More on Spectrograms (Optional)

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 understand 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 1.4, a spectrogram estimates the frequency content over short sections of the signal. Long sections give excellent frequency resolution, but fail to track frequency changes well. Shorter sections have poor frequency resolution, but good tracking. 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 can be found in Chapter 9 and Lab 11.

A beat note signal may be viewed as a single frequency signal whose amplitude varies with time, *or* as two signals with different constant frequencies. 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) $T_{\text{dur}} = 0.26$ sec

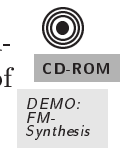
(iii) $f_s = 8000$ Hz, or 11,025 Hz

(iv) $f_0 = 2000$ Hz

(b) Find the spectrogram using a window length of 2048 using the commands:
`specgram(x,2048,fsamp); colormap(1-gray(256))`.
Comment on what you see.

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

4 Lab B: FM Synthesis of Instrument Sounds



Frequency modulation (FM) can be used to make interesting sounds that mimic musical instruments, such as bells, woodwinds, drums, etc. The goal in this lab is to implement one or two of these FM schemes and hear the results.

We have already seen that FM defines the signal $x(t)$ to have a time-varying phase

$$x(t) = A\cos(\psi(t))$$

and that the instantaneous frequency (4) changes according to the oscillations of $\psi(t)$. If $\psi(t)$ is linear, $x(t)$ is a constant-frequency sinusoid; whereas, if $\psi(t)$ is quadratic, $x(t)$ is a chirp signal whose frequency changes linearly in time. FM music synthesis uses a more interesting $\psi(t)$, one that is sinusoidal. Since the derivative of a sinusoidal $\psi(t)$ is also sinusoidal, the instantaneous frequency of $x(t)$ will oscillate. This is useful for synthesizing instrument sounds because the proper choice of the modulating frequencies will produce a fundamental frequency and several overtones, as many instruments do.

The general equation for an FM sound synthesizer is:

$$x(t) = A(t)\cos\left(2\pi f_c t + I(t)\cos(2\pi f_m t + \phi_m) + \phi_c\right) \tag{6}$$

where $A(t)$ is the signal's amplitude. It is a function of time so that the instrument sound can be made to fade out slowly or cut off quickly. Such a function is called an *envelope.* The parameter f_c is called the *carrier* frequency. Note that when you take the derivative of $\psi(t)$ to find $f_i(t)$,

$$\begin{aligned} f_i(t) &= \frac{1}{2\pi}\frac{d}{dt}\psi(t) \\ &= \frac{1}{2\pi}\frac{d}{dt}\left(2\pi f_c t + I(t)\cos(2\pi f_m t + \phi_m) + \phi_c\right) \\ &= f_c - I(t) f_m \sin(f_m t + \phi_m) + \frac{dI}{dt}\cos(2\pi f_m t + \phi_m) \end{aligned} \tag{7}$$

f_c will be a constant in that expression. It is the frequency that would be produced without any frequency modulation. The parameter f_m is called the *modulating* frequency. It expresses the rate of oscillation of $f_i(t)$. The parameters ϕ_m and ϕ_c are arbitrary phase constants, usually both set to $-\pi/2$ so that $x(0) = 0$.

The function $I(t)$ has a less obvious purpose than the other FM parameters in (6). It is technically called the *modulation index envelope.* To see what it does, examine the expression for the instantaneous frequency (7). The quantity $I(t)f_m$ multiplies a sinusoidal variation of the frequency. If $I(t)$ is constant or $\dfrac{dI}{dt}$ is relatively small, then $I(t)f_m$ gives the maximum amount by which the instantaneous frequency deviates from f_c. Beyond that, however, it is difficult to relate $I(t)$ to the sound made by $x(t)$ without some rather tedious mathematical analysis.

In our study of signals, we would like to characterize $x(t)$ as the sum of several constant-frequency sinusoids instead of a single signal whose frequency changes. In this regard, the following comments are relevant: when $I(t)$ is small (e.g., $I \approx 1$), low multiples of the carrier frequency (f_c) have high amplitudes. When $I(t)$ is large ($I > 4$), both low and high multiples of the carrier frequency have high amplitudes. The net result is that $I(t)$ can be used to vary the harmonic content of the instrument sound (called overtones). When $I(t)$ is small, mainly low frequencies will be produced. When $I(t)$ is large, higher harmonic frequencies can also be produced. Since $I(t)$ is a function of time, the harmonic content will change with time. For more details see the paper by Chowning.[1]

4.1 Generating the Bell Envelopes

Now we take the general FM synthesis formula (6) and specialize for the case of a bell sound. The amplitude envelope $A(t)$ and the modulation index envelope $I(t)$ for the bell are both decaying exponentials. That is, they both have the following form:

$$y(t) = e^{-t/\tau} \tag{8}$$

where τ is a parameter that controls the decay rate of the exponential. Notice that $y(0) = 1$ and $y(\tau) = 1/e$, so τ is the time it takes a signal of the form (8) to decay to $1/e = 36.8\%$ of its initial value. For this reason, the parameter τ is called the *time constant.*

Use (8) to write a MATLAB function that will generate a decaying exponential to be used later in synthesizing a bell sound. The file header should look like this:

[1] Ref: John M. Chowning, "The Synthesis of Complex Audio Spectra by means of Frequency Modulation," *Journal of the Audio Engineering Society*, vol. 21, no. 7, Sept. 1973, pp. 526–534.

```
function  yy = bellenv(tau, dur, fsamp);
%BELLENV produces envelope function for bell sounds
%
%        usage: yy = bellenv(tau, dur, fsamp);
%
%               where tau = time constant
%                     dur = duration of the envelope
%                   fsamp = sampling frequency
%                 returns:
%                      yy = decaying exponential envelope
%
%   note: produces exponential decay for positive tau
```

The function will be one or two lines of MATLAB code. The first line should define your time vector based on `fsamp` and `dur`, and the second generates the exponential (8).

The bell's amplitude envelope, $A(t)$, and modulation index envelope, $I(t)$ are identical, up to a scale factor.

$$A(t) = A_0 e^{-t/\tau} \qquad \text{and} \qquad I(t) = I_0 e^{-t/\tau}$$

Hence, one call to the `bellenv` function will generate the shape for both envelopes.

4.2 Parameters for the Bell

Now that we have the bell's amplitude and modulation index envelopes, we can create the actual sound signal for the bell by specifying all the parameters in the general FM synthesis formula (6). The frequencies f_c and f_m must be given numerical values. The ratio of carrier to modulating frequency is important in creating the sound of a specific instrument. For the bell, a good choice for this ratio is 1:2, e.g., $f_c = 110$ Hz and $f_m = 220$ Hz.

Now write a simple M-file `bell.m` that implements (6) to synthesize a bell sound. Your function should call `bellenv.m` to generate $A(t) = A_0 e^{-t/\tau}$ and $I(t) = I_0 e^{-t/\tau}$.

```
function  xx = bell(ff, Io, tau, dur, fsamp)
%BELL     produce a bell sound
%
%     usage:   xx = bell(ff, Io, tau, dur, fsamp)
%
%     where:  ff = frequency vector (containing fc and fm)
%              Io = scale factor for modulation index
%            tau = decay parameter for A(t) and I(t)
%            dur = duration (in sec.) of the output signal
%          fsamp = sampling rate
```

4.3 The Bell Sound

Test your `bell( )` function using the parameters of case #1 in the table. Play it with the `sound()` function at 11,025 Hz.[2] Does it sound like a bell? The value of $I_0 = 10$ for scaling the modulation

[2] A higher sampling rate of 11,025 Hz is used because the signal contains many harmonics, some of which might alias if a lower f_s were used. You should experiment with lower values of f_s to see if you can hear a difference, e.g.,

index envelope is known to give a distinctive sound. Later on, you can experiment with other values to get a variety of bells.

CASE	f_c (Hz)	f_m (Hz)	I_0	τ (sec)	T_{dur} (sec)	f_s (Hz)
1	110	220	10	2	6	11,025
2	220	440	5	2	6	11,025
3	110	220	10	12	3	11,025
4	110	220	10	0.3	3	11,025
5	250	350	5	2	5	11,025
6	250	350	3	1	5	11,025

The frequency spectrum of the bell sound is very complicated, but it does consist of spectral lines, which can be seen with a spectrogram. Among these frequencies, one spectral line will dominate what we hear. We would call this the *note frequency* of the bell. It is tempting to guess that the note frequency will be equal to f_c, but you will have to experiment to find the true answer. It might be f_m, or it might be something else—perhaps the fundamental frequency which is the greatest common divisor of f_c and f_m.

For each case in the table, do the following:

(a) Listen to the sound by playing it with the `sound()` function.

(b) Calculate the fundamental frequency of the "note" being played. Explain how you can verify by listening that you have the correct fundamental frequency.

(c) Describe how you can hear the frequency content changing according to $I(t)$. Plot $f_i(t)$ versus t for comparison.

(d) Display a spectrogram of the signal. Describe how the frequency content changes, and how that change is related to $I(t)$. Point out the "harmonic" structure of the spectrogram, and calculate the fundamental frequency, f_0.

(e) Plot the entire signal and compare it to the envelope $A(t)$ generated by `bellenv`.

(f) Plot about 100–200 samples from the middle of the signal and explain what you see, especially the frequency variation.

If you are making a lab report, do the plots for two cases—choose one of the first four and one of the last two. Write up an explanation only for the two that you choose.

4.4 Comments about the Bell

Cases #3 and #4 are extremes for choosing the decay rate τ. In case #3, the waveform does not decay very much over the course of three seconds and sounds a little like a sum of harmonically related sinusoids. With a "faster" decay rate, as in case #4, we get a percussion-like sound. Modifying the fundamental frequency f_0 (determined in part (d) above) should have a noticeable effect on the tone you hear. Try some different values for f_0 by changing f_c and f_m, but still in the ratio of 1:2. Describe what you hear.

Finally, experiment with different carrier to modulation frequency ratios. For example, in his paper, Chowning uses a fundamental frequency of $f_0 = 40$ Hz and a carrier to modulation frequency ratio of 5:7. Try this and a few other values. Which parameters sound "best" to you?

$f_s = 8000$ Hz.

5 Woodwinds

As an alternative to the bell sounds, this section shows how different parameters in the same FM synthesis formula (6) will yield a clarinet sound, or other woodwinds.

5.1 Generating the Envelopes for Woodwinds

There is a function on the CD-ROM called `woodwenv` which produces the functions needed to create both the $A(t)$ and $I(t)$ envelopes for a clarinet sound. The file header looks like this:

```
function  [y1, y2] = woodwenv(att, sus, rel, fsamp)
%WOODWENV     produce normalized amplitude and modulation index
%               functions for woodwinds
%
%    usage: [y1, y2] = woodwenv(att, sus, rel, fsamp);
%
% where  att = attack TIME
%        sus = sustain TIME
%        rel = release TIME
%      fsamp = sampling frequency (Hz)
% returns:
%      y1 = (NORMALIZED) amplitude envelope
%      y2 = (NORMALIZED) modulation index envelope
%
%  NOTE: attack is exponential, sustain is constant,
%          release is exponential
```

The outputs from `woodwenv` are normalized so that the minimum value is zero and the max is one. Try the following statements to see what the function produces:

```
fsamp = 8000;
Ts = 1/fsamp;
tt = delta : Ts : 0.5;
[y1, y2] = woodwenv(0.1, 0.35, 0.05, fsamp);
subplot(2,1,1), plot(tt,y1), grid on
subplot(2,1,2), plot(tt,y2), grid on
```

5.2 Scaling the Clarinet Envelopes

Since the woodwind envelopes produced by `woodwenv` range from 0 to 1, some scaling is necessary to make them useful in the FM synthesis equation (6). In this section, we consider the general process of linear re-scaling. If we start with a *normalized* signal $y_{\text{norm}}(t)$ and want to produce a new signal whose max is $y_{\max}$ and whose min is $y_{\min}$, then we must map 1 to $y_{\max}$ and 0 to $y_{\min}$. Consider the linear mapping:

$$y_{\text{new}}(t) = \alpha\, y_{\text{norm}}(t) + \beta \tag{9}$$

Determine the relationship between α and β and $y_{\max}$ and $y_{\min}$, so that the max and the min of $y_{\text{new}}(t)$ are correct.
Test this idea in MATLAB by doing the following example (where $\alpha = 5$ and $\beta = 3$):

```
ynorm = 0.5 + 0.5*sin( pi*[0:0.01:1]);
subplot(2,1,1),  plot(ynorm)
alpha = 5;    beta = 3;
ynew = alpha*ynorm + beta;          %<------ Linear re-scaling
subplot(2,1,1),  plot(ynew)
max(ynorm),  min(ynorm)      %<--- ECHO the values
max(ynew),  min(ynew)
```

What happens if we make α negative?
Write a short one-line function that implements (9) above. Your function should have the following form: `function y = scale(data, alpha, beta).`

5.3 Clarinet Envelopes

For the clarinet sound, the amplitude $A(t)$ needs no scaling—the MATLAB function `sound` will automatically scale to the maximum range of the D/A converter. Thus, $A(t)$ equals the vector `y1`. From the plot of `y1` shown in Fig. 1, it should be obvious that this envelope will cause the sound to rise quickly to a certain volume, sustain that volume, and then quickly turn off.

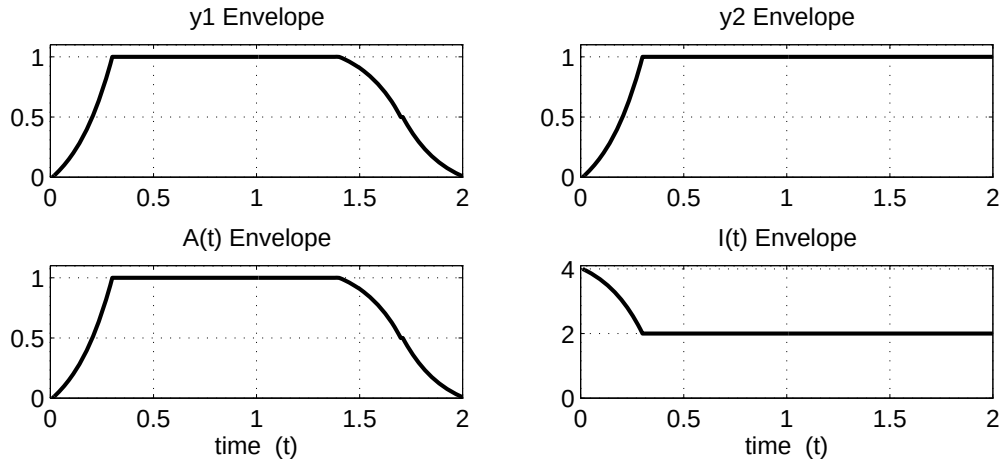


Figure 1: Envelopes for the woodwinds. The functions $A(t)$ and $I(t)$ are produced by scaling `y1` and `y2`, the outputs of `woodwenv`.

The modulation index envelope, $I(t)$, however, does not equal `y2`. The range for $I(t)$ lies between 2 and 4 as in Fig. 1. Furthermore, there is an inversion so that when `y2` is zero, $I(t)$ should equal 4, and when `y2` is one, $I(t)$ should be 2. Using this information solve for the appropriate α and β then use `scale` to produce the modulation index envelope function (`I`) for a clarinet sound.

5.4 Parameters for the Clarinet

So far we have a general equation for FM signals, an amplitude envelope for the clarinet, and a modulation index envelope for the clarinet. To create the actual sound signal for the clarinet, we need to specify the additional parameters in (6). The ratio of carrier to modulating frequency is important in creating the sound of a specific instrument. For the clarinet, this ratio should be 2:3. The actual note frequency will be the greatest common divisor of the carrier and modulating

frequencies. For example, when we choose $f_c = 600$ Hz and $f_m = 900$ Hz, the synthesized signal will have a fundamental frequency of $f_0 = 300$ Hz.

Write a simple M-file `clarinet.m` that implements the FM synthesis equation (6) to synthesize a clarinet note. Your function should generate the envelopes $A(t)$ and $I(t)$ using textttscale and textttwoodwenv. The function header should look like this:

```
function  yy = clarinet(f0, Aenv, Ienv, dur, fsamp)
%CLARINET     produce a clarinet note signal
%
%     usage:  yy = clarinet(f0, Aenv, Ienv, dur, fsamp)
%
%     where:   f0 = note frequency
%             Aenv = the array holding the A(t) envelope
%             Ienv = the array holding the I(t) envelope
%              dur = the amount of time the signal lasts
%            fsamp = the sampling rate
```

5.5 Experiment with the Clarinet Sound

Using your `clarinet( )` function, create a 250 Hz clarinet note with $f_s = 8000$ or 11,025 Hz. Play it with the `sound( xnote, fs )` function. Does it sound like a clarinet? How can you verify that its fundamental frequency is at 250 Hz?

Explain how the modulation index $I(t)$ will affect the frequency content versus time of the clarinet sound. Describe how you can hear the frequency content changing according to $I(t)$? Plot the instantaneous frequency $f_i(t)$ versus t for comparison.

Plot the entire signal and compare it to the amplitude envelope function `y1` in Fig. 1. Plot about 100–200 samples from the middle of the signal and explain what you see.

Finally, synthesize other note frequencies. For example, make the C-major scale (defined in Lab 3) consisting of seven consecutive notes.

Lab 4
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: _____ Date: _____

Part 2.1 Explain chirp signal:

Verified: _____

Part 2.1 Complete the `mychirp.m` function:

Verified: _____