**MAS160: Signals, Systems & Information for Media Technology**

Problem Set 2

*Instructors: V. Michael Bove, Jr.*

## Problem 1: Phase and Time shifting

(a) *DSP First* **2.17(a)**

**SOLUTION :**

$$\begin{aligned}
x(t) &= 5\cos(\omega_0 t + 3\pi/2) + 4\cos(\omega_0 t + 2\pi/3) + 4\cos(\omega_0 t + \pi/3) \\
&= \Re e\left\{\frac{5}{2}\left(e^{j\omega_0 t}e^{j3\pi/2} + e^{-j\omega_0 t}e^{-j3\pi/2}\right)\right. \\
&\quad +2\left(e^{j\omega_0 t}e^{j2\pi/3} + e^{-j\omega_0 t}e^{-j2\pi/3}\right) \\
&\quad \left.+2\left(e^{j\omega_0 t}e^{j\pi/3} + e^{-j\omega_0 t}e^{-j\pi/3}\right)\right\} \qquad (1) \\
&= \Re e\left\{e^{j\omega_0 t}\left(\frac{5}{2}e^{j3\pi/2} + 2e^{j2\pi/3} + 2e^{j\pi/3}\right) + e^{-j\omega_0 t}\left(\frac{5}{2}e^{-j3\pi/2} + 2e^{-j2\pi/3} + 2e^{-j\pi/3}\right)\right\} \\
&= \Re e\left\{e^{j\omega_0 t}\left(\frac{4\sqrt{3}-5}{2}\right)e^{j\pi/2} + e^{-j\omega_0 t}\left(\frac{4\sqrt{3}-5}{2}\right)e^{-j\pi/2}\right\} \qquad (2) \\
&= \left(4\sqrt{3}-5\right)\cos(\omega_0 t + \pi/2)
\end{aligned}$$

## Problem 2: Frequency-domain & Time-domain Conversions

(a) *DSP First* **3.2**
(b) *DSP First* **3.3(a),3.3(b)**

**SOLUTION :**

*(a)* *(a) see figure 2*
*(b) Yes, the signal is periodic. The period is the least common multiple of the periods of the components. The periods of those components are 400 Hz, 600 Hz and 800 Hz. The least common multiple is 200 Hz.*
*(c) The spectrum is changed. Now there are additional spikes of height 5/2 $e^{j\pi/2}$ at ± 500 Hz. The overall signal is still periodic, but now the period is reduced to 100 Hz.*

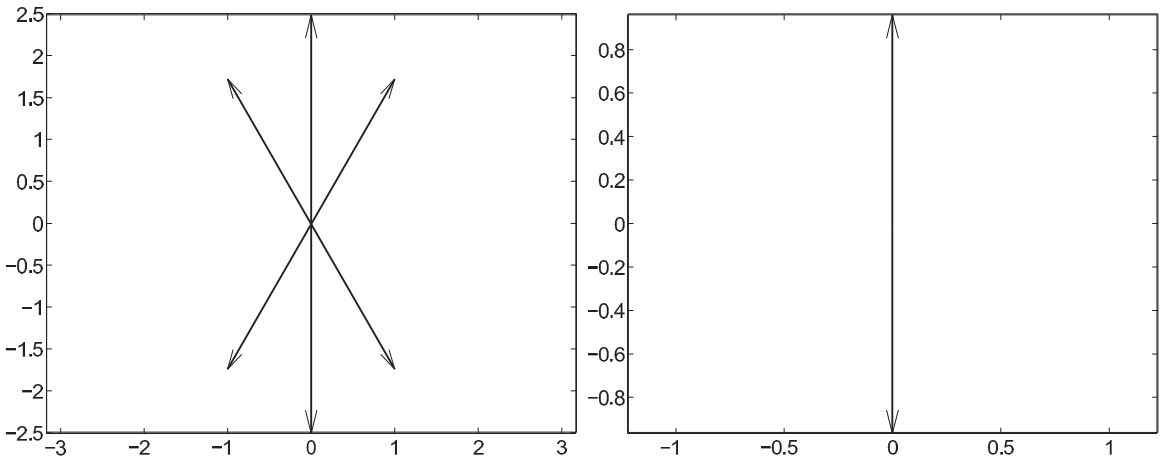

Figure 1: phasors

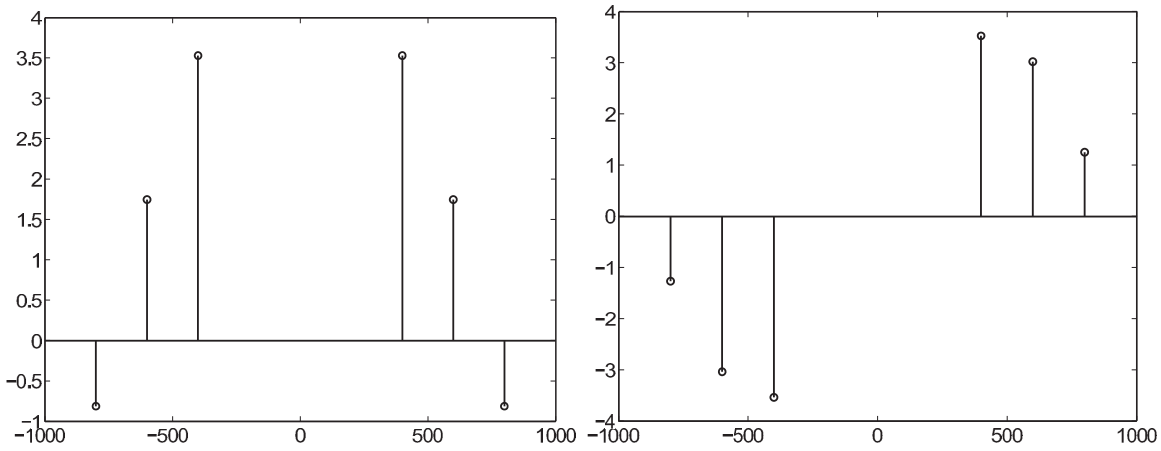

Figure 2: The Real and Imaginary Parts of the Spectrum

*(b)* *(a) From the spectrum, there are two periodic components and one DC component:*

$$\textit{at 0 Hz, amp. } 11 \rightarrow 11$$

$$\textit{at} \pm \textit{ 50 Hz, amp. } 7e^{\pm j\pi/3} \rightarrow 7 \cdot 2\cos\left(50(2\pi)t - \frac{\pi}{3}\right)$$

$$\textit{at} \pm \textit{ 175 Hz, amp. } 4e^{\pm j\pi/2} \rightarrow 4 \cdot 2\cos\left(175(2\pi)t - \frac{\pi}{2}\right)$$

*so,*

$$x(t) = 11 + 14\cos\left(100\pi t - \frac{\pi}{3}\right) + 8\cos\left(350\pi t - \frac{\pi}{2}\right)$$

*(b) A sum of harmonic functions is also periodic. The period will be the least common multiple of the two periods:*

$$T_1 = 1/50 \textit{ sec} \qquad\qquad T_2 = 1/175 \textit{ sec}$$

*The least common multiple is 2/50 = 7/175 = 1/25, so the overall period is* $T = 1/25$ *sec.*

3. Fourier Series
Determine the Fourier series for the following periodic signals of period $T_0$:

(a) $x(t) = |t|, \quad -T_0/2 \le t < T_0/2$

(b) $x(t) = \begin{cases} t, & 0 \le t < T_0/2 \\ t^2, & T_0/2 \le t < T_0 \end{cases}$

**Solutions:**

(a) $x(t) = |t|, \quad -T_0/2 \le t < T_0/2$

$$X_0 = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} x(t)\,dt = \frac{1}{T_0}\int_{-T_0/2}^{0} -t\,dt + \frac{1}{T_0}\int_{0}^{T_0/2} t\,dt = \frac{1}{T_0}\frac{T_0^2}{4} = \frac{T_0}{4}$$

$$X_k = \frac{2}{T_0}\int_{-T_0/2}^{T_0/2} e^{-j2\pi kt/T_0}x(t)\,dt = \frac{2}{T_0}\int_{-T_0/2}^{0} -te^{-j2\pi kt/T_0}\,dt + \frac{2}{T_0}\int_{0}^{T_0/2} te^{-j2\pi kt/T_0}\,dt$$

$$= -\frac{T_0}{(\pi k)^2}(1-(-1)^k)$$

(b) $x(t) = \begin{cases} t, & 0 \le t < T_0/2 \\ t^2, & T_0/2 \le t < T_0 \end{cases}$

$$X_0 = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} x(t)\,dt = \frac{1}{T_0}\int_{0}^{T_0/2} t\,dt + \frac{1}{T_0}\int_{T_0/2}^{T_0} t^2\,dt = \frac{1}{T_0}\left(\frac{T_0^2}{8} + \frac{7T_0^3}{24}\right) = \frac{T_0}{8} + \frac{7T_0^2}{24}$$

$$X_k = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} e^{-j2\pi kt/T_0}x(t)\,dt = \frac{1}{T_0}\int_{0}^{T_0/2} te^{-j2\pi kt/T_0}x(t)\,dt + \frac{1}{T_0}\int_{T_0/2}^{T_0} t^2e^{-j2\pi kt/T_0}x(t)\,dt$$

$$= \frac{T_0}{4\pi^2k^2}[2T_0 - T_0(-1)^k - 1 + (-1)^k] + j\frac{T_0}{4\pi k}\left[\frac{T_0}{(\pi^2k^2)}(1-(-1)^k) - 2T_0 + (-1)^kT_0/2 + (-1)^k\right]$$

Hopefully you found Mathematica (or another similar software) helpful in performing this integration!

For the following lab exercises (found in Appendix C of the *DSP First* text), please turn in a hard copy of your functions.

4. *DSP First* **Lab 3**
You only need to synthesize one of the 5 musical pieces given (your choice).
Items to be turned in:

(a) Your `note` function.

(b) Your `play_scale` function.

(c) A function that outputs sound for one of the given musical pieces

(d) (**MAS.510** Now that you have listened to your synthesized notes, aren't the transitions between different notes very choppy and abrupt? Generate a function that outputs the same piece of music you had selected in (c) but with a smoother transition or basically gives the notes a nice fade. *Hint: make a mathematical expression or function that reduces the magnitude of the note against time.*

**SOLUTION :**

*(a)*
```
function tone = note(keynum,dur)
% NOTE  Produce a sinusoidal waveform corresponding to a
%       given piano key number
%
% usage:  tone = note(keynum, dur)
%
%     tone = the ouput sinusoidal waveform
%   keynum = the piano keyboard number of the desired note
%      dur = the duration (in seconds) of the output note
%

%fs = 11025;
fs = 8000;
tt = 0:1/fs:dur;

if (keynum ~= 0)
  freq = 440*2^((keynum-49)/12);
  tone = sin(2*pi*freq*tt);
else
  tone = zeros(1,length(tt));
end
```

*(b)*
```
function play_scale()
% PLAY_SCALE  Function to synthesize a major scale

keys = [ 40 42 44 45 47 49 51 52];
% Notes: C  D  E  F  G  A  B  C
% key #40 is middle-C
%
dur = 0.25 * ones(1,length(keys));
fs = 11025;
xx = zeros(1,sum(dur)*fs+1);
n1 = 1;

for kk=1:length(keys)
  keynum = keys(kk);
  tone = note(keynum,0.25);
  n2 = n1 + length(tone) - 1;
  xx(n1:n2) = xx(n1:n2) + tone;
  n1 = n2;
end
sound(xx,fs)
```

*(c) & (d)*

```
function out = play_song(t,tdur,b,bdur,tempo)
% PLAY_SONG  Function to play a song
%
% usage:  out = play_song(t,tdur,b,bdur,tempo)
%
% t = vector of key numbers (treble clef)
%  tdur = vector of durations (treble clef)
% b = vector of key numbers (bass clef)
%  bdur = vector of durations (bass clef)
% tempo = tempo in beats per minute

dur = sum(tdur)*60/tempo;
fs = 11025;
xx = zeros(1,dur*fs+1);
n1 = 1;

for kk=1:length(t)
  ttone = note(t(kk),tdur(kk)*60/tempo);
  en = env(length(ttone));
  n2 = n1 + length(ttone) - 1;
  xx(n1:n2) = xx(n1:n2) + en.*ttone;
  n1 = n2;
end

yy = zeros(1,dur*fs+1);
n1 = 1;

for kk=1:length(b)
  btone = note(b(kk),bdur(kk)*60/tempo);
  en = env(length(btone));
  n2 = n1 + length(btone) - 1;
  xx(n1:n2) = xx(n1:n2) + en.*btone;
  n1 = n2;
end

out = xx+yy;
soundsc(out,fs)
```

## Problem 5: *DSP First* Lab 4

You only need to synthesize one of the FM instruments (bell or clarinet).

Items to be turned in:

(a) Your `mychirp` function (this should look familiar :).
(b) Your `beat` function.
(c) Plots and answers to questions specified in C.4.3.3.
(d) Either your `bellenv` and `bell` functions, or your `woodwenv` and `clarinet` functions.

**SOLUTION :**

*(a)*
```
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)
  fsamp = 8000;
end

t = 0:1/fsamp:dur;
f = (f2 - f1)/dur;

xx = cos(pi*f*t.^2 + 2*pi*f1*t);
```

*(b) The spectrogram with a window length of 2048 (on the left) has enough frequency resolution to resolve both frequencies of the beat signal. The spectrogram with the window of length 16 (on the right) has a much poorer frequency resolution, but much better time resolution. It is impossible to tell from this spectrogram that there are two distinct frequencies, but the timing of the beats are indicated by the vertical stripes.*

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

tt = 0:1/fsamp:dur;
xx = sumcos([fc-delf fc+delf], [A B], fsamp, dur);
```

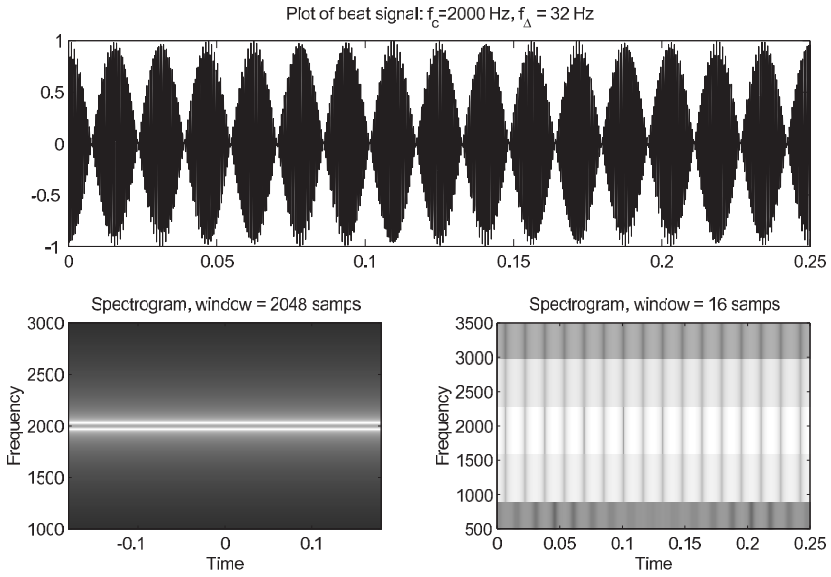


Figure 3: Plot and spectrograms of beat-note signal

(c)
```
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 an exponential decay for positive tau

yy = exp(-[0:1/fsamp:dur]/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

t = 0:1/fsamp:dur;
env = bellenv(tau, dur, fsamp);

xx = env.*cos(2*pi*ff(1)*t + Io*env.*cos(2*pi*ff(2)*t));


function  [y1,y2] = woodwenv(a,s,r,Fs)
%WOODWENV produces amplitude and mod. index envelope functions
% for woodwinds
%    usage: [y1,y2] = woodwenv(a,s,r,Fs);
% where a = attack TIME
%       s = sustain TIME
%       r = release TIME
%      Fs = sampling frequency (Hz)
% returns:
%      y1 = amplitude envelope
%      y2 = modulation index envelope
%  note: attack is exponential, sustain is constant, release is exponential
```

```
ta = 0:1/Fs:a;
y1 = exp(ta/a*1.5)-1;
y1 = y1/max(y1);
y1 = [y1 ones(1,s*Fs)];
tr = 0:(1/Fs):(r/2);
y3 = exp(((r/2)-tr)/r*3)-1;
y3 = y3/max(y3)/2;
y4 = 1-y3(length(y3):-1:1);

y2 = [y1 ones(1,(r*Fs) + 1)];
y1 = [y1 y4 y3 0];

len = min([length(y1) length(y2)]);
y1 = y1(1:len);
y2 = y2(1:len);


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

fc = f0*2;
fm = f0*3;

Ienv = scale(Ienv,-2,4);

t = 0:1/fsamp:dur;

yy = Aenv.*cos( 2*pi*fc*t + Ienv.*cos(2*pi*fm*t) );
```

## Problem 6: Playing with sounds in your environment (for MAS.510)

For this exercise you'll need to be able to get recorded sounds into your computer in WAV format. I'll be talking about how to do this in the commonly used operating systems in recitation on Friday. I'll also be talking a little bit about the mechanics of conversions between different sound formats (like say mp3).

(a) Record a simple "pure" tone. Choose any length of time you desire. Plot the sound in time and also using a spectrogram (use the `specgram` function in MATLAB). Try to determine the dominant pitch in the simple tone and justify how it was determined.

(b) Record your favorite piece of music or any sound for a time duration of 2 secs(in WAV format, using `wavread` command in MATLAB). Plot the spectrogram of the sound you just recorded. Suggest a way in which you could determine the pitch from the spectrogram if you didn't know what it was to begin with.

**SOLUTION :**

*(a) Whistling isn't exactly a completely pure tone, but you can see in the spectrogram that one frequency is heavily dominate.*

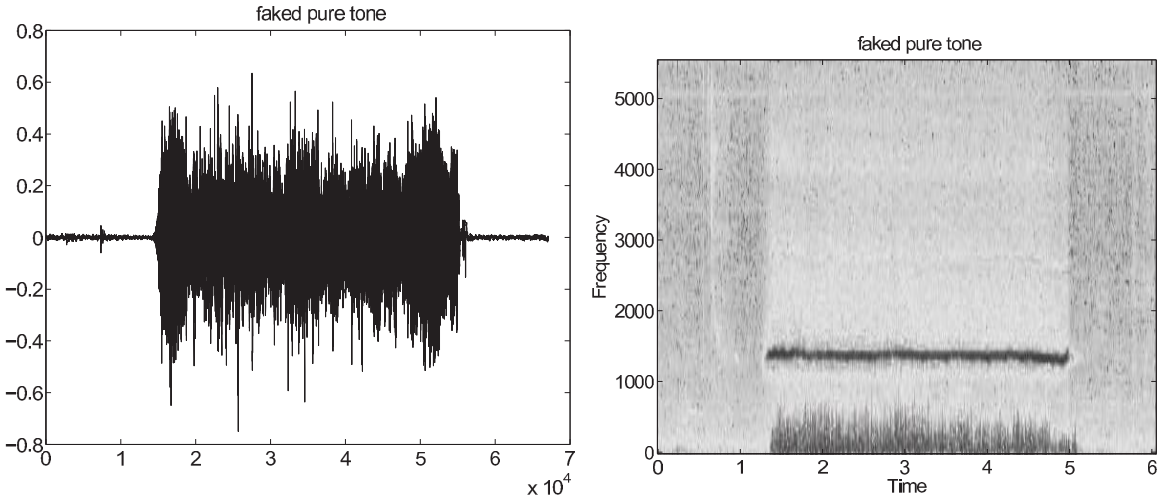


Figure 4: Plot and Spectrograms of the T.A. Whistling

*(b) The recorded music is the first 10 seconds of Gold Dust Woman by Fleetwood Mac. The vertical stripes in the spectrogram correspond to drum beats in the music.*

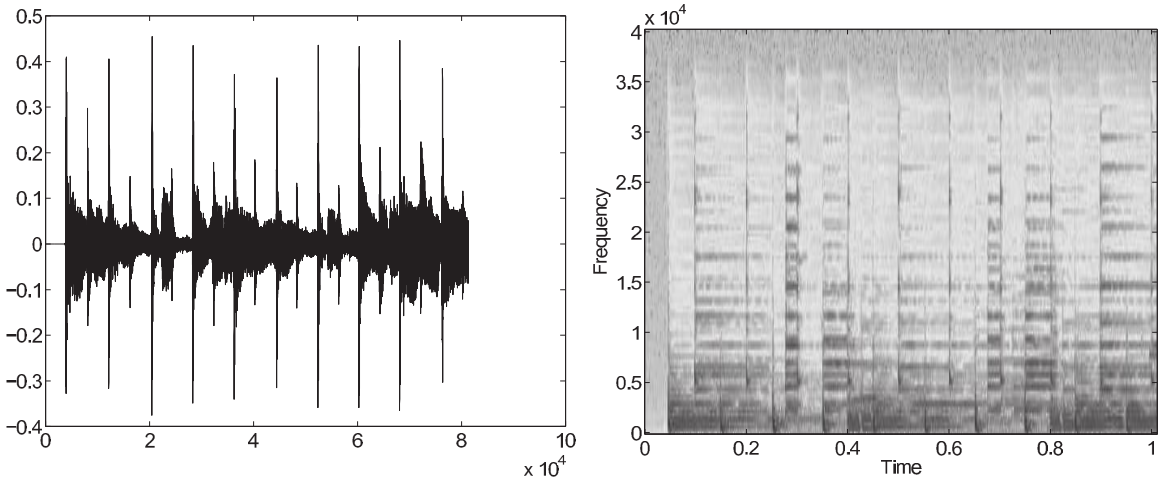


Figure 5: Plot and Spectrograms of the First 10 Seconds of Gold Dust Woman