ECE3230 - Practicum III

Synthesis of Audio Signals

Reporting Requirements: Follow report instructions for Practicum I.

Objective: To gain experience with and an understanding of the representation of a signal in terms of its frequency content. Critical to this is the idea of synthesizing signals as a linear combination (i.e. weighted sum) of sinusoids of different frequencies.

Background:
Sampling was discussed on an elementary level (adequate for this Practicum) during the lecture in signal processing course, so review your Section 1 Notes in Dr. Kevin Buckley's notes.

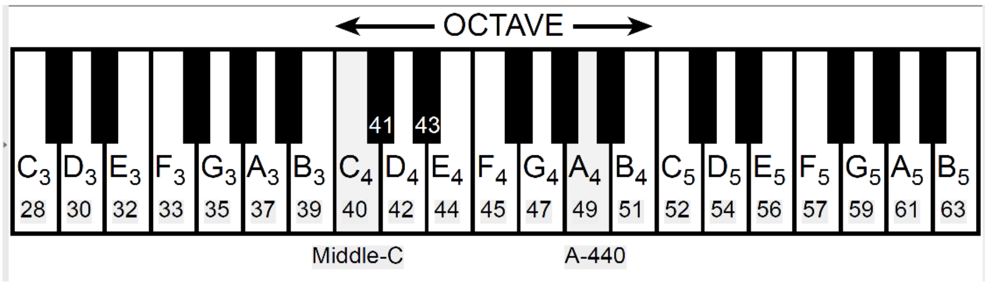
Basically, the ideal A/D converter assumed here forms a discrete time signal x[n] from an underlying continuous time signal x(t) as

$$x[n] = x(nT_s) \tag{1}$$

where T_s is the sampling interval in seconds (so $f_s = 1/T_s$ is the sampling frequency in Hz which is samples/second).

A D/A converter operates to reconstruct x(t) by effectively interpolating between the samples (in x[n]). As you will see later in the lecture in ECE 3225, x(t) can be exactly reconstructed only if $f_s > 2f_{max}$ where f_{max} is the maximum frequency component of x(t).

Music Score: In this Practicum, you will learn how to synthesize a musical score. Please refer to the additional references given at the end of this document to learn more on reading a music score and on converting notes to their fundamental frequencies on a piano keyboard with key numbers as given below:



Review these before the practicum sessions.

Depending on the first letter of your last name, as assigned below, you will synthesize one of the four music scores.

1. If your last name begins with A,B,C,D,E,F or G, work on Beethoven's Fur Elise.
2. If your last name begins with H,I,J,K,L,N or M, work on Vivaldi's Spring
3. If your last name begins with O,P,Q,R,S or T, work on Brahm's Lullaby.
4. If your last name begins with U,V,W,X,Y or Z, work on Beethoven's Fifth Symphony.

Alternatives: As an alternative you may obtain permission from the instructor to either:

a) work with another score that you select and provide; or
b) generate a sequence of voice sounds (e.g. vowels a,e,i,o and u).

Procedures: This practicum follows Lab C.3 in the reference text "DSP First: A Multi-media Approach", by McClellan, Schafer and Yoder, Prentice Hall, 1998.
This Lab consists of 5 procedures, the first 3 of which are to be performed the 1-st week. Procedures 4 and 5 should be performed the 2-nd week. Read the Background above before the week-1 lab. Procedure 1 should also be complete before the week-1 lab session. Procedures 2 & 3 should be completed at the week-1 lab session. Procedure 4 should be completed before the week-2 lab session. Procedure 5 should be completed at the week-2 lab session.

1. ***sumcos - a sinusoid linear combiner:*** Write an m-file that will synthesize a waveform of the form

$$x(t) = \sum_{k=1}^{M} A_k \cos(2\pi f_k \mathrm{t} + \varphi_k) \qquad (2)$$

where M is the number of sinusoidal components, and A_k, $\omega_k = 2\pi f_k$ and φ_k are, respectively, the amplitude, angular frequency and phase of the kth sinusoidal component. The first few lines of the m-file should look like:

```
function xx = sumcos(f, X, fs, dur)
% SUMCOS Function to synthesize a sum-of-cosine signal usage:
% xx = the synthesized signal
% f = vector of frequencies (in Hz)
% X = vector of complex exponentials: (Amp*exp{j*phase})
% fs = the sampling rate in samples per second
% dur = time duration of the signal
%
% Note: f and X must be the same length
% X(1) corresponds to frequency f(1), X(2) corresponds to frequency f(2), etc.
```

Within the m-file, you can use the command "length(f)" to determine the number of sinusoidal components. You should use a "for" loop over different frequencies. Within this loop use the Matlab "abs" and "angle" functions to strip the magnitude and phase off of each X(k). Do not generate complex-valued sinusoids.

Test your "sumcos" m-file with the following cases:

(a) xx = sumcos([20], [1], 500, 0.25);
(b) xx = sumcos([40], [-j/2], 500, 0.25);
(c) xx = sumcos([20 40 60], [1 -1 .25$e^{j\pi/4}$], 500, 0.25).

Plot the results using the proper independent axis scale.

2. ***Generating, plotting and listening to tones*:**

(a) Compute a vector *x1* of samples of a sinusoid with A = 100, $\omega_0 = 2\pi(1000)$ and $\varphi = 0$. Use a sampling rate of 8000 samples/second, for a duration of 2 seconds. Using the Matlab command "soundsc( )" (in Matlab, type "help soundsc" for an explanation of this command), play the resulting vector through the D/A converter of your computer. Listen to the output (you may need headphones). Plot (use the "plot" command) x1 for $0 \le t \le 0.02$ seconds.

(b) Compute a vector *x2* as samples, over a 2 second duration, of a sinusoid with A = 100, $\omega_o = 2\pi(500)$ and $\varphi = \pi/3$. Plot x2 for $0 \le t \le 0.02$ seconds. Again, listen to the reconstructed signal. How does it compare to the signal from 2.(a)?

(c) Concatenate the two signals from 2(a) and 2(b) as follows:

xx = [x1 zeros(1, 2000) x2]; (3)

and listen to this signal. Explain what you hear. Plot the spectrogram of xx using the "spectrogram" command. (Type help spectrogram to determine its usage. Make sure your axes are properly labelled with time and frequency *e.g. use spectrogram(xx,256,250,512,fs, 'yaxis').)*

3. *Generating, plotting and listening to musical notes*:

(a) Write an m-file, called "note.m", which is a function that generates a desired note for a desired duration. A note will be a periodic signal consisting of harmonically sinusoidal components of the form: $x(t) = \sum_{k=1}^{M} A_k \cos(2\pi k f_o t)$ where f_o is the fundamental frequency, A_k is the harmonic amplitude of the kth harmonic which is the sinusoid oscillating at frequency kf_o

The note.m function should have the following form:

```
function tone = note(keynum,fs,harmonic nums,harmonic vals, dur)
% NOTE Produces a sinusoidal signal corresponding to a
% piano key number
% usage:
% tone = output signal
% keynum = the piano keyboard number of the desired note
% fs = specified sampling frequency
% harmonic nums = array containing note harmonic numbers
% harmonic vals = array containing note harmonic amplitudes
% dur = time duration of the signal
%
fund freq =
tone =
```

For the "fund freq" line, use the formula that appears in the reference document as given in the additional references below which can also be found on blackboard.
Start with the reference note (middle C (261.6 Hz.) or A−440 is recommended), and solve for the fundamental frequency based on this reference. For "tone", use your sumcos function to superimpose the harmonic components specified by "harmonic nums" at the levels specified by "harmonic vals".

(b) Generate a note of 2 second duration, at 11,025 samples/sec. rate, consisting of the 1-st five harmonics at levels [1 .5 .3 .2 0.3], that represents the note D$_5$ (key 54) above A−440 (key 49; 440 Hz.). Listen to this note and plot its spectrogram.

(c) Complete the following m-file to play scales in Octave 5:

```
function  play_scale.m
%
keys =
% keys = array containing Notes C5 D5 E5 F5 G5 A5 B5 C6 in Octave 5
% key # 52 is C5
%
dur = 0.25 * ones(1,length(keys));
fs = 11025;
```

```
harmonic nums = 1:5;
harmonic vals = [1 .5 .3 .2 .3];
xx = zeros(1,sum(dur)*fs + 100);
n1 = 1;
        for kk = 1:length(keys);
         keynum = keys(kk);
        tone =
        n2 = n1 + length(tone) - 1;
        xx(n1:n2) = xx(n1:n2) + tone;
        n1 = n2+1;
        end
soundsc(xx, fs)
```

For "tone", use your "note.m" to generate the signal for each keynum of its duration (dur(kk)).

(d) Plot the spectrogram of xx, comment on your observations.

Use a sampling frequency of f_s = 11025 samples/sec. for all steps of procedure 4. and 5.

4. *Determine the frequencies vs. time for the audio you are to synthesize*. Record this information in a table. (For example, if you are generating a musical score, before the practicum session determine the keynumber, start time and time duration needed for each note. Record these in a table.)

5. *Synthesize your musical score or sequence of voice sounds:*

(a) In Matlab, synthesize your musical score as a sequence of superimposed notes and their durations. Use your "sumcos" or "note" function similar to "play_scale.m". There are various approaches to this. For example, each track (e.g. base and treble) can be generated in a separate array. For each track, concatenate the notes using an m-file similar to "play scale.m". Then mix (sum) the separate tracks. If you choose to generate a sequence of voice sounds, alter this procedure as needed.

(b) Using the Matlab command "soundsc", play your audio signal for the instructor or TA, so that s/he can verify the completion of this task.

(c) Generate the spectrogram of the resulting signal.

(d) Compare and comment on the relationships between the audio signal, the spectrogram and what you hear.

You might achieve a more pleasing sounding signal by applying a separate amplitude to each note or vowel, and/or by changing the note amplitude over its duration (e.g. tapering the amplitude so its intensity decreases over time). Changing harmonic content can result in sounds similar to other musical instruments or speakers. Feel free to experiment with these and other tweaks to the basic signal you generated.

Additional References:

[1] Piano keys, musical notes and frequency mapping: Please refer to the document on blackboard: pianokey-frequencies.pdf
[2] How to read musical notes: Please go to the website:
http://www.musicnotes.com/blog/2014/04/11/how-to-read-sheet-music/

Practicum III

Instructor/TA Sign Off Sheet

Student's Name: ____

1. Procedure 1: plots for (a,b,c) ________________________________

2. Procedure 2: plots for (a,b,c) _____________________________
   Comment on the questions

3. Procedure 3(d): audio and spectrogram plot ______________________
   Comment on the questions

4. Procedure 5(b,c): audio and spectrogram ________________________
   Comment on the questions