ECE 2200 Fall 2014
Lab 2: Audio Signals and Spectrograms

Lang Tong
Irwin and Joan Jacobs Professor of Engineering

September 29–October 3, 2014

1 Pre-Introduction

Because of Preliminary Exam 1 last Thursday, there has been no time to do all of the work described in the Pre-lab. But I still want to have lab this week. Therefore, I am giving you all of the `.m` files and do not want you to turn in any Pre-lab report!

As with Lab 1, if you do the lab and you show your TA your results then you get 100%!

2 Introduction

In this lab, you will experiment with audio signals in MATLAB. Additionally, you will be focusing on looking at signals in the frequency domain and manipulating them there. You will be introduced to the spectrogram: a useful tool for the analysis and visualization of "non-stationary" signals whose frequency contents vary with time.

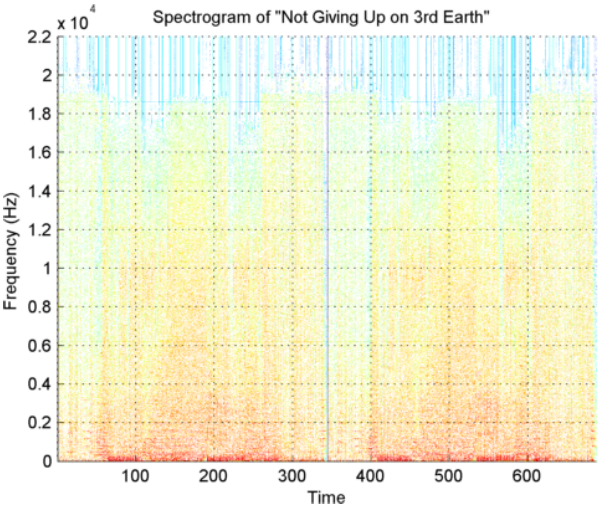

Figure 1: Spectrogram of a song.

The spectrogram, shown above, is a plot of a sequence of complex spectra, each corresponding to a segment of a signal at a particular time. In Shazam (`http://www.shazam.com/`), the spectrogram is the starting point from which the features of a piece of music is extracted.

3 Grading

1. Pre-lab: song for Shazam database, MATLAB functions, plots, and answers (40%)
2. In-lab: completing tasks, demonstrations (60%)

4 Pre-lab

Submit all of your code as well as all the plots and answers to the pre-lab questions on paper at the beginning of the lab section. Throughout this lab, assume a sampling frequency of $f_s = 44.1$ kHz.

4.1 sin_gen.m

Write a function that generates a sinusoid with a specified frequency and phase.

```
function x = sin_gen(f, ph, d, fs)
% Returns a sinusoid with frequency f [Hz], phasor representation ph,
% and duration d [seconds], sampled at frequency fs [Hz].
```

1. Generate an A4 (440 Hz) sinusoid using `sin_gen()` with a duration of one second.
2. Give the Fourier Series coefficients for the signal of Item 1 assuming that the signal starts at time $-\infty$ and continues on until time $+\infty$ rather than just existing for one second.
3. Generate the signal of Item 1 from its Fourier Series coefficients in Matlab by using Matlab to evaluate the synthesis equation

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}. \tag{1}$$

4. Plot the sampled signal and the synthesized signal on the same plot. They should overlap.
5. **Download the `fs_fft()` function from Blackboard and read its usage specifications.** Use it throughout the lab to compute Fourier series coefficients and use it to plot the magnitude and phase spectra. Compare the spectra with your previous plotted analytically derived spectra.
6. **Also download the `toofast_toofourier()` function from Blackboard and read its usage specifications.** Use it throughout the lab to generate complex magnitude and phase spectra when the signal is not well-represented by a sum of harmonics.

4.2 Compute base frequencies

Hopefully everyone knows a tiny amount about “Western” music. An octave is a doubling in frequency. A piano has white keys and black keys. In one octave the white keys are C, D, E, F, G, A, B. Between the white keys are black keys that are “sharps”/“flats” where sharps are denoted by, e.g., A#, and flats are denoted by, e.g., Bb. The notes A# and Bb are the same note. There is a sharp/flat between every pair of white keys except E and F. This means that there are 12 keys total (white and black) per octave.

Compute the base frequencies for the notes C4, D4, E4, F4, G4, A4, and B4 where A4 has freqency 440 Hz. Neighboring notes have a constant frequency ratio of $r = 2^{1/12}$, e.g., if A4 has frequency 440 Hz, then A#/Bb (the next key to the right on a piano) has frequency $440 \times 2^{1/12}$.

4.3 `beat.m`

Write a function that generates the sum of two sinusoids with specified frequency and phases.

```
function x = beat(freq1, phase1, freq2, phase2, d, fs)
% Returns the sum of two sinusoids with specified frequency and phases
% with duration d sampled at frequency fs
```

Generate the sum of two sinusoids (A4 and F4) with specified frequencies and phases with a duration of one second. Plot the magnitude and phase spectra of the resulting signal.

4.4 `amodulate.m`

Write a function that multiplies one signal with a sinusoid of certain carrier frequency ω_c.

```
function y = amodulate(x, wc, d, fs)
% Returns the amplitude-modulated version of x modulated by a cosine of unit
% amplitude and carrier frequency wc [rad/s]. The result has duration d sampled at
% frequency fs
```

Use C4 and F4 to generate a C4 sinusoid modulated with an F4 sinusoid (or vice versa) with a duration of one second. Plot the resulting magnitude and phase spectra. **Please note that there is already a built-in function named `modulate()`; avoid naming a function or variable the same name as a built-in function/variable.**

4.5 Spectrogram

Download the `instaspectrogram()` function from Blackboard and read its usage specifications. Use it throughout the lab to generate spectrograms of signals.

Download the Cornell Alma Mater recording `.mat` file from Blackboard. You can load it into MATLAB using `load cello.mat`, provided the file is in your working directory. Generate the spectrogram of the recording.

4.6 Choose a music piece!

Select a piece of music to be used as part of the class's Shazam database. Place the music file into your working directory and read it into MATLAB using the `audioread()` function. Truncate the result to about one minute in duration.

```
[y, Fs] = audioread('myjam.mp3')
myjam = 1/2*(y(1:Fs*60, 1) +  y(1:Fs*60, 2)) % Extract 60 seconds of samples
% y contains two channels, one for each column. We'd like to process a 1D signal
% so here we can average the two channels.
% myjam = 1/2*(y(Fs*60:Fs*120, 1) +  y(Fs*60:Fs*120, 2)) % Extract from 1:00 to 2:00
```

Note that older versions of MATLAB may not have the `audioread()` function. If your version of MATLAB does not have the `audioread()` function built-in, one solution (for Windows machines) is to use the `mp3read()` function. Here is how to install it:

1. Download the "mp3readwrite.zip" file from Blackboard.
2. Extract the .zip to your working directory. The contents should be in a folder.
3. In MATLAB, the folder should appear in your "Current Folder." Right-click the folder you just extracted and click Add to Path > Selected Folders and Subfolders.
4. Type "help mp3read" into your command window. If done properly, you should see the help file for the function.

Save your workspace (File > Save Workspace As while in the command window) as `netID_music.mat` and submit the `.mat` file to Blackboard under the Pre-Lab assignment. Include the song information in the submission entry box.

5 In-Lab

Complete the following tasks. You must have all of the TA checks completed by a TA in order to receive full credit for the lab.

5.1 Modulating A4

Use the `amodulate()` function that you wrote with an appropriate carrier frequency so that an A4 sinusoid gets modulated to the C4 frequency. In other words, choose a carrier frequency and use the frequency-shifting action of modulation in order to create a C4 frequency component from a pure A4 sinusoid. Note that the result will not purely be an C4 sinusoid.

Plot the magnitude and phase spectra of the resulting signal. Compute the Fourier series coefficients of the modulated signal using one of the provided functions.

5.2 A dissonant chord

Generate a one second long signal consisting of all 7 of the keys whose base frequencies you computed in the pre-lab being played simultaneously. Plot the magnitude and phase spectra of the resulting signal and check to see if the 7 highest magnitude frequency components are those of the base frequencies of the 7 keys being played.

```
y = 0;
keys = [your frequencies here];

for n = 1:length(keys)
     y = y + sin_gen(keys(n), 1, 1, fs);
end
```

5.3 Play a C scale

Generate a signal that, when played back, sequentially plays all the white keys for a (half-second long each) from C4 to B4. (C4, D4, E4, ..., B4). Plot the magnitude and phase spectra of the resulting signal as well as its spectrogram.

```
y = [];
keys = [your frequencies here];

for n = 1:length(keys)
     y = [y, sin_gen(keys(n), 1, 1/2, fs)];
end
```

TA CHECK: Play your signal and show your spectrogram. How does the spectrogram help you "visualize" the sound signal? How would it look different if the keys were played for different durations? With different keys?

5.4 Audio compression

Using the song segment that you submitted in the pre-lab, plot the magnitude and phase spectra.

Synthesize the original song while discarding:

1. 5% of the smallest magnitude frequency components
2. 10% of the smallest magnitude frequency components
3. 20% of the smallest magnitude frequency components

Here is some example code to help with this task.

```
[y44, fs44] = audioread('3rd Earth.mp3');
y = 1/2*(y44(1:fs44*60, 1) + y44(1:fs44*60, 2));

X = fft(y);                                 % Computes the Fourier spectrum of y
[Y, I] = sort(X, 'ascend');                 % Sorts the spectrum
amt = 10/100;                               % Sets the % of components to discard
I_discard = I(1:round(amt*length(I)));      % Indices of lowest magnitude components
Y(I_discard) = 0;                           % Sets them equal to zero
Y_comp(I) = Y;                              % Spectrum with components discarded

Y_compressed = ifft(Y_comp)/length(y);      % Transform back to time domain
soundsc(real(Y_compressed), fs44)           % Play back the edited audio
```

Play the resulting signals and compare with the original song segment. What do you observe?

TA CHECK: Play part of the song with 20% of the frequency components discarded. In each case above, were you able to hear a difference between the original song segment and the version with frequency components discarded? How does this relate to audio compression? Would the amount of coefficients that you can discard without dramatically altering the audio differ depending on the song?

5.5 Spectrogram peaks

Generate the spectrogram of your selected song. Give the time and frequency locations of the top ten highest magnitudes in the spectrogram. The specification of `instaspectrogram()` contains some helpful information for this task.

Some useful commands: `sort()`, `find()`

TA CHECK: Show the spectrogram of your selected song and your time-frequency locations of the spectrogram peaks. Describe how you located the top ten highest magnitudes in the spectrogram.