Digital Audio Signal Processing Exercise

Submission Guidelines:

- This exercise is good for two weeks. Deadline is on **05 April 2019, 11:59PM (Friday)**.
- Submit all codes (do not compress) to **updsp.app@gmail.com** with the subject: ***DASP_LastName***.
- Your codes must be executable from the MATLAB command line and should not require user inputs (except for functions). Use sections to separate codes for different subparts of the exercise, if necessary.
- Upon execution, figures that are required to be displayed should be displayed (no need to save images), and generated audio vectors should be played.
- All plots should have proper titles, axis labels, and legends (if applicable).
- Provide adequate comments in your codes, explaining what is happening in your programs.

Audio Synthesis

Background

An air raid siren is essentially a fan whose rotating component, the rotor, serves to draw in air which is blown out of its housing, the stator, through radially cut holes. The rotor is constructed with vanes or blades that periodically cover and open the holes as they spin, causing air to be expelled in bursts that generate a distinct tone when the rotor is spun fast enough. Two tones at different pitches can be generated by having a rotor with different numbers of blades and matching holes at two different outputs.

The resulting tones contain both odd and even harmonics, with the lower harmonics being the most prominent. Sirens usually sweep upwards in frequency while being sounded before settling on their natural tones once the rotor reaches its maximum speed and sweep downwards in frequency as the rotor winds down.

In this exercise, we will be synthesizing a two-tone air raid siren in MATLAB using additive synthesis, a low pass filter, and (unconventional use of) frequency modulation with a generated "pitch envelope" signal. The siren will be synthesized at a sampling frequency of 16000 Hz and have two tones with fundamental frequencies 580 Hz and 690 Hz, and odd and even harmonics up to the 6^{th} harmonic.

(Source: http://civildefencemuseum.ca/how-sirens-work)

Steps

1. Generate the pitch envelope – a normalized signal that will define the modulation of the frequency of a sinusoid over time from 0 to a given maximum frequency deviation
    a. Generate 5 seconds of an *exponential rise* from 0 to 1, $\tau = 5$
    b. Generate 3 seconds of the *maximum value of the exponential rise after 5 seconds, held constant*
    c. Generate 5 seconds of an *exponential fall* from the hold value in (b) to 0
    d. Concatenate the rise (a), hold (b), and fall (c) parts of the envelope

2. Synthesize a tone of the siren via additive synthesis
    a. For each nth harmonic, generate a sinusoid that sweeps from 0 to frequency f_n over your generated pitch envelope using frequency modulation via the *ffmod()* function - the pitch envelope serves as the message signal; the carrier signal is initially 0 Hz, with a frequency deviation of $\pm f_{n.}$
    Note: see function help page for information on function usage
    b. Get the summation of all harmonics generated as in (b)
3. Repeat for the second tone
4. Add the two tones and normalize the resulting sum to prevent clipping any audio playback or file output
5. Apply a low pass filter with cutoff at around 3000 Hz to the sum of the two tones in order to shape the higher harmonics

Save the code used to synthesize the siren as *LastName_synthesis.m* and your synthesized air raid siren as *LastName_synthesis.wav*.

Free Audio Analysis and Manipulation

Given the recorded audio signal, use the different audio manipulation methods to make the signal have at least 3 different tones. *Note: You can use other manipulation techniques not included in the lectures. You may also use this link as guide to the different frequencies of notes.* *https://pages.mtu.edu/~suits/notefreqs.html* .

Save the code used to manipulate the signal as *LastName_manipulation.m* and your manipulated audio signal as *LastName_tone1.wav* for the first tone, *LastName_tone2.wav* for the second tone and so on.