Full Name: ______________________________

EEL 3135 (Spring 2021) – Lab #02 **Due: 11:59 PM, Feb. 2 - Feb. 8 (On Lab Day)**

Submission Notes

- Your laboratory solutions should be submitted on Canvas as a single published MATLAB PDF.
- Use the provided skeleton code as the basis for your solutions (easier for you and the graders).

Question #1: *(Audio Synthesizer)* Download `eel3135_lab02_comment.m` from Canvas, replace each of the corresponding comments with the corresponding descriptions. This is designed to show you how to work with audio in MATLAB.

Note: You should run the code to help you understand how it works and help you write your comments. You will use elements of this MATLAB code for the rest of the lab assignment.

Question #2: Consider the three sinusoids below:

$$s_1(t) = 3\cos\left(800\pi t - \frac{\pi}{3}\right)$$
$$s_2(t) = 2\cos\left(800\pi t - \frac{\pi}{4}\right)$$
$$s_3(t) = 2\cos\left(1200\pi t - \frac{\pi}{4}\right)$$

(a) Use the sampling frequency $f_s = 8000$ Hz to generate the three sinusoids $s_1(t)$, $s_2(t)$, and $s_3(t)$ over the time range from $t = 0$ to 0.01 second in MATLAB. Put the generated sinusoids in MATLAB vectors `s1`, `s2`, and `s3`, respectively. Plot the three sinusoids in three separate figures or subfigures. Label the axes of your figures appropriately. Use MATLAB's `axis` function to plot only the first four periods of the sinusoids.

(b) Use the MATLAB function `audiowrite` to generate a `.wav` file for each of the three sinusoids in (a). Note that you'll need to rescale the values in your vectors to within the range $[-1, 1]$ before passing them as input to `audiowrite` by doing, say,

```
>> s1_scaled = s1/max(abs(s1));
```

and then pass `s1_scaled` to `audiowrite` as input. Remember to set also the sampling frequency argument in `audiowrite` to the value of f_s in (a). Save $s_1(t)$ to `s1.wav`, and so on. **Submit the `.wav` files together with your published report to Canvas.** [1]

(c) Let $x_1(t) = s_1(t) + s_2(t)$. Plot the first four periods of $x_1(t)$ and label the figure appropriately. Use your hand-calculated expression of $x_1(t)$ to verify the plot. Check whether the amplitude, phase, and frequency of the expression agrees with your theoretical expectation.

(d) Let $x_2(t) = s_2(t) + s_3(t)$. Plot the first four periods of $x_2(t)$ and label the figure appropriately. Use your hand-calculated expression of $x_1(t)$ to verify the plot. Check whether the amplitude, phase, and frequency of the expression agrees with your theoretical expectation.

[1] You can also use the MATLAB function `soundsc` to listen to the sinusoids without saving a file. Remember to set the sampling frequency argument in `soundsc`. Note: if you use MATLAB through UF Apps or the ECE cluster, you may not be able to hear any sound played by `soundsc`. In such case, play the `.wav` files generated in (b) to listen to the sinusoids instead.

Question #3: This exercise expands the rudimentary music synthesizer in the comment code. When you press and hold a key, a step sinusoid (more commonly refer to as a *note*) with a desired frequency, phasor (amplitude and phase), and duration is played depending on the particular key we press, how hard we press it, and for how long we hold it, respectively.

(a) Create a script that uses `key_to_note` and `build_song(As, keys, durs)` from the previous question to play "Mary had a Little Lamb." Let `fs=8000;` The `A` values, `key` values, and `dur` values for the song can be stored in vectors:

```
As   = [1  1  1  1  1  1  1  1  1  1  1  1  1];
keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
durs = [1  1  1  1  1  1  2  1  1  2  1  1  2]*1/4;
```

Run the script and **get the sound checked off by your lab TA before moving on.**

(b) Musical instruments generate signals with multiple sinusoids. Each sinusoid has varying amplitudes and phases and the frequencies are integer multiples of a fundamental frequency (just like a Fourier series). The harmonics determine the timbre (sound quality) of the note.

The following table contains the harmonics (integer multiples of frequency), amplitudes, and phases to emulate a trumpet. Create a new function `key_to_note_trumpet` that creates a trumpet sound.

k – Harmonic	A_k - Amplitude	ϕ_k – Relative Phase
1	0.1155	-2.1299
2	0.3417	+1.6727
3	0.1789	-2.5454
4	0.1232	+0.6607
5	0.0678	-2.0390
6	0.0473	+2.1597
7	0.0260	-1.0467
8	0.0045	+1.8581
9	0.0020	-2.3925

(c) Create a new version of `build_song(As, keys, durs)` and name this new function `build_song_trumpet(As, keys, durs)`. Use `key_to_note_trumpet` in the function and recreate the Mary song with trumpet notes. Make `key` now correspond to the fundamental frequency. **Get this sound checked off by your lab TA.**

(d) Create a new song building function that accepts a different set of parameters: `build_song_time(As, keys, start_time, end_time)`. Replace `dur` input with the following two inputs (both are in units of seconds):

```
start_time =  [0  1  2  3  4  5  6  7  8  9  10 11 12]*1/4;
end_time   = ([0  1  2  3  4  5  7  8  9  11 12 13 15]+0.2)*1/4;
```

Run the script and **get the sound checked off by your lab TA.** You can use the trumpet or the original sound.

(e) Plot the output of `build_song`, `build_song_trumpet`,and `build_song_time` versus time (in seconds) using the `plot` function. Be sure to properly label all axes (Amplitude and Time), and title (using the `title` function) your plots to differentiate them.

(f) **Answer:** What are the key differences between the plots in (d)?