Laboratory Project M2:
Manipulations of Sound Waveforms

***Abstract*-You will focus on the use of array indexing and functions to modify a sound waveform.**

I. Preparation

Read *Matlab Primer* pages 1-6 to 1-12, 1-17, 1-24 to 1-25, 1-27 to 1-28, and 2-2 to 2-25. Be sure you **bring your earbuds** to all labs so you can listen to sounds produced by your Matlab® programs.

II. Learning Objectives

1) Use indexing arrays to extract or modify segments of a sound sample.
2) Use predefined functions to distort or add noise to the sound sample.

III. Procedure

A. Lab Report

For each part of this lab, you will write a script file. At the conclusion of the lab, send your TA an email whose body is the contents of all these script files. That is, put all the script file contents together, put them in the body of the email unless the TA instructs you to do otherwise. Use comments in the script files to identify them and how they work. For part *H*, include comments in your script file describing your sound effect. Demonstrate your sound effect to the TA before leaving the lab.

B. Read Music File into Matlab®

Create a Matlab script file called load_sound.m to read a sound sample into Matlab®. Matlab® has the following built-in sounds that may be easily loaded into the Matlab® workspace: chirp, gong, handel, laughter, splat, and train. For example, `>> load chirp` followed by `>>my_music = y;` reads the chirp sound from a file into an array called `y` and then sets my_music equal to `y`.

Listen to the music sample using the sound( ) function in Matlab®.

C. Plot First 100 samples of Music

Create a script file called start_sound.m for this part of the lab. Using the colon operator, create an array that contains the integers from 1 to 100. Use this array as the index of my_music to extract the first 100 values. Plot these first 100 values versus time, label the *x*-axis and *y*-axis appropriately, and add a title to the plot.

Note: Where necessary below, you may use the colon operator, as you have here, to shorten the sound sample. An array of 8000 samples corresponds to one second of sound. If you use a sample that is too short, you may not be able to hear it.

Note: There is a plotting lab on the course website that you may wish to refer to.

D. Down Sampling

Create a script file called down_samp.m for part *D*. Using an integer array for indexing, create a new array that is the music array with every other sample deleted. Hints:

1) To delete a sample, set its value to [ ].

2) use the colon operator to create indices of values you wish to delete. Use the colon indexing on the left side of the equals sign.

Play the sound using the sound( ) function and, in a comment, describe how it sounds.

E. Chopping

Call the script file for this part of the lab chop_it.m. Using an integer array for indexing, set segments of (a copy of) my_music array to zero. The my_music array will have n samples on following by n samples zeroed out. Use the last two digits of your student ID number as the value of n. This pattern then repeats to the end of the array. Example (last two digits of student IC number are 04):

my_music = [5, 2, -3, -7, -4, 6, 9, 12, 8, 3, -2, -5, -1, 5, 8, 3, 1, -1, ...]

my_music_chop = [5, 2, -3, -7, 0, 0, 0, 0, 8, 3, -2, -5, 0, 0, 0, 0, 1, -1, ...]

Hints:

1) Create an array of the first n ones and n zeros. Use [ ]'s and a ones command and a zeros command to do this.
2) Use the repmat function to create the repeating array of ones and zeros that you will multiply my_music with element-by-element. repmat is like a rubber stamp that makes copies of an array to create a larger array. You tell repmat what array to duplicate, and you tell repmat how many copies of that array you want vertically and horizontally. Example: (replicating a 2 x 2 matrix M = [1, 2; 3, 5])

```
>> big_array = repmat(M, 2, 3)  % 2 copies of M vertically and 3 copies of M horizontally
big_array =
     1     2     1     2     1     2
     3     5     3     5     3     5
     1     2     1     2     1     2
     3     5     3     5     3     5
```

3) Type "help repmat" at the Matlab® prompt to learn more about repmat.

Play the sound using the sound( ) function and, in a comment, describe how it sounds.

F. Concatenated speech

For this part of the lab, call your script file scramdel.m and use the `handel` sound. Using the method described in part *B*, above, extract portions of the music corresponding to several syllables of words being sung. Store these syllables in arrays, and concatenate them in a different order to create a nonsense phrase. Listen to your sound, and describe in comments what reordering was done and how it sounds.

G. Nonlinear function

For this part of the lab, call your script file distort_it.m. Apply a weighted sum, (e.g., 0.2 * func1(my_music) + 0.3 * func2(my_music)), of two (but not just the first two) of the following functions (or any other functions you may discover in Matlab®) to every value in the my_music array and listen to the result.

.^3 [cube every sample value] power() abs() exp() sign() sinh() erf().^2

For a list of interesting functions you might use, type "help elfun" at the >> Matlab® command prompt. In a comment, describe how your distorted waveform sounds.

H. Noise

For this part of the lab, call your script file noise_shaped.m. For a list of interesting functions you might use, type "help elfun" at the >> Matlab® command prompt. Describe in comments how your distorted waveform sounds. Your task is to create an array of noise

samples, alter it by applying functions to it, and add this noise to your music sample. That is, you will create an array equal to my_music plus shaped random noise. If your student ID # is even, use the rand() function to generate your initial array of noise samples. If your student # is odd, use the randn( ) function to generate your initial array of noise samples.. Before adding the shaped noise to the music, multiply the shaped noise by a constant equal to some fraction less than one (exact fraction your choice) of the maximum value of the my_music array. Use the max( ) function to determine this maximum value.

I. Fade

Call your script file for this part of the lab fade_out.m. Your task to use the `exp` (exponential) function applied to a scaled array of values from 1 to 40,000 to create an array of values that decays over 5 seconds from 1 to 0.01. Solve the following equation to find the scaling factor, k, for the array that contains values from 1 to 40000. Note that k will be negative.

$$e^{k(40000)} = 0.01$$

After scaling the array of values from 1 to 40000, multiply my_music by this array (point-by-point) to create a music sample that fades out. Note that you will have to shorten my_music to make it the same length as your exponential decay array. Play the sound using the sound( ) function and comment in your script file on how it sounds . In particular, comment on whether the sound seems to fade out at a steady rate.

REF: [1] The Mathworks, Inc, *Matlab® Primer,* Natick, MA: The Mathworks, Inc, 2012.