Digital Signal Processing (ECE 4750)

Project 1

Submission Type: Canvas, Online

MATLAB practices. You must prepare a report containing the answers to each question, include the figures and results. Also, you will need to submit your code scripts.

1) (10 points) MATLAB Practice: Explain what 'polyval' and 'polyfit' commands do in MATLAB.
2) (10 points) MATLAB Practice: Explain what 'filter' command do in MATLAB and what the input arguments can be.
3) (25 points) MATLAB Practice: **Echo Generation**-The most basic of all audio effects is that of *time delay*, or echoes. It is used as the building block of more complicated effects such as reverb or flanging. In a listening space such as a room, sound waves arriving at our ears consist of *direct* sound from the source as well as *reflected* off the walls, arriving with different amounts of attenuation and delays. Echoes are delayed signals, and as such are generated using delay units. For example, the combination of the direct sound represented by discrete signal $y[n]$ and a single echo appearing *D* samples later (which is related to delay in seconds) can be generated by the equation of the form (called a difference equation)

$$x[n] = y[n] + \alpha y[n - D], |\alpha| < 1$$

where $x[n]$ is the resulting signal and α models attenuation of the direct sound. Difference equations are implemented in MATLAB using the filter function. Available in MATLAB is a short snippet of Handel's hallelujah chorus, which is a digital sound about 9 seconds long, sampled at 8192 samp/sec. To experience the sound with echo in (1.2), execute the following fragment at the command window. The echo is delayed by *D* = 4196 samples, which amount to 0.5 sec of delay.

```
load handel; % the signal is in y and sampling freq in Fs
sound(y,Fs); pause(10); % Play the original sound
alpha = 0.9; D = 4196; % Echo parameters
b = [1,zeros(1,D),alpha]; % Filter parameters
x = filter(b,1,y); % Generate sound plus its echo
sound(x,Fs); % Play sound with echo
```

Create an mfile that contains the above set of command lines. Then, listen to the final sound 'x' and explain what you learned from this problem. Don't forget to attach your code.

4) (25 points) MATLAB Practice: In this problem you will learn how to convert the sampled results back to somehow a continuous domain and interpolation. This is a surrogate problem showing how D/A converter works (using zero-order hold, 1-order hold and so forth).

a) Generate 10 points equally spaced along a sine curve in the interval [0,4*pi] using 'linspace' command.

b) Use 'polyfit' to fit a 0th-degree polynomial to the points. Then, evaluate the polynomial on a finer grid (using 'polyval' command) and plot the results along with the data points in part a).

c) Use 'polyfit' to fit a 1st-degree polynomial to the points. Then, evaluate the polynomial on a finer grid (using 'polyval' command) and plot the results along with the data points in part a).

d) Use 'polyfit' to fit a 2nd-degree polynomial to the points. Then, evaluate the polynomial on a finer grid (using 'polyval' command) and plot the results along with the data points in part a).

e) Use 'polyfit' to fit a 7th-degree polynomial to the points. Then, evaluate the polynomial on a finer grid (using 'polyval' command) and plot the results along with the data points in part a).

5) (30 points) MATLAB Practice: In this problem you will learn how to work with the samples of a sound signal, interpolate for the missing data, and generate an approximated version of the original signal.

a) The signal 'Signal_undersampled.mat' is given. The file can be found on the same directory as Project 1 on Canvas. Load the data in MATLAB.
b) Play the sound using 'sound' command. Consider the sampling frequency of Fs = 8192 Hz.
c) Plot the samples of the signal.
d) Use 'polyfit' command and come up with a reasonable order of polynomial that you think will be good fit to the samples of the signal.
e) Use 'polyval' command and evaluate a point between each of the two successive samples of the signal. Apply it to all the two successive samples of the original sampled signal.
f) Generate the new signal which has now twice samples than the original signal. Also, plot the new signal.
g) Use 'sound' command and play the new signal using the same sampling frequency Fs.

Good Luck