ECE 4750: Digital Signal Processing

Project 1

Angel Rodriguez

Utah Valley University

# Introduction

The objective of this lab is to learn how to ‘polyval’ and ‘polyfit’ commands in MATLAB to make a regenerated output signal with double the samples of the input signal. In this lab, echo generation is also covered with a modeling equation of x[n] which is to be defined. In the process of recreating the input signal (Figure 5), there was a trade-off situation that was brought to attention. To accomplish the regeneration of the signal with double samples, the input signal needed to be split into chunks to apply ‘polyfit’ and ‘polyval’ to. In order for the ‘polyfit’ function assignment p = polyfit(x,y,n) to work properly, ‘n’, the degree of the polynomial, must be greater than the length of ‘x’ and ‘y’ (‘x’ and ‘y’ must be equal in length or MATLAB will generate an error). This means that using too small chunks with a polynomial degree equal or greater than the chunk sizes would be an issue. This issue occurs when MATLAB generates the warning, “*Warning: Polynomial is not unique; degree >= number of data points*” demonstrating the need to increase datapoints (length of ‘x’ and ‘y’) or decrease the polynomial degree (‘n’). Demonstrations shown in Figures 1-4 (provided with descriptions) show the effectiveness of a greater polynomial degree but splitting the signal into smaller chunks would also be effective since it is easier/efficient to apply a curve of fit to a section instead of the entire signal. Achieving adequate results requires find the best balance between the chunk sizes and the polynomial degree. Results are shown below and in the attached file.

# Tasks

**1) Explain what ‘polyval’ and ‘polyfit’ commands do in MATLAB.**y = polyval(p,x) evaluates the polynomial p at each point in x. Argument p is a vector of length n+1 whose elements are the coefficients (in descending powers) of an nth-degree polynomial. This command in addition to ‘polyfit’ may be used to manipulate/regenerate an output signal.  
p = polyfit(x,y,n) returns the coefficients for a polynomial p(x) of degree n that is a best fit (in a least-squares sense) for the data in y. The coefficients in p are in descending powers, and the length of p is n+1. This can be used to generate fitted curves of a signal.

**2) Explain what ‘filter’ command do in MATLAB and what the input arguments can be.**[y](https://www.mathworks.com/help/matlab/ref/filter.html#bt_vs4t-1-y) = filter([b](https://www.mathworks.com/help/matlab/ref/filter.html#bt_vs4t-1-b),[a](https://www.mathworks.com/help/matlab/ref/filter.html#bt_vs4t-1-a),[x](https://www.mathworks.com/help/matlab/ref/filter.html#bt_vs4t-1-x)) filters the input data x using a [rational transfer function](https://www.mathworks.com/help/matlab/ref/filter.html#buagwwg-2) defined by the numerator and denominator coefficients b and a. Here is an example in MATLAB of how filter can be used given H(e^jw) and input signal x[n] to find y[n]:  
n = 0:0.1:10\*pi;  
x = 10 + 4\*cos((pi/4).\*n + pi/8) + 3\*cos((pi/3).\*n - pi/4);  
b = 1;  
% H(e^jw) = 1 - e^-jw + e^-j2w  
% h[n] = delta[n] - delta[n-1] + delta[n-2]   
a = [1 -1 1];  
% y[n] = ?  
y = filter(a, b, x);

**3)** **Then, listen to the final sound ‘x’ and explain what you learned from this problem.**  
I have learned that echoes created from an original sounds y[n] can be modeled as ay[n-D] where ‘D’ is the delay from the original signal to the echo (the echo appears D samples later) and ‘a’ is the attenuation of the direct sound (y[n]). This difference equation of the sum of the direct sound and echo is modeled as 𝑥[𝑛] = 𝑦[𝑛] + 𝛼𝑦[𝑛 − 𝐷], |𝛼| < 1. The code for the echo generation can be found under the ‘Attachments’ section.

**4) See attached MATLAB file and Figures 1-4.**

**5) See attached MATLAB file and Figures 5-6.**

# Figures

Chart, scatter chart

Description automatically generated  
**Figure 1:**  
Used ‘polyfit’ to fit a 0th-degree polynomial to the points. Then, evaluated the polynomial on a finer grid (using ‘polyval’ command) and plotted the results along with the data points in part a). It is observed that the result does not come close to being an adequate fitted line.

Chart, scatter chart

Description automatically generated  
**Figure 2:**  
Used ‘polyfit’ to fit a 1st-degree polynomial to the points. Then, evaluated the polynomial on a finer grid (using ‘polyval’ command) and plotted the results along with the data points in part a). It is observed that the result does not come close to being an adequate fitted line, but it is a change from the 0th polynomial results.

Chart, scatter chart

Description automatically generated  
**Figure 3:**  
Used ‘polyfit’ to fit a 2nd-degree polynomial to the points. Then, evaluated the polynomial on a finer grid (using ‘polyval’ command) and plotted the results along with the data points in part a). It is observed that the result does not come close to being an adequate fitted line and is about identical to the 1st degree polynomial results.

Chart, line chart

Description automatically generated  
**Figure 4:**  
Used ‘polyfit’ to fit a 7th-degree polynomial to the points. Then, evaluated the polynomial on a finer grid (using ‘polyval’ command) and plotted the results along with the data points in part a). It is observed that the result is an adequate fitted line. This experiment demonstrates better results with higher polynomial degrees.

Chart, histogram

Description automatically generated

**Figure 5:**  
The signal from MATLAB data file ‘Signal\_undersampled.mat’

Chart, line chart

Description automatically generated

**Figure 6:**  
Generated the output signal which has twice as many samples than the original signal (the length is double the input signal). Looks very similar to Figure 6 with an increase amplitude in both the positive and negative directions.

# Conclusion

The result of the regenerated signal with double samples (Figure 5) has an amplitude greater that the input signal (Figure 4) but sounds clearer than the input signal. While the input signal sounds high-pitched and coherent words cannot be made from it, the sound of the regenerated signal appears low-pitched, slower, and it is clear that “*Halleluiah*” is being said multiple times. Even though words can be made from the output signal, it still has unnecessary noise preventing it from sounding perfectly clear. It has been observed that the output signal becomes clearer the smaller the chunks used are, but the chunks cannot be greater than or equal to the polynomial degree. It is concluded based on the results of the plotted signals and the sound of the output signal that the combination which made the output signal most clear was smaller chunks with a polynomial degree of 1.

# Attachments

**MATLAB Code of ‘Project\_1.mlx’:**

% 3)

clear, clc, close all;

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

% 4)

clear, clc, close all;

x = linspace(0, 4\*pi, 10);

y = sin(x);

x1 = linspace(0,4\*pi);

% plot(x, y);

p1 = polyfit(x,y,0);

y1 = polyval(p1,x1);

plot(x,y,'o')

hold on

plot(x1,y1)

title('0th-degree polynomial')

hold off

p2 = polyfit(x,y,1);

y2 = polyval(p2,x1);

plot(x,y,'o')

hold on

plot(x1,y2)

title('1st-degree polynomial')

hold off

p3 = polyfit(x,y,2);

y3 = polyval(p3,x1);

plot(x,y,'o')

hold on

plot(x1,y3)

title('2nd-degree polynomial')

hold off

p4 = polyfit(x,y,7);

y4 = polyval(p4,x1);

plot(x,y,'o')

hold on

plot(x1,y4)

title('7th-degree polynomial')

hold off

% 5)

clear, clc, close all;

load Signal\_undersampled.mat

a = Signal\_undersampled;

Fs = 8192;

chunksize = floor(length(a)/18000);

% initalize y

y = [];

for index = 1:(floor( length(a) / chunksize)) - 1

p = polyfit(1:chunksize, a(index\*chunksize:(index\*chunksize)+chunksize-1), 1);

% Add new values to y without erasing existing values

y = [y, polyval(p, 1:chunksize\*2)];

end

plot(a); title('Original Signal')

plot(y); title('Regenerated Signal with Double Samples')

% This commented out code adjusts the amplitude of regenerated signal to

% better match the original signal

% Coefficient = max(a)/max(y);

% y = y\*Coefficient;

% plot(y); title('Regenerated Signal with Double Samples and Adjusted Amplitude')

sound(a,Fs);pause(5);

sound(y,Fs);

% Save the file

% audiowrite('Regenerated Signal.wav',y,Fs)