**Final Project**

**Voice Input and Filtering in Matlab**

**ENGE 330**

**Signals and Networks**

**George Fox University**

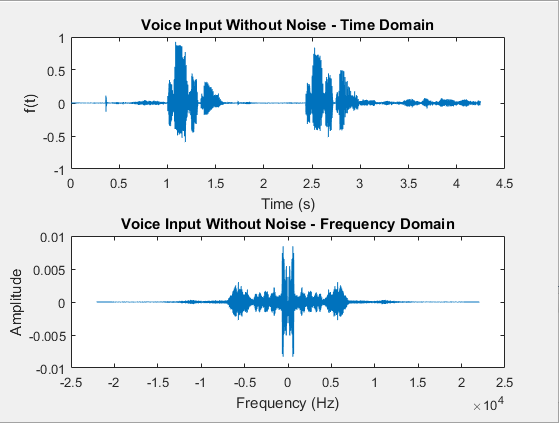
**Newberg, OR**

**December 9, 2016**

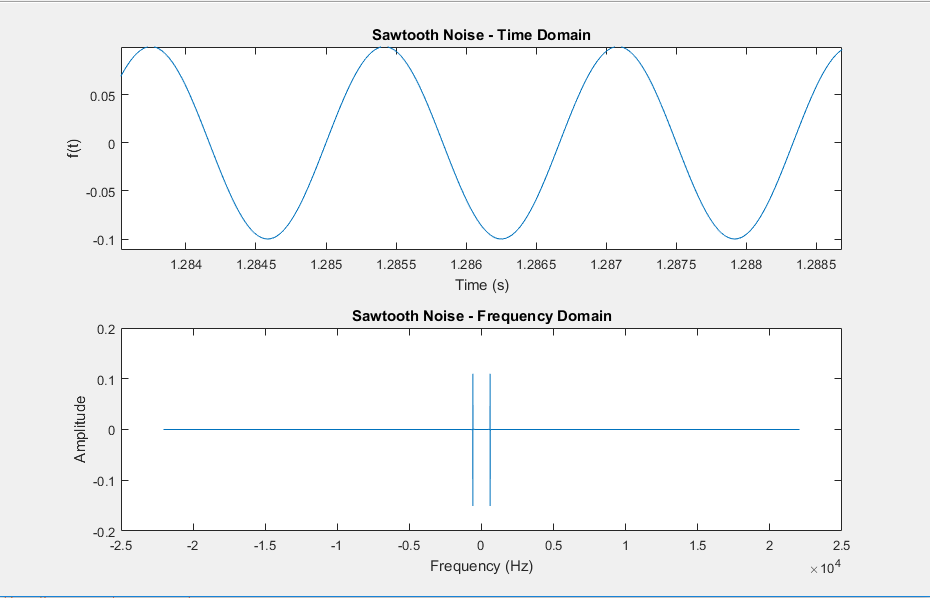
**Zach Wilson**

**Project Description:**

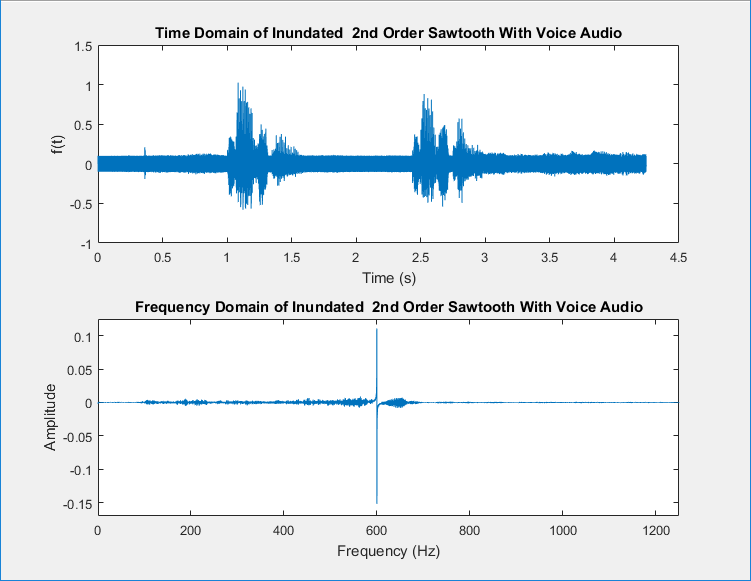
The goal of this project was to create a .wav voice file and then overlay a sawtooth of varying harmonics which were selected by the user. The program would then create an nth order notch filter to process out all the corresponding harmonics of the sawtooth and produce a clean voice signal that matches the original.



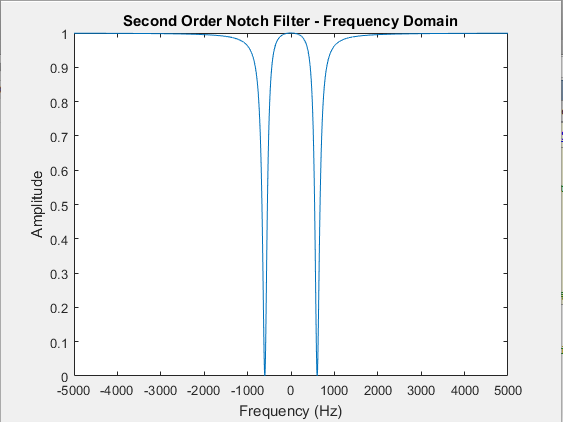
**Fig 1.**  This is the pure .wav voice input.



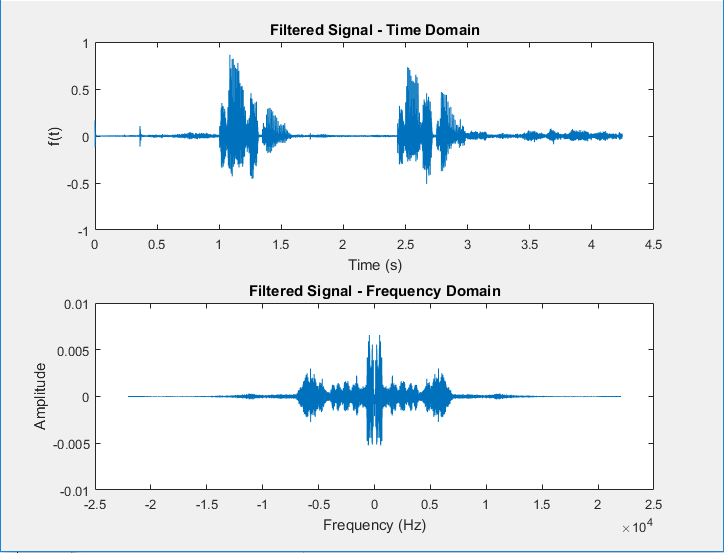
**Fig 2.**  This is the sine wave at 600 Hz used in the initial parts of the project.



**Fig 3.**  This is the inundated voice signal.

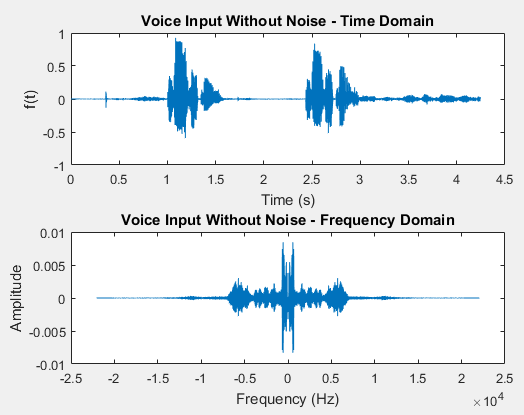


**Fig 4.**  This is the 2nd order notch filter.

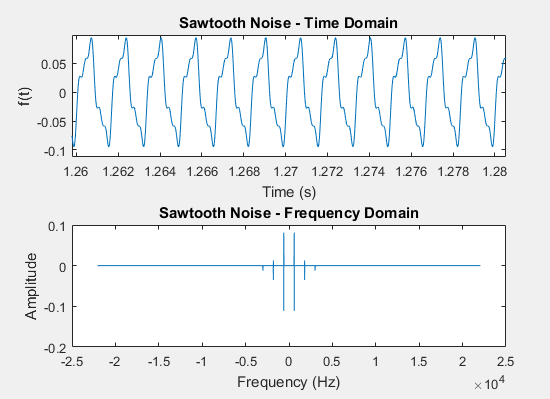


**Fig 5.**  This is the filtered output.

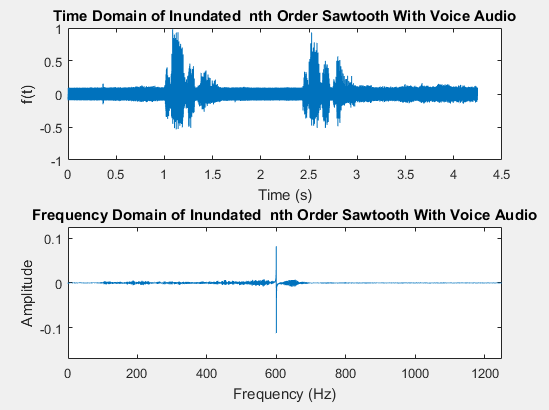
This ends the first part of the project.



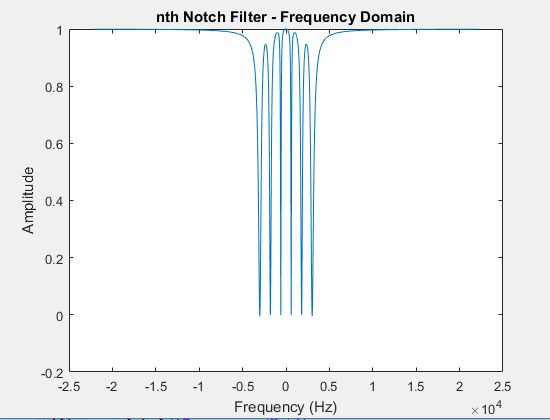
**Fig 6.**  This is the pure .wav input of the voice signal.



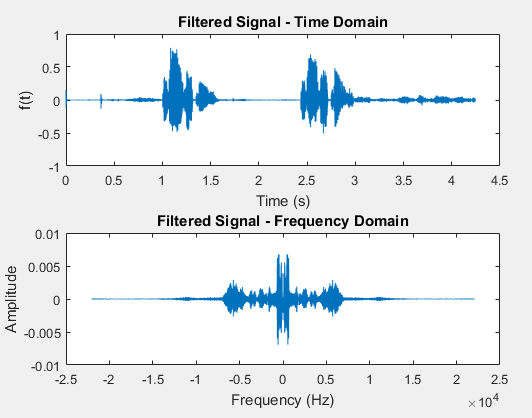
**Fig 5.**  This is the nth order sawtooth function.



**Fig 7.**  This is the inundated voice signal.



**Fig 8.**  This is the nth order notch filter.



**Fig 9.**  This is the filtered output.

**Description:**

There is some variance in the filtering abilities of the filter depending on how many harmonics are picked and how wide the user makes the bandwidth. There is a medium ground of around roughly 300 Hz for the bandwidth and around 5 or more harmonics in order to produce a clean signal. The good part about this is that you don’t need a ton of components to produce a quality filtering effect considering that around 5 harmonics works well. If you go in to a greater number of harmonics, there is a problem because you have to physically build a circuit of that many cascaded passive notch filters. Hardware implementation is always something to be considered when building filtering systems, and thankfully the filter I’ve built yields a potentially compact design.

**This is the Matlab code used in the project**

% Zach Wilson

% ENGE 330

% Final Project

% 12/3/16

clear;

%%%%%%%%%%%%%%%%%%%%% Part A %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%

[a,Fs] = audioread('testing.wav'); %Import the .wav file

N = length(a);

tmax = (N-1)/Fs;

t = linspace(0,tmax,N);

t = t'; %storing as column vector, to match x

% This FFT is for the Magnitude plot of a(t) -> A(f)

[AF, AF\_freq] = fft330(a, Fs);

% This was for part A

noise = sin(600\*2\*pi.\*t);

[NOISE, NOISE\_f] = fft330((0.1\*noise), Fs);

x = a + (0.1 .\* noise);

[X, f] = fft330(x, Fs);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

prompt = ('Enter the number of harmonics:');

n\_Max = inputdlg(prompt);

n\_Max = str2num(n\_Max{:})

%%% The Following is to set up the Fourier Series of 15.19 %%%

% These are the max values to set the parameters for the loops:

wo = 600 \* 2 \* pi; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Change this used to be fc

c = 0; % This will hold the sawtooth function

A = 0.1; % Magnitude, Prevents sawtooth from over powering my voice

for n = 1:2:n\_Max

c = c + ((((2\*A)/(pi\*n))\*sin(n\*wo\*t)) + (((-4\*A)/(pi^2\*n^2))\*cos(n\*wo\*t)));

end

[C, CF] = fft330(c, Fs);

c\_noise = c + a;

[C\_Noise, C\_f] = fft330(c\_noise, Fs);

%%%%%%%%%%%%%%%%%%%%% Part C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%

figure

plot(f, abs(C\_Noise));

title('Frequency Domain of Inundated nth Order Sawtooth With Voice Audio');

xlim([0, 1250]); % This defines the limits of the plot

% text\_1 = text(f + 20, abs(X)

% Set up mouse capture

[mouse\_x, mouse\_y] = ginput(2);

% These are used in Part B

fc = mouse\_x(1);

fh = mouse\_x(2);

fl = (fc^2)/fh;

bw = abs(fh-fl);

%bw = abs((abs(fh) - abs(fc)) \* 2);

% 1 means left button click, 2 is middle, 3 is right

% These set the values for the transfer function

w = 2\*pi.\*f;

wc = fc\*2\*pi;

s = j.\*w;

z = bw/(2\*fc); % zeta

order = n\_Max; % This is set by the user

Hn = ones(size(a)); % creates a vector of ones

for n = 1:2:order

fc\_new = fc \* n;

wc\_new = wc \* n; %wc is the cutoff frequency determined by the user

fl = ((fc\_new)^2)/(fh\*n);

bw\_new = abs(fh-fl);

z\_new = bw\_new/(2\*fc\_new); % zeta

H\_new = (s.^2 + wc\_new.^2) ./ (s.^2 + 2\*z\*wc\_new.\*s + wc\_new.^2);

Hn = Hn .\* H\_new;

end

Cleansed\_Signal = (Hn .\* (C\_Noise));

[cleansed\_signal, cleansed\_freq] = ifft330(Cleansed\_Signal, Fs); %time domain output

sound((cleansed\_signal), Fs);

%%%%%%%%%%%%%%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%

% These are the initial voice input in freq and time domain

figure

subplot(2, 1, 1);

plot(t, a);

title('Voice Input Without Noise - Time Domain');

xlabel('Time (s)');

ylabel('f(t)');

subplot(2, 1, 2);

plot(f, AF);

title('Voice Input Without Noise - Frequency Domain');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are the inundated signal in time and freq

figure

subplot(2, 1, 1);

plot(t, c\_noise);

title('Time Domain of Inundated nth Order Sawtooth With Voice Audio');

xlabel('Time (s)');

ylabel('f(t)');

subplot(2, 1, 2);

plot(f, C\_Noise);

title('Frequency Domain of Inundated nth Order Sawtooth With Voice Audio');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

xlim([0, 1250]); % This defines the limits of the plot

ylim([-0.17, 0.125]); % This defines the limits of the plot

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are the Plot B spectral graphs, After the voice has been filtered

figure

subplot(2, 1, 1);

plot(t, cleansed\_signal);

title('Filtered Signal - Time Domain');

xlabel('Time (s)');

ylabel('f(t)');

subplot(2, 1, 2);

plot(f, Cleansed\_Signal);

title('Filtered Signal - Frequency Domain');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are the Plot B Sawtooth graphs

figure

subplot(2, 1, 1);

plot(t, (c)); % noise has its amplitude reduced

title('Sawtooth Noise - Time Domain');

xlabel('Time (s)');

ylabel('f(t)');

subplot(2, 1, 2);

plot(f, C);

title('Sawtooth Noise - Frequency Domain');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These are the Plot B Transfer Function Plots

figure

plot(f, Hn);

title('nth Notch Filter - Frequency Domain');

xlabel('Frequency (Hz)');

ylabel('Amplitude');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%