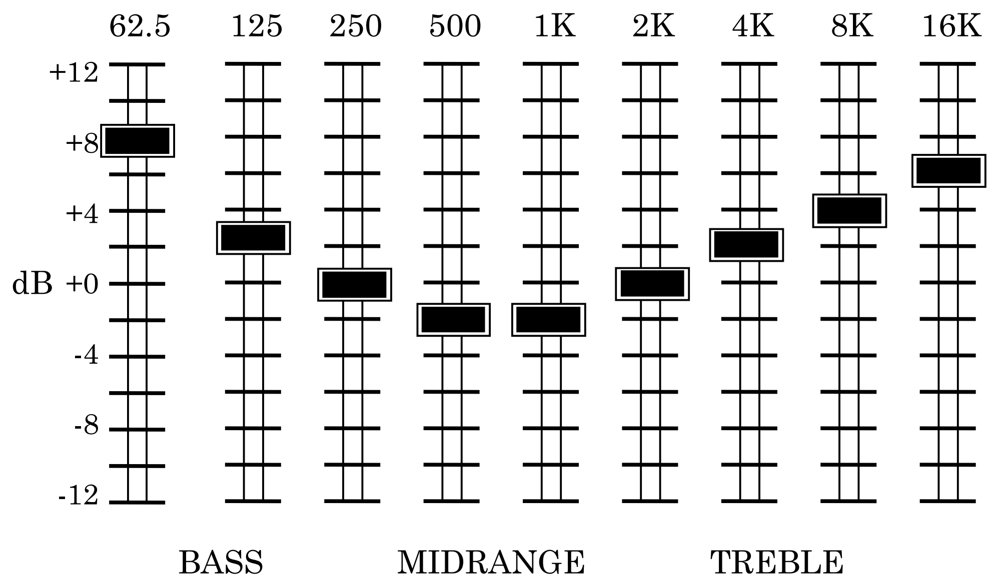
EE 419 - Project 11 - Fun Filtering Times (FFTs)

Names: Chris Adams & Aiku Shintani **Lab Date: 3/12/19**

Bench #: 9 **Section: 2**

1) Windowed Frequency Sampling FIR Graphic Equalizer Design

EQUALIZER SETTINGS



FIR Filter Design by Frequency Sampling method

Features:

- 9 EQ Bands – Logarithmically spaced center frequencies (as shown above)
- Tukey Window smoothing of magnitude response values
- Stereo processing (2 channels)
- Amplitude clipping prevention
- .wav file input and output

2) ADD AN ECHO EFFECT TO YOUR EQUALIZER

```
[echo_filter_hn]= echo_filter(Dk_delays_msec,alphak_gains,Fsample);
```

Features:

- Fast Convolution processing
- Acoustic impulse response with an arbitrary # of delay/gain combinations
- .wav file input and output

INTEGRATING YOUR TWO FUNCTIONS:

```
Processed_wav = equalize_and_reverb wavefile(inwavfilename, EQdBsettings,
        Dk_delays_msec, alphak_gains, outwavfilename);
```

TEST RESULTS:

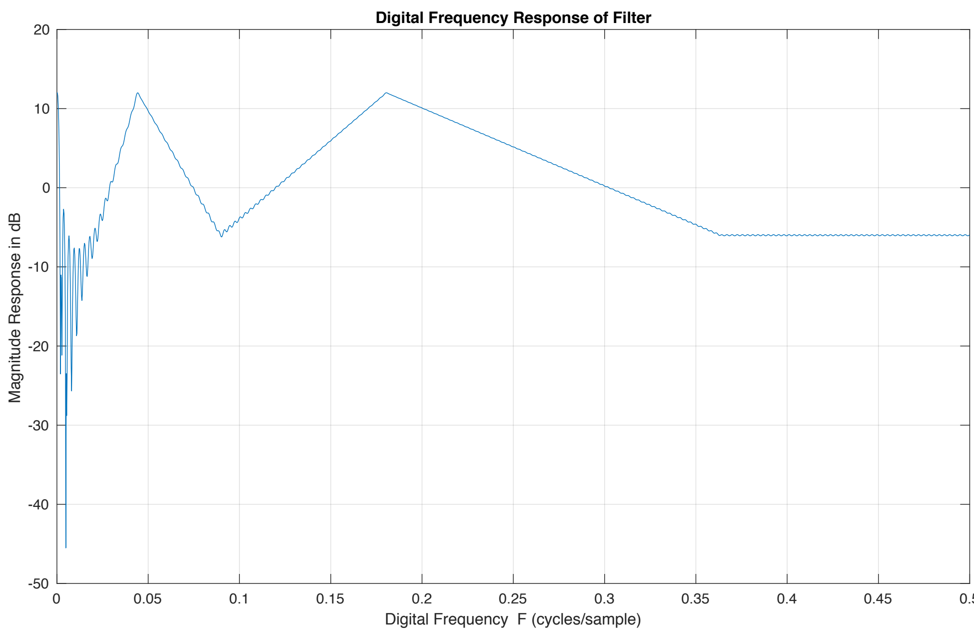
"Awful" Test Case:

Equalizer Frequencies (Hz):	**62.5**	**125**	**250**	**500**	**1000**	**2000**	**4000**	**8000**	**16000**
Equalizer Settings (dB):	**+12**	**-6**	**-12**	**-12**	**-6**	**+12**	**-6**	**+12**	**-6**

D_k Echo Delays (msec):	**250**	**400**	**520**	**660**	**750**	**1220**			
Fractional α_k Gains:	**0.7**	**0.6**	**0.5**	**0.33**	**0.2**	**0.8**			

Test Results:

"Awful" Test Case EQ Frequency Response Plot (dB magnitude response)



"Good" Test Case:

Equalizer Frequencies (Hz):	**62.5**	**125**	**250**	**500**	**1000**	**2000**	**4000**	**8000**	**16000**
Equalizer Settings (dB):	**-1**	**3**	**2**	**1**	**-1**	**-1**	**-2**	**-3**	**-2**

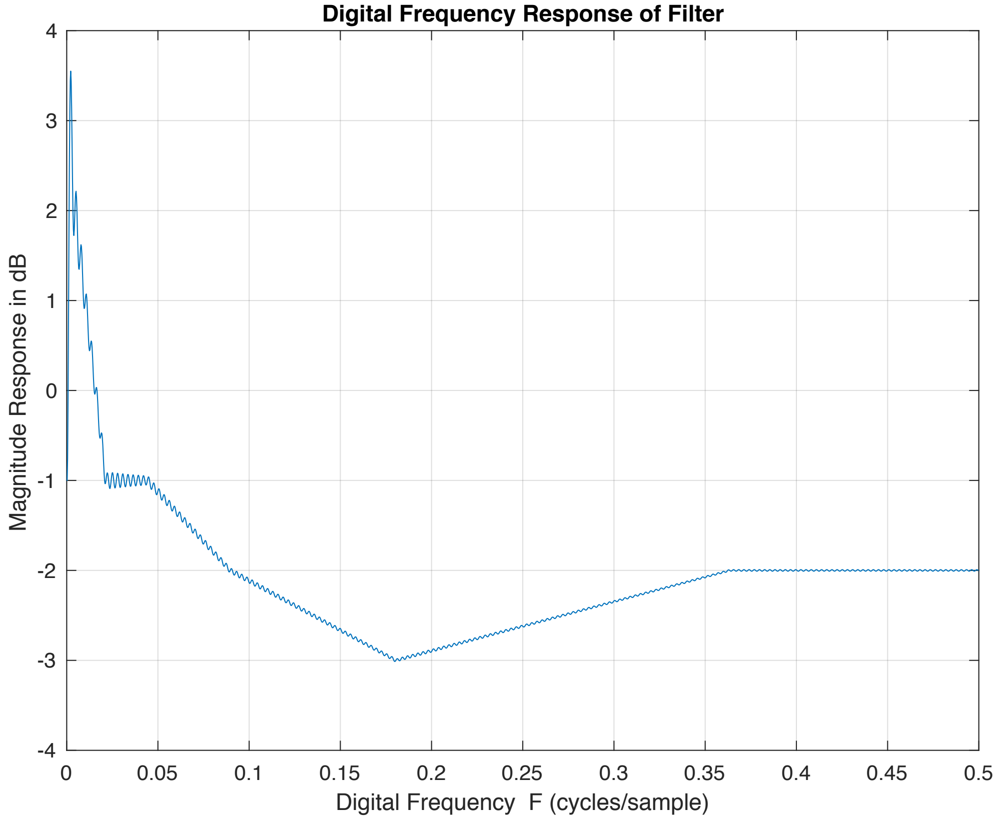
D_k Echo Delays (msec):	**50**	**75**	**133**	**210**	**300**	**500**			
Fractional α_k Gains:	**0.8**	**0.75**	**0.6**	**0.33**	**0.2**	**0.1**			

Rationale for Settings Selection: (Describe why you chose these settings – What were you trying to accomplish")

When listening, it was noticed that certain frequencies were much louder than others. This is due to the jumping of values in the equalizer settings. To combat that, the values were largely reduced to get a smoother frequency transition. Furthermore, it was clear that the large delays created quite a bit of noise. Therefore, the delays and gains were toned down a bit.

Test Results:

"Good" Test Case EQ Frequency Response Plot (dB magnitude response)



Matlab M-File:

```
function process_WAV(set_equalizer,Dk_delays_msec,alphak_gains,fs,input_wav,new_wav_filename)
%

% Frequency sampling and Tukey Windowing (up to line 119ish)
M = 707;
Fc1 = 62.5/fs; Fc2 = 125/fs; Fc3 = 250/fs;
Fc4 = 500/fs; Fc5 = 1e3/fs; Fc6 = 2e3/fs;
Fc7 = 4e3/fs; Fc8 = 8e3/fs; Fc9 = 16e3/fs;
Fc = [Fc1, Fc2, Fc3, Fc4, Fc5, Fc6, Fc7, Fc8, Fc9];

%calculate interpolation input parameters
for d=1:length(Fc)-1
    interp_x{d} = [round(Fc(d)*M),round(Fc(d+1)*M)];
    interp_xq{d} = round(Fc(d)*M):round(Fc(d+1)*M);
end

i = 1;
k = 1; %index through the equalizer settings
       %set_equalizer() is the input array of dB values at specified freqs.

for i=1:(M-1)/2

    %LP below Fc1
    if (i <= Fc1*M)
         HF_mag_samples_dB(i) = set_equalizer(k);

    %interpolate points b/w Fc1 and Fc2
```

```
elseif (i > round(Fc1*M) && i <= round(Fc2*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{1},interp_mag,interp_xq{1});
    HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    if i == round(Fc2*M)
        k = k +1;
    end

%interpolate points b/w Fc2 and Fc3
elseif (i > round(Fc2*M) && i <= round(Fc3*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{2},interp_mag,interp_xq{2});
    if i == round(Fc3*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc3 and Fc4
elseif (i > round(Fc3*M) && i <= round(Fc4*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{3},interp_mag,interp_xq{3});
    if i == round(Fc4*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc4 and Fc5
elseif (i > round(Fc4*M) && i <= round(Fc5*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{4},interp_mag,interp_xq{4});
    if i == round(Fc5*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc5 and Fc6
elseif (i > round(Fc5*M) && i <= round(Fc6*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{5},interp_mag,interp_xq{5});
    if i == round(Fc6*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc6 and Fc7
elseif (i > round(Fc6*M) && i <= round(Fc7*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{6},interp_mag,interp_xq{6});
    if i == round(Fc7*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc7 and Fc8
elseif (i > Fc7*M && i <= Fc8*M)
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{7},interp_mag,interp_xq{7});
    if i == round(Fc8*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end

%interpolate points b/w Fc8 and Fc9
elseif (i > round(Fc8*M) && i <= round(Fc9*M))
    interp_mag = [set_equalizer(k),set_equalizer(k+1)];
    interp = interp1(interp_x{8},interp_mag,interp_xq{8});
    if i == round(Fc9*M)
        k = k +1;
        HF_mag_samples_dB = [HF_mag_samples_dB,interp(2:end)];
    end
% @Fc1,Fc2,Fc3,Fc4,Fc5,Fc6,Fc7,Fc8,Fc9 digital frequencies
else
```

```matlab
        HF_mag_samples_dB(i) = set_equalizer(k);

    end

end

%make the HF magnitude samples symmetrical
HF_mag_samples_dB_right = fliplr(HF_mag_samples_dB);
HF_mag_samples_dB_right = HF_mag_samples_dB_right(1:length(HF_mag_samples_dB_right)-1);
HF_mag_samples_dB = [HF_mag_samples_dB, HF_mag_samples_dB_right];

%convert from dB to linear magnitude
HF_mag_samples_lin = 10.^(HF_mag_samples_dB/20);

%get unit sample response, Bk's
[freq_samp_hn, HF, F] = FIR_Filter_By_Freq_Sample(HF_mag_samples_lin,1);

%window the frequency response acquired from frequency sampling in order to
%reduce "ringing and ripple." Tukey was chosen so that there is little
%impact on the transition width and frequency resolution
equalization_filter_hn = freq_samp_hn.*(tukeywin(length(freq_samp_hn)).'); %make sure to
transpose window


%produce echo filter using echo_filter_hn function
echo_filter_hn = echo_filter(Dk_delays_msec,alphak_gains,fs);

%combine the equalization filter resp. with echo filter resp.
best_filter = fftconv(equalization_filter_hn, echo_filter_hn);
freqz(best_filter, 1, 2^20)

% Load Audio Signal
[audio_in,fs] = audioread(input_wav);

% Check if 2 Channel Input
if size(audio_in,2) == 2    % size(x, 2) returns # of cols. in x
    stereo = 1;
    audio_in = audio_in.';  %transpose for processing
    Ch1 = audio_in(1,:);
    Ch2 = audio_in(2,:);
% Check if 1 Channel Input
else
    stereo = 0;
    audio_in = audio_in.';
end

% Filter Audio Input with the best filter
if stereo == 1
    Ch1_processed = fftconv(Ch1,best_filter);
    Ch2_processed = fftconv(Ch2,best_filter);
    Processed_audio(1,:) = Ch1_processed/max(abs(Ch1_processed));
    Processed_audio(2,:) = Ch2_processed/max(abs(Ch2_processed));
    Processed_audio = Processed_audio.'; %re-transpose after processing
else
    Ch1_processed = fftconv(audio_in,best_filter);
    Processed_audio(1,:) = Ch1_processed/max(abs(Ch1_processed));
    Processed_audio = Processed_audio.'; %re-transpose after processing
end

audiowrite(new_wav_filename,Processed_audio,fs);

end


% Produce Echo Filter
function echo_filter_hn = echo_filter(Dk_delays_msec,alphak_gains,fs)
%

%convert delay to seconds
Dk_delays_sec = Dk_delays_msec/1e3;
n = Dk_delays_sec.*fs; %n = t/Ts = t*fs
```

```
for n = 1:length(n)
    Bk(n) = alphak_gains(n);
end

echo_filter_hn = Bk;

end

function [hn, HF, F] = FIR_Filter_By_Freq_Sample(HF_mag_samples, figure_num)
%This function takes two input arugments: HF_mag_samples which correspond
%to the H_k magnitude (positive only and for a DC gain of <1 all
%coefficients are <1) response that the user want and the figure #. The
%function returns the corresponding unit sample response (h[n]), the
%frequency response (HF), and the digital freq (F).

% part a
k = 0:length(HF_mag_samples)-1; %# of mag samples is M
M = length(k);                  % M is equal to # k samples for DFT

angle_rad = -pi.*k*(M-1)/M;     %compute angle argument (in radians)

%correct angles arguments so they're within -pi and pi
for x=1:length(angle_rad)

    while (angle_rad(x) < -pi)
        angle_rad(x) = angle_rad(x)+2*pi;
    end

    while (angle_rad(x) > pi)
        angle_rad(x) = angle_rad(x)-2*pi;
    end
end

Hk_angle = exp(1j*angle_rad);    %Hk_angles
Hk = HF_mag_samples.*Hk_angle;  %Hk in complex form
hn = real(ifft(Hk));            %get the unit sample response

% part b
HF_no_pad = fft(hn); %for use in low-res FFT ==> DFT, discrete
                     %compute non-padded HF

M_pad = 2^12;        %for use in high-res FFT ==> "DTFT", psuedo continuous
HF = fft(hn, M_pad); %compute padded HF
F = 0:1/(M_pad-1):1; %sample freq. spacing
Fk = 0:1/M:(M-1)/M;  %for stem plots

HF_mag = abs(HF);                  %compute the magnitude of padded HF
HF_mag_no_pad = abs(HF_no_pad); %compute the magnitude of non-padded HF
HF_ang = angle(HF)/pi;          %compute the angle of padded HF
HF_ang_no_pad = angle_rad/pi;   %compute the angle of non-padded HF

%plot digital frequency response
figure(figure_num)
subplot(2,1,1)
plot(F, HF_mag) %plot magnitude response (linear)
xlabel('Digital Frequency  F (cycles/sample)')
ylabel('Magnitude Response')
title('Digital Frequency Response of Filter')

hold on

%superimpose non-padded DFT magnitude
stem(Fk, HF_mag_no_pad, '.', 'MarkerSize', 20, 'Linewidth', 2);

%plot phase response
subplot(2,1,2)
plot(F, HF_ang)
xlabel('Digital Frequency  F (cycles/sample)')
ylabel('Phase Response/pi')

hold on
```

```
%superimpose non-padded DFT phase
stem(Fk, HF_ang_no_pad, '.', 'MarkerSize', 20, 'Linewidth', 2);

%part c, plot magnitude response (in dB this time)
F_c = 0:1/(M_pad-1):0.5;                    %F = 0-0.5 instead of F =0-1 like before
figure(figure_num + 1)
plot(F_c, 20*log10(HF(0:0.5*(M_pad - 1)))) %only take up to F = 0.5 worth of HF
xlabel('Digital Frequency  F (cycles/sample)')
ylabel('Magnitude Response in dB')
title('Digital Frequency Response of Filter')

end

function yn = fftconv(xn, hn)
% This function takes two input arguments: the xn coefficients and unit
% sample response. The function computes the FFT and returns one figure
% with 6 subplots: stem plots of x[n], h[n], and y[n] and plots of the
% magnitude responses for them.

Mh = length(hn);       %compute lengths
Mx = length(xn);
M = Mh + Mx - 1;       %number of samples for fft is sum of lenghts - 1
n = 2^(nextpow2(M));   %for most efficient fft computation

k_x = [0 :1 : Mx - 1]; %for plotting x[n], x-axis
k_h = [0: 1: Mh - 1];  %for plotting h[n], x-axis
k_y = [0: 1: M - 1];   %for plotting y[n], x-axis

Xk = fft(xn, n);       %fft of x[n]
Hk = fft(hn, n);       %fft of h[n]
Yk = Xk.*Hk;           %convolution in time domain is multiplication in freq.

yn = real(ifft(Yk));   %get good parts of fft of y[n] after doing the inv fft

if (Mh > 1000) || (Mx > 1000) %check to see if excessive # of points to plot
    produce_plots = 0;
else
    %Generate plots
    figure(1)

    %make stem plot of input
    subplot(3, 2, 1)
    stem(k_x, xn, '.', 'MarkerSize', 20, 'Linewidth', 2);
    xlabel('Sample index')
    ylabel('Amplitude')
    title('x[n] sequence')

    %make stem plot of unit sample response
    subplot(3, 2, 3)
    stem(k_h, hn, '.', 'MarkerSize', 20, 'Linewidth', 2);
    xlabel('Sample index')
    ylabel('Amplitude')
    title('h[n] sequence')

    %make stem plot of output
    subplot(3, 2, 5)
    stem(0:M-1, yn(1:M), '.', 'MarkerSize', 20, 'Linewidth', 2);
    xlabel('Sample index')
    ylabel('Amplitude')
    title('h[n] sequence')

    %plot magnitude responses
    %plot Xk_mag
    Fd = (0:(length(Yk)-1))/length(Yk); %compute the sampled digital F, x-axis
    Xk_mag = abs(Xk);

    subplot(3, 2, 2)
    plot(Fd, Xk_mag)
    xlabel('Sample index')
    ylabel('Magnitude Response')
    title('X[k] Spectrum')
```

```
    %plot Hk_mag
    Hk_mag = abs(Hk);

    subplot(3, 2, 4)
    plot(Fd, Hk_mag)
    xlabel('Sample index')
    ylabel('Magnitude Response')
    title('H[k] Spectrum')

    %plot Yk_mag
    Yk_mag = abs(Yk);

    subplot(3, 2, 6)
    plot(Fd, Yk_mag)
    xlabel('Sample index')
    ylabel('Magnitude Response')
    title('Y[k] Spectrum')

end
end
```

Feedback on the EE 459 Lab Projects:

Indicate which one of this quarter's lab projects was the "Most Helpful/Interesting" and why; and which was the "Least Helpful/Interesting" and why?

Proj 2: Developing a Filter Analysis Program in Matlab
Proj 3: DFT/FFT Signal Processing – Fast Convolution
Proj 4: Block Processing and FFT Spectrum Analysis – Touchtone Decoder
Proj 5: Correlation Detection – Hunt for Red October
Proj 6: Efficient IIR Implementations and Quantization Effects
Proj 7: FIR Filter Design Comparison
Proj 10: IIR Filter Design Comparison
Proj 11: Audio Signal Processing

Most Helpful / Interesting Project: Project #5

Why?: Project #5 provided the most intuition as to how digital signal processing can be used to perform a critical task. Relative to the other projects, the procedure for this one was more understandable. One step led to the next and each step provided a method of checking whether or not the project was progressing in the correct direction. The task at hand, detecting objects in the water, was the most appealing out of all projects.

Least Helpful / Interesting Project: Project #10

Why?: Project #10 did not provide much intuition as to how IIR filter design is implemented via Bilinear transform of an analog filter. Much of the concepts of the Bilinear transform are hidden by the Matlab functions which perform them for you. But, this project was successful in conveying the different advantages and disadvantages of each of the IIR digital filters.

Estimate the number of hours you each typically spent each week on this lab class outside of the lab (not including the 3 scheduled hours in the lab):

Name: Aiku Shintani Hours: 6
Name: Chris Adams Hours: 5

Other suggestions for improving this lab in the future: (include at least one per person)

1. **This project could be more beneficial is if there were more test cases to verify functionality. Sometimes it's difficult to tell based on sound so more tests would be helpful.**
   -Aiku
2. **This project could be improved by having students choose their own audio file in addition to the provided wave file. I think students would be more interested in using an equalizer on some music that they're more interested in.**
   -Chris