Lab 8 Digital Filter Design I

Elena Montalvo
Data collected with Darren Ray
ECE3101L - Andrew Pagnon
November 3, 2022

Objective

The student will be able to design digital FIR & IIR filters by placing poles & zeros at appropriate locations in the Z-Domain unity circle.
The student will be able to write simple programs to implement digital filters and filter audio signals.

Summary

Digital filter

Digital filters are used in a variety of applications. One of the applications it could be used is digital filters can be used in audio systems that allow the listener to adjust the bass and the treble of audio signals also known a as the low and high frequencies. Digital filter design requires the use of both frequency domain and time domain techniques. Note digitall filter design is usually un the frequency domain however to actually implemented is done in time domain Typically, frequency domain analysis is done using the Z-transform and the discrete-time Fourier Transform (DTFT).
In general, a linear and time-invariant causal digital filter with input *x*(*n*) and output *y*(*n*) may be specified by its difference equation

$$y(n) = \sum_{i=0}^{N-1} b_i x(n - i) - \sum_{k=1}^{M} a_k y(n - k)$$

here are two general classes of digital filters: infinite impulse response (IIR) and finite impulse response (FIR)

Infinite impulse response (IIR) is a property applying to many linear time-invariant systems that are distinguished by having an impulse response which does not become exactly zero past a certain point, but continues indefinitely. While finite impulse response (FIR) system in which the impulse response does become exactly zero

The DTFT may be thought of as a special case of the Z-transform where *z* is evaluated on the unit circle in the complex plane.

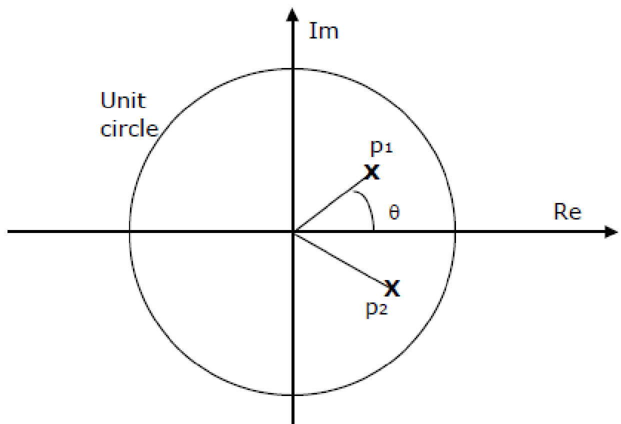
$$X\left(e^{jw}\right) = X(z)|_{z=e^{jw}} = \sum_{i=-\infty}^{\infty} x(n)e^{-jwn}$$

From the definition of the Z-transform, a change of variable *m* = *n* − *k* shows that a delay of *k* samples in the time domain is equivalent to multiplication by z^{-k} in the Z-transform domain.

$$x(n - k) \leftrightarrow z^{-k}X(z)$$

IIR

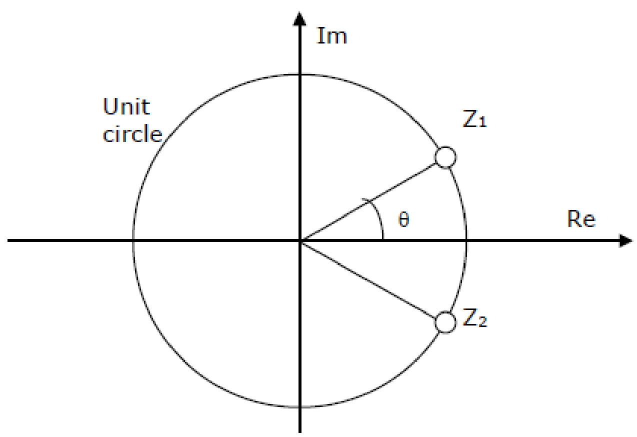
Infinite impulse response (IIR) is a property applying to many linear time-invariant systems that are distinguished by having an impulse response which does not become exactly zero past a certain point, but continues indefinitely.



We will be looking into polar coordinates this coming week

FIR

finite impulse response (FIR) system in which the impulse response does become exactly zero



Results

Prelab 8

Low pass

$$H(z) = b_0 \frac{z+1}{z-.95} = 1$$

Let $z = 1$

therefore $b_0 = 0.025$

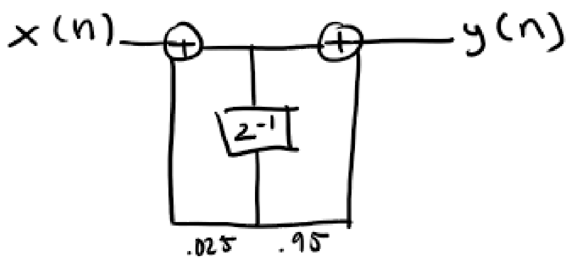
$$\frac{Y(z)}{X(z)} = 0.025 \frac{(z+1)}{z-.95}$$

$$y(z)(z-.95) = x(z)(.025)(z+1)$$

by inverse z-transfer

$$y(n) - .95y(n-1) = .025x(n) + .025x(n-1)$$

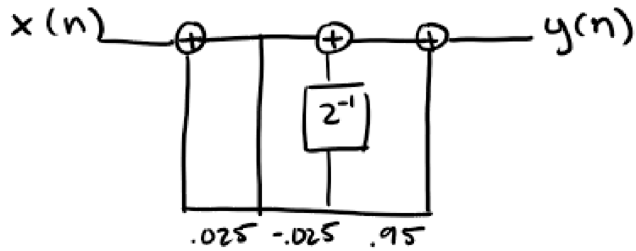
$$y(n) = .025x(n) - .025x(n-1) + .95y(n-1)$$



High pass

$$H(z) = .025 \frac{z-1}{z+.95}$$

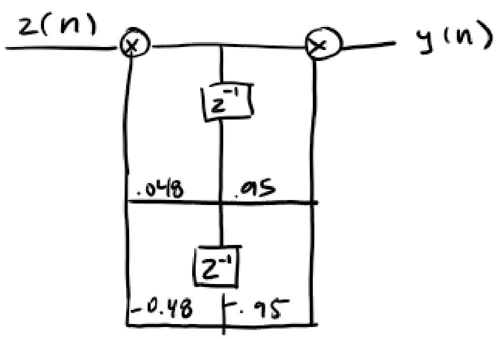
$$y(n) = .025\,x(n) - .25\,x(n-1) - .95\,y(n-1)$$



Band pass

$$H(z) = .04875 \frac{(z+1)(z-1)}{(z-0.95i)(z+.95i)}$$

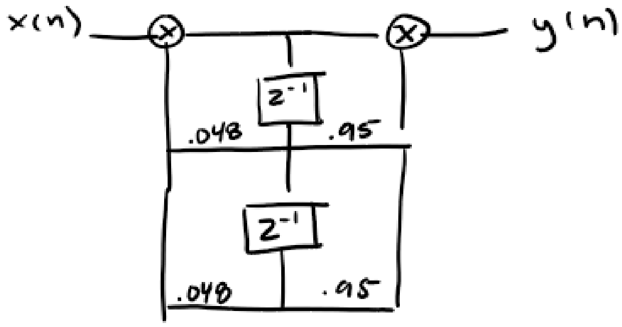
$$y(n) = .04875\,x(n) - .4875\,x(n-2) - .95\,y(n-2)$$



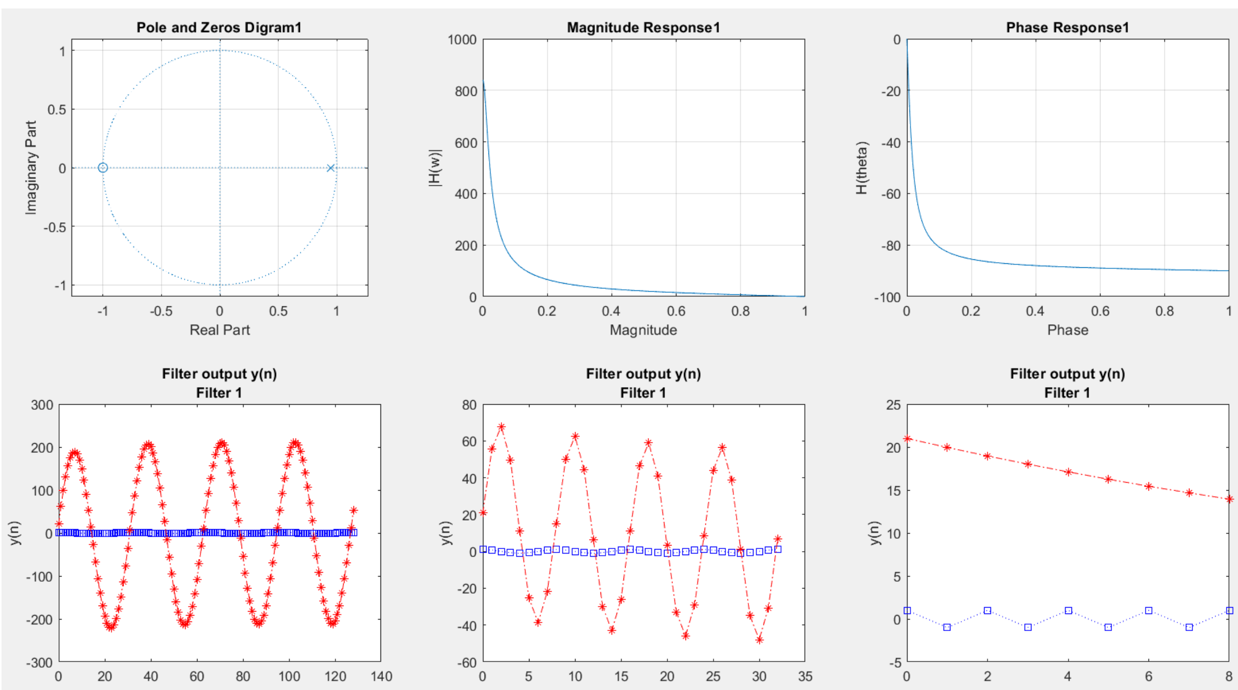
Band Stop filter

$$H(z) = .04875\,\frac{(z+i)(z-i)}{(z-.95)(z+.95)}$$

$$y(n) = .04875\,x(n) + .04875\,x(n-2) + .95\,(y(n-2)$$



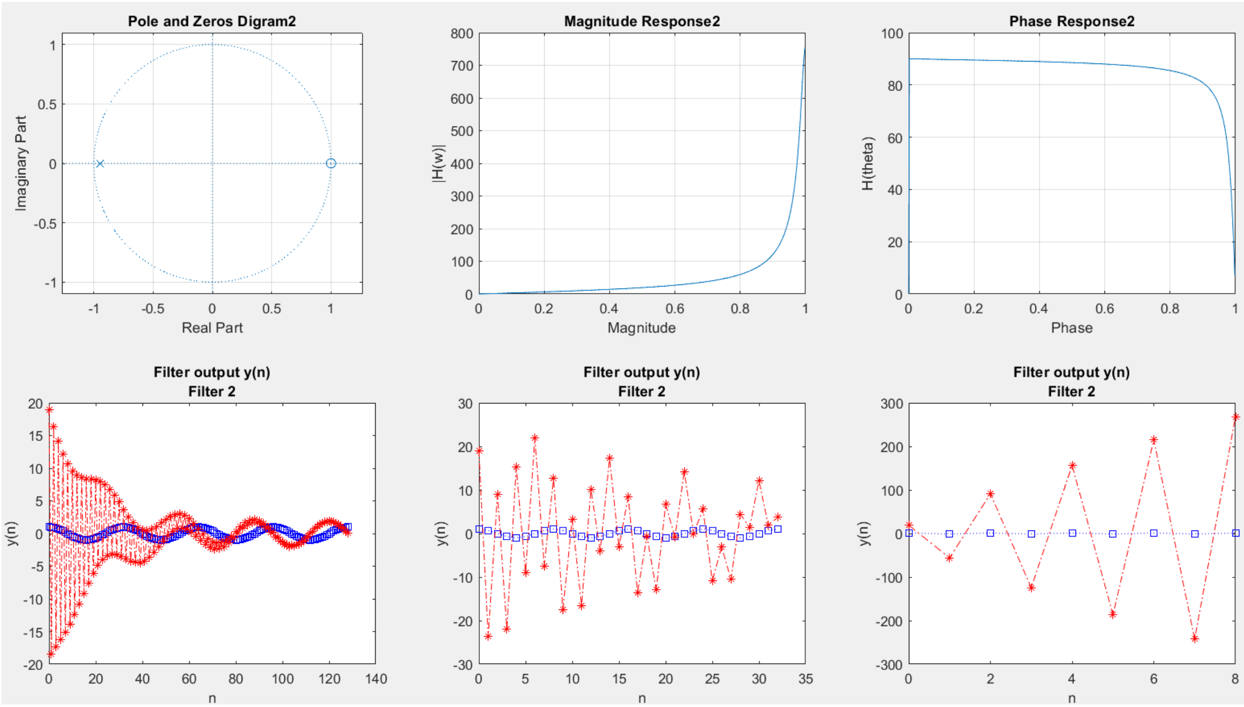
Low pass



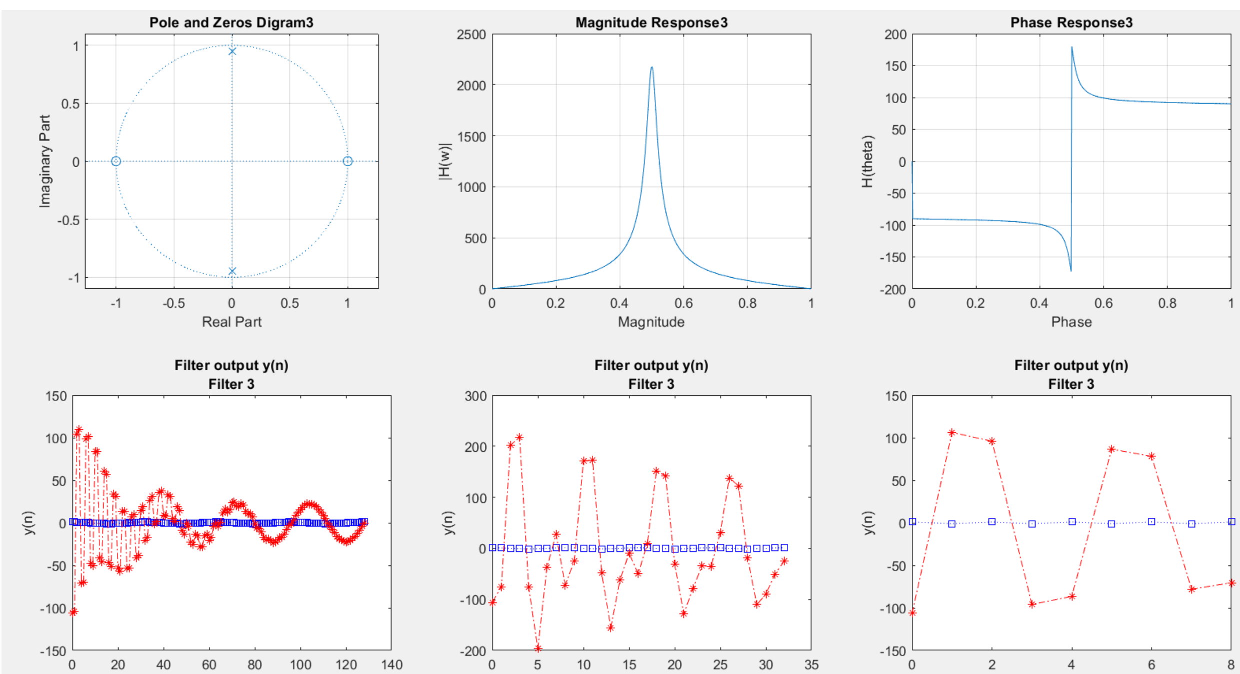
Need to show more samples

There is an issue with the output as it is larger than the input

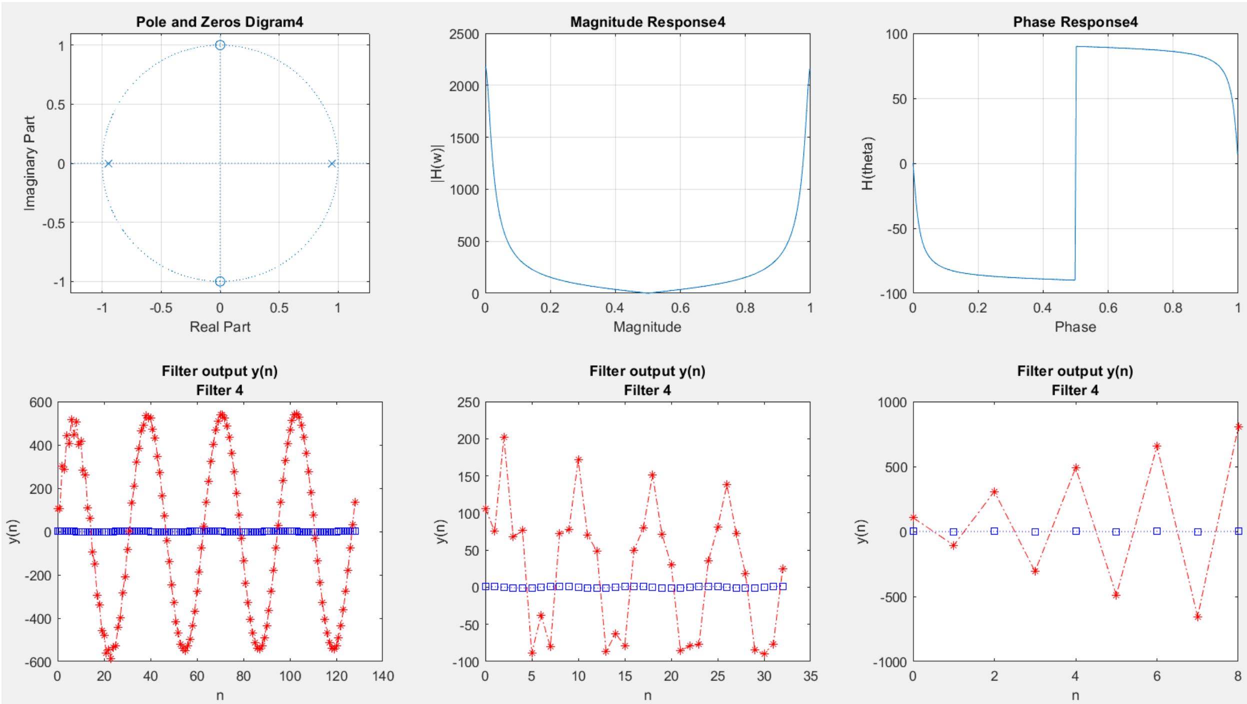
High pass



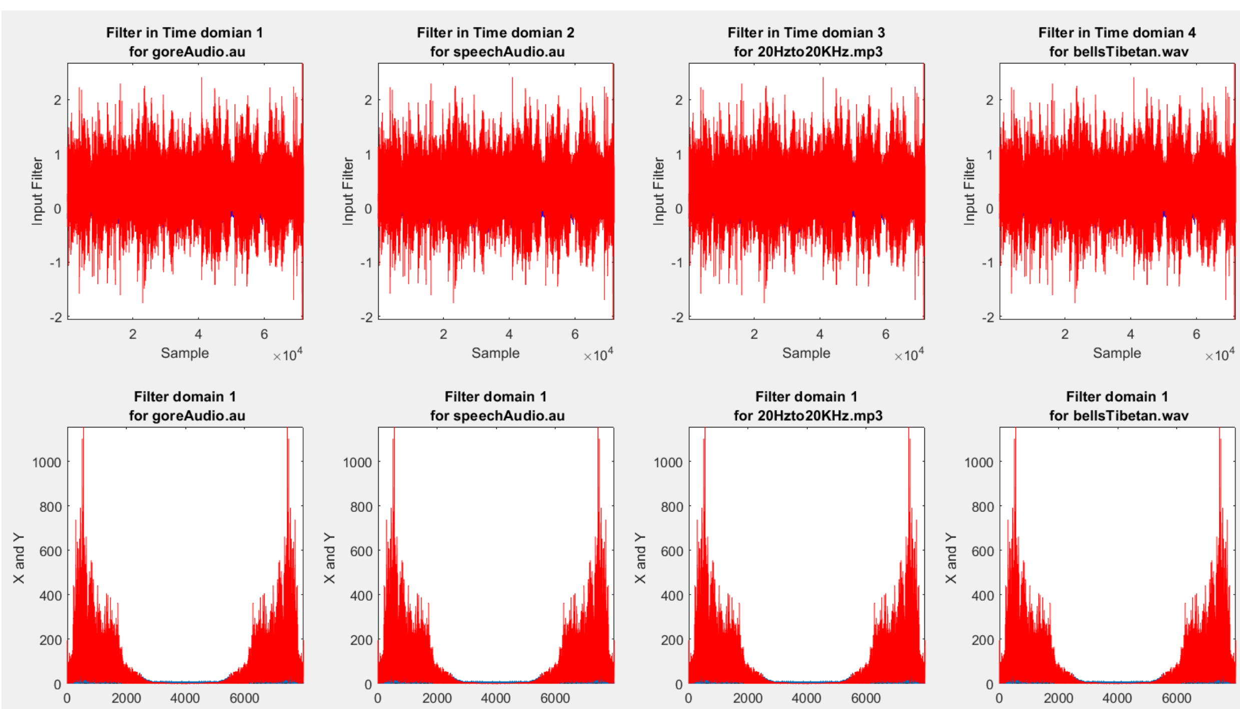
Band pass

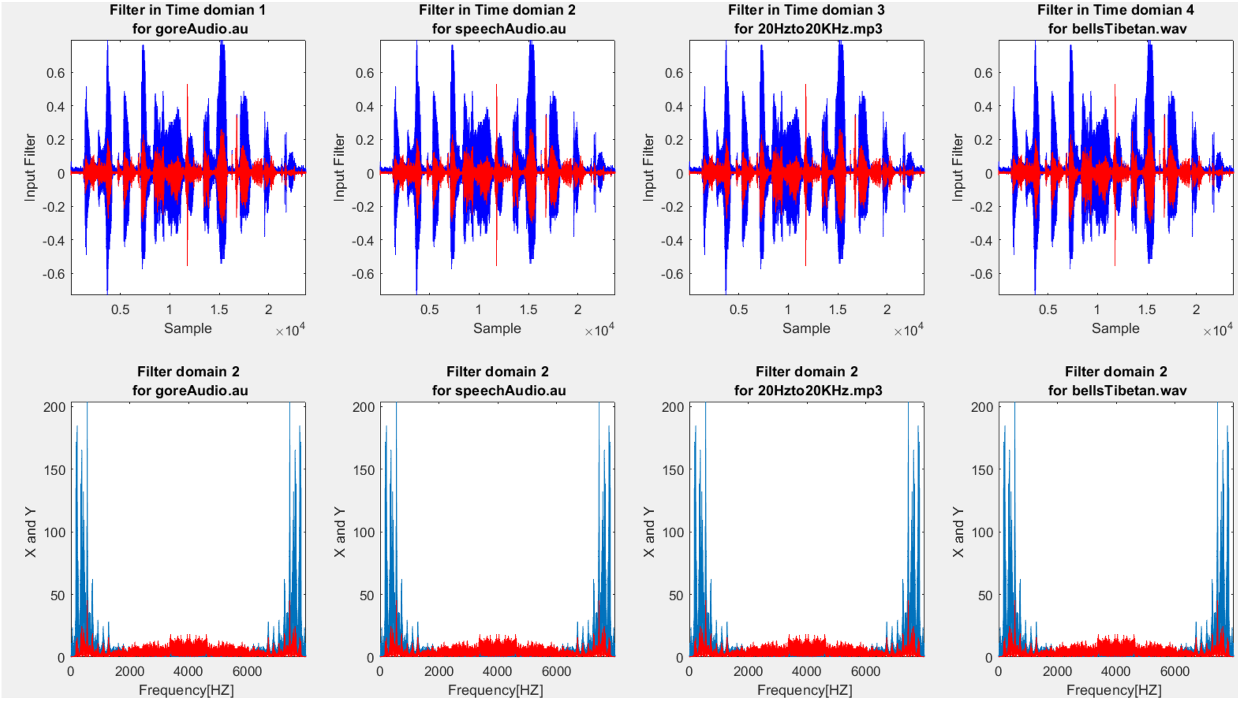


Band Stop filter

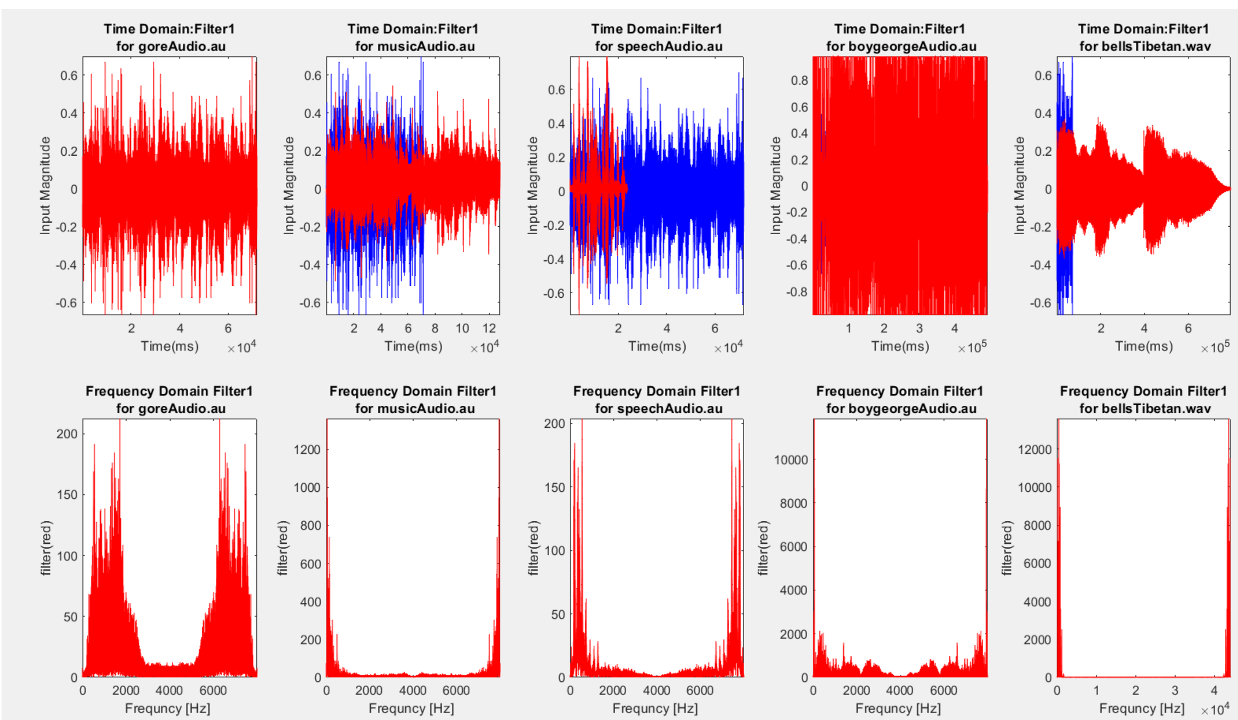


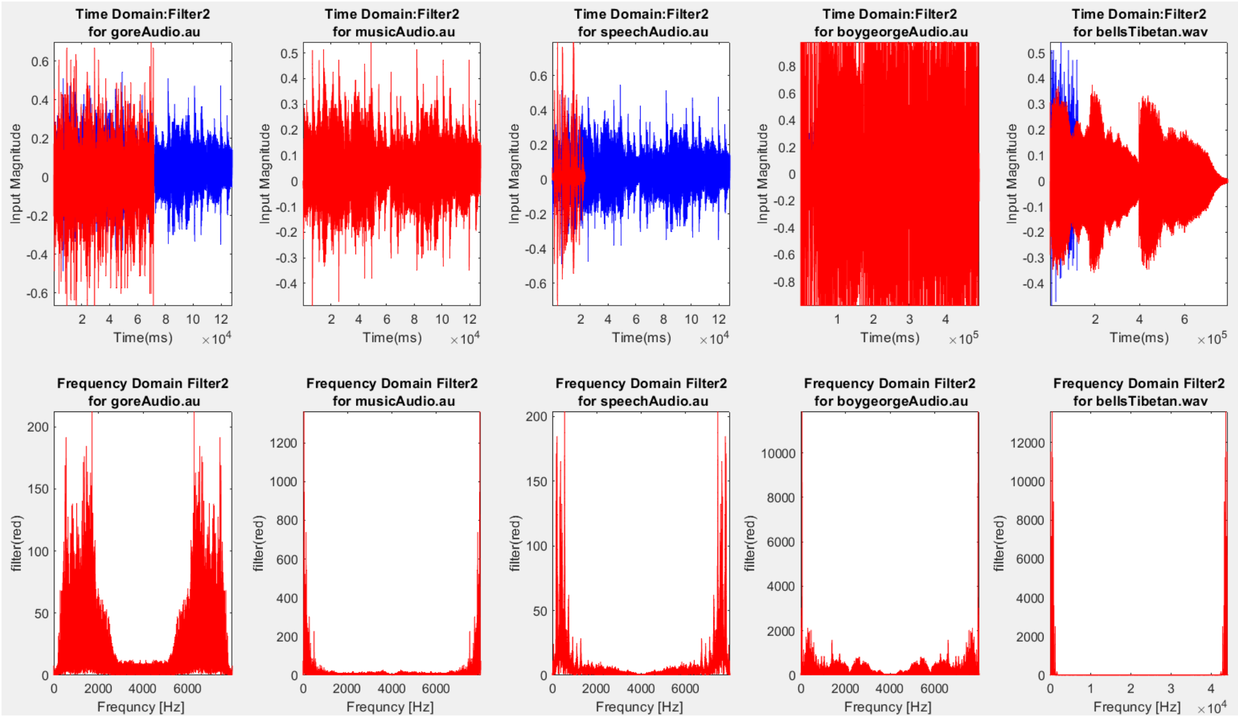
Lab 8 part A

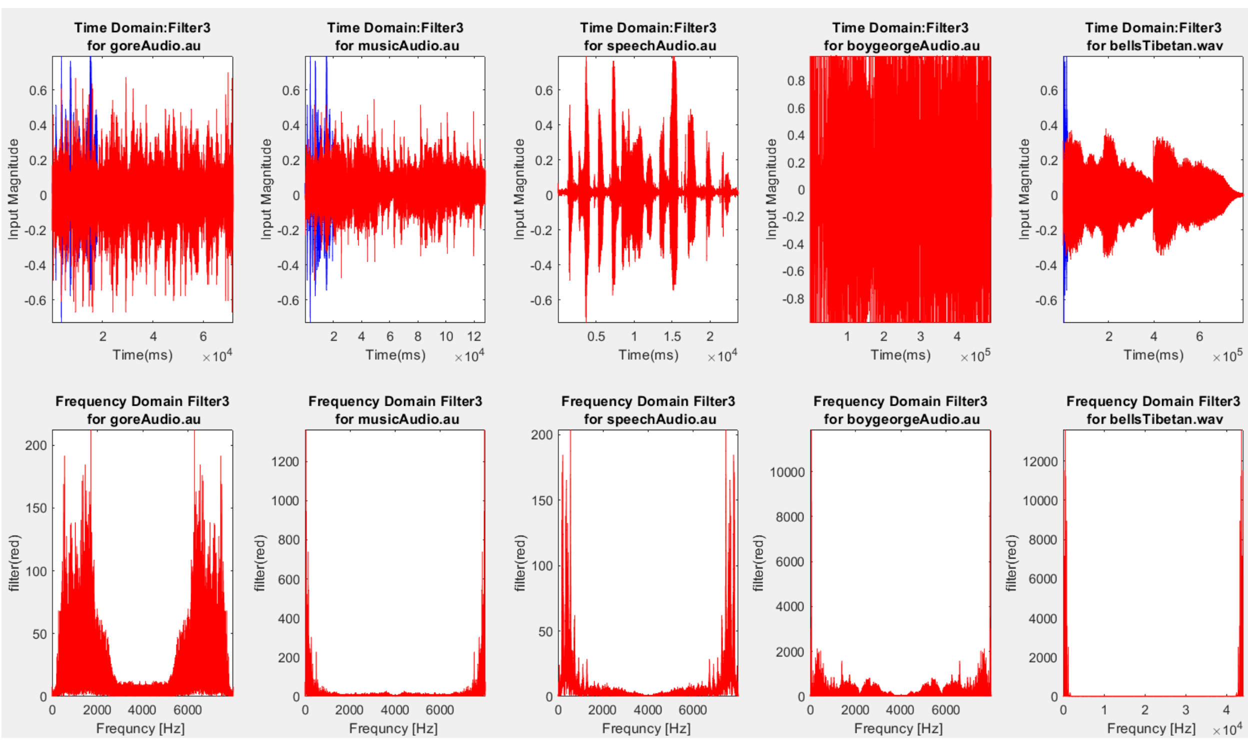


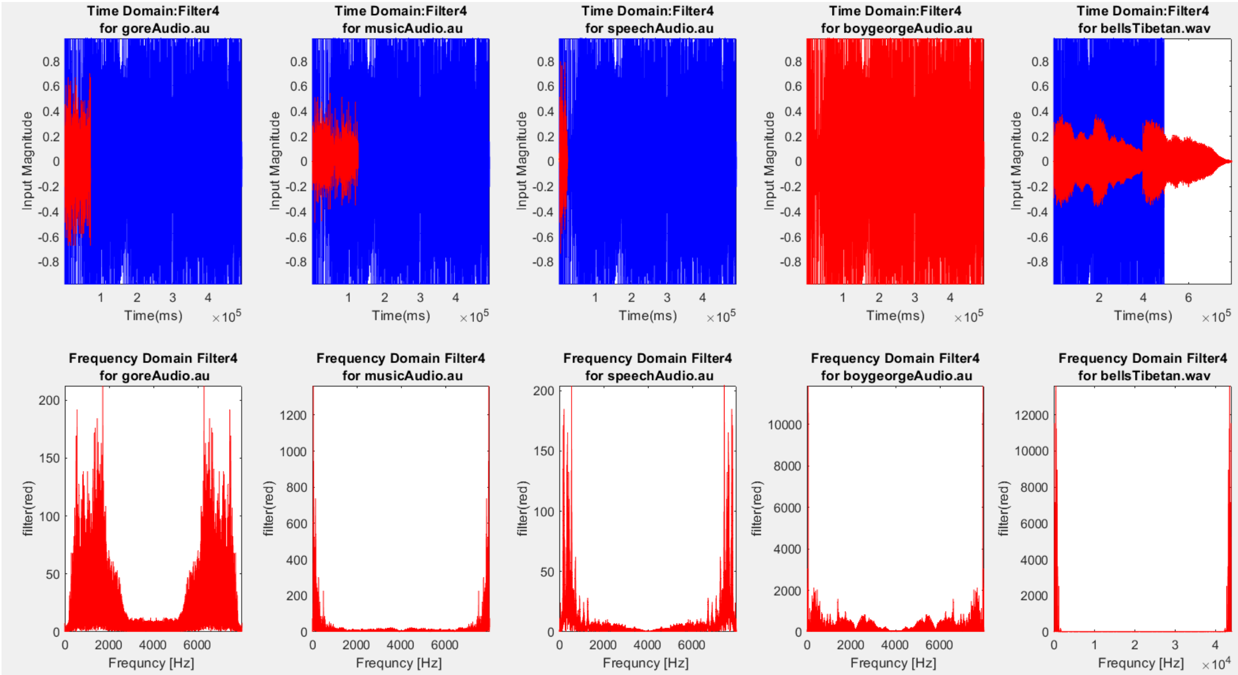

Filter in Time domian 1
for goreAudio.au
Filter in Time domian 2
for speechAudio.au
Filter in Time domian 3
for 20Hzto20KHz.mp3
Filter in Time domian 4
for bellsTibetan.wav
Input Filter
Sample
Filter domain 2
for goreAudio.au
for speechAudio.au
for 20Hzto20KHz.mp3
for bellsTibetan.wav
X and Y
Frequency[HZ]

Lab 8 part B


Time Domain:Filter1
for goreAudio.au
for musicAudio.au
for speechAudio.au
for boygeorgeAudio.au
for bellsTibetan.wav
Input Magnitude
Time(ms)
Frequency Domain Filter1
filter(red)
Frequncy [Hz]


Time Domain:Filter2 for goreAudio.au
Time Domain:Filter2 for musicAudio.au
Time Domain:Filter2 for speechAudio.au
Time Domain:Filter2 for boygeorgeAudio.au
Time Domain:Filter2 for bellsTibetan.wav
Input Magnitude
Time(ms)
Frequency Domain Filter2 for goreAudio.au
Frequency Domain Filter2 for musicAudio.au
Frequency Domain Filter2 for speechAudio.au
Frequency Domain Filter2 for boygeorgeAudio.au
Frequency Domain Filter2 for bellsTibetan.wav
filter(red)
Frequncy [Hz]


Time Domain:Filter3 for goreAudio.au
Time Domain:Filter3 for musicAudio.au
Time Domain:Filter3 for speechAudio.au
Time Domain:Filter3 for boygeorgeAudio.au
Time Domain:Filter3 for bellsTibetan.wav
Input Magnitude
Time(ms)
Frequency Domain Filter3 for goreAudio.au
Frequency Domain Filter3 for musicAudio.au
Frequency Domain Filter3 for speechAudio.au
Frequency Domain Filter3 for boygeorgeAudio.au
Frequency Domain Filter3 for bellsTibetan.wav
filter(red)
Frequncy [Hz]



Conclusion

In conclusion we were able to understand how is signals could be analyse signal in frequency domain and time domain while also understanding there characteristics Z-transform and the discrete-time Fourier Transform we where able to understand how 4 different signal interacted with the four different filter the poles and zeros that interacted in the prelab correspond to different to the low pass filter, the high pass filter, the band pass filter and the band stop filter

Resources (Matlab code)

Prelab 8

```
clc, clear all, close all;
%transfer function (TF) gain and poles and zeros (z polynomial roots)
zr{1} = [-1]; % Filter 1 zero (TF numerator polynomial roots)
pl{1} = [.95]; % Filter 1 pole (TF denominator polynomial roots)
zr{2} = [1]; %  Filter 2 zero (TF numerator polynomial roots)
pl{2} = [-.95]; % Filter 2 pole (TF denominator polynomial roots)
zr{3} = [1,-1]; % Filter 3 zero (TF numerator polynomial roots)
pl{3} = [.95*1i,-.95*1i]; % Filter 3 poles (TF denominator polynomial roots)
zr{4} = [1i,-1i]; % Filter 4 zero (TF numerator polynomial roots)
pl{4} = [.95,-.95]; % Filter 4 poles (TF denominator polynomial roots)
%frequency w(rads/s) at which the filter gain is maximum
max_gain_freq=[0,pi,pi/2,0];
%Plot pole-zero z-domain, frequency response and sampled sinusoid for each
%of the 4 filters
for flt=1:4
    z_max_gain=exp(1i*max_gain_freq(flt));%z=e^jw at w at which the filter has ↙
maximum gain
    %given numerator(zeros)&denominator(poles)TF polynomial roots, find the ↙
polynomial coefficients
    tfnum_polycoef_zero=poly(zr{flt}) %numerator TF z-polynomial coefficients
    tfden_polycoef_pole=poly(pl{flt}) %denominator TF z-polynomial coefficients
    % gain factor b0 to ensure max TF gain is 1 at max gain frequency
    %polyval(polynomial coefficients, polynomial variable z value)
    tf_b0_gain=real(polyval(tfnum_polycoef_zero,z_max_gain/polyval ↙
(tfden_polycoef_pole,z_max_gain)))
    %plot pole-zero diagram======================
    fsf(flt)=figure(flt);
    set(fsf(flt), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);%fullscreen
    subplot(2,3,1);
    %zplane(zero polynomial, pole polynomial)
    zplane(tfnum_polycoef_zero, tfden_polycoef_pole);grid;
    title(['Pole and Zeros Digram',num2str(flt)]);
    %plot magnitude response |H(theta)|===========
    %freqz(TF numerator, TF denominator, number of frequency points)
    [H,w]=freqz(tf_b0_gain* tfnum_polycoef_zero, tfden_polycoef_pole,500);
    subplot(2,3,2);
    plot(w/pi,abs(H));grid;
    xlabel('Magnitude');ylabel('|H(w)|');
    title(['Magnitude Response',num2str(flt)]);
    %plot phase response angle(H(theta))
    subplot(2,3,3);
    plot(w/pi,angle(H)*180/pi);grid;
    xlabel('Phase');ylabel('H(theta)');
    title(['Phase Response',num2str(flt)]);
    %plot y(n)& x(n) together
    number_of_cycles=4;%some filters take many cycles to converge
    fundamental_frequency = 30; %fo in Hertz
    samples_per_cycle = [32, 8, 2];%fs=fo*sample_per_cycle
    for sampling_frequency_index=1:length(samples_per_cycle)
```

```
        sampling_frequency_fs=samples_per_cycle(sampling_frequency_index)*2;
        number_of_samples=number_of_cycles*samples_per_cycle↙
(sampling_frequency_index);
        n=0:number_of_samples;
        xn=cos(n*2*pi/samples_per_cycle(sampling_frequency_index));
        %filter(TF nummerator, TF denominator, sampled signal)
        y=filter(tfnum_polycoef_zero * tf_b0_gain, tfden_polycoef_pole, xn);
        subplot(2,length(samples_per_cycle),3+sampling_frequency_index);
        %plot sampled sinusoid filter input
        plot( n, xn,':bs');
        %hold the input plot while overlaying the output plot on top of it
        hold on
        %plot sampled sinusoid filter output
        plot( n, y,'-.r*');
        xlabel('n');ylabel('y(n)');
        title({['Filter output y(n)'],[ 'Filter ',num2str(flt)]});
    end
end
```

Lab 8 part A

```
clc, clear all, close all;
audiofiles = {'goreAudio.au','speechAudio.au','20Hzto20KHz.mp3','bellsTibetan.wav'};
 % Z polynomial coefficients for numerator (for zeros) and denominator
%(for poles) for each filter
tfnum_polycoef_zero{1}=[1 1]; % Filter 1
tfden_polycoef_pole{1}=[1 -0.9];
tfnum_polycoef_zero{2}=[1 -1]; %Filter 2
tfden_polycoef_pole{2}=[1 .9];
tfnum_polycoef_zero{3}=[1 0 -1]; %Filter 3
tfden_polycoef_pole{3}=[1 0 .81];
tfnum_polycoef_zero{4}=[1 0 1]; %Filter 4
tfden_polycoef_pole{4}=[1 0 -.81];
max_gain_freq=[.05,.05,.095,.095];
for flt=1:4 %filter audio with 4 filters, plot time/frequency domain output
    z_max_gain=exp(1i*(max_gain_freq(flt))); %z=e^jw at w where filter gain is maximum
    % gain factor to ensure max TF gain is 1 at max gain frequency
    tf_b0_gain=real(polyval(cell2mat(tfnum_polycoef_zero(flt)),max_gain_freq)/polyval ↙
(cell2mat(tfden_polycoef_pole(flt)),max_gain_freq));
    for audfls = 1:length(audiofiles)
        [input_audio{audfls}, Fs{audfls}] = audioread(audiofiles{flt});
        filtered_audio=filter(cell2mat(tfnum_polycoef_zero(flt))*tf_b0_gain,cell2mat ↙
(tfden_polycoef_pole(flt)),input_audio{audfls});
        fsf(flt)=figure(flt);
        set(fsf(flt), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(2,length(audiofiles),audfls);
        plot(input_audio{audfls},'b')
        axis tight;
        hold on
        plot(filtered_audio,'r')
        title({['Filter in Time domian ', num2str(audfls) ],[' for ' audiofiles↙
{audfls}]});
        xlabel('Sample'); ylabel('Input Filter');
        axis tight;
        % FFT
        fft_input_audio=fft(input_audio{audfls},length(input_audio{audfls}));
        fft_input_audio(1)=0;
        nf=Fs{audfls}*(0:length(input_audio{audfls})-1)/length(input_audio{audfls});
        subplot(2,length(audiofiles),audfls+ length(audiofiles));
        plot(nf,abs(fft_input_audio));
        fft_filtered_audio=fft(filtered_audio, length(filtered_audio));
        fft_filtered_audio(1)=0
        axis tight;
        hold on
        nf=Fs{audfls}*(0:length(input_audio{audfls})-1)/length(input_audio{audfls});
        plot(nf,abs(fft_filtered_audio),'r')
        title({['Filter domain ', num2str(flt) ],[' for ' audiofiles{audfls}]});
        xlabel('Frequency[HZ] ')
        ylabel('X and Y ')
        axis tight;

         if (audfls==length(audiofiles) && flt==1) %play the last filtered audio file
             soundsc(filtered_audio,Fs{audfls},16)
         end
     end
 end
```

Lab 8 part B

```
clc, clear all, close all;
audiofiles = {'goreAudio.au','musicAudio.au','speechAudio.au','boygeorgeAudio.↙
au','bellsTibetan.wav'};
 %transfer function (TF) gain and poles and zeros (z polynomial roots)
zr{1} = [-1]; % Filter 1
pl{1} = [0.95];
zr{2} = [1]; % Filter 2
pl{2} = [-0.95];
zr{3} = [1,-1]; % Filter 3
pl{3} = [.95*1i,-.95*1i];
zr{4} = [1i,-1i];; % Filter 4
pl{4} = [.95,-.95]
max_gain_freq=[0,pi,pi/2,0];
for flt=1:4
    z_max_gain=exp(1i*max_gain_freq(flt));
    %numerator(zeros)&denominator(poles) TF polynomial coefficients
    tfnum_polycoef_zero = poly(zr{flt});
    tfden_polycoef_pole = poly(zr{flt});
    % gain factor to ensure max TF gain is 1 at max gain frequency
    tf_b0_gain=real(polyval(tfnum_polycoef_zero,z_max_gain)/polyval↙
(tfden_polycoef_pole,z_max_gain));
    for audfls = 1:length(audiofiles)
        [input_audio{audfls}, Fs{audfls}] = audioread(audiofiles{audfls});
        filtered_audio=filter(tf_b0_gain*tfnum_polycoef_zero,tfden_polycoef_pole, ↙
input_audio{audfls});
        fsf(flt)=figure(flt);
        set(fsf(flt), 'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
        subplot(2,length(audiofiles),audfls);
        plot(input_audio{flt},'b')
        axis tight;
        hold on
        plot(filtered_audio,'r')
        title({['Time Domain:Filter', num2str(flt) ],[' for ' audiofiles{audfls}]});
        xlabel('Time(ms)'); ylabel('Input Magnitude');
        axis tight;
        % FFT
        fft_input_audio=fft(input_audio{audfls},length(input_audio{audfls}));
        fft_input_audio(1)=0;
        nf=Fs{audfls}*(0:length(input_audio{audfls})-1)/length(input_audio{audfls});
        subplot(2,length(audiofiles),audfls+ length(audiofiles));
        plot(nf,abs(fft_input_audio));
        fft_filtered_audio=fft(filtered_audio, length(filtered_audio));
        fft_filtered_audio(1)=0;
        axis tight;
        hold on
        nf=Fs{audfls}*(0:length(input_audio{audfls})-1)/length(input_audio{audfls});
        plot(nf,abs(fft_filtered_audio),'r')
        title({['Frequency Domain Filter', num2str(flt) ],[' for ' audiofiles↙
{audfls}]});
```

```
        xlabel('Frequncy [Hz] ')
        ylabel('sample(?),filter(?)')
        axis tight;
        if (audfls==length(audiofiles) && flt==1) %play the last filtered audio file
            soundsc(filtered_audio,Fs{audfls},16)
        end
    end
end
```