DSP Project : Design Audio Enhancement System

2015104000 김재현
2016110556 송은빈
2016104063 조성환

1. Purpose

Design the best audio enhancement system using digital filters under the noisy environment

2. Noisy signal generation

[code]

```
clear all; close all;
[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

figure(1);
plot(t,sig); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

figure(2);
plot(t,noise); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

figure(3);
plot(t,noisy_sig); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

audiowrite('noisy_sig.wav',noisy_sig,fs);
```

original audio signal에 Noise를 더한 audio signal이 Noisy signal이다.

실험에 사용할 original audio signal로 Sampling frequency $f_s = 44.1\,\mathrm{kHz}$인 'No9seg.wav'를 사용하였다. 또한 original signal이 stereo channel이므로 실험 조건에 맞게 mono channel로 바꾸어 주었다.

SNR = 20dB인 (0,1) random normal distributed Noise signal을 생성하기 위해 $\mathrm{var}[aX] = a^2\mathrm{var}[X]$에서의 a 값을 코드에서 바꿔가며 조정하였고, 20dB에 가까운 snr=20.0382 값으로 생성된 noise를 original audio signal에 합하여 noisy signal을 생성하였다. 이 signal을 audiowrite 함수를 통해 wav 파일로 저장하였다.

[plot]

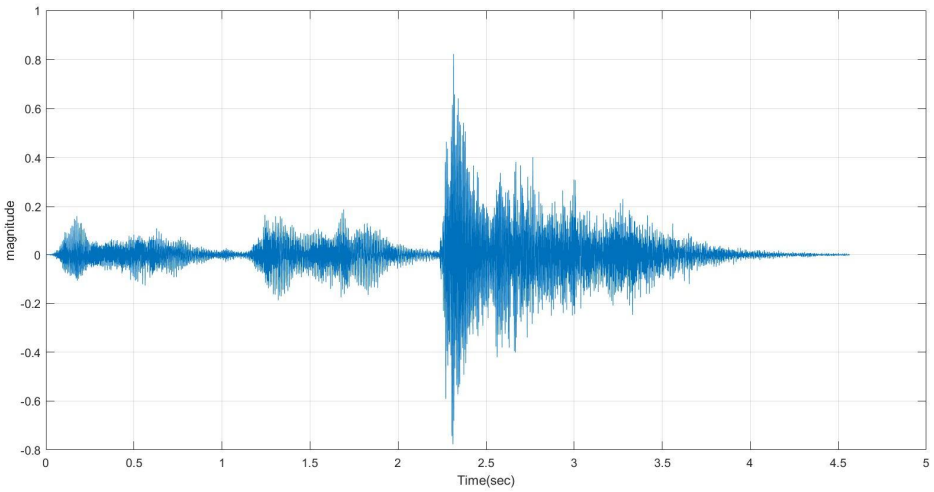

Figure 1. Original Signal

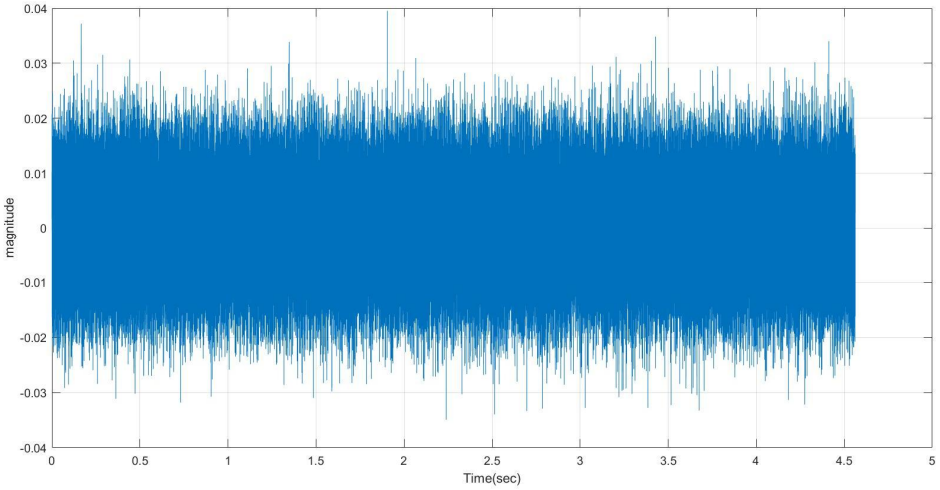

Figure 2. Noise

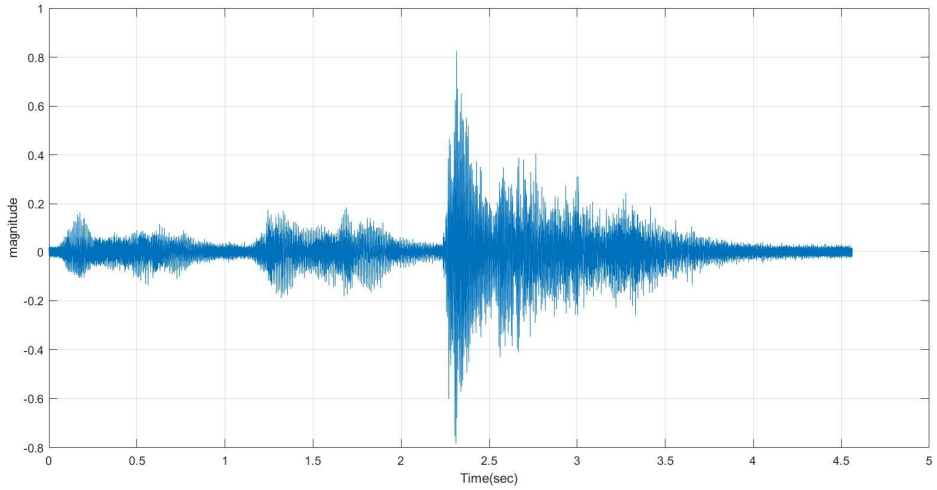

Figure 3. Noisy Signal (Original Signal + Noise)

3. Spectrum analysis for specify the filter requirements

Original signal과 Noisy signal을 Spectrum 분석하기 위해,

신호를 FFT하였다.

[code]

```
clear all; close all;
[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

F = abs(fft(sig))/n;
noiseF = abs(fft(noise))/n;
Noisy_sigF = abs(fft(noisy_sig))/n;

figure(4);
plot(f,2*F(2:n/2+1)); grid; axis([0 25000 0 0.026]);
xlabel('Frequency(Hz)'); ylabel('Spectrum');

figure(5);
plot(f,2*Noisy_sigF(2:n/2+1)); grid; axis([0 25000 0 0.026]);
xlabel('Frequency(Hz)'); ylabel('Spectrum');
```

[plot]

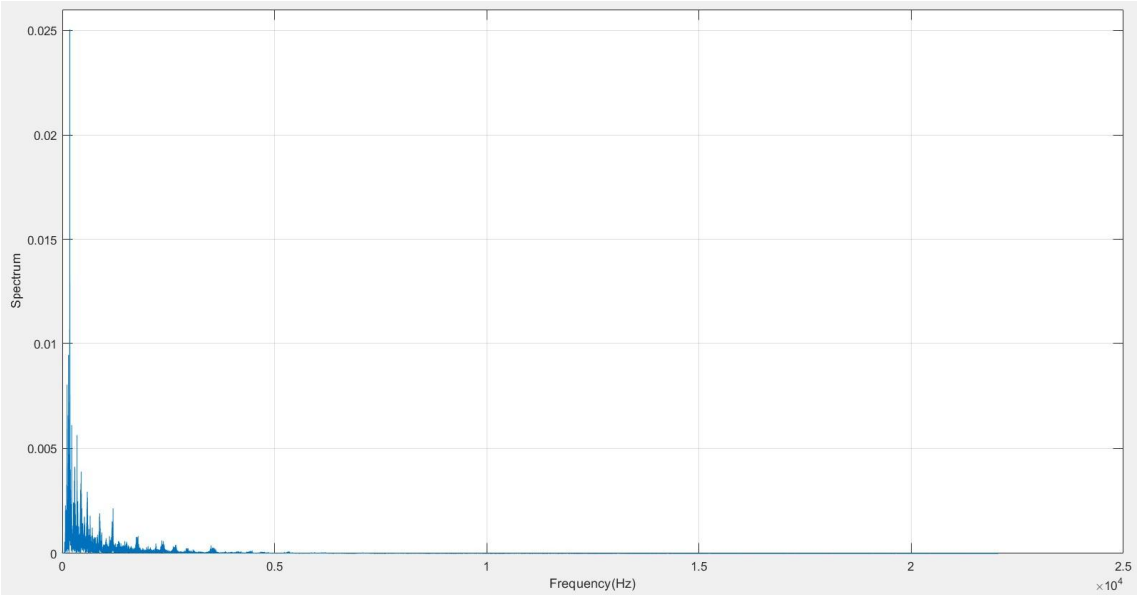

Figure 4.1 Original Signal spectrum

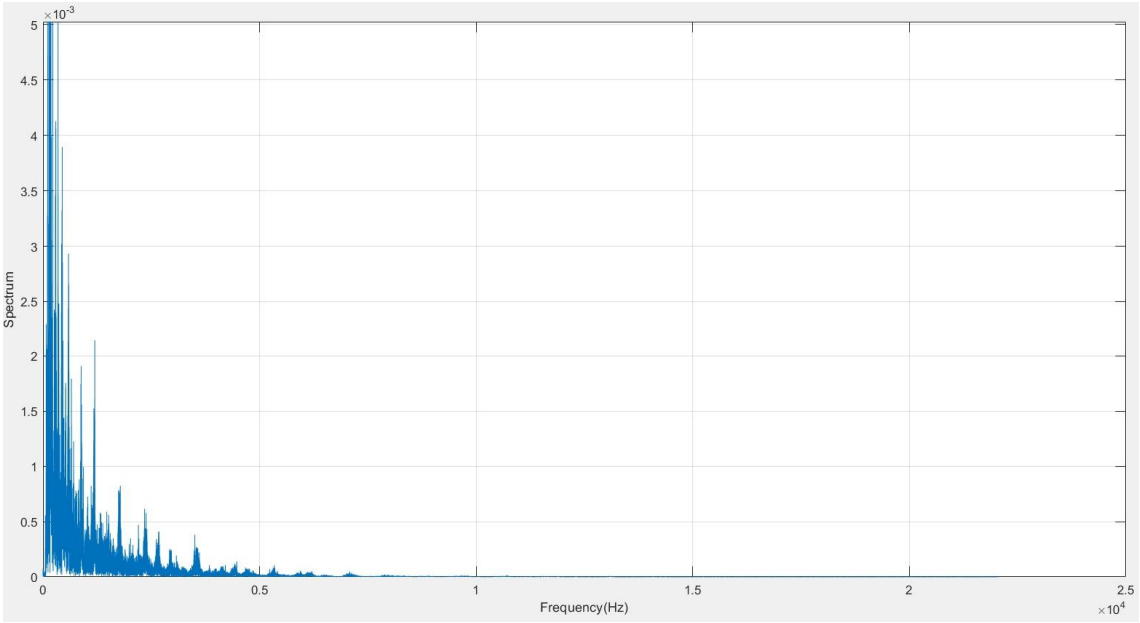

Figure 4.2 Original Signal spectrum (확대)

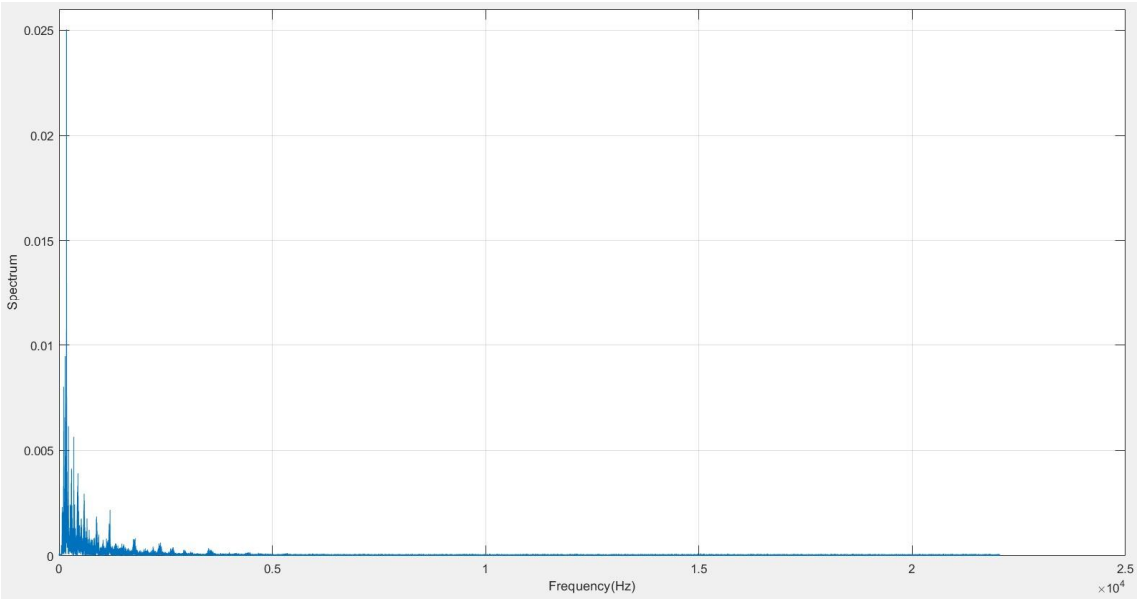

Figure 5.1 Noisy Signal spectrum

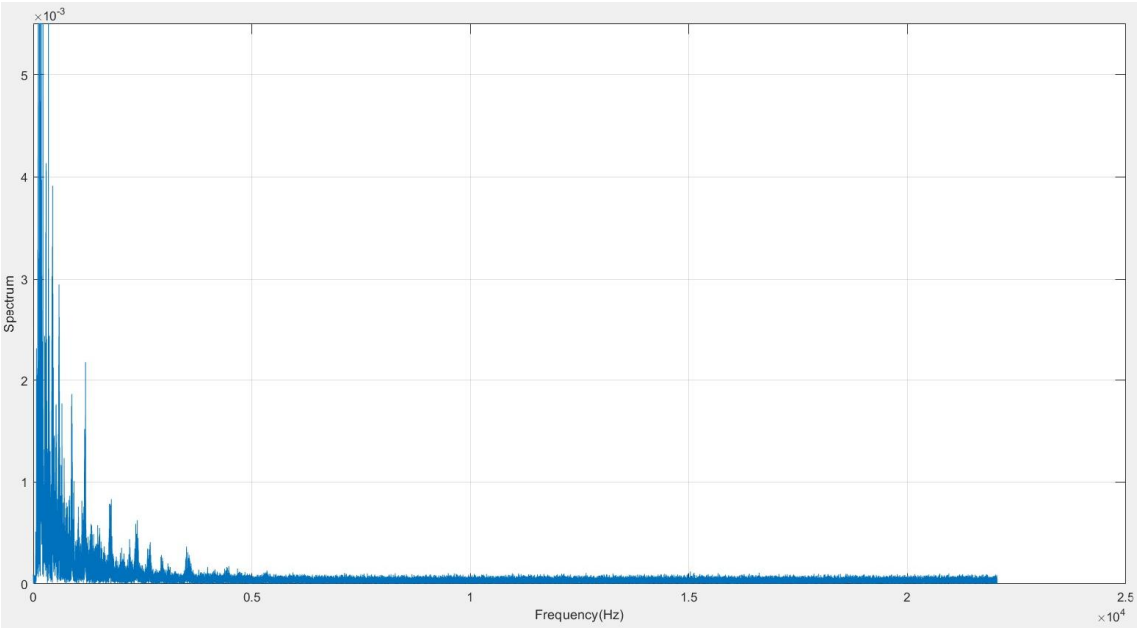

Figure 5.2 Noisy Signal spectrum (확대)

Original signal과 비교했을 때 Noisy signal은 noise가 추가되어 signal spectrum이 전 주파수 대역에 존재했다.

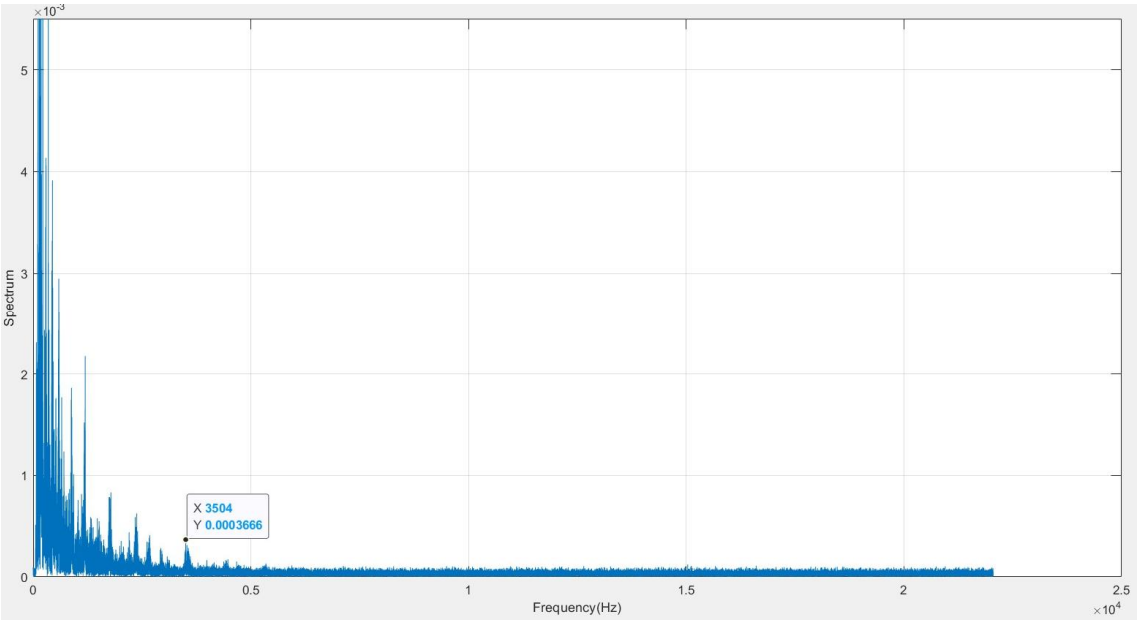

Figure 5.3 Determine Passband frequency

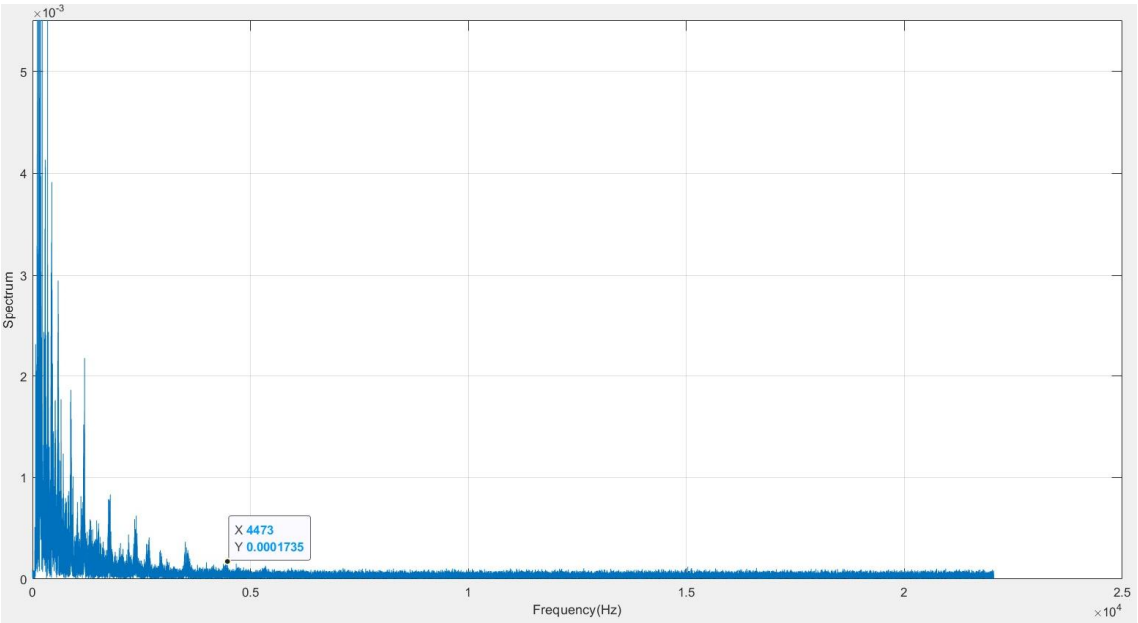

Figure 5.4 Determine Stopband frequency

따라서 Passband : 0 - 3,500 Hz, Stopband : 4,500 - 22,050 Hz인 Low Pass Filter를 설계하여 Noisy signal을 original signal에 가깝게 만들고자 하였다.
Stopband frequency 값을 결정할 때, frequency를 3500Hz에 가깝게 할수록 transition band는 좁아져 filter가 sharp해진다. 필터가 sharp해지면 filter coefficient 개수가 증가하는데, 이는 필터의 설계 비용 증가로 이어지기 때문에 비용적인 측면을 고려해서 적절한 stopband frequency를 설정하였다.

따라서 설계할 필터의 cutoff frequency는 $f_c = \frac{3500+4500}{2} = 4000\,\mathrm{Hz}$ 이다.

Passband ripple과 stopband attenuation은 필터의 성능을 결정한다.
좋은 성능을 가진 필터일수록 설계 비용 또한 증가하게 된다. 필터 성능을 조금 향상 시키기 위해 과도한 비용을 소모한다면, 이는 적절하지 않은 설계이다. 따라서 성능뿐만이 아닌 비용적인 측면도 동시에 고려하여 적절한 Passband ripple과 Stopband attenuation 값을 설정해 줄 필요가 있다.

Passband ripple과 stopband attenuation의 적절한 값을 결정하기 위해서 필터된 소리를 직접 들어보며 비교하기 위해 필터를 임시로 설계하였는데, 이때 passband ripple 값과 stopband attenuation 값을 바꿔가며 비교하기 상대적으로 쉬운 optimal design method를 사용하여 필터를 임시로 설계하였다. 그 이유는 optimal design method는 값을 비교하기 위해 필터를 수정할 때 w 값만 바꾸면 비교가 가능하기 때문이다.
passband ripple 값 변화에 따른 필터 성능을 비교하기 위해 stopband attenuation이 동일한 조건에서 passband ripple을 일정 간격으로 변화시켜 필터의 성능 차이를 비교한 다음, 반대로 stopband attenuation 값 변화에 따른 필터 성능을 비교하기 위해 passband ripple이 동일한 조건에서 stopband attenuation을 일정 간격으로 변화시켜서 필터의 성능 차이를 비교하였다. 이 때 filtered signal을 직접 들어봤을 때 이전 신호와의 큰 성능 차이를 느끼지 못한 지점에서 각각 ripple과 attenuation 값을 결정하였다. 이 이유는 직접 들어봤을 때 큰 차이를 느끼지 못한다면 filter의 성능 증가에 대비하여 비용적인 손실이 크기 때문에 이 지점에서 filter 성능을 더욱더 향상시키는 것은 비효율적일 것이기 때문이다.

Folding frequency : $\frac{f_s}{2} = 22050\mathrm{Hz}$
· For 0 Hz : 0/22050 = 0, magnitude: 1
· For 3500 Hz : 3500/22050 = 0.1587, magnitude: 1
· For 4500 Hz : 4500/22050 = 0.2041, magnitude: 0
· For 22050 Hz : 22050/22050 = 1, magnitude: 0

∴f=[0 0.1587 0.2041 1]
∴m=[1 1 0 0]

$$\delta_p = 10^{\left(\frac{\delta_p dB}{20}\right)} - 1$$

$$\delta_s = 10^{\left(-\frac{\delta_s dB}{20}\right)}$$

$$\frac{\delta_p}{\delta_s} = \frac{W_s}{W_p}$$

$$\therefore w=[\ W_p \quad W_s\ ]$$

여러 값을 대입하며 w 값을 바꿔가면서 소리를 비교하여 들어본 결과,
ripple 값은 1dB, stopband attenuation 값은 20dB로 결정하였다.

- Passband : 0 - 3,500 Hz,
- Stopband : 4,500 - 22,050 Hz
- Passband ripple = 1 dB
- Stopband attenuation = 20 dB
- Sampling rate = 44100 Hz
- Cutoff frequency = 4000 Hz

4. Design the best digital filter Based on Analysis of System Performance

1) FIR filter - window method (Rectangular Window)

[code : FIR filter - window method (Rectangular Window)]

```
clear all; close all;
[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

F = abs(fft(sig))/n;
noiseF = abs(fft(noise))/n;
Noisy_sigF = abs(fft(noisy_sig))/n;

Fc=4000/fs; % (cut-off frequency/folding frequency)
deltaf=(4500-3500)/fs;

N=round(0.9/deltaf); %determine N
if mod(N,2)==0
    N=N+1 % N 홀수로 보정
else
    N
end

Brec = fir1(N-1,Fc,boxcar(N));
[hrec,f]=freqz(Brec,1,512,fs);
prec=180*unwrap(angle(hrec))/pi;
fir_noise = filtfilt(Brec,1,noise);
fir_rect = filtfilt(Brec,1,noisy_sig);
snr(fir_rect,fir_noise)
audiowrite('fir_rect.wav',fir_rect,fs);

figure(6);
subplot(2,1,1);
plot(f,20*log10(abs(hrec)));
grid; axis([0 4000 -100 10]); grid on;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -100 5]);
subplot(2,1,2);
plot(f,prec); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -450 5]);

figure(7)
fir_rectInfo = audioinfo('fir_rect.wav');
t = 0:seconds(1/fs):seconds(fir_rectInfo.Duration);
[rect,Fs] = audioread('fir_rect.wav');
t = t(1:end-1);
plot(t,rect);
title('FIR filter - window method');
xlabel('Time');
ylabel('Audio Signal');
```

[result]

N = 41

[plot]

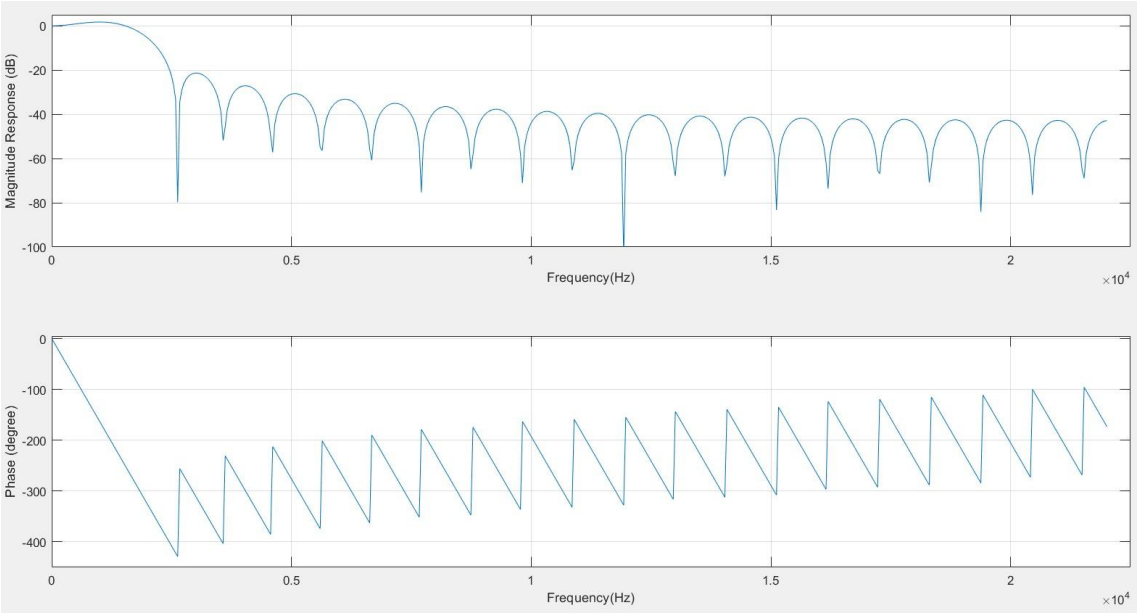

Figure 6. FIR filter (window method - rectangular)

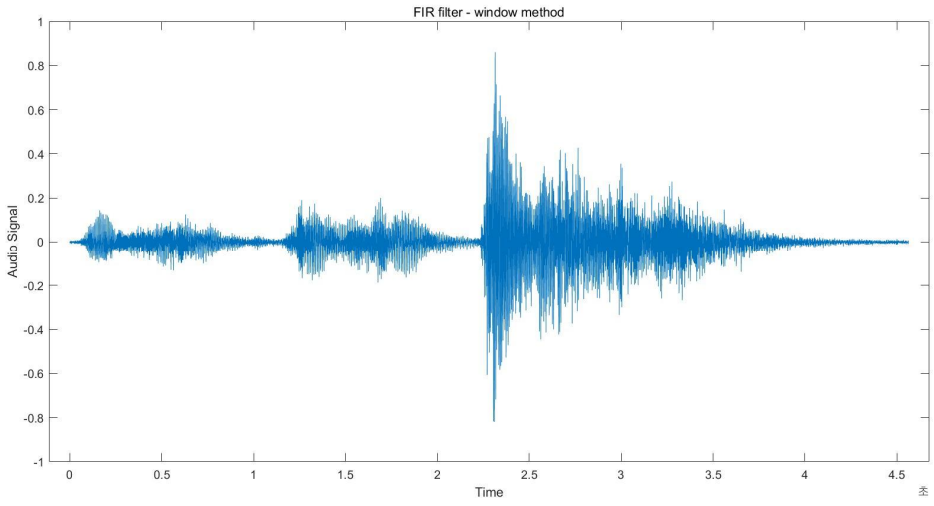

Figure 7. FIR filter (window method - rectangular) audio data

[objective testing]

SNR : noisy signal	SNR : FIR(window method (rectangular)) filtered noisy signal
ans = 20.038160281204380	ans = 29.616859756816105

[subjective testing]

noise는 매우 제거되었다고 느꼈지만,

원신호보다 소리가 상당히 뭉개졌음을 느낄 수 있었다.

2) FIR filter - optimal design method

Folding frequency : $\frac{f_s}{2} = 22050\text{Hz}$

· For 0 Hz : 0/22050 = 0, magnitude: 1
· For 3500 Hz : 3500/22050 = 0.1587, magnitude: 1
· For 4500 Hz : 4500/22050 = 0.2041, magnitude: 0
· For 22050 Hz : 22050/22050 = 1, magnitude: 0

∴f=[0 0.1587 0.2041 1]
∴m=[1 1 0 0]

$$\delta_p = 10^{\left(\frac{\delta_p dB}{20}\right)} - 1 = 10^{\frac{1}{20}} - 1 = 0.1220$$

$$\delta_s = 10^{\left(-\frac{\delta_s dB}{20}\right)} = 10^{-\frac{20}{20}} = 0.1$$

$$\frac{\delta_p}{\delta_s} = 1.22 \approx \frac{122}{100} = \frac{W_s}{W_p}$$

$$W_s = 122$$
$$W_p = 100$$

∴w=[100 122]

[code : FIR filter - optimal design method]

```
close all; clear all;
[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

F = abs(fft(sig))/n;
noiseF = abs(fft(noise))/n;
Noisy_sigF = abs(fft(noisy_sig))/n;

Fc=4000/fs; % (cut-off frequency/folding frequency)
deltaf=(4500-3500)/fs;

N=29; % (주어진 조건에 최대한 만족하는 N 값 설정)
f_edge=[0 0.1587 0.2041 1]; %edge frequencies
m=[1 1 0 0]; %ideal magnitudes
w=[100 122]; %error weight factors
b=firpm(N,f_edge,m,w);
format long
[h,freq]=freqz(b,1,512,fs); %determine frequency response
p=180*unwrap(angle(h))/pi; %determine phase response
fir_noise = filtfilt(b,1,noise);
fir_od = filtfilt(b,1,noisy_sig);
snr(fir_od,fir_noise)
audiowrite('fir_optimaldesign.wav',fir_od,fs);

figure(8);
subplot(2,1,1);
plot(freq,20*log10(abs(h))); grid;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -80 5]);
subplot(2,1,2);
plot(freq,p); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -650 5]);

figure(9)
fir_optInfo = audioinfo('fir_optimaldesign.wav');
t = 0:seconds(1/fs):seconds(fir_optInfo.Duration);
[opt,fs] = audioread('fir_optimaldesign.wav');
t = t(1:end-1);
plot(t,opt);
title('FIR filter - Optimal method');
xlabel('Time');
ylabel('Audio Signal');
```

[plot]

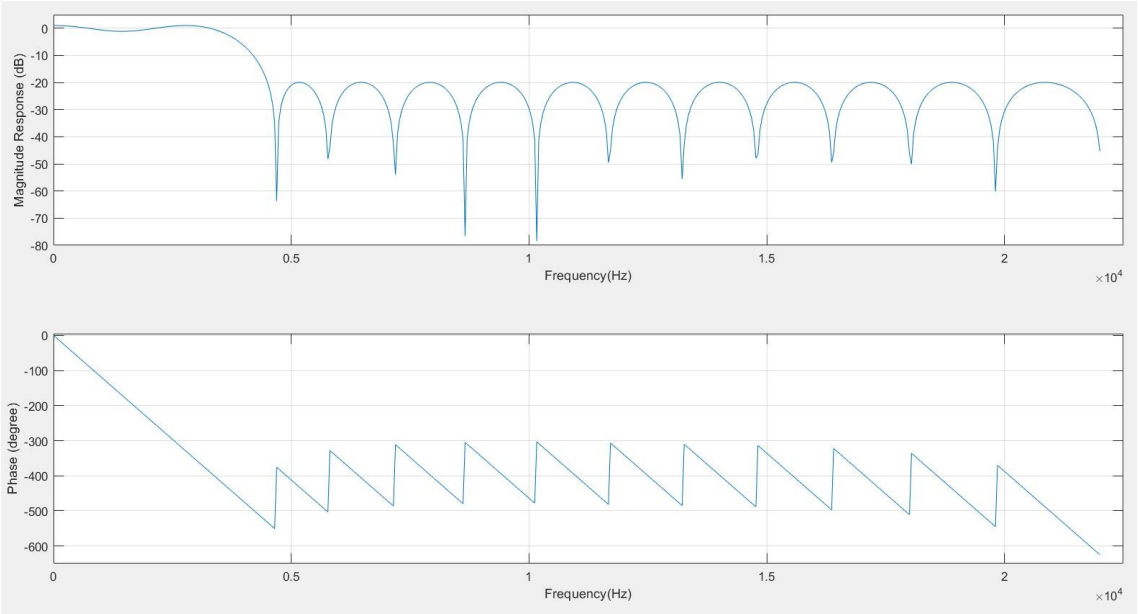

Figure 8. FIR filter (optimal design method)

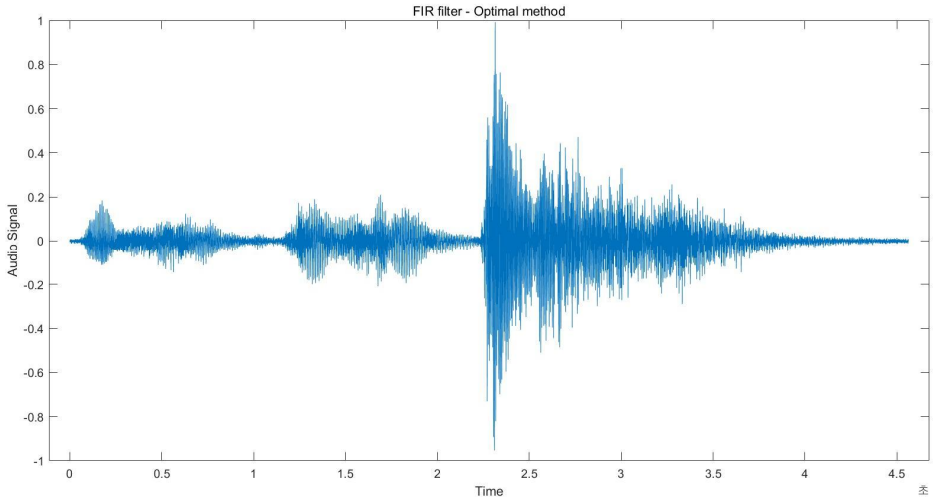

Figure 9. FIR filter (optimal design method) audio data

[objective testing]

SNR : noisy signal	SNR : FIR(optimal design method) filtered noisy signal
ans = 20.038160281204380	ans = 29.212509676416289

[subjective testing]

처음 noisy signal에 비해 noise 소리가 많이 제거되었고,
원신호처럼 선명하게 들리지는 않지만, 앞서 들어보았던 window method FIR filter보다 음질이 매우 향상되었음을 알 수 있었다.

3) IIR filter - Chebyshev Filter

$\omega_{dp} = 2\pi f_p = 2\pi(3500) = 7000\pi$ rad/sec

$\omega_{ds} = 2\pi f_s = 2\pi(4500) = 9000\pi$ rad/sec

$\omega_d = 2\pi f = 2\pi(4000) = 8000\pi$ rad/sec

$$T = \frac{1}{f_s} = \frac{1}{44100} \text{ sec}$$

$$\omega_{ap} = \frac{2}{T}\tan\left(\frac{\omega_p T}{2}\right) = 88200 \times \tan\left(\frac{7000\pi/44100}{2}\right) = 2.2458 \times 10^4 \text{ rad/sec}$$

$$\omega_{as} = \frac{2}{T}\tan\left(\frac{\omega_s T}{2}\right) = 88200 \times \tan\left(\frac{9000\pi/44100}{2}\right) = 2.9284 \times 10^4 \text{ rad/sec}$$

$$\omega_a = \frac{2}{T}\tan\left(\frac{\omega_d T}{2}\right) = 88200 \times \tan\left(\frac{8000\pi/44100}{2}\right) = 2.5133 \times 10^4 \text{ rad/sec}$$

using low-pass prototype specifications,

$$\nu_s = \frac{w_{as}}{w_{ap}} = 1.3039$$

$A_s = 20\text{dB}$ 이고,

$\varepsilon^2 = 10^{0.1A_p} - 1 = 10^{0.1\times 1} - 1 = 0.2589$ 이므로,

$$\frac{10^{0.1A_s} - 1}{\varepsilon^2} = \frac{99}{0.2589} = 382.3870$$

$$\left(\frac{10^{0.1A_s} - 1}{\varepsilon^2}\right)^{0.5} = \sqrt{382.3870} = 19.5547$$

$$n = \frac{\cosh^{-1}(19.5547)}{\cosh^{-1}(\nu_s)} = \frac{3.6657}{0.7611} = 4.8163$$

따라서 prototype filter order로 $n = 5$를 선택한다.

[code : IIR filter - Chebyshev Filter (1 dB passband ripple)]

```
close all; clear all;
[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

F = abs(fft(sig))/n;
noiseF = abs(fft(noise))/n;
Noisy_sigF = abs(fft(noisy_sig))/n;

Fc=4000/fs; % (cut-off frequency/folding frequency)
deltaf=(4500-3500)/fs;

format long
[B A]=lp2lp([0.1228],[1 0.9368 1.6888 0.9744 0.5805 0.1228],2.5133*10^4);
%n=5 IIR Chebyshev filter design (1dB ripple)
[b a]=bilinear(B,A,fs); %digital filter
[h,freq]=freqz(b,a,512,fs);
p=180*unwrap(angle(h))/pi;

iir_noise = filtfilt(b,a,noise);
iir_cheb = filtfilt(b,a,noisy_sig);
snr(iir_cheb,iir_noise)
audiowrite('iir_chebyshev.wav',iir_cheb,fs);

figure(10);
subplot(2,1,1);
plot(freq,20*log10(abs(h))); grid;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -350 20]);
subplot(2,1,2);
plot(freq,p); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -600 10]);

figure(11)
iir_optInfo = audioinfo('iir_chebyshev.wav');
t = 0:seconds(1/fs):seconds(iir_optInfo.Duration);
[chebyshev,fs] = audioread('iir_chebyshev.wav');
t = t(1:end-1);
plot(t,chebyshev);
title('IIR filter - Chebyshev');
xlabel('Time');
ylabel('Audio Signal');
```

[plot]

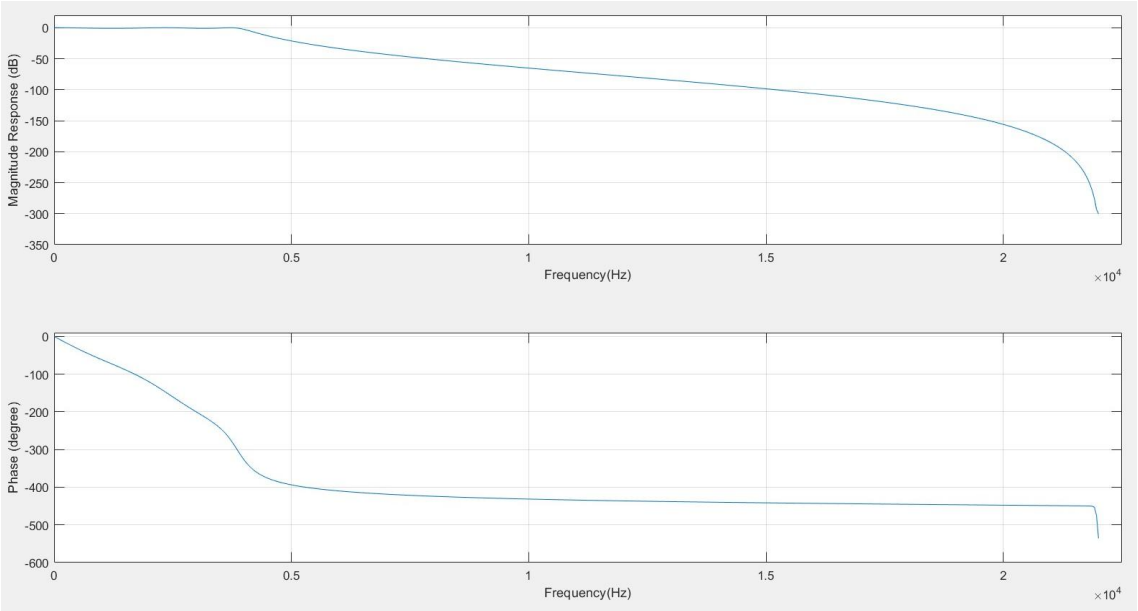

Figure 10. IIR filter (Chebyshev filter)

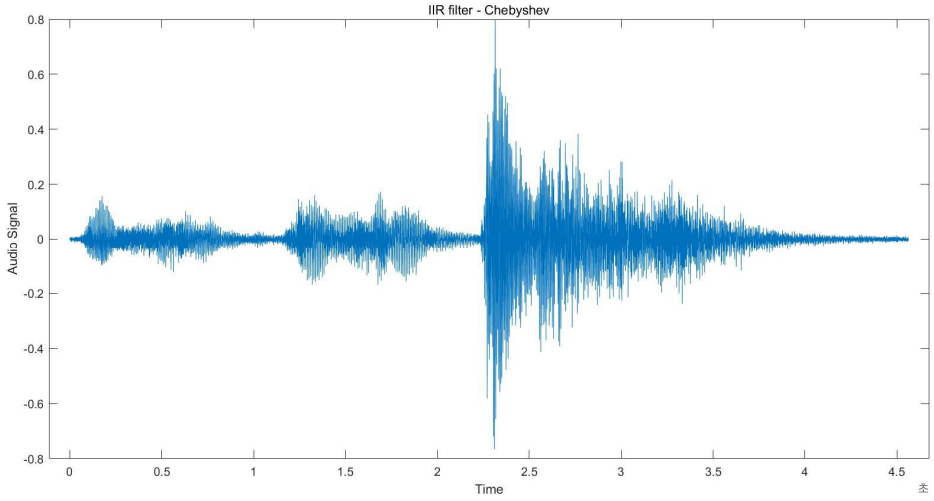

Figure 11. IIR filter (Chebyshev filter) audio signal

[objective testing]

SNR : noisy signal	SNR : IIR(Chebyshev Filter) filtered noisy signal
ans = 20.038160281204380	ans = 28.160777269945658

[subjective testing]

음질은 FIR filter보다 전체적으로 좋았으나, noise 소리는 세 개의 filter 중 가장 잘 들렸다.

4) Decide best filter

best filter를 결정하기 위해선 objective testing (snr), subjective testing, cost를 모두 고려하여 선택해야 한다.

Objective testing의 관점에서 본다면, window method를 사용한 filtered signal의 snr이 가장 높았다. 이 snr을 측정할 때 delay를 고려해 filtfilt 함수를 사용하여 이를 보정해주었다.

Subjective testing의 관점에서 본다면, window method는 noise 소리가 가장 잘 제거된 것 같지만 원신호와 비교했을 때 소리가 많이 뭉개졌고, optimal design method는 noise도 잘 제거되었고 window method를 사용했을 때보다 현저히 음질이 좋았다.
Chebyshev filter는 optimal design method를 사용했을 때보다 음질이 약간 좋아졌다는 느낌을 받았으나 noise가 optimal design method에 비해 잘 들렸다.

Cost의 측면에서 본다면 IIR filter가 FIR filter보다 계산량이 적기 때문에 비용적인 측면에서 유리하다. 또한 window method와 optimal design method를 비교하면, 같은 조건에 window method는 N=41, optimal design method는 N=29로, optimal design method가 더 비용이 적게 든다.

window method는 subjective testing을 할 때 소리가 너무 뭉개졌기 때문에, filter 조건은 잘 맞추어졌으나 좋은 filter라고 할 수 없었고, cost의 측면에서도 가장 좋지 않았다.
Chebyshev filter는 비용적인 측면에서는 유리했으나 subjective testing을 할 때 셋 중 고주파 대역이 제일 잘 들리는 느낌을 받았고, noise도 세 filter 중 가장 잘 들렸다.

optimal design method는 비용적인 측면에서 window method와 비교했을 때 많이 유리하였고, objective testing으로 세 필터를 비교하였을 때 SNR 값이 좋았다. 또한 subjective testing을 할 때 Chebyshev filter보다 noise가 잘 제거된 느낌을 받았고 window method보다 소리가 깔끔하였다.

따라서 가격 효율성과 성능을 모두 고려해봤을 때, 설계한 filter specification 조건 하에 optimal design method를 사용한 filter를 best filter로 결정하게 되었다.

5. Conclusion

이번 프로젝트에선, noise를 생성하여 original signal에 더하여 noisy signal을 만들어 보았고, signal spectrum을 분석하여 passband와 stopband 값을 정하였다.
이 때 filter가 sharp 해지면 coefficient의 개수가 증가하기 때문에 비용이 많이 든다는 점을 중요하게 고려하여 transition band를 넓게 조정하여 설계하였다.
그 후 optimal design method를 통해 임시로 filter를 설계하여 ripple 값과 attenuation 값을 각각 바꿔보면서 차이점을 크게 느끼지 못할 때까지 반복해서 들어보았고, 이를 통해 cost와 성능의 관계를 알아보며 비용의 효율성에 대해 알아볼 수 있었다.
filter 조건을 정한 후, 여러 가지의 필터를 설계해보며 각각의 필터를 거친 noisy signal에 대해 objective testing (snr), subjective testing, cost를 고려하였고 이 세 가지를 모두 고려하여 best filter를 결정하였다.
이번 프로젝트를 통해 FIR filter와 IIR filter의 특성에 대해 깊이 고찰할 수 있었다.
frequency response를 분석했을 때, FIR filter의 phase plot은 선형 위상을 가지고 있어 왜곡에 강하고 stable함을 알 수 있었고, IIR filter의 phase plot은 비선형 위상을 가

지고 있어 왜곡에 약하며 FIR filter보다 stable하지 않다는 것을 알 수 있었다.

또한 필터를 설계했을 때 FIR filter는 IIR filter에 비하여 coefficient의 개수가 많아 계산량이 상대적으로 많았는데, 이는 비용적 측면과 연관된다.

필터를 설계할 때 비용적인 측면을 중요하게 고려하여 filter의 조건을 transition band가 sharp하지 않게 설정하였고, 더불어 passband ripple을 크게, stopband attenuation을 작게 결정해서 coefficient의 개수가 적게 필요한 필터 조건을 결정하였다.

필터의 coefficient 개수가 적다면, 같은 filter 조건에서 FIR과 IIR의 비용 차이가 크지 않기 때문에 상대적으로 안정한 FIR filter를 사용하는 것이 좋다는 결론을 얻을 수 있었다.

만약 필터의 coefficient 개수를 많이 필요로 하는 필터 조건으로 결정하여 설계했더라면, 같은 조건 하에서 계산량 차이가 상대적으로 많이 발생하기 때문에 비용 차이 또한 많이 발생할 것이다. 그렇기 때문에 이 경우에는 비용적인 측면을 고려하여 계산량이 적은 IIR filter를 사용하는 것이 좋을 것이다.

따라서 optimal design method를 best filter로 결정하게 되었는데, 이 프로젝트를 통해 설계의 조건에 따라 각 filter의 특성에 맞게 적절한 filter를 선택해야 한다는 점을 배울 수 있었다.

Choosing Between FIR and IIR Filters

	FIR	IIR
Phase	Linear	Nonlinear
Stability	Always	Not always
Wordlenth effect	Less severe	Severe
Number of Coefficient	Large	Small
Analog counterpart	No	Yes
Cutoff frequency	Smooth	Sharp

Figure 12. Choosing Between FIR and IIR Filters

[전체 코드]

```
clear all; close all;
%% Load 'No9seg.wav'

[y,fs] = audioread('No9seg.wav');
fs
sig = (y(:,1)+y(:,2))/2;
T=1/fs;
n=length(y);
t=[0:1:n-1]*T;
f=[1:1:n/2]*fs/n;

%% Original signal

F = abs(fft(sig))/n;

figure(1);
plot(t,sig); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

figure(2);
plot(f,2*F(2:n/2+1)); grid; axis([0 25000 0 0.026]);
xlabel('Frequency(Hz)'); ylabel('Spectrum');
```

```
%% Noise, Noisy_sig generation

rng(8287);
noise=randn([n,1])*sqrt(1/100*var(sig));
noisy_sig=sig+noise;
snr(noisy_sig,noise)

noiseF = abs(fft(noise))/n;
Noisy_sigF = abs(fft(noisy_sig))/n;

figure(3);
plot(t,noise); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

figure(4);
plot(t,noisy_sig); hold on; grid on;
xlabel('Time(sec)'); ylabel('magnitude');

figure(5);
plot(f,2*Noisy_sigF(2:n/2+1)); grid; axis([0 25000 0 0.026]);
xlabel('Frequency(Hz)'); ylabel('Spectrum');

audiowrite('noisy_sig.wav',noisy_sig,fs);
```

```
%% Design FIR filter - window method

Fc=4000/fs; % (cut-off frequency/folding frequency)
deltaf=(4500-3500)/fs;

N=round(0.9/deltaf); %determine N
if mod(N,2)==0
    N=N+1 % N 홀수로 보정
else
    N
end

Brec = fir1(N-1,Fc,boxcar(N));
[hrec,f]=freqz(Brec,1,512,fs);
prec=180*unwrap(angle(hrec))/pi;
fir_noise = filtfilt(Brec,1,noise);
fir_rect = filtfilt(Brec,1,noisy_sig);
snr(fir_rect,fir_noise)
audiowrite('fir_rect.wav',fir_rect,fs);

figure(6);
subplot(2,1,1);
plot(f,20*log10(abs(hrec)));
grid; axis([0 4000 -100 10]); grid on;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -100 5]);
subplot(2,1,2);
plot(f,prec); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -450 5]);

figure(7)
fir_rectInfo = audioinfo('fir_rect.wav');
t = 0:seconds(1/fs):seconds(fir_rectInfo.Duration);
[rect,Fs] = audioread('fir_rect.wav');
t = t(1:end-1);
plot(t,rect);
title('FIR filter - window method');
xlabel('Time');
ylabel('Audio Signal');
```

```
%% Design FIR filter - Optimal method

N=29; % (주어진 조건에 최대한 만족하는 N 값 설정)
f_edge=[0 0.1587 0.2041 1]; %edge frequencies
m=[1 1 0 0]; %ideal magnitudes
w=[100 122]; %error weight factors
b=firpm(N,f_edge,m,w);
format long
[h,freq]=freqz(b,1,512,fs); %determine frequency response
p=180*unwrap(angle(h))/pi; %determine phase response
fir_noise = filtfilt(b,1,noise);
fir_od = filtfilt(b,1,noisy_sig);
snr(fir_od,fir_noise)
audiowrite('fir_optimaldesign.wav',fir_od,fs);

figure(8);
subplot(2,1,1);
plot(freq,20*log10(abs(h))); grid;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -80 5]);
subplot(2,1,2);
plot(freq,p); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -650 5]);

figure(9)
fir_optInfo = audioinfo('fir_optimaldesign.wav');
t = 0:seconds(1/fs):seconds(fir_optInfo.Duration);
[opt,fs] = audioread('fir_optimaldesign.wav');
t = t(1:end-1);
plot(t,opt);
title('FIR filter - Optimal method');
xlabel('Time');
ylabel('Audio Signal');
```

```
%% Design IIR filter - Chebyshev

format long
[B A]=lp2lp([0.1228],[1 0.9368 1.6888 0.9744 0.5805 0.1228],2.5133*10^4);
%n=5 IIR Chebyshev filter design (1dB ripple)
[b a]=bilinear(B,A,fs); %digital filter
[h,freq]=freqz(b,a,512,fs);
p=180*unwrap(angle(h))/pi;

iir_noise = filtfilt(b,a,noise);
iir_cheb = filtfilt(b,a,noisy_sig);
snr(iir_cheb,iir_noise)
audiowrite('iir_chebyshev.wav',iir_cheb,fs);

figure(10);
subplot(2,1,1);
plot(freq,20*log10(abs(h))); grid;
xlabel('Frequency(Hz)'); ylabel('Magnitude Response (dB)'); axis([0 22500 -350 20]);
subplot(2,1,2);
plot(freq,p); grid;
xlabel('Frequency(Hz)'); ylabel('Phase (degree)'); axis([0 22500 -600 10]);

figure(11)
iir_optInfo = audioinfo('iir_chebyshev.wav');
t = 0:seconds(1/fs):seconds(iir_optInfo.Duration);
[chebyshev,fs] = audioread('iir_chebyshev.wav');
t = t(1:end-1);
plot(t,chebyshev);
title('IIR filter - Chebyshev');
xlabel('Time');
ylabel('Audio Signal');
```