임베디드 신호처리 실습
결과리포트 LAB9

전자공학부 임베디드시스템
2014146012 박동훈
2016146048 한대성
김수민 교수님

1.

3.1 Audio file 입력 및 재생

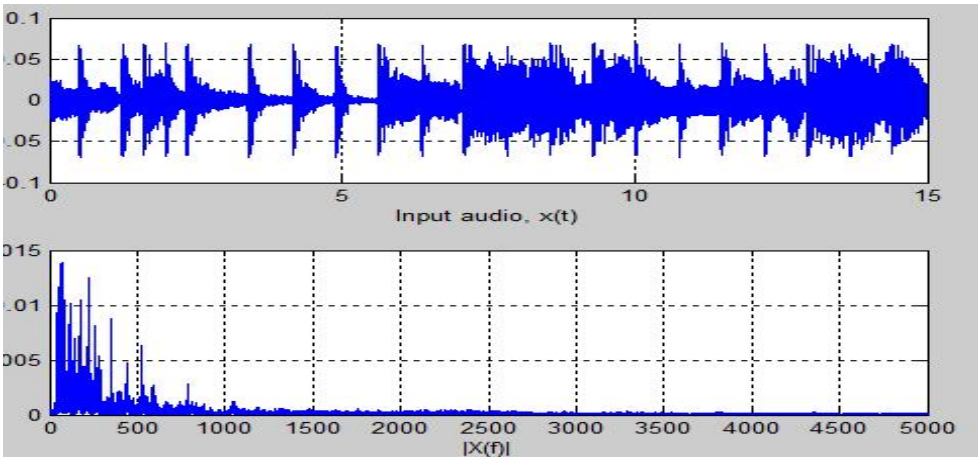
(1) 실습 제공되는 샘플 오디오 파일을 Matlab에서 읽어들여 스펙트럼을 분석하라. (audioread, myfun_SA 사용)

(2) 실습 DEMO 샘플 오디오 신호와 이 신호의 크기 스펙트럼을 그래프에 표시하라. (그림 3 참고)

- **코드**

```
---------------------------------------------------------------------------
[y, fs]=audioread('Audio_Pop01_15sec.wav'); % 오디오 파일 입력

f = linspace(0, 15, length(y));

[f0, X] = myfun_SA(f,y); % 주파수 영역 변환 내장함수

figure(1)
subplot(211); % 오디오 신호의 스펙트럼
plot(f, y);
grid on;
xlabel('Input audio, x(t)'); ylabel('Time[sec]');

subplot(212); % 신호의 크기 스펙트럼
plot(f0, abs(X));
grid on;
xlim([0 5000]);
xlabel('|X(f)|'); ylabel('Frequency[Hz]');

sound( y, fs );
---------------------------------------------------------------------------
```

- 결과 그래프



오디오 신호를 크기 스펙트럼으로 보아 위의 오디오 신호의 대다수의 주파수가 500Hz이하에 머무르는 것으로 저주파 성분인 것을 확인했다.
인간의 귀로 소리의 주파수를 들을 때 주파수가 클수록 고음, 작을수록 저음인 것으로 정의하는데 위의 오디오 신호를 sound() 함수를 통해 들었을 때 저음인 것을 확인할 수 있었다.

2.

3.2 디지털 audio equalizer 설계

(1) 실습 다음과 같은 사양의 IIR Butterworth 필터를 설계하라. (butter 사용)

- (Ch-1) 7차 LPF, Cutoff 주파수 500Hz
- (Ch-2) 12차 BPF, 통과대역 500Hz ~ 1500Hz
- (Ch-3) 12차 BPF, 통과대역 1500Hz ~ 2500Hz
- (Ch-4) 12차 BPF, 통과대역 2500Hz ~ 3500Hz
- (Ch-5) 10차 HPF, Cutoff 주파수 3500Hz

(2) 실습 DEMO 위에서 설계한 다섯 필터의 주파수 응답의 크기와 pole-zero plot을 그래프에 표시하라. (그림 4.5 참고)

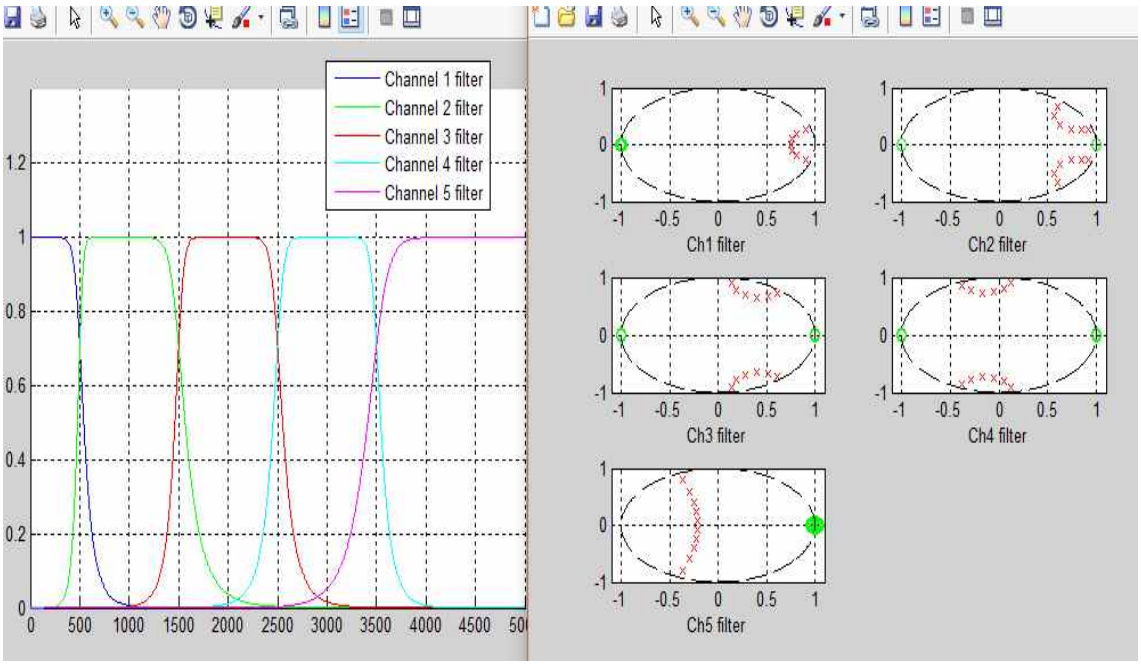
- 코드

```
[y, f_s]=audioread('Audio_Pop01_15sec.wav');

n = linspace(-pi,pi,360); % 단위원 범위 설정
w = linspace(0, 5000*2*pi, 10000);
w2 = w/2/pi;

fs = f_s/2; % butterWorth 필터는 [0 1] 도메인을 사용한다.
```

```
w3 = w/2/fs;
wn = [500/fs 1500/fs];
wn2 = [1500/fs 2500/fs];
wn3 = [2500/fs 3500/fs];

[num, den] = butter(7, 500/fs, 'low'); % 디지털 Butterworth 필터
[num2, den2] = butter(6, wn, 'bandpass'); % BPF 는 차수가 2N 임으로 12 >> 6
[num3, den3] = butter(6, wn2, 'bandpass');
[num4, den4] = butter(6, wn3, 'bandpass');
[num5, den5] = butter(10, 3500/fs, 'high');% HPF

db = freqz(num, den, w3); % 디지털 시스템의 주파수 응답
db2 = freqz(num2, den2, w3);
db3 = freqz(num3, den3, w3);
db4 = freqz(num4, den4, w3);
db5 = freqz(num5, den5, w3);

figure(2) % 주파수 응답
hold on; grid on;
plot(w2, abs(db), 'b');
plot(w2, abs(db2), 'g');
plot(w2, abs(db3), 'r');
plot(w2, abs(db4), 'c');
plot(w2, abs(db5), 'm');
ylim([0 1.4]);
legend('Channel1 filter','Channel 2 filter','Channel 3 filter','Channel
4 filter','Channel 5 filter');

[z, p, k] = tf2zp(num, den); % 전달함수를 이용하여 pole-zero 구현
[z2, p2, k2] = tf2zp(num2, den2);
[z3, p3, k3] = tf2zp(num3, den3);
[z4, p4, k4] = tf2zp(num4, den4);
[z5, p5, k5] = tf2zp(num5, den5);

figure(3) % pole-zero plot
subplot(321);
plot(real(z), imag(z) , 'go');
hold on; grid on;
plot(real(p), imag(p) , 'rx');
plot(cos(n), sin(n), '--k'); xlim([-1.1 1.1]);
xlabel('Ch1 filter');
```

```
subplot(322);
plot(real(z2), imag(z2) , 'go');
hold on; grid on;
plot(real(p2), imag(p2) , 'rx');
plot(cos(n), sin(n), '--k'); xlim([-1.1 1.1]);
xlabel('Ch2 filter');

subplot(323);
plot(real(z3), imag(z3) , 'go');
hold on; grid on;
plot(real(p3), imag(p3) , 'rx');
plot(cos(n), sin(n), '--k'); xlim([-1.1 1.1]);
xlabel('Ch3 filter');

subplot(324);
plot(real(z4), imag(z4) , 'go');
hold on; grid on;
plot(real(p4), imag(p4) , 'rx');
plot(cos(n), sin(n), '--k'); xlim([-1.1 1.1]);
xlabel('Ch4 filter');

subplot(325);
plot(real(z5), imag(z5) , 'go');
hold on; grid on;
plot(real(p5), imag(p5) , 'rx');
plot(cos(n), sin(n), '--k'); xlim([-1.1 1.1]);
xlabel('Ch5 filter');
--------------------------------------------------------------------------------
```

- 결과 그래프



ploe-zero plot으로 cutoff 주파수가 커짐에 따라 각주파수 값이 변화하기 때문에 LPF일때는 저주파수쪽, HPF는 고주파수쪽, BPF는 구간별로 pole값이 위침함을 확인했다. 즉, 극점값이 우측에서 좌측으로 이동하는 것을 확인했다.
중간 채널 (ch. 2~4)은 cutoff 주파수가 두 개의 BUF로 설정이 되어있기 때문에 극점이 대칭되어 두 곳으로 나오는 것을 확인했다.
pole값이 단위원 내부에 존재함으로 안정성을 확인했다.

3.

3.3 디지털 audio equalizer 동작 확인

(1) 실습 실습 3.1의 입력 신호 $x[n]$을 실습 3.2에서 설계한 다섯개의 필터에 입력하여 각 채널의 출력 $y_1[n]$, $y_2[n]$, $y_3[n]$, $y_4[n]$, $y_5[n]$을 구하고 각 신호의 스펙트럼을 구하라. (filter 사용)

(2) 실습 DEMO 위에서 구한 각 필터의 출력과 크기 스펙트럼을 그래프에 표시하라. (그림 6 참고)

(3) 실습 각 필터의 출력을 재생하여 소리를 확인하라.

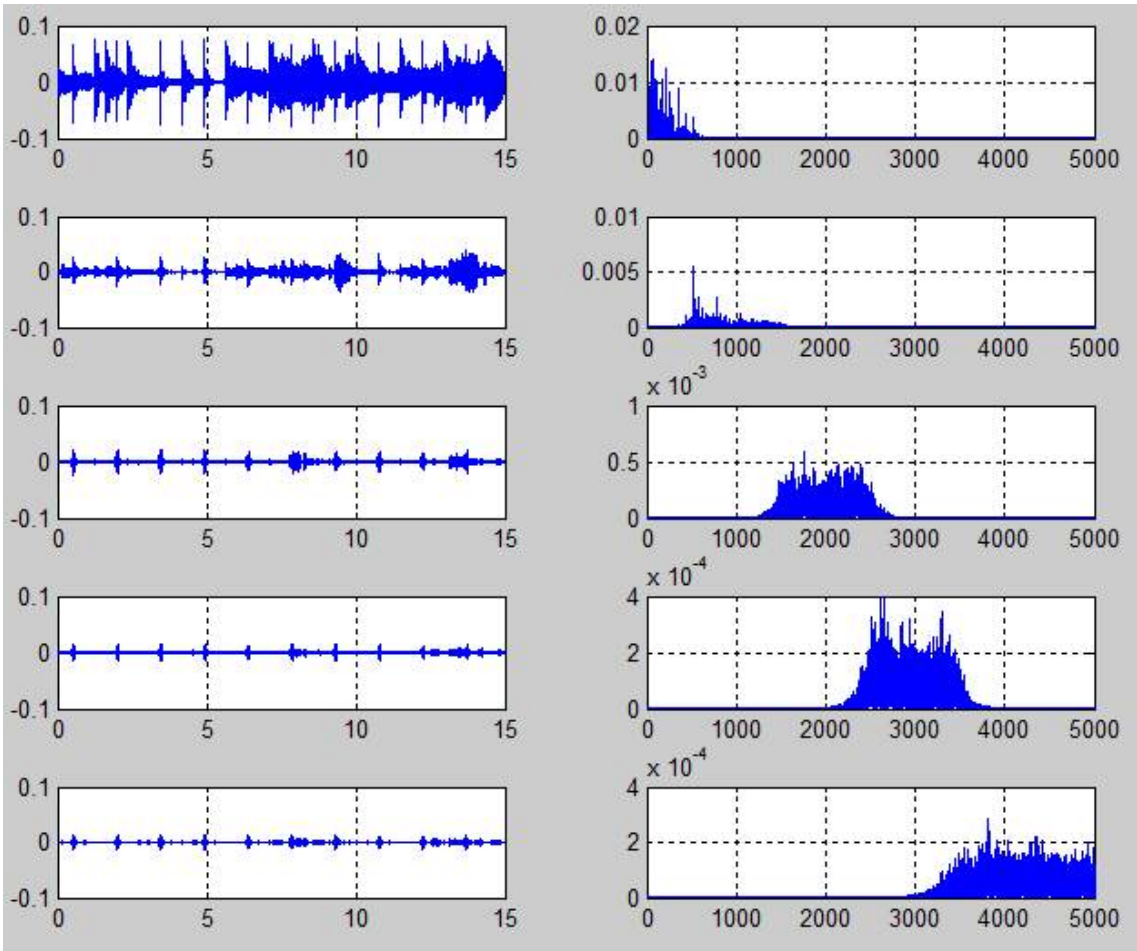
- 코드

```
[y, f_s]=audioread('Audio_Pop01_15sec.wav');
% 음성 파일을 읽어 시간축 y와 샘플링 주파수 f_s 추출
```

```
t = linspace(0, 15, length(y)); % 0부터 15까지 신호의 길이
fs = f_s/2; % butterWorth 필터는 [0 1] 도메인을 사용한다.

wn = [500/fs 1500/fs];
wn2 = [1500/fs 2500/fs];
wn3 = [2500/fs 3500/fs];

[num, den] = butter(7, 500/fs, 'low');
[num2, den2] = butter(6, wn, 'bandpass'); % BPF 는 차수가 2N 이므로 12 >> 6
[num3, den3] = butter(6, wn2, 'bandpass');
[num4, den4] = butter(6, wn3, 'bandpass');
[num5, den5] = butter(10, 3500/fs, 'high');

Y1 = filter(num, den, y);   [f, x] = myfun_SA(t, Y1); %디지털시스템의출력
Y2 = filter(num2, den2, y); [f2, x2] = myfun_SA(t, Y2); %디지털시스템의출력
Y3 = filter(num3, den3, y); [f3, x3] = myfun_SA(t, Y3); %디지털시스템의출력
Y4 = filter(num4, den4, y); [f4, x4] = myfun_SA(t, Y4); %디지털시스템의출력
Y5 = filter(num5, den5, y); [f5, x5] = myfun_SA(t, Y5); %디지털시스템의출력

F = [f; f2; f3; f4; f5];
X = [x x2 x3 x4 x5];
Y = [Y1 Y2 Y3 Y4 Y5];
k = 0; j = 0;

figure(4)
for i = 1:10
    subplot(5,2,i);

    if(rem(i,2) == 0)
        k = k + 1;
        plot(F(k, :), abs(X(:,k)));
        xlim([0 5000]);grid on;
    end

    if(rem(i,2) == 1)
        j = j + 1;
        plot(t, Y(:,j));
        ylim([-0.1 0.1]);
        grid on;
    end
```

```
end
% subplot(5,2,1)
% sound(Y1,f_s);

% subplot(5,2,3)
% sound(Y2,f_s);

% subplot(5,2,5)
% sound(Y3,f_s);

% subplot(5,2,7)
% sound(Y4,f_s);
```

-결과 그래프



5가지 주파수 대역을 시간 축과 크기 스펙트럼에 대하여 나타내주었다.
1번째 채널은 일반적으로 재생시킨 오디오 파일과 동일하게 나왔다.
2번째 채널은 주파수 구간이 약간 고주파 영역으로 옮겨가서 소리가 조금은 작지만 그래도 잘 들렸다.
3번째 채널은 드럼의 심벌 소리만 살짝 들렸다.
4번째 채널은 거의 소리가 들리지 않았다. 전반적으로 오디오가 저주파 구간에 있다는 것을 알 수 있다.
5번째 채널은 전체적으로 소리가 줄어들었다.

(4) **실습** 각 필터의 출력에 적당한 채널이득 c_1, c_2, c_3, c_4, c_5를 곱하여 믹싱하고 식 (3)을 적용해 믹서 출력 $y_{mix}[n]$을 구하라. c_0는 얼마인가? (그림 2 참고., 채널 이득의 합은 1이 되어야 함)

$$\sum_{i=1}^{N} c_i = 1 \qquad c_0 = \sqrt{\frac{\sum_{n=0}^{L-1} x^2[n]}{\sum_{n=0}^{L-1} y_0^2[n]}}$$

왼쪽의 식은 채널 이득의 합은 1이 되어야 한다는 식이고 오른쪽은 c0의 값을 구하는 식이다. 실습의 코드에서는 5개의 채널이득이 주어졌는데, 이 이득의 합이 1이 되도록 조정하여 c0의 값을 구해보았다. 기존의 입력 신호는 저주파 구간에 주파수가 몰려 있다. 저주파 구간에서 채널 이득이 클 경우(*C(1) = 0.6, C(2) = 0.1, C(3) = 0.1, C(4) = 0.1, C(5) = 0.1*) 입력과 출력의 전력이 차이가 크게 나지 않게 1과 비슷한 값이 나온 반면, 고주파 구간에서 채널 이득이 크면(*C(1) = 0.1, C(2) = 0.1, C(3) = 0.1, C(4) = 0.1, C(5) = 0.6*) 저주파 구간의 전력이 줄어들기 때문에 출력전력이 입력전력보다 값이 떨어지게 나왔다. 그래서 출력전력을 입력전력과 동일하게 조정하기 위해서는 더 큰 c0 값이 필요한 것이다.

(5) **실습** Master volume $w = 1$을 적용하여 최종 출력 $y[n]$을 구하고 재생하여 소리를 확인하라.

(6) **실습** **DEMO** 채널 이득이 다음과 같을 때, audio equalizer의 입력과 출력을 시간 영역과 주파수 영역에서 비교하라. (그림 7 참고)

- $c_1 = 1,\ c_2 = 5,\ c_3 = 4,\ c_4 = 2,\ c_5 = 1$
- $w = 1$

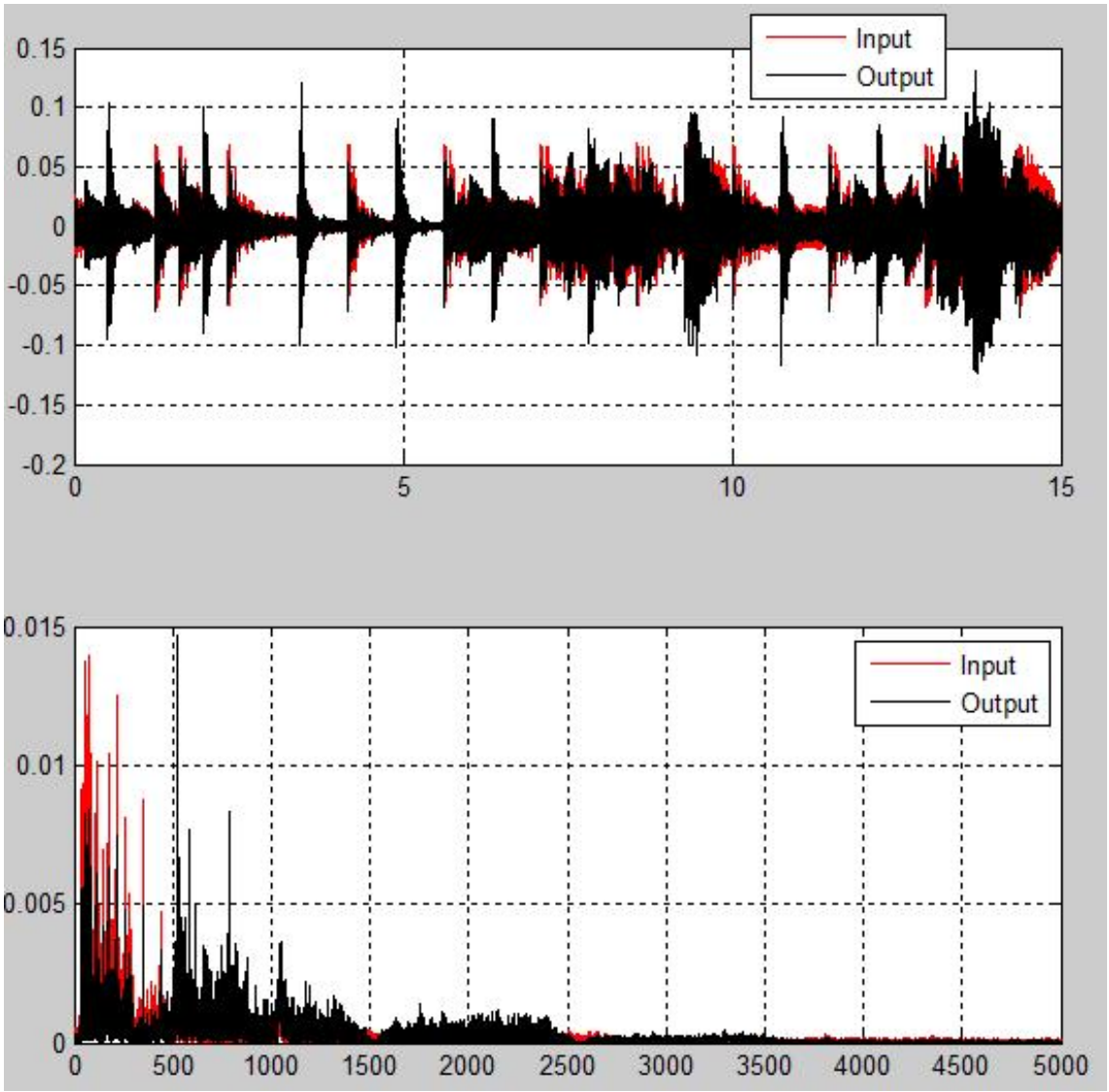
(7) **실습** 추가로 제공되는 다양한 오디오 파일을 이용해 소리를 확인하라.

- **코드**

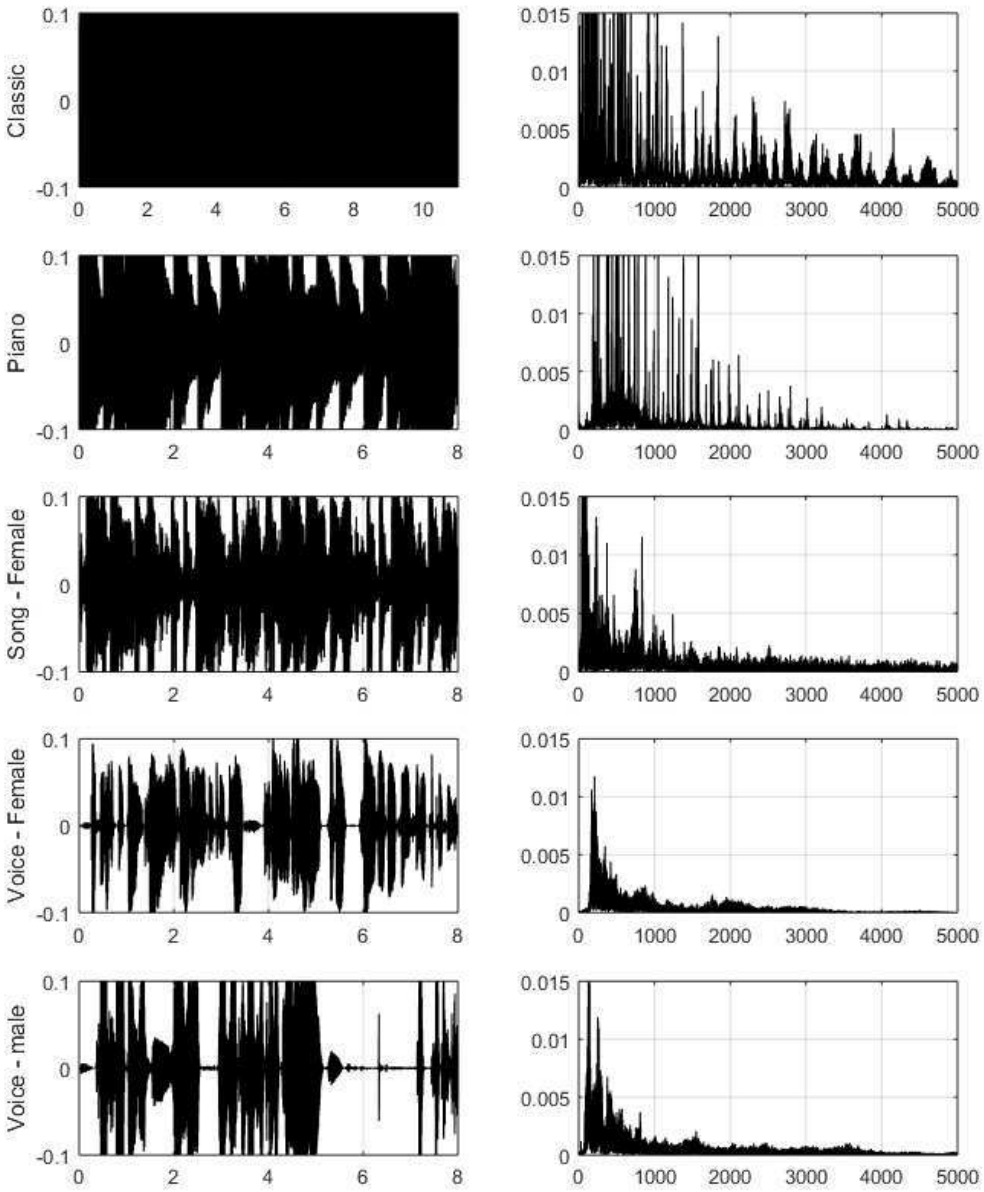
```
C = [1 5 4 2 1];
w = 1; %마스터 볼륨

Z = C(1)*Y(:,1) + C(2)*Y(:,2) + C(3)*Y(:,3) + C(4)*Y(:,4) + C(5)*Y(:,5);
%믹서출력
C0 = sqrt((sum(y.^2))/(sum(Z.^2))); %입출력 신호의 전력을 맞추기 위함

Z_mix = C0 * Z; %채널 이득을 곱하여 믹싱
yy = w * Z_mix;
[F, XX] = myfun_SA(t,y);
[F2, XX2] = myfun_SA(t,yy);

 sound(yy,f_s);

figure(5)
subplot(211);
plot(t, y, 'r');
hold on; grid on;
plot(t ,yy, 'k');
legend('Input','Output');

subplot(212);
plot(F, abs(XX), 'r');
hold on; grid on;
plot(F2, abs(XX2), 'k');
xlim([0 5000]);
legend('Input','Output');
```

c0는 입력 신호의 전력을 출력 신호의 전력으로 나눈 것에 제곱근을 씌우면 구할 수 있다. c0는 그저 가중치일 뿐이며, 출력신호와 입력신호의 전력이 동일 해지도록 조정하기 위함이다. 주어진 채널 이득과 식을 이용하여 c0를 구하면 0.6019라는 값이 도출된다.

w는 볼륨의 크기를 조절해주는 것인데, 1보다 크면 볼륨이 커지고 1보다 작으면 볼륨이 작아진다. 하지만 직접 듣기로는 볼륨의 차이를 느끼지 못했다.



Input은 기본 오디오 파일의 시간축 신호와 크기 스펙트럼이고, Output은 5가지의 채널 이득을 필터에 적용하여 시간축 신호와 크기 스펙트럼으로 나타내었다. 소리를 재생하여 확인해 보았으나 큰 차이는 없었지만, 출력 신호의 소리에 echo가 조금 있는 것 같다.



나머지 오디오 파일들도 시간축과 스펙트럼을 이용하여 분석해 보았다. 전반적으로 공통점이 있는데 대부분의 주파수가 저주파 영역에 몰려있다는 것이다. 이를통해 일반적으로 사람이 귀로 들을 수 있는 주파수 대역은 대부분 저주파 대역이라는 사실을 알 수 있다.