DSP 설계과제: 필터설계 및 C 프로그램 구현

1. MATLAB을 사용하여 FIR과 IIR 필터를 설계한다 (즉, 필터계수를 구한다).
2. MATLAB으로 구한 필터계수를 C 프로그램에서 읽어 FIR과 IIR 필터링을 구현한다.
3. MATLAB과 C 프로그램으로 수행한 결과를 MATLAB을 사용하여 비교한다.

※ 설계과제 보고서에 포함될 내용

1. MATLAB 프로그램 소스 코드
2. MATLAB 프로그램 실행 결과 및 간단 설명
3. C 프로그램 소스 코드
4. MATLAB과 C 프로그램 실행 결과 비교 및 간단 설명
5. 고찰

I. FIR 필터 설계 (차수: 64, 변경 가능)

>>edit myfir????.m

```
close all; clear; clc

% Make signal ------------------------------------
dur = 1; % [sec]
fm = 300; % frequency of message signal
fn = 3000; % frequency of noise signal
Fs = 10000; % sampling frequency
N = Fs * dur; % total number of samples
Ts = 1/Fs; % sampling period

n = 0:N-1;
xm = 0.5*sin(                    ); % message signal
xn = 0.5*sin(                    ); % noise signal
x = xm + xn; % -1.0 ~ 1.0
x = x * 32767; % -32767 ~ 32767
```

```
% Filter design ------------------------------------
M = 64; % filter order
fc = 2000; % cutoff frequency (Hz). 0 < fc < Fs/2
% normalized cutoff frequency (rad). 0 < wc < 1
wc = fc/(Fs/2);
B = fir1(    ,    , 'low'); % filter design

% write the filter coefficients to a file
fid = fopen('coeffs_fir.txt','w');
fprintf(fid,'%e₩n',B);
fclose(fid);

L = Fs*0.02; % a segment for short-time FFT (20 msec)
[H, w] = freqz(   ,    , L); % frequency response
% magnitude response
magH = abs(H);
dBH = 20*log10(magH./max(magH)); % in dB

y=filter(B,1,x); % filtering

Xk = fftshift(              ); % first L samples
Yk = fftshift(              ); % first L samples
delta_f = Fs/L;
freq_axis = delta_f*(-L/2:L/2-1); % -Fs/2 ~ (Fs/2 - delta_f)
```

```
% Plotting ------------------------------------
figure(1)
subplot(3,1,1); stem(0:M,B);grid;
title('Filter coefficients');
subplot(3,1,2); plot(w/pi,magH);grid;
title('Magnitude response');
subplot(3,1,3); plot(w/pi,dBH); grid;
title('Magnitude response in dB');
print -djpeg 'FIR_fig1.jpg'

figure(2)
subplot(2,2,1); plot(0:L-1,x(1:L));
title('Signal amplitude before filtering');
subplot(2,2,2); plot(freq_axis,abs(Xk));
title('Freq. components before filtering');
subplot(2,2,3); plot(0:L-1,y(1:L));
title('Signal amplitude after filtering');
subplot(2,2,4); plot(freq_axis,abs(Yk));
title('Freq. components after filtering');
print -djpeg 'FIR_fig2.jpg'

figure(3)
subplot(2,1,1); spectrogram(x, hamming(256), 'MinThreshold',15, 'yaxis')
title('Spectrogram before filtering');
subplot(2,1,2); spectrogram(y, hamming(256), 'MinThreshold',15, 'yaxis')
title('Spectrogram after filtering');
print -djpeg 'FIR_fig3.jpg'

fid = fopen('input.snd','wb');
fwrite(fid,x,'int16'); % write the input signal to a file
fclose(fid);
fid = fopen('output_FIR.snd','wb');
fwrite(fid,y,'int16'); % write the output signal to a file
fclose(fid);

soundsc(x,Fs); % before filtering
pause(2*N/Fs) % 2 * dur
soundsc(y,Fs); % after filtering
```

IIR 필터 설계 (차수: 8, 변경 가능)

>>edit myiir????.m

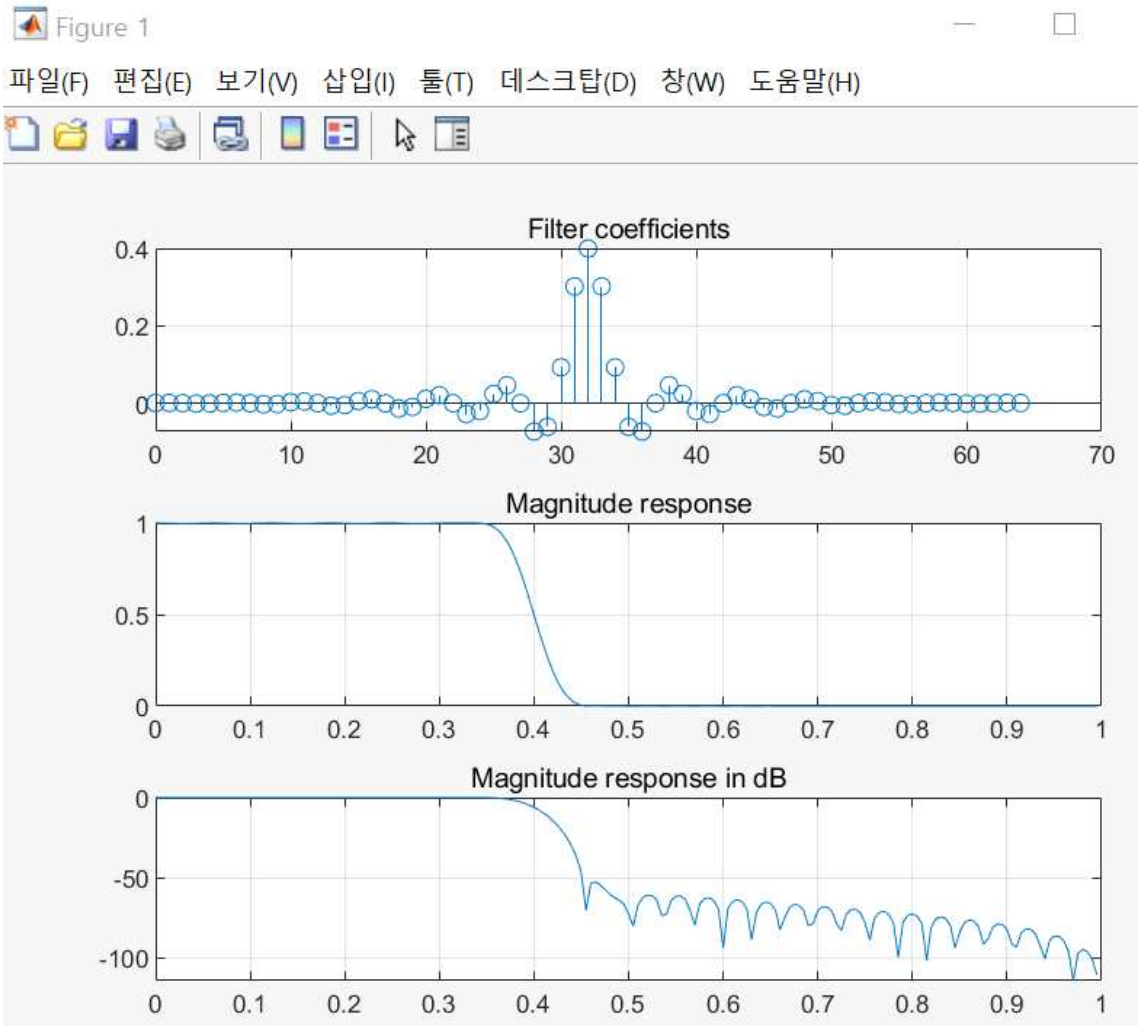
>>> FIR 필터설계 코드 앞부분과 일치 <<<

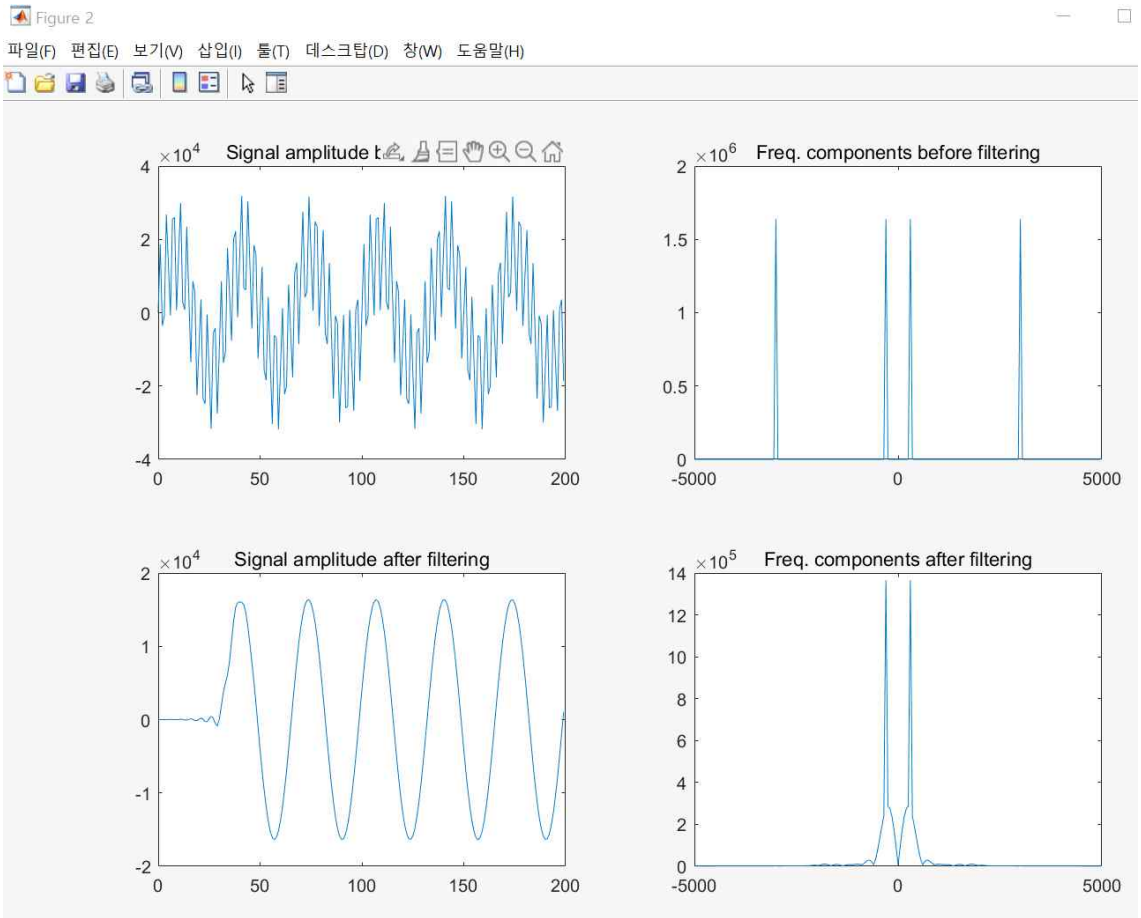
```
% Filter design ------------------------------------
M = 8; % filter order
fc = 2000; % cutoff frequency (Hz). 0 < fc < Fs/2
% normalized cutoff frequency (rad). 0 < wc < 1
wc = fc/(Fs/2);

[B, A] = cheby1(M, 0.5, wc, 'low'); % filter design

% write the filter coefficients to a file
fid = fopen('coeffs_fir.txt','w');
fprintf(fid,'%e₩n',B);
fprintf(fid,'%e₩n',A);
fclose(fid);

L = Fs*0.02; % a segment for short-time FFT (20 msec)

[H, w] = freqz(    ,    , L); % frequency response
% magnitude response
magH = abs(H);
dBH = 20*log10(magH./max(magH)); % in dB
```

>>> FIR 필터설계 코드 뒷부분과 일치 <<<

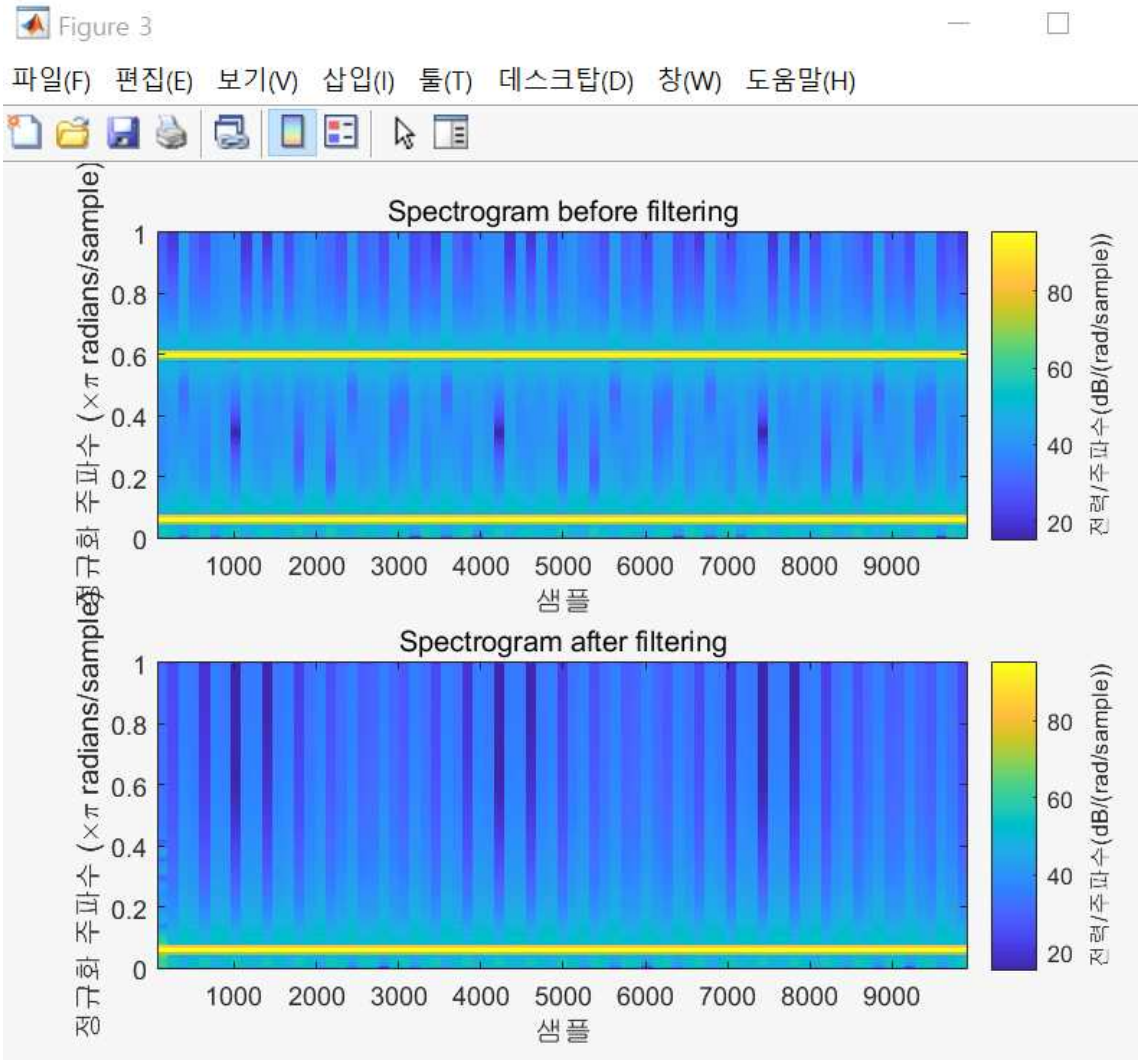
<실행 결과>

1. FIR Filter (64차)



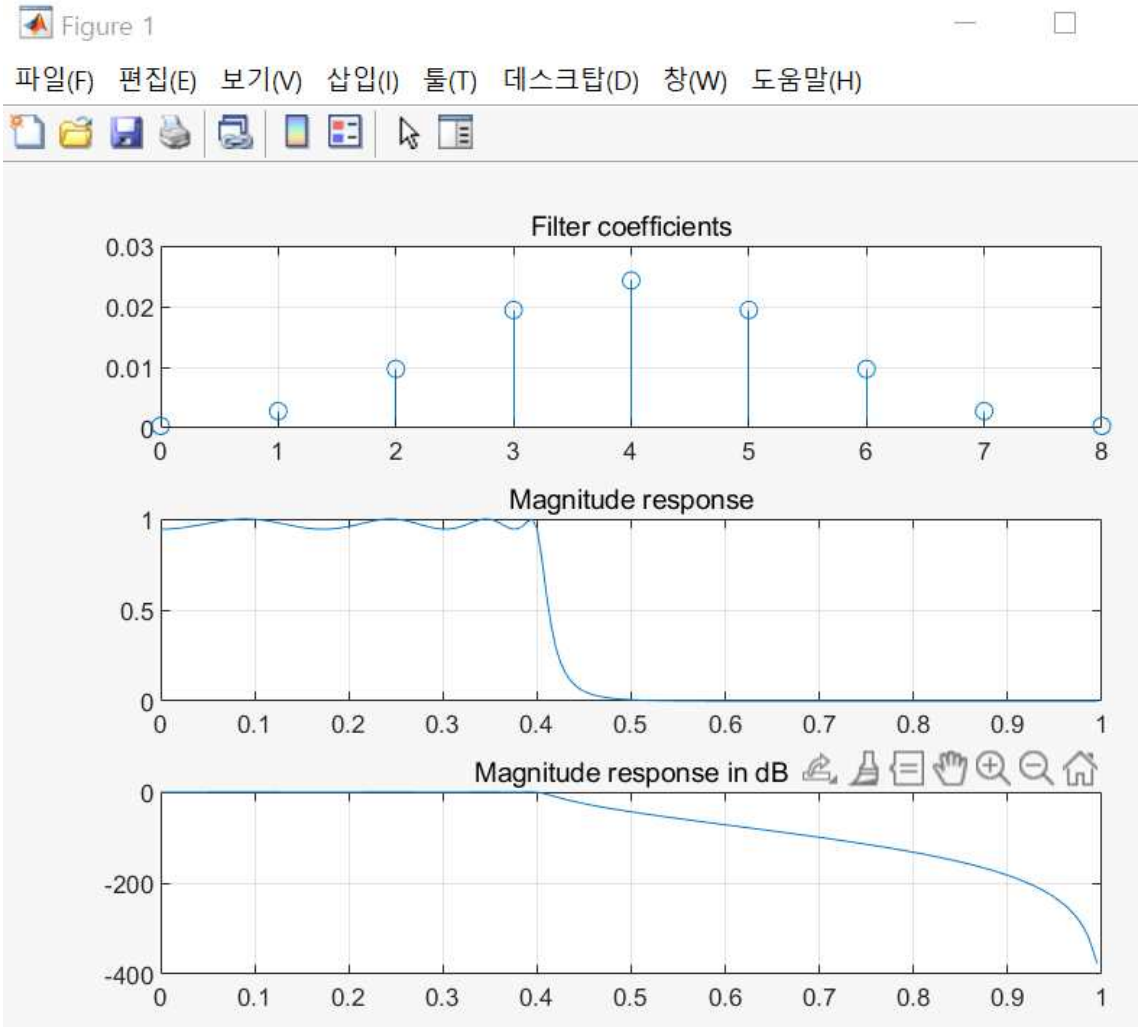


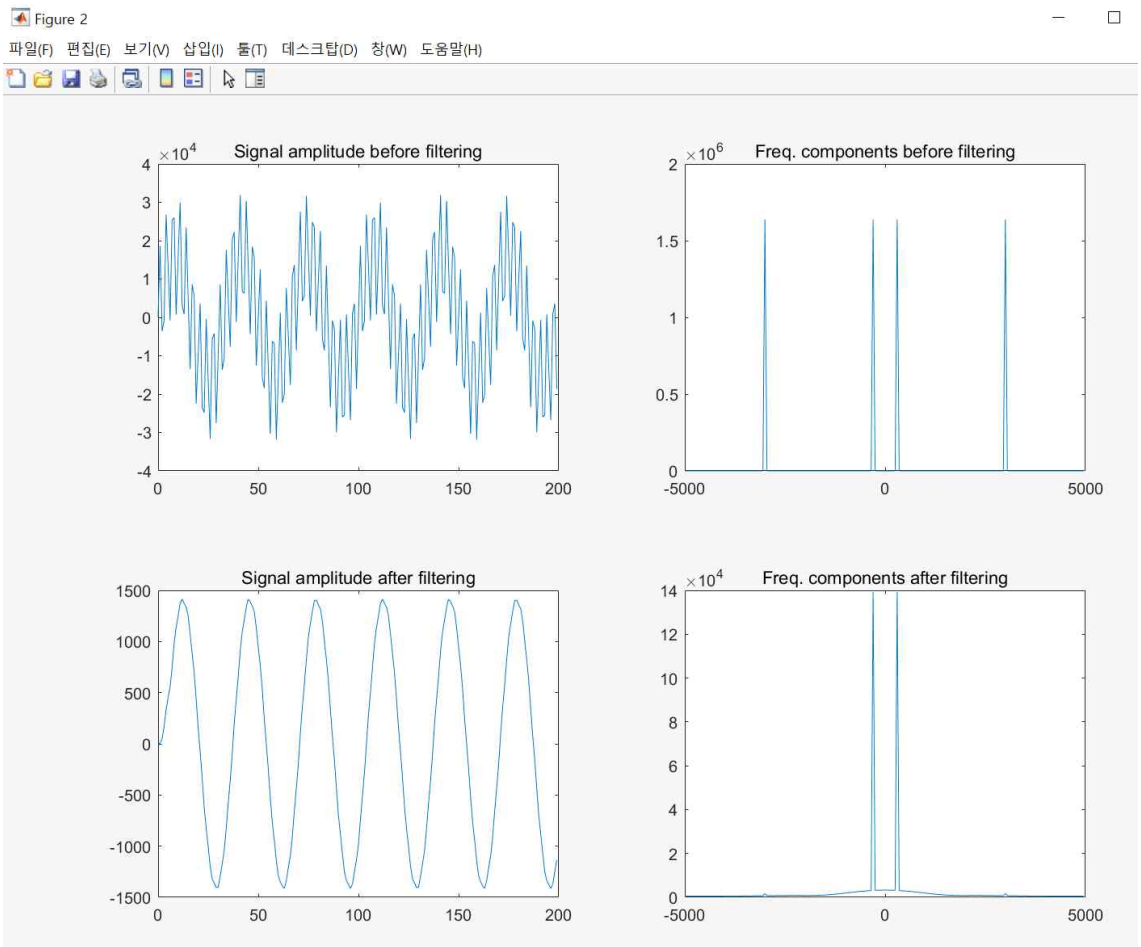
※ 저역통과필터(LPF)를 통해 고주파 신호가 제거되었음을 파형과 주파수 성분 그림을 통해 알 수 있다.



※ 저역통과필터(LPF)를 통해 고주파 신호가 제거되었음을 스펙트로그램을 통해 알 수 있다.

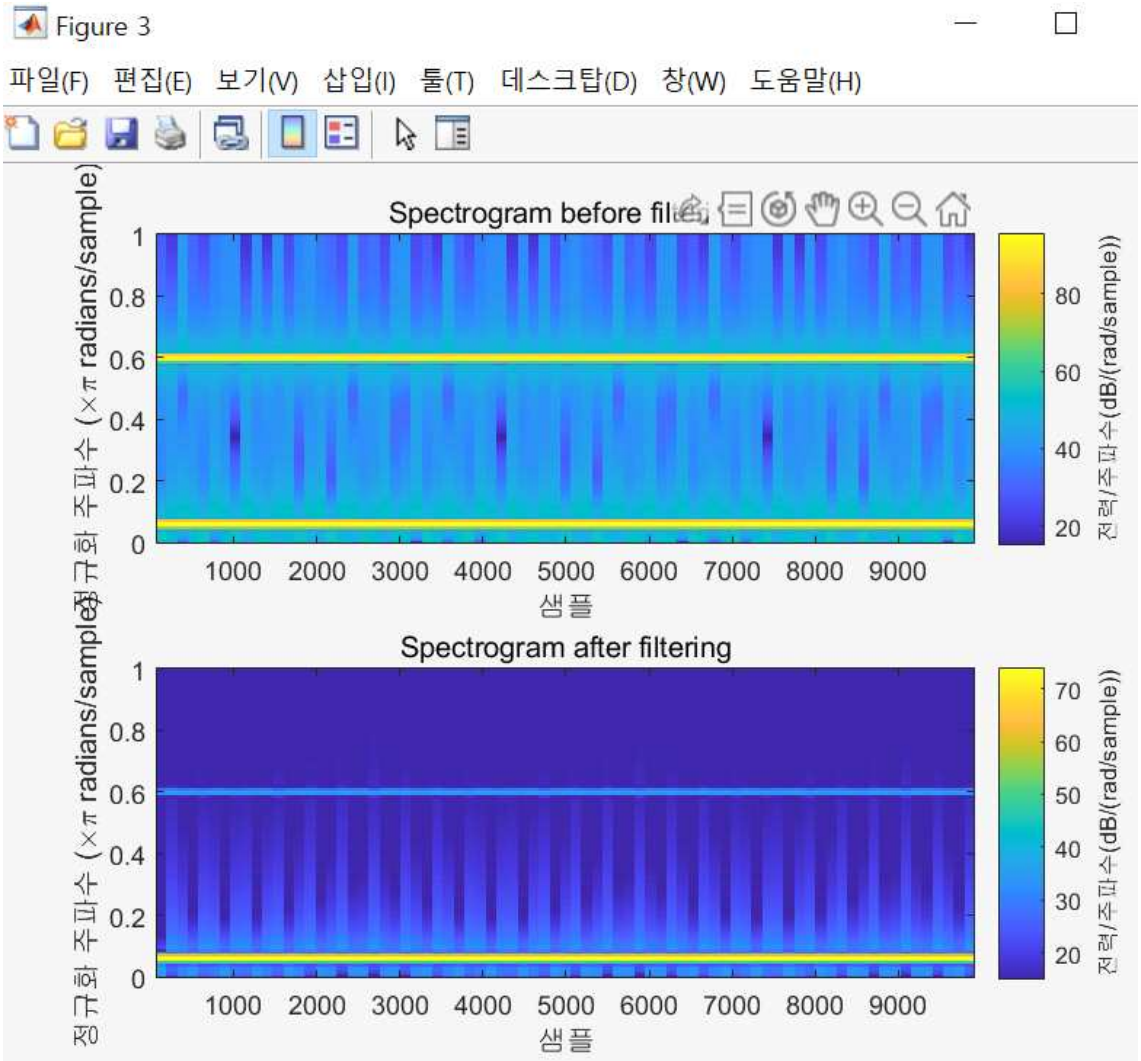
2. IIR Filter (8차)





※ 저역통과필터(LPF)를 통해 고주파 신호가 제거되었음을 파형과 주파수 성분 그림을 통해 알 수 있다.

※ FIR 필터보다 훨씬 적은 차수로 비슷한 성능을 얻었음을 알 수 있다.



※ 저역통과필터(LPF)를 통해 고주파 신호가 제거되었음을 스펙트로그램을 통해 알 수 있다.

※ C 프로그램 일부분 예시 참고

```
fopen_s(&fi, argv[1],"rb");
if(fi == NULL) {
        printf("Input file [%s] is not found. \n", argv[1]);
        getchar(); exit(-1);
}

 fopen_s(&fo, argv[2], "wb");
if(fo == NULL) {
        printf("Onput file [%s] is not found. \n", argv[2]);
       getchar(); exit(-1);
}

filter_len=atoi(argv[3]);
fopen_s(&fi_coeff, "coeffs_iir.txt", "r");

if(fi_coeff == NULL) {
        printf("Could not find the file!!! \n");
        getchar();  exit(-1);
}


for(k=0; k<filter_len; k++) {
        fscanf_s(fi_coeff, "%e", &coeffs_b[k], sizeof(float));
        printf("%e \n",coeffs_b[k]);
}

for(k=0; k<filter_len; k++) {
        fscanf_s(fi_coeff, "%e", &coeffs_a[k], sizeof(float));
        printf("%e \n",coeffs_a[k]);
}
fclose(fi_coeff);

printf("Press any key \n");
getchar();

while(1) {
```

※ MATLAB과 C 프로그램 수행 결과 비교는 다음 MATLAB 프로그램 예시 참고

```
close all; clear; clc

Fs = 10000;

fi = fopen('output1.snd','rb');
x = fread(fi, inf, 'int16');
fclose(fi);
fi = fopen('output2.snd','rb');
y = fread(fi, inf, 'int16');
fclose(fi);

L = Fs*0.02; % a segment for short-time FFT (20 msec)

figure(1)
subplot(2,1,1); plot(x(1:L));
subplot(2,1,2); plot(y(1:L));

figure(2)
subplot(2,1,1); spectrogram(x, hamming(256), 'yaxis')
subplot(2,1,2); spectrogram(y, hamming(256), 'yaxis')
```