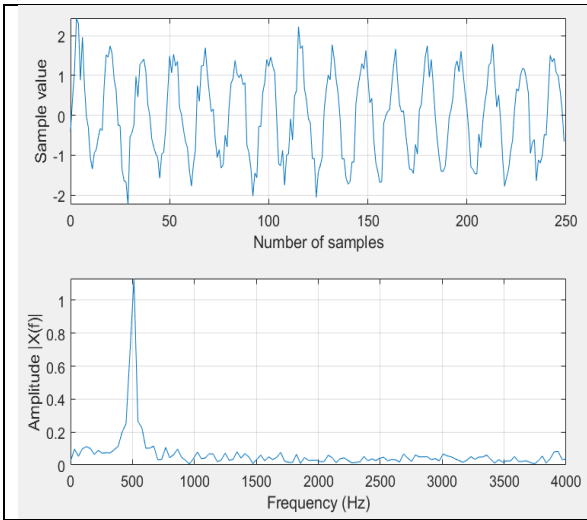
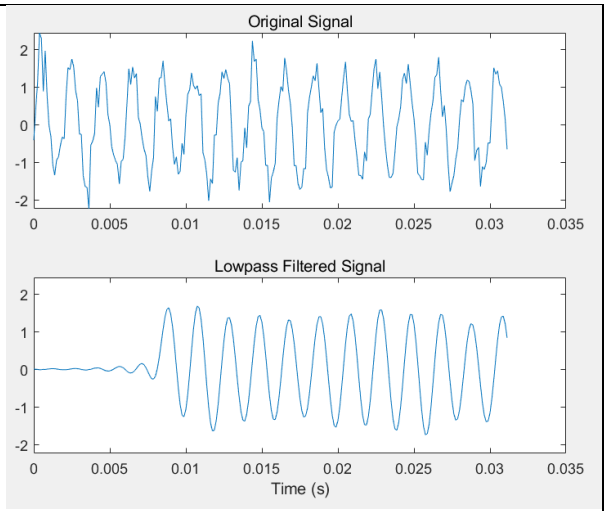
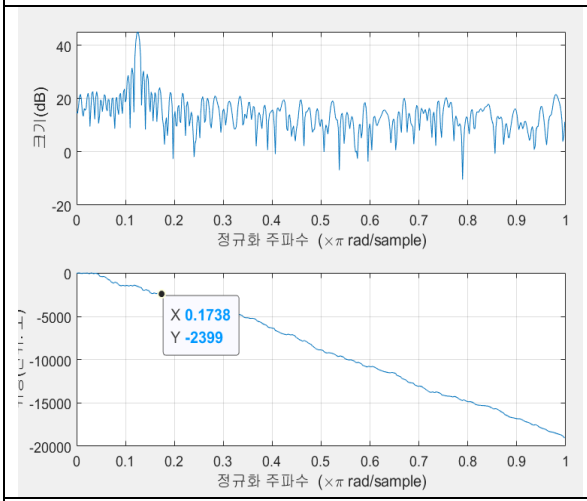
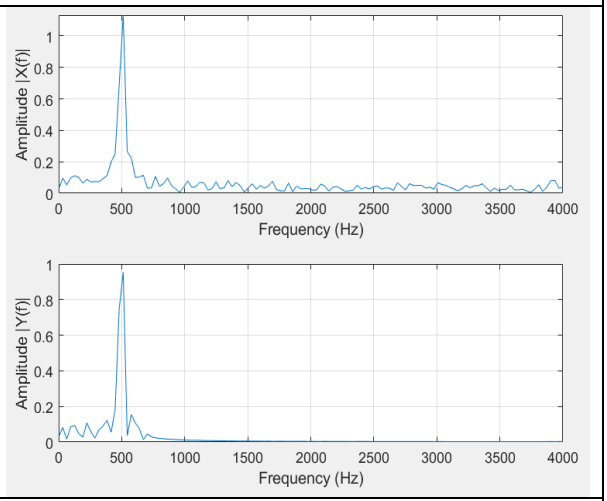
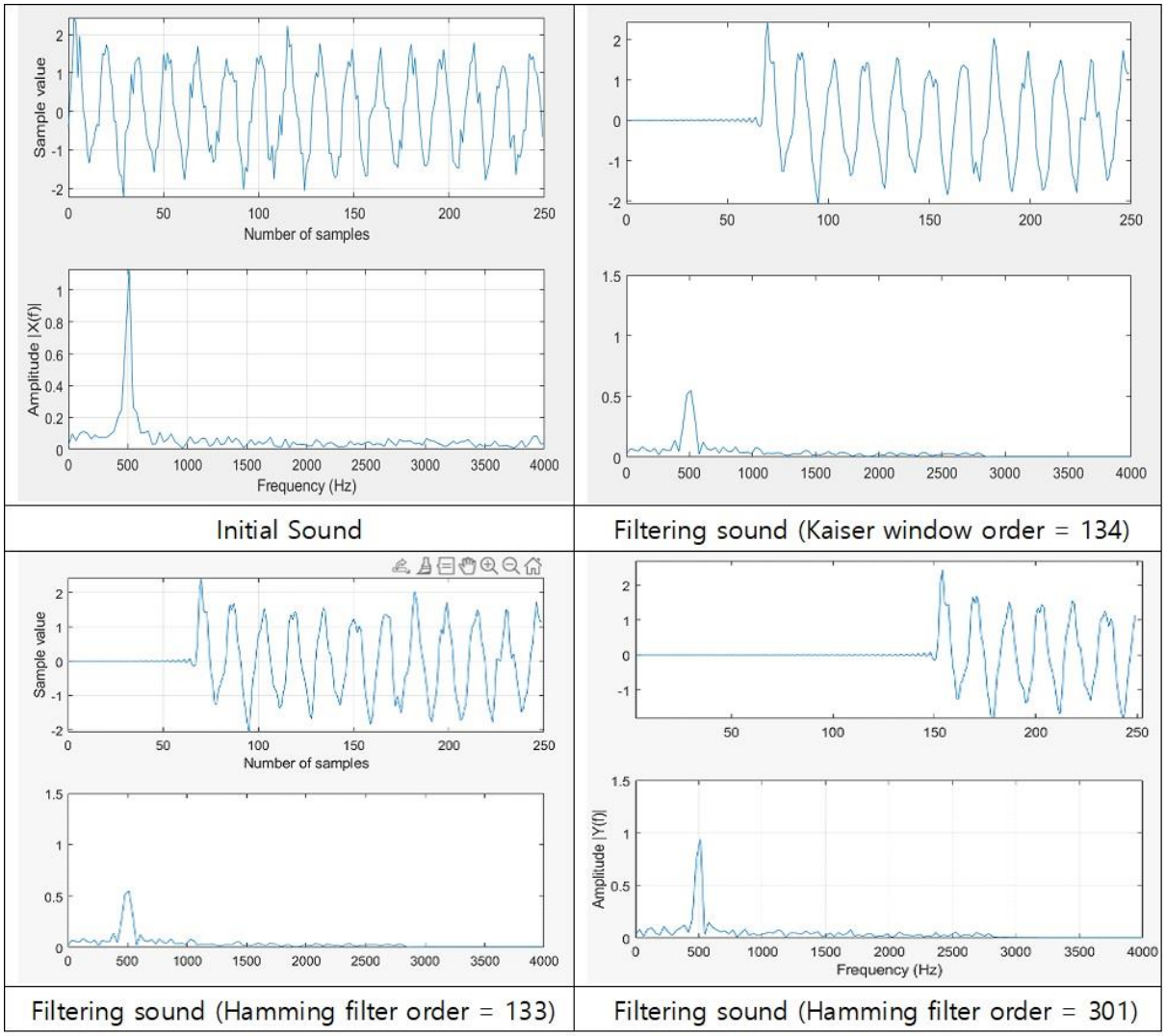
DSP Assignment #16

융합공학부 20153865 김민석

1. Noise Reduction

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Initial Sound</td>
<td>Time domain (original vs filtering sound)</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Digital Frequency Response</td>
<td>Frequency domain (original vs filtering sound)</td>
</tr>
</table>



=> 초기 sound의 frequency 그래프를 보았을 때, low pass filter를 이용하면, 중요한 정보는 살리고, 잡음을 축소시킬 수 있다고 판단하여, low pass filter를 적용하였다.

그리고, cutoff 지점을 찾기 위해, Digital Frequency Response를 확인했다. 그래서 X가 0.17인 지점으로 low pass filter를 만들었다. 적용한 필터는 체비쇼프 윈도우 필터를 적용했으며, 원본 데이터에 비해 더 깨끗한 소리가 들렸다.

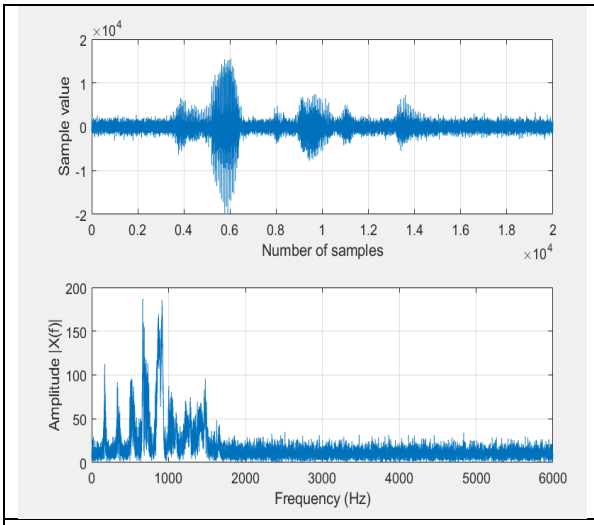
그리고 체비쇼프 윈도우 function을 적용하기 전, hamming window function과 Kaiser window function을 적용한 실험을 했으며, filter의 차수도 늘려보았지만, 체비쇼프만큼 소리가 깔끔하지 않았으며, 그래프 도한 깔끔하지 않았다.

[Code]

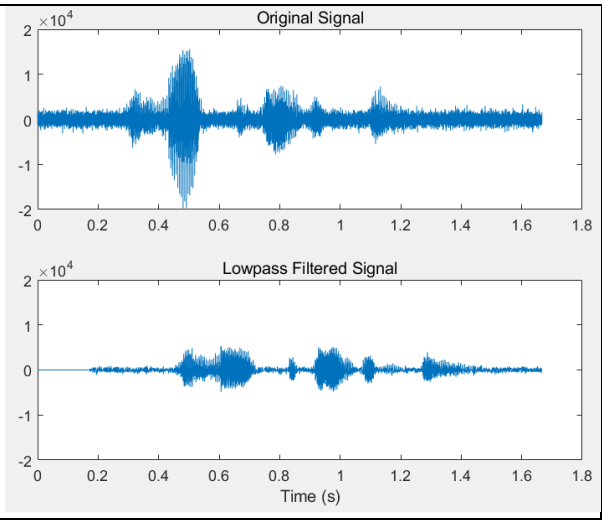
```
>> %7.4.1 Noise Reduction
>> fs = 8000;     %Sampling rate
>> T = 1/fs;      %Sampling period
>> v = sqrt(0.1)*randn(1,250);                    %Generate Gaussian random noise
>> n = 0:1:249;
>> x = sqrt(2)*sin(2*pi*500*n*T)+v;                        %Generate 500Hz + noise

>> freqz(x,1)                                     %Digital frequency response

>> figure(3)                                      %Time domain
>> blo = fir1(133,0.17,chebwin(134,30));
>> outlo = filter(blo,1,x);
>> subplot(2,1,1);
>> t = (0:length(x)-1)/fs;
>> plot(t,x);
>> title('Original Signal');
>> ys = ylim;;
>> subplot(2,1,2)
>> plot(t,outlo);
>> title('Lowpass Filtered Signal');
>> xlabel('Time (s)');
>> ylim(ys);

>> N= length(x);
>> f = [0:N/2]*fs/N;
```

```
>> figure(4)                                                    %frequency domain
>> Axk = 2*abs(fft(x))/N;Axk(1)=Axk(1)/2;
>> subplot(2,1,2); plot(f,Axk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |X(f)|');grid;
>> subplot(2,1,1); plot(f,Axk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |X(f)|');grid;
>> Ayk = 2*abs(fft(outlo))/N;Ayk(1) = Ayk(1)/2;
>> subplot(2,1,2);plot(f,Ayk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |Y(f)|');grid;
```

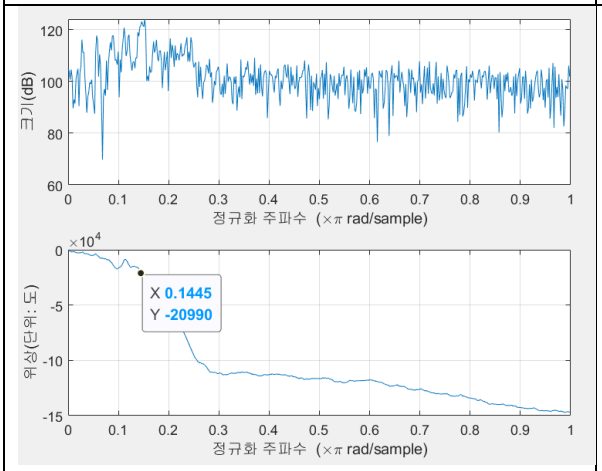
2. Speech Noise Reduction



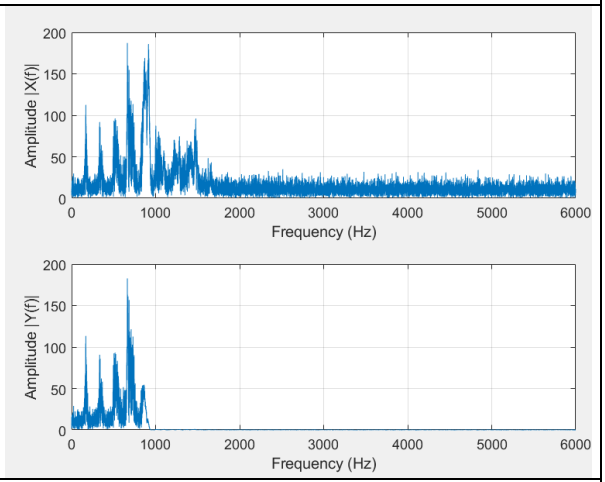
Original sound



Time domain (original vs filtering sound)



Digital Frequency Response



Frequency domain (original vs filtering sound)

=> Sound data를 불러오고나서 spectrum을 plotting 해보니, lowpass filter가 최적의 소리를 만들어 줄 것이라고 판단하였다. 어디서부터 cutoff를 잡으면 좋을지 찾기 위해, digital frequency response ('freqz function' 이용)를 plotting 하여서, x는 0.14에서 끊었다.

그리고, filtering으로는 chebwin window function을 적용했으며, 원본 데이터에 비해 더 깨끗한 소리가 들렸다.

Code

```
>> load('speech_noise.dat');
>> x = speech_noise;
>> fs = 12000;                                          %Sampling rate
>> n = 0:1:19999;
>> subplot(2,1,1);plot(n,x);
>> sound(x)
>> xlabel('Number of samples');ylabel('Sample value');grid;
>> N = length(x)                                        %20000
>> f = [0:N/2]*fs/N;
>> Axk = 2*abs(fft(x))/N;Axk(1)=Axk(1)/2;
>> subplot(2,1,2); plot(f,Axk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |X(f)|');grid;


>> freqz(x,1)


>> figure(3)
>> blo = fir1(133,0.14,chebwin(134,30));
>> outlo = filter(blo,1,x);
>> subplot(2,1,1);
>> t = (0:length(x)-1)/fs;
>> plot(t,x);
>> title('Original Signal');
>> ys = ylim;;
>> subplot(2,1,2)
>> plot(t,outlo);
```

```
>> title('Lowpass Filtered Signal');
>> xlabel('Time (s)');
>> ylim(ys);

>> figure(4)
>> Axk = 2*abs(fft(x))/N;Axk(1)=Axk(1)/2;
>> subplot(2,1,2); plot(f,Axk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |X(f)|');grid;
>> subplot(2,1,1); plot(f,Axk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |X(f)|');grid;
>> Ayk = 2*abs(fft(outlo))/N;Ayk(1) = Ayk(1)/2;
>> subplot(2,1,2);plot(f,Ayk(1:N/2+1));
>> xlabel('Frequency (Hz)'); ylabel('Amplitude |Y(f)|');grid;
```