DSP Simulation Project #4

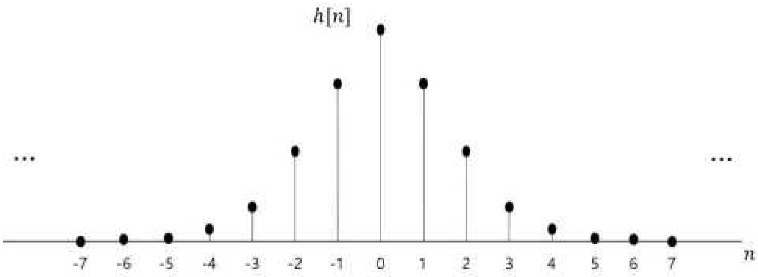
2018142023 조성민

< *Wiener Least Square Filter* >

1. 주어진 음성 신호 $y[n]$('y.txt', Fs = 50,000Hz)을 이용하여 진행하는 실험입니다. $y[n]$은 원본 음성 신호 $x[n]$이 필터 $h[n]$을 통과한 후 노이즈 $u[n]$에 간섭을 받아 열화된 신호입니다. 이는 다음과 같이 표현될 수 있습니다.

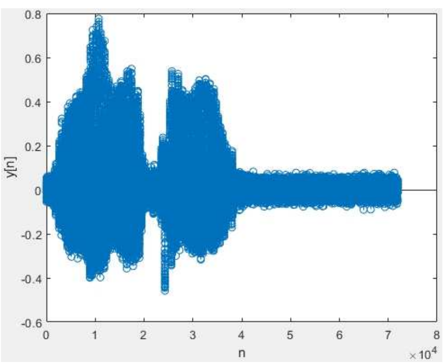
$$y[n] = h[n] * x[n] + u[n]$$

주어진 필터 $h[n]$('h.txt')는 13-point의 Gaussian low pass 필터입니다. 이 필터 $h[n]$의 모든 계수의 합은 1이며, 다음과 그림과 같이 나타낼 수 있습니다.

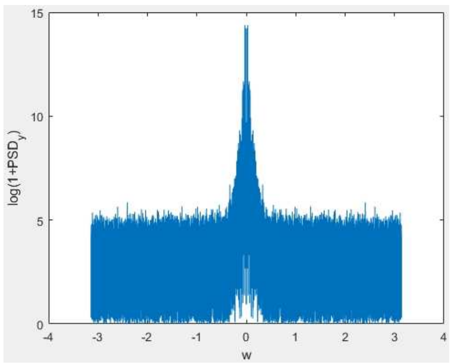


13-point Gaussian low pass filter

① 주어진 입력 신호 $y[n]$과 PSD(Power Spectral Density)의 그래프를 그립니다.

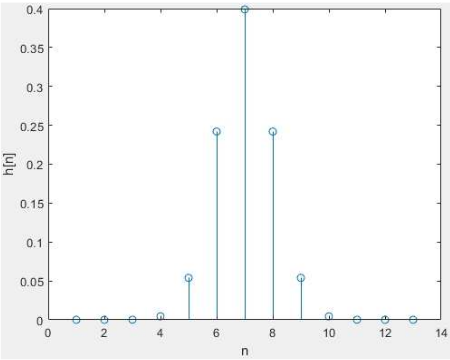


$y[n]$

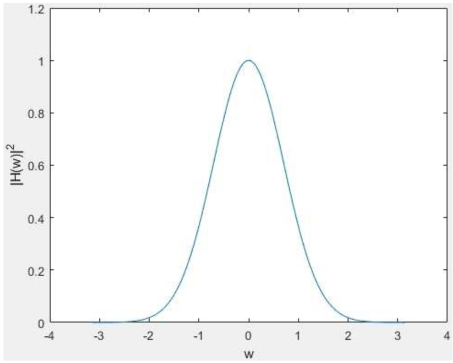


PSD_y

② 필터 $h[n]$과 Frequency Response $H(w)$의 magnitude 제곱 그래프를 그립니다.

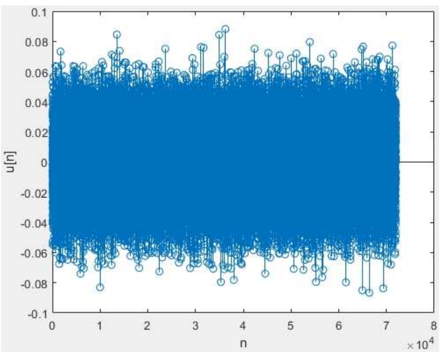


$h[n]$

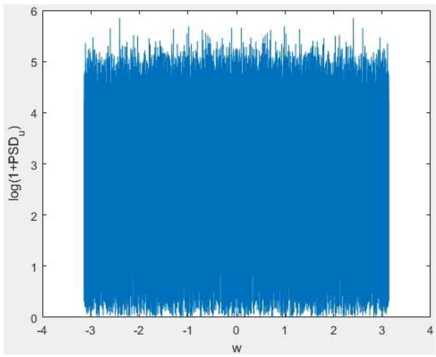


$|H(w)|^2$

③ 노이즈 $u[n]$('noise.txt')는 원본 신호에 대해 15dB SNR을 가지는 Gaussian noise입니다. $u[n]$과 PSD 그래프를 그립니다.



$u[n]$



PSD_u

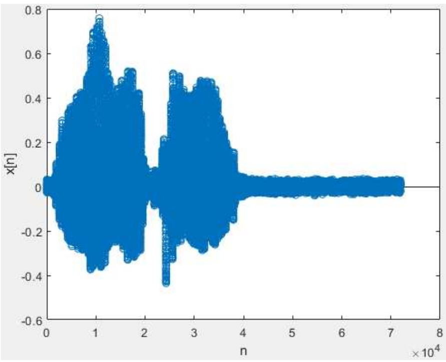
④ Wiener filter를 이용하여 열화된 신호 $y[n]$으로부터 $\hat{x}[n]$을 복원합니다. 복원된 신호 $\hat{x}[n]$과 $\hat{x}[n]$의 PSD를 그리고, 이를 각각 $y[n]$과 $y[n]$의 PSD와 비교 분석하여 토의합니다. Wiener Filter를 구현할 때는 다음을 참고합니다.

$$H_{wiener}(\omega) = \frac{H^*(\omega)}{|H(\omega)|^2 + \frac{S_{uu}(\omega)}{S_{xx}(\omega)}}$$

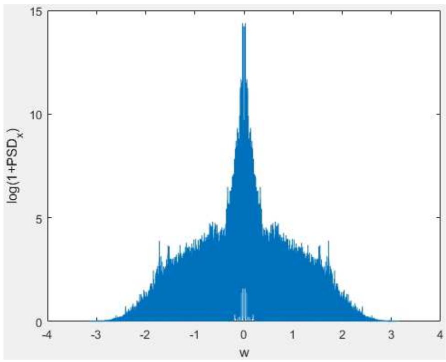
$$\hat{X}(w) = H_{wiener}(w) \cdot Y(w)$$

$$\hat{x}[n] = h_{wiener}[n] * y[n]$$

$x[n]$은 $y[n]$을 통해 복원하고자 하는 **주어지지 않은 원본 신호**이기 때문에 복원과정에 활용될 수 없습니다. 따라서, $S_{xx}(\omega)$ 대신 $S_{yy}(\omega)$를 이용하여 Wiener filter를 설계합니다.



$x[n]$



PSD_x

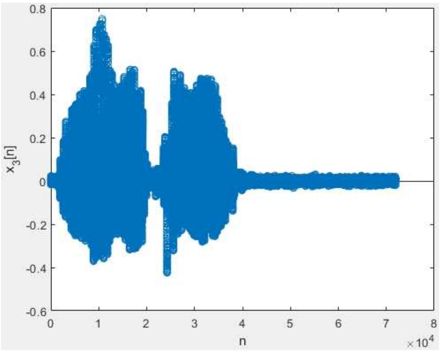
$$-H_{wiener}(w) = \frac{H^*(w)}{|H(w)|^2 + \frac{S_{uu}(w)}{S_{yy}(w)}}$$

-$x[n]$과 $y[n]$을 비교하면, Wiener Filter를 한번 통과한 $x[n]$이 $y[n]$보다 파형이 아주 조금 부드러워진 것을 확인할 수 있다.

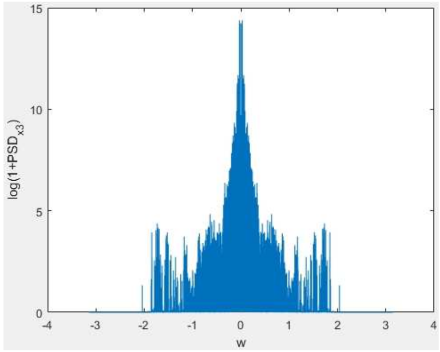
-Wiener Filter를 한번 밖에 적용하지 않았고, $S_{xx}(w)$ 대신 $S_{yy}(w)$가 필터에 사용되었기 때문에 노이즈가 제대로 제거되지 않은 것을 알 수 있다.

-PSD_x와 PSD_y를 비교하면, 고주파 영역에서 신호가 필터링 된 것을 확인할 수 있다.

⑤ Iterative Wiener filter는 복원된 신호 $\hat{x}_k[n]$를 이용하여 더욱 정밀하게 신호 $\hat{x}_{k+1}[n]$를 복원합니다.(k는 iteration 횟수) **예를 들어,** $\hat{x}_1[n]$은 $S_{xx}(\omega)$ 대신 $S_{yy}(\omega)$를 이용하여 복원합니다. 그리고 $\hat{x}_2[n]$은 $S_{xx}(\omega)$대신 $\hat{x}_1[n]$의 power spectrum $S_{\hat{x}_1\hat{x}_1}(\omega)$을 이용하여 더욱 정밀하게 복원합니다. 열화된 신호 $y[n]$으로부터 $\hat{x}_3[n]$을 복원합니다. 복원된 신호 $\hat{x}_3[n]$과 $\hat{x}_3[n]$의 PSD를 그리고, $y[n]$과 $\hat{x}[n]$ 및 각각의 PSD와 비교 분석하여 토의합니다.



$x_3[n]$



PSD_{x_3}

-$x_3[n]$을 $x[n]$, $y[n]$과 비교하면 필터를 더 적용함에 따라 노이즈가 더 많이 제거되어 신호의 파형이 더 부드러워진 것을 확인할 수 있다.

-PSD_{x_3}을 보면 Wiener Filter를 3번 적용하게 되면서 고주파 영역의 신호가 모두 필터링 된 것을 알 수 있다.

⑥ 열화된 신호 $y[n]$, 복원된 신호 $\hat{x}[n]$과 $\hat{x}_3[n]$을 음성으로 들어보고, 비교 분석하여 토의합니다.

-$y[n]$은 음성 “안녕하세요”에서 노이즈가 많이 들리고, $x[n]$은 노이즈가 살짝 줄어든 음성이 들린다.

-$x_3[n]$에서는 노이즈가 거의 다 필터링 되어 음성이 어느 정도 깔끔하게 들린다.

-처음에 주어진 신호가 입력 $x[n]$ 대신 출력 $y[n]$과 노이즈 $u[n]$라고 해도 Wiener Filter를 통해 $x[n]$을 복원시킬 수 있다는 것을 이번 실험을 통해 직접 확인할 수 있었다.

<Appendix>

Q 1-1

```
clc;
clear;

Fs=50000;
y=readmatrix('y.txt');
N=length(y);
Y=fftshift(fft(y));
PSD_y=abs(Y).^2;

n=1:N;
w=2*pi*(-N/2:N/2-1)/N;

figure(1)
stem(n,y);
xlabel('n');
ylabel('y[n]');

figure(2)
plot(w,log(1+PSD_y));
xlabel('w');
ylabel('log(1+PSD_y)');
```

Q 1-2

```
clc;
clear;

y=readmatrix('y.txt');
N_Y=length(y);
h=readmatrix('h.txt');
H=fftshift(fft(h,N_Y));
H_2=abs(H).^2;
w=2*pi*(-N_Y/2:N_Y/2-1)/N_Y;

figure(1)
stem(h);
xlabel('n');
ylabel('h[n]');

figure(2)
plot(w,H_2);
xlabel('w');
ylabel('|H(w)|^2');
```

Q 1-3

```
clc;
clear;

Fs=50000;
u=readmatrix('u.txt');
N=length(u);
U=fftshift(fft(u));
PSD_u=abs(U).^2;

n=1:N;
w=2*pi*(-N/2:N/2-1)/N;

figure(1)
stem(n,u);
xlabel('n');
ylabel('u[n]');

figure(2)
plot(w,log(1+PSD_u));
xlabel('w');
ylabel('log(1+PSD_u)');
```

Q 1-4

```
clc;
clear;

Fs=50000;
y=readmatrix('y.txt');
h=readmatrix('h.txt');
u=readmatrix('u.txt');
N_Y=length(y);
Y=fft(y);
PSD_y=abs(Y).^2;
H=fft(h,N_Y);
H_2=abs(H).^2;
U=fft(u);
PSD_u=abs(U).^2;

H_w=conj(H)./(H_2+PSD_u./PSD_y);
X=Y.*H_w;
PSD_x=abs(X).^2;
x=ifft(X);

PSD_x=circshift(PSD_x,N_Y/2);
n=1:N_Y;
w=2*pi*(-N_Y/2:N_Y/2-1)/N_Y;

figure(1)
stem(n,x);
xlabel('n');
ylabel('x[n]');

figure(2)
plot(w,log(1+PSD_x));
xlabel('w');
ylabel('log(1+PSD_x)');
```

Q 1-5

```
clc;
clear;

Fs=50000;
y=readmatrix('y.txt');
h=readmatrix('h.txt');
u=readmatrix('u.txt');
N_Y=length(y);
Y=fft(y);
PSD_y=abs(Y).^2;
H=fft(h,N_Y);
H_2=abs(H).^2;
U=fft(u);
PSD_u=abs(U).^2;

% x1
H_w1=conj(H)./(H_2+PSD_u./PSD_y);
X1=Y.*H_w1;
PSD_x1=abs(X1).^2;
x1=ifft(X1);

% x2
H_w2=conj(H)./(H_2+PSD_u./PSD_x1);
X2=Y.*H_w2;
PSD_x2=abs(X2).^2;
x2=ifft(X2);

% x3
H_w3=conj(H)./(H_2+PSD_u./PSD_x2);
X3=Y.*H_w3;
PSD_x3=abs(X3).^2;
x3=ifft(X3);

PSD_x3=circshift(PSD_x3,N_Y/2);
n=1:N_Y;
w=2*pi*(-N_Y/2:N_Y/2-1)/N_Y;

figure(1)
stem(n,x3);
xlabel('n');
ylabel('x_3[n]');

figure(2)
plot(w,log(1+PSD_x3));
xlabel('w');
ylabel('log(1+PSD_x_3)');
```

Q 1-6

```
clc;
clear;

Fs=50000;
y=readmatrix('y.txt');
h=readmatrix('h.txt');
u=readmatrix('u.txt');
N_Y=length(y);
Y=fft(y);
PSD_y=abs(Y).^2;
H=fft(h,N_Y);
H_2=abs(H).^2;
U=fft(u);
PSD_u=abs(U).^2;

% x1
H_w1=conj(H)./(H_2+PSD_u./PSD_y);
X1=Y.*H_w1;
PSD_x1=abs(X1).^2;
x1=ifft(X1);
% x2
H_w2=conj(H)./(H_2+PSD_u./PSD_x1);
X2=Y.*H_w2;
PSD_x2=abs(X2).^2;
x2=ifft(X2);
% x3
H_w3=conj(H)./(H_2+PSD_u./PSD_x2);
X3=Y.*H_w3;
PSD_x3=abs(X3).^2;
x3=ifft(X3);

sound(y,Fs);
pause(2);
sound(x1,Fs);
pause(2);
sound(x3,Fs);
```