Delft University of Technology

Statistical Digital Signal Processing and Modeling
EE4C03

Group 10.3 Speech Enhancement

Hang Ji (4759745)

Yuanyuan Zhang (5072972)

November 14, 2019

Abstract

In the field of audio and acoustic signal processing, filters that can remove noise from audio messages are diverse and indispensable. A lot of speech processing depends on the fact that the speech being processed is clean, so it is very important to improve the quality of speech. Obtaining a clean speech estimation can be realized by passing a linear filter. Our objective is to realize speech enhancement by the method of noise reduction. The optimum filter–Wiener filter is helpful to reduce the noise. In this assignment, we firstly review the definition of Wiener filter and design an optimum filter, Wiener filter, to implement the speech enhancement finally.

keywords: Wiener filter, speech enhancement, optimum filter, speech estimation, noise reduction

Introduction

With many applications, including human-to-human communication, human-to-machine, speech enhancement is really important. In most of the uses of signal processing, we cannot observe the desired signals directly because they are always influenced by noises. However, a lot of speech processing applications are based on the fact that speech is noiseless. Consequently, it is necessary to get rid of noise and get clean speech. For the most part, we cannot observe the desired signals directly because they are always influenced by noises. Consequently, it is necessary to get rid of noise and get clean speech. Designing optimum linear filters for estimating one process to another is useful to apply in diverse applications in signal processing, such as speech, radar signals, images. Using Wiener filter[1], the noisy speech signal passes it, and we can obtain a clean speech estimation. In this essay, we focus on the speech signal which is often distorted by various noises, including white noise, babble noise. Our aim is to obtain clean speech signal by designing proper Wiener filter which is also an optimum digital filter. In the following sections, we firstly introduce the modelling and working principle of Wiener filter. And then we present the process of doing the experiment of speech enhancement by the method of designing proper Wiener filter. Finally, we come to a conclusion of the experiment by comparing test results.

FIR Wiener Filter

In the 1940s, Norbert Wiener designed a filter, Wiener filter which can produce the optimum estimate of a signal from a noisy observation. In this essay, we apply FIR Wiener filter to remove the noise. Specifically, the observed speech is denoted by $x(n)$ which is influenced by noise $v(n)$, the desired speech is $d(n)$ which is clean.

$$x(n) = d(n) + v(n) \tag{1}$$

Assuming $d(n)$ and $v(n)$ are wide-sense stationary random processes. From the equation (1), we can find that the desired speech $d(n)$ is affected by noise $v(n)$, so we cannot observe clean speech directly. To achieve the goal of obtaining desired speech $d(n)$, it is necessary to use optimum filter–FIR wiener filter to estimate $d(n)$ and make the mean square error minimum. The illustration of general Wiener filtering problem is shown clearly in figure 1.

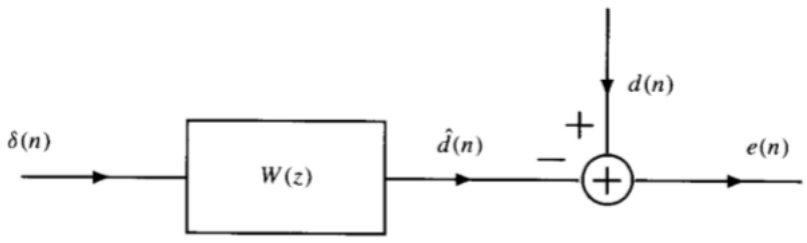

Figure 1: illustration of the general Wiener filtering problem

$\hat{d}(n)$ represents the desired signal $d(n)$ predicted by the FIR wiener filter.The error between $\hat{d}(n)$ and $d(n)$ is denoted by $e(n)$. Furthermore, the minimum mean square error is used to measure the error between the predicted value and the real value. The minimum mean square error is represented by equation (2).

$$\xi = E\left\{|e(n)|^2\right\} \tag{2}$$

where

$$e(n) = d(n) - \hat{d}(n) \tag{3}$$

Assuming that $x(n)$ and $d(n)$ are jointly wide-sense stationary. It is assumed that the autocorrelations of $x(n)$ and $d(n)$ and the cross-correlation between them are known and respectively expressed as $r_d(x)$, $r_x(x)$ and $r_{dx}(x)$. $w(n)$ denotes the unit sample response of the Wiener filter and the filter is assumed as an $a(p-1)$st-order filter. The system function is indicated as equation 4.

$$W(z) = \sum_{n=0}^{p-1} w(n) z^{-n} \tag{4}$$

From figure 1, it is obvious that the desire signal $d(n)$ passes FIR Wiener filter, and then $\hat{d}(n)$ is obtained. The input of the filter is $x(n)$, the output is $\hat{d}(n)$. $\hat{d}(n)$ is the result of convolving $x(n)$ and $w(n)$ as equation 5.

$$\hat{d}(n) = \sum_{l=0}^{p-1} w(l) x(n-l) \tag{5}$$

Substituting equation 3 to equation 2 comes to equation 6.

$$\xi = E\left\{|e(n)|^2\right\} = E\left\{|d(n) - \hat{d}(n)|^2\right\} \tag{6}$$

In order to get a set of filter coefficients to minimize mean sqare error, it is sufficient to make the derivative of $w^*(k)$ with respect to ξ equal to zero for $k = 0, 1, \ldots, p-1$.

$$\frac{\partial \xi}{\partial w^*(k)} = \frac{\partial}{\partial w^*(k)} E\left\{e(n) e^*(n)\right\} = E\left\{e(n) \frac{\partial e^*(n)}{\partial w^*(k)}\right\} = 0 \tag{7}$$

Substituting equation 5 to eqation 3

$$e(n) = d(n) - \sum_{l=0}^{p-1} w(l) x(n-l) \tag{8}$$

it follows that

$$\frac{\partial e^*(n)}{\partial w^*(k)} = -x^*(n-k) \tag{9}$$

Substituting equation 8 and 9 to equation 7

$$E\left\{d(n) x^*(n-k)\right\} - \sum_{l=0}^{p-1} w(l) E\left\{x(n-l) x^*(n-k)\right\} = 0 \tag{10}$$

Because $x(n)$ and $d(n)$ are jointly WSS, we can obtain Wiener-Hopf equations

$$\sum_{l=0}^{p-1} w(l) r_x(k-l) = r_{dx}(k) \quad ; \quad k = 0, 1, \ldots, p-1 \tag{11}$$

$$\mathbf{R}_x \mathbf{w} = \mathbf{r}_{dx} \tag{12}$$

Where R_x is a $p \times p$ Hermitian Teoplitz matrix of autocorrelations, w is the vector of Wiener filter coefficients. Substituting the results of w to equation 6, the minimum mean square error can be computed by equation 13

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w} \tag{13}$$

Assuming the noise $v(n)$ has zero mean, and it is uncorrelated with $d(n)$. Therefore

$$E\left\{d(n) v^*(n-k)\right\} = 0 \tag{14}$$

Thus

$$\begin{aligned} r_{dx}(k) &= E\left\{d(n) x^*(n-k)\right\} \\ &= E\left\{d(n) d^*(n-k)\right\} + E\left\{d(n) v^*(n-k)\right\} \\ &= r_d(k) \end{aligned} \tag{15}$$

With $v(n)$ and $d(n)$ uncorrelated

$$r_x(k) = r_d(k) + r_v(k) \tag{16}$$

Taking the Fourier transform of equation 15 and 16

$$P_{dx}(\omega) = P_d(\omega) \tag{17}$$

$$P_x(\omega) = P_d(\omega) + P_v(\omega) \tag{18}$$

Consequently, obtaining the frequency response of Wiener filter

$$W(\omega) = \frac{P_{dx}(\omega)}{P_x(\omega)} = \frac{P_d(\omega)}{P_d(\omega) + P_v(\omega)} \tag{19}$$

Periodogram

It is useful to estimate the power spectrum density of a wide-sense stationary random process in the field of signal detection and tracking. For example, sonar arrays can be used to listen for acoustic signals to determine the directions or velocities of ships according to estimated power spectrum density theorem. In this essay, we will utilize the method of periodogram which is a kind of methods of estimating the classical or nonparametric spectrum to compare the experimental results of noise deduction experiment. The autocorrelation sequence and the power spectrum of a wide-sense stationary random process are a pair of Fourier transform pair. Therefore, the problem of spectrum estimation can be translated into autocorrelation estimation. The estimation of autocorrelation is denoted by $\hat{r}_x(k)$, and the estimation of the power spectrum known as the periodogram is denoted by $\hat{P}_{per}\left(e^{j\omega}\right)$. As a result, it is feasible to compute the periodogram according to equation 20.

$$x_N(n) \overset{\text{DFT}}{\longrightarrow} X_N(k) \longrightarrow \frac{1}{N}\left|X_N(k)\right|^2 = \hat{P}_{per}\left(e^{j2\pi k/N}\right) \tag{20}$$

The Periodogram

```
function Px = periodogram(x,n1,n2)
%
   x   = x(:);
   if nargin == 1
      n1 = 1;  n2 = length(x);  end;
   Px = abs(fft(x(n1:n2),1024)).^2/(n2-n1+1);
   Px(1)=Px(2);
   end;
```

Figure 2: computing the periodogram of $x(n)$ by MATLAB [1]

Speech Enhancement Experiment

The experiment purpose

Using the Wiener filter to design a basic speech enhancement system which can effectively remove the babble noise and the stationary speech-shaped noise in the noisy speech to obtain the estimation of clean speech in the noisy speech. Finally, comparing the signal noise ratio(SNR) between the estimation of clean speech and the originally noisy speech and showing the similarity between the originally clean speech and the estimation of clean speech.

The experiment process and results

At the first step of our experiment, we generate a sequence of white noise and mix the white noise with the clean speech. The SNR (signal-to-noise ratio) is set as **15** in our experiment.
The second step is to design the wiener filter. In this experiment, we use the wiener filter in the frequency domain to design wiener filter. We apply equation 19 to our implementation of the wiener filter. We estimate the noise from the difference between the desired speech (clean speech) and the observed speech. We calculate

the estimated power spectrum density of the estimated noise and the observed speech and obtain the wiener filter in the frequency domain. Then, we transfer the wiener filter in the time domain. The length of the wiener filter is chosen based on empirical observation. It is set as **100** for our implementation. In fact, the choice of model order is always trade-off. The order of the wiener filter affects the three following issues: the ability of filter to model and remove the distortion and to reduce noise; the computational complexity of filter; numerical stability of the wiener filter solution.
The third step is to filter the observed speech by the obtained wiener filter. The SNR difference[1] is **0.1635**. Figure 3 compares both the noisy speech and the enhanced speech with the desired (clean) speech in the time domain. It can be observed that the noise is effectively reduced. Figure 19 compares both the noisy speech and the enhanced speech with the desired (clean) speech in the frequency domain. Figure 5 compares the noisy speech with the enhanced speech both in time and frequency domain.

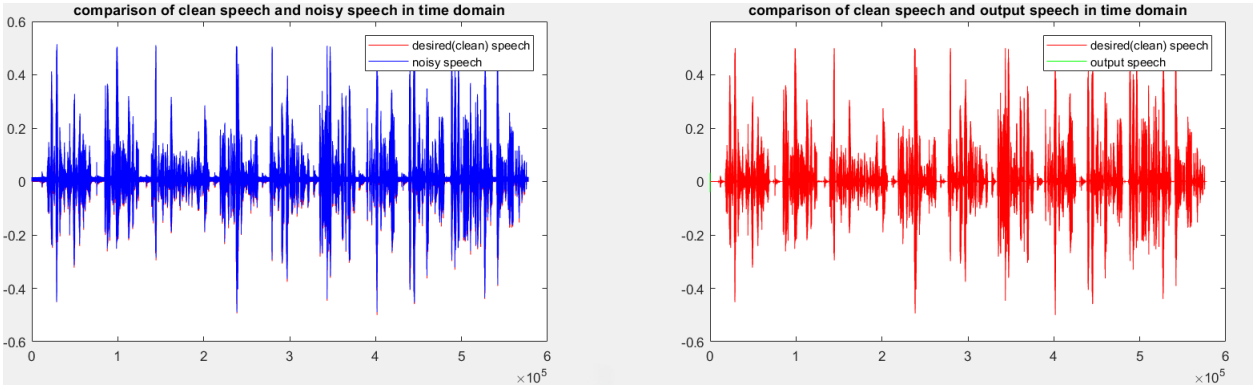

Figure 3: Comparison of noisy speech and enhanced speech in time domain

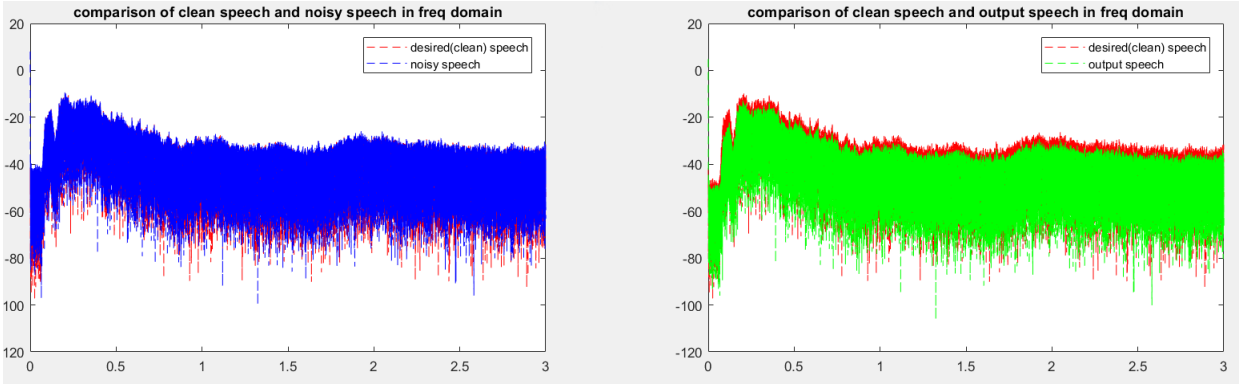

Figure 4: Comparison of noisy speech and enhanced speech in frequency domain

In the fourth step, we apply the wiener filter on other noise scenarios. The first scenario is clean speech mixed with speech shaped noise. Figure 6 shows the noisy speech and the enhanced speech in the time domain. It can be seen that the noise is effectively reduced by the wiener filter. The SNR difference is **0.1512**. The second scenario is clean speech mixed with babble noise. Figure 8 shows the noisy speech and the enhanced speech in the time domain. The noise is also effectively reduced by the wiener filter. The SNR difference is **0.1556**.

Conclusion

In this assignment, we solve the speech enhancement problem by the implementation of wiener filter. To design a wiener filter, we mix the provided clean speech with generated white noise. To test the performance of the wiener filter, we stimulate two scenarios in real life. The first one is clean speech mixed with speech shaped noise, which is likely to happen in some industry scenarios,such as factories. The second one is clean speech mixed with babble noise, which is likely to happen in our daily life, such as a cocktail party. We evaluate the performance of our results by SNR differences, waveform in the time domain, the estimated power spectrum

[1]the SNR difference is defined as snr_diff in the MATLAB code

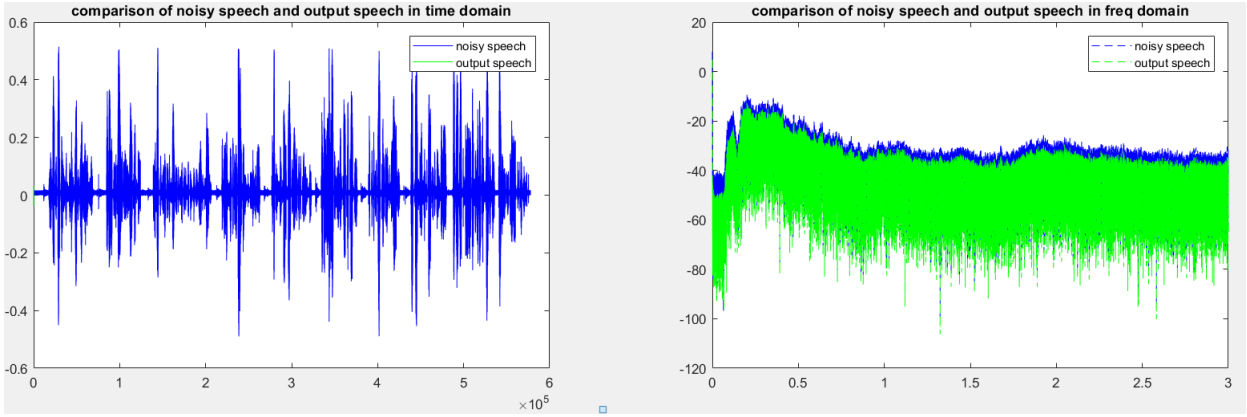

Figure 5: Comparison of noisy speech and enhanced speech in time and frequency domain

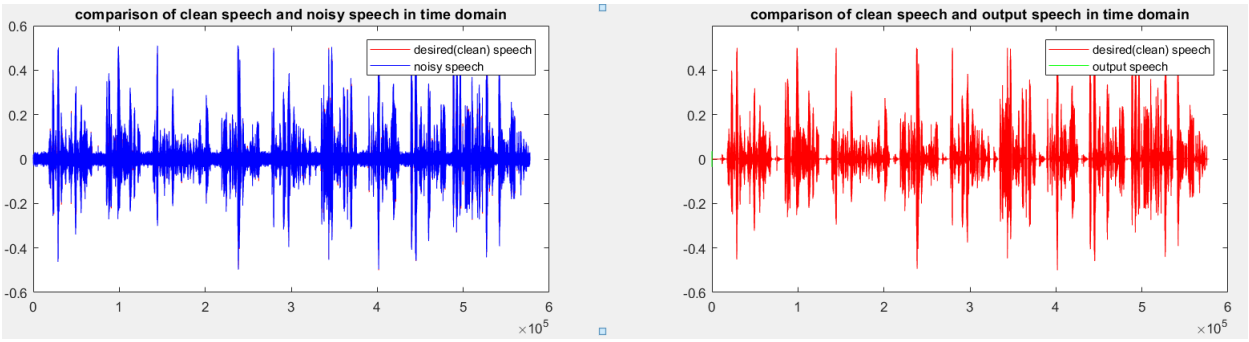

Figure 6: wiener filter applied to clean speech mixed with speech shaped noise in time domain

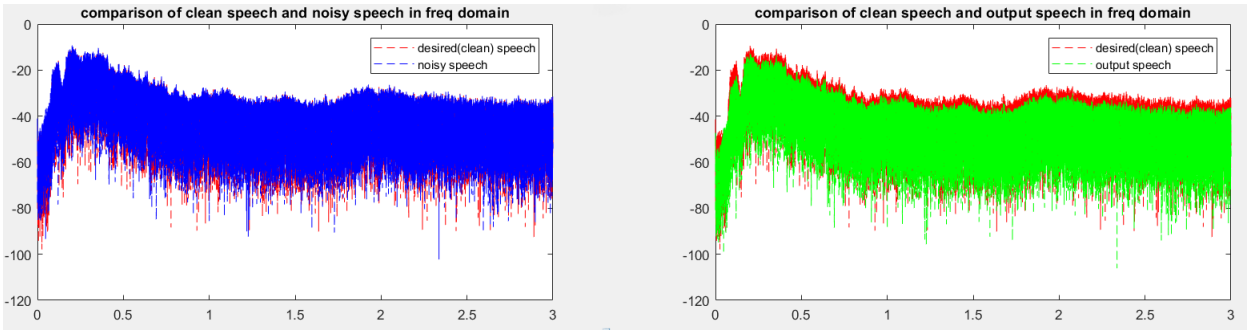

Figure 7: wiener filter applied to clean speech mixed with speech shaped noise in frequency domain

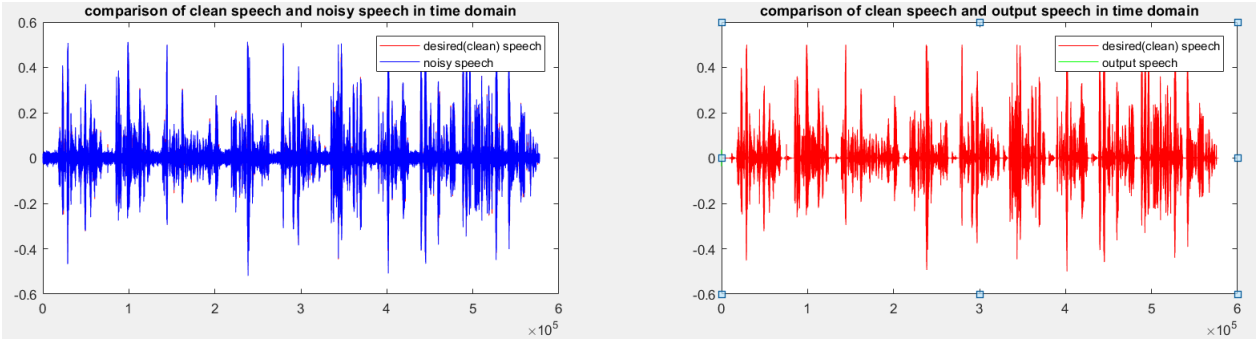

Figure 8: wiener filter applied to clean speech mixed with babble noise in time domain

in the frequency domain. The results show that our wiener filter can reduce the noise in the above mentioned scenarios.

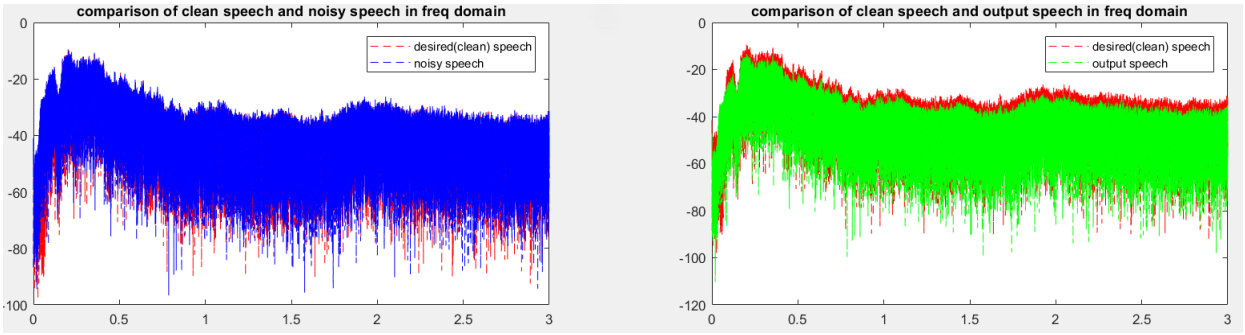

Figure 9: wiener filter applied to clean speech mixed with babble noise in frequency domain

The future work we observed during this assignment includes how to design the length of the wiener reasonably, the relationship between the wiener filter and the SNR, and implementation of adaptive wiener filter.

References

N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series.* John Wiley & Sons, 1940.

Appendix

AddNoise.m

```
function [noisy_signal] = AddNoise(signal, dsnr_db)

% generating zero mean white noise
num = length(signal);
mu = 0;
sigma = 1;
noise = sigma*rand(1, num)+mu;
% compute power of signal
p_signal = sum(abs(signal).*abs(signal))/num;
p_noise = sum(abs(noise).*abs(noise))/num;

% scaling factor
k = (p_signal/p_noise)*10^(-dsnr_db/10);

new_noise = sqrt(k)*noise;
noisy_signal = signal + new_noise;
end
```

TestOtherNoise.m

```
function [noisy_speech] = TestOtherNoise(clean_speech,noise,fs_clean,fs_noise,snr)
% function [noisy_signal] = TestOtherNoise(clean_speech,noise)
% clean_speech
% noise
% noisy_speech mix clean_speech with other noise to test wiener filter
clean_speech = clean_speech(:);
clean_speech = (clean_speech-mean(clean_speech)).';
clean_speech_L = length(clean_speech);

if fs_clean ~= fs_noise
    n = resample(noise,fs_clean,fs_noise);
else
    n = noise;

n = n(:);
n = (n-mean(n)).';
n_L = length(n);

if n_L > clean_speech_L
    n = n(1:clean_speech_L);
else
    % noise length smaller than clean speech length
    % padded with zeros
    n = [n; zeros(clean_speech_L-n_L,1)];
end

L = clean_speech_L;

p_clean_speech =  sum(abs(clean_speech).*abs(clean_speech))/L;

p_noise =  sum(abs(n).*abs(n))/L;

k = (p_clean_speech/p_noise)*10^(-snr/10);

new_noise = sqrt(k)*n;

noisy_speech = clean_speech + new_noise;

end
```

ComputeSNR.m

```
function snr = ComputeSNR(desired_sig,oobs_sig,moobs_sig)
% function [snr] = ComputerSNR(desired_sig,obs_sig,moobs_sig)
% desired_signal clean speech
% oobs_sig noisy speech
% moobs_sig filtered noisy speech
% Speech enhancement 14-11-2019
P_desired_sig = sum(abs(desired_sig).*abs(desired_sig))/length(desired_sig);
P_oobs_sig = sum(abs(oobs_sig).*abs(oobs_sig))/length(oobs_sig);
P_moobs_sig = sum(abs(moobs_sig).*abs(moobs_sig))/length(moobs_sig);

snr_1 = P_desired_sig/P_oobs_sig;
snr_2 = P_desired_sig/P_moobs_sig;

snr = snr_2-snr_1;

end
```

SpeechEhancement.m

```
% speech enhancement 14-11-2019

clear;
clc;
close all;
% load clean speech
clean_speech_path = 'C:\Users\Karen\Desktop\SpeechEnhancement\OriginalData\clean_speech.WAV';
% desired snr ratio
snr_ratio = 15;

% add white noise to the original signal
[y,Fs] = audioread(clean_speech_path);
signal = transpose(y);
noisy_signal = AddNoise(signal,snr_ratio);

% compute wiener filter
NFFT = 100;
wiener_filter = WienerFilter(NFFT,signal,noisy_signal);

% apply wiener filter
output_speech = filter(wiener_filter,1,noisy_signal);

% outcome analysis
% compute difference of snr after filtering
snr_diff = ComputeSNR(signal,noisy_signal,output_speech);

% plot frequency periodogram
[epsd_dsig,wd] = periodogram(signal);
[epsd_nsig,wn] = periodogram(noisy_signal);
[epsd_osig,wo] = periodogram(output_speech);


% plot time domain the clean speech, the noisy speech and the filtered
% speech
figure(1);
subplot(1,2,1);
plot(signal,'r');
hold on;
plot(noisy_signal,'b');
legend('desired(clean) speech', 'noisy speech');
title('comparison of clean speech and noisy speech in time domain');
subplot(1,2,2);
plot(signal,'r');
hold on;
plot(output_speech,'g');
legend('desired(clean) speech','output speech');
title('comparison of clean speech and output speech in time domain');

% plot frequency domain the clean speech, the noisy speech and the filtered
% speech
```

```
figure(2);
subplot(1,2,1);
plot(wd,10*log10(epsd_dsig),'r--');
hold on;
plot(wn,10*log10(epsd_nsig),'b--');
legend('desired(clean) speech','noisy speech');
title('comparison of clean speech and noisy speech in freq domain');

subplot(1,2,2);
plot(wd,10*log10(epsd_dsig),'r--');
hold on;
plot(wo,10*log10(epsd_osig),'g--');
legend('desired(clean) speech','output speech');
title('comparison of clean speech and output speech in freq domain');


% compare between noisy speech and output speech
figure(3)
subplot(1,2,1);
plot(noisy_signal,'b');
hold on;
plot(output_speech,'g');
legend('noisy speech','output speech');
title('comparison of noisy speech and output speech in time domain');
subplot(1,2,2);
plot(wn,10*log10(epsd_nsig),'b--');
hold on;
plot(wo,10*log10(epsd_osig),'g--');
legend('noisy speech','output speech');
title('comparison of noisy speech and output speech in freq domain');

% save wiener filter paramter
save('C:\Users\Karen\Desktop\SpeechEnhancement\SpeechEnhancement\TempData\WienerFilter.mat',
'wiener_filter');
```

SpeechEhancementApply.m

```
% speech enhancement applied in other noisy signal

clc;
clear;
close all;

load('C:\Users\Karen\Desktop\SpeechEnhancement\SpeechEnhancement\TempData
\WienerFilter.mat','wiener_filter');

clean_speech_path = 'C:\Users\Karen\Desktop\SpeechEnhancement\OriginalData\clean_speech.WAV';
noise_path = 'C:\Users\Karen\Desktop\SpeechEnhancement\OriginalData\Speech_shaped_noise.WAV';
% noise_path = 'C:\Users\Karen\Desktop\SpeechEnhancement\OriginalData\babble_noise.WAV';
[clean_speech,Fs_clean] = audioread(clean_speech_path);
[noise, Fs_noise] = audioread(noise_path);

clean_speech = transpose(clean_speech);
noise = transpose(noise);
snr = 15;
noisy_speech = TestOtherNoise(clean_speech,noise,Fs_clean,Fs_noise,snr);

% speech enhance by wiener filter
output_speech = filter(wiener_filter,1,noisy_speech);
% compute diffenrence of snr after filtering
snr_diff = ComputeSNR(clean_speech,noisy_speech,output_speech);

% plot frequency periodogram
[epsd_dsig,wd] = periodogram(clean_speech);
[epsd_nsig,wn] = periodogram(noisy_speech);
[epsd_osig,wo] = periodogram(output_speech);

% plot time domain the clean speech, the noisy speech and the filtered
% speech
figure(1);
```

```
subplot(1,2,1);
plot(clean_speech,'r');
hold on;
plot(noisy_speech,'b');
legend('desired(clean) speech', 'noisy speech');
title('comparison of clean speech and noisy speech in time domain');
subplot(1,2,2);
plot(clean_speech,'r');
hold on;
plot(output_speech,'g');
legend('desired(clean) speech','output speech');
title('comparison of clean speech and output speech in time domain');

% plot frequency domain the clean speech, the noisy speech and the filtered
% speech
figure(2);
subplot(1,2,1);
plot(wd,10*log10(epsd_dsig),'r--');
hold on;
plot(wn,10*log10(epsd_nsig),'b--');
legend('desired(clean) speech','noisy speech');
title('comparison of clean speech and noisy speech in freq domain');

subplot(1,2,2);
plot(wd,10*log10(epsd_dsig),'r--');
hold on;
plot(wo,10*log10(epsd_osig),'g--');
legend('desired(clean) speech','output speech');
title('comparison of clean speech and output speech in freq domain');


% compare between noisy speech and output speech
figure(3)
subplot(1,2,1);
plot(noisy_speech,'b');
hold on;
plot(output_speech,'g');
legend('noisy speech','output speech');
title('comparison of noisy speech and output speech in time domain');
subplot(1,2,2);
plot(wn,10*log10(epsd_nsig),'b--');
hold on;
plot(wo,10*log10(epsd_osig),'g--');
legend('noisy speech','output speech');
title('comparison of noisy speech and output speech in freq domain');
```

WienerFilter.m

```
function [wiener_filter] = WienerFilter(h_length, d_signal, x_signal)
% speech enhancement 14-11-2019
% h_order the order of the wiener filter
% d_signal clean speech
% x_signal clean speech with noise

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% estimated power spectrum density by welch method
Pxx = cpsd(x_signal,x_signal,[],[],1024);
% hamming window, noverlap
% estimated power spectrum density by welch method
Pdd = cpsd(d_signal,d_signal,[],[],1024);
% estimate noise and power spectrum density
v = x_signal-d_signal;
Pvv = cpsd(v,v,[],[],1024);
% estimate cross power spectrum density
Pxd = cpsd(x_signal,d_signal,[],[],1024);

% plot estimiate power spectrum
figure(1);
subplot(2,1,1);
plot(Pxx,'b');
```

```
hold on;
plot(Pxd,'g');
legend('estimated power spectrum density of observed signal','estimated cross power spectrum');
title('estimated power spectrum density');


%  wiener filter


H_fq = Pdd./(Pdd+Pvv); % in frequency domain

H_time = ifft(H_fq);


wiener_filter = (H_time(1:h_length));

end
```