Final Project of Adaptive Signal Processing

Audio Adaptive Noise Cancellation Using LMS Algorithm

Zeyu Liu

Professor Milos

ECE 6880

Spring 2020

The George Washington University

Abstract

This paper describes the basic concept of audio ANC (adaptive noise cancellation), and built an adaptive filter using LMS algorithm. Simulated and analyzed different types of noises mixed signal using by MATLAB. It is observed that LMS algorithm is an effective technique for removal of noise as it has less complexities and can be run fast in test samples. Simultaneously, shows the LMS adaptive filter algorithm has significant noise cancellation effect during a wide frequency range. Through the noise reduction processing of the AR and MA noise models, I get a conclusion that the stronger the correlation between the reference noise and the actual noise, the stronger the independence from the original audio signal, the better the noise cancellation effect. In addition, the project also analyzes the influence of step size μ, filter length M, and signal iterations on the LMS adaptive filter.

Keywords: ANC, LMS, AR model, MA model

Introduction

By studying the course of Adaptive Signal Processing, I suddenly realized that LMS algorithm is useful in many areas like noise cancellation. The theory of noise reduction is very simple. The propagation of sound is achieved by the vibration of the medium. If a wave is opposite to the wave, it will be cancelled under theoretical conditions. Noise canceller will create a completely opposite sound through the processor filter to eliminate the noise. The usual method of estimating a signal corrupted by additive noise is to pass it through a filter that tends to suppress the noise while leaving the signal relatively unchanged i.e. direct filtering[5].

Figure 1. The usual method of estimating noisy signal

However, how to achieve this method? I have learned about the LMS (Least Mean Square) algorithm in the class, it can be used to solve the problem.

Filters used for direct filtering can be either Fixed or Adaptive. The design of fixed filters requires a priori knowledge of both the signal and the noise. However, it is hard to get the priori knowledge of the signal. So, we use adaptive filters, on the other hand, require little or no a priori knowledge of the signal and noise characteristics. Moreover, adaptive filters have the capability of adaptively tracking the signal under non-stationary conditions[5].

Noise Cancellation is a variation of optimal filtering that involves producing an estimate of the noise by filtering the reference input and then subtracting this noise estimate from the primary input containing both signal and noise[5]. This is showing *Figure 2.*

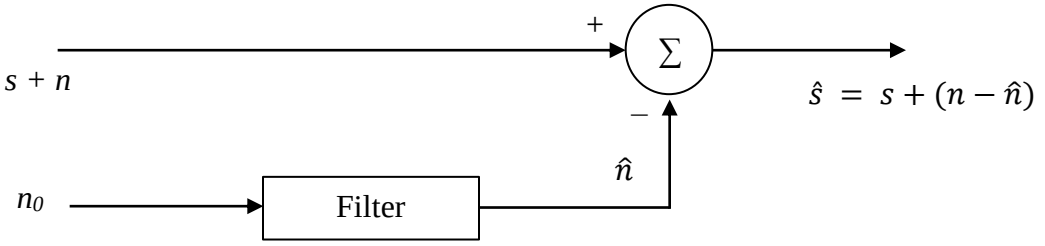

Figure 2. The usual method of estimating noisy signal

I will use the LMS algorithm to design and simulate the noise reduction part. It is a kind of adaptive filter. Showing in *Figure 3.*

The *Figure 3* shows the basic principle of using LMS adaptive noise cancellation technology to solve the problem of signal extraction in the noise background. The main input receives the signal *s* from the signal source but is disturbed by the noise source and receives noise *n*, The reference signal at the reference input n_0 is a noise signal which is not related to the useful signal *s* but related to *n*. Using the correlation between the two input noises and the independence of the sound signal *s* and noise, the reference input n_0 is approximated and subtracted from the noise component *n* in the main input *d* through the adaptive filter, and an error signal *e* is output. It is named as the actual system output $\hat{s}$.

The adaptive filtering algorithm determines the processing of the reference signal n_0 by the filter, so that the output of the filter is as close as possible to the interference component n in the main input. Therefore, in the sense of the best criterion, the filter output $\hat{n}$ approaches n is equivalent to the system output e approximate s.

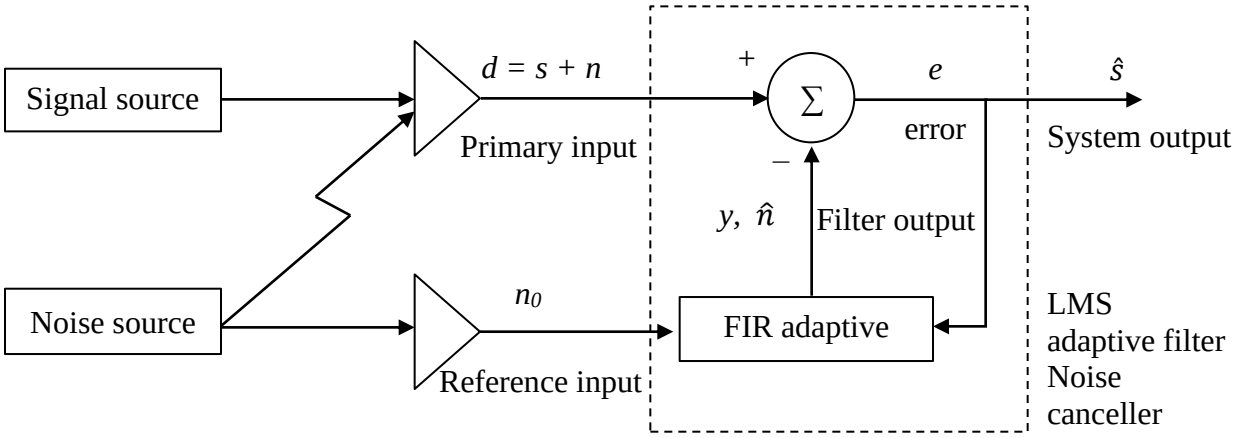

Figure 3. Adaptive Noise Canceller using LMS algorithm

Assume that s, n_0, n and y are statistically stationary and have zero means. The signal s is uncorrelated with n and n_0, and n is correlated with n_0.[4]

$$\hat{s} = s + (n - \hat{n})$$

Squaring, we obtain:

$$\hat{s}^2 = s^2 + (n - \hat{n})^2 + 2s(n - \hat{n})$$

Taking expectation of both sides and realizing that s is uncorrelated with n and $\hat{n}$:

$$E[\hat{s}^2] = E[s^2] + E[(n - \hat{n})^2] + 2E[s(n - \hat{n})]$$
$$E[\hat{s}^2] = E[s^2] + E[(n - \hat{n})^2]$$

The signal power $E[s^2]$ will be unaffected as the filter is adjusted to minimize $E[\hat{s}^2]$. Accordingly, the minimum output power is:

$$min\ E[\hat{s}^2] = E[s^2] + min\ E[(n - \hat{n})^2]$$

When the filter is adjusted so that $E[\hat{s}^2]$ is minimized, $E[(n - \hat{n})^2]$ is also minimized. The filter output $\hat{n}$ is then a best least square estimate of the primary noise no. Moreover, when $E[(n - \hat{n})^2]$ is minimized, $E[(\mathrm{s} - \hat{\mathrm{s}})^2]$ is also minimized[6]. Since:

$$\hat{s} - s = n - \hat{n}$$

Adjusting or adapting the filter to minimize the total output power is thus tantamount to causing the output $\hat{s}$ to be a best least squares estimate of n the signal s for the given structure and adjustability of the adaptive filter and for the given reference input. Since the signal in the output remains constant, minimizing the total output power maximizes the output signal-to-noise ratio[6].

The Adaptive filter using LMS algorithm is simulated in the project using MATLAB R2019b. The file function LMSfilter.m was used for creating the desired LMS filter. The file function finalproject.m was used for creating the main program included showing all the figures and results in the experiments.

I used audio editing tool to get the original sound source, noise source. They are original test audio.wav, white noise.wav, pink noise.wav and car noise.wav files. I will use these sources to achieve the noise cancellation simulation.

There is the important concept I will use in the project:

Least mean square (LMS) algorithm

The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959 is an adaptive algorithm, which uses a gradient-based method of steepest decent. LMS algorithm uses the estimates of the gradient vector from the available data. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to the minimum mean square error. Compared to other algorithms LMS algorithm is relatively simple; it does not require correlation function calculation, nor does it require matrix inversions. [1]

Here is the LMS algorithm equations:

Filter output:

$$y(n) = \hat{\boldsymbol{w}}^H(n)\boldsymbol{u}(n)$$

Estimation error (In this project, error is the desired output):

$$e(n) = d(n) - y(n)$$

weights adaptation:

$$\hat{\boldsymbol{w}}(n+1) = \hat{\boldsymbol{w}}(n) + \mu\boldsymbol{u}(n)e^*(n)$$

Can be explain as the *Figure 4*:

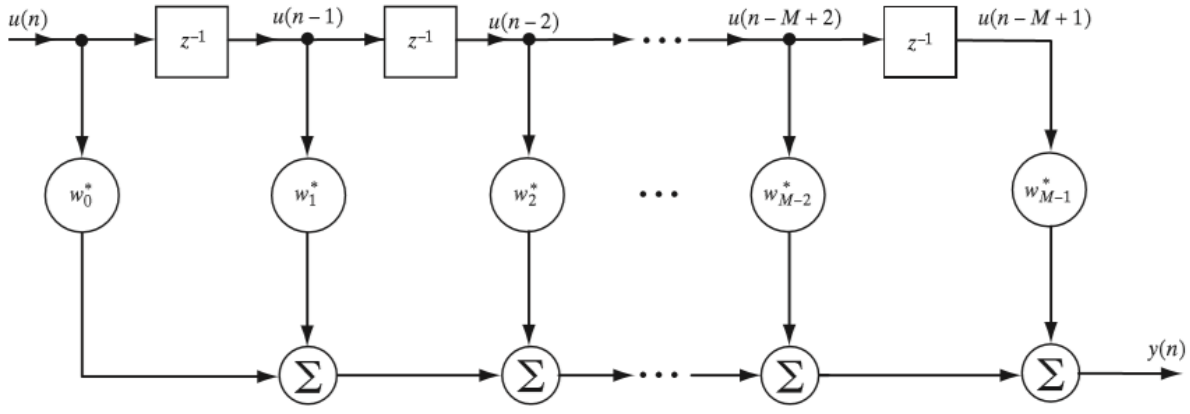

Figure 4. Tap-weights adaptation.[8]

In these equations, the tap inputs $u(n)$, $u(n-1)$, …, $u(n-M+1)$ form the elements of the reference signal $u(n)$, In the project it is same as n_0. where $M-1$ is the number of delay

elements. $d(n)$ denotes the primary input signal as $s + \mathrm{n}$. $e(n)$ denotes the error signal and constitutes the overall system output. $\widehat{\boldsymbol{w}}^H(n)$ denotes the tap weight at the nth iteration[9]. $\widehat{\boldsymbol{w}}^H(n)$ is hermit matrix, $e^*(n)$ is conjugate of $e(n)$.

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

The LMS algorithm is convergent in the mean square if and only if the step-size parameter satisfies. More practical test for stability is[7]:

$$0 < \mu < \frac{2}{\text{Input signal power}}$$

Larger values for step size will increases adaptation rate (faster adaptation) as convergent rate is fast and increases mean-squared error (MSE). This conclusion will show in simulation figures.

Autoregressive model (AR model)

AR model describes the relationship between current and historical values. The notation AR(p) refers to the autoregressive model of order p. The AR(p) model is written[2]:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

Where $\varphi_1, \ldots \varphi_p$ are parameters, c is a constant, and the random variable ε_t is white noise.

moving average model (MA model)

MA model describes the error accumulation of the autoregressive part. The notation MA(q) refers to the moving average model of order q[2]:

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

Where $\theta_1, \ldots \theta_p$ are parameters, μ is the expectation of X_t(often assumed to equal 0), and the random variable ε_t , $\varepsilon_{t-1}, \ldots$ are white noise.

In my project, to prove an assumption after three samples experiments which is the stronger the correlation, the better the noise reduction effect. I use white noise to generate AR model noise as system noise and use white noise to generate MA model noise as reference correlated noise n. So, we can use LMS filter to compare the result with other samples and adjust AR and MA parameters to improve the result. Finally, I will use this model as a new basic sample to change the step-size μ and filter length M and observe the simulation results.

Elaboration and Simulation

The steps of the LMS adaptive filter algorithm implementation is showing in *Figure 5.*

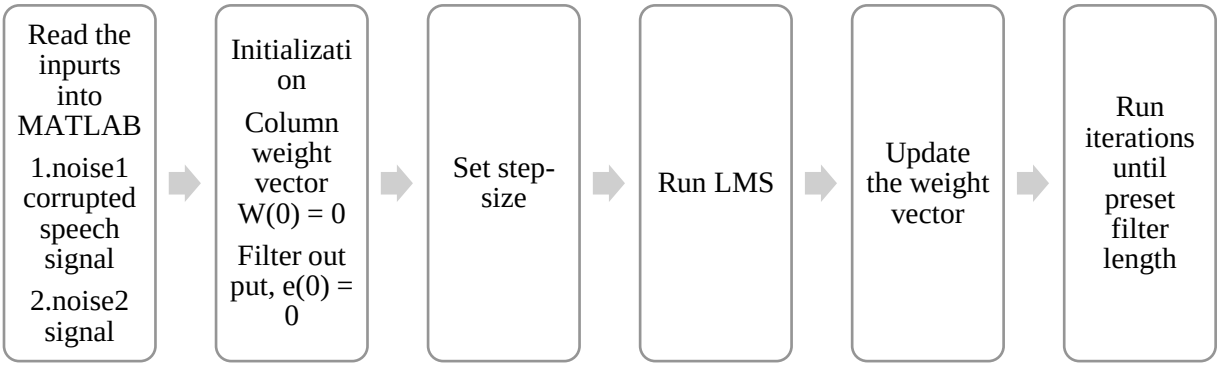

Figure 5. Flow chart of the LMS adaptive filter algorithm implementation

1. Write LMS algorithm as LMSfilter.m in MATLAB.

```
function [yn,e]=LMSfilter(xn,d,M,mu)
N = length(xn);
e = zeros(N,1); % error between prediction and true results
W = zeros(M,N); % M:taps, W:Filter weight matrix, Initial value is 0

for n = M:N     % n-th iteration
    x = xn(n:-1:n-M+1); % M taps inputs
    y = W(:,n-1)'*x; % output of LMS filter
    e(n) = d(n)-y; % n-th error of iteration
    W(:,n)=W(:,n-1)+mu*x*e(n); % update weight
end

% The best output sequence of the optimal filter
yn = ones(size(xn));
for k = M:N
    xb = xn(k:-1:k-M+1);
    yn(k) = W(:,end)'* xb; % Use the best estimate weight get the output
end
```

Figure 6. LMS algorithm code in MATLAB

This algorithm function will be used in many times.

2. Read original test audio and noise.

Read original test audio into a vector s, white noise as n1, pink noise as n2, car noise as n3. Ensure the bit rate of the audio source is consistent. And Extract 1 channel of the 2-channels' original audio to make the simulation clearly. Plot and sound them. Showing in *Figure 7.*

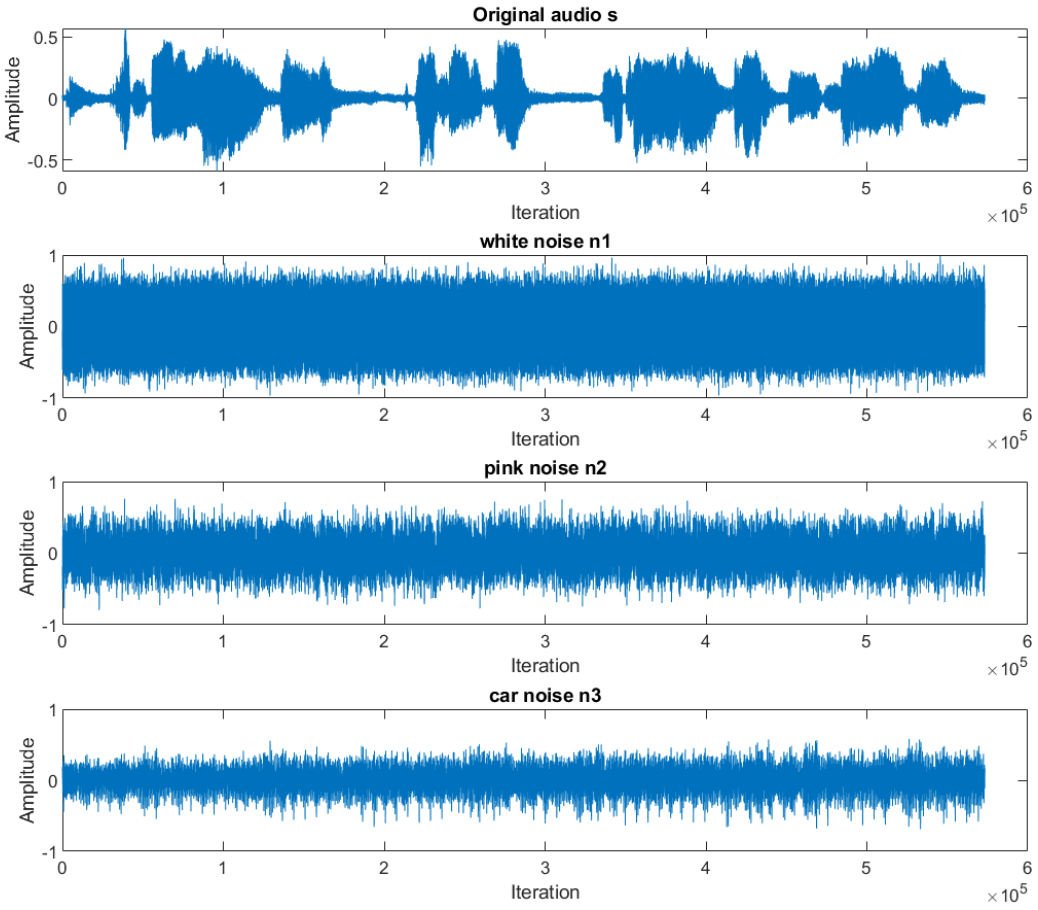

Figure 7. Original audio s and three kind of noise.

3. Plot the power spectral of three noise.

The so-called white noise means that the power of frequency components in a sound is uniform in the entire audible range (0 ~ 20KHZ). White noise is a random noise. The power distribution is uniform.

Pink noise is the most common noise in nature. The power of the frequency component of pink noise is mainly distributed in the middle and low frequency bands. Pink noise is also random noise like white noise. Its power spectral density is inversely proportional to frequency.

The car noise is like pink noise, it mainly composed of lower frequencies signal.

The power spectral of three noise is showing in *Figure 8.*

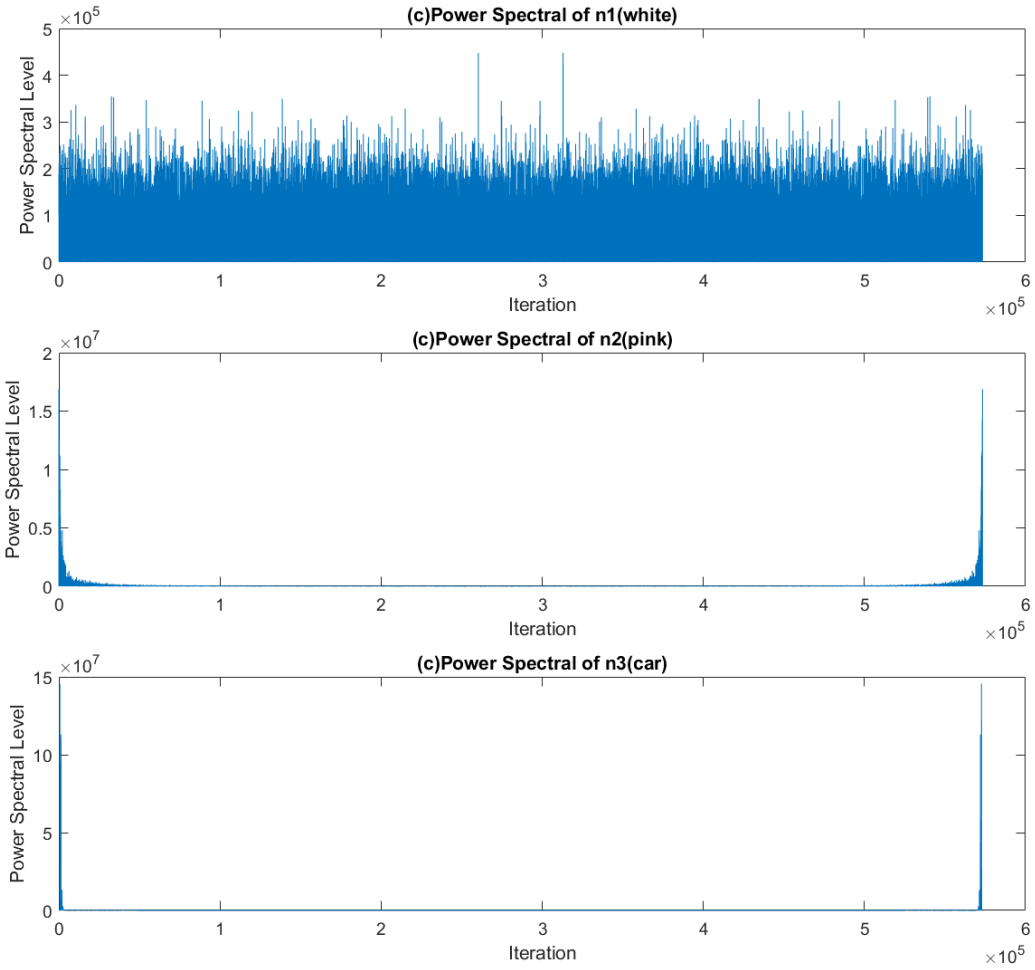

Figure 8. The power spectral of three noise

From *Figure 8* we can see that white noise power and frequency distribution is stable and uniform. Pink noise is mainly low-frequency signal, has a little high-frequency signals. The car noise is mainly low-frequency signal, has less high-frequency signal, even lower than pink noise. So, with this result, if after LMS adaptive filter, we can get noise cancellation well. Then we can say that LMS filter can be work in both high and low frequency signal,

4. Generate the desired signal d.

We know get three desired signals named as:

d1 = s + n1 representative the sound with white noise.

d2 = s + n2 representative the sound with pink noise.

d3 = s + n3 representative the sound with car noise.

Plot and sound them, after that, record them as “sound with white noise.wav”, “sound with pink noise.wav”, “sound with car noise.wav”. Showing in *Figure 9.*

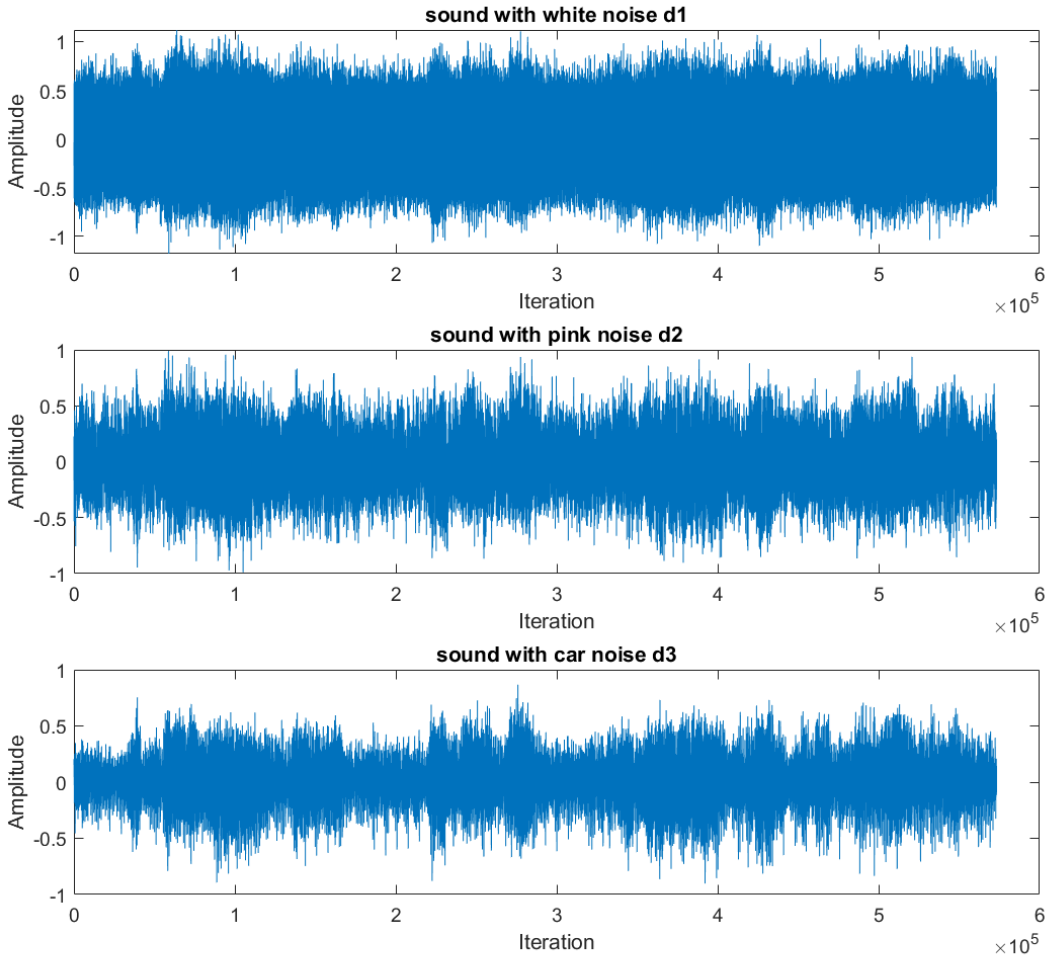

Figure 9. desired signal d1, d2, d3.

The audio signal we found in the picture is mixed with strong noise, and even the noise overwhelms the real signal. In further step, we will put the desired signal pass the LMS adaptive filter to confirm whether it can remove the noise.

5. Generate reference correlated noise n.

I use white gaussian noise as reference correlated noise which variance = 0.1, mean is zero. Because white noise is correlated with three of them and uncorrelated with the sound signal s. Showing in *Figure 10.*

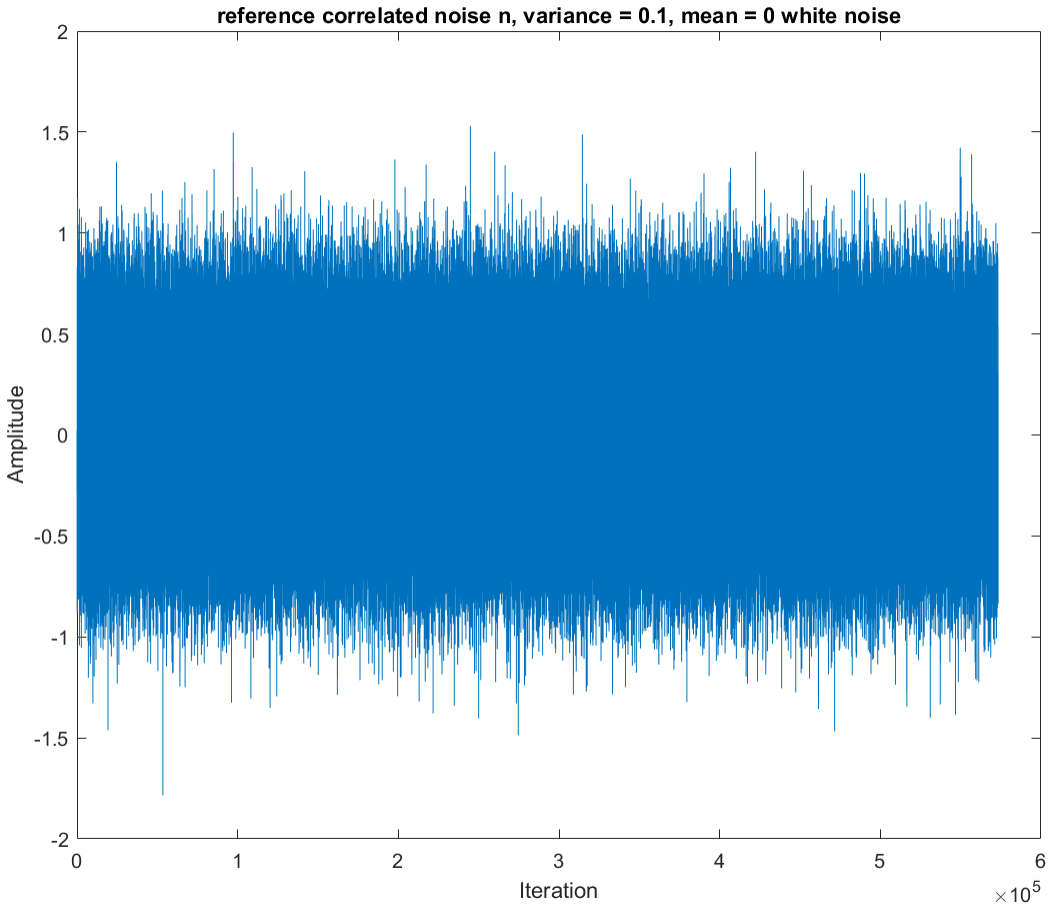

Figure 10. Reference correlated noise n, variance = 0.1, mean = 0, white noise.

6. Using LMS algorithm with n, d, M, μ

Now, we can use the LMS filter to start the noise reduction. Initialize the filter length M = 10, step-size μ = 0.005. Reference input n = n1, n2, n3, respectively. Desired signal d = d1, d2, d3, respectively.

After the training process, we can get yn1, yn2, yn3 which are the best output sequence of LMS filter approach correlated noise n and the system output, which mean sound after noise reduction e1, e2, e3.

e1 sound with white noise after noise reduction

e2 sound with pink noise after noise reduction

e3 sound with car noise after noise reduction

The result showing in *Figure 11.* Plot and sound them, record them as “sound with white noise after noise redection.wav”, “sound with pink noise after noise redection.wav”, “sound with car noise after noise redection.wav”.

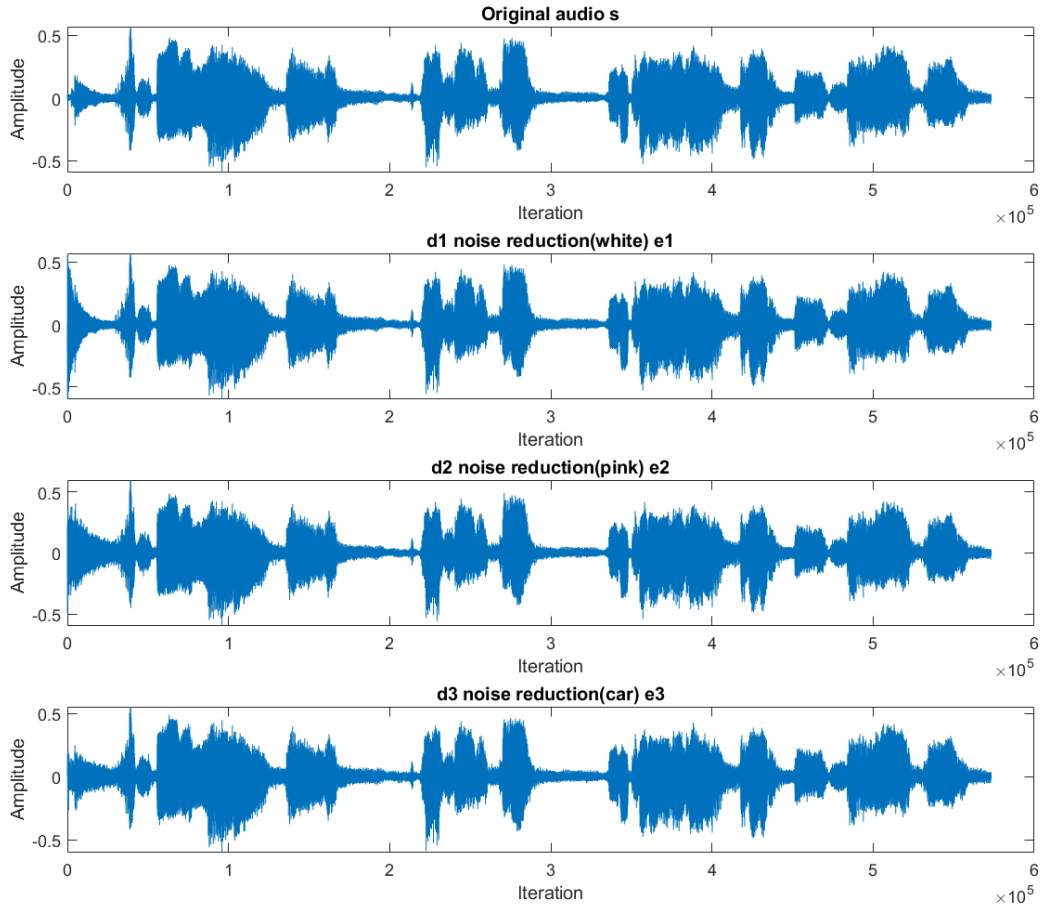

Figure 11. Sound with three noise after noise reduction, s, e1, e2, e3.

By observing and carefully listening to the sound after noise reduction. Although the real audio signal is mixed with strong noise, even the noise drowns the real signal, but after passing our LMS adaptive filter, the real signal can be recovered well. So, the adaptive filter has good denoising performance. So, the adaptive filter has good denoising performance.

We could find this LMS filter has the best effect on eliminating white noise. For pink noise, except for the beginning of the noise, most of them cannot hear the difference from the real sound. For car noise, in addition to the noise generated at the beginning of the convergence, there will be a certain noise floor in the subsequent part. Since the car noise is closer to the real environment, the filter cannot completely eliminate the noise. The main reason maybe that the correlation between the white noise of the reference input and the car noise is not as strong as the pink noise.

To confirm this point. I will use white noise to generate AR model signal and MA model signal in the further simulation.

But overall, the filter did an excited work of eliminating three types of noise. Regardless of high frequency or low frequency, the adaptive filter has a good effect on both of them.

7. Plot the noise reduction process curve, residual noise v.

In order to quantify the quality of the noise reduction effect, here we plots the noise reduction process curve, that is, the residual noise in *Figure 12.*

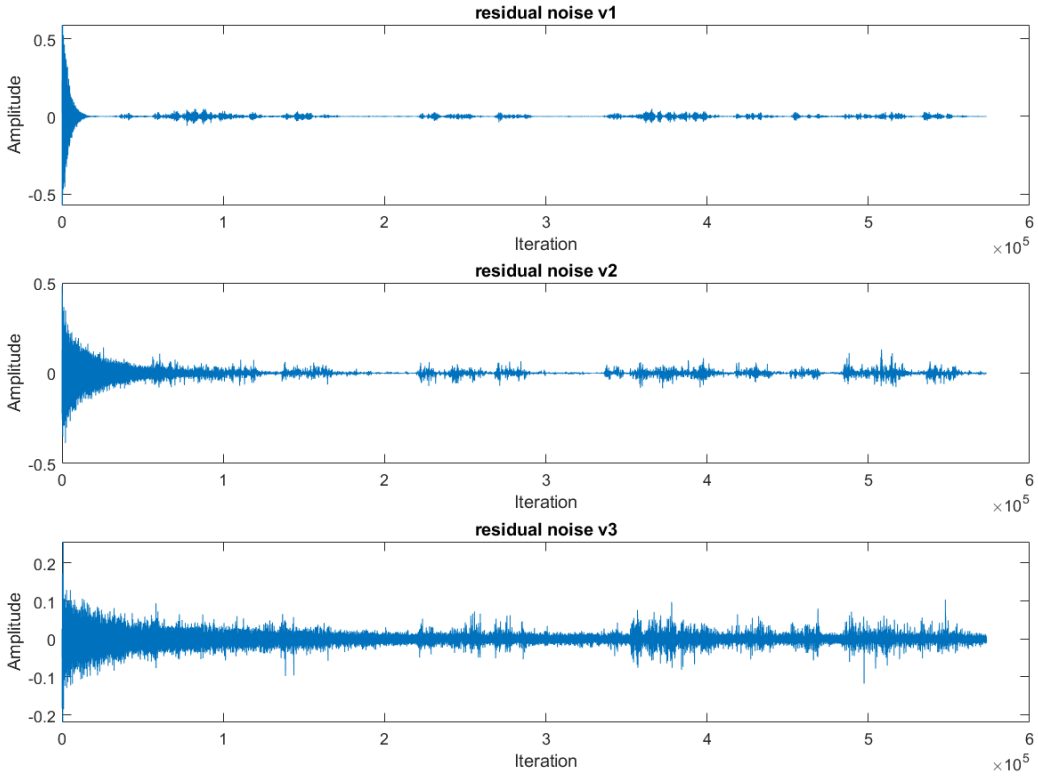

Figure 12. Noise reduction process, residual noise v of three noise

The v1 is the residual noise of sound with white noise, v2 is the residual noise of sound with pink noise, v1 is the residual noise of sound with car noise. Obviously, the filter eliminates white noise best and the car noise worst.

8. Sampling comparison of noise reduction

Now for more clearly. we extract a short signal to compare the three signals: s, e, v in *Figure 13.*

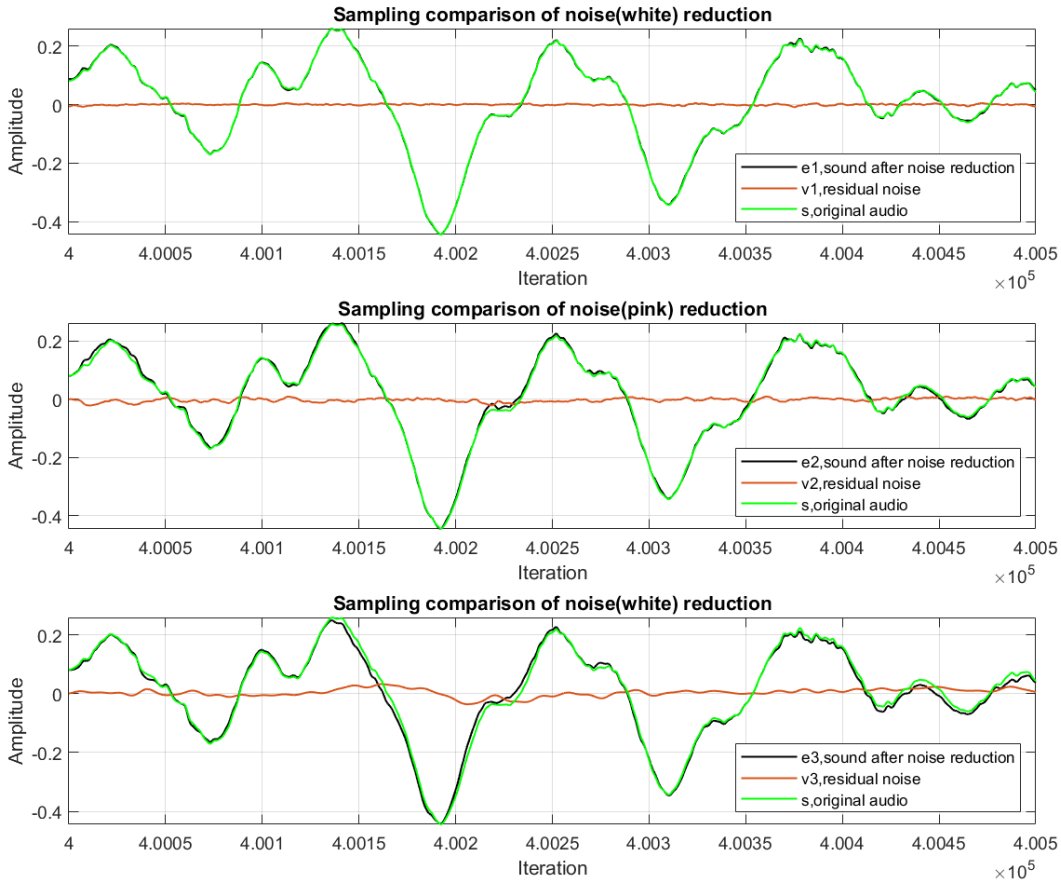

Figure 13. Sampling comparison of noise reduction

Choose 500 iterations from 400000 to 400500. Compared with the original signal, the signal with white noise after noise reduction restored by the adaptive filter is still the best for fitting. The residual noise is also smoother than others.

9. Sampling comparison of noise reduction

To understand the best performance of the adaptive filter. Use the best estimate weight(the last updated weight) to get the best output sequence yn. yn is approximately equal to reference correlated noise n. The results showing in *Figure 14.*

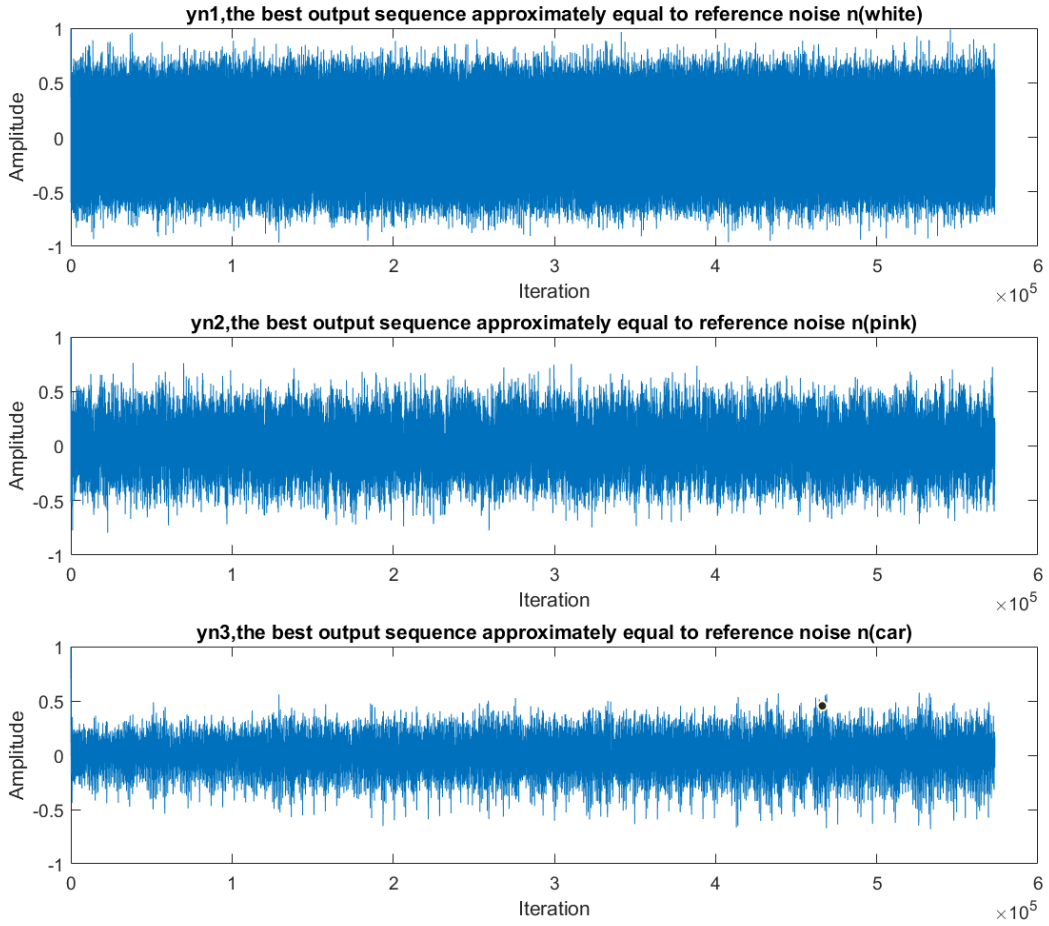

Figure 14. Use the best estimate weight to get the best output sequence yn

It can be seen from the figure that since the reference noise of the filter is white noise, the white noise reduction effect is the best. The pink noise and car noise are poorly restored and cannot be completely eliminated, this caused residual noise in the system.

10. Using white noise generate AR model and MA model

To prove the point that the stronger the noise correlation, the better the noise reduction effect I mentioned before. I use white noise to generate AR model noise as signal noise n-ar and MA model noise as reference correlated noise n-ma. MA model noise is the correlated noise of AR model noise. So, if the noise reduction is acceptable, we can prove that the correlation of the two noise is critical to the filtering result. we can also adjust AR and MA parameters to improve the result. Plot AR model noise as n-ar, MA model noise as n-ma in *Figure 15.* Plot the noise reduction processing in *Figure 16.*

After sound, and look at the figures, the LMS adaptive filter work really well than before. The sound after noise reduction recovered well. As a human I cannot hear the difference from the original audio. Therefore, when the correlation between the reference noise and the signal noise is strong, and the correlation with the original signal is poor, the filtering effect is better.

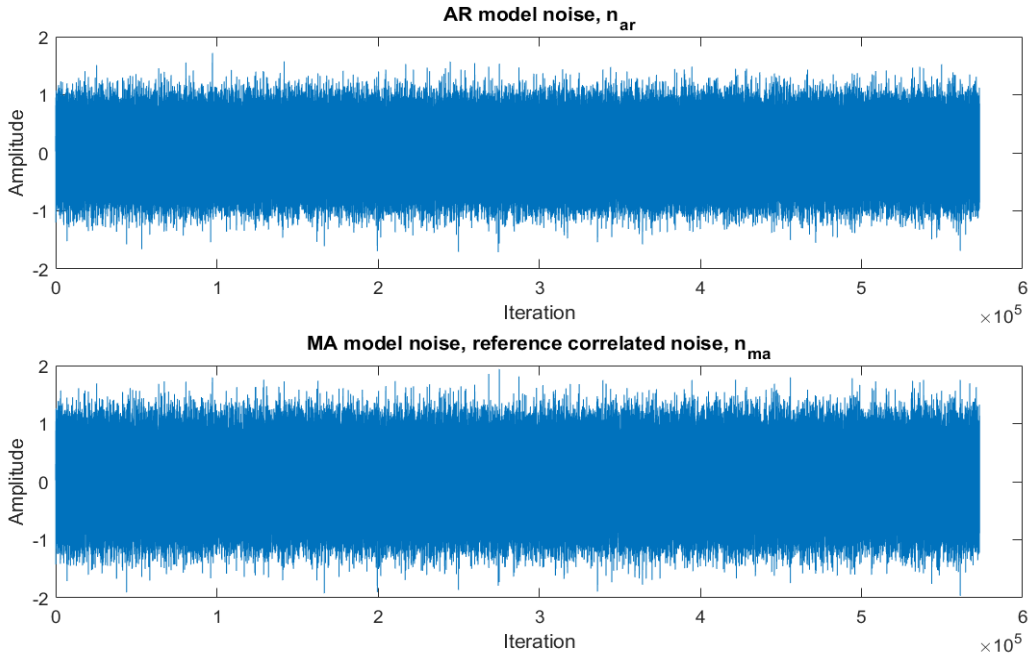

Figure 15. AR model noise and MA model noise.

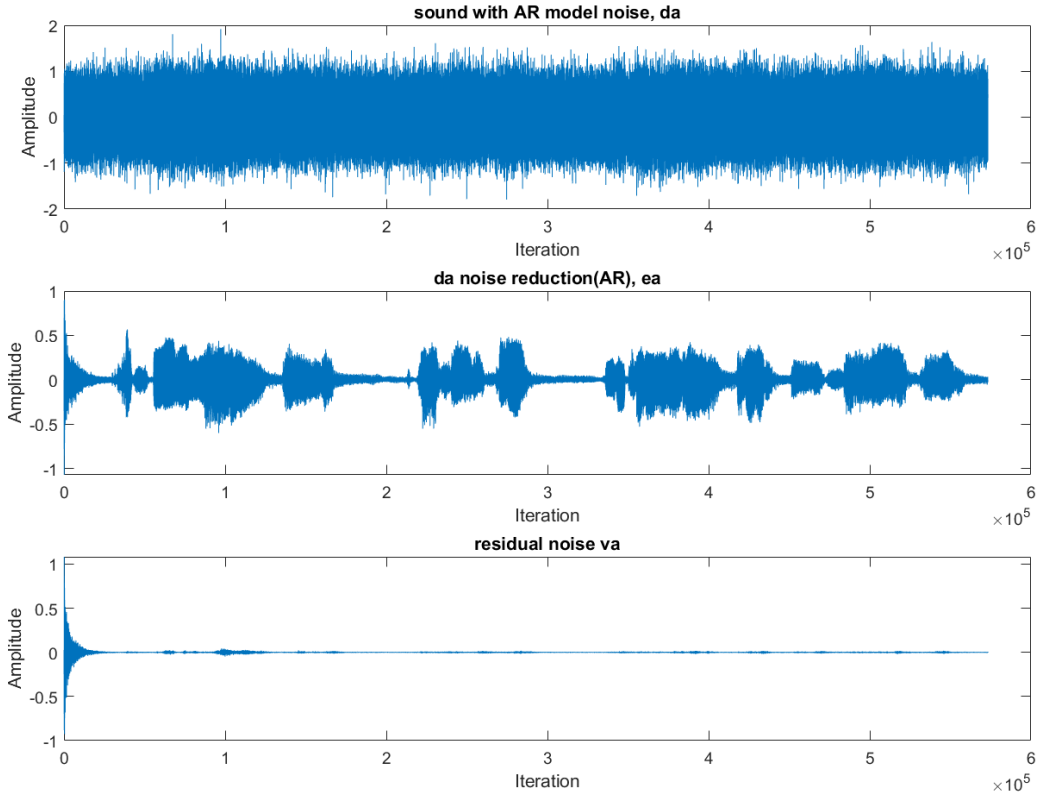

Figure 16. AR noise and MA model noise reduction processing.

We can also use SNR to judge the noise reduction effect. Signal-to-noise ratio (abbreviated SNR or S/N) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to the noise power, often expressed in decibels. A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise[3].

$$\mathrm{SNR} = \frac{\mathrm{P}_{\mathrm{signal}}}{\mathrm{P}_{noise}} = \frac{\mathrm{A}^2_{\mathrm{signal}}}{\mathrm{A}^2_{noise}}$$

$$\mathrm{SNR(dB)} = 10\log_{10}\frac{\mathrm{P}_{\mathrm{signal}}}{\mathrm{P}_{noise}} = 20\log_{10}\frac{\mathrm{A}_{\mathrm{signal}}}{\mathrm{A}_{noise}}$$

From the sound with AR model noise, we can get: SNR1 = 0.3929

From the sound with AR model noise after noise reduction, we can get: SNR2 = 17.9667

In SNR1 the noise is even more than sound! In SNR2 Most of the noise are eliminated. SNR is increased. The noise reduction effect of the LMS filter is acceptable.

11. When change μ or M, the result also changed. Using AR and MA model as example.

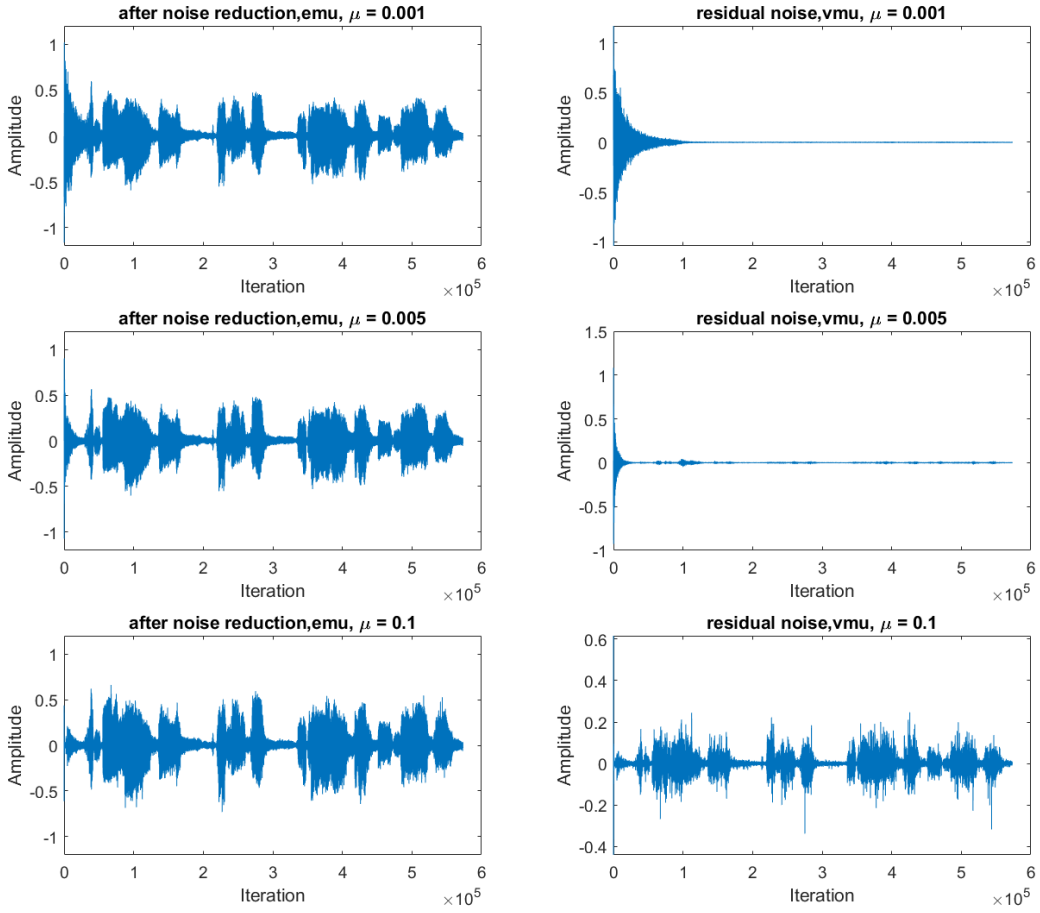

Figure 17. Using same M, different μ in AR and MA noise model.

From *Figure 17.* As the step size μ increases, the convergence time gradually decreases. When μ is small, the convergence is slow and mixed with noise. However, when μ increases, the audio signal becomes clearer. But when the step size exceeds a certain value, there is a lot of noise appear in the audio. Then increase the step size μ again, some audio signal will be filter out.

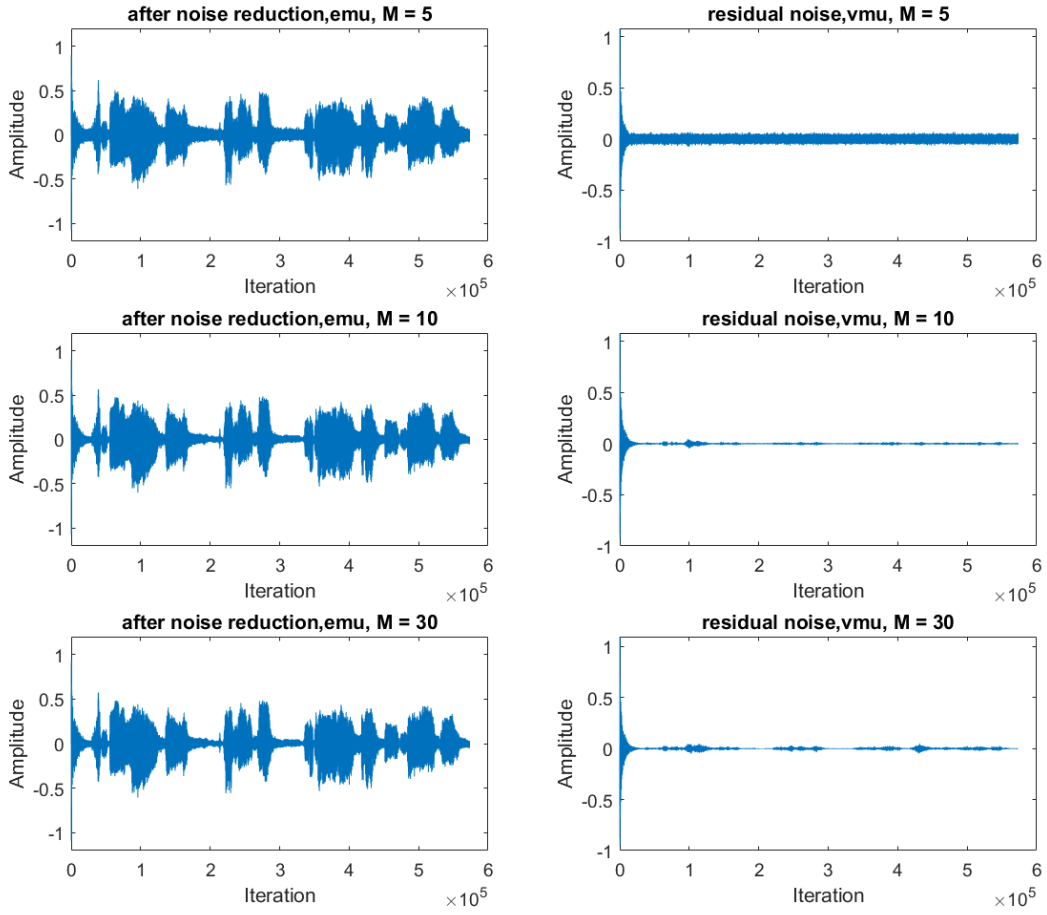

Figure 18. Using same μ, different M in AR and MA noise model.

From *Figure 18.* A small filter length M will cause poor convergence, and too large M will make the filter program run slower, but the convergence speed cannot be significantly improved.

Conclusions

LMS adaptive filter is a very useful tool in audio processing as it has the capability to remove the noise from the signal even high-frequency or low-frequency noise.

I was going to draw a learning curve *J*, and then find the MSE of different μ and M. However, due to too many iterations in the system. it is too slow to calculate all MSE, so I just calculate some key points' MSE showing in *Figure 19.* The first column M = 10 unchanged. The second column μ = 0.005 unchanged. The result is showing *Table 1* and *Table 2.*

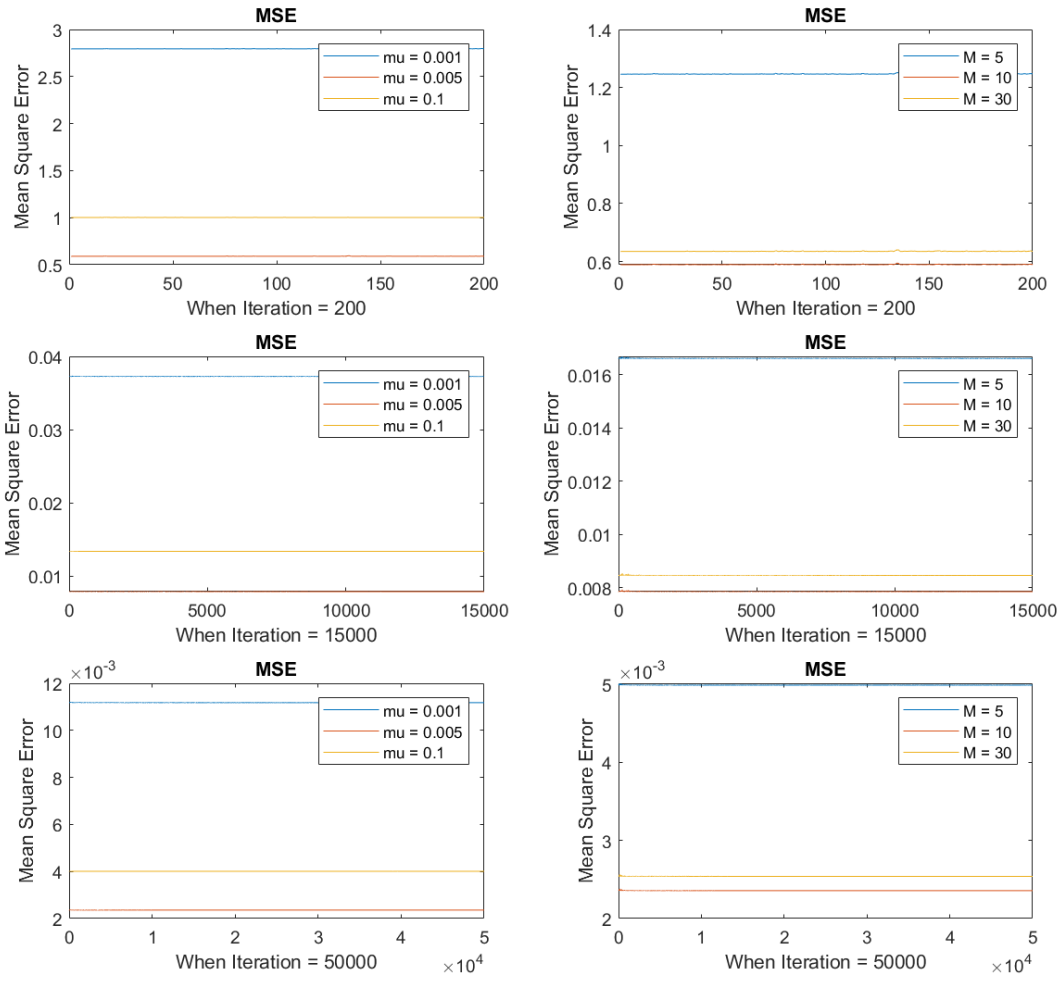

Figure 19. MSE with different iteration and μ or M.

In the simulation it was observed that in the beginning of the signal the filters output was not tracing the desired signal. The residual noise signal was also high. But after some moment, when the filter adapts to the signal and the filter output is almost tracing the desired signal. Hence it reduces the residual noise signal to very low value.

M = 10	Step size μ	iterations	MSE	iterations	MSE	iterations	MSE
	0.001	200	2.7945	15000	0.0373	50000	0.0112
	0.005	200	0.5892	15000	0.0079	50000	0.0024
	0.1	200	1.0023	15000	0.0134	50000	0.0040

Table 1. Convergence of LMS filter for different values of step size μ

μ = 0.005	taps M	iterations	MSE	iterations	MSE	iterations	MSE
	5	200	1.2461	15000	0.0116	50000	0.0050
	10	200	0.5892	15000	0.0079	50000	0.0024
	30	200	0.6349	15000	0.0085	50000	0.0025

Table 2. Convergence of LMS filter for different values of filter length M

In the *Table 1* and *Table 2*, we can get some conclusions:

From the observations of MSE (mean square error) for different values of step size μ and filter length(taps) M, it is concluded that optimized results is achieved for filter length 10 and step size 0.005.

There exists a trade-off between step size and filter length, directly affecting the convergence rate of LMS adaptive filter. When M is unchanged, although a large step-size or learning rate μ can accelerate convergence processing, it will get in a higher MSE. A small μ will get a smaller MSE. However, if the step-size μ is extremely small for the filter, the MSE decreases slowly, the convergence will be slow and will become a little overfitting. However, if μ is too large, it will cause unstable convergence even divergence. At the same time, we also notice that as the iteration increases, the MSE gradually decreases and the filtering effect gradually increases. So, find a suitable μ is important.

When μ is stable if the filter length is not enough to adaptive reduce the noise, that means M is very small, the MSE will be large, the effect of noise reduction will be worse than expect. On the other hand, though a large M can make the noise reduction effect better, it will Increased filter program runtime. Compared with the change of μ, large M leads to a lower probability of overfitting. If we want to apply the LMS adaptive filter more effectively, choosing the appropriate taps is also a key issue.

This work can be extended to other adaptive algorithm applications. Like RLS, NLMS and so on. The experimental and simulated process is similar.

Acknowledgement

ECE6880 is the last course for me in my graduate career. To be honest, the content of the course is really difficult. Fortunately, Professor Milos is very friendly and has excellent teaching ability. I think I still learned a lot, at least when I saw a new filtering algorithm, even if I do not know the principle, I can write it step by step. Although this year everyone is affected by the coronavirus-19, I hope this project can draw a happy ending for my graduate studies, I hope more, and better adaptive algorithms will appear in the future to benefit mankind. Finally, I hope you everyone healthy, wash your hands frequently, and wear masks!

References

[1] Dhobale, S., Boldhan, V., & Burange, R. A. (2013). Implementation of adaptive noise canceller using lms algorithm. In *National Conference on Innovative Paradigms in Engineering & Technology*.

[2] Mills, T. (1990). Time series techniques for economistscambridge university press. *Cambridge et al*, *52*, 68.

[3] Ross, L., & Russ, J. C. (2011). The image processing handbook. *Microscopy and Microanalysis*, *17*(5), 843.

[4] Chhikara, J., & Singh, J. (2012). Noise cancellation using adaptive algorithms. *International Journal of Modern Engineering Research*, *2*(3), 792-795.

[5] Singh, A. (2001). Adaptive noise cancellation. *Dept. of Electronics & Communication, Netaji Subhas Institute of Technology*, *1*.

[6] Singh, G., Savita, K., Yadav, S., & Purwar, V. (2013). Design of adaptive noise canceller using lms algorithm. *International Journal of Advanced Technology & Engineering Research (IJATER)*, *3*(3), 85-89.

[7] Thakkar, V. (2017). Noise Cancellation using Least Mean Square Algorithm. *International Journal of Engineering Research and Application*, *7*(10), 46-57.

[8] Avalos, J. G., Espinobarro, D., Velazquez, J., & Sanchez, J. C. (2009, August). Adaptive Noise Canceller using LMS algorithm with codified error in a DSP. In *2009 52nd IEEE International Midwest Symposium on Circuits and Systems* (pp. 657-662). IEEE.

[9] Dhobale, S., Boldhan, V., & Burange, R. A. (2013). Implementation of adaptive noise canceller using lms algorithm. In *National Conference on Innovative Paradigms in Engineering & Technology*.

Appendix

The full code and audio source are both in the same files.
File **finalproject.m** and **LMSfilter.m** write by MATLAB2019b.

MATLAB code:

LMSfilter.m

```
% ECE6880
% Zeyu Liu
% 5/3/2020
% Final project of adaptive filter processing
% Audio Adaptive Noise Cancellation Using LMS Algorithm
function [yn,e]=LMSfilter(xn,d,M,mu)
N = length(xn);
e = zeros(N,1); % error between prediction and true results
W = zeros(M,N); % M:taps, W:Filter weight matrix, Initial value is 0

for n = M:N     % n-th iteration
    x = xn(n:-1:n-M+1); % M taps inputs
    y = W(:,n-1)'*x; % output of LMS filter
    e(n) = d(n)-y; % n-th error of iteration
    W(:,n)=W(:,n-1)+mu*x*e(n); % update weight
end

% The best output sequence of the optimal filter
yn = ones(size(xn));
for k = M:N
    xb = xn(k:-1:k-M+1);
    yn(k) = W(:,end)'* xb; % Use the best estimate weight get the output
end
```

finalproject.m

```
% ECE6880
% Zeyu Liu
% 5/3/2020
% Final project of adaptive filter processing
% Audio Adaptive Noise Cancellation Using LMS Algorithm
function finalproject
% Read original test audio into a vector x.
[x,Fs] = audioread('original test audio.wav');
N = length(x); % N = 573300, Fs = 44100
s = x(:,1);% Extract 1 channel of the 2-channels' audio as original signal s
subplot(4,1,1);
plot(s);
sound(s,Fs);
title('Original audio s');
xlabel('Iteration');
ylabel('Amplitude');
pause;

% white noise
[n1,Fs1] = audioread('white noise.wav');
n1 = n1(:,1);
subplot(4,1,2);
plot(n1);
sound(n1,Fs1);
title('white noise n1');
xlabel('Iteration');
ylabel('Amplitude');
pause;
% pink noise
[n2,Fs2] = audioread('pink noise.wav');
n2 = n2(:,1);
subplot(4,1,3);
plot(n2);
sound(n2,Fs2);
title('pink noise n2');
xlabel('Iteration');
ylabel('Amplitude');
pause;
% car noise
[n3,Fs3] = audioread('car noise.wav');
n3 = n3(:,1);
subplot(4,1,4);
plot(n3);ylim([-1 1]);
sound(n3,Fs3);
title('car noise n3');
xlabel('Iteration');
ylabel('Amplitude');
pause;

clc;
close all;
```

```
% power spectral, LMSfilter can be used in high and low frequency signal
subplot(3,1,1)
plot((abs((fft(n1)))).^2); % white noise frequency distribution is stable
title('(c)Power Spectral of n1(white)')
xlabel('Iteration')
ylabel('Power Spectral Level')
subplot(3,1,2)
plot((abs((fft(n2)))).^2); % pink noise is mainly low-frequency signals
title('(c)Power Spectral of n2(pink)')
xlabel('Iteration')
ylabel('Power Spectral Level')
subplot(3,1,3)
plot((abs((fft(n3)))).^2); % car noise is mainly low-frequency signals, lower than pink noise.
title('(c)Power Spectral of n3(car)')
xlabel('Iteration')
ylabel('Power Spectral Level')
pause;

clc;
close all;

% Generate the desired signal d
Fs = 44100;
d1 = s + n1; % sound with white noise
subplot(3,1,1);
plot(d1);
sound(d1,Fs);
title('sound with white noise d1');
xlabel('Iteration');
ylabel('Amplitude');
pause;
d2 = s + n2; % sound with pink noise
subplot(3,1,2);
plot(d2);
sound(d2,Fs);
title('sound with pink noise d2');
xlabel('Iteration');
ylabel('Amplitude');
pause;
d3 = s + n3; % sound with car noise
subplot(3,1,3);
plot(d3);
sound(d3,Fs);
title('sound with car noise d3');
xlabel('Iteration');
ylabel('Amplitude');
pause;
audiowrite('sound with white noise.wav',d1,Fs); % record d1.
audiowrite('sound with pink noise.wav',d2,Fs); % record d2.
audiowrite('sound with car noise.wav',d3,Fs); % record d3.

clc;
close all;

% Generate reference correlated noise n. I use white gaussian noise as reference correlated noise
rng(3); % random seed to control the reference noise
n = sqrt(0.1)*randn(N,1); % set variance = 0.1
subplot(1,1,1);
plot(n);
title('reference correlated noise n, variance = 0.1, mean = 0 white noise');
xlabel('Iteration');
ylabel('Amplitude');
pause;

clc;
close all;

% using LMS algorithm with n, d, M, mu, another function named LMSfilter.m
M = 10;% taps, filter order
mu = 0.005;% LMS learning parameter
% y is the best output sequence of LMSfilter approach correlated noise n.
[yn1,e1] = LMSfilter(n1,d1,M,mu);
[yn2,e2] = LMSfilter(n2,d2,M,mu);
[yn3,e3] = LMSfilter(n3,d3,M,mu);

% plot
% original audio s
subplot(4,1,1);
plot(s);

sound(s,Fs);
title('Original audio s');
xlabel('Iteration');
ylabel('Amplitude');
pause;
% sound with white noise after noise reduction
subplot(4,1,2);
plot(e1);
sound(e1,Fs);
title('d1 noise reduction(white) e1');
xlabel('Iteration');
ylabel('Amplitude');
pause;
% sound with pink noise after noise reduction
subplot(4,1,3);
```

```
plot(e2);
sound(e2,Fs);
title('d2 noise reduction(pink) e2');
xlabel('Iteration');
ylabel('Amplitude');
pause;
% sound with car noise after noise reduction
subplot(4,1,4);
plot(e3);
sound(e3,Fs);
title('d3 noise reduction(car) e3');
xlabel('Iteration');
ylabel('Amplitude');
pause;
audiowrite('sound with white noise after noise redection.wav',e1,Fs); % record e1.
audiowrite('sound with pink noise after noise redection.wav',e2,Fs); % record e2.
audiowrite('sound with car noise after noise redection.wav',e3,Fs); % record e3.

clc;
close all;

% noise reduction process curve,residual noise v
v1 = s - e1;
v2 = s - e2;
v3 = s - e3;
subplot(3,1,1);
plot(v1);
title('residual noise v1');
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,2);
plot(v2);
title('residual noise v2');
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,3);
plot(v3);
title('residual noise v3');
xlabel('Iteration');
ylabel('Amplitude');
% sound(v1,Fs); % we don't care about the sound of residual noise.
pause;

clc;
close all;

% Sampling comparison noise reduction
t = (400000:400500);
subplot(3,1,1);
plot(t,e1(400000:400500),'k',t,v1(400000:400500),t,s(400000:400500),'g','linewidth',1);grid;
title('Sampling comparison of noise(white) reduction');
legend('e1,sound after noise reduction','v1,residual noise','s,original audio','Location','southeast')
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,2);
plot(t,e2(400000:400500),'k',t,v2(400000:400500),t,s(400000:400500),'g','linewidth',1);grid;
title('Sampling comparison of noise(pink) reduction');
legend('e2,sound after noise reduction','v2,residual noise','s,original audio','Location','southeast')
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,3);
plot(t,e3(400000:400500),'k',t,v3(400000:400500),t,s(400000:400500),'g','linewidth',1);grid;
title('Sampling comparison of noise(white) reduction');
legend('e3,sound after noise reduction','v3,residual noise','s,original audio','Location','southeast')
xlabel('Iteration');
ylabel('Amplitude');
pause;

clc;
close all;

% Use the best estimate weight to get the best output sequence yn
% yn is approximately equal to reference correlated noise n
subplot(3,1,1);
plot(yn1);
title('yn1,the best output sequence approximately equal to reference noise n(white)');
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,2);
plot(yn2);
title('yn2,the best output sequence approximately equal to reference noise n(pink)');
xlabel('Iteration');
ylabel('Amplitude');
subplot(3,1,3);
plot(yn3);
title('yn3,the best output sequence approximately equal to reference noise n(car)');
xlabel('Iteration');
ylabel('Amplitude');
pause;

clc;
close all;

% To prove that the stronger the correlation, the better the noise reduction effect.
% I generate AR model noise as signal noise and MA model noise as reference correlated noise n,
```

```
% we can adjust AR and MA parameters to improve the result.

% Generating noise of AR model
ar = [1, 1/2];   % AR parameter
n_ar=filter(1,ar,n);
subplot(2,1,1);
plot(n_ar);
title('AR model noise, n_a_r');
xlabel('Iteration');
ylabel('Amplitude');
% Generating noise for MA model, it's the correlated noise of AR model noise
ma = [1,-0.8,0.4,-0.2];   % MA parameter
n_ma = filter(ma,1,n);
subplot(2,1,2);
plot(n_ma);
title('MA model noise, reference correlated noise, n_m_a');
xlabel('Iteration');
ylabel('Amplitude');
pause;

clc;
close all;

Fs = 44100;
da = s + n_ar; % sound with AR model noise
subplot(3,1,1);
plot(da);
sound(da,Fs);
audiowrite('sound with AR model noise.wav',da,Fs); % record da.
title('sound with AR model noise, da');
xlabel('Iteration');
ylabel('Amplitude');
pause;
subplot(3,1,2);
[~,ea] = LMSfilter(n_ma,da,M,mu);
plot(ea);
sound(ea,Fs); % sound after noise reduction
audiowrite('sound with AR model noise after noise reduction.wav',ea,Fs); % record ea.
title('da noise reduction(AR), ea');
xlabel('Iteration');
ylabel('Amplitude');
pause;
va = s - ea; % residual noise va
subplot(3,1,3);
plot(va);
title('residual noise va');
xlabel('Iteration');
ylabel('Amplitude');
% SNR
r1 = snr(da,n_ar);
r2 = snr(ea,va);
disp(r1); % r1 = 0.3929
disp(r2); % r2 = 17.9667
pause;

clc;
close all;

% When change mu or M, the result also changed. Using AR-MA model noise.
% M is the same, change mu
M = 10;
mu = [0.001,0.005,0.1];
for m = 1:3
    subplot(3,2,2*m-1);
    [~,emu] = LMSfilter(n_ma,da,M,mu(m));
    plot(emu);ylim([-1.2 1.2]);
    title(['after noise reduction,emu, \mu = ',num2str(mu(m))]);
    xlabel('Iteration');
    ylabel('Amplitude');
    vmu = s - emu; % residual noise vmu
    subplot(3,2,2*m);
    plot(vmu);
    title(['residual noise,vmu, \mu = ',num2str(mu(m))]);
    xlabel('Iteration');
    ylabel('Amplitude');
    sound(emu,Fs); % sound after noise reduction
    pause;
end

clc;
close all;

% mu is the same, change M
M = [5,10,30];
mu = 0.005;
for m = 1:3
    subplot(3,2,2*m-1);
    [~,emu1] = LMSfilter(n_ma,da,M(m),mu);
    plot(emu1);ylim([-1.2 1.2]);
    title(['after noise reduction,emu, M = ',num2str(M(m))]);
    xlabel('Iteration');
    ylabel('Amplitude');
    vmu1 = s - emu1; % residual noise vmu
    subplot(3,2,2*m);
    plot(vmu1);
```

```
    title(['residual noise,vmu, M = ',num2str(M(m))]);
    xlabel('Iteration');
    ylabel('Amplitude');
    sound(emu1,Fs); % sound after noise reduction
    pause;
end

clc;
close all;
% learning curve J, find the MSE of different mu and M.
% However, It's too slow to calculate all MSE, so just calculate some point MSE.
% M is the same, change mu
M = 10;
mu = [0.001,0.005,0.1];
h = [200,15000,50000]; % change h to calculate the MSE in h iteration.
for mm = 1:3
subplot(3,2,2*mm-1);
    for m = 1:3
        [~,emu] = LMSfilter(n_ma,da,M,mu(m));
        Ja = sum((s - emu).^2);
        J = ((s - emu).^2 + Ja)./h(mm);
        plot(J(1:h(mm)));
        mean(J)
        hold on
    end
title('MSE')
xlabel(['When Iteration = ',num2str(h(mm))]);
ylabel('Mean Square Error');
legend('mu = 0.001','mu = 0.005','mu = 0.1')
hold off
end
% mu is the same, change M
M = [5,10,30];
mu = 0.005;
for mm = 1:3
subplot(3,2,2*mm);
    for m = 1:3
        [~,emu] = LMSfilter(n_ma,da,M(m),mu);
        Ja = sum((s - emu).^2);
        J = ((s - emu).^2 + Ja)./h(mm);
        plot(J(1:h(mm)));
        mean(J)
        hold on
    end
title('MSE')
xlabel(['When Iteration = ',num2str(h(mm))]);
ylabel('Mean Square Error')
legend('M = 5','M = 10','M = 30')
hold off
end
pause;
% figure1: 2.7945, 0.5892, 1.0023
% figure2: 1.2461, 0.5892, 0.6349
% figure3: 0.0373, 0.0079, 0.0134
% figure4: 0.0166, 0.0079, 0.0085
% figure5: 0.0112, 0.0024, 0.0040
% figure6: 0.0050, 0.0024, 0.0025
```