SIMULATION AND PERFORMANCE ANALYSIS OF ACTIVE NOISE CONTROL IN A DUCT

FINAL REPORT

GUOFENG XU
SUPERVISOR: DR. WEN ZHANG
Australian National University
4/11/2016

Contents

Table of Figures

Abstract

Active noise control (ANC) is achieved by generating an ‘anti-noise’ through an electronic system using a specific signal processing algorithm and is widely used for attenuating low-frequency noise. Over the past decade, real-world applications involving ANC technology have been developed. This report illustrates the system performance of two widely used control strategies using filtered-x least mean square algorithm (FxLMS) in MATLAB. Extensive simulations are tested for analyzing system performance by two control strategies for both broadband and narrowband noise in a simulated duct. Studies on main leading factors in the system is conducted. A recommendation on choosing the optimum step size and adaptive filter order is proposed based on the previous studies of ANC and the author’s simulation results.

Keywords — Active noise control, FxLMS, feedforward, feedback, adaptive

Acknowledgement

With deep gratitude and respect, the author wishes to thank Dr. Wen Zhang and her Ph.D. student Mr. Fei Ma for their help and guidance. The author also wishes to thank all the prior research and studies of ANC by other researchers.

1. Introduction

1.1. Overview of ANC

Passive acoustic noise control techniques are valued for their high attenuation over a broad frequency range; however, they are ineffective and costly at low frequencies [1]. Especially, large ventilation systems are very noisy but widely used in human community. Accordingly, there is a need to decrease the noise level by other methods. The idea of active noise control (ANC) was firstly introduced by Paul Lueg in 1936 [2]. In his proposed system as shown in Figure 1, a reference microphone detects the primary (unwanted) noise and gives input signal to the electronic system to drive the loudspeaker, generating a canceling signal to attenuate the primary noise. Specifically, the canceling ‘anti-noise’ has the same amplitude but opposite phase with the primary noise, thus resulting in the cancellation of both noises based on the principal of superposition. Since the characteristics of duct environment and noise source propagation are time-varying, the amplitude and phase of the canceling noise generated from the electronic system have to be adjusted to achieve a satisfying performance [3]. Therefore, an ANC system must be adaptive to such changes. The beginning of adaptive algorithms can be traced back to 1970 by B. Widrow [4, 5] and the advancement in hardware digital signal processing technologies has encouraged extensive research and studies on ANC. Adaptive filter based on least mean square (LMS) algorithm was firstly introduced in 1975 [6] and then has been developed into a vast number of ANC systems and algorithms [7]. Particularly, a filtered-x least mean square (FxLMS) algorithm is the most popular and widely used algorithm to update the controller as it has the advantages of simple computation and robust performance. Thus, feedforward and feedback control strategies based on the FxLMS algorithm and their performance analysis will be mainly focused and discussed in this report.

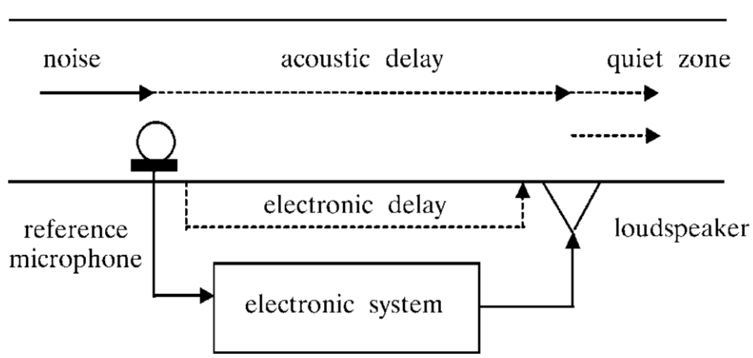

Figure 1. Lueg’s system

1.2. Current applications

ANC is an effective means to achieve large amounts of noise cancellation in a small package, especially at low frequencies. Over the past decade, active noise control has developed into a practical technology with numerous real-world applications. Current applications for ANC include attenuation of unavoidable noise in the following equipment [1]. More details of ANC applications are introduced in [8].

1) Automotive: Noise attenuation inside vehicle passenger compartments. For example, Ford released its new Ford Mondeo, which is provided with ANC technology. Ford's ANC emits sound waves that cancel out unwanted noises in the cabin such as engine, transmission and wind noise [9].
2) Appliances: Including air-conditioning ducts, refrigerators, vacuum cleaners, room isolation, kitchen exhaust fans, and so on.
3) Industrial: Including fans, air ducts, transformers, blowers, power generators, and so on.
4) Transportation: Including airplanes, ships, helicopters, and so on.
5) Electronic equipment: Noise cancellation headphone. Headphones like Bose A20 has shown its incomparable advantage over other non-ANC headphones in terms of auditory experience [10].

1.3. Noise source

Real-time environment involves a variety of acoustic noises such as narrowband noise, broadband noise or a combination of the two. Narrowband and broadband noises have their unique acoustic characteristics. Therefore, they require different control strategies.

For narrowband noise, its energy is concentrated at specific (discrete) frequencies. In time domain, it can be regarded as periodic or near periodic signals, such as sinusoidal noise signal. Thus, noise generated by rotating machines such as fans, diesel engines, large-scale cutting machines usually contains several or more sinusoidal noise signals [7].

For broadband noise, it is caused by turbulence and totally random. Therefore, its energy is distributed over a wide range of frequency bands. Opposite to the narrowband noise, broadband noise is non-periodic in time domain so that it is harder to be removed using adaptive filter.

1.4. Basic Principals – feedback and feedforward control

Active noise control is based on feedforward or feedback control strategies. In the feedforward control case, a reference noise is assumed to be available for the adaptive filter. Depending on the type of primary noise, feedforward control can be categorized as two cases: broadband noise and narrowband noise. In the broadband feedforward control case, a reference microphone is used for detecting a reference noise so that the primary noise correlating with the reference noise can be canceled out. While for the narrowband feedforward case, the reference noise is generated internally using information of mechanical motion of the noise sources detected by appropriate sensors [5]. The feedforward ANC system utilizes a controller (adaptive filter) which drives a loud speaker to generate an anti-noise. An error microphone is used at the end of the duct to pick up residual noise, where functioned as updating the adaptive filter and monitoring the entire ANC system. The single-channel feedforward ANC system which will be mainly discussed in the following sections is widely used for industrial applications such as attenuating duct noise [11].

The feedback ANC system contains an error microphone and a loudspeaker, but without using an 'upstream' reference sensor. Analog feedback control based on a simple negative feedback is applied and widely used in headphones. However, it is quite difficult to reduce broadband noise since the controllable bandwidth is limited by the throughput of the overall control system [11]. Therefore, adaptive feedback ANC system is efficient and effective in attenuating narrowband noise [7].

1.5. Research statement and paper outline

In this report, two control strategies using FxLMS algorithm are implemented in MATLAB to analyze the performance of ANC system for both narrowband and broadband noise. The algorithm introduction and discussion will be shown in Section 2. Detailed performance analysis on different system parameters will be shown in Section 3. Finally, a summary of the report will be concluded in Section 4.

2. Methodology

2.1. Feedforward FxLMS

The most popular and widely used adaptive algorithm for ANC applications is called FxLMS algorithm [12]. The block diagram for a single channel feedforward ANC system using FxLMS algorithm is shown in Figure 2.

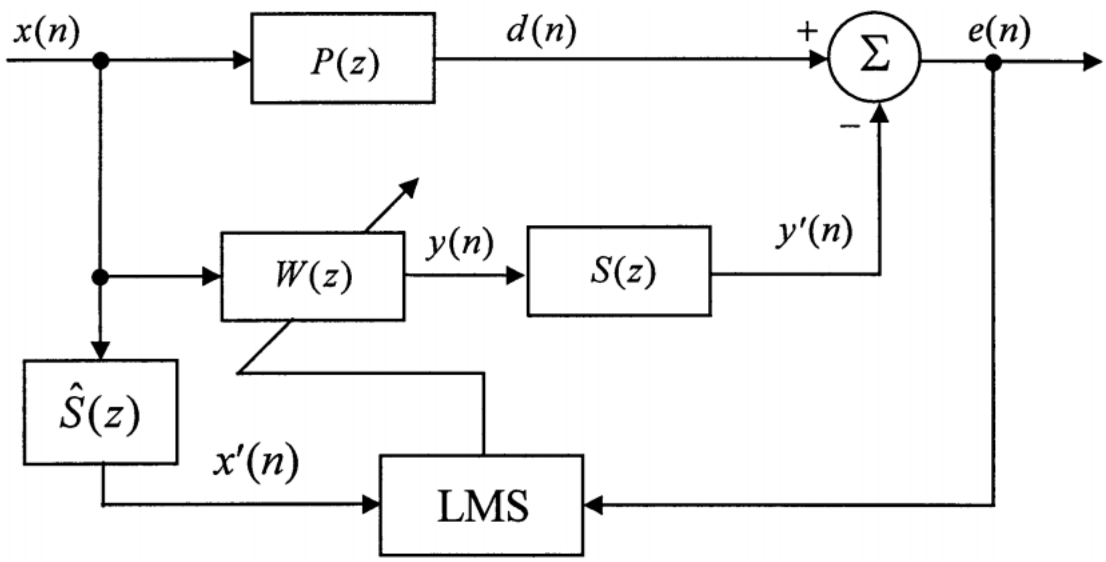

Figure 2. Block diagram of feedforward ANC system using FxLMS algorithm

In the diagram, $P(z)$ is the primary acoustic path between the reference noise source and error microphone. $d(n)$ is the desired signal when $x(n)$ passes through $P(z)$. To reduce the desired noise, an anti-noise $y(n)$ is generated by using the adaptive filter $W(z)$. Before $y(n)$ arrives at the sensor (error microphone), there is a secondary path stays between the adaptive filter and error microphone. Therefore, in order to obtain an accurate weight update for the adaptive filter, it is necessary to compensate for the secondary path effect $S(z)$ from $y(n)$ to $e(n)$, which means input signal is sent through an estimated transfer function $\hat{S}(z)$ before it can be used in the adaptive algorithm [1].

The residual noise $e(n)$ can be expressed as

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

where n is the time index, $d(n)$ represents desired signal (unwanted noise), $y'(n)$ represents the filtered signal of adaptive filter output $y(n)$ through the secondary path $S(z)$. Therefore, the expressions of $d(n)$, $y(n)$ and $y'(n)$ can be computed as follows.

$$d(n) = x(n) * p(n)$$

$$y(n) = w^T(n)x(n) = \sum_{i=0}^{n-1} w_i(n)x(n-i)$$

$$y'(n) = y(n) * s(n)$$

In the above equations, $p(n)$ is the impulse response of primary path $P(z)$, $s(n)$ is the impulse response of secondary path $S(z)$, $w(n) = [w_0(n)\ w_1(n) \ldots w_{L-1}(n)]^T$ is defined as the coefficient vector of $W(z)$, and $x(n) = [x(n)\ x(n) \ldots\ x(n-L+1)]^T$ is defined to be the reference signal vector of $W(z)$. L is the filter order and $*$ denotes linear convolution.

The objective of the adaptive filter is to minimize the instantaneous squared error. This is defined as the cost function $\xi(n) = e^2(n)$.

By using the steepest descent algorithm, the update equation for the coefficients of $W(z)$ is derived.

$$w(n+1) = w(n) + \mu x'(n)e(n),$$

where $x'(n)$ is filter reference signal through secondary path estimate $\hat{S}(z)$.

$$x'(n) = \hat{s}(n) * x(n)$$

If $\hat{S}(z) = S(z)$, Figure 2 can be simplified to Figure 3.

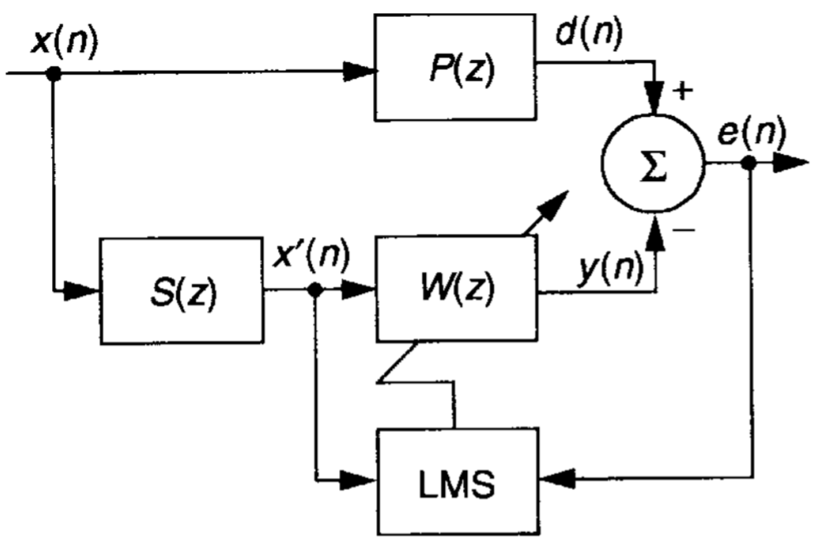

Figure 3. Equivalent diagram of Figure 2 when $\hat{S}(z) = S(z)$

One important consideration when using this method is to ensure that the convergence rate must be slow, i.e. when the step size μ is small. The accuracy impact of secondary path estimate will be discussed in the Section 3.5.

2.2. Feedback FxLMS

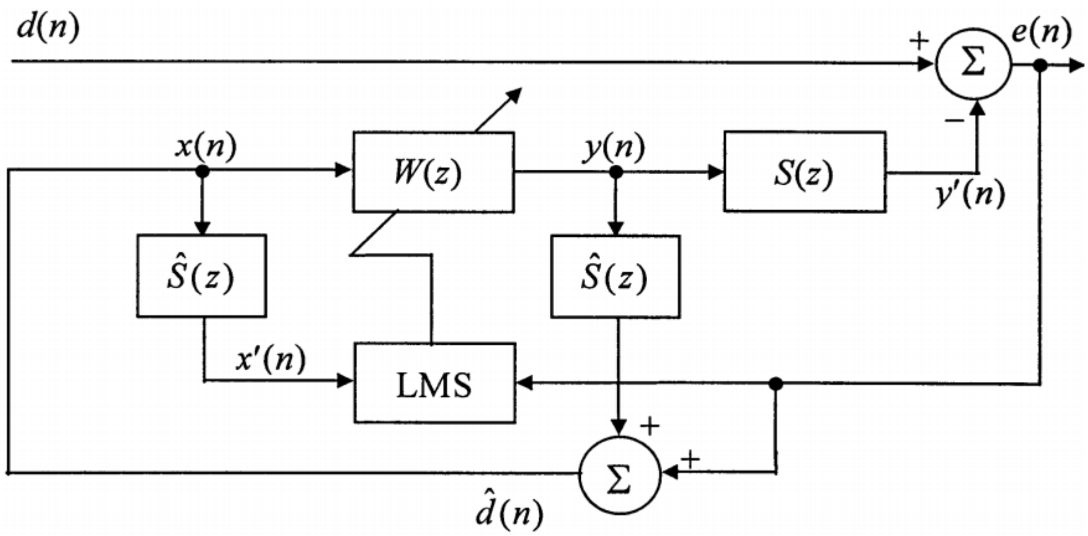

Figure 4. Block diagram of feedback ANC system using FxLMS algorithm

Different from feedforward control, feedback control does not require a reference microphone. Instead, this feedback FxLMS algorithm synthesizes (regenerates) the reference signal $x(n)$ based only on the adaptive filter output and error signal. Therefore, $x(n)$ is generated internally and can be expressed as

$$x(n) \approx d(n) = e(n) + y(n) * \hat{s}(n)$$

Weight updates equation is the same with feedforward case.

3. Results and discussion

3.1. Simulation construction

In terms of implementing FxLMS algorithm into MATLAB, it contains the following steps.

Step 1: Set up the sampling frequency and simulation duration.

Step 2: Since the primary and secondary path are unknown, it is necessary to make an assumption of these two unknown paths. Besides, although the primary and secondary path impulse responses are not essential for the simulation, it is desirable to make it as close to a real-world impulse response as possible (i.e. an exponential decay). The primary and secondary path length are dependent on sampling frequency and duct length. In the simulation, sound speed is known as 340 m/s and sampling frequency was set to be 8000 per second; distance between reference microphone and error microphone was set to be 1 meter and loud speaker was assumed right in the middle of the duct. In other words, primary path order and secondary path order were set to be 24 and 12 respectively.

Step 3: Estimate secondary path $S(z)$. In this step (offline modelling), it contains the following sub-steps.

a) Generate a white noise signal.
b) Use FIR filter to model the output through secondary path $S(z)$.
c) Send it to the actuator and measure it at the sensor position.
d) Start the identification process.
e) Set up learning step μ and apply least mean square algorithm.
f) Plot the identification error and coefficients of both secondary path $S(z)$ and secondary path estimate $\hat{S}(z)$ to check if estimation matches.

Step 4: Apply FxLMS algorithm in the system for adaptive active noise control. Essentially, it is an iterative process of 'measure,' 'control', and 'adjust' sample by sample. In this step, it contains the following sub-steps.

a) Generate a noise source; it could be either broadband noise or narrowband noise.
b) Use FIR filter to model the output through primary path $P(z)$.
c) Measure the arriving signal of noise source at the sensor position.
d) Initiate the system and apply FxLMS algorithm.
e) Plot the magnitudes of residual noise, desired noise and observe the performance.

Step 5: To further analysis and evaluate how effective is the algorithm (performance measure), there are two aspects.

a) Slice the entire sample points into eight blocks and do Discrete Fourier Transform (FFT in MATLAB) on each block for both desired signal and error signal, then compute their ratio and convert into decibels. By doing this process, we can have a general view of the noise reduction on each frequency for each block. The corresponding plots (Figure 7) and analysis will be shown in Section 3.2.
b) To quantify the system performance (effect of the algorithm), mean noise reduction (MNR) is introduced. It is defined as $MNR = 10 \log_{10}(\frac{\sum_{n}^{n+100} e^2(n)}{\sum_{n}^{n+100} d^2(n)})$. One hundred independent

trials run separately for every simulation condition. Plots of the mean result of 100 simulations will be shown and analyzed in Section 3.2, 3.3 and 3.4.

3.2. Broadband feedforward FxLMS system

3.2.1. Qualitative analysis of ANC performance in time and frequency domain

In the simulation, adaptive filter order L was chosen to be 16. Primary and secondary path impulse responses are shown in Figure 5. Noise source was selected as white noise with mean zero, variance one. Background noise was added to the desired noise and with a magnitude of -30dB of the original noise source. Step size μ was set as 0.15 for applying FxLMS algorithm and 0.1 for secondary path estimate (offline modelling).

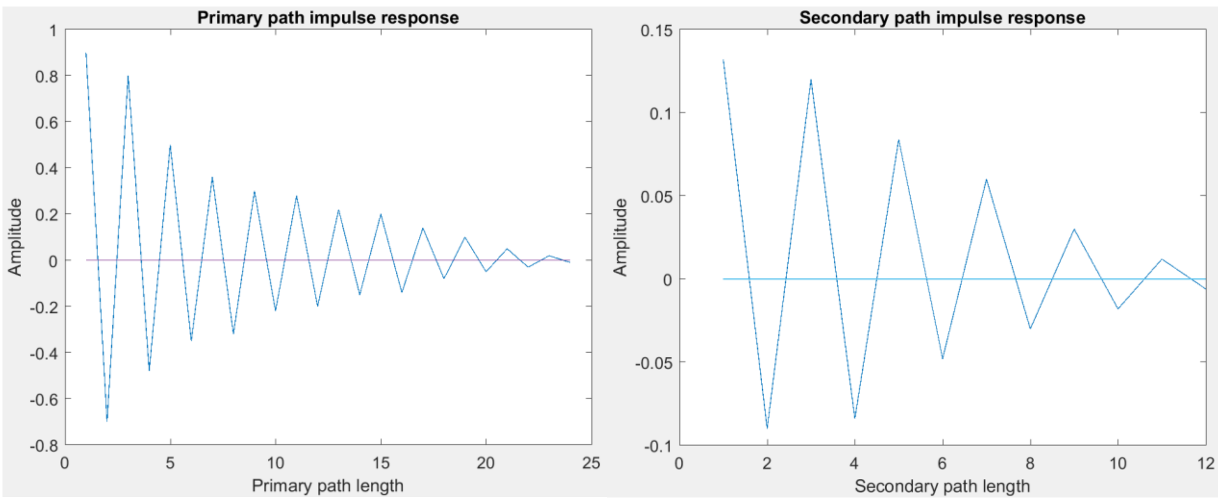

Figure 5. Impulse response of primary path (left); Impulse response of secondary path (right)

As can be seen in Figure 6 (left), the identification error tends to be zero after 1000 iterations; the coefficients of secondary path $S(z)$ and secondary path estimate $\hat{S}(z)$ are almost identical (there are slight difference when zooming in the plot in MATLAB), which means the secondary path estimate process is applicable and satisfying. Effects of inaccuracy of secondary path estimate will be discussed in Section 3.5. In Figure 6, the lower plot shows that noise signal $d(n)$ and control signal $y'(n)$ share a considerable overlapping except for a few initial sample points (1-500 approximately). The difference between $d(n)$ and $y'(n)$ is $e(n)$, and its magnitude plot with respect to discrete time index n is shown in the upper plot of Figure 6. From the plot, it is concluded that the algorithm starts converging at the very beginning and arrives to steady state convergence after around 2000 iterations. This conclusion will be compared with the frequency analysis result and performance measure in MNR.

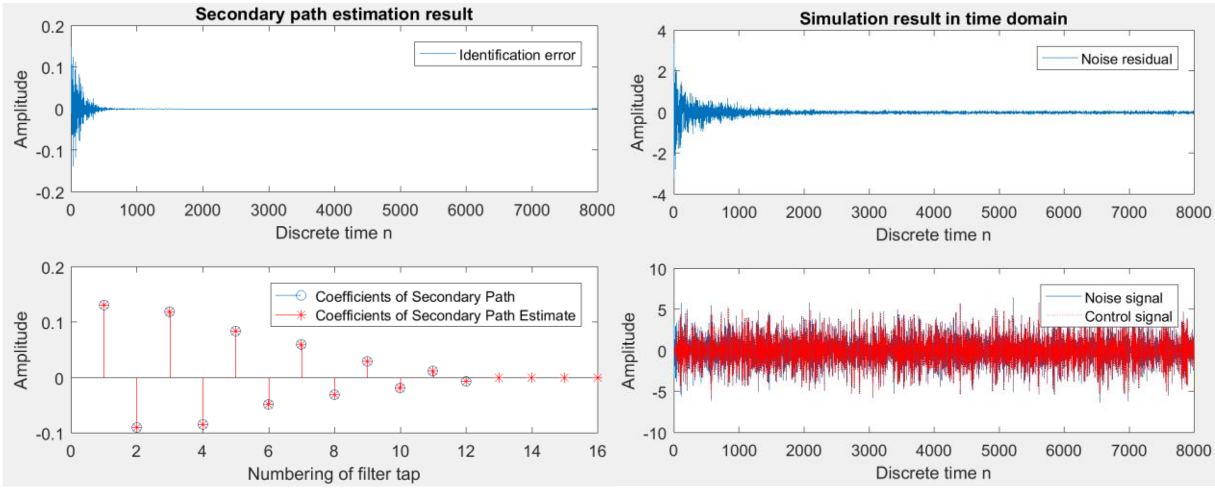

Figure 6. Secondary path estimate result (left); System simulation result in time domain (right)

The y-axis of Figure 7 is defined as $20\log_{10}\frac{FFT(e)}{FFT(d)}$, as it shows how effective of the system on every frequency for the noise source. In other words, the ratio in decibel of residual noise and desired noise can be computed using Fast Fourier Transform in MATLAB in order to analysis the system performance in frequency domain. The entire sample points are distributed equally by 8 sections. From the first section (1-1000 sample points), most of the frequency are lower than 0 dB while a minority of them are above 0, which illustrates that the algorithm starts converging. From the second section (1001-2000 sample points), frequencies which above 0 dB still exist but much less than the first section, meaning that the algorithm is still converging to its steady state. From the third and fourth section, it is found that every frequency on the spectrum is below 0 dB, therefore, the algorithm has converged to its steady state from section 3.

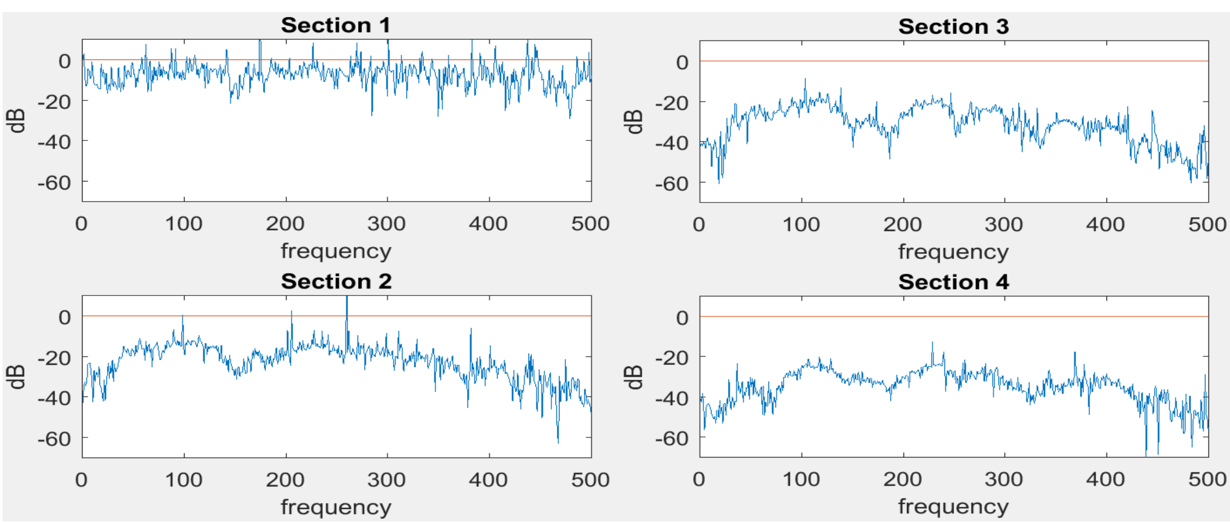

Figure 7. Frequency response of 1-1000 sample points (upper left), 1001-2000 sample points (lower left), 2001-3000 sample points (upper right), 3001-4000 sample points (lower right)

3.2.2. Quantitative analysis of ANC performance by using MNR in frequency domain

So far, we have qualitatively analyzed the ANC performance. To quantify the algorithm effectiveness, mean noise reduction (MNR) is used here to analyze the system performance measure more accurately. From Figure 8, it can be seen that MNR is -30.49 dB when n = 2000 while the steady state value is around -33.5 dB. Considering the error difference between the two is within 10%, accordingly, it is very difficult to identify this difference from the qualitative analysis in time and frequency domain. Therefore, the conclusion we have drawn before is generally correct but not accurate and precise. The real steady state starts from 4000 iterations, however, performance behaviors from 2000 iterations can be regarded as in the neighborhood of stability.

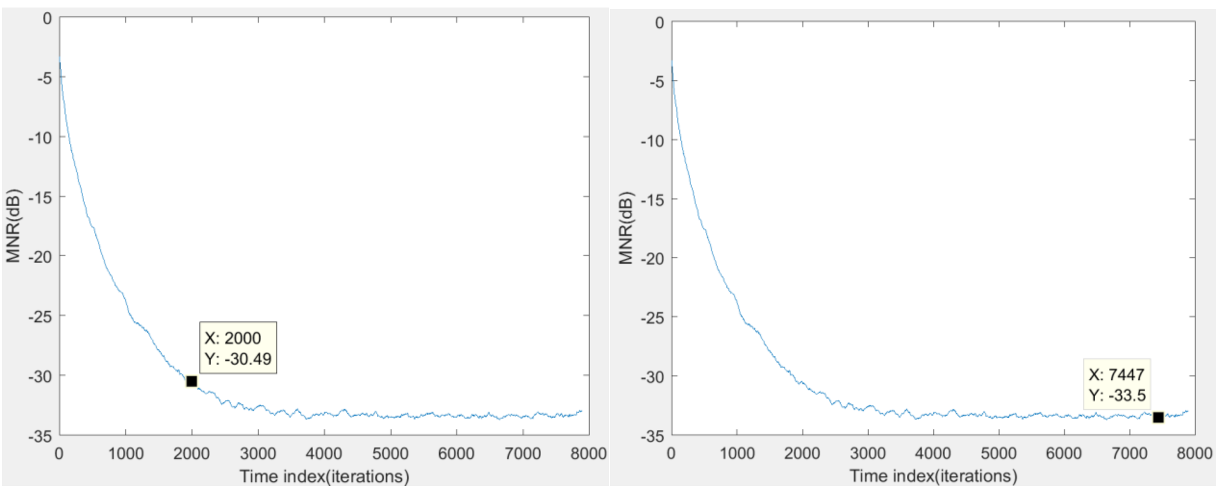

Figure 8. Mean MNR of 100 independent runs with data cursor on n = 2000(left); Mean MNR of 100 independent runs with data cursor on steady state(right)

3.2.3. Performance analysis by changing parameters in the system

In this feedforward FxLMS ANC system, apart from input noise source, there are some other parameters which may affect the system performance measure.

3.2.3.1. Step size μ

Except for step size μ applying in the FxLMS algorithm (μ for offline modelling remains the same), all other parameters in the system were set identical with the description in Section 3.2.1. As can be seen in Figure 9, step size μ determines the stability, convergence rate and fluctuation of the performance. If μ is too large, we observe numerous huge fluctuations with a degradation of ANC performance. For example, when $\mu = 0.3$, the blue line in Figure 9 is extremely unstable and has some significant fluctuations, meaning that step size above 0.3 (e.g. $\mu = 0.31$) will cause system unstable. Therefore, 0.3 can be assumed to be the maximum value of μ; and this value was also confirmed by simulating in MATLAB. If μ is too small, the convergence rate will be extremely slow. For example, when $\mu = 0.08$, the green line in Figure 9 has not converged to its steady state yet at the end of x-axis. Therefore, by increasing step size μ, the performance of ANC is degraded and the system is more unstable although convergence rate is faster. However, for the system, there exists a step size μ, which has both relatively decent convergence rate and ANC performance in steady state. Based on the research done by Ardekani in 2010 [13], the optimum step size should be $0.5\mu_{max}$, which is 0.15 in this case. To be specific, in Figure 9, purple line shows a fast convergence speed and approximately same noise reduction performance compared to the green line (smaller step size condition).

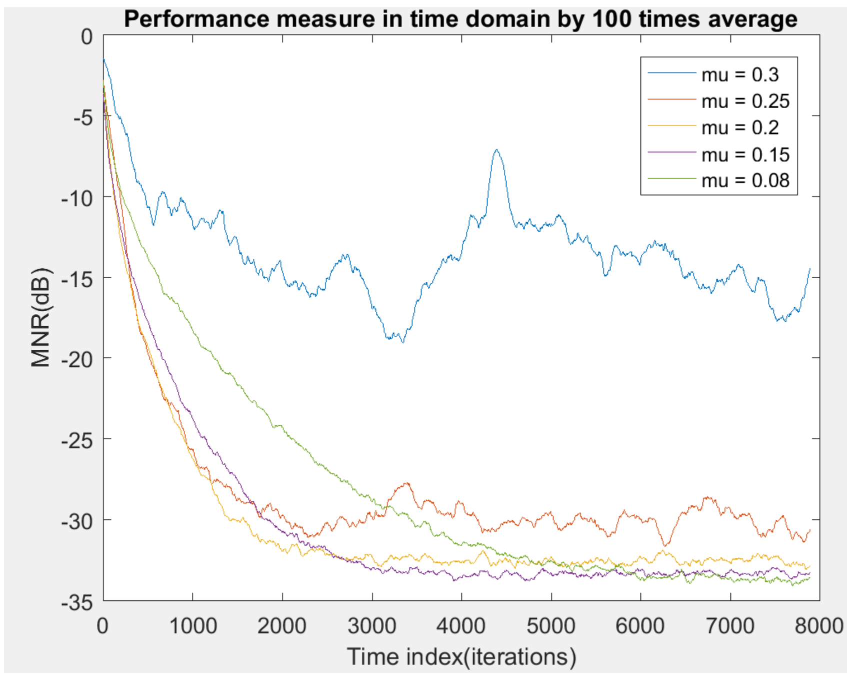

Figure 9. Performance measure in MNR with different step size μ

3.2.3.2. Adaptive filter order L

Except for adaptive filter order L, all other parameters in the system were set identical as described in Section 3.2.1. According to Figure 10, it is found that the noise reduction performance is improved as the filter length increases while the convergence rate is slower. However, the improvement becomes no more significant when L is equal to or above 16. In real world hardware implementation, it is desirable to choose the minimum filter length so as to decrease the system complexity. Therefore, the optimum filter length in this case is 16. Normally, adaptive filter coefficient vector $W(z)$ must be of sufficient order to accurately model the response of the physical system [1]. To be specific, in real-world implementation, this order is chosen to be slightly larger than the theoretical calculated order value, which is based on the distance between reference microphone and controller.

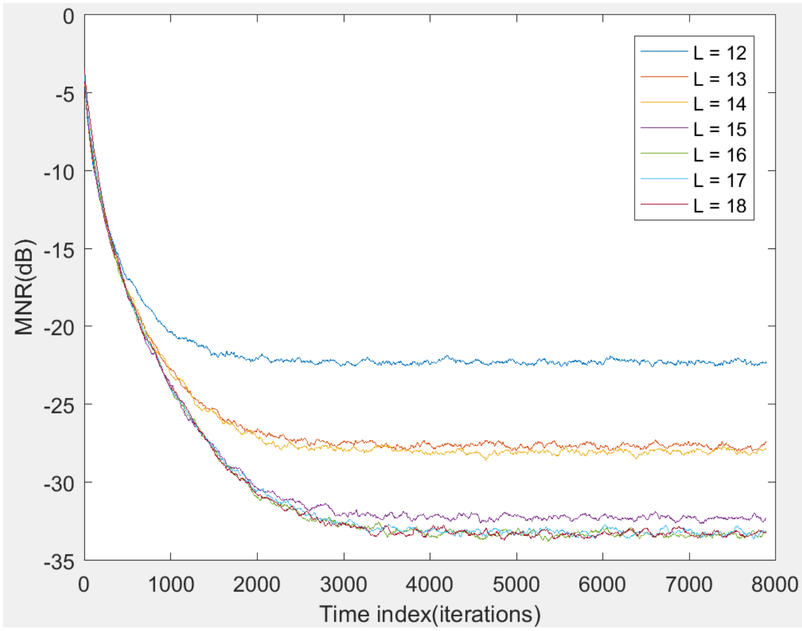

Figure 10. Performance measure in MNR with different filter order L

3.2.3.3. Background noise

Except for the magnitude of background noise, all other parameters in the system were set identical to the description in Section 3.2.1. It is observed from Figure 11 that; the noise reduction performance is degraded by using the background noise with a large magnitude (variance). However, larger background noise can make the algorithm converge faster. Normally, a magnitude of (or less than) -30dB background noise is selected as the condition for the study of other parameters.

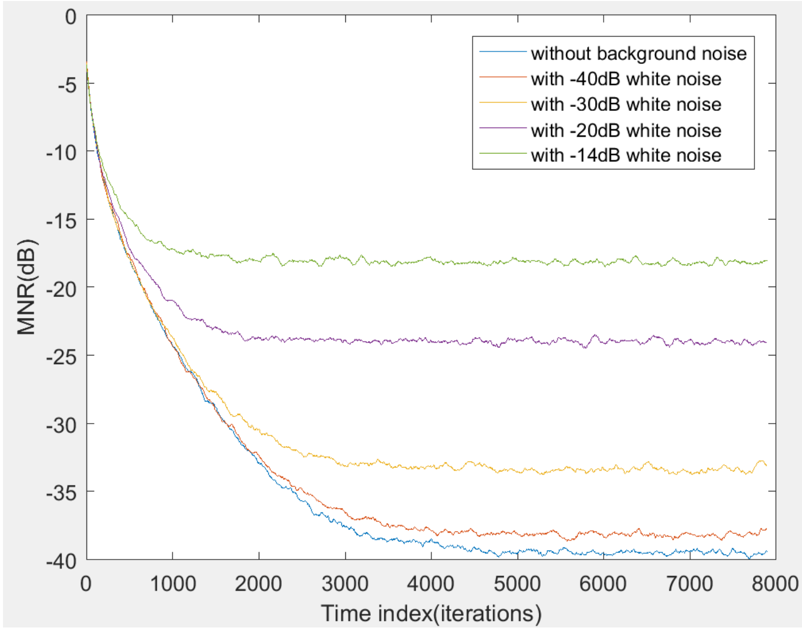

Figure 11. Performance measure in MNR with different background noise

3.3. Narrowband feedforward FxLMS system

As explained in Section 2.2, feedforward FxLMS algorithm is able to cope with not only broadband noise but also narrowband noise. In Section 3.3.1, influence on ANC performance by different parameters in the system will be illustrated and discussed.

3.3.1. Performance analysis by changing parameters in the system

3.3.1.1. Step size μ

Except for step size μ applying in the FxLMS algorithm (μ for offline modelling remains the same), all other parameters in the system were set identical with the description in Section 3.2.1. The narrowband noise source was set as two different cases in order to avoid the experimental error. In case one, the noise source was chosen as the combination of two sinusoidal noises with period of 100 and 4 respectively. In case two, the noise source was chosen as a sinusoidal noise with a period of 5. To observe Figure 12 clearly, number of iterations are adjusted; however, in the simulation, it was set as 8000.

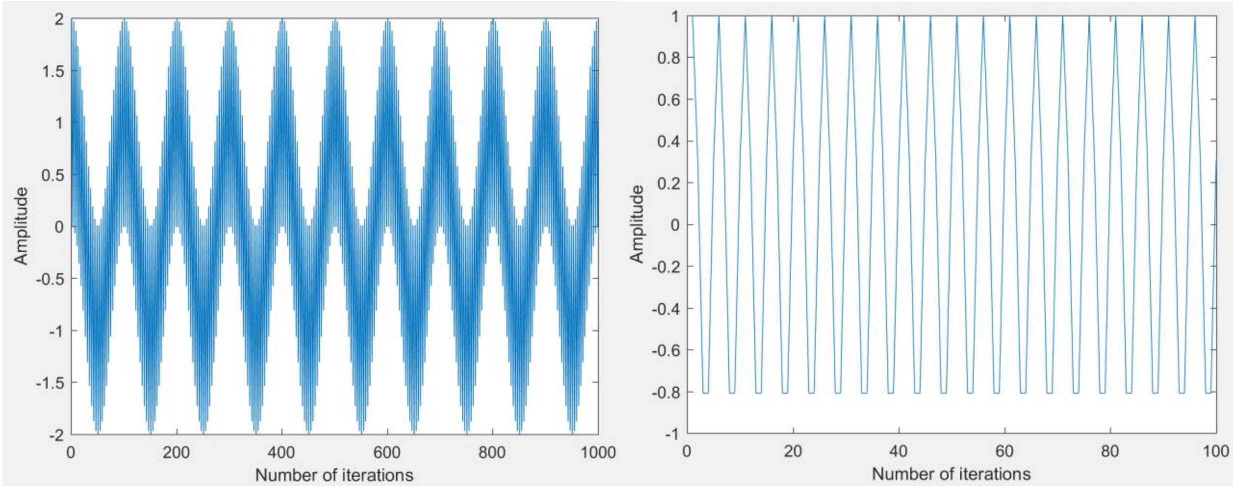

Figure 12. Case 1:cos $\left(\frac{2\pi}{100}[0{:}T-1]\right)$*+cos* $\left(\frac{2\pi}{4}[0{:}T-1]\right)$*(left); Case 2: cos* $\left(\frac{2\pi}{5}[0{:}T-1]\right)$ *(right), T is the number of iterations in MATLAB*

From Figure 13, it is found that the learning rate (step size) of the algorithm does not have a significant effect on the noise reduction in steady state. However, although the other system parameters remain the same, it is deduced that ANC performance is dependent on the noise source since case 1 and 2 have shown different values of MNR in steady state (-28dB and -22dB respectively). To further explain the relationship between input narrowband noise and ANC performance, the correlation will be shown in 3.3.1.2. It is also noted that with the increase of step size, the convergence rate becomes faster.

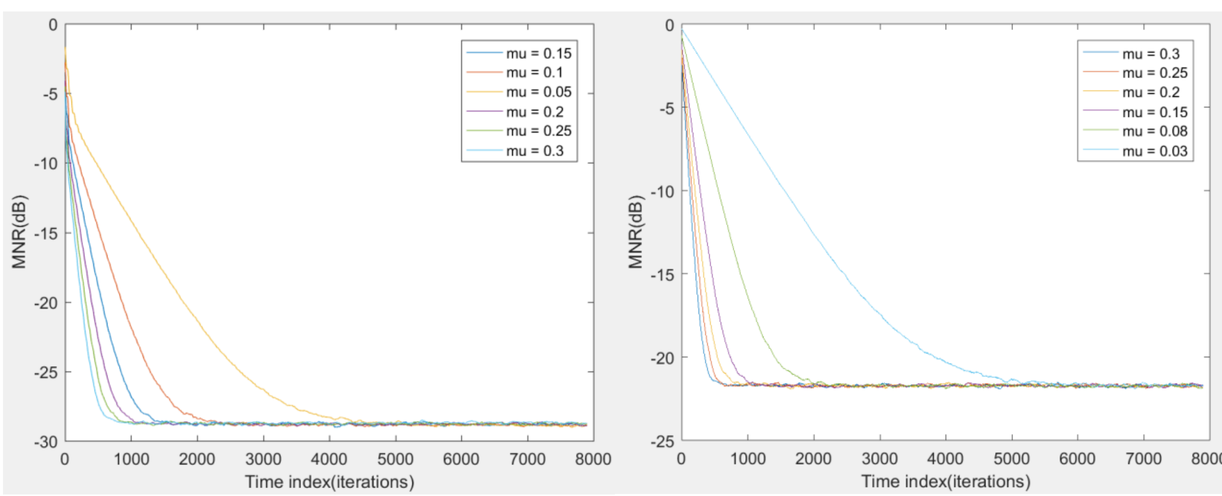

Figure 13. Performance measure in MNR with different step size μ for case 1(left); Performance measure in MNR with different step size μ for case 2(right)

3.3.1.2. Noise source

Except for the noise source, all other parameters in the system were set identical to the description in Section 3.2.1. Figure 14 shows that different noise sources result in different ANC performances.

Therefore, it is concluded that FxLMS algorithm is more effective on low-frequency narrowband noise.

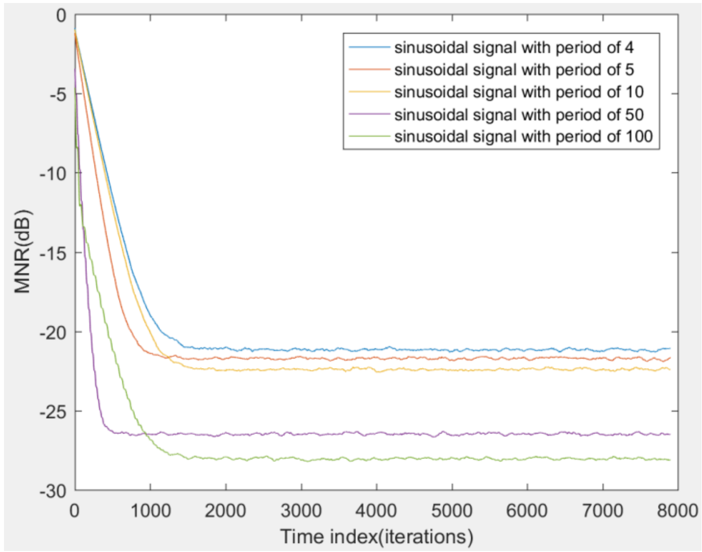

Figure 14. Performance measure in MNR with different noise source

3.3.1.3. Adaptive filter order L

Except for the adaptive filter order and noise source, all other parameters in the system were set identical to the description in Section 3.2.1. The noise source was selected as the sinusoidal signal with a period of 5. The plot of the noise source is shown in Figure 12 (right). From Figure 15, it is found that the impact of adaptive filter order on the ANC performance in steady state is negligible. L only has effect on the convergence rate. The algorithm can converge faster as the adaptive filter order increases.

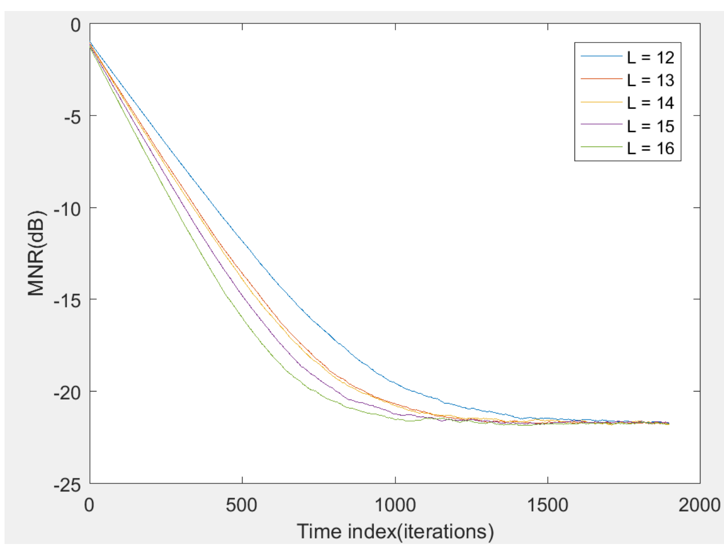

Figure 15. Performance measure in MNR with different filter order for case 1

3.3.1.4. Background noise

Except for the adaptive filter order and noise source, all other parameters in the system were set identical to the description in Section 3.2.1. The noise source is selected as the sinusoidal signal with a period of 5. The plot of the noise source is shown in Figure 12. Figure 16 shows by using feedforward FxLMS algorithm; noise reduction performance is excellent for narrowband noise without a background noise. However, if the desired noise contains a -40 dB background noise, the ANC performance will be degraded dramatically since background noise cannot be cancelled through the adaptive filter in the system. Hence, compared with Figure 11, it is concluded that FxLMS performs better on narrowband noise than broadband noise without background noise. However, the impact of background noise on the system performance for narrowband noise is much larger and more evident than broadband noise.

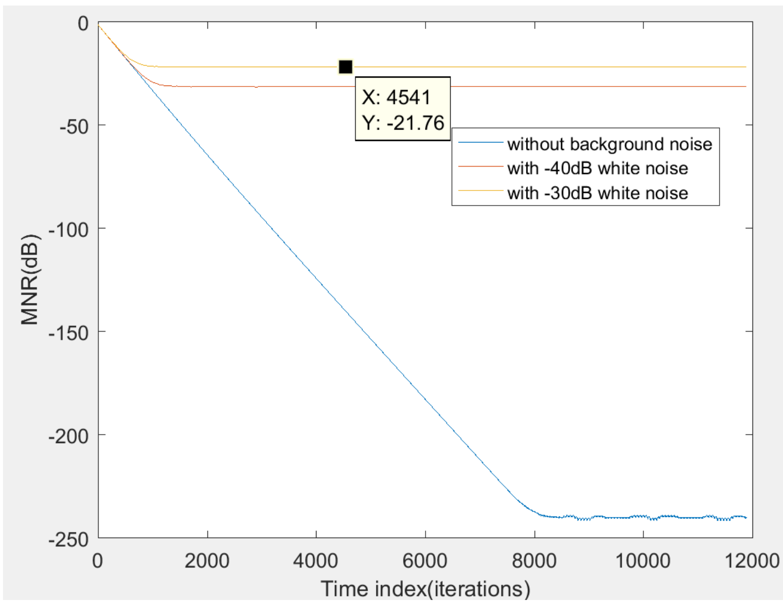

Figure 16. Performance measure in MNR with different background noise for case 1

3.4. Narrowband feedback FxLMS system

Different from feedforward control strategy, feedback control does not use reference microphone. Instead, it uses linear predictor to generate a reference signal. Therefore, feedback FxLMS algorithm is only suitable for attenuating narrowband noise. As shown in Figure 17, if input noise source is selected as the combination of broadband noise and narrowband noise, the algorithm does not converge.

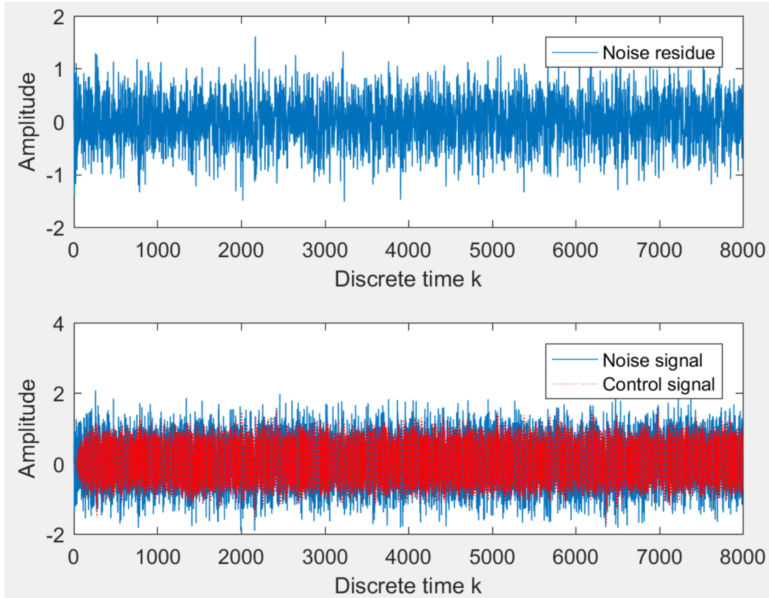

Figure 17. Simulation result in time domain when noise source X = 0.3randn (1, T) + cos ($\frac{2\pi}{5}$[0: T-1])

3.4.1. Qualitative analysis of ANC performance in time and frequency domain

Figure 18 shows the system performance in time domain for two different input noise sources (case 1 and case 2). Except for the control strategy and input noise source, all other parameters were set identical to the description in Section 3.2.1. From Figure 18, it is observed that for whatever narrowband noises, feedback FxLMS algorithm has a good effect on them. It is also noted that noise reduction in steady state for case 1 is better than case 2, meaning that the algorithm is more effective on low-frequency narrowband noise. The quantitative ANC performance is shown in Figure 19 (right), which also confirmed the observation in Figure 18. Figure 19 (left) also shows the algorithm performs perfectly on narrowband noise without background noise as the residual noise tends to be very close to zero in steady state. To be specific, as the blue line illustrated in Figure 16, the steady state MNR value is approximately -240db. It denotes that feedback control has an excellent effect on narrowband noise without background noise.

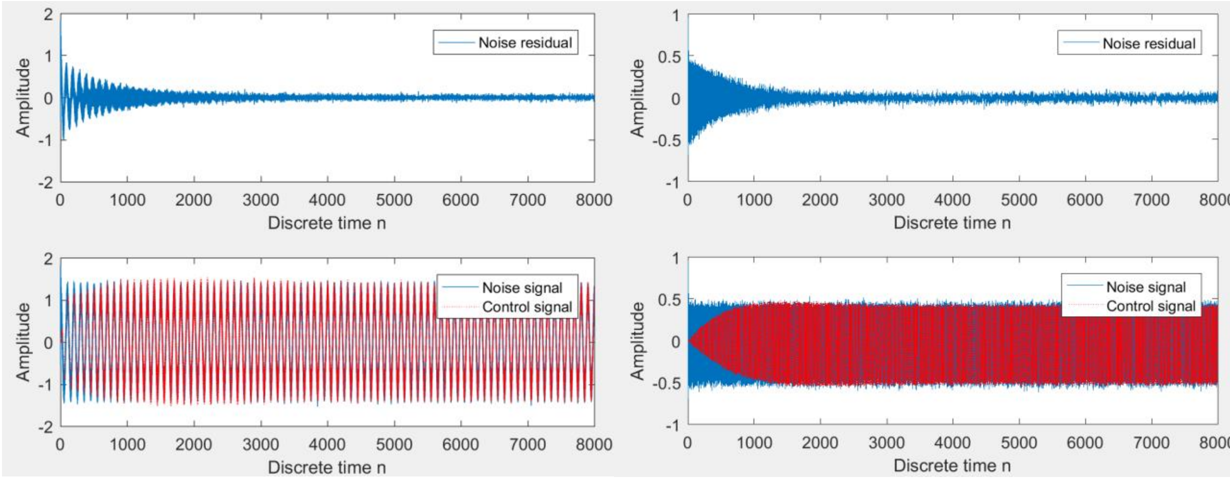

Figure 18. System simulation result in time domain for case 1 (left); System simulation result in time domain for case 2 (right)

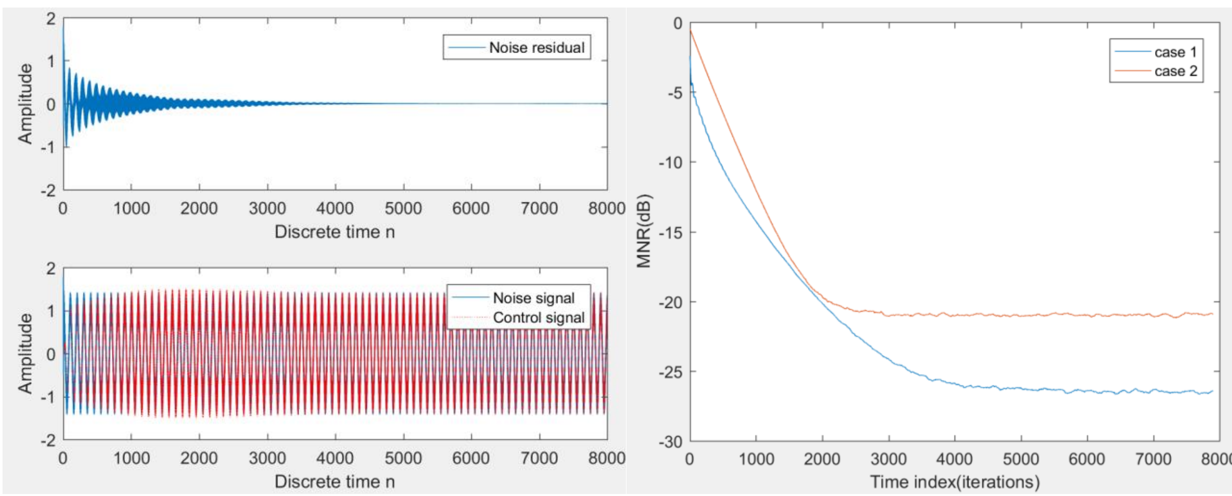

Figure 19. System simulation result in time domain for case 1 without background noise (left); Performance measure in MNR for case 1 and case 2 (right)

3.4.2. Performance analysis by changing parameters in the system

3.4.2.1. Step size μ

Except for the control strategy, input noise source and step size μ, all other parameters were set identical with the description in Section 3.2.1. Noise source was chosen as two types of narrowband noises as described in 3.3.1.1. As observed in Figure 20, step size has negligible effect on ANC performance in steady state. Convergence rate can become faster as step size increases.

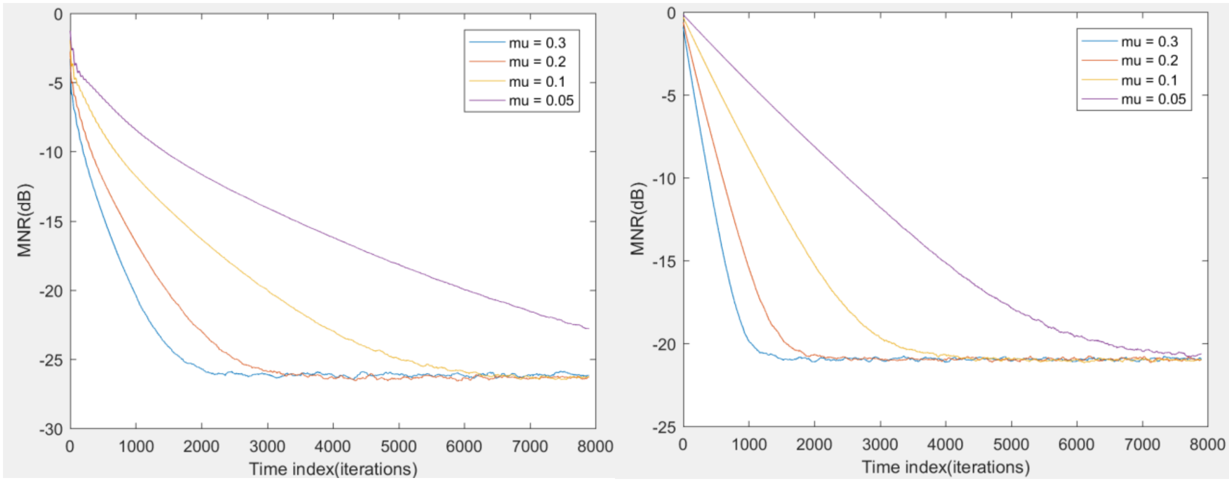

Figure 20. Performance measure in MNR with different step size μ for case 1 (left); Performance measure in MNR with different step size μ for case 2 (right)

3.4.2.2. Adaptive filter order L

Except for the control strategy, input noise source, and adaptive filter order, all other parameters were set identical to the description in Section 3.2.1. Noise source was chosen as two types of narrowband noises as described in 3.3.1.1. From Figure 21, it is founded that by reducing the adaptive filter, the performance of noise cancellation is degraded.

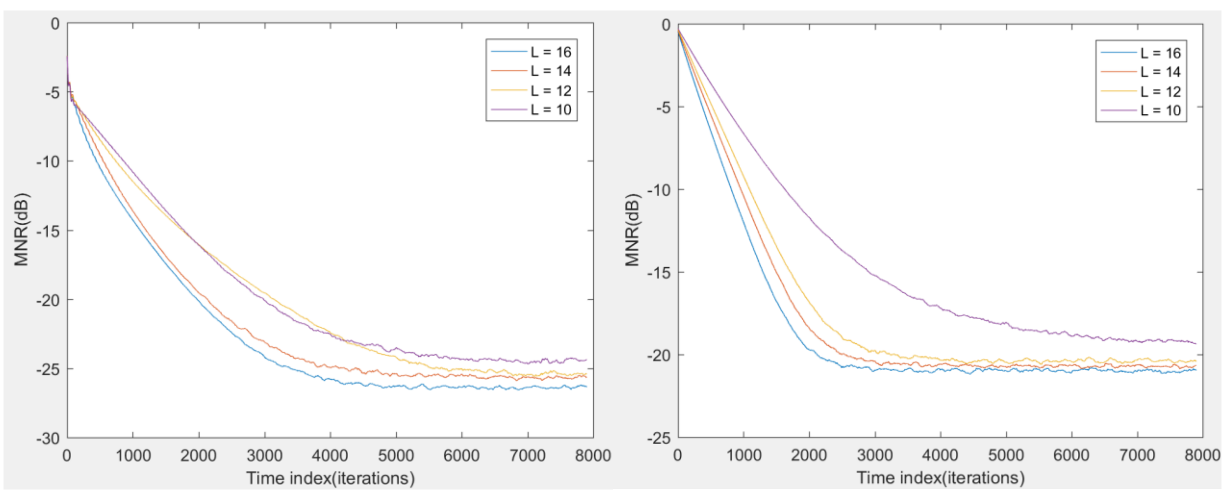

Figure 21. Performance measure in MNR with different filter order for case 1(left); Performance measure in MNR with different filter order for case 2(right)

3.4.2.3. Background noise

Figure 22 illustrates that the background noise has a significant influence on the system performance. Compared to Figure 11, such influence is more evident than broadband noise source. To further analyze the algorithm, it is founded that feedback control converges slower than feedforward control strategy. In comparison with Figure 16, two Figures were plotted under the same condition except for the control strategy. However, lines in Figure 16 converge faster than in Figure 22. Since it is necessary to use linear relationship to generate reference signal $x(n) \approx d(n) = e(n) + y(n) * \hat{s}(n)$ in feedback control, it increases the complexity of ANC system, leading to a slower convergence rate than feedforward control in FxLMS algorithm.

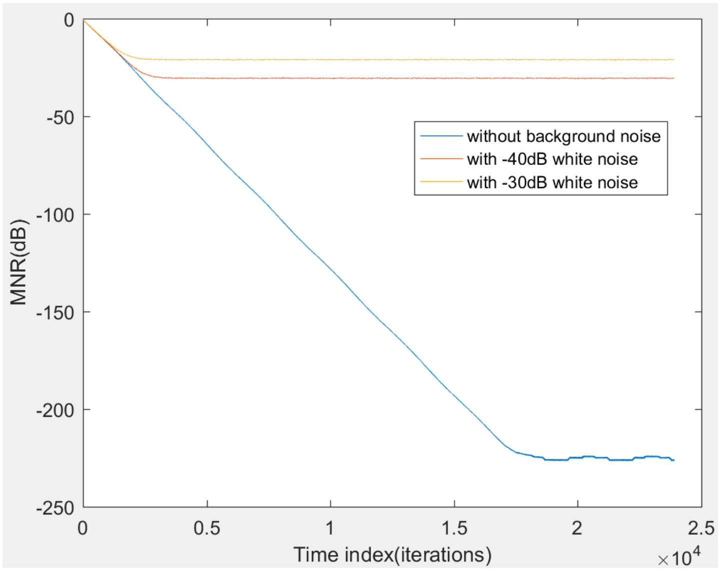

Figure 22. Performance measure in MNR with different background noise for case 2

3.5. Secondary path estimate

3.5.1. Accuracy impact

In FxLMS algorithm, it is essential and necessary to estimate the secondary path in order to compensate its effect to the ANC system. Therefore, studies on secondary path estimate is an important research area in ANC field. The author made a study on the effect of secondary path estimate accuracy to the ANC performance. To get an inaccurate estimate of secondary path, a random Gaussian number (e.g. 0.0001randn (1, T) (left) and 0.0005randn (1, T) (right) for the two cases in Figure 23 respectively) was added to the weight update equation of coefficient vector of $\hat{S}(z)$. Therefore, the secondary path effect can be observed in MNR plot in Figure 24.

Figure 23 presents two inaccurate secondary path estimate results. Compared with previous secondary path estimate result which is shown in Figure 6 (left), both identification error and coefficients of secondary path estimate show a clear inaccuracy than before. Therefore, this inaccuracy can result in different ANC system performances and may cause system unstable.

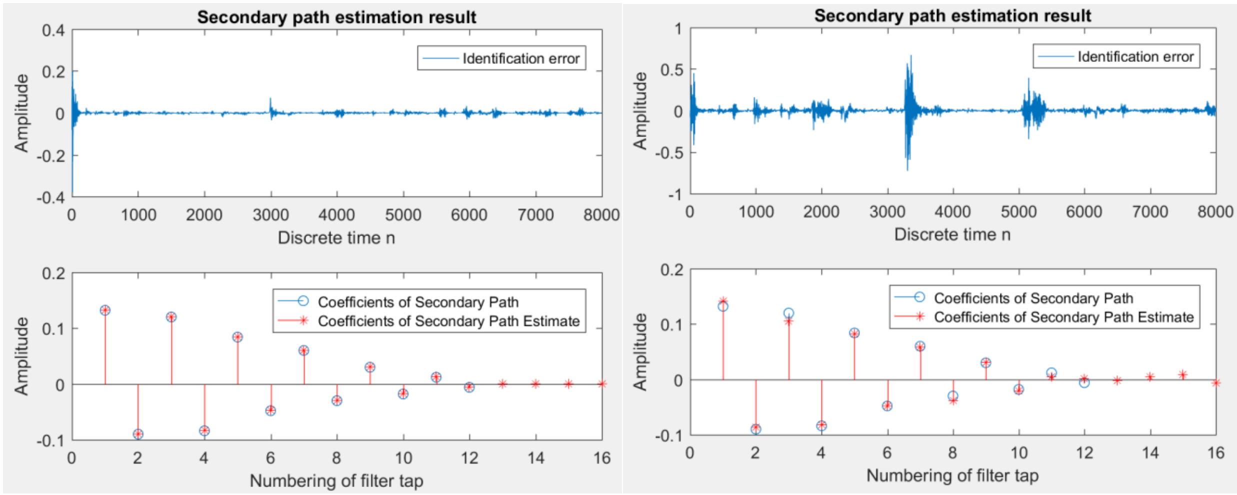

Figure 23. Inaccurate secondary path estimate result (blue line in Figure 24) (left); Very inaccurate secondary path estimate result (purple line in Figure 24) (right)

As can be seen in Figure 24, the previous secondary path estimate shares a similar performance with the theoretical perfect secondary path estimate (i.e. $S(z) = \hat{S}(z)$). This result has testified that the secondary path estimate in the algorithm is superlative and satisfying. However, with a slight change in the algorithm which makes the coefficient of secondary path estimate imperfectly matched with true secondary path coefficient, the ANC performance will be degraded as shown in Figure 24 by the blue line. Purple line in Figure 24 shows that a very inaccurate offline modelling result can inevitably cause the system unstable. Thus, it is concluded that an initial training of secondary path (offline modelling) prior to the operation of the ANC system is really important [1], and its estimate accuracy can have a significant effect on the ANC performance. Inaccurate estimate will lead to instability of the ANC system and failure to reduce the noise. For some applications, the secondary path may be time varying. Therefore, it is desirable to estimate the secondary path online when the ANC is in operation [1].

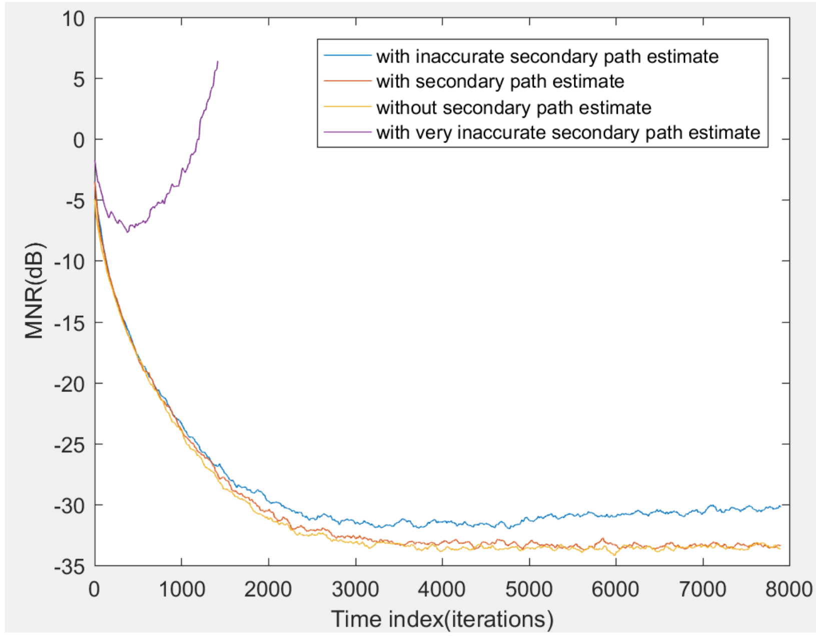

Figure 24. Performance measure in MNR with different secondary path estimate results

3.5.2. Probability of instability

The author has found that even with the same inaccuracy secondary path estimate result, the system will become unstable frequently. Figure 25 shows the performance measure in MNR with the same inaccurate secondary path estimate by six independent simulations, and each simulation is the average of 100 independent runs. The inaccurate secondary path estimate result used for the simulation here is shown in Figure 23 (left). It is founded that even though the system performance keeps stable sometimes, the ANC performance still has a high probability to become unstable if there is an error added in the offline modeling. This is because the error added in the offline modeling is totally random and stochastic, thus resulting in different secondary path estimate results and system performances. System performances using such inaccurate offline modeling could become sometimes stable (green and yellow line in Figure 25) or unstable (red and dark blue lines, etc. in Figure 25). However, a very inaccurate secondary path estimate result such as the one shown in Figure 23 (right) will always cause instability of the ANC system and never show an expected outcome of performance measure.

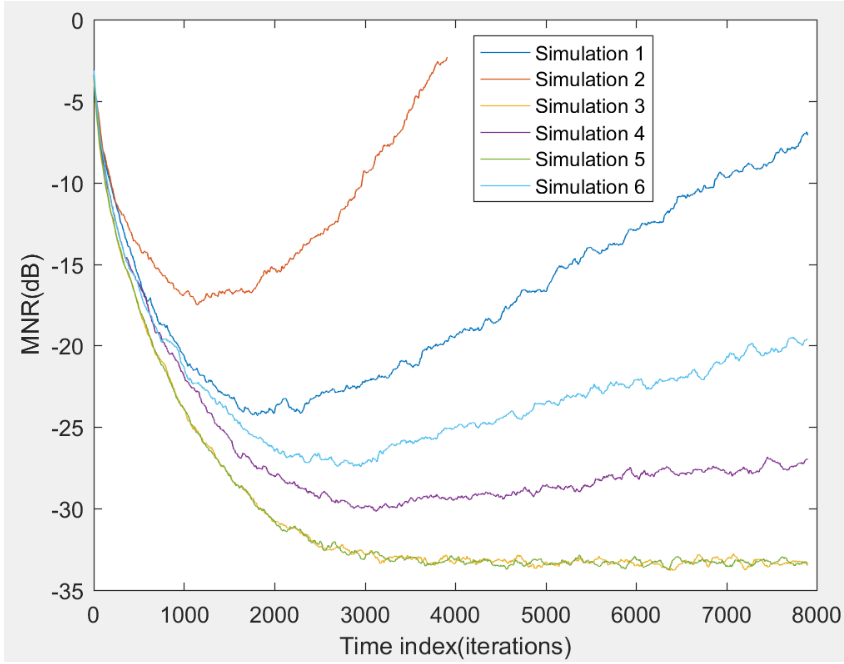

Figure 25. Performance measure in MNR with inaccurate secondary path result by 6 independent simulations

4. Conclusion

Active noise control cancels unwanted noise by generating anti-noise with equal amplitude and opposite phase of primary noise by using an adaptive filter. This report has concluded the history of ANC development, presented the most widely used adaptive algorithm (Filtered-x LMS algorithm), implemented and analyzed the algorithm for both feedforward and feedback control strategy through extensive simulations by MATLAB. Two different noise sources have been selected as the input for two control strategies. The main leading parameters such as step size, adaptive filter order, background noise, secondary path estimate accuracy and noise source have been evaluated and analyzed. Effects of primary path impulse response, secondary path length, upstream propagation of anti-noise are not discussed in the report due to their uncertainty and complexity, and these research aspects could be the future work of the project. The author also has illustrated how to find the optimum step size and adaptive filter order for ANC system based on the observation and analysis from the performance measure plot. It is concluded that feedforward control gives good performance on both broadband and narrowband noise while feedback can only cope with narrowband noise. The optimum step size should equal to half of μ_{max}, and the optimum adaptive filter order should be sufficient to accurately model the response of the physical system. All the simulations were also run by MATLAB internal function `dsp.LMSFilter` (see Appendix 4.6) and similar ANC performances were derived using such function.

References

[1] S. M. Kuo and D. R. Morgan, "Active noise control: a tutorial review," *Proceedings of the IEEE*, vol. 87, pp. 943-973, 1999.

[2] L. Paul, "Process of silencing sound oscillations," ed: Google Patents, 1936.

[3] B. Somek, M. Dadić, and M. Maletić, "Active noise control in ducts," *AUTOMATIKA: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, vol. 42, pp. 5-12, 2001.

[4] M. Asteborg and N. Svanberg, "Implementation Considerations for Active Noise Control in Ventilation Systems," 2006.

[5] M. S. R. Muhammad Moazzam, "Performance Evaluation of Different Active Noise Control (ANC) Algorithms for Attenuating Noise in a Duct," Master of Science in Electrical Engineering Specialization Signal Processing, Department of Applied Signal Processing, Blekinge Institute of Technology, Karlskrona, Sweden., 2013.

[6] N. Nakrani and N. Patel, "Feed-Forward and Feedback Active Noise Control System Using FxLMS Algorithm for Narrowband and Broadband Noise," in *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, 2012, pp. 577-580.

[7] B. Huang, Y. Xiao, J. Sun, and G. Wei, "A variable step-size FXLMS algorithm for narrowband active noise control," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, pp. 301-312, 2013.

[8] S. M. Kuo and D. Morgan, *Active noise control systems: algorithms and DSP implementations*: John Wiley & Sons, Inc., 1995.

[9] Ford. (2015). *HOW NEW FORD TECHNOLOGY CAN MAKE YOUR CAR WORK LIKE A GIANT PAIR OF NOISE-CANCELLING HEADPHONES*. Available: https://media.ford.com/content/fordmedia/feu/en/news/2015/11/11/how-new-ford-technology-can-make-your-car-work-like---a-giant-pa.html

[10] BOSE. (2016). *EXPLORE OUR HEADPHONES*. Available: https://www.bose.com.au/en_au/products/headphones.html

[11] Y. Kajikawa, W.-S. Gan, and S. M. Kuo, "Recent advances on active noise control: open issues and innovative applications," *APSIPA Transactions on Signal and Information Processing*, vol. 1, p. e3, 2012.

[12] E. Kim, B. Kim, and S. Lee, "Active noise control in a duct system based on a frequency-estimation algorithm and the FX-LMS algorithm," *International Journal of Automotive Technology*, vol. 14, pp. 291-299, 2013.

[13] I. T. Ardekani and W. H. Abdulla, "Theoretical convergence analysis of FxLMS algorithm," *Signal Processing*, vol. 90, pp. 3046-3055, 2010.

Appendix

The MATLAB coding part was done with the help of an original code from Agustinus Oey.

4.1. Feedforward FxLMS algorithm

```
function result_time_domain=fxlms_time_domain(k)
 %simulation duration
fs=8000;
t=0:1/fs:k;
T=length(t)-1;
% set primary path
Pw=[0.9 -0.7 0.8 -0.48 0.5 -0.35 0.36 -0.32 0.3 -0.22 0.28 -0.2...
    0.22 -0.15 0.2 -0.14 0.14 -0.08 0.1 -0.05 0.05 -0.03 0.02 -0.01];
Sw=Pw(13:end)*0.6;
%% secondary path estimation (offline modelling)
x_iden = randn(1,T); % generate a white noise
% send it to the error microphone, i.e. the identification error
y_iden = filter(Sw,1,x_iden);
%% offline modelling
L = 16; %adaptive filter order
S_hat_x = zeros(1,L); %define the reference signal vector of  S(z)_hat
S_hat_w = zeros(1,L); %define the coefficient vector of the S(z)_hat
e_iden = zeros(1,L); %define the identification error in the estimation process of...
% error = 0.0001*randn(1,T); %for produce inaccurate offline modelling
% apply LMS algorithm
mu_1 = 0.1;
for n = 1:T %time index
    S_hat_x = [x_iden(n) S_hat_x(1:L-1)]; %update the reference signal vector
    Shy = sum(S_hat_x.*S_hat_w); % calculate output of Secondary path estimate
    e_iden(n)=y_iden(n)-Shy;    % calculate error
    S_hat_w=S_hat_w + mu_1*e_iden(n)*S_hat_x;   % adjust the weight
%     S_hat_w=S_hat_w + mu_1*e_iden(n)*S_hat_x + error(n); %for produce
%     inaccurate offline modelling
end
```

```
%plotting
figure
subplot(2,1,1)
plot([1:T], e_iden)
ylabel('Amplitude');
xlabel('Discrete time n');
legend('Identification error');
title('Secondary path estimation result')
subplot(2,1,2)
stem(Sw)
hold on
stem(S_hat_w, 'r*')
ylabel('Amplitude');
xlabel('Numbering of filter tap');
legend('Coefficients of Secondary Path', 'Coefficients of Secondary Path Estimate')
```

```
%% Then we can propagate the seconday path estimate into the entire system
X = randn(1,T); % define the input noise(source signal)
% X=cos(2*pi*(1/100)*[0:T-1])+cos(2*pi*(1/4)*[0:T-1]); %narrowband noise
n  = 0.03*randn(1,T);  % background noise
d = filter(Pw,1,X)+n; % desired signal (signal at the error microphone)
% Initialization of Active Noise Control
Cx = zeros(1,L); %reference signal vector of W(z)
Cw = zeros(1,L); %coefficient vector of W(z)
Sx = zeros(size(Sw));% use for convolution
e = zeros(1,T); %define error value
X_s = zeros(1,L); %filtered X
%% Apply FXLMS
mu = 0.15;
for n = 1:T %time index
    Cx = [X(n) Cx(1:L-1)]; %update the reference signal vector
    Cy = sum(Cx.*Cw); %Adaptive filter output
    Sx = [Cy Sx(1:length(Sx)-1)]; %propagate to secondary path
    y = sum(Sx.*Sw); %output passing through secondary path, to the reference microphone
    e(n) = d(n) - y; %calculating the residual error
    S_hat_x  = [X(n) S_hat_x(1:L-1)]; %update the vector signal of secondary path estimate
    X_s = [sum(S_hat_x.*S_hat_w) X_s(1:L-1)];
    Cw = Cw + mu*e(n)*X_s; %update the coefficiet vector of adaptive filter
end
```

```
%% Time domain plot
% Plotting
figure
subplot(2,1,1)
plot([1:T],e)
ylabel('Amplitude');
xlabel('Discrete time n');
legend('Noise residual');
title('Simulation result in time domain');

subplot(2,1,2)
plot([1:T],d)
hold on
plot([1:T],d-e,'r:')
ylabel('Amplitude');
xlabel('Discrete time n');
legend('Noise signal','Control signal');
%% Timde dOmain dB
% ratio = e.^2;
ratio = zeros(1,T-100);
for i = 1:T-100
    ratio(i) = (sum(e(i:(i+99)).^2))/(sum(d(i:(i+99)).^2));
end
dB = 10*log10(ratio);
result_time_domain=dB;
end
```

4.2. Script for plotting system performance in MNR using feedforward FxLMS

```
fs = 8000;
k = 1;
T = fs*k;
num = 100;

result=zeros(1,T-100);
for i = 1:num
    result = fxlms_time_domain(k)+result;
end
r=result/num;
plot(1:T-100,r)
xlabel('Time index(iterations)');
ylabel('MNR(dB)');
% title('Performance measure in time domain by 100 times average');
```

4.3. Feedback FxLMS algorithm

```
function result_time_domain=feedback_FxLMS(k)
fs=8000;
t=0:1/fs:k;
T=length(t)-1; %simulation duration
% set primary path
Pw=[0.9 -0.7 0.8 -0.48 0.5 -0.35 0.36 -0.32 0.3 -0.22 0.28 -0.2...
    0.22 -0.15 0.2 -0.14 0.14 -0.08 0.1 -0.05 0.05 -0.03 0.02 -0.01];
Sw=Pw(13:end)*0.6;% Sw=Pw*0.25;
%% secondary path estimation (offline modelling)
x_iden = randn(1,T); % generate a white noise
% send it to the error microphone, i.e. the identification error
y_iden = filter(Sw,1,x_iden);
% y_iden = conv(Sw,x_iden);
%% offline modelling
L = 16; %adaptive filter order
S_hat_x = zeros(1,L); %define the reference signal vector of  S(z)_hat
S_hat_w = zeros(1,L); %define the coefficient vector of the S(z)_hat
e_iden = zeros(1,L); %define the identification error
% apply LMS algorithm
mu_1 = 0.1;
for n = 1:T %time index
    S_hat_x = [x_iden(n) S_hat_x(1:L-1)]; %update the reference signal vector
    Shy = sum(S_hat_x.*S_hat_w); % calculate output of Secondary path estimate
    e_iden(n)=y_iden(n)-Shy;     % calculate error
    S_hat_w=S_hat_w + mu_1*e_iden(n)*S_hat_x;    % adjust the weight
end
```

```
%% Then we can propagate the seconday path estimate into the entire system
X=cos(2*pi*(1/4)*[0:T-1])+cos(2*pi*(1/100)*[0:T-1]); %feedback can only deal with narrowband
% X=cos(2*pi*(1/5)*[0:T-1]);
% X=0.3*randn(1,T)+cos(2*pi*(1/5)*[0:T-1]);
n  = 0.03*randn(1,T);  % background noise
d = filter(Pw,1,X)+n; % desired signal (signal at the error microphone)
% Since we do not have any reference signal, we first define a vector for
% reference signal
x_ref = zeros(size(X));
% Initialization of Active Noise Control
Cx = zeros(1,L); %reference signal vector of W(z)
Cw = zeros(1,L); %coefficient vector of W(z)
Cyx = zeros(1,L); %filtered adaptive filter output
Sx = zeros(size(Sw));%used for convolution
e = zeros(1,T); %define error value
X_s = zeros(1,L); %filtered X
%% send the first sample of control signal
n = 1;
Cx = [x_ref(n) Cx(1:L-1)];
Cy = sum(Cx.*Cw); %adaptive filter output
Sx = [Cy Sx(1:length(Sx)-1)];
e(n) = d(n) - sum(Sx.*Sw);
```

```
%%
mu = 0.15;
for n = 2:T
    Cyx = [Cy Cyx(1:L-1)];
    x_ref(n) = e(n-1) + sum(Cyx.*S_hat_w);

    Cx = [x_ref(n) Cx(1:L-1)];
    Cy = sum(Cx.*Cw);
    Sx = [Cy Sx(1:length(Sx)-1)];
    e(n) = d(n) - sum(Sx.*Sw);

    S_hat_x  = [X(n) S_hat_x(1:L-1)]; %update the vector signal of secondary path estimate
    X_s = [sum(S_hat_x.*S_hat_w) X_s(1:L-1)];
    Cw = Cw + mu*e(n)*X_s; %update the coefficiet vector of adaptive filter
end

ratio = zeros(1,T-100);
for i = 1:T-100
    ratio(i) = (sum(e(i:(i+99)).^2))/(sum(d(i:(i+99)).^2));
end

dB = 10*log10(ratio);
result_time_domain=dB;

end
```

The screen shot of codes only contains the algorithm; the time domain plot part is the same with feedforward case.

Script for plotting system performance in MNR using feedback FxLMS is the same with Appendix 4.2.

4.4. Script for analyzing system performance in frequency domain

The following script slices the entire simulation duration into eight sections using FFT. The script is compatible with both feedback and feedforward control.

```
figure
interval = [(1:0.125*T);(0.125*T:0.25*T-1) ;(0.25*T:0.375*T-1); (0.375*T:0.5*T-1);...
    (0.5*T:0.625*T-1);(0.625*T:0.75*T-1);(0.75*T:0.875*T-1);(0.875*T:T-1)];
for i = 1:8
    F1 = fft(d(interval(i,:)))./length(interval(i,:));
    df = (1/T)*fs;
    f = df*interval(1,:);

    subplot(8,3,i*3-2)
    plot(f,abs(F1))
    a = sprintf('Section %d',i)
    title(a);

    F2 = fft(e(interval(i,:)))./length(interval(i,:));
    df = (1/T)*fs;
    f = df*interval(1,:);

    subplot(8,3,3*i-1)
    plot(f,abs(F2))
    a = sprintf('Section %d',i)
    title(a);

    ratio = abs(F2./F1);
    dB = 20*log10(ratio);
    subplot(8,3,3*i)
    plot(f,dB,f,zeros(1,length(interval(i,:))));
    a = sprintf('Section %d',i)
    title(a);
    axis([0 500 -70 10]);
end
```

4.5. MATLAB coding using [dsp.LMSFilter] function

```
function result_time_domain=adaptfilt_filtxlms(k)
fs=8000;
t=0:1/fs:k;
T=length(t)-1; %simulation duration
x  = randn(1,T);     % Noise source
% x=cos(2*pi*(1/5)*[0:T-1]);
g  = fir1(24,0.4);        % FIR primary path system model
n  = 0.03*randn(1,T);   % Observation noise signal
d  = filter(g,1,x)+n;     % Signal to be cancelled
b  = fir1(12,0.5);        % FIR secondary path system model
mu = 0.008;               % Filtered-X LMS step size
lms = dsp.FilteredXLMSFilter(16, 'StepSize', mu, 'LeakageFactor', ...
     1, 'SecondaryPathCoefficients', b);
[y,e] = step(lms,x,d);
plot(1:T,d,1:T,e);
title('Active Noise Control of a Random Noise Signal');
legend('Desired','Residual');
xlabel('Time Index'); ylabel('Signal Value');  grid on;
ratio = zeros(1,T-100);
for i = 1:T-100
    ratio(i) = (sum(e(i:(i+99)).^2))/(sum(d(i:(i+99)).^2));
end
dB = 10*log10(ratio);
result_time_domain=dB;
end
```

Script for plotting system performance in MNR using the above function is the same with Appendix 4.2.