Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Monika Charlotte Hansen Gilde

The use of Active Noise Control to reduce wind noise in motorcycle helmets

Master's thesis in Cybernetics and Robotics
Supervisor: Ole Morten Aamo
August 2019

NTNU
Norwegian University of Science and Technology

Monika Charlotte Hansen Gilde

Active Noise Control used to reduce wind noise in motorcycle helmets

Master's thesis in Cybernetics and Robotics
Supervisor: Ole Morten Aamo
August 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

The collaboration with Daal Noise Control Systems AS resulted in the writing of this master thesis. Daal provided the model of the secondary path of the active noise control system, as well as counselling and advice as needed. Two people in particular from Daal have been invaluable, Sigmund Birkeland and Alexander Eide Thorstensen. Ole Morten Aamo, the master thesis supervisor, provided continuous guidance and advice. Without the continual support and ideas given by Daal and Ole Morten, this thesis would never have seen the light of day.

Several people have helped in the process of writing this thesis, either by giving me perspective or by helping me clarify my thoughts. The list includes Bjørn Andre Kristiansen and Stine Aakredalen, two students from NTNU, Jan Morten Skarphol, Anita Skarphol, Maria Elisabeth Sletvold, Berit Hansen Gilde and Ingrid Hansen, my fiancee and his mother, my sister, mother and grandmother, Ismael Aakredalen and Kristin Pettersen, two of my close friends. My colleges helped me with the technical problems, while my family helped with proofreading. Conversations with my friends resulted in brilliant insight, although not necessarily directly connected to the problem in the thesis.

Abstract

Motorcycle riders are exposed to wind noise at highway speeds as loud as 110 dB. Prolonged exposure can lead to hearing loss, and the noise prevents the rider from picking up vital traffic noise. Wind noise can be reduced using active noise control, ANC. In this paper, the history and principle of ANC are explored, as well as different variations on the filtered least means squared algorithm. Some different estimation techniques are also explored, the ANC system is simulated, and the results discussed.

Contents

1 Introduction **1**

2 Wind noise and active noise control **2**
2.1 The properties of wind noise 2
2.2 The history and principle of active noise control 4

3 The filtered least means squared algorithm **8**
3.1 Continuous FxLMS 9
3.2 Block FxLMS 11
3.3 FFT Block FxLMS 12

4 Plant model estimation **17**
4.1 SPR-Lyapunov design 17
4.2 Gradient algorithm based on instantaneous cost 20
4.3 Pure Last-squares algorithm 21

5 Simulation of ANC system **23**
5.1 The ANC system with exact estimation of $\hat{S}(z)$ 23
5.2 The ANC system with flawed estimation of $\hat{\hat{S}}(z)$ 26

6 Discussion **29**

7 Conclusion **31**

8 Appendix **33**
8.1 Mathematics 33
8.1.1 Kalman-Yakubovich-Popov (KYP) Lemma 33
8.1.2 Meyer-Kalman-Yakubovich (MKY) Lemma 33
8.2 Code 33

1 Introduction

For motorcycle riders, wind noise is a common encounter. Prolonged exposure to noise can lead to loss of concentration and result in an accident, affects the persons' ability to hear the environment, including the motorcycle's motor and other road users, and can result in permanent damage to the riders hearing.

The threshold for risking permanent damage to a person's hearing is 85 dB, and higher volume requires less exposure for the damage to occur. At highway speeds, the rider can encounter wind noise as loud as 110 dB, well over the threshold for permanent hearing loss.

The volume of the wind noise is so loud that even when using earplugs or other passive noise control solutions, the rider's hearing is still at risk. Besides, passive noise control, or soundproofing, hinders the riders ability to pick up crucial traffic sounds, including sirens and other road users. Soundproofing also takes up a lot of space, a commodity the helmet already has presses little off, and can come at the expense of safety.

One way to mitigate the problem is by using active noise control, ANC. With ANC, it is possible to filter out the wind noise, and still keep the vital traffic and motor noise. ANC works by sending out noise with the same amplitude and frequency as the noise one wants to reduce but invert the sign on the amplitude. The noise the ANC system sends out cancels out the noise with the same frequency. With perfect ANC the resulting noise the rider hears does not include noise in the frequency range the ANC has been set to operate.

This paper explores the principle behind ANC, reviews several estimation methods for estimating the changing noise pattern, explores different variations on the filtered least means square (FxLMS) algorithm, and simulations of the system with and without measurement noise in the estimations.

2 Wind noise and active noise control

This section explores the properties of wind noise, its frequency in particular, and the theory behind active noise control.

2.1 The properties of wind noise

The volume of sound is described through the amplitude of the sound. The higher the volume, the larger the amplitude.

The wind noise a motorcycle rider experiences has very low frequency, with the highest volume in the frequency range of 10 to 100 Hz. In comparison, the human hearing has a range of 20 to 20 000 Hz but is most sensitive for noise in the range of 1 000 to 4 000 Hz [1]. The threshold for hearing noise is dependent on the frequency of the noise, as shown in figure 1.

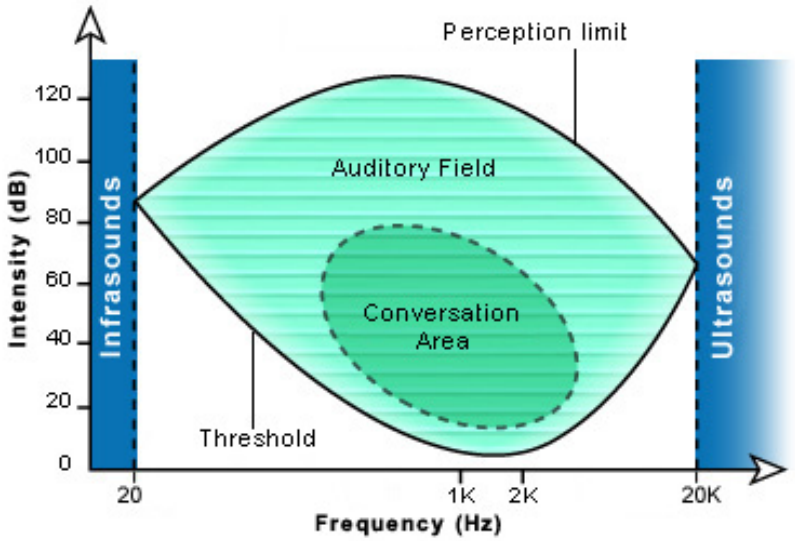


Figure 1: The human auditory field

Depending on the frequency composition of the noise, the best user experience is not necessarily achieved by removing only the harmful high volume noise. Although the most harmful wind noise has a frequency range lower than 100 Hz, one might want to remove noise with frequencies up to 300 Hz, or even higher, to achieve a better user experience.

In the literature, noise with the same properties as wind noise is often called Brownian noise after the botanist Robert Brown, who discovered Brownian motion [2], or red noise, a term from the white noise/white light analogy. Red noise has a higher amplitude at lower frequencies, similar to the red end of the visible spectrum.

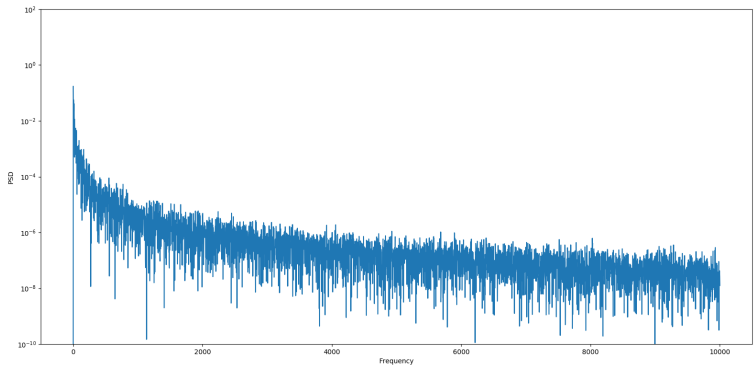


Figure 2: Spectogram of the wind noise

The power spectrum of the wind noise is shown in figure 2. From the figure, it is clear that red noise has more energy at lower frequencies than at higher frequencies. Similarly, figure 3 shows the power spectrum density of red noise. As the frequency increases, the power density decreases, which is consistent with wind noise.

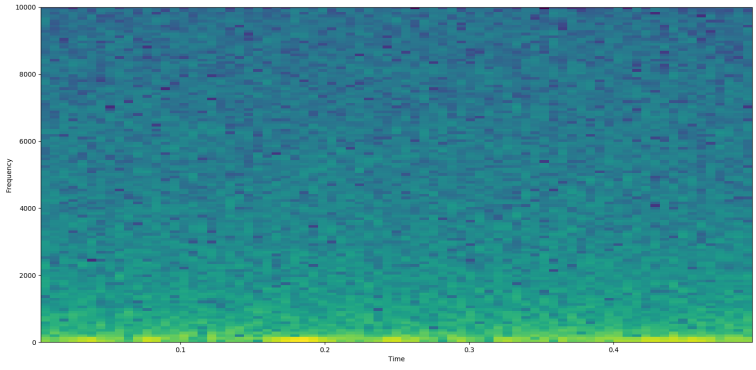


Figure 3: Power spectral density of the wind noise

There are several ways to simulate wind noise, all based on the properties of red noise. Integrating white noise is a popular method, but requires care, so the signal does not grow without bound. Another common method is by filtering pink noise with the same filter that turned white noise into pink noise. A third option is generating red noise with a random walk. With this method, as with the first, one must take care, so the signal does not grow without bound. All of

these methods take advantage of the properties of red noise to generate a noise signal for simulating wind noise.

2.2 The history and principle of active noise control

The field of noise control can be broadly classified into two domains, passive and active. The passive noise control, PNC, domain aims at reducing the noise levels by using sound absorbers or barriers. Even though PNC techniques are effective over a wide range of frequencies, the efficient implementation of these methods at lower frequencies is costly and makes the noise control system bulky [3]. In order to overcome the limitations of PNC techniques at low frequencies, researchers have developed an active method of cancelling noise.

Paul Lueg patented the first active noise control system in 1934, and the patent got approved in 1936 [4]. Figure 4 shows the patent and how Lueg envisioned the ANC working. The patent describes how to cancel noise close to a loudspeaker by phase-advancing the wave and cancelling arbitrary sounds by inverting the polarity.

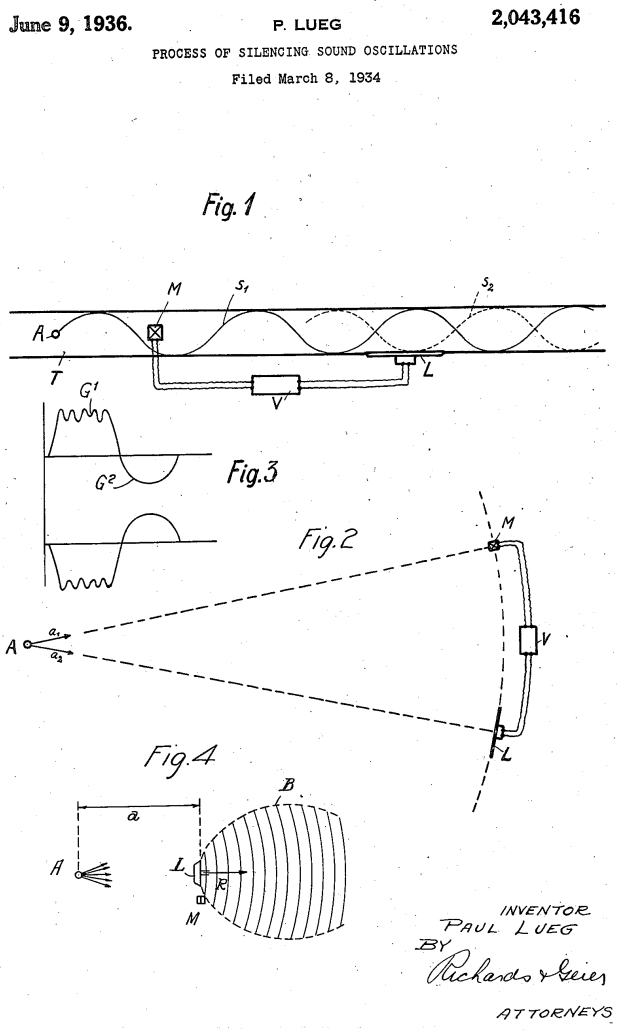


Figure 4: Paul Lueg's patent for a system with active noise control

In the 1950s ANC was developed for use in helicopter and aeroplane cockpits and was quickly adopted by the United States Air Force. In 1957 Willard Meeker developed a working model of active noise control applied to a set of earmuffs that completely covered the outer part of the ear. This headset had an effective attenuation bandwidth of approximately 50 to 500 Hz, with a maximum attenuation of approximately 20 dB [5].

During their around the world flight in 1986, Dick Rutan and Jeana Yeager used a set of prototype headsets built by Bose [6]. Today there is a wide variety in commercially available headsets with ANC built-in, with Bose and several other large companies still working on improving active noise control for commercial use.

Antinoise or counter noise is made by inverting the amplitude of the original noise and adding the two signals together. Figure 5 shows the principle. If the antinoise is a perfect reflection of the original noise, the mean amplitude will be zero, and the result will be silence.

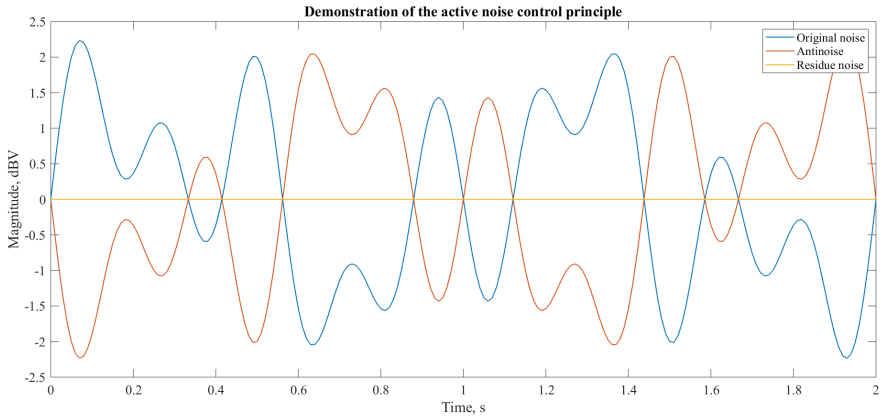


Figure 5: Demonstration of the active noise control principle

For ANC to work, the antinoise must be close to a mirror image of the original noise. If the antinoise is distorted compared to the original noise, for example, with differences in amplitude or with time delay, the ANC system will not work correctly.

The ANC system consists of a loudspeaker, an adaptive controller and one or two microphones, depending on the methods used. If the system uses two microphones, one is placed so that it picks up the original noise, while the other picks up the residue noise, that is the original noise combined with the noise from the loudspeaker. If the system uses only one microphone, it is placed to pick up the residue noise, and the original noise is estimated.

A system with two microphones is called a feedforward ANC system. The reference microphone sense the original noise, and the loudspeaker reproduces the noise, but with the opposite amplitude [7]. The error microphone measures

the level of noise cancellation achieved. Figure 6 illustrates how both the noise x(n) and the residual noise e(n) are used to update the adaptive controller, which drives the active loudspeaker.

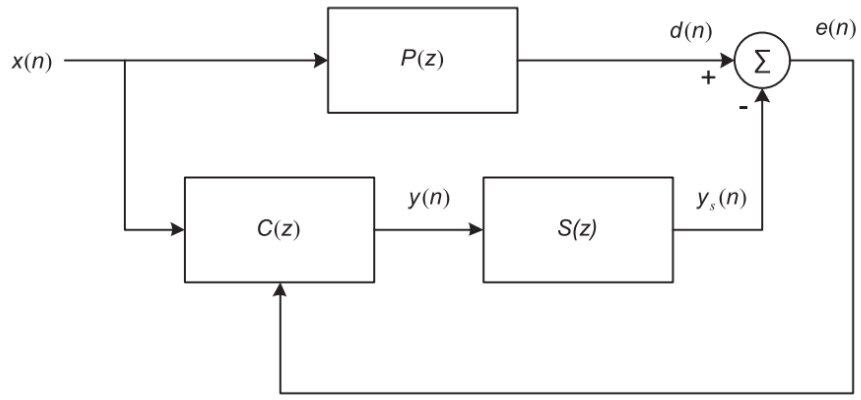


Figure 6: Block diagram feedforward ANC.

A feedback ANC system does not use the a priori information provided by the reference microphone but achieves noise cancellation utilizing an active loudspeaker, an adaptive controller, and an error microphone. After obtaining suitable mathematical models for the electro-acoustic components in a feed-forward ANC system, the same may be represented in a block diagram form, as shown in figure 6. In this figure, C(z) represents the transfer function of the adaptive controller, P(z) denotes the transfer function of the primary path (acoustic path from the reference microphone to the error microphone) and S(z) is the transfer function of the secondary path (the electro-acoustic path from the output of the controller to the output of the error microphone).

The presence of the physical path between the controller and the error sensor leads to instability when adapted using the LMS algorithm. In order to alleviate this problem, a filtered-x least-mean-square (FxLMS) algorithm was developed independently by Burgess, Widrow and Stearns [3]. The FxLMS algorithm uses x(n) filtered through a model of the secondary path $\hat{S}(z)$ as the reference signal for the conventional LMS algorithm. In many applications, the secondary path is time-varying, and online secondary path modelling is a requirement for active control in such scenarios.

The ANC system in focus for this paper is a feedback ANC system with a fixed secondary path.

3 The filtered least means squared algorithm

The most frequently used algorithm in the world of active noise control, are variations on the least mean squared algorithm [8]. Figure 7 shows the basic setup for the LMS algorithm in a feedback ANC system.

The presence of the secondary path S(z), the path from the output of the controller to the input of the error microphone, after the controller will generally cause instability since the presence of the secondary path will cause the error signal and the reference signal to be inappropriately aligned in time [3]. To combat this problem, one uses estimations of the secondary path, $\hat{S}(z)$ to filter x(n) and y(n) as seen in the figure.

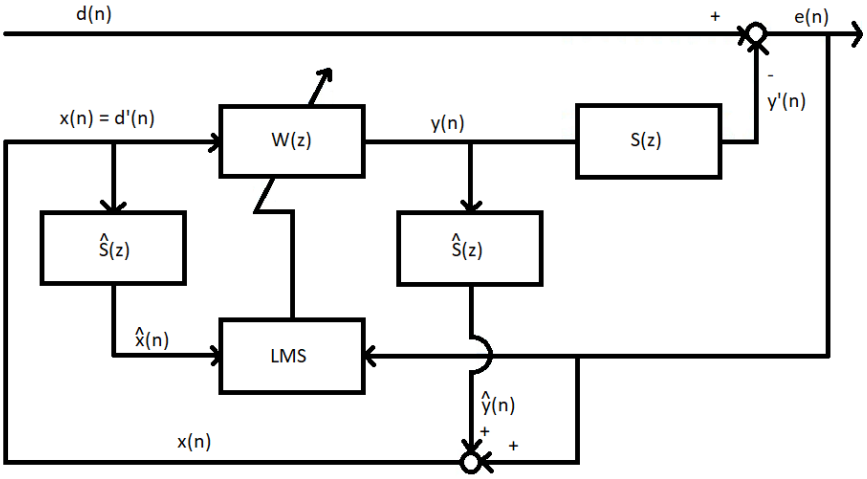


Figure 7: Adaptive feedback ANC system with FxLMS algorithm.

In a feedback ANC system, there is no reference microphone to pick up the original noise, only an error microphone to pick up the resulting noise after the use of ANC.

A consequence of using feedback ANC is that one must estimate the original noise d(n) by x(n). This estimate is feed into the controller that produces the anti-noise y(n). This anti-noise travels through the secondary path, which changes the properties of the anti-noise. This new anti-noise y'(s) is the noise that is subtracted from the original noise, d(n).

The estimation of the secondary path must be sufficiently accurate to the real secondary path. Inaccuracies in the estimation will affect the quality of the ANC system. If the estimation of the secondary path is close to the real secondary path, the ANC system can be very effective. If the estimation is too far off, the ANC system can fail to remove the noise, or even worse, risk amplifying it.

3.1 Continuous FxLMS

Figure 8 shows the basic FxLMS algorithm with an adaptive FIR filter as the adaptive controller W(z).

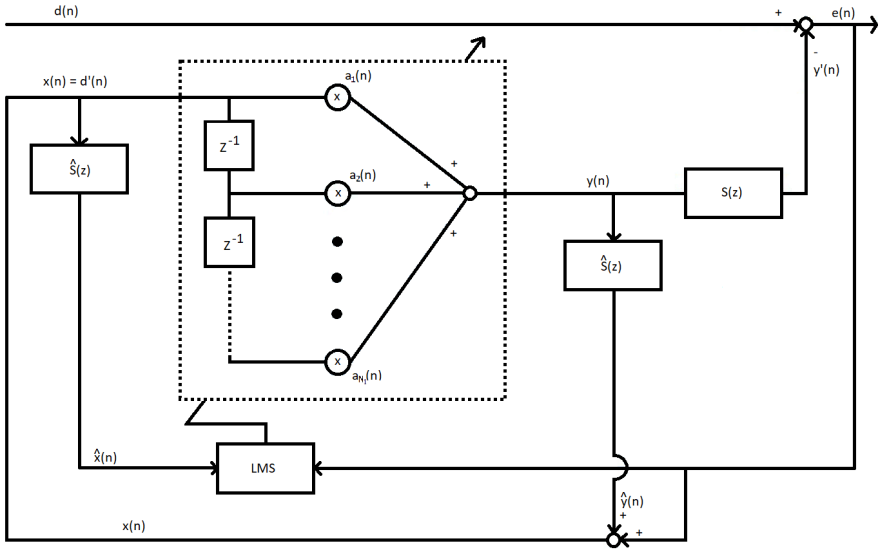


Figure 8: Block diagram of FxLMS ANC system

The output of the adaptive controller at time n is given by

$$y(n) = w(n)^T x(n) \tag{1}$$

where $w(n) = [w_1(n), w_2(n), ..., w_{N_1}(n)]^T$ is the controller weight vector, and $x(n) = [x(n), x(n-1), ..., x(n-N_1+1)]^T$ is the signal vector [7]. The residual noise is given by the equation

$$e(n) = d(n) - y'(n) \tag{2}$$

where d(n) is the original noise and y'(n) is the anti-noise y(n) that has passed through the secondary path S(z). An objective function J(n) based on e(n) is defined as

$$J(n) = [e(n)]^2 \tag{3}$$

The objective function J(n) is minimized using the steepest descent algorithm. This algorithm gives the following equation for updating the filter coefficients in w(n).

$$w(n+1) = w(n) - \frac{\mu}{2}\nabla J(n) \tag{4}$$

where ∇J(n) is the gradient of the objective function with respect to the filter coefficients of the adaptive filter, and μ is the step-size. The gradient of the objective function can be written as

$$\nabla J(n) = 2e(n)\nabla e(n) \tag{5}$$

From figure 8, y'(n) can be written as

$$y'(n) = s(n) * w(n) * x(n) \tag{6}$$

where s(n) is a vector representing the impulse response of the actual secondary path, w(n) is a vector representing the impulse response of the adaptive filter and x(n) is a vector of samples of the estimated reference signal. * is the linear convolution operator. For slowly changing filters, the order of s(n) and w(n) can be changed, and equation 6 can be written as

$$y'(n) = w(n) * s(n) * x(n) \tag{7}$$

From equation 2 we see that the gradient of e(n) is defined as

$$\nabla e(n) = \nabla d(n) - \nabla y'(n) \tag{8}$$

The original noise is independent of the weights of the adaptive filter, so the gradient of the residual noise with respect to the weights of the adaptive filter can be written as the gradient of the secondary noise alone. Rewriting equation 8 with this in mind results in equation 9.

$$\nabla e(n) = -\nabla y'(n) \tag{9}$$

From equation 7 the gradient of y'(n) can be written as

$$\nabla y'(n) = \nabla(w(n) * s(n) * x(n)) \tag{10}$$

which in turn results in writing the gradient of e(n) from equation 9 as

$$\nabla e(n) = -\nabla(w(n) * s(n) * x(n)) \tag{11}$$

Assuming that the model of the secondary path is accurately known, s(n) in equation 11 can be replaced with $\hat{s}(n)$ where $\hat{s}(n)$ represents impulse response of the secondary path model and equation 11 can be written as:

$$\nabla e(n) = -\nabla(w(n) * \hat{s}(n) * x(n)) = -\nabla(w(n) * \hat{x}(n)) = -\hat{x}(n) \tag{12}$$

where $\hat{x}(n)$ is a vector of estimated samples of the filtered reference signal. Combining equations 4 and 5, the equation for updating of adaptive filter weights can be written as

$$w(n+1) = w(n) - \mu e(n)\nabla e(n) \tag{13}$$

With the objective to minimize the mean-square error cost function using a gradient descent approach, the FxLMS algorithm is derived by inserting equation 12 into equation 13:

$$w(n+1) = w(n) + \mu e(n)\hat{x}(n) \tag{14}$$

This equation is the equation for updating the weights of the adaptive filter in the filter least means squared algorithm.

Deriving this equation does depend on replacing the secondary path S(z) with the estimation $\hat{S}$(z), so this method will not work correctly if the secondary path is poorly estimated.

Under the limitation of slow adaptation, the FxLMS algorithm will converge within $\pm 90°$ of the phase error between $\hat{S}$(z) and S(z) [3]. Therefore, the offline modelling of the secondary path using adaptive system identification with the LMS algorithm and white noise as an excitation signal can be used to estimate S(z) during an initial training stage before the operation of noise control for most ANC applications.

3.2 Block FxLMS

Continuous FxLMS algorithm updates the weights of the adaptive filter after each new sample of the residue noise. One way to reduce computing power at the cost of storage is to update the weights of the filter after a few new samples of the residue noise, and this can be done if the weights of the adaptive filter change sufficiently slowly over time.

The equation for updating the weight of the adaptive filter in the Block FxLMS algorithm is based on the equation for updating the weight of the adaptive filter for the continuous FxLMS algorithm from section 3.1, equation 14:

$$w(n+1) = w(n) + \mu e(n)\hat{x}(n)$$

rewriting the equation with L instead of 1 gives the equation

$$\begin{aligned} w(n+L) &= w(n+L-1) + \mu e(n+L-1)\hat{x}(n+L-1) \\ &= w(n+L-2) + \mu e(n+L-2)\hat{x}(n+L-2) \\ &\quad + \mu e(n+L-1)\hat{x}(n+L-1) \\ &= w(n) + \mu \sum_{i=0}^{L-1} e(n+i)\hat{x}(n+1) \end{aligned} \tag{15}$$

Introduce a second time-index k such that n = kL with a fixed integer L, equation 15 can be written as

$$w(kL+L) = w((k+1)L) = w(kL) + \mu \sum_{i=0}^{L-1} e(n+i)\hat{x}(n+i) \tag{16}$$

If the parameters are changed only at the moment kL, the notation can be changed from w(kL) to w(k).

$$w(k+1) = w(k) + \mu \sum_{i=0}^{L-1} e(kL+i)\hat{x}(kL+i) \tag{17}$$

The filter vector is updated every Lth sample.

3.3 FFT Block FxLMS

In this section, a method is proposed to more effectively adapt the value of the convergence coefficient when frequency characteristics of the primary noise are continuously changing. Figure 9 shows a block diagram of the proposed method.

The proposed method is explained, assuming that the primary noise consists of a single frequency component that may vary with time. The proposed method is comprised of two steps. The first step is predicting the frequency of primary noise at some near time, t, in future and the second step is updating the value of convergence coefficient to its optimum value using the information of the frequency of the primary noise predicted in step 1. In the present work, step 1 is achieved by FFT and line-fit method [9].

In the FFT and line-fit method, the reference signal is continuously estimated and saved in blocks of N samples. A fast Fourier transform (FFT) of a block of estimates is carried out to determine the frequency at the mean time of that block. In this way, two or more numbers of such blocks are analyzed to obtain frequency information of the primary noise at different time instants. A linear curve is fitted to the frequency-time data to predict frequency at a future time instant.

Figure 10 shows that at intermediate time instants tm1, tm2 and tm3, the frequencies identified by the FFTs of the blocks of time records are fm1, fm2 and fm3. This time-frequency data is fitted linearly, which allows the prediction of frequency, f, for a continuously varying time, t. A line-fit has been considered because many times, in practice, the variation of frequency of primary noise is linear [9]. However, if the actual variation is not linear, then still the variation can be tracked reasonably well through a sequence of lines as shown in figure 11.

Figure 12 explains the sequential steps of the FFT and line fitting in the proposed algorithm using two blocks. In the beginning, two blocks of the reference signal are obtained, and their FFTs are computed. A line is fitted in frequency-time data, and that line is used to predict frequency in the near future. When a new block is obtained, a new line is fitted using the recent two blocks, which now forms the basis for predicting the frequency. In this way, the processes of computation of FFT and the prediction of the frequency using the line fit go in parallel with the line fit being continuously updated as and when the FFT of a new block of data is available. Due to periodic updating of the line fit, it can closely predict any variation in the frequency of the noise. A line-fit

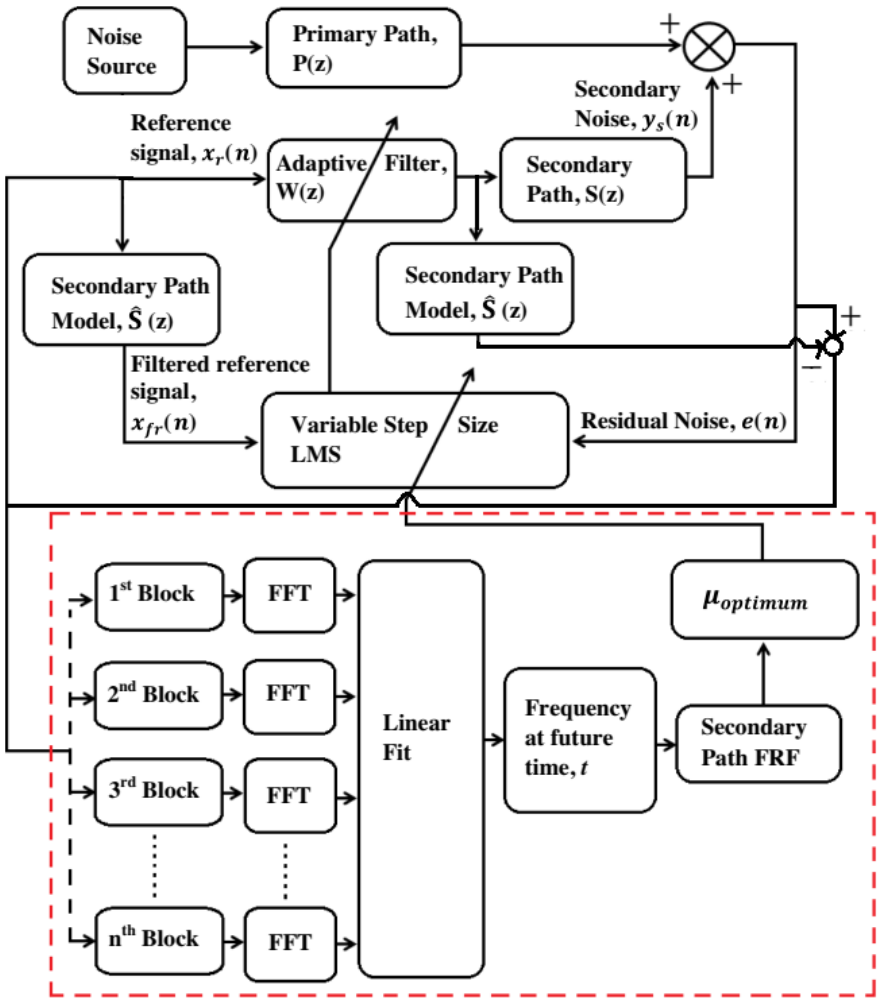


Figure 9: Block diagram of the proposed variable step-size FxLMS algorithm.

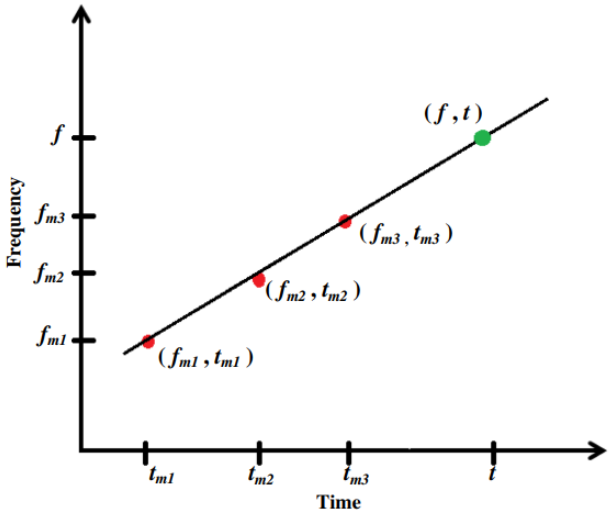


Figure 10: Linear fit in time, frequency data to predict frequency at future time t.

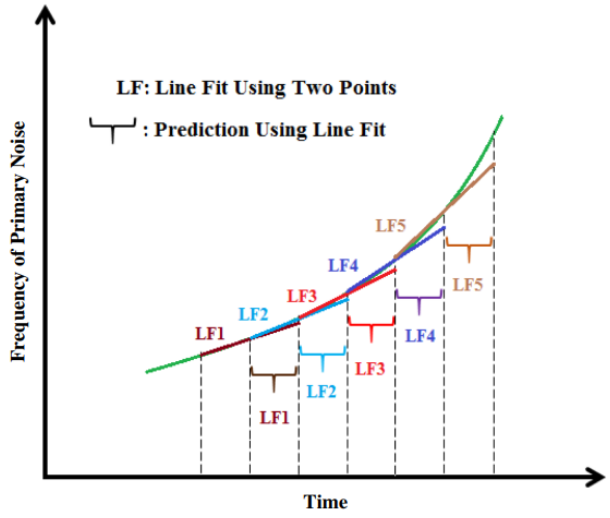


Figure 11: Several line-fits approximating a curve of frequency variation with time

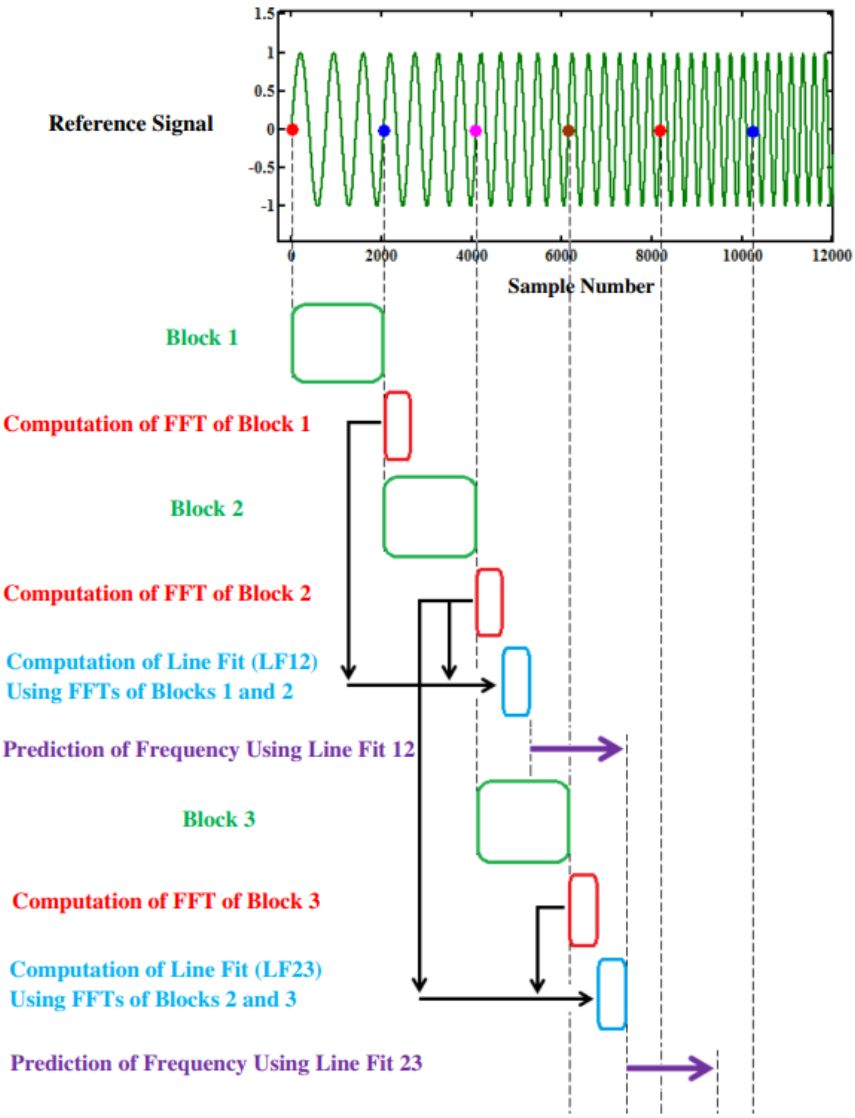


Figure 12: Schematic diagram showing sequential steps of the proposed FFT and line fitting method.

further offers the advantage that it is computationally less expensive and hence, amenable for practical implementation.

4 Plant model estimation

Given the structure of the model, the model response is determined by certain constants referred to as plant or model parameters. In many applications, it is not possible to measure or calculate the parameters using laws of physics or the properties of materials. They have to be deduced by observing the system's response to specific inputs.

If the parameters are fixed for all time, their determination is more straightforward, especially if the system is linear and stable. In such cases, simple frequency or time domain techniques may be used to deduce the unknown parameters by processing the measured response data off-line [10].

If the parameters are unknown and changing over time, one must provide frequent estimates of the parameters of the plant model by suitably processing the plant input-output data online. These estimation schemes are called on-line estimation schemes.

In this system, the wind noise changes over time depending on the speed of the motorcycle, the weather and other road users; therefore, it is vital to use an on-line estimation scheme. The secondary path, on the other hand, does not change significantly over time, and it is enough to estimate the secondary path with an off-line method.

The essential idea behind on-line estimation is the comparison of the observed system response y(t), with the output of a parameterized model $\hat{y}(\theta, t)$, whose structure is the same as that of the plant model. The parameter vector $\theta(t)$ is adjusted continuously so that $\hat{y}(\theta, t)$ approaches y(t) as t increases. Under certain input conditions, $\hat{y}(\theta, t)$ being close to y(t) implies that $\theta(t)$ is close to the unknown parameter vector θ^* of the plant model.

The on-line estimation procedure involves three steps: 1. Select an appropriate parameterization of the plant model. 2. Select the adaptive law for generating or updating $\theta(t)$. 3. Design of the plant input so that the properties of the adaptive law imply that $\theta(t)$ approaches the unknown plant parameter vector θ^* as $t \to \infty$.

In adaptive control, the third step is not always required. In the case of ANC systems, it is not necessary for the estimated plant parameters $\theta(t)$ to approach the unknown plant parameters θ^*, only that the parameterized model $\hat{y}(\theta, t)$ approaches the observed system y(t).

Many different types of plant model estimation exist. Three different types of estimation are explored in this paper, SPR-Lyapunov design, gradient algorithm based on instantaneous cost and Pure Last-Squares algorithm.

4.1 SPR-Lyapunov design

This approach involves the development of a differential equation that relates the estimation or normalized estimation error with the parameter error through an SPR (strictly positive real) transfer function [10]. Once in this form, one can choose an appropriate Lyapunov function V whose time derivative $\dot{V}$ is made non-positive by properly choosing the differential equation of the adaptive law.

These properties of V and $\dot{V}$ are used to establish convergence for the estimation error.

The linear parametric model of the plant is given as z = W(s)$\theta^{*T}\psi$. Since θ^* is a constant vector, we can rewrite the model as

$$z = W(s)L(s)\theta^{*T}\phi \tag{18}$$

where $\phi = L^{-1}(s)\psi$ and L(s) is chosen so that $L^{-1}(s)$ is a proper stable transfer function and W(s)L(s) is a proper SPR transfer function.

Let $\theta(t)$ be the estimate of θ^* at time t. Then the estimate $\hat{z}$ of z at time t is constructed as

$$\hat{z} = W(s)L(s)\theta^{T}\phi \tag{19}$$

The estimation error ϵ_1 is generated as

$$\epsilon_1 = z - \hat{z} \tag{20}$$

and the normalized estimation error as

$$\epsilon = z - \hat{z} - W(s)L(s)\epsilon n_s^2 = \epsilon_1 - W(s)L(s)\epsilon n_s^2 \tag{21}$$

where n_s is the normalizing signal which we design to satisfy

$$\frac{\phi}{m} \in \mathcal{L}_\infty,\ m^2 = 1 + n_s^2$$

Typical choices for n_s are $n_s^2 = \phi^T\phi$ or $n_s^2 = \phi^T P\phi$ for any $P = P^T > 0$. If $\phi \in \mathcal{L}_\infty$, the equation over can be satisfied with m = 1 which gives $n_s = 0$, and $\epsilon = \epsilon_1$.

We examine the properties of ϵ by expressing equation 21 in terms of the parameter error $\tilde{\theta} = \theta - \theta^*$, and obtain

$$\epsilon = WL(-\tilde{\theta}^T\phi - \epsilon n_s^2) \tag{22}$$

For simplicity, let us assume that L(s) is chosen so that WL is strictly proper and consider the following state-space representation of equation 22:

$$\dot{e} = A_c e + B_c(-\tilde{\theta}^T\phi - \epsilon n_s^2),\ \epsilon = C_c^T e \tag{23}$$

where A_c, B_c and C_c are the matrices associated with a state space representation that has a transfer function $W(s)L(s) = C_c^T(sI - A_c)^{-1}B_c$.

The error equation 23 relates ϵ with the parameter error $\tilde{\theta}$ and is used to construct an appropriate Lyapunov type function for designing the adaptive law of θ. Note that the normalized estimation error ϵ and the parameters A_c, B_c and C_c can all be calculated from equation 21 and the knowledge of WL, but the state error e cannot be measured or generated because of the unknown input $\tilde{\theta}^T\phi$.

Let us now consider the following Lyapunov-like function for the differential equation 23:

$$V(\tilde{\theta}, e) = \frac{e^T P_c e}{2} + \frac{\tilde{\theta}^T \Gamma^{-1} \tilde{\theta}}{2} \tag{24}$$

where $\Gamma = \Gamma^T > 0$ is a constant matrix and $P_c = P_c^T > 0$ satisfies the algebraic equation

$$P_c A_c + A_c^T P_c = -qq^T - \nu L_c, \; P_c B_c = C_c \tag{25}$$

for some vector q, matrix $L_c = L_c^T > 0$ and a small constant $\nu > 0$. Equation 25 is guaranteed by the SPR property of $W(s)L(s) = C_c^T(sI - A_c)^{-1}B_c$ and the KYP Lemma or the MKY Lemma depending on whether (A_c, B_c, C_c) is minimal or not. See the appendix for the definitions of the KYP and MKY Lemmas.

The time derivative $\dot{V}$ along with the solution of equation 23 is given by

$$\dot{V}(\tilde{\theta}, e) = -\frac{1}{2} e^T qq^T e - \frac{\nu}{2} e^T L_c e + e^T P_c B_c [-\tilde{\theta}^T \phi - \epsilon n_s^2] + \tilde{\theta} \Gamma^{-1} \dot{\tilde{\theta}} \tag{26}$$

We now need to choose $\dot{\tilde{\theta}} = \dot{\theta}$ as a function of signals that can be measured so that the indefinite terms in $\dot{V}$ cancel each-other out. Because e is not available for measurement, $\dot{\theta}$ cannot depend on e explicitly. Therefore, at first glance, it seems that the indefinite term $-e^T P_c B_c \tilde{\theta}^T \phi = \tilde{\theta}^T \phi e^T P_c B_c$ cannot be cancelled because the choice $\dot{\theta} = \dot{\tilde{\theta}} = \Gamma \phi e^T P_c B_c$ is not acceptalbe due to the presence of the unknown signal e.

Here, however, is where the SPR property of WL becomes handy. We know from equation 25 that $P_c B_c = C_c$ which implies that $e^T P_c B_c = e^T C_c = \epsilon$. Therefore, equation 26 can be written as

$$\dot{V}(\tilde{\theta}, e) = -\frac{1}{2} e^T qq^T e - \frac{\nu}{2} e^T L_c e - \epsilon \tilde{\theta}^T \phi - \epsilon^2 n_s^2 + \tilde{\theta}^T \Gamma^{-1} \dot{\tilde{\theta}} \tag{27}$$

The choice for $\dot{\tilde{\theta}} = \dot{\theta}$ to make $\dot{V} \leq 0$ is now obvious:

$$\dot{\theta} = \dot{\tilde{\theta}} = \Gamma \epsilon \phi \tag{28}$$

which results in the equation

$$\begin{aligned} \dot{V}(\tilde{\theta}, e) &= -\frac{1}{2} e^T qq^T e - \frac{\nu}{2} e^T L_c e - \epsilon \tilde{\theta}^T \phi - \epsilon^2 n_s^2 + \tilde{\theta}^T \Gamma^{-1} \Gamma \epsilon \phi \\ &= -\frac{1}{2} e^T qq^T e - \frac{\nu}{2} e^T L_c e - \epsilon \tilde{\theta}^T \phi - \epsilon^2 n_s^2 + \tilde{\theta}^T \epsilon \phi \\ &= -\frac{1}{2} e^T qq^T e - \frac{\nu}{2} e^T L_c e - \epsilon^2 n_s^2 \leq 0 \end{aligned} \tag{29}$$

Equation 28 is the adaptive law we have been seeking. The equation guarantees that $\theta, \epsilon \in \mathcal{L}_\infty$, and $\epsilon, \epsilon n_s, \dot{\theta} \in \mathcal{L}_2$ [10]. In other words, the equation guarantees that both the parameter vector and the normalized estimation error

are bounded and that the normalized estimation error and the derivative of the parameter vector are integrable. Therefore, the normalized error ϵ will tend toward 0 as time t increases.

4.2 Gradient algorithm based on instantaneous cost

As in section 4.1, the parametric model is given as $z = W(s)\theta^{*T}\psi$. Since θ^* is constant, the parametric model can be written as

$$z = \theta^{*T}\phi \tag{30}$$

where $\phi = W(s)\psi$ [10]. Using the equation above, the estimate $\hat{z}$ of z at time t is generated as

$$\hat{z} = \theta^T\phi \tag{31}$$

where $\theta(t)$ is the estimate of θ^* at time t. The normalized estimation error ϵ is then constructed as

$$\epsilon = \frac{z - \hat{z}}{m^2} = \frac{z - \theta^T\phi}{m^2} \tag{32}$$

where $m^2 = 1 + n_s^2$ and n_s is the normalizing signal designed so that

$$\frac{\phi}{m} \in \mathcal{L}_\infty \tag{33}$$

Typical choices for n_s are the same as those in section 4.1. For analysis purposes we express ϵ as a function of the parameter error $\tilde{\theta} = \theta - \theta^*$ and get

$$\epsilon = -\frac{\tilde{\theta}^T\phi}{m^2} \tag{34}$$

Let us consider the simple quadratic cost function

$$J(\theta) = \frac{\epsilon^2 m^2}{2} = \frac{(z - \theta^T\phi)^2}{2m^2} \tag{35}$$

that we like to minimize with respect to θ. Applying the gradient method, the minimizing trajectory $\theta(t)$ is generated by the differential equation

$$\dot{\theta} = -\Gamma\nabla J(\theta) \tag{36}$$

where $\Gamma = \Gamma^T > 0$ is a scaling matrix that we refer to as the adaptive gain. From the cost function above we have

$$\nabla J(\theta) = -\frac{(z - \theta^T\phi)\phi}{m^2} = -\epsilon\phi \tag{37}$$

and therefore, the adaptive law for generating $\theta(t)$ is given by

$$\dot{\theta} = \Gamma\epsilon\phi \tag{38}$$

We refer to this equation as the gradient algorithm, and this adaptive law guarantees that $\epsilon, \epsilon n_s, \theta, \dot{\theta} \in \mathcal{L}_\infty$ and $\epsilon, \epsilon n_s, \dot{\theta} \in \mathcal{L}_2$.

Again, we see that the normalized error ϵ will tend toward 0 as time t increases.

4.3 Pure Last-squares algorithm

The basic idea behind the least-squares is fitting a mathematical model to a sequence of observed data by minimizing the sum of the squares of the difference between the observed and computed data [10]. In doing so, any noise or inaccuracies in the observed data are expected to have less effect on the accuracy of the mathematical model.

The method is simple to apply and analyze in the linear parametric model

$$z = \theta^{*T}\phi \tag{39}$$

The estimate $\hat{z}$ of z is generated as

$$\hat{z} = \theta^T\phi \tag{40}$$

and the normalized estimation error is generated as

$$\epsilon = \frac{z - \hat{z}}{m^2} = \frac{z - \theta^T\phi}{m^2} \tag{41}$$

where $m^2 = 1 + n_s^2$, $\theta(t)$ is the estimate of θ^* at time t, and m satisfies $\phi/m \in \mathcal{L}_\infty$. As previously, typical choices for n_s are listed in section 4.1.

We consider the following cost function

$$J(\theta) = \frac{1}{2}\int_0^t e^{-\beta(t-\tau)}\frac{[z(\tau) - \theta^T(t)\phi(\tau)]^2}{m^2(\tau)}d\tau + \frac{1}{2}e^{-\beta t}(\theta - \theta_0)^T Q_0(\theta - \theta_0) \tag{42}$$

where $Q_0 = Q_0^T > 0, \beta \leq 0, \theta_0 = \theta(0)$. Because $z/m, \phi/m \in \mathcal{L}_\infty$, $J(\theta)$ is a convex function of θ over $\mathcal{R}^n$ at each time t. Hence, any local minimum is also a global minimum and satisfies

$$\nabla J(\theta(t)) = 0, \ \forall t \leq 0 \tag{43}$$

which results in the equation

$$\nabla J(\theta) = e^{-\beta t}Q_0(\theta(t) - \theta_0) - \int_0^t e^{-\beta(t-\tau)}\frac{z(\tau) - \theta^T(t)\phi(\tau)}{m^2(\tau)}\phi(\tau)d\tau = 0 \tag{44}$$

which yields the so-called nonrecursive least-squares algorithm

$$\theta(t) = P(t)\left[e^{-\beta t}Q_0\theta_0 + \int_0^t e^{-\beta(t-\tau)}\frac{z(\tau)\phi(\tau)}{m^2(\tau)}d\tau\right] \tag{45}$$

where

$$P(t) = \left[e^{-\beta t} Q_0 \int_0^t e^{-\beta(t-\tau)} \frac{\phi(\tau)\phi^T(\tau)}{m^2(\tau)} d\tau \right]^{-1} \tag{46}$$

Because $Q_0 = Q_0^T > 0$ and $\phi\phi^T$ is positive semidefinite, $P(t)$ exists at each time t. Using the identity

$$\frac{d}{dt} P P^{-1} = \dot{P} P^{-1} + P \frac{d}{dt} P^{-1} = 0 \tag{47}$$

we can show that P satisfies the differential equation

$$\dot{P} = \beta P - P \frac{\phi\phi^T}{m^2} P, \;\; P(0) = P_0 = Q_0^{-1} \tag{48}$$

Therefore, the calculation of the inverse in equation 46 is avoided by generating P as the solution of the differential equation 47. Similarly, differentiating $\theta(t)$ with respect to t and using equation 47 and $\epsilon m^2 = z - \theta^T \phi$, we obtain

$$\dot{\theta} = P \epsilon \phi \tag{49}$$

We refer to equations 48 and 49 as the continuous-time recursive least-squares algorithm with forgetting factor.

The stability properties of the least-squares algorithm depend on the value of the forgetting factor β.

Setting $\beta = 0$, i.e without the forgetting factor, equations 48 and 49 becomes

$$\dot{\theta} = P \epsilon \phi; \;\; \dot{P} = -\frac{P \phi \phi^T P}{m^2}, \;\; P(0) = P_0 \tag{50}$$

5 Simulation of ANC system

In this section, the simulation of the ANC system is explored. For ease of reading, the ANC system diagram is shown in figure 13, and is a copy of figure 7 from section 3.

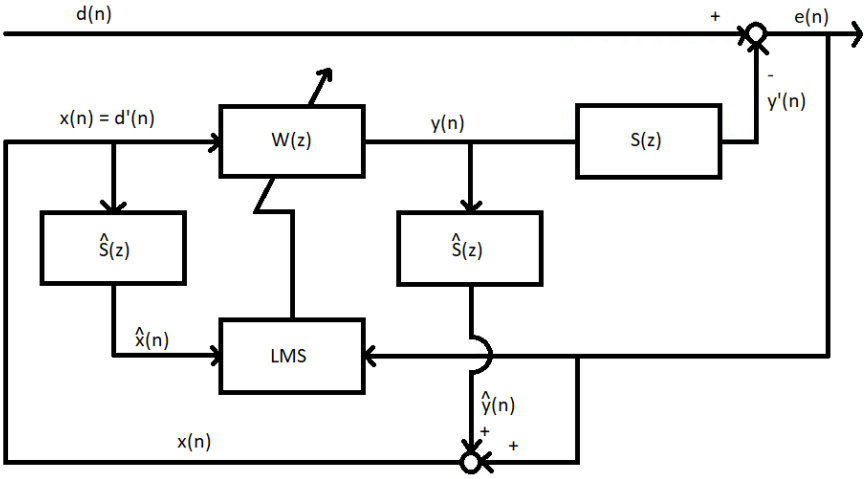


Figure 13: Feedback ANC system, copy of figure 7

The wind noise is simulated using a random walk. One of the simulations is shown in figure 14.

In a real feedback ANC system, the original noise is not available; only the residue noise is. The original noise has to be simulated to simulate the system accurately.

To find out how well the ANC system works, one compares the original noise to the residue noise. The ANC system works as long as the amplitude of the residue noise is lower than the amplitude of the original noise.

Since the secondary path is estimated, one can see how well it is estimated by comparing the original noise $d(n)$ with the estimated noise $x(n)$. The closer $x(n)$ is to $d(n)$, the closer $\hat{S}(z)$ is to $S(z)$, that is, the better the estimation of the secondary path.

5.1 The ANC system with exact estimation of $\hat{S}(z)$

In this section, to test how well the ANC system performs, $\hat{S}(z)$ is equal to $S(z)$. $S(z)$ is represented in figure 15.

Figure 16 consists of two plots. The first plot shows the original noise overlaid with the residue noise. The second plot shows the comparison between the original noise and $x(n)$.

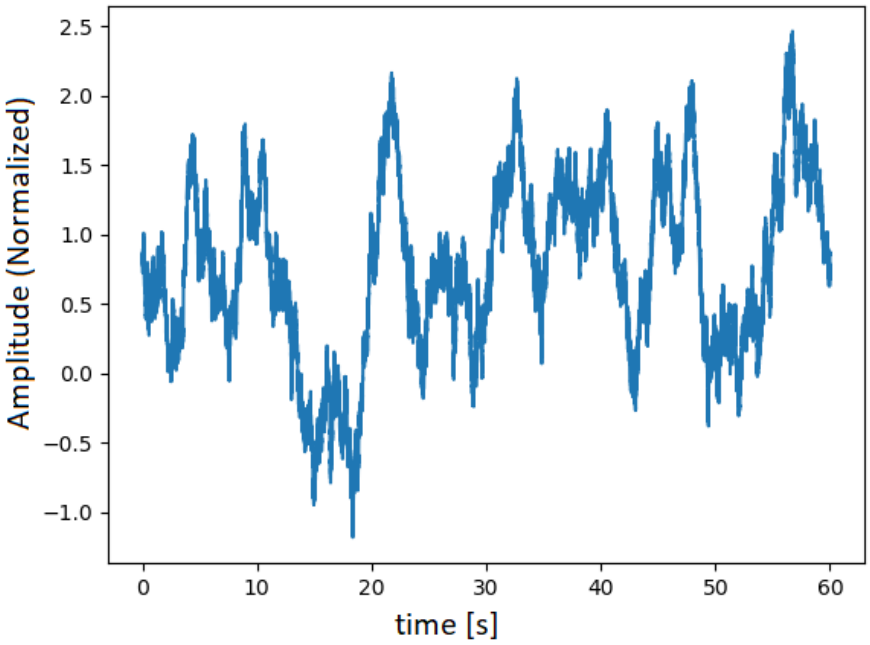


Figure 14: Wind noise simulated using random walk.

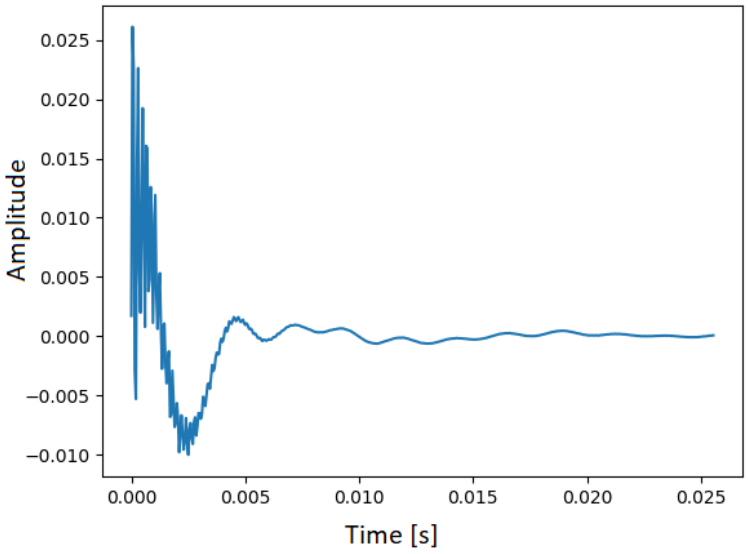


Figure 15: Model of the secondary path.

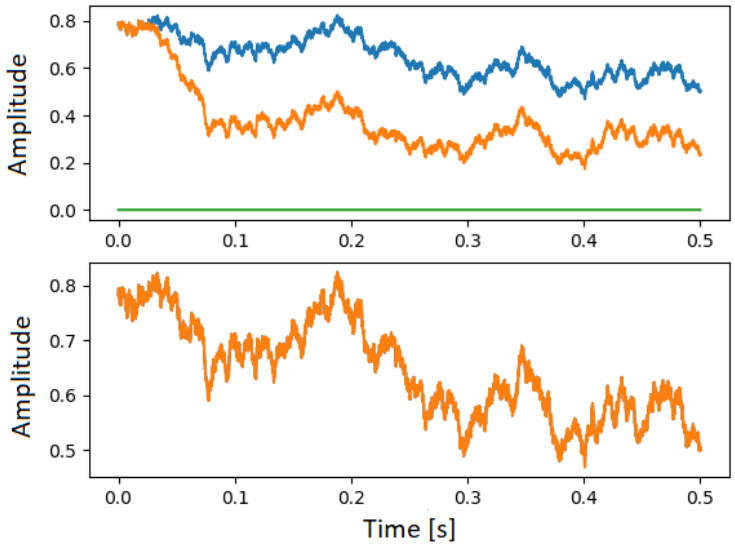


Figure 16: The top plot is the original noise vs the residual noise, while the bottom is the original noise vs the estimation of the noise, both given $\hat{S}(z) = S(z)$

From the first plot in figure 16, it is clear that the residue noise has the same or lower amplitude than the original noise. The second plot shows that the estimation of the original noise follows the real noise perfectly.

5.2 The ANC system with flawed estimation of $\hat{S}(z)$

In this section, the system is simulated with $\hat{S}(z)$. $\hat{S}(z)$ is not equal to $S(z)$, and two different $\hat{S}(z)$ are used, that is, two different estimation of $S(z)$. Figure 17 show $S(z)$ along with the first estimation of $S(z)$, $\hat{S}(z)_1$.

Figure 18 consists of two plots. The first plot shows the original noise overlaid with the residue noise. The second plot shows the comparison between the original noise and $x(n)$.

Again, the residue noise has the same or lower amplitude that the original noise, as shown in the first plot in figure 18. The second plot shows that the estimation of the original noise follows the real noise very well. It is not possible to distinguish the real noise from the estimated noise.

Figure 19 show $S(z)$ along with the the second estimation of $S(z)$, $\hat{S}(z)_2$. Figure 20 consists of two plots. The first plot shows the original noise overlaid with the residue noise. The second plot shows the comparison between the original noise and $x(n)$.

Again, the residue noise has the same or lower amplitude than the original noise, as shown in the first plot in figure 20. The second plot shows that the estimation of the original noise follows the real noise well, as well as with $\hat{S}(z)_1$. It is not possible to distinguish the real noise from the estimated noise.

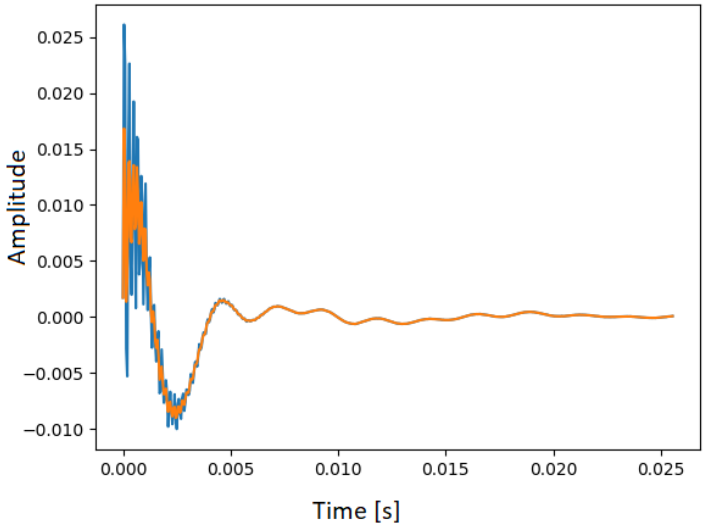


Figure 17: The model of the secondary path $S(z)$ with the estimation of the secondary path $\hat{S}(z)$

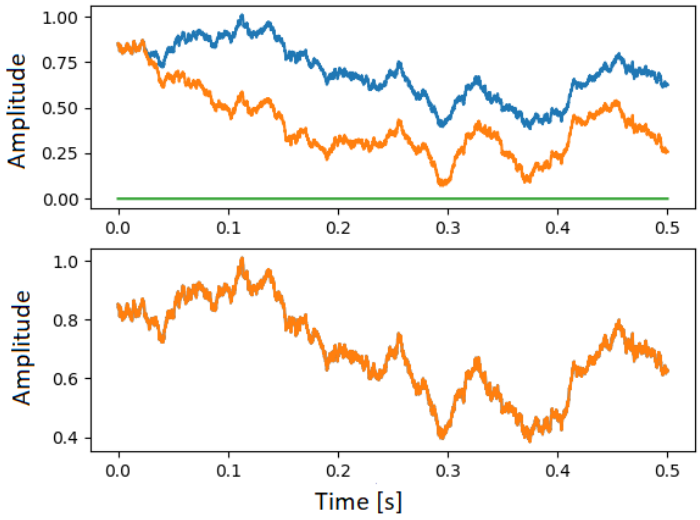


Figure 18: The top plot is the original noise vs the residual noise, while the bottom is the original noise vs the estimation of the noise, both given $\hat{S}(z) \approx S(z)$

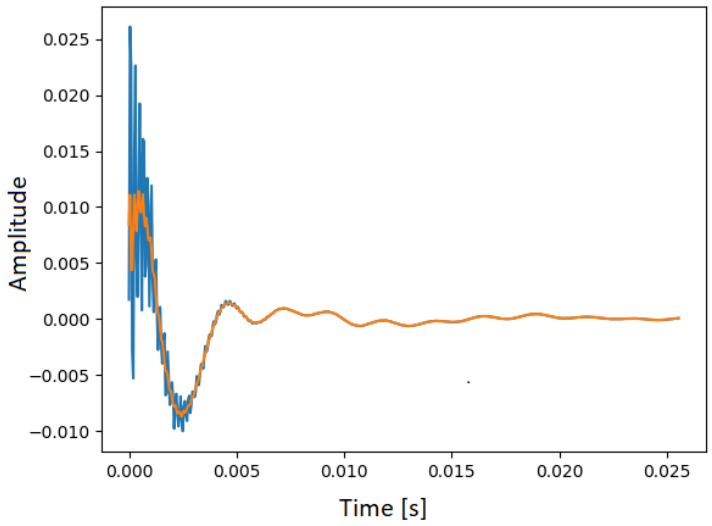


Figure 19: The model of the secondary path S(z) with the estimation of the secondary path $\hat{S}$(z)

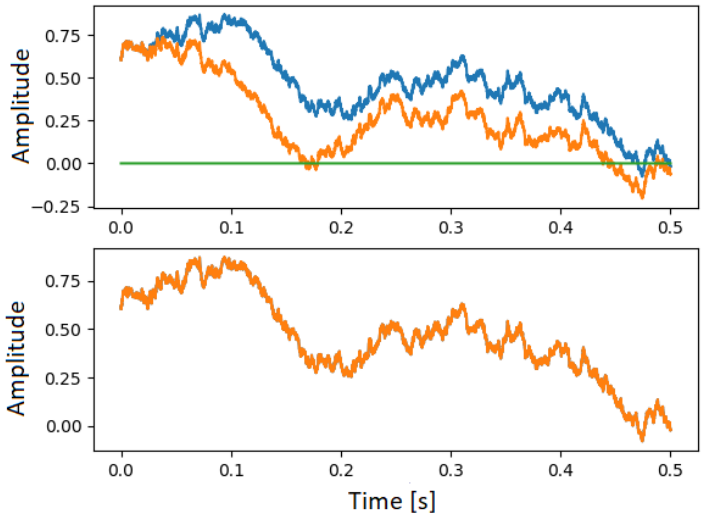


Figure 20: The top plot is the original noise vs the residual noise, while the bottom is the original noise vs the estimation of the noise, both given $\hat{S}(z) \approx S(z)$

6 Discussion

So far, the problem of time delay has not been explored. The reason for this is that the time delay of the ANC system does not necessarily affect the system. The error microphone and the active loudspeaker is not placed directly adjacent to each other. This placement results in the system being causal even if there is time-delay in the system.

As an example, take the case where the distance between the loudspeaker and the error microphone is 1 cm. The speed of sound is 343.2 m/s. With this information, one can calculate the maximum time delay the system can handle:

$$\frac{1\ cm}{343.2\ m/s} = \frac{0.01\ m}{343.2\ m/s} = 2.91 \cdot 10^{-5}\ s = 29.1\ \mu s.$$

The larger the distance between the error microphone and the loudspeaker, the larger the time delay the system can handle, in theory. In practice, sound distortion can occur, and the larger the distance, the more prominent this distortion is.

If the time delay is too large, the ANC system can become unstable, and amplify the noise instead of dampening it. Figure 21 and 22 illustrates the problem with time delay.

The problem of time-delay can be avoided altogether by using a fast controller with a well-written and fast algorithm.

Time delay is not the only way the ANC system can fail. If the estimation of the original noise is flawed, the ANC system runs the risk of amplifying the noise.

Section 5.2 demonstrates that it is possible to estimate the original noise well, even if the estimate of the secondary path is not exact. This result shows that the success of the ANC system is not heavily dependent on the estimation of the secondary path, but instead on the adaptive controller and the method used to regulate the controller.

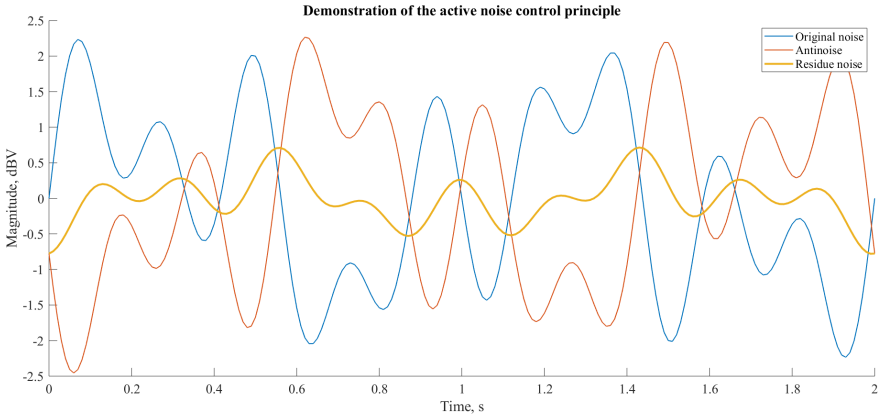


Figure 21: Demonstration of the active noise control principle with time delay.

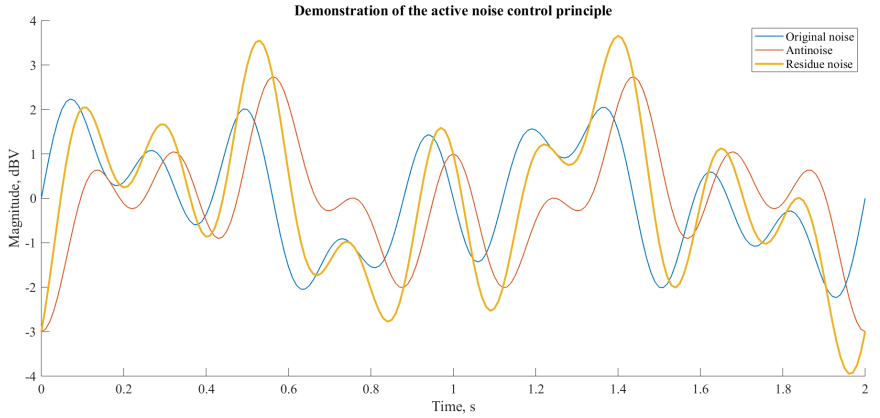


Figure 22: Demonstration of the active noise control principle with severe time delay.

7 Conclusion

The focus of this paper has been the feedback active noise control system with filtered least means squared algorithm. The purpose has been to compare the performance of the ANC system with different estimations of the secondary path.

The original noise was estimated based on the residue noise, and the original noise and residue noise was compared. The weights of the controller were calculated based on the least means squared algorithm and the estimation of the original noise.

It is found that the ANC system works very well despite the flawed estimation of the secondary path. To improve the performance of the ANC system, one can change some of the parameters in the FxLMS algorithm when updating the weights of the controller. One can also use other algorithms. Estimating the secondary path with other methods, including the ones explored in this paper, might also improve the ANC system.

References

[1] R. Pujol, "Human auditory range," 2018.

[2] P. W. van der Pas, "The discovery of brownian motion," *Scientiarum Historia*, 1971.

[3] Y. Kajikawa, W.-S. Gan, and S. M. Kuo, "Recent advances on active noise control: open issues and innovative applications," 2012.

[4] P. Lueg, "Process of silencing sound oscillations," 1934.

[5] "Evaluation of an improved active noise reduction microphone using speech intelligibility and performance-based testing, n.d," 2015.

[6] K. Cunefare, "147th ASA Meeting, New York, NY," 2004. https://web.archive.org/web/20070509115208/http://www.acoustics.org/press/147th/active-noise.htm.

[7] N. V. George and G. Panda, "Advances in active noise control: A survey, with emphasis on recent nonlinear techniques," *Signal Processing*, 2013.

[8] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Prentice-Hall, 1985.

[9] A. Puri and K. Gupta, "A variable step-size filtered-x least mean square algorithm for continuously varying noise," *Noise Control Engineering Journal*, 2016.

[10] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. 2012.

8 Appendix

8.1 Mathematics

8.1.1 Kalman-Yakubovich-Popov (KYP) Lemma

Given a square matrix A with all eigenvalues in the closed left half complex plane, a vector B such that (A,B) is controllable, a vector C and a scalar d $\geq$ 0, the transfer function defined by

$$G(s) = d + C^T(sI - A)^{-1}B$$

is positive real if and only if there exists a symmetric positive definite matrix P and a vector q such that

$$A^T P + PA = -qq^T$$

$$PB - C = \pm(\sqrt{2d})q$$

8.1.2 Meyer-Kalman-Yakubovich (MKY) Lemma

Given a stable matrix A, vectors B,C and a scalar d $\geq$ 0, we have the following: If

$$G(s) = d + C^T(sI - A)^{-1}B$$

is strictly positive real, then for any given L = $L^T > 0$, there exists a scalar $\nu > 0$, a vector q and a P = $P^T > 0$ such that

$$A^T P + PA = -qq^T - \nu L$$

$$PB - C = \pm(\sqrt{2d})q$$

8.2 Code

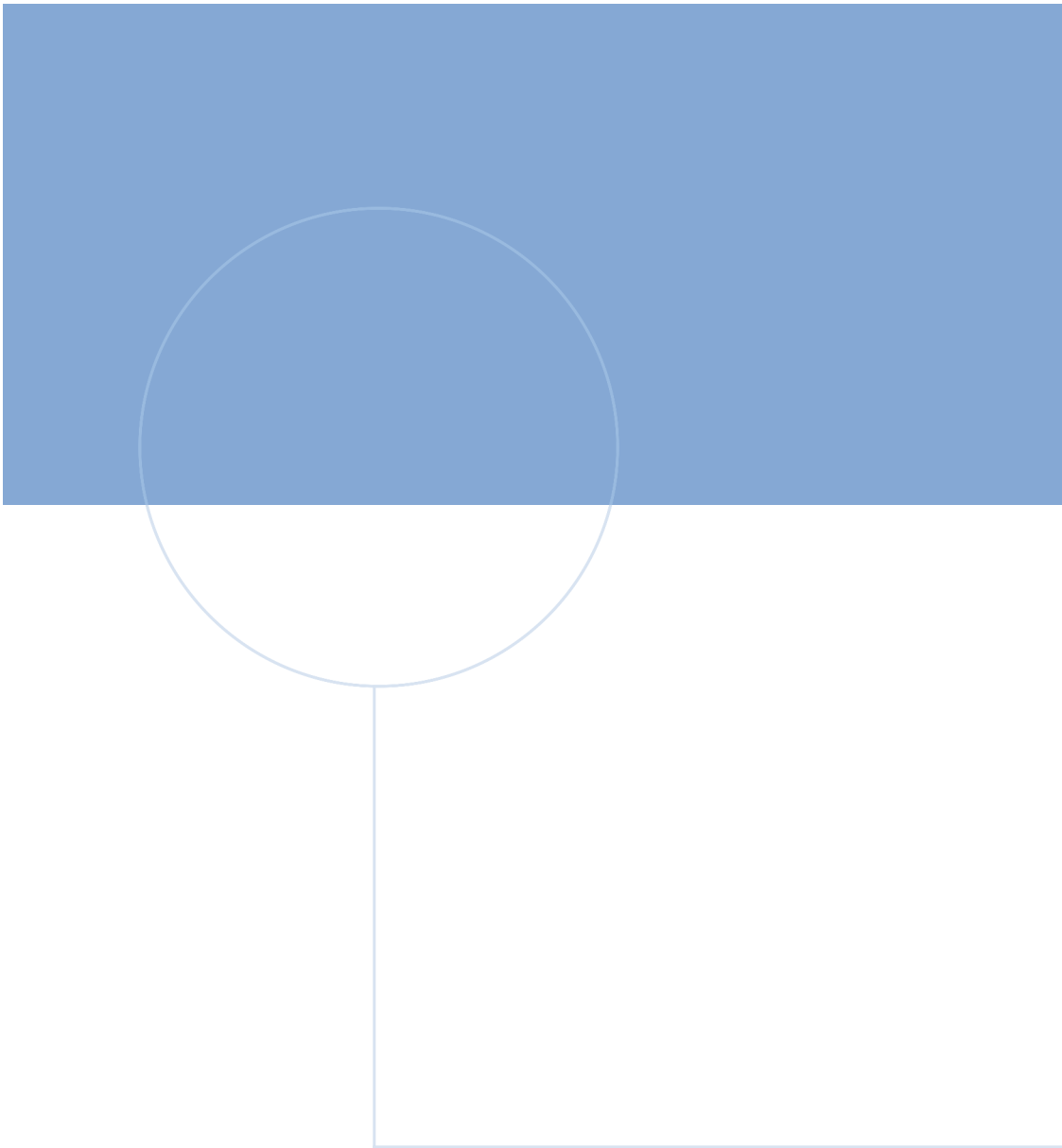
```
from pyhht.visualization import plot_imfs
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy import signal
from pyhht import EMD
import numpy as np
import acoustics
import csv


def makeNoise():
    array = acoustics.generator.brown(1200000)
    return array

def getModel(fileName, array):
```

```
    list = []
    with open(fileName) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        for row in csv_reader:
            list.append(row)
    for i in range(len(list)):
        string = str(list[i])
        l = len(string)
        string = string[2:l-2]
        array.append(float(string))

def plot(signal, frequency, signal2 = None, signal3 = None):
    t = np.arange(0., len(signal)/frequency, 1/frequency)
    plt.plot(t,signal)
    if type(signal2) == np.ndarray:
        plt.plot(t,signal2)
    if type(signal3) == np.ndarray:
        plt.plot(t,signal3)
    plt.show()

def plot2(signal, frequency, signal2 = None, signal3 = None):
    t = np.arange(0., len(signal)/frequency, 1/frequency)
    plt.plot(t,signal)
    if type(signal2) == np.ndarray:
        plt.plot(t,signal2)
    if type(signal3) == np.ndarray:
        plt.plot(t,signal3)

def plotConv(signal, signal2, frequency):
    newSignal = np.convolve(signal,signal2)
    plot(newSignal,frequency)

def initialValues(noise):
    s,s_hat = [],[]
    getModel('h_5.csv',s),getModel('h_5.csv',s_hat)
    #s_hat = smoothing(s_hat)
    #s_hat = smoothing2(s_hat)
    shortNoise = np.array(noise[0:len(s)])
    return np.array(s),np.array(s_hat),np.array(shortNoise)

def smoothing(a):
    for i in range(len(a)-3):
        a[i+1] = (a[i]+a[i+1]+a[i+2])/3
    return a

def smoothing2(a):
    a[0] = (a[0]+a[1]+a[2]+a[3]+a[4])/5
    a[1] = (a[1]+a[2]+a[3]+a[4]+a[5])/5
    a[2] = (a[2]+a[3]+a[4]+a[5]+a[6])/5
    for i in range(len(a)-5):
        a[i+2] = (a[i]+a[i+1]+a[i+2]+a[i+3]+a[i+4])/5
    return a

def inital(s_hat,x):
    return np.convolve(s_hat,x,mode='same')

def updateW(w,mu,e,x_hat):
```

```
    for i in range(len(w)-1):
        w[i] = w[i+1]
    w[-1] = w[-1] - mu*e[-1]*x_hat[-1]
    etta = 1
    if abs(w[-1]) >= etta:
        w[-1] = (etta*w[-1])/abs(w[-1])
    return w

def updateX(x,e,y_hat):
    for i in range(len(x)-1):
        x[i] = x[i+1]
    x[-1] = e[-1] - y_hat[-1]
    return x

def updateE(e,d,y_prime):
    for i in range(len(e)-1):
        e[i] = e[i+1]
    e[-1] = d[-1] + y_prime[-1]
    return e

def conv(result, first, second):
    sum, second, l, r = 0, list(reversed(second)), len(result), 100
    for i in range(l):
        if i != 0:
            result[i-1] = result[i]
    for i in range(r):
        sum = sum + (first[l-i-1]*second[l-i-1])
    result[-1] = sum
    return result

def convY(result, first, second):
    sum, second, l, r = 0, list(reversed(second)), len(result), 1
    for i in range(l):
        if i != 0:
            result[i-1] = result[i]
    for i in range(r):
        sum = sum + (first[l-i-1]*second[l-i-1])
    result[-1] = sum
    return result

def updateNoise(shortNoise,noise):
    first = noise[0]
    noise, shortNoise = np.delete(noise, 0), np.delete(shortNoise,
    0)
    noise, shortNoise = np.append(noise,first),np.append(shortNoise
    ,first)
    return shortNoise,noise

def initiate():
    noise = makeNoise()
    s, s_hat,shortNoise = initialValues(noise)
    y, y_prime, y_hat, w = np.zeros(len(s)), np.zeros(len(s)), np.
    zeros(len(s)), np.zeros(len(s))
    e, x = np.array(shortNoise), np.array(shortNoise)
    x_hat = inital(s_hat,x)
    XN, NN, EN, RN = np.array([]), np.array([]), np.array([]), np.
    array([])
```

```
126     return noise,s,s_hat,shortNoise,y,y_prime,y_hat,w,e,x,x_hat,XN,
        NN,EN,RN
127 
128 def initiateW(w,mu,e,x_hat):
129     for i in range(len(w)):
130         if i == 0:
131             w[i] = -mu*e[i]*x_hat[i]
132         else:
133             w[i] = w[i-1] - mu*e[i-1]*x_hat[i-1]
134     return w
135 
136 def makeSpectogram(sampleFreq,noise):
137     powerSpectrum, freqenciesFound, time, imageAxis = plt.specgram(
        noise, Fs=sampleFreq)
138     plt.xlabel('Time')
139     plt.ylabel('Frequency')
140     plt.show()
141 
142 def makePSD(frequency,noise):
143     f, Pxx_den = signal.periodogram(noise, frequency)
144     plt.semilogy(f, Pxx_den)
145     plt.ylim([1e-10, 1e2])
146     plt.xlabel('Frequency')
147     plt.ylabel('PSD')
148     plt.show()
149 
150 def ANCprinciple(noise,fs):
151     t = np.arange(0., len(noise)/fs, 1/fs)
152     fig = plt.figure()
153     ax = fig.add_subplot(1, 1, 1)
154     ax.plot(t,noise)
155     ax.plot(t,-noise)
156     ax.plot(t,noise-noise)
157     plt.show()
158 
159 def ANCtimedelay(noise,fs,l):
160     a = np.array(-noise)
161     for i in range(l):
162         f = a[0]
163         a = np.delete(a,0)
164         a = np.append(a,f)
165     t = np.arange(0., len(noise)/fs, 1/fs)
166     fig = plt.figure()
167     ax = fig.add_subplot(1, 1, 1)
168     ax.plot(t,noise)
169     ax.plot(t,a)
170     ax.plot(t,noise+a)
171     plt.show()
172 
173 def main():
174     fs,mu = 20000, 0.25
175     noise,s,s_hat,shortNoise,y,y_prime,y_hat,w,e,x,x_hat,XN,NN,EN,
        RN = initiate()
176     makeSpectogram(fs,noise)
177     makePSD(fs,noise)
178     for i in range(10000):
179         if i%10000 == 0:
```

```
            print(i)
        w = updateW(w,mu,e,x_hat)
        y = conv(y,w,x)
        y_hat = conv(y_hat,s_hat,y)
        y_prime = conv(y_prime,s,y)
        shortNoise,noise = updateNoise(shortNoise,noise)
        e = updateE(e,shortNoise,y_prime)
        x = updateX(x,e,y_hat)
        x_hat = conv(x_hat,s_hat,x)
        XN = np.append(XN,x[-1])
        NN = np.append(NN,shortNoise[-1])
        EN = np.append(EN,e[-1])
        RN = np.append(RN,y_prime[-1])
    plot(y_prime,fs,y_hat)
    z = np.zeros(len(XN))
    plt.subplot(2,1,1)
    plot2(NN,fs,EN,z)
    plt.subplot(2,1,2)
    plot2(XN,fs,NN)
    plt.show()

main()
```



NTNU
Norwegian University of
Science and Technology