Noise Cancellation Simulation Framework

1st Nathan Hunsberger
Vanderbilt University
nathan.t.hunsberger@vanderbilt.edu

2nd Berke Lunstad
Vanderbilt University
berke.k.lunstad@vanderbilt.edu

***Abstract*—This paper introduces a Python-based framework for simulating and evaluating active noise cancellation (ANC) techniques. The modular framework supports diverse ANC algorithms, facilitating the comparison of their effectiveness in various scenarios (different sound sources). The framework serves as a valuable tool offering flexibility and practicality in algorithm testing. Through demonstrations of common ANC scenarios, we showcase the framework's utility in evaluation of noise cancellation techniques.**

I. Introduction

Establishing a comprehensive framework for understanding noise cancellation requires delving into the nature of unwanted sound and the methodologies employed in its cancellation. Noise cancellation, as a concept, can be broadly categorized into two distinctive approaches: Passive Noise Cancellation (PNC) and Active Noise Cancellation (ANC). In the subsequent introduction sections, we will give a brief description of the basics of Active Noise Cancellation (ANC) and Passive Noise Cancellation (PNC), diving into their underlying principles, applications, and challenges. By examining these components in greater detail, we aim to provide a groundwork for the framework presented in this paper.

A. Passive Noise Cancellation

Passive Noise Cancellation (PNC), often viewed as a conventional method of noise mitigation, relies on the utilization of physical barriers to dampen observed sound through mechanisms such as shielding or seals. Despite its effectiveness in various applications, it is not without limitations. The primary constraint lies in the necessity for sufficiently bulky materials to achieve optimal noise cancellation. Consequently, in scenarios where mobility or compact size is a paramount requirement, the use of PNC presents notable drawbacks that necessitate the exploration of alternative solutions [8]. The trade-off between achieving optimal noise reduction and maintaining a practical form factor becomes particularly apparent in consumer electronics, such as headphones and earphones, where users seek a delicate balance between portability and performance. This intrinsic challenge has spurred ongoing research and innovation in the field of passive noise cancellation, aiming to overcome these limitations and enhance the adaptability of PNC technologies across diverse applications [11].

B. Active Noise Cancellation

The motivation behind the development of Active Noise Cancellation (ANC) stemmed from the aforementioned constraints of PNC. Instead of resorting to bulky methodologies of blocking noise, ANC uses destructive interference, wherein a noise of identical amplitude but opposite phase is employed. ANC techniques can be broken up into three distinct categories, namely filtered, feedback, and a synthesis of the two.

The filtered category involves the utilization of a reference microphone, a loudspeaker, and an error microphone. Upon detecting incoming noise, the loudspeaker attempts to play a cancelling tone; however, there are issues with convergence stalling in specific instances and the incapacity of the reference microphone to faithfully represent sound as it reaches the loudspeaker. This is due to the controller not being given the error microphone output. However, filtered controllers are useful to us in the sense that we can provide a framework for testing pre-trained controllers on new sound sources. Thus, part of our simulation framework implements this filtered category of ANC.

The feedback category removes the reference microphone, relying instead on the amalgamation of the loudspeaker and error microphone. This approach, aside from being more cost-effective and straightforward to enact, yields outcomes that are more resilient to multiple sources of sound. Nevertheless, issues persist, notably in terms of system stability, as it falls short in effectively suppressing noise across all frequencies.

Finally, the hybrid approach attempts to remedy these issues by relying on input from the reference and error microphones combined to deliver output to the speaker. As a result it has more flexibility and robustness. This produces a methodology that supports filtering the reference microphone while still being able to learn performance information from the error microphone. We based our ANC controller structure on this hybrid approach. [8].

C. Application

Motivation for ANC largely stemmed from its diverse applications such as noise cancellation for vehicles and headphones [8]. However, its applications go beyond this and have the capability to be life saving. Ambulance sirens create sound of up to 120 dB while transporting patients in a critical state. This can result in increased stress and the worsening of their conditions. Introducing ANC helps to mitigate these issues providing support for patients and emergency workers in need [4].

II. ANC Algorithms

In the realm of audio signal processing, the pursuit of enhancing sound quality and mitigating undesirable noise has led to the development of sophisticated ANC algorithms. This section delves into the foundational aspects of ANC algorithms, focusing on three prominent methodologies: Least Mean Squares (LMS), Normalized LMS (NLMS), and Recursive Least Squares (RLS).

The LMS algorithm, an integral component of ANC frameworks, operates by iteratively adjusting filter coefficients to minimize the mean squared error between the estimated and actual signals. Its simplicity and computational efficiency make it a popular choice for real-time applications. Building upon the LMS framework, the NLMS algorithm introduces normalization to adaptively adjust step sizes based on the input signal characteristics, enhancing convergence performance across diverse noise conditions. Furthermore, the Recursive Least Squares (RLS) algorithm, another noteworthy ANC approach, leverages recursive computations to efficiently update filter coefficients, offering a compelling solution for scenarios with time-varying noise characteristics. As we explore the intricacies of these ANC algorithms, a nuanced understanding of their underlying principles and trade-offs will unfold, providing a comprehensive foundation for the subsequent analysis and simulation framework development [8].

All of the algorithms used in simulation for this paper were taken from implementations by the padasip library [2].

A. Least Mean Squares

The Least Mean Square (LMS) algorithm is one of the foundational algorithms used for noise cancellation. The algorithm relies on adjusting its filter coefficients via gradient descent of the square error signal. Details of this algorithm are given by the following equations

$$w(n+1) = w(n) + \mu \cdot x(n) \cdot e(n) \tag{1}$$
$$e(n) = d(n) - y(n) \tag{2}$$
$$y(n) = d(n) - w(n)x(n) \tag{3}$$

Where the variables are defined as below:

$e(n)$	error signal
$d(n)$	desired signal
$x(n)$	reference signal
$w(n)$	filter coefficients

Convergence of LMS depends upon the step size, filter length, and signal power of the input signal and thus cannot be generally guaranteed [4], [10].

B. Normalized Least Mean Squares

The NLMS algorithm is one of the most important algorithms in ANC because it provides an effective speed of convergence while providing steady state response. The change to the algorithm, normalization of $x(n)$, the reference signal, as shown below helps solve the stability issues of the LMS algorithm. Other than the change to the calculation of w, the filter coefficients, everything else in NLMS stays identical to LMS thus variables can be assumed to reference their definitions above.

$$w(n+1) = w(n) + \beta \frac{x(n)}{\epsilon + ||x(n)||^2} e(n) \tag{4}$$

Where β is the new step size and $||x(n)||$ is the L_2 norm which decreases the reaction of LMS when ϵ is some positive real number preventing issues that could occur when the L_2 norm were 0. When β obeys the following inequalities, NLMS converges

$$0 < \beta < 2$$

Therefore NLMS is a powerful algorithm in ANC [10].

C. Recursive Least Squares

The RLS algorithm calculates the least-squares solution in each iteration. While it has a smaller steady state error and faster convergence, it suffers from higher computational complexity in comparison to LMS.

$$k(n) = \frac{\sigma^{-1} Q(n-1) x'_c(n)}{x'^T_c(n) \lambda^{-1} Q(n-1) x'_c(n) + 1} \tag{5}$$
$$y(n) = w^T(n) x(n) \tag{6}$$
$$e(n) = d(n) - y_c(n) \tag{7}$$
$$w(n+1) = w(n) - k(n)e(n) \tag{8}$$
$$Q(n) = \lambda^{-1} Q(n-1) - \lambda^{-1} k(n) x'^T_c(n) Q(n-1) \tag{9}$$

Where the variables are defined as below:

$k(n)$	gain factor
$Q(n)$	inverse of input signal autocorrelation matrix evaluated recursively
$x'_c(n)$	vector of the filtered input signal
$x'^T_c(n)$	transpose of $x'_c(n)$
$y_c(n)$	output signal of the forward path

Where $Q(0) = I\delta^{-1}$ where δ is some small positive number and $w(0) = 0$. [10]

III. Model

A. Primary Assumptions

For the purposes of our framework, a series of assumptions have been made. In this section those assumptions will be detailed along with a brief explanation as to the reasoning behind the assumption.

We assume that the output speaker is not detected in the reference microphone in any filtered category ANC algorithm. This assumption prevents a feedback loop and focuses on the algorithms' abilities to predict and cancel noise without worrying about adverse effects of those attempts.

We assume sound can be modelled in a 1D, linear fashion, as opposed to a 2D or 3D representations. This is due to the ease of modelling and simulation when representing sound in a

1D fashion. Additionally, 2D and 3D representations introduce unwanted consequences to our simulation, like reflection of waves off of boundaries. For this reason, we stuck to 1D sound for this simulation. However, we did spend time looking into discrete representations of sound waves with concepts like cellular automata, though we did not implement them into this project [7].

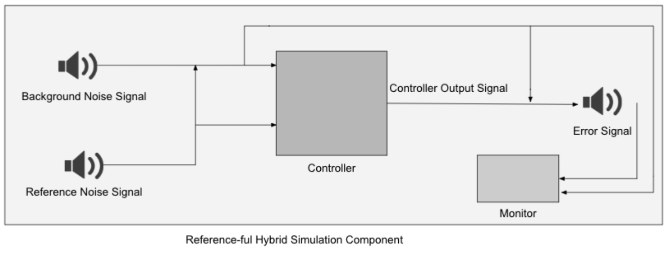
B. Simulation Classes

Our simulation model can be broken down into 3 sub-models: Referenceful, Referenceless, and Filtered. The difference between each simulation class is the setting in which the controller for the simulation will be applied. The reason for this distinction in sub-models is to give users more control over how their controller will be evaluated, as well as explicitly delineate the goal of the controller being tested. Some controllers are meant to block all background noise (Referenceless), whereas some are meant to block all background noise but allow a reference signal through (Referenceful). Additionally, we include a Filtered simulation class for pre-trained controller evaluation. These simulation classes were implemented as Python classes, and were built with extensibility in mind, allowing users to easily interchange which simulation is being ran with minimal work required to match each simulation's interface. The data types passed in each edge is wav signals.

In Figure 1 we outline the component diagram of our Referenceful simulation class. The Referenceful name stems from the simulation being reference-based, as there is a background signal and a reference signal, with the goal being that the background noise shall be cancelled by the controller, but the reference noise shall still be heard. Reference-based noise cancellation is a common use-case of ANC that has been around for decades, so we felt it was important to have this functionality available for testing within our framework [1]. This is the stereotypical ANC case of noise cancelling headphones, in which all external noise can be seen as background noise, and the song to play is the reference noise. As can be seen in our diagram, the system passes two inputs to the controller, one being the background + reference noise signals together, and one just the reference noise. This gives controllers the opportunity to learn what they should be cancelling as they go. The simulation accepts an output signal which is used to create an error signal. This error signal can be thought of as a microphone on the other side of the controller, listening to the sound that is let through the controller. This error signal is passed to a monitor used to check that safety and liveness conditions are met for the controller. The Referenceful simulation makes used of Hybrid controllers, as discussed in Section I.B.

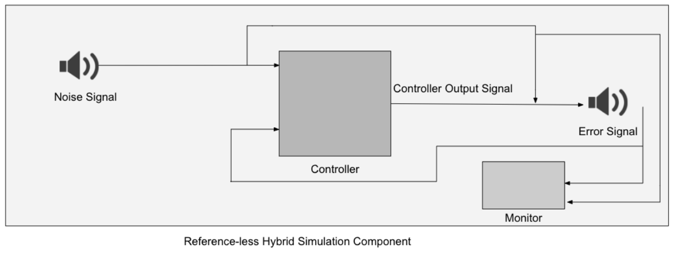
In Figure 2 we outline the component diagram of our Referenceless simulation class. The Referenceless name stems from the lack of a reference signal in the simulation, as there is only a background noise signal being cancelled. The goal

Fig. 1. Referenceful Simulation Component Diagram



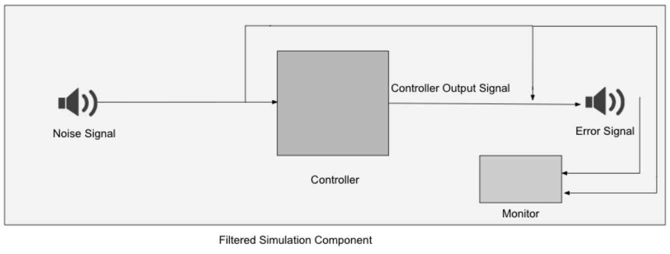
of this simulation is that the background noise is cancelled by the controller, leaving no external noise heard. This is a very common application of ANC, and it is being used today for things like cancelling the sound of an ambulance siren for emergency workers [4]. As can be seen in our diagram, the system passes two inputs to the controller, one being the background noise signal, and one the error signal of the previous timestep. This gives controllers the opportunity to receive feedback on their performance in cancelling the signal in the previous timestep and update their weights accordingly. This allows controllers to implement feed-forward logic, updating filter weights as they make predictions about the signals they are being given. The simulation accepts an output signal which is used to create an error signal. This error signal is passed to a monitor used to check that safety and liveness conditions are met for the controller, alongside being passed back to the controller itself for input in the next timestep. The Referenceless simulation makes used of Hybrid controllers, as discussed in Section I.B.

Fig. 2. Referenceless Simulation Component Diagram



In Figure 3 we outline the component diagram of our Filtered simulation class. The filtered name stems from the ANC controller structure as outlined in Section I.B, as there is no feed-forward logic. The goal of this simulation is similar to that of the Referenceless sub-model: the background noise is cancelled by the controller, leaving no external noise heard. The only difference is that controllers in this simulation do not implement feed-forward logic. Rather, they are expected to be pre-trained for the signals being passed in. A pre-trained controller is one with fixed weights for the task at hand. These controllers do not update their weights as timesteps pass, rather it is expected that their training has suited them well for the distribution of signals being passed in. One present-day application of filtered controllers is controllers that make use of deep learning to train weights [9]. Testing and evaluating these controllers once trained is crucial to their long term success, so we felt a Filtered simulation sub-model would be useful for users. As can be seen in our diagram, the system passes only one input to the controller, the background noise signal. The simulation accepts an output signal which is used to create an error signal. This error signal is passed to a monitor used to check that safety and liveness conditions are met for the controller. Note that the error signal is not sent back as input to the controller in the next timestep.

Fig. 3. Filtered Simulation Component Diagram



C. Safety and Liveness Properties

In order to evaluate controller performance and ensure the validity of each simulation, we have implemented safety and liveness properties. The purpose of these properties is two-fold: both to ensure each controller is evaluated fairly during the simulation as well as monitor controller performance over the course of the simulation. The safety property is uniform for all simulation sub-models, but the liveness property has separate definitions for the Referenceful simulation and the Referenceless/Filtered simulations.

The safety condition is straight-forward and uniform for all sub-models: always (for each timestep) the controller will receive an input signal, and always (for each timestep) the error signal will make use of the controller's output. The purpose of this property is to ensure the simulation is always fair to each controller, as this property makes sure the controller is given input each timestep and effects the output each timestep. If this property were ever violated, a controller would suddenly be at a disadvantage compared to other controllers, as it would not be able to control the output for that timestep. Thus, safety ensures our simulation is a level playing field for all controllers.

The liveness property is used to determine whether or not a controller was successful in cancelling the background noise over the course of a simulation. The idea is that if liveness is satisfied for a controller, the controller is performing well in cancelling background noise. Performance is evaluated differently for Referenceful vs Referenceless/Filtered simulation sub-models, thus liveness is defined separately for each. For Referenceful simulations, liveness is defined as: eventually, averaged over a window size of 1,000 timesteps, the amplitude of the error signal will be at most 10% different than the amplitude of the reference signal for at least one timestep window. This means that the controller will dampen the input background noise signal such that the amplitude is close to that of the reference signal. While this is a good indicator of sound level, ie volume, and that the controller produces sound similar in volume to the reference signal, it is not an all-encompassing test. Volume is only one aspect of sound, as frequency effects what you are hearing as well. We discuss this result further in the Results section, in which we use spectrograms to compare controller performance. However, we still employ this amplitude test as part of our liveness properties as a good controller will in fact be one with volume levels similar to the reference signal.

For Referenceless and Filtered simulations, liveness is defined as: eventually, averaged over a window size of 1,000 timesteps, the amplitude of the error signal will be at most 70% as large as the amplitude of the input noise signal for at least one timestep window. This means that the controller will dampen the input background noise signal such that the amplitude is reduced by a noticeable difference. We use 70% as our benchmark a that was what we found to be the point at which a noticeable reduction in volume occurs. Thus, a controller that satisfies this liveness property will produce a significant decrease in volume compared to the input background noise signal. As we discuss above, amplitude is not the only sound metric, as frequency plays a role in what you are hearing. We assert that, for the Referenceless and Filtered simulations, amplitude alone is enough to gauge whether a controller was successful or not. This is because the objective is not to cancel background noise to a specific reference signal (ie, to a specific sequence of frequencies). Rather, the objective is to simply decrease the volume of the input background noise signal such that the input sound waves are ”cancelled”. By decreasing the amplitude of the output sound waves, the volume will in fact be decreased, no matter the frequency of output. Thus, amplitude is a useful liveness metric when determining successful controllers.

D. Controller Framework

To give users an extensible interface they can use to define their own controllers, we used the ABC library shipped with Python to define an abstract base Controller class. This base class defines an extensible interface that makes up the basic operations our simulations expect from a controller. Users can implement the Controller class in order to test their own controllers inside the simulation framework. There are two functions inside the Controller class (besides the constructor): $input()$ and $feed\_forward()$. $input()$ is an abstract method, thus all controllers must implement it. $input()$ is meant to receive the background noise signal that should be cancelled, and is used by controllers for all simulations. $feed\_forward()$ is not abstract, and the Controller base class actually defines the method as just a $pass$, essentially a no-op. The $feed\_forward()$ method is meant to accept the feed forward logic of the Referenceful and Referenceless simulations, thus controllers designed for those simulations should override the method with their feed forward logic (subsequently, Filtered controllers need not override the method). The $feed\_forward()$ method input will be slightly different depending on the simulation, as the Referenceful simulation will input the reference noise signal, whereas the Referenceless simulation will input the error signal of the previous timestep. The simulation classes will call these functions in order to interface with the user-defined controllers.

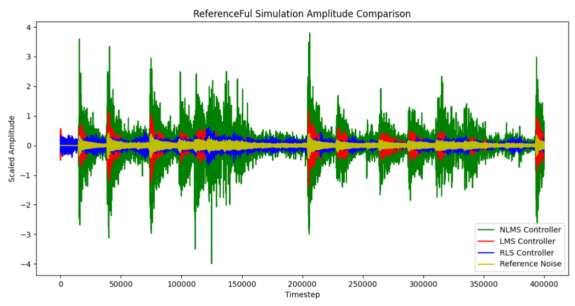
To give users controllers they can compare their own controllers against, we implemented a few popular ANC algorithms (discussed in Section II) using the Padasip library in Python [2]. Note that while these controllers are meant to be used with either the Referenceful or Referenceless simulations, they can be used to gauge performance in the Filtered simulation as well (however, their performance may be degraded, as they were designed as Hybrid controllers). We built upon the Controller base class to define controllers for LMS, NLMS, and RLS algorithms. These controllers are what we base our subsequent experiments and results off of.

IV. Results

To demonstrate an example run of the simulation, we compared the LMS, NLMS, and RLS controllers for a series of sound files and generated example results. While the resulting wav files can be found in our Github (https://github.com/lunstb/cs6376-anc), we present amplitude graphs and spectrograms and argue that are good indicators of controller performance. For the purpose of demonstrating results in this report, we used a sampling frequency of 44.1kHz, and ran each simulation for 400,000 timesteps. The results displayed in this report only use the $coffeeshop.wav$ background noise and $song.wav$ reference noise (found in the $songs/$ directory of our Github) [3] [6] [5]. While we acknowledge that certain controllers perform better for some background noise signals as opposed to others, we assert that these results were representative of controller performance across all background noise and reference noise signals tested.

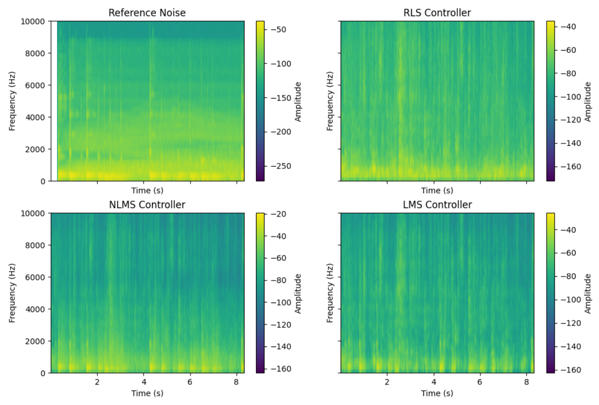
In Figure 4, we see the amplitude comparison chart for a Referenceful simulation. The goal of this simulation is to dampen background noise to a specific reference noise signal. Thus, "good" performance would be an amplitude output that is similar to the reference noise. From the graph, we see that the RLS and LMS controllers perform well in dampening the background noise to an amplitude flow that is similar to the reference noise. However, amplitude is only one aspect of what is heard playing a wav file, as frequency is another aspect that affects sound output. Thus, it is possible to have a controller output an amplitude similar to the reference noise, but not sound like the reference noise. Thus, it is worthwhile to compare both frequency and amplitude. To do so, our simulation generates a spectrogram for the experiment, as seen in Figure 5.

Fig. 4. Referenceful Simulation Amplitude Comparison



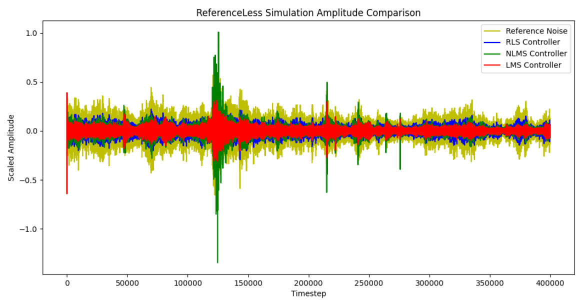
The spectrogram for the Referenceful simulation, as found in Figure 5, displays a comparison of frequency and amplitude over time for each controller, as well as the reference noise. "Good" performance is a controller that creates a spectrogram similar to the reference noise, as this implies the controller is outputting similar frequencies and amplitudes as the reference noise, essentially sounding the same as the reference noise. When examining the chart, it shows that the RLS spectrogram is most similar to the reference noise spectrogram, with amplitudes similar to that of the reference noise alongside frequencies. This result holds when playing back the output wav files for each simulation, as the RLS controller output qualitatively sounds closer to the reference noise with minimal background noise as opposed to the other simulations.

Fig. 5. Referenceful Simulation Spectrogram Comparison



In Figure 6, we see the amplitude comparison chart for a Referenceless simulation. The goal of this simulation is to dampen all background noise. Thus, ”good” performance would be an amplitude output that is closer to 0 scaled amplitude, as this implies a quieter error signal. From the graph, we see that the LMS controller performs best in dampening the background noise to minimal scaled amplitude. As we discuss above, amplitude is only one aspect of what is heard playing a wav file, as frequency impacts sound output. However, the goal of a Referenceless simulation is to dampen all background noise, thus we assert that while frequency impacts sound output, reduced amplitude is the main factor to success for Referenceless simulations. Thus, amplitude reduction is a good predictor of controller success.

Fig. 6. Referenceless Simulation Amplitude Comparison



Note we did not run experiments on the Filtered simulation class, as we have implemented Hybrid controllers, but the results of that simulation will resemble those of the Referenceless simulation.

V. Conclusion

This framework brings the ability to accurately simulate and compare the performance of any number of noise cancellation controllers against each other. This simulation can be done under conditions modeled by sound captured in wav files. Using this framework, researchers can test either untrained or pretrained models under several situations while being able to trust that the benchmarks they receive are accurate to the performance of their algorithms. In addition, the simple nature of the frameworks interface does not require the researcher to understand the inner workings of the simulation to get the results they are looking for.

While this noise cancellation simulation framework provides flexibility and ease of adaptation, there is still room for further work. Specifically in relation to modeling the environment of the noise cancellation. As mentioned earlier, our simulation provides a 1D space for noise cancellation and, while this may be useful in several situations, providing more advanced 3D modeling capabilities including texture and damping information could significantly improve usability.

References

[1] G. R. Barnes and A. A. Ioannides. Reference noise cancellation: Optimizations and caveats. In Cheryl J. Aine, Gerhard Stroink, Charles C. Wood, Yoshio Okada, and Stephen J. Swithenby, editors, *Biomag 96*, pages 7–10. Springer New York, 2000.

[2] Matous Cejnek. Padasip. https://web.archive.org/web/20211014055549/https://matousc89.github.io/padasip/. Accessed: 2023-11-29.

[3] C_Rogers. 370973__waweee__coffee-shop-ambience_remastered.mp3.

[4] Pooja Gupta, Manoj Kumar Sharma, and Sharmelee Thangjam. Ambulance siren noise reduction using noise power scheduling based online secondary path modeling for anc system. In *2015 International Conference on Signal Processing, Computing and Control (ISPCC)*, pages 63–67, 2015.

[5] InspectorJ. Airplane, seat belt beep, a.wav.

[6] InspectorJ. Car alarm, distant, a.wav.

[7] Toshihiko Komatsuzaki, Yoshio Iwata, and Shin Morishita. Modelling of incident sound wave propagation around sound barriers using cellular automata. In Georgios Ch. Sirakoulis and Stefania Bandini, editors, *Cellular Automata*, pages 385–394, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[8] Lu Lu, Kai-Li Yin, Rodrigo C. de Lamare, Zongsheng Zheng, Yi Yu, Xiaomin Yang, and Badong Chen. A survey on active noise control techniques – part i: Linear systems, 2021.

[9] Zhengding Luo, Dongyuan Shi, and Woon-Seng Gan. A hybrid sfanc-fxnlms algorithm for active noise control based on deep learning. *IEEE Signal Processing Letters*, 29:1102–1106, 2022.

[10] Muhammad Moazzam and Muhammad Shoaib Rabbani. Performance evaluation of different active noise control (anc) algorithms for attenuating noise in a duct. 2014.

[11] Vineeth P. Ramachandran and Prabhu Rajagopal. Acoustic superscatterers for passive suppression of cylindrical source radiation in the forward direction, 2021.