Active Noise Control

Signal Processing Final Project

Team-7 Chibson

Adithya Sunil Edakkadan	**2019102005**
Pavani Chowdary	**2019112005**

Overview

Acoustic noise creates a major problem in industrial equipment and automobiles. The passive techniques to control noise have proven to be expensive, take up a lot of space and are ineffective at low frequencies. This brings us to Active Noise Control. Active noise control involves use of electroacoustic or electromechanical systems to cancel unwanted noise based on the principle of superposition. ANC fixes most of the shortcomings of the passive techniques. ANC systems are also cheaper and a lot less bulky. ANC systems must be adaptive in order to cope with variations in the noise. This idea of this project is to examine such ANC systems and analyse their implementations.

Objectives

1. **Learn about adaptive filtering:** Develop a decent understanding of adaptive signal processing and common methods of adaptive filtering.
2. **Examine common adaptive filtering algorithms used for ANC:** Understanding and analyzing the working of LMS and filtered-X LMS algorithms.
3. **Implement some of the algorithms and test their working:** Implement the LMS and filtered-X LMS algorithms from scratch.

Adaptive filtering

The need for adaptive signal processing arises due to the fact that we do not always know the transfer function of all systems. In such cases, we require adaptive filtering techniques in order to get as close to the original transfer function as possible. Adaptive filtering algorithms can be deployed in dynamic and unknown environments to reduce the noise as much as they can. For our use case, which is active noise control, we require adaptive filtering techniques to cancel noise irrespective of the acoustic paths. An adaptive filter requires an additional input d(n), called the desired response, and returns an additional output signal, e(n), the error signal. The FIR coefficients of the filter are updated over time. Here, that adaptation of the filter coefficients depends on minimizing the mean squared error between the filter output y(n) and the desired signal. Some of the most common adaptation algorithms are the Least Mean Square (LMS) and the Recursive Least Square (RLS). While the RLS algorithm offers a higher convergence speed compared to LMS, we are implementing the LMS algorithm in the filter design due to its computational simplicity.

Applications

1. **Automobiles:** Noise attenuation inside vehicles and electronic mufflers for exhaust and induction system
2. **Industrial:** Cancelling noise from almost any industrial appliance
3. **Appliances:** Noise from daily appliances like ACs, exhausts, etc can also be controlled using ANC

Theory

We start by defining some of the concepts of probability theory that are used in understanding adaptive filters and the various algorithms.

- Expectation and variance

  The expected value or mean of a discrete random variable X is a weighted average of the possible values that x can take. It is written as $E(X)$. It can be calculated as follows

$$E(X) = \sum_{i=1}^{\infty} x_i \; p(x_i)$$

  where x_i is one of the possible values random variable X can take and $p(x_i)$ is the probability of X having the value x_i

  Variance is a measure of the spread of the possible values of the random variable. It is represented by σ^2. The variance of a discrete random variable X is calculated as follows

$$\sigma^2(X) = E[(X - E(X))^2]$$
$$= E\,[X^2] - (E[X])^2$$

- Covariance

  Covariance is the measure of joint variability of 2 random variables. For 2 jointly distributed real values random variables X and Y, the covariance is defined as the expected value of the product of deviations from their expected value. For real-valued random variables covariance is defined as follows

  $$Cov_{XY} = E[(X - E[X])(Y - E[Y])]$$
  $$= E[(XY] - E[X]E[Y]]$$

  For complex-valued random variables

  $$Cov_{XY} = E[(X - E[X])(Y - E[Y])^*]$$ (Complex conj. of 2nd factor is taken)

- Correlation

  Correlation is any statistical relationship between 2 random variables. Mathematically, it is defined as the quality of least-squares fitting to the original data. It is obtained by taking the ratio of the covariance of the two variables in the question of our numerical dataset, normalized to the square root of their variances.

  $$Corr_{XY} = \frac{Cov_{XY}}{\sigma_X \sigma_Y}$$

  We can say 2 random variables are uncorrelated if their covariance is zero.

- Stationary process

  We ideally model random processes such as noise as stationary processes. This means that the randomness of this variable is homogeneously distributed throughout the samples. The fluctuations in the values are not localized. This means that the e=mean and variance of the data remain constant throughout time.

  $E[x(n)] = \mu_n = \mu$ First order stationary

  $E[x(n)x^*(n-k)] = function\ of\ k = r(k)$ Second order stationary/Wide sense stationary if both conditions are satisfied

  Here, $r(k)$ can be considered a correlation function known as auto-correlation function.

- Hermitian transpose or conj. transpose

  For an $n \times m$ matrix A the hermitian transpose is defined as

$$(A^H)_{ij} = A^*{}_{ji}$$

Using the above equation we can obtain the following result for autocorrelation function $r(k)$

$$r^*(-k) = E[x^*(n)\, x(n+k)] \;= r(k)$$

- Power spectral density

  Power spectral density is the Fourier transform of the autocorrelation function.

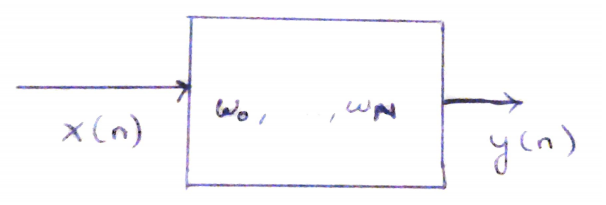
$$S_{xx}(\omega) = \int_{-\infty}^{\infty} R_{XX}(\tau) e^{-jw\tau} d\tau$$

  where S_{XX} is power spectral density and R_{xx} is the autocorrelation function

Now we can start exploring adaptive filters. We begin by looking at a Wiener filter.

Wiener filter

For a filter as shown in the figure the output $y(n)$ is given by

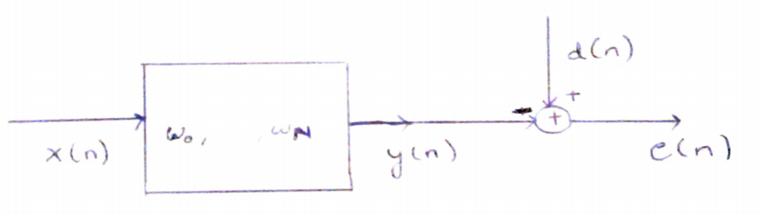


$$y(n) = w_0 x(n) + w_1 x(n-1) + \cdots + w_p x(n-N)$$

$$= w^T x(n)$$

as w and x(n) are both column vectors and to produce required result we need transpose of w

Now we consider as system as shown where we want y(n) to be an estimate of a target signal d(n)



If x(n) is stationary then y(n) is also stationary. x(n) and d(n) will be jointly stationary.

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

Taking autocorrelation we get $R_{XX} = E[x(n)\, x^T(n)] = R$
Taking cross-correlation we get P a column vector such that $P = E[d(n)x(n)]$
We need to minimize $e(n)$ to ensure $y(n)$ is a close estimate of $d(n)$. Taking mean squared error or expected value of $e^2(n)$

$$E[e^2(n)] = E[\, (d(n) - w^T x(n))\, (d(n) - w^T x(n))^T\,]$$ substituting $y(n)$ as $w^T x(n)$

$$E[e^2(n)] = E[d^2(n)] - 2E[w^T x(n)\, d(n)] + E[w^T x(n)\, x^T(n)\, w\,]$$

Using properties of expectation and substituting cross correlation and autocorrelation

$$E[e^2(n)] = E[d^2(n)] - 2w^T P + w^T Rw$$

Taking gradient of this vector we get

$\nabla_w E[e^2(n)] = 0 - 2\nabla_w A + \nabla_w B$ where A and B are $w^T P$ and $w^T Rw$ respectively

Here,

$$A = w^T P = w_0 P(0) \cdots + w_k P(k) + \cdots + w_N P(N)$$

where,

$$\frac{\delta A}{\delta w_k} = P(k)$$

Gradient of B will be the vector containing the partial derivatives of B with respect to each w i.e., w_k for all values of k from 1 to N

$$\therefore \nabla_w A = P$$

$$B = w^T Rw$$

$$B = \sum_{i=0}^{N} w_i (Rw)_i$$

$$= \sum_{i=0}^{N} w_i \sum_{j=0}^{N} R_{ij} w_j$$

$$= \sum_{i=0 \;\; i\neq k}^{N} w_i \sum_{j=0}^{N} R_{ij} w_j + w_k \sum_{j=0}^{N} R_{kj} w_j$$

$$\frac{\delta B}{\delta w_k} = \sum_{i=0 \;\; i\neq k}^{N} R_{ki} w_i + \sum_{j=0}^{N} R_{kj} w_j + R_{kk} w_k$$

$$= 2 \sum_{i=0}^{N} R_{ki} w_i$$

Gradient of B will be the vector containing the partial derivatives of B with respect to each w i.e., w_k for all values of k from 1 to N

$$\therefore \nabla_w B = 2Rw$$

Substituting $\nabla_w A$ and $\nabla_w B$ into the gradient equation we get

$$\nabla_w E[e^2(n)] = -2P + 2Rw$$

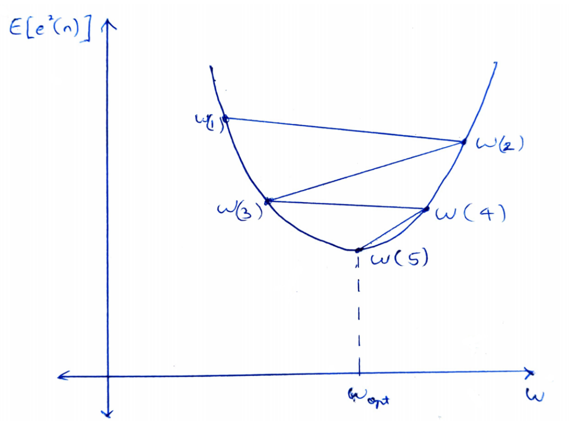
Therefore, for the optimum filtering condition where the error is minimized

$$w = w_{opt} = R^{-1} P$$

where $R = E[x(n)\, x^T(n)]$ and $P = E[x(n)\, d(n)]$

Steepest descent algorithm

We know that for this system $E[e^2(n)]$ is a function is filter coefficients.



We take the $E[e^2(n)]$ plot and find the gradient at the point of filter weights we are currently at. If the gradient at that point is positive then we should subtract a value proportional to the gradient from the current filter weights in order to arrive at the minima. If the gradient is negative then we add a term proportional to the gradient. This algorithm continues to go back and forth about the minima till it finally converges at the minima point. This process is shown in the figure.

Mathematically, steepest descent algorithm can be represented as-

$$\begin{aligned} w(i+1) &= w(i) + \tfrac{\mu}{2} \nabla_w E[e^2(n)]|_{w=w(i)} \\ &= w(i) - \tfrac{\mu}{2}[2Rw - 2P]_{w=w(i)} \\ &= w(i) + \mu[P - Rw(i)] \end{aligned}$$

This expression is for offline computation of filter weights. In order to carry this out in real time, we need P and R also in real time.

$$R = E[x(n)\,x^T(n)]$$

Taking a crude estimate of R,

$$R \approx x(n)x^T(n)$$

Similarly,

$$P = E[x(n)d(n)] \approx x(n)d(n)$$

$$\begin{aligned} w(n+1) &= w(n) + \mu[P - Rw(i)] \\ &= w(n) + \mu[x(n)\,d(n) - x(n)\,x^T(n)\,w(n)] \\ &= w(n) + \mu x(n)[d(n) - x^T(n)\,w(n)] \\ &= w(n) + \mu\, x(n)[d(n) - x^T(n)w(n)] \\ &= w(n) + \mu\, x(n)\,[d(n) - y(n)] \\ w(n+1) &= w(n) + \mu x(n)\,e(n) \end{aligned}$$

This derivation is for real valued $x(n)$. For complex case we can derive the exact same expression for filter weights but we have to consider conjugate transpose instead of the normal transpose taken in R.

Implementation

1. Adaptive filter controlled by LMS algorithm

The implementation of the adaptive filter using the LMS algorithm is quite simple and straightforward. We generate a random noise signal and pass it through the acoustic path. For the purpose of demonstration, we have assumed a transfer function P(n) for this acoustic path. The result of the convolution of the noise signal and acoustic path transfer function is the signal we need to cancel using our adaptive filter. The adaptive filter weights are controlled by the LMS algorithm and we have the algorithm summarised as follows:

$$y(n) = w^T(n)\,x(n)$$

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

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

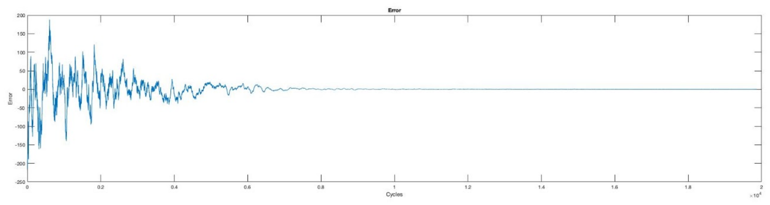
Here, w(n) is the weights vector in the instant n, y(n) and x(n) are the input and output signals respectively. e(n) is the filter's error. w(n+1) is the weights vector at the instant n+1, implying that it is constantly changing.

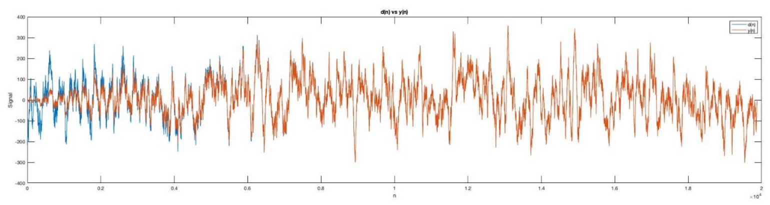
Here, μ is the step size

$$\mu \propto Convergence\ speed$$

Running this algorithm in cycles through our noise signal causes our adaptive filter to converge to P(n) as we have shown in the previous section of this report. When the filter converges we are able to cancel out maximum noise.

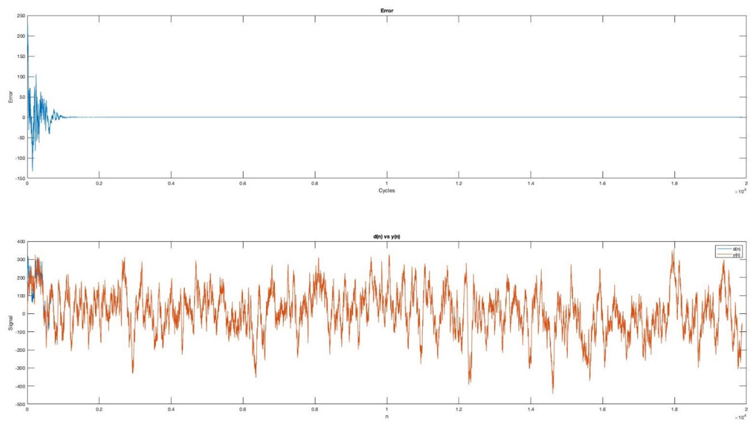
Results





For step size (μ) = 0.01, the error plot is as given above. The error signal fluctuates initially before finally reaching zero after approximately 7500 cycles. The error plot is a direction representation of the convergence of d(n) and y(n) signals. The desired input signal (d(n)) and filter output (y(n)) converge around the same time as the error signal reaches zero

$\mu = 0.05$



From the resulting plots, we can clearly observe the adaptive nature of the filter as the generated signal y(n) slowly converges to the desired signal d(n). We also observe that for a very low value of step size the filter takes a lot longer or a lot more cycles to converge to d(n).

2. Adaptive filter controlled by NLMS algorithm

The implementation of the adaptive filter using the NLMS algorithm is very similar to that of the previously discussed LMS algorithm. The only difference is that we change the learning rate or step size each cycle in order to make the algorithm more resilient to input scaling and guarantee stability. The NLMS algorithm can be summarised as follows:

$$y(n) = w^T(n)\, x(n)$$

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

$$w(n+1) = w(n) + \frac{\mu x(n) e(n)}{x^T(n) x(n)}$$

Results

As a result of normalizing, the LMS algorithm is now stable and less sensitive to input scaling. We normalize the step size with the power of the input signal. By observing the error and convergence plots, we can notice that the relation between the step size and the convergence speed remains unchanged, but the maximum value of u for which the algorithm remains stable increases.

3. Adaptive filter controlled by FXLMS algorithm

The filtered-X LMS algorithm passes the input values through a filter before the LMS algorithm that compensates for the secondary path effect present in the system. This filter $\hat{S}(n)$ is also an adaptive filter that converges to $S(n)$ which is the secondary path transfer function. Hence, we need to first decide the filter weights of $\hat{S}(n)$ using input signal after passing though $S(n)$ as the desired output. Once we have $\hat{S}(n)$ closely mimicking $S(n)$ we can use the filtered-X values in our LMS algorithm for the adaptive filter weights.

Results

FXLMS algorithm compensates for the secondary path effect and the convergence rate is directly proportional to the step size taken.

4. Adaptive filter controlled by FXNLMS algorithm

The filtered-X NLMS algorithm uses NLMS instead of LMS to adapt the filter weights.

Results

As compared to the FXLMS algorithm we notice that the algorithm is a lot more stable in case of FXNLMS due to the normalization of the step size at every cycle making it possible for the filter to converge.

5. Testing on an audio clip

We generated an audio clip of a popular Beethoven piece using the synthesizer made in lab 5 and added noise to it. We passed this noisy signal through our NLMS controlled ANC system. We also tried doing the same with a voice recording.

Results

On passing the noisy signal through the ANC system, we were able to filter out the noise almost entirely and recover the original audio clip. We see that the filter works effectively irrespective of the audio clip as long as we are able to separate the ambient noise from the signal we would like to preserve.

Conclusions

- Active noise cancellation has a significant number of applications in automobiles and industries that use heavy machinery. ANC is also evidently cheaper, takes up less space and is generally more effective than the passive counterparts.
- Adaptive filtering is used to implement active noise cancellation.
- LMS algorithm offers sufficient convergence rates for its low computational complexity.
- Normalized LMS and FXNLMS algorithms are more stable (we are talking about the stability of the algorithm and not the filter) as the step size keeps decreasing on getting closer and closer to the optimum weight and makes it possible for the filter to converge as close as possible to the optimum weights while in the basic algorithms without normalization there is a chance of the step size being too large and skipping over the optimum and diverging.
- These algorithms can be used to cancel noise effectively in most applications as verified by our experiment using audio clips.

References

1. Active Noise Control: A Tutorial Review by Sen M Kuo and Dennis R Morgan

   https://ieeexplore.ieee.org/document/763310

2. Adaptive Filtering based on LMS Algorithm by Sireesha N, K Chithra and Tata Sudhakar

   https://ieeexplore.ieee.org/document/6701910

3. Adaptive Filter Theory (5th edition) by Simon Haykin

4. On the convergence of real-time active noise control systems

   https://www.researchgate.net/publication/220226701_On_the_convergence_of_real-time_active_noise_control_systems