Active Noise Cancellation

A project report submitted by

Jayant Sorte (B20CS022)

Keshav Mundra (B20CS026)

Neehal Bajaj (B20AI026)

in partial fulfilment of the requirement for the award of the degree of

B.Tech

Indian Institute of Technology Jodhpur

Department of Computer Science Engineering

July 22

Declaration

I hereby declare that the work presented in this Project Report titled 'Active Noise Control using Machine Learning technique' submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of B. Tech is a Bonafide record of the work carried out under the supervision of Amrita Puri**.** The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Jayant Sorte (B20CS022)

Keshav Mundra (B20CS026)

Neehal Bajaj (B20AI026)

Certificate

This is to certify that the Project Report titled ‘Active Noise Control using Machine Learning techniques’, submitted by Jayant Sorte (B20CS022), Keshav Mundra (B20CS026) and Neehal Bajaj (B20AI026) to the Indian Institute of Technology Jodhpur for the award of the degree of **B. Tech**, is a Bonafide record of the work done by him/her under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Amrita Puri

TABLE OF CONTENTS

Abstract -

In many practical applications the acoustic noise generated from dynamical systems is nonlinear and deterministic or stochastic, coloured, and non-Gaussian. It has been reported that the linear techniques used to control such noise exhibit degradation in performance. In this paper, a comparative study of various algorithms such as LMS, FxLMS and KLMS. The weights of the filter are updated using gradient descent technique. These algorithms are tested upon noises produced by random signals. The comparison of the algorithms is based in terms of mean squared errors produced when tested upon different type of noises. Finally, the thesis explores performance of these algorithms in terms of convergence time, steady state noise, stability and computational complexity to reduce different noises.

INTRODUCTION -

What is ANC?
Active Noise Cancellation lessens ambient noise using speakers and microphones. The most popular kind is typically found in over-ear headphones. . It is basically a method for reducing unwanted sound by the addition of a second sound specifically designed to cancel the first, unlike passive noise cancellation which just physically seals out unwanted noise. This can be achieved by using different algorithms as further explained in the report.

PROBLEM DEFINITION AND OBJECTIVE –

Problem:

Our problem was to solve the problem of active noise cancellation in the real world. This can be achieved by taking help from non linear filters. In layman's terms, we have to learn to apply these filters to continuous inputs.

Apparently, creating a nonlinear adaptive filter is a considerably more challenging task. Cascading a static nonlinearity with a linear filter is one straightforward method of implementing a nonlinear adaptive filter. Though, in this strategy, the nonlinearity is highly selected despite the poor modeling capabilities. During training, there exist local minima that are problem-dependent.

Our goal was to create
a filter that can convert the input into a high-dimensional space so that we can give continuous input to the filter and get a good result.

METHODOLOGY AND PROPOSED ALGORITHMS -

Our methodology to know better about this topic was as follows:

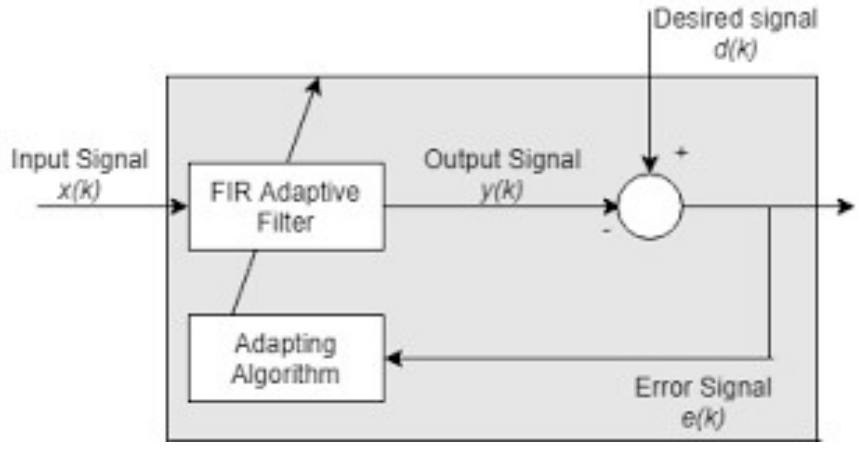
1. We started learning about what is ANC and why is it used.
2. The first algorithms we encountered were LMS and FxLMS.
3. We understood their implementation and basic difference between them.
4. We learned about the basic filters required in order to proceed further.
5. We got to know more about the kernel functions and the learning step size and how can we implement one in our filter.
6. We created a basic nonlinear adaptive filter using the kernel trick.

LMS -

The LMS algorithm estimates the results continuously updating the filter weights in a manner to converge the optimum filter weight. It is a computationally simple algorithm to implement. The algorithm begins with assumption of small weights and at each step, the weights are updated using the mean square error.

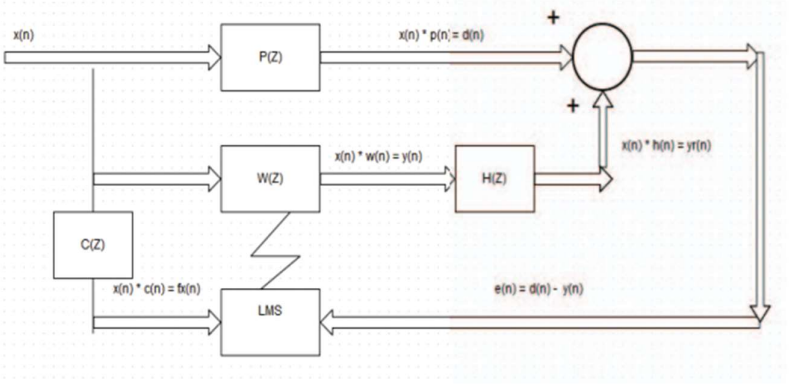
This is simple algorithm stated as follows

- Initialization
- w(0) = 0, choose η
- Computation
- while {u(i), d(i)} available do
- e(i) = d(i) − wT(i − 1)u(i)
- w(i) = w (i − 1) + η e (i)u(i)
- end while



FxLMS –

The FxLMS algorithm is computationally simple where secondary path effects are also includes, but its convergence speed is slow.FxLMS algorithm is that it is computationally simple like the most commonly used Least Mean Square (LMS) algorithm. In addition it includes secondary path effects. The secondary path estimation should be more precise and accurate in order to get better and more effective results..



KLMS -

The main limitation of a Linear adaptive filter is its limited computational power, in general, complex real-world applications require more expressive hypothesis spaces than linear functions.

The main draw of building filters is to take use of the space in the linear structure to execute well-known linear adaptive algorithms and produce nonlinear filters in the input space. This alternative design strategy outperforms neural networks in terms of universal approximation capabilities, convex optimization (i.e., no local minima), and yet manageable computing complexity. Because it bridges the two crucial fields of adaptive filtering and neural networks, it occupies a special place.

Out of a research paper we got to know about basic structure of the filter:

- Initialization
- choose step - size parameter η and kernel κ
- a 1 (1) = η d (1), C (1) = {u (1)}, f 1 = a 1 (1) κ (u (1),·)
- Computation
- while {u(i), d(i)} available do
- %compute the output
- $fi{-}1(u(i)) = \sum aj\ (i-1)k(u(i), u(j))\ i{-}1\ j{=}1$
- %compute the error
- e (i) = d (i) – fi-1 (u (i))
- %store the new centre
- C (i) = {C (i − 1), u (i)}
- %compute and store the coefficient

- ai (i) = η e (i)
- end while

we can define Gaussian Kernel as

$$k(u, u') = exp(-a\|u - u'\|\ 2\ )$$

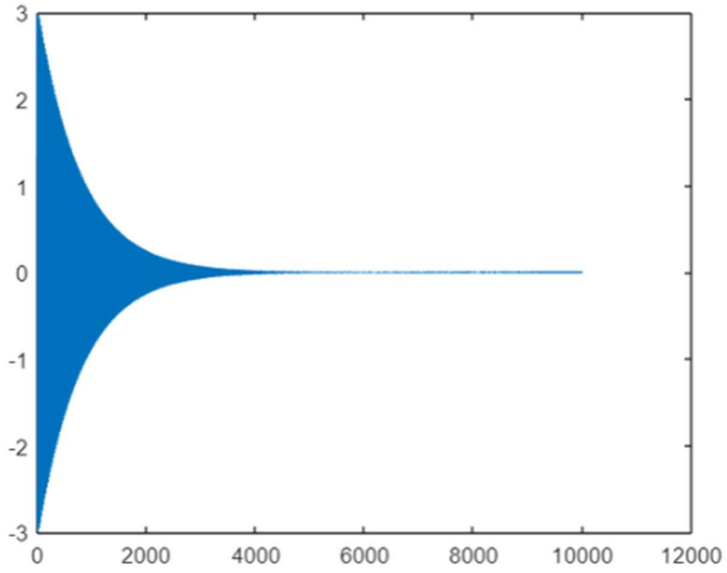
NUMERICAL STUDY AND RESULTS –

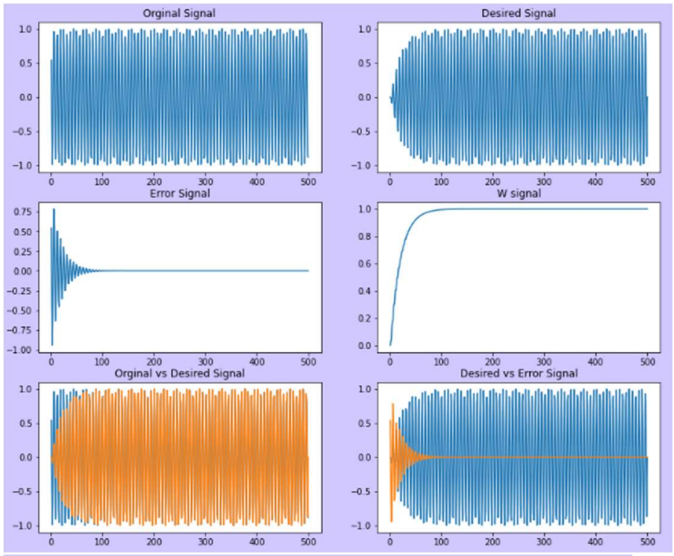
LMS -

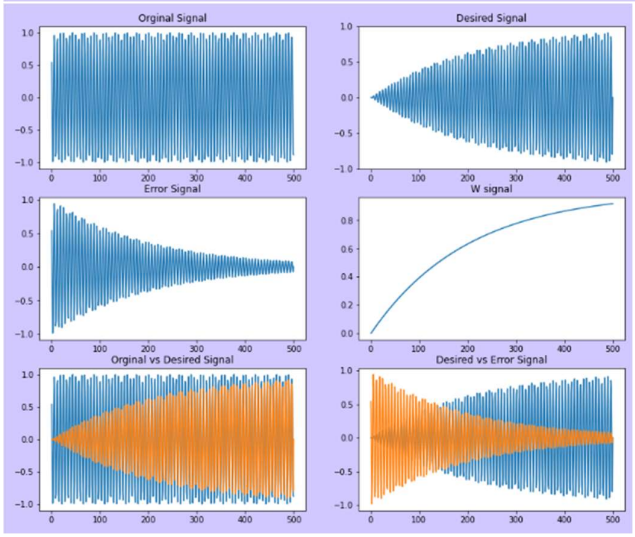
The steepest descent adaptive filter has a weight-vector update equation given by -
W(n+1) = W(n) + uE{e(n)X*(n)}

Dt = 1/sampling_freq

$$y(n) = \sum_{k=0}^{M-1} w_k(n)u(n-k)$$




Orginal Signal
Desired Signal
Error Signal
W signal
Orginal vs Desired Signal
Desired vs Error Signal


Orginal Signal
Desired Signal
Error Signal
W signal
Orginal vs Desired Signal
Desired vs Error Signal

FxLMS –

Then, estimate $\nabla J(n)$ as follows,

$$\nabla J(n) = \nabla e^2(n)$$

$$\nabla J(n) = 2e(n)\nabla e(n) \quad (2)$$

Now to estimate $\nabla e(n)$, the derivation is as follows based on the block diagram,

$$e(n) = d(n) + s(n) * y(n)$$

Where s(n) is the secondary path impulse response.

$$\nabla e(n) = s(n) * \nabla y(n) \quad (3)$$

Now to estimate $\nabla y(n)$, the derivation is as follows based on the block diagram,

$$y(n) = W^T x(n)$$

Where W is the controller weight vector and x is the reference signal tap vector (of the same length as the controller length)

Now $\nabla y(n)$ can be expressed by,

$$\nabla y(n) = \frac{\delta y(n)}{\delta W}$$

$$\nabla y(n) = x(n) \quad (4)$$

Substitute (4) into (3),

$$\nabla e(n) = s(n) * x(n) \quad (5)$$

Substitute (5) into (2),

$$\nabla J(n) = 2e(n).s(n) * x(n) \quad (6)$$

Substitute (6) into (1),

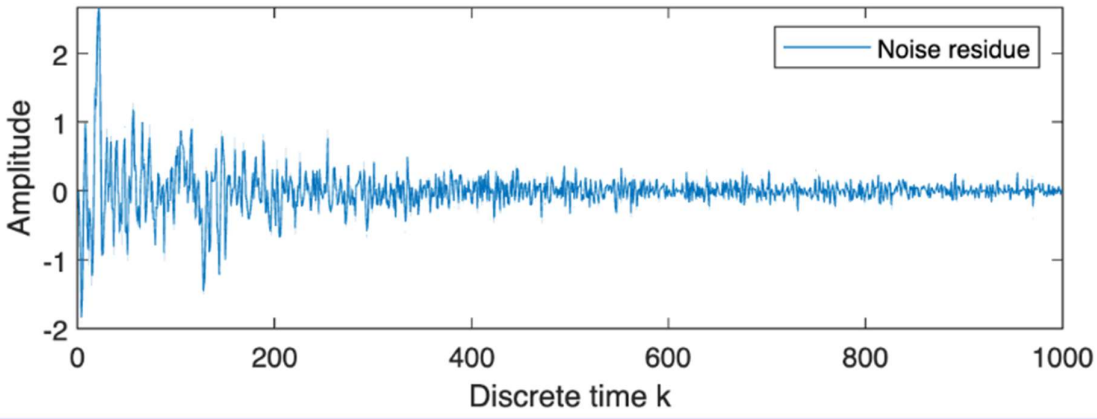
$$W_{New} = W_{Old} + 2\mu e(n).s(n) * x(n) \quad (7)$$

The reference signal is filtered by $\hat{s}(n)$ before passing through the standard LMS algorithm. Therefore resulting the compensation for secondary path. s(n) should be estimated through off-line or online secondary path techniques. If $\hat{s}(n)$ denotes an estimate of s(n), then

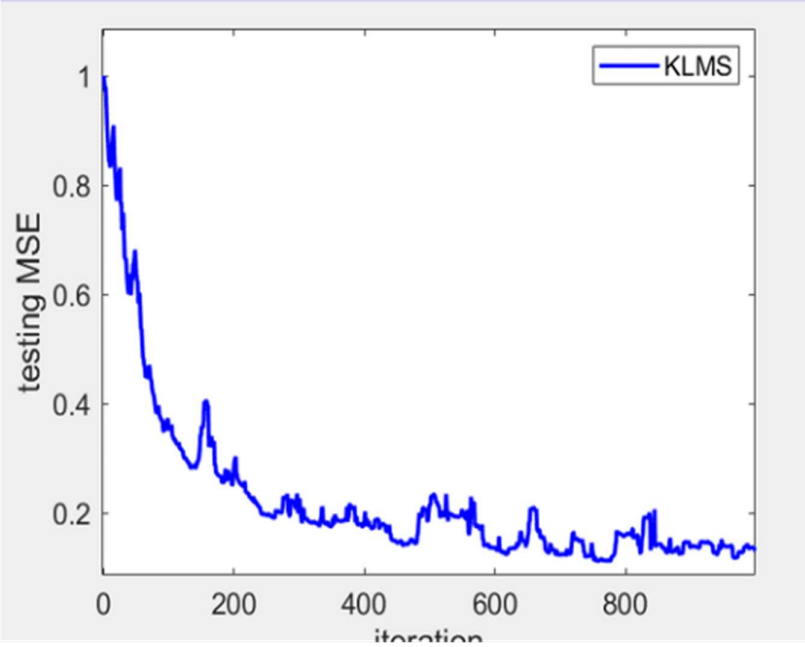
$$W_{New} = W_{Old} + 2\mu e(n).\hat{s}(n) * x(n)$$

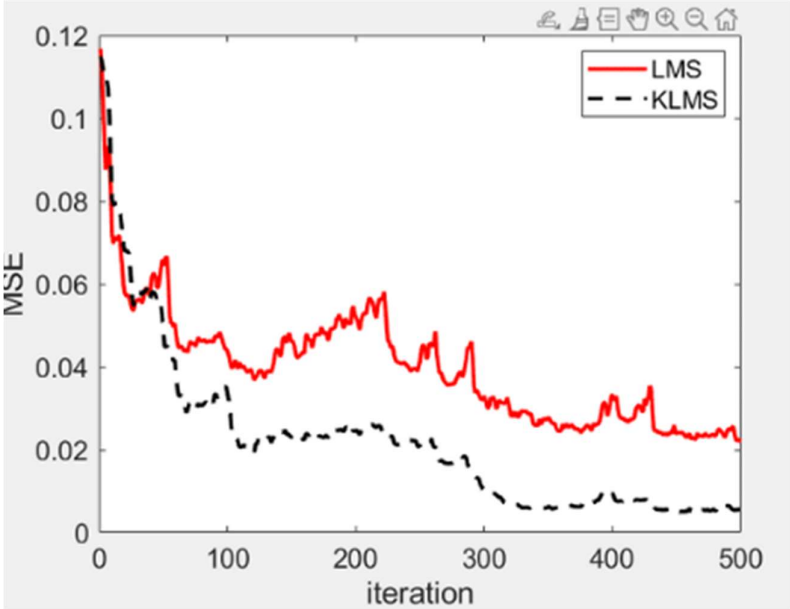
OR

$$W_{New} = W_{Old} + 2\mu e(n).x_f(n)$$



KLMS –


KLMS
testing MSE
1
0.8
0.6
0.4
0.2
0
200
400
600
800


LMS
KLMS
MSE
0.12
0.1
0.08
0.06
0.04
0.02
0
0
100
200
300
400
500
iteration

CONCLUSION –

LMS is based on steepest descent method, but do not include secondary path effects, so precise anti noise signal cannot be generated. Its basic idea is to keep on updating the weights of the filter to minimize the MSE. The FxLMS algorithm is again computationally simple where secondary path effects are also includes, but its convergence speed is slow. It is LMS with KLMS functions. on comparing them we found that KLMS performs drastically better than LMS. The main reason for the same is that KLMS turns the input data input data in high dimensional feature space via a reproducing kernel. we do it in such a way that the inner product operation in the feature space can be computed efficiently through the kernel evaluations.

References -

https://www.youtube.com/watch?v=kXJ_WQlweiI
Research paper from Weifeng Liu CNEL (July 1, 2008)
https://www.math.uh.edu/~razencot/MyWeb/docs/workshop/NicolaosMitsakos_KernelRegression.pdf