DSP LAB PROJECT REPORT

Real-time Adaptive Noise Cancellation

Name of the students:

Anshu Mathur (am10263)
Megha Veerendra (mv1807)
Vishnu Annapareddy (mra503)

Professor	**Ivan Selesnick**
Department	**Electrical and Computer Engineering**
Course	**DSP Lab**
Date	**12/21/2020**

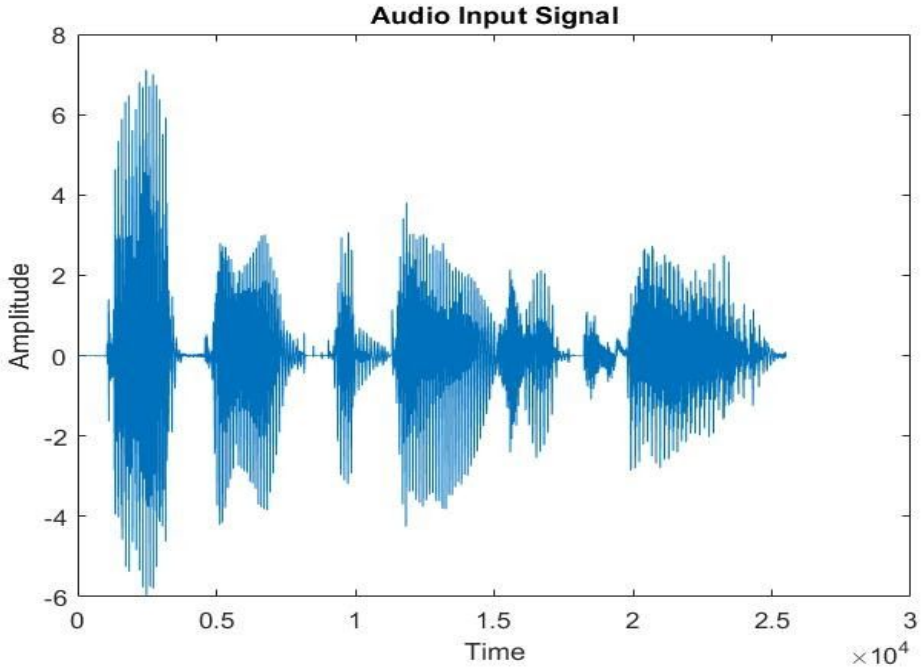
AIM : Objective of this project is to perform adaptive noise cancellation (ANC) using Adaptive Filters. The concept that we applied here is of destructive interference of two sound waves. We perform Active Noise Cancellation using three algorithms - LMS, NLMS and RLS. We also draw comparisons between the three algorithms.

INTRODUCTION

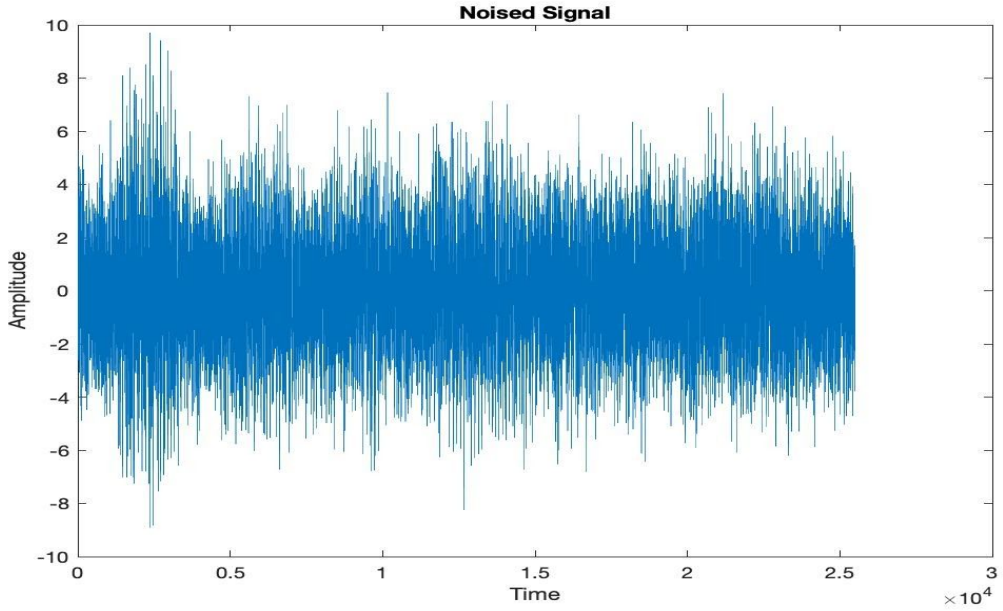
Adaptive Noise Cancellation refers to the application of Adaptive filtering algorithms to noise cancellation. Noise cancellation in signal processing means removal of unwanted noise or unwanted frequencies from the signal. Active Noise Cancellation makes use of Adaptive Filtering, which is a filtering method that looks at sampling audio waves analysing the input audio for noise and then produces a destructive wave of the same frequency and amplitude as the noise present in the audio wave which results in cancellation of the noise. We implement three algorithms - Least Mean Squares (LMS), Normalized Least Mean Squares (NLMS) and Recursive Least Squares (RLS). We compare the performance of the three adaptive filters. We designed a Graphical User Interface (GUI) using Matlab where we plot various graphs for showing the transition of the input audio from an unfiltered noisy signal to a filtered signal.

THEORY

Adaptive Filters use a cost function for optimization which is generally the mean square (MSE) of the error signal between the output (given by the adaptive filter) and the digital signal. The filters modify the filter weights so as to minimize the MSE. We get the output of the filter as very close to the desired output. They are used to estimate a random signal by using its current value (noisy signal).



The Input Signal



Input Signal with white gaussian noise

FILTERS USED

1. Least Mean Squares (LMS) Adaptive Filter

LMS Algorithm is derived from the steepest descent algorithm. We find out the gradient at every iteration. LMS filters find the filter weights by producing the least mean square squares of the error signal. In this algorithm, the filter is updated based on the error at the given current time. The LMS Algorithm iteratively changes the weights of a Finite Response Filter (FIR).
The output signal, y(k) is calculated as follows :

$$y(k) = u(k)^{\mathrm{T}} * w(k)$$

Where,
u(k) : input filter vector
w(k) : weights (a FIR filter)
μ : fixed step size of the filter
d(k) : reference signal to the adaptive filter

Error signal [e(k)] can be calculated as :

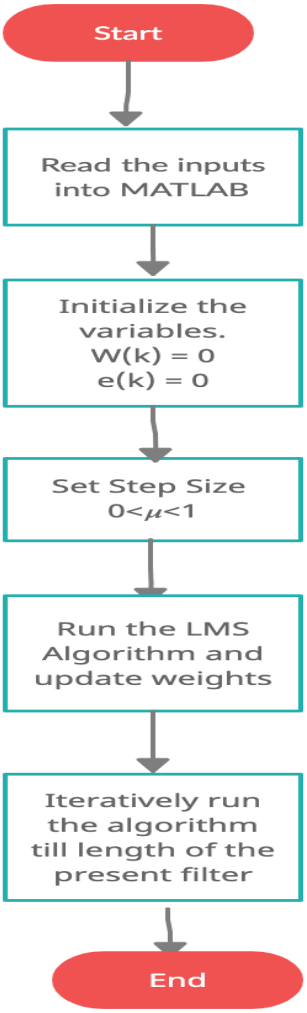
$$e(k) = d(k) - y(k)$$

Filter coefficients are updated based on e(k) :

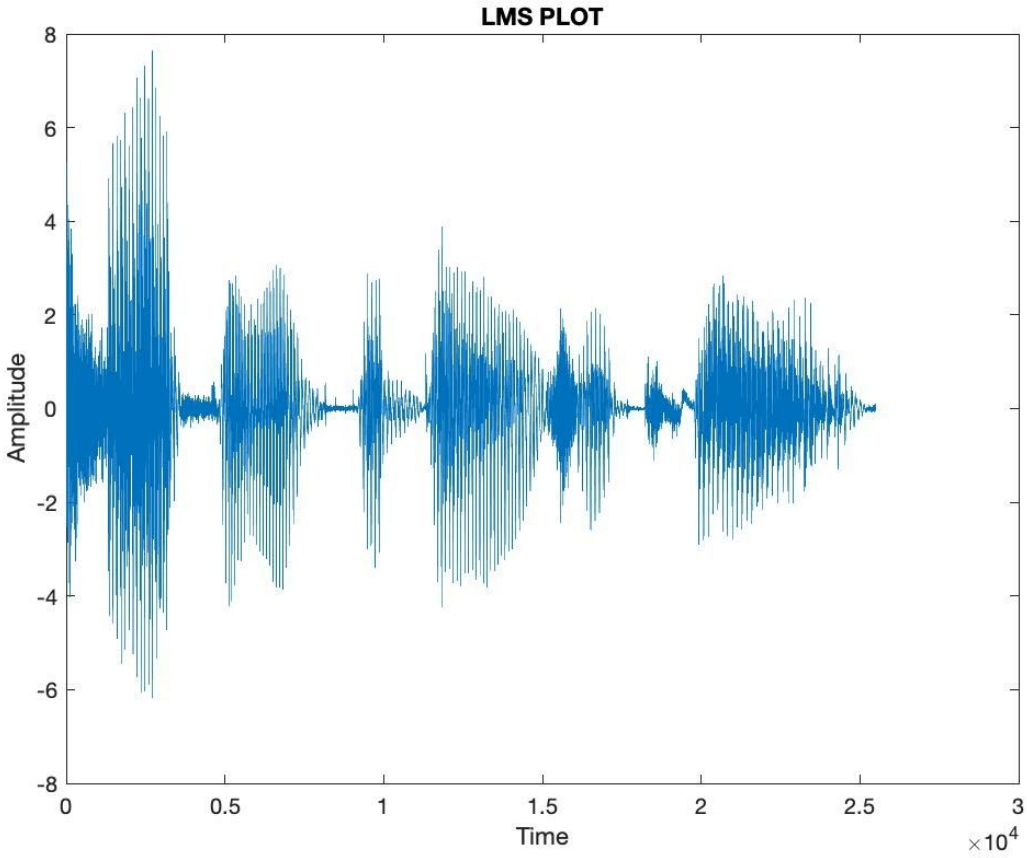
$$w(k+1) = (1-\mu).w(k) + \mu.e(k).u(k)$$

Larger values of μ will increase the adaptation rate but it will also lead to increase in residual mean-squared error.

Flow Chart for LMS :


Start
Read the inputs into MATLAB
Initialize the variables.
W(k) = 0
e(k) = 0
Set Step Size
0<μ<1
Run the LMS Algorithm and update weights
Iteratively run the algorithm till length of the present filter
End

Plot for LMS



LMS Filtered output audio signal

LMS takes longer time to converge. Even though LMS does not make use of the exact values of expectations and weights would never reach the optimal weights in absolute sense, still convergence of the mean is possible. If the variance is large, then the convergence could be misleading. A larger value of μ may lead to a faster convergence but may be less stable around the minimum value.

2. *Normalized Least Mean Squares (NLMS)*

NLMS is similar to the LMS Algorithm. NLMS varies from LMS due to the difference in step size. LMS is not scalable with increasing input. It becomes really difficult to choose a learning rate μ that guarantees stability of the algorithm. In NLMS, the step size is adapted according to the signal power and amplitude.
The output vector is calculated as :

$$y(n) = v_2(n)\, w(n)H$$

Where,
w(n) : current weight
v2(n) : input signal
y(n) : output signal

Error signal is calculated as :
e(n) = d(n) - y(n)

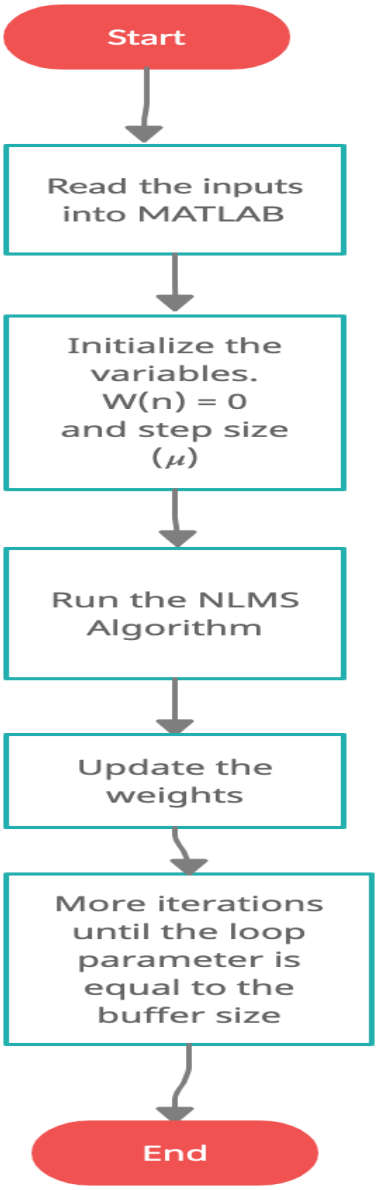
Where,
e(n) : error signal
d(n) : desired signal

The weight in NLMS algorithm is :

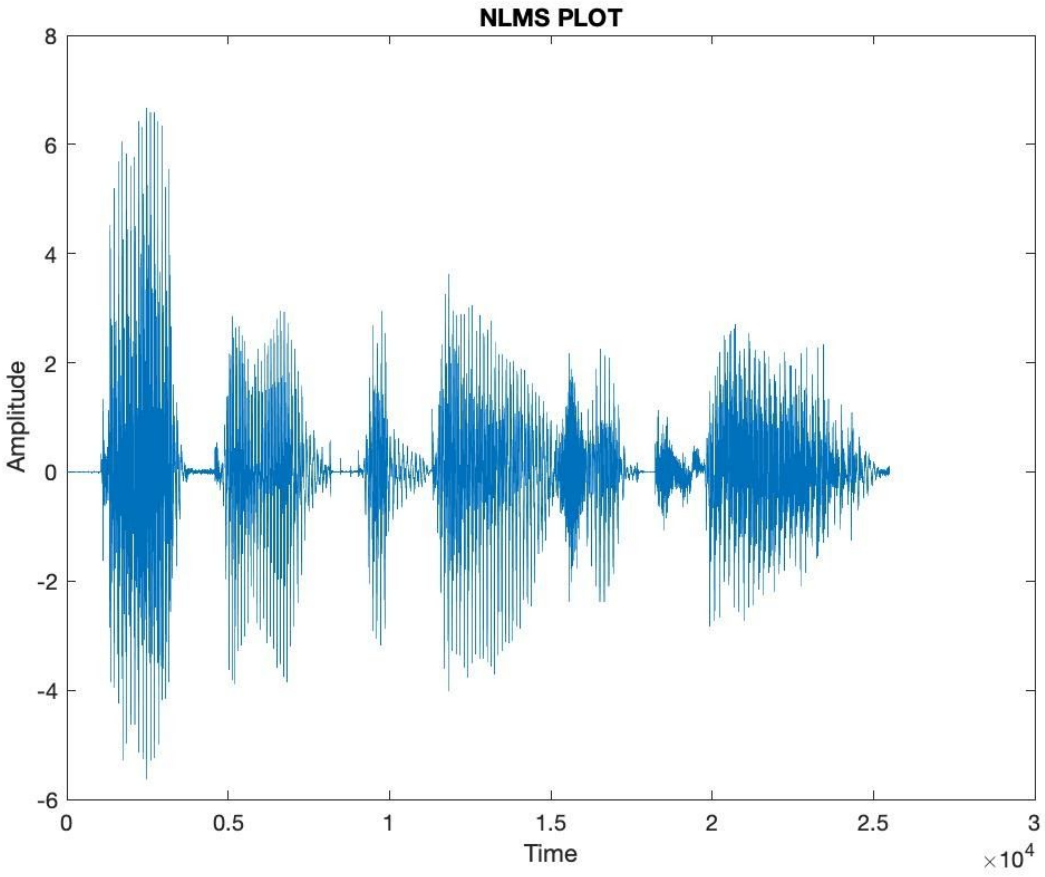
$$w(n+1) = w(n) + \frac{\mu}{\varepsilon + ||v_2(n)||^2} v_2(n)e(n)$$

We consider both Mean Square Error and Signal-to-Noise Ratio for better comparison.

Flow Chart for NLMS :



Plot for NLMS



NLMS Filtered output audio signal

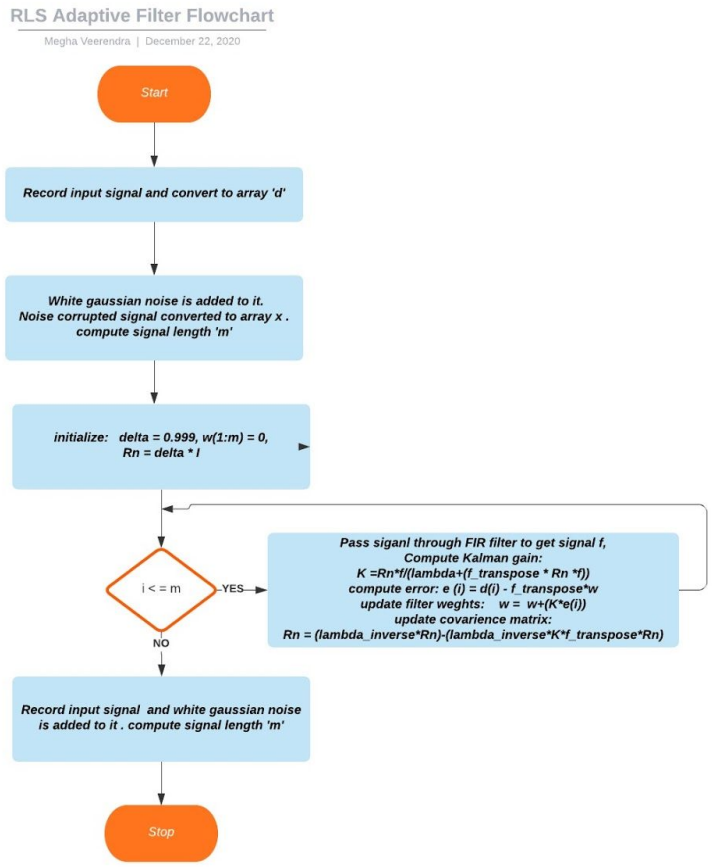
3. Recursive Least Squares (RLS)

RLS is an algorithm that finds the filter weights recursively so as to minimize a weighted linear least squares cost function with respect to the input signals. RLS works best in time varying environments but has very high complexity at the same time. In RLS, we need to know the estimate of previous samples of output signal, error signal and filter weight and hence, higher memory requirements.

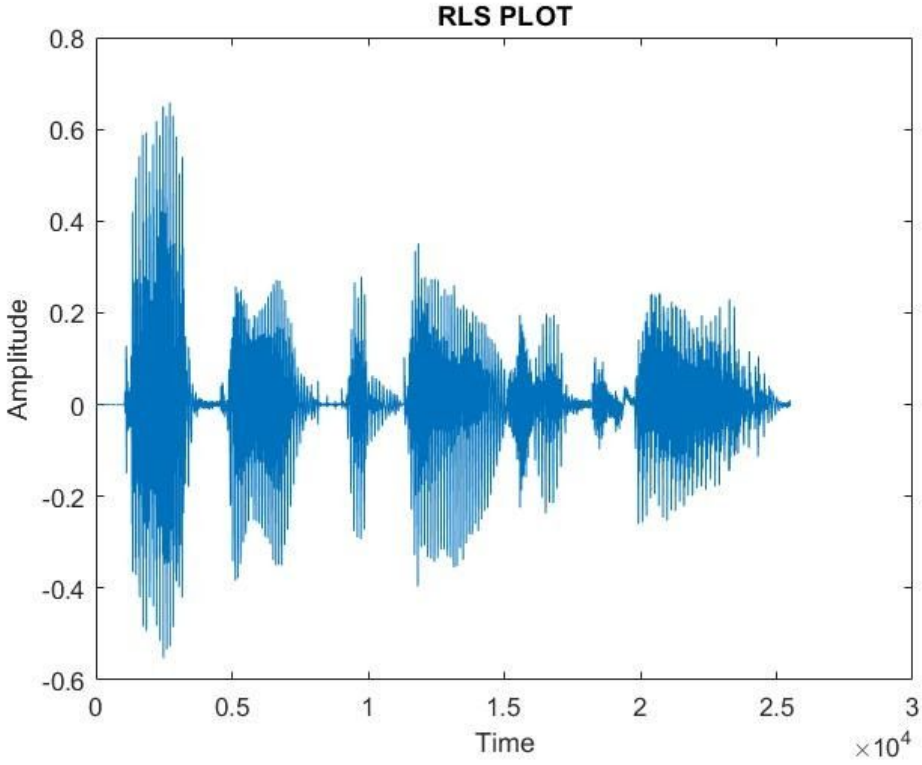
The weight vector is updated using the following equations. Where ‘k’ is the *Kalman Gain*. RLS works similar to a Kalman Filter. It gives an estimate of the state of the system as an average of the system’s predicted state and of the new measurement using a weighted average. Kalman Gain is the relative weight

given to the measurements and the current state estimate and it can be tuned to achieve the desired performance.

Flow Chart for RLS :



Plot for RLS



NLMS Filtered output audio signal

Comparison between LMS, NLMS and RLS

We will compare LMS, RLMS and RLS considering the following parameters for which we have data as shown.

- **Time for Code Execution** - We calculated time of execution for each algorithm - LMS, NLMS and RLS.

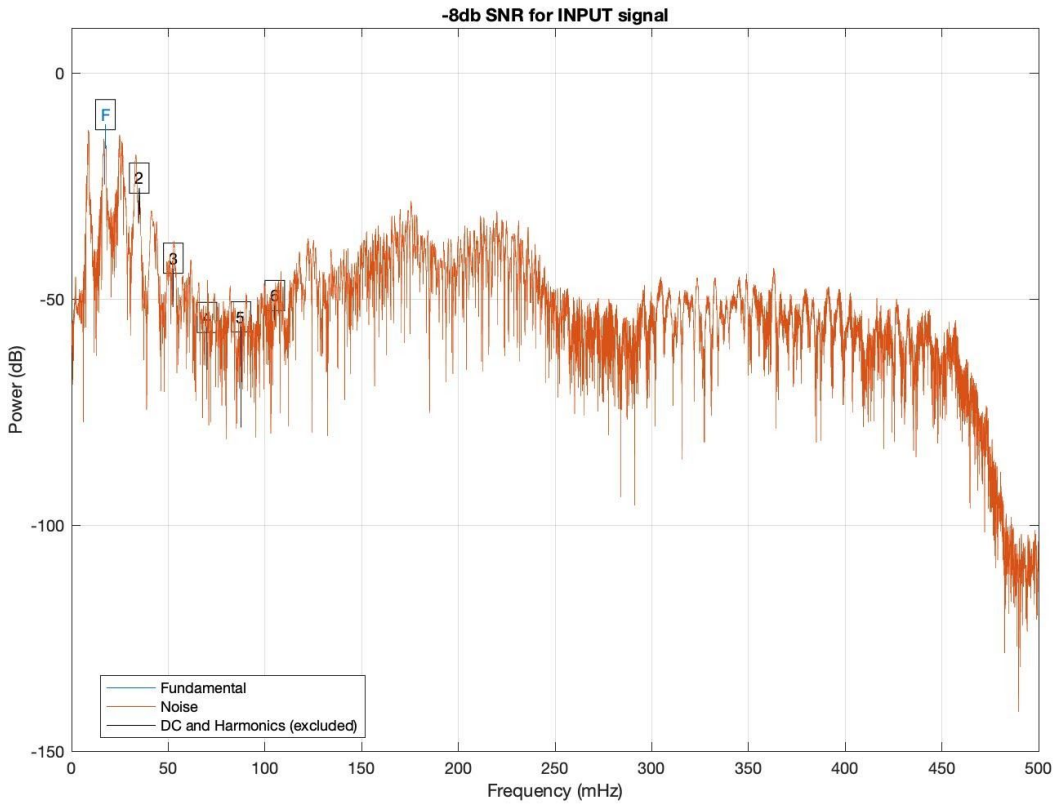
Algorithms	**Time for Code Execution(in seconds)**
LMS	0.069
NLMS	0.108
RLS	2.518

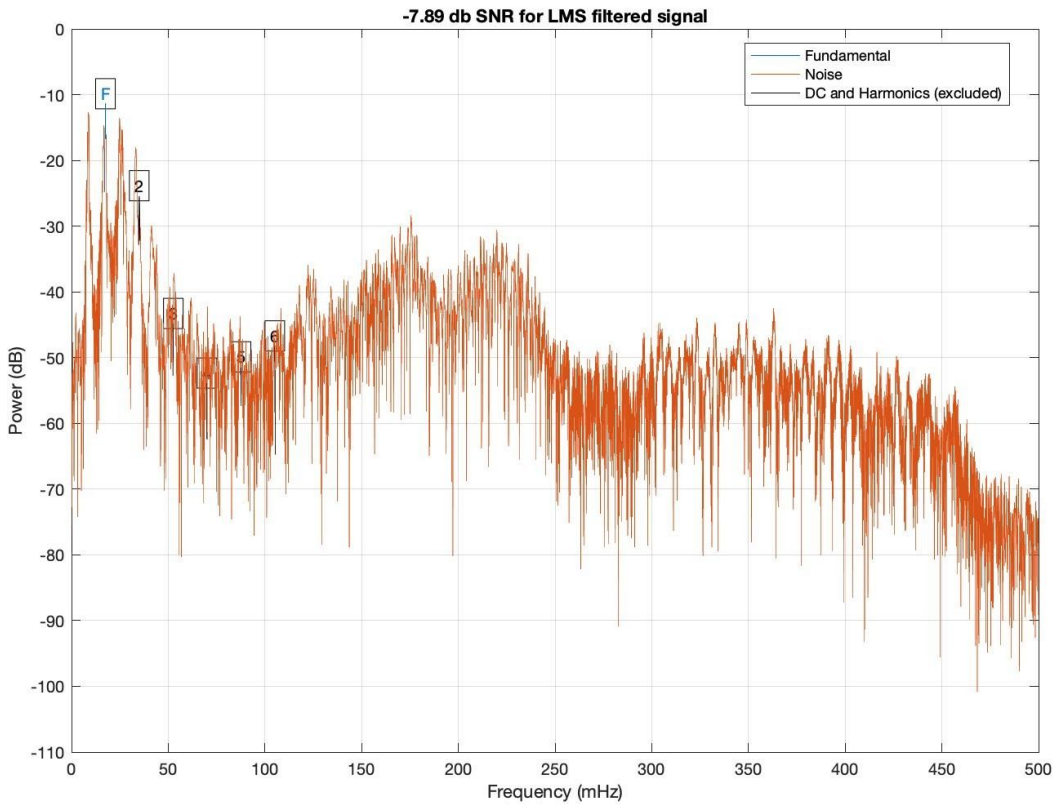
- **Mean Squares Error (MSE)** - It is the mean of the squares of the errors. It is observed that RLS has the least MSE, followed by LMS and NLMS.

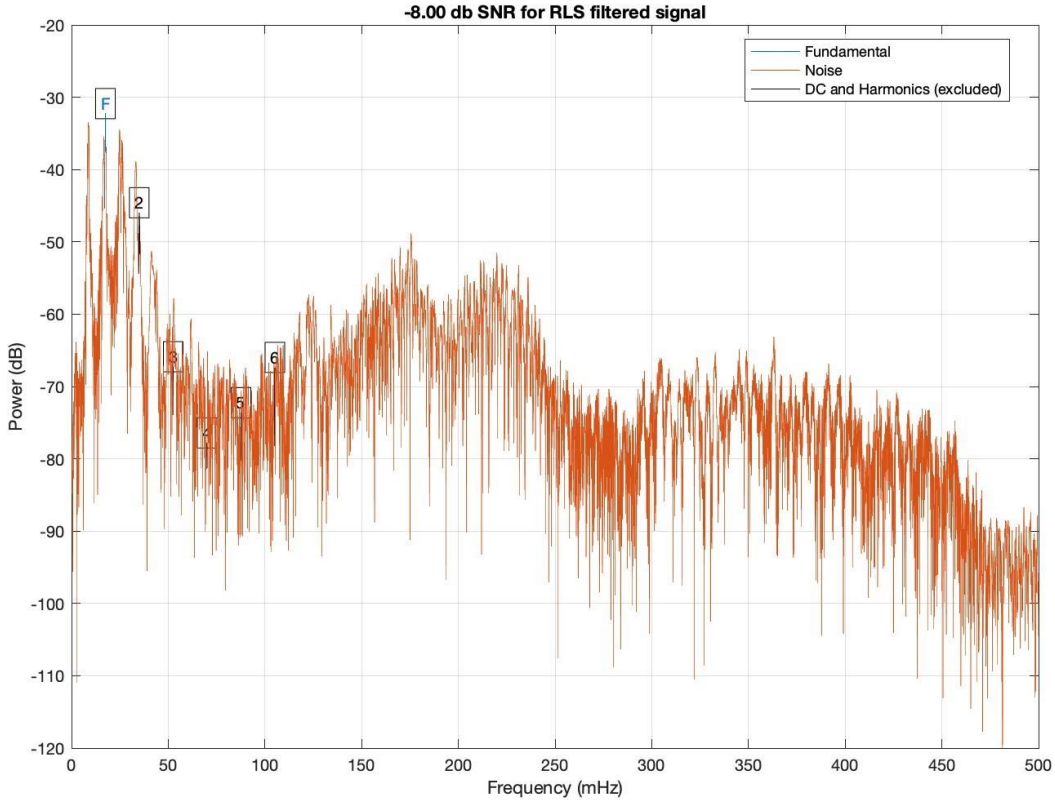
Algorithms	**Mean Square Error(MSE)**
LMS	0.0021
NLMS	0.0599
RLS	0.0014

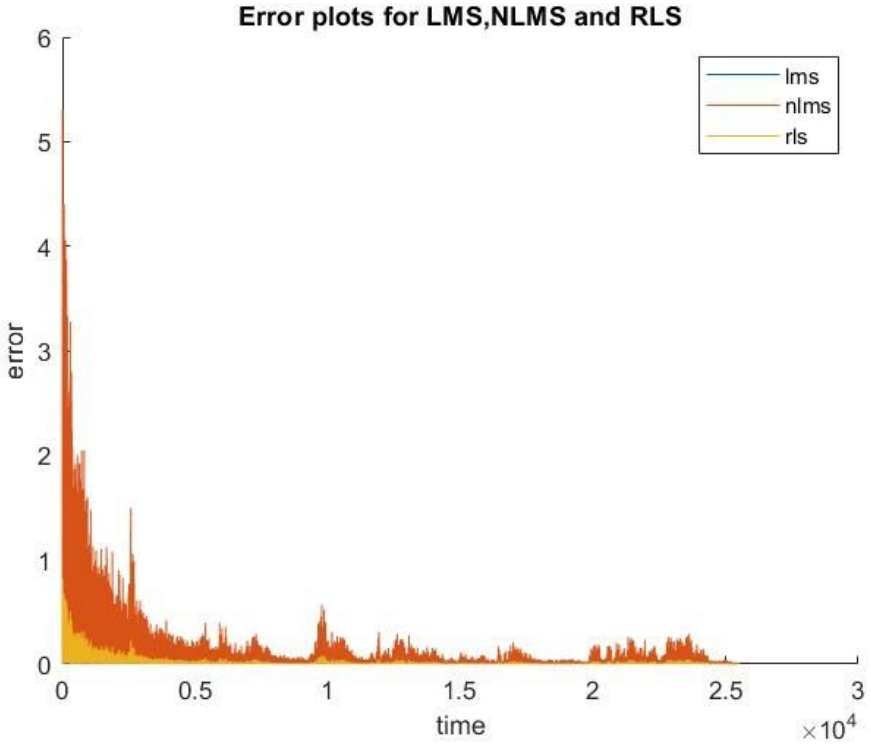
- **Signal to Noise Ratio (SNR)** - SNR defines the difference in level between the signal and the noise for a given signal level. The lower the noise generated by the receiver, the better the signal to noise ratio. We will plot SNR for all the three algorithms. From the plots below, it can be clearly seen that RLS has the best SNR value.

	LMS	NLMS	RLS
Time of Code Execution (in seconds)	0.069	0.108	2.518
SNR (in dB)	-7.89	-7.99	-8



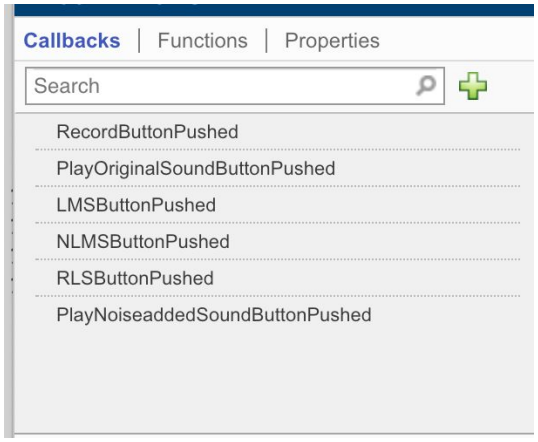

-7.89 db SNR for LMS filtered signal
Fundamental
Noise
DC and Harmonics (excluded)
F
2
3
4
5
6
Power (dB)
Frequency (mHz)
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
0
50
100
150
200
250
300
350
400
450
500

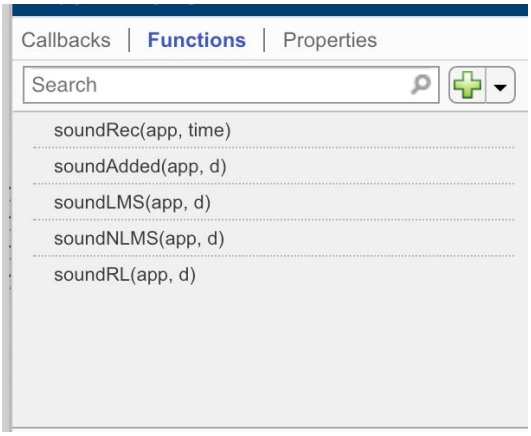

-8.00 db SNR for RLS filtered signal
Fundamental
Noise
DC and Harmonics (excluded)
F
2
3
6
5
Power (dB)
Frequency (mHz)
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
0
50
100
150
200
250
300
350
400
450
500


Error plots for LMS,NLMS and RLS
lms
nlms
rls
error
time
×10^4
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3

Observations using GUI

We used ***Matlab Appbuilder*** to make a GUI which will ease the process of comparing noise cancellation using different filters and make our code more user friendly. So after recording the voice the green lamp goes off, then by clicking the filter buttons we select the filter that will be used to cancel the noise. It will play the noise filtered signal as soon as it processes the signal. One is the original signal, second is the Noise added signal and final is the filtered result Noise canceled one.





CONCLUSION

We perform adaptive noise cancellation using three Adaptive Filter Algorithms - LMS, NLMS and RLS. We draw comparisons between the three algorithms based on some parameters for which we have data. From the results we conclude that RLS is a better filter than LMS and NLMS. However, the execution time of RLS is higher than the other two algorithms.

Project Zip File contents

The zipfile contents are as follows:

1. Matlab GUI
2. LMS.m, NLMS.m and RLS.m - contain the respective algorithms
3. Plots.m - used to plot the data from the above files
4. Data files: data_lms.mat, data_nlms.mat and data_rls.mat which contains the data files saved from the files in 2.
5. 2,3 and 4 form Comparison_codes

***Contribution of each team member** :*

Anshu Mathur - LMS

Megha Veerendra - RLS

Vishnu Annapareddy - NLMS

The GUI and report has been completed as a combined effort by all the three team members.