# SSY130- Project 2

Adaptive Noise Cancellation

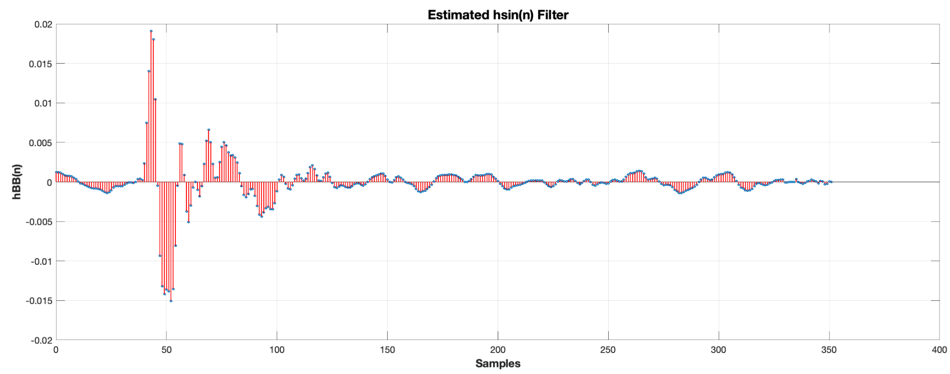
Group 8

Kashif Shabir
Haitham Babbili
Muhammad Haris Khan

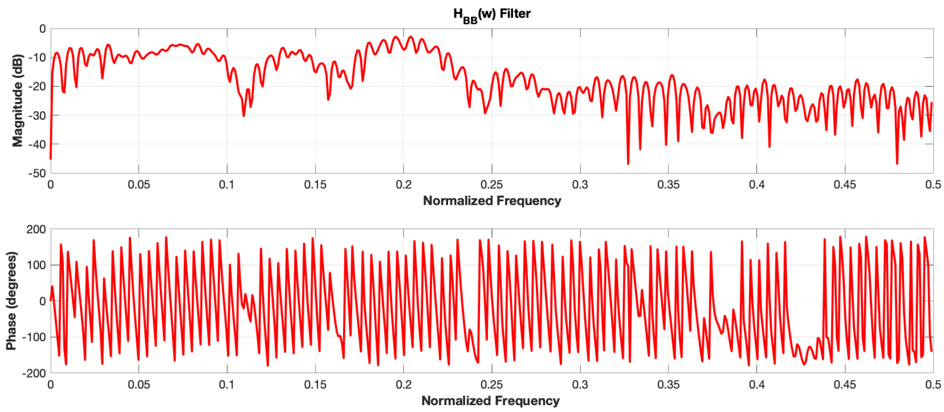
January 26, 2020

## Empirical Section

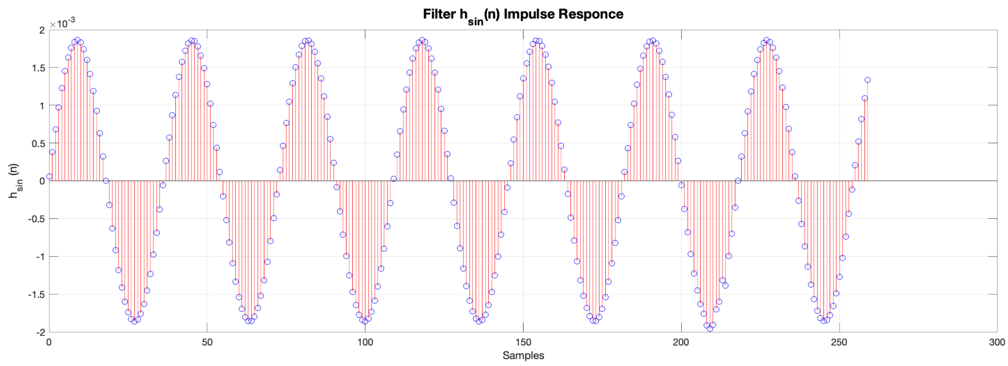
**1.**



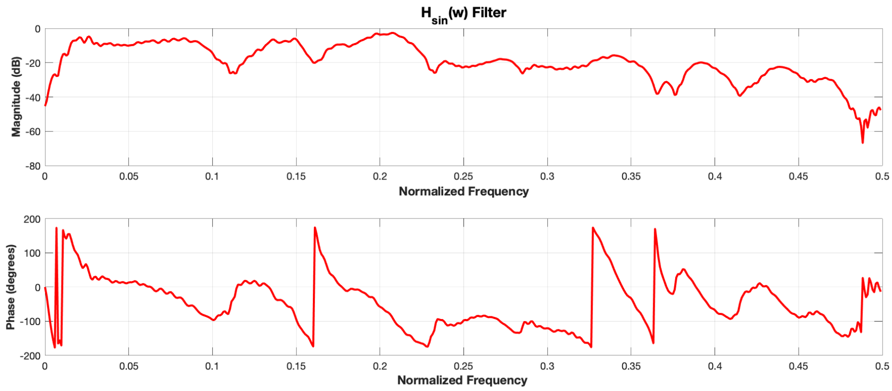

(a) Impulse response for $h_{BB}$




(b) Amplitude and phase response for $H_{BB}$




(c) Impulse response for $H_{sin}$




(d) Amplitude and phase response for $H_{sin}$

**Figure 1**

**2.**

(a)

In this section, we will compare the error performance of the channel without updating the estimated channel. Figure 2.1 shows the error performance before changing the channel and 2.2 shows the error performance after changing the channel. The error values increased as compared to previous because we didn't update our system according to the requirements since the estimated channel is different than the actual channel that's why we are getting high errors and figure 2.2 depicts the increase in error as expected.

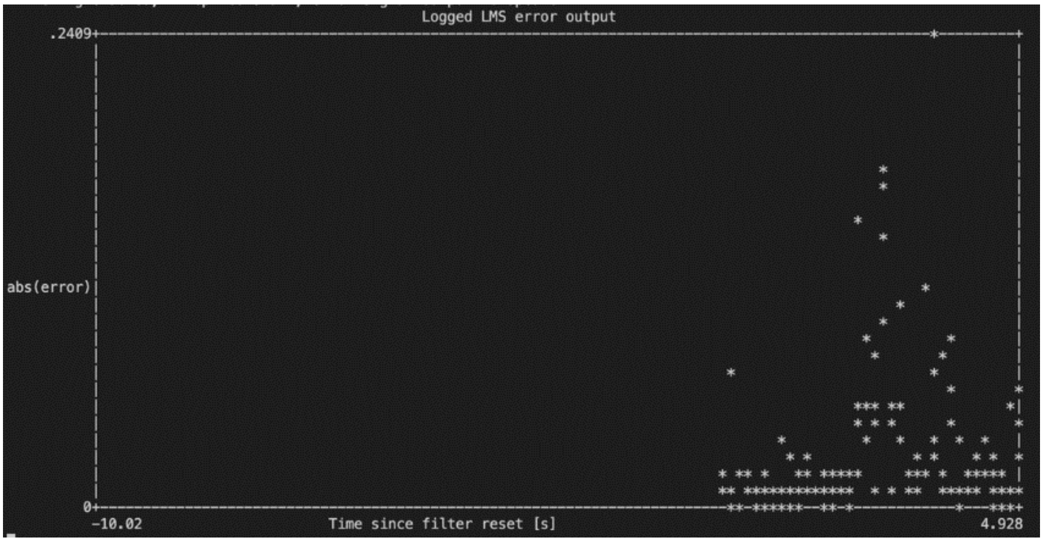



**Figure 2.1** Before changing the channel

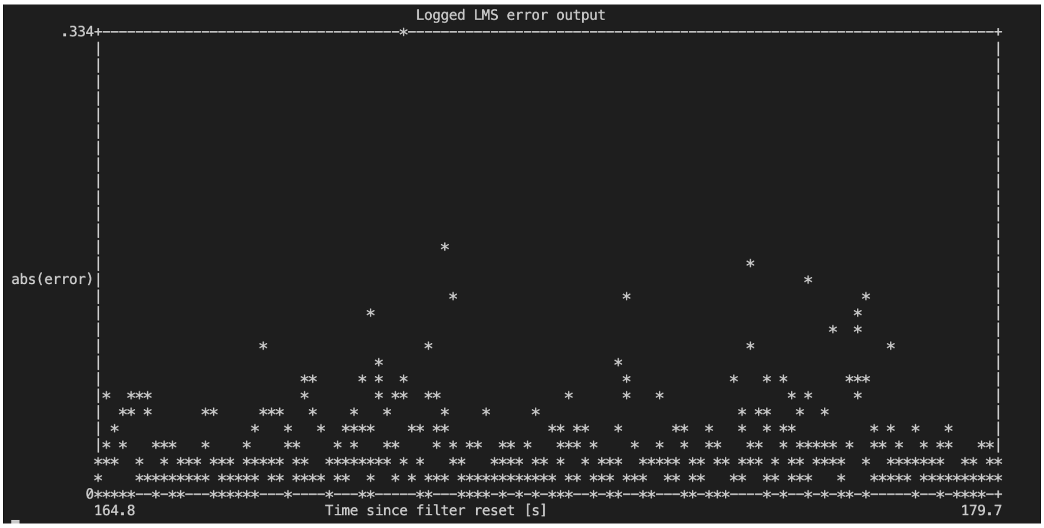



**Figure.2.2** After changing the channel

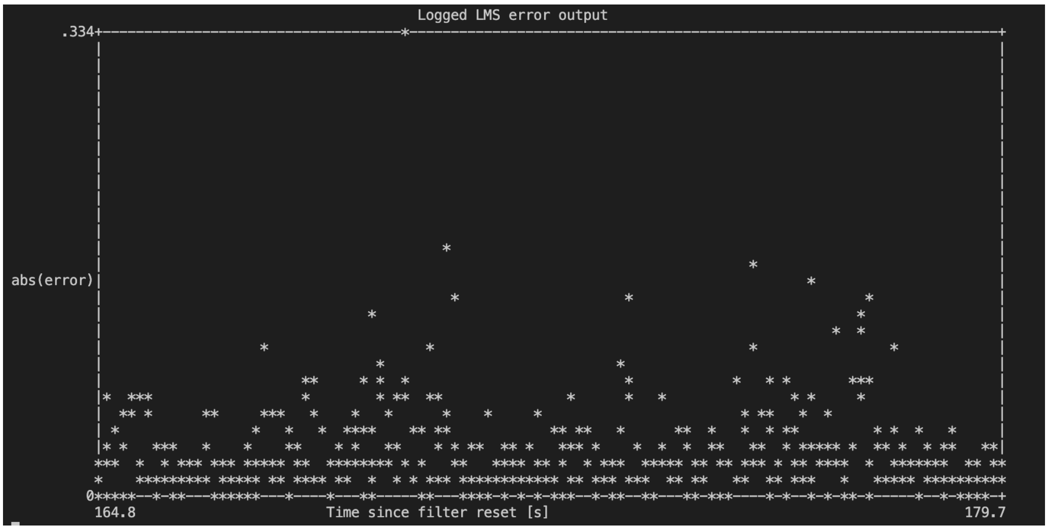
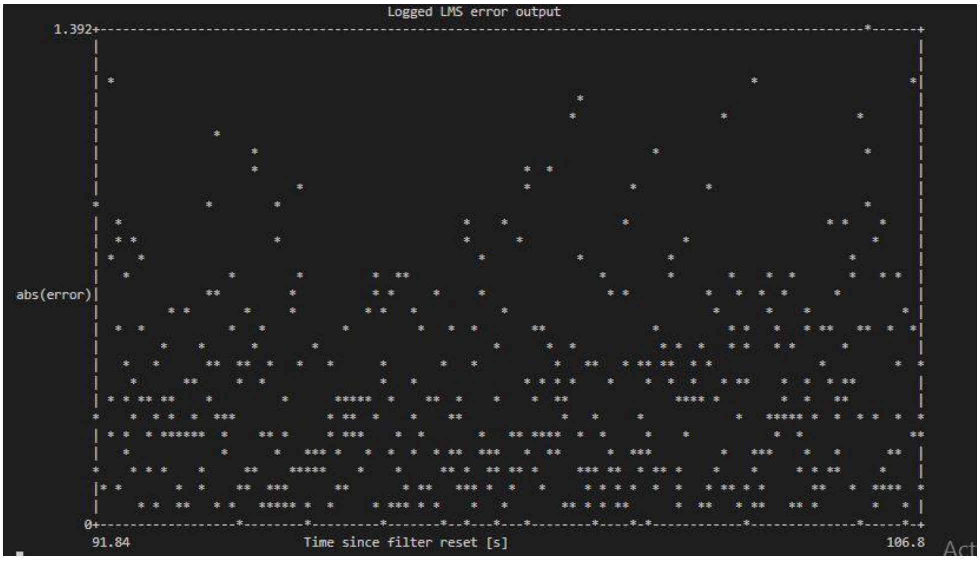
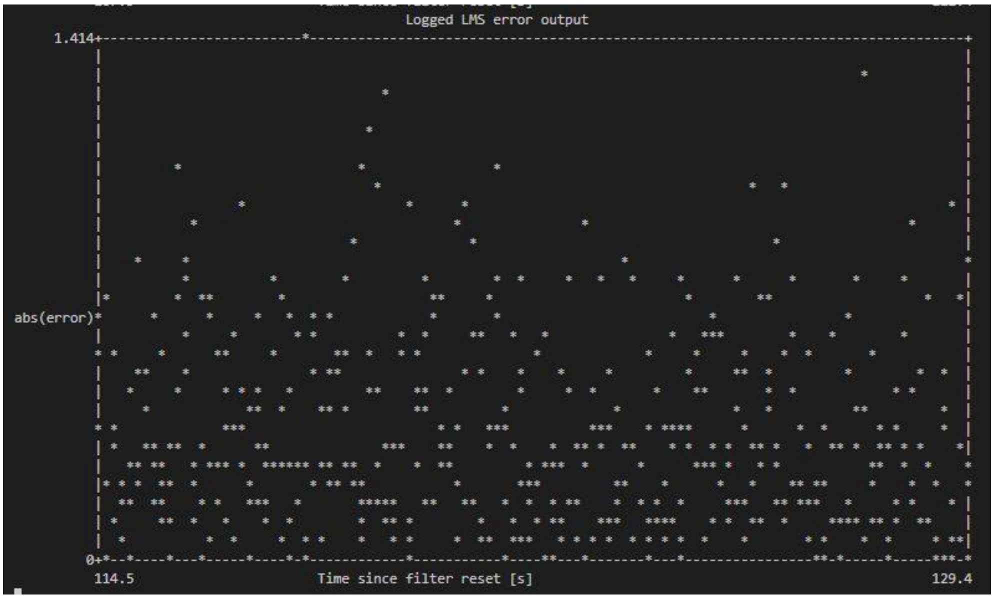



**Figure.2.2** Error after changing channel

(b) Here, we will compare the error performance by fixing the filter coefficients without updating them against the increase and decrease in volume by pressing +/- keys on DSP-kit. The increase in volume means we are increasing amplitude of the signal that will give us less error due to assigning high energy and decreasing the volume means less energy which ultimately give us high error probability and figure 3 results are in exact agreement what we mentioned above.



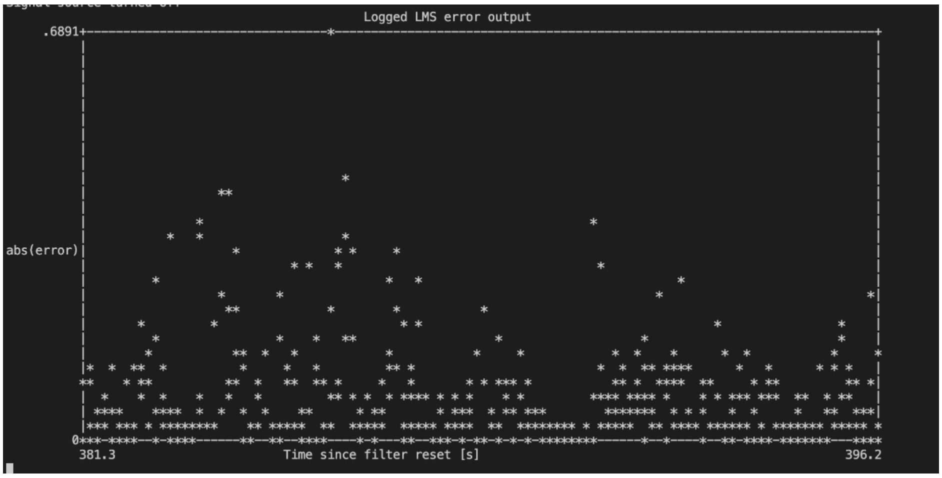

(a)



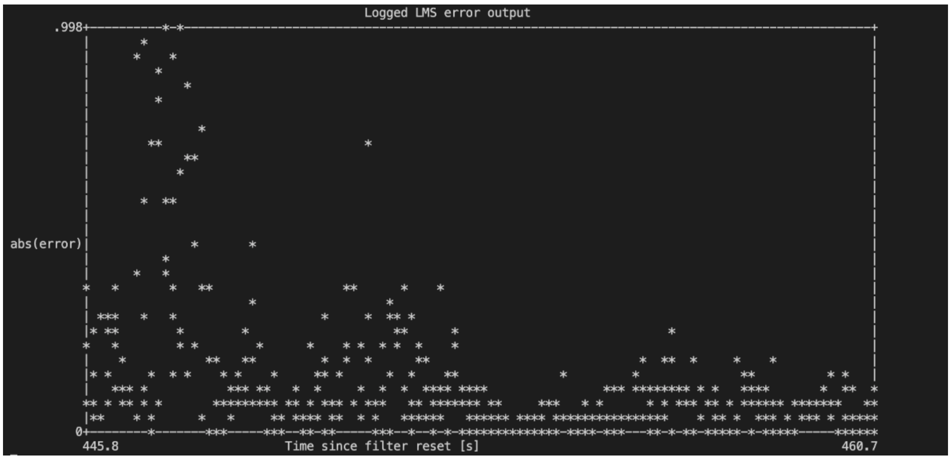

**(b)**

**Figure.3** The effect of (a) increasing and (b) decreasing the volume throw the time

(c) We hear the better voice as compared to previous case because when we play music on our mobile the channel noise almost acts constant and our LMS is quite able to eliminate or suppress this noise effectively because error function= music (mobile). Even moving the mobile around the DSK-kit will change the channel but still our algorithm suppress the noise effectively since still channel is almost same for the same noise. The figure 4 shows the whole story.




(a) When mobile is so close




(b) When mobile move around

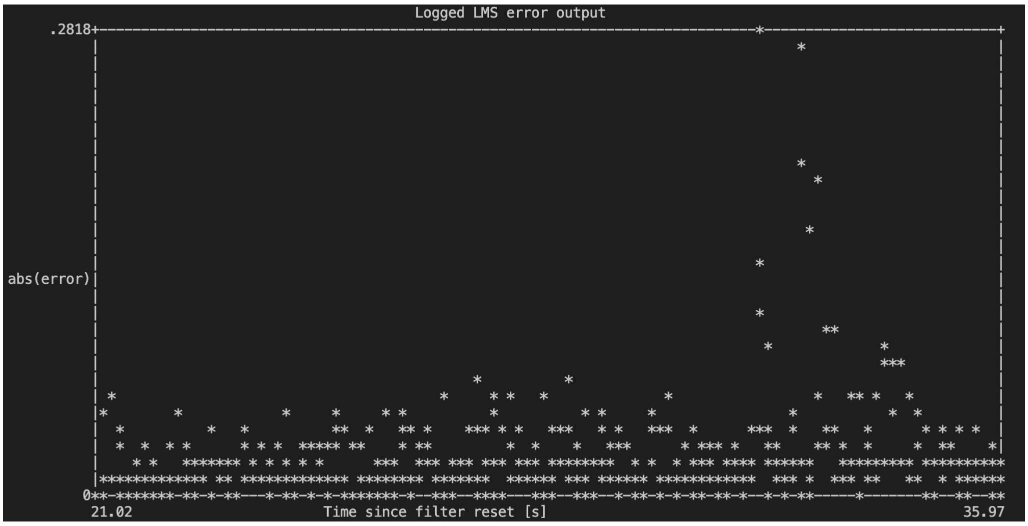
**Figure.4**

**3.**

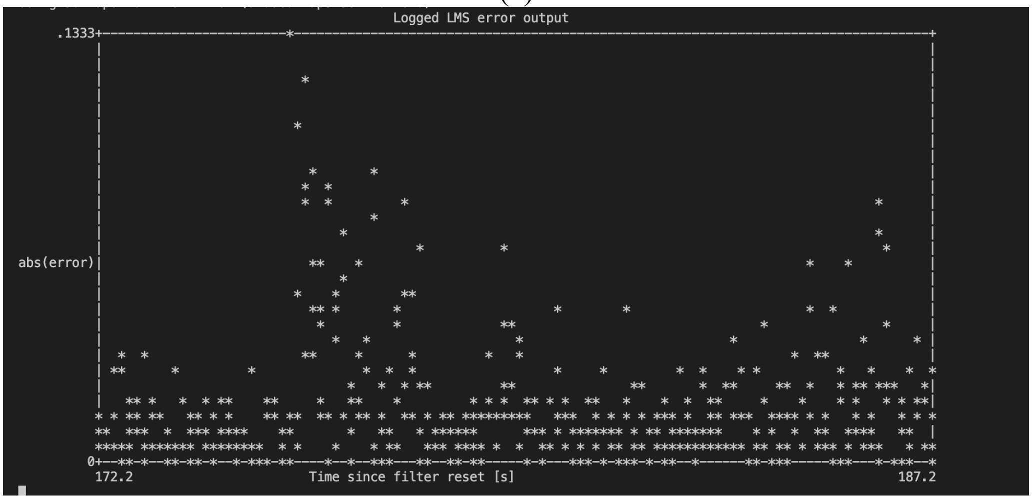
It is observed while performing experiments that the results for both cases (resetting and w/o resetting) are alike with almost same scale of errors.

However, filter error performance isn't working equally well for all filter taps. From the figure (5) we can say that by decreasing the filter taps, error will increase, and noise cancelation performance also affected negatively and vice versa.

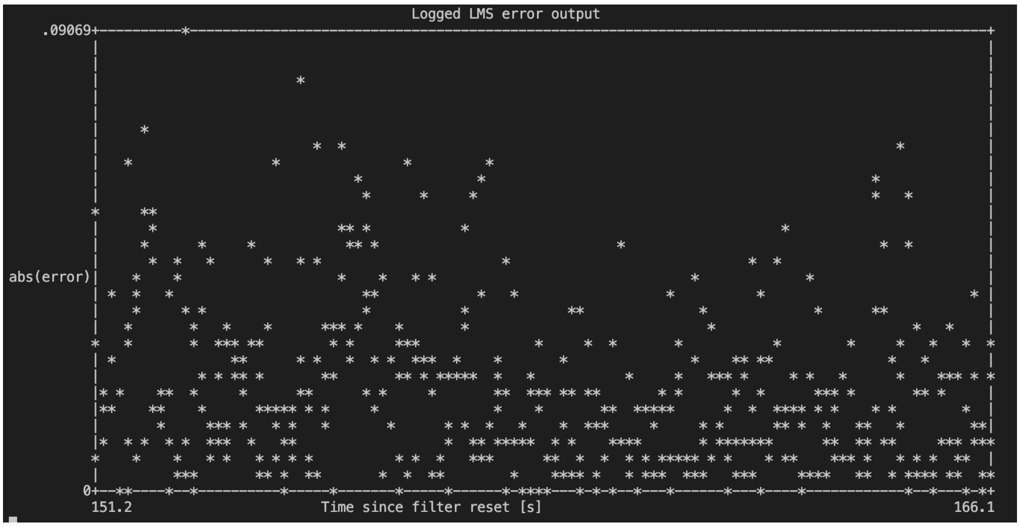
While performing experiments we found that when we reduce taps below 65, the noise cancellation quality decreased sharply. Hence, 65 number of taps will be the threshold point in order to achieve optimal performance. Below that the delay in the transmission and reception of the signal became high in such a way that we became unable to eliminate noise easily.




(a)




(b)

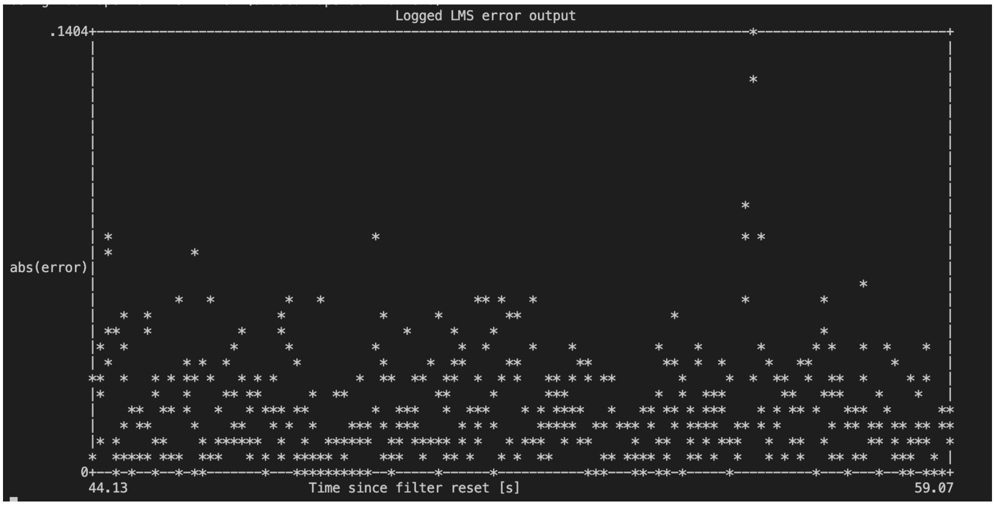



(c)

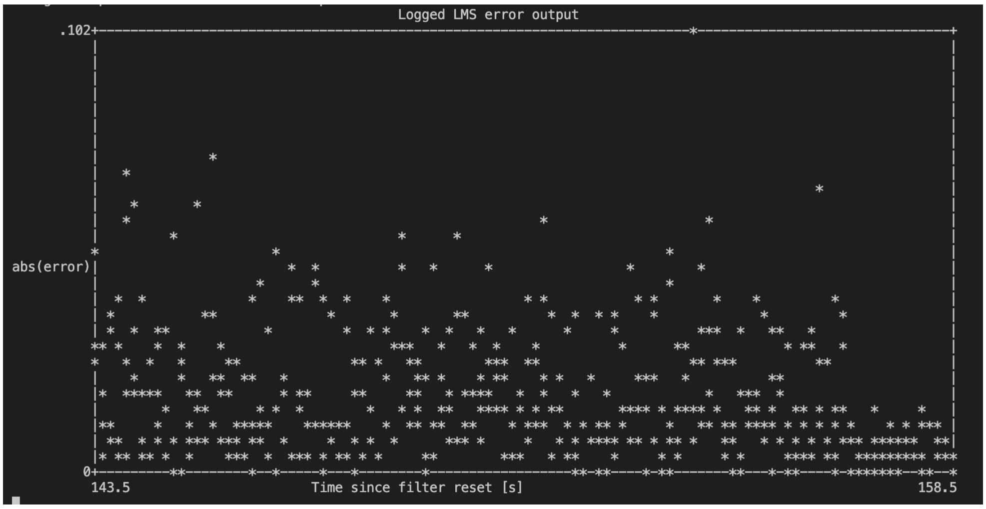
**Figure.5** LMS error signal at different taps (a = 100, b = 50, c = 25)

**4.**

Our observations for the hsin is that the shape of the coefficients remain same when we decrease values from 100 to 10 for both cases. Even, resetting and filter length aren't affecting error performance means we are getting same performance. This can be clearly seen in the figure below.




(a) Without resetting the coefficients




(b) Resisting the coefficients

**Figure (6)** $h_{sin}$ from 100 to 10

We repeat the test by increasing the coefficient's value from 10 to 100 obviously the amplitude and the phase remain the same and the additional coefficients added to the end with zero initial value. Based on that we don't have to reset the coefficients value when we change the length of the filter. The coefficients will converge under specific convergence conditions with some delay since it belongs to the convex optimization problem.

According to the figure we see $h_{sin}$ filter works well compared figure (7) until n = 0 and the results are quite better for coefficients greater than 10. The results for both filters are similar having very low error, even the filter coefficients are different. This means that $h_{sin}$ is not different based on we may use more parameter than we need.

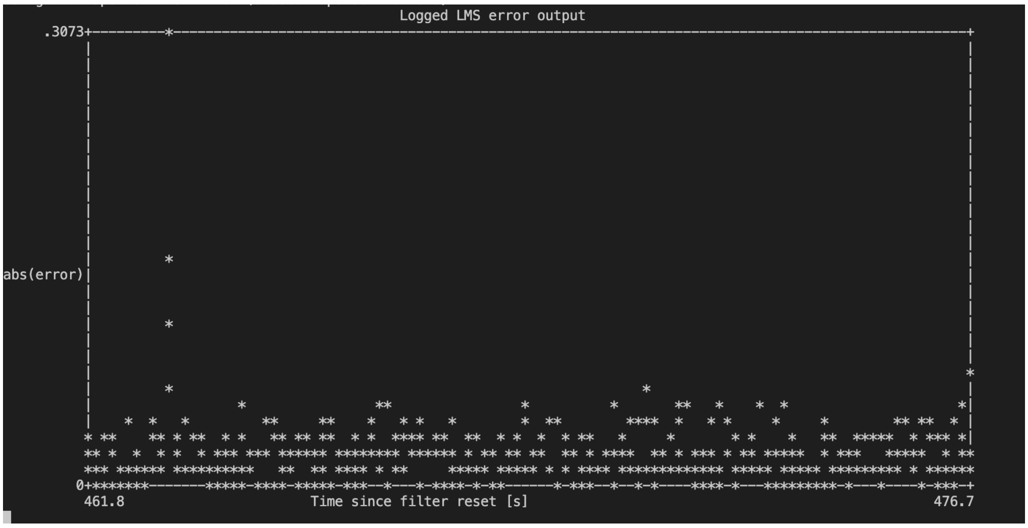
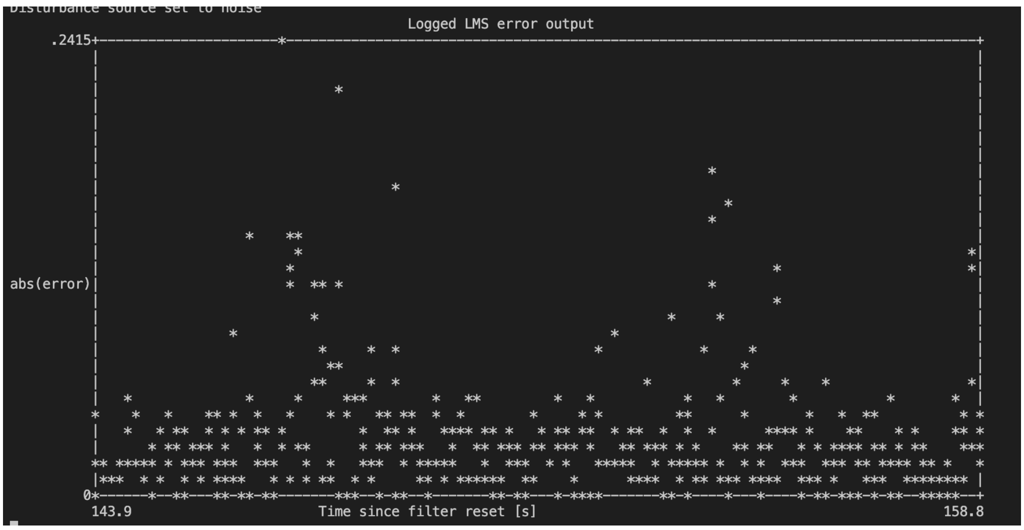


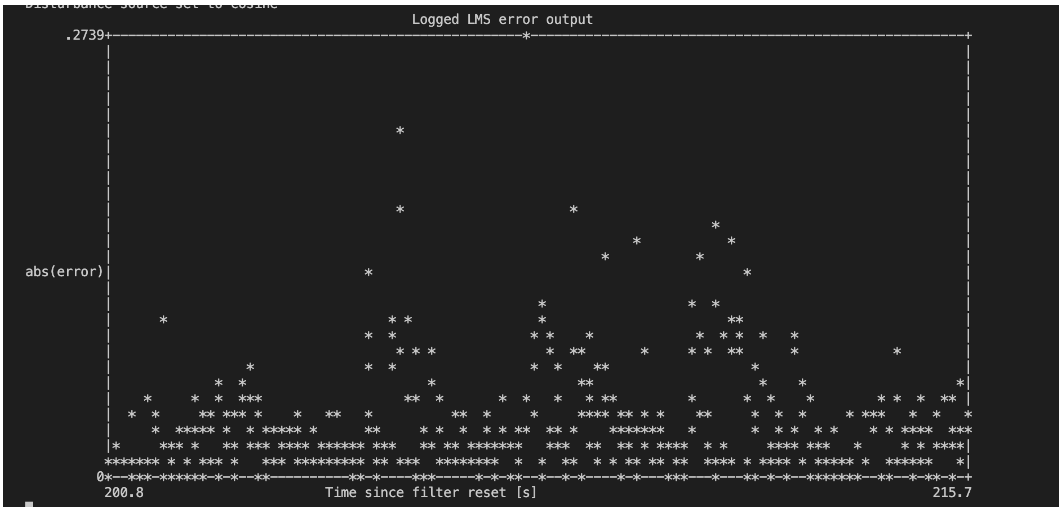

**Figure (7)** $h_{sin}$ from 10 to 100

**5.** When we trained for hBB and then switched to hsin, the error performance remains same which means hBB perfectly removed the sinusoidal disturbances. Contrarily, if we trained our filter for sinusoidal disturbance and then switched to hBB the noise level changed a lot which means hsin isn't able to remove broad band noise.

The reason behind that why both situations aren't showing the same performance is that when we trained our system for hBB which contains multiple frequencies and when switched to sin disturbance it eliminated the sinusoidal effect because that particular frequency was already present in hBB whereas in the second scenario when we trained for sin and switched to hBB the whole situation shifted because now we trained for one frequency but we are giving him multiple frequencies and for that our system was trained that is the reason we are getting high errors as compared to the previous case.




(a) $h_{BB}$ in sinusoidal noise




(b) $h_{sin}$ in broad-band noise
**Figure.8**

**6.** The signal from cellphone speaker which is large enough to saturate the channel or we can say in other words it will clip the channel. Non-linearity has been start observing in channel just because of this clipping, so convolution and linear operations can no longer be applied. The coefficients of the estimated saturated channel $h_{BB,sat}$ change completely from the not saturated channel $h_{BB}$. The LMS algorithm is trying to minimize the error, but because of non-linearity, error results will always be very high.

hBB,sat and hBB both are different as shown in figure 9 because when we use mobile music at high volume it saturated the channel which results in the form of clipping which can also be termed as non-linearities. This non-linear behavior is the major cause for that clipping and LMS is trying to equalize the saturated channel considering it linear by using linear filtering. However, due to non-linearity linear operations can no longer be applicable as expected that's the reason we are getting different hBB.

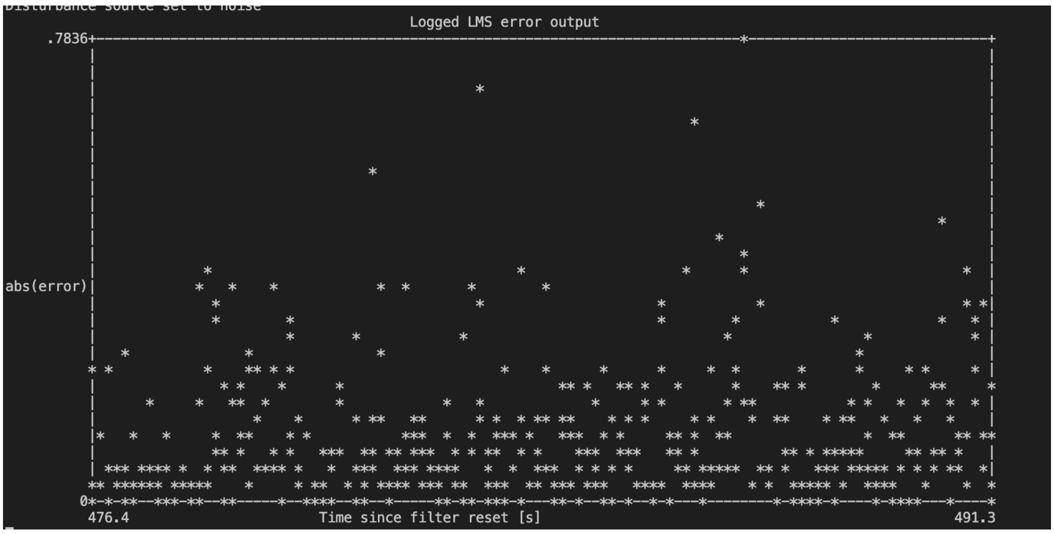



**Figure.9** Coefficient of the filter in saturation mode

**Analytical section**

1.
a- The step size of the filter defines the convergence rate of the filter. If step size is large, the filter will converge fast because it takes less time to achieve local optima and vice versa. The Eigen values of the auto-correlation matrix of received signal determine the optimal choice of mu.

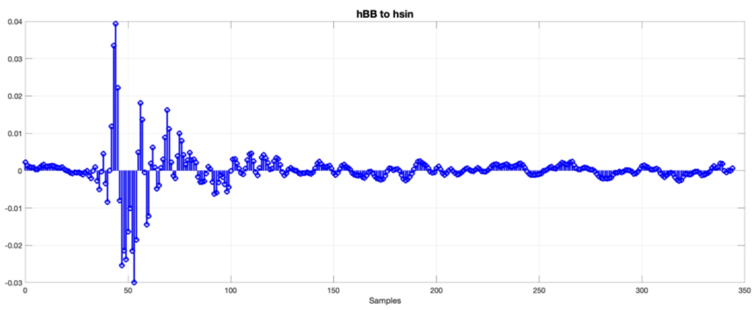
b- When we use large step size the estimated channel may diverge because the error function will start diverging from any specific point because we are taking less filter coefficients and each iteration actually depends on two points: the previous error signal and mu. If selected mu is too large or error increases if we take large step size then LMS algorithm will diverge.

c- After achieving convergence, the step size affects the filter performance as well. Since step size leads to the rate of the convergence, greater the step size faster the rate of convergence and vice versa. However, larger step size give us a bigger zone and vice versa. After achieving convergence, if we the filter behavior lies within those convergence zones then it will show similar performance else it will start diverging again.

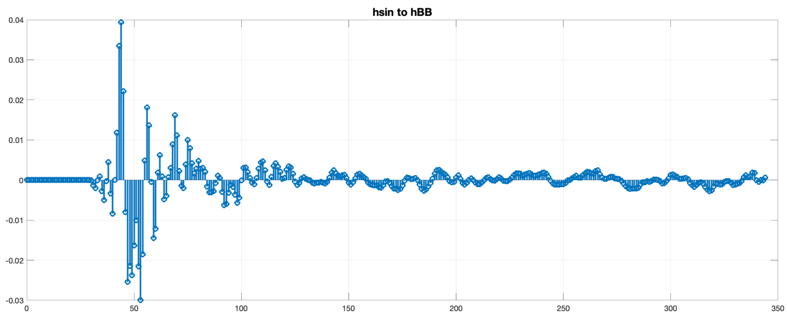
**2.**
Case I: Decreasing filter length
In case of sinusoidal noise there is no difference in error performance when we decrease filter taps from 100 to 25, however, the error increases when we take broad band noise. The reason behind this different behavior is that in order to compensate broad band noise a filter needs lot of coefficients in order to capture lot of parameters whereas small taps are enough to estimate sinusoidal disturbances. Therefore, the optimal length of the equalizer filter after looking empirical section question 3 may be between 100 to 150 in order to compensate channel effects for broad band disturbances.

Case II: Increasing filter length
When we use broad band noise there is no difference of changing the length of the filter if we reset the filter taps or not. However, filter showing same performance since LMS objective isn't unique, this is because this approach is already close to initial LMS performance.

**3.**




$h_{sin \to BB} h_{BB \to sin}$,




$h_{BB}, h_{sin}$

**Figure 10**

In both cases from $h_{BB\to sin}$and $h_{sin\to BB}$ filter responses are almost same as shown in figure 10.

In the 1st case when we trained for $h_{BB}$ and then we feed $h_{sin}$ the equalizer will also change according to the requirements because LMS tries to compensate the other frequencies components. That's why we are getting the same shape as like $h_{BB}$

In the 2nd case when we trained for $h_{sin}$ and we feed him $h_{BB}$ the filter already trained for the frequency which already present in the broad band. Therefore, we are getting the same behavior due to that reason.

**4.**

We observe from the figure (8) that $Hsin(f_0) = 0.37\angle 0.19$ and $H(f_0)_{BB} = 0.38\angle 0.17$. In reality they should have same values, but they are little bit different. The reason for that difference is that for sinusoidal the system is performing accurately because he just needs to optimize LMS objective for only one frequency whereas for broad band he has to optimize around the range of frequencies. In fact, Ysin and YBB at $f_0$ showing the same behavior but for rest of the frequencies they are completely different whatever the values of h are.

**5.**

We need to cancel a sinusoidal disturbance of the FIR filter to determine the number of filter coefficient that we need.
We can this task in frequency domain using the following minimization problem

$$\hat{H} = arg\ min\ \frac{1}{\omega} \int_0^{\omega s} |H(\omega)|^2 S_{SS}(\omega) + (|H(\omega)|^2 - |\bar{H}(\omega)|^2) S_{yy}(\omega)\ d\omega$$

$S_{SS}(\omega)$ the frequency responses for input signals and $S_{yy}(\omega)$ are the frequency responses for output signals.

Let give assumption that the channel is complex with amplitude and phase:

$$H(\omega_0) = A_0 e^{j\theta}$$

The DFT can be written as

$$A_0 e^{j\theta} = H_0(\omega_0) = \sum_{K=0}^{M-1} h(k) e^{-jw_0 \Delta t}$$

That mean one coefficient will be not enough to estimate the length of it.

$$H(\omega_0) = \hat{H}(\omega_0)$$
$$= h(0) + h(1) e^{-jw_0 \Delta t}$$

So minimum filter length $M \geq 2$, at that we will be able to cancel the sinusoidal disturbance with at 2 coefficients.

**6.**

It can be seen in figure 3 in the empirical section that the FIR filter shows a sinusoidal shape for longer coefficients. The filter updating steps can be written as follows

$$e(N) = x(n) - \hat{h}(N)y(N)$$
$$\hat{h}(N+1) = \hat{h}(N) + 2\mu y(N)e(N)$$

Looking that equation telling us that the next filter coefficient depends on the error function of the previous coefficient value with some scaling factor with different phases. And if we have a sin in the input in the 1st step it will be a scalar but next value will become like sinusoidal because now we are giving them sin. Similarly, for a particular frequency like $f_0$ we will get sum of sinusoids with that same frequency.

Appendix – LMS C Code

Listing 1: Noise cancelling LMS algorithm

```
    int n ;

      for ( n=0; n<block_size ; n++)
{
        arm_dot_prod_f32( lms_coeffs , lms_state+n, lms_taps , xhat+n ) ;
        e[n] = (x[n] - xhat[n]);
        for (int j=0; j<lms_taps; j++ )
        {
            lms_coeffs [j] +=2*lms_mu*e[n]*lms_state[n+j];


        }
}
```