Project 2 – Adaptive Filtering

Normalized Least Mean Squared Error is an incredible technique in adaptive filtering. Its most common application is that in active noise cancellation – where the filter only evaluates the error between the system output and the desired output at a given time. The nLMS filter helps work around the computationally complex direct solutions to such problems, by applying an instance based stochastic gradient descent methodology.

In order to better explore this topic, for the purpose of this project, we tried to understand the impulse response of different rooms without exposing the room to large audio signal. In order to make it more realistic, we introduced a source of white noise, as well as a secondary source of ambient noise coming from an external audio file – in my experiment, I used a song audio file cut to the appropriate sample size, to make all vectors in this project of the same size. Using the model of a quiet room and by creating an excitation signal while having an ambient source of noise as well as white noise, helps us design an adaptive filter that can help phase out the ambient noise to understand the room impulse response that we desire (RIR).

For the source of my ambient noise, I used a humorous and popular track – Gucci Gang, by artist Lil Pump. The randn() function was used to generate the excitation signal as a source of white noise. The results from experiment 1 are listed down below. I used the RIR signals of 05x30y and 60x25y. The white noise excitation signal was convolved with the first RIR, and the ambient noise was convolved with the second RIR. Both of these convolutions were summed and normalized to generate the overall signal for the room. Results from this experiment are listed in Table 1.

Noise Level (%)	SNR	N_MSE	N_ MSE (400 samples)	MSE tail
0	infinite	4.8346e-6	0.0035	3.4501e-4
10	20	4.8644e-4	0.1161	0.0493
20	13.9794	0.0022	0.9428	0.1972
30	10.4576	0.0047	2.0226	0.4325
40	7.9588	0.0066	2.6999	0.5666
50	6.0206	0.0111	8.3277	1.0294
60	4.4370	0.0213	10.3177	1.8352
70	3.0980	0.0189	11.9982	1.4624
80	1.9382	0.0316	19.0676	2.5188
90	0.9151	0.0329	11.1534	3.1149
100	0	0.0441	20.6274	3.1166

Table 1. Results of nLMS filter used for RIR estimation at various noise levels

The plots for some of the RIRs and their respective errors are displayed below. In order to show more distinct differences, you will first find the 10% noise and 40% noise level plots, followed by the 60% and 90% noise level plots. Along with this, you will also find the 0% noise and 100% noise level plots to show the stark comparison.

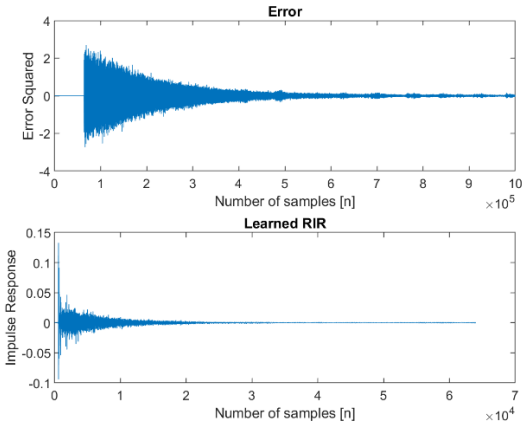

Fig 1. Learned RIR and error with 10% noise level

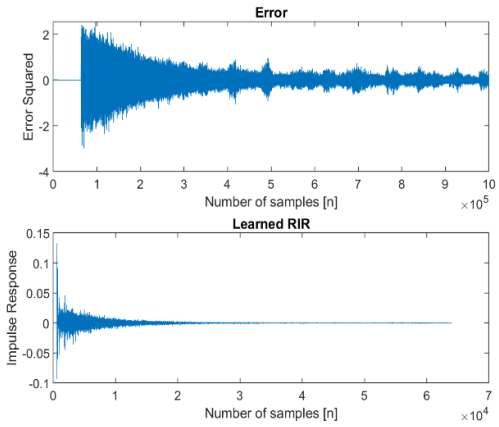

Fig 2. Learned RIR and error with 40% noise level

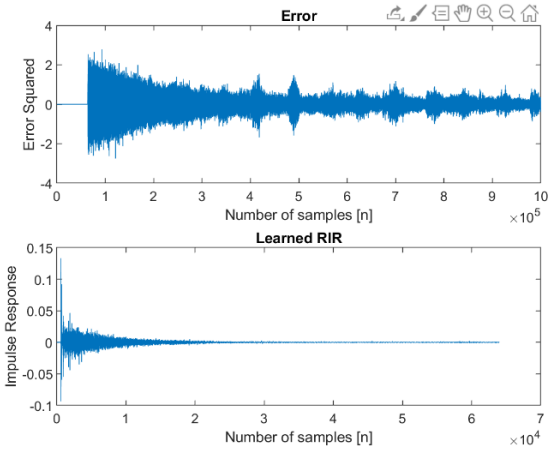

Fig 3. Learned RIR and error with 60% noise level

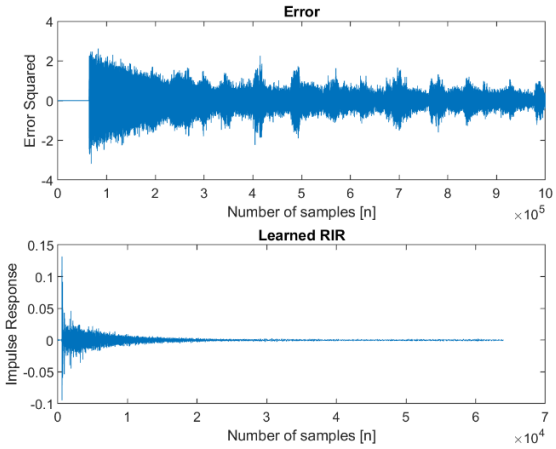

Fig 4. Learned RIR and error with 90% noise level

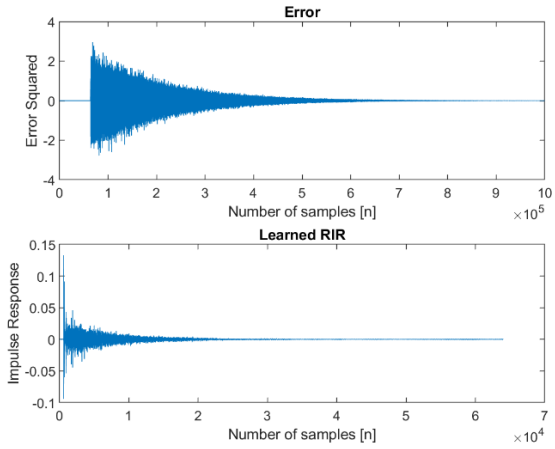

Fig 5. Learned RIR and error with 0% noise level

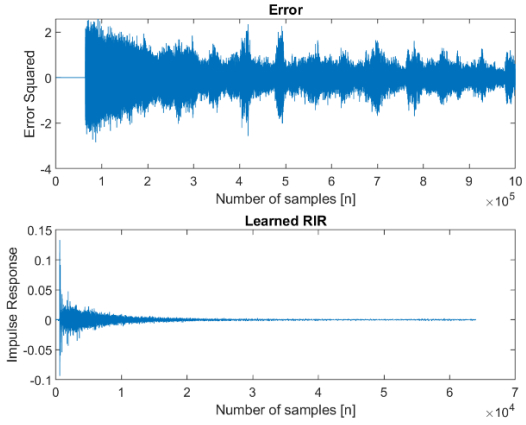

Fig 6. Learned RIR and error with 100% noise level

As we can see from Figures 1-4, as the noise level increases, so does the distortion in the error signal. The only way to truly understand and segment these plots is to look at the start level of the

noise, and also the overall converging trend for the same – the distortions show difficulty in successful adapting at those samples.

An nLMS filter was used to calculate the coefficients of the RIR. I used the step size of 0.5 for β – conventionally this is supposed to be under 2, and practically it is restricted to be between 0 and 1. The closer the step size was to 0, the more effective would be the filtration, however this would be very computationally exhaustive. Similarly, the closer the step size was to 1, the faster would be the computation, but the less effective the filtration. Hence I decided to go with a β value of 0.5. In order to improve the resolution, I experimented with sample sizes of 250,000 all the way to 3,00,00,00. There was so significant advantage in using sample sizes greater than 1,00,00,00, and hence I chose to use that as my sample size during this experiment.

We can see from Table 1, that the normalized MSE values were increasing as noise level increased, specifically due to larger oscillations at the beginning of the simulation. Similarity, the first 400 samples of the normalized MSE also showed a greater increasing trend again caused due to the larger variations in the error signal at the beginning before the error function converged – since we were picking the initial 400 samples, we chose the error region with maximum variations. The tail end of the MSE signal also showed a great increasing trend as the error signal increased with increase in noise level – we can see an increased number of distortions at the tail end of the error signal, as noise level increases.

For the second part of this experiment, we took a variety of RIR signals and ran them through the same nLMS filter we wrote for part 1.

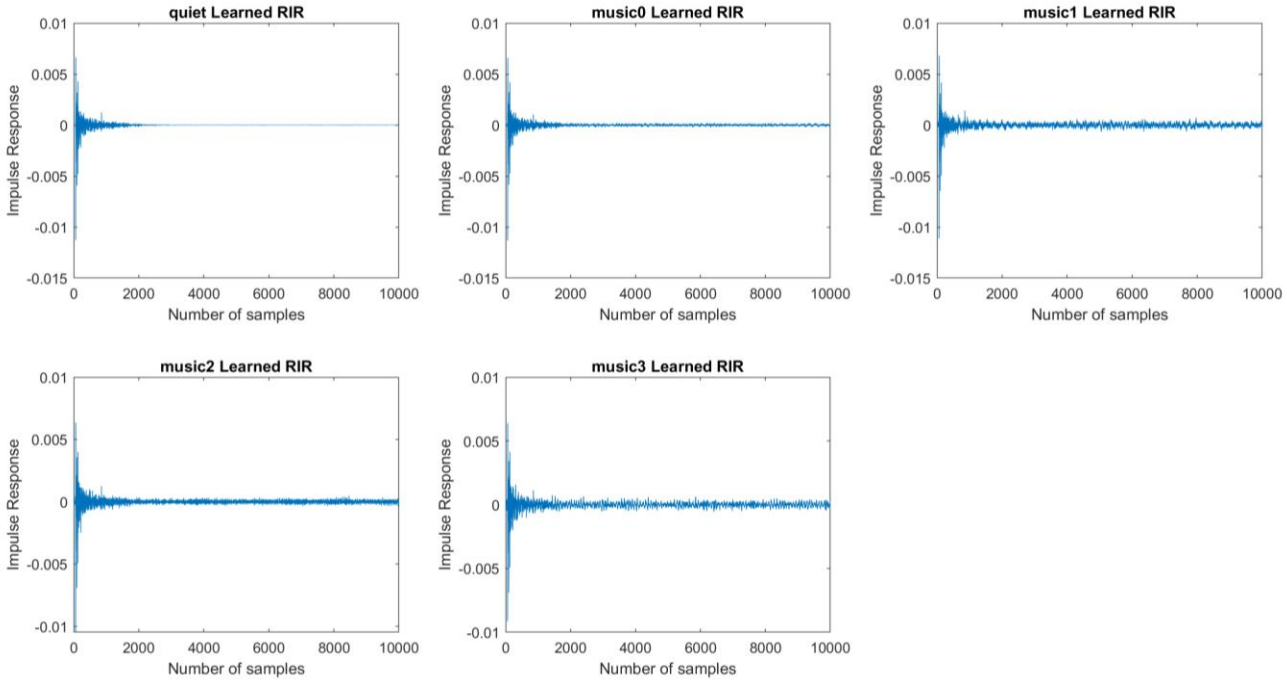

Fig 7: Learned RIR signals for different music signals in Carpet Room

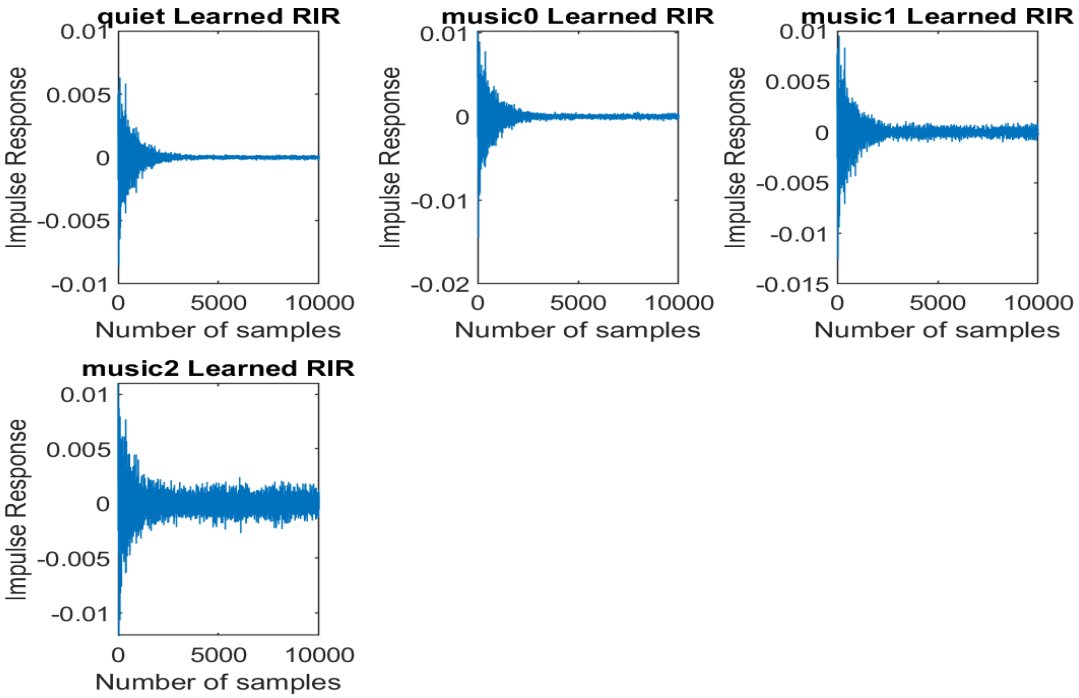

Fig 8: Learned RIR signals for different music signals in tile Room

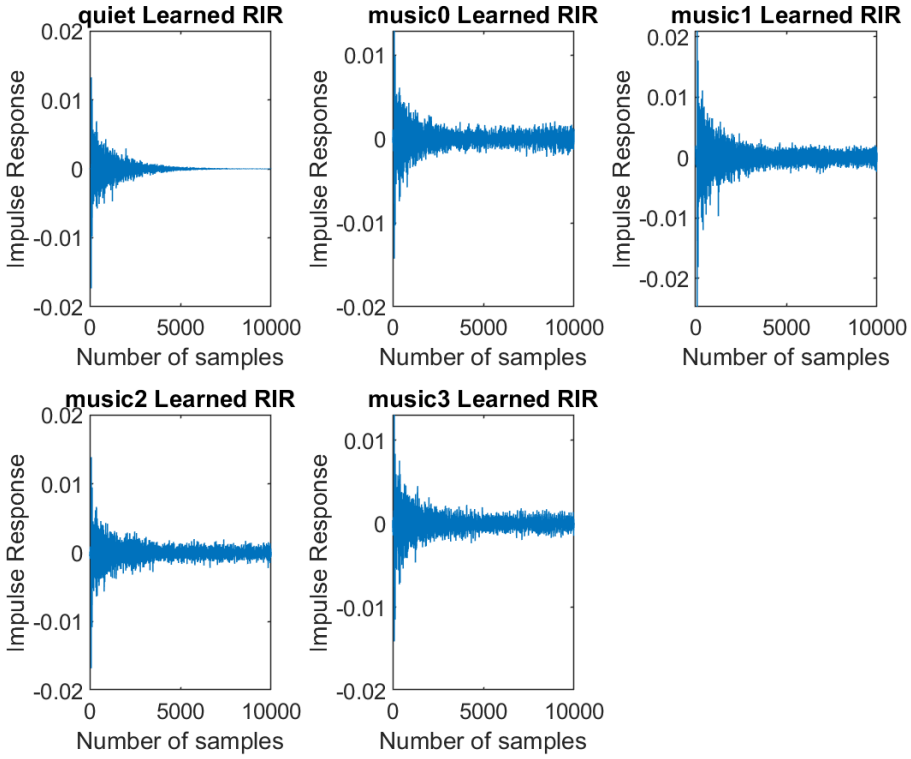

Fig 9: Learned RIR signals for different music signals in woodfloor Room

As we can see from the figures above, the carpet room performed the best dampening and the wood floor room performed the worst. This can be expected since carpets have greater sound absorption capacity when compared to wood. Since wood and tiles have similar surfaces, their RIR trends are similar, however the performance is not. The wood room retains more noise than the tile room, and we can see this from the RIR plots above.

The only change made to the nLMS filter for this part of the project was the inclusion of a delay in the input to the filter, to account for the transmission and computation delay involved in the sound reaching from the speaker of the computer to its microphone. In order to do this, I visually observed the array of data for each RIR, and noticed that the values up to around 17000 indices in the array are very close to 0. Hence I indexed the vectors from after 17000 to the end in order to avoid any of those close to 0 values. Moreover, the optimal sample size was determined to be 10,000 for better resolution.