Adaptive Filtering

Kaan Okumuş, Clémence Altmeyerhenzien
EPFL - Switzerland

***Abstract*—The purpose of this report is to present and analyse a practical implementation of adaptive filter algorithms in order to have a better understanding of denoising problems in real time. We study here the Least Mean Square (LMS) algorithm, a normalized version of LMS (NLMS), the momentum LMS algorithm, signed LMS and then more advanced tools such as the Recursive Least Square (RMS) algorithm, the Fast Block Least Square (BLS) algorithm and finally the Affine Projection (AP) algorithm. We will show concrete evidence of the differences between these algorithms through tests on a simulated signal and on real data.**

I. Introduction

This report aims to introduce different approaches to adaptive filters in a noise cancellation setting. Throughout this mini-project, we have studied and tested algorithms that try to improve to some degree standard Least Mean Square (LMS) algorithm. The algorithms presented are either more computationally efficient such as Signed LMS or Fast Block Least Square (BLS), or they improve filtering, such as Momentum LMS. Lastly, they might solve a diversity of problems that might arise, such as Normalized LMS, Recursive Least Square (RMS), Affine Projection (AP), and Discrete Cosine Transform (DCT-LMS) LMS algorithm.

As we implement these algorithms, we expect the real data to be more unpredictable than the simulated data and to give a poorer performance. We also expect our results to be implementable in real time, meaning that each iteration does not cause excessive delay, as this is the primary constraint.

This report is organized as follows: part 1 focuses on the theory behind these filters, part 2 considers our results on simulated data, and eventually, part 3 presents how the algorithms perform on real data.

II. Theory

Let us take a look at the theoretical adaptive filter.

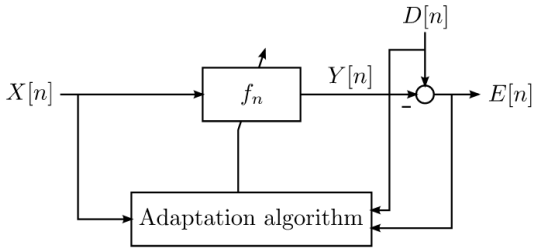

Fig. 1: Theoretical adaptive filter

At sample n, $X[n]$ is the noise that we want to cancel, f_n the adaptive filter, $Y[n]$ the correction signal, $D[n]$ the noisy signal we want to perform the noise filtering on, and $E[n]$ is the output, meaning the denoised signal.

A. Basic tools

1) LMS Algorithm: The goal is to create a filter f_n that successively adapts after each iteration using the recurrence relation

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \mathbf{X}_n E[n]$$

where $\mathbf{X}_n$ is the buffered noise (only the last K samples), and μ is the correction term, $0 < \mu < \frac{2}{\lambda_{max}}$ with λ_{max} the largest eigenvalue of $\mathbf{R}_{X,n}$ [1].

2) Normalized LMS Algorithm: This algorithm uses a normalized version of the noise and can be described by the recurrence relation

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \frac{\mathbf{X}_n}{||\mathbf{X}_n||^2} E[n]$$

This modification can improve the convergence speed of the adaptive filter at the cost of increased computational time [2].

3) Momentum LMS Algorithm: The eigenvalue spread is defined by

$$N_k = \frac{\lambda_{max}}{\lambda_{min}},\ \lambda_k = eigval(R),\ k = 0, 1, ..., N-1$$

where R is the auto-correlation matrix of $X[n]$. When this eigenvalue spread is small, standard LMS gives good performance, whereas for large values the convergence speed slows down, which is why Momentum LMS is introduced. This algorithm buffers two past versions of the filter to get the following recurrence relation for f_{n+1}

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \mathbf{X}_n E[n] + \beta(\mathbf{f}_n - \mathbf{f}_{n-1})$$

where $\beta(\mathbf{f}_n - \mathbf{f}_{n-1})$ is the momentum term which smooths the trajectory of the adaptive filter coefficients and leads to accelerated convergence when the eigenvalue spread is large [3]. However, it introduces twice as many multiplications as compared to standard LMS.

4) Signed LMS Algorithm: In order to apply LMS algorithms on hardware targets like DSP devices, FPGAss, and ASICS, this algorithm is used as a simplified version of LMS. Applying the sign function to standard LMS returns the following three types of sign LMS algorithms:

- **Sign-error LMS:** applies sign function to $E[n]$.
- **Sign-data LMS:** applies sign function to $X[n]$.
- **Sign-sign LMS:** applies sign function to both $E[n]$ and $X[n]$.

Sign-sign LMS is the best of the three to decrease computational time. When choosing the step size as a power of 2, shift operations can also be used. Signed LMS algorithms result in slower convergence speed and more significant displacement compared to standard LMS [2].

B. Advanced tools

1) RLS Algorithm: The RLS algorithm has a degree of freedom that other algorithms do not have. It takes more or less into account the previous samples thanks to the μ parameter, which can be valuable when there are sudden changes in noise, as the algorithm can adapt faster. RLS has parameters K, the number of filter taps, ϵ, $0 \ll \mu < 1$, which corresponds to the forgetting factor (the smaller it is, the less information from previous samples the algorithm uses). According to literature, usually $0.98 < \mu < 1$ and $0 < \varepsilon \ll 1$.

$$\mathbf{f}_n = 0, \mathbf{\Omega}_0 = \varepsilon^{-1}\mathbf{I}_K$$
$$\mathbf{g}_n = \frac{\mathbf{\Omega}_n \mathbf{x}_n}{\mu + \mathbf{x}_n^T \mathbf{\Omega}_n \mathbf{x}_n}$$
$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mathbf{g}_n e[n]$$
$$\mathbf{\Omega}_{n+1} = \mu^{-1}(\mathbf{\Omega}_n - \mathbf{g}_n \mathbf{z}_n^T)$$

2) Fast Block LMS Algorithm: In an acoustic echo cancelation problem, the adaptive filter may have an extended impulse response to cope with an equally long echo duration. In the case of the LMS algorithms, we adapted the filter coefficients in the time domain, and the computational complexity is considerable for this kind of application. Two complementary methods are applied to solve the problem:

- Block implementation of FIR filter, which allows efficient use of parallel processing and thereby results in an increased computational speed
- FFT algorithm for performing fast convolutions to permit adaptation of filter parameters in the frequency domain

The flowchart of the algorithm is present in Figure 2.

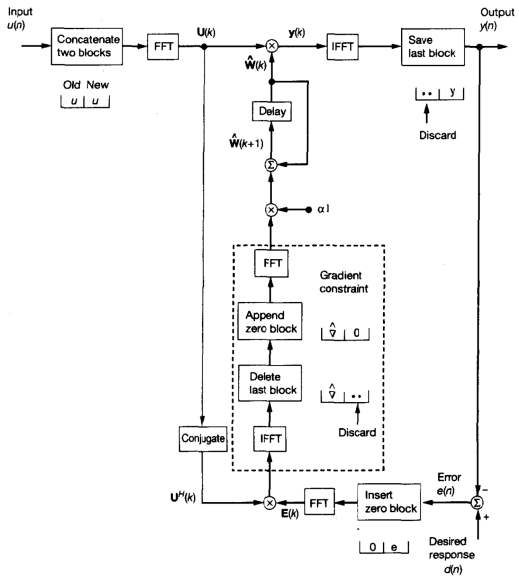

Fig. 2: Fast block LMS algorithm

3) Affine Projection Algorithm: Affine projection algoritm is an extension of LMS using multiple input vectors at each iteration.

$$\mathbf{X}_n = (\mathbf{x}_n, ..., \mathbf{x}_{n-L})$$

The number of input vectors is called *projection order* and denoted as L. The equation for the filter is as follows

$$\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \mathbf{I}_{L,n} \mathbf{X}_n^T (\epsilon \mathbf{I} + \mathbf{X}_n \mathbf{X}^T)^{-1} \mathbf{E}[n]$$

AP is designed to be effective for highly correlated data and improve the convergence properties of adaptive filters linked to the projection order. As the order increases, convergence speed should increase, but more significant computational cost results.

III. Materials and Methods

We tested the algorithms on simulated data and real data. The simulated data is a sum of two sinusoids at frequencies 3 and 5 Hz, with sampling frequency 44100Hz, played between 1 and 3 seconds and on which $(h * x)$ has been added, with x a random Gaussian noise (RGN) with standard deviation $\sigma = 1$ and h of size 5, drawn at random.

On the advanced tools, we performed additional testing with the same sinusoid but with a different noise: to see how well the algorithms would react to a sudden change in noise, we added a RGN with standard deviations $\sigma = 5$ to the first RGN that had $\sigma = 1$. Then we convoluted with h.

We chose to judge our results based on convergence time and misplacement. Indeed, we would like our filter to adapt quickly to hear noise as briefly as possible, which we represent through convergence time but not to the detriment of sound quality, which we measure using misplacement.

- Convergence time is not necessarily "time" so to speak, but rather the first sample in which $E[n]$ is deemed sufficiently close to $S[n]$ according to a threshold ϵ. We measured this by taking the Mean Square Error (MSE) of $E[n]$ and $S[n]$, and then taking the first n_0 for which $\forall n > n_0$, $MSE(E(n), S(n)) < \epsilon = 0.1$. While this ϵ is arbitrary, it ensures the noise is low enough to hear the sinusoid clearly.
- Misplacement is measured as the absolute value of the difference between the filtered signal and the original signal, meaning $|S[n] - E[n]|$ after convergence.

IV. Results on simulated data

A. Basic tools

1) LMS: From Figure 3., we plotted the correct adaptation of the standard LMS on simulated data for the optimal $\mu = 2e - 3$. The convergence point was at 0.05 sec, and misadjustment was at 0.13 (Figure 8).

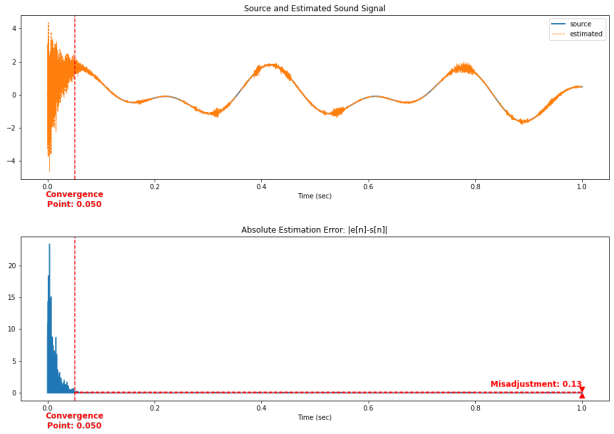

Fig. 3: Source and estimated signals and absolute estimation error signal of standard LMS with optimal $\mu = 2e-3$.

2) Normalized LMS: Normalized LMS yielded a faster convergence speed than the standard LMS algorithm for the simulated data. However, we observed a slight increase in misadjustment. The convergence point was at 0.043 sec, and misadjustment was at 0.28 (Figure 9).

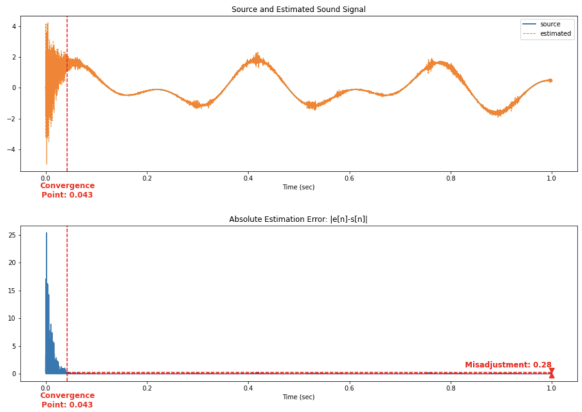

Fig. 4: Source and estimated signals and absolute estimation error signal of normalized LMS with optimal $\mu = 1e-2$.

3) Momentum LMS: For small eigenvalue spread noise signal $X[n]$, we observed that there is almost no change in convergence rate, while small increase in misadjustment observed in momentum LMS when compared to the standard LMS. From Figure 10, convergence point was at 0.046 and misadjustment was calculated as 0.28.

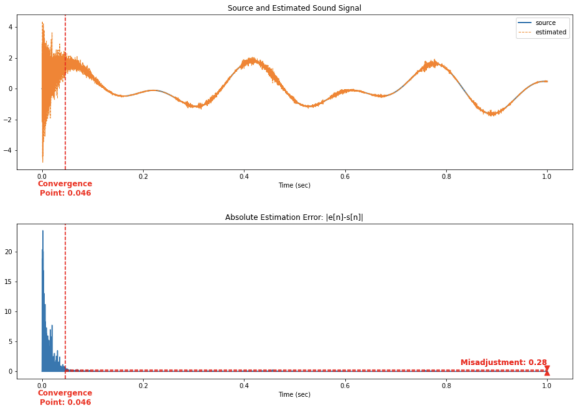

Fig. 5: Source and estimated signals and absolute estimation error signal of momentum LMS with optimal $\mu = 2e-3$ and $\beta = 1e-2$.

4) Signed LMS: We only implemented sign-sign LMS, where we observed a significant increase in convergence point and misadjustment (Figure 11). The convergence point was at 0.117 sec, and misadjustment at 1.34. we expected this result as sign-sign LMS is a distorted version of standard LMS. By contrast, we saw a minimal improvement in the computational time of the algorithm during the experiment for the simulated data.

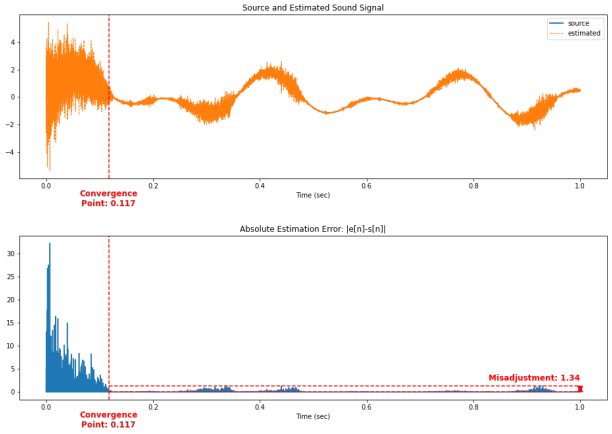

Fig. 6: Source and estimated signals and absolute estimation error signal of sign-sign LMS with optimal $\mu = 2e-3$.

B. Advanced tools

1) RLS algorithm: We measured convergence and misadjustment for different K values and parameters $\mu = 0.999$ and $\varepsilon = 0.0001$. We observed that K had a significant impact on convergence, and we concluded that having a larger K than five made performances worst for this context. As for μ, we observed that values very close to 1 guaranteed low misadjustment. However, when μ went to 0.98, the convergence got faster.

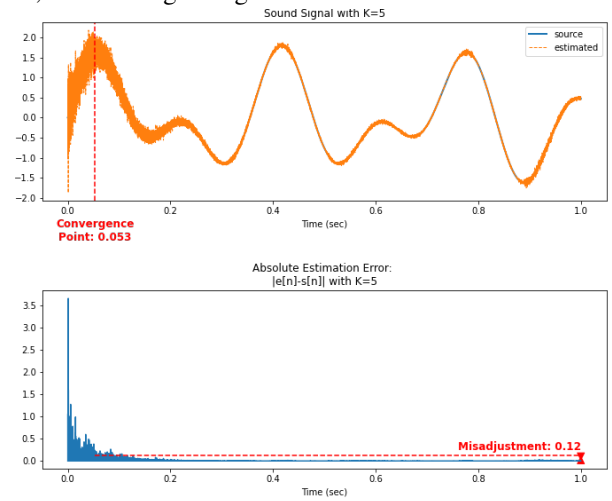

Fig. 7: Convergence and Misadjustment measures on RLS

2) Fast block LMS algorithm: We saw that a much bigger value of K was needed for the filter to converge for this algorithm. The filtering performance that this algorithm presented was significantly worst than other algorithms. However, the computation time was quick! We saw that despite the average convergence time (around 0.25s for any value of K), the misadjustment was higher than any other algorithm (values ranging from 5 to 20) and the filtered signal was still noisy (Figure 8). We used the parameters K=2500 and $\mu = 0.0001$, which gave us the best convergence and misadjustment.

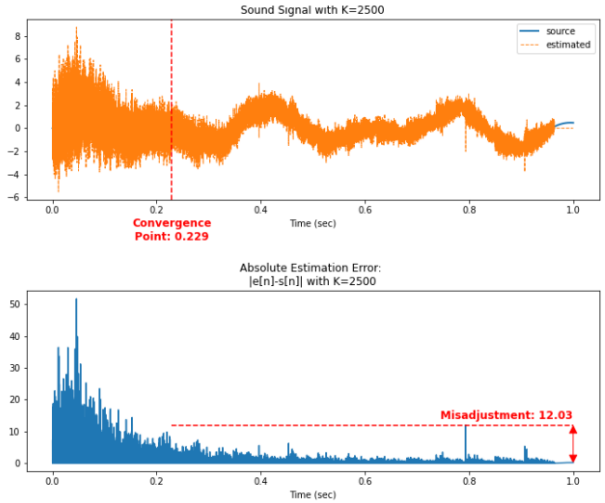

Fig. 8: Convergence and Misadjustment measures on BLS

3) Affine Projection Algorithm: We tested this algorithm with a projection order of 3. For larger projection order, it gave better results, but computational time was considerable. For optimal projection order, we saw that it outperformed the convergence properties of the fast block LMS algorithm, and it was even better than standard LMS performances. The computational time of AP was not better than standard LMS. However, it was better than RLS.

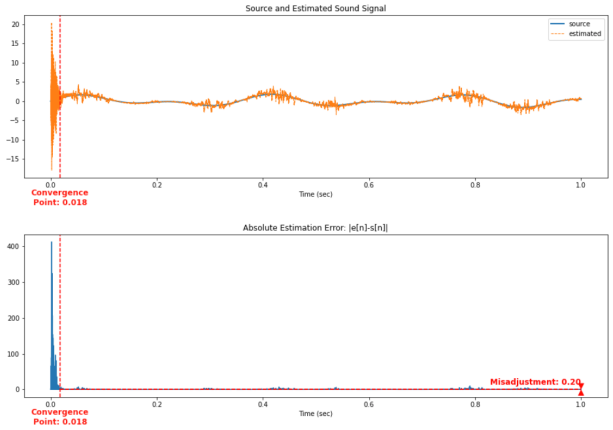

Fig. 9: Convergence and Misadjustment measures on AP for added noise with $K = 1000$ and $\mu = 1 \times 10^{-5}$ of first test.

C. Additional tests for advanced tools

1) RLS algorithm: The second test clearly showed the advantage of RLS over LMS in the shape of the filtered signal and convergence time for a similar misadjustment. Note that we chose the parameters for both LMS and RLS each time to obtain the best outcome in terms of misadjustment, but with the same $K = 10$ (to have great performances). For RLS, we chose $\mu = 0.98$ and for LMS $\mu = 0.001$. When convergence time is the same for RLS and LMS (0.023s), the misadjustment of RLS is much better (0.01 against 0.9).

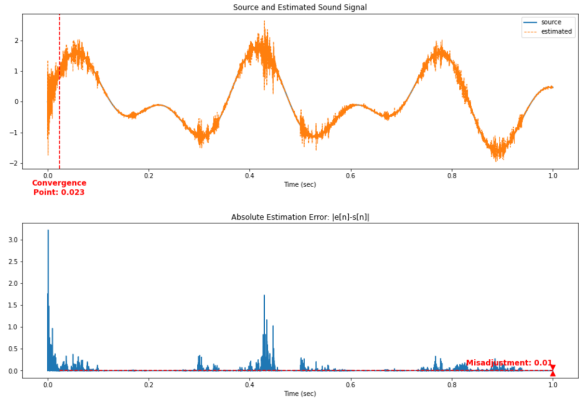

Fig. 10: Convergence and Misadjustment measures on RLS for added noise

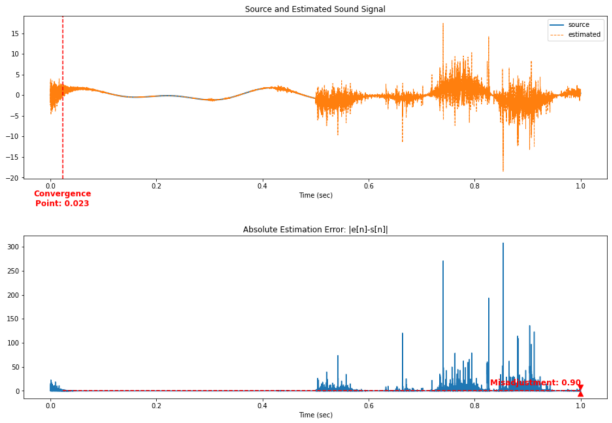

Fig. 11: Convergence and Misadjustment measures on LMS for added noise with $\mu = 0.001$

2) Fast block LMS algorithm: Once again, computational time was improved as filtering decreased for this algorithm.

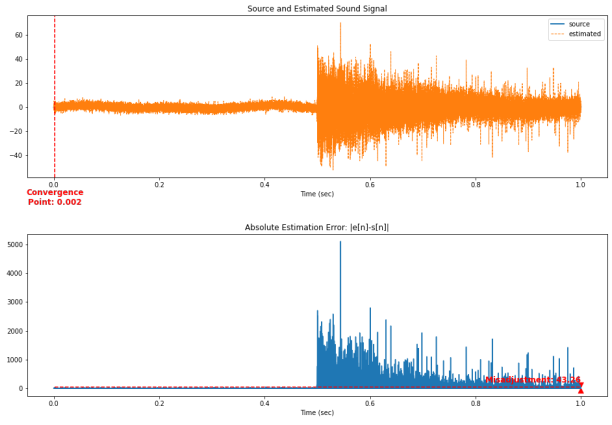

Fig. 12: Convergence and Misadjustment measures on BLS for added noise with $K = 1000$ and $\mu = 1 \times 10^{-5}$ of second test.

3) Affine Projection Algorithm: This second test showed that AP was not as robust to abrupt noise changes as RLS. The computational time of AP was not better than standard LMS. However, it was better than RLS.

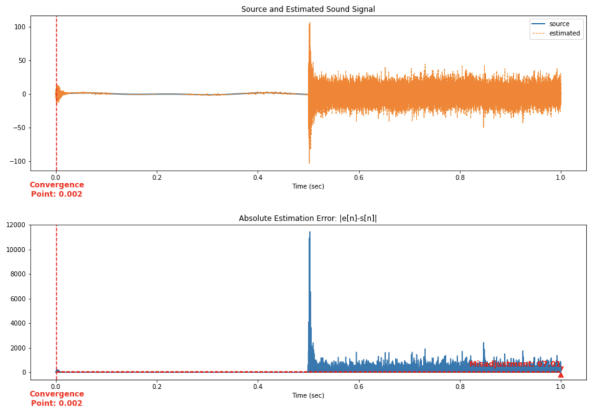

Fig. 13: Convergence and Misadjustment measures on AP for added noise with $K = 1000$ and $\mu = 1 \times 10^{-5}$ of second test.

V. Results on real data

A. Basic tools

For the real data, we obtained $0 < \mu < \mu_{max} = 2.339 \times 10^{-5}$ which was given by "TalkingNoise.wav". We then performed the algorithms on this data to obtain the following results, plotted in the frequency domain to see the difference in noise. Indeed, the bass line that we want to hear is between 20Hz and 20 000 Hz, which is the human ear frequency range (we are choosing a very large bound because basses will be between 20Hz and 1000Hz in reality). For visualization purposes, we selected the most interesting range of frequencies.

1) LMS: We saw that some noise was canceled (the blue part is much higher than the orange one at around 2000Hz) while most low frequencies were kept.

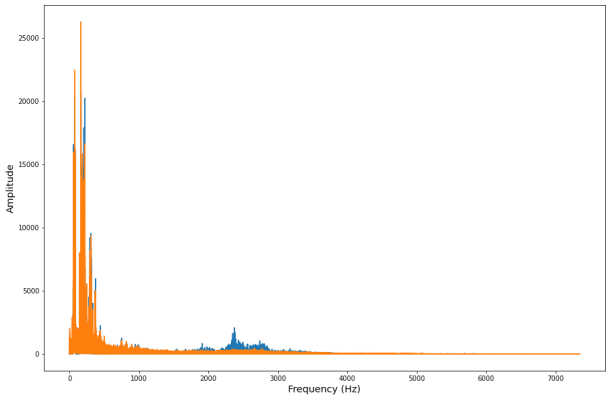

Fig. 14: Fast Fourier Transform of (in blue:) the noisy data and (in orange:) the filtered data under the standard LMS algorithm

2) Normalized LMS: Again, a similar amount of noise compared to the LMS was canceled while most of the low frequencies were kept. However, it seemed to be less efficient for this signal since frequencies superior to 800Hz were kept when we would want them to be canceled.

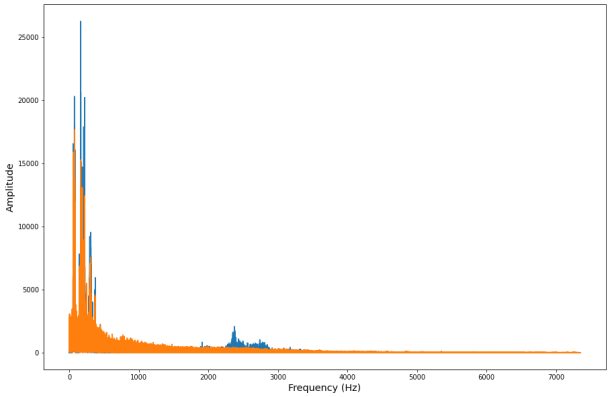

Fig. 15: Fast Fourier Transform of (in blue:) the noisy data and (in orange:) the filtered data under the normalized LMS algorithm

3) Momentum LMS: The Momentum Algorithm gave the inverse of the Normalized LMS, as most frequencies superior to 500Hz were canceled more harshly, which is an excellent fit for our bass. However, there was more average noise than the previous algorithms.

Since the Signed Algorithms were designed to perform more efficiently, it is with little surprise that we saw them cancel overall less noise, and it even appeared to be creating some.

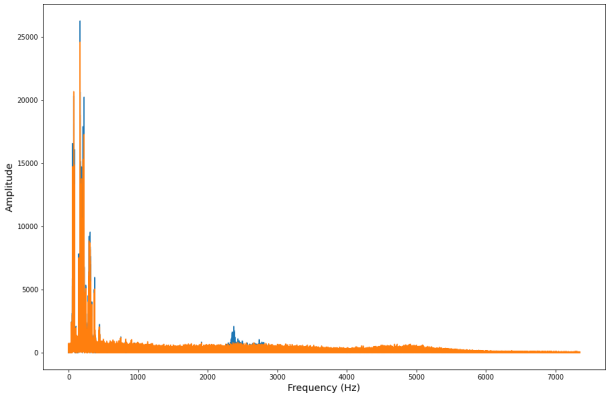

Fig. 16: Fast Fourier Transform of (in blue:) the noisy data and (in orange:) the filtered data under the Momentum LMS algorithm

4) Signed LMS: Sign-sign appeared to perform better than other signed algorithms, so we decided only to plot this one. It had less noise on average, whereas Sign-Data and Sign-Error introduced much noise.

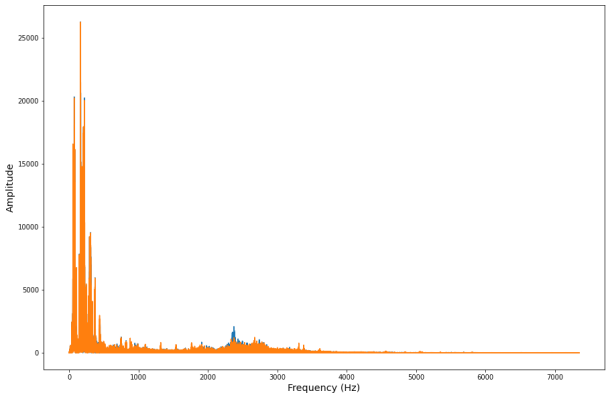

Fig. 17: Fast Fourier Transform of (in blue:) the noisy data and (in orange:) the filtered data under the Sign-Sign LMS algorithm

B. Advanced tools

Parameters		
Algorithm	K	μ
LMS	10	0.1
RLS	10	0.98
BLS	3000	0.0001
DCT	50	0.00006
Affine	20	0.08

1) RLS algorithm: We were able to see that RLS was better at attenuating higher frequencies than LMS (second picture below) but kept some noise near the frequencies of the bass line (third picture).

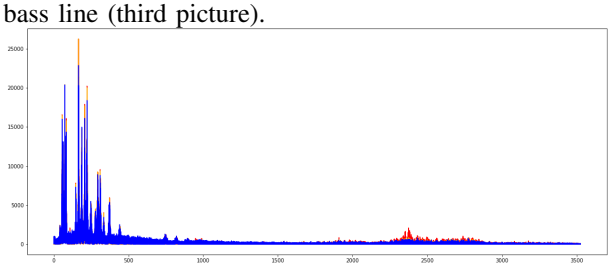

Fig. 18: FFT of the talking noise (red), the LMS-filtered signal (yellow), the RLS-filtered signal (blue) as a function of the sample number

2) Fast block LMS algorithm: We observed that once again, BLS conducted poorer than the other algorithms in higher frequencies where it introduces more noise (in pink on the second picture) than the received signal. Surprisingly, it performed similarly to LMS in lower frequencies. The advantage of this algorithm was verified as it is faster than any other algorithm that we tested.

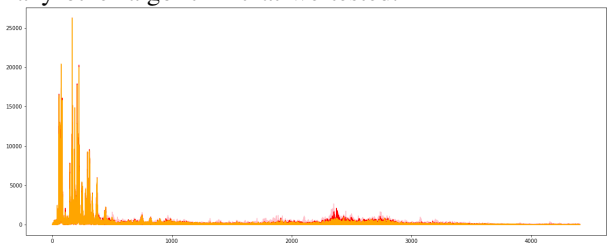

Fig. 19: FFT of the talking noise (red), the LMS-filtered signal (yellow), the BLS-filtered signal (pink) as a function of the sample number

3) Affine Projection algorithm: For the Affine algorithm, not only did the computation time perform poorly, but the filtering was also flawed! The only highlight of this algorithm was that it surprisingly gave better results in the low-frequency range (third picture below of Figure 10).

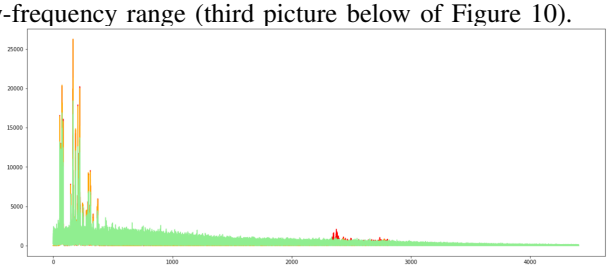

Fig. 20: FFT of the talking noise (red), the LMS-filtered signal (yellow), the Affine-filtered signal (green) as a function of the sample number

VI. Discussion

Normalized LMS speeded the algorithm, but sometimes negligible increases in misadjustment were observed. However, the increase in convergence speed was small for our simulated data . Still, in general, it can be said that to ensure that the step size is appropriately chosen, the normalized LMS is better than the standard LMS.

When in the auto-correlation matrix of $X[n]$, eigenvalues are not spread widely, the improvement in convergence speed is minimal in momentum LMS compared to the standard LMS. However, we also observed a tiny increase in misadjustment. Also, the computational power is more considerable for momentum LMS. In order to see its improvement in terms of convergence speed, a larger eigenvalue spread must be used. To sum up, momentum LMS is better than the standard LMS only when a large eigenvalue spread is observed.

Signed LMS algorithms are simplified versions of standard LMS. Much slower convergence rates and much larger misadjustments are observed, which makes it undesirable. This algorithm can only be useful for a system with very limited processing power because its computational time is much less than the other algorithms. However, this improvement is negligibly small in its implementation on standard computers.

RLS had very robust convergence properties. However, it had much more computational time, which we think makes it not applicable for real-time implementation. On the other side, it gave the best convergence properties among all algorithms for offline applications. The second test that we conducted showed the strength of RLS when dealing with changes in noise. We explain this outcome by a change in the parameter μ, which "forgets" the previous samples. In the case of a fast shift in noise, we saw that it dealt with changes much faster than LMS: we could barely see a peak in the middle of the signal. We saw that fast block LMS performed much faster than standard LMS using FFT implementation. Thus, it seemed to be suitable for real-world data. However, it might be inadequate for applications that need high-quality data as it gives much worse convergence properties, particularly with larger misadjustment.

Fast Block LMS is incredibly fast to the cost of convergence and misadjustment. While it is applicable in real-time, it might not cancel noise at all, and it even seemed to introduce supplementary noise.

In order to achieve both low computational cost and good convergence properties, we introduced affine projection. This algorithm outperformed standard LMS in terms of convergence but was not as efficient in reducing the noise as RLS. Computational time is not as large as RLS, making it suitable for real-world applications. However, on real data, its computation time was very poor.

VII. Conclusion

To conclude, we have reviewed the adaptive filtering technology in the case of noise cancellation, implemented it on simulated and real data, and given precise details on how one can successfully re-create these experiences. While the theory is quite simple, many different algorithms can be derived from the basic Least Mean Square algorithm, for example, to make this approach more time-efficient (which is crucial for a real-time implementation). Precise details were given on how one can successfully re-create these experiences.

Different algorithms were therefore presented, each with their characteristics and performances. While we could see some algorithms perform better on real data, it is essential to keep in mind that each algorithm performs differently on different signals.

We noted the tradeoff between convergence time and misadjustment through this report and determined which parameters we could use. We observed how different the parameters were, depending on the algorithms (especially the buffer size).

While these algorithms kept their promises, we were still disappointed by the general results on real data. An exploration of other filters, maybe non-adaptive such as the Kalman filter, might be a new track for a future report.

REFERENCES

[1] CLARKSON, P. *Optimal and Adaptive Signal Processing*, 1st ed. ed. CRC Press, 1993.

[2] NI. Least mean squares (lms) algorithms (digital filter design toolkit), 2014.

[3] TANRIKULU, O. *Adaptive Signal Processing Algorithms with Accelerated Convergence and Noise Immunity*. PhD thesis, University of London and Imperial College London, 1995.