Adaptive filtering applied to Acoustic Echo Cancellation

A comparison between LMS, NLMS, APA and RLS

MIAS Project Report

Rosilde Tatiana Irene
Davide Fiocchi

Index

- Theoretical background
  - LMS
  - NLMS
  - APA
  - RLS
- Experiments
  - Impulse response description
  - Parameter description
  - Stage one: manipulated impulse response
    - Single impulse response
    - Multiple impulse response
  - Stage two: real impulse response
    - Single impulse response
    - Multiple impulse response
- Final remarks
  - NLMS and APA
  - RLS adaptation
  - Algorithm comparison

Theoretical Background

The problem of acoustic echo arises in various situations in the telecommunications networks such as hands-free telephony and teleconferencing systems. Generally, it is verified when the signal emitted by the loudspeaker is propagated into the environment (that works as a filter) and returns in input to the telecommunication system through the microphone, as shown in the following scheme.

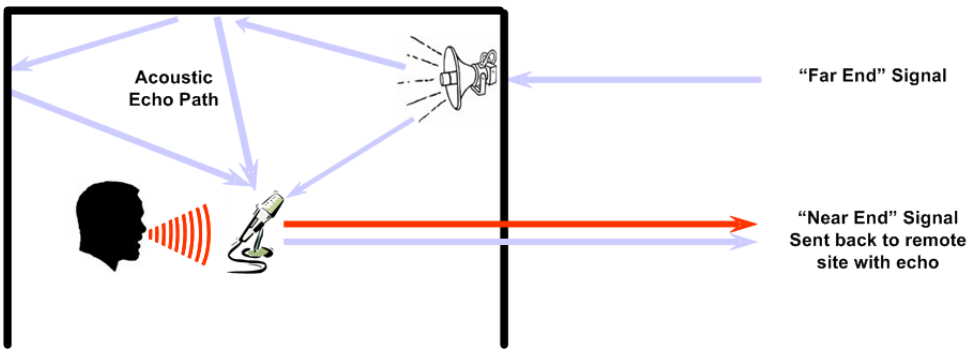


Figure a: Taken from https://files.mtstatic.com/site_8490/208/0?Expires=1501062302&Signature=f9Xjp04-rI4WsE70uEExgly64xmRbk8ZY~iV1vSLFBuLmbuJdzAjSlXn4~7kNs5na9mj2x3DCRLouEbGdzRhv3yrXBgq67YgpsZPpO-osOaF-oAplmwqAk2N~dRkeP66lIXukakuEuaTuryQQDpXKHD4HUm~voUy3uF0~PYOX

This phenomenon is one of the major issues for the speech intelligibility but also for the stability of the system, given the resultant audio feedback to which the system is exposed.

The proposed solution is the Acoustic Echo Cancellation through the adaptive filtering in which:

- The enclosure where the communication occurs is modelled as a filter with impulse response $h(t)$ that processes the input $x(t)$ obtaining the echo signal $y(t)$.
- An adaptive algorithm is used to compute an estimate of the environment impulse response $\hat{h}(t)$ that, filtering itself the input x(t), computes the estimate for the echo signal $\hat{y}(t)$.
- $\hat{y}(t)$ is subtracted to the speech signal $d(t)$ (the sum of actual speech and echo), and if the estimate is optimal the resulting signal is "clean" from the echo component.

The previous sequence is schematized in the following figure.

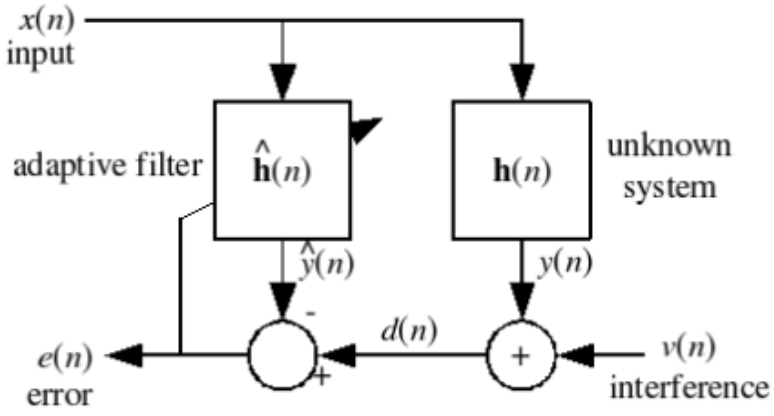


Figure b: Taken from: http://utah.technologypublisher.com/files/sites/lms_filter.png

The aim if the subsequent paragraphs is to describe the different adaptive algorithms that can be implemented for the filter estimate.

Least Mean Square Algorithm (LMS)

In general, an adaptive filtering system can be represented as:

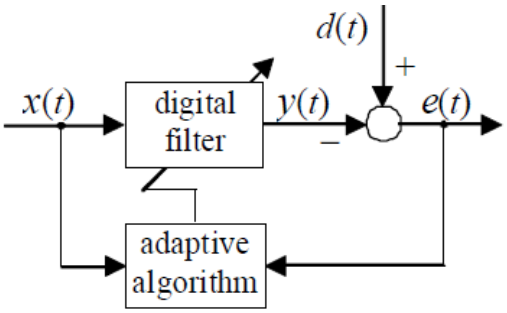


Figure c: Taken from MIAS course slides

As can be seen from this block scheme, it is composed by two elements:

- A digital filter whose output is subtracted from the desired signal $d(t)$
- An adaptive algorithm that adjusts the weights of the filter in an adaptive fashion

The digital filter is chosen to be structured as a FIR filter given the guaranteed stability property of the latter: all the poles of the relative transfer function are positioned inside the unit circle, at the origin.

The index of performance of the adaptive filter is the **Mean Square Error (MSE)** defined as:

$$J(t) = E[e^2(t)] = E\left[\left(d(t) - w(t)^T \cdot x(t)\right)^2\right] = E[d^2(t)] - 2p^T \cdot w(t) + w(t)^T \cdot R \cdot w(t)$$

Where $p = E[d(t) \cdot x(t)]$ is the cross-correlation vector between $d(t)$ and $x(t)$, whereas $R = E[x(t) \cdot x(t)^T]$ is the auto-correlation matrix of the input signal, that is semi-definite positive by construction. If $R > 0$, $J(t)$ is a quadratic function of the weights vector $w(t)$. It follows that $J(t)$ has only one minimum J_{MIN} at the corresponding optimum weight vector w^{opt}, given by:

$$\frac{dJ(t)}{dw} = -2p^T + 2\,w^T R = 0 \;\rightarrow w^{opt} = R^{-1} \cdot p$$

The latter is the **Wiener solution**, and can be found iteratively using the **Newton's Method**, in which the update equation is defined as:

$$w(t+1) = w(t) - \left(\frac{\partial^2 J(t)}{\partial w(t)^2}\right)^{-1} \cdot \frac{\partial J(t)}{\partial w(t)}$$

Where

- $\frac{\partial J(t)}{\partial w(t)} = -2p^T + 2\,w^T R$ is the gradient
- $\frac{\partial^2 J(t)}{\partial w(t)^2} = 2R$ is the Hessian matrix

This method relies on the knowledge of the autocorrelation matrix and is computationally expensive because needs the inversion of the matrix at every iteration.
A simplification of the Newton Method is the **Steepest Descent Method** in which the Hessian matrix is approximated with a positive constant, resulting in the following update rule:

$$w(t+1) = w(t) - \frac{\mu}{2} \cdot \frac{\partial J(t)}{\partial w(t)} = w(t) + \mu \cdot p^T + \mu \cdot w^T \cdot R$$

The major issue connected to these methods is that the statistics of the input signal and the desired signal are not usually known, in other words we don't have p and R, and an approximation of the gradient is needed.

It is possible to consider as a performance index only the square of the instantaneous error:

$$J(t) = e(t)^2$$

From this definition follows the approximation for the gradient, known as **stochastic gradient**:

$$\frac{\partial \hat{J}(t)}{\partial w(t)} = -2x(t) \cdot e(t)$$

Replacing the gradient with its stochastic approximation in the update rule of the Steepest Descent method one obtains:

$$w(t+1) = w(t) + \mu \cdot x(t) \cdot e(t)$$

which is the update rule of the **Least Mean Square Algorithm.**

Properties of the LMS Algorithm

The stability and the convergence speed of the LMS depends on the learning rate μ (also known as "step-size"), in particular for the algorithm to converge at the optimal solution w^{opt} the step size is constrained to the interval $\left(0, {}^{2}/_{\lambda_{MAX}}\right)$, where λ_{MAX} is the greatest eigenvalue of the autocorrelation matrix R.

The overall convergence time is determined by

$$\tau_{MAX} \approx \frac{T_s}{\mu \cdot \lambda_{MIN}}$$

where T_s is the sampling time.

Moreover, we have to take in consideration the Excess MSE, defined as the resultant oscillation of the parameters estimate once the convergence is achieved, that is calculated as:

$$J_{excess} \approx \frac{\mu}{2} L \cdot P_x \cdot J_{MIN}$$

where L is the adaptive filter length and P_x is the power of signal P_x.

From this definition, some remarks on the choice of the step size values can be made:

- A smaller step size, in the face of a slow convergence, provides a small excess MSE. For these properties, a small step size is useful for stationary signals, for which is preferred to achieve a better steady state performance rather than the real-time tracking properties of the algorithm
- A greater step size provides an elevate speed of convergence, but also a higher excess MSE. That is why it is preferable for nonstationary signals, where the convergence rate and the tracking are the most important properties.
- As an alternative, the step size can be made adaptive: can have an initial high value to achieve a faster convergence and then can be decreased to reduce the oscillation after convergence.

However, the performance after convergence depends also on the filter length, to which J_{excess} is directly proportional, in fact:

- A longer filter allows a better modelling of the reverberant environment that processes and in which the echo signal is propagated, but generates also a greater excess MSE
- A shorter filter provides a better performance at convergence (smaller excess MSE) in front of a poorer representation of the propagation properties of the environment.

Normalized Least Mean Square (NLMS) Algorithm

Generally, the greatest eigenvalue of the autocorrelation matrix R is approximated as follows:

$$\lambda_{MAX} < \sum_{l=0}^{L} \lambda_l = tr\,(R) = L \cdot\, r_{xx}(0) = L \cdot P_x$$

So that the stability constraint of the LMS becomes:

$$0 < \mu < \frac{2}{L \cdot P_x}$$

Based on this last observation one can derive a normalization of the step size in order to reduce the convergence time and the excess MSE, while maintaining the steady state performances of the algorithm. This normalization is the base for the **Normalized Least Mean Square algorithm (NLMS)**, whose update rule is the same of the LMS:

$$w(t+1) = w(t) + \mu(t) \cdot x(t) \cdot e(t)$$

But in this case the step size is made adaptive such as:

$$\mu(t) = \frac{\alpha}{L \cdot \widehat{P_x}(t)}$$

where α $(0 < \alpha < 2)$ is a normalized step size and $\widehat{P_x}(t)$ is an estimate at time t of the power of the input signal.

The power of $x(t)$ can be estimated with a rectangular moving window of length M, such that if

$M = L$ the update equation of the NLMS becomes:

$$w(t+1) = w(t) + \alpha \frac{x(t) \cdot e(t)}{x(t) \cdot x(t)^T + \delta}$$

Where δ is a regularization factor useful to prevent the denominator from going to zero.

Affine Projection Algorithm

Instead of relying only on the most recent error sample $e(t)$, as LMS and its normalized variant do, it is possible to update the filter coefficient taking in account of a vector of past error sample of length P, providing improved convergence properties. From this idea is built the update rule of the **Affine Projection Algorithm (APA)**.

Let

- $\boldsymbol{d}(t) = [d(t), d(t-1), \dots, d(t-P+1)]^T$ vector of the last P samples of $d(t)$
- $\boldsymbol{e}(t) = [e(t), e(t-1), \dots, e(t-P+1)]^T$ vector of the last P samples of $e(t)$
- $\boldsymbol{X}(t) = [\boldsymbol{x}(t), \boldsymbol{x}(t-1), \dots, \boldsymbol{x}(t-P+1)]^T$ LxP matrix of the last P input vectors

The update rule of the APA is defined as:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(\boldsymbol{t}) + \mu \cdot \boldsymbol{X}(t) \cdot [\boldsymbol{X}(t) \cdot \boldsymbol{X}(t)^T + \delta \cdot \boldsymbol{I}]^{-1} \cdot \boldsymbol{e}(t)$$

The advantages of the APA are better explained with a geometrical interpretation. Is importance to notice that if $P = 1$ the APA is equivalent to NLMS. The latter can be interpreted as a projection of the filter weights $\boldsymbol{w}(t)$ towards an affinity that contains w^{opt} and is orthogonal to the input vector, such as depicted in the following figure.

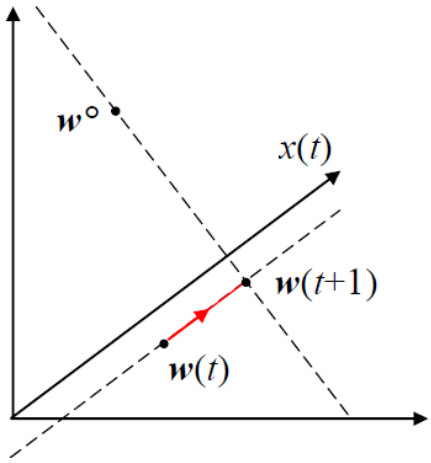


Figure d:Taken from MIAS course slides

The update is repeated until convergence, and the convergence rate will be higher if the directions of the input vectors are uncorrelated.

On the other hand, the APA the filter vectors are projected upon an affinity that is orthogonal not only to the most recent input vector, but to the last P ones,as shown in the following plot, so that even if their directions are correlated the filter weigths are effectively corrected.

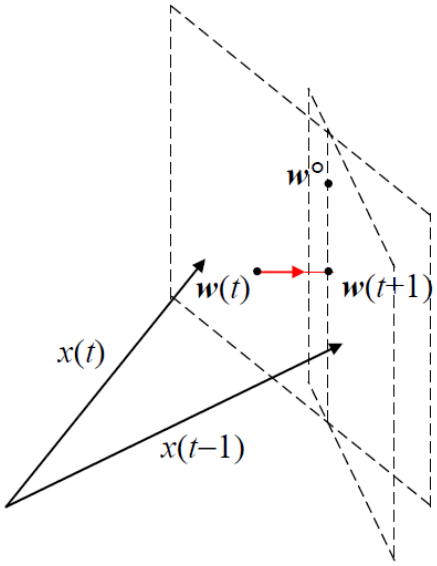


Figure e: Taken from MIAS course slides

That is the reason why the speed of convergence of the APA not only increases with μ, but also increasing P.

Recursive Least Square (RLS) Algorithm

The **Recursive Least Square (RLS)** is the recursive implementation of the batch LS identification method and its update is defined as follows:

RLS-1st form

- Basic recursion:

$$w(t) = w(t-1) + K(t) \cdot e(t)$$

- Where: $K(t)$ is the gain matrix computed as the product of the inverse of $S(t) = \sum_{i=1}^{t} \phi(i) \cdot \phi(i)^T$ and $\phi(t) = [y(t-1), \dots, y(t-n_a), u(t-1), \dots, u(t-n_b)]^T$
- While: $e(t) = y(t) - \phi(t)^T \cdot w(t-1)$ is the error generated from the estimate at the previous time instant.
- Auxiliary recursion (to compute the matrix $S(t)$):

$$S(t) = S(t-1) + \phi(t) \cdot \phi(t)^T$$

Being the matrix $S(t)$ divergent in stationary conditions for $t \to \infty$ it can be $R(t) = \frac{S(t)}{t}$ which converges in stationarity conditions. This substitution leads to the second form of the RLS algorithm.

RLS-2nd form

- Basic recursion:

$$w(t) = w(t-1) + K(t) \cdot e(t)$$

- Where $K(t) = \frac{1}{t} \cdot R(t)^{-1} * \phi(t)$ and $e(t) = y(t) - \phi(t)^T \cdot w(t-1)$
- Auxiliary recursion:

$$R(t) = R(t-1) + \frac{1}{t} \cdot \left(\phi(t) \cdot \phi(t)^T - R(t-1)\right)$$

Although the problem of the divergence of $S(t)$ is solved, it remains the problem of the inversion of the matrix $R(t)$ that yields to a high computational effort. This issue can be avoided using the Householder's lemma for which the inversion of the matrix $S(t)$ can be performed knowing $S(t-1)^{-1}$ and inverting a scalar, as follows in the subsequent equation:

$$S(t)^{-1} = S(t-1)^{-1} - S(t-1)^{-1} \cdot \phi(t) \left(1 + \phi(t)^T \cdot S(t-1)^{-1} \cdot \phi(t)\right)^{-1} \cdot \phi(t)^T \cdot S(t-1)^{-1}$$

Defining $V(t) = S(t)^{-1}$, the matrix inversion lemma leads to the third form of the RLS, the one we've implemented.

RLS-3rd form

- Basic recursion

$$w(t) = w(t-1) + K(t) \cdot e(t)$$

- Where $K(t) = V(t) \cdot \phi(t)$ and $e(t) = y(t) - \phi(t)^T \cdot w(t-1)$
- Auxiliary recursion

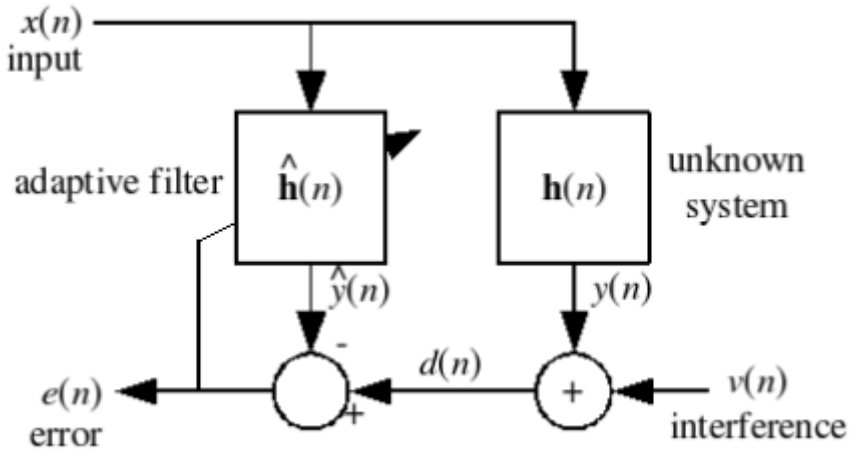
$$V(t) = V(t-1) - \beta(t-1)^{-1} V(t-1)\, \phi(t)\, \phi(t)^T\, V(t-1)$$

- Where $\beta(t-1) = 1 + \phi(t)^T S(t-1)^{-1} \phi(t)$

Experiments

The general environment has the following variables:

- $x(n)$ is the input signal at time n
- $y(n)$ is the echoed version of x that enters in the microphone at time n
- $v(n)$ is the voice signal at time n
- $d(n)$ is the desired filter signal at time n
- $e(n)$ is the error signal at time n



We considered a single-channel environment and we use the adaptive algorithms explained in the theoretical section to update the tap of a FIR filter of length $L = 200$.

We carried out two stages of experiments:

1. We used a manipulated set of impulse responses just to validate the capability of tracking of the different algorithms implemented
2. Then we applied the same tests using real impulse responses to analyse the behaviour in a real world application

Each stage is also subdivided in four sub-experiments depending on:

- The type of $x(n)$, which could be a white noise (stationary) or a real voce signal (non-stationary)
- If the unknown system is time-varying or not. If the unknown system is static, we used one impulse response to simulate the room echoing; otherwise multiple impulse responses are used to simulate the time-varying $h(n)$.

Those are explained in detail in Single Impulse Response and Multiple Impulse Response sections of the corresponding experiment.

Impulse responses description

In this section, we explain the issues we had dealing with real impulse responses and what we did to avoid those problems.

The impulse response is divided in two sections: early reflections and late reflections.

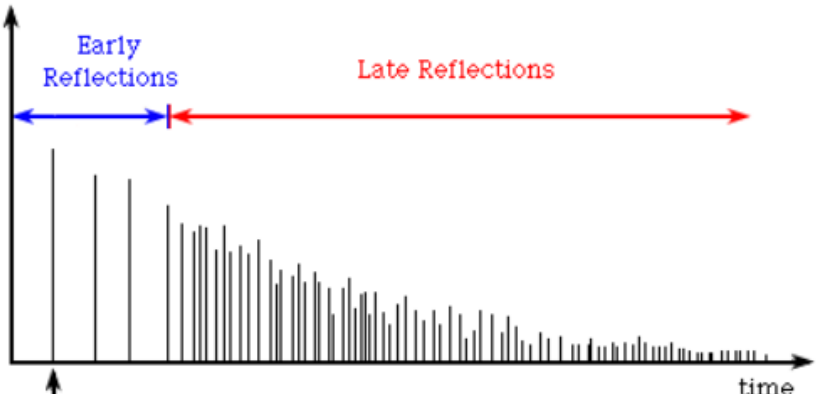
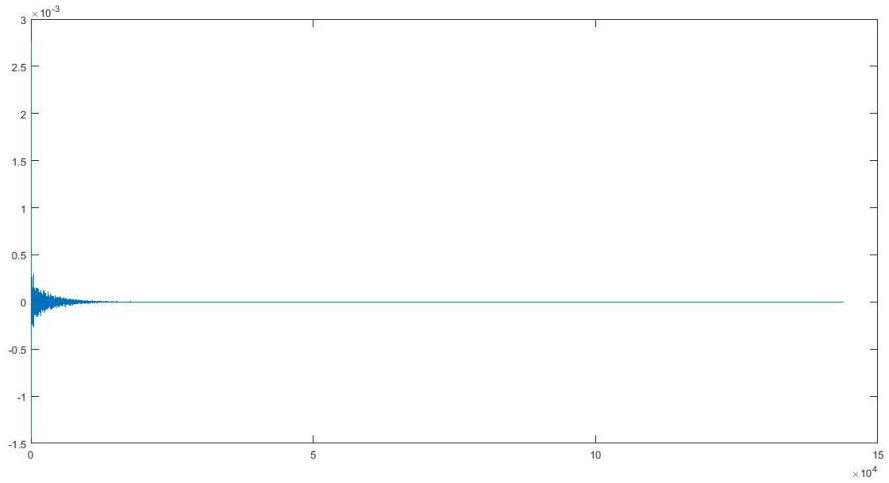


Figure f: Taken from: https://www.vocal.com/wp-content/uploads/2013/05/early_fig2.png

All the impulse responses we used are sampled at 48kHz and look like the following



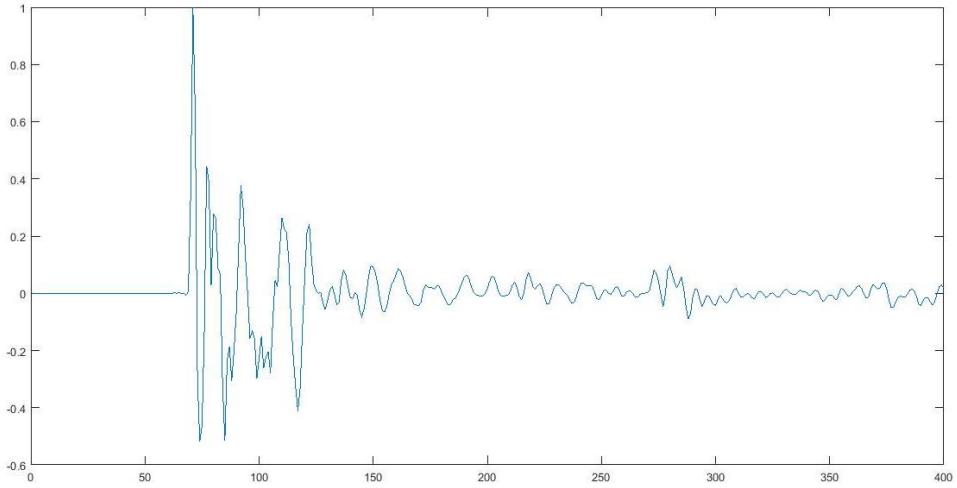
Which has some unwanted characteristics:

1. very long
2. a lot of useless values (close to zero)
3. very small amplitude (0.0028 in the impulse response displayed above)

These can cause some problems in the adaptive algorithms because:

- the adaptation of a FIR filter of such length (144000 samples which would correspond to 144000 taps) would be very impractical due to computational issues. Moreover, because of point 2, we can easily discard some values since they are close to zero
- the very small amplitude may give some problems in the adaptation since a change from one impulse response to another wouldn't be detected

So, for the first stage of experiments (using the manipulated set of impulse responses) we truncated those to the early reflection without considering the late reverberation. We also normalized them to avoid the small amplitude issues. After this processing, the impulse responses look like the following



We downloaded the impulse responses from: https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/

Parameter description

In this section, we explain how we set the parameter of the adaptive algorithm we've used.

- $L = 200$ this is the filter length. Since we've decided to use a FIR filter to emulate the impulse response of the room, we need a filter long enough to represent the impulse response we have used. Even if the filter isn't as long as the impulse response, is able to track and cancel most of the effect related to the usage of the impulse response.
- $P = 20$ this is the number of past vector considered by the APA algorithm. We needed that $P \ll L$ and P has to be big enough to improve its performance with respect to the NLMS performance.
- $\mu_{LMS} = 0.0005$. Initially, we have decided to use the same learning rate for all the adaptive algorithms but this was a wrong choice because leaded to unexpected results: the LMS algorithm seems to converge way faster than the other algorithms (because the μ_{LMS} was too big and μ_{NLMS} and μ_{APA} were too small). Then, we lowered a lot the μ_{LMS} until we didn't get expected results with $\mu_{LMS} = 0.0005$.
- $\mu_{NLMS} = \mu_{APA} = 1$. This isn't the real learning rate of the algorithm because it is balanced with the power of the input signal. With reference to the theoretical section, μ_{NLMS} corresponds to α in the NLMS update equation. μ_{APA} corresponds to μ in the APA update equation.

We also used other parameters (such as k the constant trace value or the minimum forgetting factor μ_0) for the forgetting factor version of RLS but we decided to not use that because of the nature of the experiment we decided to implement.

Stage one: manipulated impulse response

Single impulse response

In this setting, we considered a static environment, so:

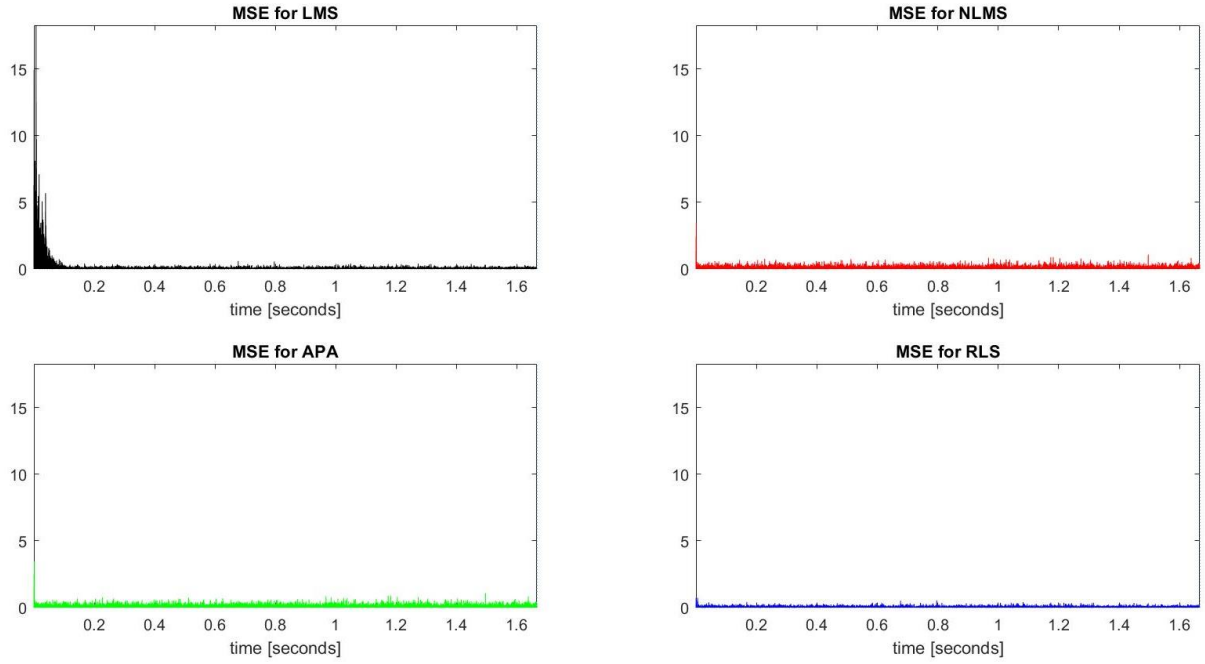
- The room doesn't change
- The acquisition microphone is still

The signal $y(n)$ is obtained as the convolution of $x(n)$ with the impulse response of the room.

$$y(n) = x * h = \sum_j x(j) \cdot h(n - j + 1)$$

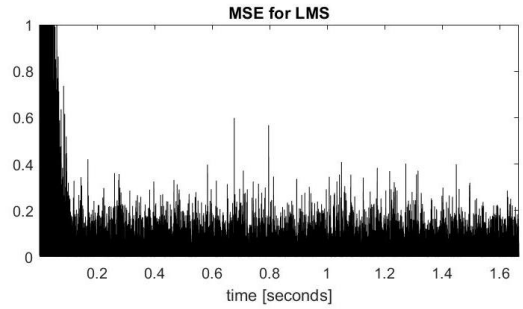
White noise as input

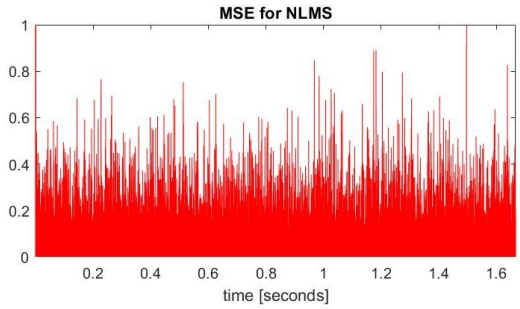
In this case $x(n) \sim WN(0,1)\ \forall n$.

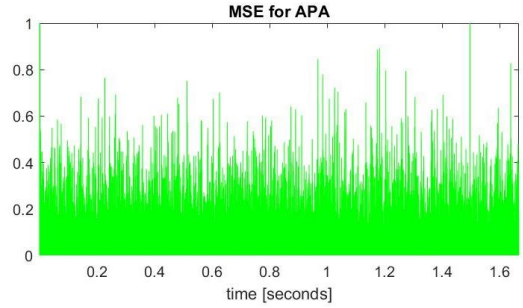


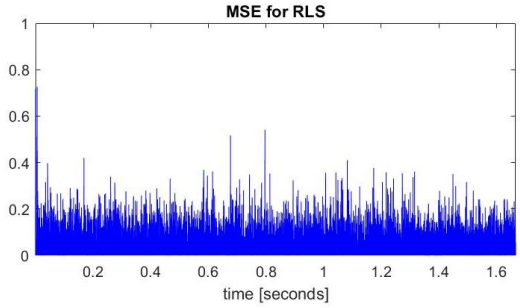
As expected, the LMS algorithm is the slowest one but has very little MSE at convergence. NLMS and APA have somehow the same behavior (also this is expected since APA ≡ NLMS when $P = 1$). RLS is the fastest and the one that has minimum MSE at convergence.

To better see the differences, we applied a smaller y-scale to all the plots



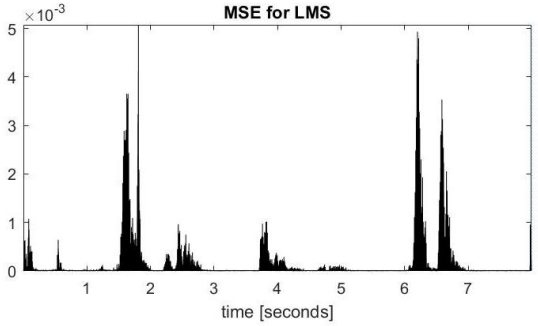


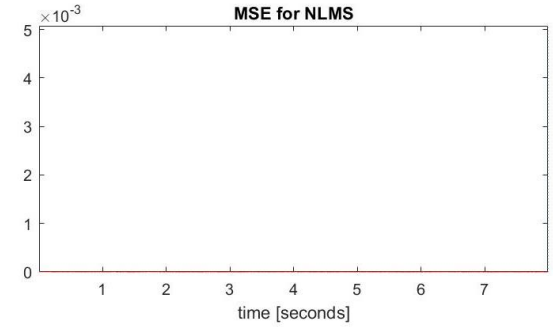


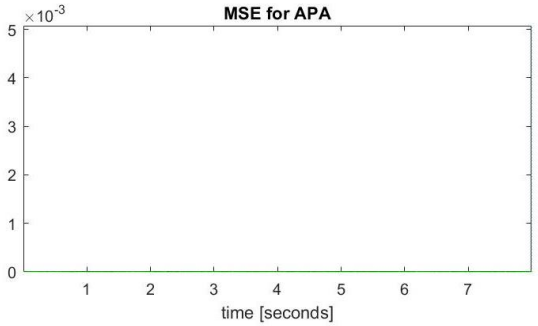


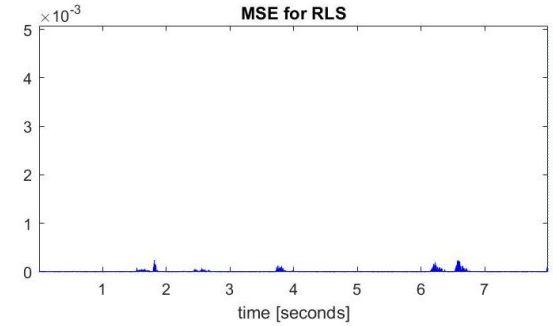
This is somehow expected since the overall learning rate for NLMS and APA is bigger than the one we've used in the LMS. Bigger learning rate means faster convergence and bigger convergence error.

Voice as input

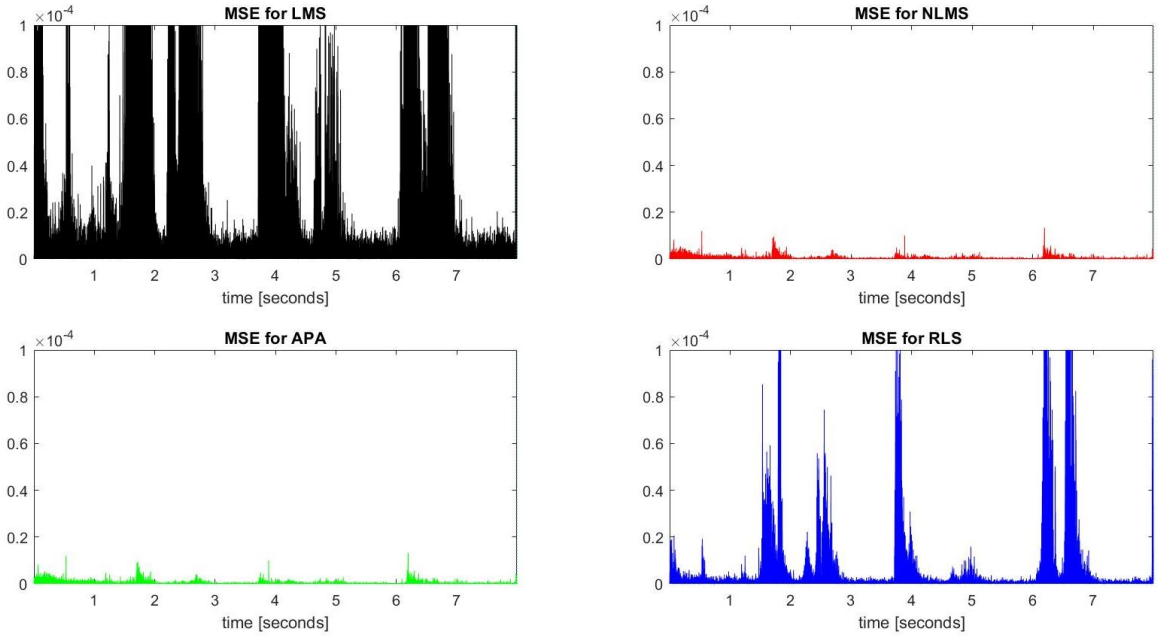








To better see the differences, we applied a smaller y-scale to all the plots



Even with a non-stationary input signal, all the adaptive algorithms have good performances.

We obtain a smaller MSE in comparison with the one obtained in the previous section since the voice signal we've used has less overall power than the white noise we have used before.

Multiple impulse response

In this experiment, we want to simulate a variation over time of the impulse response. This happens if there is a variation on the acoustic path h over time.

To simulate this variation, we convolve the input signal with a different impulse response depending on the time instant n considered. Consider a signal x of length Lx, the echoed version of x that enters in the acquisition microphone is calculated as

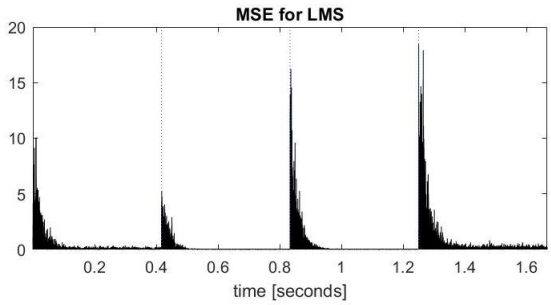
$$y(n) = \begin{cases} x * h_1 = \sum_j x(j) \cdot h_1(n-j+1), & 0 \le n \le \frac{Lx}{4} \\ x * h_2 = \sum_j x(j) \cdot h_2(n-j+1), & \frac{Lx}{4} + 1 \le n \le \frac{Lx}{2} \\ x * h_3 = \sum_j x(j) \cdot h_3(n-j+1), & \frac{Lx}{2} + 1 \le n \le 3 \cdot \frac{Lx}{4} \\ x * h_4 = \sum_j x(j) \cdot h_4(n-j+1), & 3 \cdot \frac{Lx}{4} + 1 \le n \le Lx \end{cases}$$

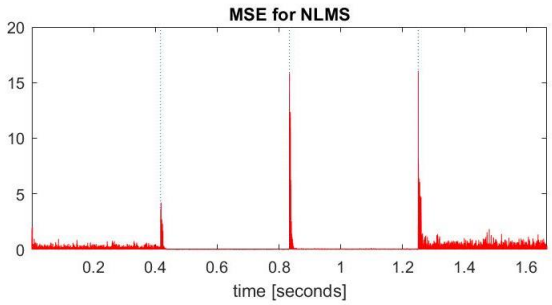
We used impulse response of different types of environment: an office, a meeting room, a lecture room and a bathroom.

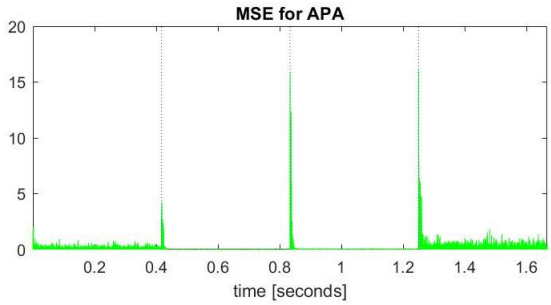
Note: We used the plain version of the RLS algorithm (since we have an abrupt change in the parameter at time instants $n = k \cdot \frac{Lx}{4}$, with $k = 0,1,2,3$) instead of using the version with the forgetting factor. We needed to make the RLS understand when there was a change in the impulse response and we tried various condition but none of those actually worked. The following RLS plots are computed restoring "from the outside" the original matrix V every time there is a change in the impulse response.

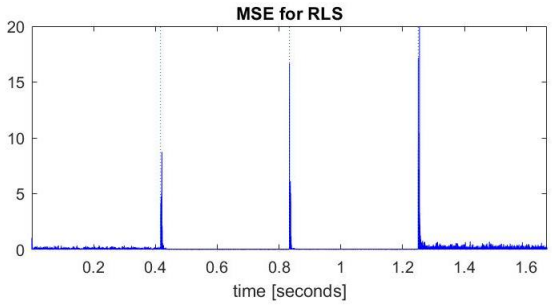
White noise as input

In this case $x(n) \sim WN(0,1)\ \forall n$.



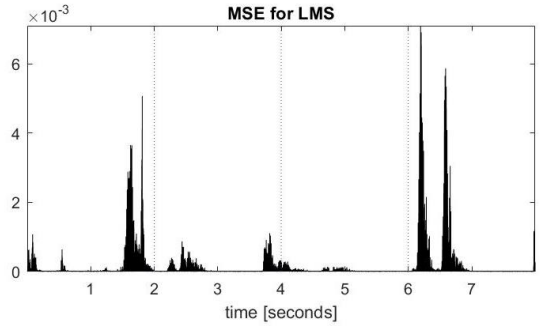


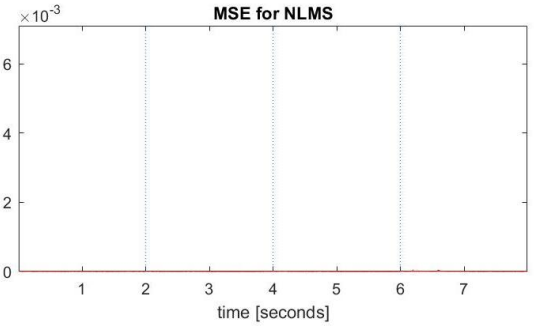


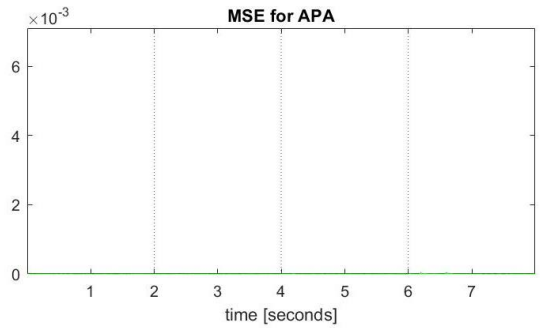


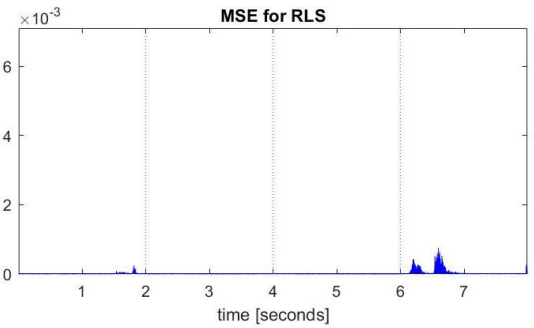
It seems clear that every algorithm can detect the change in the impulse response and adapts itself to it.

Voice as input

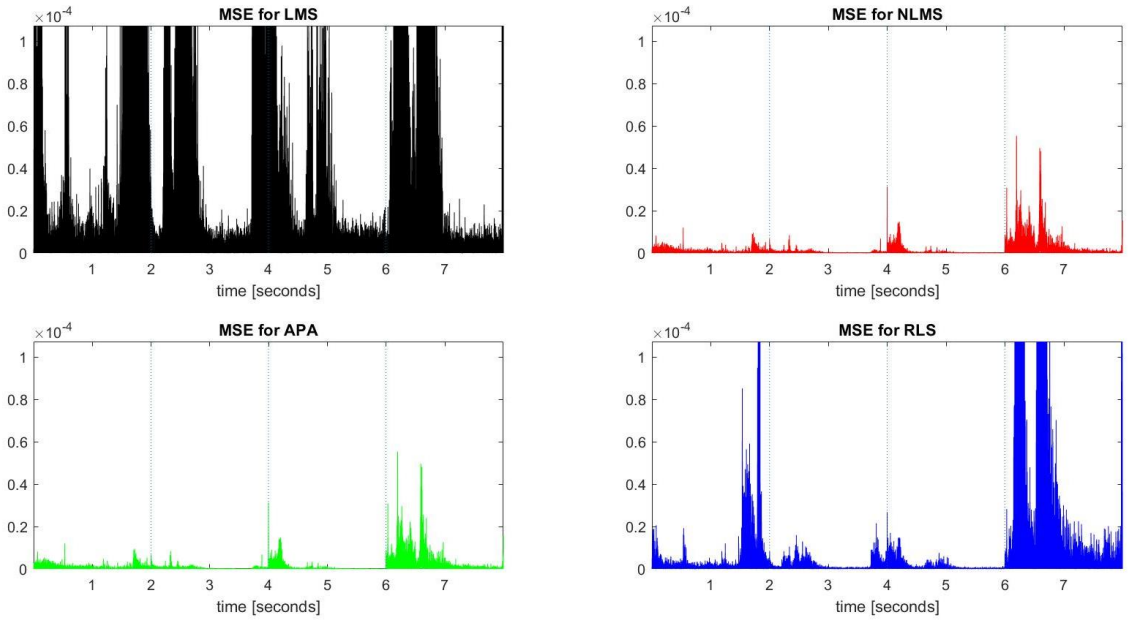








To better see the differences, we applied a smaller y-scale to all the plots



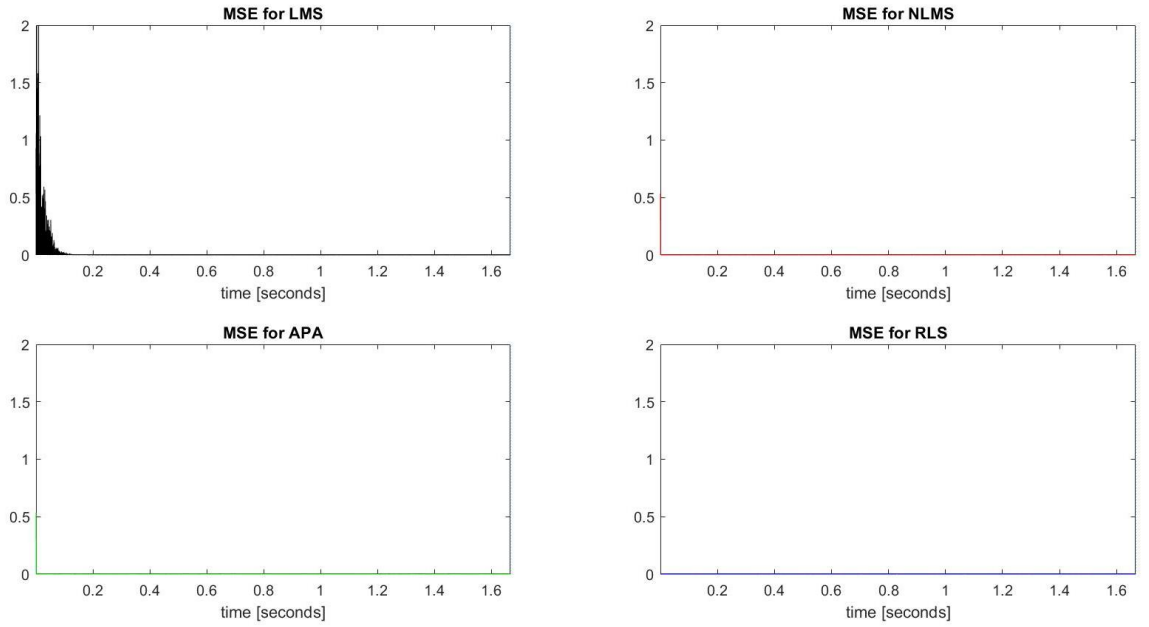
Since the signal is non-stationary, we can see that we have peaks in the plots of the MSE that aren't related to the change in the impulse response. However, the performance (in terms of small values of the overall MSE) seems to be good as the case of the voice input in Single impulse response section.

Stage two: real impulse response

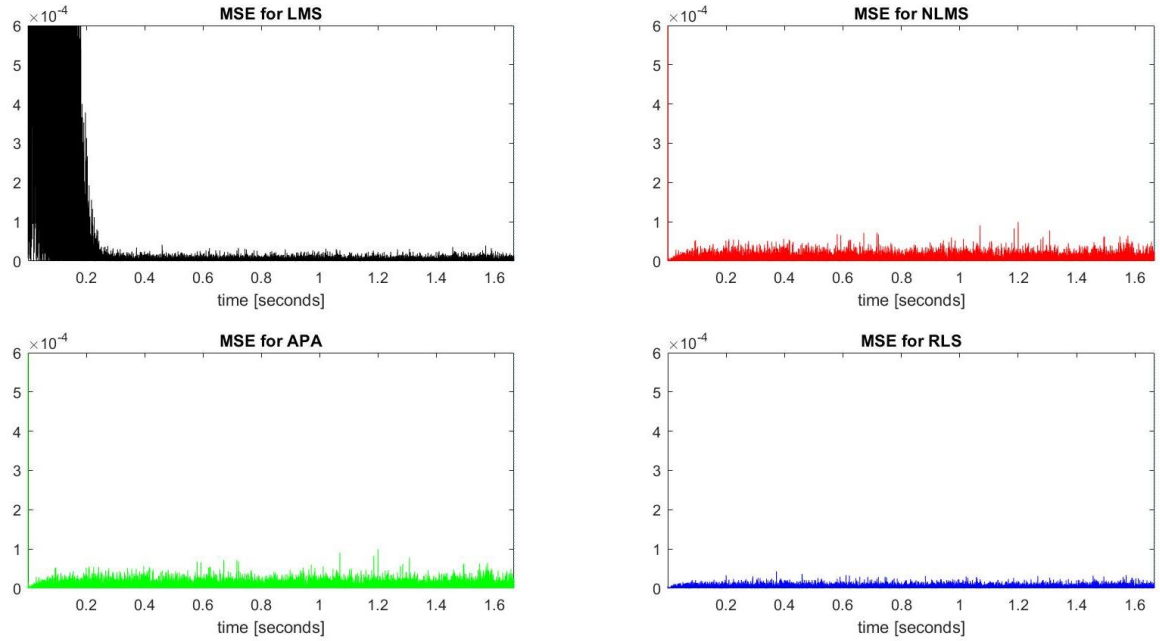
In this section, we're going to show the results obtained using real impulse responses (not manipulated at all). All the sub-experiments are the same as before: the various differences between single/multiple impulse responses and white noise/voice are explained in the paragraph Stage one.

Single impulse response

White noise as input

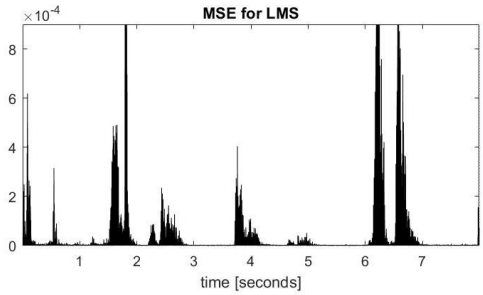


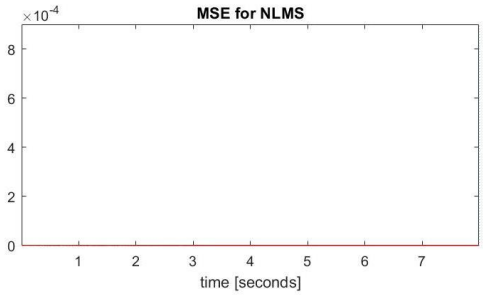
To better see the differences, we applied a smaller y-scale to all the plots

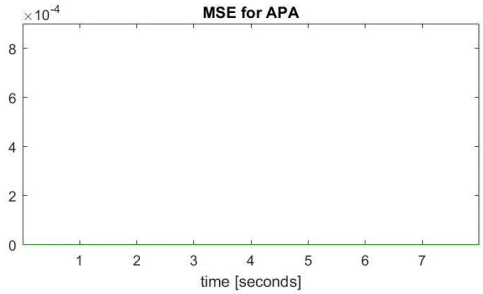


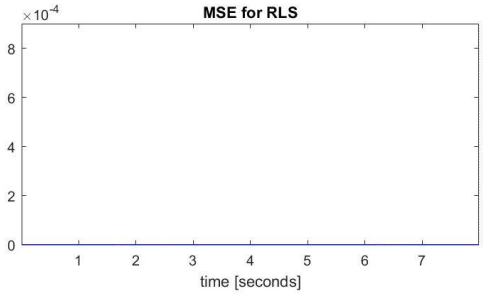
We have an unexpected result: except for LMS, we can't see the usual decreasing trend in the MSE but we have a high initial value and then the MSE abruptly decreases and start an increasing trend up to the convergence value.

Voice as input

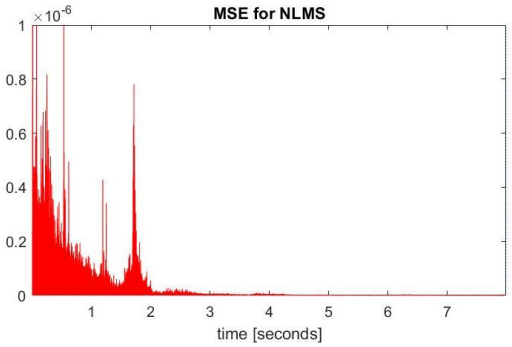


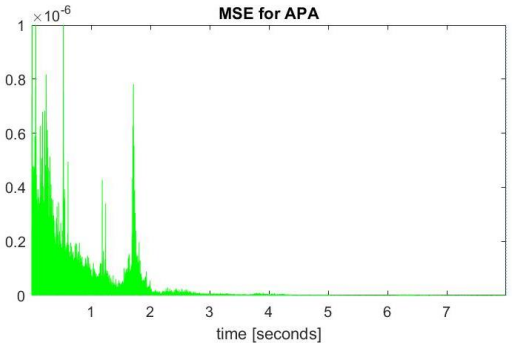




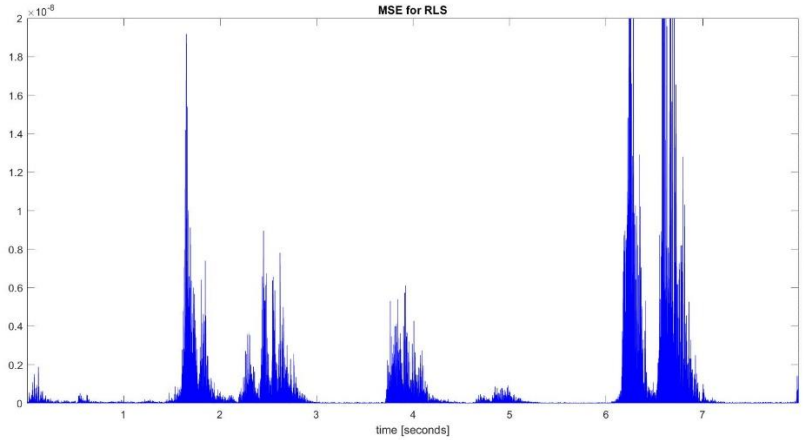


To better see the differences, we need to zoom a bit.





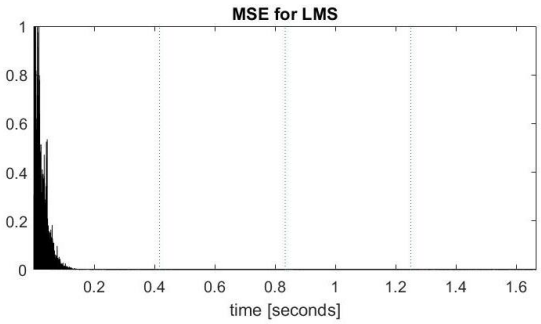
NLMS and APA have somehow the same behavior: even with a non-stationary signal, their MSE trend goes towards zero as if the algorithms have learnt the parameters of the system (which is exactly what we want).

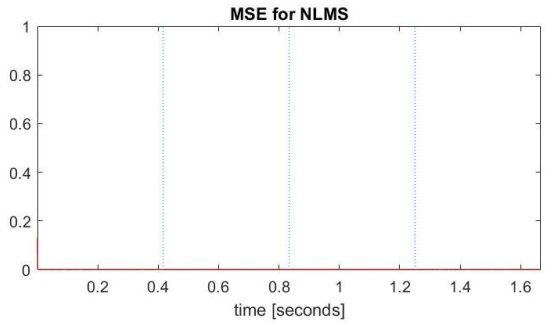


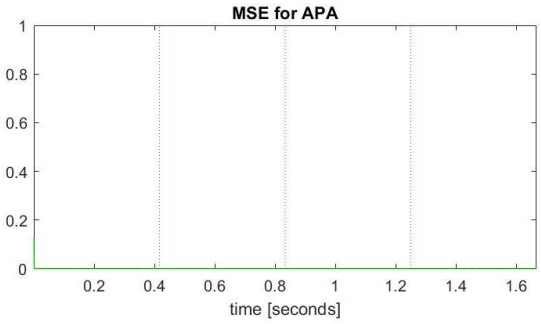
The RLS have the same behavior of LMS but with a much smaller MSE (around 5 orders of difference).

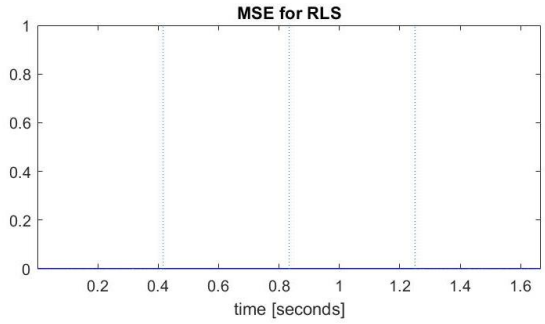
Multiple impulse response

White noise as input

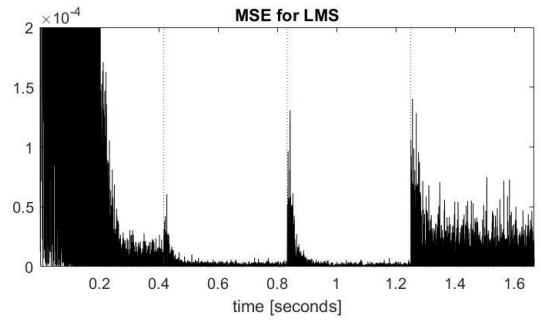


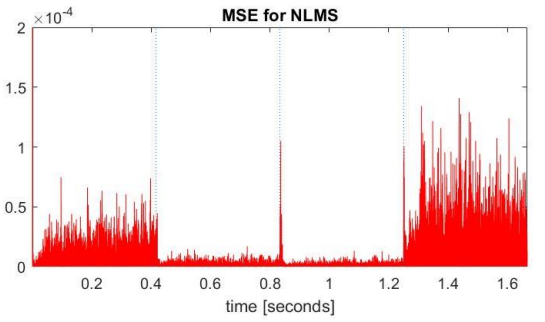


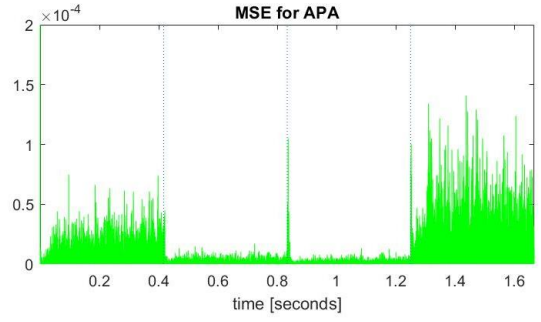


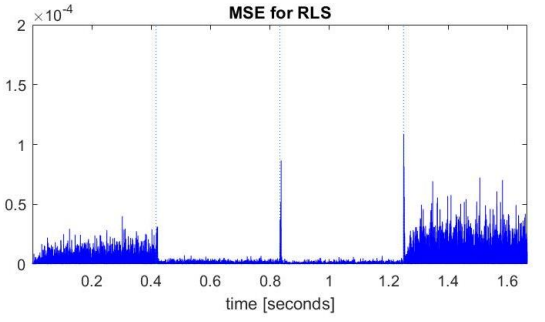


To better see the differences, we applied a smaller y-scale to all the plots



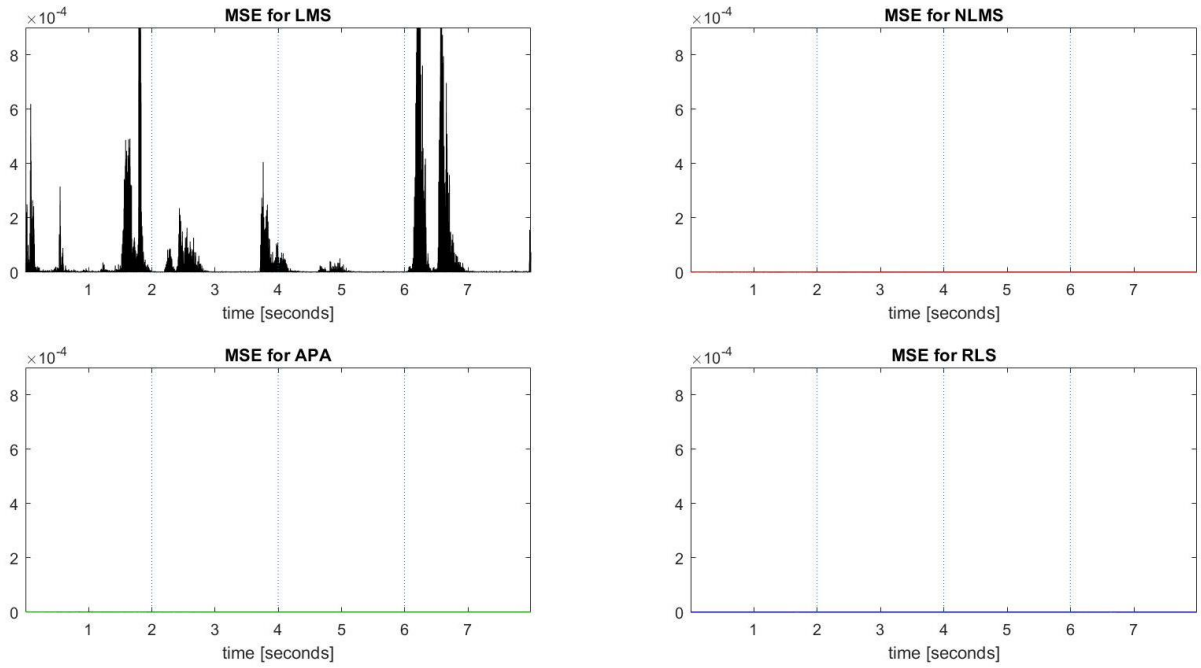






We can see that the algorithms maintain the ability of tracking the variation in the system, even with the real impulse responses.

Voice as input



Even if the y-scale gets smaller, all the algorithms maintain the same behavior as in the case of Single impulse response. This probably happens because the voice is non-stationary and the impulse responses have small amplitude values, so small that a change in the impulse response is irrelevant with respect to variation in the input signal.

Final remarks

NLMS and APA

One of the most evident results from the various experiments we have performed, is that NLMS and APA have almost the same results. In fact, their performances in term of MSE difference are very low, as shown in the following plots.

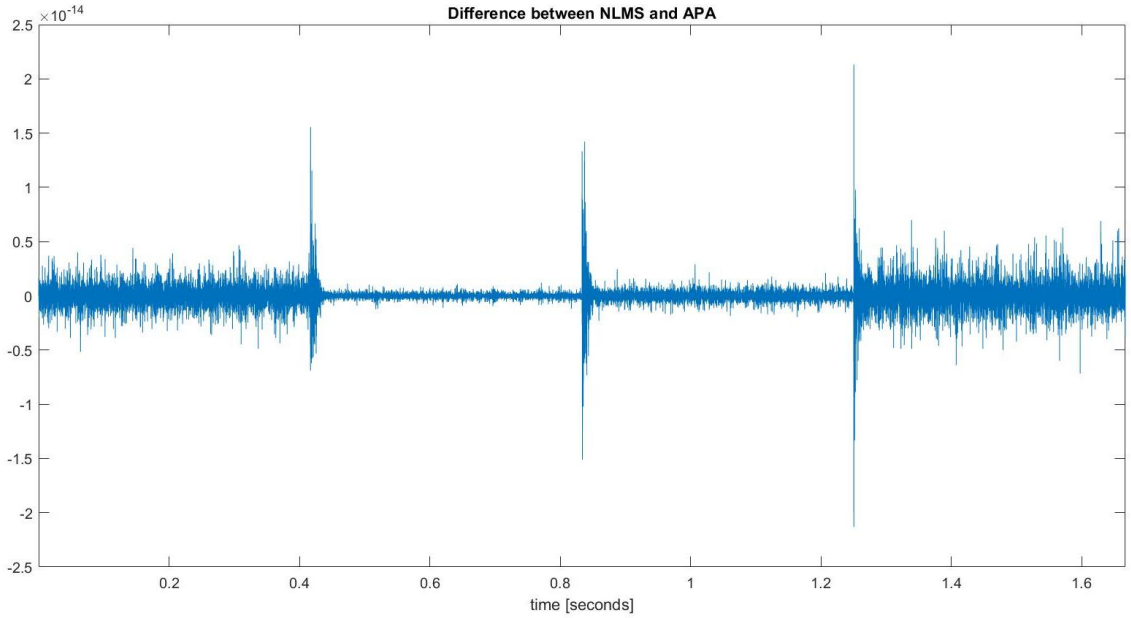


Figure g: Difference between NLMS and APA with white noise as input and four impulse responses

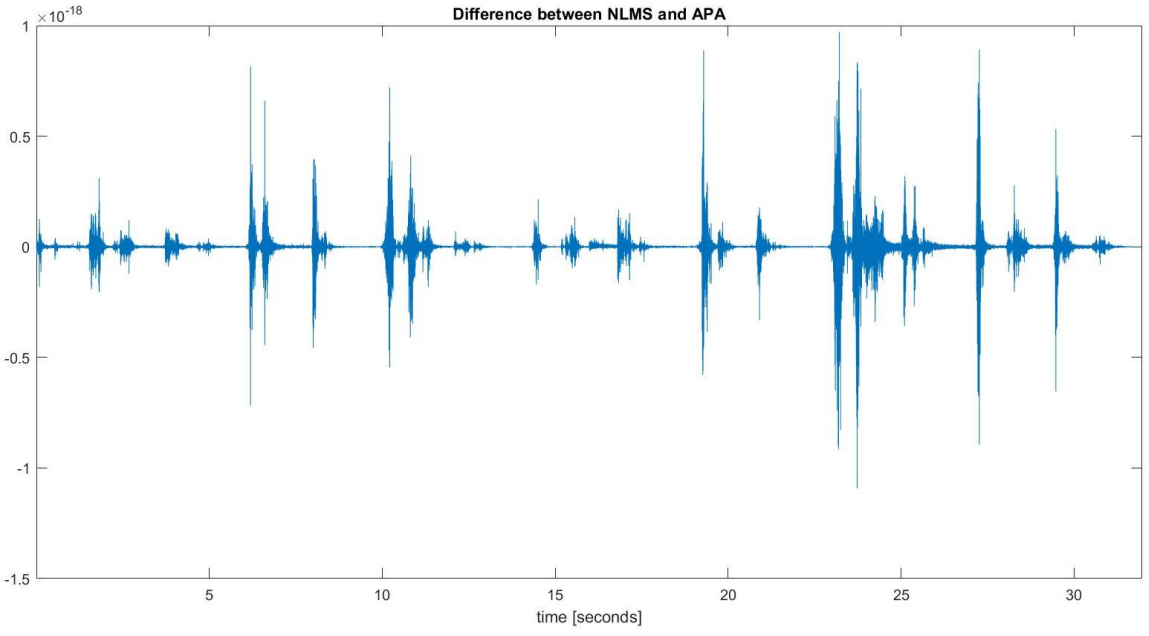


Figure h: Difference between NLMS and APA with voice as input and four impulse responses

In the latter, we had expected a bigger difference due to the high correlation of the input vectors that would make the convergence rate of NLMS worst and enlighten the better performance of APA in this case.

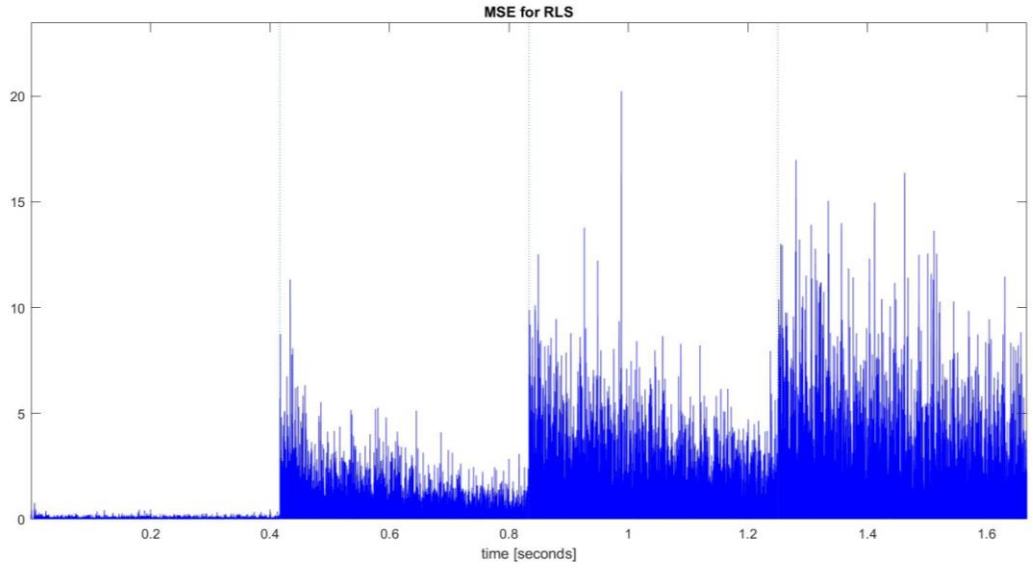
A very slight improvement of APA with respect to NLMS is manifested if the number P of past samples considered is increased up to the half of the filter length ($P = 100$), but due to the related high computational load we decided to keep $P = 20$.

Considering the case of white noise, the difference is around 10^{-14} and the MSE has the highest peak at around 21 (for both NLMS and APA): the difference is totally negligible and non-visible nor audible.

We have concluded that for real time applications the use of NLMS would be preferable instead of the APA due to the higher computational load of the latter and the negligible difference between the two.

RLS adaptation

The plain version of RLS after a while is not reactive with respect to changes because $V \to 0$ and so $K \to 0$ as $t \to \infty$. To test its inability to track changes in the system, we carried out an experiment using white noise as input and multiple impulse response, resulting in the following behaviour



The system converges really fast at the beginning but then the model isn't able to adapt since the error is multiplied by a constantly decreasing gain $K(t)$.

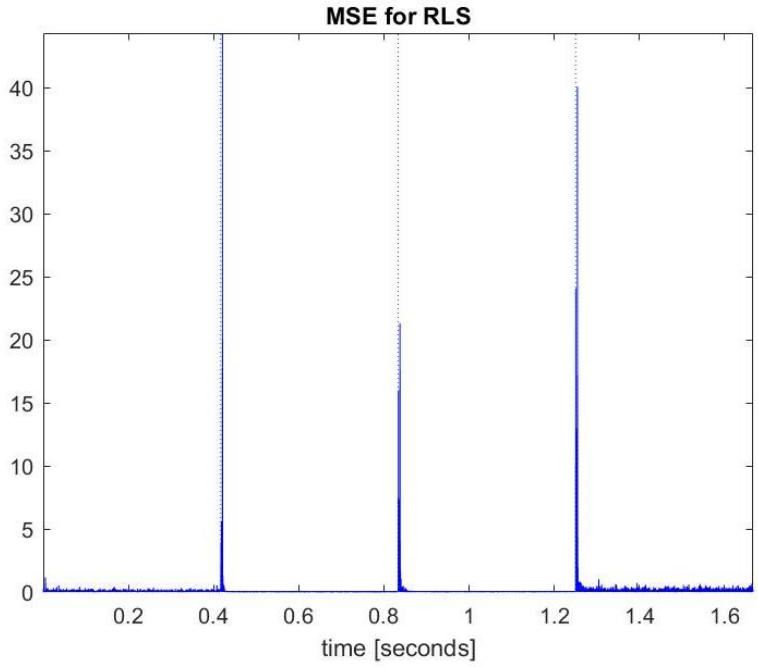
Since the values of the impulse response vary by jumps, it was necessary to discard the option of using a forgetting factor, and we try to implement the reset of V every time there is a change in the impulse response. This wasn't easy to implement: we tried different strategies all based on the comparison between the "actual" error and the "past" error but none of them worked properly because the matrix V was updated many times instead of just one whenever there was a change in the acoustic channel.

Some examples of strategies that we've tried:

- if $\frac{e(t)}{e(t-1)} > \delta$ restore the initial value of V, where δ is a tunable threshold
- every l_{MSE}, check if $\frac{mean(\boldsymbol{MSE}(t))}{mean(\boldsymbol{MSE}(t-l_{MSE}))} > \delta$ restore the initial value of V, where
  - δ is a tunable threshold
  - $\boldsymbol{MSE}(t) = [MSE(t), \dots, MSE(t - l_{MSE})]$ is a buffer of the last l_{MSE} MSE samples
  - l_{MSE} is the tunable length of MSE buffer

Depending on the value of δ chosen, V gets updated too frequently (causing a burst in the MSE) or never gets updated (falling in the plain RLS case).

Then, we've decided to force the reinitialization of V whenever there is a change in the impulse response resulting in this behaviour.



Algorithm comparison

From the experiments we carried out, it seems that RLS is the most performant method both in terms of speed of convergence and convergence error. Remember that the version of the RLS we used is not actually "adaptive", as explained before, but is forced to adapt to the variations of the system by re-initializing the matrix V every time there is a change.

We considered that acoustic path changes abruptly in time (so we have jumps in the parameters) which might not be a real case scenario. Probably in a real application, the usage of a forgetting factor might be more suitable and so the RLS will become really adaptive to changes.

Even if the LMS seems to be the slowest method, when it reaches convergence it has good result in terms of excess MSE and shows good performances in tracking the real system. Moreover, if we look at the plots in section Single impulse response with white noise as input, we can clearly see that it takes about 0.2 seconds to converge, not so much if we consider the small learning rate we've used. It is possible to use a variable forgetting factor: higher at the beginning (to speed up convergence) and lower when convergence is reached (to decrease the excess MSE). So, a good option can be the Correlation LMS as an adaptive algorithm to use. With respect to the algorithm we have developed, the LMS could be hard to tune dealing with real world implementations (in terms of choice of the step size value).

NLMS and APA are very fast algorithms but the produce a higher excess MSE with respect to LMS or RLS. So, as said before, we wouldn't use APA due to its higher computational load and its negligible increase of performance with respect to NLMS. Besides the convergence error, NLMS and APA seemed really easy to tune which is a positive aspect when dealing with real applications.

To resume

RLS

- Not easy to make fully adaptive (in case of parameter jumps)
- Easiest to tune
- Minimum excess MSE
- Fastest convergence

LMS

- Completely adaptive
- Not so easy to tune
- Minimum excess MSE
- Slowest convergence

NLMS and APA

- Completely adaptive
- Easy to tune
- High excess MSE
- Fast convergence