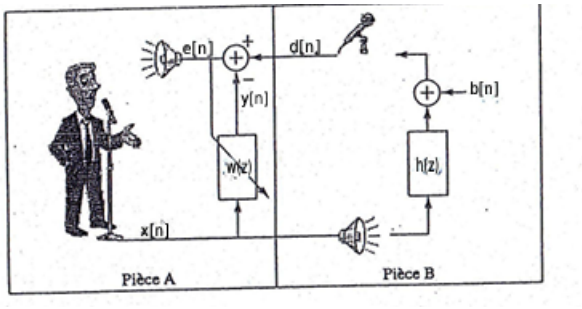
GHILENE RAYANE
LASSO NICOLAS

Tp2 Méthodes de Signal Avancées
Annulation d'Écho Acoustique

The goal of this TP:

In this Tp, we will try to eliminate/ reduce the echos in the case of a teleconference as presented in the following image :



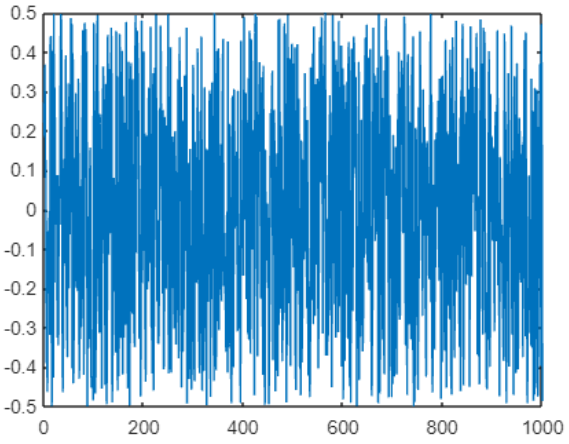
2.1 Preparation:

The equations of the LMS algorithm considering the different elements in the image above are the following:

$$w[n+1] = w[n] + \mu \cdot e[n] \cdot x[n]$$
$$e[n] = d[n] - y[n]$$

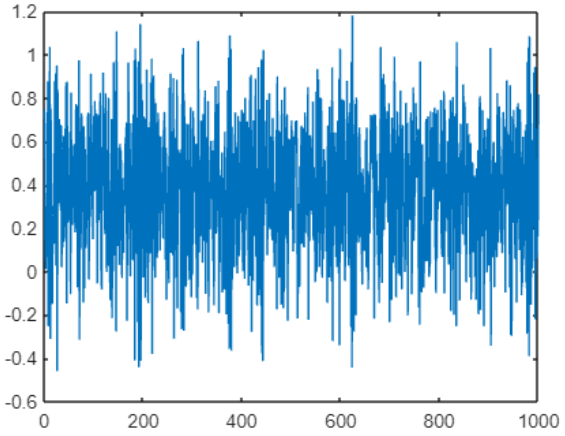
2.2 Test signal Generation:

We started by defining our input signal x as a white noise:



Plot 1. Input signal x

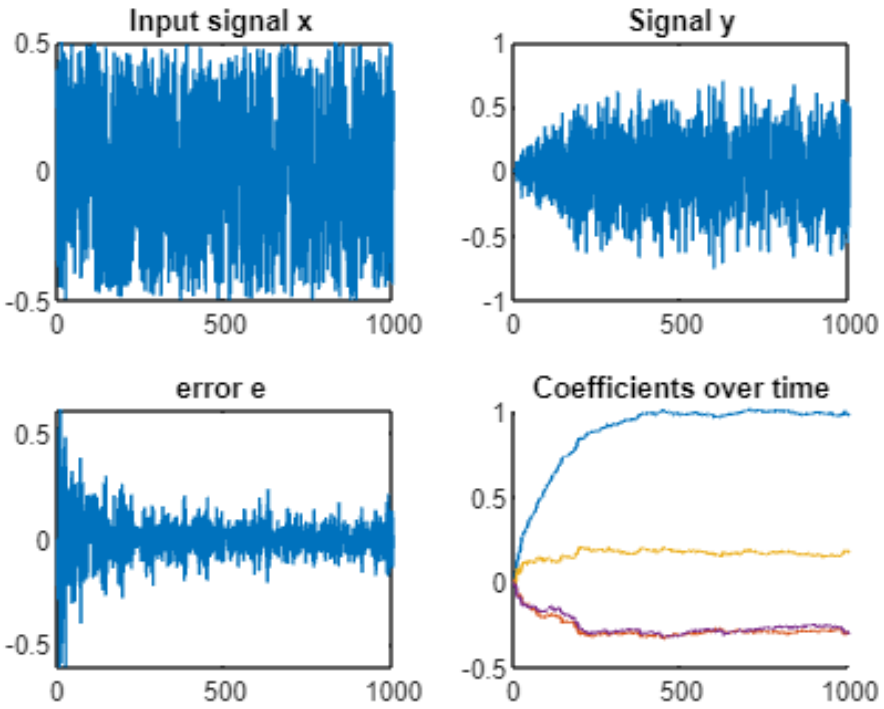
We sent this signal through a finite impulse response filter in order to obtain the filtered signal d:
The filter coefficients are defined as Follows: W = [1 0.3 -0.1 0.2]t



Plot 2. Filtered response d of the input signal x

2.4 Validation of the LMS algorithm

We started by applying the algolms function we created onto our input signal x and we observed its output:
Hyperparameters :
n = 10000 (length of the input signal x)
P = 4 (filter order)
μ = 0.001 (standard deviation)



Plot 3. filtered response d of the input signal x

As we can see our filter has the following values
w = [0.5755 -0.1788 0.1122 -0.1736]t

- What should the value of Wopt be?

Ideally, the W filter should mimic the H filter this has the same values
h = Wopt = W = [1 0.3 -0.1 0.2]t

When calling our algolms function using an input signal having n = 100000
(n = length of the input signal) we obtain

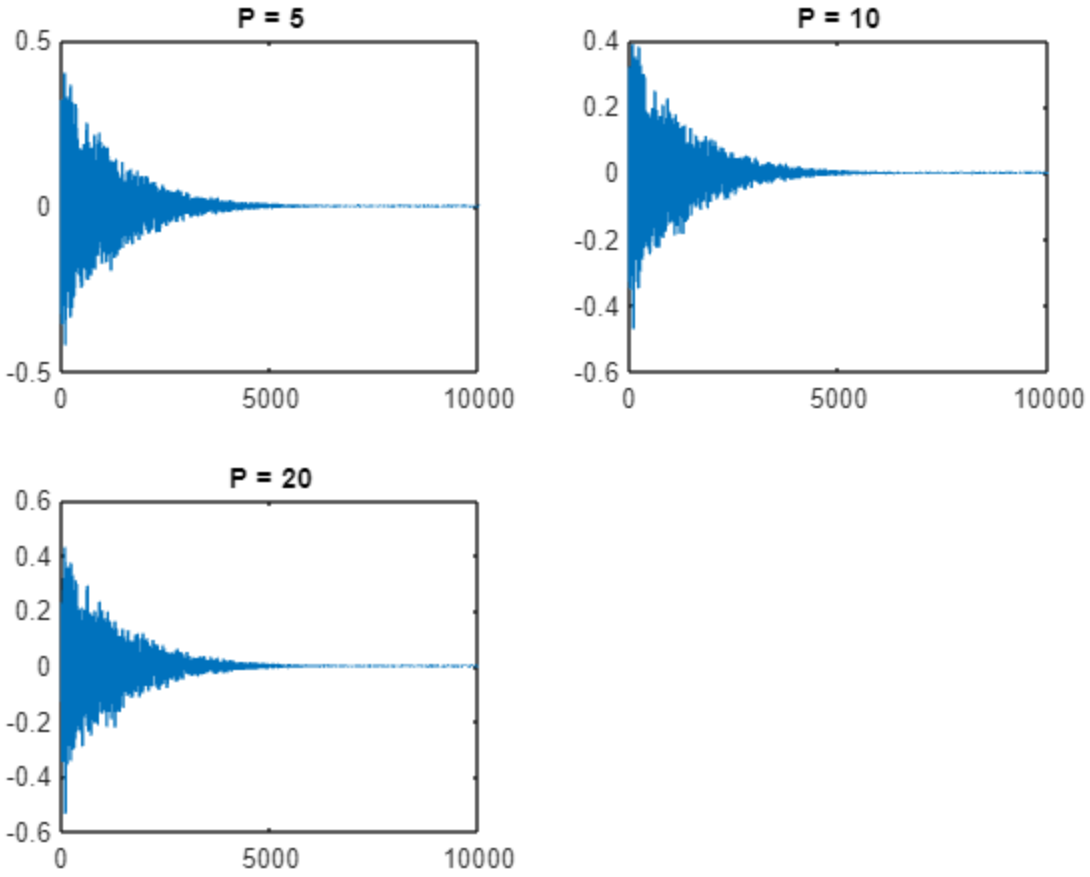
w =[1.0022 -0.3029 0.1926 -0.2895]t which approaches the ideal filter description which is understandable having our algorithm converge. Considering the simplicity of the LMS implementation used and the trivial case, it's understandable that, even though the algorithm converges to a value close to the real coefficients, the sign of these might be switched.

Side note: Since we're using a Gaussian white noise as our input signal, mu won't have an impact on the convergence of our filter since the covariance matrix is the Identity matrix

2.5 Test of the LMS algorithm:

Testing the effect of different P values on our signal:
We started by iterating through different values of P in order to test the effect of this hyperparameter on the convergence of the error:

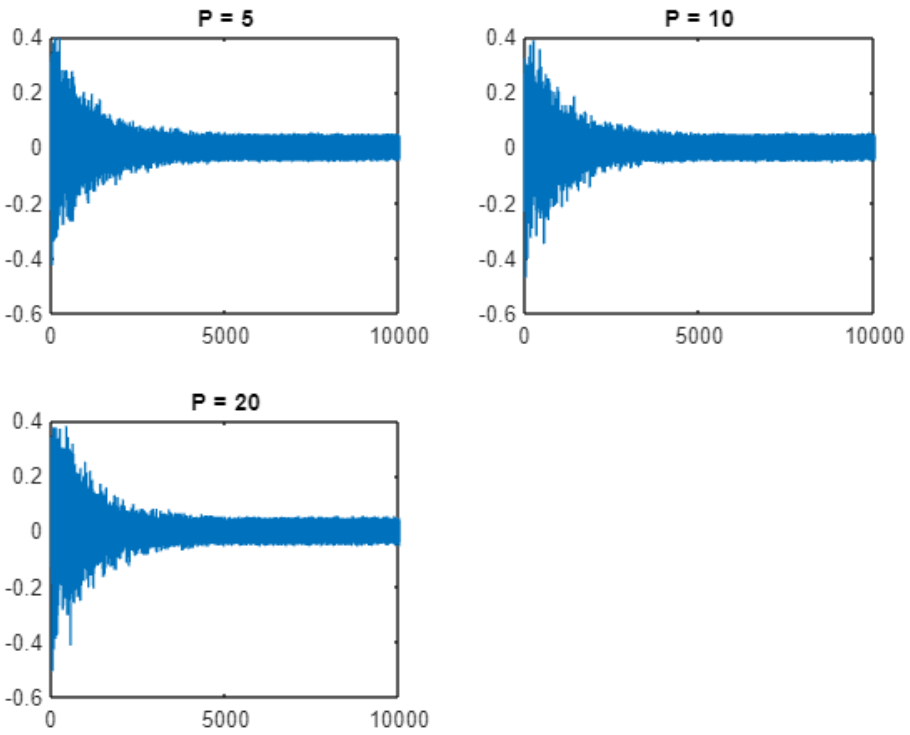
we will test the filter for the following values P = [5, 10, 20]



Plot 4. Effect of P on the convergence of the error

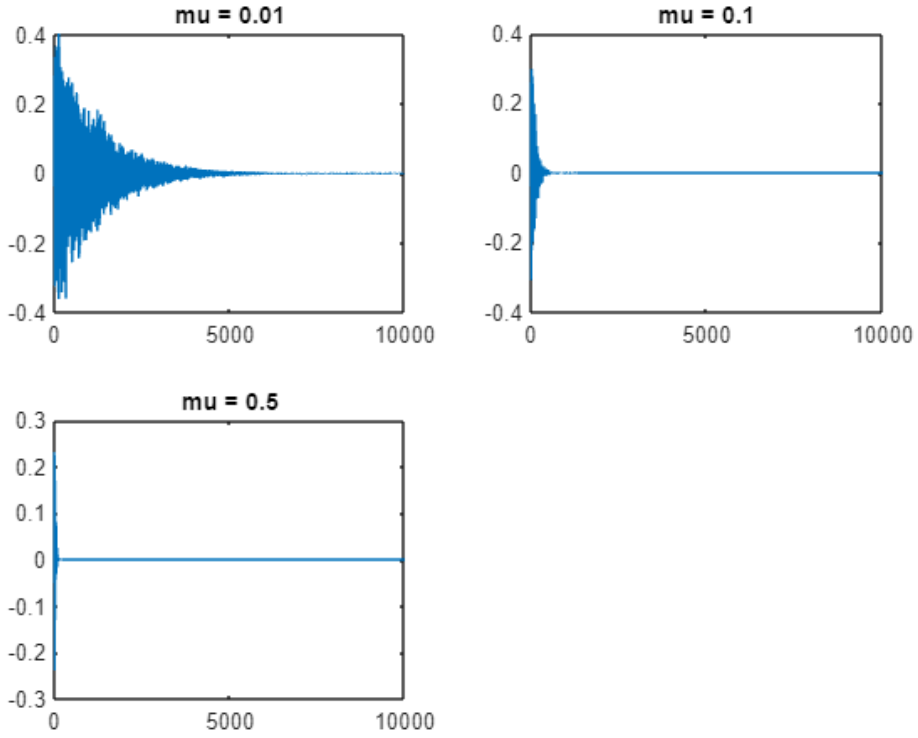
The response of the filter ends up converging, and minor differences in the response for different values of P

When adding noise to our signals we get the following plot:



Plot 5. Effect of P on the convergence of the error with added error

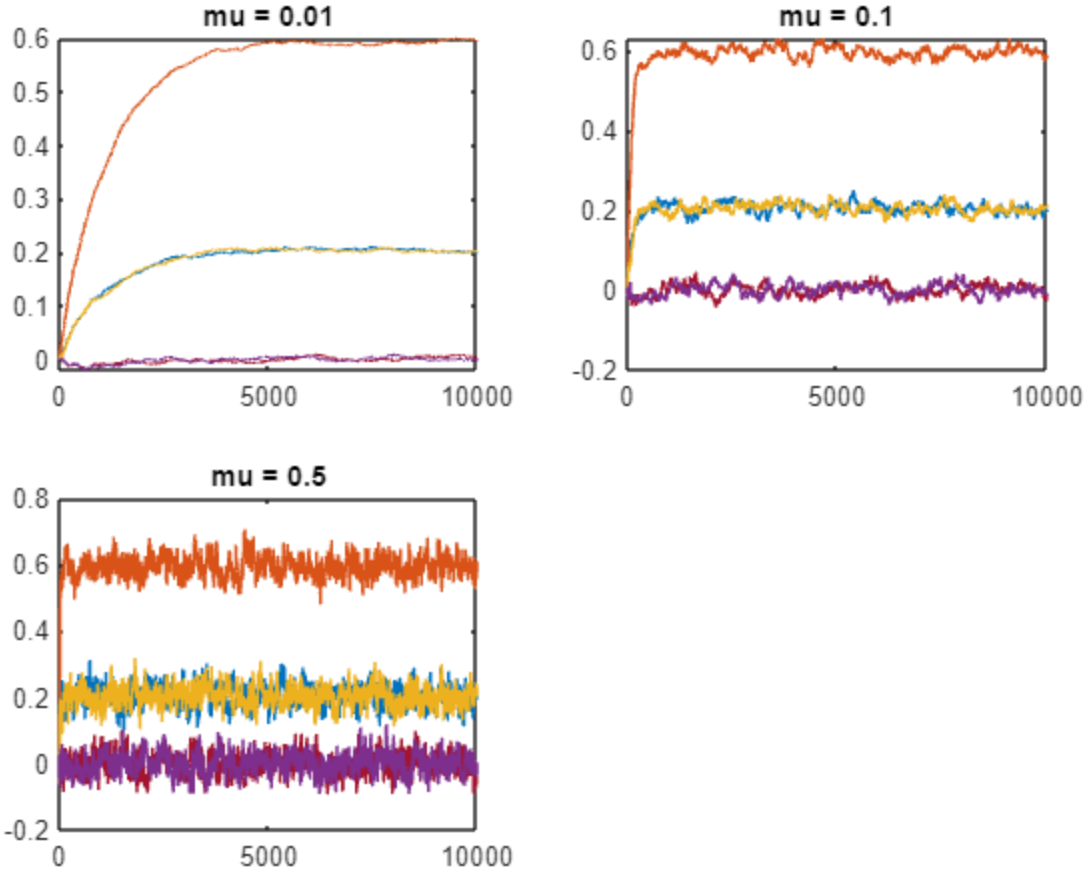
Testing the effect of different mu values on our signal:



Plot 6. Effect of mu on the convergence of the error

The spectral order of the filter is one of the most influential parameters for this algorithm convergence. This to the point that even changing the learning rate, we still converge to the real values. Said this, we see that the learning rate affects how fast we get to that value, and we can appreciate that in plot 6.

When we leave P fixed and we study the effect of the learning rate we can observe something similar. All of the coefficients in the plot 7 converge. When we choose an optimal learning rate the convergence is smooth, and when it is too big we overstep but always with a mean value on the real values of the original filter.



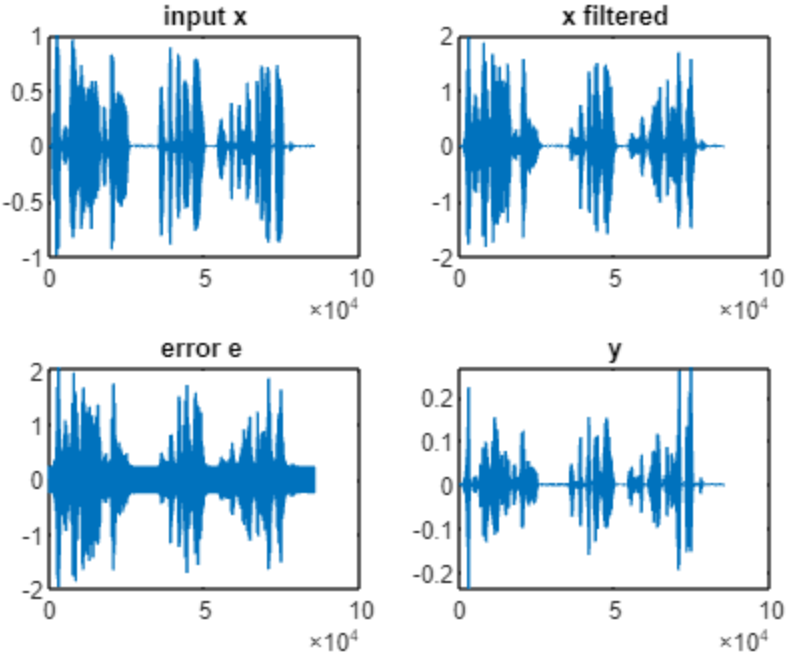
Plot 7. Effect of mu on the convergence of the Learning rates

II. Application

Side note: wavread was removed from matlab, we used audioread() instead.
First

First Application: One Voice Audio Signal

For the first application, we added white noise to our input signal (x) and we ran that through the LMS algorithm.



Plot 7.First application

As observed in our plot, we retrieved most of the input signal which was attenuated a little bit, but its fundamentals remained the same.

Using a higher Learning rate or a lower filter order might result in a faulty attenuation of the echo. A very low Learning rate isn't good either because the output signal's amplitude will be very low

Second Application: Two Voice Audio Signal

For our second application, we want to eliminate the echo on a mixed audio signal composed of two voices. In this case, following the model described on the image at the beginning of this TP, we create our own audio signal and filter it following again, the impulse response of

the room, the only difference here is that the signal consists of a “close” audio and a “far” audio signal, where only the “far” one is filtered.

Playing around with the parameters of our method we can see what we observed before on Plot 7, increasing the learning rate (lr=2) let us hear some background noise, and even some of the remains of the filtered echo signal, and a really low learning rate returns a converging filter with a low amplitude signal, resulting in too low volume.

End.