Adaptive filtering

Cristian Iñiguez Rodriguez, 1566514@uab.cat

***Abstract*—A statistical filter using the LMS algorithm will be implemented and tested on an audio signal with colored AWGN noise, and then used to cancel a spoofed signal. The performance of the filter will be compared to a notch filter.**

Introduction

In this report, a statistical filter will be implemented to eliminate an unwanted component of a signal, using the LMS algorithm. The idea is to first check that the filter works correctly by removing "colored" AWGN noise from an audio signal that has been distorted with that type of noise, that is, by actively canceling the noise. Once the LMS has been implemented, it will be used to estimate a filter necessary to cancel the spoofed signal. A comparison will be made with a notch filter, specifically designed for this purpose, and a study will be made on its convergence.

I. Answer to Question 1

The convolution is a mathematical operator that transforms two functions into a third function. In digital processing, it represents an output for a linear system. It can be written as

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{1}$$

To implement the convolution operation using the p filter coefficients, which are stacked into a $(P \times 1)$ vector h_n, it is necessary to stack the input signal $x(n)$ into an $(P \times 1)$ vector x_n, according to the following definition

$$x_n(n) = \begin{bmatrix} x(n) \\ \vdots \\ x(n-p+1) \end{bmatrix}$$

Once we know how to represent x_n, the representation of the convolution in vector form is given by

$$y[n] = h_n^H x_n = \sum_{k=0}^{p-1} h^*[k] \cdot x[n-k] \tag{2}$$

II. Answer to Question 2

The parameter μ, also known as the step size, determines the rate at which the weights of the filter are updated. In order to ensure that the Least mean squares algorithm converges, the step size μ should be chosen carefully. If μ is too large, the algorithm may diverge and fail to converge. On the other hand, if μ is too small, the convergence may be slow. In general, the step size μ is chosen to be in the interval

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{3}$$

where λ_{max} is the largest eigenvalue of the input correlation matrix.

However, the bound on the step size for convergence provided by Eq. 3 has limited practical use. One reason for this is that the upper bound is typically too large to guarantee the stability of the LMS algorithm. Additionally, the upper bound is expressed in terms of the largest eigenvalue of the correlation matrix, which means that it is necessary to know the value of R_x in order to use this bound. If R_x is unknown, then it is necessary to estimate its largest eigenvalue, denoted as λ_{max}. One solution to this problem is to use the fact that the trace of R_x can provide an upper bound on λ_{max} [1],

$$\lambda_{max} \leq \sum_{k=0}^{p-1} \lambda_k = tr(\mathbf{R}_x)$$

Therefore, if $x(n)$ is wide-sense stationary, then R_x is Toeplitz and the trace becomes $tr(\mathbf{R}_x) = p \cdot r_x(0) = p \cdot P_x$. As a result, we get a more conservative bound

$$0 < \mu < \frac{2}{p \cdot P_x} \tag{4}$$

This interval is valid for the unnormalized LMS algorithm. In the case of the normalized algorithm, the step size is given by

$$\mu(n) = \frac{\beta}{||x(n)||^2} \tag{5}$$

where β is the normalized step size with values $0 < \beta < 2$. The LMS weight vector is updated using $\mu(n)$ instead of μ.

III. Answer to Question 3

An adaptive filter is a system that adjusts its coefficients over time in order to achieve a desired outcome, such as minimizing an error signal. The LMS (Least Mean Squares) algorithm is a widely used method for adapting the coefficients of an adaptive filter. It is a type of stochastic gradient algorithm, meaning that the filter coefficients are updated based on the gradient of an optimization cost function, typically the squared magnitude of an error signal. The block diagram of an adaptive filter using the LMS algorithm is shown in Figure 1.

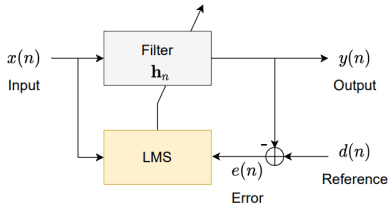


Fig. 1: Block diagram implementing the LMS

It is a practical implementation of the optimal Wiener filter, which minimizes the MSE. Instead of implementing the optimal solution directly, the LMS algorithm uses a gradient-based approach to iteratively minimize the MSE cost function. The main advantage of the LMS algorithm is that it does not require inverting the correlation matrix of the input signal, as the closed-form Wiener solution does.

The key feature of the LMS algorithm is that it employs the "instantaneous" gradient of the MSE instead of the MSE cost function. It is obtained by replacing the second order moments of the original gradient with their instantaneous estimates at time instant n. This gives us the result:

$$\nabla_{h_n} MSE(n) = \mathbf{R}_n \mathbf{h}_n - \mathbf{p}_n \Rightarrow$$
$$\nabla_{h_n} MSE_{inst}(n) = (\mathbf{x}_n \mathbf{x}_n^H)\mathbf{h}_n - d^*(n)\mathbf{x}_n \tag{6}$$

The coefficients of the filter are iteratively updated using the instantaneous gradient of the MSE through the following recursive process

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu \nabla_{h_n} MSE_{inst}(n) \tag{7}$$

The $(P \times 1)$ vector $\mathbf{h}_n$ contains the P coefficients of the filter at time n, and μ is the step size. By stacking the samples of the input signal $x(n)$ into a $(P \times 1)$ vector $\mathbf{x}_n$, the output of the filter can be calculated using Eq. 2. By substituting the instantaneous gradient of the MSE from (6) into the filter coefficients recursion in (7) and then using the result in (2), the LMS algorithm yields the following filter coefficients

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu \mathbf{x}_n (d^*(n) - y^*(n)) = \mathbf{h}_n - \mu \mathbf{x}_n e^*(n) \tag{8}$$

To make the normalized LMS algorithm, we would use (8) and substitute μ with the definition of $\mu(n)$ in (5), giving us the following equation

$$\mathbf{h}_{n+1} = \mathbf{h}_n - \mu(n) \mathbf{x}_n e^*(n) = \mathbf{h}_n - \frac{\beta \mathbf{x}_n e^*(n)}{||\mathbf{x}_n||^2} \tag{9}$$

Matlab implementation of normalized LMS algorithm

```
function [H, y, e] = LMS_algorithm(d, x, P, mu)
    N = length(x);
    e = zeros(1, N);
    y = zeros(1, N);
    H = zeros(P, N);

    for n = P:N
        x_n = x(n:-1:n - P + 1);

        y(n) = H(:, n)' * x_n;

        e(n) = d(n) - y(n);

        H(:, n+1) = H(:, n) + (mu * ...
            conj(e(n)) * x_n)/(x_n'*x_n);
    end
end
```

IV. Answer to Question 4

The LMS algorithm can be used to remove unwanted noise from a desired audio track that has been corrupted during recording. However, an additional recording of the isolated noise source is needed to make it possible to apply the LMS algorithm.

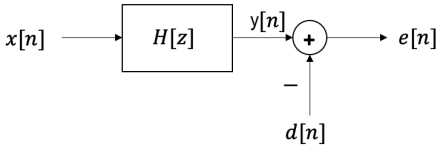
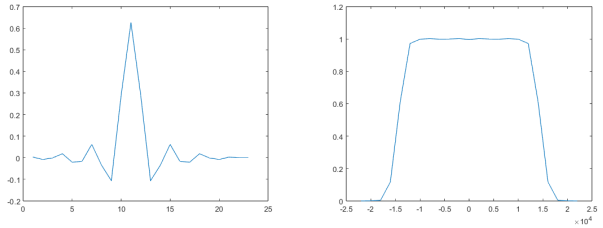


Fig. 2: Active noise cancellation

The input signal in Figure is the external noise, which we will assume is additive white Gaussian noise (AWGN), $x(n) = w(n)$. The reference signal is the recording of the tuning plus the noise captured through the wall's filtering effect, $d(n) = s(n) + w(n)$. The goal of the LMS algorithm is to find the coefficients $\mathbf{h}_n$ such that the output $y(n)$ is as close as possible to the colored noise picked up by the internal microphone, provided by $x(n)$. By subtracting the estimated noise from $d(n)$, the resulting signal $e(n)$ will contain only the desired recording as long as the LMS algorithm is properly implemented and the μ is correctly chosen, for our case β. With the following Matlab code, we are able to listen to a song with little AWGN

```
[H, y, e] = LMS_algorithm(referencia, ...
    entrada,  23, 0.2);
audio = audioplayer(e, Fs);
play(audio);
```

V. Answer to Question 5



(a) Time domain filter (b) Frecuency domain filter

Fig. 3: Filter representation

VI. Answer to Question 6

A notch filter is a type of band-eliminating filter that is designed to remove a specific, narrow range of frequencies or a single tone, while allowing all other frequencies to pass through unchanged. It does this by creating two contiguous step bands on either side of the targeted frequency range, effectively "notching out" the targeted band. This makes the notch filter useful for removing specific frequency interference. They can be designed using either finite impulse response (FIR) filters or infinite impulse response (IIR) filters. FIR filters have the advantage of being stable. However, they often require a high order, which ends up in slow calculation and high computational cost. In contrast, IIR filters are more efficient in terms of computational cost but may be unstable and may introduce non-linear phase effects.

For our case, we will implement a second-order IIR notch filter. In an IIR filter, the outputs at that time depend on the outputs at previous times, meaning it is a recursive filter defined by the following finite difference equation (EDF)

$$\begin{aligned} y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \\ -a_1 y(n-1) - a_2 y(n-2) \end{aligned} \tag{10}$$

with a_k i b_k certain sets of coefficients where $a_0 = 1$. As such, the transfer function is,

$$H(z) = \frac{\sum_{k=0}^{B} b_k z^{-k}}{1 + \sum_{k=1}^{A} a_k z^{-k}} \tag{11}$$

where A and B are the terms that determine the number of poles and zeros in the transfer function, and the order of the filter is given by $\max\{A, B\}$.

To make the band very narrow, the zeros need to be located on the unit circumference, and the poles need to be close to the zeros, obtaining the following filter [2]

$$H_{int}(z) = K \frac{1 - 2\cos(\omega_{int}) z^{-1} + z^{-2}}{1 - 2r\cos(\omega_{int}) z^{-1} + r^2 z^{-2}} \tag{12}$$

Where ω_{int} is the normalized angular frequency in radians that represents the spoofing frequency to be eliminated, the parameter r allows you to adjust the rejection bandwidth, and K is a scale factor intended to make the filter gain unitary. As r approaches 1, the rejection band becomes narrower, but the output signal quality may degrade. In our case, it is necessary for r to be in the interval $0.975 < r < 1$ in order to correctly remove the interference frequency without eliminating the frequency of interest.

```
r = 0.99;              w = 0.214*2*pi;
b = [1 -2*cos(w) 1]; a = [1 -2*r*cos(w) r^2];
```

In order to represent the notch filter in the frequency domain, we must substitute z with $e^{j\omega}$.

```
w = 0:0.004:2*pi; N = length(w);
down = a(1) + a(2) .* exp(-1i * w) + a(3) .* ...
     exp(-2i * w);
up = b(1) + b(2) .* exp(-1i * w) + b(3) .* ...
     exp(-2i * w);
H = up./down; f = linspace(-0.5, 0.5,N );
plot(f, fftshift(10*log10(abs(H))));
```

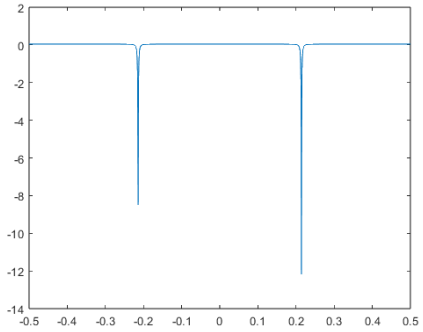


Fig. 4: Notch filter $r = 0.99$ and $w_{int} = 0.214$

VII. Answer to Question 7

```
filtered = filter(b, a, Tone_signal_noisy);
signal   = compute_periodogram(filtered);
N        = length(signal);    Fs = 1e7;
f2       = fftshift(linspace(-Fs/2, Fs/2, N));
plot(f2/Fs, signal);
```

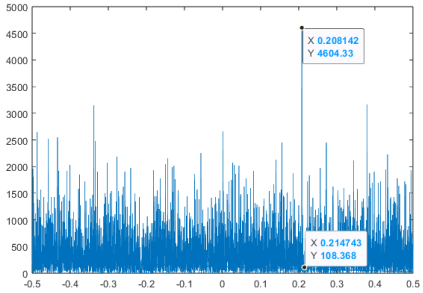


Fig. 5: Filtered signal with notch

Since it is a notch filter with a valid r value, we can see how it is able to eliminate the interference frequency without eliminating the one of interest.

VIII. Answer to Question 8

The problem of interference cancellation is characterized by having a single input, which is the received signal by the GNSS receiver with the interference to be removed, denoted as $x(n)$. The goal of the LMS algorithm in this case is to estimate the filter coefficients h so that the output signal $y(n)$ of the filter is as similar as possible to the reference signal $d(n)$, which is the interference-free signal. To achieve this, the mean squared error (MSE) between the reference signal and the filter output is minimized, that is, the MSE of the error signal $e(n)$ is minimized. It is important to note that this differs from noise cancellation, where there are two inputs, the desired signal is $e(n)$ and the signal from which noise has to be removed is $d(n)$. The following Matlab code represents the periodogram of the signal filtered with the LMS.

```
[H2, y, e] = LMS_algorithm(filtered, ...
     Tone_signal_noisy, 150, 0.087);
plot(f2/Fs, compute_periodogram(y));
```

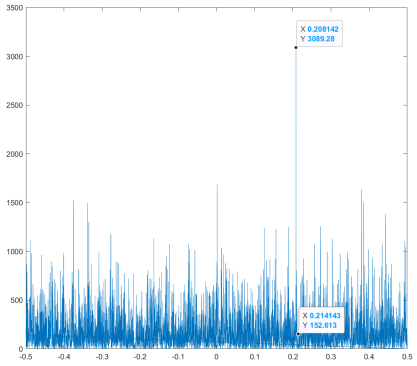


Fig. 6: Filtered signal with LMS

As it is represented, it can be said that the LMS allows the removal of interference from the GNSS receiver.

IX. Answer to Question 9

It can be seen that with a correct value of μ, the LMS algorithm achieves its purpose and estimates coefficients that generate a spectrum with two very narrow peaks at two frequencies that are removed from $x(n)$.

```
filt = compute_periodogram(H2(:, end))*n;
f3   = linspace(-0.5, 0.5, length(filt));
plot(f3, fftshift(10*log10(filt)));
```

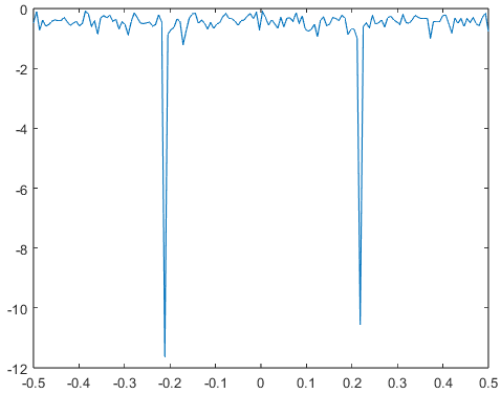


Fig. 7: Frecuency response LMS filter

X. Answer to Question 10

If we compare the notch filter with its estimation made with the LMS, we see clear differences. One of the main differences is that the LMS filter is not able to identify the frequency of the interference signal. Instead, the LMS filter increases the eliminated bandwidth, while the notch filter has a narrow bandwidth at the interference frequency. Therefore, the wider bandwidth of the LMS filter could eliminate frequencies of interest close to the interference frequency. In addition, the LMS filter does not manage to stay at 0 dB, which causes slight attenuation of the spectrum.

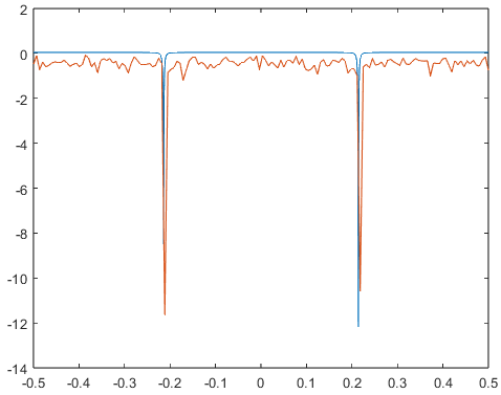


Fig. 8: Difference between notch and LMS

XI. Answer to Question 11

```
beta = [0.01 0.087 0.2];
figure;
hold on;
for b = beta
    [H, ¬, ¬] = LMS_algorithm(filtered, ...
        Tone_signal_noisy, 100, b);
    err = abs(max(H)-max(imp));
    plot(1:length(err), err);
end
legend("$\beta = " + string(beta) +"$", ...
    "Interpreter","latex");
```

As was discussed before in the report (Answer 2), we can see that choosing the μ is really important for LMS performance. Because a small μ is not enough to reach correct values and a large μ can be too large to ever reach the optimum value. So in the end, you need to estimate what would be a good μ for your problem and play with it until you get an expected result. Playing with β is similar to playing with μ, but it involves changing the interval from (4) to $0 < \beta < 2$.

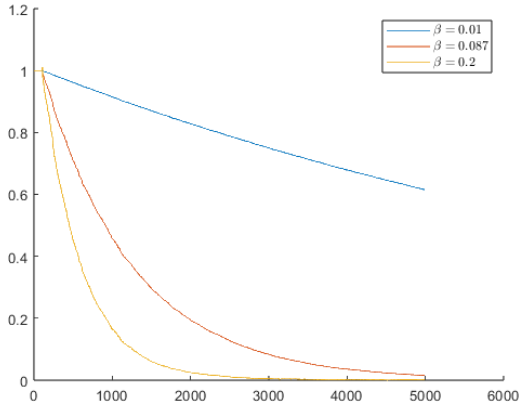


Fig. 9: Convergence behavior

Conclusion

In summary, we have seen that the LMS algorithm allows us to perform tasks such as active noise cancellation and eliminate interference from a GNSS signal as long as we correctly define the reference and input signals. We have also discussed the importance of choosing a suitable value of μ. If it is too large, the algorithm quickly converges but does not reach the optimal value. If it is too small, the algorithm will never reach the optimal value. The range has been bounded at $0 < \mu < \frac{2}{p \cdot P_x}$. Finally, we have seen that although LMS is able to perform its tasks correctly, it is not able to determine the frequency of interference but instead uses a bandwidth in which the frequency is located, which can cause problems such as eliminating a second frequency of interest.

References

[1] Hayes, Monson H., "Statistical Digital Signal Processing and Modeling", John Wiley and Sons, Inc., 1st, pp. 506-515, March 1996

[2] TI Laakso, J. Ranta, SJ Ovaska, "Design and implementation of efficient IIR notch filters with quantization errors feedback", IEEE Trans. Instrument. Meas., Vol. 43, pp. 449-456, June 1994