Adaptive Noise Attenuation

Márcia Vagos , Nuno Sousa, Sílvia Bessa , FEUP

***Abstract*— Active noise cancellation (ANC) constitutes nowadays a major field of research with a myriad of applications in acoustics and signal processing technology. Its potential for extracting signal characteristics of interest in a noisy and otherwise unintelligible content shows the importance of the development of these systems. In this work, an Adaptive Noise Attenuation application has been developed to attenuate the noise component of a signal acquired inside a box using a signal captured near the noise source as reference. The application works on the basis of adaptive filtering NLSM algorithm. The optimal adaptation step (μ) has been determined and the extent of noise attenuation has been evaluated over a variety of different scenarios. The noise attenuation was achieved in offline and real time modes using Matlab as platform. A C implementation of the real time approach has also been developed to improve the results.**

***Index Terms*—ANC, adaptive filtering, NLMS, Matlab, VAD**

I. Introduction

This work envisioned the development of a system for cancelling the noise associated with a signal of interest without altering its characteristics and the informative contents. The source of the signal of interest was considered to be located inside an enclosed box, constituting a closed system, and the noise source outside in an arbitrary location in space. Also, the noise signal and signal of interest are, for practical purposes, uncorrelated.

The idea involved the construction of a system on the basis of a microphone/loudspeaker set-up (Fig.1). A microphone was placed on the outside close to the noise source for capturing a reference uncorrelated signal, and another microphone was placed inside the box (in an arbitrary point of space) so as to acquire the signal to be processed for noise attenuation.

The original idea involved the construction of an acoustic cancelling system on the basis of the same microphone/loudspeaker set-up, but with a second loudspeaker acting as the cancelling sound field source instead of the signal of interest. The physical principal underlying this idea is sound field destructive interference, whereby a counter wave is emitted from a cancelling loudspeaker with amplitude-phase characteristic equal to the noise wave and synchronized so that the waves match with opposite phases to cancel out. Though it may look like a simple problem, in fact it poses many challenges because of the highly complex behavior of sound waves, especially in enclosed spaces. For one side, the multiple delayed reflections and reverberation phenomena cause the noise signal to be heavily distorted in the primary propagation path ($P(n)$acoustic response between the noise source and error microphone), and also make it virtually impossible to produce a cancelling wave that match this superimposed sound filed with precision. Some other considerations regarding the secondary path ($S(n)$ acoustic response between the cancelling speaker and error microphone), that filters the cancelling sound itself and the feedback path to the noise source aggravate even more the scenario to the point where a system of this nature requires high level of expertise to be accomplished. For further reference on this problematic and the studies carried out to evaluate the issue can be addressed in the website of this project mentioned in the Appendix.

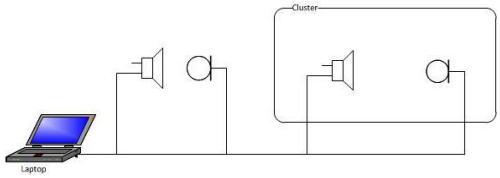


Fig. 1. Apparatus of the physical implementation of the system.

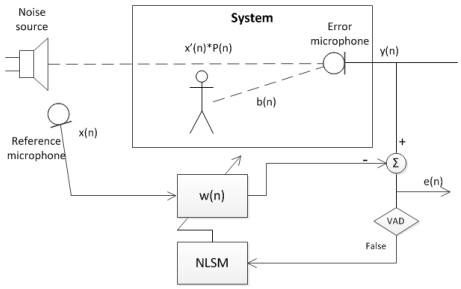


Fig. 2. Block diagram of the system.

II. Materials

Based on the physical implementation presented in Fig. 1, a HP Pavillion Dv6590 was used as laptop and the code was developed in a 32 bit Matlab 7.12.0 (Mathworks). The conversation of the Matlab routines into C language was performed in Arch Linux using the ALSA (Advanced Linux Sound Architecture). The loudspeakers used were a 2.1 Primax stereo sound system (90W) and the microphones were a Sony F-V410 and an unidirectional Philips SBC3030. An external USB 7.1 sound card (Sweex) was used because the set-up uses two microphones and two speakers all working at the same time and the laptop sound card is not prepared for it. The box was constructed by the group and the procedure and materials used are described in Appendix I.

III. Methods

This work involved several phases, including the box construction (described in the website) and the Matlab Simulation (Appendix), followed by the simulation with real signals in which the microphones were tested, the optimal filter parameters were estimated and a statistical analysis was

performed. Finally, a real-time approach was implemented first in Matlab followed by its conversation to C language.

A. Adaptive Filtering Algorithm

The approach followed down to achieve the desired attenuation was adaptive filtering. This powerful technique is already well established in both research and industrial contexts for its high quality results and application versatility. For reference, the method involves passing the signal with the noise to be cancelled through a filter whose coefficients are dynamically updated in an iterative processing scheme so as to adapt it to an ideal hypothetical solution at each iteration. The coefficients are updated by defining some error signal and determining the coefficients of the filter that minimize the energy of that error signal. This way, the algorithm converges to the ideal solution over a time span, eventually producing the desired output. One of the main solutions for this problem is the so called LMS algorithm, or otherwise NLMS, which a variant that bares some advantages to be explored later. Depending on the configuration of the microphone/loudspeaker and the parameters known to the system, the algorithm can be implemented in various ways and produce different results.

B. Attenuation in offline mode with real signals

In this particular study case, the scenario (shown in Fig. 3) evidences that the variables available are simply the two signals captured with the microphones.

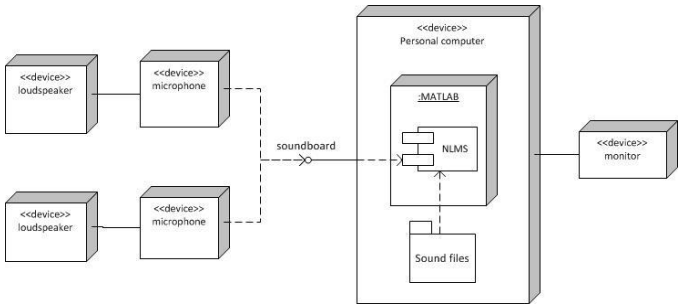


Fig. 3. Deployment diagram of the active noise cancellation system.

With these two signals, the algorithm must be configured to output one signal that is virtually the subtraction of them. It was expected to encounter extra difficulties in obtaining a good convergence and noise attenuation derived from the fact that the real $P(n)$ and $S(n)$ are indeed time varying responses in contrast to the steady simulated responses previously used. The time variations of the system conditions can have various causes such as temperature and atmospheric pressure fluctuations. Moreover, the equipment itself has inherent nonlinearities and analog/digital conversion associated errors that further aggravate this situation.

Since the filter is in fact modeling the *impulse response*(attempt on Fig. 2 for better understand of the notation) $P(n)$ (the path between the noise source and the inside microphone) from the noise, $w(n)$ is dependent of the nature of that noise, whether it is music or white noise or speech, the low pass characteristics, as well as the cross correlation with $b(n)$ can influence results. In this sense, the filter $w(n)$ ought to adapt only when $b(n)$ (the signal to be extracted) is virtually zero, allowing it to converge to $P(n)$ to reach the ideal solution. Then, it is ready to cancel out the noise coming with $b(n)$ whatever it might be, assuming $x(n)$ and $b(n)$ are non-correlated. This works well supposing that $P(n)$ does not significantly change between consecutive adaptations. Stopping the adaptation process once the filter has reached the ideal solution prevents it from diverging in the presence of the disturbing signal $b(n)$.

Bearing this idea in mind, the noise attenuation was carried out in a two module scheme, simulating a perfect VAD. In the first stage, $w(n)$ was allowed to adapt to $P(n)$ setting $b(n) = 0$. In the second stage, a series of noise samples were introduced along with $b(n)$ (the same for all the tests) and the inside microphone signal was filtered in real time with this estimation of $w(n)$.

Have seen that, this task was divided into three subtasks: firstly, the microphones were tested and calibrated (appendix III.1), then the optimal μ (a filter parameter) was determined using an automated routine with white noise and classical music as noise signals, with μ varying within the range 0.1-1.0 and finally, the system's efficiency was determined by performing a quantitative evaluation of the results using the predetermined optimal μ and a specific music playlist.

The results were the quantitative analyzed considering a measure for the degree of noise attenuation. A measure frequently used is the ERLE coefficient, defined as below:

$$ERLE = 20log_{10}\frac{E\{y\}}{E\{e\}} \quad (1)$$

which should be a positive value when there is noise attenuation, or zero when there is not. ERLE is a value determined for each sample of the captured signals and thus provides a mean for assessing the evolution of attenuation degree throughout the cancelling process. The final ERLE values were used for comparison purposes between the different noise clips.

The box attenuation coefficient (BAC) was also determined using a similar definition, with $b(n) = 0$:

$$BAC = 20log_{10}\frac{E\{x\}}{E\{y\}} \quad (2)$$

C. Cancellation in Real Time mode

The next challenge was to transpose the implementation to a real time application, that is, performing the filtering of the signal from the interior microphone with an estimative of $w(n)$ while the signal is being captured. This naturally carries some technical considerations regarding parallel programming knowledge in programming with Matlab.

One important consideration regarding this point is the concept of real time acquisition. Being a Matlab application, a true real time processing is never perfect, since the triggering can only occur when some input buffer has been received from the sound board. Plus, there is also a time lapse caused by the start acquisition process, so all these delays make the output delayed. Fig. 4. presents the activity diagram which illustrates the parallel methodology used. Equations (3)-(6) define the referred times.

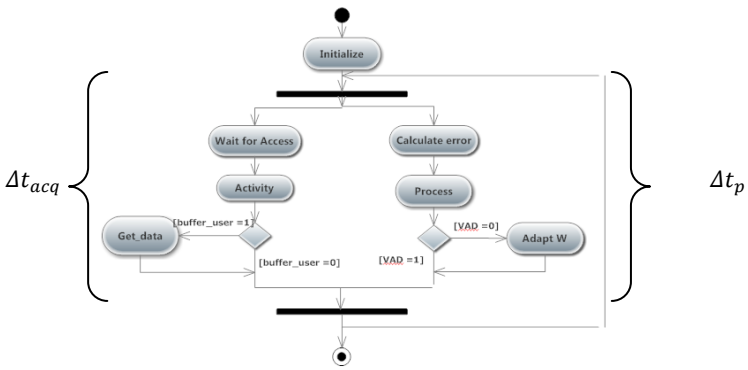


Fig. 4. Activity diagram of the real time mode.

$$\Delta t_p < \Delta tacq \text{ and } \Delta t_p = K \times Ns \quad (3)(4)$$

$$\Delta t_{acq} = \frac{Ns}{Fs} + Q \quad (5)$$

$$Fs < \frac{Ns}{\Delta t_p - Q} = \frac{Ns}{KNs - Q} \quad (6)$$

where Δt_p is the processing time, Δt_{acq} is the acquisition time, Ns is number of samples in the input buffer. K is a constant determined by the computation complexity and the CPU processing power. It was assumed to be kept unchanged. Q is also a constant defined as the time lapse of the start acquisition process and can determined offline in a pre-phase.

Since Matlab imposes a minimum sampling frequency and the constant K is very big, the theoretical maximum sampling frequency is actually lower than the minimum frequency that can be used in the input, which means, processing always takes more time than acquisition. This implies that some sound frames are dropped, and the signal is not perfectly sampled and processed. For instance: processing 0.1s at 10 KHz of sampling takes approximately 0.8s for a 1000 order filter.

Although Matlab has a large pack of parallel programming tools which implement hardware semaphores and thread management to synchronize threads (in this case processing and acquisition), unfortunately that toolbox was not available, and threading was done using global variables for semaphores which revealed plenty of technical problems in the propagation of scope of variables. Eventually minor hacks enabled such method to work.

Should processing and acquisition be synced and the Voice Activity Detector (VAD) work optimally, the results are expected to be exactly equal to those with offline mode (post acquisition processing since the signals are the same, only the time at which processing occurs changes).

D. C implementation:

The need for a dedicated binary routine which performs the given tasks became obvious when the real time performance with Matlab was known. Windows itself introduces many delays, though it facilitates direct access to the hardware which enables serious producers to create good audio applications. However, Linux provided a much more suitable environment to implement the native code. Its Advanced Sound Architecture allows for easy direct access to the hardware without having to circumvent any of the hardware managing programs that are present in windows. The advantage of binary compiled applications over interpreted code is well-known and ultimately led us to try using the lower level languages. C was the obvious choice for its ubiquity in resources examples and portability in Linux.

The choice of algorithms in the real time implementation allowed for direct conversion between the Matlab algorithm and the binary implementation. The idiomatic paradigms of C also make it much easier to optimize certain aspects of the program, such as calculating signal energy (adding and removing squared samples to an energy variable). The optimizations and the velocity of running binaries allowed for the port to directly work without losing samples over time.

E. VAD

An implementation of a VAD system was devised for the sake of testing the adaptation/cancellation in a single module scheme, as opposed to was had been done before. The basic idea consisted in performing the cross correlation between some x and y samples previously obtained, and determining the value of the peaks in the graph corresponding to instants where the baby sound is almost inaudible, which then constitutes the threshold for turning on the adaptation process. These thresholds would then be set "manually" in the algorithm so that each time the cross correlation reached such value the VAD would be triggered.

In truth, the design of a VAD algorithm is very hard, since the complexity of the mathematical/signal processing considerations make the development of a fully-fledged VAD well above the scope of this project. It should also be noted that the sheer complexity of the already developed algorithms make the VDA implementation unsuitable for our live time applications (which are the ones VAD is to be used on). There are little resources available for continuous light implementations of VAD systems.

IV. Results And Discussion

A. Cancellation in offline mode with real signals

When it comes to testing the algorithm with real signals acquired using the microphones, some degradation of the performance was noticeable. One serious problem encountered when acquiring real signals was that they were highly attenuated and distorted as well as low pass filtered, especially inside the box, as already mentioned. This posed some practical issues on properly assessing the filter's efficiency. At first it was somehow challenging to perceive the differences between the interest sound the filter outputted (error signal) and the recorded signal, yet with the optimization of the system the results are much more palpable, being that now there is a perceived qualitative attenuation of the noise.

Considering what has been explained concerning the filtering process, it is worth to notice that in practice, however, the filter can only approximate to the ideal solution, thus some noise will be constantly present in the noise-cancelled signal, though with acceptable attenuation. The system can only work as long as $P(n)$ is kept unchanged throughput the process; otherwise the adapted w is constantly deviating from the ideal solution and little or no noise is canceled at all. Note that in this work $P(n)$ was assumed to be constant during the periods where $b(n)$ is not zero and w is not adapting, because some conditioning

measures were taken, namely keeping the loudspeaker/microphone set-up in fixed positions.

B. *Optimal parameters*

Attending to Fig. 5 it can be concluded that the system works optimally with μ=0.1, seeing as these values corresponded to the greatest amount of noise cancellation. As μ approaches unity the amount of attenuation tends to zero for the white noise and slightly decreases for the Beethoven sample.

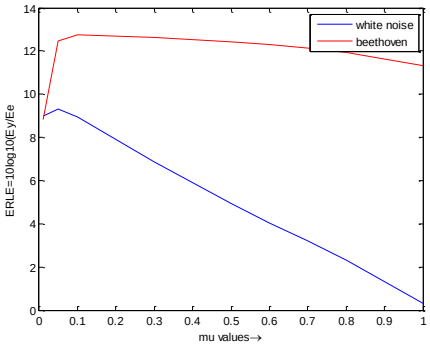


Fig. 5. ERLE values versus μ (range 0.1-1.0) with increments of 0.1.

C. *Noise cancellation Analysis*

In Fig. II4 and II5 are presented the converged $w(n)$ in offline mode, used to perform the noise attenuation stage with each music of the playlist. Note the similarity to the filter used in the simulation for modeling $P(n)$ in Fig.I2. This response is somewhat according to what could be expected regarding the effect of the box in the sound waves.

In Fig. 6 and Table I the results of the conducted test battery of musics as noise vs the sound of a baby crying are shown. The results are unanimous in showing a clear attenuation further the natural one provided by the box. With an average 17dB attenuation (-17dB gain) between the filtered signal and the signal with the baby crying plus the noise. Qualitatively the baby can be heard crying much better in the filtered signal, in some cases (such as very strong white noise) the baby cannot be heard at all in the original signal, and can be heard very well as the main source of sound in the filtered signal, proving the effectiveness of the system.

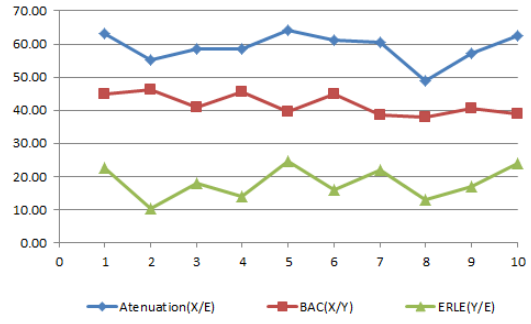


Fig. 6. ERLE final values obtained in offline mode for each of the twelve music clips in the playlist using parameters μ=0.1, lw=1000, Fs=10kHz and a duration of 40 seconds.

Table 1. ERLE an BAC values of the graphic in Fig. 5

	b(n)≠0		*b(n)=0*
Music Reference	$20\log_{10}$(E/X)	ERLE (Y/E)	BAC (Y/X)
acdc	63.16	22.53	44.81
apocalyptica	55.20	10.33	46.19
epocha	58.36	18.11	40.89
hallowed	58.54	14.09	45.64
hendrix	64.22	24.77	39.56
iron	61.34	16.06	44.79
moonspell	60.39	21.89	38.73
overture	48.87	12.88	37.76
kings of lion	57.34	16.95	40.62
skunk	62.52	23.85	38.92
teardrop	53.64	13.71	41.22
wonderful	63.16	22.53	44.81
Average	58.51	17.74	41.74
σ	4.59	4.89	3.06

D. *Cancellation in Real Time mode*

As a fully functional system required the sound of interest inside the box to be audible in real time, the algorithm was adapted so as to allow for parallel acquisition and processing of the input signals. The real time adaptative filter in matlab converged to a solution somewhat similar to the one in offline mode, however, the effect of a lossy adaptation is evident the steady behavior of the filter, especially in the later samples.

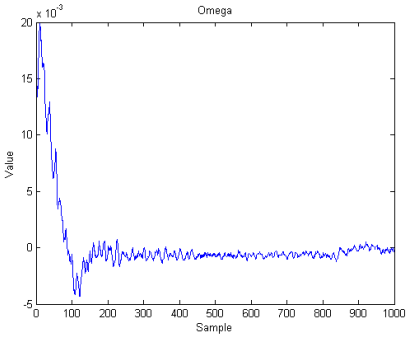


Fig. 7. Real time w: Note that the peak corresponds roughly to the distance between the microphones.

V. Conclusion

The real world technical problems associated with audio applications make the seemingly trivial algorithms of simulated work very complicated. The gratification when the said algorithm work in the real world far exceeds simulation as well. Our initial goals of 3dB attenuation were overcome, and the actual effectiveness of this system was well above what we thought we might get in the real world. Despite our best efforts the VAD detector didn't work as expected, compromising the live time implementation. Given the right tools for handling audio this system could work very well and very fast indeed.

Appendix

Further images and code routines, can be found at http://paginas.fe.up.pt/~bio07040/, projects section, LIEBIII.

Acknowledgment

The authors gratefully acknowledge Professor Bruno Bispo for the orientation in this project.

APPENDIX I: MATLAB SIMULATION

In this phase NLMS algorithm was implemented using simulated signals and a model of the system to test its adaptation behavior and determine the optimal parameters as well as testing the influence of the nature of the signals employed.

The two signals can be described by the following equations:

$$xs(n) = x(n) * S(n) \quad (1)$$

$$y(n) = x(n) * P(n) + b(n) \quad (2)$$

where $P(n)$ models the impulse response of the primary path, and $S(n)$ models some secondary path from the noise source to the reference microphone. Naturally, these functions are unknown and thus the algorithm must be independent of their values. $xs(n)$ is the sound field output by the noise source, without any distortions due to distance, echo and reverberation phenomena. Note that $P(n)$ is a function that models the path all the way from speaker to error microphone, which includes delays due to propagation distance, reflections and the impedance to high frequencies provided by the box. Similarly, $S(n)$ models the distortion of the sound field from speaker to reference microphone, which is much less severe than that occurring in the primary path. For simulation purposes, $S(n)$ was modeled with three different functions: as a unity transfer function ($S(n) = 1$), as a simple delay ($S(n) = \Delta$), and as a train of delays in much the same way as P(n) was modeled in the simulation phase. This three functions were considered for $S(n)$ in order to assess the influence of this parameter in the outcome. Note that in the two last cases it was necessary to guarantee that the longest delayed pulse is not greater than the time propagation of sound in the primary path.

The governing equations are the following:

Error signal:

$$e(n) = y(n) - xs(n) = x(n) * P(n) + b(n) - x(n) * S(n) \quad (3)$$

Filter adaptation coefficients:

$$w(n+1) = w(n) + \frac{\mu}{y(n)^T \cdot y(n) + \delta} e(n) x(n - N + 1 \rightarrow n) \quad (4)$$

The acoustic impedance of the walls of the box posed a natural low pass filtering response to the incoming sound waves, thus preventing the noise high frequency contents from reaching the inside microphone and disturbing the crying sound.

An offline adaptive filtering application was developed in Matlab implementing the well-known NLSM algorithm for its good performance in general purpose noise cancelling. The signals used to simulate the inside microphone were a clip of a Mozart composition and a baby carrying sound. The corresponding waveforms are shown in Fig.I1. The outside microphone was simulated with white noise.

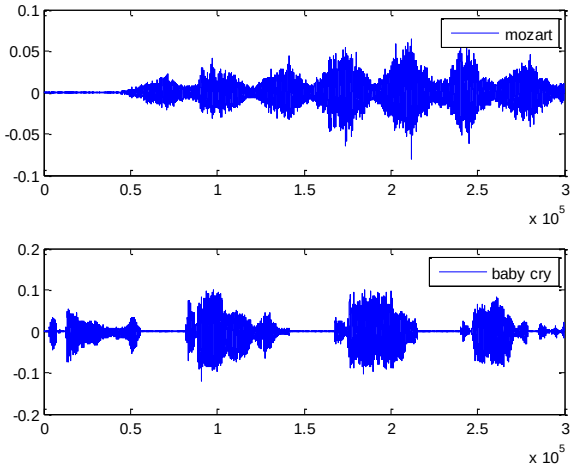


Fig. I1. Signals used to simulated the sound captured by the inside microphone.

Table I1. ERLE values of the final outputs obtained in each scenario (×1.0e+006).

	Mozart	Baby crying
S1(n)	6.0556	5.8474
S2(n)	6.0887	5.8302
S3(n)	0.1290	-0.0986

The secondary path $S(n)$ was modeled with three different functions:

$$S1(n) = 1$$
$$S2(n) = \Delta$$
$$S3(n) = train\ of\ decaying\ delays$$

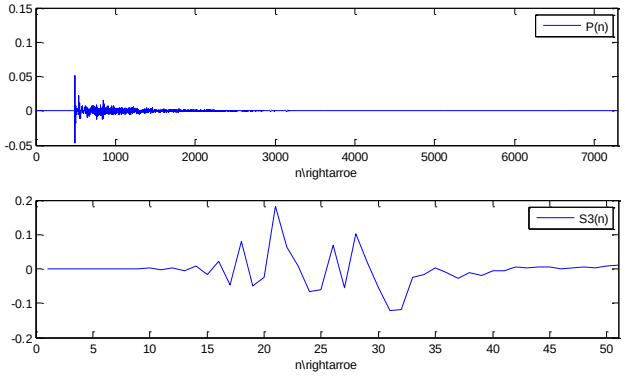


Fig. I2. Filters used to model the $P(n)$ (upper) and $S3(n)$ (lower) in the simulation.

The output obtained with the "Mozart" signal and using $S2(n)$, along with the original signal, is presented in Fig. II3 below. From this figure, it can be seen that the resemblance between the original crying signal and the output is strikingly visible, which shows that the filter managed to adapt very well over a reduced number of samples. Apart from some delay caused by propagation distance in the filters used to mode$P(n)$and $S(n)$some high frequency attenuation, the result was fully satisfactory. The error signals can be seen to have converged over time and the ERLE values exhibited an increasing behaviour towards the final samples (see Figs. I4 and I6).

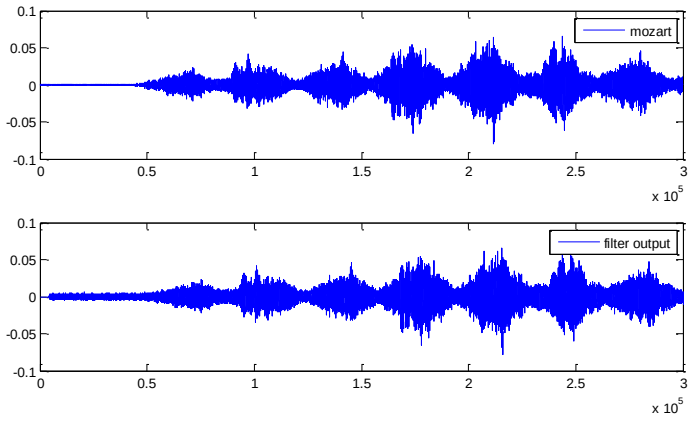


Fig. I3. “Mozart” signal before (upper) and after (lower) noise cancellation with w(n).

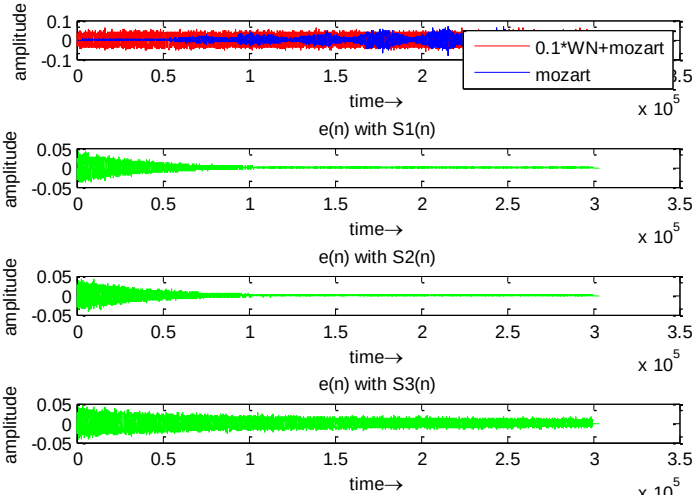


Fig. I4. Error signals from the Matlab Simulation with “Mozart”, using μ=0.1, lw=4000.

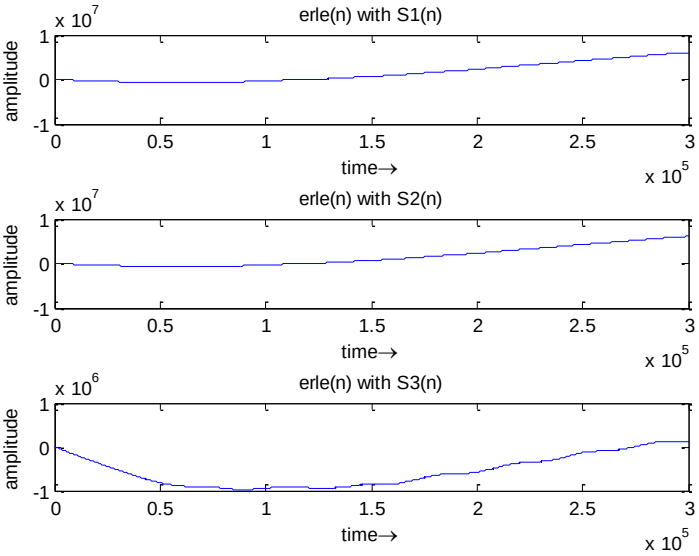


Fig. I5. ERLE values throughout the processing with w(n), using “Mozart” inside signal.

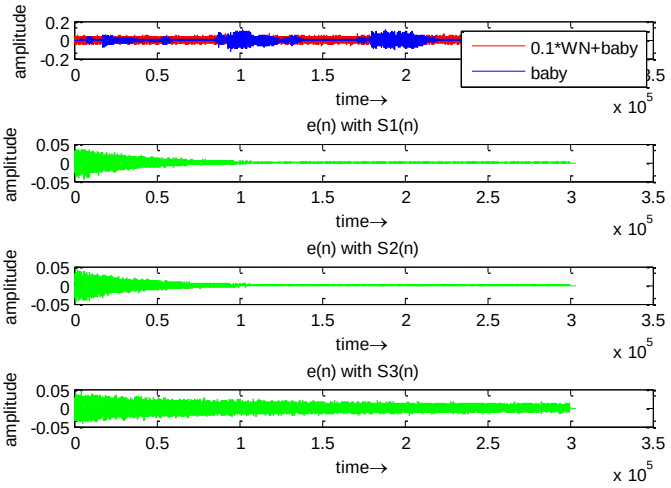


Fig. I6. Error signals from the Matlab Simulation with a baby crying cound, using μ=0.1, lw=4000.

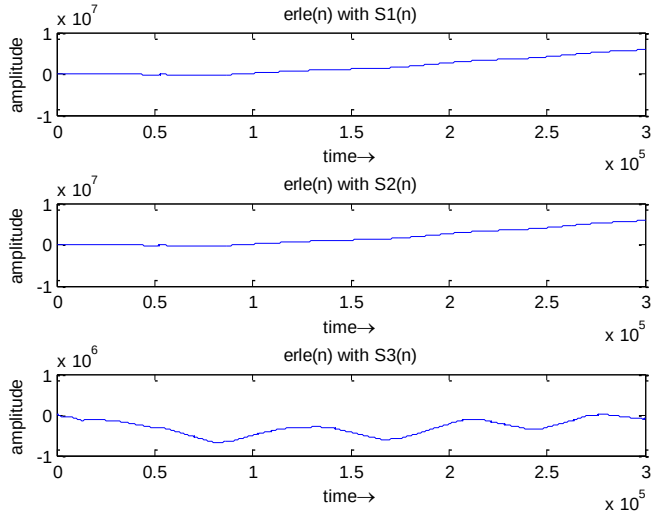


Fig. I7. ERLE values throughout the processing with w(n), using “Mozart” inside signal.

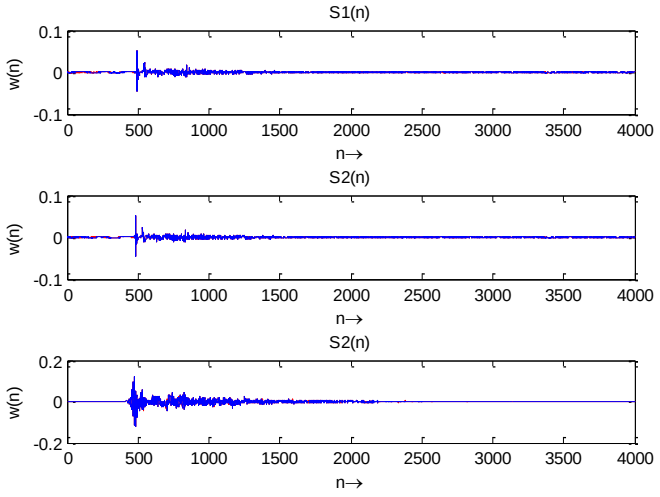


Fig. I8. w(n) vectors of the simulations with either “Mozart” and baby crying signals (superimposed).

Note that the results in both cases were closely similar. The ERLE values seemed to be better for the case where $S1(n)$ or $S2(n)$were used, as would be expect. When the secondary path was modeled with $S3(n)$, the results were not so good. Not only the error did not manage to adapt so well, as can be seen from the larger error, but also the ERLE vector assumed negative values, which is somewhat unexpected. As a matter of fact, it was not surprising that the result was worse for the case where we used $S3(n)$, which introduces some instability in the convergence of $w(n)$, but still the reason for such low ERLE values is not straightforward.
Considering the $w(n)$ vectors in Fig. II8, the fact that for each case the filters resulting from both simulations are completely superimposed evidences the algorithm's capability of adapting to the primary path impulse response regardless of the nature of the inside sound.

The algorithm's performance was also studied with different noise signals for the outside microphone, namely white noise and classical music and in both situations the outcome was a clean undistorted sound. It could be noted that the initial instants were still affected by some noise with decaying magnitude as the filter actively adapted to the incoming sample.

The algorithm and values of the parameters used in the simulation can be consulted in website aforementioned.

APPENDIX II: SIMULATION WITH REAL-SIGNALS

II.1. Microphone Calibration

Before acquiring the real signals for the simulation with real signals, the microphones were tested by capturing a live played signal from an electric guitar and varying their distance to the signal source. The acquired signals are presented in Fig. II1. Although the signal from the Philips microphone had a higher value of offset, an offset can be recognized for both microphones. In order to minimize the offset effect, the mean of each acquired signal was assessed and subtracted to the original one. Fig.II2 presents the signals after this normalization, evidencing that the signals from both microphones have the same amplitude and phase which is desirable for this work.

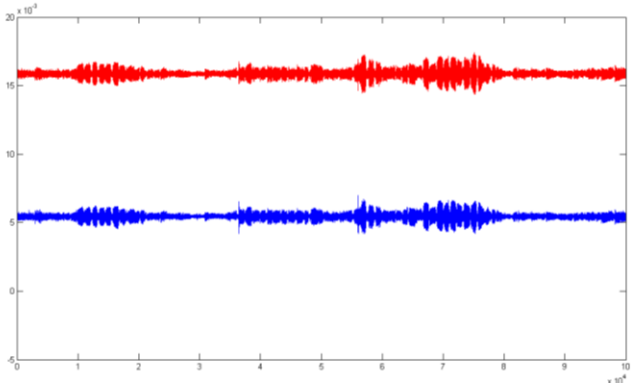


Fig. II1. Calibration signals captured by Philips microphone (red signal) and Sony microphone (blue signal).

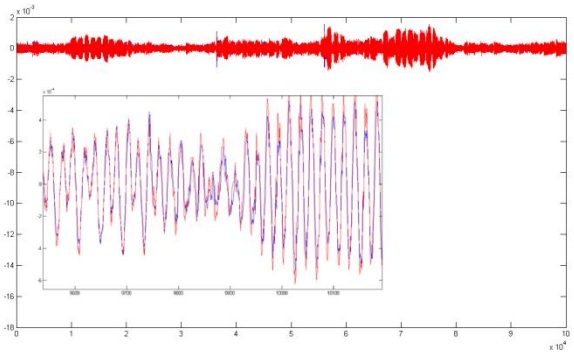


Fig.II 2. Calibration signals captured by Sony microphone (blue) and Philips microphone (red) normalized by mean.

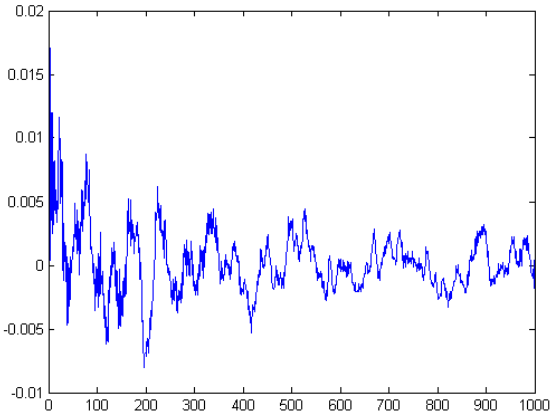
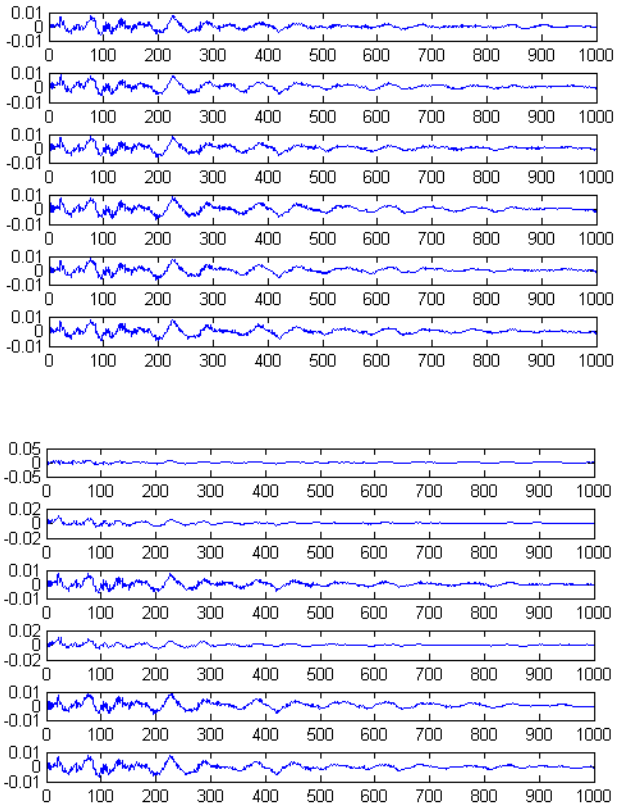


Fig. II3. Converged $w(n)$ in offline mode.

II.1. Microphone Calibration

The $w(n)$ filters obtained in the real time mode for the playlist (in the same order) are presented in Figs. IIV3 and IV3 below.



Figs. II4 and II5. Resultant filter for each music of the playlist. The filters are presented in the same order specified in Table I.

APPENDIX D: REAL-TIME MODE

The algorithm implemented in Mattlab previously referred to in the real time cancellation section can be described by the diagram presented in Fig. IV1.

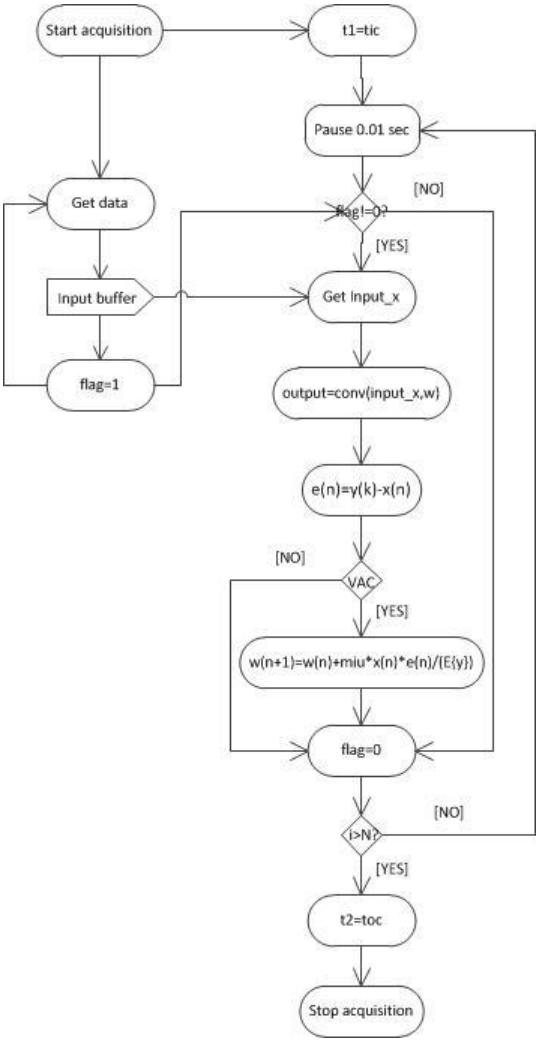


Fig. III1. Flowchart of the real time mode.