EIT DEPARTMENT, LUND TEKNISKA HOGSKOLA

ETIN80 Algorithms in Signal Processors: Adaptive Line Enhancer

By: Kevin Skaria Chacko, Roshan Cherian
Instructor: Mikael Swartling
March 6, 2018

1 INTRODUCTION

The purpose of the project is to implement an Adaptive Line Enhancer (ALE) on the DSP. The goal of the ALE is to remove any tonal components from a signal by using adaptive filtering. This can be done by using a LMS algorithm, which predicts the filter weights for a Wiener filter.

This implementation can be used in noisy environments (industrial and military) where you want to suppress tonal components (e.g. sound from horns or industrial equipment) but retain the speech. Another application of this could be in sound recording, to get rid off the noise. In a real life situation it is most likely that the disturbing tonal signal is not known. Therefore the ALE needs an adaptive filter to predict what signal to be removed.

Figure 1 shows the block diagram of the LMS algorithmic implementation using the Wiener filter as described above. The input signal is delayed and sent through the LMS algorithm and then compared to the non delayed input signal. This process is repeated until the optimal filter coefficients are found. The delay needs to be so adjusted so as to make all the uncorrelated parts of the signal pass through unaltered and all the correlated parts to be suppressed.

The algorithm was first needed to be implemented on matlab and then subsequently needed to be coded for the DSP. Implementation on the DSP was the critical part of this project since algorithm implementation in matlab is not real time as compared to the DSP. That is the computation in a DSP needs to be performed before the occurrence of the next sample. Implementing the algorithm by keeping these considerations is the key to a efficient implementation

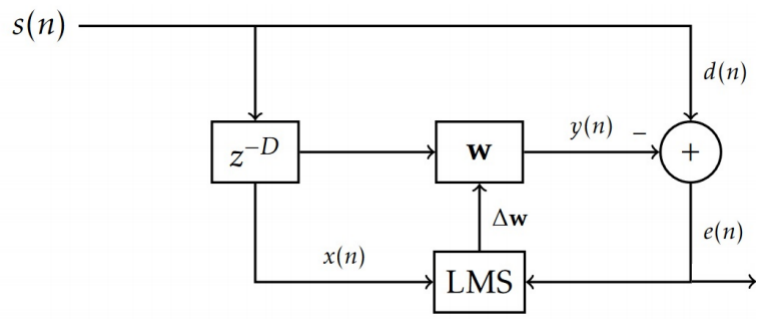


Figure 1.1: Block Diagram of ALE

of the the algorithm on the DSP. The memory limit on the DSP puts further restriction on the implementation and forces to take care of the coding style.

New problems might arise when implementing an algorithm on a DSP versus matlab as the DSP has real-time requirements that the architecture of the DSP has to account for. This mean that the computation must be completed before the next sample occurs. Choosing the right algorithm and implementing it efficiently is the key to meet the computational requirements. The DSP is also limited in memory which means that it is important handle data correctly and efficiently.

2 Hardware Setup

The DSP selected for this project was an Analog Devices ADSP-21262. The code is written in C and compiled using Visual DPS 5.1. The hardware setup consists of a line in to the tonal sound source, a headphone out(or speaker out), four buttons (on board). The 4 switch buttons are used for the following tasks:

- Switch to tonal input source
- Switch to filtered ALE output source
- Reset filter coefficients with a default predefined step size
- Reset filter coefficients with a smaller predefined step size (for realizing the working of the filter)

The image of the complete setup is given in figure 2

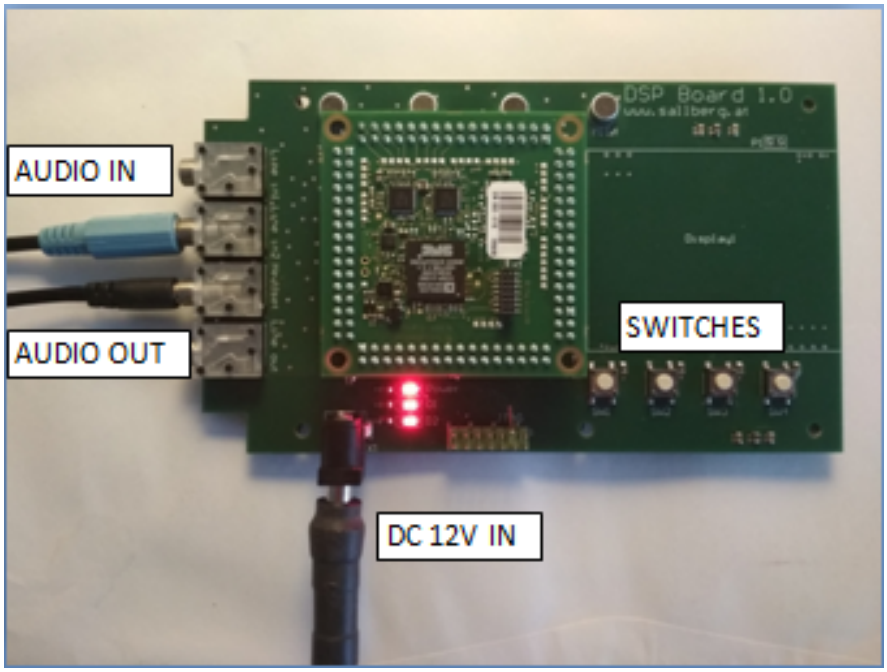


Figure 2.1: Hardware Setup for ALE based tone eliminator

3 Theory on ALE, Wiener filter, LMS

The Adaptive Line Enhancer (ALE) is an effective learning filter for reducing Gaussian noise with a large SNR. The filter adjusts the filter weights to pass the desired input signal while reducing the noise portion of the signal with little to no filter roll-off up to the Nyquist rate (Fs/2). An adaptive filter can alter its own frequency response in order to improve the filterâĂŹs performance on-the-fly.

The Wiener filter is used because since there is an input signal and a desired signal and the data regarding their occurrences are not known. It is based on the Steepest Decent concept where the filter converges to the optimal solution. The following figure explains the working of the Wiener filter.

Least mean squares (LMS) is class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired and the actual signal). The filter coefficients wo = [w1, ..., wn] are tuned such that they minimize the error e(n) in the mean squared sense.

The purpose of the LMS algorithm is to find the near optimal filter coefficients for a given input signal. The LMS method is based on the method of the steepest descent with the difference that the coefficients are continuously estimated. This feature of the LMS algorithm makes it adaptive and suitable for input signals that change over time.The difference between

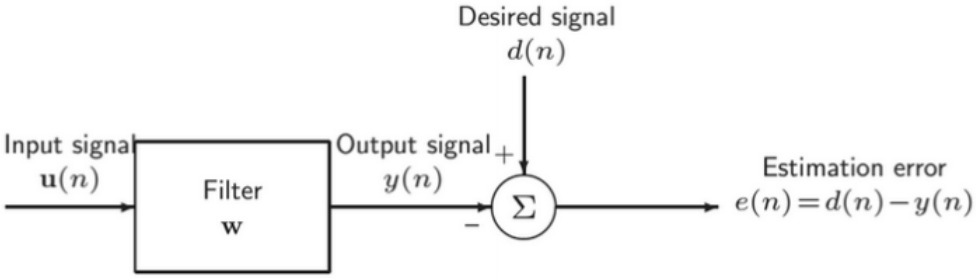
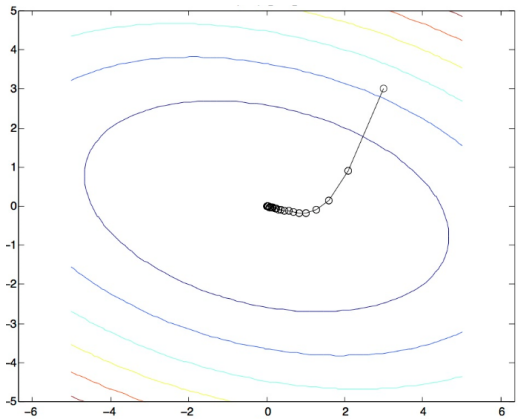
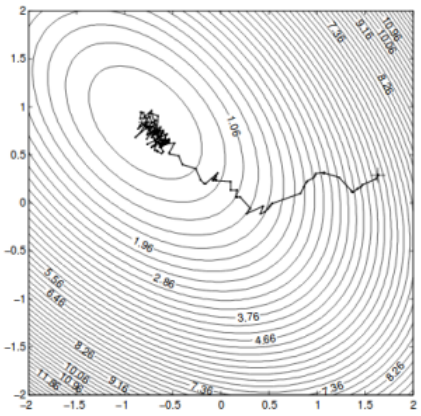


Figure 3.1: Wiener filter

the Wiener filter and the LMS algorithm can be seen in figure 3 where the Wiener filter using steepest descent converges to the optimal solution, but the LMS has a constant error and never fully reaches the optimum.



(a) Convergence of Steepest D.



(b) Convergence of the LMS

One problem with the standard LMS algorithm is that the step size is fixed. This creates problems when,for example, the amplitude of the signal is altered. If this is the case the algorithm might not converge in sufficient time, or diverge. To safeguard against this NLMS normalizes the amplitude.

3.1 Implementation Steps

3.1.1 Algorithm

The algorithm was first implemented in matlab to test and verify that the algorithm was working, and the filter used gave the correct response. Using matlab is a good way to focus on the algorithm and to get it working without putting much thought about the hardware. The tonal wave files were read into matlab and played back after implementation of the filter to verify successful implementation of the filter in matlab. Once the verification was done, fine

tuning of the parameters was done till a clean sound signal free from tones was obtained from the headset.

3.1.2 Implementation on DSP

When the implementation was successful and approved, it was translated to C code. The algorithm was then integrated with the framework provided in the course and modified as per our requirements. The final tests were done by streaming music with an added/superimposed sinusoid made using a sound editing software and through the algorithm and listening to whether or not the tonal sinusoid is successfully removed. Again the parameters were modified to get the optimal response from the DSP.

3.2 Conclusions

The ALE along with the LMS was used for our implementation.It took a lot of trial and error with the step-size and delay-size to get it to sound as good as possible. The theoretical and matlab code did not fully comply with the final values we had to use for the DSP. When the C code was run on the DSP in the Debug(Default) mode, it did not give results as expected. On changing the mode to Release, the algorithm worked smoothly. This was because of the optimization done in the Release mode. The goal of the course was finally met. It gave us a better understanding of adaptive filters and LMS algorithm. Thus, the study of LMS algorithm and its application in ALE was understood and successfully implemented on the DSP.

3.3 Discussion

The LMS was successfully implemented in C and run on the DSP. Having a small step size caused the signal to converge slowly and if it was to big caused it to diverge. We observed that the step-size of 0.1 was ideal and at a step size of 0.01 for slow convergence (to show case the effect of convergence) for the DSP implementation. The improved version of LMS, the NLMS was implemented on matlab but due to time constraints were not able to implement on the DSP. We set the delay for the DSP at 400 samples. This delay had to be large enough for the music to be uncorrelated, but small enough to not use up too much memory. In contrast to matlab, we found that it was hard to hear the difference after a while when increasing delay. We used a filter size of 100. The filter was similar to the one we used in matlab and was fairly consistent.