Analysis of the normalized least mean square algorithm for ECG extraction

David Andersson

January 17, 2016

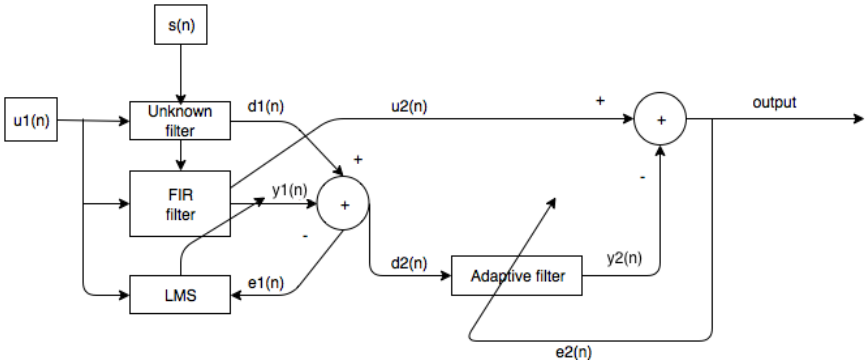

Figure 1: Block diagram of the system.

Abstract

This paper will essentially deal with the process of adaptive noise cancellation, where the noise is composed of a combination of white, Laplacian and an echo components. To make it a bit more interesting, I assume that faulty equipment distorts the signal as would it be a high pass notch filter. This will include a system identification part. To accomplish this the normalized least mean square algorithm will be used. The purpose is to explore and evaluate the algorithm within the context of the ECG extraction and optimal and adaptive signal processing.

1 Introduction

Electrocardiography (ECG) is the signal generated by heartbeats, and is a vital tool for determining a range of medical conditions. Being susceptible to a verity of noise sources it's an interesting signal to use when developing filters.

What I wanted to do was to simulate a fairly reasonable (mostly interesting) scenario that might occur when measuring the ECG signal. From what I understand there are a few things to consider: Interference from muscle activity, here simulated as Laplacian noise. The recording noise, here represented as white Gaussian noise. And also echo (reflections), here represented as a low amplitude, down sampled, delayed version of the original signal. I will also use the scenario that faulty equipment give rise to a high-pass notch filter, which needs to be compensated for. The idea is that since the signals and noise are correlated, but not something you could remove through subtraction, it will serve as an non-trivial benchmark surface for the algorithms.

The normalized least mean square (NLMS) seems a good choice since it takes in to account the power of the signal, which is obviously fluctuating when working with ECG. It also converges faster than the least square method (LMS) and converges to a more optimal solution than the leaky least mean square method (LLMS). The analysis I will preform will circle a lot around the different parameters available in the NLMS,

such as step size and filter size, in the hope of getting better acquainted with the difference between theory and practice. To visualize the problem the block diagram in figure 1 is used.

The reader is assumed to have a basic understanding of adaptive filter theory.

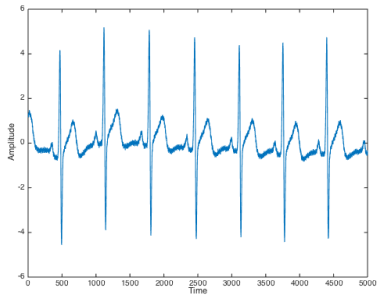

Figure 2: Original signal.

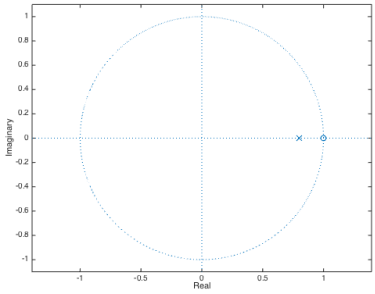

Figure 3: Z plane plot of the high pass notch filter.

2 Methods

The LMS algorithm is as stochastic gradient descent method, meaning we do not know the the statistics behind the input, in stead we estimate them and use the method of gradient descent to find the minimum of a cost function. The update rule for the NLMS which i have been using can be seen below.

$$\hat{w}(n+1) = \hat{w}(n) + \frac{\tilde{\mu}}{\delta + \|u(n))\|^2} * \mathbf{u(n)} * e^*(n) \tag{1}$$

As provided by [1, p. 337]. Where lambda is a safety term making the algorithm less likely to have a division close to zero. As mentioned there are two parts to the problem, one is the system identification, and one is the noise cancellation, which means that the NLMS algorithm is used in two ways. For the system identification I assume that a pure noise signal is available, which serves as a reference when modeling the system. That means that $u1(n)$ will be our pure noise signal, and $d1(n)$ will be the filtered noise signal. My idea was that since the noise contains all frequencies (white noise) it will serve as a good modeling tool also for the real input, the heartbeat, which will be run through the same model of the channel. The result will be a deconvoluted signal $u2(n)$ and a deconvoluted noise $d2(n)$. These will serve as the input to the next stage, which will be an adaptive noise canceler.

I got the idea from the fact that when I ran the normal noise canceler without any convolution I got a great result. But, I also found that the signal $u(n)$ could be pretty well extracted without an adaptive filter, but in stead using a smoothing filter. This is seen in figure 5. But, when I run the signal though a high pass notch filter, the output will be not be so easily extracted. This is shown in figure 4. I guess this applies to ECG signal in the sense that a lot of useful information is contained outside of the main lobe, which is not visible with smoothing of the convoluted signal.

The filter is a high pass notch filter of order one. Figure 3 contains the filter plotted in the z-plane.

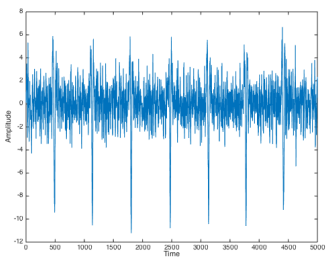

Figure 4: Convoluted signal through a smoothing filter.

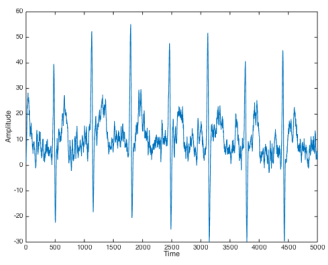

Figure 5: Original signal through a smoothing filter.

3 Results

3.1 System identification

As expected, to model an AMRA (both poles, and zeros) process with just AR taps, requires a high order. Below 20, I could not find a suitable model, and the signal ended up just being less noisy, but distorted. I got a much better result with 25 - 30 parameters. The result of a a few trials with different filter lengths can be seen in figure 7.

The idea behind the model order choice is that since we can't use MA parameters, we substitute with a higher order AR, in accordance with the equation below.

$$\begin{gathered} y_t - a_1 y_{t-1} = e_t - c_1 e_{t-1} \Leftrightarrow y_t - a_1 y_{t-1} = (1 - c_1 z^{-1}) e_t \Leftrightarrow \\ (y_t - a_1 y_{t-1})(\frac{1}{1 - c_1 z^{-1}}) = e_t \Leftrightarrow (y_t - a_1 y_{t-1}) \sum_{k=0}^{\infty} c_1^k z^{-k} = e_t \\ |c_1| \leq 1 \end{gathered} \tag{2}$$

Insipired by [2, p. 62].

So the optimal model should contain an infinite amount of AR parameters, but since that's not an option, well have to settle for less.

My initial attempts to this problem was with the LMS algorithm, but it was clear from the start that it was to difficult to tune the step size. I tried using the rule of thumb from the theory, which states that a step size should be lower than two divided by the sum of the eigenvalue, which can be approximated by the sum of the diagonal elements of the covariance matrix of the input signal. Since the covariance matrix is toeplitz structured, the rule can be summarizes as in the equation below (3). Using this I got a decent result, but I still had to tune it depending on the realization of the noise, making it useless.

$$\mu = \frac{2}{M * r(0)} \tag{3}$$

As provided by [1, p. 304].

In contrast the NLMS converged with ease, according to the theory the algorithm should converge with $0 < \tilde{\mu} < 2$. Which is a simplification of [1, p. 337]. In figure 6 we can see a few different step sizes and their corresponding mean square error corresponding to three different filter sizes. With filter sizes larger than 30, there is not much change in the curve.

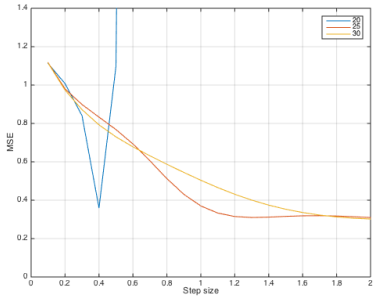

Figure 6: Step size trial.

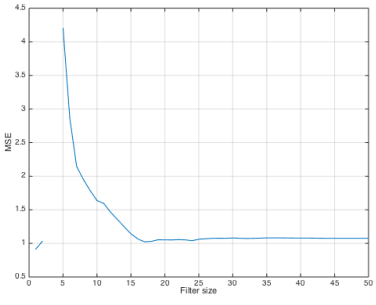

Figure 7: Filter length for deconvolution.

3.2 Noise cancellation

The noise cancellation part was very robust. To test it out I tried combinations of additive noise to make $d(n)$ and $u(n)$ more corrupted. I'm not sure what is a reasonable amount of noise, but I tried Gaussian white noise with a variance up to 2. Above that, I could not extract a useful signal with these methods. The result in mean square error can be seen in figure 8. As you can see, $u(n)$ is much less affected than $d(n)$, which soon diverges. I also tried different delays. There is a strong connection between delay and filter

length. Using a filter length which was the delay + 1 I got really good result. The upper bound as I saw it was mostly the amount of parameters that are reasonable to have in a filter, beyond that the MSE was manageable. Figure 9 shows the mean square error curve for trials with delay up to 30. As seen, there is a clear difference between zero, one and two additional parameters. Additional parameters did not give a noticeable contribution.

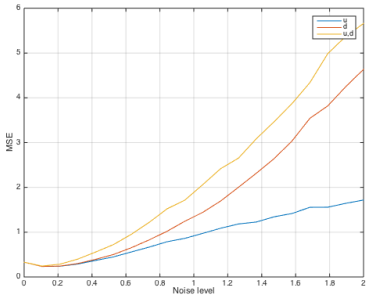

Figure 8: Mean square error for different amount of noise.

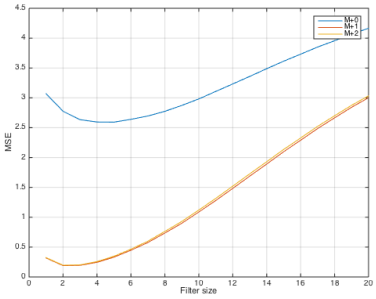

Figure 9: MSE for different delays

To get a better understanding of the convergence I looked at the eigenvalue spread of different filter sized. The theory states that the larger the eigenvalue spread, the larger the convergence rate. The eigenvalue spread is defined in the equation below.

$$X(R) = \frac{\lambda_{max}}{\lambda_{min}} \tag{4}$$

As provided by [1, p. 227].

In figure 11 we can see the increase of the spread with the filter size which is quite linear. For this application, a small filter, 3-4 taps was sufficient to filter out the noise. As seen in figure 10 the MSE does increase with the size of the filter, if the step size is not chosen correctly. For example, if we would have a filter of size two, it's error surface would take on a more and more elliptic shape as the eigenvalue-spread gets larger than 1. If the start value is not chosen very precisely, you will have a longer route to the center.

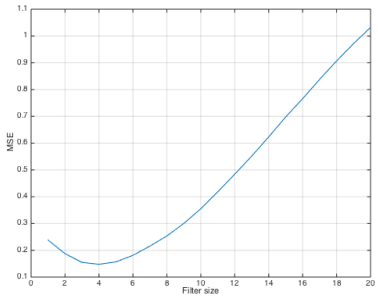

Figure 10: MSE for different spreads.

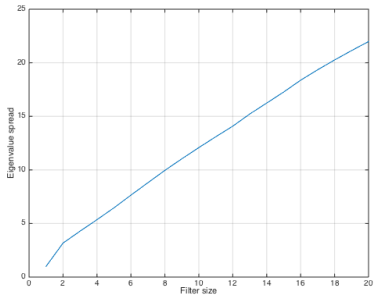

Figure 11: Eigenvalue spread.

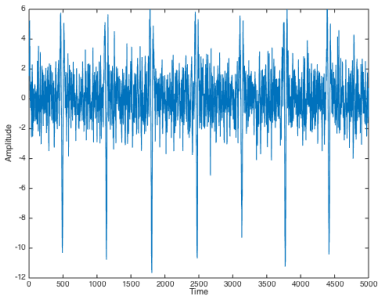

Figure 12: Input to the system.

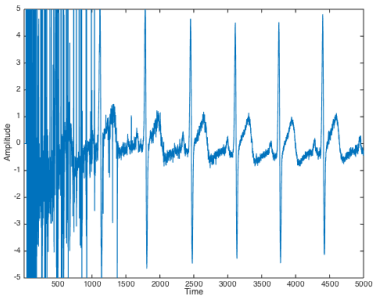

Figure 13: Result of the algorithm.

4 Discussion and Conclusions

In general, the NLMS algorithm is sub optimal to the Wiener-Hopf equations. The lower bound of the Wiener-Hopf is given by:

$$J_{min} = \sigma_d^2 - \mathbf{p}^H * \mathbf{R}^{-1} * \mathbf{p} \tag{5}$$

As provided by [1, p. 122].

And the corresponding bound given by the NLMS is:

$$J(\infty) = J_{min} + \mu J_{min} \sum_{k=1}^{M} \frac{\lambda_k}{2 - \mu\lambda_k} \tag{6}$$

As provided by [1, p. 303].

But sub optimal is a relative term. The conclusion I can draw from the scenario I set up is that the NLMS is very stable and robust. it required very little information about the signal in order to work, and the result was better than I had imagined. When comparing to the theory I found that the lower bound of the NLMS was barley noticeably increasing with the step size and eigenvalues. I guess there are applications that need larger precision, but from what I could see from the resulting plots there were very little difference as long as I did not use an unreasonably long filter and the algorithm converged properly. One example of a convergence of the algorithm is shown in figure 13, when feeding the signal in figure 12 to the system.

Of course there was a difference between the two scenarios. Not surprising the system identification was a lot more sensitive to additive noise than the noise cancellation. One needed to have a very good "desired" signal to find a suitable model.

As mentioned earlier I first did the filtering with the standard LMS algorithm, but quickly abandoned it since I had to spend a lot of time adjusting the step size. I did not include any results from these trials since they were basically identical.

When doing the trials there were some parts where I wanted to, for example, evaluate the filter size, but I also had to choose a step size. To make the trials as fair as possible I used the theoretical "rule of thumb" since it at least gave me some less biased step size than tuning it myself. In, for example, the opposite case, I used a known stable filter size to test the step size. For example 4 and 16 seemed like the optimal choices for noise filtering and identification respectively. A further study could take in to account more variations in stead of making assumptions.

One could also include trials concerning the δ in equation 1, but I found values between 100 and 1000 to give stable results so it was not very interesting in this scenario.

References

[1] Simon S Haykin. *Adaptive filter theory*. Pearson Education, 2014.

[2] Andreas Jakobsson. *An Introduction to Time Series Modeling*. Studentlitteratur, 2013.