SGN-21006 Advanced Signal Processing

Adaptive noise canceling project

Assistant: Pekka Astola

1 Project requirements

This document presents the adaptive noise canceling project that is mandatory for passing the Advanced Signal Processing course.

Altogether, there are ten points possible to get from the project. The minimum pass grade for the project is three points. The points are specified more carefully later in this document. The tasks grow in difficulty so please try to solve the bonus tasks only after solving the mandatory tasks.

Your submission should include:

- The Matlab code to produce all requested results.
- A short `pdf`-document describing the problem as you understood it. It should include the requested figures and a short comment (one paragraph) for each one of them.

Send your finished project to pekka.astola@tut.fi

Deadline: 14.12.2018 23:59

2 Introduction to Adaptive noise canceling

In this project work, we consider an adaptive filtering application known as adaptive noise canceling (ANC). ANC effectively analyzes the effects of a certain environment (a room) on the recording of audio data. The setup is presented in Fig. 1a. It can be seen that the room contains two sound sources $c(t)$ and $v(t)$. The source $c(t)$ is considered to be a clear speech sound without any perturbations. The source $v(t)$ is an unwanted sound (noise) and our goal is to remove it. The trick is, that we do not observe exactly the noise $v(t)$ since the environment corrupts the sound traveling from the source $v(t)$ by the time it is recorded as the output sequence $s(t)$. The output $s(t)$ is, thus, a superposition of the clear source $c(t)$ and the noise that passed through the room. In reality, such a scenario is present for example in our mobile phones which typically contain two microphones: one at the top observing the noise and one at the bottom observing our speech and the corrupted noise. Note that in a real application, the clear sound $c(t)$ is not available! We are, however, also using a simulated environment where $c(t)$ is available and can be used for error analysis.

Your task is to find an adaptive noise canceler, whose structure is depicted in Fig. 1b. An FIR filter with coefficients that can change in time is used to cancel the noise. For this purpose you use the algorithms NLMS and RLS.

The application is based on the important observation that the signals $v(t)$ and $c(t)$ are not correlated i.e. the speech sound is not travelling to the microphone that records the noise. When this assumption is true, we have

$$s(t) = v_0(t) + c(t), \tag{1}$$

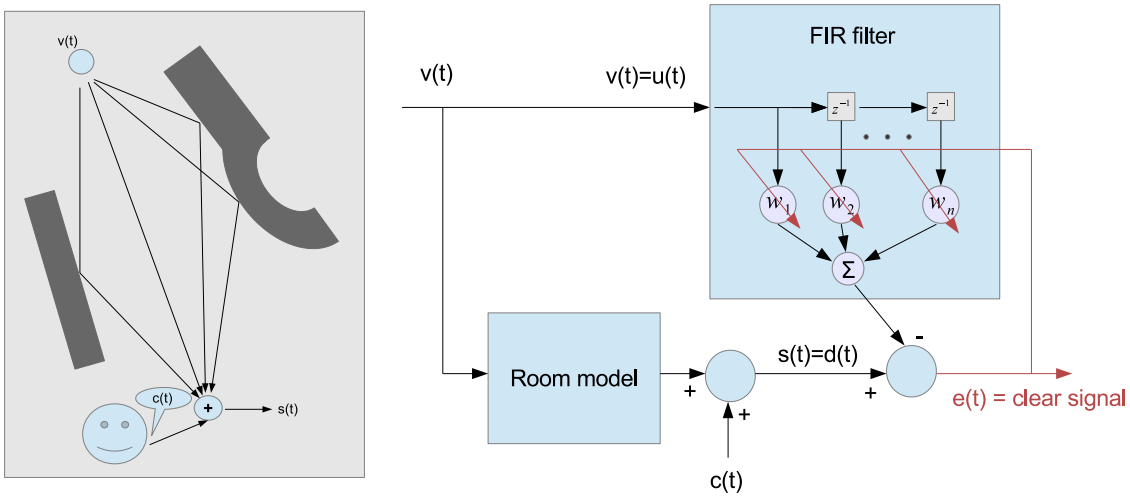
where $v_0(t)$ is the perturbed version of $v(t)$. If we compute the error we, it results that

$$e(t) = s(t) - \sum_{i=1}^{n} w_i(t)v(t-i) = c(t) + v_0(t) - \sum_{i=1}^{n} w_i(t)v(t-i). \tag{2}$$

The goal is to use an adptive algorithm to minimize the square error. This translates to finding the coefficients $w_i(t)$ for each time instant t such that $E\{e(t)^2\}$ is minimized. Under our hypothesis ($v(t)$ and $c(t)$ are uncorrelated) it results that

$$E\{e(t)^2\} = E\{c(t)^2\} + E\{(v_0(t) - \sum_{i=1}^{n} w_i(t)v(t-i))^2\}. \tag{3}$$

From (3), since $E\{c(t)^2\}$ does not depend on the coefficients $w_i(t)$, it results that $E\{e(t)^2\}$ is minimized if $E\{(v_0(t) - \sum_{i=1}^{n} w_i(t)v(t-i))^2\}$ attains its minimum. In such case, we can conclude that statistically $e(t)$ will be close to $c(t)$ and thus that the output error approximates the clear signal $c(t)$.



(a) Audio recording setup.

(b) ANC block diagram.

Figure 1: Adaptive noice canceling.

2.1 Test signals

For testing ANC you are given genarated signals. You need to test your algorithms with all the given simulated signals.

Bonus task 6 asks you to use also real recorded audio.

2.1.1 Simulated data

For testing, we consider one clear speech source, two different noise signals and two different rooms: one constant (imagine a person standing still in a quiet room) and one varying (the person is moving or something is happening around him).

- `clear_speech.wav` - the clear speech source $c(t)$.
- `noise_source.wav` - a random noise source $v(t)$.
- `speech_and_noise_through_room_1.wav` - the measured signal $s(t)$. This corresponds to the signal $v(t)$ distorted by passing it through an FIR filter of length 200, the filter coefficients are constant.
- `speech_and_noise_through_room_2.wav` - the measured signal $s(t)$. This corresponds to the signal $v(t)$ distorted by passing it through an FIR filter of length 200, the filter coefficients are chaning in time.

and for the structured noise

- `clear_speech.wav` - the clear speech source $c(t)$.
- `structured_noise_source.wav` - a random noise source $v(t)$.
- `speech_and_structured_noise_through_room_1.wav` - the measured signal $s(t)$. This corresponds to the signal $v(t)$ distorted by passing it through an FIR filter of length 200, the filter coefficients are constant.
- `speech_and_structured_noise_through_room_2.wav` - the measured signal $s(t)$. This corresponds to the signal $v(t)$ distorted by passing it through an FIR filter of length 200, the filter coefficients are changing in time.

2.1.2 Recorded data

Tom wants to be a recording artist. He sets up a home studio and records his playing and singing with a microphone. It's hot in his apartment and he wants to keep the window open. Crickets make noise outside and disturb Tom's recording `recorded_song.wav`. Tom decides to put another microphone pointing outside at the window. It records `crickets.wav`. Can you help Tom to remove the annoying crickets from his recording?
In practise, the data was recorded like in Fig. 2.

Figure 2: Recoding real data.

3 Summary of tasks

A short summary of the tasks you need to do for the project is given below (a more detailed description follows in the next sections). All the methods you should test with both noise sources and both room models.

3.1 Mandatory tasks

1. [1pt] Implement the NLMS algorithm and analyze its performance as a function of the adaptation step μ. See Section 4.2 for details.
2. [1pt] Implement the RLS algorithm and analyze its performance as a function of the forgetting factor λ. See Section 4.3 for details.
3. [2pt] Segment the audio data and generate the noise canceler by minimizing the error with the plain LS for each segment. See Section 4.4 for details.
4. [1pt] Use model order selection criteria to find the optimal filter order. See Section 4.5 for details.

3.2 Bonus tasks

5. [1pt] Use `spectrogram` to estimate the spectra of the signals you used this far. See Section 4.6 for details.
6. [1pt] Use NLMS and RLS with the real recorded audio described in Section 2.1.2. Find such parameters that the algorithms work well. Listen the result and show the spectrograms.
7. [1pt] Use the Yule-Walker equations to obtain a parametric spectrum of the test signals. Compare the result with `spectrogram`. See Section 4.7 for details.
8. [2pt] Implement a sparse recovery algorithm, known as orthogonal matching pursuit, to find the important coefficients. Run your adaptive algorithms for the sparse basis and analyze their performance as a function of the number of coefficients M. See Section 4.8 for details.

4 Methods in detail

4.1 Performance analysis

We want to know how good is our adaptive algorithm. We can analyze it qualitatively or quantitatively. Any good implementation should produce an output error $e(t)$ that approximates well the signal $c(t)$. It should look similar to Fig. 3. *Always listen to both $c(t)$ and $e(t)$ for a subjective evaluation of the performance!* You can use the function `soundsc` in Matlab for this. Also, always plot the signals, like in Fig. 3, so that you get an understanding of the scale of the signals.

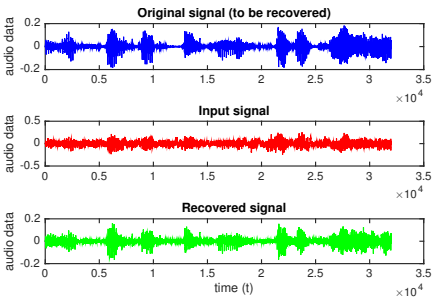


Figure 3: An example of audio data.

As the performance criterion, we use the average square error. At any given time, we can compute the error $\epsilon(t)$ between $c(t)$ and the cleared signal $e(t)$, $\epsilon(t) = c(t) - e(t)$ since we know $c(t)$ (in a real scenario this most probably is unavailable). We define the average squared error for the last n samples as

$$ASE = \frac{\|\mathrm{c}(t-n+1:t) - \mathrm{e}(t-n+1:t)\|_2^2}{\|\mathrm{c}(t-n+1:t)\|_2^2}.$$

We consider c and e to be vectors containing the values of $c(t)$ and $e(t)$ for all t. In practice we will pick $n = t/2$; we should be in a stationary region there. You are asked to compute the ASE often in the following tasks. Always compare to the situation when the error signal is the desired signal itself, this sets a reference below which the error of all methods needs to go!

4.2 Normalized least mean squares

Implement the NLMS algorithm and use it in the ANC setup described in Fig. 1b. Use at least all synthetic signals to test your algorithm. You have to produce and comment the figures for the average square error as a function of the descent parameter μ. The parameter μ should be in the interval $(0,\ 2)$; you may use a grid with the step equal to 0.05 to produce the figure. An example of the figure that you have to generate is found in Fig. 4.
What you need to submit:

- The implementation of the the NLMS algorithm
- A test file to produce and plot the average square error for different adaptation steps $\mu = 0.05 : 0.05 : 1.95$.
- Include in the document the figure and comment the results. Comment especially the effect of the noise source and the room model.

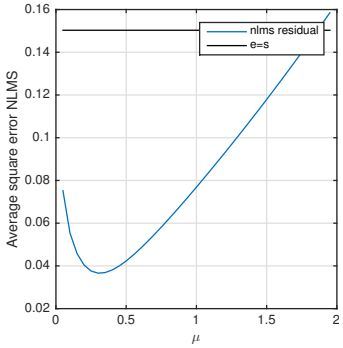


Figure 4: The average square error as a function of the adaptation step μ.

It doesn't matter if the noise source is noise or a structured sound. Explain, why not? Why the ASE-curve behaves the way it does?

4.3 Recursive least squares

Implement the RLS algorithm, as it is given in slide 24 of lecture 7 and run it in the setup described in Fig. 1b. Pay especially attention to initializing the inverse autocorrelation matrix!

When you have the implementation, you should produce and comment the figures for the average square error as a function of the forgetting factor λ. See e.g. Fig. 5.

What you have to submit:

- The implementation of the RLS algorithm
- A test file to produce and plot the average square error for different forgetting factors. Think what is the role of the forgetting factor and how it is used in the algorithm. Select the range of λs carefully!
- Include in the document the figure and comment the results. Explain especially the behaviour of the forgetting factor.

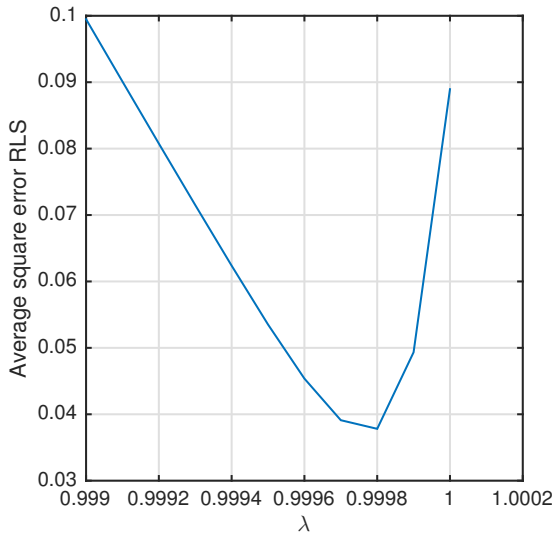


Figure 5: The average square error as a function of the forgetting factor λ.

4.4 Segmentation and least squares

For this task you need to partition the signals into segments and for each segment you need to find the LS solution to the minimization problem associated with the noise canceling problem. This is no longer an adaptive setup, we process the data in a batch mode. You should keep this in mind when analyzing the results! We want to see how the error $\epsilon(t)$ behaves if we have a constant environment (i.e. the coefficients don't change) or if the environment is variable (i.e. the coefficients change in time). Compare this to the signal descriptions!

You have to produce the average square error for different number $i = 1 : 10$ of segments and to compare the results between the constant and the variable environments. To exemplify if $i = 1$ you have to compute the LS solution for the whole interval of the original signal. For $i = 2$ the interval is split in two; two solutions, w_1 and w_2, need to be computed, one for each half; after this the results are aggregated to produce the output $e(t)$ for the whole interval. Thus to compute $e(t)$ you need to use weights w_1 for the first segment and w_2 for the second segment. In a matrix form, if e contains all the output data, we have $e = [e_1\ e_2]$ where e_i is generated using w_i. For $i = 3 : 10$ the task is similar just that there are more segments. Care needs to be taken if the segmentation can not be performed into equally sized segments (in this case one segment can be larger).

The LS solution results from the system of equations,

$$Aw = d, \tag{4}$$

where the matrix A contains the input data, in our case $v(t)$, and the vector d contains the desired data $d(t)$. *You should build the matrix* A *using the prewindowing method* (Presented in Lecture 7,

slide 6.). The system of equations is generally overdetermined and the LS solution is given by

$$\mathrm{w} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathrm{d}. \tag{5}$$

In Matlab this can be easily implemented using the command $w = A\backslash d;$.

What you have to submit:

- The implementation of the segmentation and the LS solution
- A test file to produce and plot the average square error for different segmentations $i = 1 : 10$ of the same input interval. You should report the average square error computed for the whole interval.
- Include in the document the figure and comment the results. Comment especially the difference of the room models and the relation to the nlms result.

4.5 Order selection

This far, the order of the (adaptive) filters was held fixed: say at 200. Now, we want to estimate the order from the data using a model order selection criterion. The goal here is to select an order that decribes well the source generating the data, which in our case is the room model, but not the noise. You are required to run the adaptive algorithms, nlms and rls, for several orders, and compute the model order selection criteria given in the last page of lecture 8, namely the final prediction error (FPE), akaike information criterion (AIC) and the minimum description length (MDL).

What you need to submit:

- A test file to produce and plot the criterion values for different model orders.
- Include in the document the figure and comment the results.

4.6 Spectral estimation

Figure 6 shows an example of a spectrogram for a similar audio signal that is used in this project. You should use the MATLAB function `spectrogram` and choose such parameters that you get a figure resembling Figure 6 for the audio signals you are given. Explain the meaning of the parameters and what you see in the figure.
Create the spectrogram for all the signals in the ANC application ($\boldsymbol{c}$, $\boldsymbol{v}$, $\boldsymbol{s}$, $\boldsymbol{e}$) and explain how you see the effect of the adaptive filtering algorithm in the result.
Explain also how `periodogram` is related to `spectrogram` and try to create a figure similar to Fig. 6 with it using e.g. `mesh` or `pcolor`.

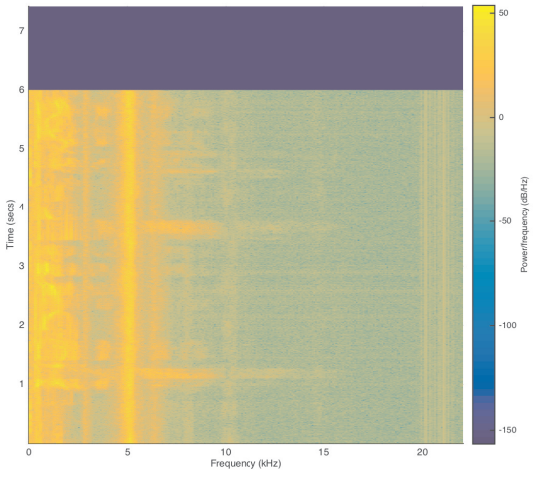


Figure 6: Spectrogram using a Short-Time Fourier Transform (STFT)

4.7 Parametric spectral estimation

Assume that signals follow an autoregressive model with parameters $\begin{bmatrix} a_1 & a_2 & \dots & a_M \end{bmatrix}$.

Solve the Yule-Walker equations

$$\begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r(1) & r(0) & & \vdots \\ \vdots & & \ddots & r(1) \\ r(M-1) & \dots & & r(0) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(M) \end{bmatrix}, \tag{6}$$

where $w_k = -a_k$, manually using estimated autocorrelations r to obtain a parametric spectrum estimate. Create spectra similar to Section 4.6 by displaying the value

$$P_{AR}(f) = \frac{\sigma^2}{|A_x(e^{j2\pi f})|^2},$$

where $\sigma^2 = \sum_{k=0}^{M} a_k r(k)$ is the variance of white noise and $A_x(z) = 1 + \sum_{k=1}^{M} a_k z^{-k}$. Does the parametric spectrum model fit our signals? Repeat the same using functions `aryule` and `freqz`.

4.8 Sparse filtering: Orthogonal matching pursuit

Different applications involve filters with a particular structure. In recent years there has been much interest directed towards sparse representations. The data that we use (the audio files) were generated using a sparse filter to model the room environment. An example of such a filter can be found in Fig. 7. You can see, that most of the coefficients are zero. We want to find which coefficient locations are nonzero, using an algorithm known as Orthogonal matching pursuit.

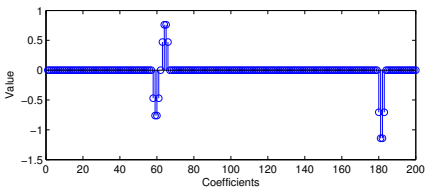


Figure 7: Example of a sparse filter.

If the coefficient locations would also change in time, we would need to use an algorithm that combines the algorithm for finding a sparse basis and the adaptive filtering. One such is adaptive matching pursuit [1]. Here we, however, restrain to the simpler case where the nonzero locations stay fixed and we simply find them once. After that we can run the normal nlms or rls algorithm for those nonzero filter locations. The details of OMP can we found in [2], that you can download from `http://www.stat.yale.edu/~snn7/courses/stat679fa13/references/omptrogil.pdf`. Essentially you need to extract a segment of the signals, say 10000 samples, build a matrix system $\boldsymbol{Aw} = \boldsymbol{d}$ as in the segmentation task, and then find the columns of $\boldsymbol{A}$ that best correspond to $\boldsymbol{d}$ using the OMP algorithm.

What you have to submit:

- The implementation of the the OMP algorithm and the modified NLMS that updates only the nonzero coefficients of the filter.
- A test file to produce and plot the average square error for different sparsity levels $M = 5 : 25$. You may also include a point having a large M for example $M = 80$.
- Include in the document the figure and comment the results.

References

[1] S.F. Cotter and B.D. Rao. The adaptive matching pursuit algorithm for estimation and equalization of sparse time-varying channels. In *The 34th Annual Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1772–1776, 2000.

[2] Joel A. Tropp and Anna C. Gilbert. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 2005.