SSY130 - Applied Signal Processing

Project 2:
Adaptive Noise Cancellation
Rev 1

T. McKelvey

2016

1 Introduction

The purpose of this project is to learn how to implement and use an adaptive filter algorithm, the *least mean squares* algorithm (LMS). The implementation platform is the STM32F407 processor, a digital signal processor from ST Microelectronics. The processor is low-cost and has a low power consumption. The particular application which we are targeting is noise cancellation. Noise cancellation means that given a measured signal, which is contaminated with a disturbance signal, and a given separate measurement of the source of the disturbance we try to find a filter which can reduce the disturbance from the desired signal. Active noise cancelling headsets, which today are available on the market use adaptive filtering techniques similar to the LMS technique you will use in this project.

The project is divided into two parts:

Part A Implement the LMS algorithm using the C-language and test it.

Part B Investigate the properties of the LMS filter applied to the active noise cancellation problem using audio capabilities of the DSP-kit.

2 Noise cancellation

Consider the case when the audio in a room is measured using a microphone. We asume the audio is composed of two separate sound sources. One source is the audio we want to listen to, e.g. a singer or a speaker. The other audio signal origins from a disturbance source which also propagates through the room and are also picked up by the microphone. Hence, if $x(n)$ is the sound signal sampled by the audio system at the microphone location we can write

$$x(n) = s(n) + v(n) \tag{1}$$

where $s(n)$ is the desired audio signal and $v(n)$ is the noise signal. Sometimes it is possible to directly measure the noise source (independently of $s(n)$). We call this noise source signal $y(n)$. We assume the propagation from the noise source $y(n)$ to the microphone which records $x(n)$, the noise path, is a linear filter with impulse response h_k and we can write

$$v(n) = \sum_{k=0}^{\infty} h_k y(n-k) \tag{2}$$

If we would know the impulse response of the noise path we could approximate it with a finite length filter (FIR) with M filter coefficients. Using this filter and the measurement of the noise source $y(n)$ we can subtract the noise from the microphone signal as

$$e(n) = x(n) - \hat{x}(n) = x(n) - \sum_{k=0}^{M-1} \hat{h}_k y(n-k) \tag{3}$$

where $\hat{x}(n)$ is the predicted disturbance at the microphone location and $\hat{h}_k$ is the approximated impulse response of the disturbance path. If the filter is good the "error signal" $e(n)$ would then be the desired audio signal $s(n)$.

The key issue is how to obtain the filter coefficients of the disturbance path. A closer look at equation (3) indicates that $e(n)$ is the error in the standard adaptive filter setting. If we assume that $s(n)$ and $y(n)$ are statistically uncorrelated we can obtain the desired result by minimizing the variance of e. Why is it the assumption of uncorrelation crucial here? We can use the LMS filter algorithm to both provide the filtering and the adaptation at the same time. The improved signal (cleaned from noise) is the error $e(n)$ from the adaptive algorithm.

2.1 Adaptive Filtering with LMS

The filtering (convolution) with a FIR filter (at sample n) with M coefficients can be described by

$$\hat{x}(n) = \hat{\mathbf{h}}^T(n-1)\mathbf{y}(n) \tag{4a}$$
$$e(n) = x(n) - \hat{x}(n) \tag{4b}$$
$$\hat{\mathbf{h}}(n) = \left[\hat{h}_{M-1}(n), \cdots, \hat{h}_0(n)\right]^T \tag{4c}$$
$$\mathbf{y}(n) = [y(n-M+1), \ldots, y(n)]^T \tag{4d}$$
$$\tag{4e}$$

where $\hat{\mathbf{h}}(n)$ is the parameter vector that needs to be estimated, and $\mathbf{y}(n)$ is the regressor vector that contains the present and $M-1$ past input signals $y(n)$.

The optimal parameters estimate $\hat{\mathbf{h}}(t)$ for a stationary application is defined as the minimizing arguments of the mean-squared error criterion:

$$\hat{\mathbf{h}}_{\text{opt}} = \arg\min_{\mathbf{h}} \mathsf{E}[e(n)^2] \tag{5}$$

where error $e(n)$ is given by (4b) and E is the expectation operator.

An approximate solution to the criterion (5) is obtained by using the LMS algorithm [1]:

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + 2\mu\mathbf{y}(n)\underbrace{\left(x(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{y}(n)\right)}_{e(n)} \tag{6}$$

where 2μ is the step length. More details on adaptive filters is provided in the lectures and in Chapter 8 of the text book [1].

2.2 Project setup

To illustrate the active noise cancellation application we will use DSP-kit. The noise path will be represented by the propagation of sound between the left loudspeaker and the on-board microphone on the DSP-board. The DSP will mix the desired music signal s with the noise source y and then transmit the signal to the left loudspeaker. The microphone will pick upp this signal which we call x. The improved signal e is sent to the right output channel on the board.

Summary:

- DSP microphone input : $x(n)$ = signal + noise through noise path
- DSP left output: $s(n) + y(n)$ signal + noise source
- DSP right out: $e(n)$ = improved signal

For the purpose of evaluation the DSP has both the signal and noise sources stored in memory. Through the monitor you can change the character of the noise source signal by pressing key "t". This will toggle between a pure cosine signal and a more wide band noise source.

2.3 Project Task: Part A code development and testing

Design a C-function which performs LMS adaptive filtering for a block of input data. To support block based filtering of streaming data some of the past input samples y need to be saved to be able to reuse them when performing the filtering operation on the next block. To calculate the output of the first sample in a new block we need access to the $M-1$ input samples from the input data supplied in the previous call to the filtering function. We accomplish this by using a "state" vector of length $M-1$ plus the size of the input data block-size. When the function is called it is assumed that the $M-1$ "old" samples of the input are stored in the beginning of the state vector. By then also adding the new input data to the same vector we have one long vector which contain all data needed to produce the entire block filtered output (by convolution). Finally we need to update the state vector by placing the last $M-1$ input data y first in the state-vector.

In the project folder `asp_proj_lms/src` the source file `lab_lms.c` is located. In this source file you find the skeleton of the `my_lms()` function. In the supplied code the data movement operations are already supplied in the `my_lms()` function code. It is your task to complete the code such that the function performs the LMS algorithm on a block of data. The function should perform the operations according to the description given by equations (4a) to (6).

To help you debug your C-implementation of the LMS algorithm you can set the macro variable `TESTMODE` to be equal to `TEST`, i.e. near the top of in the `lab_lms.c` file you write

```
#define TESTMODE TEST
```

With this modification the code will only run a test of the `my_lms` function and print out the result and then halt execution. You can follow the code of the test function by looking into the `lab_lms_init()` function. The test signal `x_data` has been created by filtering `y_data` through a FIR filter of length 8.

When you don't want the test mode any longer, change the macro line to

```
#define TESTMODE NORMAL
```

in order to enable the normal mode of operation.

2.4 Project Task: Part B real-time noise cancellation

After completing the previous section, you should be confident that your code for the LMS algorithm is correct.

1. To restart the program from the beginning you can use the reset button on the board.
2. The music signal and a noise signal should now be output through the left channel. Initially the signal picked up by the microphone on the board is directly transmitted to the right output channel.
3. If you move the left audio channel speaker (or headphone) close to the board you should be able to hear the signal also in the right output channel.
4. By pressing button 'u' you start the LMS algorithm including the filter updating. If everything is working the noise signal should be decreased and removed from the right audio channel.
5. Pressing the 'f' button disables the filter coefficient updating but the filtering is still active. If you now change the distance between the left speaker (headphone) and the microphone the noise signal should appear again. Restoring the original distance should remove the noise again.
6. The step size μ can be modified with the '+' and '-' buttons.
7. You can print the filter coefficients in the monitor by pressing 'p'. The vector printed is formated so it can easily be copied into MATLAB for further processing.
8. You can toggle between two different noise sources. A sinusoidal signal and a more wideband noise signal. You toggle between them by pressing 't'.
9. Try the different signals and see if this influences the filter. Examine the adaptive filter coefficients. Do they converge to similar values regardless of the input signals used? Try to explain the effects.
10. Use the lms algorithm implementation to remove the sinusoidal disturbance Listen to the result.

- How long filter `LAB_LMS_TAPS` is required for the disturbance to go away? What is expected from the theory? (Hint: Remember that the signal we want to remove has a very special structure in the frequency domain.)
- How does the rate of convergence depend on the step size
- Use a rather large step size, but small enough to make the music sound OK. Then set the step size to zero by pressing 'f'. Did the filtering result now get worse or is it the same? Why?

11. Now test the using the wide band noise disturbance instead.

    - Redo the investigations listed above. What length of the filter is required now (less than 256). Explain the difference from above. Plot the resulting impulse response. What do you see?
    - What is the performance if the adaptation is turned off after an initial training period? What happens if you then change the character of the input signal or the channel without enabling the filter updating again? Is there a fundamental difference between training the filter using a broadband disturbance source or using a narrow-band disturbance source?

2.5 Report

Write a report that describes your work and findings.

2.6 Project examination

The work of each project group is evaluated through the report, the C-code and an oral exam. Write a report that describes your work and findings. Try to include answers to the questions posed above. Completed reports and code in `lab_lms.c` for Project 2 shuld be uploaded to pingpong before the deadline indicated. The report is limited to 6 pages including 1 title-page. The title-page should contain the project number, group name and student names.

In the oral exam the teacher will discuss different aspects of the project with the students. It is an opportunity for the students to ask questions to learn more, and for the teacher to sense the level of understanding in the group.

Referenser

[1] B. Mulgrew, P. Grant, and J. Thomson. *Digital Signal Processing - Concepts and Applications.* MacMillan Press Ltd., Basingstoke, United Kingdom, 1999.

[2] T. McKelvey *Applied Signal Processing - An introduction to the C-programing language.* Available on pingpong, 2016.