ECE4703 B Term 2011 Laboratory Assignment 6

Project Code and Report Due by 3:00pm 15-Dec-2011

The goals of this laboratory assignment are:

- to familiarize you with adaptive filtering and two applications of adaptive filtering on the TMS320C6713 and
- to familiarize you with the Least Mean Squares (LMS) algorithm and its implementation on a floating point DSP.

1 Problem Statement

In this assignment, you will write a C program for the TMS320C6713 DSK that realizes an *adaptive* FIR filter via the Least Mean Squares (LMS) algorithm. The LMS algorithm was developed in the 1960's and has been called the "workhorse of adaptive filtering". LMS-adapted FIR filters are simple to implement and have been used in a wide variety of applications. In this assignment, you will use an LMS-adapted FIR filter for two useful applications: system identification and noise cancellation.

For all code written in this project, the AIC23 sampling rate should be set to 44.1kHz. All computations should be done in **single-precision floating point** math. It is highly recommended that you convert all inputs directly to floats immediately after reading them. All multiplies and adds should be done in floating point and the outputs should only be cast to shorts (and put in the appropriate 32-bit containers) prior to sending them to the AIC23. Using floating point math should considerably simplify your code and allow for easier troubleshooting.

Your filter update algorithm should be LMS as discussed in lecture. Specifically, at each sample index n, your FIR filter coefficients should be updated as

$$b[k, n+1] = b[k, n] - \mu e[n] x[n-k] \qquad k = 0, 1, \ldots, N-1$$

where μ is a fixed filter update step size that you select, $e[n]$ is the error signal at time n, and $b[k, n]$ denotes the k^{th} filter coefficient at time n. Recall that selecting μ too large will cause your filter to "blow up".

The following parts of the assignment will use your adaptive filtering code to do two useful signal processing tasks.

1.1 Part I: System Identification

In this part of the assignment, you will use your LMS-adapted FIR filter to estimate the impulse response of an unknown system. A block diagram showing the AIC23 codec channels for the system

identification system is shown in Figure 1. Here, $x[n]$ is the input to the unknown system and $d[n]$ is the output of the unknown system. Recall that you control the input to the unknown system. Typically, white noise is a good choice for the input signal. The goal is to determine the impulse response of the unknown system. LMS will do this automatically for us since it will adapt the FIR filter coefficients to minimize the mean squared error between its output $y[n]$ and the output $d[n]$ of the unknown system. This can only occur if the LMS-adapted filter's impulse response converges to match to the unkemptnown impulse response. After your filter has adapted, its impulse response should be very similar to the unknown system and the mean squared error should be very small.

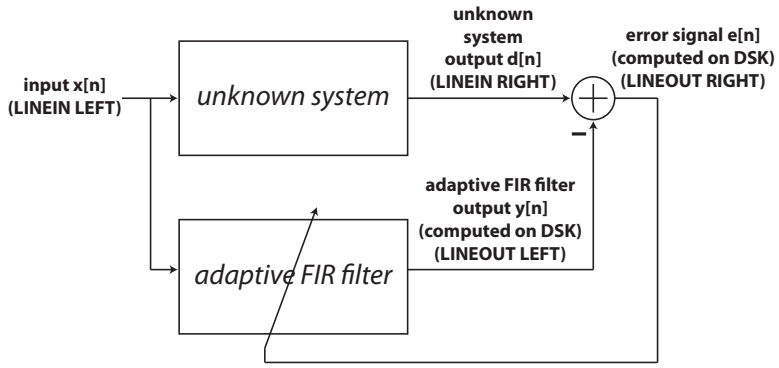

Figure 1: System identification application of adaptive filtering showing codec inputs and outputs.

To test your adaptive filter's ability to identify an "unknown" system, try making up your own test signals (in Matlab) with a known system (you could, for example, use your bandpass filter from Lab assignment 2 as your "unknown" filter). Generate a white noise input $x[n]$ and use the Matlab `conv` or `filter` function to produce the output of the "unknown" test system $d[n]$. Play these signals into the left and right line inputs on the DSK (making sure to avoid clipping), allow the adaptive filter to converge, and stop the execution of your code before the signals finish playing to halt the adaptation and view the results. Plot the impulse response of your adaptive filter. Does it make sense? Can you verify that your filter converged correctly? Experiment with various filter lengths N and step sizes μ to see which combinations are able to most effectively identify the "unknown" system. Is your error signal (right channel output) going to zero? Also check the frequency response of your adaptive filter. Does it make sense?

Once you are confident that your adaptive filter is working correctly, download the files in `lab6part1.zip` from the course web page and test your adaptive filter on some truly unknown systems. In this case, you are provided with the white noise input and the unknown system output in the left and right channels of the .wav file. To perform system identification, you simply apply these signals to the line inputs of the DSK in the same way that you did when you tested your code with a known system. Your adaptive filter should converge such that the error signal becomes small. You should experiment with different values for N and μ to see which lead to the smallest residual MSE.

1.2 Part II: Noise Cancellation

In this part of the assignment, you will use your LMS-adapted FIR filter to cancel undesired noise from an audio signal. The structure for this adaptive filtering application was discussed in

lecture. A block diagram of the AIC23 channels used in the noise cancellation system is shown in Figure 2. The left input channel signal contains $d[n] + w'[n]$, the desired component corrupted by an undesired noisy component correlated with the externally measured noise $w[n]$. The right input channel contains $w[n]$, the externally measured noisy signal. Your goal is to cancel as much of the noise as possible while also minimizing distortion on the desired signal. The LMS-adapted filter does just that by adapting its coefficients so that the mean squared error ($e[n] = d[n] + w'[n] - y[n]$) is minimized. The is accomplished by adapting the filter coefficients so that $y[n]$ (the filtered version of $w[n]$) approximates $w'[n]$ and the residual signal $e[n]$ contains only the desired component. Note that, unlike the system identification application, the residual error $e[n]$ will not become very small. The residual error $e[n]$ will actually contain the music after the noise has been removed.

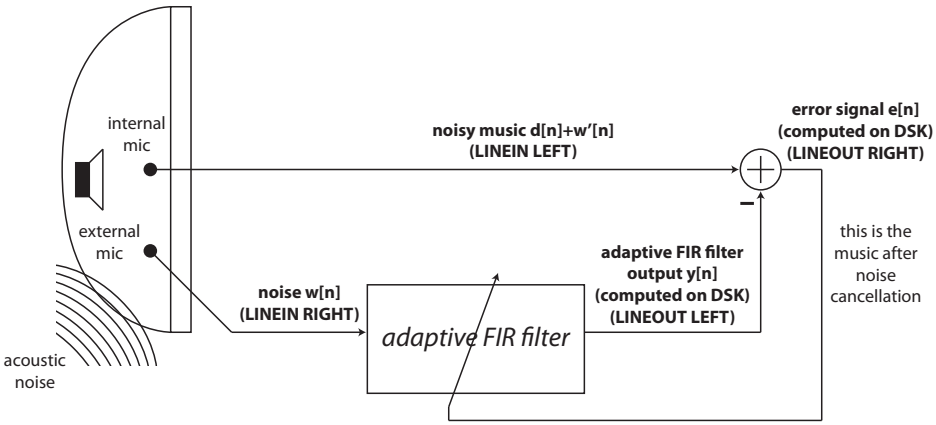

Figure 2: Inputs and outputs of adaptive filtering system for noise cancellation.

As in Part I, you should test your noise cancellation system with known signals first. Generate some noise $w[n]$ and filter it with an arbitrary filter to produce $w'[n]$ (the correlated noise signal). Also generate a desired signal $d[n]$ (some music or speech is a good choice), add the noise signal $w'[n]$ to it, and test your adaptive filter. If your filter is working correctly, the right output channel $e[n]$ should sound very similar to the desired signal $d[n]$. There may be some residual noise in the output, but it should be small. Experiment with various filter lengths N and step sizes μ to see which combinations are able to most effectively mitigate the undesired noise.

Once you are confident your noise cancellation system is working well, download the files in `lab6part2.zip` from the course web page and test your adaptive noise cancellation system on unknown signals.

2 Suggested Procedure for Software Design

1. A good first step would be to implement fixed FIR filtering on the left channel input. Leave your code modular to allow for any length FIR filter. Ensure that your fixed FIR filter is working correctly before proceeding. You are encouraged to use code developed in earlier lab assignments, if it works correctly.

2. Modify your code to allow for adaptive filtering. Here, you will need to implement the LMS filter coefficient update. It might be a good idea to set $\mu = 0$ and ensure that nothing has

changed from your non-adaptive case.

3. Select a good value for μ and test your adaptive filter on known signals that you've generated. You can play these stereo signals out of the line out jack on the computer using Matlab or another audio player (or you could use your phone for playback). Make sure whatever audio player you use is not adding any special effects to the signals[1].

4. Also don't forget to initialize your filter coefficients to something reasonable (like all zeros) before you adapt them.

3 In Lab

You will work with the same lab partner as in the prior laboratory assignments. Please contact the instructor if your lab partner has dropped the course or if you have concerns about your lab partner's performance on the prior assignment.

4 Code Submission and Specific Items to Discuss in Your Report

Since this assignment is due on the last day of classes, no late submissions can be accepted. Please make sure you submit your project code and report on time.

The emphasis of this assignment is on the implementation of the LMS algorithm and two applications of adaptive filtering. Your report should focus on any implementation issues that you encountered including any results and insight gained in terms of selecting N and μ for the various signals. Your report should also focus on any insight gained from the two applications including presenting results that conclusively demonstrate that your filter is converging to the optimum solution. In the system identification application, you should generate your own $x[n]$ and $d[n]$ with a known system (using the Matlab `conv` or the `filter` command) and show that your filter converges correctly by plotting its impulse response over the known system's impulse response. In the noise cancellation application, you may want to record the entire signal during the convergence of your filter and plot selected parts of this signal to show the presence of the noise at the beginning of convergence and the cleaned-up signal later in the convergence process.

[1] Media players (and sometimes the Windows soundcard driver itself) sometimes add unexpected effects to the playback/recording signals which will affect your system identification results. Make sure all effects are off. You should confirm the frequency response of the playback/recording system is approximately flat.