California Polytechnic State University San Luis Obispo

College of Engineering

Electrical Engineering Department

Active Noise Cancellation

By

Sihyung Lee

Supervisor

Prof. Wayne Pilkington

June 13, 2011

Table of Contents

List of Tables and Figures

Acknowledgements

I would like to thank Dr. Pilkington for providing me great instructions.

Abstract

In this project, I tried to build a noise-cancellation system using DSK. Two microphones and one speaker were used in the system: the first microphone is to sample unexpected noises from the outside and the second microphone is to collect all the sounds, including both the desire sounds and unexpected noises. The second microphone detects and evaluates how well the noise cancelation works. I used Least Mean Square algorithm to implement Noise Cancellation system. The output of the adaptive filters in LMS algorithm would be phase shifted by 180 degrees and sent to a speaker to generate the anti-noise sound, which would cancel noises.

Introduction

This Noise Cancellation system is motivated to improve passengers' comforts in the aircraft industry, such as Bose. There are a lot of noise source in an aircraft from its engine or the friction created between aircraft and air. Those noises can hurt passengers' ears. As the one property of sound, sound can be destructed by inverting and adding it to the same sound. The result of inverting and adding sound is shown in Figure 1. But, this approach should have a perfect distance of microphone and speaker, or it can generate a greater noise if there is some time lag. In the worst case, noise can be twice larger than the original noise.

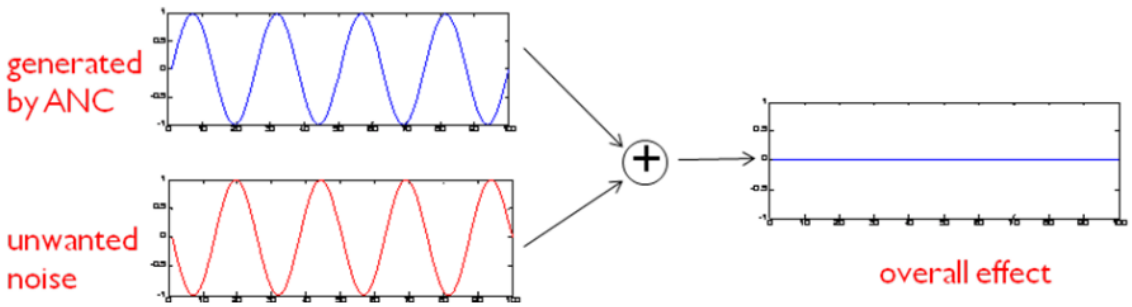

Figure 1: A sound cancellation with destructive interference.

A better method for noise cancellation system can be implemented by using adaptive filters. Adaptive filter is a filter that adjusts coefficients of its transfer function by itself, according to the optimization algorithm driven by an error signal. This method, which reduces undesired noises gradually, is called Active Noise Cancellation. Actually, there are two ways for noise cancellation: passive and active noise cancellation. Passive noise cancellation cancels noise with using some material, which can block the sound. Oppositely, active noise cancellation cancels noise, generating 180 degree phase shifted sound. The benefit of active noise cancellation is more efficient in low frequency and possible to block noise signal selectively. Most industries use both passive and active noise cancellation system to optimize the whole system. In this project, I used Least Mean Square algorithm, which is active noise cancellation algorithm and one of adaptive filters algorithm to reduce noise. This report will demonstrate the approaches that I use for this project.

Background

Usually, noises are random and have some characteristics of amplitude, phase, and speed of sound. To control this random noise well, the active noise control system should be used. Adaptive filter algorithm adjusts its internal filter coefficients to minimize error signals between unknown system and adaptive filter. The least-mean-squares algorithm is one of adaptive filters algorithm; the LMS algorithm and its relatives are all adaptive filtering algorithms. The basic block diagram is shown in Figure 2 and Figure 3.

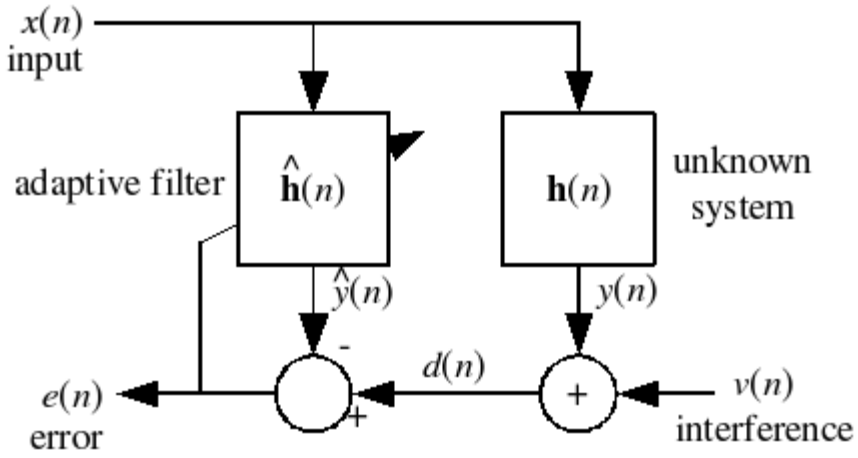

Figure 2: A block diagram of the whole system

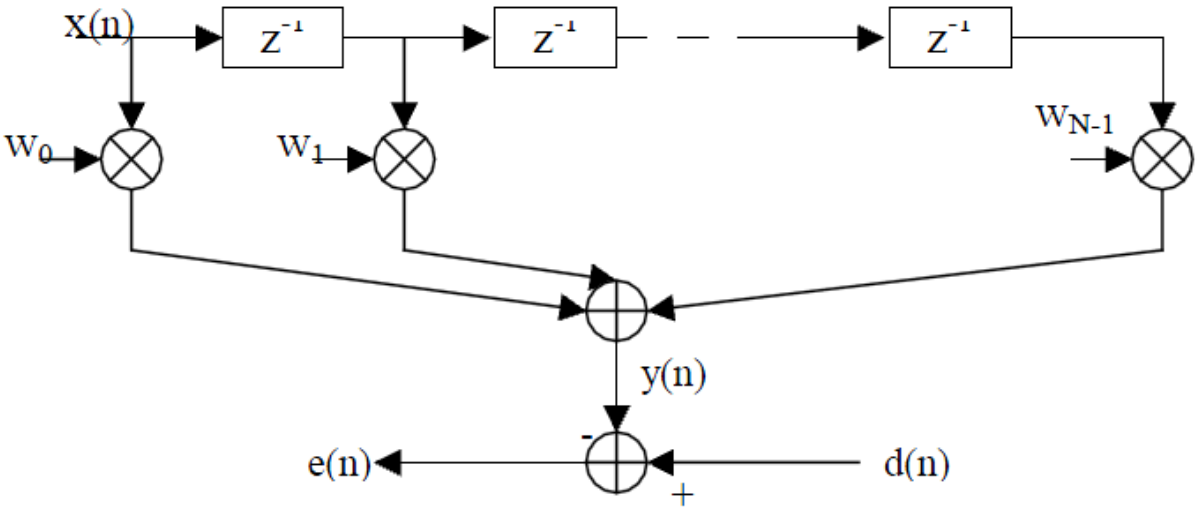

Figure 3: A block diagram of an adaptive filter part in LMS algorithm

- x[n] is the input to the system and collects noise
- y[n] is the output of the unknown system H(z). It can be any desire signal.
- ŷ[n] is the output of the adaptive filter Ĥ(z), which has adaptive filter coefficients
- d[n] = y[n] + v[n]
- e[n] = d[n] − ŷ[n]
- μ is the "step size" or "learning rate"
- Definition of symbols

$$\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-p+1)]^T$$

$$\mathbf{h}(n) = [h_0(n), h_1(n), \ldots, h_{p-1}(n)]^T, \quad \mathbf{h}(n) \in \mathbb{C}^p$$

$$y(n) = \mathbf{h}^H(n) \cdot \mathbf{x}(n)$$

$$e(n) = d(n) - \hat{y}(n) = d(n) - \hat{\mathbf{h}}^H(n) \cdot \mathbf{x}(n)$$

The basic performance of adaptive algorithm is to make $\hat{H}(z)$ as similar as the unknown system H(z) by the calculation of equations shown below.

1. $Y(n) = H^T \times X(n)$ for filtering

2. E(n)=D(n)-Y(n) for error estimation

3. $H(n+1) = H(n) + 2\mu * E(n) \times X(n)$ for updating filter coefficients

As time goes on, the last equation, shown above, lets the adaptive filter coefficients will be updated to be the same as coefficients of the unknown filter. Therefore, the error between the unknown filter and adaptive filter will be minimized because those two filters get more similar as time goes on.

Requirements

The general requirement for this project is to cancel out noise from a speaker. If a speaker is not made from some good materials to cover ears perfectly, then noises disturb users to listen to only what they want; which way is known as a passive noise cancellation. The main problem to make a perfect passive noise cancellation system is that size and shape of each human ear are different and money costs a lot for good materials. It is very difficult to cancel out some low frequency sound as well. Therefore, many users want to have the active noise cancellation system, which cancel out noise decently.

Design, Development and construction

First, I considered about the position of a speaker and microphone, which collects output sound and noise, because the distance between a speaker and microphone are critical. It can cause addition of sound wave's properties; the system should have destruction, not addition. It should have the distance of the full wave length of sound wave to work correctly. However, I assume that DSP board has infinite fast which cannot be compared with the speed of sound wave. I just tried put microphones and speaker together as close as possible. The schematic of whole system is shown in Figure 3.

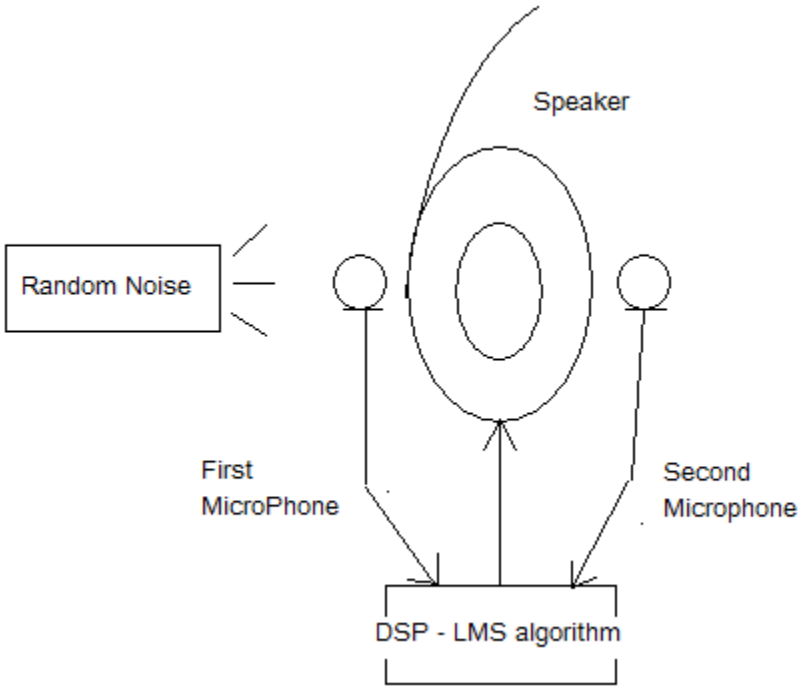

Figure 3: Schematic of Noise Cancellation system

Most of works are done in programmable language. First, using built-in function in Matlab, I tested how least mean square algorithm is running with variety of filter length and μ,

step-size. With the filter length and step-size chosen in Matlab, I designed least mean square algorithm in C language and tested it to make sure that the program works properly.

The next step was to implement the real program. I used the “Starter program,” which is used in EE 419 to develop the real time digital filters by Dr. Fred DePiero, Cal Poly State University. I replaced two functions for initializing variables and evaluating with my code for least mean square algorithm. The initializing variables function would be called only once to assign value into variables right after program is running and evaluating function would be called at every sampling time.

Test plans

I focused on designing the noise cancellation system, which can destruct only a constant noise first because least mean square algorithm is the slowest algorithm in adaptive algorithms. Therefore, some random noise can be skipped and difficult to see whether noise is canceled or not, if that random noise is generated for a short time.

Test plans of this project are simple. The first job is to mount my code into DSP board, and connect a speaker and microphone to DSP board. I connect a desired signal to the input of an oscilloscope and the output from a headphone to the output of an oscilloscope. I contrast the output and input of an oscilloscope.

A constant noise was generated by a speaker, using Matlab. If the contrasted value of output and input of an oscilloscope is decreased, then it is a good starting point to optimize the noise cancellation system by changing the filter length and step-size.

Analysis and test results

First, I used a build-in function, which is adaptfilt.lms, in Matlab. This adaptfilt.lms function would show me how least mean square algorithm works with some certain filter length and step-size. I tried to look for a good filter length first for this project. Three graphs of least mean square algorithm with different filter length are shown on the next page.

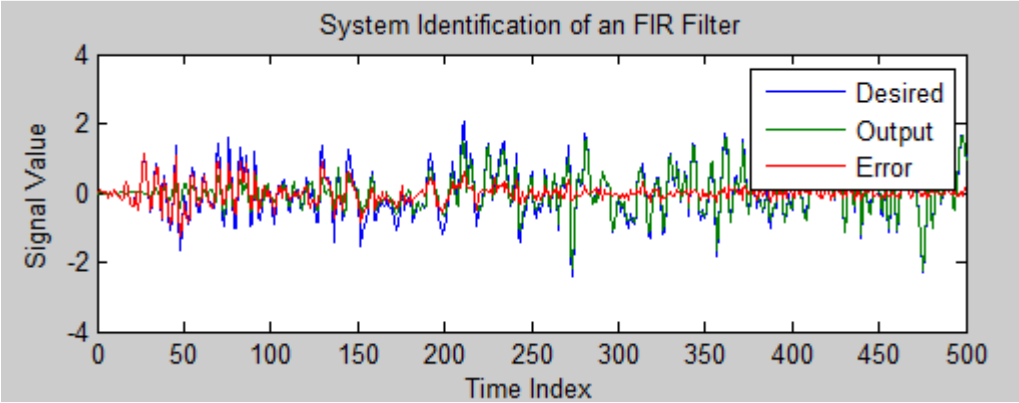

Figure 5: A graph of least mean square algorithm with 30 filter length and .008 step-size

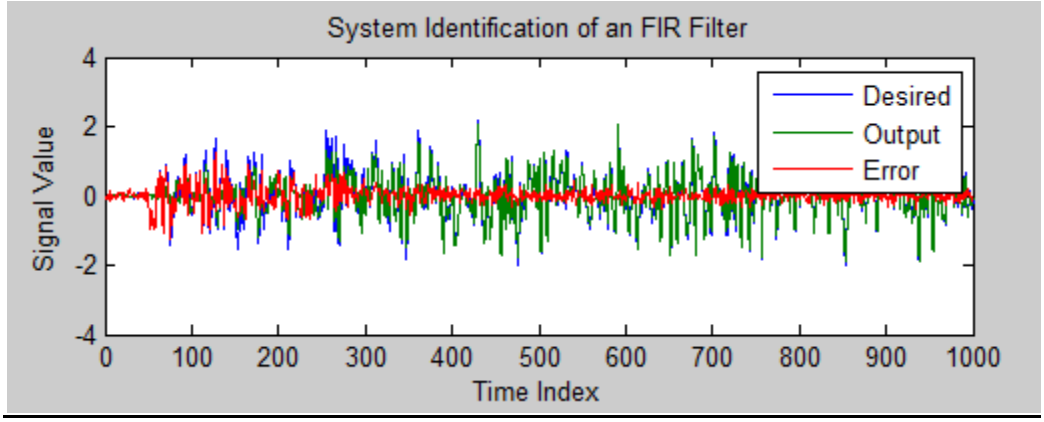

Figure 6: A graph of least mean square algorithm with 100 filter length and .008 step-size

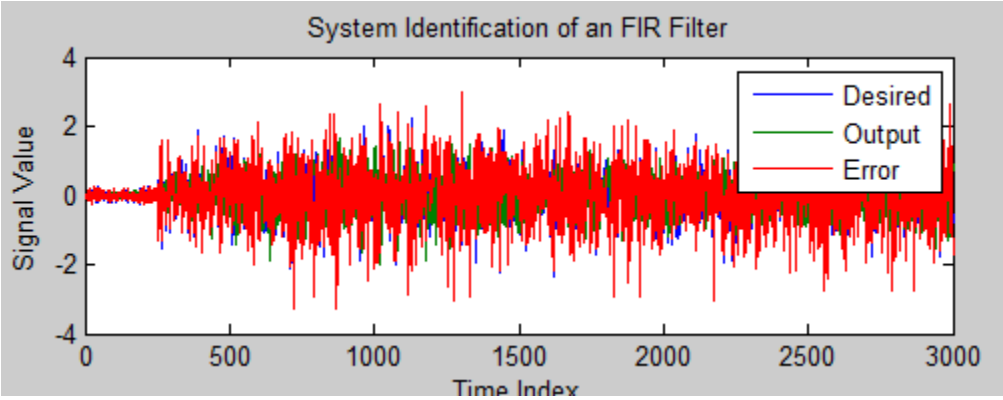

Figure 7: A graph of least mean square algorithm with 500 filter length and .008 step-size

Three adaptive filters with same step-size and different filter length show me quite bit different result. Actually, an adaptive with 500 filter length cannot be used for this project because the DSP board has limitation of some filter length and it is not good value to reduce error definitely as shown in Figure 6. The system with 500 filter length is not stable. According to longer filter length, the system requires to do more math works. Especially, transfer function in least mean square, takes double array. In other words, the system gets slower by (its filer length times its filter length).

Two adaptive filters with 30 filter lengths and 100 filter lengths reduce noises well as shown in Figure 5 and Figure 6. Both filter start reducing error properly around 300 iterations after it begins running. However, longer filter lengths require math work exponential more. Therefore I chose 30 filter lengths to build my least mean square algorithm.

The next step was to find a certain value of step size. I tried to find a certain step size, which can reduce noise the most and fastest. Simulation results are shown in Figure 8, Figure 9, and Figure 10.

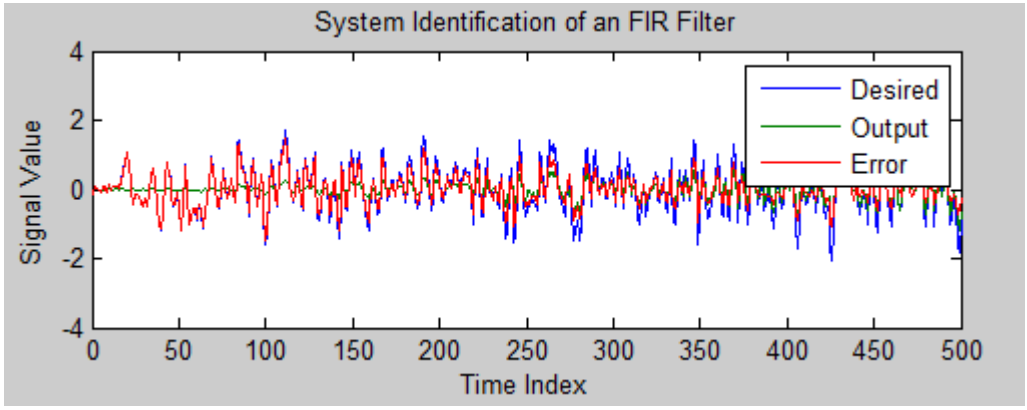

Figure 8: A graph of least mean square algorithm with 30 filter length and .002 step-size

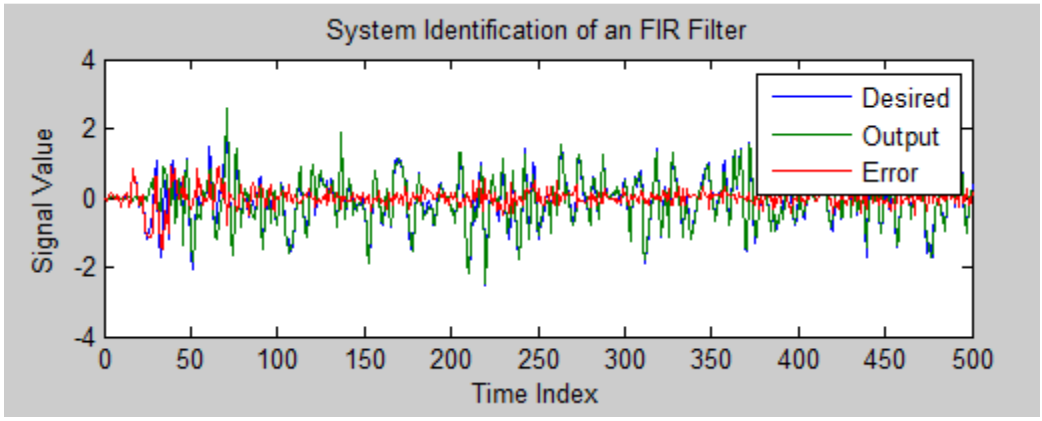

Figure 9: A graph of least mean square algorithm with 30 filter length and .04 step-size

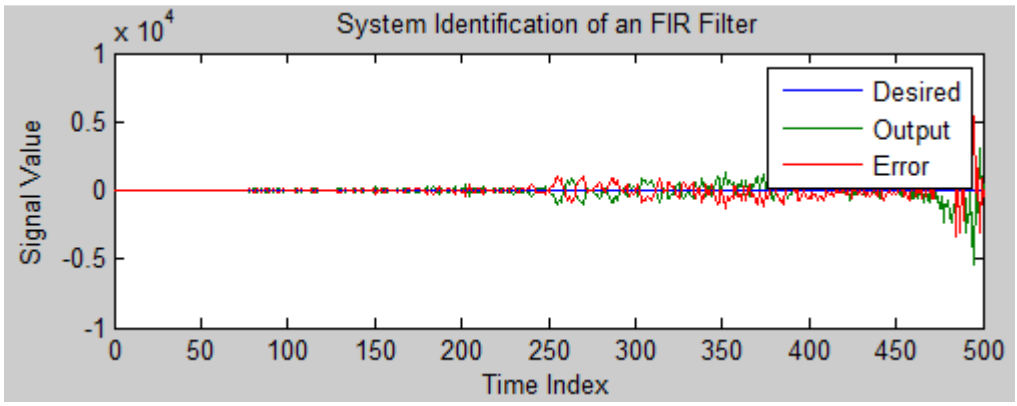

Figure 10: A graph of least mean square algorithm with 30 filter length and .08 step-size

Actually, step-size is the key factor to reduce error. If it has too low value, then it takes long time to become stable. If it has too high value, system can be unstable. In Figure 8, the system has step-size of 0.002. There is almost no difference from the beginning to 500 iterations. Users will hear all the noise without any cancellation. In Figure 10, the system seems to reduce noise well at the beginning, but its scale of signal value is ten thousand. Additionally, the error increases exponentially. The system is unstable. The system with step size of .04 works properly. It starts reduce noises around at 100 iterations and noises would be reduced well. In the result of Matlab simulation, I chose to use filter length of 30 and step size of 0.04. I designed the real program with those determined values using in C language. Excel is used to display changes of the error, which is difference between the desire signal and output of the adaptive filter, in C language is shown in Figure 11.

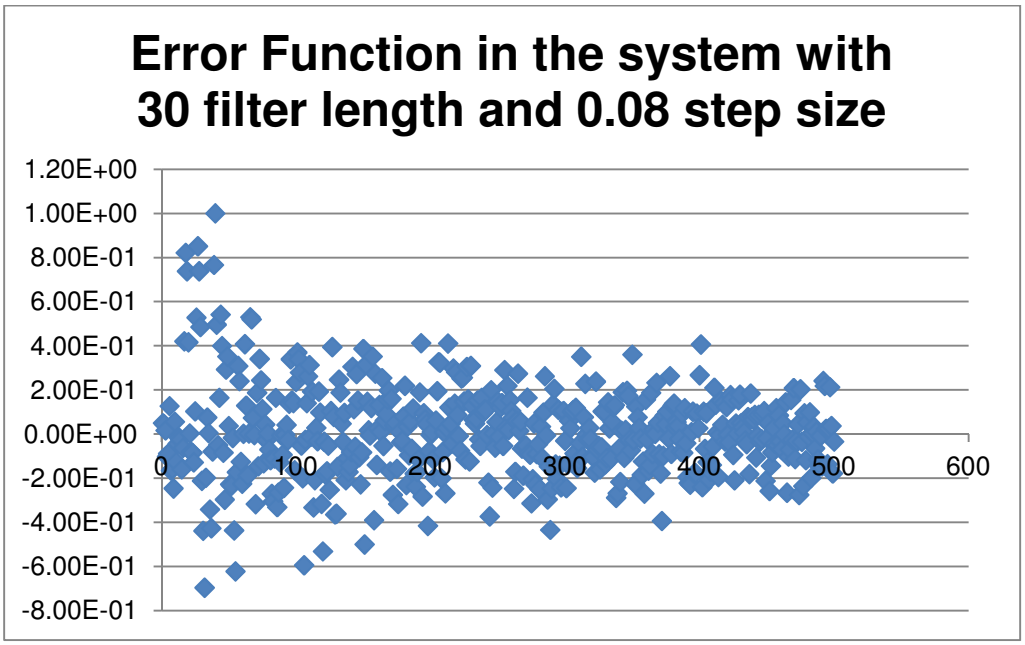

Figure 11: Error Function in the system with 30 filter length and 0.08 step size using C language

The system works properly as shown in Figure 11; it reduces noise well and fast.

The main problem, in which I could not finish, occurs when I make a real system. The DSP board did not generate any signal, but random noises. I am sure that software part is working correctly as shown in Figure 11. The first assumption, which causes problems, is the microphones are too sensitive. Some cheap microphones capture all the signals without filtering out very small noises. I designed my noise cancellation, which can hear the only noise I generated. However, my microphones collect more than I expected.

I assume the second problem is the linking up the hardware and software. I used the "Starter program," which is used in EE 419 to develop the real time digital filters by Dr. Fred DePiero, Cal Poly State University. The starter program can be different from what I need. I used two microphones and a speaker in my system. There are two ports for a microphone; one is for mono input and another is for stereo input. The starter program may use mono a microphone port, not a stereo port, in which my system needs.

So far, I guessed those two problems can make my system not work properly. If I am allowed to have more time and money for this project, first I will get better microphones and go over the starter program.

Conclusion

The most difficult part of this project was that I had never used DSP board before and did not know how least mean square algorithm works. I could not fully understand how least mean square algorithm works. However, least mean square algorithm is just math work. Anyone

can make code for least mean square algorithm without understanding it fully as put equation in the right order. When the problem occurs, I tried to go over program of the given “Starter program” partially. However, the starter program is not simple program, which I understand in a short time.

Specifications

The main object for this project is to design active noise cancellation system, which reduce noises as much as possible with using adaptive filters. It should reduce noises very fast. I assume the reduction time can be calculated by program time multiplied by how much iteration to be stable divided by sample rate, which is 48000 samples per a second.

Parts List and Costs

DSP board and Matlab should be purchased for designing this noise cancellation system project. Fortunately, I could use DSP board and Matlab in laboratory in Cal Poly EE department.

Two mini microphones:	5 dollars for two microphones
Headphone:	25 dollars
Audio splitter, stereo to two monos:	5 dollar
Two audio combiner, Mono to stereo:	10 dollar
Total:	45 dollar

Schedule - Time Estimates

I had total 20 weeks to work on this project.

First week to seventh week: Collect and understand information of adaptive filters and noise cancellation system.

Simulate least mean square algorithm in Matlab.

Eighth week to twelfth week: Design least mean square algorithm in C language.

Thirteenth to eighteenth week: Test the noise cancellation system on DSP board.

Fix any problems shown during testing.

Improve the system by changing some value of variables.

Program Listing

- Code of Matlab
- Testing code for least mean square algorithm in C language
- Actual running code for least mean square algorithm in C language

Hardware Configuration/Layout

It is shown in Figure 3.

Appendix – Matlab Code

```
x  = randn(1,500);     % Input to the filter
b  = fir1(30,0.5);     % FIR system to be identified
n  = 0.1*randn(1,500); % Observation noise signal
d  = filter(b,1,x)+n;  % Desired signal
mu = 0.04;             % LMS step size.
ha = adaptfilt.lms(30,mu);
[y,e] = filter(ha,x,d);
subplot(2,1,1); plot(1:500,[d;y;e]);
title('System Identification of an FIR Filter');
legend('Desired','Output','Error');
xlabel('Time Index'); ylabel('Signal Value');
subplot(2,1,2); stem([b.',ha.coefficients.']);
legend('Actual','Estimated');
xlabel('Coefficient #'); ylabel('Coefficient Value');  grid on;

%This will plot the result of least mean algorithm on a graph
```

Appendix – Code in C language for least mean square algorithm

```
/*
 * Here are two functions student_diffeq_initialize_stereo,
student_diffeq_evaluate_stereo
 * The function, student_diffeq_initialize_stereo, will be called
only once at the start of execution
 * The function, student_diffeq_evaluate_stereo, will be called
for every sample
 * student_diffeq_evaluate_stereo contains the information of
Least mean square algorithm
 * and keep updating coefficients.
 */


#include "dsk5416.h"

#include "cpee_student_diffeq_stereo.h"



/* place any global variables here... */
   int   N;                        /* FIR Filter Coefficient */
```

```
    int   i,j;                          /* Instant variable for
loops */
    int   y_answer;
    float u, pi;
    int   e[30];
    int   x[30], d[30], y[30];
    int   W[30][30];




/* this function is called ONCE, at the start of execution */
/*  it is useful for initializing filter coefficients, or other
variables... */
void student_diffeq_initialize_stereo()
{
    /* initialize global variables as desired... */
    N=100;                          /* FIR Filter Coefficient */
    u=.04;                          /* Steepness */
    pi=3.1415926;



    // adjust sample rate, if desired
```

```
    //  acceptable values: 48000, 24000, 12000, 8000
    diffeq_set_sample_rate(48000);

    return;
}



/*  this function is called once for EVERY SAMPLE */
/*  it should be used to evaluate your difference equation, or
other processing */
/*  input and output values are 16-bit signed integers */
void student_diffeq_evaluate_stereo(Int16 left_input,Int16
right_input,
                        Int16 *left_output,Int16 *right_output)
{
    /* evaluate difference equation here... */

    x[N] = left_input;                      /* First microphone
connected to x[N] */
    d[0] = right_input;                    /* Second microphone
connected to d[0] */
```

```
for (i = 0; i < N; i++){
    x[N - i - 1] = x[N - i];
    d[i + 1] = d[i];
}

for (i = 0; i < N; i++){
    for (j = 0; j < N - 1; j++) {
        W[i][j + 1] = W[i][j] + 2 * u * e[i] * x [i];
    }
}

for (i = 0; i < N; i++){
    j = 0;
    y_answer = 0;
    while(j <= i){
        y_answer += W[j][i] * x[i - j];
        j++;
    }
    y[i] = y_answer;
}

for (i = 0; i < N; i++){
```

```
        e[i] = d[i] - y[i];
    }

    /* return current output */
    *left_output = left_input;
    *right_output = left_input;
}

//Generate a constant sound using in Matlab
//f=250; tt=7; soundsc(sin(2.*pi.*f.*[0:1/44100:tt]),44100);
```

References

http://en.wikipedia.org/wiki/Least_mean_squares_filter

http://dsp-book.narod.ru/DSPMW/18.PDF

http://ese.wustl.edu/ContentFiles/Research/UndergraduateResearch/CompletedProjects/WebPages/sp10/Keller/Senior_Project_Report.pdf

http://cwww.ee.nctu.edu.tw/course/asp/ASP04.pdf