Active noise reduction by the modified LMS algorithm: Board level experiments

Nirav Desai
Assistant Professor
Department of ECE
ITM Universe, Vadodara, Gujarat, India
desai.nirav.12.09@gmail.com

***Abstract*—The Least Mean Square algorithm has been used for estimation of auto regressive processes before [1]. Here in, a modified LMS algorithm is presented that can be used for estimation of processes with high correlation. Board level experiments are carried out to test the effectiveness of the algorithm in active noise cancellation. Results are summarized in the end.**

***Index Terms*—Least Mean Square Algorithm, Active Noise Reduction, Real Time Embedded Systems, Arduino Due**

I. INTRODUCTION

Subsequent to the publication of the paper "Active Noise Reduction by the modified filtered x-LMS algorithm with online secondary path modelling [2]" in the Recent Trends in Electrical, Electronics and Communications Engineering Conference at ITM Universe, Vadodara in January 2014, board level experiments were carried to verify the validity of the algorithm and its performance in real time environment. Results of these experiments are presented herewith.

II. SYSTEM BLOCK DIAGRAM

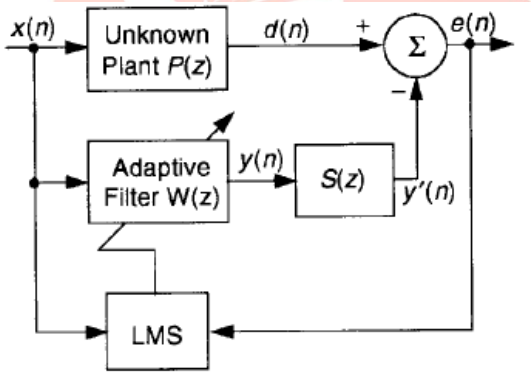

Figure 1: System block diagram for active noise cancellation by the LMS algorithm

The above figure indicates the system model for active noise cancellation by the LMS algorithm. The input noise process x(n) is fed to the adaptive filter W(z) which will be used to estimate the next sample value based on previous values. If the signal correlation is known, it can be used for tap weight update as was shown in [2]. For steady state solutions, the gradient of error is zero and thus gradient descent algorithms do not require feedback in the optimal case. For a dynamically varying environment, where the correlations are not strong, the feedback error signal is used to update the step size of the filter updates. This part of the block diagram has not been implemented here. The update equations are given below [2]:

$$y(n) = a1 \cdot x(n-1) + a2 \cdot x(n-2) + a3 \cdot x(n-3) \dots (1)$$

$$d(n) = x(n) \dots (2)$$

$$e(n) = d(n) - y(n) = x(n) - a1 \cdot x(n-1) - a2 \cdot x(n-2) - a3 \cdot x(n-3) \dots (3)$$

$$E[e(n) \cdot e(n)] = E[\{x(n) - a1 \cdot x(n-1) - a2 \cdot x(n-2) - a3 \cdot x(n-3)\} \cdot \{x(n) - a1 \cdot x(n-1) - a2 \cdot x(n-2) - a3 \cdot x(n-3)\}] \dots (4)$$

$$E[e(n)^2] = \sigma_x^2 - a1R_{xx}(-1) - a2R_{xx}(-2) - a3R_{xx}(-3) - a1R_{xx}(-1) + a1^2R_{xx}(0) + a1a2R_{xx}(-1) + a1a3R_{xx}(-2) + a2a3R_{xx}(-1) + a2^2R_{xx}(0) + a1a2R_{xx}(-1) - a2R_{xx}(-2) - a3R_{xx}(3) + a1a3R_{xx}(2) + a2a3R_{xx}(1) + a3^2R_{xx}(0) \dots \quad (5)$$

$$\frac{\partial E[e(n)^2]}{\partial a1} = (2a2 - 2)R_{xx}(-1) + 2a1R_{xx}(0) + 2a3R_{xx}(2) \dots (6)$$

$$\frac{\partial E[e(n)^2]}{\partial a2} = 2a2R_{xx}(0) + (2a1 + 2a3)R_{xx}(-1) - 2R_{xx}(-2) \dots (7)$$

$$\frac{\partial E[e(n)^2]}{\partial a3} = 2a3R_{xx}(0) + 2a2R_{xx}(1) + 2a1R_{xx}(2) - 2R_{xx}(3) \dots (8)$$

$$-\nabla\overrightarrow{a1} = -2R_{xx}(0)\overrightarrow{a1} - R_{xx}(1)\left(2\overrightarrow{a2} - 2\widehat{a1}\right) - 2R_{xx}(2)\overrightarrow{a3} \ldots (9)$$
$$-\nabla\overrightarrow{a2} = -2R_{xx}(0)\overrightarrow{a2} - R_{xx}(1)\left(2\overrightarrow{a3} + 2\overrightarrow{a1}\right) + 2R_{xx}(2)\widehat{a2} \ldots (10)$$
$$-\nabla\overrightarrow{a3} = -2R_{xx}(0)\overrightarrow{a3} - R_{xx}(1)\left(2\overrightarrow{a2}\right) - R_{xx}(2)\left(2\overrightarrow{a1}\right) + 2R_{xx}(3)\widehat{a3} \ldots (11)$$

III. Experimental Setup

Human speech has a bandwidth of about 3kHz and ears can hear frequencies up to 20kHz. Thus, any digital algorithm to modify sound would have to operate at a rate of 40ksamples/s giving a sample time of 25 micro seconds. The Arduino Uno board has a clock frequency of 16MHz with an ADC clock obtained by dividing this by a pre-scale factor of 128, 64, 32 or 16. At a pre-scale factor of 16, the ADC clock would be 1MHz and it would take 7 clock cycles for 1 conversion giving 77kilosamples/sec ADC sampling rate. While this would be acceptable for the present application, the 16 MHz clock would slow down the subsequent algorithm.

In active noise reduction as given in [2], the goal is to compute input correlations from sampled data and update the filter co-efficients adaptively to make an estimate of next sample value based on previous values. This computation has to be completed in under 25 micro seconds for the present audio noise reduction algorithm and thus requires a very high speed clock. The Arduino Due with the ARM Cortex M3 processor clocked at 84 MHz was chosen for this purpose. The ADC performance is not a limitation, however it should be noted that the Arduino Due has an ADC with 12 bit resolution giving 4095 levels in a voltage range of 0 to 3.3 Volts. The default ADC pre-scale factor was selected in this case and the start-up time parameter was modified to give an ADC sampling period of around 4 micro seconds. Since this is the minimum time resolution measurement possible with Arduino Due, errors in measurement of sampling period are possible. The micros() function in the Arduino Due library was used to measure sample time.

Below is a sample code on the execution of the modified Least Mean Square algorithm:

```
for(i=8;i<100;i++)
 {
  start_times[i]=micros();

  values[i]=analogRead(A2);

  //this scales the read value to the corresponding voltage
  values_f[i]=3.3*(float)values[i]/4095;

  //computation of auto correlation function in a recursive manner
  r[i][0] = ((i-1)*r[i-1][0] + values_f[i]*values_f[i])/(i);
  r[i][1] = ((i-1)*r[i-1][1] + values_f[i]*values_f[i-1])/(i);
  r[i][2] = ((i-1)*r[i-1][2] + values_f[i]*values_f[i-2])/(i);
  r[i][3] = ((i-1)*r[i-1][3] + values_f[i]*values_f[i-3])/(i);


  //tap co efficient update
  a[i][0] = a[i-1][0] + 0.5*delta*(-2*(a[i-1][0]*r[i][0]+ 2*r[i][1]));
  a[i][1] = a[i-1][1] + 0.5*delta*(-2*(a[i-1][1]*r[i][0]+ 2*r[i][2]));
  a[i][2] = a[i-1][2] + 0.5*delta*(-2*(a[i-1][2]*r[i][0]+ 2*r[i][3]));

  //calculated estimate
  y[i] =  (a[i][0]*values_f[i-1] + a[i][1]*values_f[i-2] + a[i][2]*values_f[i-3]);

  filter_mag=sqrt(a[i][0]*a[i][0]+a[i][1]*a[i][1]+a[i][2]*a[i][2]);
  y[i]=(-1)*y[i]/filter_mag;


  ys[i]= (unsigned long int)(((y[i])*4095)/5.0);

  analogWrite(DAC0, ys[i]);

  stop_times[i]=micros();

  if(i==100)
  {
   for(j=1;j<8;j++)
    a[j][0]=a[i][0];
    a[j][1]=a[i][1];
    a[j][2]=a[i][2];
    r[j][0]=r[i][0];
    r[j][1]=r[i][1];
    r[j][2]=r[i][2];
```

```
    r[j][3]=r[i][3];
  }
 }
```

The above code is designed to read data from the input port, compute the correlation values at delays of 0, 1, 2 and 3 samples, update the filter tap coefficients using the recursive version of the modified LMS algorithm and compute an estimate for next sample value which is put on the output port. The recursive LMS used here is based on the reference paper [2].

IV. MICROPHONE INTERFACE

An electret microphone was purchased from Sparkfun Electronics and used for sampling sound signals. This microphone was coupled to an inverting amplifier, which level shifted the input signal to the mid-point of the dynamic range of the ADC of 3.3V. This also inverted the recorded sounds so that the algorithm will estimate the required anti-noise signal directly. The figure below indicates the setup of the experiment with the microphone interfaced to the microprocessor board:

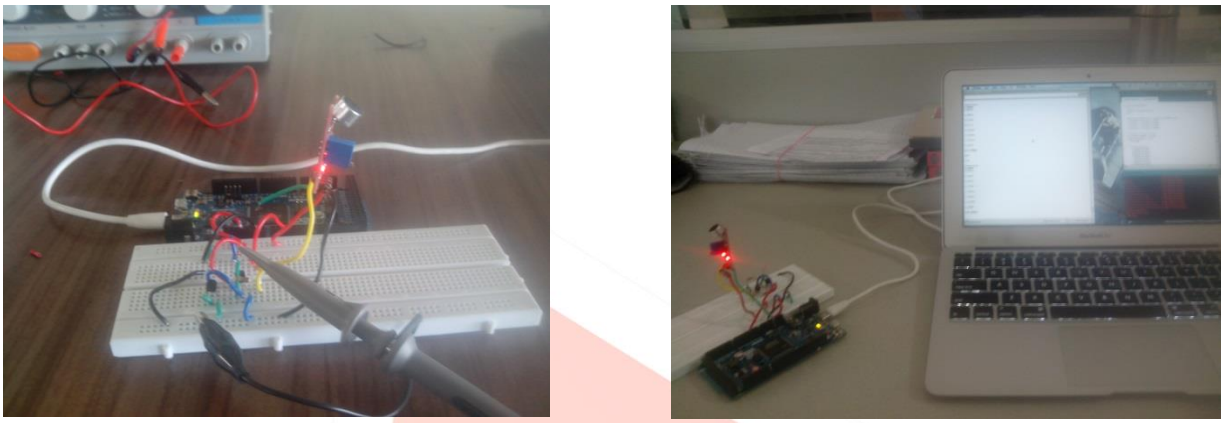

Figure 2: Experimental set-up of electret micro phone interfaced to the Arduino Due processor board. The Arduino Due connects to the laptop through the USB port, which is used to transfer the code on to the processor board and collect the data from the processor which is displayed in the serial monitor.

V. OBSERVATIONS

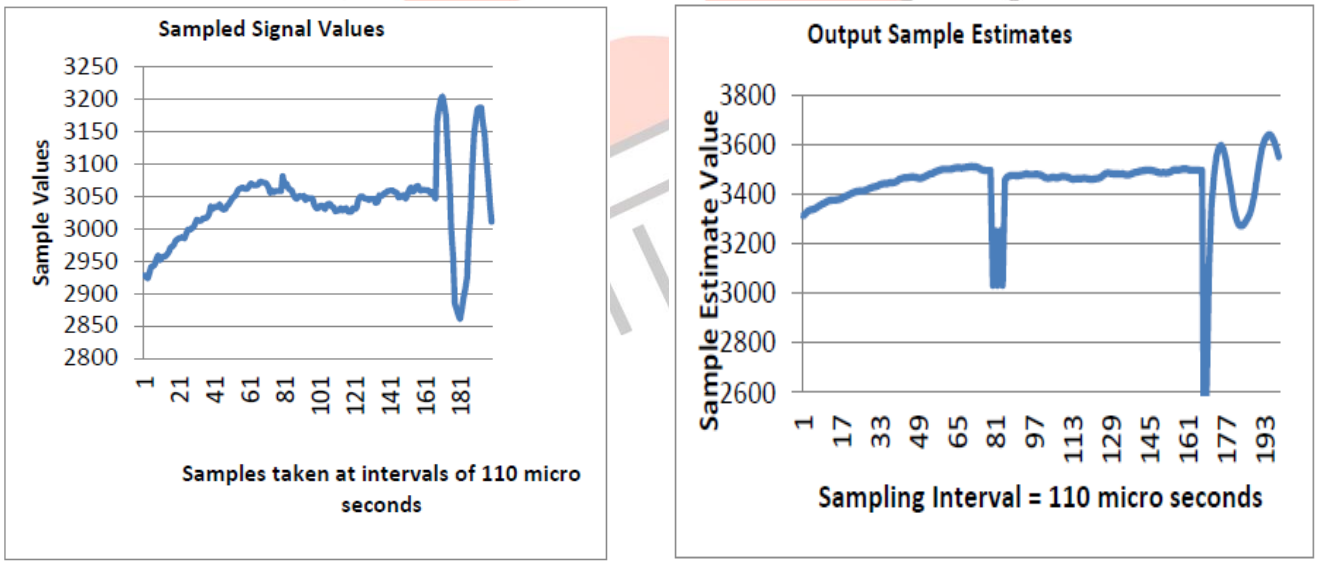

Figure 3: [L]Sample values from the input ADC of Arduino Due processor board. These are unsigned long integer data format read directly from the input port. Run time for each loop of the code is approximately 110 micro seconds. [R]Sample estimates computed from the algorithm. The algorithm operates on floating point values which are then typecast into unsigned long integer format to output on the DAC port. The discontinuities at samples 81 and 169 are due to the buffer refresh, which could be eliminated with a continuous buffer update.

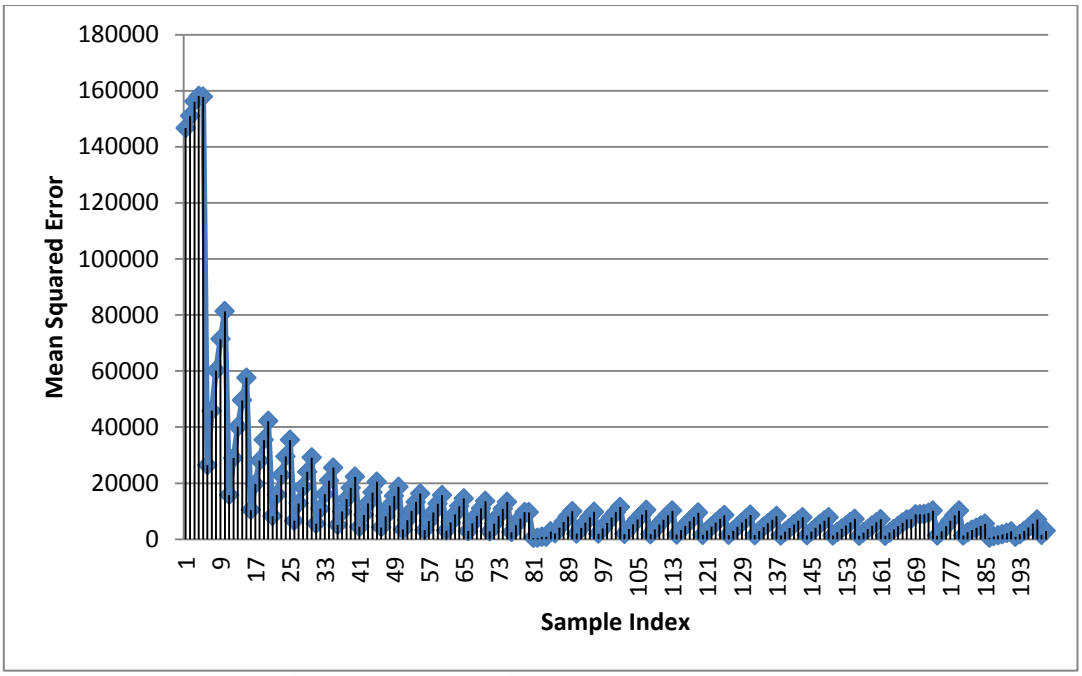

Figure 4: Mean squared error in sample estimates as compared to the actual sample values. The error reduces as the algorithm runs for a longer period of time.

VI. CONCLUSIONS

The experimental evidence indicates that for ambient sounds having high correlation (human voice in this case), the algorithm converges to a steady state filter estimate. Since the filter tap weight updates are calculated from the gradient of the error function with respect to tap weights, the steady state indicates zero gradients or a local or global minimum for the algorithm. Since the error is reducing, this has to be the minimum. The calculation of error from real time data has not been carried out here and this part of the control loop is still missing. However, as per the paper, the error magnitude is necessary only for the step size update in situations where correlations are not very strong and we need to adapt to a very dynamically changing environment. This will be carried out in the subsequent experiments. This experiment proves the stability of the algorithm in estimating filter co-efficients for highly correlated random processes.

REFERENCES

[1] Farhang-Boroujeny, B., "Fast LMS/Newton algorithms based on autoregressive modeling and their application to acoustic echo cancellation," Signal Processing, IEEE Transactions on , vol.45, no.8, pp.1987,2000, Aug 1997 doi: 10.1109/78.611195

[2] Nirav A. Desai, "Active Noise Cancellation by the modified Filtered X-LMS algorithm with online secondary path modeling," Recent Trends in Electrical, Electronics and Communications Engineering Conference, ITM Universe, January 2014, Special issue of International Journal of Engineering Development and Research

[3] ADC characterization on Arduino Due: http://www.microsmart.co.za/technical/2014/03/01/advanced-arduino-adc/

[4] Arduino User Forums: http://forum.arduino.cc/index.php/topic,6549.0.html