Frequency-Domain Adaptive Noise Cancellation

Suhas Ranganath[1], Rushil Anirudh[2]
Department of ECEE, Arizona State University

***Abstract*—This article demonstrates the use of an Adaptive Filter in Noise Canceling applications.**

I. Introduction

DISCRETE-TIME filters are used everywhere in signal processing applications. Filters are used to achieve spectral characteristics , and to reject unwanted noise or interferences. The concept of Adaptive Filters is to alter parameters of a filter according to a minimization algorithm based on a reference signal. Adaptive filters can adjust its coefficients according to the changing spectral and temporal signal characteristics.

The adapting process involves the use of an error function, which is a criterion for optimum performance of the filter (for example, minimizing the noise component of the input), to feed an algorithm, which determines how to modify the filter coefficients to minimize the cost on the next iteration.

As the power of digital signal processors has increased, adaptive filters have become much more common and are now routinely used in devices such as mobile phones and other communication devices for example - camcorders and digital cameras, medical monitoring equipment etc.

One of the earliest applications mentioned in [1] used for Adaptive Filtering is the cancellation of the maternal ECG waveform in fetal electrocardiography. In this case, the primary signal is obtained from an ECG pickup on the mother's abdomen. It consists of a weak, typically high rate, ECG waveform produced by the fetal heartbeat in the presence of large-amplitude lower-rate ECG puises at the maternal heart rate. This signal is also usually corrupted by the presence of muscle noise and 60Hz power line pickup. In the majority of cases, the weak, desired fetal signal is not evident upon visual inspection of this signal. A reference signal set xr(k) is obtained by placing K ECG sensors on the mothers chest. The maternal heartbeat signal observed at these sensors is strongly correlated with that in the abdominal lead but usually has a substantially different time waveshape caused by the difference in transfer function between the source (heart) and the sensors. Due to their distance from the fetus, the fetal heartbeat signal is essentially absent in the chest lead signals.

The primary input signal (acquired by a microphone at the source) consists of a desired signal sequence x(n) (here a speech sample) corrupted by additive noise $w_1(n)$. A second microphone is placed at the source of the noise and this serves as the second input to the adaptive filter, as w(n). The function of the adaptive filter here is to model the acoustic path taken by the noise. This is performed by calculating an

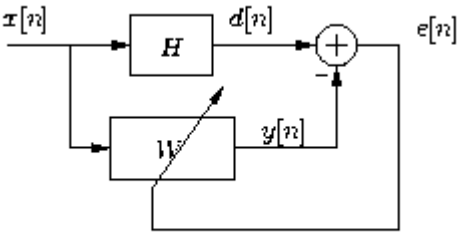

Fig. 1: System Identification Block Diagram[2]

error signal which is given by

$$\mathbf{e(n)=x(n)-w(n)} \tag{1}$$

This error signal is fed back to the filter as shown in figure[1], so that its coefficients can be updated for the next iteration. Ideally, the error signal should be able to cancel out the noise completely giving only clean speech. Because of the room acoustics the w(n) cannot be cancelled and the output of the filter w'(n) approximates it. The FIR filter B(z) is needed to model the reflections and delays in the acoustic path between the noise source and the primary microphone.

The acoustic path is time varying because of dynamics, movement, reflections in the room. Since the acoustic path is time varying, we need to estimate it continuously. For this reason an adaptive FIR Filter with transfer function B(z) is required. The coefficients of B(z) can be estimated using a time or a frequency-domain adaptive algorithm.

The coefficient update relation is a function of the error signal and is given by:

$$h_{n+1} = h_n[i] + \tfrac{\mu}{2}(-\tfrac{\partial(e)^2}{\partial h_n[i]})$$

The term inside the parentheses represents the gradient of the squared-error with respect to the i^{th}coefficient. The gradient is a vector pointing in the direction of the change in filter coefficients that will cause the greatest increase in the error signal. Because the goal is to minimize the error, however, the above equation updates the filter coefficients in the direction opposite the gradient; that is why the gradient term is negated.The constant μ is a step-size, which controls the amount of gradient information used to update each coefficient. After repeatedly adjusting each coefficient in the direction opposite to the gradient of the error, the adaptive filter should converge; that is, the difference between the unknown and adaptive systems should get smaller and smaller.

1-1202034727 2-1203006917, Graduate Students

Upon derivation [3], the final LMS coeficient update equation is

$$h_{n+1}[i] = h_n[i] + (\mu)(e)x[n-i]$$

The step-size μ directly affects how quickly the adaptive filter will converge toward the unknown system. If μ is very small, then the coefficients change only a small amount at each update, and the filter converges slowly. With a larger step-size, more gradient information is included in each update, and the filter converges more quickly; however, when the step-size is too large, the coefficients may change too quickly and the filter will diverge. (It is possible in some cases to determine analytically the largest value of μ ensuring convergence.)

II. Frequency Domain Adaptive Filtering

Adaptive Filtering in the frequency domain can be accomplished by the fourier transformation of the input signal and independent weighting of the contents of each frequency bin. The frequency domain filter performs similarly to a conventional adaptive transversal filter but promises a significant reduction in computation when the number of weights $\geq$ 16.[4]

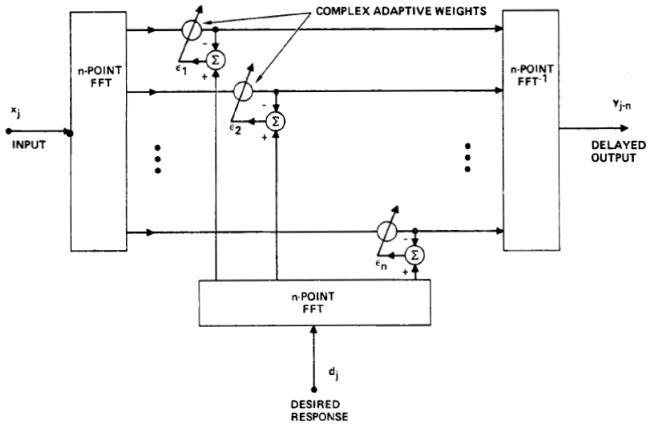

Fig. 2: LMS Adaptive Filtering in the frequency domain [4]

Adaptive filters are used in a wide variety of applications, including statistical prediction, interference canceling, array phasing, and channel equalization in communication systems.In the normal implementation the outputs of a tapped delay line are weighted and summed under control of a recursive algorithm to form the filter output. The most commonly used algorithm at the present time is the Widrow-Hoff least-mean-square (LMS) algorithm [5]. The frequency-domain LMS as shown in figure[2] is similar in general configuration to the time domain filter. The input signal x_j and the desired response d_j however are accumulated in buffer memories to form n-point FFT's. Each of the FFT outputs comprises a set of n complex numbers. The desired response transform values are subtracted from the input transform values at corresponding frequencies to form n complex error signals There are n complex weights, one corresponding to each spectral bin. Each weight is independently updated once for each data block. The weighted outputs are not summed but are fed to an inverse FFT operator to produce the output signal y_{j-n}, delayed by the number of samples n in the input data block.

Since each weight is adapted only once for each n-point data block, the number of adaptations required to obtain output data similar to those obtained with the conventional time-domain filter is reduced by a factor of n. The value of the adaptive constant μ may accordingly be increased by a factor of n. The larger value of μ, corresponding to less frequent adaptation, permits a lowering of the weight resolution requirements for the frequency-domain filter by a factor of n, so that the number of bits required to store each weight can be reduced by $\log_2 n$, simplifying the weight update arithmetic.

The best indicator of the advantage of using FDAF is seen in the number of multiplier operations required to produce a given amount of data. To produce n output data points with the conventional filter requires n^2 adaptations and $2n^2$ real multiplies. To produce the same output with the frequency-domain filter requires $(\frac{3n}{2})\log_2 n$ complex multiplies for the 3 n-point FFT's and 2n complex multiplies for the complex weighting and weight updating. The ratio of the complex multiplies required by the frequency-domain filter to real multiplies required by the conventional filter is thus :

$$\frac{ComplexMultiplies}{RealMultiplies} = \frac{(\frac{3n}{2})\log_2 n + 2n}{2n^2} = \frac{3\log_2 n + 4}{4n} \tag{2}$$

While the reduction in computational complexity may not be evident for small values of n, for larger values there is significant difference.

III. MATLAB Code implementing FDAF

This section shows the MATLAB code used to implement the Frequency Domain Adaptive Filtering for the Noise Cancelling application in Speech. The speech signal is processed frame by frame with N samples/frame, where N is predefined by the user. An optimal value of N would be a value equal to the filter length being used in the system [2]. If N was larger then there would be a few weights in the filter which would remain unused and hence wasted. If N was smaller than the frame length then it would result in redundancy. This version of FDAF implements circular convolution, a more effective way would be to perform Linear Convolution using Overlap and save method. The MATLAB implementation is shown in the Appendix - B

A. Use of Other Windows

Frame by frame processing of the speech signal is equivalent to multiplying the signal by a rectangular window. Here, the effects of using a Kasier window with a varying β are explored.

```
%Using the kaiser window to select the n points of the
current frame
s1(n)=diag(samp1(n))*kaiser(N,beta);
s2(n)=diag(samp2(n))*kaiser(N,beta);
%Transforming to the frequency domain
S1(k,:)=fft(s1(n)); S2(k,:)=fft(s2(n));
```

```
%Calculating the Error Function
S2_diag=diag(S2(k,:));
E(k,:)=S1(k,:)-B(k,:)*(S2_diag)';
%The recursion equation to calculate coefficients of Transfer
function B(z)
B(k+1,:)=B(k,:)+2*mu*E(k,:)*(S2_diag');
%Getting back to time-domain
e(k,:)=ifft(E(k,:));
```

IV. Results

These values are constructed with β=1 i.e the normal case with a rectangular window.

N	$SNR_{\mathbf{dB}}^{improvement}$	Subjective Rating	Best μ
4	2.12269	2	0.01
16	3.3426	2.5	0.0065
64	7.2628	3	0.006
128	9.6159	3.5	0.004
256	12.0084	4	0.0045

The following table shows the improvement in the SNR with increasing values of β. The best value of μ was chosen and used.

N	$SNR_{\mathbf{dB}}^{improvement}$	Subjective Rating	β
128	11.7385	4	2
128	12.4830	4.5	3
128	12.2034	3.5	4
256	14.2154	4	2
256	14.8727	4.5	3
256	14.27	4	4

For the speech signal corrupted by regular music

N	$SNR_{\mathbf{dB}}^{improvement}$	Subjective Rating	Best μ
4	8.9120	2.5	0.01
16	10.5898	3	0.03
64	9.4909	2.5	0.007
128	7.8469	2	0.003
256	7.7925	2	0.003

Table showing changes in SNR for varying values of the Kaiser parameter β

N	$SNR_{\mathbf{dB}}^{improvement}$	Subjective Rating	β
128	8.79	3.5	2
128	8.6920	3	3
256	8.3341	3	2
256	8.1149	3	3

V. Convergence Curves

The Convolution Curves for different Values of N are shown here.

Intuitively we would expect the convergence rate to be higher with an increased value of N, since N is the frame length and the filter length. This can be observed from the above graphs. The convergence rates are plotted for the best value of μ in each case.

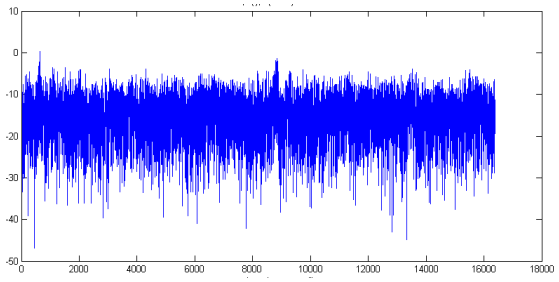

Fig. 3: Plot of $|E|^2$ vs Iteration Count for N=4

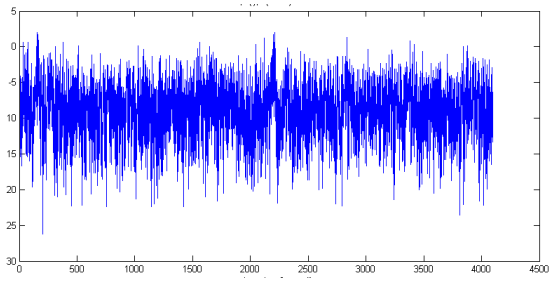

Fig. 4: Plot of $|E|^2$ vs Iteration Count for N=16

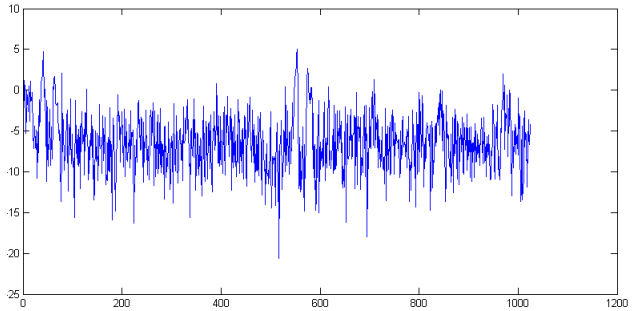

Fig. 5: Plot of $|E|^2$ vs Iteration Count for N=64

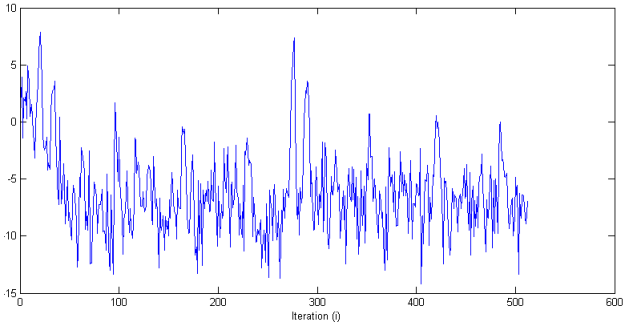

Fig. 6: Plot of $|E|^2$ vs Iteration Count for N=128

VI. Comparison of Convergence Rates

A study of different convergence rates vs N was performed. Since the parameter μ is a step size, it determines how fast or slow the algorithm converges. A large μ will lead to quicker convergence. However, a very large value may result in instability of the algorithm and the coefficients of the filter B(i) can become unbounded (overflow). Here convergence rates for a slow and fast convergence are shown for N=64 and N=128.

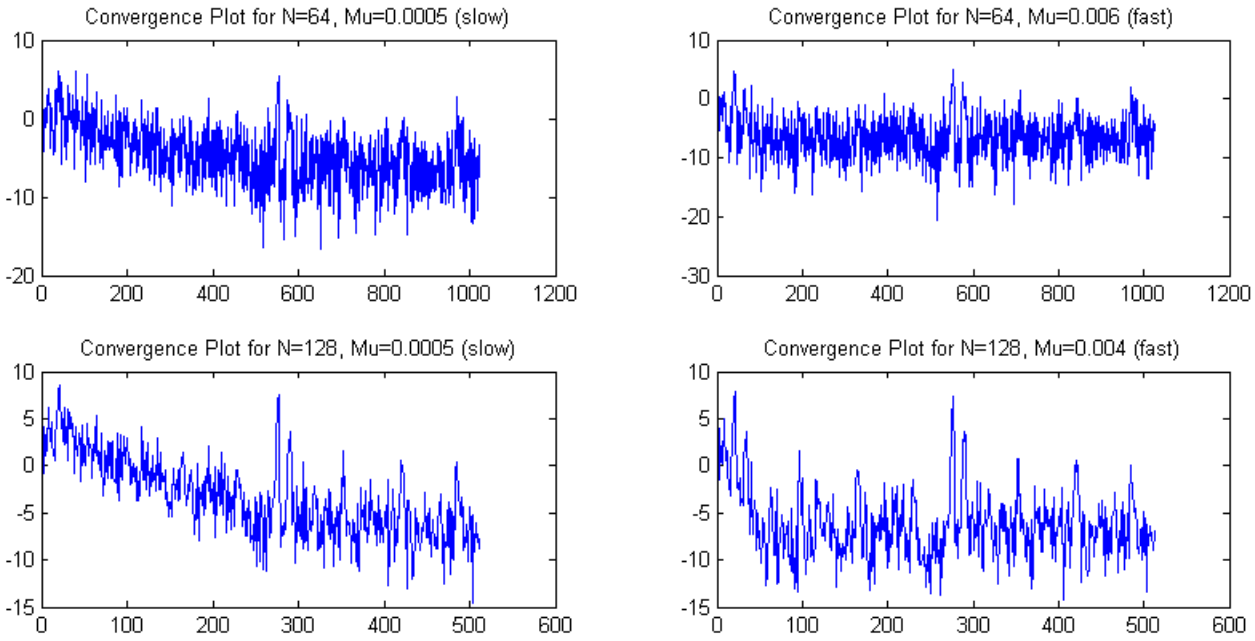

Fig. 8: Slow and Fast Convergence Rates for N=64 and N= 128

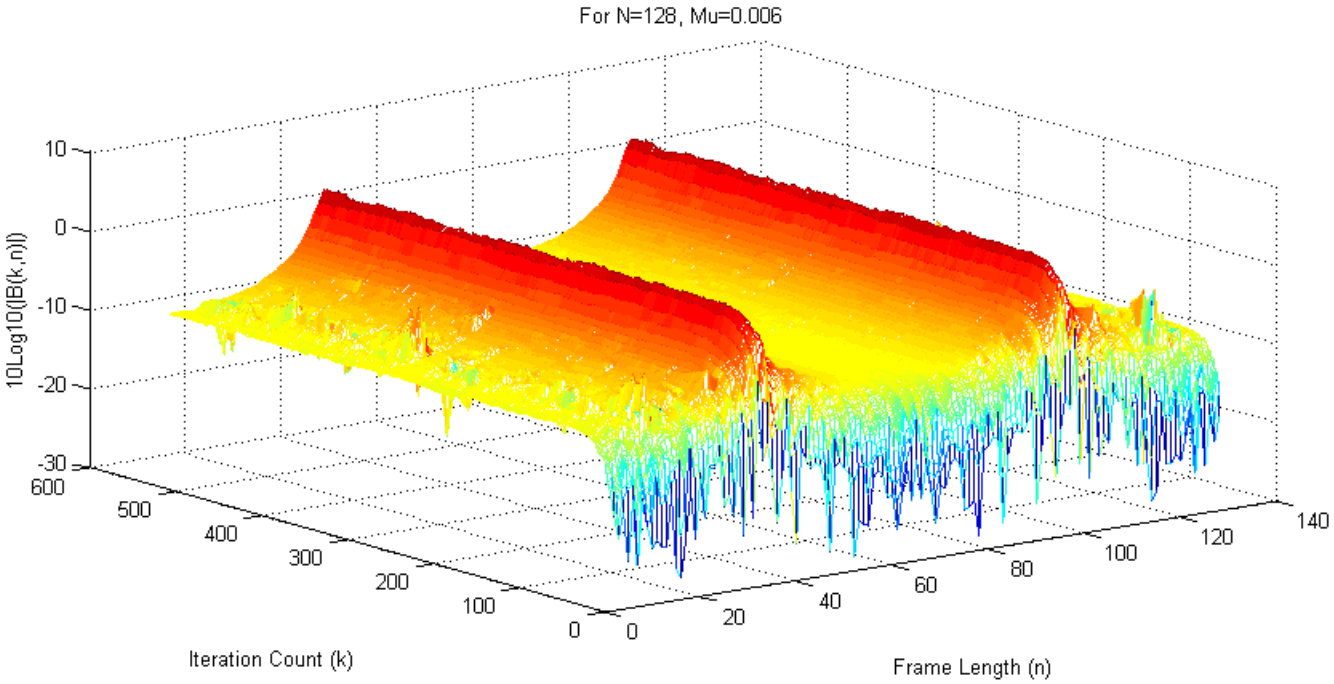

Fig. 9: 3-D Mesh Plot

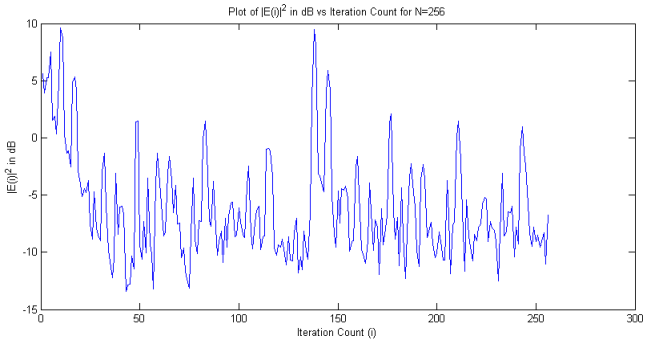

Fig. 7: Plot of $|E|^2$ vs Iteration Count for N=256

VII. Convergence Curves for Speech Corrupted by Music

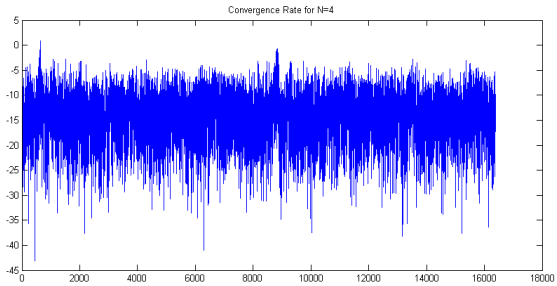

Fig. 10: Plot of $|E|^2$ vs Iteration Count for N=4

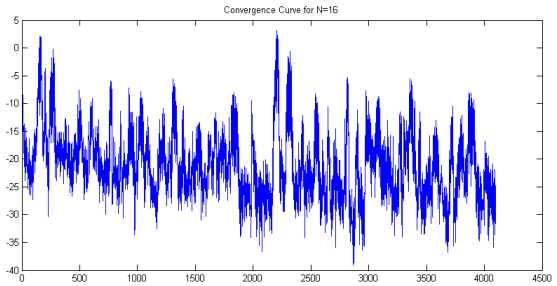

Fig. 11: Plot of $|E|^2$ vs Iteration Count for N=16

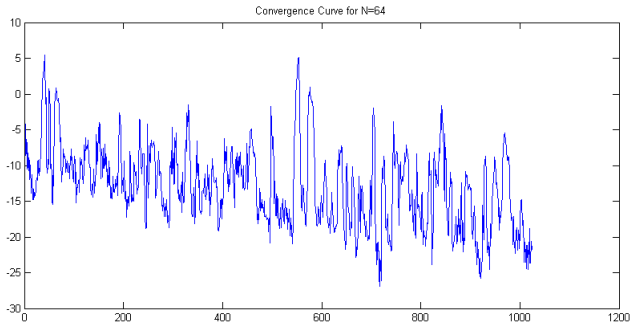

Fig. 12: Plot of $|E|^2$ vs Iteration Count for N=64

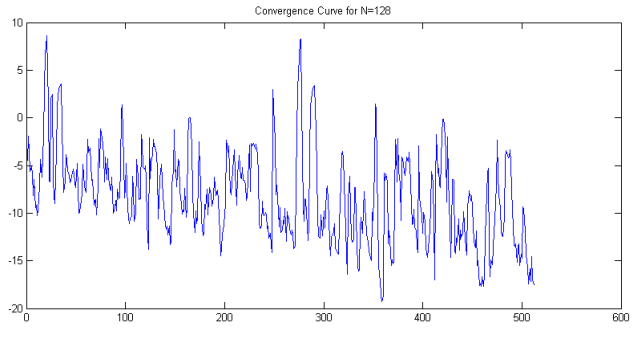

Fig. 13: Plot of $|E|^2$ vs Iteration Count for N=128

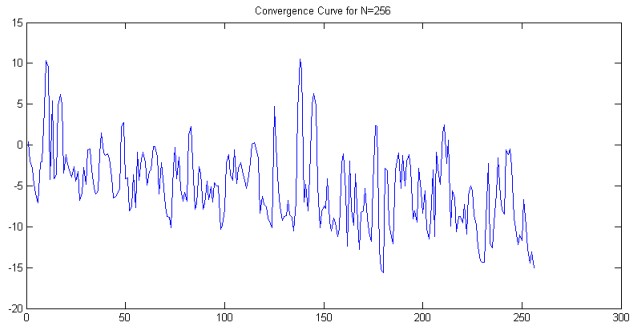

Fig. 14: Plot of $|E|^2$ vs Iteration Count for N=256

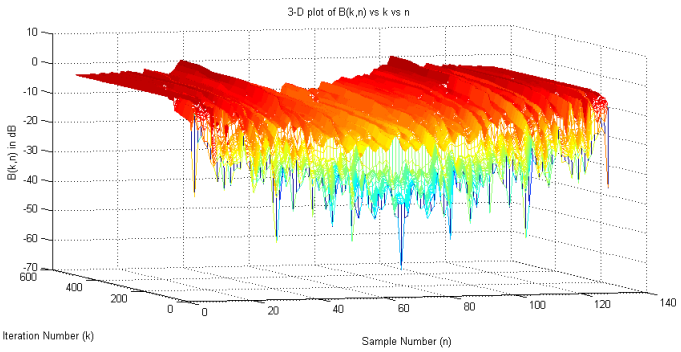

Fig. 15: 3-D plot of B(k,n) vs k vs n

VIII. Remarks

Q. What do the frequency components B(k) represent in terms of filter properties?

A. They represent the frequency domain filter coefficients.

Q. What is the effect of N on the SNR?

A. As N increases SNR for noise signal increases while it decreases for the music signal. Probably because of the uniform presence of the noise signal. While the music signal is present in select frequencies. Hence does not get filtered as well.

- [Q.]How is the size of the FFT (N) related to the order of the filter?

A. If the order of the filter is less than size(fft(N)) redundancy is introduced,if it is greater than size of fft(N) the coefficients are wasted. So the filter size should be equal to the size of the fft(N).

Q. What is the effect of μ on the quality of speech (small mu vs big mu)?
A. As μ increases upto a certain level,quality of speech increases then it decreases because the coefficients of B(z) become unbounded after a particular value causing the loss of filtering action.
Q. How does the SNR correlate with speech quality?
A. SNR and quality of speech are correlated in general but not always. For Example, upon performing Linear Convolution the SNR value was found to be lower than that obtained with circular convolution even though the speech quality is better. This is mostly because the last frame is clipped in the processing of the algorithm.
Q. Did you observe any artifacts in your subjective evaluations?
A. Initially the noise is high but it decreases as the signal progresses,albeit a clicking sound. The clicking is significantly reduced when linear convolution is used by the overlap and save method. It is also observed that the clicking sound is reduced with increasing value of the Kaiser Parameter upto a certain value beyond which the signal begins to get distorted.
Q. Why does the algorithm minimize e(n)?
A. e(n) is minimised beacuse in the speech signal, maximum noise reduction is obtained when the difference between the noise currupted signal and the noise is minimum.
Q. How do you determine the impulse response of the filter from B(i)
A. Impulse response can be calculated by taking the Inverse Fourier Transform of B(K).

IX. Observations

1. An application for Adaptive Filtering for noise cancelling was demonstrated in this project.
2. The noise in the initial few frames is due to the convergence of the algorithm. Once the filter adjusts its coefficients, the noise is estimated more accurately and the error becomes minimized.
3. The use of a Kaiser window in place of the standard rectangular window further increases the quality of speech and SNR. Noise reduction is maximum for an optimal value of β.
4. Upon increasing the value of N beyond 512, a significant echo is observed in the filtered signal.
5. Use of Overlab and Save Method for Linear Convolution significantly reduces the background noise and enhances the speech quality.

X. Scope for Further Research

The LMS algorithm used here, has a slower rate of convergence when compared to Recursive Least Square (RLS) algorithms. These algorithms maybe implemented for better noise reduction[6]. Output signal could be further processed by echo cancellation devices.

XI. Appendix

A. MATLAB Code - I

```
  function [K,N,B]=anc_fall2010(N,mu,beta)
F=N; s1=zeros(1,N);s2=s1;S1=s1;S2=s2;E=s1;e=s1;f=s1;
[samp1,fs,bits]=wavread('musicmic1');
[samp2]=wavread('musicmic2');
cs=wavread('clean2');
  l1=length(samp1); l2=length(samp2);
M=min(l1,l2); K=fix(M/N); B(1,:)=zeros(1,F);
  for k = 1:K n = (1:N)+(N*(k-1));
s1(n)=diag(samp1(n))*kaiser(N,beta);
s2(n)=diag(samp2(n))*kaiser(N,beta);
S1(k,:)=fft(s1(n)); S2(k,:)=fft(s2(n));
S2_diag=diag(S2(k,:)); E(k,:)=S1(k,:)-B(k,:)*(S2_diag);
B(k+1,:)=B(k,:)+2*mu*E(k,:)*(S2_diag');
e(k,:)=ifft(E(k,:)); f(k)=10*log10(E(k,:)*E(k,:)'/N);
end
  [r,c]=size(e); e1 = reshape(e,1,r*c);
dif=[cs]-samp1; %Nc=numel(dif);
  SNR_bef=10*log10(conj(cs)'*cs/(conj(dif)'*dif));
dif2=cs-e1'; SNR_aft=10*log10((conj(cs)'*cs)/(conj(dif2)
SNR=SNR_aft-SNR_bef; [str]=sprintf('sound_out
%d',10*beta); cd 'C:'; wavwrite(e1',fs,bits,str)
plot(1:K,f)
```

B. MATLAB Code for implementation of Linear Convolution

```
  function [SNR]=anc_fall20101(N,mu,beta)
s1=zeros(1,2*N);s2=s1;S1=s1;S2=s2;E=s1;e=s1;
F=N; [samp1,fs,bits]=wavread('mic1');
[samp2]=wavread('mic2');
[cs]=wavread('clean2');
  o=cs(1:65280); q=samp1(1:65280);
  l1=length(samp1); l2=length(samp2);
M=min(l1,l2); K=fix(M/N); B(1,:)=zeros(1,2*F);
  for k = 1:K-1 n = (1:2*N)+(N*(k-1));
s1(n)=diag(samp1(n))*kaiser(2*N,beta);
s2(n)=diag(samp2(n))*kaiser(2*N,beta);
S1(k,:)=fft(s1(n)); S2(k,:)=fft(s2(n));
S2_diag=diag(S2(k,:)); E(k,:)=S1(k,:)-B(k,:)*(S2_diag);
B(k+1,:)=B(k,:)+2*mu*E(k,:)*(S2_diag');
e(k,:)=ifft(E(k,:)); f(k,:)=e(k,N+1:2*N);
g(k)=10*log10(f(k,:)*f(k,:)'); end
  [r,c]=size(f); e1 = reshape(f,1,r*c);
  dif1=o-q;
  SNR_bef=10*log10((conj(cs)'*cs)/(conj(dif1)'*dif1));
dif2=o-e1'; SNR_aft=10*log10((conj(cs)'*cs)/(conj(dif2)'
SNR=SNR_aft-SNR_bef;
  str
=sprintf('sound_out %d',10*beta); cd 'C:';
wavwrite(e1',fs,bits,str) plot(1:K-1,g)
```

References

[1] L. J. Griffiths, *"An Adaptive Lattice Structure for Noise Cancelling Applications"* IEEE Trans. 1978
[2] S. Haykin, *Adaptive Filter Theory*, 3rd Edition, Prentice-Hall, 1996.
[3] Jones, D. et al, *Adaptive Filtering: LMS Algorithm*, Connexions Web site. http://cnx.org/content/m10481/2.14/, June 1,2009.
[4] M. Dentino, J. Mc.Cool, B. Widrow *"Adaptive Filtering in the Frequency Domain" proc. IEEE(Lett.), Vol. 66, pp1658-1659*, December 1978.
[5] B. Widrow *"Stationary and nonstationary learning characteristics of the LMS adaptive filter. " proc. IEEE(Lett.), Vol. 64, pp1151-1162*, August 1976.
[6] J.G. Proakis, D.G Manolakis, *"Digital Signal Processing : Principles, Algorithms and Applications "* 4th Edition, Prentice-Hall, 2007.