Real-Time DSP Implementation of an Acoustic-Echo-Canceller with a Delay-Sum Beamformer

A Thesis

Submitted to the Faculty

of

Rose-Hulman Institute of Technology

by

Todd Goldfinger

In Partial Fulfillment of the Requirements for the Degree

of

Master of Science in Electrical Engineering

December 2005

Copyright \bigcirc 2005 by Todd Goldfinger

ABSTRACT

Todd Goldfinger MSEE

Rose-Hulman Institute of Technology

December 2005

Real-Time DSP Implementation of an Acoustic-Echo-Canceller with a Delay-Sum Beamformer

Dr. Wayne T. Padgett

Traditional telephony uses only a single receiver for speech acquisition. If the speaker is standing away from the telephone, the signal will be weak and there will be interference sources from room reverberation. In addition, there is acoustic echo coming from the loudspeaker, which further interferes with the signal of interest. This research investigated the combination of common solutions to these problems. Electronic beamforming steered an array of microphones within software to enhance the signal power. Echo cancellation removed the echo coming from the loudspeaker. In combination these processing techniques can greatly enhance user experience.

CONTENTS

A	bbre	viation	as and Symbols	iv
\mathbf{Li}	st of	Figur	es	viii
1	Intr	oducti	ion	1
2	Acc	oustic l	Echo Cancellation	3
	2.1	Syster	n Identification for AEC	3
	2.2	Adapt	vive Algorithms	5
		2.2.1	RLS	5
		2.2.2	NLMS	6
		2.2.3	Affine Projections	7
		2.2.4	Fast Affine Projections	8
		2.2.5	Block Exact Fast Affine Projections	14
		2.2.6	Controlling Convergence	17
3	Doι	ıble Ta	alk Detection	25
	3.1	Echo (Cancellation Scenarios	25
	3.2	Detect	tion Methods	26
		3.2.1	Geigel	26

		3.2.2	Cheap Normalized Cross-Correlation (CNCR) $\ldots \ldots \ldots$	27
		3.2.3	Complexity Comparison	29
4	Dela	ay-Sun	n Beamforming	32
	4.1	Electro	onic Beamforming	32
		4.1.1	Inter-Element Spacing	33
		4.1.2	Wide Beam Response	33
	4.2	Time-I	Delay Estimation (TDE)	34
		4.2.1	Least Squares Estimation	38
	4.3	Delay	Filters	38
5	AE	C with	Beamforming	41
	5.1	Using	Poles	42
6	Rea	l-Time	e Implementation	46
	6.1	Setup		46
	6.2	Experi	iments	49
	6.3	Conclu	usion and Further Work	59
Re	efere	nces		60
Aj	ppen	dices		64
A	Fari	row Fi	lters Maple Code	65
в	Fou	rier Tr	ansform Parts	70

List of Symbols and Abbreviations

Abbreviations	description
AEC	Acoustic Echo Cancellation
AGC	Automatic Gain Control
APA	Affine Projection Algorithm
BEFAP	Block Exact Fast Affine Projection Algorithm
CGFAP	Conjugate Gradient Fast Affine Projection Algorithm
CNCR	Cheap Normalized Cross-Correlation
DTD	Double Talk Detection
DSP	Digital Signal Processor
ERLE	Echo Return Loss Enhancement
FAP	Fast Affine Projection Algorithm
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FTF	Fast Transversal Filters
GSFAP	Gauss-Seidel Fast Affine Projection Algorithm
MMSE	Minimum Mean Square Error
MSE	Mean Square Error

NLMS	Normalize Least Mean Squares
ROC	Receiver Operating Characteristic
RLS	Recursive Least Squares
RTDX	Real Time Data Exchange
RTF	Room Transfer Function or Room Impulse Response
SDA	Steepest Descent Algorithm
SIR	Signal-to-Interference Ratio
SNR	Signal-to-Noise Ratio
SWFTF	Sliding Window Fast Transversal Filter
TDE	Time-Delay Estimation
ULA	Uniform Linear Array
VIRE	Variational Impulse Response

Symbols	description
С	speed of sound
L	block length of a block adaptive algorithm
m	length of geigel detector
n	echo canceller length in coefficients
Р	projection order for APA
Т	detection statistic stretch parameter
e_n	error or AEC output
n_n	noise
s_n	near end speech

x_n	far end speech reference signal
y_n	input signal or desired signal
\hat{y}_n	filtered approximation of input signal
$\mathbf{R}_{\mathbf{xx},n}$	autocorrelation matrix
\mathbf{d}_n	cross-correlation vector
\mathbf{h}_n	room transfer function
$\mathbf{\hat{h}}_{n}$	adaptive filter representing estimation of the RTF
μ	step size or relaxation
ξ	detection statistic
σ_x^2	signal power of x
d_k	k^{th} insertion delay
f_s	sample rate (Hz)
h_{id}	ideal delay filter
L_k	pathloss to k^{th} microphone
$r_{m_k}(t)$	distance to the k_{th} microphone form the source
$s_{m_k}(t)$	signal received at the k_{th} microphone
$X_{m_k,m_{k+1}}(\omega)$	cross-power spectrum between consecutive microphones
$s_{m_k}(t)$	signal received at the k_{th} microphone
$ au_{m_k,m_{k+1}}$	k^{th} time delay between consecutive microphones
Δ	fractional sample delay
λ_k	k_{th} eigenvalue of a matrix
δ	regularization parameter
ε	normalized residual error from adaptive filter
\overline{x}	complex conjugate of x

${\cal F}$	Fourier	transform

 \doteq defined as

List of Figures

2.1	Acoustic Echo Cancellation Problem	4
2.2	Gauss-Seidel FAP	11
2.3	Misalignment Comparison	12
2.4	ERLE Comparison	13
2.5	Block Exact FAP	15
2.6	NLMS Convergence	18
2.7	BEGSFAP Convergence	20
2.8	Convergence with AGC	22
2.9	Condition Number	23
3.1	ROC	27
3.2	Comparison of Thresholding versus Proportionality	31
4.1	Array Patterns	34
4.2	Low Pass Filtering Effect of Array.	35
4.3	Cross-Spectrum Phase Plots	37
4.4	Filter Magnitude Error	40
5.1	Structure of AEC with Beamforming	42

5.2	Structure of AEC with Poles.	44
5.3	AEC with and without Poles	45
6.1	Array Setup	48
6.2	Array Gain	51
6.3	(A) Real-time ERLE of 'ah', (B) with Beamforming.	58

ACKNOWLEDGEMENTS

I would like to thank Dr. Padgett for helping throughout the duration of my project and Dr. Yoder for help and for getting me resources. I would not have finished without both of them. Also thanks to Dr. Broughton, Dr. Finn, Dr. Song, and Dr. Leader whom I also consulted on this project. And finally, thanks to Mr. Mark Crosby, Mr. Gary Burgess, and Mr. Ben Webster for getting me equipment.

Chapter 1

Introduction

In hands-free telephony it is necessary to remove acoustic echo in the near end room from the speech acquisition system. This echo is created when the loudspeaker and microphone are placed in the same room – a necessary setup for teleconferencing. Echo removal is typically done using a method such as Normalized Least Mean Squares (NLMS) or Recursive Least Squares (RLS).

An enhancement to the speech acquisition system is to use an array of microphones. By spatially sampling the acoustic wavefront, it is possible to increase the Signal-to-Noise Ratio (SNR) and the Signal-to-Interference Ratio (SIR). This is necessary when the talker is far from the microphone because the power level falls off proportional to the square of the distance. In addition, the microphone array is directional and can help cancel the acoustic echoes that do not propagate along the same path as the near end talker's speech.

This thesis investigated Acoustic Echo Cancellation (AEC) in combination with a Delay-Sum Beamformer using a microphone array. The final implementation was done in real time on a Digital Signal Processor (DSP). For the thesis research, all discussions assumed there were two rooms. Both rooms had one talker, one loudspeaker, and a Uniform Linear Array (ULA).

Chapter 2

Acoustic Echo Cancellation

2.1 System Identification for AEC

AEC is a system identification problem. It is first necessary to identify the Room Transfer Function (RTF). Figure 2.1 depicts the problem. Assume the near end speech, s_n , is zero, and the noise, n_n , is zero. The subscript n denotes the current sample. Identification of the filter requires solving a system of equations using the reference signal, x_n , and the input signal (or desired signal), y_n , for each time step. LMS or RLS are often used to solve the problem of finding the adaptive filter. Neither solution is sufficient for this application. Further discussion is found in Section 2.2.3. The RTF is modeled using a linear filter. The length of the filter is chosen based on the reverberation time of the room and the desired level of Echo Return Loss Enhancement (ERLE). A reasonable reverberation time is between 0.1 and 0.3 seconds. This means the filter must have more than a thousand taps for a sample rate of at least 8 kH Nonlinear signals and echoes with a reverberation time greater than the filter length



Figure 2.1: Acoustic Echo Cancellation Problem

are not modeled and consequently are not removed. Once the RTF is found, it is used to filter the reference signal. The result is subtracted from y_n . This is the error, e_n . For AEC, the error is the signal of interest. The filter itself is not used except as part of the algorithm.

2.2 Adaptive Algorithms

There are many algorithms that can be applied to solve the AEC problem. Following is a description of algorithms that are often used. The Affine Projection Algorithms were specifically developed to deal with highly correlated signals such as speech.

2.2.1 RLS

One way to find the filter at each time step is by using the Least Squares method. Define the hermetian transpose, x^H , of a matrix, x, such that $x^H \doteq \overline{x^T}$. Superscript T designates the transpose such that the elements $x_{i,j}$ are swapped with $x_{j,i}$ where i and j are row and column indices, respectively. Minimize the error,

$$E_n \doteq \sum_{n=0} |y_n - \hat{\mathbf{h}}_n^H \mathbf{x}_n|^2 \tag{2.1}$$

This can be rewritten as the normal equations using the time averaged autocorrelation matrix and cross-correlation vector. Define

$$\mathbf{d}_n \stackrel{.}{=} \sum_{n=0} \mathbf{x}_n \overline{y_n}$$

 $\mathbf{R}_{\mathbf{xx},n} \stackrel{.}{=} \sum_{n=0} \mathbf{x}_n \mathbf{x}_n^H$

Then the linear Least Square Error (LSE) estimator can be written

$$E_n = \sum_{n=0} |y_n^2| - \hat{\mathbf{h}}_n^H \mathbf{d}_n - \mathbf{d}_n^H \hat{\mathbf{h}}_n + \hat{\mathbf{h}}_n^H \mathbf{R}_{\mathbf{xx},n} \hat{\mathbf{h}}_n$$

$$= y_n^2 - \mathbf{d}_n^H \mathbf{R}_{\mathbf{xx},n}^{-1} \mathbf{d}_n + (\mathbf{R}_{\mathbf{xx},n} \hat{\mathbf{h}}_n - \mathbf{d}_n)^H \mathbf{R}_{\mathbf{xx},n}^{-1} (\mathbf{R}_{\mathbf{xx},n} \hat{\mathbf{h}}_n - \mathbf{d}_n) \qquad (2.2)$$

And the minimum sum of squared errors is $y_n^2 - \mathbf{d}_n^H \hat{\mathbf{h}}_n$. For a positive definite matrix, Equation 2.2 is maximized under the following condition, known as the normal equations,

$$\mathbf{R}_{\mathbf{x}\mathbf{x},n}\mathbf{\hat{h}}_n = \mathbf{d}_n \tag{2.3}$$

It would be impractical to solve this equation for every new sample. Instead, the inverse of the autocorrelation matrix is updated recursively. The algorithm is known as the Conventional RLS. For a full explanation and the CRLS equations see the work of Manolakis [14]. CRLS is still complexity $O(n^2)$, which is not acceptable for a long filter. Although there are even faster versions, most are numerically unstable and difficult to implement.

2.2.2 NLMS

Another way to solve for the normal equations is to use an iterative method such as the Steepest Descent Algorithm (SDA). The SDA uses Equation 2.2 to solve the normal equations. To make the SDA adaptive in time, the autocorrelation matrix and cross-correlation vector are replaced with instantaneous time estimates and the iteration index is changed to be over time. The autocorrelation matrix is now rank one and a vector can be factored out so that the matrix is implicit. The result is the LMS. An improvement upon LMS is to normalize the step by dividing by the power of the input signal. This makes the step size independent of the power, and the result is known as Normalized LMS.

Let $\hat{\mathbf{h}}_n$ be the adaptive filter of length n, μ be the step size, and \mathbf{x}_n be the previous n samples of x_n . The NLMS equations are

$$\hat{y}_n = \hat{\mathbf{h}}_n^T \mathbf{x}_n \tag{2.4}$$

$$e_n = y_n - \hat{y}_n \tag{2.5}$$

$$\hat{\mathbf{h}}_{n+1} = \hat{\mathbf{h}}_n + \frac{\mu e_n \mathbf{x}_n}{\mathbf{x}_n^T \mathbf{x}_n}$$
(2.6)

Equation 2.4, does the filtering. It estimates the next input sample, y_n , using the adaptive filter and the reference signal. Equation 2.5 finds the error between the estimated input signal and the actual input signal. The step size will be proportional to the error. Equation 2.6 updates the adaptive filter by taking a step based on the size of the error and the gradient.

2.2.3 Affine Projections

System Identification is an iterative process that uses the error at each time step to converge. Growing memory RLS will converge to the Minimum Mean Square Error (MMSE), which is the same as the minimum LSE for an ergodic source. For selfcorrelated signals (which include speech), NLMS will converge slowly and not to the MMSE as seen in Manolakis [14]. There is some tradeoff between the rate of convergence and the steady state MSE, which can be controlled by adjusting the step size. NLMS also has a slow tracking speed due to its slow convergence. The convergence and tracking speed of NLMS are not sufficient for AEC according to Benesty [3]. In addition, RLS is not feasible due to the computational complexity resulting from the long echo path.

Another algorithm, known as the Affine Projection Algorithm (APA) by Ozeki [15], is a generalization of NLMS. In NLMS, the adaptive filter is modified by a step in the direction determined only by \mathbf{x}_n . APA overcomes this limitation by looking at previous vectors and correcting the direction of the step. It has faster convergence for correlated signals and a lower complexity than RLS. APA requires a projection

parameter, p, which should be between 1 and 50 for speech (p=1 is NLMS). Let $\mathbf{X}_{\mathbf{n}} \doteq [\mathbf{x}_n, \dots, \mathbf{x}_{n-(P-1)}]$. The APA algorithm as defined by Gay [11] is

$$\hat{\mathbf{y}}_n = \mathbf{X}_n^T \hat{\mathbf{h}}_n \tag{2.7}$$

$$\mathbf{e}_n = \mathbf{y}_n - \mathbf{\hat{y}}_n \tag{2.8}$$

$$\varepsilon = (\mathbf{X}_n^T \mathbf{X}_n + \delta \mathbf{I})^{-1} \mathbf{e}_n$$
(2.9)

$$\hat{\mathbf{h}}_{n+1} = \hat{\mathbf{h}}_n + \mu \mathbf{X}_n \varepsilon \tag{2.10}$$

The δ parameter is known as regularization and is used to decrease the condition number of the autocorrelation matrix so that the algorithm converges. The cost of regularization is a bias in the solution. A real symmetric positive definite matrix has positive eigenvalues as seen in Weisstein [20]. Let **A** be positive definite and λ_k be the eigenvalues of $\mathbf{B} = \mathbf{A} + \delta \mathbf{I}$. Using the definition of eigenvalues

$$\mathbf{B}\mathbf{x} = \lambda_k \mathbf{x}$$
$$(\mathbf{A} - (\lambda_k - \delta)\mathbf{I})\mathbf{x} = 0$$

The $\lambda_k - \delta$ are the eigenvalues of **A**. So, $\lambda_k > \delta$ and the new condition number is

$$\frac{\max\{\lambda_k\} + \delta}{\min\{\lambda_k\} + \delta} < \frac{\max\{\lambda_k\}}{\min\{\lambda_k\}}$$
(2.11)

This parameter needs to be chosen carefully. If it is too large, the filter will not converge to the correct answer. If it is too small, the filter will diverge.

2.2.4 Fast Affine Projections

A faster version of APA, known as the Fast Affine Projection Algorithm (FAP) by Gay [11], reduces the complexity enough to be feasible for AEC. FAP is more computationally intensive than NLMS, but it is also comparable to RLS in convergence speed at even a low order. To achieve the reduction in complexity, FAP finds only an approximation of \mathbf{e}_n . Through eigenvalue analysis it is shown that

$$\mathbf{e}_n \approx \begin{pmatrix} e_n \\ (1-\mu)\mathbf{e}_{n-1} \end{pmatrix}$$
(2.12)

Another complexity reduction is to update an approximation to the adaptive filter instead of the true adaptive filter. Because the a priori error, e_n , is calculated from $\hat{\mathbf{h}}_n$, an intermediate calculation of the approximate a priori error must be found before the true error is found. Finally, the update of the inverse of the autocorrelation matrix is done using the Sliding Window Fast Transversal Filter (SWFTF) by Cioffi [8]. It works by finding the normalized residual echo, ε , through updating forward and backward linear predictor vectors and prediction error energies. The savings occur because the vectors are the length of the projection order instead of the length of the filter as would be true in the usual case. Unfortunately, the FTF algorithm is unstable, even in floating point, and can be difficult to implement.

In Ding [9], an additional assumption is made that $\mu \approx 1$. This means the right side of Equation 2.12 simplifies to a scalar. This causes Equation 2.9 to be simplified so that only the left most column of the inverse of the autocorrelation matrix is needed. One iteration of the conjugate gradient method is used for each sample period to update the left column. Thus the algorithm is called the Conjugate Gradient Fast Affine Projection algorithm (CGFAP). A small modification is to use the lower complexity Gauss-Seidel iterations instead. This thesis used GSFAP by Albu [2] shown in Figure 2.2. It requires less computation than the other algorithms while still maintaining stability. Equation 2 in the figure is two rank one updates of the autocorrelation matrix. Equation 3 is the update of P – the left row of the inverse of the autocorrelation matrix. Equation 4 is the update of the approximation filter. Equation 5 is the update of the a priori error. The first two terms are the approximation error. The next term is the correction to get the true output error. Equation 6 updates the residual echo. Equation 7 updates the sum of residual echo vector.

Figure 2.3 compares the misalignment of Gauss-Seidel and Fast Transversal Filter FAP and NLMS. Sample refers to the sample number at 8 kH. Misalignment is a measure how close the adaptive filter is to the true filter and is defined as

$$\frac{\parallel \mathbf{\hat{h}}_n - \mathbf{h}_n \parallel}{\mathbf{h}_n} \tag{2.13}$$

The Gauss-Seidel filter converged by at least 12 dB more than FTFFap and NLMS. Figure 2.4 compares the ERLE for these same algorithms. ERLE is a measure of how much of the echo power has been removed and is defined as the ratio of the input power to the output power

$$10\log_{10}\frac{y_n^2}{e_n^2} \tag{2.14}$$

Again, the Gauss-Seidel version has around 10 dB more ERLE. The most likely reason for the FTFFap's poor performance is that a much larger regularization value than the one used for GSFap was needed for it to converge. Initialization

$$V(-1) = 0, \eta(-1) = 0, s(-1) = 0,$$

$$\mathbf{R}(-1) = \delta \mathbf{I}, \alpha = 1, P(-1) = b/\delta$$
(1)

Processing in sampling interval n

$$\mathbf{R}(n) = \mathbf{R}(n-1) + \xi(n)\xi^{T}(n) - \xi(n-L)\xi^{T}(n-L)$$
(2)

Solve $\mathbf{R}(n)\mathbf{P}(n) = \mathbf{b}$ using one GS iteration (3)

$$V(n) = V(n-1) + \alpha \eta_{N-1}(N-1)X(n-N)$$
(4)

$$e(n) = d(n) - V^T(n)X(n) - \alpha \eta^T(n-1)\tilde{R}(n)$$
(5)

$$\varepsilon(n) = e(n)P(n) \tag{6}$$

$$\eta(n) = \begin{pmatrix} 0 \\ \eta(n-1) \end{pmatrix} + \varepsilon(n)$$
(7)

Figure 2.2: Gauss-Seidel Fast Affine Projection Algorithm. (Reproduced with the permission of IEEE, Albu [2]).



Figure 2.3: Misalignment comparison of NLMS, FAP with FTF, FAP with GS. (This is only an approximation for FAP. The true adaptive filter is never found.)



Figure 2.4: ERLE comparison of NLMS, FAP with FTF, FAP with GS.

2.2.5 Block Exact Fast Affine Projections

Even NLMS is computationally expensive for a long echo path. It is desirable to reduce the computation further. The savings may be necessary if there are many echo cancellers. One way to do this is to use a block adaptive filter. A block algorithm only adapts the filter every L samples, where L is the block length. When the filter is not changing, convolution implemented using the Fast Fourier Transform (FFT) can be used to decrease the amount of computation as noted by Shynk [17]. Block adaptive filters achieve the minimum computation when the FFT is the same length as the filter. However, for a long filter, this would introduce too much delay.

Although block algorithms are more efficient, the filter converges slower. A block exact algorithm corrects the errors, and the output is exactly equivalent to the original algorithm except for numerical error. The cost of using a block exact algorithm is delay and more memory storage. This thesis used the Block Exact Fast Affine Projection Algorithm (BEFAP) by Tanaka [18]. However, instead of using FTF to update the gain, Gauss-Seidel iterations were used for stability reasons. The algorithm is reproduced in Figure 2.5. Equation 1 of the figure is block filtering. This can be done using fast matrix operations or implementing fast convolution via the FFT. The next set of equations recursively updates the quantities needed to update the prefilter coefficients. Equation 3 finds the error and corrects it. Equation 4 updates the error vector. This is not used with the Gauss-Seidel version. Equation 5 updates the autocorrelation matrix and the inverse. Equation 6 updates the prefilter coefficients needed to correct the error and the filter. Equation 8 updates the approximation filter.

0. Initialization

$$k_{\text{current}} = N_2 - v$$
, $k = k_{\text{current}} - v$. v : delay to avoid data shortage
 $k' = k - N_1 + 1$,
 $r_j(k'-1) = \mathbf{x}^T(k'-1)\mathbf{x}(k'-1-i)$, $j = 0, 1, ..., N_2 + p - 2$
 $\rho_{i+p-1}(k'-jN_2+i) = \sum_{l=0}^{i} x(k'-jN_2+l)x(k'-jN_2-i-p+1+l)$,
 $i = 0, 1, ..., N_2 - 1$, $j = 1, 2, ..., L/N_2$
 $\mathbf{e}(k'-1) = [y(k'-1), y(k'-2), ..., y(k'-p)]^T$,
 $\mathbf{s} = \mathbf{0}$,
 $\mu(k'-1) = 0$

Figure 2.5: Block Exact Fast Affine Projection Algorithm. (Reproduced with the permission of IEEE, Tanaka [18].)

#	computation	computational complexity
For $i = 0, 1,$, $N_2 - 1$ do 1. to 7.	
1. If i mod	$N_1 = 0$ then $\mathbf{\hat{y}}'(k) = \mathbf{X}_{L,N_1}^T(k)\mathbf{z}(k')$.	(L/N_1) MVP $(N_1) + N_1 \log_2 N_1$
$2. \; r_j(k'+i)$	$= r_j(k'+i-1) + x(k'+i)x(k'+i-j),$	2(i+p-1)
-x(k'-L)	$(i) + i + i + i - j), j = 0, 1, \dots, i + p - 2$	
$p_{i+p-1}(k$	$(t+i) = \sum_{k=0}^{i} x(k'+i)x(k'-i-p+1+i)$	<i>i</i> + 1
$r_{l+p-1}(k$		2
e(k' + i) = 3.		<i>i</i> + <i>p</i>
$4. \begin{bmatrix} \mathbf{e}(k'+i) \\ * \end{bmatrix}$	$= \begin{bmatrix} e(k'+i) \\ (1-\mu(k'+i-1))\mathbf{e}(k'+i-1) \end{bmatrix}$	<i>p</i> – 1
5. Constitute	$\mathbf{R}(k'+i)$ from $r_j(k'+i-l)$, $(j, l = 0, 1,, p-i)$	1),
and solve	$\mathbf{R}(k'+i)\mathbf{g}(k'+i) = \mathbf{e}(k'+i).$	15p, if FTF is used.
6. $s_{i+\rho}(k' +$	$i) = \begin{bmatrix} 0\\ \mathbf{s}_{i+p-1}(k'+i-1) \end{bmatrix} + \mu(k'+i) \begin{bmatrix} \mathbf{g}(k'+i)\\ 0 \end{bmatrix}$	р
7. $k = k + 1$		
8. $z(k' + N_2)$	$= \mathbf{z}(k') + \mathbf{X}_{L,N_2}(k'-p+N_2)\mathbf{s}\big _{p+N_2-1}^{p}(k'+N_2-$	1) (L/N_2) MVP $(N_2) + N_2 \log_2 N_2$
9. $k' = k' + b$	N ₂	

Figure 2.5 (cont.)

2.2.6 Controlling Convergence

As mentioned in Section 2.2.3, the condition number of the autocorrelation matrix determines convergence. It is shown in Figures 2.6 and 2.7 how the regularization, used to control the condition number, and step size parameter effects convergence. The regularization parameter needs to be chosen based on the maximum size of the condition number. The size of the condition number depends on the power level. If there is a way to reduce the range of the power level, it might be possible to choose a smaller regularization parameter without causing divergence. One option is to try Automatic Gain Control (AGC) on the reference signal. Although it is usually a bad idea to modify the reference signal, AGC can be a desirable feature in telephony. An unfortunate side effect is the added delay by the amount of the block size.

Figure 2.8 shows the results of experiments applying gain control for better convergence at a given condition number. The total power was normalized to 1 dB. All parameters were identical. In this case, the filter nearly diverged right at the start for the non-AGC control case, but thereafter began to reconverge. However, even after ten seconds, the error was still 40 dB larger than the AGC case.

In Figure 2.9A only a small improvement of the condition number when using AGC is shown. However, as seen in Figure 2.9B the smallest singular values are much larger with AGC. According to Rombouts [16], the FAP algorithms amplify noise due to small singular values. This may partially explain the improvement in convergence.



Figure 2.6: (A) and (B) NLMS with a step size of 0.98 using a 1024 tap adaptive filter on composite source signals. (C) and (D) Using a step size of 0.15. (A) and (C) In the absence of noise. (B) and (D) With noise 21.45 dB down from the average power of the source.



Figure 2.6 (cont.)



Figure 2.7: (A) and (B) BEGSFAP with a step size of 0.98 using a 1024 tap adaptive filter on composite source signals. (C) and (D) Using a step size of 0.15. (A) and (C) In the absence of noise. (B) and (D) With noise 21.45 dB down from the average power of the source.



Figure 2.7 (cont.)



Figure 2.8: Output error with and without AGC.



Figure 2.9: (A) Effect of AGC on condition number. (B) On the smallest singular value.

In Figure 2.6, the regularization parameter of NLMS varied over two orders of magnitude, and the result was about 5 dB to 10 dB of convergence difference. In Figure 2.7, the regularization parameter of GSFAP varied about one order of magnitude, and this resulted in as much as 10 dB to 20 dB of convergence difference. The results of Figure 2.8 show that AGC can control convergence. And it is seen in Figure 2.9 how AGC effects the condition number and smallest singular values.

Chapter 3

Double Talk Detection

3.1 Echo Cancellation Scenarios

The assumption of Section 2.1 was that $s_n = 0$. If the near talker is speaking, $s_n \neq 0$ and the filter will diverge. In this case, adaptation of the filter must be disabled. Table 3.1 shows four cases that need to be addressed.

It is relatively easy to detect when the reference signal, x_n , is zero. In the other two

s_n	x_n	adapt
=0	=0	no
$\neq 0$	=0	no
=0	$\neq 0$	yes
$\neq 0$	$\neq 0$	no

Table 3.1: Adaptation Cases
cases a detection statistic is calculated and compared against a threshold to distinguish between $s_n \neq 0$ and $s_n = 0$.

3.2 Detection Methods

A common Double Talk Detection (DTD) scheme is the Geigel method. It is computationally cheap but not effective for AEC. The Normalized Cross-Correlation method has good performance for AEC but is expensive. The cheap version, however, has an excellent performance-to-computation ratio. The Receiver Operating Characteristic (ROC) in Figure 3.1 compares the performance of Geigel, CNCR, and the Variance Impulse Response (VIRE) algorithm by Åhgren [1], which is not discussed here. The goal of DTD is to minimize both the probability of false double talk and the probability of a double talk miss. Ideally, the ROC should lie on the axes, indicating perfect DTD. Comparatively, CNCR has done well.

3.2.1 Geigel

The Geigel method is

$$\xi_n \doteq \left| \frac{y_n}{\max\{|x_n|, \dots, |x_{n-m+1}|\}} \right| \tag{3.1}$$

For far end speech only, y_n should be smaller than the last several samples of x_n . During double talk, y_n should be about the same as x_n due to the power of the near end speech. Thus during double talk, ξ_n should be larger than for far end speech only. Unfortunately it is difficult to choose m for a long echo path. And the threshold depends on the acoustic echo path. Note, however, that this method is completely independent of the adaptive filter.



Figure 3.1: ROC

3.2.2 Cheap Normalized Cross-Correlation (CNCR)

The Normalized Cross-Correlation method by Benesty [4] uses a detection statistic that compares the estimated power of the input signal to the true power of the input signal. The statistic is

$$\xi_n \doteq \sqrt{\mathbf{d}_n^T (\sigma_{y,n}^2 \mathbf{R}_{\mathbf{x}\mathbf{x},n})^{-1} \mathbf{d}_n}$$
(3.2)

Using Equation 2.3 and

$$\sigma_{y,n}^2 = \mathbf{d}_n^T \mathbf{R}_{\mathbf{x}\mathbf{x},n}^{-1} \mathbf{d}_n \tag{3.3}$$

The statistic is rewritten as

$$\xi_n = \sqrt{\frac{\mathbf{h}_n^T \mathbf{R}_{\mathbf{xx},n} \mathbf{h}_n}{\mathbf{h}_n^T \mathbf{R}_{\mathbf{xx},n} \mathbf{h}_n + \sigma_{s,n}^2}}$$
(3.4)

When the near end speech power, $\sigma_{s,n}^2$, is zero, $\xi_n = 1$. Otherwise, $\xi_n < 1$.

The cheap version replaces the RTF, \mathbf{h}_n , with the model of the RTF, $\mathbf{\hat{h}}_n$, and simplifies the numerator using Equation 2.3 resulting in

$$\xi_n = \sqrt{\frac{\mathbf{d}_n^T \hat{\mathbf{h}}_n}{\hat{\sigma}_{y,n}^2}} \tag{3.5}$$

One last simplification shows that the numerator is just the cross-correlation between the input signal and the filtered approximation of the input signal \dot{A} hgren [1]. Then the detection statistic is

$$\hat{\sigma}_{y,n}^{2} = \mathbf{d}_{n}^{T} \hat{\mathbf{h}}_{n}$$

$$= \hat{\sigma}_{y,n-1}^{2} + y_{n}^{2} - y_{n-N}^{2}$$
(3.6)

$$\tilde{\sigma}_{y,n}^2 = \tilde{\sigma}_{y,n-1}^2 + y_n \hat{y}_n - y_{n-N} \hat{y}_{n-N}$$
(3.7)

$$\xi_n = \sqrt{\frac{\tilde{\sigma}_{y,n}^2}{\hat{\sigma}_{y,n}^2}} \tag{3.8}$$

If the echo canceller has not fully converged, the detection statistic for CNCR will be invalid, and so it is difficult to choose a meaningful threshold. The problem is the worst during double talk when the detection statistic is most needed. Instead of using a threshold for this thesis, the detection statistic was interpreted as the probability of single talk. The rate of adaptation was made proportional to the detection statistic

$$\mu \doteq \frac{\xi - T}{1 - T} \tag{3.9}$$

where T was a parameter to control how ξ is mapped onto μ . If $\xi < T$, μ was set to zero to prevent the relaxation from becoming negative. Figure 3.2 compares this method to using various thresholds. The results of the second figure from the top show the adaptation step size for three different thresholds and the method proposed here. Notice that between 30000 and 35000 samples, when the simulation switched from double talk to single talk and back to double talk, the proposed method was correctly adapting, while the other methods were not.

3.2.3 Complexity Comparison

From the ROC it is clear that CNCR is a desirable detection statistic. Table 3.2 shows that it has an excellent computational complexity.

Table 3.2: The computational complexity calculation assumes one operation for add, subtract, multiply, absolute value, and 10 operations for divide.

Detection Statistic	Complexity
CNCR	18
VIRE	n+9
Geigel	2m+11



Figure 3.2: Comparison of Thresholding versus Proportionality

Chapter 4

Delay-Sum Beamforming

4.1 Electronic Beamforming

It is possible to increase the quality of a signal by using an array of microphones. The signal power is increased by sampling the acoustic waveform in multiple places. This also makes it possible to steer the array from within software or electronically.

Assume the source is far from the array and that the wavefront is planar. A simple way to steer the array is to align the signals in time and sum, thereby increasing signal power in one direction while decreasing noise and interference in all other directions. In the frequency domain, the beamformer is

$$S(\omega) \doteq \sum_{k=1}^{K} L_k S_k(\omega) e^{-j\omega d_k}$$

$$(4.1)$$

$$S_k(\omega) \doteq s(\omega)e^{-j\omega\tau_k}$$
 (4.2)

where L_k is the path loss of the signal, $s_k(\omega)$ is the signal received at the kth microphone, τ_k is the time for the wave to propagate from the source to microphone k, and d_k is the insertion delay used to shift and align the signals. Ideally d_k are chosen so that

$$\tau_k + d_k = \tau_j + d_j = C, \forall k, j \tag{4.3}$$

The result is a spatially matched filter. Under the condition of Equation 4.3, the array output of the matched filter is

$$S(\omega) = s(\omega)e^{-j\omega C} \sum_{k=1}^{K} L_k$$
(4.4)

If the L_k are constant, the beamforming gain is K according to Manolakis [14]. The theoretical array response for two situations is shown in Figure 4.1. Refer to Yu-Kang [13] for an in depth analysis of a delay-sum beamformer.

4.1.1 Inter-Element Spacing

The microphones were spaced one half the minimum wavelength apart in a linear array. If the spacing were greater, the array would have spatial aliasing problems. If the spacing had been less, the aperture of the array would be smaller than it needed to be for optimum performance as seen in Manolakis [14].

4.1.2 Wide Beam Response

A perfectly aimed array will pass all frequencies. However, if any delays are off, it acts as a low pass filter. A few attempts at constructing an inverse filter did not yield useful results. Correcting the wide beam response is left for further research. Figure 4.2 shows the frequency response of the array summed overall angles.



Figure 4.1: (A) Theoretical array response aimed at 0 degrees. (B) Aimed at 45 degrees.

4.2 Time-Delay Estimation (TDE)

Aligning the time signals required finding the time delay between consecutive microphones. Finding the TDE was done using the cross-correlation. Since the delay needed to be accurate to a fraction of a sample, phase information was necessary to find the subsample delay. Thus the delays were found using the slope of the cross-power spectral phase. This was another reason microphones were spaced half a wavelength apart of the maximum frequency. The maximum frequency travels only half a cycle



Figure 4.2: Low Pass Filtering Effect of Array.

between microphones. This means the angle is limited to the range $[-\pi, \pi]$ and the phase did not wrap, keeping the slope estimation simple.

Let c be the speed of sound and r_{m_k} be the distance from the source to microphone k. As in Yu-Kang [13], define the signal at the k_{th} microphone and its spectrum to be

$$s_{m_k}(t) \doteq L_k s(t - \frac{r_{m_k}}{c}) \tag{4.5}$$

$$S_{m_k}(\omega) \doteq L_k S(\omega) e^{-j\omega \frac{r_{m_k}}{c}}$$
 (4.6)

In the absence of noise and reverberation, the cross-power spectrum between any two consecutive microphones is

$$X_{m_k,m_{k+1}}(\omega) = S_{m_k}(\omega)\overline{S_{m_{k+1}}(\omega)}$$
(4.7)

$$= L_k L_{k+1} |S(\omega)|^2 e^{-j\omega \frac{r_{m_k} - r_{m_{k+1}}}{c}}$$
(4.8)

$$= L_k L_{k+1} |S(\omega)|^2 e^{-j\omega\tau_{m_k,m_{k+1}}}$$
(4.9)

The phase, $\tau_{m_k,m_{k+1}}$, is also the time delay, and the sample delay is $\Delta = \tau_{m_k,m_{k+1}} \frac{f_s}{2\pi}$, where f_s is the sample frequency.

To keep the amount of computation low, only the time delays were found. The source location was implicit as the goal was to increase SNR. The total delay for each microphone is the sum of all previous delays. One microphone must be chosen as the starting point, and a constant offset must be added to each delay to remove any negative delays.

The cross-spectrum phase under various noise conditions is shown in the simulations of Figure 4.3. It does not take a significant amount of noise to corrupt the results when a pure tone is used. However, using a square wave appears to have better results. It is also interesting to note that the cross-spectrum appears to have more phase corruption at frequencies distant from the pure tone used.



Figure 4.3: Cross-Spectrum Phase Plots. (A) Noise is 56.2 dB below signal power (B) Noise is 76.3 dB below signal power (C) Square wave



С

Figure 4.3 (cont.)

4.2.1 Least Squares Estimation

After finding the cross-power spectral phase, the slope of the line needs to be estimated. Noise and reverberation will corrupt the slope of the line. To improve the estimate of the fit, estimates of the SNR were used at each frequency as weighting factors as seen in Yu-Kang [13] and Chan [6].

4.3 Delay Filters

Satisfying the condition of Equation 4.3 requires noninteger delays. One way to do this is filtering with a sinc, but this has significant amplitude error compared to other methods as shown in Cain [7]. The polynomial interpolation method of Farrow [10] has little error over more than half the frequency bandwidth as seen in Figure 4.4. The error between the ideal delay filter and the fractional delay filter is shown for all frequencies and delays between -0.5 and 0.5 samples. Because most of the power in speech is in the first 2 kH, error near half the sampling frequency can simply be ignored. Furthermore, large errors in the estimated delays themselves makes it unnecessary for the filters to have a near perfect delay response.

Each filter tap is a polynomial function of the fractional portion of the delay. The polynomials are found by minimizing the MSE between the desired filter and the ideal delay filter, $h_{id} = e^{j\omega\Delta}$, over a given frequency range. Polynomial coefficients allow the delay to be variable at every sample, and calculating a new filter is still computationally cheap. For a delay-sum beamformer, the delays were only changed once per frame. The Maple code is provided in Appendix A for finding the polynomials.



Figure 4.4: Filter Magnitude Error

Chapter 5

AEC with Beamforming

There are two ways to combine echo cancellation with beamforming. One method is to place a single echo canceller after the beamformer. This greatly reduces the computation required. However, the beamformer will become a part of the echo path. The long acoustic filter of the slow converging NLMS cannot adapt quickly enough to the short adaptive filter of the beamformer according to the work of Benesty [3]. The simpler method is to place an echo canceller after every microphone and make the beamformer operate on the output of the echo cancellers as in Figure 5.1. Obviously the amount of computation will depend heavily on the number of microphones.

This thesis studied only the later method due to time constraints. It was a straightforward matter to combine AEC with beamforming. One advantage of using this method in combination with the Block Exact GSFAP was that the solution to the affine projection equations needed only to be calculated once per sample for all microphones. This occurred because the equations depended only on the reference signal, which was common to all input channels. Computation was further reduced by updating the equations less than once per sample.



Figure 5.1: Structure of AEC with Beamforming

5.1 Using Poles

Most adaptive echo cancellation techniques are zero-only models. The room transfer function zeros depend on the size, shape, and materials of the room as well as the source, receiver and object positions in the room. The poles correspond to room resonances that are independent of the location of objects within the room. The method of Haneda [12] estimates the theoretical room poles using a Common-Acoustical-Pole and Zero Model. The estimation uses a number of different source and receiver positions because poles are expressed at all source and receiver positions.

It is impractical to model all or even most of the poles in room. The theoretical number of poles in a room of volume V is given by Haneda [12] as

$$poles = \frac{\pi}{3}V(\frac{f_s}{c})^3 \tag{5.1}$$

For even low sample rates, such as those used in telephony, and given a small room, the number of poles is too large. So the number of poles estimated is much less than the theoretical number in the room. The calculation done this way yields poles corresponding to the largest resonances in the room.

Once the poles are known, they can be placed before, as shown in Figure 5.2, or after the adaptive portion of the filter. By using a pole and zero model, the order of the echo cancellers can be reduced while maintaining a given level of ERLE. Thus, it would be possible to increase the aperture size of the array at a given amount of computation if memory and physical constraints allow.

The comparison in Figure 5.3 was generated by using fifteen simulated RTFs, as they would appear on a ULA, to generate the poles. The sixteenth input was filtered against the poles and run through NLMS. It is seen that the ERLE was better by about 5 dB with poles. The poles version used a filter with 1024 coefficients and another 500 coefficients for the poles. The other filter was length 2048. Additional research is necessary to determine how well this works in a real environment with estimated RTFs. Finding the poles required solving a matrix equation on the order of the number of poles. A real-time implementation might require starting the array with only adaptive echo cancellers and introducing poles at some later time.



Figure 5.2: Structure of AEC with Poles.



Figure 5.3: AEC with and without Poles.

Chapter 6

Real-Time Implementation

6.1 Setup

The microphone array consisted of seven microphones in a nearly ULA arrangement and is shown in Figure 6.1. Each input was sampled at 8 kHz and the DC removed. Although there was a time delay of $\frac{1}{16}$ sample between sampling of consecutive microphone signals, time-delay estimation and delay filters corrected this problem.

Each of the seven signals were run through a 1024 coefficient adaptive filter of projection order 16. Each signal was stored in a frame buffer of length 256. This was also the block size for convenience. The reference signal was sampled, run through AGC, and placed in the output buffer. The previous reference signal frame was used for echo cancellation. The modified CNCR method proposed in this thesis was applied to the left most microphone with an experimentally determined scaling parameter to find the relaxation parameter for AEC. The same relaxation parameter was used for all microphones. One Gauss-Seidel iteration was done each cycle and the solution was used for all seven signals. Block filtering was done using optimized FFT routines before and after the error correction and step direction adjustment phases. To reduce computation further, the imaginary portion of the FFT was used to calculate two simultaneous FFTs (Appendix B).

Following completion of the echo removal, the beamformer processed the resultant signals. The implementation was nearly the same as in Yu-Kang [13], but with temporal averaging and group variance thresholding disabled for experiments. The seven microphones were spatially Hamming windowed. Each signal was temporarily Hanning windowed to preserve phase before taking the 256 point FFT. Time delay estimation used the generalized least squares function on the positive frequencies with the last twenty samples left off to avoid possible phase wrapping due to noise. The time delays were sub-array averaged. The delays were shifted so that the left most microphone had no delay and the right most microphone was the sum of the previous delays. Delays greater than one sample were thrown out. A fourth degree, twelfth order Farrow filter was used for fractional delay. The order and degree were chosen arbitrarily. The last step was to sum and place the result in the output buffer.



Figure 6.1: Array Setup.

6.2 Experiments

An array gain was generated in real-time by passing measurements over Real Time Data Exchange (RTDX) to Matlab. The results are shown in Figure 6.2. Parts A through D were with the array steered at 0 degrees, 1.0 m from the array. Parts E through H were with the array steered at 45 degrees, 0.65 m from the array. Parts I through L were with the array steered at 90 degrees, 0.65 m from the array. In all cases, the largest signal gain was at 1500 Hz. The square wave also had large signal gain in most cases.

Due to a number of problems with electrical and acoustic noise and interference and the clock rate not being exactly at 8000 Hz, the following procedure was adopted.

- (1) All input was low pass filtered at 350 Hz.
- (2) The time domain signal was Hamming windowed.
- (3) All frequency bins with harmonics of 500 Hz excluding the frequency of interest were discarded.
- (4) The frequency bins before and after those same harmonics were discarded.
- (5) For the square wave, only the first harmonic was used.
- (6) The signal power in the bin of interest from the middle microphone was calculated.
- (7) The bins before and after were also included in the calculation.
- (8) The noise power between 2000 and 3000 Hz was summed and multiplied by four.

- (9) This was done every fourth frame, and the results were sent to Matlab.
- (10) The array gain, signal gain, and noise gain were run through a 20 point running average filter.
- (11) The output was observed until the array gain appeared to be near the maximum.
- (12) The last two measurements were skipped because halting the program caused noise that was effecting the results.
- (13) The author then counted back 20 measurements and chose the one with the largest array gain. This was necessary to estimate the peak gain due to the wildly fluctuating real-time measurement results.

When the noise gain was above 4 dB, the cause was likely due to interference or measurement error due to FFT leakage. The array gain in this situation was not reliable. Also, note that the sampled signal from the last microphone was observed to be distorted and of less power than the others. The magnitude was corrected in software. The results in Figure 4.3 suggest that the time delay should be most accurate at 500 - 1500 Hz. However, at higher frequencies, the TDE needs to be more accurate to maintain a given level of array gain. So, a more accurate TDE does not imply higher gain.



Figure 6.2: Array Gain with Signal and Noise Components versus SNR at the Center Microphone.



Figure 6.2 (cont.)



Figure 6.2 (cont.)



Figure 6.2 (cont.)



Figure 6.2 (cont.)



Figure 6.2 (cont.)

The results of Figure 6.3 show a measurement of ERLE with and without beamforming. The speech segment 'ah' was played repeatedly for this experiment. ERLE was averaged over each frame and the results were sent to Matlab. With beamforming enabled, the array was aimed at 0 degrees and the speaker placed at 90 degrees. The output was divided by the theoretical magnitude increase of the array to normalize it back to the size of the input. Because the array was aimed away from the echo source, the echo acted as an interference source and was suppressed. This increased the ERLE.



В

Figure 6.3: (A) Real-time ERLE of 'ah', (B) with Beamforming.

6.3 Conclusion and Further Work

By qualitative and quantitative analysis, this work has shown that a practical realtime beamformer can enhance signal quality. A real-time acoustic echo canceller can improve signal quality. In addition, the two can work together and enhance one another. With a faster DSP or even an FPGA implementation, a number of improvements can be made without the need for higher power algorithms. The sample rate can be increased. The number of microphones can be increased. The echo path length and projection order can be increased. Higher quality electronics can be used.

However, the best improvements will come from better analysis and algorithms. The delay-sum beamformer is the simplest type of beamforming. An optimum beamformer could replace it. Then nulls could be placed over interference sources such as the loudspeaker, further reducing echo. Better time delay estimation techniques, such as the state space type algorithms of Ward [19], will certainly improve performance. Higher power acoustic echo cancellation is an active research area. The technology to implement these algorithms has only become available within the past several years.

Last, a strategy for combining algorithms was presented. This work mostly dealt with two very independent methods of enhancing signal quality: beamforming and echo cancellation. However, the work of Kellermann [3] shows that the two can be combined in such a way as to remove all but a single echo canceller. In addition, other algorithms used were double talk detection and pole calculation. The echo cancellation algorithm was built on the work of several previous echo cancellers. The block exact version even offers the choice of what type of fast calculation and matrix solver to use. The version implemented used the FFT for fast calculations and Gauss-Seidel iterations for a matrix solver. The suggested implementation uses a fast matrix multiplication construct new ones. By understanding how each step of these algorithms operate, it is possible to customize and combine them to solve a particular problem.

Bibliography

General References

Allen, Jont B., and David A. Berkely. "Image Method for Efficiently Simulating Small-Room Acoustics." *Journal of the Acoustical Society of America* 65 (1979): 943-951.

Stéphenne, Alex, and Benoît Champagne. "Cepstral Prefiltering for Time Delay Estimation in Reverberant Environments." *ICASSP* (1995): 3005-3058.

Cited References

- Ahgren, Per. "On System Identification and Acoustic Echo Cancellation." PhD Thesis. Uppsala Universitet, April 2004.
- [2] Albu, Felix, Jiri Kadlec, Nick Coleman, and Anthony Fagan. "The Gauss-Seidel Fast Affine Projection Algorithm." *IEEE Workshop on Signal Processing Systems* (2002): 109-114.
- [3] Benesty, Jacob, Tomas G\u00e4nsler, Dennis R. Morgan, and M.Mohan Sondhi, Steven L. Gay. Advances in Network and Acoustic Echo Cancellation. New York: Springer-Verlag, 2001.
- [4] Benesty, Jacob, D.R. Morgan, and Jun H. Cho. "A New Class of Doubletalk Detectors Based on Cross-Correlation." *IEEE Transactions on Speech and Audio Processing* 8 (2000): 168-172.
- [5] Brandstein, Michael, and Darren Ward. *Microphone Arrays*. New York: Springer-Verlag, 2001.
- [6] Chan, T. T., Richard V. Hattin, and J. B. Plant. "The Least Squares Estimation of Time Delay and Its use in Signal Detection." *IEEE Transactions on Acoustics*, *Speech, and Signal Processing* 26 (1978): 217-222.
- [7] Cain, G. D., N. P. Murphy, and A. Tarczynski. "Evaluation of Several Variable FIR Fractional-Sample Delay Filters." *ICASSP* (1994): 621-624.
- [8] Cioffi, John M. "Fast Transversal Filters for Communications Applications." PhD Thesis, Stanford University 1984.
- [9] Ding, Heping. "A stable fast affine projection adaptation algorithm suitable for low-cost processors." *ICASSP* (June 2000): 360-363.
- [10] Farrow, C. W. "A Continuously Variable Digital Delay Element." Proceedings of the 1988 IEEE International Symposium on Circuits and Systems 3 (1998): 2641-2645
- [11] Gay, Steven L., and Sanjeev Tavathia. "The Fast Affine Projection Algorithm." ICASSP (1995): 3023-3026.

- [12] Haneda, Yoichi, Shoji Makino, and Yutaka Kaneda. "Common Acoustical Pole and Zero Modeling of Room Transfer Functions." *IEEE Transactions on Speech* and Audio Processing 2 (1994): 320-328.
- [13] Ku, Yu-Kang. "Real-Time DSP Implementation of a Delay-Sum Beamformer." Masters Thesis, Rose-Hulman Institute of Technology, May 1999.
- [14] D.G. Manolakis, K.I. Vinay, and S.M. Kogon, Statistical and Adaptive Signal Processing. Boston: McGraw-Hill Companies, Inc, 2000.
- [15] Ozeki, Kazuhiko, and Tetsuo Umeda. "An Adaptive Filtering algorithm Using and Orthogonal Projection to an Affine Subspace and Its Properties." *Electronics and Communications in Japan* 67-A (May 1984): 19-27.
- [16] Rombouts, Geert, and Marc Moonen. "A Sparse Block Exact Affine Projection Algorithm." *IEEE Transactions on Speech and Audio Processing* 10 (2002): 100-108.
- [17] Shynk, John J. "Frequency-Domain and Multirate Adaptive Filtering." IEEE Signal Processing Magazine 9 (1992): 14-37.
- [18] Tanaka, Masashi, Shoji Makino, and Junji Kojima. "A Block Exact Fast Affine Projection Algorithm." *IEEE Transactions on Speech and Audio Processing* 7 (1999): 79-86.
- [19] Ward, Darren B., Eric A. Lehmann, and Robert C. Williamson. "Particle Filtering Algorithms for Tracking an Acoustic Source in a Reverberant Environment." *IEEE Transactions on Speech and Audio Processing* 11 (2003): 8265-836.

[20] Weisstein, Eric W. "Positive Definite Matrix." MathWorld-A Wolfram Web Resource. http://mathworld.wolfram.com/PositiveDefiniteMatrix.html. November 2005.

Appendix A

Farrow Filters Maple Code

- > #Design a Farrow filter
- > restart:
- > N:=11: #Number of taps (minus 1)
- > Pord:=4:
- > MaxUseFreq:=.74: #upper useful frequency
- > MinUseFreq:=.5: #lower useful frequency
- > #Set up the summation in the inside of the MSE integral
- > #Then force maple to use sines and cosines and expand the expression
- > inside1:=omega->sum(sum('alpha^m*c[n,m]*exp(I*n*omega)',m=0..Pord),n=0
- > ..N)-exp(I*omega*(alpha+N/2)):
- > inside2:=expand(evalc(inside1(omega)*inside1(-omega))):

- > #This is the magnitude squared
- > #Integrate over alpha
- > inside3:=int(inside2,alpha=-0.5..0.5):
- > #Pass the derivative through the omega integral
- > for n from 0 to N do
- > for m from 0 to Pord do
- > dfinside[n,m]:=diff(inside3,c[n,m]);
- > df0[n,m]:=int(dfinside[n,m],omega);
- > #Maple won't evaluate this integral correctly; so, it's done this way
- > df[n,m]:=evalf(subs(omega=MaxUseFreq*Pi,df0[n,m])-subs(omega=-MaxUseFr
- > eq*Pi,df0[n,m]));
- > #df[n,m]:=evalf(subs(omega=Pi,df0[n,m])-subs(omega=MinUseFreq*Pi,df0[n
- > ,m]))+evalf(subs(omega=-MinUseFreq*Pi,df0[n,m])-subs(omega=-Pi,df0[n,m
- >]));
- > od:
- > od:

```
> #Generate the additional linear constraints
> for n from 0 to N do
> G[n]:=c[n,0]+.5*c[n,1]+.25*c[n,2]+.125*c[n,3]+.0625*c[n,4]=0;
> G[n+(N+1)]:=c[n,0]-.5*c[n,1]+.25*c[n,2]-.125*c[n,3]+.0625*c[n,4]=0;
> od:
> pos:=6:neg:=5:Gall:=0:#OR IS THIS pos:=3:neg:=4?
  G[pos]:=c[pos,0]+.5*c[pos,1]+.25*c[pos,2]+.125*c[pos,3]+.0625*c[pos,4]
>
> =1:
> G[neg+(N+1)]:=c[neg,0]-.5*c[neg,1]+.25*c[neg,2]-.125*c[neg,3]+.0625*c[
> pos,4]=1:
> #Construct the lambda portion of the Lagrangian
> for k from 0 to (2*N+1) do
> Gall:=Gall+lambda[k]*lhs(G[k]):
> od:
> #Take derivatives with respect to the Lagrangian
> for n from 0 to N do
> for m from 0 to Pord do
> dg[n,m]:=diff(Gall,c[n,m]);
> dL[n*(Pord+1)+m]:=df[n,m]+dg[n,m]=0;
> od;
  od;
>
> for k from (N+1)*(Pord+1) to (N+1)*(Pord+1)+2*N+1 do
> dL[k]:=G[k-(N+1)*(Pord+1)]:
```

```
> od:
```

```
> soln:=solve(
```

- > {dL[0],dL[1],dL[2],dL[3],dL[4],dL[5],dL[6],dL[7],dL[8],dL[9],dL[10],d
- > L[11],dL[12],dL[13],dL[14],dL[15],dL[16],dL[17],dL[18],dL[19],dL[
- > 20],dL[21],dL[22],dL[23],dL[24],dL[25],dL[26],dL[27],dL[28],dL[29],dL[
- > 30],dL[31],dL[32],dL[33],dL[34],dL[35],dL[36],dL[37],dL[38],dL[39],dL[
- > 40],dL[41],dL[40],dL[42],dL[43],dL[44],dL[45],dL[46],dL[47],dL[48],dL[
- > 49],dL[50],dL[51],dL[52]
- > ,dL[53],dL[54],dL[55],dL[56],dL[57],dL[58],dL[59],dL[60],dL[61],dL[62]
- > ,dL[63],dL[64],dL[65],dL[66],dL[67],dL[68],dL[69],dL[70],dL[71],dL[72]
- > ,dL[73],dL[74],dL[75],dL[76],dL[77],dL[78],dL[79],dL[80],dL[81],dL[82]
- > ,dL[83],dL[84],dL[85]}):
- > assign(soln):
- > printf("vn0=[%f %f %f
- > %f];",c[0,0],c[1,0],c[2,0],c[3,0],c[4,0],c[5,0],c[6,0],c[7,0],c[8,0],
- > c[9,0],c[10,
- > 0],c[11,0]);
- > printf("vn1=[%f %f %f
- > %f];",c[0,1],c[1,1],c[2,1],c[3,1],c[4,1],c[5,1],c[6,1],c[7,1],c[8,1],
- > c[9,1],c[10,1],c[11,1]);
- > printf("vn2=[%f %f %f
- > %f];",c[0,2],c[1,2],c[2,2],c[3,2],c[4,2],c[5,2],c[6,2],c[7,2],c[8,2],
- > c[9,2],c[10,2],c[11,2]);
- > printf("vn3=[%f %f %f
- > %f];",c[0,3],c[1,3],c[2,3],c[3,3],c[4,3],c[5,3],c[6,3],c[7,3],c[8,3],
- > c[9,3],c[10,3],c[11,3]);
- > printf("vn4=[%f %f %f
- > %f];",c[0,4],c[1,4],c[2,4],c[3,4],c[4,4],c[5,4],c[6,4],c[7,4],c[8,4],
- > c[9,4],c[10,4],c[11,4]);

vn0=[-.004719 .016077 -.039617 .085120 -.184715 .627821 .625296 -.182414 .083279 -.038324 .015297 -.004374];

vn1=[.001468 -.005018 .013651 -.036614 .124506 -1.254325 1.254317 -.125201 .037565 -.014622 .005845 -.001998];

vn2=[.018227 -.068488 .178188 -.388799 .808219 -.546326 -.536929 .797894 -.380001 .171311 -.063427 .015693];

vn3=[-.005873 .020073 -.054605 .146457 -.498023 1.018700 -1.017268 .500806 -.150259 .058487 -.023380 .007994];

vn4=[.002597 .016717 -.078875 .193274 -.277439 .137372 .142973 -.272958 .187542 -.072064 .008953 .007211];

Appendix B

Fourier Transform Parts

A simple trick allows a complex signal to be broken into four parts: even-real, even-imaginary, odd-real, and odd-imaginary. Because only a real-valued time domain signal is used, it is possible to simultaneously take the FFT of two signals at once. The individual parts can be reconstructed in the frequency domain. This demonstrated below. In practice, there may be significant overhead when dividing each sample by two. However, for this thesis, the division was not necessary until the Inverse FFT was found. Thus the division was merged with the scaling constant in the Inverse FFT at no extra cost.

Define the even, odd, real, and imaginary parts of a function to be

$$f_e(t) \doteq \frac{f(t) + f(-t)}{2} \tag{B.1}$$

$$f_o(t) \doteq \frac{f(t) - f(-t)}{2} \tag{B.2}$$

$$f_r(t) \doteq \Re\{f(t)\} \tag{B.3}$$

$$f_i(t) \doteq \Im\{f(t)\} \tag{B.4}$$

and the Fourier Transform

$$F(\omega) \doteq \mathcal{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$
(B.5)

Note that

$$\overline{F(\omega)} = F(-\omega), for f_r(t)$$
(B.6)

$$\overline{F(\omega)} = -F(-\omega), for f_i(t)$$
(B.7)

$$\mathcal{F}{f_{e,r}(t)} = \frac{1}{2} \int_{-\infty}^{\infty} f_r(t) e^{-j\omega t} dt + \frac{1}{2} \int_{-\infty}^{\infty} f_r(-t) e^{-j\omega t} dt$$
$$= \frac{1}{2} F_{[f_r]}(\omega) + \frac{1}{2} F_{[f_r]}(-\omega)$$
$$= F_{[f_r]e}(\omega)$$
(B.8)

Using Property B.6, it is easy to show $F_{[f_r]e}(\omega) = \frac{F_{[f_r]}(\omega) + \overline{F_{[f_r]}(\omega)}}{2}$, which implies $F_{[f_r]e}(\omega) = F_{[f_r]e,r}(\omega)$.

$$\mathcal{F}{f_{o,r}(t)} = \frac{1}{2} \int_{-\infty}^{\infty} f_r(t) e^{-j\omega t} dt - \frac{1}{2} \int_{-\infty}^{\infty} f_r(-t) e^{-j\omega t} dt$$

$$(\text{Let } u \doteq -t.)$$

$$= \frac{1}{2} F_{[f_r]}(\omega) - \frac{1}{2} \int_{-\infty}^{\infty} f_r(u) e^{-j\omega u} du$$

$$= \frac{1}{2} F_{[f_r]}(\omega) - \frac{1}{2} F_{[f_r]}(\omega)$$

$$= \frac{1}{2} F_{[f_r]}(\omega) - \frac{1}{2} F_{[f_r]}(-\omega)$$

$$= F_{[f_r]o}(\omega)$$
(B.9)

Again using Property B.6, $F_{[f_r]o}(\omega) = F_{[f_r]o,i}(\omega)$.

The remaining two cases can be shown by repeating this procedure and using Property B.7.