Adaptive Beamforming Using a Microphone Array for Hands-Free Telephony

By

David K. Campbell

Thesis submitted to the faculty of

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Approved:

_____

Dr. A.A. (Louis) Beex

_____ _____

Dr. Jeffrey H. Reed Dr. Ira Jacobs

February 16, 1999

Blacksburg, Virginia

Keywords: Adaptive Beamforming, Microphone Array, Generalized Sidelobe Canceler, MUSIC

Adaptive Beamforming Using a Microphone Array for Hands-Free Telephony

By

David K. Campbell

Committee Chairman: Dr. A.A. (Louis) Beex

The Bradley Department of Electrical and Computer Engineering

Abstract

This thesis describes the design and implementation of a 4-channel microphone array that is an adaptive beamformer used for hands-free telephony in a noisy environment. The microphone signals are amplified, then sent to an A/D converter. The microprocessor board takes the data from the 4 channels and utilizes digital signal processing to determine the direction-of-arrival of the sources and create an output which 'steers' the microphone array to the desired look direction while trying to minimize the energy of interference sources and noise. All of the processing for this thesis will be done on a computer using MATLAB.

The MUSIC algorithm is used for direction finding in this thesis. It is shown to be effective in estimating direction-of-arrival for 1 speech source and 2 speech sources that are spaced fairly apart, with significant results down to a -5 dB SNR even. The MUSIC algorithm requires knowledge of the number of sources *a priori*, requiring an estimator for the number of sources. Though proposed estimators for the number of

sources were examined, an effective estimator was not encountered for the case where there are multiple speech sources.

Beamforming methods are examined which utilize knowledge of the source direction-of-arrival from the MUSIC algorithm. The input is split into 6 subbands such that phase-steered beamforming would be possible. Two methods of phase-steered beamforming are compared in both narrowband and wideband scenarios, and it is shown that phase-steering the array to the desired source direction-of-arrival has about 0.3 dB better beamforming performance than the simple time-delay steered beamformer using no subbands.

As the beamforming solution is inadequate to achieve desired results, a generalized sidelobe canceler (GSC) is developed which incorporates a beamformer. The sidelobe canceler is evaluated using both NLMS and RLS adaptation. The RLS algorithm inherently gives better results than the NLMS algorithm, though the computational complexity renders the solution impractical for implementation with today's technology.

A testing setup is presented which involves a linear 4-microphone array connected to a DSP chip that collects the data. Tests were done using 1 speech source and a model of the car noise environment. The sidelobe canceler's performance using 6 subbands (phase-delay GSC) and using 1 band (time-delay GSC) with NLMS updating are compared. The overall SNR improvement is determined from the signal and noise input and output powers, with signal-only as the input and noise-only as the input to the GSC. The phase-delay GSC gives on average 7.4 dB SNR improvement while the time-delay GSC gives on average 6.2 dB SNR improvement.

Acknowledgements

I would like to thank my academic advisor, Dr. A.A. (Louis) Beex, for his guidance and patience over the last year-and-a-half. I would also like to give thanks to Dr. Beex for closely examining my thesis numerous times, and to Dr. Jeffrey H. Reed and Dr. Ira Jacobs for carefully reviewing my final draft. Thank you to Sundar Sankaran for helping me out in this project in more ways than I can think of. Thanks to my parents for encouraging me to get the M.S. degree and helping me through my college career. This would not have been possible without the support of my friends and family.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Conventional microphones have to be near the user at all times, forcing the user to either wear the microphone or have it move with the speaker (e.g., telephone, teleconferencing). Microphone beamforming has eliminated the need for a movable microphone or telephone. Currently, there are ways to use many microphones to create beam patterns that will focus on one speaker in a room. The motivation for steerable microphones comes mainly from teleconferencing and car telephony applications. The difference in these two applications is that in the car environment, we usually deal with much lower SNR's, while in the teleconferencing environment, we usually have to change the beam direction more often, as well as requiring larger spatial coverage. This thesis deals with the application in a car environment, which allows us to impose some restrictions on the array.

The concept of an adaptive antenna system was first thoroughly treated by Widrow et al. [1]. The motivation at that time was applications in radar and sonar. The two major problems associated with array design are direction-of-arrival estimation for sources, and subsequently creating an optimal output. Significant direction-of-arrival estimators came from Capon in 1969 [2], Schmidt in 1980 [10], and McDonough in 1983 [3]. The first step in adaptive interference canceling was developed by Howells in 1965 [4]. Since then, the most prominent noise rejection structure in use has been Griffith and

Jim's Generalized Sidelobe Canceler [21], which drew from the work of Levin [5] and Frost [19], among others. This implementation has been used succesfully by Nordholm et al. [6] among others. Other techniques for speech enhancement have been studied by Lim and Oppenheim [7].

Recently, the advent of high-speed digital signal processors has allowed the implementation of even the simplest beamformers that were not possible even before 1990. Since that time there have been many different beamformer designs. This thesis focuses on two different beamformer designs: application of the MUSIC algorithm to beamformer output and the Generalized Sidelobe Canceler. These two procedures will be tested with the same microphone data to determine relative and absolute performance.

This thesis also focuses on other aspects of array design: estimators for the number of sources, speech activity detectors, broadband source considerations, near field source considerations, covariance matrix updating, and hardware setup, all of which are critical to the overall performance of the array system.

Chapter 2

Microphone Array Fundamentals

2.1 Representation of Array Input

Figure 1 shows the layout of a linear microphone array consisting of *M* microphones with *K* incident signals from arrival angles θ_k. In this analysis, the incident

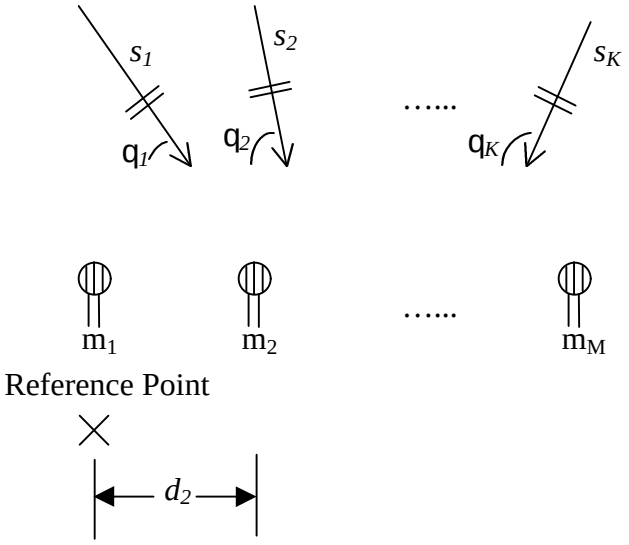

Figure 2.1 Microphone array layout

waves are assumed to be plane waves. The non-planar wave scenario will be discussed in Section 2.7. Each of the microphones in the array will thus receive each of the signals, s_k, but these will be time-delayed or time-advanced versions of each other. There needs to be a reference point for our coordinate system such that we can compute the phase of each signal; any point will do, but for simplicity we will choose microphone 1 as our origin. The distance from microphone m_1 to any microphone m_i is d_i (d_2 is shown in Figure 2.1). The distance the planar sound travels from each source s_k to each of the

microphones m_i relative to the distance to microphone m_1 will be $d_i \cos\theta_k$. The corresponding time delays τ_i to each of the microphones m_i are thus

$$\tau_i = \frac{d_i \cos\theta_k}{v} \tag{2.1}$$

where v is the velocity of sound (343 m/s). The input x_i at each of the microphones due to signal s_k is just

$$x_i(t) = s_k(t - \tau_i). \tag{2.2}$$

One other thing to note here is that a source arriving at a θ_k between 0° and -180° will generate the same $x_i(t)$ in (2.2) as a source arriving at its positive angle counterpart. This can be easily seen physically, and also in the plane wave hypothesis because $\cos(\theta_k) = \cos(-\theta_k)$. Assuming a narrowband scenario (in Section 2.8 we will discuss the wideband case), this signal can be represented as a phase shift of the incoming signal by the notation

$$x_i(t) = s_k(t) e^{-j\omega_0 \tau_i} \tag{2.3}$$

or equivalently,

$$x_i(t) = s_k(t) e^{-j 2\pi f_0 d_i \cos\theta_k / v} \tag{2.4}$$

and with $\lambda = v/f_0$,

$$x_i(t) = s_k(t)e^{-j2\pi d_i \cos\theta_k / \lambda}. \tag{2.5}$$

The reason for representing the signal as in (2.3) is because it is computationally easier to work with than the representation in (2.2). Assume now that there are *K* sources and some noise arriving at each of the microphones. The total received signal at the i^{th} sensor is a combination of noise and *K* incoming signals:

$$x_i(t) = \sum_{k=1}^{K} s_k(t)e^{-j2\pi d_i \cos\theta_k / \lambda} + n_i(t) \tag{2.6}$$

2.2 Array Output

The oldest technique to generate the output of an array is the delay-and-sum technique. The output of the array is written as

$$y(t) = \sum_{i=1}^{M} w_i x_i(t - \tau_i). \tag{2.7}$$

The weighting factors w_i and delays τ_i in (2.7) are chosen to help enhance the beam shape and reduce sidelobe levels. For our model, we want to generate the output using real weights also, but with phase delays in the representation instead of time delays. For now, weight factors for all sensors will be $1/M$; this allows an 'averaging' such that the output has nearly the same amplitude as the input. The output for our model is thus

$$y(t) = \frac{1}{M}\sum_{i=1}^{M} e^{j2\pi d_i \cos\theta/\lambda} x_i(t). \tag{2.8}$$

As for the narrowband scenario (wavelength has small variations around λ), you can see that (2.8) phase shifts the signals at the sensors such that the energy arriving from q is coherently combined. Using this solution, it is necessary to generate the quadrature component of the incoming signal, so that the signal will have phase characteristics that allow us to phase shift the signal as in (2.8). Then the actual audio output will be the real part of (2.8).

2.3 Output Covariance Matrix

All of the classical beamforming techniques use the output covariance matrix to determine source direction-of-arrival estimates. Another useful property of the covariance matrix is that we can see the output powers at each of the sensors, and the matrix can be easily modified to make the gains at all of the sensors equal. For a 4-microphone scenario we have

$$\mathbf{R} = E\left[\mathbf{x}(t)\mathbf{x}^H(t)\right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} \tag{2.9}$$

where $\mathbf{x}^H$ denotes the Hermitian (complex-conjugate) transpose of **x**. With **N** denoting the total number of time snapshots used to estimate **R**, r_{ij} is written as

$$r_{ij} = E\left[x_{i,n} x_{j,n}^{*}\right] = \frac{1}{N}\sum_{n=1}^{N} x_{i,n} x_{j,n}^{*} \ . \tag{2.10}$$

2.4 MUSIC Algorithm

A powerful technique was developed by Schmidt [10] to determine direction-of-arrival angles for multiple sources. The covariance matrix in (2.9) can be expressed in terms of its *M* eigenvalues and eigenvectors. Assuming that there are *K* sources arriving into the array, the largest *K* eigenvalues of **R** represent a function of the power of each of the *K* sources, while their eigenvectors are said to span the *K* dimensional signal subspace of **R.** The smallest *M-K* eigenvalues represent the noise power, and theoretically they are equal, under the white noise assumption. The eigenvectors that are associated with these eigenvalues are said to span the *M-K* dimensional noise subspace of **R.** It is shown [10] that the eigenvectors associated with the smallest *M-K* eigenvalues are orthogonal to the direction vectors corresponding to the arrival angles of the sources. The MUSIC algorithm can be written down computationally as

$$P_{MUSIC}(\theta) = \frac{1}{\sum_{i=K+1}^{M} \left|\beta_i^{H} \mathbf{a}(\theta)\right|^2} \tag{2.11}$$

where the β_i represent the eigenvectors, and $\mathbf{a}(\theta)$ represents a vector of phase factors for each array element:

$$\mathbf{a}(\theta) = 1/M\left[e^{-j2\pi d_1 \cos\theta/\lambda} \quad e^{-j2\pi d_2 \cos\theta/\lambda} \quad \ldots \quad e^{-j2\pi d_M \cos\theta/\lambda}\right]^{\mathrm{T}} \tag{2.12}$$

with the first term equal to 1 since d_1 is defined to be zero. As θ is varied over [0°, 180°], $P_{MUSIC}(\theta)$ can be graphed and examined to see where the peaks are (see Chapter 3,

starting at Section 3.3). If we assume that the noises into the microphones are uncorrelated with each other and have a common variance, then this method can theoretically distinguish between arbitrarily close targets, unlike most other estimators. $P_{MUSIC}(\theta)$, though, does not estimate the signal power associated with each arrival angle; other estimators (Capon's minimum variance estimator, linear prediction based estimator) could be used for that purpose. The MUSIC algorithm is used for our array problem because we want to get the best estimate of the directions the sources are coming from.

2.5 Estimation of Number of Sources

Since the MUSIC algorithm requires knowledge of the number of sources to determine the arrival angles, we need to first estimate this number. If we ordered the eigenvalues λ_i of the matrix **R** such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$, then we should notice in practice that the *M-K* lowest eigenvalues are clustered together and thus we should easily be able to determine how many sources there are. The problem here is that it is a subjective *-measure – how close is close? Since in practice we cannot subjectively measure the number of sources at every instant, we desire a criterion that will do this for us. Preferably we would like to use an estimation scheme that also computes the significance level for us.

Several solutions have been shown to be effective in estimating the number of sources; the most prominent are Akaike's Information Criterion [11] and the Minimum Description Length Criterion [12]. Wax and Kailath [13] extended these criteria to estimate the number of signals. Both methods require been computation of the likelihood ratio for the (*M-K*) lowest eigenvalues of **R**, and this has been shown to be [14]

$$\delta_K(\lambda) = \frac{\frac{1}{M-K}\sum_{i=K+1}^{M}\lambda_i}{\left[\prod_{i=K+1}^{M}\lambda_i\right]^{\frac{1}{M-K}}} \tag{2.13}$$

which is the ratio of the arithmetic and geometric mean of the eigenvalues. Akaike's Information Criterion (AIC) can be written as

$$AIC(K) = 2N(M-K)\ln\delta_K(\lambda) + 2\nu(K,M) \tag{2.14}$$

where $\nu(K,M)$ denotes the number of independent parameters that are to be estimated for a given K. This is found to be [8]

$$\nu(K,M) = K(2M-K)+1\ . \tag{2.15}$$

The optimal solution is the value of K that minimizes (2.14). The Minimum Description Length Criterion (MDL) is similar to the AIC and is written as

$$MDL(K) = 2N(M-K)\ln\delta_K(\lambda) + \nu(K,M)\ln N\ . \tag{2.16}$$

Both the AIC and MDL will be evaluated later on in terms of their effectiveness to estimate the number of sources.

2.6 Microphone Spacing Considerations

The spacing of microphones inside a car will probably be fixed. The benefits of a moving microphone array are marginal compared to the mechanical complexity of

implementation. Therefore it is necessary to consider different microphone configurations and devise an optimal configuration for our problem.

An important consideration is that we want to avoid aliasing in the spatial frequency domain. This means that for the temporal frequencies that we are interested in, we want phase delays between adjacent microphones to be between -180° and 180°. This is shown by looking at (2.5), which gives us the spatial frequency μ of each incoming signal s_k into microphone m_i as (assuming each adjacent element is spaced apart by d)

$$\mu_k = \frac{2\pi d \cos\theta_k}{\lambda}. \tag{2.17}$$

We want μ_k to go from $-\pi$ to π; this requires that $d/\lambda = ½$. So we do not want our interelement spacing any smaller than $\lambda/2$. Also, if our interelement spacing is greater than $\lambda/2$, we may have multiple mainlobes. This is shown in [9], pp. 41-42. It can be seen that as d/λ gets smaller, fewer sidelobes and nulls appear in the directional pattern, eventually forming an omnidirectional pattern as d/λ goes to zero.

The array beam pattern for a 4-element linear equispaced array is shown Figure 2.2. So for a 4-element array we have 3 nulls and a main lobe to take into consideration when choosing the optimal position for the angle that we want to steer the array to. It may not be optimal to choose the exact angle that the source is coming from as the steering angle of the array. It may be better to choose an angle that nulls the interference source(s), and have a somewhat attenuated gain for the desired source, than to choose a unity gain for the desired source (with less attenuated interference sources). In Section 2.8 we will examine considerations for wideband sources.

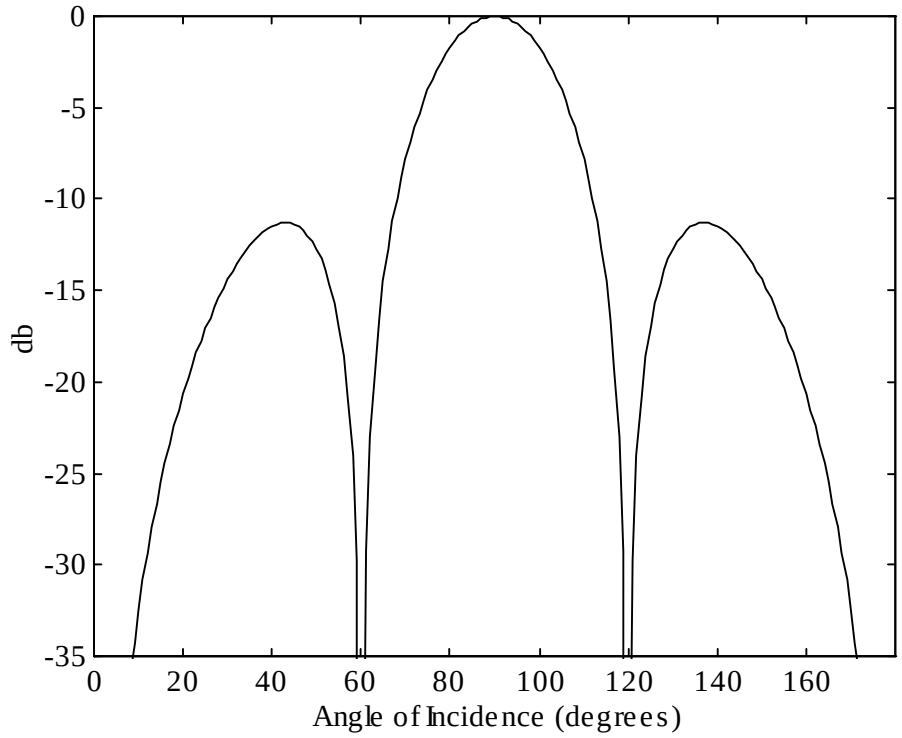

Figure 2.2 Normalized array beam pattern for 4 element array with d/λ=1/2

2.7 Near Field Considerations

Most of the literature on adaptive arrays uses the far-field (planar wave) hypothesis for the incoming wavefront. In situations where microphone arrays are used, the near-field (spherical wave) hypothesis must be taken into account. The planar wave configuration has 1 unknown parameter, θ, which can be solved for using algorithms like MUSIC. The spherical wave configuration, though, has two unknown parameters, θ and r (the radial distance from the source to the array). This makes the model much more complex, and no analytical solution has been developed yet to solve this problem. Therefore, we would like to know just how far off the plane wave assumption is for a given r and a given interelement spacing.

The setup for an incoming wave from (r,θ) from the center of a linear 4-microphone array with interelement spacing d is shown in Figure 2.3. Under the spherical

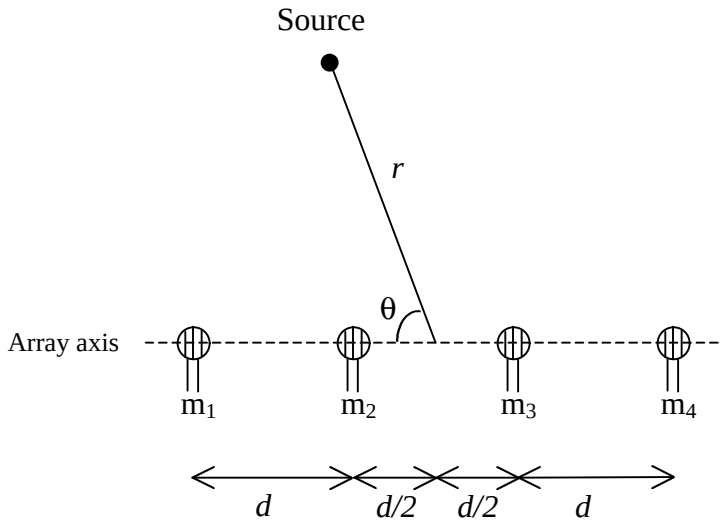

Figure 2.3 Four-microphone array configuration for spherical wave hypothesis

wave hypothesis, the distance a wave travels from the point source to any point is equal to the length of the line connecting the two points. Therefore, the distance the wave travels to each of the microphones, r_i, is equal to the distance from the point source to each of the microphones. This distance can be found using the parameters r,θ, and d, and the law of cosines. So the distances (r_1, r_2, r_3, r_4) to each of the microphones (m_1,m_2,m_3,m_4) are

$$\begin{aligned} r_1 &= \sqrt{r^2 + \left(3d/2\right)^2 - 3dr\cos\theta} \\ r_2 &= \sqrt{r^2 + \left(d/2\right)^2 - dr\cos\theta} \\ r_3 &= \sqrt{r^2 + \left(d/2\right)^2 + dr\cos\theta} \\ r_4 &= \sqrt{r^2 + \left(3d/2\right)^2 + 3dr\cos\theta} \end{aligned} \tag{2.18}$$

The relative time delay into each of the microphones can be found by subtracting the smallest r value from each of the r values (r_1, r_2, r_3, r_4), and then dividing the result by the speed of sound. Figure 2.4 shows the comparison between the relative delays under the spherical and planar wave hypothesis for r = 1 m and d = 0.2 m. As can be seen from Figure 2.4, the biggest discrepancies between the spherical and plane wave assumptions occur around θ=90°. Also, the two hypotheses will produce equal results at angles along the array axis (θ=0° and 180°).

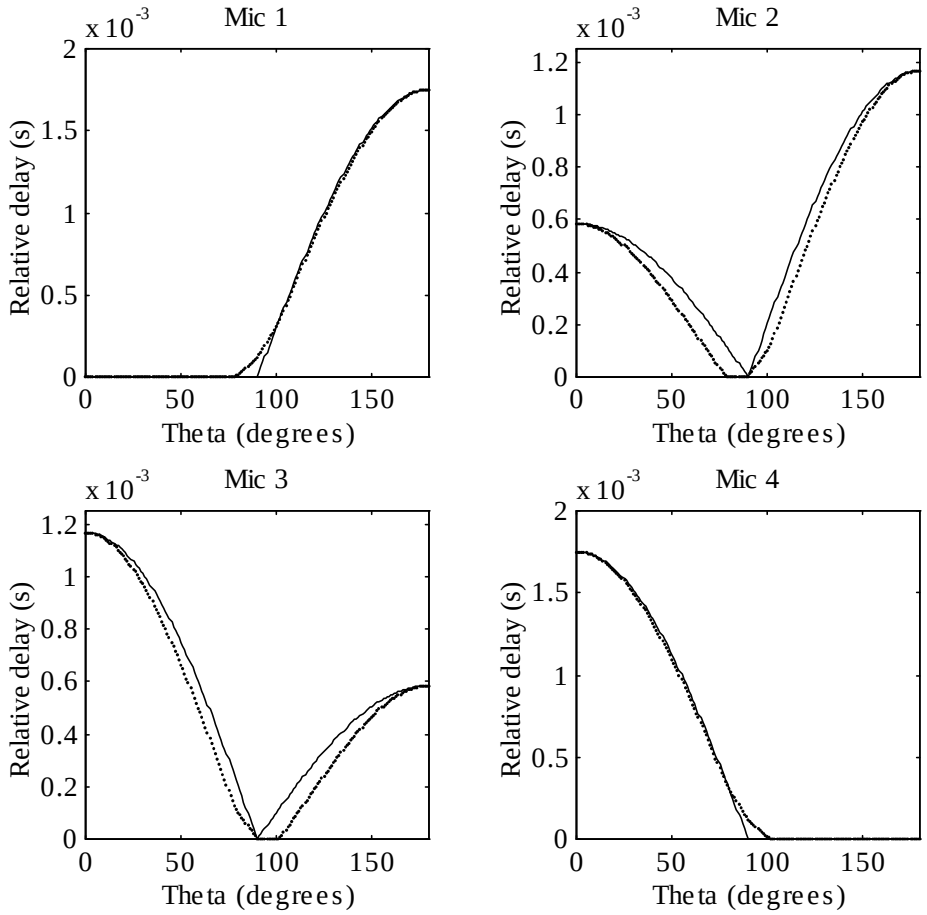

Figure 2.4 Relative time delays from source at r = 1m from array to the 4 microphones with d = 0.2 m (---- plane wave, …. spherical wave)

2.8 Broadband Source Considerations

Broadband array processing is much more difficult than narrowband array processing. The problem lies in the expression for $P_{MUSIC}(\theta)$. In (2.12), we have to use a specific wavelength to determine $P_{MUSIC}(\theta)$. The estimate $P_{MUSIC}(\theta)$ is thus frequency sensitive, therefore we need to find a way to improve our estimate, hopefully with minimal computational complexity. One way to deal with this is to use conventional subband techniques, but this only reduces computational complexity [15] and does not help us in determining the source directions more accurately.

Another way to deal with broadband signals is to apply narrowband filters to the signals so that we split the overall problem into a number of subproblems. Therefore we will have narrower band signals, and if we choose λ in (2.12) as the center of each of the frequency bands, we can make the error obtained from the estimate arbitrarily small by increasing the number of subbands. If we make the frequency bands very small, not only will we have to deal with many more frequency intervals, we will require higher and higher order filters. So we have to choose the number of frequency bands to compromise between computational complexity and performance. The frequency band that we are interested in is the telephone frequency range, 400-3400 Hz. Also, because the frequency that will be used in (2.12) will be off from the actual frequency in each by the ratio f/f_0 , where f_0 is the center frequency of each band, it will be best to space the frequency bands with equivalent ratios f_{hi}/f_{lo}, corresponding to the upper and lower frequencies of each band. Table 2.1 shows the frequency ratios for different numbers of subbands. In

Section 3.7 we will examine the effects of choosing different numbers of bands, and choose the number of frequency bands used for the rest of this thesis.

Table 2.1 Frequency ratios f_{hi}/f_{lo} for different numbers of subbands

Bands	4	5	6	7	8	9	10	11	12	13
f_{hi}/f_{lo}	.586	.652	.700	.737	.765	.788	.807	.823	.837	.848

Chapter 3

Simulations for DOA Estimation

3.1 Testing Setup

In this chapter, we are simulating the performance of the direction-of-arrival estimation capabilities of the microphone array using as many different parameters as possible to determine what effect each of the different parameters will have on the overall array performance. Parameters will be discussed for single and multiple narrowband sources, and single and multiple speech sources. The programs that contain the algorithms used in the simulations and the actual tests were written in MATLAB. First we need to determine what kind of noise exists in the car environment so that we can incorporate noise into our simulations.

3.2 Simulating Car Noise

For our tests to be as close as possible to the actual situation, we would like to examine the car noise environment. We would like to know the frequency content of the noise, as well as any major discrete noise sources. Tests to determine these parameters were carried out using a two-microphone array mounted on top of the dashboard of a Dodge Caravan. The omnidirectional microphones were placed an inch from the dash. The 2 microphones were 11 cm apart, positioned halfway from the sides of the car. The sampling rate for the test was 8 kHz. Tests were done at different speeds, and with the driver's window opened and closed.

The tests showed some useful trends. The first thing to note is that the noise power increases as speed increases. This is due to the engine noise increasing, and wind hitting the vehicle at faster speeds. Also, the noise remains fairly constant as long as the speed doesn't change by too much. Oncoming traffic produces a quick transient, and is noticed much more when the driver's window is open, but lasts a very short time, less than half a second. A passing car, on the other hand, usually takes more than two seconds. The noise spectrum due to a passing car shows a little more emphasis in the frequency range from 200-1000 Hz, but it is difficult to differentiate noise spectrums with and without a car passing. Other forms of noise result from road anomalies (which are short-lived but can sometimes be periodic, as due to bridge holes created for water drainage) and horns and sirens. These disruptions will be treated in Chapter 7 for their effect on system performance.

We would like to have an idea of what car noise looks like spectrally, as well as in terms of relative power so that we can perform more accurate simulations. The next four figures, Figure 3.1 through Figure 3.4, show spectral powers of car noise in 4 different situations; at speeds of 25 and 55 mph and with the driver's window open and closed. It is seen from each of these figures that car noise behaves in a manner similar to pink noise (defined as 3 dB/octave rolloff), but with about 5 dB/octave rolloff for low frequencies going to about 15 dB/octave for high frequencies. It is also notable that the noise spectrum with the windows shut at 55 mph is similar to the noise spectrum with the windows open at 25 mph (Figures 3.2 and 3.3). Now we would like to look at the spatial frequency content of the noise.

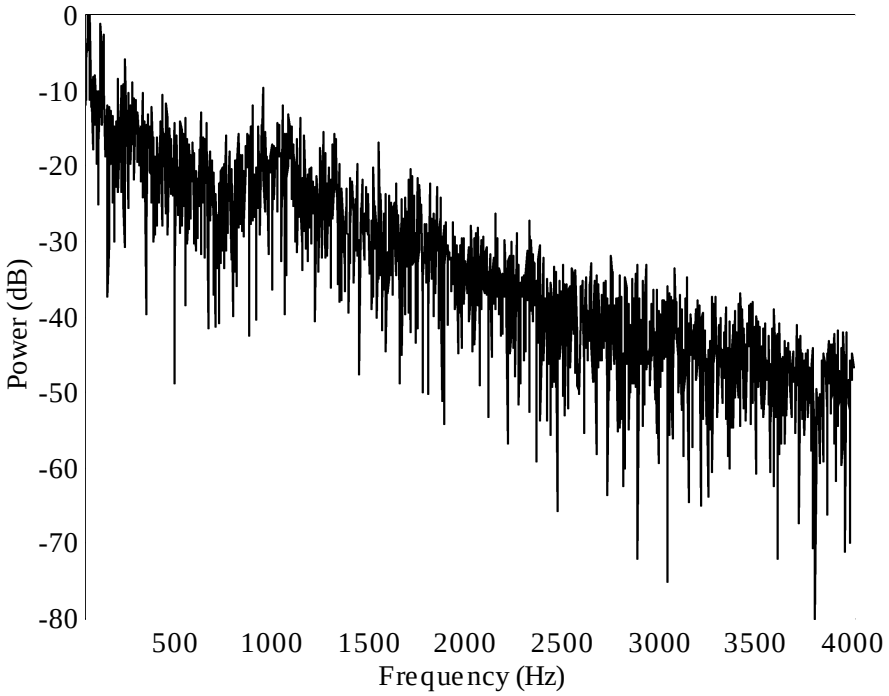

Figure 3.1 Car noise from 40-4000 Hz at 55 mph, driver's window open

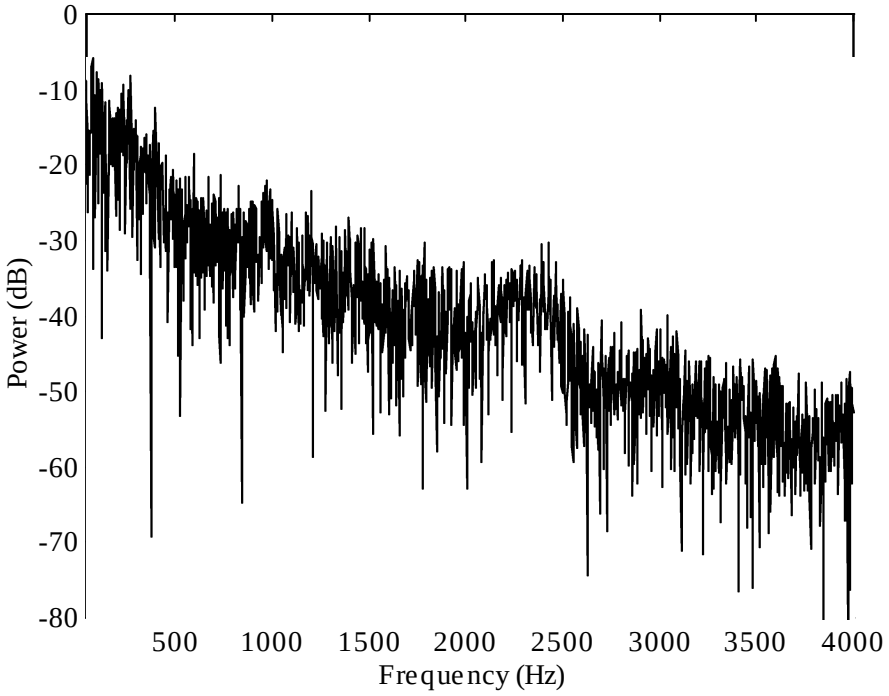

Figure 3.2 Car noise from 40-4000 Hz at 55 mph, windows shut

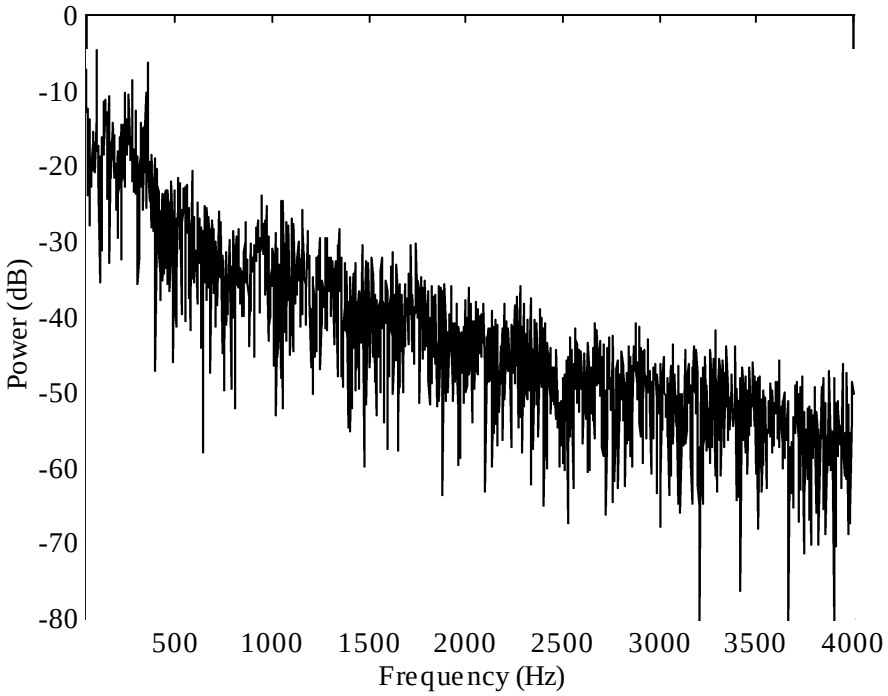

Figure 3.3 Car noise from 40-4000 Hz at 25 mph, driver's window open

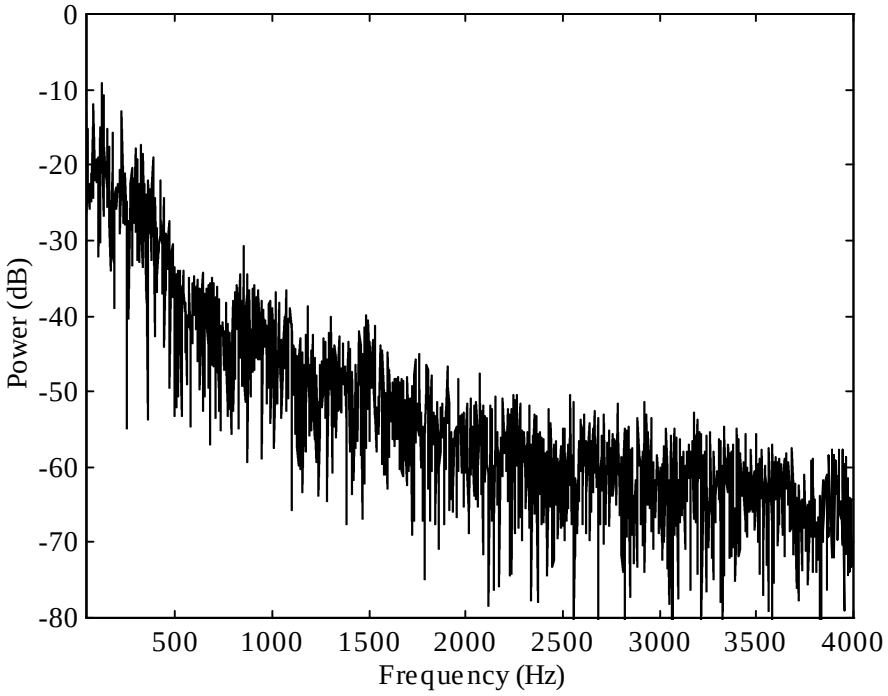

Figure 3.4 Car noise from 40-4000 Hz at 25 mph, windows shut

It is a little more difficult to analyze the spatial frequency of the incoming noise. Even though we are using only 2 sensors, we should be able to get some idea of where the noise is coming from. The spatial frequency response for the noise-only case was examined for different cases using Capon's minimum variance estimator to determine powers at each arrival angle. For these measurements, 0° corresponds to the array line on the driver's side, and 180° corresponds to the array line on the passenger side. The array cannot tell whether sound is coming from behind the array or in front of the array. We will first look at when the car is going 55 mph with the driver's window closed. Figure 3.5 and Figure 3.6 show power estimation results for the prominent noise frequency range. From these graphs it is seen that the noise is more omnidirectional for lower

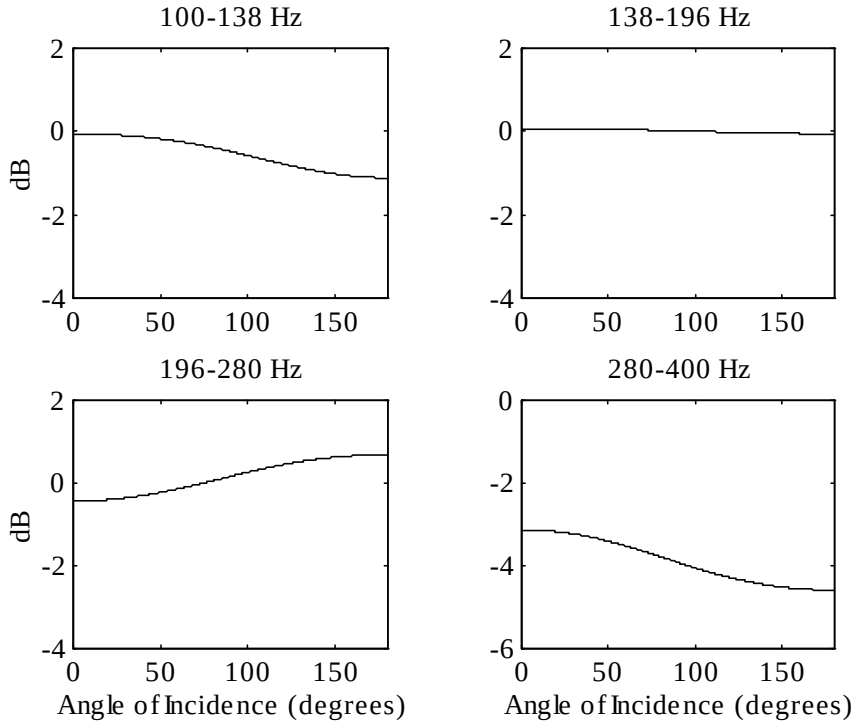

Figure 3.5 Power estimations in car at 55 mph with window closed, 100-400 Hz

frequencies (100-400 Hz), though some of the flatness of the estimations is due to the small interelement spacing. At higher frequencies the noise is more centered towards 90°. This is what we should expect; with the windows shut, most of the noise comes from the engine.

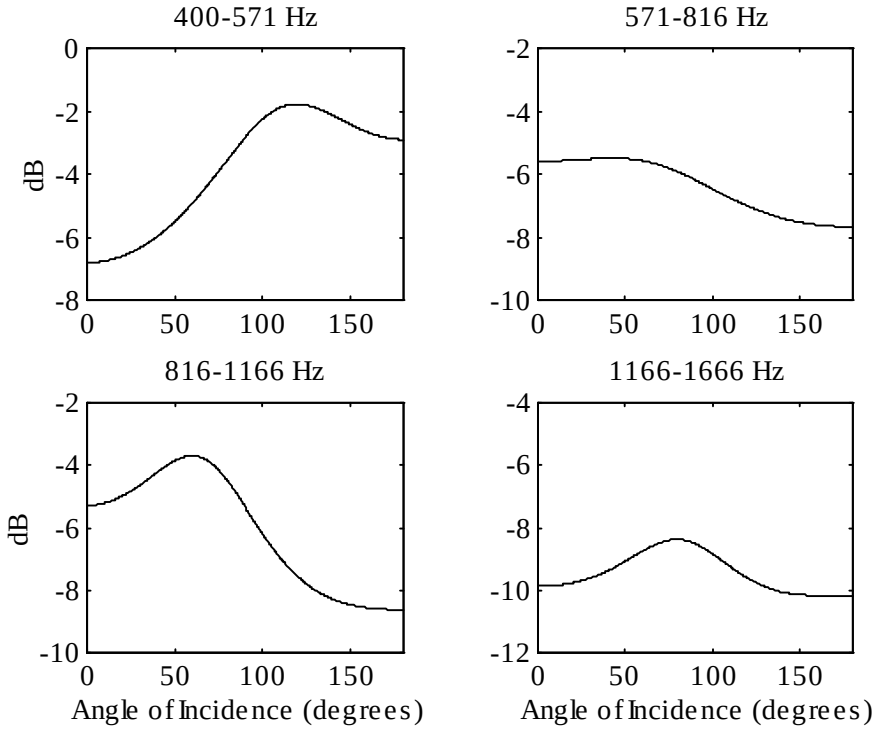

Figure 3.6 Power estimations in car at 55 mph with window closed, 400-1666 Hz

We will look at a case with the driver's window open to examine the noise directionality. Results with the car going 25 mph are shown in Figure 3.7 and Figure 3.8. Here there is more directionality towards the driver's window at low frequencies. The higher frequency energy is centered towards 90° as before. We have discovered some trends in the directionality of the car noise, although ideally we would like to model all directional noise sources as well as diffuse noise.

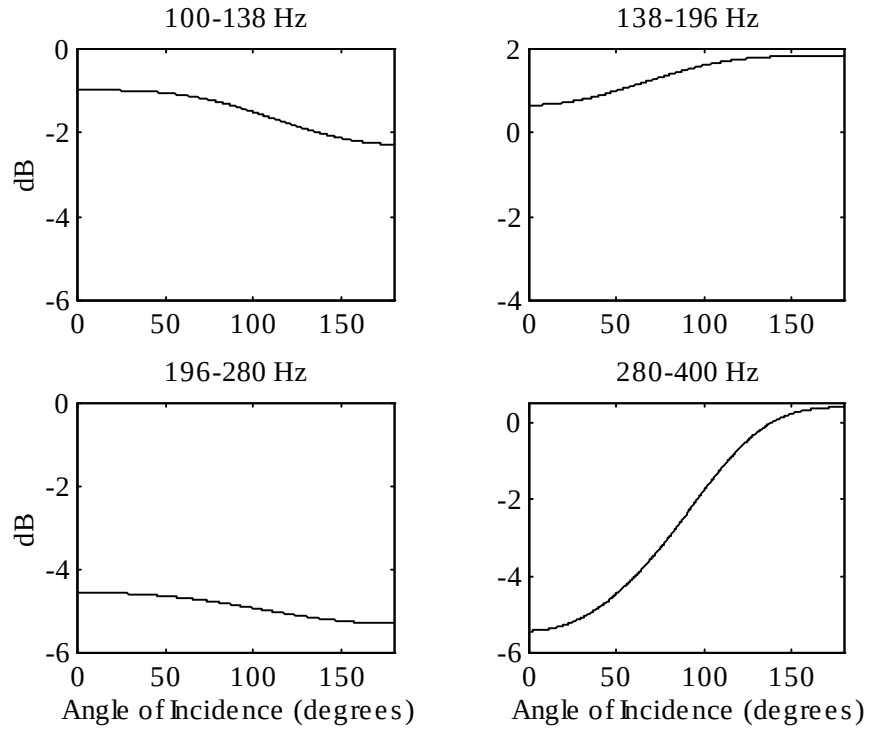

Figure 3.7 Power estimations in car at 25 mph with window open, 100-400 Hz

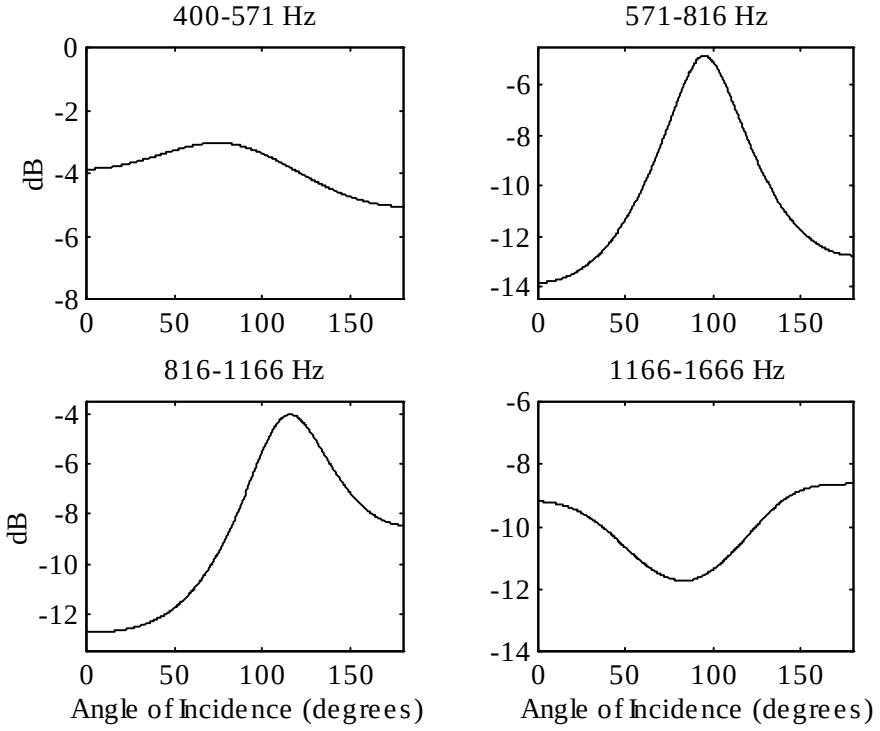

Figure 3.8 Power estimations in car at 25 mph with window open, 400-1666 Hz

Another result we would like to get from these simulations is the relative SNR's in the car environment. Table 3.1 shows the results for a typical talkspurt. These results do

Table 3.1 SNR's for different car situations

Mph / window	25 closed	25 open	55 closed	55 open
SNR (dB)	10.6	5.8	4.7	-0.7

not include traffic noise. Traffic noise greatly alters the SNR's when the driver's window is open, but only for short amounts of time. In general it is found very unlikely that we will have SNR's less than -5 dB. It was also found that air conditioning noise is not significant unless it's on full blast when the car is otherwise quiet. This noise also is omnidirectional.

Unfortunately, we were not able to obtain data from a 4-microphone array in the car environment. This would have allowed us a more accurate way of determining system performance. What we can do with the measured noise data is to simulate these different pink noises coming from different angles. This is the main way that we will simulate noise for the rest of this thesis.

3.3 One Narrowband Source

The goal of this section is to show, simulating a single narrowband source, how well the MUSIC algorithm works in detecting the actual arrival angle of the source. A sine wave was generated at a specified frequency, and 500 samples of this sine wave were phase-delayed appropriately, under the far-field assumption, to create each of the 4

microphone inputs. The next step in the program was to create analytic signals from these time samples. This was accomplished by designing a Hilbert transformer, whose characteristics are shown in Figure 3.9. The Hilbert transformer is a 102^{nd} order filter based on the Kaiser window design. As can be seen in Figure 3.9, this filter closely approximates the ideal Hilbert transformer, which has unity gain for all frequencies and a phase angle of -90° for positive frequencies and 90° for negative frequencies. The Hilbert transformer was convolved with each of the 4 inputs, and then the 500 points after the delay were taken as the imaginary part of the complex signal.

Now that the analytic signal was generated, the covariance matrix can be estimated, as in (2.9). The 4 eigenvalues and eigenvectors are then computed. Next we have to calculate the $\mathbf{a}(\theta)$ vector, as in (2.12). For now, λ is the wavelength corresponding to the frequency bin that is maximum in absolute value, as calculated by taking the FFT of the input. Once $\mathbf{a}(\theta)$ is calculated, we can use the MUSIC algorithm to find $P_{MUSIC}(\theta)$, according to (2.11). This equation requires knowledge of the number of sources. For now we will assume a priori knowledge of this number; we will do tests later to determine how well our source estimators in Section 2.5 do. Figure 3.10 shows the result of the MUSIC algorithm with a sine wave source at 120°.

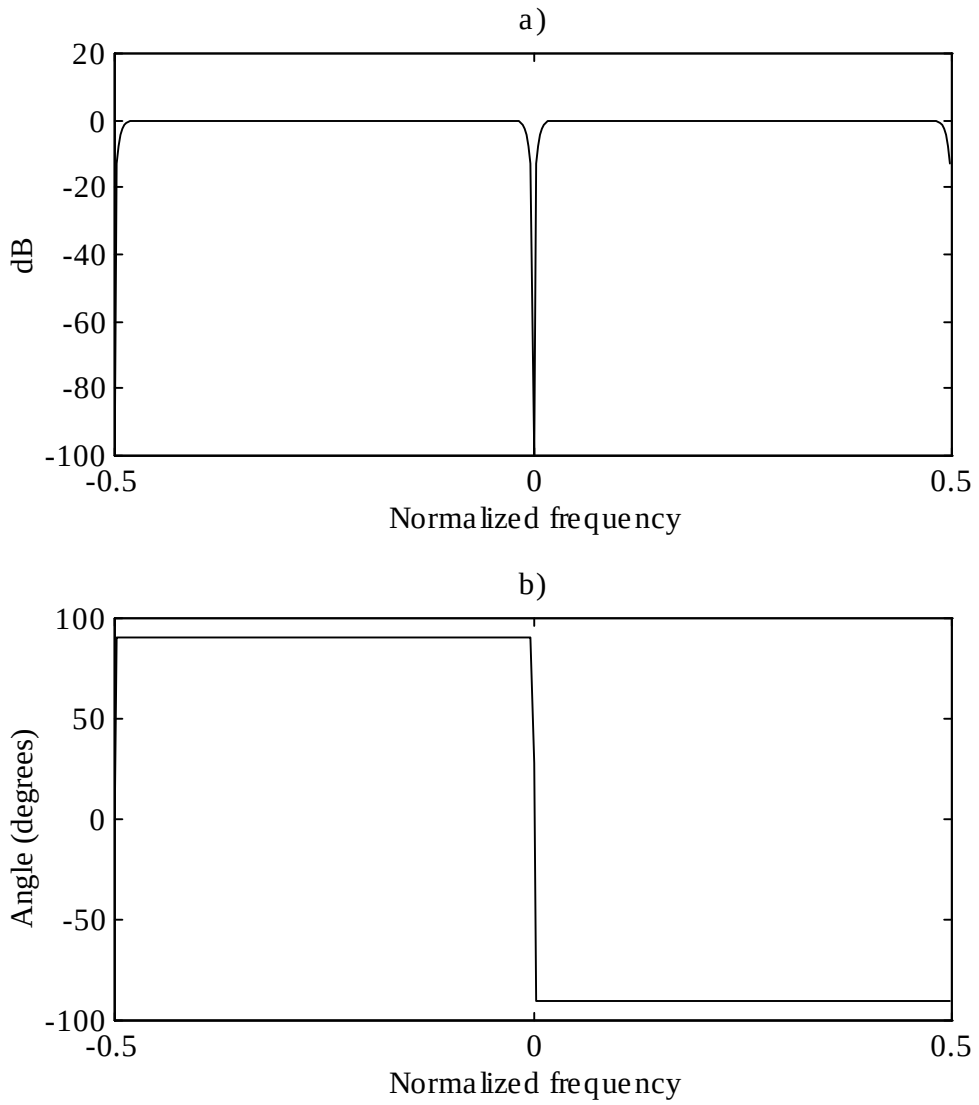

Figure 3.9 Magnitude (a) and phase (b) response of Hilbert Transformer

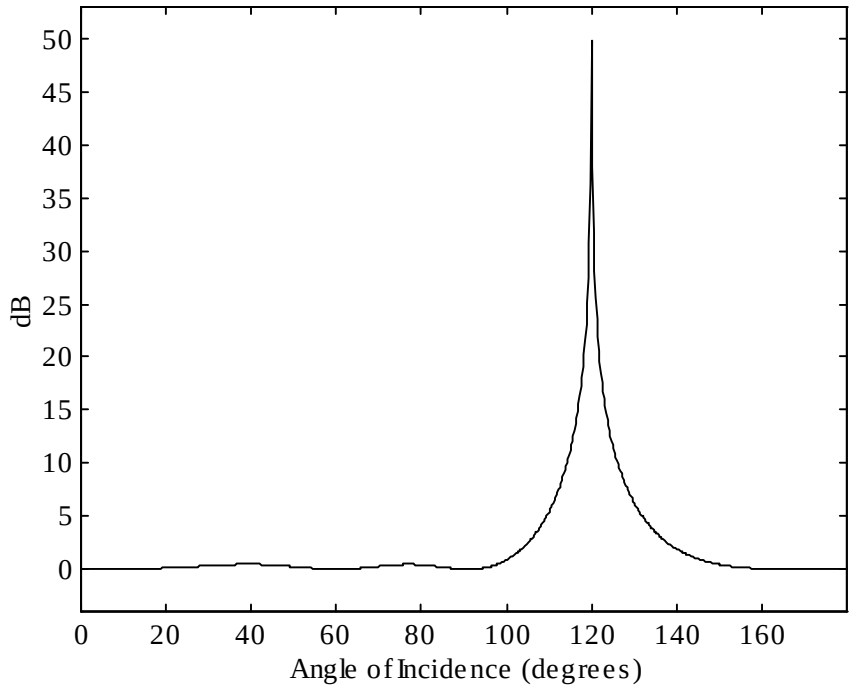

Figure 3.10 $P_{MUSIC}(\theta)$ for sine wave at 120°

The estimator shows a sharp peak at exactly 120° (resolution of 0.1°). The sine wave was modeled as a plane wave. Also, the interelement spacing was equal to $\lambda/2$. Table 3.2 shows the estimator's performance for 10° increments of the source arrival angle. The estimator used 500 snapshots to estimate the covariance matrix **R**.

Table 3.2 Comparison of actual and MUSIC DOA angles (degrees), 1 sine source.

Actual DOA	0	10	20	30	40	50	60	70	80	90
MUSIC DOA	0.0,180.0	10.0	19.9	29.9	40.0	50.0	60.0	70.0	80.0	90.0

Actual DOA	100	110	120	130	140	150	160	170	180
MUSIC DOA	100.0	110.0	120.0	130.0	140.0	150.1	160.1	170.0	180.0,0.0

From Table 3.2, it is seen that the MUSIC DOA performance at any angle θ is mirrored by the performance at angle (180-θ). This is due to the fact that all of the sensor signals undergo the same delays, except reversed for these two scenarios. There are essentially three different performance regions as a function of θ. Generally, $P_{MUSIC}(\theta)$ has a similar shape to that in Figure 3.10. The P_{MUSIC} graph in Figure 3.11 is unique in its sharpness. The extremely sharp peak at 90° can be attributed to the fact that the inputs are already in phase, giving us an exact correlation matrix **R** yielding 1 large eigenvalue, with the other 3 eigenvalues being exactly zero. In practice, of course, we shouldn't expect this since there will always be noise even 30 dB below the source, as well as near field effects and quantization constraints. The other performance region occurs as θ goes below 20° and above 160°. This is shown in Figure 3.12.

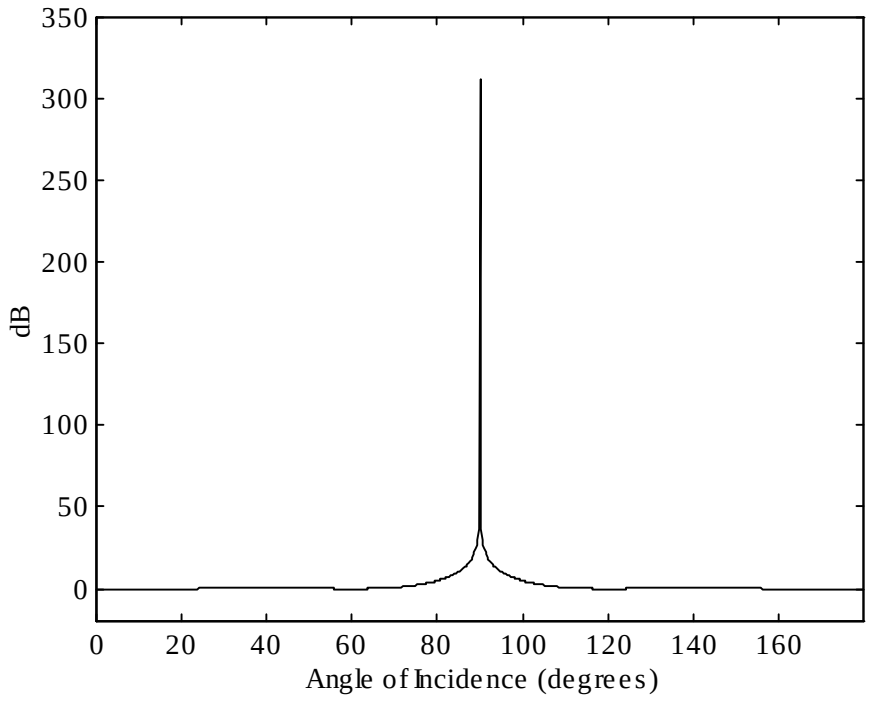

Figure 3.11 $P_{MUSIC}(\theta)$ for sine wave at 90°

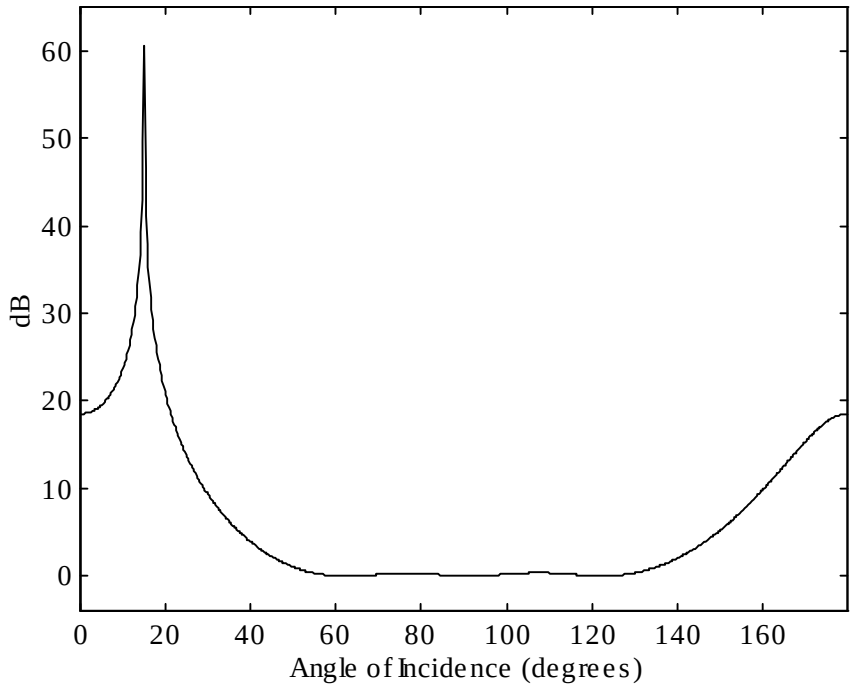

Figure 3.12 $P_{MUSIC}(\theta)$ for sine wave at 15°

3.4 One Narrowband Source with Noise

This section deals with simulated noise being added to the array. For direction-of-arrival purposes we would like to look at the worst-case noise scenario, and that is uncorrelated white noise at all inputs. We will look at angular estimates at different SNR's. With *a priori* knowledge of the signal frequency, we'd like to see how well the MUSIC algorithm works. Table 3.3 shows results at the different SNR's, using 500 snapshots to generate the correlation matrix estimate. With an SNR of 10 dB, the MUSIC algorithm works pretty well. At an SNR of 0 dB, the performance suffers a little, but is still very accurate. The algorithm still works well in high noise environments. The variability of the angle estimations increases with lower SNR's, but not too rapidly. The average peak separations from the average level for the 0, -5, and -10 dB SNR's were

Table 3.3 DOA results for 1 narrowband source with uncorrelated white noise

Actual DOA	10	40	70	100	130	160
SNR=10 dB	10.5	40.0	70.2	100.1	129.8	159.5
SNR=0 dB	7.8	39.4	70.1	99.9	130.0	161.3
SNR=-5 dB	11.0	39.1	69.2	100.3	129.2	166.7
SNR=-10 dB	0	45.3	68.9	99.9	131.7	160.4

24 dB, 18 dB, and 14 dB. Going to lower SNR's, the algorithm breaks down when the eigenvalues from the correlation matrix get closer to one another. It then cannot be determined which eigenvectors span the noise subspace, which we need to know for our algorithm. This occurs around -20 dB. It is very unlikely, though, that we will come across a situation in a car with SNR of less than -5 dB. It is also noteworthy to mention that the DOA results will improve as the number of snapshots used to generate the correlation matrix increases.

3.5 Multiple Narrowband Sources

In this section we will examine the performance of the MUSIC algorithm with multiple narrowband sources as the input. A two-source case is shown in Figure 3.13. The 900 Hz sine wave arrived at 60° and the 600 Hz sine wave arrived at 150°. The sine waves were at equal amplitudes. The arrival angles, as indicated by the graph, are 53.1° and 133.8°. This demonstrates the frequency selectiveness of the MUSIC algorithm, because the representation of the input that the MUSIC algorithm is based on, (2.6), requires a single λ. The value for λ used in (2.12) to compute P_{MUSIC} was chosen to correspond to the average of the two source frequencies (750 Hz). Therefore, with the frequencies of the sources off from the value used in (2.12) by 20%, the corresponding

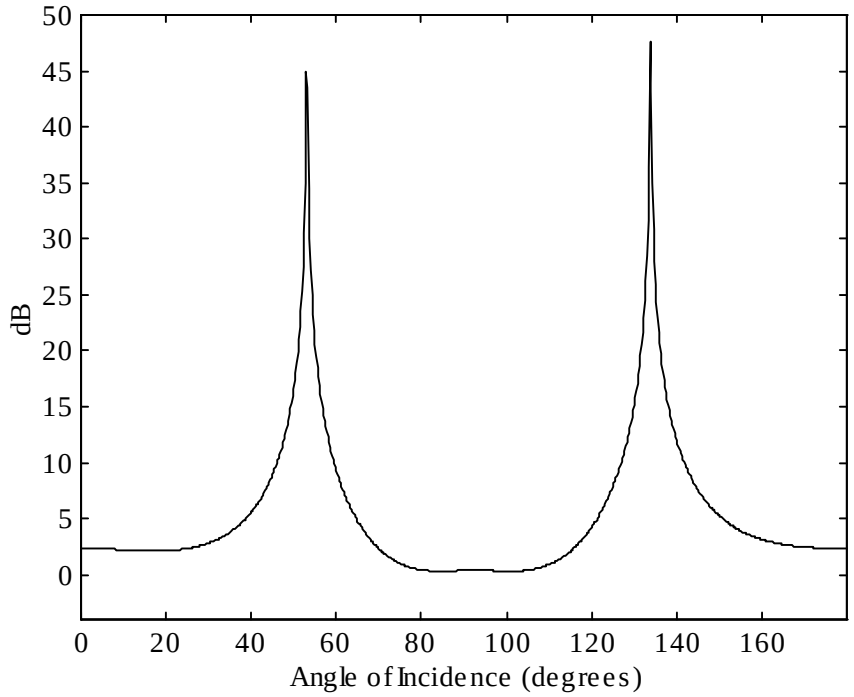

Figure 3.13 $P_{MUSIC}(\theta)$ for 2 sine waves at 60° (900 Hz) and 150° (600 Hz).

predicted arrival angles of the sources were off by 6.9° and 16.2°. This problem must be addressed because speech signals are mostly in the 100-4000 Hz range. Several simulations were run with one of the sources varying from 500-996 Hz while the other source was fixed at 1 kHz to see how much the frequencies affect the accuracy of the output. The results are shown in Table 3.4. If we are to use the MUSIC algorithm in our final speech detection system, we will have to subdivide the input into frequency bands. From there we can do our array processing on each of the bands.

Table 3.4 MUSIC estimates of DOA of sine wave at 60° (variable frequency f_l) and sine wave at 150° (1 kHz).

f_l (Hz)	500	600	700	800	900	950	975	985	992	996
60°	48.2°	51.3°	54°	56.2°	58.2°	59.1°	59.6°	59.7°	59.8°	60.0°
150°	125.0°	130.5°	135.5°	140.3°	145.1°	147.6°	148.8°	149.3°	149.6°	150.0°

Another aspect of the multiple source scene is determining the limit of resolution of this algorithm. Now in a practical sense, if the sources are very close to each other, with the beamforming capabilities of a 4 microphone array it would be impossible to make the beam sharp enough to produce any noticeable degradation of only one of the sources. Though in theory the MUSIC algorithm can resolve arbitrarily close sources [10], we would like to see how accurate it is at close angular spacings. Figure 3.15, and Figure 3.16 show the results with an angular difference of 15°, 10°, and 5°, respectively. The frequencies used for the sources were 996 Hz and 1 kHz. The angles predicted by the MUSIC estimator were no more than .5° off from the actual arrival angles. Since the simulated conditions are essentially perfect conditions, it seems highly unlikely that this estimator will be able to resolve multiple sources less than 10° apart practically, due to the quantization of the input signals. In order to achieve a desired resolution with a given

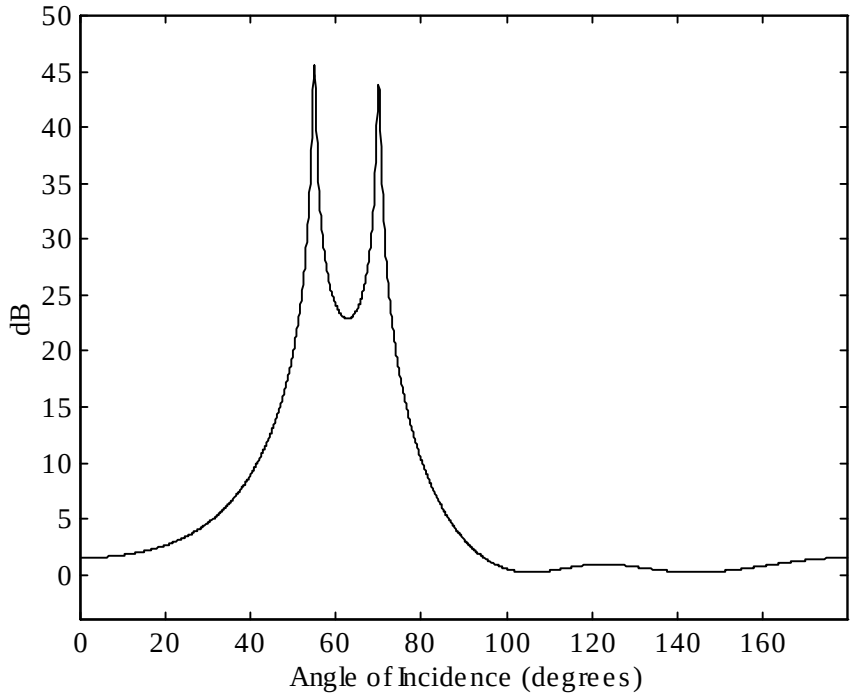

Figure 3.14 $P_{MUSIC}(\theta)$ with sine sources at 55° and 70°

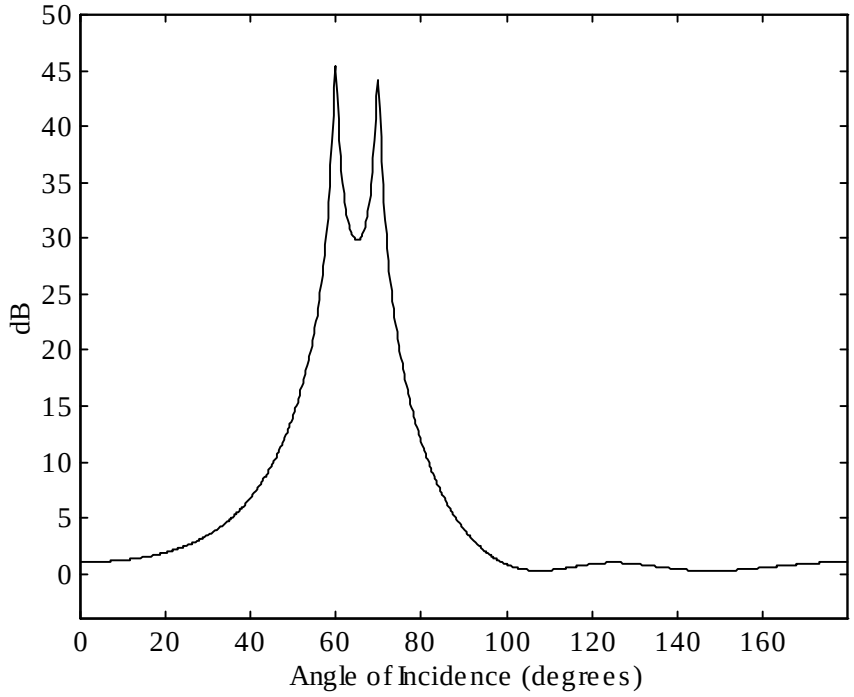

Figure 3.15 $P_{MUSIC}(\theta)$ with sine sources at 60° and 70°

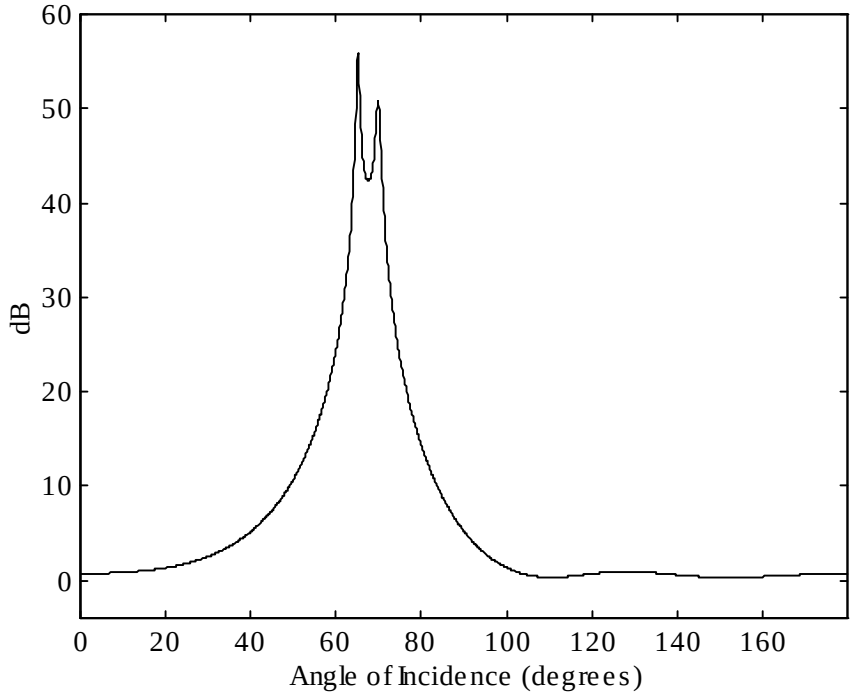

Figure 3.16 $P_{MUSIC}(\theta)$ with sine sources at 65° and 70°

quantization constraint, it is best to increase the number of snapshots used to estimate the sample covariance matrix.

The MUSIC algorithm applied to three sine sources is shown in Figure 3.17. The three sources at 50°, 80°, and 140° were estimated to exist at 50.5°, 80.3°, and 140.1°. The source frequencies were 996, 1000, and 1004 Hz, respectively. In the two source case it was shown that the angles matched (with .1° precision) the actual values with the sources at 996 and 1000 Hz. Therefore, the algorithm is more accurate with a smaller number of sources. We can have at most 3 sources for which we want to find the actual directions in a 4-microphone scenario, because if we wanted to find more sources we wouldn't have a noise subspace in **R** (2.9) to compute the arrival angles from.

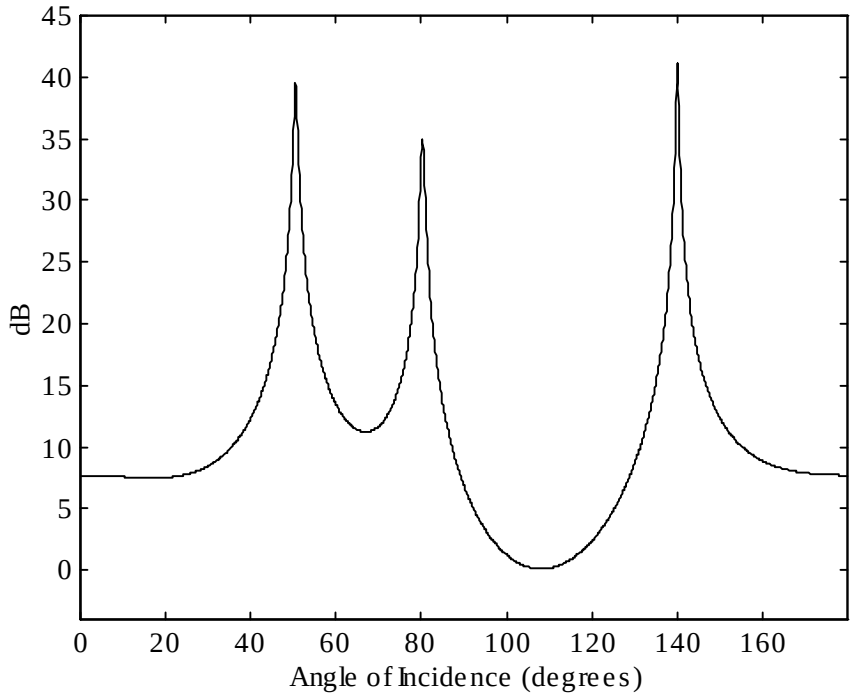

Figure 3.17 $P_{MUSIC}(\theta)$ with sine sources at 50°, 80°, and 140°

3.6 Multiple Narrowband Sources with Noise

This section deals with the noise case for multiple narrowband sources. First we will look at angle estimation for two different narrowband sources at different SNR's. We will use 500 snapshots of two sinusoids at 996 and 1000 Hz at a sample frequency of 8 kHz. One source will be fixed at 120° (estimate for actual driver location in a car array) and the other source will vary. The wavelength λ corresponds to the average of the two frequencies for all scenarios in this section. The results are shown in Table 3.5.

Table 3.5 DOA results for 2 narrowband sources, spatially and temporally white noise, and 500 snapshots

Actual DOA	10,120	40,120	70,120	100,120	130,120	160,120
SNR=10 dB	10.5,120.2	39.9,120.1	69.9,120.0	99.1,121.3	126.0,119.1	120.6,159.0
SNR=0 dB	14.8,118.9	40.0,118.9	71.5,122.9	111.4	124.4	121.7,166.4
SNR=-5 dB	0,159.9	32.6,118.3	71.0,114.4	94.2,128.0	124.1	133
SNR=-10 dB	27.5	79.2	76.5,116.7	128.0	124.8	153.3

The results are pretty good for a 10 dB SNR. At lower SNR's, the MUSIC algorithm has difficulty separating closely spaced sources. At 0 dB SNR, even a 20° difference causes ambiguity. Reasonably decent performance regions get smaller and smaller as SNR decreases, and the algorithm is useless at -10 dB. But there are some things we can do about this. The first thing we can do is to increase the number of snapshots used to generate the sample correlation matrix that the MUSIC algorithm uses. This gives us a better estimate of **R**. The results for the same test performed with 5000 snapshots is shown in Table 3.6. These results are much better than for the 500 snapshot case. Even at a -10 dB SNR, the performance is quite good, except for not being able to resolve two sources 10° apart. A typical result for the -5 dB SNR case for sources at 130° and 120° is

Table 3.6 DOA results for 2 narrowband sources, spatially and temporally white noise, and 5000 snapshots

Actual DOA	10,120	40,120	70,120	100,120	130,120	160,120
SNR=10 dB	10.5,120.1	40.1,120.1	70.0,120.1	100.0,120.0	120.0,129.6	120.0,159.8
SNR=0 dB	11.7,120.2	39.9,120.1	70.2,120.4	99.9,120.4	120.9,129.7	120.4,159.4
SNR=-5 dB	12.4,120.2	40.2,120.0	70.0,119.9	100.0,120.7	120.7,130.4	119.8,159.4
SNR=-10 dB	13.3,119.6	42.7,119.4	72.1,119.5	99.4,121.9	125.3	120.3,157.9

shown in Figure 3.18. From the figure it is seen that there could be a second source around 130°. This resolution problem seems to be more likely for the narrowband case with a low SNR, which corrupts the signal enough to make the two frequencies indistinguishable. This situation won't be encountered in the car array environment.

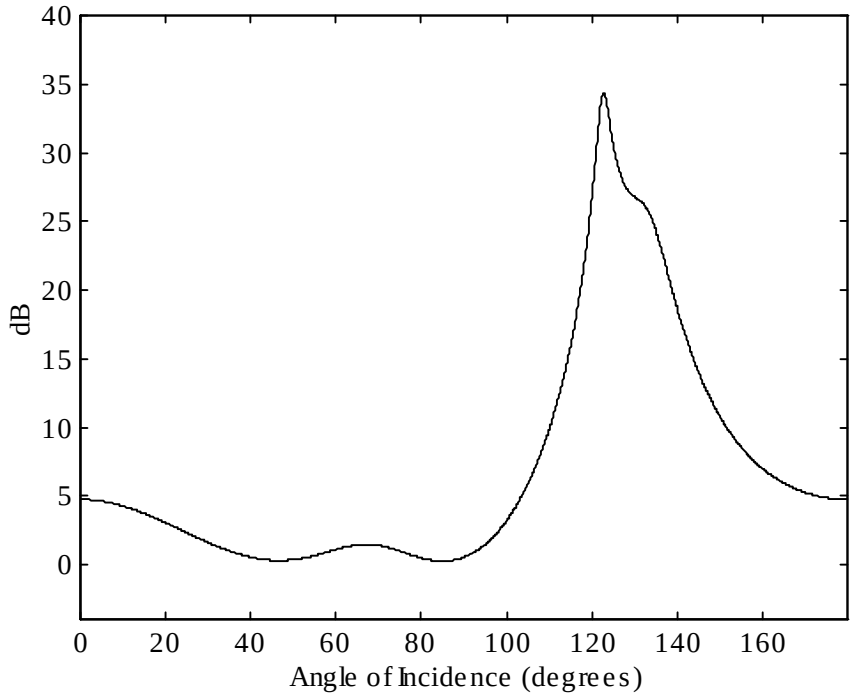

Figure 3.18 $P_{MUSIC}(\theta)$ with sine sources at 120° and 130°, -5 dB SNR

In addition to using more snapshots to generate the covariance matrix, we could also use algorithms besides MUSIC to estimate DOA. Two newer schemes are ESPRIT [16] and GEESE [17]. These techniques utilize the rotational invariance among signal subspaces from array sensor subsets to estimate DOA. These techniques could be tested if the MUSIC algorithm does not perform well in the car environment.

3.7 Single Speech Source

The single speech source case is similar to the single sine source case except that we are dealing with a wide range of frequencies instead of a single frequency. In this scenario we need to closely match the center frequency of the speech source with the parameter l as in (2.12). Because of this, we need to subdivide the input into different frequency bands. We can subdivide the input as much as we want easily if we are using a computer program such as MATLAB, but it won't be as easy to do using a microcontroller. Another problem is that since we get the best array response when the interelement spacing is $\lambda/2$, it would be nice to use different interelement spacings for different frequency ranges. This is going to cause us some problems, since our data acquisition board can only take in 4 microphone inputs. Therefore, we have to somehow optimize the microphone spacings for all scenarios, since in the end these will have to be fixed. For these tests, the interelement spacing was chosen to be 0.11 m. This gives a normalized (look direction = 90°) array response as shown in Figure 3.19 for 6 subbands. From Figure 3.19 you can see that for lower frequencies there is little discrimination with frequency ranges, and for high frequencies grating lobes appear. For these graphs, the response for (0°, 180°) matches the response for (0°, -180°) due to array symmetry (i.e.,

the array can't distinguish whether the source is at some angle ϕ or $-\phi$). The choice of using 6 subbands needs to be explained also.

There are different reasons as to why 6 subbands will be used for the rest of this thesis. Looking at Table 2.1, the frequency ratio for each of the 6 subbands f_{hi}/f_{lo}= 0.7. Now looking at Table 3.4, you can see that when the 60° source is at 700 Hz (f_{hi}/f_{lo}= 0.7 also), the angle estimates are off by 6° and 14.5°. This is somewhat of a different case

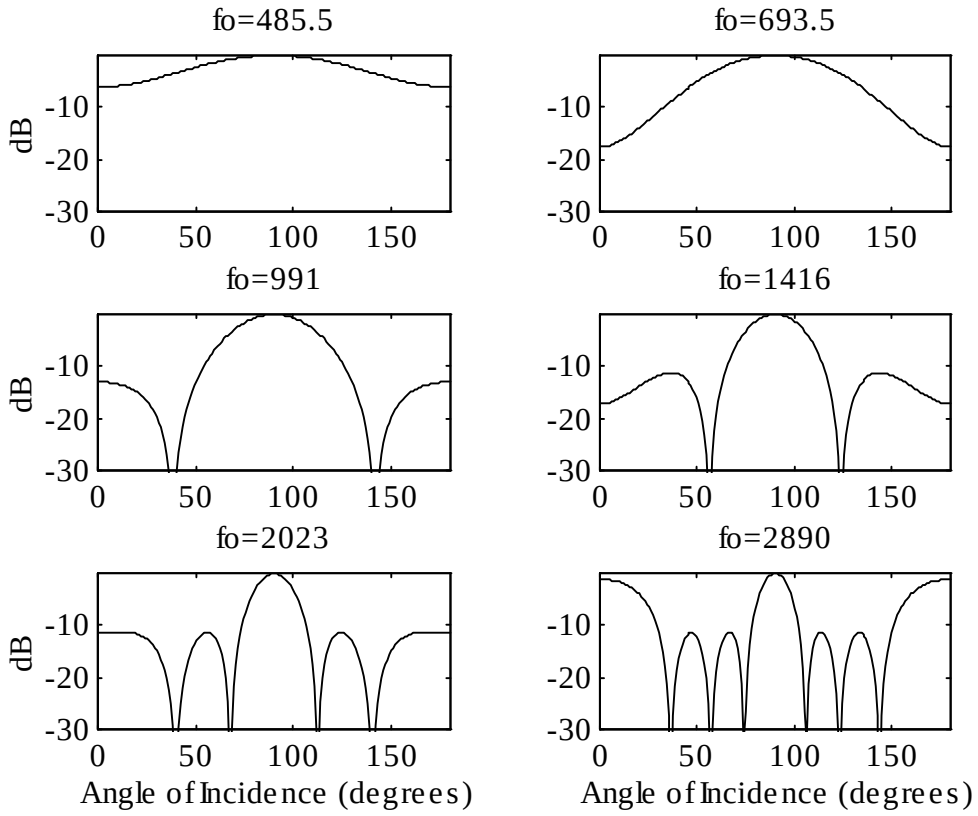

Figure 3.19 Normalized array beampattern for *d* = 0.11 m, 6 subbands

than in the broadband situation, since if we had two sources they would be occupying a wide frequency range, but it does represent a worst-case scenario, with two sources. So it should seem reasonable that we will get good estimates for the one source and even the two-source cases. Another thing that we would like to look at is the beampatterns for

each of the frequency ranges at f_{hi} and f_{lo}. These will be different since λ in (2.12) will be fixed to the center of each of the frequency ranges, and we would like to see the maximum deviation from the beampattern that we will assume and be using to derive optimum performance of the array. Figure 3.20 shows the beampatterns for frequencies f_{hi}, f_{lo}, and f_0 for each of the frequency ranges. From this figure it is seen that the major beampattern differences occur at angles closer to 0° and 180°. In the middle range (40°-

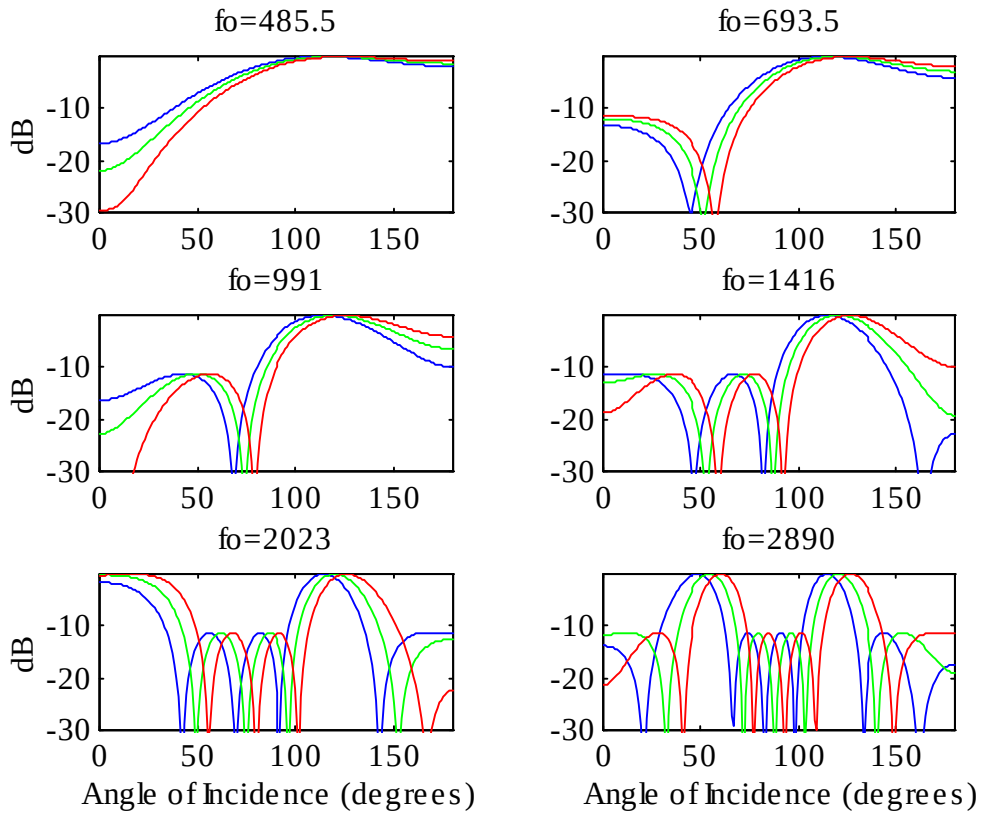

Figure 3.20 Beampatterns for f_{lo} (blue), f_0 (green), and f_{hi} (red) for 6 subbands

140°), the beampattern peaks and nulls are at most off from each other by 14°. This is the worst case scenario for actual data though, and once again it will be highly unlikely that most of the energy from the speech source(s) will be at f_{hi} and f_{lo} of any band.

DOA results for 1 speech source with the subband decomposition as described is shown in Table 3.7. Results are mirrored for the region [90°, 180°]. Since the microphone delays were simulated, we needed to sample the speech as fast as possible. The speech was sampled at 40 kHz. Though ideally we would want a much higher sampling rate, the results wouldn't be much better, because we would still have broadband effects in the subbands. It is seen that the DOA estimate gets better as the DOA gets closer to 90°. The latter can be attributed to more distinguishable time delay

Table 3.7 DOA results for 1 speech source, 1000 snapshots

Actual DOA	0	10	20	30	40	50	60	70	80	90
MUSIC DOA	16.3	17.0	21.6	36.5	42.4	50.9	60.9	70.5	80.0	90.0

differences between equally spaced angles at angles closer to 90°. This is shown in Table 3.8, using the microphone arrangement as in Figure 2.3. This trend can also be observed graphically in Figure 2.4. The distance delays for microphones 2, 3, and 4 are

Table 3.8 Relative distance delays from microphone 1 at different DOA's

DOA	0	5	10	20	30	40	50	60	70	80
$d_2\cos\theta$ (cm)	11.0	11.0	10.8	10.3	9.5	8.4	7.1	5.5	3.8	1.9
$d_3\cos\theta$ (cm)	22.0	21.9	21.7	20.6	19.1	16.9	14.1	11.0	7.5	3.8
$d_4\cos\theta$ (cm)	33.0	32.9	32.5	31.0	28.6	25.3	21.2	16.5	11.3	5.7

$d_2\cos\theta$, $d_3\cos\theta$, and $d_4\cos\theta$, where θ is the DOA angle, as defined in Section 2.1. The distance delay is the extra distance the assumed planar sound wave travels to reach each microphone compared with the distance to microphone 1, with the distance to microphone 1 being the shortest for $\theta < 90°$. From Table 3.8, we can see that the

distance delays from 0° to 10° are almost indistinguishable, percentage-wise. We can generate these delays pretty well using a high sample rate and interpolation, but practically we won't be able to get good results at low and high DOA angles because our subbands need to be extremely closely-spaced to counteract the resolution problem.

Although ideally we would like to use many more subbands, using 6 subbands is still preferred. The subbands are close enough to account for inaccuracies due to frequency components close to subband edges, while still being able to give a good average estimate (since there are many signal frequency components in each subband). The computational complexity associated with increasing the number of subbands is not justified by the improvement in the DOA estimate. The results in Table 3.7 were generated by averaging the single largest peak in each of the first four subbands (because aliasing occurs in the estimate in the last two subbands), with two additional constraints. One constraint throws out a peak if the averaged DOA estimate with the peak is more than 3° off from the averaged DOA estimate without the peak. The second constraint allows the use of peaks that are at 0° or 180°, as long as the other peaks are less than 40° or greater than 140°, respectively. This helps out the estimate at high and lower DOA's, since the estimate from MUSIC tends to produce results biased towards 90°. The estimate in Table 3.7 can be improved upon by using covariance matrix updating, as will be seen in Section 6.2.

The ideal place to put the desired speaker is at a location where the DOA estimates (for planar waves) are fairly good, and the near-field effects are minimal. Near field effects are more prominent closer to 90° and less prominent towards 0° and 180°. In

a car situation, the speaker is most likely the driver, and with the array in the dashboard the driver would be between 100° and 150°, which is a good performance region.

3.8 Single Speech Source with Noise

With a speech source now, we would like to model as closely as possible the noise environment. We discovered in Section 3.2 that car noise contains a directional component and a diffuse component. Therefore we will evaluate the MUSIC algorithm at different SNR's for two cases using actual noise data from a car: 1) uncorrelated at the 4 microphones, and 2) 10 different noise data samples at 10 random arrival angles. The input was split up into frequency bands as in Section 3.7. Results for the uncorrelated noise case are shown in Table 3.9. It was assumed that we knew there was 1 source for the MUSIC algorithm. Results for the correlated noise case are shown in Table 3.10.

Table 3.9 DOA results for 1 speech source, uncorrelated car noise, 1000 snapshots

Actual DOA	10	40	70	100	130	160
SNR=10 dB	17.5	43.1	72.6	100.5	127.6	157.9
SNR=0 dB	10.3	41.5	74.5	101.6	128.5	156.5
SNR=-5 dB	10.5	35.0	74.5	101.4	117.0	161.7
SNR=-10 dB	75.5	76.8	91.3	101.7	102.0	99.8

Table 3.10 DOA results for 1 speech source, correlated car noise, 1000 snapshots

Actual DOA	10	40	70	100	130	160
SNR=10 dB	9.0	41.7	68.7	99.8	129.1	158.3
SNR=0 dB	16.3	38.5	74.0	99.0	129.8	155.5
SNR=-5 dB	24.8	47.5	63.5	106.2	131.1	140.6
SNR=-10 dB	127.0	60.5	76.3	119.5	129.0	100.1

At first glance, the estimates for the correlated noise case seem to be worse than those for the uncorrelated case. The variances in the estimates in the subband regions, though, appear to be similar. The estimates are still fairly good at a -5 dB SNR, which is probably close to the worst-case scenario. The results at -10 dB, though, are almost useless. The estimates would be much worse if exact replicas of the noise sources appeared at each of the inputs at delays of integer multiples of the sampling frequency. This case never appears in practice, but must be carefully avoided when doing simulations (can be achieved by sampling noise at a much higher rate than used for the data given to the MUSIC algorithm).

3.9 Multiple Speech Sources

In the multiple speech source problem we would like to see how well the MUSIC algorithm pinpoints multiple speech sources. Here we will use two speech sources with no noise; we will not look at the 3-speech source case since this case would be very unlikely to occur. An example of the DOA results for two speech sources with equal gain, using 400 snapshots (f_s = 8 kHz) to estimate the covariance matrix, is shown in Figure 3.21. With no noise, the estimation peaks are extremely sharp, making it very easy to pick out the peaks. Also, as seen previously, the results for the last two frequency ranges are useless for peak-picking purposes. The peaks from the first four frequency ranges are averaged to give the final DOA results. The estimations for these sources are 80.0° and 129.4°.

We would like to examine DOA estimation results over a wide range of arrival angles. Some results, using 400 snapshots to estimate the covariance matrix, are shown

in Table 3.11. Results are very accurate for most angles. Estimates at low and high angles are off due to the inherent resolution problems (Section 3.7) coupled with the fact that creating proper time delays for the data at 8 kHz requires infinite resolution of the speech data to be time delayed (the speech here was digitized at 40 kHz).

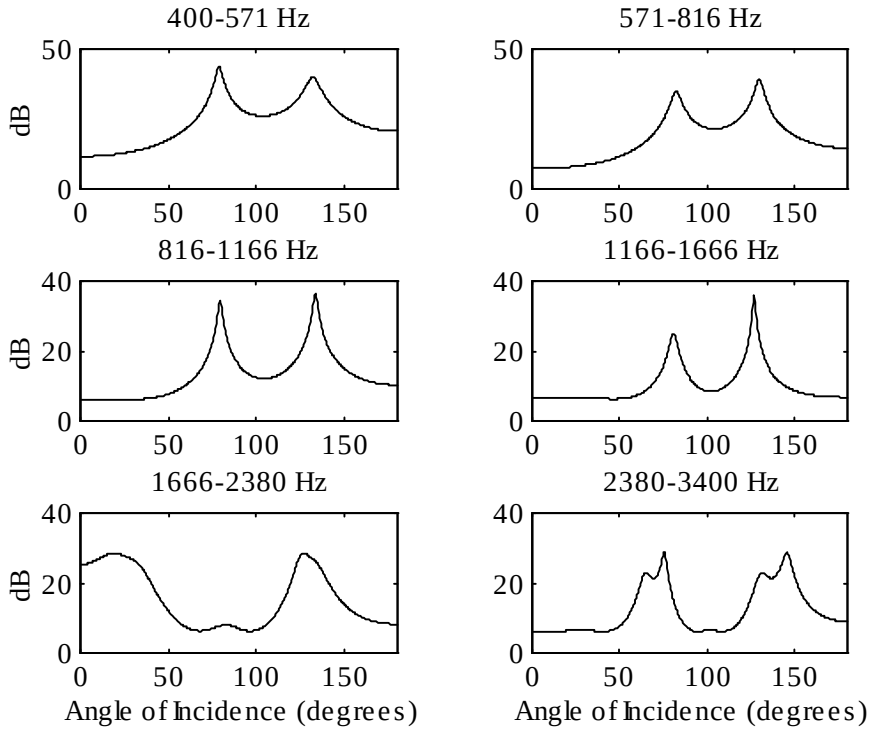

Figure 3.21 $P_{MUSIC}(\theta)$ for equal-gain sources at 80° and 130°

Table 3.11 MUSIC DOA results, 2 speech sources, 400 snapshots

Actual DOA	10,40	40,70	40,100	40,130	40,160
MUSIC DOA	13.0, 44.0	42.9, 70.4	42.4, 100.0	41.8, 127.5	41.3, 162

Actual DOA	20,80	50,80	80,100	80,130	80,160
MUSIC DOA	22.1, 77.6	49.8, 78.9	80.0, 100.1	80.0, 129.4	80.7, 156.6

We would like to examine the effect that increasing the number of snapshots for the covariance matrix has on distinguishing two closely-spaced peaks. Figure 3.22 shows results for two sources, at 40° and 60°, with varying numbers of snapshots. There is almost no difference between using 2000 and 10000 snapshots, though using these

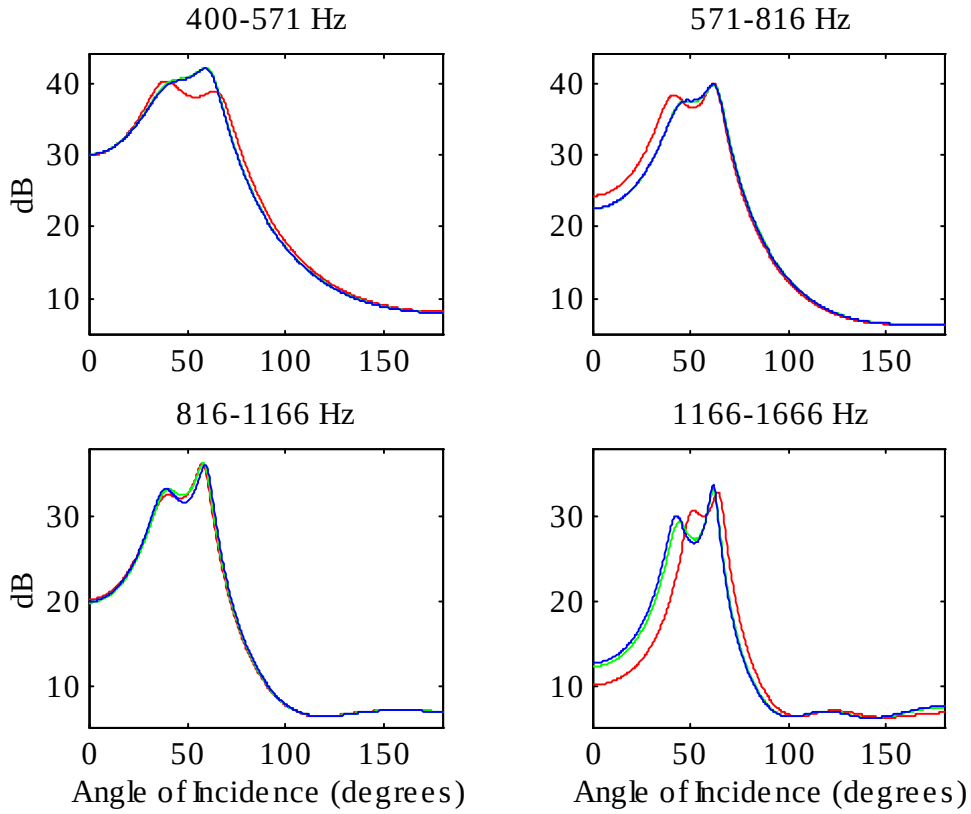

Figure 3.22 $P_{MUSIC}(\theta)$ for sources at 40° and 60°, using 400 (red), 2000 (blue), and 10000 (green) snapshots

numbers of snapshots does improve the DOA estimate, especially with better results from the 1166-1666 Hz range. Aside from helping somewhat to resolve closely-spaced peaks, though, there is little advantage in using more than 400 snapshots to improve covariance matrix estimation for this subband scenario and in the noise-free case.

3.10 Multiple Speech Sources with Noise

With multiple speech sources and noise, the problem of DOA estimation with 4 microphones becomes much more complex. Whereas without noise we could just pick the two peaks, with the addition of noise we have to be more concerned with which peak goes with which source, and maybe which peaks don't correspond to any sources at all.

We will first examine the case where the noise is 10 dB below each speech source, with the speech sources having equal gain. The noise sources are somewhat evenly spaced from 0-180°. Peaks from the first 4 subbands were picked as peaks if $P_{MUSIC}(\theta)$ for the peak at θ was greater than 1/4 of the way up (in dB) from the lowest $P_{MUSIC}(\theta)$ to the highest $P_{MUSIC}(\theta)$ in each subband. This helps eliminate small peaks due to noise. Also, when the peaks are somewhat close, there may only be 1 peak instead of 2, so subband regions which only contain 1 peak will have that estimate thrown out, and the other estimates will be averaged to determine the peak location. These ad hoc restrictions helped obtain reasonable results from the given MUSIC estimates for the 10 dB scenario. Results for different DOA angles are shown in Table 3.12. The estimates

Table 3.12 MUSIC DOA results, 2 speech sources, 10 dB SNR

Actual DOA	10,40	40,70	40,100	40,130	40,160
400 snapshots	22.3, 65.5	43.5, 77.2	44.6, 102.6	44.0, 131.6	44.5, 162.4
2000 snapshots	36.8, 117.5	51.5, 84.3	44.9, 102.4	43.6, 131.5	43.4, 157.9
20000 snapshots	36.3, 112.6	47.4, 80.1	43.9, 102.1	42.6, 130.4	42.4, 156.7

Actual DOA	20,80	50,80	80,100	80,130	80,160
400 snapshots	25.5, 79.1	54.4, 81.4	81.1, 104.1	79.8, 130.9	80.1, 160.2
2000 snapshots	26.0, 81.1	54.3, 85.0	82.0, 106.9	79.1, 131.1	80.0, 158.0
20000 snapshots	27.0, 81.0	54.6, 84.8	82.3, 106.9	79.9, 130.5	79.9, 156.3

for [10°, 40°] and [40°, 70°] are more off than the rest, especially at [10°, 40°] where the sources have merged into a single peak. Otherwise, the estimates are all within 7°. A significant trend shown by Table 3.12 is that the estimate has little improvement when increasing the number of snapshots used for generating the sample covariance matrix.

At low SNR's, we can expect it to be very difficult to resolve closely spaced sources. Also, we can expect the noise sources to bias the results, especially if the noise is directional. This requires additional algorithms to differentiate between speech sources and noise sources. Since our final implementation will not be general enough to warrant a thorough review of resolving multiple speech sources, we will only look at how well the aforementioned peak-detection algorithm works at low SNR's.

Table 3.13 and Table 3.14 show results at different DOA's for the 0 dB and -5 dB SNR cases, respectively. Even in the 0 dB case, we cannot distinguish between a source at 20° and 80°. MUSIC can distinguish between sources easier if they are on opposite sides of 90°. This is because the relative time delays of the sources to each sensor compared to the reference sensor (sensor 1) are negative for one source and positive for

Table 3.13 MUSIC DOA results, 2 speech sources, 0 dB SNR

Actual DOA	20,80	50,80	80,100	80,130	80,160
400 snapshots	36.8, 90.5	66.5, 119.8	84.0, 125.3	78.5, 134.6	78.9, 156.4
2000 snapshots	51.3, 117.0	63.4, 123.8	84.2, 132.7	79.3, 135.0	79.4, 151.3

Table 3.14 MUSIC DOA results, 2 speech sources, -5 dB SNR

Actual DOA	20,80	50,80	80,100	80,130	80,160
400 snapshots	56.4, 136.6	63.0, 139.2	88.8, 142.4	77.8, 141.0	79.2, 151.8
2000 snapshots	55.6, 140.3	64.5, 141.8	87.5, 141.0	78.7, 138.5	78.8, 144.3

the other. Considering that the car environment has low SNR's, it would be tough to resolve two closely-spaced sources on the same side of 90°. If the two sources are on opposite sides of 90°, these results show that we can obtain the DOA fairly well.

The results in Table 3.13 and Table 3.14 are derived from an algorithm mentioned previously, and it would be useful to compare these MUSIC DOA results with those pertaining to the noiseless case (Figure 3.21). Figure 3.23 and Figure 3.24 show results with a similar speech environment as Figure 3.21 (400 snapshots) at 0 dB and -5 dB SNR's, respectively. In the 0 dB case all of the peaks in the first four subbands are less sharp than they were in the noiseless case. The 80° peak in the 1166-1666 Hz range is getting flat, and at -5 dB SNR that peak doesn't exist. In the -5 dB case, the ad hoc restrictions apply, and the data for this subband is thrown out, and the results from first 3 subbands were averaged to obtain the estimates.

The performance of MUSIC in the 2-speech source scenario gives us guidelines as to what can be accomplished in the car environment using a 4-microphone array with MUSIC. If we have the driver and the front seat passenger speaking at the same time, we will be able to find a reasonable DOA for both people. This holds too for the driver and the right rear seat passenger. If we have the driver and left rear seat passenger, or the right seat passengers, speaking at the same time, we probably wouldn't be able to distinguish between the two. We may be able to differentiate in a high SNR scenario, but overall it will be tough to suppress either speaker.

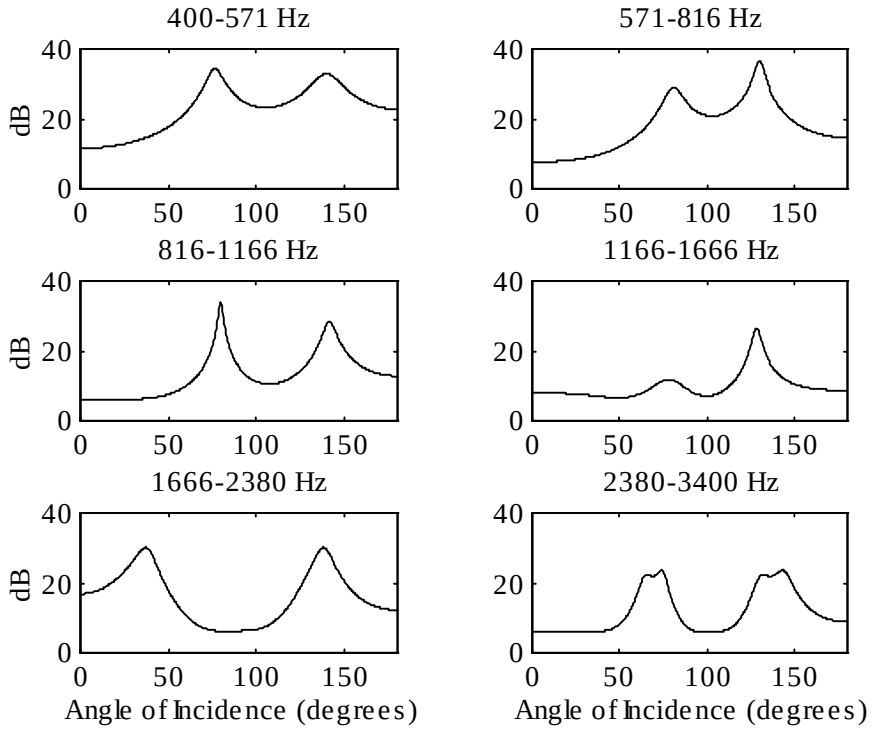

Figure 3.23 $P_{MUSIC}(\theta)$ for equal-gain sources at 80° and 130°, 0 dB SNR

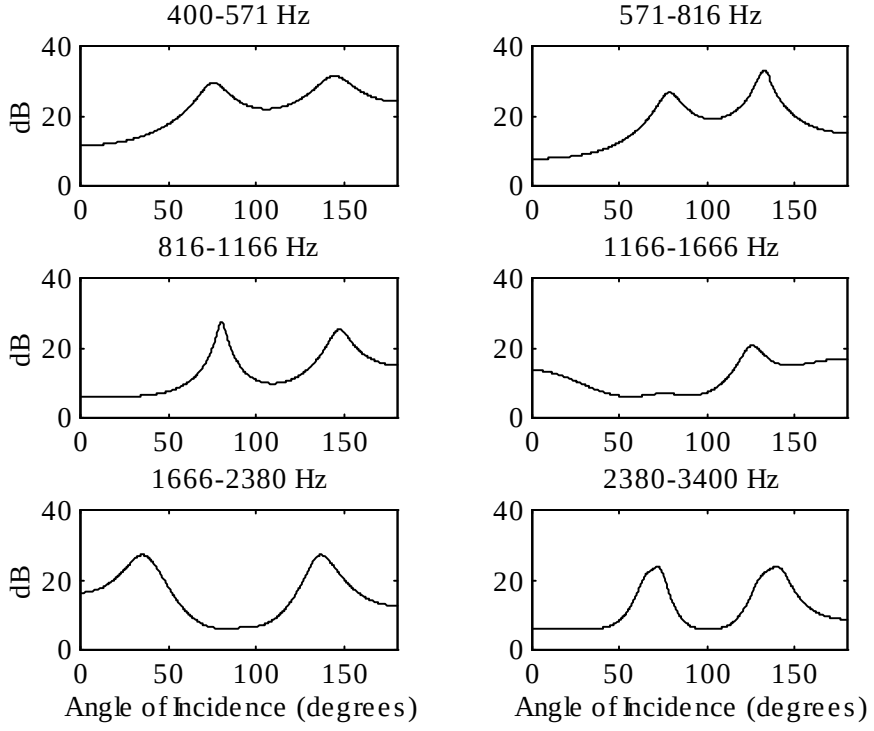

Figure 3.24 $P_{MUSIC}(\theta)$ for equal-gain sources at 80° and 130°, -5 dB SNR

Chapter 4

Simulating Beamformer Output

4.1 Introduction

In the previous section we used the MUSIC algorithm to generate our DOA estimate. Now with this estimate, we want to produce an output with the highest possible SNR. We will go through cases of increasing difficulty, and on the way we will determine which methods are best to use.

4.2 Multiple Narrowband Sources

Once we have found which directions the sources are coming from, we need to generate an optimum output. If we are interested in one source only, then we want the signal to noise plus interference ratio (SNIR) maximized. This is the same as saying that we want the array output power minimized while keeping the gain in the look direction (θ) equal to 1. From (2.8), the output of the array can be rewritten as

$$y(t) = \mathbf{w}^H \mathbf{x}(t). \tag{4.1}$$

Therefore the average array output power is

$$P = E\left[\left|y(t)\right|^2\right] = \mathbf{w}^H E\left[\mathbf{x}(t)\mathbf{x}^H(t)\right]\mathbf{w} = \mathbf{w}^H \mathbf{R}\mathbf{w}. \tag{4.2}$$

So our problem is that we want to minimize (4.2) while making the array gain $\mathbf{w}^H\mathbf{a}(\theta)$ equal to 1, with $\mathbf{a}(\theta)$ as in (2.12). The solution to this problem is

$$\mathbf{w} = \frac{\mathbf{R}^{-1}\mathbf{a}(\theta)}{\mathbf{a}^{H}(\theta)\mathbf{R}^{-1}\mathbf{a}(\theta)}. \tag{4.3}$$

This weight vector is referred to as a *linearly constrained minimum variance filter* (LCMVF). Once we find the source arrival angle from the MUSIC algorithm, we can compute the optimum weight vector and generate the output.

Though using (4.3) gives theoretically an optimum output, we would like to set up the problem statement of generating the optimal output with a desired narrowband source and an interfering narrowband source. For an example, we will use two sinusoids, at 398 and 400 Hz; these frequencies are close enough to make the broadband effects negligible in our observations. Figure 4.1 shows the array beampattern of the 4-microphone array steered to 70°. The nulls are centered at 32.7°, 99.1°, and 131.2°.

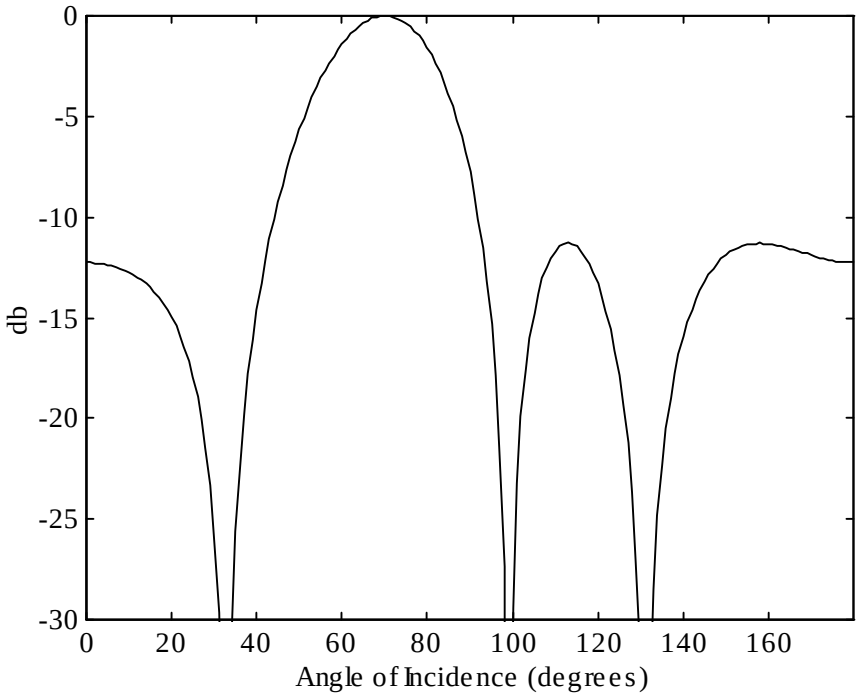

Figure 4.1 Array beampattern steered to 70°

Table 4.1 gives us some numerical measures of the width of the sidelobe nulls. From Table 4.1 it is seen that the region around the 32.7° and 131.2° nulls behaves very similarly, while the region around the 99.1° null varies in dB much faster.

Table 4.1 Beamwidths around nulls of array beampattern in Figure 4.1

	-25 dB beamwidth	-20 dB beamwidth	-15 dB beamwidth
32.7°	29.7°- 35.2° (5.5°)	27.0°- 36.9° (9.9°)	20.1°- 39.7° (19.6°)
99.1°	97.7°- 100.6° (2.9°)	96.6°- 102.0° (5.4°)	94.9°- 104.8° (19.9°)
131.2°	128.5°- 134.0° (5.5°)	126.4°- 136.3° (9.9°)	122.4°- 141.4° (20.0°)

Looking again at Figure 4.1, our goal in a single desired and interfering source scenario should be to maximize the gain ratio of the desired source to the interfering source. At first glance, it would seem optimal to steer one of the nulls onto the interfering source. This gives us the optimum solution theoretically, but we still need to decide which null gives us the best solution. In a narrowband scenario, the maximum possible attenuation is mainly dependent on the quantization level. Assuming that the spacing between microphone elements is $\lambda/2$ (we have been assuming this for the narrowband scenario), the equation which represents the gain of the beamformer at angle θ with steered direction θ_0 is [8]

$$G(\theta) = \left[\frac{\sin(\pi M(\cos(\theta) - \cos(\theta_0))/2)}{M \sin(\pi(\cos(\theta) - \cos(\theta_0))/2)} \right]^2 . \tag{4.4}$$

So once we find the source arrival angles θ_{desired} and $\theta_{\text{interference}}$, we can determine the optimum steering angle θ_0 by maximizing the difference

$$G(\theta_{\text{desired}})_{dB} - G(\theta_{\text{interference}})_{dB}. \tag{4.5}$$

Equation (4.5) represents the signal-to-noise ratio as well as the array gain for 1 signal and 1 noise (interference) source. As an example, our desired source will be at 70° and the interference source will be at 110°, both with equal gains. The plot of (4.5) versus θ is shown in Figure 4.2.

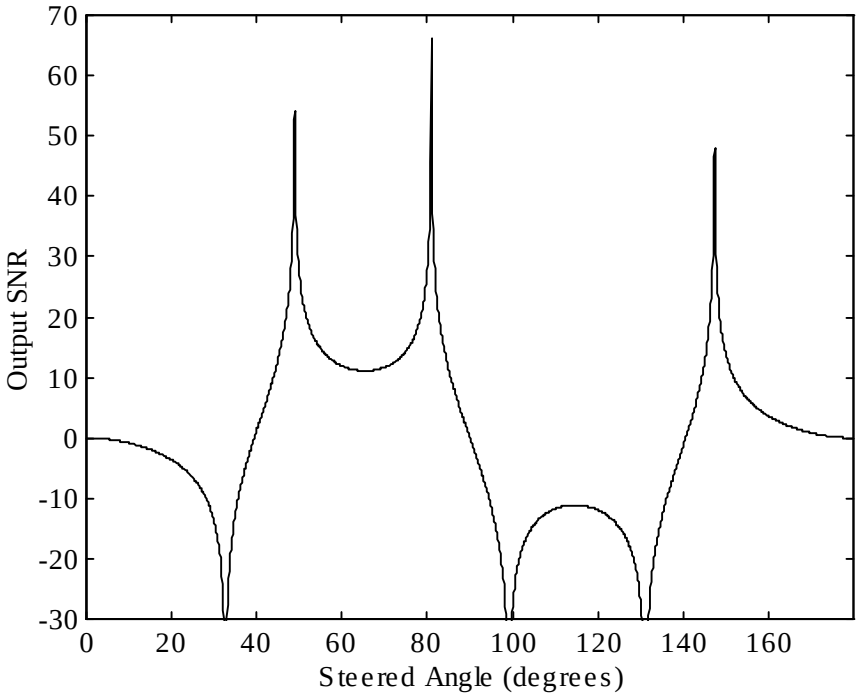

Figure 4.2 Output SNR with $\theta_{\text{desired source}}$=70°, $\theta_{\text{interference source}}$=110°

From Figure 4.2 the highest SNR's occur when the array is steered to 49°, 81°, or 147.5°. A plot of the corresponding array beampatterns when the array is steered to each of these angles is shown in Figure 4.3. Each of these graphs has a null in the beampattern at 110°.

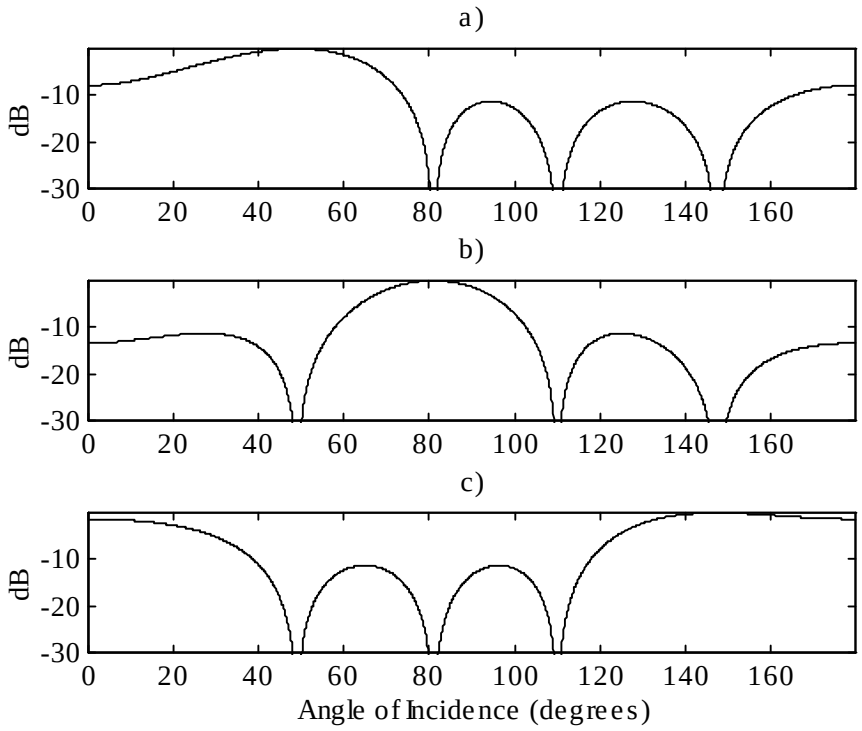

Figure 4.3 Beampatterns with array steered to a) 49°, b) 81°, c) 147.5°

At first glance it would probably be better not to steer the array in the 147.5° direction since the gain at 70° is almost 10 dB smaller than for the other two cases. Analytic solutions to the maximization problem, (4.5), can be derived, but we will evaluate only the beamforming solution in (4.3) and compare it to the simple phase-steering solution (using the desired source arrival angle for the weights in (2.8)).

We want to simulate the performance of the linearly constrained minimum variance filter (LCMVF) in (4.3) and compare it to using the desired source arrival angle for the weights in (2.8). The test will involve two sine waves, at 996 and 1000 Hz, with the common wavelength corresponding to 998 Hz. The 996 Hz wave is the desired source arriving at 70°, while the 1000 Hz wave is the interference source arriving at 110°. Since the frequencies of the sources are so close, 5000 samples were taken to show the

amount of interference rejection. Figure 4.4 shows one of the inputs into the beamformer so we have something to compare the outputs to. Both sources have a gain of 10, so we

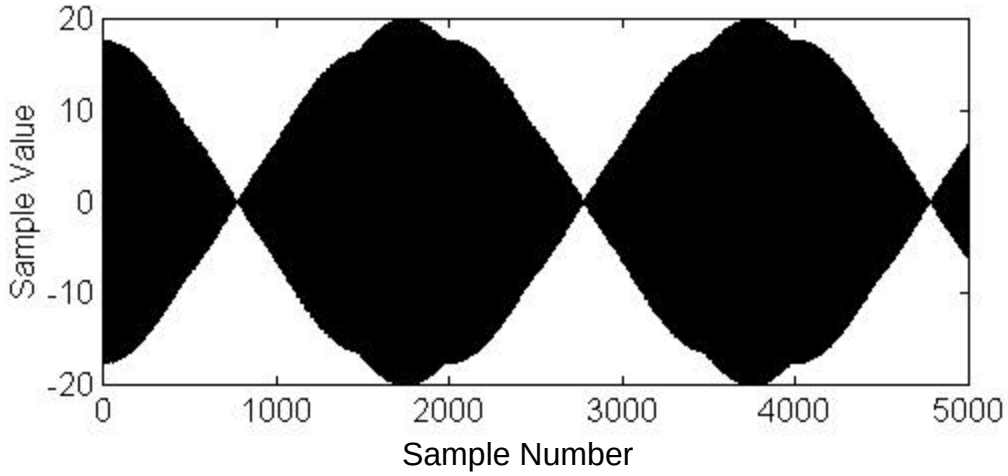

Figure 4.4 One of the beamformer inputs with sources at 70° and 110°

would expect the optimal output to be a 996 Hz sine wave with an amplitude of 10. Figure 4.5 shows the results of the tests. The phase-steered solution provides an amount of rejection which can be determined from Figure 4.2 with θ=70°, which looks to be about 12 dB. The LCMVF output is very noisy. Spatially and temporally white noise at 50 dB below the sources was added to the inputs so that the sample correlation matrix **R** would not be singular (since we need to compute the inverse of **R** in (4.3)). Unfortunately, the output seems to have overemphasized the noise. Looking at the array gain pattern with the LCMVF weights, Figure 4.6, we see that this is indeed the case. From Figure 4.6 we can also see that there is unity gain at 70°, so the LCMVF solution still holds to that restriction. Also, the output is minimized as well, considering that the solution that we want, which is just the 70° source as the output, would require the output

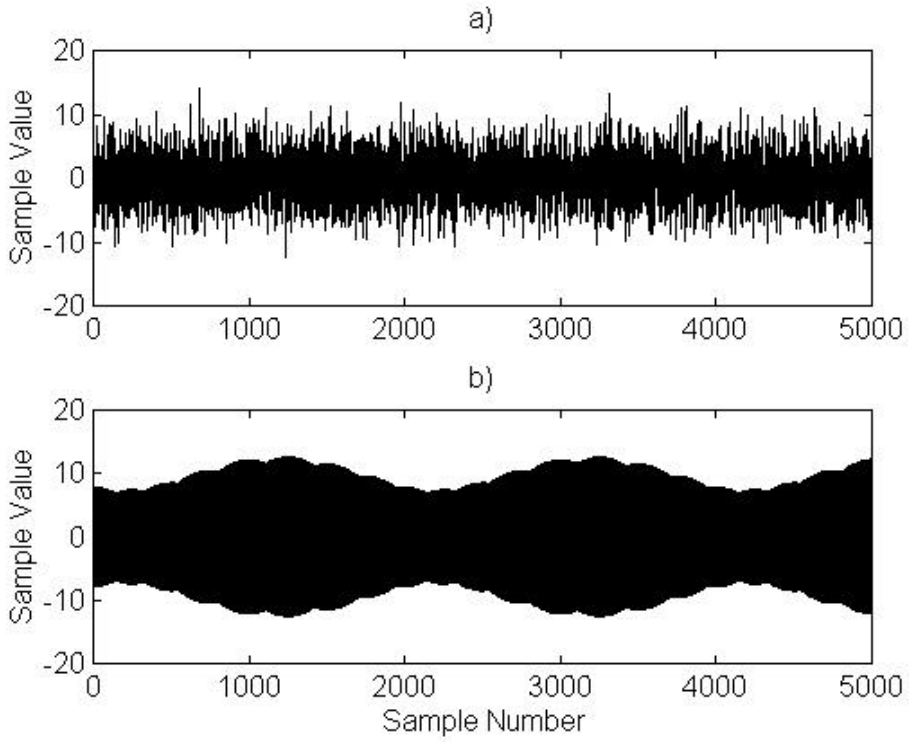

Figure 4.5 Beamformer output for 70° and 110° sources a) using LCMVF b) phase-steering to 70°

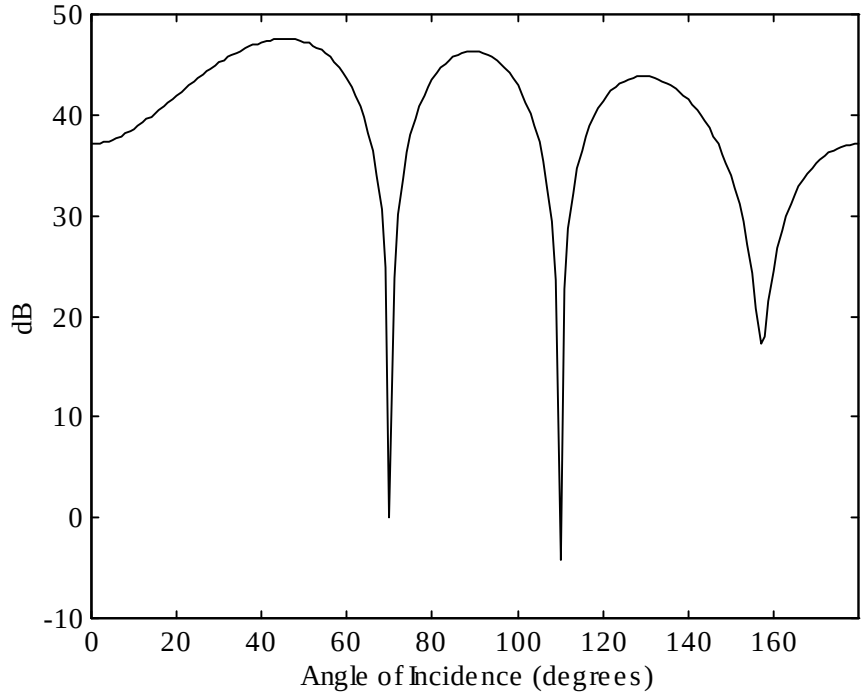

Figure 4.6 Beampattern using LCMVF weights, 50 dB SNR

to have an amplitude of 10, whereas in Figure 4.5 we see that the output has an average amplitude of about 8.5. The gain at 110° is -4 dB, only 4 dB below the 70° source. Even though the algorithm doesn't break down, the performance is definitely poor for the high SNR case.

In the next section, we will examine a case with a lower SNR, and see how well the two algorithms perform. For the high SNR case, the LCMVF solution doesn't work as well as the simple phase-steered solution.

4.3 Multiple Narrowband Sources with Noise

We would like to examine a realistic case of a narrowband source in a somwhat noisy environment. A test was performed, similar to the one in the previous section, except with the spatially and temporally white noise at 20 dB down from each of the signals present at the four inputs. The results are shown in Figure 4.7. The performance of the LCMVF is much better here than when the SNR was 50 dB as in the previous section. The LCMVF gain pattern should support this claim. From the gain pattern in Figure 4.8, the maximum gain at any angle is only 2.5 dB, as compared with 48 dB in the previous section. The 110° source now has -18.2 dB of rejection, which is improved over the phase-steering solution by about 6 dB. Though in some circumstances we may want the interfering source to be rejected more, the LCMVF solution is only concerned about minimizing the output power.

Now that the LCMVF solution has shown promise, its performance will be examined at a much lower SNR. Figure 4.9 shows the outputs using the two methods. The output looks similar to the 20 dB case, Figure 4.7, but with more noise. The LCMVF

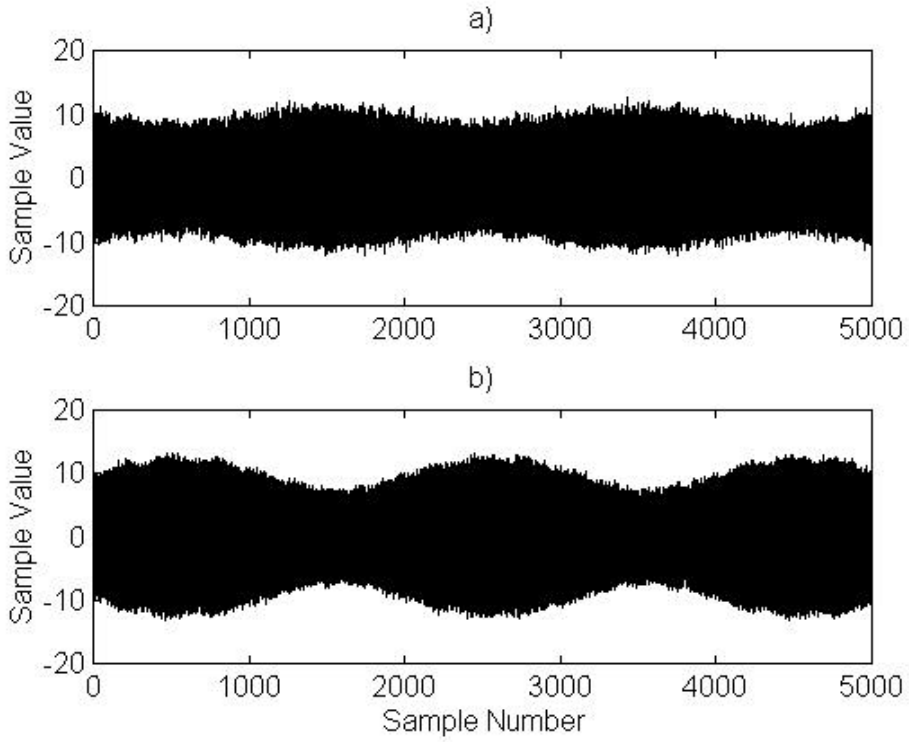

Figure 4.7 Beamformer output for 70° desired and 110° interfering source, white noise 20 dB below sources a) LCMVF b) phase-steering to 70°

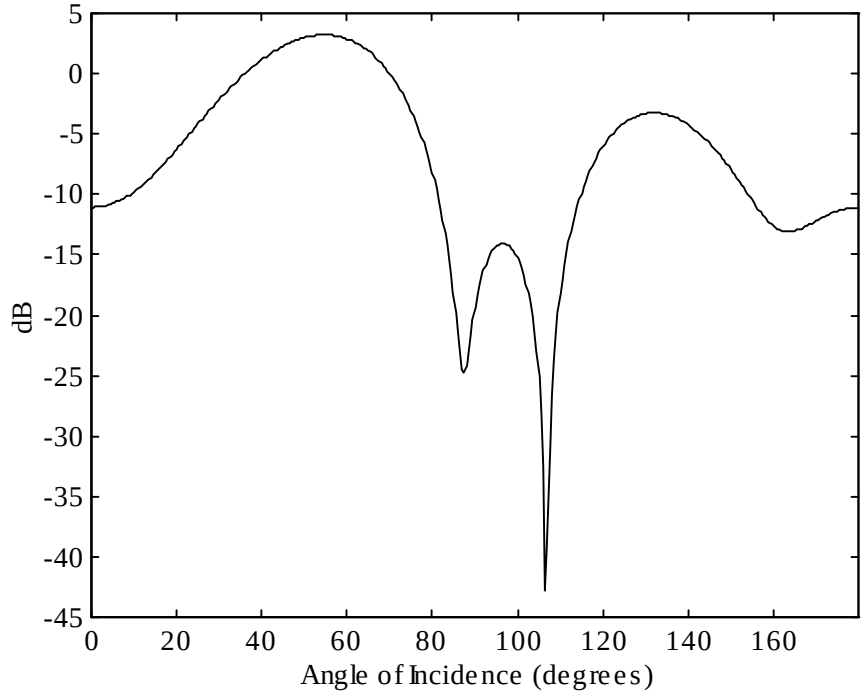

Figure 4.8 Beampattern using LCMVF weights, 20 dB SNR

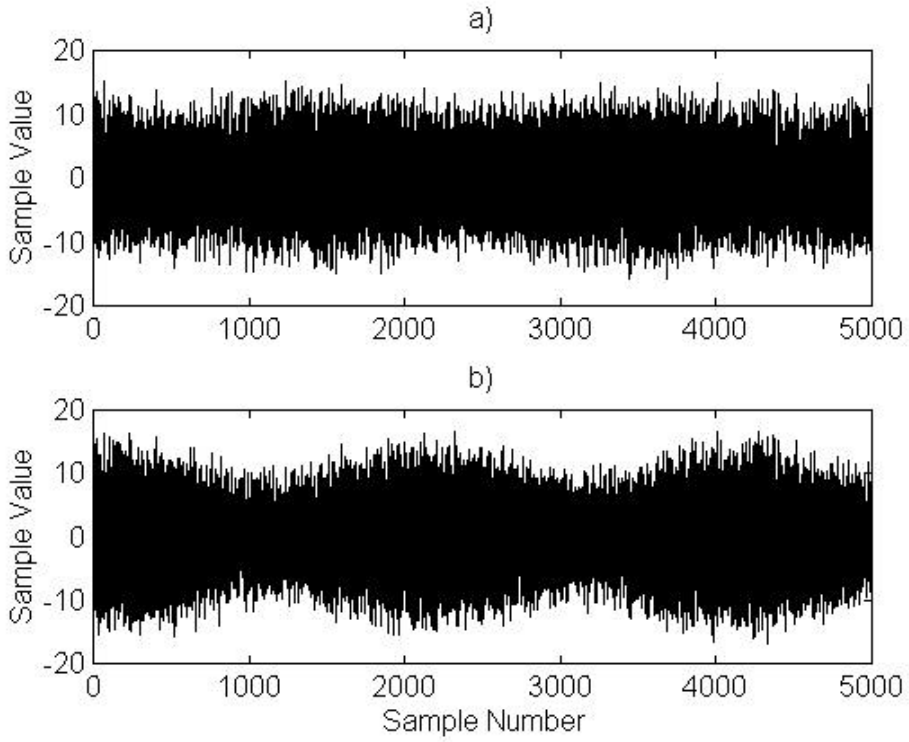

Figure 4.9 Beamformer output for 70° desired and 110° interfering source, white noise 5 dB below sources a) LCMVF, b) phase-steering to 70°

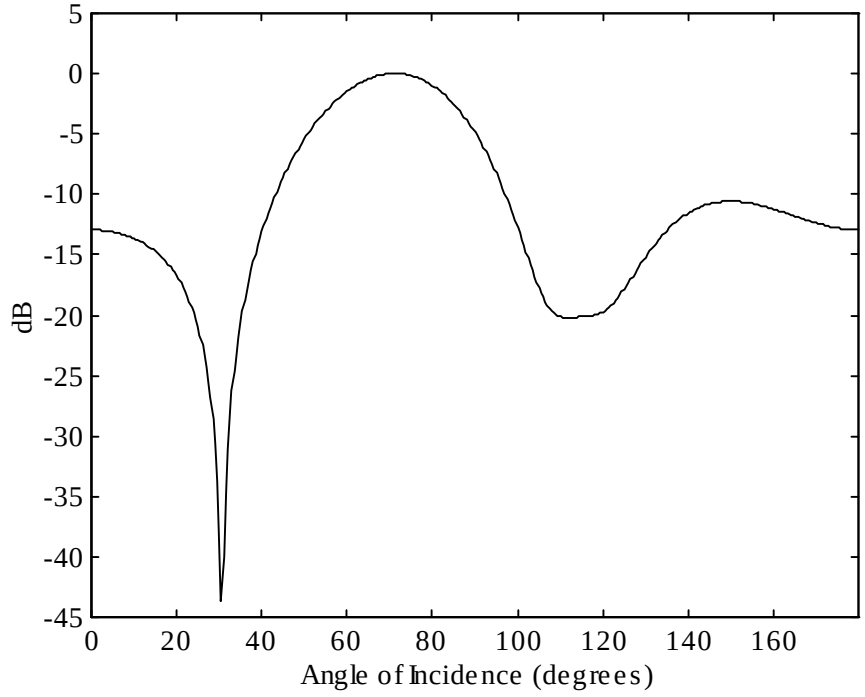

Figure 4.10 Beampattern using LCMVF weights, 5 dB SNR

beampattern is shown in Figure 4.10. It is seen that the center of the mainlobe is nearly at 70°. There is also 20 dB of rejection at 110°. It is useful to note that as the SNR decreases, the center of the mainlobe of the LCMVF beampattern moves closer to the desired source direction. This beampattern is desirable over the phase-steered solution's beampattern, Figure 4.1, for low SNR's. In a high SNR scenario it may be much better to simply steer the interfering source into a null than perform some other optimization method.

The results we have obtained so far were using narrowband assumptions, which included setting the array interelement spacing equal to half of the wavelength of the narrowband source, as well as using the wavelength of the source in computing the $\mathbf{a}(\theta)$ vector in (2.12). For the speech case, we will split the frequency range into 6 bands, as in Section 3.7. Then we will test the performance of the LCMVF and phase-steering methods using broadband inputs and broadband noise.

4.4 Single Speech Source with Noise - using Narrowband Methods

We examine different noise scenarios with one speech source and see what improvement in SNR is obtained using the methods from the previous section. As with the speech simulations in Chapter 3, the speech signal was digitized at 40 kHz to better simulate the delays for the speech data, then downsampled to 8 kHz for the simulations. Also, the covariance matrix estimate was calculated in 400 snapshot increments (1 frame), with no memory between frames. The first case is with the speech source at 135° and a single noise source (taken from the car environment) at 70°. Results are shown in Figure 4.11. The phase-steered output looks to have restored the speech better than the

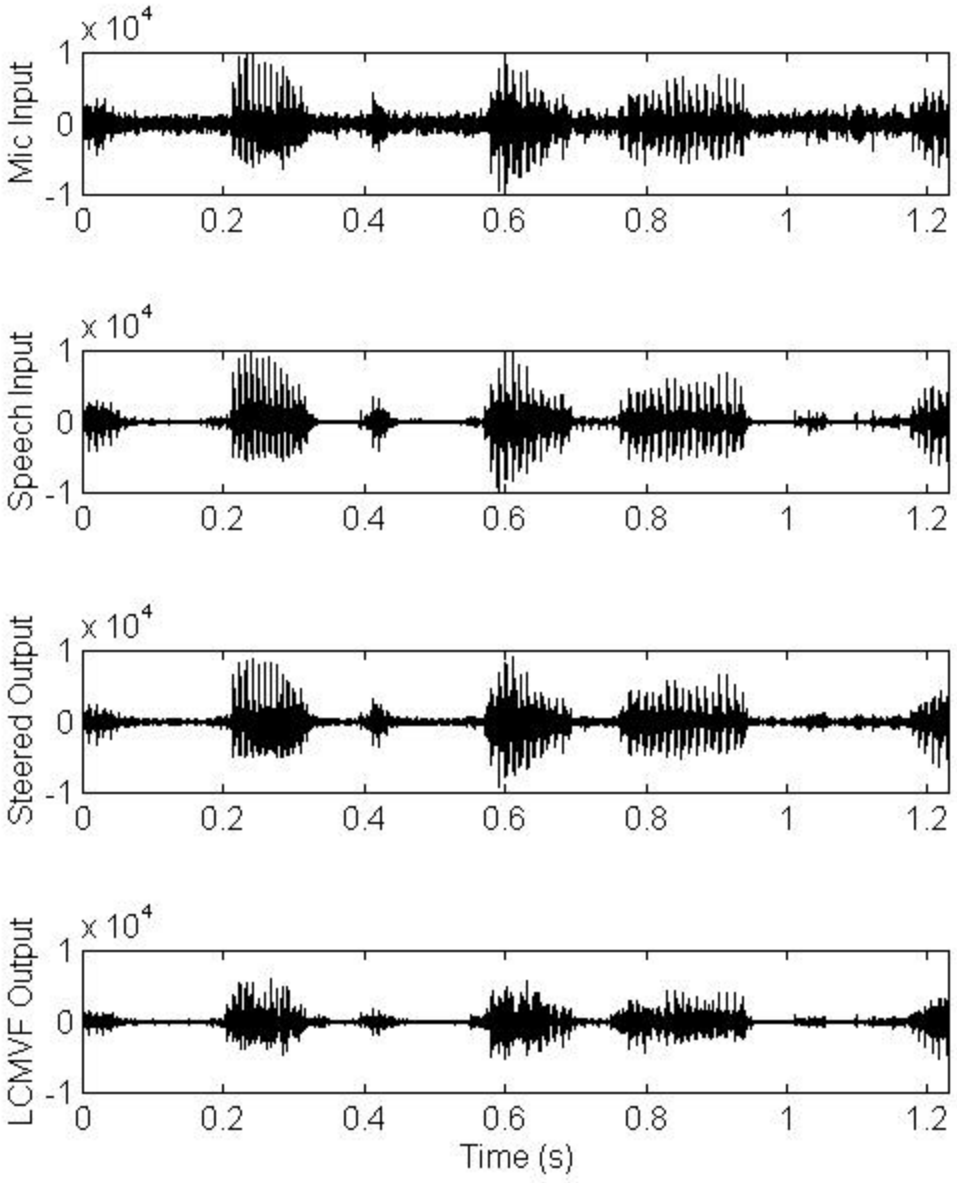

Figure 4.11 Microphone input, speech input, and beamformer outputs (phase-steered, LCMVF) with desired source at 135°, noise at 70°

LCMVF output. Also, listening to the output shows that the phase-steered output is clearer, and also shows that the LCMVF output has some echo in it. Also note that in these simulations the exact source direction is assumed known, so these results

correspond to the best-case scenario at given source arrival angles using these methods. With this in mind, let's increase the number of directional noise sources.

The output using 4 noise sources at 20°, 69°, 93°, and 159° is shown in Figure 4.12. In this case the problems with the LCMVF output are the same as in the previous

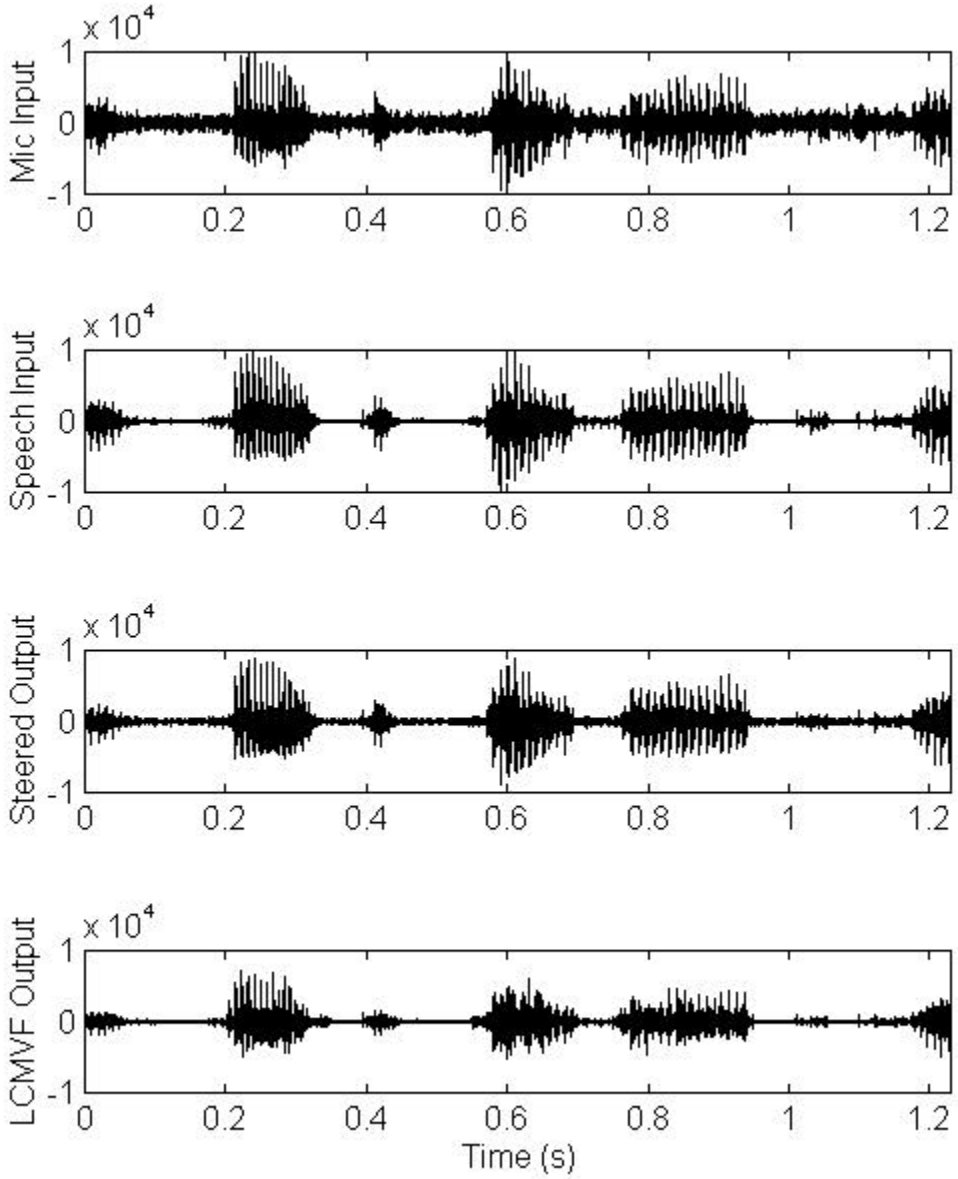

Figure 4.12 Microphone input, speech input, and beamformer outputs (phase-steered, LCMVF) with sources at 135° (desired), and at 20°, 69°, 93°, 159° (noise)

case. Both of these cases are at a 16 dB SNR during talkspurts. We can look at the beampattern using the weights of the LCMVF at a time when significant speech is present to see our beamformer gains. Figure 4.13 shows the beampatterns for each of the 6 subbands at the center frequency of each subband. The center frequency of each

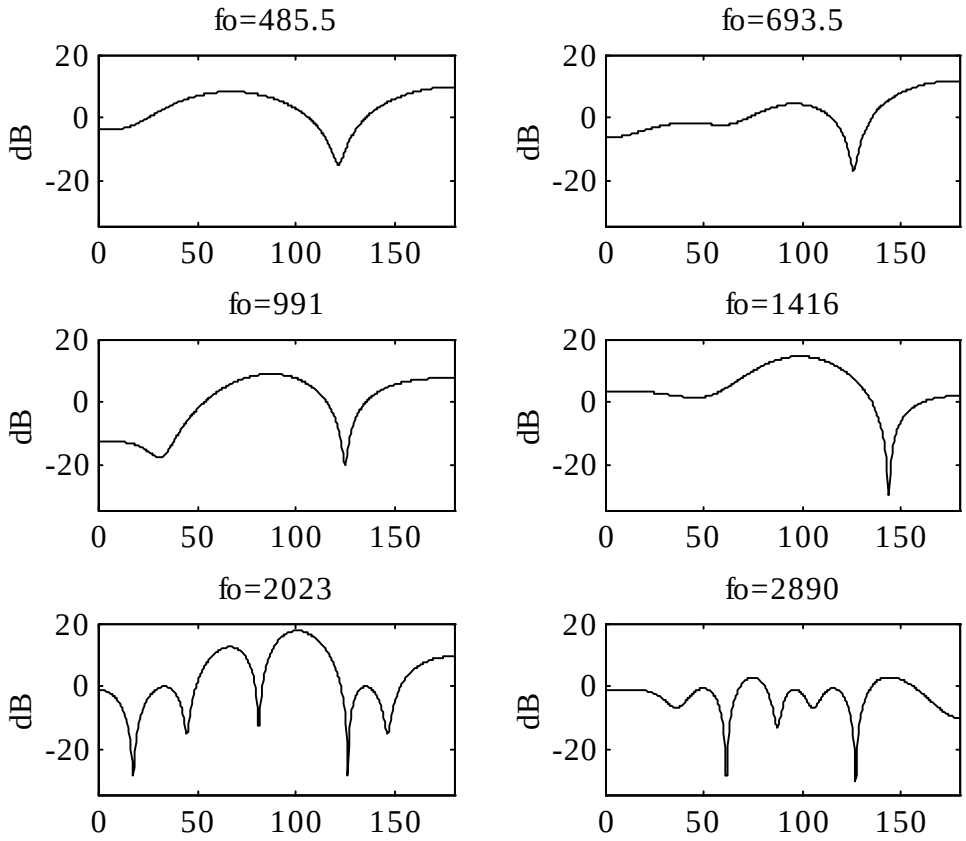

Figure 4.13 LCMVF beampatterns at 0.6 s into speech of Figure 4.12

subband was used to calculate each of the **a**(θ) vectors in (4.3), so we should expect unity gain at 135° in all cases, which is what we get. We can see from these graphs, unfortunately, that the mainlobes of the beampatterns are off from 135° by an average of 30°, and that some angles have almost 20 dB of gain, with some noise sources getting 10 dB or more gain in some of the subbands. As with the narrowband case, we should

expect the gain patterns to have its mainlobe centered more towards the desired source angle for lower SNR's, but the very poor performance at this SNR warrants us to devise an alternative beamformer solution.

There have been many methods for SNR improvement in a noisy environment with 1 microphone [7]. We would like to use our array to help improve the output. Frost [19] discovered that if we can get samples of the noise (with no signal), we can apply filters to the noise such that when combined with the signal plus noise (microphone inputs), the noise portion will be cancelled out. This adaptive array concept was developed by Frost and subsequently further developed by Applebaum [20] and Griffiths and Jim [21]. These methods try to minimize the output of the system, and require some method to implement it, usually some form of the least-mean-squares (LMS) algorithm. In this thesis we will focus on using Griffiths and Jim's Generalized Sidelobe Canceler (GSC).

4.5 Generalized Sidelobe Canceler (GSC)

This section discusses the layout of the GSC and some implementation issues. The general layout is shown in Figure 4.14. The four microphone outputs are time-delayed steered to produce 4 signals which ideally have the desired signal in phase with each other. These 4 signals are then sent to the blocking matrix. The purpose of the blocking matrix is to block out the desired signal from the lower part of the GSC. We want to adaptively cancel out noise, therefore we only want noise to go into the adaptive filters FIR1, FIR2, and FIR3. In the implementation used in this thesis, the blocking

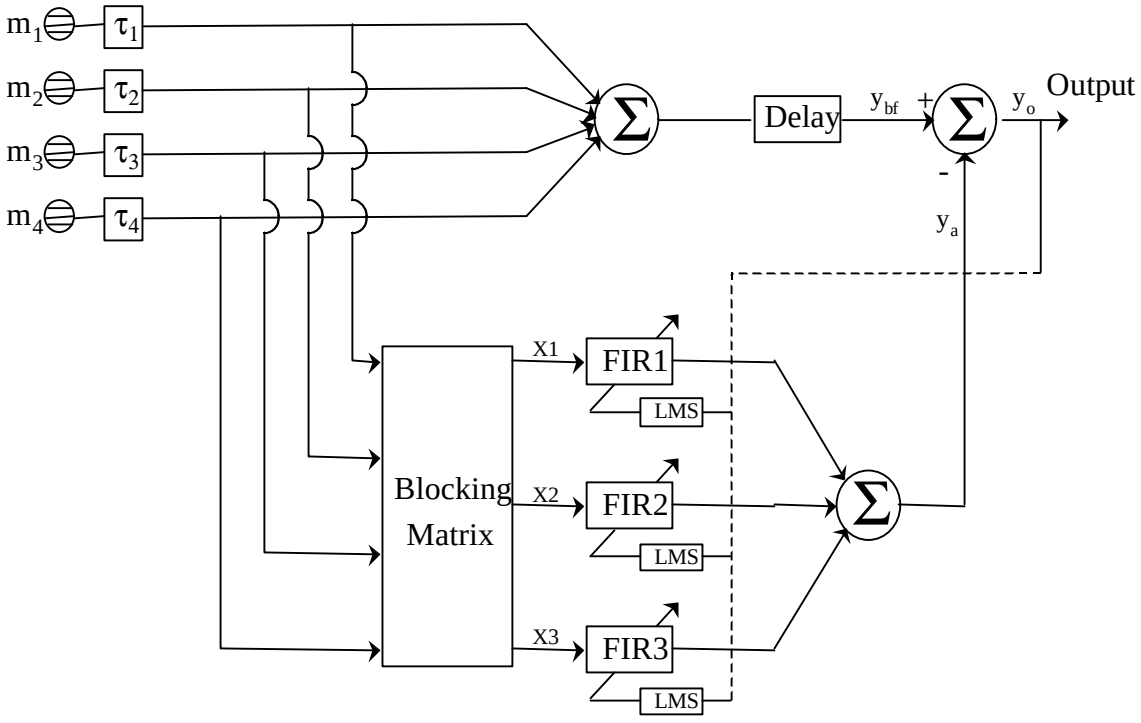

Figure 4.14 Generalized sidelobe canceler block diagram

matrix takes the following form:

$$\mathbf{B} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{4.6}$$

This just means that the outputs of the blocking matrix are the difference between successive signal samples. If the inputs to the blocking matrix are perfectly in phase, then this form works fine. The top branch of the sidelobe canceler produces the beamformed signal. The bottom branch, after the blocking matrix, contains 3 filtered versions of the noise components in the signal in the top branch. Now we would like to combine these three signals in such a way as to best approximate the noise component in

the upper (beamformed) signal, generating a linear combination of filtered noise components as the approximation. This can be done by minimizing the output of the GSC, which is the signal from the upper (beamformed) channel minus the signal from the lower (noise approximation) channel. Now we have to choose which method we want to use to adapt the FIR filter weights.

The ordinary LMS algorithm requires a user-specified step-size for filter coefficient updating. We would like the step-size to be correlated with the power of the received signals, such that we are confident about the convergence of the algorithm. The received signals change in power over time, so that the NLMS algorithm is a better approach. Now we need to establish some notation. The vectors $\mathbf{a}_{1,k}$, $\mathbf{a}_{2,k}$, and $\mathbf{a}_{3,k}$ contain the filter coefficients of FIR1, FIR2, and FIR3, respectively, at time k. These coefficient vectors can be combined to create an overall filter coefficient vector

$$\mathbf{a}_k = \begin{bmatrix} \mathbf{a}_{1,k} \\ \mathbf{a}_{2,k} \\ \mathbf{a}_{3,k} \end{bmatrix}. \tag{4.7}$$

The vectors $\mathbf{x}_{1,k}$, $\mathbf{x}_{2,k}$, and $\mathbf{x}_{3,k}$ contain the inputs into FIR1, FIR2, and FIR3, respectively, at time k. Each $\mathbf{x}_{i,k}$ satisfies the following equation:

$$\mathbf{x}_{i,k} = \begin{bmatrix} x_{i,k-\phi} \\ x_{i,k-\phi+1} \\ \vdots \\ x_{i,k} \end{bmatrix} \tag{4.8}$$

where ϕ is the filter order. The overall input vector is

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{1,k} \\ \mathbf{x}_{2,k} \\ \mathbf{x}_{3,k} \end{bmatrix}. \tag{4.9}$$

Now we can write the NLMS filter update equation at time k+1:

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \frac{\mu}{\|\mathbf{x}_k\|^2} \mathbf{x}_k y_{o,k}. \tag{4.10}$$

where μ is the step size. The current output, $y_{o,k}$, is

$$y_{o,k} = y_{bf,k} - y_{a,k} \tag{4.11}$$

where

$$y_{a,k} = \mathbf{a}_k^H \mathbf{x}_k \tag{4.12}$$

This algorithm requires 3(ϕ+1) multiplies, 3(ϕ+1)+1 adds, 3(ϕ+1) multiply-adds, and 3(ϕ+1) divisions. Fortunately, the updating for the filters can be done in parallel with 3 processors, thus reducing the computational complexity by approximately 3. This is one advantage of using the NLMS algorithm.

Finally, a delay is introduced in the beamformer section of the GSC. This delay occurs before the adaptive component $y_a(k)$ is subtracted from the beamformer component $y_{bf}(k)$ to produce $y_o(k)$. The delay is 1/2 of the adaptive filter order. This ensures that the component in the middle of each of the adaptive filters at time k corresponds to $y_{bf}(k)$ (meaning that both samples were generated from the same samples before their paths split).

Another method of implementing the adaptive filters is to use the RLS algorithm. Though it is much more complex than the NLMS algorithm, it does have a much faster convergence rate. The output $y_o(k)$ is generated the same way as before, but the adaptive filtering algorithm itself is much more involved. The three filters cannot be adapted separately, thus we cannot have parallel implementation. The weight vector $\mathbf{a}_k$ and input vector $\mathbf{x}_k$ are the same as before. We need to use the inverse of the correlation matrix in our computations. The correlation matrix correlates the samples in the FIR filters with each other; therefore the matrix will have $3(\phi+1)$ rows and columns. This matrix is usually initialized to a large positive constant times the identity matrix. We will denote this matrix as $\mathbf{P}$. The gain vector $\mathbf{k}_k$ at time k is then given as

$$\mathbf{k}_k = \frac{\mathbf{P}_{k-1}\mathbf{x}_k}{\lambda + \mathbf{x}_k^H \mathbf{P}_{k-1}\mathbf{x}_k} \tag{4.13}$$

where λ is a constant that relates to the memory of the system (λ=1 corresponds to infinite memory). Now the weight vector $\mathbf{a}_k$ is updated as

$$\mathbf{a}_k = \mathbf{a}_{k-1} + \mathbf{k}_k y_{o,k} \tag{4.14}$$

and the inverse of the correlation matrix is updated as

$$\mathbf{P}_k = \frac{\mathbf{P}_{k-1} - \mathbf{k}_k \mathbf{x}_k^H \mathbf{P}_{k-1}}{\lambda}. \tag{4.15}$$

Most of the computational complexity in this algorithm is in the updating of $\mathbf{k}_k$ and $\mathbf{P}_k$. Since $\mathbf{P}_k$ is a matrix, the number of computations used to calculate $\mathbf{P}_k$ gets exponentially

worse as the filter order ϕ increases. The next section will discuss the convergence of these two algorithms with different noises.

4.6 GSC Performance in Noise

This section discusses the performance of the GSC in noise. All cases discussed will have the beamformer steered to 50° (source direction). The performance of the GSC will be measured in terms of the noise improvement after the time-delay or phase-delay beamforming (y_{bf}). This noise improvement is equivalent to the output noise power with filtering compared to the power without filtering (termed 'relative noise power' from here on). Both LMS (NLMS) and RLS algorithms will be tested for convergence performance. In order to generate the fastest convergence rate for the NLMS algorithm, we will choose μ=1. A filter order of 40 will be used in all cases for comparison purposes, but performance does not significantly improve using filters with higher orders. In the first case we will use 1 noise source at 144°. Results are shown in Figure 4.15. The RLS algorithm converges much faster than the LMS algorithm in this case. Also, in the RLS implementation the noise power converges to a considerably lower value than the LMS algorithm. This is because the RLS algorithm theoretically produces zero misadjustment in a stationary environment, with infinite memory (though we will use λ=.9999 in the simulations).

Results for 4 pink noise sources arriving on the array at 20°, 69°, 93°, and 149° are shown in Figure 4.16. Again the RLS algorithm converges faster than the LMS algorithm, but the steady-state performance is similar. This is because the noise is much more complex than in the previous case. Unfortunately, at present we were unable to

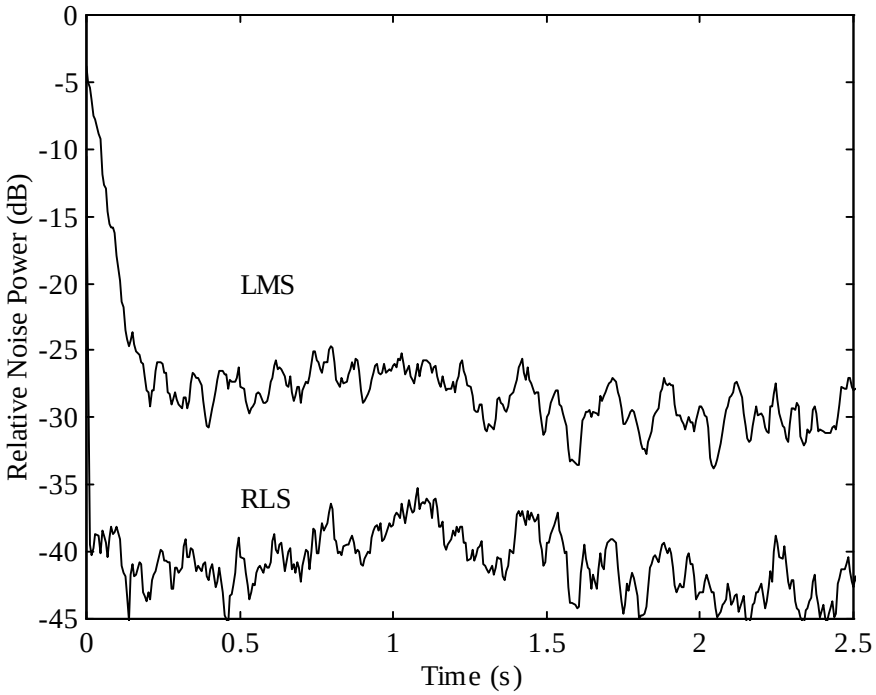

Figure 4.15 GSC noise improvement after beamforming using LMS and RLS algorithms, 1 pink noise source, f_s=8 kHz, FIR order = 40

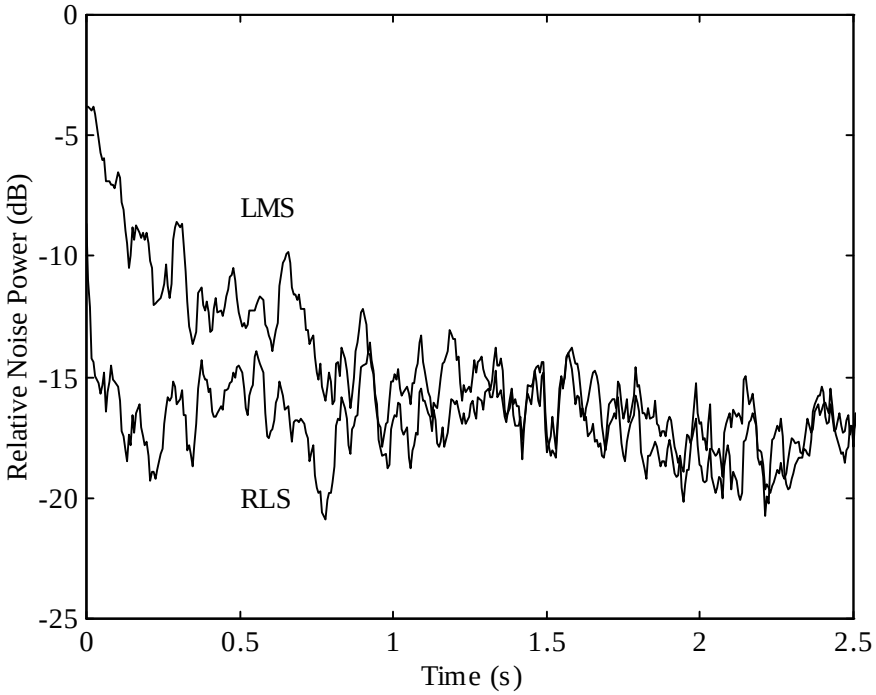

Figure 4.16 GSC noise improvement after beamforming using LMS and RLS algorithms, 4 pink noise sources, f_s=8 kHz, FIR order = 40

get 4-channel data from a car environment, so we can't precisely model the car noise environment. Therefore we will have to look at the worst-case scenarios. Results for 10 pink noise sources are shown in Figure 4.17. Both algorithms have similar performance, due to the fact that the noise is becoming more and more uncorrelated at the sensor inputs. The 10 dB performance improvement is still significant.

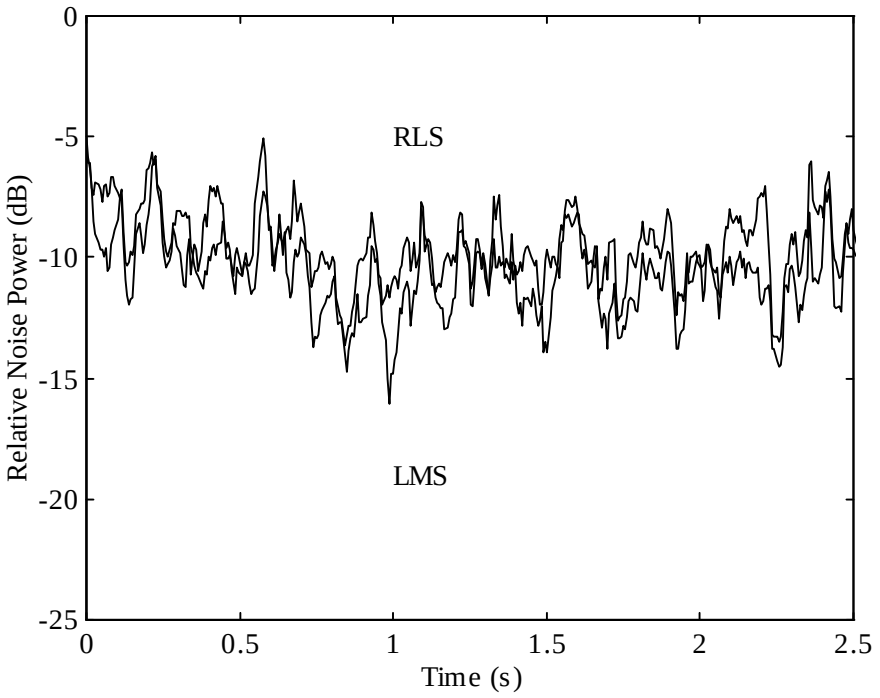

Figure 4.17 GSC noise improvement after beamforming using LMS and RLS algorithms, 10 pink noise sources, f_s=8 kHz, FIR order = 40

The spatially uncorrelated noise case is shown in Figure 4.18. The performance here is very poor, as expected. The LMS algorithm gives relatively better results for this case, because previous data, that the RLS algorithm uses, is uncorrelated with the current data. Though this case will never occur in a car environment with closely spaced microphones, it is useful to note the trend as the noise environment becomes more complex; use of the LMS algorithm becomes more advantageous over the RLS

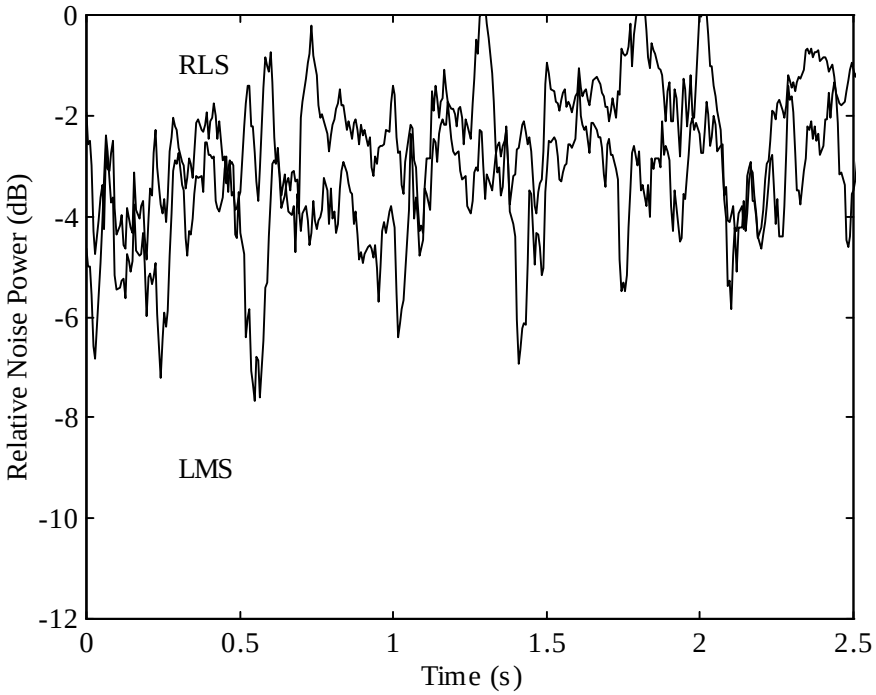

Figure 4.18 GSC noise improvement after beamforming using LMS and RLS algorithms, uncorrelated pink noise sources, f_s=8 kHz, FIR order = 40

algorithm due to its competitive performance, and its inherent relative computational simplicity. The decision as to which algorithm to use should be based on hardware restrictions first, and then on the relative performance between the algorithms in the actual car environment.

Another decision needs to be made as to the optimal order of the FIR filters. In the 4 pink noises case, convergence of the LMS algorithm was slightly slower as the FIR order increased. Figure 4.19 shows the performance with the FIR order equal to 10, 20, 30, and 40. The performance at an order of 10 is significantly worse than for the other three orders, whose performances are almost indistinguishable. So the best filter order for this scenario would be between 10 and 20. Once again, the final decision as to the

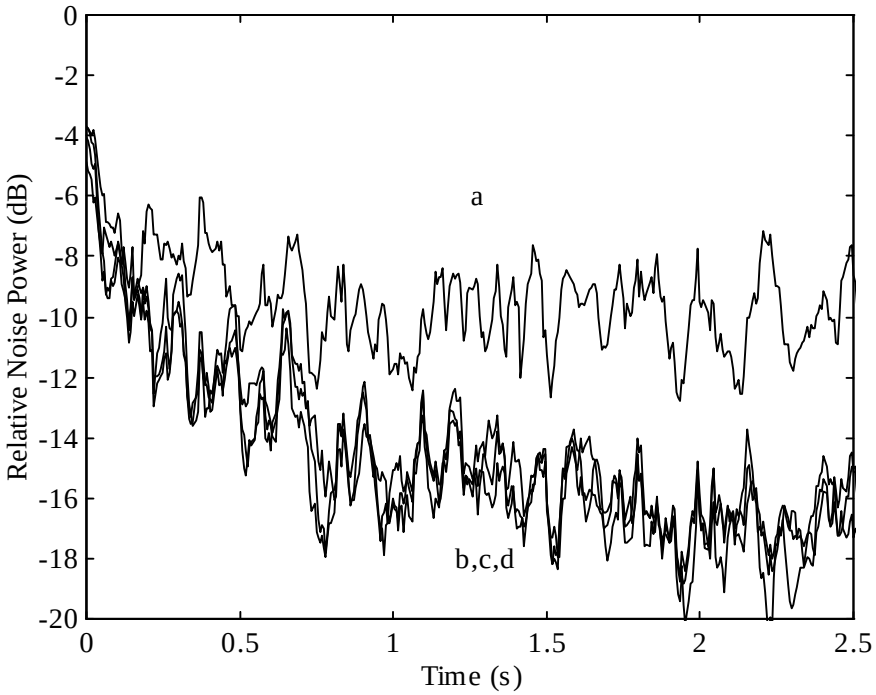

Figure 4.19 GSC noise improvement after beamforming using LMS algorithm, 4 pink noise sources, Fs=8 kHz, FIR order = a) 10, b) 20, c) 30, d) 40

optimal filter order must be made when actual 4-channel data can be taken from the car noise environment.

4.7 Single Speech Source with Noise

We have shown the improvement in using the GSC to cancel out noise, but we need to see how it reacts with speech. The adaptive algorithms try to minimize the output; therefore, the algorithms will try to reconstruct the speech in the upper part of the GSC with the noise in the lower part. We are going to simulate the GSC with noise first, then speech. In this simulation we are using the exact arrival angle for the source to compute the delays for each channel, thereby isolating the speech reconstruction problem. Figure 4.20 shows one of the microphone inputs and the GSC output for a 10

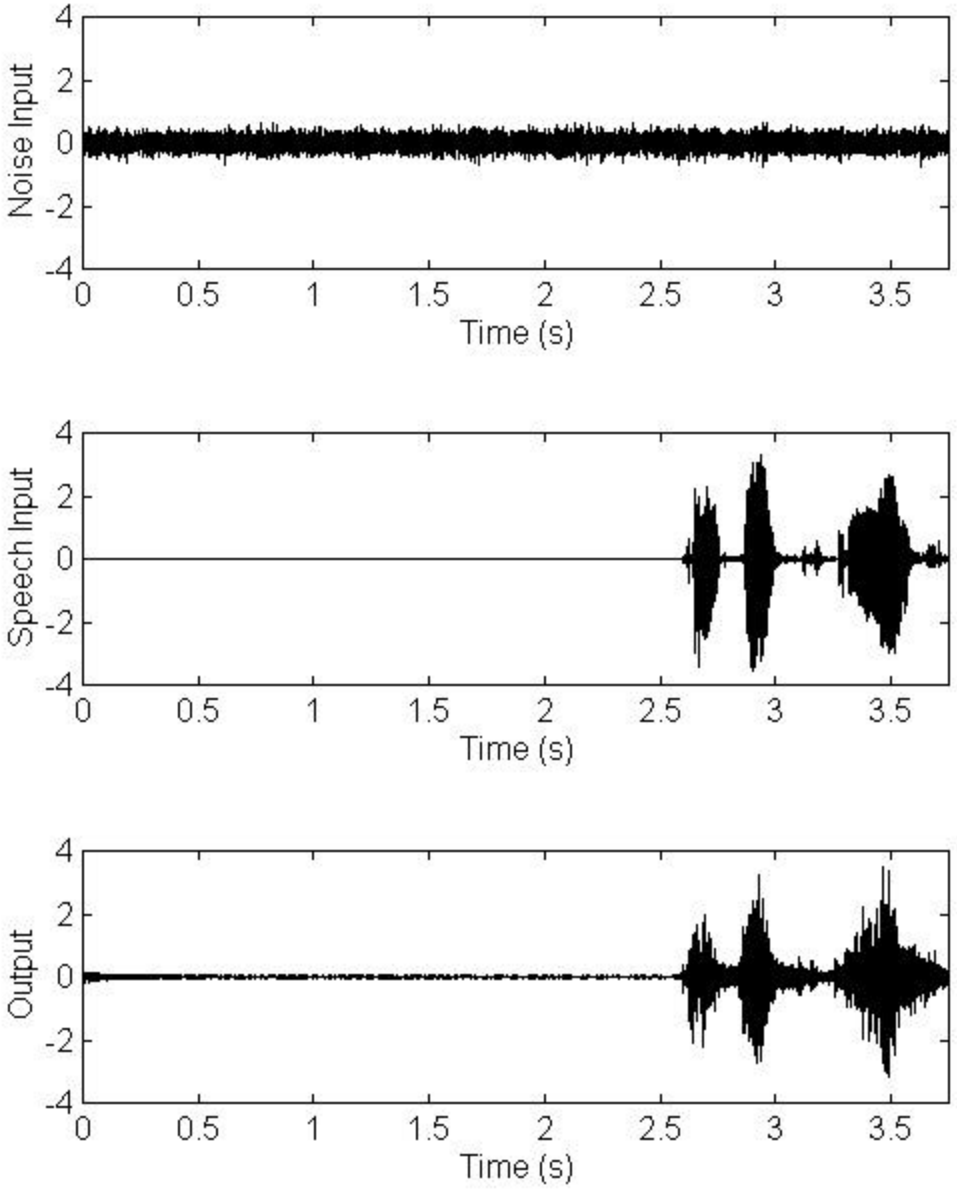

Figure 4.20 GSC results using 20th order LMS-adaptive filters, 4 pink noise sources, mic inputs = speech + noise

dB SNR scenario. The GSC adaptively cancels out the noise for the first 2.6 seconds, then the speech signal arrives at the inputs. Ideally, the output should equal the speech signal, but this is not the case as seen in Figure 4.20. After listening to the output, it is

obvious that there is a large echo problem. This echo problem results from the adaptive filters trying to reconstruct the speech signal from noise (the algorithm is trying to minimize the output power only so it does not differentiate between minimizing speech and noise). This can be overcome by adaptive echo cancellation algorithms, which try to synthesize a replica of the echo and subtract it from the output. There is another problem with adaptation of filter coefficients while there is speech input. If some speech leaks through to the lower part of the GSC (realistic case), then the filters will also try to reconstruct the speech, causing cancellation of the speech. The alternative is to freeze the filter coefficients during speech segments, thereby avoiding speech reconstruction and echo problems**.** Results for the same scenario as in Figure 4.20 but with adaptation during noise-only segments is shown in Figure 4.21. The output now closely matches the speech input. But we must first make sure that the filters have adapted for a sufficient amount of time. Figure 4.21 shows that the speech signal diminishes by very little with the adaptive noise cancellation. Therefore we can examine how long the adaptive filters need to get a good noise environment estimate by examining how well the filters cancel out noise once its weights are frozen after different lengths of time.

First we will examine the performance of the LMS algorithm. Figure 4.22 shows the performance of the LMS algorithm with weights fixed after different numbers of adaptations. From this data we notice two trends (μ=1). The first trend is that the performance after weight fixing gets better as the number of samples of adaptation increases. The second trend is that the performance after weight fixing is 4-5 dB worse than before the weights were fixed. This is a result of the filters converging around the optimal value but not moving closer. This can be remedied by using a smaller μ. It

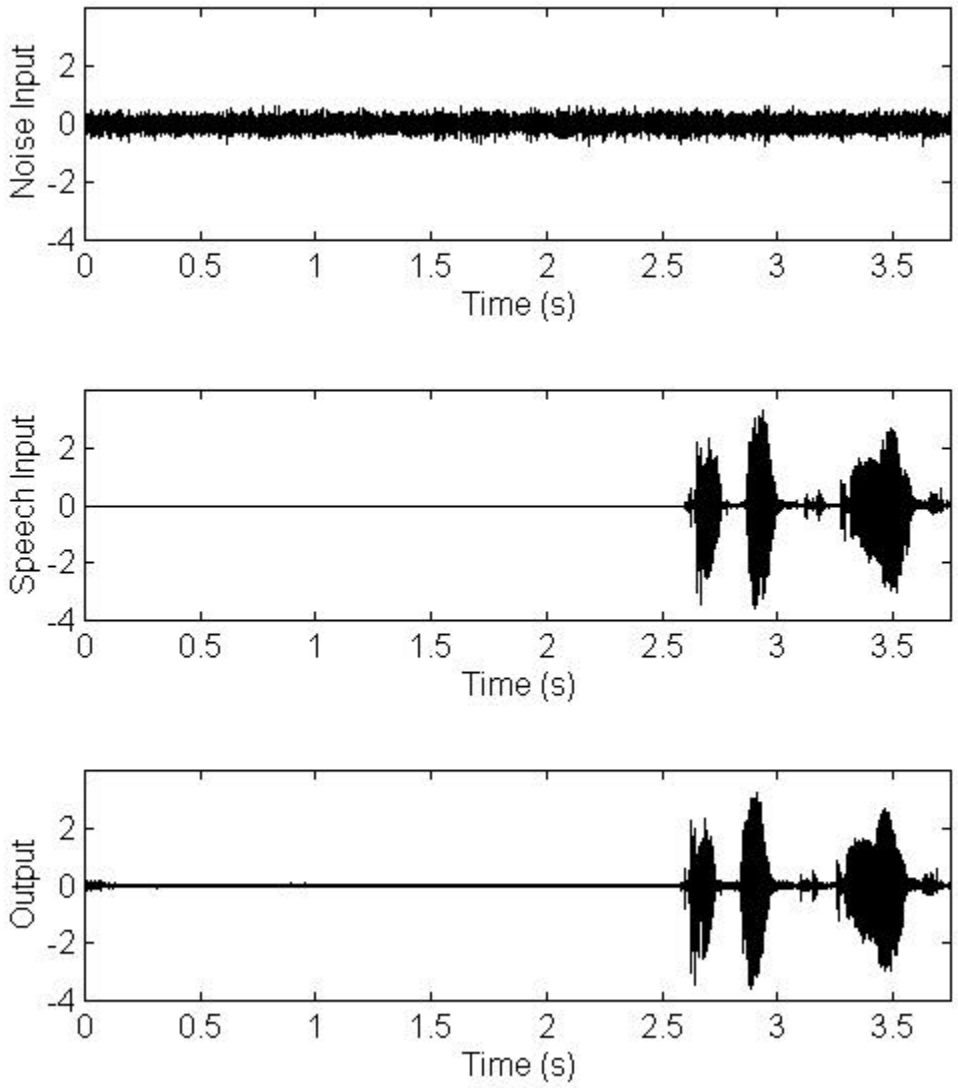

Figure 4.21 GSC results using 20th order LMS-adaptive filters, 4 pink noise sources, mic inputs = speech + noise, adaptation during noise-only segments

would be helpful to speed up the convergence initially by using μ=1 for a small amount of time, and then moving to a lower μ. Figure 4.23 shows results with μ=1 for the first 1200 snapshots, then moving to μ=0.1 for the rest of the snapshots. Although with the new scheme the performance when the weights are adapting is about 2 dB worse than

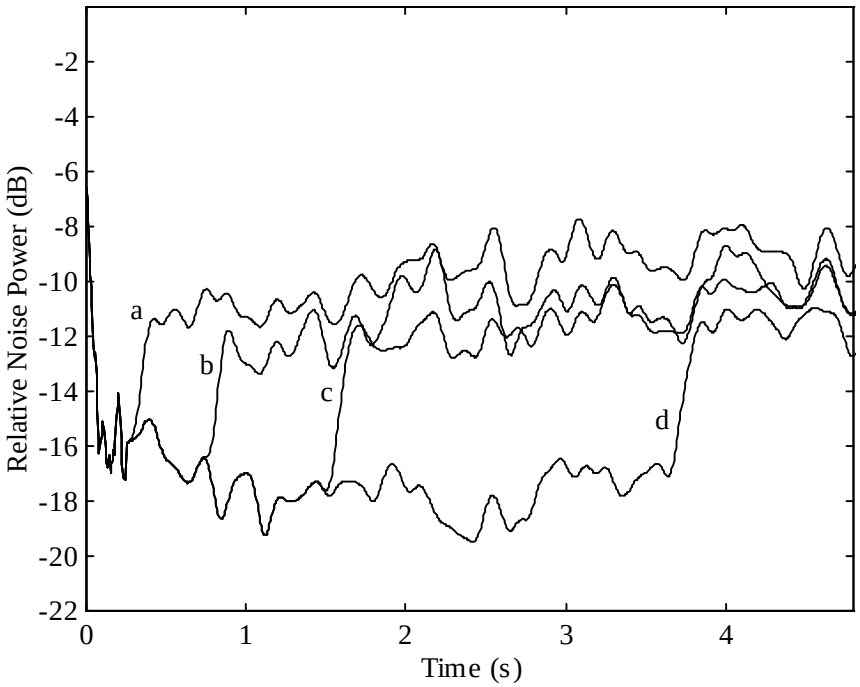

Figure 4.22 GSC performance (LMS, order=20, μ=1) with weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

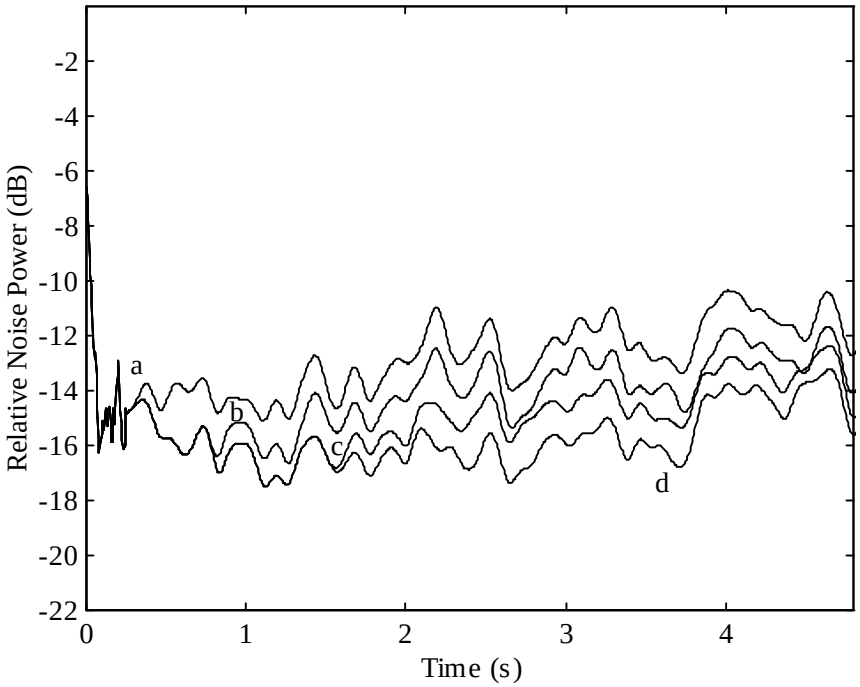

Figure 4.23 GSC performance (LMS, order=20, μ=1 for 1st 1200 snapshots, then μ=0.1 for rest) with weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

before, the performance once the weights are fixed are between 3 and 5 dB better. As can be seen from these graphs, a smaller μ gives us better performance once the weights are fixed, but performance suffers when the weights are adapting. With this in mind, the results in Figure 4.23 achieve a good balance.

The RLS algorithm, though much more computationally complex than the LMS algorithm, should give improved performance over the LMS algorithm. The RLS algorithm inherently has lower misadjustment over the LMS algorithm for similar filter orders. Results of the noise performance with weight fixing, at times similar to those in Figure 4.22, are shown in Figure 4.24. The results after weight fixing are much improved over the LMS algorithm implementation, and are very similar to the adaptive results even 4 seconds after the weights have been fixed. The advantage of using the RLS algorithm

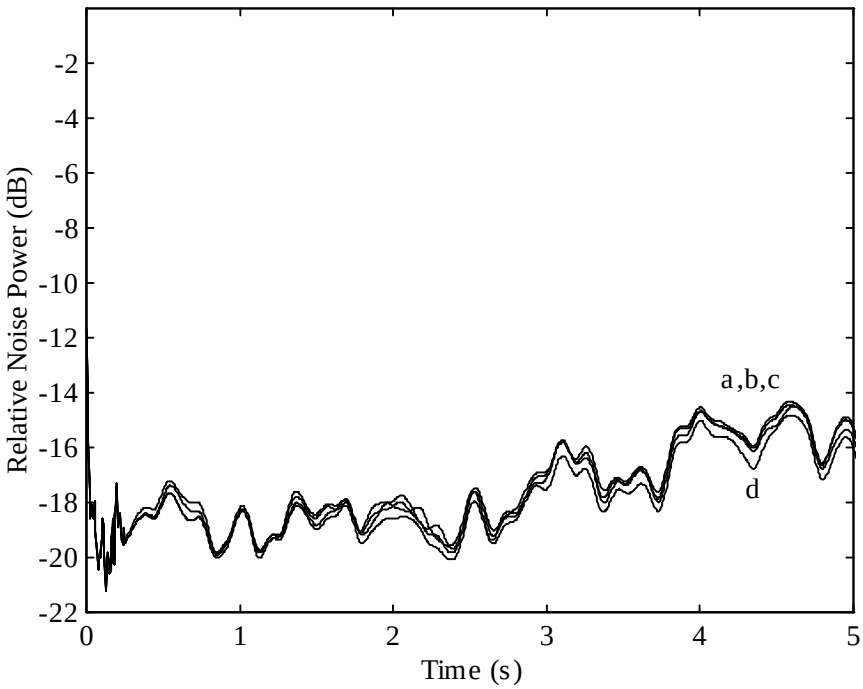

Figure 4.24 GSC performance (RLS, order=20) with weights fixed after a) 1200 snapshots, b) 2400 snapshots, c) 6000 snapshots, d) always adapting

is not only its excellent performance during weight adaptation, but also its rapid convergence, which means that only a fraction of a second is required to obtain near-optimal filters. The computational complexity, though, may preclude this from being an economically viable option.

Aside from the decision of which algorithm and order to use for the adaptive filters, there are other aspects of the GSC which need to be looked at more closely. We need a robust speech detection algorithm. If the speech detector mistakes speech for noise, then not only will we encounter echo problems discussed previously, but with inherent signal leakage into the adaptive filters, speech will also be cancelled. If the speech detector mistakes noise for speech, then the output will not be as good, plus the DOA estimate will be completely wrong. Speech detectors will be discussed in Section 6.3. Another aspect of the GSC that needs to be examined is the optimization of phase aligning the desired speech in the 4 microphone inputs. Since the adaptive filters are fixed during speech segments, it would be undesirable to change the time-delay or phase-delay at the beginning of the GSC. Also, in case the speech detector detects speech as noise, having the channels aligned as close as possible will help mitigate source cancellation. Other aspects of the adaptive beamformer will be discussed in Chapter 6.

4.8 Alternative GSC Implementation

The GSC described previously uses time-delay steering to align the 4 microphone signals for use in the beamformer and noise-canceler. Alternatively, we could divide the inputs into subbands, as discussed in Section 3.7, and then create separate GSC's for each subband. Any number of subbands could be used, but we will stick with using 6

subbands. The GSC for each subband *i* is as shown below:

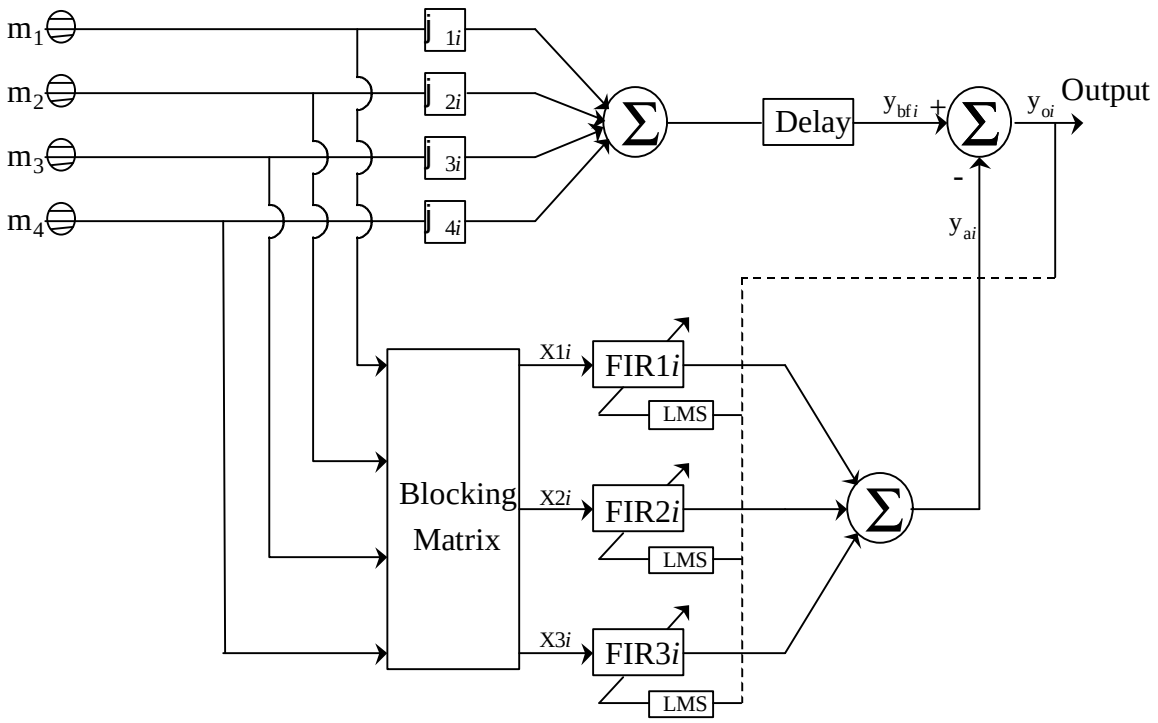

Figure 4.25 GSC block diagram for subband *i*

Each subband has its own GSC, thus we will have 6 separate GSC's. The phase factors φ_{ji} for the beamformer for each microphone *j* and subband *i* is

$$\varphi_{ji} = e^{j2\pi d_j \cos\theta/\lambda_i}, \tag{4.16}$$

using the notation as in Chapter 2 with λ_i corresponding to the center wavelength of each frequency band. Using this notation, the blocking matrix $\mathbf{B}_i$ for each subband *i* is

$$\mathbf{B}_i = \begin{bmatrix} 1 & -\varphi_{2i} & 0 & 0 \\ 1 & 0 & -\varphi_{3i} & 0 \\ 1 & 0 & 0 & -\varphi_{4i} \end{bmatrix}. \tag{4.17}$$

Ideally, the blocking matrix phase-aligns each microphone with microphone 1. Though this gives us better resolution for alignment over the time-delay GSC implementation, our subbands aren't exactly narrowband, so some frequencies will be better aligned than others.

We would like to see how well this GSC implementation performs relative to the time-delay implementation. All tests will use 4 pink noise sources at 20°, 69°, 93°, and 159°, as in the previous section. The first test is the same test used in the time-delay GSC for which results were shown in Figure 4.22. Results in the previous sections using the time-delay GSC and those in this section show improvement *after* beamforming, so we will also examine the relative beamforming performance as well (which will be done in the next section). For computational reasons, the filter order for the results in this section has been reduced to 20, since we found out from Figure 4.19 that in this 4 pink noises scenario there is an almost indistinguishable difference. Results using the new phase-steered GSC with filter orders of 20 and μ=1 are shown in Figure 4.26. From Figure 4.26 we see that the filters provide reasonable attenuation when the weights are varying, but once the weights are fixed, the GSC actually hurts the SNR. The reason for this is that adaptive filters inherently have a harder time converging if the input signal takes up a small portion of the range [0, f_s /2]. We should then try a lower μ for the filters so that once the weights are fixed, the filters are closer to the overall optimal solution.

Results for μ=0.1 are shown in Figure 4.27. Though the performance when the weights are adapting are about 4 dB worse than with μ=1, the performance once the weights are fixed is much better. Still, performance is worse than the time-delay

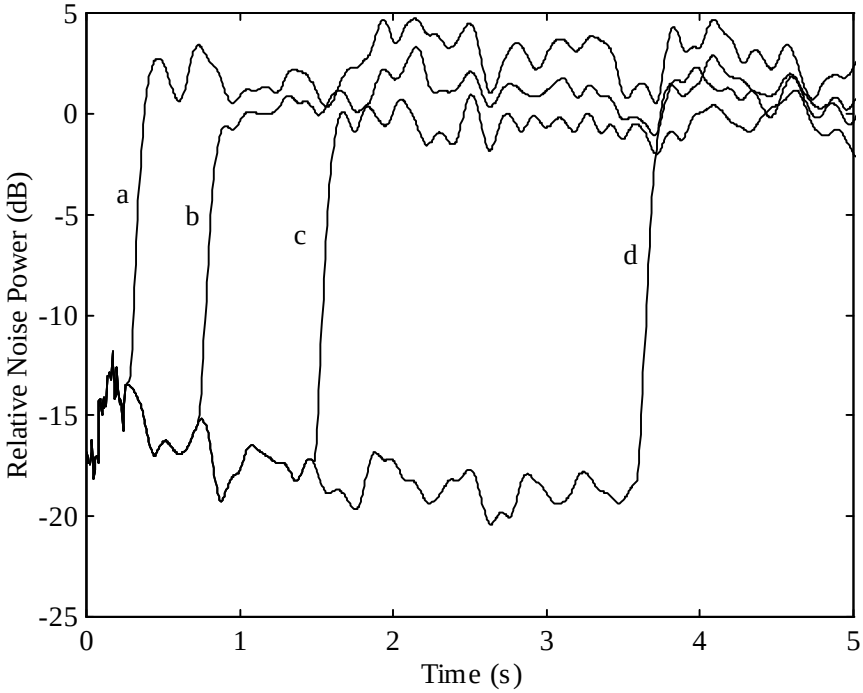

Figure 4.26 GSC performance (LMS, 6 subbands, order=20, μ=1) with weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

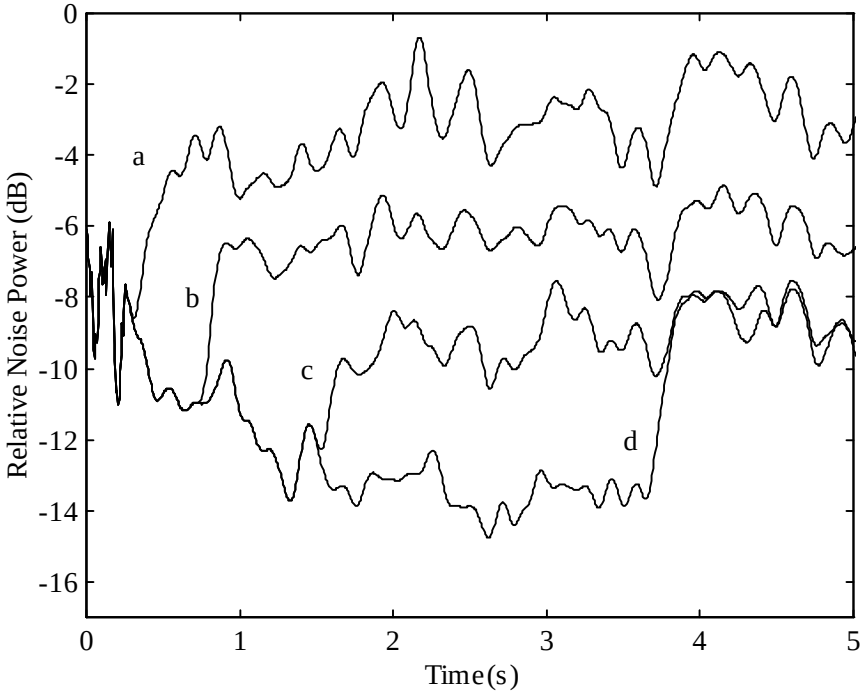

Figure 4.27 GSC performance (LMS, 6 subbands, order=20) for μ=0.1 with weights fixed after a) 2400, b) 6000, c)12000, d) 28800 snapshots

GSC performance. Looking at the relative rise in noise power once the weights were fixed at 28800 snapshots, we get 4.0, 2.5, 1.7, 4.7, 3, and 1.3 dB in the lowest to highest bands, respectively. It may be better to switch the step size down to μ=0.01 after some time. Figure 4.28 shows results when μ stays at 0.1 and when μ switches to 0.01 after 3 seconds of adapting. Figure 4.28 shows that it is better to keep μ at 0.1, before and after the weights get fixed.

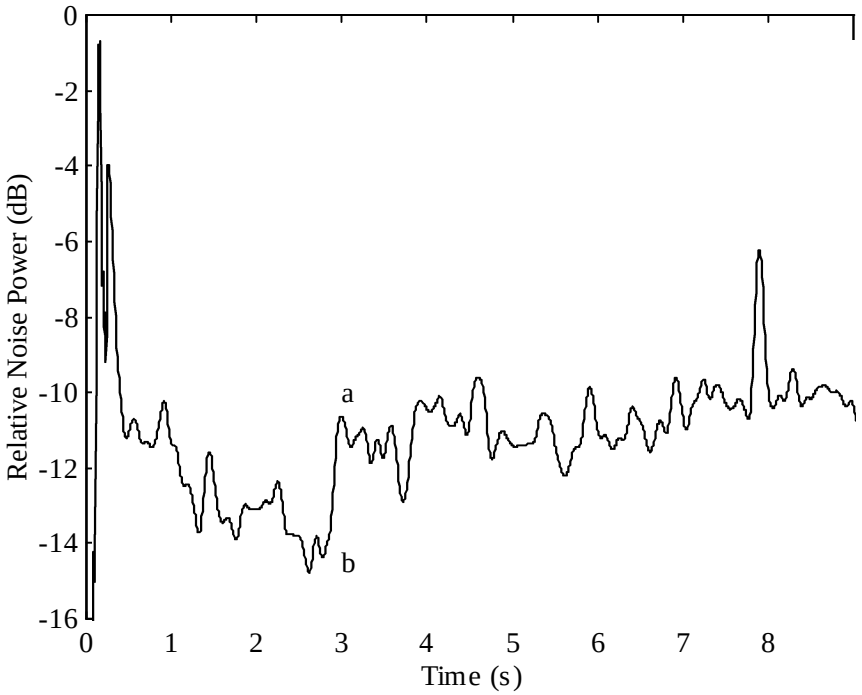

Figure 4.28 GSC performance (LMS, 6 subbands, order=20) with weights fixed after 6 seconds a) μ=0.1 for 1st 3 seconds, μ=0.01 for next 3 seconds b) μ=0.1

We are trying to find ways to speed up the convergence of the filters in the GSC. Another idea is to set μ=1 for a small number of snapshots, then switch to a lower μ. Results using this method in a test similar to Figure 4.27 are shown in Figure 4.29. Though in this case the adaptive filters have 2400 extra samples at μ=1, it is seen from

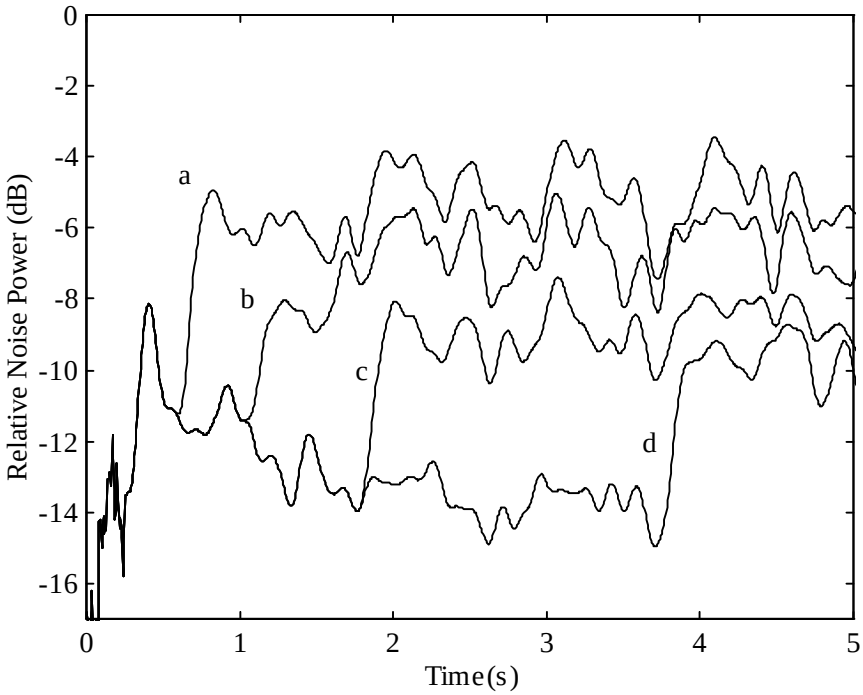

Figure 4.29 GSC performance (LMS, 6 subbands, order=20, μ=1 for 1st 2400 snapshots, then μ=0.1) with weights fixed after a) 4800, b) 8400, c)14400, d) 31200 snapshots

both figures that the performance when the weights are adapting are almost identical.

We should look at the performance of the GSC with RLS adaptation for 6 subbands. We have performed a test similar to that shown in Figure 4.24, except now we are using 6 subbands, and an FIR order of 20 for each subband. Results are shown in Figure 4.30. The results here are about 4 dB worse than the results using the time-delay GSC, even though theoretically the RLS algorithm performs independently of the frequency content of the inputs. The reason for this is a result of the different beamforming methods used for both cases. With the time-delay GSC, each component of the noise signal was delayed a fixed amount for the beamformer and blocking matrix inputs. With the phase-delay GSC, each component is phase-delayed by a different

amount, depending on the frequency components of the noise, plus there is some mismatch due to the deviation from the narrowband assumption in the subbands. This mismatch changes with time because the noise is random. Based on performance after beamforming, RLS should not be used in the subband case, since the theoretical performance will be at best as good as that in the single band case, and the computational complexity will increase by a factor of 6.

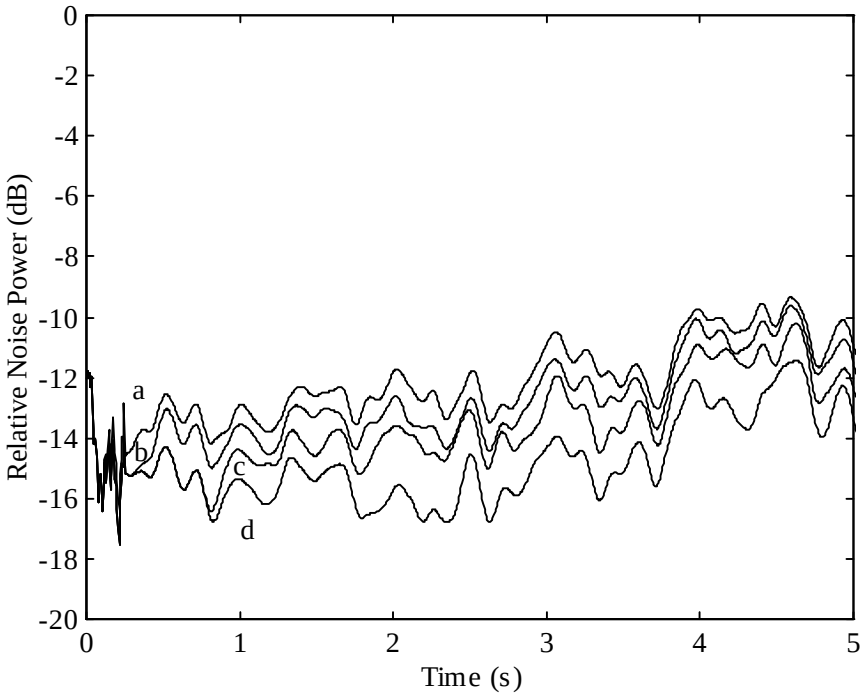

Figure 4.30 GSC performance (RLS, 6 subbands, order=20) with weights fixed after a) 1200 snapshots, b) 2400 snapshots, c) 6000 snapshots, and d) always adapting

4.9 Performance of time-delay and phase-delay beamformers

Now we need to determine the relative overall improvement using subbands in the GSC implementation. Previous results are based on improvement with respect to the beamformed output. But what is the relative performance of the phase-delay and time-

delay beamformers? The phase-delay beamformer would seem to better reject signals not coming from the source direction, with signals outside of the main lobe getting at least 10 dB of rejection (Figure 3.20), whereas the time-delay beamformer will have random alignment of the noise with the signal at different frequencies. It would be best, though, to objectively evaluate the beamformer performances.

One way to objectively measure the beamformers' performance is to measure the segmental SNR between the speech input and the time and phase-delay outputs. The segmental SNR computes an SNR value for small segments of the speech (usually 15-25 ms) and can be computed for a segment of *n* snapshots as follows:

$$\mathrm{SNR}_{\mathrm{seg}} = 10\log_{10}\frac{\sum_{n} s^2(n)}{\sum_{n}[s(n)-o(n)]^2} \tag{4.18}$$

where $s(n)$ is the speech input and $o(n)$ is the time-delay or phase-delay output. These segmental SNR's can be averaged over time to produce an SNR for a longer segment. Care must be taken to insure that the speech signal and the beamformed signal are in phase for accurate comparison.

We would like to look at both beamformer outputs and compare them to the input for a typical case. Sources were originally sampled at 40 kHz to better simulate the delays into the microphones, and then resampled to 8 kHz. Beamformer outputs and speech and mic inputs for the 135° source (speech) and 20°, 69°, 93°, and 159° sources (pink noise) case is shown in Figure 4.31. It is hard to see any noticeable difference between the time-delay and phase-delay outputs. Figure 4.32 shows the segmental SNR plot, which shows us that during noise segments there isn't much difference between the

two, but during the speech segments the phase-delay beamformer works a little bit better. The average SNR's were 0.6 dB for the phase-delay and 0.3 dB for the time-delay, which is very little difference.

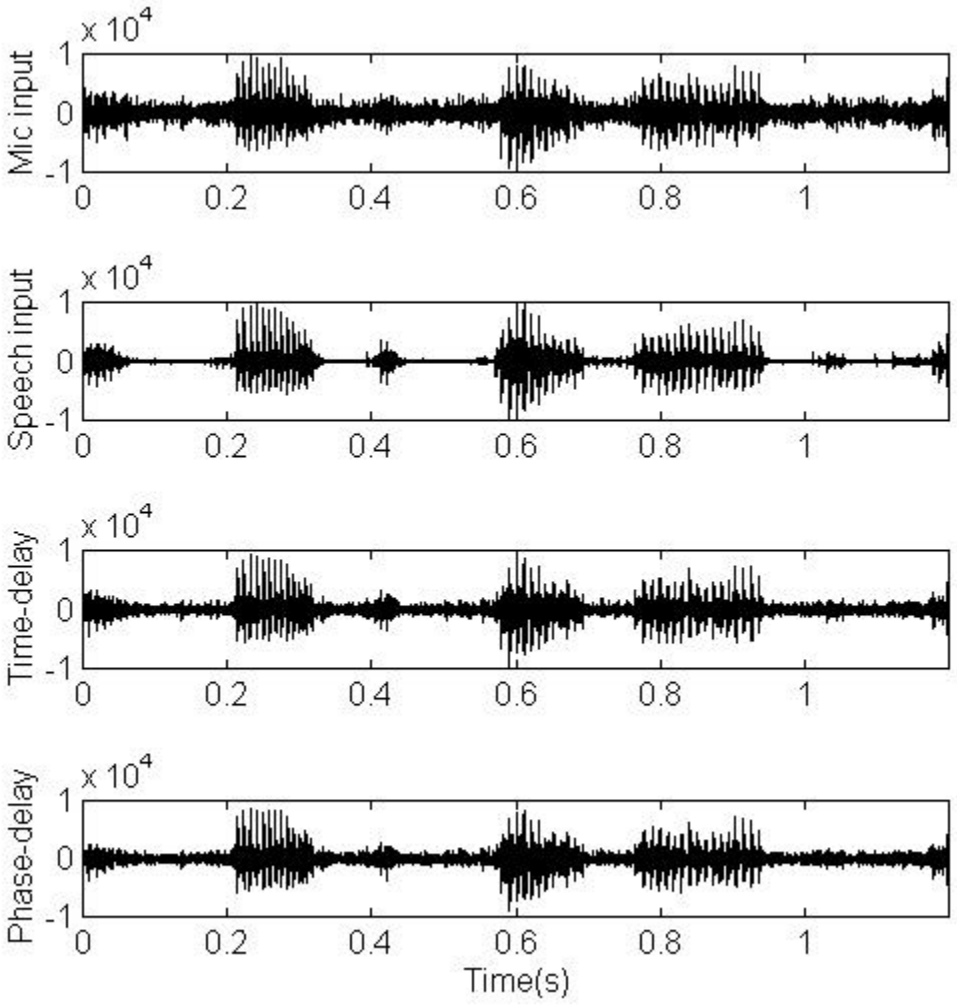

Figure 4.31 Beamformer outputs using phase delays (6 subbands) and time delays, sources at 135° (speech), and 20°,69°,93°,159° (pink noises)

Now that we have an objective way to measure beamformer performance let's see what results we get for a variety of noise scenarios. In the first case 4 pink noise sources were input at random angles, while the speech source was incident at 135°. In the second

case 4 pink noise sources were incident at random angles and the speech source was incident at 110°. The average SNR's of the beamormers' outputs for these two cases are shown in Table 4.2. The phase-delayed output has an average of about 0.3 dB better

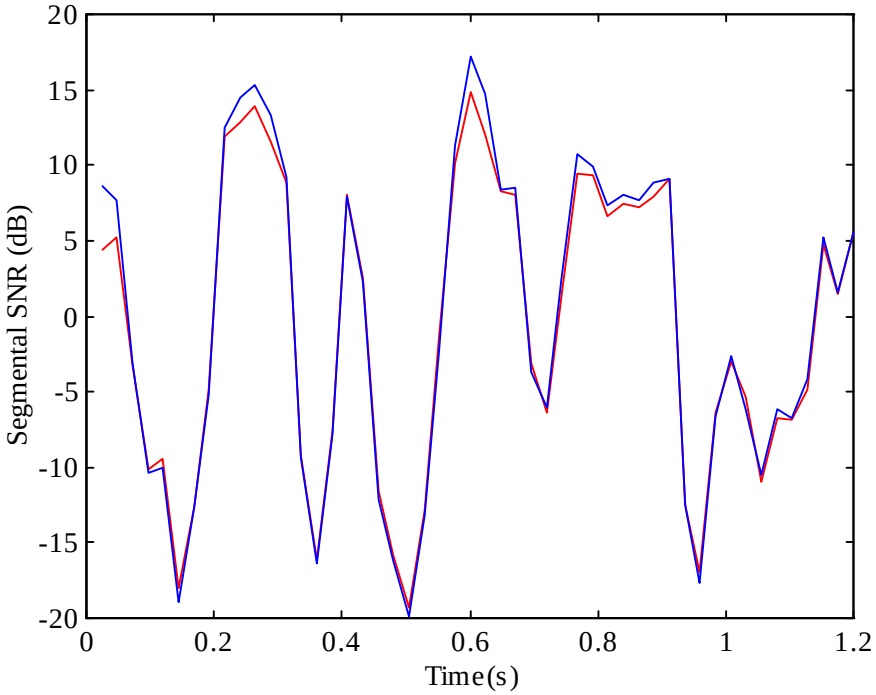

Figure 4.32 Segmental SNR's from Figure 4.31, time-delay (red), phase-delay (blue)

Table 4.2 SNR's for phase-delay output (PD) and time-delay output (TD)

	135° speech cases					110° speech cases				
	1	2	3	4	5	1	2	3	4	5
PD	0.6	-1.0	-2.3	-0.8	0.8	0.0	-0.5	-1.0	3.4	0.3
TD	0.3	-0.2	-2.6	-0.7	1.2	-0.2	-0.8	-1.2	3.0	-0.2

SNR than the time-delayed output for these cases. From the data in this section, though, we have discovered that there is very little difference in the beamformer performance

using phase-delays over time delays, though the phase-delay beamforming gives us better performance, as we have expected.

4.10 GSC Conclusions

From what has been discussed in this chapter, we need to determine the characteristics of the GSC that we will use for the experiments in Chapter 7. We have found that, discounting computational costs, the performance using 1 band as compared to 6 subbands are similar enough to warrant testing both of them with real data. Using the phase-delay GSC, we will go with a step size of μ=0.1. If the DOA angle doesn't change by much, then we shouldn't alter the phase-delay coefficients so that the output doesn't suffer. When we use the time-delay GSC, we will start with μ=1. Though after a small amount of time, we will decrease μ and see if that has any effect on the performance after weight fixing.

Using 6 subbands increases the computational complexity of the GSC by a factor of 6. When we test the GSC's in Chapter 7 we will discover which GSC performs better, and if the phase-delay GSC performs better, whether the extra computational complexity is warranted. The testing setup in Chapter 7 will not only have a different noise environment than the 4 directional noise sources used in this chapter, but also will have near-field effects and non-ideal microphones. Ultimately, though, the decision as to which order and step size configuration to use for the adaptive filters in the both GSC's need to be determined from tests in different car environments.

Chapter 5

Microphone Array Data Acquisition Setup

5.1 Overview

The purpose of the hardware and software developed for this project, and discussed in this section, is to acquire acoustic samples from four microphones simultaneously. The signals from the microphones need to be amplified first, since we want to use as much of the A/D converter range (quantization levels) as possible. Then each of the signals needs to be converted to a digital one simultaneously. These digital signals then must be sent to a microcontroller that will do all of the array processing and adaptive beamforming. The processing and beamforming can also be done using a computer, if there is an interface between the microcontroller and the computer, as long as there is software developed that will download data real-time from the microcontroller. This chapter shows the layout of the beamformer system using the computer to collect samples from the 4-microphone array. The beamforming will be done using the computer, and the results will be shown in the next chapter.

5.2 Component Details

This section details the major components used in the data acquisition setup. The components detailed are: 1) the ADSP 2181 microcomputer, which collects 4-channel data from the A/D converter and is programmed to send the data to the computer for processing and beamforming, 2) the AD7874 A/D converter, which converts 4 analog channels into a serial 12-bit digital stream, 3) the AD SSM2135 op-amp, which amplifies

the signal from the microphone, and 4) the Labtec AM-242 microphone and AD SJ-353-100 microphone adapter, which converts air pressure (sound) to an analog voltage.

The ADSP 2181 microcomputer is the core of the beamforming system. This microcomputer takes in serial 12-bit digital data from the AD7874 A/D converter.

5.3 Wiring Details

This section details the overall hardware layout. The hardware layout is shown on Figure 5.1. The four Labtec AM-242 microphones are each plugged into an SJ-353-100 microphone connector. This connector has stereo output, on pins 3 and 2, and the other pins are grounded. The microphone sends stereo data to the SJ-353-100, which means that both channels contain the same data. We only need one channel, so we arbitrarily use pin 3 for the output. We also need DC power for the microphone, because we need to charge the capacitor inside the microphone, separating the sides of the capacitor such that incoming sound pressure will be able to change the distance between the sides, which allows capacitance changes that change the output. This is done by sending 1.5 volts into the other channel of the SJ-353-100. Since both channels are electrically connected (because the same signal is sent out of both channels), this 1.5 V will appear at the other output, and thus we have applied "phantom power" to the microphones. If the same power supply was used for all microphones, there would be crosstalk between the channels. It was also discovered that the microphones require a varying amount of power, so to facilitate this we used a resistor divider and used a battery voltage of 3 volts. The output from each of the SJ-353-100's is then sent to an SSM2135 op-amp.

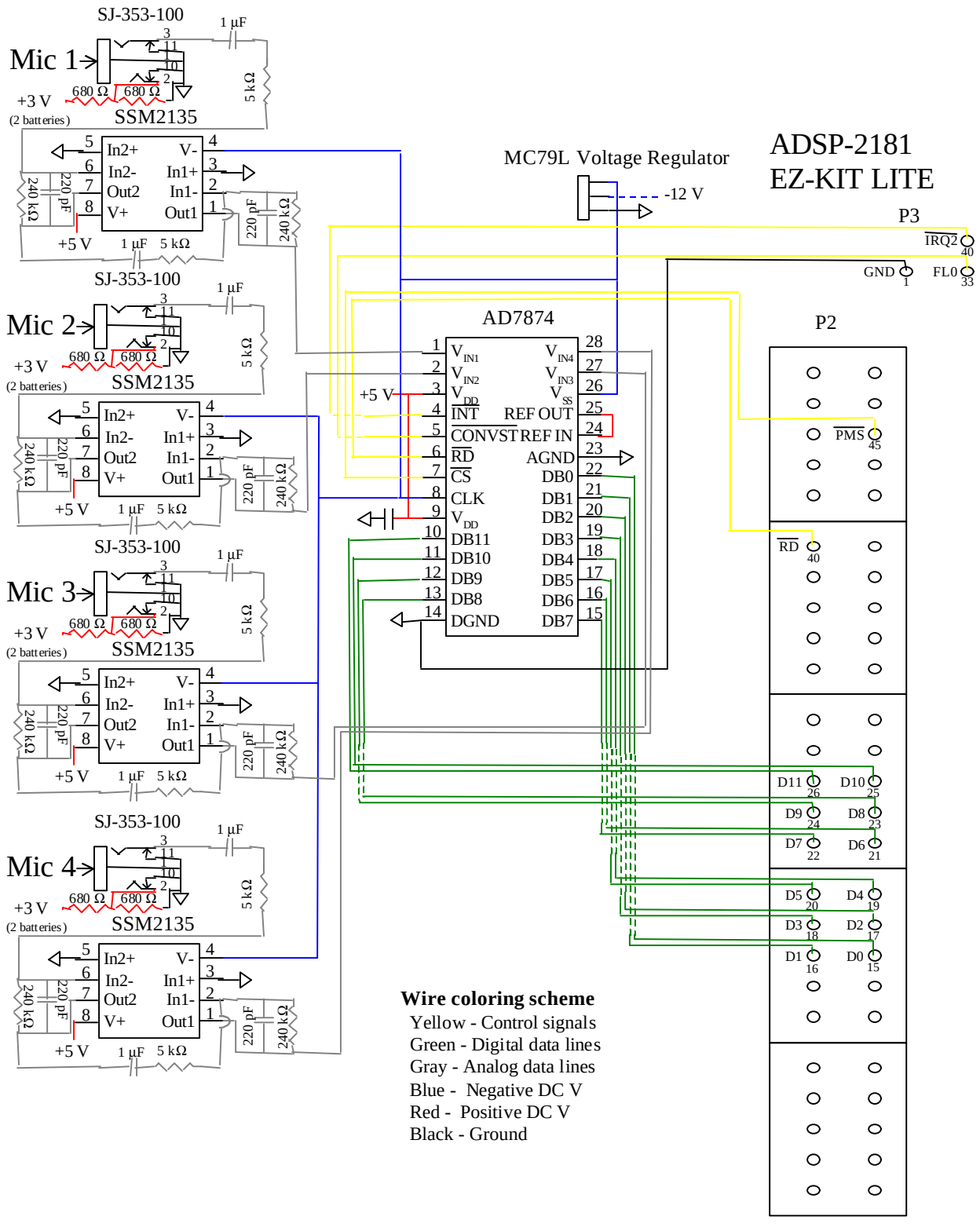

Figure 5.1 Wiring diagram for data acquisition

Component values used to determine amplifier gain were the values used on the on-board amplifiers of the ADSP-2181. Unfortunately, the gain of each amplifier was not enough. Instead, we cascaded two stages with the aforementioned components on a single chip to give us desired gain (~ 66 dB). As a consequence of the high gain, the 120 Hz component in the DSP supply leaked through significantly to the microphone inputs. The power supply used for the DSP board had to be changed to the regulated supply that the rest of the chips were using.

5.4 Data Acquisition using the Computer

Once the hardware is set up, we need a program that will take in data samples from the ADSP-2181 and make the input available to a program written in MATLAB. The ADSP-2181 is packaged as a kit which easily allows specialized programming. Connecting the ADSP-2181 directly to a PC does not allow us much in the way of programming or debugging options. Fortunately, the ADSP-2181 ICE contains a hardware interface between a PC and the ADSP-2181 which allows us to program and debug the ADSP-2181. The ADSP-2181 ICE board connects from the ADSP-2181 to a serial port on a PC. A program was written that takes in 2048 samples from each of the 4 microphones. These samples are then taken into MATLAB for processing.

The program gets the values from the A/D board in a simple manner. First, the program sets flag FLG0. FLG0 is connected to CONVST* on the AD7874, which puts the track-and-hold amplifier into hold mode and starts converting each of the 4 input channels sequentially. Then, the IRQ2* flag is set, which sets INT* on the AD7874. This starts the data transfer process between the AD7874 and the ADSP2181. RD* and PMS* are then enabled on the ADSP-2181, and the value in D0-D12 is stored in data memory in

the ADSP-2181. RD* and PMS* are enabled 4 times, 1 for each input, and then the cycle is done. The next set of 4 input data points gets collected by doing this whole procedure over again (starting with FLG0).

Chapter 6

Adaptive Array Considerations

6.1 Introduction

In our GSC adaptive array, there are many things to consider besides the adaptive noise-canceling filters. One problem is that of speech detection. In a typical conversation the user is speaking 40% of the time. Considering the car environment, we need to find speech activity algorithms robust to car noise as well as traffic noise. Also, there needs to be a predetermined standard as to what the array should do when there are multiple speakers present. Since the performance of the MUSIC algorithm is critical to system performance, other source estimation methods should be examined. Another factor to consider is how quickly the MUSIC algorithm should adapt to changes in DOA angles. We will look at this problem first.

6.2 Output Tracking to DOA

The output of the beamforming section of the GSC is written as in (2.8) and (4.1)

$$y(t) = \mathbf{w}^H \mathbf{x}(t). \tag{6.1}$$

We should consider how fast we should change the weights (which are determined from the DOA estimate in the MUSIC algorithm) with changes in the DOA. This is worthwhile because we don't want the output to vary drastically because a loud transient (traffic) or any other interruption occurred, or if by any other manner the DOA angle

from the MUSIC algorithm changed drastically. Since the angles are computed from the microphone output covariance matrix (2.9), it would be helpful to generate a 'running' covariance matrix that contains past statistics.

We will generate a sample covariance matrix at regular intervals. We will use an exponential forgetting factor α to update the covariance matrix such that

$$\mathbf{R}_t = \alpha \mathbf{R}_{t-1} + \mathbf{R} \tag{6.2}$$

where $\mathbf{R}$ is computed using (2.9) for the current sample frame. If we are concerned about the sample covariance matrix $\mathbf{R}_t$ getting large over time, one solution could be to multiply $\mathbf{R}$ in (6.2) by (1-α). Either way we can specify α to determine how fast to track the data. This is helpful in our GSC implementation. Since the filters don't adapt during speech, once the speech is finished we would like to get our best estimate of where the source is located. This estimate will be used immediately once speech stops, and will be used when the speaker talks again. Using (6.2), if we want more of an average of the speaker location for the last talkspurt, then α will be close to 1. If we want more of an estimate of where the speaker was last, then α will be closer to 0.

6.3 Speech Activity Detection

For our covariance matrix updating, we need to know whether we have speech or not; if we don't have speech present, we don't want to update the covariance matrix. But we want to make sure that speech is not cut off, so we would like to have some different estimators so that we have a 'failsafe' if one of them doesn't correctly predict speech. Also, if speech isn't detected in one time period we shouldn't assume there is no speech.

Instead, we should use some criterion that accounts for the fact that speech wasn't detected for 2 or 3 consecutive intervals. Since normal conversations have one speaker talking for at least 3 or 4 seconds at a time, there won't be this 'flickering' of speech activity. So for our speech detection algorithm we will use a few detectors and will output whether the source covariance matrix should be updated (speech present) or whether there should be no updating (speech absent).

One method of speech detection is to generate cepstral coefficients for a number of utterances, and then compare these coefficients to the speech by using a distance measure, and set a threshold to determine if speech is present or not. This distance measure is termed the Mahalanobis distance, d_{MCEP}, and is defined as

$$d_{MCEP} = \sqrt{(\mathbf{c}_i - \mathbf{c}_r)^T \mathbf{V}^{-1} (\mathbf{c}_i - \mathbf{c}_r)} \tag{6.3}$$

where $\mathbf{c}_i$ and $\mathbf{c}_r$ are feature column vectors containing the cepstral coefficients of the input signal and a reference to be compared to, respectively, and $\mathbf{V}$ is the covariance matrix of the feature vector. The weighting matrix $\mathbf{V}$ is necessary because the variances of cepstral coefficients tend to decrease for higher order coefficients.

We need to find ways to minimize the computational burden of (6.3). This is because we are going to compute $\mathbf{c}_i$ and $\mathbf{V}$ for each 400-sample frame. Tohkura [22] assumes that the off-diagonal terms of $\mathbf{V}$ are small compared to the diagonal terms, so they can be ignored. Now $\mathbf{V}$ just contains the variances of each of the cepstral coefficients. Also, Juang *et al* [23] shows that cepstral coefficients of an order higher than around 10 have high variability among similarly spoken words, so these terms are considered noisy and can be thrown out. Therefore, we will only need 10 cepstral

coefficients, and the matrix **V** is just a 10 by 10 diagonal matrix. This technique has been used to determine whether a given segment was speech or not [24], and we would like to examine its effectiveness in our situation. Kobatake [24] compared the input segments to 5 vowels and recorded the minimum cepstral distance, computing the cepstral coefficients from LPC coefficients. Furui [25] has shown that there is very little difference in performance when the cepstral coefficients have been computed from LPC coefficients or from taking the real cepstrum of the DFT, with slightly better performance from taking the real cepstrum of the DFT. Therefore, we shall compute cepstral coefficients by taking the real cepstrum of the DFT for each 400-sample frame.

Another method of speech detection involves pitch detection. A simple form of pitch detection involves using the short term autocorrelation function

$$r_s(\eta) = \frac{1}{N} \sum_{n=\eta+1}^{N} x(n)x(n-\eta) \tag{6.4}$$

where N is the number of time samples. One can look at r_s for different values of η, and if there are significant peaks at integer multiples of any of the η values, then this is the period estimate of the input sample. One problem with this is that any periodic sound will be classified as speech. Since speech is periodic only in the short-term, long-term periodicity can be detected by comparing the period at many closely-spaced frames and if it is similar for most of the frames, then it will be classified as non-speech.

A more complicated form of pitch detection involves using a least-squares periodicity estimator [26]. This method calculates the periodic component, $s_0(i)$, of the signal $s(i)$ as

$$s_0(i) = \sum_{h=0}^{K} \frac{s(i + hP_0)}{K}, \qquad 1 \le i \le P_0 \tag{6.5}$$

where P_0 is an estimate for the pitch period, and K is the number of periods of $s_0(i)$ in the sample frame. The pitch period P_0 is varied over a certain range (pitch frequencies mostly range from 80-400 Hz), and the goal is to find the $s_0(i)$ which minimizes the mean-square error of the actual signal and periodic signal,

$$\sum_{i=1}^{N} \left[s(i) - s_0(i) \right]^2 \tag{6.6}$$

where N is the number of samples in the sample frame. The normalized periodicity measure takes into account biasing for larger values of P_0 and the energy in each frame. Windows can also be used on the data to emphasize the 'cleaner' portion of the signal. In the end, we have a measure that gives a periodicity value from 0 to 1 for different pitches, allowing us to define a threshold value which determines whether there is a periodic signal or not. These three speech detectors will be evaluated in Section 7.3.

6.4 Source Estimators

We would like to have other different types of methods to estimate the number of sources since we do not want our estimate to be wrong. For example, if our estimator decided that there were 1 source instead of 2, our DOA estimator (MUSIC for this thesis) will pick an angle which corresponds to one of the two sources, and it will choose the more powerful source in that instant. On the other hand, if the estimator decided there

were two sources instead of one, then the DOA estimation from the MUSIC algorithm will degrade somewhat, depending on how much noise is in the operating environment.

Considering the powerful nature of DOA estimators, it would be beneficial to use other DOA estimators which don't need the number of sources to compute arrival angles, which in turn can give us the number of sources using peak-picking. The two estimators that will be used to accomplish this are Capon's Minimum Variance Estimator [2], and the linear prediction method [3].

Capon's Minimum Variance Estimator is based on the weight vector found in (4.3). Substituting this weight vector into the expression for the array output power, (4.2), we get

$$P_C(\theta) = \frac{1}{\mathbf{a}^H(\theta)\mathbf{R}^{-1}\mathbf{a}(\theta)} \ . \tag{6.7}$$

The linear prediction method is derived from one of the array sensor's outputs being predicted as a linear combination of the other sensors' outputs. The prediction coefficients are selected to minimize the mean square error. Note that we are only interested in how many significant peaks there are in these algorithms. Unlike the MUSIC algorithm, these algorithms give us an estimate of the power arriving from each direction, which is useful if we need to determine the relative loudness of each speaker (i.e., a situation where we need to focus on the louder speaker). Also, it is important to note that the linear prediction method is a higher resolution estimator than the minimum variance estimator, meaning that it can resolve more sources with more accuracy.

Now we have 3 methods of source detection: 1) MDL or AIC, 2) linear prediction, and 3) minimum variance estimation. The performance of these three methods will be compared in Chapter 7.

Chapter 7

Results

7.1 Introduction

Previously we have discussed all aspects of the adaptive array that will be used in the car environment. In this chapter we will determine the effectiveness of the array system, but we will have to simulate the car noise environment, since we currently cannot power up the array hardware inside the car. Also, though there may be multiple speech sources, we will examine the situation where we focus on a single speech source, and try to reject everything else. We will first examine three critical components of the array system: the DOA estimator, the source estimator, and the speech activity detector. Results for these three subsystems will be examined separately, and then we will examine the results for the complete array system.

7.2 MUSIC DOA Results

This section evaluates the performance of the MUSIC algorithm in estimating DOA's in the 1 and 2 speech source scenes. Tests were done with the sources 50 cm and/or 80 cm from the array. These distances are near the distance limits that the driver will be from the center of the microphone array, if the array is positioned at the top center of the front dashboard for a typical car. Tests were also done using 400 snapshots, for two reasons. The first reason is that in the final implementation, this will be the number of snapshots used for the covariance matrix generation for each frame. The second

reason is that there is dominant speech for the 400 snapshots in all cases. We could have used all 2000 snapshots that were captured, but most of the other parts do not have speech in them. Using 400 snapshots of speech for each case gives us a good idea of how the final implementation will perform. The final characteristic of all our tests is that the array interelement spacing was set to 11 cm, which has been used for most of this thesis.

The first case that we will examine involves 1 source, with noise added. The noise floor in the room was found to be an average of 25 dB below the power in the 400-snapshot long segments used. DOA results for a variety of arrival angles, and at distances of 50 and 80 cm from the array are shown in Table 7.1. Results were generated

Table 7.1 MUSIC DOA results, 1 source, 25 dB SNR

	10°	40°	70°	100°	130°	160°
50 cm	15.1°	49.1°	76.0°	100.4°	125.9°	152.2°
80 cm	23.8°	46.3°	71.7°	102.5°	123.3°	152.9°

from the MUSIC estimations by using peak-picking and the ad hoc restrictions mentioned in Section 3.7. The results seem to be biased towards 90°, but the estimations are reasonably close, except for angles close to the array line. As the source was prerecorded data played through a computer speaker, the error due to speaker misalignment was at most ± 3°. The results at 50 cm seem to be better than the results at 80 cm. The reason why the arrival angles are biased towards 90° may be due to something we haven't considered since Chapter 2: near-field effects.

We should compare the deviation from the assumed far-field hypothesis of waves coming from 50 cm and 80 cm from the array. We could make the comparison in a manner similar to Figure 2.4, but for both cases here the 2 lines would be closer together.

Instead we will make a comparison of the relative time-delay difference (in samples) between the far-field assumption and the actual cases at 50 and 80 cm. Results are shown in Figure 7.1. As can be seen from this figure, the most that the far-field assumption is

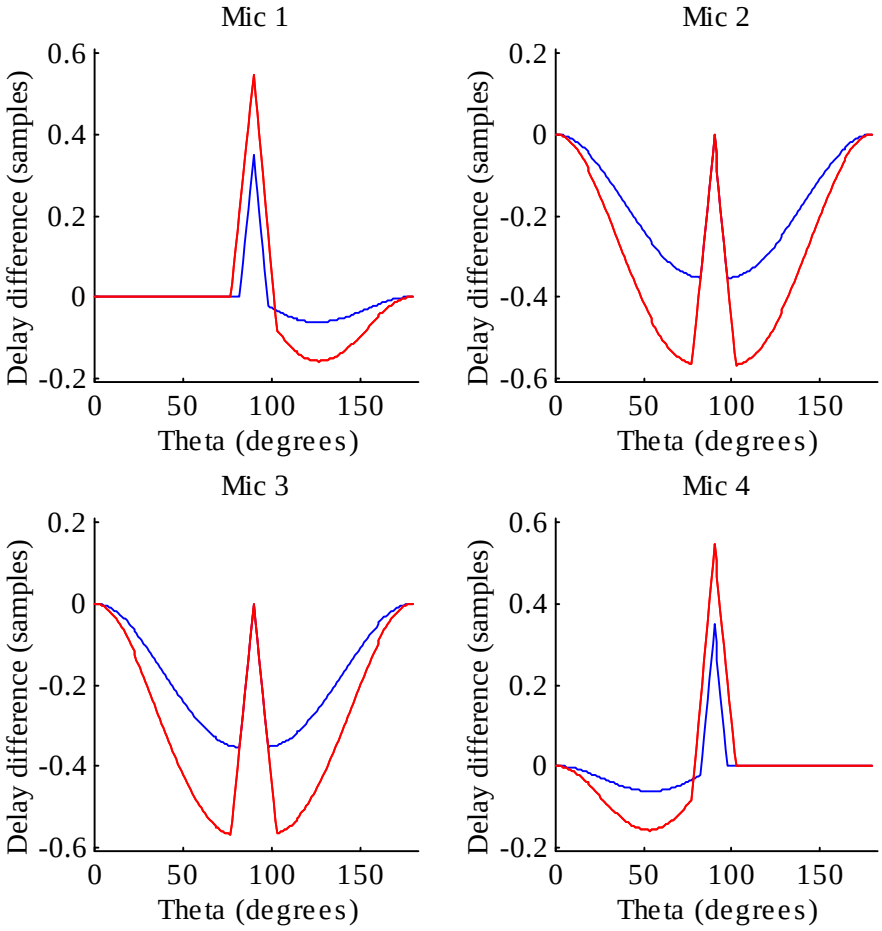

Figure 7.1 Relative time delay difference (in samples at 8 kHz) between far-field hypothesis and (red) 50 cm (blue) 80 cm distance from array with *d*=11 cm

off by is about 0.55 samples at 50 cm and 0.35 samples at 80 cm. The area where these values are most off from the far-field assumption is around 90°, but we get our best estimates there. That is partly because that region is also where the change in delays

between closely spaced angles vary the most, as discovered in Table 3.8, and retabulated in Table 7.2 with the delays expressed in terms of samples for comparison.

Table 7.2 Time (in samples at 8 kHz) for speech to reach microphones compared to microphone 1 in the far-field assumption

DOA	0	5	10	20	30	40	50	60	70	80
Mic 2 (samples)	2.6	2.5	2.5	2.4	2.2	2.0	1.7	1.3	0.9	0.4
Mic 3 (samples)	5.1	5.1	5.1	4.8	4.5	3.9	3.3	2.6	1.7	0.9
Mic 4 (samples)	7.7	7.7	7.6	7.2	6.7	5.9	4.9	3.8	2.6	1.3

We can make comparisons using Table 7.2 and Figure 7.1. Let's take a look at when the source arrives at 40°, 50 cm from the array. From Figure 7.1, the far field assumption is off by 0.35, 0.35, and 0.12 samples into microphones 2, 3, and 4, respectively. Taking these values and subtracting them from the column in Table 7.2 corresponding to 40°, we get 1.65, 3.55, and 5.8 sample delays into microphones 2, 3, and 4, respectively. These delays closely match the 50° results for mic 2, the 40° results for mic 4, and in-between 40° and 50° for mic 3. So considering these near-field effects we should expect the DOA estimator to give us around 46° for the estimation. The estimate from MUSIC that we got was 50.3°, which is still a few degrees off, but we can see that the near-field effects definitely account for some of the inaccuracies in the final DOA result.

Near-field corrections can be made to improve the MUSIC estimate. Triangulation can be done using the current DOA estimate to determine how far the source is from the array, then correction factors can be applied to the current estimate, as done previously, to improve the DOA estimate. Additionally, we can help out the performance of the beamformer in the GSC by figuring out what direction the source is

coming from *for each microphone*. Now that we are in the near-field, the angle at which the source arrives into each microphone is different. This was shown in the previous paragraph as well. It would be useful to know if these corrections improve the beamformer output.

We are looking at the case with a 40° source 50 cm from the center of the array. The input into each of the microphones is shown in Figure 7.2. Since we do not have

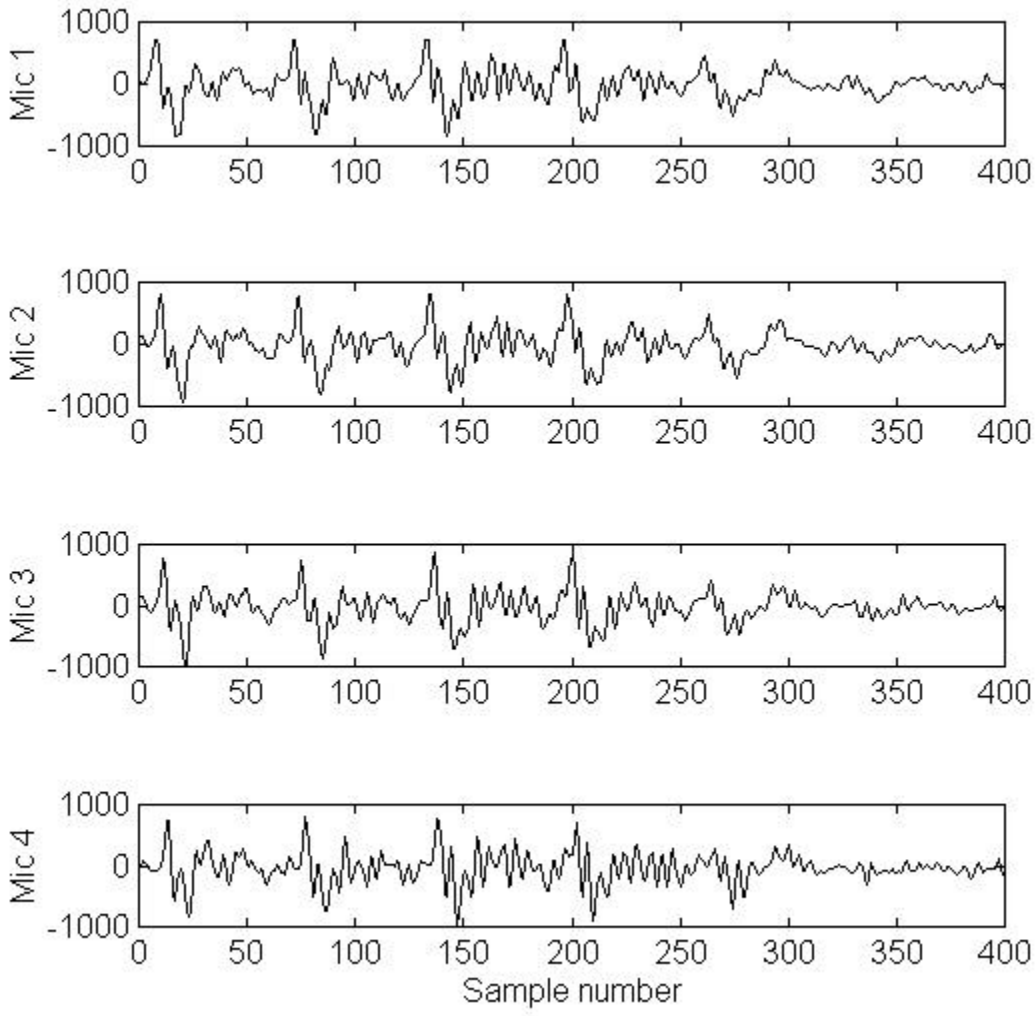

Figure 7.2 Inputs into array with speech source at 40°, 50 cm

perfectly matched microphone characteristics, the 4 input segments were gain-corrected (the power in each input segment was equal to the average of the 4 input powers). From this data it is seen that though the inputs are similar, there are many places where the data is significantly different for this 25 dB SNR scenario. This discrepancy is partly responsible for the precision in the DOA estimation. Since these inputs are not perfectly matched (taking into account the inherent phase delay of the source into each microphone), we should expect the beamformed output not to be very close to any one of the microphone inputs since it is taking an average of the inputs. Now we would like to see how different beamformed outputs compare to the input.

The three cases of beamformed output that will be examined are 1) phase-steered along the true DOA (40°) 2) phase-steered along the MUSIC DOA (48°) and 3) each microphone phase-steered with near-field corrections. These three cases are shown in Figure 7.3. The outputs look very close, and a comparison with the speech input into microphone 1 using the segmental SNR mentioned in Section 4.9 gave a 2.5 dB SNR for using near-field corrections and the MUSIC DOA, but gave us a 2.3 dB SNR when using the true DOA. This result will force us to use the MUSIC DOA estimate in the beamformer, not only because it performed better in this case over using the true DOA, but the additional computational complexity in using triangulation to pinpoint the source produces little if no benefits.

The performance of the MUSIC algorithm in estimating a single source in various noise conditions is shown in Table 7.3. We have modeled the car noise background by placing 3 speakers angled towards the wall behind the array to reflect the sound (these 3 speakers are situated about 6 inches from the wall), and the 4^{th} speaker to the right of the

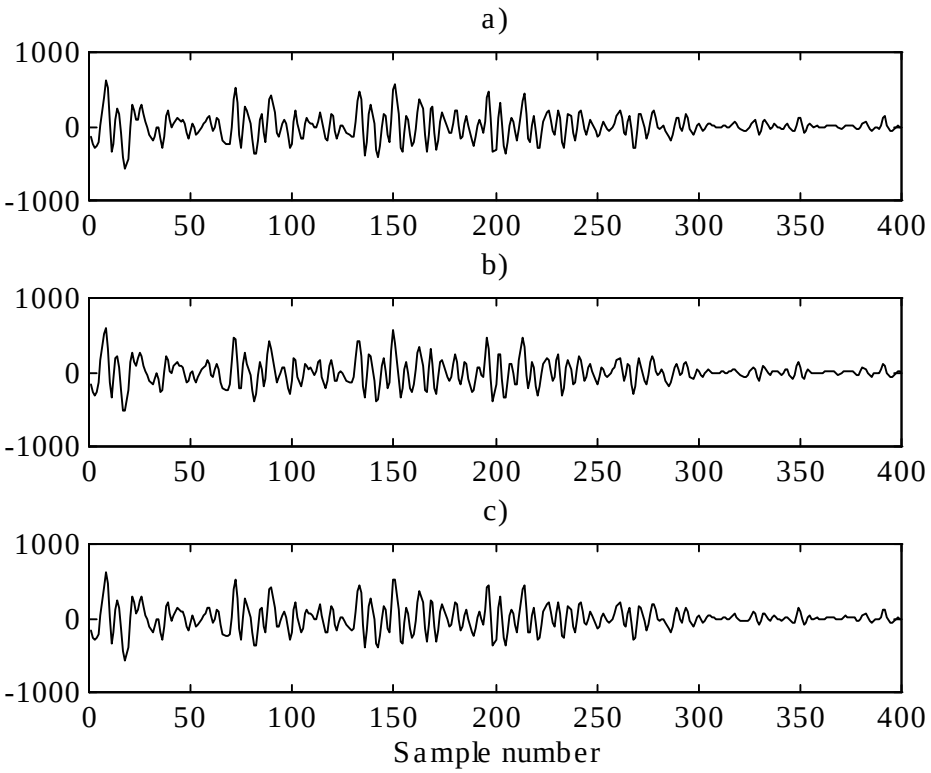

Figure 7.3 Phase-delayed output with mics steered to a) true DOA (40°), b) MUSIC DOA (48°), c) near-field corrected DOA (40°, 45°, and 50° for mics 2,3 and 4)

Table 7.3 MUSIC DOA results, 1 source, various SNR's

	SNR	10°	40°	70°	100°	130°	160°
50 cm	10 dB	15.0°	48.0°	75.8°	99.8°	126.2°	145.2°
	0 dB	25.3°	41.2°	74.1°	101.7°	122.3°	143.2°
	-5 dB	31.5°	48.2°	74.3°	88.7°	123.5°	142.0°
80 cm	10 dB	24.6°	49.5°	70.2°	98.8°	131.3°	159.5°
	0 dB	35.4°	42.8°	66.0°	102.0°	122.0°	138.3°
	-5 dB	48.0°	42.8°	62.5°	105.0°	119.2°	150.3°

array, pointing towards the wall. Results at 10 dB SNR are similar to those shown previously at a 25 dB SNR. Some of the errors in the estimation can be traced to near-field problems, but the problems at low SNR's is due to the predominance of the

somewhat directional noise sources. We can use Capon's minimum variance estimator to determine what the power in the noise field looks like. This is shown in Figure 7.4. It is

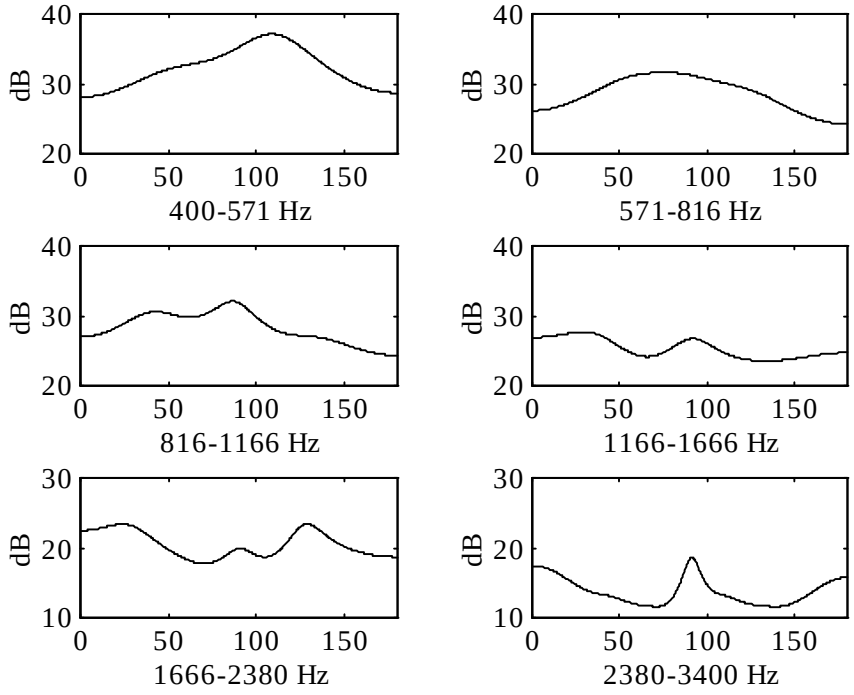

Figure 7.4 Noise field power using minimum variance estimator

seen that in the lower frequency bands that there is more power towards the middle of the array. There also seems to be a similar peak around 90° for the 816-1166 Hz and the 1166-1666 Hz bands. Comparing Figure 7.4 to the results taken from the car with 2 microphones, Figure 3.6 and Figure 3.8, we see that the noise here is less omnidirectional. Though part of this is due to the increased resolution of using 4 microphones instead of 2, we still have a fairly diffuse noise field for our experiment in this section.

Now let us look at how MUSIC handles multiple speech sources. Results with one source at 40° with both sources 50 cm from the array, and with one source at 80° with both sources 80 cm from the array, are shown in Table 7.4. We can see that

Table 7.4 MUSIC DOA results, 2 speech sources, 25 dB SNR

	50 cm			80 cm
40°, 10°	45.7°		80°, 20°	84.5°, 21.3°
40°, 70°	53.3°		80°, 50°	70.0°
40°, 100°	45.9°, 90.8°		80°, 100°	80.0°
40°, 130°	52.1°, 126.8°		80°, 130°	77.8°, 130.5°
40°, 160°	47.3°, 145.5°		80°, 160°	83.2°, 143.8°

MUSIC has difficulty resolving sources spaced 30° or less apart using 400 snapshots. Also, these results depend on a source estimator that correctly estimates the number of sources for the MUSIC algorithm. With sources spaced 50° or more apart, the performance is reasonable. The MUSIC DOA results with sources at 80° and 100° are shown in Figure 7.5. In the 571-816 Hz band it is shown that there may be 2 closely spaced sources, but the other bands do not give any indication for a second source. In the simulations in Chapter 3 the MUSIC algorithm was able to resolve closely spaced sources across 90°, but that is not the case here. The main reason for this is that under the near-field assumption, the inputs do not all have relatively negative delays for one source and positive delays for the other, as was true for the far-field assumption. The sources here are closer in terms of delays to the microphone inputs.

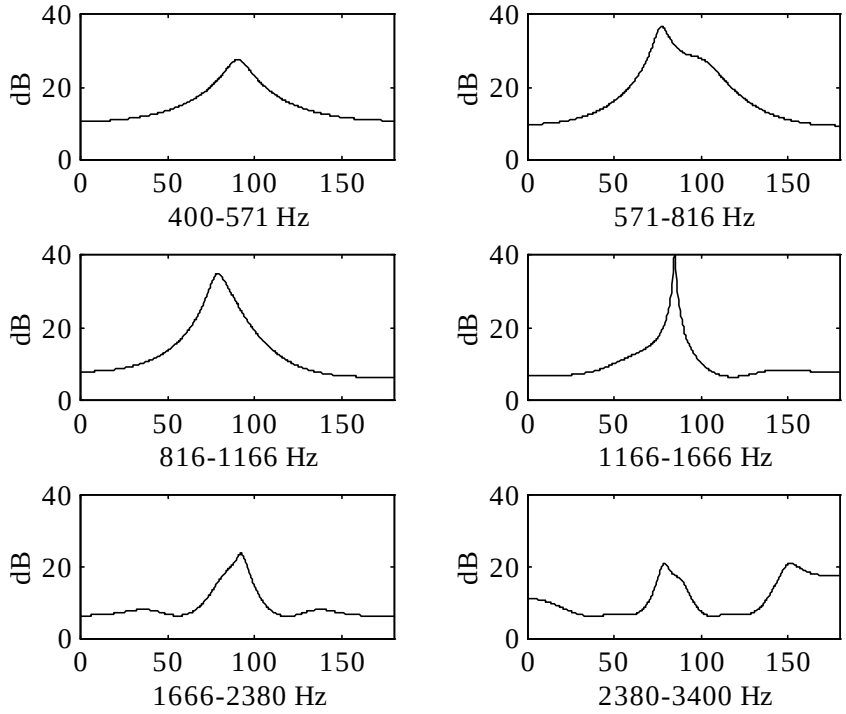

Figure 7.5 $P_{MUSIC}(\theta)$ with sources at 80° and 100°, 80 cm from array, 25 dB SNR

The performance of the MUSIC algorithm with 2 sources at lower SNR's is shown in Table 7.5. The performance with one source fixed at 40° seems to be better than when one source is fixed at 80°. Noting that speech is rarely lower in amplitude than the noise field, we get reasonable results when the sources are widely spaced apart with 1 source fixed at 40° and at a distance of 50 cm from the array. The reason that the results for the 80° source at 80 cm from the array are off is partly due to the extra reflections of the source caused by the extra distance. The MUSIC algorithm, though, is helpful in the two-source scenario, but we will not examine it any further because we will examine the GSC performance for the 1-source case only.

Table 7.5 MUSIC DOA results, 2 speech sources, 50 and 80 cm from array center (1st number for 10 dB SNR, 2nd number for 0 dB SNR, 3rd number for -5 dB SNR)

	50 cm			80 cm
40°, 10°	45.5° 46.2° 46.5°, 99.1°		80°, 20°	86.0°, 32.0° 91.8° 97.8°, 44.5°
40°, 70°	52.0°, 76.3° 52.3°, 96.8° 51.3°, 93.5°		80°, 50°	126.0°, 66.0° 93.3°, 44.2° 94.0°, 42.7°
40°, 100°	47.2°, 105.2° 49.7°, 108.3° 50.3°, 110.2°		80°, 100°	65.5°, 105.3° 86.8° 49.0°, 99.0°
40°, 130°	49.1°, 123.9° 48.3°, 115.3° 49.5°, 110.7°		80°, 130°	62.3°, 120.8° 60.3°, 115.0° 50.2°, 102.2°
40°, 160°	52.0°, 145.7° 58.0°, 144.8° 59.2°, 142.0°		80°, 160°	83.8°, 151.8° 79.8°, 140.8° 76.9°, 131.0°

7.3 Source Estimator Performance

This section evaluates the performance of the source estimators discussed in Section 6.4. The three methods that will be examined are the MDL criterion, linear prediction, and Capon's minimum variance estimation.

The first case we will look at is with 1 speech source incident on the microphone array. We will examine results at various arrival angles, and with the source at 50 cm and 80 cm from the center of the array (array interelement spacing = 0.11 m). These results were taken using 400 samples. Also, the speech was at an average of 25 dB above the noise floor of the room. The results for the three methods discussed are shown in Table 7.6. The MDL estimator provides poor results in this situation. This problem

Table 7.6 Source estimations for 1 source, 25 dB SNR, various DOA's, 50 and 80 cm from array center

	50 cm			80 cm		
	MDL	MVE	LP	MDL	MVE	LP
10°	3	1	1	3	1	2
40°	3	1	1	3	1	1
70°	3	1	1	3	1	1
100°	3	1	2	3	1	1
130°	3	1	1	3	1	1
160°	3	1	1	3	1	3

is due to the reflections off the walls in the room. As mentioned in Section 7.2, there is a wall 70 cm behind the microphone array, as well as a computer 1 m to the left of the array and a bookshelf 40 cm to the right of the array (40 cm from the wall). The MDL criterion relates an eigenvalue to a source even when the eigenvalue is a few orders of magnitudes smaller than another eigenvalue. Since in the car environment we can expect reflections such as the ones in this testing setup, if not more, the MDL estimator is a poor estimator for actual sources and will not be used.

On the other hand, Capon's Minimum Variance Estimator (MVE) performs flawlessly for the given data. As compared to the linear prediction method (LP), the MVE has less resolution, therefore it is less likely to resolve source reflections. The MVE, though, won't be able to resolve closely spaced sources as well as the LP method. With both the MVE and LP methods, we need to have some algorithm to determine which peaks are considered sources. After the peaks (in dB) are found by simple peak-picking (finding angles where the power at the angles immediately to the left and right is smaller than at the current angle), the average of the powers at all angles is computed. Then this value is averaged with the highest peak value to determine the threshold value

for peaks to be qualified as sources. This method eliminates small waves in the output which don't correspond to sources, but gives a reasonable threshold for any situation. It is useful to note that this technique is a little more likely to resolve sharp peaks than broad peaks, but that has little effect on the results here.

The LP method works fairly well for both cases. A comparison of the LP results for these 2 cases with a DOA of 160° is shown in Figure 7.6. The power levels of the 50

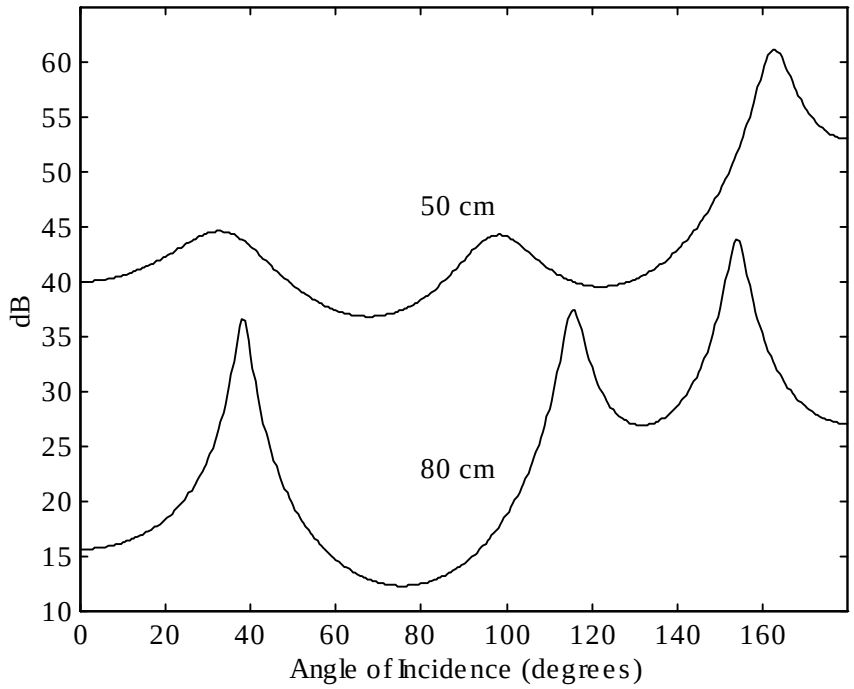

Figure 7.6 LP estimator DOA results with source 50 cm and 80 cm from array

cm case were adjusted such that both curves were not on top of each other. Here we see that moving the source back causes more reflections for the LP estimator to pick up. All 3 peaks in the 80 cm case would be considered sources no matter how we chose to pick the peaks. So with the 1-source, low noise case, the MVE criterion produces the best results of the three cases examined.

Now we want to determine the performance of the source estimators at lower SNR's. Noise was simulated using the speaker configuration mentioned in the previous section. Results are given for 10, 0, and -5 dB SNR's. The results using the MDL, MVE, and LP methods are shown in Table 7.7. Once again we see the poor performance of the

Table 7.7 Source estimations for 1 source, various DOA's, 50 and 80 cm from array center (1st number for 10 dB SNR, 2nd number for 0 dB SNR, 3rd number for -5 dB SNR)

	50 cm			80 cm		
	MDL	MVE	LP	MDL	MVE	LP
10°	3 3 3	1 1 1	1 1 1	3 3 3	1 1 2	3 3 2
40°	3 3 3	1 1 1	1 3 2	3 3 2	1 1 2	1 1 2
70°	3 2 3	1 1 1	1 1 1	3 3 3	1 1 1	1 3 3
100°	3 3 3	1 1 1	2 2 2	3 3 3	1 1 1	1 2 1
130°	3 3 3	1 1 1	1 3 1	3 3 3	1 2 2	1 3 2
160°	3 3 3	1 1 1	1 1 1	3 3 3	1 1 1	3 2 2

MDL estimator and the excellent performance of the MVE estimator. The MVE estimator was flawless for the 50 cm source distance. For the 80 cm distance, the MVE estimator was off mostly for the -5 dB SNR scenario. The performance of the LP estimator seems to be worse for the longer source distance. From these three performance measures in the 1-source scenario, the MVE estimator works the best, and the overall source estimator would consist just of this one. Now let us look at the multiple source case.

Source estimator results for 2 sources and no noise are shown in Table 7.8. The

Table 7.8 Source estimations for 2 sources, 25 dB SNR, various DOA's, 50 and 80 cm from array center

	50 cm					80 cm		
	MDL	MVE	LP			MDL	MVE	LP
10°, 40°	3	1	3		20°, 80°	3	2	3
70°, 40°	3	2	3		50°, 80°	3	1	2
100°, 40°	3	1	1		100°, 80°	3	1	2
130°, 40°	3	1	1		130°, 80°	3	1	1
160°, 40°	3	1	2		160°, 80°	3	1	2

results here are poor. The MDL estimator is useless again. The only real difference in the data here as compared to the data with 1 source (Table 7.6) is in the LP estimator at 50 cm. As a result of the software used to collect the data, it was extremely difficult to get a 400 snapshot segment of data where both sources were prevalent, as well as being able to get the software to collect the data at similar times for all variations of arrival angles. This in turn makes it difficult for the source estimators to supply consistent data.

Though in this thesis we will only be examining final results using 1 source and therefore do not need to be concerned with estimating the number of sources, we have discovered that estimating multiple sources is a fairly complicated issue. Not only do we have to be concerned that echoes will not be mistaken as sources, but also we need to find some 'signature' left by each source so that we can track individual sources, and determine whether they are active or inactive. The results in this section are for a given segment of speech where we know that speech is present at the time we are estimating the sources. Unfortunately, that is not the case in real time, and in a typical conversation we can expect the speaker to be inactive 40% of the time. For the rest of this thesis we will assume that there is 1 speaker present.

7.4 Speech Activity Detector Performance

This section evaluates the performance of the speech activity detectors discussed in Section 6.3. The three methods discussed are the minimum cepstral distance estimator, pitch estimation using the autocorrelation method, and pitch estimation using the least-squares periodicity estimator.

The minimum cepstral distance estimator requires a speaker-dependent system, because the cepstral coefficients vary between speakers. Also, it is necessary to have samples of prerecorded speech to compare speech segments. Even with these parameters, the performance of the minimum cepstral distance estimator is weak for low SNR's. At high SNR's, this estimator is useful to distinguish other periodic and quasi-periodic sounds from speech.

The autocorrelation method is the simplest way to estimate the pitch period. The final algorithm using this method starts by computing the autocorrelation function for a segment of each sample frame (400 samples). Then the peaks are found by finding which samples (η) have samples to the immediate right (η+1) and left (η-1) which are smaller in value. The 4 largest peaks are taken, and then it is determined if any of these peaks are harmonically related, specifically if any peaks are between 0.49 and 0.51 times another peak. These values were chosen to maximize detection of speech and minimize detection of noise as speech. Performance of this method, though, is also poor at low SNR's. Let's examine the performance of the autocorrelation method for a speech segment. Data was taken from a Dodge Caravan, with the windows open and traveling at 25 mph. The

results are shown in Figure 7.7. Speech is present at 3 distinct times, starting at about 1 second, 6 seconds, and 10 seconds. A decision of '1' corresponds to speech

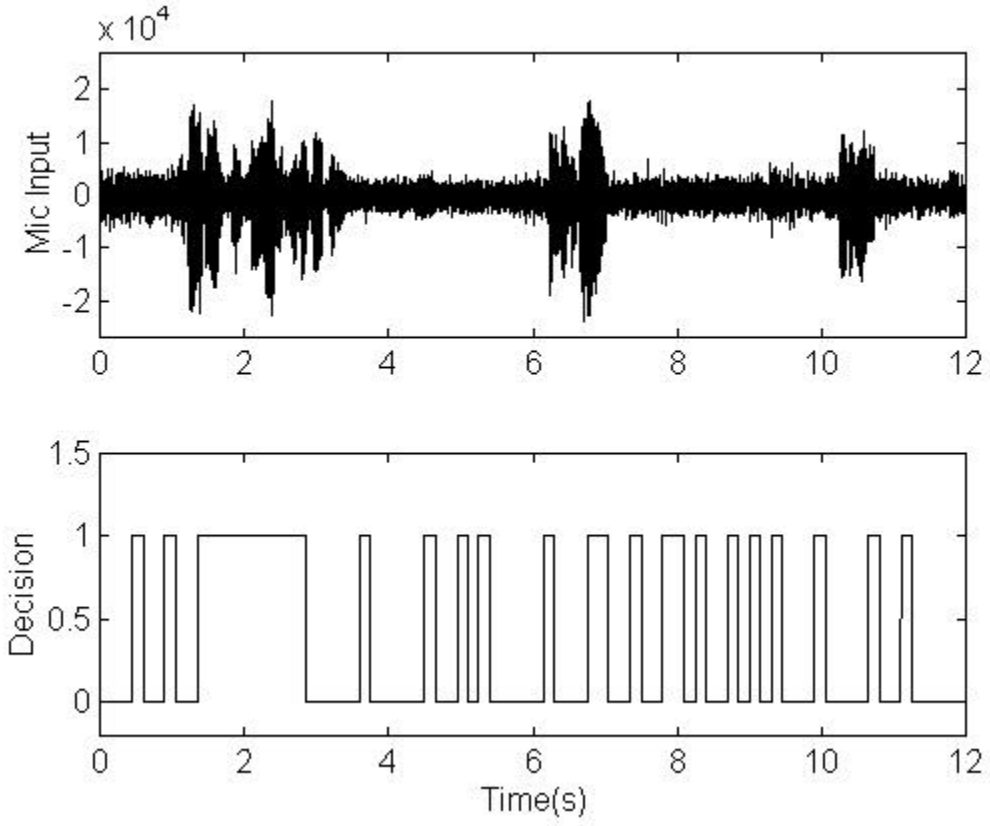

Figure 7.7 Speech detector output using autocorrelation method, car at 25 mph, window open

detected. It is seen here that the autocorrelation method is a poor estimator in this scenario. The last two speech segments are hardly detected, while noise is frequently mistaken as speech. This method cannot be used as a speech detector in the car environment.

The least-squares periodicity estimator (LSPE) can be used to make an effective speech detector. In the scheme developed, three periodicity values are obtained for the three 400-sample intervals that comprise the current frame (1200 sample frame). A

threshold needs to be determined so that all periodicities below a certain value are assumed to be due to noise. This value will be subtracted from all periodicity values, and values below the threshold are zeroed, so that we have a normalized measure. It was determined that a threshold of 0.35 maximizes speech detection and minimizes noise detected as speech. The three periodicity values are summed to generate the periodicity value for the frame. Now there needs to be a threshold that determines whether the frame is noise or speech. A threshold of .24 was determined to provide good results. These thresholds were determined subjectively; there is no theoretical basis for these values. The performance of the LSPE using the same data as in Figure 7.7 is shown in Figure 7.8. The LSPE correctly identifies the speech segments and does not mistake noise for speech. The LSPE does not identify every part of each speech segment, though. This is because the SNR for the unvoiced sounds are usually very small.

The goal of the speech detectors is to tell us when there is noise. The GSC implementation described in Chapter 4 adapts its filters during noise-only segments. Therefore we want to find the best way to use the speech detector estimates to determine when the filters should adapt. The LSPE gives a 'conservative' estimate of speech, that is, it doesn't detect all of the speech but does not mistake noise for speech often. Therefore, the filters should adapt when speech has not been detected for some number of frames greater than 1. If the number of frames before adaptation is too high, then we may be missing out on time that may help improve the noise estimate if speech gets detected shortly thereafter. Also, DOA estimates may be incorrect if no speech is present, though with a reasonably high forgetting factor in (6.2) the effects will be small. If the number

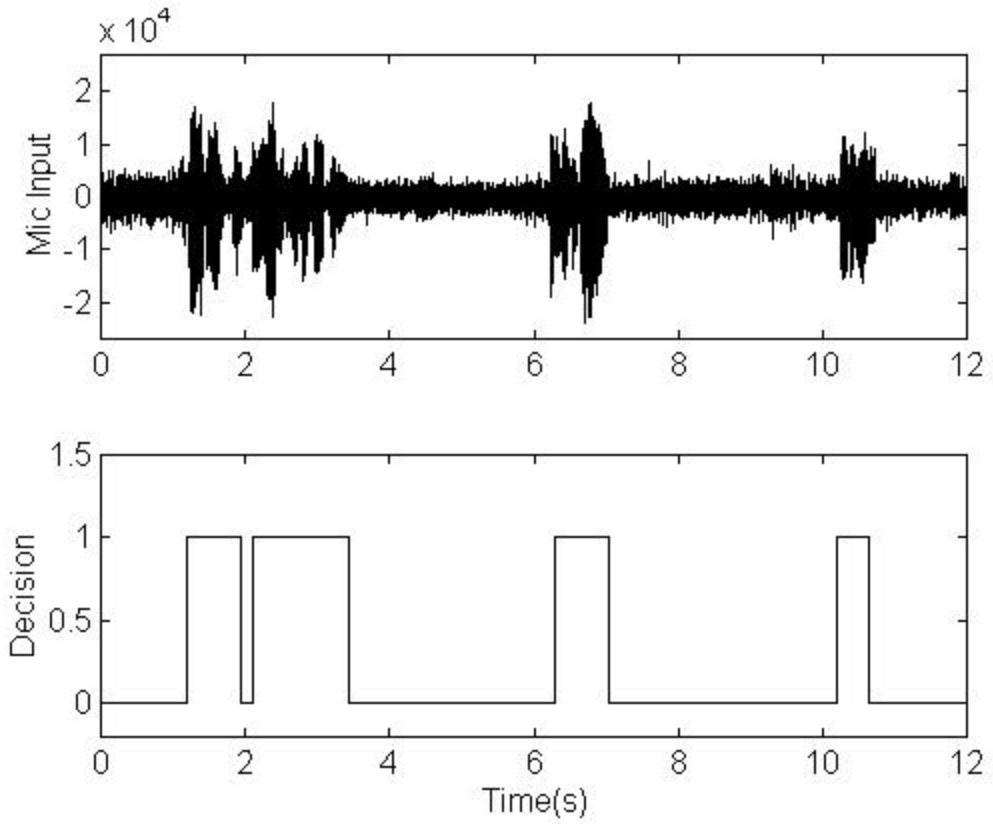

Figure 7.8 Speech detector output using the LSPE, car at 25 mph, window open

of frames before adaptation is too low, and filters adapt during speech, then not only will there be echo effects and cancellation of the current speech, the adaptive filters' coefficients will need to be corrected before the next time speech is detected.

With these considerations, it is determined that the filters will adapt at the 4^{th} frame after the last speech has been detected. Also, once speech has been detected, the adaptive filter coefficients used will be those from the second frame before the current (speech detected) frame. Results using this adaptation scheme will be examined in Section 7.6.

7.5 Performance of Array System, Noise Only

Before we examine the output of the array system to see what improvement we will obtain using the phase-delay and time-delay GSC's, we need to determine parameters for the array system. First we need to determine the best way to simulate the car noise environment for our in-lab experiments. Next we need to determine the optimal orders for the adaptive filters in both GSC's. Once we have determined the optimal orders, then we can determine what adaptation constant μ should be used for optimal performance of the NLMS adaptive filters. With these parameters in place, we can show noise power improvement results using both GSC's.

We were able to capture 2-channel car noise data, and we have used one of these channels many times in this thesis. We need to find a reasonable way to use this 2-channel data to simulate the noise field in the lab. The desire to accurately reproduce the sound field from the car warrants our using high-quality speakers. Two Inifinity speakers, with a frequency response range of 75 Hz - 20 kHz were used to reproduce the sound. The sound from the computer was sent from the PC to a Kenwood receiver and then to the speakers. One of the speakers was positioned 23 cm from the back wall at an angle of 135°, and 55 cm to the left of the array center. The other speaker was positioned 18 cm from the back wall at an angle of 45°, and 25 cm to the left of the array center. The back wall is 70 cm from the array center. The speakers are pointed towards the back wall to help spread out the sound. This setup will be used to produce the noise field for this section and the next.

We can now collect data from the noise field to determine the optimal filters for our sidelobe cancelers. A 4.5-second segment of noise data will be used for these

experiments. The noise data was acquired when the car was going 55 mph with the driver's window open. Also, the beamformer in all cases will be focused at an angle of 135° since this will be the arrival angle of the speech source. First we will examine the time-delay GSC results. Figure 7.9 shows noise improvement compared to beamformer output with different FIR orders, using μ=1 for fastest convergence. Figure 7.9 shows

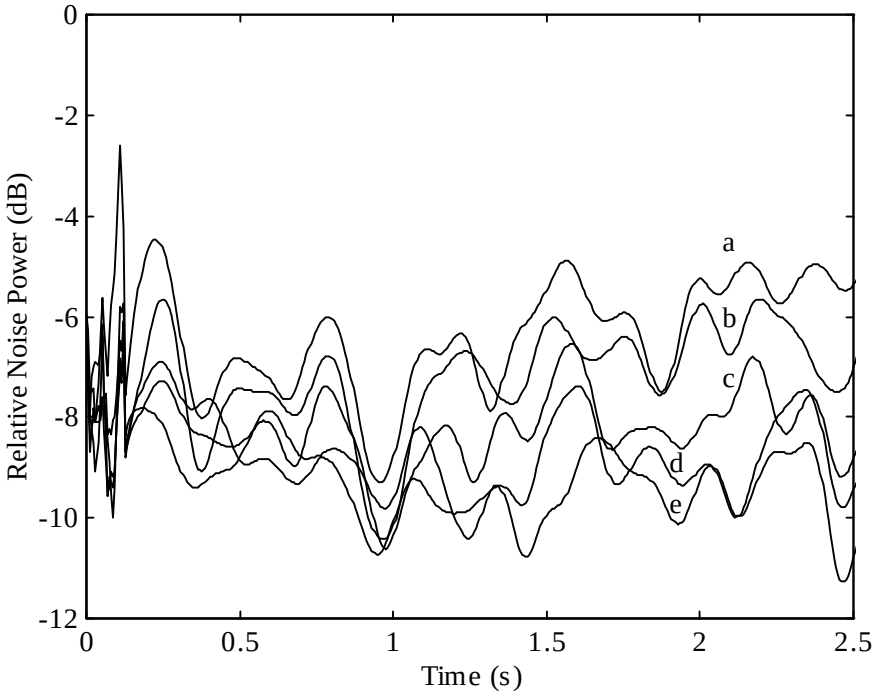

Figure 7.9 Time-delay GSC noise improvement after beamforming (LMS, μ=1) FIR order = a) 10, b) 20, c) 40, d) 80, e) 120

that using higher orders produces better results, though the overall difference between using an order of 10 and 120 is only 4 dB. The noise improvement compared to the beamformed output is small enough that we need to get the best results possible. For this reason, we will use a filter order of 80. A similar graph for the phase-delay GSC implementation is shown in Figure 7.10. The improvement here also increases with filter order, and the improvement for any order is better than in the time-delay case. We will

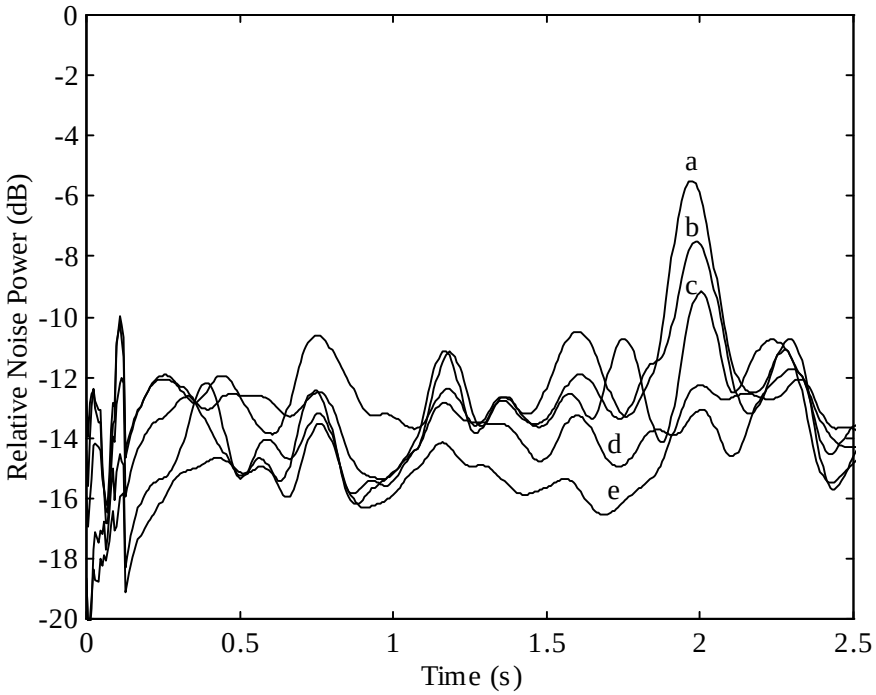

Figure 7.10 Phase-delay GSC noise improvement after beamforming (LMS, μ=1) FIR order = a) 10, b) 20, c) 40, d) 80, e) 120

also use a filter order of 80 for this case as well. Though the improvement with using an order of 120 is on average 0.5 dB better, we want some limit on the order, especially considering that the lower subbands require longer times to converge when the order increases.

We have decided to use FIR filter orders of 80 for both the time-delay and phase-delay GSC. Now we need to determine what choice for μ we should use for the NLMS adaptive filters. We will first examine the time-delay GSC with μ=1. Results before and after weight fixing at different times are shown in Figure 7.11. The performance once the weights are fixed is for the most part worse than if we did not use the sidelobe canceler.

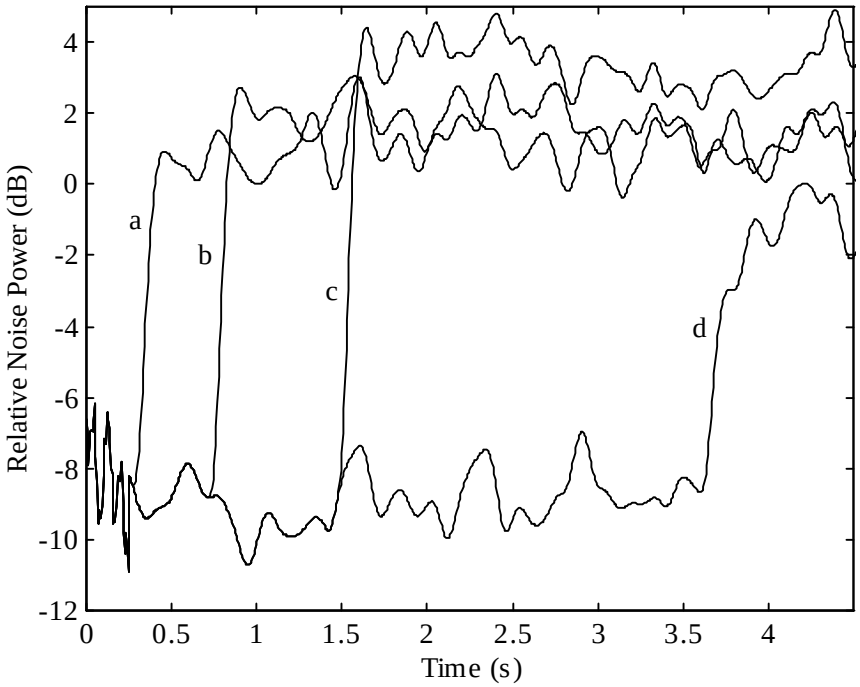

Figure 7.11 Time-delay GSC noise improvement (LMS, FIR order=80, μ=1), weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

We can improve on the performance when the weights are fixed after a small number of snapshots by decreasing μ, as was done in Section 4.6. We will use μ=1 for the first 1200 snapshots, and then use μ=0.1 for the rest of the snapshots up until weight fixing. Figure 7.12 shows that this scheme produces much better results, especially when the weights are fixed after a small amount of time. The last case, case d), shows us that after a few seconds we can expect the performance after weight fixing to give us about 6 dB of noise power improvement. Therefore, our scheme for the time-delay GSC will be to use μ=1 for the first 1200 snapshots and then set μ=0.1. From Figure 7.12 we can expect to see between 3 and 7 dB improvement in output noise power compared to the beamformer output, depending on how long the filters adapt.

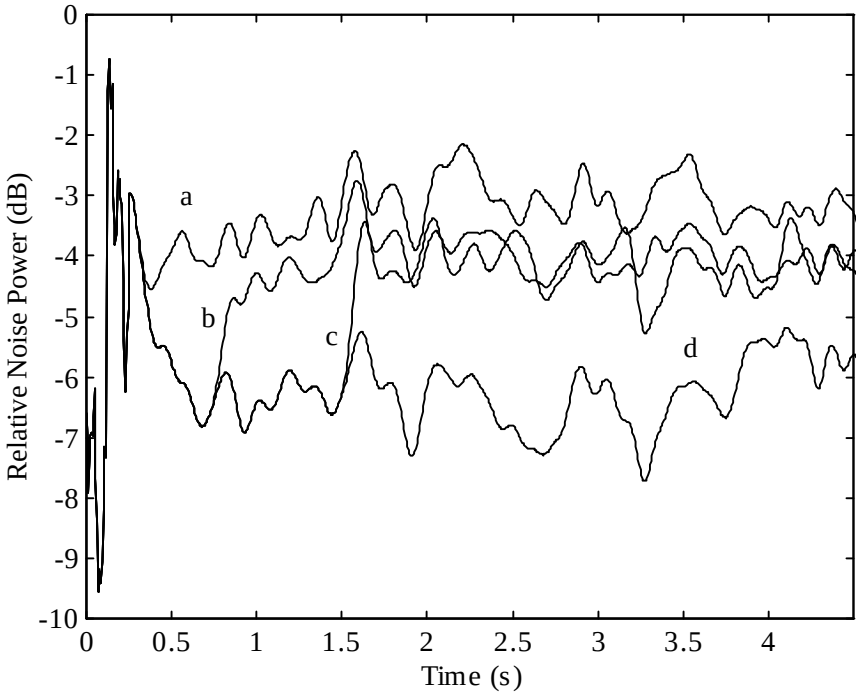

Figure 7.12 Time-delay GSC noise improvement (LMS, FIR order=80, μ=0.1) with weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

We need to go through similar tests to decide the best choice of μ for the phase-delay GSC. Figure 7.13 shows noise improvement using μ=1 with weight fixing at different times. Comparing these results with those for the time-delay case, we see about 5 dB improvement in the phase-delay case when the weights are adapting, though the results when the weights are fixed are about 5 dB worse. We will decrease μ to improve the performance after weight fixing. As before, it would be advantageous to use μ=1 for the first 1200 samples to speed up convergence. Results using μ=0.1 are shown in Figure 7.14. These results are significantly better after weight fixing than the results using μ=1. From these results we still have about 4 dB of performance loss once the weights are

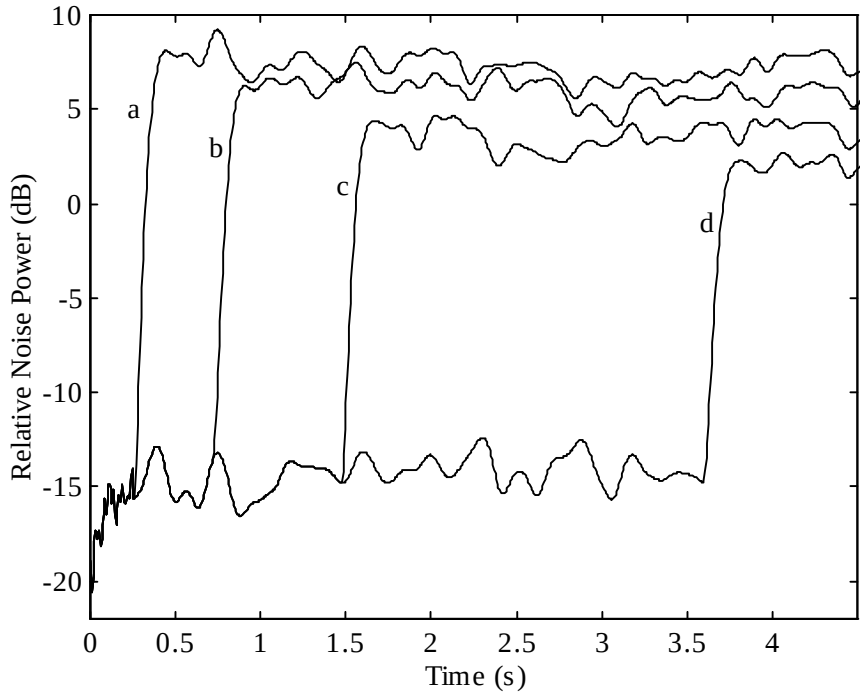

Figure 7.13 Phase-delay GSC noise improvement (LMS, FIR order=80, μ=1), weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

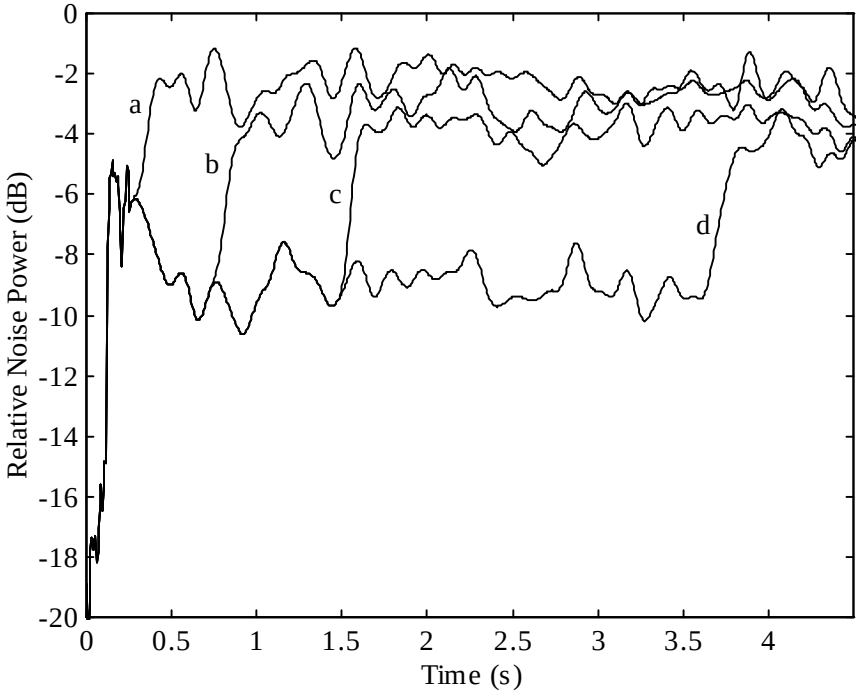

Figure 7.14 Phase-delay GSC noise improvement (LMS, FIR order=80, μ=0.1), weights fixed after a) 2400, b) 6000, c) 12000, d) 28800 snapshots

fixed. It may be advantageous to decrease μ once again after a certain amount of time. Therefore we will check out the results when we switch to μ=0.01 after 6000 snapshots of adaptation (the first 1200 snapshots will still use μ=1). Figure 7.15 shows results with weight fixing after 12000 and 28800 snapshots. Comparing Figure 7.15 with

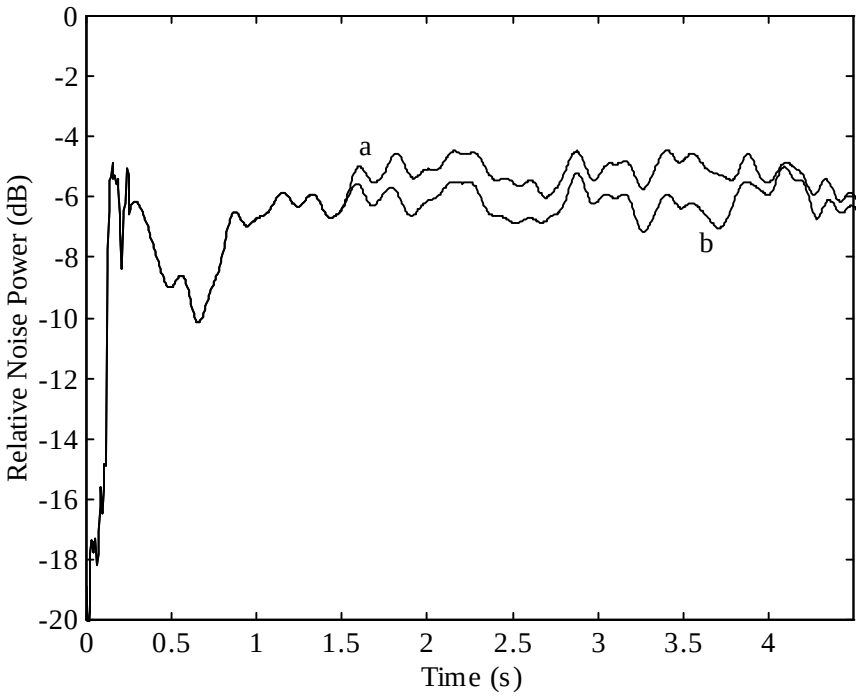

Figure 7.15 Phase-delay GSC noise improvement (LMS, FIR order=80, μ=1 for first 1200 snapshots, μ=0.1 for next 4800 snapshots, then μ=0.01), weights fixed after a) 12000, b) 28800 snapshots

Figure 7.14, we see between 1 and 2 dB better performance once the weights are fixed by changing to the lower μ, but the performance during adaptation is 2 dB worse. Performance after weight fixing is more important than performance before weight fixing, so we will use the scheme just developed. With this scheme, we can expect 2 to 6

dB improvement in the output noise power compared to the beamformer output, though we should expect 5 to 6 dB of improvement after a second of adaptation.

With the schemes developed for the time-delay and phase-delay GSC, we are now ready to test our schemes using speech data. We have discovered that with the noise setup that we have used, the performance of the time-delay and phase-delay GSC's are similar, with the phase-delay GSC giving on average 1 dB higher noise output power compared to beamformer output than the time-delay GSC. In the next section we will compare the outputs of the phase-delay and time-delay GSC's, and compare them to the speech and noise inputs into one of the microphones to determine the overall SNR improvement.

7.6 Performance of Array System, Speech and Noise

This section details results of the phase-delay and time-delay GSC's using a speech segment and noise. The speech segment consists of 3 phrases, the first two about 1.5 seconds long each, and the third about a second long. There is 2.5 seconds before the first speech segment, 1.5 seconds between the first and second and the second and third segments, and 1 second from the last segment until the end. The speech and noise were recorded separately such that we could quantitatively obtain a reasonable estimate of the overall SNR improvement. The speech source was situated at 135°, 65 cm from the array center. The reason for using this angle and distance is that they are typical for the car driver. One of the noise inputs was scaled so that we had equal SNR's at the inputs. The reason for having equal SNR's into each microphone input is that the SNR improvement that we compute will be the same compared to all inputs, plus we would expect similar input SNR's in the car environment.

Some of the results will be the same for the time-delay and phase-delay GSC's. The output of the speech activity detector will be the same for both, since it is using data directly from one of the microphones. Figure 7.16 shows the output of the speech detector, and the corresponding microphone input data that was used. The detector

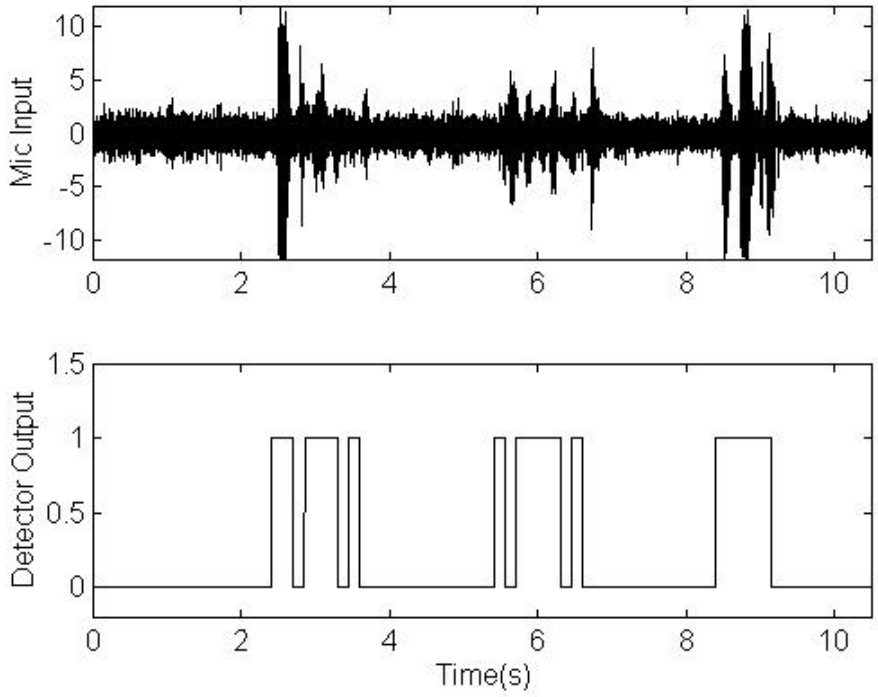

Figure 7.16 Input into one of the microphones and speech detector output

correctly identifies the speech segments, and using the scheme outlined in Section 7.4, we can expect the weights to be fixed during the duration of each of the 3 speech segments. Also, since the weights used during speech are the ones from 2400 snapshots previously, we should expect no speech to leak into the adaptive filters when the weights that will be used during speech are computed.

The other results that are the same for both GSC's are the DOA estimates. The covariance matrix updating scheme used is (6.2) with the additional $(1-\alpha)$ factor. The

value of α used was 0.6. The initial DOA was set to the actual source arrival angle, 135°. Then the DOA angle used after the first speech segment is set to the final DOA estimate from the first speech segment, and similarly after the other speech segments. The angles computed at the end of the three speech segments were 122.8°, 124.7°, and 126.1°, respectively. These estimates are somewhat off from the actual arrival angle. Part of the reason for this is that the noise in the third subband (816-1166 Hz) is louder than in any other band, and is situated more towards 90°. In the GSC implementations, after each speech segment, the beamformer and blocking matrix will be redefined for these different angles. We will see how making these changes when the DOA varies by such small angles affects the overall output.

Figure 7.17 shows the results using the time-delay GSC. We can see that there is significant noise reduction between one of the microphone inputs and the beamformer output, and between the beamformer output and the overall output. Unfortunately, there is also signal degradation. Degradation at the output of the beamformer is due to the misalignment of the input signals due to DOA measurement error, near-field effects, and microphone mismatching. Degradation between the beamformer output and the GSC output is due to the fixed filters operating on speech that leaked through the blocking matrix, due to similar reasons that caused beamformer output degradation. In the first speech segment, the actual DOA angle was used, and we still see some degradation due to near-field effects and microphone mismatching. Listening to one of the microphone inputs and the output demonstrates a noticeable increase in SNR. We would like to quantitatively measure this difference as well.

Since the speech and noise segments were recorded separately, we can use characteristics such as DOA angles, filter coefficients, and speech detector outputs of this speech plus noise case to determine the output with speech only as the input, and noise only as the input. Then we can compare powers and see the actual SNR improvement.

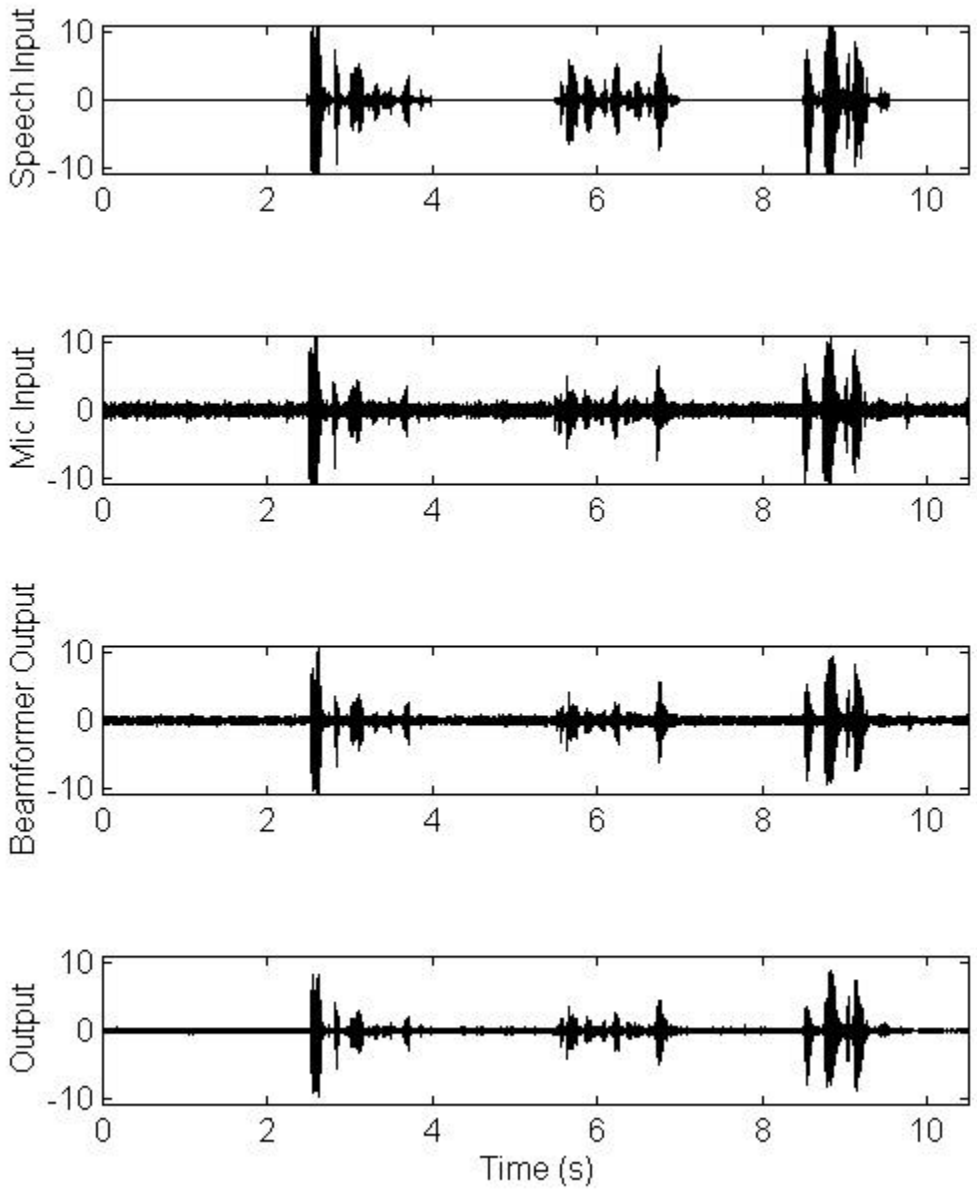

Figure 7.17 Time-delay GSC results

Results from these tests are shown in Table 7.9. It is significant to note that the noise output power increases after the first speech segment. This is attributed to the new time delays for the beamformer and blocking matrix, as a result of the change in the DOA

Table 7.9 Power results from noise-only and speech-only GSC (time-delay) compared with input powers.

Power, dB	0-2.5 s	2.5-3.9 s	3.9-5.5 s	5.5-7.0 s	7.0-8.5 s	8.5-9.6 s	9.6-10.5 s
Noise in	-8.5	-8.5	-8.2	-8.1	-9.0	-8.2	-9.2
Noise out	-19.7	-18.7	-17.0	-15.9	-17.3	-15.8	-20.5
Speech in	-	0.7	-	-3.6	-	3.3	-
Speech out	-	-2.3	-	-5.8	-	1.6	-

estimate from 135° to 122.8°. Although after the first speech segment μ was set to 1 for the first 1200 snapshots for faster convergence at the new DOA, the filters were not able to get as much noise rejection during adaptation as before and this loss, about 3 dB, is significant. It may be better just to set all of the filter coefficients to zero and start the adaptive process again, or keep using the same DOA until it varies by more than a specified amount. From Table 7.9 we can determine the SNR improvement during speech segments. We get about 7.3, 5.6, and 5.9 dB improvement for the speech segments (2^{nd}, 4^{th}, and 6^{th} columns in Table 7.9), respectively. Figure 7.18 shows the input and output speech spectrum during the first speech segment. It is shown that there is signal degradation, varying with frequency, but little distortion. It also was hard to hear any distortion in the output speech segment. Figure 7.19 shows the input and output noise spectrum during the first speech segment. There is good noise rejection at most frequencies, with notably less rejection in the 1100-1400 Hz range. Now we will examine results with the phase-delay GSC.

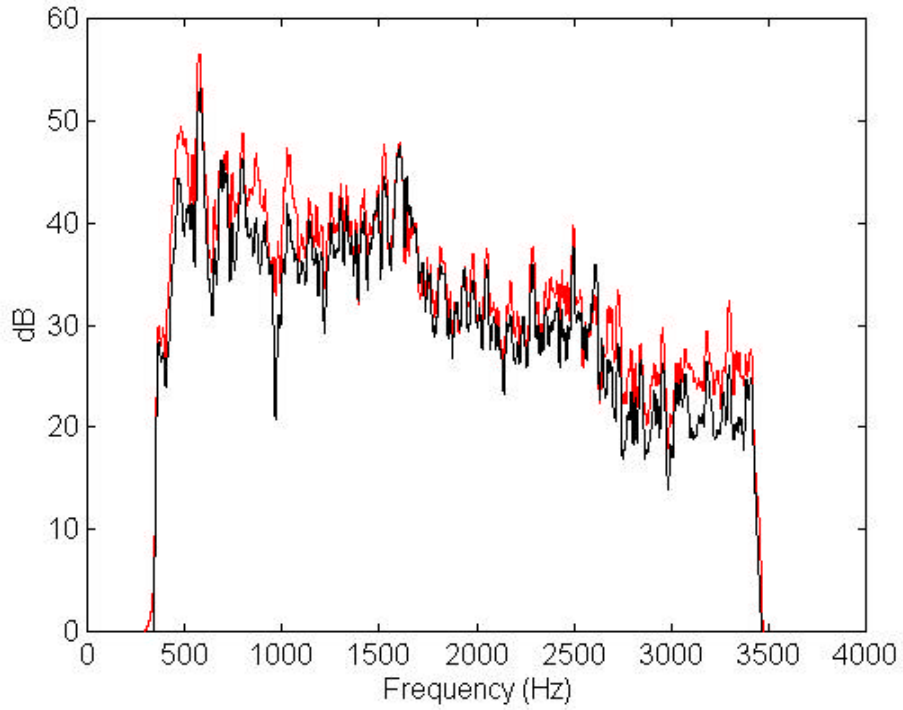

Figure 7.18 Time-delay GSC speech spectrum during 1st speech segment, microphone input (red), GSC output (black)

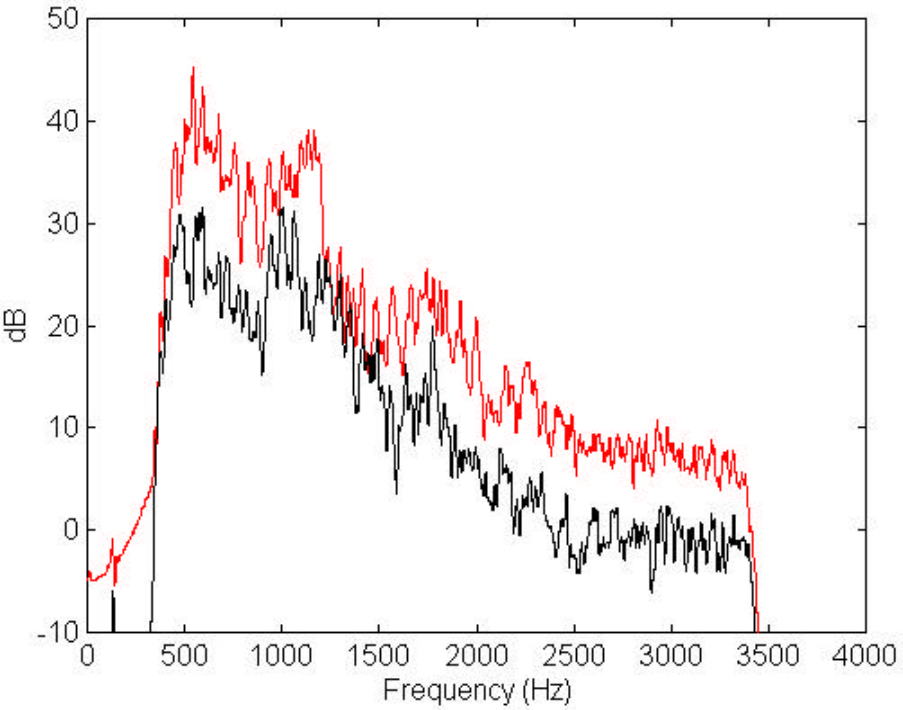

Figure 7.19 Time-delay GSC noise spectrum during 1st speech segment, microphone input (red), GSC output (black)

Figure 7.20 shows results from the phase-delay GSC. As with the time-delay GSC, there is significant reduction in noise between the stages, but also reduction in the speech signal. There is also a noticeable distinction between a microphone input and the overall output. At first glance, the speech degradation in the second and third segments seems to be more than the degradation in the time-delay GSC. As before, we will

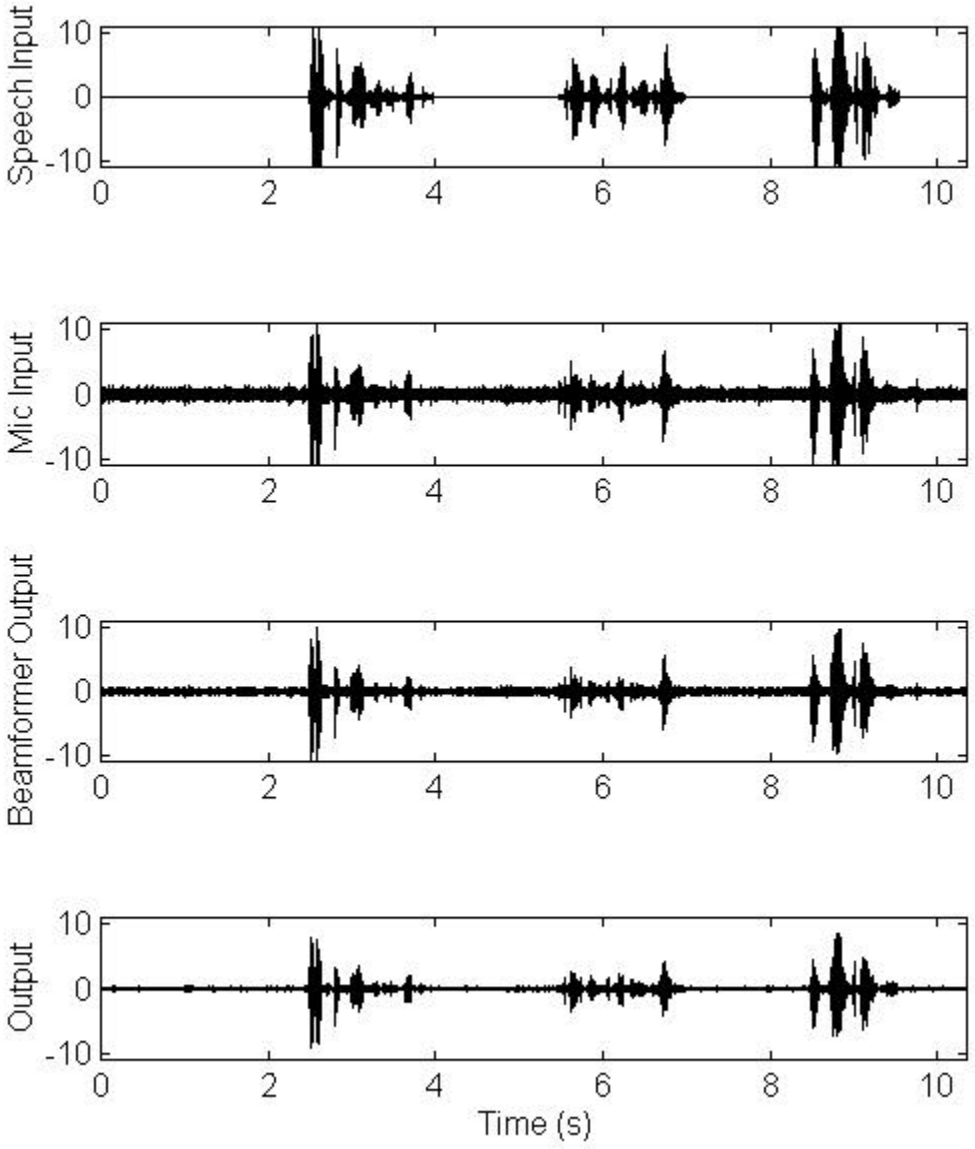

Figure 7.20 Phase-delay GSC results

examine the output powers of the GSC due to noise and due to speech to give us a quantitative measure of the SNR improvement. These results are shown in Table 7.10.

Table 7.10 Power results from noise-only and speech-only GSC (phase-delay) compared with input powers.

Power, dB	0-2.5 s	2.5-3.9 s	3.9-5.5 s	5.5-7.0 s	7.0-8.5 s	8.5-9.6 s	9.6-10.5 s
Noise in	-8.5	-8.5	-8.2	-8.1	-9.0	-8.2	-9.2
Noise out	-20.4	-19.2	-20.0	-18.2	-20.3	-18.0	-21.3
Speech in	-	0.7	-	-3.6	-	3.3	-
Speech out	-	-2.3	-	-6.4	-	0.8	-

The speech and noise input powers are the same as before. The output noise power is less for all segments than in the time-delay case. The speech power, though, is less in this case for the second and third segments than in the time-delay case. The SNR improvement during the three speech segments is 7.7, 7.3, and 7.2 dB, respectively. The input and output spectrum for the first speech segment is shown in Figure 7.21. As with the time-delay GSC, there is signal reduction but no noticeable distortion, visual or aural. The spectrum also shows that the difference in the speech powers for lower frequencies is more than that for higher frequencies. Figure 7.22 shows the input and output noise power spectrum. We have more noise rejection that with the time-delay GSC, and there also is very little rejection in the 1100-1400 Hz range.

The performance of the time-delay and phase-delay GSC have shown that there is audible improvement using the sidelobe cancelers over using just 1 microphone to gather speech data in a noisy environment. Though the phase-delay GSC performed a little better than the time-delay GSC, the improvement requires 6 times as much calculation effort for the adaptive filters. These results are for a simulated car noise environment, so

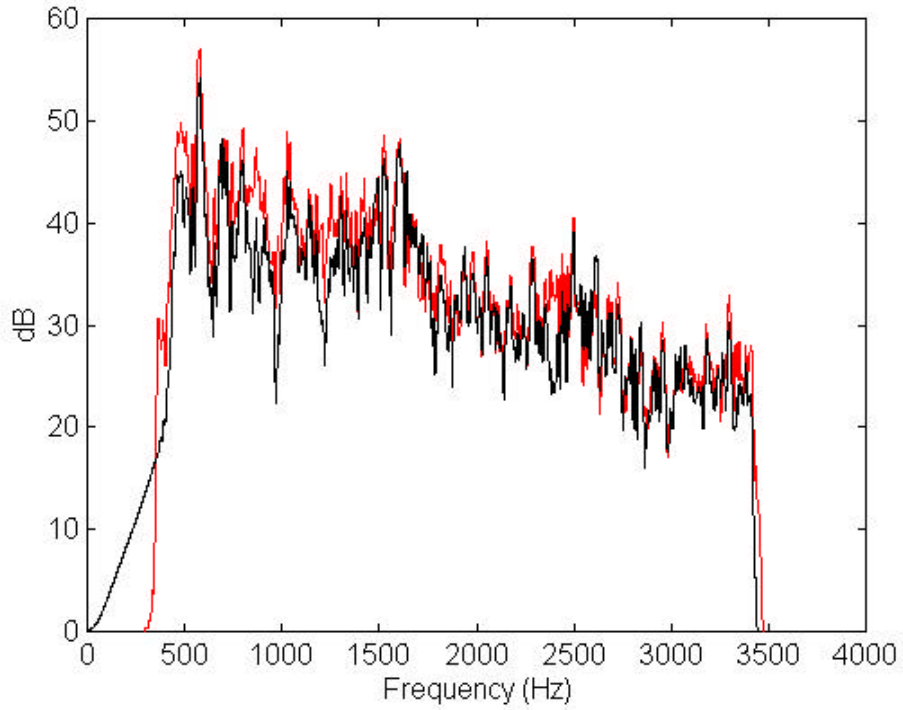

Figure 7.21 Phase-delay GSC speech spectrum during 1st speech segment, microphone input (red), GSC output (black)

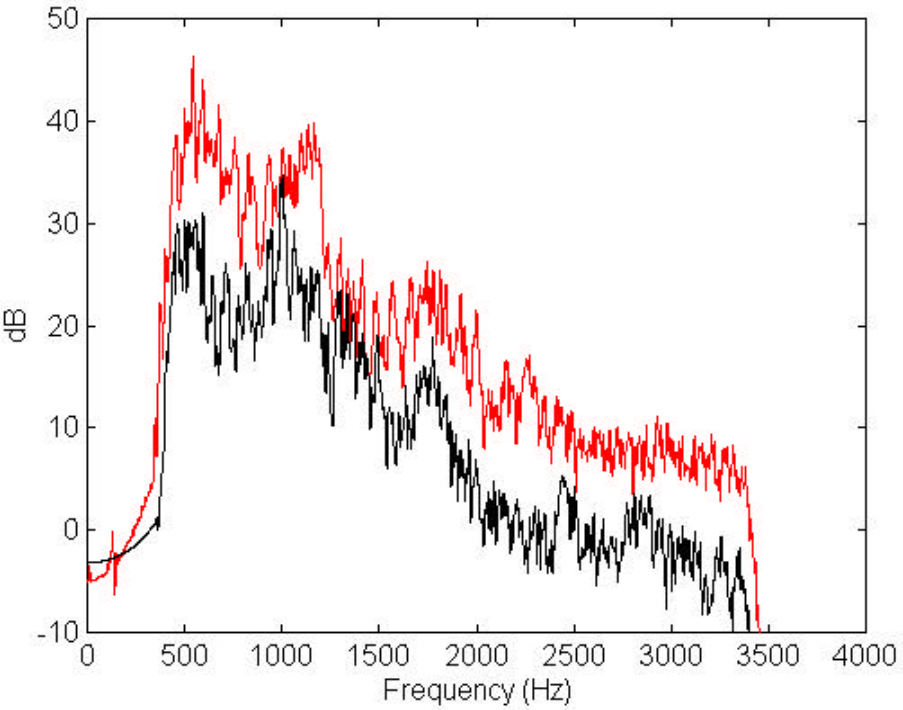

Figure 7.22 Phase-delay GSC noise spectrum during 1st speech segment, microphone input (red), GSC output (black)

actual tests inside a car will be required to determine the performance of the phase-delay and time-delay GSC's developed in this thesis.

Chapter 8

Conclusions

8.1 Conclusions

Results from Chapters 4, and 7 show many qualities about array processing. The array configuration started from a beamformer and then a sidelobe canceler was added to gives us two processes incorporated together to help improve SNR. From Chapter 4 we discovered that in the case of 2 directional broadband sources, we could achieve 40 dB rejection of one of the sources. Also it was discovered that the overall performance difference between using 6 subbands and just 1 band was small, with the 1-band system performing better.

Aspects of the performance of the 4-microphone array in a simulated car noise environment were discovered in Chapter 7. We found out that conventional source estimation, as well as source estimation using direction-finding algorithms were insufficient in estimating multiple sources in both high and low SNR environments. It was discovered that the MUSIC algorithm works very well in the 1-source scene, and also works well in the 2-source scene as long as the angular spacing between sources is more than 40° and some ad hoc restrictions are imposed. On average, performance of the MUSIC algorithm was best at angles near the center of the array (θ=90°) and worst at angles closer to broadside (θ=0°, 180°).

We examined the performance of the overall phase-delay and time-delay GSC's in Chapter 7. The time-delay GSC gave us between 5.6 and 7.1 dB of SNR improvement

during speech segments. With the phase-delay GSC we were able to obtain consistently about 7.4 dB improvement in SNR.

8.2 Further Work

Although we have covered many aspects of direction finding, beamforming, and the GSC, there are areas that require further research, showing promise for improved results using a microphone array structure. Though the MUSIC algorithm works well in our tests, newer methods such as TLS-ESPRIT and GEESE may be more helpful than MUSIC especially in the two-source scene. Also, for the two-source scene, we need to find better source estimators, as well as an algorithm that tracks each of the sources independently. The final speech activity detector used triggers on any periodic signal; we would have to modify the speech detector such that it doesn't trigger on other periodic signals like sirens and whistles. In the phase-delay GSC, adaptive filters for the lower-frequency subbands require much more time to converge than for the higher frequencies because the bands are so narrow; conventional subband techniques could be used to spread each band across the entire range of frequencies so that the filters will converge faster. If the 1-source algorithm is to be implemented, then we need to consider computational issues, most notably how to calculate eigenvalues and eigenvectors for the MUSIC algorithm.

With additional hardware, much more progress can be made in the development of hands-free car telephony. If we could acquire 4-channel data in the car, this would allow us to evaluate the spatial noise environment, echo effects, typical source movements, and overall performance during long periods with differing conditions. This

data would be invaluable to help us improve all aspects of the GSC. Additional microphones may improve overall performance, especially in the beamformer. Not only could additional microphones sharpen the main beampattern and lower sidelobes, but it could allow different interelement spacings for different frequency bands, thus making the beampattern in the lower frequency bands much narrower.

A realistic goal in the 1-source scene is to achieve on average 10 dB or better SNR improvement using fairly inexpensive components under any condition, with all processing done on a single DSP. This requires further testing, making first the obtaining of car data with 4 channels or more a necessity.

References

[1] B. Widrow, P.E. Mantley, L.J. Griffiths, B.B. Goode, "Adaptive Antenna Systems," *Proceedings of the IEEE*, vol. 55, December 1967.

[2] J. Capon, "High Resolution Frequency-Wavenumber Spectrum Analysis," *Proceedings of the IEEE*, vol. 57, pp. 1408-1418, August 1969.

[3] R.N. McDonough, "Application of the Maximum-Likelihood Method and the Maximum-Entropy Method to Array Processing," in *Nonlinear Methods of Spectral Analysis*. S. Haykin, Ed., Springer-Verlag, New York, 1983.

[4] P.W. Howells, "Intermediate Frequency Sidelobe Canceler," U.S. Patent 3202990, August 24, 1965.

[5] M.J. Levin, "Maximum-Likelihood Array Processing," in *Seismic Discrimination Semi-Annual Technical Summary Report*, M.I.T. Lincoln Laboratory, Lexington, MA, Technical Report DDC 455743, December 1964.

[6] S. Nordholm, I. Claesson, and B. Bengtsson, "Adaptive Array Noise Suppression of Handsfree Speaker Input in Cars," *IEEE Transactions on Vehicular Technology*, vol. 42, November 1993.

[7] J.S. Lim and A.V. Oppenheim, "Enhancement and Bandwidth Compression of Noisy Speech," *Proceedings of the IEEE*, vol. 67, December 1979.

[8] S.U. Pillai, *Array Signal Processing*, Springer-Verlag, New York, 1989.

[9] R.A. Monzingo and T.W. Miller, *Introduction to Adaptive Arrays*, John Wiley and Sons, New York, 1980.

[10] R.O. Schmidt, “Multiple Emitter and Signal Parameter Estimation,” *Proceedings, RADC Spectral Estimation Workshop*, pp. 243-258, October 1979.

[11] H. Akaike, “A New Look at the Statistical Model Identification,” *IEEE Transactions on Automation Control*, vol. AC-19, pp. 716-723, December 1974.

[12] J. Rissanen, “Modeling by Shortest Data Description,” *Automatica,* vol. 14, pp. 465-471, 1978.

[13] M. Wax and T. Kailath, “Detection of Signals by Information Theoretic Criteria,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, pp. 387-392, April 1985.

[14] K.V. Mardia, J.T. Kent, and J.M. Bibby, *Multivariate Analysis*, Academic, New York, 1979.

[15] F. Lorenzelli, A. Wang, D. Korompis, R. Hudson, and K. Yao, "Optimization and Performance of Broadband Microphone Arrays," *Proceedings, SPIE*, vol. 2563, pp. 158-168, February 1995.

[16] A. Paulraj, R. Roy, and T. Kailath, "Estimation of Signal Parameters via Rotational Invariance Techniques - ESPRIT," *Proceedings of the 19th Asilmar Conference*, November 1985.

[17] S.U. Pillai and B.H. Kwon, "GEESE (GEneralized Eigenvalues utilizing Signal subspace Eigenvectors) - A New Technique for Direction Finding," *Proceedings of the 22nd Asilmar Conference*, November 1988.

[18] R.L. Moses and A.A. (Louis) Beex, "Instrumental Variable Adaptive Array Processing," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, pp. 192-201, March 1988.

[19] O.L. Frost, III, "An Algorithm for Linearly-Constrained Adaptive Array Processing," *Proceedings, IEEE*, vol. 60, pp. 926-935, August 1972.

[20] S.P. Applebaum and D.J. Chapman, "Adaptive Arrays with Main Beam Constraints," *IEEE Transactions on Antennas and Propagation*, vol. AP-24, pp. 650-662, September 1976.

[21] L.J. Griffiths and C.W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming," *IEEE Transactions on Antennas and Propagation*, vol. AP-30, pp. 27-34, January 1982.

[22] Yoh'ichi Tohkura, "A Weighted Cepstral Distance Measure for Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 1414-1422, October 1987.

[23] B.H. Juang, L.R. Rabiner, and J.G. Wilpon, "On the Use of Bandpass Liftering in Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 947-954, July 1987.

[24] Hidefumi Kobatake, Katsuhia Tawa, and Akira Ishida, "Speech/Nonspeech Discrimination for Speech Recognition System Under Real Life Noise Environments," *Proceedings, ICASSP-'89*, vol. 89CH2673-2, February 1989.

[25] Sadaoki Furui, "Cepstral Analysis Techique for Automatic Speaker Verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, pp. 254-272, April 1981.

[26] D.H. Friedman, "Pseudo-Maximum-Likelihood Speech Pitch Extraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, pp. 213-221, June 1977.

Vita

David Kemp Campbell was born in Naples, FL in 1975. He received his Bachelors degree in Electrical Engineering at Virginia Tech in 1997, graduating Magna Cum Laude and was a Commonwealth Scholar. Since then he has been pursuing his Masters degree in Electrical Engineering at Virginia Tech. His research areas of interest are in music and speech signal processing. He is a member of IEEE, Tau Beta Pi, Eta Kappa Nu, Phi Eta Sigma, and the Audio Engineering Society.