Master Thesis

Anshul Kant Saxena

Wideband Audio Source Localization using

Microphone Array and MUSIC Algorithm

Fakultät Technik und Informatik
Department Informations- und
Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and
Electrical Engineering

Anshul Kant Saxena

Wideband Audio Source localization using Microphone Array and MUSIC Algorithm

Master thesis based on the examination and study regulations for
the Master of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner : Prof. Dr.-Ing. Hans Peter Kölzer
Second examiner : Prof. Dr.-Ing. Ulrich Sauvagerd

Day of delivery March 19th 2009

Anshul Kant Saxena

Title of the Master Thesis

Wideband Audio Source Localization using Microphone Array and MUSIC Algorithm.

Keywords

DSP, MUSIC, SVD, LPC, Self-Calibration, Microphone Array Processing, Wideband signal

Abstract

In this project a DSP based real time system has been developed to localize the audio source in reverberant environment. The System is developed on TI's TMS320C6713 DSP. The Incoherent Wideband MUSIC Algorithm used in the project is based on Eigenvalue decomposition method. The algorithm is simulated in MATLAB under real time constraint. The project is developed in C using TI's Code Composer Studio.

Anshul Kant Saxena

Thema der Masterarbeit

Breitband Audio-Quellen Lokalisierung mit einem Mikrofon Array und MUSIC Algorithmus.

Stichworte

DSP, MUSIC, SVD, LPC, Selbst-Kalibrierung, Microfon Array Verarbeitung, Breitband Signal

Kurzzusammenfassung

In dieser Arbeit wurde ein DSP basiertes Echtzeitsystem entwickelt, dass zur Lokalisierung einer Audioquelle in einem nachhallendem Raum eingesetzt wird. Das System wurde auf dem TI's TMS320C6713 DSP entwickelt. Der verwendete Incohärenter Breitband MUSIC-Algorithmus basiert auf der Eigenwert Dekomposition. Der Algorithmus wurde in MATLAB unter Echtzeitbedingungen simuliert. Die Implementierung auf dem DSP erfolgte in der Programmiersprache C.

ACKNOWLEDGEMENT

I wish to express my deepest appreciation to Prof. H. P. Kölzer for the opportunity he has provided, the guidance and the motivation. His courtesy, professionalism and patience made working with him very rewarding and gratifying. Throughout the entire thesis work, he provided me with timely and invaluable suggestions.

I would also like to extend my sincere gratitude and appreciation to Prof. U. Sauvagerd for his advice, guidance, and encouragement throughout the whole Master studies.

I would be failing in my duty without paying my special thanks to Mr. J. Pflüger who helped me a lot with construction and designing of microphone array.

I owe a lot to my family for their constant support, inspiration and affectionate assistance in all my endeavors.

Lastly I acknowledge the help and assistance of my friends who have lent their support for the completion of the thesis.

Hamburg, March 2009 **Anshul Kant Saxena**

ABSTRACT

The detection and estimation techniques that are used in Microphone Array Processing depend on the spatial and temporal characteristics of the signals that arrive on the Microphone Array. A combination of Microphone Arrays and sophisticated signal processing has been used to acquire the high-quality speech audio. These applications exploit the spatial filtering ability of Microphone Array.

In this thesis wideband array processing is considered to develop a real-time DSP system based on an adaptive, robust, wideband algorithm to localize the speech source in reverberant environment. The approach is based on sampling the spectrum of the source signal to generate narrowband frequency bins and then these separate estimates at multiple frequencies are combined into single direction of arrival. For this purpose Incoherent Signal-Subspace method with high resolution MUSIC algorithm is used.

The algorithm is first developed in MATLAB and it can be shown that the algorithm is effective in locating the audio source with high resolution, with significant results for SNR down to -5 dB. The MUSIC algorithm requires the knowledge of frequency of audio source in wideband environment *a priori*.

The Incoherent wideband MUSIC algorithm is then implemented on DSP TMS320C6713 with PCM 3003 codec. A self-calibrating algorithm is employed to calibrate the microphone's signal obtained from Microphone Array. The self-calibrated algorithm was tested in anti-acoustic room and the performance of algorithm was good. Thereafter, complete system was tested in seminar room for an audio source under strong reverberation effect and noisy environment with six and eight microphones. The performance of algorithm in real time was good and the results obtained were quite satisfactory and were within the acceptable range of deviation.

TABLE OF CONTENTS

List of Figures

List of Tables

Nomenclature

*	Convolution
Re{.}	Real part of complex number
$(\,.\,)^T$	Transpose
$(\,.\,)^H$	Hermitian (complex-conjugate transpose)
I	Identity matrix
E{.}	Expectation operator
diag{$a_1, \ldots, a_D$}	Diagonal matrix
span{.}	Range (column) space
det(.)	Determinant
$\|\|.\|\|$	Norm

Abbreviations

CCS	Code Composer Studio
DFT	Discrete Fourier Transform
EDMA	Enhanced Direct Memory Access
EDOA	Estimated Direction of Arrival
DSP	Digital Signal Processor
DTFT	Discrete-Time Fourier Transform
EVD	EigenValue Decomposition
FFT	Fast Fourier Transform
GUI	Graphical User Interface
LPC	Linear Predictive Coding
MUSIC	MUltiple SIgnal Classification
SPL	Sound Pressure Level
STFT	Short-Time Fourier Transform
SVD	Singular Value Decomposition
TI	Texas Instruments
ULA	Uniform Linear Array
VLIW	Very Long Instruction Word

1. INTRODUCTION

In speech applications, where a conventional microphone need to be placed very near to Speaker forces the Speaker to either wear the microphone or need to be monitored by Human interface. However this restriction is undesirable and inconvenient for many applications like teleconferencing, car telephony and localizing a source in reverberant environment. For these applications a combination of microphone arrays and sophisticated signal processing can be used as they exploit the spatial filtering ability of microphone array. A microphone array is known to be effective method to enhance the SNR in noisy environments resulting in significant improvement of speech characteristics.

Most of the applications like speech enhancement for Human Computer interface or in hearing aids requires accurate localization techniques to produce direction of arrival or estimates at a high rate with minimum latency i.e. in real time. While localizing a source a system must produce reliable location estimates. The movement of Speaker must be negligible for the duration of computation of data set or the refreshing rate must be high enough to avoid errors.

In all these applications, one thing is common i.e. to determine the direction of arrival of the acoustic source in reverberant environment. Reverberation which is being defined as the complicated set of reflections that are produced when a sound wave travels from source to listener by bouncing off many different surfaces. This phenomenon is very common in closed space like conference hall. The reverberation effect can severely degrade the performance of direction of arrival algorithms. The motivation of this thesis is to develop a real time DSP system to localize an acoustic source in reverberant environment.

1.1 Background of this Thesis

The basic theory behind the estimation of direction of arrival using Microphone Array is to make use of the phase information present in the signals picked by sensors which are spatially separated. When the microphones are spatially separated the sound source signal arrive at them with time differences. For known array geometry, these time-delayed signals are dependent on the direction of arrival of the signal.

In so far as the estimation of direction of arrival for narrowband sources are concerned, the theory is well established and lots of literature is available. Within many direction of arrival algorithms, MUSIC (Multiple Signal Classification) [1, 2] has been most widely studied. The MUSIC algorithm is based on Eigen-value Decomposition (EVD) method. The EVD method divides the cross-correlation matrix of the array signals into signal and noise subspaces. The popularity of MUSIC algorithm is due to its generality i.e. it is applicable to arrays of arbitrary but known configurations and response, and can be used to estimate multiple parameters per source. The condition is that array response must be known for all the possible combinations of source parameters.

The narrowband MUSIC produces a sharp beam patterns, but requires the frequency bin to have high SNR. In general, any narrowband direction of arrival technique will not exploit the wideband nature of the acoustic sources. To exploit as much of the multispectral content from the acoustic source as possible, improve accuracy and stability of the direction of arrival estimates, a wideband direction of arrival algorithm is required.

One approach is to implement wideband MUSIC algorithm using Incoherent Signal-Subspace method [3] in frequency domain. This approach is useful if there is sufficient or high SNR in multiple frequency bins, so that narrowband MUSIC algorithm yields good results independently for each bin. Over each processing interval it is assumed that a single frequency bin is occupied by a single source only. This takes advantage of the non stationary nature of the source and simplifies the algorithmic complexity of the algorithm. This assumption is justified because different wideband sources are not likely to occupy all of the same bins in any given processing interval and keeps on changing bins as function of time.

1.2 Thesis Organization

The dissertation is organized as follows. In this chapter the background and motivation behind this work is discussed. In the following chapter some array processing techniques and the concepts behind the MUSIC algorithm is discussed. Also the theoretical implementation of the MUSIC algorithm in wideband is introduced. In Chapter 3, the characteristics of microphone array are explained. The DSP sub-system and interfacing between them is also discussed. In Chapter 4, the methods and techniques used for the implementation of algorithm is discussed. Also the simulation results of the

algorithm are analyzed. In Chapter 5, the implementation of algorithm in real time is explained. In Chapter 6, the systematic analysis of tests conducted in real time is presented. In Chapter 7, the summarization of the work done with results is concluded and further improvement as well as possible future work is discussed.

2. THEORY

An array processing is usually performed in two steps: detection and localization. Detection is a terminology used for the procedure that determines the number of signals arriving at array. Localization (also called estimation) is a process to estimate the spatial parameters of the signals such as direction of arrival. The methods that are used for detection and localization are generally categorized as subspace decomposition or beam forming techniques. The subspace decomposition has a better resolution in comparison to beam forming techniques but on the other hand their implementation is more complex. The most common subspace decomposition techniques are MUSIC and ESPRIT algorithms. In this thesis MUSIC algorithm is used.

2.1 Fundamentals of Array Processing

In a normal environment, a wave field at a spatially fixed microphone is linearly related to an assumed signal, s(t). This is true for an enclosed space (e.g. conference hall) as well as for free space. In free space or non-reverberant environment, sound waves propagate without any interference from different objects such as wall or other sources. But in a closed space such a free-space model is not realistic. However it accurately describes the direct-path propagation from source to microphone, even in reverberant environment. The linearity of the medium allows the microphone signal to be modeled as the superposition of a direct path component plus the reflected sound waves. Signal processing algorithms rely on separating the direct path component from reflected waves and noise as it is able to parameterize the location of Speaker.

2.1.1 Direction of Propagation and Arrival

Figure 2.1 in next page shows the layout of linear microphone array consisting of *M* microphones with *K* incident signals from a sound source. In this analysis, the incident waves are assumed to be plane waves (spherical wavefront) i.e. the sound source is approximated as at a much greater distance than the distance between microphones. This assumption implies that the sound waves reaching at different microphones are parallel to each other because the far-field arrays cannot resolve the range of the source. As the sound waves have to travel different distances to reach their respective microphones, this means that these sound waves will be time-delayed or time-advanced versions of each other. As depicted in Figure 2.1 the direction perpendicular to the

microphone array will be taken as reference direction of arrival. The angles in the clockwise direction with respect to this reference will be positive and in anticlockwise direction will be negative.

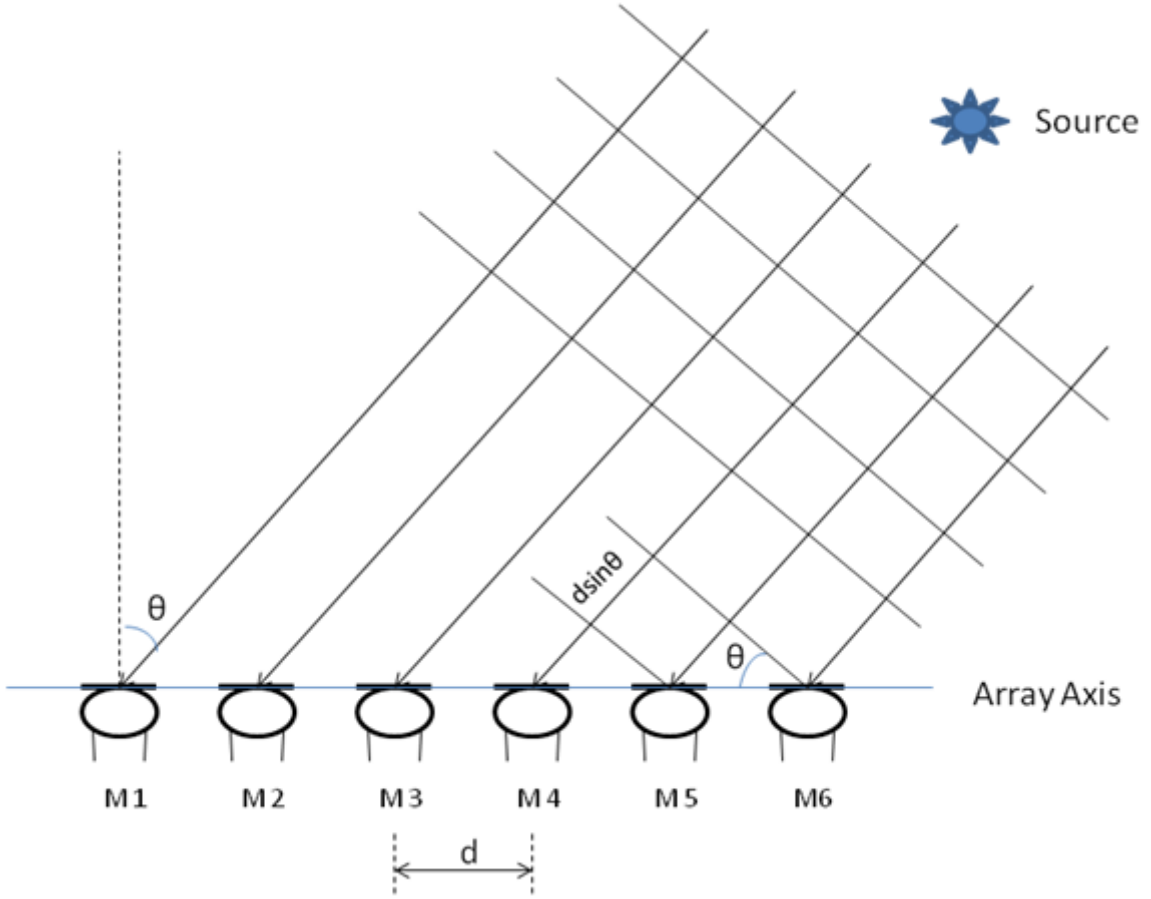

Figure 2.1 Uniform Linear Array with Far Field Source

For conventional purpose, the first microphone M_1 is chosen as the reference microphone. The distance between any microphone pair is constant, say d. The distance to be travelled by sound waves from a source to a microphone M_i with respect to reference microphone M_1 will be given by Eq. 2.1

$$\tau_i = -\frac{d_i}{v}\sin\theta \qquad (2.1)$$

This equation indicates that the sound wave incident on microphone M_i will be time-delayed version of reference microphone.

To summarize, lets assume D source signals to be narrowband and impinge on the array from directions $\{\theta_1,...,\theta_k,...,\theta_D\}$. The DOAs on the different microphones of the source signals are the same. The received signal with k^{th} source signal with center frequency w_k is written as

$$\tilde{s}_k(t) = u_k(t)\cos(\omega_k t + v_k(t)) \tag{2.2}$$

where, $u_k(t)$ and $v_k(t)$ are slowly varying functions of time that defines the amplitude and phase of the signal.

In complex form this signal is represented as follows

$$\begin{aligned} s_k(t) &= u_k(t)e^{j(\omega_k t + v_k(t))} \\ \tilde{s}_k(t) &= \mathrm{Re}\{s(t)\} \end{aligned} \tag{2.3}$$

As $u_k(t)$ and $v_k(t)$ are slowly varying functions, this implies that for a small propagation delays, $\tau_i(\theta_k)$, the following equations are held

$$\begin{aligned} u_k(t-\tau_i(\theta_k)) &\approx u_k(t) \\ v_k(t-\tau_i(\theta_k)) &\approx v_k(t) \end{aligned} \tag{2.4}$$

Hence,

$$\begin{aligned} s_k(t-\tau_i(\theta_k)) &= u_k(t-\tau_i(\theta_k))e^{j(\omega_k(t-\tau_i(\theta_k))+v_k(t-\tau_i(\theta_k)))} \\ &\approx u_k(t)e^{j(\omega_k(t-\tau_i(\theta_k))+v_k(t))} \\ &\approx u_k(t)e^{j(\omega_k t+v_k(t))}e^{-j\omega_k\tau_i(\theta_k)} \end{aligned}$$

Now Eq. 2.3 can be written as

$$s_k(t-\tau_i(\theta_k)) \approx s_k(t)e^{-j\omega_k\tau_i(\theta_k)} \tag{2.5}$$

In the presence of sound-reflecting surfaces and noise, the sound waves produced by a single source propagate along multiple acoustic paths. This gives rise to the reverberation effect (as depicted in Figure 2.2). In this model, the received signals are expressed as

$$x_i(t) = h_i(t) * s_k(t-\tau_i(\theta_k)) + w_i(t) \tag{2.6}$$

where, $h_i(t)$is the impulse response of the i^{th} microphone.

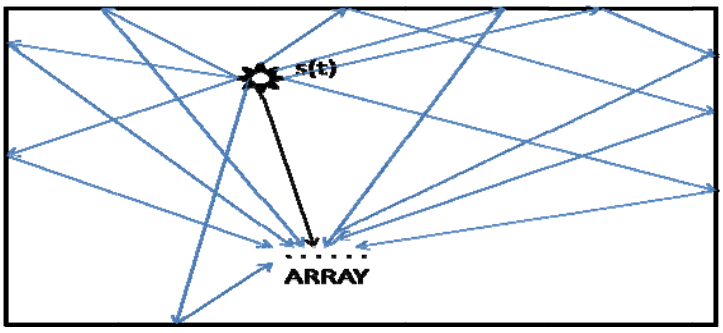

Figure 2.2 Reverberation effect

Assuming that effects of microphones are very small we can ignore the attenuation and propagation delay caused by microphones. We also assumed that microphones are ideal and their impulse response is 1. These assumptions will lead to Eq. 2.7

$$x_i(t) = \sum_{k=1}^{D} s_k(t) e^{-j\omega_k \tau_i(\theta_k)} + w_i(t) \tag{2.7}$$

Eq. 2.7 can be rewritten if we consider that D (assumption: D < No. of Microphone) sound sources are impinging on the microphone array from different directions. Then the received signal at the i^{th} microphone will be expressed as

$$x_i(t) = \sum_{k=1}^{D} a_i(\theta_k) s_k(t) + w_i(t) \tag{2.8}$$

where $s_k(t)$ is the signal of the k^{th} audio signal, $a_i(\theta_k)$ is the complex response of the i^{th} microphone to the k^{th} audio signal and $w_i(t)$ is the additive noise at the i^{th} microphone.

2.1.2 Spatial Aliasing Effect

From Eq. 2.6 the location vector of an array is defined as the frequency response of the array for a given direction of arrival. For an array of M sensors, the location vector is a column vector with M components and is represented by a(ω, θ). The location vector of a uniform linear array with the phase reference taken at the first sensor is given by

$$a(\theta) = \left[1 \; e^{-j\pi \frac{2d \sin\theta}{\lambda}} \; \ldots\ldots\ldots e^{-j\pi \frac{2d(M-1)\sin\theta}{\lambda}} \right]^T \tag{2.9}$$

We can see that the distance between microphones and λ or frequency of incident signal is related to each other. As it is known that for a narrowband direction of arrival estimation, the received signals between microphones have a phase delay with respect to each other. This phase difference between a pair of microphone should not be more than π because the phase difference of φ > π is indistinguishable from a phase lead of 2π – φ or vice-versa. Spatial Aliasing [4] occurs if the phase delay between a pair of microphone is greater than π then the signals that are located at θ_1 and θ_2 will give the same array output. Because any phase difference greater than ±π will be wrapped around in the range. This spatial undersampling will cause aliasing of higher frequency components down into the frequency band of interest and will result into the wrong interpretation of delays in time domain which subsequently will lead to wrong estimation of direction of arrival. Figure 2.3 on next page depicts this situation.

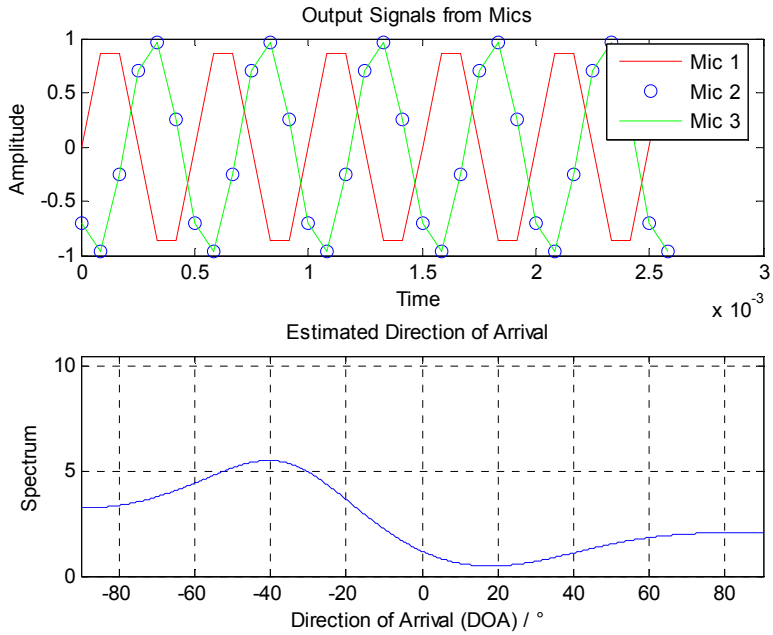

Figure 2.3(a) Spatial Aliasing Effect and Estimated Direction of Arrival using MUSIC

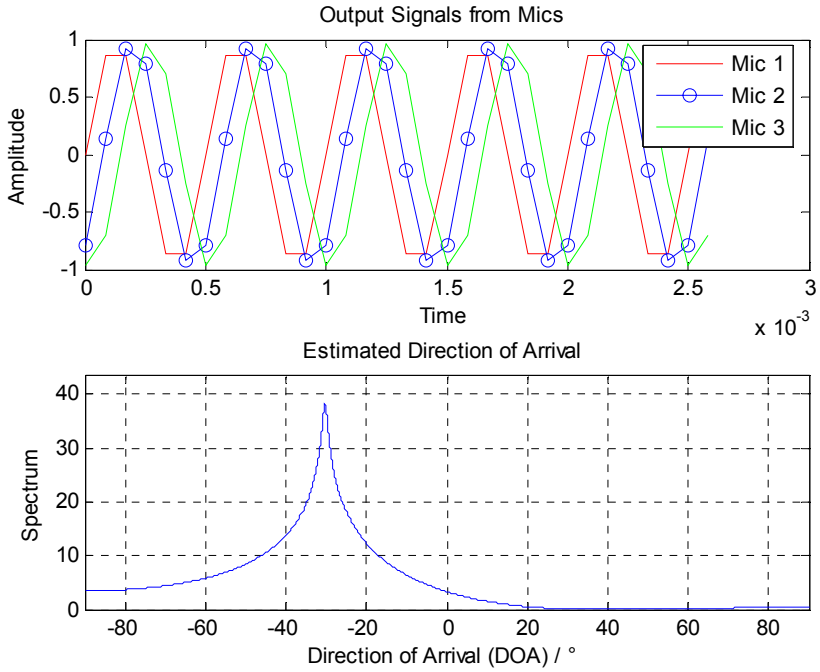

Figure 2.3(b) No Spatial Aliasing Effect and Estimated Direction of Arrival using MUSIC

Wideband spectrum contains many frequency components (for speech applications the range of interest is between 100 – 3400 Hz) and maximum frequency component should

be chosen in the given spectrum, say f_{max} and the condition for avoiding spatial aliasing is given in Eq. 2.10

$$2\pi f_{\max}\tau \le \pi \tag{2.10}$$

Substituting for τ, we can calculate the minimum required distance between adjacent microphones to avoid spatial aliasing, which is given by Eq. 2.11

$$d \le \frac{1}{2}\left(\frac{v}{f_{\max}\sin\theta}\right) \tag{2.11}$$

Hence the relationship between d and λ is given by Eq. 2.12

$$d \le \frac{1}{2}\left(\frac{\lambda_{\min}}{\sin\theta}\right) \tag{2.12}$$

where λ_{min} is the smallest wavelength.

Therefore in a worst case scenario i.e. when θ = 90˚, the distance between two adjacent microphones should not be greater than half of λ_{min} present in the signal. According to [4] if the distance between two adjacent microphones is greater than $\frac{\lambda_{\min}}{2}$, it will lead to multiple main lobes, which is undesirable.

If the spatial sampling rate is kept at less than $\frac{\lambda_{\min}}{2}$ for the highest frequency of interest, the spacing between microphones can be adjusted according to our requirement. As the distance between microphones get closer, the far field signals in the microphones are more highly correlated and the Microphone array has better overall background noise suppression over a wider range of frequencies. As the spacing gets further apart, the array will have less overall suppression and becomes restricted to lower frequency responses.

2.1.3 Relation between Source-Array Distance and SPL

The commonest model for sound sources is the point source which assumes that the sound waves are radiating from a point. As in real time the Source-Array distance is varied and it is important to know in principle how much signal power is available for array processing. In this section, a mathematical formula is shown to relate the Source-Array distance with signal power.

Sound Pressure Level (SPL) decreases proportionally with distance say ‘R’ from the sound source. The Figure 2.4 in next page shows a source of sound with two listening positions: the closer one is r cm away; the farther one is R cm away.

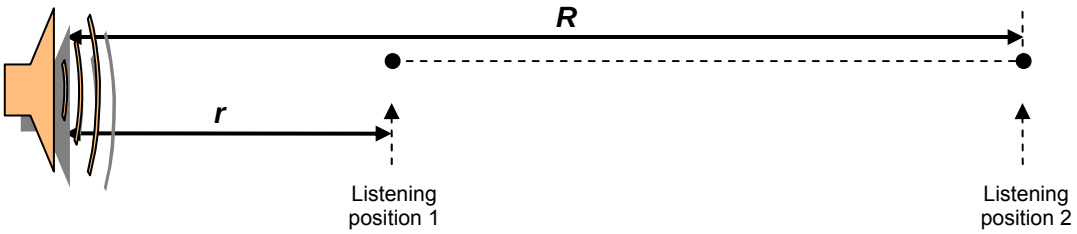

Figure 2.4 Effect of distance in SPL

The change in the sound pressure level between the two positions is given by:

$$Change\,in\,the\,Sound\ \ pressure\,Level = 20 \times \mathrm{Log}\left(\frac{R}{r}\right)$$

For speech applications, the reference point is generally accepted as 96 dB SPL approximately 1 cm (r) from the lips of a person when he or she speaks [12].

The equation which we can plot (Figure 2.5) is shown below.

$$Sound\ \ pressure\,Level\,(dB) = 96 - 20 \times \mathrm{Log}\left(\frac{R}{r}\right)$$

The Figure 2.5 shows the plot.

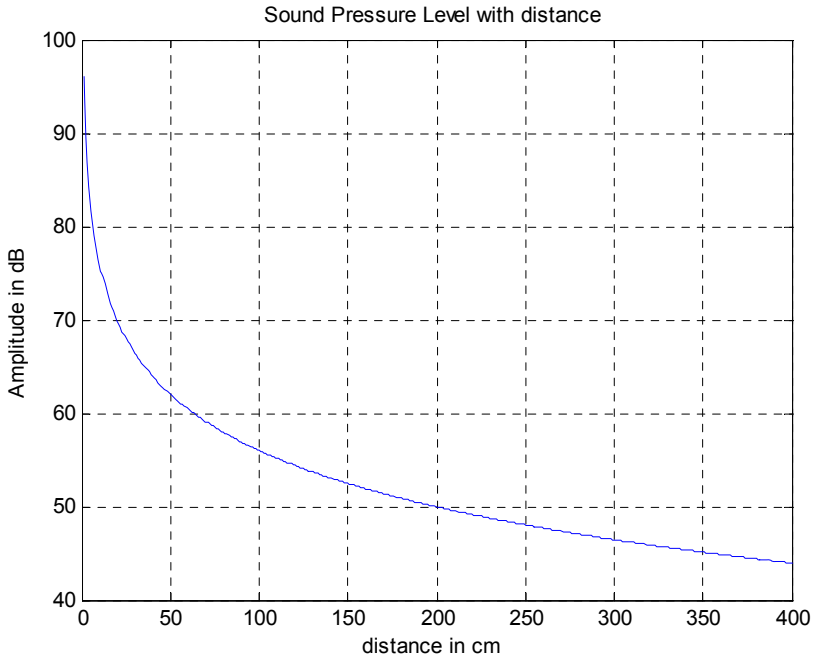

Figure 2.5 Change in SPL with increase in Source-Array Distance

The curve above shows a loss of 6 dB with every doubling of distance. We can see that for initial distance say up to 50 cm the fall in SPL is rapid. We can observe that the SPL

at 150 cm is approximately 52.5 dB and at 300 cm 46.5 dB. If microphones are placed at a distance of say 5 cm from each other in an array. Then the distance between 1st and 8th microphone for array having 8 microphones will be 35 cm and difference in SPL will be 1 dB, but at 300 cm the overall power of signal is dropped to 45.5 dB for the farthest microphone in the array and if high noise level is present in spectrum then there is a possibility of losing the main signal for large Source-Array distance like 400 cm and beyond that. In chapter 4.3, the effect of low SNRs on the performance of algorithm is discussed.

2.2 MUSIC Algorithm in Frequency Domain

Many subspace decomposition methods divide the observation space into the signal and noise subspaces. The first step in these techniques is to estimate the signal and noise subspaces by decomposing the array correlation matrix into its eigen-structure form. The subspace spanned by the eigenvectors of Covariance matrix corresponding to dominant eigenvalues is termed as Signal subspace. The detection methods use the fact that signal eigenvalues are larger than the noise eigenvalues. One of the most popular method is MUSIC (MUltiple SIgnal Classification) which can be characterize as the method for estimating the individual frequencies of multiple time-harmonic signals. The algorithm is considered in frequency domain [5]. Eq. 2.8 can be written as follows

$$\mathbf{X(t)} = \boldsymbol{A(\theta)} \cdot s\mathbf{(t)} + \mathbf{w(t)} \tag{2.13}$$

The frequency domain representation of the vector x(t) can be obtained by taking the Fourier transform of above equation

$$\mathbf{X(\omega,\theta)} = \boldsymbol{A(\omega,\theta)} \cdot \mathbf{S(\omega)} + \mathbf{W(\omega)} \tag{2.14}$$

where, $\mathbf{X(\omega,\theta)} = [\mathrm{X}_1(\omega,\theta),...,\mathrm{X}_\mathrm{N}(\omega,\theta)]^{\boldsymbol{T}}$ is the N-dimensional vector of array output

$\mathbf{S(\omega)} = [\mathrm{S}_1(\omega),...,\mathrm{S}_\mathrm{D}(\omega)]^{\mathrm{T}}$ is the D-dimensional vector of audio signal

$\mathbf{W(\omega)} = [\mathrm{W}_1(\omega),...,\mathrm{W}_\mathrm{N}(\omega)]^{\boldsymbol{T}}$ is the N-dimensional of noise

As the microphones are assumed to be identical and under far-field consideration, the array steering vector $\boldsymbol{A(\omega,\theta)}$ can be written as

$$\boldsymbol{A(\omega,\theta)} = [\boldsymbol{a(\omega,\theta_1)},...,\boldsymbol{a(\omega,\theta_D)}] = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ e^{-j\omega\tau_1} & e^{-j\omega\tau_2} & \cdots & e^{-j\omega\tau_D} \\ e^{-j2\omega\tau_1} & e^{-j2\omega\tau_2} & \cdots & e^{-j2\omega\tau_D} \\ \vdots & \vdots & \vdots & \vdots \\ e^{-j(N-1)\omega\tau_1} & e^{-j(N-1)\omega\tau_2} & \cdots & e^{-j(N-1)\omega\tau_D} \end{pmatrix} \tag{2.15}$$

All of the classical beamforming techniques use the output covariance matrix to determine direction of arrival of source. Another important property of the covariance matrix is that one can see the output power at each of the sensors and the matrix can easily be modified to make the gains at all of the sensors.

The symmetric covariance matrix of $\boldsymbol{X}$ of Eq. 2.14 is given by

$$\hat{\mathrm{R}}(\omega,\theta) = \mathrm{E}[\boldsymbol{X}\cdot\boldsymbol{X}^{\boldsymbol{H}}] = \frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}} \boldsymbol{X}(\omega,\theta)\boldsymbol{X}(\omega,\theta)^{\boldsymbol{H}} \tag{2.16}$$

where, E[·] is the expectation operator and {}$^{\mathrm{H}}$ denotes complex conjugate transpose.

On replacing with their respective values, we get

$$\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}} \left(\boldsymbol{A}(\omega,\boldsymbol{\theta})\cdot \mathrm{S}(\omega)+\boldsymbol{W}(\omega)\right)\cdot\left(\boldsymbol{A}(\omega,\boldsymbol{\theta})\cdot \mathrm{S}(\omega)+\boldsymbol{W}(\omega)\right)^{\mathrm{H}} \tag{2.17}$$

The Eq. 2.17 is further calculated as

$$\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}} \left(\boldsymbol{A}(\omega,\boldsymbol{\theta})\cdot \mathrm{S}(\omega)+\boldsymbol{W}(\omega)\right)\cdot\left(\mathrm{S}^{\mathrm{H}}(\omega)\cdot\boldsymbol{A}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta})+\boldsymbol{W}^{\boldsymbol{H}}(\omega)\right) \tag{2.18}$$

$$\begin{aligned}\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}} &\left(\boldsymbol{A}(\omega,\boldsymbol{\theta})\boldsymbol{S}(\omega)\boldsymbol{S}^{\mathrm{H}}(\omega)\boldsymbol{A}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta})+\boldsymbol{A}(\omega,\boldsymbol{\theta})\boldsymbol{S}(\omega)\boldsymbol{W}^{\boldsymbol{H}}(\omega)\right)\\ &+\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\left(\boldsymbol{W}(\omega)\boldsymbol{S}^{\mathrm{H}}(\omega)\boldsymbol{A}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta})+\boldsymbol{W}(\omega)\boldsymbol{W}^{\boldsymbol{H}}(\omega)\right)\end{aligned} \tag{2.19}$$

Eq. 2.19 is rearranged as following

$$\begin{aligned}\boldsymbol{A}(\omega,\boldsymbol{\theta})&\left[\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{S}(\omega)\boldsymbol{S}(\omega)^{\mathrm{H}}\right]\boldsymbol{A}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta})+\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{W}(\omega)\boldsymbol{W}^{\boldsymbol{H}}(\omega)\\ &+\boldsymbol{A}(\omega,\boldsymbol{\theta})\left[\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{S}(\omega)\boldsymbol{W}^{\boldsymbol{H}}(\omega)\right]+\left[\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{W}(\omega)\boldsymbol{S}^{\mathrm{H}}(\omega)\right]\boldsymbol{A}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta})\end{aligned} \tag{2.20}$$

As it is assumed that the noises and incoming signals are not correlated, which means the last two parts are equal to zeros, so Eq. 2.20 is reduced to

$$\boldsymbol{A}(\omega,\boldsymbol{\theta})\left[\frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{S}(\omega)\boldsymbol{S}(\omega)^{\boldsymbol{H}}\right]\boldsymbol{A}(\omega,\boldsymbol{\theta})^{\boldsymbol{H}} \;+\; \frac{1}{\mathrm{M}}\sum_{\mathrm{n=1}}^{\mathrm{M}}\boldsymbol{W}(\omega)\boldsymbol{W}(\omega)^{\boldsymbol{H}} \tag{2.21}$$

Hence the Covariance matrix is given by

$$\underset{\mathrm{N\times N}}{\hat{\mathrm{R}}(\omega,\theta)} = \underset{\mathrm{N\times D}}{\boldsymbol{A}(\omega,\boldsymbol{\theta})}\cdot\underset{\mathrm{D\times D}}{\hat{\mathrm{S}}}\cdot\underset{\mathrm{D\times N}}{\boldsymbol{A}(\omega,\boldsymbol{\theta})^{\mathrm{H}}} + \underset{\mathrm{N\times N}}{\sigma^2\cdot\boldsymbol{I}} = \hat{\mathrm{R}}_{\mathrm{S}}(\omega,\theta)+\hat{\mathrm{R}}_{\mathrm{R}}(\omega,\theta) \tag{2.22}$$

$\boldsymbol{I}$ is the N x N identity matrix. σ^2 is the noise variance in each channel if the noise is white, $\sigma^2 \boldsymbol{I}$ is the N x N covariance matrix of the noises. $\hat{\mathrm{S}}$ is the D x D Auto-covariance matrix of input signal.

For MUSIC to be applicable the input signals are assumed to be uncorrelated, so the covariance matrix $\hat{\mathrm{S}}$ will be a diagonal matrix having full rank D,

$$\hat{\mathbf{S}} = \mathrm{diag}\{\mathrm{P}_1, \ldots, \mathrm{P}_\mathrm{D}\} \tag{2.23}$$

where, $\mathrm{P}_\mathrm{k} = \mathrm{E}\left[|\mathrm{S}_{\boldsymbol{k}}(\omega)|^2\right], \quad \mathrm{k} = 1, \ldots, \mathrm{D}$.

P_k is the spectral power density of the k^{th} signal. $\hat{\mathrm{S}}$ will be positive-definite if and only if the *D* signal vectors are linearly independent.

Under these assumptions, $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$ is a positive semidefinite N x N matrix of rank D with

$$rank(\boldsymbol{A(\omega,\theta)} \cdot \hat{S} \cdot \boldsymbol{A(\omega,\theta)}^{H}) = span[\boldsymbol{a(\theta_1)}, \ldots, \boldsymbol{a(\theta_D)}] < N \tag{2.24}$$

Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\mathrm{N}$ denote the eigenvalues of $\hat{\mathrm{R}}$ and $\eta_1 \geq \eta_2 \geq \ldots \geq \eta_\mathrm{N}$ denote the eigenvalues of matrix $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$ respectively. From Eq. 2.22 we can easily see the following relation.

$$\lambda_\mathrm{i} = \eta_\mathrm{i} + \sigma^2, \quad \mathrm{i} = 1, \ldots, \mathrm{N} \tag{2.25}$$

Since the rank of $\hat{\mathrm{S}}$ is D and the number of sources D is smaller than the number of microphones N, the matrix $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$ is singular, i.e.

$$\det(\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}) = 0 \tag{2.26}$$

Eq. 2.26 implies that the D columns of $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$ span a D-dimensional subspace of N-dimensional complex space. This subspace is referred as Signal Subspace. The smallest (N-D) eigenvalues of $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$ are zero, i.e. $\eta_{\mathrm{D}+1} = \cdots = \eta_\mathrm{N} = 0$.

Determining the direction of arrivals for the no-noise is simply a matter of finding the D unique elements of $\boldsymbol{A(\omega,\theta)}$ that intersect the subspace. But if we consider the presence of the noise component $\sigma^2\mathbf{I}$ to the matrix $\boldsymbol{A(\omega,\theta)} \cdot \hat{\mathrm{S}} \cdot \boldsymbol{A(\omega,\theta)}^{\mathrm{H}}$, then the matrix $\hat{\mathrm{R}}$ will be of full rank. The noise component will not affect the corresponding eigenvectors because it simply increases all eigenvalues by same amount as shown in Figure 2.6[5] in next page.

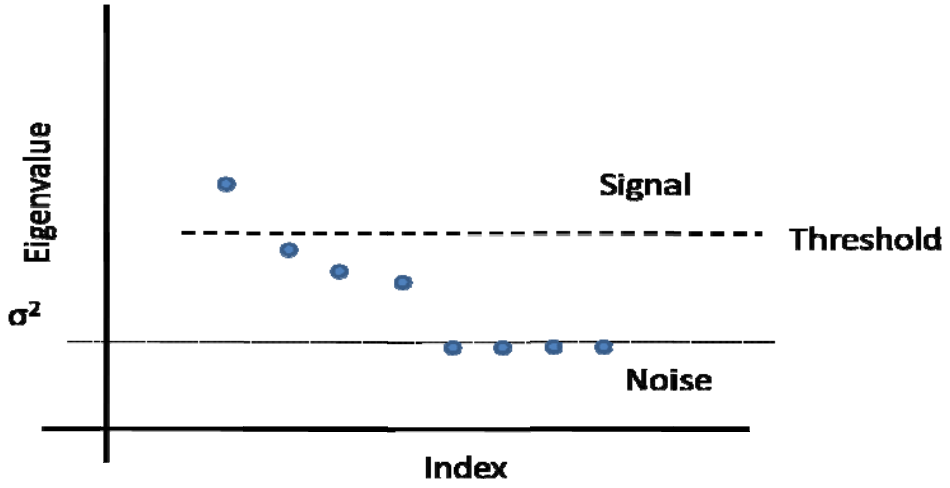

Figure 2.6 Eigenvalues representing signal and noise

Since N×N matrix $\hat{\mathrm{R}}$ has a set of linearly independent eigenvectors, the eigenvalue decomposition (EVD) can be performed as follows:

$$\hat{\mathrm{R}} = \boldsymbol{E}\cdot\boldsymbol{Q}\cdot\boldsymbol{E}^{H} = \left(\boldsymbol{E}_s\boldsymbol{E}_R\right)\begin{pmatrix}\boldsymbol{Q}_S & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{Q}_R\end{pmatrix}\begin{pmatrix}\boldsymbol{E}_S{}^{H}\\ \boldsymbol{E}_R{}^{H}\end{pmatrix} = \boldsymbol{E}_S\cdot\boldsymbol{Q}_S\cdot\boldsymbol{E}_S^{H} + \boldsymbol{E}_R\cdot\boldsymbol{Q}_R\cdot\boldsymbol{E}_R^{H}$$
$$= \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_{\mathrm{i}}\boldsymbol{e}_i\boldsymbol{e}^{H}{}_i = \sum_{\mathrm{i}=1}^{\mathrm{D}}(\eta_{\mathrm{i}}+\sigma^2)\boldsymbol{e}_i\boldsymbol{e}^{H}{}_i + \sum_{\mathrm{i}=\mathrm{D}+1}^{\mathrm{N}}\sigma^2\boldsymbol{e}_i\boldsymbol{e}^{H}{}_i \tag{2.27}$$

where, e_{i} is eigenvector of the matrix $\hat{\mathrm{R}}$. Because $\hat{\mathrm{R}}$ is a Hermitian matrix of full rank, each eigenvector e_{i} of $\hat{\mathrm{R}}$ is mutually orthogonal to each other, i.e.

$$\boldsymbol{e}_i{}^{H}\boldsymbol{e}_j = \delta_{\mathrm{ij}}, \text{ where } \delta_{\mathrm{ij}} = \begin{cases}1, & \mathrm{i}=\mathrm{j}\\ 0, & \mathrm{i}\neq\mathrm{j}\end{cases} \tag{2.28}$$

Because λ_{i} denotes the eigenvalue and $\boldsymbol{e}_i$ denotes the eigenvector of matrix $\hat{\mathrm{R}}$, this implies

$$\hat{\mathrm{R}}\boldsymbol{e}_i = \sigma^2\boldsymbol{e}_i, \quad \mathrm{i}=\mathrm{D}+1,\ldots,\mathrm{N}$$
$$\text{or} \tag{2.29}$$
$$(\hat{\mathrm{R}}-\sigma^2\mathrm{I})\boldsymbol{e}_i = 0, \quad \mathrm{i}=\mathrm{D}+1,\ldots,\mathrm{N}$$

Hence Eq. 2.22 can be rewritten as

$$\boldsymbol{A}(\omega,\boldsymbol{\theta})\cdot\hat{\mathrm{S}}\cdot\boldsymbol{A}(\omega,\boldsymbol{\theta})^{\mathrm{H}}\boldsymbol{e}_i = 0, \quad \mathrm{i}=\mathrm{D}+1,\ldots,\mathrm{N} \tag{2.30}$$

Since covariance matrix $\hat{\mathrm{S}}$ is a real, positive, full rank diagonal matrix, it follows that

$$\boldsymbol{A}(\omega,\boldsymbol{\theta})^{\mathrm{H}}\boldsymbol{e}_i = 0, \quad \mathrm{i}=\mathrm{D}+1,\ldots,\mathrm{N} \tag{2.31}$$

The above equation implies that the subspace spanned by the eigenvectors $\{\boldsymbol{e}_{D+1}, \boldsymbol{e}_{D+2}, \ldots \boldsymbol{e}_N\}$ is the orthogonal complement of the subspace spanned by the steering vector $\{\boldsymbol{a}(\omega,\theta_1), \boldsymbol{a}(\omega,\theta_2), \ldots, \boldsymbol{a}(\omega,\theta_D)\}$. This is represented as

$$span[\boldsymbol{e}_{D+1}, \ldots, \boldsymbol{e}_N] \perp span[\boldsymbol{a}(\omega,\theta_1), \ldots, \boldsymbol{a}(\omega,\theta_D)] \tag{2.32}$$

As mentioned above that the eigenvectors of the covariance matrix $\hat{\mathrm{R}}$ are orthogonal to each other, so $\mathbf{E}_S$ and $\mathbf{E}_R$ are orthogonal complement. This can be expressed as

$$\mathbf{E}_S = [e_1, \cdots, e_D]^T \perp \mathbf{E}_R = [e_{D+1}, \cdots, e_N]^T \tag{2.33}$$

Thus we can see that the columns of $\boldsymbol{E}_S$ span the D-dimensional signal subspace of N-dimensional complex space in the same way as the column vectors of matrix $\boldsymbol{A}(\omega,\boldsymbol{\theta})$ i.e.

$$span[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_D] = span[\boldsymbol{a}(\omega,\theta_1), \ldots, \boldsymbol{a}(\omega,\theta_D)] \tag{2.34}$$

The subspace spanned by the D eigenvectors corresponding to the D largest eigenvalues of $\hat{\mathrm{R}}$ is referred as the signal subspace. The noise space is the subspace spanned by the N-D eigenvectors of $\hat{\mathrm{R}}$ associated with the N-D smallest eigenvalues. The signal subspace and noise subspace are orthogonal complement to each other.

The direction of arrival can be determined by searching for the steering vectors which are orthogonal to the noise subspace, namely by finding vectors on the array manifold that have zero projection in the noise subspace. In practice, $\hat{\mathrm{R}}$ is unknown, but it can be estimated from the available data as in Eq. 2.22. But in real time consideration, there are many errors which are unavoidable, thus one can only search for the steering vectors which are most closely orthogonal to the noise subspace.

To obtain a mathematical measure of closeness to orthogonality, it is beneficial to define an orthogonality error vector $\boldsymbol{\varepsilon}(\boldsymbol{a}(\omega,\boldsymbol{\theta}))$ whose kth element is the inner product of $\boldsymbol{a}(\omega,\boldsymbol{\theta})$ and the kth eigenvector $\boldsymbol{e}_k$ of the noise subspace. Thus the error vector $\boldsymbol{\varepsilon}(\boldsymbol{a}(\omega,\boldsymbol{\theta}))$ can be written as

$$\boldsymbol{\varepsilon}(\boldsymbol{a}(\omega,\boldsymbol{\theta})) = [\varepsilon(D+1), \cdots, \varepsilon(N)]^T = [< \boldsymbol{a}(\omega,\boldsymbol{\theta}), \boldsymbol{e}_{D+1} >, \cdots, < \boldsymbol{a}(\omega,\boldsymbol{\theta}), \boldsymbol{e}_N >] \tag{2.35}$$

The Euclidean norm of $\varepsilon(\boldsymbol{a}(\omega,\boldsymbol{\theta}))$ is

$$\left\|\boldsymbol{\varepsilon}(\boldsymbol{a}(\omega,\boldsymbol{\theta}))\right\|^2 = \left(\left|\varepsilon(D+1)\right|^2 + \cdots + \left|\varepsilon(N\right|^2\right)$$

$$\left\|\boldsymbol{\varepsilon}(\boldsymbol{a}(\omega,\boldsymbol{\theta}))\right\|^2 = \sum_{i=D+1}^{N} \left|\boldsymbol{e}_i^{\boldsymbol{H}} \cdot \boldsymbol{a}(\omega,\boldsymbol{\theta})\right|^2 = \boldsymbol{a}^{\boldsymbol{H}}(\omega,\boldsymbol{\theta}) \cdot \mathbf{P} \cdot \boldsymbol{a}(\omega,\boldsymbol{\theta}) \tag{2.36}$$

$$\text{Where } \mathbf{P} = \boldsymbol{E}_{\boldsymbol{R}} \boldsymbol{E}_{\boldsymbol{R}}^{\boldsymbol{H}}$$

The minimum squared Euclidean norm associated with a steering vector $\boldsymbol{a}(\omega,\boldsymbol{\theta})$ that is an optimum one. The direction steered by the optimum steering vector is the true

direction of arrival. The MUSIC algorithm estimates the DOAs as the peaks of the MUSIC spectrum as

$$\hat{S}_{MUSIC}(\omega,\theta)=\frac{1}{\boldsymbol{a}^{H}(\omega,\theta)\cdot\mathbf{P}\cdot\boldsymbol{a}(\omega,\theta)}=\frac{1}{\boldsymbol{a}^{H}(\omega,\theta)\boldsymbol{E}_{R}\boldsymbol{E}_{R}{}^{H}\boldsymbol{a}(\omega,\theta)} \tag{2.37}$$

2.3 Wideband Array Processing

The sub decomposition methods were initially developed for narrowband signal localization and are not applicable to wideband cases. To solve this problem many approaches were suggested. One of them is by using conventional sub-band techniques, but these techniques do not offer any improvement. Another approach is to apply narrowband filters to the signals and then treat them separately as narrowband problem. If we analyze a speech spectrum of a wave file as shown in Figure 2.7. We can see that the speech signals have significant power over a wide range of frequencies and also speech signals exhibit formant frequencies.

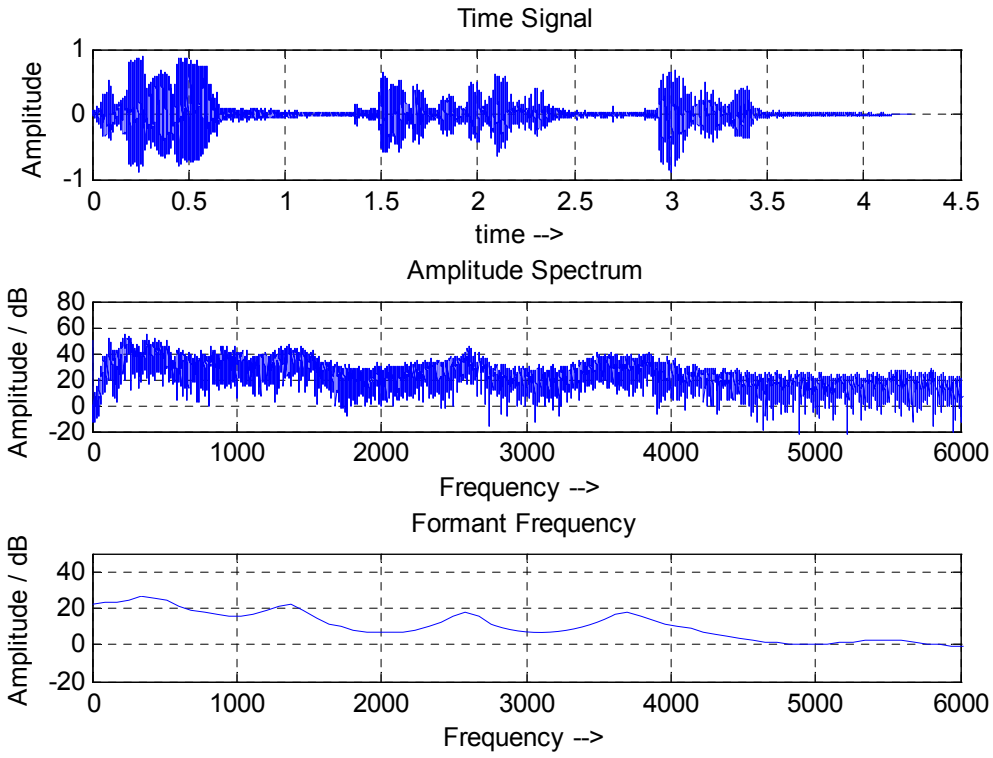

Figure 2.7 Speech spectrum of a woman's voice

In the whole spectrum, there are specific frequencies which exhibit more power than other frequencies. It makes sense to use these frequencies for estimating the direction of arrival.

One of approach is suggested by Su & Morf [6], according to them narrowband high resolution subspace methods like MUSIC can be used to determine narrowband beampatterns over many temporal frequencies and then combine them. The only condition is that there should be sufficient or high SNR in multiple frequency bins so that narrowband methods can yield good results independently for each bin. Another assumption is that every look angle has only one target, so that signal subspace consists of one eigenvector, with the other N-1 eigenvectors forming the noise subspace [7]. This way one can apply faster eigen-analysis algorithms.

Based on this approach Incoherent or Coherent wideband array processing techniques can be employed for detection and tracking of audio source. Figure 2.8 shows the basic principle of Incoherent method.

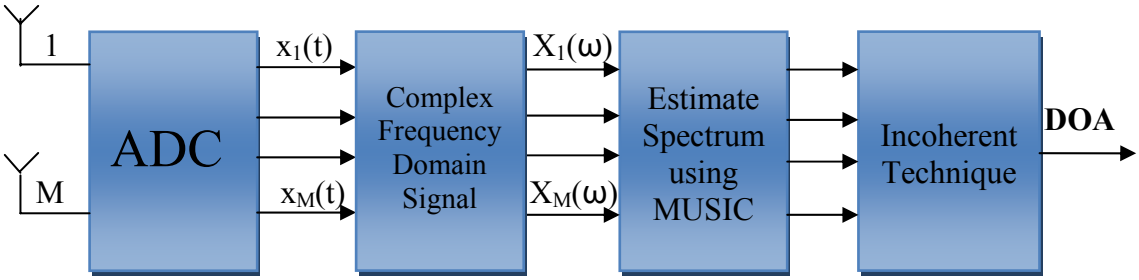

Figure 2.8 Basic Principle of Incoherent Method

The basic steps in both techniques are as follows:

i. Use block-adaptive pre-processing to adaptively select the narrowband frequency bins.
ii. Apply MUSIC algorithm, and apply Incoherent or Coherent techniques.
iii. Estimate the directions of the sources from the resulting beam form.

The difference between the two techniques is of computational complexity. For incoherent the computational complexity is given in Eq. 2.38 and for coherent is *given in Eq. 2.39*,

$$M[O(N^2) + O(N^3) + S * O(N^2)] \quad (2.38)$$

$$S[M * O(N^2) + O(N^3) + O(N^2)] \quad (2.39)$$

where *M* is the number of frequency bins and *S* is the number of look angles.

The first squared term in the bracket corresponds to the formation of the correlation matrix; the cubic term is for a SVD (Singular Value Decomposition) calculation to perform Eigenvalue Decomposition as EVD and SVD both gives the same result (will be discussed in Chapter 4.2 in detail) and last term for computation of every look angle. For both methods, the most expensive computational cost is the SVD which is $O(N^3)$. This term defines the complexity of algorithms and we can see that for incoherent it is $M*O(N^3)$ while for coherent it is $S*O(N^3)$.

For real time applications, there is no *priori* knowledge of the source directions. For coherent wideband processing, microphone array scans in all direction which means it requires a very large number of SVD calculation. However for incoherent wideband processing, the number of SVD calculations depends on the number of frequency bins which is actually just a fraction of SVD calculations required in coherent wideband processing. Therefore for low cost, low power DSP processor like TI's C6713, coherent technique is quite computationally intensive. The incoherent wideband technique is somewhat more suited for our application.

The steps are explained as follows:

1) The first step is to overcome the non-stationary nature of the source. This is being done by segmenting the data in fixed size of blocks. Therefore the samples are collected from each channel of ADC and are stored in terms of data blocks for further processing.
2) The second step involves the conversion of real Time domain signal to complex Frequency domain as the algorithm works in Frequency domain. This is being done by performing FFT for each data blocks.

$$\mathbf{X}_i(\mathbf{t}) = A(\theta) \cdot s_i(\mathbf{t}) + \mathbf{w}(\mathbf{t})$$

$$\downarrow$$

$$\mathbf{X}_i(\omega_k, \theta) = A(\omega_k, \theta) \cdot \mathbf{S}_i(\omega_k) + \mathbf{W}(\omega_k)$$

3) The third step is to compute the average power spectrum and then adaptively select the M frequency components for wideband processing.

4) Next step is to form the estimated narrowband spatial correlation matrix $\hat{R}_x(\omega_k,\theta)$ for every adaptively selected frequency ω_k for k = 1,2,….M.

5) In this step narrowband MUSIC algorithm is applied for each spatial correlation matrix $\hat{R}_x(\omega_k,\theta)$.

6) In this second last step all beam-patterns or pseudospectrum are incoherently averaged together as shown in Figure 2.40 to give resultant spectrum,

$$\hat{P}_{Incoherent}(\omega_k,\theta)=\sum_{k=1}^{M}\frac{1}{\boldsymbol{a}^{H}(\omega_k,\boldsymbol{\theta})\boldsymbol{E_R}\boldsymbol{E_R}^{H}\boldsymbol{a}(\omega_k,\boldsymbol{\theta})} \tag{2.40}$$

7) The last step is to employ a Tracker algorithm to determine the direction of arrival.

The flow chart for implementing the Incoherent Wideband MUSIC algorithm is shown in Figure 2.9 on next page.

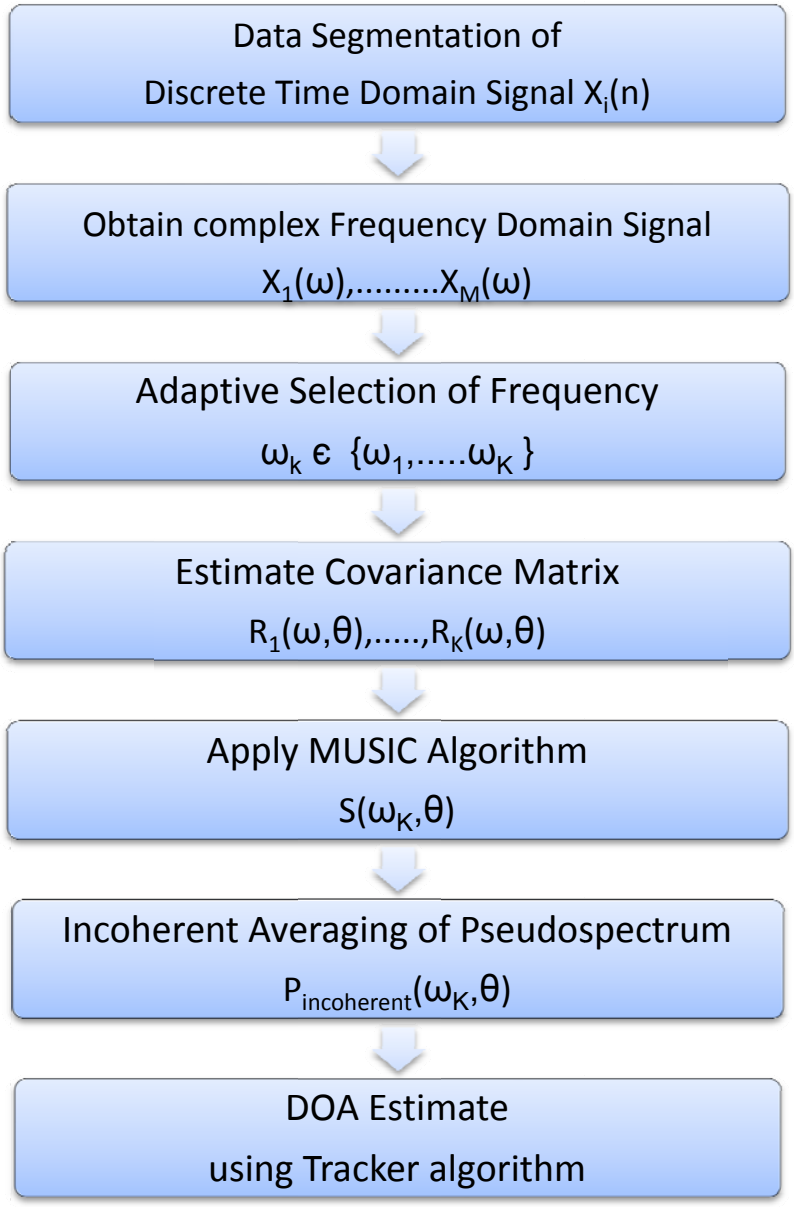

Figure 2.9 Incoherent Wideband MUSIC Algorithm

3. DSP-Based Real time System

Microphone array technology has been proposed for various speech applications. By localizing the direction of arrival of desired speech source, attenuating background noises and rejecting discrete spatial interferes, a microphone array can enhance the SNR in a noisy environment with notable improvement in speech intelligibility. The Microphone array is constructed with 8 microphones with pre-amplifying circuit. The output of pre-amplifying circuits are connected to the ADC channel of the PCM Codec, which in turn is connected to the DSK 6713 through an adapter. Figure below shows block diagram of Hardware connections.

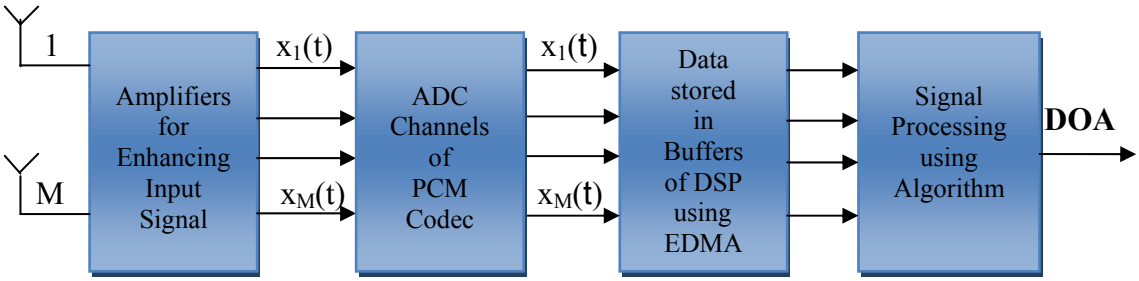

Figure 3.1 Main blocks of DSP Sub-system

In first section, Microphone array system is described and then in the next sections the subsystems of DSP and interfacing between them is discussed.

3.1 Microphone Array

In this section the properties of microphone and the construction of microphone array is described. An omnidirectional elektret condenser microphone capsule WM-52BT is used in the application. It is a low voltage, 1.5 volt operation microphone and suitable for all voice applications as it has frequency range of 20 - 16000 Hz [8].

One of the important things is having an ideal flat frequency response, so that microphone will remain equally sensitive to the whole frequency range of interest. For this application the most desirable thing is having a flat frequency response through the whole spectrum and the microphone used in the application has good flat frequency response. A typical flat frequency response curve [8] is shown in Figure 3.2 in next page.

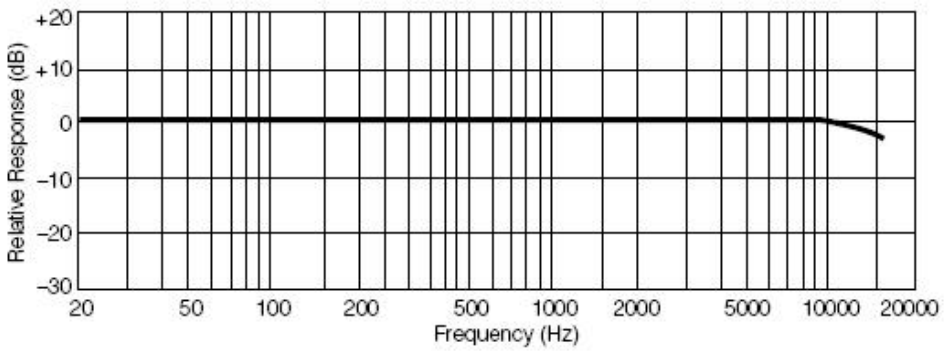

Figure 3.2 Flat Frequency Response Curve for WM-BT52

The internal circuit diagram of this condenser is shown in Figure 3.3.

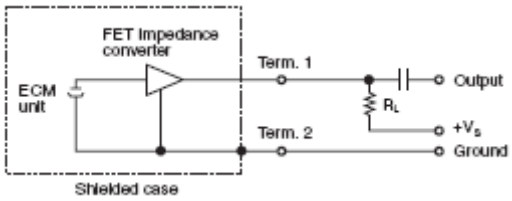

Figure 3.3 Schematic Diagram of WM-52BT

The output of microphone is less than 100 mV and varies across microphones whereas the required input voltage for ADC of PCM 3003 Codec is 1.8 V_{pp}; therefore the signal needs to be amplified before feeding into the PCM Codec. To match the incoming signal with the input characteristics of the PCM Codec, pre-amplifying circuit is used. Non-uniform delays and gains among different microphones may lead to sub-optimum processing of the receiving waveforms. Therefore microphones need to be calibrated. The calibration of microphones will be discussed in Chapter 5. The circuit diagram of the pre-amplifier used is shown in Figure 3.4.

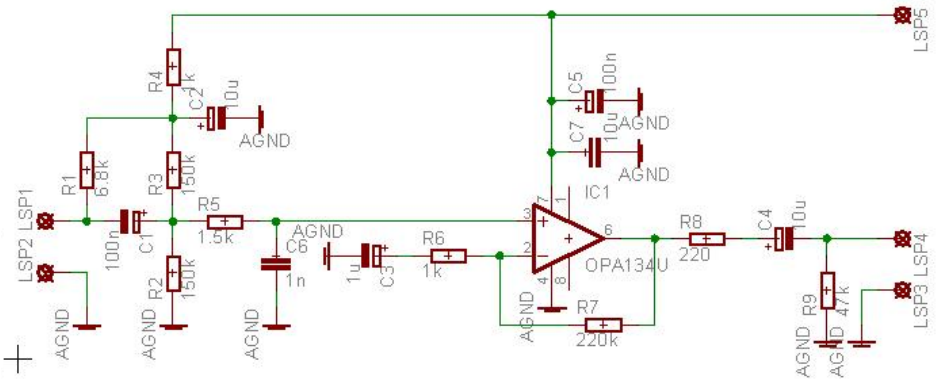

Figure 3.4 Amplifying Circuit Diagram for Microphone

The ADC channel of I/O card PCM3003 has the input voltage range of 0 - 3.3 Volt. But the output voltage from microphone varies from negative to positive. Therefore an offset is provided to shift the range to 0 – 3.3 Volt which will be removed from the data segment to get a DC free input signal. As 8 microphones are used in the application, they are connected to 8 ADC channel from 1 to 8. The Figure 3.5 below shows the pictorial view of microphone's amplifying PCB (courtesy to DSP Lab, Hamburg University of Applied Science) used in this project.

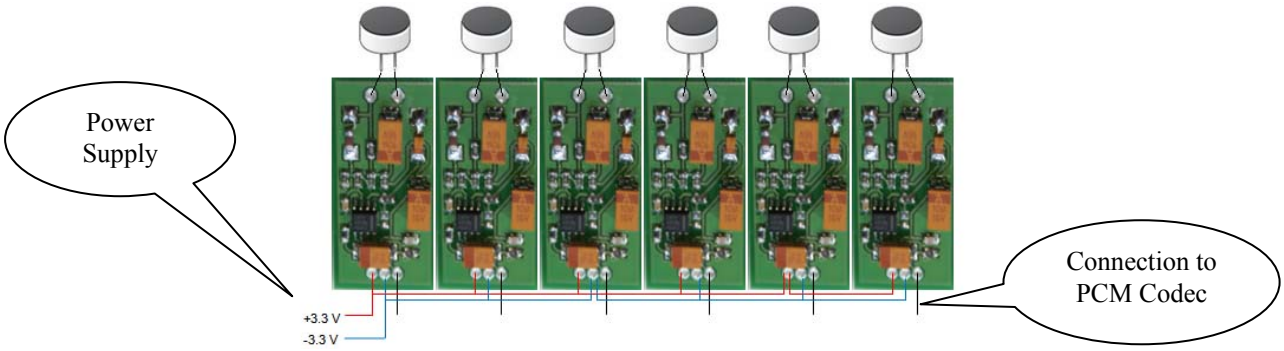

Figure 3.5 Pictorial view of Microphone Array's amplifying circuit

3.2 DSP Sub-System

The DSK 6713 includes DSP board (TI's C6713) with I/O Card (PCM 3003). The complete pictorial setup of DSK 6713 is shown in Figure 3.6.

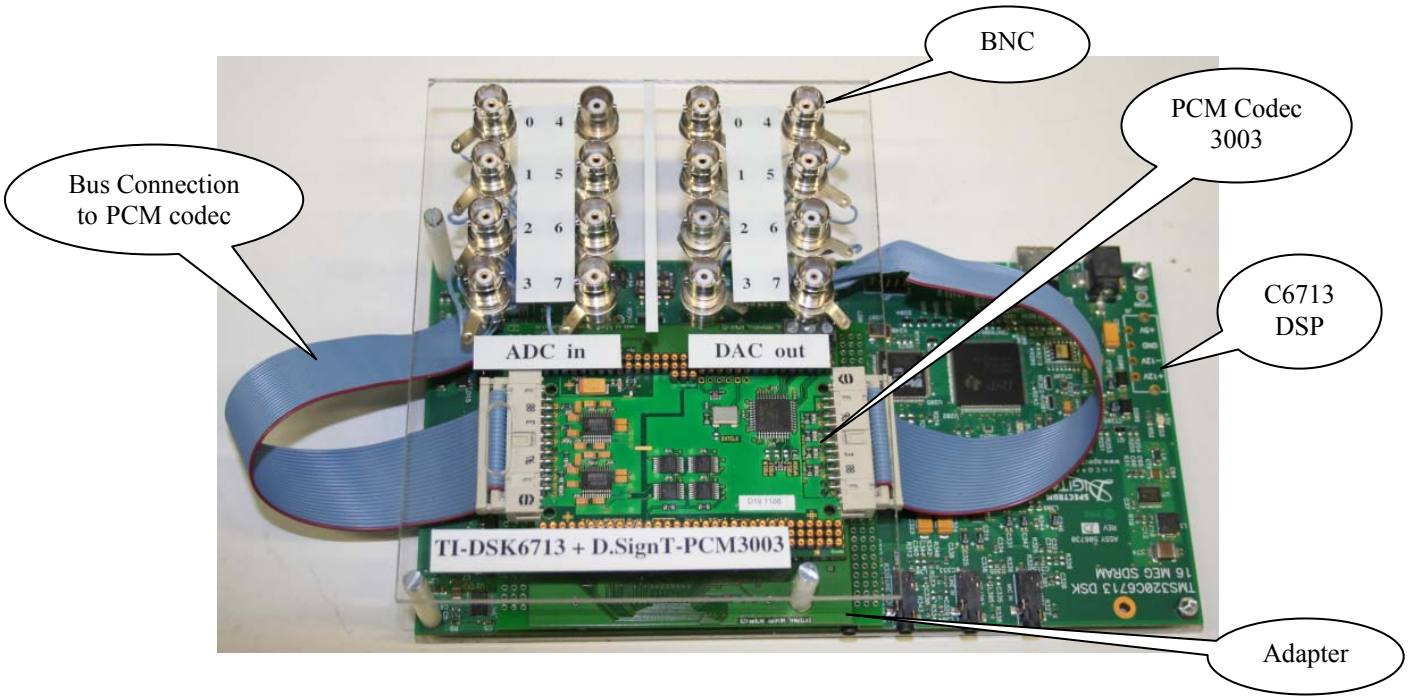

Figure 3.6 Complete setup of DSP Sub-System

3.2.1 PCM 3003 Audio Daughter Card

In this application 8-channel Audio daughter card PCM 3003 [10] from D.SignT is used. The four PCM 3003 Codec are connected via a special adapter board with serial ports McBSPs of DSK 6713. Clocks and frame synchronization signals are generated by the PCM 3003 and acts as a master device. The ADCs employ delta-sigma modulation with 64x oversampling. The ADC includes a digital decimation filter and the DAC include a digital interpolation filter. The codec operate with left-justified and right-justified formats and has 16 bit resolution. The codec has the sampling frequency from 8 – 48 KHz and all ADC operate synchronously with the sampling frequency. The block diagram of the PCM3003 [9] is shown in Figure 3.7.

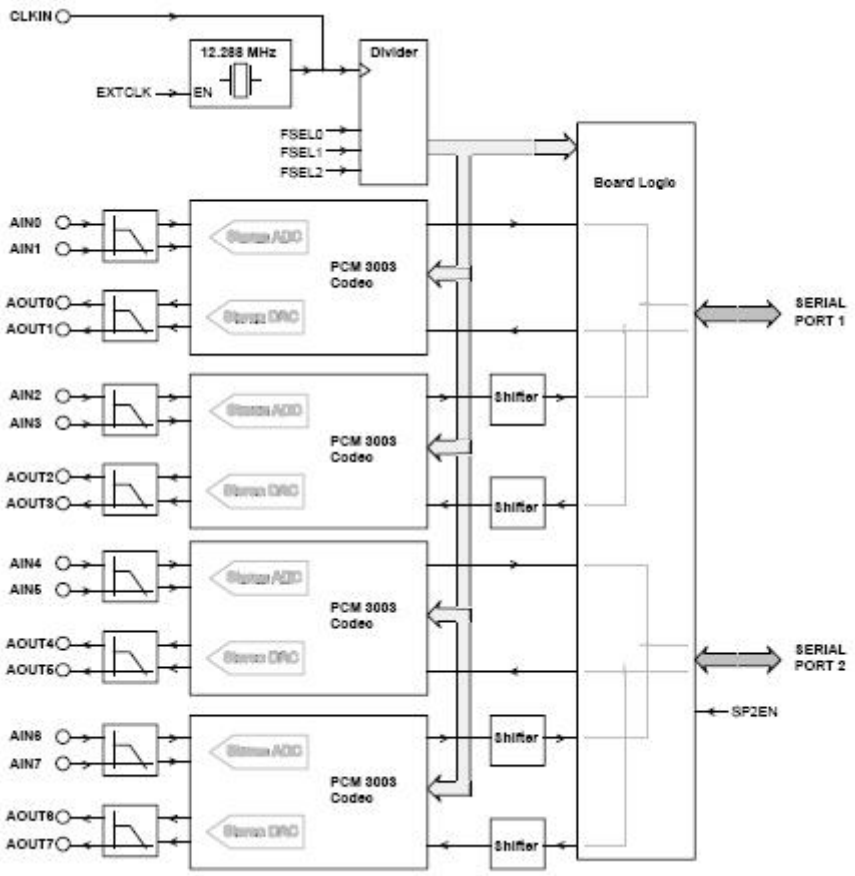

Figure 3.7 Block Diagram of PCM 3003 Card

3.2.2DSP:TMS320C6713

TMS320C6000 devices are first DSP processors that use an enhancement of the Very

Long Instruction Word (VLIW) architecture which is very well suited for numerically intensive algorithms as it allows achieving high performance through instruction level parallelism. The internal program memory is structured in such a way, so that a total of eight instructions can be fetched every cycle.

In this project, 225 MHz TMS320C6713 floating point DSP is used, which at the clock rate of 225 MHz can process information at a rate of 1.35 Giga-floating-point operations per second (GFLOPS). The functional block diagram [10] of C6713 is shown in Figure 3.8.

It has the following features:

- Memory
  - 16 MB SDRAM
  - 512 KB Flash Memory
  - 264 KB Internal Memory
- General purpose I/O
  - 4 LEDs
  - 4 DIP Switches
- Eight Execution Units composed of six ALUs and two Multiplier Units
- 32-bit External Memory Interface
- Multichannel Buffered Serial Port
- USB interface to PC

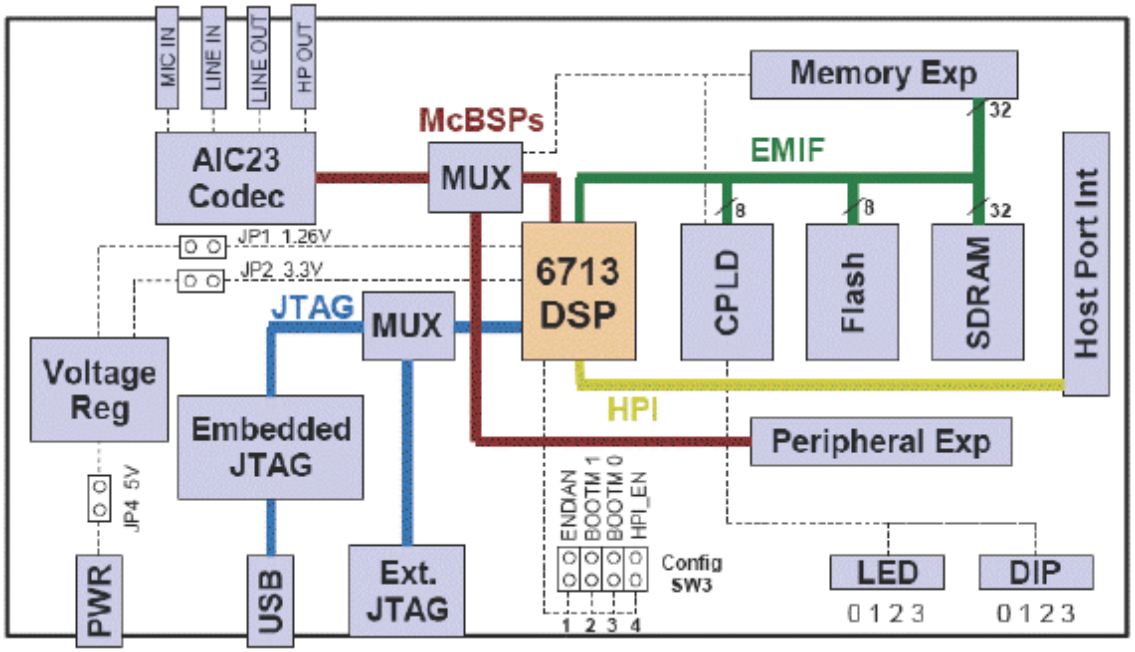

Figure 3.8 Functional Block Diagram of C6713

3.3 Code Composer Studio

Code Composer Studio (CCS) provides an integrated development (IDE) for real-time DSP applications based on the C programming language. It incorporates a C compiler, an assembler and a linker. It has good graphical capabilities and supports real-time debugging. The software development flow [11] for C6713 is shown in Figure 3.9.

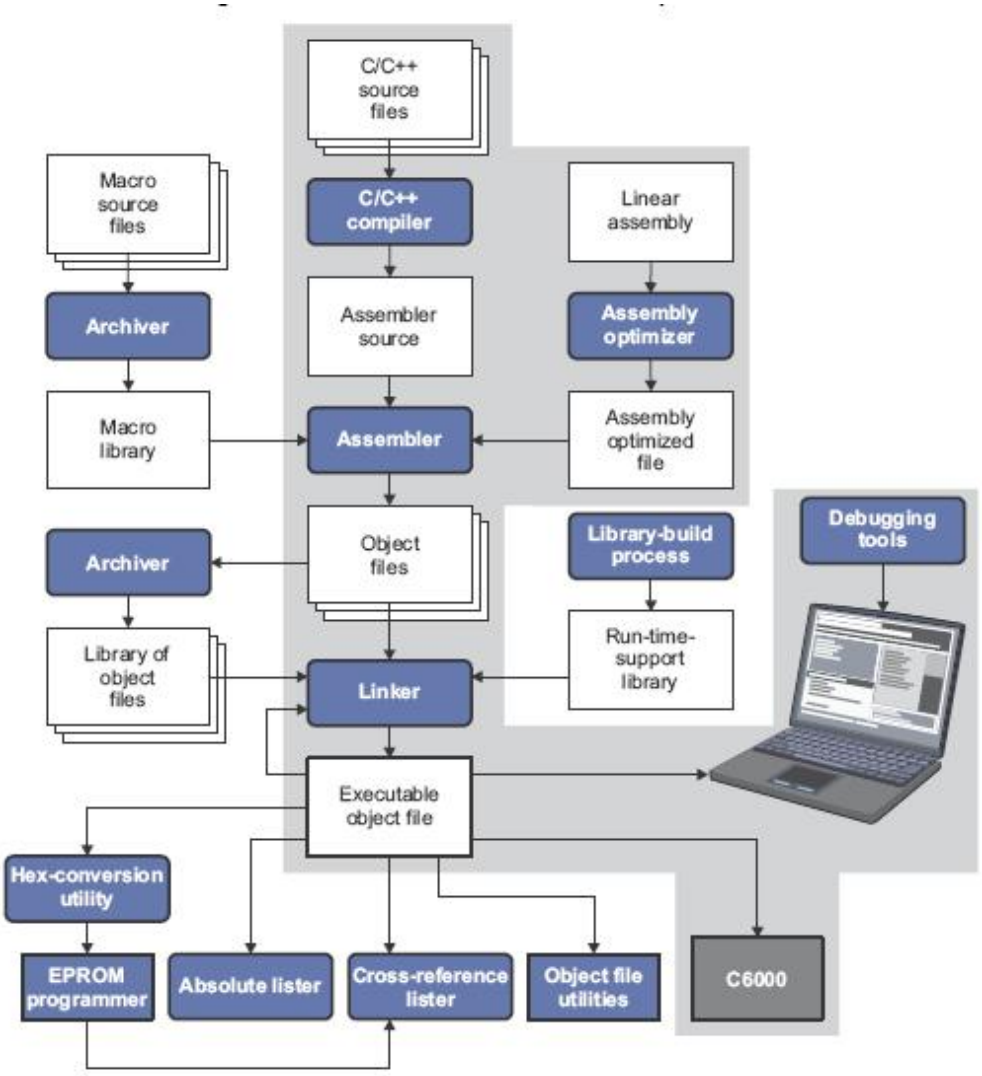

Figure 3.9 TMS320C6713 Software Development Flow

3.4 Interfacing between DSP and PCM 3003

A DSP motherboard may make use of some or all of the signals presented by the PCM Codec. For interfacing between DSK 6713 and PCM3003 codec, DSK board has two McBSPs and each one of them is connected to the PCM 3003 codec via six signals. The McBSP consists of a data path and a control path that is connected to the PCM codec.

Data communication between the device and the McBSP takes place via the data transmit (DX) pin for transmission and via the data receive (DR) pin for reception. Control information (clocking and frame synchronization) is communicated via CLKS, CLKX, CLKR, FSX, and FSR. The Figure 3.10 shows functional block diagram taken from spra488c [14].

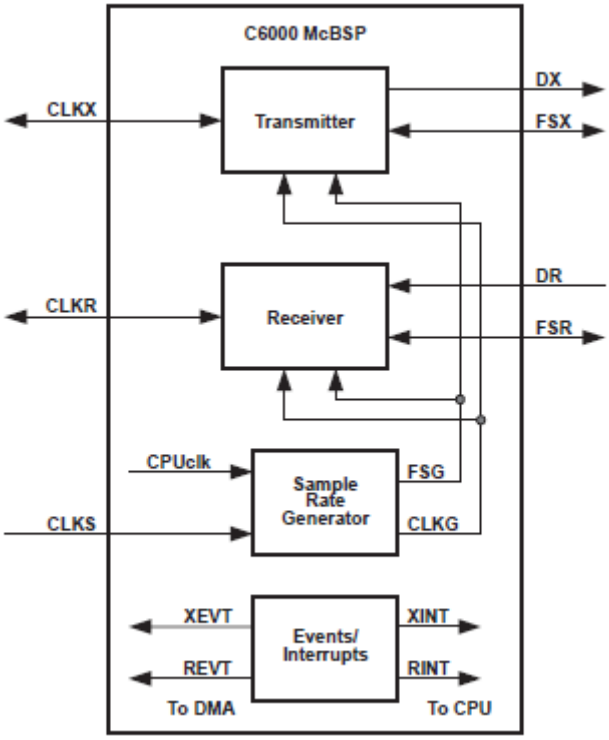

Figure 3.10 Function Block Diagram of McBSP

The blocks are explained as follows:

Transmitter: The data to be transmitted is written in DXR using EDMA and the contents of this register are copied to the XSR (transmit shift register). The transfer starts as soon as the FSX (transmit frame sync) is detected and one bit of data is shifted out of XSR on every transmit clock CLKX.

Receiver: The data received on the DR pin is shifted into the RSR (receive shift register) on every receive clock (CLKR). The data in RSR is copied to RBR (receive buffer register) and then to DRR (data receive register).

Sample Rate Generator: Here control signals such as clocks (CLKR/X) and frame sync (FSR/X) are generated. Both are bidirectional pins and can be used as input or output.

Events/Interrupt Generation: The McBSP generates sync events to the EDMA to

indicate that data is ready in DRR or that DXR is ready for new data. They are REVT (read sync event) and XEVT (write sync event). In the same way CPU can read/write to the McBSP based on interrupts generated by the McBSP.
For example in establishing the data transmission between the PCM 3003 and McBSP0, the DAT_RX0 is connected to DRR0 (Serial Port Receive Data), CLK_RX0 is connected to CLKX0 (Serial Port Transmit Clock) and FS_RX0 is connected to FSR0 (Serial Port Receive Frame) as shown in Figure 3.12.
PCM 3003 multiplexes the data outputs of two ADC onto a single serial data input to the C6713 and similarly a single serial data output of the C6713 is de-multiplexed and feeds the inputs of two DAC.
The serial transmission sequence between PCM 3003 and C6713 is not an actual serial channel number. The transmission sequence is as follows:

First Codec Left audio channel	→	Channel 0
Second Codec Left audio channel	→	Channel 2
First Codec Right audio channel	→	Channel 1
Second Codec Right audio channel	→	Channel 3
Third Codec Left audio channel	→	Channel 4
Forth Codec Left audio channel	→	Channel 6
Third Codec Right audio channel	→	Channel 5
Forth Codec Right audio channel	→	Channel 7

Because of the existence of a frame sync pulse at the starting of both left and right channel in serial data format it is difficult to distinguish between them. Therefore the channels need to be tracked while developing the software. Because all codec operate synchronously, only the McBSP0 interrupt is used. The McBSP1 transmitter is processed along with McBSP0. The functioning of McBSPs is shown in Figure 3.11.

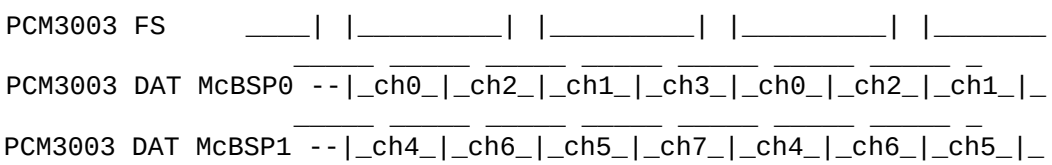

Figure 3.11 Accessing of data through McBSPs

One of the software objectives is to collect the block of samples from ADC channels of the Codec into the DSP as soon as possible and the most efficient way to do this is through EDMA (Enhanced DMA). The multichannel buffered serial ports (McBSPs) are the only on-chip peripherals that would require servicing by the EDMA. As already explained each McBSP has a data receive register (DRR), a data transmit register (DXR), a receive-event signal (REVT), and a transmit-event signal (XEVT). The DRR and DXR are memory-mapped registers, and the events are set when data is transferred in to (REVT) or out of (XEVT) the McBSPs respectively. The Data transfer block diagram between the devices using EDMA and McBSP0 is shown in Figure 3.12.

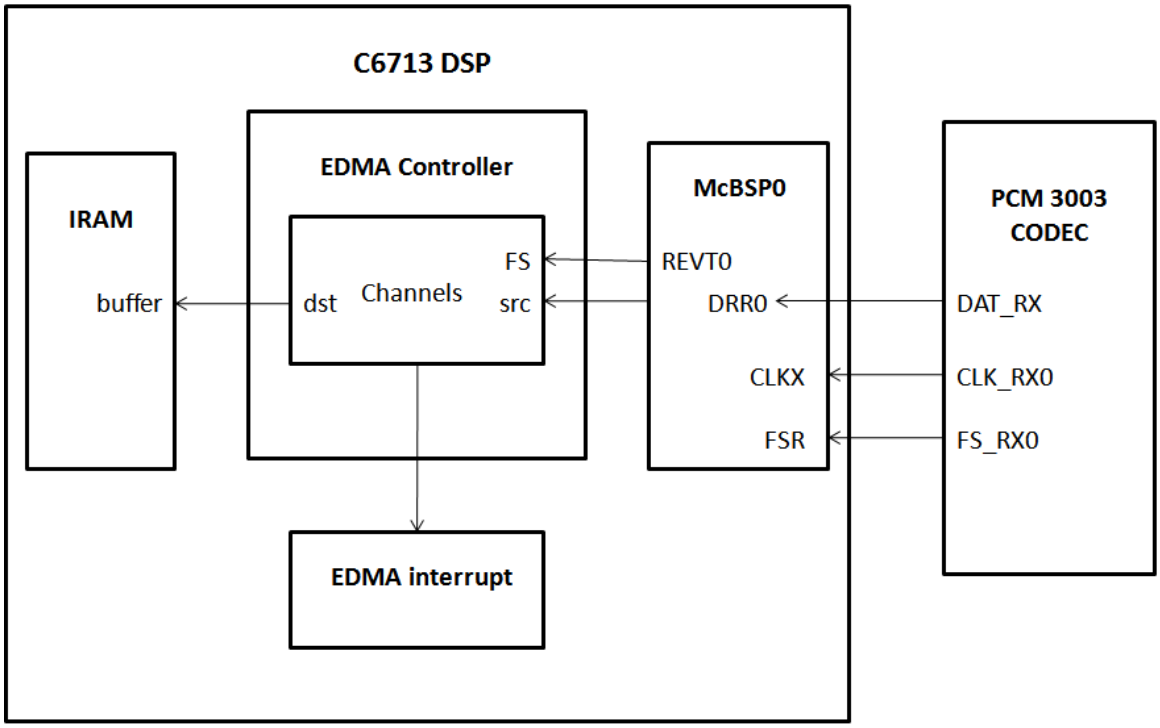

Figure 3.12 Data Transfer Block Diagram using EDMA

In this application a demo program provided by the Dsign.T is used for the transmission of samples from PCM codec to the DSK 6713. This demo program did the initialization and has configured the PCM Codec with the MCBSPs of C6713. For more information regarding configuration please refer to SPRA488C [14].

As EDMA transfer samples from codec to DSP the CPU still performs its operation during this time and will only be interrupted by EDMA when the transmission of data is finished and ready for processing. Hence the CPU is free from the data transmission

control, thus increases the speed and efficiency of the application. The frequency of interruption depends on the block size, less the block size more will be the interruption or vice-versa. Figure 3.13 shows the data transmission through EDMA.

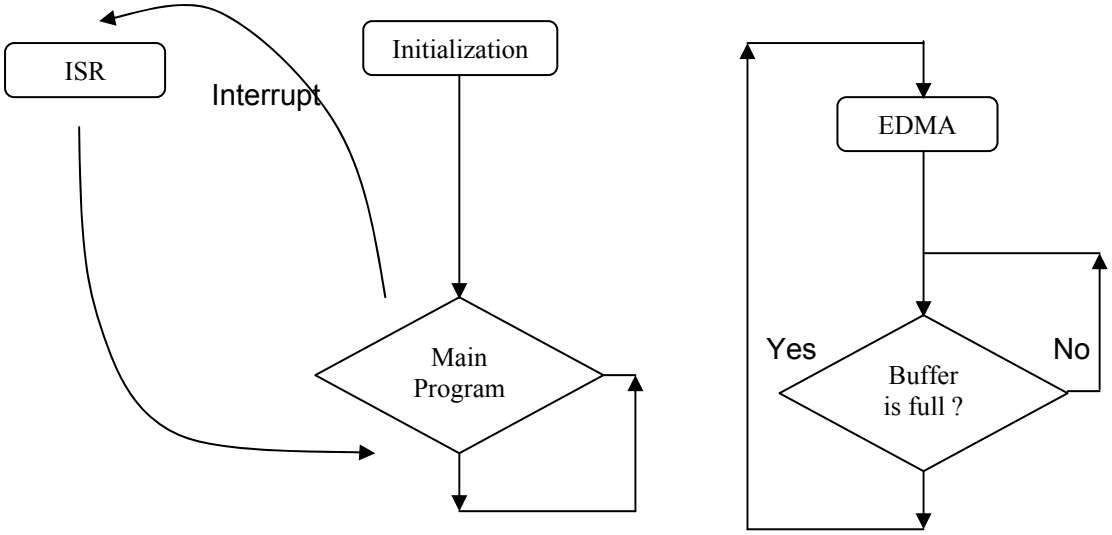

Figure 3.13 Data transmission through EDMA

The transmission of data can be summarized as follows: The program sets the appropriate values in the EDMA registers and starts the EDMA. Then it will wait for the EDMA interrupts which indicates that the data block is ready. Once an interrupt occurred, processing will start. When the processing is finished, the program will start processing on new set of data samples and the process will repeat itself.

The function flowchart of this demo program for software interfacing between PCM 3003 Codec and C6713 DSP is shown in Figure 3.14.

Main()

Reset PCM3003 Codec
Initialize data arrays
Configure McBSPs
Configure EDMA channels
Setup EDMA Controller Interrupt
Enable EDMA Complete Interrupt
Take PCM3003 Codec out of RESET
Main Program Loop

Figure 3.14 Function Flowchart of main program

Although the configuration presented here allows the EDMA to transfer samples but it presents a number of restrictions to the CPU. Since the buffers for each channel are continuosly filled and emptied by EDMA, the CPU must match its pace in processing the data. If the algorithm to be implemented is numerically intensive, which is mostly the case there is a very good chance of mismatch between the two's pace. To avoid this problem a simple technique called PingPong buffering technique [13] is used in this demo program, which allows the CPU activity to be decoupled from the EDMA activity. This simply means that there are two sets of data buffers for all incoming data streams. While the EDMA is transferring data in to the ping buffers, the CPU is manipulating the data (copy data and process) in the pong buffers. The whole process can be explained as follows; the EDMA controller reads audio data from the McBSP and places it in a buffer in memory. On the data receive side there are two logical buffers, PING and PONG. When the first data comes in, it is placed in the PING buffer. When it is full the new data is redirected to the PONG buffer and the DSP is free to process the PING data without fear of it being overwritten. When the PONG buffer fills up, the configuration is reversed. The ping-pong data transfer continues indefinitely with one buffer always hosting the active transfer and one remaining stable for the DSP to operate on. Using PingPong buffering technique, the DSP can take as long as the time it takes to fill an entire buffer to process the data, making it much easier to meet a real-time schedule. Hence for every channel there are in effect three buffers namely Ping buffer, Pong buffer and Process buffer. The ping-pong scheme with EDMA is shown below in Figure 3.14.

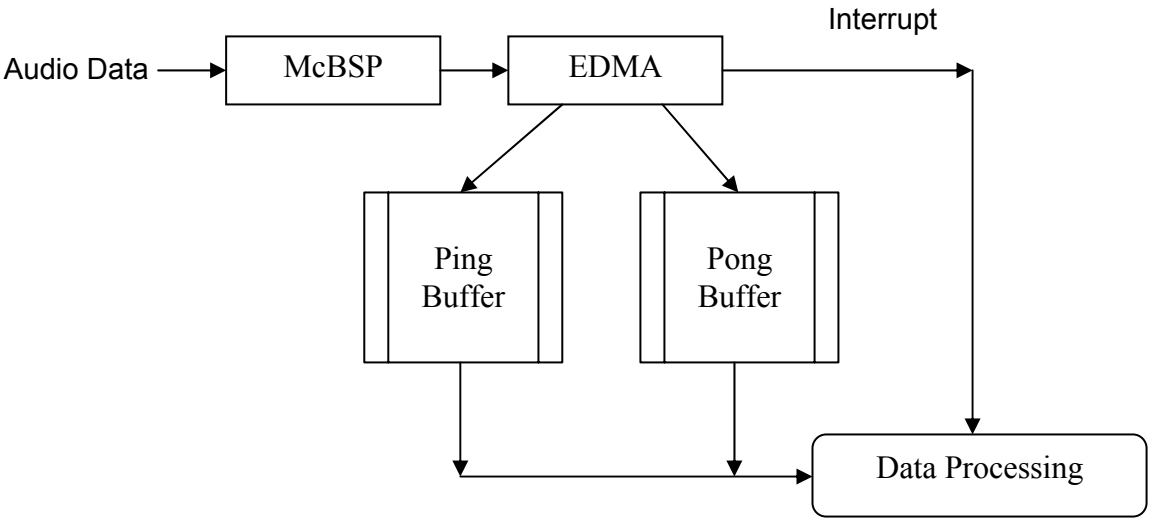

Figure 3.15 Ping-Pong Buffering for McBSP DATA

The 2-d buffer structure being employed in the demo program for this interface is shown in Figure 3.16. Each channel uses two blocks (0 & 1). While EDMA receives from block0, the CPU can copy and process block1 and vice-versa.

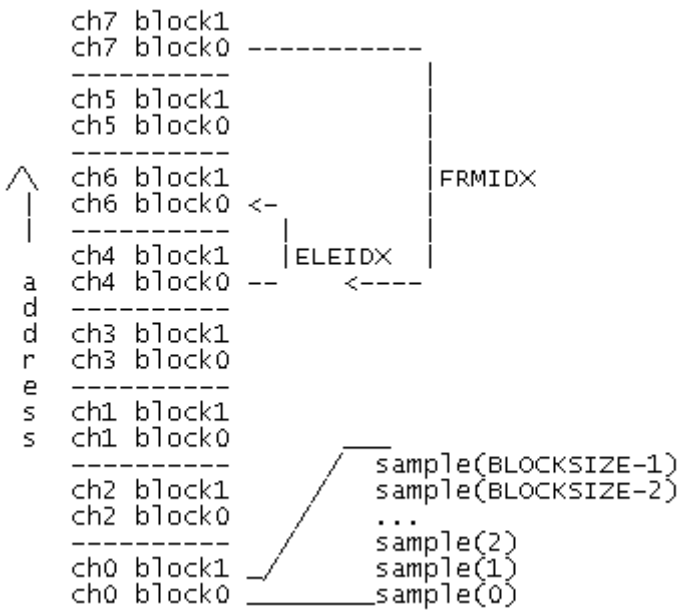

Figure 3.16 Ping-Pong Buffer Structure

4. Algorithm: Methods and Simulation

The main motive of the Incoherent Wideband MUSIC Algorithm is to pick the dominant/resonant frequencies of the source in wideband spectrum and apply the narrowband MUSIC algorithm separately for each frequency component and finally find the mean of separately calculated MUSIC spectrum to determine the approximate direction of arrival. The picking of peaks with respect to dominant frequency in a spectrum can be achieved by Linear Prediction Analysis or Spectrogram (average threshold method) method. After finding the peaks, SVD (Singular Value Decomposition) method is applied on output matrix obtained from microphones with respect to peaks to separate the signal subspace from noise subspace. The direction of arrival is approximated by averaging the separately calculated Narrowband MUSIC spectrum. In this chapter the methods to find the dominant frequencies in a speech signal and after that SVD method is discussed. Later on the influence of parameters on algorithm is analyzed and respective simulation results are presented.

4.1 Spectral Analysis

Spectral analysis [38] is a signal processing technique aimed at presenting the dynamic patterns of signals in the frequency domain. In the coming subsection two of the most commonly used method for spectral analysis based on Fourier methods is discussed.

4.1.1 Linear Predictive Analysis

LPC is one of the most powerful speech analysis technique [15] and also one of the most useful methods for encoding good quality speech at a low bit rate. This method exploits the predictable nature of speech signals. Linear prediction models the human vocal tract as an infinite impulse response system that produces the speech signal (Figure 4.1). For vowel sounds and other voiced regions of speech, which have a resonant structure and high degree of similarity over time-shifts that are multiples of their pitch period, this modeling produces an efficient representation of the speech.

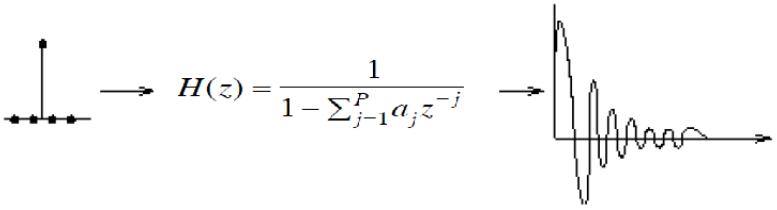

Figure 4.1 Linear Prediction Model of Speech

In this method the basic step is to find the formant or dominant frequency and that can be determined using difference equation, which expresses each sample of the signal as a linear combination of previous samples.

The linear prediction method can be stated as finding the coefficients a_j which result in the best prediction of the speech sample y[n] in terms of past sample y [n-j], j = {1,…..,P}. The predicted sample $\tilde{y}$ [n] is then given as

$$\tilde{y}\,[n] = \sum_{j=1}^{P} a_j\, y\,[n-j] \tag{4.1}$$

where P is the number of past samples of y [n].

These coefficients of the difference equation also called prediction coefficients characterize the formant frequency. To estimate these coefficients the LPC system minimizes the mean square error between the predicted signal and actual signal.

These coefficients are used to obtain the frequency response of the speech signal. In Eq. 4.1, when the predicted sample equals the actual signal then we have

$$y\,[n] = \sum_{j=1}^{P} a_j\, y\,[n-j] \tag{4.2}$$

$$H(z) = \sum_{j=1}^{P} (a_j\, H(z)\, z^{-j}) \tag{4.3}$$

$$H(z) = \frac{1}{1-\sum_{j=1}^{P} (a_j\, z^{-j})} \tag{4.4}$$

The Eq. 4.4 gives a direct implementation of the spectral model as an all-pole filter in the complex z domain.

Simulations are run in Matlab to see the performance of this method to determine the resonant frequency components present in the speech. For this purpose a wav file (TheForce.wav provided by DSP Lab, HAW Hamburg) is used.

Matlab function ‘lpc’ from Matlab Signal Processing Tool Box is employed to determine the linear prediction coefficients (lpf) of the P^{th} order forward linear predictor and then ‘freqz’ is used to find the frequency response.

```
lpf=lpc(Y,P);
[h,f]=freqz(1,lpf,N,fs);
```

It is very important to choose a correct number of coefficients P because too many coefficients will yield a good fit to signal spectrum but will miss spectral envelope. On the other hand, few coefficients will miss formants. A reasonable number is 10 to 20.

Normally in a spectrum the presence of formant will be at interval of approx. 1000Hz i.e.

the degree is usually the same as the sampling frequency in KHz, therefore a good idea is to choose *'((sampling frequency/1000)+2)'* coefficients [16].

The Figure 4.2 below shows a Time domain representation and Frequency domain spectrum using Linear Prediction Analysis. The peaks shown in the frequency domain indicates the main frequencies present in the speech, the values are shown in Table 4.1.

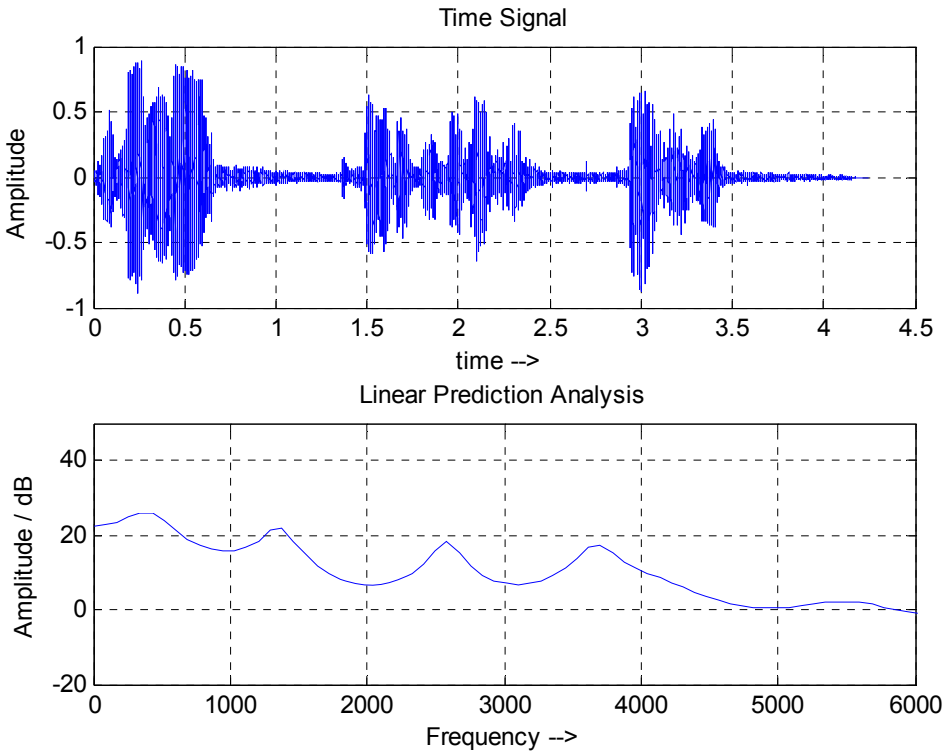

Figure 4.2 Determination of dominant frequency using Linear Prediction Analysis

Now the simulations are repeated again with wave file which also contains noise at 1 KHz. The simulation results are shown in next page (Figure 4.3). This time Linear Prediction Analysis failed to indicate the noise present in the spectrum as well as other dominant frequencies and the difference between the dominant frequencies keeps on increasing with the increase in number of frequency components. The formant frequencies which are calculated in both cases are shown in Table 4.1.

Frequency / Hz →	Frequency 1	Frequency 2	Frequency 3	Frequency 4
Speech	413	1363	2575	3667
Speech + noise	500	1310	1920	2800

Table 4.1 Comparison of calculated dominant frequency using LPC method for speech

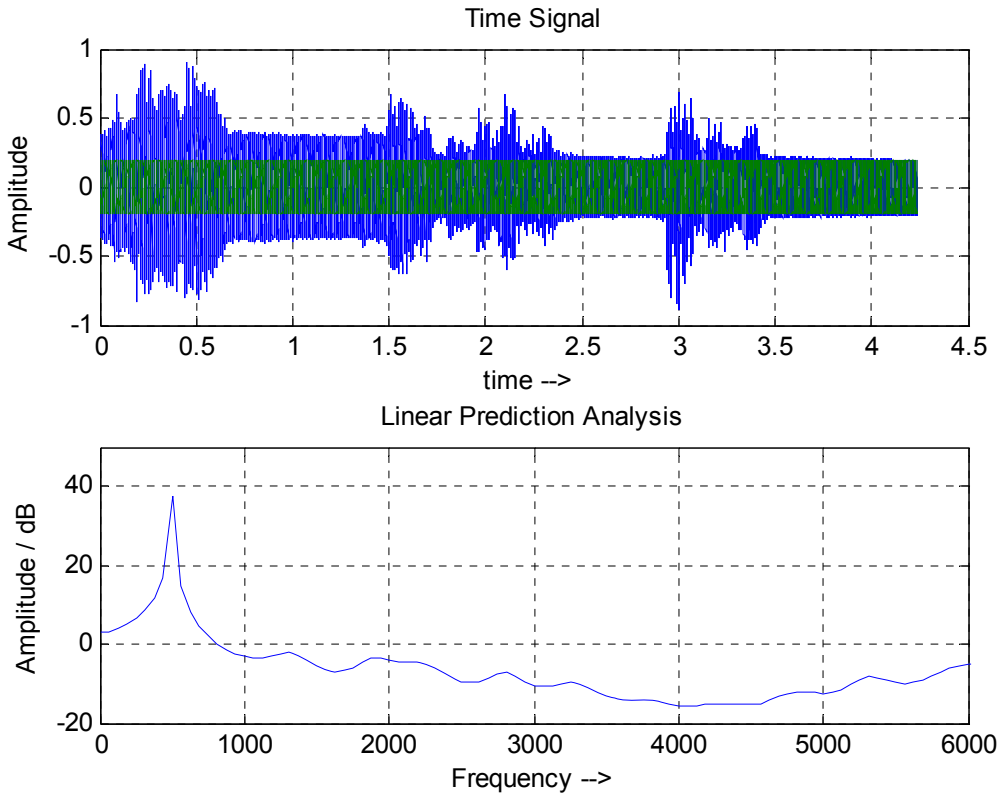

Figure 4.3 Determination of dominant frequency using LPC for speech+noise signal

Even though the LPC method is very efficient in finding the dominant frequencies in a speech spectrum but fails when any dominant frequency contains noise which in turns affects other frequencies present in the speech spectrum. But in real time high noise level can affect any dominant frequency; therefore this method is not suitable for source localizationas in current form and a modified LPC method can be tried. In addition to this Levinson-Durbin algorithm [38] which is required to use for real time is also very numerically intensive, that makes it avoidable for low speed DSP.

4.1.2 Spectrogram

Spectrogram is a powerful general-purpose method for spectral analysis. The basic idea is to assume non-stationary signals as a set of adjacent quasi-stationary signals. This is being achieved by considering a time-varying signal obtained by sliding a rectangular window across the time signal and then perform a Short Time Fourier Transform (STFT). The STFT has a fixed resolution in time and frequency, in the same manner as the Fourier Transform.

The frequency resolution Δf = 1/NΔt is controlled by the length of the window. The modulus square of STFT $|S(\omega)|^2$ is called spectrogram.

The spectrogram gives us the power distributed in the frequency domain. By finding the peaks corresponding to the frequency component having high spectral density, one can approximate the dominant frequencies present in the spectrum.

In Matlab, at first FFT is applied to the time domain signal to obtain the complex frequency domain spectrum and then absolute value of the spectrum is calculated to get Spectrogram, which shows the high spectral densities around dominant frequencies present in the speech.

Simulations are run for the same speech signal as used in LPC with and without noise signal to see the effect in determining the dominant frequencies. The Figure 4.4 in next page shows the power spectrum and it can be seen how the power is distributed in the whole frequency range. If we compare the dominant frequencies estimated by spectrogram method and LPC method, they are same (Table 4.3). Now the simulations were repeated for the same speech signal but this time with noise (Figure 4.5). This time one can see that the noise has the highest power spectral density, but it has no effect on other dominant frequencies (Table 4.2) and the noisy signal can be filtered out.

Frequency / Hz →	Frequency 1	Frequency 2	Frequency 3	Frequency 4
Speech	413	1363	2575	3667
Speech + noise	413	1000	1363	2575

Table 4.2 Comparison of calculated dominant frequency using Spectrogram method

In Table 4.3 it can be easily seen that the approximation of dominant frequencies with Spectrogram method is accurate for speech+noise signal then the LPC method.

Frequency / Hz →	Frequency 1	Frequency 2	Frequency 3	Frequency 4
LPC without noise	413	1363	2575	3667
Spectrogram without noise	413	1363	2575	3667
LPC with noise	500	1310	1920	2800
Spectrogram with noise	413	1000	1363	2575

Table 4.3 Comparison between Spectrogram and LPC method

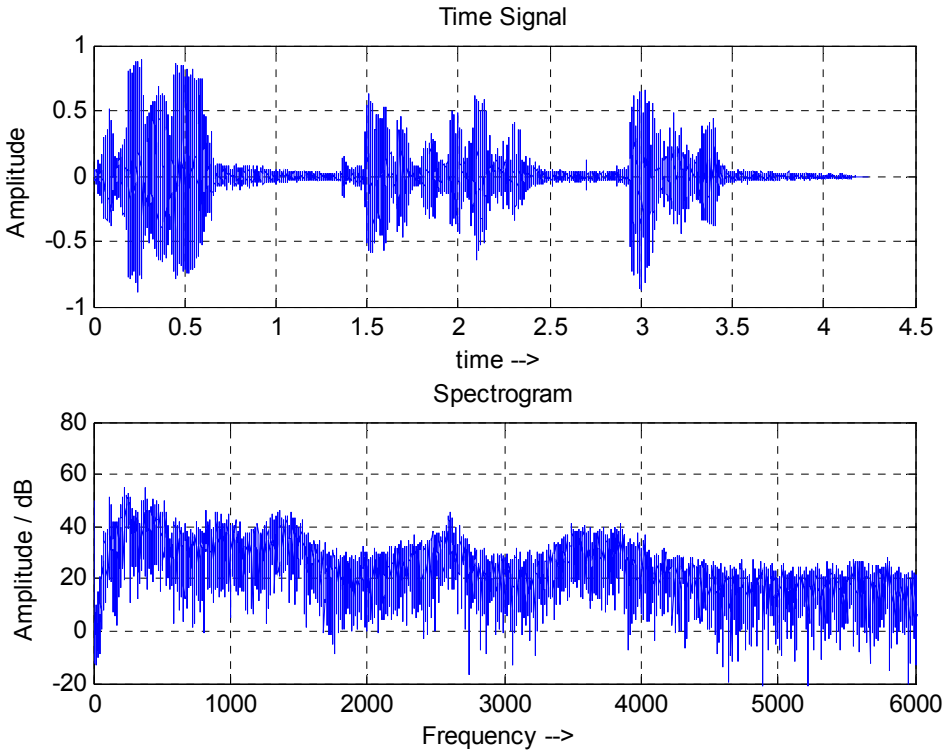

Figure 4.4 Spectrogram of speech signal

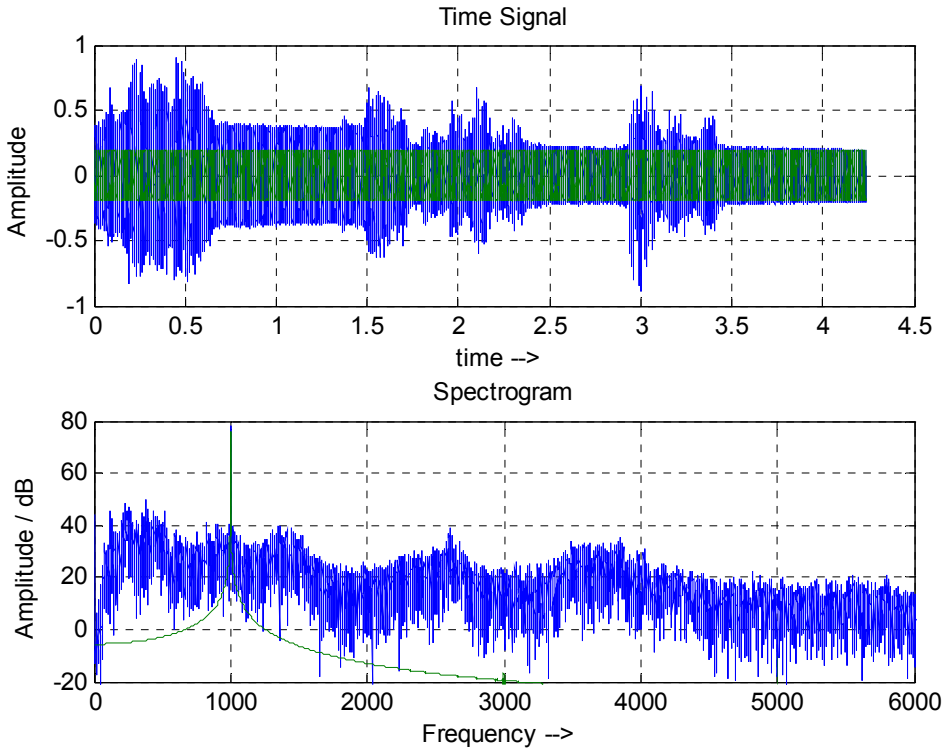

Figure 4.5 Spectrogram of speech signal with noise signal at 1 KHz

For the localization of speaker in wideband more than one frequencies present in the spectrum are considered, therefore if in one bin the SNR is very low, still with the approximation of other frequencies it is possible to localize the position of speaker and in most cases the noise will not be concentrated in one frequency but will be spread in the whole range.

4.2 MUSIC Algorithm

After finding the main frequency component present in the spectrum, next step is to define the matrix around these main frequencies and calculate the MUSIC spectrum. This step is being accomplished in two steps: first step is to separate the signal subspace from noise subspace for each frequency, which is being achieved by using SVD (Singular Value Decomposition) and the second step is to calculate the MUSIC spectrum for each frequency and averaged them to find the location of Source.

4.2.1 Singular Value Decomposition

As discussed in Chapter 2.2, it is clear that covariance matrix is defined from the output matrix of the microphone signals and then EVD is performed to get the eigenvalues and eigenvectors. But EVD has some limitations like it is applicable to only on square and non-defective matrices. Another approach is to use SVD; it is introduced by Golub and Kahan [17] in 1965 as a decomposition technique for calculating the singular values, pseudo inverse and rank of a matrix. It is applicable to both complex and real matrices and is motivated by the following geometric fact [18]:

The image of the unit sphere under any M X N matrix is an ellipse.

It can be directly implemented on the output matrix defined from microphones without any need to derive covariance matrix. The relation between EVD and SVD is as follows: SVD method decomposes a matrix say, A into three new matrices as shown below

$$A = U \cdot W \cdot V^T \tag{4.5}$$

where

U termed as left eigenvectors is a matrix whose columns are the eigenvectors of AA^T

W (Σ) called diagonal matrix, whose diagonal elements are the singular values of A

V termed as right eigenvectors is a matrix whose columns are the eigenvectors of A^TA

Therefore we can say that U and V are *M X M* and *N X N* orthonormal matrices and W is an *M X N* diagonal matrix with the nonnegative singular values w_j , j = 1,…,min(M,N), arranged in decreasing order along the diagonal.

For U and V, in both case the order of product of W and W^T is square diagonal matrix and keeping in mind the definition of EVD and recalling Eq. 2.27, we can say

$\mathbf{A^TA = VW^TWV^T}$ is the EVD of $\mathbf{A^TA}$ and $\mathbf{AA^T = UWW^TU^T}$ is the EVD of $\mathbf{AA^T}$

If we assume that *A* is square and symmetric, then each eigenvector for *A* with eigenvalue *w* is an eigenvector for $A^2 = AA^T = A^TA$ with eigenvalues w^2. Hence the left and right singular vectors for *A* are the absolute value of its eigenvalues. That is the EVD and SVD essentially coincide for symmetric *A* and are actually identical [19]. Therefore SVD can be used instead of EVD in MUSIC algorithm.

An important geometrical interpretation of SVD is shown in Figure 4.6, for M = N = 2. Let assume that the columns of U and V are denoted by the vectors u_j, j = 1,...M and v_j, j = 1,…N. If we analyze the three factors of SVD separately, we can see that V^T is nothing but pure rotation of the circle. Figure 4.6 shows how the axes v_1 and v_2 are rotated by V^T to coincide with the coordinate axes. In next step the circle is stretched by W (Σ) in the directions of the coordinate axes to form an ellipse. The third step rotates the ellipse by U into its final position. It can be easily noted that v_1 and v_2 are rotated to end up as u_1 and u_2, the principal axes of the final ellipse.

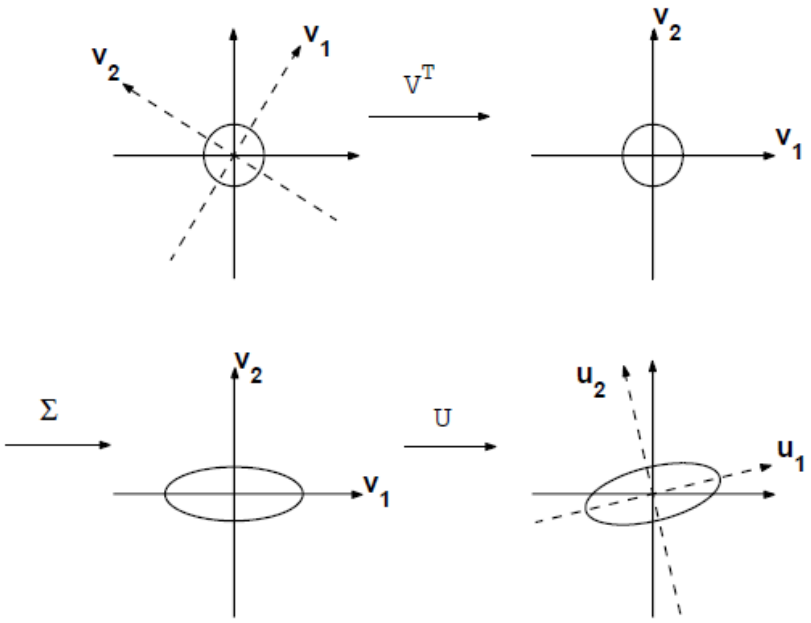

Figure 4.6 Geometrical Meaning of SVD

This SVD algorithm is being called Full SVD and is numerically very intensive. But this algorithm has good geometrical properties which can be exploited to reduce the computational time. Let assume matrix A (*M X N*) with *M* ≥ *N* and A has full rank N. Then we can see that A has 'n' singular values i.e. w_j, j = 1,…,N , arranged in decreasing order along the diagonal. Also the vectors u_j, j = 1,...,N will be numbered corresponding to the singular values. Thus the vector $w_j \cdot u_j$ will be the j^{th} largest corresponding principal semi axis. Similarly v_j, j = 1,…N are the preimages of corresponding principal semi axis. Hence it can be shown that

$$A \cdot v_j = w_j \cdot u_j, \quad 1 \le j \le N \tag{4.6}$$

The above equation can also be represented as $A \cdot V = \hat{U} \cdot \hat{W}$, where W is a N X N diagonal matrix with positive real values, U is an M X N matrix with orthonormal columns and V is an N X N matrix also with orthonormal columns. As we can see that U is unitary and therefore its inverse can be multiplied to the right side of above equation. We get,

$$A = \hat{U} \cdot \hat{W} \cdot V^T \tag{4.7}$$

This factorization of A is called as a Reduced Singular Value Decomposition or Thin SVD. This Thin SVD is being implemented in Numerical Recipes in C [20], which will be used for the 'C' implementation. The collection of vector equations shown in Eq. 4.11 can be expressed as a matrix equation,

$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U} \\ \\ \end{pmatrix} \cdot \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \cdots & \\ & & \cdots & \\ & & & w_N \end{pmatrix} \cdot \begin{pmatrix} \\ \mathbf{V}^T \\ \\ \end{pmatrix}$$

Figure 4.7 Reduced Form of SVD Algorithm

After finding out the eigenvalues, the signal subspace is separated from noise subspace corresponding to the eigenvalues.

4.2.2 MUSIC Spectrum

The next step is to calculate the MUSIC spectrum using the noise subspace obtained from the SVD method and already calculated steering vectors corresponding to the dominant frequencies. Pseudospectrum will be calculated for each frequency component separately and then will be averaged to get wideband MUSIC spectrum to localize the position of speaker. The Eq. 2.40 used to calculate the incoherent wideband spectrum has been derived in Chapter 2.3 and is shown below again,

$$\hat{P}_{Incoherent}\,(\omega_k, \theta) = \sum_{k=1}^{M} \frac{1}{steer(\omega_k,\theta) \cdot Noisespace \cdot steer'(\omega_k,\theta)} \tag{4.8}$$

where Noisespace is the Noise subspace obtained corresponding to eigenvalues for each frequency present in the spectrum

After calculating the incoherent wideband MUSIC spectrum, a tracker algorithm is employed to find the highest peak value in the spectrum and then find the angle (DOA) corresponding to the peak value and also find the error in estimation.

The flow chart in Chapter 2 (Figure 2.9) has summarized the complete algorithmic implementation.

Till now only the main parts of the algorithm and the related theory have been discussed. Before proceeding to the simulations of the algorithm, the complete algorithm is summarized as follows:

The sine waves are generated at the specified frequencies for each microphone and 256 samples of these sine waves are phase-delayed respectively with number of microphones, which are six & eight in our case fulfilling the far-field assumption. The next step is to apply FFT on each microphone's time signal to obtain complex frequency domain signal. Then bin-based threshold method is used to find the dominant frequency component present in the spectrum, this is being accomplished by creating bins or subband and searching for the maximum absolute value in each bin and zeroing of others. After finding out the dominant frequencies the output matrix is created around these frequencies. Thereafter eigenvalues and eigenvectors are calculated for respective output matrix. After that the signal subspace is separated from noise subspace corresponding to the eigenvalues. The next step is to calculate the MUSIC spectrum for each dominant frequency component present in the spectrum and average them to have a wideband MUSIC spectrum. Of course for the calculation of MUSIC

spectrum pre-knowledge of frequencies is required. In the end a tracker algorithm is employed to localize the source.

4.3 Simulations and Influence of Parameters

Before implementing an algorithm for real time system, it is always desirable to verify the algorithm using simulations. The performance of the algorithm for estimation of Direction of Arrival is tested in simulated environment using Matlab. This is being achieved by varying the different parameters which can have effect on the Microphone Array, algorithm and the complete system. Simulations are performed keeping in mind the real system setup as in the end it would be interesting to compare the simulation's result with real system's result. For this purpose a GUI (Graphical User Interface) version of the algorithm is developed which helps a lot in analyzing the behavior of algorithm with the change of parameters. The GUI of the algorithm is shown in Figure 4.8.

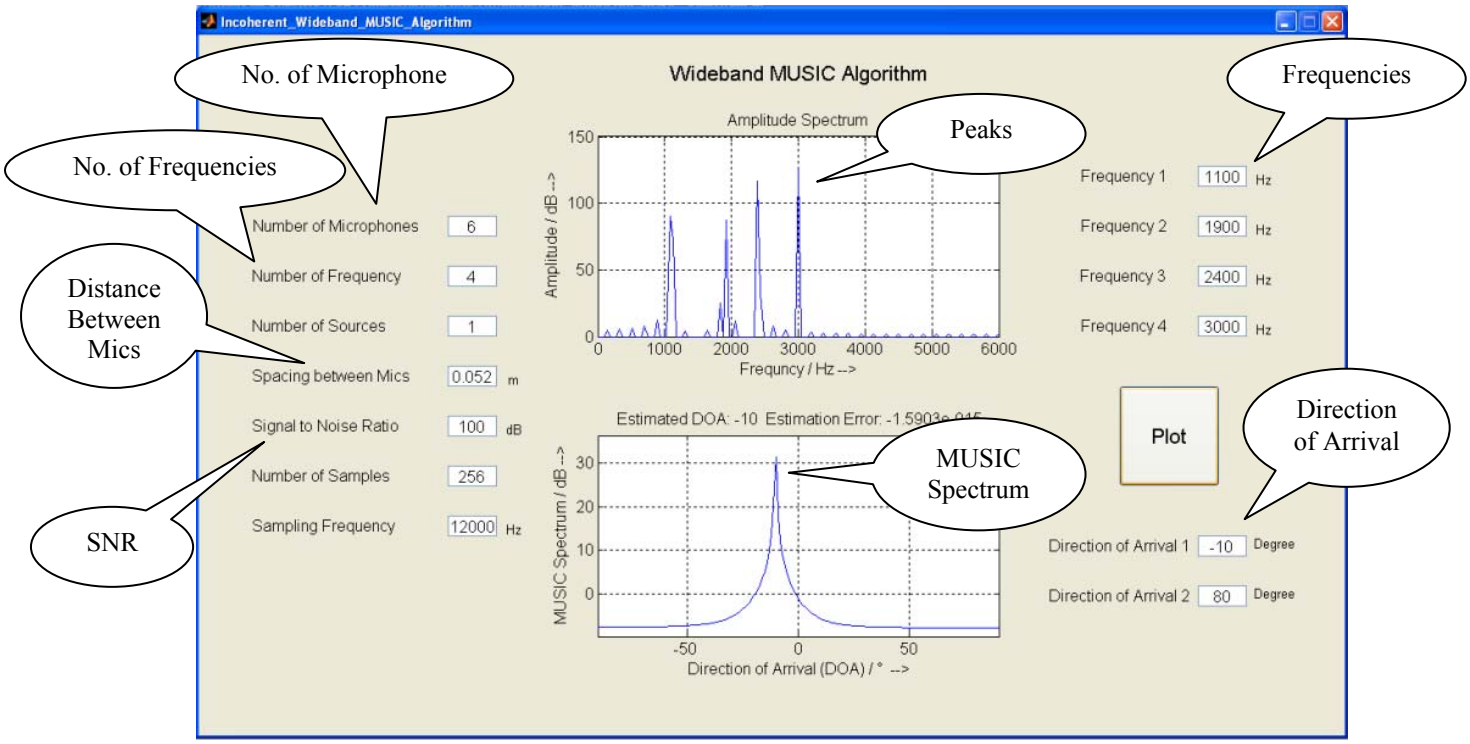

Figure 4.8 GUI implementation of Incoherent Wideband MUSIC Algorithm

As shown in Figure 4.8 the main parameters which can influence the algorithm are on the left side of figure. The number of microphones are fixed at 6 for simulations, the number of frequencies which can be analyzed are up to 4 because as discussed earlier

mostly the dominant frequencies are at a distance on average of 500 to 800 Hz in a speech spectrum, theoretically MUSIC algorithm can localize N-1 (N is number of microphones) sources but keeping in mind the objective of the work only two sources are considered, the length of uniform linear array plays very important role in fulfilling the far-field assumption but also determines the resolution of the algorithm, spacing between microphones is kept at 5.2 cm because of the limitations imposed by physical Microphone array which is used in the real system experiments, Signal to Noise Ratio (SNR) plays very important role as in real time SNR can varies from 5 to 40 dB and it would be very interesting to see the behavior of algorithm at low SNRs, the number of samples are kept constant and also the sampling frequency. The simulations are performed for the frequency varies from 1 to 4; SNR varies from -10 dB to 10 dB, keeping spacing between microphones at 5.2 cm. The frequencies which are being shown on the right side of graphs in Figure 4.8 are chosen after analyzing the speech spectrum of many wave files (corresponds to different speakers) and are the approximation of those speech files. The direction of arrival is varied from -90° to +90° for the simulations.

At first simulations are performed with very high SNR to analyze the behavior of algorithm. The Figure 4.9 shows the deviation of estimated direction of arrival from actual direction of arrival.

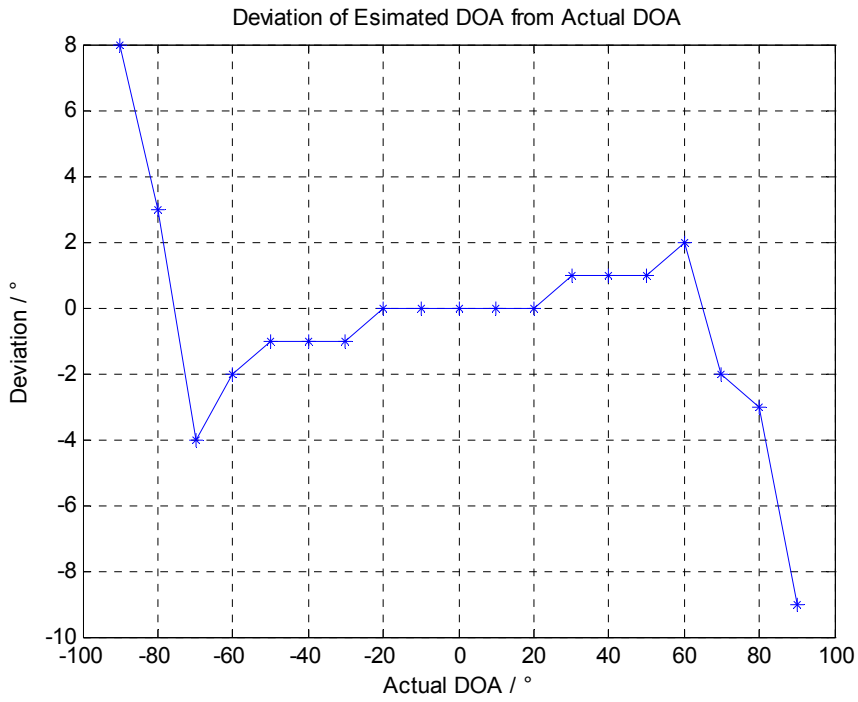

Figure 4.9 Estimation of DOAs for 1 source, 4 frequencies, with SNR = 100 dB

We can draw some inference from the Figure 4.9 regarding the performance of the algorithm. It can be easily seen that Estimated DOA at any angle θ is approximately mirrored by the same angle θ in opposite direction. This is because of the reason that received microphone signal undergo the same delays irrespective of whether they are coming from positive direction or the negative direction. We can also see that when the source is beyond ±70° the variation in Estimated DOA from actual DOA generally increases with the increase in θ. Another important feature which was noticed as shown in Figure 4.10 is that the Peak of the spectrum at angle 0° is infinity. This feature can be attributed to the fact that the all microphone signals are in phase, which in turn gives one very large eigenvalue and others being equal to zeros. However in real time environment one cannot expect the same as there will always be noise around 30dB below the source or less as well as because of the quantization constraints and near field effects.

In real time for localizing a source the area of interest is normally much smaller than what is simulated here. Generally speaking, in a seminar room the area of interest ranges between ±40° and the algorithm works very well in this range, but without the presence of noise. In next sections the behavior of algorithm is analyzed with noise and also by varying other parameters, which can influence the algorithm in real time.

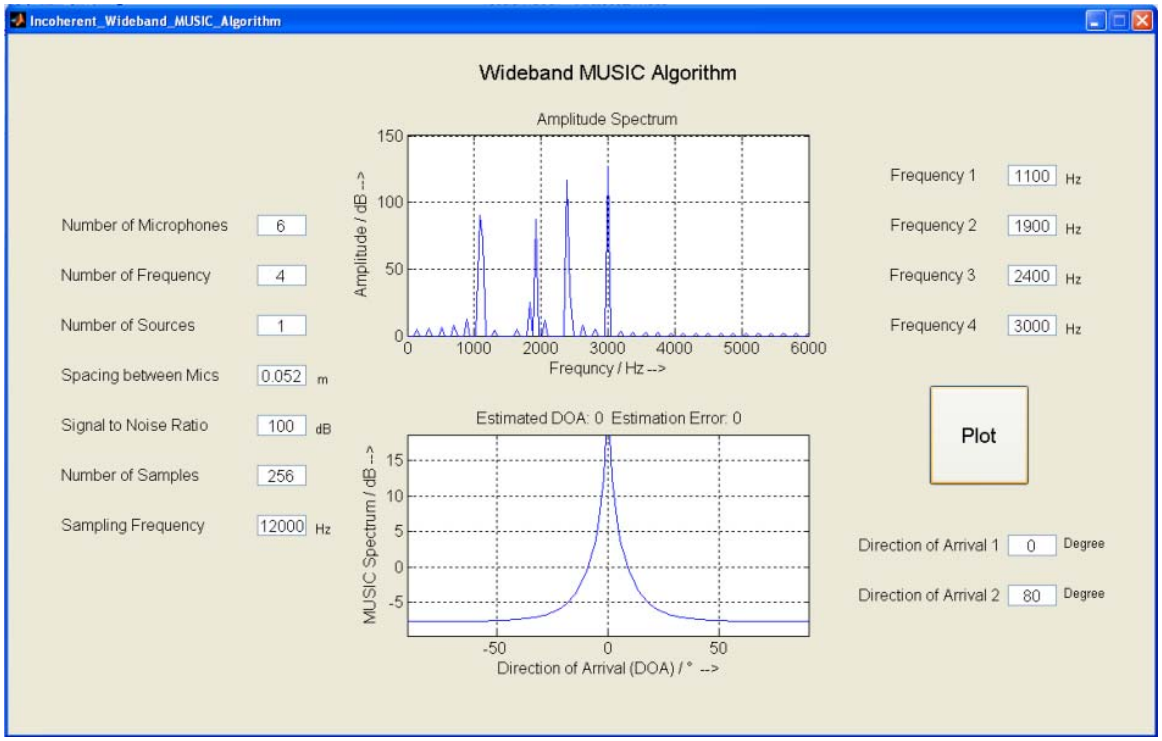

Figure 4.10 Estimated DOA for 4 frequencies at angle 0° with SNR = 100dB

4.3.1 Simulations with Narrowband MUSIC

In this section the simulations of algorithm are performed in presence of white noise as well as by varying SNRs. Even though all parameters have influence on the performance of algorithm, like numbers of samples are taken as 256 but if 128 samples are taken then there was little bad effect on the stability of algorithm and change in sampling frequency also have an effect.

In this section only the performance of Narrowband MUSIC algorithm is discussed, as simulations are performed when only one source is present in the spectrum. The stability of algorithm is checked by varying the SNR from 10 to -10 dB because Signal to Noise Ratio keep on changing in real time and it would be interesting to see the performance of algorithm with varying SNRs. All results are averaged value of 10 approximations.

Simulations are done with SNRs 10, 5, -5 and -10 dB for direction of arrival ranging from -80° to +80°. Figure 4.11 shows the deviation in estimations for different SNRs.

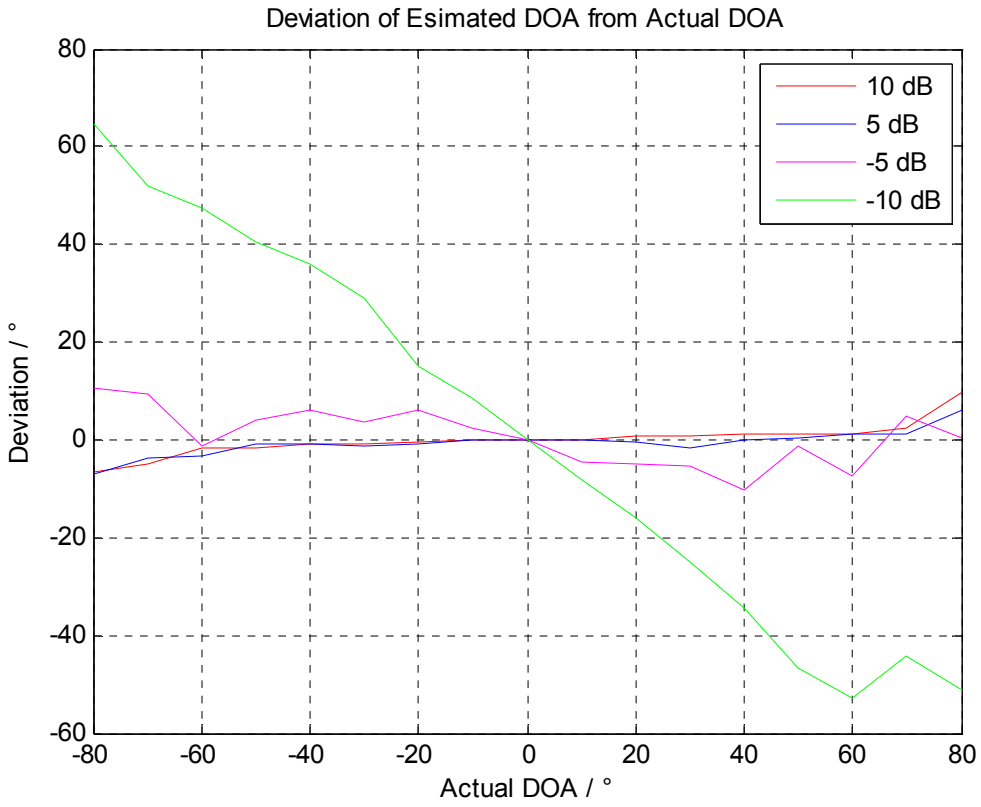

Figure 4.11 Estimated DOAs for narrowband source with varying SNR

One can see that at SNR equal to 10 dB, the algorithm performs quite well even for one frequency and also it was observed that above 10 dB the estimated direction of arrivals

are quite stable. Even though we can notice that at 5 dB the algorithm performs very well or it is difficult to differentiate the estimated direction of arrival from that of 10 dB, but the values which are shown one are actually averaged value. The range of estimated direction of arrival is much more than what is being observed for 10 dB. Below 0 dB the algorithm has failed to measure the DOAs accurately and variation in estimation is quite large. Also we can see that at SNR -10 dB the algorithm breaks down. It is because of the reason that eigenvalues obtained from SVD of matrix cannot span the noise subspace accurately and hence the wrong estimation of DOAs.

4.3.2 Simulations with Wideband Sources

In this section algorithm is simulated with two, three and four frequencies. The parameters are same as what is used for simulations of narrowband MUSIC. The simulations are performed for two frequency sources i.e. 1.1 KHz and 1.9 KHz, three frequency sources i.e. 1.1 KHz, 1.9 KHz and 2.4 KHz and four frequency sources i.e. 1.1 KHz, 1.9 KHz, 2.4 KHz and 3 KHz and the results are shown in Figure 4.12, Figure 4.13 and Figure 4.14 respectively.

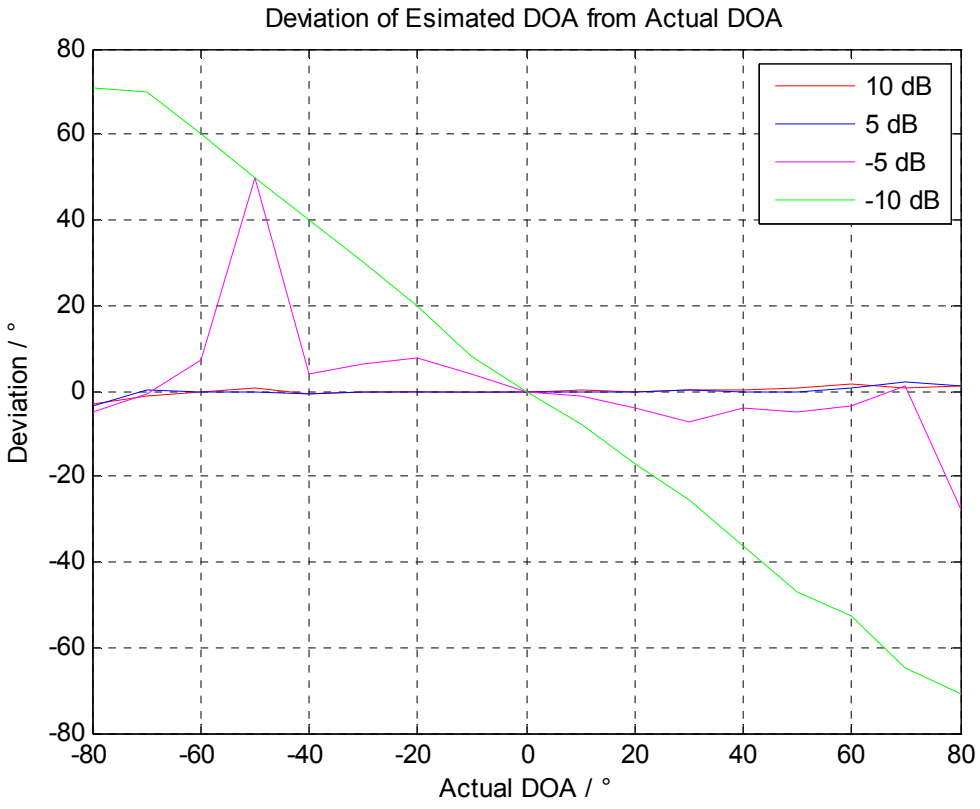

Figure 4.12 Estimated DOAs for two sources (1100 Hz and 1900 Hz) with varying SNR

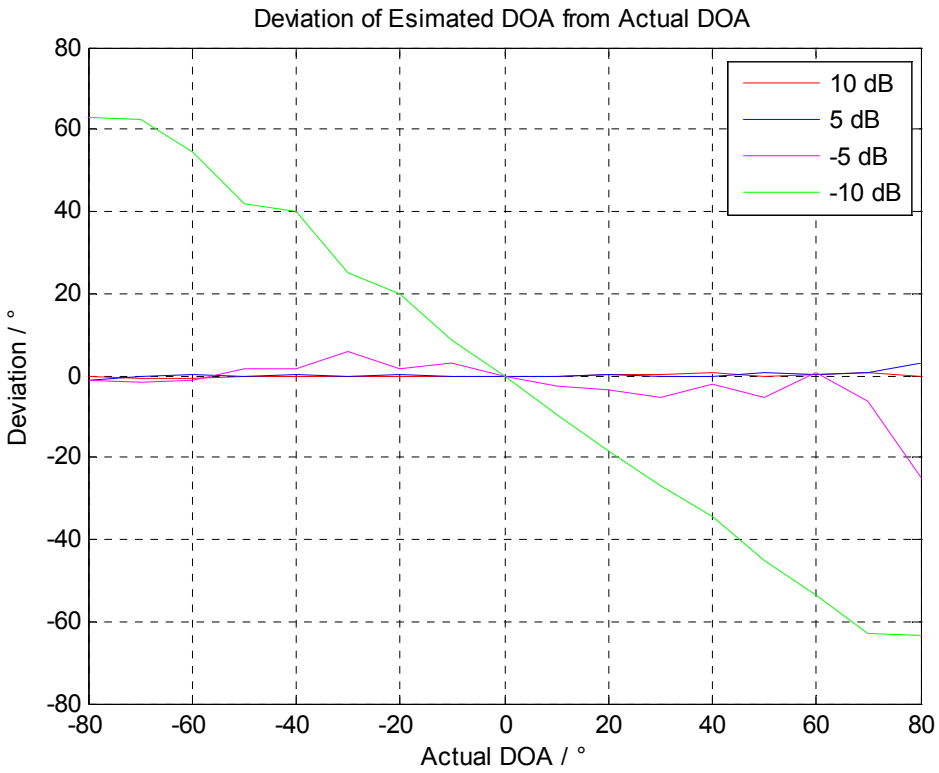

Figure 4.13 Estimated DOAs for three sources with varying SNR

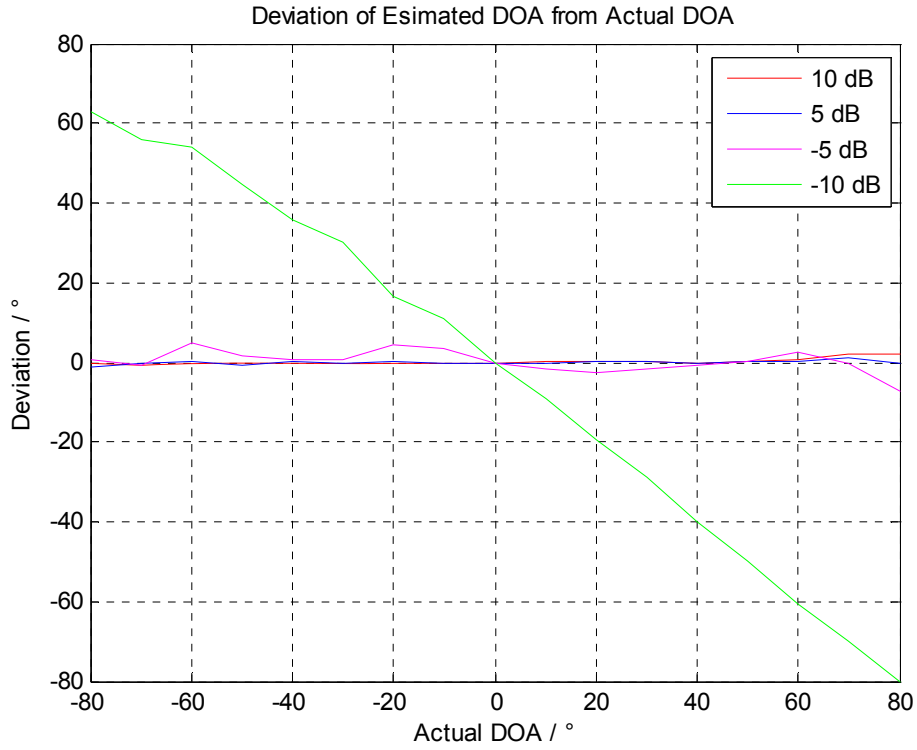

Figure 4.14 Estimated DOAs for four sources with varying SNR

It can be easily interpreted from the figures that with the increase in number of frequency component the performance of algorithm shows remarkable improvement for SNR equal to -5 dB, but for SNR equal to -10 dB the algorithm still breaks down. Other than that it was also observed that algorithm is much more stable with 3 and 4 frequencies. Also in all cases the variability in estimated direction of arrivals increases with decrease in SNRs, but not so rapidly. For example the deviation in estimated direction of arrival for four frequencies is within the range of ±2° especially for ±50° for SNR equal to 10 dB. One can also observed that for the range of ±50° the difference between estimated direction of arrival for 3 and 4 frequencies is not noticeable for SNR equal to 5 and 10 dB, but below that the differences can be easily noticed. It was also noticed that for SNR less than 5 dB the spectrum is much wider. Although it is very unlikely that one will encounter the SNR below -5 dB in real time situation still we have seen that the algorithm works very well in simulated high noise environment. But in real time there are many external factors like near field effects, quantization effect on input signals which cannot be accounted in simulated environment. The figures below show an example of estimated direction of arrival for -40° for all the four setups for SNR equal to -5 dB.

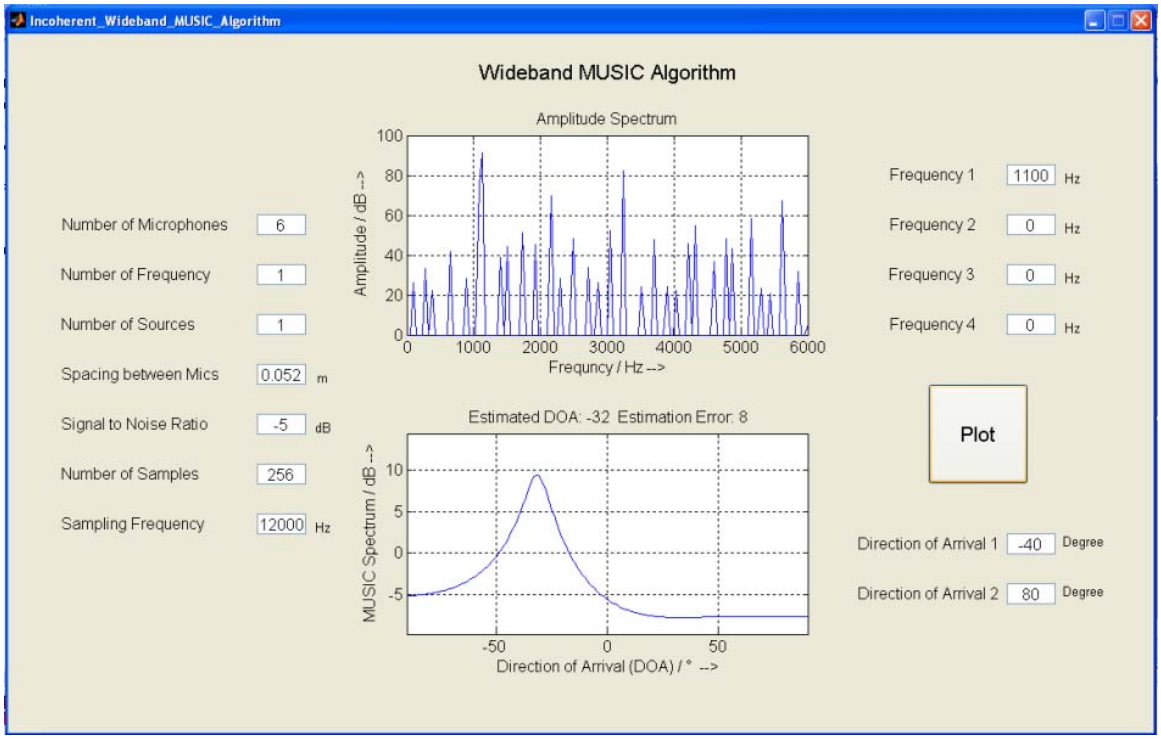

Figure 4.15 EDOA for 1100 Hz at -40° for SNR = -5 dB

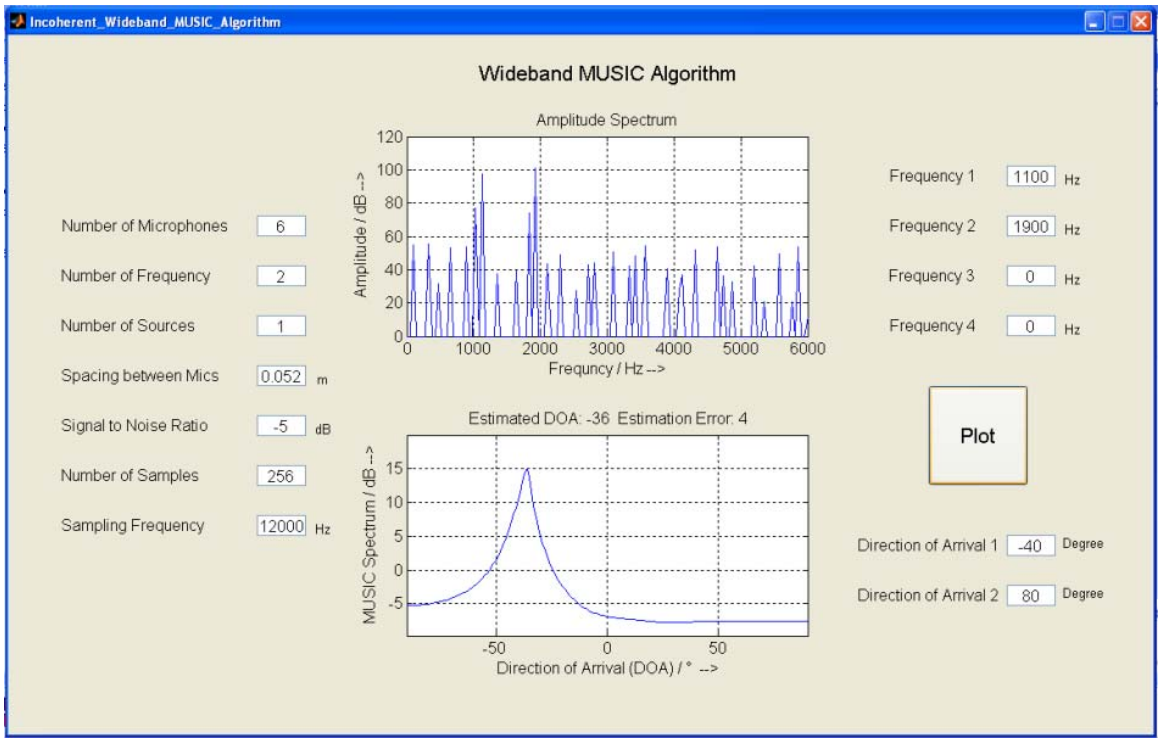

Figure 4.16 EDOA for 1100, 1900 Hz at -40° for SNR = -5 dB

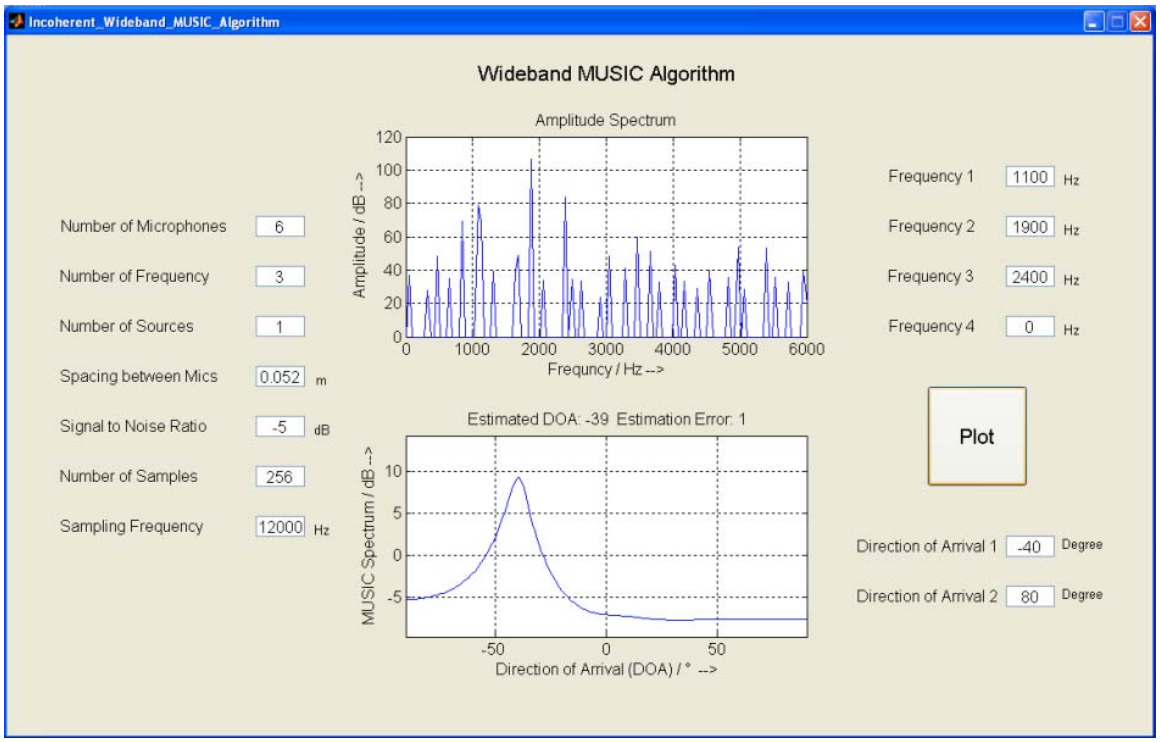

Figure 4.17 EDOA for 1100, 1900 & 2400 Hz at -40° for SNR = -5 dB

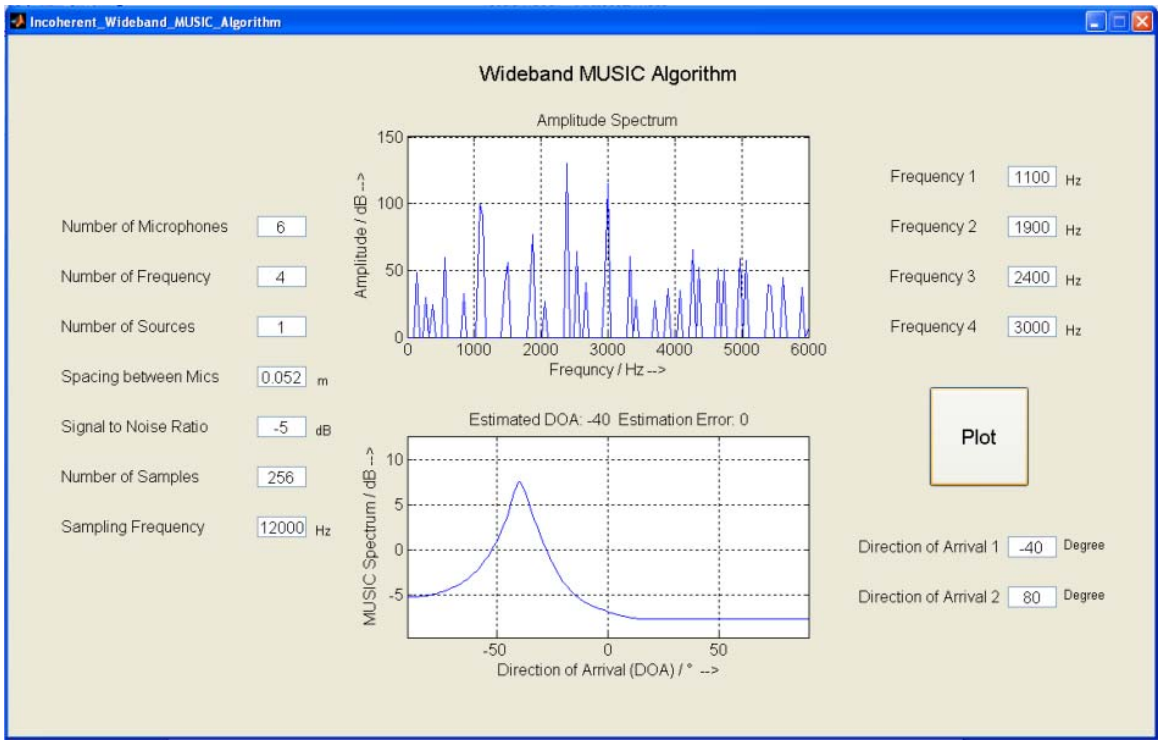

Figure 4.18 EDOA for 1100, 1900, 2400, 3000 Hz at -40° for SNR = -5 dB

4.3.3 Resolution of Algorithm

Another important factor in determining the capability of algorithm is to see the limit of resolution in terms of estimating the two sources separately and also the required distance between two frequencies i.e. frequency selectiveness. Theoretically speaking the MUSIC algorithm should be able to resolve two arbitrarily close sources and it would be quite interesting to see the how accurately the algorithm works for close angular spacing as well as frequency.

It was observed in the simulation for angular closings, that for SNRs above -5 dB the resolution is quite good and the algorithm is able to separate the two sources.

To see the frequency resolution of the algorithm the simulations are done by keeping one source at fixed frequency and varying another. It was done for every main frequency i.e. 1 KHz, 1.9 KHz, 2.4 KHz and 3.0 KHz and for angular direction of arrival of -40°. When two frequencies are used the frequency resolution was approximately 300 Hz for SNRs above 10 dB. For three frequencies the resolution was 250 Hz. And for four frequencies the resolution was 200 Hz. All the estimated direction of arrivals was in the range of ±1° for SNRs 10 dB and above.

Figure 4.19 in next page shows the frequency resolution with four frequencies.

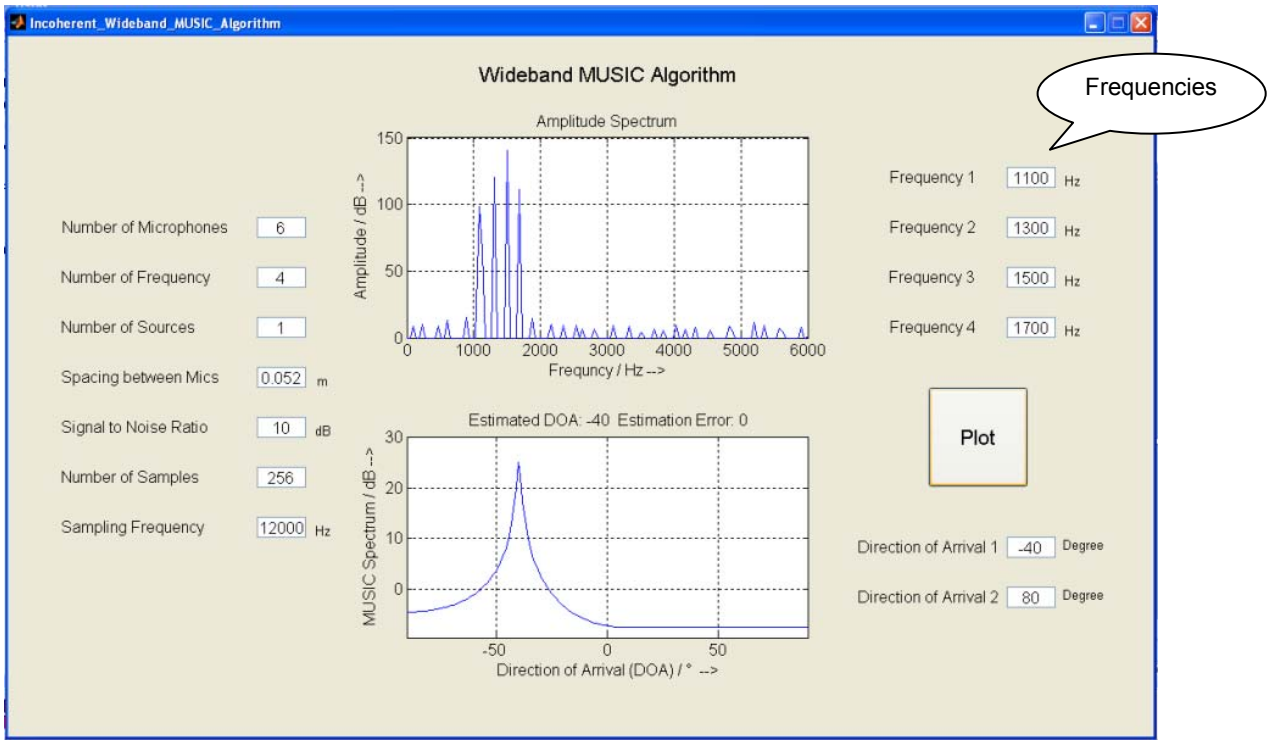

Figure 4.19 Frequency Resolution for 1.1 KHz, 1.3 KHz, 1.5 KHz and 1.7 KHz at -40°

5. Implementation of Algorithm

In chapter 4.3 the performance of algorithm has been seen in simulated environment and the results are excellent ones and now the algorithm is implemented on DSK C6713 (TI's TMS320C6000 DSP) board. The simulated environment provides an ideal environment and in real time there are always surprises. In this chapter implementation of algorithm in C using Code Composer Studio 3.1 platinum edition is discussed. In the first section adaptive array algorithm (calibration of array) is introduced and explained which is needed as the linear microphone array is bounded by physical constraints like; no two microphones can be similar in characteristics because of the material used. After implementing the adaptive algorithm, the next step is to find the dominant frequencies present in the spectrum using spectrogram method, which is being discussed in subsequent section. The third main step (third section of the chapter) is to define the output matrix around the dominant frequencies and apply incoherent wideband MUSIC algorithm and in the last section of the chapter the Tracker algorithm employed is discussed. The Figure 5.1 below shows all implementation steps in the form of block diagram.

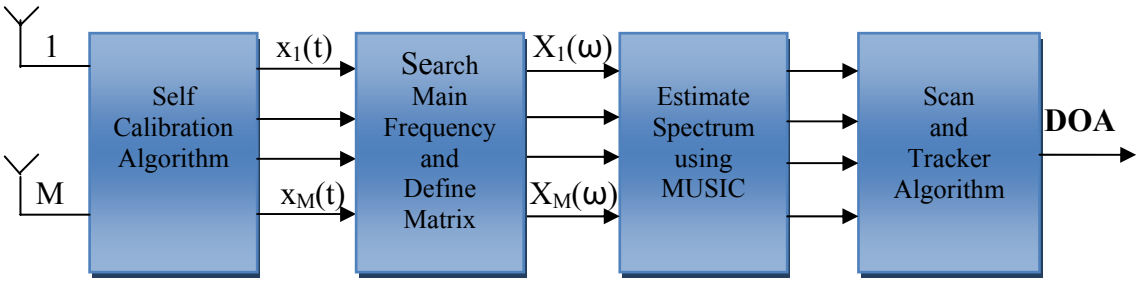

Figure 5.1 Main blocks of Incoherent Wideband MUSIC Algorithm

5.1 Self-Calibrating Microphone Array

In many cases microphone array used for speech applications, does not produce the desired results. One of the reasons is the difference in the signal paths of every microphone in the array which is mainly caused because of the different sensitivity of the microphones or due to mismatch between microphone preamplifiers. Because of the differing tolerances between different microphones in an array introduce gain and phase error, which will obviously undermine the performance of any real time system for speech application. Therefore calibration of array is required to minimize this mismatch.

One of the most commonly method to avoid this mismatch is to preselect the microphones for an optimal match or to predetermine calibration filters in a special measurement and apply them afterwards. Another approach is to calibrate each microphone separately by comparing it with a reference microphone in specialized environment. But these approaches are very expensive as they require manual calibration for each microphone and specialized equipment. It is appropriate for calibration of microphones prepared for precise acoustic measurements, but not for general purpose microphone array as in our case. Also the microphone's characteristics usually changes over time due to aging effect or environmental influences. Hence a self-calibrating adaptive algorithm is required for tracking these changes.

The self-calibrating algorithm used in the project is based on the method proposed by Van Compernolle [21] and later on modified by Buck, Haulick and Pfleiderer [22]. Van Compernolle proposed a self-calibrating algorithm where an adaptive filter unit (A.U.) is placed just after the ADC channel. This unit compensates for the mismatch between the microphones during the normal operation in the background. One microphone is chosen as reference and all other microphones of the array are adaptively matched to this reference. But using a single microphone's signal as a reference signal runs the risk of choosing a microphone with bad characteristics. Buck, Haulick and Pfleiderer suggested taking an average of all microphones' signal in an array and using it as a reference signal. The method suggested by them is simple and straightforward. Block diagram of the self-calibrating algorithm is shown in Figure 5.2. The calibrated output signals of every channel are filtered versions of the desired signal and contain the required spatial information which will be used for further algorithmic processing.

The method used is explained below:

The first input $x(k)$ is the desired signal which is being matched to the reference signal $d_{avg}(k)$, which is a second input by the adaptive filter $w_l(k)$.

The reference signal $d_{avg}(k)$ is given by

$$d_{avg}(k) = \sum_{m=1}^{M} \frac{1}{M} x_m(k),$$

where M is Number of Microphones

The output signals of the adaptive filter unit (A.U.) are the calibrated signal $x^c(k)$ and the error signal $e(k)$. This calibrated output signal is the enhanced and relatively flat version of the incoming signal received by microphone array in terms of amplitude.

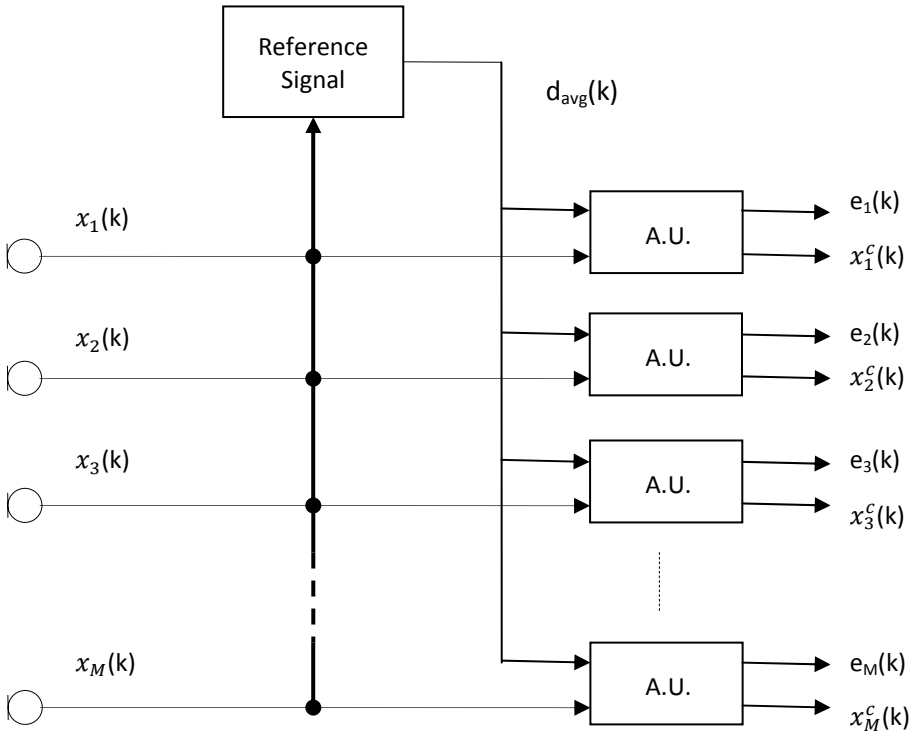

Figure 5.2 Block Diagram of Self-Calibrating Microphone Array

The coefficients of the adaptive filter are optimized using an LMS (Least Mean Square) algorithm [23] based on the error signal. The output of the adaptive filter is given by

$$x^c(k) = \sum_{l=0}^{N-1} w_l(k)x(k-l) \tag{5.1}$$

where $w_l(k)$ represents N coefficients for a specific time k.

The coefficients $w_l(k)$ are adjusted such that a mean squared error function is minimized. This mean squared error function is $E[e^2(k)]$, where E represents the expected value. Since there are 'l' coefficients, a gradient of the mean squared error function is required, but instead an estimate using the gradient of e^2(k) can be found

$$w_l(k+1) = w_l(k) + \beta e(k)x(k-l) \quad \text{whr, k = 0,1,....N-1} \tag{5.2}$$

The above equation gives a simple but powerful and efficient means of updating the coefficients, without the need for averaging or differentiating.

For each specific time k, each coefficient $w_l(k)$ is updated by new coefficient unless the error signal e(k) is zero. After the filter's output $x^c(k)$, the error signal e(k) and each of

the coefficients $w_l(k)$ are updated for a specific time k, a new sample is acquired from ADC and the adaptation process is repeated. The flow chart is shown in Figure 5.4.

The important parameters in LMS algorithm is adaptation rate β, which defines the rate of convergence and accuracy of calibration process and the number of coefficients 'l', which defines the speed and stability of the LMS algorithm. After testing the system in an anti-acoustic room, the optimum values reached are 6 and 1 for number of coefficients and adaptation rate respectively. The audio source was placed 300 cm from the array and played 1.1 KHz frequency, then the algorithm was run. The average gain (amplification) factor obtained for every channel after performing the self-calibrating algorithm varied from 4.23 to 5.55. The directions of arrivals estimated with self-calibrating algorithm are good and will be presented in Chapter 6.2.

The self-calibrating algorithm is being implemented in fixed point format and as a following function (`cali.c`) in the project:

```
void calibration(short int *in_buffer1,short int *in_buffer2,
                 short int *in_buffer3,short int *in_buffer4,
                 short int *in_buffer5,short int *in_buffer6)
```

where

```
short int in_bufferM;    // points to the buffer storing
                            microphone signal, M is No. of Mic
```

Figure below shows the un-calibrated and calibrated signals for six channels.

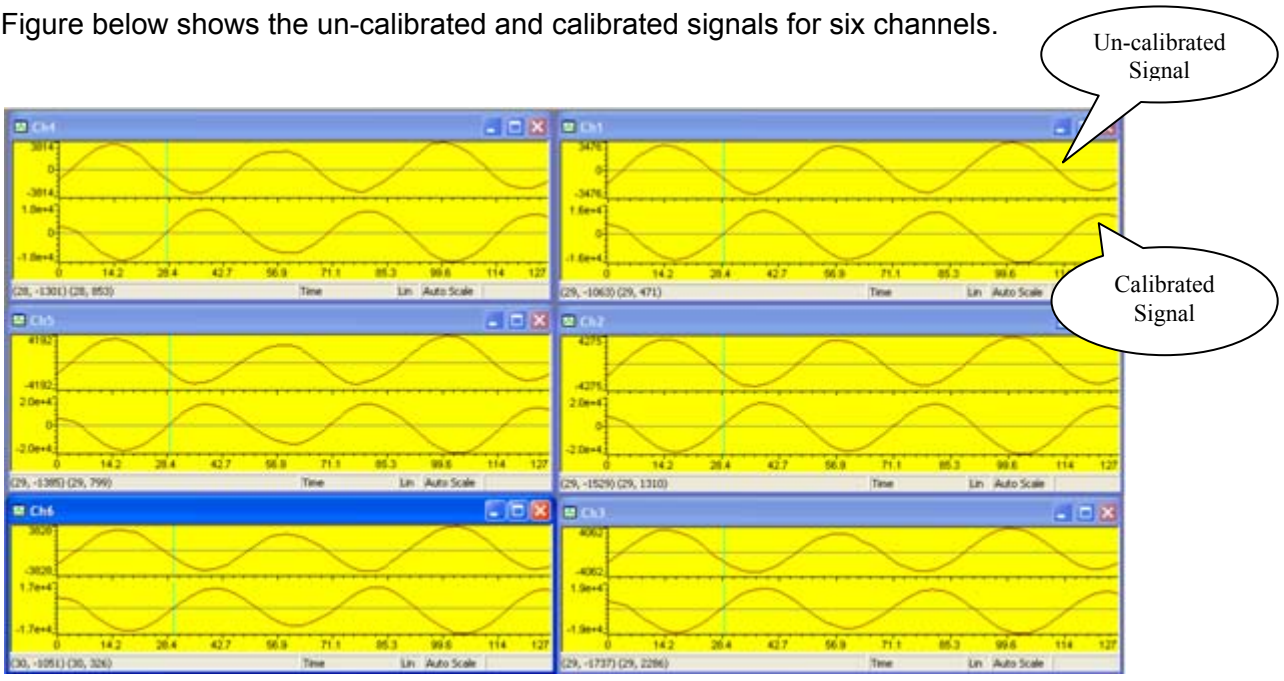

Figure 5.3 Self-Calibrated signals in real time for six channels

Figure below shows the flow chart of the self-calibrating algorithm.

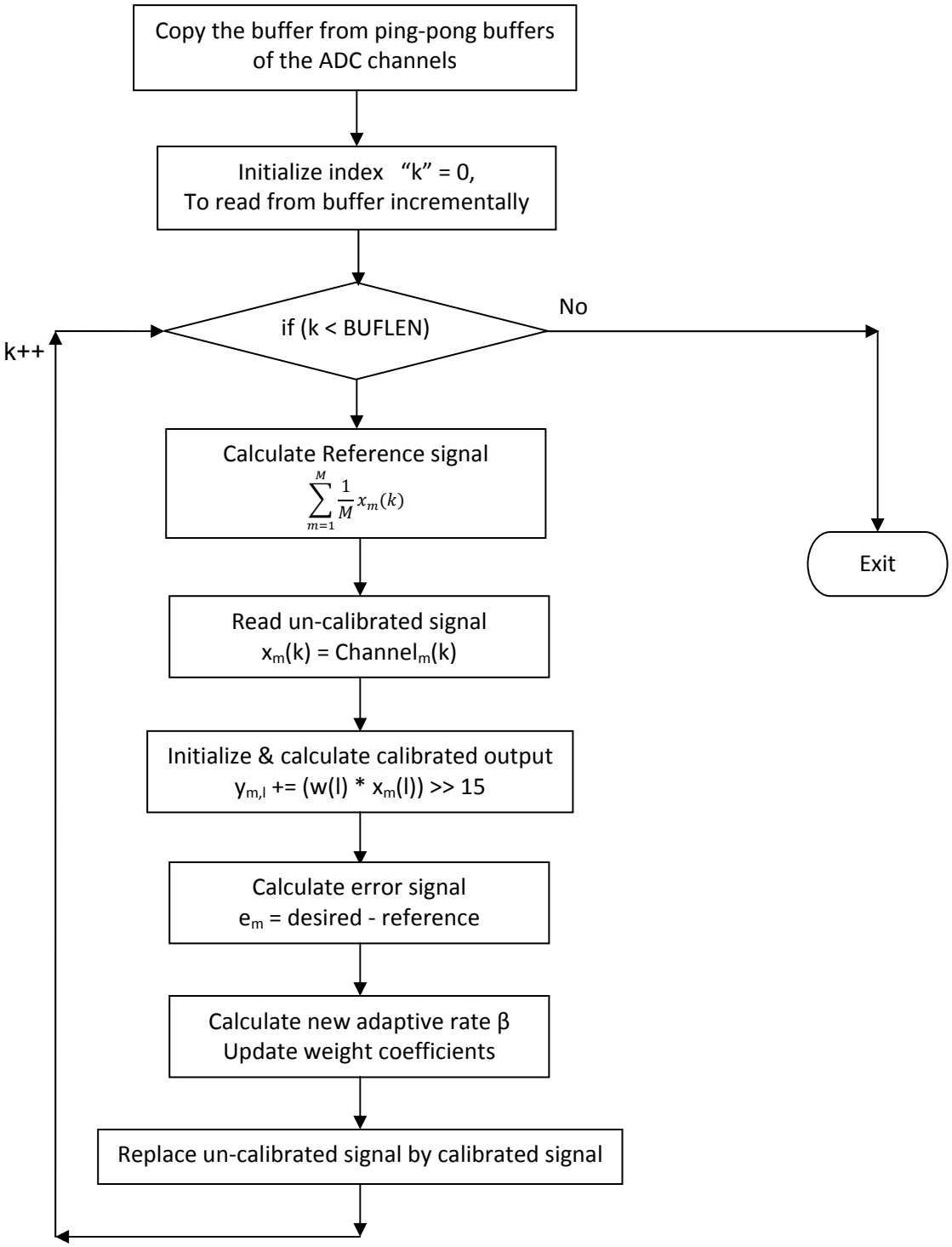

Figure 5.4 Flow chart of self-calibrating algorithm

5.2 Spectral analysis in Real time

As discussed in Chapter 4, spectrogram method was used to find the main frequencies present in the spectrum. In Matlab it is comparatively easier to calculate power spectrum using FFT function and finding the main frequencies using sort function as we can use complete CPU resources of system and inbuilt functions. But in real time system the speed and memory both plays a critical role and it is important to have an efficient implementation of algorithm on DSP system. In this section of the chapter, conversion of real calibrated signal to complex frequency domain signal and then adaptive selection of the main frequencies present in the power spectrum is discussed.

5.2.1 Complex Frequency Domain signal

The fixed point calibrated real time domain signal need to be converted into floating point frequency domain as the signal in Eq. 2.5 is in complex form. This is because the effect of a time delay on the time domain signals is just the phase difference in frequency domain and the incoherent wideband MUSIC algorithm is applicable in frequency domain. If we are interested in analytic signal, this can be accomplished by Hilbert transform, which introduces 90° phase shift between real and imaginary part of signal. Also the Hilbert transformed analytic series has the same amplitude and frequency content as the original real signal and includes phase information that depends on the phase of the real time domain signal. But we are interested in frequency domain signal.

At first fixed point need to be converted to floating point because SVD routine which is used for Eigenvalue decomposition requires floating point values as input. As the fixed point signal is being represented as `short int` (Q15 format)[24] and the mantissa part of floating point is larger than ADC resolution of the PCM Codec, fixed point is simply type casted into floating point. Still the safest way for typecasting is shown below:

$$\texttt{channelM} = ((\texttt{float})(\texttt{ibufferM}/2^{15}-1))$$

where `ibufferM` is the input fixed point sample.
The next step is to perform FFT (Fast Fourier Transform) to obtain frequency domain signal. When a normal Radix-2 512-point FFT algorithm was used, it was too much numerically intensive and was taking lots of cycles. As six channels and later on eight channels are used in the test setup corresponding to the number of microphones,

therefore using this normal FFT algorithm is not a good option, as for every channel containing 256 complex samples it was consuming approximately 320,000 cycles.
Another option is to use TI optimized Radix-2 Complex FFT algorithm from TMS320C6713 DSP Library [25]. This routine required N (=256 in project) complex floating-point numbers arranged as successive real and imaginary pairs, N/2 complex twiddle coefficients in bit-reversed order and length of FFT (N). This function is being used as follows:

```
gen_w_r2(W, BUFLEN);
bit_rev(W, BUFLEN>>1);
cfftr2_dit(channelM,W,BUFLEN);
bit_rev(channelM, BUFLEN);
```

The twiddle coefficients “W” were generated once in the starting of main program and passed on to the function `calculatefft() in fftcalculation.c`. This function required only between 11,500 to 13,000 cycles. This improvement in terms of cycles can be attributed to the assembly optimized FFT by TI specifically for TMS320C6713 DSP, which takes full advantage of the eight parallel functional units of the VLIW architecture.
The function to obtain complex frequency domain signal is being implemented as the following function (`fftcalculation.c`) in the project

```
void calculatefft(short int *ibuffer1,short int *ibuffer2,
short int *ibuffer3,short int *ibuffer4,short int *ibuffer5,
short int *ibuffer6,float *W,COMPLEX *channel1, COMPLEX *channel2,
COMPLEX *channel3,COMPLEX *channel4,COMPLEX *channel5,COMPLEX *channel6)
```

The parameters used are explained as below:

```
short int *ibufferM   // M channels having real calibrated signal
float W               // Twiddle factors
short int BUFLEN      // Data block size 256, in project
COMPLEX *channelM     // M channels having frequency domain signal
```

5.2.2 Adaptive Selection of Main Frequencies

The next step is to compute the power spectrum of the signal and then adaptively select the main frequencies present in the power spectrum. A simple threshold method based on frequency bin is used to select the main frequencies. In this method depending on the number of frequency component we would like to analyze a threshold is defined and the

frequencies above that threshold will be chosen. As in real time the speech spectrum keeps on changing because of various factors like change in speakers pitch, changing SNR etc. Therefore the threshold value cannot be fixed to a numerical value. Also to find the main frequencies sorting the whole power spectrum is time consuming. The better approach is to define the whole power spectrum in terms of bin and choose the highest frequency component in that bin and then fix the threshold adaptively depending on the number of frequency component and select the main frequencies above that threshold. The bin size can be chosen depending on the frequency resolution.

In implementing this threshold based method, initially power spectrum is calculated from complex frequency domain signal for half of the BUFLEN. This part of the algorithm is implemented in function `peakfind()`. The parameters required for implementation are shown below:

```
short int bin=4;          // depends on frequency resolution
float localmax;           // varies for every bin
float threshold;          // depends on number of main frequencies
short int d= 64;          // BUFLEN/bin; BIN
float swap;               // shell sort to find main frequencies
static int flag2=1;       // to indicate a swap occurred
```

At first the power spectrum is divided in terms of bin, then largest frequency component is searched and selected as local maximum in that bin and the other components are zeroed as shown in Figure 5.6. These local maximas are stored in a separate array called Threshold array. In next step this array is sorted in decreasing order and then depending on the number of frequency components to be analyzed threshold is chosen. For example, if 4 frequencies need to be analyzed, then the threshold can be chosen as an arbitrary value less than `T[4]` as shown in Figure 5.6 in next page. In the end corresponding frequencies are selected which are greater than the threshold.

This part of the algorithm is implemented as the following function (`peakfind.c`) in the project:

```
peakfind(float *S, float *T, short int *freq)
```

where

```
float *S          // Power spectrum S[BUFLEN/2]
float *T          // Threshold array T[BIN]
short int *freq   // Indexes of main frequencies
```

The figures in next page shows the flow chart for implementation of function peakfind() and the power spectrum obtained in real time before and after the implementation of

bin based threshold method.

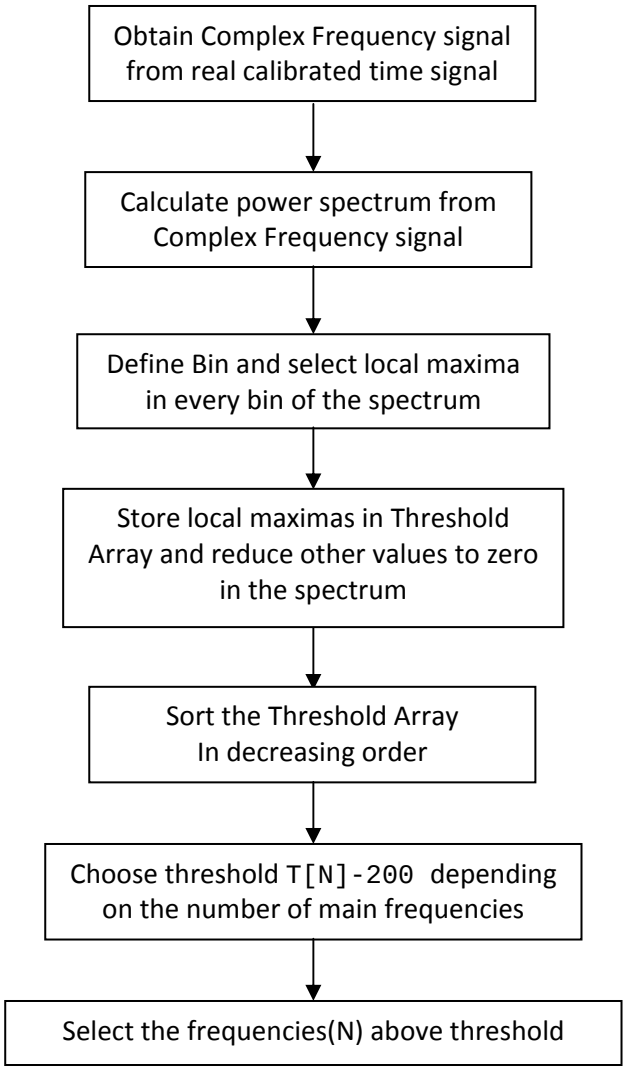

Figure 5.5 Implementation of function peakfind()

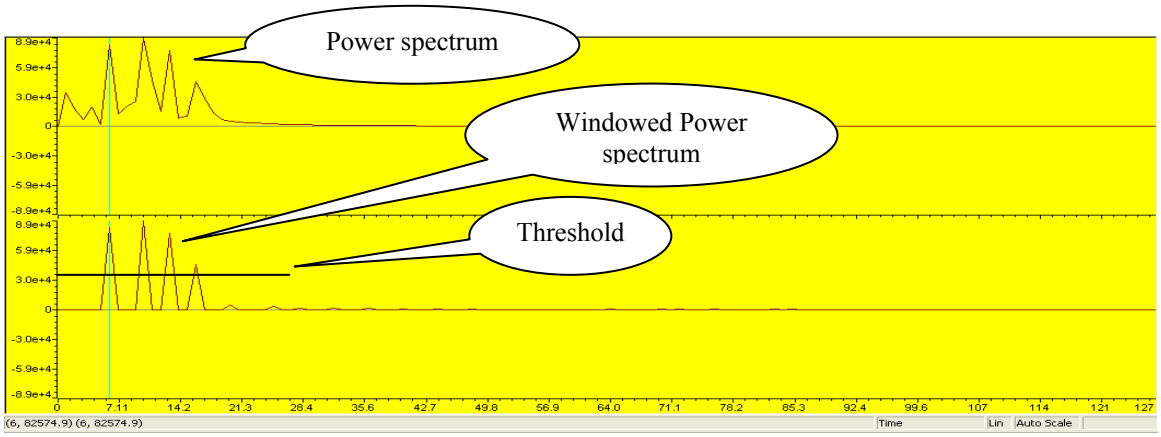

Figure 5.6 Power Spectrum for 4 frequencies before and after the peakfind function

5.3 Narrowband Spectrum & Incoherent Averaging

The next step is to calculate the narrowband MUSIC spectrum for every main frequency present in the spectrum and do the incoherent averaging of resultant MUSIC spectrums. At first output matrix are defined around main frequencies, for that purpose the complex values around these frequencies are picked from frequency spectrum. The output matrix is defined as M x N matrix where M indicates the number of complex points taken around the frequency and N indicates the number of microphones (M ≥ N). The main parameters used for this part of algorithm are declared in header file `micvariable.h` and explained below.

```
#define CUT 2           // Samples to be taken around frequency
#define M 6             // Number of complex points / samples
#define N 6             // Number of Microphones
#define SOURCE 1        // Number of Source
#define NROW 6          // Number of rows for MUSIC Spectrum
#define NCOL 6          // Number of columns for MUSIC spectrum
#define DIRECTION 181   // DOA Range of [−π/2, π/2]
```

The sample complex matrix obtained for one main frequency is `b[2*M][N]`. After obtaining the matrices for every main frequency, the next step is to perform the singular value decomposition of these matrices separately as discussed in Chapter 4.2.1 to obtain the eigenvalues and then the corresponding signal and noise subspace. These two sub-steps will be discussed in next two subsections.

5.3.1 Singular Value Decomposition of Complex Matrix

Earlier practical methods for computing the SVD resembles closely the Jacobi eigenvalue algorithm, which uses plane rotations or Givens rotations. However, these were replaced by the methods suggested by Golub & Kahan [17], which uses Householder transformations. In 1971, Golub & Reinsch [26] published a variant of the Golub/Kahan algorithm that is still the one most-used today.

The SVD of a matrix is typically computed by a two-step procedure. In the first step, the matrix is reduced to a bidiagonal matrix. This step takes $O(MN^2)$ flops. The second step is to compute the SVD of the bidiagonal matrix. This step can only be done with an iterative method, so the second step is in principle infinite (just as for eigenvalue algorithms). However, in practice it suffices to compute the SVD up to a certain

precision, like the machine epsilon. If this precision is considered constant, then the second step takes O(N) iterations, each costing O(N) flops. Thus, the first step is more expensive.

Implementing SVD algorithm in C is very complex and daunting task. One of the many variants of Golub/Kahan algorithms is suggested by Golub & Businger [27] and modified by Burkardt in FORTRAN [28], which can be used directly on complex matrix. The complex SVD is implemented as the following function in the project:

```
void CSVD(COMPLEX a[][N], int Mmax, int Nmax, int M, int N, int P,
          int NU, int NV, float *W, COMPLEX u[][M], COMPLEX v[][N])
```

The complex matrix `a[][N]` is destroyed by CSVD and also M ≥ N, otherwise rows should be filled with zero; the parameters used are explained as below:

```
COMPLEX a[][N]      // M x N matrix on which singular value
                    // decomposition is performed
Mmax = M            // Dimension of rows of complex matrix a and u
                    // also column of complex matrix u
Nmax = N            // Dimensions of complex matrix v
NU = M              // No. of columns to be computed in matrix u
NV = N              // No. of columns to be computed in matrix v
W                   // Computed Singular values of dimension 1 X N
COMPLEX u[][M]      // M x M matrix gives the NU columns of u
COMPLEX v[][N]      // N x N matrix gives the NV columns of v
```

This algorithm requires 200,000 cycles for one time calculation and as we are interested in up to four narrowband MUSIC spectrum, that means will consume approximately 800,000 cycles for estimating one DOA with four frequencies. Even though this SVD algorithm works fine, but has some stability issues, which is completely undesirable for a real time system. Other than that this algorithm has one more deficiency, all the dimensions defined in this algorithm are under the assumption that N ≤ 100. Although it suffices the current requirement but in future may cause adaptability problem, if there is a need to use dimensions greater than that what is specified.

The reasons discussed above leads us to use another SVD algorithm given in Press, Teukolsky and Vetterling [20]. This routine called svdcmp, performs SVD on matrix a and replacing it by matrix u and give W and v separately. This routine is based on a routine by Forsythe [29] and modified by Li & Cheng [30], which in turn based on the

original routine of Golub & Reinsch [26]. But this algorithm is real valued algorithm that means it does not perform SVD on complex matrix. The complex matrix should be converted to real matrix before using in this svdcmp algorithm. A method for converting a complex matrix into real matrix is suggested by Scibor-Marchocki [32]. This method is based on the following algorithm:

The complex-matrix C is isomorphic to the sum (A + i B), which in turn is isomorphic to the real-matrix (A, B; -B, A). The real-matrices A and B are obtained as the real and imaginary parts of the given complex-matrix C.

Using this algorithm *M x N* complex matrix is converted into real matrix as shown below:

$$(A + iB) \cdot (u + iv) = w(u + iv) \tag{5.4}$$

to the following 2M x 2N real matrix

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} = w \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.5}$$

This implies that if w_1, w_2……w_N are the eigenvalues of complex matrix, then eigenvalues of Eq. 5.5 are w_1, w_1, w_2, w_2…… w_N w_N. The $(u + iv)$ and $i(u + iv)$ are eigenvector pair according to the same eigenvalue. Therefore, one eigenvalue and one eigenvector are selected from each pair for separating the signal and noise subspace.
This SVD algorithm does not sort the eigenvalues and their respective eigenvectors as in Matlab SVD function. Hence the next logical step is to sort the paired singular values and their respective eigenvectors in decreasing order. After sorting these values the signal subspace is separated from noise subspace under the assumption that each main frequency has one source i.e. the highest value represents the signal subspace and the rest represents the noise subspace.
The whole SVD algorithm is implemented in the project as the following functions:

```
void cmplextoreal(float b[2*M][N], float a[2*M+1][2*N+1])
void svdcmp(float a[][2*N+1], int M, int N, float w[], float v[][2*N+1])
void sortnoisespace(float w[], float v[][2*N+1],
                    float noise[][2*N],int microphone)
```

The parameters used in these functions are explained as follows:

```
float b[2*M][N]       // 2*M x N complex matrix, where each column
                      // contains M complex samples
float a[2*M+1][2*N+1] // 2*M+1 x 2*N+1 real matrix on which SVD is
                      // performed and replaced by matrix u
M                     // Number of samples taken aroung main frequency
N                     // Number of Microphones in Microphone Array
W                     // Computed Singular values of dimension 1 X 2*N+1
float v[2*N+1][2*N+1] // 2*N+1 x 2*N+1 matrix
float noise[N-1][2*N] // Noise subspace
```

The indices of the array used in this routine starts from 1, therefore the dimension of each array in this routine is incremented by 1. This routine required 120,000 cycles for one SVD calculation along with conversion of complex matrix to real matrix that means needs 480,000 cycles for SVD calculation of four frequencies. Also this algorithm is stable in compare to the earlier one.

5.3.2 Spectrum and Tracker Algorithm

Once the signal and noise subspaces are separated, the next step is to find the MUSIC spectrum for every frequency present in the speech and then incoherent wideband MUSIC spectrum is calculated using Eq. 2.40, represented here again.

$$P_{INCOHERENT}(w_k,\theta) = \sum_1^M \frac{1}{a^H(w_k,\theta)Pa(w_k,\theta)} \quad (5.6)$$

As the MUSIC algorithm requires the prior knowledge of the steer vectors of the whole space for every main frequency, which are being calculated in a standalone program and added to the project (`spectrum.c`) as follows:

```
float steer[2* DIRECTION * NROW]       // E_R(ω,θ) in Eq. 2.35
float steer_herm[2* DIRECTION * NROW]  // E_R^H(ω,θ) in Eq. 2.35
```

where

```
DIRECTION 181  // spans the DOA from -π/2 to π/2 Resolution angle is 1°
NROW 6         // Number of rows (N) for Spectrum calculation
```

The steering vectors are calculated for every main frequency assumed to be present in

the spectrum based on the premise of converting the DOA back to phase delay for every microphone present in the array. DOA gives $x_{i+1}(t)$ a time delay of $\tau_i\theta(k) = \frac{d\ sin\theta_k}{v}$, the distance covered by the planar wave during this time is $d\ sin\theta_k$, and the period is $d\ sin\theta_k/\lambda$, so the complex steering vectors based on phase delay is given by $e^{j2\pi d\ sin\theta_k/\lambda}$. The flow chart for the calculation is shown in Figure 5.7.

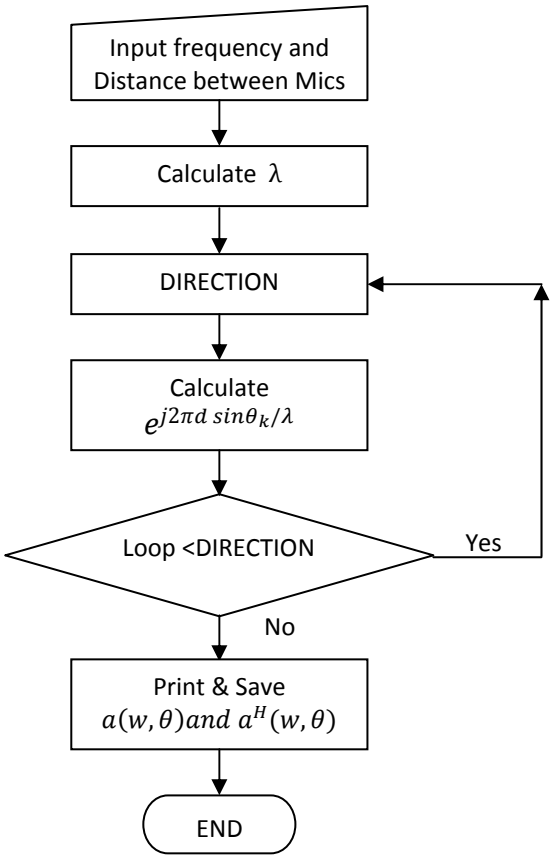

Figure 5.7 Process for calculating steering vector

In this last step of the complete algorithm, the calculated narrowband MUSIC spectrum for each main frequency are averaged together incoherently to get a resultant wideband MUSIC spectrum. The narrowband MUSIC spectrum is calculated as follows; MUSIC spectrum is calculated with the steering vector $\mathrm{a}(\mathrm{w},\theta), \mathrm{a}^{\mathrm{H}}(\mathrm{w},\theta)$ of each direction and the noise space vectors { e_{D+1}, e_{D+2}……e_N }, the latter are the output of the previous module—Noise Space. Based on $span[\mathrm{e}_{\mathrm{D}+1}, \mathrm{e}_{\mathrm{D}+2} \ldots\ldots \mathrm{e}_{\mathrm{N}}] \perp span[a(w,\theta_1), \ldots, a(w,\theta_D)]$, the MUSIC spectrum reaches its maximum when $\mathrm{a}(\mathrm{w},\theta)$ is the steering vector of DOA

for that frequency. Then a tracker algorithm is employed to search the approximated direction of arrival from the wideband MUSIC spectrum. This is being accomplished as the following function (`spectrum.c`) in the project:

```
void spectrum(float noise[][2*N],short int NOFREQ)
```

where

```
float noise[N-1][2*N]  // Noise subspace
short int NOFREQ    // Number of frequency present in the spectrum
```

Figure 5.8 shows the flow chart for calculating the Wideband spectrum and tracker algorithm used for four main frequencies presented in the spectrum.

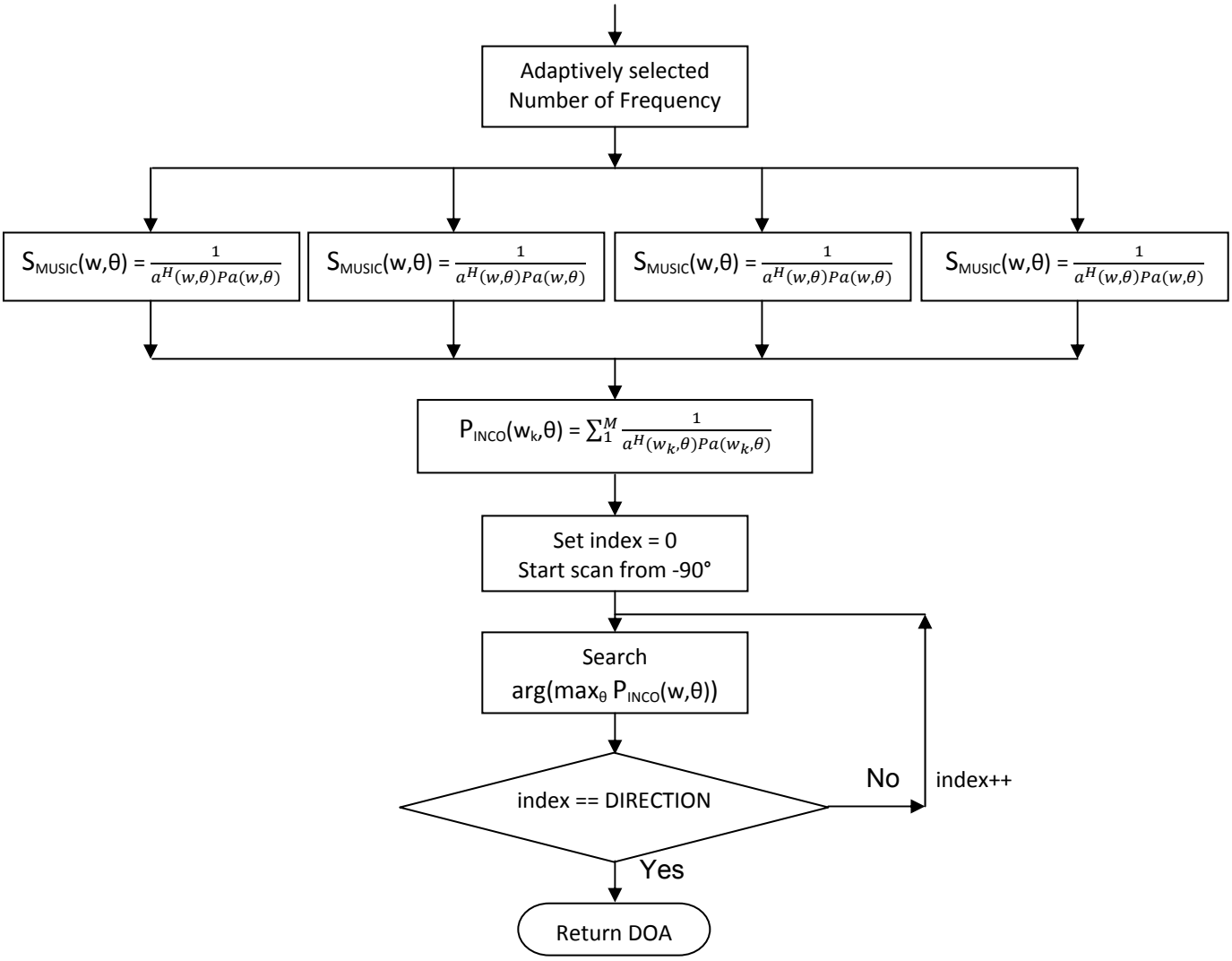

Figure 5.8 Incoherent Wideband Spectrum and Tracker algorithm for 4 Frequencies

5.4 Tests with Simulated Input Signal

After implementing the complete algorithm on DSP, algorithm's functionality is tested with simulated input signal generated from Matlab. The tests have been conducted for one frequency and two frequencies. The uncorrelated white noise is also added to the generated signals to get a feeling of the behavior of the algorithm in the presence of simulated noisy environment. The numbers of microphones assumed are 6, the spacing between microphones is 5.2 cm and the frequencies are 1100 Hz and 1900 Hz.

The simulated microphone signals generated from the Matlab are saved in the header file `microphone.h` in the project. The algorithm is ran for direction of arrivals from -40° to 40° at the step size of 10° and resolution of algorithm is 2°.

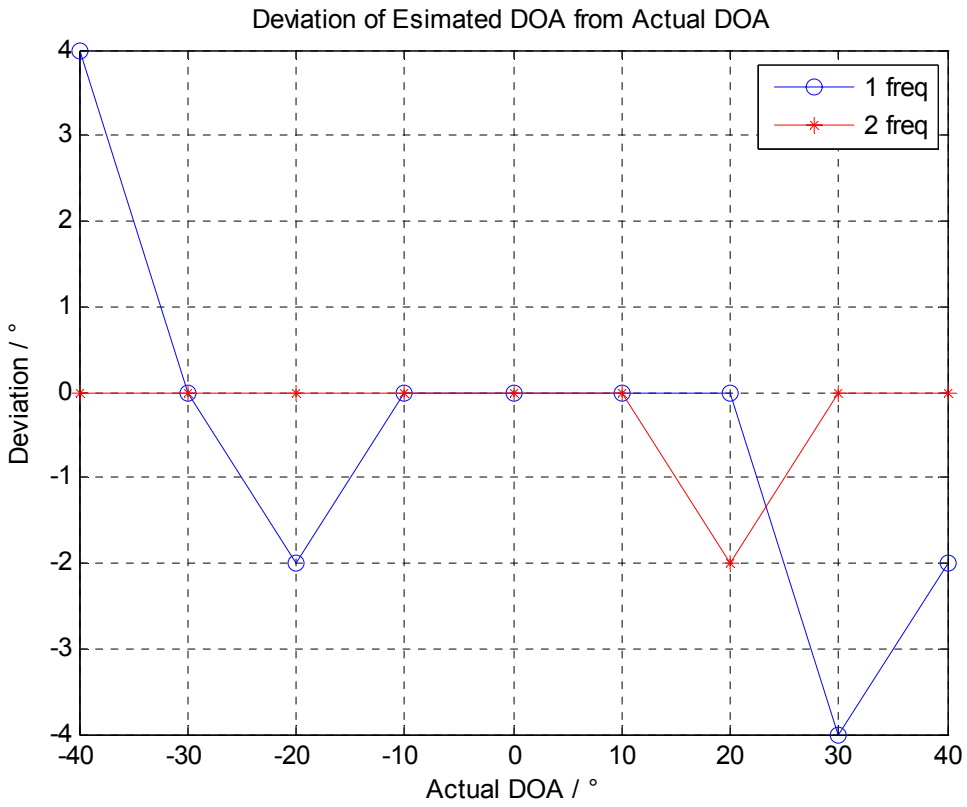

Figure 5.9 Estimated DAO for simulated input signal of 1.1 KHZ and 1.9 KHz

It can be easily observed that with simulated input signals the algorithm performed well even for one frequency and the results obtained are almost comparable except in few cases to the simulation results discussed in Chapter 4.3. Hence we can conclude that algorithm works fine and in principle do the required job.

6. Tests in Real time

To see the functionality and behavior of the algorithm in real time tests are conducted in a systematic way. At first the performance of self-calibrating algorithm is verified in estimating the DOA. For this purpose tests are done in anti-acoustic room. After verifying the algorithm, the whole system is moved to a classroom and tests are done in real time. In the following sections the microphone array setup and the other components of the system are introduced, thereon the results of systematic tests are discussed.

6.1 System setup

The microphones and their amplifying circuit are molded on the small PCB board, which are arranged in the linear fashion on a supporting frame. The microphones are connected to the power supply with separate connectors, below the microphones and the ADC channel of PCM3003 codec connected to the microphones using buses which in turn are connected to the DSK 6713. The whole setup is shown below

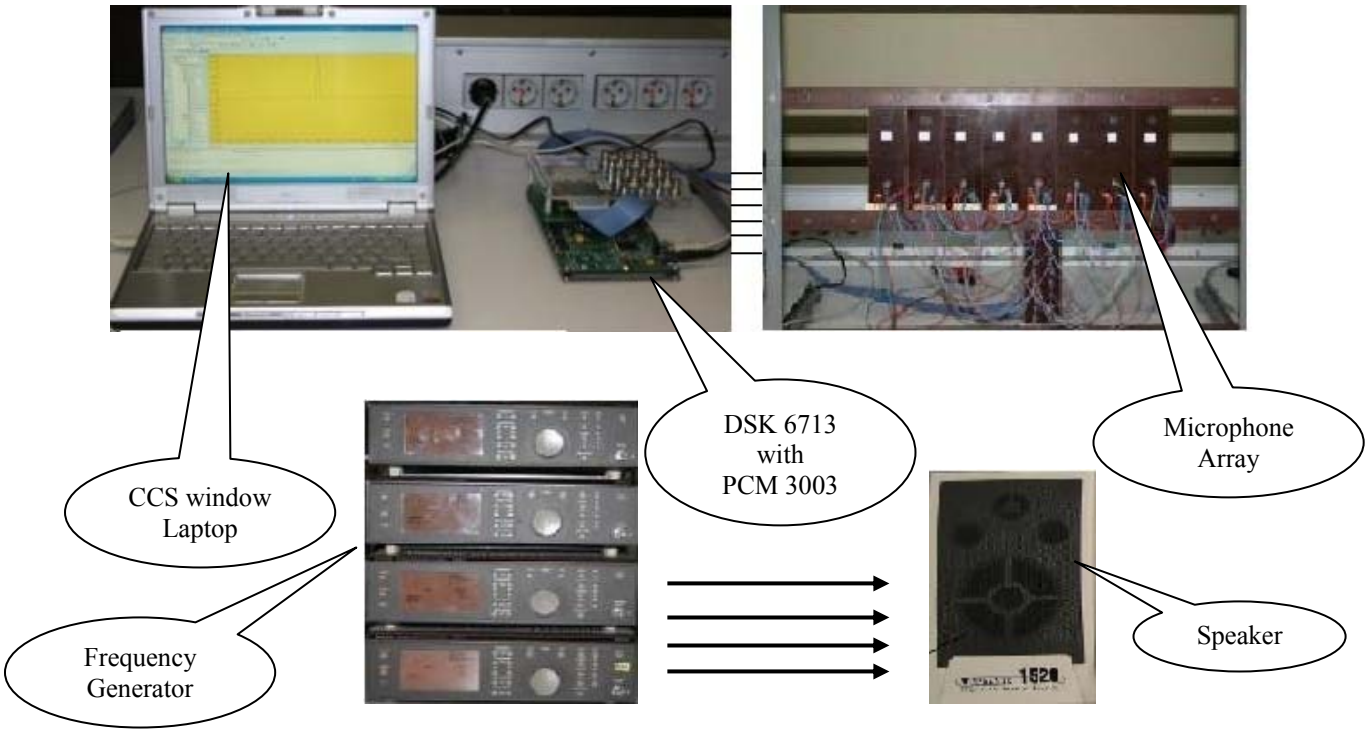

Figure 6.1 Pictorial view of Microphone array system with User PC

The audio sources (sine waves) are generated using frequency generators and are played using speaker. The distance between the array and speaker are varied from 2 to

4 meters, avoiding the spatial aliasing effect (depending on the length of array and highest frequency present in the spectrum). The frequencies of audio sources which has been used are 1100 Hz, 1900 Hz, 2400 Hz and 3000 Hz. For our application the optimum distance (R) between the array and source is measured as follows:

Maximum frequency present in the spectrum (f_{max}):	3400 Hz
Spacing between microphones of array (d):	5.2 cm
Length of microphone array (L):	27 cm
Velocity of sound waves at room temperature (c):	343 x 10^2 cm s^{-1}

$$R = \frac{2 * (L)^2}{c} f_{max}$$

The minimal calculated distance for satisfying far field condition is 144.52 cm. As the source is kept at 200 cm and beyond that, hence all test cases satisfy the far-field requirement.

6.2 Tests in anti-acoustic room

The complete system is placed in an empty part of anti-acoustic room. The microphone array is placed in the middle of the room and then the DSK 6713 and the supporting stuff along with laptop are placed behind it. The speakers are placed near the wall at a distance of 300 cm from array and the frequency generators behind that. This room can be assumed to be having SNR greater than 40 dB. The whole purpose of testing the system in this room is to see the performance of self-calibrating algorithm and as a whole the functionality of the complete algorithm. The tests are conducted with audio source having one, two and three frequencies without using the self-calibrating algorithm and then with the self-calibrating algorithm. It was observed that the performance of the algorithm does not vary noticeably with change in distance between source and array like 200 cm to 400 cm and also they need not to be in light of sight condition, especially in case of two and three frequencies. Initially the functionality of the algorithm and the system is checked by running some tests. The results obtained were the expected one, even though it was not done in a systematic way. The systematic test results for one frequency, two frequencies and three frequencies with un-calibrated and calibrated array are shown in Figure 6.2, 6.3 and 6.4 respectively.

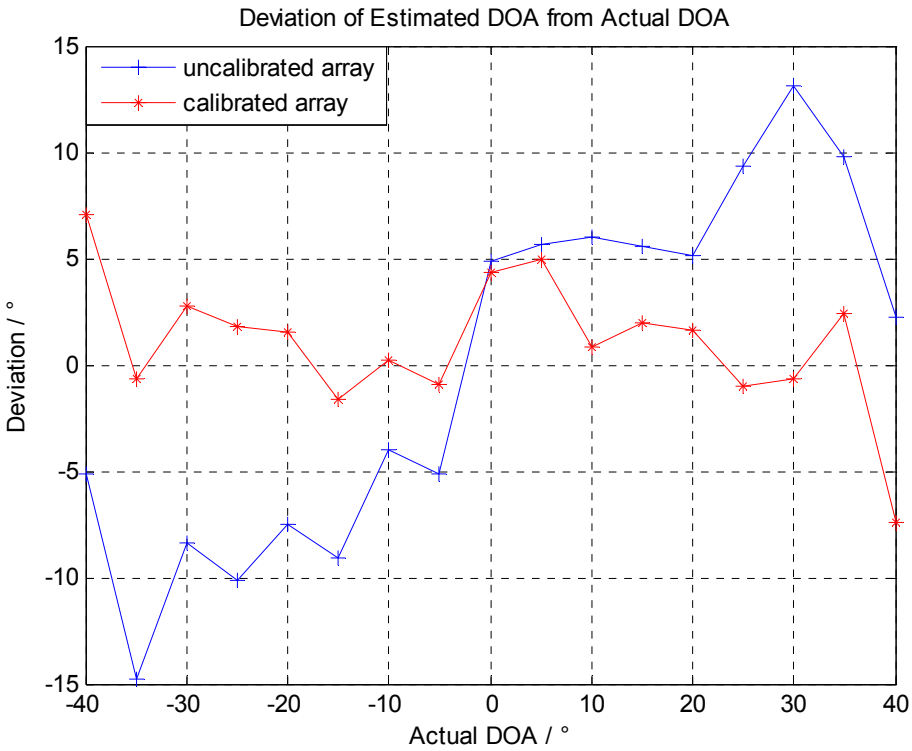

Figure 6.2 Estimated DOA for 1.1 KHz

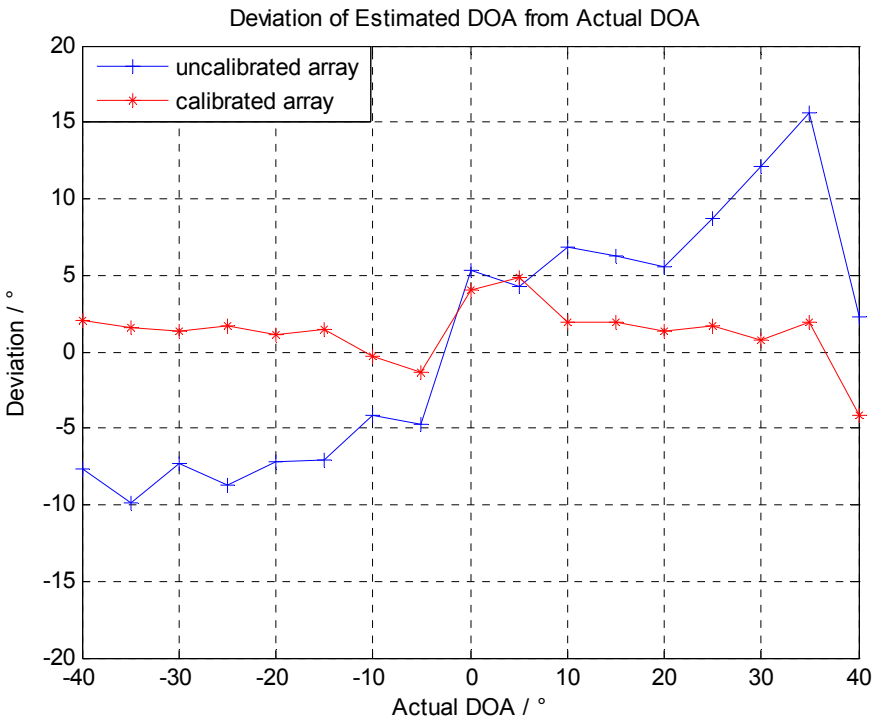

Figure 6.3 Estimated DOA for 1.1 KHz and 1.9 KHz

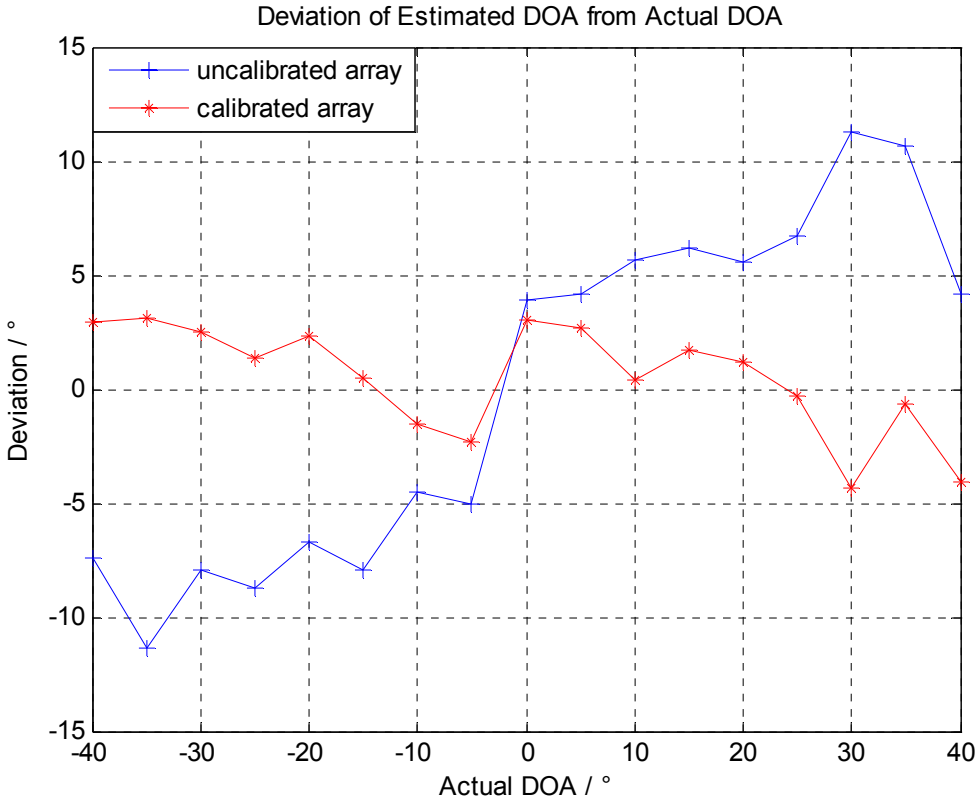

Figure 6.4 Estimated DOA for 1.1 KHz, 1.9 KHz and 2.4 KHz

It can be easily observed that the performance of the system is quite good with self-calibrating array and for two and three frequencies the variation in estimating the DOA is not so much different. For two frequencies the estimated DOAs is well within the range of ±5° of actual DOA, whereas for three frequencies the range is between ±4° of actual DOA. We can also observe that with the increase in number of frequency component present in the spectrum, the performance of the algorithm improves for un-calibrated array as well. Also a person can observe that the estimated directions of arrivals at θ and 180-θ are approximately mirror images of each other. This is because of the reason that the microphones in an array undergo a same delay irrespective of the ± direction of arrival.

6.3 Tests in reverberant environment

Now the complete setup is moved to a classroom and the tests are carried out by varying the distance between source and array as well as number of frequency. The microphone array is placed in the open end (middle) of the classroom and the speakers nearby the opposite wall in front of array. The distance between array and opposite wall

is 4.5 m and the source-array distance is varied between the two. The setting of setup in the room is shown in Figure 6.5. As the room has strong reverberation effect from all the three walls and also the floor because of high reflection coefficient, the array and source both are placed at the same height from floor (110 cm) to have line of sight propagation, which is required to differentiate between the direct path and indirect path. Although the room chosen has normal white noise for the testing, but lifts working just besides the room and behind the array (approximately at a distance of 5 m) makes the room very nearby to the real environment. All the tests are carried under the same condition as far as the parameters of the systems like spacing between microphones, samples etc. are concerned except the noise level keeps on changing in room because of lift's continuous movement. All estimated direction of arrivals shown and discussed are average values of 10 estimations.

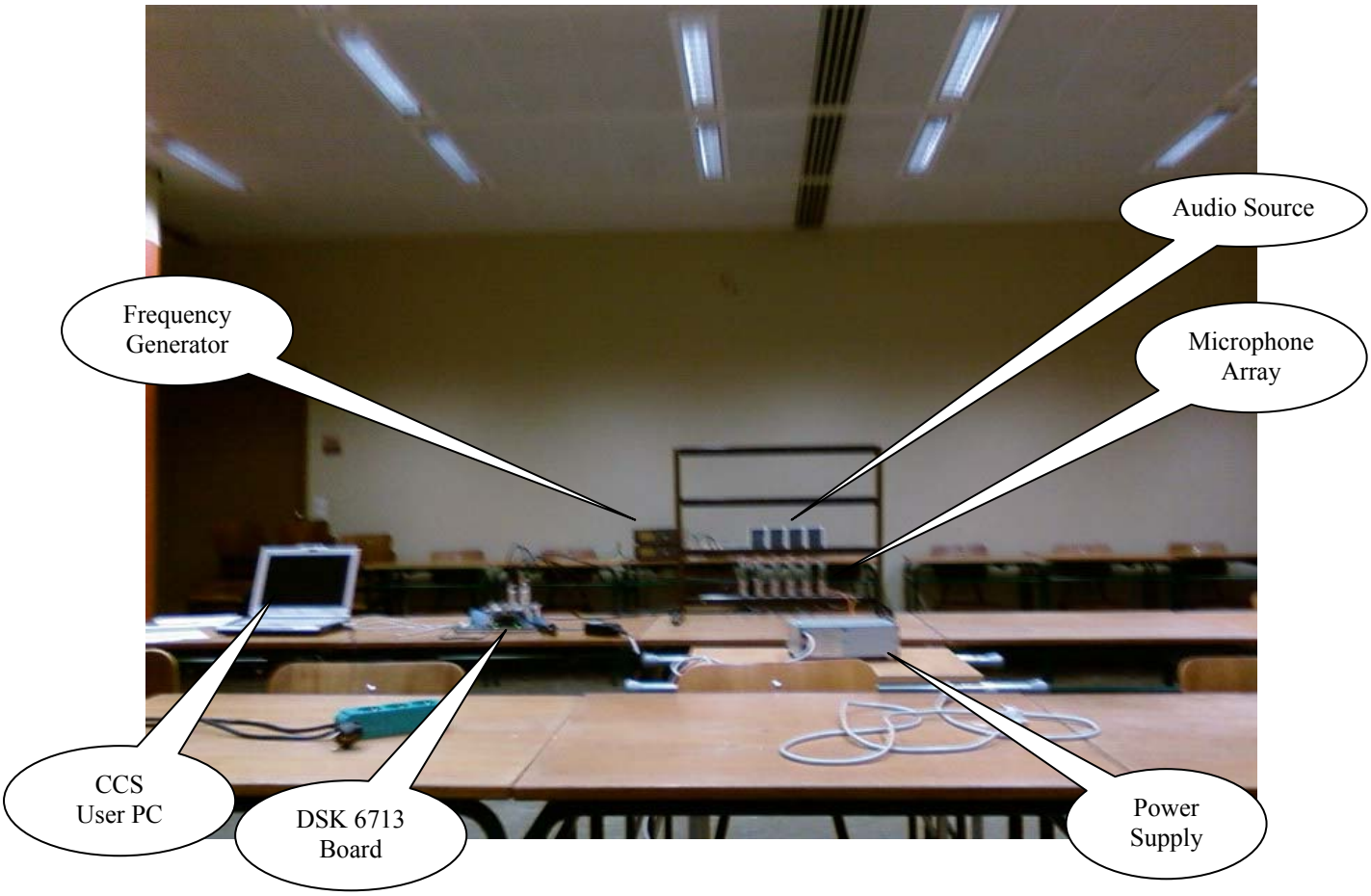

Figure 6.5 Setup of system in classroom

6.3.1 Tests with 6 Microphones

Initially the system is tested with the source placed at a distance of 300 cm from the array and moved the source from -40° to +40° at a step size of 5°. Of course the distance between the source and array will increase with the moving of source away from 0°, but in a general scenario a speaker will not move in a semicircular fashion. The results are shown in Figure 6.6. We can observe that for one frequency the results are worse, for direction of arrival beyond ±20° the estimation is bad that means the MUSIC algorithm is not able to differentiate between the calculated eigenvalues to span the signal subspace from noise subspace. Also it was observed that for some direction of arrival like +5° the estimated DOA is -5°, it can be attributed to the high reflections from the floor which was causing change in phases. For two frequencies the deviations are more or less similar to the one frequency except in few cases. For the three frequencies the estimation is far better than the two previous cases except for +25° the range of deviation is between ±8°. Then the performance of system is checked with four frequencies and as expected the results are improved over three frequencies. The general range of deviations is between ±6°.

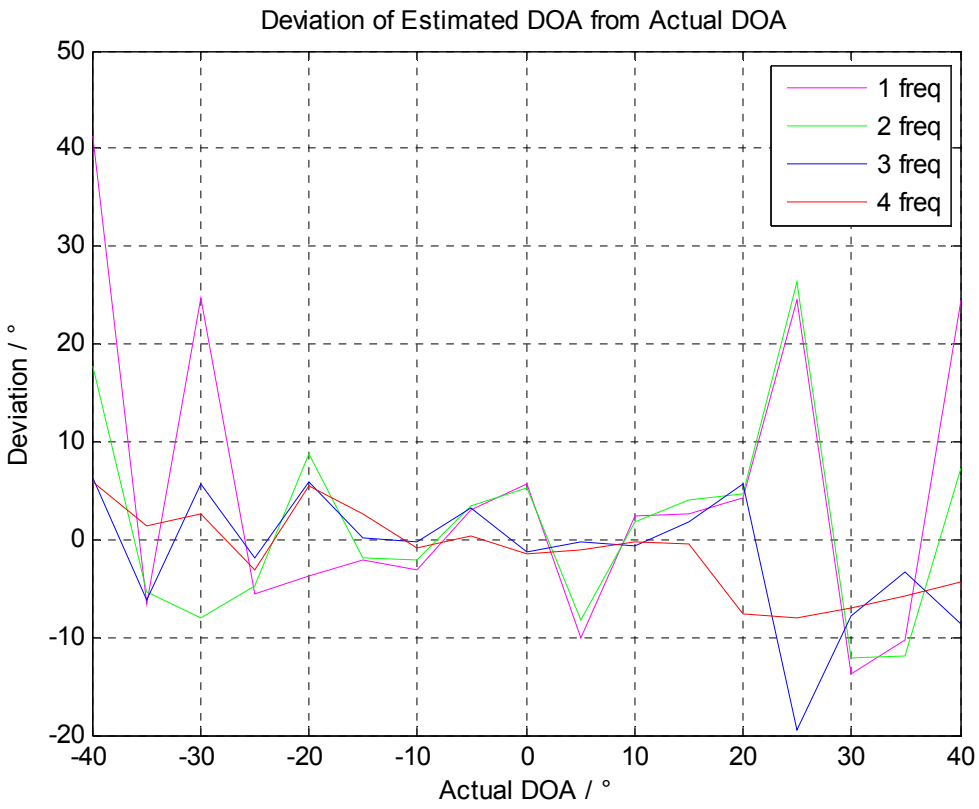

Figure 6.6 Comparison of EDOA for all frequencies for 6 Microphones & 300 cm

Another important parameter is the MUSIC spectrum; it was observed that as the number of frequencies increased the MUSIC spectrum becomes narrower. That means for four frequencies the MUSIC spectrum was the narrowest one and for 1 frequency it was the broadest one. Also it was noticed that as the source moves away from 0°, the wideband MUSIC spectrum becomes wider and wider. The results obtained in the class room are not better if a comparison is made with the estimations in anti-acoustic room. Figure 6.7 shows a comparison between the results obtained in anti-acoustic room and classroom for two and three frequencies. In anti-acoustic room the difference between two and three frequencies was not so much, but in a class room the difference is clearly visible and quite high. The reasons for this difference can be attributed to the SNR and reflections in both cases. In anti-acoustic room the SNR is well above 40 dB, whereas in real environment the SNR is low and also keeps on varying. In an anti-acoustic room there were only one or two reflections, whereas in classroom there are multiple sources of varying reflections.

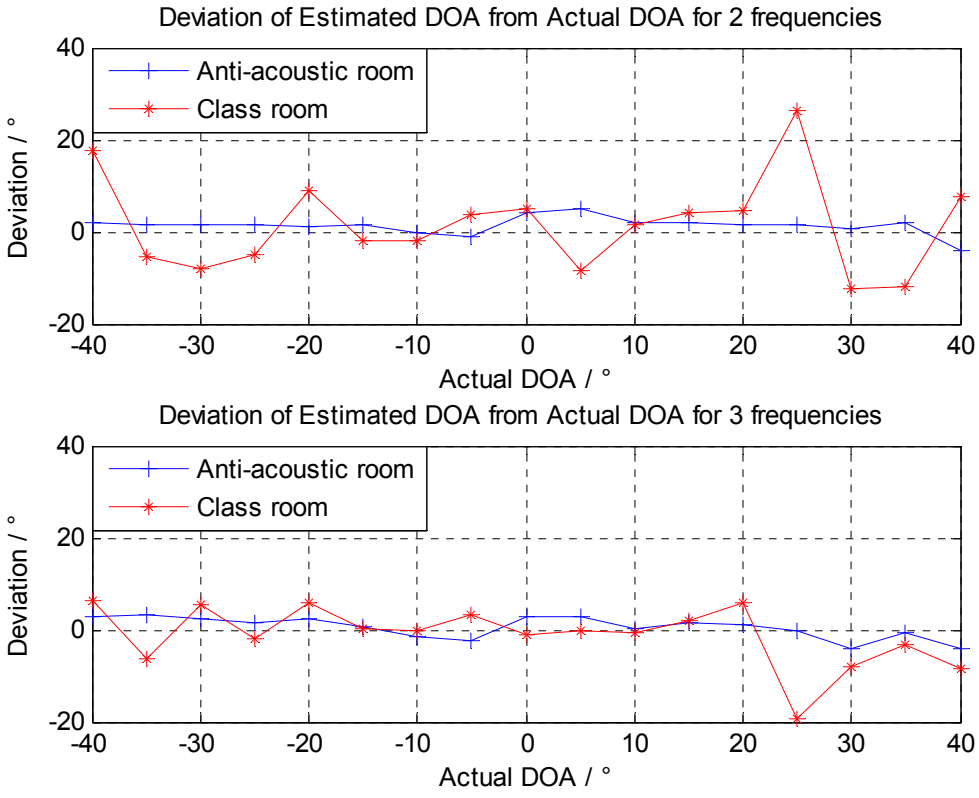

Figure 6.7 Comparison between class room and anti-acoustic room

We can easily see that with four frequencies the results are better than three frequencies and the general range of deviation from actual DOAs is between ±6°.

The reason for such a high deviation for +25° in all cases can be attributed to very high reverberation effect of the room at this particular angle and distance because when the source is moved left and right of this direction, the deviation was within the average range. As with the increase in the number of frequencies presents in the spectrum it actually improved. For one and two frequencies the MUSIC spectrum was around 0°, whereas for three frequencies it was around 11° and for four frequencies it was around 17°. It is a remarkable improvement in a kind of worst case scenario.

6.3.2 Effect of Source-Array Distance

It would be interesting to see the effect on system by increasing the Source-Array distance as it is known that the signal power drops with the increase in distance. Now the source is placed 400 cm away from the array and the tests are repeated for all the four frequency combinations. The results are shown in Figure 6.8. As expected the results are much worse than the earlier case for almost all frequencies, but not for all direction of arrival.

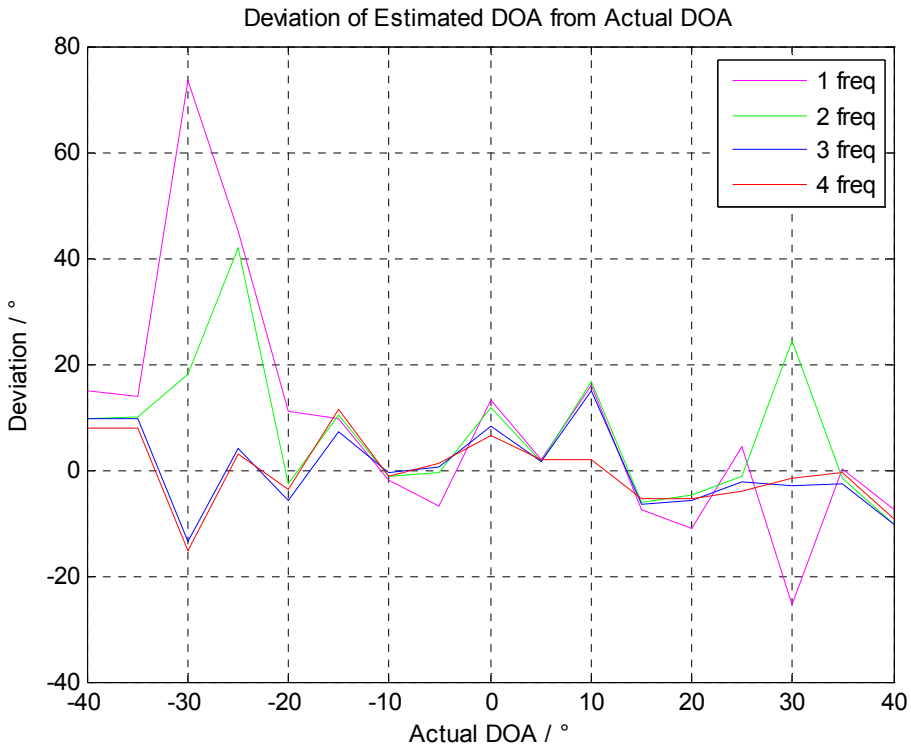

Figure 6.8 Comparison of EDOA for all frequencies for 6 Microphones & 400 cm

We can notice that the results in case of four frequencies are better than the other combinations and except for few directions of arrivals the results are almost comparable

to the earlier test with 300 cm.

When estimated directions of arrivals are compared for only four frequencies with the Source-Array distance varied from 200 cm to 400 cm. One can see that (as shown in Figure 6.9) the results for 200 cm and 300 cm are almost comparable, but a close look on the graph will reveal that for 200 cm the results are qualitatively better. But for 400 cm that's not a case and the results are bad in comparison to other two cases.

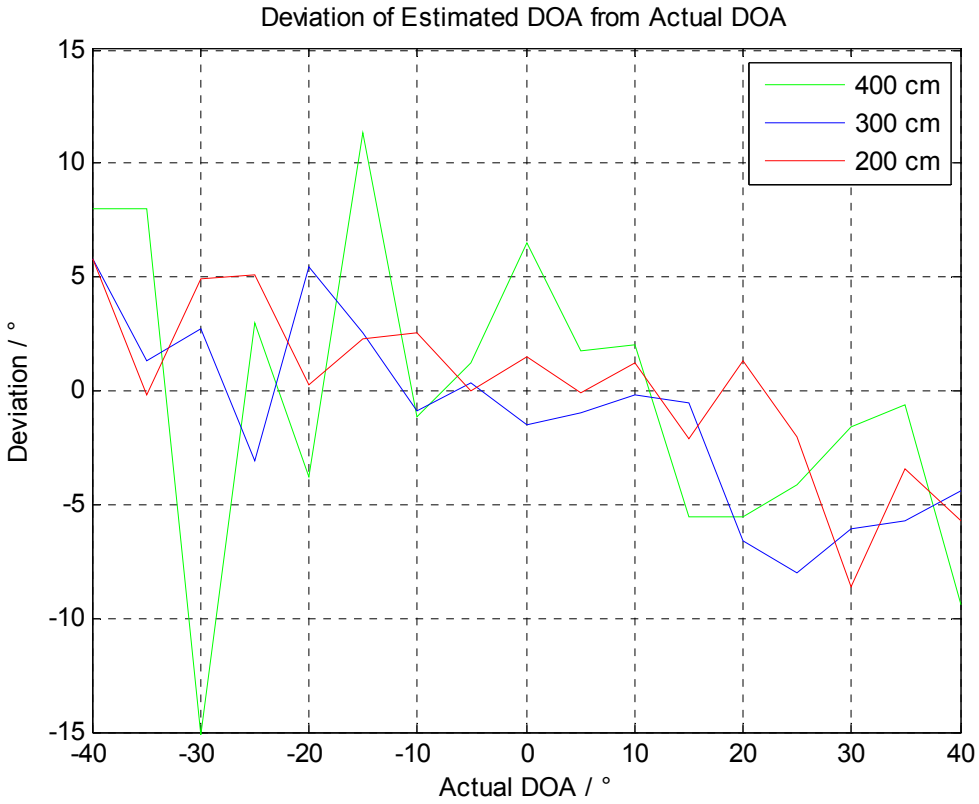

Figure 6.9 Comparison of four frequencies with Source-Array Distance

6.3.3 Tests with 8 Microphones

After performing a complete set of tests with six microphones, now the tests are repeated with eight microphones for two, three & four frequencies and Source –Array Distance of 300 cm. The aim is to see how much improvement will be achieved by increasing the number of microphones in an array. As we are interested in wideband spectrum therefore the tests are not performed for one frequency.

The results obtained are as per on expected line, for four frequencies the results are better than two and three frequencies as shown in Figure 6.10. The general range of deviation for four frequencies is between ±5°. Another important observation is the

behavior of algorithm at -40°, for all the combinations the deviation was above 10°. This deviation must have caused by the reverberation effect in the room at this particular angle. Also it was noticed that for four frequencies the estimated direction of arrival for many directions is almost equal to the actual direction of arrival or within a span of ±1°.

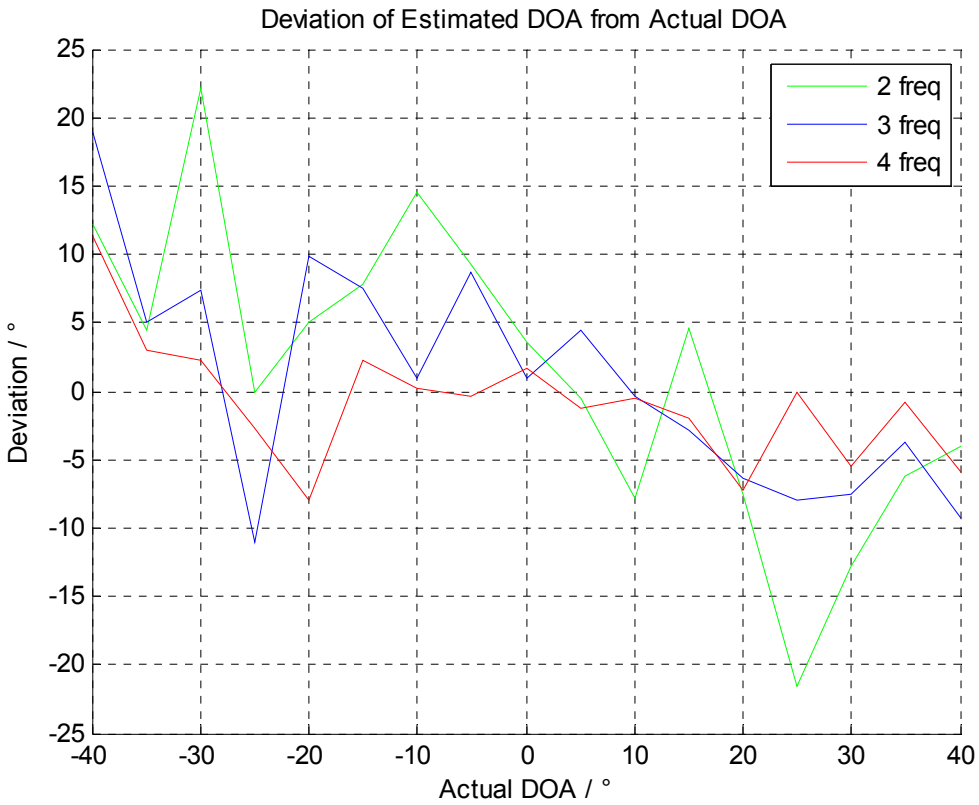

Figure 6.10 Comparison of EDOA for all frequencies for 8 Microphones & 300 cm

It would be quite interesting to make a case by case comparison between 6 and 8 microphones. The comparison is shown in Figure 6.11 in next page. One can notice that for the four frequencies the results with 8 microphones are qualitatively better than the results with 6 microphones. When a comparison is made between the wideband MUSIC spectrums obtained from the two cases, it was found that microphone array with 6 microphones has narrower wideband MUSIC spectrum than with 8 microphones. That means in most of the cases the MUSIC spectrum's beamform for six microphones was much narrower than that of 8 microphones whereas in terms of amplitudes of wideband MUSIC spectrum, the values is higher for 8 microphones.

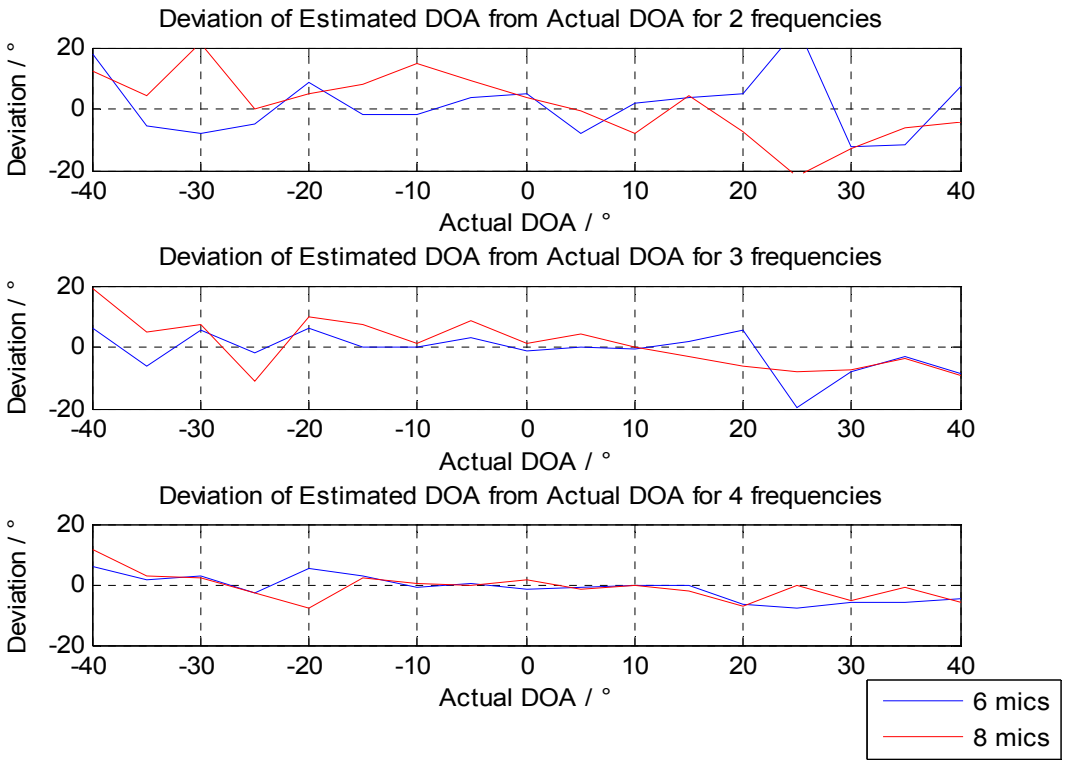

Figure 6.11 Comparisons between EDOA for 6 and 8 Microphones

6.4 Stability in Estimated DOAs

In the results discussed till now it is very clear that system performance is far better with four frequencies in almost all cases. Till now all the results discussed are average values of the estimated direction of arrival and it would be interesting to see the variation between the every estimated value for every direction of arrival with respect to the number of microphones. The comparative results between the 6 and 8 microphones for every direction of arrival are shown in following figures. It can be easily observed that in general for 8 microphones the variation between the estimated values is much less fluctuating in comparison to the 6 microphones barring few exceptions. In figures shown in next page the blue indicates estimated direction of arrival for 6 microphones and red indicates for 8 microphones and for every angle the graph shows 10 iterations.

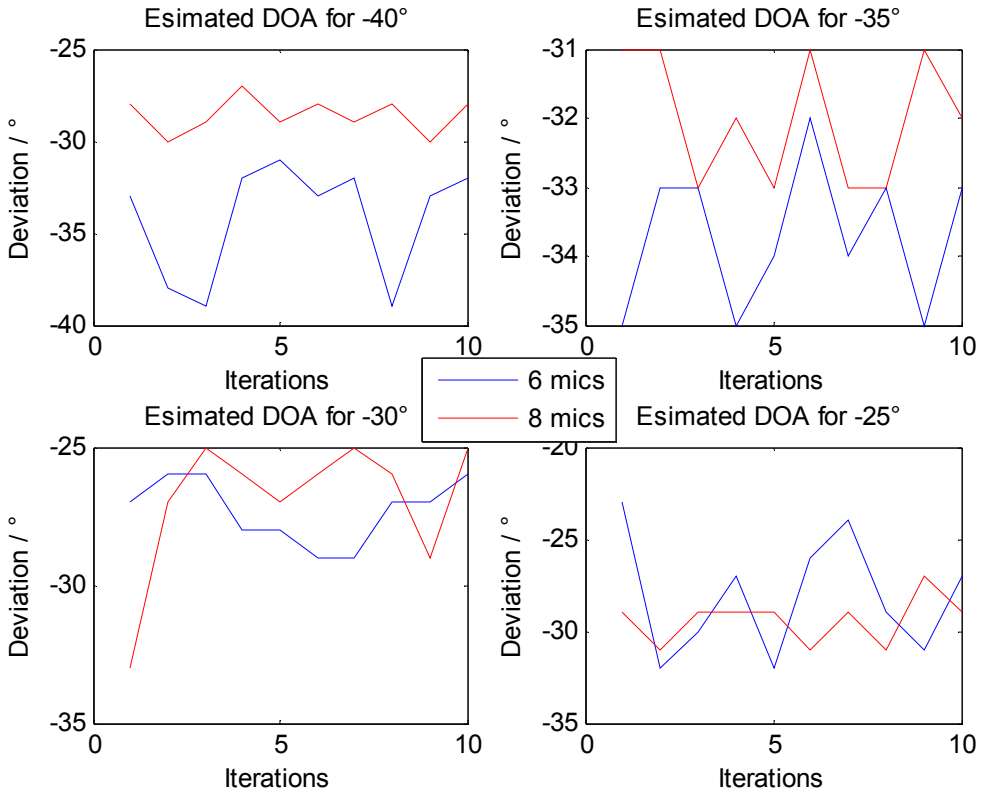

Figure 6.12 Estimated values for four frequencies

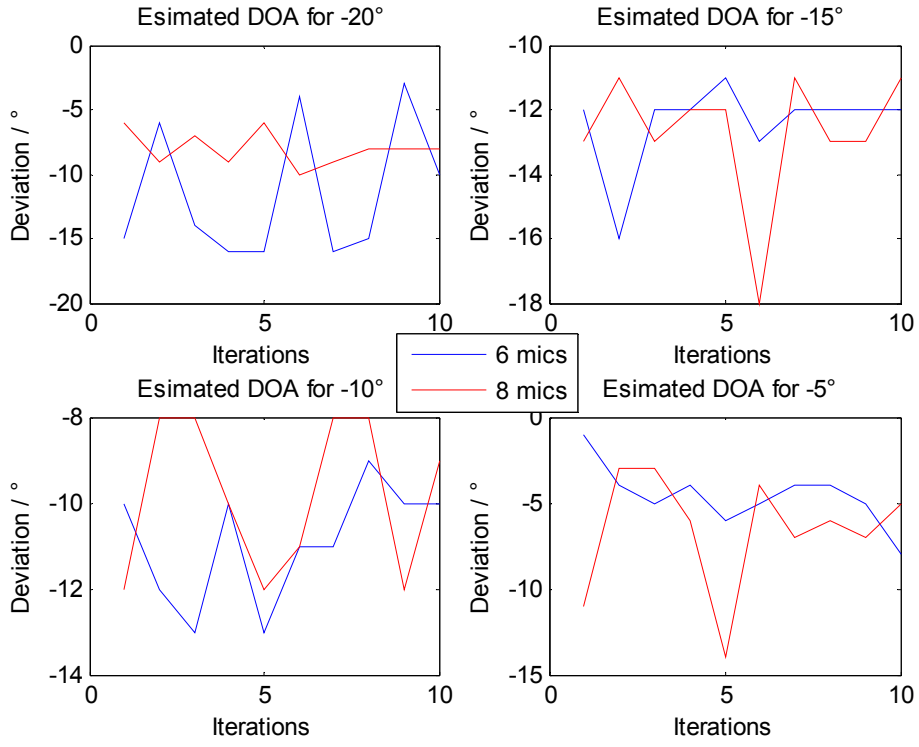

Figure 6.13 Estimated values for four frequencies

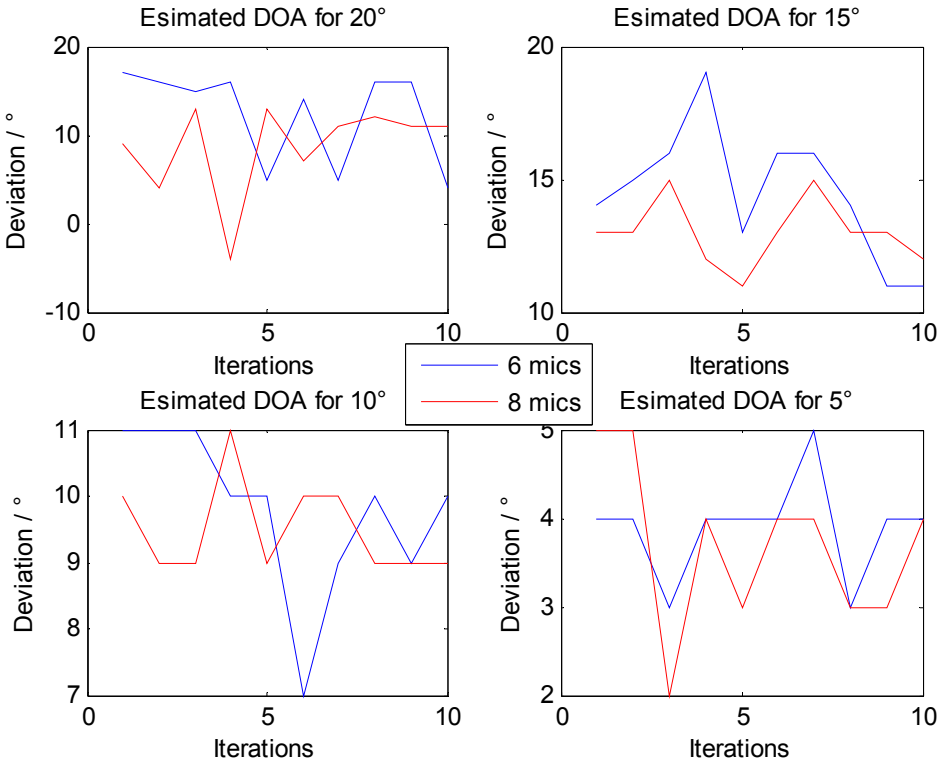

Figure 6.14 Estimated values for four frequencies

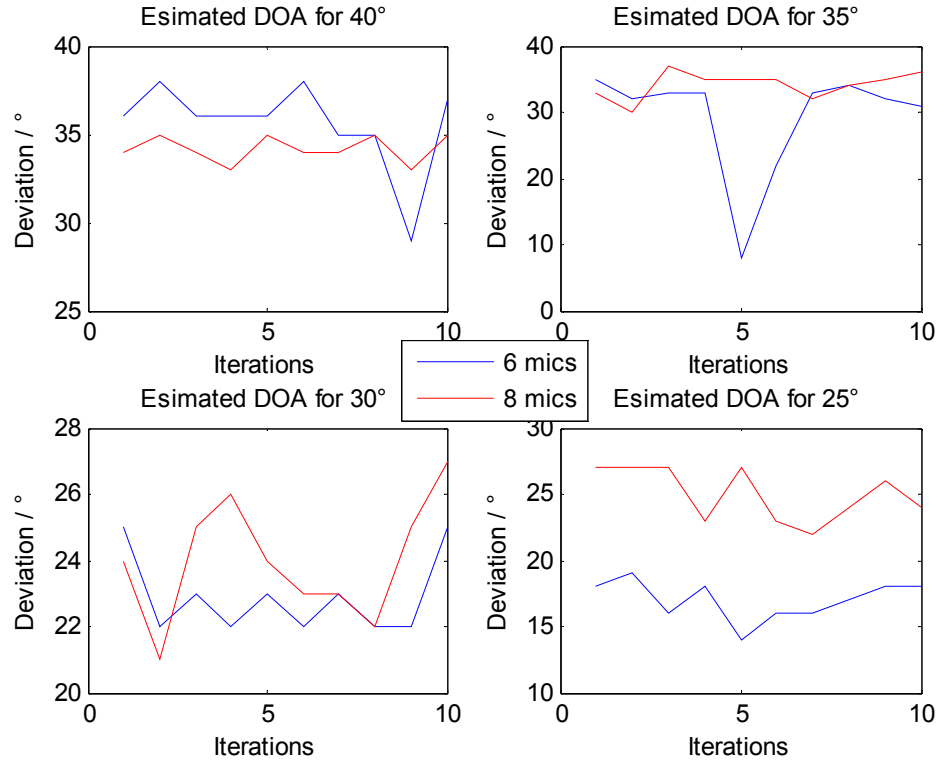

Figure 6.15 Estimated values for four frequencies

6.5 Comparison of Wideband MUSIC Spectrum

Another parameter to see the behavior of algorithm is to look at the beam pattern formed in the Code Composer Studio. The beam patterns showed remarkable change with the change in distance between source and array and with the movement of source from one angle to another. Mostly MUSIC spectrum is narrower and has highest amplitude at 0° and keeps on widening with the increase in angle. It was also observed that beam patterns formed were also affected with the change in power of main frequencies present in the spectrum. In the figures below are shown the MUSIC spectrum got in the Code Composer Studio for four frequencies with 6 and 8 microphones for -40°, 0° and +40°.

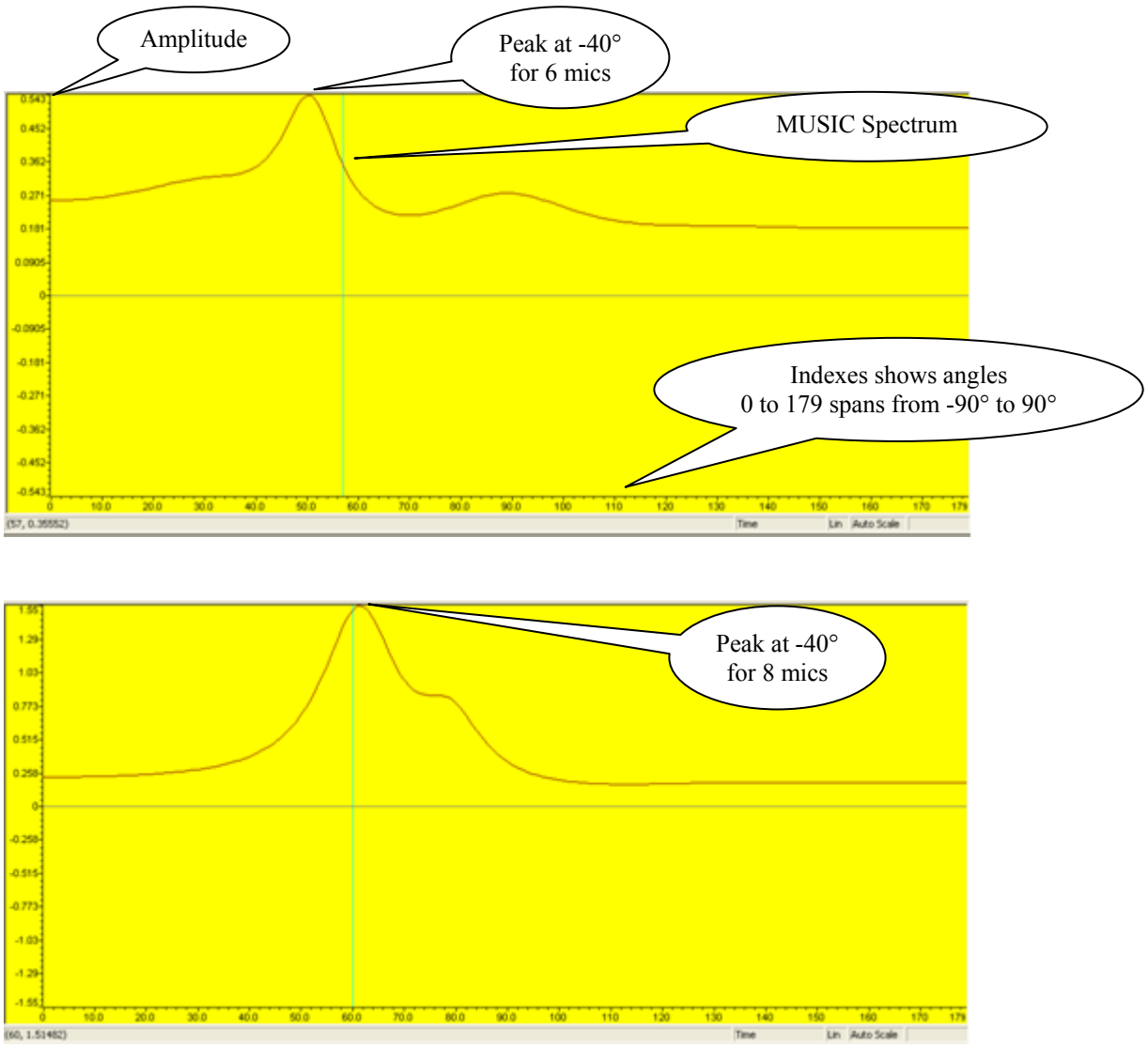

Figure 6.16 MUSIC Spectrum at -40° with 4 frequencies for 6 and 8 microphones

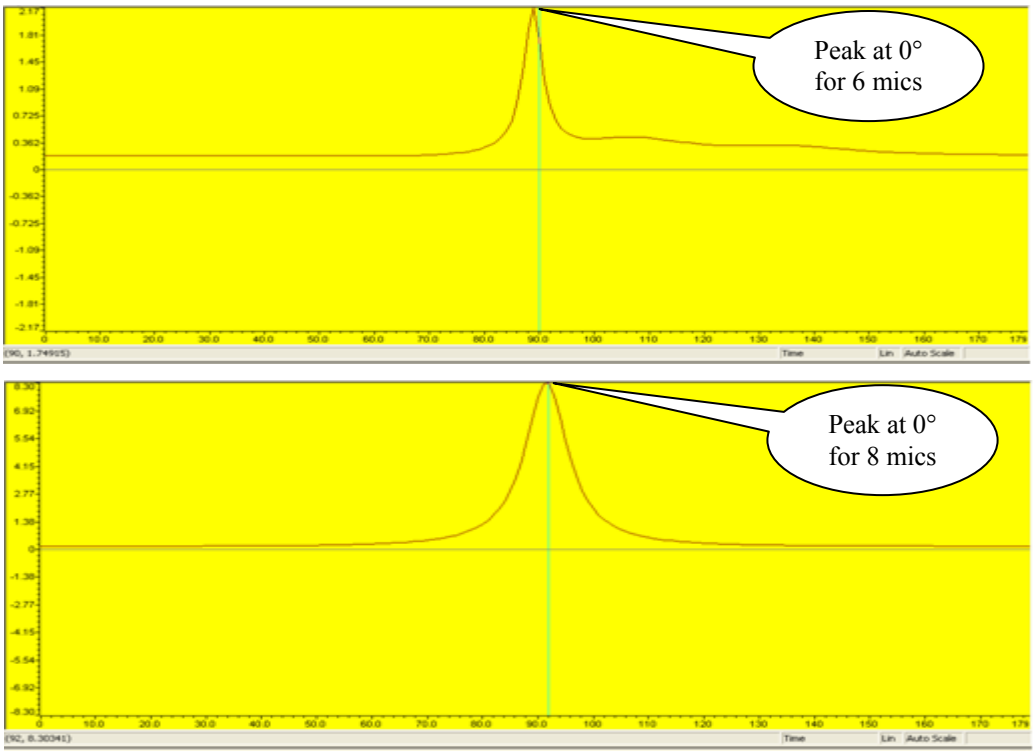

Figure 6.17 MUSIC Spectrum at 0° with 4 frequencies for 6 and 8 microphones

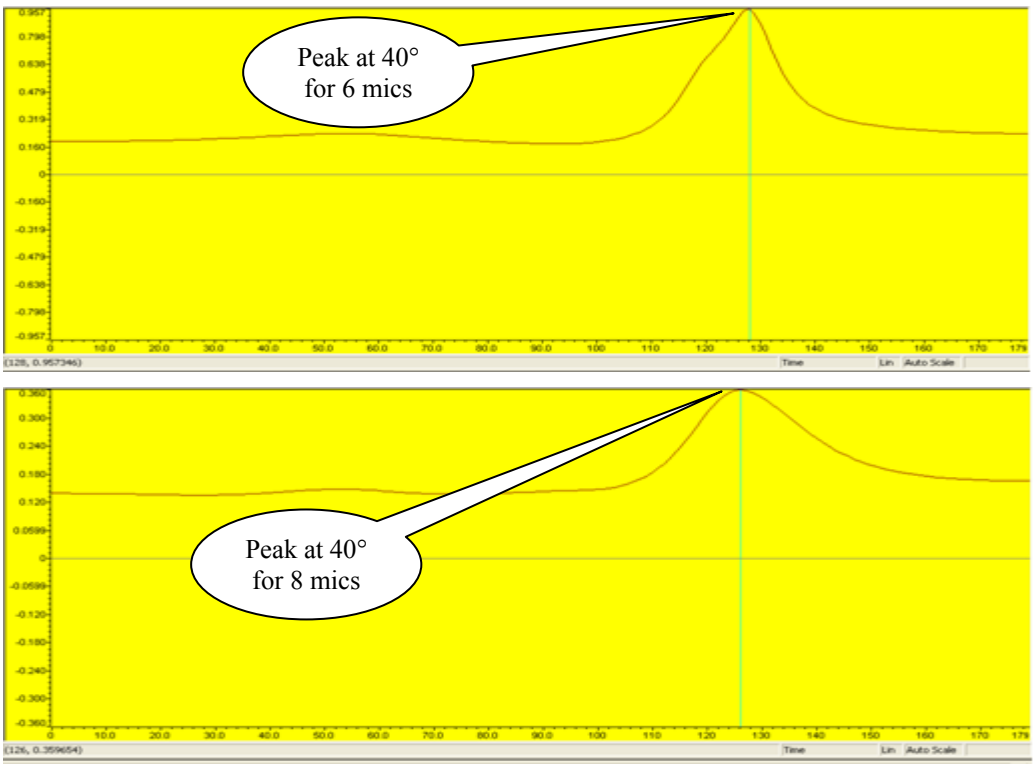

Figure 6.18 MUSIC Spectrum at 40° with 4 frequencies for 6 and 8 microphones

6.6 Analysis of Tests

The performance of algorithm in principle is good and the results obtained in real time environment are quite satisfactory. Although the results obtained in real time are not approximately similar to the results got in anti-acoustic room. Still the results obtained are quite good enough considering the real time environment in which the tests are performed. The tests with four frequencies were far better than other combination of frequencies in almost all cases i.e. irrespective of number of microphones in an array and the source-array distance. When the tests conducted with 6 and 8 microphones are compared, the results were almost comparable but qualitatively they were better for 8 microphones. Another important observation from the tests is that for most of the cases the algorithm works quite well in between ±20°, the deviation from actual angle was mostly within the range of 5°. This range is quite good enough if we look at kind of applications where this algorithm can be used like steering video camera towards the speaker in a seminar. As the source goes beyond ±20° the deviation is comparatively higher for some cases or angles. This is because of the reason that as the source move from 0° the source-array distance also increases. If at 0° the source-array distance is 300 cm that +40° it will be approximately 392 cm this implies that the signal power has been dropped by approximately 3 dB as discussed in Chapter 2.1.3. Hence the effect of noise increases as the source moves away from 0°. When the level of noise present in the room is measured, it was found that the Signal-to-Noise ratio was varying and sometimes goes below 15 dB during the all tests. As well as the room chosen has comparatively high reflections from floors in compare to a normal seminar room. In some tests for one or two direction of arrivals the estimation was bad in compare to average deviations. The only possible reason for this could be the high reflections from the floor or very low SNR at these particular angles and frequencies. When source is moved there was an improvement. Also it was observed that the wideband MUSIC spectrum gives a sharp beam at 0° and get broadens as the source moves towards ±40° also as the number of frequencies present in the spectrum increases the MUSIC spectrum becomes narrower and narrower. Also it was observed that the algorithm is not able to differentiate more than the two sources and only if they are at least 20° apart.

7. Conclusion & Future work

7.1 Conclusion & Summary of work

In this thesis work, a DSP based real time system to localize a source in wideband is developed, which is based on incoherent wideband MUSIC algorithm. The system is developed using TI's DSK 6713 board (TMS320C6713 DSP) with daughter card PCM 3003 codec and adjustable Microphone array with pre-amp.

The algorithmic implementation can be divided in four phase. In first phase, a self-calibrating algorithm is developed for the Microphone array to calibrate the signals received through different microphones in terms of amplitude and phase, as the microphones used do not have similar characteristics. In second phase, a peak-search algorithm is developed based on bin-threshold method to select the frequencies having higher energy than other frequencies and define a subband around these main frequencies. In the third phase, the wideband signal is decomposed into the number of narrowband spectrum depending on the main frequencies and then MUSIC algorithm is applied to each subband using SVD method. In the last phase, all MUSIC spectrums calculated are incoherently averaged and then a tracker algorithm is employed to search for the peak and the corresponding Direction of arrival.

Before implementing the algorithm on hardware, the behavior of algorithm was tested in simulated environment by varying the various parameters. It was observed that the performance of algorithm was very good in the simulated environment even at very low SNR value. The performance of algorithm improves with the increase in number of frequencies, especially with three and more frequencies. The algorithm works very finely when the source is in front of array or within the range of ±70° and starts behaving abruptly when the source goes beyond ±70°.

The system was at first tested in anti-acoustic room having high SNR and with almost no reverberation effect. The purpose was to test the system with and without self-calibrating algorithm and it was found that the results obtained with self-calibrating algorithm were quite good enough.

Thereafter the performance of the system was tested in a class room with varying SNR

and strong reverberation effect. The tests were conducted in a systematic way, initially the system was tested with six microphones and the source-array distance is varied and also the source was moved between ±40° having frequencies from one to four. After that the system was tested with eight microphones. The algorithm is able to localize the source with the deviation of ±5° in the range of interest with four frequencies. It was observed that the system works better with the increase in number of frequencies and was best with four frequencies for both six as well as eight microphones. The performance of system decreases with the increase in source-array distance. Also it was seen that in real time the algorithm is not able to separate more than two sources and that too when they are at least 20° apart from each other. Still the performance of system depends on the room and its environment.

7.2 Improvement & Future work

Although the system worked quite satisfactory still there is a room for improvement, especially in case of microphone array. The microphone array can be mounted on a special frame structure having less coupling effect as well as having less reflection. Also the distance between the microphones was not the optimal one as well as all microphones connections should be made behind the microphone array to avoid near field effects.

Another improvement in the performance of system could be achieved using a 2-D array instead of 1-D array used in this project. It can be because of the fact that with 2-D array there will be more microphones and hence the more information will be available for processing in algorithm and probably the better estimation in direction of arrival.

To make the system more adaptive to real speaker the steering vector corresponding to main frequencies can be calculated within the algorithm instead of using pre-calculated steering vectors. Also with large Source-Array distance the results were not good as the signal received by microphone array weakens, to further enhance the signal an Automatic Gain Control can be implemented.

For localizing the audio source incoherent wideband MUSIC algorithm works fine in real time, but performance degrades when the source moves far away from centre. To further enhance the performance of the system a maximum power (MP) beamforming

array algorithm along with the wideband MUSIC algorithm can be implemented as suggested by Tung, Chen, Hudsen and Reed [36]. The algorithm suggested by them is for 2-D source localization and developed in two steps. Initially source is localized using wideband MUSIC algorithm and after that maximum power beamforming algorithm is applied to enhance the desired signal and attenuate undesired spatially distributed interferences and background noises.

Though the MUSIC algorithm works well in our tests and has high resolution property, newer algorithm such as ESPRIT algorithm can also be employed. ESPRIT (***Estimation of Signal Parameters via Rotational Invariance Techniques***) is a recently developed eigenspace-based technique that has the same excellent resolution properties as MUSIC, but is termed as much more computationally efficient.

8. REFERENCE

[1] R. Schmidt, “A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation”, PhD Thesis, Stanford University, 1981.

[2] R. Schmidt, “Multiple Emitter Location and Signal Parameter Estimation”, IEEE Trans. on Antennas and Propagation, VOL. AP-34, No. 3, March 1986.

[3] T. Pham, M. Fong, “Real-time implementation of MUSIC for wideband acoustic detection and tracking”, US ARL 1997, Automatic Target Recognition VII, 1997.

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

[5] H.P. Kölzer, “Mikrofonarray zur Ermittlung der Sprecherposition”, Fachvortrag, HAW, Hamburg, April 2004

[6] G.Su and M.Morf, “The signal subspace approach for multiple wide-band emitter location”, IEEE Trans. ASSP, Vol.. 31, No. 6, December 1983.

[7] T. Pham and B.M. Sadler, “Adaptive Wideband Aeroacoustic Array Processing”, IEEE 1996

[8] Anglia’s Panasonic Electronic components, Date 24-Sept-2008 URL: www.angliac.co.uk/product_search/datasheets/process.asp?datasheet_id=16700

[9] D.SignT D.Module.PCM3003 Technical Datasheet version 1.0, Feb 2005

[10] D.R. Brown, “Digital Signal Processing and Applications with the TMS320C671 3 DSK”, Workshop, Worcester Polytechnic Institute, Oct 2007

[11] Texas Instruments, “TMS320C6000 Optimizing Compiler v.6.1”, SPRU187O, User’s Guide, May 2008

[12] Kenneth Boyce, “Use microphone arrays for background acoustic noise suppression in portable devices”, July 2008.

[13] Texas Instruments, “Application using the TMS320C6000 Enhanced DMA”, SPRA636A, Application report, May 2001

[14] Texas Instruments, “TMS320C6000 McBSP Initialization”, Application Report, SPRA 488C, March 2004

[15] Speech Processing: Theory of LPC Analysis and Synthesis URL: http://cnx.org/content/m10482/

[16] Depatment of EEE, Imperial College London, Date 2-May-2008 URL: http://www.ee.ic.ac.uk/hp/staff/www/voicebox/lpc.html

[17] G. Golub and W. Kahan, “Calculating the singular values and pseudo inverse of

a matrix." J. SIAM NUmer. Anal. 2 (1965), 205-224

[18] N. Muller, L. Magaia and B. Herbst, "The Singular Value Decomposition and Image Processing.", University of Stellenbosch, South Africa

[19] D. Kalman, "A Singularly Valuable Decomposition: The SVD of a matrix", The American University, USA, Feb 2002

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, "Numerical Recipes in C: The Art of Scientific Computing", 2nd edition, 1995

[21] D. Van Compernolle, "Switching adaptive filters for enhancing noisy and reverberant speech from microphone array recordings", Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing vol. 2, Albuquerque, NM, USA, April 1990

[22] M. Buck, T. Haulick and H.Pfleiderer, "Self-calibrating microphone arrays for Speech signal acquisition: A systematic approach", ACM: Signal Processing Volume 86, Issue 6, Oct 2005

[23] R. Chassaing, "Digital Signal Processing and Applications with the C6713 and C6416 DSK", Wiley-Interscience, 2005

[24] E.C. Ifeachor and B.W. Jervis, "Digital Signal Processing: A Practical Approach", Edition II, Prentice Hall, 2002

[25] Texas Instruments, "TMS320C67x DSP Library", Programmer's Reference Guide, SPRU657B, March 2006

[26] G. H. Golub and C. Reinsch, "Singular Value Decomposition and Least Square Solutions**",** In J. H. Wilkinson and C. Reinsch, editors, Linear Algebra, volume II of Handbook for Automatic Computations, chapter I/10, pages 134-151. Springer Verlag, 1971

[27] G. Golub and P. Businger, "Singular Value Decomposition of a Complex Matrix", Communications of the ACM, Volume 12, Number 10, Oct 1969

[28] J. Burkardt, Virginia Polytechnic Institute & State University, 5-Aug-2008, URL: http://people.sc.fsu.edu/~burkardt/f77_src/toms358/toms358.f

[29] G.E. Forsythe, M.A. Malcolm and Moler, "Computer Methods for Mathematical Computations", 1977

[30] Y. Cheng and B. Li, **"**DSP based Audio Source Localization", Master Thesis, Hamburg University of Applied Science, Germany, Oct 2004

[31] S. Mey and R. Cajina, **"**DSP-gesteuertes Mikrofonarray zur Sprecherlokalisieru -ng mit Hilfe eines breitbandigen MUSIC-Algorithmus", Diplomarbeit, Hamburg University of Applied Science, Germany, March 2006

[32] R.I. Scibor-Marchocki, 22-Aug-2008 URL: http://www.rism.com/LinAlg/complex.htm

[33] HSR Medialab, Hochschule für Technik Rapperswill URL: http://www.medialab.ch/ds/praktikum/praktikum_2/original/Praktikum2.doc

[34] K. Varma, "Time-Delay-Estimate Based Direction-of-Arrival Estimation for Speech in Reverberant Environment", Master Thesis, Virginia Polytechnic Institute and State University, USA, Oct 2002

[35] S. Valaee, "Array Processing for Detection and Localization of Narrowband, Wideband and Distributed Source", PhDThesis, McGill University, Canada, May 1994

[36] T.L. Tung, D. Chen, R.E. Hudsen and C.W. Reed, "Source Localization and Spatial Filtering using Wideband MUSIC and Maximum Power Beamforming For Multimedia Applications", IEEE, 1999

[37] M.Brandstein and D. Ward, "Microphone Arrays: Signal Processing Techniques and Applications", Springer, 2001

[38] M.H. Hayes, "Statistical Digital Signal Processing and Modeling", Wiley, 1996

APPENDIX

This Master report contains an appendix of program listings, hardware descriptions etc. on a CD. This Appendix is deposited with Prof. Dr.-Ing Hans Peter Kölzer and Prof. Dr.-Ing Ulrich Sauvagerd.

Contents:

- Source Code
    - Project for 6 Microphones
    - Project for 8 Microphones
    - Project for steer vector calculations
    - Matlab GUI implementation of algorithm
- Master Thesis
    - Thesis report in pdf format

Declaration

I/we declare within the meaning of section 25(4) of the Examination and Study Regulations of the International Degree Course Information Engineering that: this Master report has been completed by myself/ourselves independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

City, Date, Signature

……..