A STUDY OF BEAMFORMING TECHNIQUES AND THEIR BLIND APPROACH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

in

Electrical Engineering

By

DEBASHIS PANIGRAHI, ABHINAV GARG & RAVI S. VERMA

Department of Electrical Engineering

National Institute of Technology

Rourkela

2007

A STUDY OF BEAMFORMING TECHNIQUES AND THEIR BLIND APPROACH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
in
Electrical Engineering

By

DEBASHIS PANIGRAHI, ABHINAV GARG & RAVI S. VERMA
Under the Guidance of

Dr. SUSHMITA DAS

Department of Electrical Engineering

National Institute of Technology

Rourkela

2007

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled "A Study Of Beamforming Techniques And Their Blind Approach" submitted by Sri Debashis Panigrahi , Abhinav Garg and Ravi S. Verma in partial fulfillment of the requirements for the award of Batchelor of Technology Degree in Electrical Engineering at National Institute of Technology Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Date

Prof Mrs. S.Das
Dept. of Electrical engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGMENTS

Its not to often one gets the chance to thank people for their influence on your life, show appreciation for their support and prove how truly grateful you are .I want to first begin by thanking my advisor Dr. Sushmita Das. Everyday for nearly a year's time I showed up at her office door with questions not only concerning this thesis but everything else for that matter. She had an open door policy, never turning me away. She has had a profound influence on my life not only professionally but personally as well by showing me that there is more to life than just studying. Her character is remarkable, her dedication to teaching is consummate and her compassion for students is unwavering. I would like to thank my mother for being a wonderful influence on my life. I want to thank her for supporting me whether it is food, money, a place to stay or more importantly with her attention. Thanks. I also want to thank my father for his continuous support from a time as early as I can remember straight through to this very day. He never missed one single practice, game, speech, conference, meeting, ceremony or any other activity for that matter. I think back on the experiences we shared together and realize that the greatest gift he ever gave me was his

time. Doing without asking in return, looking out for my best interest, and offering unconditional love are all ways in which he has impacted my life. He helped mold my character while serving as a moral compass. I wish everyone could enjoy a relationship with their father as I do with mine. There are not too many people in this life which solely give without asking for anything in return. All I can offer him is my heart felt thanks for many years of love and support .It is not the easy things in life that are the most rewarding; it is the things you labor over night after night.. All the long hours and late nights, not realizing at the time that it's the relationships formed in this life which shape the meaning of your existence. If you did not have other people to share experiences with, admit when you're wrong, encourage when you're right while offering

then life would be pretty dull. To all the people I may have forgotten thank you for your support, I can assure you that I greatly appreciate it.

Debashis Panigrahi, Abhinav Garg and

Ravi S. Verma

CONTENTS

ABSTARCT

Beamforming is a technique in which an array of antennas is exploited to achieve maximum reception in a specified direction by estimating the signal arrival from a desired direction (in the presence of noise) while signals of the same frequency from other directions are rejected. This is achieved by varying the weights of each of the sensors (antennas) used in the array. It basically uses the idea that, though the signals emanating from different transmitters occupy the same frequency channel, they still arrive from different directions. This spatial separation is exploited to separate the desired signal from the interfering signals. In adaptive beamforming the optimum weights are iteratively computed using complex algorithms based upon different criteria.

INTRODUCTION

Beamforming is generally accomplished by phasing the feed to each element of an array so that signals received or transmitted from all elements will be in phase in a particular direction. The phases (the inter element phase) and usually amplitudes are adjusted to optimize the received signal.

EXPERIMENTATION AND SIMULATION

For simulation purposes a 4-elemnt linear array is used with its individual element spaced at half-wavelength distance. The desired signal arriving is a simple complex sinusoidal-phase modulated signal of the following form,

$$s(t)=e^{j\sin(wt)}$$

The interfering signals arriving are also of the above form. by doing so it can be shown in simulations how interfering signals of the same frequency can be separated to achieve rejection of co-channel interference. Illustrations are provided to give a better understanding of the different aspects of the lms algorithm with respect to adaptive beamforming.For simplicity purpose the reference signal d(t) is considered to be the same as the desired signal

ANALYSIS

The various methods used by us to achieve adaptive antenna beamforming i.e.

$$y(t) = w^H x(t)$$

And then estimating the weight vector w using various algorithms listed below.

1.SMI algorithm

In this algorithm the weights are chosen such that the mean-square error between the beamformer output and the reference signal is minimized.

$$E\left[\{r(t) - w^H x(t)\}^2\right] = E[r^2(t)] - 2w^H R_r + w^H R_m w$$

2. LMS algorithm

This algorithm like the preceding one requires a reference signal and it computes the weight vector using the equation

$$w(n+1) = w(n) + \mu x(n)[d^*(n) - x^H(n)w(n)]$$

3. CMA algorithm

The configuration of CMA adaptive beamforming is the same as that of the SMI system discussed above except that it requires no reference signal

$$J_n = \frac{1}{2}E[(|y(n)|^2 - y_0^2)^2]$$

RESULTS

Among various methods used 4 beams firming is found to be the most effective algorithm is CMA algorithm.

CONCLUSION

The study is important in the determination of blind beam firming**.** The CMA technique of adaptive beamforming is better then lms algorithm technique.

REFERENCE

[1] Compton, R.T. Jr. Adaptive Antennas – Concepts and Performance. Prentice Hall. Englewood Cliffs, New Jersey. 1988.

[2] Haykin, Simon. *Adaptive Filter Theory.* Prentice Hall, Englewood Cliffs, New Jersey. 3rd Ed. 1996 .

List of figures

Chapter 1

INTRODUCTION

Adaptive Beamforming
Adaptive Beamforming Problem Setup

1.1 ADAPTIVE BEAMFORMING [5]

Adaptive Beamforming is a technique in which an array of antennas is exploited to achieve maximum reception in a specified direction by estimating the signal arrival from a desired direction (in the presence of noise) while signals of the same frequency from other directions are rejected. This is achieved by varying the weights of each of the sensors (antennas) used in the array. It basically uses the idea that, though the signals emanating from different transmitters occupy the same frequency channel, they still arrive from different directions. This spatial separation is exploited to separate the desired signal from the interfering signals. In adaptive beamforming the optimum weights are iteratively computed using complex algorithms based upon different criteria.

Beamforming is generally accomplished by phasing the feed to each element of an array so that signals received or transmitted from all elements will be in phase in a particular direction. The phases (the interelement phase) and usually amplitudes are adjusted to optimize the received signal. The array factor for an N-element equally spaced linear array is given,

$$AF(\phi) = \sum_{n=0}^{N-1} A_n e^{jn(\frac{2\pi d}{\lambda}\cos\phi+\alpha)}$$

Note that variable amplitude excitation is used.

The interelement Phase shift is given by

$$\alpha = -\frac{2\pi d}{\lambda_0}\cos\phi_0$$

φ_0 is the desired beam direction. At wavelength λ_0 the phase shift corresponds to a time delay that will steer the beam to φ_0.

1.2 Adaptive beamforming problem setup

To illustrate different beamforming aspects, let us consider an adaptive beamforming configuration shown below in figure.

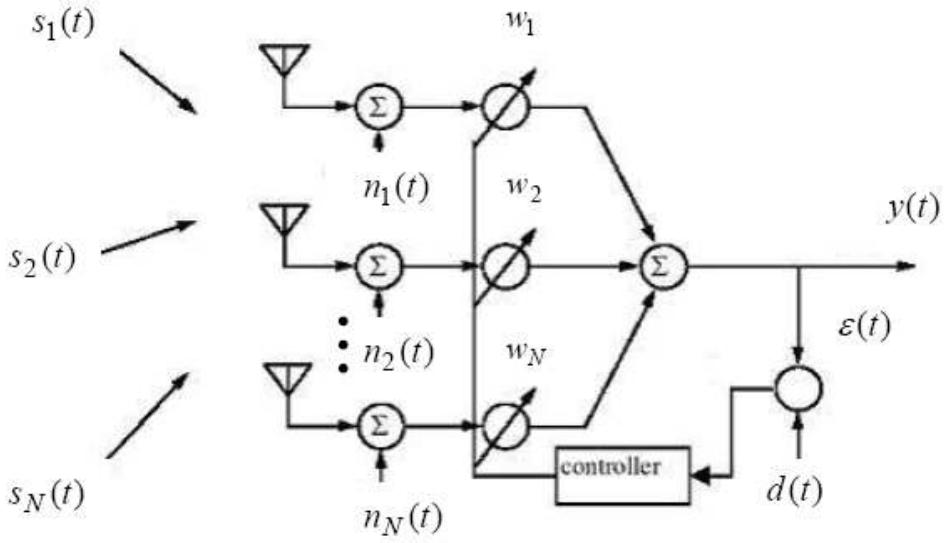

Figure 1.1: An Adaptive Array System

The output of the array y(t) with variable element weights is the weighted sum of the received signals si(t) at the array elements and the noise n(t) the receivers connected to each element. The weights wme iteratively computed based on the array output y(t) reference.

Signal d(t) that approximates the desired signal, and previous weights. The reference signal is approximated to the desired signal using a training sequence or a spreading code, which is known at the receiver. The format of the reference signal varies and depends upon the system where adaptive beamforming is implemented. The reference signal usually has a good correlation with the desired signal and the degree of correlation influences the accuracy and the convergence of the algorithm.

The array output is given by

$$y(t) = w^H x(t)$$

Where denotes the complex conjugate transpose of the weight vector w.

In order to compute the optimum weights, the array response vector from the sampled data of the array output has to be known. The array response vector is a function of the incident angle as well as the frequency. The baseband received signal at the N-th antenna is a sum of phase-shifted and attenuated versions of the original signal. Si(t).

$$x_N(t) \cong \sum_{i=1}^{N} a_N(\theta_i) s_i(t) e^{-j2\pi f_c \tau_N(\theta_i)}$$

The si(t) consists of both the desired and the interfering signals.

Rk (θi) is the delay, fc is the carrier frequency.

$$a(\theta_i) = [a_1(\theta_i)\ e^{-j2\pi f_c \tau_1(\theta_i)}, a_2(\theta_i)\ e^{-j2\pi f_c \tau_2(\theta_i)} \ldots\ldots\ldots\ldots\ldots a_N(\theta_i)\ e^{-j2\pi f_c \tau_N(\theta_i)}]^T$$

Now,

$$A(\theta) = [a(\theta_1)a(\theta_2)....a(\theta_d)]$$

$$S(t) = [s_1(t)s_2(t)....s_d(t)]^T$$

So that,

$$x(t) = A(\theta)S(t)$$

With noise,

$$x(t) = A(\theta)S(t) + n(t)$$

a(θ) is referred to as the array propagation vector or the steering vector for a particular value of a.

The beamformer response can be expressed in the vector form as,

$$r(\theta, \omega) = w^H a(\theta, \omega)$$

This includes the possible dependency of a(θ) on ω as well.

To have a better understanding let us re-write x(t) in equation by separating the desired signal from the interfering signals. Let s(t) denote the desired signal arriving at an angle of incidence θ0 at the array and the ui(t) denotes the number of undesired interfering signals arriving at angles of incidence θ_i. It must be noted that, in this case, the directions of arrival are known a priori using a direction of arrival (DOA) algorithm.
The output of the antenna array can now be re-written as;

$$x(t) = s(t)a(\theta_0) + \sum_{i=1}^{N_u} u_i(t)a(\theta_i) + n(t)$$

where,
a(θi) I s the array propagation vector of the i^{th} interfering signal.
a(θ0) is the array propagation vector of the desired signal.

Therefore, having the above information, adaptive algorithms are required to estimates(t) from x(t) while minimizing the error between the estimate s(t) and the original signal s(t).

Let d*(t) represent a signal that is closely correlated to the original desired signal s(t).d*(t) is referred to as the reference signal, the mean square error (MSE) ε2(t) between the beamformer output and the reference signal can now be computed as follows;

$$\varepsilon^2(t) = [d^*(t) - w^H x(t)]^2$$

After taking an expectation on both sides of the equation we get,

$$E\{\varepsilon^2(t)\} = E\{[d^*(t) - w^H x(t)]^2\}$$

$$E\{\varepsilon^2(t)\} = E\{[d^2(t)\} - 2w^H r + w^H Rw$$

where r=E{[d*(t)x(t)]} is the cross-correlation matrix between the desired signal and the received signal R=E[x(t)xh(t)] is the auto-correlation matrix of the received signal also known as the covariance matrix. The minimum MSE can be obtained by setting the gradient vector of the above equation with respect to equal to zero, i.e.

$$\nabla_w(E\{\varepsilon^*(t)\}) = -2r + 2Rw$$
$$= 0$$

Therefore the optimum solution for the weight w_{opt} is given by

$$w_{opt} = R^{-1}r$$

This equation is referred to as the optimum Weiner solution.

Chapter 2

TRADITIONAL ADAPTIVE BEAMFORMING APPROACHES

Side Lobe Canceller
Linearly Constrained Minimum Variance(LCMV)
Null Steering Beamforming

The following discussion explains various beamforming approaches and adaptive algorithms in a brief manner.

2.1 SIDE LOBE CANCELLERS [5]

This simple beamformer shown below consists of a main antenna and one or more auxiliary antennas. The main antenna is highly directional and is pointed in the desired signal direction. It is assumed that the main antenna receives both the desired signal and the interfering signals through its sidelobes. The auxiliary antenna primarily receives the interfering signals since it has very low gain in the direction of the desired signal. The auxiliary array weights are chosen such that they cancel the interfering signals that are present in the sidelobes of the main array response.

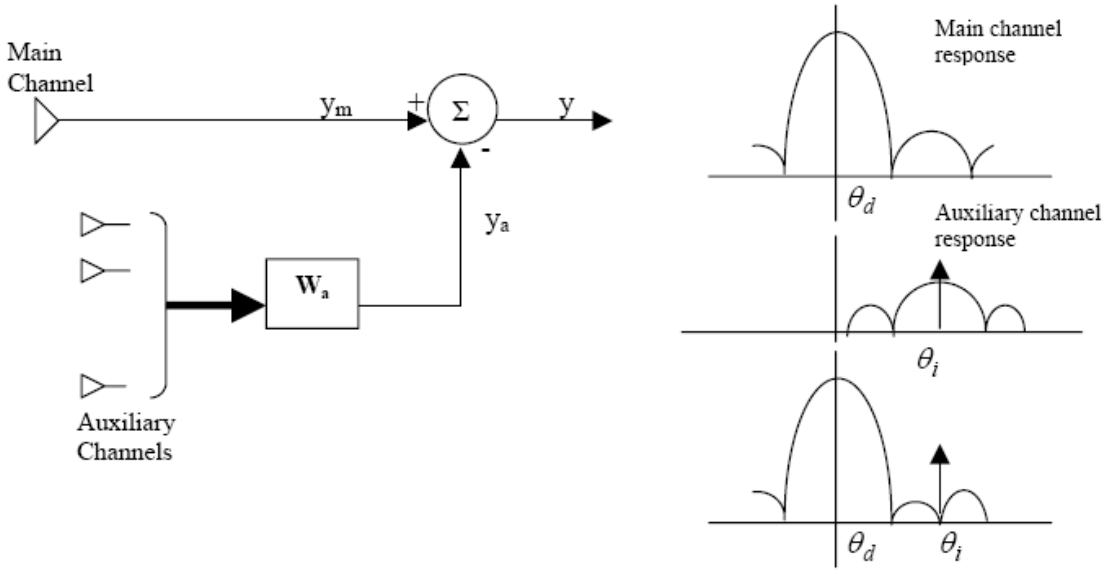

Figure2.1: Sidelobe canceller beamforming

If the responses to the interferers of both the channels are similar then the overall response of the system will be zero, which can result in white noise. Therefore the weights are chosen to trade off interference suppression for white noise gain by

minimizing the expected value of the total output power. Therefore the criteria can be expressed mathematically as follows;

$$\underbrace{\min}_{w_a} E\{\left|y_m - w_a^H x_a\right|^2$$

The optimum weights which correspond to the sidelobe canceller's adaptive component were found to be

$$w_a = R_a^{-1} r_{ma}$$

$R_a = E\{x_a x_a^H\}$ is the auxiliary array correlation matrix and the vector is the cross correlation between auxiliary array elements and the main array. This technique is simple in operation but it is mainly effective when the desired signal is weaker compared to the interfering signals since the stronger the desired signal gets (relatively), its contribution to the total output power increases and in turn increases the cancellation percentage. It can even cause the cancellation of the desired signal.

2.2 LINEARLY CONSTRAINED MINIMUM VARIANCE (LCMV)[5]

Most of the beamforming techniques discussed require some knowledge of the desired signal strength and also the reference signal. These limitations can be overcome through the application of linear constraints to the weight vector. LCMV spatial filters are beamformers that choose their weights so as to minimize the filter's output variance or power subject to constraints. This criterion together with other constraints ensures signal preservation at the location of interest while minimizing the variance effects of signals originating from other locations.

In LCMV beamforming the expected value of the array output power is minimized, i.e.

$E\{|y|^2\} = w^H R_x w$ is minimized subject to $C^H w = f$;

where Rx denotes the covariance matrix of x(t), C is the constraint matrix which contains K column vectors and is the response vector which contains K

scalar constraint values. The solution to the above equation using Lagrange multipliers gives the optimum weights as

$$w_{opt} = R_x^{-1} C (C^H R_x^{-1} C)^{-1} f$$

This beam forming method is flexible and does not require reference signals to compute optimum weights but it requires computation of a constrained weight vector. C

2.3 NULL STEERING BEAMFORMING [5]

Unlike other algorithms null steering algorithms do not look for the signal presence and then enhance it, instead they examine where nulls are located or the desired signal is not present and minimize the output signal power. One technique based on this approach is to minimize the mean squared value of the array output while constraining the norm of the weight vector to be unity.

$$\min_w w^H R w \quad \text{subject to } w^H A w = 1$$

The matrix A, a positive-definite symmetric matrix, serves to balance the relative importance of portions of the weight vectors over others. The optimum weight vector must satisfy the following equation;

$$Rw = -\lambda A w$$

2.4 SAMPLE MATRIX INVERSION (SMI) ALGORITHM: [5]

In this algorithm the weights are chosen such that the mean-square error between the beamformer output and the reference signal is minimized. The mean square error is given by

$$E\left[\{r(t) - w^H x(t)\}^2\right] = E[r^2(t)] - 2w^H R_r + w^H R_m w$$

x(t) is the array output at time t; r(t) is the reference signal; is the signal covariance matrix. Rr=E[r(t)x(t)] defines the covariance between the reference signal and

the data signal. The weight vector, for which the above equation becomes minimum, it is obtained by setting its gradient vector with respect to, to zero, i.e.

$$\nabla_w \{E\left[\{r(t) - w^H x(t)\}^2\right]\} = -2R_r + 2R_m w = 0$$

Therefore,

$$w_{opt} = {R_m}^{-1} R_r$$

The optimum weights can be easily obtained by direct inversion of the covariance matrix. This algorithm requires a reference signal and is computational intensive. It is definitely faster than LMS.

Chapter 3

NON BLIND ADAPTIVE BEAMFORMING ALGORITHM

Introduction
Non Blind Adaptive Beamforming Algorithms
Sample Matrix Inversion (SMI)
Least Mean Square (LMS)
Recursive Least Square (RLS)

3.1 Introduction [1] [8]

The preceding chapter, an overview of beamforming was studied in terms of the Physical components needed to perform such a task. While at this point that topic is well Understood, it is still not known how to determine the weights necessary for beamforming. In the following discussion, it is desired to study means in which specific characteristics of the received signal incident upon the array (in addition to the spatial separation among users in the environment) can be exploited to steer beams in directions of desired users and nulls in directions of interferers. In particular, the Mean Square Error (MSE) criterion of a particular weight vector will be minimized through the use of statistical expectations, time averages and instantaneous estimates. As well, the distorted constant modulus of the array output envelope due to noise in the environment will be restored. Finally, knowledge of the spreading sequences of a CDMA mobile environment will be utilized to improve the performance of algorithms exploiting the two criterions discussed above. Each of the characteristics described above correspond to adaptive algorithms which can be classified into two categories: 1.) Non-Blind Adaptive algorithms & 2.) Blind Adaptive Algorithms. Non-blind adaptive algorithms need statistical knowledge of the transmitted signal in order to converge to a weight solution. This is typically accomplished through the use of a pilot training sequence sent over the channel to the receiver to help identify the desired user. On the other hand, blind adaptive algorithms do not need any training, hence the term “blind”. They attempt to restore some type of characteristic of the transmitted signal in order to separate it from other users in the surrounding environment. Note on Notation: For the discussion to follow which includes, scalars, vectors and matrices, the following notation will be followed: vector – lower case letter with arrow (Ex. x□□), scalar – lower case letter (Ex.. d) Matrix – Capitalized letter (Ex. R)

3.2 Non-Blind Adaptive Beamforming Algorithms [1] [2] [4] [8]

As was noted above, non-blind adaptive algorithms require a training sequence, *d(k)* in order to extract a desired user from the surrounding environment. This in itself is undesirable for the reason that during the transmission of the training sequence, no communication in the channel can take place. This dramatically reduces the spectral efficiency of any communications system. Additionally, it can be very difficult to understand the statistics of the channel in order to characterize a reasonable estimate of *d(k)* needed to accurately adapt to a desired user. With this in mind, the following summarizes the basic concepts of non-blind adaptive algorithms.

WEINER OPTIMUM SOLUTION

Consider the *least mean square* (LMS) adaptive array shown below in Figure

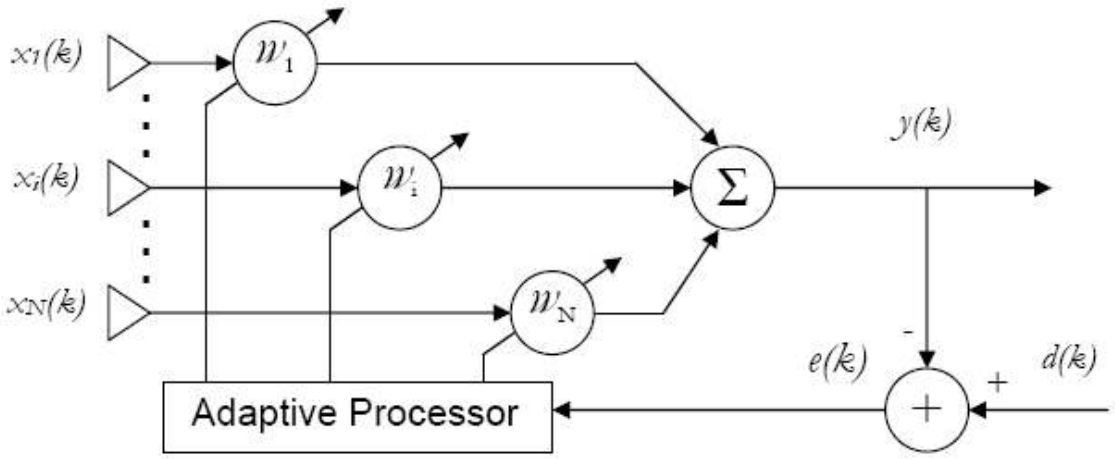

Figure3.1: LMS Adaptive Array

Through a feedback loop the weights, *w1* , ... , *wN* , are updated by the time sampled error signal: $e(k) = d(k) - y(k)$ **(4.1)** *where:* The training sequence, *d(k)*, is a near replica of the desired signal and *y(k)* is the output of the adaptive array described by equation 3.40. The feedback system attempts to direct the weights at each element to their optimal weights, *opt w*. The adaptive processor adjusts the weight vector to minimize the *mean square error* (MSE) of the error signal, *e(k)*, given by:

$$\left[{}_{2} \right] \left[{}_{2} \right] E\, e(k) = E\, d(k) - y(k)$$

where: E is the expectation operator. Substituting equation 3.41 into equation 4.2 and expanding the argument of the MSE term

$$E\left[\left|e(k)\right|^2\right] = E\left[\left\{d(k)-y(k)\right\}\left\{d(k)-y(k)\right\}^*\right]$$

$$= E\left[\left\{d(k)-\bar{w}^H\bar{x}(k)\right\}\left\{d(k)-\bar{w}^H\bar{x}(k)\right\}^*\right]$$

$$= E\left[\left|d(k)\right|^2 - d(k)\bar{x}^H(k)\bar{w} - \bar{w}^H\bar{x}(k)d^*(k) + \bar{w}^H\bar{x}(k)\bar{x}^H(k)\bar{w}\right]$$

$$= E\left[\left|d(k)\right|^2 - \bar{r}_{xd}{}^H\bar{w} - \bar{w}^H\bar{r}_{xd} + \bar{w}^H R_{xx}\bar{w}\right]$$

- $E\left[\bar{x}(k)\bar{x}^H(k)\right] = R_{xx}$ is the *NxN Covariance Matrix* of the input data vector, $\bar{x}(k)$
- $E\left[\bar{x}(k)d^*(k)\right] = \bar{r}_{xd}$ is the *Nx1 Cross-Correlation Vector* between the input data vector, $\bar{x}(k)$, and the training sequence, $d(k)$.

It is apparent from either equation 4.3 or 4.4 that the MSE of the LMS adaptive array is a quadratic function in *w* where the extremer of this quadratic surface is a unique minimum. By plotting the MSE vs. the weights for equation 4.3, we achieve the following quadratic surface, which is also called the *performance surface*.

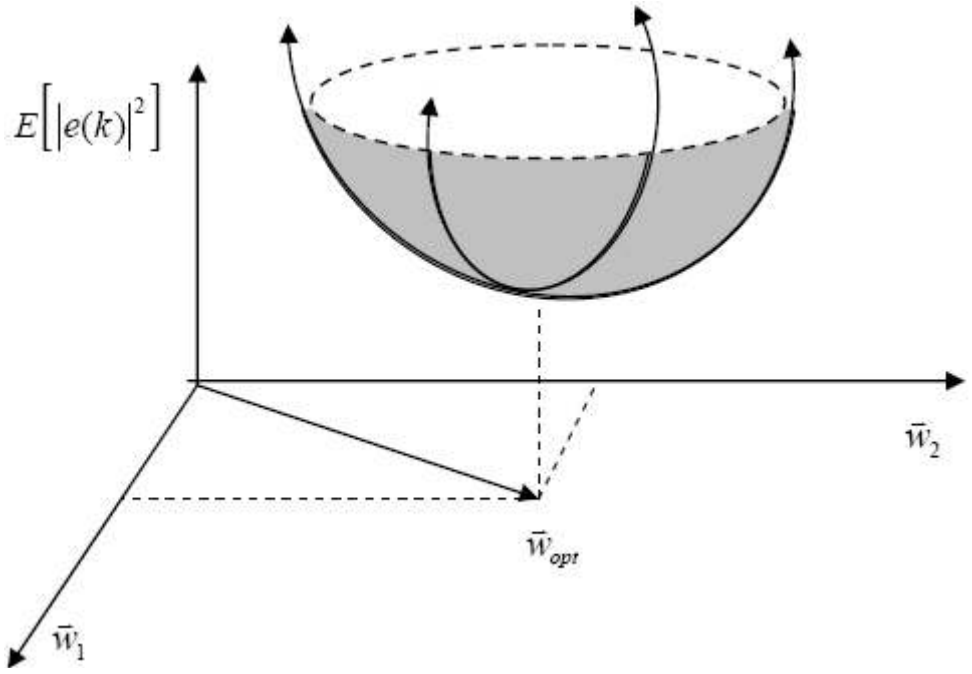

Figure3.2: Quadrative Surface for MSE Criterion of Adaptive LMS Array

The shape, location, and orientation of the performance surface depicted above depend on the array geometry and the signals incident upon the array. If the incident signals, their AOA's and power are time variant, then the performance surface will move around in space thus altering the value of *opt w* . It is the job of the adaptive array to force the optimum weight vector to track to the bottom of the surface. The unique minimum of the performance surface can be found by performing the vector gradient operator, □ ( ), of the mean square error defined in equation 4.4 with respect to the array weights and setting the result equal to zero.

$$\nabla\left(E\left[\left|e(k)\right|^2\right]\right)=\frac{\partial}{\partial \bar{w}^*}E\left[\left|e(k)\right|^2\right]=0$$

which yields:

$$-2\bar{r}_{xd}+2R_{xx}\bar{w}_{opt}=0$$

Solving for the optimum weights gives:

$$\boxed{\bar{w}_{opt}=R_{xx}^{-1}\bar{r}_{xd}}$$

There exist some disadvantages in using the Weiner Solution to determine the optimum weight vector. In particular, if the number of elements in the ULA is large, then it is computationally complex to invert the *NxN* covariance matrix, *Rxx*. This is a problem only if the inverse of the covariance matrix is nonsingular, which is not always the case. It first would have to be assumed that the matrix is positive semi-definite to begin with in order to use the Weiner solution. Additionally, the Weiner solution requires the use of an expectation operator in both *Rxx* and *xd r*□ . This assumes that the statistics of the communications channel and the desired signal estimate, *d(k)*, can be perfectly characterized to produce an adequate training sequence.

3.3 SAMPLE MATRIX INVERSION (SMI)

In practice, the mobile channel environment is constantly changing making estimation of the desired signal quite difficult. These frequent changes will require a continuous update of the weight vector, which would be difficult to produce for reasons already stated. However, Reed, Mallet, and Brennan [31] proposed an estimate to the Weiner solution through the use of time averages called *Sample Matrix Inversion* (SMI). Suppose we take *K* time samples of the received signal to form an *input data matrix*, *X*, defined by:

$$X = \left[\vec{x}(1), \quad \vec{x}(2), \quad \cdots \quad , \vec{x}(K)\right]$$

where: $\vec{x}(1) = s(1)\vec{a}(\theta) + \vec{n}(1)$ and so on, for the input data model defined in equati

An estimate of the *NxN* covariance matrix, $\hat{R}_{xx}$, can then be formed by tal average over *K* samples given by:

$$\hat{R}_{xx} = \frac{1}{K}\sum_{k=1}^{K} \vec{x}^{*}(k)\vec{x}^{T}(k)$$

which can be expressed in matrix form as:

$$\hat{R}_{xx} = \frac{1}{K} X^{*} X^{T}$$

Likewise, we can express the estimate of the *Nx1* cross-correlation vector, $\hat{r}_{xd}$, as:

$$\hat{r}_{xd} = \frac{1}{K}\sum_{k=1}^{K} \left[d(k)X^{T}(k)\right]^{*}$$

which is expressed in vector form as:

$$\hat{\vec{r}}_{xd} = \frac{1}{K}\left(X\vec{d}^{\,T}\right)^{*}$$

where: $\vec{d}$ is the desired signal vector given by:

$$\vec{d} = \left[d(1), \quad d(2), \quad \cdots \quad , d(K)\right]$$

then the estimated optimum weight vector, $\hat{\vec{w}}$, is given by:

$$\hat{\vec{w}} = \hat{R}_{xx}^{-1}\hat{\vec{r}}_{xd}$$

For a rapidly changing environment, it is possible to estimate blocks of dat one to repeat the process periodically. We can alter the input data matrix, *X*, to ref dynamic block size of *K* samples:

$$X(l) = \left[\vec{x}(1+lK), \quad \vec{x}(2+lK), \quad \cdots \quad , \vec{x}(K+lK)\right] \quad for\ l = 0, 1, \cdots, L$$

The desired signal vector can be altered to reflect this dynamic block size as well.

where: L is the number of iterations required for the algorithm to converge.

Typically it is a rule of thumb to allow the block size, $K > 2N$. This means the number of samples must be greater than or equal to twice the number of elements in the adaptive array. For a further discussion on why this is true, the reader is referred to for the dynamic block size SMI method; the MSE for each element can be determined by:

$$e = \hat{R}_{xx}\vec{w} - \hat{r}_{xd}$$

below depicts the MSE of the 3rd element for the dynamic SMI method with a block size of 10 where the received signal consists of one desired user whose signal is polar NRZ and one interfering multipath component.

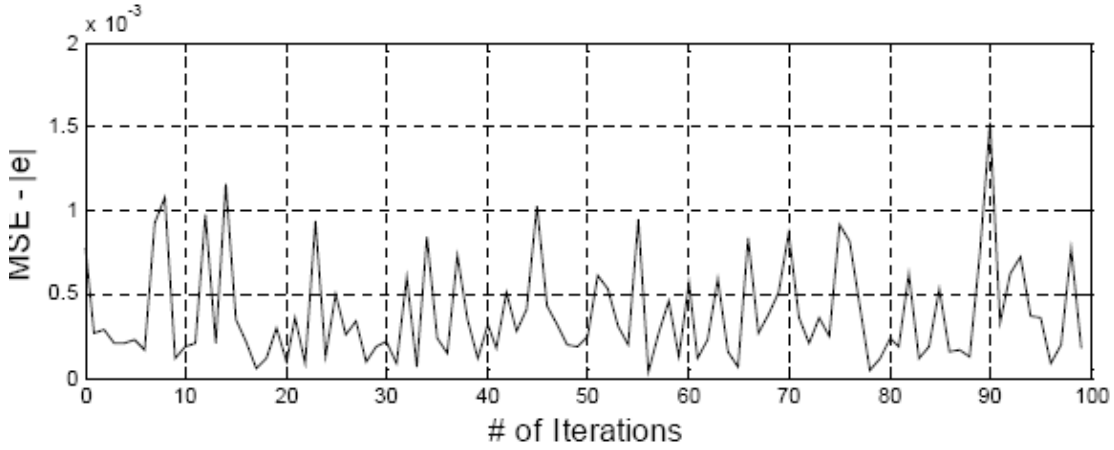

Figure 3.3: MSE of Dynamic SMI Method w/block size of 10

From the above results, we can see that the error for each iteration is very small. The stability of the SMI method depends on the ability to invert the *NxN* estimate of the covariance matrix given in equation 4.11. Typically, noise is added to the system to offset the diagonal elements of the input data vector in order to avoid singularities when inverting the covariance matrix. These singularities are caused by the number of received signals to be resolved being less than the number of elements in the array. The SMI method is a particularly desirable algorithm to determine the complex weight vector due to the fact that the convergence rate is usually greater than a typical LMS adaptive array and is independent of signal powers, AOA's and other parameters. The number of multiplications needed to form the estimated covariance matrix is proportional to *N 3*. Also, the number of linear equations needed to solve equation 4.16 increases as *N 3*.

Therefore, the SMI method operates at its best when the number of elements in the adaptive ray is small. Figure 4.4 below depicts the beampattern for an 8-element ULA where the weights ere determined using the SMI method. We assume a multipath scenario where the received signal is a polar NRZ waveform whose values appear with equal probability. The desired user's amplitude was five times greater than that of the multipath component. The desired user's AOA was -45o and the interferer's AOA were 30.

3.4 LEAST MEAN SQUARES [1] [2] [4] [6]

A very computationally efficient adaptive algorithm is the *least mean squares* (LMS) algorithm. LMS is an iterative solution to solve for the weights which track to the bottom of the performance surface. In the case of LMS, the statistics of the channel and incident signal **b**eampattern for 8-Element ULA using SMI Method 22 are not known. The LMS algorithm estimates the gradient of the error signal, *e(k)*, by employing the *method of steepest descent*, which is summarized below. Let *k w* represent the *Nx1* weight vector at time sample *k*. The weight vector can be updated at time sample *k+1* by offsetting) (*k w* by some small quantity which drives the weight vector one step closer to the bottom of the performance surface. This small quantities the value of the error gradient for time sample *k*, which is given as:

$$\bar{w}(k+1) = \bar{w}(k) + \mu\left[-\nabla(J(k))\right]$$

where: $J(k) = E\left[\left|e(k)\right|^2\right]$ defines the MSE cost function.

Referring to equation 4.8, the value of the error signal at time sample *k* is given by:

$$\nabla(J(k)) = -2\bar{r}_{xd} + 2R_{xx}\bar{w}(k)$$

Substituting equation 4.20 into equation 4.19 results in:

$$\bar{w}(k+1) = \bar{w}(k) + 2\mu\left[r_{xd} - R_{xx}\bar{w}(k)\right]$$

where: μ is an incremental correction factor known as the *step-size* parameter or *weight convergence* parameter. It is a real valued positive constant generally less than one. Substituting equations 4.5 & 4.6 into the above equation yields:

$$\bar{w}(k+1) = \bar{w}(k) + 2\mu E\left[\bar{x}(k)d^*(k) - \bar{x}(k)\bar{x}^H(k)\bar{w}(k)\right]$$

$$= \bar{w}(k) + 2\mu E\left[\bar{x}(k)\{d(k) - y(k)\}^*\right]$$

$$\boxed{\bar{w}(k+1) = \bar{w}(k) + 2\mu E\left[\bar{x}(k)e^*(k)\right]}$$

When a proper step size parameter is chosen and enough iteration is performed then the above result will converge to the optimum weight vector given in equation 4.9. However, as was previously stated obtaining the statistics of the channel and developing an adequate estimate of the received signal are quite difficult. In this case, we can use the LMS algorithm to form an instantaneous estimate of the error signal. Dropping the expectation operator in equation 4.22 allows the algorithm to update the weights as the data are sampled, which is given by:

$$\boxed{\bar{w}(k+1) = \bar{w}(k) + 2\mu \bar{x}(k)e^*(k)}$$

The LMS algorithm is a very desirable algorithm in many circumstances. One of its great weaknesses is its slow convergence rate. The algorithm updates the weight vector for every incoming sample of the received signal vector, $x\square(k)$ which is offset by the step size parameter, μ. If μ is large, then the algorithm will converge faster, but your resulting weight vector will be less accurate, vice versa for when it is small. Below is a plot of several convergence curves for the LMS algorithm given different step sizes.

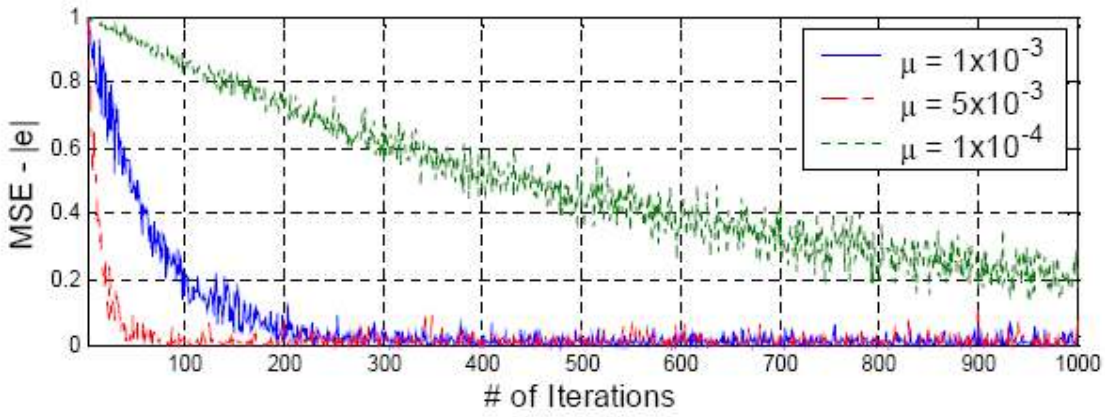

Figure 3.4: LMS Algorithm Convergence Curves for different Step Sizes

In the above plot, an identical scenario used for displaying the results of the SMI method in Figure 3.6 was assumed, which was a multipath scenario where the received signal is a polar NRZ waveform. The desired user's amplitude was five times greater than the multipath component. The desired user's AOA was -45o and the interferer's AOA was 30o. *Additive White Gaussian Noise* (AWGN) with a signal to noise ratio of 10 *dB* was assumed, 23 Figures 4.5 & 4.6 created by program LMS.

66 which is quite high. It is apparent that if the step size parameter is decreased, then a slower convergence is achieved. Also note the maladjustment at each iteration. This is somewhat due to the noise present in the channel, but mostly caused by the gradient search method used to track the weights to the bottom of the performance surface. Below is a plot of the resulting beam pattern for this particular scenario. In the backdrop is a shadow of the beampattern using the SMI method, which is used as a benchmark. It is clear to see that the two beampatterns are nearly identical. Also, it can be seen from the asterisk marking the interferer arriving at 30o that the SMI method provides a deeper null than that using the LMS algorithm.

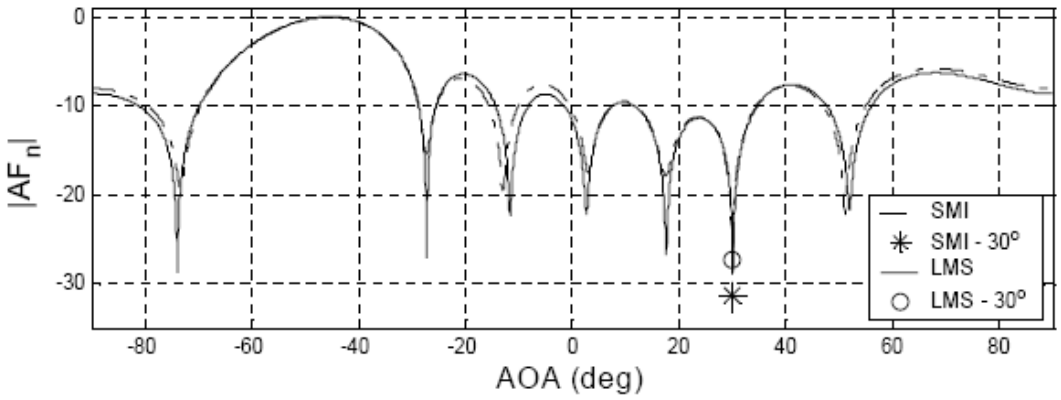

Figure 3.5: Beampattern for 8-element ULA using SMI

3.5 RECURSIVE LEAST SQUARES [4]

Contrary to the LMS algorithm, which uses the steepest descent method to determine the complex weight vector, the *Recursive Least Squares* (RLS) algorithm uses the *method of least squares*. The weight vector is updated by minimizing an exponentially weighted cost function consisting of two terms: 1.) sum of weighted error squares and 2.) a regularization term. Together the cost function is given by:

$$\varepsilon(k) = \underbrace{\sum_{i=1}^{k} \lambda^{k-i} \left| e(i) \right|^2}_{\text{Sum of Weighted Error Squares}} + \underbrace{\delta \lambda^k \left\| \bar{w}(k) \right\|^2}_{\text{Regularization term}}$$

where: e(i) is the error function defined by equation 4.1 and λ is called the *forgetting factor*, which is a positive constant close to, but less than one. It emphasizes past data in a Non-stationary environment so that the statistical variations of the data can be tracked Sum of Weighted Error Squares Regularization term 67 and not "forgotten". In a stationary environment, $\lambda = 1$ corresponds to infinite memory. Expanding equation 4.26 and collecting terms, the weight summation of the covariance matrix for the received signal, $x\square(k)$, can be determined by:

$$\Phi(k) = \sum_{i=1}^{k} \lambda^{k-i} \bar{x}(k) \bar{x}^H(k) + \delta \lambda^n I$$

Performing the matrix inversion lemma to the result in equation 4.27, we can create a

recursive equation to solve for the complex weight vector. A thorough description of the derivation is described in [19]. A summary of the RLS algorithm is provided below:

$$\vec{k}(k) = \frac{\lambda^{-1} P(k-1)\vec{x}(k)}{1 + \lambda^{-1}\vec{x}^{H}(k)P(k-1)\vec{x}(k)}$$

$$e(k) = d(k) - \vec{w}^{H}(k-1)\vec{x}(k)$$

$$\vec{w}(k) = \vec{w}(k-1) + \vec{k}(k)e^{*}(k)$$

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-1}\vec{k}(k)\vec{x}^{H}(k)P(k-1)$$

At time sample $k = 1$, the initial conditions for the RLS algorithm can be described by:

1.) Set $\vec{w}(0)$

to either a column vector of all zeros, or to the first column vector of an NxN identity matrix.

2.) Setting $k = 0$ in equation 4.27 yields: $\Phi(0) = \delta I = P(0)$,

where: δ=small positive constant for high SNR

large positive constant for low SNR

RLS is a desirable algorithm because it has the ability to retain information about the input data vector, $x(k)$, since the moment the algorithm was started. Therefore, the convergence of the RLS algorithm is much greater than that of the LMS algorithm by nearly an order of magnitude, but at the cost of increased computational complexity. An important feature of the RLS algorithm is its ability to replace the inversion of the covariance matrix in the Weiner solution with a simple scalar division. Figure 3.6 below depicts the convergence curve for the RLS algorithm with a forgetting factor of 1 for the multipath scenario encountered previously

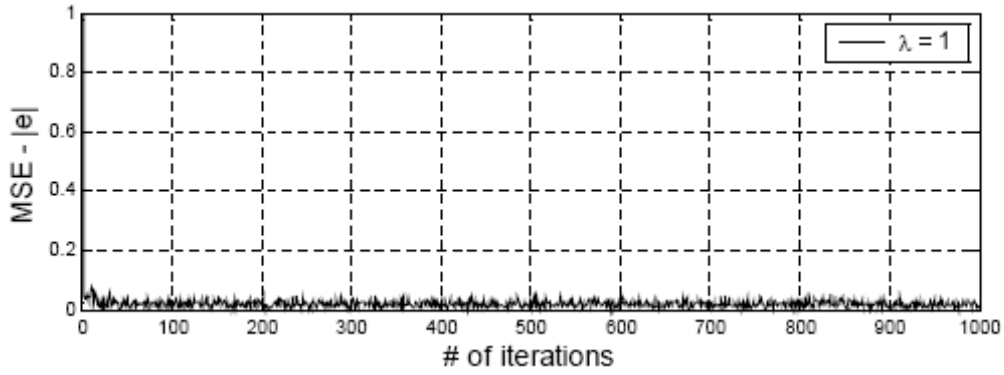

Figure 3.6: Convergence Curve for RLS Algorithm w/ λ = 1.

Additionally, Figure 3.7 below plots the beampattern for an $N = 8$ element ULA for the same senario.

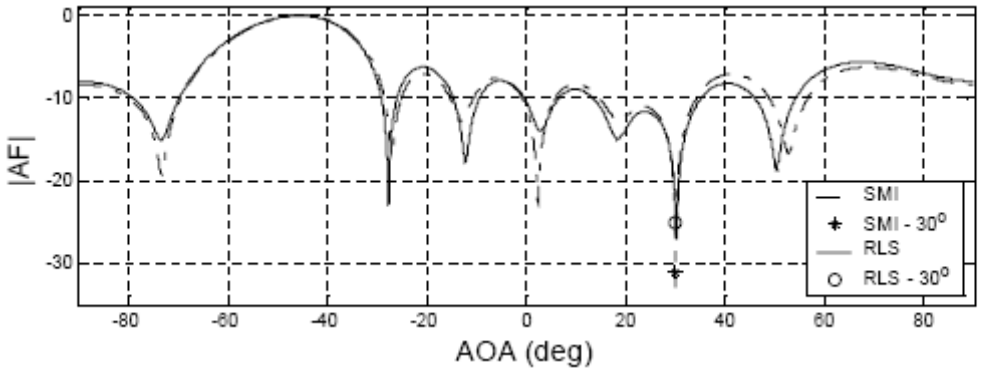

Figure 3.7: Beam pattern for RLS algorithm

Chapter 4

BLIND ADAPTIVE BEAMFORMING ALGORITHMS

Introduction
Constant Modulus Algorithm (CMA)
Steepest Descent Decision Directed Algorithm (SD-DD)
Least Square Constant Modulus Algorithm (LS-CMA)
Recursive Least Squares Constant Modulus Algorithm (RLS-CMA)

4.1 INTRODUCTION

As was stated previously, “blind” adaptive algorithms do not need a training sequence in order to determine the required complex weight vector. They attempt to restore some type of property to the received signal for estimation. A common property between polar NRZ waveforms and DS-SS signals is the constant modulus of received signals. Therefore, this study focuses on blind adaptive algorithms which exploit this characteristic.

4.2 CONSTANT MODULUS ALGORITHM (CMA) [1] [3] [4] [6] [7] [8

A typical polar NRZ signal possesses an envelope which is constant, on average. During transmission, corruption from the channel, multipath, MAI, and noise can distort 24 Figures 4.7 & 4.8 created by program RLS.m this envelope. Using the constant modulus algorithm (CMA), the envelope of the adaptive array output, *y(k)*, can be restored to a constant by measuring the variation in the signal’s modulus and minimizing it by using the cost function defined below:

The constant modulus cost function is a positive definite measure of how much the array output’s envelope varies from the unity modulus used to minimize the result. Setting p= 1, q = 2, we can develop a recursive update method to determine the proper weights by utilizing the method of steepest descent for the following cost function:

$$J(k) = E\left[\left|\left|y(k)\right| - 1\right|^2\right]$$

Taking the gradient of the above cost function yields:

$$\nabla(J(k)) = E\left[\left(\left|y(k)\right| - 1\right)\frac{\partial\left|y(k)\right|}{\partial \bar{w}^*(k)}\right]$$

$$= E\left[\left(\left|y(k)\right| - 1\right)\frac{\partial\left(y(k)y^*(k)\right)^{1/2}}{\partial \bar{w}^*(k)}\right]$$

$$= E\left[\left(\left|y(k)\right| - 1\right)\frac{\partial\left(\bar{w}^H(k)\bar{x}(k)\bar{x}^H(k)\bar{w}(k)\right)^{1/2}}{\partial \bar{w}^*(k)}\right]$$

$$= \frac{1}{2}E\left[\left(\left|y(k)\right| - 1\right)\left(\bar{w}^H(k)\bar{x}(k)\bar{x}^H(k)\bar{w}(k)\right)^{-1/2}\frac{\partial\left(\bar{w}^H(k)\bar{x}(k)\bar{x}^H(k)\bar{w}(k)\right)}{\partial \bar{w}^*(k)}\right]$$

$$= \frac{1}{2}E\left[\left(\left|y(k)\right| - 1\right)\left(\frac{1}{\left|y(k)\right|}\bar{x}(k)\bar{x}^H(k)\bar{w}(k)\right)\right]$$

$$= \frac{1}{2}E\left[\left(1 - \frac{1}{\left|y(k)\right|}\right)\bar{x}(k)y^*(k)\right]$$

$$= \frac{1}{2}E\left[\bar{x}(k)\underbrace{\left(y(k) - \frac{y(k)}{\left|y(k)\right|}\right)^*}_{e(k)}\right]$$

$$J(k) = E\left[\left|\left|y(k)\right|^p - 1\right|^q\right] \quad , p = 1, 2 \text{ or } q = 1, 2$$

The p = 1, q = 2 solution is typical because it provides the deepest nulls of the four configurations and provides the best *signal to interference noise ratio* (SINR). Figure 4.9 below depicts the convergence curve for the constant modulus algorithm with p =1, q = 2.

Also, shows the beampattern for an $N = 8$ element ULA for the multipath scenario discussed previously.

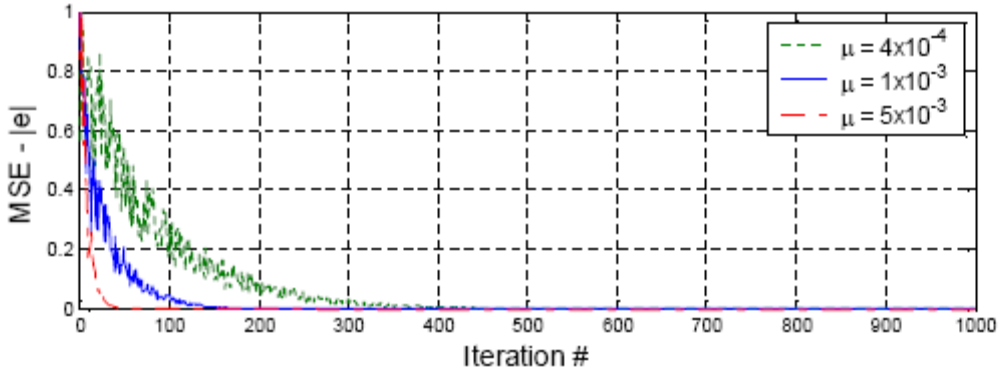

Figure 4.1: Convergence Curves for CMA w/ $p = 1$, $q = 2$.

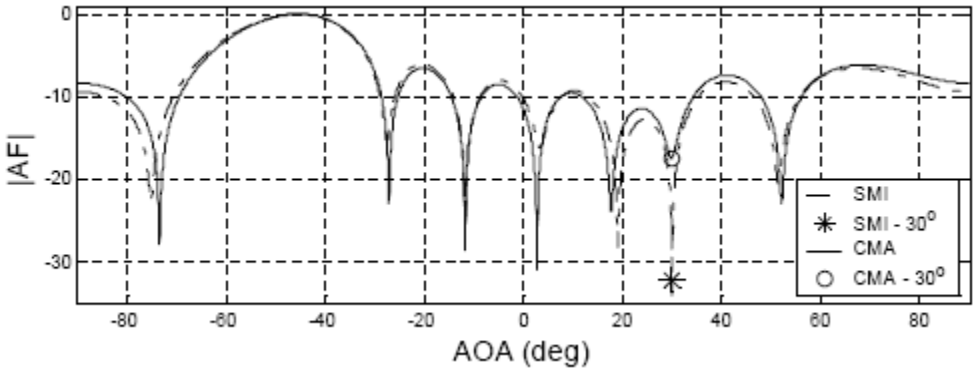

Figure 4.2: Beampattern for CMA

It is obvious from the above results that by decreasing the value of the step size parameter, a faster convergence is achieved. Likewise, due to the high correlation between the transmitted signal and its multipath component, the null formed at 30o is very shallow compared to that formed by the SMI method.

4.3 STEEPEST DESCENT DECISION DIRECTED ALGORITHM (SD-DD) [1] [3] [7]

For BPSK communications scenarios with low *bit error rates* (BER), the complex limited constant modulus reference signal, $\frac{y(k)}{|y(k)|}$, can be replaced by a decision directed term defined as: $\text{sgn}(\text{Re}(y(k))$. This decision directed form uses the demodulated signal to reference the variation in the modulus of the array output. The SD-DD algorithm is summarized as follows:

$$y(k) = \vec{w}^H(k)\vec{x}(k)$$

$$e(k) = y(k) - \text{sgn}(\text{Re}(y(k)) \qquad (4.42)$$

$$\vec{w}(k+1) = \vec{w}(k) - \frac{1}{2}\mu\vec{x}(k)e^*(k)$$

4.4 LEAST SQUARES CONSTANT MODULUS ALGORITHM (LS-CMA)

As was studied previously, the constant modulus algorithm utilizes the method of steepest descent to adapt the weight vector. Additionally, Agee proposed in [1] an algorithm based upon the method of nonlinear least squares, also known as Gauss' Method which states that if a cost function can be expressed in the form:

$$F(\vec{w}) = \sum_{k=1}^{K} |g_k(\vec{w})|^2 = \|\bar{g}(\vec{w})\|_2^2$$

where: $\bar{g}(\vec{w}) = \begin{bmatrix} g_1(\vec{w}) & g_2(\vec{w}) & \cdots & g_K(\vec{w}) \end{bmatrix}^T$

Then, the cost function has a Taylor series expansion with the sum of squares form:

$$F(\vec{w} + \vec{d}) \approx \left\|\bar{g}(\vec{w}) + D^H(\vec{w})\vec{d}\right\|_2^2$$

where: $\vec{d}$ is an offset vector and $D(\vec{w})$ is defined as:

$$\left[\nabla(g_1(\vec{w}))\quad \nabla(g_2(\vec{w}))\quad \cdots\quad \nabla(g_K(\vec{w}))\right] \tag{4.46}$$

Then, taking the gradient of equation 4.43 with respect to $\vec{d}$ and setting the result equal to zero yields an offset which minimizes the cost function defined in equation 4.45:

$$\vec{d} = -\left[D(\vec{w})D^H(\vec{w})\right]^{-1} D(\vec{w})\vec{g}(\vec{w}) \tag{4.47}$$

The weight vector can then be updated by adding this offset to the current weight vector:

$$\vec{w}(k+1) = \vec{w}(k) - \left[D(\vec{w}(l))D^H(\vec{w}(l))\right]^{-1} D(\vec{w}(l))\vec{g}(\vec{w}(l)) \tag{4.48}$$

Extending the result to include the constant modulus cost function results in:

$$F(\vec{w}) = \sum_{k=1}^{K}\left\|y(k)\right|-1\Big|^2 = \sum_{k=1}^{K}\left|\vec{w}^H\vec{x}(k)-1\right|^2 \tag{4.49}$$

and,

$$\vec{w}(l+1) = \vec{w}(l) - \left[XX^H\right]^{-1} X\left(y(l)-r(l)\right)^*$$

where: X is the input data matrix defined by: $X = \left[x(1)\quad x(2)\quad \cdots\quad x(K)\right]$

and $\vec{x}(1) = s(1)\vec{a}(\theta) + \vec{n}(1)$ and so on, for the input data model defined in equation 3.34. Also, $\vec{r}(l) = \left[\frac{y(1)}{|y(1)|}\quad \frac{y(2)}{|y(2)|}\quad \cdots\quad \frac{y(K)}{|y(K)|}\right]^T$ for $y(l) = \left[\vec{w}^H(l)X\right]^T$ (4.50)

defines the complex limited array output.

Substituting equation 4.50 into equation 4.49 and collecting terms yields:

$$\boxed{\vec{w}(l+1) = \left[XX^H\right]^{-1} X r^*(l)} \tag{4.51}$$

Extensive detail on the derivation can be found in [3], [11], [20].

The result in equation 4.51 is considered to be the *static Least Squares CMA* algorithm since it updates the weight vector for a single block of data samples with length K. The result can be extended to include dynamic blocks of data where the input data matrix is altered to become:

$$X(l) = \left[\vec{x}(1+lK),\quad \vec{x}(2+lK),\quad \cdots\quad ,\vec{x}(K+lK)\right] \quad \textit{for } l = 0, 1, \cdots, L$$

and,

$$y(l) = \left[\vec{w}^H(l)X(l)\right]^T = \left[y(1+lK),\quad y(2+lK),\quad \cdots\quad ,y(K+lK)\right]$$

The resulting weight update equation is given by:

$$\vec{w}(l+1) = \left[X(l)X^H(l)\right]^{-1} X(l)\, r^*(l) \qquad \textbf{(4.5}$$

Let,

$$\hat{R}_{xx}(l) = \frac{1}{K} X(l)X^H(l) \qquad \textbf{(4.5}$$

and

$$\hat{\vec{r}}_{xr}(l) = \frac{1}{K} X(l)r^*(l) \qquad \textbf{(4.5}$$

then the updated weight vector can be related in terms of time averages for an estimated covariance matrix, $\hat{R}_{xx}(l)$ and estimated cross-correlation vector $\hat{\vec{r}}_{xr}(l)$ given by:

$$\vec{w}(l+1) = \hat{R}_{xx}(l)^{-1}\hat{\vec{r}}_{xr}(l) \qquad \textbf{(4.5}$$

In contrast to the steepest descent (SD) CMA algorithm, the LS-CMA algorithm updates the weight vector on a block by block basis instead of each sample. This provides a dramatically faster convergence rate as compared to SD-CMA. Figure 4.11 depicts convergence curves of the dynamic LS-CMA algorithm for different block sizes.

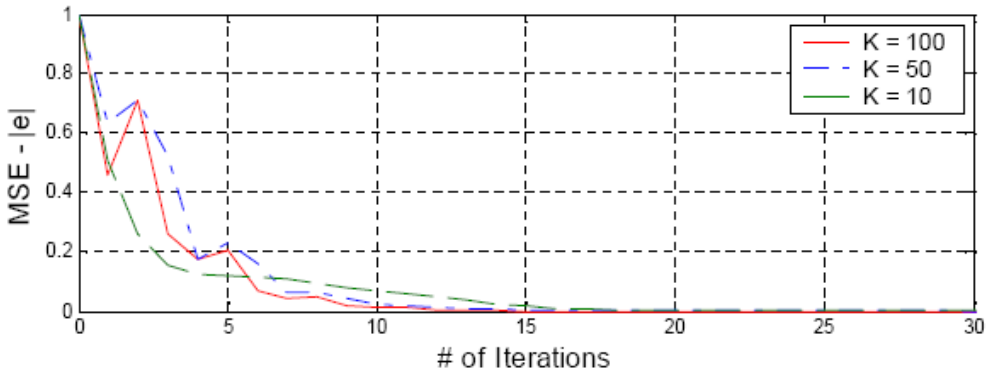

Figure 4.11: Convergence Curves for Dynamic LSCMA w/ K = 100, 50, 10. [26]

In the figure above, a multipath scenario was assumed where the two signals were both sinusoids with the desired user having an amplitude five times greater than that of the multipath component. Additionally, the multipath component was a Doppler and phase shifted version of the desired signal. AWGN of 10 *dB* was assumed added to the input dat matrix. It can be seen that if the block size is increased, then the algorithm converges faste but at the cost of increased computational complexity because the size of the estimated

covariance matrix increases. The resulting beampattern for this scenario is displayed in Figure 4.3 below

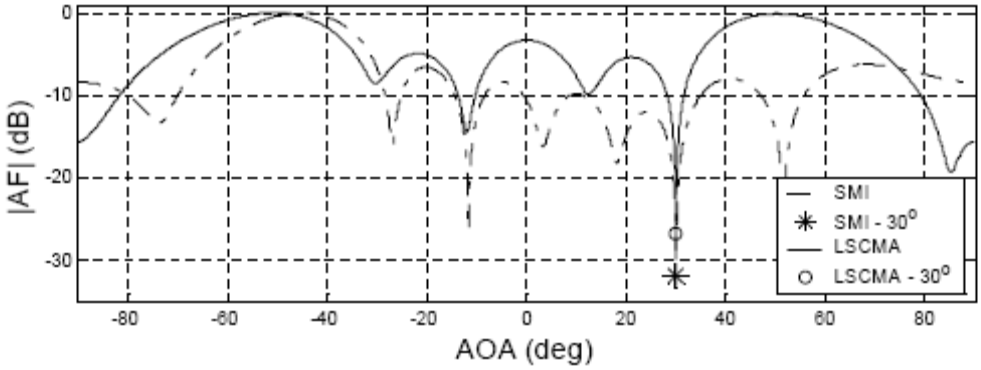

Figure 4.3: Beampattern for LSCMA

It is apparent to note that again the null formed by the SMI method is deeper than that formed by the LS-CMA algorithm. Additionally, the beampattern for the LS-CMA algorithm is nearly symmetric. This is due to the fact that when computing the estimate of the covariance matrix, much of the phase information needed is eliminated when multiplying the two input data matrices. The required maxima and minima are produced, but false maxima and minima are also formed at both 45o and -30o due to this phase ambiguity. Also, we can see that the main lobe directed to the desired user does not peak at exactly -45.

4.5 RECURSIVE LEAST SQUARES CONSTANT MODULUS ALGORITHM (RLS-CMA) [8]

The newly developed Recursive Least Squares Constant Modulus (RLS-CMA) Algorithm proposed in [8] combines adaptive beamforming using the constant modulus criterion via the RLS iterative update solution. This particular algorithm possesses the convergence properties of the RLS algorithm and the tracking capabilities of the CMA define the dynamic complex limited array output. algorithm. Together they form an algorithm which is capable of restoring the modulus of the array output. It is clear from both Figures 4.7 & 4.9 the RLS optimization technique provides a faster convergence rate than that of the CMA algorithm. However, the constant modulus cost function described

by equation 4.32 is non quadratic in the array weights, therefore making application of the RLS algorithm impossible. The RLS-CMA algorithm attempts

to modify the constant modulus cost function to allow use of the RLS algorithm for the special case where $q = 2$. Replacing the expectation operator in equation 4.32 with the exponential sum of weighted error squares yields a modified cost function for $q = 2$ given by

$$J(k) = \sum_{i}^{k} \lambda^{k-i} \left(\left| \vec{w}(k) \vec{x}(i) \right|^{p} - 1 \right)^{2}$$

The RLS-CMA algorithm is summarized as follows [8]:

$$\vec{z}(k) = \vec{x}(k)\vec{x}^{H}(k)\vec{w}(k-1)\left|\vec{x}^{H}(k)\vec{w}(k-1)\right|^{p-2}$$

$$\vec{k}(k) = \frac{\lambda^{-1}P(k-1)\vec{z}(k)}{1+\lambda^{-1}\vec{z}^{H}(k)P(k-1)\vec{z}(k)}$$

$$e(k) = \vec{w}^{H}(k-1)\vec{z}(k) - 1$$

$$\vec{w}(k) = \vec{w}(k-1) + \vec{k}(k)e^{*}(k)$$

$$P(k) = \lambda^{-1}P(k-1) + \lambda^{-1}\vec{k}(k)\vec{z}^{H}(k)P(k-1)$$

At time sample $k = 1$, the initial conditions for the RLS algorithm can be described by:

1.) Set $\vec{w}(0)$ to the first column vector of an NxN identity matrix, I.

2.) $P(0) = \delta^{-1}I$

$$\textit{where: } \delta = \begin{cases} \text{small positive constant for high SNR} \\ \text{large positive constant for low SNR} \end{cases}$$

Figure 4.4 below depicts the convergence curve for the RLS-CMA algorithm w/ p = 1, and $\lambda = 0.99$ versus the convergence curve for the CMA algorithm w/ $p = 1$, $q = 2$ and $\mu = 1\times10-3$ for the multipath scenario described in the above section

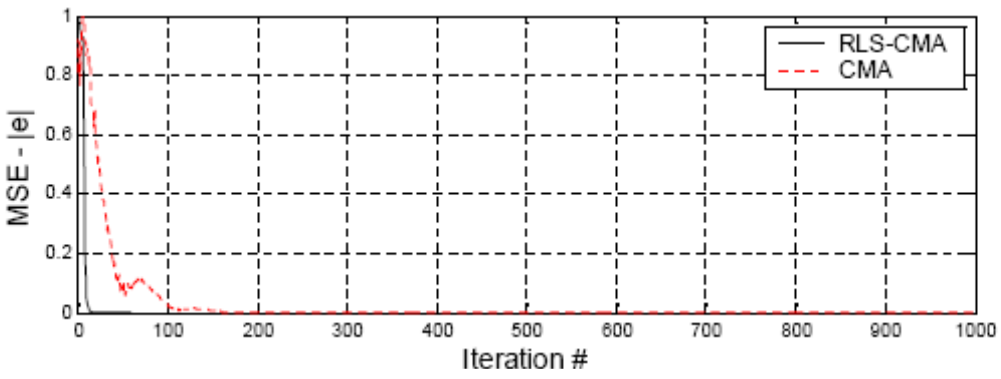

Figure 4.4: Convergence Curve for RLS-CMA vs. CMA w/ $p = 1$, $q = 2$ and $\lambda = 0.99$.

The beampattern for this scenario is provided in Figure 4.5 below.

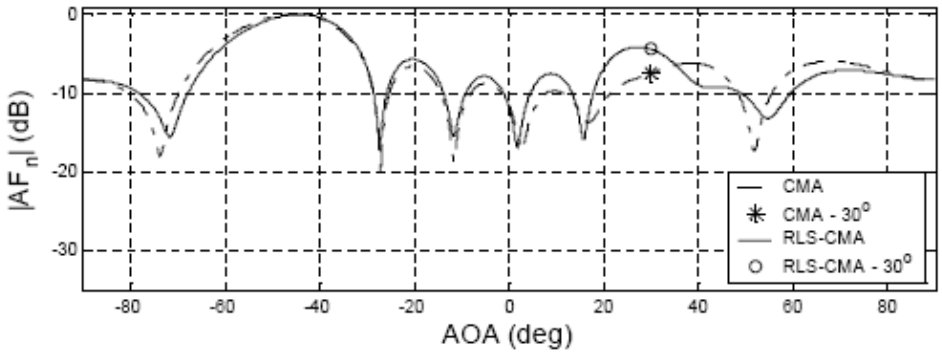

Figure 4.5: Beampattern for RLS-CMA

In the original derivation of the RLS-CMA [8], the prediction error matrix, *P(k)* was determined via:

$$P(k) = \lambda^{-1}P(k-1) - \lambda^{-1}\vec{k}(k)\vec{z}^{H}(k)P(k-1)$$

but it has been my experience that if the line is altered to reflect:

$$P(k) = \lambda^{-1}P(k-1) + \lambda^{-1}\vec{k}(k)\vec{z}^{H}(k)P(k-1)$$

then the algorithm is much more capable of adapting to the proper weights without divergent error and issues with stability no matter what signals are incident upon the array. It is quite possible that since the approximation in equation 4.59 alters the weight update to reflect previous weights, the ability of the algorithm to combat phase

ambiguities presented by tracking the variations in the envelope are diminished. Remember that for the CM cost function, the phase shifted version of a weight vector which produces an output possessing a constant envelope also produces an output with a constant envelope. This is not necessarily the case with the RLS-CMA algorithm. Its ability to measure the variations in the envelope of the array output is troubled when the two signals are highly correlated. Either way, it still provides a greater convergence rate than that of the CMA algorithm but lacks the ability to significantly decrease the contribution from interferers with equal modulus.

Chapter 5

SIMULATIONS AND RESULTS

LMS ALGORITM

CMA ALGORITHM

5.1 LMS ALGORITM [1] [5]

An adaptive array is simulated in MATLAB by using the LMS algorithm. When an array of 4 antennas is used with a separation of _/2 (_ is wavelength), there is a maximum of 3 nulls that can eliminate the interferer. Figures 2-4 shows the convergence of the array for 2 interferers. The interference signals are Gaussian white noise, zero mean with a sigma of 1. The extra system noise to all antennas is white noise with zero mean and a sigma of 0.1. The received signals are MSK signals with an up-sampling of 4 and have amplitude of 1 in the simulations. The true array output *y(t)* is converging to the desired signal *d(t)*. The resulting array vector has an amplitude response as shown in Figure 5. The interferers are cancelled by placing nulls in the direction of the interferers. The received signal arrives at an angle of 35 degrees and the array response is 0 dB. The LMS algorithm clearly works sufficient as the strong interferers are educed. The source code for the MATLAB simulations can be found in Appendix

The measurement of the gradient vector is not possible, and therefore the instantane estimate is used defined by (9) and (10).

$$\hat{r}(n) = d^*(n)x(n)$$

$$\hat{R}(n) = x(n)x^H(n) \quad (10)$$

By rewriting (7) using the instantaneous estimates, the LMS algorithm can be written i final form (11).

$$\hat{w}(n+1) = \hat{w}(n) + \mu x(n)\left(d^*(n) - x^H(n)\hat{w}(n)\right)$$

$$= \hat{w}(n) + \mu x(n)\varepsilon^*(n) \quad (11)$$

\

The graph is obtained between phase of desired signal n LMS output. Here we see tow lines a red and a blue. red is the phase of desired signal and blue is the phase of lms output. So in this way we see there is not much difference in the desired and the obtained output.

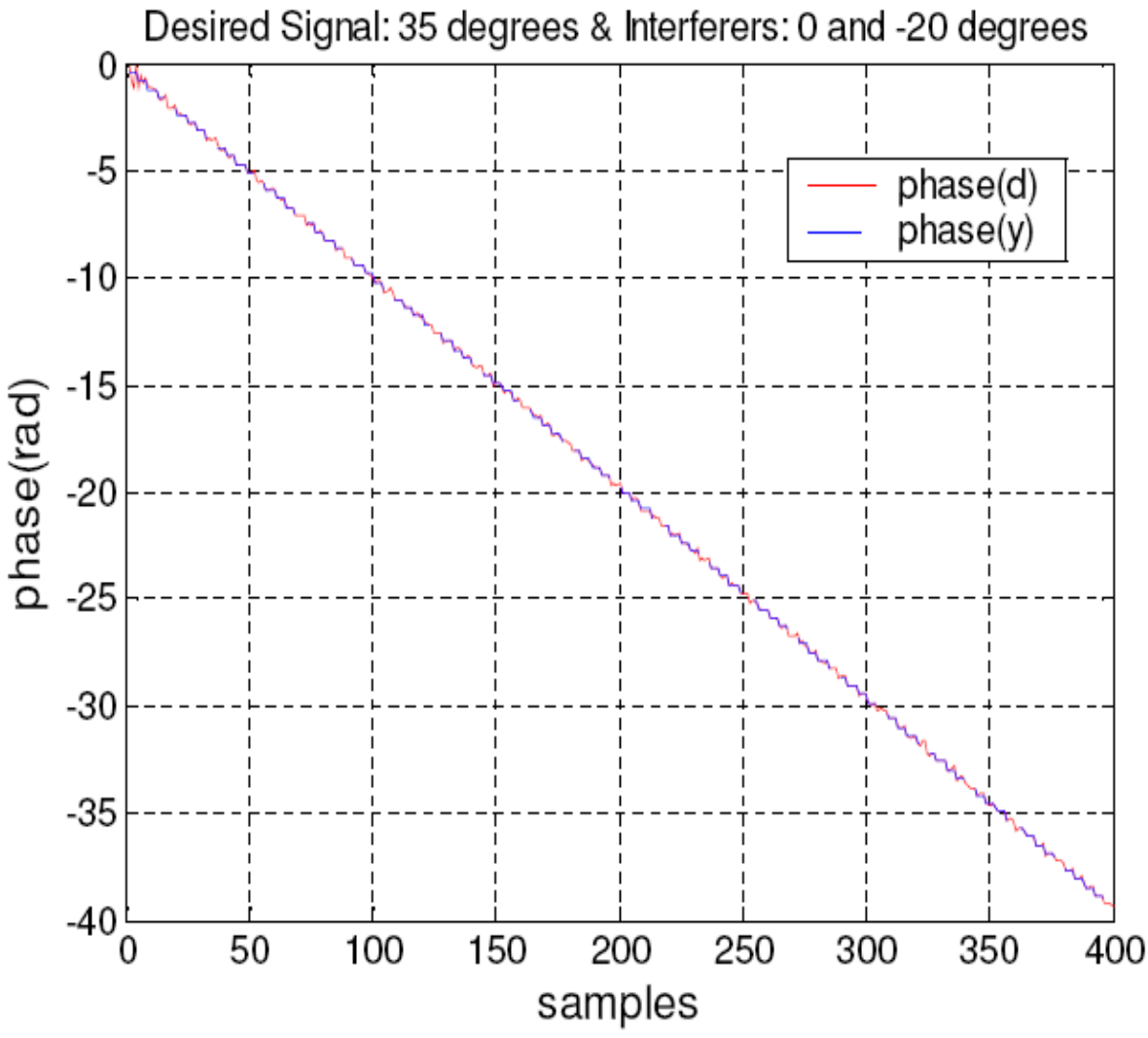

Figure 5.1: Phase of desired signal and LMS output

This is graph obtained between magnitude of desired signal and lms output. In the figure we see that the lms output is a blue line while the desired output shows a little about the Lms output line.

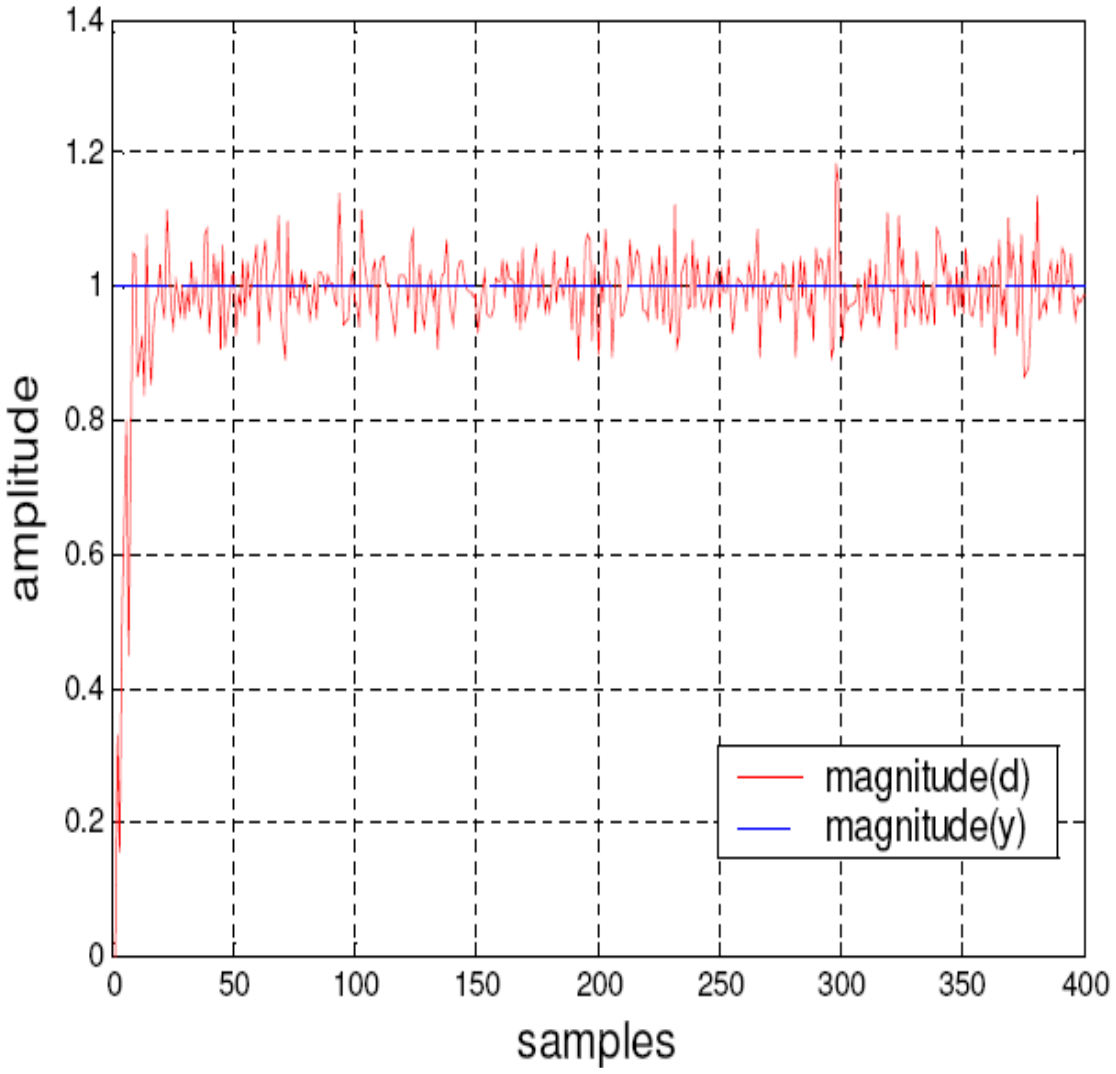

Figure 5.2: Magnitude of desired signal and LMS output

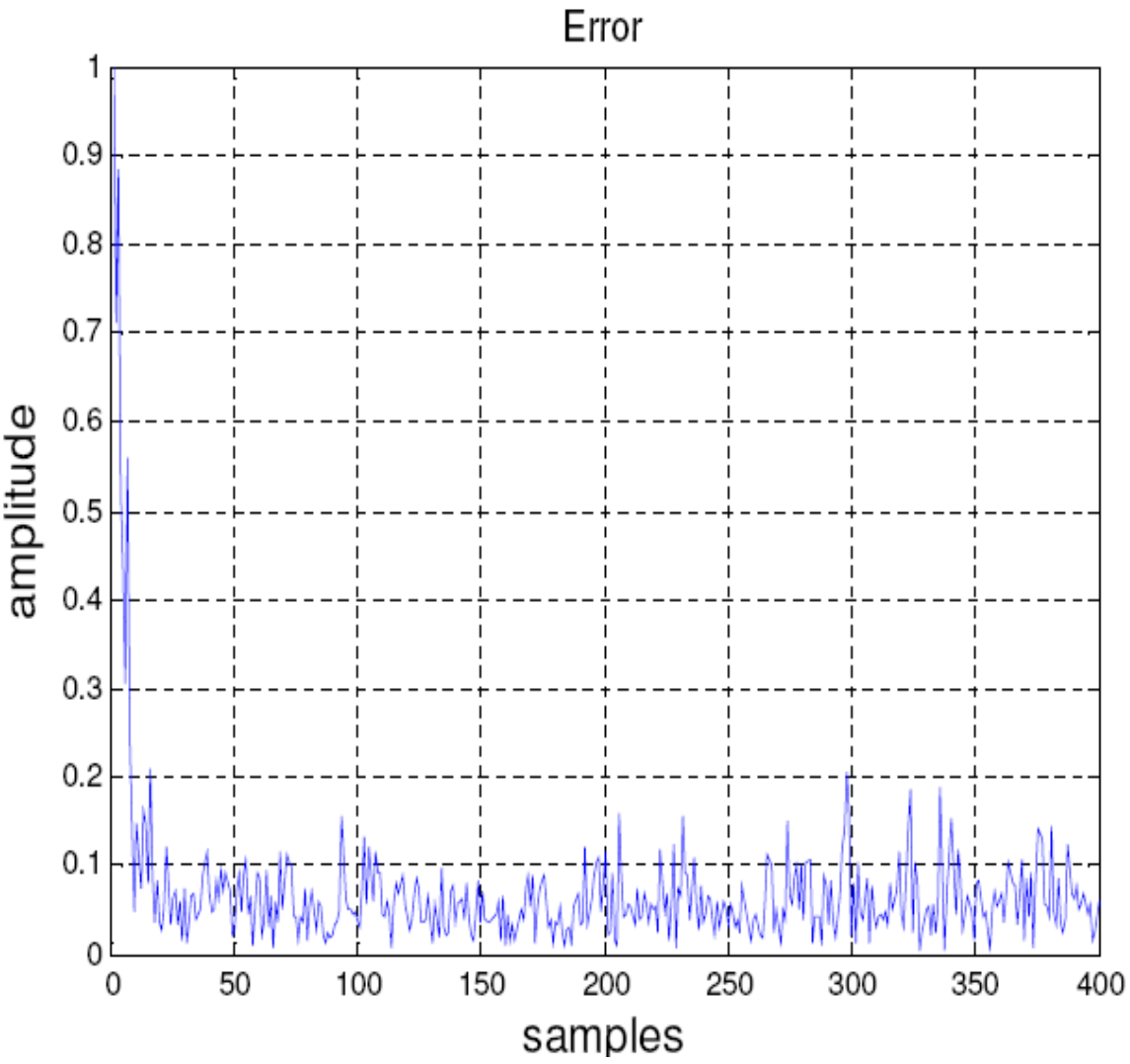

Figure 5.3: Error between desired signal and LMS output

Another graph is obtained for the amplitude response after beamforming.

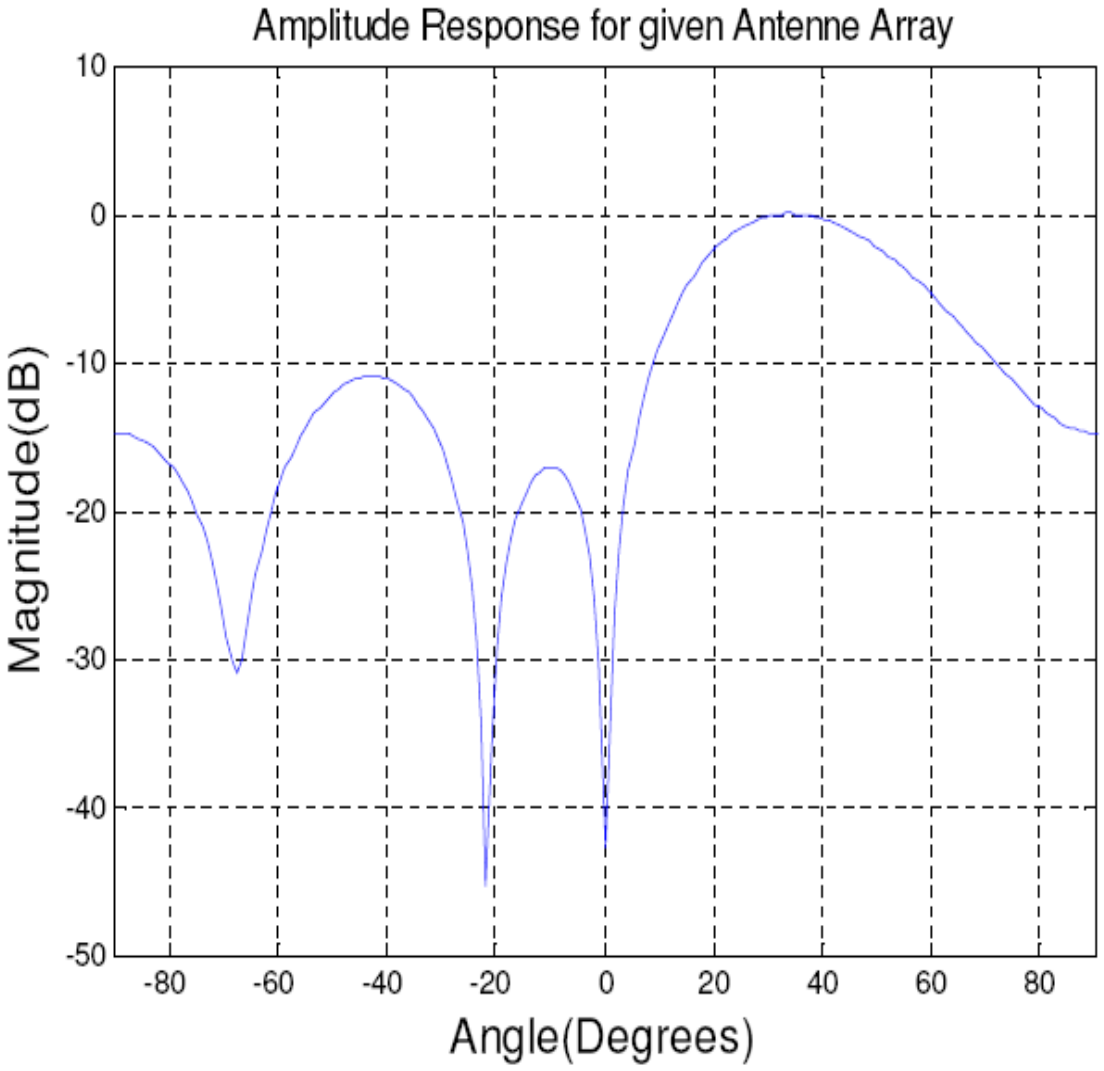

Figure 5.4: Amplitude response after beamforming

5.2 CONSTANT MODULUS ALGORITHM:

The graph is obtained between phases of desired signal n CMA output. Here we see tow lines a red and a blue. red is the phase of desired signal and blue is the phase of CMA output. So in this way we see there is not much difference in the desired and the obtained output.

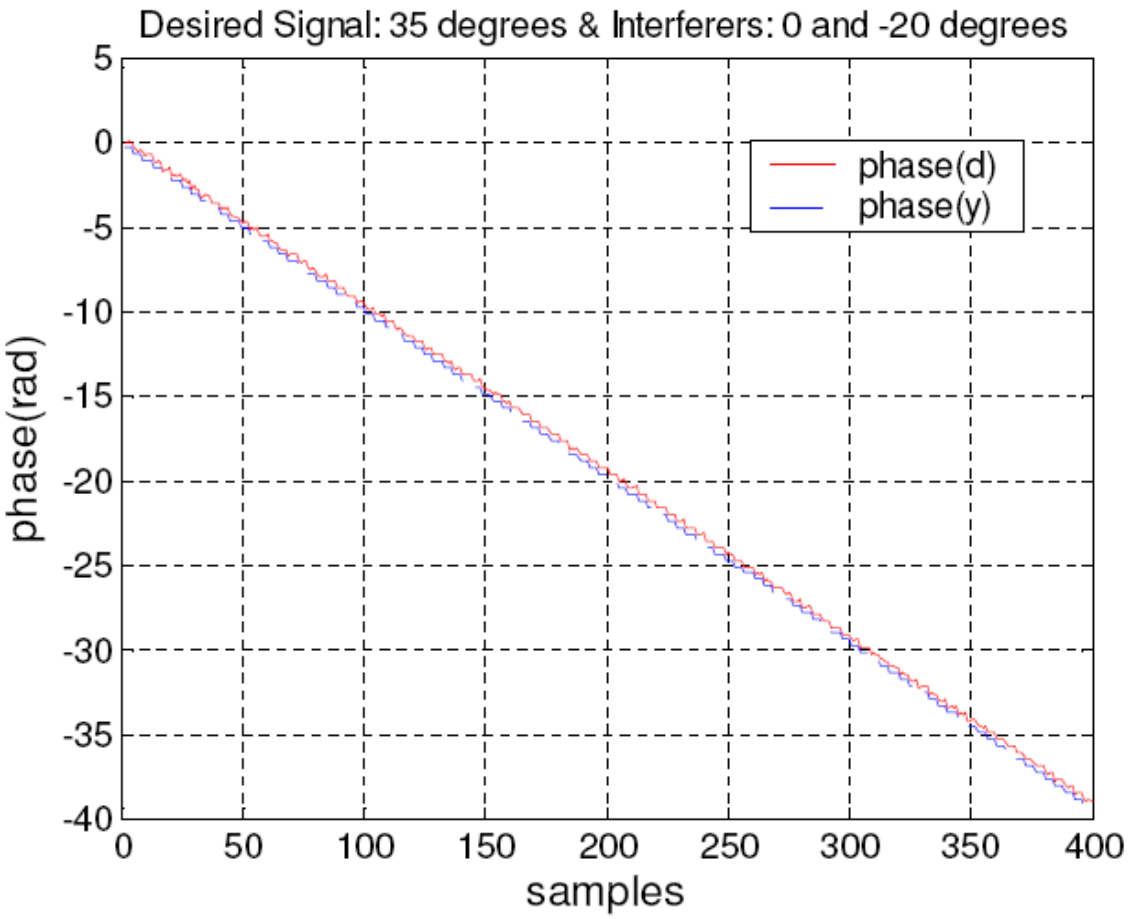

Figure 5.5: Phase of desired signal and CMA output

This is graph obtained between magnitude of desired signal and CMA output. In the figure we see that the CMA output is a blue line while the desired output shows a little about the

Lms output line

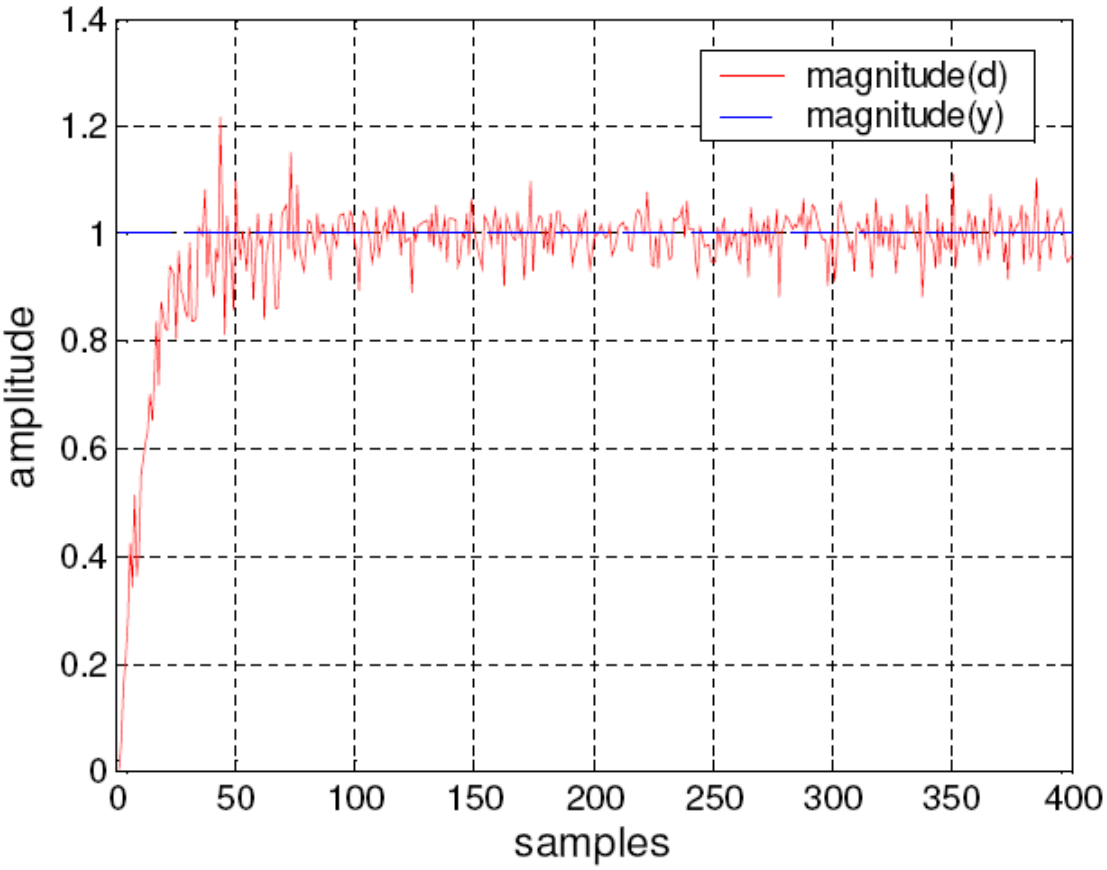

Figure 5.6: Magnitude of desired signal and CMA output

This graph is between the errors of desired signal and CMA output

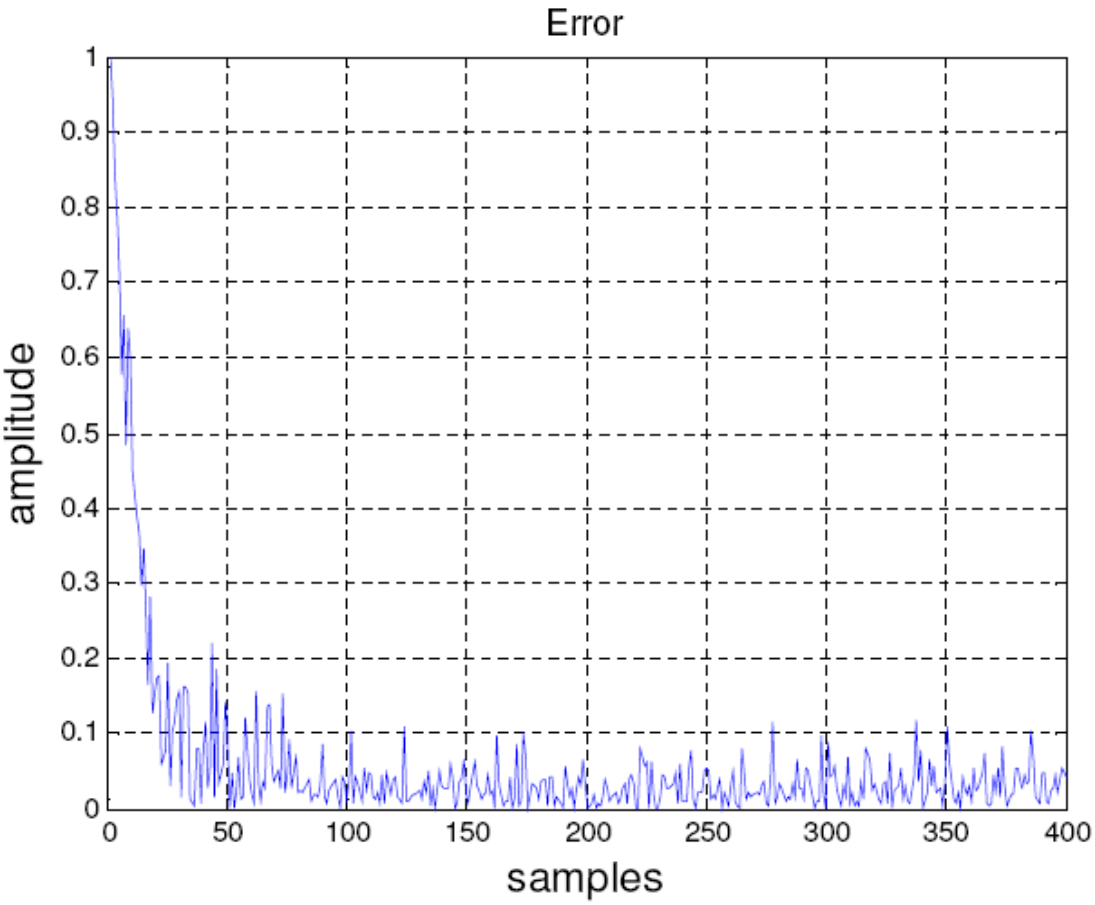

Figure 5.7: Error between desired signal and CMA output

Another graph is obtained for the amplitude response after beamforming

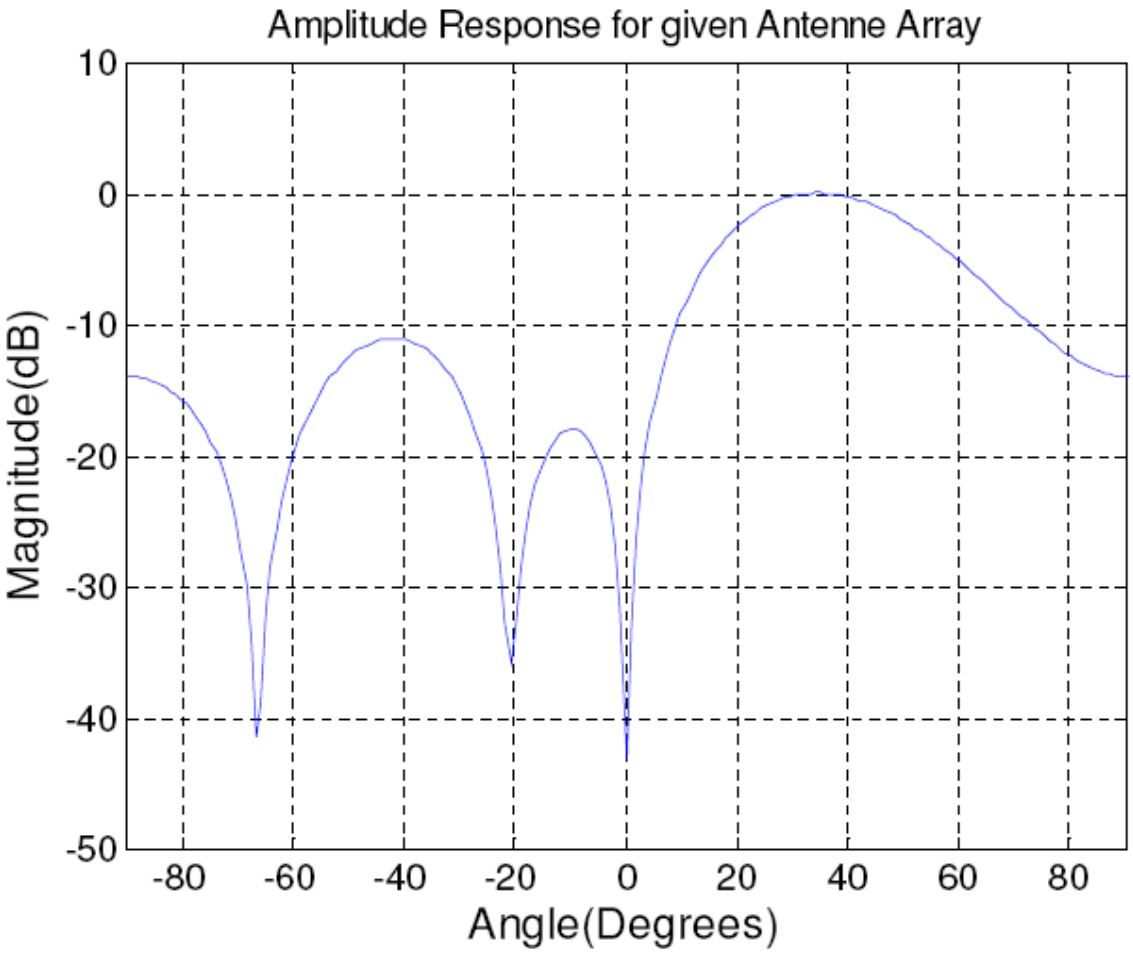

Figure 5.8: Amplitude response after beamforming

CM algorithm converges slower than LMS algorithm. During the efforts to simulate the CM algorithm it was clear that the algorithm is less stable than the LMS algorithm. CM algorithm seems to be more sensitive to gradient constant μ and also for both algorithms, μ can be calculated adaptively.

MATLAB Code for comparing beamforming using LMS algo with CMA (blind) algo

```
clc;
close all;
clear all;
% INITIALIZATIONS
NumofAntenna = 4; % Number of antennas in the array
NumofSamples = 100; % Number of bits to be transmitted
SigmaSystem = 0.1; % System Noise Variance
theta_x = 35 * (pi/180); % direction of signal x
theta_n1 = 0 * (pi/180); % direction of noise source 1
theta_n2 = -20 * (pi/180); % direction of noise source 2
% TIME SETTINGS
theta = pi*[-1:0.005:1];
BitRate = 100;
SimFreq = 4*BitRate; % Simulation frequency
Ts = 1/SimFreq; % Simulation sample period
% GENERATE A COMPLEX MSK DATA TO BE TRANSMITTED
for k=1:NumofSamples
q=randperm(2);
Data(k)=-1^q(1);
end
Data = upsample(Data, SimFreq/BitRate); % Upsample data
t = Ts:Ts:(length(Data)/SimFreq); % Timeline
faz=(cumsum(Data))/8;
signal_x = cos(pi*faz)+j*sin(pi*faz); % The signal to be received
% GENERATE INTERFERER NOISE -> uniform phase (-pi,pi), gaussian amplitude
% distribution(magnitude 1)
signal_n1 = normrnd(0,1,1,length(t)).*exp (j*(unifrnd(-pi,pi,1,length(t))));
signal_n2 = normrnd(0,1,1,length(t)).*exp (j*(unifrnd(-pi,pi,1,length(t))));
% GENERATE SYSTEM NOISES for EACH ANTENNA -> uniform phase (-pi,pi),
gaussian
% amplitude distribution(magnitude 1)
noise = zeros(NumofAntenna, length(t));
for i = 0:NumofAntenna-1,
noise(i+1,:) = normrnd(0,SigmaSystem,1,length(t)).*exp (j*(unifrnd(-pi,pi,1,length(t))));
end;
% ARRAY RESPONSES for DESIRED SIGNAL (X) and INTERFERER NOISES (N1
and N2)
Kd = pi; % It is assumed that antennas are seperated by lambda/2.
response_x = zeros(1,NumofAntenna);
response_n1 = zeros(1,NumofAntenna);
response_n2 = zeros(1,NumofAntenna);
for k = 0:NumofAntenna-1,
```

```
response_x(k+1) = exp(j*k*Kd*sin(theta_x));
response_n1(k+1) = exp(j*k*Kd*sin(theta_n1));
response_n2(k+1) = exp(j*k*Kd*sin(theta_n2));
end;
% TOTAL RECEIVED SIGNAL (SUM of X.*Hx, N1.*Hn1 and N2.*Hn2)
x = zeros(NumofAntenna, length(t));
n1 = zeros(NumofAntenna, length(t));
n2 = zeros(NumofAntenna, length(t));
for i = 0:NumofAntenna-1,
x(i+1,:) = signal_x .* response_x(i+1); % received signal from signal source x
n1(i+1,:) = signal_n1 .* response_n1(i+1); % received signal from noise source n1
n2(i+1,:) = signal_n2 .* response_n2(i+1); % received signal from noise source n2
end;
signal_ns = (noise + n1+n2+x); % total received signal
% EVALUATUING WEIGHTs THOSE SATISFY BEAMFORMING at DESIRED
DIRECTION
y = zeros(1,length(t)); % output
mu = 0.05; % gradient constant
e = zeros(1,length(t)); % error
method = input('Enter the type of beamforming algorithm (lms (1) or cm (2)): ');
switch method
case 1
w = zeros(1,NumofAntenna); % weights
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%LMS Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
for i=0:length(t)-1,
y(i+1) = w * signal_ns(:,i+1);
e(i+1) = signal_x(i+1)-y(i+1);
w = w + mu *e(i+1)*(signal_ns(:,i+1))';
end;
case 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%Constant Modulus Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
w = zeros(1,NumofAntenna); w(1)=eps; % weights
for i=0:length(t)-1,
y(i+1) = w * signal_ns(:,i+1);
e(i+1) = y(i+1)/norm(y(i+1))-y(i+1);
w = w + mu *e(i+1)*(signal_ns(:,i+1))';
end;
otherwise
disp('Unknown method!')
end
% PLOTS
close all;
plot(phase(y),'r');
```

```
hold;
plot(phase(signal_x),'--b');
ylabel('phase(rad)');
xlabel('samples');
title('Desired Signal: 25 degrees & Interferers: 0 and -40 degrees')
legend('phase(d)','phase(y)')
hold off;
figure;
plot(abs(y),'r');
hold;
plot(abs(signal_x),'--b');
ylabel('amplitude');
xlabel('samples');
legend('magnitude(d)', 'magnitude(y)')
hold off;
figure;
plot(abs(e));
ylabel('amplitude');
xlabel('samples');
figure;
for k = 0:NumofAntenna-1,
response(k+1,:) = exp(j*k*Kd*sin(theta));
end;
% CALCULATE ARRAY RESPONSE
R = w*response;
plot((theta*180/pi), 20*log10(abs(R)));
title('Amplitude Response for given Antenne Array');
ylabel('Magnitude(dB)');
xlabel('Angle(Degrees)');
axis([-90,+90,-50,10]);
```

CONCLUSION

In this study of beamforming we have giving an introduction to most of the adaptive beam forming algorithms. They can be characterized in two heading i.e non-blind algorithm and blind algorithm.

In non–blind algorithm we discussed Sample Matrix Inversion, {SMI}, Least Mean Square {LMS}. Recursive Least Squares {RLS}.and in the above methods we need to generate a signal having maximum co-relation with a desired signal. and in the blind algorithms such as Constant Modulus Algorithms{CMA}, Recursive Least Squares Constant Modulus Algorithm (RLS-CMA), Least Squares Constant Modulus Algorithm (LS-CMA}, Steepest Descent Decision Directed Algorithm (SD-DD}.we need to know just the signal characteristics to get the desired signal.

In this study we have compared two algorithms one is a non-blind beamforming algorithm{LMS} and other is a blind beamforming algorithm{CMA}.and have compared the results with regards to smart antenna array system.

REFERENCES

[1] Agee, B.G. "The Least Squares CMA: A New Technique for Rapid Correction of Constant Modulus Signals". *Proceedings of the IEEE ICASSP*. pgs 19.2.1-19.2.4.

[2] Chen, Yuxin. Le-Ngoc, Tho. Champagne, Benoit. Xu, Changjiang. "Recursive Least Squares Constant Modulus Algorithm for Blind Adaptive Array". *IEEE Transactions on Signal Processing*. Vol. 52, No. 5. May 2004.

[3] Compton, R.T. Jr. *Adaptive Antennas – Concepts and Performance*. Prentice Hall. Englewood Cliffs, New Jersey. 1988.

[4] Haykin, Simon. *Adaptive Filter Theory*. Prentice Hall, Englewood Cliffs, New Jersey. 3rd Ed. 1996.

[5] Litva, John & Titus Kwok-Yeung Lo. *Digital Beamforming in Wireless Communications*.
Artech House Publishers. Boston-London. 1996

[6] –Lotter, Michiel; Van Rooyen, Pieter & Van Wyk, Danie. *Space – Time Processing For CDMA Mobile Communications*. Kluwer Academic Publishers. Boston-London.2000.

[7] Proakis, John G. *Digital Communications*. 3rd Ed. McGraw Hill, New York, NY, 1995.

[8] Shetty, Kiran K. "A Novel Algorithm for Uplink Interference Suppression Using Smart Antennas in Mobile Communications". *Master's Thesis, The Florida State University*. 2004.