DEVELOPMENT OF NEURAL NETWORK BASED ADAPTIVE BEAMFORMING ALGORITHM

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

INTEGRATED DUAL DEGREE

(Bachelor of Technology & Master of Technology)

in

ELECTRONICS & COMMUNICATION ENGINEERING

(With Specialization in Wireless Communication)

By

RISHI KUMAR KHANNA

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
FEBRUARY, 2012

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation entitled, "***Development of Neural Network based Adaptive Beamforming Algorithm***" in partial fulfillment of the requirements for the award of the degree of ***Master of Technology*** in ***Electronics & Communication Engineering***, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee (India), is an authentic record of my own work carried out under the guidance of **Dr. Amalendu Patnaik**, Assistant Professor, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee.

The matter embodied in this dissertation report has not been submitted for the award of any other degree elsewhere.

Date: 3rd February
Place: IIT Roorkee

(Rishi Kumar Khanna)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 3.2.12
Place: IIT Roorkee

Dr. Amalendu Patnaik
Assistant Professor
Department of Electronics and Computer Engineering
IIT Roorkee

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, **Dr. Amalendu Patnaik** for his support, encouragement, inspiration, and patience in the research. I am greatly indebted to him for introducing me to the area of neural network based adaptive beamforming and for his guidance throughout the development of this dissertation.

I am deeply thankful to my parents, and my sister for their endless love, constant encouragement, and support. I wish to thank other fellow graduate students in IIT Roorkee with whom I had the great pleasure of interacting.

Finally I thank God for his blessing.

(Rishi Kumar Khanna)

ABSTRACT

In the last decade, the application of adaptive antennas to mobile communications has attracted considerable interest. Adaptive antennas, in a broad sense, implement spatial filtering by means of beamforming, using an antenna array with a small number of antenna elements at the base station. In uplink, the received signals of all antenna elements are complexly, and adaptively weighted in accordance to some performance criterion, to either enhance the carrier-to-interference ratio for the reception of a single mobile's signal, or ultimately, serve many mobiles, transmitting at the same time, and at the same frequency, but spatially separated by utilizing the spatial filter to separate the mobiles' signals, thus implementing spatial-domain-multiple-access, SDMA. To establish the performance criterion, adaption algorithms need a known reference for operation, either a training sequence embedded in the received signals (temporal reference), or the direction-of-arrival of the impinging signals (spatial reference), no explicit reference, except some knowledge of the impinging signal's properties (blind algorithms), or combinations thereof.

Algorithm research for adaptive antennas is a vivid and ongoing discipline, as new mobile communication applications, and requirements emerge. Performance evaluation of these algorithms is possible by computer simulations to a certain extent which is carried out in this dissertation.

Contents

List of Figures

List of Tables

Glossary of Acronyms

AoA	Angle of Arrival
AWGN	Adaptive White Gaussian Noise
BER	Bit error rate
BFN	Beam Forming Network
BS	Base station
CDMA	Code Division Multiple Access
CMA	Constant Modulus Algorithm
d	Inter-element antenna spacing
$d(k)$	Desired signal
DBF	Digital Beam Forming
DDA	Decision Directed Algorithm
DoA	Direction of Arrival
DSP	Digital Signal Processing
$E[\varepsilon^2(k)]$	Mean square error (MSE)
ESPRIT	Estimation of Signal Parameter via Rotational Invariance Technique
f_c	Carrier frequency
FDMA	Frequency Division Multiple Access
$G(\theta, \varphi)$	Beam pattern
Hz	Hertz
I	In-phase
IS-95 CDMA	Interim-95 Code Division Multiple Access
K	Number of antenna elements
LMS	Least Mean Square
LoS	Line of Sight
LS	Least Square
LSMI	Loaded Sample Matrix Inversion
MS	Mobile Station
MT	Mobile Terminal
MUSIC	MUltiple SIgnal Classification
MVDR	Minimum Variance Distortion less Response
N	Number of users in the cell
NN	Neural Network
$n(t)$	Wide-sense stationary zero-mean Gaussian
PCS	Personal Communication System

pdf	Probability Density Function
Q	quadrature-phase
QoS	Quality of Service
R_b	Bit rate
RAB	Robust Adaptive Beamforming
RBFNN	Radial Basis Function Neural Network
RF	Radio-frequency
RLS	Recursive Least Square
r_{xd}	Cross correlation
R_{xx}	Covariance matrix
SDMA	Space Division Multiple Access
SINR	Signal-to-Interference-plus-Noise Ratio
SIR	Signal-to-Interference Ration
SMI	Sample Matrix Inversion
SNR	Signal-to-Noise Ratio
SoI	Signal of Interest
TDMA	Time Division Multiple Access
$\boldsymbol{w}_p^*$	Complex beamforming weight
$\boldsymbol{w}_{opt}$	Optimum weight vector
$x_i(k)$	Array element output
X_k	Data Signal
y_k	Array output at time k
λ	Wave length
$F(\theta)$	Array Factor
σ_n^2	Variance of the background thermal noise
$\varepsilon(k)$	Error function
$\|y(k)\|$	Instantaneous amplitude of the array output
$(\cdot\,\cdot\,\cdot)^H$	Hermitian transpose
μ	Step size/Convergence rate

Chapter 1

Introduction

The growth of telecommunications, from the wired phone to PCSs, is resulting in the accessibility of wireless services that were not formerly considered realistic. In providing different types of wireless services, capacity and reliability are limited by some major impairments such as multipath fading, co-channel interference etc. In mobile communications, the fast increasing number of cellular phones demands an ever rising capacity for future mobile communications systems. In order to meet the needs of an ever increasing demand for mobile communications, the spread spectrum communication scheme (CDMA) was introduced which is an interference limited multiple access technique and incorporates frequency reuse and cell sectorization to enhance its cell capacity.

None of the proposals that include improved air interface and modulation schemes fully exploited the multiplicity of spatial channels that arises because each mobile user occupies a unique spatial location. Space is truly one of the final frontiers when it comes to new generation wireless communication systems. Spatially selective transmission and reception of RF energy promises substantial increases in wireless system capacity, coverage and quality. Filtering in the space domain can separate spectrally and temporally overlapping signals from multiple mobile units. Thus, the spatial dimension can be exploited as a hybrid multiple access technique complementing FDMA, TDMA, and CDMA.

SDMA is the technique that aims to improve the capacity and quality of wireless systems by utilizing the spatial dimension as a resource. It is based on the use of steerable antennas; incorporating electronically controllable Beamforming Networks (BFNs) thus enabling multiple users within the same radio cell to be accommodated on the same frequency and time slot, as illustrated in figure 1.1.

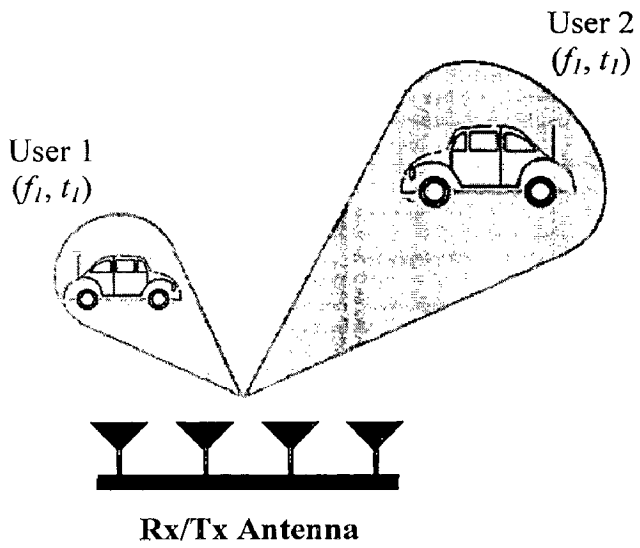


Figure 1.1: SDMA Concept

Realization of this filtering technique is accomplished using smart antennas, which are effectively antenna systems capable of modifying its time, frequency and spatial response. By exploiting the spatial domain via smart antenna systems, the operational benefits to the network operator can be summarized as follows:

- Capacity enhancement
- Coverage extension
- Ability to support high data rates
- Increased immunity to "near-far" problems
- Ability to support hierarchical cell structures

1.1 Motivation and Scope

Smart antenna is recognized as promising technologies for higher user capacity in wireless communication system. The core of smart antenna is the adaptive beamforming algorithms in antenna array. Adaptive Beamforming technique achieves 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. There are several adaptive beamforming algorithms as LMS, SMI, RLS, CMA varying in complexity based on different criteria for updating and computing the optimum weights. Adaptive beamforming is known to have resolution and interference rejection capability when the array steering vector is precisely known, however the performance of adaptive beamforming techniques may degrade severely in the presence of mismatches between assumed array response and true array response.

This problem can be overcome by neural network approach. In this dissertation, the development of a neural-network based robust adaptive beamforming algorithm, which treats the problem of computing the weights of an adaptive array antenna as a mapping problem. Using MATLAB, we investigated a novel approach to robust adaptive beamforming and show clearly how efficiently we compute the weight vector by using the neural network method. This algorithm provides excellent robustness to signal steering vector mismatches, enhances the array system performance under non ideal conditions and makes the mean output array SINR consistently close to the optimal one.

1.2 Objective of the Study

The objective of this work is to analyze and compare the performances of various conventional as well as NN based DBF techniques in terms of robustness, improved SINR. The underlying objectives can be summarized as follows:

1. Literature survey of various beamforming algorithms.

2. Implementation of conventional algorithms (LMS, CMA, SMI, LSMI)
3. Study of role of neural network in beamforming.
4. Use of ANN (in place of conventional algorithms) for beamforming to achieve sufficient reduction in time to implement in real-time.

1.4 Organization of the Dissertation

The dissertation is organized into five chapters including this introductory chapter which outlines the evolution path of various mobile communication schemes and the need of the DBF techniques to achieve special diversity. Literature survey of the work is also provided. Following the introduction, Chapter two presents the fundamentals of digital beamforming, its various types, and then evolution of the adaptive beamforming techniques with their mathematical analysis, associated requirements, assumptions, and specifications. Chapter three presents the simulation results of the two conventional beamforming techniques, LMS, CMA together with a critical analysis and comparison of the performance of the two with variations in antenna design parameters specifications. Chapter four presents the neural network based adaptive beamforming algorithm using RBFNN architecture, its mathematical model, network topology, its learning strategies and then the simulation is performed to justify the performance of the SMI, LSMI and robust adaptive beamforming. Then Chapter five concludes the dissertation with the concluding remarks and outline direction for future scope.

1.5 Literature Survey

Carl B. Dietrich has reported that Smart antennas can improve system performance, and found increasing use of it. He experimentally reported that smart handled terminals demonstrated over 20 dB of interference rejection with single- and multi-polarized arrays and shows that Adaptive beamforming improved reliability, range, talk time, and capacity in both peer-to-peer and cellular systems [1].

Brennan L. E reported the ability of AMTI (airborn moving target indication) radar to reject clutter is often seriously degraded by the motion of the radar. An adaptive receiving array can compensate for platform motion and provide excellent AMTI performance. Scattering from aircraft structure can also distort antenna patterns and reduce AMTI capability. He produced a technique that can adapt the element weights to compensate for near-field scatterers and element excitation errors [2].

Syed Shah Irfan Hussain developed a mobile tracking algorithm that has been devised for adapting the weights of the transmit antenna to attain optimal weights for a particular wireless static channel configuration. This algorithm was based on the sign gradient feedback algorithm (SGF), which was a coarse form of least mean square algorithm (LMS). This algorithm does not require knowledge of the transmit antenna configuration. It has been shown that this algorithm converges to optimum weights of the transmit beamformer as well as reduces their un-necessary perturbations around the point of convergence [3].

Mohammad Tariqul Islam developed a Matrix Inversion Normalized Least Mean Square (MI-NLMS) adaptive beam forming algorithm for smart antenna application which combined the individual good aspects of Sample Matrix Inversion (SMI) and the Normalized Least Mean Square (NLMS) algorithms and he is describe to improve the convergence speed with small BER . MI-NLMS computes the optimal weight vector based on the SMI algorithm and updates the weight vector by NLMS algorithm [4].

Ahmed H. El Zooghby used RBFNN for the direction of Arrival (DOA). He was found that networks implementing these functions were indeed successful in performing the required task and yielded good performance in the sense that the network produced actual output very close to the desired DOA. Also it was demonstrated that these networks are able to generalize, by training and testing using data sets derived from different signal conditions mainly with the effect of noise added to the data used for testing. The main advantage of the RBFNN is the substantial reduction in the CPU time needed to estimate the DOA [5].

Xin Song proposed the robust Capon beamformer (RCB) based on some types of mismatches and shows that the proposed robust Capon beamformer is much less

sensitive to some types of mismatches and the small training sample size than the standard Capon beamformer (CB). Moreover, the mean output SINR of RCB is better than that of CB in a wide range of SNR and N [6].

Michael Chryssomallis has given the overview of smart antenna and provided a basic model for determining the angle of arrival for incoming signals, the appropriate antenna beamforming and the adaptive algorithms that are used for array processing. Moreover he shows how smart antennas, with spatial processing, can provide substantial additional improvement when used with TDMA and CDMA digital-communication systems [7].

Chapter 2

Digital Beamforming and Antenna Array Processing Techniques

Beamforming is a classical method of processing temporal sensor array measurements for signal estimation, interference cancellation, estimating source direction, and spectrum estimation. Beamforming is a technique that utilizes an array of sensor elements to focus a receiver channel on a specific SoI or to transmit signal in a specified direction. It is a spatial filtering used to distinguish the spatial properties between a SoI and the noise and interference signals. Digital beamforming consists of the spatial filtering of a signal where the phase shifting, amplitude scaling, and summer circuit implemented to obtain the desired signal.

Beamforming has been extensively used in wireless systems that utilize a fixed set of antenna elements in an array [8]. Allowing for receive beamforming in uplink transmissions, the signals from these antenna elements are combined to form a movable beam pattern that can be steered to a desired direction that tracks MSs as they move. This allows the antenna system to focus RF resources on a particular mobile station and minimize the interference [9]. When beamforming is used at the mobile station, the transmit beam pattern can be adjusted to reduce the interference to unintended receivers. At a base station, receive beamforming for each desired user could be implemented separately without affecting the performance of other links.

2.1 Fundamentals of Digital Beamforming

Digital beamforming is based on capturing the RF signals and converting them into two streams of binary baseband I and Q signals which jointly represent the amplitudes and phases of signals received at the elements of the array as discussed in chapter one above. In DBF the direction of beam can be realized by forming an array with a number of elemental radiators. As the directivity of the antenna increases, the gain also increases. The increase in directivity means that the antenna receives less interference from signals that are not along its path of maximum gain or directivity.

2.1.1 Beamforming Weight Vector

As discussed before, a beamformer is a spatial signal processor, which produces an output with an emphasized desired signal compared to the input, which is the received signal at the array elements. It accomplishes this by applying a complex beamforming weight vector $\boldsymbol{w} = [w_1\ w_2\ w_3\ \ldots\ w_N]^T$ to the input signal vector:

$$y(k) = \sum_{i=1}^{N} \boldsymbol{w}_i \boldsymbol{x}_i(k) \tag{2.1}$$

where i is array element index, and k is the time index of the received signal sample being considered [10].

2.2 Antenna Array Processing

An antenna array system is composed of a collection of spatially separated antenna elements. The antenna array system has the ability to dynamically adjust the combining mechanism in order to improve system performance as presented in the following subsections.

2.2.1 Uniform Linear Antenna Array

In uniform linear array the spacing between the elements are equal. We consider a K element uniform linear array (ULA). The inter-element spacing between the arrays is d and a plane wave arrives at the array from a direction θ with respect to the array normal and the wave front impinging on the first element travels an additional (d sin θ) distance to arrive at the second element. By setting the phase of the signal at the origin, the phase lead of the signal at element k relative to that at element 0 is κkd sin θ where $\kappa = \frac{2\pi}{\lambda}$, is propagation constant in free space and λ is wave length. Adding the entire component outputs together gives [10] array factor F:

$$F(\theta) = V_0 + V_1 e^{j\kappa d \sin\theta} + \cdots + V_{K-1} e^{j(K-1)\kappa d \sin\theta} = \sum_{k=0}^{K-1} V_k e^{jk\kappa d \sin\theta} \qquad (2.2)$$

in terms of vector inner product: $F(\theta) = \boldsymbol{V}^T \boldsymbol{v}$

where $\boldsymbol{V} = [V_0\ V_1 \ldots V_{K-1}]^T$ the weight vector and $\boldsymbol{v} = \left[1\ e^{j\kappa d \sin\theta} \ldots e^{j(K-1)\kappa d \sin\theta}\right]^T$ is the array propagation vector that contains the information on the angle of arrival of the signal.

2.2.2 Circular Antenna Array

A circular array consisting of L identical isotropic elements evenly spaced in a circle of radius (R). Each element is weighted with complex weight V_l for $l = 0, 1, \ldots L-1$. Since the L elements are equally spaced around the circle of radius R, the azimuth angle of the l^{th} element is given as $\varphi_l = \frac{2\pi}{L}$. If a plane wave impinges upon the array in the direction of (θ, φ) in the coordinate system, the relative phase at the l^{th} element with respect to the center of the array is given as [11]

$$\beta_l = -\kappa R \cos(\varphi - \varphi_l) \sin\theta \qquad (2.3)$$

2.2.3 Planar Antenna Array

In addition to placing elements along a line to form a linear array, one can position them on a plane to form a planar array. Planar arrays provide additional variables which can be used to control and shape the array's beam pattern. The main beam of the array can be steered towards any point in its half space [10, 11]. One of the common configurations of planar arrays is the rectangular array, where the elements are placed along a rectangular grid.

2.3 Digital Beamforming (DBF)

An antenna can be considered to be a device that converts spatiotemporal signals into temporal signals, thereby making them available to a wide variety of signal processing techniques. In this way all of the desired information that is being carried by these signals can be extracted.

A major advantage of digital beamforming lies in the fact that once the RF information is captured in the form of a digital stream, the conversion of the RF signal at each antenna elements into two streams of binary baseband signals representing I and Q channels. The digital baseband signals then represent the amplitudes and phases of signal received at each element of the array. The process of beamforming involves weighting these digital signals, thereby adjusting their amplitudes and phases such that when added together they form the desired beam. It is possible to take a weighted sum of the antenna element out puts which maximize power in a given direction by implementing a multi-element antenna array at receiving end.

2.4 Narrowband Beamforming

A signal is considered to be narrowband if all the frequency components of the signal as they travel across the array undergo only a phase shift and not a change in the magnitude. In other words, the signal bandwidth is very small compared to the carrier frequency. If the signal bandwidth is very small compared to the carrier frequency then

the phase shifts undergone by the frequency components at the two edges of the band are almost equal. For a narrowband signal, narrowband array processing is the appropriate choice [12]. A narrowband array has one weight per element followed by a linear combiner.

$$y(t) = \boldsymbol{w}^H(t)\boldsymbol{u}(t) \tag{2.4}$$

Where superscript H denotes the Hermitian (complex conjugate) transpose and, $\boldsymbol{w}(t)$ is known as the beamformer weight vector.

$$\boldsymbol{w} = [w_0\ w_1\ w_2 \ldots w_{M-1}]^T \tag{2.5}$$

2.5 Wideband Beamforming

A signal is considered to be wideband if all the frequency components of the signal as they travel across the array undergo a phase shift with a change in the magnitude. In other words, the signal bandwidth is not very small compared to the carrier frequency. If the signal bandwidth is not very small compared to the carrier frequency then the phase shifts undergone by the frequency components at the two edges of the band are not equal. Therefore the phase shift line appears not to be flat across the bandwidth. For a wideband signal, wideband array processing is the appropriate choice [13].

Then, the output for a wideband adaptive beamformer may be expressed as,

$$y(t) = \bar{\boldsymbol{w}}^H(t)\bar{\boldsymbol{u}}(t) \tag{2.6}$$

which is identical in form to the array output of the narrowband array. The wideband beamformer is more complex than the narrowband beamformer.

2.6 Switched Beamforming

Switched beamforming provides non-uniform coverage when the MS is moving from one beam to another [13], there might be a call drop due to no coverage zone in between two beams. In addition when the MS moves from one beam to another

intra-cell handover takes place, the frequency of which directly depends on the beam width. Adaptive beamforming overcome these two problems, as the beam follows the MSs to where ever they move. It is the simpler approach where the direction of arrival (DOA) of signal is detected in that direction by multiplying pre-computed complex vector. These algorithms such as: MUSIC and ESPRIT algorithms. These algorithms restrict their applicability in a wireless mobile communications because they cannot perform steer nulls and suppressing [14] strong interference.

2.7 Adaptive Beamforming

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 while signals of the same frequency from other directions are rejected [10]. 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. Most of the algorithms are concerned with the maximization of the signal to noise ratio (SNR).

2.7.1 Adaptation Criteria

The optimum weights using different criteria are all given by the solution of the Wiener equation. These optimum criteria are as valid as in communications as in other applications. Though different forms of criteria appear in the [10], it can be shown that they all stem from the Wiener-Hopf equation. This is because the wiener solution provides the upper limit on the theoretical adaptive beamforming steady-state performance. Some of the most frequently used performance criteria are the Mean Square Error (MSE), the Maximum Likelihood (ML), Maximum Signal to Noise Ratio (MSNR) and Maximum Signal to Interference Noise Ratio (SINR) [13]. These performance criteria are usually expressed as cost functions and the weights are adapted

iteratively until the cost functions converge to a minimum value. Once the cost function is minimized we can be assured that the performance criterion is met and the algorithm is said to have converged. There are several factors that are to be considered while choosing an adaptive algorithm like the convergence rate of the algorithm, complexity and robustness. The convergence rate is the number of iterations required for the convergence of the algorithm. In mobile environments, convergence rate is important to converge to optimum before the channel conditions change [15].

2.8 Adaptive Beamforming Algorithms

The basic adaptive algorithms that have been investigated for beamforming in mobile communications include the trained and blind adaptive beamforming algorithms as classified in figure 2.1 under adaptive array (beamforming) algorithms [10, 15]. Adaptive beamformer have additional capability to steer nulls in the direction of interfering signals. An adaptive beamformer comprised of N antenna elements can effectively reject $N-1$ interfering signals [13].

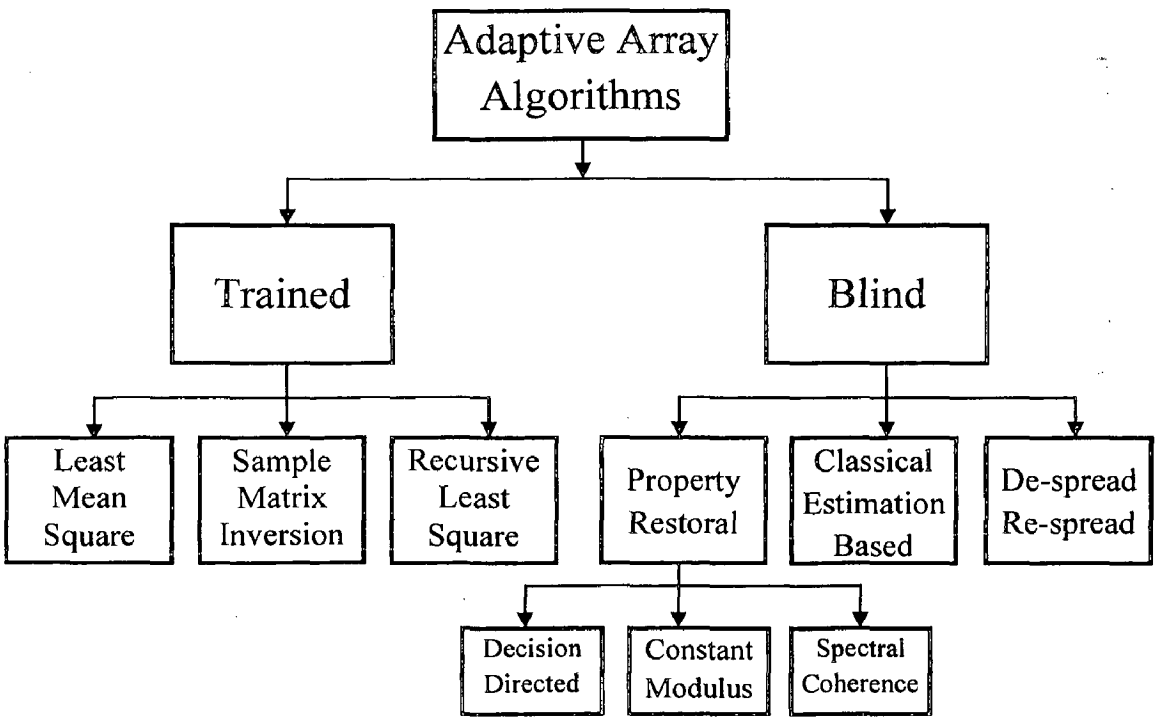


Figure 2.1: Classification of adaptive array algorithms

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 identify the desired user. On the other hand, blind adaptive algorithms do not need any training [10]. They attempt to restore some type of characteristic of the transmitted signal in order to separate it from other users in the surrounding environment.

A basic adaptive beamformer is shown in figure 2.2. The weight vector $\boldsymbol{w}$ is calculated using the statistics of signal $\boldsymbol{x}_N(k)$ arriving at the antenna array.

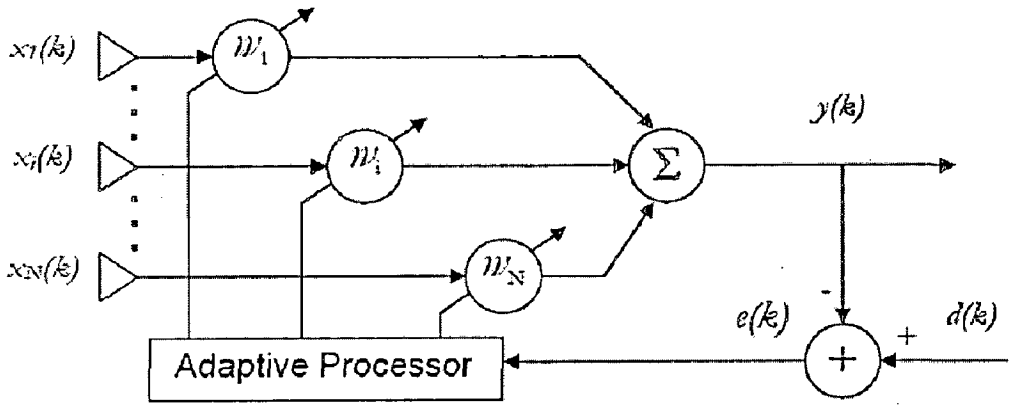


Figure 2.2: Adaptive beamforming configuration

Through a feedback loop the weights, $w_1, \ldots, w_N$ are updated by the time sampled error signal:

$$e(k) = d(k) - y(k) \tag{2.7}$$

where the training sequence, $d(k)$, is the desired signal and $y(k)$ is the output of the adaptive array,

$$y(k) = \boldsymbol{w}^H \boldsymbol{x}(k) \tag{2.8}$$

The feedback system attempts to direct the weights at each element to their optimal weights, $\boldsymbol{w}_{opt}$. The adaptive processor adjusts the weight vector to minimize the mean square error (MSE) of the error signal $e(k)$ given by

$$E[|e(k)|^2] = E[|d(k) - y(k)|] \tag{2.9}$$

The output of the array $y(k)$ with variable element weights is the weighted sum of the received signals at the array elements and the noise at the receivers connected to each

element. The weights are iteratively computed based on the array output, a reference signal that approximates the desired signal, and previous weights. The reference signal $d(k)$ 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 [16, 17].

2.8.1 Trained Adaptive Algorithms

Trained adaptive beamforming algorithms use a finite set of training symbols to adapt the weights of the array and maximize the SINR. First, a training signal, which is known to both the transmitter and receiver, is transmitted by the transmitter. The beamformer in the receiver uses the information of the training signal to compute the optimal weight vector. After the training period, data is sent and the beamformer uses the weight vector computed previously to process the receiver signal. The trained adaptive algorithms drawback is the excessive utilization of transmission time and wastage of bandwidth. The trained adaptive algorithms are categorized based on their adaptation criteria and they are the LMS, SMI and RLS methods [10, 13, 16]. For simulation study we use LMS from the trained adaptive algorithm.

2.8.1.1 LMS Algorithm

The LMS algorithm can be considered as the most common adaptive algorithm for continuous adaptation. It uses the steepest-descent method and recursively computes and updates the weight vector. Due to the steepest-descent the updated vector will propagate to the vector which causes the least mean square error (MSE) between the beamformer output and the reference signal. The LMS algorithm is simple to implement. But the dynamic range over which it operates is quite limited. Since the received signals in a mobile radio system vary by more than 20dB [18], power control is required if the LMS algorithm is to be used. The normalized LMS algorithm can be used to overcome the dynamic range limitation. The following derivation for the LMS algorithm is found in [17]. The MSE is defined by:

$$\varepsilon^2(k) = |d^*(k) - \boldsymbol{w}^H \boldsymbol{x}(k)|^2 \tag{2.10}$$

$d^*(k)$ is the complex conjugate of the desired signal. The signal $\boldsymbol{x}(k)$ is the received signal from the antenna elements, and $\boldsymbol{w}^H\boldsymbol{x}(k)$ is the output of the beamformer antenna and H is the Hermitian transposition operator. The expected value of both sides leads to [17, 18]

$$E[\varepsilon^2(k)] = E[d^2(k)] - 2\boldsymbol{w}^H\boldsymbol{r} + \boldsymbol{w}^H\boldsymbol{R}\boldsymbol{w} \tag{2.11}$$

In this relation, $\boldsymbol{r}$ and $\boldsymbol{R}$ are defined as [26, 27]

$$\boldsymbol{r} = E[d^*(k)\boldsymbol{x}(k)] \tag{2.12}$$

$$R = E[\boldsymbol{x}(k)\boldsymbol{x}^H(k)] \tag{2.13}$$

$\boldsymbol{R}$ is referred to as the covariance matrix and $\boldsymbol{r}$ is the cross-correlation. If the gradient of the weight vector $\boldsymbol{w}$ is zero, the MSE is at its minimum [17]. This leads to

$$\nabla\boldsymbol{w}(E[\varepsilon^2(k)]) = -2\boldsymbol{r} + 2\boldsymbol{R}\boldsymbol{w} = 0 \tag{2.14}$$

The solution of (2.14) is called the Wiener-Hopf equation for the optimum Wiener solution

$$\boldsymbol{w}_{opt} = R^{-1}\boldsymbol{r} \tag{2.15}$$

The LMS algorithm converges to this optimum Wiener solution. The basic iteration is based on the following simple recursive relation [17, 18]

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \frac{1}{2}\mu(-\nabla E[\varepsilon^2(k)]) \tag{2.16}$$

Combining (2.14) with (2.16) gives:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu(\boldsymbol{r} - \boldsymbol{R}\boldsymbol{w}(k)) \tag{2.17}$$

The measurement of the gradient vector is not possible, and therefore the instantaneous estimate is used as defined by (2.18) and (2.19)

$$\widehat{\mathrm{R}}(k) = \boldsymbol{x}(k)\boldsymbol{x}^H(k) \tag{2.18}$$

$$\hat{\boldsymbol{r}}(k) = d^*(k)\boldsymbol{x}(k) \tag{2.19}$$

By rewriting (2.17) using the instantaneous estimates, the LMS algorithm can be written in its final form (2.20)

$$\begin{aligned} \hat{\boldsymbol{w}}(k+1) &= \hat{\boldsymbol{w}}(k) + \mu\boldsymbol{x}(k)(d^*(k) - \boldsymbol{x}^H(k)\hat{\boldsymbol{w}}(k)) \\ \hat{\boldsymbol{w}}(k+1) &= \hat{\boldsymbol{w}}(k) + \mu\boldsymbol{x}(k)\varepsilon^*(k) \end{aligned} \tag{2.20}$$

One of the issues on the use of the instantaneous error is concerned with the gradient vector, which is not the true error gradient. The gradient is stochastic and therefore the estimated vector will never be the optimum solution. The LMS cannot filter the system noise, as it is not correlated for all given antennas. The interferers are cancelled by placing nulls in the direction of the interferers. The LMS algorithm can be the best choice for systems with a relatively small number of antenna elements.

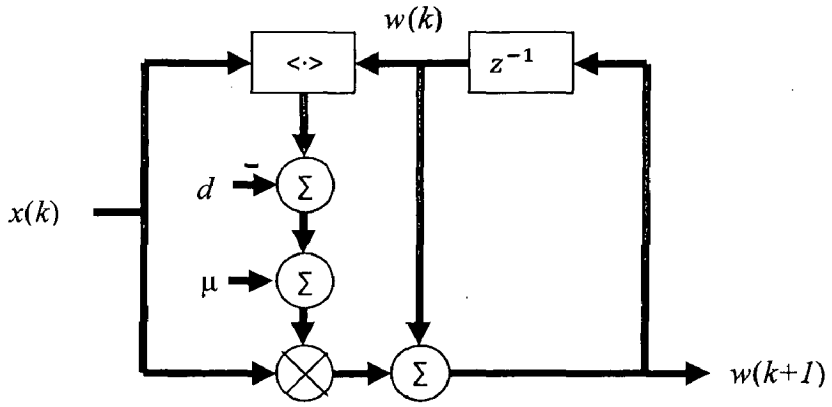


Figure 2.3: LMS algorithm Signal-flow graph

2.8.1.2 Recursive Least Square Algorithm

RLS is a recursive version of the LS approach, where the inversion of the matrix is carried out using a recursion. An important feature of RLS algorithm is that the inversion of the covariance matrix is replaced at each step by a simple scalar division. This feature reduces the computational complexity while maintaining a similar performance. For a fast fading channel, a value slightly below unity should be chosen. A value of 0.95 was reported to be reasonable [19].

RLS requires computations of the order of M^2 where M is the number of antenna array elements. RLS operates on a sample-by-sample basis and hence to process N samples, RLS requires computations on the order of M^2N operations. In contrast to the LMS algorithm, a RLS adaptive algorithm approximates the Wiener solution directly [16]. The RLS algorithm uses weighted sums for estimating R_{xx} and r_{xd} using the following equations,

$$\hat{R}_{xx} = \sum_{i=1}^{N} \gamma^{n-1} x(i) x^*(i) \tag{2.21}$$

$$\hat{r}_{xd} = \sum_{i=1}^{N} \gamma^{n-1} d^*(i) x(i) \tag{2.22}$$

The matrix inversion obtained recursively and the weight update equation is given in[10]

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + q(n)[d^*(n) - \boldsymbol{w}^H(n-1)x(n)] \tag{2.23}$$

where

$$q(n) = \frac{\gamma^{-1} R_{xx}^{-1}(n-1)x(n)}{1 + \gamma^{-1} x^H(n) R_{xx}^{-1}(n-1)x(n)} \tag{2.24}$$

where

$$R_{xx}^{-1}(n) = \gamma^{-1}[R_{xx}^{-1}(n-1) - q(n)x(n)R_{xx}^{-1}(n-1)] \tag{2.25}$$

Recalling that the statistically optimum Wiener solution is

$$\boldsymbol{w}_{opt} = R_{xx}^{-1} r_{xd} \tag{2.26}$$

The RLS algorithm converges faster than the LMS algorithm but at the expense of computational complexity. RLS requires an initial estimate of R_{xx}^{-1}matrix and a reference signal.

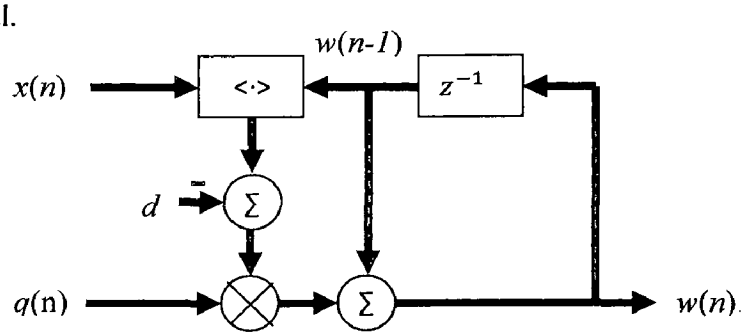


Figure 2.4: Signal-flow graph of the RLS algorithm

2.8.2 Blind Adaptive Algorithms

Adding a known training sequence prior to data transmission leads to additional overhead, which limits the channel bandwidth available for data. The techniques generally referred to as blind adaptive algorithms adapt by attempting to restore some known property of the received signal. It uses statistical information of the signal. Blind algorithms can be classified as property restoral algorithms, channel estimation algorithms and de-spreads and re-spread algorithms. The algorithms have been studied by a number of authors [10].

2.8.2.1 Constant Modulus Algorithm (CMA)

The CMA gives a single beamformer vector. This is sufficient for blind equalization applications, where the received signal consists of several temporal shifts of the same CM signal, and we do not have to recover all of them [20]. The CM algorithm is applicable to constant modulus signals and adjusts the weight vector of the adaptive array to minimize the variation of the envelope at the output of the array. After the algorithm converges, the array can steer a beam in the direction of the signal of interest (SOI), and nulls in the directions of the interference. The only disadvantage of CM algorithm is that it converges to the strongest user in the channel. Godard was the first to make use of constant modulus algorithm (CMA) property to carry out blind equalization in two-dimensional digital communications systems [21]. The output of a DBF array is given as:

$$y(k) = \boldsymbol{w}^H \boldsymbol{x}(k) \tag{2.27}$$

Assuming that the transmitted signal $S(k)$ has a constant envelope, the array output $y(k)$ should have a constant envelope as well. This can be accomplished by adjusting the array weight vector $\boldsymbol{w}$ in such a way as to minimize the cost function, which measures the signal modulus variation and is given in a general sense by

$$\varepsilon = D\{f[S(k)], f[y(k)]\} \tag{2.28}$$

Where D and f are some defined specific distance metrics. In particular, the updated value of the weight vector at time $k+1$ is computed by using the simple recursive relation

$$w(k+1) = w(k) + \frac{1}{2}\mu(-\nabla E[\varepsilon(k)]) \tag{2.29}$$

Where the gradient $\nabla E[\boldsymbol{\varepsilon}(k)]$ can be found as:

$$\nabla E[\boldsymbol{\varepsilon}(k)] = -E\{[r_p(k) - |y(k)|^2]y(k)x(k)\} \tag{2.30}$$

In reality, an exact measurement of the gradient vector is not possible, since this would require a prior knowledge of $E[|S(k)|^p]$. The strategy is to scale $S(k)$ to unity and to use the instantaneous estimate,

$$\nabla(E[\varepsilon(k)]) \cong -[1 - |y(k)|^2]y(k)x(k) \tag{2.31}$$

The weight vector can then updated as

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu[1 - |y(k)|^2]y(k)x(k) \tag{2.32}$$

The error is different and defined by [10] as

$$\varepsilon(k) = [1 - |y(k)|^2]y^*(k) \tag{2.33}$$

The CM algorithm can be found in many derived forms. The error function for a derived version is given as

$$\varepsilon(k) = \frac{y(k)}{|y(k)|} - y(k) \tag{2.34}$$

We can express the updating equation as

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu x(k)\varepsilon^*(k) \tag{2.35}$$

The advantage of the CM algorithm is the fact that it only needs the instantaneous amplitude of the array output $|y(k)|$ and therefore no synchronization is required. Due to this property, the CM algorithm is relatively simple to implement.

2.8.2.2 Decision-Directed Algorithm (DDA)

In the decision- directed algorithm, the tab weights of the adaptive equalizer are adjusted for an adaptive process and widely used for adaptive equalization to combat inter-symbol interference (ISI) in digital communications. Although the decision-directed algorithm widely used in adaptive equalization, it is only applied to diversity combining and beamforming [10]. The decision-directed beamforming for wireless communications has been studied by a number of authors [22].

Chapter 3

Analysis and Simulation

3.1 Digital Beamforming Algorithms

In mobile cellular communication frequency reuse and cell sectorization are widely implemented in order to increase cell capacity and coverage. But due to signal power radiated in directions and throughout the cell area further increment in capacity and coverage is not practically reliable [13]. Thus, Beamforming is one to minimize these constraints. There are mainly two approaches to Beamforming: switched and adaptive beamforming based on their complexity and efficiency.

The simpler one is switched BF where the direction of arrival (DoA) of the signal is detected and corresponding beam is formed in that direction by multiplying complex array factor (AF). The DOA in the switched beamforming can be computed using algorithms such as MUSIC and ESPRIT but these algorithms do not suppress strong interfering signals [13].

Adaptive beamforming is more complex, but highly efficient where the radiation pattern is constructed dynamically in which interferers are blocked by placing nulls and the beam is formed in the direction of users. Beam steering implemented in the direction of the user as it moves using fully adaptive antenna array. The complex weights are computed by using adaptive algorithms: LMS, RLS, SMI, CMA, and DDA which are discussed under chapter 2 above. Thus, the required pattern is formed by multiplying the computed weights with the signal from the antenna array. In this work,

we try to emphasize on LMS and CMA of the discussed beamforming algorithms, in the simulation study for the performance measurement. The adaptive algorithms LMS & CMA are less complex and simple to implement relative to the others.

$$y(k) = \sum_{i=1}^{N} w_i x_i(k)$$

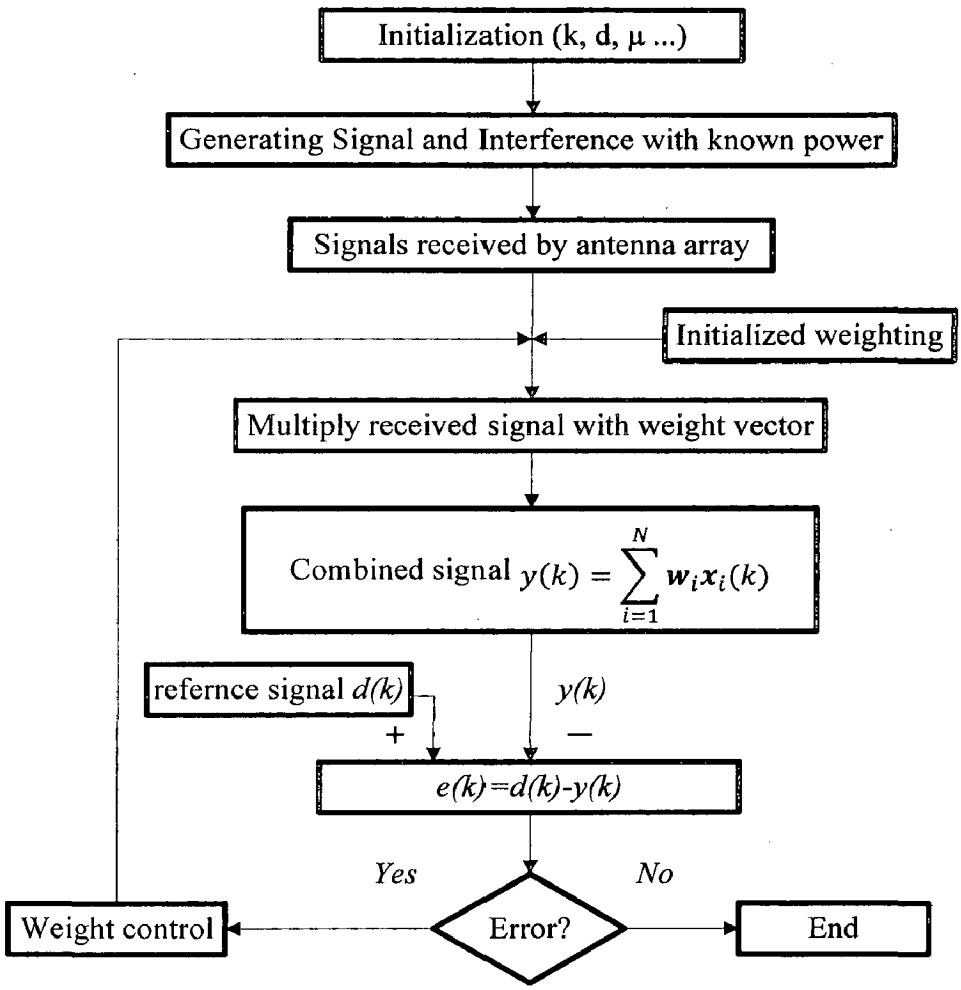


Figure 3.1: Adaptive Beamforming Algorithm Flow chart

In CDMA, although there is no hard limit on the number of mobile users served, there is a practical limit on the number of simultaneous users in a cell to control the interference among users having the same pilot signal. The system capacity bound with acceptable QoS of CDMA systems supporting voice and data traffic in the reverse link is given in many literatures [15]. We assume that antenna array is used at the base station only, and at the mobile station we have a single antenna, which results a channel

configuration of multiple input single output as shown in figure 5.1. The service area of a mobile cellular network is assumed to be covered by many identical regular hexagonal cells and mobile stations are uniformly distributed in each cell with a random speed v which has a probability density function (pdf) $f_V(v)$. Mobile stations move randomly in any direction with equal probability over $[0, 2\pi)$ angle of arrival. In mobile cellular system implementing beamforming algorithm, it is assumed that the bandwidth separation is θ_b, which is further assumed as the minimum separation angle between two consecutive mobile stations using the same physical channel [15]. The inter-antenna elements spacing (d) is in terms of signal wavelength (λ) and we use few number of array antenna elements: 4 and 6 to emphasize on some of the beamforming algorithm effective with few number of antenna elements.

3.2 Simulation Model and Parameters

The simulation has been performed using MATLAB codes. From the Beamforming algorithm discussed above we choose for the simulation study only two of them from adaptive algorithm: LMS and CMA. The simulating codes have loops to update the weight vectors that are used to estimate the desired signal. The parameters used in this simulation study are: antenna noise figure (base station system noise), the step size (μ), antenna spacing (d), and antenna array element. The efficiency of these algorithms is dependent on the selected values for the above parameters. In addition to these parameters there are other parameters which are considered to be fixed with constant values. Mainly the simulation model for these two algorithms looks for specific system model and analysis model as discussed in the next sub sections. The error is a result of the extra 'system' noise that is added to all antennas. The interference signals are Gaussian white noise, zero mean with a standard deviation of 1.

The following assumption is considered to simplify the simulation environment:

- Perfect power control.
- A beam formed by the array can be steered in any direction in the azimuth plane.

Table 3.1: Simulation parameters

Inter-antenna element spacing (d)	0.25λ, 0.5λ, and λ
Step size (μ)	0.1,0.01 and 0.001
Number of antenna array elements	4 and 6
System noise (antenna)	0.1 and 0.01
No of interference in the system	2, 3,4 and 5
Desired signal AOA	20 degree
Steering angles for null interferences	-82, -40, -19, -10, 0, and 40 (degree)
Bit rate	100
No of samples (iteration time)	210
System band width (W)	1.25 MHz
Bit-error rate (BER)	10^{-3}

3.3 Simulation Results and Discussion

3.3.1 LMS Adaptive Beamforming Algorithm

The LMS is discussed in subsection 2.8.1.1; an adaptive array is simulated in MATLAB by using the LMS algorithm explained above. When an array of 6 element antennas are used, there is a maximum of 5 nulls that can eliminate the interferer and when 4 element antennas are used, there is maximum of 3 nulls that can reject interferers. Synchronization of the LMS algorithm usually involves the use of correlators in the digital beamformer to align the desired vector with the incoming signal. The interferers are cancelled by placing nulls in the direction of the interferers. Figure 3.2 below is the simulation result obtained using 6 antenna elements and assuming five interfering signals. The inter-antenna spacing is equal to half wavelength. From this figure one can observe that, LMS adaptive beamforming weighted signal $|y|$ starts from zero and estimate the normalized desire signal magnitude. The error amplitude also is minimized as the number of iteration (samples) becomes greater. It means LMS error rapidly decreases up to 60 samples and then very slowly. The reason why the LMS error start from one and weighted signal from zero is that, trained adaptive algorithm uses reference signals and initialized the weight vector to zero.

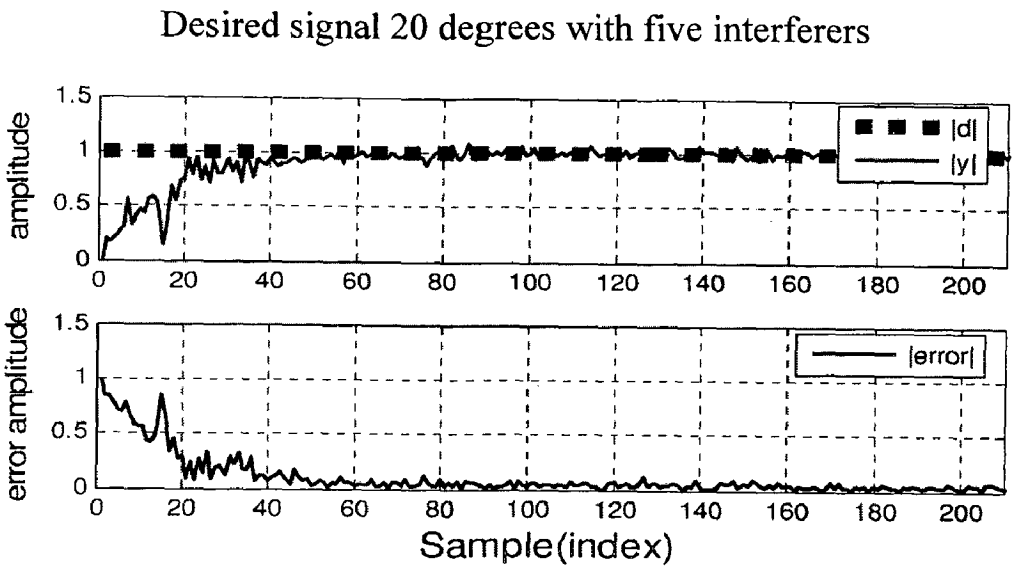


Figure 3.2: LMS algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.1, $\mu = 0.01$, and $d=0.5\lambda$, Amplitude versus time of iteration.

Figure 3.3 is similar to figure 3.2 but their difference lies on the simulation parameter value. Comparing them, we can observe that Figure 3.3 is more efficient in estimation.

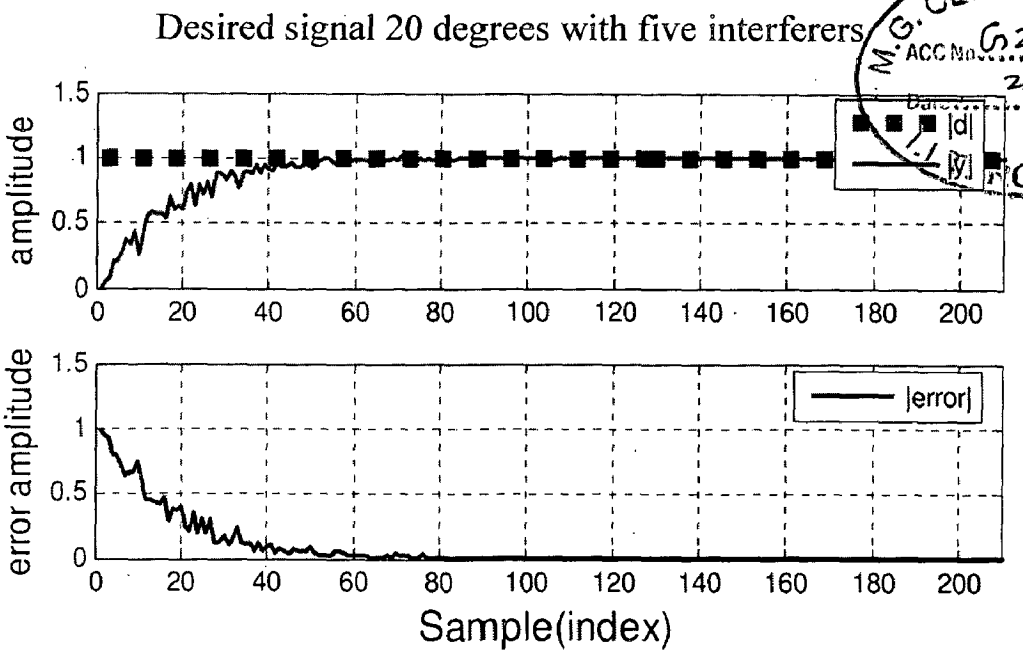


Figure 3.3: LMS algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, and $d = 0.5\lambda$, Amplitude versus time of iteration.

of the desired signal after some samples and it highly minimizes LMS error. In this simulation study we use the system (antenna) noise figure smaller than in case of Figure 3.2. This shows it has a direct effect on the performance of beamforming algorithms and that is discussed under Table 3.1 why we chose it as a simulation parameter.

Figure 3.4 below is the array factor versus angle. From this figure we see that the desired signal is steered at 20 degree and we have five nulls where strong interfering signals steered to be rejected. As shown by the simulation result, the desired signal properly received with array factor of 0dB. The simulation performed assuming one desired signal and 5 interfering signals, this 6 antenna arrays can reject 5 strong interfering signals. This simulation result shows that the LMS algorithm is able to

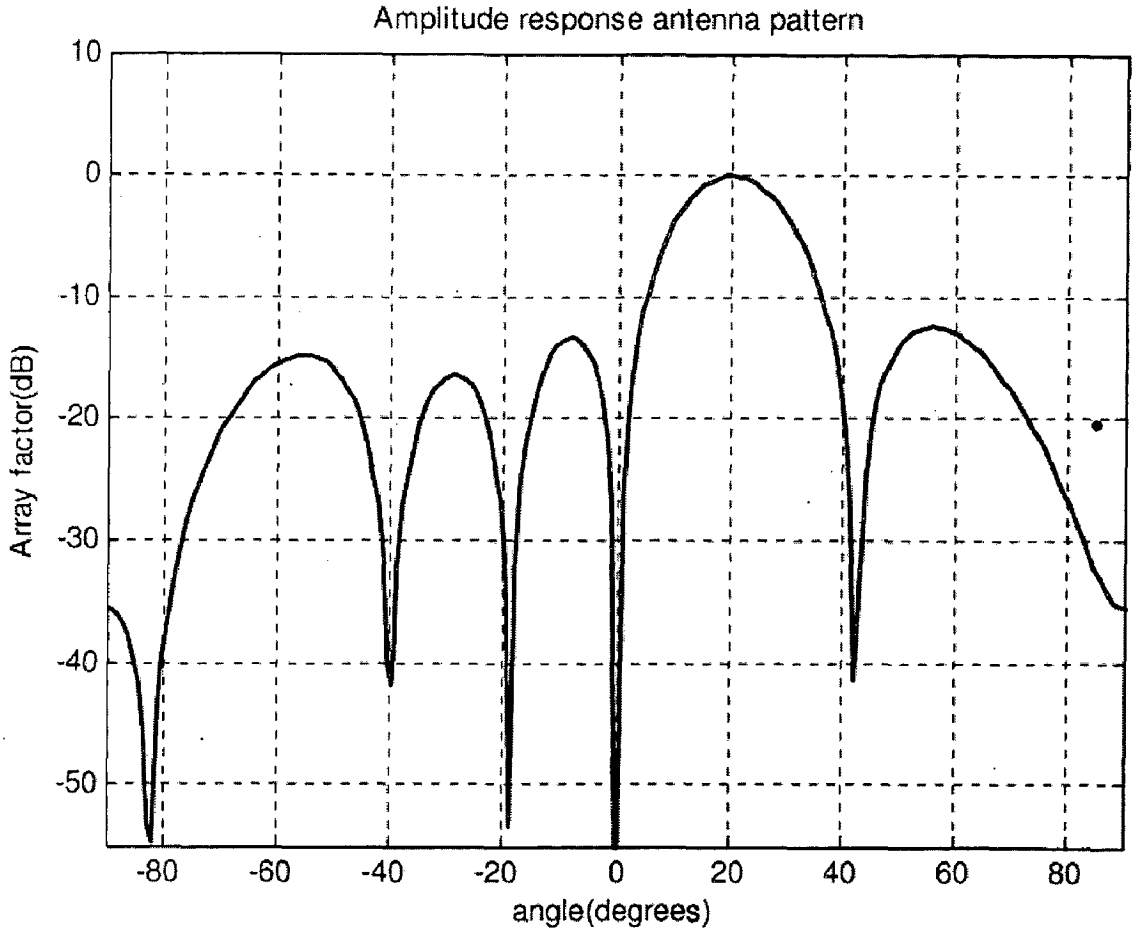


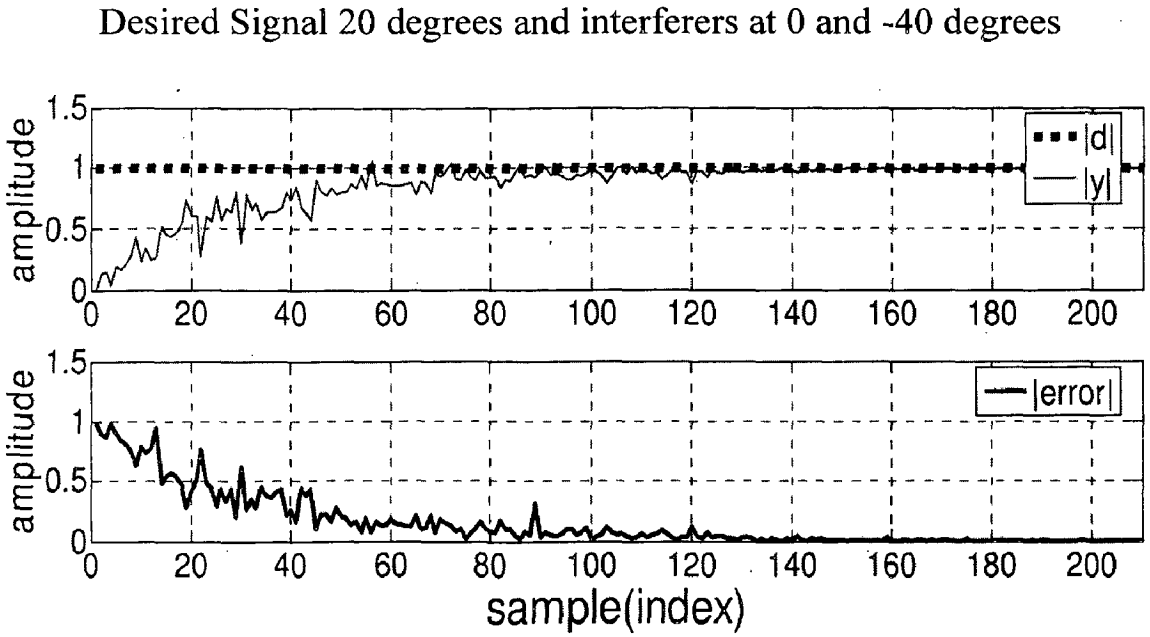
Figure 3.4: Array factor for LMS algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, and $d = 0.5\lambda$.

Array factor (dB) versus theta (degrees)

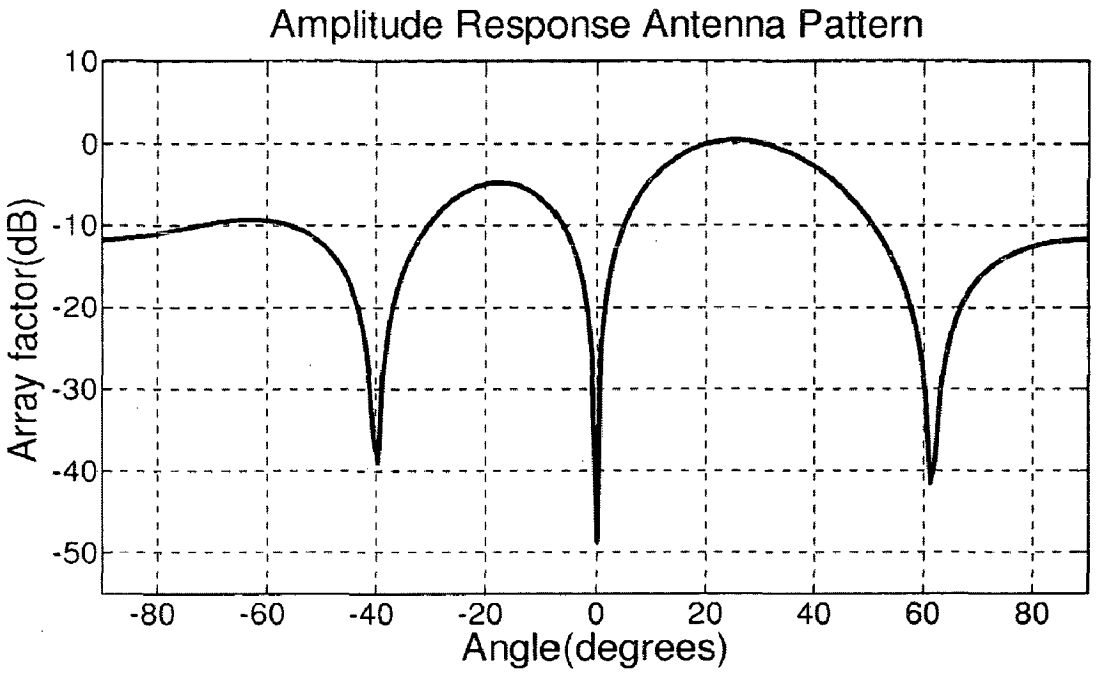
iteratively update the weights to force deep nulls at the direction of the interferers and achieve maximum beam in the direction of the desired signal. The nulls can be seen deep at below -45dB level from the maximum observed at 20 degree.

The simulation result given below under figure 3.5(a) and 3.5(b) is using the same simulating parameter with the result discussed above in figure 3.2 and 3.3, but they are differing in number of antenna used. Here we use 4-element antenna to maximally reject three interfering signals. The weighted signal estimates the desired signal efficiently after 60 samples as given by figure 3.3; whereas the estimation is attained best result after 120 samples when 4 antenna elements are used as shown in figure 3.5(a) below.

Thus, from these two simulation results as the number of antenna element at base station increase, the LMS algorithm for digital beamforming performance increases. In other words, adaptation of the LMS algorithm is very high and converges rapidly. As the number of antenna array element increases the width of the main lobe decreases, the number of the side lobe increases and the number of nulls in the pattern increases. Therefore, using more antenna at base station maximize the performance of digital beamforming by rejecting more interferences which maximize the SINR.



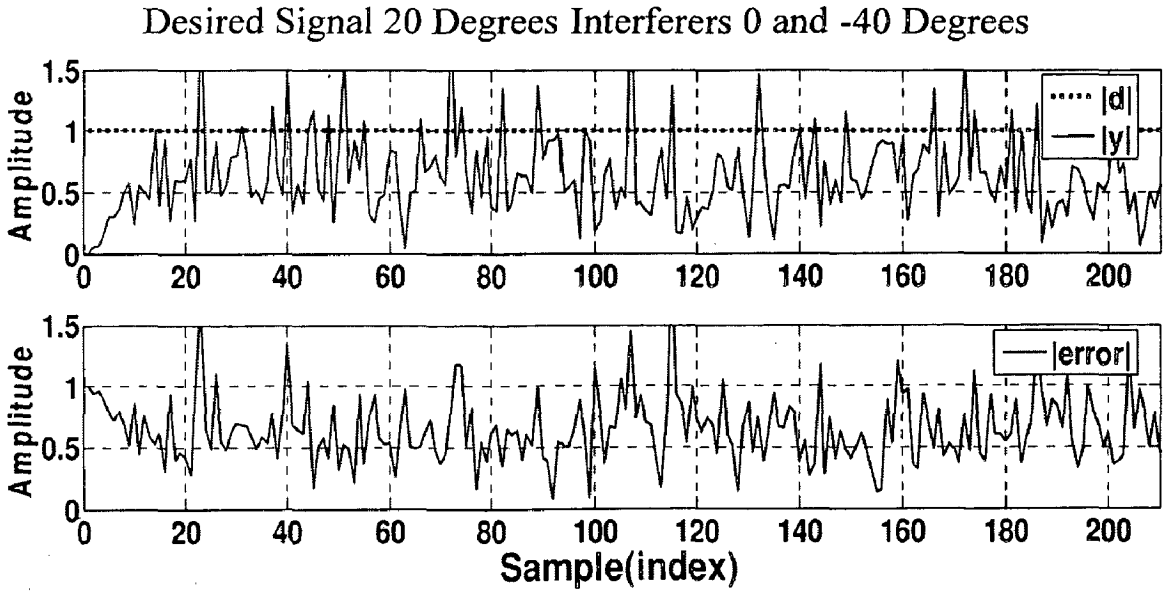
(a) Amplitude versus time of iteration



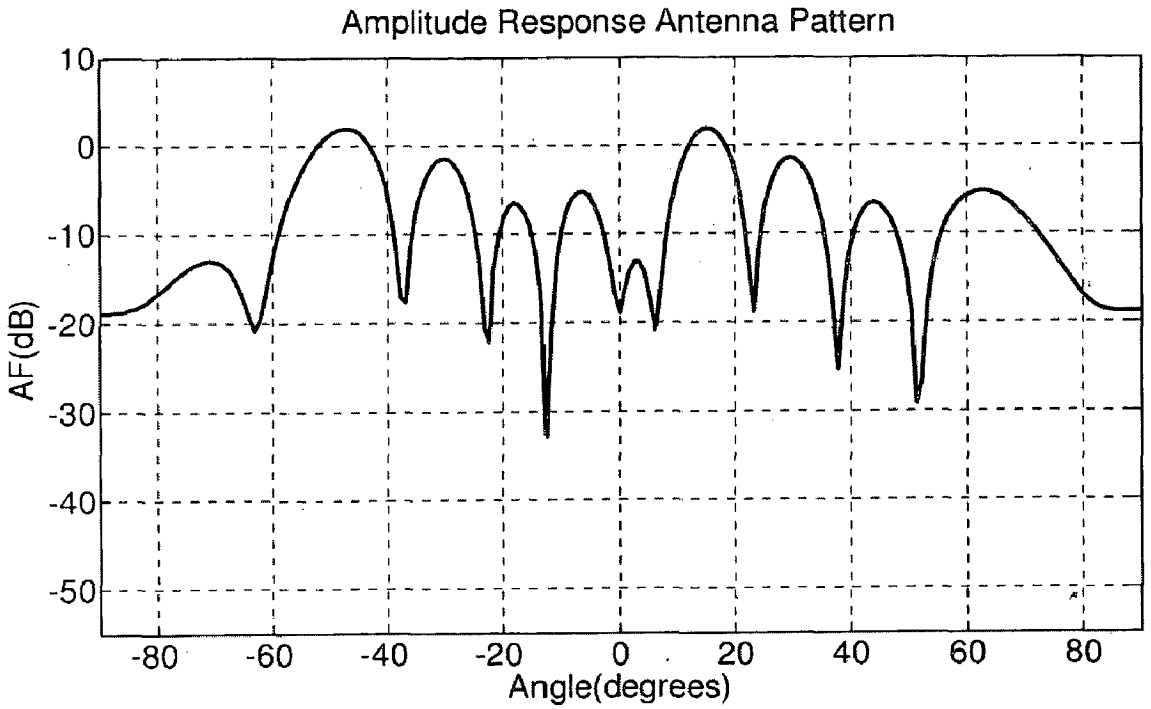
(b) Array factor (dB) versus theta (degrees)

Figure 3.5: LMS algorithm for an adaptive array with 4-element antenna and 3 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, $d = 0.5\lambda$.

From figure 3.6 below, we can see how the inter antenna element spacing affects the performance of the adaptive beamforming algorithms. In this simulation study we assume that the inter-antenna element spacing distance is full wavelength. The LMS algorithm for CDMA system capacity improvement is strongly affected by the antenna array arrangement, from this simulation result the normalized weighted signal magnitude is about 0.5 or half of the desired signal and the overall LMS error obtained averages to over 0.5 or it results in over 50% error. Thus, using this value deteriorate the performance of LMS algorithm as we can see from figure 3.6, this result cannot be accepted. Increasing inter antenna element spacing causes the main lobe in undesired direction, which causes grating lobe.



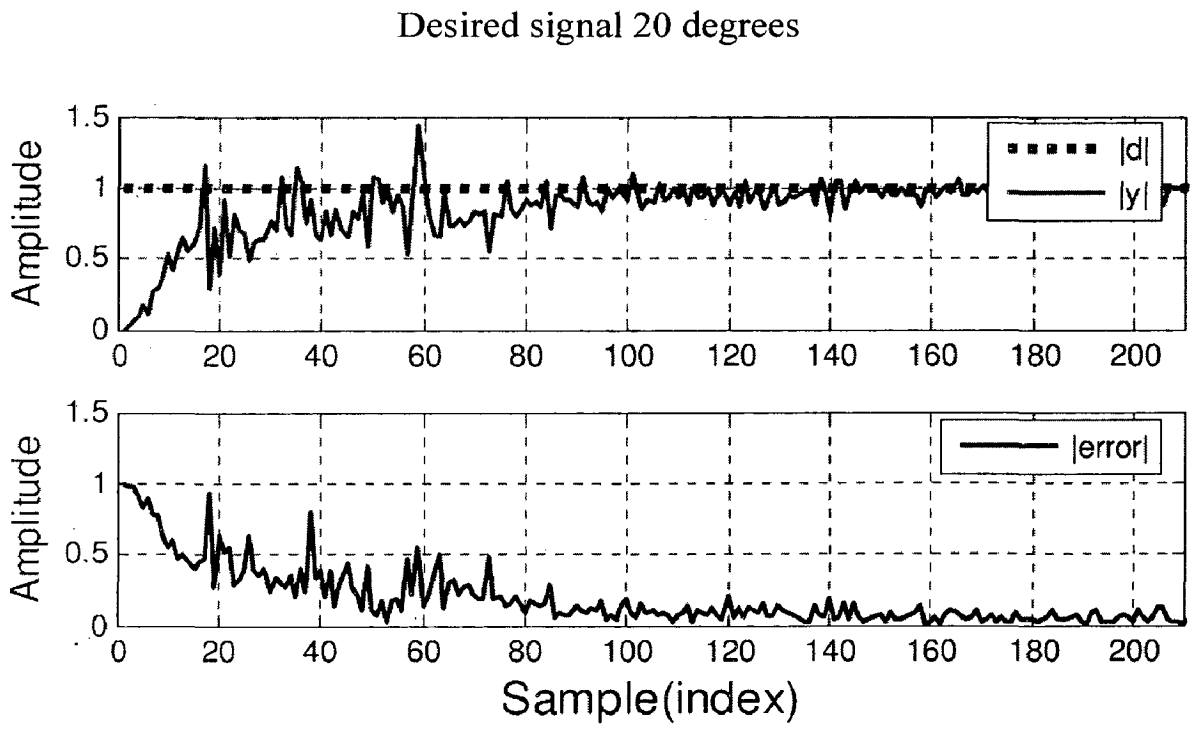
(a) Amplitude versus time of iteration



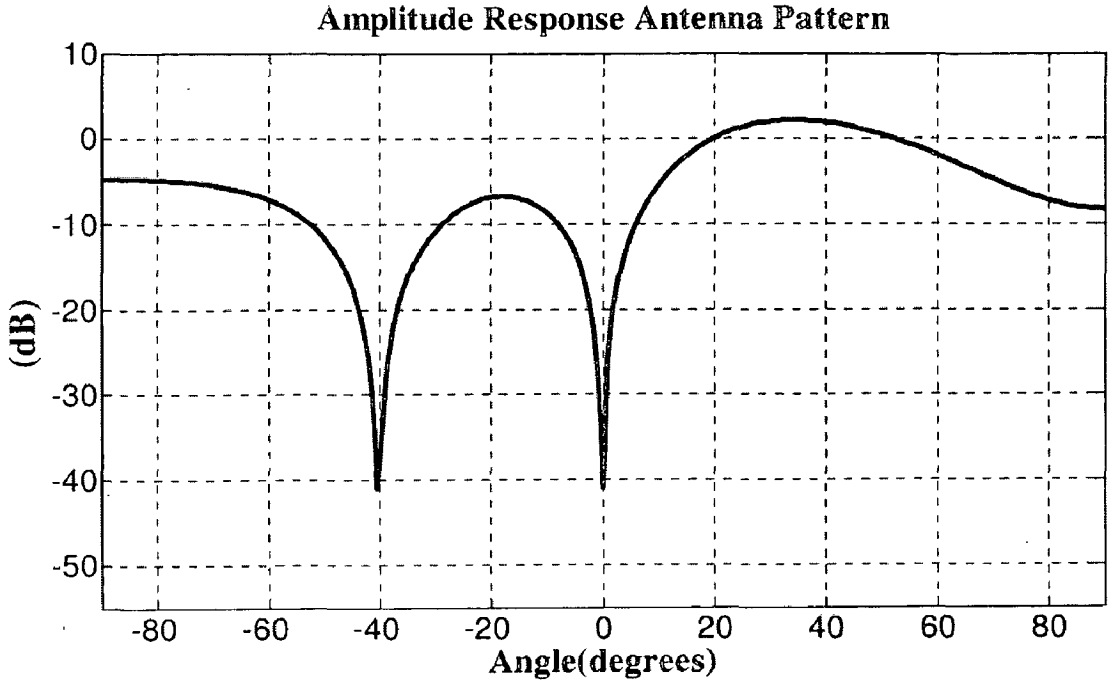
(b) Array factor (dB) versus theta (degrees)

Figure 3.6: LMS algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.1 and $\lambda = d$ and $\mu = 0.01$

From figure 3.7 below the inter antenna element spacing distance is one fourth wavelength (0.25λ). From this simulation result the normalized weighted signal magnitude is very slowly adapt to the desired signal and the LMS error obtained is also deceases slowly. But comparing with the case where inter-antenna element spacing distance is full wavelength (d = λ), the result obtained by this simulation study is more practical and acceptable. Also from the simulation result obtained for the array factor, only two interfering signals would be rejected even though from figure 3.4 using the same numbers of antenna element five strong interfering signals are rejected. As seen from figure 3.7(b) the inter-element spacing decreases the number of rejected interferences is limited so, the system capacities and coverage becomes reduced. Thus using such value deteriorates the performance of beamforming algorithm used to enhance the system performance.



(a) Amplitude versus time of iteration

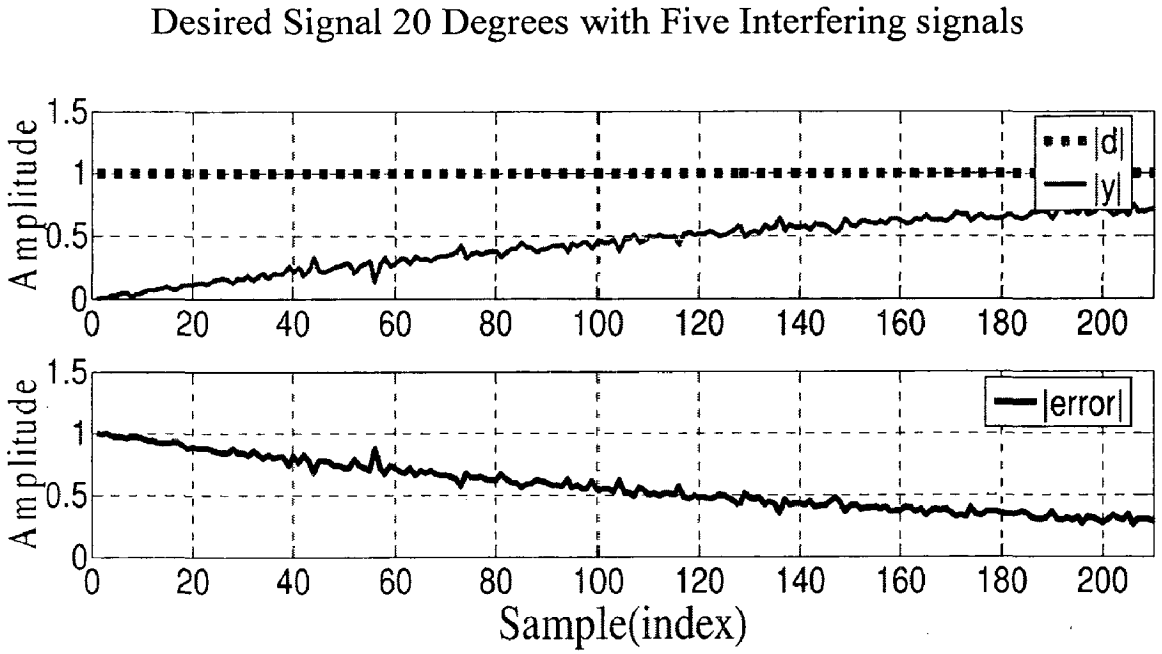


(b) Array factor (dB) versus theta (degrees)

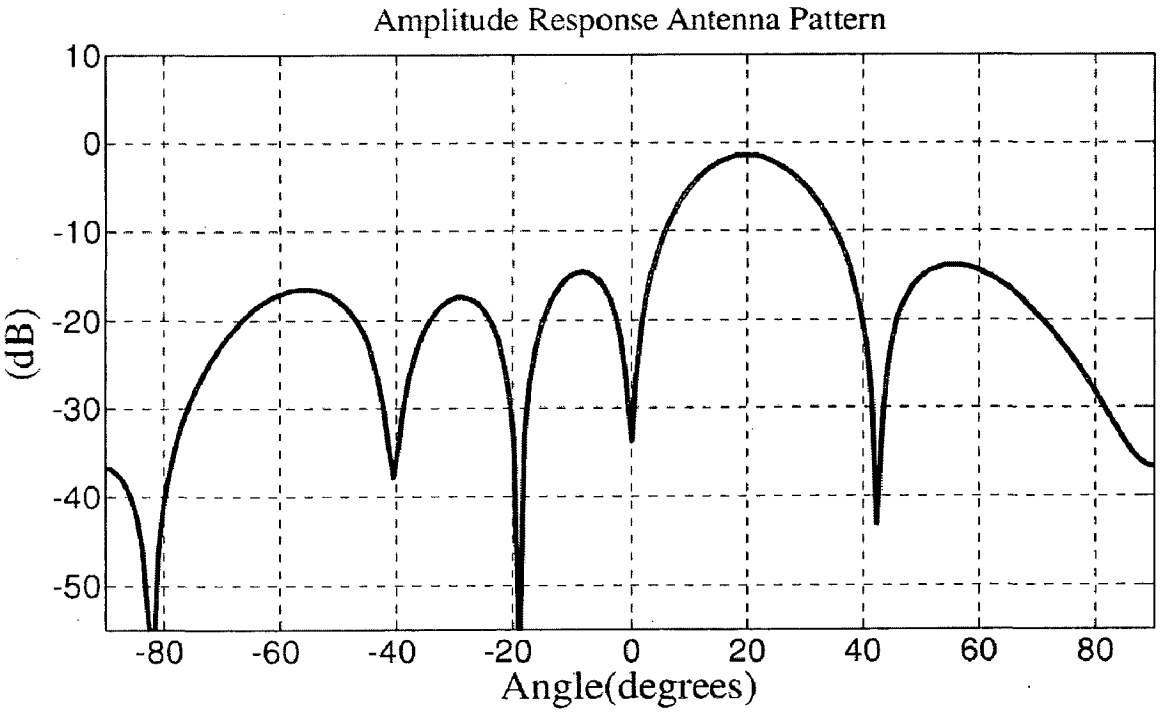
Figure 3.7: LMS algorithm for an adaptive array with 6-element antenna and 4 interferers for antenna noise figure equal to 0.1 and d = 0.25λ and μ = 0.01.

The simulating parameters implemented for the results given by figure 3.8 below is the same with parameters that result figure 3.3, only differ by the step size μ. Here μ = 0.001, using this value and keeping other parameters the same, we get the results shown in figure 3.8 below. From this simulation result the normalized weighted signal magnitude is very slow to adapt toward the desired signal and the LMS error obtained also very slowly decreases. As one can see over the considered sample the normalized weighted signal magnitude is under 0.6 and the LMS error is above 0.4.

When number of samples (at number of iterating time) is 120 the weighted signal normalized amplitude is 0.5 and LMS error is also 0.5. Based on the simulation result given by figure 3.8 to implement small step size the LMS adaptive beamforming algorithm must require very large number of reference signal, which need more bandwidth. Since the wireless bandwidth is highly limited, it is not economical to use more reference signals as it wastes more scarce resources. Thus using such value deteriorates the performance of beamforming algorithm used to enhance the system performance.



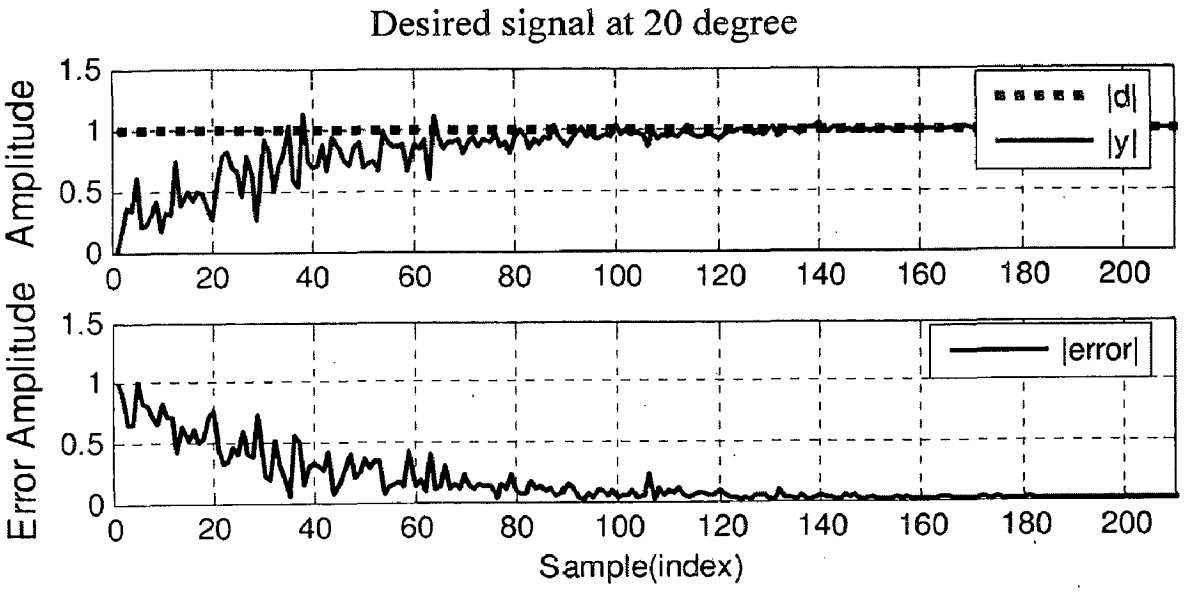
(a) Amplitude versus time of iteration



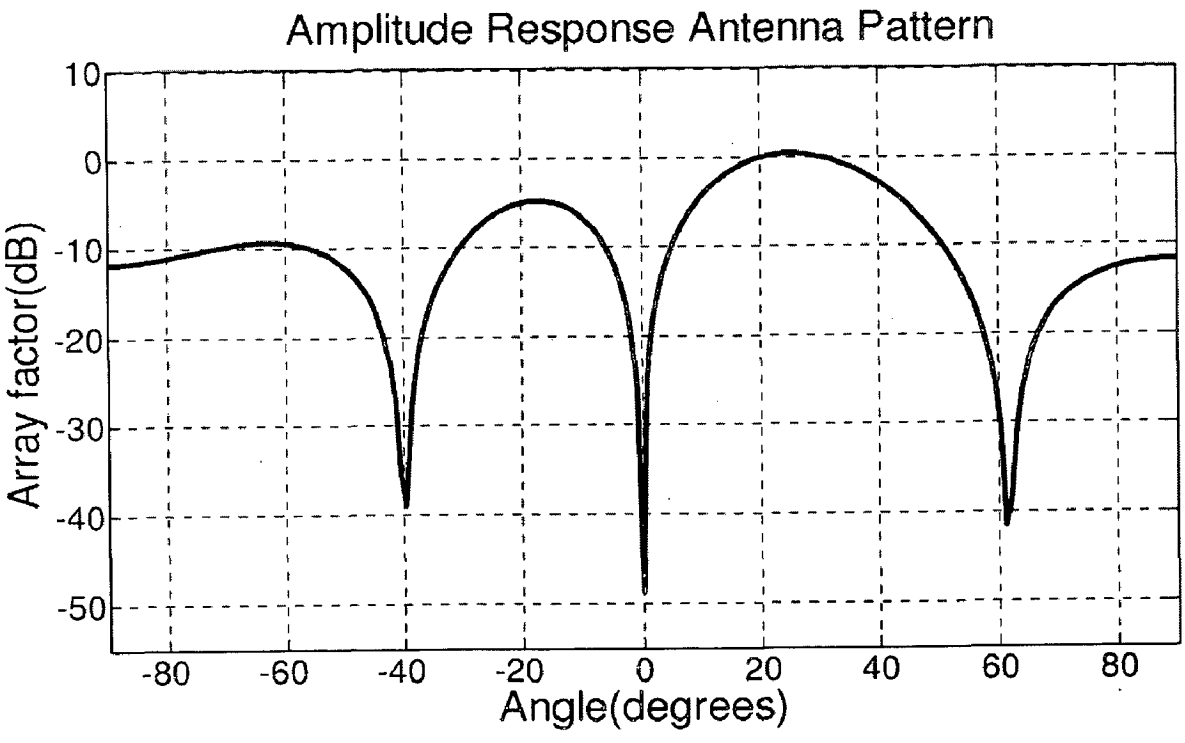
(b) Array factor (dB) versus theta (degrees)

Figure 3.8: LMS algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.01 and 0.5 $\lambda = d$ and $\mu = 0.001$.

The simulation result given by figure 3.9 below uses the same simulation parameters as



(a) Amplitude versus time of iteration



(b) Array factor (dB) versus theta (degrees)

Figure 3.9: LMS algorithm for an adaptive array with 4-element antenna and 2 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, $d = 0.5\lambda$.

in figure 3.3, but they only differ in the number of antenna elements and assumed number of interfering signals. In this simulation study, 4-element antenna arrays and 2 interfering signals are considered. The maximum interference that can be rejected using 4 antenna elements is 3. From the results obtained, we can see that the normalized weighted amplitude is slow to adapt to the desire signal and the LMS error is decreasing slowly. The result obtained from this simulation for 140 samples are relatively the same as the result obtained from figure 3.3 after 60 samples.

3.3.2 CMA for Adaptive Beamforming

In reality, an exact measurement of the gradient vector is not possible, since this would require a prior knowledge of $E[|S(k)|^p]$. The strategy is to scale $|S(k)|$ to unity and to use the instantaneous estimate discussed in sub-section 2.8.2.1. The algorithm is simulated in MATLAB and the result of the simulation can be found in figure 3.10 and 3.11 below. The algorithm converges slower than the LMS algorithm, as we can observe from figure 3.3 and figure 3.10 and figure 3.2 and figure 3.11. The simulation was done with a relatively low system noise. The interferers are the same as in the LMS experiment with the same angle of arrival of the signals.

The signals of the interferers arrive at an angle of –10, –40 and 58 degrees. The signal to be received arrives at an angle of 20 degrees as shown in figure 3.12 below. Figure 3.12 shows the amplitude response of the adaptive array factor, where three interferers are rejected. The results demonstrate the concept of the CM algorithm. From the comparison of these two algorithms we observe a difference in initialization of weighting vectors. During the efforts to simulate the CM algorithm it was clear that the algorithm is less stable than the LMS algorithm. Simulations of the algorithm using the error defined in (2.33) have not resulted in stable results. The advantage of the CM algorithm is the fact that it only needs the instantaneous amplitude of the array output $|y(k)|$ and therefore no synchronization is required whereas LMS requires synchronization as discussed in the previous section. Due to this property, the CM algorithm is relatively simple to implement.

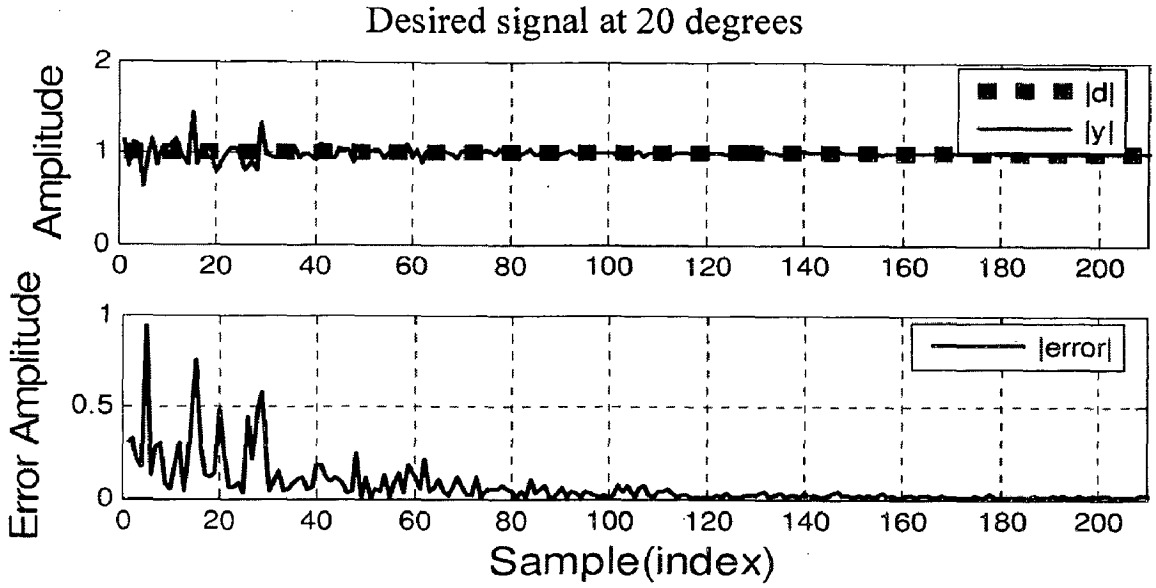


Figure 3.10: CMA algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, $d = 0.5\lambda$. Amplitude versus time of iteration

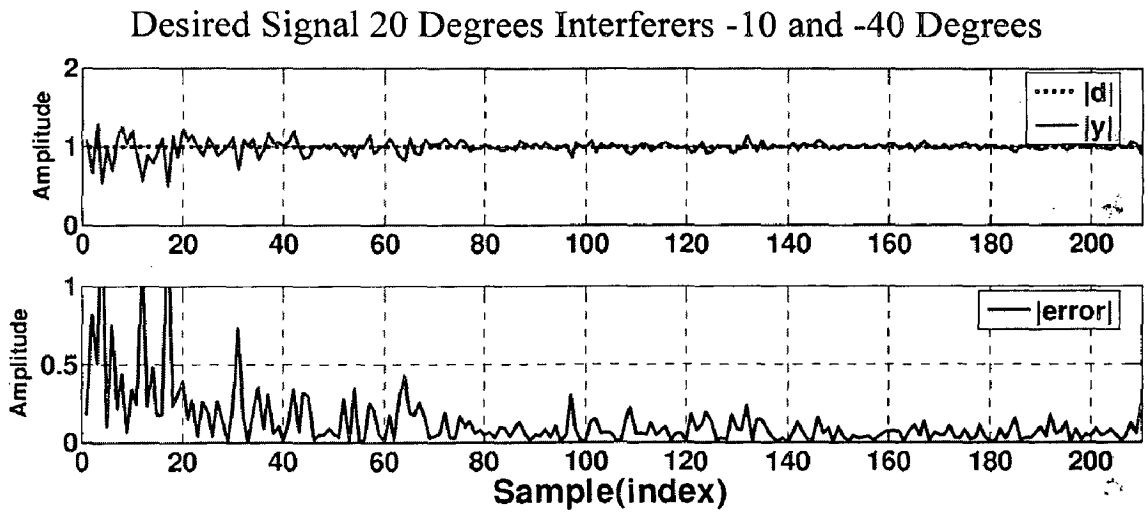


Figure 3.11 CMA algorithm for an adaptive array with 6-element antenna and 5 interferers for antenna noise figure equal to 0.1, $\mu = 0.01$, $d = 0.5\lambda$ Amplitude versus time of iteration

Figure 3.13(a) below is the simulation result obtained for CMA algorithm using the same parameters as the parameters used for simulating the figure 3.10 only differing in the inter-antenna spacing distance. Comparing the error magnitude for the two cases, we can realize that the change in inter-antenna element spacing affects the performance

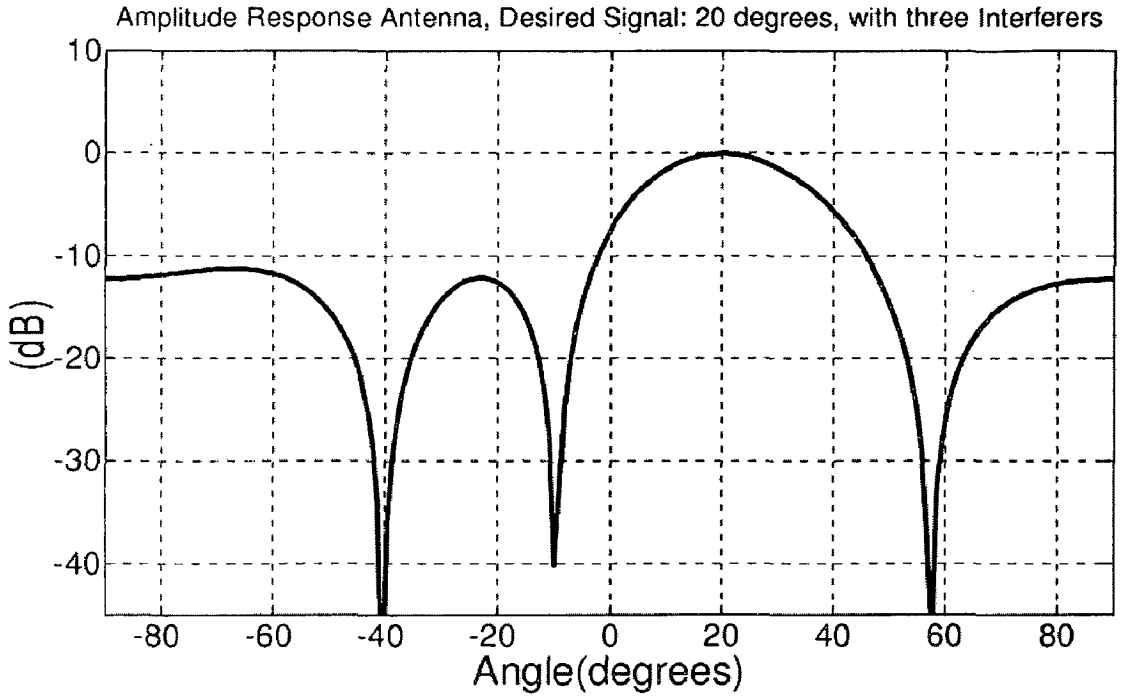
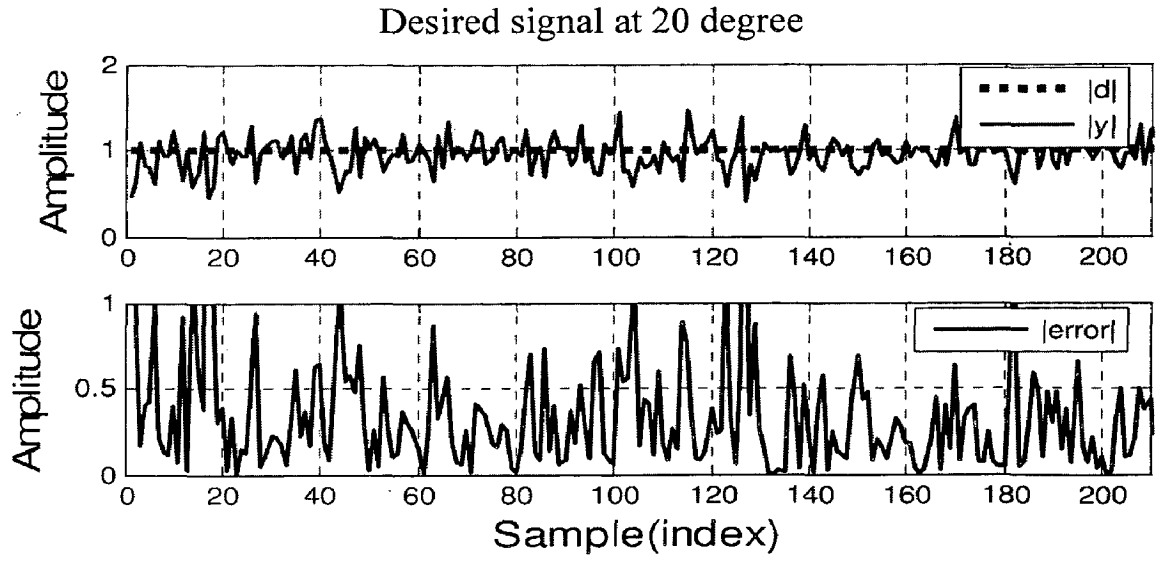
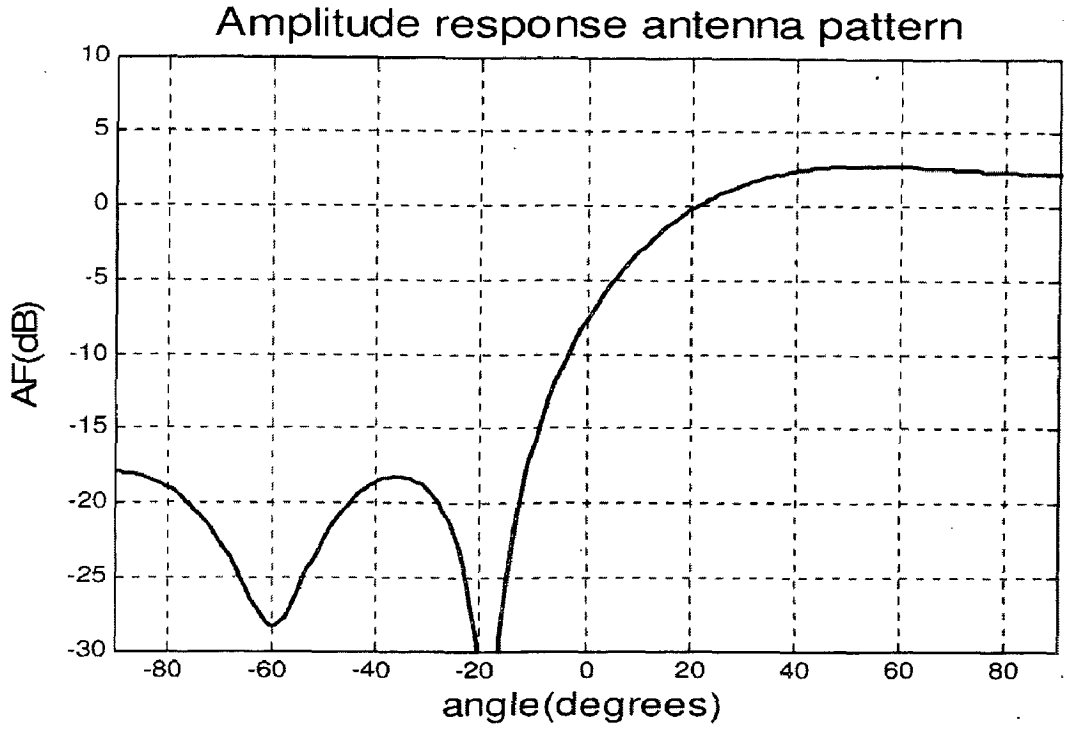


Figure 3.12: CMA algorithm for an adaptive array with 4-element antennas and 3 interferers for antenna noise figure equal to 0.01, $\mu = 0.01$, $d = 0.5\lambda$. Array factor (dB) versus theta (degrees)

of the CMA algorithm for digital beamforming to enhance system capacity. As explained above the inter element antenna spacing $d = 0.25\lambda$ is not suitable distance to implement this algorithm and it cannot properly reject the interferences.



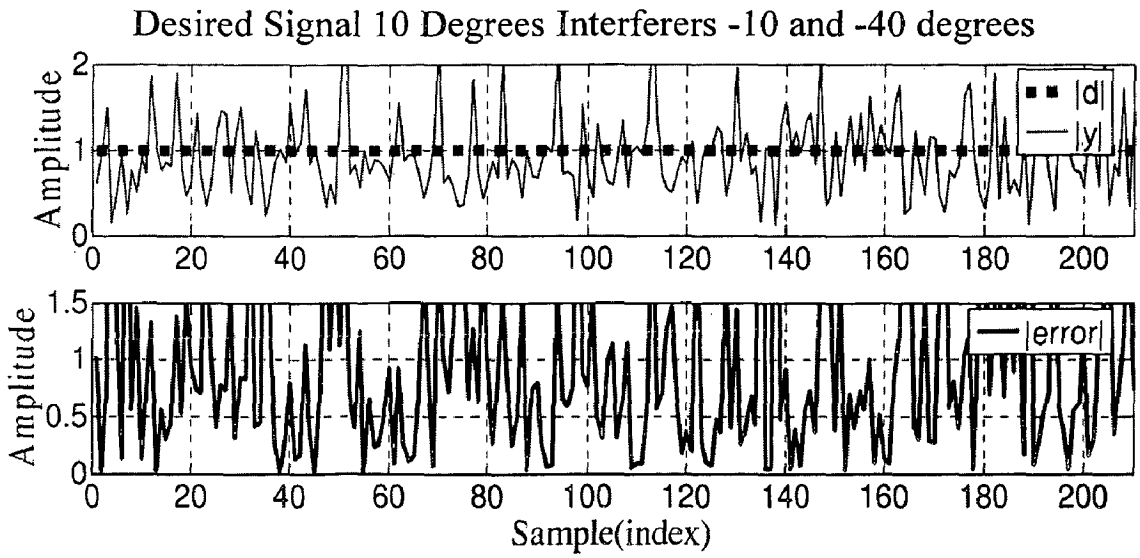
(a) Amplitude response versus time of iteration



(b)Array factor (dB) versus theta (degrees)

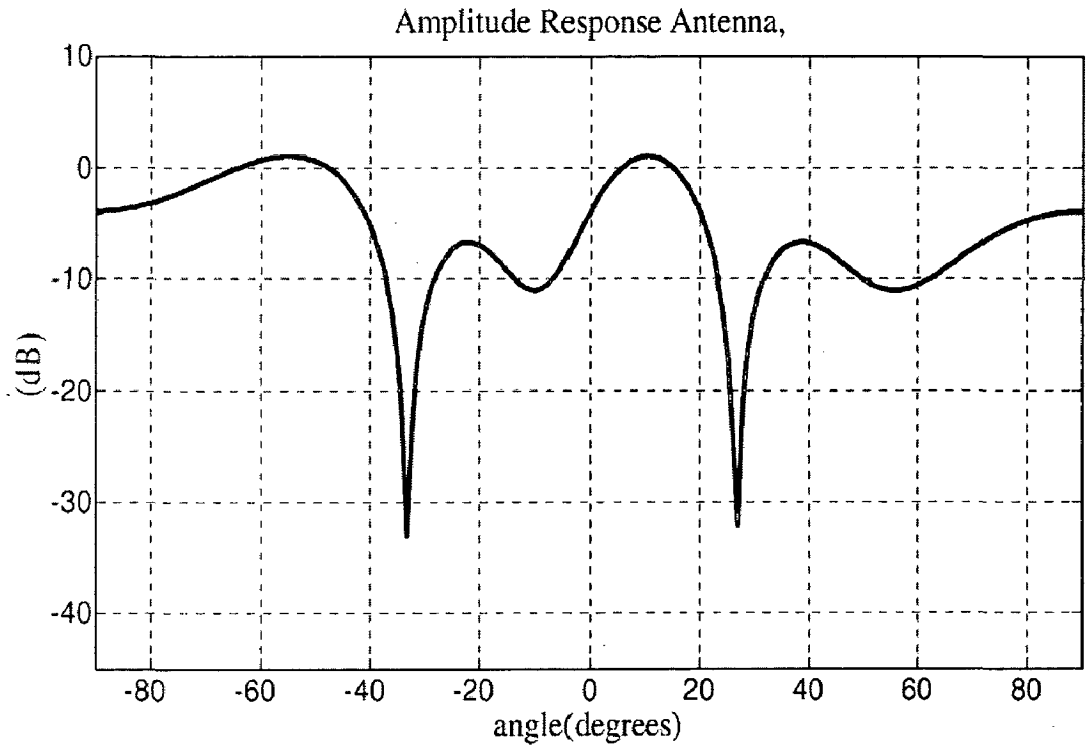
Figure 3.13: CMA algorithms for an adaptive array with 4-element antenna and 2 interferers for antenna noise figure equal to 0.01 and d = 0.25λ and μ = 0.01

From figure 3.14(a) and (b) shown below, we see that the CMA for digital beamforming is affected by the magnitude of inter-antenna element spacing comparing



(a) Amplitude versus time of iteration

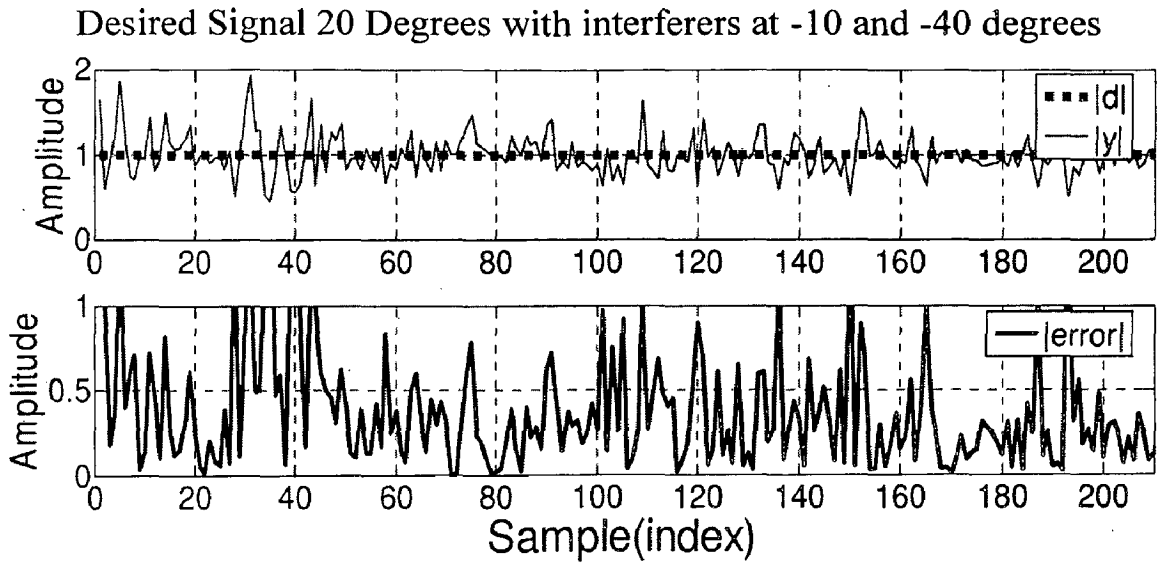
with figure 3.10 give above. Using d = λ to enhance the system performance by implementing digital beamforming concepts as observed from figure 3.6, 3.7 for LMS algorithm and figure 3.10 for CMA algorithms. The CMA error magnitude is highly fluctuating randomly and averaged to around 0.75 or 75% and the weighted signal is highly oscillating and become unstable. Therefore, the result obtained using such parameter is not practical as it shifts the desired signal arrival angle and mismatch the nulls with interfering signals as we can see from figure 3.14(b) below.



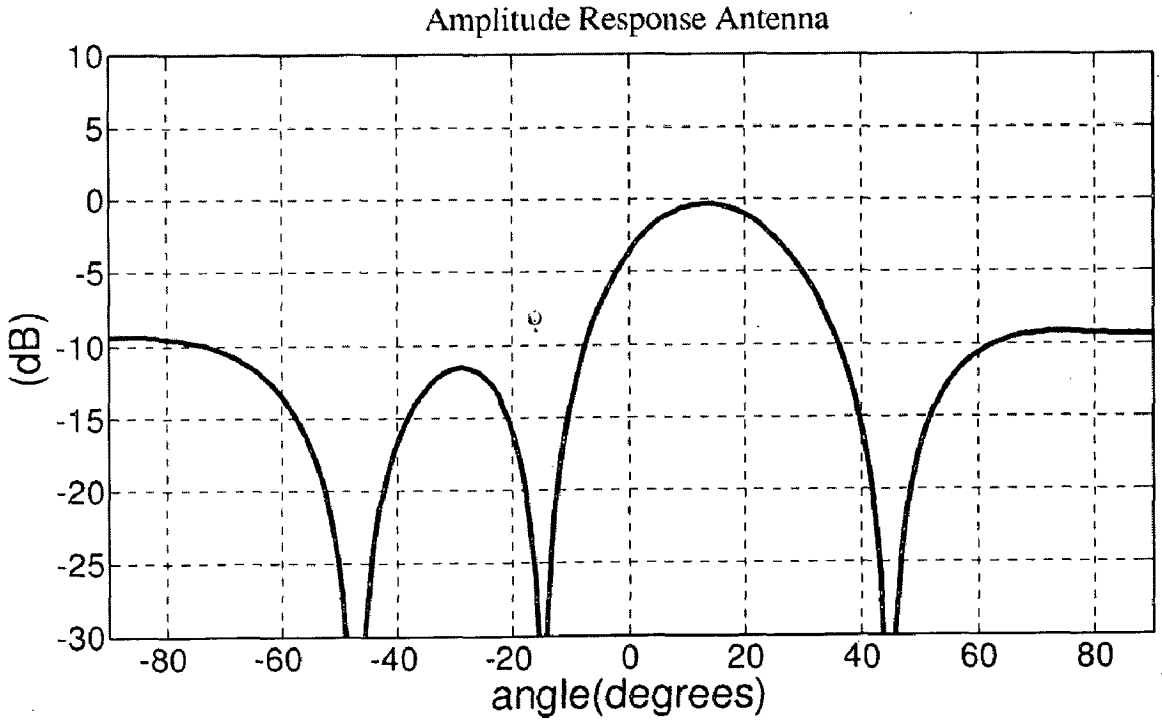
(b) Array factor (dB) versus theta (degrees)

Figure 3.14: CMA algorithm for an adaptive array with 4-element antenna and 2 interferers for antenna noise figure equal to 0.01 and λ = d and μ = 0.01.

As discussed above in figure 3.8 using very small adaptation step size affects the beamforming algorithms performance. From these two simulation result for different adaptive algorithm we observe that in order to adapt to the desired signal the adaptation step size should be reasonable. But comparing to the result obtained from figure 3.8 the result obtained from simulation result given by figure 3.15 are highly unstable and it cannot show any improvement in error amplitude and estimation of desired signal. These indicate that the LMS algorithm converges as the number of sample increases while CMA cannot do this.



(a) Amplitude versus time of iteration



(b) Array factor (dB) versus theta (degrees)

Figure 3.15: CMA algorithm for an adaptive array with 4 antennas and 3 interferers for antenna noise figure equal to 0.01 and 0.5 $\lambda = d$ and $\mu = 0.001$.

Chapter 4

Neural Network based Robust Adaptive Beamforming Algorithm

Neural networks have found numerous applications in the field of signal processing [5, 23], mainly because of their general purpose nature, fast convergence rates, and new VLSI implementations. The aspect of antenna array signal processing focuses on adaptive beamforming. Adaptive beamforming is used for enhancing a desired signal while suppressing noise and interference at the output of an array of sensors. When adaptive arrays are applied to practical problems, the performance degradation of adaptive beamforming techniques may become even more pronounced than in the ideal case because some of underlying assumptions on the environment, sources, or sensor array can be violated and this may cause a mismatch between the presumed and actual signal steering vectors. To account for the signal steering vector mismatches, additional linear constraints (point and derivative constraints) can be imposed to improve the robustness of adaptive beamforming. But, the beamformers lose degrees of freedom for interference suppression. Diagonal loading has been a popular approach to improve the robustness of adaptive beamforming algorithms. However, a serious drawback of the approach is that there is no reliable way to choose the diagonal loading factor.

Neural network methods possess such advantages as general purpose nature, nonlinear property, passive parallelism, adaptive learning capability, generalization capability and fast convergence rates. Neural network method is typically used in two steps: training phase and performance phase. Neural network is first trained with known input/output

pattern pairs. It can be implemented off-line, although a large training pattern set is required for network training. After the training phase, it can be used directly to replace the complex system dynamics. By these inherent advantages of the neural network, this thesis presents the development of a neural network-based robust adaptive beamforming algorithm, which treats the problem of computing the weights of an adaptive array antenna as a mapping problem.

4.1 Mathematical Model

Consider a uniform linear array (ULA) with M omni-directional sensors spaced by the distance d and D narrow-band incoherent plane waves, impinging from directions $\{\theta_1, \theta_2, \ldots, \theta_{D-1}\}$.

The observation vector is given by

$$\begin{aligned} X(k) &= s(k) + i(k) + n(k) \\ &= s_0(k)a + i(k) + n(k) \end{aligned} \tag{4.1}$$

where $X(k)$ is the complex vector of array observations and it expressed as

$$X(k) = [x_1(k), x_2(k), \ldots \ldots x_M(k)]^T \tag{4.2}$$

$s_0(k)$ = the signal waveform, a is the signal steering vector,

$i(k)$ is the interference component, $n(k)$ is the noise component.

The output of a narrowband beamformer is

$$y(k) = w^H X(k) \tag{4.3}$$

where w is the complex vector of beamformer weight and it expressed as

$$w = [w_1, w_2, \ldots \ldots, w_M]^T \tag{4.4}$$

The signal to interference plus noise ratio (SINR) has the following form

$$SINR = \frac{w^H R_s w}{w^H R_{i+n} w} \tag{4.5}$$

where R_s is M x X signal matrix that is statistical expectation of signal vector and it is

$$R_s = E\{s(k)s^H(k)\} \tag{4.6}$$

and R_{i+n} is signal plus noise covariance matrix as

$$R_{i+n} = E\{(i(k) + n(k))(i(k) + n(k))^H\} \tag{4.7}$$

The adaptive beamformer weight vector is computed in order to optimize the performance in terms of a certain criterion. Although several criteria can be used, we limit our consideration by the output SINR criterion, which is rewritten as

$$SINR = \frac{\sigma_s^2 |w^H a|^2}{w^H R_{i+n} w} \tag{4.8}$$

where σ_s^2 is the signal power.

The problem of finding the maximum of (4.8) is equivalent to the following optimization problem

$$\min w^H R_{i+n} w \text{ subject to } w^H a = 1 \tag{4.9}$$

From (4.9), the following solution can be found for the optimal weight vector

$$w_{opt} = \frac{R_{i+n}^{-1} a}{a^H R_{i+n}^{-1} a} \tag{4.10}$$

Inserting (4.10) into (4.8), we obtain that the optimal SINR is given as

$$SINR_{opt} = \sigma_s^2 a^H R_{i+n}^{-1} a \tag{4.11}$$

where equation (4.11) gives an upper bound on the output SINR (4.8).

4.1.1 Sample Matrix Inversion (SMI) Algorithm

The sample matrix is a time average estimate of array correlation matrix using N-time samples. If random process is ergodic in the correlation, the time average estimate will equal the actual correlation matrix. In this method we use N–length block of data. In practical applications, the exact interference-plus-noise covariance matrix R_{i+n} is unavailable. Therefore, the sample covariance matrix is used instead of R_{i+n}.

$$\hat{R} = \frac{1}{N} \sum_{i=1}^{N} X(i) X^H(i) \tag{4.12}$$

where N is the number of snapshots available.

Thus weight of SMI algorithm is

$$w_{SMI} = \alpha \hat{R}^{-1} a \tag{4.13}$$

where $\alpha = a^H R^{-1} a$ is the normalization constant that does not affect the output SINR.

The SMI algorithm is very sensitive to the mismatch between the presumed and actual spatial signature vectors.

4.1.2 Loaded Sample Matrix Inversion (LSMI) Algorithm

One of the most popular robust approaches is the loaded SMI (LSMI) algorithm, which attempts to improve the robustness of the SMI technique against an arbitrary spatial signature mismatch by means of diagonal loading of the sample covariance matrix [20]. The essence of LSMI algorithm is to replace the conventional sample covariance matrix R by the so-called diagonally loaded covariance matrix.

$$\hat{R}_{dl} = \hat{R} + \xi I \tag{4.14}$$

where ξ is a diagonal loading factor. So that, we can write the LSMI weight vector in the following form

$$w_{LSMI} = \hat{R}_{dl}^{-1} a = (\hat{R} + \xi I)^{-1} a \tag{4.15}$$

So the LSMI algorithm can improve the performance of SMI algorithm in scenarios with an arbitrary steering vector mismatch, this improvement is not significant because LSMI algorithm exploits the presumed steering vector and, therefore, its performance degrades when the norm of the error vector is large. Furthermore, the proper choice of ξ represents a serious problem in practical applications because ξ depends on the unknown signal and interference parameters.

4.1.3 Robust Adaptive Beamforming

We assume that the norm of the steering vector distortion a_e can be bounded by some known constant ε^2

$$\|a_e\|^2 \leq \epsilon^2 \tag{4.16}$$

Then, the actual signal steering vector

$$\tilde{a} = a_e + \bar{a} \tag{4.17}$$

where $\bar{a}$ is the assumed steering vector.

Cost function of robust adaptive beamforming algorithm minimizes the mean output power subject to the inequality constraint. Thereby, the optimization problem can be formulated as

$$\min(a_e + \bar{a})^H R^{-1}(a_e + \bar{a}) \text{ subject to } \|a_e\|^2 \leq \epsilon^2 \tag{4.18}$$

The solution to (18) can be obtained using Lagrange multiplier method by minimizing the function

$$H = (a_e + \bar{a})^H R^{-1}(a_e + \bar{a}) + \lambda(a_e^H a_e - \epsilon^2) \tag{4.19}$$

where λ is Lagrange multiplier.

For finding the norm of steering vector computing this gradient of (4.19) and equating it to zero yields

$$a_e = -\left(\hat{R}^{-1} + \lambda I\right)^{-1} \hat{R}^{-1} \bar{a} \tag{4.20}$$

So by equations (4.18) and (4.20), we get

$$\bar{a}^H \hat{R}^{-1} \left(\hat{R}^{-1} + \lambda I\right)^{-2} \hat{R}^{-1} \bar{a} = \varepsilon^2 \tag{4.21}$$

The covariance matrix decompose into Eigen value and eigenvector form as

$$\hat{R} = U \Lambda U^H \tag{4.22}$$

Where columns of U are the eigenvectors and diagonal elements of Λ are known values of R.

Then inserting (4.22) into (4.21), we can obtain

$$\bar{a}^H U \Lambda^{-1} (\Lambda^{-1} + \lambda I)^{-2} \Lambda^{-1} U^H \bar{a} = \varepsilon^2 \tag{4.23}$$

Let $F = U^H \bar{a}$ and above equation can be simplified as

$$f(\lambda) = \sum_{i=1}^{M} \frac{|F_i|^2}{(1+\lambda\gamma_i)^2} = \varepsilon^2 \tag{4.24}$$

Left side of (4.24) is a monotonically decreasing function of Λ, and we can obtain a unique solution $\lambda > 0$. And hence λ can be obtained efficiently by Newton's method [7].

From (4.24), we have

$$\sum_{i=1}^{M} \frac{|F_i|^2}{(1+\lambda\gamma_i)^2} = \varepsilon^2 < \sum_{i=1}^{M} \frac{|F_i|^2}{(\lambda\gamma_i)^2} \tag{4.25}$$

This gives the upper bound on λ

$$\lambda < \frac{1}{\varepsilon}\left(\sum_{i=1}^{M} \frac{|F_i|^2}{\gamma_i{}^2}\right)^{\frac{1}{2}} \tag{4.26}$$

By replacing the γ_I in (4.24) with γ_1 and γ_M respectively, we get

$$\frac{\|\bar{a}\| - \varepsilon}{\gamma_1 \varepsilon} \leq \lambda \leq \frac{\|\bar{a}\| - \varepsilon}{\gamma_M \varepsilon} \tag{4.27}$$

We can combine (4.26) and (4.27) to give the following upper and lower bounds on the solution of λ

$$\frac{\|\bar{a}\| - \varepsilon}{\gamma_1 \varepsilon} \leq \lambda \leq \min\left\{\frac{\|\bar{a}\| - \varepsilon}{\gamma_M \varepsilon}, \frac{1}{\varepsilon}\left(\sum_{i=1}^{M} \frac{|F_i|^2}{\gamma_i{}^2}\right)^{\frac{1}{2}}\right\} \tag{4.28}$$

Solving (4.22) for λ by a Newton's method using that the solution is unique and it follows the above condition. Thus the weight vector for RAB written as

$$W_{RAB} = \frac{\hat{R}^{-1}((\lambda\hat{R}+I)^{-1} - I)\bar{a}}{\bar{a}^H \hat{R}^{-1}((\lambda\hat{R}+I)^{-1} - I)^2 \bar{a}} \tag{4.29}$$

$$= \frac{U\lambda^{-1}((\lambda\Lambda+I)^{-1} - I)U^H \bar{a}}{\bar{a}^H U\Lambda^{-1}((\lambda\Lambda+I)^{-1} - I)^2 U^H \bar{a}} \tag{4.30}$$

4.2 Radial Basis Function Neural Network (RBFNN)

The weight vector of the above algorithm is a nonlinear function of the sample covariance matrix, and is not suitable for real-time implementation. Therefore, it can be approximated using a suitable architecture such as RBFNN in this dissertation. The array outputs are preprocessed, and then applied to the RBFNN. The sample covariance

matrix $\widehat{\boldsymbol{R}}$ is presented to the input layer of the RBFNN, and the vector w_{RAB} is produced at the output layer. As it is the case, with most neural network, the RBFNN is designed to perform an input-output mapping, trained with examples $(\widehat{\boldsymbol{R}}; \boldsymbol{w}_{\mathrm{RAB}})$, $l = 1, 2, \ldots, N_T$, where N_T stands for the number of examples contained in the training set.

4.2.1 Radial Basis Function

Radial Basis Functions emerged as a variant of artificial neural network in late 80s. However, their roots are entrenched in much older pattern recognition techniques as for example potential functions, clustering, functional approximation, and spline interpolation and mixture models. The RBF originated in the study for the interpolation problems of multi-variable and is still a main research area in numeric analysis. From other standpoint, the design of a neural network can also be viewed as a surface fitting (reconstruction) problem in a hyperspace where the RBF method is a nature choice. As one of the most popular neural network models, RBF network has attracted lots of attentions on the improvement of its approximation as well as the construction of its architecture. RBF's are embedded into a two-layer feed forward neural network. Such a network is characterized by a set of inputs and a set of outputs. In between the inputs and outputs there is a layer of processing units called hidden units. Each of them implements a radial basis function. The output units implement a weighted sum of hidden units outputs. The input into a RBF network is non-linear while the output is linear. Due to their nonlinear approximation properties, RBF networks are able to model complex mapping, while perceptron neural networks can only model by means of multiple intermediary layers.

In order to use a radial Basis function Network we need to specify the hidden unit activation function, the number of processing units, a criterion for modeling a given task and a training algorithm for finding the parameters of the network. Finding the RBF weights is called network training. If we have at hand a set of input-output pairs, called training set, we optimize the network parameters in order to fit the network outputs to the given inputs. The fit is evaluated by means of a cost function, usually assumed to be the mean square error. After training, the RBF network can be used with data whose underlying statistics is similar to that of training set.

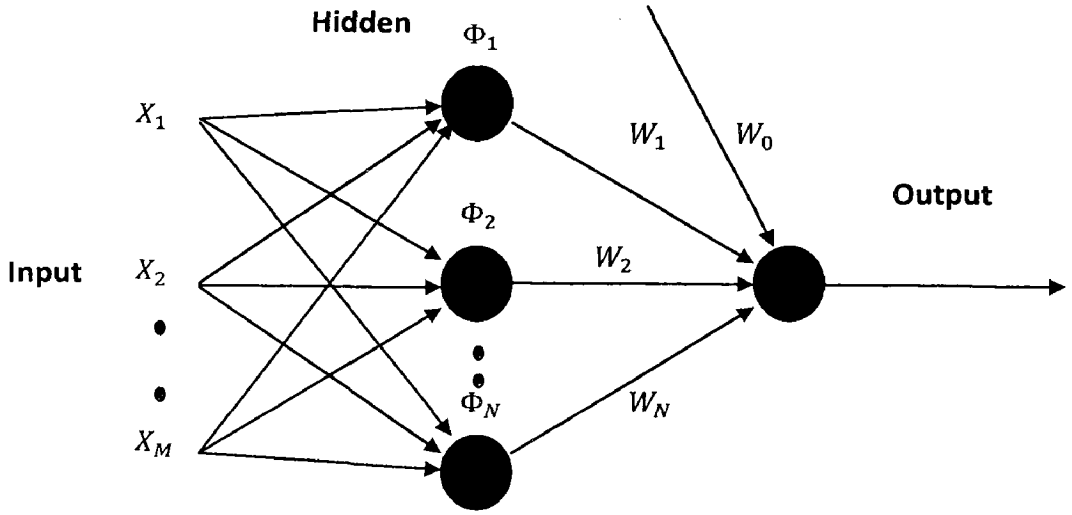


Figure 4.1: Structure of RBF Neural Network

4.2.1.1 Network Topology

Basic principle of the RBF method is detailed in the remarkable literature of Haykin [23]. The construction of a RBF network, in its most basic form, involves three layers with entirely different roles. The input layer is made up of source nodes (sensory units) that connect the network to its environment. The second layer, the only hidden layer in the network, applies a nonlinear transformation from the input space to the hidden space; in most applications the hidden space is of high dimensionality. The output layer is linear, supplying the response of the network to the activation pattern (signal) applied to the input layer. The way in which the network is used for data modeling is different when approximating time-series and in pattern classification. In the first case, the network inputs represent data samples at certain past time-laps, while the network has only one output representing a signal value. In the pattern classification applications the inputs represent feature entries, while each output corresponds to a class. Generally, for given set of different points x, RBF technique uses a function $F^*(x)$ of the following form

$$F^*(x) = \sum_{i=1}^{m_1} w_i \, \varphi_i(x) \tag{4.31}$$

where, $\varphi_i(x|I = 1, 2, \ldots, m_1)$ is a new set of basis functions that we assume to be linearly independent without loss of generality, $G(x, t_i)$ is a Green function centered at

t_i, w_i constitute a new set of weights, and m_1 is the number of centers (or the size of the hidden layer). Typically, the number of basis functions is less than the number of data points (i.e., $m_1 \leq N$). A commonly used Green function is the multivariate Gaussian function.

$$G(x, t_i) = exp\left(-\frac{1}{2\sigma_i^2}\|x - t_i\|^2\right) \tag{4.32}$$

where ||.|| denotes a norm that is usually Euclidean.

4.2.1.2 Learning Strategies

There are different strategies that are used in the design of an RBF network, depending on how the centers of the redial basis functions of the network are specified. These design strategies pertain to an RBF network whose formulation is based on interpolation theory. Here we used Supervised Selection of Centers as a learning strategy.

In this approach, the centers of the radial basis functions and all other free parameters of the network undergo a supervised learning process; in other words, the RBF network takes on its most generalized form. A natural candidate for such a process is error-correlation learning, which is most conveniently implemented using a gradient descent procedure that represents a generalization of the LMS algorithm.

The first step in the development of such a learning procedure is to define the instantaneous value of the cost function

$$\xi = \frac{1}{2}\sum_{j=1}^{N} e_j^2 \tag{4.33}$$

where N is the size of the training sample used to do the learning, and e_j is the error signal defined by

$$e_j = d_j - F^*(x_j) \tag{4.34}$$

$$= d_j - \sum_{i=1}^{m_1} G(\|x_j - t_i\|^2)_{C_i} \tag{4.35}$$

The requirement is to find the free parameters w_i, t_i and $\sum_t^{-1}()$ the latter being related to the norm-weighting matrix C_i) so as to minimize ξ. The results of this minimization are summarized below:

1. Linear weights (output layer)

$$\frac{\partial \xi(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n)\, G(\|x_j - t_i(n)\|)_{C_i} \tag{4.36}$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial \xi(n)}{\partial w_i(n)}, \qquad i = 1, 2, \ldots, m_1 \tag{4.37}$$

2. Positions of Centers (hidden layer)

$$\frac{\partial \xi(n)}{\partial t_i(n)} = 2\, w_i(n) \sum_{j=1}^{N} e_j(n)\, G'\left(\|x_j - t_i(n)\|\right)_{C_i} \Sigma_i^{-1}[x_j - t_i(n)] \tag{4.38}$$

$$t_i(n+1) = t_i(n) - \eta_2 \frac{\partial \xi(n)}{\partial t_i(n)}, \qquad i = 1, 2, \ldots, m_1 \tag{4.39}$$

3. Spreads of centers (hidden layer)

$$\frac{\partial \xi(n)}{\partial \Sigma_i^{-1}(n)} = -w_i(n) \sum_{j=1}^{N} e_j(n)\, G'\left(\|x_j - t_i(n)\|\right)_{C_i} Q_{ji}(n) \tag{4.40}$$

$$Q_{ji}(n) = [x_j - t_i(n)]\ [x\_j - t\_i\ (n)]^T \tag{4.41}$$

$$\Sigma_i^{-1}(n+1) = \Sigma_i^{-1}(n) - \eta_3 \frac{\partial \xi(n)}{\partial \Sigma_i^{-1}(n)} \tag{4.42}$$

where the term $e_j(n)$ is the error signal of output unit j at time n. The term G'(.) is the first derivative of the Green's function G(.) with respect to its argument. The update equation for w_i, t_i and $\sum_t^{-1}()$ are assigned different learning-rate parameters η_1, η_2, and η_3, respectively. The covariance matrix determines the receptive field of the Gaussian radial-basis function $G(\|x-t_i\|_C)$ given in the equation

$$G(\|x - t_i\|_C = exp\left[-\frac{1}{2}(x - t_i)^T \Sigma^{-1} (x - t_i)\right] \tag{4.43}$$

Here the required training input/output pairs of the training set, that is $\{R, w_{RAB}\}$. In the application, desired sources are located at elevation angels θ ranging from −90° to +90° to span the field of view of the antenna. Once the RBFNN is trained with a representative set of training input/output pairs, it is ready to function in the performance phase. In the performance phase, the RBFNN produces estimation of the weight vector w_{RAB}.

4.2.1.3 Performance Phase of the RBFNN

After the training phase is complete, the RBFNN has established an approximation of the desired input-output mapping. In the performance phase, the neural network is expected to generalize, that is, respond to inputs that has never seen before, but drawn from the same distribution as the inputs used in the training set. In the performance

phase, the RBFNN produces outputs to previously unseen inputs by interpolating between the inputs used in the training phase.

1. Generate the rearranged covariance matrix;

2. Present the array output vector at the input layer of the trained RBFNN. The output layer of the trained RBFNN will produce the estimation of the weight vector for the array output.

Unlike the SMI, the least mean-square, or recursive least squares algorithms, where the optimization is carried out whenever the directions of the desired or interfering signals change, in our algorithm, the weight vector of the trained network can be used to produce the optimum weight vector needed to steer the narrow beams of the adaptive array to the directions of the desired signal in real time.

4.2.2 Simulation and Results

We present here some simulations to justify the performance of the SMI, LSMI and robust adaptive beamforming.

4.2.2.1 Array Factor Plots with variation of number of array elements with different element spacing:

We determined that the element spacing must be $d \leq \lambda / 2$ to prevent spatial aliasing. Here we relax this restriction and look at various element spacing with different element linear array and resulting array characteristics, namely, their beam-pattern. Here we show the beam-pattern plots for different algorithm when the angle of arrival of desired user is at 30° and interferer at -60° for different element spacing $\lambda/2$, $\lambda/4$ and $\lambda/8$. We note that from simulation the algorithm places adaptively the maxima in the direction of desired user and nulls at the AoA of the interferer for various values of N.

SMI algorithm

a) The array factor plots of SMI algorithm for different element spacing as $\lambda/2$, $\lambda/4$ and $\lambda/8$ with N = 5, 8, 10 are as follows:

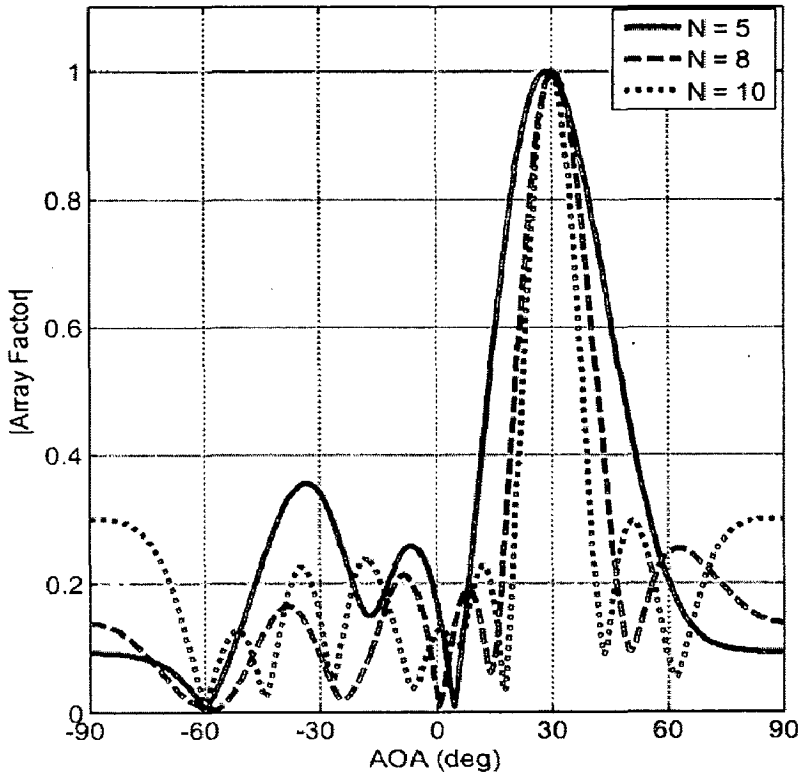


Figure 4.2: Array Factor plots for SMI algorithm (for d = 0.5λ)

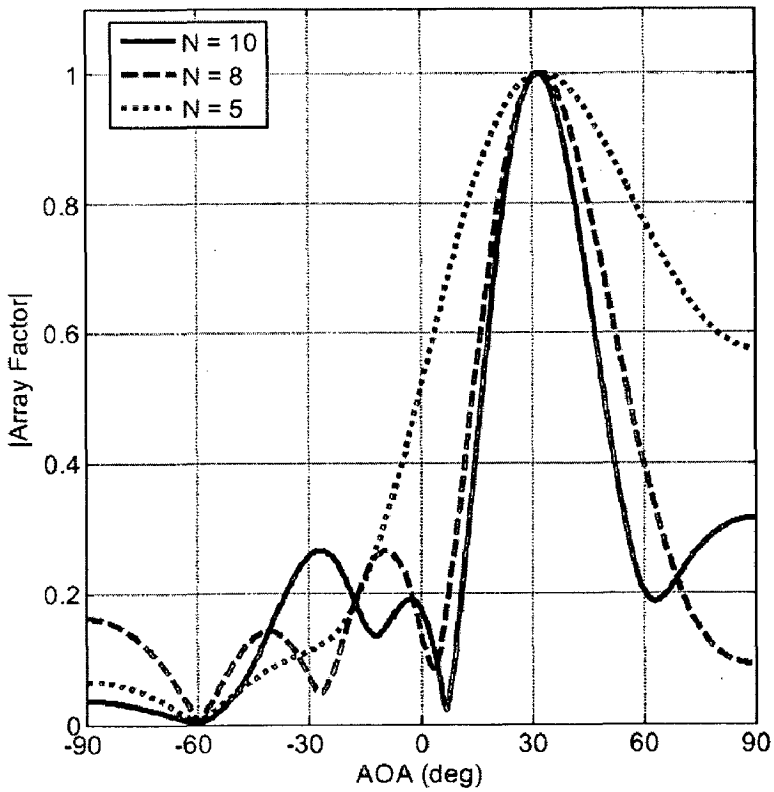


Figure 4.3: Array Factor plots SMI algorithm (for d = 0.25λ)

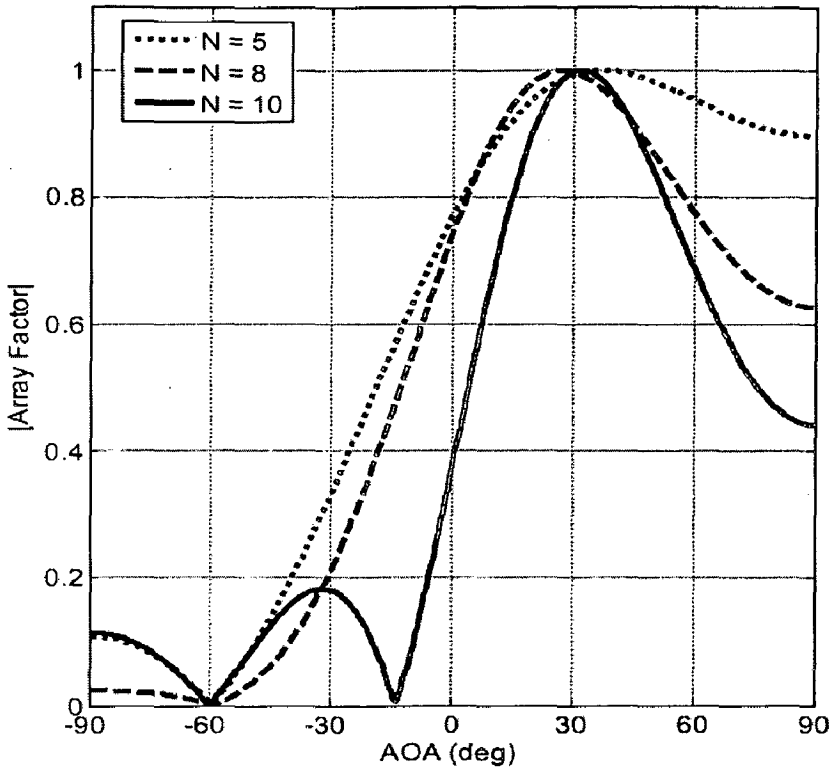


Figure 4.4: Array Factor plots for SMI algorithm (for d=0.125λ)

b) The array factor plots of LSMI algorithm for different element spacing as λ/2, λ/4 and λ/8 with N = 5, 8, 10 are as follows:

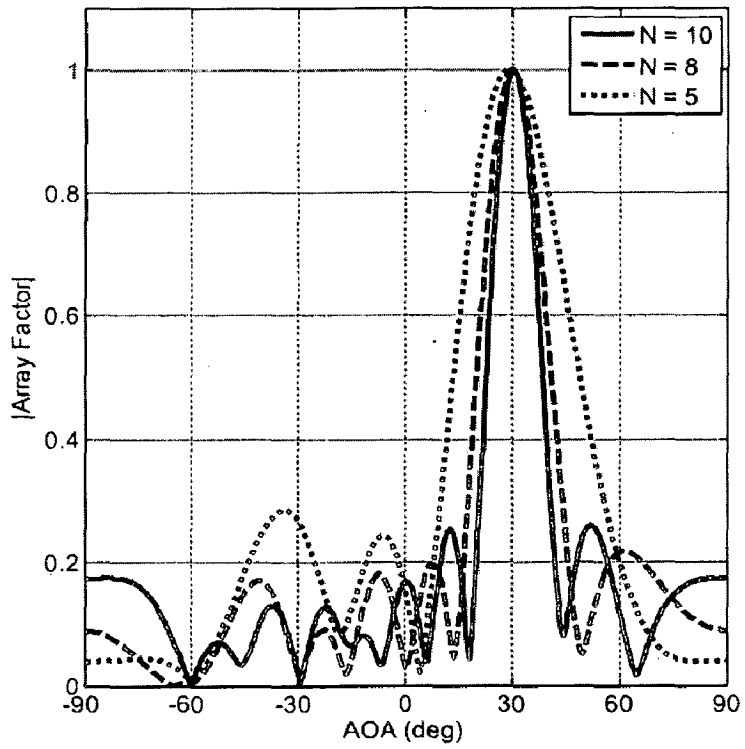


Figure 4.5: Array Factor plots for LSMI algorithm (d = 0.5λ)

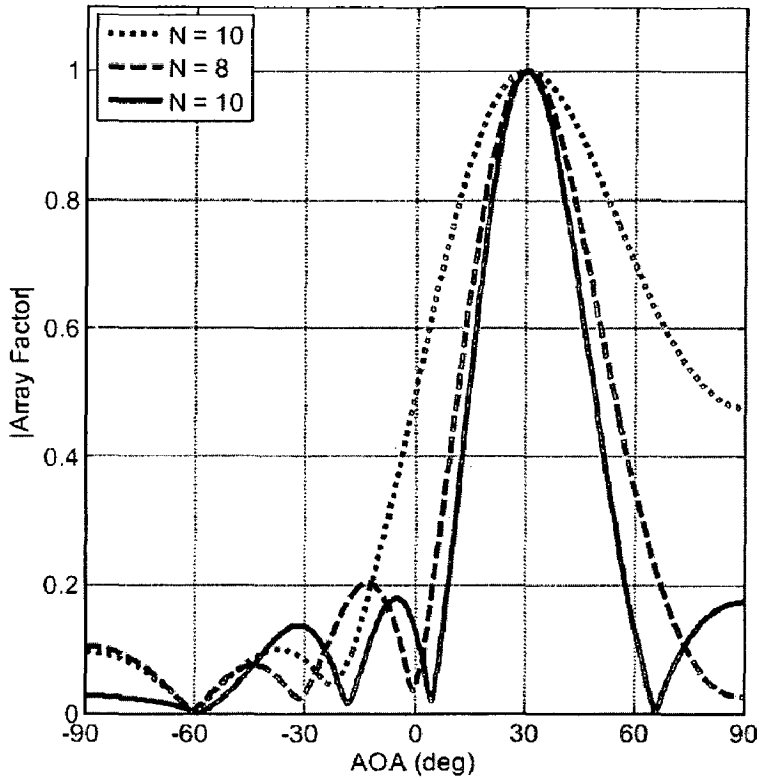


Figure 4.6: Array Factor plots for LSMI algorithm (d = 0.25λ)

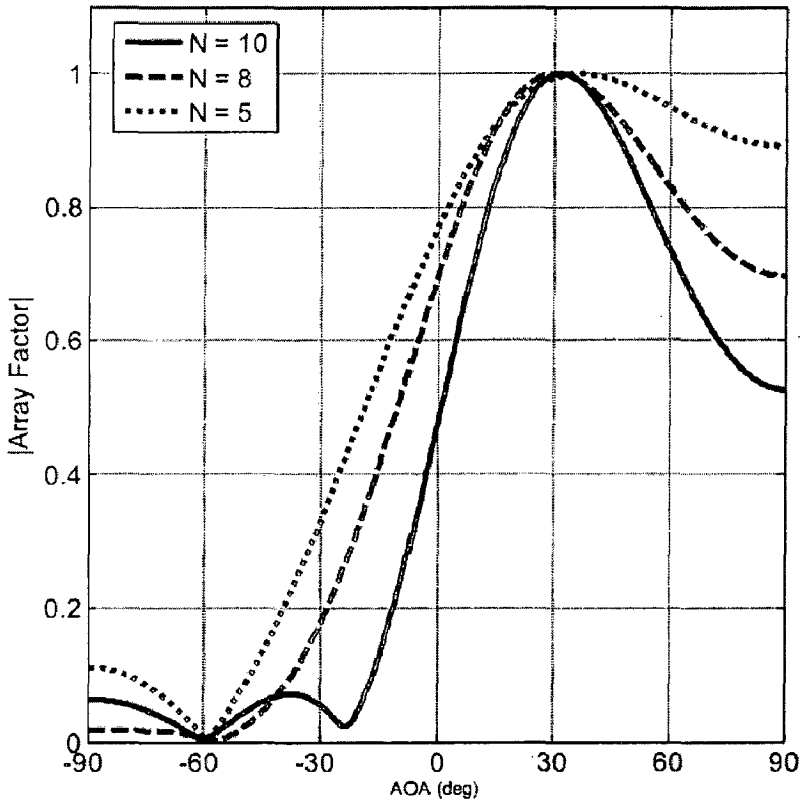


Figure 4.7: Array Factor plots for LSMI algorithm (d = 0.125λ)

c) The array factor plots of Robust Adaptive Beamforming algorithm for different element spacing as $\lambda/2$, $\lambda/4$ and $\lambda/8$ with N = 5, 8, 10 are as follows:

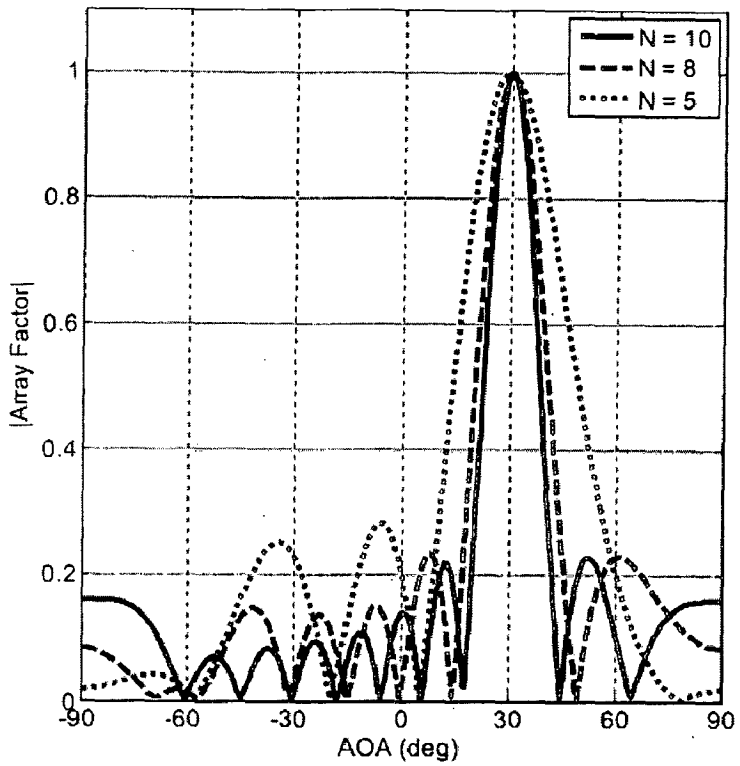


Figure 4.8: Array Factor plots for RAB algorithm (d = 0.5λ)

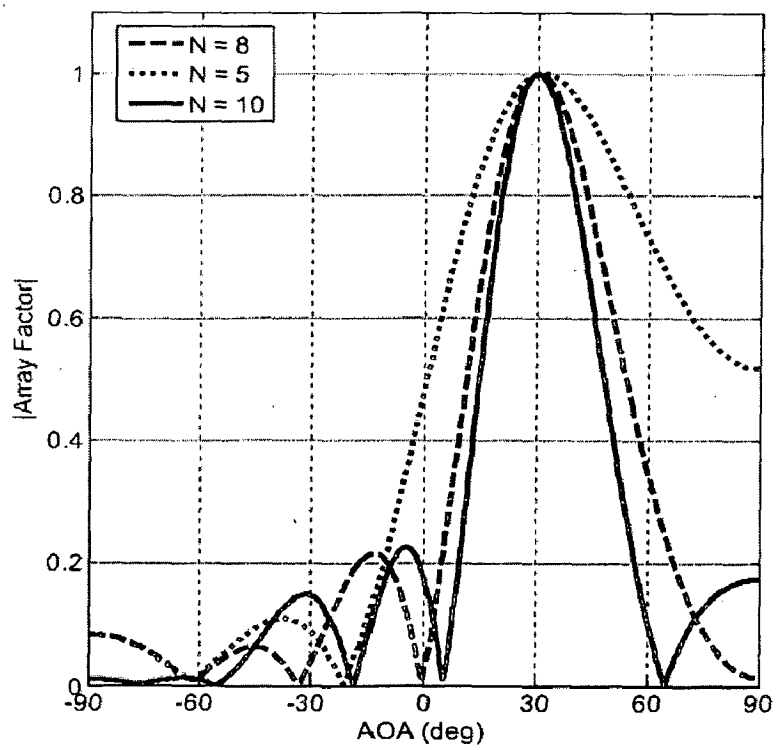


Figure 4.9: Array Factor plots for RAB algorithm (d = 0.25λ)

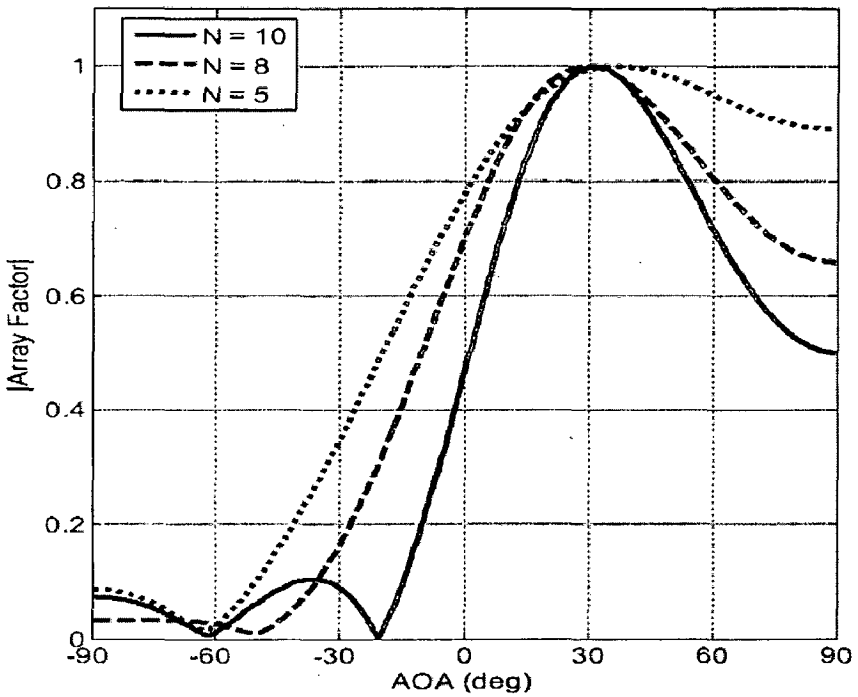


Figure 4.10: Array Factor plots for RAB algorithm (d = 0.125λ)

From figures with different element spacing it is evident that the optimum spacing between elements is half wavelength and as number of element spacing increases width of main lobe decreases, this is crucial for the application of smart antennas when single narrower beam is required to track the mobile, and number of side lobes increases these represents power radiated or received in potentially unwanted directions. So in a wireless communication system side lobes will contribute to the level of interferences spreads in the cell or sector by a transmitter as well as level of interference seen by a receiver when antenna arrays are used. It is evident that more elements an array has or alternatively the larger the array gets, the better the characteristics of radiation pattern as for as its shape and degree of freedom.

From these figures we get that array factor with different element spacing λ/2, λ/4 and λ/8 for Robust Adaptive beamforming algorithm is better than the SMI and LSMI algorithms.

4.2.2.2 Comparison of Array Beampatterns of Algorithms

We assume a uniform linear array with M = 10 omnidirectional sensors spaced half a wavelength apart. For each scenario, 100 simulation runs are used to obtain each

simulated point. In the training phase, desired sources are located at elevation angles θ ranging from -90° to $+90^{\circ}$. In all examples, two interfering sources are assumed to impinge on the array from the directions of arrival (DoAs) 30° and 50°, respectively. The diagonal loading factor $\xi = 10\ \sigma_n^2$ is taken in the LSMI algorithm, where σ_n^2 the noise power.

We assume that both the presumed and actual signal spatial signatures are plane waves impinging from the DoAs 0° and 2°, respectively. Figure 4.11 displays the beampatterns of the methods tested for the fixed SNR =10dB for the no-mismatch case.

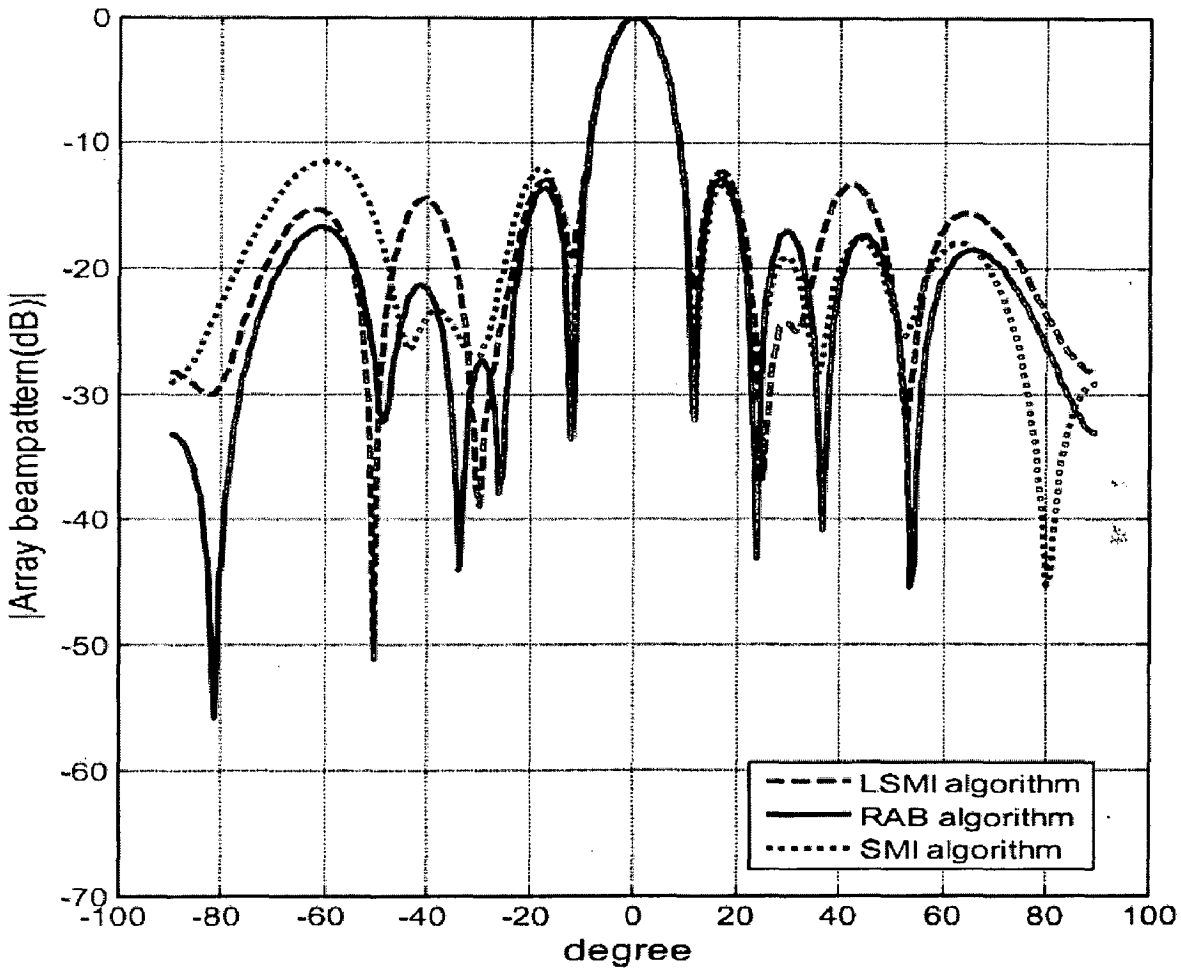


Figure 4.11: Comparison of beampatterns (for no mismatch)

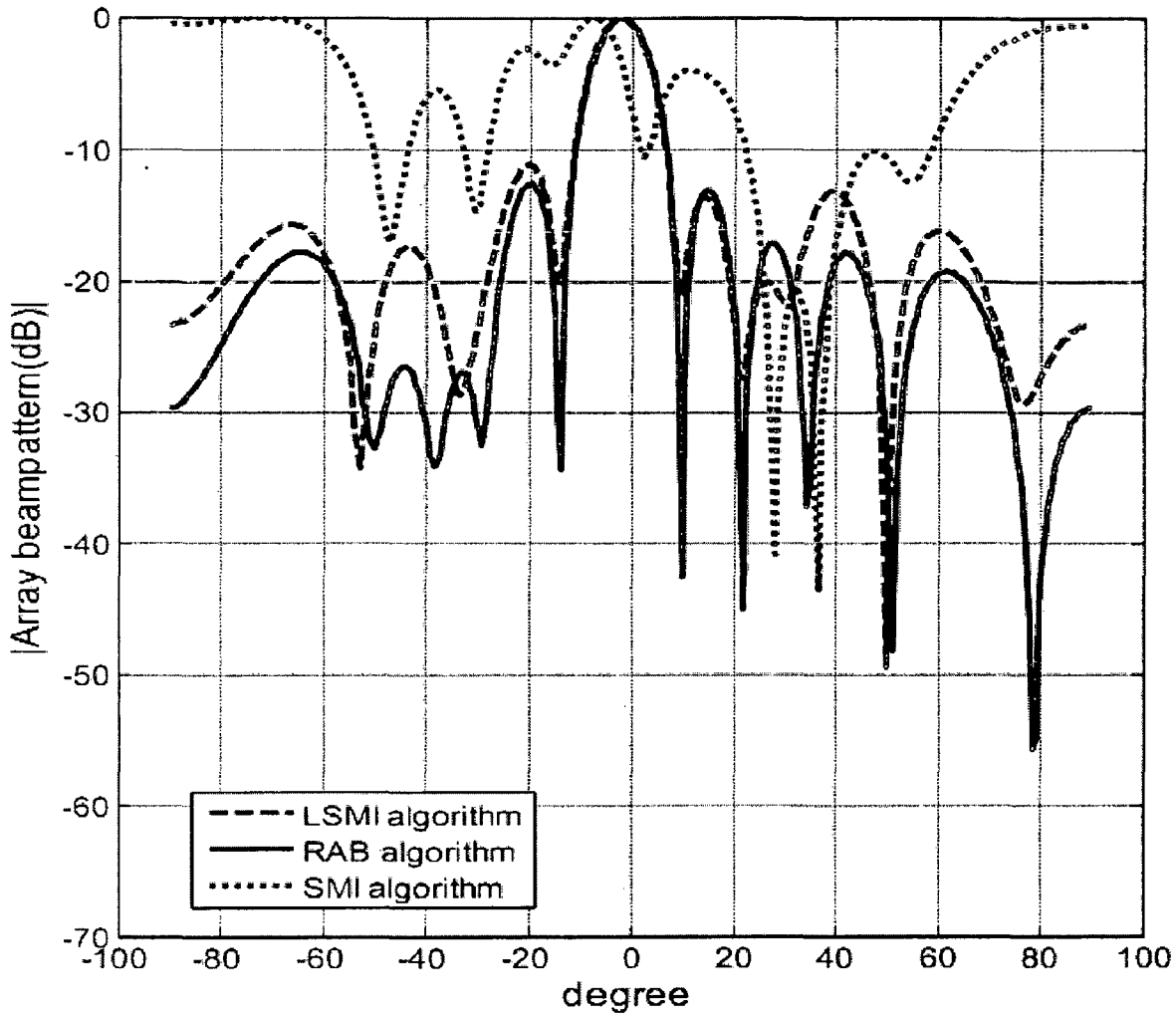


Figure 4.12: Comparison of beampatterns (for 2° mismatch)

From figure 4.11, we note that the robust adaptive beamforming algorithm based on RBFNN can adapt the radiation pattern of the antenna to direct narrow beam to the desired signal and nulls interfering sources. Figure 4.12 displays the beampatterns of the methods tested for the fixed SNR =10dB for a 2° mismatch. From figure 4.12, we note that although the beampatterns of the robust adaptive beamforming algorithm based on RBFNN do not have nulls at the DoAs of the interferences as deep as those of the SMI algorithm, the interferences are sufficiently suppressed by our algorithm.

Chapter 5

Conclusions and Future Scope

5.1 Conclusions

In the first part of the report, we studied the performance of the conventional algorithms, LMS and CMA and found out some key observations regarding their performance.

The inter-element spacing between the antenna elements is an important factor in the design of an antenna array: a distance of more than $\lambda/2$ between the elements (where is the wave length of the incoming or outgoing signals) produces grating lobes. If the inter element spacing between the antenna elements is less than $\lambda/2$ the mutual coupling between the elements cannot be neglected any more. That's why the optimum element spacing for beamforming application distance, d = $\lambda/2$. In the LMS, we need a reference signal $d(k)$ to which we want to have the output converge. In CMA, however, the reference signal is not necessary, we use the a priori information that |y| = 1 in the absence of interfering signals.

In the second part of the report, the NN based DBF is presented with a detailed mathematical analysis and then afterwards its simulation is performed using MATLAB. In order to highlight the robustness of the technique, its results are compared with the results of conventional beamforming methods, SMI and LSMI. The robust adaptive beamforming algorithm is based on explicit modeling of uncertainty in the desired signal array response and three layer radial basis function neural network which treats

the problem of computing weights of an adaptive array antenna as a mapping problem. We have seen that SMI, LSMI and neural network based robust adaptive beamforming algorithm to track the desired signal while simultaneously nulling the interference sources.

- These algorithms have optimum spacing between array elements is $d = 0.5\lambda$ and it is found that more elements an array has or alternatively the larger the array gets, the better the characteristics of radiation pattern as for as its shape and degree of freedom.

- Robust adaptive beamforming algorithm based on RBFNN is much less sensitive to signal steering vector mismatch but the SMI algorithm is very sensitive even to slight mismatches. The robust adaptive beamforming algorithm based on RBFNN adapted the radiation pattern of antenna to direct narrow beam to desired signals and nulls the interference sources.

- The robust adaptive beamforming algorithm based on RBFNN consistently enjoys excellent performance because it achieves the values of SINR that are close to the optimal one in a wide range of the SNR and N but values of SMI and LSMI algorithm did not achieve to the optimal one.

So, it is concluded that the robust adaptive beamforming algorithm based on neural network consistently enjoys a significantly improved performance as compared with other existing algorithms.

5.2 Scope of future work

- Neural network like Recurrent Neural Network (RNN) with reduced structural complexity can be incorporated for adaptive beamforming.

- Adaptive Neuro-Fuzzy Inference System (ANFIS) may be considered better robustness to the beamforming algorithms.

References

[1] Carl B. Dietrich, Jr., Warren L. Stutzman, Byung-Ki Kim, and Kai Dietze, "Smart Antennas in Wireless Communications: Base-Station Diversity and Handset Beam forming", *IEEE Antennas and Propagation Magazine*, vol. 42, no. 5, October 2000.

[2] Brennan L. E., Mallet J. D. and Reed I. S., "Adaptive Arrays in Airborne MTI Radar," *IEEE Trans. Antennas Propagation*, vol. 24, pp. 607-615, 1976.

[3] Syed Shah Irfan Hussain, Syed Amjad Hussain Shah and Mohammad Imran Sheikh, "A Mobile Tracking Algorithm for Adaptive Array Smart Antennas by Adapting the Weights of Transmit Antenna," *IEEE transaction on Smart Antenna*, pp. 58-63, July 2004.

[4] Mohammad Tariqul Islam, Zainol Abidin Abdul Rashid, "MI-NLMS adaptive beamforming algorithm for smart antenna system applications," *Journal of Zhejiang University SCIENCE A*, vol. 10, pp. 1709-1716, July 2006.

[5] A. H. El Zooghby, C. G. Christodoulou, and M. Georgiopoulos, "Performance of radial basis function networks for direction of arrival estimation with antenna arrays," *IEEE Trans. Antennas Propagation*, vol. 45, pp. 1611-1617, Nov. 1997

[6] Xin Song, Jinkuan Wang, and Yinghua Han, "Robust Capon Beamforming in the Presence of Mismatches," *Proceedings of ISCIT 2005*, pp. 135-138, July 2005.

[7] Michael Chryssomallis, "Smart Antennas", *IEEE Antennas and Propagation Magazine*, vol. 42, no. 3, June 2000.

[8] Jack H. Winters, "Smart Antennas for Wireless Systems," IE*EE Personal Communications*, vol. 1, pp. 23-27, February 1998.

[9] J. C. Liberti and T. S. Rappaport, "Smart Antennas for Wireless Communications: S-95 and Third Generation CDMA Applications," NJ, Prentice Hall, 1999.

[10] J. Litva, "Digital Beamforming in wireless communications," Artech House Publishers, 1996.

[11] R. C. Hansen, "Phased Array Antennas," *John Wiley and Sons*, New York, 1997.

[12] A. F. Nagub, A. Paulraj, and T. Kailath, "Capacity improvement with base station antenna arrays in cellular CDMA," *IEEE Transaction on Vehicular Technology*, vol. 43, no. 3, pp. 691-698, August 1994.

[13] A. E. Zooghby. "Smart Antenna Engineering", *Artech House*, Boston, 2005

[14] M. D. Zoltowski, G. M. Kautz, and S. D. Silverstein, "Beamspace root-MUSIC," *IEEE Trans. Signal Processing*, vol. 41, pp. 344–364, January 1993.

[15] H. Liu and Q. A. Zeng, "Modeling and performance analysis of mobile communication systems using adaptive beamforming technique," *Proceeding of the 3rd Wireless Telecommunications Symposium*, Pomona, CA, pp. 33–38, (2004).

[16] Y. Wang and J. R. Cruz, "Adaptive antenna arrays for cellular CDMA Communication systems," *Proc. IEEE Int'l. Conf. Acoustics, Speech and Signal Processing*, Detroit, pp. 1725-1728, 1995.

[17] D. H. Johnson and Dan E. Dudgeon, "Array signal processing: concepts and techniques," *P T R Prentice-Hall*, 1993.

[18] J. H. Winters, "Optimum combating in digital mobile radio with co-channel interference," *IEEE Trans. Vehicular Technology*, vol. 33, pp. 144-155, August 1984.

[19] J. fernadez, I. R. Corden, and M. Barrett, "Adaptive array algorithms for optimal combining in digital mobile communications systems," *IEEE Eighth Int. Conf. Antenna and propagation*, Heriot-Watt University, U.K., 1993.

[20] J. Li and P. Stoica. "Robust Adaptive Beamforming," *John Wiley & Sons, Inc.*, Hoboken, New Jersey, 2006.

[21] D. N. Godard, "Self-recovering equalization and carrier tracking in a two-dimensional data communication system," *IEEE Trans. Comm.*, vol. 28, pp. 1867-1875, 1980.

[22] A. M. Viterbi and A. J. Viterbi, "Erlang Capacity of a Power Controlled CDMA System," *IEEE Journal on Selected Areas in Communication*, vol. 11, pp. 892–900, August 1993.

[23] P. R. Chang, W. H. Yang and K. K. Chan, "A neural network approach to MVDR beamforming problem," *IEEE Trans. Antennas Propagation*, vol. 40, pp. 313-322, 1992.

[24] R. T. Compton, "Adaptive Antennas: Concepts and Applications," *Prentice Hall*, Englewood Cliffs, NJ, 1988.

[25] Frank Gross, "Smart Antenna for Wireless Communication," *Mcgraw-hill*, September 14, 2005.

[26] Lal C. Godara, "Application of antenna arrays to mobile communications, part II: beam-forming and direction-of-arrival considerations," *Proc. IEEE*, vol. 85, no. 8, pp. 1195-1234, August 1997.

[27] Simon Haykin, "Adaptive filter theory," Fourth edition, *Pearson education Asia*, Second Indian reprint, 2002.

[28] Xin Song, Jinkuan Wang, Xuefen Niu, "Robust Adaptive Beamforming Algorithm Based on Neural Network," *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 1844-1849, Sep. 2008.

Appendix

[1] MATLAB code for the adaptive array demonstrator for DBF using LMS algorithm

```
close all;
clear all;
clc;
%adaptive array demonstrator

sigSourceTheta = 25;     %degrees, direction of signal x
sigSourceTheta = (2*pi/360)*sigSourceTheta;
nSource1Theta = 0;       %degrees, direction of noise source 1
nSource1Theta = (2*pi/360)*nSource1Theta;
nSource2Theta = -40;     %degrees, direction of noise source 2
nSource2Theta = (2*pi/360)*nSource2Theta;

theta = pi*(-1:0.005:1);
arrLen = 4;              %nr of antennas
bitRate = 100;
simFreq = 4*bitRate;     %simulation frequency
TSim = 1/simFreq;        %simulation sample period

messageBits = [1 -1 1 1 1 -1 1 -1 -1 -1 1 1 -1 1 -1 1 1 1 -1 -1 -1 1 1
1 1 1 -1 1 ...
               -1 1 1 1 -1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 1 -1 1 -1 1
1 1];
messageBits = upsample(messageBits, simFreq/bitRate);%upsample message
t = TSim:TSim:(length(messageBits)/simFreq);             %timeline

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%generate a complex-MSK signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
messageInt=(cumsum(messageBits))/8;     %1xlength(t) vector
Q = cos(pi*messageInt);
I = sin(pi*messageInt);
sourceSig = I+j*Q;       %the signal to be received, 1xlength(t) vector

% the undesired complex noise singals-> uniform phase, gaussian
amplitude distribution
noiseSig1 = normrnd(0,1,1,length(t)).*exp (j*(unifrnd(-
pi,pi,1,length(t))));    %1xlength(t) vector
noiseSig2 = normrnd(0,1,1,length(t)).*exp (j*(unifrnd(-
pi,pi,1,length(t))));    %1xlength(t) vector

systemNoise =zeros(arrLen, length(t));% system noise for every antenna
for i = 1:arrLen
    systemNoise(i,:) = normrnd(0,0.1,1,length(t)).*exp (j*(unifrnd(-
pi,pi,1,length(t))));
```

```
46  end;
47
48  %%
49  Kd = pi;          %distance between antenna elements = lambda / 2
50
51  %%
52  % array responses for the desired signal x and noise (unwanted
53  signals) n1 and n2
54  steerVecSource = zeros(1,arrLen);
55  steerVecN1 = zeros(1,arrLen);
56  steerVecN2 = zeros(1,arrLen);
57
58  for k = 1:arrLen
59      steerVecSource(k) = exp(j*(k-1)*Kd*sin(sigSourceTheta));
60      steerVecN1(k) = exp(j*(k-1)*Kd*sin(nSource1Theta));
61      steerVecN2(k) = exp(j*(k-1)*Kd*sin(nSource2Theta));
62  end;
63
64  recdSource = zeros(arrLen, length(t));
65  recdN1 = zeros(arrLen, length(t));
66  recdN2 = zeros(arrLen, length(t));
67
68  for i = 1:arrLen
69      recdSource(i,:) = sourceSig .* steerVecSource(i);
70      % received signal from signal source x
71      recdN1(i,:) = noiseSig1 .* steerVecN1(i);
72      % received signal from noise source n1
73      recdN2(i,:) = noiseSig2 .* steerVecN2(i);
74      % received signal from noise source n2
75  end;
76
77  %total received signal is the
78  %+-                                -+
79  %| signal received at antenna 1    |
80  %| signal received at antenna 2    |
81  %|                  |              |
82  %| signal received at antenna n    |
83  %+-                                -+
84  totalRecdSig = (systemNoise + recdN1 + recdN2 + recdSource);
85
86  %%
87  % define weight vector
88  w = zeros(1,arrLen);
89  mu = 0.05;
90  y = zeros(1,length(t));      %output
91  e = zeros(1,length(t));      %error
92
93  %%
94  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
95  %LMS Algorithm
96  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
97  for repeat=1:1       % use repeat for more convergence on the same
98  message
99      for i=1:length(t)
100         y(i) = w * totalRecdSig(:,i);
101         e(i) = sourceSig(i)-y(i);
102         w = w + mu *e(i)*(totalRecdSig(:,i))';
```

```
103         end;
104     end;
105
106     %%
107     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
108     %Plot all figures
109     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110     figure;
111     subplot 211;
112     plot(abs(sourceSig),':','LineWidth',2);
113     hold on;
114     plot(abs(y),'r');
115     legend('Source signal amplitude, |d|', '|y|.');
116     ylabel('amplitude');
117     %xlabel('sample(index)');
118     hold off;
119
120     subplot 212, plot(abs(e));
121     legend('|error|');
122     ylabel('amplitude');
123     xlabel('sample(index)');
124     axis([0 210 0 1]);
125     arrayvec=zeros(arrLen,length(theta));
126     for k = 0:arrLen-1
127         arrayvec(k+1,:) = exp(j*k*Kd*sin(theta));
128     end;
129
130     %calculate response of array
131     F = w*arrayvec;
132     figure;
133     plot(((theta/(2*pi))*360), 20*log10(abs(F)));
134     title('amplitude response antenne pattern');
135     ylabel('(dB)');
136     xlabel('angle(degrees)');
```

[2] MATLAB code for adaptive array demonstrator for Digital Beamforming using CM algorithm

```
1   close all;
2   clear all;
3   clc;
4   typeofinterferer1='noise';
5   %set to 'noise' for guassian type of interferer
6   %set to 'sign1' for MSK type of interferer
7
8   sigSourceTheta = 10;     %degrees
9   sigSourceTheta = (2*pi/360)*sigSourceTheta;
10  nSource1Theta = -10;     %degrees
11  nSource1Theta = (2*pi/360)*nSource1Theta;
12  nSource2Theta = -40;     %degrees
13  nSource2Theta = (2*pi/360)*nSource2Theta;
14
```

```
theta = pi*(-1:0.005:1);
arrLen = 4;              %nr of antennas
bitrate = 200;
simFreq = 4*bitrate;     %simulation frequency
TSim = 1/simFreq;        %simulation sample period
messageBits = [1 -1 1 1 1 -1 1 -1 -1 -1 1 1 -1 1 -1 1 1 1 -1 -1 -1 1 1
1 1 1 -1 1 ...
               -1 1 1 1 -1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 1 -1 1 -
1 1 1 1];
messageBits = upsample(messageBits, simFreq/bitrate);%upsample message
t = TSim:TSim:(length(messageBits)/simFreq);             %timeline

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%generate a complex MSK signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
messageInt=(cumsum(messageBits))/8;     %1xlength(t) vector
Q = cos(pi*messageInt);
I = sin(pi*messageInt);
sourceSig = I+j*Q;       %the signal to be received
referenceSig = sourceSig(1:end);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%generate 2 interferers n1 and n2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if the interfere n1 needs to be gaussian noise -> uniform phase,
normal amplitude distribution
if (strcmpi(typeofinterferer1, 'noise'))
    noiseSig1 = normrnd(0,1,1,length(t)).*exp(j*(unifrnd(-
pi,pi,1,length(t))));
end;

% if testing with an MSK signal as interferer is required: noise
source n1 will be MSK signal
if (strcmpi(typeofinterferer1, 'sign1'))
    messageBitsN1 = [-1 -1 1 -1 1 -1 -1 1 -1 -1 -1 1 1 1 -1 -1 1 1 1 1
-1 1 -1 1 -1 1 -1 -1 ...
                     -1 1 1 -1 1 -1 -1 -1 1 -1 1 -1 -1 -1 -1 -1 1 1 1
-1 1 -1 1 -1 -1];
    messageBitsN1 = upsample(messageBitsN1, simFreq/bitrate);
%upsample message
    t = TSim:TSim:(length(messageBits)/simFreq); %timeline
    messageIntN1=(cumsum(messageBitsN1))/8;
    Qn1 = cos(pi*messageIntN1);
    In1 = sin(pi*messageIntN1);
    noiseSig1 = In1+j*Qn1; %the signal to be received
end;

%noise source 2 is gaussian
noiseSig2 = normrnd(0,1,1,length(t)).*exp(j*(unifrnd(-
pi,pi,1,length(t))));

systemNoise = zeros(arrLen,length(t));% system noise for every antenna
for i = 1:arrLen
    systemNoise(i,:) = normrnd(0,0.1,1,length(t)).*exp(j*(unifrnd(-
pi,pi,1,length(t))));
end;

```

```
%%
%lambda = sym('lambda');
%K = 2*pi/ lambda
%d = 0.5*lambda
Kd = pi;         %distance between antenna elements = lambda / 2
%alpha = -Kd*sin(theta_nul);

%%
%array responses for the desired signal x and noise n1 and n2
steerVecSource = zeros(1,arrLen);
steerVecN1 = zeros(1,arrLen);
steerVecN2 = zeros(1,arrLen);

for k = 1:arrLen
    steerVecSource(k) = exp(j*(k-1)*Kd*sin(sigSourceTheta));
    steerVecN1(k) = exp(j*(k-1)*Kd*sin(nSource1Theta));
    steerVecN2(k) = exp(j*(k-1)*Kd*sin(nSource2Theta));
end;

recdSource = zeros(arrLen, length(t));
recdN1 = zeros(arrLen, length(t));
recdN2 = zeros(arrLen, length(t));

for i = 1:arrLen
    recdSource(i,:) = sourceSig .* steerVecSource(i);
    % received signal from signal source x
    recdN1(i,:) = noiseSig1 .* steerVecN1(i);
    % received signal from noise source n1
    recdN2(i,:) = noiseSig2 .* steerVecN2(i);
    % received signal from noise source n2
end;

%total received signal is the
%+-                            -+
%| signal received at antenna 1 |
%| signal received at antenna 2 |
%|              |               |
%| signal received at antenna n |
%+-                            -+
totalRecdSig = (systemNoise + recdN1 + recdN2 + recdSource);

%%
w = zeros(1,arrLen);
%start weight vector, may be in the direction of the desired signal:
for i=1:arrLen
    w(i) = (1/4)*exp(-j*pi*(i-1)*sin(sigSourceTheta));
end;
%start vector may be any type:
%w = [0.1 0.1 0.1 0.1]
wstart = w;
%mu = 0.007;
mu = 0.001;
y = zeros(1,length(t));
e = zeros(1,length(t));

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Constant Modulus Algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for repeat=1:1
    for i=1:length(referenceSig)
        y(i) = w * totalRecdSig(:,i);
        %e(i) = 1 - y(i)
        e(i) = y(i)/((abs(y(i))))^2-y(i);
        %SATO's principle for updating the error
        w = w + mu*e(i)*(totalRecdSig(:,i))';
    end;
end;

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot all figures
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot 211, plot(abs(sourceSig),':');
hold on;
grid on;
axis([0 210 0 2]);
plot(abs(y),'r');
legend('Source signal, |d|', 'Estimate of the source signal,
|y|','Location', 'Best');
ylabel('amplitude');
hold off;

subplot 212, plot(abs(e));
grid on;
legend('|error|', 'Location', 'Best');
ylabel('amplitude');
xlabel('sample(index)');
axis([0 210 0 1]);
arrayvec=zeros(arrLen,length(theta));
for k = 1:arrLen
    arrayvec(k,:) = exp(j*(k-1)*Kd*sin(theta));
end;

%calculate response of array
F = w*arrayvec;
F2 = wstart*arrayvec;
figure;
plot(((theta/(2*pi))*360), 20*log10(abs(F)));
hold on;
plot(((theta/(2*pi))*360), 20*log10(abs(F2)),'r');
title('amplitude response antenne pattern, desired signal: 10 degrees,
interferers: -10 and -40 degrees');
ylabel('(dB)');
xlabel('angle(degrees)');
grid on;
```