2-14-2014

Randomly Spaced Smart Antennas

Maialen Ciaurriz Velasco

Maialen Ciaurriz Velasco

Candidate

Electrical and Computer Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Christos Christodoulou , Chairperson

Manel Martinez-Ramón

Zhen Peng

Randomly Spaced Smart Antennas

by

Maialen Ciaurriz Velasco

Telecommunication Engineering, "Universidad Pública de Navarra"
University, 2010

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2013

Dedication

To my Basque and New Mexico Family. Concretely to my amatxo, aitatxo and Garbanzo Beam.

Acknowledgments

I would like to thank my family, who pushed my to this adventure, and of course my Apodaca-Joffe family, who treat my as heir Spanish daughter and sister.

I would like to thank my advisor Dr. Christodoulou, who gave me this incredible opportunity of being member of his group.

In addition, I would like to thank Dr. Martínez-Ramón for all his work during the last month.

Also my thanks to all the lab members, specially to Elizabeth, who was there listening my doubts and issues with the thesis.

Finally, I would like to thank the University of New Mexico Electrical and Computer Engineering Department, concretely Elmyra Grelle for all her hard work.

Eskerrik asko.

Randomly Spaced Smart Antennas

by

Maialen Ciaurriz Velasco

Telecommunication Engineering, "Universidad Pública de Navarra" University, 2010

M.S., Electrical Engineering, University of New Mexico, 2013

Abstract

The goal of this thesis is to develop "adaptive array" features out of randomly spaced antenna elements. Most optimization techniques that have been presented so far for non-equidistant antenna arrays have been restricted to the analysis of symmetric or linear geometries. Several direction of arrival (DOA) and adaptive beamforming algorithms are implemented and analyzed for both linear and planar randomly spaced antenna configurations; such as Capon and MUSIC for direction of arrival, and LMS, normalized LMS, leaky LMS, generalized normalized LMS, RLS and variable forgetting factor RLS algorithms for the adaptive beamforming.

The advantages and disadvantages of smart antennas based on random array configuration are presented and discussed. Several algorithms have been investigated to study the most suitable ones for optimizing the distribution of the excitation coefficients.

The work discussed herein can be extended to space applications using a cluster of small satellites, UAVs, or any array of sensors that are not aligned together in a standard linear or planar geometrical configuration. Furthermore, the proposed

approach can also be extended to standard, two-dimensional linear arrays when one or more elements fail.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Extensive literature is available on the subject of uniform and nonuniform array processing [2], [3], and [4]. However, the techniques have been restricted to symmetric or linear geometries. Therefore, the main objective of this thesis is to implement and study array processing for randomly spaced antenna array elements. In this thesis, several algorithms for the direction of arrival (DOA) and beamforming are investigated to study the advantages and disadvantages of randomly spaced smart antennas.

This thesis is organized as follows: Chapter 2 first provides an overview of smart antennas in general. The differences between the adaptive array system and the switched-beam system are presented. The adaptive array system was chosen as the main smart system for randomly spaced antenna arrays. Chapter 3 presents an analysis of a theoretical model of array processing. First, it illustrates and explains the signal model for randomly spaced arrays. Afterwards, it continues by analyzing various methods of estimating the direction of arrival. The last section of this chapter reviews the adaptive beamforming techniques used to achieve the desired radiation pattern. Chapter 4 presents the performance of the analyzed algorithm for ran-

domly spaced antenna arrays and illustrates some of the corresponding comparisons. Chapter 5 summarizes the obtained results and provides some suggestions for further research. This thesis concludes with Appendix A, which shows the implemented algorithms for the randomly spaced smart antennas using MATLAB® software.

Chapter 2

Fundamental Concepts of Smart Antennas

Over the few last decades, wireless technology has taken on an important role in the field of communication. The accelerated growth of mobile industries has resulted in an unprecedented demand for better coverage, capacity, and spectrum efficiency [2]. Smart antennas, also known as adaptive array antennas, have been selected as one of the most important technologies to fulfill these requirements, principally due to their ability to control the process of generated beamforming.

Although smart antennas have been used recently for commercial purposes, the Bartlett beamformer was already implemented during the Second World War as an application of the Fourier-based spectral analysis for spatio-temporally sampled data [5].

In addition, to the required demand, the improvements on low-cost digital signal processors, application specific integrated circuits (ASIC), and signal processing techniques have contributed to the commercialization of smart antennas [1].

Despite the fact that these antennas are known as smart antennas, the antennas that are used are not actually smart themselves; the digital signal processing is what makes them intelligent. Using adequate signal processing, adaptive array antennas are able to adapt their radiation pattern in accordance with desired signals and interference signals. Adaptive beamforming algorithms enhance the desired signal while suppressing interference signals or jammers when outputting array antennas.

An antenna array is a combination of antenna elements (identical elements in most cases) whose main objective is to improve the performance of a single element. The total electromagnetic field of the array is obtained by the coherent or incoherent combination of each element [6]. Accordingly, the performance of the array is determined by the distance between each element, the phase excitation, and the amplitude excitation. In the case of adaptive array antennas, digital signal processing calculates the desired amplitude excitation, also known as the weight vector, in order to obtain optimum beamforming.

Generally, smart antennas are categorized according to two major configurations: switched-beam and adaptive array systems.

A switched-beam system is formed by multiple fixed beams and is considered an extension of the cell sectoring system. The desired signal is detected by one of the predetermined fixed beams, and as the desired signal moves throughout the sectored area, the system moves from one beam to another by changing the phase difference of the signals used to feed the antenna elements (as illustrated in Figure 2.1). The switched-beam system increases the gain compared to the conventional sectored system; however, due to the fact that the beams are fixed, the desired signal may not be located in the middle of the main beam. Therefore, the desired signal or signal of interest (SOI) may receive less intensity than the interference signals or signals not of interest (SNOI).

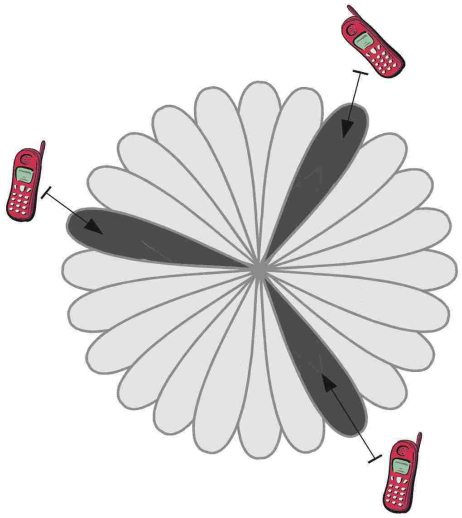
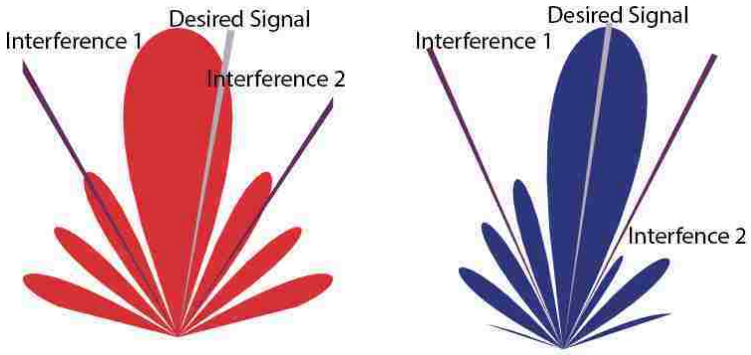


Figure 2.1: Switched-beam system [1]

An adaptive array system adapts the radiation pattern by optimizing the performance of the system in real time. In other words, an adaptive array system has the ability to create a radiation pattern that enhances the desired signals while at the same time suppressing the interference signals [2].

The difference between the radiation pattern of these two configurations is shown in Figure 2.2. Both cases adapt their main beam in the direction of the desired signal; nevertheless, the switched-beam system detects interference signals while the adaptive array system creates nulls at those positions. For this reason, the adaptive array system obtains better performance when a high level of interference is present.

The use of smart antennas has grown exponentially principally due to their capability to reject the interference signals from the desired signal, thus increasing the capacity and the range of these systems. In addition to these advantages, smart antennas have improved security problems because generated beams are more di-



(a) (b)

Figure 2.2: Smart antenna: (a) switched-beam, and (b) adaptive array [1].

rectional; for this reason, intruders have to be located in the same direction as the user. Nevertheless, these systems require more powerful numeric processors and control systems, making them more complicated than the omnidirectional or sectorized systems.

Digital signal processing (DSP) plays an important role in adaptive beam systems. After the array antenna elements receive the incoming signals and the system digitizes them, DSP algorithms are applied to estimate the direction of arrival of the desired and interference signals and to create the optimum beamforming, direction-of-arrival (DOA) and adaptive beamforming algorithm respectively. An adaptive array system block diagram is shown in Figure 2.3.

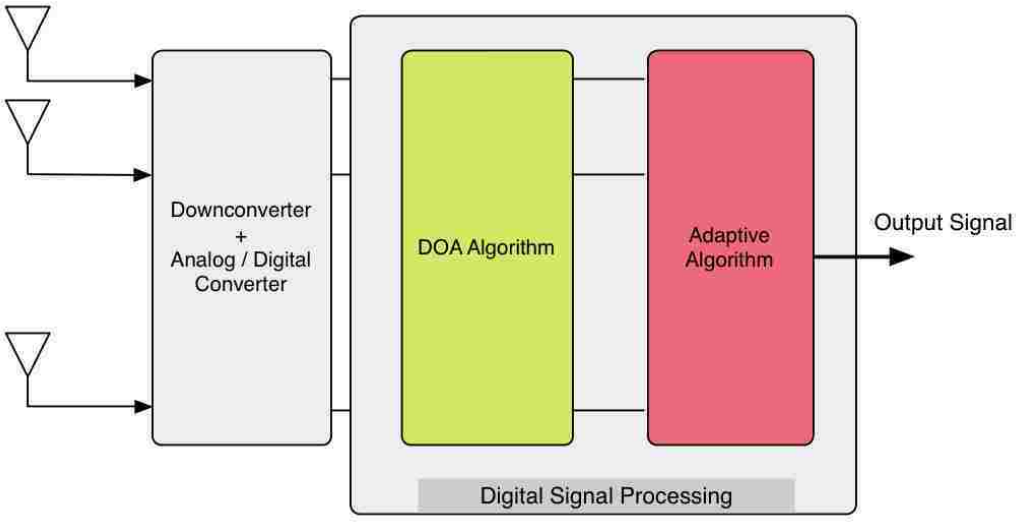


Figure 2.3: Adaptive array system block diagram

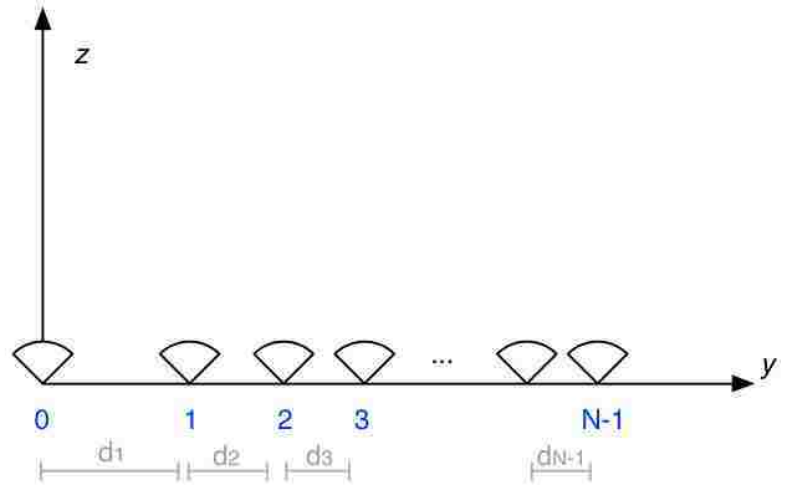
Chapter 3

Array Processing

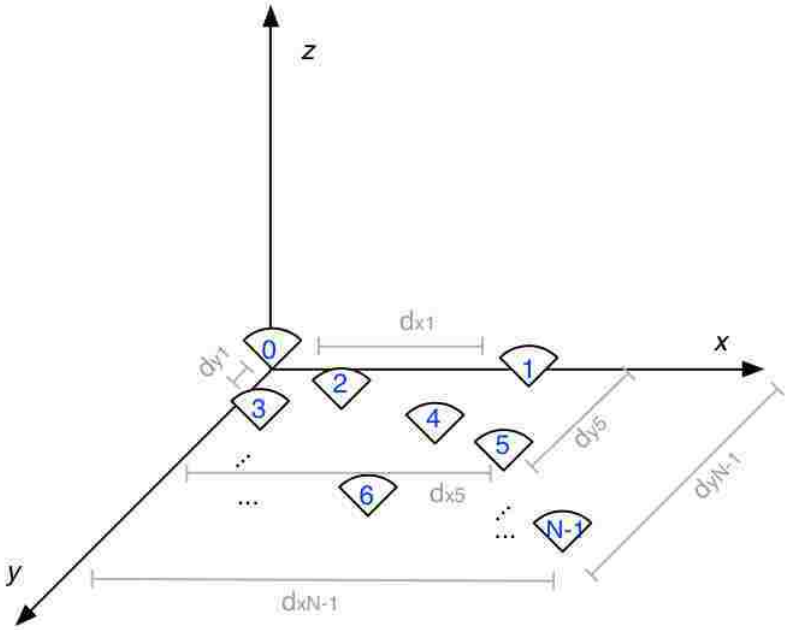
The main objective of digital signal processing is to obtain the most information possible from the signals arriving into the system. In the special case of adaptive array antennas, digital signal processing uses the incoming data to predict the direction of arrival of the signals, and that estimation is used to create the desired beamforming, maximizing the power towards to the users and suppressing any interference.
In this chapter, different DOA and adaptive beamforming algorithms will be explained and analyzed with respect to the randomly spaced antenna array model.

3.1 Signal Model

Within the framework of this thesis, a randomly spaced linear and planar array antenna will be considered for the main beamforming model. Unlike regular smart array antennas in which the elements are spaced apart by a constant uniform value d, random array antennas are composed of randomly distributed elements. A random linear array (RLA) and a random planar array (RPA) are illustrated in Figure 3.1.



(a)



(b)

Figure 3.1: Random smart antennas: (a) random linear array, and (b) random planar array

For both cases, RLA and RPA, the array will be composed of N isotropic antenna elements, and K narrowband signals arriving from different directions $(\{\theta_0, \phi_0\}; \{\theta_1, \phi_1\}; ...; \{\theta_{K-1}, \phi_{K-1}\})$.

Due to the assumption that the incoming signals are located in the far-field and can be considered point sources, the arrival signals can be written as a uniform plane wave:

$$\widetilde{S}_i(t) = S_{iI}(t)cos(\omega_c t) - S_{iQ}(t)sin(\omega_c t) = \mathbb{R}\{S_i(t)e^{j\omega_c t}\} \tag{3.1}$$

where $S_{iI}(t)$ and $S_{iQ}(t)$ are phase and quadrature components respectively for each i-source, $S_i(t) = \sqrt{(S_{iI}(t)^2 + S_{iQ}(t)^2}$ is the envelope of the signal, and ω_c is the angular frequency of the signal.

Having a reference antenna, the signals arriving at the other antennas are related to the reference by a time delay:

$$\tau_{1D} = \frac{d_x sin(\theta_i)}{\nu_0} \quad \text{For } 1-\text{Dimension random linear array and} \tag{3.2a}$$

$$\tau_{2D} = \frac{d_x sin(\theta_i)cos(\phi_i) + d_y sin(\theta_i)sin(\phi_i)}{\nu_0} \tag{3.2b}$$

$$\text{For } 2-\text{Dimension random planar array}$$

where d_x and d_y are the distances between elements. In general, the reference antenna is set at the origin to have zero phase.

As a consequence of this phase difference, the total signal arriving at the nth antenna element in snapshot t is calculated as follows:

$$X_n(t) = \sum_{i=0}^{K-1} S_i(t)e^{-jk_i\tau_n} + n(t) \tag{3.3}$$

where $k_i = \frac{2\pi}{\lambda_i}$ is the wavenumber vector, τ_n is the time delay for the nth element, and $n(t)$ is the noise signal.

The steering vector is defined as follows:

$$\mathbf{a}(\theta_i, \phi_i) = [1, e^{-\frac{2\pi j}{\lambda_i}\tau_1}, ..., e^{-\frac{2\pi j}{\lambda_i}\tau_{N-1}}]^T$$

As the vector that contains the responses of all of the array's elements for a certain source, the total signal can be expressed as a superposition of signals from all the sources by:

$$\mathbf{X}(t) = \mathbf{A}(\boldsymbol{\theta}, \boldsymbol{\phi})\mathbf{S}(t) + \mathbf{n}(t) \tag{3.4}$$

where $\mathbf{X}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), ..., \mathbf{x}_N(t)]^T$ is the total baseband signal matrix, and $\mathbf{A}(\theta, \phi) = [\mathbf{a}(\theta_0, \phi_0), \mathbf{a}(\theta,_1 \phi_1), ..., \mathbf{a}(\theta_{K-1}, \phi_{K-1})]$ is the steering matrix containing all of the incoming sources. Considering all the snapshots, the total signal can be rewritten as follows:

$$\mathbf{X} = \mathbf{A}(\boldsymbol{\theta}, \boldsymbol{\phi})\mathbf{S} + \mathbf{N} \tag{3.5}$$

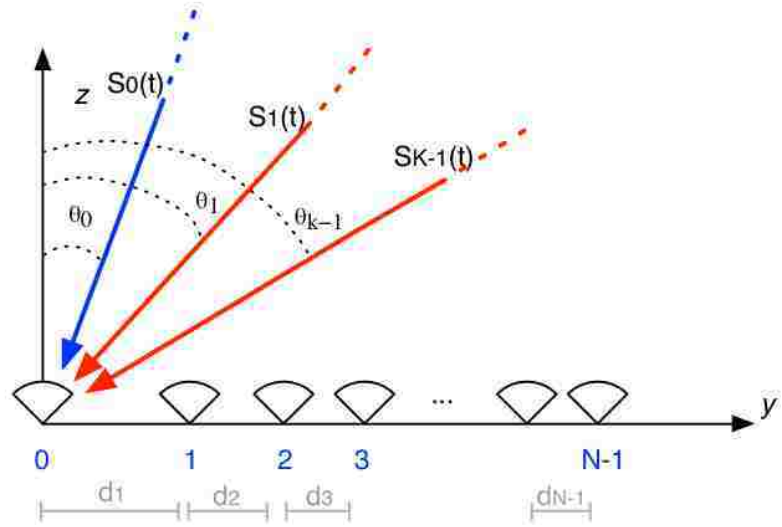
where

$$\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(L)] \tag{3.6a}$$

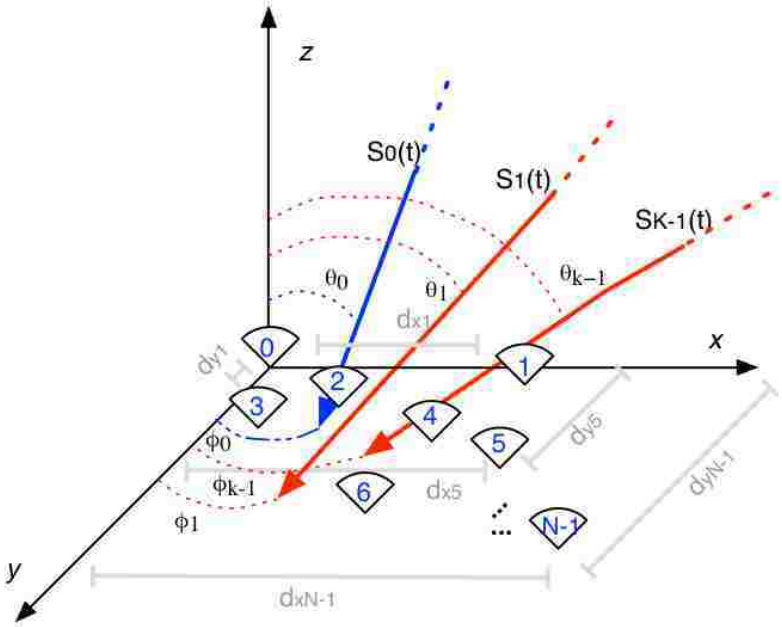
$$\mathbf{S} = [\mathbf{s}(1), \mathbf{s}(2), ..., \mathbf{s}(L)] \tag{3.6b}$$

$$\mathbf{N} = [\mathbf{n}(1), \mathbf{n}(2), ..., \mathbf{n}(L)] \tag{3.6c}$$

The described signal model is illustrated in the Figure 3.2:



(a)



(b)

Figure 3.2: Random signal model for: (a) a random linear array , and (b) a random planar array.

3.2 Direction-of-Arrival Estimation

The direction-of-arrival (DOA) estimation of multiple signals poses a large problem in array signal processing. DOA estimation techniques are divided into four different categories according to the data analysis and implementation used. In this thesis, conventional methods and the subspace-based method will be applied to the randomly distributed array.

3.2.1 Capon Algorithm

One of the most well-known high-resolution non-parametric spectral estimation algorithms is Capon's minimum variance method, also known as the minimum variance distortionless response (MVDR) [7], [4]. The Capon algorithm is an optimization of the delay-and-sum method, or the Bartlett method, and belongs to the conventional methods [1]. Capon's method minimizes the output power with the constraint that the gain in the desired direction is equal to one [8]:

$$\min_{w} \mathbf{w}^H \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{w} \tag{3.7a}$$

$$\text{subject to } \mathbf{w}^H \mathbf{a}^H(\theta_i, \phi_i) = 1 \tag{3.7b}$$

By applying the Lagrange multipliers and optimizing with respect to all the parameters, the optimum weight vector is calculated by [9]:

$$\mathbf{w} = \frac{\mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{a}(\theta, \phi)}{\mathbf{a}^H(\theta, \phi) \mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{a}(\theta, \phi)} \tag{3.8}$$

and consequently, the output Capon power can be obtained by:

$$P_{Capon} = \mathbf{w}^H \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{w} = \frac{1}{\mathbf{a}^H(\theta, \phi) \mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{a}(\theta, \phi)} \tag{3.9}$$

Capon's method improves the resolution problem presented by the Bartlett method. Nevertheless, this method fails when the incoming signals are correlated or when the SOI and the SNOI have the same angle of arrival.

3.2.2 MUSIC Algorithm

The MUltiple SIgnal Classification (MUSIC) method is the most studied algorithm within the signal subspace category. MUSIC is based on the decomposition of the eigenvector structure for the input covariance matrix [10].

After the antenna elements receive the desired signals and the interference signals, in addition to the noise signal, the MUSIC algorithm calculates the input covariance matrix.

$$\mathbf{R}_{\mathbf{x}(k)\mathbf{x}k)} = E[\mathbf{x}(k)\mathbf{x}(k)] = \mathbf{A}\mathbf{R}_{\mathbf{s}(k)\mathbf{s}(k)}\mathbf{A} + \sigma_n^2 I \tag{3.10}$$

where $A = [a(\theta_0, \phi_0), a(\theta_1, \phi_1), ..., a(\theta_{K-1}, \phi_{K-1})]$ is the steering vector,
$S = [S_0, S_1, ..., S_{k-1}]$ is the signal vector, σ_n^2 is the noise variance, and I is the identity matrix. By applying the eigendecomposition of the covariance matrix, Equation 3.10 can be rewritten as follows [11]:

$$\mathbf{R}_{\mathbf{x}(k)\mathbf{x}k)} = \widehat{\mathbf{Q}}_x \Lambda \widehat{\mathbf{Q}}_x^H = \widehat{\mathbf{Q}}_s \Lambda_s \widehat{\mathbf{Q}}_s^H + \widehat{\mathbf{Q}}_n \Lambda_n \widehat{\mathbf{Q}}_n^H \tag{3.11}$$

where $\mathbf{Q}_x = [q_1, q_2, ..., q_N]$ is the eigenvector of the $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}k)}$, $\mathbf{Q}_s = [q_1, q_2, ..., q_k]$ is the eigenvector of the $\mathbf{R}_s$, $\mathbf{Q}_n = [q_{k+1}, q_{k+2}, ..., q_N]$ is the eigenvector of the noise covariance, $\mathbf{R}_n$, $\Lambda_s = diag[\lambda_1, \lambda_2, ..., \lambda_k]$ is the eigenvalue of the signal S, and $\Lambda_n = diag[\lambda_{k+1}, \lambda_{k+2}, ..., \lambda_N]$ is the eigenvalue of the noise [9]. The eigenvectors $\mathbf{Q}_s$ and $\mathbf{Q}_n$ form an orthogonal base. When determining the subspace, the DOAs of the incoming signals are estimated by calculating the power spectrum.

$$P_{MUSIC} = \frac{\mathbf{a}^H(\theta, \phi)\mathbf{a}(\theta, \phi)}{\mathbf{a}^H(\theta, \phi)\mathbf{Q}_n\mathbf{Q}_n^H\mathbf{a}(\theta, \phi)} \tag{3.12}$$

Due to the orthogonality between the steering vector and noise vector, the MUSIC power spectrum shows the estimated DOA for the incoming signals. The MUSIC algorithm is a modification of the Capon algorithm in which the eigendecomposition of the signal is considered zero [9]:

$$\begin{aligned} P_{Capon} &= \frac{1}{\mathbf{a}^H(\theta,\phi)\mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{a}(\theta,\phi)} \\ &= \frac{1}{\mathbf{a}^H(\theta,\phi)(\widehat{\mathbf{Q}}_s\Lambda_s\widehat{\mathbf{Q}}_s^H + \widehat{\mathbf{Q}}_n\Lambda_n\widehat{\mathbf{Q}}_n^H)\mathbf{a}(\theta,\phi)} \\ &= \frac{1}{\mathbf{a}^H(\theta,\phi)(\widehat{\mathbf{Q}}_n\Lambda_n\widehat{\mathbf{Q}}_n^H)\mathbf{a}(\theta,\phi)} \end{aligned} \tag{3.13}$$

The MUSIC algorithm obtains optimum and very precise DOAs; however, this algorithm requires knowledge of the incoming data in addition to the second-order spatial statistics of the interference field and noise.

3.3 Adaptive Beamforming Fundamentals

3.3.1 LMS Algorithm

The least-mean-squares (LMS) algorithm is based on the approximation of the steepest-descent method [12]. The optimum weight vector, $\mathbf{w}_{optimum}$= $[w_1, w_2, ..., w_N]^T$ generally called the Wiener weight vector, is obtained by applying the minimum mean-square error (MMSE) criteria, which minimize the mean square error of the reference signals and the outputs of the antenna array, also known as the cost factor.

$$J_{MMSE} = \min_{w}\{E|\varepsilon(k)^2|\}$$

where $\epsilon(k)$ is the error for each observation and is derived from

$$\varepsilon(k) = S(k) - y(k) = S(k) - \mathbf{w}^H \mathbf{x}(k) \tag{3.14}$$

Substituting 3.14 into the cost function:

$$\begin{aligned} J_{MMSE} &= \min_w [E\{|\varepsilon(k)^2|\}] \\ &= \min_w [E\{|S(k) - \mathbf{w}^H \mathbf{x}(k)|^2\}] \\ &= \min_w [E\{S(k)^2\} - 2\mathbf{w}^H E\{S(k)\mathbf{x}(k)\} + \mathbf{w}^H E\{\mathbf{x}(k)\mathbf{x}(k)^H\}\mathbf{w}] \\ &= \min_w [S(k)^2 - 2\mathbf{w}^H \mathbf{r}_{\mathbf{x}(k)S(k)} + \mathbf{w}^H \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}] \end{aligned} \tag{3.15}$$

where $\mathbf{r}_{\mathbf{x}(k)S(k)}$ is the cross-correlation vector between the input signal x(k) and the desired response S(k) and $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}$ is the correlation matrix of the input vector x(k). In order to minimize the cost function 3.15, it is necessary to set the first derivative equal to zero. Thus obtaining Equations 3.16a and 3.16b:

$$\frac{dJ_{MMSE}}{d\mathbf{w}} = 0 \Leftrightarrow -2\mathbf{r}_{\mathbf{x}(k)S(k)} + \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w} = 0 \tag{3.16a}$$

$$\mathbf{w}_{optimum} = \mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{r}_{\mathbf{x}(k)S(k)} \tag{3.16b}$$

The steepest-decent method adjusts the optimum weight vector in the direction of the gradient at each step as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu(-\nabla_k)$$

where μ is a scalar constant that regulates the step constant and ∇ is the gradient vector calculated in Equation 3.16a. As a consequence of substituting the gradient of the cost function vector into the steepest-decent, the LMS algorithm updates the weight vector as the following expression:

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + \mu[-(-2\mathbf{r}_{\mathbf{x}(k)S(k)} + \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k))] \\ &= \mathbf{w}(k) + 2\mu\mathbf{r}_{\mathbf{x}(k)S(k)} - \mu\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) \end{aligned} \tag{3.17}$$

By considering the instantaneous estimates for the cross-correlation and correlation matrices, $\mathbf{r}_{\mathbf{x}(k)S(k)} = E\{S(k)\mathbf{x}(k)\} = S(k)\mathbf{x}(k)$ and $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} = E\{\mathbf{x}(k)\mathbf{x}(k)^H\} = \mathbf{x}(k)\mathbf{x}(k)^H$ respectively, Equation 3.17 can be rewritten as follows:

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) + 2\mu S(k)\mathbf{x}(k) - \mu\mathbf{x}(k)\mathbf{x}(k)^H\mathbf{w}(k) \\ &= \mathbf{w}(k) + \mu\mathbf{x}(k)[S(k) - \mathbf{x}(k)^H\mathbf{w}(k)] \\ &= \mathbf{w}(k) + \mu\mathbf{x}(k)[S(k) - \mathbf{w}(k)^H\mathbf{x}(k)] \end{aligned} \tag{3.18}$$

Replacing Equation 3.14 with 3.18 obtains the recursive relation for updating the weight vector:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\varepsilon(k)\mathbf{x}(k)$$

By rearranging these equations, the LMS algorithm can be written accordingly:

$$y(k) = \mathbf{w}(k)^H\mathbf{x}(k) \tag{3.19a}$$

$$\varepsilon(k) = S(k) - y(k) \tag{3.19b}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\varepsilon(k)\mathbf{x}(k) \tag{3.19c}$$

The necessary and sufficient condition for the LMS algorithm convergence in the mean-squared sense, which is derived by choosing the step constant, μ, bounded by the following interval [13]:

$$0 < \mu < \frac{2}{\lambda_{max}}$$

where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}$. The maximum eigenvalue is related to the total power of signal $\mathbf{x}$ by the following expression:

$$\lambda_{max} \leq trace\{\mathbf{R}_{\mathbf{xx}}\} \text{ where } trace\{\mathbf{R}_{\mathbf{xx}}\} = \sum_{i=1}^{N} \varepsilon\{x_i^2\}$$

3.3.2 Leaky LMS Algorithm

The leaky least-mean-square (Leaky LMS) algorithm is a variation of the LMS algorithm, which stabilizes the LMS algorithm's undamped modes, forcing them to

zero by introducing a leakage coefficient. These undamped modes appear when the largest eigenvalue is equal to zero, $\lambda_{max} = 0$; as a consequence, the step constant will be infinity, $\mu = \infty$ [14]. To overcome this problem, the Tikhonov regularization method is applied, which consists of adding a scaled identity matrix [15].

As a consequence, the leaky LMS algorithm's cost function used to obtain the optimum weight vector, $\mathbf{w}_{optimum} = [w_1, w_2, ..., w_N]^H$, is obtained by [16]:

$$J_{MMSE} = min\{\epsilon(k)^2 + \gamma \mathbf{w}(k)^H \mathbf{w}(k)\}$$

where the term $\gamma \mathbf{w}(k)^H \mathbf{w}(k)$ is the Tikhonov regularization, and $0 < \gamma \ll 1$ is the regularization parameter, also known as the leakage coefficient. As with the LMS algorithm, the leaky LMS algorithm is based on the steepest-decent method; for this reason, the optimum weight vector is obtained by following a similar procedure as in Section 3.3.1:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(-\nabla_k J_{MMSE})$$

In this case, ∇_k has the following expression:

$$\nabla_k = \frac{dJ_{MMSE}}{dw} = -2\mathbf{x}(k)\mathbf{d}(k) + \mathbf{x}(k)\mathbf{x}(k)^H \mathbf{w}(k) + \gamma \mathbf{w}(k) \tag{3.20}$$

By substituting this equation into the steepest-decent equation:

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{w}(k) + \mu(+2\mathbf{x}(k)\mathbf{d}(k) - \mathbf{x}(k)\mathbf{x}(k)^H \mathbf{w}(k) - \gamma \mathbf{w}(k)) && \text{(3.21a)} \\
&= (1 - \mu\gamma)\mathbf{w}(k) + \mu \mathbf{x}(k)\mathbf{d}(k) - \mu \mathbf{x}(k)\mathbf{x}(k)^H \mathbf{w}(k) && \text{(3.21b)} \\
&= (1 - \mu\gamma)\mathbf{w}(k) + \mu \mathbf{x}(k)(\mathbf{d}(k) - x(k)^H \mathbf{w}(k)) && \text{(3.21c)} \\
&= (1 - \mu\gamma)\mathbf{w}(k) + \mu \mathbf{x}(k)\epsilon(k) && \text{(3.21d)}
\end{aligned}$$

By taking the expected value of both sides of Equation 3.21a and considering the limit as n approaches infinity, one can observe that the tap-weight converges in mean

to a biased solution:

$$\begin{aligned} E\{\mathbf{w}(k+1)\} &= E\{\mathbf{w}(k) + \mu(2\mathbf{x}(k)\mathbf{d}(k) - \mathbf{x}(k)\mathbf{x}(k)^H\mathbf{w}(k) - \gamma\mathbf{w}(k))\} \\ &= (1 - \mu(\gamma + \mathbf{x}(k)\mathbf{x}(k)^H))E\{\mathbf{w}(k)\} + \mu(\mathbf{x}(k)\mathbf{d}(k)) \\ &= (1 - \mu(\gamma + \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}))E\{\mathbf{w}(k)\} + \mu\mathbf{r}_{\mathbf{x}(k)S(k)} \\ &= (\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} + \gamma)^{-1}\mathbf{r}_{\mathbf{x}(k)S(k)} \end{aligned} \tag{3.22}$$

As $n \to \infty \Rightarrow \mathbf{w}(k+1) \cong \mathbf{w}(k)$ consequently:

$$\lim_{n\to\infty} E\{\mathbf{w}(k+1)\} = (\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} + \gamma)^{-1}\mathbf{r}_{\mathbf{x}(k)S(k)}$$

By comparing this equation to Equation 3.18, one observes that the correlation matrix $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}$ of the LMS algorithm is substituted with $\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} + \gamma\mathbf{I}$; for this reason, the leaky LMS algorithm's step constant μ is bounded by the following interval:

$$0 < \mu < \frac{2}{\lambda_{max} + \gamma}$$

Due to the leakage coefficient condition, $0 < \gamma \ll 1$, the leaky LMS algorithm suppresses the LMS unstable modes.

By reorganizing all of the equations, the leaky LMS algorithm can be written as follows:

$$y(k) = \mathbf{w}(k)^H\mathbf{x}(k) \tag{3.23a}$$

$$\varepsilon(k) = S(k) - y(k) \tag{3.23b}$$

$$\mathbf{w}(k+1) = (1 - \mu\gamma)\mathbf{w}(k) + \mu\mathbf{x}(k)\epsilon(k) \tag{3.23c}$$

3.3.3 Normalized LMS Algorithm

A normalized LMS algorithm (NLMS) is an adaptation of the LMS algorithm, which overcomes the LMS algorithm's stability problems due to its weight vector's actualization, $\mathbf{w}(k+1)$, being directly proportional to the input vector $\mathbf{x}(k)$ [3]. For this reason, if the total power of the input signal ($trace\{\mathbf{R_{xx}}\} = \sum_{i=1}^{N} \varepsilon\{x_i^2\}$) is high, then the step constant will not be bounded, thus making the algorithm stable. To correct this problem, the weight vector must be normalized in each iteration with respect to the squared Euclidean norm of the input signal as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\tilde{\mu}}{\|\mathbf{x}(k)\|^2}\mathbf{x}(k)\epsilon(k) \tag{3.24}$$

In this actualization, one observes that the weight vector at time $k+1, \mathbf{w}(k+1)$, updates with minimum variation with respect to the known value at time $k, \mathbf{w}(k)$. Equation 3.24 can be seen as the 3.19c of the LMS algorithm with a time-varying step-size parameter:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)\mathbf{x}(k)\epsilon(k) \tag{3.25}$$

where $\mu(k) = \frac{\tilde{\mu}}{\|\mathbf{x}(k)\|^2}$. The NLMS algorithm converges in the mean square sense only if the constant $\tilde{\mu}$ satisfies the following condition:

$$0 < \tilde{\mu} < 2$$

To overcome the undamped modes that occur when the input signal, $\mathbf{x}(k)$, is very small, it is recommended that one introduce an offset parameter, $\delta > 0$, into the time-varying step-size parameter:

$$\mu(k) = \frac{\tilde{\mu}}{\|\mathbf{x}(k)\|^2 + \delta}$$

In comparison to the LMS algorithm, the normalized LMS algorithm improves the time of convergence of the weight vector; however, the computational complexity of the NLMS is higher.

3.3.4 Generalized Normalized LMS Algorithm

To overcome the divergence and low performance problems of LMS and NLMS algorithms respectively when the input signal has a large dynamic range, Mandic [17] derived the generalized normalized LMS algorithm (GNLMS) based on the NLMS algorithm. The GNLMS algorithm makes the offset or compensation term, δ, step-size gradient adaptive as follows:

$$\delta(k+1) = \delta(k) - \rho \nabla_{\delta(k-1)} E(k) \tag{3.26}$$

By applying the chain rule, the gradient $\nabla_{\delta(k-1)} E(k)$ is expressed as:

$$\begin{aligned} \frac{\partial E(k)}{\partial \delta(k-1)} &= \frac{\partial E(k)}{\partial \epsilon(k)} \frac{\partial \epsilon(k)}{\partial y(k)} \frac{\partial y(k)}{\partial \mathbf{w}(k)} \frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} \frac{\partial \mu(k-1)}{\partial \delta(k-1)} \\ &= \frac{\epsilon(k)\epsilon(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{(\|\mathbf{x}(k-1)\|_2^2 + \delta(k-1))^2} \end{aligned} \tag{3.27}$$

As a consequence, the GNLMS algorithm can be expressed as:

$$y(k) = \mathbf{w}(k)^H \mathbf{x}(k) \tag{3.28a}$$

$$\epsilon(k) = S(k) - y(k) \tag{3.28b}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)\mathbf{x}(k)\epsilon(k) \tag{3.28c}$$

$$\mu(k) = \frac{\tilde{\mu}}{\|\mathbf{x}(k)\|_2^2 + \delta(k)} \tag{3.28d}$$

$$\delta(k) = \delta(k-1) - \rho \frac{\epsilon(k)\epsilon(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{(\|\mathbf{x}(k-1)\|_2^2 + \delta(k-1))^2} \tag{3.28e}$$

Equation 3.28e shows that the GNLMS algorithm adapts to the learning rate in accordance with the dynamics of the input signal, thus resulting in faster convergence.

The convergence in the mean square sense of the GNLMS algorithm is obtained if the adaptive step size $\mu(k)$ is bounded by the limits of the step size of the NLMS algorithm, $0 < \tilde{\mu} < 2$. Optimizing the convergence point doubles the computational complexity of the NLMS algorithm.

3.3.5 Constrained LMS Algorithm

A constrained LMS algorithm, also known as Frost's algorithm, is based on the approximation of the stochastic gradient-descent algorithm [18], which minimizes the expected value of the output power subject to a specific restriction:

$$\min_{w} \mathbf{w}^H \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{w} \tag{3.29a}$$

$$\text{subject to } \mathbf{C}^H \mathbf{w} = \mathcal{F} \tag{3.29b}$$

where $\mathbf{C}$ is the constraint matrix defined as: $\mathbf{C} \triangleq [c_1, ..., c_J]$ with J being the number of restrictions and $\mathcal{F}$ being a J-dimensional vector of the weights of the angle of arrival of the signals: $\mathcal{F} \triangleq \begin{bmatrix} f_1 \\ f_2 \\ ... \\ f_J \end{bmatrix}$. For this reason, a constrained LMS algorithm requires that the DOA and the frequency band of interest be known.

The constrained cost function is obtained by applying Lagrange multipliers to constraint function 3.29b:

$$J_{constrained}(w) = \frac{1}{2} \mathbf{w}^H \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)} \mathbf{w} + \lambda^H (\mathbf{C}^H \mathbf{w} - \mathcal{F}) \tag{3.30}$$

where $\lambda \in \mathbb{C}^K$ is the Lagrange multiplier vector.

The optimum weight vector,$\mathbf{w}_{optimum}$, is obtained by minimizing the cost function and setting it equal to zero.

$$\frac{\partial J_{constrained}(w)}{\partial \mathbf{w}} = 0 \Leftrightarrow \tag{3.31}$$

$$\frac{\partial J_{constrained}(w)}{\partial \mathbf{w}} = \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w} + \lambda^H \mathbf{C}^H = \mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w} + \mathbf{C}\lambda = 0 \tag{3.32}$$

$$\mathbf{w}_{optimum} = -\mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{C}\lambda \tag{3.33}$$

By substituting this equation into constraint equation 3.29b, which must satisfy the weight vector, the Lagrange multiplier vector can be obtained as follows:

$$\mathbf{C}^H\mathbf{x} = \mathcal{F} = \mathbf{C}^H(-\mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{C}\lambda) \Rightarrow$$

$$\lambda = -\left[\mathbf{C}^H\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{C}\right]^{-1}\mathcal{F} \tag{3.34}$$

As a consequence, Equation 3.32 can be rewritten as follows:

$$\mathbf{w}_{optimum} = \mathbf{R}^{-1}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{C}[\mathbf{C}^H\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{C}]^{-1}\mathcal{F} \tag{3.35}$$

The optimum weight vector is updated recursively by the gradient-descent algorithm as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(-\nabla_k J_{constrained}) \tag{3.36}$$

Replacing Equation 3.32 into the equation above results in:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu(\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) + \mathbf{C}\lambda) \tag{3.37}$$

It is necessary to choose Lagrange multipliers that satisfy the constraints at time $k+1$:

$$\begin{aligned}\mathcal{F} &= \mathbf{C}^H\mathbf{w}(k+1) = \mathbf{C}^H(\mathbf{w}(k) - \mu(\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) + \mathbf{C}\lambda)) \\ &= \mathbf{C}^H\mathbf{w}(k) - \mu\mathbf{C}^H\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) - \mu\mathbf{C}^H\mathbf{C}\lambda\end{aligned} \tag{3.38}$$

Lagrange multipliers can be obtained from this equation as follows:

$$\lambda = -[\mathcal{F} - \mathbf{C}^H\mathbf{w}(k) + \mu\mathbf{C}^H\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k)](\mu\mathbf{C}^H\mathbf{C})^{-1} \tag{3.39}$$

Using this equation, the weight update vector can be rewritten as:

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) - \mu\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) \\ &\quad - \mu\mathbf{C}[\mathcal{F} - \mathbf{C}^H\mathbf{w}(k) + \mu\mathbf{C}^H\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k)](\mu\mathbf{C}^H\mathbf{C})^{-1} \\ &= \mathbf{w}(k) - \mu[I - \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{C}^H]\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k) \\ &\quad + \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}[\mathcal{F} - \mathbf{C}^H\mathbf{w}(k)] \end{aligned} \tag{3.40}$$

By defining $P \triangleq I - \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{C}^H$ and $F \triangleq \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathcal{F}$, Equation 3.40 can be expressed as:

$$\mathbf{w}(k+1) = P[\mathbf{w}(k) - \mu\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)}\mathbf{w}(k)] + F \tag{3.41}$$

By rearranging all of the equations, the constrained LMS algorithm can be written as follows:

$$\text{Initialization} \Rightarrow \mathbf{w}(0) = F = \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathcal{F} \tag{3.42a}$$

$$\mathbf{w}(k+1) = P[\mathbf{w}(k) - \mu\mathbf{y}(k)\mathbf{x}(k)] + F \tag{3.42b}$$

The constrained LMS algorithm is considered a blind-algorithm because it does not required the knowledge of the desired signal. However, the computational implementation of this algorithm is more complex. Frost demonstrates that a constrained LMS algorithm will converge if step-size constant μ satisfies this bound condition [18]:

$$0 < \mu < \frac{2}{3trace(\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)})}$$

3.3.6 Recursive Least-Squares Algorithm

A recursive least-squares (RLS) algorithm calculates the Wiener solution by using the method of least squares (LS) [19], which recursively minimizes the sum of error

squares over a particular period of time [3]:

$$\min_{w}\{J_{LS}\} = \min_{w} \sum_{i=1}^{k} \beta(k)|e(k)|^2 \tag{3.43}$$

where $\beta(k)$ is the forgetting factor and $e(k)$ is the difference between the desired signal $S(k)$ and the output signal $y(k) = \mathbf{w}^H\mathbf{x}(k)$.

The forgetting factor provides different weights according to the time instant. In this case, the exponential weight factor or forgetting factor is considered, which emphasizes the most recent data and forgets the past data:

$$\beta(k) = \lambda^{k-i}, \qquad i = 0, 1, ..., k \tag{3.44}$$

where $0 < \lambda \leqslant 1$. In the case where $\lambda = 1$, the stationary environment, all data past and present have the same weight in accordance with an infinite memory system. Placing Equation 3.44 into the cost function produces the following result:

$$J_{LS} = \sum_{i=1}^{k} \lambda^{k-i}|S(k) - \mathbf{w}^H\mathbf{x}(k)|^2 \tag{3.45}$$

The optimum weight vector is obtained by differentiating the cost function with respect to $\mathbf{w}$ and setting it equal to zero:

$$\frac{\partial J_{SL}}{\partial \mathbf{w}} = -\sum_{i=1}^{k} \lambda^{k-1} S(k)x(k) + \sum_{i=1}^{k} \mathbf{x}(k)\mathbf{x}(k)^H\mathbf{w}(k) = 0 \tag{3.46}$$

$$\mathbf{w}_{optimum}(k) = \sum_{i=1}^{k} \lambda^{k-i} S(k)x(k)[\sum_{i=1}^{k} \lambda^{k-i}\mathbf{x}(k)\mathbf{x}(k)^H\mathbf{w}(k)]^{-1} \tag{3.47}$$

After defining the correlation matrix as $\mathbf{R}(k) = \sum_{i=1}^{k} \lambda^{k-i}\mathbf{x}(k)\mathbf{x}(k)^H$ and the cross-correlation matrix as $\mathbf{p}(k) = \sum_{i=1}^{k} \lambda^{k-1}x(k)d^*(k)$, Equation 3.47 can be rewritten as:

$$\mathbf{w}_{optimum}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k) \tag{3.48}$$

In order to obtain the recursive implementation of the weight vector, it is necessary to write the correlation and cross-correlation matrix recursively:

$$\begin{aligned}\mathbf{R}(k) &= \sum_{i=1}^{k} \lambda^{k-i}\mathbf{x}(k)\mathbf{x}(k)^H = \lambda \sum_{i=1}^{k-1} \lambda^{k-i-1}\mathbf{x}(k)\mathbf{x}(k) + \mathbf{x}(k)\mathbf{x}^H(k) \\ &= \lambda\mathbf{R}(k-1) + \mathbf{x}(k)\mathbf{x}^H(k) \qquad (3.49)\end{aligned}$$

$$\mathbf{p}(k) = \sum_{i=1}^{k} \lambda^{k-1} S(k)x(k) = \lambda\mathbf{p}(k-1) + \mathbf{x}(k)\mathbf{d}^*(k) \qquad (3.50)$$

Applying the matrix inversion lemma to the correlation matrix, $\mathbf{R}(k)$, results in the following:

$$\mathbf{R}^{-1}(k) = \lambda\mathbf{R}^{-1}(k-1) - \frac{\lambda^{-2}\mathbf{R}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)}{(1+\lambda^{-1}\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k))} \qquad (3.51)$$

After defining the gain vector as:

$$\mathbf{k}(k) \triangleq \frac{\lambda^{-1}\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{1+\lambda^{-1}\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k)} \qquad (3.52)$$

$$\Rightarrow \mathbf{k}(k)[1+\lambda^{-1}\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k)] = \lambda^{-1}\mathbf{R}^{-1}(k-1)\mathbf{x}(k)$$

$$\Rightarrow \mathbf{k}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1)\mathbf{x}(k) - \mathbf{k}(k)\lambda^{-1}\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k) \qquad (3.53)$$

the recursive correlation matrix can be rewritten as follows:

$$\mathbf{R}^{-1}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1) - \lambda^{-1}\mathbf{k}(k)\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1) \qquad (3.54)$$

Replacing Equation 3.54 demonstrates that the gain vector can be expressed by:

$$\mathbf{k}(k) = \mathbf{R}^{-1}(k)\mathbf{x}(k) \qquad (3.55)$$

Substituting Equations 3.50, 3.54, and 3.55 into the optimum weight vector, produces the following result:

$$\begin{aligned}\mathbf{w}_{optimum} &= \mathbf{R}^{-1}(k)\lambda\mathbf{p}(k-1) + \mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{d}^*(k) \\ &= \mathbf{R}^{-1}(k-1)\mathbf{p}(k-1) - \mathbf{k}(k)\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{p}(k-1) + \mathbf{k}(k)\mathbf{d}^*(k) \\ &= \mathbf{w}(k-1) + \mathbf{k}(k)[\mathbf{d}^*(k) - \mathbf{x}^H(k)\mathbf{w}(k-1)] \\ &= \mathbf{w}(k-1) + \mathbf{k}(k)\epsilon^*(k) \qquad (3.56)\end{aligned}$$

where $\epsilon(k) = \mathbf{d}(k) - \mathbf{w}^H(k-1)\mathbf{x}(k)$ is the estimation error.
After rearranging all of the equations, the RLS algorithm can be summarized as follows:

$$\mathbf{k}(k) = \frac{\lambda^{-1}\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{1+\lambda^{-1}\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k)} \tag{3.57a}$$

$$\epsilon(k) = \mathbf{d}(k) - \mathbf{w}^H(k-1)\mathbf{x}(k) \tag{3.57b}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{k}(k)\epsilon^*(k) \tag{3.57c}$$

$$\mathbf{R}^{-1}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1) - \lambda^{-1}\mathbf{k}(k)\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1) \tag{3.57d}$$

The forgetting factor, λ, determines the convergence rate and the stability of the system. On the one hand, when the forgetting factor is close to one, the algorithm obtains low misadjustment and good stability, but it is not able to track the input signal correctly. On the other hand, when the forgetting factor is close to zero, the RLS algorithm improves its tracking capability by having increased the misadjustment and stability problems. Nevertheless, the complexity has been increased.

When comparing the LMS algorithm to the RLS algorithm, LMS algorithm converges to the Wiener solution in mean, but the weight vector presents a variance that is directly proportional to the step-size parameter, μ. Nevertheless, the RLS algorithm converges both in mean and variance; thus, the RLS algorithm improves the slow performance of the LMS algorithm. In general, the RLS algorithm converges one order of magnitude faster than the LMS algorithm; nevertheless, the computational complexity has been increased.

3.3.7 Variable Forgetting Factor Recursive Least Squares Algorithm

A variable forgetting factor recursive least squares (VFF-RLS) algorithm is an adaptation of the RLS algorithm, which considers a time-variable forgetting factor, $\lambda(k)$, with the unique purpose of improving RLS performance [20].

The derivation of the VFF-RLS algorithm is based on recovering the system noise in the error signal of the adaptive filter. Considering the a priori estimation error and the weight vector update from the RLS algorithm, in Equation 3.57b and Equation 3.57c respectively, the posteriori error can be expressed as follows:

$$\begin{aligned}\mathbf{e}(k) &= \mathbf{d}(k) - \mathbf{w}^H(k)\mathbf{x}(k) = \mathbf{d}(k) - \mathbf{w}^H(k-1)\mathbf{x}(k) + \epsilon(k)\mathbf{x}^H(k)\mathbf{k}(k) \\ &= \epsilon(k)[1 - \mathbf{x}^H(k)\mathbf{k}(k)] \end{aligned} \tag{3.58}$$

Defining $E\{\mathbf{e}^2(k)\} = \sigma_v^2$, where $E\{v^2(k)\} = \sigma_v^2$ is the power of the system noise and considering the Equation 3.57a, the expected value of the a posteriori error estimation can be obtained by:

$$\begin{aligned} & E\left\{\left[1 - \frac{\mathbf{x}^H(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{\lambda(k) + \mathbf{x}^H(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}\right]^2\right\} \\ &= \left\{\left[1 - \frac{q(k)}{\lambda(k) + q(k)}\right]^2\right\} = \frac{\sigma_v^2}{\sigma_e^2(k)} \end{aligned} \tag{3.59}$$

where $q(k) = \mathbf{x}^H(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)$.
After solving the equation above, the time-variable forgetting factor can be obtained by:

$$\lambda(k) = \frac{\sigma_q(n)\sigma_v}{\sigma_e(n) - \sigma_v} \tag{3.60}$$

where $E\{q^2(k)\} = \sigma_q^2$.

To recursively estimate these power vectors, the following expressions can be used:

$$\widehat{\sigma}_e^2(k) = \alpha\widehat{\sigma}_e^2(k-1) + (1-\alpha)e^2(k) \tag{3.61a}$$

$$\widehat{\sigma}_q^2(k) = \alpha\widehat{\sigma}_q^2(k-1) + (1-\alpha)q^2(k) \tag{3.61b}$$

$$\widehat{\sigma}_v^2(k) = \beta\widehat{\sigma}_v^2(k-1) + (1-\beta)e^2(k) \tag{3.61c}$$

Where $\alpha = 1 - \frac{1}{K_\alpha L}$ and $\beta = 1 - \frac{1}{K_\beta L}$, with $K_\beta > K_\alpha > 2$ being the weighting factors. Rearranging all of these equations, the VFF-RLS algorithm can be expressed as follows:

$$\mathbf{k}(k) = \frac{\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{\lambda(k) + \mathbf{x}(k)^H\mathbf{R}^{-1}(k-1)\mathbf{x}(k)} \tag{3.62a}$$

$$\epsilon(k) = \mathbf{d}(k) - \mathbf{w}^H(k-1)\mathbf{x}(k) \tag{3.62b}$$

$$\lambda(k) = \frac{\sigma_q(n)\sigma_v}{\sigma_e(n) - \sigma_v} \tag{3.62c}$$

$$\widehat{\sigma}_e^2(k) = \alpha\widehat{\sigma}_e^2(k-1) + (1-\alpha)e^2(k) \tag{3.62d}$$

$$\widehat{\sigma}_q^2(k) = \alpha\widehat{\sigma}_q^2(k-1) + (1-\alpha)q^2(k) \tag{3.62e}$$

$$\widehat{\sigma}_v^2(k) = \beta\widehat{\sigma}_v^2(k-1) + (1-\beta)e^2(k) \tag{3.62f}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{k}(k)\epsilon^*(k) \tag{3.62g}$$

$$e(k) = \mathbf{d}(k) - \mathbf{w}^H(k)\mathbf{x}(k) \tag{3.62h}$$

$$\mathbf{R}^{-1}(k) = \lambda^{-1}(k)\mathbf{R}^{-1}(k-1) - \lambda^{-1}(k)\mathbf{k}(k)\mathbf{x}(k)^H\mathbf{R}^{-1}(k-1) \tag{3.62i}$$

As Paleologu states in *"A Robust Variable Forgetting Factor Recursive Least-Squares Algorithm for System Identification"* [20], the VFF-RLS algorithm improves the LMS and RLS algorithms' performance for tracking and misadjustment for stationary and non-stationary input signals.

Chapter 4

Experiments

In order to analyze and study the performance of the DOA and the adaptive beamforming algorithms for a randomly spaced smart antenna array, the algorithms studied in the Chapter 3 were tested using MATLAB®. The DOA and adaptive beamforming algorithms are illustrated in the Appendix A.
This chapter presents the results obtained by applying the DOA and adaptive beamforming algorithms into a different array environments and applications.

4.1 Study of Array Processing Algorithms for Random Arrays

4.1.1 Comparison of DOA Algorithms

Two different scenarios were taken into consideration for analyzing and studying the Capon and the MUSIC algorithms in a random array environment.

Scenario 1:

In the first scenario, the signal model was composed by $N = 8$ isotropic antenna elements randomly spaced along the x-axis (LRA model) and 3-narrowband signals, one desired and two interferences, arriving from $\theta_0 = 0\,^\circ$, $\theta_1 = -30\,^\circ$, and $\theta_2 = 30\,^\circ$ at 2-GHz frequency. The power of the incoming signals was 20dB and 10dB for the desired and interference signals respectively.

The distribution of the element was obtained as shown in Figure 4.1:

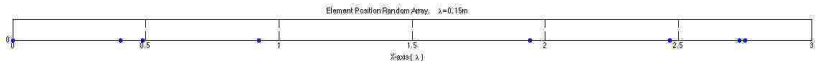


Figure 4.1: Random linear antenna distribution.

One observes that the elements in a randomly spaced array may be separated by more than $\frac{\lambda}{2}$ unlike the linear array model.

Capon Algorithm

The power spectrum obtained by applying the Capon method for this concrete scenario is shown in the Figure 4.2. When analyzing the Capon power spectrum, one can observe that the obtained peaks coincide with the DOA of the incoming signals to the system. These peaks have the following normalized power values:

Table 4.1: Capon detection DOAs:

$\theta_0 = 0\,^\circ$	0 dB
$\theta_1 = -30\,^\circ$	-9.384 dB
$\theta_2 = 30\,^\circ$	-9.825 dB

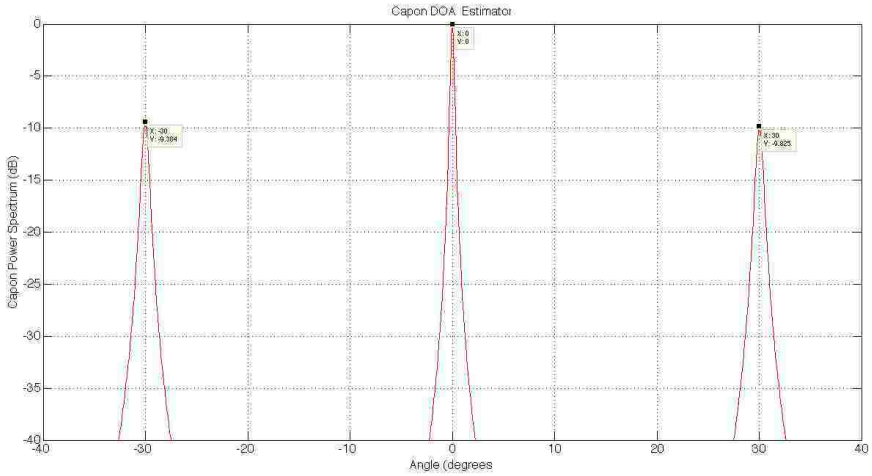


Figure 4.2: Capon angular power spectrum for $\theta_0 = 0\,^\circ$, $\theta_1 = -30\,^\circ$ and $\theta_2 = 30\,^\circ$.

MUSIC Algorithm

The MUSIC algorithm detects the DOAs of the incoming signals for a randomly spaced antenna array, as shown in Figure 4.3. In this case, the peaks obtained by

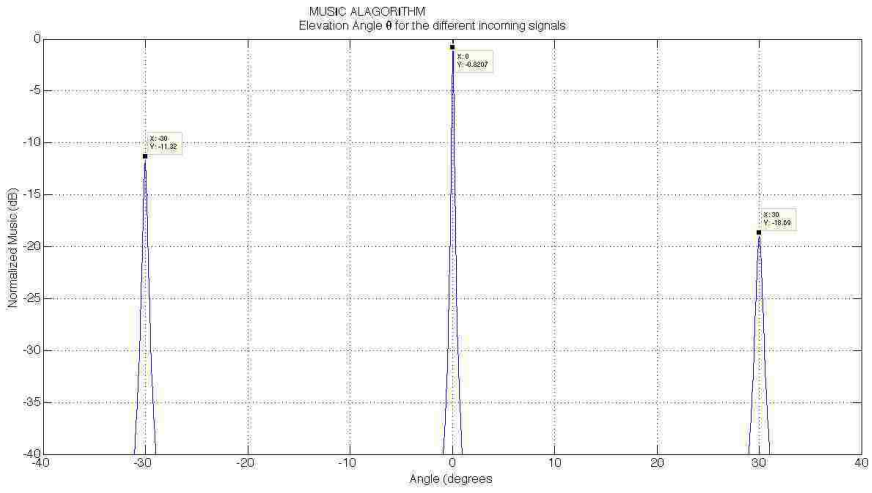


Figure 4.3: MUSIC angular power spectrum for $\theta_0 = 0\,^\circ$, $\theta_1 = -30\,^\circ$ and $\theta_2 = 30\,^\circ$.

applying MUSIC algorithm are shown in Table 4.2.

Table 4.2: MUSIC detection DOAs:

$\theta_0 = 0\,^\circ$	-0.82 dB
$\theta_1 = -30\,^\circ$	-11.32 dB
$\theta_2 = 30\,^\circ$	-18.69 dB

When comparing the Capon and MUSIC algorithms for the first scenario, one observes that the MUSIC algorithm improves the accuracy of the DOAs in addition to arranging them according to their corresponding input power as shown in Figure 4.4.

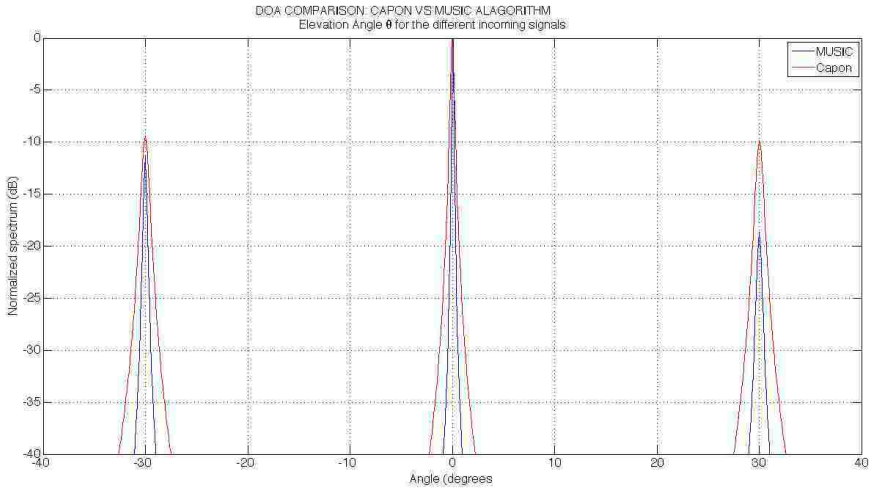


Figure 4.4: Capon and MUSIC algorithm comparison for the first scenario.

Scenario 2:

For the second scenario, a 8-element RPA was considered. In this case the incoming signals arrived from $(\theta_0 = -20\,^\circ, \phi_0 = 10\,^\circ)$, $(\theta_1 = 15\,^\circ, \phi_1 = 0\,^\circ)$ and

($\theta_2 = 48\,^\circ, \phi_2 = 45\,^\circ$). The obtained element distribution is shown in Figure 4.5.

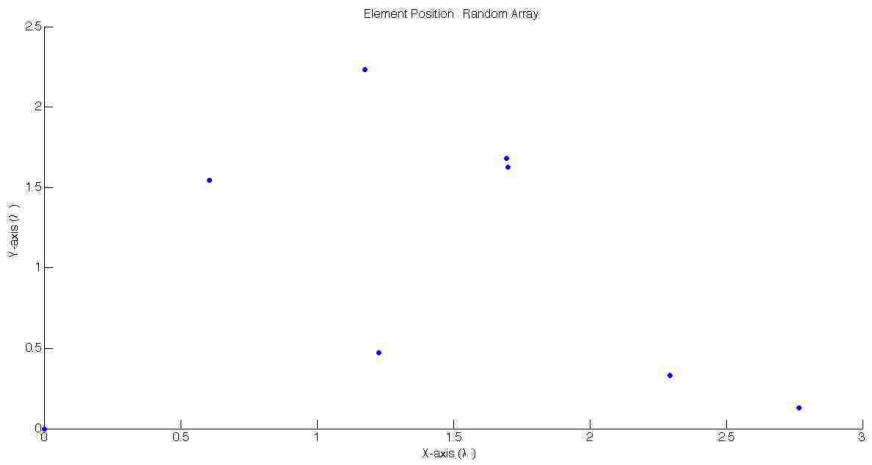


Figure 4.5: Element distribution for the second scenario.

Capon Algorithm

For this planar case, the Capon algorithm detected the following DOAs for the incoming signal as shown in Table 4.3 and Figures 4.6a and 4.6b.

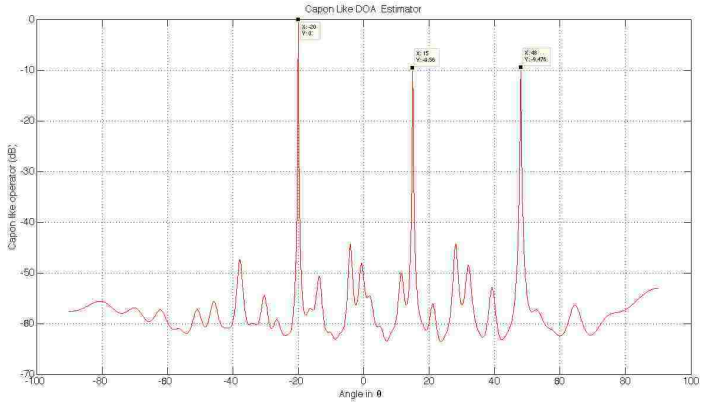
Table 4.3: Capon detection DOAs

(a) DOA's for the elevation angles θ

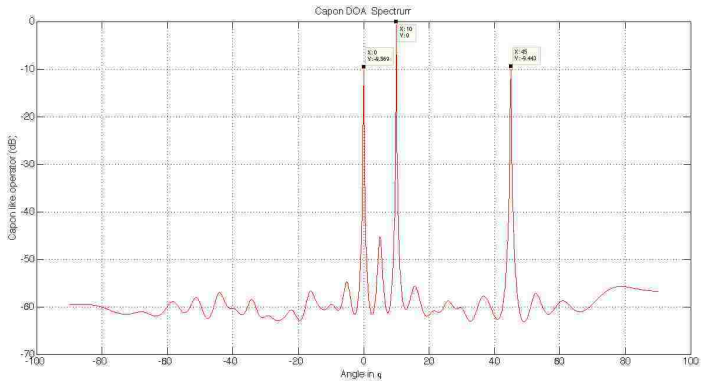
$\theta_0 = -20\,^\circ$	0 dB
$\theta_1 = 15\,^\circ$	-9.56 dB
$\theta_2 = 48\,^\circ$	-9.47 dB

(b) DOA's for the azimuth angles ϕ

$\phi_0 = 10\,^\circ$	0 dB
$\phi_1 = 0\,^\circ$	-9.56 dB
$\phi_2 = 45\,^\circ$	-9.44 dB



(a)



(b)

Figure 4.6: Capon DOAs: (a) elevation angles , and (b) Azimuth angles.

MUSIC Algorithm

The results obtained by applying the MUSIC algorithm are illustrated in the Table 4.4 and Figure 4.8:

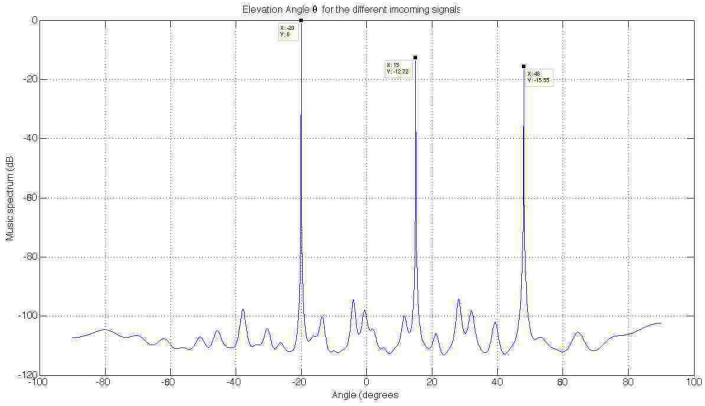
Table 4.4: MUSIC detection DOAs

(a) DOA's for the elevation angles θ

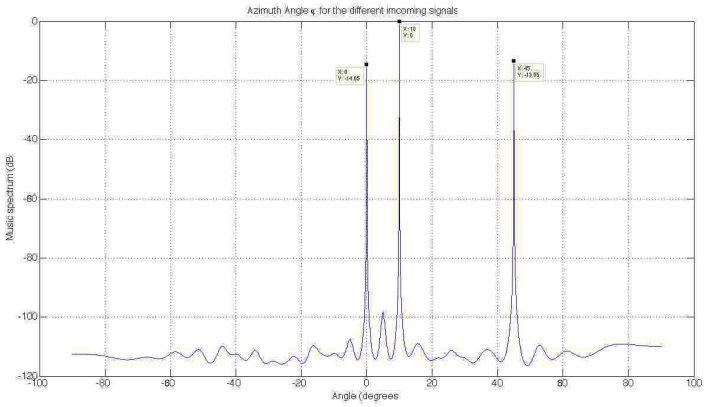
$\theta_0 = -20\,^\circ$	0 dB
$\theta_1 = 15\,^\circ$	-12.72 dB
$\theta_2 = 48\,^\circ$	-15.55 dB

(b) DOA's for the azimuth angles ϕ

$\phi_0 = 10\,^\circ$	0 dB
$\phi_1 = 0\,^\circ$	-14.65 dB
$\phi_2 = 45\,^\circ$	-13.55 dB



(a)

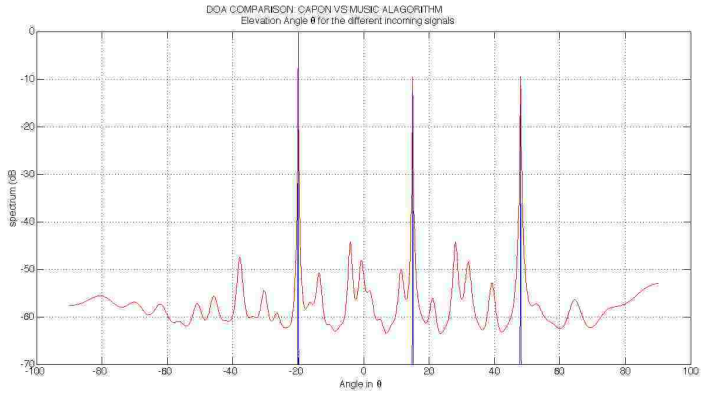


(b)

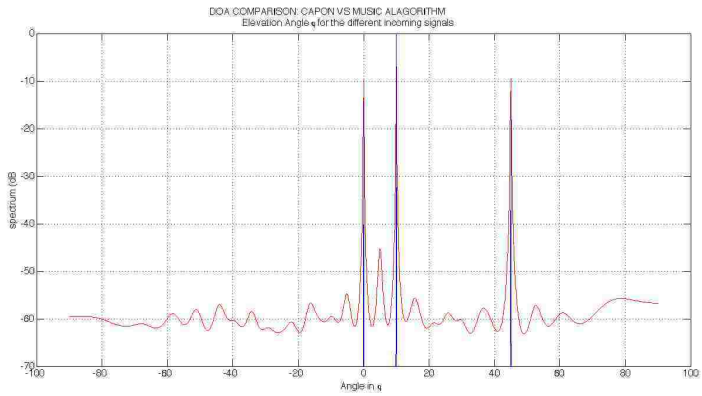
Figure 4.7: MUSIC DOAs: (a) elevation angles, and (b) azimuth angles.

When comparing the Capon and MUSIC algorithm for the second scenario, one

can observe that as in the linear case, the MUSIC algorithm obtains more accurate DOAs; however, this case does not present a significant improvement as in the LRA case.



(a)



(b)

Figure 4.8: Capon and MUSIC comparison: (a) elevation angles, and (b) azimuth angles.

4.1.2 Comparison of: Adaptive Beamforming Algorithm

This section examines the adaptive beamforming studied in Chapter 3.3 with respect to randomly spaced smart antennas. In addition, in order to demonstrate that these algorithms can be applied in a random environment, the performance for different parameters are shown for the linear case.

LMS Algorithm

The LMS algorithm, as previously stated, converges to the Wiener solution in mean with a step-size μ, which controls the rate of the convergence and stability of the algorithm. Three different μ parameters were considered in order to study this convergence rate in a randomly spaced antenna array: $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.0001$.

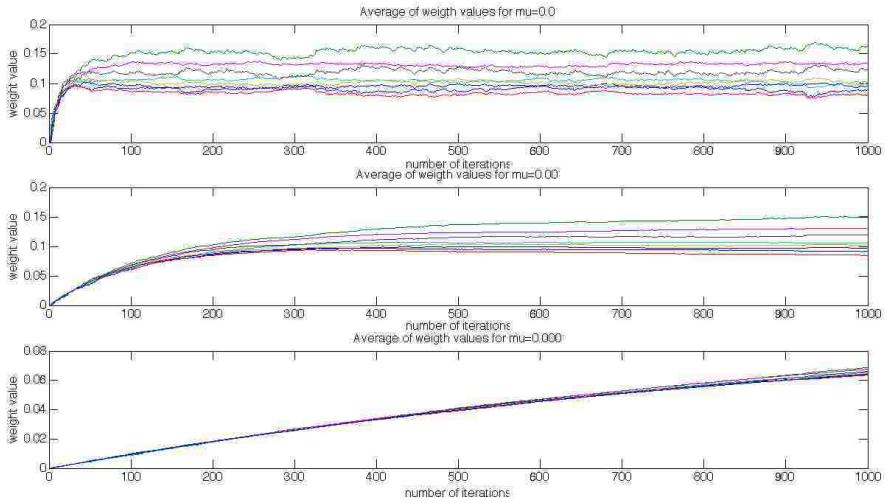


Figure 4.9: Weight vector for LMS algorithm with $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.0001$.

The optimum weight vector performance is illustrated in Figure 4.9. One observes that the smaller the step-size, the slower the convergence. Nevertheless, the step-size needs to satisfy the bound condition, $0 < \mu < \frac{2}{\mu_{max}}$, in order to have a stable system. In this concrete case, when $\mu = 0.01$, the system needs less than 50 snapshots in order to obtain the optimum weight vector; however, when $\mu = 0.0001$, more than 1000 snapshots are required.

The average mean-square error for the different step-size is shown in Figure 4.10. The average mean-square error is directly related to the weight vector; when the system converges to the optimum weight vector values, the mean-square error asymptotically converges to the minimum error.

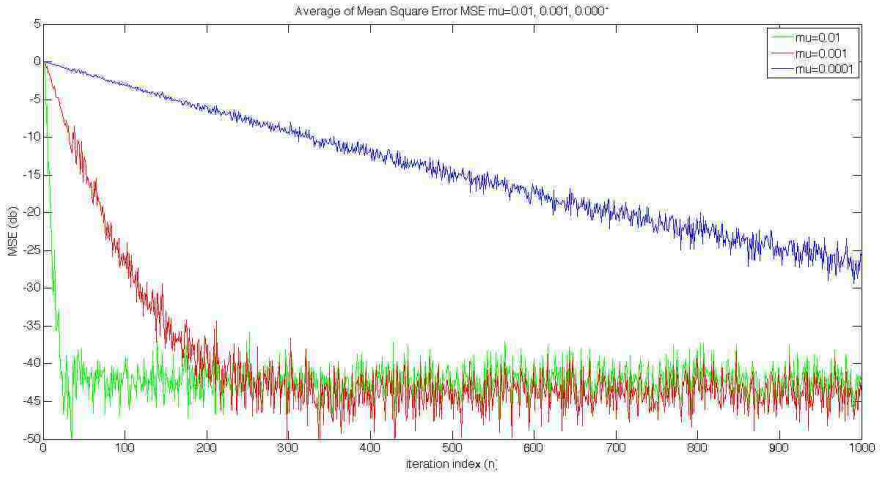


Figure 4.10: Average mean-square error for LMS algorithm with $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.0001$.

The adaptive beamforming for the different step-sizes is shown in Figure 4.11. These three beamforming actions do not present a significant variation since the last snapshot is considered to create the array pattern. However, one can notice that the

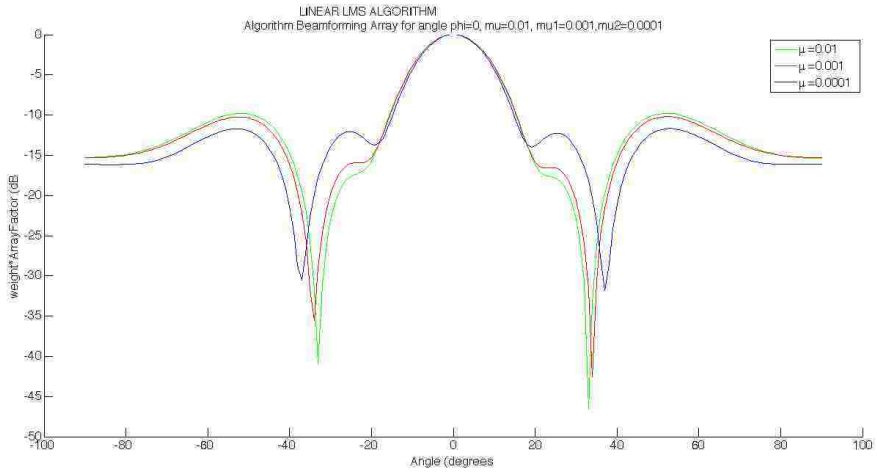


Figure 4.11: Adaptive beamforming LMS algorithm for $\theta_0 = 0\,^\circ$, $\theta_1 = -30\,^\circ$, and $\theta_2 = 30\,^\circ$.

beamforming for $\mu = 0.01$ obtains better performance in suppressing the interference signals.

When considering the variability of the antenna element position due to the random distribution, a statistical study was applied to analyze the generated beamforming according to the antenna position. For this analysis the signal model is composed by 10-elements randomly spaced along x-axis and 3-narrowband incoming signals, one desired at $\theta_0 = 0\,^\circ$, and interference signals at $\theta_1 = -20\,^\circ$, $\theta_2 = 20\,^\circ$. After simulating the implemented LMS algorithm 100 times, the ensuing results are illustrated in Figures 4.12 and 4.13.

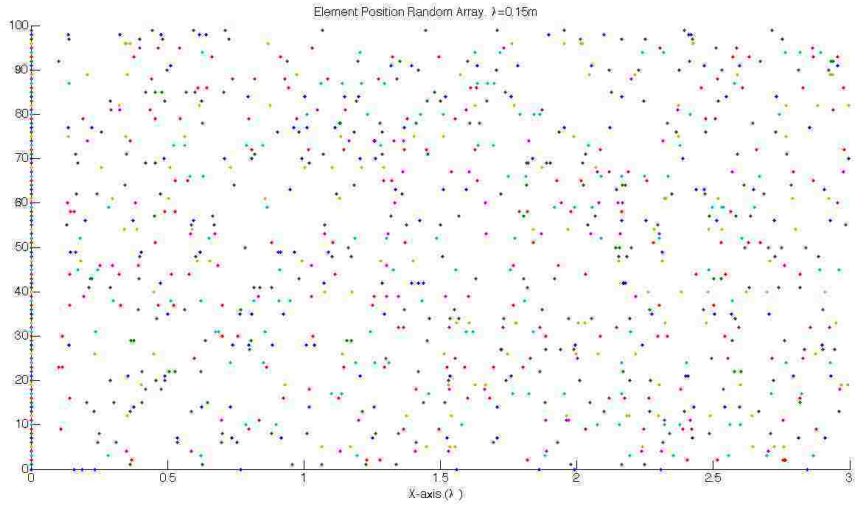


Figure 4.12: Antenna elements position for the 100 simulations

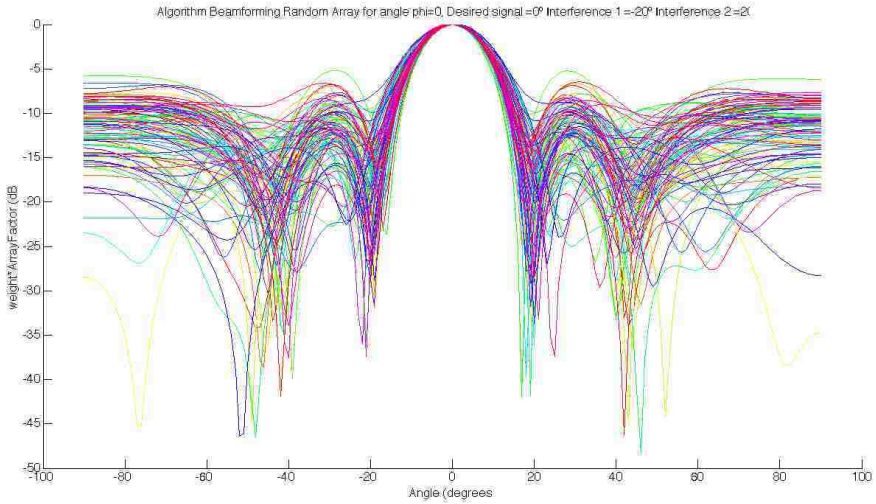


Figure 4.13: LMS Adaptive beamforming for the 100 simulations

The average and the average plus and minus the variance LMS adaptive beam-

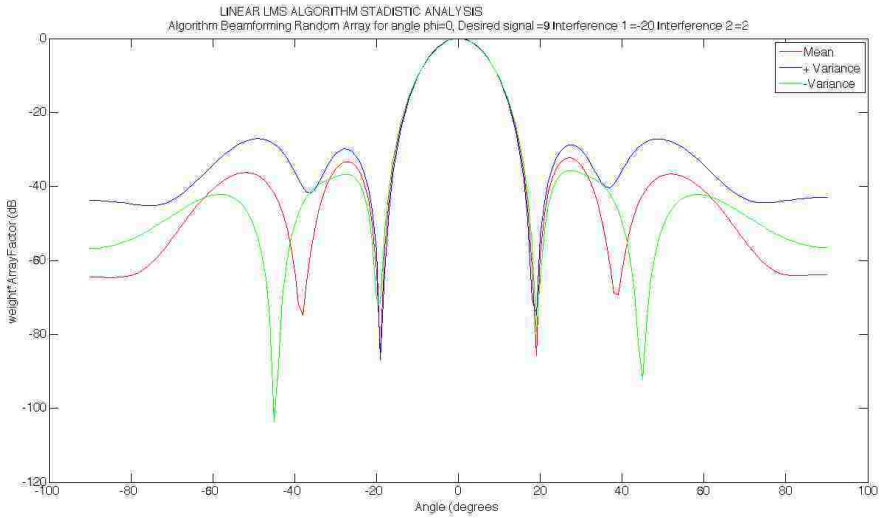


Figure 4.14: The statistical LMS adaptive beamforming for mean and variance

forming are shown in Figure 4.14. Table 4.5 shows the main and secondary minima positions obtained for each case.

Table 4.5: Minima values

(a) Mean minima

$\theta_0 = -19\,^\circ$	-84.61 dB
$\theta_1 = 19\,^\circ$	-85.87 dB
$\theta_2 = -38\,^\circ$	-74.83 dB
$\theta_2 = 39\,^\circ$	-69.29 dB

(b) Mean + variance minima

$\theta_0 = -19\,^\circ$	-86.83 dB
$\theta_1 = 19\,^\circ$	-74.92 dB
$\theta_2 = -36\,^\circ$	-41.74 dB
$\theta_2 = 37\,^\circ$	-40.29 dB

(c) Mean - variance minima

$\theta_0 = -19\,^\circ$	-71.69 dB
$\theta_1 = 19\,^\circ$	-79.45 dB
$\theta_2 = -45\,^\circ$	-103.6 dB
$\theta_2 = 45\,^\circ$	-92.11 dB

As expected, the obtained radiation patterns have a good performance for suppressing the interference signals at $\theta_1 = -20\,^\circ$ and $\theta_2 = 20\,^\circ$ despite the fact that the minima are not located at the exact position.

Leaky LMS Algorithm

The leaky LMS algorithm improves the LMS stability problems by introducing a leaky coefficient; thus, the performance of the leaky coefficient is analyzed in this section. Three different leaky coefficients were considered: $\gamma = 0.1$, $\gamma = 0.3$, and $\gamma = 1$.

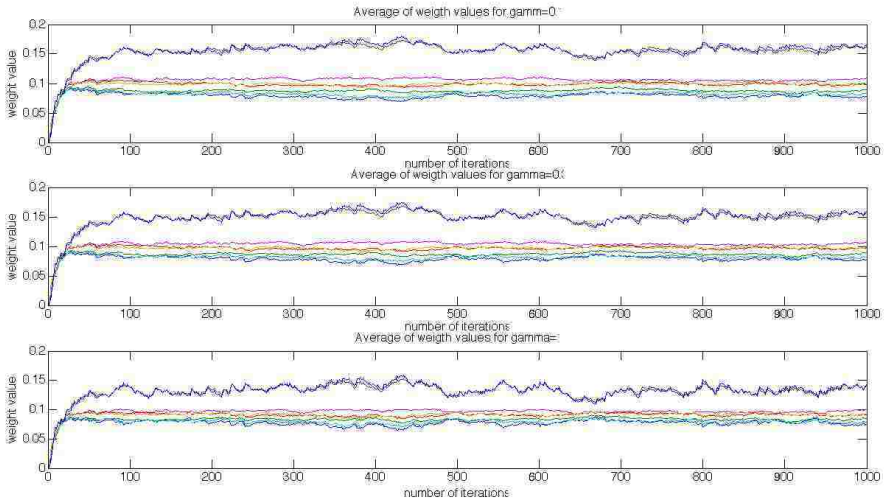


Figure 4.15: Weight vector for leaky LMS algorithm with $\gamma = 0.1$, $\gamma = 0.3$, and $\gamma = 1$.

When analyzing Figures 4.15, 4.16, and 4.17, one observes that for this case the leaky coefficient does not generate a significant variation in the optimum weight vector; and as a consequence, in the generated beamforming. One of the main reasons for which this might happens is because the eigenvalues of the system are not very small, which means that the system is already stable.

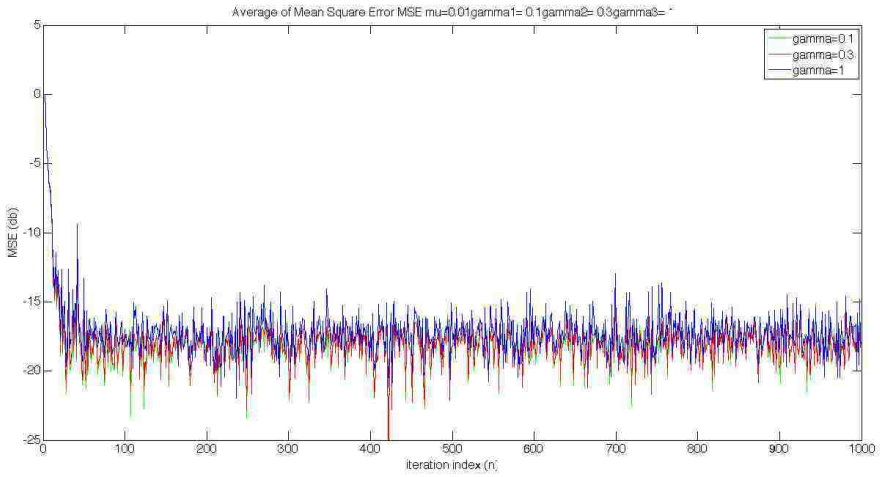


Figure 4.16: Average mean-square error for leaky LMS algorithm with $\mu_1 = 0.01$, $\mu_2 = 0.001$, and $\mu_3 = 0.0001$.

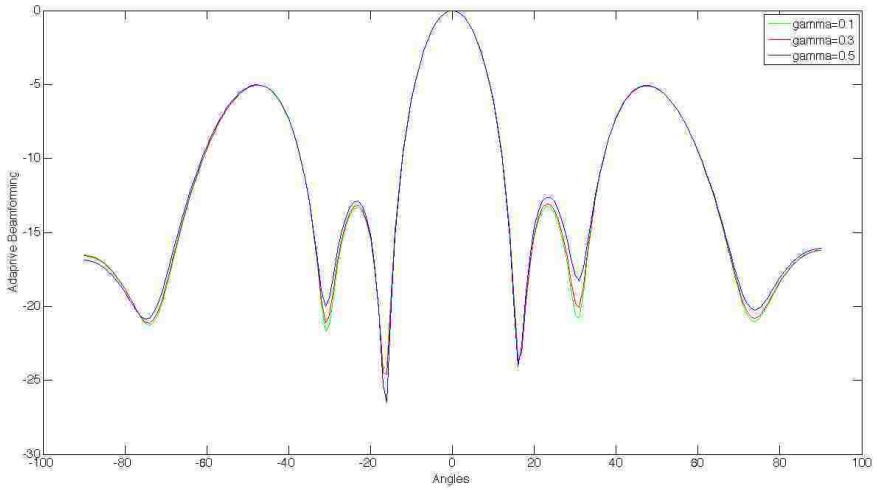


Figure 4.17: Adaptive beamforming leaky LMS algorithm for $\theta_0 = 0\,^\circ$, $\theta_1 = -30\,^\circ$, and $\theta_2 = 30\,^\circ$.

Normalized LMS Algorithm

As stated in Section 3.3.3, the normalized LMS algorithm actualizes the weight vector by normalizing in each iteration with respect to the squared Euclidean norm, $\|\mathbf{x}(k)\|^2$. As a consequence of this normalization, the step-size for the NLMS algorithm, μ, is not bounded by the input power, $0 < \mu < 2$. In this section, six different step-size are considered for analyzing the performance of the NLMS algorithm: $\mu_1 = 1.5$, $\mu_2 = 0.5$, $\mu_3 = 0.1$, $\mu_4 = 0.01$, $\mu_5 = 0.001$, and $\mu_6 = 0.0001$.

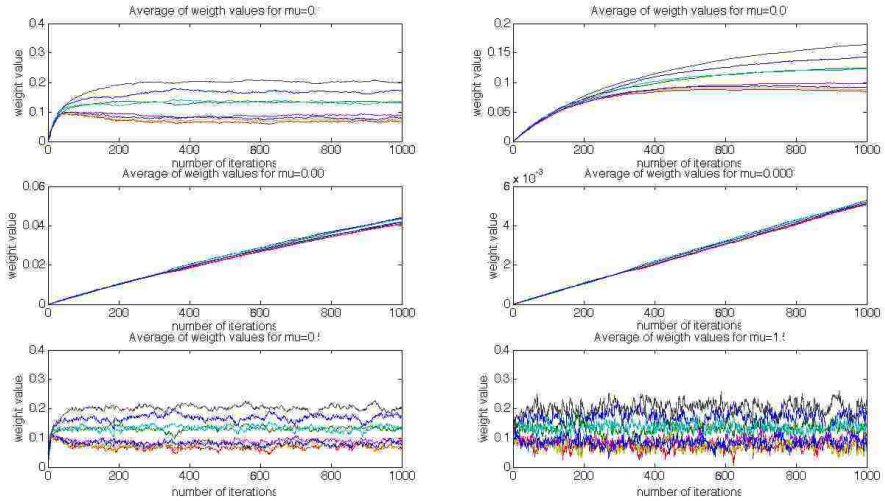


Figure 4.18: Optimum weight vector for $\mu_1 = 1.5$, $\mu_2 = 0.5$, $\mu_3 = 0.1$, $\mu_4 = 0.01$, $\mu_5 = 0.001$, and $\mu_6 = 0.0001$.

Figure 4.18 illustrates the performance of the NLMS algorithm for the different step-

sizes. One can observe that the bigger the step-size, the faster the converge to the optimum value. When $\mu_1 = 1.5$ is applied, the algorithm only needs 1 iteration in order to obtain the optimum weight vector; however when the step-size is less than 0.001, more than 1000 iterations are required.

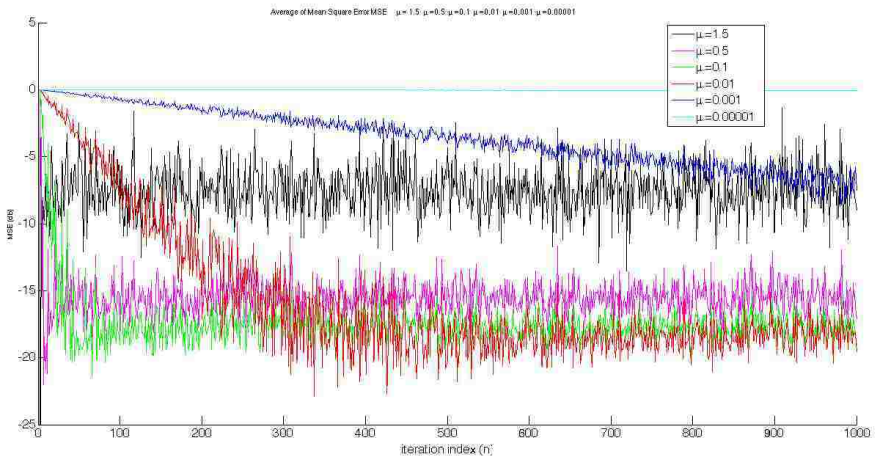


Figure 4.19: Mean square error for $\mu_1 = 1.5$, $\mu_2 = 0.5$, $\mu_3 = 0.1$, $\mu_4 = 0.01$, $\mu_5 = 0.001$, and $\mu_6 = 0.0001$.

When analyzing the mean square error, Figure 4.19, one observes that the mean-square error is stabilized when the optimum weight vector is obtained. Nevertheless, the obtained MSE for $\mu = 0.5$ is more stable than the obtained MSE for $\mu = 1.5$; it presents less fluctuations.

The generated array patterns are illustrated in Figure 4.20. Considering these patterns, better beamforming is obtained when $\mu = 1.5$.

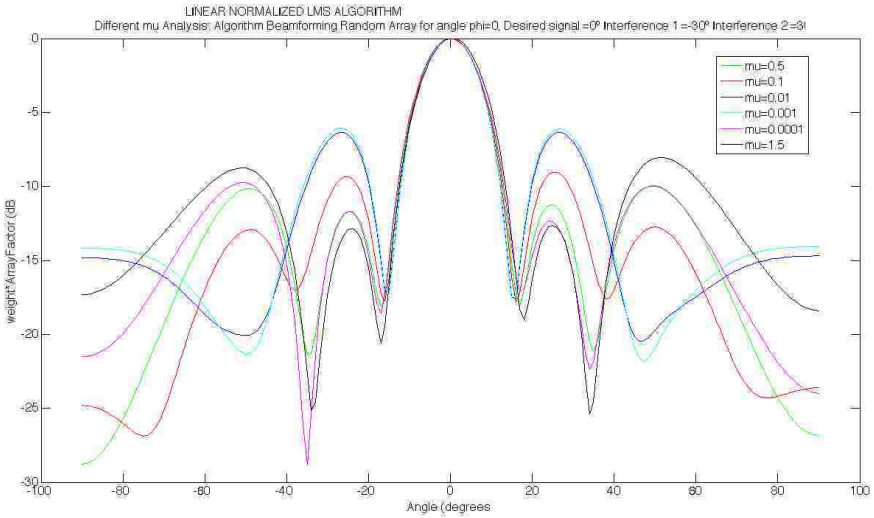


Figure 4.20: NLMS adaptive beamforming for $\mu_1 = 1.5$, $\mu_2 = 0.5$, $\mu_3 = 0.1$, $\mu_4 = 0.01$, $\mu_5 = 0.001$, and $\mu_6 = 0.0001$.

Generalized Normalized LMS Algorithm

The GNLMS algorithm introduces an adaptive step-size, whose value is updated at each iteration. In this way, the step-size varies according to dynamics of the input signal. In order to analyze the performance of the GNLMS algorithm, three different initial $\delta(k)$ were considered: $\delta_1(k) = 1$, $\delta_2(k) = 0.01$, and $\delta_3(k) = 0.001$

The obtained optimum weight vector, mean-square error and the adaptive beamforming are shown in Figures 4.21, 4.22, and 4.23 respectively. When analyzing these figures, one can notice that the different values of $\delta(k)$ do not result in evident changes. As with the leaky LMS algorithm, the cause of this lack of variation may be caused by the stability of the signal.

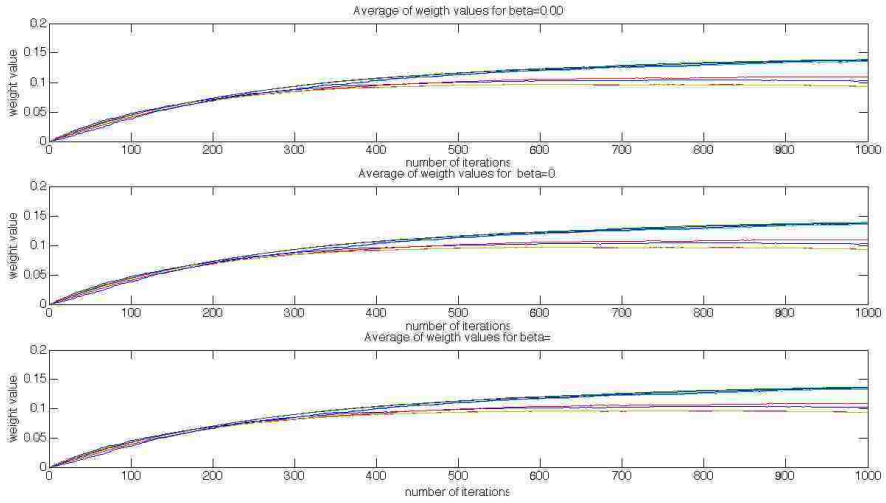


Figure 4.21: Optimum weight vector for $\delta_1(k) = 1$, $\delta_2(k) = 0.01$, and $\delta_3(k) = 0.001$.

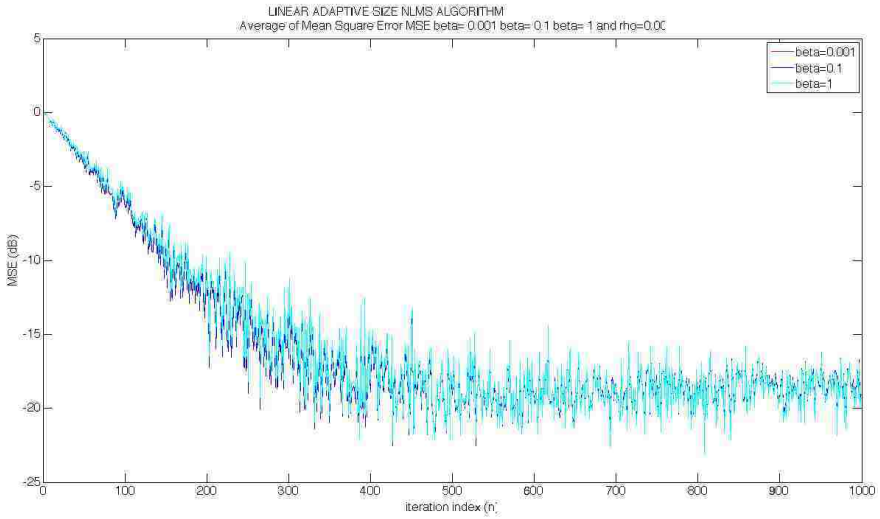


Figure 4.22: Mean square error for $\delta_1(k) = 1$, $\delta_2(k) = 0.01$, and $\delta_3(k) = 0.001$.

Constrained LMS Algorithm

The constrained LMS algorithm obtains the optimum weight vector by applying a constraint. According to the step-size, μ, the constrained LMS algorithm obtains

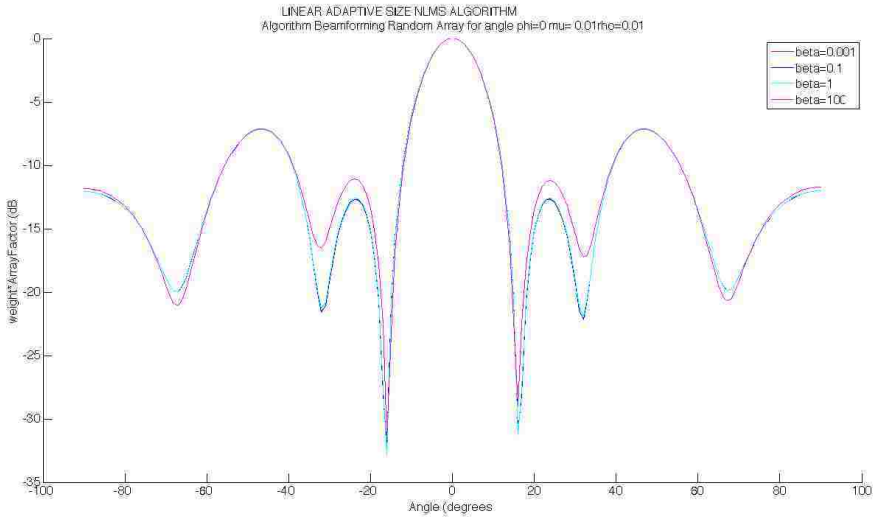


Figure 4.23: GNLMS adaptive beamforming for $\delta_1(k) = 1$, $\delta_2(k) = 0.01$, and $\delta_3(k) = 0.001$.

different weight vectors and as a consequence, different array patterns. In order to analyze the performance of Frost's algorithm, four different step-sizes were considered: $\mu_1 = 0.1$, $\mu_2 = 5$, $\mu_3 = 10$, and $\mu_4 = 5000$. These values were chosen taking into account the bound limits: $0 < \mu < \frac{2}{3\text{trace}(\mathbf{R}_{\mathbf{x}(k)\mathbf{x}(k)})}$.
The obtained optimum weight vector and beamforming are illustrated in Figures 4.24 and 4.25.

Frost's algorithm obtains the optimum weight vector at the first iteration. For this reason, Frost's algorithm has very precise results; however, the optimum weights vary according to the step-size. When the step-size is not choose correctly, the obtained beamforming does not satisfy the requirements. When analyzing Figure 4.25, one can notice that the array pattern obtained with $\mu = 5000$ does not satisfy the condition of having the desired signal at $\theta = 0\,^\circ$. The smaller the step-size, the better the array pattern.

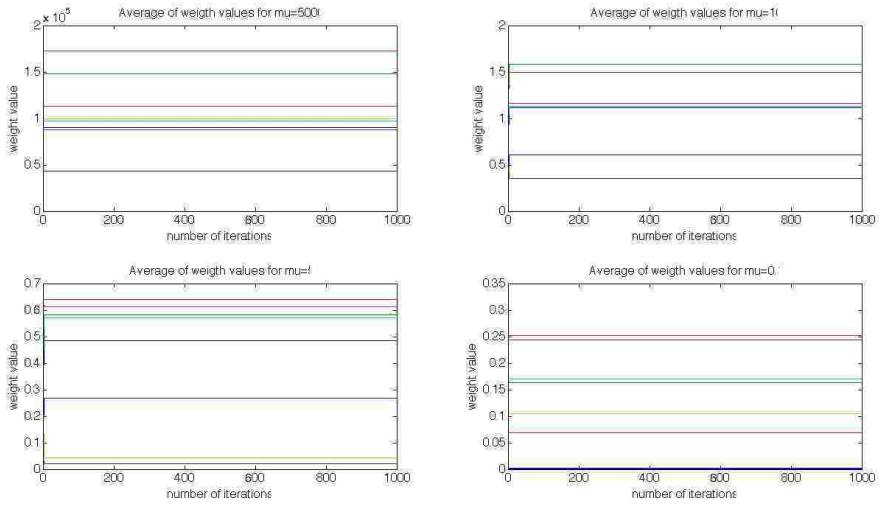


Figure 4.24: Optimum weight vector for $\mu_1 = 0.1$, $\mu_2 = 5$, $\mu_3 = 10$, and $\mu_4 = 5000$.

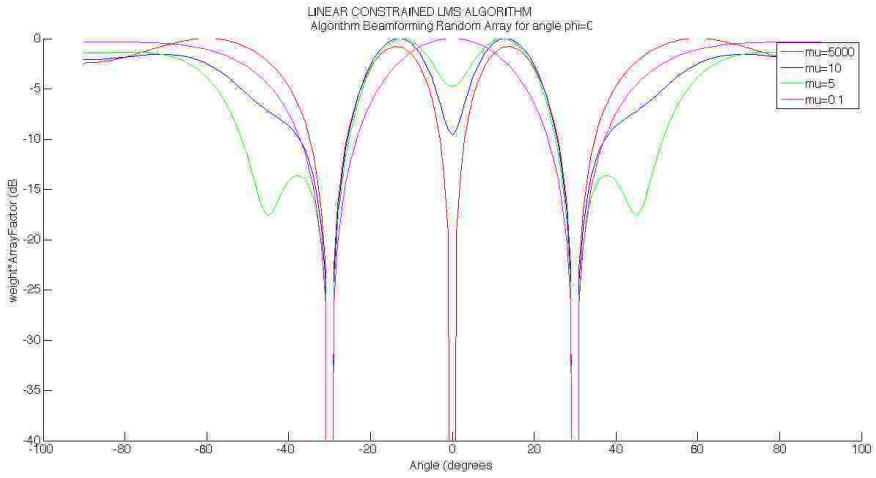


Figure 4.25: Constrained LMS adaptive beamforming for $\mu_1 = 0.1$, $\mu_2 = 5$, $\mu_3 = 10$, and $\mu_4 = 5000$.

RLS Algorithm

The RLS algorithm introduces a forgetting factor in order to improve the convergence rate and the stability of the system. In this case, three different forgetting factor were considered to evaluate the performance of the RLS algorithm: $\lambda_1 = 0.8$, $\lambda_2 = 0.9$, and $\lambda_3 = 0.99$.

When analyzing the optimum weight vector (Figure 4.26), one notices that for forgetting factors closer to zero, the optimum weight vector is obtained more quickly; however, the stability decreases, ergo this system is more unstable. On the other hand, when the forgetting factor is closer to one, $\lambda = 0.99$, the optimum weight vector is obtained after 300 iterations, making the system more stable and obtaining better beamforming 4.28.

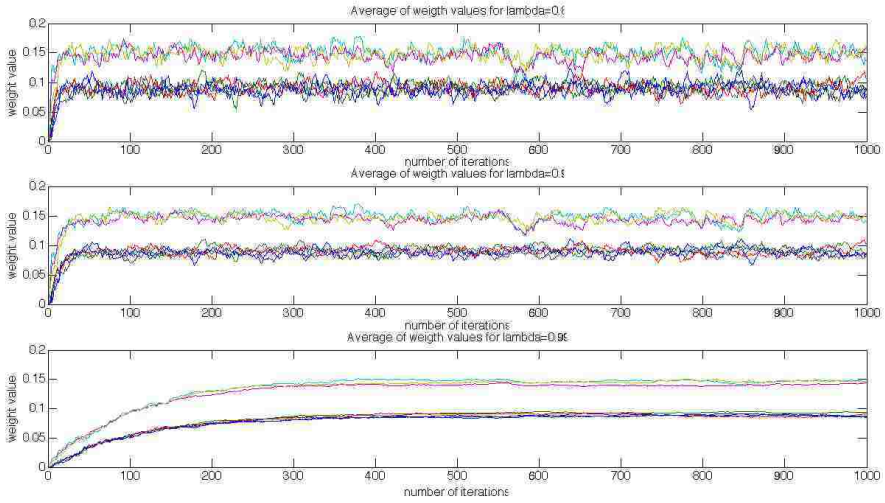


Figure 4.26: Optimum weight vector for $\lambda_1 = 0.8$, $\lambda_2 = 0.9$, and $\lambda_3 = 0.99$.

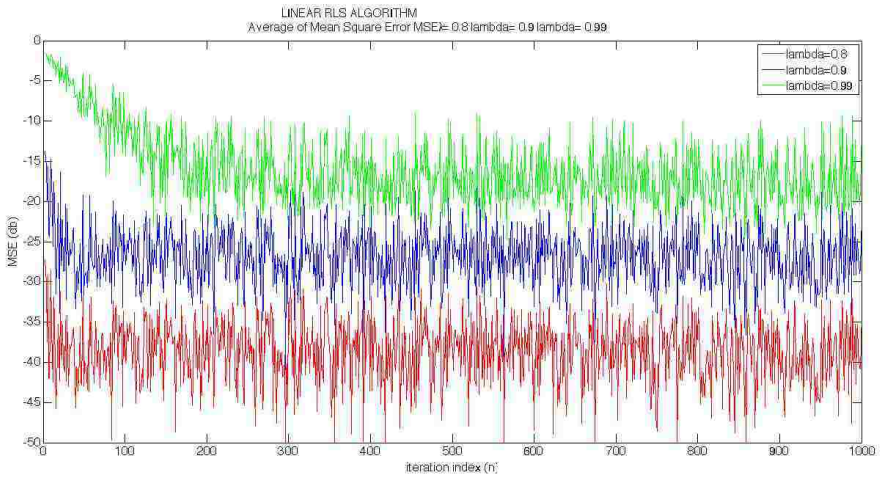


Figure 4.27: Mean square error for $\lambda_1 = 0.8$, $\lambda_2 = 0.9$, and $\lambda_3 = 0.99$.

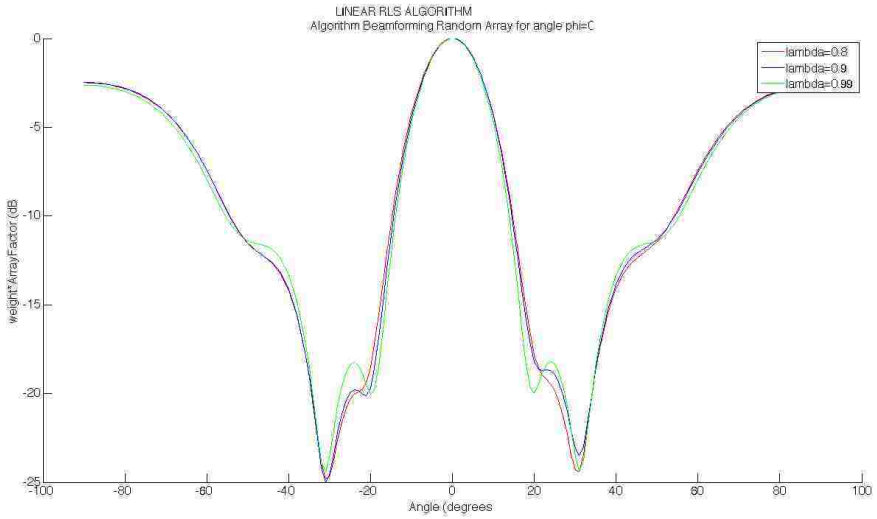


Figure 4.28: RLS adaptive beamforming for $\lambda_1 = 0.8$, $\lambda_2 = 0.9$, and $\lambda_3 = 0.99$.

Variable Forgetting Factor RLS Algorithm

The VFF-RLS algorithm is a modification of the RLS algorithm, which considers a time-variable forgetting factor, $\lambda(k)$, that varies at each iteration. Three different initial forgetting factor were considered to evaluate and analyze the VFF-RLS algorithm: $\lambda_1(1) = 0.8$, $\lambda_2(1) = 0.9$, and $\lambda_3(1) = 0.99$.

As with the RLS algorithm case, the VFF-RLS obtains a more stable system as the step-size gets closer to the unity. Nevertheless, the VFF-RLS algorithm has improved the convergence rate for all of the forgetting factors: in the case of $\lambda_1 = 0.8$, the RLS algorithm requires approximately 300 iterations in order to obtain the optimum weight vector, while the VFF-RLS requires less than 50.

When analyzing the MSE and the adaptive beamforming in Figures 4.30 and 4.31 respectively, one can observe that higher forgetting factors obtain higher MSEs and a more optimal array pattern. However, if the forgetting factor is close to one, the system is not able to track the input signal correctly as with lower forgetting factors.

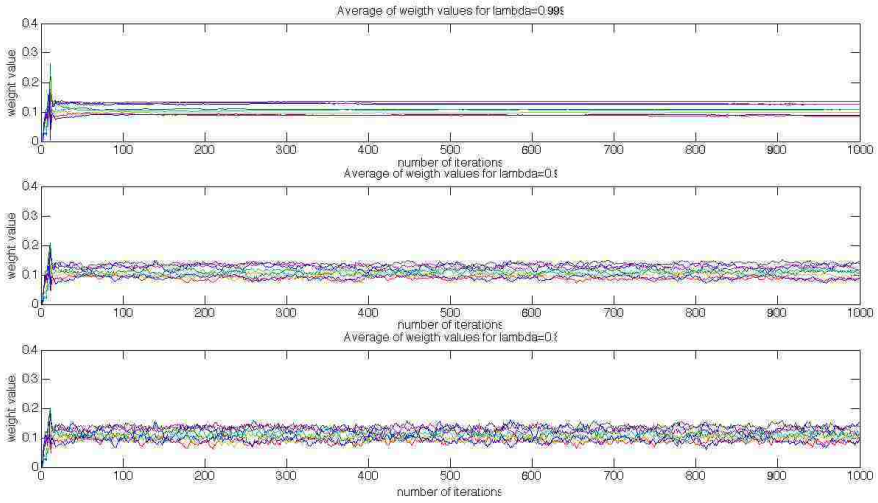


Figure 4.29: Optimum weight vector for $\lambda_1(1) = 0.8$, $\lambda_2(1) = 0.9$, and $\lambda_3(1) = 0.999$.

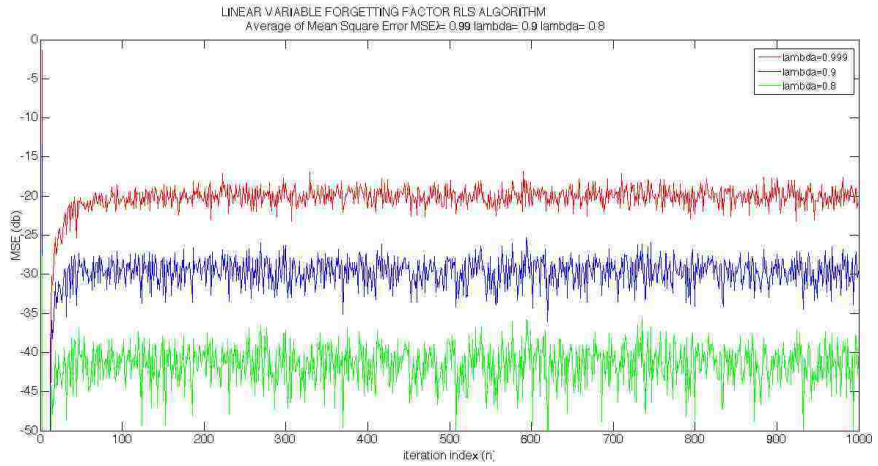


Figure 4.30: Mean-square error for $\lambda_1(1) = 0.8$, $\lambda_2(1) = 0.9$, and $\lambda_3(1) = 0.999$.

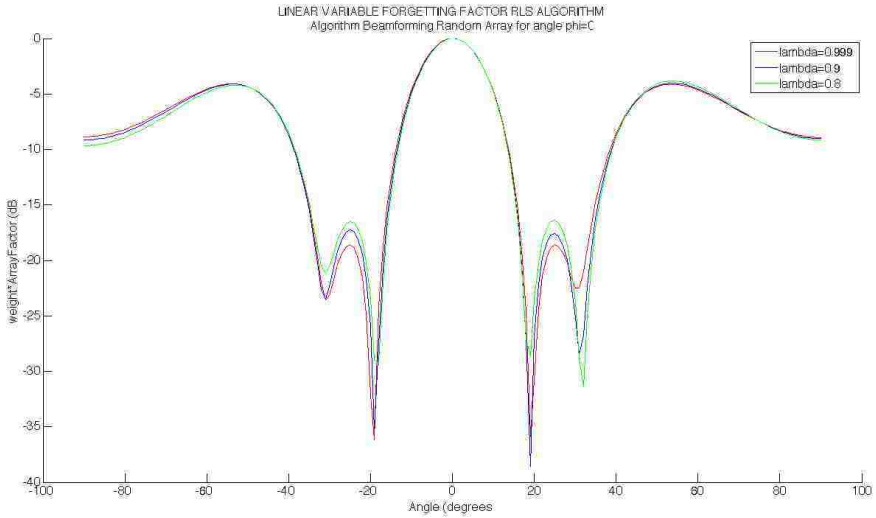


Figure 4.31: VFF-RLS adaptive beamforming for $\lambda_1(1) = 0.8$, $\lambda_2(1) = 0.9$, and $\lambda_3(1) = 0.999$.

4.2 Algorithm Robustness Analysis

4.2.1 Rectangular Array vs Random Planar Array

This section will evaluate and analyze the difference between URAs (Uniformly spaced array) and RPAs. For the URA system, the elements are uniformly spaced; therefore, the distance between elements must be less than $\frac{\lambda}{2}$, $d < \frac{\lambda}{2}$, in order to satisfy the Nyquist criterion to avoid aliasing. Nevertheless, the RPA elements do not need to satisfy this criterion in order to be able to have separation between them.

The following section, 4.2.2, will provide a complete analysis of this random separation and the number of elements between this distance is studied. Furthermore, it will compare the URA and RPA systems, exploring the possibility of obtaining a desired or similar array pattern with the RPA system by using less randomly spaced elements.

For the uniform case, the signal model is composed of 16-isotropic elements with a constant distance, $\frac{\lambda}{2}$, between them and 3 narrowband signals, one desired and two interference, arriving from $(\theta_0 = 0\,^\circ, \phi_0 = 0\,^\circ)$, $(\theta_1 = -30\,^\circ, \phi_1 = 0\,^\circ)$ and $(\theta_2 = 30\,^\circ, \phi_2 = 0\,^\circ)$ at a 2-GHz frequency. The random signal model is applied to the same signal model, but the number of elements is variable.

CASE 1: URA and RPA with 16 elements.

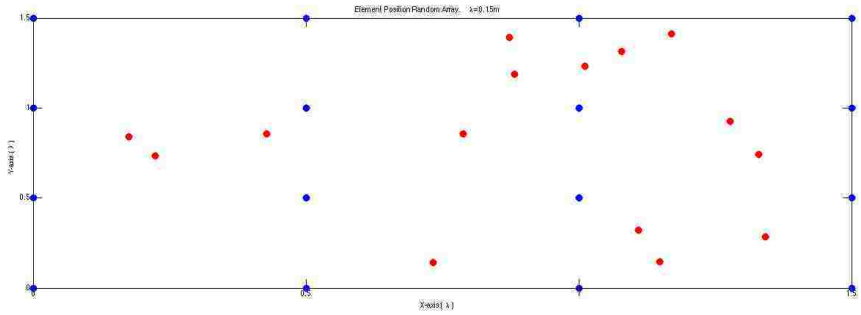


Figure 4.32: Element position for URA vs RPA.

In the first case, Figures 4.32, 4.33, and 4.34 are obtained. The obtained array pattern for the $\phi = 0\,^\circ$ plane satisfies the requirements of suppressing the interference signals at $(\theta_1 = -30\,^\circ, \phi_1 = 0\,^\circ)$ and $(\theta_2 = 30\,^\circ, \phi_2 = 0\,^\circ)$ by having -25 dB and -21 dB respectively. For the $\phi = 90\,^\circ$ plane, the randomly spaced antenna array has a less restrictive pattern emitting / receiving with a $60\,^\circ$ main beam.

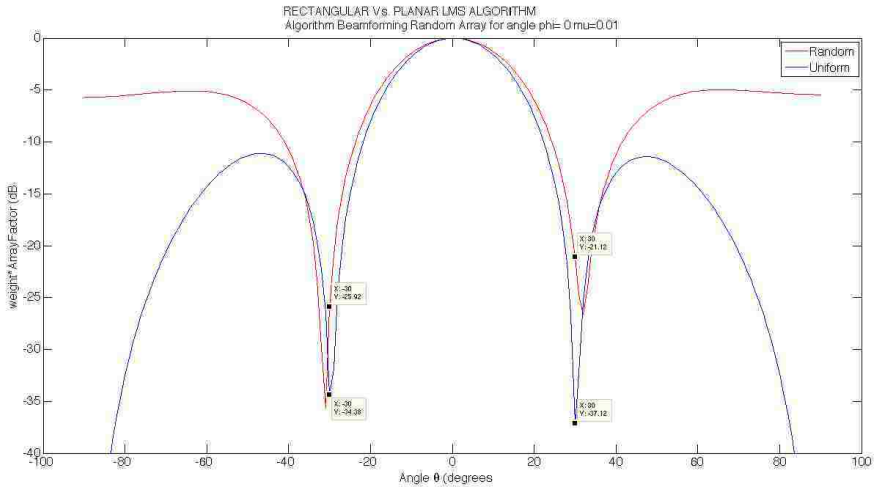


Figure 4.33: Adaptive beamforming $\phi = 0\,^\circ$ plane for URA and RPA.

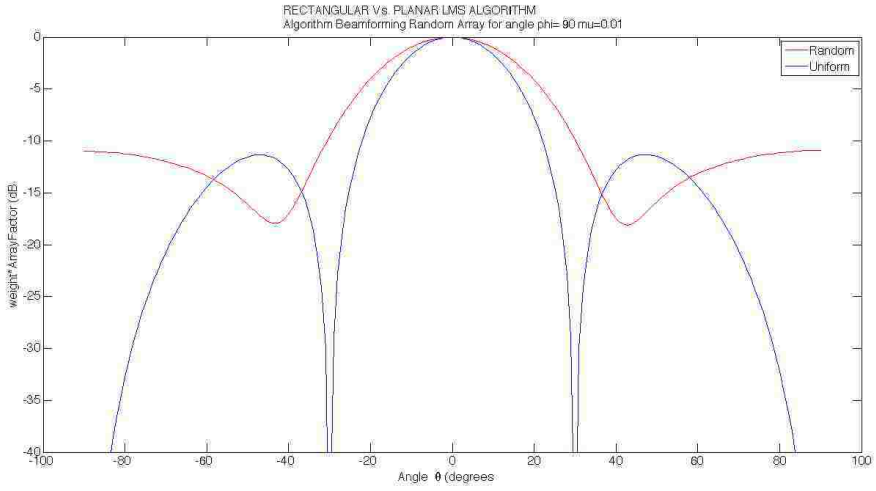


Figure 4.34: Adaptive beamforming $\phi = 90\,^\circ$ plane for URA and RPA.

CASE 2: URA with 16 elements and RPA with 12.

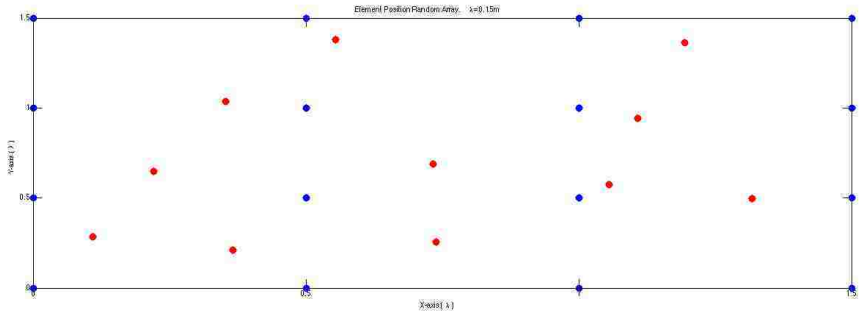


Figure 4.35: Elements position for URA vs RPA.

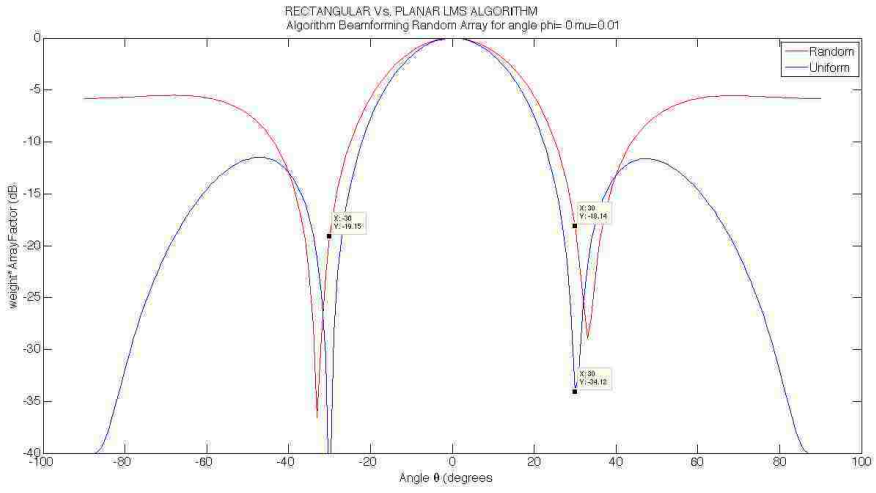


Figure 4.36: Adaptive beamforming $\phi = 0\,^\circ$ plane for URA and RPA.

When analyzing Figures 4.36 and 4.37, one can observe that the obtained beamforming for $\phi = 0\,^\circ$ and $\phi = 90\,^\circ$ demonstrates good performance. The interference

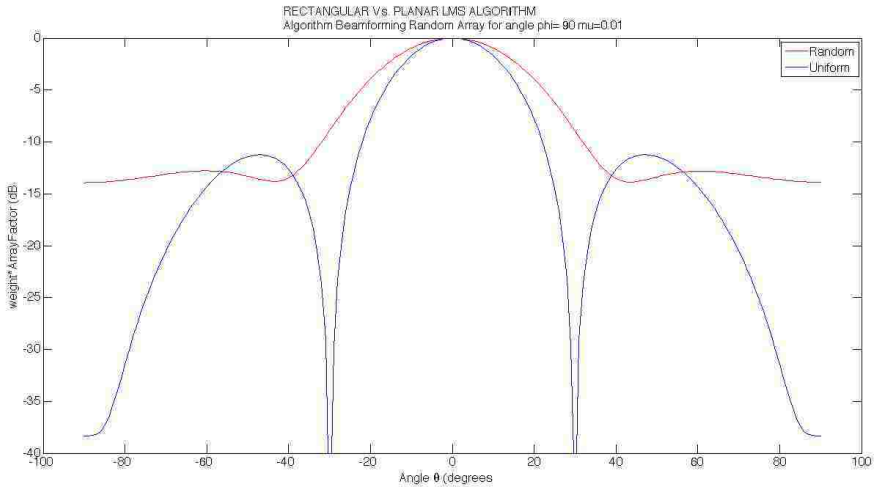


Figure 4.37: Adaptive beamforming $\phi = 90\,^\circ$ plane for URA and RPA.

signals are suppressed by -19 dB and -18 dB.

CASE 3: URA with 16 elements and RPA with 8.

Considering the figures obtained for this case, where the randomly spaced array is composed of half of the URA elements, 8, the plane $\phi = 0\,^\circ$, (Figure 4.39), demonstrates adequate performance by suppressing the interference signals with -16 dB and -19 dB respectively. In this case, the $\phi = 90\,^\circ$ plane transmits / receives along the entire band, Figure 4.40.

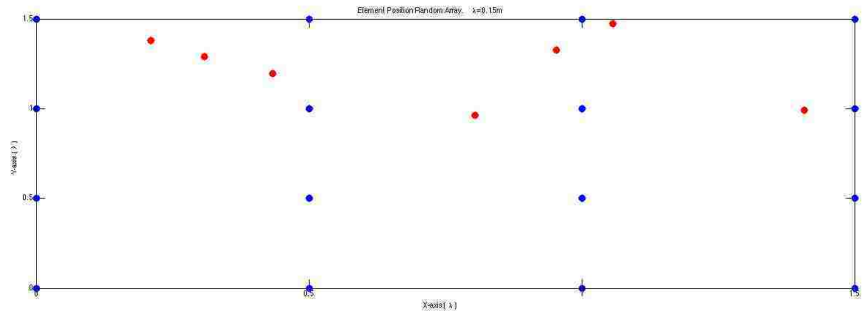


Figure 4.38: Elements position for URA vs RPA.

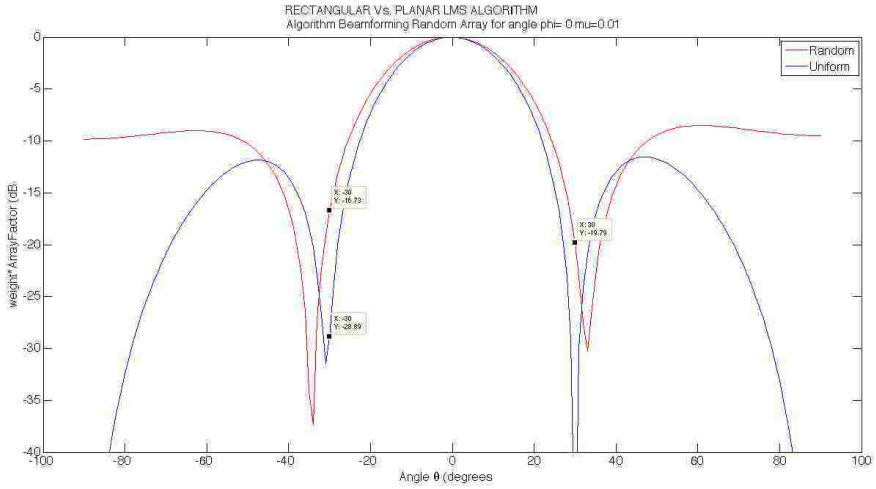


Figure 4.39: Adaptive beamforming $\phi = 0\,^\circ$ plane for URA and RPA.

CASE 4: URA with 16 elements and RPA with 4.

In the last case, the RPA system is composed of one quarter of the URA elements, 4. When analyzing Figure 4.42 for this particular case, both the rectangular and planar array obtain very similar beamforming: -31.7 and -31.96 dB for the interference signal at $\theta = -30\,^\circ$ and -26,49 and -33.69 dB for $\theta = 30\,^\circ$. Nevertheless, the obtained beamforming for $\phi = 90\,^\circ$ plane in the RPA is less restrictive than the rectangular

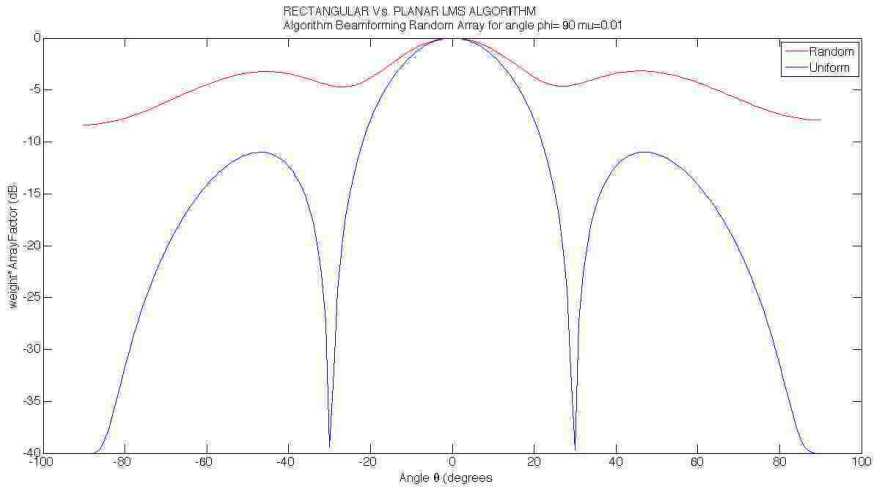


Figure 4.40: Adaptive beamforming $\phi = 90\,^\circ$ plane for URA and RPA.

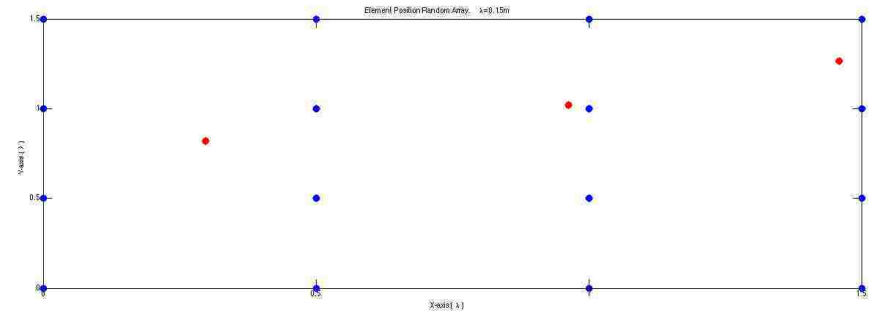


Figure 4.41: Elements position for URA vs RPA.

beamforming.

This section conducts a comparison between URA and RPA systems both for the same number of elements and for different number of elements illustrated. One can observe that the obtained array patterns for the case of fewer elements satisfies the condition of maximizing the beamforming towards the desired signal while at the same time suppressing towards the interference signals. However, the obtained

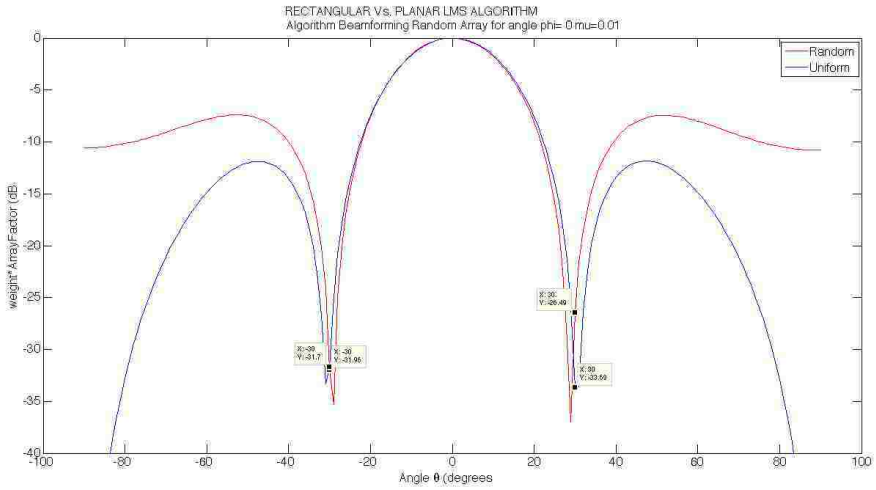


Figure 4.42: Adaptive beamforming $\phi = 0\,^\circ$ plane for URA and RPA.

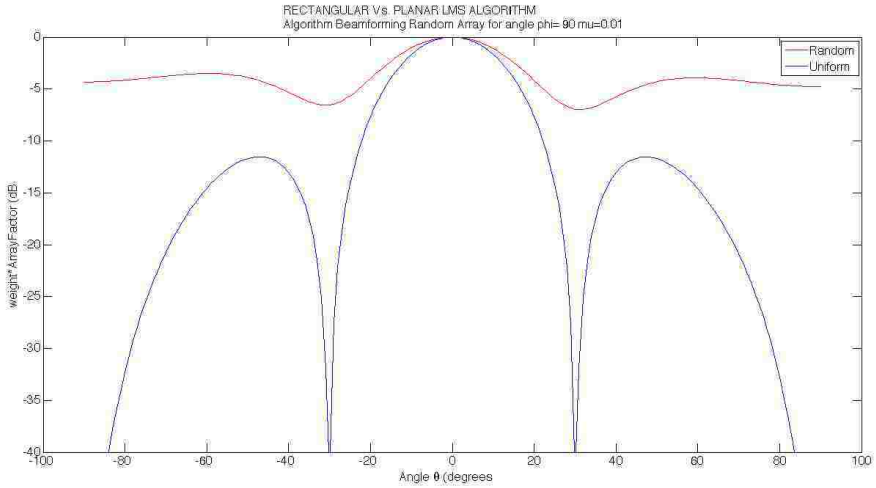


Figure 4.43: Adaptive beamforming $\phi = 90\,^\circ$ plane for URA and RPA.

beamforming $\phi = 90\,^\circ$ does not present any restriction as URA does. As a consequence, depending on the application and the importance of the restriction towards $\phi = 90\,^\circ$ plane, one can consider using less antenna elements, thus making an RPA more economic and more simple to implement.

4.2.2 The Necessary Elements

Considering a fixed length for the dimensions of an RLA or RPA, one can logically assume that as the number of elements within this length increases, the beamforming performance improves. This section considers two different scenarios in both environments (linear and rectangular)in order to demonstrated this assumption.

SCENARIO 1:

In the first case, a 10-λ RLA system was considered. The RLS algorithm was applied for three different element numbers: $N_1 = 8$, $N_2 = 20$, and $N_3 = 50$. The system is composed of 3-narrowband signals arriving at $\theta_0 = -20\,^\circ$, $\theta_1 = 15\,^\circ$, and $\theta_2 = 56\,^\circ$. The obtained beamforming and the element positions are illustrated in Figures 4.50, 4.51, and 4.52.

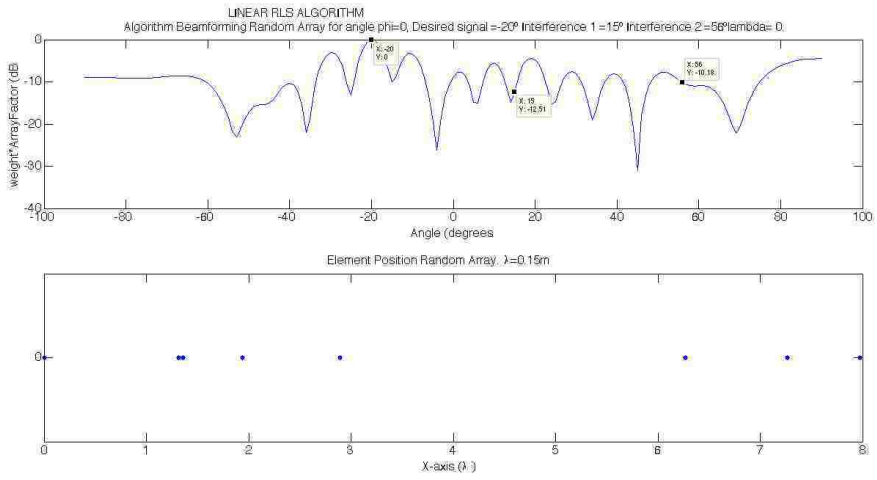


Figure 4.44: RLS algorithm for N=8 elements.

As expected, the beam pattern's performance improves as the number of elements increases, as shown in Table 4.6. The three different cases satisfy the requirements for the DOAs; however, from N_1 to N_2 the performance improves significantly.

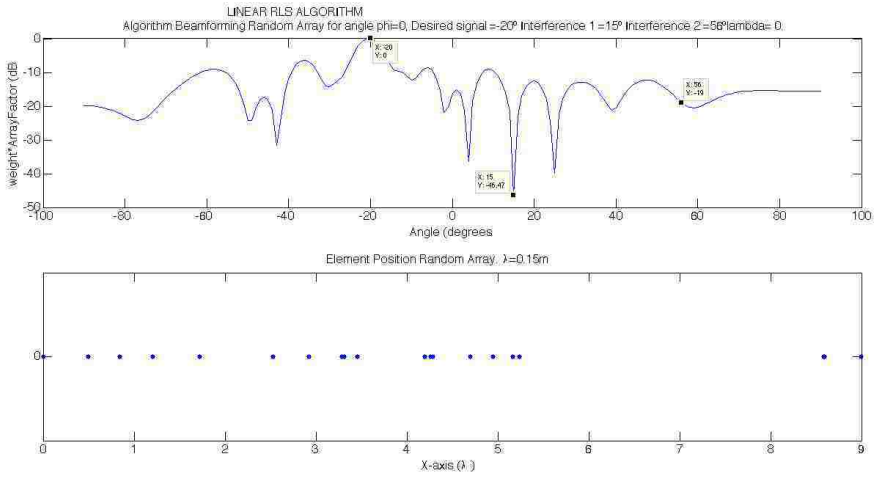


Figure 4.45: RLS algorithm for N=20 elements.

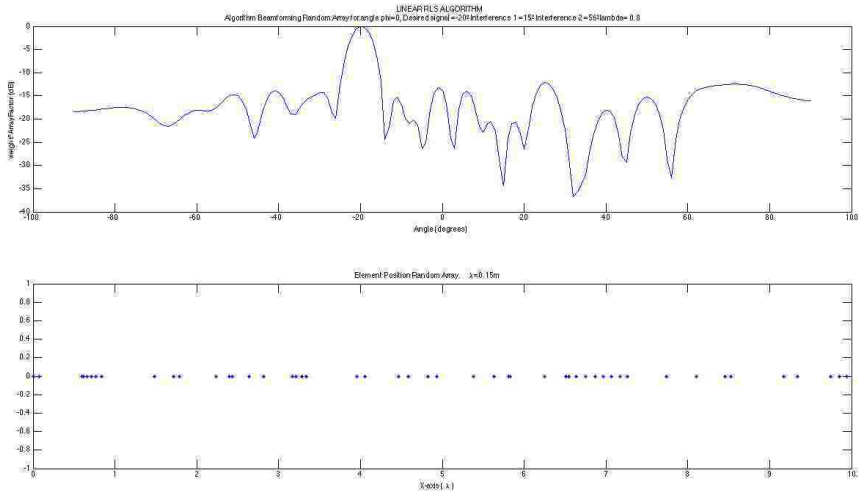


Figure 4.46: RLS algorithm for N=50 elements.

SCENARIO 2:

For the second scenario, a 10-λ RPA system was considered. The LMS algorithm

Table 4.6: Obtained Array Pattern Value 1st Scenario (dB):

	$N_1 = 8$	$N_2 = 20$	$N_3 = 50$
$\theta_0 = -20\,^\circ$	0	0	0
$\theta_1 = 15\,^\circ$	-12.51	-46.47	-34.39
$\theta_2 = 56\,^\circ$	-10.18	-19	-32.6

was applied to three different element numbers: $N_1 = 8$, $N_2 = 20$, and $N_3 = 50$. The system is composed of 3-narrowband signals arriving at ($\theta_0 = 0\,^\circ$,$\phi_0 = 0\,^\circ$), ($\theta_1 = 15\,^\circ$,$\phi_1 = 0\,^\circ$), and ($\theta_2 = 56\,^\circ$,$\phi_2 = 0\,^\circ$). The obtained beamforming and the element positions are illustrated in Figures 4.50, 4.51, and 4.52.

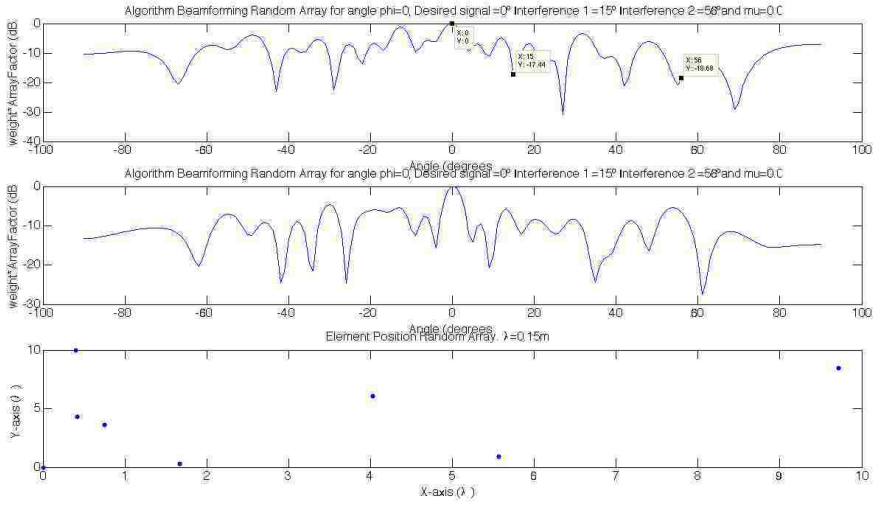


Figure 4.47: LMS algorithm for N=8 elements.

When analyzing Table 4.8, one observes that the beamforming obtained satisfies the system requirements by suppressing the interference signals while at the same time enhancing the desired signal.

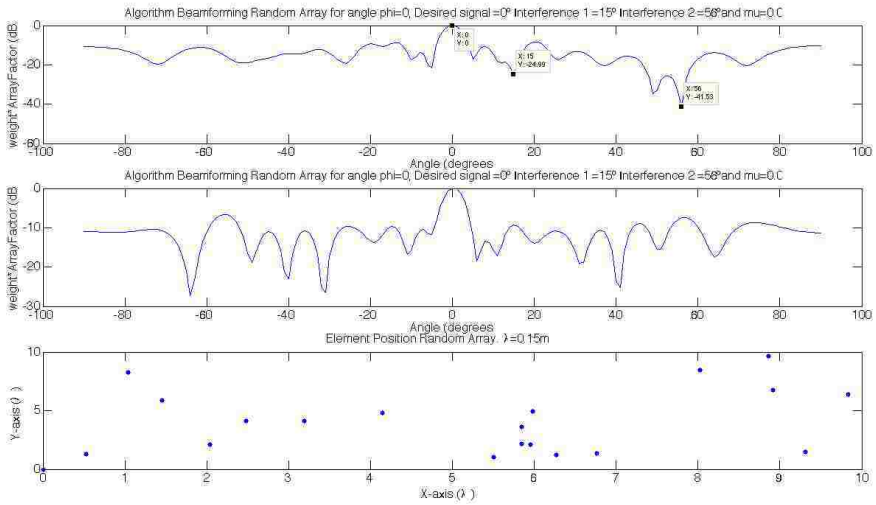


Figure 4.48: LMS algorithm for N=20 elements.

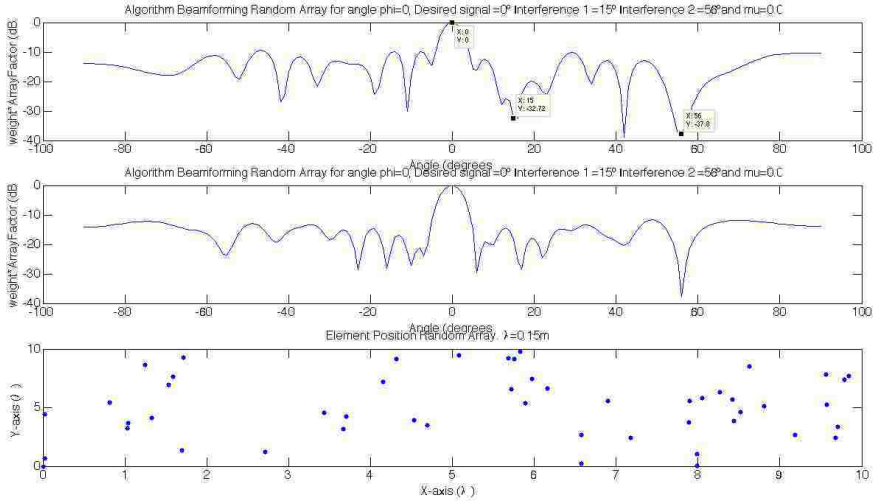


Figure 4.49: LMS algorithm for N=50 elements.

These two scenarios, prove that increasing the number of elements for a fixed length, improves the overall performance of the system. This demonstration was concluded

Table 4.7: Obtained Array Pattern Value 2nd Scenario (dB):

	$N_1 = 8$	$N_2 = 20$	$N_3 = 50$
$\theta_0 = 0\,^\circ$	0	0	0
$\theta_1 = 15\,^\circ$	-17.44	-24.99	-32.72
$\theta_2 = 56\,^\circ$	-18.68	-41.53	-37.8

with an RLA and an RPA environment by applying RLS and LMS algorithms.

4.2.3 Analysis for the Number of Nulls

As Balanis states in his book, *Introduction to Smart Antennas*: "In theory, with M antenna elements $M-1$ sources of interference can be nulled out, but this number will normally be lower due to the multipath propagation environment" [1]. Knowing that mathematically that statement is true due to the fact that the number of equations is reduced to the number of antennas, in a system composed by M antenna elements, 1 desired signal and an M-1 interference can be considered, producing M equations for M unknowns.

This section takes two different cases into consideration for evaluating and analyzing this condition for randomly spaced antenna arrays.

CASE 1:

In the first case, a 10-element RLA composed of 11 narrowband signals, two desired and 9 interference is considered. Desired signals arrive at $\theta_0 = -45\,^\circ$ and $\theta_{01} = 45\,^\circ$. The interference signals arrive at $\theta_1 = -75\,^\circ$ $\theta_2 = -60\,^\circ$, $\theta_3 = -30\,^\circ$, $\theta_4 = -20\,^\circ$, $\theta_5 = 0\,^\circ$, $\theta_6 = 20\,^\circ$, $\theta_7 = 30\,^\circ$, $\theta_8 = 60\,^\circ$, and $\theta_9 = 75\,^\circ$.

One can observe by applying the constrained LMS algorithm, that the obtained array pattern satisfies the system requirements regarding interference signals; however, half of the power is lost for the desired signals. Depending on the application and

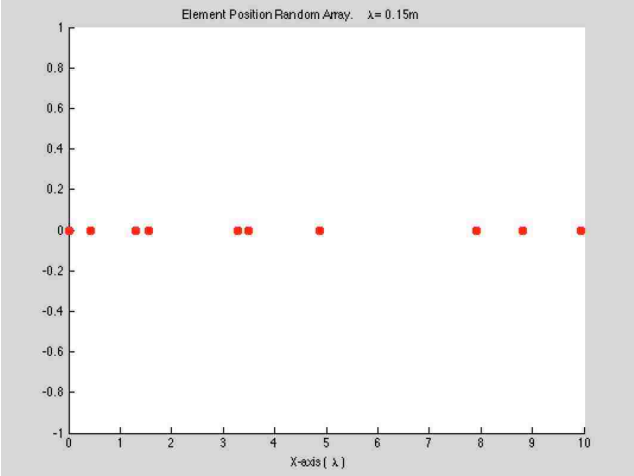


Figure 4.50: Element linear distribution N=10.

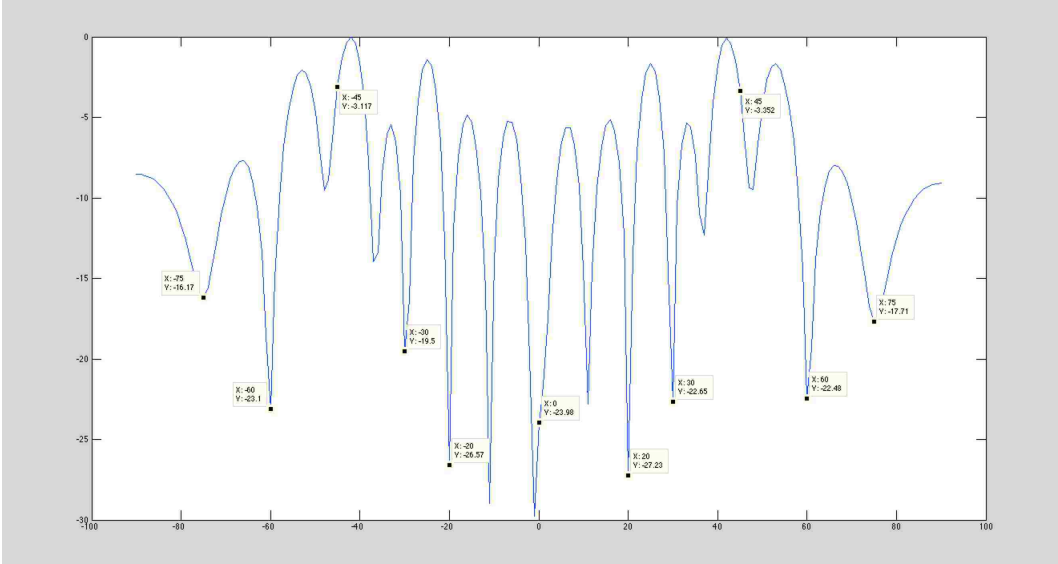


Figure 4.51: Constrained LMS algorithm for N=10 elements.

the situation, it may be preferable to lose this power towards the desired signals.

Table 4.8: Obtained Array Pattern Values 8-element (dB):

$\theta_0 = -45\,^\circ$	-3.11
$\theta_{01} = 45\,^\circ$	-3.34
$\theta_1 = -75\,^\circ$	-16.17
$\theta_2 = -60\,^\circ$	-23.1
$\theta_3 = -30\,^\circ$	-19
$\theta_4 = -20\,^\circ$	-26.57
$\theta_5 = 0\,^\circ$	-23.98
$\theta_6 = 20\,^\circ$	-27.23
$\theta_7 = 30\,^\circ$	-22.65
$\theta_8 = 60\,^\circ$	-22.48
$\theta_9 = 75\,^\circ$	-17.71

CASE 2:

In the second case, the system is composed of 10 isotropic elements randomly distributed along the x-axis. 11 narrowband signals arrive at the system, 1 desired and 10 interference. $\theta_0 = 0\,^\circ$, $\theta_1 = -68\,^\circ$ $\theta_2 = -60\,^\circ$, $\theta_3 = -50\,^\circ$, $\theta_4 = -45\,^\circ$, $\theta_5 = -10\,^\circ$, $\theta_6 = 25\,^\circ$, $\theta_7 = 45\,^\circ$, $\theta_8 = 50\,^\circ$, $\theta_9 = 68\,^\circ$, and $\theta_{10} = 72\,^\circ$.

In this second case, the obtained results are shown in Figure 4.53 and Table 4.9. The system composed of 10-elements is able to detect 11 signals, suppressing the 10-interference signals and generating a maximum towards the desired signal.

One can observe that the randomly spaced antenna array is able to create a desired beam pattern when the number of elements is less than the number of incoming DOAs. However, the obtained beamforming may not be optimal. Depending on the applications, one may consider it more important to detect more interference signals,

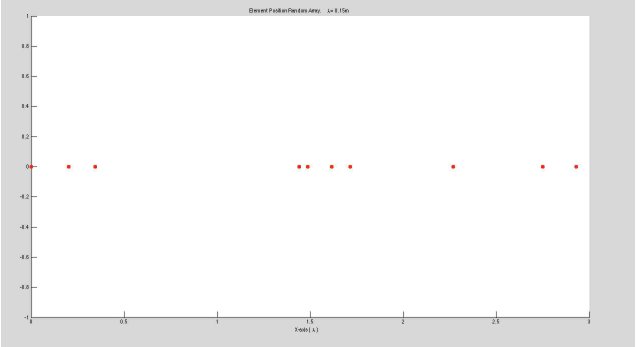


Figure 4.52: Element linear distribution N=10.

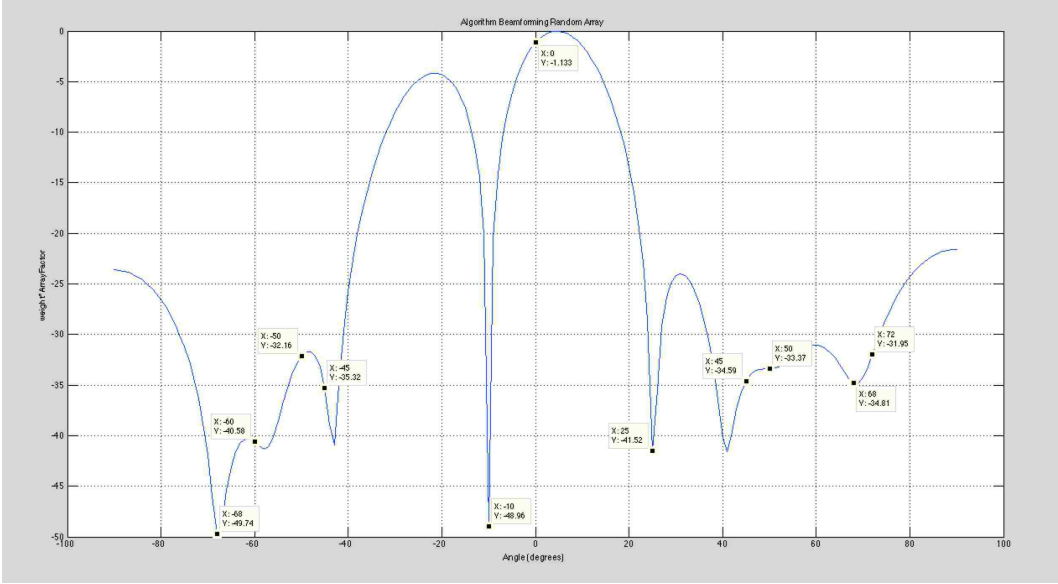


Figure 4.53: Constrained LMS algorithm for N=10 elements.

thus reducing the optimal beamforming, or to obtain a more precise pattern, thus decreasing the number of detectable signals. Therefore, this is a compromise between the number of signals detected and the obtained beamforming.

Table 4.9: Obtained array pattern values 8-element (dB):

$\theta_0 = 0\,^\circ$	-1.33
$\theta_1 = -68\,^\circ$	-49.74
$\theta_2 = -60\,^\circ$	-40.58
$\theta_3 = -50\,^\circ$	-32.16
$\theta_4 = -45\,^\circ$	-35.32
$\theta_5 = -10\,^\circ$	-48.96
$\theta_6 = 75\,^\circ$	-17.71
$\theta_7 = 45\,^\circ$	-34.59
$\theta_8 = 50\,^\circ$	-33.37
$\theta_9 = 68\,^\circ$	-34.81
$\theta_{10} = 72\,^\circ$	-31.95

Chapter 5

Conclusions and Future Work

This study presented an analysis of randomly spaced smart antennas. First, randomly spaced linear and planar signal models were studied. After understanding the nonuniform model, several DOAs and adaptive beamforming algorithms were theoretically analyzed; such as the Capon and MUSIC algorithms for direction of arrival, and the LMS, normalized LMS, leaky LMS, generalized normalized LMS, RLS and variable forgetting factor RLS algorithms for the adaptive beamforming.

These algorithms were implemented by using MATLAB®. The first part of the experiments analyzes the DOA algorithms (Capon and MUSIC) by considering two different scenarios. Both algorithms yield a good results for detecting DOAs; however, the MUSIC algorithm yields more accurately. Within this section, the performance of adaptive beamforming algorithms also illustrated, while varying the step size and the forgetting factor for LMS algorithms and RLS algorithms, respectively. It was shown that the performance of the adaptive beamforming algorithms is directly proportional to the step-size and forgetting factor. Also, the variability of the elements due to the randomness was studied for the LMS algorithm. A complete

statistical analysis for all the algorithms was not implemented because of computer limitations; however, this analysis is proposed as a possible extension for future work. The second section focus on the algorithm robustness. These cases included: a comparison between linearly spaced rectangular and randomly spaced antenna arrays, a demonstration to show that the improvement of radiation patter is directly proportional to the number of elements that compose it, and an analysis of the possible number of nulls in a radiation pattern. As a summary of these cases, one observed that a trade off between the number of elements and radiation pattern needs to be considered. A system with fewer antenna elements may satisfy the necessary requirements for suppressing and enhancing the interference and desired signals; however, the obtained beamforming is not necessary optimized.

Future work includes new implementations of these beamforming and DOA approaches based on machine learning theory. In particular, it is interesting to change the well known Least Squares (LS) approach used in all the algorithms presented in this work by other criteria. Indeed, LS is maximum Likelyhood only in the presence of Gaussian statistics, but it will lead to suboptimal solutions when the statistics is nongaussian. Also, in communications and related areas, only small data samples are often available for training the systems, which can lead to numerically ill-posed problems and, in general, produce over fitted solutions, a situation that produces solutions biased from the optimal ones.

Alternative solutions include regularized functionals with non quadratic objective functions, that can be closer to optimal in non Gaussian environments. A prominent example of such solutions are SVMs, whose optimization strategy is based on a linear cost function (actually a function similar to the Huber cost function [21]) plus a quadratic regularization. This and other solutions also regularized can include non-

linear processing that, in many situations, enhance the performance of algorithms. This is the case of kernel [22] and [23] versions of linear methods.

References

[1] C. A. Balanis and P. Ioannides, *Introduction to Smart Antennas*. Morgan & Claypool Publishers, 2007.

[2] C. A. Balanis, *Antenna Theory: Analysis and Design, 3rd Edition*. Wiley-Interscience, 2005.

[3] S. Haykin, *Adaptive Filter Theory (3rd ed.)*. Upper Saddle River, NJ, USA: Prentice Hall, 1996.

[4] H. Van Trees, *Detection, Estimation, and Modulation Theory*. New York: John Wiley & Sons, 2002.

[5] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *Signal Processing Magazine, IEEE*, vol. 13, no. 4, pp. 67–94, 1996.

[6] R. Haupt, *Antenna Arrays:A Computational Approach*. Wiley-IEEE Press, 2010.

[7] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

[8] E. Khan and D. T. M. Slock, "Direction of arrival estimation using quadratic time frequency distributions," in *IST 2003, 12th IST Mobile and wireless communications Summit, June 15-18, 2003, Aveiro, Portugal*, (Aveiro, PORTUGAL), 06 2003.

[9] A. El Gonnouni, M. Martinez-Ramon, J. Rojo-Alvarez, G. Camps-Valls, A. Figueiras-Vidal, and C. Christodoulou, "A support vector machine music algorithm," *Antennas and Propagation, IEEE Transactions on*, vol. 60, no. 10, pp. 4901–4910, 2012.

[10] L. Gupta and R. P. Singh, "Array signal processing: Doa estimation for missing sensors," in *Power, Control and Embedded Systems (ICPCES), 2010 International Conference on*, pp. 1–4, 2010.

[11] P. Stoica and K. Sharman, "Novel eigenanalysis method for direction estimation," *IEE Proceedings F, Radar and Signal Processing*, vol. 137, no. 1, pp. 19–26, 1990.

[12] B. Widrow, J. McCool, and M. Ball, "The complex lms algorithm," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 719–720, 1975.

[13] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Upper Saddle River, New Jersey, USA: Prentice-Hall, Inc., 1985.

[14] A. D. Poularikas and Z. M. Ramadan, *Adaptive Filtering Primer with MATLAB*. 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, Florida, USA: CRC/Taylor & Francis, 2006.

[15] T. van Waterschoot, G. Rombouts, and M. Moonen, "Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement," *Signal Processing*, vol. 88, no. 3, pp. 594 – 611, 2008.

[16] K. Mayyas and T. Aboulnasr, "Leaky lms: a detailed analysis," in *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, vol. 2, pp. 1255–1258 vol.2, 1995.

[17] D. Mandic, "A generalized normalized gradient descent algorithm," *Signal Processing Letters, IEEE*, vol. 11, no. 2, pp. 115–118, 2004.

[18] O. I. Frost, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, pp. 926–935, Aug 1972.

[19] R. Plackett, "Some theorems in least squares," *Biometrika*, vol. 37, pp. 149–157, 1950.

[20] C. Paleologu, J. Benesty, and S. Ciochina, "A robust variable forgetting factor recursive least-squares algorithm for system identification," *Signal Processing Letters, IEEE*, vol. 15, no. 2, pp. 597–600, 2008.

[21] J. L. Rojo-Álvarez, M. Martínez-Ramón, M. de Prado-Cumplido, A. Artés-Rodríguez, and A. R. Figueiras-Vidal, "Support vector method for robust arma system identification," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 52, pp. 155–164, JAN 2004.

[22] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[23] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge Univ. Press, 2000.

Appendices

Appendix A

MATLAB ALGORITHMS

Generate Input Data MATLAB

```
F=input('Signal Frequency (GHz) = ');
F=F*1*10^9; Fs=2.5*F;
T=1/F;lambda=(3*10^8)/F;
Samples=input('Number of Samples = ');
t=[0:Samples-1]*(1/Fs);
N=input('Number Elements of the array = ');
Jammers=input('Introduce the number of interference signals
    = ');
Dx=input('Array Dimension x direction (\lambda) = ');
Dx=Dx*lambda;
Dxnorm=Dx/lambda;
Dy=input('Array Dimension y direction (\lambda) = ');
Dy=Dy*lambda;
Dynorm=Dy/lambda;
```

```
14 dx=sort(random('Uniform',lambda/N,Dx,1,N-1));
15 dy=sort(random('Uniform',lambda/N,Dy,1,N-1));
16 dx=dx(randperm(length(dx)));
17 dy=dy(randperm(length(dy)));
18 dx=[0,dx];
19 dy=[0,dy];
20 Theta0=input('Angle of arrival (AOA) (Degree) =');
21 Theta0=Theta0*pi/180;
22 Phi0=input('Angle of arrival (AOA) Phi (Degree) =' );
23 Phi0=Phi0*pi/180;
24 Power0=input('Power desired signal (dB) =');
25
26 Thetai=input('Angle of interference signal  (Degree) =');
27 Thetai=Thetai*pi/180;
28 Phii=input('Angle of interference Phi signal  (Degree) =');
29 Phii=Phii*pi/180;
30 Poweri=input('Power interference signal (dB) =');
31
32 Thetaii=input('Angle of interference signal  (Degree) =');
33 Thetaii=Thetaii*pi/180;
34 Phiii=input('Angle of interference Phi signal  (Degree) =');
35 Phiii=Phiii*pi/180;
36 Powerii=input('Power interference signal (dB) =');
37
38 doas=[Theta0 Thetai Thetaii];
39 doas2=[Phi0 Phii Phiii];
40 P=[Power0 Poweri Powerii];
41
```

```
S=cos(2*pi*F*t);
u=rand(1,length(S));
v=rand(1,length(S));
u1=rand(1,length(S));
v1=rand(1,length(S));
I1=sqrt(-2*log(1-u)).*cos(2*pi*v);
I2=sqrt(-2*log(1-u1)).*cos(2*pi*v1);
Signal=[S;I(1,:);I(2,:)];
A=exp(-1i*2*pi*dx'*(lambda^-1)*sin([doas(:).'])-1i*2*pi*dy
    '*(lambda^-1)*sin([doas2(:).']));
Signal=[S;I(1,:);I(2,:)];
noise=sqrt(0.5*10^(-20/20))*randn(N,Samples);

X=A*diag(sqrt(P))*Signal + noise;
```

DOA MATLAB Algorithms

```
%% Capon  ALGORITHM
R=X*X'/length(S);
Rinv=inv(R);
theta=(-90:0.1:90)*pi/180;
C1=zeros(1,length(theta));
for  t=1:length(theta)
      Alf=exp(-1i*2*pi/lambda*dx'*sin(theta(t)));
      C1(t)=(Alf'*Alf)./abs(Alf'*Rinv*Alf);
end
theta=(-90:0.1:90)*pi/180;
C2=zeros(1,length(theta));
```

```
for t=1:length(theta)
    Alf1=exp(-1i*2*pi/lambda*dy'*sin(theta(t)));
    C2(t)=(Alf1'*Alf1)./abs(Alf1'*R1inv*Alf1);
end
```

```
%%MUSIC ALGORITHM
R=X*X'/length(S);
[Q ,D]=eig(R);
[D,J]=sort(diag(D),1,'descend');
Q=Q(:,J);
% The signal eigenvectors
Qsignal=Q(:,1:length(doas));

% The noise eigenvectors
Qnoise=Q(:,length(doas)+1:N);
```

Adaptive Beamforming MATLAB Algorithms

```
%% ****** LMS algorithm****** %%
for n=2:length(S)
            error(n)=S(n)-W(:,n-1)'*Y(:,n);
            W(:,n)=W(:,n-1)+mu*(Y(:,n)*conj(error(n)));
            MSELMS(n)=(error(n))^2;
end
```

```
%% ****** Leaky LMS algorithm****** %%
 for n=2:length(S)
            y(n)=w(:,n-1)'*Y(:,n);
```

```
4                   e(n)=S(n)-y(n);
5                   w(:,n)=(1-mu*gam)*w(:,n-1)+ mu*conj(e(n))*(Y(:,
                        n));
6                   MSE(n)=e(n)^2;
7   end

1   %% ****** Normalized LMS algorithm****** %%
2     for n=2:length(S)
3                 muvariable(n)=(mu)/(Y(:,n)'*Y(:,n));
4                 error(n)=S(n)-W(:,n-1)'*Y(:,n);
5                 W(:,n)=W(:,n-1)+muvariable(n)*(Y(:,n)*conj(error
                      (n)));
6                 MSE(n)=(error(n))^2;
7   end

1   %% ****** Adaptive Size Normalized LMS algorithm****** %%
2     for n=2:length(S)
3                   y(n)=w(:,n-1)'*Y(:,n);
4                   e(n)=S(n)-y(n);%ANLMS
5                   cfactor(n)=e(n)*e(n-1)*Y(:,n)'*Y(:,n-1);
6                   num=rho*mu*cfactor(n);
7                   den=Y(:,n-1)'*Y(:,n-1)+ beta(n-1);
8                   beta(n)=beta(n-1)- num/den;
9                   clear num den cfactor y
10                  eta(n)=(mu)/ ((Y(:,n)'*Y(:,n)) + beta(n));
11                  w(:,n)=w(:,n-1)+ eta(n)*Y(:,n)*conj(e(n));
12                  MSE(n)=e(n)^2;
13  end

1   %% ****** Constrained LMS algorithm****** %%
```

```
2            c1=ones(User,1)';
3            c2=zeros(Jammers,1)';
4            c=[c1  c2]';
5       Pa=eye(N)-A*inv(A'*A)*A';
6       Wa=A*inv(A'*A)*c;
7       W=zeros(N,length(S));
8       W(:,2)=Wa;
9       W1=zeros(N,length(S));
10      W1(:,2)=Wa;
11      W2=zeros(N,length(S));
12      W2(:,2)=Wa;
13      W3=zeros(N,length(S));
14      W3(:,2)=Wa;
15  for  n=2:length(S)
16           Z(:,n)=W(:,n)'*Y(:,n);
17           error(n)=S(n)-Z(:,n);
18          W(:,n)=Pa*(W(:,n-1)-mu*(Z(:,n-1)*Y(:,n-1)))+Wa;
19           MSE(n)=(error(n))^2;
20  end
```

```
1  %% ******  RLS algorithm ****** %%
2    for  n=2:length(S)
3                z=Rinv*Y(:,n);
4                g= z/ (lambda1 + Y(:,n)'*z);
5                alpha=S(n)-wRLS(:,n-1)'*Y(:,n);
6                wRLS(:,n)=wRLS(:,n-1)+g*conj(alpha);
7                y=wRLS(:,n)'*Y(:,n);
8                e(:,n)=S(n)-y;
9                Rinv = (lambda1^(-1)*Rinv)- (lambda1^(-1)*g*Y(:,
```

```
                n)'*Rinv);
10              MSE1(n)=e(n)^2;
11  end
12 %% ******  VFF-RLS algorithm ****** %%
13 for n=2:length(S)
14                  z1=Rinv1*Y(:,n);
15                  g1=z1 / (lambda1(n-1) + Y(:,n)'*z1);
16                  alpha1=S(n)-wRLS1(:,n-1)'*Y(:,n);
17                  wRLS1(:,n)=wRLS1(:,n-1)+g1*conj(alpha1);
18                  y1=wRLS1(:,n)'*Y(:,n);
19                  e1(:,n)=S(n)-y1;
20                  q1(n)=Y(:,n)'*Rinv1*Y(:,n);
21                  sigmae1(n)=a*sigmae1(n-1)+(1-a)*e1(n)^2;
22                  sigmaq1(n)=a*sigmaq1(n-1)+(1-a)*q1(n)^2;
23                  sigmav1(n)=b*sigmav1(n-1)+(1-b)*e1(n)^2;
24                  gamma1(n)=sigmae1/sigmav1;
25                  Rinv1 = (lambda1(n-1)^(-1)*Rinv1)- (lambda1(
                        n-1)^(-1)*g1*Y(:,n)'*Rinv1);
26                  h1=sigmaq1(n)*sigmav1(n)/(c+abs(sigmae1(n)-
                        sigmav1(n)));
27                  lambda1(n)=min(h1,lambdamax1);
28                  MSERLS1(n)=e1(n)^2;
29 end
```