A Novel NEC2/Matlab Interface

Applied To

A Study of Adaptive Antenna Array Signal Processing

BY

JAMES W. DIETRICH

A Thesis submitted to
The Faculty of Graduate Studies
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

The Faculty of Graduate Studies
500 University Centre, University of Manitoba
Winnipeg, Manitoba R3T 2N2

Phone: (2040 474 9377
Fax: (204) 474 7553
graduate_studies@umanitoba.ca

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

A Novel NEC2/Matlab Interface

Applied To

A Study of Adaptive Antenna Array Signal Processing

BY

JAMES W. DIETRICH

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

The purpose of this thesis is two-fold. First, to develop an interface between Matlab and Numerical Electromagnetics Code (NEC2). Matlab is a popular commercial mathematical programming environment. NEC2 is a command-line driven numerical electromagnetic field solver in popular use due to its powerful capabilities to handle many varieties of wire antennas and their feed networks, and the fact that its Fortran source code is freely available. The second and main purpose of this thesis is to use this NEC2/Matlab interface to study adaptive antenna array processing.

With its origins in adaptive signal processing and adaptive filter design, adaptive antenna array processing has developed much by direct analogy of spatial filtering in antenna beam-space with signal filtering in frequency space. Two methods of generalized sidelobe cancelation are studied, the Linearly Constrained Minimum Variance (LCMV) method and the Least Mean Squares (LMS) method. The objective is to reduce antenna array gain at the angle of arrival of interference. It is shown that both methods have strengths and weaknesses, but that the LMS method is more robust for a wider range of interference angles. The LCMV method, at least as formulated in this thesis, has some extreme problems that are identified and discussed.

Typically, geometric interference patterns, known as the array factor, are used for a first-order approximation to antenna array patterns but do not address the effects of mutual coupling between antennas. The computed solution of NEC2 does account for this, however its need for cryptic text files to represent structures and define simulations

is cumbersome. The Matlab interface automates much of this process and overcomes this obstacle in addition to offering powerful graphical display options for the output.

The interface allows for the automated study of the adapted array patterns throughout the adaptive process and over a range of conditions. This study shows that the computed radiation pattern of the array can differ dramatically from the array factor. This knowledge can be used to design optimized antenna arrays to perform closer to expectation in real world applications.

Acknowledgements

I would like to thank Dr. Abdel Sebak for his patience and support throughout this process, and in particular for being an inspiring teacher of electromagnetics and antenna theory throughout my studies. I express my gratitude for the financial support received from NSERC via Dr. Sebak.

I am thankful to have supportive family and friends who have inspired me with encouragement, assistance and example. I am particularly grateful to my parents, for their assistance and faith in me, and to Dr. Grant Sims, a friend and mentor who supports what he believes in and " walks the walk" .

I have really enjoyed and must acknowledge the great classmates and colleagues I have met through the University of Manitoba. It has been a truly open-minded think tank fuelled by the pursuits of knowledge, science as service, fun and just figuring things out.

And finally, I must pay my respects to the myriad experiences of life itself. Life, the teacher, with all its challenges and opportunities to be and to do, the ones that get us up early and keep us up late at night because of the quest, the itch, the thrill or the hunger for life that tells us there' s plenty of time to sleep when we' re dead. I like the way it was expressed by the character Roy Batti, Nexus 6 Replicant, in the movie *Bladerunner*:

" All these moments lost in time,
like teardrops in the rain..."

Table of Contents

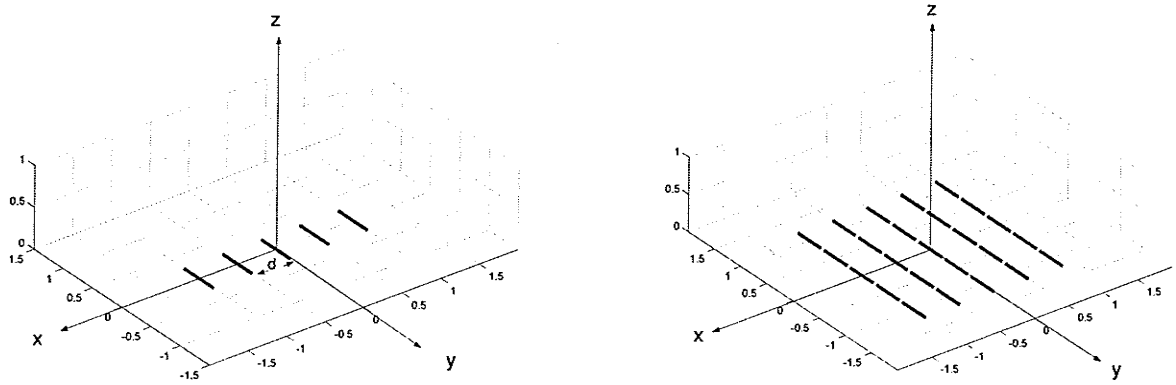
List of Figures

List of Tables

1.0 Introduction

The purpose of this introductory chapter is to introduce the three facets of this thesis, antenna arrays, NEC– the numerical electromagnetic computation program with its Matlab interface, and adaptive processing as applied to antenna arrays.

A brief overview of each topic is given, and the appropriate conventions for symbols, configuration and notation used in this document are established. Vectors and arrays appear in boldface while scalars are given in italics.



a) Linear array **b)** Planar Array

Figure 1.1– Linear and planar array of wire antennas.

1.1 Array Theory Background

Consider a linear or planar array of antenna elements, as shown in Figure 1.1. For the case of all elements being coplanar, as is considered in this work, the antenna elements are defined to lie in an *xy* plane, with the *z*-axis perpendicular to the array and pointing in

a direction defined as *broadside*. An *antenna element* can be any physical antenna, such as a wire monopole or dipole, a microstrip patch or a microwave horn.

Incoming radiation is assumed to be a plane wave, and its direction is measured with respect to the z-axis using the standard spherical polar coordinates θ, the angle as measured from the z-axis and ϕ, the angle as measured from the x-axis. Angles can be in radians or degrees, and units will be given.

The three basic considerations of an antenna array are *1)* the array factor– the geometric interference of waves given the geometry of the array, *2)* the element pattern– the radiation pattern produced by a single antenna, and *3)* the mutual coupling of antenna elements when positioned in an array– a basic effect of Faraday' s law of induction.

This thesis deals only with the case of a linear array of wire dipole antennas, but the analysis can readily be extended to planar arrays of wire antennas or wire-mesh approximations to surfaces using NEC.

The array factor of an array of antenna elements is the geometric interference pattern resulting from the positions of the elements. In general, elements are spaced apart distances less than or equal to a half-wavelength ($d \leq \lambda/2$) of the design frequency. Inter-element spacing larger than $\lambda/2$ gives rise to grating lobes. Grating lobes are a periodic repetition of the primary pattern due to the spacing, a form of spatial sampling theorem directly analogous to the Nyquist theorem for discrete time sampling.

Given a linear array of elements, with an incident plane wave arriving at θ_o, the signal received at each element will differ in phase based upon the propagation constant, k, and the element spacing, d, as shown in Figure 1.1. As can be seen, the delay distance is

$d \sin\theta_o$, which, when combined with the propagation constant, $k = {}^{2\pi}/_{\lambda}$, produces a phase delay of $\phi_o = kd \sin\theta_o$, between adjacent elements. The desired signal is obtained at the array output by delaying and summing the signal at each antenna element.

The inter-element phase delay, given by

$$\phi_o = kd \sin\theta_o \tag{1.1}$$

is known as electrical angle of arrival.

Consider the linear antenna array of M elements shown in Figure 1.2, illuminated by an incident plane wave, which when sampled is of the form

$$u(n) = e^{j(\omega_u n - k\bar{r})} \tag{1.2}$$

where $\omega = 2\pi f$ is the angular frequency, k is the wavenumber, and n would take on

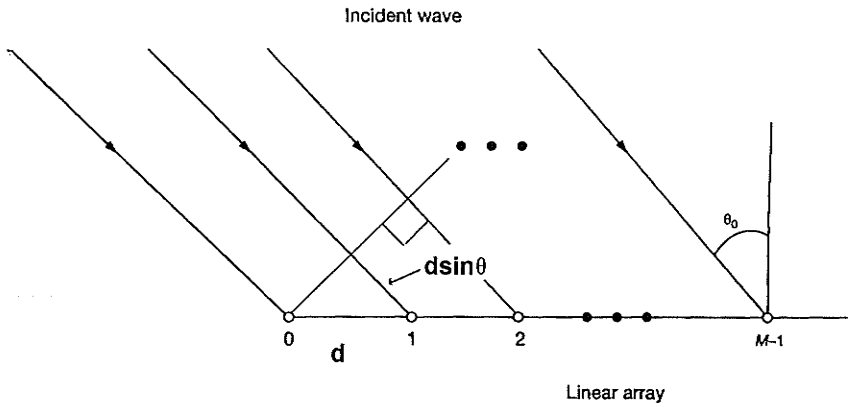


Figure 1.2– An incident plane wave arriving at a linear array at an angle of θ_o from normal to the array [1].

uniformly spaced instances in time given the sampling frequency, F_s.

For a linear array of elements, with an incident plane wave arriving at θ_o, the signal received at each element will differ in phase based upon the propagation constant, k, and the element spacing, d, as shown in Figure 1.2. Given the separation distance of d metres, the path length difference is $d \sin\theta_o$ metres. At any given instant in time, a line perpendicular to the direction of propagation of the plane wave is, by definition, a line of constant phase. For a propagation constant, or wavenumber, of $k = {}^{2\pi}/_{\lambda}$ radians per metre, this translates into a phase delay of $\phi_o = kd \sin\theta_o$ radians between each element. This quantity ϕ_o, called the electrical angle of arrival, is an inter-element phase delay. The desired signal is obtained at the array output by delaying and summing the signal at each antenna element.

If an end element is designated as the first, or 0^{th}, element, the phase progression across the array is given by

$$s(\phi) = [1, e^{-j\phi}, \ldots, e^{-j(M-1)\phi}]^T \qquad \textbf{(1.3)}$$

where T is the transpose operator, and M is the number of antenna elements. This vector, $s(\phi)$, is called the steering vector because it steers the direction of the mainbeam of the array. That is, by multiplying the vector of signals received at the antenna terminals by the steering vector, the inner product is the delayed and summed output of the array.

The output of the array, $y(n)$, is given by

$$y(n) = u_0(n) \sum_{k=0}^{M-1} w_k^* e^{-jk\phi_0} \qquad \textbf{(1.4)}$$

where $u_o(n)$ is the signal received at time n by the end antenna element #0 in Figure 1.2.

The w_k^* in (1.4) represent the driving-point voltage or current for a transmit mode, or the scaling factor (weighting) of the electrical signals (current or voltage) at the antenna terminals in receive mode.

The normalized array factor for a uniform weighting is given by

$$AF = \frac{1}{M}\sum_{k=0}^{M-1} e^{-jk(\phi_0-\phi)} = w_q^H s(\phi) \qquad \textbf{(1.5)}$$

where H is the Hermitian operator of conjugate transpose, ϕ_o is the electrical scan angle and ϕ is the electrical angle of the point in space where the array factor is to be calculated. To produce a normalized directive gain radiation pattern for a given array scanned to an electrical angle ϕ_o across the field of view given by $-90° \le \theta \le 90°$, one need only choose a suitable number of θ angles as far-field sample points, convert them to the electrical angle using (1.3), and perform the summation given in (1.5), or perform the vector inner product of the weights with the steering vector.

A typical array factor directive gain pattern is as shown in Figure 1.3 for an array of 7 elements spaced at $d = \lambda/2$ and scanned to $\theta = 15°$.

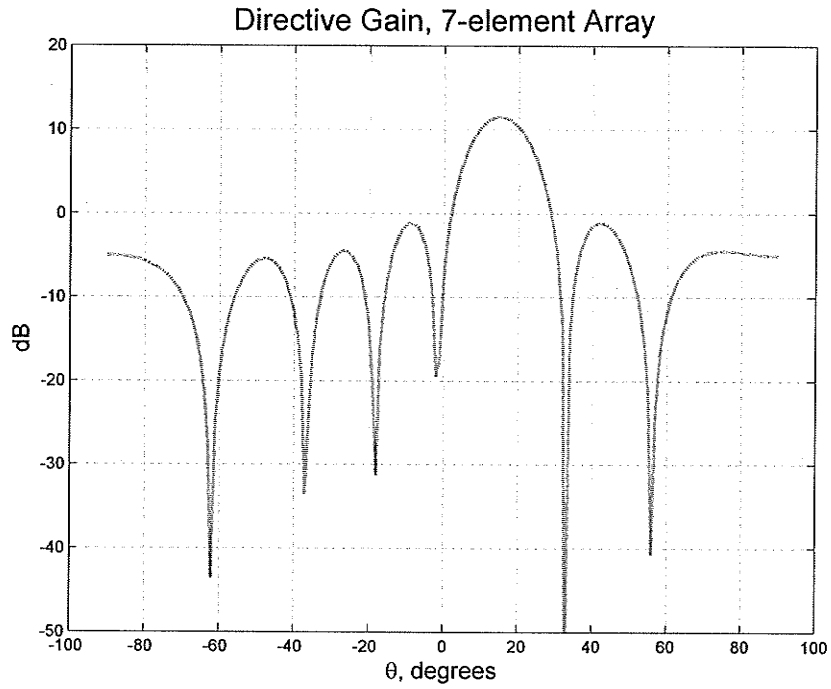


Figure 1.3– Array factor example for 7-element array scanned to 15°.

1.2 NEC2– Numerical Electromagnetic Computations

NEC2 is a popular program, with available source code, for the purpose of numerical electromagnetic computations based on the moment method. The original program NEC was written in the early 1980's at Lawrence Livermore Laboratories, and was designed to take its input from punch cards. A text file has replaced the punch card, but the data arrangement follows the same 80-column format. The output is written to an ASCII text file, with headers followed by tables of values.

NEC is a "full-wave" electromagnetic field solver, which means that it determines the overall electromagnetic behavior from the charge and current distribution resulting from

sources applied to a radiating structure. NEC2 is double precision, a modification introduced in 1985. The Internet NEC Archives [2] contains the updated source code in C and Fortran, with executables compiled for several platforms.

The original Numerical Electromagnetics Code (NEC) was developed during the late 1970 s at the Lawrence Livermore Laboratory by contract of the United States Naval Ocean Systems Center and the Air Force Weapons Laboratory. Further development by G. J. Burke and A. J. Poggio of Lawrence Livermore Laboratory resulted in the presently available NEC.

The popularity of NEC is apparent by the volume of daily email from the Internet mailing list, NEC-LIST [3], and the number of links returned using an Internet search engine. The NEC-LIST site gives links to the available Fortran source code and some pre-compiled NEC executables. Subscribers to the NEC-LIST enter a global forum for exchanging discussions, questions and answers about using NEC for a wide variety of applications, from antenna analysis to electromagnetic compatibility. Often, the discussion centers on how to get a better graphical representation of the tabular output files.

There are several commercial NEC packages available, built on the freely available source code, and offering a graphical user interface (GUI) and post-processing visualization tools. There is Mini-NEC from EM Scientific Corp. [4], the NEC-Win series from Nittany Scientific, Inc. [5] and the popular SuperNEC from The Poynting Group, [6]. The prices range from a few hundred to a few thousand dollars, with some limited version free downloads available.Each commercial package has varying strengths and weaknesses, and is priced accordingly.

In order to implement any systematic study of many small changes in an array geometry in Mini-NEC, for example, would have the user navigating through GUI menus manually setting each one. Such a tedious and time-consuming activity is eliminated with an interface that performs these changes automatically as part of an optimization-driven process. SuperNEC offers such a process in GUI environment, but is at the top of the price scale for Software built on the NEC engine. There are other independent full-wave electromagnetic field solvers on the market for tens of thousands of dollars, utilizing various solver techniques - the *finite-element method* (FEM), the *method of moments* (MOM), the *boundary-element method* (BEM) or the *finite-difference time domain method* (FTDT). The FEM calculates a solution based on a discretization of space with absorbing boundaries to represent infinite space. The MoM and the BEM compute a solution based on discretization of the model structure and fields can be calculated anywhere in space. The FDTD produces a solution by considering the instantaneous interaction at discrete points in time.

NEC2 returns a steady-state frequency domain solution to the input problem file based on the MoM. The underlying principle of the moment method is the numerical solution of integral equations for the currents induced on a structure by sources and incident electromagnetic fields. In general, the desired unknown quantity is either the current or charge distribution, and it appears within the integrand of the integral equation. In order to solve this inverse problem, the integral is reduced to a series of linear algebraic equations and solved using matrix equation methods.

Briefly described, the moment method solution has the general form

$$F(g) = h \tag{1.6}$$

where F is the known linear operator (integration), h is the known excitation function, and g is the desired response function (charge or current). The moment method technique is to represent g as a linear combination of N terms, which, for a thin wire located on the z-axis, is given as [7]:

$$g(z') = \sum_{n=1}^{N} a_n g_n(z') \tag{1.7}$$

By convention, primed coordinates represent source points and unprimed coordinates represent observation points.

The coefficients a_n are unknown constants to be determined, and the g_n are known basis functions, also called expansion functions. Substituting (2) into (1) and using the principle of linearity to bring F into the summation gives

$$\sum_{n=1}^{N} a_n F(g_n) = h \tag{1.8}$$

However, this is still one equation with N unknowns. In order to solve for the N unknown coefficients, a_n, (3) must be evaluated while enforcing the boundary conditions at N points. This, known as point-matching, or collocation, results in N equations of the form [7]:

$$\sum_{n=1}^{N} a_n F(g_n) = h_m, \quad m = 1, 2, \ldots N \tag{1.9}$$

which can readily be expressed as the matrix equation

$$[F(g_n)][a_n] = [h_m] \tag{1.10}$$

where $[F(g_n)]$ is an $N \times N$ matrix. The unknown a_n can be found using matrix inversion and multiplication, as shown in (6).

$$[a_n] = [F(g_n)]^{-1} [h_m] \tag{1.11}$$

The above has been a simplified overview of the Method of Moments. To improve the accuracy of the collocation solution given in (4), an inner product of the basis function with a weighting, or testing, function is frequently implemented. The problem is that the boundary conditions are only enforced at the collocation points in (4) and may deviate significantly between points. This deviation, called a residual, can be minimized such that its average over the entire structure is zero using a method of" weighted residuals" in conjunction with the inner product. There are many published works on this subject. When the basis and weight functions are the same, the procedure is known as Galerkin' s method. In NEC, the basis and weight functions are different. The weighting function is the Dirac delta function, resulting in the boundary conditions only being enforced at discrete points of the structure. For wires, the sample point is the midpoint of each segment, and for surface patches, it is at the center of each patch.

For time-harmonic electromagnetic fields, the integral equations are commonly represented as the electric field integral equation (EFIE) and the magnetic field integral equation (MFIE). The EFIE enforces the boundary condition on the tangential electric field, and is well suited for thin-wire structures of small conductor volume. The EFIE is valid for open or closed conductor surfaces, and is preferred for thin structures or surfaces separated by a small distance. The MFIE enforces the boundary condition on the tangential magnetic field components, which fails for the thin wire case due to being

valid only for closed surfaces. It is preferable, however, for large structures of smooth surfaces. The NEC code uses both the EFIE and the MFIE, and for models containing both wires and surfaces, the two are coupled. A rigorous derivation of how the EFIE and MFIE are used in NEC is given in [8][9].

NEC can support non-uniform segmentation of wires, and represents the current on each short, straight segment by three terms - a constant, a sine and a cosine. The added advantage of using sinusoids is that the integrations can be computed analytically, thus sparing computation time and discretization error. Currents on surface patches are modeled as a set of pulse functions (constant levels), except where a wire joins a surface patch. Current continuity and charge balance are enforced at all nodes.

Due to the nature of the Green' s functions (kernels), the choice of basis functions is much more critical for wires than for surface patches. All patches will use the pulse basis functions, but for wire geometry, the user has the choice of a thin-wire, extended thin-wire or large-distance kernel. The thin-wire kernel is appropriate when the ratio of the wire length to the wire radius is greater than 8, and the extended thin-wire for a ratio greater than 2 but less than 8. In both cases, the wire should be divided into at least 10 segments per wavelength. The large-distance kernel uses a current element approximation, modeling the segment current as an infinitesimal current element at the segment' s center.

1.3 Automating NEC2 with Matlab

Matlab® is a very powerful matrix and math software package, with advanced file I/O and graphical display capabilities. This work demonstrates the combination of NEC2, for

the electromagnetic computation, with Matlab, for the composing of input files and the graphical display of the results. Because the input file to NEC2 adheres to strict rules, the numerical data can be prepared in Matlab and then concatenated with the appropriate strings and written to a text file. Matlab can call the NEC2 executable from within one of its scripts (called m-files), and then read in the resulting NEC2 output file for further analysis. In essence, NEC2 becomes a solver driven by Matlab. Given an optimization strategy, Matlab can iterate the above-mentioned process until convergence or another criteria are met. This work presents Matlab-NEC2 automation as applied to adaptive array antenna synthesis and design.

1.4 Adaptive Antenna Array Background

Adaptive array antennas define a class of antenna arrays combined with signal processors for the purpose of recursively or adaptively changing the antenna parameters. The aim is to optimize the overall antenna pattern for desired performance. One example is to minimize the reception of noise interference from the current environment, ideally as the noise background changes. The adaptive signal processing is implemented in the feed structure of the array.

Whereas a fixed beam, switched beam or phased array will have a pre-selected amplitude distribution with the beam directed by a progressive phase shift across the elements, an adaptive array uses the amplitude and phase of each element as a complex weight that can be optimized.

There are applications, such as broadcasting, which benefit from wideband omni-directional antenna systems, but there is increasing demand for highly directive antennas

that can be rapidly scanned or beamformed to any desired direction or shape. Applications where high power interference or jamming is a problem require some method of steering nulls to the source of interference, while maintaining gain in the desired direction. Cellular systems where capacity is limited by multi-user interference can utilize greater capacity using beamforming and null-steering networks [10][11][12]. Target tracking and intelligent signal detection systems can be optimized by developing antennas that focus on the moving target while eliminating sources of interference that lead to false alarms [13][14]. These systems are in use in such applications as radar, air traffic control, and naval ice detection.

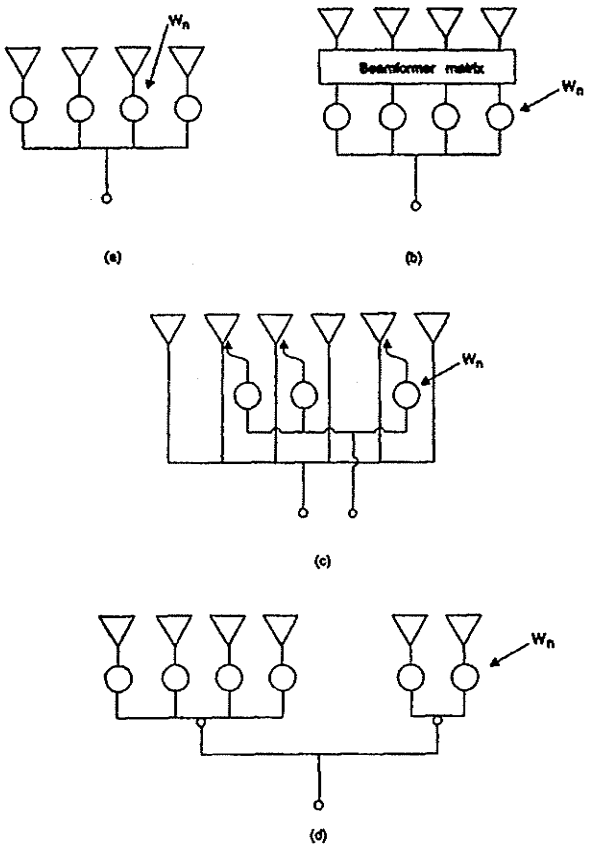
An adaptive array uses signal processing to adjust the amplitude and phase weights of each element. By so doing, the main beam can be steered to the desired direction while nulls are placed in the directions of interference.

Investigations into array adaptivity began in the mid-1950 s with Paul W. Howells and Sidney Applebaum [15], who began working with the Heavy Military Equipment Department of the United States to overcome radar sensitivity to jamming. Their initial method is as shown in Figure 1.4(d), and was patented as the IF Sidelobe Canceller.

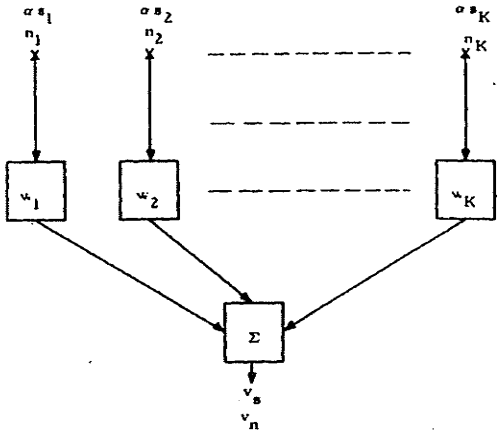
It was known that a uniformly weighted array gives the optimal signal to noise ratio (SNR) when the noise power at each element was equal and uncorrelated. This is valid for inherent element and sky noise. It was found, however, that when there is directional noise from other in-band transmitters or jammers, the noise is correlated and may arrive at the elements with unequal power. Thus, uniform weighting will not yield the best SNR. Sidney Applebaum published a seminal paper [16] in which he tackled the problem

of calculating the array weights to maximize the SNR for any type of noise environment. The result became known as the *Howells-Applebaum Method.*

Applebaum's block diagram for the optimum linear combiner is shown in Figure 1.4(e). He referred to this adaptive array as a linear combiner because of the weighted summation of antenna element outputs. Compare the similarity of this diagram to that shown in Figure 1.4(a).



Types of adaptive arrays: **a)** Delay and sum, **b)** Beamformer matrix, delay and sum, **c)** Partially adaptive, and **d)** Auxiliary detection [17].



e) Applebaum's block diagram for an optimum adaptive array [1].

Figure 1.4– **a)** to **d)** Basic types of adaptive arrays, **e)** Sydney Applebaum's 1966 initial investigation into the subject is an application of type a).

In the" delay and sum" beamformer, the signal at the antenna terminal is multiplied by the complex weight w_i to scale it (amplify or attenuate) by the modulus and delay it by the phase before being summed with the other antenna outputs. It turns out that this is a direct analogy to transversal filtering. The antenna array is a spatial representation of the discrete time-domain filtering process.

This thesis looks at the delay and sum beamformer as a generalized sidelobe canceller (GSC) using two different algorithms.

2.0 Development and Testing of NEC/Matlab Interface

2.1 NEC2 Input/Output Format and MATLAB

The applied source can be an incident plane wave, for scattering or electromagnetic compatibility analysis, or an applied delta-gap voltage source. In addition, non-radiating networks such as transmission lines, lumped circuit networks, or a combination of both, connecting different parts of the model can be included. In this way, impedance loading at various locations of the design can be modeled. The presence of a ground plane can be modeled, which requires an additional program, SOMNEC, to be run prior to NEC, to produce a text file to be used in conjunction with the Sommerfeld/Norton approximation.

The output file consists of a restatement of the input problem - the geometry, frequency, loading, environment, excitation and the time to fill the matrix. The segmentation data is listed, giving the coordinates of each segment' s center, its length and radius, and the nodes to which it is connected.

The remaining output depends on the requested quantities, and may be the currents on each segment, the near fields, far-field power patterns, charge densities on wire segments, maximum coupling between segments and the driving-point impedance for each assigned voltage source.

Originally, the input was entered on a set of 80-column punch cards, one card per command. This has been replaced with an input text file. The typical format is one line per command, with the first two columns containing a mnemonic indicating the type of command, such as GW for" geometry wire" . The remaining columns specify the wire' s

details, such as the wire number (called Tag number), number of segments, radius and coordinates. An example of an entry line to generate a string of segments to represent a straight wire is:

```
GW,1,15,-2.9,-0.24,0.25,-2.9,0.24,0.25,0.001
```

The line describes wire number 1 composed of 15 segments, from (x_1, y_1, z_1) = (-2.9,-0.24,0.25) to (x_2, y_2, z_2) = (-2.9,0.24,0.25), and with a radius of 1mm. All dimensions are given in metres.

The fields are described in Table 2.1, where if a parameter begins with" I' the field will contain an integer, and if it begins with" F' the field will contain a floating-point value.

If the user wishes to use units other than metres, the geometry must be scaled to metres using a Geometry Scale (GS) command. The GS command must appear after the geometry input, and before the Geometry End (GE) command.

The input file format begins with comment lines, then geometry data, then program control cards. The program control cards indicate such parameters as the kernel to be used (KH), frequency (FR), presence of ground (GN), excitation (EX), loading (LD), and then the output desired such as coupling (CP), near (NE, NH) or far (RP) fields, output plotting data (PL) and finally the command to execute the solver (XQ). These commands can appear in any order, and can cause several executions of the solver within one file. These commands and there use are described fully in the manual given in [1].

Matlab, like C or Fortran, has powerful file input and output capabilities. In addition, Matlab offers the user a workspace in which to easily view data and apply mathematical and numerical procedures.

Table 2.1– NEC Geometry input parameters for a wire element.

Cols.	Parameter	Description
1-2	GW	Wire specification
3-5	I1	ITG - Tag number
6-10	I2	NS - # of segments
11-20	F1	XW1 - x_1 coord.
21-30	F2	YW1 - y_1 coord.
31-40	F3	ZW1 - z_1 coord.
41-50	F4	XW2 - x_2 coord.
51-60	F5	YW2 - y_2 coord.
61-70	F6	ZW2 - z_2 coord.
71-80	F7	RAD - radius

External executables can be called from within Matlab M-files (program scripts) by preceding the command with a" !" character. Thus, for this work, the NEC2 code was modified to always look for a file named" input.nec" and always output to a file named "output.txt". Matlab can then be directed to write the data to the input.nec text file, run the solver, and then read in the output.txt file to plot the output, or make decisions based on the output and run another simulation.

It should be noted that NEC has powerful capabilities to run various scenarios within one input file. For instance, if the user wants to calculate the scattered fields due to several

incident fields, it is more efficient to execute the solver for each incident field within one file, rather than have Matlab re-run the analysis for each incident field. This way, the matrix need only be built once in NEC and only the excitation changes. For small files, there may not be much difference, but for large models the time saving could be considerable. The result is only that Matlab needs to be programmed to read in and sort out a larger output file. However, with Matlab's text string search capabilities, this poses no difficulty.

In addition, NEC has commands to exploit symmetry, rotate geometry and repeat geometry, so the user can take advantage of these commands in the composition of input files. Finally, a fixed structure and its environment can be modeled and the factored interaction matrix saved in a file. This is called the numerical Green's function (NGF) and in subsequent runs it can be read in to replace the free-space Green's function. The purpose is to avoid re-calculating an entire complex model when only a part of the model will be modified. Thus, if an antenna is moved in an otherwise fixed environment, the self-interaction matrix of the environment can be saved and only the self-interaction matrix for the antenna and the mutual antenna to environment interactions need to be calculated. These points are made to indicate that some motivations for automation in Matlab may already be more efficiently handled in the NEC solver. Knowing this, these features can still be taken advantage with in the Matlab workspace.

2.2 Using MATLAB with NEC2 Input/Output Files

The NEC output files are composed of headers followed by columns of data. The headers describe the subsequent data, such as radiation pattern, and contain the column headings to identify the data below - angle, field quantity, polarization and so on. If a run is taking advantage of NEC' s ability to evaluate volumetric spatial data, particularly for several excitations, these files can become very large. Without some form of visualization tool, the data is difficult to review.

With Matlab' s text search capabilities, the data can readily be parsed into the workspace. Particular identifying strings in the headers are found, and then knowing the format of the following data, it is read into an array.

The following example calculates the current distribution and scattered fields for a linear array of 13 dipoles. The dipoles are parallel to the y-axis, 0.4836m in length and 0.25m above the xy-plane. The array is centered on the x-axis; the elements are separated by 0.4836m. The excitation is a linear y-polarized plane wave, arriving from $\theta = 75°$ and $\phi = 0°$.

A Matlab script composes the NEC input file, and requests the radiation pattern for 1801 points from $-90° \leq \theta \leq 90°$ when $\phi = 0°$, and for 361 points for $0° \leq \phi \leq 360°$ for $\theta = 90°$. In addition, the current magnitude and phase at the central node of each element is returned. One example of the output is as shown in Figure 2.1. Matlab has an extensive graphical visualization capability, including three-dimensional plots with shading.

In addition to the generated figure, the following data is printed in the Matlab workspace, shown in Figure 2.2.

From the output and workspace data, we see that scattered field peaks at -73.3°, which is nearly the angle of incidence, and that the beam is quite wide. Glo and Ghi are the power gain at the two points where the beamwidth was calculated. As can be seen, due to the angle, it is not quite a 3 dB beamwidth. For such a simple array, the program executed in a fast 4.44 seconds. Figure 2.1 indicates that the phase progression is what is expected for the angle of the scattered field, and the current magnitude is greater at the incoming end of the array. The azimuthal pattern shows strong endfire scattering, and in combination with the elevation pattern indicates an overall disk-shaped scattered field in 3-D space.

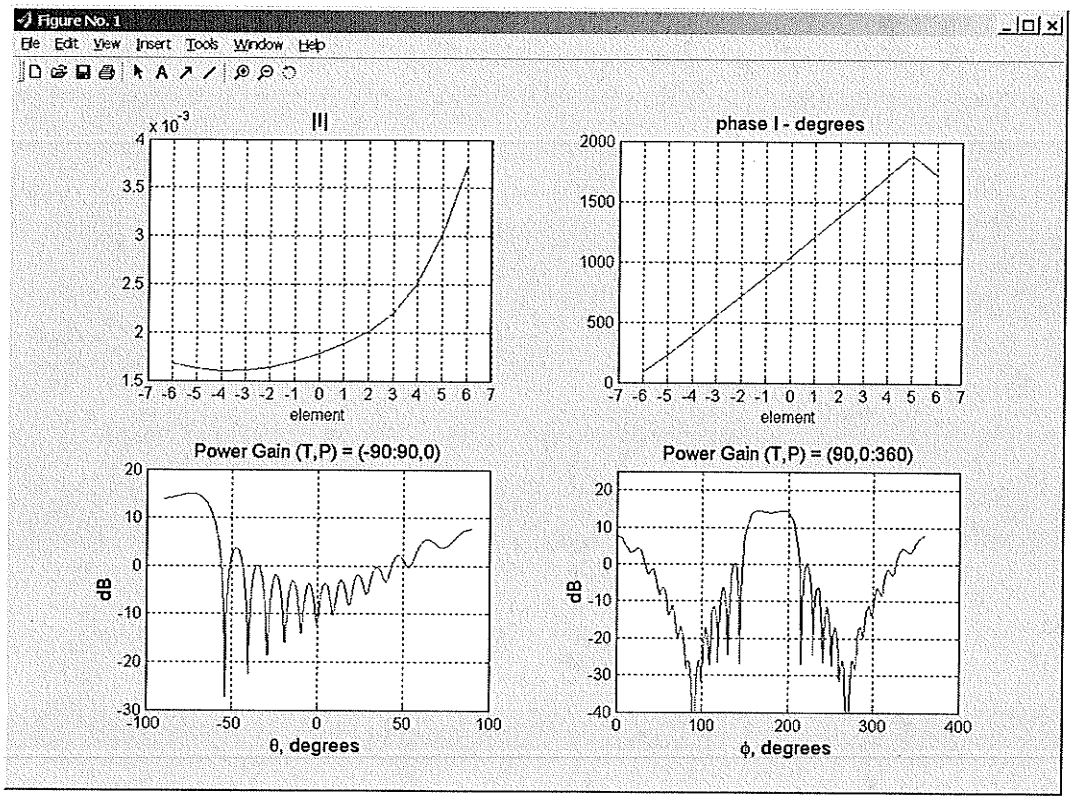


Figure 2.1 - Matlab figure of output data.

```
>> runit

Numerical Electromagnetics Code, double precision version (nec2d)

developed at Lawrence Livermore Lab., Livermore, CA., by G. Burke

(burke@icdc.llnl.gov) and A. Poggio.

Fortran file was created 4/11/80, last changed: Jan 15, 96, by

J. Bergervoet (bergervo@prl.philips.nl)



Maximum number of segments in core : MAXMAT=       2800



Stop - Program terminated.

time = 4.436 seconds.



----- job complete -----

Max Gain: 15.0326 at theta = -73.3 degrees.

HPBW = 27.1

Glo = 13.9836, 1.049 dB down

Ghi = 12.0644, 2.9682 dB down

>>
```

Figure 2.2 - Matlab workspace output.

2.3 NEC2 for Mutual Coupling Analysis of a Finite Phased Array Antenna

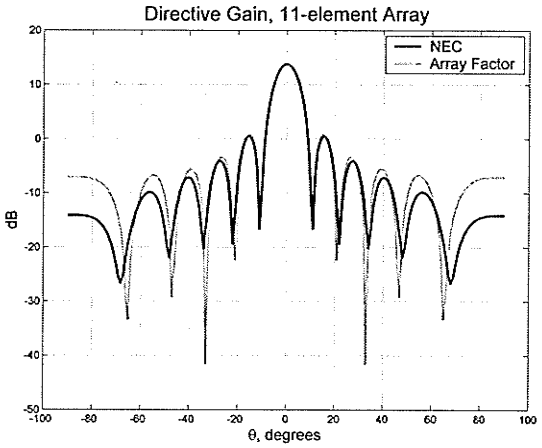
Having tested and verified the operation of the Matlab interface with NEC, it is of interest to compare the output of NEC with that of the array factor to observe the computed effects of mutual coupling. It should be noted that the Matlab interface is not necessary to compare NEC results with array factor results. An individual can create the NEC input files with a text editor, run NEC2 from a command line, parse the text file output into a graphical plotting or visualization package and then do the same with the output of an array factor calculation package. An individual can easily program the array factor output in the language of their choice using the equation given in (1.5). The Matlab interface as described in the previous chapter is intended to simplify and automate this process while taking advantage of Matlab' s powerful visualization tools.

The first comparison looks at an 11-element array of half-wavelength dipoles arranged as shown in Figure 1.1(a). Figure 2.3 shows the plots comparing the output of NEC2 with the Array Factor for scan angles 0°, 20°, 40°and 60°.

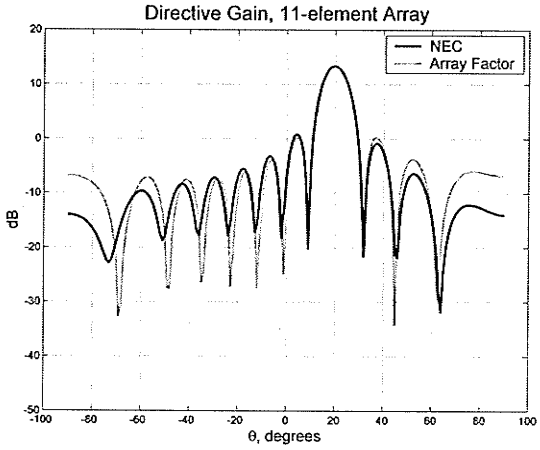
Matlab has features that allows the user to zoom in and out and pick points from anywhere in the graphical region, on or off of a trace. It returns the coordinates to a variable in the workspace. These features are useful in identifying points of interest on output plots.

From Figure 2.3(a), the radiation pattern of NEC is seen to be in excellent agreement with the array factor to the third sidelobe level (SLL). The broadside half-power

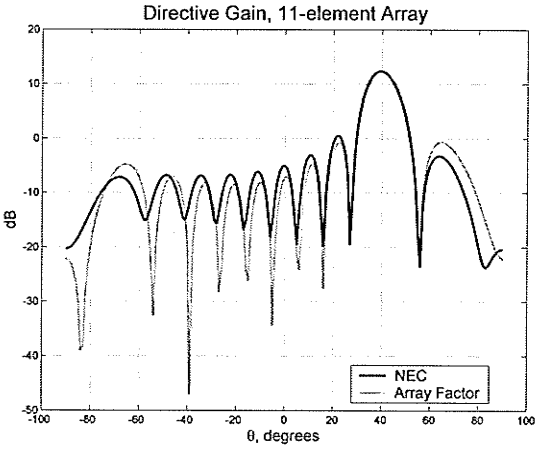
beamwidth (HPBW) is measured at 9.5° with a first SLL 13.3dB down from the mainbeam.



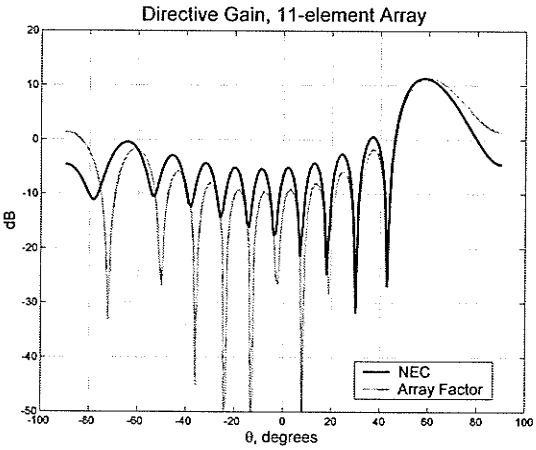
a) Mainbeam at 0°.



b) Mainbeam at 20°.



c) Mainbeam at 40°.



d) Mainbeam at 60°.

Figure 2.3– Plots comparing the output of NEC2 with the Array Factor for scan angles 0°, 20°, 40°and 60°.

The array factor theory as outlined in [7] gives closed-form equations to approximate the performance of a broadside array of M elements. A comparison of computed points and the results of these equations are given in Table 2.2 alongside a zoom in of the broadside

Table 2.2– Comparison of computed NEC output and Array Factor output with theoretically expected results.

Broadside Array ***M = 11*** **elements**	**NEC2**	**Array Factor Theory [7].**
Mainbeam Gain, dB	13.737	10.4
HPBW, °	9.5	9.24
First Null, °	± 11	± 10.5
First SLL, dB	0.6	0.28
First SLL, °	± 15.3	15.8
Second Null, ° (AF)	± 21	± 21.3
Second Null, ° (NEC)	± 22	± 21.3

mainbeam and first sidelobes shown in Figure 2.4. The broadside maximum gain is 13.7dB.

Looking at Figure 2.3 and Figure 2.4, the NEC computed effects of mutual coupling show a slight broadening of the mainbeam, and an outward displacement of the nulls starting from the second null. Also, NEC shows a general lowering of the outward nulls for broadside and near scan, and a rising of the outward nulls for larger scan angles. This generally is not an issue in practice, as the outer sidelobes usually are of low enough gain (>20dB down at the third SLL) to not be a concern unless a very strong source of interference is located in that direction.

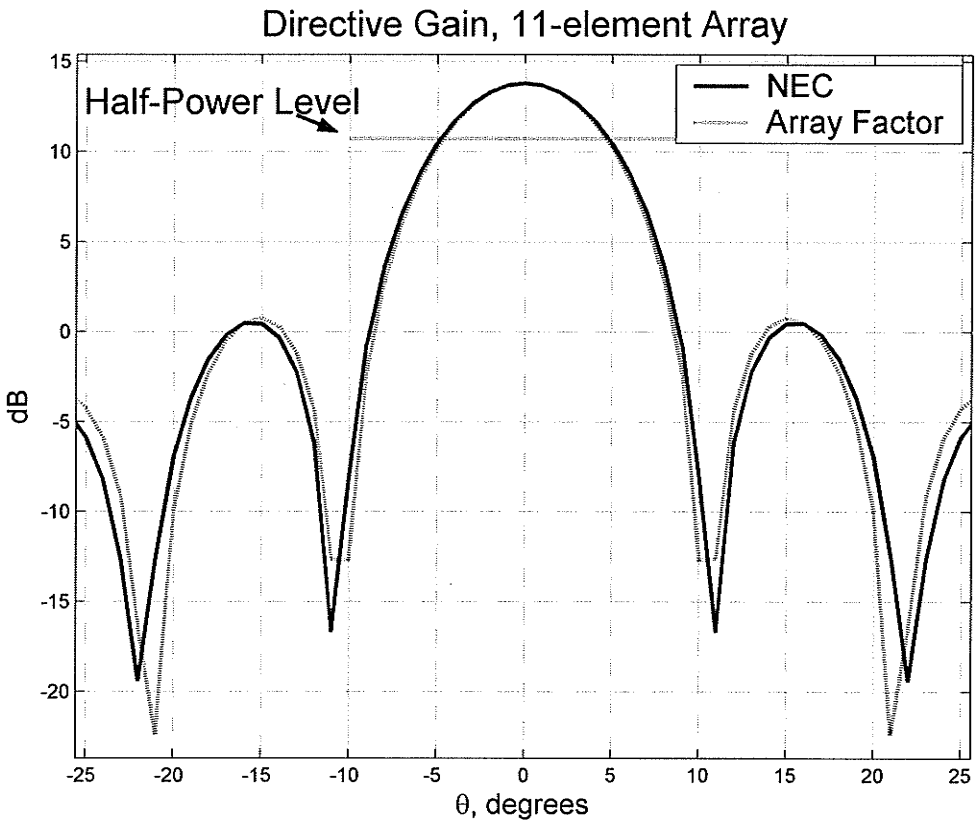
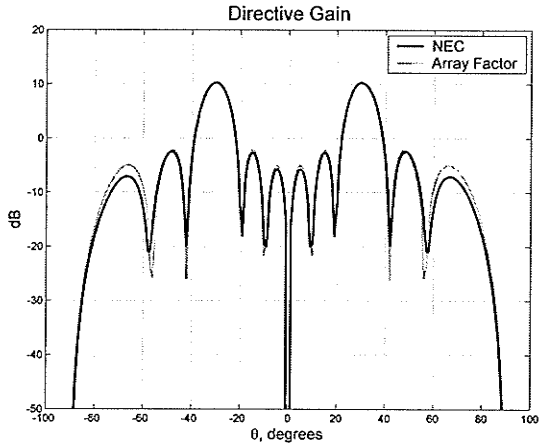


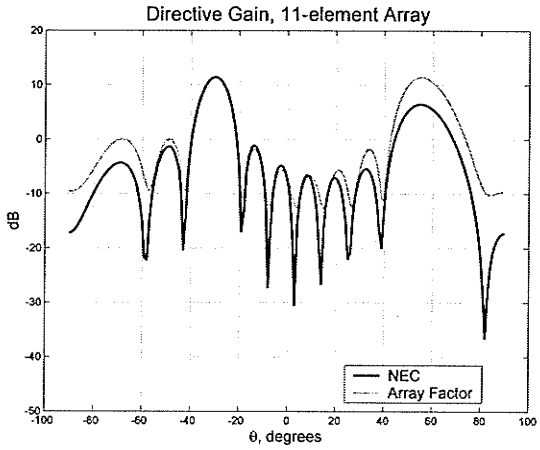
Figure 2.4– Zoom of plot comparing the output of NEC2 with the Array Factor for AoA = 0°.

Interesting to note is the rise of the coma lobe as shown in Figure 2.3(c) and (d), for scan angles of 40° and 60°. The coma lobe is the first sidelobe trailing the mainbeam scan and tends to rise with scan angle while the leading sidelobe tends to be suppressed. These effects are much more pronounced in the computed results of NEC than the array factor. This is because the effects of mutual coupling become greater for large scan angles, as will be shown in the next section where element impedance as a function of scan angle is presented.

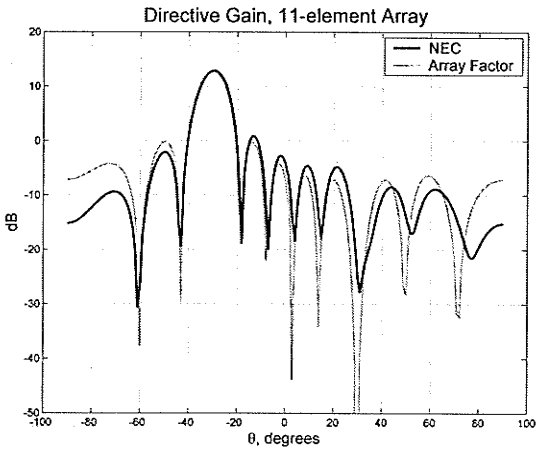
Now consider the application of a constraint matrix **C** as a beamformer, the theory for which is developed in the next chapter. By designating the constraint matrix to be a multi-column steering vector of electrical angles of interest, and by composing the gain vector to indicate whether each angle is a beam or a null, various beamformed array patterns can be created, as shown in Figure 2.5 for an 11-element array.



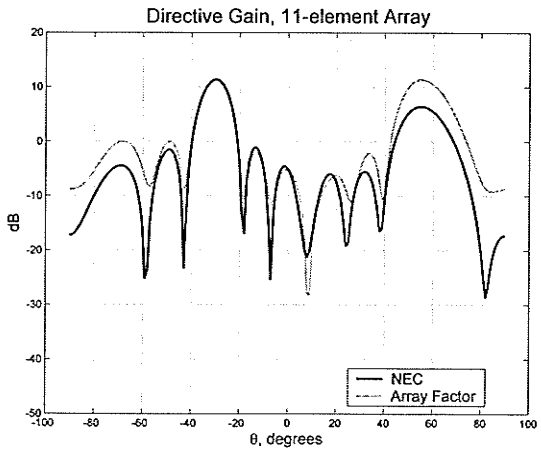
a) Two mainbeams, at -30° and 30°.



b) Two mainbeams, at -30° and 55°.



c) Mainbeam at -30° and null at 30°



d) Two mainbeams, at -30° and 55° with a null at 8.5°

Figure 2.5– Comparison of NEC2 with the Array Factor output for multiple-beam and constrained-null beamforms.

The full-wave results of NEC2, which take into consideration the inter-element mutual coupling, indicate very good agreement with the array factor at the mainbeams and constrained nulls. The output of NEC shows lower far sidelobes than anticipated by the array factor, which is consistent in both the scan tests of Figure 2.3 and the beamformed tests of Figure 2.5. Note in Figure 2.5, however, that the computed gain for the beams at 30° are consistent with the array factor, but the beam at 60° shows more drop in gain than indicated by the array factor, or as indicated by the computed NEC results in Figure 2.3(d).

The beamforming as shown in Figure 2.5 is an example of static, not adaptive, beamforming. This would be appropriate in a communication link with known fixed sources of interference, such as other transmitter towers. Adaptive interference techniques can be combined with fixed beamforming to suppress the effects from intermittent or mobile sources of interference.

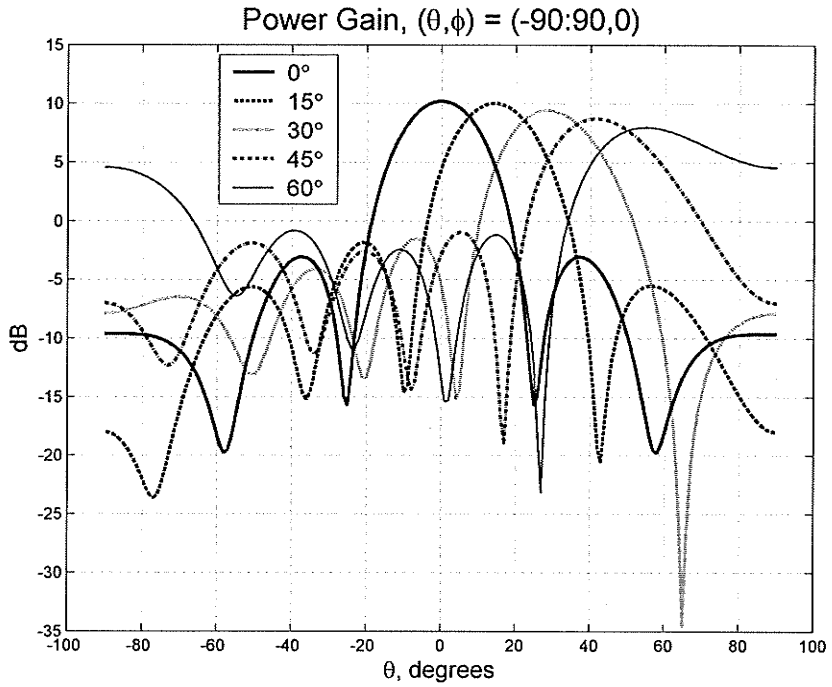
It is important to keep in mind that the computed results of NEC include the inter-element effects of mutual coupling, as seen in the radiation patterns, but don' t include the effects of impedance mismatch with the feed network due to impedance changes with scan angle, as will be demonstrated in the following section.

2.4 Application to Study Active Impedance of a Phased Array Antenna

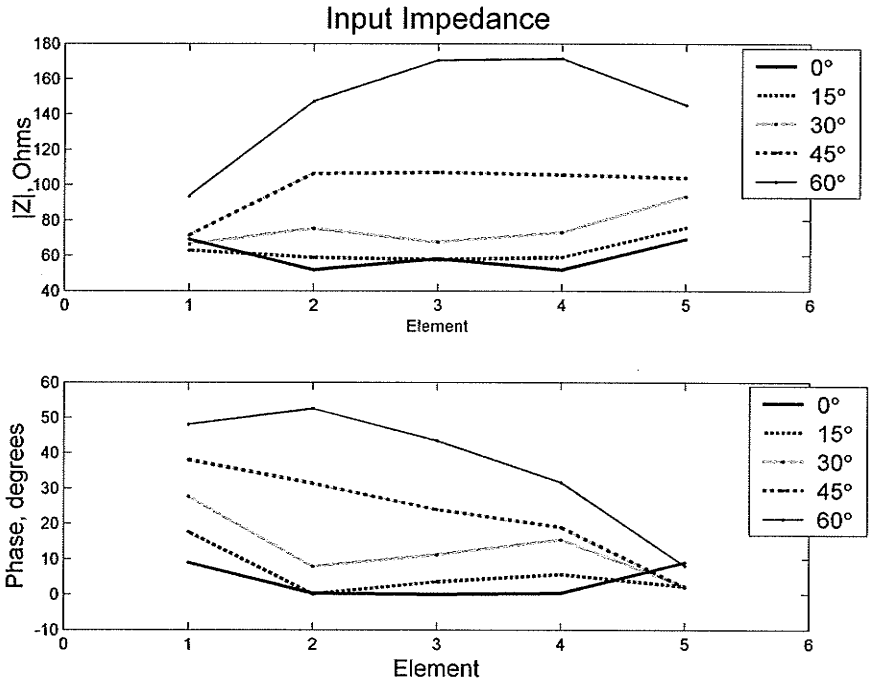
The Matlab-NEC2 interface can be used to study a wide variety of parameters of antennas, arrays, and any interaction with radiating or reflective structures. For example, the input impedance or the radiation pattern can be studied as a function of element length or spacing, or as a function of objects in the vicinity. Non-uniform variation of the element length and spacing in an array or Yagi-Uda design can be studied to optimize desired characteristics such as gain, wideband performance or efficiency using user-directed optimization strategies such as random or gradient searches or genetic algorithms.

In this section, the radiation pattern and input impedance is observed as a function of scan angle. Figure 2.6 shows the results for a 5-element array scanned from 0° to 60° in 15° increments. The radiation pattern shows the broadening of the beam and the reduction in gain as the scan angle increases. The impedance plot shows that the magnitude of the impedance is fairly constant for broadside, with an increase at the end elements. The phase plot shows a slightly non-uniform phase progression, but clearly indicates the tilt with scan.

As the scan angle increases, the magnitude of the impedance becomes increasingly non-uniform. The element furthest away from the scan angle generally has the lowest magnitude but largest phase angle, indicating the most reactive end.



a) Radiation pattern.



b) Impedance magnitude and phase.

Figure 2.6– Radiation pattern and impedance behavior as a function of scan angle for a 5-element array.

A half-wavelength dipole has to be shortened somewhat to attain a purely resistive impedance, which is approximately 73 ohms (Ω). From the impedance plots of Figure 2.6, it is shown that for scan angles from 0° to 30°, the impedance magnitude remains fairly consistent between 50 and 80 ohms, and apart from the furthest end element, the phase plot shows very little reactive component.

Typically, coaxial feedlines are either 50Ω (RG-8, RG-58, RG-213) or 72Ω (RG-11, RG-59). In order to deliver all of the power to the antenna, the feedline impedance must be matched to the element impedance. A mismatch results in a reflection of power, known as return loss, and is related to the reflection coefficient, Γ, given by

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} \tag{2.1}$$

where Z_L is the antenna, or load, impedance and Z_0 is the characteristic impedance of the feedline. When the impedances are matched, $Z_L = Z_o$, there is zero reflection of power and all of it is delivered to the load.

A measure of mismatch is defined as the standing wave ratio (SWR) and is given by

$$SWR = \frac{1+|\Gamma|}{1-|\Gamma|} \tag{2.2}$$

which we recognize is a real number between 1 and infinity, with $SWR = 1$ indicating a perfectly matched load at the resonant frequency.

Standard measure for good performance requires a SWR less than 2, which corresponds to a maximum reflection coefficient of $|\Gamma| = {}^1/_3$. Looking at the impedance plot,

considerable mismatch can be predicted at large scan angles. Using a 72-ohm line and (2.1), the magnitude of the reflection coefficient is presented in Table 2.3.

Table 2.3 - Magnitude of the reflection coefficient for 5-element scanned array.

$\lvert\Gamma\rvert$	**Antenna 1**	**Antenna 2**	**Antenna 3**	**Antenna 4**	**Antenna 5**
$\theta = 0°$	0.081427	0.16221	0.10552	0.16221	0.081427
$\theta = 15°$	0.16954	0.10009	0.11444	0.11021	0.032477
$\theta = 30°$	0.25032	0.072998	0.10283	0.13636	0.1316
$\theta = 45°$	0.34509	0.34065	0.28884	0.25334	0.18321
$\theta = 60°$	0.46444	0.59279	0.56196	0.49418	0.34465

With a rough guideline of $|\Gamma| < 0.33$, unacceptable levels of mismatch begin at the far-end elements for a scan angle of 45°, and all elements show poor matching at 60°. This type of impedance mismatch leads to a phenomenon known as *scan blindness*. At certain angles of scan, the mismatch is so bad that little or no power is delivered to the load in transmit mode, or in receive mode all of the incoming signal is reflected away from the array.

Table 2.3 gives the reflection coefficient as the ratio of reflected wave to the incident wave on the terminal, based on the impedance of the feed line and the element. The waves can be currents or voltages. Another standard measure is the return loss (*RL*), a power measure expressed in decibels based on the magnitude of the reflection coefficient as given by (2.3).

$$RL = 20*log_{10}(|\Gamma|), \text{ dB} \tag{2.3}$$

The dB value of the return loss, also known as the S_{11} scattering parameter in microwave network theory, ranges from 0 dB for total reflection of the incoming wave to -∞ dB for zero reflection and total transmission into a perfectly matched load (ideal). For a SWR ≤ 2, $|\Gamma| \leq 0.33$ such that the return loss $S_{11} \leq$ -9.54dB.

Because it is typical for a network analyzer to display the S_{11} return loss in dB over a range of frequencies, this analysis is applied to the 5-element array for the scan angles studied above. The point where S_{11} is minimum indicates the resonant frequency, with the

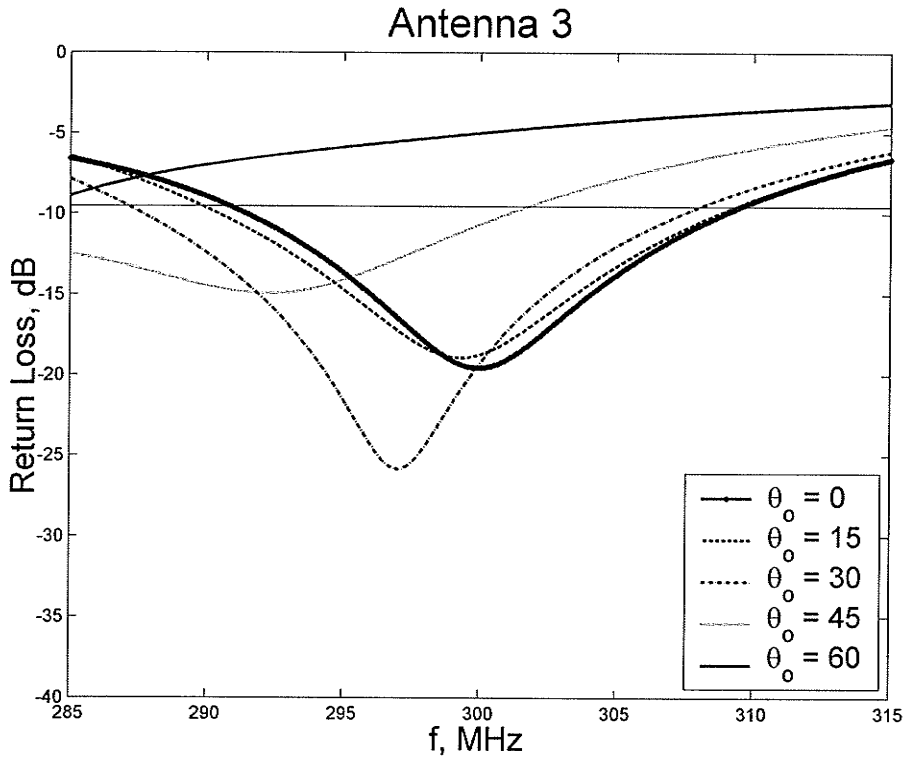


Figure 2.7– Return Loss of center element of 5-element array over 10% bandwidth. The horizontal line at -9.54dB corresponds to a SWR = 2.

ideal antenna perfectly matched to the feed network returning S_{11} = -∞ dB at resonance.

Figure 2.7 shows the S_{11} return loss for the center element as the frequency is scanned from 285MHz to 315MHz for each of the five scan angles. This represents a 10% bandwidth study for a center frequency of 299.79 MHz, chosen as the design frequency to normalize the free-space wavelength to 1 metre. The horizontal line at -9.54dB corresponds to a SWR = 2. Points below this line indicate good performance where SWR < 2.

Clearly, the impedance mismatch at a scan angle of 60° results in too much reflected power over the entire bandwidth indicating that this element will contribute to scan blindness at this angle. In fact, the array does not have a 10% bandwidth even for broadside radiation, as can be seen by the $\theta_0 = 0°$ curve. This curve crosses the SWR = 2 line at 290MHz and 310MHz, for an approximately 7% bandwidth.

In Figure 2.8, the return loss of the remaining elements shows that the array as a whole has an approximate 7% bandwidth at broadside but shifts and narrows with scan.

Note that for this study, the antenna elements are indexed from 1 to 5, with 3 being the center element and 1 and 5 being the end elements. Generally, either zero indexing (0, 1, …, M-1) or unit indexing (1, …, M) can be used as long as one element is assigned zero phase with respect to the radiation, and the other elements are phased with respect to this one according to the steering vector.

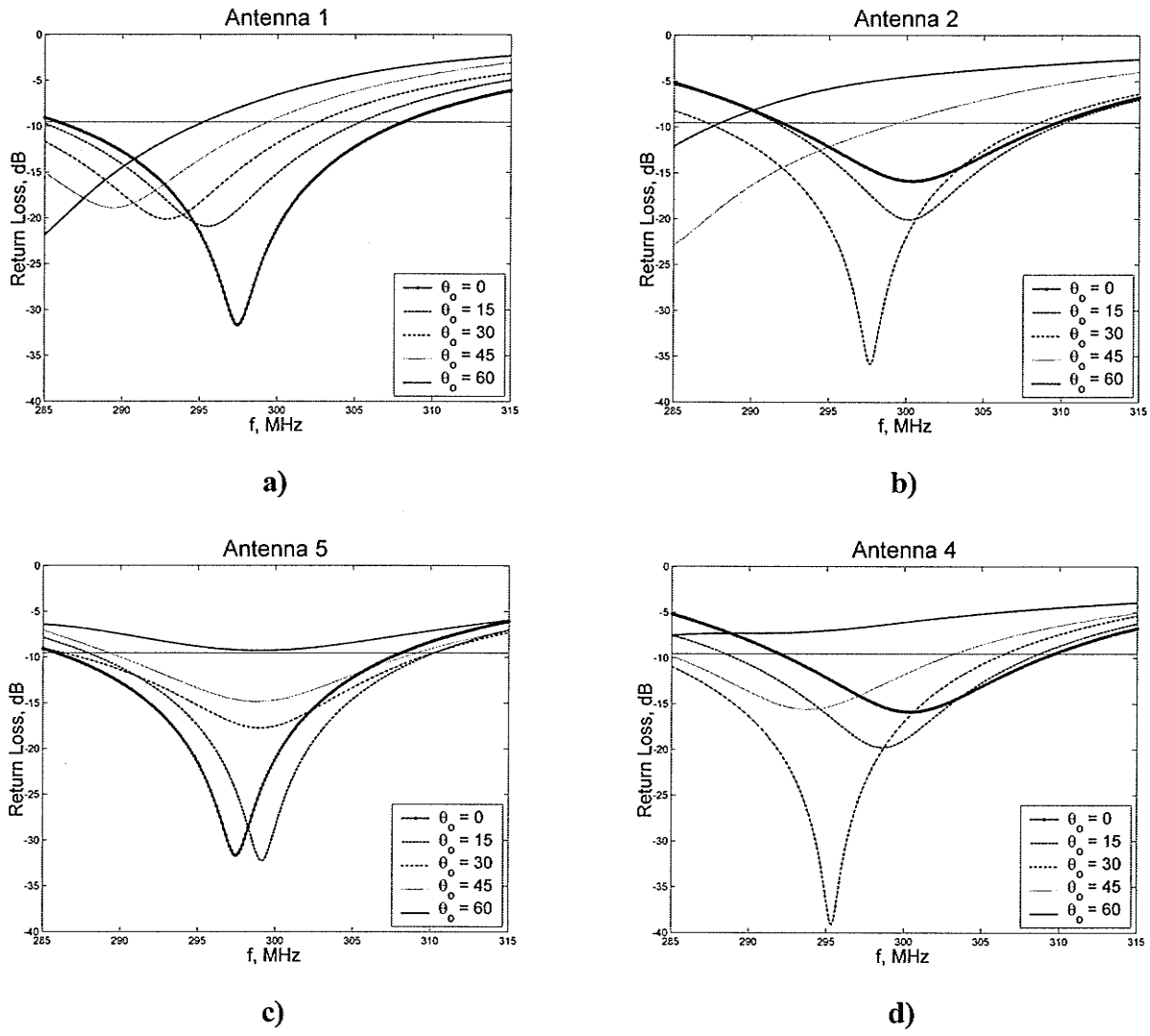


Figure 2.8– Return Loss of non-center elements of 5-element array over 10% bandwidth. The horizontal line at -9.54dB corresponds to a SWR = 2.

The plots of Figure 2.8 are arranged with antenna 1 above antenna 5 and antenna 2 above antenna 4 for comparison. By symmetry, the $\theta_o = 0°$ curves are identical for these pairs.

It is apparent that the array will have poor performance at a scan angle of 60° and potentially poor performance at 45°, particularly at the high end of the bandwidth.

It is interesting to note that except for antenna 5, at the end to which the beam is scanned, all elements shift the point of minimum return loss to a lower resonant frequency as the scan angle increases. Antenna 5 is the first antenna to receive the incoming radiation, or conversely is the last element to be directing energy from the array. In either case, there are no elements "upstream" of it to be reflecting or scattering radiation to induce mutual coupling. Only the elements downstream contribute. This may account for the more consistent shape of the curves with antenna 5, but raises the question as to why the resonant frequency shifts down with scan.

Figure 2.6(b), which shows how the impedance changes with scan angle, may offer some insight. Notice how the downstream elements become more inductively reactive with increased scan angle, as indicated by the increased phase shift for antennas 1 to 4. It is standard to inductively load antennas to increase the electrical length of an antenna that is physically short with respect to the wavelength. As the mutual coupling due to the scan angle increases the inductance of the antenna's impedance, the element appears electrically longer. A longer antenna is resonant at a lower frequency, and so the downward shift of the resonant frequency is observed with increased scan angle.

In summary of this section, it is important to note that the impedance and return loss behavior of this 5-element array have been studied but not optimized. The NEC2/Matlab interface developed in this work is well suited to the optimization task of minimizing return loss for a specified bandwidth, but remains for future work.

3.0 Adaptive Array Processing and Design Overview

This chapter develops the specific theory of adaptive array signal processing using adaptive filter theory by considering the antenna array as a spatial filter as opposed to a frequency filter.

There are many applications for adaptive signal processing broadly pertaining to system identification, inverse modeling of an unknown noisy process, best prediction of the present value of a random process based on previous values, and interference canceling. Hence, there is a wealth of information on adaptive filtering and signal processing from which a specific implementation of the antenna array as an adaptive spatial filter can be developed.

Regardless of which application is being studied, the parameters common to each are:

1. An applied input signal, $u(n)$, at the antenna terminals.
2. The output of the filter or array, $y(n)$.
3. A desired response, $d(n)$.
4. An estimation error, $e(n)$, where $e(n) = d(n) - y(n)$.

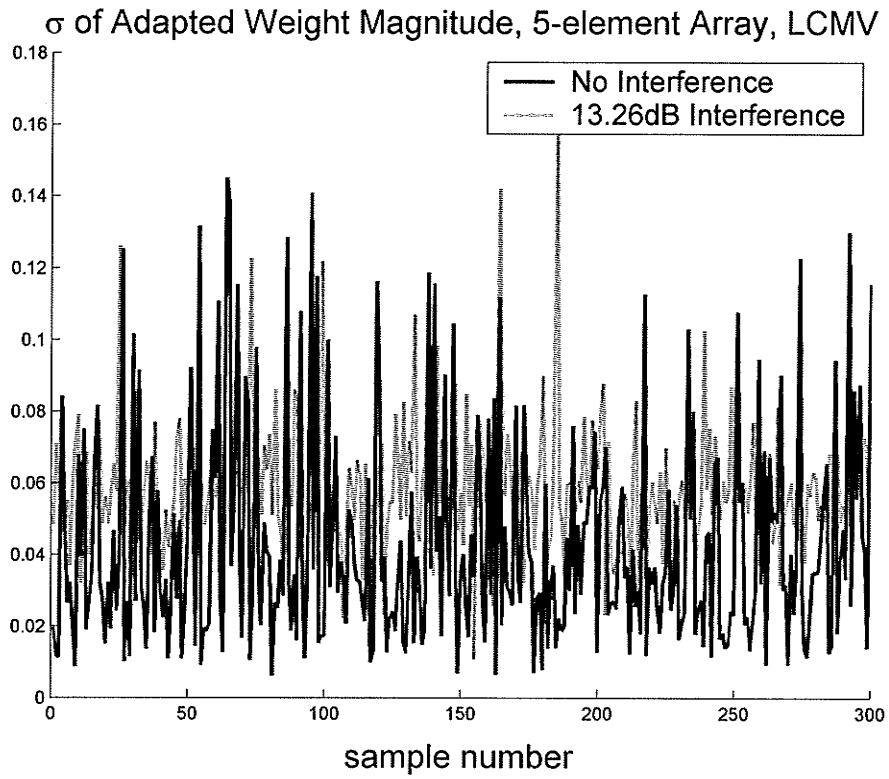
Given these basic parameters, the objective becomes an optimization of the output signal, $y(n)$, by a minimization of the estimation error, $e(n)$. Two specific methods of this will be presented.

A reasonable observation is that the desired signal is the transmitted signal, $x(n)$, before being corrupted by noise and loss as given in $u(n)$. This signal could be random and unknown a priori, such as with a communication message. On the other hand, in an application such as radar, the signal may already be known.

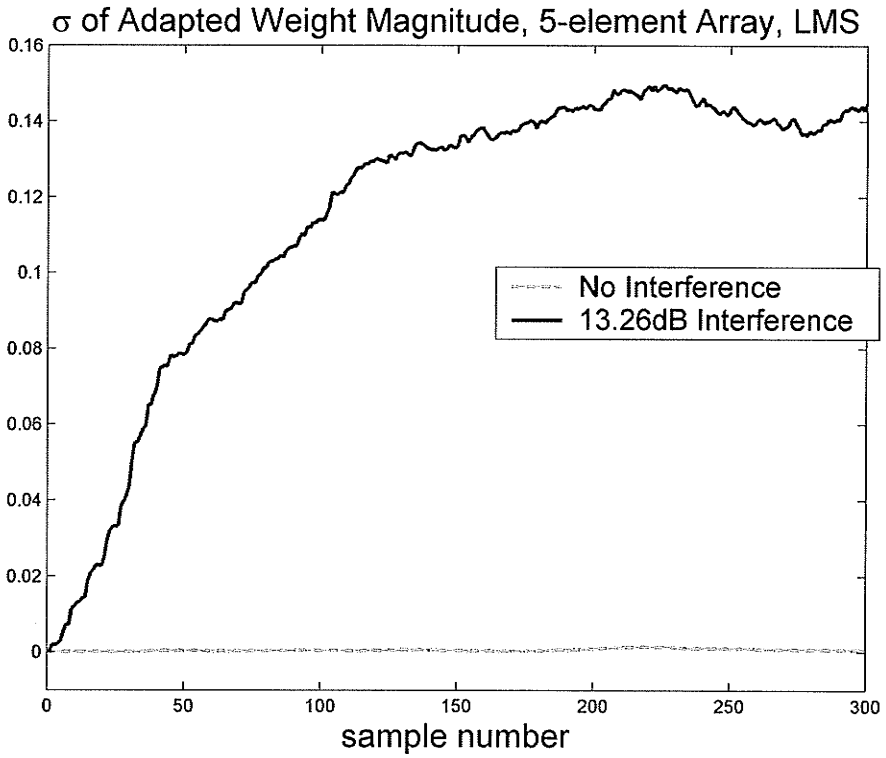
Generalized Sidelobe Cancellation (GSC) is just one specific method in the broad field of antenna array beamforming. As will be shown in this chapter, the complex weighting, $\mathbf{w}$, of the antenna elements is separated into a quiescent part, $\mathbf{w}_q$, and an adaptive part, $\mathbf{w}_a$, related by the signal-blocking matrix, $\mathbf{C}_a$, such that $\mathbf{w} = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a$, where $\mathbf{C}_a$ is the orthogonal complement to the constraint matrix $\mathbf{C}$. The quiescent portion of the weight, $\mathbf{w}_q$, forms the beam according to the underlying constraints given in $\mathbf{C}$ such as single or multiple main beam direction and fixed deep nulls, as was shown in Figure 2.5. The adaptive part, $\mathbf{w}_a$, works to cancel the unwanted effects of a changing noise environment by minimizing the output power of the array.

The noise environment is measured according to Signal-to-Noise Ratio (SNR) for the additive white Gaussian noise (AWGN) appearing at the antenna terminals due to thermal noise and the general noise background, assumed to be uncorrelated. The interference environment is due to unwanted coherent narrowband signals arriving at the array from a direction other than the mainbeam scan angle. This interference will be correlated along the array as it is just a delayed version of the same signal appearing at the antenna terminals. Interference is measured either as interference-to-noise ratio (INR) or interference-to-signal ratio (ISR). Given the SNR, either can be determined from the other. It is assumed that the signals are zero-mean, such that the signal power is the ac power given by the variance, σ^2.

In the case of no jammer, both the LCMV and the LMS push towards the uniform amplitude distribution described by $\mathbf{w}_q$, as shown by the lowered standard deviation of the weight vector amplitudes in Figure 3.1 for a 5-element array.



a) LCMV



b) LMS

Figure 3.1– Sidelobe canceler, standard deviation of weight vector over 300 iterations.

In this figure, 300 iterations, or time steps, are considered. At each iteration, the standard deviation of the magnitudes of the five weights is calculated. For the LMS algorithm, it is clearly seen to be the case, but for the LCMV algorithm, it tends to be the case. This raises some issues as to the weaknesses of the LCMV algorithm, which will be discussed later. Looking at the LMS results in Figure 3.1(b), however, note that in the absence of interference the magnitude of the weights remain virtually uniform for the entire 300 iterations, as indicated by the standard deviation remaining basically zero. In the presence of an interference source, it can be noted that the algorithm starts with the uniform weight distribution, but converges to a non-uniform distribution in about 150 iterations.

In the subsequent analyses, all arrays are composed of wire dipole antennas with the initial parameters as given in Table 3.1.

Table 3.1– Initial parameters common to tests.

Frequency, f	299.792458 MHz, for a free-space wavelength $\lambda = 1$m.
Element separation, d	half-wavelength, $\lambda/2$.
Dipole length, L	$\lambda/2 * 0.996$; half-wavelength dipole slightly shortened for optimization.
Wire radius, r	1 mm.
Number of elements, M	Number of antenna elements in the coplanar linear array.
Angle of Arrival, AoA	θ_o, scan angle of mainbeam to source direction (transmit or receive), degrees or radians.
Angle of Interference, AoI	θ_i, direction of arrival of interference, degrees or radians.
Spatial phase factor, ϕ	$\phi = kd\sin\theta = \pi\sin\theta$, since wavenumber $k = 2\pi/\lambda$ and $d = \lambda/2$.

As defined in Table 3.1, the spatial phase factor will be used to determine the signals at the antenna terminals due to both the source signal and the interference, via the corresponding steering vectors. Note that the source signal is the signal (transmit or receive) at the antenna terminals after having been corrupted with interference and noise, and is not to be confused with the original transmitted signal. In addition to these basic initial parameters, Table 3.2 lists the definitions of the signals and noise.

Table 3.2– Definitions of source and interference signals and terminal noise.

Original signal, *x(n)*	$x(n) = e^{j(\omega_u t - k\bar{r})}$, a complex signal of unit power, $\sigma_u^2 = 1$.
Interference signal, *m(n)*	$m(n) = \sqrt{\sigma_m^2}\, e^{j(\omega_m t - k\bar{r})}$, a complex signal of power σ_m^2.
Terminal noise, *v(n)*	$v(n) = \sqrt{\frac{\sigma_v^2}{2}}(v_r + j \cdot v_i)$, of power σ_v^2, where v_r and v_i are random values taken from a zero-mean, Gaussian normal distribution with a variance of one, i.e. $\sigma^2 = 1$. Also known as complex Additive White Gaussian Noise (AWGN).
Source signal, *u(n)*	$u(n) = x(n) + m(n) + v(n)$.
Steering vector, $s(\phi)$	$s(\phi) = [1, e^{-j\phi}, \ldots, e^{-j(M-1)\phi}]^T$, where T represents the transpose operator and this steering vector is for an end element being the reference, or 0^{th}, element.

In the definitions in Table 3.2, $\omega = 2\pi f$ is the angular frequency and k is the wavenumber. In general, $\omega_u = \omega_m$, and since the incoming signals are plane waves, the *exp(-jkr)* term appears in the simulations as the inter-element phase shift with respect to a reference element (usually an end or center element) and the spatial phase factor, ϕ, as it appears in the steering vector.

The following sections present the simulations of the LCMV and LMS algorithms for a 5-element array with no interference and one source of interference. The tests consider noise and interference environments of differing ISR and SNR, but for comparison of the two methods the same" random" noise will be applied to both the LCMV and the LMS methods. That is, the complex random noise is generated and saved, and then applied to each system so that a direct comparison of the two systems can be made for the same noise environment.

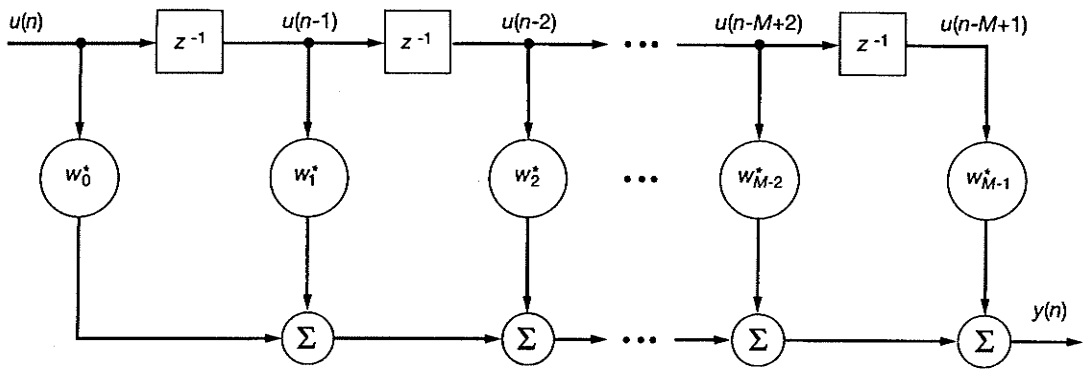


Figure 3.2– The Transversal Filter [1].

3.1 The Filter Model

Both the Linearly Constrained Minimum Variance (LCMV) method and the Least Mean Squares (LMS) method are applications of a linear optimum discrete-time filtering solution known as the Wiener filter, after Norbert Wiener [18]. The LCMV is a direct application of the Wiener optimal filter theory, while the LMS converges to the optimal Wiener solution in a stochastic gradient search method. The optimal estimate of filtering

a process corrupted by noise requires the solution of a set of integral equations, known as the Wiener-Hopf equations, which in discrete matrix form is given by

$$\mathbf{R}\mathbf{w}_o = \mathbf{p} \tag{3.1}$$

where **R** is the M x M correlation matrix of the tap inputs, $\mathbf{w}_o$ is the M x 1 vector of optimum weights and **p** is the M x 1 cross-correlation vector between the tap inputs and the desired response.

Consider the transversal filter model shown in Figure 3.2 for M stages, where $u(n)$ is a discrete-time digital signal and z^{-1} is the unit-delay operator, the complex weights, w_i, are conjugated as denoted by the asterisk, and the output of each weight is connected to an adder. The filter order is denoted by the number of delay elements, which is M-1. The transversal filter, also called a tapped delay-line filter, is a *finite-duration impulse response* (FIR) filter, as is evident by the lack of feedback from the output to an input. An FIR filter is inherently stable, in contrast to the *infinite-duration impulse response* (IIR) filter, which can become unstable in certain situations.

Because the adaptive process includes a form of feedback through the error signal, $e(n)$, an FIR filter is usually used to reduce complexity and stability issues. Although an IIR filter can be used, extra care must be taken in the implementation to ensure stability.

Within the class of FIR filters, three structures are predominantly applied to adaptive filters, the transversal filter, the lattice filter and the systolic array. Definitions of each of these filter structures can be found in [1]. Because of the direct analogy to the antenna array, this thesis focuses on the transversal filter implementation.

The output of the transversal filter shown in Figure 3.2 is given by the convolution sum in (3.2).

$$y(n) = \sum_{k=0}^{M-1} w_k^* u(n-k) \tag{3.2}$$

Consider the linear antenna array of M elements shown in Figure 3.3, illuminated by an incident plane wave, which when sampled is of the form

$$u(n) = e^{j(\omega_u n - k\bar{r})} \tag{3.3}$$

where $\omega = 2\pi f$ is the angular frequency, k is the wavenumber, and n would take on uniformly spaced instances in time given the sampling frequency, F_s.

Recall from the introductory chapter that given the element spacing, d, the path length difference is $d \sin\theta_o$ metres, which with $k = {}^{2\pi}/_{\lambda}$ radians per metre, translates into a phase delay of $\phi_o = kd \sin\theta_o$ between each element. The electrical angle, ϕ_o, is an inter-element phase delay and is analogous to the digital unit-delay operator, z^{-1}. The term " electrical angle" can be misleading, since although it is related to the spatial angle of arrival, it really defines the phase delay between adjacent elements due to that angle of arrival.

The output of the array, $y(n)$, is given by

$$y(n) = u_0(n) \sum_{k=0}^{M-1} w_k^* e^{-jk\phi_0} \tag{3.4}$$

where $u_o(n)$ is the signal received at time n by the end antenna element #0 in Figure 3.3.

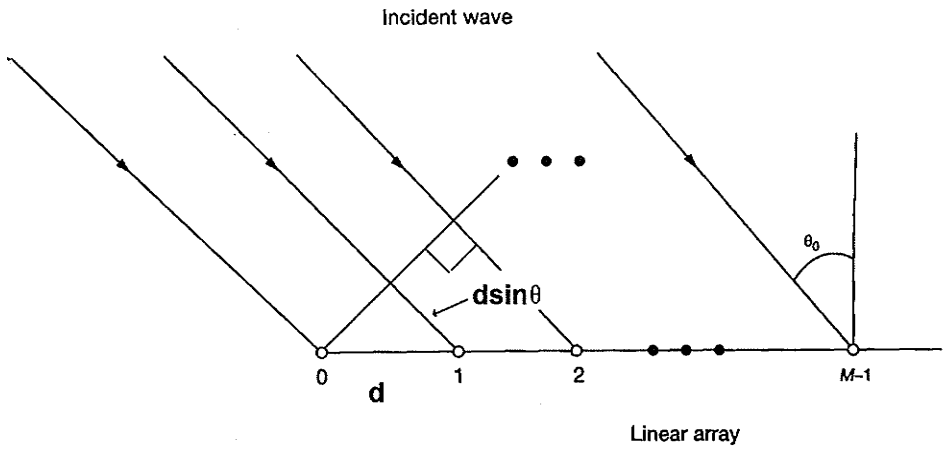


Figure 3.3– An incident plane wave arriving at a linear array at an angle of θ_o from normal to the array

Comparing Figure 3.3 with Figure 3.2, and (3.4) with (3.2), the direct analogy between the transversal filter and an array of antenna elements can clearly be seen for a sinusoidal excitation, *u(n)* and uniform antenna spacing, *d*.

The next two sections will present the specific methods with which the LCMV and LMS algorithms adaptively determine the complex weights, w_k, for a particular interference. In both cases, the formulations are dependent upon generating a constraint matrix, **C**, and its orthogonal complement, $\mathbf{C}_a$, which acts as the signal blocking matrix. This presentation is adapted from the detailed development given in [1] and forms the basis of the code development and analysis used in this thesis.

Looking at (3.4) and noting that for the specified scan angle ϕ_o, at a given time *n* the output *y(n)* can be interpreted as the input at the zeroth element, $u_o(n)$ multiplied by a complex scalar gain factor, g, given by

$$\sum_{k=0}^{M-1} w_k^* e^{-jk\phi_0} = g \tag{3.5}$$

On the left-hand side of this (3.5), one can recognize the vector inner-product of the tap-weight vector, **w**, and the steering vector, $\mathbf{s}(\phi_o)$ such that

$$\mathbf{w}_o^H \mathbf{s}(\phi_0) = g \tag{3.6}$$

In terms of an optimization process, the gain g is considered a constraint such that the output of the array is constrained to have gain g when pointed to ϕ_o regardless of the varying elements of the weight vector **w**. In general, (3.6) can be extended to include multiple linear constraints by introducing a constraint matrix, **C**, and a gain vector, **g**, given by

$$\mathbf{C}^H \mathbf{w} = \mathbf{g} \tag{3.7}$$

such that each column of the constraint matrix is the steering vector for the electrical angle at which to apply the constraint given at the corresponding element in the gain column vector **g**.

For example, the constrained weights to form the beamformer output shown in Figure 2.5(d) were obtained by setting up the following constraint equation

$$[\mathbf{s}(\phi_0)\ \ \mathbf{s}(\phi_1)\ \mathbf{s}(\phi_2)]^H \mathbf{w} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

where $\phi_i = (\phi_o, \phi_1, \phi_2)$ are the electrical angles for $\theta_i = (-30°, 8.5°, 55°)$ as defined in Table 3.1, $\mathbf{s}(\phi_i)$ is the steering column vector as defined in Table 3.2 and the gain vector constrains two mainbeams, at -30° and 55° with a null at 8.5°. It should be noted that the gain vector need not be only ones and zeros. A range of values will give more gain to the beam in the corresponding direction.

Also, for the example given in (3.8), **C** is obviously an M x 3 matrix, so can' tjust be inverted to determine **w**. Given L number of constraints, **C** is size M x L and this is where the M x (M-L) orthogonal complement matrix, $\mathbf{C}_a$, is required to form an M x M partitioned matrix **U** such that

$$\begin{aligned} &\mathbf{C}^H\mathbf{C}_a = \mathbf{C}_a^H\mathbf{C} = \mathbf{0}, \ \textit{and} \\ &\mathbf{U} = [\mathbf{C} \vdots \mathbf{C}_a] \end{aligned} \tag{3.9}$$

where **0** is a 0 matrix of size L x (M-L) or (M-L) x L. By defining the M x 1 weight vector in terms of **U**, such that $\mathbf{w} = \mathbf{U}\mathbf{q}$, or conversely $\mathbf{q} = \mathbf{U}^{-1}\mathbf{w}$, this vector **q** can be partitioned such that

$$\mathbf{q} = \begin{bmatrix} \mathbf{v} \\ \cdots \\ -\mathbf{w}_a \end{bmatrix} \tag{3.10}$$

where v is an L x 1 vector, and $\mathbf{w}_a$ is *(M-L)* x 1 and represents the part of the weight vector that can be used for adaptive weighting.

From $\mathbf{w} = \mathbf{U}\mathbf{q}$ we get

$$\mathbf{w} = \left[\mathbf{C} \vdots \mathbf{C}_a\right] \begin{bmatrix} \mathbf{v} \\ \cdots \\ -\mathbf{w}_a \end{bmatrix} = \mathbf{C}\mathbf{v} - \mathbf{C}_a\mathbf{w}_a \tag{3.11}$$

Using (3.7) yields

$$\mathbf{C}^H\mathbf{C}\mathbf{v} - \mathbf{C}^H\mathbf{C}_a\mathbf{w} = \mathbf{g} \tag{3.12}$$

But, by definition of the orthogonal complement, $\mathbf{C}^H\mathbf{C}_a = \mathbf{0}$ so that $\mathbf{C}^H\mathbf{C}\mathbf{v} = \mathbf{g}$, or subsequently, $\mathbf{v} = (\mathbf{C}^H\mathbf{C}\mathbf{v})^{-1}\mathbf{g}$. Now, looking at (3.11) where $\mathbf{w} = \mathbf{C}\mathbf{v} - \mathbf{C}_a\mathbf{w}_a$, a quiescent weight vector can be defined such that

$$\mathbf{w}_q = \mathbf{C}\mathbf{v} = (\mathbf{C}^H\mathbf{C})^{-1}\mathbf{g} \tag{3.13}$$

so that (3.11) becomes

$$\mathbf{w} = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a \tag{3.14}$$

Multiplying this weight vector by $\mathbf{C}^H$ as given in (3.7) results in

$$\mathbf{C}^H\mathbf{w}_q - \mathbf{C}^H\mathbf{C}_a\mathbf{w}_a = \mathbf{g}$$

or

$$\mathbf{C}^H\mathbf{w}_q = \mathbf{g} \tag{3.15}$$

since $\mathbf{C}^H\mathbf{C}_a = \mathbf{0}$. This shows that with respect to (3.7), it is the quiescent portion of the weight vector that satisfies the constraints. Indeed, the quiescent weight vector is determined by the constraints. In contrast, by (3.11) and (3.12) we see that the constraints in $\mathbf{C}$ do not affect the adaptive weight portion, $\mathbf{w}_a$.

Thus, to summarize, the Generalized Sidelobe Canceller (GSC) operates by minimizing sensitivity to interference while maintaining maximum sensitivity to a prescribed" look" direction, or scan angle. It does this by minimizing the output power while maintaining the constraints established in **C** with respect to the gain vector, **g**. This is intuitively sensible, as the output power of the array will be the sum of the desired signal received by a mainbeam and the interference received by a sidelobe region. If the constraints on the desired signal are being maintained, the minimization process will affect only the interference. The adaptive portion of the weight vector should have no effect on the constraints, as they have to be maintained. This is shown by (3.15). The quiescent weight vector, $\mathbf{w}_q$, is called such first because it is the fundamental weighting that forms the desired beam in a quiet environment without considering interference, and second because in a quiet environment with adaptive processing, the adapted weight vector will converge to the quiescent weight vector. That is, the adaptive portion, $\mathbf{w}_a$, will tend to zero.

By (3.4) and (3.6), and by noting that

$$\mathbf{u}(n) = u_o(n)\left[1, e^{-j\phi_o}, \ldots, e^{-j(M-1)\phi_o}\right] = u_o(n)\mathbf{s}(\phi_o) \tag{3.16}$$

where $\mathbf{u}(n)$ is the signal at the antenna terminals across the array, $u_o(n)$ is the signal arriving at the zeroth element and $\mathbf{s}(\phi_o)$ is the steering vector, the output of the array can be written in matrix form as

$$y(n) = \mathbf{w}^H\mathbf{u}(n) \tag{3.17}$$

which, by substitution of (3.14) becomes

$$y(n) = \mathbf{w}_q^H \mathbf{u}(n) - \mathbf{w}_a^H \mathbf{C}_a^H \mathbf{u}(n) \tag{3.18}$$

where it is expressed in terms of the quiescent and adaptive weight vectors. This leaves the specific methods for determining the optimum adaptive weight vector $\mathbf{w}_a$ to be established in the subsequent sections on the LCMV and the LMS algorithms.

Having established the constraint matrix $\mathbf{C}$ and its orthogonal complement $\mathbf{C}_a$, the essence of the filtering problem is to minimize the mean-square value of the estimation error, $e(n)$, where $e(n) = d(n) - y(n)$. In terms of the output, this is given as

$$y(n) = d(n) - e(n) \tag{3.19}$$

By considering the mean-square value of $e(n)$, the problem is formulated in terms of a second-order equation of the unknown weights and takes on a paraboloid shape with a unique minimum defining the optimal solution. Because the problem is formulated in terms of second-order statistics, *wide-sense stationary* stochastic signals are assumed.

Thus, the cost function to be minimized is given by

$$\begin{aligned} J &= E\left[e(n)e^*(n)\right] \\ &= E\left[|e(n)|^2\right] \end{aligned} \tag{3.20}$$

where E is the statistical expectation operator.

The result to this minimization problem is found by defining a complex gradient operator in terms of the first-order partial derivatives of the unknown complex weight coefficients, w_k. The specific details, as outlined in [1] and from which these derivations are drawn, result in

$$\nabla_k J = -2E[u(n-k)e^*(n)] \tag{3.21}$$

which is a minimum when this gradient is set equal to zero. Thus, letting e_o denote the estimation error when the determined weight vector is optimal, then

$$E[u(n-k)e_o^*(n)] = 0 \tag{3.22}$$

By using (3.2) and $e(n) = d(n) - y(n)$, (3.22) can be formulated as:

$$E\left[u(n-k)\left(d^*(n) - \sum_{i=0}^{\infty} w_{oi}u^*(n-i)\right)\right] = 0, \quad k = 0,1,2,\ldots \tag{3.23}$$

where w_{oi} is the i^{th} coefficient in the impulse response of the optimal filter.

Rearranging the terms in (3.23) yields

$$\sum_{i=0}^{\infty} w_{oi}E[u(n-k)u^*(n-i)] = E[u(n-k)d^*(n)] \tag{3.24}$$

Note that the expectation on the left-hand side of the equation is the autocorrelation function of the filter input for a lag of $i - k$, denoted as $r(i-k)$, and the expectation on the right-hand side of the equation is the cross-correlation of the filter input and the desired response for a lag of $-k$, denoted by $p(-k)$.

This results in an infinitely large system of simultaneous equations, known as the Wiener-Hopf equations, describing the optimal filter solution, and expressed as

$$\sum_{i=0}^{\infty} w_{oi}r(i-k) = p(-k) \tag{3.25}$$

In matrix form for an *M*-tap filter, the Wiener-Hopf equations are expressed in terms of **R**, an *M* x *M* correlation matrix of the tap inputs, **u**(n), the *M* x *1* tap input vector, and **p**, the *M* x *1* cross-correlation vector **u**(n) and d(n), and are expressed in compact matrix form as

$$\begin{aligned} &\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}^H(n)] \\ &\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)] \\ &\mathbf{p} = E[\mathbf{u}(n)d^*(n)], \quad and \\ &\mathbf{R}\mathbf{w}_o = \mathbf{p} \end{aligned} \tag{3.26}$$

where the boldface represents a matrix or vector, and the non-boldface represents a scalar.

Solving the matrix equation for the unknown optimal weight vector is obtained by multiplying both sides of the equation by the inverse of the correlation matrix, giving

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{3.27}$$

Thus, determination of the optimum weight vector reduces to requiring the knowledge of two quantities, the autocorrelation of the tap inputs, and the cross-correlation of the tap inputs with the desired signal. This, combined with basic principle that the output is used to provide an estimate of a desired response, is the essence of Wiener linear optimal filtering. As stated in the introduction to this chapter, the best desired signal would be the transmitted signal *x(n)* itself, before being corrupted by noise, interference and loss like how it arrives at the array as *u(n)*. This is reasonable in an application such as channel modeling, where both the transmit and receive ends know the transmitted signal and the received signal is used to determine the channel characteristics. However, everyone

would agree that a wireless communication system would be unnecessary if the transmitted signal was already known. In this situation, it is reasonable to say that the desired signal is the received signal without the presence of interference.

From the point of view $\mathbf{w}_o = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_{ao}$ as expressed in (3.14), and the formulation for the output, $y(n) = \mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H\mathbf{C}_a^H\mathbf{u}(n)$, from (3.18), the optimal Wiener linear filtering problem must be put into the form of (3.19), namely $y(n) = d(n) - e(n)$, where the error $e(n)$ is phrased in terms of the variable of optimization, $\mathbf{w}_a$. The next two sections present the LCMV and the LMS implementations of this concept. Both the LCMV and the LMS methods were implemented in Matlab code based on the above formulations, and it was verified that for a signal with no noise, the output, $y(n)$, equals the input, $u(n)$, identically.

3.2 The Linearly Constrained Minimum Variance (LCMV) Method

The LCMV method is a direct application of the Wiener linear optimal filter theory as expressed in (3.26) and (3.27), with the minimization of error obtained by the minimization of output power.

To rewrite (3.18), the output of the array is

$$y(n) = \mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H\mathbf{C}_a^H\mathbf{u}(n) \tag{3.28}$$

By defining

$$\begin{aligned} d(n) &= \mathbf{w}_q^H\mathbf{u}(n) \\ \mathbf{x}(n) &= \mathbf{C}_a^H\mathbf{u}(n) \end{aligned} \tag{3.29}$$

The standard Wiener filter form becomes

$$y(n) = d(n) - \mathbf{w}_a^H \mathbf{x}(n) \tag{3.30}$$

where optimization of $\mathbf{w}_a$ can be performed over the data space of $\mathbf{x}(n)$ instead of $\mathbf{u}(n)$. Thus (3.27) is with respect to $\mathbf{x}(n)$ but is linked to $\mathbf{u}(n)$ through (3.29).

The optimization problem becomes a minimization of the variance of $y(n)$, assumed zero-mean, as given by

$$\min_{\mathbf{w}_a} E[|\, y(n)\,|^2] = \min_{\mathbf{w}_a}(\sigma_d^2 - \mathbf{p}_x^H \mathbf{w}_a - \mathbf{w}_a^H \mathbf{p}_x + \mathbf{w}_a^H \mathbf{R}_x w_a) \tag{3.31}$$

Using the definitions of autocorrelation, crosscorrelation and (3.29), the optimum solution of $\mathbf{w}_{ao}$ is determined as given in (3.32), where it is shown to be a function of the quiescent weight vector, the autocorrelation of the antenna terminal inputs and the orthogonal complement of the constraint matrix.

$$\mathbf{w}_{ao} = \left(\mathbf{C}_a^H \mathbf{R} \mathbf{C}_a\right)^{-1} \mathbf{C}_a^H \mathbf{R} \mathbf{w}_q \tag{3.32}$$

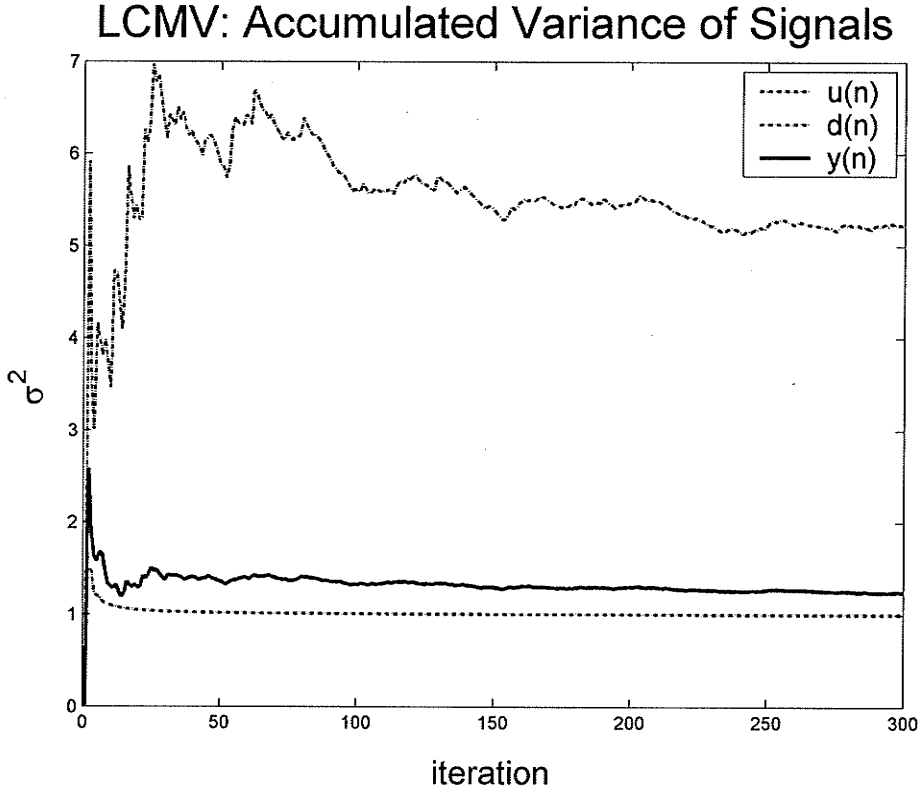
Because the solution contains the correlation matrix $\mathbf{R}$ in the denominator with $\mathbf{C}_a$, the LCMV requires there to be some noise in order to avoid a singularity. The reason for this can be seen by considering (3.16) and (3.26), with respect to $\mathbf{u}(n)$, $\mathbf{R}$, and the orthogonal relationship of $\mathbf{C}$ and $\mathbf{C}_a$.

In particular, if there is no random noise component, then $\mathbf{u}(n) = u_o(n)\mathbf{s}(\theta_o)$. Since, by definition, $\mathbf{s}(\theta_o)$ is orthogonal to $\mathbf{C}_a$, without sufficient random noise the denominator in (3.32) will be, or tend to, zero.

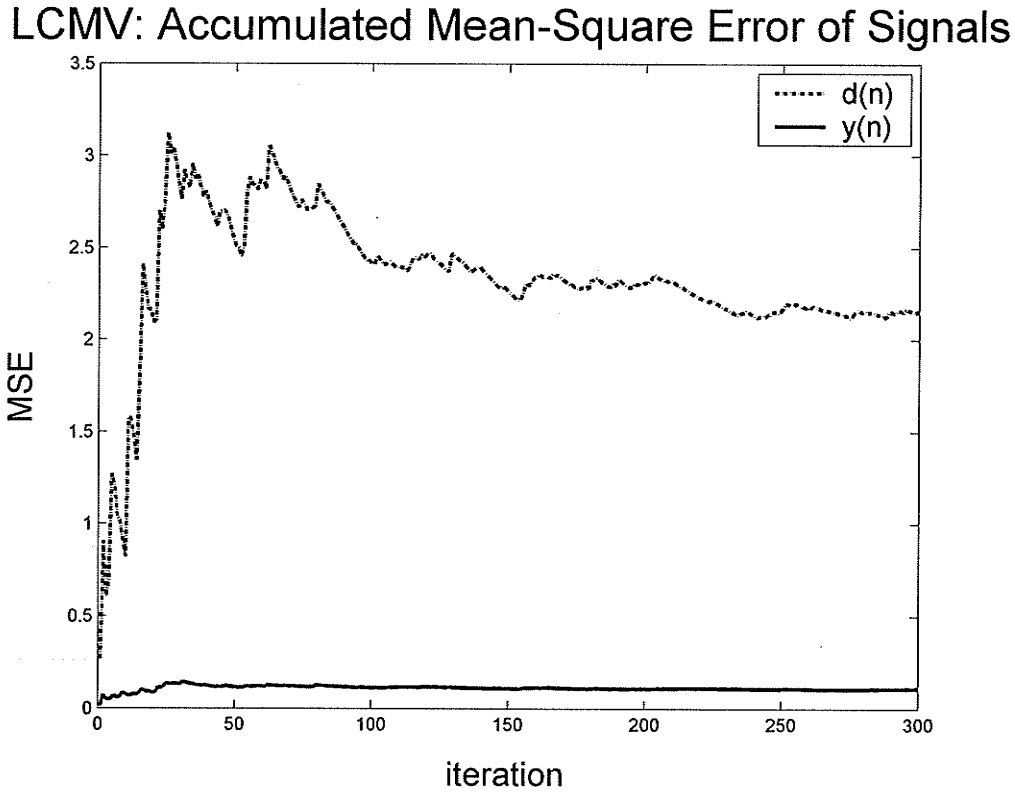
An implementation of this method is shown in Figure 3.4 for a 5-element array scanned to -11.54° with a SNR of 10dB and an ISR of 10dB incident at 0°. The figure shows the accumulated variance of the signal and the accumulated mean-square error for 300 iterations, or time steps. With a sample frequency of almost 900MHz, three times the carrier frequency, 300 iterations are about $1/3^{rd}$ of a microsecond. A start-up transient is evident because the data is accumulated, meaning the statistic is calculated as the vector grows.

There are several ways to consider the performance of an adaptive algorithm. The adapted output of the array, $y(n)$, and the desired output of the array, $d(n)$, can be compared to the original noiseless signal, $x(n)$, by comparing the variance of the signals and by comparing the mean-squared error of $y(n)$ and $d(n)$ with respect to $x(n)$. Recall that $x(n)$ is normalized with a variance of one, $d(n)$ is output of the array using only the quiescent weights, and $y(n)$ is the output of the array adapted to the noise and interference environment.

Looking at Figure 3.4, two things are immediately observable. First, from the lowered variance and mean-squared error of the adapted output compared to the output of the quiescent weights alone, the algorithm is working very well. Secondly, the LCMV is working well right away – there is no iteration to a good solution. This is because the LCMV calculates the best weights instantaneously at each sample, based on the signals at the array elements at that instant in time.



a) Variance.



b) Mean squared error.

Figure 3.4– Variance and Mean-Squared Error for LCMV 5-element array, AoA = -11.537°, AoI = 0°.

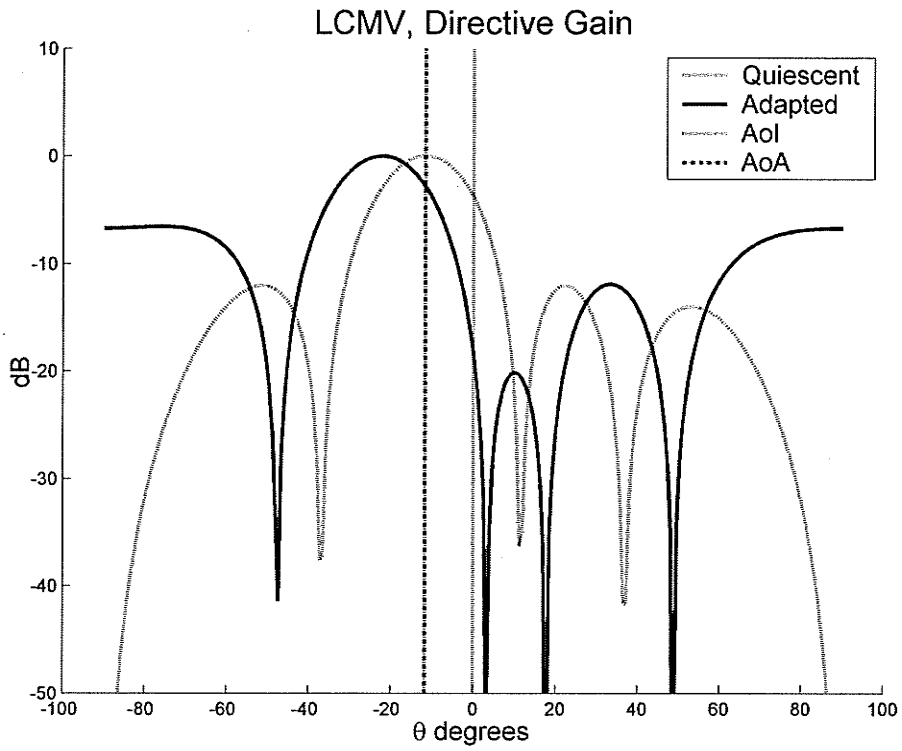


Figure 3.5– LCMV adaptive nulling, AoA = -11.53°, AoI = 0°, SNR = 10dB, ISR = 10dB for an INR = 20dB.

Another obvious way to view the performance of the adaptive algorithm is by looking at the far-field radiation patterns for the quiescent and adapted weights using either the array factor, NEC or both. As is seen in Figure 3.5 for the array factor, the interference is arriving in the mainbeam, and the LCMV method has shifted the main beam over by almost 11°. This has had the effect of lowering the gain in the AoA " look" direction by 2.7dB but with an improvement in nulling out the interference by 14dB, lowering the gain at the AoI from -3.46dB to– 17.8dB.

Consider a source of interference that is not in the mainbeam, but rather a sidelobe. The radiation pattern is shown in Figure 3.6 for ISR = 11dB, SNR = 10dB, an AoA = -11.53° and an AoI = -53°.

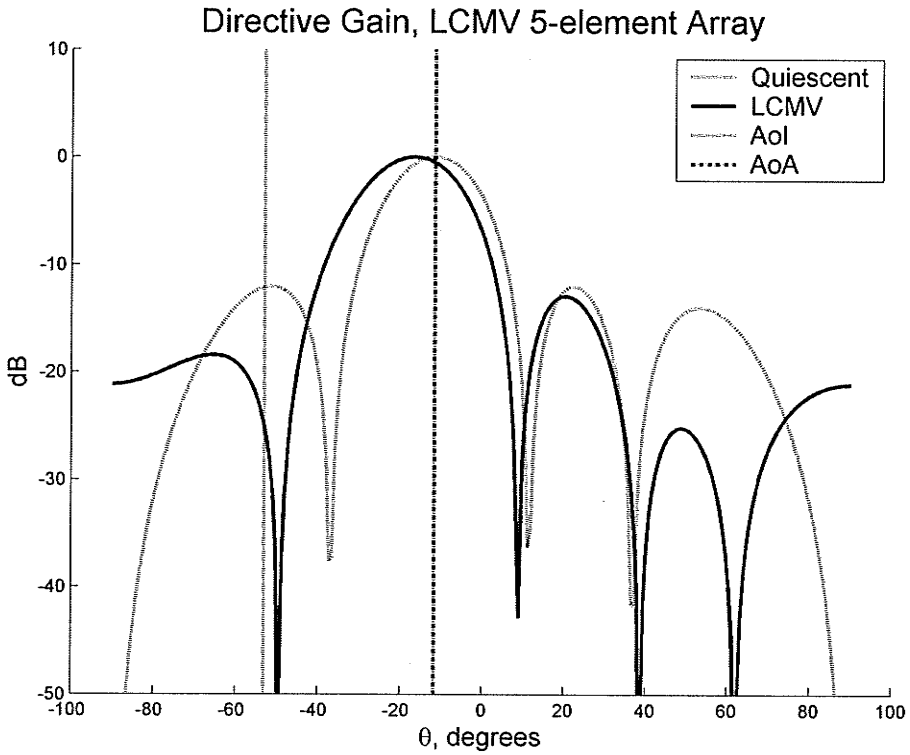
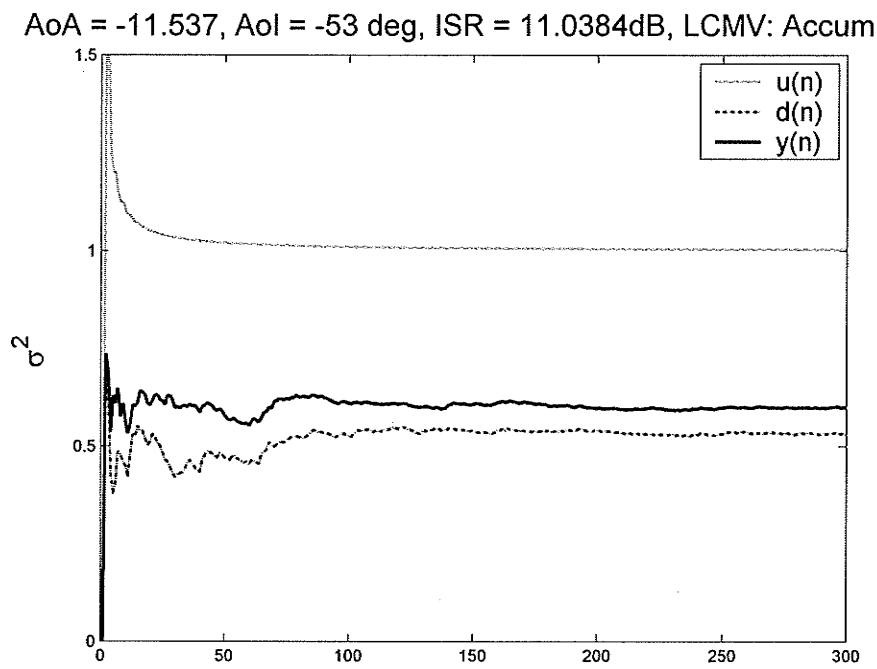
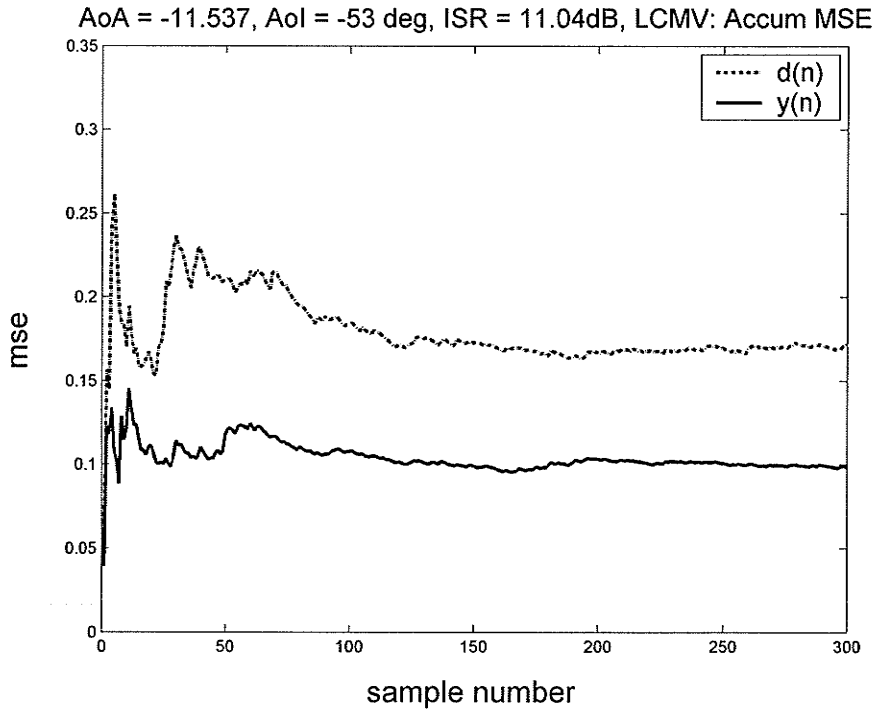


Figure 3.6– Gain Pattern, LCMV, AoA = -11.53°, AoI = -53°, ISR = 11dB SNR = 10dB.

Here, less displacement of the mainbeam is observed, but with a slight widening of 3-dB beamwidth. The main beam is shifted less than 5° with a corresponding drop in gain of less than 0.5dB. At the AoI, the gain has dropped 13dB from – 12.1dB to –25.1dB, which represents approximately $^{1}/_{20}^{th}$ of the previously received interference signal strength. The corresponding variance and error performance are shown in Figure 3.7.



a) Variance



b) Mean-squared error

Figure 3.7– Variance and mean-squared error, LCMV, AoA = -11.53°, AoI = -53°, ISR = 11dB, SNR = 10dB.

In the above simulations, the interference was a delayed version of *u(n)*, modulated by random noise as given by (3.33). Note in Figure 3.7 that the variance of the output is less than that of the input signal, but the variance of *d(n)* is still" farther away" from that of *x(n)* in an absolute sense. The power spectral density plot of the above noise source, shown in Figure 3.8, indicates it is a noisy narrowband signal at the carrier frequency of 300MHz, the same as *u(n)*.

$$j(n) = \sqrt{\sigma_j^2}\left(1 + {}^{v_j}\!/_{2}\right)u(n)\,e^{j\frac{3\pi}{5}} \tag{3.33}$$

where v_j is a 0-mean, unit variance normal random variable.

As will be shown in more detail in the next section, the representation of the interference, whether AWGN, pure sinusoidal or noise-modulated sinusoidal, has an effect on the behavior and performance of both the LCMV and the LMS methods.

Seeing an output variance less than the input, as in Figure 3.7, is at first counter-intuitive because we would think that the presence of any interference at a sidelobe would increase the output power of the array. This, however, is a result of destructive cancellation with the interference signal that is at the same carrier frequency as *u(n)* but phase shifted by ${}^{3\pi}\!/_{5}$ radians. As can be seen in Figure 3.7(b), the error is still less in the adapted output.

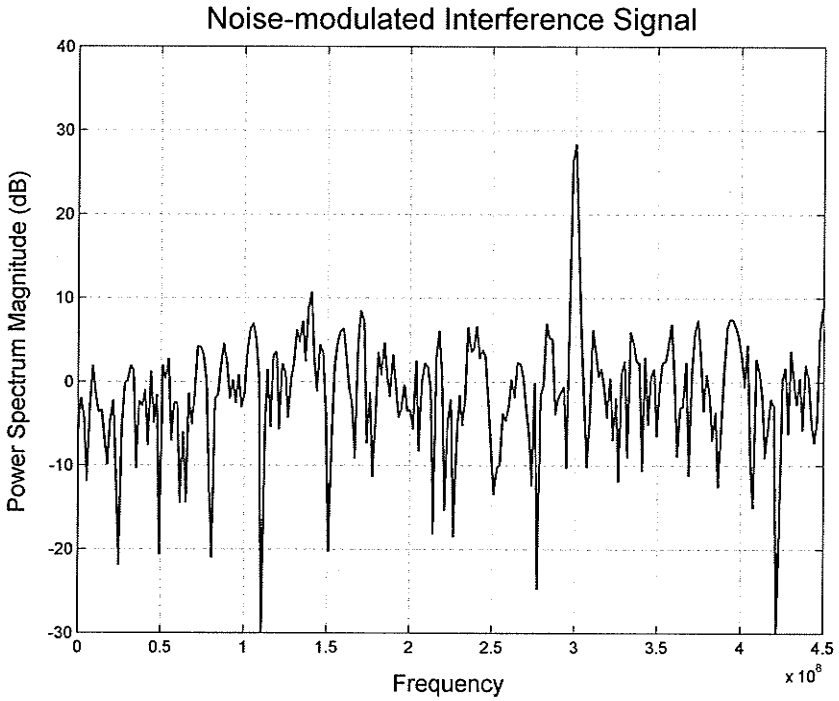


Figure 3.8– Power spectral density of sinusoidal interference amplitude modulated with noise.

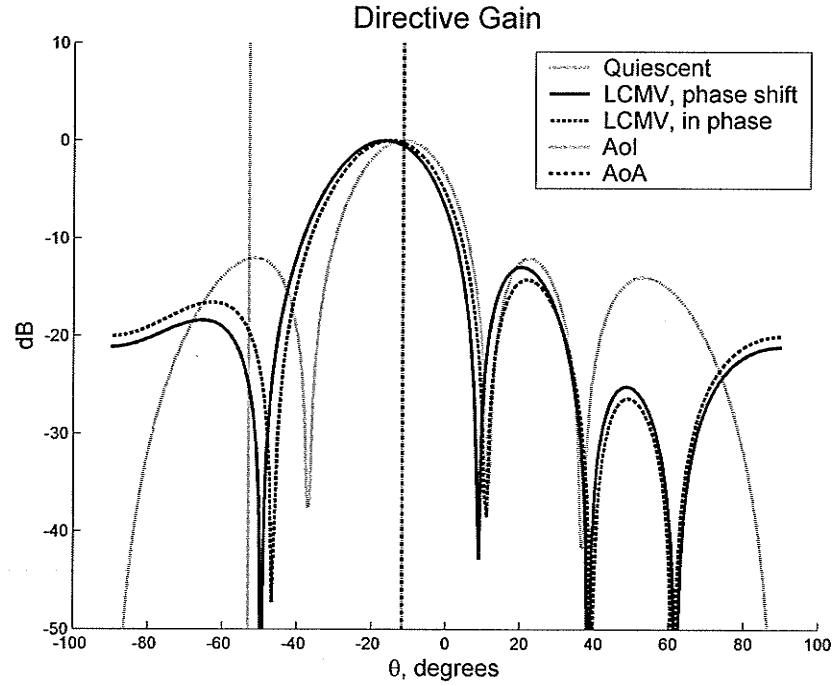
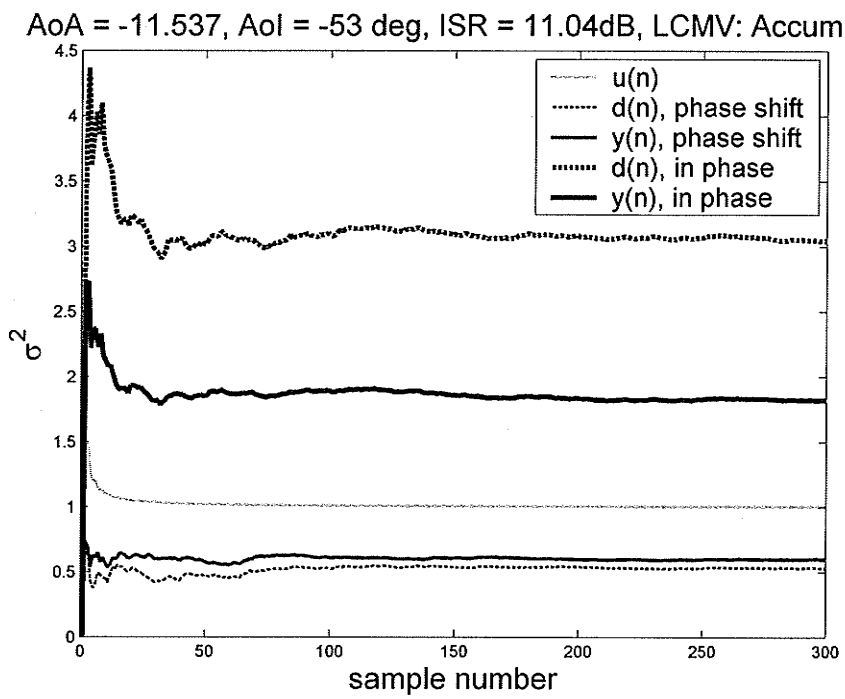


Figure 3.9– Directive gain comparison of noise amplitude-modulated interference in-phase and phase-shifted with *u(n)* at the carrier frequency.

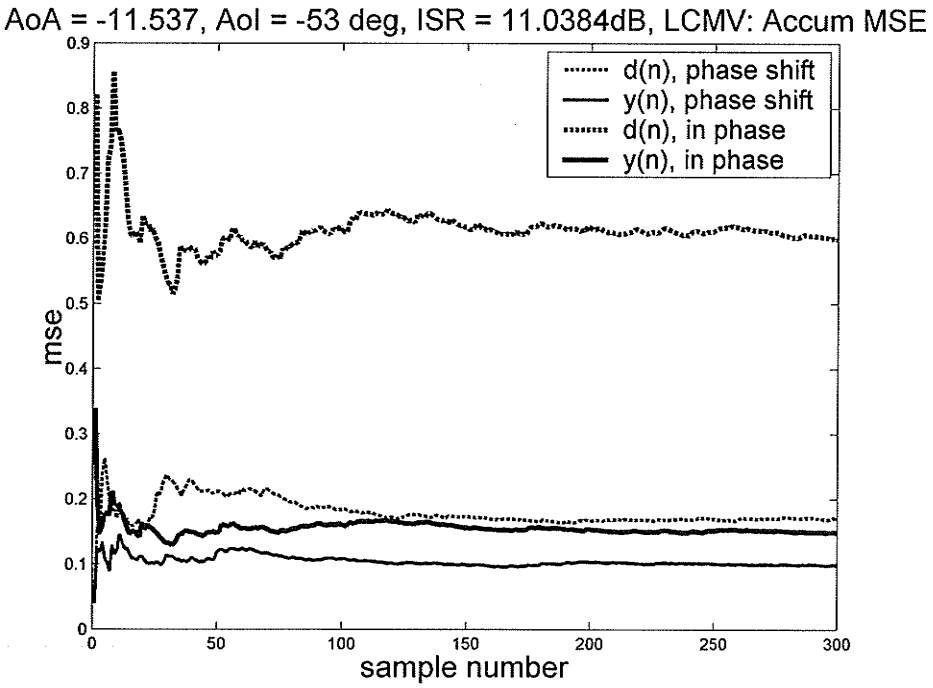
Figure 3.9 shows the array factor results of the LCMV method for both the in-phase and phase-shifted amplitude modulated noise interference. Note that performance differs, but is good in both cases.

To further illustrate this, Figure 3.10 compares the data of Figure 3.7 with an interference signal that is in phase with *u(n)*. In both simulations, for consistency, exactly the same terminal and modulating noise was used. The only difference is the phase shift.

To summarize this section on the LCMV method, the first observation is that the algorithm calculates the best weights based on the statistics of the current sample instant across the array. The second observation is that modulated noise interference signals have results that vary slightly depending on the phase of the interference with respect to the desired signal, but both show good performance.



a) Variance



b) MSE

Figure 3.10– Accumulated variance and mean-squared error comparison of noise amplitude-modulated interference in-phase and phase-shifted with *u(n)* at the carrier frequency.

3.3 The Method of Least-Mean Squares (LMS)

The formulation for the LMS method is different for the LCMV, where the weights were determined by the correlations of the instantaneous signals at the array antenna terminals. Still, the LMS method is based on the ideas of the Wiener linear optimum filter, but converges on the solution by means of a stochastic gradient search. The object is still to estimate the gradient vector of the quadratic error surface as given in (3.21) and to find the minimum.

The error surface, given in (3.31) as the minimization of the output signal power, can be expressed as

$$J(n) = \sigma_d^2 - \mathbf{w}^H(n)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) \qquad \textbf{(3.34)}$$

A *steepest descent* algorithm takes an initial value weight vector, $\mathbf{u}(n)$, and computes the complex gradient vector of $J(n)$, denoted by $\nabla J(n)$. Unless there is some *a priori* knowledge of $\mathbf{u}(n)$, the initial guess is usually the null vector. The next guess for $\mathbf{u}(n)$ is calculated by making a *step-size* change in the direction opposite to that of the gradient vector. Intuitively, by moving in the direction of the negative to the gradient, *which is the direction of steepest descent*, the algorithm will arrive at the minimum error at the optimum value $\mathbf{w}_o$.

The steepest descent algorithm is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \tfrac{1}{2}\mu[-\nabla J(n)] \qquad \textbf{(3.35)}$$

Looking at (3.34) and differentiating with respect to **w**, an estimation of the gradient of the error surface is

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n) \tag{3.36}$$

but since **R** and **p** are unknown for the true signal, they must be estimated from the instantaneous values.

An estimate of the instantaneous values of since **R** and **p** are denoted with a caret, as given in (3.37) and (3.38).

$$\hat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n) \tag{3.37}$$

$$\hat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n) \tag{3.38}$$

Thus the corresponding instantaneous estimate of the gradient vector is:

$$\hat{\nabla} J(n) = -2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n) \tag{3.39}$$

which is a biased estimate, since it depends on the random vector □(*n*) (the tap weight estimate vector) which in turn depends on the tap input vector, **u***(n)*. Now (3.39) can be substituted into the steepest descent algorithm in (3.35) to obtain

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)] \tag{3.40}$$

where the hat symbol indicates that the resultant weight vector is not the ideal weight vector that would be returned by (3.35). The significant feature of the LMS method is

that the algorithm uses the product $e_k^*\, u(n-k)$ to estimate $\nabla J(n)$ for the *kth* element. A true steepest descent algorithm would use the expectation, $E\{e_k^*\, u(n-k)\}$ to estimate $\nabla J(n)$. This reduces the number of computations, making the algorithm more efficient, but has the detrimental effect of introducing *gradient noise*.

We can now write the algorithm in terms of its input and output:

1. Filter output:

$$y(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \tag{3.41}$$

2. Estimation error:

$$e(n) = d(n) - y(n) \tag{3.42}$$

3. Tap-weight adaptation:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*n \tag{3.43}$$

Now, consider the definitions for $d(n)$ and $\mathbf{x}(n)$, given in (3.29) and repeated here in (3.44) for convenience, but instead let the error signal $e(n)$ be as given in (3.45).

$$\begin{aligned} d(n) &= \mathbf{w}_q^H\mathbf{u}(n) \\ \mathbf{x}(n) &= \mathbf{C}_a^H\mathbf{u}(n) \end{aligned} \tag{3.44}$$

$$\begin{aligned} e(n) &= \mathbf{w}^H\mathbf{u}(n) \\ &= \mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H\mathbf{C}_a^H\mathbf{u}(n) \end{aligned} \tag{3.45}$$

Note that in (3.45), the definition for **w** is the same as that used for the LCMV and given in (3.14), only now *e(n)* is defined as *y(n)* was for the LCMV in (3.18). This relationship is depicted in the signal flow chart block diagram of Figure 3.11 [1].

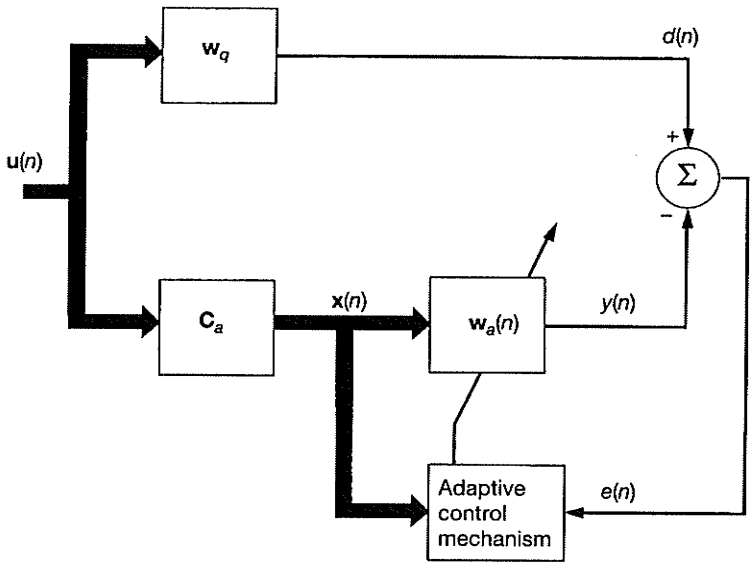


Figure 3.11– Block diagram of LMS implementation of the GSC [1].

The block diagram of Figure 3.11 shows the feedback path inherent in the LMS adaptive algorithm. Because the LMS algorithm utilizes feedback, stability must be ensured by requiring that $J(n) \rightarrow J(\infty)$ as $n \rightarrow \infty$. If the algorithm satisfies this condition, it is said to be convergent in the mean square. In order to satisfy this condition, the step-size, μ, has to satisfy certain conditions related to the correlation matrix of the inputs. If a small value of μ is used, the adaptive process progresses slowly and the effects of gradient noise are reduced.

Now the LMS algorithm for the calculation of the adaptive weights can be derived from the block diagram and based on (3.41) to (3.43), but with a change of notation of *e(n)* for *y(n)* in (3.41), $\mathbf{w}_a$ for □ and **x***(n)* for **u***(n)* in (3.43), giving

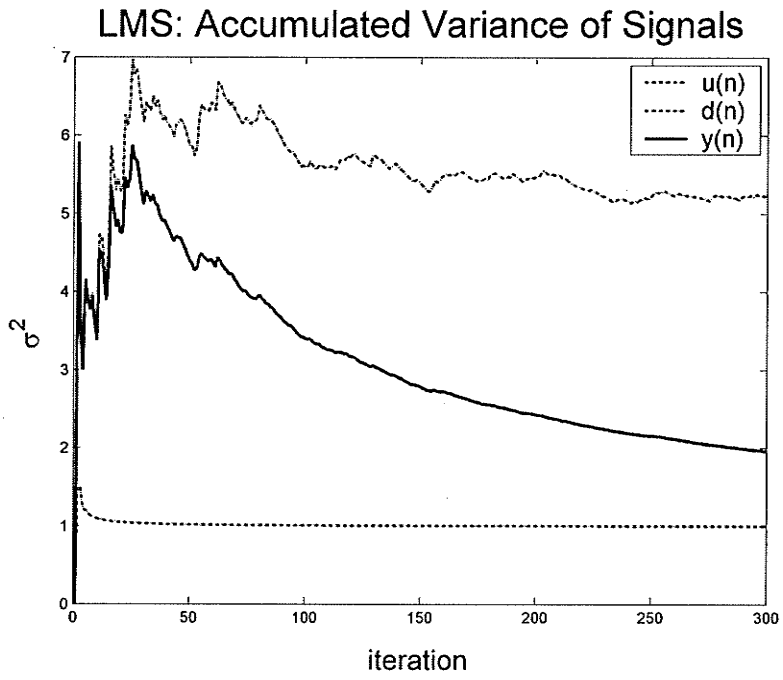
$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) + \mu \mathbf{x}(n) e^*(n) \tag{3.46}$$

The formulation in (3.46) and the calculation of **C** and $\mathbf{C}_a$ form the basis for the Matlab code development that the following analyses use. It should be noted that now *e(n)* is the beamformer output.

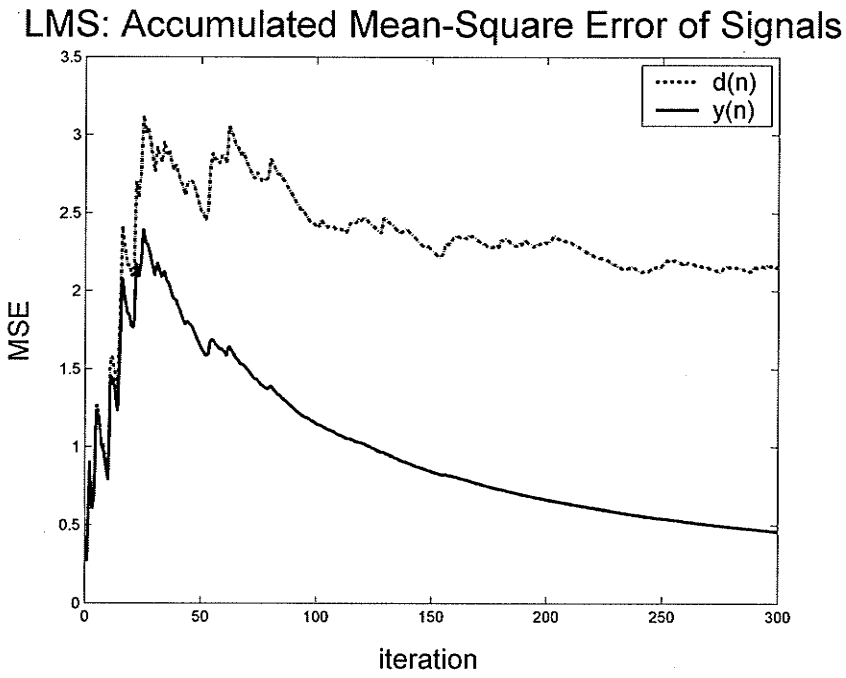
It is interesting to note that the LCMV implementation derives the best weights based on the noise environment in each moment, whereas the LMS method iterates to a best solution in response to a consistent noise presence. This can be seen in both Figure 3.12 and Figure 3.13, as compared to Figure 3.4.

The behavior over 900 iterations, in Figure 3.13, shows the convergence of the algorithm and compares the rate of convergence for μ = 1e-3 and μ = 1e-4.

Figure 3.14 shows the radiation pattern for the statistical parameters plotted in Figure 3.12. When the interference is in the mainbeam, the LMS algorithm also shifts the mainbeam, as was the case for the LCMV in Figure 3.5. the mainbeam is shifted 7.2° away from the AoI, with a corresponding gain reduction of -1.64dB at the AoA. However, at the AoI the gain is reduced from -3.8dB to -26.7dB.

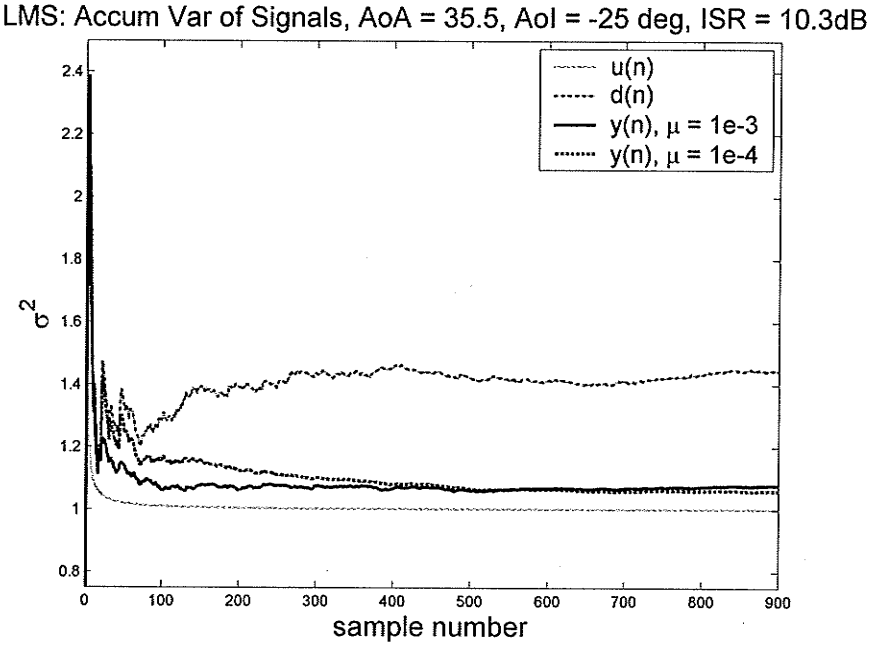


a) Accumulated variance.

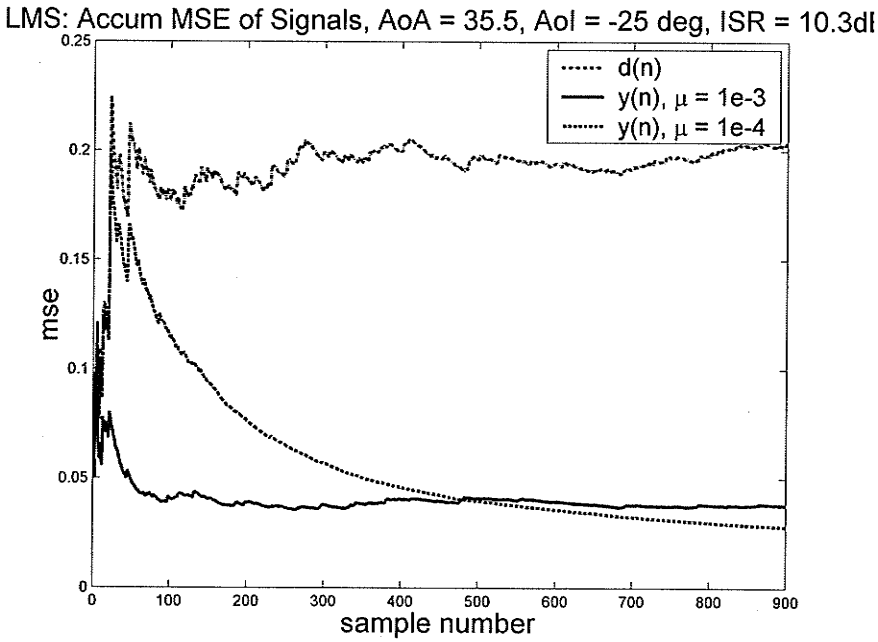


b) Accumulated mean-squared error.

Figure 3.12– LMS Method, Variance and MSE, 5-element array, 300 iterations, μ = 1e-4.



a) Accumulated variance.



b) Accumulated mean-squared error.

Figure 3.13– LMS Method, Variance and MSE, 5-element array, 900 iterations for μ = 1e-3 and μ = 1e-4.

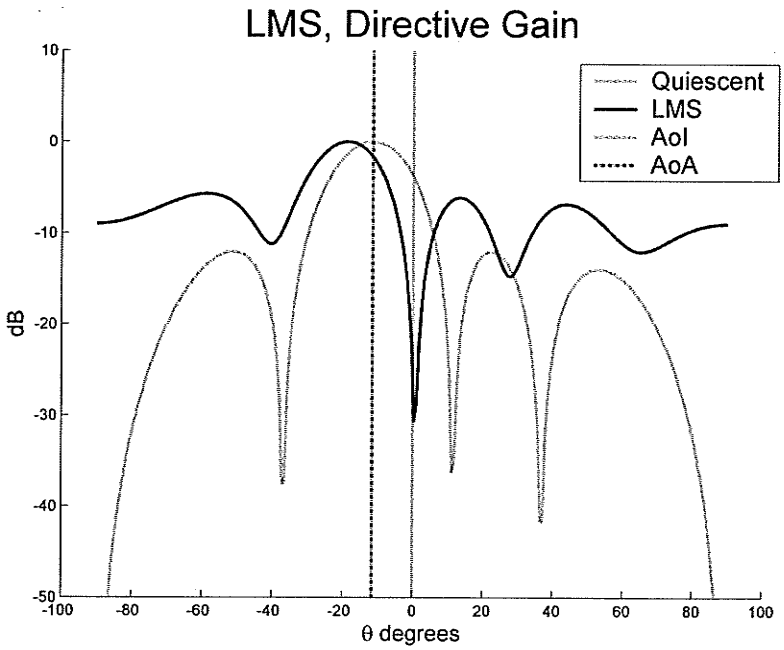


Figure 3.14– LMS, Quiescent and Adapted Array Factor. 5-element Array, AoA = -11.54°, AoI = 0°, μ = 1e-4, after 300 iterations.

Consider a source of interference that is not in the mainbeam, but rather a sidelobe. The radiation pattern is shown Figure 3.15 in for ISR = 10dB, SNR = 10dB, an AoA = -11.53° and an AoI = -53°.

Here, a displacement of less than 2° of the mainbeam is observed, but also with a slight widening of 3-dB beamwidth. At the AoA, there is a drop in gain of less than 0.1dB. At the AoI, however, the gain has dropped 13dB from – 12.2dB to – 31.6dB, but shows the deepest null being slightly off of the desired AoI. The corresponding variance and error performance are shown in Figure 3.16.

It is important to consider the performance of the LMS algorithm for in-phase and phase-shifted interference as was considered for the LCMV in Figure 3.10. Because the LMS also shows the impact of sinusoidal cancellation on lowering the variance of the output, the analysis is combined with a study of the adaptation scaling parameter μ.

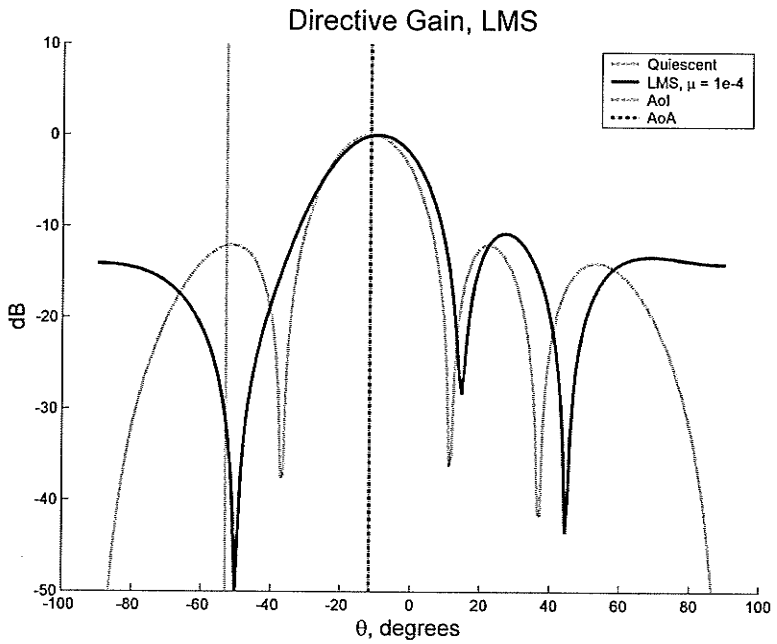
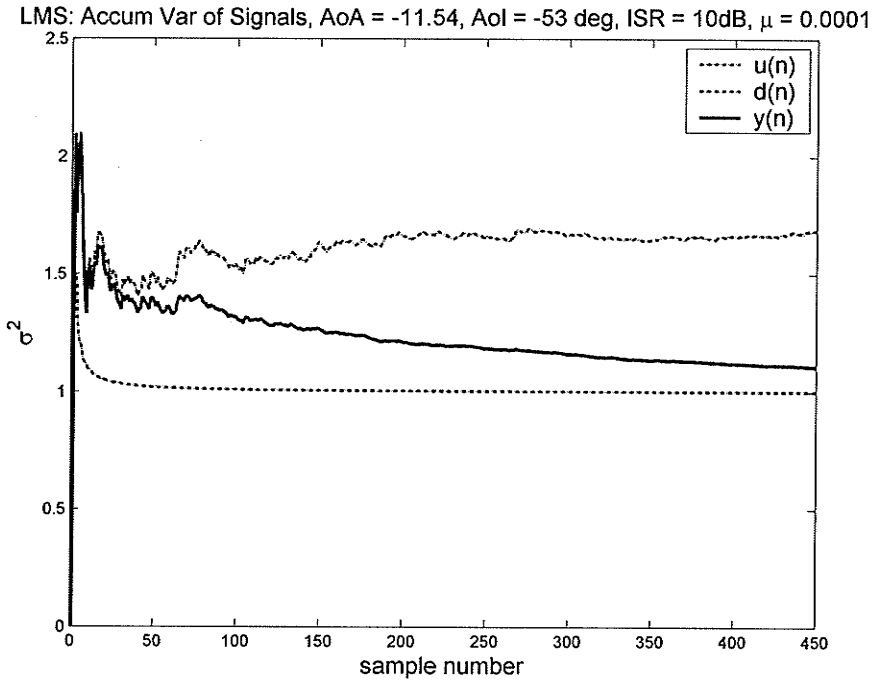
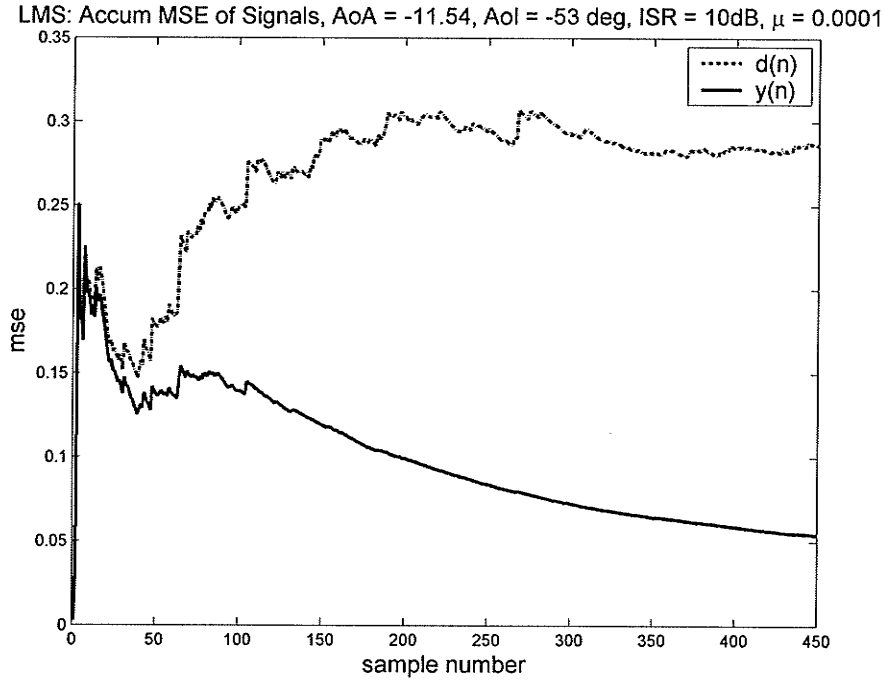


Figure 3.15– LMS cancellation of off-mainbeam interference. Quiescent and Adapted Array Factor for 5-element Array, AoA = -11.54°, AoI = -53°, μ = 1e-4, after 300 iterations.



a) Accumulated variance.



b) Accumulated mean squared error.

Figure 3.16– LMS Method, off-mainbeam interference, Variance and MSE, 5-element array, 450 iterations, μ = 1e-4.

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7
1e-5	2.1544e-5	4.6416e-5	0.0001	0.00021544	0.00046416	0.001

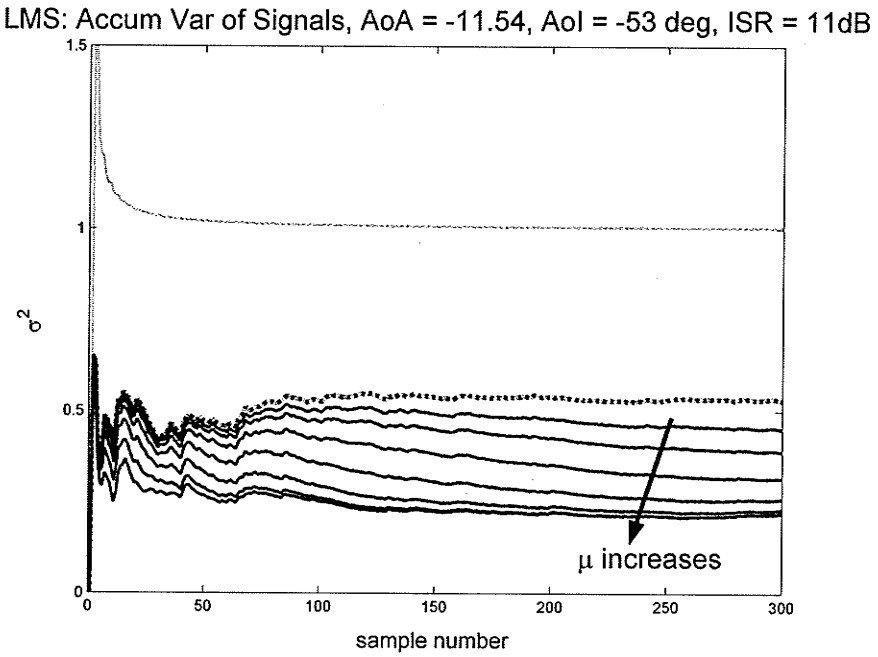
Table 3.3– Seven logarithmically spaced values for μ for testing LMS.

To do this, seven logarithmically spaced values for μ are used, as listed in Table 3.3.

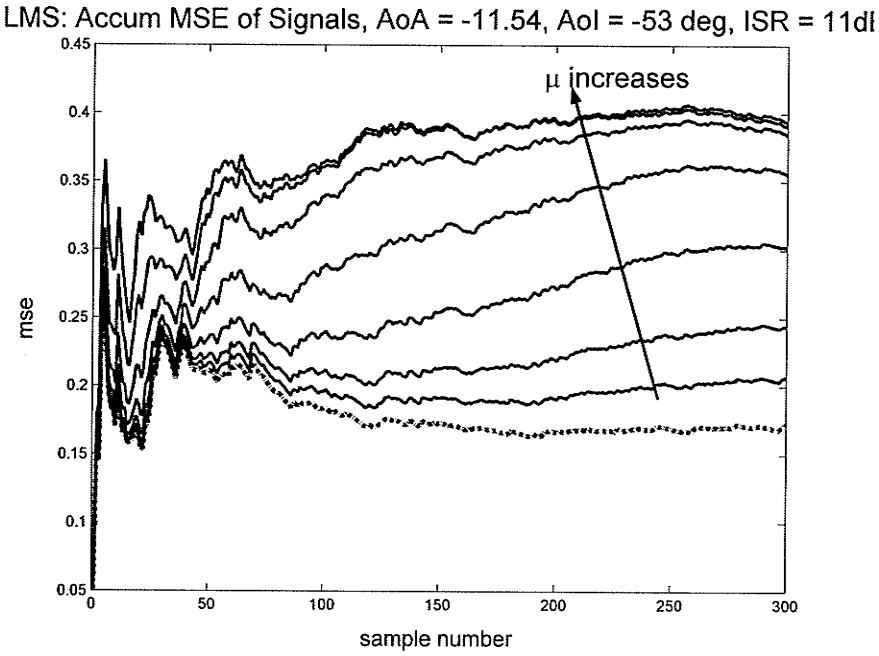
In Figure 3.17, not only does this sinusoidal cancellation produce an output with lower variance and higher mean squared error, but also it is observed that the best output is that of the quiescent weights, and that increasing μ only makes things worse.

To further illustrate this, Figure 3.18 shows the LMS output with an interference signal that is in phase with *u(n)*. Here the output is more according to expectation, with the LMS method improving performance over just the quiescent mode. Also, the rate of convergence as a function of increasing μ is apparent, but also shows the effect that if the choice of μ causes the variance of the adapted output to drop below that of the original signal, then the result is an increase in the mean square error of the output.

In fact, the points where the variance of the output of the array (the solid lines of $y_i(n)$ in Figure 3.18) drop below the unit variance of the original signal cause a corresponding inflection in the diminishing curves of the mean squared error. It is seen that the third value, μ = 4.64e-5, provides the best performance in this case.

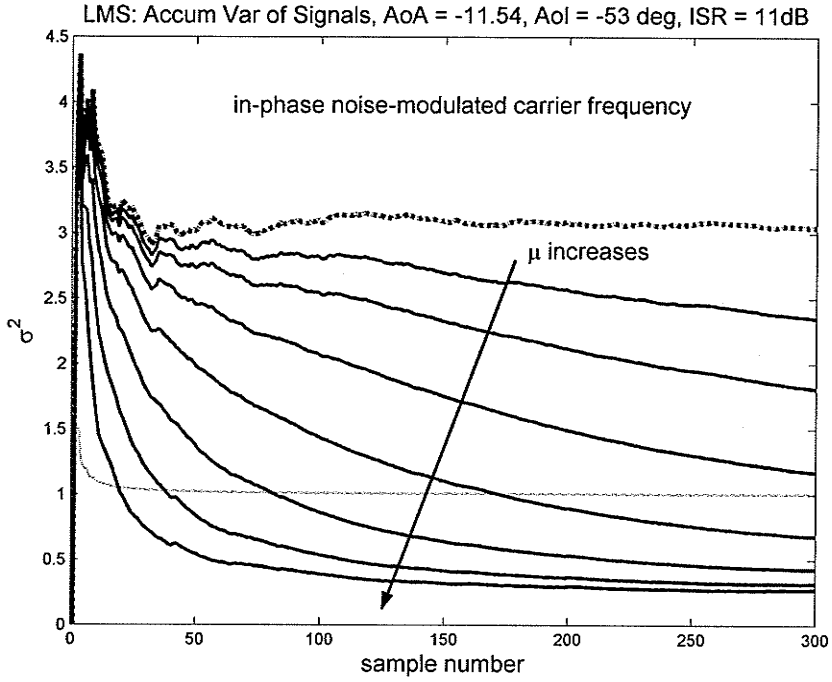


a) Variance.

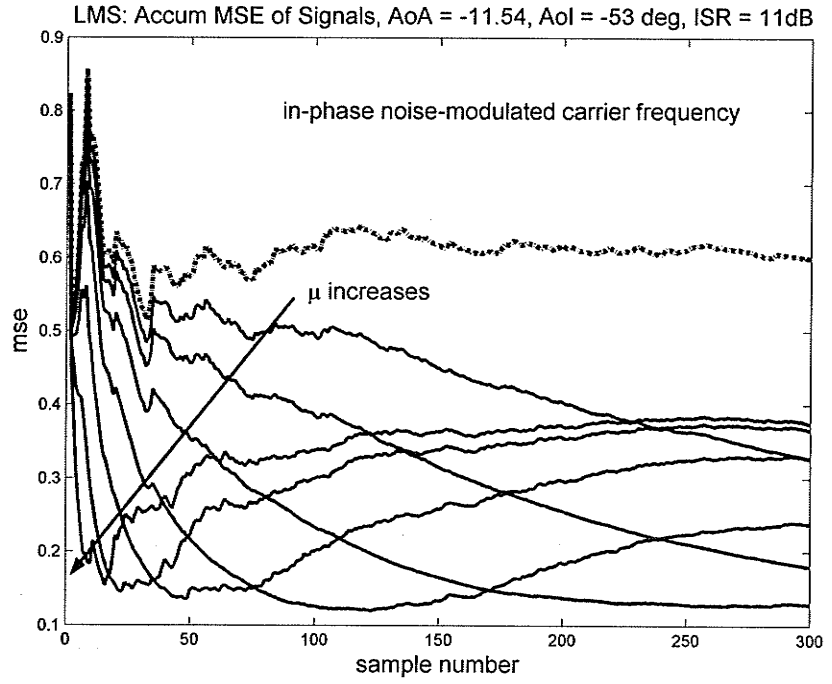


b) Mean-squared error

Figure 3.17– Variance and mean-squared error as μ varies, LMS, AoA = -11.53°, AoI = -53°, ISR = 11dB, SNR = 10dB. Quiescent output, *d(n)*, is dashed line. Interference is phase-shifted noise-modulated carrier frequency.



a) Variance



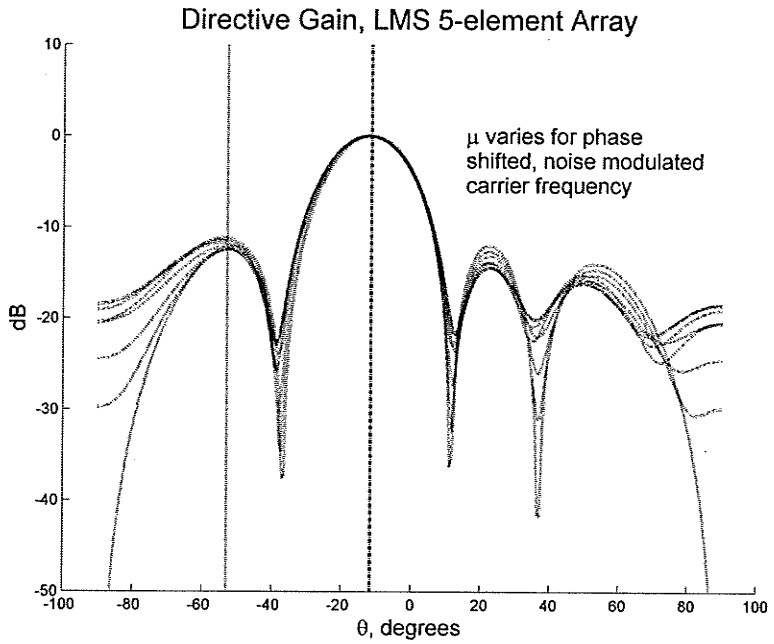
b) MSE

Figure 3.18– Accumulated variance and mean-squared error comparison of noise amplitude-modulated interference in-phase with $u(n)$ at the carrier frequency.

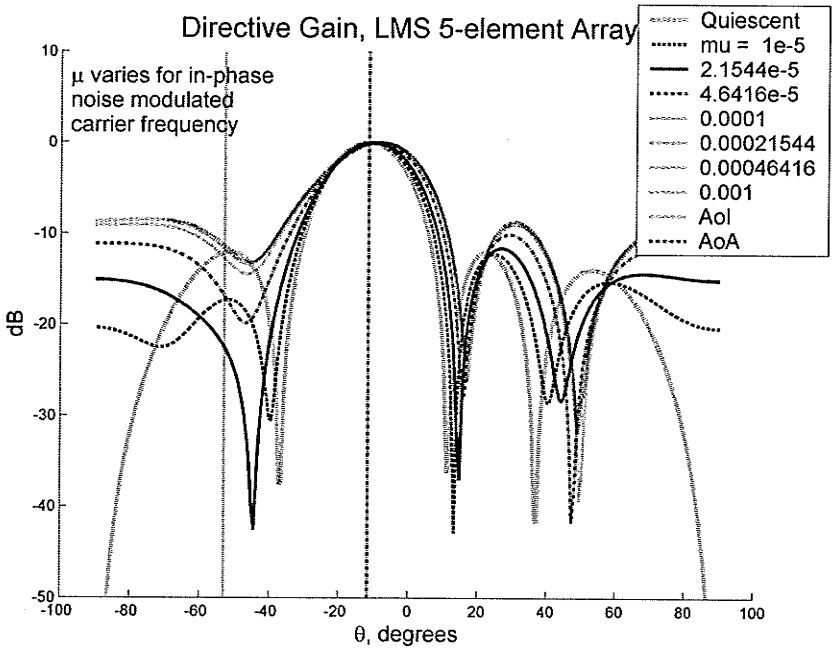
Thus we can see judicious choice of the adaptive parameter, μ, is very important in obtaining satisfactory performance from LMS algorithm. The directive gain patterns for both the in-phase and phase-shifted instances of noisy interference are shown in Figure 3.19. Here, it is observed that the LMS algorithm has no beneficial effect when there is phase related cancellation between the desired signal and the interference. If there is phase coherence, then there can be some adaptive nulling and Figure 3.19(b) indicates that the performance is best for μ_3.

Finally, consider the response of the LMS algorithm to broadband white noise, such as may be downconverted to baseband in a frequency modulated system. In this case, the LMS method performs really well, as shown in Figure 3.20.

The reason for the consistent shape of the mean-squared error curves as μ increases is because the identical noise environment has been used for each test to keep the noise constant while observing the effects of changing μ.

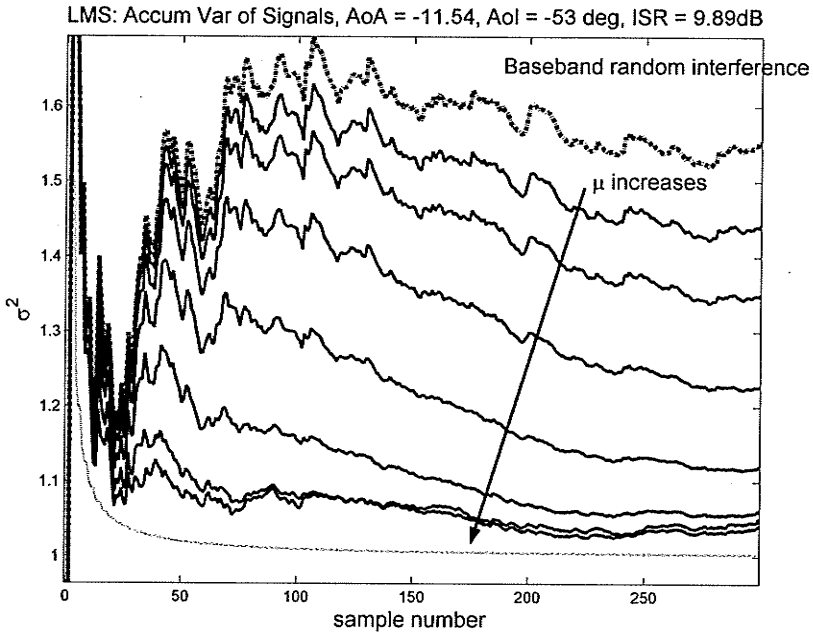


a) Phase-shifted sinusoidal interference.

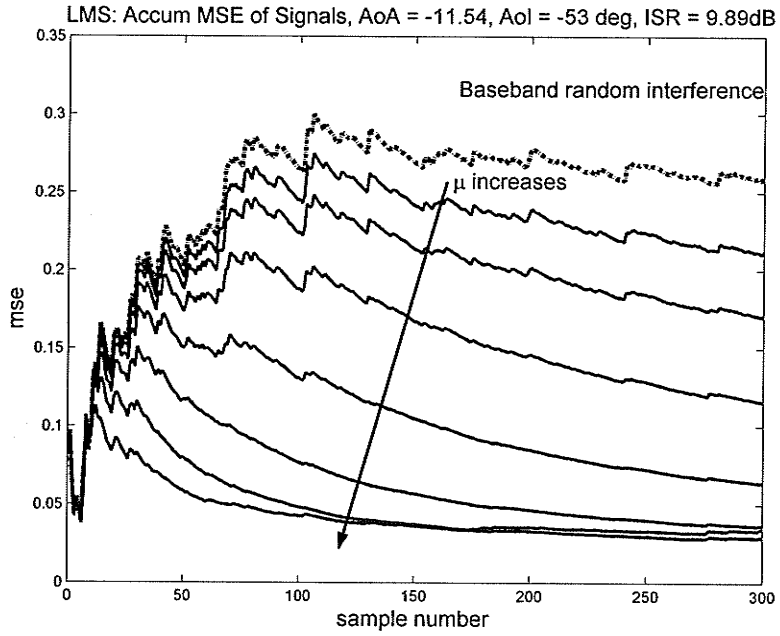


b) In-phase sinusoidal interference.

Figure 3.19– Array factor for 5-element array with sinusoidal interference for various μ.



a) Variance.



b) Mean-squared error.

Figure 3.20– Variance and MSE for 5-element array with baseband noise interference for various μ.

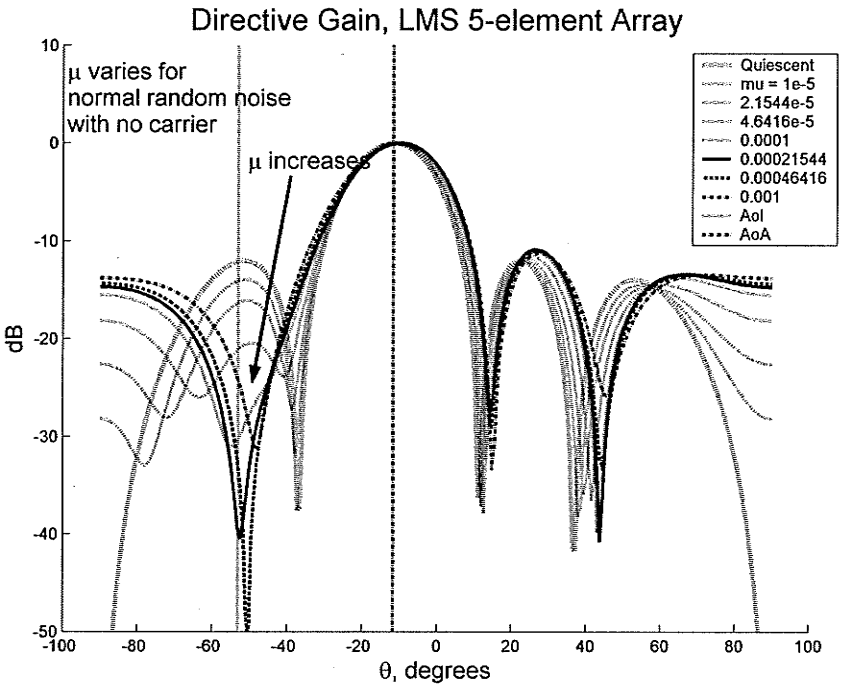


Figure 3.21– Array factor for 5-element array with baseband noise interference for various μ.

Figure 3.21 shows the array factor directive gain patterns for baseband random noise interference. The LMS method clearly performs well for all μ_i, with the best performance being for μ on the order of 2e-4.

3.4 Summary

In summary of this chapter, the overview of adaptive antenna array processing as an analogy to spatial digital filtering has been developed. The specific methods for determining both the quiescent and the adaptive weight portions of the composite weight vector have been presented for both the LCMV and the LMS algorithms. It was shown that both methods perform well in steering a null toward the source of interference for the angles of arrival and interference presented.

The LCMV method, based on instantaneous statistics, determines the best adaptive weight component at each sample instant in real time. This method is equally robust for sinusoidal interference, modulated carrier interference and baseband noise interference as was seen in Figure 3.9. It showed difficulty with phase-shifted carrier interference due to sinusoidal cancellation, but still produced results that were superior to the non-adapted output. When the source of interference was in the mainbeam, the algorithm shifted the mainbeam over while steering the null into the interference. This effect was not so pronounced for interference in the sidelobes.

The LMS method iterates to the best solution at a rate dependent on an adaptive scaling parameter μ. The method had difficulty with the phase-shifted carrier interference, again due to sinusoidal cancellation. The method produced an output inferior to the non-adapted output for all values of μ. If the interference was in-phase with the desired signal, the method produced an output which performed better than without adaptation. Still, the best performance of the LMS method was observed when the interference was baseband random noise. This implies that the LMS adaptive process should be applied at a stage where the incoming array signals have already been downconverted to baseband.

For interference in the mainbeam region, the LMS method produced less shifting of the mainbeam and a lower drop in gain at the angle of arrival of the desired signal than did the LCMV method.

4.0 Performance Measures of LCMV and LMS

This chapter investigates the behavior of the Linearly Constrained Minimum Variance (LCMV) method and the Least-Mean Squares (LMS) technique of adaptive antenna array processing as applied to the Generalized Sidelobe Canceller (GSC). The LCMV and LMS methods will be studied for several noise and interference environments and for a variety of angles of arrival of signal and interference. The methods will be compared from the performance criteria of the variance and the mean-squared error of the output with respect to the input. In the following chapter, the effects of adaptation on the overall array pattern will be observed and compared using Array Factor (AF) theory and NEC program output, as driven by the Matlab interface.

In the previous chapter, which presented the theory, formulations, testing and verification of the LMS and LCMV algorithms, only a few angles of arrival (AoA) and angles of interference (AoI) were considered. In particular, the LCMV was tested for an AoA = -11.537° with the AoI = 0° and -53°. The LMS method was tested for AoA = -11.537° and 35.5°, for the AoI = 0°, -25° and -53°. The performance of both methods was shown to work well at these angles, with the exception of sinusoidal cancellation effects degrading the performance of the LMS method as discussed.

In order to develop a greater performance measure of these methods, a larger set of pairs of angles for AoA and AoI need to be considered, and for differing levels of interference. This has yielded some interesting results. In particular, the LCMV method only works well for certain regions of angle pairs, and it seems to be an odd function with respect to the diagonal line of AoA = AoI on a scatter plot. The LMS algorithm performs well for

any arbitrary pair of angles. In the region where the LCMV works well, it maintains good performance right up to the ISR test limit of 50dB. In contrast, the LMS method performs well to an ISR threshold of about 35dB, after which it is worse than the non-adapted output. The details of these analyses are the subject of this chapter.

4.1 LCMV Performance Tests and Discussion

Recall that the LCMV process calculates the adaptive weight component by considering the instantaneous correlation matrix of the signals across the array. Thus, except for the assumption of wide-sense stationary noise and interference across the array, the weights from moment to moment are independent of each other.

From the figures showing the accumulated variance and mean squared error of the output with respect to the original signal, the instantaneous adaptive nature of the LCMV can be readily seen. Given roughly 50 samples for the transient to settle down, the LCMV method subsequently provides almost constant performance. It was observed, however, that the performance of the LCMV was not consistent for arbitrary angles of arrival (AoA) of the desired signal and arbitrary angles of interference (AoI) of the jammer. This will now be discussed in detail.

It was initially found that for a scan angle of AoA = $\sin^{-1}(-0.2) = -11.537°$, the LCMV performs well for an AoI of 0°, in the mainbeam, and -53°, in the first sidelobe. The array factor for (AoA,AoI) = (-11.54°,0°) is shown in Figure 4.1 for two sets of adapted weights, one from the 150^{th} iteration and one from the 300^{th} iteration.

This test is for a broadband interference noise of 10dB over the desired signal strength. The element SNR is also 10dB. The shifted mainbeam and subsequent gain at the angles of interest are consistent at the two time snapshots, but the sidelobe pattern varies considerably. This can be due to the fact that at the two instants the random interference and terminal noise components are different and the weights are adapted accordingly.

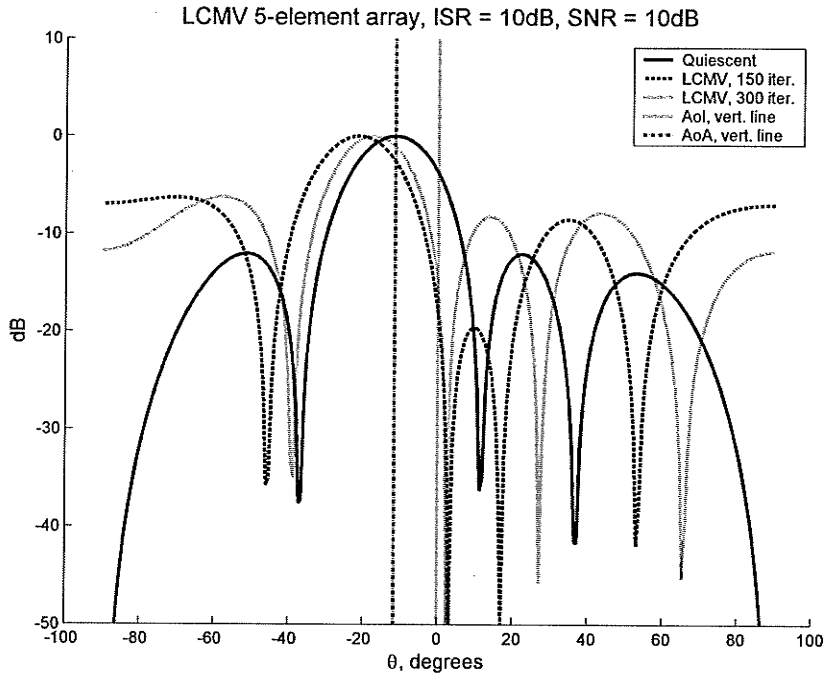


Figure 4.1– Array factor radiation patterns for AoA = -11.537°, AoI = 0°.

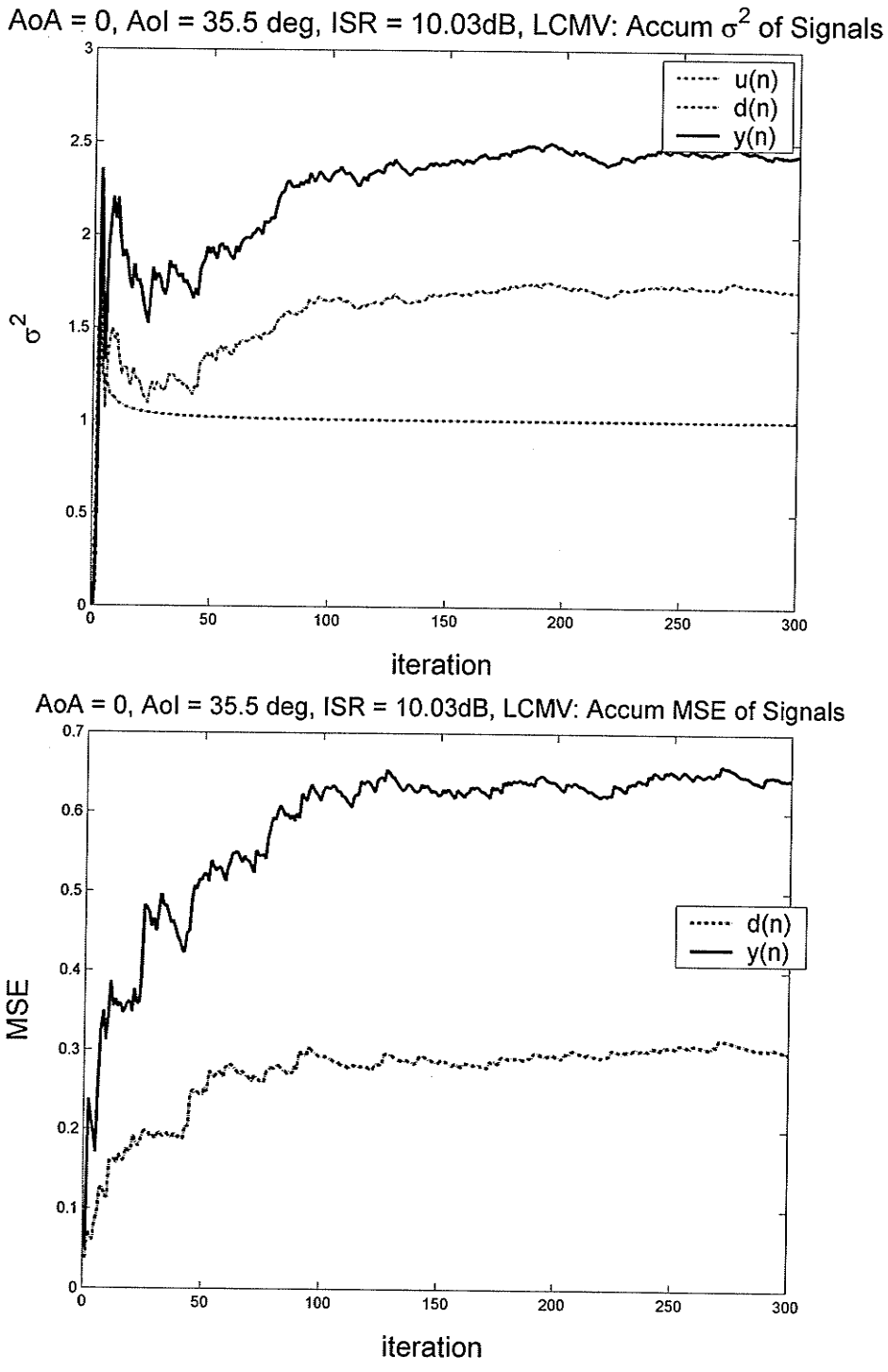
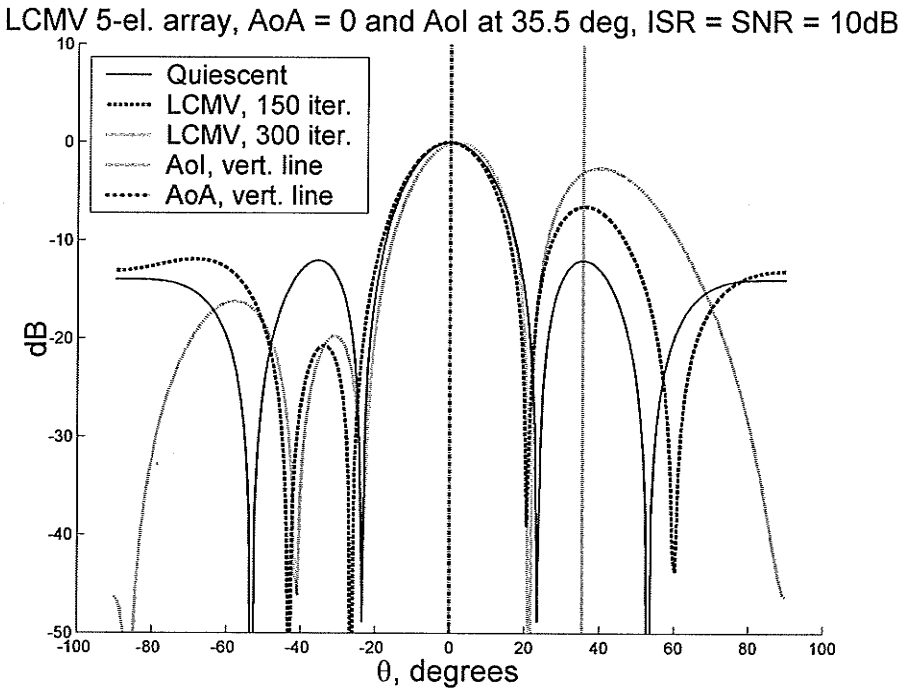


Figure 4.2– LCMV 5-element array poor performance for (AoA,AoI) = (0°,35.5°).

By translation, one would expect that moving the mainbeam to broadside at 0° and directing the interference to the first sidelobe would yield equally good behavior, but such was not the case, as seen in Figure 4.2. Figure 4. shows that a 5-element broadside array produces its first sidelobe at ±35.5°. The output statistics are worse and the array factor showed a suppression of the opposite first sidelobe and a gain in the sidelobe to be suppressed. Intuitively, one immediately thinks of a forgotten negative sign, or a phase shift of 90° or 180°, or even a complex conjugate issue. The formulations have been examined extensively and no error was found. The following performance evaluation and discussion will present more detail.



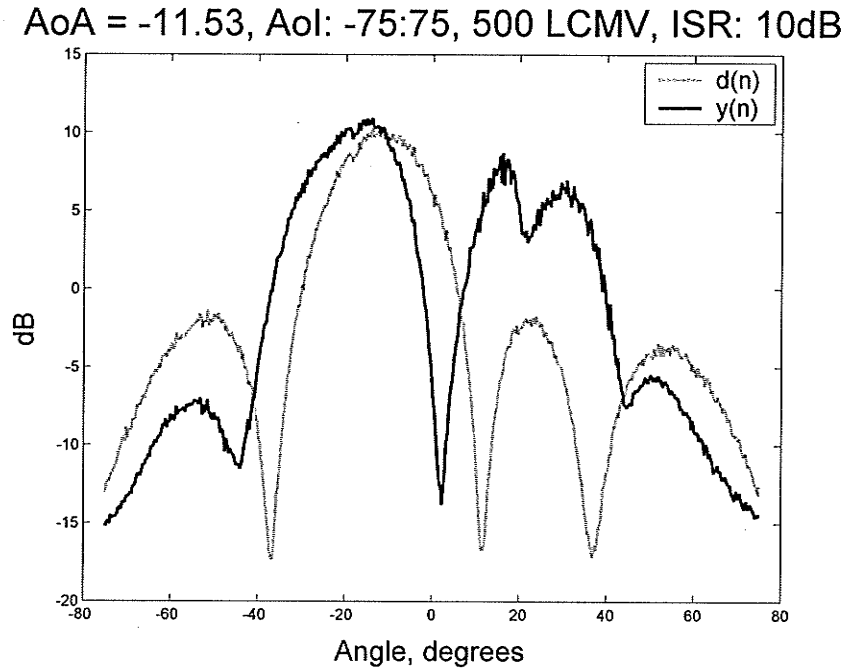
b)

Figure 4.3– LCMV 5-element array radiation pattern for (AoA,AoI) = (0°,35.5°).

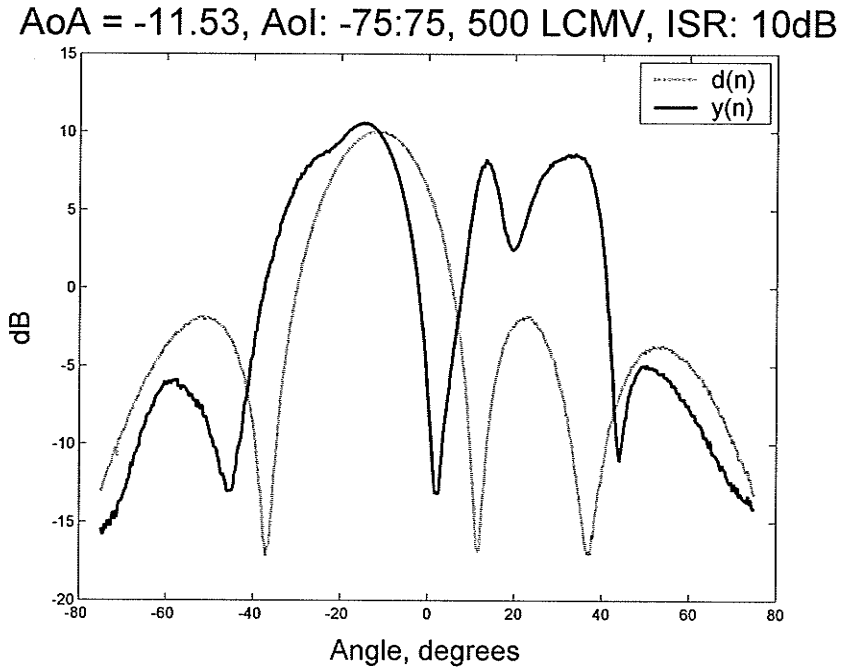
First, the LCMV method was tested by varying element noise power to find that higher SNR resulted in worse performance. This test was to determine if the potentially ill-conditioned denominator of (3.32) was causing the problem. It was not.

This led to an investigation of the performance of the LCMV and LMS algorithms as a function of AoA, AoI, interference and noise power. First, an AoA with known good LCMV performance for an AoI was chosen, and the AoI was swept 500 equally spaced points from -75° to 75°. The AoA = -11.537°, as was used in the previous examples for an AoI = 0°, and shown in Figure 4.1 where reasonable working performance was established. Shown in Figure 4.4 for both random baseband and pure sinusoidal interference at an ISR = 10dB and with SNR = 10dB, this test indicates that the LCMV method is an improvement on quiescent weighting for angles of interference from -75° to -40°, -11° to 10° and above 45° for both types of interference. The curves are of the mean squared error (MSE) in dB of the quiescent and adaptive output with respect to the original signal, *u(n)*. Indeed, it indicates that good performance is to be expected for an AoI = 0°, as has been studied. However, there are clearly regions where the LCMV method worsens the performance of the array.

Looking at the behavior of Figure 4.4, it is seen that for an AoA = -11.54°, an AoI = -51° gives good performance and an AoI = 23° yields inferior performance, at least for an ISR = 10dB. Thus, this became the focus of a power level test, by varying the interference power level from 10^{-3} to 10^{5}, for -30dB ≤ ISR ≤ 50dB, as shown in Figure 4.5 and Figure 4.6 below.

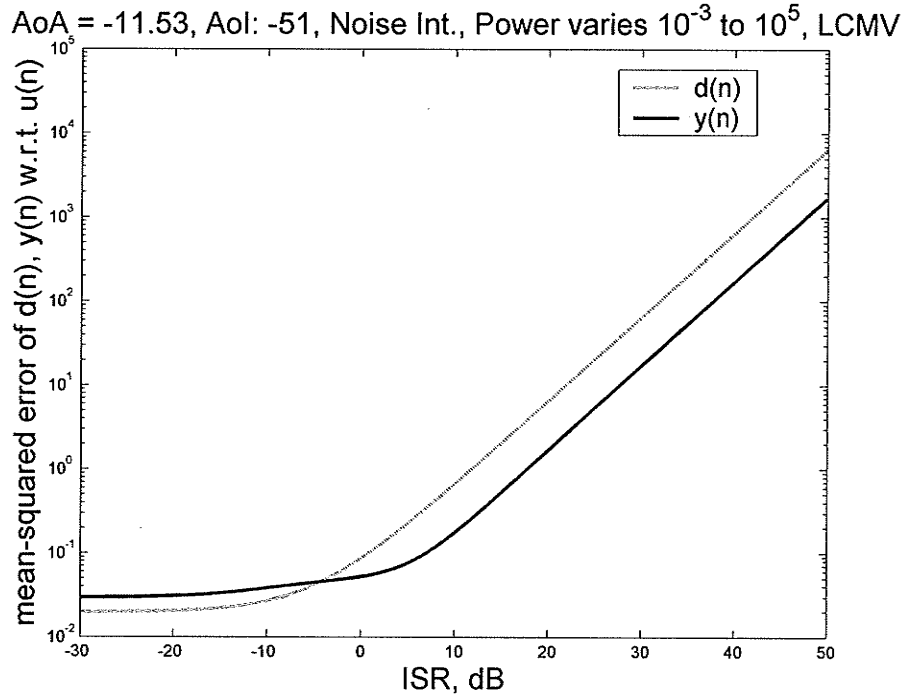


a) Random noise interference.

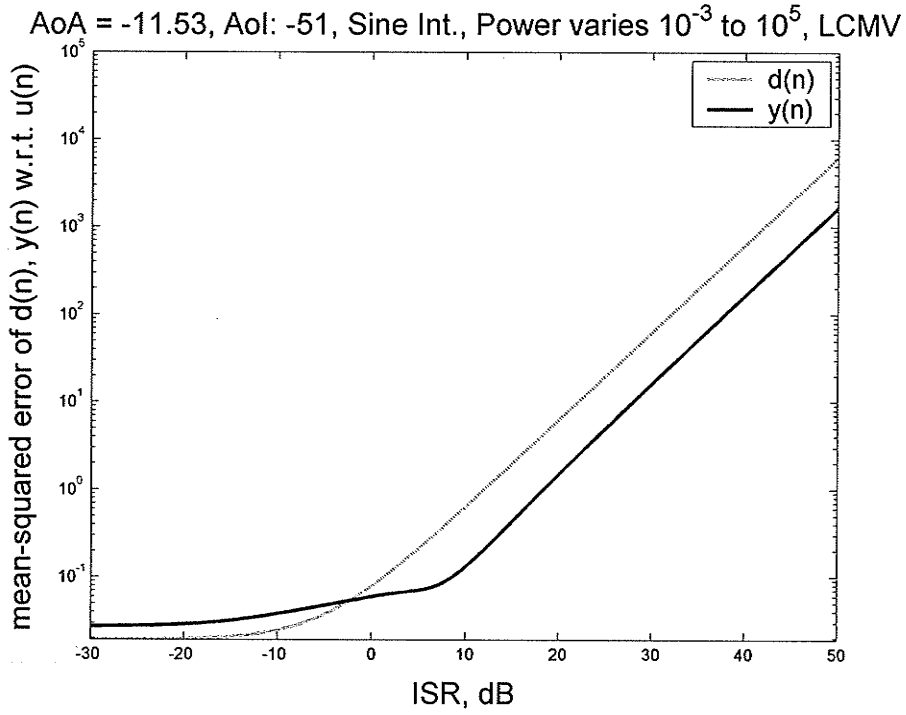


b) Sinusoidal interference.

Figure 4.4– A sweep of AoI over 500 points from -75° to 75° for an AoA of– 11.537°, using both baseband random and sinusoidal interference, showing the MSE in dB vs. angle. Lower levels indicate better performance.



a) Baseband noise interference.



b) Sinusoidal interference.

Figure 4.5– Mean squared error performance of quiescent output *d(n)* and LCMV adapted output *y(n)* as interference power level varies for -30dB ≤ ISR ≤ 50dB for (AoA,AoI) = (-11.54°,-51°).

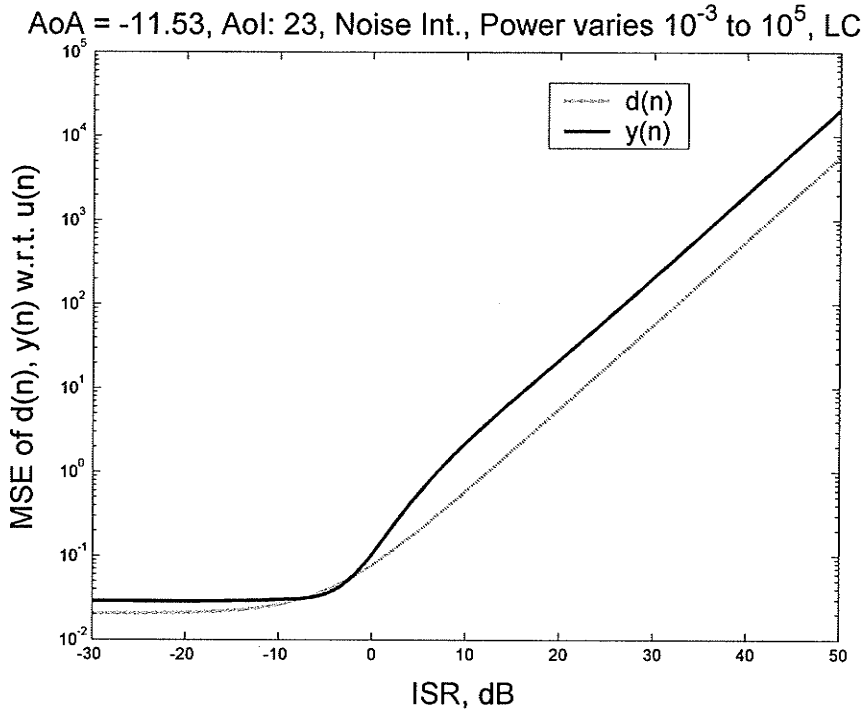
Figure 4.5 indicates that indeed the LCMV method gives improvement for ISR levels above 0dB, but is surprisingly worse for levels below. In addition, the lines are parallel indicating that the gain in performance is a constant ratio regardless of the ISR, right up to the maximum level of 50dB. Note that the *y*-axis is on a log scale.

Figure 4.6, on the other hand, shows that the LCMV method offers no improvement at the troublesome angles of (AoA,AoI) = (-11.54°,23°), except for marginal results between 0 and -10dB. It is interesting to note that now the degradation in performance is a constant ratio, as indicated by the parallel lines. In both cases it was found that whether the interference was sinusoidal or random noise did not make much of a difference.

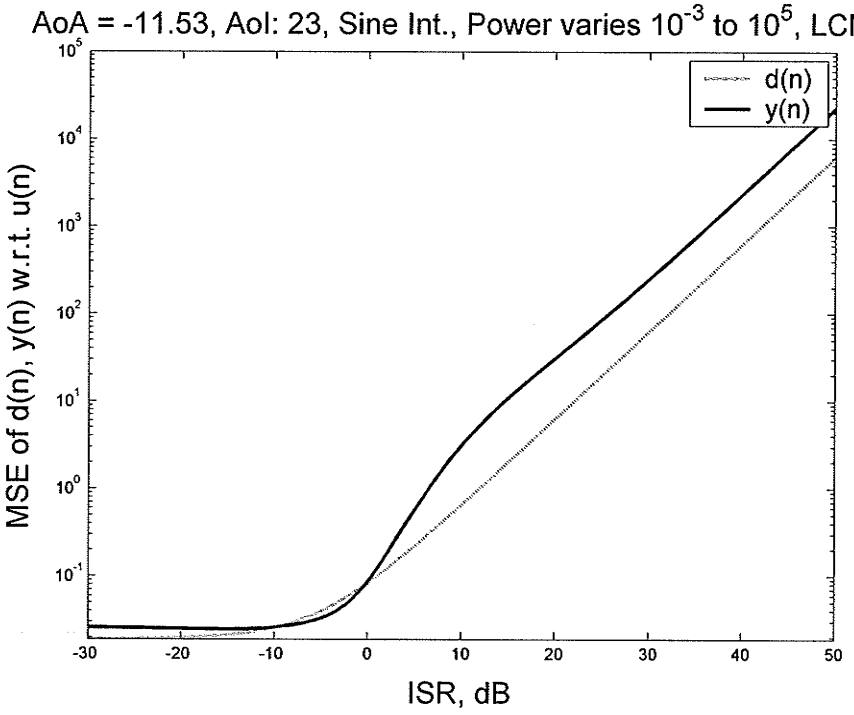
From these tests, it was concluded that neither the SNR nor the ISR was the cause of the poor performance of the LCMV algorithm for certain AoA and AoI angle pairs. In both cases, improvement of the output by LCMV adaptive processing did not become apparent until the ISR was greater than 0dB.

The next consideration is to determine how the LCMV method behaves when either the AoA or the AoI is set to 0° for a range of values of the other. In addition, it will be useful to look at the behavior of the output as a scatter plot of points of better or worse performance for AoI versus AoA.

It was found that if the AoA was set to 0° (broadside), the performance was worsened for almost all angles of interference. However, if the AoI was set to 0°, the LCMV method improved the performance for all angles, as shown in Figure 4.7. Note that the fundamental quiescent patterns are similar to the directive gain pattern of the array.



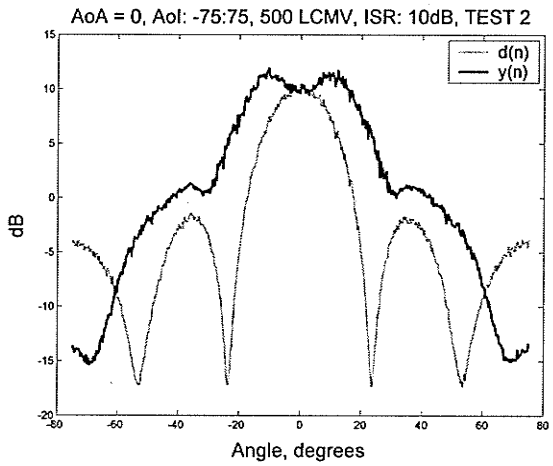
a) Baseband noise interference.



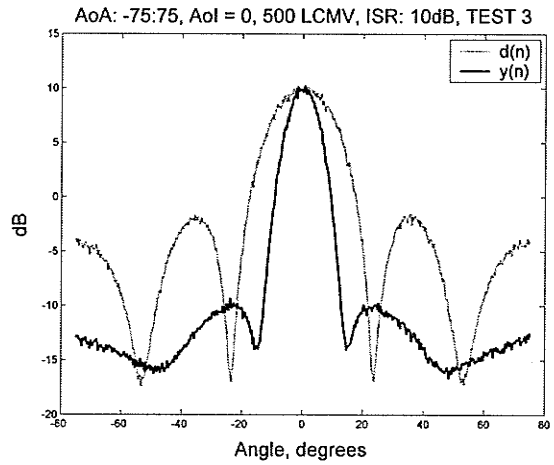
b) Sinusoidal interference.

Figure 4.6– Mean squared error performance of quiescent output *d(n)* and LCMV adapted output *y(n)* as interference power level varies for -30dB ≤ ISR ≤ 50dB for (AoA,AoI) = (-11.54°,23°).

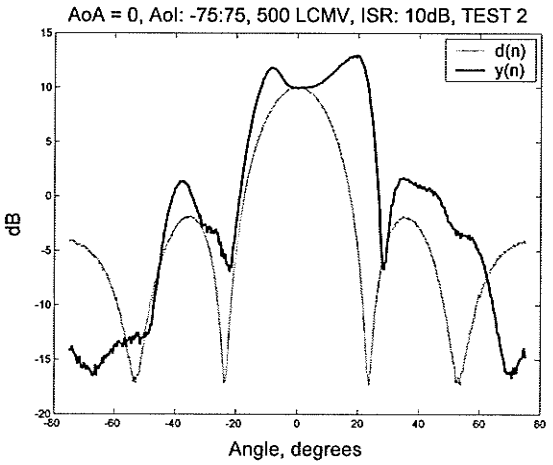
Looking at Figure 4.7 from the point of view that this error curve relates to the directive gain pattern, it confirms expectation that maximum error would be at angles of maximum gain. Where the adapted output is worse than the quiescent output in a null, it is really of no consequence to the validity of the method.



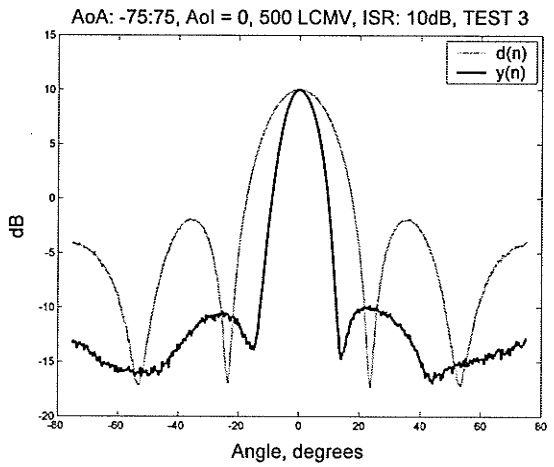
a) Noise interference



c) Noise interference



b) Sinusoidal interference



d) Sinusoidal interference

AoA = 0°, -75° < AoI < 75° **AoI = 0°, -75° < AoA < 75°**

Figure 4.7– LCMV shows poor performance for AoA = 0° but good performance when AoI = 0°.

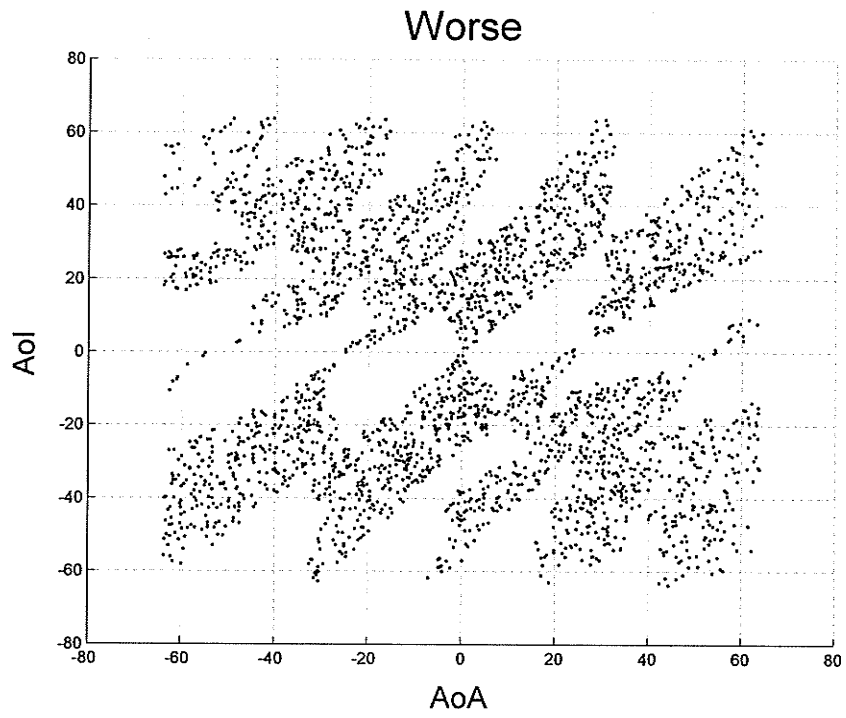
If the interference is already appearing in a null of the quiescent array pattern, it is already being suppressed. Thus, Figure 4.7(c) and (d) show excellent performance.

Thus, to investigate this unexpected performance more exhaustively, a test was set up to randomly select 5000 angle pairs for (AoA,AoI), between -75° and 75°, and determine the LCMV method's performance at each point. The test was performed for a complex-noise amplitude modulated interference signal with an ISR of 15.7dB, as given in (3.33) and shown in Figure 3.8. The results of this test are shown in the scatter plots of Figure 4.8 and Figure 4.9, where better performance means that the array factor gain at the AoI is lower for the adapted weight vector than for the quiescent weight vector.

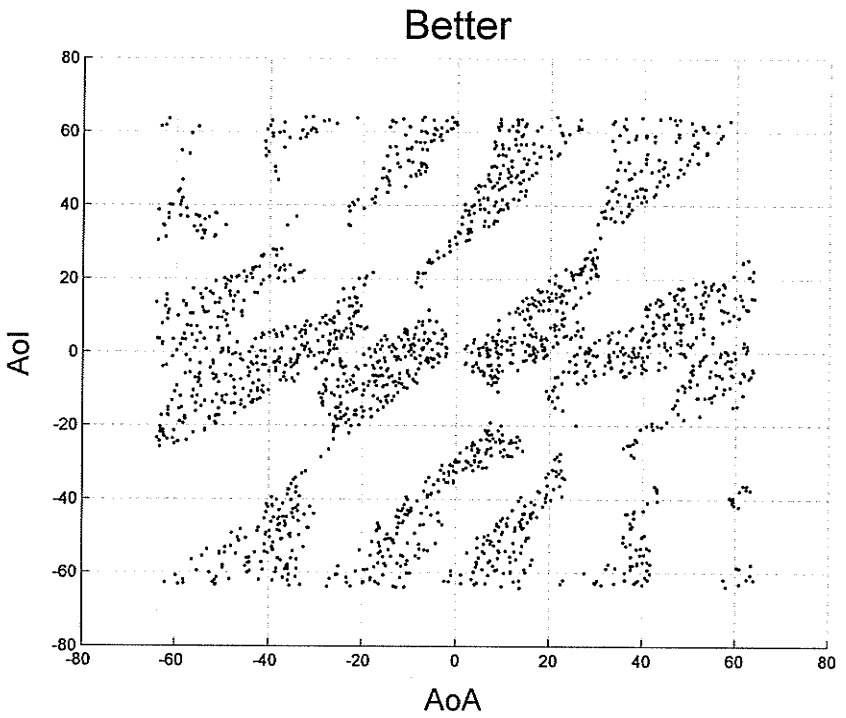
Figure 4.8 clearly indicates a performance problem. In fact, of the 5000 test points, 2904 returned adapted output that was worse than quiescent output. Note how these scatter plots seem to be an odd function with respect to the diagonal line of AoI = AoA.

Figure 4.9(a) shows the better and worse performance together, but only for regions where the quiescent gain pattern would be within 20dB down from the peak. Thus, the bands of deep nulls are visible. The odd function, as rotated around the AoI = AoA line is again apparent and hints of some fundamental difficulty in the formulations.

Figure 4.9(b) shows a scatter plot of adapted gain versus quiescent gain, with a point for each (AoA,AoI) pair. The diagonal line is where the adapted gain is equal to the quiescent gain. Any point below this line indicates a lowering of gain at the angle of interference and thus an improvement in performance. The distribution about this diagonal indicates that for arbitrary angle pairs the user cannot be confident that the LCMV method will yield a net improvement in performance.



a) Performance worse than quiescent.

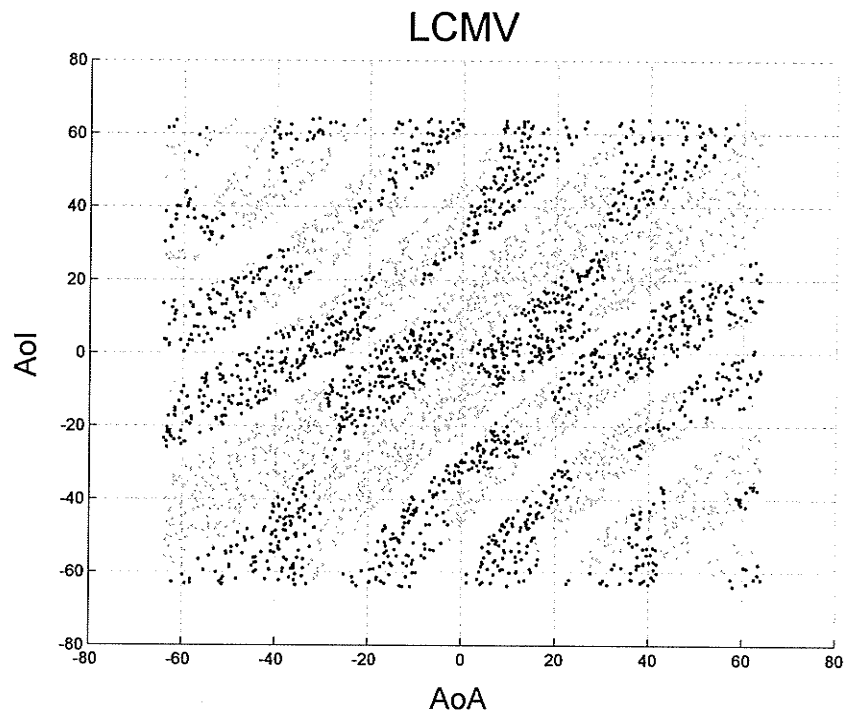


b) Performance better than quiescent.

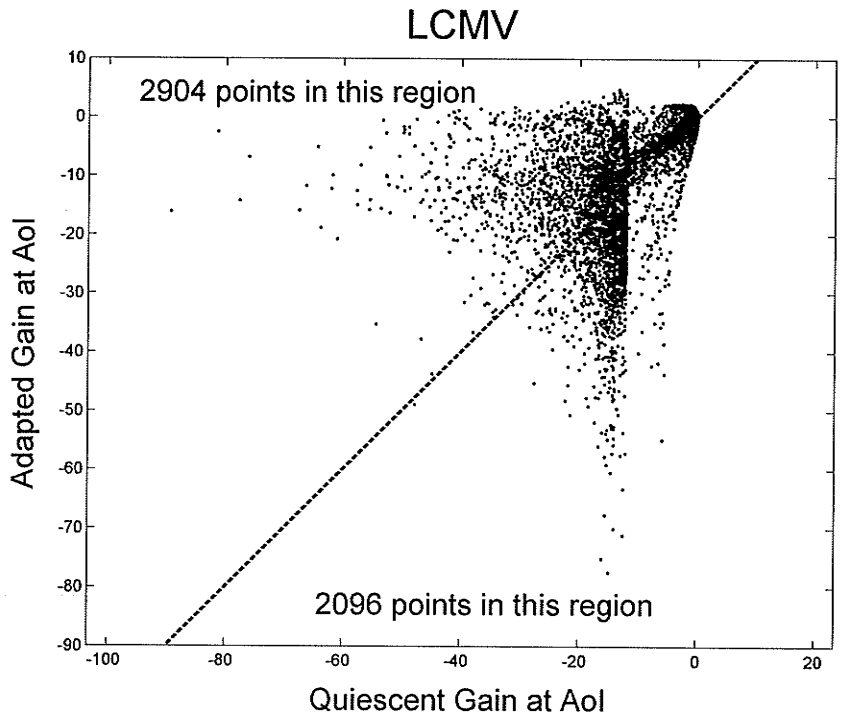
Figure 4.8– Scatter plots of LCMV method, AoI versus AoA. Performance is better if the adapted output *y(n)* has a lower mean squared error than the non-adapted, or quiescent, output *d(n)* with respect to the original signal *u(n)*.

As a brief discussion of this, it is important to note two things. First, there are two stages in the computation of the algorithm - *1)* the formulation of the constraint matrix **C** and its orthogonal complement $\mathbf{C}_a$ and the steering vector for the interference from the given (AoA,AoI) and *2)* the formulation of the subsequent LCMV algorithm using these quantities. The LMS method, which works very well as will be shown in the next section, uses the exact same quantities, **C**, $\mathbf{C}_a$ etc., determined in the first stage and differs only in the formulation of the subsequent LMS algorithm. Since the LMS method demonstrates exceptional performance, if there is a flaw in the formulations it cannot come from the first stage of computations.

The second point to note is that the underlying assumption in the formulation of the LCMV method is that there is some knowledge of the true probability density functions of the signals as expressed by the presence of the mathematical expectation operator. However, the true statistics are being estimated by the sample statistics via the *mean ergodic theorem*, which if it applies states that the statistics along the discrete sample (time, or in our case the number of antennas in the array) converge to the statistics across the process as the length of the discrete sample tends to infinity. For a 5-element array, the discrete sample is only of length five. It is difficult to assume that five instantaneous values represent the statistics of a stochastic process. The test was performed using larger arrays, up to 33 elements, but performance did not improve. Further, the mean ergodic theorem applies if the process is asymptotically uncorrelated. The random element noise is uncorrelated, but the desired signal and the interference are both correlated across the array by the inter-element phase delay.



a) Only points where quiescent pattern > -20dB. Gray is worse, Black is better performance.



b) Scatter plot of adapted gain versus quiescent gain, measured at the AoI. Points below the diagonal indicate improved nulling.

Figure 4.9– LCMV scatter plots of AoI versus AoA and Adapted Gain versus Quiescent Gain.

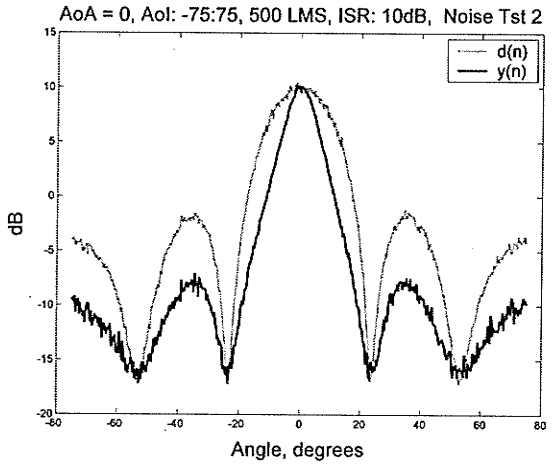
Two possible avenues of investigation have not been explored. One is that the statistics across the array at each iteration could be stored and accumulated such that as the process continues from start-up it bases the correlation estimates on larger and larger numbers of samples. However, accumulating and averaging the autocorrelation of the signals across the array before composing the correlation matrix was studied, but with no benefit. The other is that perhaps there is some normalization going on, or lack thereof, that is not being considered, either analogous to the normalized angular frequency of filter theory, or in terms of appropriately scaling the adaptive weight portion with respect to the quiescent weight portion. The latter was investigated to a small extent with inconsistent results. A tangible explanation for the poor general performance of the LCMV method remains inconclusive.

4.2 LMS Performance Tests and Discussion

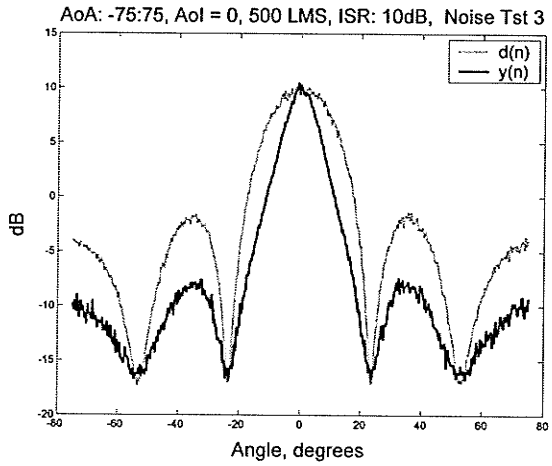
This section repeats much of the testing of the previous section using the LMS algorithm. In general, the LMS method performs very well given an appropriate rate of convergence μ and understanding the effects of sinusoidal interference. Unless otherwise stated, the value μ = 1e-4 is used in the analyses of this section.

Similar to Figure 4.7 for the LCMV method, the first test holds the AoA fixed at 0° while sweeping the AoI from -75° to 75°, and then holds the AoI fixed at 0° while sweeping the AoA from -75° to 75°. The ISR = SNR = 10dB. Each test is performed for an in-phase noiseless sinusoid and for baseband random noise interference.

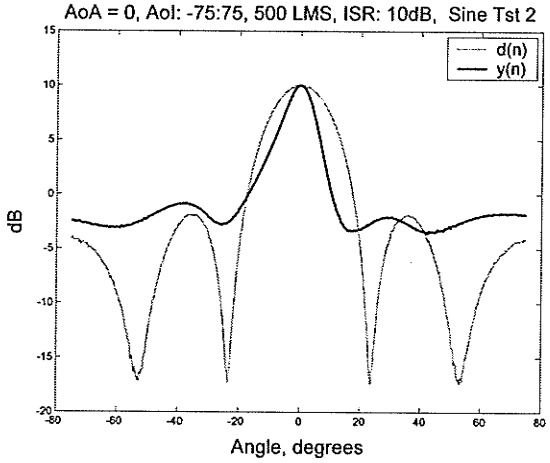
The results of this test are shown in Figure 4.10 and indicate that the LMS method is less sensitive to varying angles, but more sensitive to sinusoidal interference. One sees that



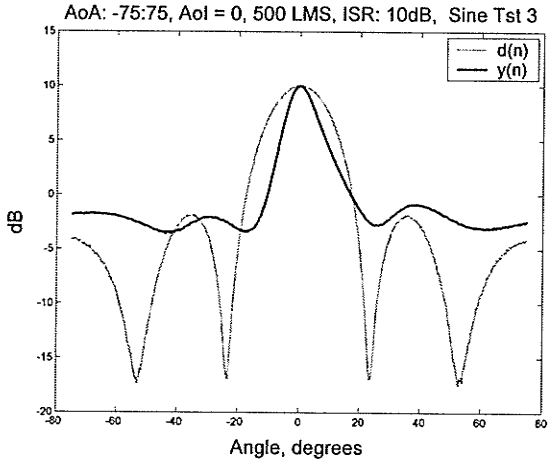
a) Noise interference



c) Noise interference



b) Sinusoidal interference

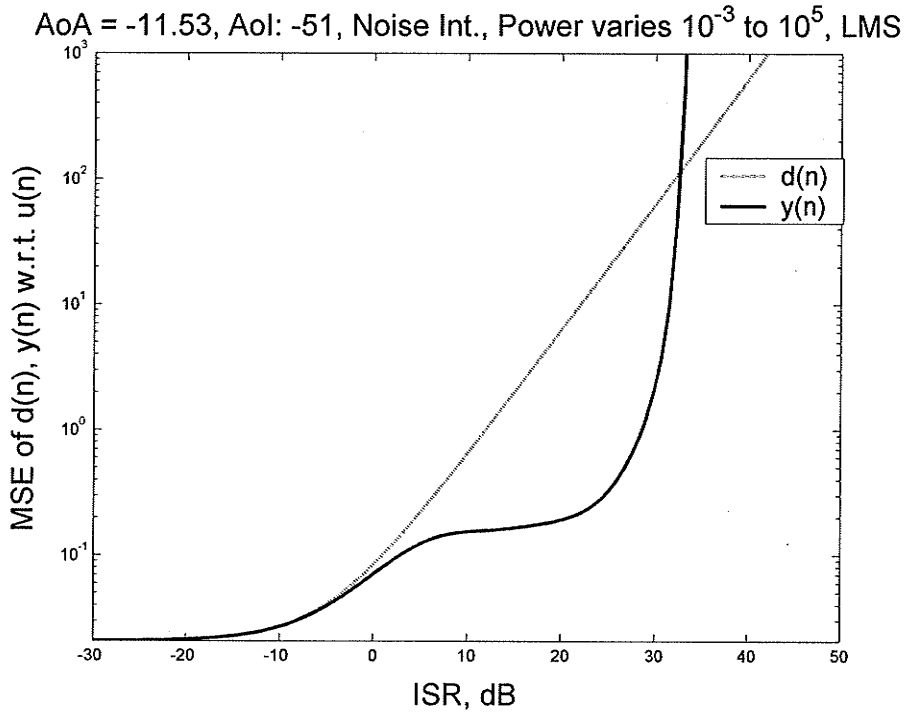


d) Sinusoidal interference

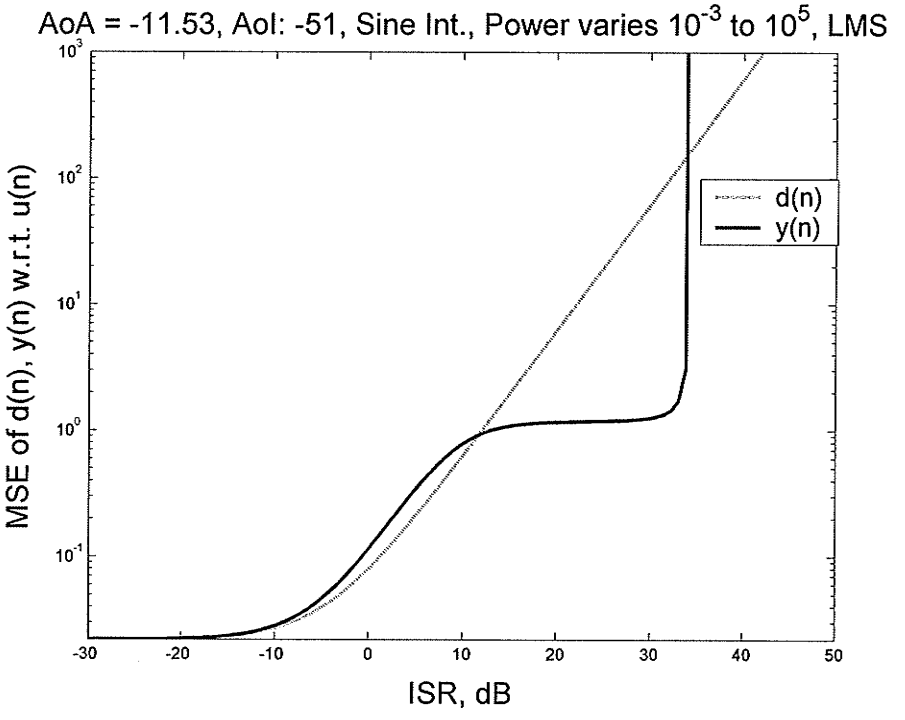
AoA = 0°, -75° < AoI < 75° **AoI = 0°, -75° < AoA < 75°**

Figure 4.10– LMS shows good performance for noise interference, ISR = SNR = 10dB, but good performance only in the mainbeam when the interference is sinusoidal.

the noise interference results in good performance everywhere, but the sinusoidal interference is canceled sufficiently at this power level only if it appears in the mainbeam. Sinusoidal interference to the sidelobes is not handled well.



a) Baseband noise interference.



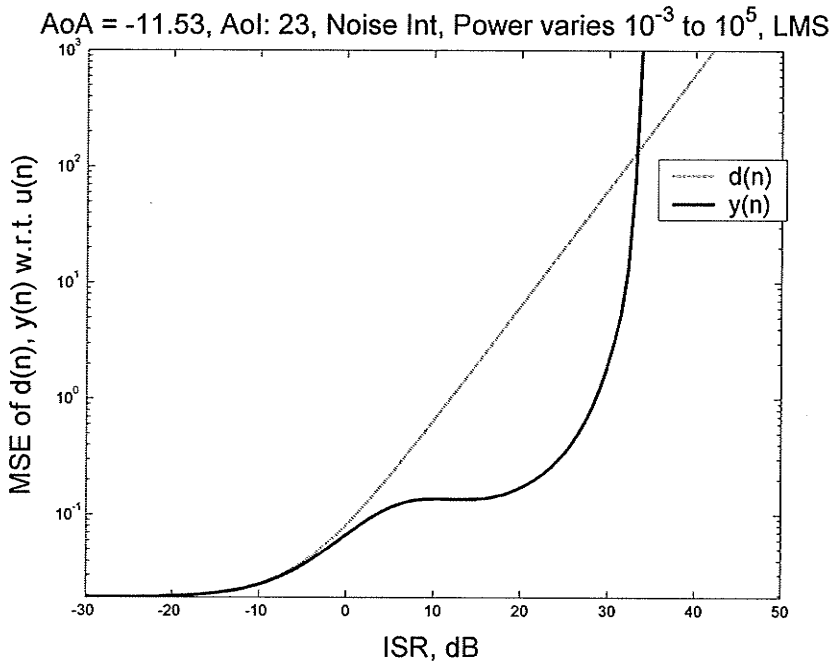
b) Sinusoidal interference.

Figure 4.11– Mean squared error performance of quiescent output *d(n)* and LMS adapted output *y(n)* as interference power level varies for -30dB ≤ ISR ≤ 50dB for (AoA,AoI) = (-11.54°,-51°).

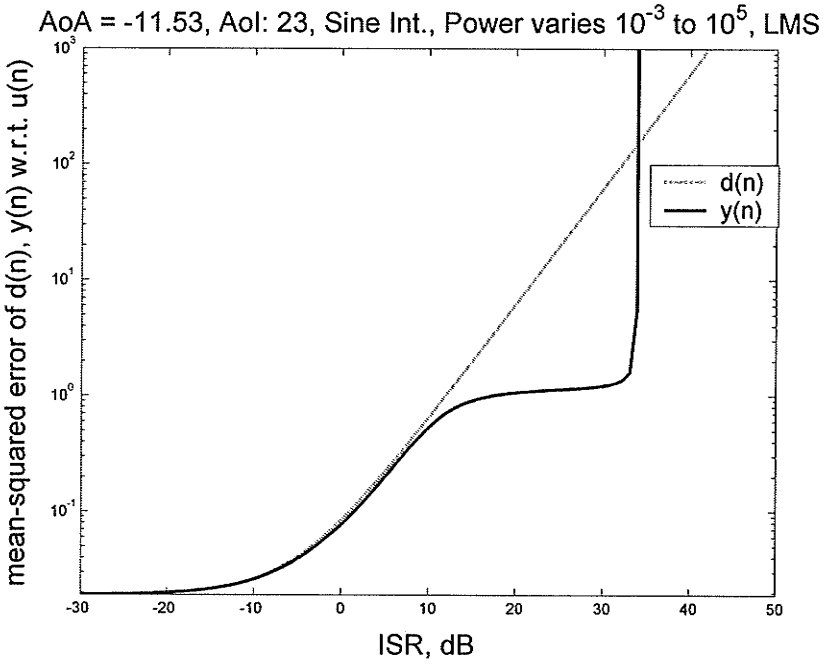
Figure 4.11 shows the LMS method performance as a function of interference power levels ranging over -30dB ≤ ISR ≤ 50dB, for (AoA,AoI) = (-11.54°,-51°). The performance is plotted as the mean squared error versus ISR, with the MSE on a log scale. Figure 4.12 shows the same for (AoA,AoI) = (-11.54°,23°). Several interesting points are worth observing, the first of which is the incredible improvement in canceling out interference up to a threshold of approximately ISR = 33dB for both noise and sinusoidal interference.

Now look at the AoI sweep test performed for AoA = -11.537° and ISR = SNR = 10dB in Figure 4.13. Clearly, when the AoI = -51°, the LMS algorithm should improve the performance for noise interference and worsen the performance for sinusoidal interference. In addition, we expect to see an improvement in performance for both interference types at AoI = 23°, although marginal for the sinusoidal interference.

With respect to Figure 4.11 and Figure 4.12, for which a zoom of the low-ISR regions is shown in Figure 4.14, poor performance is expected for sinusoidal interference at (AoA,AoI) = (-11.537°,-51°) up to approximately ISR = 11dB, after which the method will show an improvement. At 23°, marginal improvement in interference suppression is expected, and will improve dramatically as the ISR increases. In all cases, we see marginal or no improvement in performance for low ISR levels. This is as to be expected, since for low levels of interference, the algorithm should converge to the quiescent weight vector.

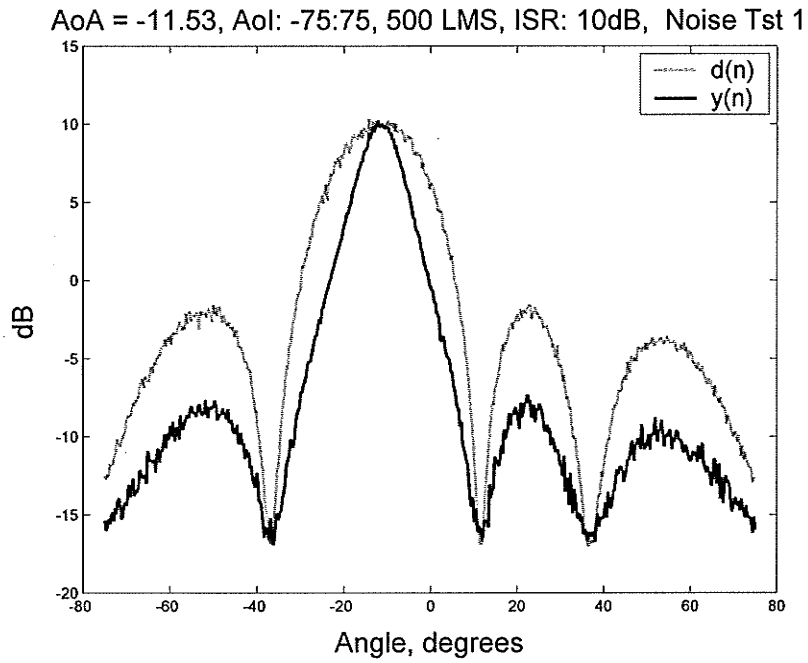


a) Baseband noise interference.

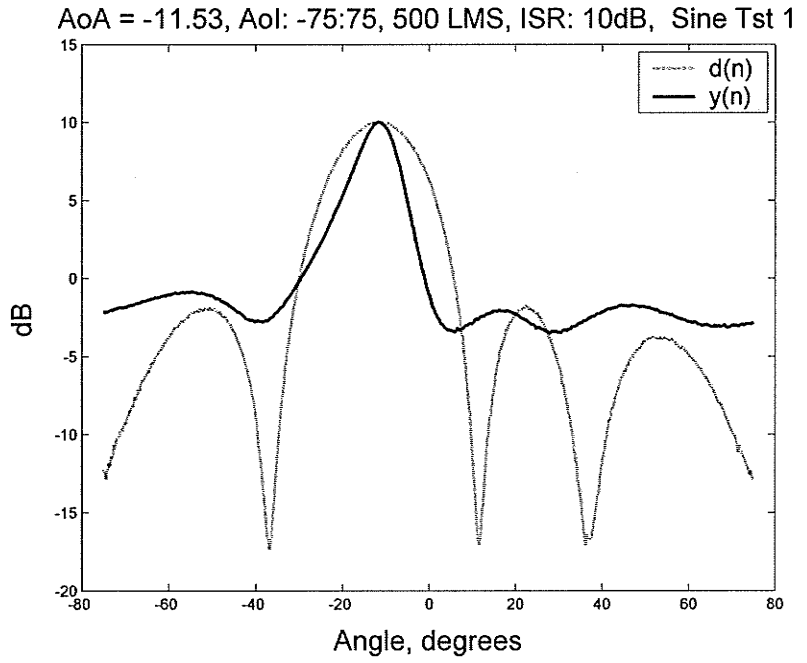


b) Sinusoidal interference.

Figure 4.12– Mean squared error performance of quiescent output *d(n)* and LMS adapted output *y(n)* as interference power level varies for -30dB ≤ ISR ≤ 50dB for (AoA,AoI) = (-11.54°,23°).



a) Baseband noise interference.



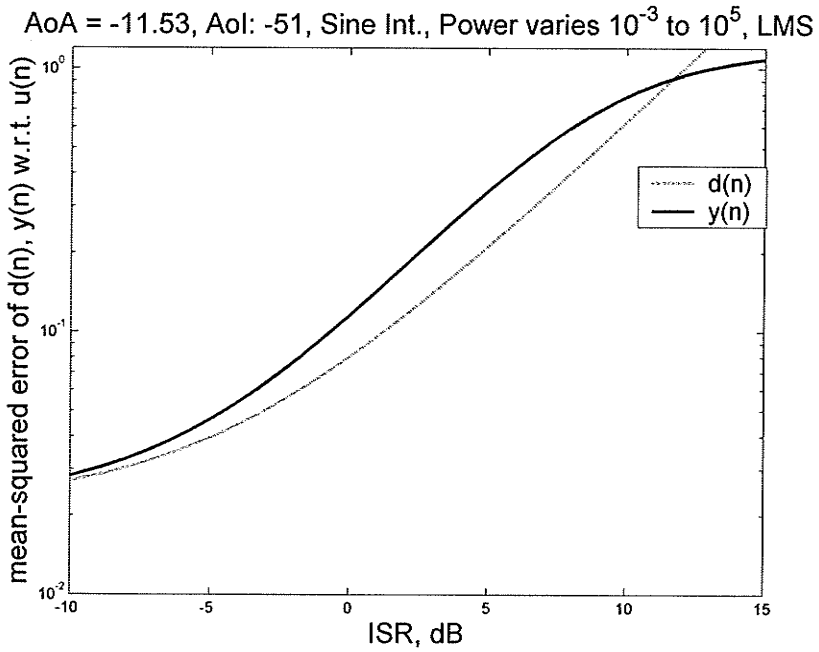
b) Sinusoidal interference.

Figure 4.13– LMS performance for AoA = -11.537°, -75° ≤ AoI ≤ 75°. ISR = SNR = 10dB.

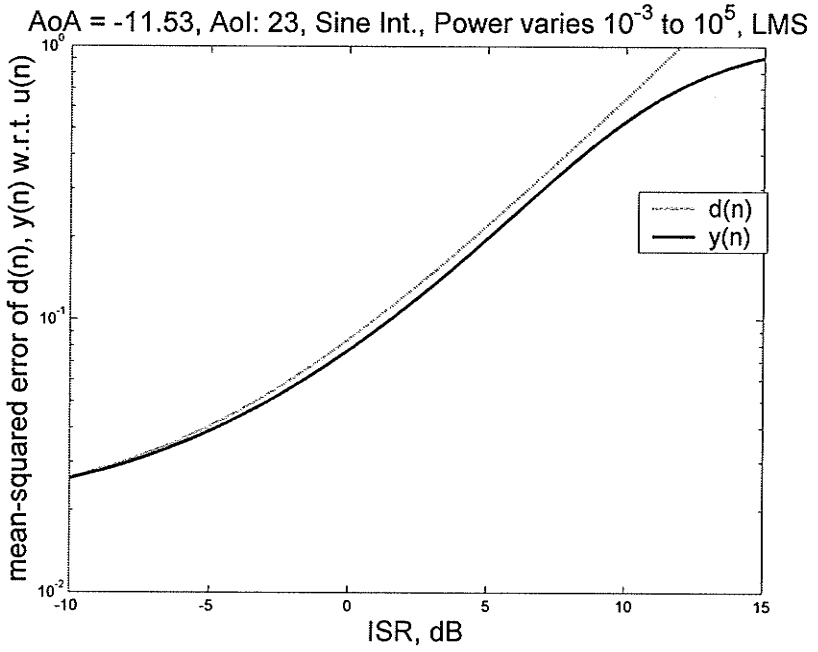
A few key points can now be summarized. The LMS algorithm performs well for noise interference and coherent sinusoidal interference above a minimum ISR. The striking observation in general is that the mean squared error of the adapted output reaches a threshold and remains constant as the ISR increases, up to a certain ISR level. At that peak ISR level something in the algorithm breaks down, rendering the output useless. This is apparent by the MSE curve for $y(n)$ going vertical in Figure 4.11 and Figure 4.12.

This threshold of the mean squared error of the adapted output indicates that as the interference gets" louder" the null gets deeper. This behavior is illustrated graphically in the array factor directive gain patterns of Figure 4.15.

It is reasonable to observe the improvement in performance by looking at the variance, MSE or directive gain plots and still wonder what is the effect on the time signal itself. To look at this, the sample frequency was increased to $360f$ and the simulation was run for 3 cycles of $f \approx 300$MHz at ISR = 10dB and ISR = 20dB. The output is shown in Figure 4.16. Note that the transient settling time is noticeable at the beginning of the output. It should be realized that the transient settling time is related to the number of iterations, which would usually span several cycles of f because the sampling frequency is usually $2f$ (Nyquist) or greater. A test sample frequency of $360f$ was used to produce an output clearly showing the sinusoidal nature by having a point per degree.



a) Sinusoidal interference. Zoom of Figure 4.11(b).



b) Sinusoidal interference. Zoom of Figure 4.12(b)

Figure 4.14– Close-up of low ISR regions in Figure 4.11 and Figure 4.12.

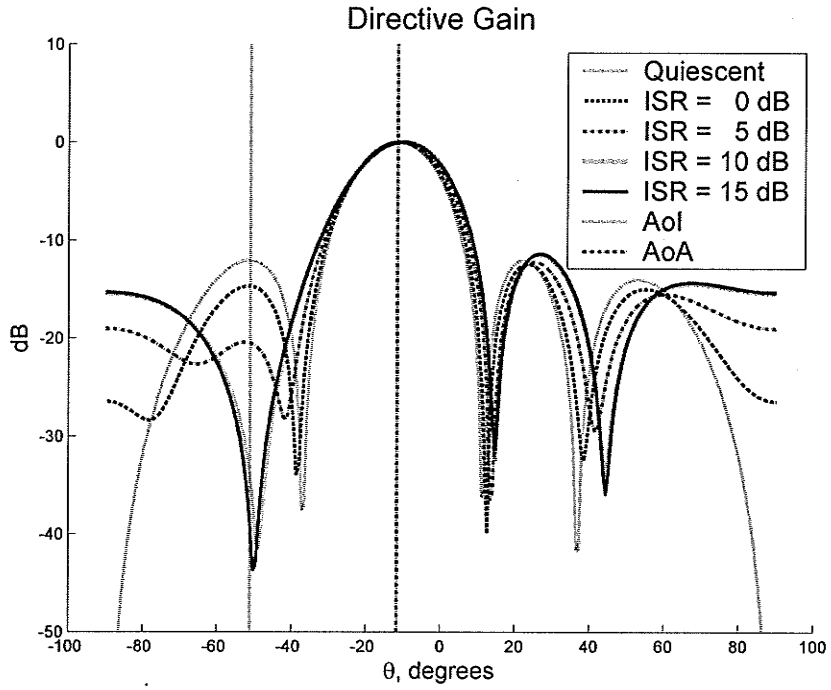
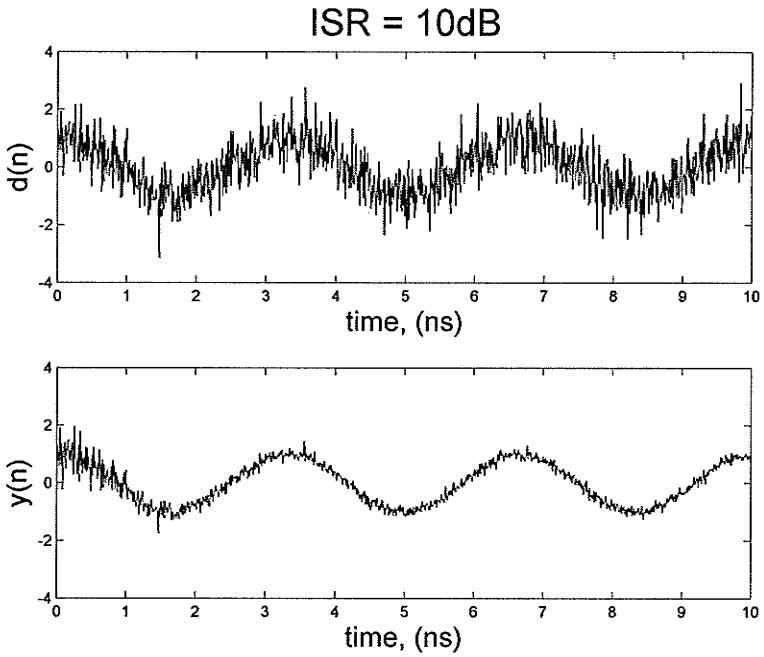


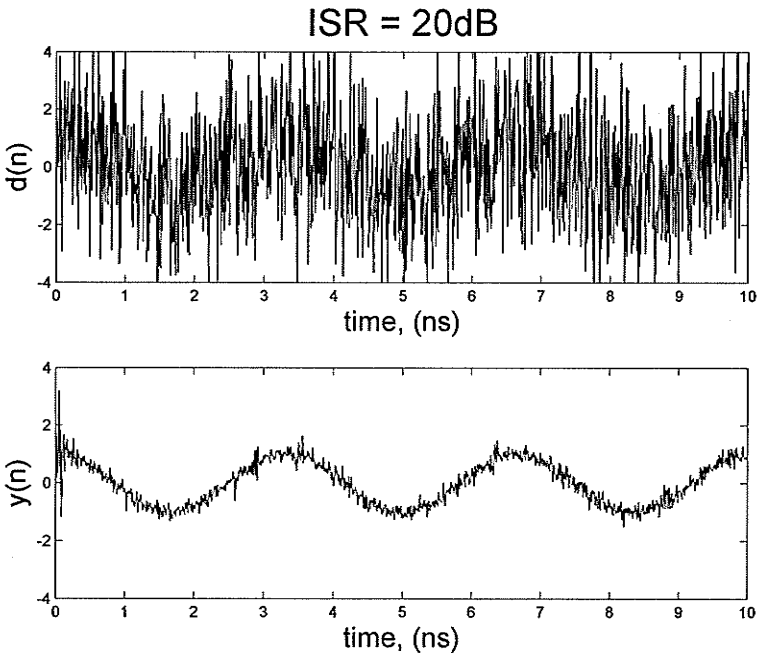
Figure 4.15– Array factor directive gain patterns for LMS method as ISR increases.
(AoA,AoI) = (-11.54°,-51°), μ = 1e-4, 440 iterations.

The plots of Figure 4.16 indicate very good performance and demonstrate the LMS algorithm' s ability to output adapted signals with comparable mean squared error over a range of ISR, as compared to the original noiseless signal *u(n)*. It should be kept in mind that the signal at the array includes terminal noise of 10dB SNR.

Because the random terminal noise vector is calculated and saved in the simulation, the



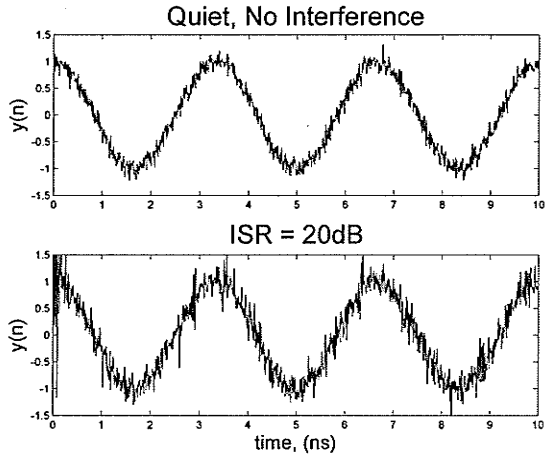
a)



b)

Figure 4.16– Three cycles of time output for *d(n)* and *y(n)* given a) ISR = 10dB and b) ISR = 20dB.

exact test can be re-run without the presence of interference. Thus, we can look at both the quiescent and adapted output of the array in either a quiet environment or one with interference, as is shown by the array output and table of variances of signals in Figure 4.17. A quiet environment is one with no interference.



a)

Table 4.1 - Variance of Array Signals.

Var. of:	***u(n)***	**Output due to $u_A(n)$**	***d(n)***	***y(n)***
ISR = 20dB	1.0009	1.0194	7.2703	1.1131
Quiet	1.0009	1.0153	1.0153	1.0153

b)

Figure 4.17– a) Array output of original with and without 20dB of interference. b) Table of variances.

In Table 4.1, the output of the array due to $u_A(n)$ is the weight vector applied to the original signal with only terminal noise, according to (3.17). The output of the array due to $d(n)$ is the quiescent weight vector applied to the total signal received by the array, including interference. Note that they are the same in a quiet environment. A plot of the output of the array due to $u_A(n)$ using the adapted weight vector for 20dB ISR would be slightly noisier than that shown in Figure 4.17(a) for the quiet environment.

Finally, the performance of the LMS method over a random selection of 5000 (AoA,AoI) points was performed for ISR = 30dB and SNR = 10dB, as was done for the LCMV method.

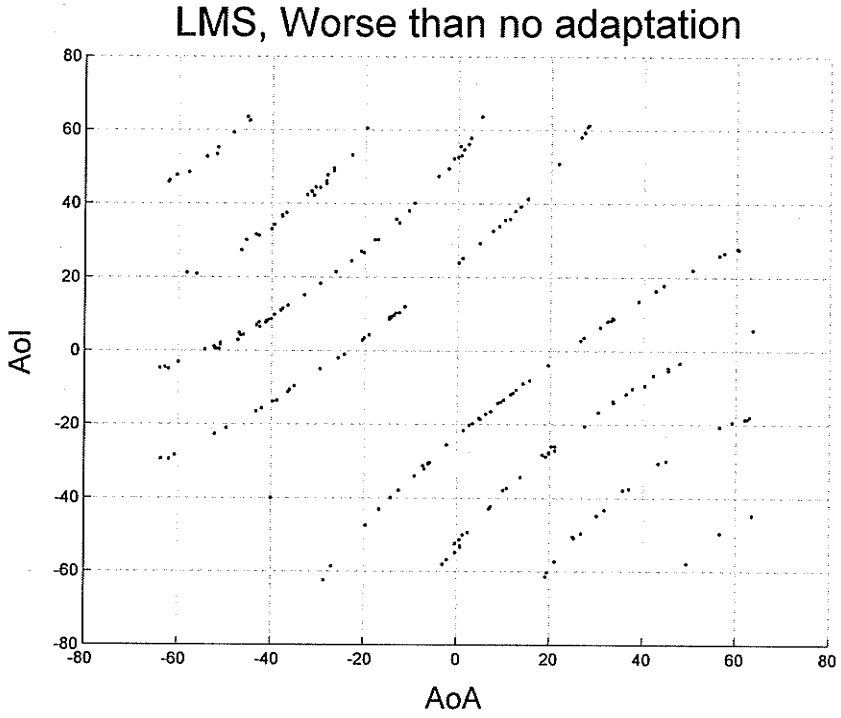
Except for one random point where the AoA and AoI were at the same place (-39.94°,-39.952°), the only places where the LMS performance did not exceed the quiescent performance is where the AoI was already deep in a null over 30dB down. This is shown in Figure 4.18(a), where as before, better performance means that the array factor gain at the AoI is lower for the adapted weight vector than for the quiescent weight vector.

The small table of values in Table 4.2 shows that where the quiescent pattern's gain is already 30 to 80 dB down and lower than the adapted gain, the adapted pattern's gain is limited to 33.4 dB down. This limiting effect can be seen in Figure 4.18(b) for all points, where the points pictured in Figure 4.18(a) appear to the upper-right of the diagonal.

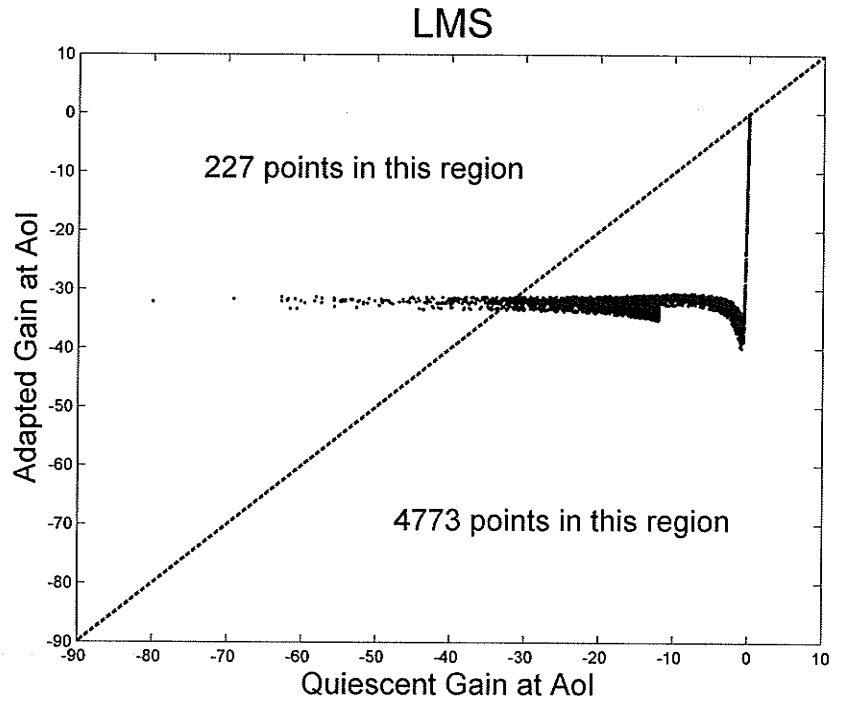
Table 4.2 - Gain statistics for points where the quiescent pattern is already in a null 30dB down and $G_y > G_d$.

Stats	**Gain, d(n)**	**Gain, y(n)**	$\mathbf{G_y - G_d}$
max	-31.24	-31.227	47.917
min	-80.005	-33.418	0.012742

This limiting effect should be studied to determine why it is occurring. Perhaps it is the result of gradient noise, causing the LMS algorithm to oscillate around a best solution 30dB down. However, this has not been studied nor have the effects of varying μ or ISR.



a) Performance worse than quiescent.



b) Gain after LMS vs. before LMS.

Figure 4.18– Scatter plots of LMS performance. ISR = 30dB and SNR = 10dB.

To summarize this section of LMS performance tests, it has been shown that the algorithm works very well in a wide range of conditions. The LMS section of the previous chapter showed the relationship that increasing the adaptive scaling factor μ increased the rate of convergence. This section has shown that increasing the interference power has the effect of deepening the null at that AoI. In addition, sinusoidal interference, although difficult to cancel at low power levels, can effectively be nulled if it is above a minimum power level.

4.3 A Comparison of LCMV and LMS Results

This section will briefly compare and discuss the performance results of the LCMV and LMS methods. The first point to note is that in the absence of noise and interference, the LMS method returns the results of the quiescent weights, whereas the LCMV method requires some terminal noise to avoid a singular matrix inversion. Even with only terminal noise, the LMS method converges to the quiescent weight vector exactly, as shown in Figure 3.1(b) and in Table 4.2 for the quiet environment. However, Figure 3.1(a) shows that the LCMV method tends to produce a more uniform weight vector without interference as indicated by the lowered standard deviation, but still with much variation.

It was shown that for the (AoA,AoI) regions where the LCMV worked well, it worked very well. It produces an instantaneous weight vector that nulls the effects of interference immediately, with no convergence criteria. Figure 4.5 shows that the LCMV method provides a constant ratio of suppression right up to the test limit of 50dB ISR, based on the MSE of the output, whereas Figure 4.11 indicates that the LMS method provides and

increasing ratio of suppression. It shows that there is a range of constant MSE for increasing ISR, but only up to a limit of about 33dB after which the performance degrades entirely.

The definitive performance test, however, is seen in the results of the 5000 random (AoA,AoI) pairs, shown in Figure 4.9 for the LCMV and Figure 4.18 for the LMS, where better performance means that the array factor gain at the AoI is lower for the adapted weight vector than for the quiescent weight vector.

From this test we find that the LMS algorithm performed extremely well, lowering the gain at the AoI for all angles except a few which were already deep in a null. The curious behavior is that the adapted null is limited to about -35dB. The LCMV method, on the other hand, showed dramatically poor performance on average. Over half of the angles tested returned an adapted gain at the AoI higher than that of the quiescent. Nevertheless, where it works, there is no limit on the suppression and indeed we see that in some places the LCMV method reduced the gain at the AoI from within a few dB of peak to over -60dB.

For both of these anomalies, the -35dB suppression limit for the LMS and the patchwork pattern of better/worse performance of the LCMV, no explanation is readily apparent. However, the conclusion to be drawn is that the best algorithm for the job would be the LMS method.

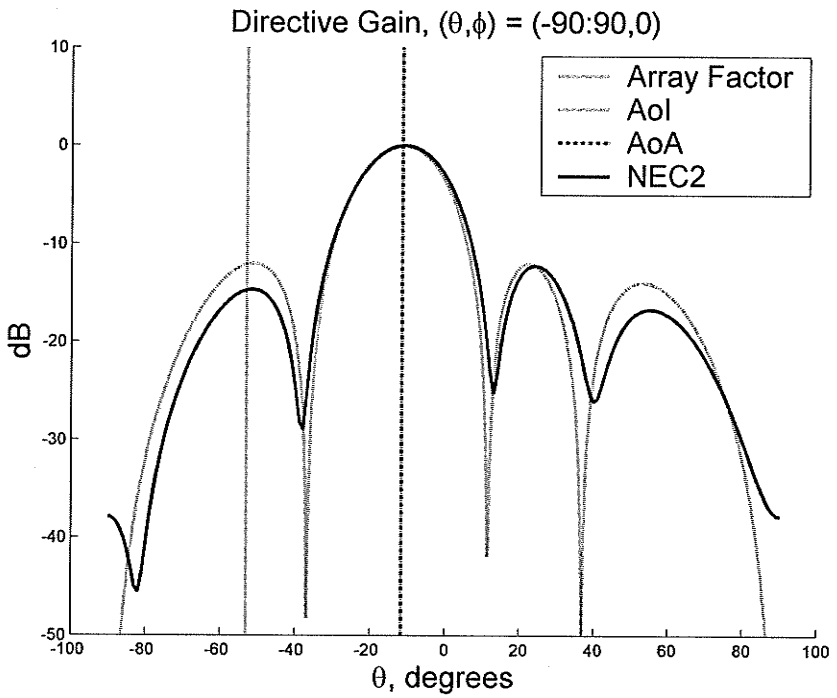
This conclusion is supported by the fact that the LMS method is widely used due to its simplicity, ease of implementation and excellent performance given a judicious choice of scaling factor μ.

The LCMV is a direct application of the Wiener Linear Optimal Filter, but suffers from relying on ensemble statistical parameters being estimated by small discrete samples. In essence, the LCMV algorithm presents the basis for the optimum instantaneous choice of adaptive weighting for generalized sidelobe cancellation, and the LMS method iteratively approaches that solution in a robust fashion.

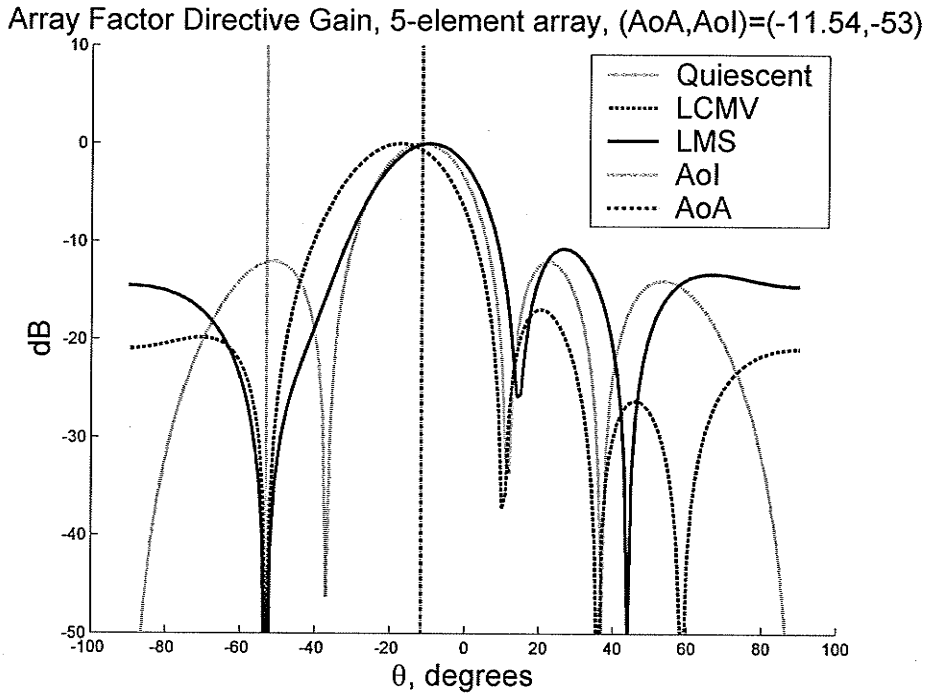
5.0 Using the LCMV and LMS Output with NEC2/Matlab

Having developed and tested the NEC2/Matlab interface, and further developed, tested and verified the performance of the LCMV and LMS methods, this chapter looks at the full-wave output of NEC2 in comparison to that of the array factor for selected adapted weight vectors. The purpose of this is to see how close the gain patterns of the adapted output match for the array factor and the full-wave NEC output, and from that observe the effects of mutual coupling interactions.

Given that the LCMV has some known limitations, the first example will be for a configuration previously shown to work for both the LCMV and LMS algorithms. Consider a 5-element array aimed at AoA = -11.537° with 20dB ISR interference coming in at AoI = -53°, as was shown in Figure 3.6 and for which the quiescent array factor and NEC2 patterns are shown in Figure 5.1(a). The best-case array factor directive gain patterns for the adapted results of both the LCMV and LMS are shown in Figure 5.1(b). Note how the LCMV method causes a shift of mainbeam while the LMS method tries to preserve the gain at the AoA, for the" best case" weight vector.



a) Quiescent array factor and NEC for AoA = -11.537°.

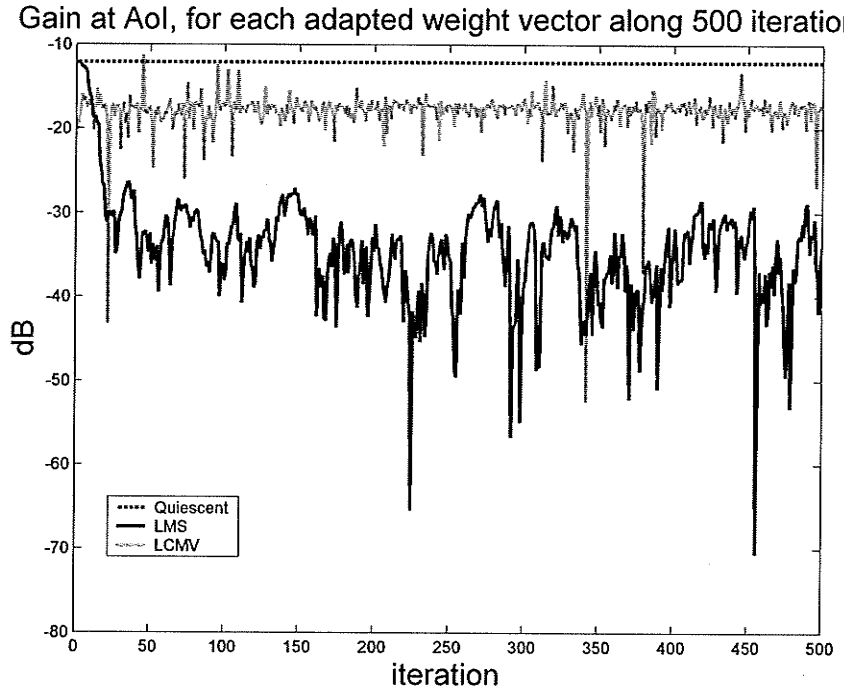


b) Adapted array factor for (AoA,AoI) = (-11.537°,-53°).

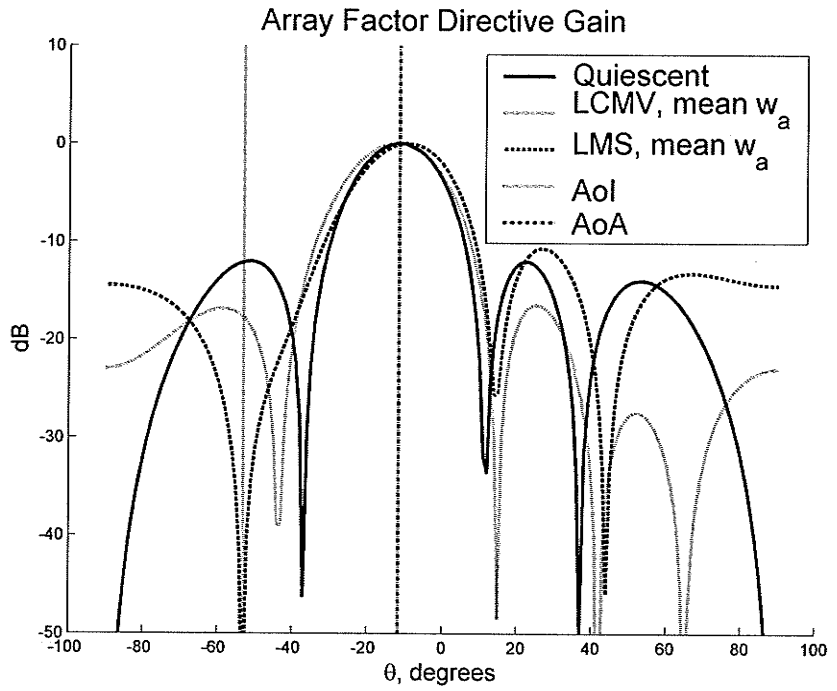
Figure 5.1– 5-element array with (AoA,AoI) = (-11.537°,-53°).
a) Quiescent directivity pattern for Array Factor and NEC output.
b) Directivity pattern for Array Factor adapted output of LCMV and LMS methods using best $\mathbf{w}_a$.

In Figure 5.1(b), the" best case" adapted weight vectors were used to generate the results. This requires some explanation. The simulation was run for 500 iterations and at each iteration the LCMV method comes up with an instantaneous adaptive portion of the weight vector while the LMS method converges towards a best solution. Once this 500-element $\mathbf{w}_a$ vector is obtained for each of the methods, it is a simple calculation to determine the gain at the AoI for each iteration using (3.14) and then (3.6) with the steering vector calculated at the electrical angle of the AoI. The array factor gain at the AoI for each iteration of the adapted weight vector is shown in Figure 5.2(a). The lowest value is chosen as the" best case" , as the intention is to reduce the gain at the AoI. The array factor directive gain pattern based on the average of $\mathbf{w}_a$, calculated over the 500 iterations, shown in Figure 5.2(b).

With a quiescent gain of -12.1dB at the AoI, the average gain of the LCMV method is -17.9dB and of the LMS method is -34.5dB. The best cases produce a gain of -52.45dB at the 342nd iteration for the LCMV and -70.6 at the 456th iteration for the LMS. Note how the 3dB beamwidth is wider for the LCMV method. Looking at Figure 5.1(b) and Figure 5.2(b), it can be concluded that the mean $\mathbf{w}_a$ for the LMS method produces a pattern very similar to the best case, and so implies that the method is very consistent.



a) Array factor gain at AoI for each iteration.



b) Directivity of array factor based on mean $\mathbf{w}_a$.

Figure 5.2– **a)** Directive gain at AoI at each iteration and **b)** array factor pattern based on mean value of $\mathbf{w}_a$. (AoA,AoI) = (-11.537°,-53°), ISR = 20dB, SNR = 10dB.

The patterns for the LCMV method, however, show dramatic difference between average and best performance. In particular, note how the best performance produces the null at the desired location, but shifts the mainbeam considerably. In contrast, the average performance does not shift the mainbeam, but also does not deliver a deep null to the AoI. The adapted null is closer to the mainbeam. This may imply that some of the difficulties leading to the overall poor performance of the LCMV are due to the conflicting constraints of trying to maintain the gain at the AoA while steering a null to the AoI.

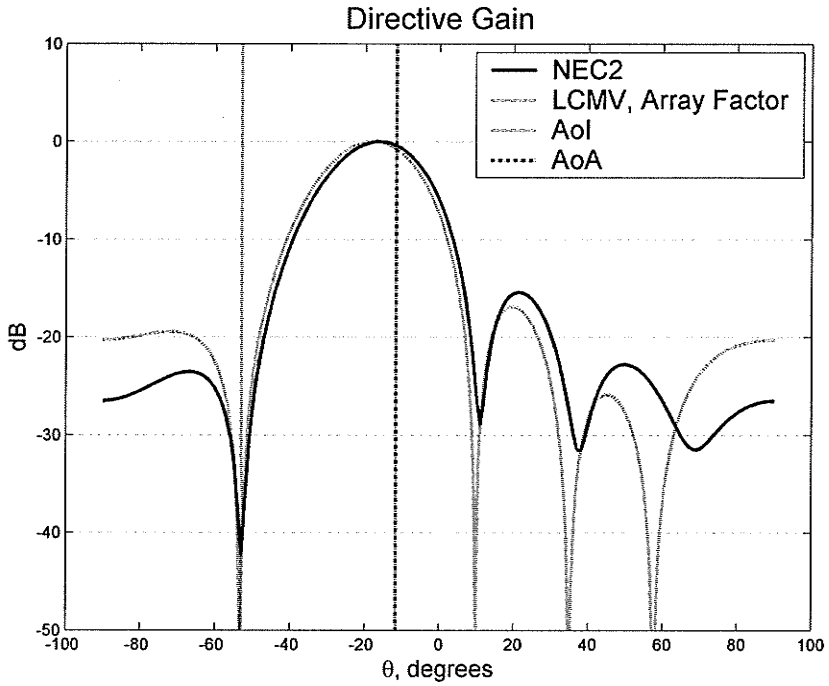
The full-wave solution of NEC, based on the charges and currents on the modeled radiating structure, produce results that include the interactions between these sources and structures in the environment. The fields, based upon the electric and magnetic field integral equations as discussed in the chapter on NEC, are computed using the current distribution as sources. However, the modeling of the structure requires voltage sources to be attached to a node. If the elements' input impedances were constant across the array, the relationship between current sources and voltage sources would just be the impedance, a complex constant. However, as was seen, the element impedances vary due to mutual coupling and are a function of source values and scan angle.

When using array factor, it doesn't matter whether one is assigning the source as a voltage or a current, as impedance is not a consideration. Truly, the weight vector alone can be applied as the source and the directive gain pattern will be as given by the array factor. In a physical array, the complex weight is usually implemented using an amplifier or attenuator and a phase shifter. The specific values of the sources don't affect the radiation patterns (but do affect the radiated power); it is the distribution of the magnitude

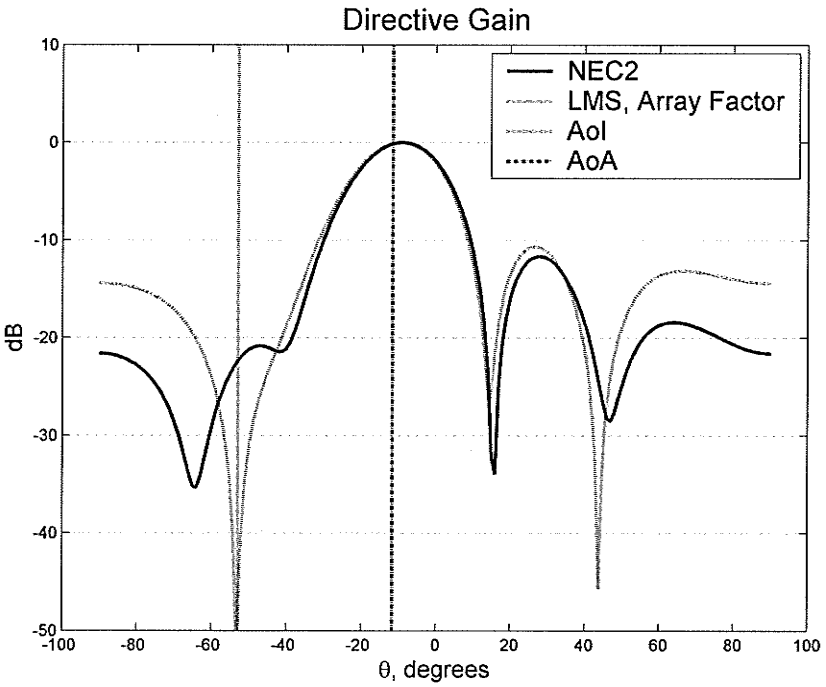
of the sources and the phase relationship between them that determines the radiation pattern. Thus, the calculated weight vectors can be applied to NEC as the voltage sources, normalized by magnitude for consistency, and the resultant full-wave solution can be computed. However, due to the mutual coupling intrinsic in a real situation, which the full-wave computation tries to model, applying the weight vector as a voltage source will not necessarily yield the same output as applying the weight vector as a current source.

This idea of voltage source versus current source can be seen in Figure 5.3, the results of which are somewhat surprising. The NEC output matches the array factor for the LCMV distribution very well, but indicates much inferior performance from the LMS method than expected from the array factor. The quiescent gain at the sidelobe AoI is -12.2dB, and the results of NEC2 show it reduced to -22.4dB rather than the -70dB of the array factor. The best case of the LCMV has reduced the gain at the AoI to -41.8dB, but this cannot be expected as typical behavior. From Figure 5.2(b), -17.7dB is the average gain at the AoI for the LCMV, and it must be kept in mind that the performance is very inconsistent, at least as formulated in this work.

To understand the inconsistent behavior of the LMS method in the NEC2 simulation, as shown in Figure 5.3(b), consider the comparison of V_{in} and Z_{in} for the LCMV and LMS solutions, as shown in Figure 5.4. Here it is observed that although the weight vector (voltage source) is quite consistent between the two methods, the impedance shows more variation, particularly at the null-side of the array, which shows the greatest difference in phase shift. Note the increased slope of the phase of the voltage source for the LCMV method, which relates to the shifting of the mainbeam towards the null.

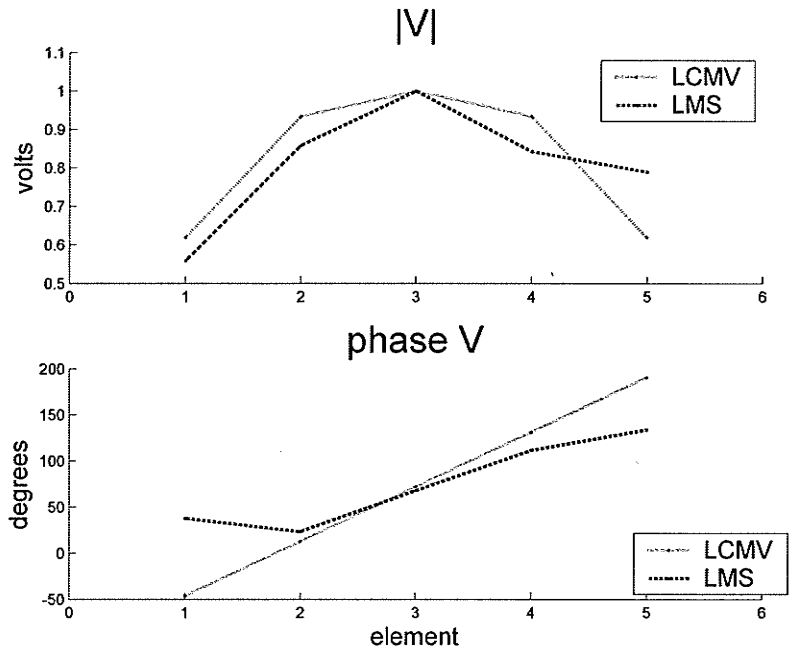


a) LCMV

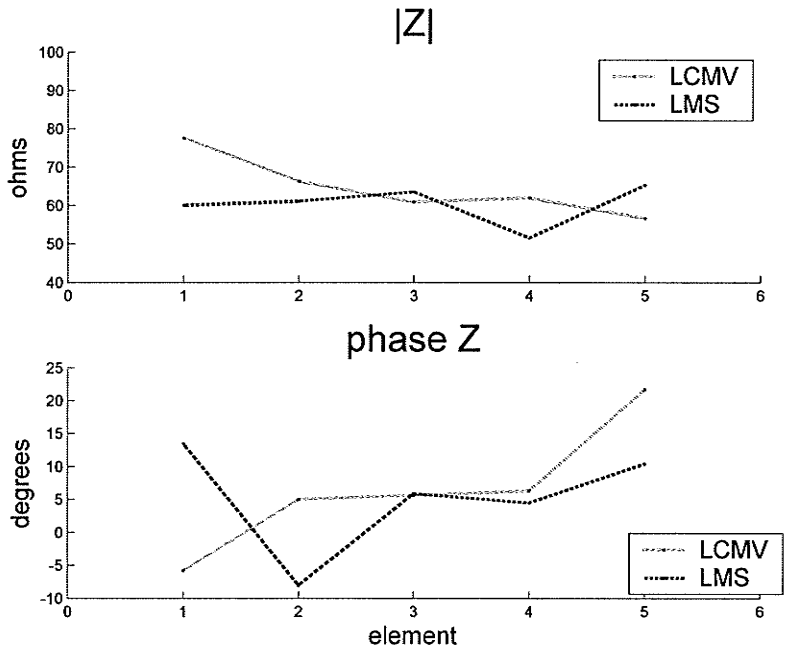


b) LMS

Figure 5.3– NEC2 and Array Factor adapted output using" best case" weights for (AoA,AoI) = (-11.537°,-53°).



a) Applied input voltage, or adapted weight vector.



b) NEC2 computed element impedances.

Figure 5.4– Input voltage (weight) and input impedance comparison for (AoA,AoI) = (-11.537°,-53°).

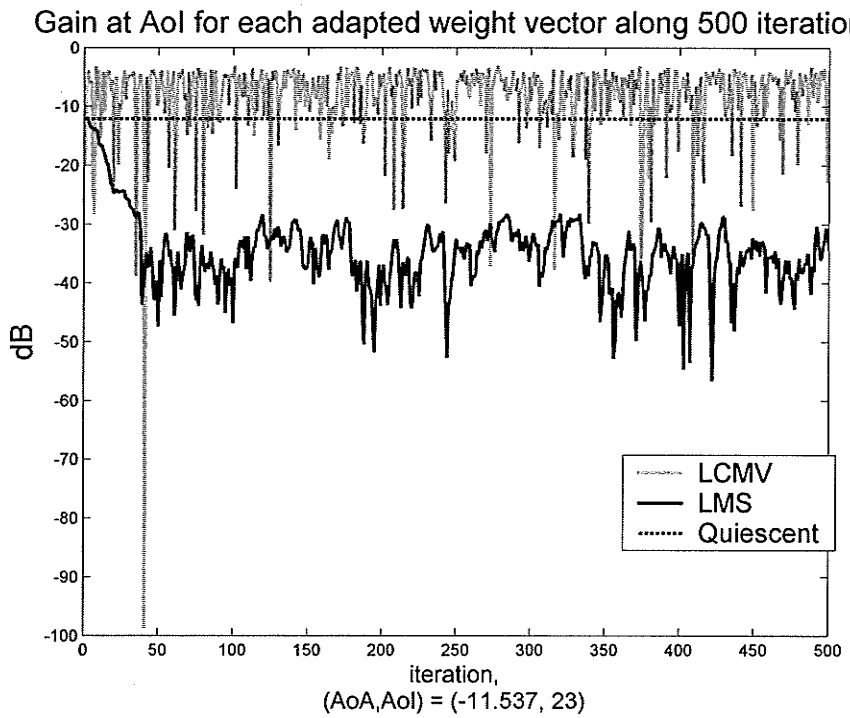
Interestingly, over a set of simulations at (AoA,AoI) = (-11.537°,-53°), all of the best case patterns were similar to that of Figure 5.3, but in some cases the magnitudes of the voltages and impedances were identical for the two methods. Even in these cases, the phase behavior was markedly different, leading to the patterns shown.

For the next example, consider cancellation of the opposite sidelobe, at $\theta = 23°$. Recall that the array factors were previously studied at this AoI, showing that the LCMV method did not perform well but that the LMS method produced good results.

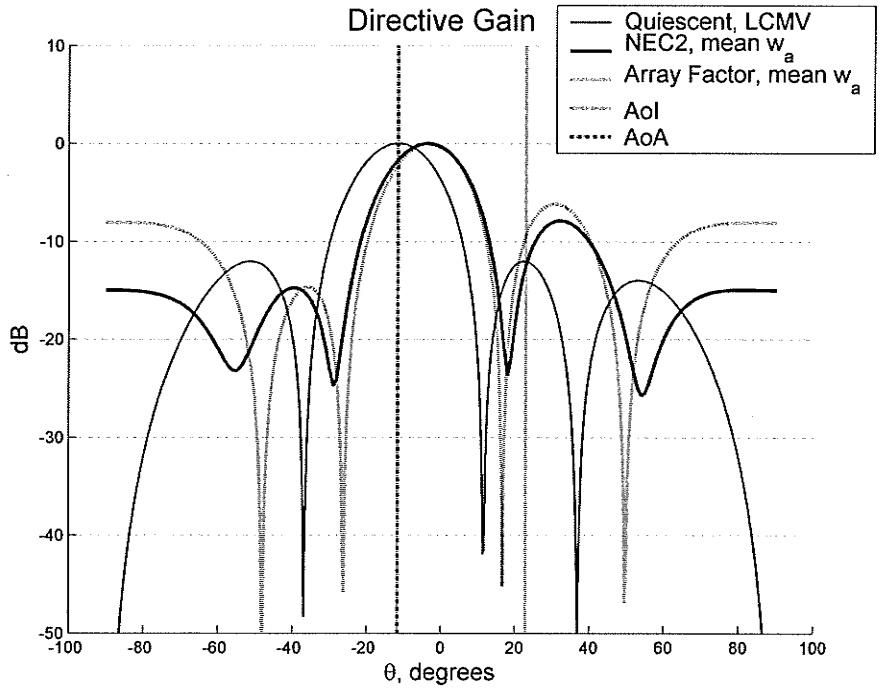
Figure 5.5(a) shows that the quiescent gain is -12.1dB at the sidelobe. The mean gain of the LCMV method is -8.7dB and of the LMS method is -34.8dB, indicating that the LCMV method performs worse than quiescent on average. The best cases produce a gain of -98.6dB at the 41st iteration for the LCMV and -56.5 at the 422nd iteration for the LMS, for this simulation. Due to the random nature of noise and interference, it changes with each simulation.

Figure 5.5(b) shows the quiescent, LCMV adapted and NEC2 gain patterns computed using the mean of the adaptive portion of the weight vector, $\mathbf{w}_a$. Here we see that the adapted gain is higher than quiescent at the AoI, but the NEC2 pattern is slightly lower, showing the same gain as quiescent.

The array patterns for the best case adapted weights are given in Figure 5.6, showing good agreement between NEC2 and the array factor for both methods. The typical effect of mutual coupling in displacing the nulls outwards can be seen.



a) Array factor gain at AoI for each iteration.



b) Array factor and NEC patterns based on mean $\mathbf{w}_a$.

Figure 5.5– **a)** Directive gain at AoI at each iteration and **b)** LCMV array factor and NEC pattern based on mean value of $\mathbf{w}_a$, still showing poor performance at this AoI. (AoA,AoI) = (-11.537°,23°), ISR = 20dB, SNR = 10dB.

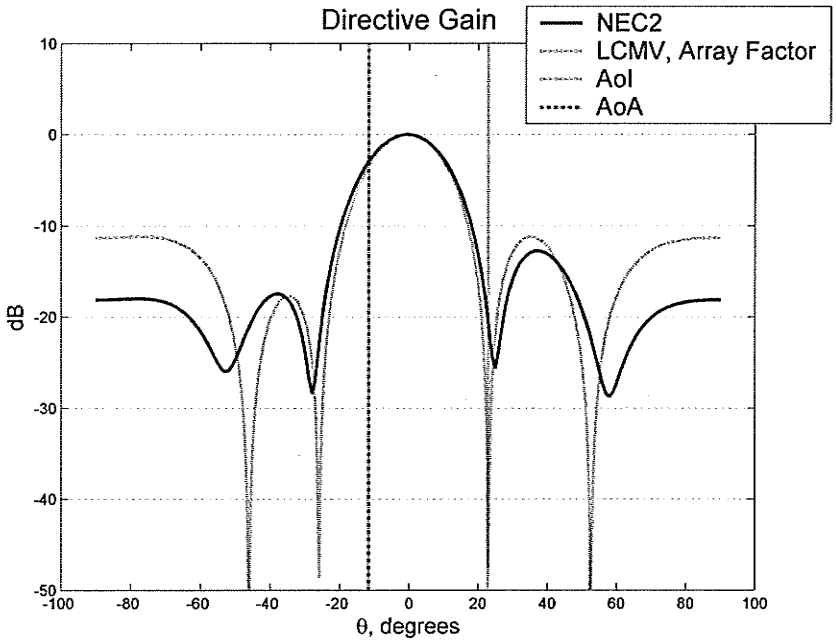
In Figure 5.6, notice that both algorithms are performing well with the" best case" weight vector. In particular, note that the LMS algorithm puts more emphasis on maintaining the gain at the scan angle. The LCMV method has shifted the mainbeam almost 11° toward the adapted null, causing a -2.8dB drop in gain at the desired AoA.

The pattern for the LMS algorithm calculated using the mean of $\mathbf{w}_a$ over the 500 iterations appears almost identical to that in Figure 5.6(b), and so is not shown separately. This again gives evidence to the consistent good performance, on average, of the LMS method. In contrast, look at the differences seen between the" best case" and mean of $\mathbf{w}_a$ for the LCMV method as shown in Figure 5.5(b) and Figure 5.6(a). As stated previously, this is evidence of the poor performance, on average, of the LCMV method.

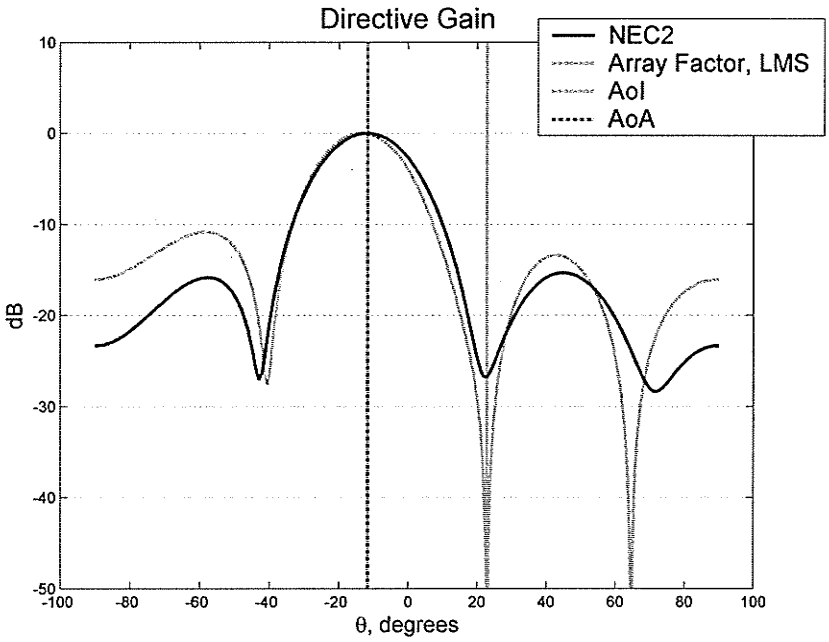
The corresponding voltage and impedance behavior is shown in Figure 5.7, where the leveling out of the source phase (input voltage or weight vector) indicates that the mainbeam has been shifted considerably from -11/54° towards 0°.

The taper of the magnitude of the source voltage is typical of non-uniform amplitude source distribution for sidelobe level suppression, such as with a cosine distribution [7]. Thus, it is not surprising to see the general effect of lowered sidelobes in Figure 5.6.

It is important to note the scale of the ordinate axes in Figure 5.7(b), as it appears that there is quite a variation in the phase of the impedance. The scales of the axes in Figure 5.4 and Figure 5.7have been kept the same for comparison. Since it is the current distribution that governs the radiation equations, Figure 5.8 looks at the source current and the input impedances with a larger scale of the ordinate axis.

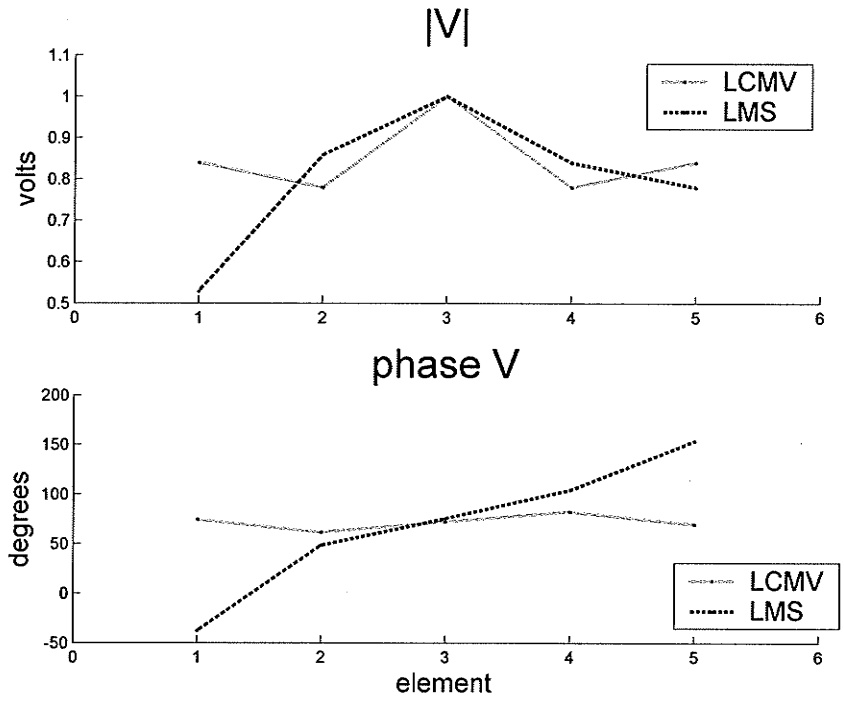


a) LCMV

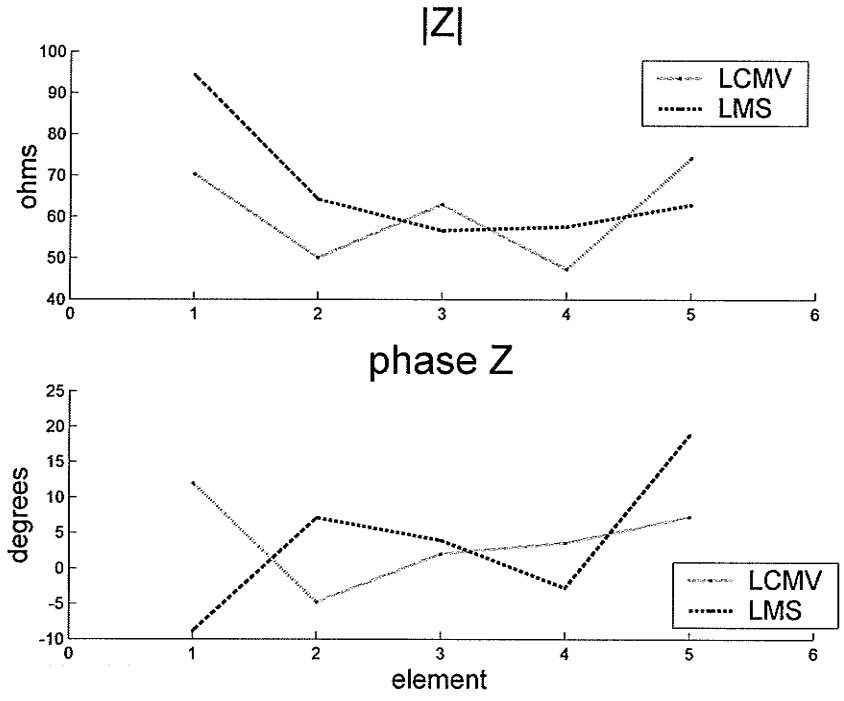


b) LMS

Figure 5.6 – NEC2 and Array Factor adapted output using" best case" weights for (AoA,AoI) = (-11.537°,23°).



a) Applied input voltage, or adapted weight vector.



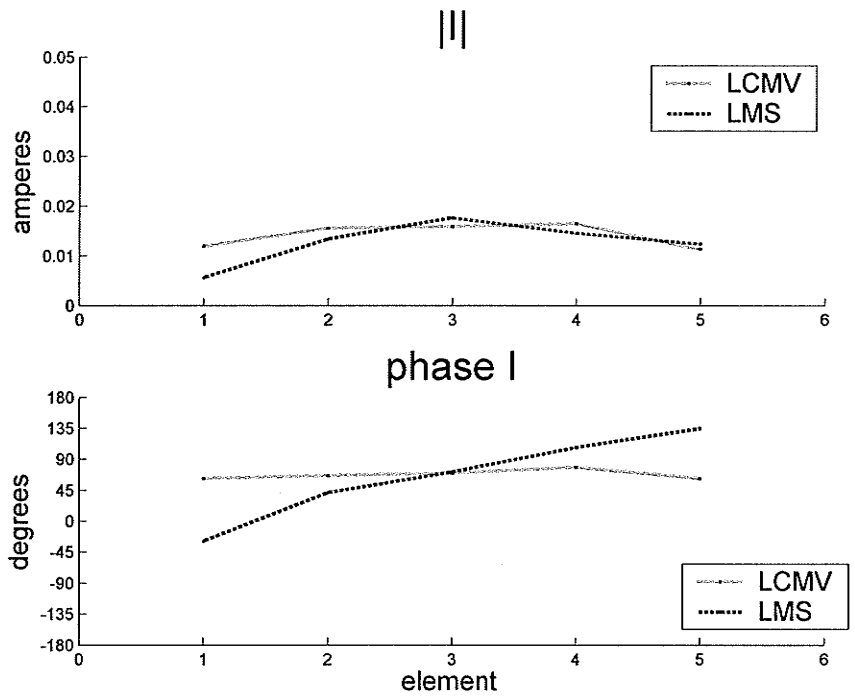
b) NEC2 computed element impedances.

Figure 5.7– Input voltage (weight) and input impedance comparison for (AoA,AoI) = (-11.537°,23°).

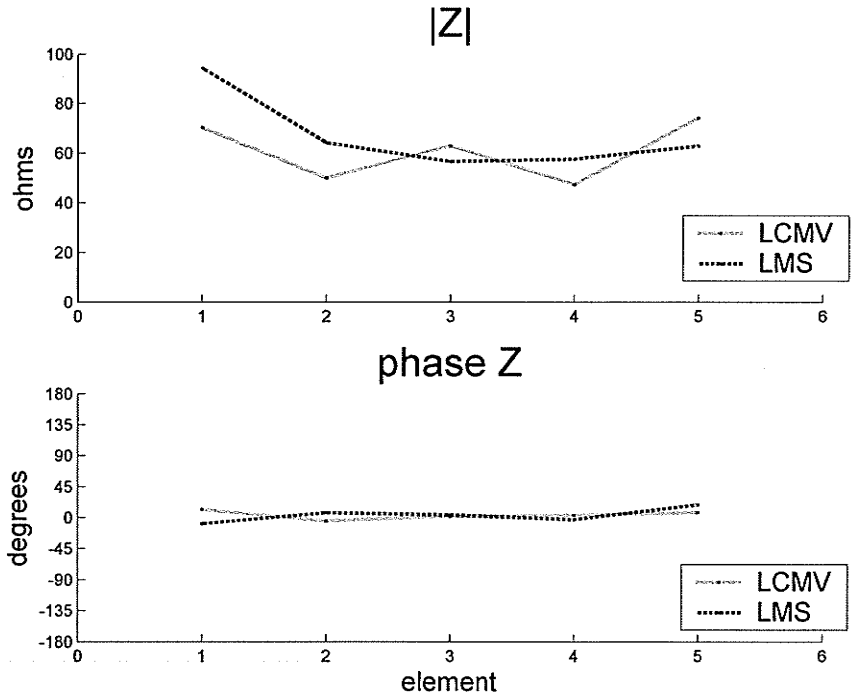
From this, we see the taper of the source current, which leads to sidelobe suppression, and the current phase angle, which relates to the scan angle. In the impedance plot, one notices that the phase is very consistently near zero, implying that the element impedances are very close to being purely real and that the array is operating close to its optimum resonant frequency. The distribution of the impedance amplitudes shows a relatively constant value, with magnitude increasing at the scanned end of the array. This behavior is consistent with what was observed previously during the element impedance tests as a function of scan angle.

In summary of this chapter, it can be concluded that although the array factor can give a good approximation to the behavior of an antenna array, the true results may be quite different than expected due to the source and radiated field interactions in the structure. Clearly, the output of NEC2 has not been compared with physically measured results from an actual array in an antenna test site, so a definite comment on the" true" results cannot be concluded. It is reasonable and generally accepted, though, that a full-wave computation that considers the interactions of mutual coupling is much closer to true behavior than the geometric interference calculations of the array factor.

This work has combined the determined best-adapted weight vector, computed using the LCMV and LMS algorithms, with the input-file driven NEC2 for full-wave computations to determine the simulated output. To automate the process, NEC2 has been controlled by the Matlab interface. However, the LCMV and LMS algorithms calculate the adapted weight vector in conjunction with array factor (steering vector) closed-form analytical expressions.



a) Input current.



b) Impedance.

Figure 5.8– Input current and element impedances.

The obvious next step is to use NEC2 in receive mode to calculate the response at the antenna elements due to the incident plane waves of the desired and interference signals, and subsequently add AWGN to the element' s signals. Then, use these NEC2-computed received signals to feed into the LCMV or LMS optimization algorithms.

By doing this, the degraded results for the LMS method, seen in Figure 5.3 may be overcome. The difficulty of this is that the NEC2 solver can only account for one incident plane wave at a time, so the results would have to be stored and summed. Any mutual coupling interactions induced due to the signals arriving at the array at the same time would not be included. This study has been left for future work.

6.0 Conclusion

Summary

This thesis has presented work in three broad areas - the use of full-wave numerical electromagnetic computations, via NEC2, in the study of antenna array interactions, the interfacing of a compiled command-line driven program based on text file I/O with the powerful mathematical and visualization environment of Matlab, and most importantly the development and testing of adaptive signal processing techniques as applied to adaptive array antennas and the application of the NEC2/Matlab interface to study this field.

Upon introducing the NEC2 program, the interface with Matlab was developed, tested and verified to work. Although the inerface is command-line driven from withing Matlab, it is only a drawback in that it is not a fully integrated graphical user interface. The Matlab environment does not have the restrictions encountered with a usual operating system' s command line. From the Matlab environment, all modules of the program can quickly be linked or detached, modified and run in an unattended and evolving process. That is, the interface automatically composes the NEC2 input files, runs the solver, imports the output data, evaluates it (either automatically according to defined rules or via a pause with user intervention) and then re-composes input files to continue the optimization process.

The theory of adaptive array antenna signal processing has been introduced and studied in relation to the generalized sidelobe canceller using the linearly constrained minimum variance (LCMV) method and the least mean squares (LMS) method. The LCMV

method is a direct application of Wiener linear optimum filtering as applied to the spatial analogy of an antenna array as a finite impulse response transversal filter. Its basis lies in the correlation processing of the signals at the taps (antenna terminals) of the array. It was found to only work consistently well in certain situations and other than the potential cause being that the statistics of a random ensemble are being estimated by one small sample, a more concrete explanation has remained elusive. The LMS method is from the class of stochastic gradient search algorithms, and has been shown to work extremely well in a wide variety of situations. The user need only chose the best scaling factor for rapid but accurate convergence and then the algorithm will iterate towards the Wiener optimal solution. Once again, the statistics of the process are being estimated from the sample and this leads to a phenomenon known as gradient noise. It has been inconclusively speculated in this work that the gradient noise may be the reason for the apparent nulling limit of -35dB.

When the NEC2/Matlab interface is run with the adapted weights as calculated by the LCMV and LMS algorithms, the typical mutual coupling effects of beam broadening, null shifting and the rise of a coma lobe during scan are seen. In addition, although the array factor computed weight vectors indicate deep nulling, the full-wave results may be quite different.

Finally, in both classic and adaptive array analysis, the NEC2/Matlab interface can be used to quickly determine the effects of element length, diameter and spacing on the overall impedance behavior of the array during scan. It was shown how this analysis, combined with reflection coefficient calculations, can assist the user in addressing

phenomena such as scan blindness from feed network to antenna element impedance mismatch.

Future Work

The development of static beamforming using the constraint matrix gives rise to the discussion that perhaps sub-space methods of determining AoI in conjunction with calculating the beamform via the constraint matrix may rival the LMS and LCMV algorithms in speed and performance. The static beamforming was shown to be very effective in placing deep nulls and other main lobes in desired locations as established in the constraint matrix. Target and interference location methods could be used in conjunction with this constraint matrix to iteratively update the best fixed weight vector.

To examine the impedance behavior as developed in Chapter 2.4, the results were taken from the NEC2 output files generated at different frequencies. The impedance matrix of the array could be determined and used to generate the frequency and scan-angle dependent behavior of the array. The results of NEC2 could be used to determine the coefficients of the impedance matrix. The matrix could then be used to quickly determine the impedance at any frequency or scan angle.

The NEC2/Matlab interface is applicable to a wide variety of antenna studies. It is particularly suitable to iterative optimization strategies. Applying this interface to study the application of genetic algorithms to antenna array optimization is a suitable course for future work.

With respect to continuing and improving the work of this thesis, the immediate future work involves basing the feedback of the algorithm on the output of NEC itself. Once this

is established, and a suitable set of parameters have been established, a graphical user interface could be developed to be more immediately usable by a new user. The ability to operate in the Matlab workspace could always be maintained as an option.

This work has only considered the presence of one desired signal and one source of interference. Future work should look into beamforming for multiple desired signals while suppressing multiple sources of interference and the dependence on the number of elements in the array.

Concluding Comments

Finally, the diverse scope of this work has been extremely challenging and rewarding to develop. Not only have existing fields of knowledge been studied and expanded, but also an effective tool for the analysis of antenna structures has been developed. Although similar commercial packages are available at considerable cost to do much the same kind of analysis, the work of this thesis has resulted in a free, readily customizable and optimizable interface with an existing full-wave solver that still is in popular use today.

The greatest contribution of this thesis ironically was the largest stumbling block. By applying the adaptive algorithms to a large set of (AoA,AoI) pairs, inherent inexplicable weaknesses in the LCMV method became apparent. Furthermore, given that initial development of the adaptive algorithms proceeded using array factor analysis, important performance-limiting affects have clearly been shown when the full-wave analysis of NEC2 was subsequently applied. Thus, there is greater sensitivity between the adapted weights and the subsequent radiation pattern than was expected based on the relationship between array factor and full-wave results as applied to conventional scanning array

design. Thus, adaptive array antenna design should include a full-wave analysis in the interim adaptation process for more accurate design according to specification.

I have appreciated the opportunity to explore this work and look forward to further developing and applying this tool to other areas of enquiry.

References

[1] Haykin, Simon, *Adaptive Filter Theory*, 3rd Ed., Prentice-Hall, 1996.

[2] Anderson, Raymond," The *unofficial* Numerical Electromagnetic Code (NEC) Archive' , Internet website: http://www.qsl.net/wb6tpu/swindex.html.

[3] Michelson, Dave," NEC-LIST Information' , http://emlib.jpl.nasa.gov/EMLIB/nec.html, 2002.

[4] EM Scientific Software, Inc., Source for MiniNEC program, http://www.emsci.com.

[5] Nittany Scientific, Source for NEC-Win software, http://www.nittany-scientific.com.

[6] Poynting Group, Source for SuperNEC software, http://www.poynting.co.za.

[7] Balanis, C. A., *Antenna Theory, Analysis and Design*, 2nd Ed., John Wiley & Sons, 1997.

[8] Burke, G.J. and Poggio, A.J., "Numerical Electromagnetic Code (NEC) - Method of Moments' , *NOSC Technical Document 116*, Naval Ocean Systems Center, Jan. 1980.

[9] Poggio, A. J. and Miller, E. K.," Integral Equation Solutions of Three-Dimensional Scattering Problems' , *Computer Techniques for Electromagnetics*, Ch. 4, edited by R Mittra, Pergamon Press, New York, 1973.

[10] Winters, Jack H.," Smart Antennas for Wireless Systems' , *IEEE Personal Communications*, Vol 1., February 1998, pp. 23-27.

[11] Paulraj, A. J. and C. B. Papadias," Space-Time Processing for Wireless Communications' , *IEEE Personal Communications*, Vol 14, No. 5, November 1997, pp. 49-83.

[12] Naguib, Ayman F., A Paulraj, T. Kailath," Capacity Improvement with Base Station Antenna Arrays in Cellular CDMA" , *IEEE Transactions on Vehicular Tech.*, Vol. 43, No. 3, August 1994, pp. 691-698.

[13] van der Veen, A. J., M. C. Vanderveen, A. Paulraj," Joint Angle and Delay Estimation Using Shift-Invariance Techniques' , *IEEE Trans. on Signal Processing*, Vol 46, No. 2, February 1998, pp. 405-418.

[14] Preston, Stephanie, D. V. Thiel, T. A. Smith, S. G. O' Keefe, J. W. Lu," Base-Station Tracking in Mobile Communications Using a Switched Parasitic Antenna Array' , *IEEE Trans. on Antennas and Propagation*, Vol 46, No. 6, June 1998.

[15] Howells, Paul W.," Explorations in Fixed and Adaptive Resolution at GE and SURC' , *IEEE Trans. Antennas & Prop.*, vol. AP-24, No. 5, pp.575-584, Sept. 1976.

[16] Applebaum, Sidney," Adaptive Arrays' , *IEEE Trans. Antennas & Prop.*, vol. AP-24, No. 5, pp.585– 598, Sept. 1976.

[17] Mailloux, Robert J., *Phased array antenna handbook*, Artech House, Boston, 1994.

[18] Weiner, N.," On a class of singular integral equations' , *Proc. Prussian Acad. Math-Phys.Ser.*, 1931, p.696.

[19] Widrow, B. & McCool, J.," A Comparison of Adaptive Algorithms Based on the Methods of Steepest Descent and Random Search' , *IEEE Trans. Antennas & Prop.*, vol. AP-24, No. 5, pp.615 - 637, Sept. 1976.

[20] Griffiths, L.J.," Adaptive Antenna Arrays in Mobile Communications' , *Intelligent Methods in Signal Processing and Communications*, Docampo, D. *et al.*editors, Birkhauser, Boston, 1997.

[21] Johnson, J. & Rahmat-Samii, Y.," Genetic Algorithms in Engineering Electromagnetics' , *IEEE Antennas and Propagation Magazine*, Vol.39, No.4, August 1997, pp. 7 - 25.

[22] Ohnishi, K & Milton, T.," A New Optimization Technique for Adaptive Antenna Arrays', *IEEE Trans. Antennas & Prop.*, vol. AP-41, No. 5, pp525 - 532, May 1993.

[23] Ko, C.C., Chin, F. & Foo, S., " An Adaptive Algorithm for Separating and Tracking Multiple Directional Sources in Linear Arrays' , *IEEE Trans. Antennas & Prop.*, vol. AP-40, No. 3, pp261 - 267, March 1992.

Appendix 1 - Matlab Code

Runit.m - This file calls and runs NEC2 based on the parameters defined in *SysParams.m*, using *GenNECfile.m* to generate the input file. It then parses in the NEC output using *parseoutput.m* and plots the output using *plotout.m*.

```
tic;
start = toc;

if exist('c:\nec2sa\pltdat.nec','file')
   !DEL PLTDAT.NEC
end

%clear ScanAngle
SysParams
load Vs
Vexc = Vexc/max(abs(Vexc));
yloc(:,[1 end]) = yloc(:,[1 end]) * 1;
[NECfile,Nodes] =
GenNECfile(f,xloc,yloc,zloc,r,excitation,NumElmnt,ElmntList,SegsPerElmnt, ...
    THETAs,PHIs,ScanAngle,Vexc);

!nec2sa
stop = toc;
stop-start
disp('----- job complete -----')

parseoutput
plotout
hpbw

clear fid aa

if excitation
    data = zget;  %% zget is a function to read the antenna input parameters from the
output file
    Vin = data(:,3);
    Iin = data(:,4);
    Zin = data(:,5);
    %Yin = data(:,6);
    %Pin = data(:,7);
end
```

GenNECfile.m

```
function [NECfile,Nodes] = GenNECfile(f,xloc,yloc,zloc,r,excitation,NumElmnt,ElmntList,
...
    SegsPerElmnt,THETAs,PHIs,ScanAngle,Vexc)

% f must be in MHz, 300MHz

if prod(size(Vexc)) > NumElmnt
    disp(' ')
    disp(' ---------------------------------------- ')
    disp('| There are more sources than elements! |')
    disp(' ---------------------------------------- ')
end

NumSegs = NumElmnt*SegsPerElmnt; % you'll have to figure this one out for variable
segs/element
Nodes = ceil(SegsPerElmnt/2):SegsPerElmnt:NumSegs;
%ceil(NumElmnt/2):SegsPerElmnt:NumSegs;
Tstart = THETAs(1); %
Tend =  THETAs(2);%
Tnum =  THETAs(3);%
Tstep =  mean(diff(linspace(Tstart,Tend,Tnum)));%
Pstart =  PHIs(1);%
Pend =  PHIs(2);%
Pnum =  PHIs(3);%
Pstep =  mean(diff(linspace(Pstart,Pend,Pnum)));%
NECfile=char('CE ');

%% Build Elements
for aa = 1:NumElmnt
    NECfile = char(NECfile,...

['GW,',num2str(aa),',',num2str(SegsPerElmnt),',',num2str(xloc(aa)),',',num2str(yloc(1,aa)
),',',num2str(zloc(aa)),...

',',num2str(xloc(aa)),',',num2str(yloc(2,aa)),',',num2str(zloc(aa)),',',num2str(r)]);
end
NECfile = char(NECfile,['GS 0 0 1'],['GE 0'],['EK -1']); % EK -1 for normal kernel, EK 0
for extended thin-wire kernel

%% Set Sources
if excitation
    for aa = 1:NumElmnt
        NECfile = char(NECfile,...
            ['EX 0 ',num2str(aa),' ',num2str(ceil(SegsPerElmnt/2)),' 0 ',...
                num2str(real(Vexc(aa))),' ',num2str(imag(Vexc(aa)))]);
    end
else
    NECfile = char(NECfile,...
        ['EX 1 1 1 0 ',num2str(ScanAngle),' 0 90']);
end

NECfile = char(NECfile,...
['FR 0 1 0 0 ',num2str(f),' 1'],...
'PL 1 2',...
'XQ',...
'PL 3 1 0 3',...
['RP 0 ',num2str(Tnum),' 1 1000 ',num2str(Tstart),' 0 ',num2str(Tstep),' 1'],...
'PL 3 2 0 3',...
['RP 0 1 ',num2str(Pnum),' 1000 90 ',num2str(Pstart),' 1 ',num2str(Pstep)],...
'EN');
%                          XNDA, 1000 See page 81 of manual


fid = fopen('input.nec','w');
for aa=1:size(NECfile,1)
   fprintf(fid,[deblank(NECfile(aa,1:end)),'\r\n']);
end
fclose(fid);
```

parsepoutput.m

```
Output = load('pltdat.nec');
I = Output(Nodes,:);
I = I(:,1).*(cos(I(:,2)*pi/180) + i*sin((I(:,2)*pi/180)));
Ts = NumSegs + 1;
Te = Ts + THETAs(3)-1;
Ps = Te + 1;
Pe = Ps + PHIs(3)-1;
TGffT = Output(Ts:Te,:); % Total Gain far-field Theta varies
TGffP = Output(Ps:Pe,:); % Total Gain far-field Phi varies

clear Ts Te Ps Pe
```

plotout.m

```
%figure,

Ltyp1 = char('b.-','g.-','r.-','c.-','m.-','k.-'); %,'y.-'
Ltyp2 = char('b-','g-','r-','c-','m-','k-'); %,'y-'

if ~exist('Lpick','var')
    if exist('LineTypePick.mat','file')
        load LineTypePick
    else
        Lpick = 1;
    end
end

%Lpick = size(Ltyp,1); % to get black!
figure(1)

subplot(221),eval(['plot(ElmntList,abs(I),''',Ltyp1(Lpick,:),''')'])
title('|I|','fontsize',12)
grid on, hold on,xlabel('element'),set(gca,'xlim',[-ceil(NumElmnt/2)
ceil(NumElmnt/2)],'xtick',-7:7)

subplot(222),eval(['plot(ElmntList,unwrap(angle(I))*180/pi,''',Ltyp1(Lpick,:),''')'])
grid on, hold on
title('phase I - degrees','fontsize',12)
xlabel('element'),set(gca,'xlim',[-ceil(NumElmnt/2) ceil(NumElmnt/2)],'xtick',-7:7)

subplot(223),eval(['plot(TGffT(:,1),TGffT(:,2),''',Ltyp2(Lpick,:),''')'])
grid on, hold on
title('Power Gain, (\theta,\phi) = (-90:90,0)','fontsize',12)
xlabel('\theta, degrees','fontsize',12),ylabel('dB','fontsize',12)

subplot(224),eval(['plot(TGffP(:,1),TGffP(:,2),''',Ltyp2(Lpick,:),''')'])
grid on, hold on
title('Power Gain, (\theta,\phi) = (90,0:360)','fontsize',12)
xlabel('\phi, degrees','fontsize',12),ylabel('dB','fontsize',12)
set(gca,'ylim',[-40 25])
set(gcf,'Position',[32 20 751 485])

%orient tall

figure(2)
%eval(['plot(TGffT(:,1),TGffT(:,2),''',Ltyp2(Lpick,:),''')'])
if 0
    plot(TGffT(:,1),TGffT(:,2),'k-','linewidth',2)
else
    plot(TGffT(:,1),TGffT(:,2)-max(TGffT(:,2)),'k-','linewidth',2)
end
grid on, hold on
title('Power Gain, (\theta,\phi) = (-90:90,0)','fontsize',12)
xlabel('\theta, degrees','fontsize',12),ylabel('dB','fontsize',12)
%orient tall

if Lpick == size(Ltyp1,1)
    Lpick = 1;
else
    Lpick = Lpick + 1;
end

save LineTypePick Lpick
```

InitParams_1J.m - This Matlab script initializes the parameters to be used by the LCMV and LMS algorithms, set to match those of the NEC2 input file.

```
%% InitParams for 1 jammer
clear all
Jpower = 10.^([20]/10);
if ~exist('f','var')
    f = 299.792458e6;
    Fs = 3*f; %3e9;
    t=0:1/Fs:300/f;
    un = exp(j*2*pi*f*t)';% note: this is x(n) in the text
    M = 5; %11;
    AoA = [-11.537]*pi/180; % Angles of arrival or interest
    g   = [1]'; %% must be a colum vector
    AoI = -53*pi/180; %asin(0.2); %[-16]*pi/180; %asin(0);%
    idxIP = 1:500; %length(t); % M;
    Rtype = char('none','coeff','biased','unbiased');
    typnum = 1;
    jamX = sqrt(1/2)*(randn(size(un)) +i*randn(size(un)));
    jam = sqrt(Jpower)*jamX;
    %jam = sqrt(Jpower)*(1+jamX/2).*un*exp(j*3*pi/5);
    %jam = sqrt(Jpower)*un*exp(j*3*pi/5);
end

if ~exist('jamX','var')
    jamX = sqrt(1/2)*(randn(size(un)) +i*randn(size(un)));
end

jam = sqrt(Jpower)*jamX; %(1+jamX/10).*un*exp(j*3*pi/5);

if ~exist('v_fxdVAR1','var')
    v_fxdVAR1 = sqrt(1/2)*(randn(M,length(idxIP)) + i*randn(M,length(idxIP))); %to set
noise var = 1
    v_fxd = sqrt(.1)*v_fxdVAR1;
end

Cint = sphi(AoI,M);

C = sphi(AoA,M); %sphi assumes kd = pi
Ca = orthcomp(C);
%Ca = Ca*diag(-1./Ca(1,:)); %scale Ca so that first row is -1, and first lower diagonal
is nPhi 5.134
%U = [C Ca];
wq = C * (C'*C)^-1 * g;

%wq = wq/max(abs(wq)); % to test normalizing wq

%C = flipud(C);
```

LCMV.m - calculates the adapted weight vector based on the LCMV algorithm.

```
clear xn yn yn1 yn1b yn2 yn3 yn4 yn4b dn px rxvec Rx wao1 wao2 dnn msERRaccum en VARaccum
unA Ru
en = [];
VARaccum = [0 0 0];

fast = 1;
%mulmcv = .4;
%wq = wq/max(abs(wq));
for aa = 1:length(idxIP)
    v1 = v_fxd(:,aa);
    unA1 = un(aa) * C; % eq 5.117 where C(:,1) = phi0, for steering vector
    jamA = jam(aa) * Cint;
    unA(:,aa) =  sum([unA1 v1 jamA],2);
    Ru(:,aa) = xcorr(unA(:,aa),Rtype(typnum,:));
    R = corrmat(Ru(:,aa));
    wao1(:,aa) = ((Ca'*R*Ca)^-1 * Ca' * R * wq);
    dn(aa,1) = wq' * unA(:,aa);
    xn = Ca'*unA(:,aa);
    yn1(aa,1) = dn(aa,1) - wao1(:,aa)'*xn; % Eq. 5.116

    en = [en; mserror(un(aa),[dn(aa) yn1(aa)])]; %5yn1b(aa)])];
    if ~fast
        if aa > 1
            VARaccum = [VARaccum; var([un(1:aa) dn yn1])];
        end
        msERRaccum(aa,:,1) = mserror(abs(un(1:aa)),abs([un(1:aa) dn yn1]));
        msERRaccum(aa,:,2) = mserror(real(un(1:aa)),real([un(1:aa) dn yn1]));
        msERRaccum(aa,:,3) = mserror(imag(un(1:aa)),imag([un(1:aa) dn yn1]));
        msERRaccum(aa,:,4) = mserror(un(1:aa),[un(1:aa) dn yn1]);
    end
end

if fast
    return
end
%return
figure,plot(idxIP,VARaccum(:,1:2),'k-.','linewidth',1.5),hold
on,plot(idxIP,VARaccum(:,3),'k-','linewidth',2)
hleg = legend('u(n)','d(n)','y(n)');set(hleg,'fontsize',14)%,set(gca,'ylim',[0 10])
set(gcf,'position',[6   334   518   363]),
%title(['ISR = ',num2str(10*log10(var(jam)/var(un))),'dB, LCMV: Accumulated Variance of
Signals'])
title(['AoA = ',num2str(AoA*180/pi),', AoI = ',num2str(AoI*180/pi),' deg, ISR = ',...
       num2str(10*log10(var(jam)/var(un))),'dB, LCMV: Accum Var of
Signals'],'fontsize',14)
xlabel('sample number','fontsize',14),ylabel('\sigma^2','fontsize',14)

figure,plot(idxIP,msERRaccum(:,2,1),'k-.','linewidth',2),hold
on,plot(idxIP,msERRaccum(:,3,1),'k-','linewidth',2)
hleg = legend('d(n)','y(n)');set(hleg,'fontsize',14)%,set(gca,'ylim',[0 10])
set(gcf,'position',[533   334   489   366]),
%title(['ISR = ',num2str(10*log10(var(jam)/var(un))),'dB, LCMV: Accumulated Mean-Square
Error of Signals'])
title(['AoA = ',num2str(AoA*180/pi),', AoI = ',num2str(AoI*180/pi),' deg, ISR = ', ...
       num2str(10*log10(var(jam)/var(un))),'dB, LCMV: Accum MSE of
Signals'],'fontsize',14)
xlabel('sample number','fontsize',14),ylabel('mse','fontsize',14)
```

LMS.m - calculates the adapted weight vector based on the LMS algorithm.

```
clear xn yn yn1 yn1b yn2 yn3 yn4 yn4b dn px rxvec Rx wao1 wao2 dnn msERRaccum en VARaccum

fast = 1;
noisyU = [];
en = [];
VARaccum = [0 0 0];
waoAccum = [];
if ~exist('mu','var')
    mu = 1e-4;
end
%mu = 1e-4;
for aa = 1:length(idxIP)
    v1 = v_fxd(:,aa);
    unA1 = un(aa) * C; % eq 5.117 where C(:,1) = phi0
    jamA = jam(aa) * Cint;
    unA =  sum([unA1 v1 jamA],2);
    if aa==1
        waoAccum = zeros(M-length(AoA),1); %[waoAccum wao1];
    end
    dn(aa,1) = wq' * unA;
    xn = Ca'*unA;
    yn1(aa,1) = dn(aa,1) - waoAccum(:,aa)'*xn; % Eq. 5.116
    enlms = yn1(aa,1);
    w_lms = waoAccum(:,aa) + mu*xn*enlms';
    waoAccum = [waoAccum w_lms];

    noisyU(aa) = (wq - Ca*waoAccum(:,aa))' * (unA);
    en = [en; mserror(un(aa),[dn(aa) yn1(aa)])];
    if ~fast
        if aa > 1
            VARaccum = [VARaccum; var([un(1:aa) dn yn1])];
        end
        msERRaccum(aa,:,1) = mserror(abs(un(1:aa)),abs([un(1:aa) dn yn1]));
        msERRaccum(aa,:,2) = mserror(real(un(1:aa)),real([un(1:aa) dn yn1]));
        msERRaccum(aa,:,3) = mserror(imag(un(1:aa)),imag([un(1:aa) dn yn1]));
        msERRaccum(aa,:,4) = mserror(un(1:aa),[un(1:aa) dn yn1]);
    end
end

if fast
    return
end

if 0
    DoAddplot
    return
end
figure,plot(idxIP,VARaccum(:,1:2),'k-.','linewidth',1.5),hold
on,plot(idxIP,VARaccum(:,3),'k-','linewidth',2)
hleg = legend('u(n)','d(n)','y(n)');set(hleg,'fontsize',14)%,set(gca,'ylim',[0 10])
set(gcf,'position',[6   334   518   363]),
%title(['ISR = ',num2str(10*log10(var(jam)/var(un))),'dB, LMS: Accumulated Variance of
Signals'])
title(['LMS: Accum Var of Signals, AoA = ',num2str(AoA*180/pi,4),', AoI =
',num2str(AoI*180/pi,4),' deg, ISR = ',...
        num2str(10*log10(var(jam)/var(un)),3),'dB, \mu = ',num2str(mu)],'fontsize',14)
xlabel('sample number','fontsize',14),ylabel('\sigma^2','fontsize',14)

figure,plot(idxIP,msERRaccum(:,2,1),'k-.','linewidth',2),hold
on,plot(idxIP,msERRaccum(:,3,1),'k-','linewidth',2)
hleg = legend('d(n)','y(n)');set(hleg,'fontsize',14)%,set(gca,'ylim',[0 10])
set(gcf,'position',[533   334   489   366]),
%title(['ISR = ',num2str(10*log10(var(jam)/var(un))),'dB, LMS: Accumulated Mean-Square
Error of Signals'])
title([' LMS: Accum MSE of Signals, AoA = ',num2str(AoA*180/pi,4),', AoI =
',num2str(AoI*180/pi,4),' deg, ISR = ', ...
        num2str(10*log10(var(jam)/var(un)),3),'dB, \mu = ',num2str(mu)],'fontsize',14)
xlabel('sample number','fontsize',14),ylabel('mse','fontsize',14)
```

NECvAF.m - script used to generate figures comparing NEC2 output with Array Factor.

```
%% compares NEC with wq or w
CompNEC = 0;
doFig = 2;
%clear Vexc
if ~exist('Vexc','var')
    Vexc = wq;
elseif 0
    Vexc = wq - Ca*wao1(:,451);
elseif 0
    Vexc = wq - Ca*waoAccum(:,368);  %0 for LMS
end

if ~exist('M','var')
    M = NumElmnt;
end

maxG = 0;
if ~exist('TGffT','var')
    ThetaRange = linspace(-pi/2,pi/2,500)';

elseif 0
    ThetaRange = TGffT(:,1)*pi/180; %linspace(-pi/2,pi/2,250)';
    maxG = max(TGffT(:,2));
end

phi = pi*sin(ThetaRange);

WqS = zeros(length(phi),1); %size(C,2));

for bb = 1
    for cc = 1:length(phi)
        s = exp(-j*(0:M-1)'*phi(cc));
        WqS(cc,bb) = Vexc'*s;
    end
end

WqSnorm = WqS./max(abs(WqS));

if CompNEC
    figure(2),hold on
else
    figure(doFig),hold on
end

%return
plot(ThetaRange*180/pi,20*log10(abs(WqSnorm))+maxG,'k:','linewidth',2)

if exist('aoiLine','var')
    if ishandle(aoiLine)
        delete(aoiLine)
        delete(aoaLine)
    end
end
ymax = 10+round(maxG/10)*10;
set(gca,'ylim',[-50 ymax]),xlabel('\theta,
degrees','fontsize',14),ylabel('dB','fontsize',14),
title('Directive Gain','fontsize',18)
aoiLine=line([AoI AoI]*180/pi,[-50 ymax],'linewidth',2,'linestyle',':','Color','k');
aoaLine=line([AoA(1) AoA(1)]*180/pi,[-50 ymax],'linewidth',2,'linestyle','-
.','Color','k');
% plot(TGffT(:,1),TGffT(:,2)-max(TGffT(:,2)))
```