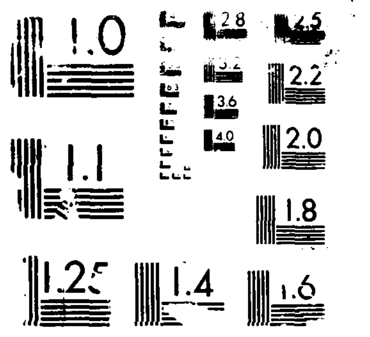
AD-A189 329 ADAPTIVE ALGORITHMS FOR HF ANTENNA ARRAYS(U) KANSAS UNIV CENTER FOR RESEARCH INC LAWRENCE TELECOMMUNICATIO D HAEGERT ET AL JUL 87 TISL-5481 RADC-TR-86-159 1/3

UNCLASSIFIED F30602-81-C-0205 F/G 9/1 NL



[illegible] RESOLUTION TEST CHART

(4)

RADC-TR-86-159
Final Technical Report
July 1987

ADAPTIVE ALGORITHMS FOR HF ANTENNA ARRAYS

University of Kansas

D. Haegert, R. Spohn, V. Frost, K. Shanmugan, G. Sargent,
R. Yarbro, B. McClendon and J. Holtzman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RADC-TR-86-159 has been reviewed and is approved for publication.

APPROVED: Peter J. Ritchie

PETER J. RITCHIE
Project Engineer

APPROVED: Bruno Beek

BRUNO BEEK, Technical Director
Directorate of Communications

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Directorate of Plans & Programs

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

[illegible]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TISL 5481	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-86-159
6a. NAME OF PERFORMING ORGANIZATION University of Kansas 6b. OFFICE SYMBOL (If applicable) TISL	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (DCCL)
6c. ADDRESS (City, State, and ZIP Code) Telecommunications and Information Sciences Lab 224 Nichols Hall, Campus West Lawrence KS 66045	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) DCCL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-81-C-0205
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62702F PROJECT NO. 4519 TASK NO 61 WORK UNIT ACCESSION NO. P2

11. TITLE (Include Security Classification)
ADAPTIVE ALGORITHMS FOR HF ANTENNA ARRAYS

12. PERSONAL AUTHOR(S)
D. Haegert, R. Spohn, V. Frost, K. Shanmugan, G. Sargent, R. Yarbro, B. McClendon, J.Holtzman

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Jun 85 TO Apr 86	July 1987	230

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
17	02	02
17	02	03

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
HF Adaptive Arrays; HF Channel Simulation; HF Communications Systems; HF Adaptive Algorithms

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
The performance of an adaptive array system will not only be dependent upon the controlling algorithm, but also upon the specific environment in which the array is used. Isolating the contributions made solely by the control algorithm represents a formidable task, considering that adaptive array systems are inherently used in situations in which the interference environment is initially unknown, time-variant, or both. If an effective algorithm choice is to be made, it is important to be able to compare array performances in identical environments. The major focus of this study is to compare and contrast the performance of various control algorithms under identical conditions in a computer simulated adaptive HF antenna array system. After examining adaptive array systems and the role played by the controlling algorithms, this study concentrates on the selection and simulation of three specific algorithms. The simulation results are then presented in the form of a comparative evaluation of the selected control algorithms and relevant conclusions are drawn.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Peter J. Ritchie	22b. TELEPHONE (Include Area Code) (315) 330-3077 22c. OFFICE SYMBOL RADC (DCCL)

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

List of Figures...v

List of Tables...viii

List of Symbols..ix

1.0 Introduction..1

2.0 Adaptive Array System Model.....................................5

- 2.1 Sensor Element Array.....................................5
- 2.2 Pattern-Forming Network..................................7
- 2.3 Adaptive Processor.......................................8

3.0 Least Mean Square Algorithm....................................10

- 3.1 Motivation for Selection of LMS Algorithm...............10
- 3.2 Mean Square Error (MSE) Performance Criterion...........11
- 3.3 MSE Performance Criterion Derivation....................13
- 3.4 LMS Algorithm Description...............................16
- 3.5 LMS Software Modules....................................19

4.0 Constrained LMS Algorithm......................................21

- 4.1 Motivation for Selection................................21
- 4.2 Constrained LMS Algorithm Description...................21
- 4.3 Derivation of Optimum Constrained Weight Solution.......25
- 4.4 Derivation of Constrained LMS Algorithm.................28
- 4.5 Constrained LMS Simulation Model........................29
  - 4.5.1 TAKAO Implementation..............................31
- 4.6 Constrained LMS Software Modules........................32

For

A-1

5.0 Update Covariance Algorithm....................................35

5.1 Motivation for Selection..................................35
5.2 Update Covariance Algorithm Description...............36
5.3 Update Covariance Software Modules....................38

6.0 Description of the HF Adaptive Array Simulation Model..........40

6.1 User-Definition of Array System and Environment.......40
6.2 Performance Evaluation....................................40
6.3 Simulation Software Description.......................44
6.3.1 Desired Signal Model..............................44
6.3.2 Interference Model..................................46
6.3.3 Thermal Noise Model................................46
6.3.4 Correlation Matrices...............................46
6.3.5 Simulation Program Operation..................48

7.0 Simulation Results and Conclusions.............................50

7.1 TEST1: Negligible Channel Effects...................54
7.2 TEST2: HF Channel with Moderate Delay Characteristics and Poor Attenuation Characteristics.................69
7.3 TEST3: HF Channel with Poor Delay Characteristics and Poor Attenuation Characteristics..................84
7.4 TEST4: Dependence of Convergence of Constrained LMS Algorithm on Signal Presence......................98
7.5 TEST5: Dependence of Adapted Antenna Patterns on Number of Required Iterations.......................107
7.6 TEST6: Investigation of an Alternate Antenna Geometry..118
7.7 Conclusions and Recommendations.....................141

References 143

Appendix A: Complex Lowpass Equivalent Representation............144

Appendix B: User Manual for HF Adaptive Antenna Array Evaluation Facility....................................146

LIST OF FIGURES

2.1 N-Element Adaptive Array System Model......................6

3.1 LMS-Controlled Adaptive Array System......................12
3.2 MSE Performance Surface..................................15
3.3 LMS Software Modules.....................................20

4.1 Signal-Aligned Broadband Adaptive Array System.............24
4.2 Narrowband Signal-Aligned Array System....................30
4.3 Constrained LMS Software Modules..........................33

5.1 Update Covariance Software Modules........................39

6.1 User-Entered Parameters..................................41
6.2 Three-Path HF Channel Model..............................43
6.3 Flow Diagram of Program Operation.........................49

7.1 Constant Test Parameters.................................51
7.2 Description of Elevation Plots.............................52
7.3 Description of Azimuthal Plots.............................53
7.4 Convergence Histogram of LMS Algorithm in Ideal Channel....55
7.5 Convergence Histogram of Constrained LMS Algorithm in Ideal Channel...56
7.6 Convergence Histogram of Update Covariance Algorithm in Ideal Channel...57
7.7 Convergence Histogram of LMS Algorithm in Channel 2........70
7.8 Convergence Histogram of Constrained LMS Algorithm in Channel 2..71
7.9 Convergence Histogram of Update Covariance Algorithm in Channel 2..72
7.10 Convergence Histogram of LMA Algorithm in Channel 3........85
7.11 Convergence Histogram of Constrained LMS Algorithm in Channel 3..86

7.12 Convergence Histogram of Update Covariance Algorithm in Channel 3 87

7.13 Convergence Histogram of Constrained LMS Algorithm in ideal Channel (no signal present) 99

7.14 Convergence Histogram of Constrained LMS Algorithm in Channel 2 (no signal present) 100

7.15 Convergence Histogram of Constrained LMS Algorithm in Channel 3 (no signal present) 101

7.16 Convergence Summary for LMS Algorithms 114

7.17 Convergence Summary for Constrained LMS Algorithm 115

7.18 Convergence Summary for Update Covariance Algorithm 116

7.19 Convergence Summary for Constrained LMS Algorithm (no signal present) 117

7.20 Alternate Antenna Geometry for Rhomboid Geometry 119

7.21 Convergence Histogram of LMS Algorithm in Channel 2 for Rhomboid Geometry 120

7.22 Convergence Histogram of Constrained Algorithm in Channel 2 for Rhomboid Geometry 121

7.23 Convergence Histogram of Update Covariance Algorithm in Channel 2 for Rhomboid Geometry 122

7.24 Convergence Histogram of LMS Algorithm in Channel 3 for Rhomboid Geometry 123

7.25 Convergence Histogram of Constrained LMS Algorithm in Channel 3 for Rhomboid Geometry 124

7.26 Convergence Histogram of Update Covariance Algorithm in Channel 3 for Rhomboid Geometry 125

T1.1-8 Unadapted and Adapted Antenna Plots at the Arrival Angles of the Jamming Signals for TEST1 61-68

T2.1-8 Unadapted and Adapted Antenna Plots at the Arrival Angles of the Jamming Signals for TEST2 76-83

T3.1-8 Unadapted and Adapted Antenna Plots at the Arrival Angles of the Jamming Signals for TEST3 90-97

T4.1-4 Unadapted and Adapted Antenna Plots at the Arrival Angles of the Jamming Signals for TEST4..............103-106

T5.1-4 Comparison of Adapted Antenna Patterns of Fastest-/ Slowest Convergence..................................108-111

T6.1-14 Unadapted and Adapted Antenna Plots with Rhomboid Geometry for TEST2 and TEST3.........................127-140

LIST OF TABLES

Table 3.1 Features of Adaptive Algorithms........................22

Table 7.1 TEST1 Summary..58

Table 7.2 TEST2 Summary..73

Table 7.3 TEST3 Summary..88

Table 7.4 TEST4 Summary..102

Table 7.5 Comparison of Convergence Properties for the Adaptive Algorithms..................................113

LIST OF SYMBOLS

N	number of antenna elements
$x_i(t)$	output of ith antenna element
$S_i(t)$	signal component of output of ith antenna element
$n_i(t)$	noise (deliberate and natural) component of output of ith antenna element
$y(t)$	overall array output
w_i	ith antenna weight
$d(t)$	reference signal
$\varepsilon(t)$	error signal
μ	convergence constant
$\underline{R}$	sample covariance matrix
$\underline{I}$	identity matrix
$\underline{C}$	constraint matrix
$\underline{P}$	projection matrix for constraint re-establishment
$\underline{\beta}$	orthogonal vector for constraint re-establishment
α	data de-weighting constant
$\underline{\lambda}$	vector of LaGrange multipliers
$\underline{f}$	response vector
ζ	mean square error

1.0 INTRODUCTION

Conventional antenna receiving systems are susceptible to performance degradation due to the presence of undesired noise signals (deliberate or natural) that enter the system. Extensive research has been conducted in the area of adaptive antenna arrays as a means of compensating for the inevitable presence of these interference signals.

Adaptive arrays can provide a vital element of flexibility to a communication system. They can respond to changes in the interference environment by steering nulls and reducing sidelobes in the directions of the interferences, while maintaining an acceptable level of response in the direction of the desired signal. More importantly, they can do so without prior information pertaining to the interfering signals. These features make adaptive array systems very attractive for applications in which the environment is changing or unknown. For example, it is quite possible that very little descriptive information concerning an intentional jamming signal will be available. Any system requiring such information could not effectively compensate for the interference.

Adaptive arrays also have a reliability advantage over their conventional counterparts. In a conventional array system, if a single sensor element becomes inoperable, the characteristic gain pattern may be noticeably affected, depending on the location of the non-working element and the total number of sensor elements. In contrast, an adaptive array system positions, nulls, and reduces sidelobes according to the monitored external signal/interference environment. Therefore, if a sensor element became incapacitated, the remaining operable elements would be adjusted to produce a pattern that is similar to the original pattern. Simply stated, adaptive array systems fail more gracefully than conventional receiving systems.

The heart of the adaptive array system is the controlling algorithm. It determines not only the method that is used to adapt, but also the speed of adaptation and the amount and complexity of hardware necessary to implement the system. Some of the early pioneering work in the area of adaptive control algorithms began in the 1960's. B. Widrow and others developed a self-training, self-optimizing control algorithm known as the Least Mean Square

(LMS) algorithm [1]. This algorithm uses gradient techniques to asymptotically approach an optimal solution. At approximately the same time, Howells and Applebaum were developing a sidelobe cancelling algorithm for radar applications [2]. This algorithm exploited the fact that the signal of interest was normally absent, and attempts to maximize a generalized signal-to-noise ratio. Numerous algorithms were developed shortly thereafter. Among the more common types are the Differential Steepest Descent algorithm, constrained algorithms such as Griffiths' P-vector and Frost's Constrained LMS, and random search algorithms [3-5].

The direct matrix inversion and recursive processors represent a significant departure from the above algorithms, and they were developed more recently. Although their heavy computational load renders them impractical for many applications, the advancements in cheap, fast digital hardware have spurred an increasing interest in these methods. Both methods involve finding the inverse of the sample covariance matrix to determine an optimum weight solution [7-9].

As mentioned, a wide variety of adaptive algorithms have been developed and utilized. This fact suggests that there is no "clearly superior" adaptive control algorithm that should be implemented without regard to the application. In fact, the performances of adaptive control algorithms are very application dependent. The selection of a specific algorithm is based on many factors including the quantity of a priori information, the convergence speed requirements, implementation cost considerations, and the transmission medium, among others. The process of selecting a control algorithm is further complicated by the fact that adaptive array systems are generally placed in changing surroundings. These natural uncertainties often make it impossible to analytically predetermine how well a specific algorithm will perform. This, combined with the fact that it is often economically unreasonable to build a system in order to test it, makes computer simulation a useful tool for array evaluation.

The importance of application considerations has been stressed. This study is concerned with the simulation of a communication system operating in the HF frequency band. The HF band, which spans the 3-30 MHz range, is commonly used in military communication systems, and has been modeled as a slowly varying channel. Other factors that played a significant role in the algorithm selection process include:

- Knowledge of interleaved code is periodically available for reference signal generation.

- Knowledge of the angle of arrival of the desired communication signal is known a priori.

- Antenna array geometry.

Three adaptive control algorithms have been selected for computer simulation in this study. They were selected on the basis of their usefulness and applicability in the system of interest as well as the fact that they comprise a fairly representative set of adaptive algorithms in general. These algorithms are:

1. Least Mean Square Algorithm
2. Constrained LMS Algorithm
3. *Update Covariance Algorithm (recursive)*

The motivation for their selection as well as their attributes and operation will be discussed in detail later in the report.

Once the control algorithms had been selected, attention was turned to the development of the computer simulation models. The overall adaptive array system model was defined first, followed by the software implementation of the chosen control algorithms. In order to obtain relevant results regarding algorithm performance, it was imperative that the control algorithm be isolated in the overall system model. In this way, identical environments could be reproduced, and discrepancies in array performance could be attributed solely to the differences in algorithms. It was also necessary to define performance measures and develop a scheme to monitor the results in order to make a comparative evaluation of the selected control algorithms.

The first section of this report is devoted to the definition of the basic adaptive array system model. The principle system elements and their operation are explained, and common notation which will be used in following discussions is presented.

Once these fundamental aspects have been examined, each of the selected algorithms is discussed in detail. Each discussion will include the motivations for that algorithm's selection, and a description of how the algorithm operates. It will also contain an explanation of the simulation model that has been used.

The next section gives a presentation of the testing procedure and all parameters that must be specified. The signal models, performance measures, and simulation processes will be explained. The final section presents the results in the form of graphs and offers conclusions concerning the algorithm recommendations.

2.0 ADAPTIVE ARRAY SYSTEM MODEL

The purpose of this section is to introduce the adaptive array system model that is used throughout the study. Although such a model is explicitly required for a computer simulation, it is primarily intended to aid in the understanding of the qualities and objectives of the simulated array sytem. A common representation of an adaptive array system is shown in Figure 2.1. This basic model, while not possessing an abundance of detail, can be used to examine the principle system elements and their functions.

Virtually all adaptive antenna array systems consist of three principle components as depicted in Figure 2.1. These components include: 1) An array of sensor elements; 2) A pattern-forming network; 3) An adaptive processor. Although not explicitly shown, it is also important to realize that the system is located in an environment in which the desired and deliberate jamming signals impinge on the array at various angles of azimuth and elevation in the presence of thermal noise. Each of the components will now be discussed.

2.1 Sensor Element Array

The array consists of N spatially separated sensor elements. Their function is to monitor the external signal and interference environment and distribute this information to the other system components. The sensor elements should be chosen according to their ability to perform in the specified medium. These elements can then be physically located in a planar configuration to produce a suitable gain pattern over the desired region. Care should be taken when choosing the type of sensor element and their locations, as both factors place fundamental limitations on the ultimate capability of the adaptive array system.

The output of each sensor element, $x_i(t)$, is simply the sum of the signal components that arrive at that element.

$$x_i(t) = s_i(t) + n_i(t)$$

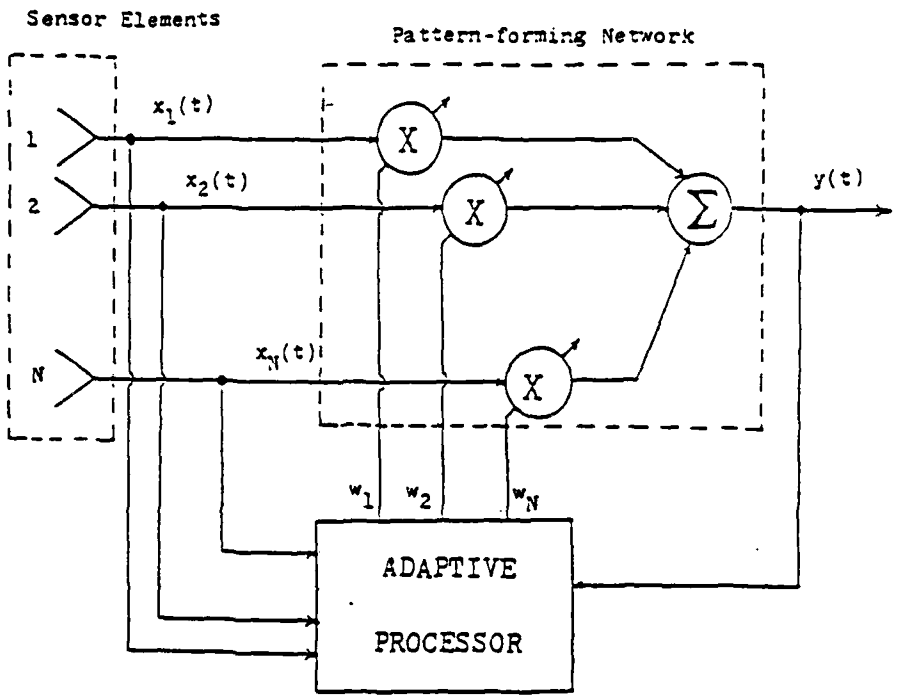


Figure 2.1 N-element Adaptive Array System Model

where

$s_i(t)$ is the desired signal component at element i

$n_i(t)$ is the combined noise components at element i from both deliberate and natural sources.

Also note that components of the same signal will differ from element to element due to phase delay caused by the spatial separation of the sensors.

The sensor element array model that has been utilized in this study contains some differences with models used in much of the related literature. The most notable departure from virtually every simulation conducted, is the use of a dipole model for the sensor elements. The use of isotropic elements is almost universal among simulated studies. Factors such as frequency dependence and directional gain of the elements themselves can then be ignored. Although using dipole elements complicates the model, "reality" is not compromised as severely. Also, many simulations are limited to linear array alignments. Such alignments make phase calculations almost trivial, but may inhibit the array's ability to distinguish signals on the basis of elevation arrival angle.

2.2 Pattern-Forming Network

The pattern-forming network consists of N variable complex weights and a summing device. The function of the network is very straightforward. It accepts a set of weights from the adaptive processor and multiplies each of them with their corresponding sensor element output. This multiplication can be thought of as an amplitude and phase adjustment of the element outputs. The complex weights can be written as

$$w_i = A_i e^{j\Theta_i} \tag{2.1}$$

where

$$A_i = \sqrt{(\mathrm{Re}\{w_i\})^2 + (\mathrm{Im}\{w_i\})^2}$$

and

$$\Theta_i = -\tan^{-1} (\mathrm{Im}\{w_i\}/\mathrm{Re}\{w_i\})$$

Thus, the output of the sensor element i is assigned a gain, A_i, and delayed by Θi radians at the discretion of the adaptive processor.

The sensor output/weight products are summed to produce the overall array output signal, y(t), which can be written as

$$y(t) = \sum_{i=1}^{N} w_i \, x_i(t) = \underline{x}^T \underline{w} \tag{2.2}$$

where the column vectors $\underline{x}$ and $\underline{w}$ are defined as

$$\underline{x} = \begin{bmatrix} x1(t) \\ x_2(t) \\ x_3(t) \\ \cdot \\ \cdot \\ \cdot \\ x_N(t) \end{bmatrix} \qquad \underline{w} = \begin{bmatrix} w1 \\ w2 \\ w3 \\ \cdot \\ \cdot \\ \cdot \\ wN \end{bmatrix} \tag{2.3}$$

Throughout the report, matrices will be denoted with an underscored capital letter. Vectors will be denoted with an underscored lower case letter. Complex conjugates will be assigned a superscript asterisk (*), and transpositions will be denoted with a superscript "T". Scalar quantities will be written as a lower case letter with no special notation.

2.3 Adaptive Processor

The adaptive processor controls the operation of the array system. It is comprised of the control algorithm and any supplemental signal processing components required by that particular algorithm. The adaptive processor uses the sensor element and array system outputs to compute a new set of complex weights. These weights are passed to the pattern-forming network to adjust the amplitude and phase of the sensor element outputs. The resulting array system response, which is hopefully acquiring a greater resemblance to the desired communication signal, is then used by the adaptive processor to help compute the next weighting vector.

The variable complex weights are updated by the processor in such a way that the array system output is optimized according to some specified performance criterion. This criterion governs the control algorithm by defining the parameters that are to be optimized through adaptation. Different control algorithms have different performance criteria. In other words, despite the fact that they are all trying to accomplish similar tasks, algorithms strive for optimization in different ways. The following three sections are devoted to the selected algorithms (processors), and include explanations of what the algorithms are trying to accomplish and how they achieve their respective goals.

Before leaving this section, it should be mentioned that although the array system model was depicted using continuous time representation, computer simulations require discrete samples. In the discussions and derivations that follow, signals will be represented as a sampled value. Also, in order to represent bandpass signals, complex lowpass equivalent representation (CLPE) has been utilized. A discussion of CLPE is given in Appendix A.

3.0 LEAST MEAN SQUARE (LMS) ALGORITHM

The LMS algorithm was introduced and developed by Widrow in the mid 1960's [1]. It uses gradient estimation techniques to arrive at an optimal solution. The LMS algorithm is probably the most popular of all adaptive algorithms. It has been used in a variety of adaptive applications including channel equalization, noise cancellation, and antenna array systems. It has become somewhat of a standard and is frequently used as a performance barometer for other adaptive algorithms studies.

It is important to note that the LMS algorithm requires the generation of an error signal, which in turn requires the generation of a reference signal. This reference signal represents the signal that it is desired to receive and it is generated using some approximation technique. For some applications, the reference signal requirement is unattainable, and the LMS algorithm cannot be used. Communication systems in general, however, lend themselves to reference generation, and systems that involve the use of known codes have been shown to be particularly conducive to the generation of a satisfactory reference signal [11-13].

3.1 Motivation for Selection of LMS Algorithm

As its popularity suggests, the LMS algorithm has many attractive qualities. Probably its most attractive quality is its overall simplicity. Few computations must be performed in order to update the complex weights. It takes on the order of 2N computations, where N is the number of sensor elements, to update the weights. Also, the adjustment of one weight does not affect the adjustment of another. Therefore, the weights can be updated simultaneously. This keeps processing time to a minimum.

A natural outgrowth of the LMS algorithm's computational simplicity is the relatively small hardware requirements. It can be easily implemented in either analog or digital form. For many applications, the LMS algorithm represents a good trade off between speed of convergence* and implementational

* The speed of convergence is measured relative to the number of times the complex weighting vector must be updated before the antenna pattern has converged.

feasibility. As a general rule, algorithms with rapid convergence rates are very complex. The LMS algorithm, while not possessing convergence rates as rapid as those offered by recursive processors, has rates that are satisfactory for most slowly varying environments.

Finally, the operation of the LMS algorithm is straightforward and easily understood. The algorithmic steps are clear and well defined. This along with the other factors mentioned, make the LMS algorithm a prime candidate for our study.

The operation of the LMS algorithm is governed by the Mean Square Error performance criterion. Before discussing the inner workings of the LMS algorithm, this criterion will be explained as it adds valuable insight into how the algorithm operates.

3.2 Mean Square Error (MSE) Performance Criterion

The LMS-controlled adaptive array system is shown in Figure 3.1 and will be used to present the fundamental manner in which the MSE criterion is used. For the moment, assume that the reference signal, d(k), is the actual value that is sent by the desired communicator. An error signal, e(k), is defined as the difference between what was sent and what was received.

$$e(k) = d(k) - y(k) = d(k) - \underline{w}^T \underline{x}(k) \qquad (3.1)$$

The LMS algorithm uses this error signal, along with the sensor element output information, to calculate a new set of complex weight values. The weights are computed such that the resulting error signal, and thus the MSE is reduced. As this process is repeated, the mean square error approaches zero, signifying that the value that was received was approximately equal to the value that was sent.

Although correct in principle, the preceding discussion has one blatant flaw. If the actual value sent by the communicator was truly known at the receiver, no information would be conveyed and there would certainly be no need to perform complicated adaptive techniques. The desired signal cannot be known with certainty at the receiver and therefore must be estimated. As mentioned, systems using known codes, such as the current system of interest, have been shown to be particularly adept at reference signal generation.

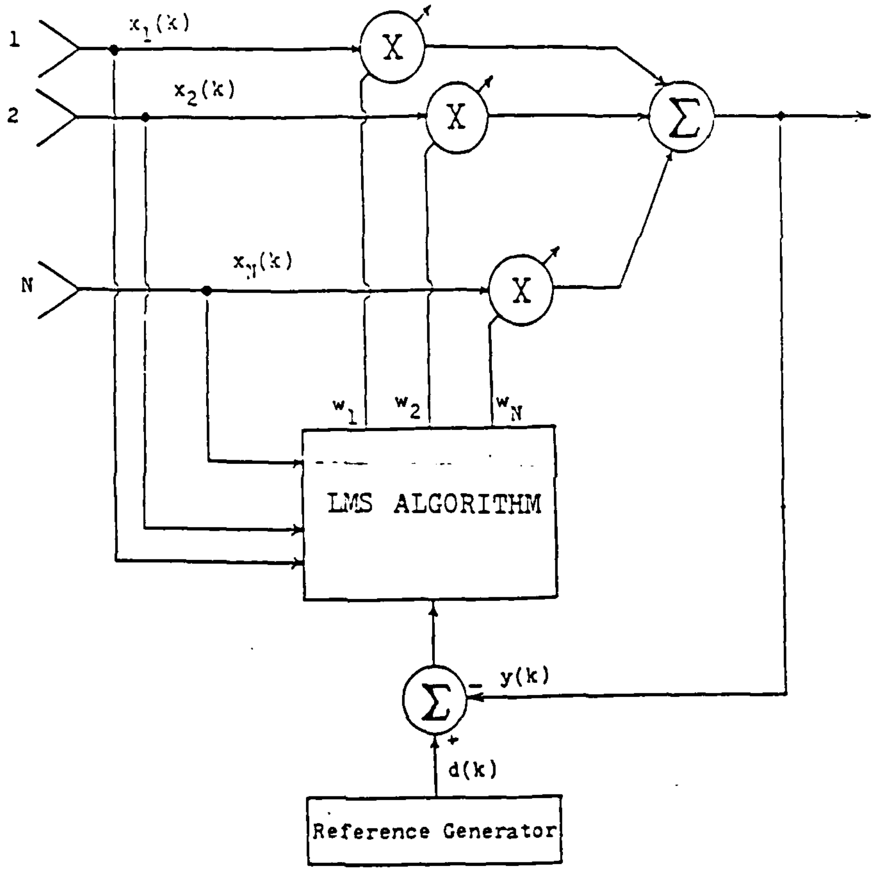


Figure 3.1 LMS-controlled Adaptive Array System

3.3 MSE Performance Criterion Derivation

The goal of the LMS algorithm is to adjust the complex weights in order to minimize mean square error. An expression of mean square error as a function of the weight values will now be derived.

The required error signal is defined as the difference between the generated reference signal and the adaptive array output signal.

$$e(k) = d(k) - y(k) = d(k) - \underline{w}^T \underline{x}(k)$$

The square of the error signal can then be written as

$$e^2(k) = d^2(k) - 2d(k)\, \underline{x}^T(k)\, \underline{w} + \underline{w}^T\, \underline{x}(k)\, \underline{x}^T(k)\, \underline{w} \qquad (3.2)$$

Taking the expected value of both sides yields

$$E\{e^2(k)\} = E\{d^2(k)\} + \underline{w}^T\, E\{\underline{x}(k)\, \underline{x}^T(k)\}\, \underline{w} - 2E\{d(k)\underline{x}^T(k)\}\, \underline{w}. \qquad (3.3)$$

The above equation represents the mean square error as a function of the complex weights.

In order to simplify the notation, define the vector p as the cross correlation between the desired response and the sensor element output and the matrix R as the input correlation matrix.

$$\underline{p} = E \begin{bmatrix} d(k)x_1(k) \\ d(k)x_2(k) \\ \cdot \\ \cdot \\ \cdot \\ d(k)x_N(k) \end{bmatrix} \qquad (3.4)$$

where d(k) is the scalar desired response and $x_i(k)$ is the output of sensor element i.

$$\underline{R} = E \begin{bmatrix} x_1(k)\, x_1(k) & x_1(k)\, x_2(k) & \cdots & x_1(k)\, x_N(k) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ x_N(k)\, x_1(k) & x_N(k)\, x_2(k) & \cdots & x_N(k)\, x_N(k) \end{bmatrix} \qquad (3.5)$$

Once again, $x_i(k)$ is the output of sensor element i. The mean square error can then be defined as

$$\text{MSE} = \xi = E\{e^2(k)\} = \underline{w}^T \underline{R}\, \underline{w} - 2\underline{p}\, \underline{w} + E\{d^2(k)\} \qquad (3.6)$$

It is very important to note that the MSE is a positive quadratic function of the weights. This function is a concave hyperparabaloidal (bowl-shaped) surface that contains no local minima. This is a performance surface and is depicted in Figure 3.2.

Notice that the minimum mean square error corresponds to the global minimum of the performance function. It is also known that the gradient of the function is zero at this minimum. Therefore, in order to find the optimal weight settings (the weights which produce minimum mean square error), the gradient of the function is found and set equal to zero.

The gradient is obtained by differentiating the MSE function with respect to the weights.

$$\nabla = \begin{bmatrix} \dfrac{\partial E\{\varepsilon^2(k)\}}{\partial w_1} \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{\partial E\{\varepsilon^2(k)\}}{\partial w_N} \end{bmatrix} = -\,2\underline{p} + 2\,\underline{Rw} \qquad (3.7)$$

Setting the gradient equal to zero, the optimal weights are found.

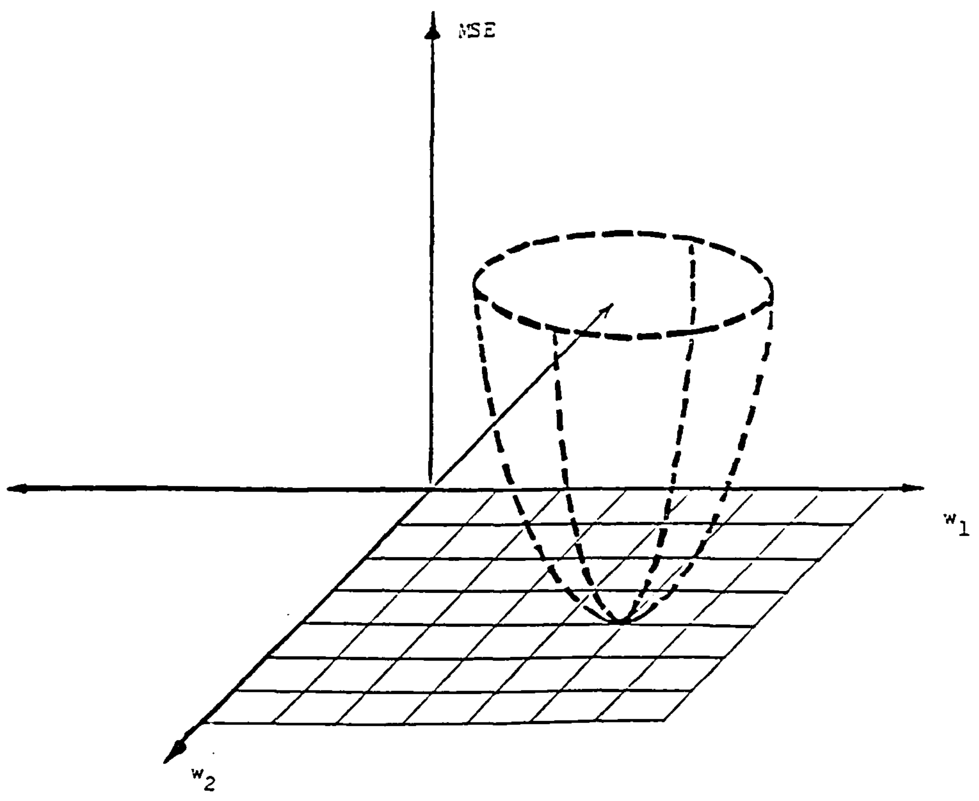


Figure 3.2 MSE Performance Surface

$$-2\underline{p} + 2\,\underline{Rw} = 0$$

$$\underline{w}_{OPTIMUM} = \underline{R}^{-1}\,\underline{p} \qquad (3.8)$$

This is an extremely important result and represents the matrix form of the Weiner-Hopf equation.

The Weiner-Hopf equation defines the optimal weight settings in the mean square sense. Intuitively, it may seem unreasonable to prescribe a complicated adaptive technique when an optimal solution is known. As will be shown, however, the solution is the problem, and the LMS algorithm is a method to avoid actually computing the Weiner-Hopf equation.

3.4 LMS Algorithm Description

An actual computation of the Weiner-Hopf equation would require the explicit measurement of all correlation functions, since it is unreasonable to assume that the correlations of deliberate interferences will be known. This is a plausible task but it would require large amounts of hardware. The requirement of matrix inversion, which for most arrays of practical size is computationally intensive, is more prohibitive. In fact, for most applications of reasonable magnitude, the process of directly inverting the correlation matrix is entirely unfeasible. The LMS algorithm is simply a practical method of finding close approximations to the Weiner-Hopf equation.

The LMS algorithm is an implementation of the Method of Steepest Descent. Using this method, the updated weight vector is equal to the past weight vector plus a change that is proportional to the negative gradient.

$$w(k + 1) = w(k) - \mu\nabla \qquad (3.9)$$

The parameter μ is a factor that controls stability and convergence rate. Updating the weights can be thought of as descending along the aforementioned performance surface in an attempt to reach the "bottom of the bowl."

The LMS algorithm avoids explicit correlation function measurement and matrix inversion by utilizing a crude but effective gradient estimate. Recall that the gradient of the MSE function is given by

$$\nabla = \begin{bmatrix} \frac{\partial E\{\varepsilon^2(k)\}}{\partial w_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial E\{\varepsilon^2(k)\}}{\partial w_N} \end{bmatrix} = -2\underline{p} + 2\,\underline{Rw}$$

The LMS algorithm estimates the gradient by using the square of a single error sample instead of the MSE and differentiating with respect to the weights.

$$\hat{\nabla} = \begin{bmatrix} \frac{\partial \varepsilon^2(k)}{w_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial \varepsilon^2(k)}{\partial w_N} \end{bmatrix} = -2\,\varepsilon(k)\,\underline{x}^*(k) \qquad (3.10)$$

Using this gradient estimate in place of the true gradient in (3.9) yields the LMS algorithm.

$$\underline{w}(k+1) = \underline{w}(k) + 2\mu e(k)\,\underline{x}^*(k) \qquad (3.11)$$

Notice that this algorithm does not require squaring, averaging or differentiation. The gradient estimate can be shown to be an unbiased estimator of the true gradient.

$$
\begin{aligned}
E\{\hat{\nabla}\} &= E\{-2e(k)\ x^{*}(k)\} \\
&= E\{-2[d(k) - y(k)]\ \underline{x}^{*}(k) \\
&= -2E\{d(k)\ \underline{x}(k) - \underline{x}(k)\ \underline{x}^{*}(k)\ \underline{w}(k)\} \\
&= 2(\underline{Rw} - \underline{p}) = \nabla \qquad (3.11)
\end{aligned}
$$

The LMS algorithm does not require the angle of arrival of the desired signal to be known a priori. If it is unknown, the weights are normally initialized to an arbitrary value of 1 < 0°.

$$\underline{w}(0) = \begin{bmatrix} 1 < 0^0 \\ 1 < 0^0 \\ \cdot \\ \cdot \\ \cdot \\ 1 < 0^0 \end{bmatrix} \qquad (3.12)$$

If the angle of arrival of the desired signal is known, however, the initial weights can be chosen such that the initial antenna pattern effectively "looks" directly at the desired signal.

$$\underline{w}(0) = \begin{bmatrix} e^{-j\Omega_1} \\ e^{-j\Omega_2} \\ \cdot \\ \cdot \\ \cdot \\ e^{-j\Omega_N} \end{bmatrix} \qquad (3.13)$$

where $-\Omega_i$ is a phase value that exactly compensates for the phase delay due to the spatial separation of the sensor elements. These values can be easily calculated if the angle of arrival and element locations are known.

3.5 LMS Software Modules

Simulation of the LMS algorithm requires two routines. The weights are initialized using the routine WEIGHTINIT. The initial weights are given as

$$w_i(0) = \exp(-j\Omega_i) \quad i = 1, 2, \cdots, N$$

where Ω_i is the phase of the desired signal at element i.

The routine LMS updates the weights according to the update equations that have been given. It requires both the sensor element outputs and the overall array system output to adjust the weights. It also requires a reference signal. In this model, it has been assumed that a known code is available. It is assumed that the code at the receiver is synchronized with the code (preamble) that is being sent. The channel model introduces delay, however, so the reference signal must be delayed accordingly if the synchronization assumption is to be satisfied.

The FORTRAN source code listings are given in Appendix C. Figure 3.3 depicts the modules discussed and illustrates the primary input and output parameters. The variable names used in the program are shown in parenthesis.

The LMS algorithm is not without its drawbacks. It does not converge terribly fast, but more importantly, it requires the presence of a reference signal to adapt. In cases such as this one where the reference signal is not always present, the weight values would have to be effectively frozen during the signal's absence. Environmental changes in this period could not be tracked.

The severity of this problem will be dependent upon the rapidity of change of the application medium. A slowly varying medium should pose no prohibitive difficulties. An adaptive algorithm which does not encounter the problem of a required reference signal will now be examined.

LMS SOFTWARE MODULES

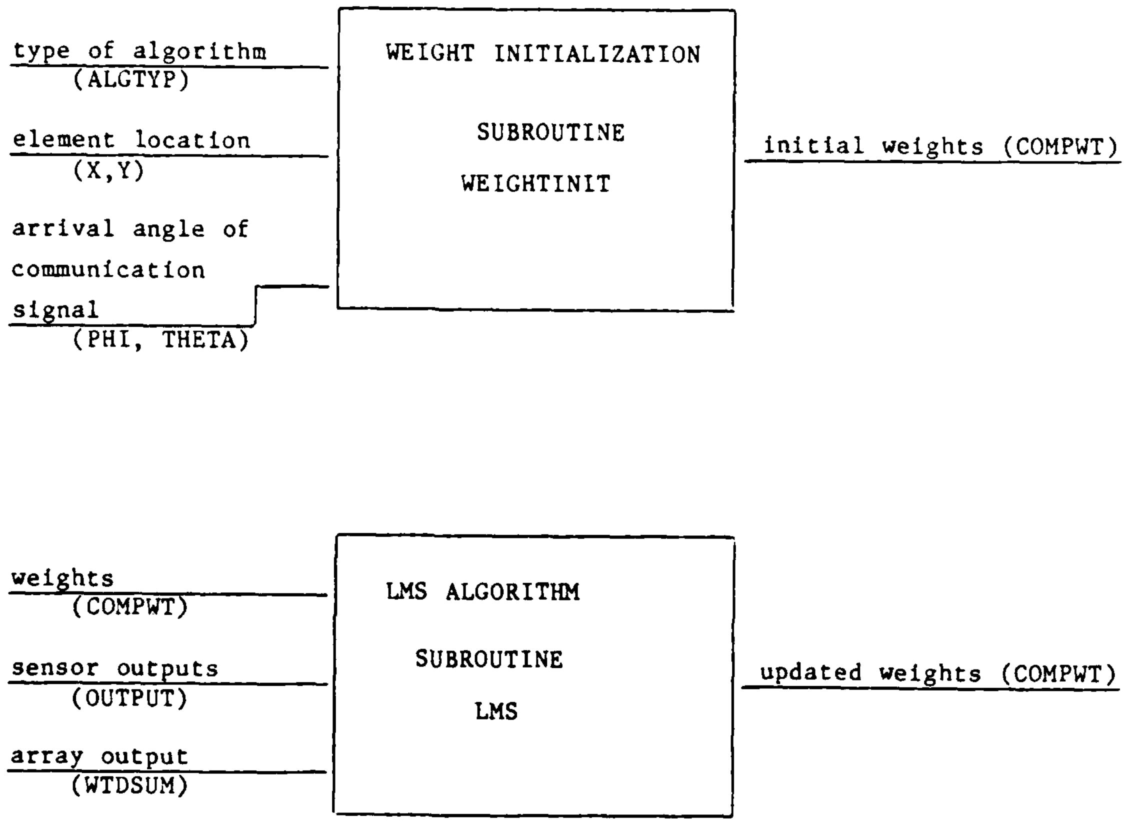


Figure 3.3 LMS Software Modules

4.0 CONSTRAINED LMS ALGORITHM

The second control algorithm selected for this study is the Constrained LMS algorithm. It was developed by O. L. Frost [5]. The name is somewhat misleading as it suggests that this algorithm is simply a permutation of the LMS algorithm. This is definitely not the case, and some of their contrasting features are illustrated below in Table 3.1. The name may have been derived from the fact that, like the LMS algorithm, the Constrained LMS algorithm uses a gradient approach.

4.1 Motivation for Selection

The primary advantage of the Constrained LMS algorithm is the elimination of the reference signal requirement. No reference signal must be present for the algorithm to reduce the effects of interferers or respond to changes in the environment. Complicated reference generation techniques can be avoided. It is, however, required that the arrival angle of the desired signal be known a priori. Secondly, the Constrained LMS algorithm requires relatively few computations in order to update the variable weights. Finally, many adaptive processors tend to degrade their own mainbeam response when attempting to place nulls in the directions of interferences. By explicitly constraining the response, the Constrained LMS algorithm prevents this from occurring.

4.2 Constrained LMS Algorithm Description

As mentioned, the Constrained LMS algorithm places fundamental limits on the adaptive array's response in the direction of interest (look direction). Throughout the adaptation process, the response in the direction of arrival of the desired signal will remain unchanged. The individual variable weights are allowed to take on any values in order to null out interferences provided that the look direction response is maintained. This type of algorithm allows the array to look in the direction of the desired signal (whether it is present or not) while ignoring signals arriving from other directions. It is true that an interfering signal that enters the array system at virtually the same angle as the desired signal will disturb the system. The same could be said for all algorithms because they rely on spatial separation to discriminate.

Table 3.1

Features of Adap'ive Algorithms

Feature	LMS	Constrained LMS
Reference Signal	Required	Not Required
Angle of Arrival of Communication Signal	Not Required	Required
Performance Criterion	M.S.E.	Maximum-Likelihood
Optimization Technique	Minimize Error	Minimize output power according to constraints

The adaptive array system model which will be used to explain the operation of the Constrained LMS algorithm was presented in [5] and is shown in Figure 4.1. Although this simulation deals with narrowband signals, the original broadband processor model will be used in this discussion. The model consists of N elements and J taps per element. When narrowband signals are used a simplified model results and it will be described later. Also shown in Figure 4.1 is an "equivalent processor" which aids in the understanding of how the Constrained LMS algorithm operates.

From Figure 4.1, it is evident that the Constrained LMS processor contains an additional component. This component, known as a spatial correction filter, performs a task that is often regarded as preprocessing. This filter compensates for the physical misalignment of the sensor elements by introducing individual delays so that the desired signal effectively arrives at the same time at each element. In other words, the spatial correction filter guarantees that the communication signal component is identical at each element output. The delays can be calculated from the array geometry and the arrival angle of the desired signal. Noise components arriving at the sensors at other angles will not produce equal components at the element outputs.

From the desired signal's vantage point, the processors in Figure 4.1 are equivalent. Each adaptive weight in the equivalent processor is simply equal to the sum of the weights in the vertical column above it. With these values, the signal components at the respective processor outputs are identical. By assigning a value to these equivalent weights, a desired frequency response in the look direction is selected. This introduces J constraint conditions. Since there are N X J adjustable weights, the remaining N X J - J degrees of freedom can be used to minimize the non-look direction noise power. Minimizing non-look direction noise power is equivalent to minimizing total output power because, regardless of how the weights are adjusted, the constraints guarantee that the response in the look direction will not be degraded.

The basic manner in which the Constrained LMS algorithm operates has been discussed. For the purpose of clarity, the primary steps taken by the algorithm will now be re-emphasized. Delays in the spatial correction filter

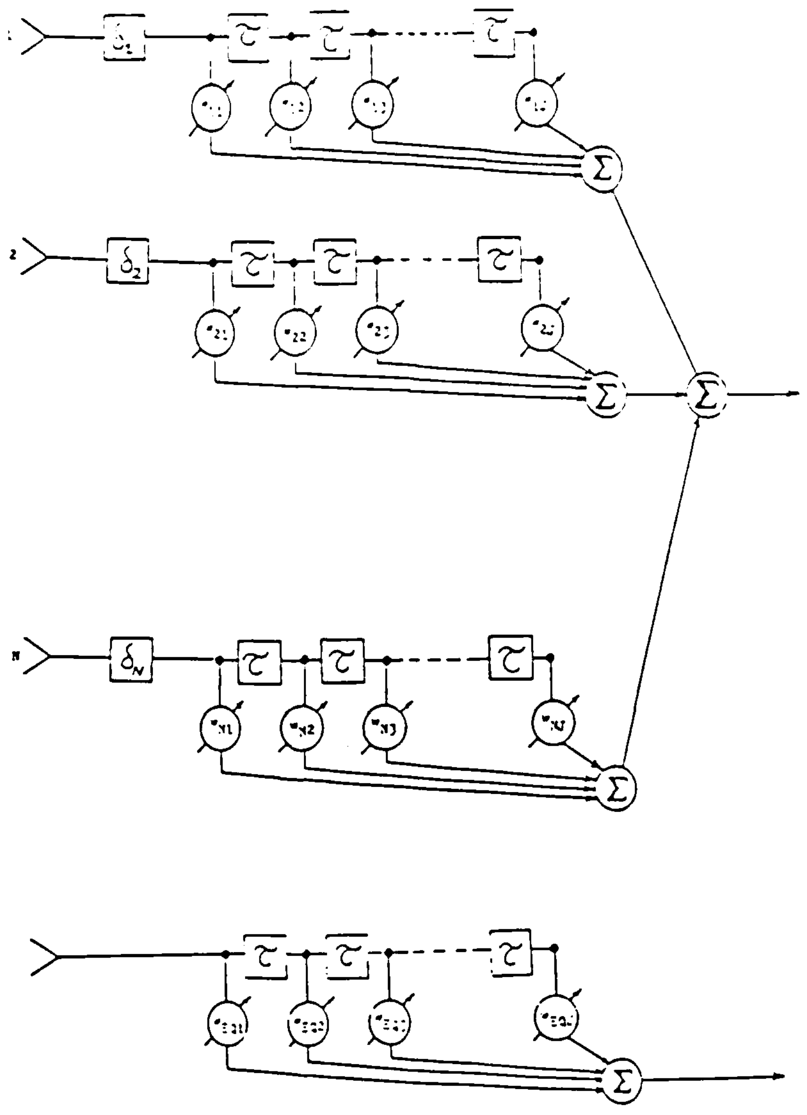


Figure 4.1 Signal-aligned Broadband Adaptive Array System

are calculated to align the communication signal components on the sensors. A desired response in the look direction is selected by assigning weight values to the equivalent processor (the sum-on-column constraints are determined). Once these tasks have been completed, adaptation begins and the processor strives to minimize the total output power. The constraints guarantee that there is no possibility of reducing power contributions made by the communication signal. Mathematical derivations of the optimum constrained weight solution and the Constrained LMS algorithm will now be presented.

4.3 Derivation of Optimum Constrained Weight Solution

The assumptions and definitions will be discussed first. Recall that the signals at the sensor element outputs can be written as the combination of the signal component and noise components

$$\underline{x}(k) = \underline{s}(k) + \underline{n}(k)$$

It is assumed that both the signal and noises can be modeled as zero-mean random processes with unknown second-order statistics. The covariance matrices are defined as follows:

$$E\{\underline{x}(k)\ \underline{x}^T(k)\} = R_{XX}$$

$$E\{\underline{s}(k)\ \underline{s}^T(k)\} = R_{SS}$$

$$E\{\underline{n}(k)\ \underline{n}^T(k)\} = R_{NN}$$

It is also assumed that the signal component is uncorrelated with the noise components.

$$E\{\underline{n}(k)\ \underline{s}^T(k)\} = 0$$

Finally, the expected value of the array output power is given by

$$E\{y^2(k)\} = E\{\underline{w}^T\ \underline{x}(k)\ \underline{x}^T(k)\ \underline{w}\} = \underline{w}^T\ \underline{\underline{R}}_{XX}\ \underline{w} \qquad (4.1)$$

Recall that the adaptive weights in the equivalent processor dictated the frequency response characteristic in the look direction. Define a J-dimensional vector that guarantees the desired frequency response and represents the summed weight values of the j vertical columns as

$$\underline{f} = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_N \end{bmatrix} \qquad (4.2)$$

The weights in the jth vertical column must sum to the selected number f_j. This constraint condition can be expressed as

$$\underline{c}_j^T \underline{w} = f_j \quad j = 1, 2, 3, \cdots, N \qquad (4.3)$$

where $\underline{c}_j$ is an NJ-dimensional vector consisting of all zeros and N ones given by

$$\underline{c}_j^T \; [\underset{N}{000..0}....\underset{N}{000..0}....\underset{N}{111..1}....\underset{N}{000..0}....\underset{N}{000..0}] \qquad (4.4)$$

A constraint matrix can then be defined that satisfies all j equations given by (4.3) as

$$\underline{C} = [c_1 \cdots c \cdots c_J] \qquad (4.5)$$

The full set of constraints can then be written as

$$\underline{C}^T \underline{w} = \underline{f} \qquad (4.6)$$

Although it seems like a complicated process, the constraint matrix $\underline{C}$ simply guarantees that the sum of the weights in the vertical columns is equal to the weights in the equivalent processor.

The constrained optimization problem statement can now be formulated. The array output power, $\underline{w}^T \underline{Rxx}\ \underline{w}$, must be minimized subject to the constraint condition $\underline{C}^T \underline{w} = \underline{f}$.

The optimum weight vector is found by using LaGrange multipliers. A cost function, similar in purpose to the MSE function of the LMS algorithm, is formed by concatenating the constraint equation with a J-dimensional vector of undetermined LaGrange multipliers $\underline{\lambda}$. This cost function is then minimized with respect to the weights.

$$\text{Cost}(w) = {}^1\!/_2\, \underline{w}^T\ \underline{R}\ \underline{w} + \underline{\lambda}\ [\underline{C}^T\ \underline{w} - \underline{f}] \qquad (4.7)$$

(a factor of $^1\!/_2$ is added to simplify the arithmetic)

Once again, notice that the cost function is a quadratic function of the weights. It is known that the gradient of this function is zero at the minimum point. The optimum weights are then found by finding the gradient of the function and setting it equal to zero.

The gradient of the cost function is found by differentiating with respect to the weights.

$$\nabla_{COST} = \underline{Rxx}\ \underline{w} + \underline{C}\ \underline{\lambda} \qquad (4.8)$$

Setting this result equal to zero yields the optimal weight solution.

$$\underline{Rxx}\ \underline{w} + \underline{C}\ \underline{\lambda} = 0$$

$$\underline{w}_{OPT} = -\ \underline{Rxx}^{-1}\ \underline{C}\ \underline{\lambda} \qquad (4.9)$$

The LaGrange multipliers are yet to be determined. They can be found by realizing that the optimal weight solution must satisfy the constraint condition.

$$\underline{C}^T\ \underline{w}_{OPT} = \underline{f} = C^T[-\underline{Rxx}^{-1}\ \underline{C}\ \underline{\lambda}]$$

$$\underline{\lambda} = -\ [\underline{C}^T\ \underline{Rxx}^{-1}\ \underline{C}]^{-1}\ \underline{f} \qquad (4.10)$$

The optimum constrained weight vector can now be expressed as

$$\underline{w}_{OPT} = \underline{Rxx}^{-1}\,\underline{C}[\underline{C}^T\,\underline{Rxx}^{-1}\,\underline{C}]^{-1}\,\underline{f} \tag{4.11}$$

4.4 Derivation of Constrained LMS Algorithm

As in the LMS algorithm, this algorithm uses the Method of Steepest Descent. Recall that this method states that the new weight vector is equal to the previous weight vector plus a change proportional to the negative gradient.

$$w(k + 1) = w(k) - \mu\,\nabla_{COST}$$

In this case, the update equation can be expressed as

$$\underline{w}(k + 1) = \underline{w}(k) - \mu[\underline{Rxx}\,\underline{w}(k) + \underline{C}\,\underline{\lambda}(k)] \tag{4.12}$$

The initial weight vector, w(0), must satisfy the constraint condition. It is chosen as

$$\underline{w}(0) = \underline{C}(\underline{C}^T\,\underline{C})^{-1}\,\underline{f} \tag{4.13}$$

The updated weight vector must satisfy the constraint condition as well. This can be written as

$$\underline{f} = \underline{C}^T\,\underline{w}(k + 1) = \underline{C}^T[\underline{w}(k) - \mu(\underline{Rxx}\,\underline{w}(k) + \underline{C}\,\underline{\lambda}(k))]$$

The LaGrange multipliers, λ(k), are then given by

$$\lambda(k) = -[\underline{C}^T\,\underline{C}]^{-1}\underline{C}^T\,\underline{Rxx}\,\underline{w}(k) - \frac{1}{\mu}\,[\underline{C}^T\,\underline{C}]^{-1}[\underline{f} - \underline{C}^T\,\underline{w}(k)] \tag{4.14}$$

and the iterative relation for the update equation is expressed as

$$\underline{w}(k+1) = \underline{w}(k) - \mu[\underline{I} - \underline{C}(\underline{C}^T\underline{C})^{-1}\underline{C}^T]\underline{Rxx}\,\underline{w}(k) + \underline{C}(\underline{C}^T\underline{C})^{-1}[\underline{f} - \underline{C}^T\,\underline{w}(k)]$$

For the sake of convenience, two definitions are made. Define the NJ-dimensional vector as

$$\underline{\beta} = \underline{C}(\underline{C}^T \underline{C})^{-1} \underline{f} \qquad (4.16)$$

and the NJ X NJ matrix P as

$$\underline{P} = \underline{I} - \underline{C}(\underline{C}^T \underline{C})^{-1}\underline{C}^T \qquad (4.17)$$

where I is the identity matrix.

The update equation can then be rewritten as

$$\underline{w}(k+1) = \underline{P}[\underline{w}(k) - \mu\underline{Rxx}\ \underline{w}(k)] + \underline{\beta}$$

The covariance matrix $\underline{Rxx}$ is unknown, however, so an approximation of $\underline{Rxx}$ at the kth iteration, $\underline{x}(k)\ \underline{x}^T(k)$, is used. Recognizing the fact that $\underline{x}^T(k)\underline{w}(k) = y(k)$, the final update equation becomes

$$\underline{w}(k+1) = \underline{P}[\underline{w}(k) - \mu y(k)\ \underline{x}(k)] + \underline{\beta} \qquad (4.18)$$

4.5 Constrained LMS Simulation Model

The tapped delay line in the broadband processor enables the user tc select a desired frequency response in the look direction. This study is not concerned with such filtering because narrowband signal models are being used. For the purposes of this simulation, it is only necessary that the response of the adaptive array in the look direction be equal to unity. This response can be achieved in the broadband model by setting one weight in the equivalent processor equal to one and the remaining weights equal to zero. This is somewhat wasteful, however, as the same response can be obtained using the simplified model shown in Figure 4.2. The explanations and definitions that have been presented are still valid, but the tapped-delay line in the broadband model now consist of a single tap (weight). The weight in the equivalent processor is assigned a value of $1 < 0°$ so that the adaptive array has an all-pass distortionless response in the look direction. The sum of the

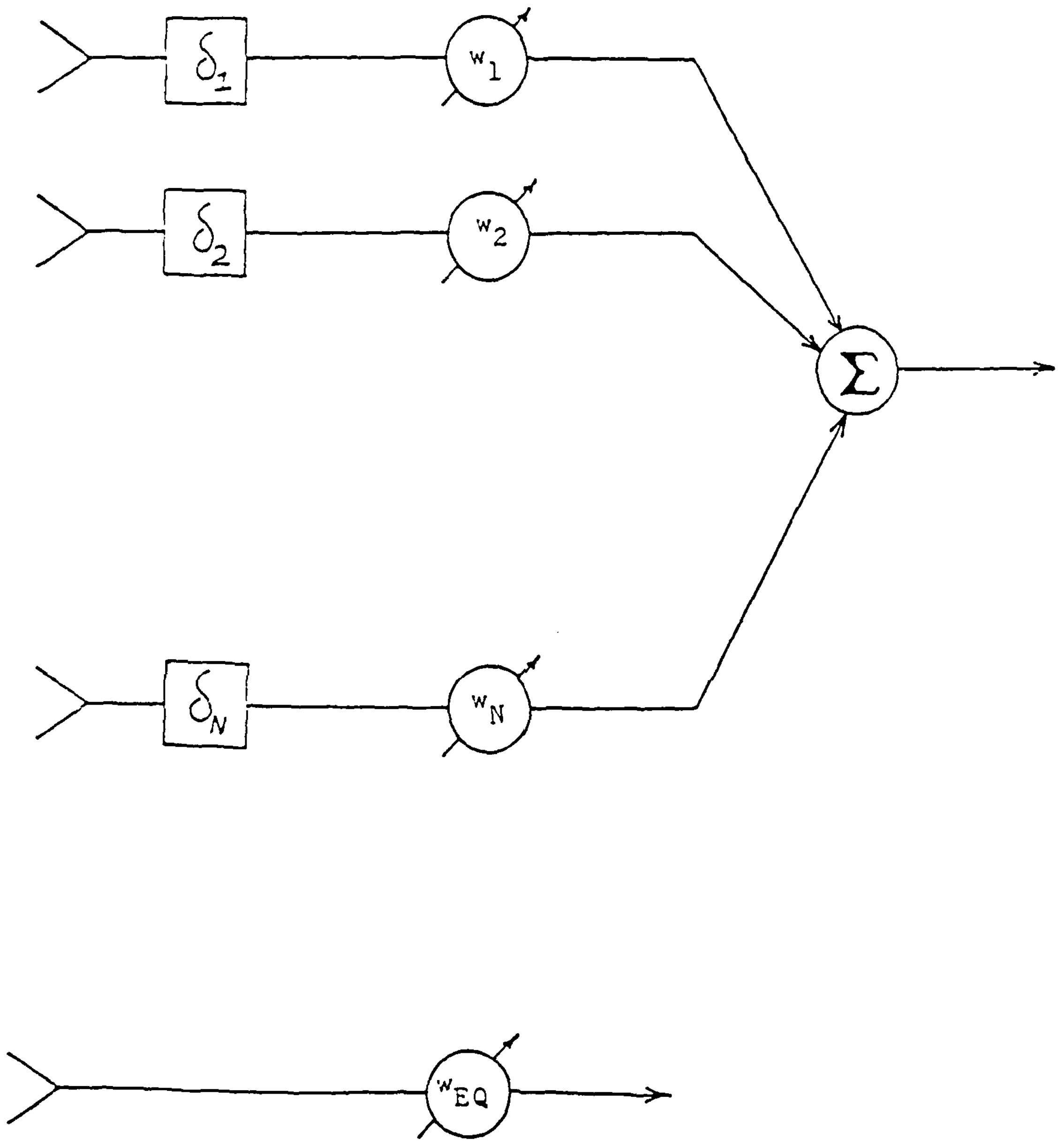


Figure 4.2 Narrowband Signal-aligned array system

weights in the single vertical column of the original processor are still required to equal that of the equivalent processor.

4.5.1 TAKAO Implementation

The Constrained LMS algorithm requires a spatial correction filter to compensate for the misalignment of the sensor elements. A method proposed by Takao et. al., [6] merges the misalignment compensation and the weight computation into a single process. The direction of arrival of the communication signal is used to generate a directional constraint to govern the weights. This method has been used in this simulation. An additional benefit of this implementation is that it helps to isolate the Constrained LMS processor from the other array system components, which was desired from the outset.

A description of the Takao implementation will now be presented. Note the similarities between these and the original equations [5], as they are for the most part equivalent.

$\underline{f} = 1 \angle 0°$ (only one weight in equivalent processor)

$(\underline{c}^*)^T \underline{w} = \underline{f}$ constraint equation

$\underline{w} = (w_1 + jw_2, w_3 + jw_4 , \cdots w_{2N} - 1 + jw_{2N})$

$\underline{c}^T = (e^{j\Omega_1}, e^{j\Omega_2}, \cdots, e^{j\Omega_N})$

where Ω_i the phase of the desired signal at sensor element i

$\underline{P} = \underline{I} - (\underline{c}^* \underline{c})/N$

$\underline{\beta} = \underline{c}/N$

$\underline{w}(k + 1) = \underline{P} [\underline{w}(k) - \mu\, y(k)\, \underline{x}^*(k)] + \underline{\beta}$ update equation

4.6 Constrained LMS Software Modules

The weights are initialized using the WEIGHTINIT routine. They are assigned a value of

$$w_i(0) = \text{steering delay } i/\text{number of elements}$$

The WEIGHTINIT subroutine is also responsible for calculating the individual steering delays that compensate for misalignment. These are given as

$$\text{steering delay } i = \exp(-j\Omega_i)$$

where Ω_i is the phase of the desired signal at element i.

Several quantities are calculated from directional information the first time the weight update routine CONLMS is called. These include

$$\underline{P} = \underline{I} - (\underline{c}\ ^{*}\ \underline{c})/N$$

$$\underline{\beta} = \underline{c}/N$$

$$\underline{c}^T = [e^{j\Omega_1}, e^{j\Omega_2}, \cdots, e^{j\Omega_N}]$$

Each time the routine is called, a new weight is calculated using

$$\underline{w}(k+1) = \underline{P}\ [\underline{w}(k) - \mu y(k)\ \underline{x}^{*}(k)] + \underline{\beta}$$

These updated weights are then passed to the pattern-forming network.

The FORTRAN source code listings are given in Appendix C. Figure 4.3 depicts the modules discussed and illustrates the primary input and output parameters. The variable names used in the program are shown in parenthesis.

The Constrained LMS algorithm also has some limitations. Although it does not require a reference signal, it does require prior information pertaining to the communication signal. It is vulnerable to steering error. Steering error arises if the supposedly known angle of arrival of the

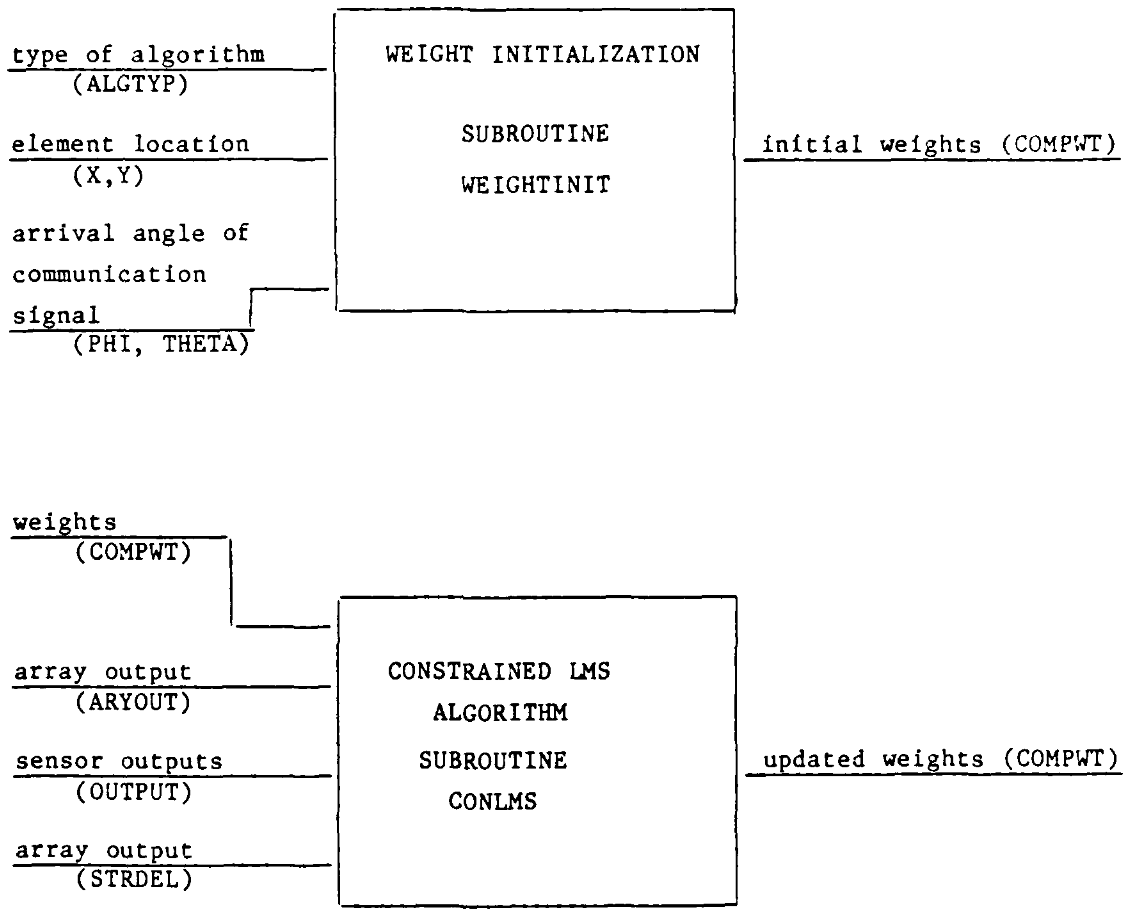


Figure 4.3 Constrained LMS Software Modules

information signal changes appreciably. In that event, the algorithm treats the signal as it would any other non-look direction signal — by placing a deep null in its direction of arrival. The algorithm would have no way of knowing that the "interference" it is trying to ignore carries desired information. Secondly, its convergence rate is comparable to that of the LMS algorithm. An algorithm that possesses a faster response will now be examined.

5.0 UPDATE COVARIANCE ALGORITHM

Both the LMS and Constrained LMS algorithms circumvent computational problems associated with the direct calculation of a set of weights by using effective estimates. The simpler calculations that result allow them to frequently update the weights in order to compensate for the time-varying environment. Recursive processors such as the Update Covariance algorithm presented by Monzingo and Miller [7] can also be used to avoid these computational difficulties. These algorithms recursively perform matrix inversion so that direct matrix inversion is never required. Although they also avoid direct matrix inversion, recursive processors represent a significant departure from the algorithms previously discussed.

Recall that the optimum weight solution given by Weiner-Hopf can be expressed as

$$\underline{w}_{OPT} = \underline{\underline{RXX}}^{-1}\,\underline{P}$$

The Update Covariance algorithm and other recursive processors recursively estimate the sample covariance matrix rather than rely on gradient methods that asymptotically approach an optimal solution. These algorithms calculate the optimal set of weights at each sampling instant based on a least-squares fit to the received data.

5.1 Motivation for Selection

The primary reason for the selection of the Update Covariance algorithm is its speed of convergence. This characteristic, which is common among recursive processors, allows the algorithm to respond to changes more rapidly than other methods of adaptation. It is simply a more direct approach to the problem of computing an optimal weight solution. Updating the complex weights does not involve descending along a performance or cost surface at a limited rate.

Another beneficial quality of the Update Covariance algorithm is that no reference signal is required for adaptation. Once again, this eliminates the need for complicated reference generation techniques. It does, however, require initial directional information pertaining to the communication signal.

Finally, recursive processors hold good promise for the future. They require a digital implementation and this has been, and still is, their primary disadvantage. The great technological strides made in the production of very fast, inexpensive, and compact digital hardware, however, have resulted in the consideration of recursive processors for applications that were previously out of the question. The improved convergence rates (measured in terms of iterations) offered by recursive processors have been documented [8] and, if technological trends continue, it may become implementationally feasible to exploit this advantage.

5.2 Update Covariance Algorithm Description

As mentioned, the Update Covariance processor estimates the sample covariance matrix in order to calculate an optimal weight solution. Unlike the other algorithms described, the operation of this algorithm can be described as a series of complex computations solely intended to calculate the optimal weight solution.

As the name implies, the Update Covariance algorithm uses the sample covariance estimate, $\underline{R}$, to summarize the effect of de-emphasizing the past data. The new sample covariance matrix estimate is given by

$$\underline{\underline{\hat{R}}}_{XX}(k+1) = \alpha \, \underline{\underline{R}}_{XX}(k) + \underline{x}^*(k+1) \, \underline{x}^T(k+1) \tag{5.1}$$

The new estimate is equal to the new computed value $\underline{x}^*(k+1)\ \underline{x}^T(k+1)$ plus the past estimate scaled by a factor of α. α is a number between 0 and 1 that is used to determine the significance of past data. The inverse estimate then becomes

$$\underline{\underline{\hat{R}}}_{XX}{}^{-1}(k+1) = \frac{1}{\alpha} \left[\underline{\underline{R}}_{XX}(k) + \frac{1}{\alpha} \underline{x}^*(k+1) \, \underline{x}^T(k+1)\right]^{-1} \tag{5.2}$$

Note that calculating the inverse in this manner however, would require matrix inversion, which is exactly what the algorithm is trying to avoid. Therefore it is useful to invoke the following matrix identity

$$[\underline{P}^{-1} + \underline{M}^{*T} \underline{Q}^{-1} \underline{M}]^{-1} = \underline{P} - \underline{PM}^{*T} [\underline{MPM}^{*T} + \underline{Q}]^{-1} \underline{MP} \tag{5.3}$$

This identity is applied to equation 5.2 to obtain $\hat{\underline{R}}_{XX}^{-1}(k+1)$ in the form

$$\hat{\underline{R}}_{XX}^{-1}(k+1) = \frac{1}{\alpha} [\hat{\underline{R}}_{XX}^{-1}(k) - \frac{\hat{\underline{R}}_{XX}^{-1}(k)\ \underline{x}^{*}(k+1)\ \underline{x}^{T}(k+1)\ \hat{\underline{R}}_{XX}^{-1}(k)}{\alpha + \underline{x}^{T}(k+1)\ \hat{\underline{R}}_{XX}^{-1}(k)\ \underline{x}^{*}(k+1)} \tag{5.4}$$

The optimum weight solution can then be found by utilizing the Weiner-Hopf equation

$$\underline{w}_{OPT} = \underline{R}_{XX}^{-1}\ \underline{p}$$

Multiplying both sides of equation 5.4 by the vector $\underline{p}$ yields the Update Covariance weight update equation.

$$w(k+1) = \frac{1}{\alpha} [w(k) - \frac{\underline{R}_{XX}^{-1}(k)\ \underline{x}^{*}(k+1)\ \underline{x}^{T}(k+1)\ \underline{w}(k)}{\alpha + \underline{x}^{T}(k+1)\ \underline{R}_{XX}^{-1}\ \underline{x}^{*}(k+1)} \tag{5.5}$$

The Update Covariance algorithm consists of the following two steps:

1. The inverse sample covariance estimate is formed using equation 5.4.

2. The weight solution is calculated using equation 5.5.

Due to the fact that the Update Covariance algorithm can be thought of as an entirely mathematical process, the simulation model will be dispensed with. The software associated with this algorithm simply performs the computations outlined in equations 5.4 and 5.5.

5.3 Update Covariance Software Modules

The weights are initialized using the WEIGHTINIT subroutine. These weights contain directional information which initially steers the antenna in the direction of the desired signal.

The first time the weight update routine UPDCOVAR is called, the inverse sample covariance estimate initialized to the identity matrix. After the first call, a new sample covariance estimate is formed by performing the necessary computations. This result is then used to calculate the new weight vector which is then passed to the pattern-forming network.

The FORTRAN source code is given in Appendix C. Figure 5.1 depicts the modules that have been discussed and illustrates the primary input and output parameters. The variable names used in the routines are shown in parenthesis.

By examining the update equations, it becomes quite evident that the Update Covariance algorithm, or any recursive processor for that matter, is computationally intensive. Although recursive estimation produces significant processing savings when compared to direct matrix inversion, it still represents somewhat of a quantum leap in terms of complexity when compared to the other algorithms presented.

Recall that the LMS algorithm required on the order of 2N computations to update the weights, where N is the number of sensor elements. The Update Covariance processor requires on the order of $5N^2$ computations to update the weights. Therefore, although the recursive processors require fewer iterations to converge, it may take a great deal of time to complete each iteration. For applications large in size (those having many elements), recursive processing may offer no improvement in actual convergence time over the gradient-based algorithms.

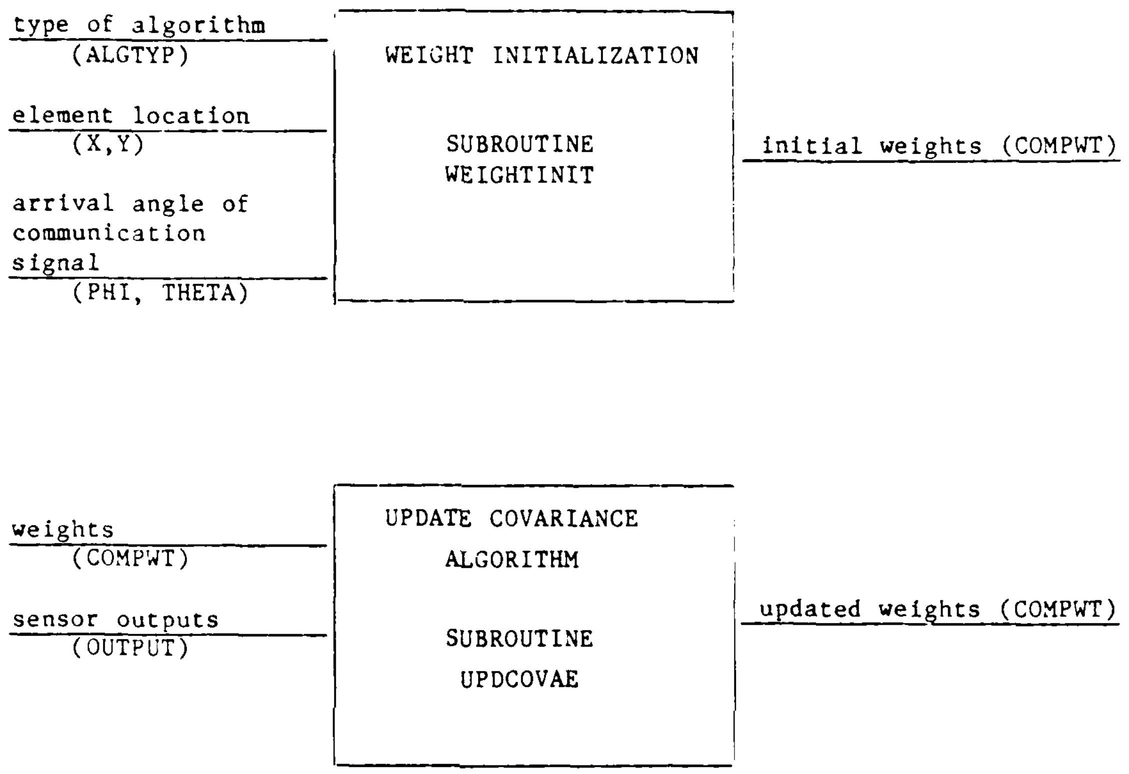


Figure 5.1 Update Covariance Software Modules

6.0 DESCRIPTION OF THE HF ADAPTIVE ARRAY SIMULATION MODEL

The purpose of this section is to explain the computer simulation program that has been developed to study adaptive algorithms for HF antenna arrays. All parameters that the user must specify will be discussed first. These parameters define the signal/interference environment as well as the array system to be tested. Secondly, the method used to evaluate the performance of the algorithms will be examined. Finally, the simulation software including signal models and the operation of the program will be presented.

6.1 User-Definition of Array System and Environment

Before an adaptive array system can be evaluated, the user must define the array system and environment to be studied. The parameters that must be specified can be divided into the following classes:

1. Signal characteristics and environment
2. Interference environment
3. Array system
4. HF channel characteristics
5. Convergence characteristics

A complete listing of the user-specified parameters is given in Figure 6.1. Appendix B contains the user manual for the simulation, giving complete descriptions of the above parameters, as well as the actual user interface. In addition to a description of the input process, the manual also defines user options for viewing the simulation output.

6.2 Performance Evaluation

The performance of the adaptive algorithms will be evaluated using a maximum signal-to-noise criterion. This criterion will now be examined.

Recall that the output of the array can be expressed as

$$y(t) = \underline{w}^T \underline{x}(t)$$

	UNITS
<u>ARRAY SYSTEM</u>	
Number of sensor elements	---
Length of antennas	meters
Location of elements in rectangualr coordinators	meters
Adaptive algorithm	---
<u>SIGNAL CHARACTERISTICS</u>	
Arrival angles, azimuth and elevation	degrees
Number of Samples/bit	---
<u>INTERFERRENCE CHARACTERISTICS</u>	
Number of Jamming signals	---
Arrival angles, azimuth and elevation	degrees
J/S ratio	dB
S/N (thermal) ratio	dB
<u>HF CHANNEL CHARACTERISTICS</u>	
Number of signal paths	---
Delay of each path	msecs
Attenuation of paths	dB
<u>CONVERGENCE CHARACTERISTICS</u>	
Number of Convergences	---
SNR tolerance	dB

Figure 6.1 User-entered parameters

where x(t) contains both signal and noise components.

The array output can then be divided into signal and noise components.

$$y_s(t) = \underline{w}^T \underline{s}(t) \qquad y_n(t) = \underline{w}^T \underline{n}(t)$$

The expected signal and noise power at the array output is given as

$$E\{|y_s(t)|^2\} = \overline{|\underline{w}^T \underline{s}|^2} = \underline{w}^{*T} [\underline{s}^* \underline{s}^T] \underline{w} = \underline{w}^{*T} \underline{\underline{R}}_{ss} \underline{w}$$

$$E\{|y_n(t)|^2\} = \overline{|\underline{w}^T \underline{n}|} = \underline{w}^{*T} [\underline{n}^* \underline{n}^T] \underline{w} = \underline{w}^{*T} \underline{\underline{R}}_{nn} \underline{w} \qquad (6.1)$$

Therefore, the signal-to-noise (noise + interference) ratio can be calculated as

$$SNR = \frac{\underline{w}^{*T} \underline{\underline{R}}_{ss} \underline{w}}{\underline{w}^{*T} \underline{\underline{R}}_{nn} \underline{w}} \qquad (6.2)$$

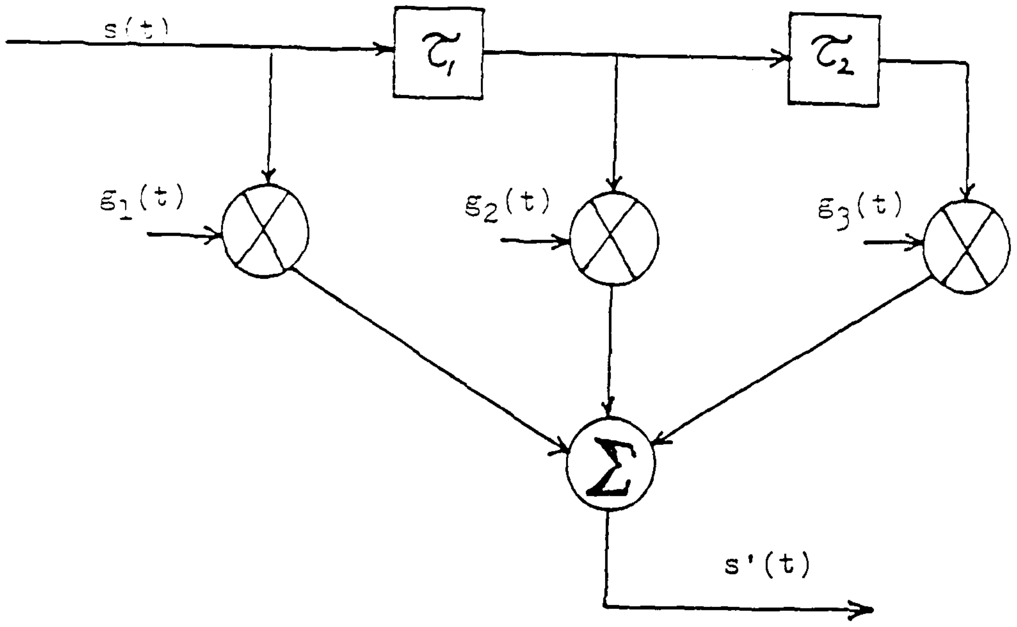
The optimum SNR can be computed using a matrix transformation as given by Monzingo and Miller [7]. The optimum SNR is given as

$$SNR_{OPT} = \underline{s}^T \underline{\underline{R}}_{nn}^{-1} \underline{s}^* \qquad (6.3)$$

As the name implies, the maximum achievable SNR is used to evaluate algorithm performance. The goal (optimum SNR) is known, and by computing the current SNR, it is possible to observe the degree of success that the algorithm is having in attaining this goal. The model is executed until it has reached a user set SNR goal.

This study is primarily interested in finding the average number of times that a particular algorithm must update the weights (iterations) before it has converged as function of the HF channel. It is useful to first consider a three-path HF channel model as depicted by Figure 6.2. The HF channel is characterized by the delays between the paths and the attenuation of each path (determined by the variances of the random complex numbers, $g_i(t)$). By selecting a set of delays and attenuations, an HF channel model is defined.

THREE-PATH CHANNEL MODEL



$g_i(t)$ is a complex gaussian zero-mean random process

The variance of $g_i(t)$ determines the attenuation of the respective path

Figure 6.2 Three-path HF Channel Model

Notice, however, that it is possible to get different channel "realizations" using the same characteristic channel model (same delays and variances) by simply using different random numbers. This plays an important role in the performance evaluation.

Due to the fact that the HF channel is slowly varying, a single channel realization is selected for each adaptation. The optimum SNR, which is channel dependent, is then calculated and adaptation begins. The current SNR is periodically calculated and if this value is not sufficiently close to the optimum value (proximity to optimum entered by user and known as SNR tolerance), the adaptation process is continued. When the SNR does approach the optimum SNR, the algorithm is said to be "converged" and the number of required iterations is recorded. A new HF channel realization (a new set of random numbers for the tap weights) is then selected. The entire process is repeated until the algorithm has converged the specified number of times (user-entered). At that time, the average number of iterations is calculated.

By performing this simulation for each of the selected algorithms, it will be possible to determine, on the average, which algorithm converges in the fewest number of iterations in an HF environment.

6.3 Simulation Software Description

The purpose of this section is to explain the signal models and the operation of the simulation program used in the study.

6.3.1 Desired Signal Model

The signal that is to be received is assumed to be a BPSK waveform at a carrier frequency f_c. This can be written in complex form as

$$s(t) = A(t) \sqrt{P_s} \exp (jw_c t)$$

where

P_s = desired signal power and

$A(t) = +1, -1$

The signal component at element i can be written as

$$s_i(t) = A(t) \sqrt{P_s} \exp(jw_c t) \exp(j\Omega_i) \qquad (6.4)$$

where Ω_i is the phase of the desired signal at element i.

Ω_i is calculated using the arrival angle information and the location of the elements.

$$\Omega_i = 2*\pi*f_c * \text{xrot}(i)*\sin\Theta/c \qquad (6.5)$$

where

$\text{xrot}(i) = y(i)\sin\phi + x(i)\cos\phi$

$x(i),y(i)$ = rectangular coordinates of element i

ϕ = azimuthal arrival angle of desired signal

θ = elevation arrival angle of desired signal

c = speed of light

Using complex envelope representation, the carrier component can be dropped from the notation. The jammer power and thermal noise power are defined as ratios relative to the signal power, P_s. For the sake of simplicity, the signal power is assigned a value of 1.

Equation 6.4 can be written for discrete sampling as

$$s_i = A(k) \sqrt{P_s} \exp(j\Omega_i)$$

The signal vector then becomes

$$\underline{s}(k) = A(k) \sqrt{P_s} \begin{bmatrix} e^{j\Omega 1} \\ e^{j\Omega 2} \\ \cdot \\ \cdot \\ \cdot \\ e^{j\Omega N} \end{bmatrix}$$

6.3.2 Interference Model

The jamming signals in this study have been modeled as complex Gaussian noises. This can be expressed as

$$nj(t) = \sqrt{Pj/2}\ [E(t) + j\ F(t)] \qquad (6.6)$$

where E(t) and F(t) are zero-mean random processes with a gaussian distribution and a variance of 1. The power of the jammer is calculated from the user-specified jammer-to-signal ratio.

The jamming signal component at each element is defined as

$$nj_i(t) = nj(t)\ \exp(j\Omega_i) \qquad (6.7)$$

where Ω_i is the phase of the jamming signal at element i. It is calculated in the same manner previously discussed using directional and element location information.

6.3.3 Thermal Noise Model

In this study, thermal noise has been modeled by adding complex Gaussian Noise to each element. This can be expressed as

$$n_t(t) = \sqrt{P_n/2}\ [E(t) + j\ F(t)] \qquad (6.8)$$

where E(t) and F(t) are zero-mean processes with a Gaussian distribution and variance of 1.

The power, P_n, is calculated from the user-specified signal-to-noise ratio.

6.3.4 Correlation Matrices

In order to evaluate the performance using a maximum signal-to-noise criterion, it is necessary to compute correlation matrices. These will now be defined.

Recall that the desired signal can be expressed as

$$s_i(k) = A(k)\sqrt{P_s}\exp(j\Omega_i)$$

The correlation matrix can then be written as

$$\underline{\underline{R}}_{ss} = P_s \begin{bmatrix} e^{-j0} & e^{-j(\Omega_1 - \Omega_2)} & \cdots & e^{-j(\Omega_1 - \Omega_N)} \\ e^{-j(\Omega_2 - \Omega_1)} & e^{-j0} & \cdots & e^{-j(\Omega_2 - \Omega_N)} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ e^{-j(\Omega_N - \Omega_1)} & e^{-j(\Omega_N - \Omega_2)} & & e^{-j0} \end{bmatrix} \qquad (6.9)$$

The noise correlation matrix can be written as the sum of the individual noise matrices.

$$R_{nn} = R_{Jammer} + R_{NTHERMAL} \qquad (6.10)$$

where

$$\underline{R}_J = \begin{bmatrix} e^{-j0} & e^{-j(\Omega_1 - \Omega_2)} & \cdots & e^{-j(\Omega_1 - \Omega_N)} \\ e^{-j(\Omega_2 - \Omega_1)} & e^{-j0} & & e^{-j(\Omega_2 - \Omega_N)} \\ \cdot & \cdot & \cdot & \cdot \\ e^{-j(\Omega_N - \Omega_1)} & & & e^{-j0} \end{bmatrix} \qquad (6.11)$$

and $R_{NTHERMAL} = \sigma^2 \underline{I}$ (6.12)

As previously mentioned, the optimum SNR is given by

$$SNR_{opt} = \underline{s}^T \underline{\underline{R}}_{nn}^{-1} \underline{s}$$

The current SNR can be written as

$$SNR = \frac{\underline{w}^{*T} \underline{\underline{R}}_{ss} \underline{w}}{\underline{w}^{*T} \underline{\underline{R}}_{nn} \underline{w}}$$

The optimal value is calculated first. The simulation is started and adaptation begins. The SNR is periodically checked, and when it is sufficiently close to the optimum value the simulation is stopped.

6.3.5 Simulation Program Operation

The flow chart of Figure 6.3 depicts the order in which the described operations are performed. The following is a brief discussion of how the program operates.

The simulation begins with the user specifying the defining parameters such as the element locations, selected algorithm, arrival angles, and relative signal strengths, among others. The program must then calculate the optimum SNR for a particular channel realization. This value will be used to determine when the algorithm has converged. The antenna weights are then initialized and the adaptation process begins.

The desired signal and jamming signals are determined first using the models described. These signals are then passed to the HF channel model. These "channelized" signals are then passed to the antenna routine which calculates the contributions of all signals at each antenna element. The output of each sensor element is multiplied by its corresponding weight, and these products are summed to form an overall array system output. This overall output and the output of each element is then passed to the selected algorithm. The algorithm uses this information to adjust the weights.

The weights are updated by the selected algorithm and the current SNR is periodically computed. If this SNR is not close to the optimum value, adaptation continues. If it is, however, the converged weights and the number of convergence iterations are stored in a file. A new channel realization is selected, the parameters are re-initialized, and the process is repeated.

When the algorithm has converged a sufficient number of times (user-specified), overall statistics are gathered and the simulation is complete.

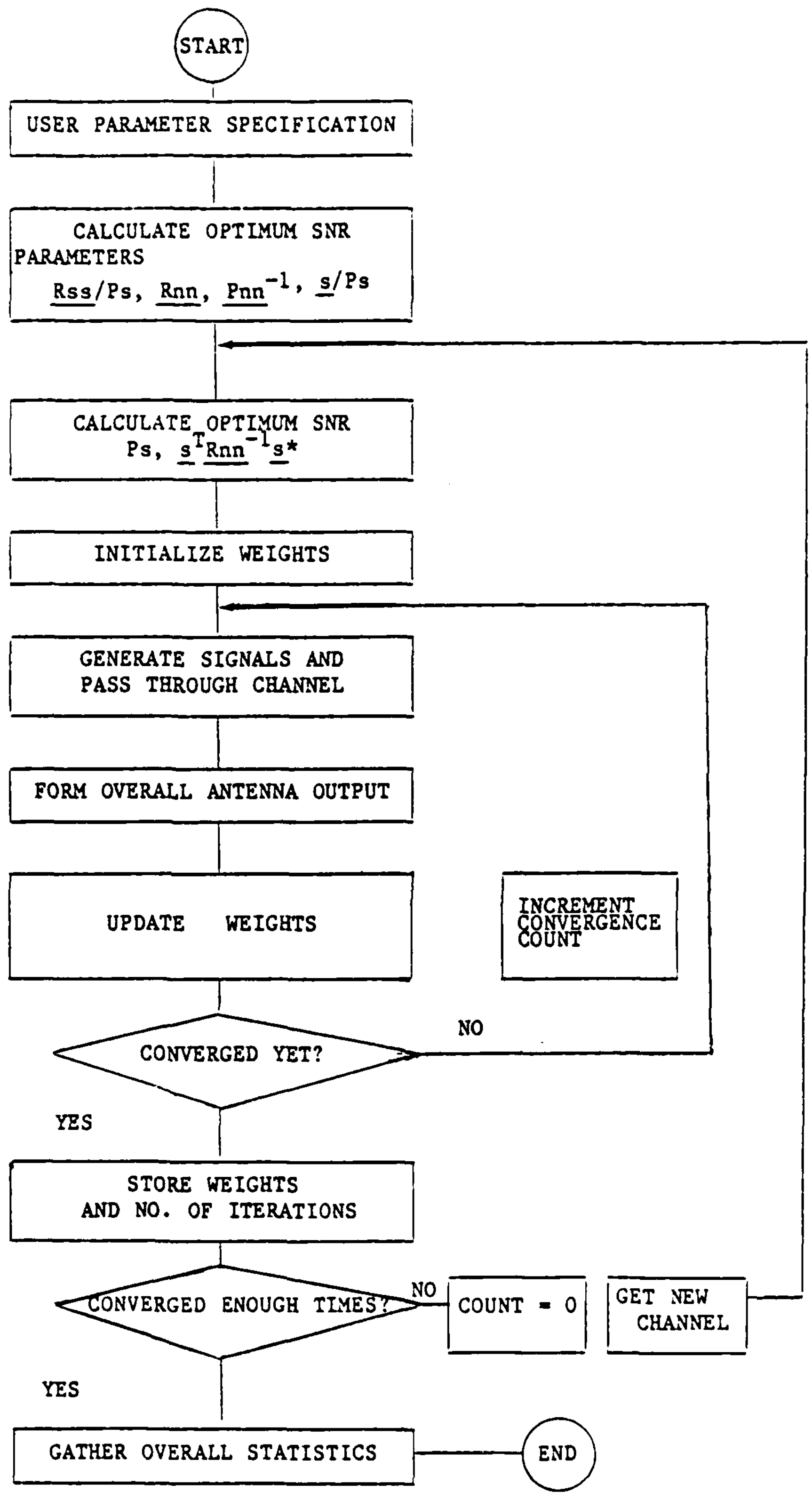

START
USER PARAMETER SPECIFICATION
CALCULATE OPTIMUM SNR PARAMETERS
Rss/Ps, Rnn, Pnn^{-1}, s/Ps
CALCULATE OPTIMUM SNR
Ps, s^{T}Rnn^{-1}s*
INITIALIZE WEIGHTS
GENERATE SIGNALS AND PASS THROUGH CHANNEL
FORM OVERALL ANTENNA OUTPUT
UPDATE WEIGHTS
INCREMENT CONVERGENCE COUNT
CONVERGED YET?
NO
YES
STORE WEIGHTS AND NO. OF ITERATIONS
CONVERGED ENOUGH TIMES?
NO
COUNT = 0
GET NEW CHANNEL
YES
GATHER OVERALL STATISTICS
END

Figure 6.3 Flow-diagram of Program Operation

7.0 SIMULATION RESULTS AND CONCLUSIONS

The purpose of this section is to illustrate the results that have been obtained using the simulation model described in the previous section. From these results, conclusions concerning the applicability of the particular algorithm to the problem of interest can be drawn.

In all of the tests to be conducted, certain parameters will remain constant. These include arrival angles and signal powers and are summarized in Figure 7.1. For each test, the average number of required convergence iterations will be given to demonstrate how quickly the algorithms approach optimality in the maximum-SNR sense. Also, polar plots will be presented in order to pictorially demonstrate the nulling capabilities of the algorithms. These plots consist of two "slices" of each antenna pattern at the azimuth and elevation angles for each incoming signal (see Figures 7.2 and 7.3 for a definition of the geometry).

SIGNAL	AZIMUTHAL ANGLE	ELEVATION ANGLE	POWER
COMMUNICATOR	90.0	60.0	0 dB
JAMMER 1	90.0	80.0	20 dB
JAMMER 2	150.0	75.0	20 dB

S/N(thermal) - 10 dB   Carrier frequency - 30 MHz

SENSOR ELEMENT LOCATIONS

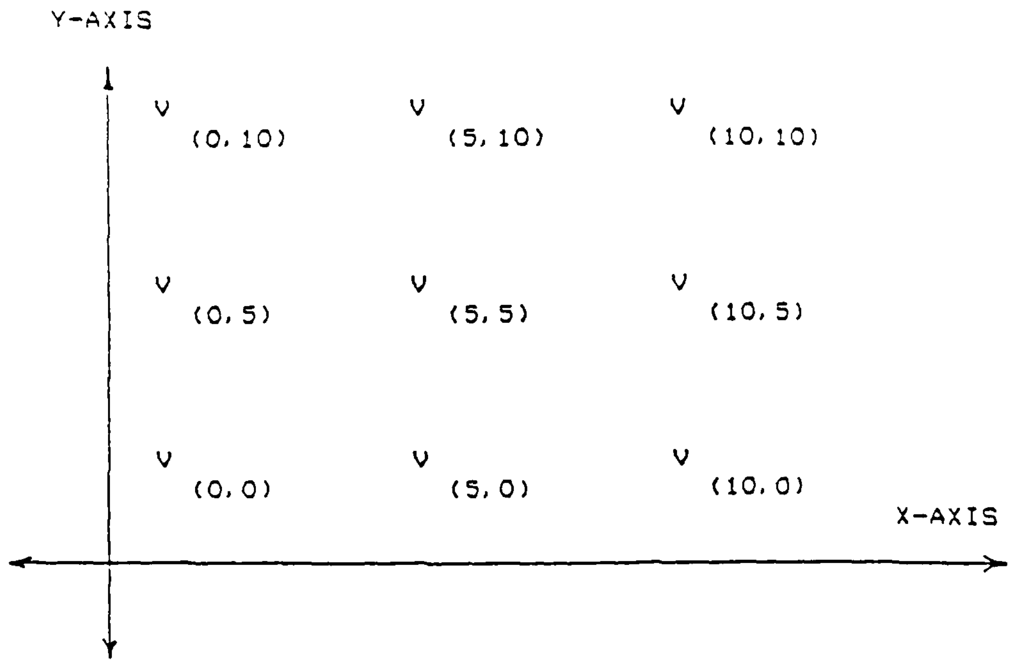


Figure 7.1 Constant Test Parameters

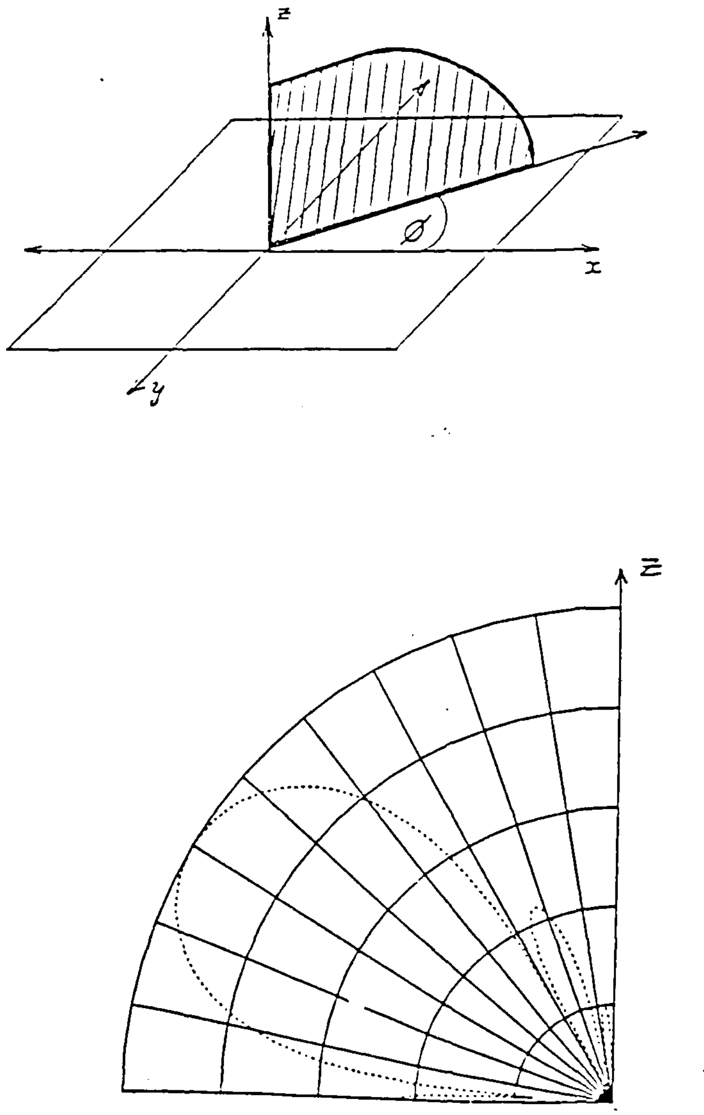
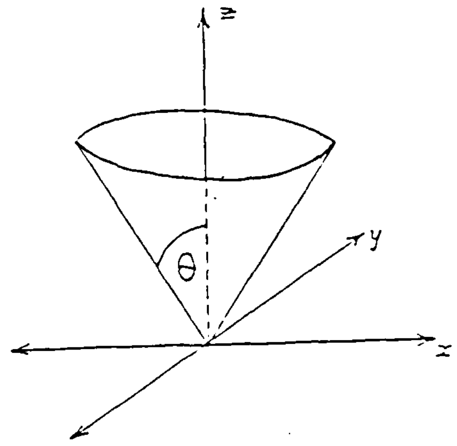


Figure 7.2 Description of Elevation Plots



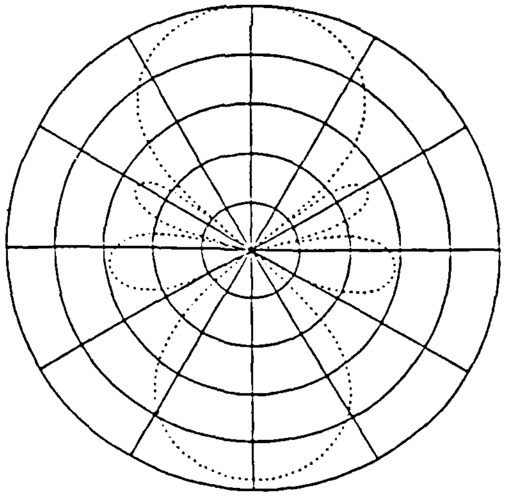


Figure 7.3 Description of Azimuthal Plots

7.1 Test 1: Negligible Channel Effects

It may be enlightening to first consider the performance of the algorithm in an unperturbed channel environment. In this way, the degrading effects of the HF channel can be monitored as well. Although it is not a terribly realistic case, it may become so if some efficient channel compensation technique could be employed. Recall that in this study the array model consists of one weight per element. This allows for beam steering, but the array cannot compensate for the smearing effects of the channel.

This case, although utilizing an ideal channel, is far from a trivial one. In fact, simulation studies frequently limit themselves to an ideal channel, because the task of nulling out strong jamming signals is a difficult one even in this setting.

The results for all algorithms were averaged over 100 convergences. The number of iterations for each convergence was recorded to produce the histograms of Figures 7.4 through 7.6. In all cases, the SNR tolerance (the proximity to the optimum SNR that determined covergence) was set to 1 dB. In other words, the algorithms "converged" when the SNR was within 1 dB of the optimum value. The results are tabulated in Table 7.1. The polar antenna plots that follow verify that all of the algorithms did an excellent job in placing nulls in the directions of the interferences.

Probably the most striking result obtained was the incredibly few iterations required by the Update Covariance algorithm to converge. This result can be misleading, however, due to the number of computations it requires per iteration. Recall that the update covariance algorithm requires about 5N complex computations for each iteration, where N is the number of antenna elements. The LMS and Constrained LMS algorithms require on the order of 2N and $5N^2$ computations per iteration respectively. Therefore, the difference in the actual number of required computations is not that great.

The Constrained LMS algorithm converged the slowest of the three algorithms. It is interesting to note that the weight changes of the LMS algorithm approach zero as the weights approach their optimal value. This can be seen from the LMS weight update equation listed below as

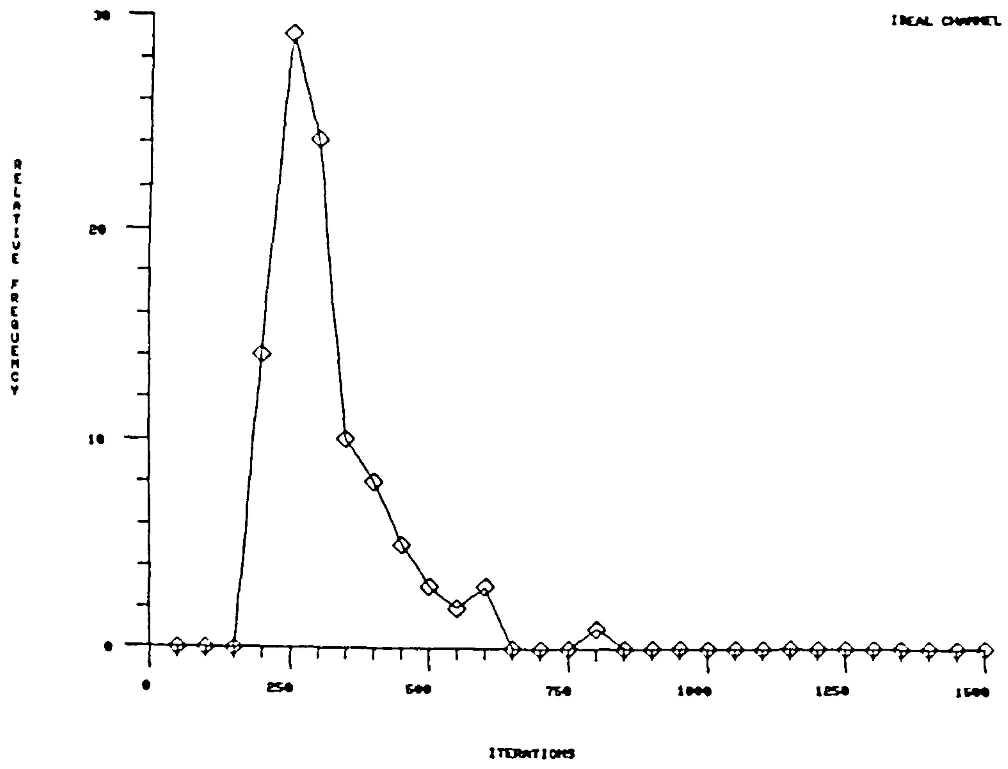


Figure 7.4 Convergence Histogram of LMS Algorithm in Ideal Channel

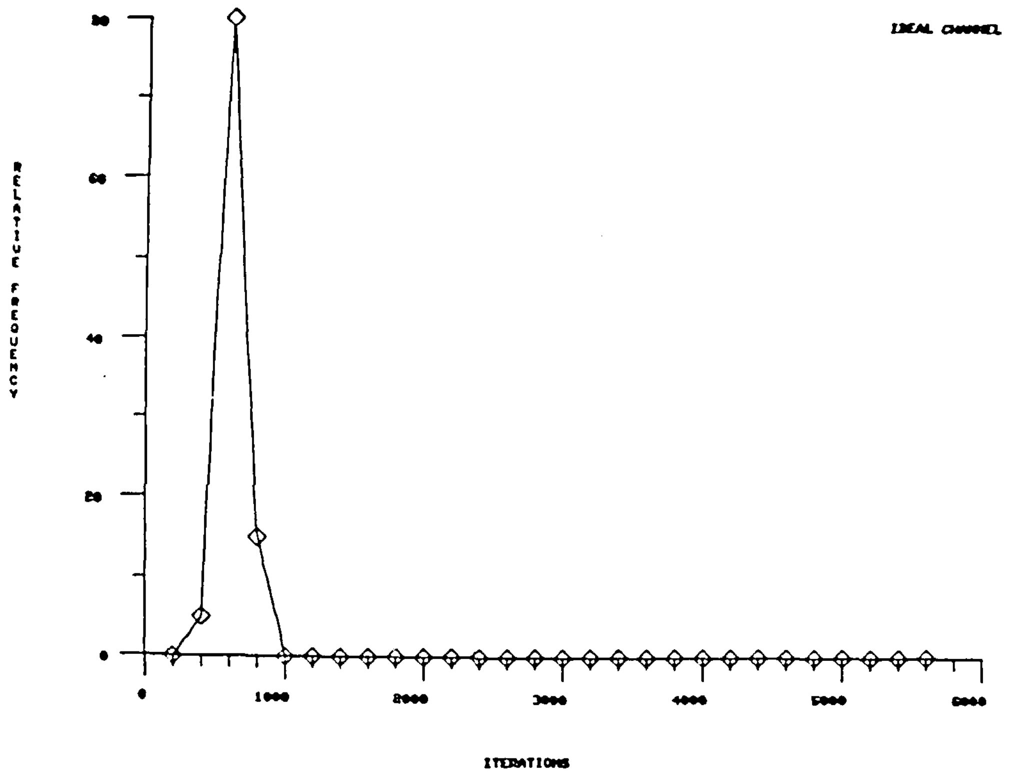


Figure 7.5 Convergence Histogram of Constrained LMS Algorithm in Ideal Channel

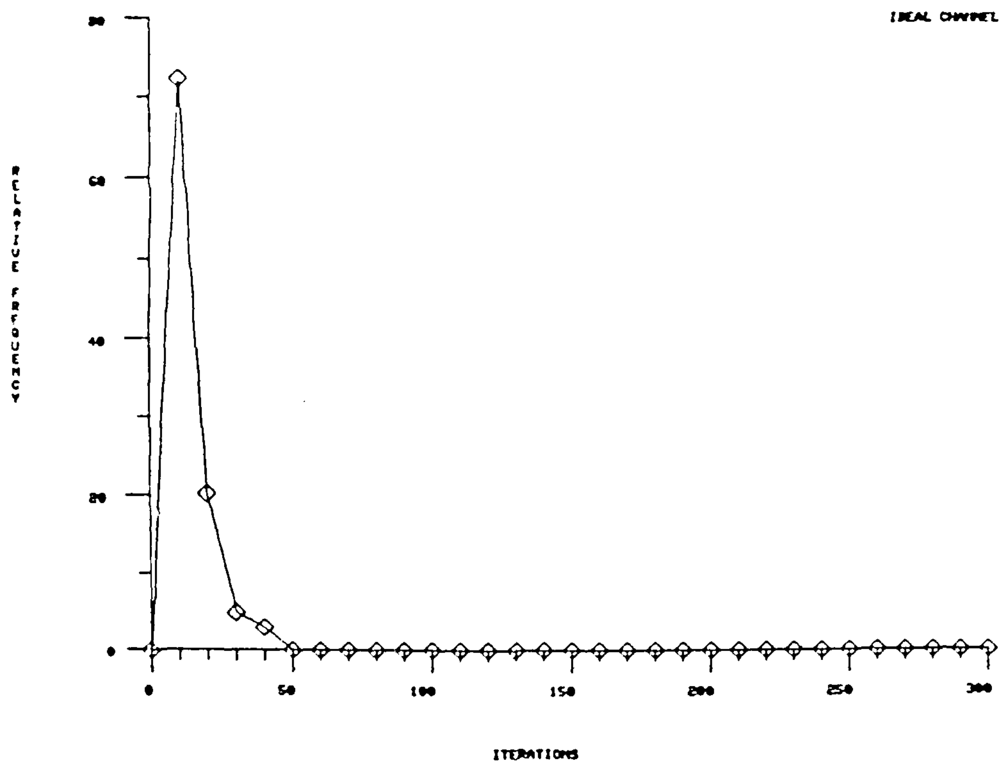


Figure 7.6 Convergence Histogram of Update Covariance Algorithm in Ideal Channel

Table 7.1 TEST1 Summary

Adaptive Algorithm	Number of Convergences	Average Number of Iterations	Standard Deviation
LMS	100	318.0	108.5
Constr. LMS	100	536.5	8.14
Update Cov.	100	9.9	6.7

$$W(k+1) = W(k) + \mu\ e(k)\ X^{*}\ (k)$$

As the error decreases, so does the amount that the weights can change.

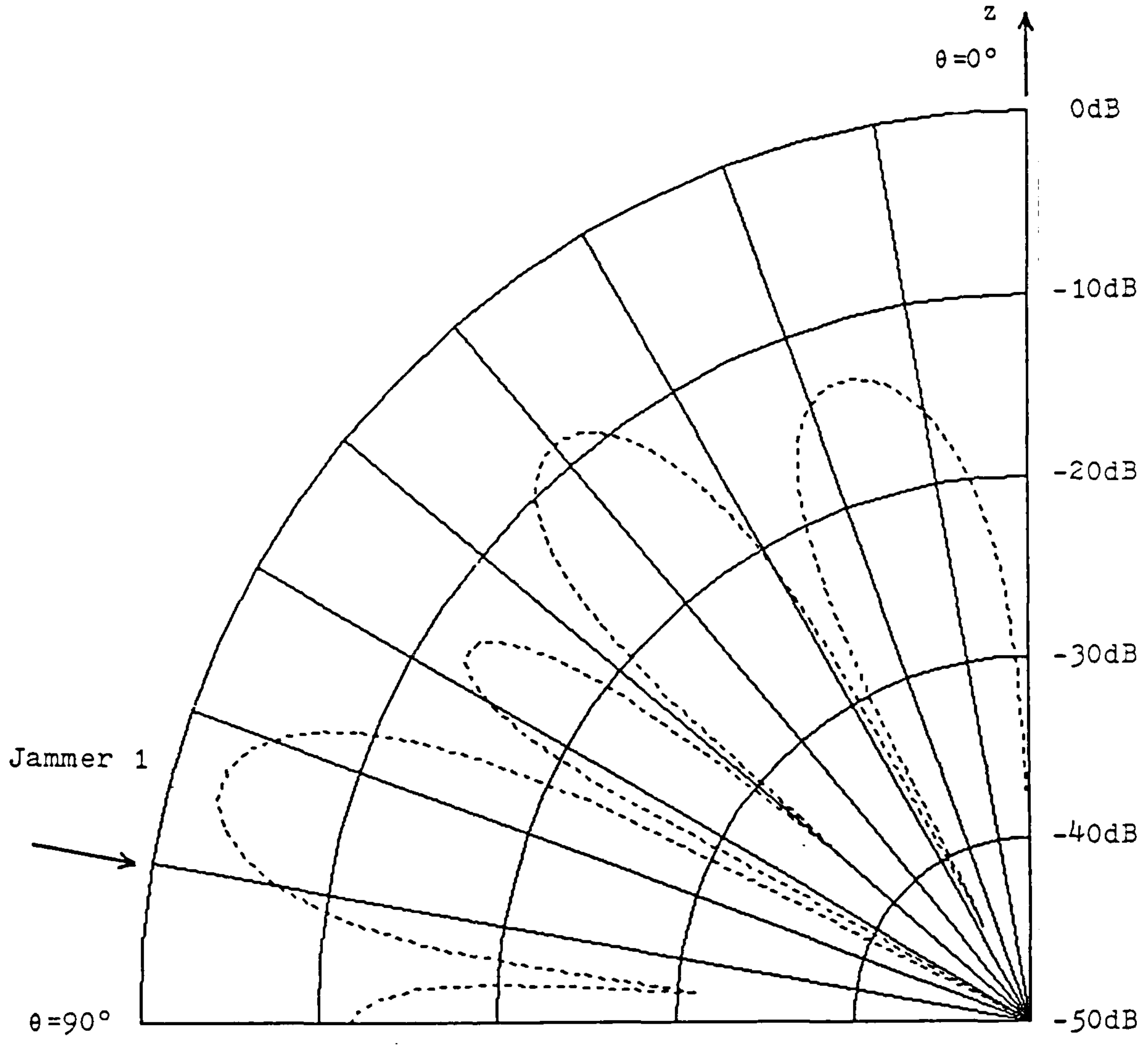
The weight changes of the Constrained LMS algorithm however, will never approach zero because the beam is constrained. Signal power will always appear at the output, even if contributions of interferences are negligible. Therefore, the weights will always change an appreciable amount, provided that the signal is present

$$W(k+1) = \text{Factor}*[W(k) - \mu y(k)\ X^{*}(k)]$$

It therefore seems reasonable that the Constrained LMS algorithm will perform better when the signal power is low or when it is absent altogether. This illustrates an important advantage possessed by the Constrained LMS algorithm. Unlike the others, the Constrained LMS algorithm could optimally adjust the weights before actual signal transmission (including preamble) begins. This is the major focus of TEST 4.

SUMMARY OF PLOTS FOR TEST1

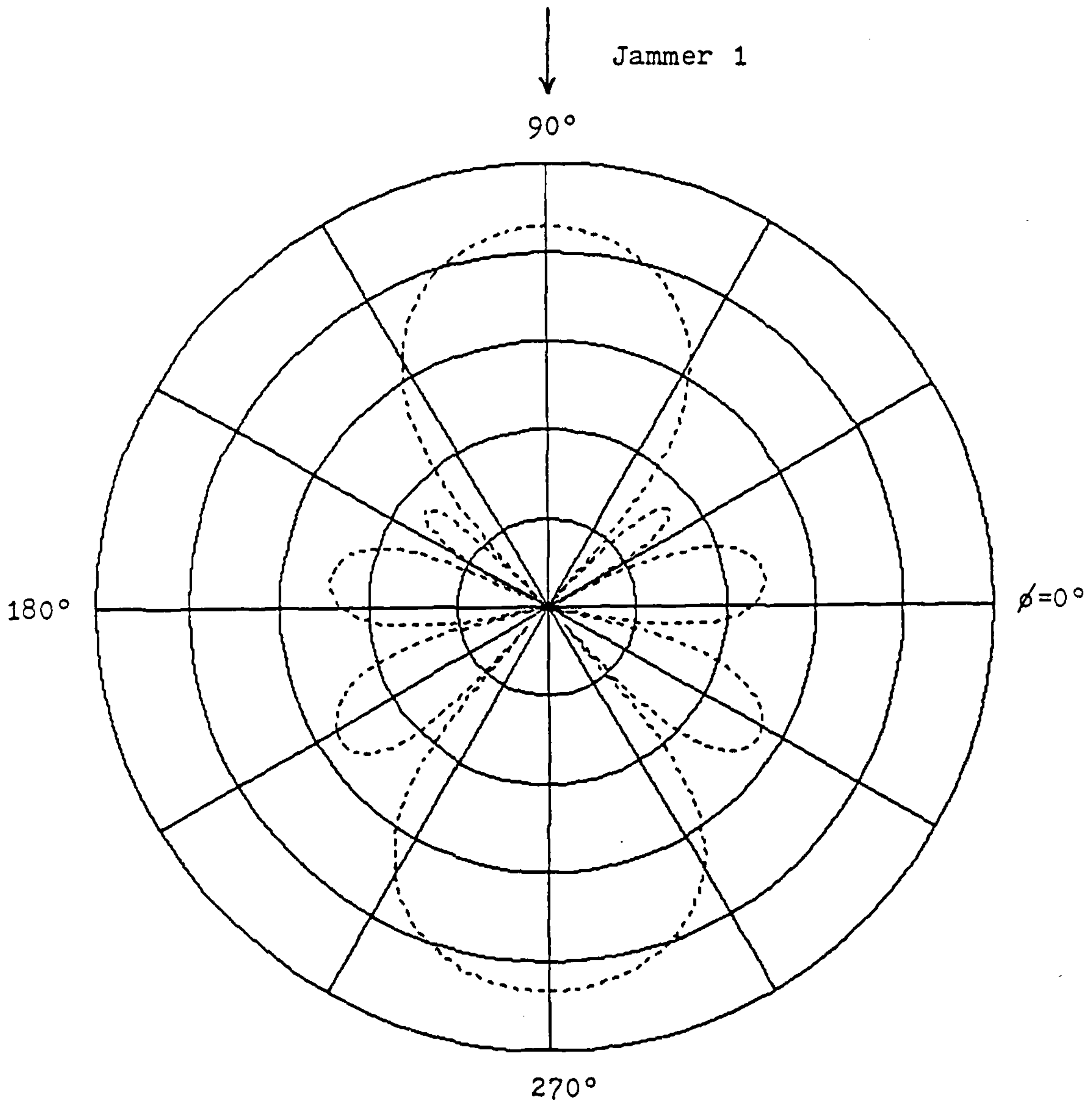
Figure T1.1:	Unadapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To indicate initial gain in direction of Jammer 1
Figure T1.2:	LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T1.3:	Constrained LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T1.4:	Update Covariance-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To demonstrate nulling of Jammer 1 by each algorithm
Figure T1.5:	Unadapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To indicate initial gain in direction of Jammer 2
Figure T1.6:	LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T1.7:	Constrained LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T1.8:	Update Covariance-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To demonstrate nulling of Jammer 2 by each algorithm



Elevation plot at 90° azimuth

Arrow indicates arrival angle of Jammer 1 (θ=80°)

Figure T1. Unadapted Antenna Pattern in Direction of Jammer 1 (Figure A)



Azimuthal plot at 80° elevation

Arrow indicates arrival angle of Jammer 1 (ϕ=90°)

Figure T1. Unadapted Antenna Pattern in Direction of Jammer 1

(Figure B)

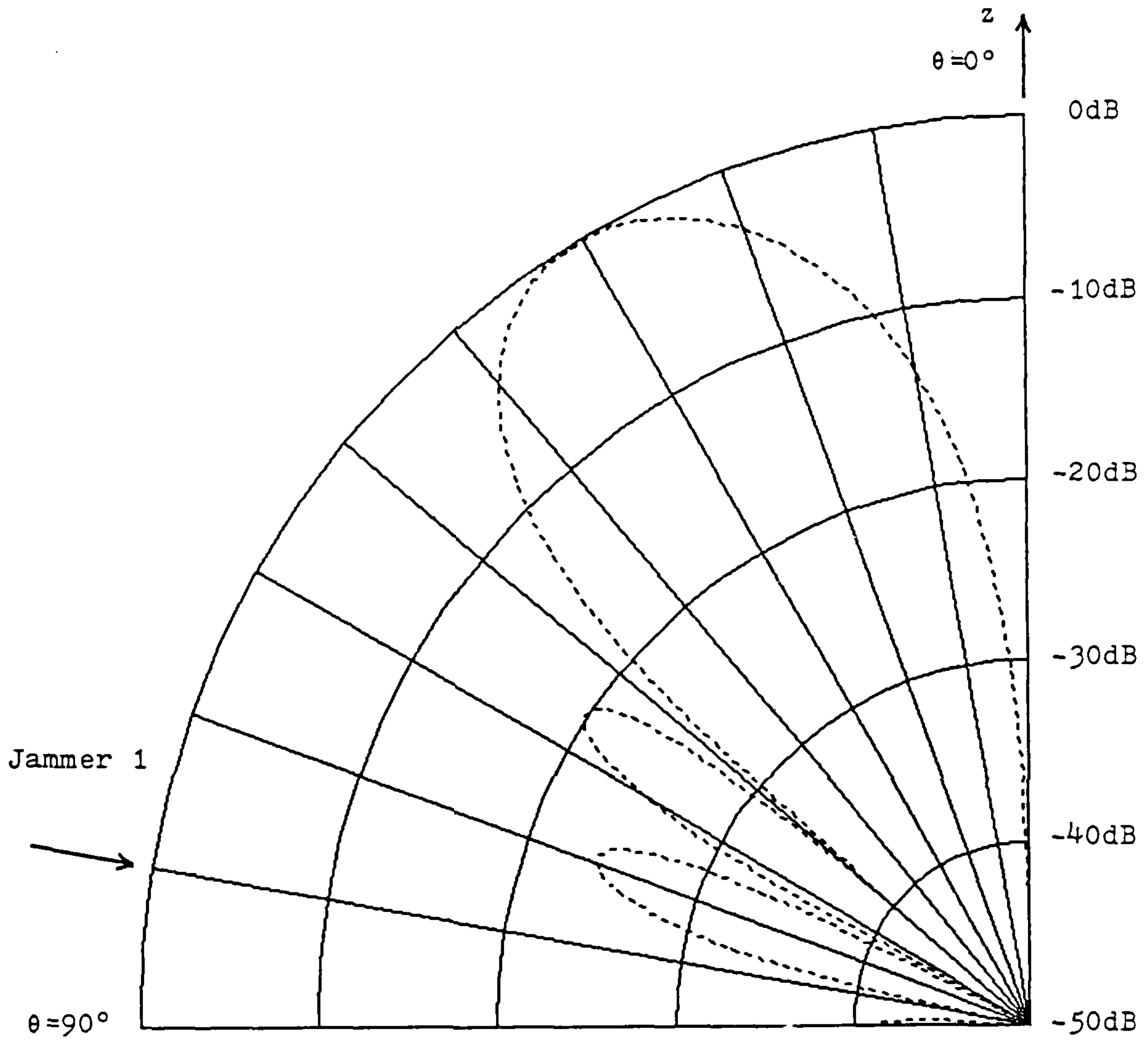


Figure T1.2 LMS-Adapted Pattern in Direction of Jammer 1

(Figure A)

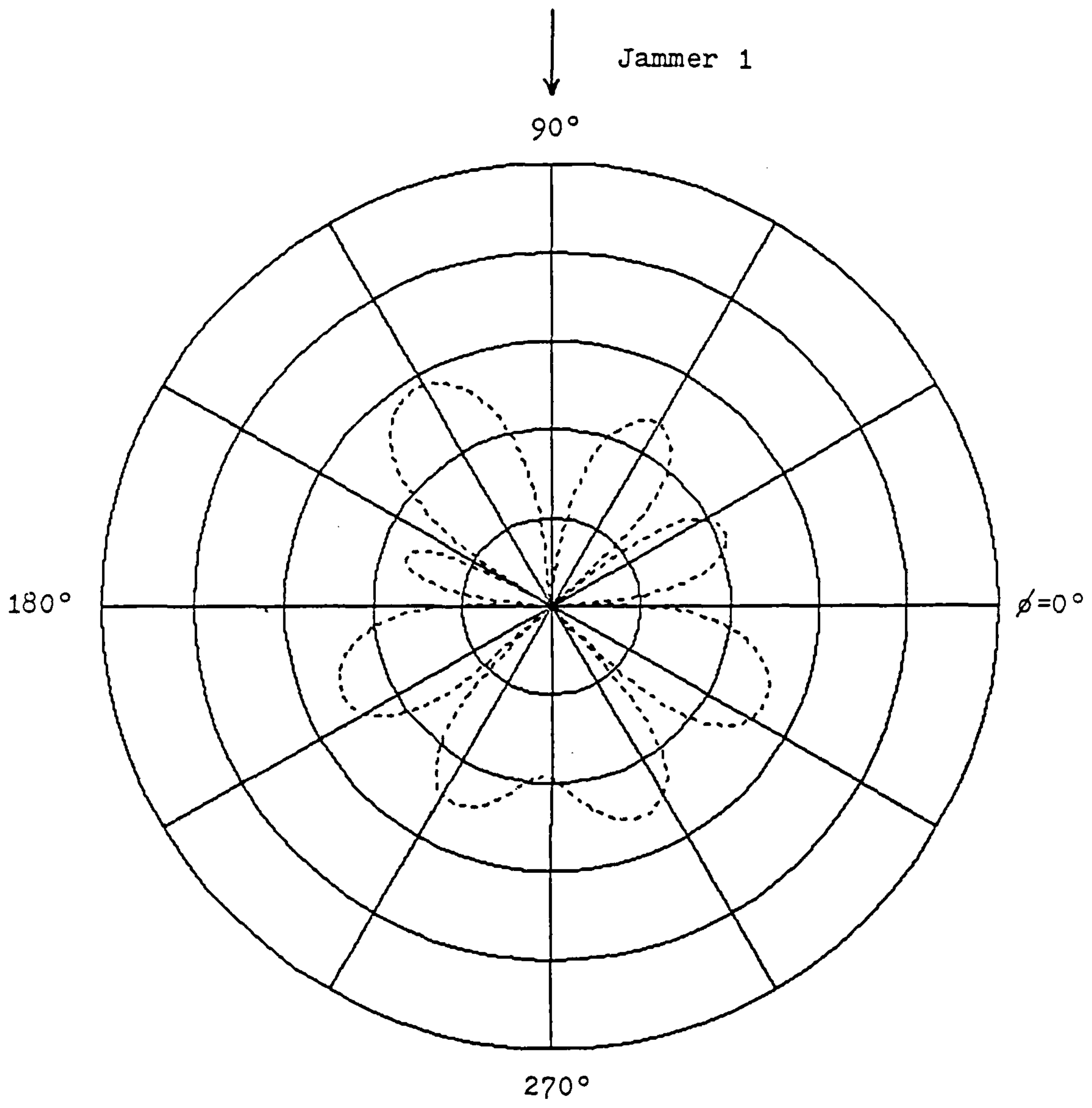


Figure T1.2 LMS-Adapted Pattern in Direction of Jammer 1

(Figure B)

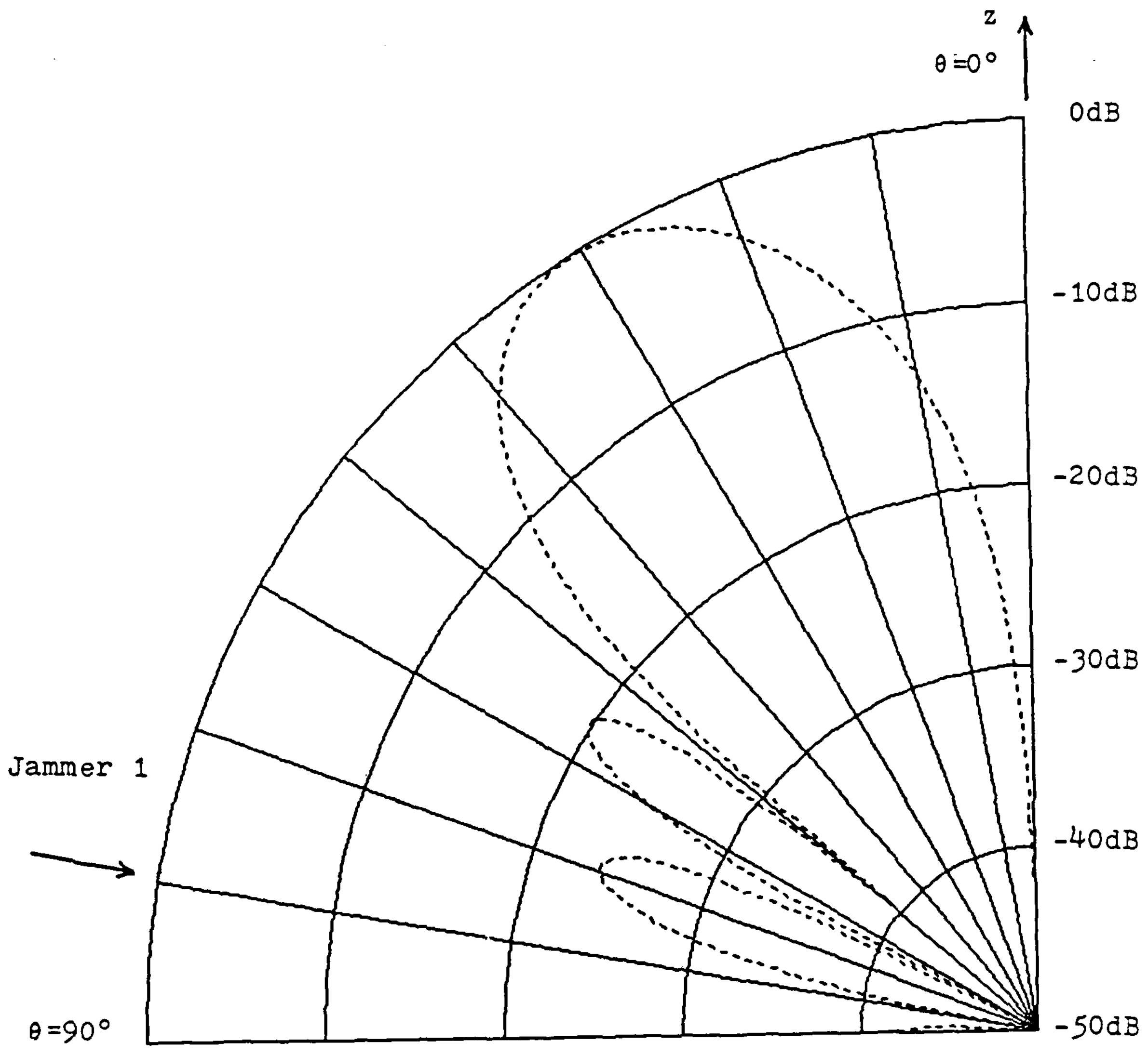


Figure T1.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

(Figure A)

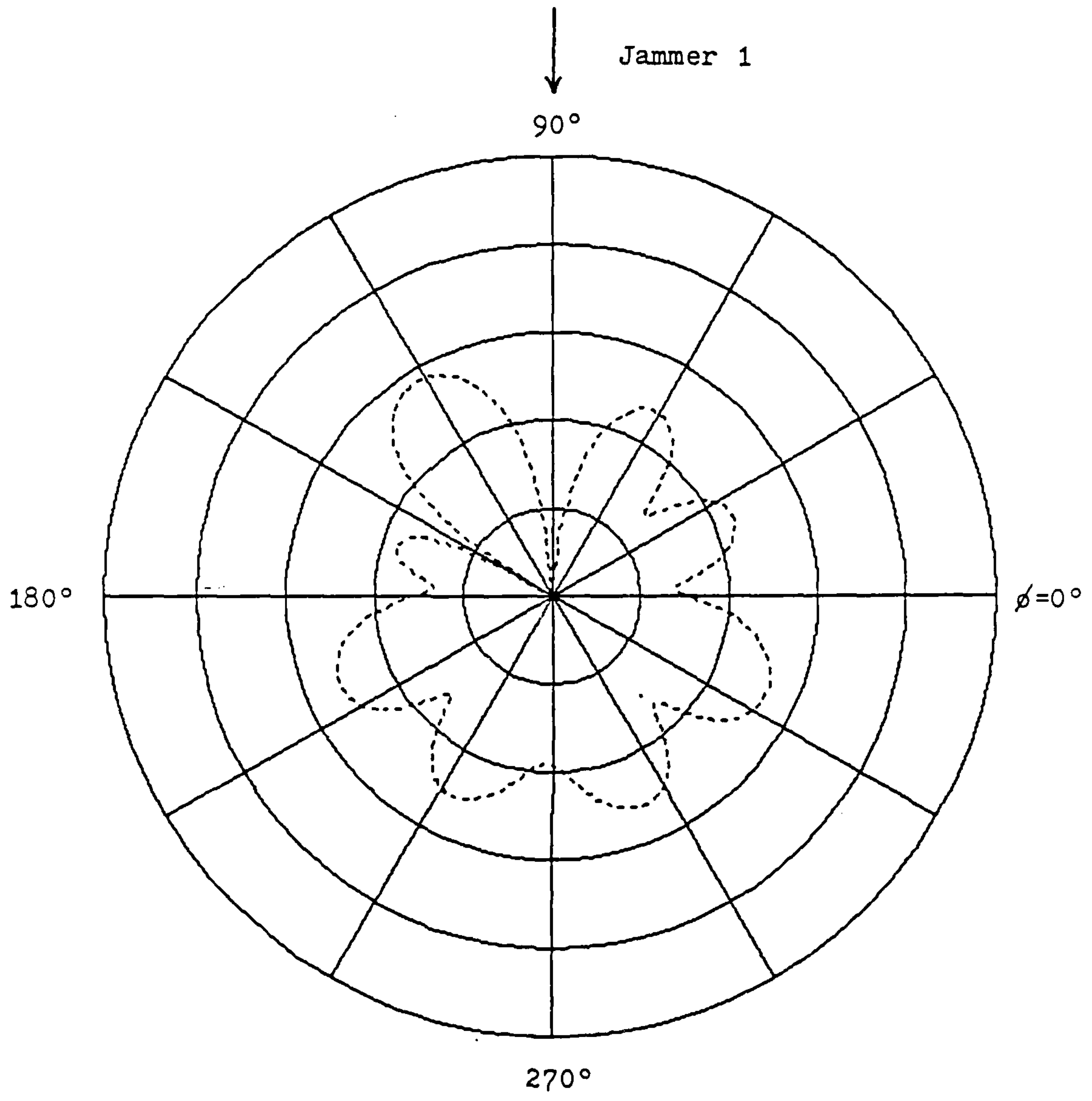


Figure T1.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

(Figure B)

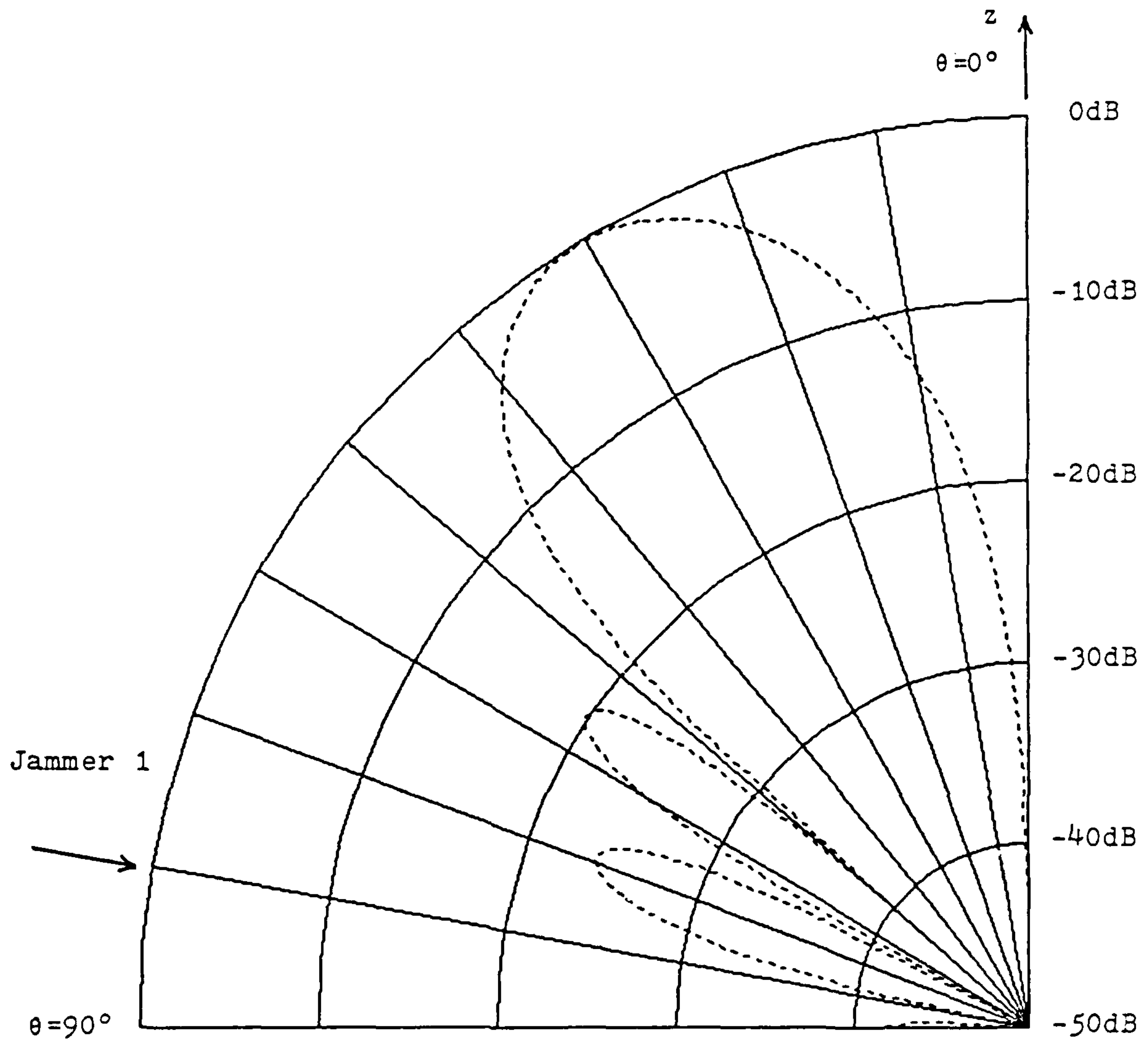


Figure T1.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure A)

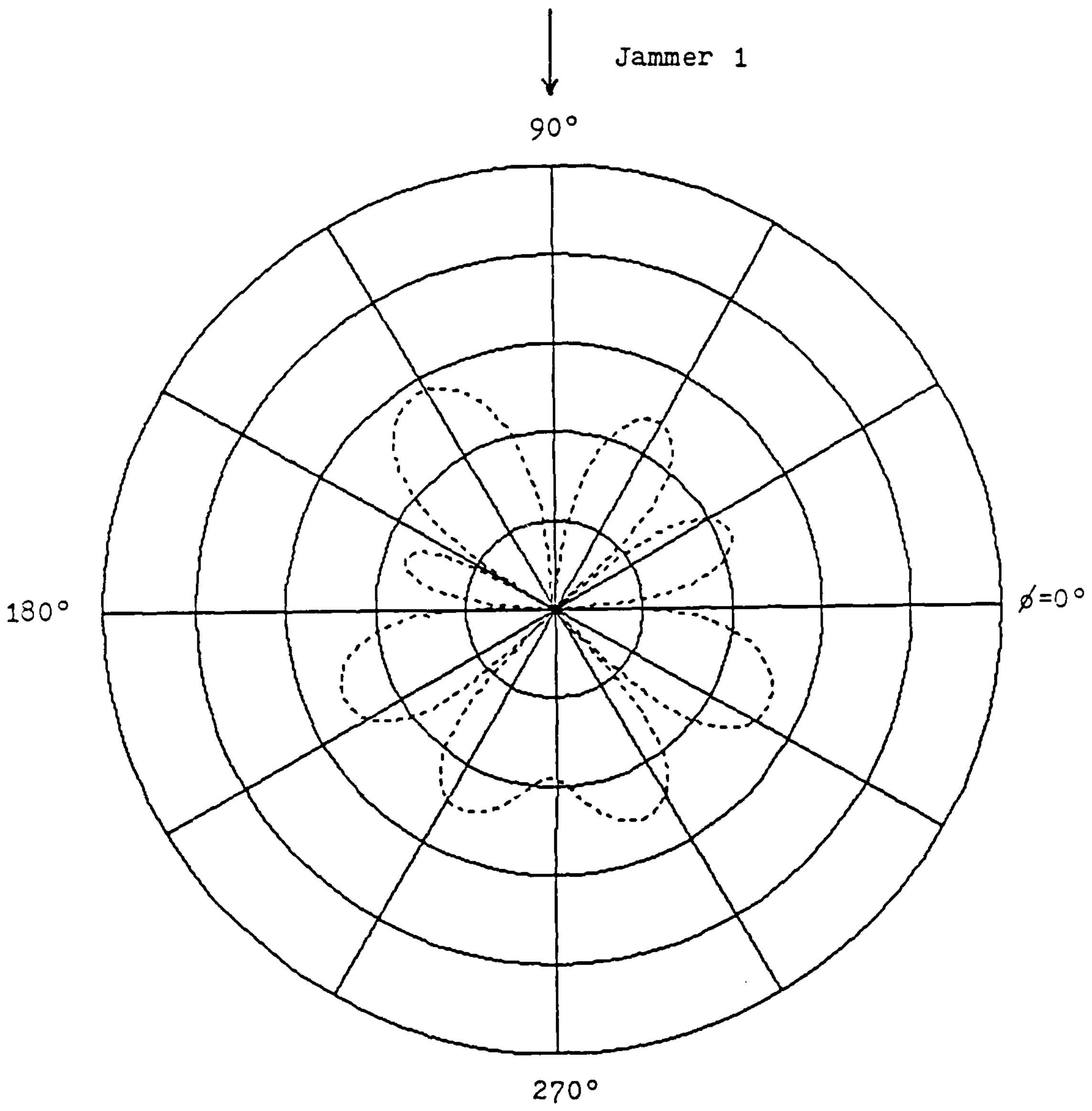
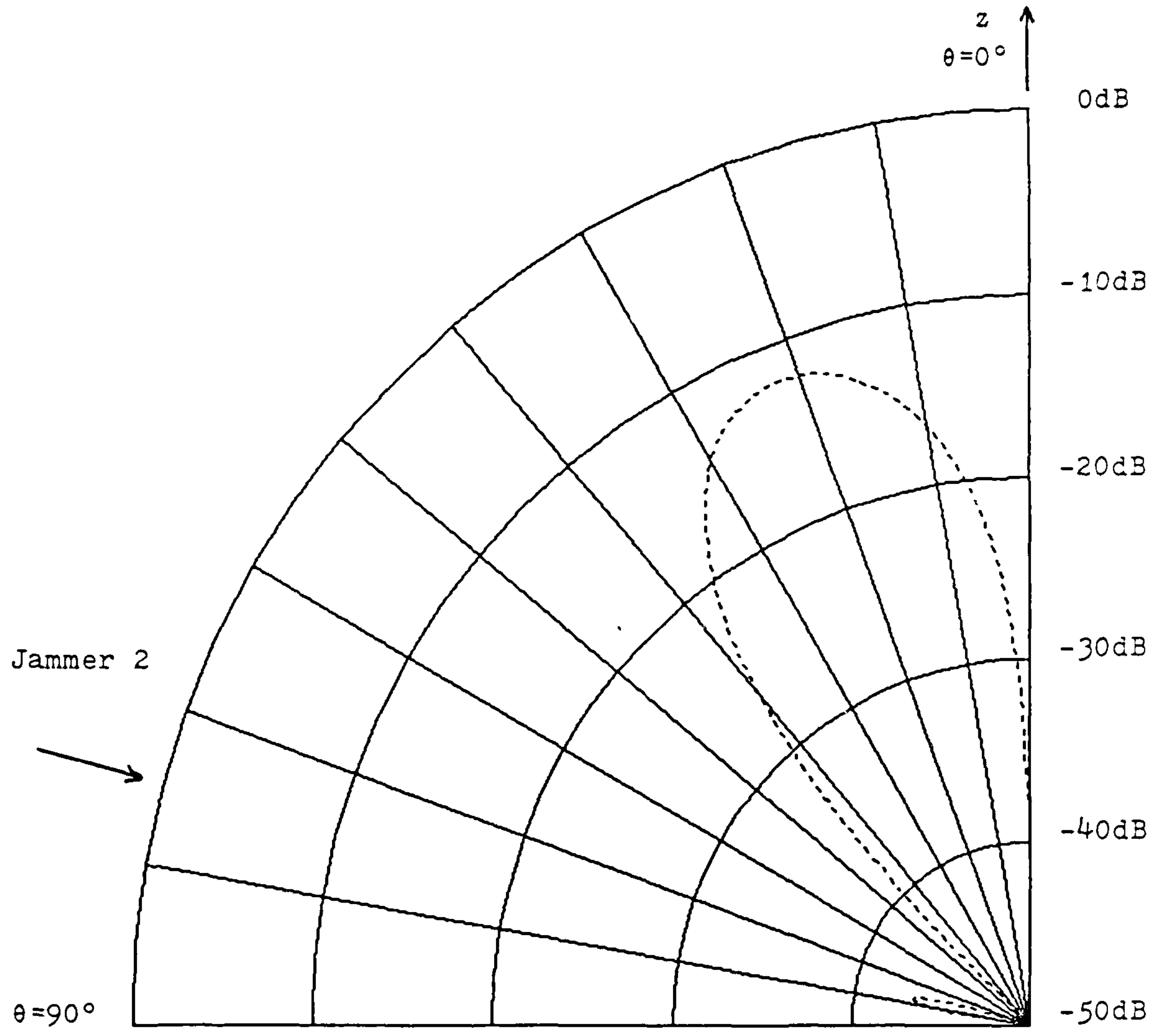


Figure T1.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure B)

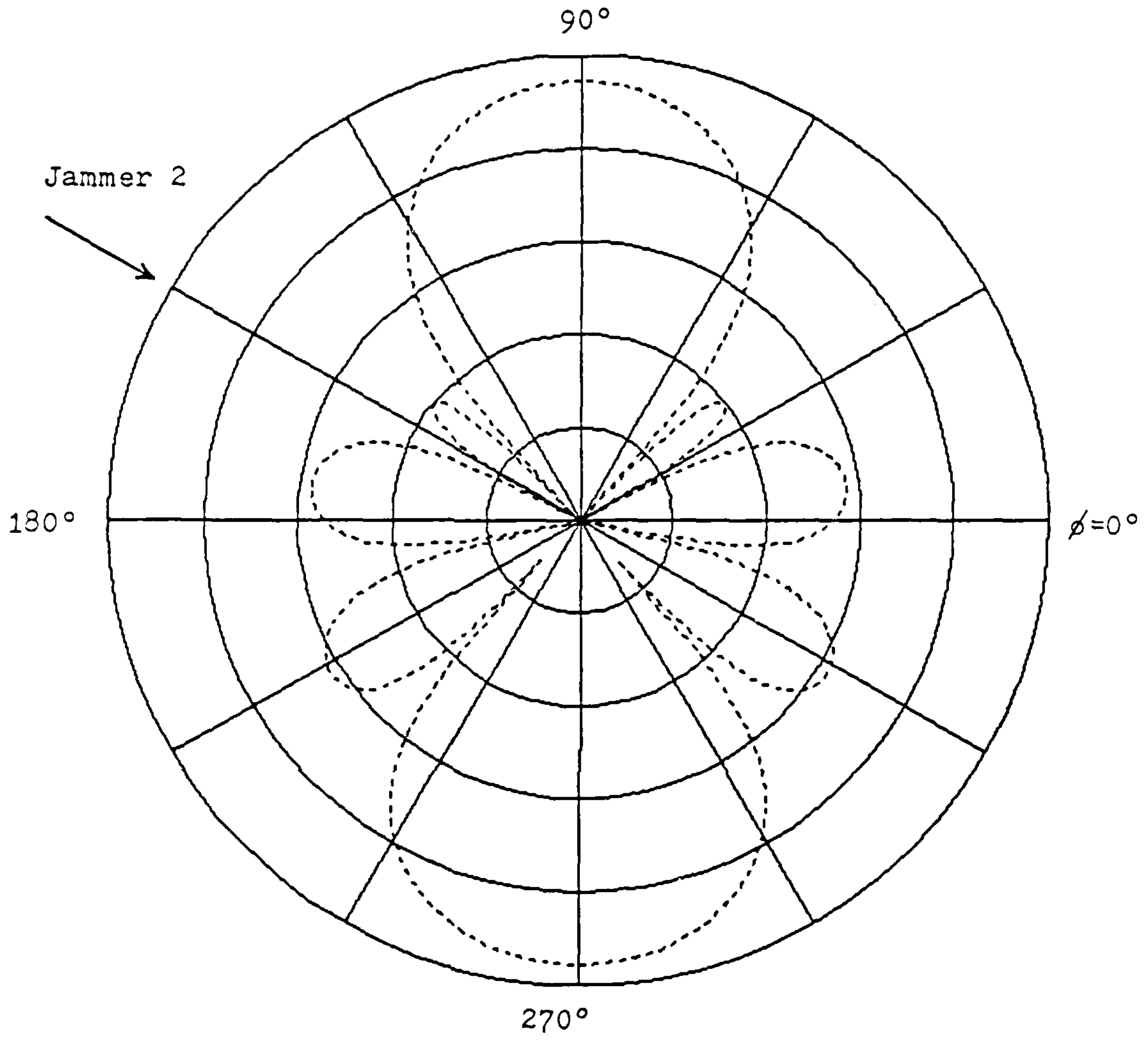


Elevation plot at 150° azimuth

Arrow indicates arrival angle of Jammer 2 (θ=75°)

Figure T1.5 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure A)



Azimuthal plot at 75° elevation

Arrow indicates arrival angle of Jammer 2 (ø=150°)

Figure T1.5 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure B)

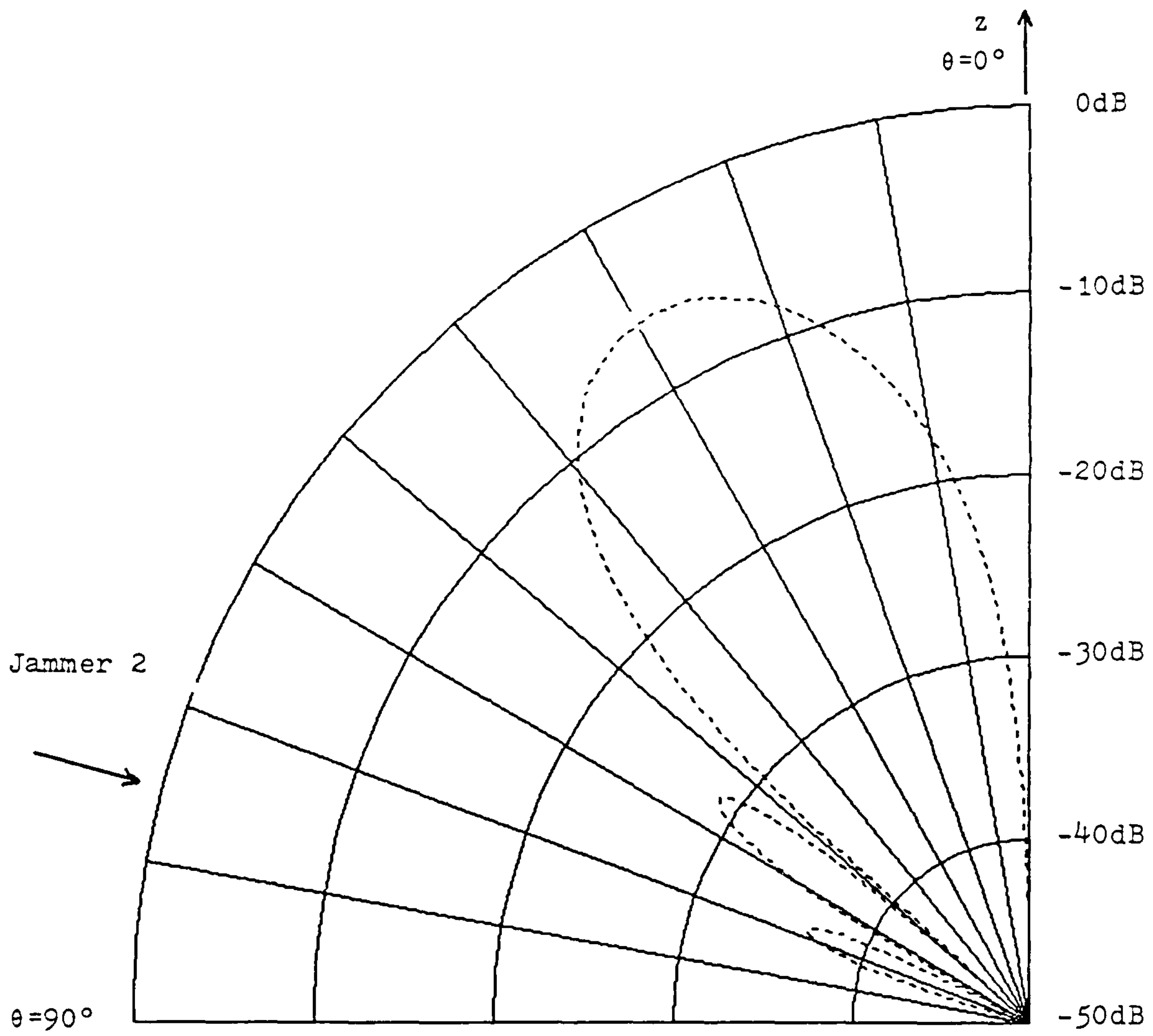


Figure T1.6 LMS-Adapted Pattern in Direction of Jammer 2 (Figure A)

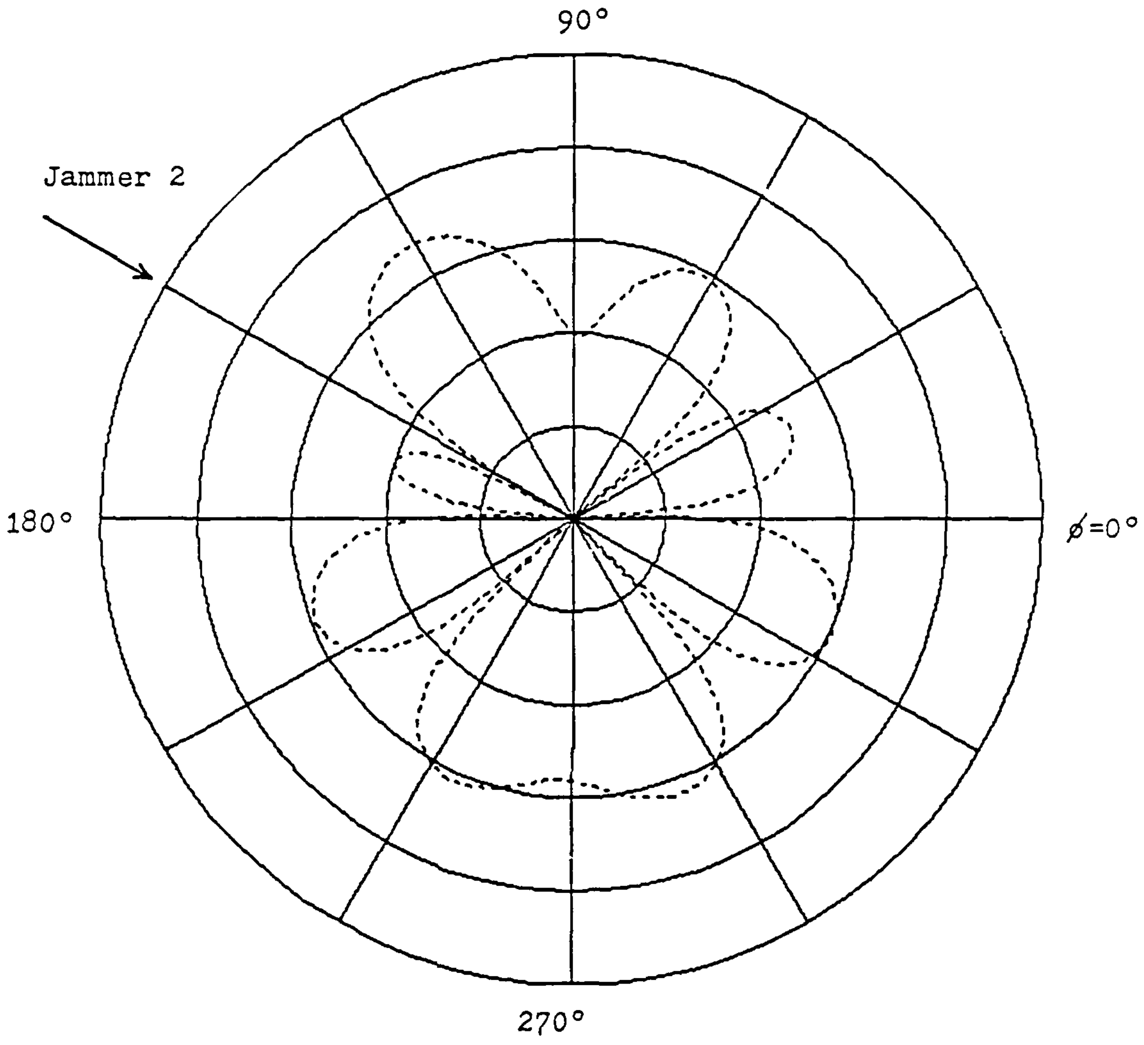


Figure T1.6 LMS-Adapted Pattern in Direction of Jammer 2

(Figure B)

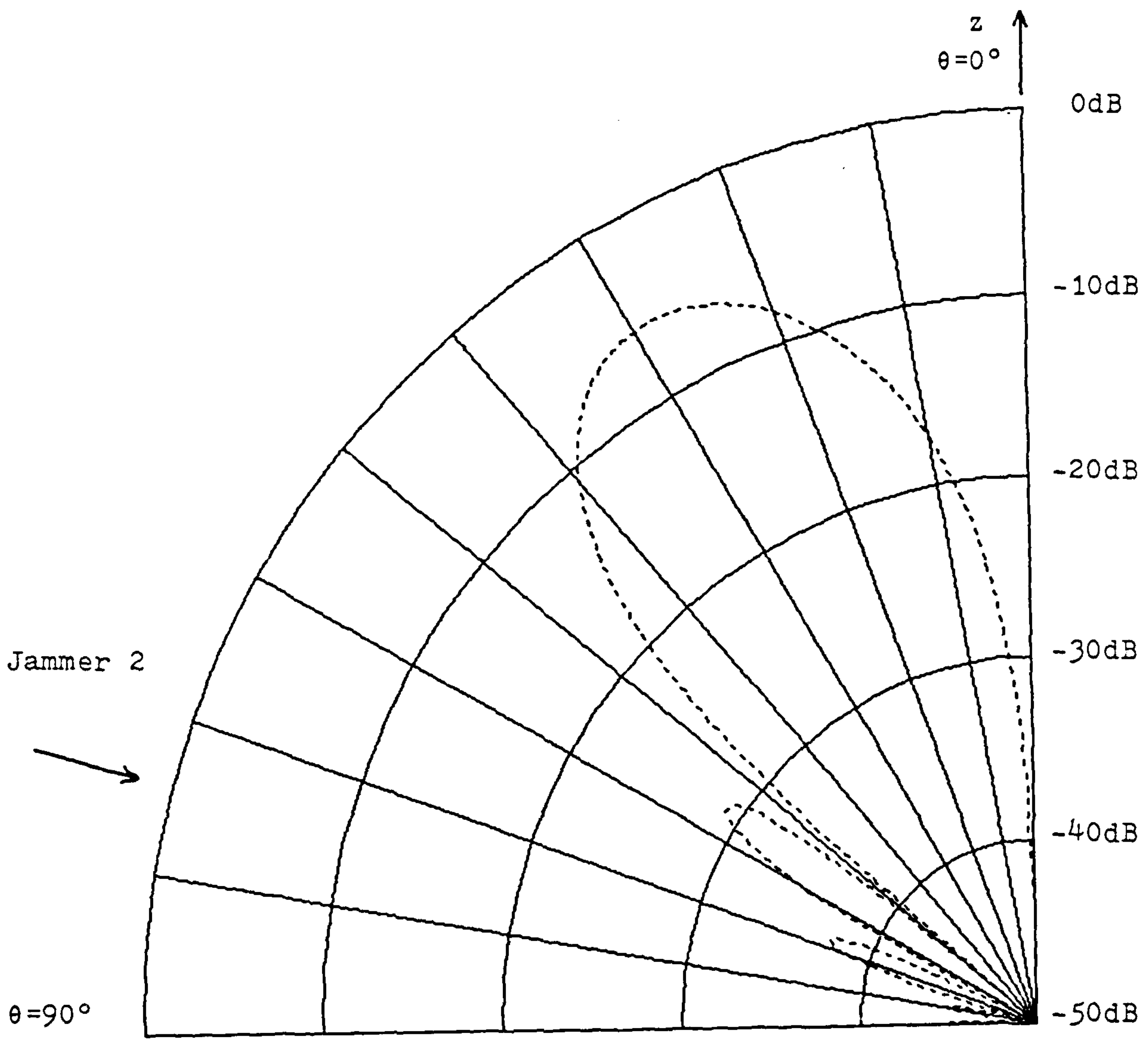


Figure T1.7 Constrained LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

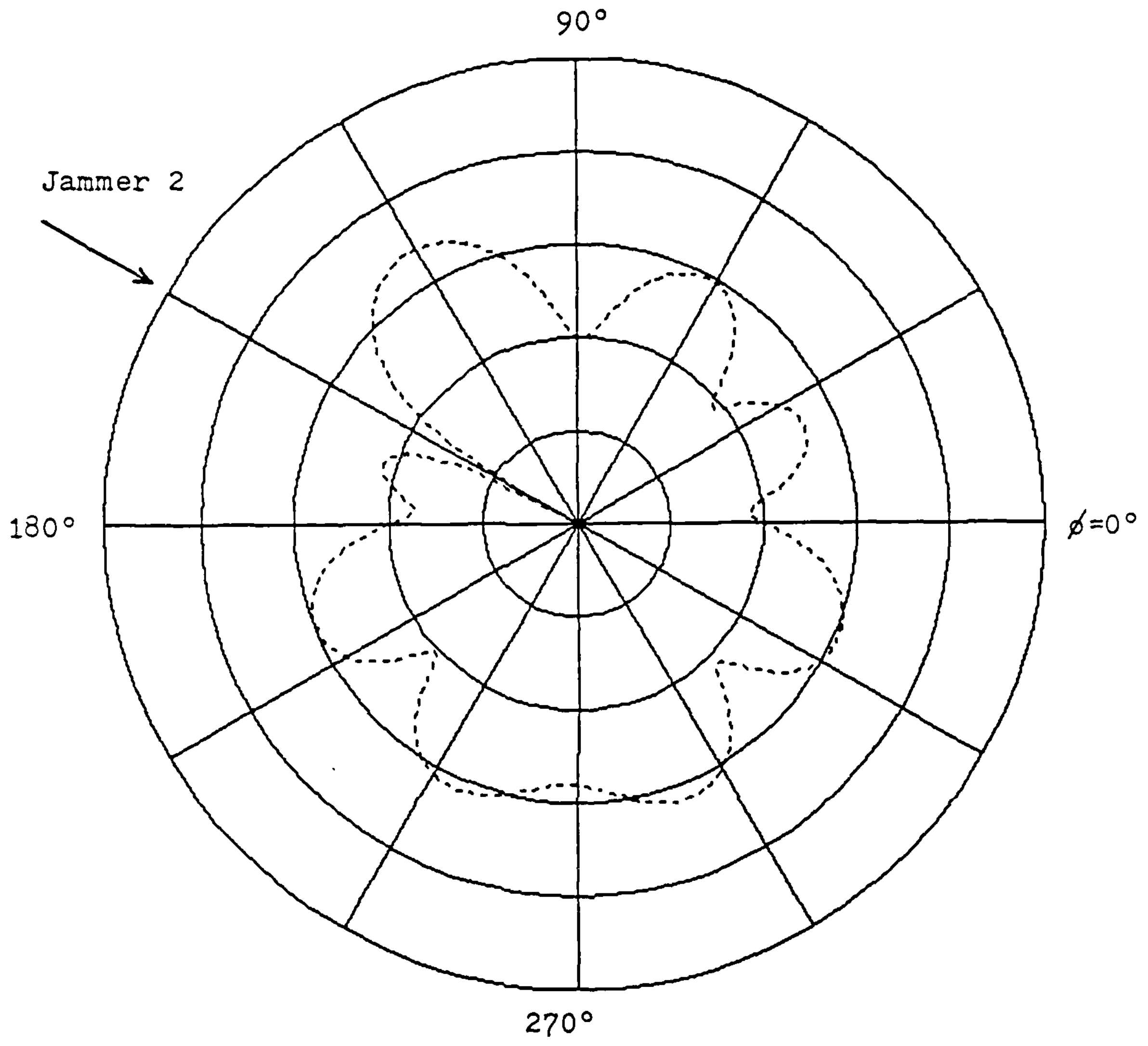


Figure T1.7 Constrained LMS-Adapted Pattern in Direction of Jammer 2

(Figure B)

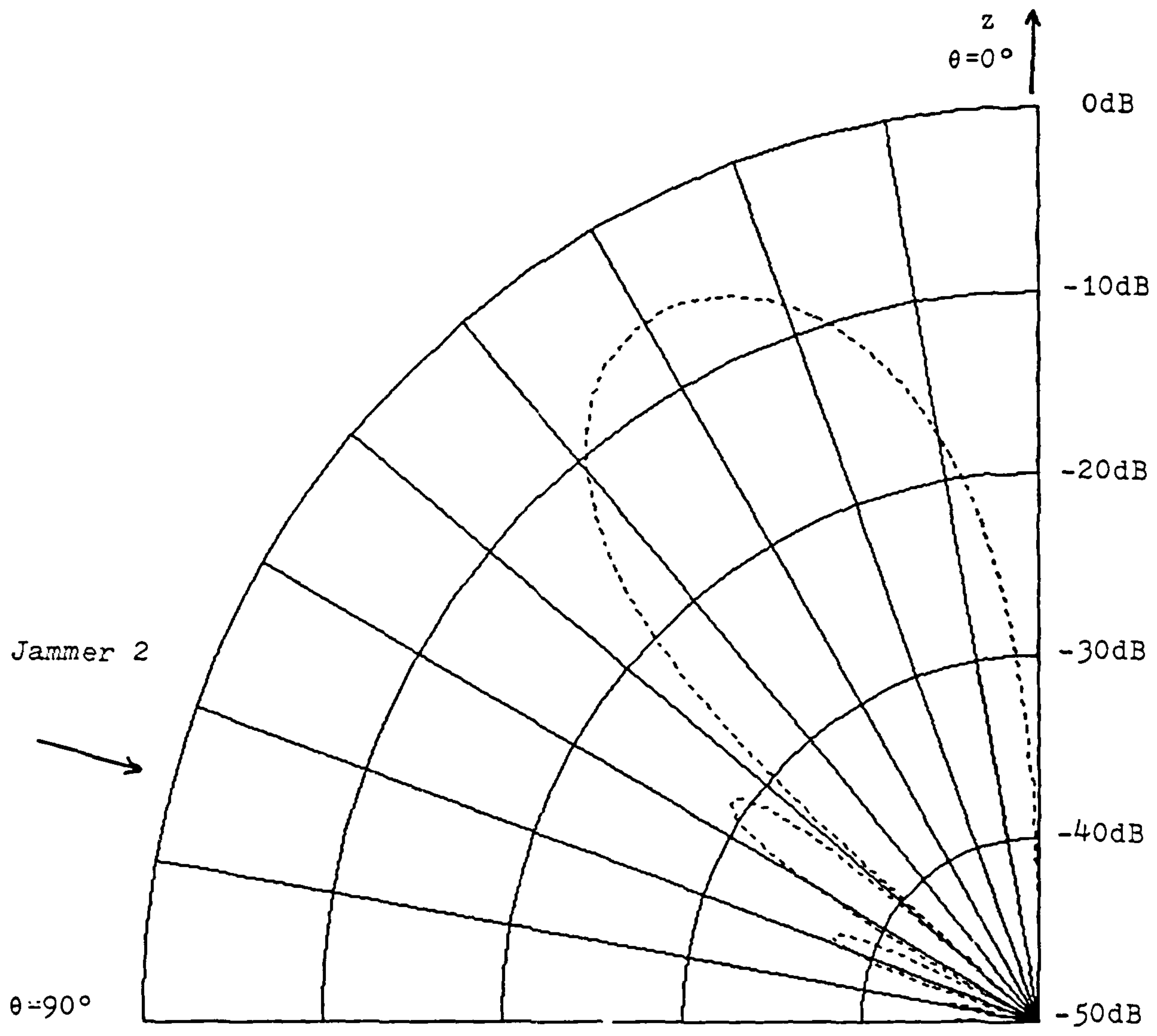


Figure T1.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure A)

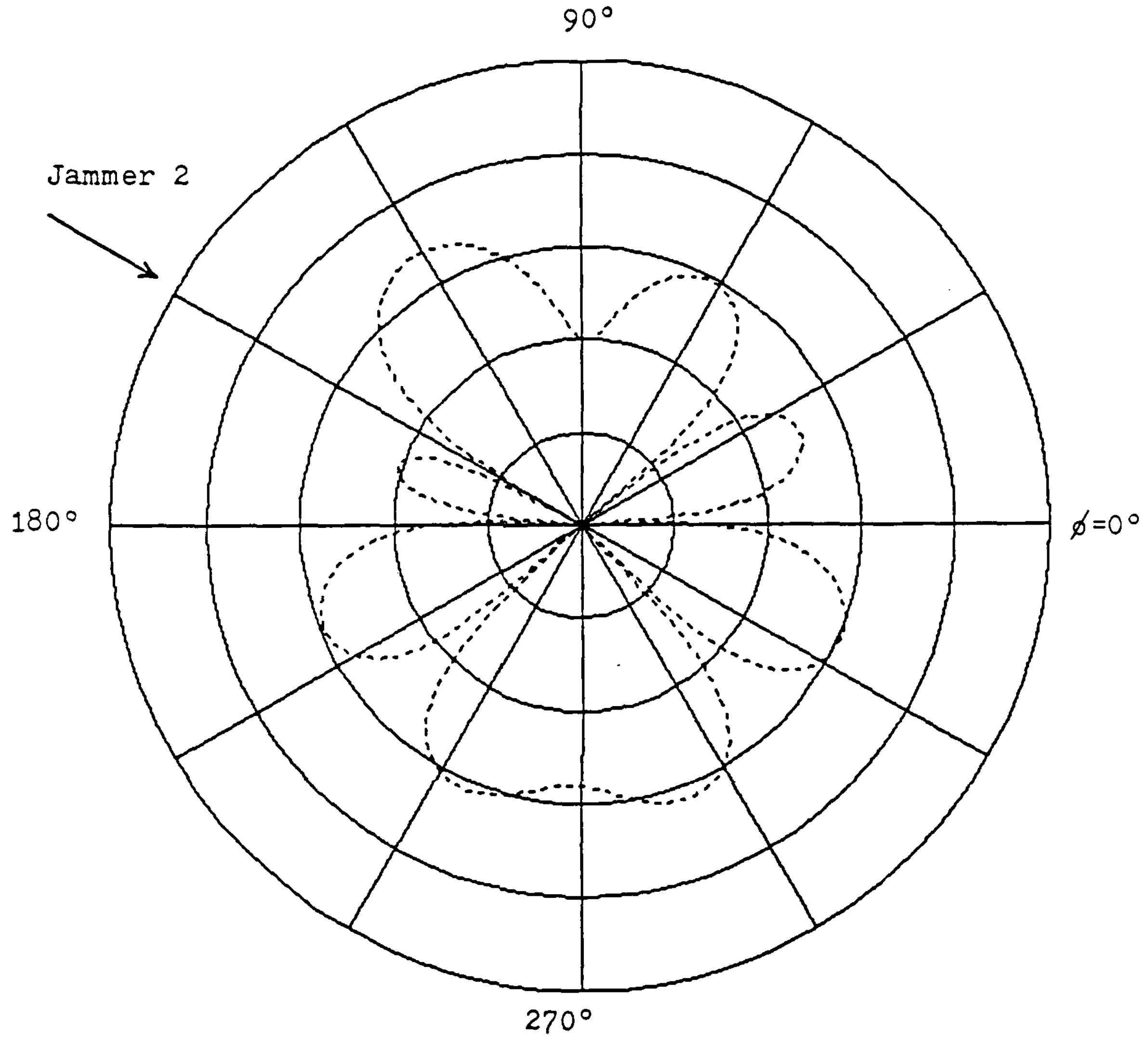


Figure T1.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure B)

7.2 Test 2: HF Channel With Moderate Delay Characteristics and Poor Attenuation Characteristics

The second test represents a significant departure from an ideal channel. The channel consists of three paths separated by 0.83333 msecs. Each of the paths is given equal weighting. In other words, signal components will not be attenuated more in one path than in another.

Once again, results for all algorithms were averaged over 100 convergences. Histograms depicting when the algorithms converged were generated and given as Figures 7.7 - 7.9. The results are tabulated in Table 7.2, and as the plots will verify, all algorithms did an effective job in nulling out the jammers.

The number of iterations required by the Update Covariance algorithm was very small. In this case, however, it is interesting to note the difference in actual computations of this algorithm and the LMS algorithm (9,846 for LMS, 10,651 for Update Covariance). If the computations all required the same amount of time to complete, the actual "convergence time" for the LMS would be shorter. This is why it is important to take the computational complexity of each iteration for the algorithms into account.

Once again, the Constrained LMS algorithm was the slowest in average convergence. The results also indicate that the algorithm had a wide fluctuation in the required number of iterations. As can be seen from Figure 7.7, the Constrained LMS algorithm had difficulty in attaining the "1 dB" threshold once the SNR had approached the optimal value. This problem can quite possibly be attributed to the weight update problem previously discussed. Even after contributions of interferences were reduced from the array output, the weights will change an appreciable amount due to the signal power (which now may be magnified by the existence of three paths). Once again, this will be studied further in TEST4.

As expected, the LMS algorithm also had more difficulty approaching an optimal value in this test than in the previous one. It should also be noted that in the event that the preamble was severely distorted by a channel, covergence of any kind would be severely hampered. In other words, if the received signal did not look much at all like the reference signal, the

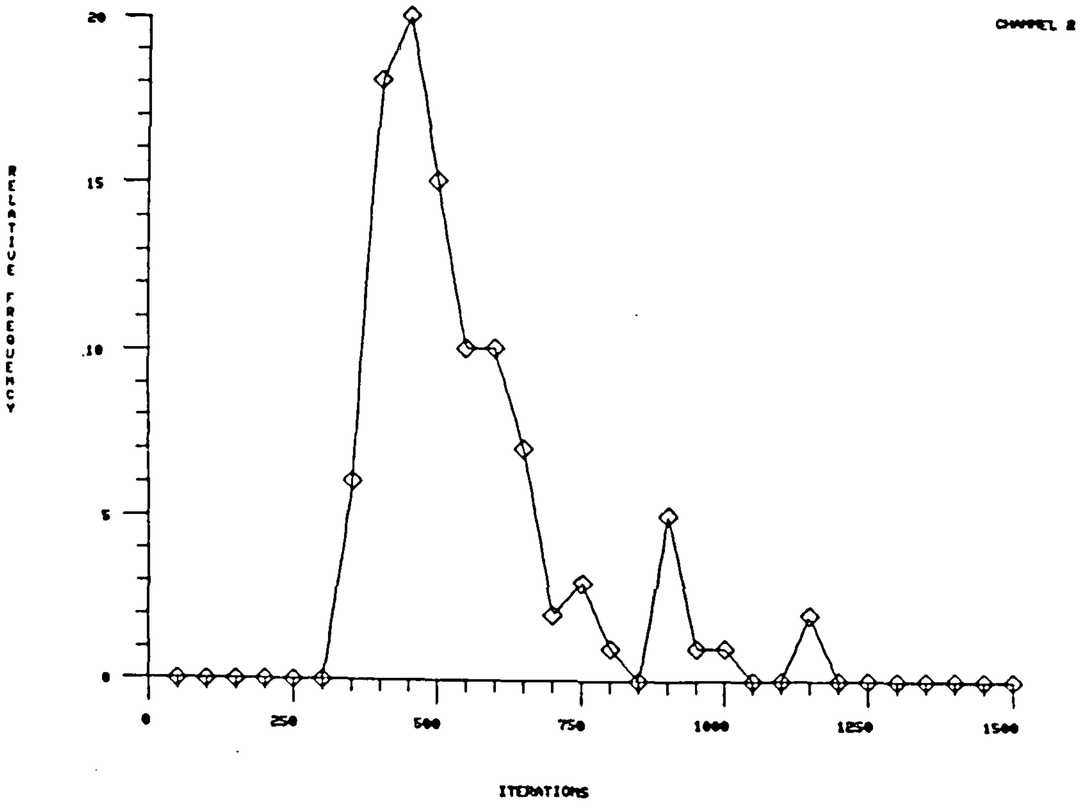


Figure 7.7 Convergence Histogram of LMS Algorithm in Channel 2

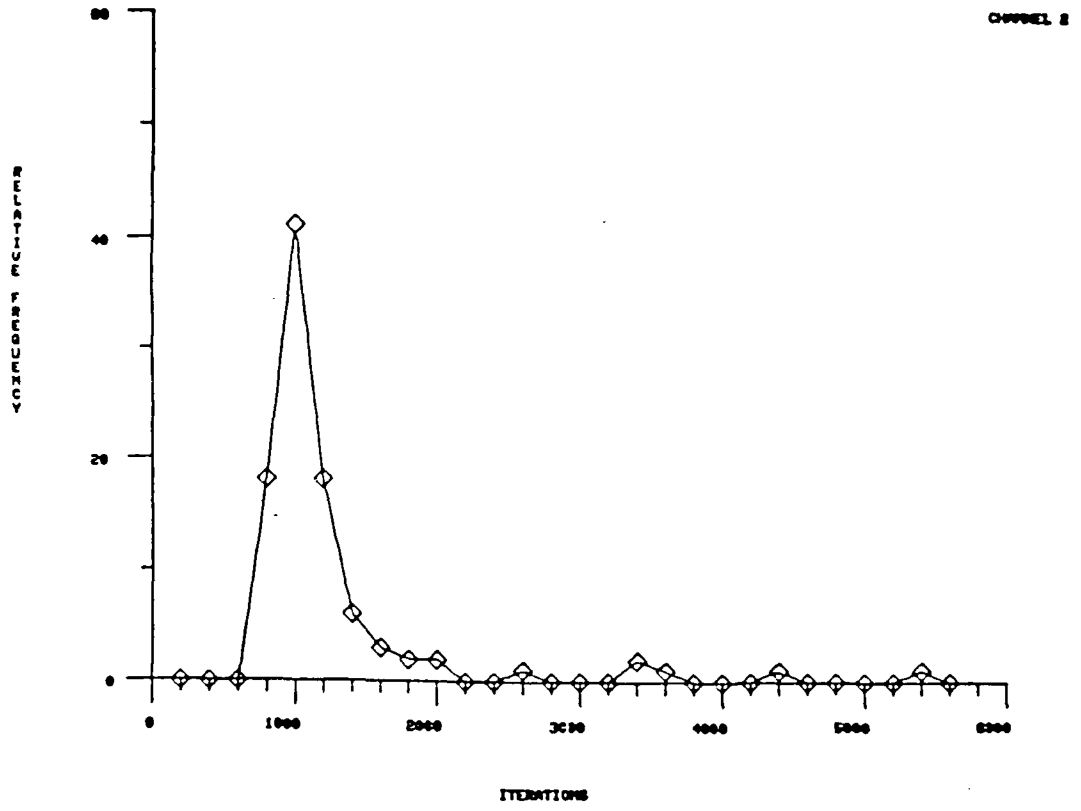


Figure 7.8 Convergence Histogram of Constrained LMS Algorithm in Channel 2

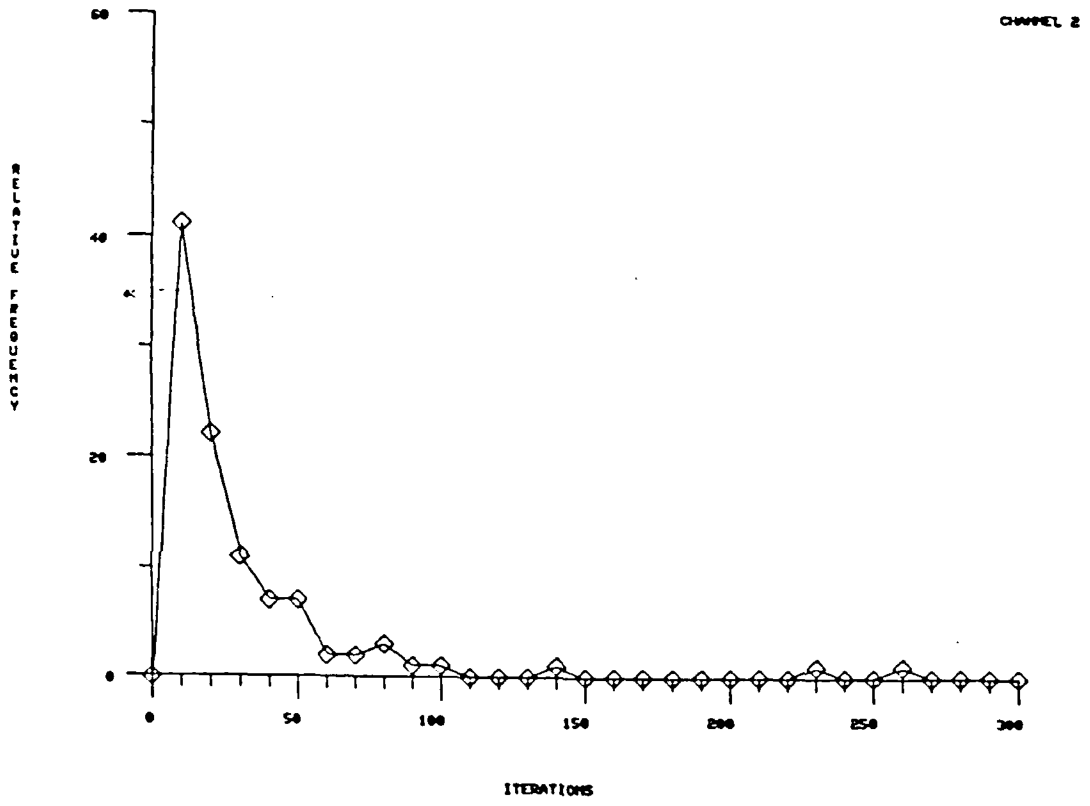


Figure 7.9 Convergence Histogram of Update Covariance Algorithm in Channel 2

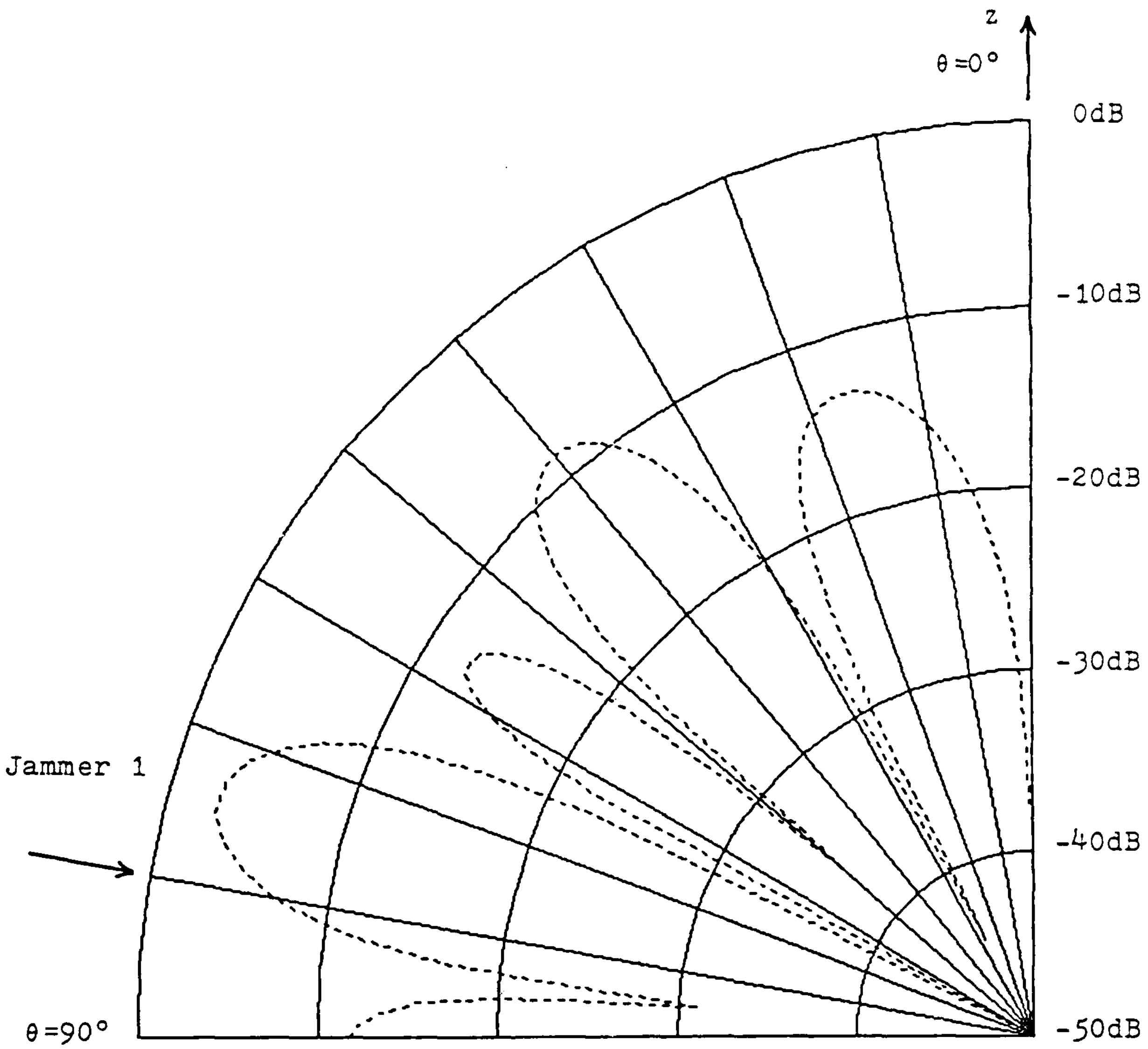
TABLE 7.2 TEST2 SUMMARY

ADAPTIVE ALGORITHM	NUMBER OF CONVERGENCES	AVERAGE NUMBER OF ITERATIONS	STANDARD DEVIATION
LMS	100	547.0	170.8
CONSTR. LMS	100	1155.0	699.2
UPDATE COV.	100	26.3	38.3

algorithm would have an extremely difficult task. After all, algorithms requiring references operate by forcing the array output to be equal to the reference signal. If this is attained, it is assumed that contributions of jammers are negligible. Consider the case, however, when the received signal does not strongly resemble the reference signal. Even if the weights were optimally adjusted, the output of the array would not look like the reference signal, and the algorithm would continue to adjust the weights in an attempt to force the output to be equal to the reference signal. In effect, it would be trying to compensate for the channel effects, although it has no true means of doing so. This is an inherent problem with algorithms that require references and is worth mentioning. It is also worth mentioning that the LMS algorithm had no problem converging in this case, and the channel produced some fairly bad smearing effects. Therefore, it can be assumed that the channel must get significantly worse than this to prevent convergence.

SUMMARY OF PLOTS FOR TEST2

Figure T2.1:	Unadapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To indicate initial gain in direction of Jammer 1
Figure T2.2:	LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T2.3:	Constrained LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T2.4:	Update Covariance-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To demonstrate nulling of Jammer 1 by each algorithm
Figure T2.5:	Unadapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To indicate initial gain in direction of Jammer 2
Figure T2.6:	LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T2.7:	Constrained LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T2.8:	Update Covariance-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To demonstrate nulling of Jammer 2 by each algorithm

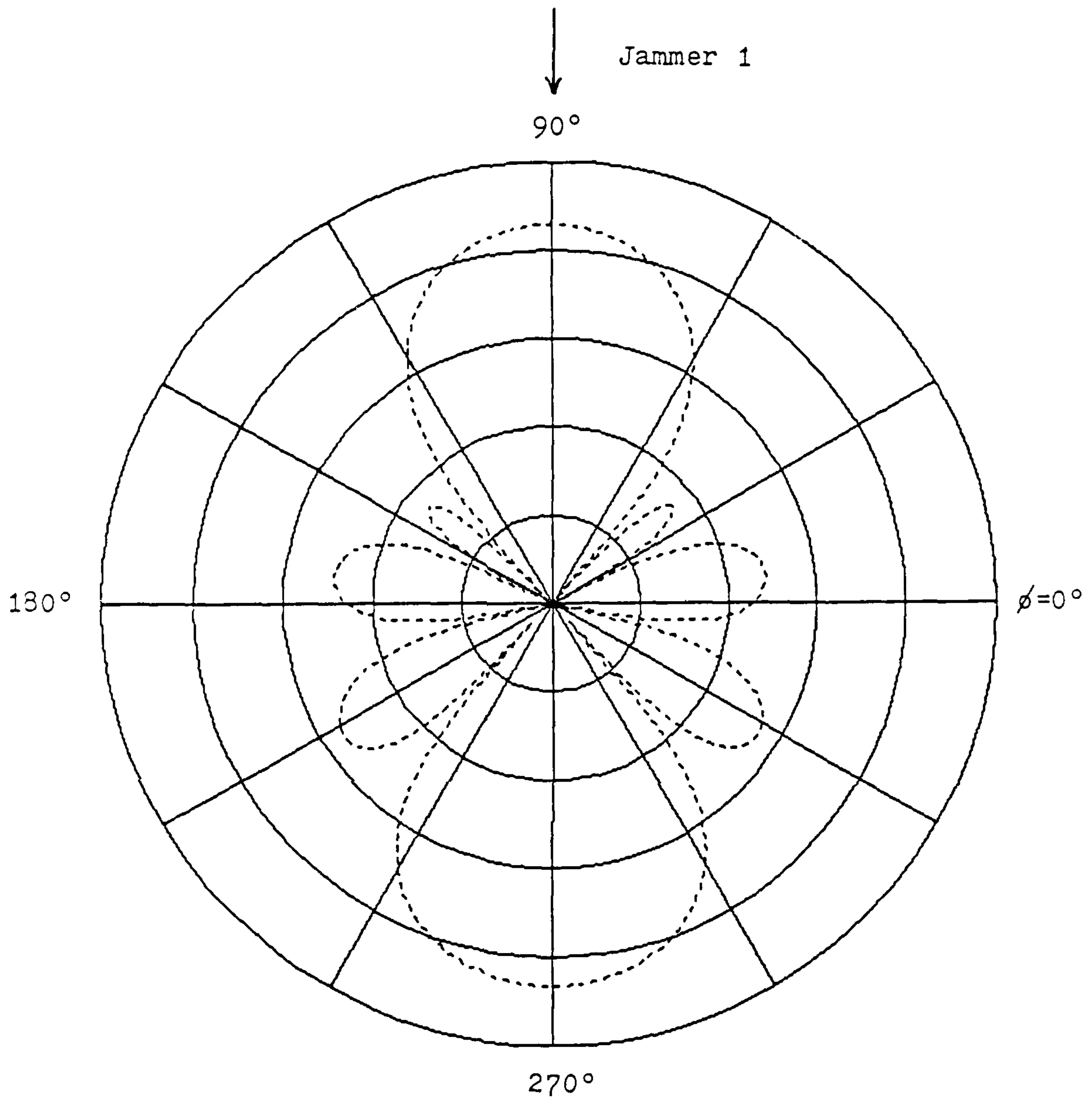


Elevation plot at 90° azimuth

Arrow indicates arrival angle of Jammer 1 (θ=80°)

Figure T2.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure A)



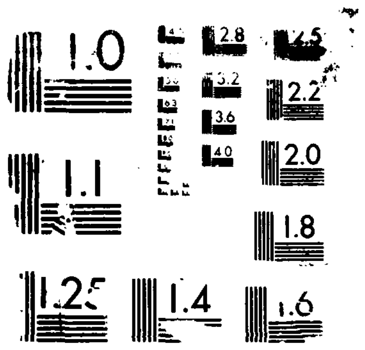
Azimuthal plot at 80° elevation

Arrow indicates arrival angle of Jammer 1 (ø=90°)

Figure T2.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure B)

AD-A189 329   ADAPTIVE ALGORITHMS FOR HF ANTENNA ARRAYS(U) KANSAS   2/3
UNIV CENTER FOR RESEARCH INC LAWRENCE TELECOMMUNICATIO
D HAEGERT ET AL JUL 87 TISL-5481 RADC-TR-86-159
UNCLASSIFIED F30602-81-C-0205   F/G 9/1   NL



MICROCOPY RESOLUTION TEST CHART

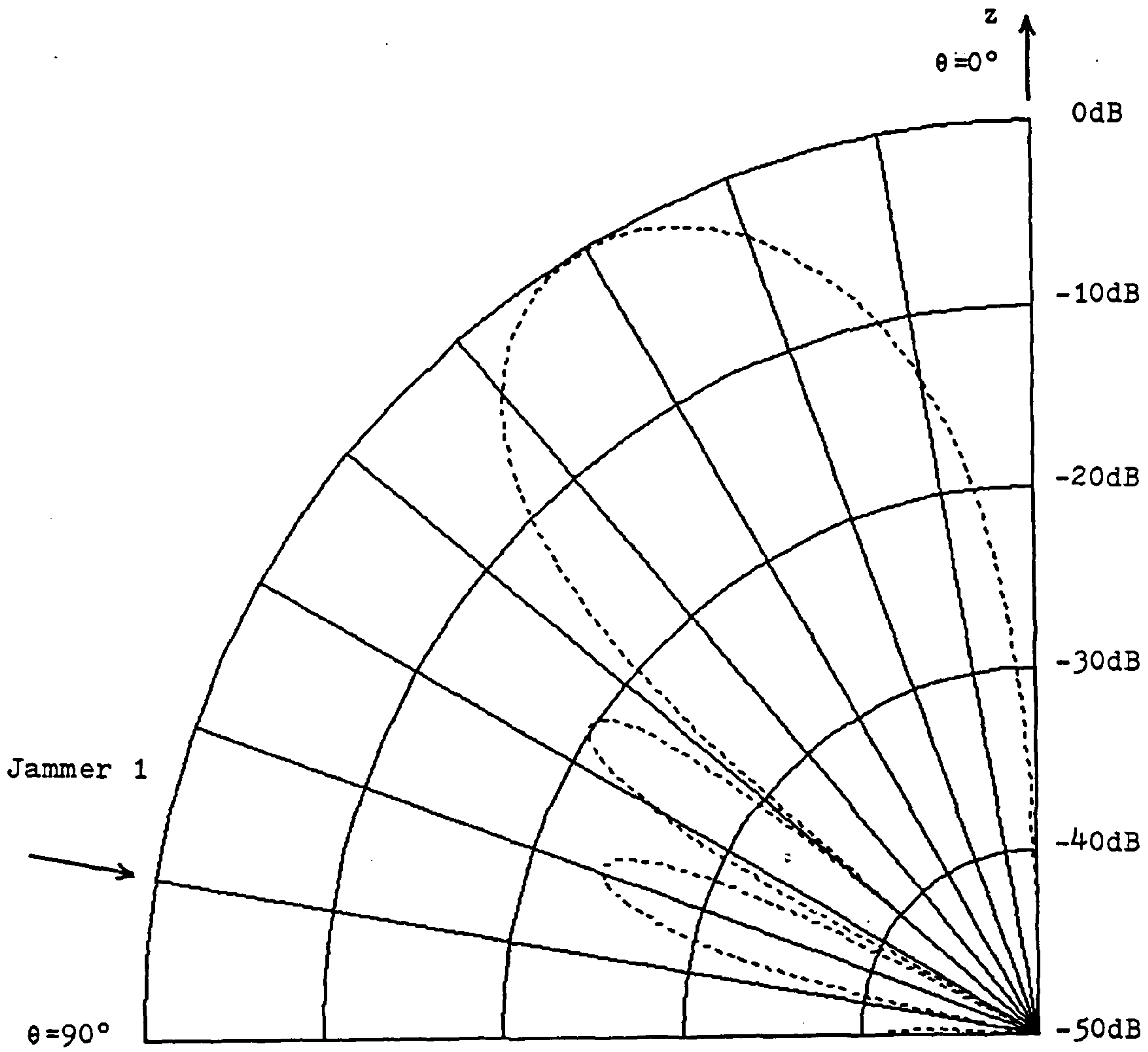


Figure T2.2 LMS-Adapted Pattern in Direction of Jammer 1

(Figure A)

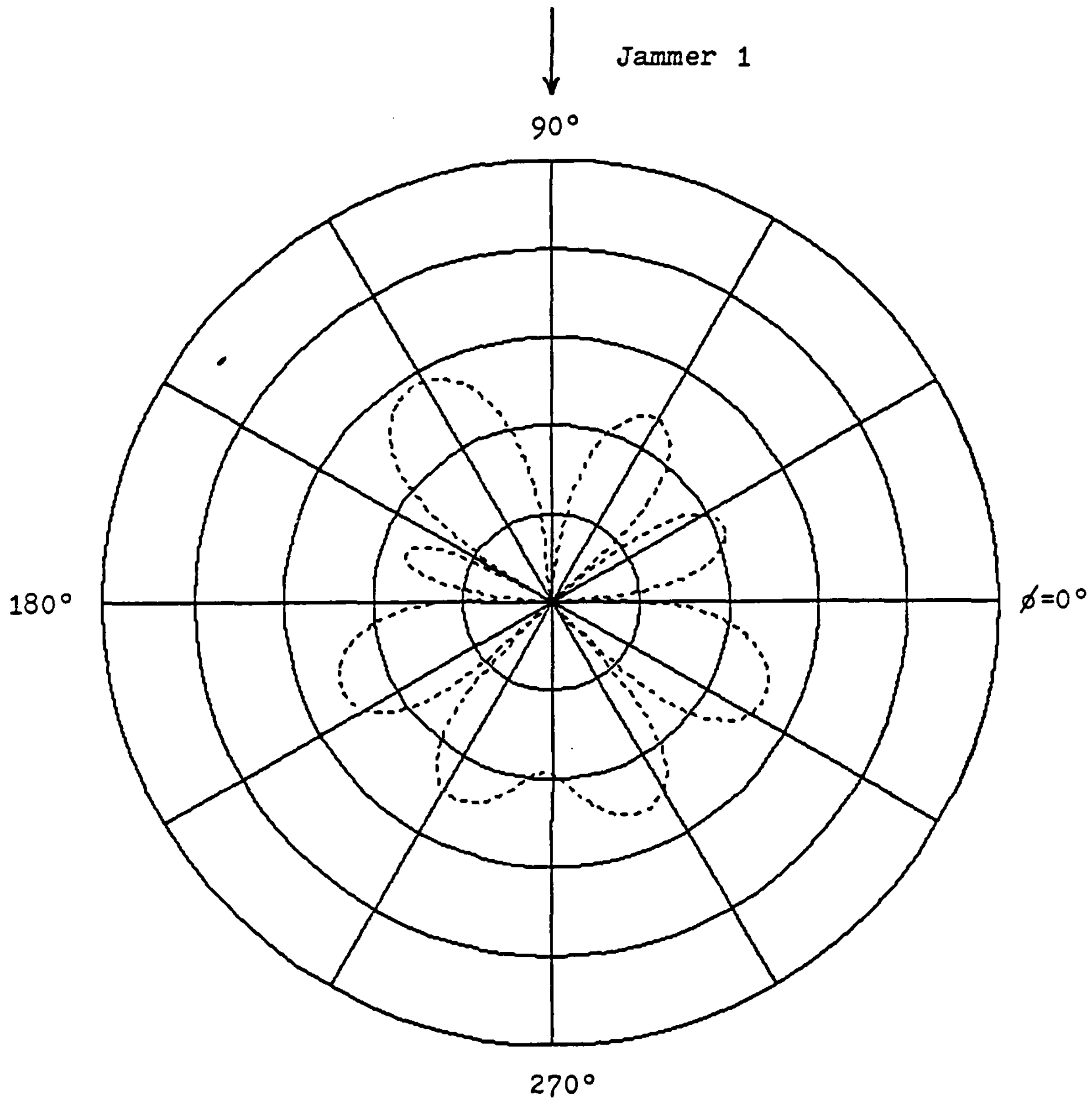


Figure T2.2 LMS-Adapted Pattern in Direction of Jammer 1

(Figure B)

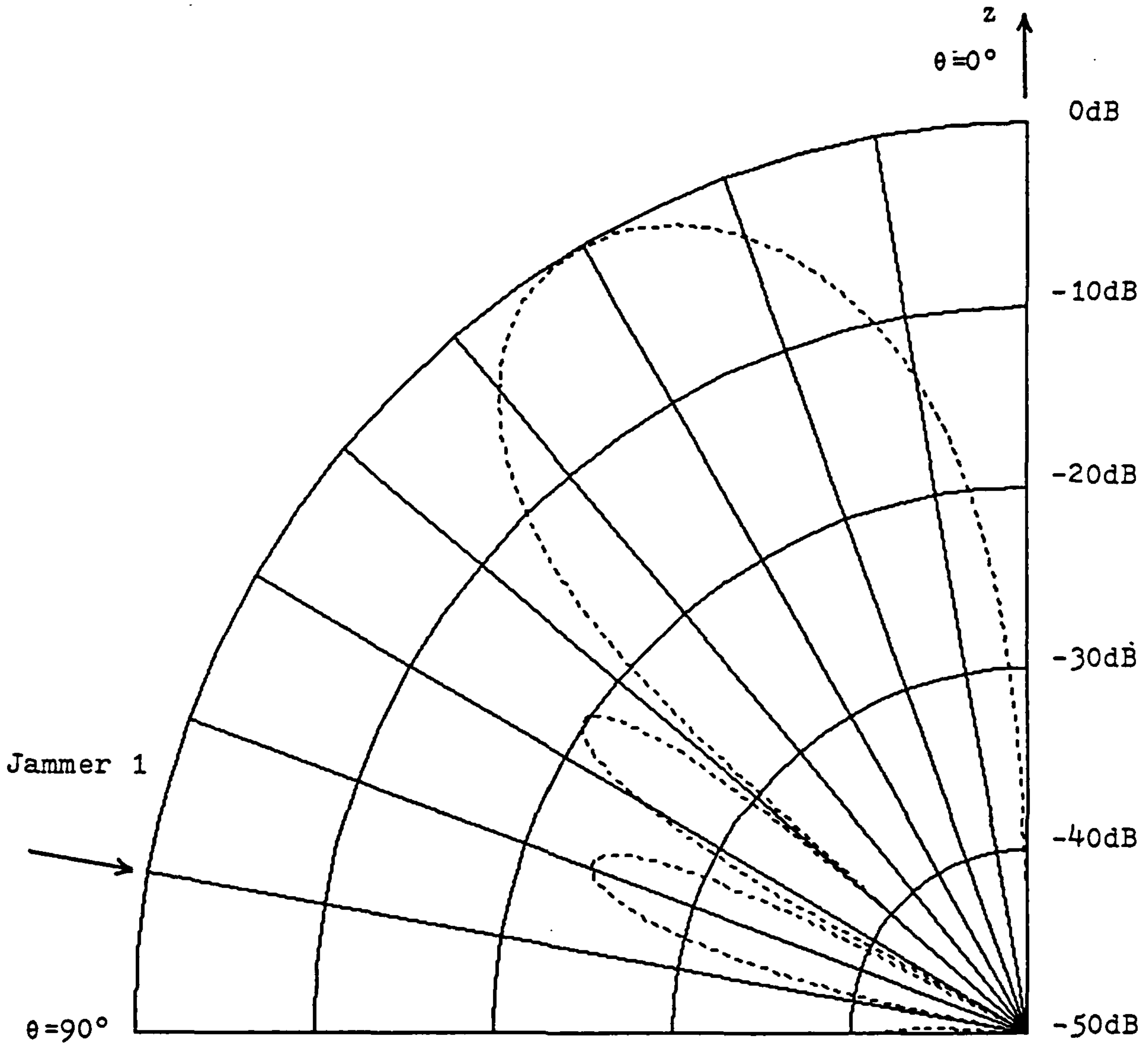


Figure T2.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

(Figure A)

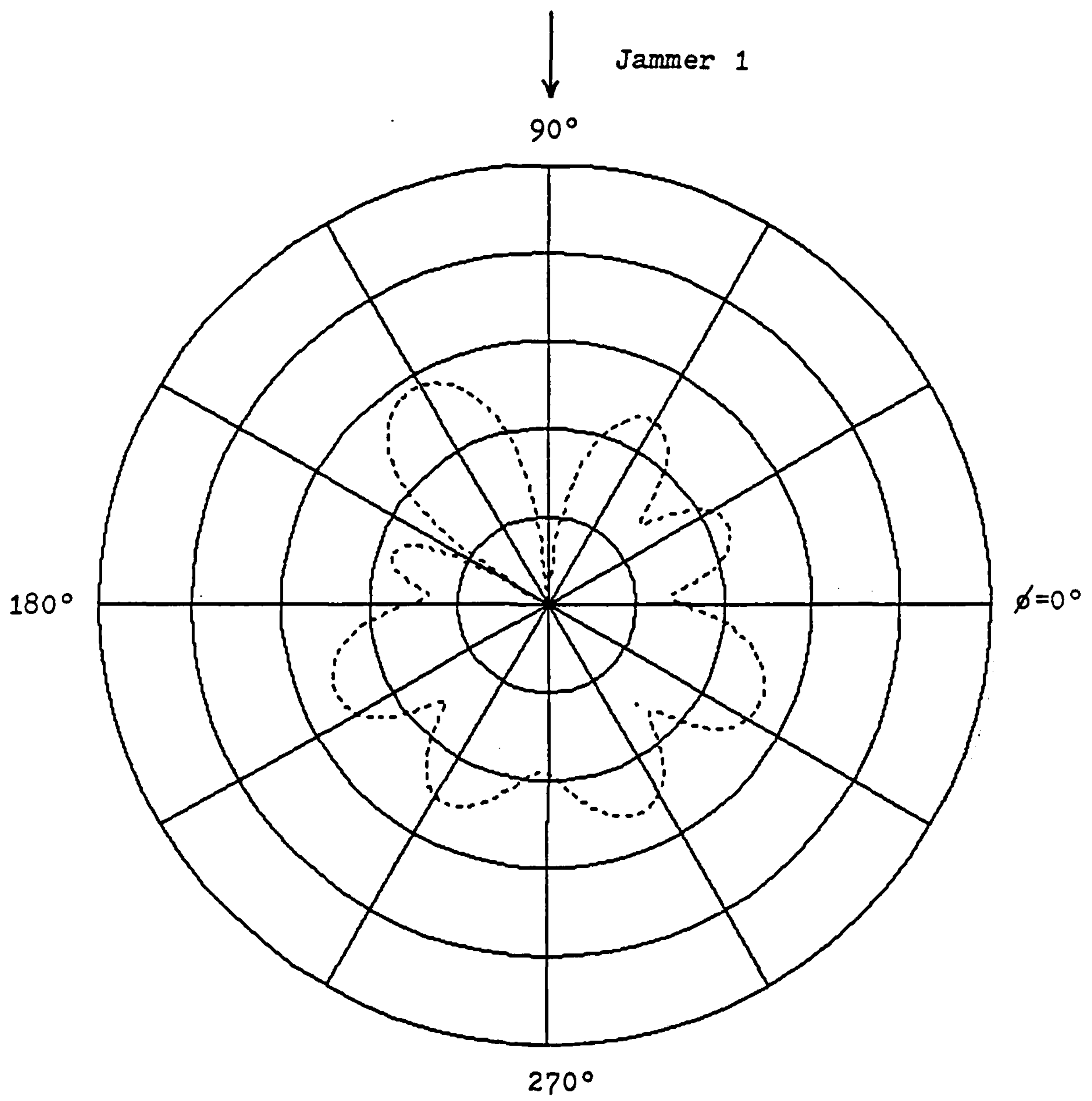


Figure T2.3 Constrained LMS-Adapted Pattern in direction of Jammer 1

(Figure B)

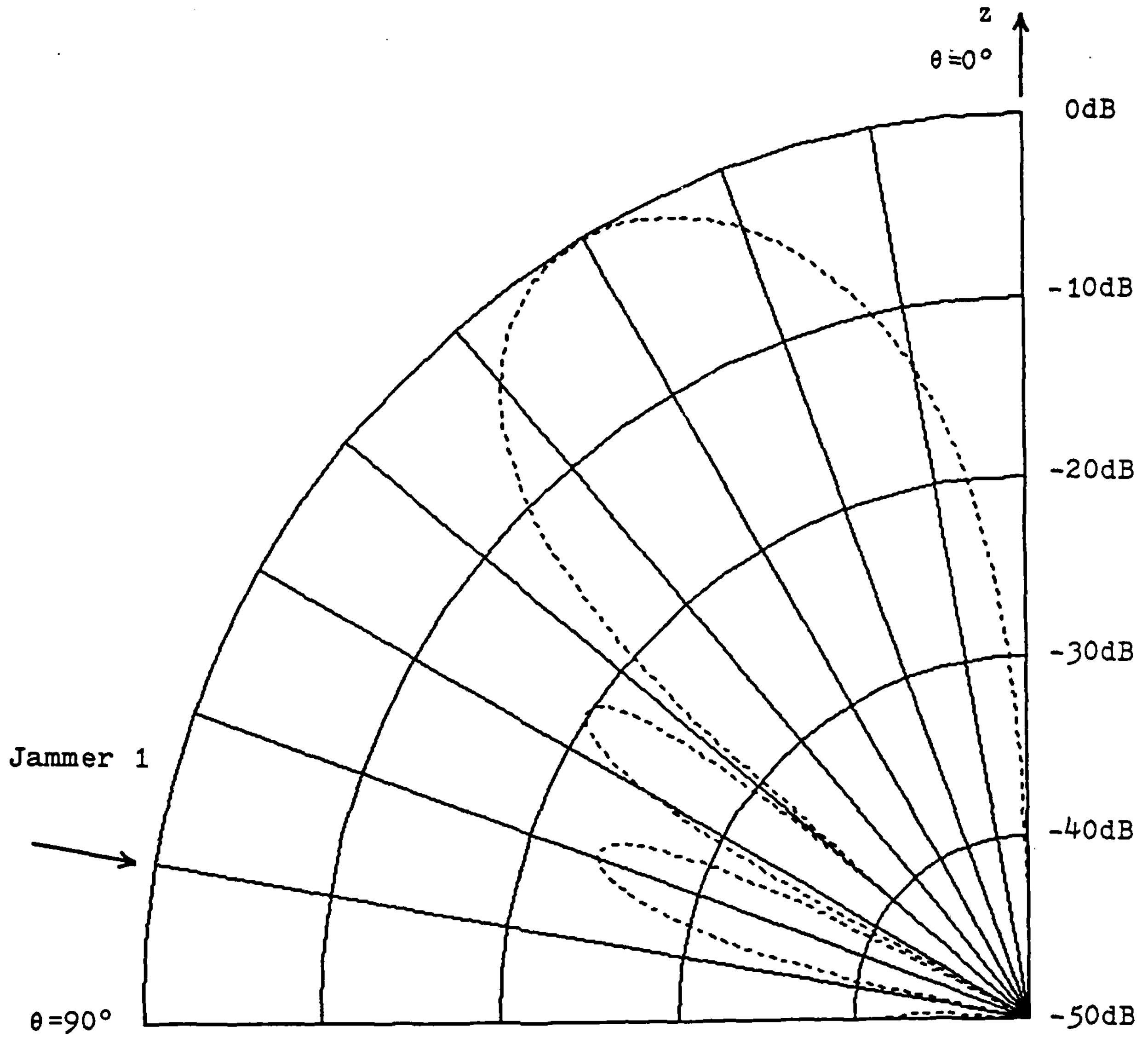


Figure T2.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure A)

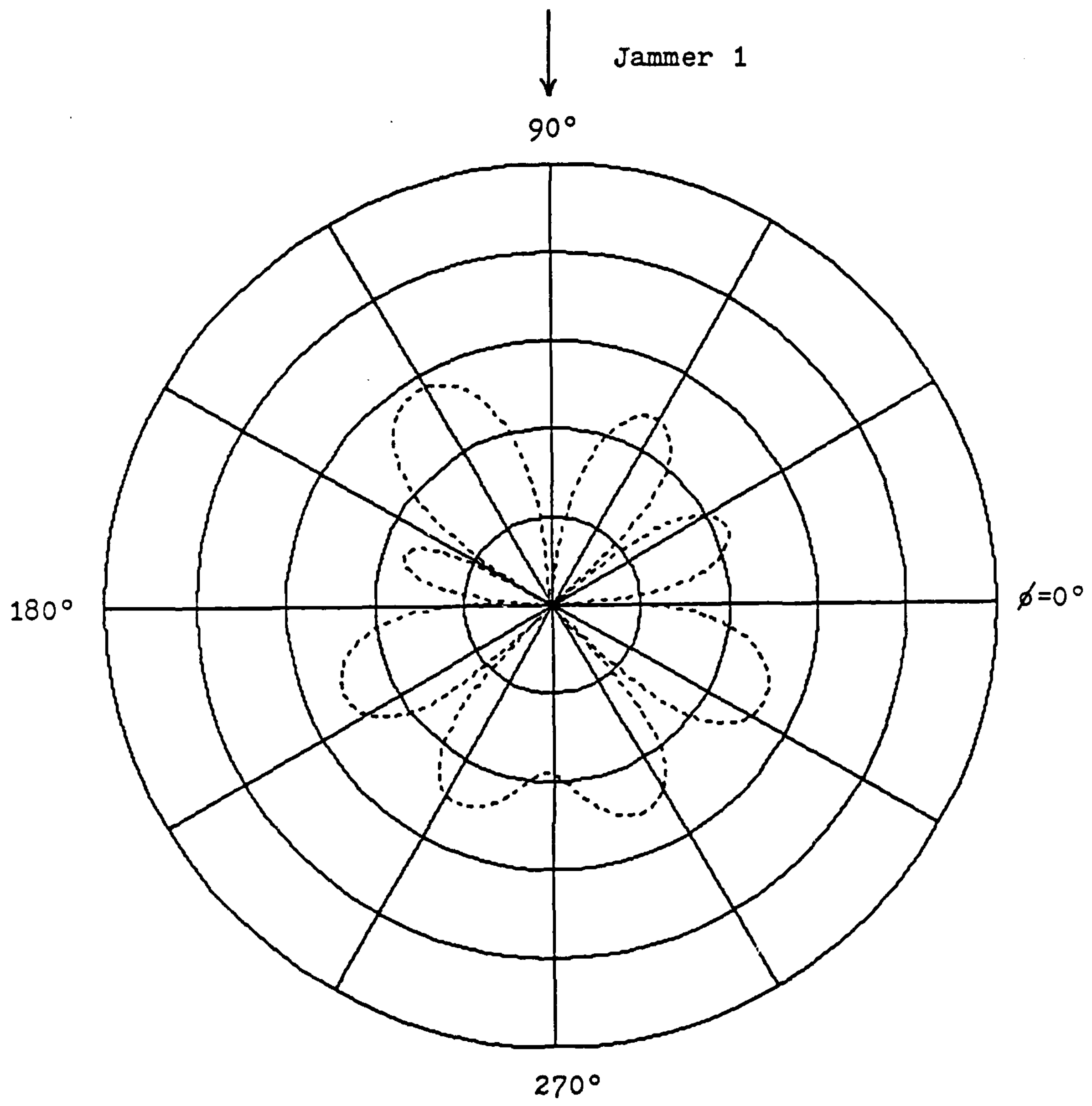
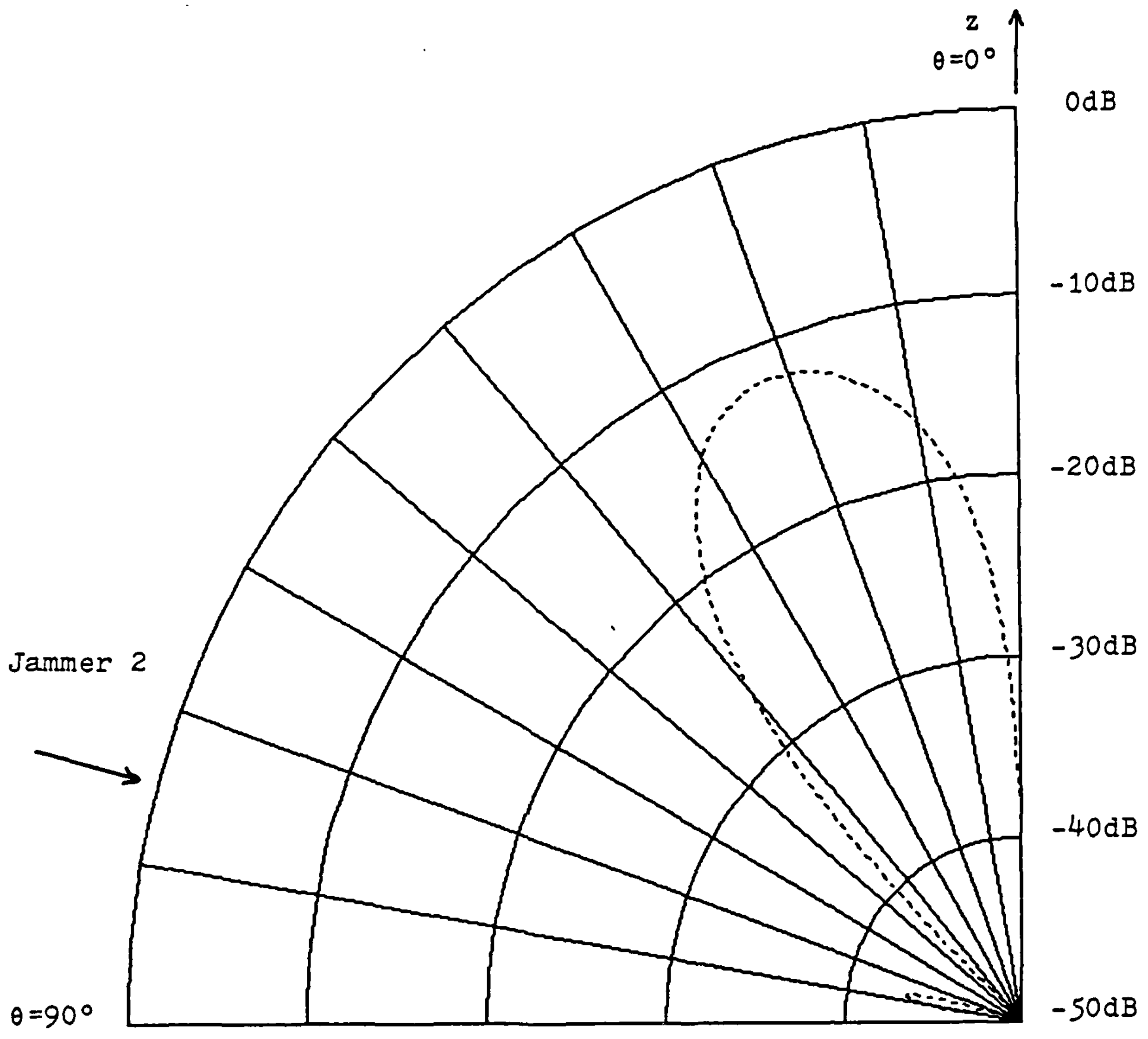


Figure T2.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure B)

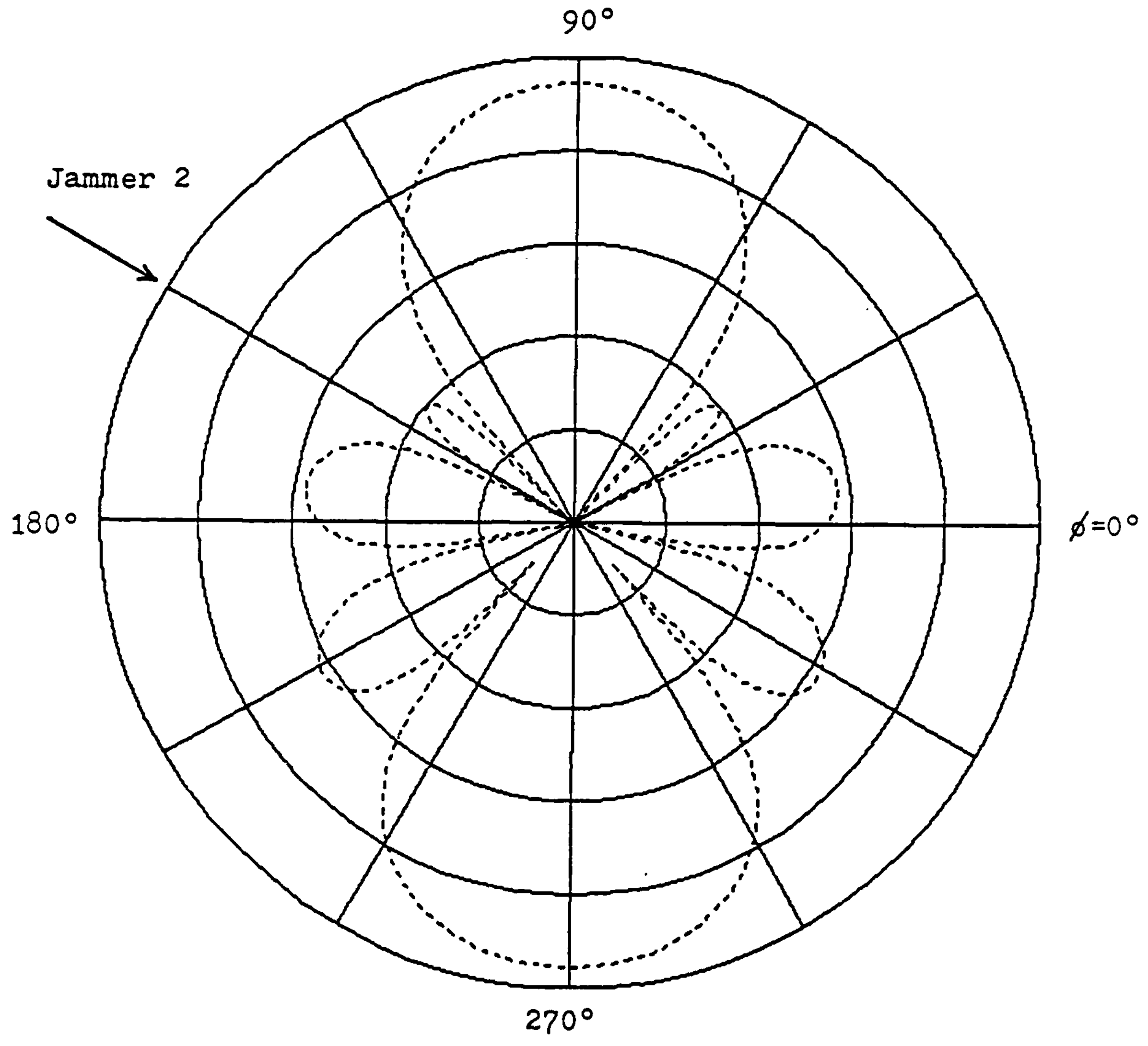


Elevation plot at 150° azimuth

Arrow indicates arrival angle of Jammer 2 (θ=75°)

Figure T2.5 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure A)



Azimuthal plot at 75° elevation

Arrow indicates arrival angle of Jammer 2 (ø=150°)

Figure T2.5 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure B)

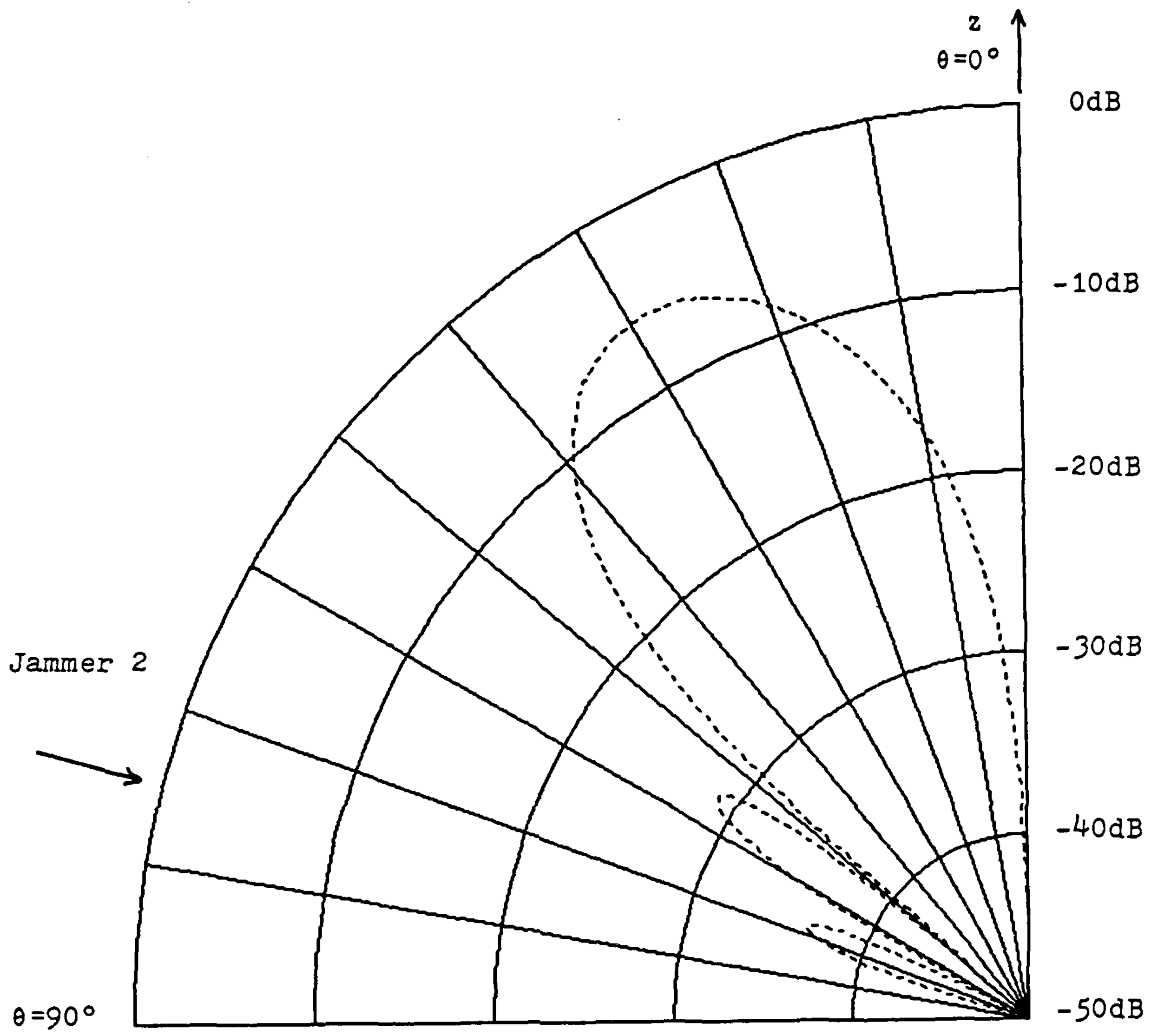


Figure T2.6 LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

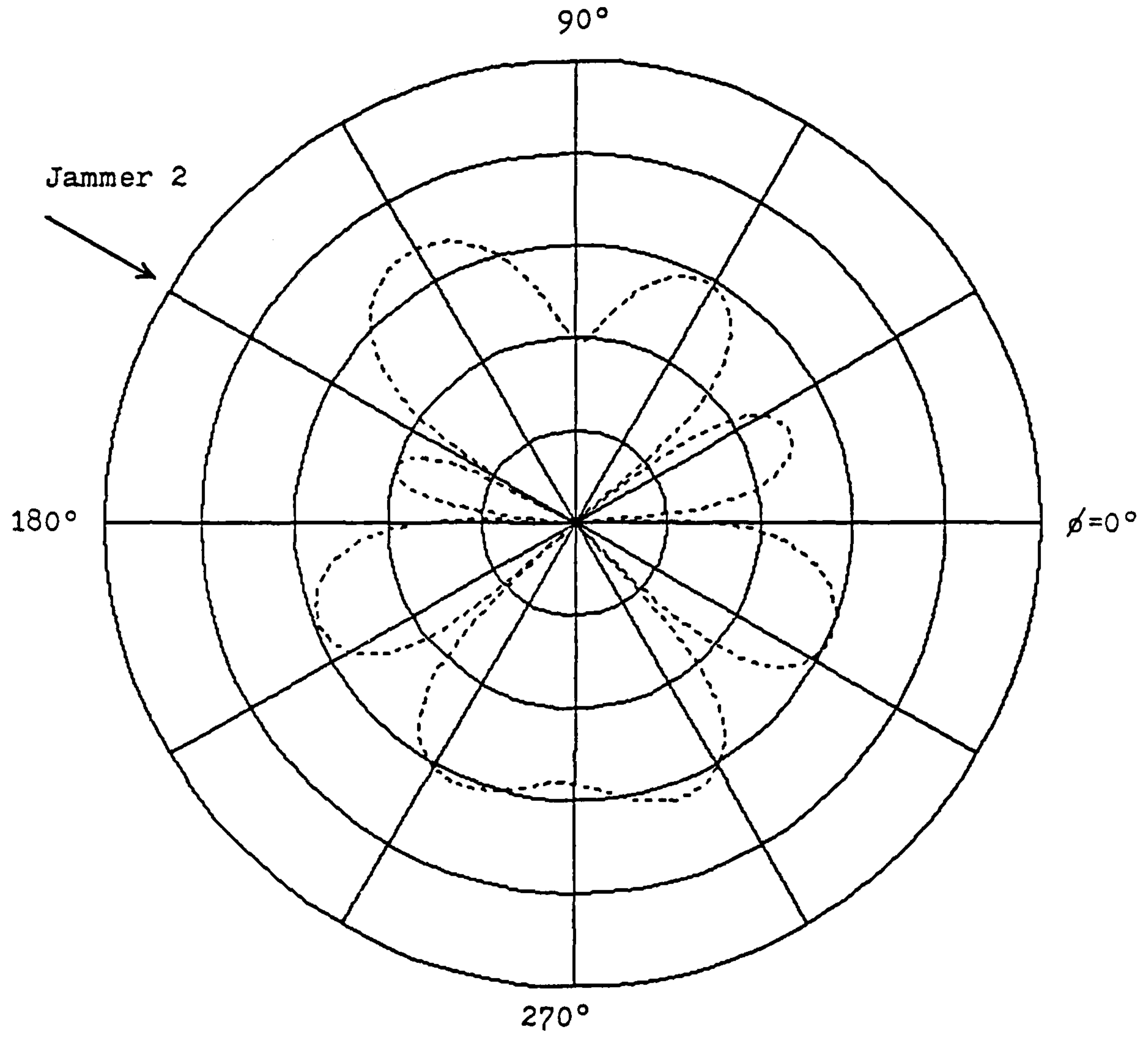


Figure T2.6 LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

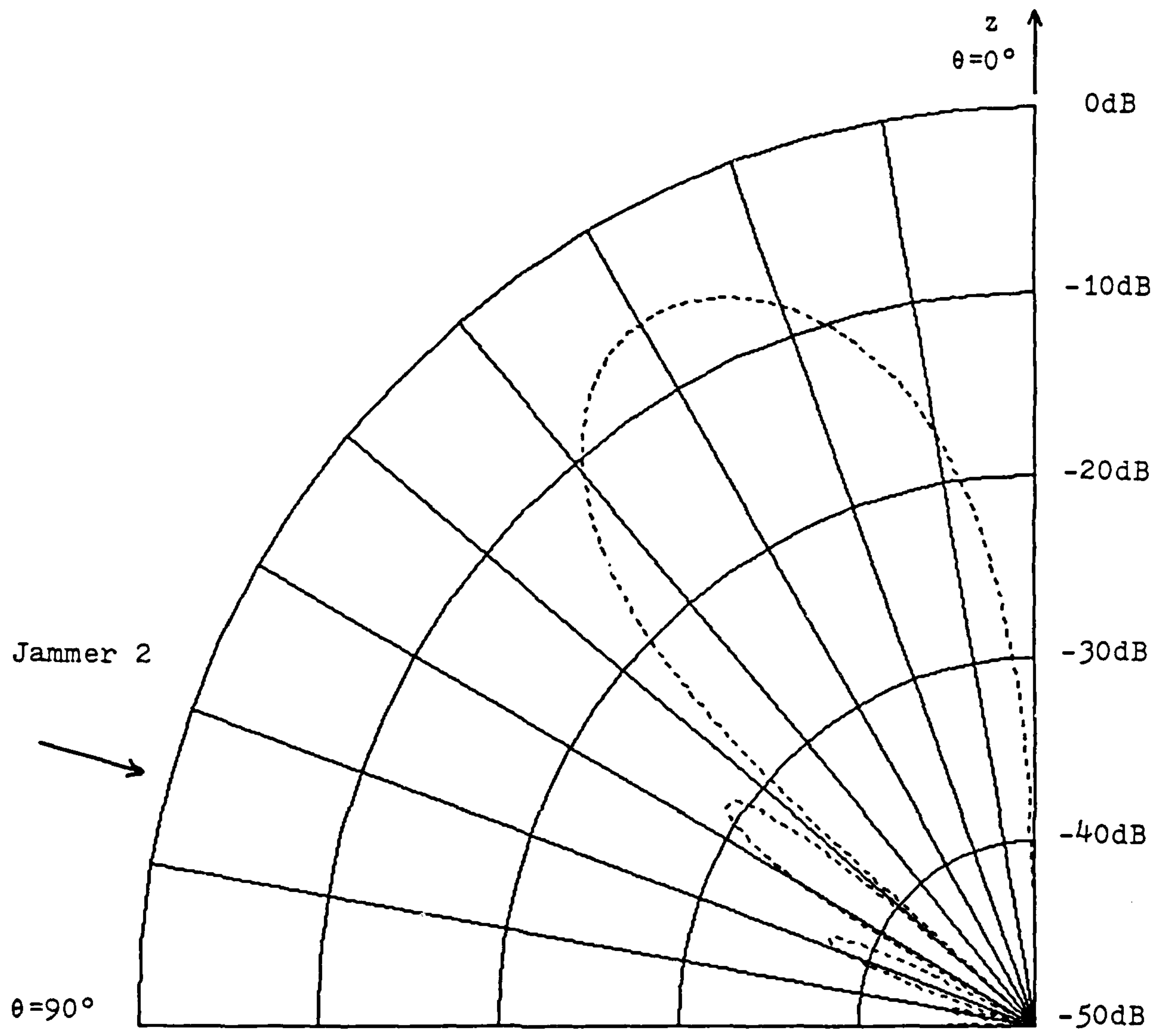


Figure T2.7 Constrained LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

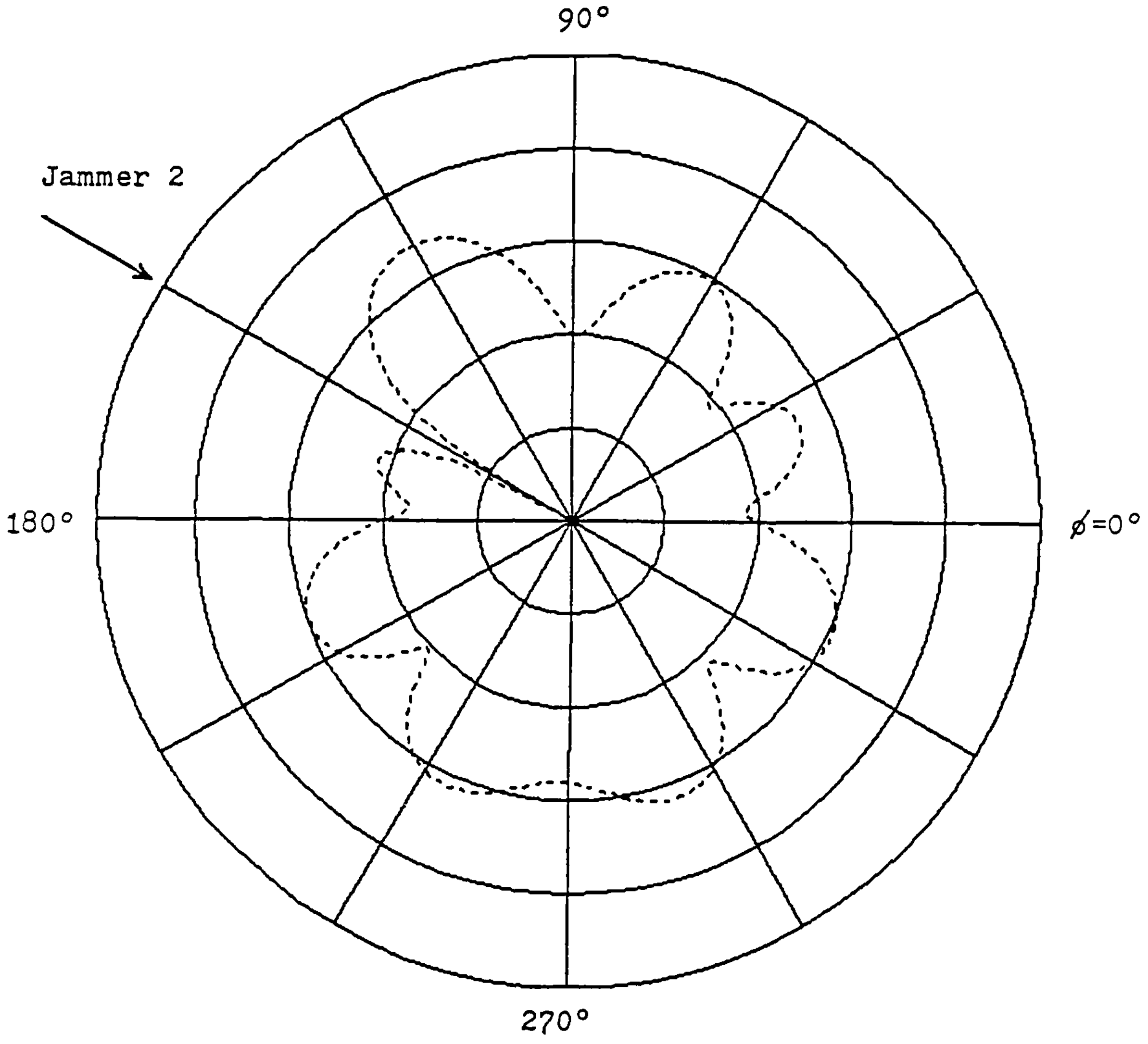


Figure T2.7 Constrained LMS-Adapted Pattern
in Direction of Jammer 2

(Figure B)

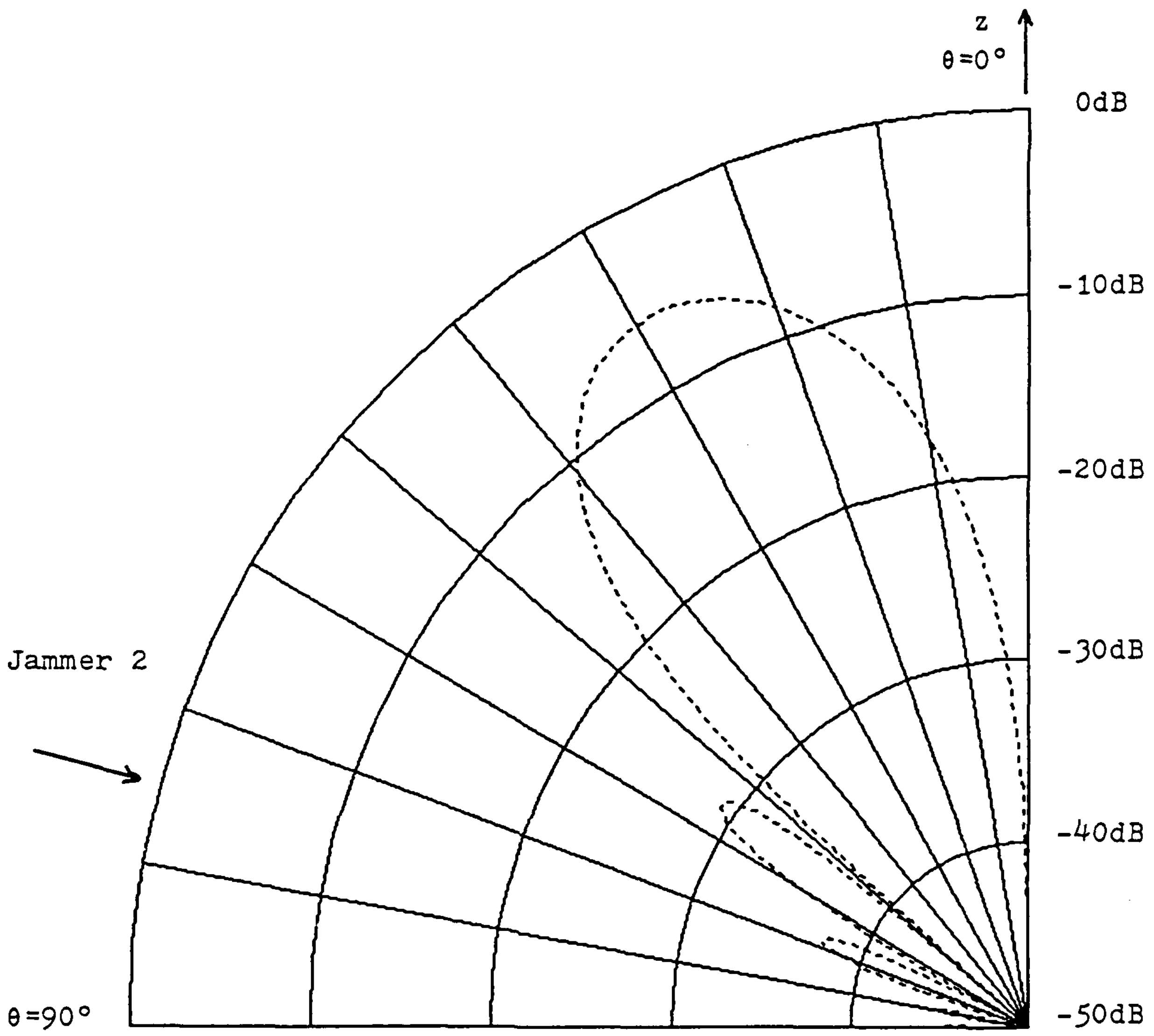


Figure $\bar{\underline{T}}$2.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure A)

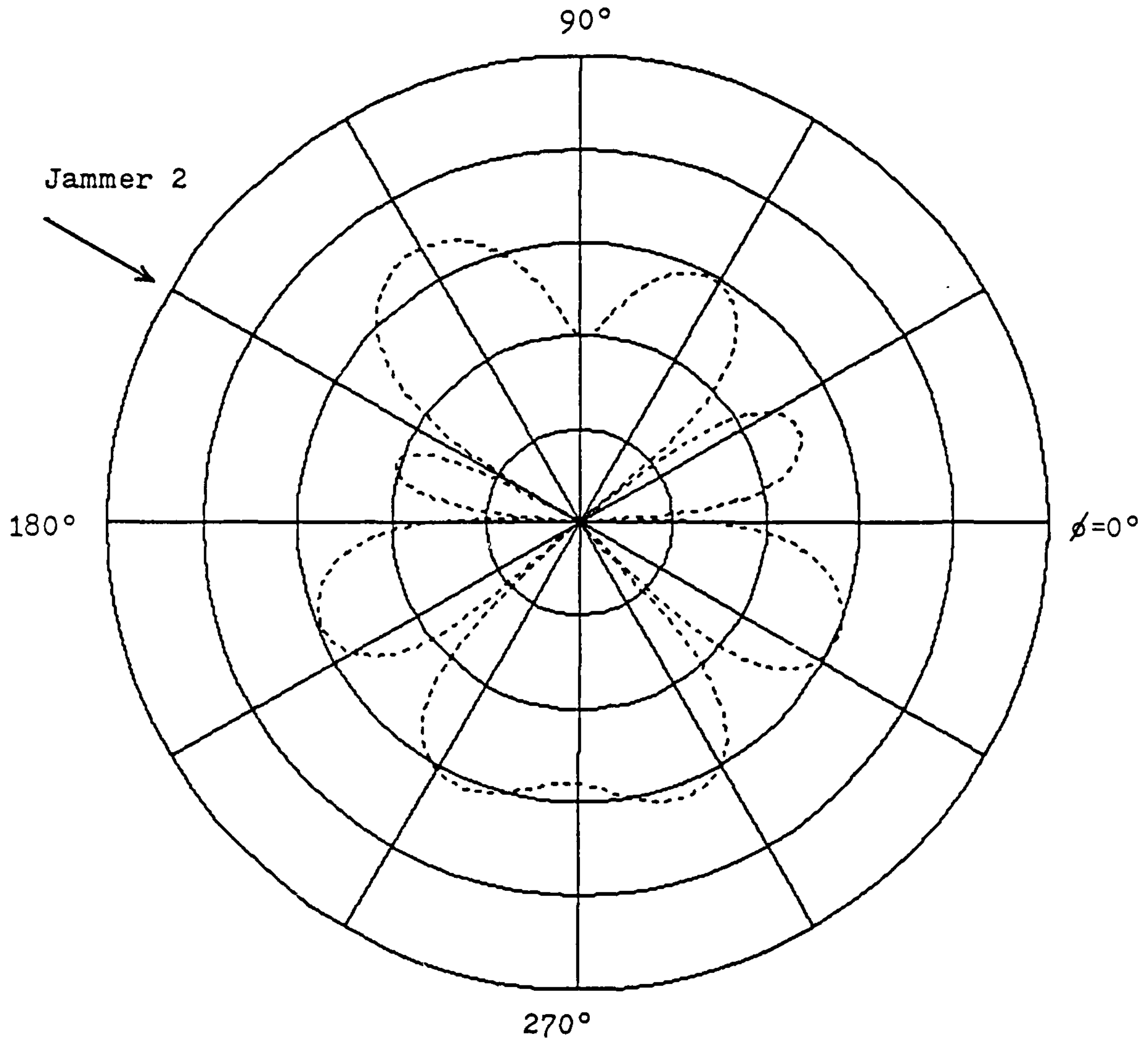


Figure T2.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure B)

7.3 TEST3: HF Channel with Poor Delay Characteristics and Poor Attenuation Characteristics

This test differs from TEST2 in that the delay between signals arriving from different paths is now doubled to 1.6666 msecs.

Once again, results for all algorithms were averaged over 100 convergences. The convergences were monitored and histograms were generated which indicate the nature of when the algorithms converged. The SNR tolerance was set to 1 dB for all trials. All of the algorithms placed deep nulls in the directions of the jamming signals, as the plots that follow will verify. The results are shown in Table 7.3.

The Update Covariance algorithm again required the fewest number of iterations by a wide margin. It is also important to note that the LMS algorithm actually required fewer computations on the average to converge. Therefore, in real time, the LMS algorithm converged faster, making the assumption that all computations take the same amount of time.

The Constrained LMS algorithm was the "loser" again. But the average number of convergence iterations did not change significantly from TEST2. This seems to indicate that the covergence difficulties are more related to the magnification of the signal (existence of three paths) than the delay between the paths.

The results indicate that the delay between the paths did not seriously affect the performance of the LMS algorithm. Again, this fact suggests that the existence of three distinct paths plays a more significant role in affecting algorithm performance. The LMS algorithm still converged in a modest number of iterations on the average.

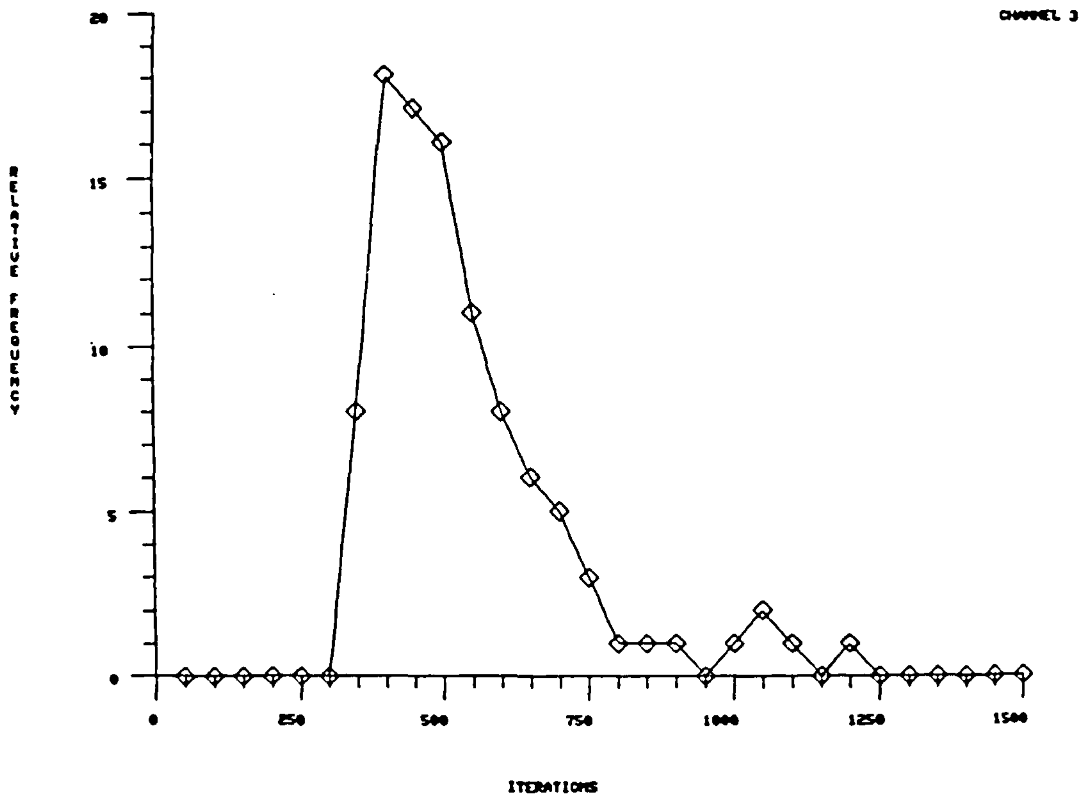


Figure 7.10 Convergence Histogram of LMA Algorithm in Channel 3

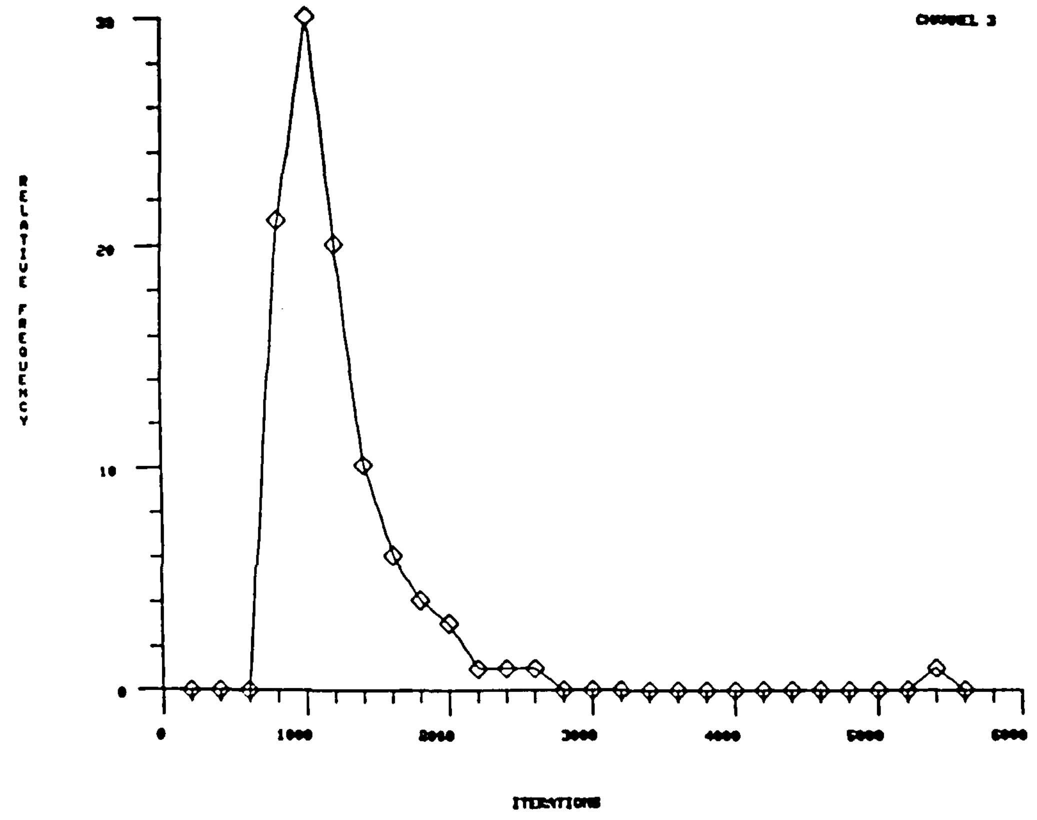


Figure 7.11 Convergence Histogram of Constrained LMS Algorithm in Channel 3

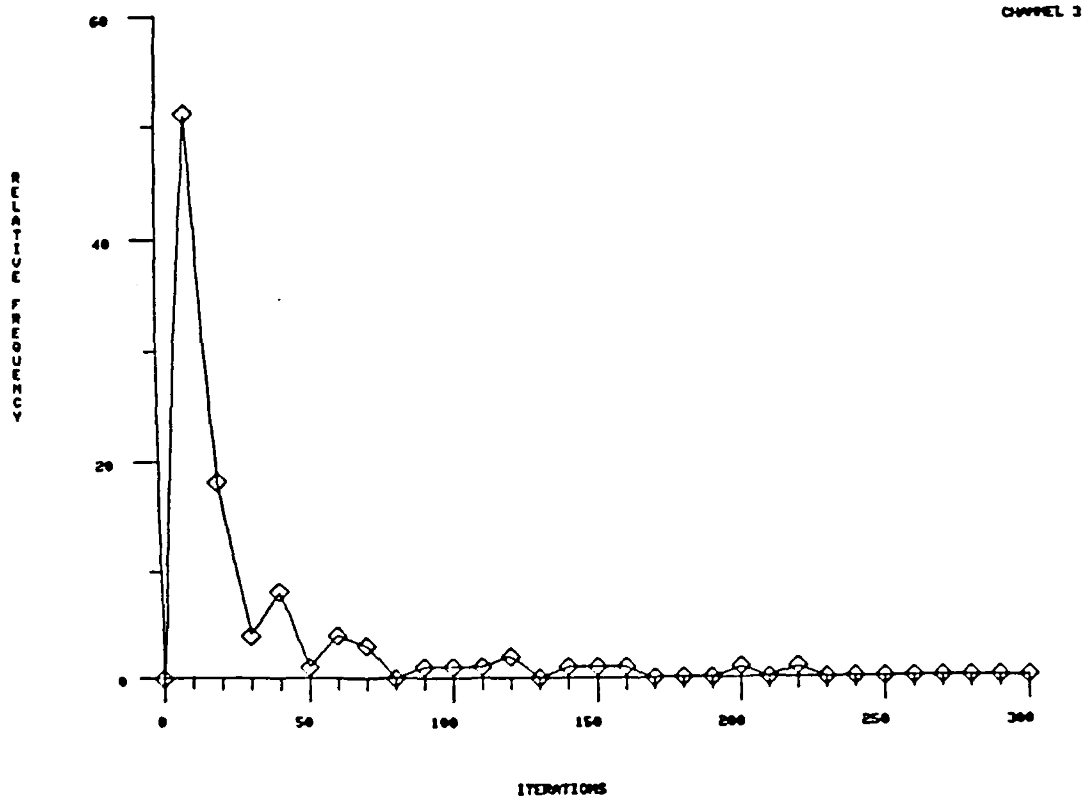


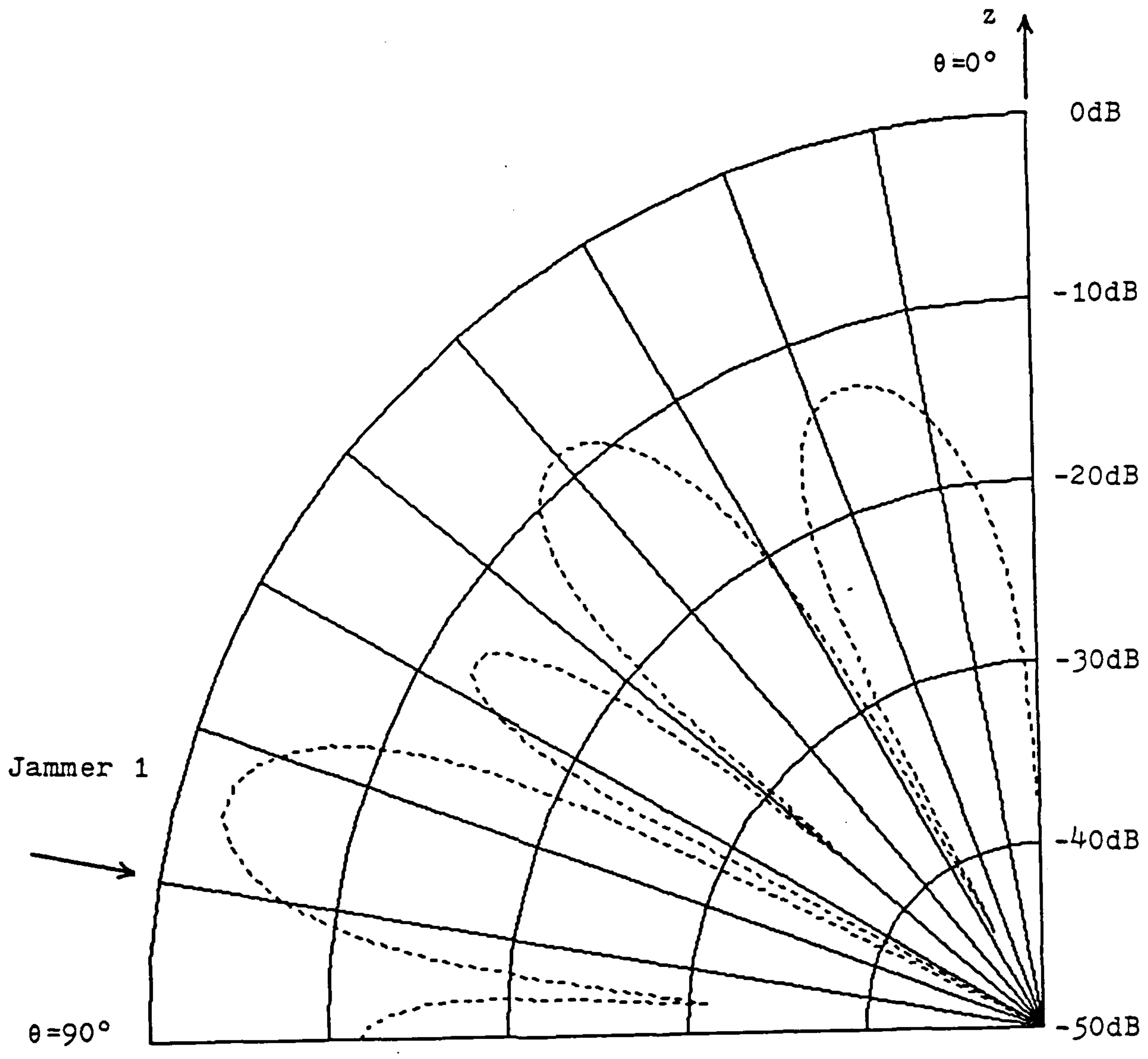
Figure 7.12 Convergence Histogram of Update Covariance Algorithm in Channel 3

TABLE 7.3 TEST3 SUMMARY

ADAPTIVE ALGORITHM	NUMBER OF CONVERGENCES	AVERAGE NUMBER OF ITERATIONS	STANDARD DEVIATION
LMS	100	542.5	172.6
CONSTR. LMS	100	1143.5	542.6
UPDATE COV.	100	28.8	43.1

SUMMARY OF PLOTS FOR TEST3

Figure T3.1:	Unadapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To indicate initial gain in direction of Jammer 1
Figure T3.2:	LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T3.3:	Constrained LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T3.4:	Update Covariance-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To demonstrate nulling of Jammer 1 by each algorithm
Figure T3.5:	Unadapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To indicate initial gain in direction of Jammer 2
Figure T3.6:	LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T3.7:	Constrained LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T3.8:	Update Covariance-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To demonstrate nulling of Jammer 2 by each algorithm

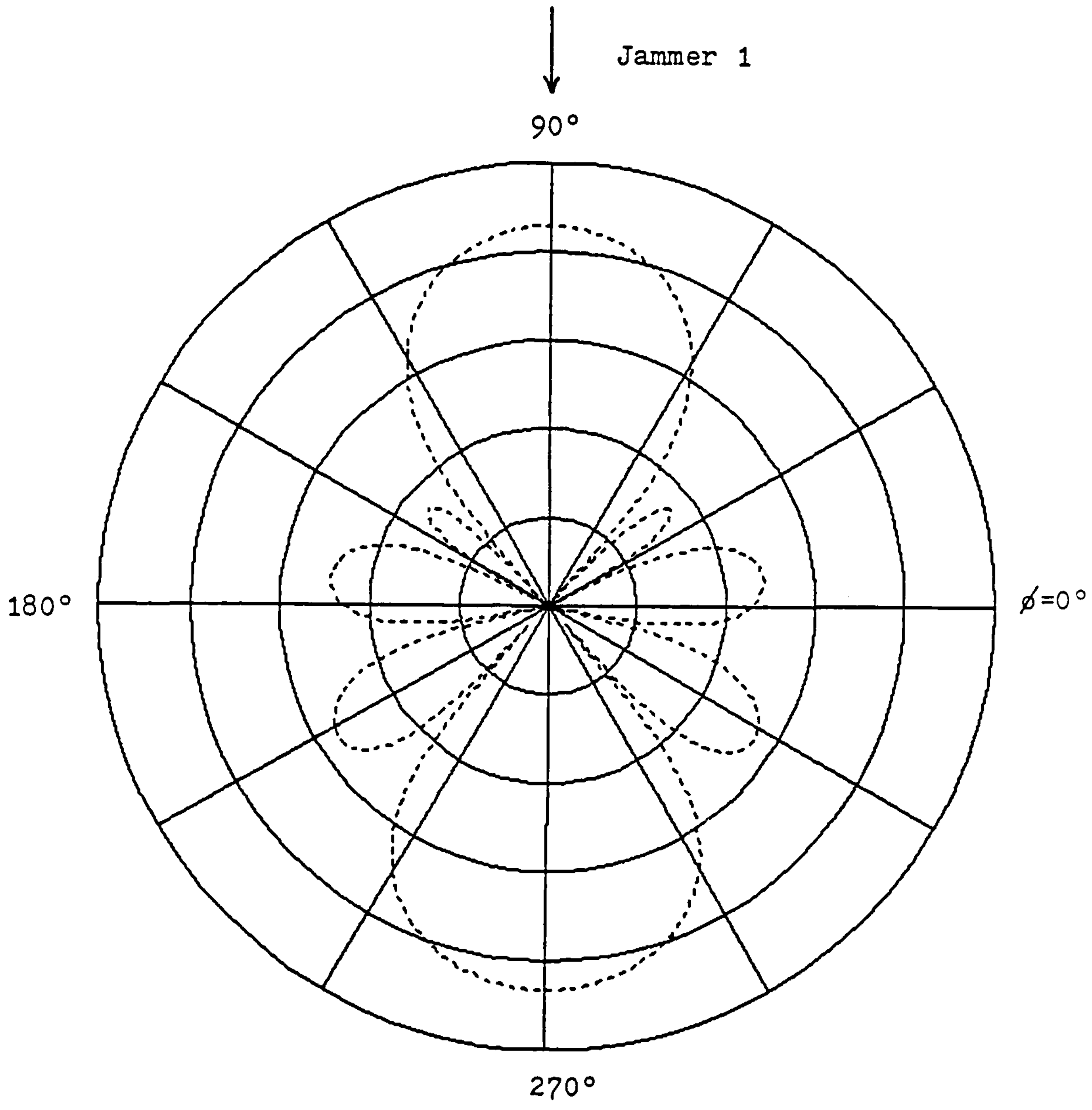


Elevation plot at 90° azimuth

Arrow indicates arrival angle of Jammer 1 (θ=80°)

Figure T3.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure A)



Azimuthal plot at 80° elevation

Arrow indicates arrival angle of Jammer 1 (ϕ=90°)

Figure T3.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure B)

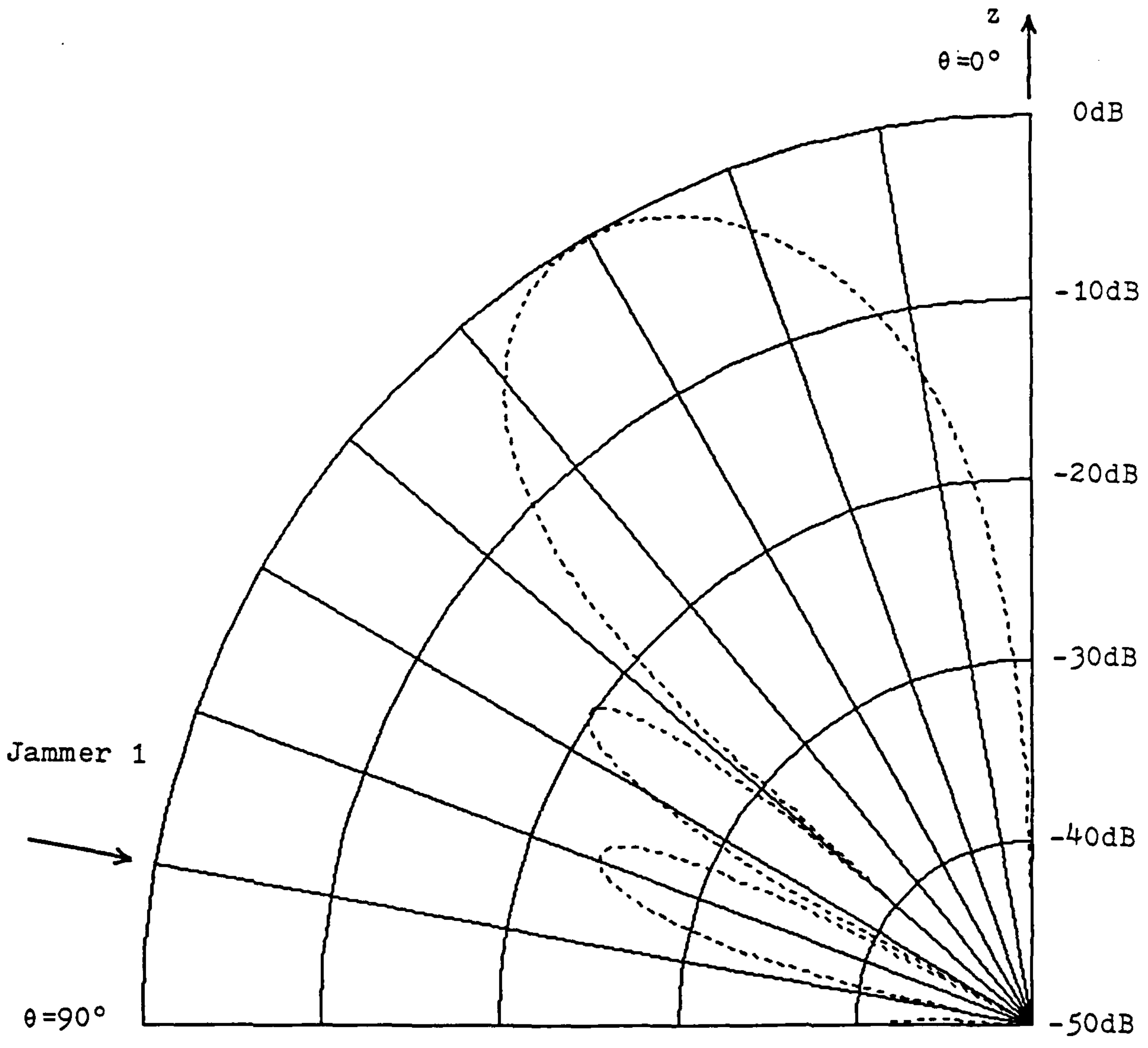


Figure T3.2 LMS-Adapted Pattern in Direction of Jammer 1
(Figure A)

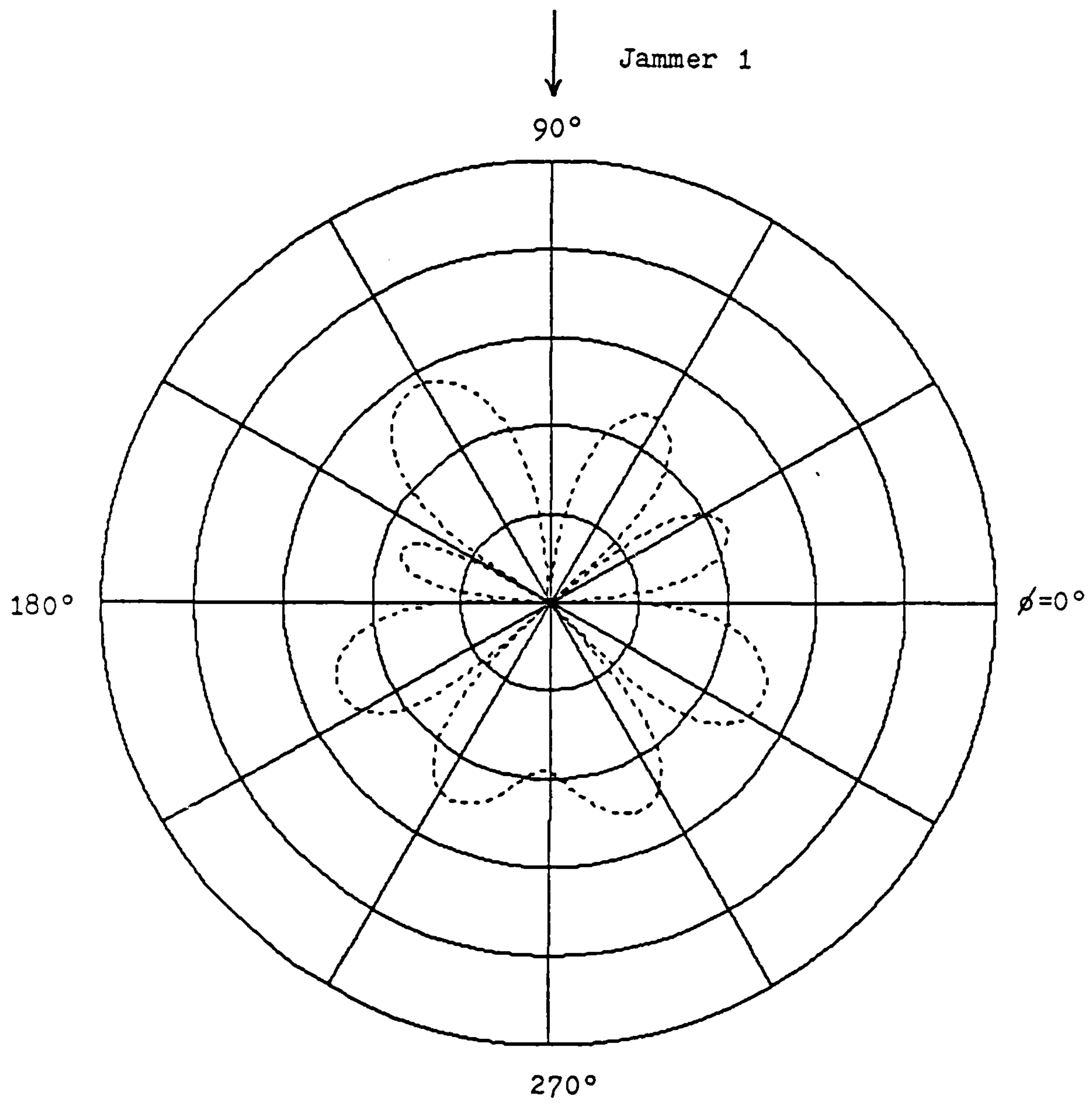


Figure T3.2 LMS-Adapted Pattern in Direction of Jammer 1

(Figure B)

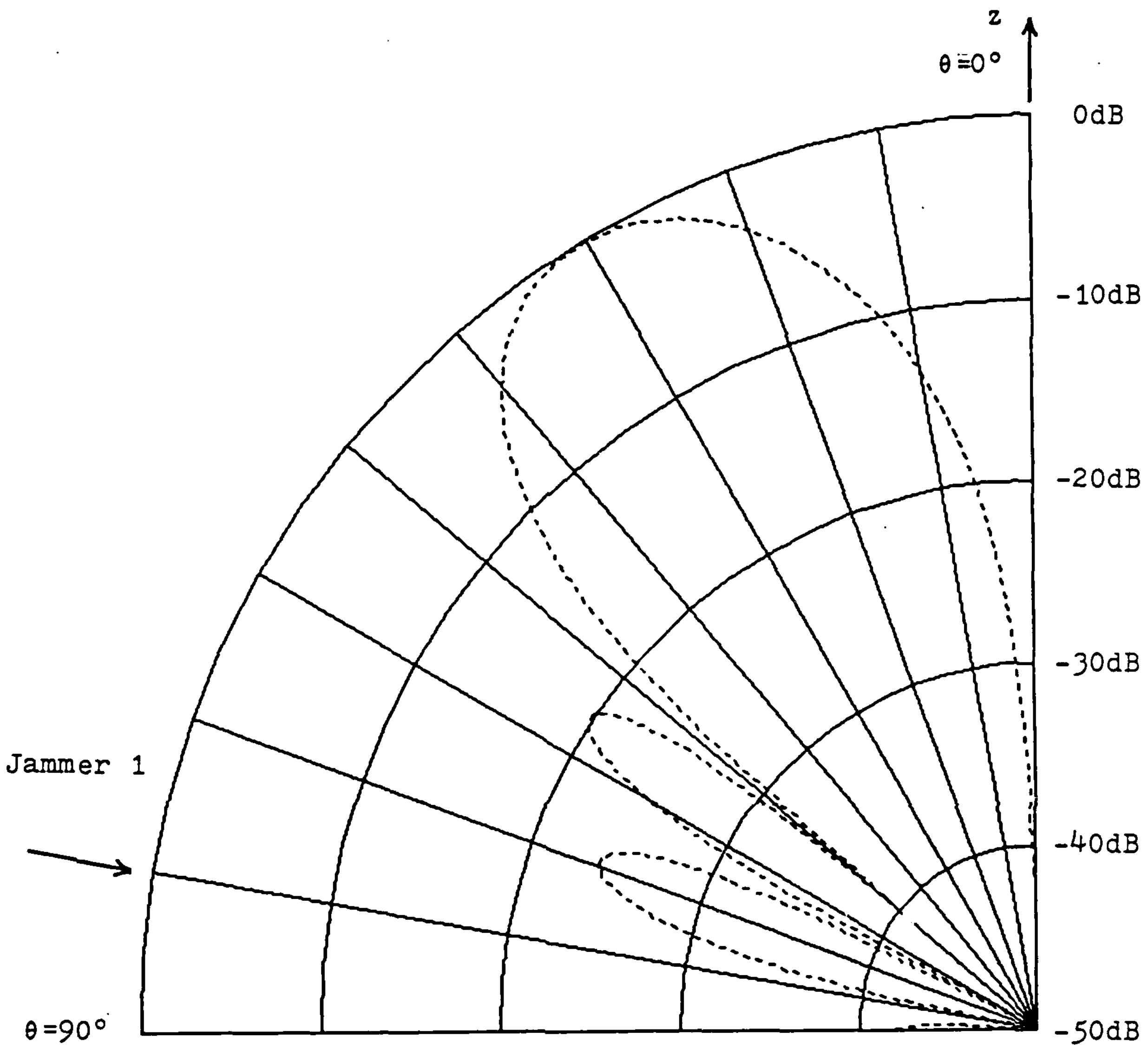


Figure T3.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

(Figure A)

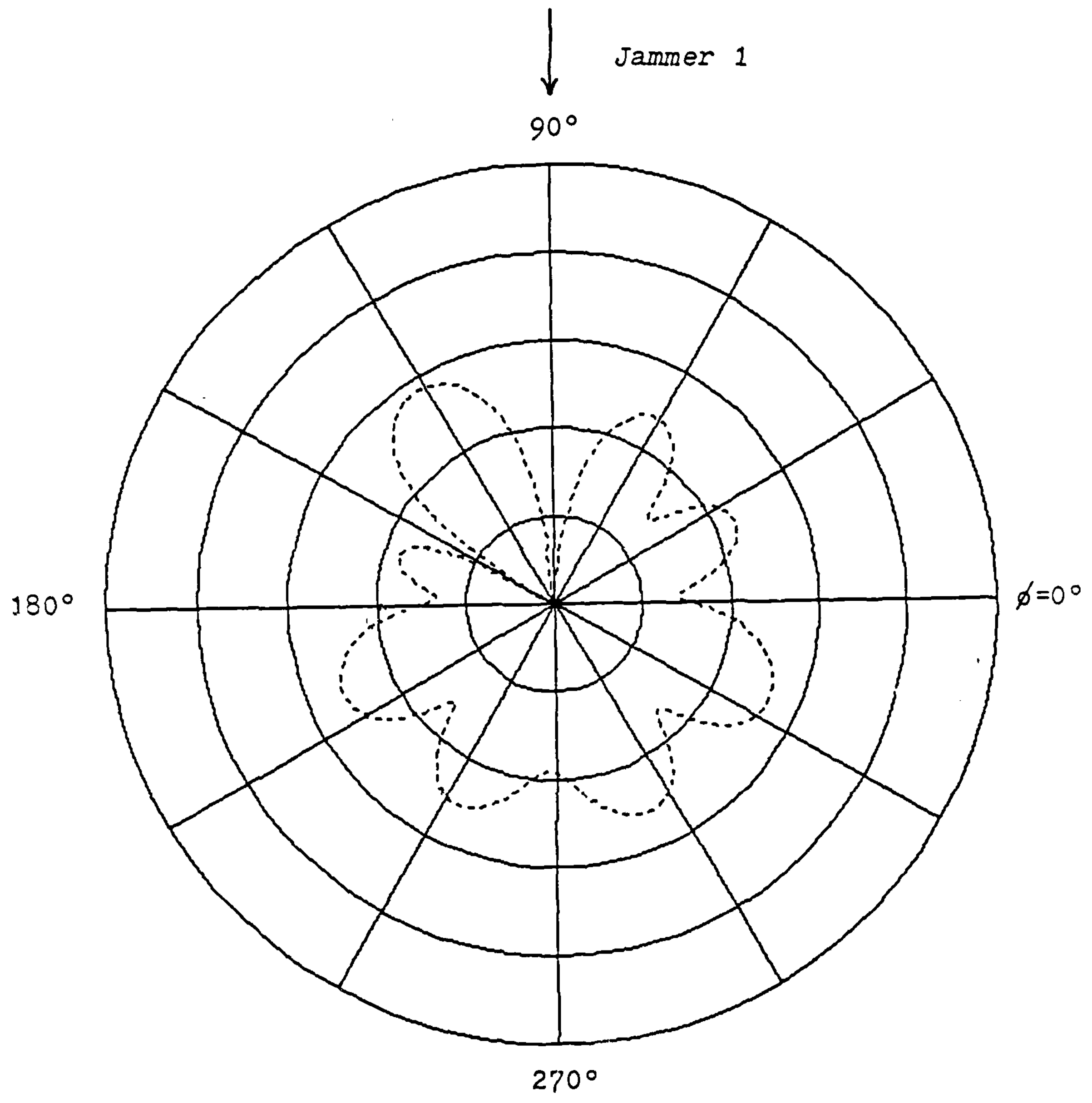


Figure T3.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

(Figure B)

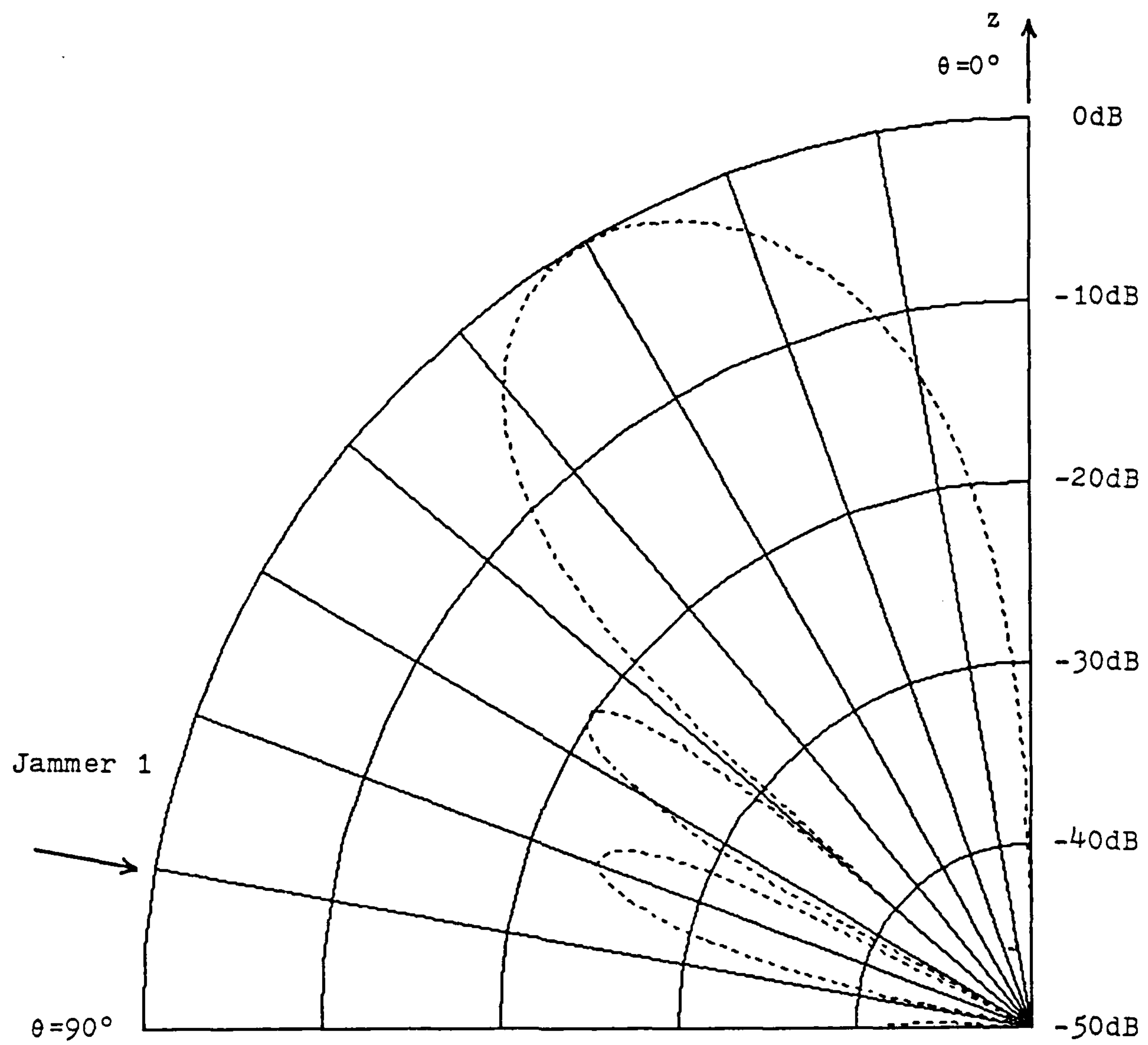


Figure T3.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure A)

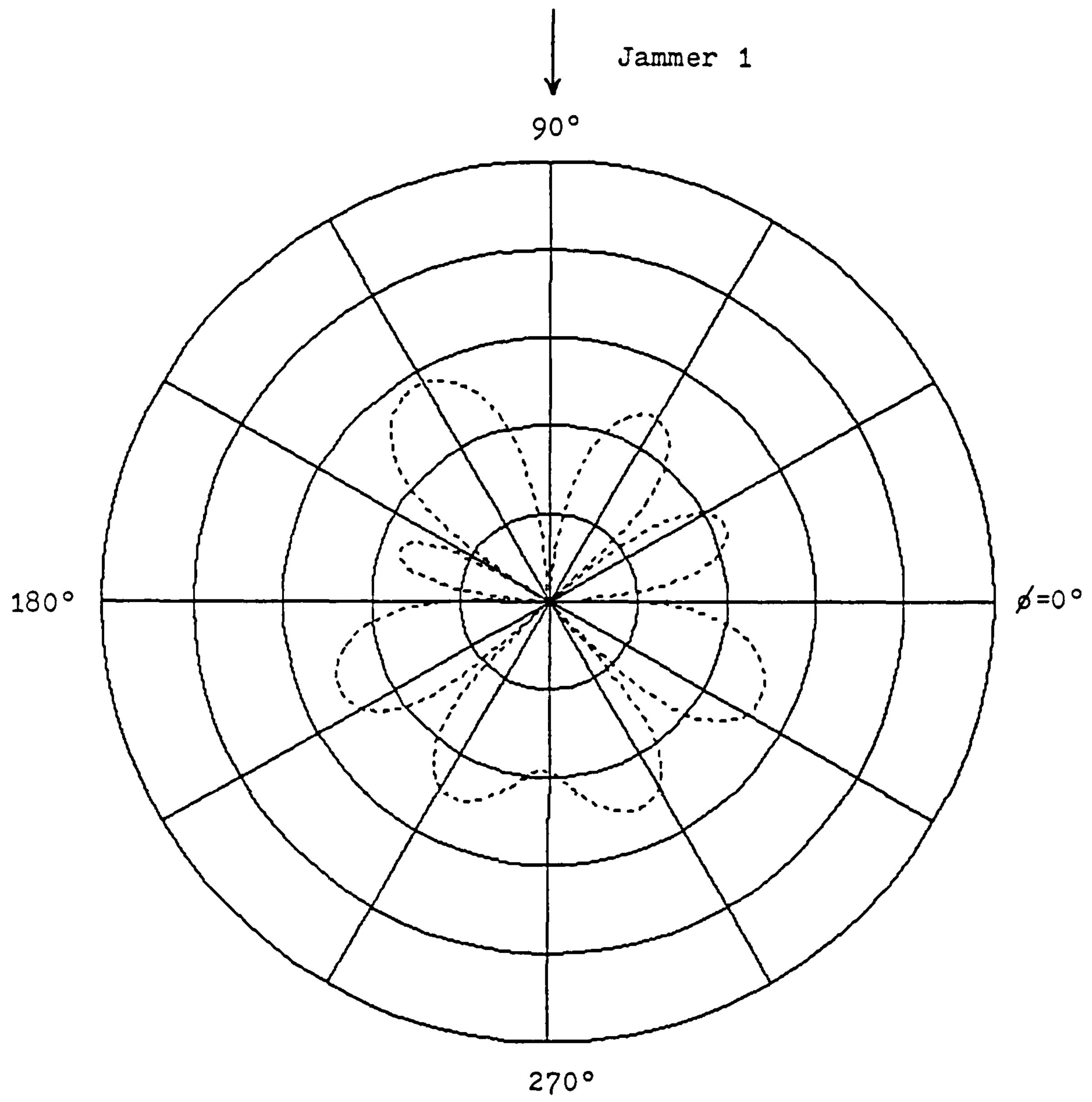
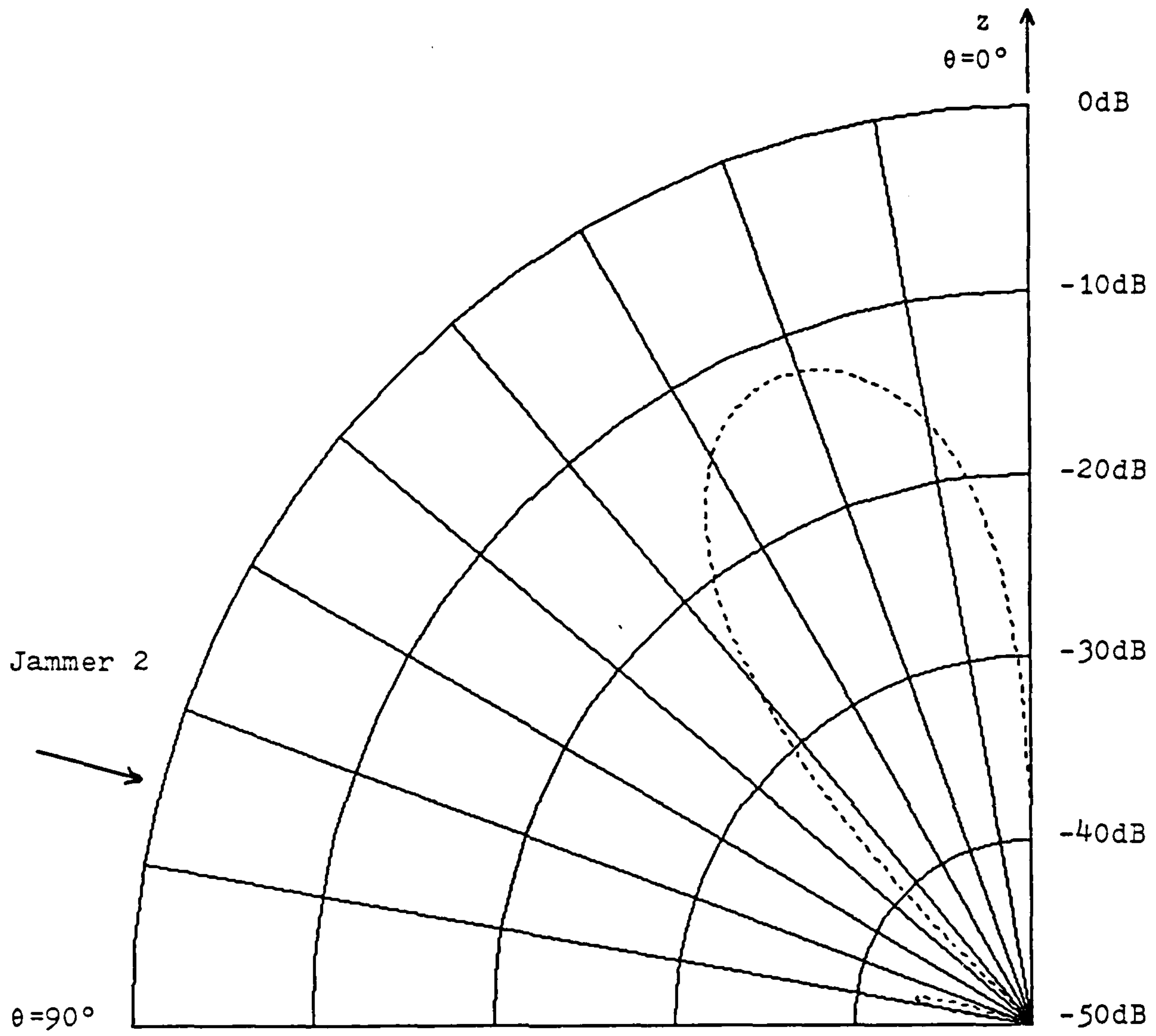


Figure T3.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure B)

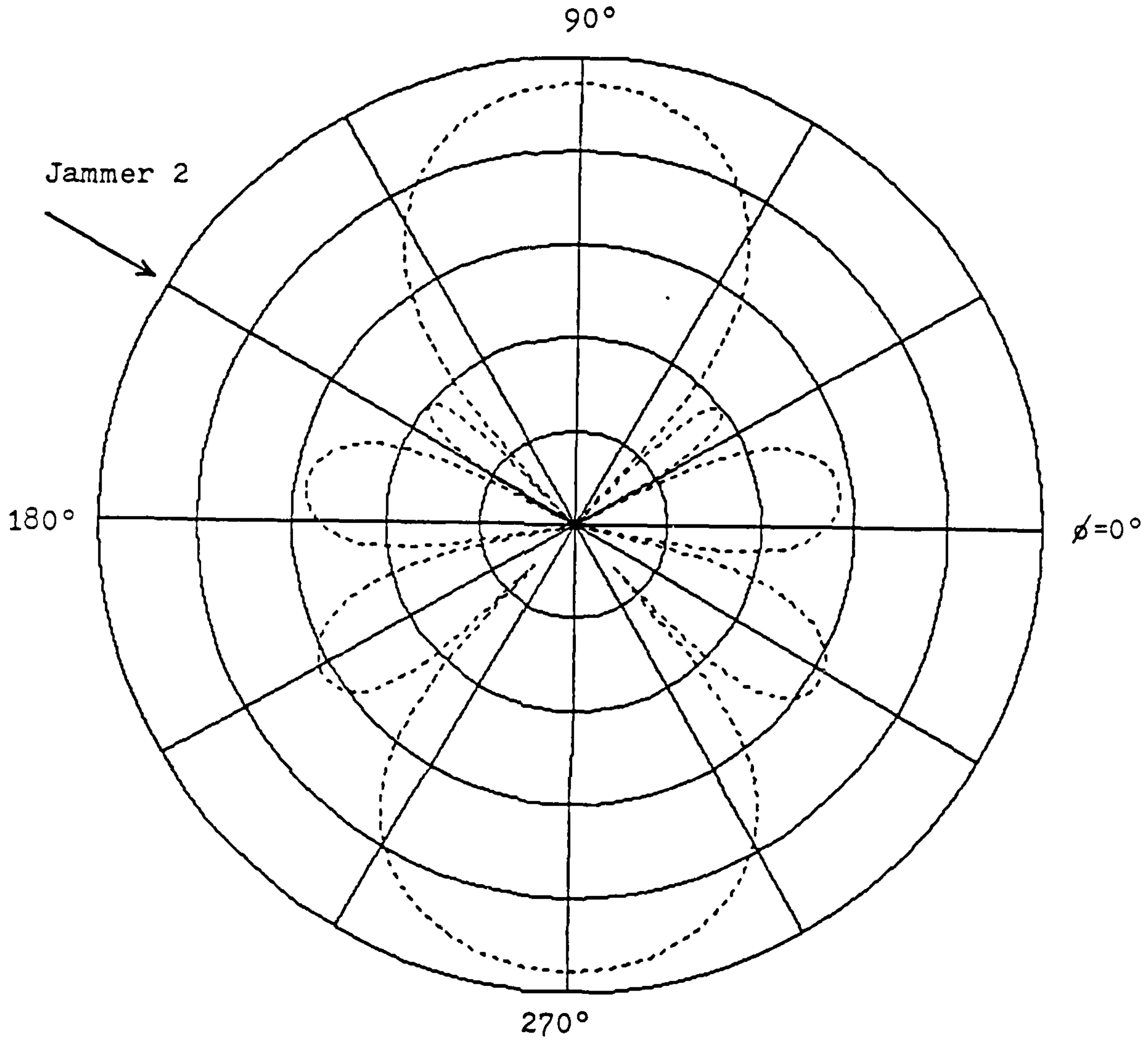


Elevation plot at 150° azimuth

Arrow indicates arrival angle of Jammer 2 (θ=75°)

Figure T3.5 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure A)



Azimuthal plot at 75° elevation

Arrow indicates arrival angle of Jammer 2 (ø=150°)

Figure T3.5 Unadapted Antenna Pattern in Direction of Jammer 2 (Figure B)

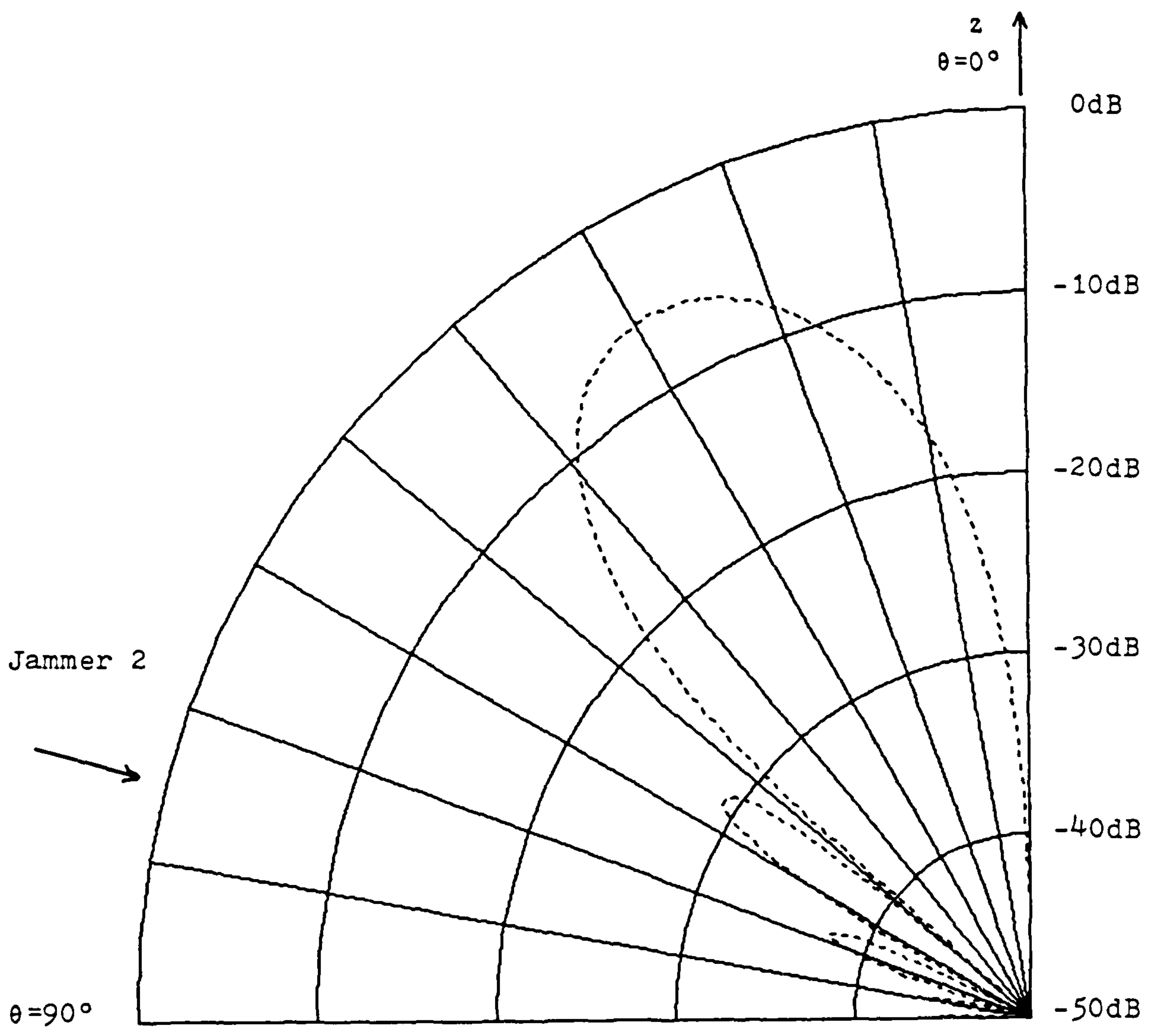


Figure T3.6 LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

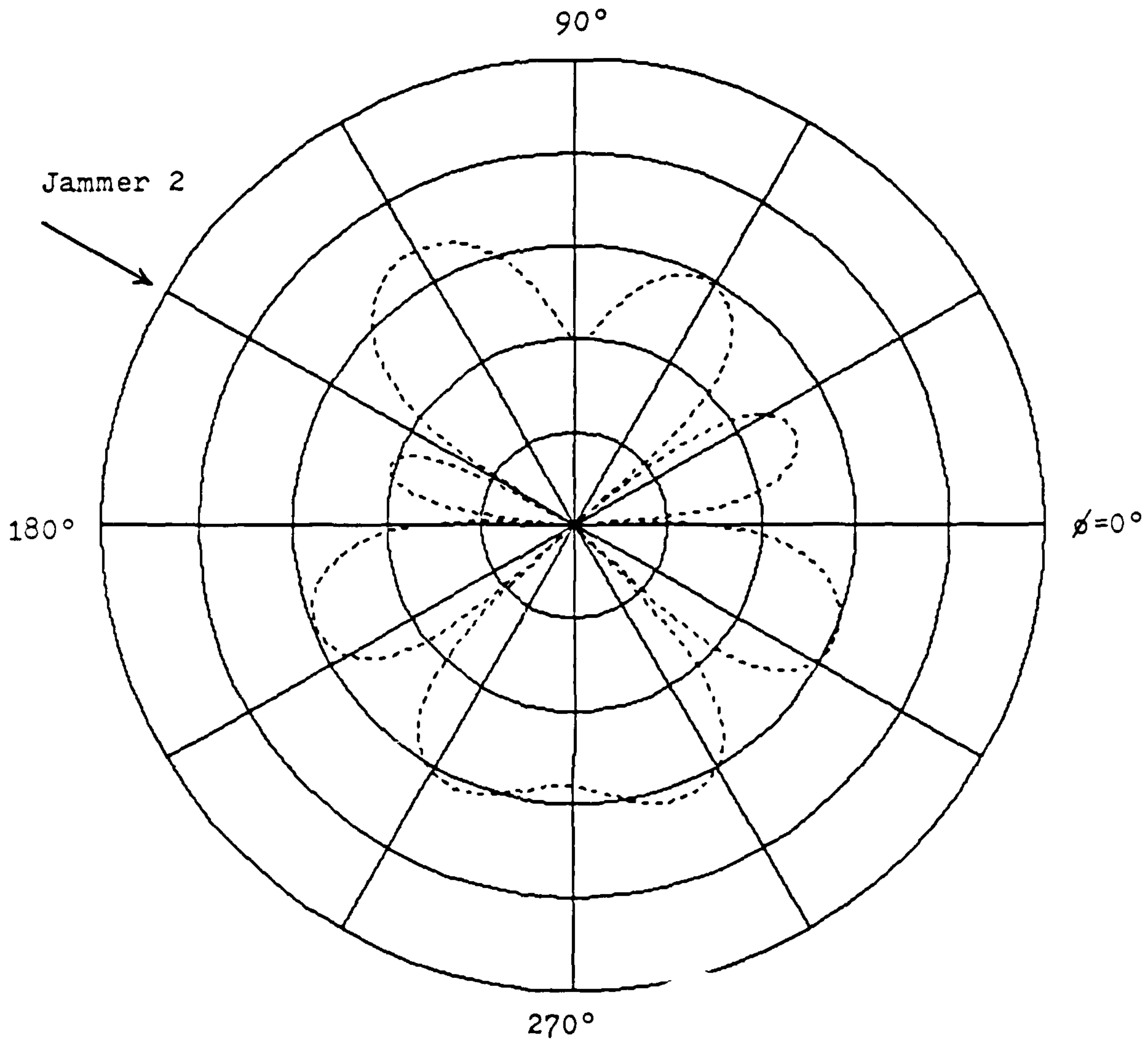


Figure T3.6 LMS-Adapted Pattern in Direction of Jammer 2

(Figure B)

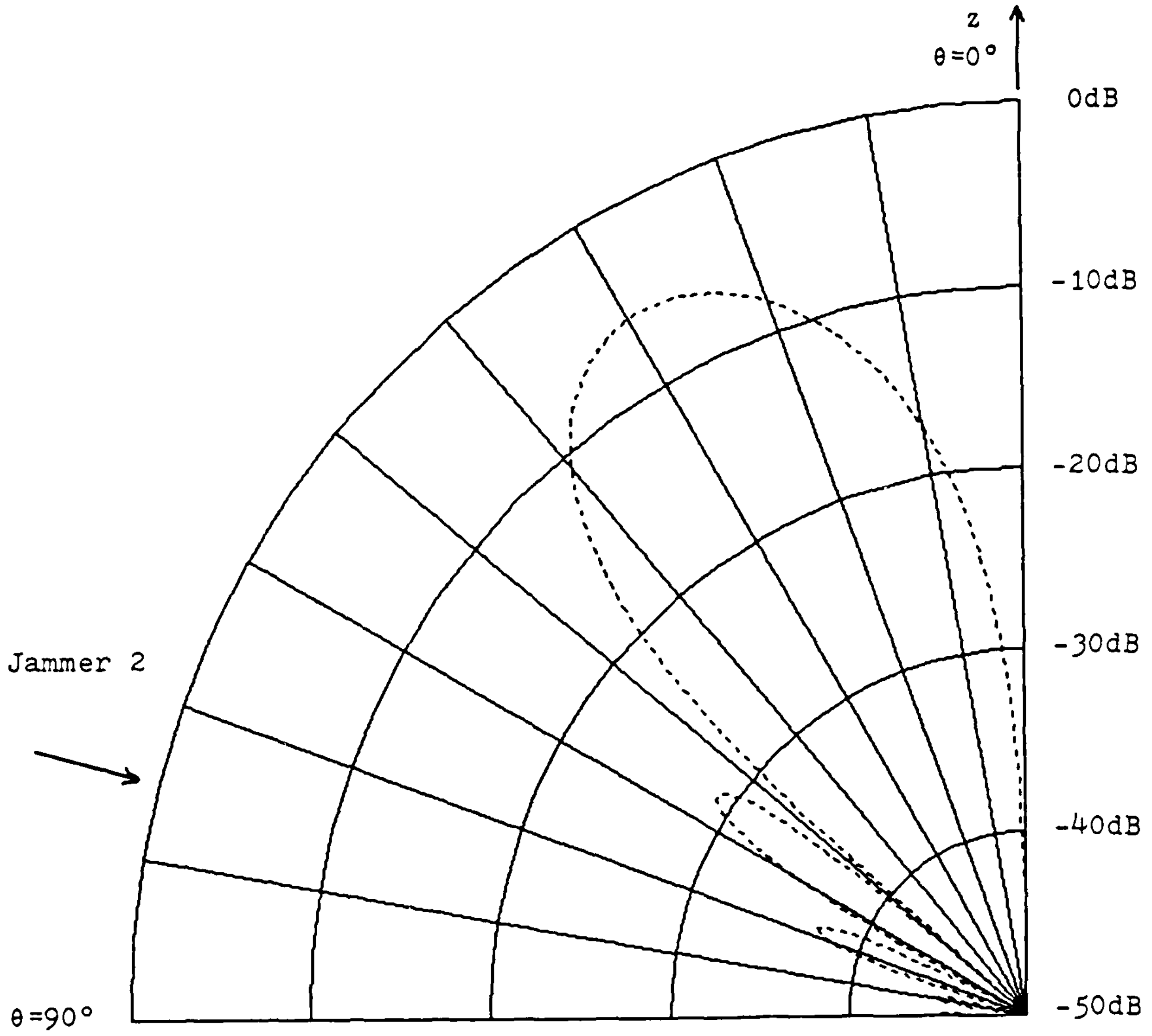


Figure T3.7 Constrained LMS-Adapted Pattern in Direction of Jammer 2

(Figure A)

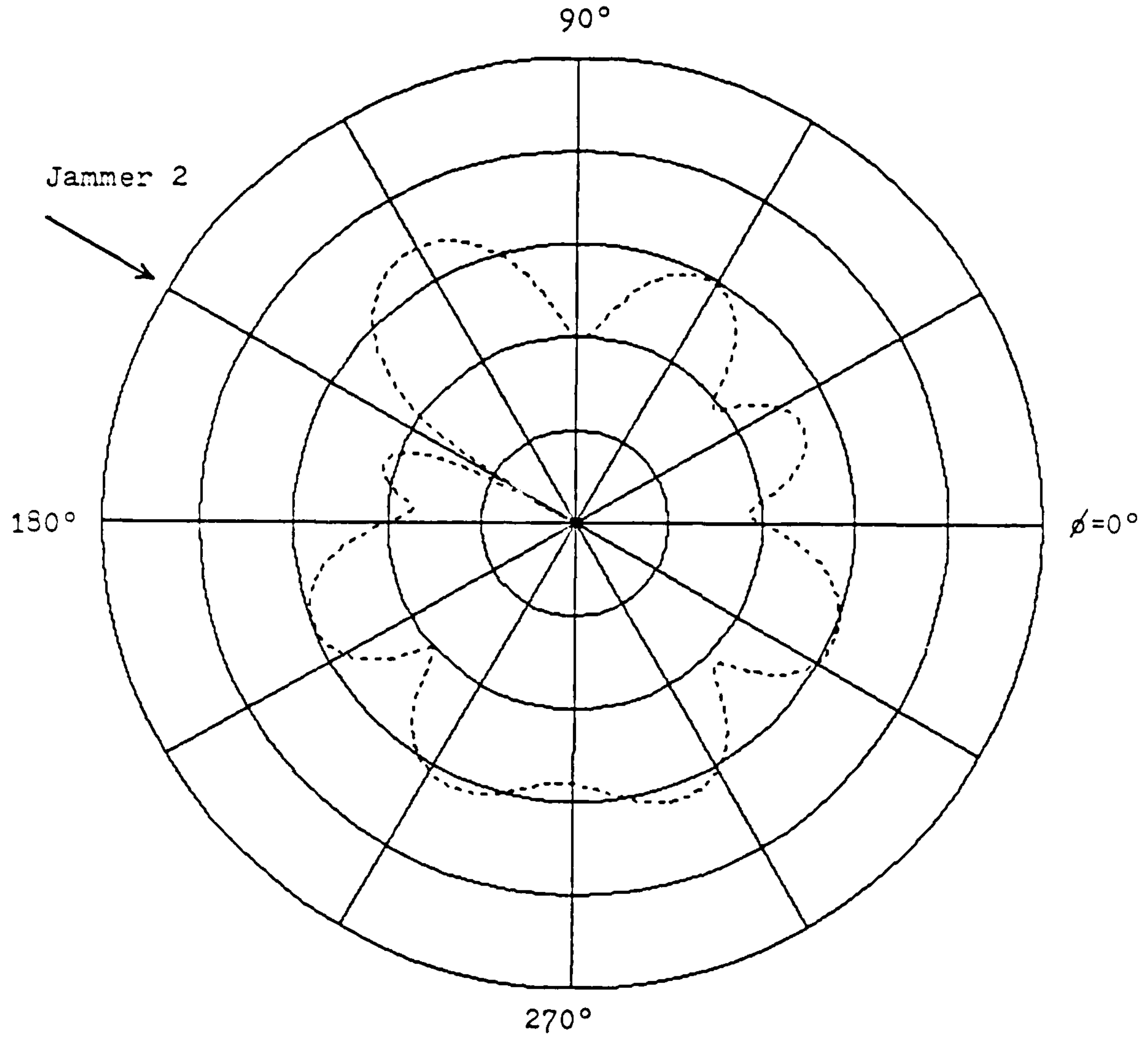


Figure T3.7 Constrained LMS-Adapted Pattern
in Direction of Jammer 2

(Figure B)

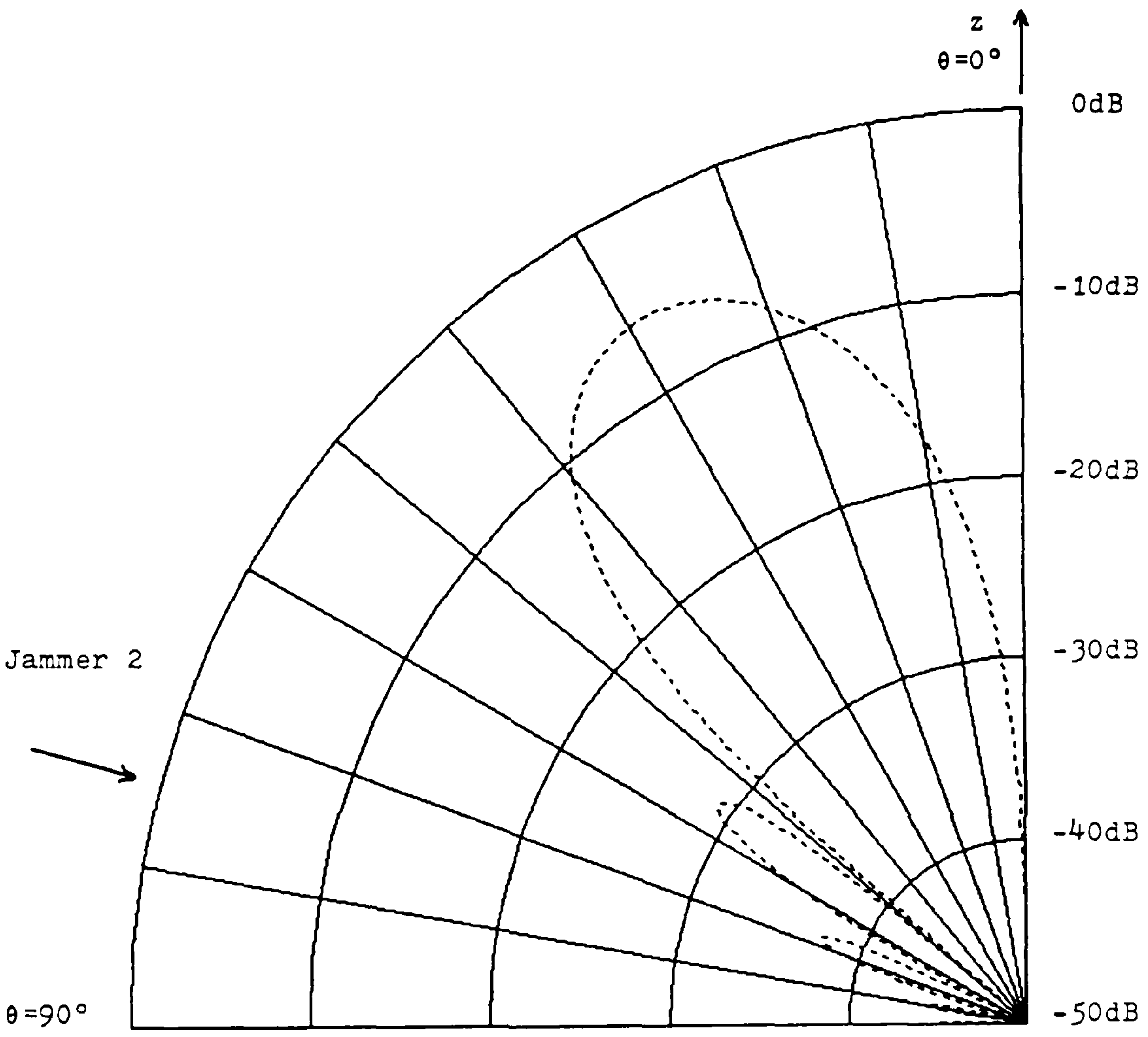


Figure T3.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure A)

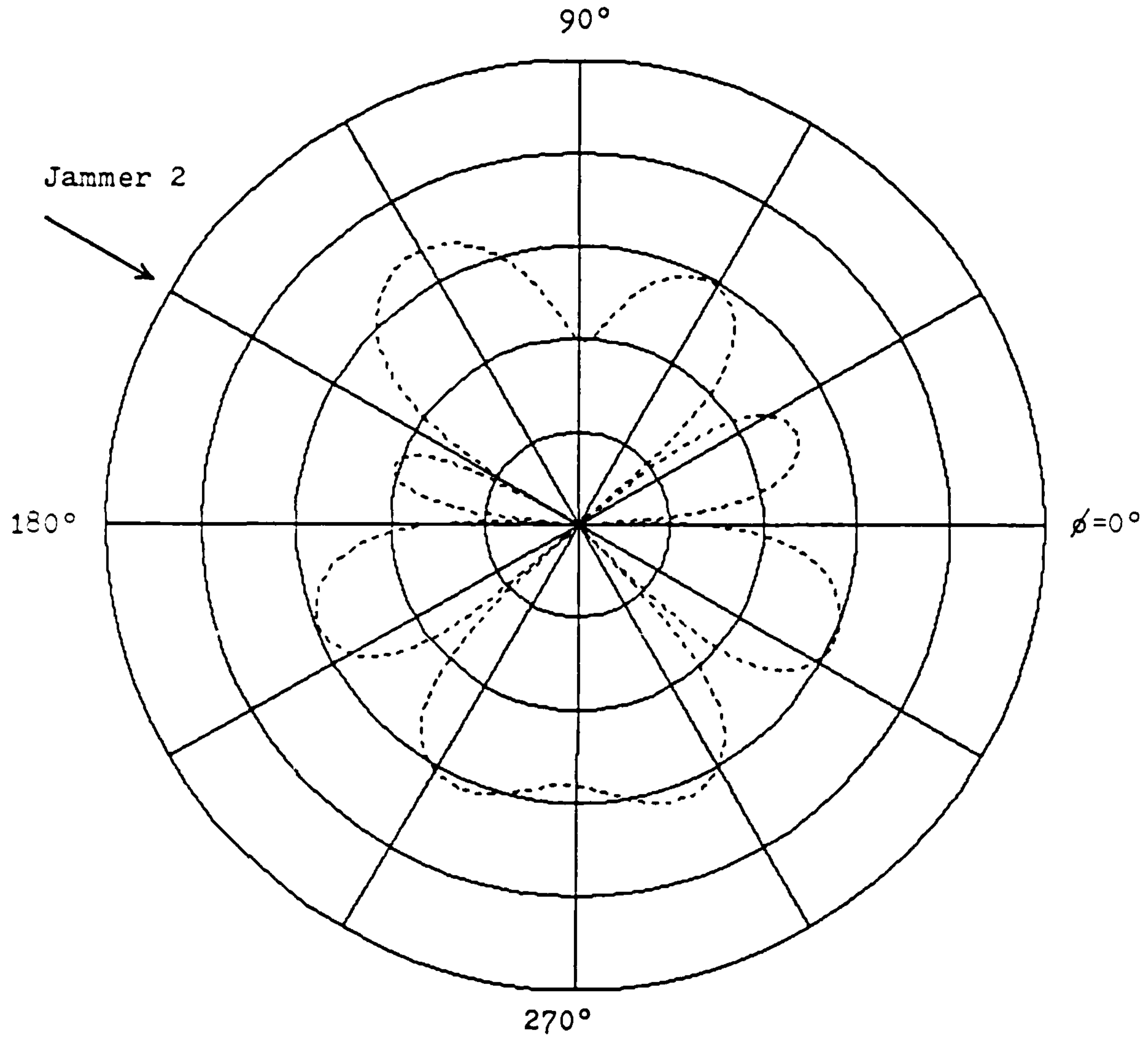


Figure T3.8 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure B)

7.4 TEST4: Dependence of Convergence of Constrained LMS Algorithm on the Presence of the Signal

It has been mentioned that the Constrained LMS algorithm will perform better when the signal is absent than when it is present. In other words, if the Constrained LMS algorithm was to be selected, sending a preamble would not only be wasteful, it would be detrimental. Tests 1, 2 and 3 will now be repeated for the Constrained LMS with the "desired" signal (preamble) being absent.

In order to perform this test, the simulation had to be effectively "fooled." If the signal power is zero, the SNR is undefined. Therefore, the optimum values will be calculated as if the signal was present (signal power = 1). In other words, the Constrained LMS algorithm will have to attain the same value of SNR as it did before.

The results are tabulated in Table 7.4. The polar antenna plots that follow verify that the Constrained LMS algorithm places deep nulls in the directions of the jammers without the benefit of any preamble whatsoever. The number of average iterations required also validates the assumption that the algorithm will perform better in the absence of the signal.

Notice that the average number of required iterations is now virtually identical to that of the LMS algorithm. Also recall that the Constrained LMS algorithm (with the signal present) suffered large fluctuations in required iterations for channels 2 and 3. When the signal is absent, however, the algorithm actually had the smallest percentage deviation of all the algorithms tested. This is graphically illustrated by the histograms of Figures 7.13 - 7.15. This fact only strengthens the claim that the existence of the three paths (and the signal magnification that results) is the primary culprit of the convergence problems suffered by the Constrained LMS algorithm. When this effect was nullified by eliminating the signal, convergence was fairly rapid and quite consistent.

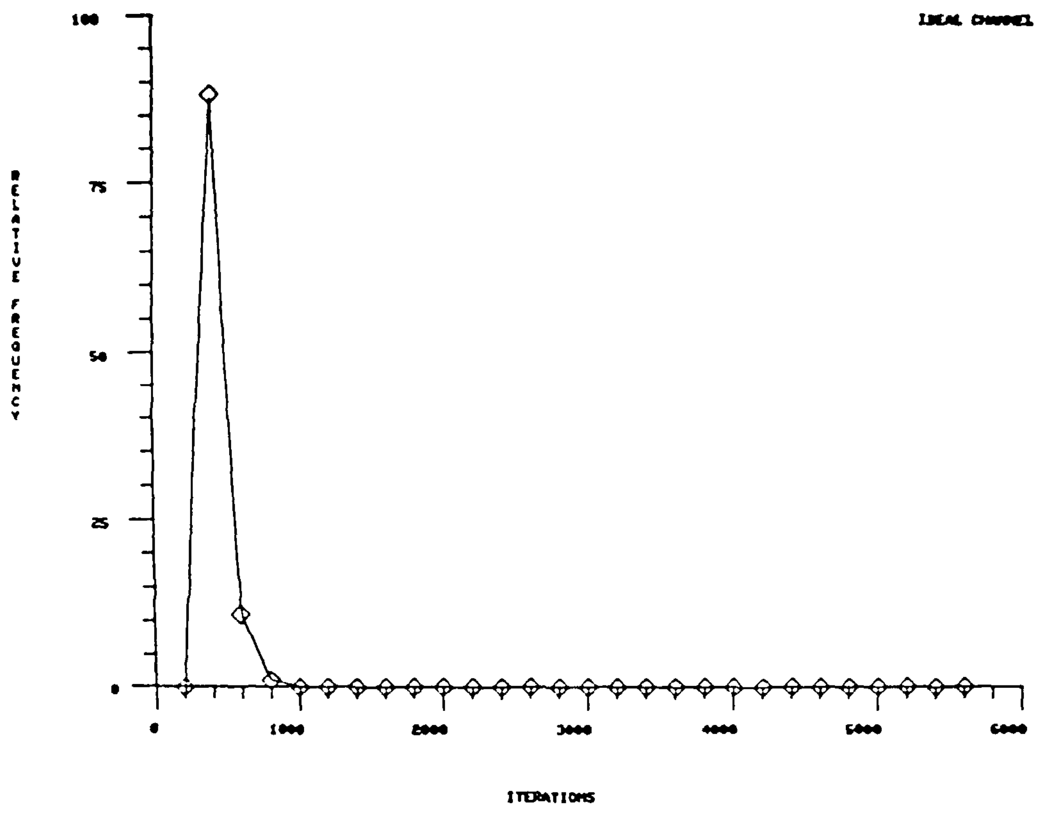


Figure 7.13 Convergence Histogram of Constrained LMS Algorithm in Ideal Channel (no signal present)

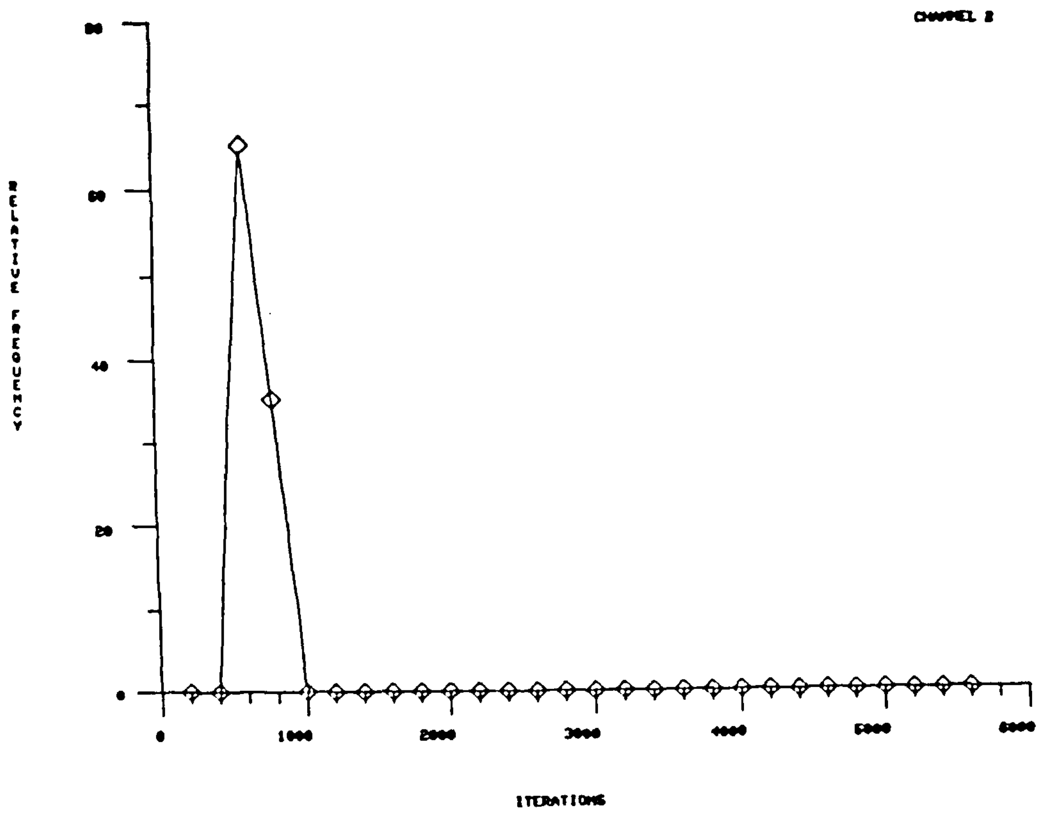


Figure 7.14 Converger.:e Histogram of Constrained LMS Algorithm in Channel 2 (no signal present)

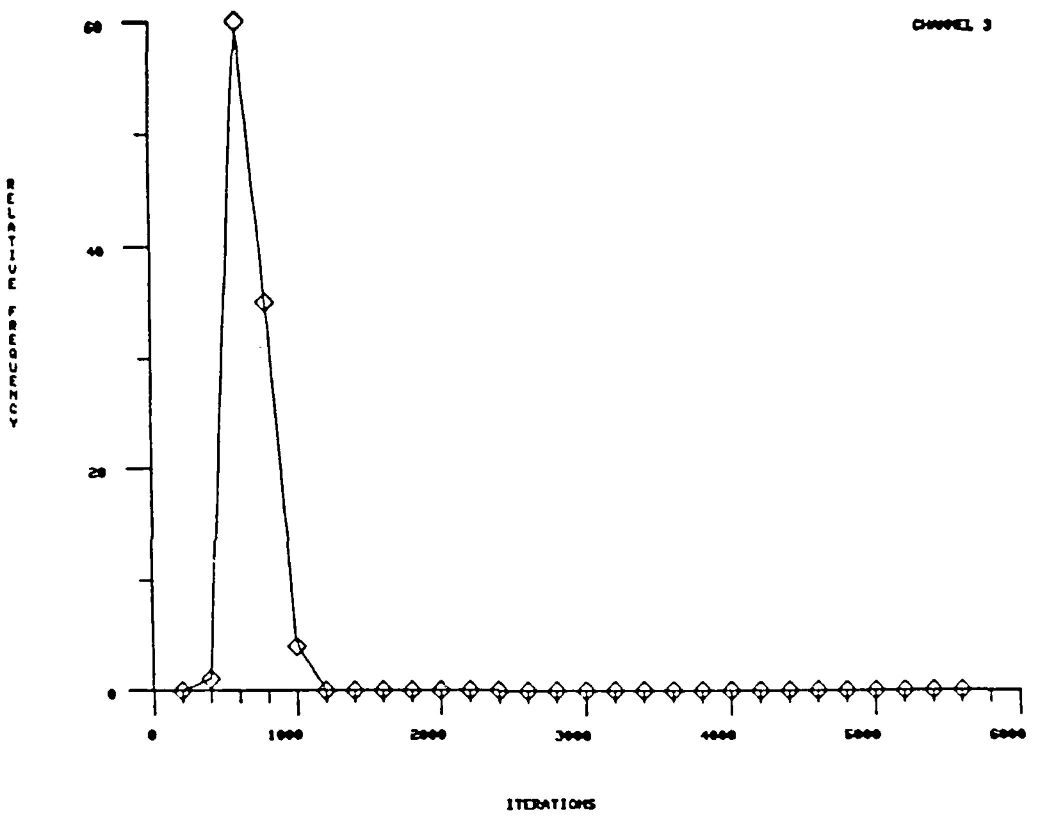
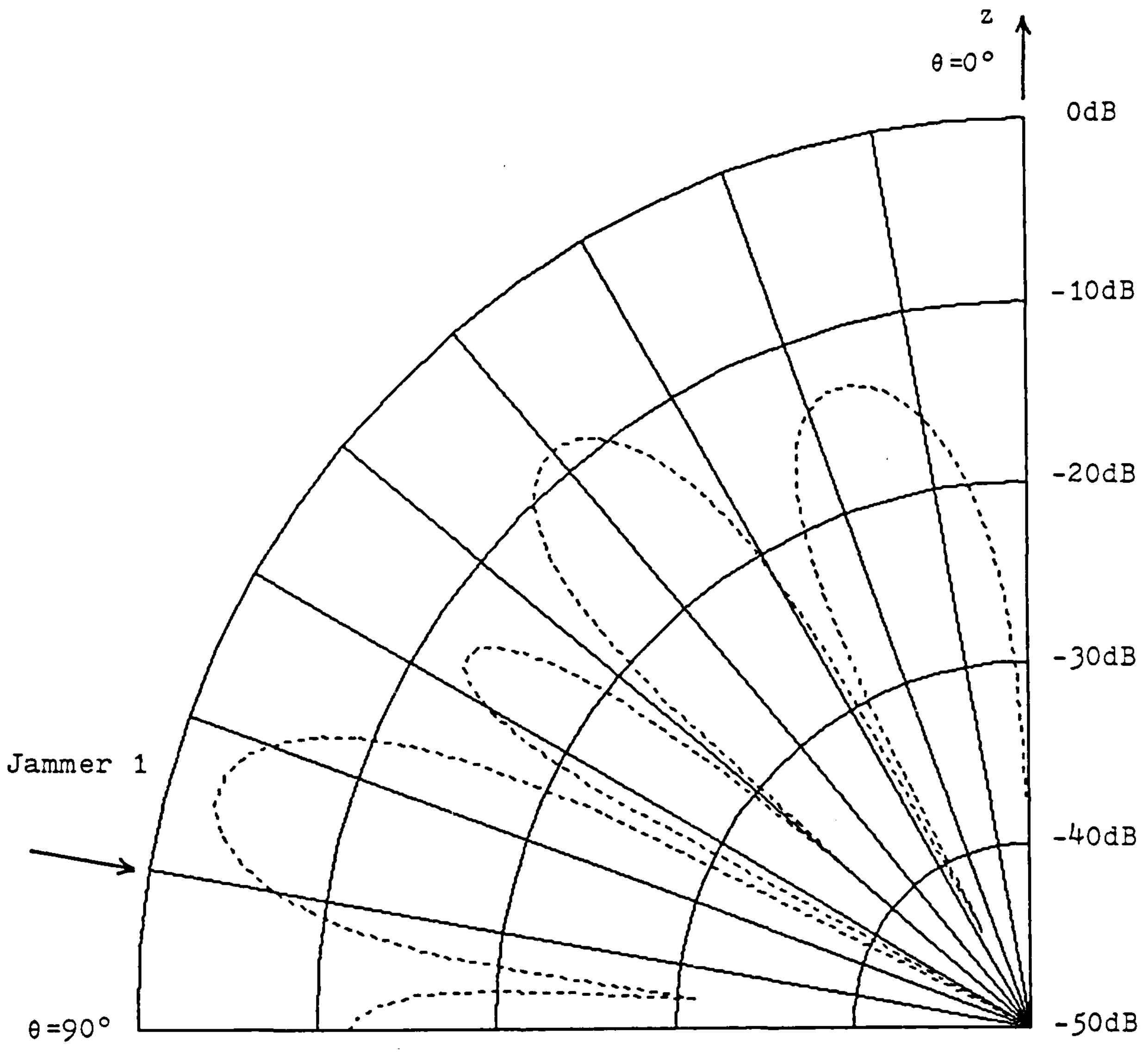


Figure 7.15 Convergence Histogram of Constrained LMS Algorithm in Channel 3 (no signal present)

Table 7.4 TEST4 SUMMARY

CHANNEL	NUMBER OF CONVERGENCES	AVERAGE NUMBER OF ITERATIONS	STANDARD DEVIATION
IDEAL	100	337.0	81.1
CHANNEL 2	100	593.0	88.1
CHANNEL 3	100	598.5	105.7

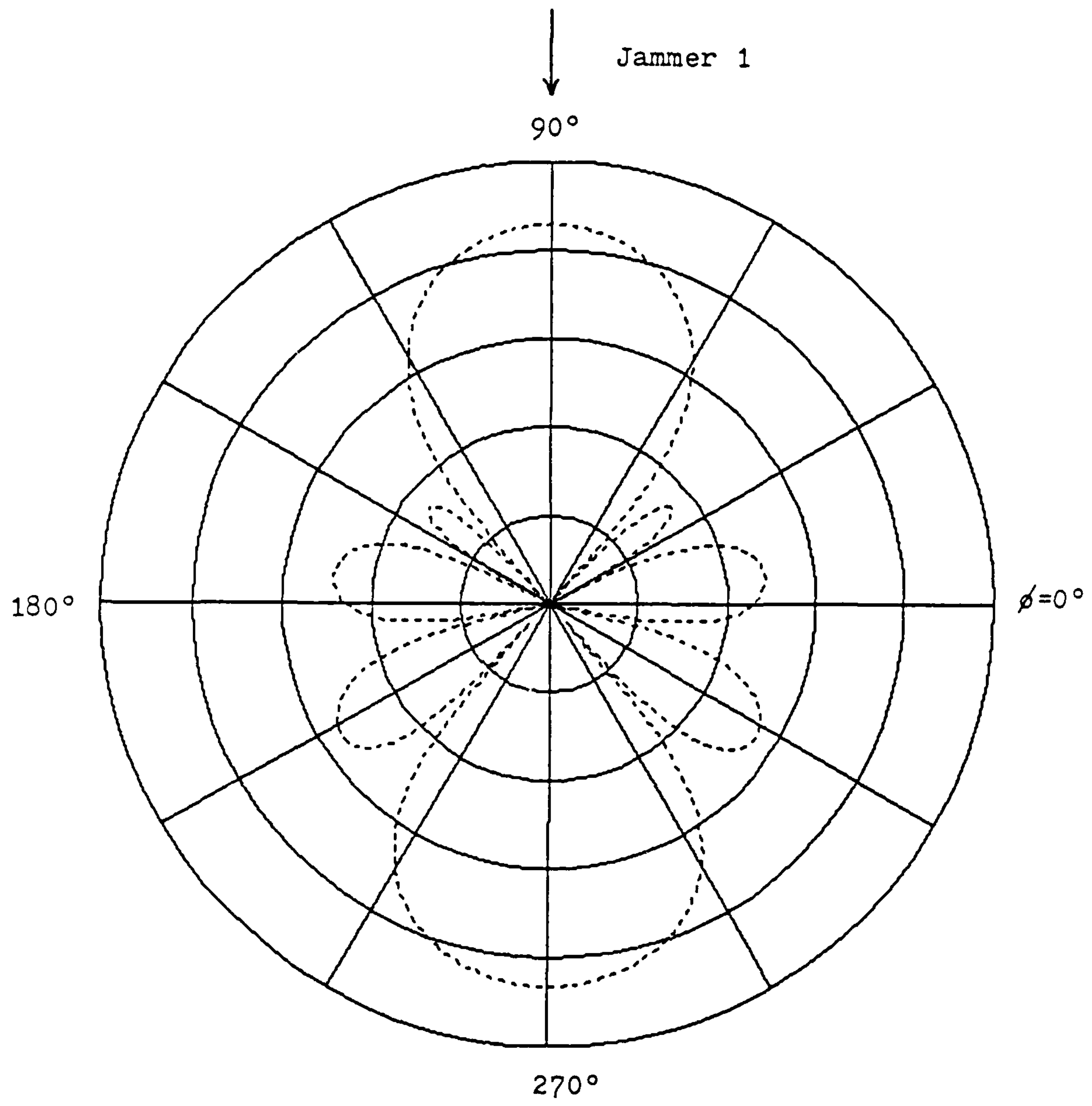


Elevation plot at 90° azimuth

Arrow indicates arrival angle of Jammer 1 (θ=80°)

Figure T4.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure A)



Azimuthal plot at 80° elevation

Arrow indicates arrival angle of Jammer 1 (ϕ=90°)

Figure T4.1 Unadapted Pattern in Direction of Jammer 1

(Figure B)

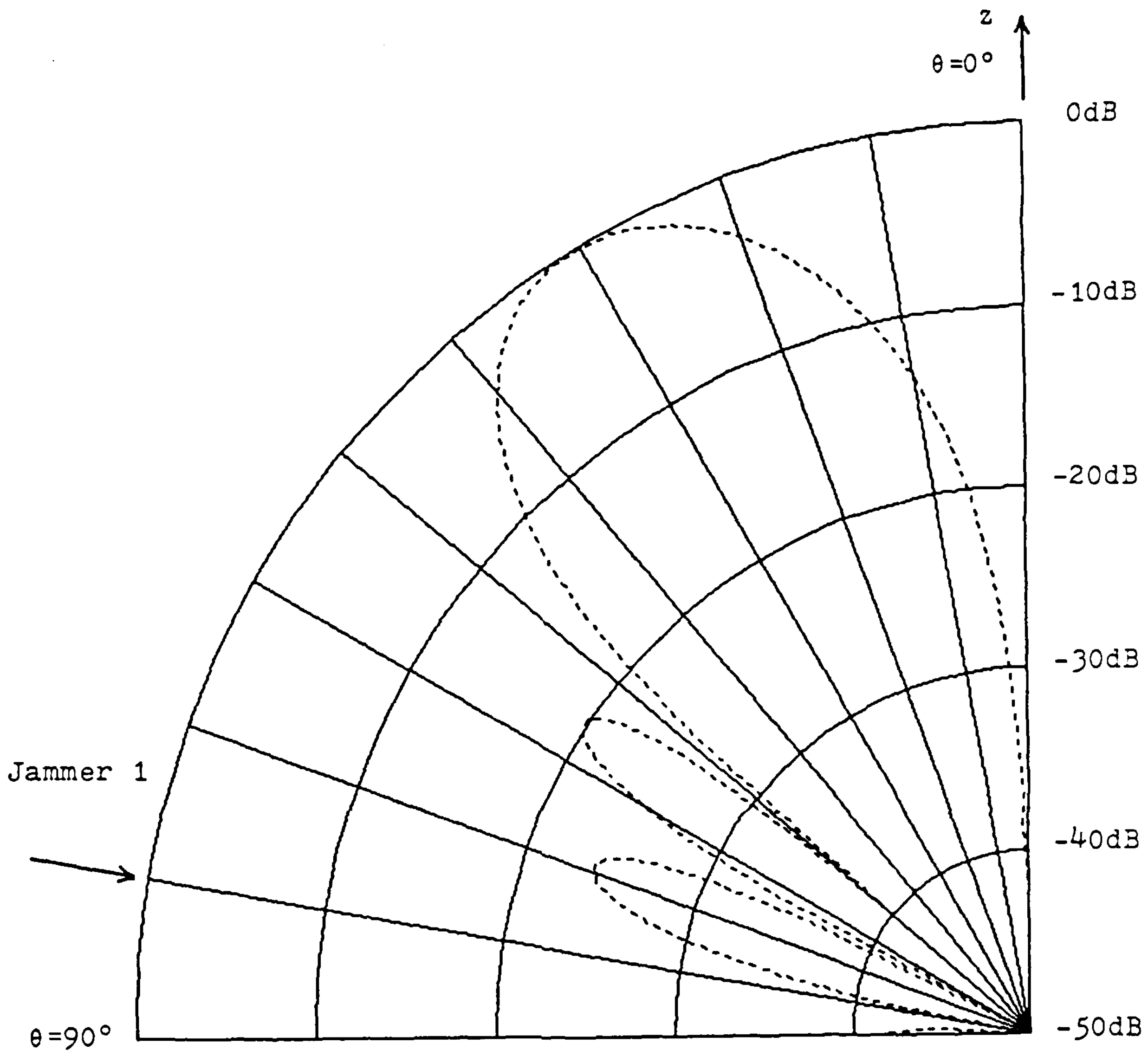


Figure T4.2 Adapted Pattern in Direction of Jammer 1
(Figure A)

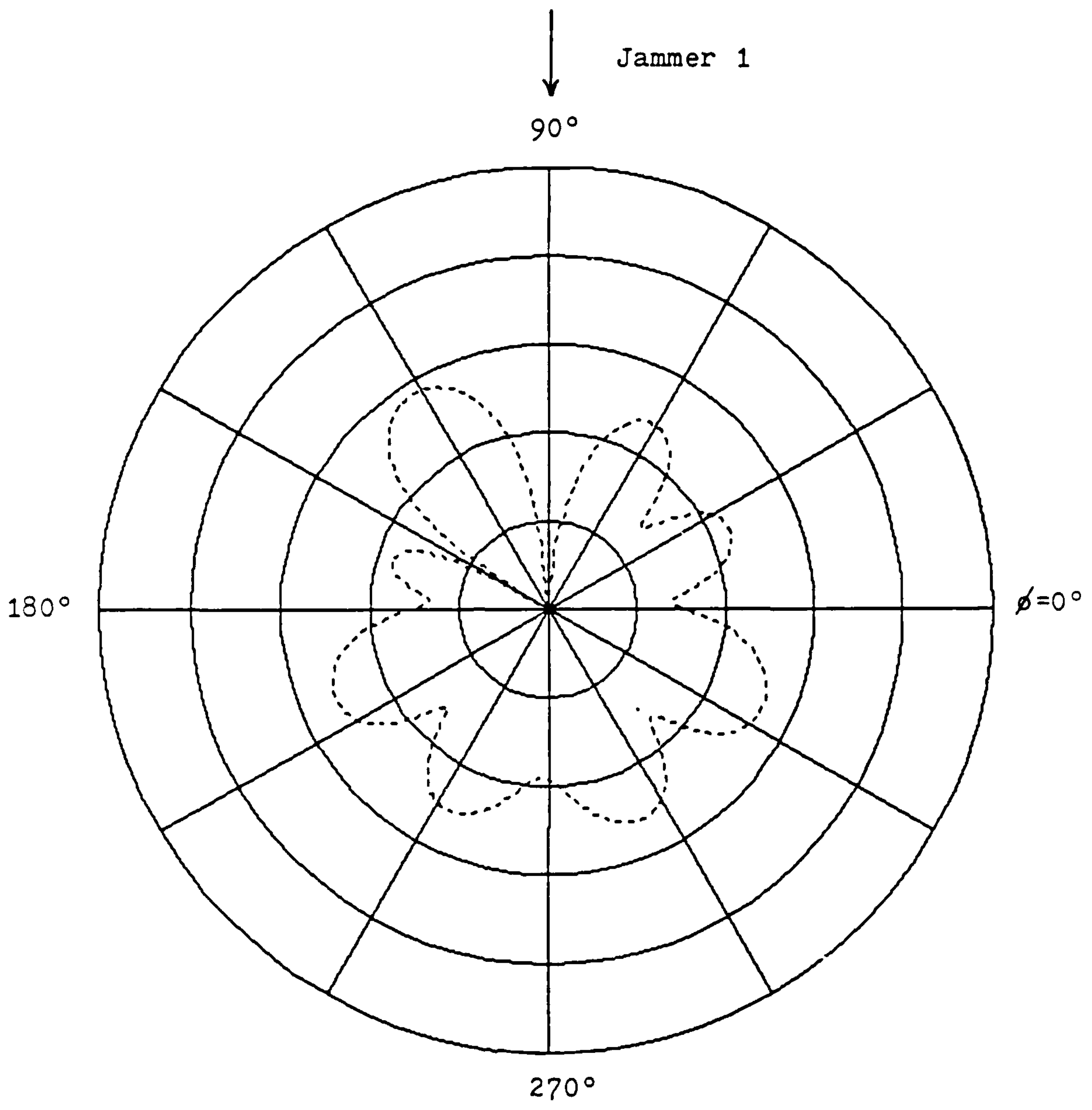
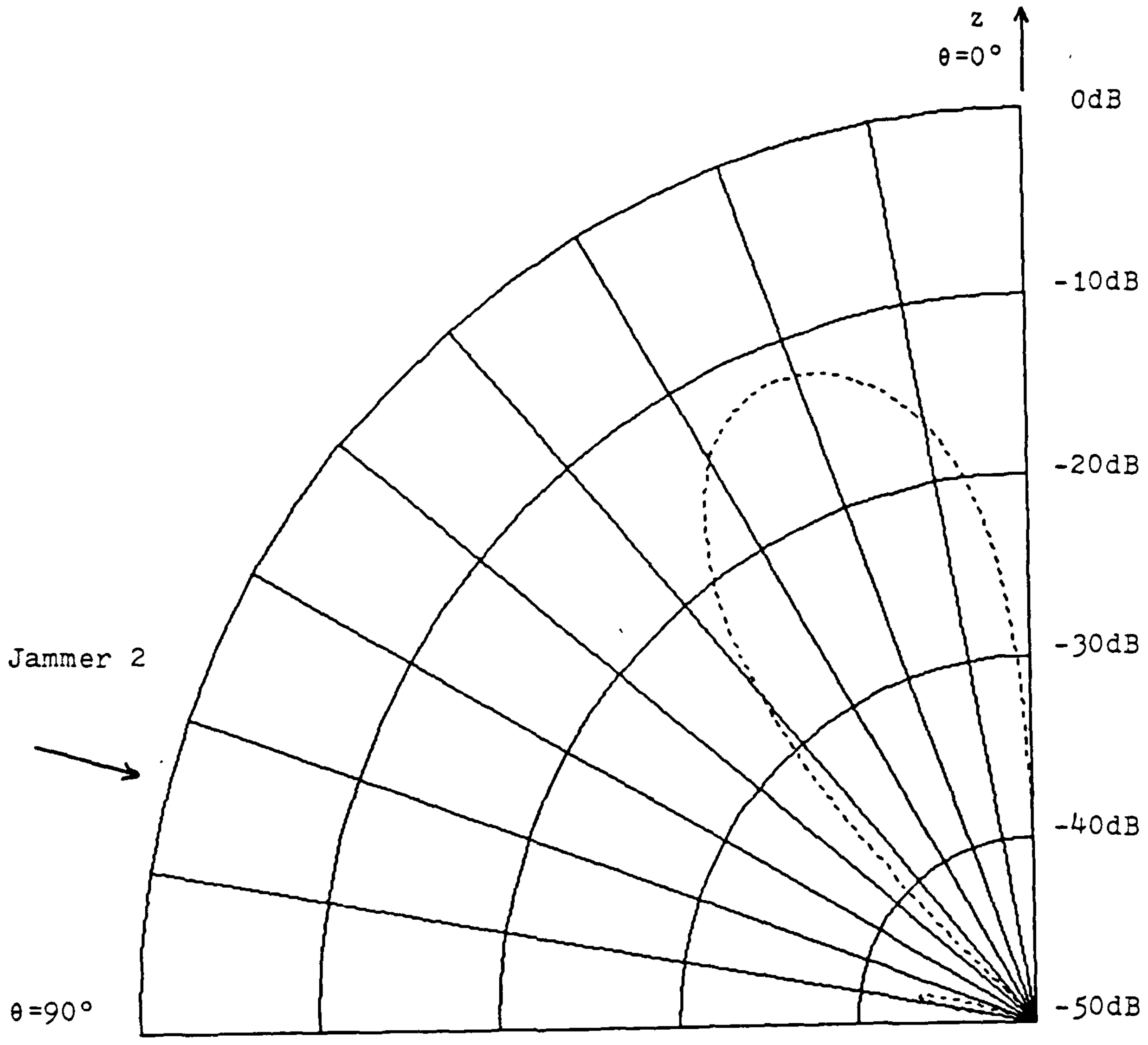


Figure T4.2 Adapted Pattern in Direction of Jammer 1

(Figure B)

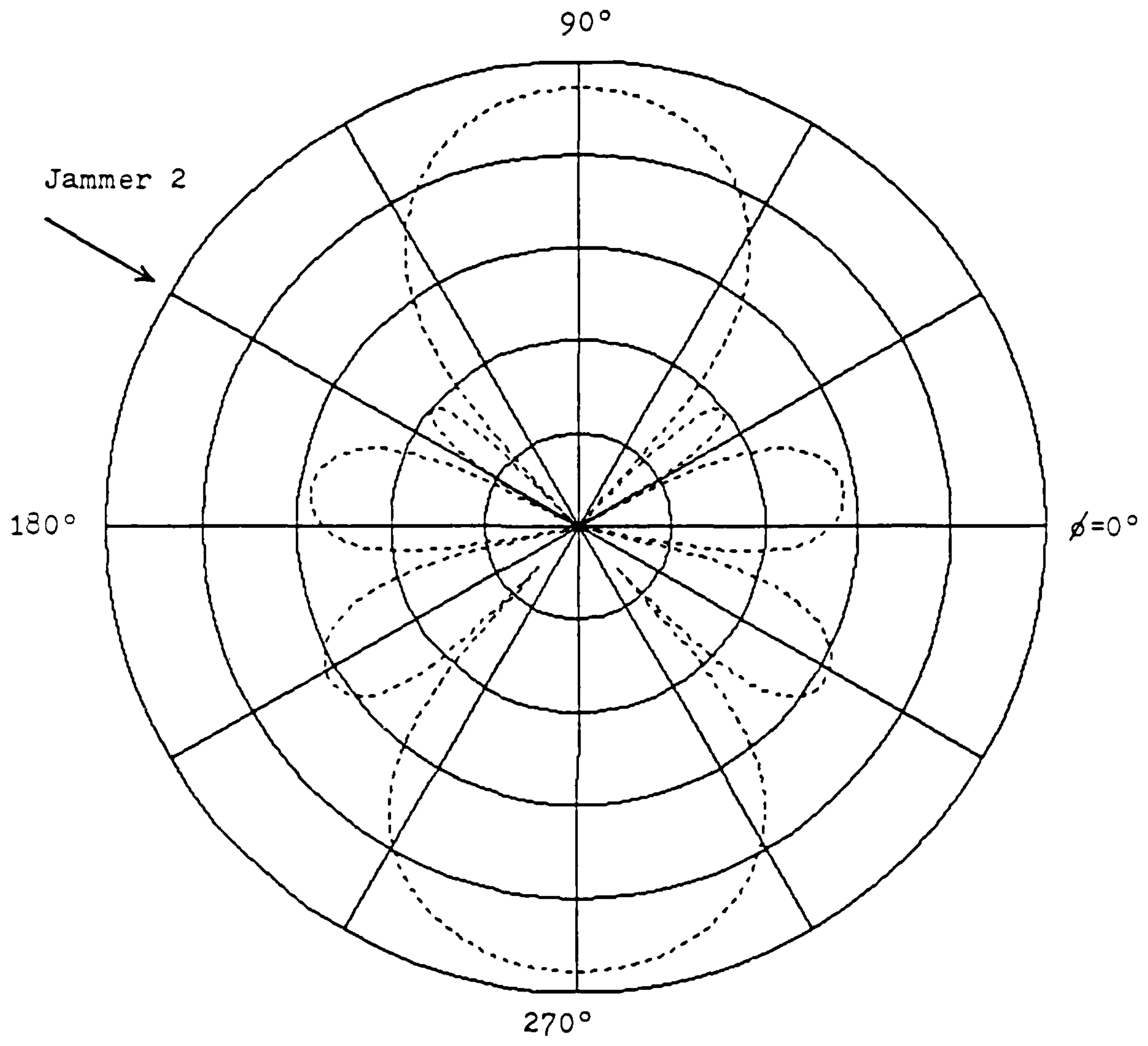


Elevation plot at 150° azimuth

Arrow indicates arrival angle of Jammer 2 (θ=75°)

Figure T4.3 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure A)



Azimuthal plot at 75° elevation

Arrow indicates arrival angle of Jammer 2 (ø=150°)

Figure T4.3 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure B)

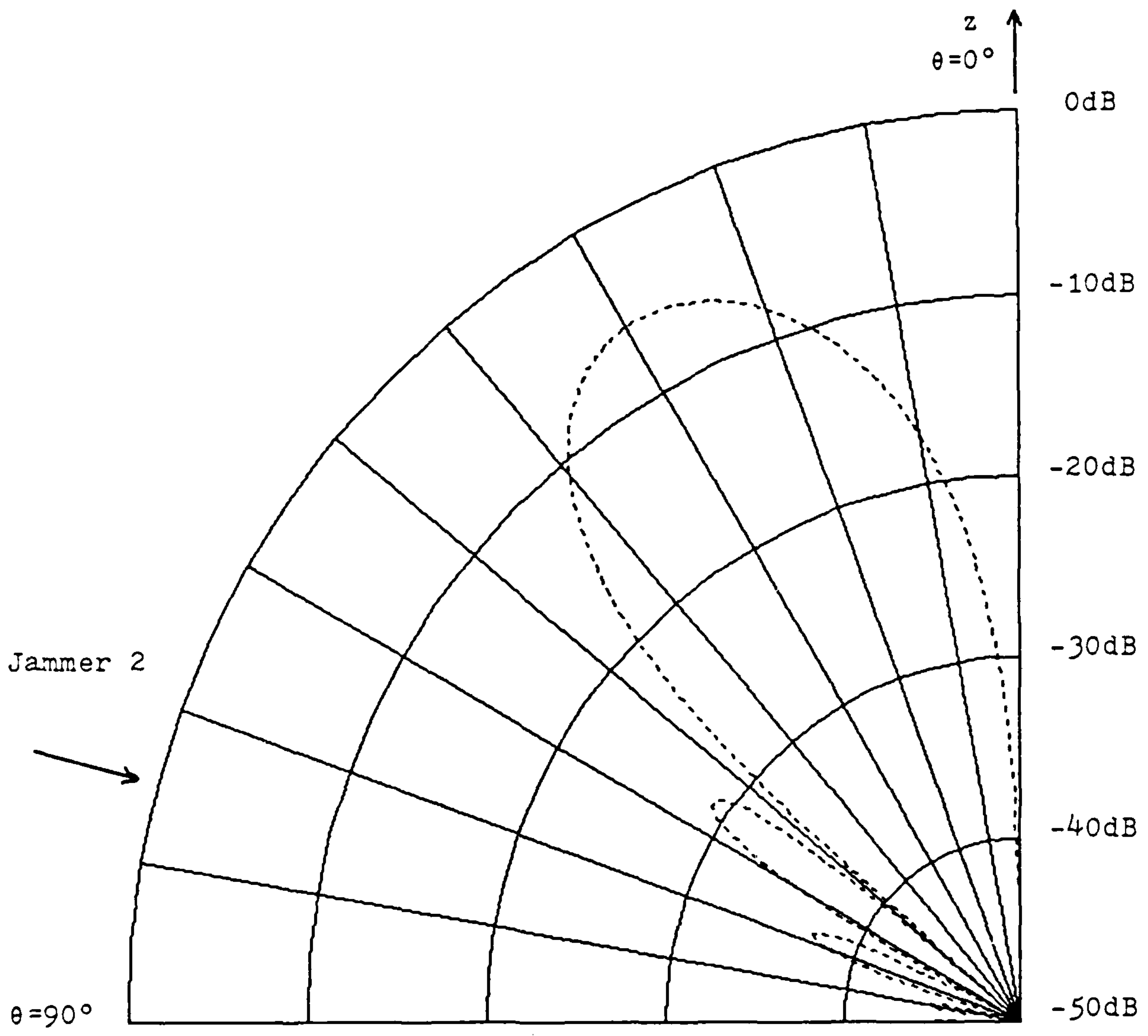


Figure T4.4 Adapted Pattern in Direction of Jammer 2

(Figure A)

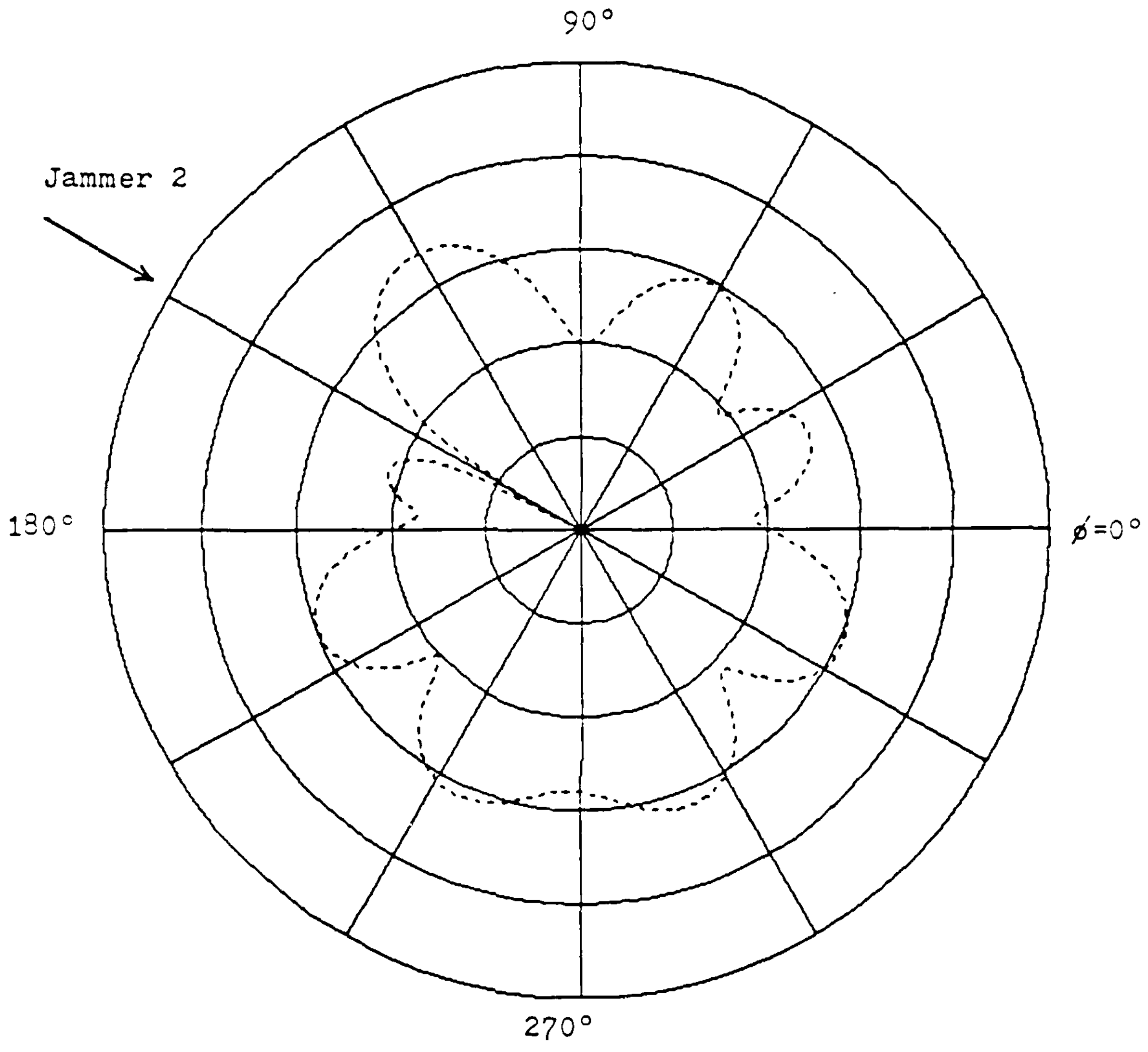


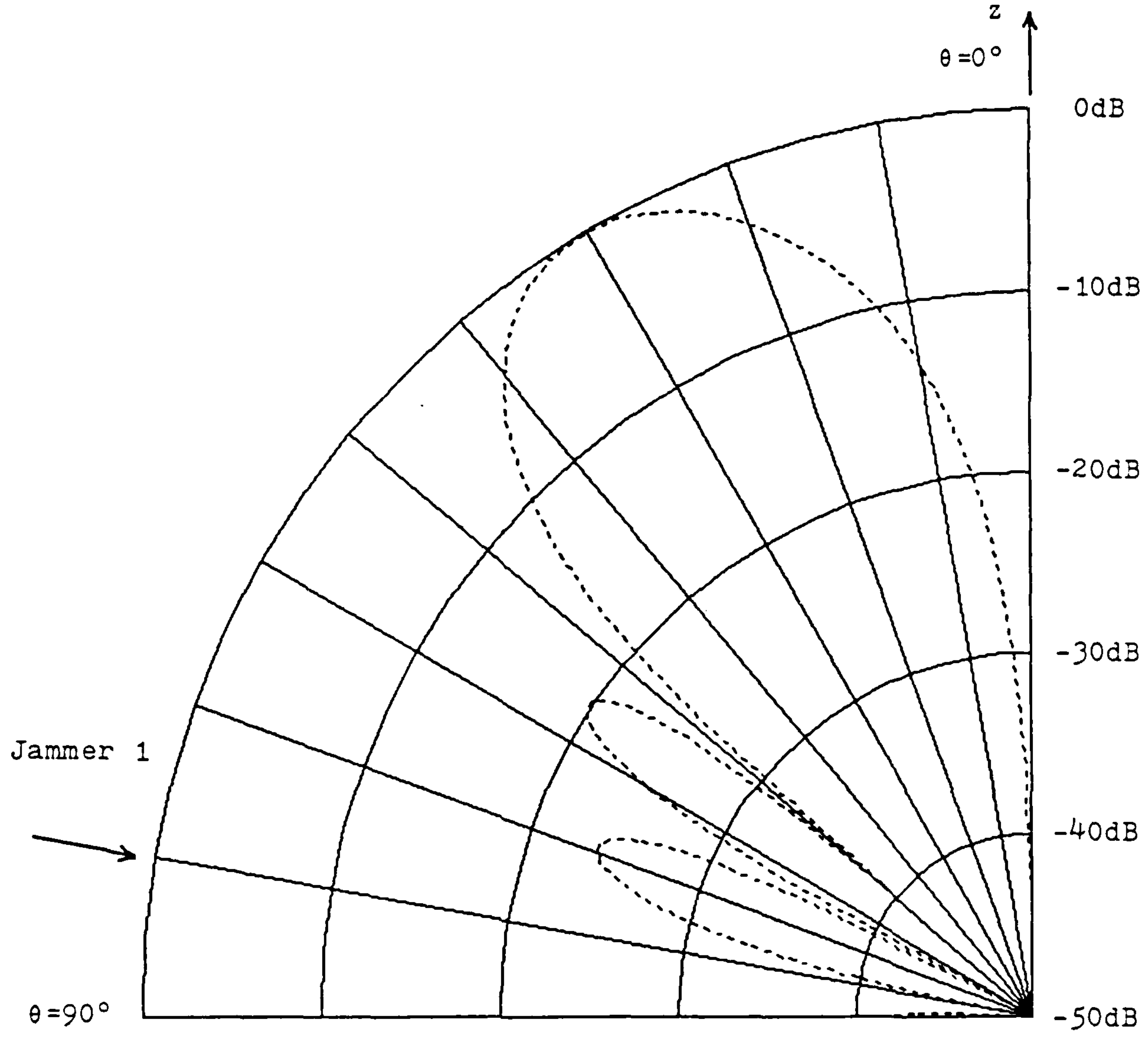
Figure T4.4 Adapted Pattern in Direction of Jammer 2

(Figure B)

7.5 TEST5: Dependence of Adapted Antenna Pattern on Number of Required Iterations

The histograms that have been presented graphically demonstrate that quite often there is a considerable difference in the number of required iterations for an algorithm to converge. The question may arise whether or not the adapted antenna patterns produced by widely separated convergences are significantly different.

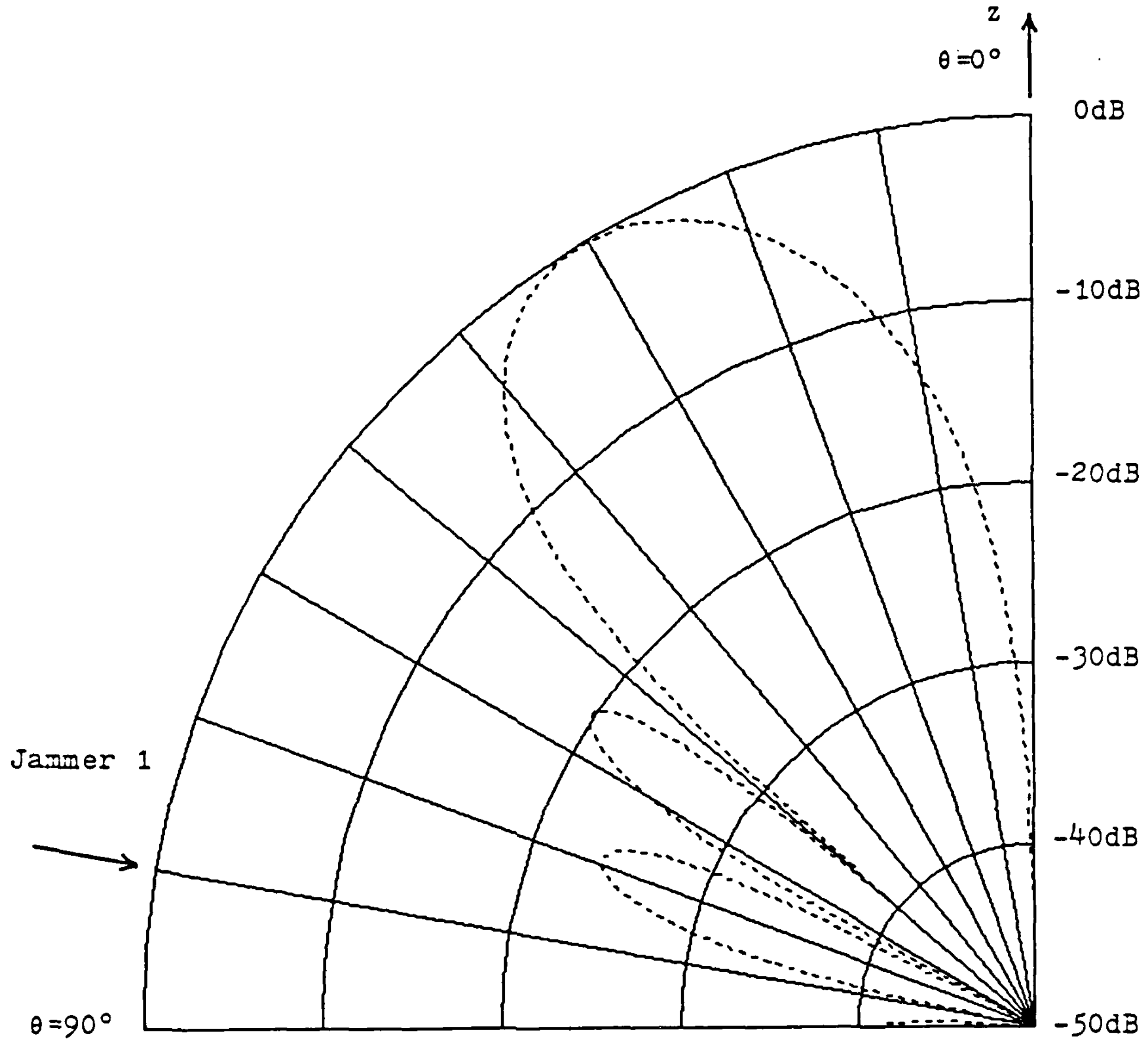
The polar antenna plots that follow were generated using the fastest and slowest convergences of the LMS algorithm of TEST2. It can be seen the antenna plots are virtually indistinguishable. This indicates that there is not a marked difference in the final adapted antenna patterns produced by the widely separated convergences. This is certainly not a surprising result, as the performance measure that has been utilized requires that the pattern be close to optimal before the algorithm is said to be converged.



Fastest Convergence

Figure T5.1 Adapted Elevation Patterns at 90° Azimuth

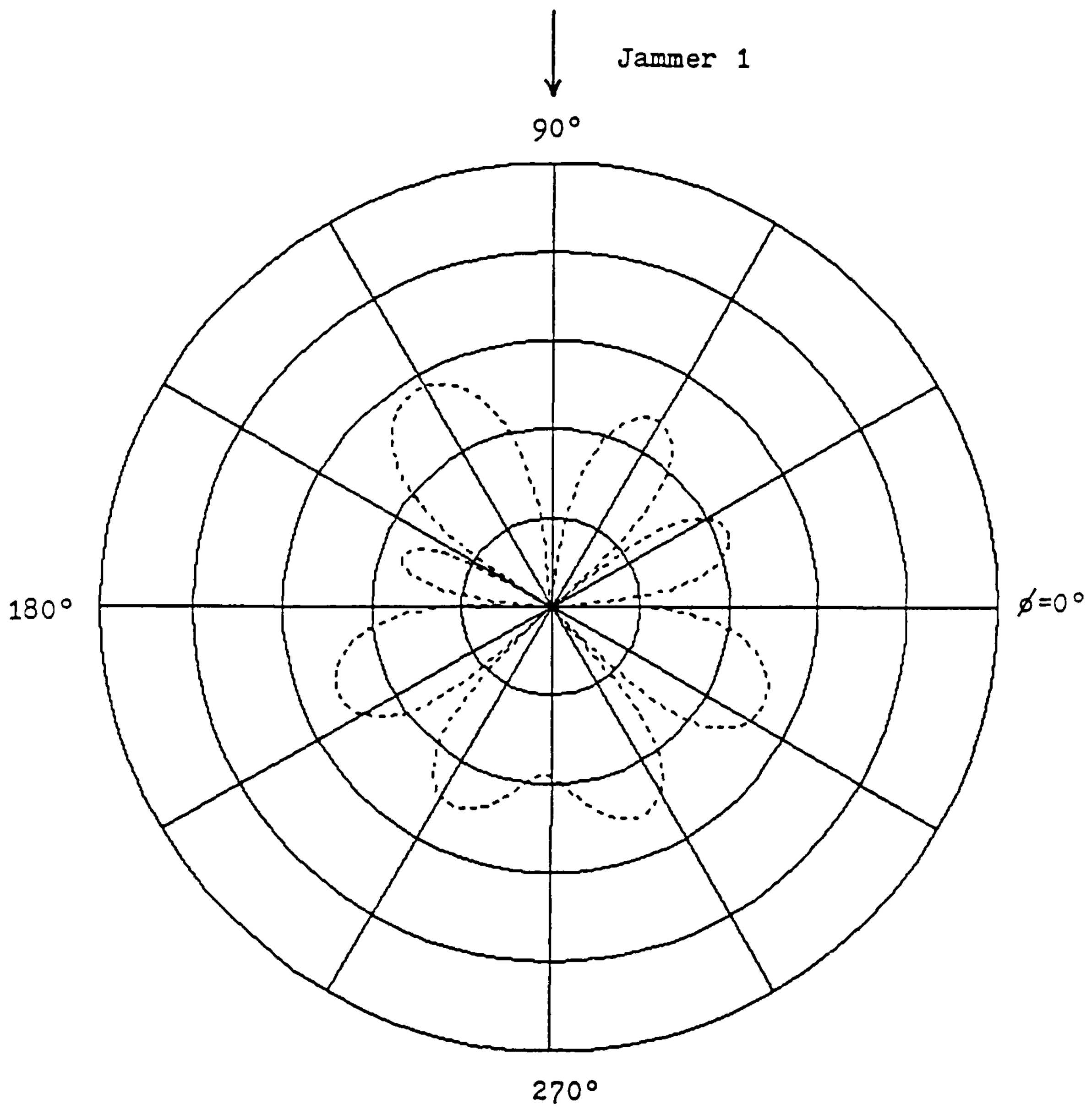
(Figure A)



Slowest Convergence

Figure T5.1 Adapted Elevation Patterns at 90° Azimuth

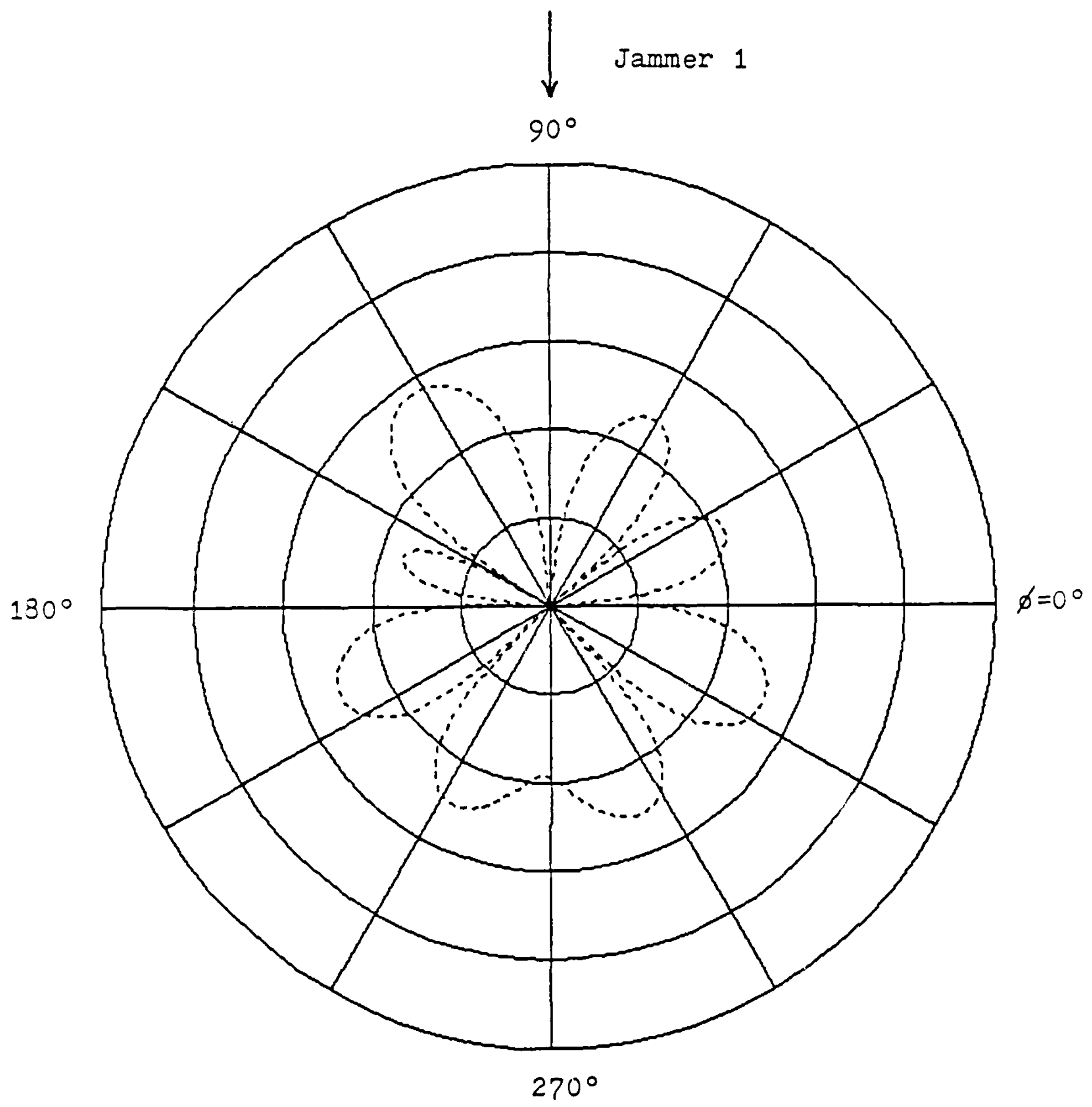
(Figure B)



Fastest Convergence

Figure T5.2 Adapted Azimuthal Patterns at 80° Elevation

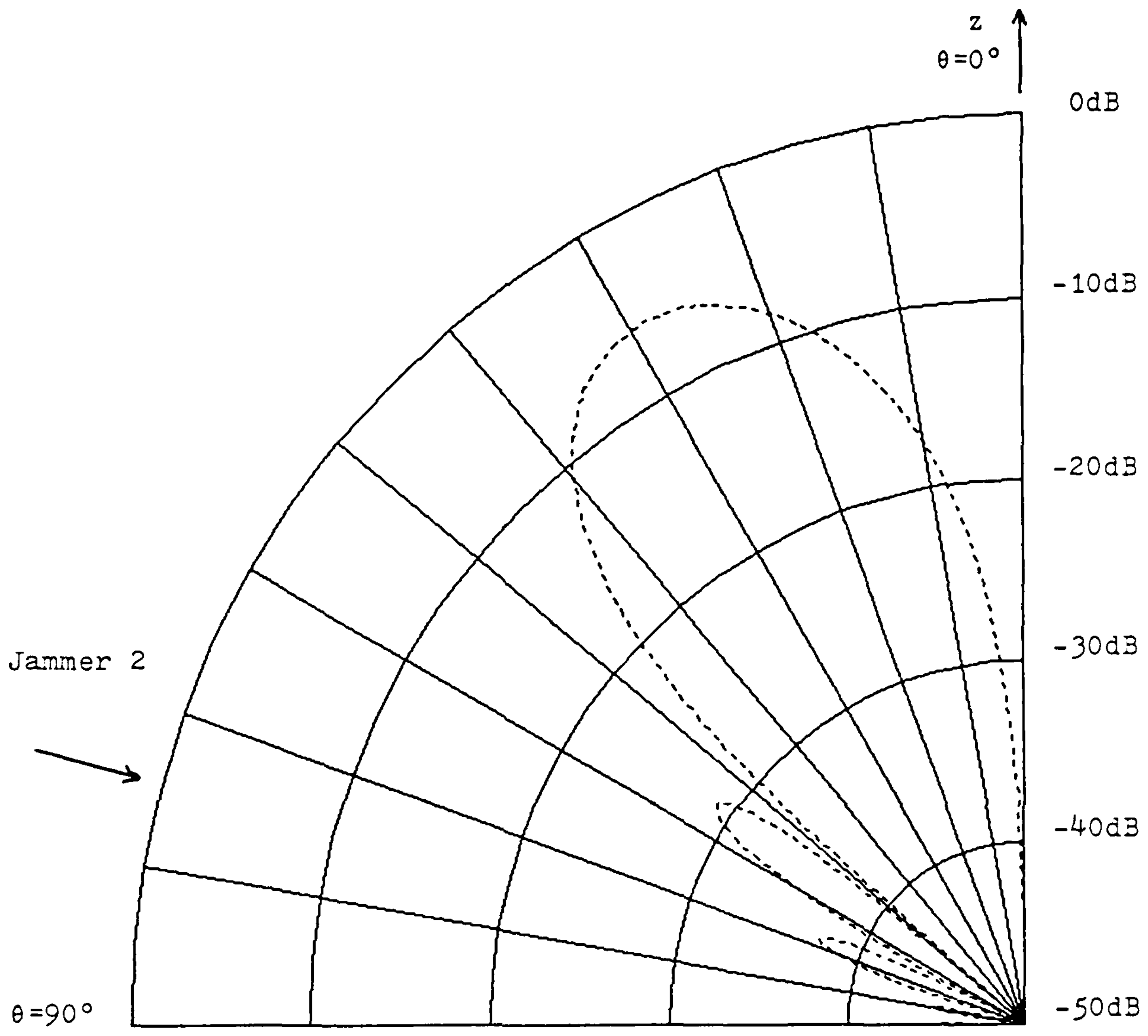
(Figure A)



Slowest Convergence

Figure T5.2 Adapted Azimuthal Patterns at 80° Elevation

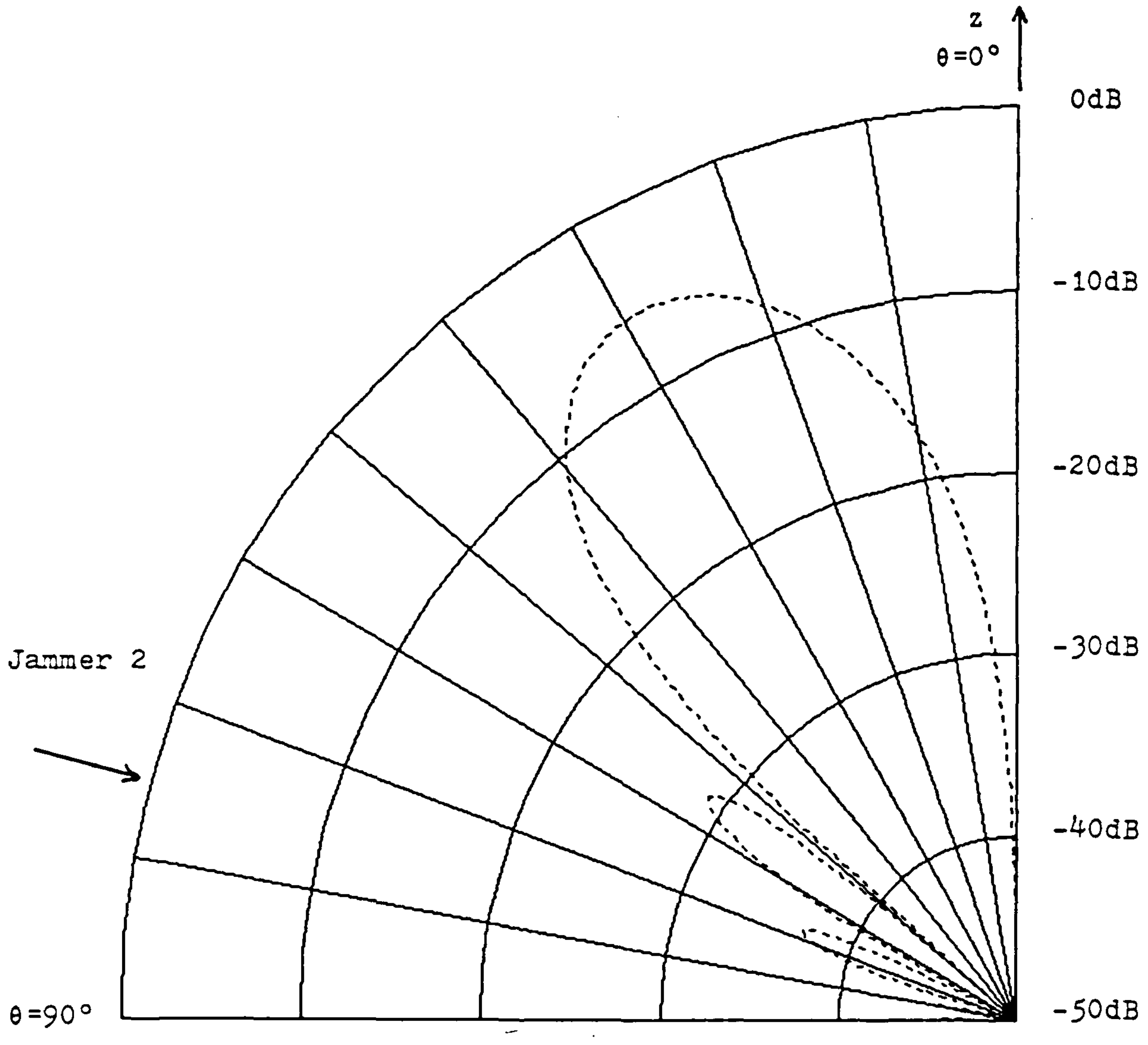
(Figure B)



Fastest Convergence

Figure T5.3 Adapted Elevation Patterns at 150° Azimuth

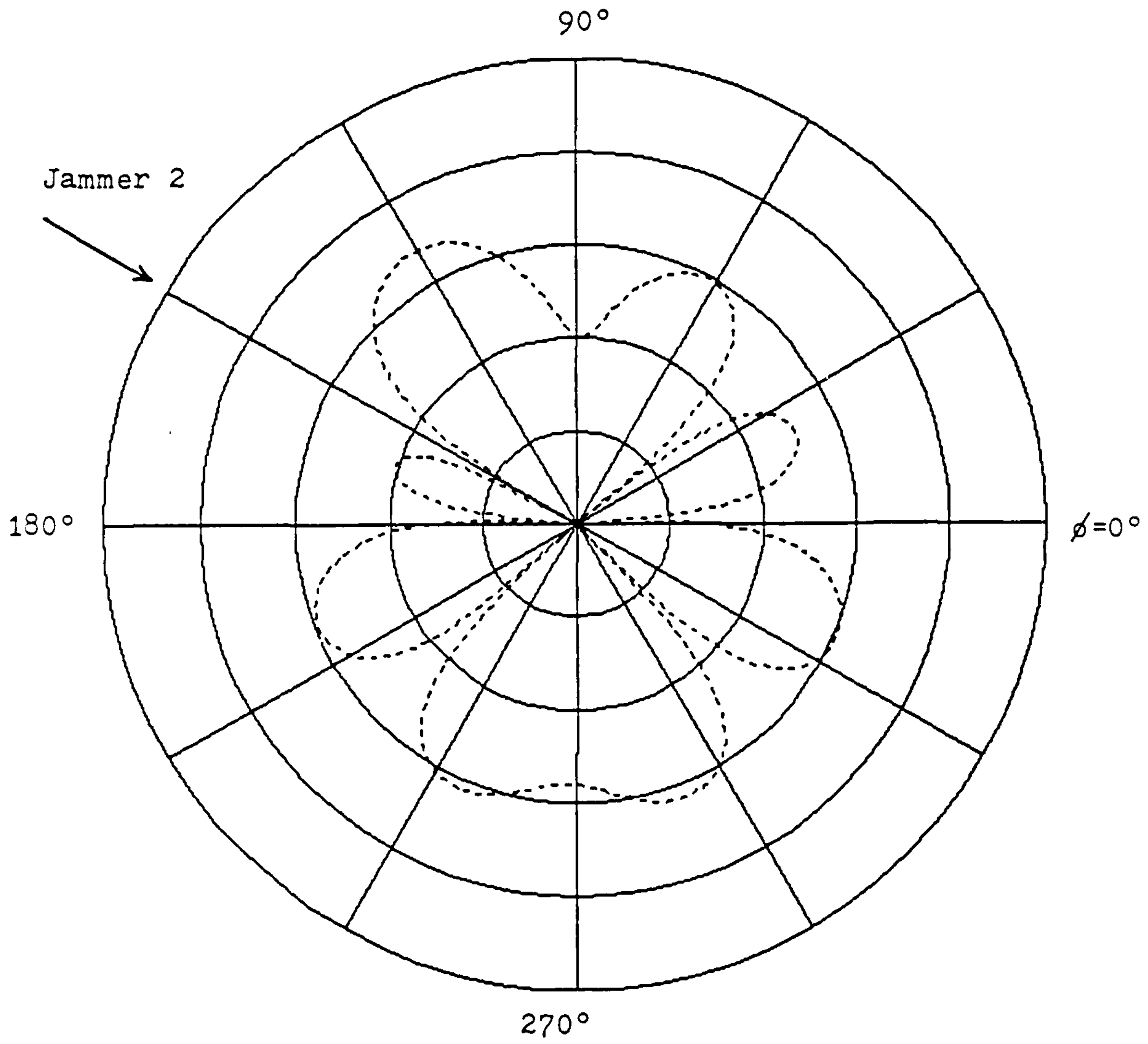
(Figure A)



Slowest Convergence

Figure T5.3 Adapted Elevation Patterns at 150° Azimuth

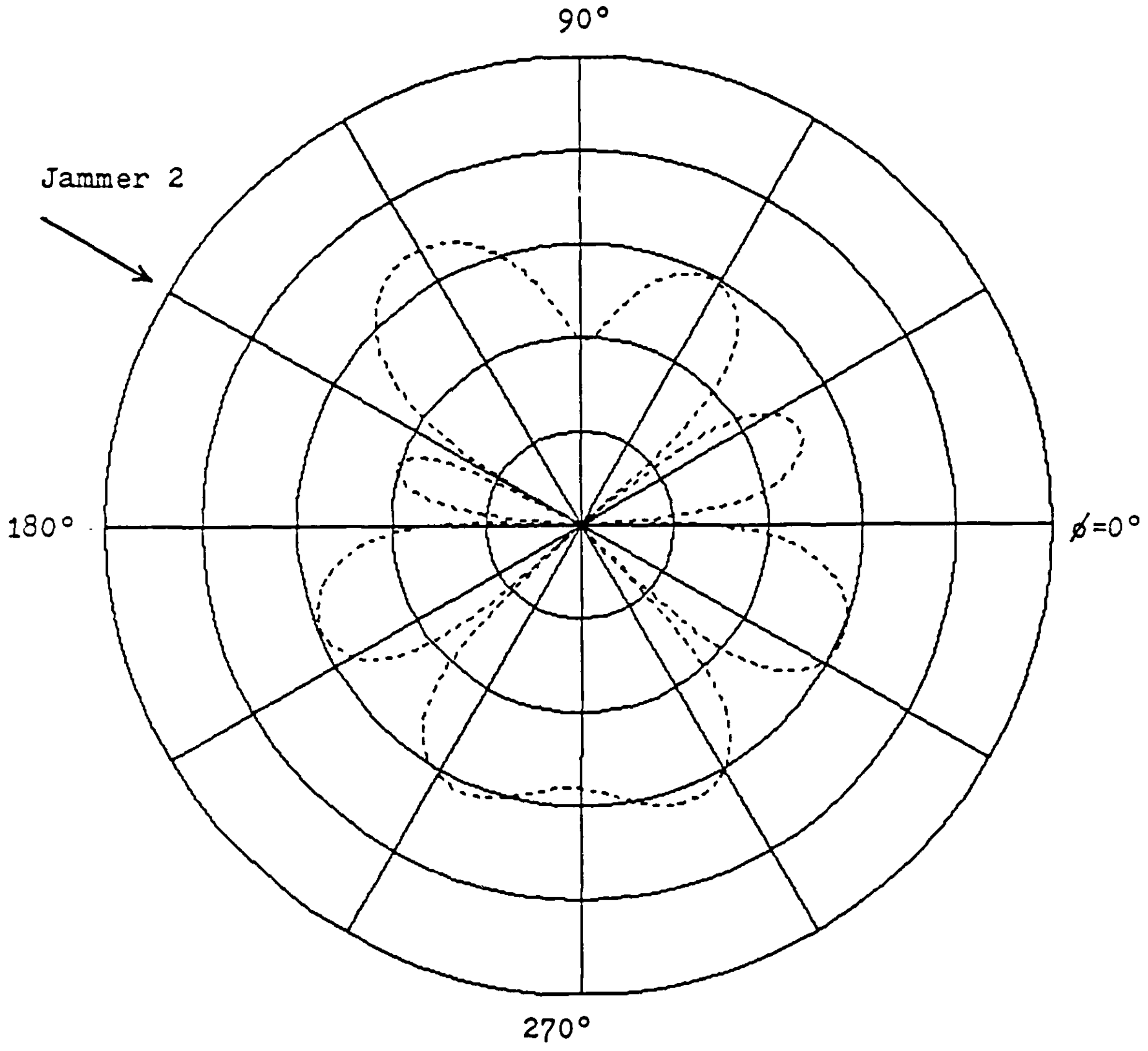
(Figure B)



Fastest Convergence

Figure T5.4 Adapted Azimuthal Patterns at 75° Elevation

(Figure A)



Slowest Convergence

Figure T5.4 Adapted Azimuthal Patterns at 75° Elevation

(Figure B)

TEST SUMMARY

For convenience, the results from Tests 1-4 are re-tabulated as shown in Table 7.5. Also, Figures 7.16 - 7.19 track the performances of each algorithm for identical convergences in each of the three channels tested.

Table 7.5 Comparison of Convergence Properties for the Adaptive Algorithms

ADAPTIVE ALGORITHM	IDEAL CHANNEL		CHANNEL 2		CHANNEL 3	
	mean	σ	mean	σ	mean	σ
LMS	313	108	547	171	543	173
CONST. LMS (with signal)	537	81	1155	699	1144	543
CONST. LMS (no signal)	337	81	593	88	598	106
UPDATE COVARIANCE	10	7	26	38	29	43

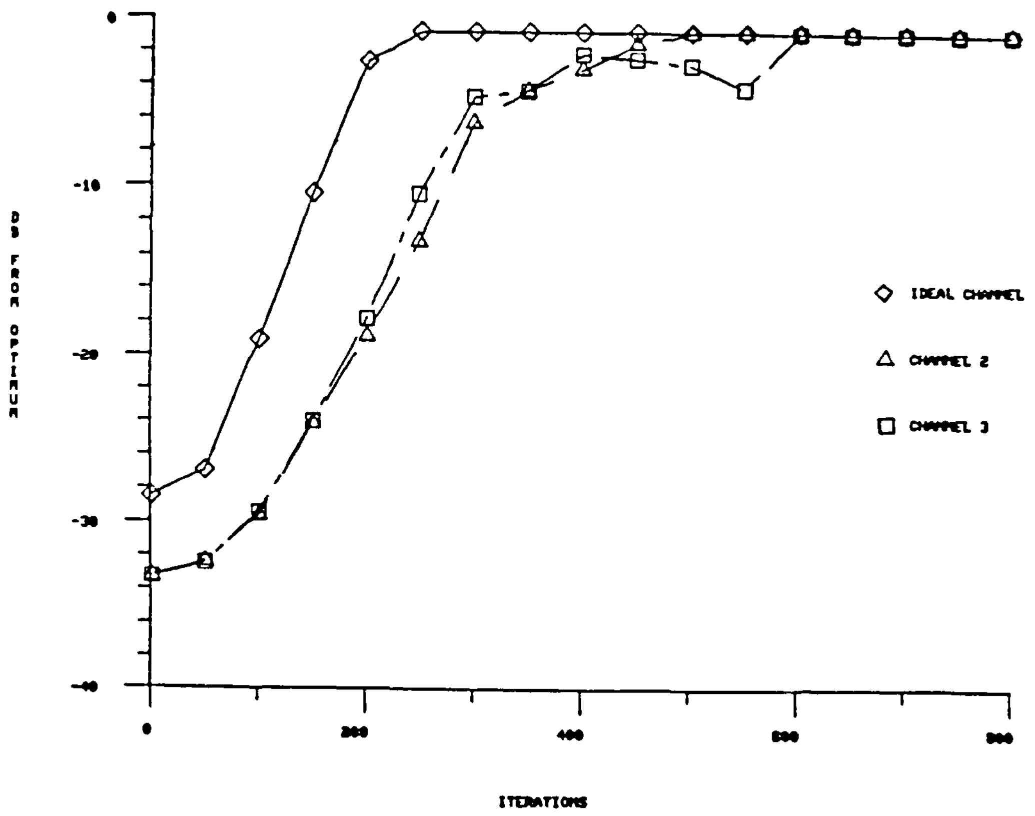


Figure 7.16 Convergence Summary for LMS Algorithms

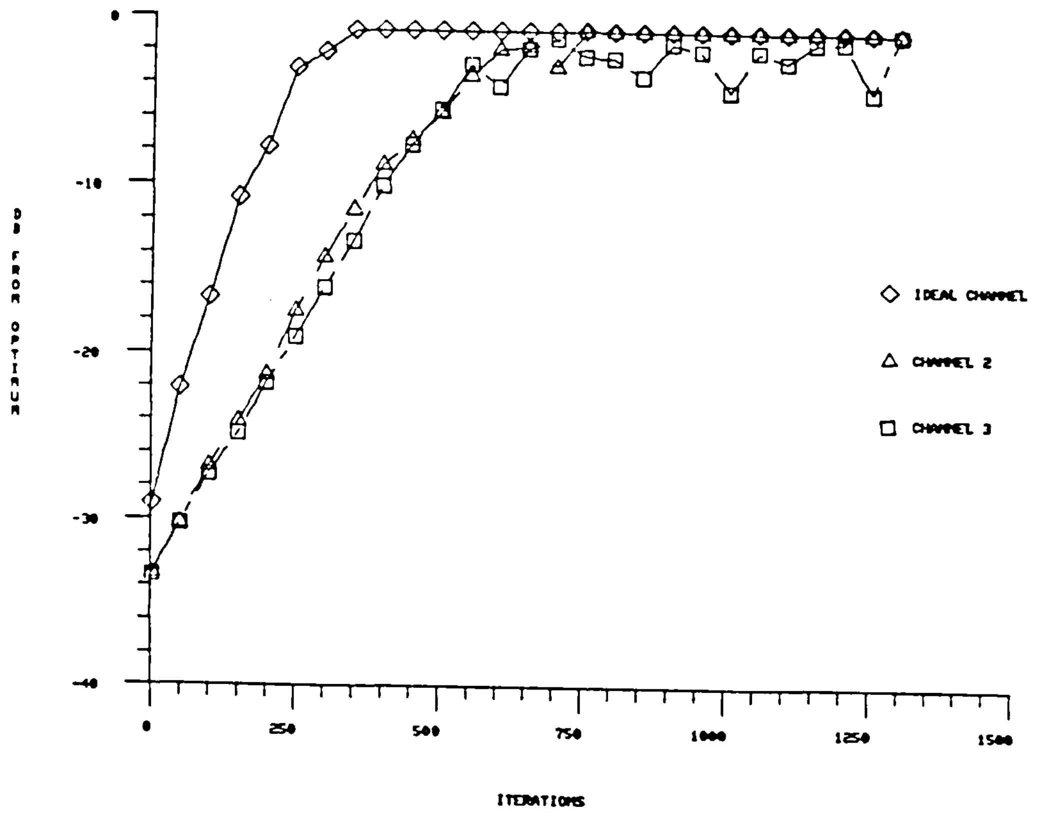


Figure 7.17 Convergence Summary for Constrained LMS Algorithm

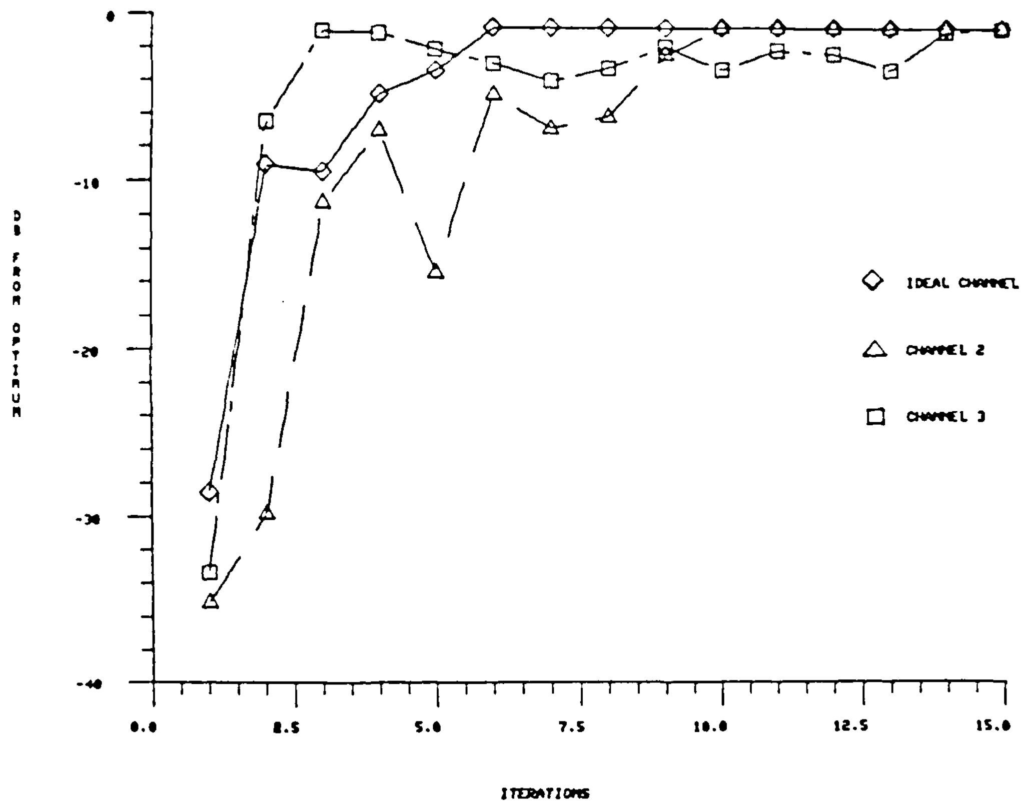


Figure 7.13 Convergence Summary for Update Covariance Algorithm

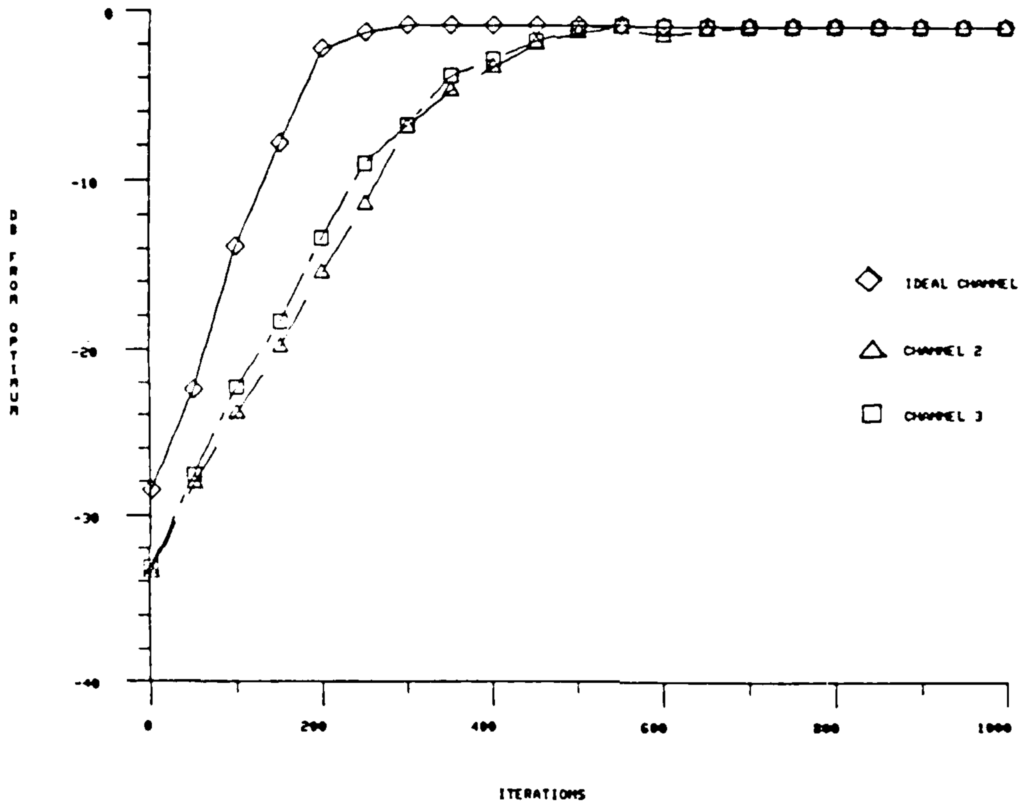


Figure 7.19 Convergence Summary for Constrained LMS Algorithm (no signal present)

7.6 Test6: Investigation of an Alternate Antenna Geometry

In an effort to understand the algorithm performance which results from a change in the antenna geometry, Tests 2 and 3 were repeated for the geometry shown in Figure 7.20. All other parameters from the tests were held fixed so that any differences may be directly attributed to the new geometry.

This rhomboid pattern was chosen somewhat at random, and is not in any way special. It was chosen simply to provide the simulation with a geometry different from the rectangular array used elsewhere.

The antenna plots for this case do show a difference in shape as compared to those for the rectangular geometry. This is simply due to the change in the antenna array factor caused by the alternate geometry. Notice, however, that the general trend of the patterns is the same for both geometries. We see only slight variations in the patterns as functions of both the controlling algorithm and the channel characteristics.

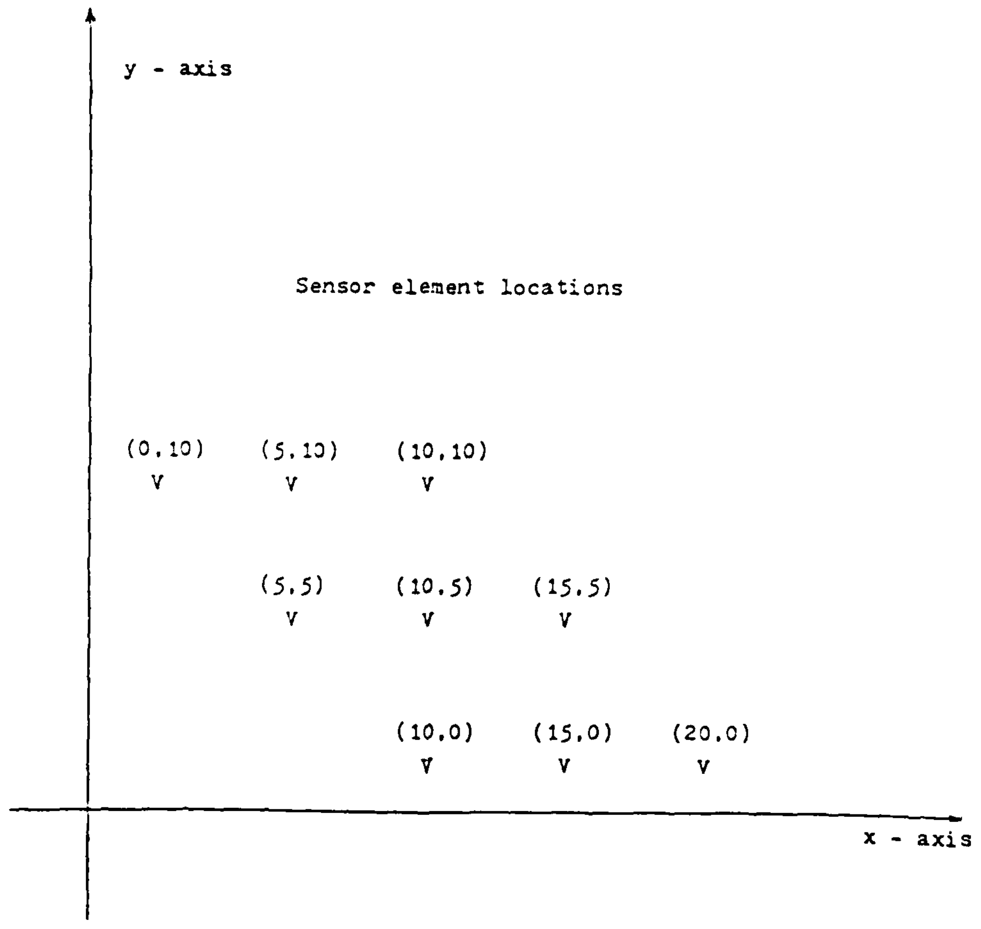


Figure 7.20 Alternate Antenna Geometry for Rhomboid Geometry

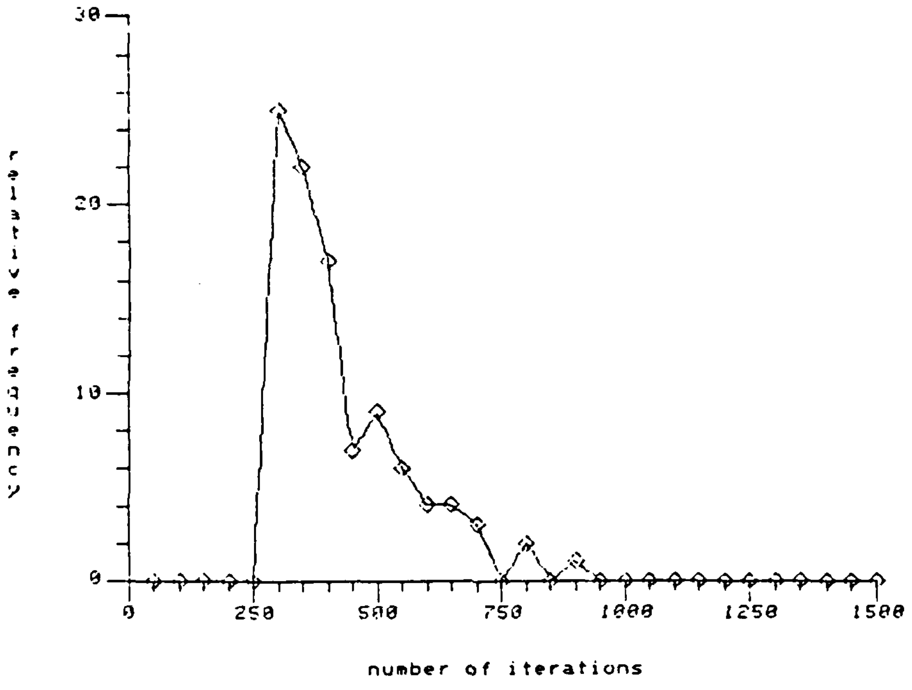


Figure 7.21 Convergence Histogram of LMS Algorithm in Channel 2 for Rhomboid Geometry

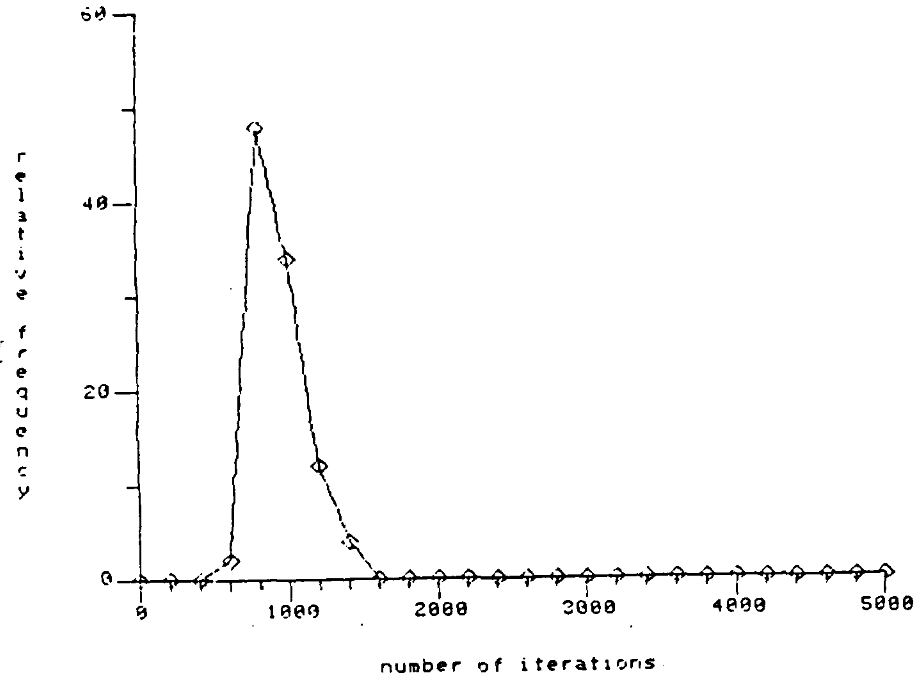


Figure 7.22 Convergence Histogram of Constrained LMS Algorithm in Channel 2 for Rhomboid Geometry

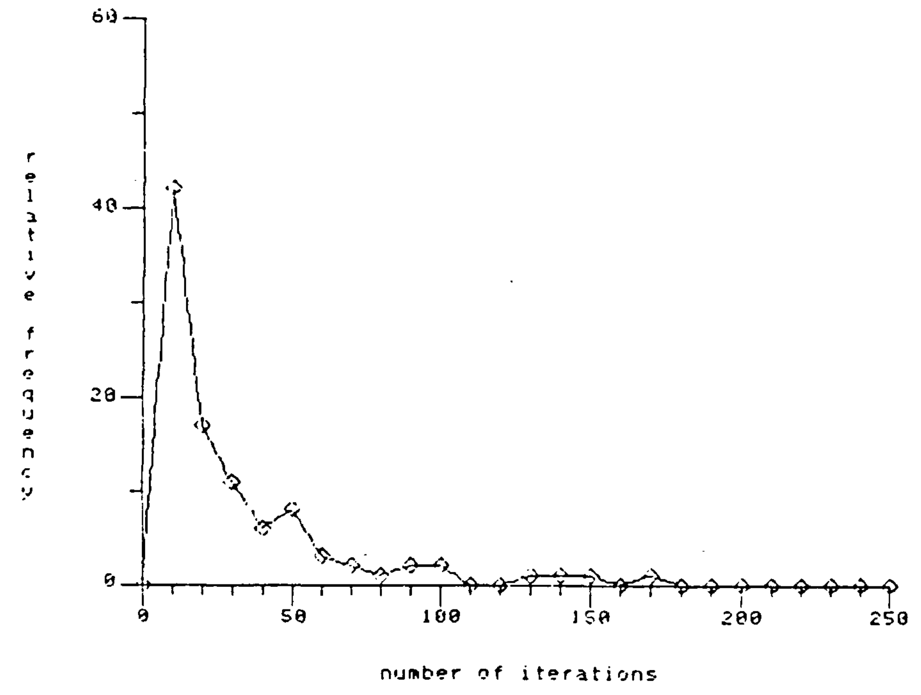


Figure 7.23 Convergence Histogram of Update Covariance Algorithm in Channel 2 for Rhomboid Geometry

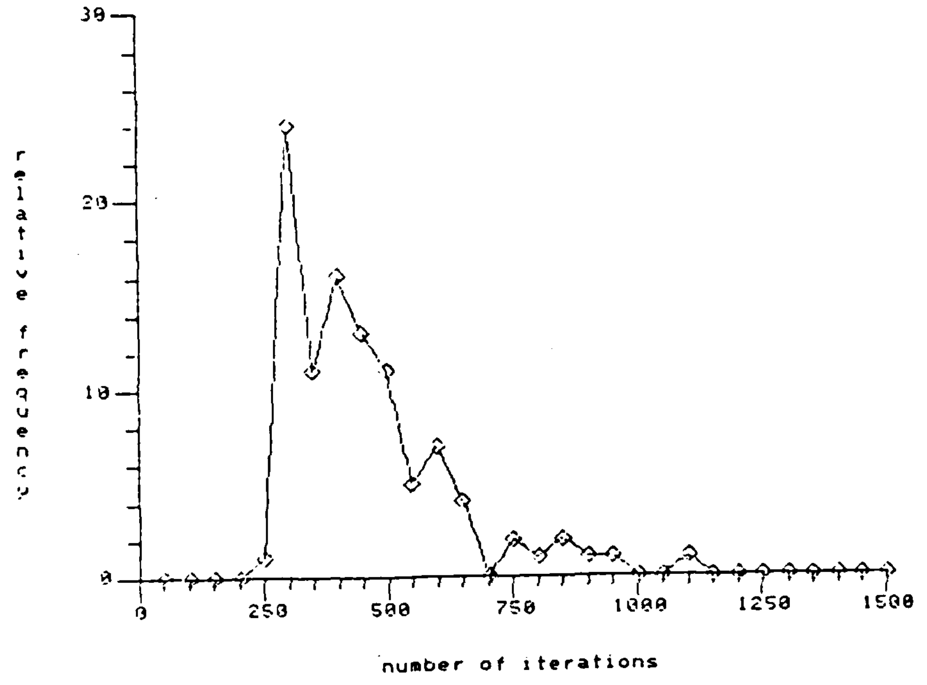


Figure 7.24 Convergence Histogram of LMS Algorithm in Channel 3 for Rhomboid Geometry

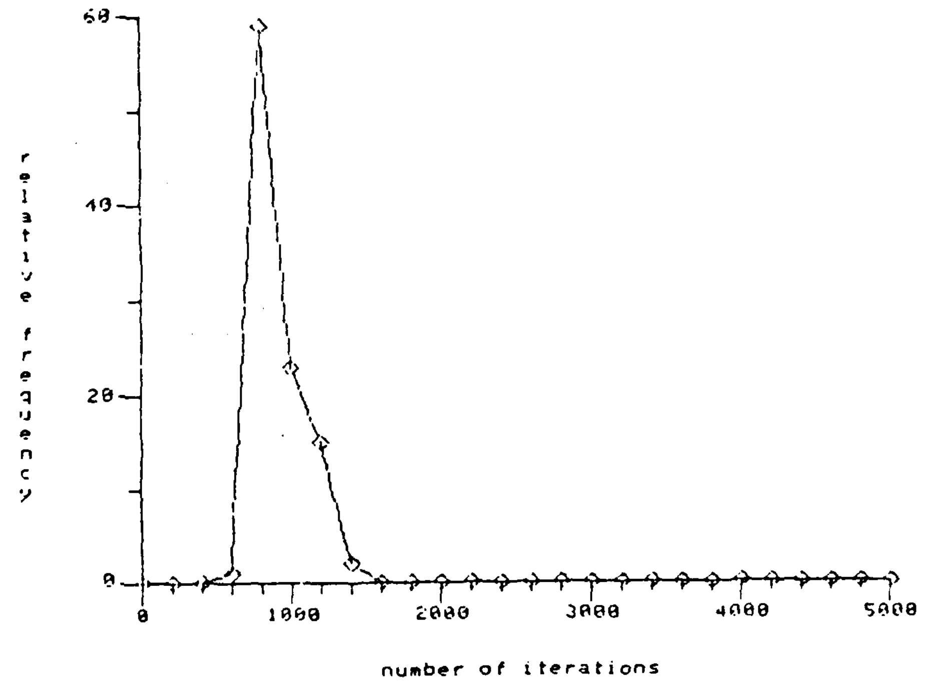


Figure 7.25 Convergence Histogram of Constrained LMS Algorithm in Channel 3 for Rhomboid Geometry

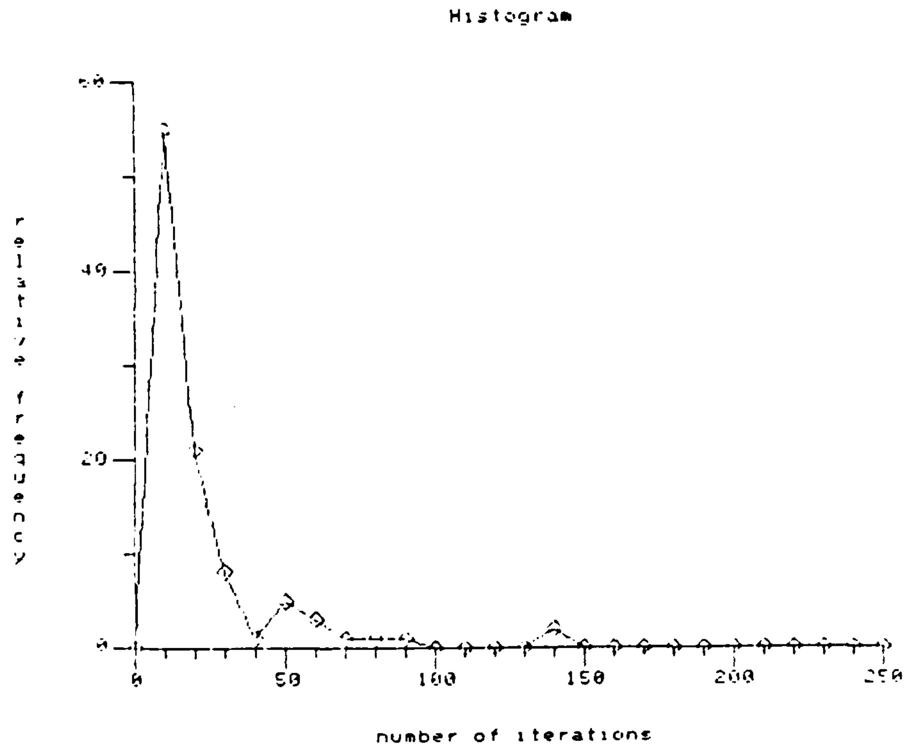
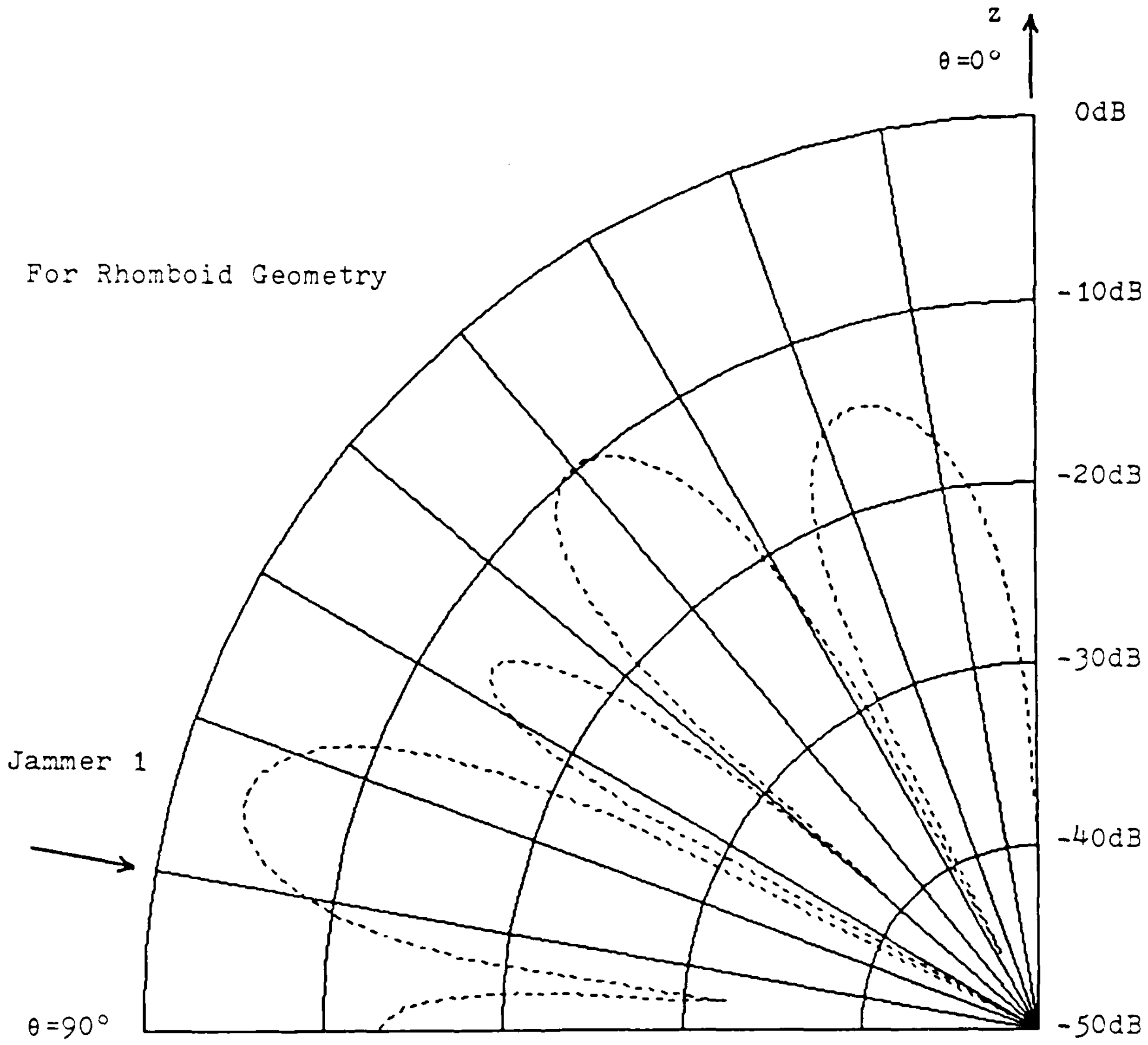


Figure 7.26 Convergence Histogram of Update Covariance Algorithm in Channel 3 for Rhomboid Geometry

SUMMARY OF PLOTS FOR TEST6 - RHOMBOID GEOMETRY

Figure T6.1:	Unadapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To indicate initial gain in direction of Jammer 1
Figure T6.2:	LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T6.3:	Constrained LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T6.4:	Update Covariance-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To demonstrate nulling of Jammer 1 (in Channel 2) by each algorithm with new antenna geometry
Figure T6.5:	LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T6.6:	Constrained LMS-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Figure T6.7:	Update Covariance-adapted antenna plot at $\phi = 90°$, $\theta = 80°$
Purpose:	To demonstrate nulling of Jammer 1 (in Channel 3) by each algorithm with new antenna geometry
Figure T6.8:	Unadapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To indicate initial gain in direction of Jammer 2
Figure T6.9:	LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T6.10:	Constrained LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T6.11:	Update Covariance-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To demonstrate nulling of Jammer 2 (in Channel 2) by each algorithm with new antenna geometry
Figure T6.12:	LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T6.13:	Constrained LMS-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Figure T6.14:	Update Covariance-adapted antenna plot at $\phi = 150°$, $\theta = 75°$
Purpose:	To demonstrate nulling of Jammer 2 (in Channel 3) by each algorithm with new antenna geometry

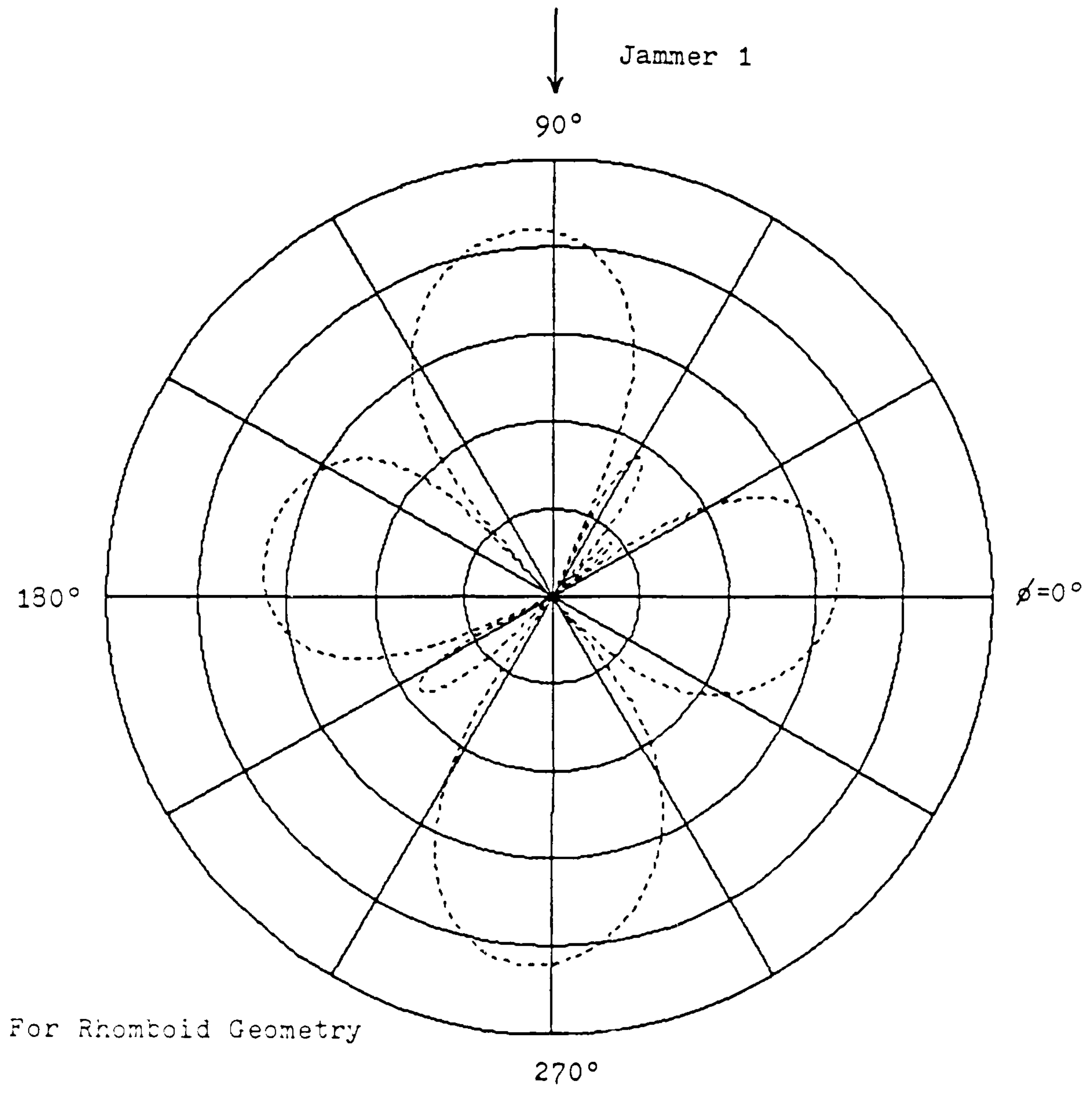


Elevation plot at 90° azimuth

Arrow indicates arrival angle of Jammer 1 (θ=80°)

Figure T6.1 Unadapted Antenna Pattern in Direction of Jammer 1

(Figure A)

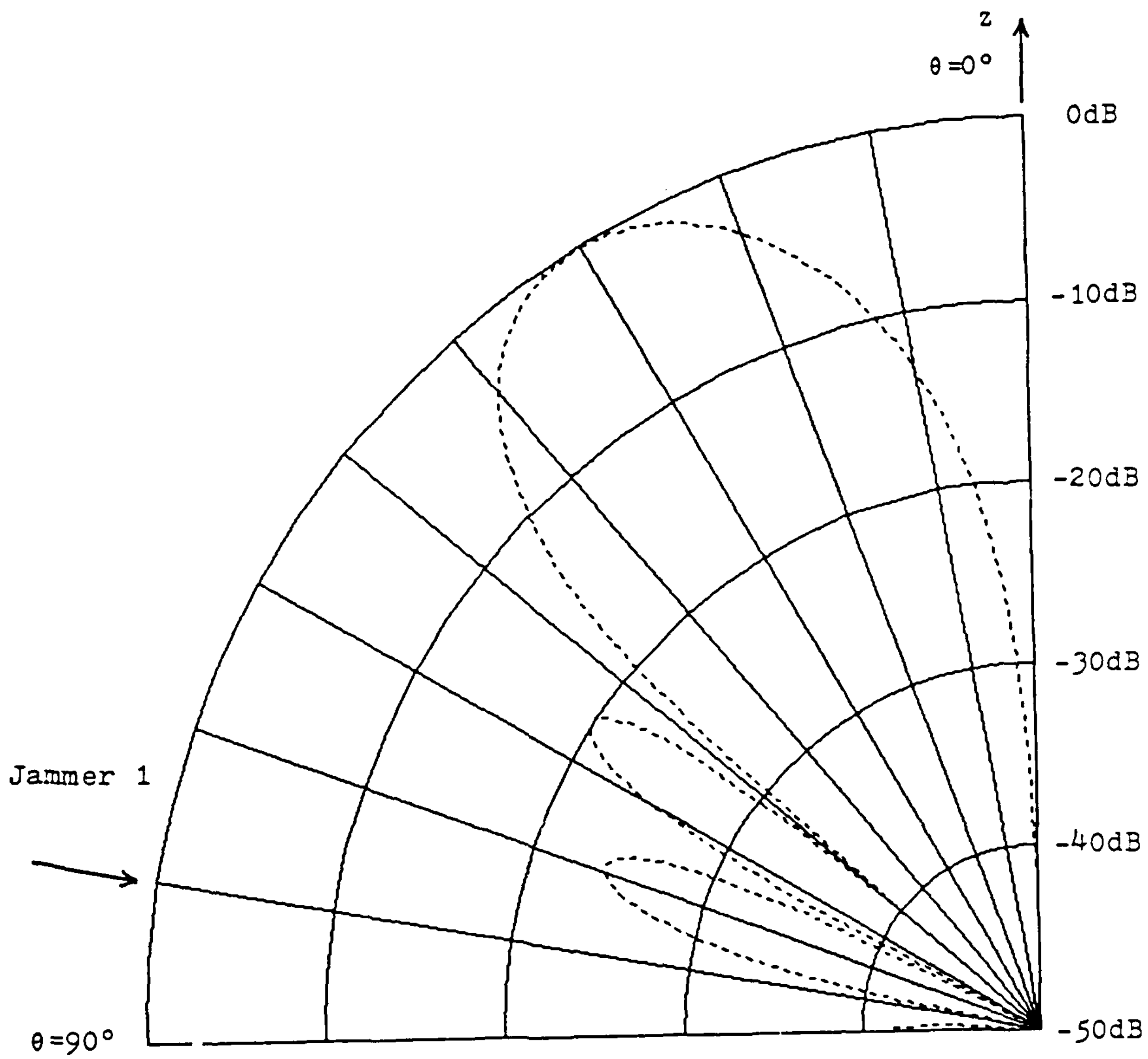


Azimuthal plot at 80° elevation

Arrow indicates arrival angle of Jammer 1 (ø=90°)

Figure T6.1 Unadapted Antenna Pattern in Direction of Jammer 1

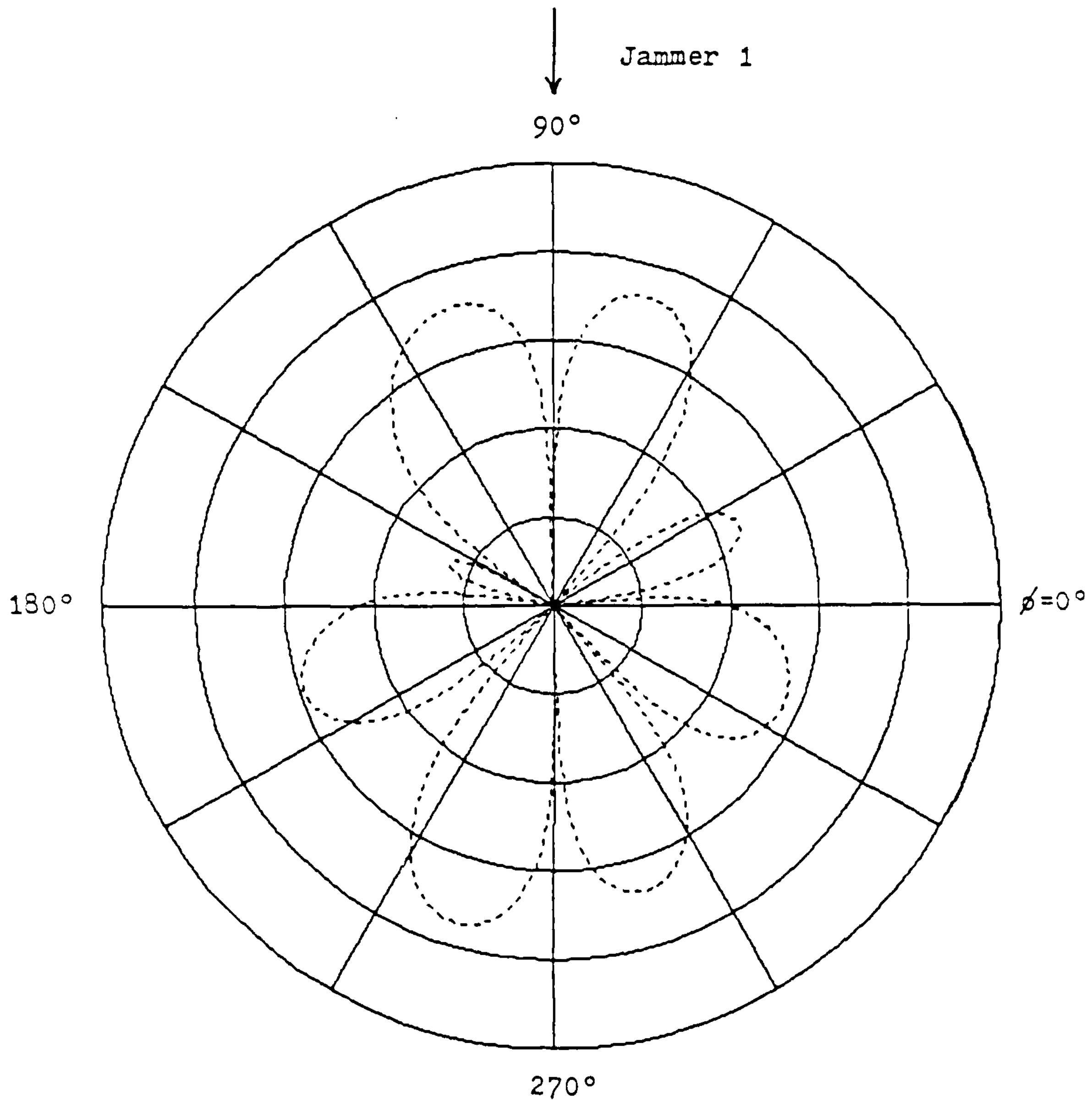
(Figure B)



For Rhomboid Geometry

Figure T6.2 LMS-Adapted Pattern in Direction of Jammer 1

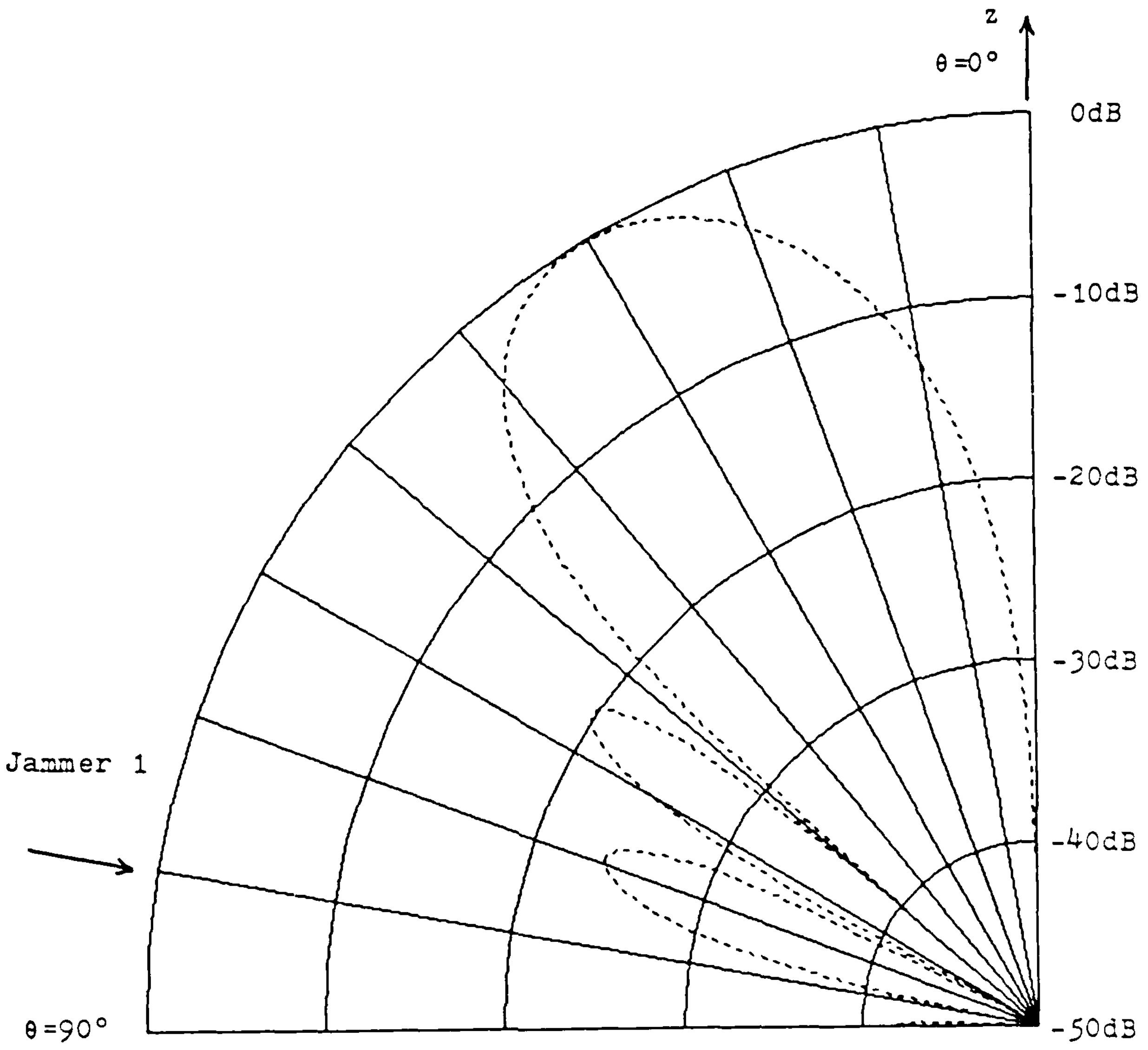
(Figure A)



For Rhomboid Geometry

Figure T6.2 LMS-Adapted Pattern in Direction of Jammer 1

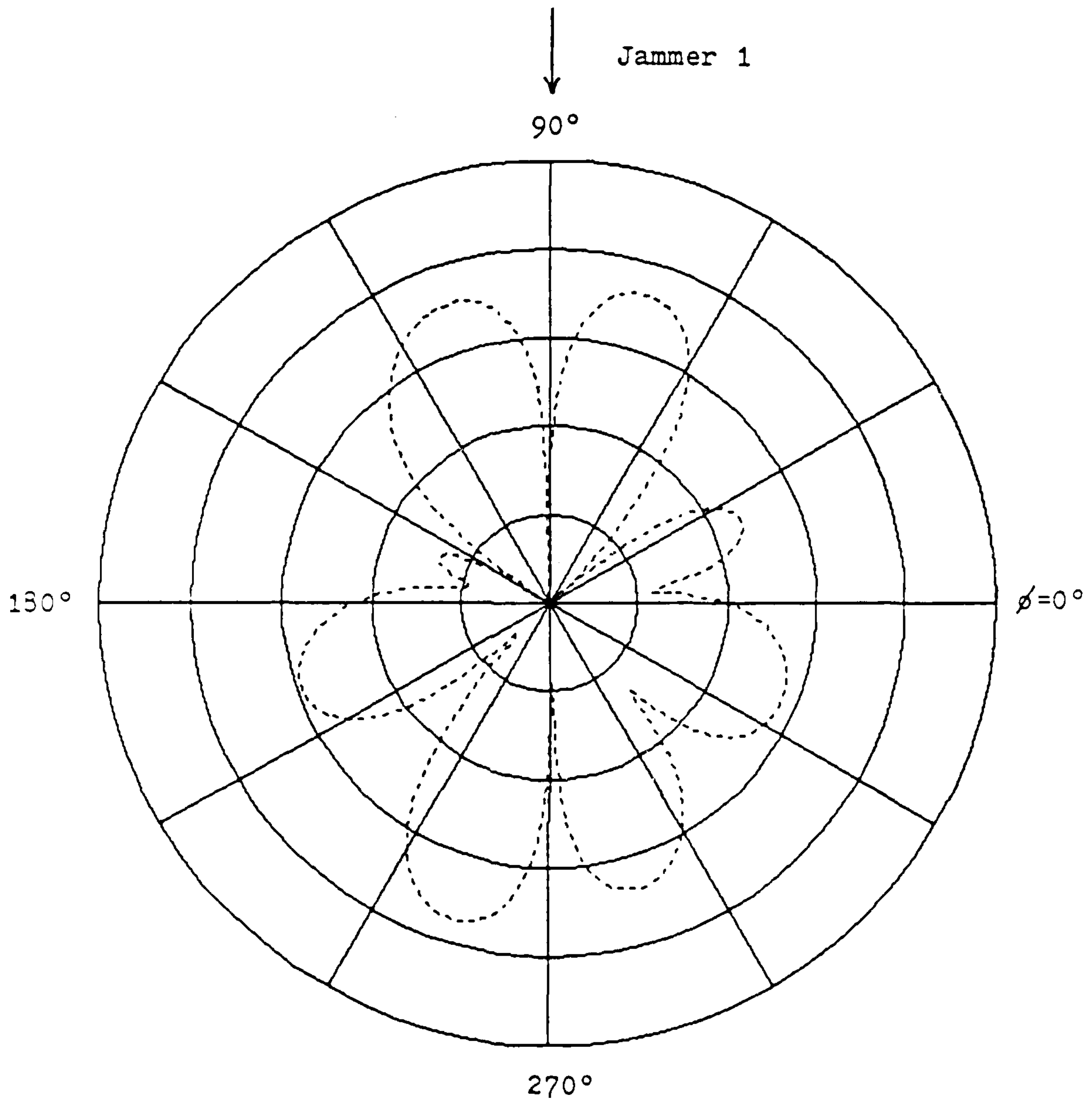
(Figure B)



For Rhomboid Geometry

Figure T6.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

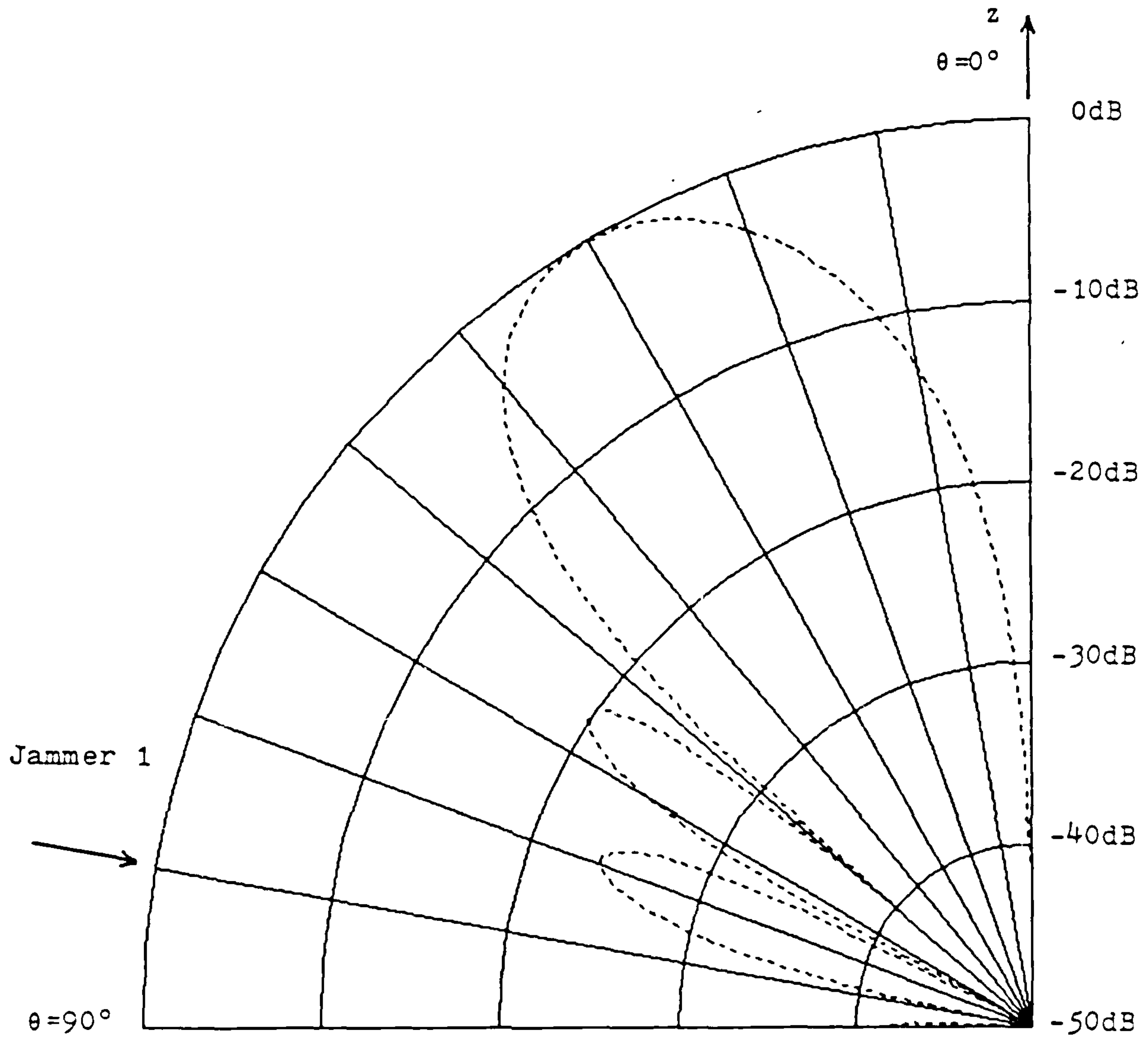
(Figure A)



For Rhomboid Geometry

Figure T6.3 Constrained LMS-Adapted Pattern in Direction of Jammer 1

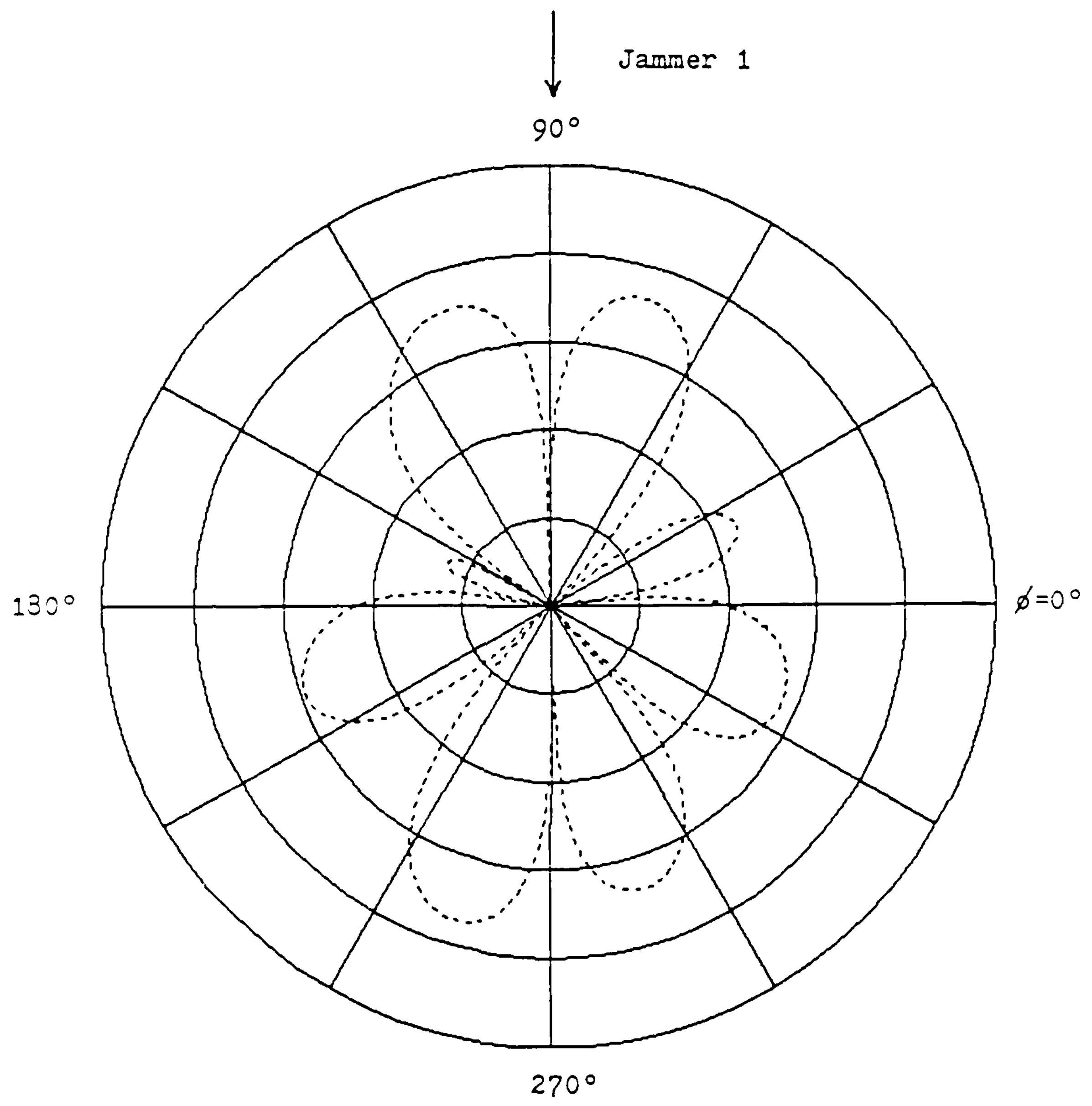
(Figure B)



For Rhomboid Geometry

Figure T6.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

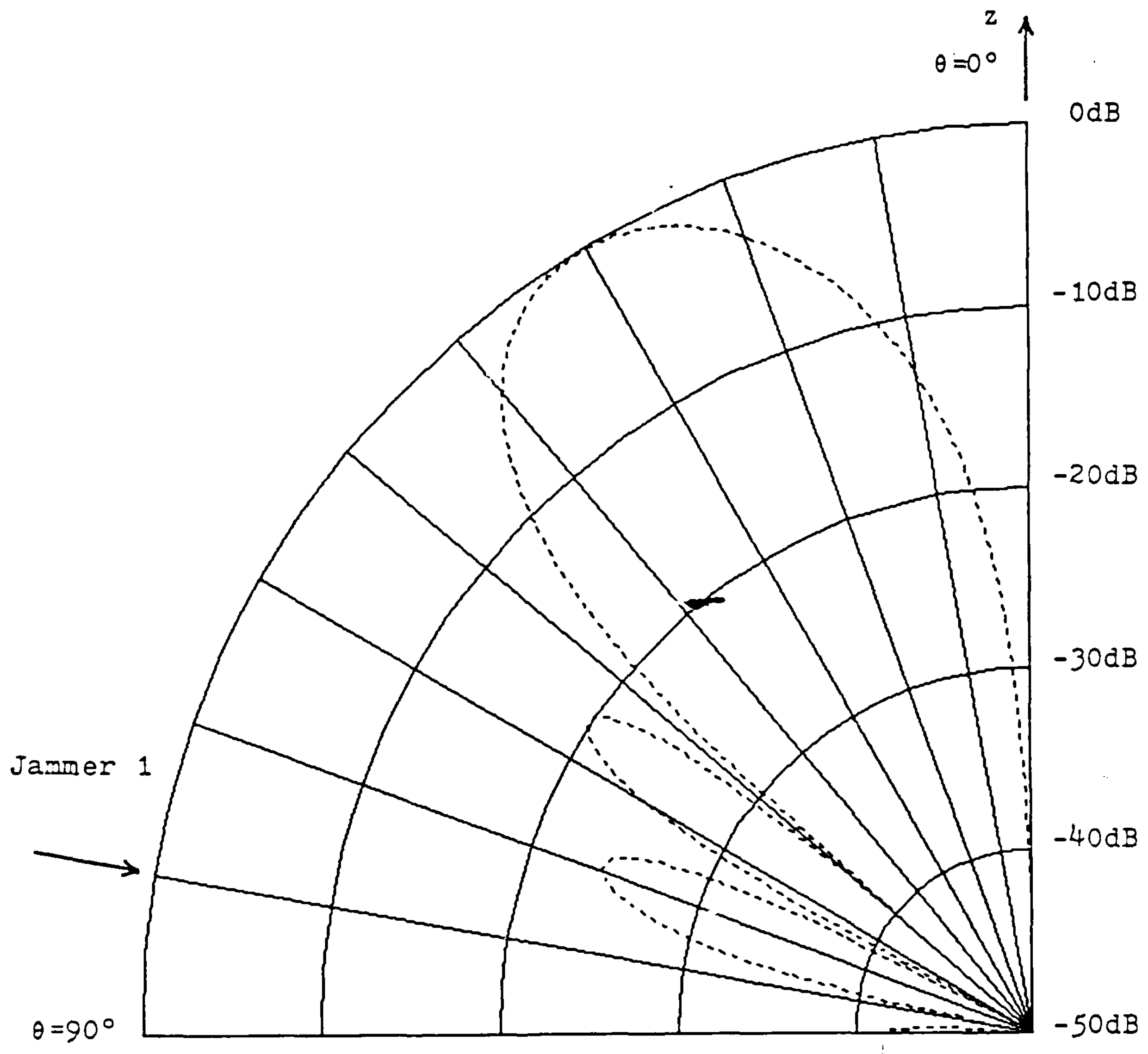
(Figure A)



For Rhomboid Geometry

Figure T6.4 Update Covariance-Adapted Pattern in Direction of Jammer 1

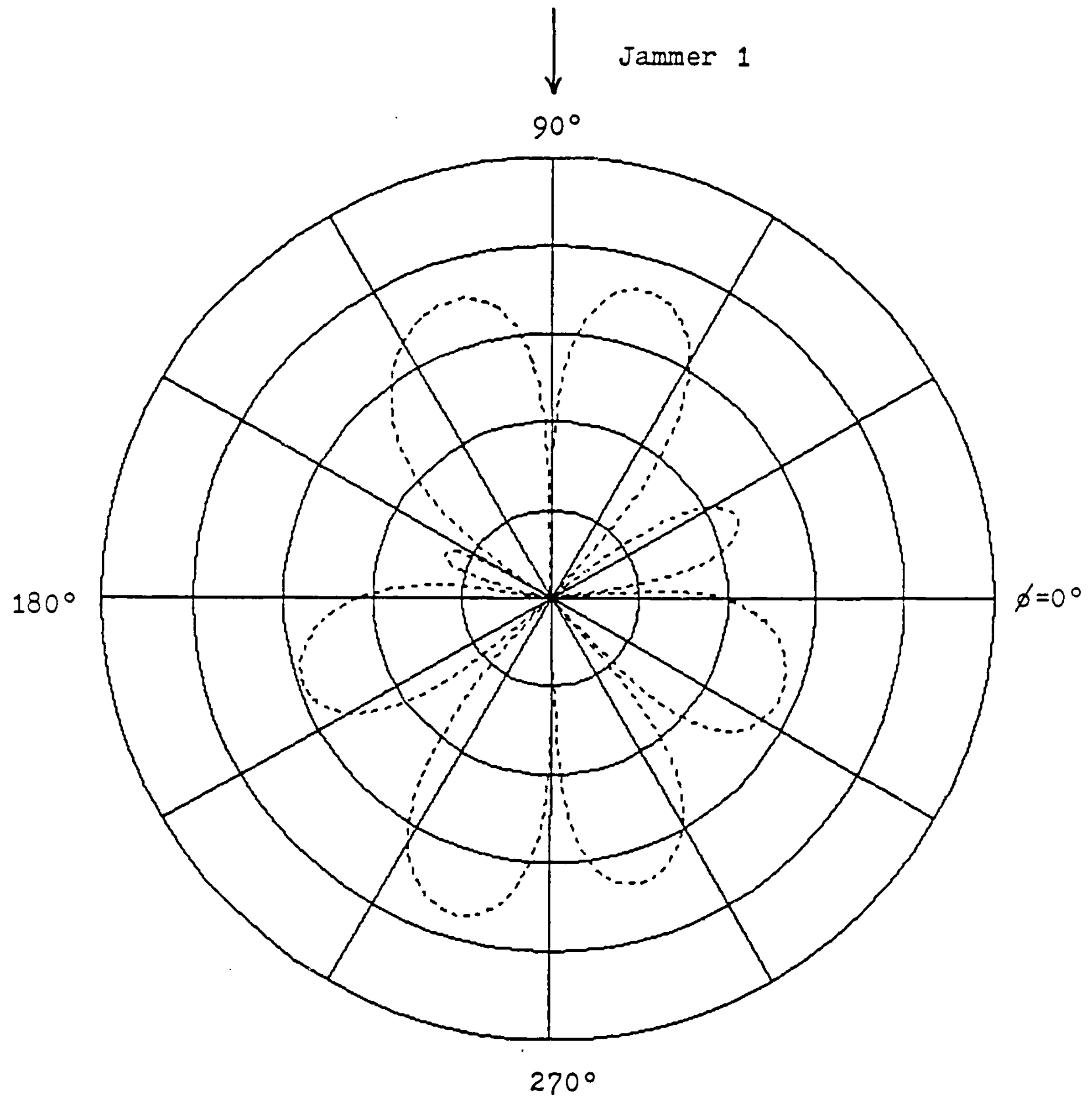
(Figure B)



For Rhomboid Geometry

Figure T6.5 LMS-Adapted Pattern in Direction of Jammer 1

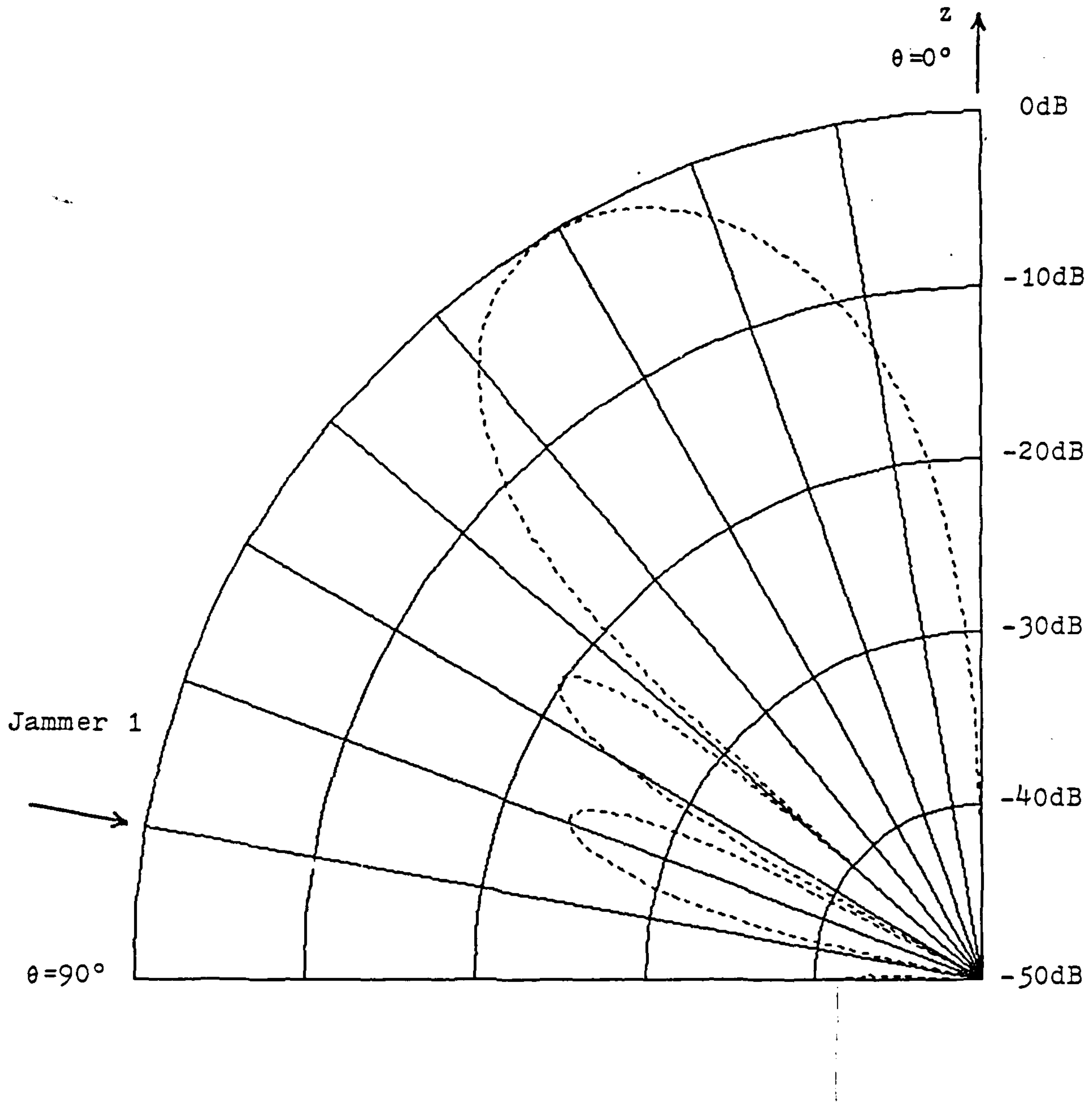
(Figure A)



For Rhomboid Geometry

Figure T6.5 LMS-Adapted Pattern in Direction of Jammer 1

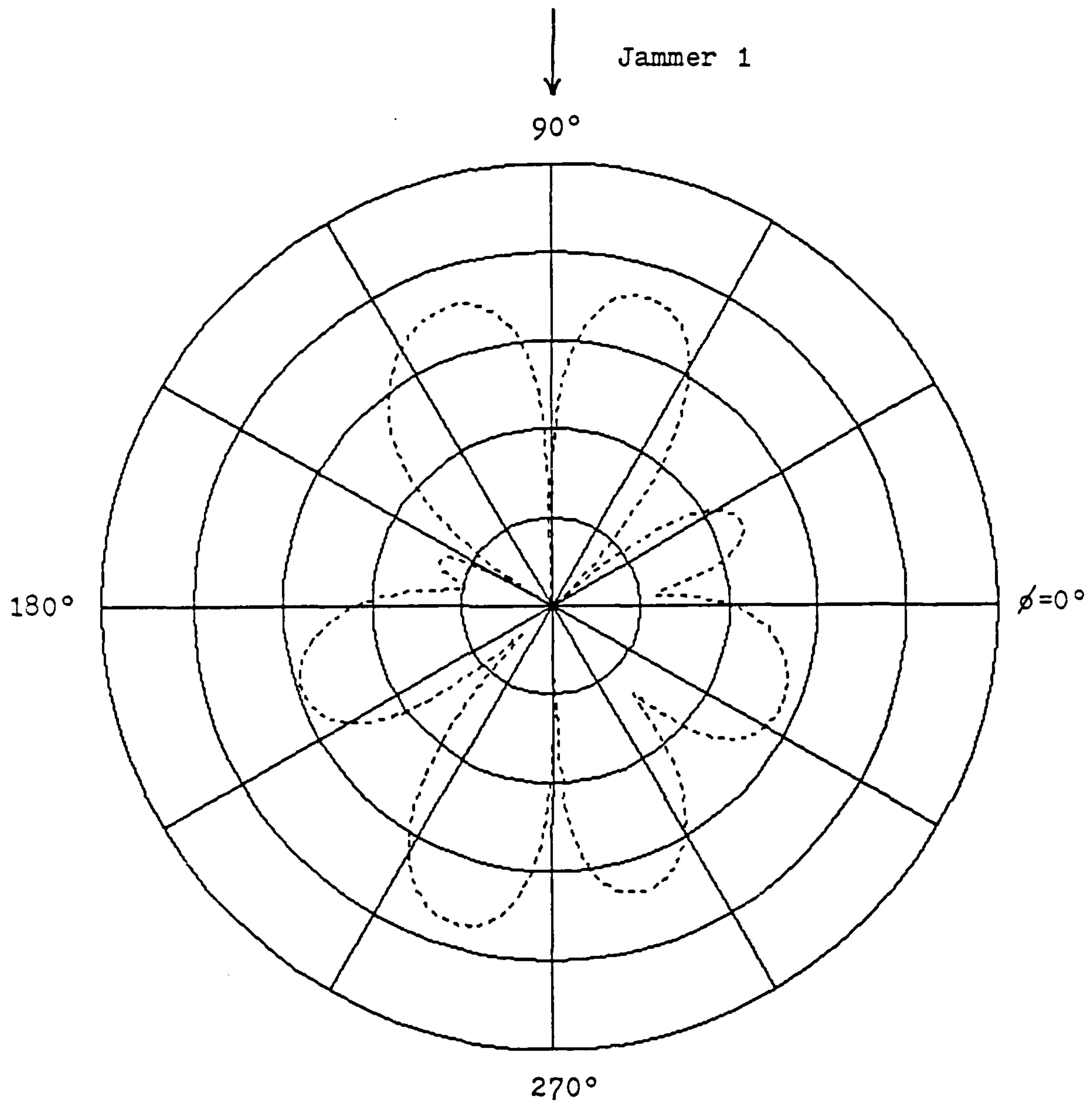
(Figure B)



For Rhomboid Geometry

Figure T6.6 Constrained LMS-Adapted Pattern in Direction of Jammer 1

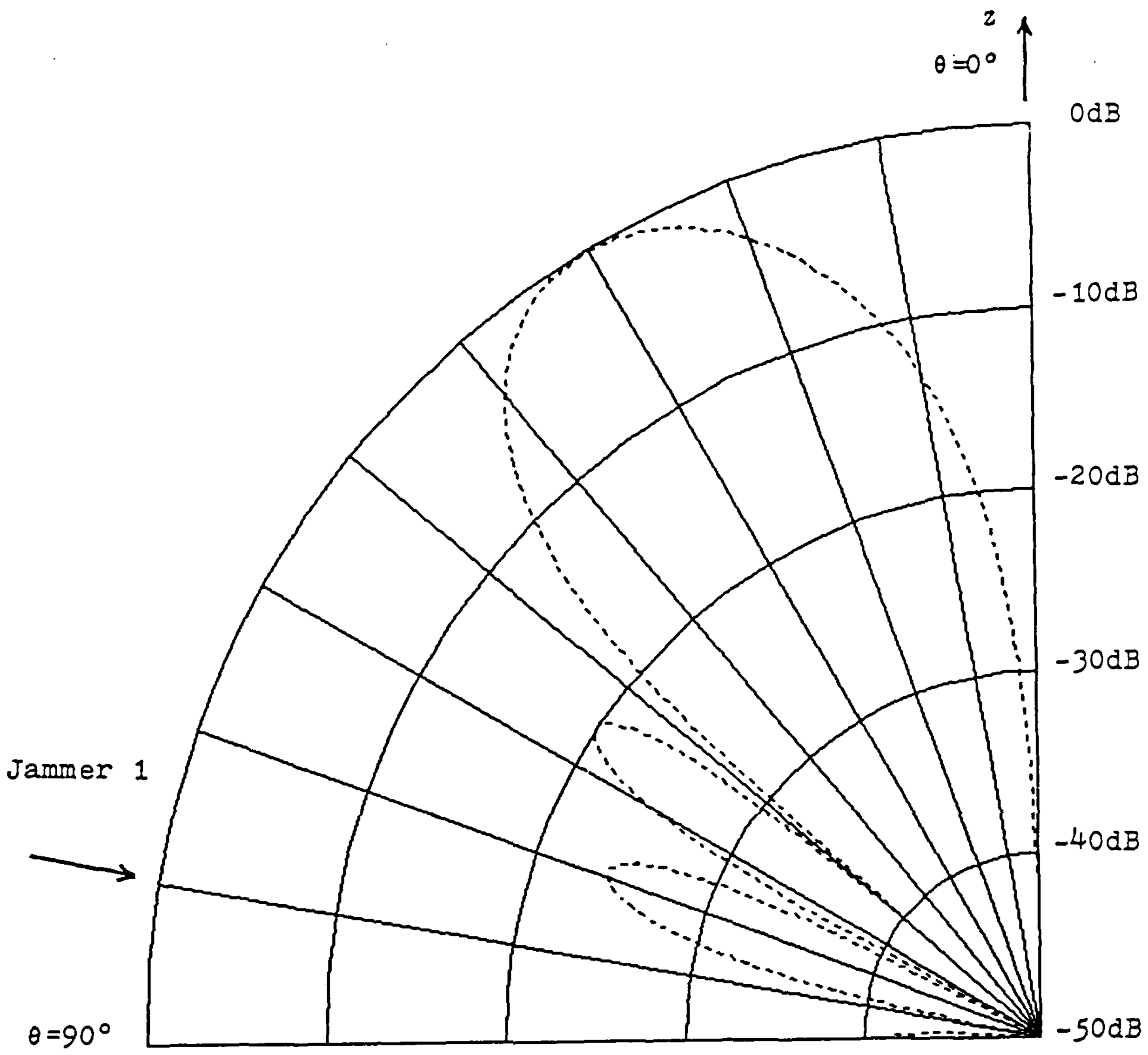
(Figure A)



For Rhomboid Geometry

Figure T6.6 Constrained LMS-Adapted Pattern in Direction of Jammer 1

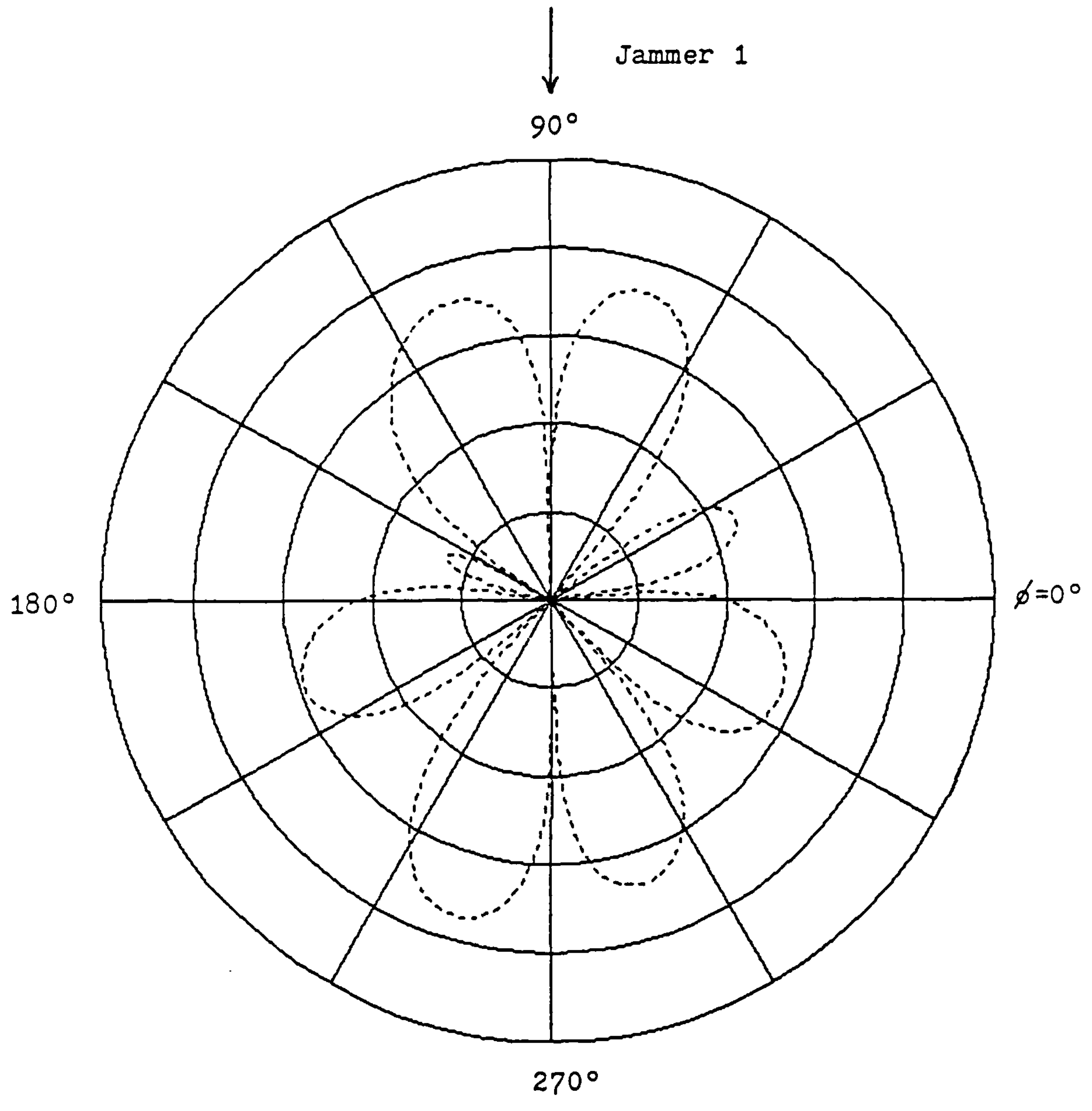
(Figure B)



For Rhomboid Geometry

Figure T6.7 Update Covariance-Adapted Pattern in Direction of Jammer 1

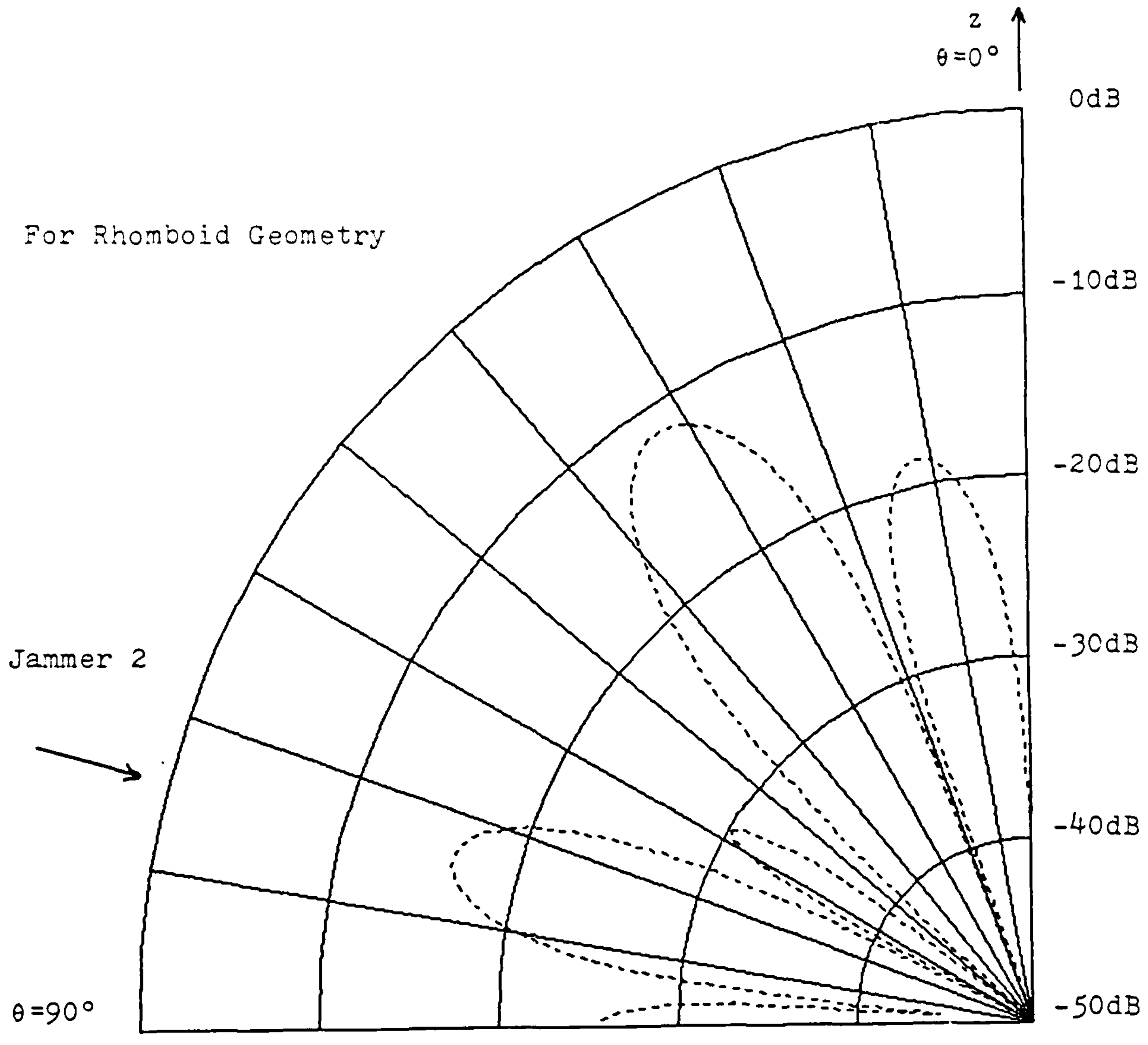
(Figure A)



For Rhomboid Geometry

Figure T6.7 Update Covariance-Adapted Pattern in Direction of Jammer 1

(Figure B)

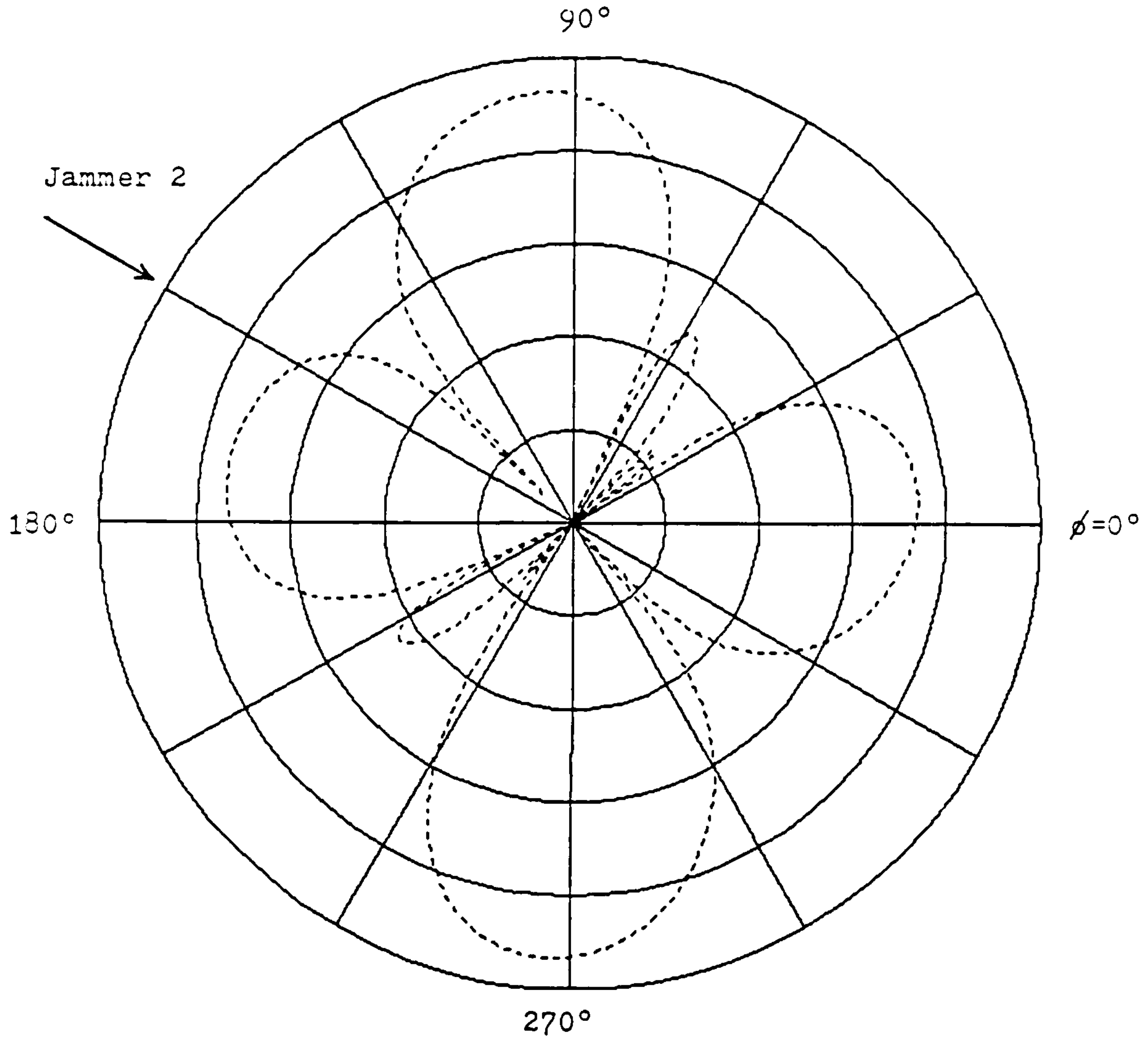


Elevation plot at 150° azimuth

Arrow indicates arrival angle of Jammer 2 (θ=75°)

Figure T6.8 Unadapted Antenna Pattern in Direction of Jammer 2

(Figure A)

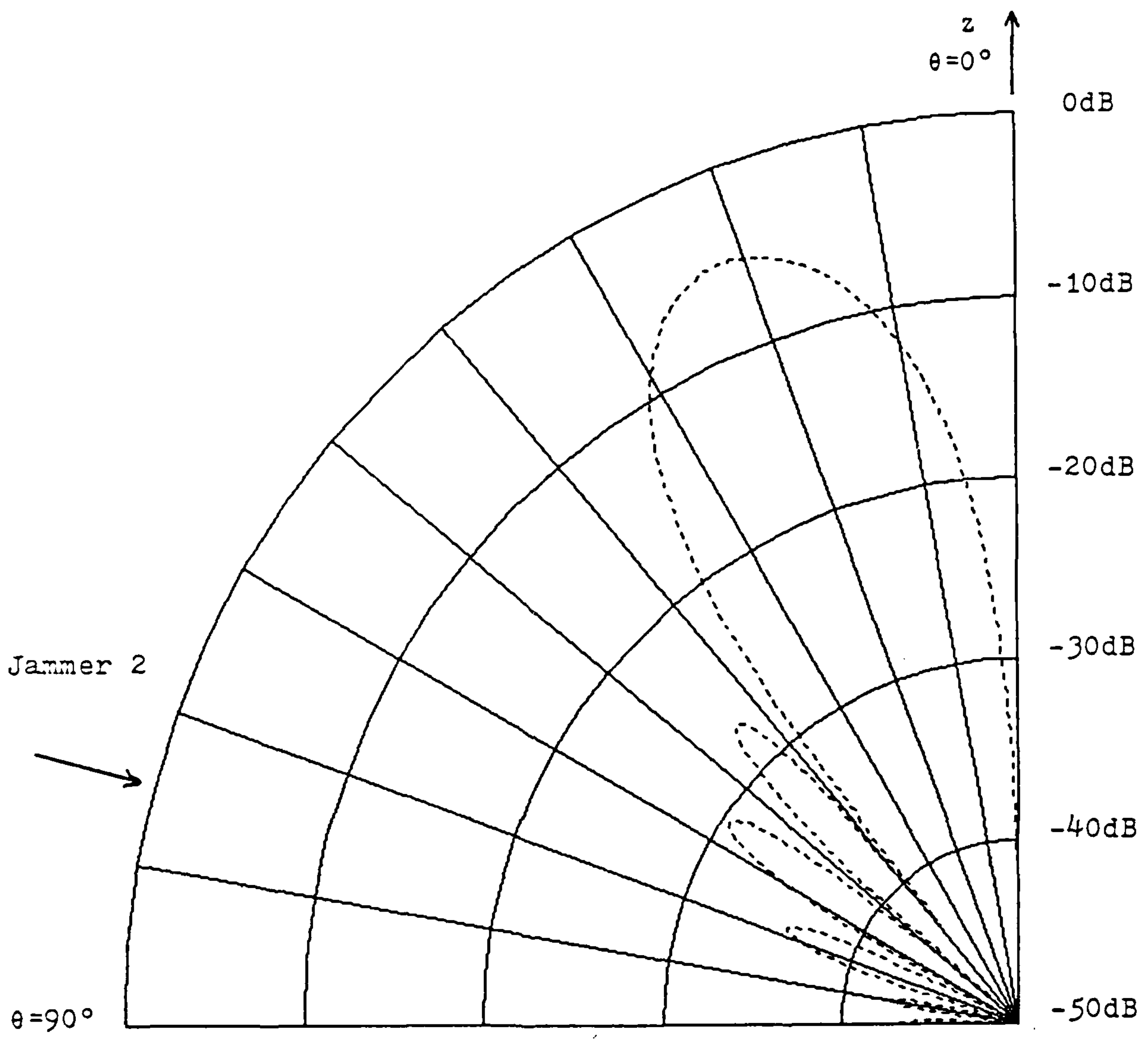


Azimuthal plot at 75° elevation

Arrow indicates arrival angle of Jammer 2 (ø=150°)

Figure T6.8 Unadapted Antenna Pattern in Direction of Jammer 2

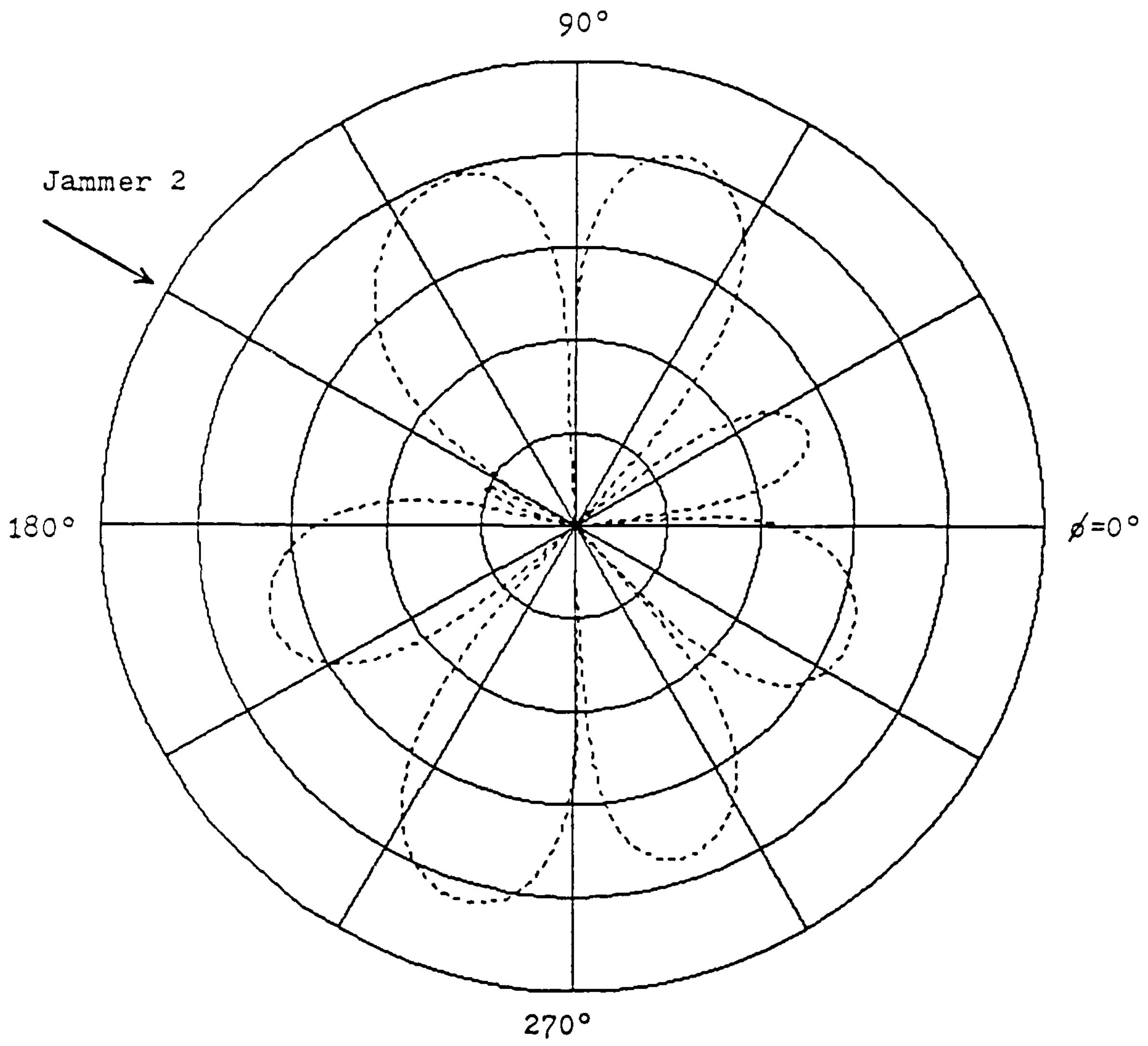
(Figure B)



For Rhomboid Geometry

Figure T6.9 LMS-Adapted Pattern in Direction of Jammer 2

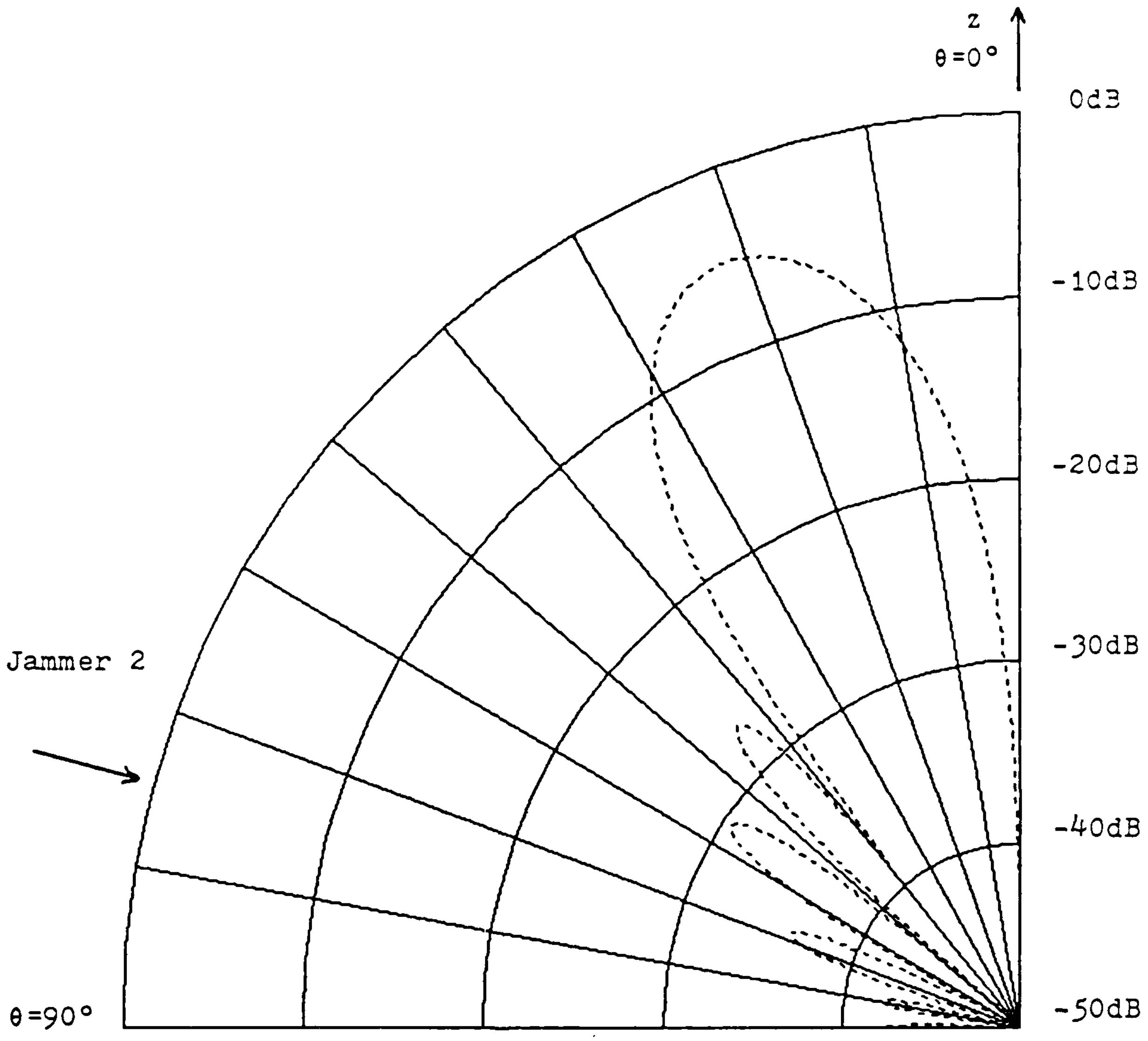
(Figure A)



For Rhomboid Geometry

Figure T6.9 LMS-Adapted Pattern in Direction of Jammer 2

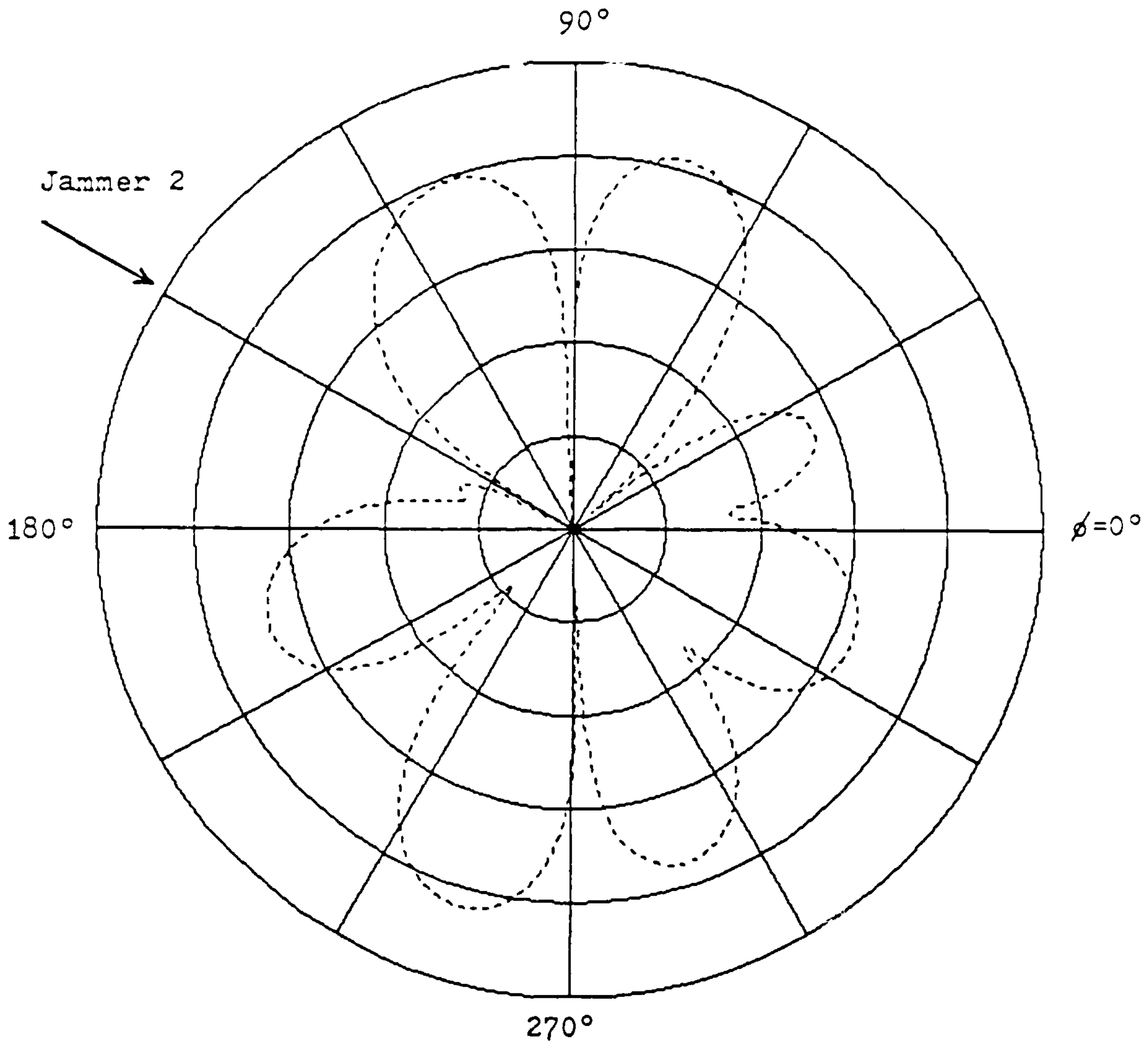
(Figure B)



For Rhomboid Geometry

Figure T6.10 Constrained LMS-Adapted Pattern in Direction of Jammer 2

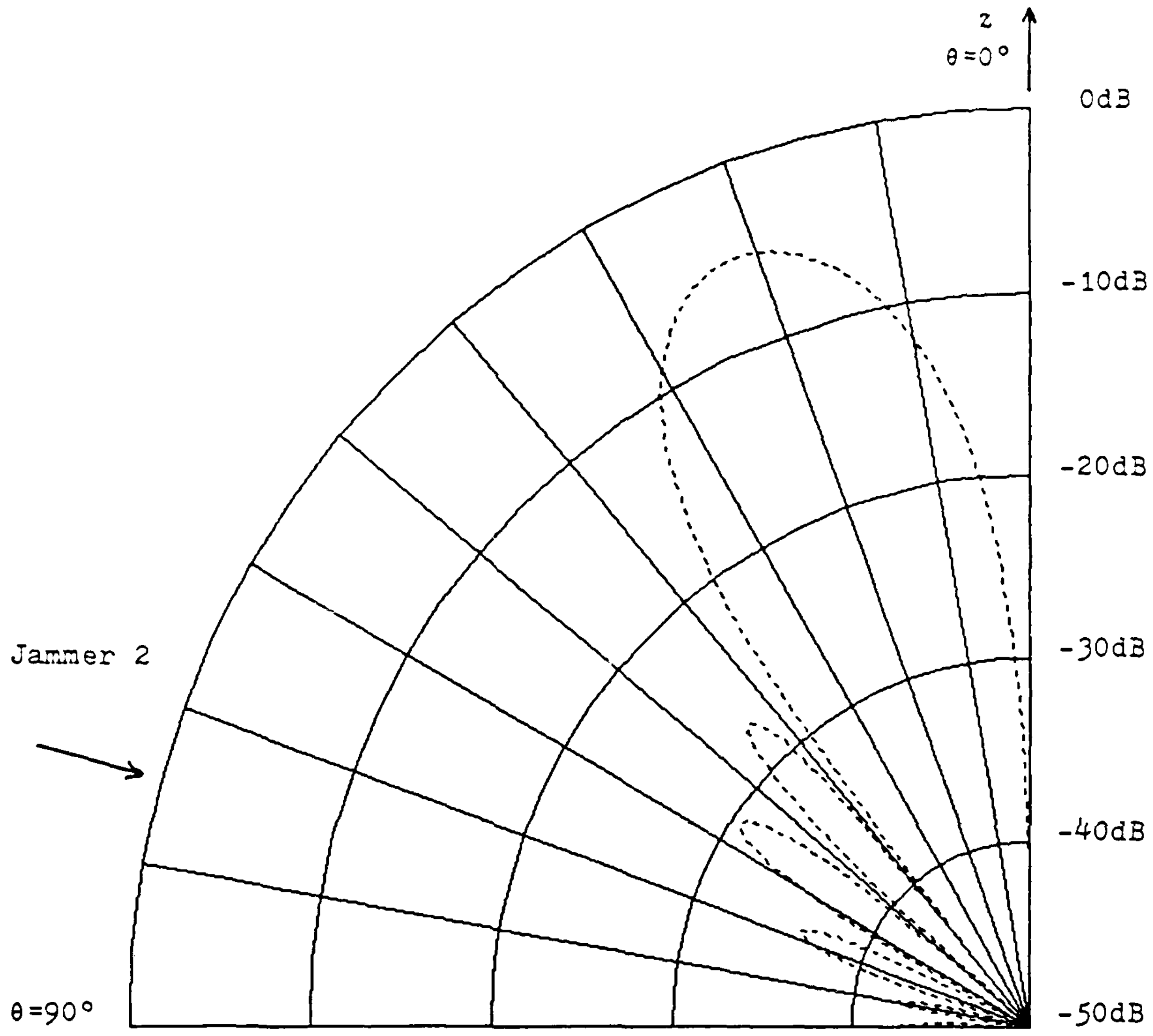
(Figure A)



For Rhomboid Geometry

Figure T6.10 Constrained LMS-Adapted Pattern in Direction of Jammer 2

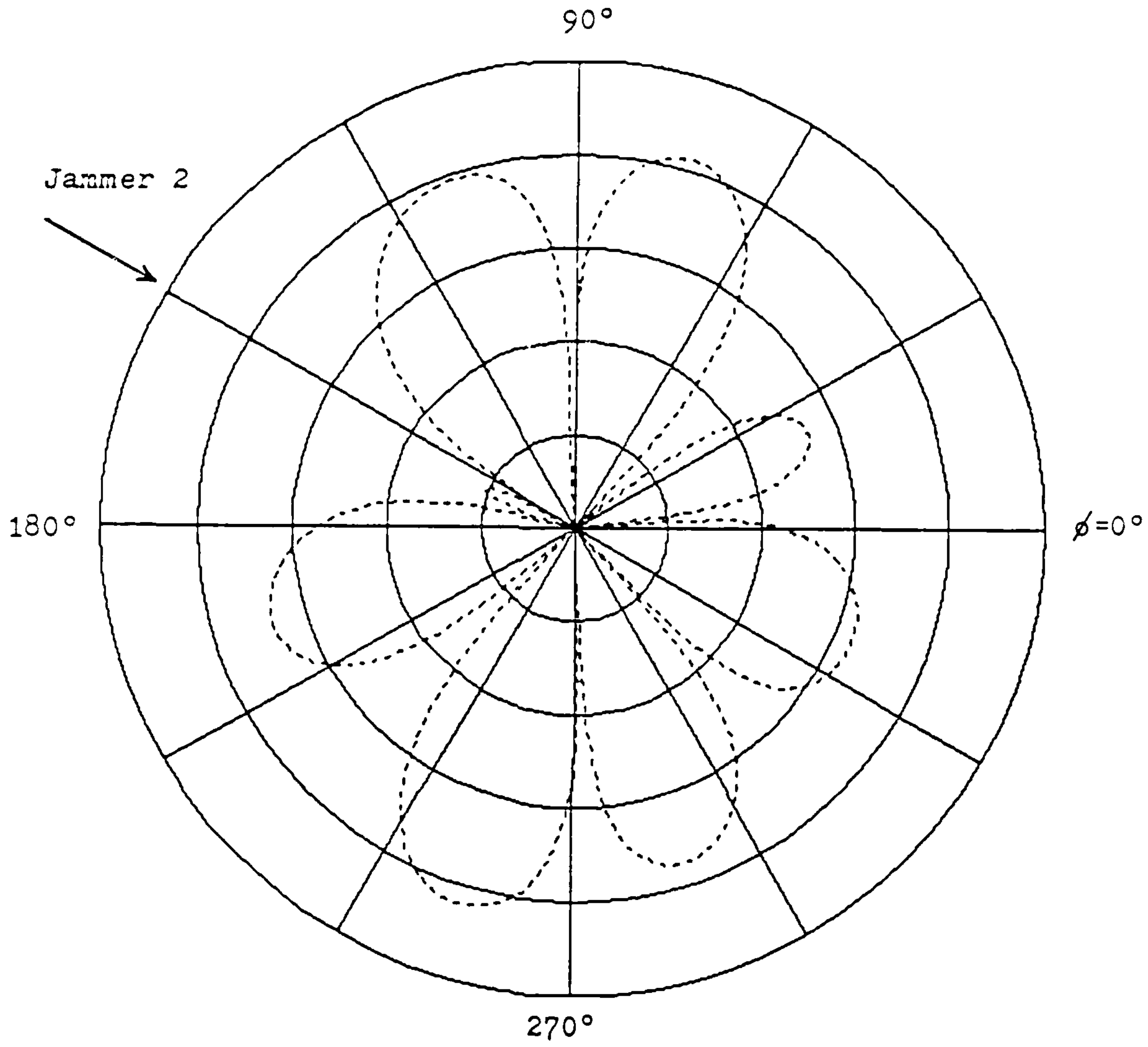
(Figure B)



For Rhomboid Geometry

Figure T6.11 Update Covariance-Adapted Pattern in Direction of Jammer 2

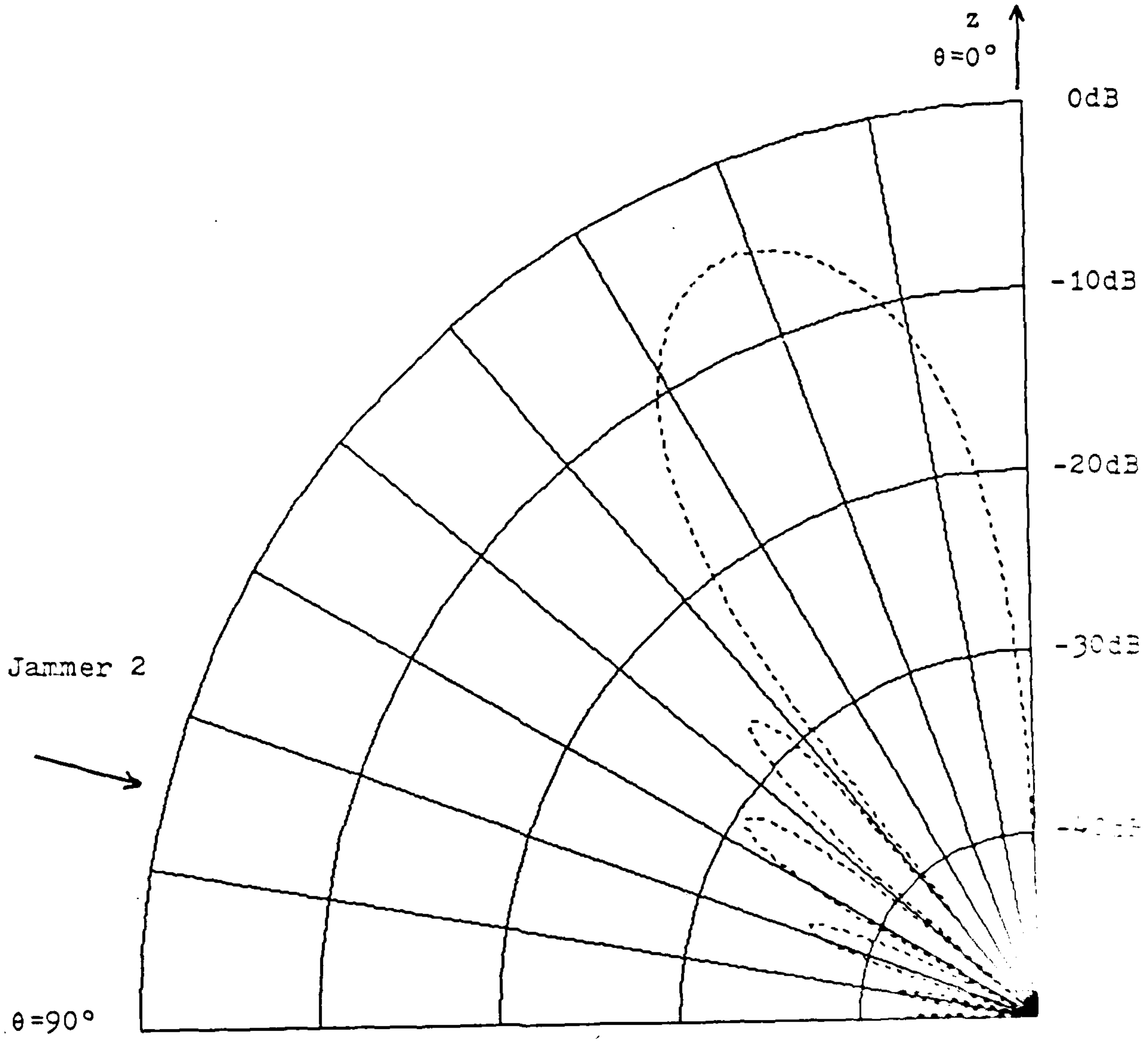
(Figure A)



For Rhomboid Geometry

Figure T6.11 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure B)



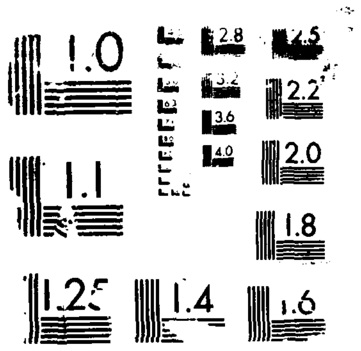
For Rhomboid Geometry

Figure T6.12 LMS-Adapted Pattern in Dir[illegible]

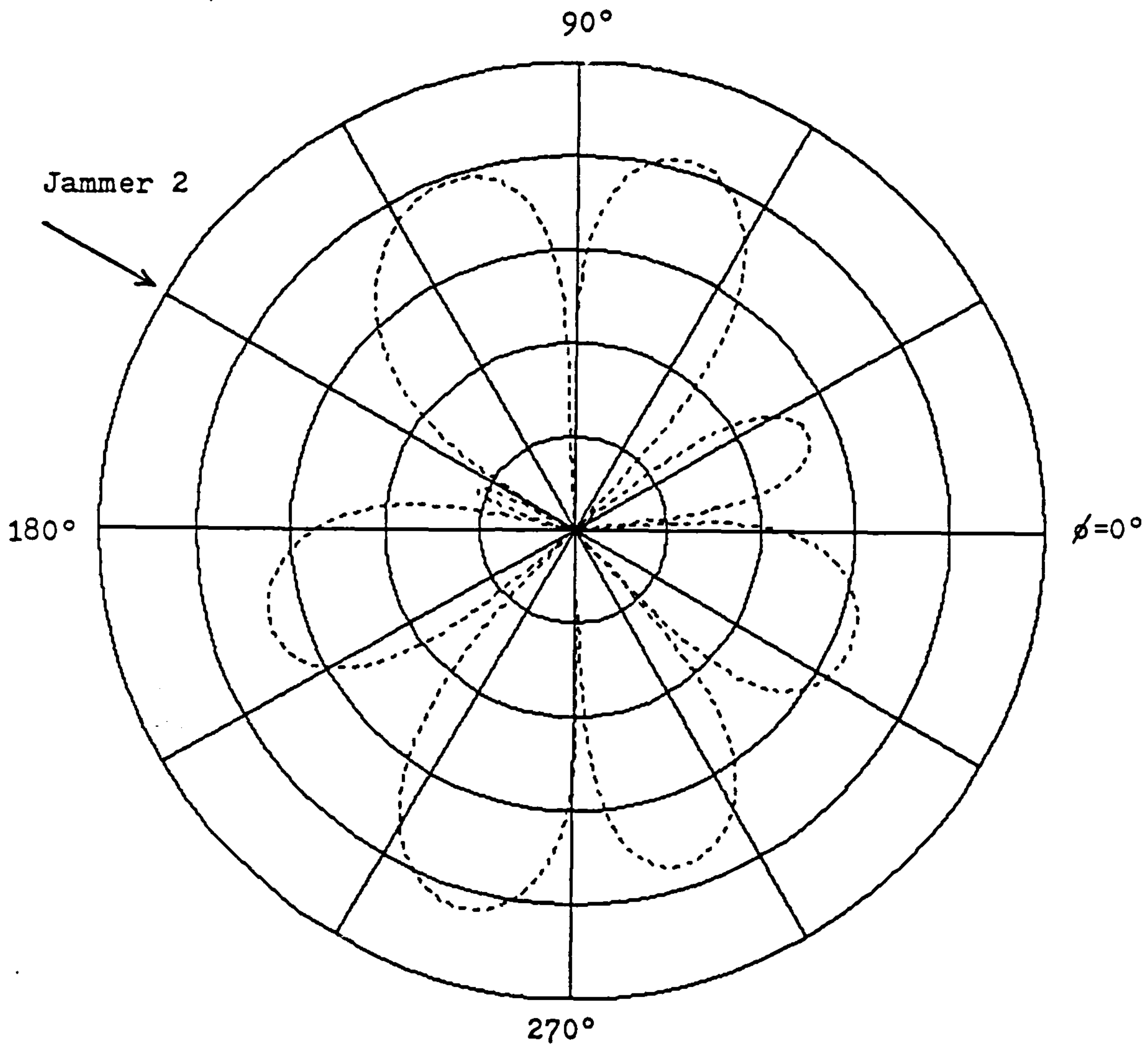
(Figure A)

AD-A189 329 ADAPTIVE ALGORITHMS FOR HF ANTENNA ARRAYS(U) KANSAS UNIV CENTER FOR RESEARCH INC LAWRENCE TELECOMMUNICATIO D HAEGERT ET AL JUL 87 TISL-5481 RADC-TR-86-159 3/3

UNCLASSIFIED F30602-81-C-0205 F/G 9/1 NL

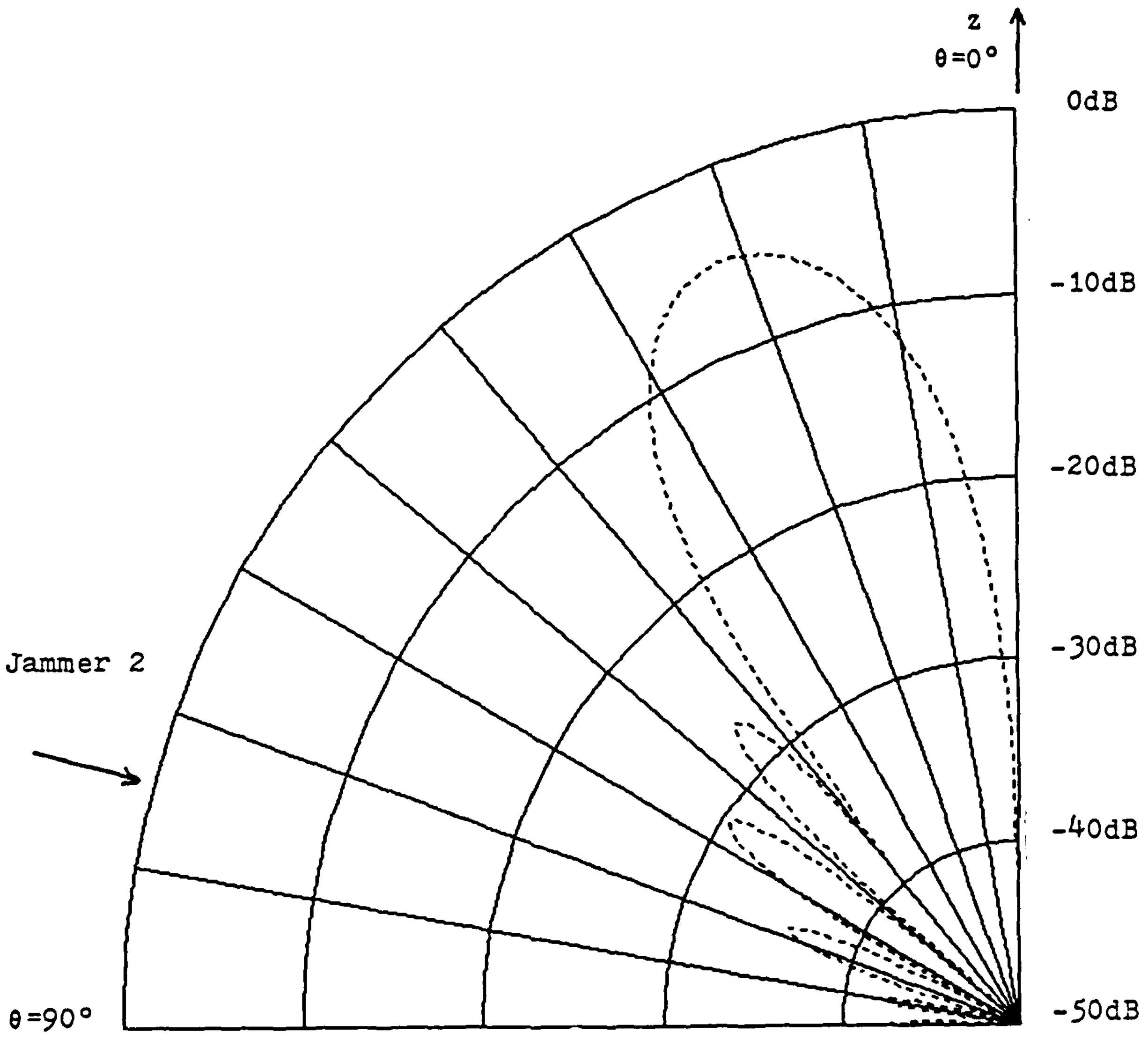


MICROCOPY RESOLUTION TEST CHART



For Rhomboid Geometry

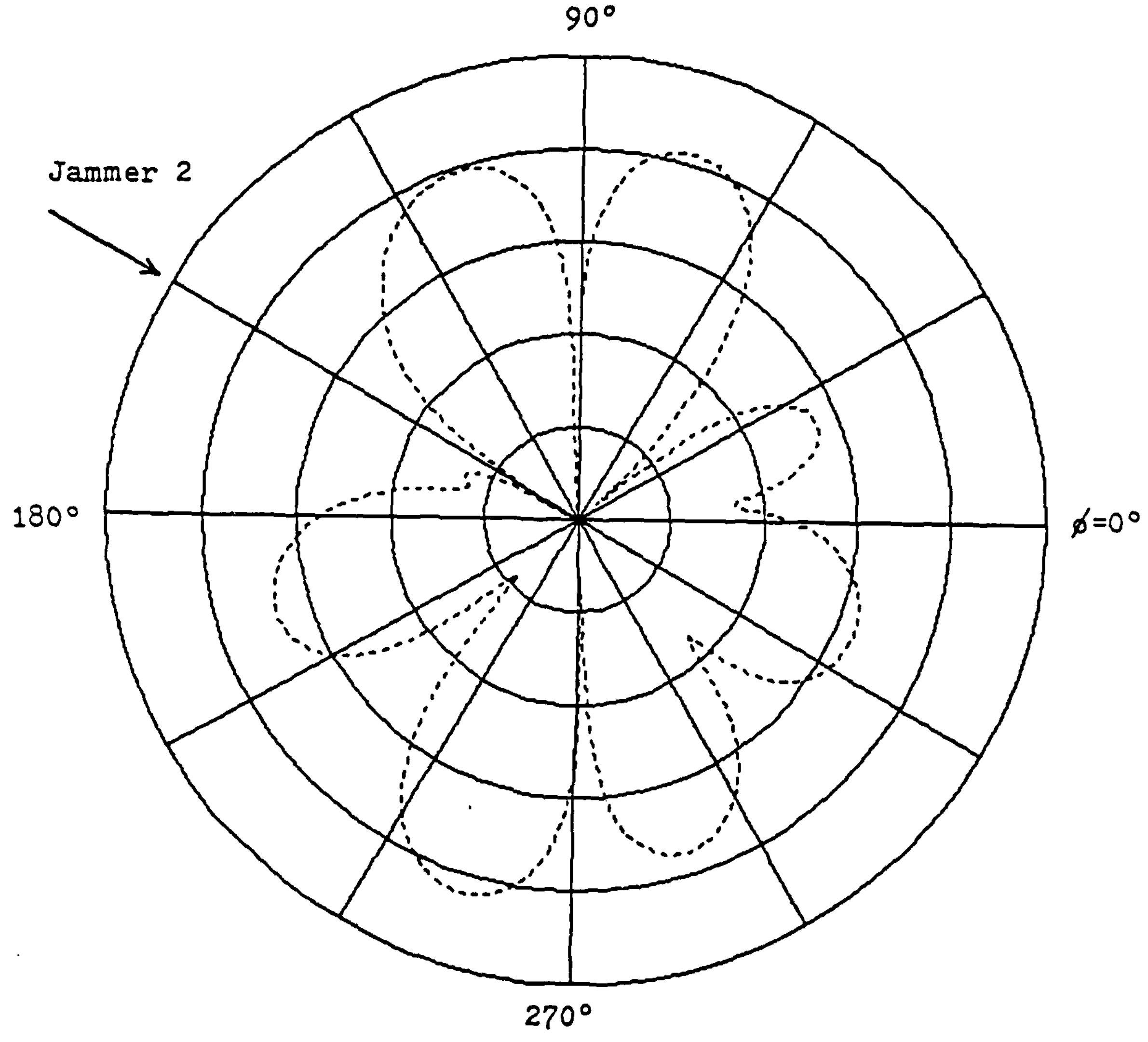
Figure T6.12 LMS-Adapted Pattern in Direction of Jammer 2
(Figure B)



For Rhomboid Geometry

Figure T6.13 Constrained LMS-Adapted Pattern in Direction of Jammer 2

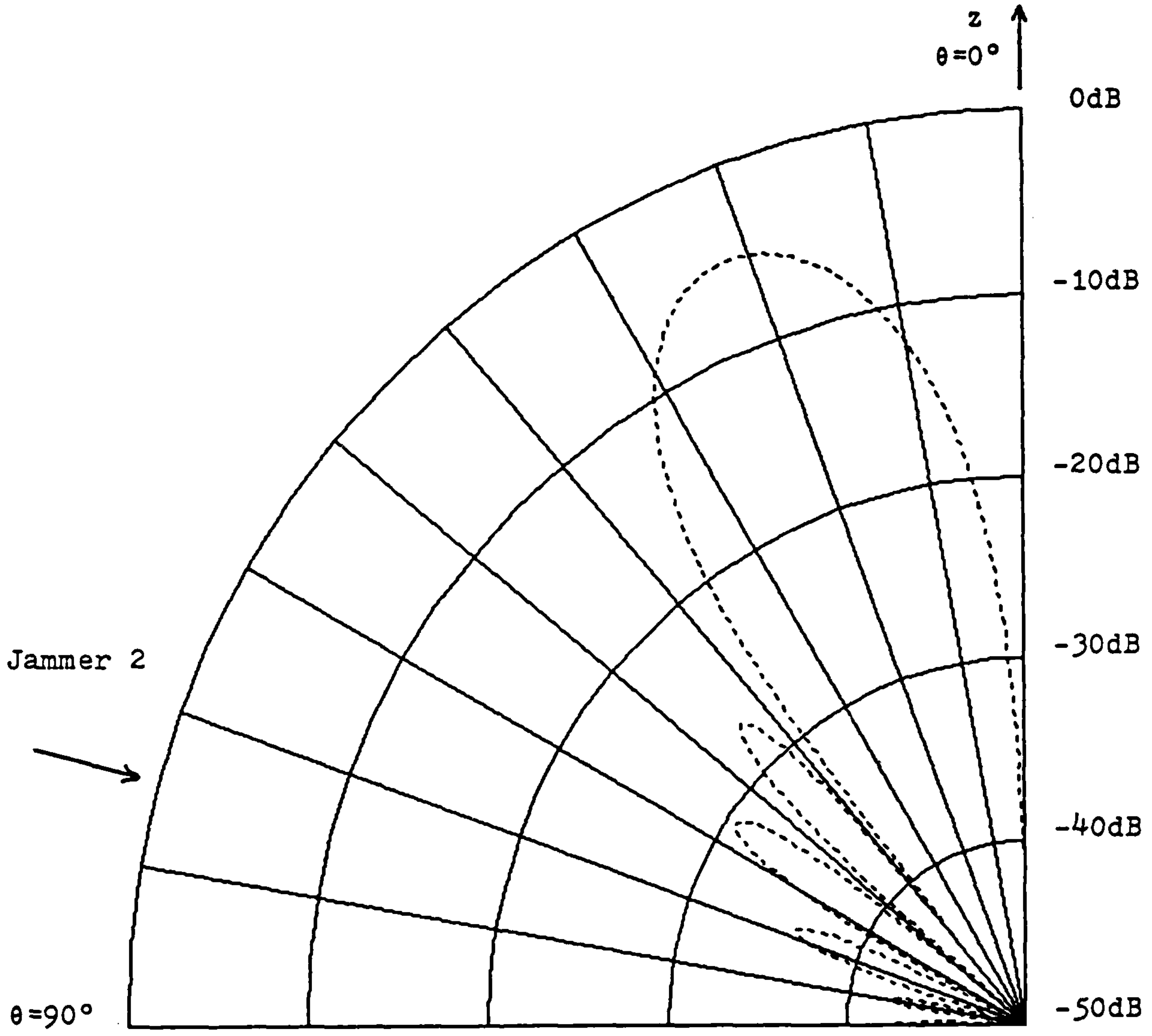
(Figure A)



For Rhomboid Geometry

Figure T6.13 Constrained LMS-Adapted Pattern in Direction of Jammer 2

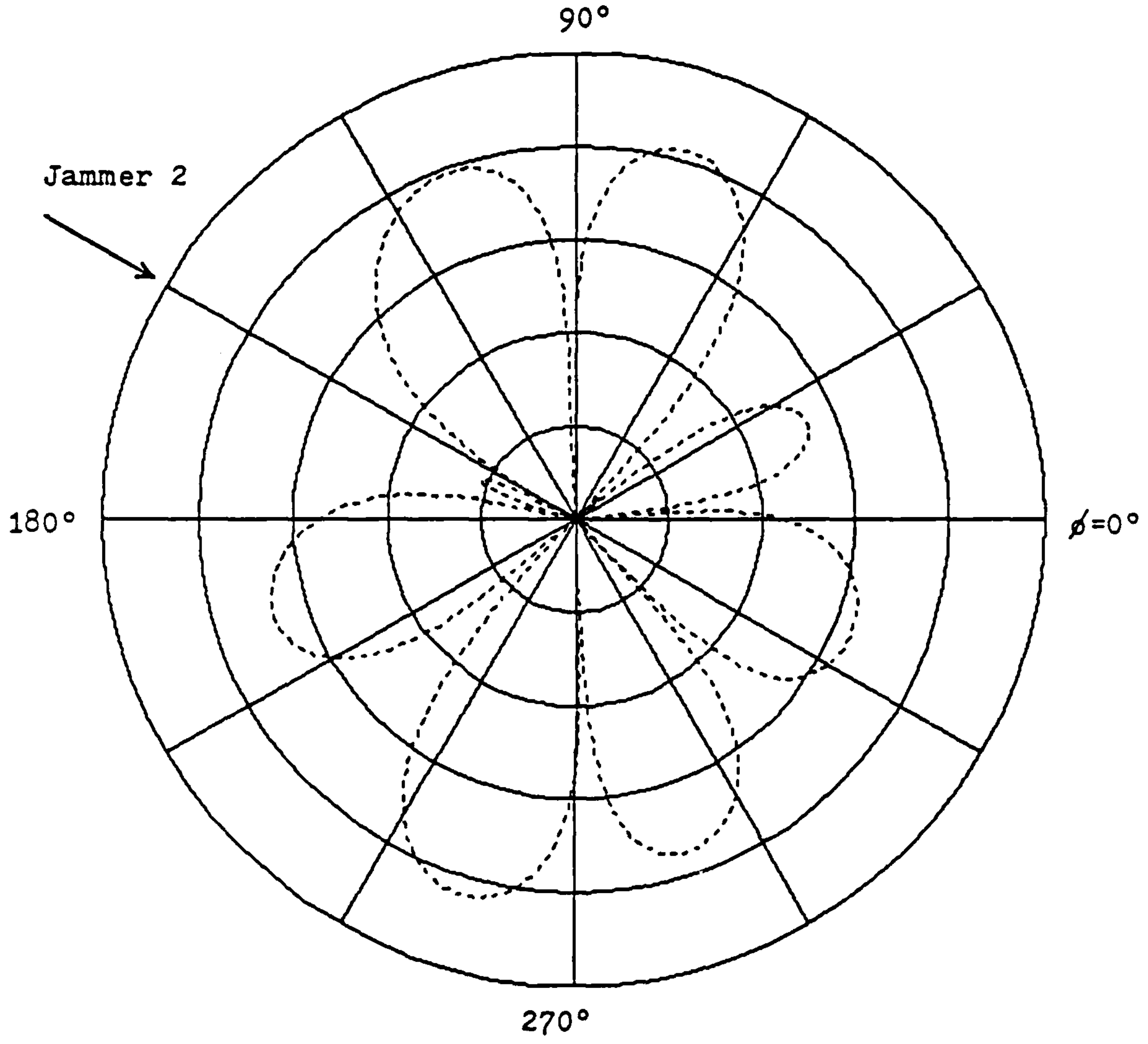
(Figure B)



For Rhomboid Geometry

Figure T6.14 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure A)



For Rhomboid Geometry

Figure T6.14 Update Covariance-Adapted Pattern in Direction of Jammer 2

(Figure B)

7.7 Conclusions and Recommendations

In all of the tests that were conducted, the Update Covariance algorithm consistently outperformed the others in terms of required iterations for convergence. It has also been shown that this can be an extremely misleading quantity, as the algorithm may require more actual computations to converge. Therefore, although the Update Covariance algorithm will converge in fewer iterations, it will undoubtedly require significantly more time to complete each iteration. As an example, assume that it was desired to build an array consisting of 36 elements, and that the algorithm must converge on a preamble 100 msecs. in duration. The Update Covariance algorithm would then require $N^3 + 3N^2 + 3N$ or approximately 50652 complex computations for each iteration. If the algorithm needs 100 iterations to converge, a grand total of 5065200 complex computations must be completed in 100 msecs. This allows approximately 19.74 nanoseconds to perform a complex computation. So despite the comparatively few iterations required, implementing the Update Covariance algorithm may place rigorous, if not unreasonable demands on the hardware. Even if this can be attained, it seems unreasonable to implement such a system when other algorithms can offer similar convergence times with much less complicated hardware. It was shown that the LMS algorithm could converge with fewer computations. And unlike the Update Covariance algorithm (which involves matrix arithmetic), the computations of the LMS algorithm can be performed simultaneously. In other words, the required 2N computations can be performed N at a time as the update of one weight is independent of the update of another. The required time for each iteration is therefore equal to the time it takes to perform 2 complex computations. This means that the actual realtime convergence for LMS algorithm is much less than that of the Update Covariance algorithm in all cases that have been presented.

The LMS algorithm is not without its drawbacks, however, as has been mentioned. Probably the most notable is the reference requirement that has been discussed. The channel model in this simulation posed not prohibitive difficulties for the algorithm and, if it is a good representation of what goes on in the HF channel as it is believed, the reference requirement should not be a major obstacle. The reference signal generation will, however, require precise synchronization between the arrival of the transmitted preamble and the preamble used as the reference. This may be a difficult task. If not, however, the simulation study indicates that the LMS algorithm would be an effective choice.

The simulation results also indicate that if a preamble is to be sent, the Constrained LMS algorithm, although capable of meeting the convergence requirements, would not be the best choice in light of its performance in comparison to the LMS algorithm in the presence of a signal. It also requires more computations for each iteration, although many of these computations can be performed simultaneously as well.

The performance of the Constrained LMS algorithm in the absence of the signal, however, introduces many interesting possibilities. TEST4 demonstrated that if preamble was not sent, the Constrained LMS algorithm would converge in approximately the same number of iterations as the LMS algorithm. Complicated synchronization and reference generation techniques could be avoided by simply not sending a preamble. Simply initiate the algorithm prior to any transmission. It may also be that the transmitted data had naturally occurring "breaks" in it. During either natural or deliberate breaks, the algorithm could be re-initiated, minimizing the effects of interferences. This would require some type of directional power sensing device. There are many interesting possibilities that could and should be considered. If the preamble and related synchronization devices are being generated solely for the benefit of the algorithm, one would have to consider the advantages of implementing the Constrained LMS algorithm and dispensing with the preamble. This would not seriously degrade the convergence performance, as evidenced by TEST4, and could possibly simplify the overall system.

REFERENCES

[1] B. Widrow, P. Mantey, L. Griffiths, and B Goode, "Adaptive Antenna Systems," Proceedings of the IEEE, Vol. 55, Dec. 1967.

[2] S.P. Applebaum, "Adaptive Arrays," IEEE Transactions on Antennas and Propagation, vol AP-24, no. 5, Sept. 1976.

[3] B. Widrow and J.M. McCool, " A Comparison of Adaptive Algorithms Based on the Method of Steepest Descent and Random Search," IEEE Transactions on Antennas and Propagation, vol AP-24, no. 5, Sept. 1976.

[4] L.J. Griffiths, "A Simple Algorithm for Real-time Processing in Antenna Arrays," Proceedings of the IEEE, vol. 57, Oct. 1969.

[5] O.L. Frost III, "An Algorithm for Linearly Constrained Adaptive Array Processors," Proceedings of the IEEE, vol. 60, no. 8, Aug. 1972.

[6] K. Takao, M. Fujita, and T. Nishi, "An Adaptive Antenna Array Under Directional Constraint," IEEE Transactions on Antennas and Propagation, vol AP-24, no. 5, Sept 1976.

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

[8] L.E. Brennan, J.D. Mallet, I.S. Reed,"Adaptive Arrays in Airborne MTI Radar," IEEE Transactions on Antennas and Propagation, vol AP-24, no. 5, Sept. 1976.

[9] M. Brobston, "Simulation of Recursive Processors for Adaptive Antenna Arrays," M.S. project report, Dept. of Electrical and Computer Engineering, Univ. of Kansas, 1981.

[10] John G. Proakis, "Digital Communication," McGraw-Hill, New York, 1983.

[11] R. T. Compton Jr., "An Adaptive Array in a Spread Spectrum Communication System," Proceedings of the IEEE, vol. 66, no. 3, March 1978.

[12] R. L. Reigler and R. T. Compton Jr.," An Adaptive Array for Interference Rejection," Proceedings of the IEEE, vol. 61, June 1973.

[13] P. Snow, " An Antenna Simulation for a Spread Spectrum System," M.S. project report, Dept. of Electrical and Computer Engineering, 1984.

APPENDIX A

COMPLEX LOWPASS EQUIVALENT REPRESENTATION

Digital information signals are often transmitted using some type of carrier modulation. Signals which have a bandwidth that is much smaller than the carrier frequency are known as narrowband bandpass signals. For convenience, it is desirable to reduce the bandpass signals to equivalent lowpass signals. The carrier component can be dispensed with because it carries no information.

A real-valued signal s(t) with a frequency concentrated in a narrow band of frequencies about the carrier frequency, f_c, can be expressed as

$$s(t) = a(t) \cos[2\pi f_c t + \Theta(t)]$$

where

$$a(t) = \text{amplitude of } s(t)$$
$$\Theta(t) = \text{phase of } s(t)$$

By expanding the cosine function in the above expression, a second representation is obtained. This is written as

$$\begin{aligned} s(t) &= a(t) \cos\Theta(t) \cos 2\pi f_c t - a(t) \sin\Theta(t) \sin 2\pi f_c t \\ &= I(t) \cos 2\pi f_c t - Q(t) \sin 2\pi f_c t \end{aligned}$$

where

$$I(t) = a(t) \cos\Theta(t)$$
$$Q(t) = a(t) \sin\Theta(t)$$

The frequency content of I(t) and Q(t) is concentrated at low frequencies. These are lowpass signals.

Finally, define the complex envelope u(t) as

$$u(t) = I(t) + jQ(t)$$

so that

$$s(t) = \mathrm{re}[u(t)\, e^{j2\pi f_c t}]$$

Therefore, a real bandpass signal s(t) is completely specified by its complex envelope, u(t), if its carrier frequency is known.

APPENDIX B

USER MANUAL FOR HF ADAPTIVE ANTENNA ARRAY EVALUATION FACILITY

The following is a brief description of the parameters and procedures needed to operate the simulation. The operations are divided into an input and an output phase, and will be described separately.

B1.0 Input Procedures

As shown in Figure B-1, the input phase is initiated with the command @INPUT. This command file asks the user to input the name of the experiment to be performed, and in this case we have chosen TEST1. The result of this is the creation of a new subdirectory which is given the name EXPERIMENT TEST1. This subdirectory, then, will provide a location for the simulation run, and will contain all important output files produced. This, however, is completely transparent to the user.

Again referring to Figure B-1, we see that as soon as the user determines the name of the experiment, he or she is immediately introduced to the first menu of the actual input program. It is this routine which creates the data file which is subsequently read by the simulation mainline. Before explaining the actual variables appearing in this and the remaining menus, we first consider the methods by which variables are input and new menus acquired.

Concentrating again on the first menu, we see that 4 variables appear (numbered 1 through 4). Assume we wish to change the number of desired convergences from the default value of 5 to a value of 20. As shown in the Figure, this is accomplished simply by typing the variable index (4) followed by a space and the new value (20). The program then returns the instructions < NEXT ENTRY PLEASE (0 TO REVIEW PAGE, -1 TO LEAVE PAGE > and as shown, 0 redisplays the menu with the new value inserted. A -1 at this point (instead of a 0) simply moves to the next menu without displaying the change. Of course, if it is desired to change nothing, it is possible to traverse the entire program simply by typing a -1 after each menu. The data file, then, will simply contain the default values. It should be noted that this program continuously updates the default file. That is to say that if a variable is changed on a run through the input program, it becomes the new default value for the next run.

Now that we may move through the input program, the variables themselves will be explained. It will be expedient to cover the variables one menu at a time since, in many cases, the variables within a menu are closely related. For the following discussion, refer to Figures B-1 through B-10.

B1.1 Menu 1: Simulation Parameters

These parameters are general and serve to set up the simulation at its most basic level. Here, many of the variables are self-explanatory.

1. Number of samples per bit. Sets up sample rate.
2. Signal to Noise Ratio (dB). This is the thermal S/N ratio which is used to simulate random electrical noise.
3. Simulation Bit Rate. Used to determine the Nyquist bandwidth of the information signal.
4. Number of simulation convergences desired. To arrive at a statistical evaluation of the simulation, the system is forced to converge a number of times. This variable simply determines the number of loops (value = 100).

B1.2 Menu 2: Adaptive Algorithm Options

Here the user is asked to choose the type of algorithm which will control the simulation.

1. Least Mean Squared Algorithm
2. Constrained Lease Mean Squared Algorithm
3. Update Covariance Algorithm

B1.3 Menu 3: Antenna Array Parameters

This menu gives the user several options to construct the antenna array to be used in the simulation. The array elements are dipole antennas (whip above ground).

1. Number of antenna elements (maximum = 9 elements)
2. Number of incoming signals. This parameter is simply the total number of impinging signals, including both the desired signal as well as one or two jammers.
3. Dipole equivalent element length (meters). The length of an equivalent dipole has been assumed to be twice the length of the actual whip antennas.
4. Carrier frequency of friendly communicator (kHz). This is simply the frequency, in kilohertz, of the desired signal. The frequency is used to determine the receiving characteristics of the dipoles.

B1.4 Menu 4: Jammer to Signal Ratios (dB)

Menu 4 is variable in length as dictated by the number of signals entered earlier. Each jammer to signal ratio may be specified independently and are used to adjust the jammer powers.

B1.5 Menu 5: Relative Antenna Element Locations (meters)

Here, the user is allowed to define the actual geometry of the array by specifying the x and y coordinates of each element. This menu, again, is variable in length depending on the number of elements entered earlier.

B1.6 Menu 6: Azimuth (PHI) and Elevation (THETA) Coordinates of Incoming Signals

The user is prompted for the arrival angles of the incoming signals (in degrees). It is always assumed that the friendly communicator is signal #1. The azimuth and elevation angles are specified in terms of the spherical coordinates PHI and THETA. Phi determines the azimuthal coordinate and is defined to be 0 degrees on the x-axis and increasing toward the y-axis. Theta, on the other hand, determines the elevation and is defined to be 0 degrees on the z-axis and increasing toward the x-y plane. Note that in our case, it is only meaningful if theta is in the range from 0° to 90° as this corresponds to the space above ground.

B1.7 Menu 7: HF Channel Parameters

This is just the number of different paths to be simulated with the HF channel. It essentially chooses the number of taps in the tapped delay line model employed by the channel.

B1.8 Menu 8: Delays of Each Propagation Mode - (mS)

Menu 8 is variable in length depending on the number of channel modes entered in menu 7. Here the delay of each path is given in milliseconds and is used in the HF channel model to simulate dispersiveness. Note that the first delay is assumed to be zero, and the others are simply relative to the first. Also, it should be noticed that the path delays should be integer multiples of (1/Nyquist Rate). This is necessary, again, due to the implementation of the HF channel model.

B1.9 Menu 9: Attenuation of Each Propagation Mode (dB)

The length of this menu is determined, again, by the number of modes. The HF channel model contains multipliers which allow signals emerging from different paths to be attenuated separately. This menu lets the user assign a different attenuation to each path.

B.10 Menu 10: Adaptive Algorithm Convergence Constant

The convergence constant is used by the LMS and constrained LMS algorithms to dictate the update step sizes as convergence is taking place. This constant is quite crucial to the convergence time and overall character of the convergence, but is also very difficult to obtain. For the most part, only a trial and error type search will produce the optimum value. A rough estimated value is calculated by the program, but it should not be trusted too far. The default value may be repeatedly changed until the simulation model is executed. Note that only 5 decimal places are provided by the input program. The convergence value may be specified out to 9 decimal locations, but unfortunately only 5 are displayed. The actual value may be viewed simply by listing the parameter file.

At this point, as can be seen from Figure B-10, once the convergence constant is specified, the input program automatically submits the simulation in batch. The important files generated by the mainline are stored, as was mentioned earlier, in the user defined directory EXPERIMENT_TEST1. When the simulation is complete, it is then possible to use the output programs to view the resulting data. The output phase will be discussed next.

B2.0 Output Procedures

The purpose of this section is to explain the use of the output programs which allow both a review of the input parameters, as well as the results, of any test.

To invoke the output procedures, simply type @OUTPUT while stationed at a graphics terminal. The program responds with the question, "For which experiment do you wish to see the output?" After the response is given, which in our case is TEST1, the program returns the menu shown in Figure B-11. We see that there are 8 choices of output, the first 3 simply being a review of the input parameters of the test, while the remaining choices are actual data output from the run. To explain the use of this menu, we will step through it one option at a time.

B2.1 Option 1: Review Antenna Parameters

To invoke this or any other option, simply type the corresponding number. Thus, in this case, after a 1 is typed, a screen similar to Figure B-12 will appear. This screen reminds us of the antenna geometry which was used in the test as well as the carrier frequency and equivalent dipole lengths of the antennas. Also displayed are the amplitude and phase distributions for each element resulting from the adaptation. To return, then, to the main menu from this option, simply type <return> and Figure B-11 will again appear.

B2.2 Option 2: Review Channel Information

The second screen which may be viewed, shown in Figure B-13, is also a review of parameters, but here it is the channel information which is displayed. Several other input quantities are also listed such as data rate, type of algorithm, number of convergences, and the convergence constant. Again, to return to the main menu, just type <return>.

B2.3 Option 3: Review Incoming Signal Information

As shown in Figure B-14, this screen lets the user review the arrival angles of the friendly signal as well as that of the jammers. Also, the jammer to signal ratios are displayed for each of the jammers.

B2.4 Option 4: View Calculated Results

This screen, arrived at by typing 4 at the main menu level, is shown in Figure B-15. This is the first screen that actually returns some data from the simulation run. The values which appear here are the average number of iterations to converge, as well as the average final signal/interference ratio. The variance of these quantities is also given, to show the user the amount of spread which has resulted in the values.

B2.5 Option 5: View Antenna Plot

Typing 5 at the main menu level allows the user to examine the antenna plots which have resulted from the simulation. This is probably the most useful portion of the output program, as it allows immediate conformation of the algorithm performance. Actually when Option 5 is chosen in the main menu, a new menu appears as shown in Figure B-16. With this menu at hand, the user is given the capability of examining any cut of the antenna pattern simply by changing the menu entries. To better understand the capabilities, we will step through each option of the sub-menu.

B2.5.1 Sub-Option 1: 1) Min Convergence Time 2) Avg. 3) Max.

When the simulation is finished, the convergence iterations of all convergences are examined and the corresponding antenna weights of the slowest, average, and fastest convergences are written to a file. With this option, it is possible to choose which set of weights will be used in computation ofthe antenna patterns. These are very similar, however, and usually indistinguishable in graph form. The default value of this parameter is set to the average.

B2.5.2 Sub-Option 2: Fixed Angle PHI for Elevation Plot

Here, the user is allowed to choose a fixed azimuth angle in order to produce an elevation plot. In other words, PHI is held fixed and THETA is allowed to vary over the range 0 to 180 degrees. For example, if PHI is equal to 0, then the resulting elevation plot will exist in the x-z plane.

B.2.5.3 Sub-Option 3: Fixed Angle THETA for Azimuth Plot

This option is very similar to the previous one in this menu, but here it is THETA which we fix instead of PHI. This value is used to produce an azimuth plot at some fixed elevation. For example, if THETA is 90 degrees, the resulting azimuth plot would exist in the x-y plane. For values other than 90 degrees, it should be realized that the resulting pattern does not exist in a plane, but is simply the projection onto a plane of the field values.

B2.5.4 Sub-Option 4: Slice Type 1) Elevation 2) Azimuth

Finally, the user must make a choice to see either an elevation or an azimuth plot. If one chooses, for example, to see an elevation plot, the PHI coordinate is set to the value dictated by Option 2, while THETA is varied to form the pattern. Thus, once elevation or azimuth are chosen, then either the Option 2 or the Option 3 values (respectively) are used; never both.

To actually see an antenna plot, this menu is exited by typing -1 as in the other menu programs. For the default values of Figure B-16, this produces the plot as shown in Figure B-17. Note that for both azimuth and elevation, 0 degrees is to the right on the plot. Thus for an azimuthal plot, right implies the x-axis, while for elevation, it implies THETA = 0 (z-axis).

At the bottom of the screen on each antenna plot, the program asks whether one wishes to see more antenna plots. If the response is yes then control is returned to the menu of Figure B-16. On the other hand, if the answer is no, the user is returned to the main menu (Figure B-11).

B2.6 Option 6: View Histogram of Convergence Counts

Upon choosing this option, one is able to look at the distribution of convergence iterations in histogram form. This is shown in Figure B-18. The bins are fixed in size and equal to 50. This value works very well for the LMS and constrained LMS, but is slightly large for the update covariance algorithm. The fixed value of 50 was chosen simply to provide ease of use.

B2.7 Option 7: View S/N Ratio vs. Time Plot

This option produces a plot of the signal to noise ratio versus the number of iterations. An example of this is shown in Figure B-19. This plot is always stopped at 6250 iterations in order to display a reasonable number of convergences while still providing an acceptable resolution. The purpose of the graph is simply to show the convergence characteristics of the test.

B2.8 Option 8: View Error Signal vs. Time Plot

The final available plot is shown in Figure B-20, and represents the magnitude of the error signal used by the LMS algorithm for adaptation. This option may only be invoked if the controlling algorithm of the test is the LMS, however. It is the only algorithm which makes use of an error signal, and consequently the mainline only produces the output file for that case. This plot is also stopped at 6250 iterations to get a reasonable number of convergences while still providing space for clarity.

```
$ @input
What is the name of this experiment?: test1
                                   SIMULATION PARAMETERS

 1  NUMBER OF SAMPLES PER BIT                                            16.00000

 2  SIGNAL TO NOISE RATIO (dB)                                           10.00000

 3  SIMULATION BIT RATE (bps)                                           300.00000

 4  NUMBER OF SIMULATION CONVERGENCES DESIRED                             5.00000

4 20

NEXT ENTRY PLEASE (0 TO REVIEW THE PAGE, -1 TO LEAVE THE PAGE)
0
                                   SIMULATION PARAMETERS

 1  NUMBER OF SAMPLES PER BIT                                            16.00000

 2  SIGNAL TO NOISE RATIO (dB)                                           10.00000

 3  SIMULATION BIT RATE (bps)                                           300.00000

 4  NUMBER OF SIMULATION CONVERGENCES DESIRED                            20.00000
```

Figure B-1 Menu #1 of input

```
-1
                                 ADAPTIVE ALGORITHM OPTIONS

 TYPE=1    ==> LMS

 TYPE=2    ==> CONSTRAINED LMS

 TYPE=3    ==> UPDATE COVARIANCE ALGORITHM

 1  ADAPTIVE ALGORITHM TYPE (1,2, OR 3)                                   1.00000

-1
```

Figure B-2 Menu #2 of input

```
                    ANTENNA ARRAY PARAMETERS

  1   NUMBER OF ANTENNA ELEMENTS                                   4.00000
  2   NUMBER OF INCOMING SIGNALS (DESIRED + JAMMERS)               3.00000
  3   DIPOLE EQUIVALENT ELEMENT LENGTH-(m)                        36.57600
  4   CARRIER FREQUENCY OF FRIENDLY COMMUNICATOR-(kHz)         30000.00000
 -1
```

Figure B-3   Menu #3 of input

```
                       JAMMER TO SIGNAL RATIOS
                               (dB)

  1   J/S RATIO FOR JAMMER  1                                     20.00000
  2   J/S RATIO FOR JAMMER  2                                     20.00000
```

Figure B-4   Menu #4 of input

```
                    RELATIVE ANTENNA ELEMENT LOCATIONS
                           MEASURED IN METERS

 1 X COORDINATE OF ELEMENT 1                                   10.00000
 2 Y COORDINATE OF ELEMENT 1                                   10.00000
 3 X COORDINATE OF ELEMENT 2                                    5.00000
 4 Y COORDINATE OF ELEMENT 2                                   10.00000
 5 X COORDINATE OF ELEMENT 3                                    0.00000
 6 Y COORDINATE OF ELEMENT 3                                   10.00000
 7 X COORDINATE OF ELEMENT 4                                   10.00000
 8 Y COORDINATE OF ELEMENT 4                                    5.00000
 9 X COORDINATE OF ELEMENT 5                                    5.00000
10 Y COORDINATE OF ELEMENT 5                                    5.00000




-1
                    RELATIVE ANTENNA ELEMENT LOCATIONS
                           MEASURED IN METERS

11 X COORDINATE OF ELEMENT 6                                    0.00000
12 Y COORDINATE OF ELEMENT 6                                    5.00000
13 X COORDINATE OF ELEMENT 7                                   10.00000
14 Y COORDINATE OF ELEMENT 7                                    0.00000
15 X COORDINATE OF ELEMENT 8                                    5.00000
16 Y COORDINATE OF ELEMENT 8                                    0.00000
17 X COORDINATE OF ELEMENT 9                                    0.00000
18 Y COORDINATE OF ELEMENT 9                                    0.00000
-1
DO YOU WANT TO REVIEW ALL THE PARAMETERS? (Y/N)
```

Figure B-5 Menu #5 of input

```
n
AZIMUTH (PHI) AND ELEVATION (THETA) COORDINATES OF INCOMING SIGNALS
PHI=0 IMPLIES THE X-AXIS. PHI INCREASES TOWARD THE Y-AXIS.
THETA=0 IMPLIES THE +Z-AXIS. (INCREASES TOWARD THE X-Y PLANE)
SIGNAL #1 IS ASSUMED TO BE THE FRIENDLY COMMUNICATOR.

 1 PHI COORDINATE OF SIGNAL  1                                   90.00000
 2 THETA COORDINATE OF SIGNAL  1                                 60.00000
 3 PHI COORDINATE OF SIGNAL  2                                   90.00000
 4 THETA COORDINATE OF SIGNAL  2                                 80.00000
 5 PHI COORDINATE OF SIGNAL  3                                  150.00000
 6 THETA COORDINATE OF SIGNAL  3                                 75.00000
```

Figure B-6   Menu #6 of input

```
                         HF CHANNEL PARAMETERS

 1  NUMBER OF MODES OF PROPOGATION (TAPS)                         3.00000
-1
```

Figure B-7   Menu #7 of input

```
                       DELAYS OF EACH PROPOGATION MODE
                          MEASURED IN MILLISECONDS
 ** SHOULD BE INTEGER MULTIPLES OF 1/(NYQUIST RATE)**

 1  DELAY OF PATH NUMBER  1                                       0.00000
 2  DELAY OF PATH NUMBER  2                                       0.00000
 3  DELAY OF PATH NUMBER  3                                       1.00000
```

Figure B-8   Menu #8 of input

```
                ATTENUATION OF EACH PROPOGATION MODE
                                (dB)

 1  ATTENUATION OF PATH NUMBER  1                                  0.00000
 2  ATTENUATION OF PATH NUMBER  2                                  0.00000
 3  ATTENUATION OF PATH NUMBER  3                                  0.00000
-1
```

Figure B-9 Menu #9 of input

```
   ESTIMATED CONVERGENCE CONSTANT=    1.0888889E-05
        ADAPTIVE ALGORITHM CONVERGENCE CONSTANT

 1  DEFAULT CONVERGENCE CONSTANT VALUE                               0.00001

-1
Job MAIN (queue SYS$BATCH, entry 740) started on SYS$BATCH
$
```

Figure B-10 Menu #10 of input

```
OUTPUT- The Main Menu

 1. Review antenna parameters
 2. Review channel information
 3. Review incoming signal information
 4. View calculated results
 5. View antenna plot
 6. View histogram of convergence counts
 7. View S/N vs. time graph
 8. View error signal vs. time graph
-1. End output

Which?
```

Figure B-11   Main menu of output

```
        Antenna Element Parameters

element          x        y           amplitude      phase(in radians)
   1         10.00    10.00             0.31           2.19
   2          5.00    10.00             0.43           2.08
   3          0.00    10.00             0.31           1.99
   4         10.00     5.00             0.03          -0.99
   5          5.00     5.00             0.12           3.55
   6          0.00     5.00             0.03           1.83
   7         10.00     0.00             0.31          -1.15
   8          5.00     0.00             0.43          -1.24
   9          0.00     0.00             0.31          -1.36

Carrier Frequency (in Hz) :     30000000.0
Antenna Length (in meters):           36.6
```

Figure B-12   Antenna parameters

```
          Channel Information
Data Rate (in bits/sec)  :          300
Number of Samples/Bit    :           16
Algorithm used:                     LMS
Number of Convergences   :           20
The Convergence Constant:          0.00001493
Channel S/N ratio(in dB):         10.00  .

                  Delay Information
    delay(in msec)            path attenuation (in dB)
       0.000                         0.00
       0.333                         0.00
       1.667                         0.00
```

Figure B-13   Channel and other assorted information

```
                Incoming Signal Information

signal        phi(in degrees)  theta(in degrees)  J/S(in dB)
friendly           90.00             60.00
jammer  1          90.00             80.00           20.000
jammer  2         150.00             75.00           20.000
```

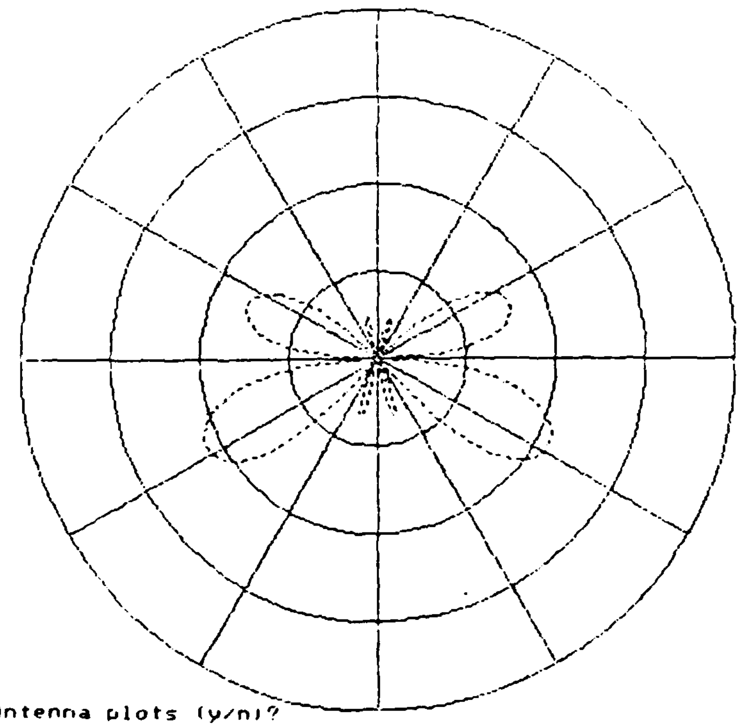
Figure B-14 Signal information

```
            Calculated Results

Average Number of Iterations:          326.00
     Variance               :        13026.32

Average Signal/Interference (in dB):   13.83
     Variance               :           2.13
```

Figure B-15 Calculated results of output

```
                    SLICE DESCRIPTION
      After exiting this menu, the graph data will be
        calculated and then plotted in polar form.

1 1) min. convergence time 2) avg.  3) max                        2.00000
2 Fixed angle Phi for elevation plot (in deg.)                    0.00000
3 Fixed angle Theta for azimuth plot (in deg.)                   45.00000
4 Slice type: 1)Elevation 2)Azimuth                               1.00000
```

Figure B-16   Antenna pattern menu



```
More Antenna plots (y/n)?
```

Figure B-17   Antenna plot from Menu 5

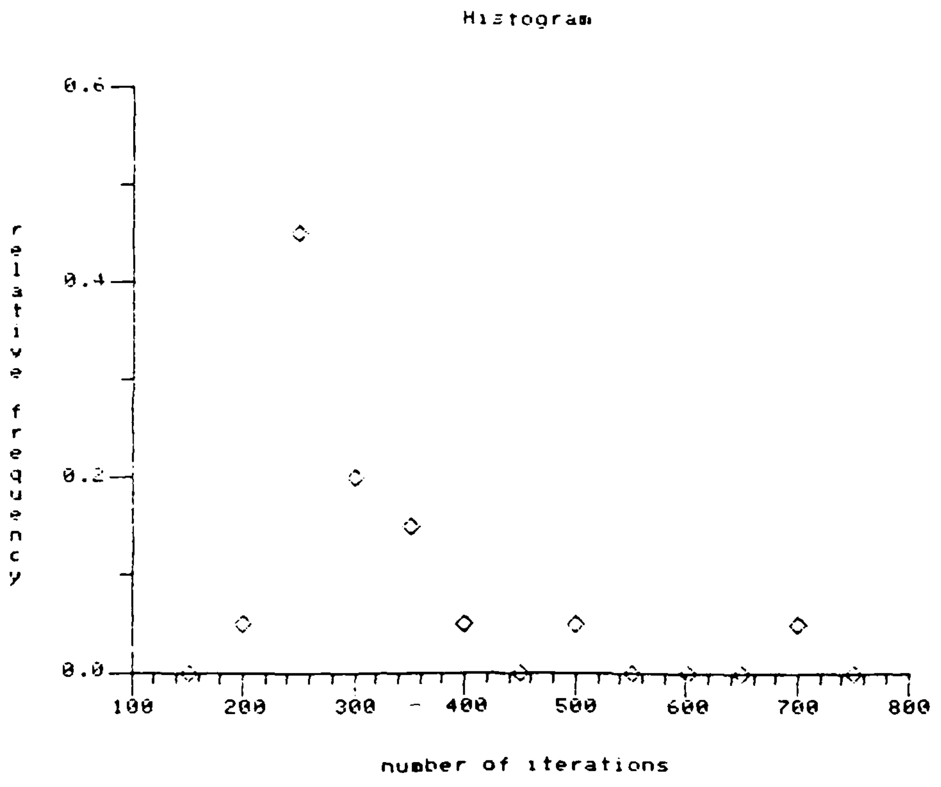


Figure B-18 Convergence histogram

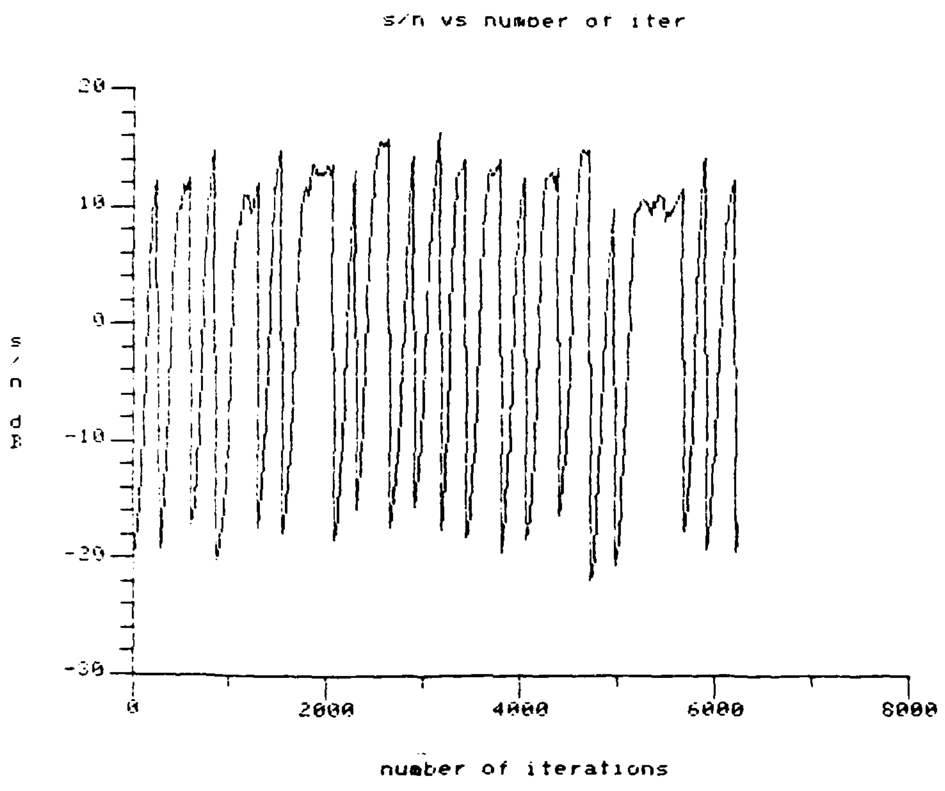


Figure B-19 Signal to noise plot

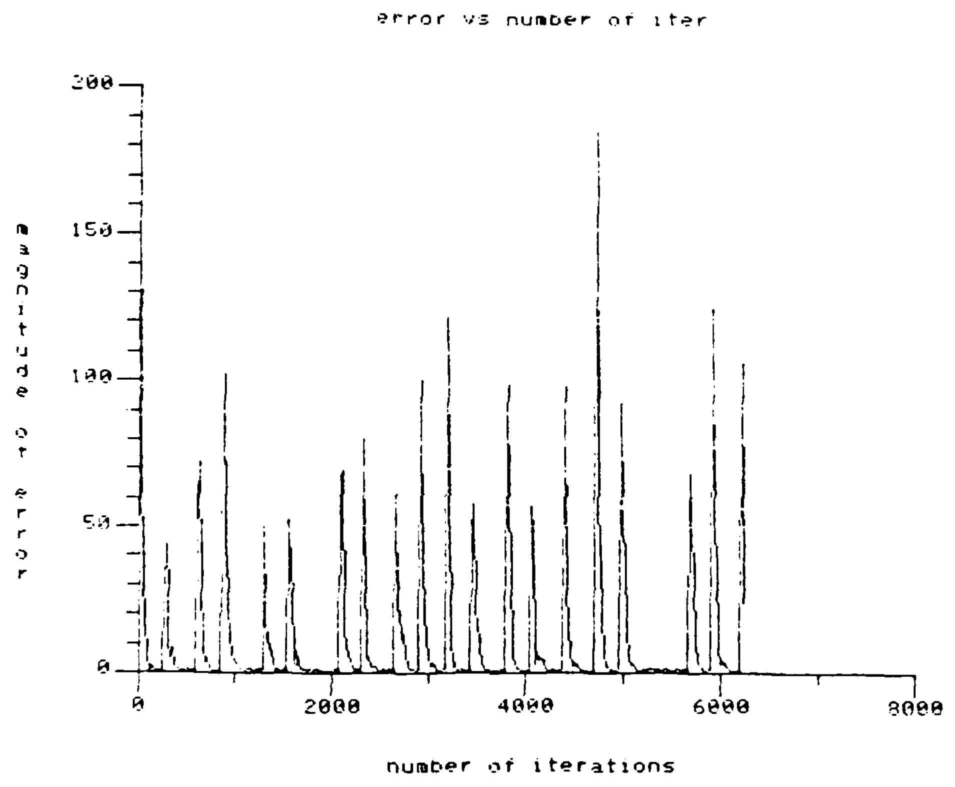


Figure B-20 Error signal plot

MISSION
of
Rome Air Development Center