T · H · E
OHIO
STATE
UNIVERSITY

An Experimental SMI Adaptive Antenna Array for Weak Interfering Signals

R.L. Dilsavor and I.J. Gupta

The Ohio State University

ElectroScience Laboratory

Department of Electrical Engineering
Columbus, Ohio 43212

Technical Report 716111-7
Grant No. NAG3-536
October 1989

National Aeronautics and Space Administration
Lewis Research Center
21000 Brookpark Road
Cleveland, Ohio 44135

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.

4. Title and Subtitle
An Experimental SMI Adaptive Antenna Array for Weak Interfering Signals

5. Report Date
October 1989

6.

7. Author(s)
R.L. Dilsavor and I.J. Gupta

8. Performing Org. Rept. No.
716111-7

9. Performing Organization Name and Address
The Ohio State University
ElectroScience Laboratory
1320 Kinnear Road
Columbus, OH 43212

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C)
(G) NAG3-536

12. Sponsoring Organization Name and Address
National Aeronautics and Space Administration
Lewis Research Center, 21000 Brookpark Road
Cleveland, Ohio 44135

13. Report Type/Period Covered
Technical Report

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

A modified SMI algorithm designed to increase the suppression of weak interference is implemented on an existing experimental array system. The algorithm itself is fully described as are a number of issues concerning its implementation and evaluation, such as sample scaling, snapshot formation, weight normalization, power calculation, and system calibration. Several experiments show that the "steady state" performance (i.e. many snapshots are used to calculate the array weights) of the experimental system compares favorably with its theoretical performance. It is demonstrated that standard SMI does not yield adequate suppression of weak interference. Modified SMI is then used to experimentally increase this suppression by as much as 13dB.

17. Document Analysis a. Descriptors

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement
A. Approved for public release;
Distribution is unlimited.

19. Security Class (This Report)
Unclassified

20. Security Class (This Page)
Unclassified

21. No. of Pages
91

22. Price

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77)
Department of Commerce

Contents

List of Figures

Chapter 1

Introduction

Adaptive antenna arrays have time-variable, controllable array patterns which enables them to suppress interference from moving sources of interference while receiving a desired signal whose source may also be moving. The output of an adaptive array is the sum of the weighted (in amplitude and phase) outputs of a number of antennas or array elements as shown in Figure 1.1. The pattern of the array at any time is a function of the weight values at that time. The weights may be controlled according to some closed-loop feedback control law as in the LMS array or Applebaum array or they may be varied according to an open-loop weight assignment rule as in the SMI array [1]. In any case, an array must be given information that allows it to distinguish the desired signal from the interference signals to be suppressed. For the Applebaum and SMI arrays, the information consists of the desired signal direction, whereas for the LMS array, a reference signal that is correlated with the desired signal must be derived. Given good information about the desired signal, each of the three array types mentioned above is known to maximize the output signal-to-

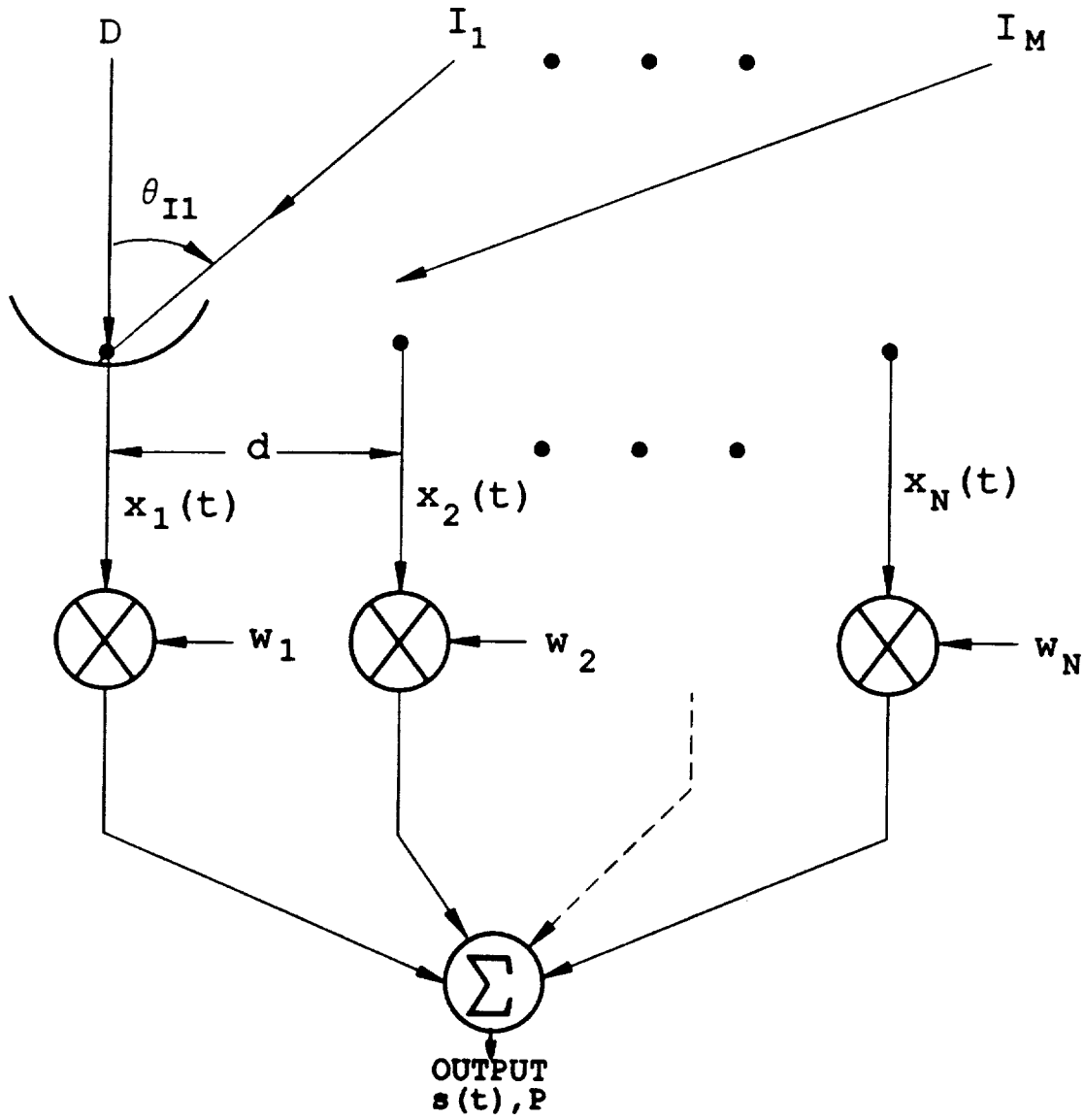


Figure 1.1: An N-element array receiving a desired signal and M interference signals.

interference-plus-noise ratio (SINR) under steady state conditions [1]. As a result, when the interference power is weak compared to the noise power the weights adapt to maximize the signal-to-noise ratio and the interference is left more or less unsuppressed.

The problem of suppressing weak interference has been recently addressed by Gupta and Ksienski [2] who proposed a hardware modification of the feedback loops which control the array weights. In this modification, two spatially separate antennas followed by their individual amplifiers are used with each feedback loop to yield decorrelated noise at the inputs of the feedback loop correlators. It was shown that, in theory, the modified feedback loops can provide the required interference suppression. Ward et. al. ([3],[4]) built an experimental adaptive array and verified the theoretical analysis.

Later, as an alternative, Gupta [5] proposed a software modification of the SMI array to enhance weak interference suppression. In the modified SMI algorithm the sample covariance matrix is redefined to subtract out the effects of the thermal noise. This is done by subtracting a fraction F of the minimum eigenvalue of the original covariance matrix from its diagonal elements. This step is justified by noting that, in theory, the only effect of thermal noise (assuming the noise is uncorrelated, zero mean, and has equal power σ^2 at each array element) on the covariance matrix is an additive σ^2 term on each diagonal element of the true covariance matrix. Furthermore, if the number of elements in the array is greater than the number of signals incident on the array, the minimum eigenvalue of the sample covariance matrix is an estimate of the noise power σ^2. Gupta [5]

showed that, in theory, the modified SMI algorithm provides the required interference suppression assuming the true covariance matrix is known.

In normal applications, the true covariance matrix is not known *a priori* due to incomplete knowledge of the receive antenna patterns and received signal characteristics. As a result, samples of the received signals (i.e. snapshots) are used to estimate the covariance matrix which, in turn, is used in the weight calculation. Dilsavor and Moses [6] have addressed the question of how many snapshots are needed to achieve good estimates of the modified SMI weights. Using Monte Carlo simulations and statistical theory, they found that the number of snapshots required for good estimates of the modified SMI weights increases with the amount of diagonal subtraction (i.e. with the fraction F). Thus, a tradeoff was found to exist between the improved suppression of weak interference provided by the modified SMI algorithm and the number of snapshots required to achieve that improvement.

The modified SMI algorithm has been implemented on the aforementioned experimental array built by Ward et.al. The experimental SMI array is fully adaptive with either two or three elements. It can simulate narrowband signal scenarios consisting of one desired signal arriving from broadside and up to two interference signals arriving from arbitrary directions. Since the original system built by Ward was a modified Applebaum array in the sidelobe canceller configuration, some hardware as well as software alterations were necessary in implementing the modified SMI algorithm. The purpose of this report is to describe the implementation of the modified SMI algorithm on the original experimental system and then to study

the performance of the experimental SMI system in comparison with theory.

The rest of the report is organized as follows. Chapter 2 offers a brief description of the original experimental system built by Ward. Chapter 3 describes the implementation of SMI on the original system and includes a description of hardware modifications, sampling procedures, the implemented algorithm, weight normalization, performance evaluation, and system calibration. Experiments conducted to evaluate the performance of the experimental system are described in Chapter 4. Chapter 5 contains our conclusions. Finally, the system software is included as an appendix.

Chapter 2

The Original Experimental System

The experimental system built by Ward, et. al. ([3],[4]) is a sidelobe canceller with an unweighted main element and two weighted auxiliary elements which employs a discrete version of the Applebaum algorithm for weight control. The signal scenarios which can be tested consist of a desired signal arriving from broadside and up to two interference signals arriving from arbitrary directions.

The system is designed to simulate the array whose auxiliary element weights are controlled by the modified feedback loops of Figure 2.1. In these modified feedback loops the effect of noise on the array weights is decreased by reducing the correlation between the noise components of the two inputs to the correlator. The reduction in correlation is achieved by using two spatially separate antennas, each followed by its own amplifier, with each auxiliary element as shown in Figure 2.1. In this configuration, the "3-element" array actually uses 5 antennas; one for the unweighted main element and two for each of the weighted auxiliary elements. For purposes

of parameter control and system performance evaluation, the experimental system can simulate the signals that would be received by the five antennas. Desired, interference, and noise signals are bench generated and combined in an array simulator to simulate the signals received by the five antennas. The experimental adaptive processor can also be used with signals received from geosynchronous satellites. In this case, the signals received by actual antenna elements are downconverted to 69MHz, the center frequency of the adaptive processor, and then fed to the experimental system. In the results presented in this document, the simulated signals are used.

A block diagram of the experimental system is given in Figure 2.2. The signal simulator generates a desired signal and two interference signals. In the array simulator, these signals are combined with each other and with noise to form the signals that would be received at the 5 antennas. The array processor applies weights to the auxiliary element signals and forms (by summing) the adapted array output. The system operates at 69 MHz with a bandwidth of 6 MHz. A digital computer (PDP 11/23) is used to implement the weight control algorithm, control the various components, and evaluate the system performance. A description of the individual system blocks is given below.

2.1 Array Simulator

Figure 2.3 shows a detailed block diagram of the array simulator. In the array simulator, the incident signals are combined and thermal noise is added to form the signals received at each array element, such that each

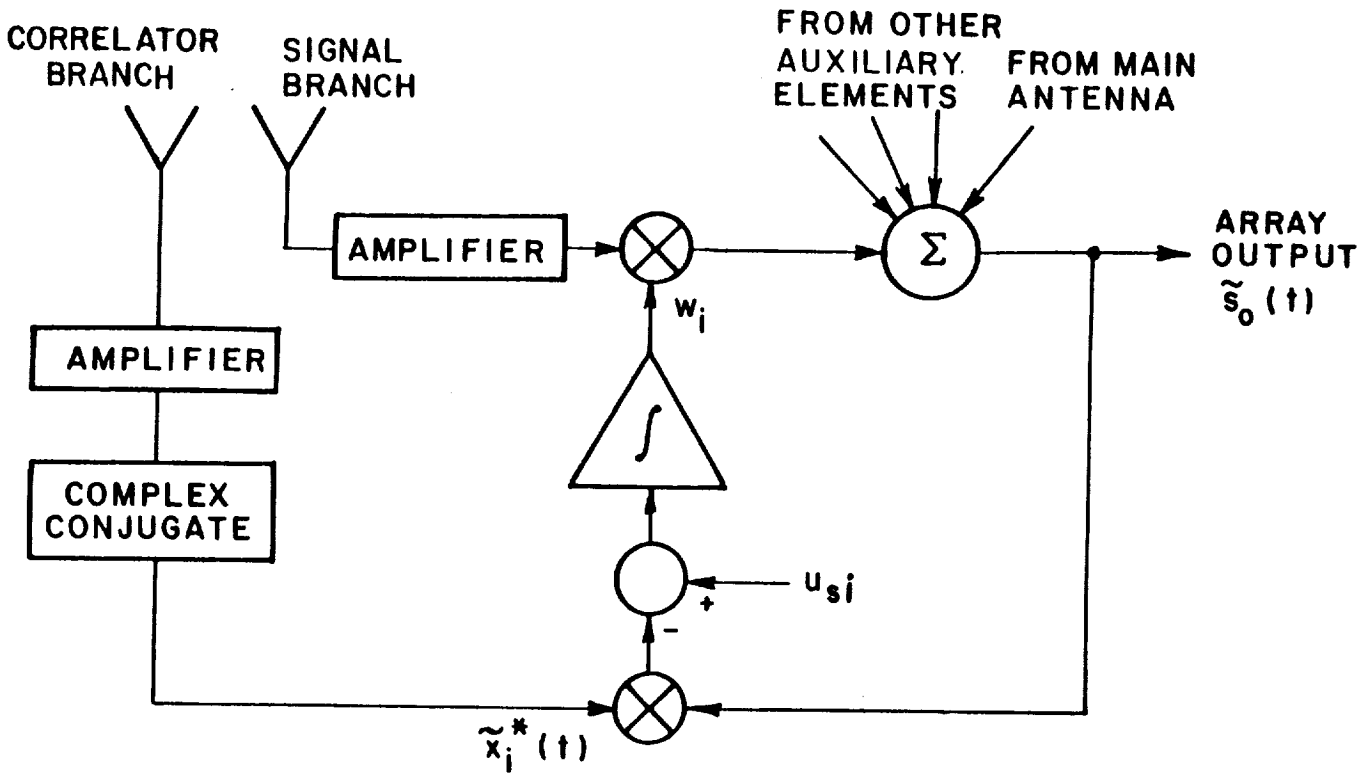


Figure 2.1: Modified feedback loop that uses separate antennas and amplifiers to decorrelate noise at the correlator inputs.

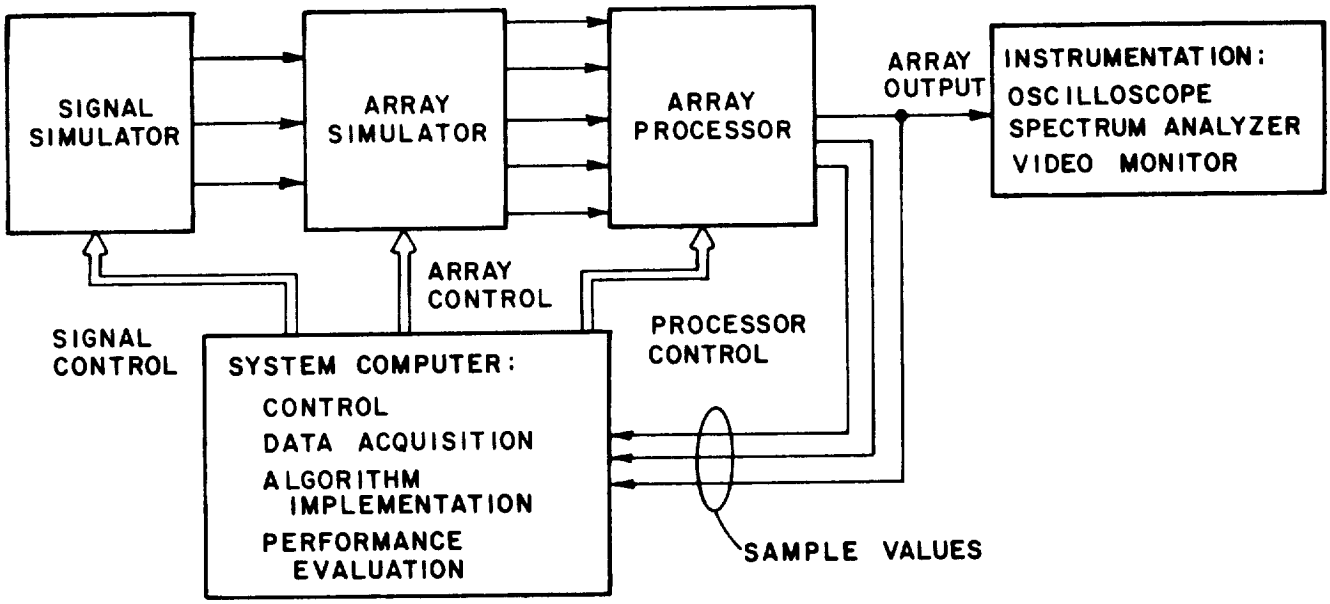


Figure 2.2: Block diagram of the experimental system.

signal contains a component due to the desired signal, components due to one or both interfering signals, and additive thermal noise. Thus the array simulator has three inputs for the three incident signals, and five outputs corresponding to the five antennas of the array. These five outputs are designated MAIN, AUX-1 SIGNAL, AUX-1 CORRELATOR, AUX-2 SIGNAL, AND AUX-2 CORRELATOR. The MAIN output is the signal received by the main element of our array.

The other outputs are the signals received by the two auxiliary elements of our sidelobe canceller with modified feedback loops. The designations SIGNAL and CORRELATOR specify the two branches of the modified feedback loop (Figure 2.1) of a particular auxiliary element. The SIGNAL branch is weighted and proceeds to the array output summer, whereas the CORRELATOR branch forms the input to the correlator. Note in Figure 2.3 that the noise components injected into the auxiliary SIGNAL branches and the MAIN channel are all from different noise sources, and thus are uncorrelated. Furthermore, the noise components in the auxiliary CORRELATOR branches originate from another noise source, and are therefore uncorrelated with the noise components of the SIGNAL branches. In Figure 2.3 the Δ's are zero-phase power dividers connected as summers. The α's are variable attenuators and the ϕ's denote variable phase shifters. N_1 through N_4 are the noise sources. The phase shifters simulate variations of the interfering signal directions of arrival by varying the interelement phase shifts between interfering signal components of different array elements . There are no phase shifters associated with the desired signal because it is assumed to arrive from broadside and thus is received with the same phase

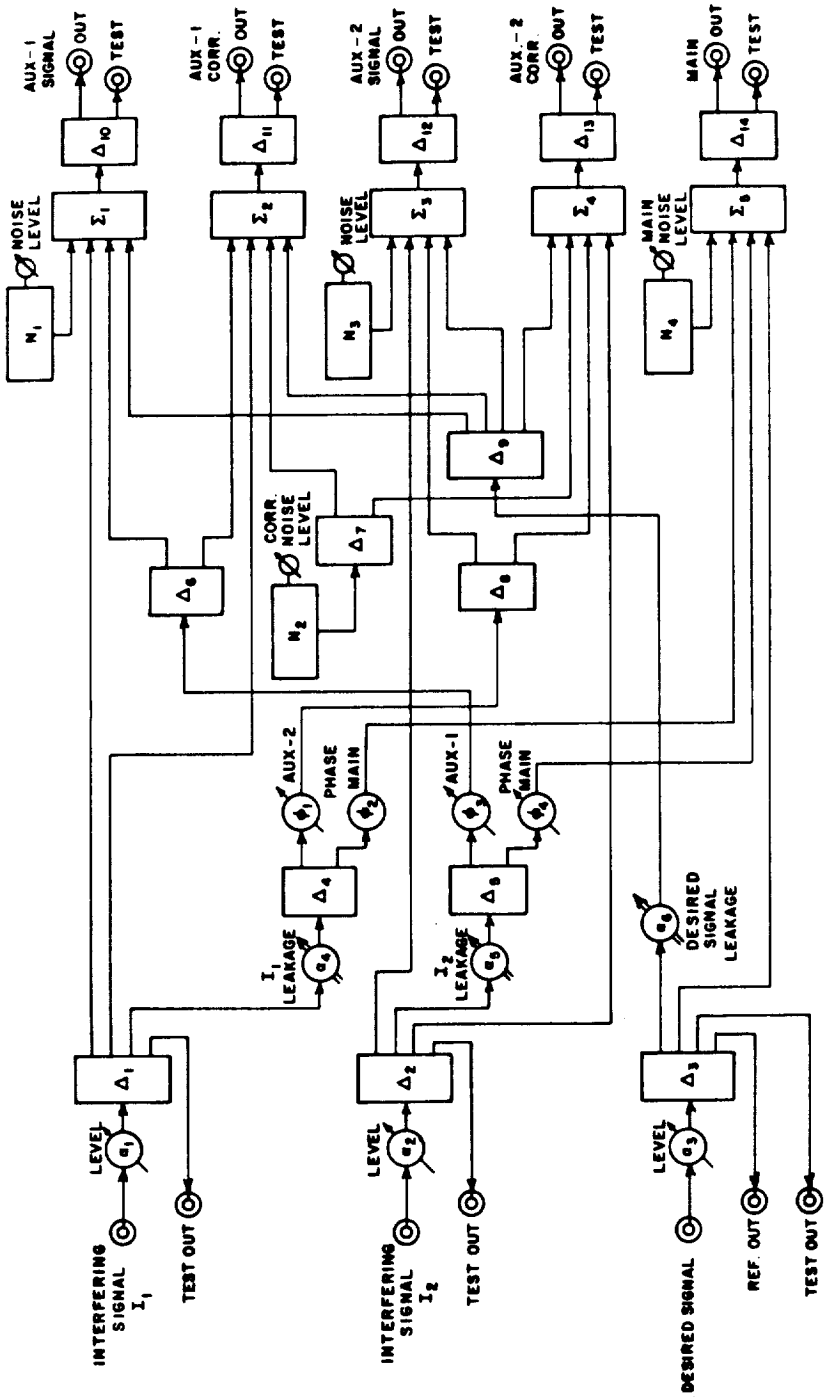


Figure 2.3: Array simulator; detailed block diagram.

at each array element. Variable attenuators are used to control the amount of each incident signal received at each output channel. This is analogous to varying the gains of the main and auxiliary elements in the directions of the incident signals. Once the desired signal scenario is set, the array simulator outputs are fed to the array processor, where the signals are sampled and the auxiliary element weights are determined by digital computer and are implemented in analog. The array processor is discussed next.

2.2 Array Processor

Together with the system computer, the array processor forms the two modified feedback loops of the sidelobe canceller, through which the weight control algorithms are implemented. A detailed block diagram of the array processor is shown in Figure 2.4. Note that the auxiliary channel correlator branch signals are down converted to baseband and quadrature detected by the vector demodulators (VDMs), as is the array output. These baseband voltages are simultaneously sampled, analog-to-digital (A/D) converted and read by the system computer which implements the weight control equation and calculates the array weights. The new weights are then (D/A) converted and applied to the auxiliary element SIGNAL branches as in-phase (I) and quadrature (Q) control voltages by the two vector modulators (VMODs). The weighted auxiliary elements are then summed with the main channel signal to form the array output.

In the array processor, the I and Q outputs of each vector demodulator are processed prior to being sampled. A low pass filter first removes the

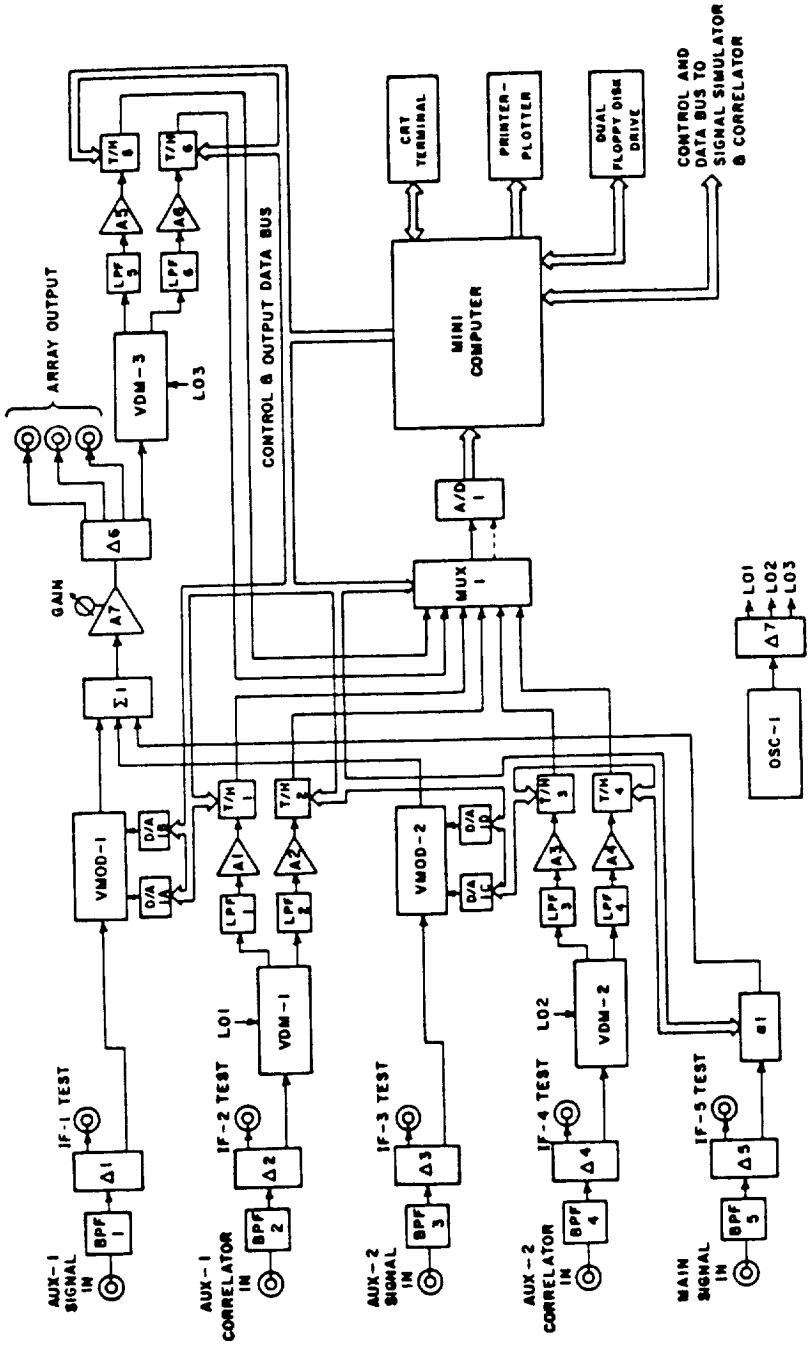


Figure 2.4: Array Processor; detailed block diagram.

second harmonic. The resultant baseband signals are then amplified to utilize the full dynamic range of the A/D converter. Track and hold devices allow the multiplexing of all six VDM outputs to the single A/D converter , so that they can be sampled simultaneously. This process preserves the signal correlation and noise correlation (if any) between the samples of different channels. For the experiments to be described here, the inputs to the processor are the signals from the array simulator. However, this same array processor may be used with signals from an actual antenna array, after downconversion to the array processor center frequency of 69 MHz [7]. The signals provided to the array simulator are described next.

2.3 Signal Simulator

The signal simulator synthesizes the desired signal and the two interfering signals, which are then combined in the array simulator to form the received signals at each array element. In order to measure adaptive array performance characteristics such as interference suppression, output interference-to-noise ratio (INR), and output signal-to-interference-plus-noise ratio (SINR), it is necessary to measure separately the desired signal power, the interference signal power, and the noise power present in each of the array elements and in the array output. Pulse modulated sinusoids are used as the desired signal and the interfering signals to accomplish this objective. As shown in Figure 2.5, the modulation on one interfering signal is staggered from the modulation on the other interfering signal, and from the desired signal modulation, such that the signal occupies a different por-

tion of the pulse repetition period. There is also a portion of the pulse period when only noise (no signal) is present. The desired and interfering signals are therefore all uncorrelated with each other (for all interelement time delays of interest). The track/hold devices of the array processor are triggered in synchronism with this modulation envelope. Because of A/D conversion speed limitations, successive samples are not taken in the same waveform period. Instead, each sample is from a different pulse repetition period , but is separated by only a small time interval from the point on the waveform at which the previous sample was taken. Thus, an effective sampling rate much higher than the sampling rate possible in real time is achieved. By varying the delay from the start of a period to a sampling instant, a sequence of samples covering the entire waveform is provided to the system computer. The levels of a particular signal component (desired, interference, or noise) are obtained by signal averaging over that portion of the pulse repetition period corresponding to the signal component under measurement. In this manner, the sampled data and the computer are also used for steady state adaptive array performance evaluation. However, as discussed in the next chapter, this pulse modulation scheme is exploited solely for performance evaluation and not in determining the auxiliary element weights. The experimental modified SMI system is described next.

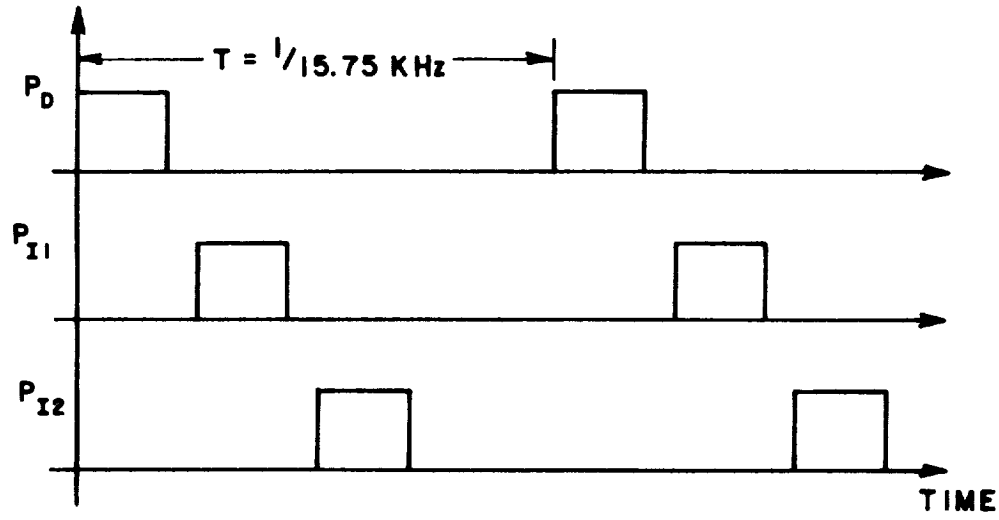


Figure 2.5: Staggered Pulse modulation of the desired and two interference signals.

Chapter 3

Experimental Modified SMI System

A number of issues were addressed in implementing the two- or three-element fully adaptive modified SMI array on the original system. These issues as well as the modified SMI algorithm itself are described in this chapter.

3.1 A Hardware Change in the Original System

The modified SMI system represents a software solution to the problem of weak interference suppression, whereas the original system embodies a hardware solution (modified feedback loops) to the same problem. Originally two antennas and amplifiers were used with each feedback loop in order to decorrelate the noise components of the two signals at each feedback loop correlator input. The original system modeled this situation by using different noise sources to generate the noise components. Separate

noise sources for the SIGNAL and CORRELATOR branches of each element of the array processor are not needed in the SMI system since this system does not need two separate antennas for each auxiliary element. Instead, the noise correlation will be reduced in the software. As a result, only three of the five outputs of the array simulator are needed by the three-element SMI array processor. One of these three is the MAIN element output of the array simulator which is fed directly to the MAIN SIGNAL INPUT of the array processor. The other two signals required by the array processor may be chosen from the remaining four array simulator outputs. Each of the two chosen signals is split into two parts using a two-way power divider. One output of the power divider is fed to the correlator branch while the other is fed to the signal branch of an auxiliary branch. For example, if the AUX-1 SIGNAL output of the array simulator is chosen to be the signal received by the second auxiliary element of our array then the AUX-1 SIGNAL output is split into two parts with a two-way power divider. One output of the power divider is fed to the AUX-2 SIGNAL input of the array processor and the other output of the power divider is fed to the AUX-2 CORR input of the array processor. Care has been taken not to choose the AUX-1 CORR and AUX-2 CORR outputs of the array simulator to be received by the array processor auxiliary elements since to do so would create a scenario in which the auxiliary element signals have correlated noise components; contrary to an assumption made in the modified SMI algorithm derivation. To see this, recall that a single noise source (N2 of Figure 2.3) is used to generate the noise components of both the AUX-1 CORR and AUX-2 CORR outputs of the array simulator.

3.2 Sampling and Weighting the Main Channel

The fully adaptive SMI array requires that the main channel as well as the auxiliary channels be sampled and weighted. Since the original system was a sidelobe canceller, the main channel was not equipped with a VDM for sampling; nor does it have a VMOD for weighting the main element signal (see Figure 2.4). The problem of sampling the main channel is solved by noticing that, if the auxiliary element weights (VMOD-1 and VMOD-2) are set to zero, the array output signal is equal to the main channel signal except for some attenuation and phase shift due to the real system components. Thus, the main channel is sampled by setting VMOD-1 and VMOD-2 to zero and sampling the output of VDM-3, the array output vector demodulator. The attenuation and phase shift will be offset by sample scaling and system calibration procedures that will be discussed in later sections of this chapter. In order to effectively weight the main channel, the weights computed by the SMI algorithm are always normalized so that the main element has unity weight. As a result, the main channel is weighted without a vector modulator.

3.3 Scaling the Samples

A significant difference between the SMI algorithm and the original Applebaum algorithm is that the SMI algorithm is open-loop whereas the Applebaum algorithm is closed-loop. In other words, the SMI algorithm derives weights using only the samples of the array element signals but the

Applebaum algorithm uses the array output to adjust the array weights. The input to the SMI algorithm consists of a set of K snapshots. Each snapshot is a vector (of length 3 for a 3-element array or length 2 for a 2-element array) of samples resulting from a simultaneous sampling of the array elements. Since the SMI algorithm derivation assumes an ideal array, the snapshots presented to the algorithm must be identified with samples of the array element signals of an ideal array. To make this identification, the samples taken from the A/D converter must be scaled in order to compensate for losses and phase shifts through the nonideal hardware components of our real array. Figure 3.1 compares the block diagrams of the ideal array and our experimental array. The 9dB attenuators in the auxiliary channels represent the inherent loss through the vector modulators when the weights are set equal to unity. The first 6dB attenuator at the output represents loss through the summer $\Sigma 1$ and the second represents loss through the 4-way power divider $\Delta 6$ of Figure 2.4. The 9dB gain accounts for the fact that A5 and A6 are set 9dB higher than A1-A4 in order to use the full dynamic range of the A/D converter. The absolute settings of A1-A6 are not important since we are concerned only with relative gains. The large box of Figure 3.1 encloses an ideal array which "exists within" our real system. The dots denote the locations where samples from the ideal array exist; and such samples shall be called ideal samples. The samples which are available at the VDM outputs shall be called the real system samples. As seen in the figure the auxiliary element ideal samples are simply the corresponding real system samples attenuated by 9dB. The main element ideal samples are the real system output samples taken with the weights

set to zero and boosted by 3dB (=6+6-9). Also note that the ideal adapted array output samples which will be needed later to calculate output powers of the adapted array are the real system output samples taken with the weights set to their adapted values and boosted by 3dB (=6+6-9). This scaling of samples is done in software as soon as the samples are retrieved from the A/D converter.

The scaling of samples described above represents a coarse amplitude adjustment in that, for example, the inherent loss through the first auxiliary element weight is not exactly 9dB. Furthermore, phase shifts through the real devices have not yet been offset. The further fine tuning which is required for the SMI algorithm to perform up to its abilities takes place in the weight calibration procedure which is discussed at the end of the chapter.

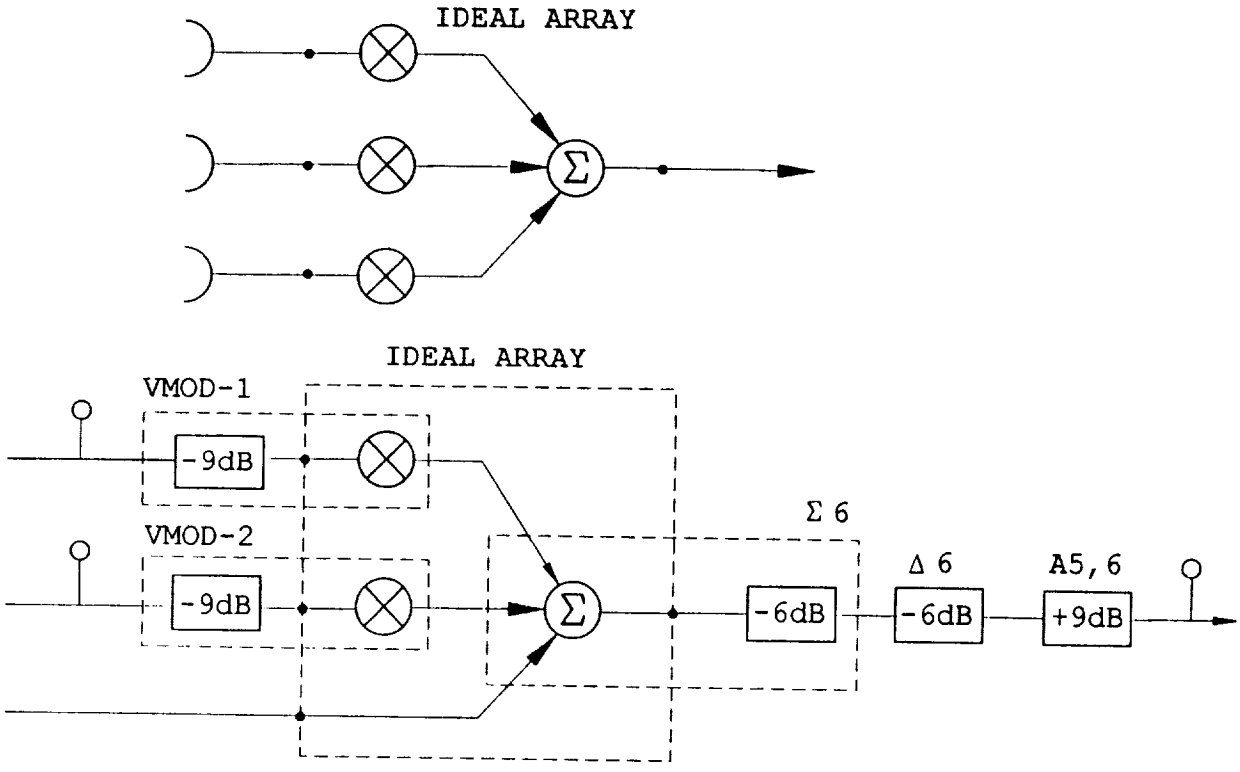
3.4 The Modified SMI Algorithm

This section describes the steps which are taken upon retrieving the scaled samples from the A/D converter that results in a set of modified SMI weights to be implemented on the array. The strategy here will be to state the equation used to generate the weights, define the terms in this equation, and thoroughly discuss how these terms are computed.

In the Modified SMI algorithm a vector of complex weight estimates $\hat{W}_K$ based on K snapshots are computed using

$$\hat{W}_K = \mu \hat{\Gamma}_K^{-1} \hat{S} \tag{3.1}$$

where the notation $\hat{\cdot}$ will be used throughout this discussion to indicate esti-



SIMPLE MODEL OF EXPERIMENTAL SYSTEM

Figure 3.1: Comparison of the ideal array and our experimental system.

mate. For example, the algorithm yields an estimate based on K snapshots of an optimal weight vector. For our 3-element array, the weight vector is defined to be

$$\hat{W}_K = \begin{bmatrix} \hat{w}_{K1} \\ \hat{w}_{K2} \\ \hat{w}_{K3} \end{bmatrix} = \begin{bmatrix} \hat{w}_{K1I} - j\hat{w}_{K1Q} \\ \hat{w}_{K2I} - j\hat{w}_{K2Q} \\ \hat{w}_{K3I} - j\hat{w}_{K3Q} \end{bmatrix} \tag{3.2}$$

where $\hat{w}_{K1}$ is the estimated complex weight for the first auxiliary element, $\hat{w}_{K2}$ is the estimated complex weight for the second auxiliary element, and $\hat{w}_{K3}$ is the estimated complex main channel weight. In order to apply a complex weight in our real array system each VMOD is provided the in-phase and quadrature components of the proper complex weight. Equation (3.2) defines the in-phase and quadrature components (subscripts $_I$ and $_Q$, respectively) of the complex weights. The minus sign is included in the definition since a VMOD input signal is split (within the VMOD) into two components such that the quadrature-weighted component *lags* the in-phase-weighted component by 90 degrees.

In Equation (3.1), μ is an arbitrary complex scalar which is chosen here so that $\hat{w}_{K3} = 1 + j0$. This normalization is necessary for our array system since the main channel is not equipped with its own VMOD. $\hat{\Gamma}_K$ is the (3×3) modified sample (estimated) covariance matrix given by

$$\hat{\Gamma}_K = \hat{\Phi}_K - F\lambda_{min}(\hat{\Phi}_K)I \tag{3.3}$$

and $\hat{S}$ is the steering vector. The method for generating a steering vector is presented at the end of this chapter.

In Equation (3.3), the standard sample covariance matrix $\hat{\Phi}_K$ based on

K complex signal snapshots $\{X_k\}_{k=1}^{K}$ is the block average

$$\hat{\Phi}_K = \frac{1}{K}\sum_{k=1}^{K} X_k X_k^H, \tag{3.4}$$

where ^{H} indicates Hermitian transpose. Also in Equation (3.3), F is the fraction between 0 and 1 which is chosen to achieve a desired level of interference suppression, $\lambda_{min}(\hat{\Phi}_K)$ is the minimum of the 3 real positive eigenvalues of the positive definite Hermitian matrix $\hat{\Phi}_K$, and I is the (3×3) identity matrix.

The procedure for computing the array weights given K complex (3×1) signal snapshots is clear. First, the standard sample covariance matrix $\hat{\Phi}_K$ is computed as in Equation (3.4) and its minimum eigenvalue is found. These quantities are used in Equation (3.3) along with the chosen value of F to yield the modified sample covariance matrix $\hat{\Gamma}_K$. Next, $\hat{\Gamma}_K$ and the approximate steering vector $\hat{S}$ are substituted into Equation (3.1) to yield the complex weights $\hat{W}_K$. Remember that μ is chosen to make $\hat{w}_{K3} = 1+j0$. Finally, the I and Q components of each complex weight are found from Equation (3.2) and are provided to the VMODs in analog. The next task is to describe how the complex signal snapshots X_k are generated.

Let x_{k1}, x_{k2}, x_{k3}, be the *scaled* (see Section 3.3) complex samples of the signals received at the first auxiliary, second auxiliary, and main channel at time index k. Recall, for example, that x_{k3} is a scaled version of a sample taken through VDM-3 while the auxiliary element weights are set to zero. The k^{th} signal snapshot X_k is a vector of these samples given by

$$X_k = \begin{bmatrix} x_{k1} \\ x_{k2} \\ x_{k3} \end{bmatrix} = \begin{bmatrix} x_{k1I} + jx_{k1Q} \\ x_{k2I} + jx_{k2Q} \\ x_{k3I} + jx_{k3Q} \end{bmatrix}. \tag{3.5}$$

In the communication signal environment, the k^{th} sample of the n^{th} element will contain desired, interference, and noise components, and thus can be written

$$x_{kn} = x_{knD} + \sum_{m=1}^{M} x_{knIm} + x_{kn\eta} \tag{3.6}$$

for M interference signals. Using (3.6) in (3.5), the k^{th} snapshot vector is written in terms of its signal component vectors

$$X_k = X_{kD} + \sum_{m=1}^{M} X_{kIm} + X_{k\eta}. \tag{3.7}$$

The SMI algorithm uses K such snapshots to form the covariance matrix estimate $\hat{\Phi}_K$ which can be written using (3.7) in (3.4) as

$$\begin{aligned}
\hat{\Phi}_K &= \frac{1}{K}\sum_{k=1}^{K}(X_{kD} + \sum_{m=1}^{M} X_{kIm} + X_{k\eta})(X_{kD} + \sum_{m=1}^{M} X_{kIm} + X_{k\eta})^H \\
&= \frac{1}{K}\sum_{k=1}^{K}(X_{kD}X_{kD}^H + \sum_{m=1}^{M} X_{kIm}X_{kIm}^H + X_{k\eta}X_{k\eta}^H) \\
&+ \frac{1}{K}\sum_{k=1}^{K}(X_{k\eta}\sum_{m=1}^{M} X_{kIm}^H + \sum_{m=1}^{M} X_{kIm}X_{k\eta}^H) \\
&+ \frac{1}{K}\sum_{k=1}^{K}(X_{kD}X_{k\eta}^H + X_{k\eta}X_{kD}^H) \\
&+ \frac{1}{K}\sum_{k=1}^{K}(X_{kD}\sum_{m=1}^{M} X_{kIm}^H + \sum_{m=1}^{M} X_{kIm}X_{kD}^H) \\
&+ \frac{1}{K}\sum_{k=1}^{K}\sum_{\substack{m_1=1 \\ m_1\neq m_2}}^{M}\sum_{m_2=1}^{M} X_{kIm_1}X_{kIm_2}^H.
\end{aligned} \tag{3.8}$$

The first line of the final expression in (3.8) approaches the true covariance matrix (which we are estimating) whereas the last four lines approach zero as $K \to \infty$.

When signals received from geosynchronous satellites are used as input to the array processor, a snapshot is formed as a vector of signal samples generated by simultaneously sampling the vector demodulator VDM outputs of the array processor (see Figure 2.4) and applying the calibration factors (see Sections 3.3 and 3.6). However, when the bench-generated signals (i.e. the outputs of the array simulator) are used as inputs to the array processor, care must be taken in forming a set of K snapshots $\{X_k\}_{k=1}^{K}$ so as not to take advantage of the special pulse-modulated signal scheme employed by our experimental system. Recall that the desired and interference signals are separated in time (see Figure 2.5) so that their powers can be calculated and our system performance evaluated. Thus, if (when using bench-generated signals) a snapshot is formed by simply combining the samples taken at a single instant of time then no single snapshot would contain both desired and interference signal components. As a result, the last two lines of (3.8) would not be present in the resulting covariance matrix estimate. Although this lack of desired-interference crossterms would not change the asymptotic value of the covariance estimate it would give an unrealistic (overly-optimistic) picture of the performance of the array based on a finite number of samples. To remedy this situation, a single snapshot X_T containing all signal components is "built" by simply summing three phase-shifted single-component snapshots; making sure that one of the addends comes from the desired signal portion of the pulse repetition period (call this snapshot X_D), another (X_{I1}) from the first interference signal portion, and the third (X_{I2}) from the second interference portion (see Figure 2.5). An addend from the noise-only portion of the pulse repetition period

is not necessary since noise is already present in the other three addends. Thus, a typical bench-generated snapshot is formed as

$$X_T = e^{j\phi_1} X_D + e^{j\phi_2} X_{I1} + e^{j\phi_3} X_{I2}. \tag{3.9}$$

The phase angles ϕ_1, ϕ_2, and ϕ_3 are i.i.d. uniform random variables on $[0, 2\pi]$. The random phases are needed to decorrelate the bench-generated desired and interference signals. This apparent correlation between received signals was a byproduct of using a single 69MHz oscillator to generate the carrier for the desired and interference pulse signals. The summing procedure (3.9) for forming bench-generated snapshots was used in [6] in order to study the performance of the system as a function of the number of snapshots.

This report is specifically concerned with the steady state performance of the modified SMI system. Thus, it is to our advantage here to omit the last two lines of (3.8) from the covariance estimate by not using the snapshot summing procedure of (3.9). In other words, in this report, a snapshot is formed as a vector of (calibrated) samples which were taken at a single instant of time. In doing so we shorten the sample-taking process.

The above presentation of the SMI algorithm was formulated in terms of a three-element array. The experimental array is also capable of implementing a two-element array by simply ignoring one of the two auxiliary elements. For this two-element configuration the algorithm must be adjusted accordingly. In particular, the estimated weight vector $\hat{W}_K$ becomes length 2, with the first entry being the auxiliary element weight and the second entry being the main element weight. As before, the weight vector is

normalized so that the main element weight (now $\hat{w}_{K2}$) is unity. Similarly, the snapshot vector becomes length 2 with the auxiliary element sample occupying the first entry and the main element sample occupying the second entry.

It may have occurred to the reader that the calculation of input and output signal and noise powers must be thought out carefully so that the performance results obtained here relate directly to results found in the literature. The power calculation procedure is discussed next, followed by the topics of system calibration and steering vector generation to conclude the chapter.

3.5 Power Calculations

In order to quantitatively examine the steady-state performance of the experimental modified SMI array it is necessary to calculate (from a set of samples) the desired and interference signal powers and the noise power in the adapted array output signal and the two or three input signals (depending on whether a two- or three-element array is being implemented). Using these power measurements, quantities such as interference suppression (IS) and output signal-to-noise ratio (SNR) can be found and compared with those of theory. Comparison of theoretical and experimental system results for several steady-state experiments appear in the next chapter. This section explains the procedure for calculating the various signal powers, IS, SNR, etc.

Figure 3.2 shows the I and Q outputs of one vector demodulator (VDM).

The D pulses are the desired signal, and I1 and I2 are the interfering signals. Due to the pulse modulation, the desired signal amplitudes are calculated from the sampled data over time interval τ_1, and the interfering signal amplitudes are calculated from the samples over τ_2 and τ_3. Samples over interval τ_4 are used to compute the noise power in each channel and, in addition, are used to calculate and correct for any DC offset voltages due to the detector amplifiers (A1-A6 of Figure 2.4). Even though these offset voltages have been nulled by adjustments at the amplifiers themselves, this correction is done in software as a precaution against any drift that may occur during the course of an experiment. Left uncorrected, such offset voltages would cause errors in the estimated covariance matrix and thus degrade adaptive array performance.

The signal level and noise power calculations are based on finite numbers of samples of a noisy signal and thus are estimates of the actual levels. In the experiments to be conducted, the suppressed interference components in the adapted array output signal will be as much as 30 dB below the noise level, making their levels very difficult to estimate accurately. The number of samples required for accurate estimates of the output interference signal levels (as well as all other signal levels) is reduced by simply turning off the noise generators while samples for signal power calculation are being taken. Of course, the noise generators are "on" when samples for weight and noise power estimation are taken..

Let x_{kJI} and x_{kJQ} be the k^{th} scaled I and Q samples of element J for $J = 1$, 2, and 3, as defined in Section 3.4. These samples are not the very same samples used to form the snapshots needed for weight computation

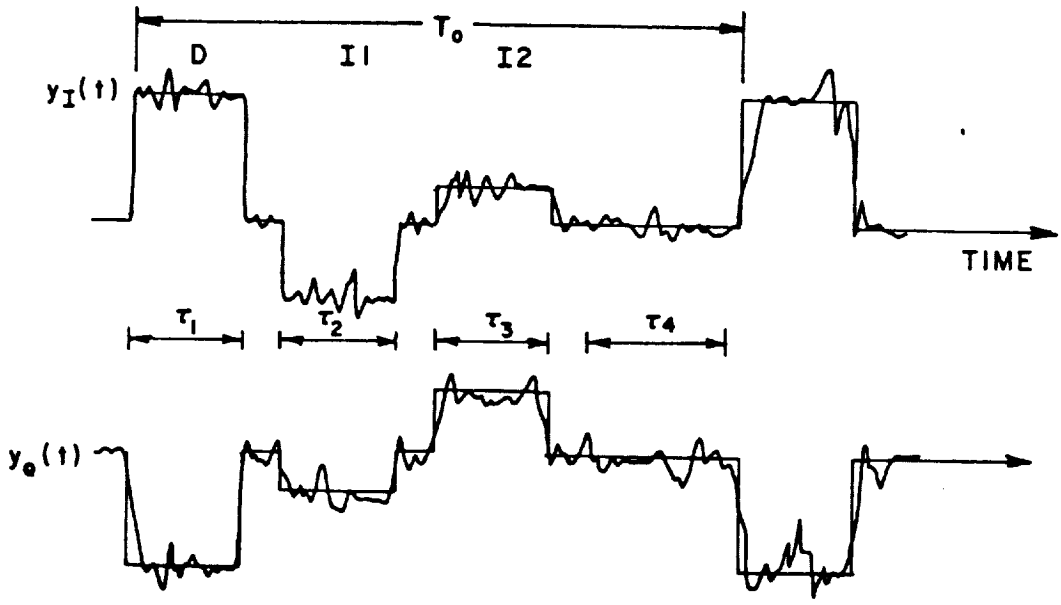


Figure 3.2: Typical I and Q VDM outputs.

but instead are different samples of the same signals. Furthermore, newly define x_{k4I} and x_{k4Q} to be the k^{th} scaled I and Q samples of the adapted array output signal. Suppose that this new data has been taken for k ranging from 1 to $K1$ in each of the 4 time intervals τ_i of Figure 3.2. Then compute

$$\hat{O}_{JP} = \frac{1}{K1}\sum_{k=1}^{K1} x_{kJP} \text{ on } \tau_4 \tag{3.10}$$

$$\hat{D}_{JP} = \frac{1}{K1}\sum_{k=1}^{K1} x_{kJP} - \hat{O}_{JP} \text{ on } \tau_1 \tag{3.11}$$

$$\hat{I1}_{JP} = \frac{1}{K1}\sum_{k=1}^{K1} x_{kJP} - \hat{O}_{JP} \text{ on } \tau_2 \tag{3.12}$$

$$\hat{I2}_{JP} = \frac{1}{K1}\sum_{k=1}^{K1} x_{kJP} - \hat{O}_{JP} \text{ on } \tau_3 \tag{3.13}$$

$$\hat{\sigma}^2_{JP} = \frac{1}{K1}\sum_{k=1}^{K1} (x_{kJP} - \hat{O}_{JP})^2 \text{ on } \tau_4 \tag{3.14}$$

for $J \in \{1,2,3,4\}$ and $P \in \{I,Q\}$. $\hat{O}_{JI}$ and $\hat{O}_{JQ}$ are the estimates of the offsets in the I and Q components of array signal J. $\hat{D}_{JI}$ and $\hat{D}_{JQ}$ are the estimates of the I and Q component amplitudes of the desired signal component in array signal J. $\hat{I1}_{JI}$, $\hat{I1}_{JQ}$, $\hat{I2}_{JI}$, and $\hat{I2}_{JQ}$ are analogous parameters for the first and second interference signals, respectively. Recall that the samples for the estimates of (3.10)-(3.13) are taken with the noise generators turned off. $\hat{\sigma}^2_{JI}$ and $\hat{\sigma}^2_{JQ}$ are the noise power estimates in the I and Q channels of array signal J.

From these equations we see that the signal amplitude calculations for the desired and interference signals are obtained by sample averaging over the particular portion of the pulse repetition period corresponding to the signal under measurement. The variances of these signal level estimates will depend on the average noise power σ^2 present in the particular array signal being averaged and on the number of samples $K1$ in the average. Specifically, the averages (3.10)-(3.13) are efficient maximum likelihood estimates of the I and Q offsets and signal amplitudes. The variance of the offset estimate of (3.10) is $\sigma^2/K1$ whereas the variance of the signal level estimates of (3.11)-(3.13) is $2\sigma^2/K1$. The noise power estimate of (3.14) is asymptotically unbiased [8].

From the I and Q signal amplitude and noise power estimates in array signal J, the average power of the desired, interference, and noise components of complex array signal J are computed as follows:

$$\hat{P}_{DJ} = \hat{D}_{JI}^2 + \hat{D}_{JQ}^2 \tag{3.15}$$

$$\hat{P}_{I1J} = \hat{I1}_{JI}^2 + \hat{I1}_{JQ}^2 \tag{3.16}$$

$$\hat{P}_{I2J} = \hat{I2}_{JI}^2 + \hat{I2}_{JQ}^2 \tag{3.17}$$

$$\hat{P}_{\eta J} = 3(\hat{\sigma}_{JI}^2 + \hat{\sigma}_{JQ}^2). \tag{3.18}$$

The factor of 3 in (3.18) is included to account for the fact that each signal snapshot used in the weight estimate contains a noise component whereas the desired signal is present in just 1/3 of these snapshots as are the first and second interference signals. Because the I and Q components of narrowband noise are uncorrelated [9], the total average noise power in a particular channel is the sum of the noise powers at the I and Q outputs of that VDM.

Once the power of each signal component (desired, interference, noise) in each array signal (first auxiliary, second auxiliary, main channel, adapted array output) has been computed any interesting performance measures can be easily found. For example, the output signal-to-noise ratio (SNR), first interference suppression (IS1), second interference suppression (IS2), and total interference suppression (IST) are just

$$SNR = \frac{P_{D4}}{P_{\eta 4}} \tag{3.19}$$

$$IS1 = \frac{P_{I14}}{P_{I13}} \tag{3.20}$$

$$IS2 = \frac{P_{I24}}{P_{I23}} \tag{3.21}$$

$$IST = \frac{P_{I14} + P_{I24}}{P_{I13} + P_{I23}}. \tag{3.22}$$

In Chapter 4, these performance measures will be used to compare the performance of the experimental system with that of an ideal modified SMI array operating in the same signal environment. Before the experimental array can perform up to its capabilities, however, the system must be calibrated. System calibration is discussed in the next section.

3.6 System Calibration

The experimental system uses a software calibration procedure to correct for non-ideal characteristics (i.e. losses and phase shifts) of the real system components. Recall that the sample scaling of Section 3.3 represented a coarse adjustment of sample amplitudes. Phase corrections and further amplitude corrections to the samples are necessary to ensure proper array performance.

The calibration procedure used here adapts the weights for the simple scenario of a 2-element array receiving only a single interference signal (no noise, no desired signal). Since the scaled samples are not ideal, the estimated weights will not be ideal. However, ideally we know that the array should suppress the interference at the array output beneath measureable levels. By scaling the amplitude and shifting the phase of the adapted auxiliary element weight, an ideal weight which completely suppresses the interference is found. The amplitude scale factor A and phase

shift ϕ required to transform the adapted auxiliary element weight to the ideal weight can be represented by the complex number $c = Ae^{j\phi}$. By applying the "fine-tuning" sample adjustment factor $1/c^*$ to the scaled auxiliary element samples in subsequent experiments the resulting adapted auxiliary element weight is made equal to the ideal weight. To see this, suppose the fine-tuning adjustment factor for the first auxiliary element samples were known; call it $1/c^*$. Then the k^{th} fine-tuned snapshot X_{kFT} may be written in terms of the k^{th} coarse-adjusted snapshot X_{kCA} (see Section 3.3) as

$$X_{kFT} = \begin{bmatrix} \frac{1}{c^*} & 0 \\ 0 & 1 \end{bmatrix} X_{kCA} \tag{3.23}$$

$$\equiv CX_{kCA}. \tag{3.24}$$

As a reminder, we are considering the two-element array in which the first auxiliary element sample occupies the first element of the snapshots and the main element sample occupies the second element of the snapshots. Thus, (3.23) implies that we are calibrating the first auxiliary element samples with respect to the main element samples. The estimated covariance matrix $\hat{\Phi}_{FT}$ based on K of the fine-tuned snapshots may be written in terms of the corresponding covariance matrix $\hat{\Phi}_{CA}$ based on the coarse-adjusted snapshots. Using (3.24) in (3.4), we have

$$\begin{aligned} \hat{\Phi}_{FT} &= \frac{1}{K}\sum_{k=1}^{K} X_{kFT}X_{kFT}^H \\ &= \frac{1}{K}\sum_{k=1}^{K}[CX_{kCA}][CX_{kCA}]^H \\ &= \frac{1}{K}\sum_{k=1}^{K} CX_{kCA}X_{kCA}^H C^H \end{aligned}$$

$$= C\left[\frac{1}{K}\sum_{k=1}^{K} X_{kCA}X_{kCA}^{H}\right]C^{H}$$
$$= C\hat{\Phi}_{CA}C^{H}. \tag{3.25}$$

Using (3.25) in (3.3) with $F = 0$ (standard SMI) and substituting the result in (3.1), we may express the standard SMI weights $\hat{W}_{FT}$ based on K fine-tuned snapshots as

$$\begin{aligned}\hat{W}_{FT} &= \mu\hat{\Phi}_{FT}^{-1}S \\ &= \mu\left[C\hat{\Phi}_{CA}C^{H}\right]^{-1}S \\ &= \mu(C^{H})^{-1}\hat{\Phi}_{CA}^{-1}C^{-1}S \\ &= \mu\begin{bmatrix} c & 0 \\ 0 & 1 \end{bmatrix}\hat{\Phi}_{CA}^{-1}\begin{bmatrix} c* & 0 \\ 0 & 1 \end{bmatrix}S. \end{aligned} \tag{3.26}$$

Let ϕ_{ij}^{-1} be the ij^{th} element of $\hat{\Phi}_{CA}^{-1}$. Also since the desired signal is not present in this scenario, we are free to choose $S = [0\ 1]^{T}$. When we make these substitutions in (3.26) and evaluate, we have

$$\begin{aligned}\hat{W}_{FT} &= \mu\begin{bmatrix} c & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \phi_{11}^{-1} & \phi_{12}^{-1} \\ \phi_{21}^{-1} & \phi_{22}^{-1} \end{bmatrix}\begin{bmatrix} c* & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \mu\begin{bmatrix} |c|^{2}\phi_{11}^{-1} & c\phi_{12}^{-1} \\ c^{*}\phi_{21}^{-1} & \phi_{22}^{-1} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \mu\begin{bmatrix} c\phi_{12}^{-1} \\ \phi_{22}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} c\hat{w}_{1CA} \\ \hat{w}_{2CA} \end{bmatrix} \end{aligned} \tag{3.27}$$

where $\hat{w}_{1CA}$, and $\hat{w}_{2CA}$ are the auxiliary and main element weights based on the corresponding course-adjusted (not fine-tuned) snapshots. In words, (3.23) and (3.27) state that the effect of fine-tuning the auxiliary element samples with the factor $1/c*$ is to change the resulting auxiliary element

weight by the factor c. Recall that this result assumes a particular scenario and $S = [0\ 1]^T$. However, the sample scale factor $1/c^*$ which depends on cable lengths and real component characteristics should be independent of signal scenario and choice of steering vector.

Our goal is to determine the scale factor c. To do this, the single-interference scenario described above is set up and the uncalibrated (coarse-adjusted but not fine-tuned) system is used to estimate a coarse-adjusted auxiliary element weight $\hat{w}_{1CA}$ which does not yield a completely suppressed interference signal at the array output. By stepping the magnitude and phase of the auxiliary element weight through a range of values, we determine the ideal fine-tuned weight $\hat{w}_{1FT}$ which causes the output interference signal to be completely suppressed. We then calculate $c = \hat{w}_{1FT}/\hat{w}_{1CA}$ (using the result in (3.27)) and apply the fine-tuning factor $1/c^*$ (see (3.23)) to the first auxiliary element samples in subsequent experiments. The analogous procedure is followed to fine tune the second auxiliary element samples.

This calibration process may be fully implemented in software by using the output interference power measurement capability described in the previous section as follows:

Calibration Procedure

I Set up the scenario

A. 2-element array (main and first auxiliary).

B. Only first interference signal, no desired signal, no added noise.

C. Set attenuator a_4 of Figure 2.3 so that a significant amount of interference is injected into the main channel to be

cancelled with the interference in the first auxiliary.

II Adapt the weight

A. Take samples for snapshot formation (enough for steady state estimation).

B. Estimate the first auxiliary element weight w. Note: Since the weight is uncalibrated, it needs to be scaled in phase and magnitude.

III Find phase and magnitude scale factors (ϕ_o and M_o).

A. Let $M_o = 1$ and $\phi_o = 0$.

B. Step the phase of the weight $wM_oe^{j\phi_o}$ through a range of values. At each step (varying ϕ):

i. Apply the phase-adjusted weight $e^{j\phi}wM_oe^{j\phi_o}$.

ii. Calculate and save the interference power at the array output.

C. Choose the phase ϕ that yielded the minimum output interference power, call it ϕ_a.

D. Set $\phi_o = \phi_o + \phi_a$.

E. Scale the magnitude of the weight $wM_oe^{j\phi_o}$ through a range of values. At each step (varying M):

i. Apply the magnitude-scaled weight $MwM_oe^{j\phi_o}$.

ii. Calculate and save the interference power at the array output.

F. Choose the magnitude scale factor M that yielded the minimum output interference power, call it M_a.

G. Set $M_o = M_oM_a$.

H. Repeat IIIB.-IIIG. until convergence of M_o and ϕ_o.

IV In all subsequent experiments modify the first auxiliary element (scaled) samples by the factor $e^{j\phi_o}/M_o$.

V Repeat I-IV using the second auxiliary element instead of the first in order to fine tune the second auxiliary element (scaled) samples.

The above procedure has been made iterative because the VMOD's do not give us completely independent control of the weight magnitude and phase. Let us now turn to the problem of generating a steering vector.

3.7 Estimation of the Steering Vector

The steering vector contains the information that allows the SMI array to distinguish the desired signal from the interfering signals. In order to estimate the steering vector for a given bench-generated signal scenario we begin by turning off the interference signals and noise generators so that only the desired signal and negligible system noise are present in the received signals. We then take snapshots and form the estimate

$$\hat{S} = \frac{1}{K} \sum_{k=1}^{K} x_{k1}^{*} X_k \tag{3.28}$$

where X_k is the k^{th} snapshot and x_{k1} is the first element of X_k. We note in passing that (3.28) is simply the first column of the standard sample covariance matrix (3.4). Once the steering vector estimate is stored in memory, the interference signals and noise generators are restored to the desired levels. This method of steering vector generation will be used in later experiments. Throughout this report, however, the main antenna is of relatively large gain compared to the auxiliary elements. This assumption

allows us to approximate the steering vector as

$$\hat{S} = [0\ 0\ 1]^T \tag{3.29}$$

where ^{T} denotes transpose and the third element of $\hat{S}$ corresponds to the main element. The subject of generating a steering vector when the received signals are from geosynchronous satellites is not addressed here. Now that the experimental modified SMI system has been described and all relevent procedures have been discussed, it is time to describe its performance through experimentation.

Chapter 4

Experimental Results

The experiments test the steady state performance of the modified SMI array system for various input signal scenarios. By steady state performance we mean that performance achieved by using many snapshots in the weight estimation. The performance is studied using a step-by-step approach. First, the system is tested using standard SMI (i.e. with $F = 0$). These tests establish the invariance of the calibration factors to changes in signal scenario, compare the experimental results with theory, and point out a lack of suppression of weak interference. The modified SMI algorithm is then used to increase the suppression of weak interference while maintaining a strong desired signal. In all experiments, the noise power in each element of the array is (approximately) the same and the desired signal, when present, is incident from broadside and present only in the main channel. The procedure used for conducting an experiment is

1. Set the attenuators of the array simulator so as to present certain desired and interference signal levels to the array processor.

2. Turn off the noise generators.

3. Set the weights to zero.

4. Estimate the desired and interference signal levels using (3.15)-(3.17) (for J = 1, 2, and 3)

5. Set the noise generators to the desired levels.

6. Estimate the noise power in each array element using (3.18) (for J = 1, 2, and 3).

7. Collect samples with which to form snapshots (making sure that enough samples are taken to yield near steady state weights).

8. Estimate the modified SMI array weights.

9. Apply the estimated weights.

10. Estimate noise power in the array output using (3.18) with J=4.

11. Turn off noise generators.

12. Estimate the desired and interference signal powers in the array output (J=4) using (3.15)-(3.17).

13. Calculate performance parameters such as SNR and IS1 of (3.19) and (3.20).

4.1 Experiment 1: Two elements, one interference, standard SMI, vary interference power

In the first experiment, each auxiliary channel is tested separately by using the two-element (main and a single auxiliary) capability of the experimental system and standard SMI ($F = 0$). No desired signal is present. A single interference of varying power is incident on the array from a fixed direction. The direction of arrival (52° off broadside in this case) is unimportant since any element pattern considerations are absorbed into the received signal powers. INR(aux)-INR(main) is fixed at 5.5dB. Figure 4.1 shows the interference suppression versus INR(main) for experiment and theory. One of the experimental curves corresponds to the two-element array consisting of the main and first auxiliary elements whereas the other corresponds to the main and second auxiliary element combination. A good agreement between theory and experiment is observed. In the linear region of the plot, a 5dB increase in interference power leads to a 10dB increase in its suppression. In the region INR(main)=10~15dB the small output interference power becomes difficult to measure even with the noise generators turned off, due to ever-present residual noise; the main sources of which are the detector amplifiers A5 and A6. Note that for weak interference (INR(main)<-5dB), standard SMI yields less than 7dB of interference suppression.

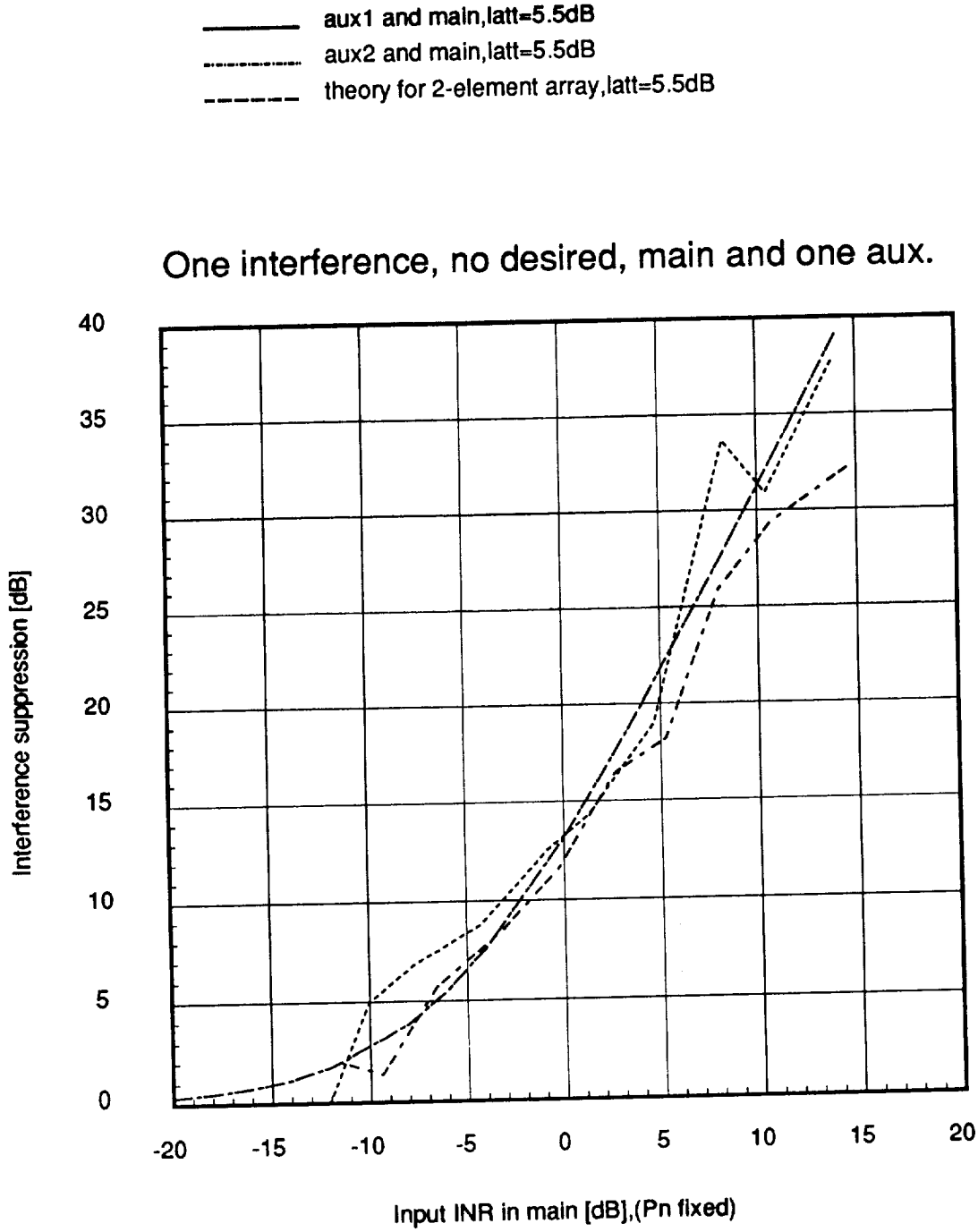


Figure 4.1: Experiment 1: Suppression of a single interference of varying power with a two-element array.

4.2 Experiment 2: Three elements, one interference, desired signal, standard SMI, vary interference arrival angle

Rather than varying the power level of the single interference we vary the direction of arrival (DOA) in Experiment 2. This time, standard SMI and all three elements are used and SNR(main)=16dB. The desired signal level in the auxiliary elements is negligible. The purpose here is to test the phase shifters of the array simulator and the phase-shifting ability of the vector modulators (weights). In addition, the invariance of the calibration factors to different scenarios is tested. The fixed input interference signal levels appear in Table 4.2. Figure 4.2 shows the results. One set of curves (theory and experiment) corresponds to the first interference signal whereas the other pair corresponds to the second interference signal. Five data points were used to form each experimental curve. There is reasonable agreement between theory and experiment since only one data point is further than 2dB from its theoretical value. As expected, interference suppression is theoretically and experimentally independent of the arrival angle (as long as desired and interference signals arrive from different directions). Also note that the level of interference suppression provided by the standard SMI algorithm for this particular scenario is about 7.5dB. This may not be enough in some applications.

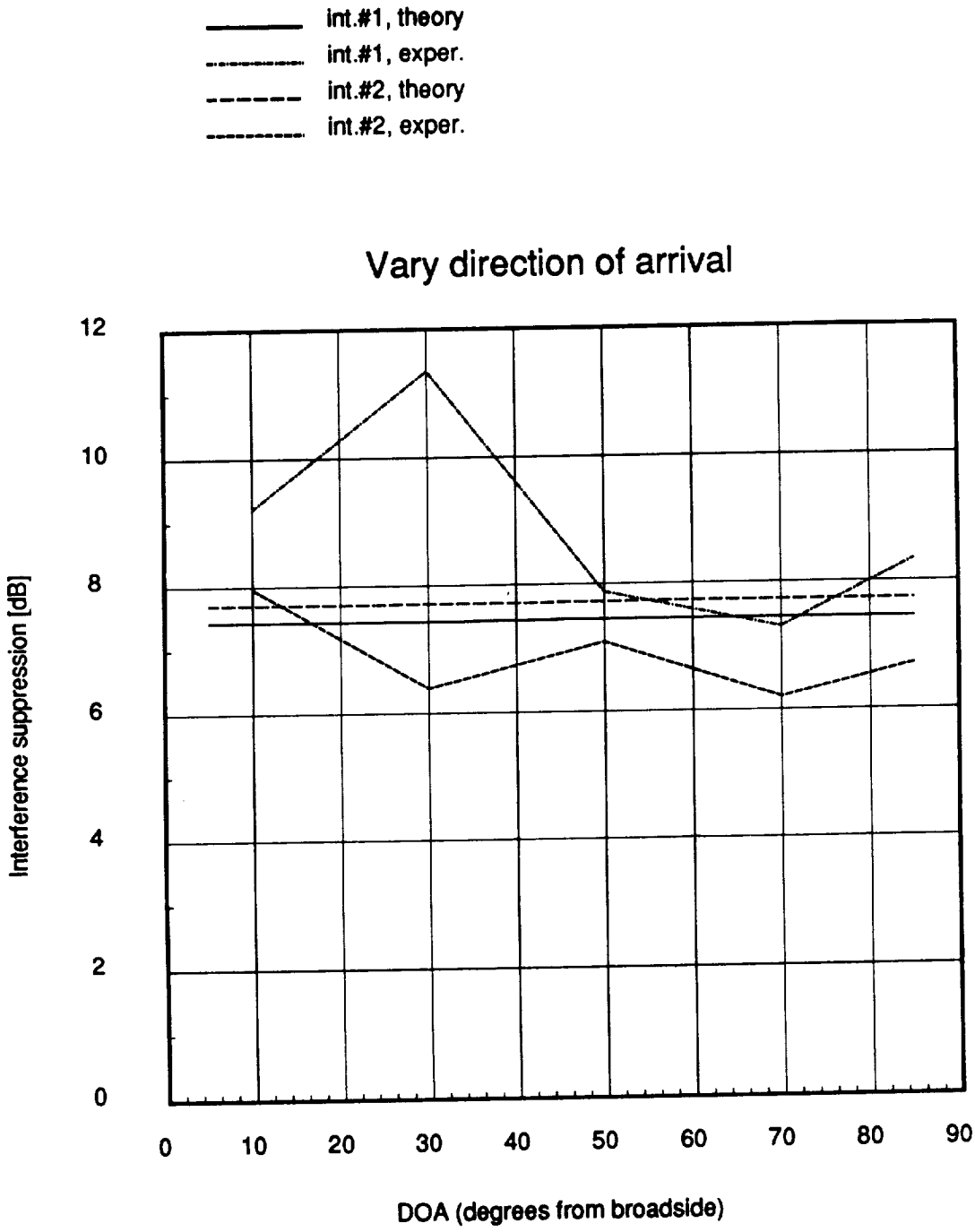


Figure 4.2: Experiment 2: Suppression of a single interference of varying arrival angle with a 3-element array and SNR(main)=16dB.

	INR(main)	INR(aux1)	INR(aux2)
Int. 1 only	-3.78	1.21	-16.84
Int. 2 only	-4.67	-16.48	1.47

Table 4.1: Interference signal levels for Experiment 2.

4.3 Experiment 3: Three elements, one interference, desired signal, standard SMI, vary interference power

Experiment 3 is a more complicated version of Experiment 1. Standard SMI and all three elements are used to suppress the first interference signal whose power level is varied. The desired signal is present with SNR(main)=16.5dB. Again, the desired signal level in the auxiliary elements is negligible. Figure 4.3 shows the results. For one pair of curves (theory and experiment), INR(aux1)-INR(main) is fixed at 6.5dB whereas for the other pair it is fixed at 9.7dB. The difference in interference suppression between the two curve pairs is the result of increasing the auxiliary element gain. Thus, in the linear portion of the curves, a 3.2dB (=9.7-6.5) increase in auxiliary element gain has led to approximately a 5dB increase in interference suppression for a given main element sidelobe level. Note again the lack of interference suppression for weak interference levels and low auxiliary element gain. For example, from the lower set of curves in 4.3 (corresponding to the lower gain auxiliary elements) we see that the array provides less than 10 dB interference suppression when $INR(main) < -4dB$. Theory and experiment show good agreement in

Figure 4.3 especially in the middle range -8dB<INR(main)<8dB. Above this range it becomes difficult to measure the small output interference power.

Figure 4.4 shows the output SNR curves for Experiment 3. Note that, as theory predicts, the SNR is maintained at the output.

4.4 Experiment 4: Three elements, one interference, desired signal, standard SMI, vary main antenna sidelobe

In the fourth experiment (Figure4.5) using standard SMI, the first interference signal is present, the first auxiliary element gain (INR(aux1)) is fixed, INR(aux2)= INR(main)-12dB, and the main element sidelobe level is allowed to vary. The second interference is off and the desired signal is present with SNR(main)=16.06dB. The top pair of curves are theory and experiment for INR(aux1)=7.63dB, and the middle and lower pair are for INR(aux1)=4.20dB and 1.20dB, respectively. The discrepancy between theory and experiment in the top pair of curves is explained, again, by the difficulty in measuring such small output interference powers. In the measureable regions, however, results are quite good. It is interesting to note that when the interference level in the main element is small relative to that in the auxiliary element (i.e. the main element sidelobe level is small compared to the auxiliary element gain), the theoretical interference suppression is relatively independent of the sidelobe level. From Figure 4.5 we see that for low-gain auxiliary antennas (INR(aux1)<1.20dB) and

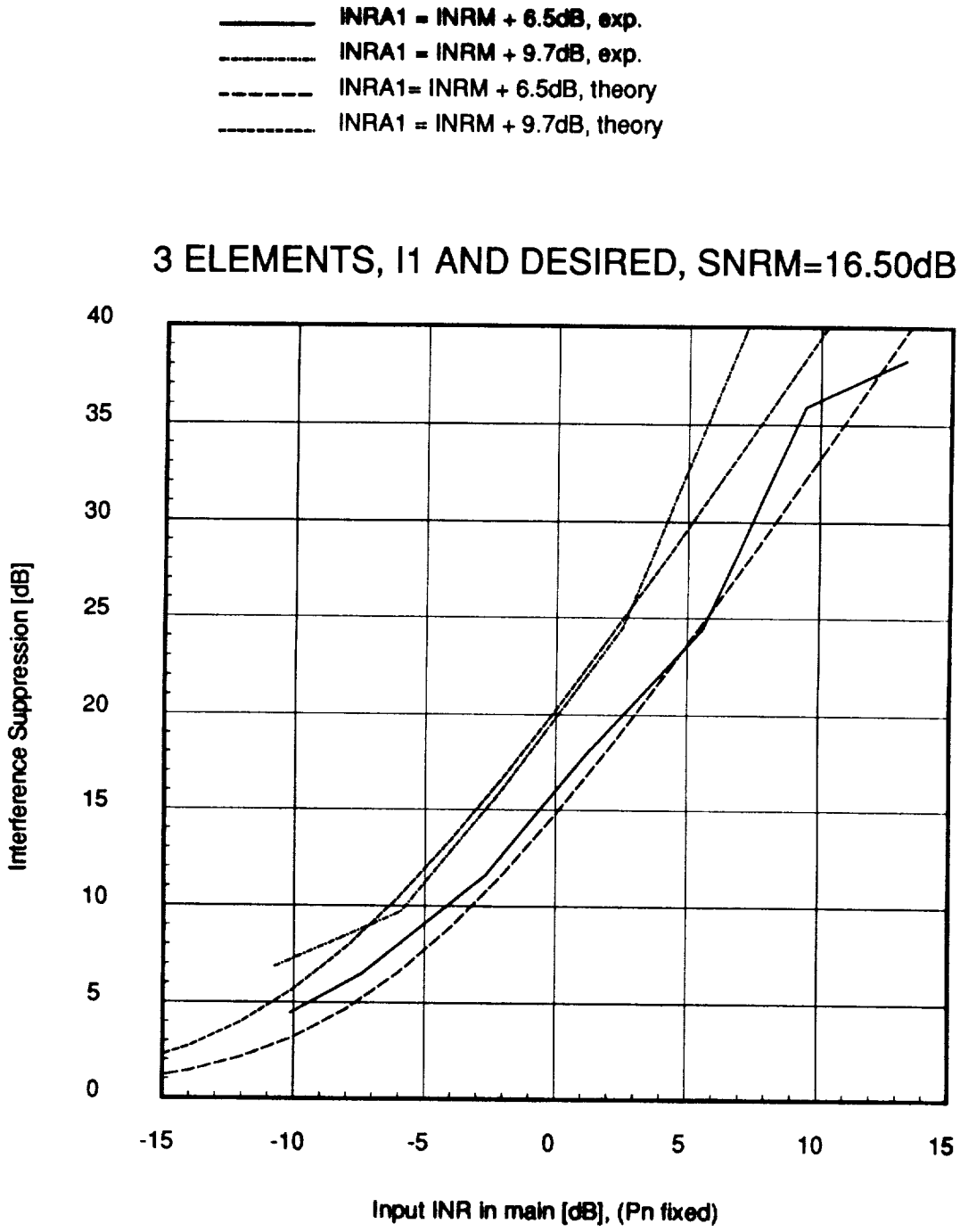


Figure 4.3: Experiment 3: Suppression of the first interference signal with a 3-element array, SNR(main)=16.5dB. The top pair of curves are theory and experiment for fixed INR(aux1)-INR(main)=9.7dB. The bottom pair are for INR(aux1)-INR(main)=6.5dB.

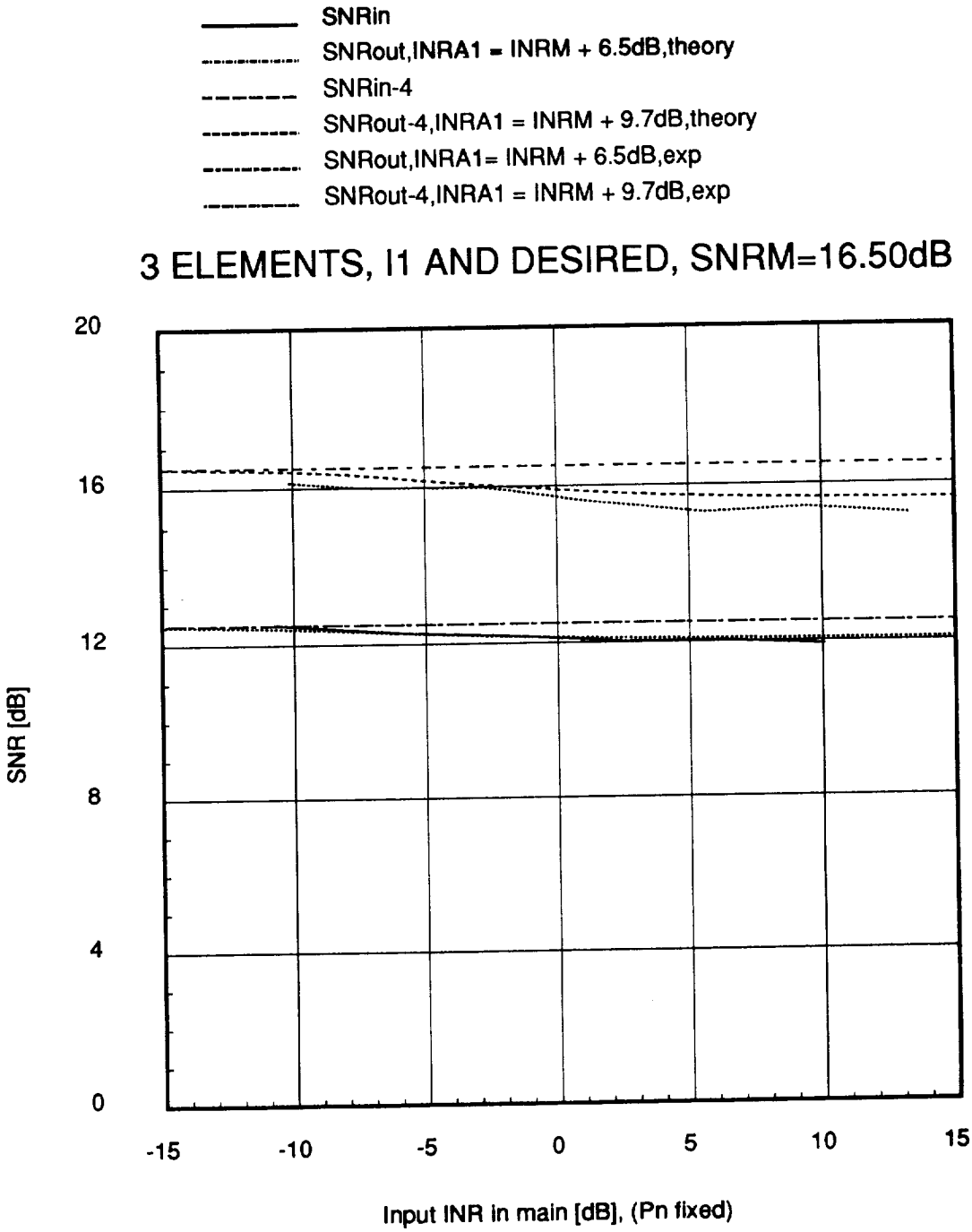


Figure 4.4: SNR plots for Experiment 3: The top three curves are input SNR(main), theoretical SNR(out), and experimental SNR(out) for INR(aux1)=INR(main) + 6.5dB. The bottom three curves are the same for INR(aux1)=INR(main) + 9.7dB and have been displaced -4dB for clarity.

weak interference in the main element (INR(main)<-5dB), there is less than 7.5dB interference suppression. Figure 4.6 shows the input SNR(main) and the three SNR(out) curves for Experiment 4. There is little or no loss in SNR from input to output. This experiment and the last will help us to quantitatively assess the value of using modified SMI.

4.5 Experiments 5 and 6: Three elements, one interference, desired signal, fixed input powers, modified SMI, vary fraction F

In the above experiments, we observed a lack of weak interference suppression in the case of low-gain auxiliary antennas. The purpose of Experiments 5 and 6 is to observe interference suppression when modified SMI is applied to such cases. In both of these experiments all 3 elements are used and only the first interference and desired signal are present. The results are plots of interference suppression and SNR as the fraction F of (3.3) is varied. Table 4.2 presents the input signal levels for the two experiments. INR(main) is fixed at -9.45dB in Experiment 5 and is -2.68dB in Experiment 6. Figures 4.7 and 4.8 show the results of Experiment 5 and Figures 4.9 and 4.10 are the results of Experiment 6. The SNR plots show input SNR(main) and theoretical and experimental SNR(out) and are presented "waterfall style" so that the curves do not lay atop one another (i.e. the SNR curves corresponding to the top pair of interference curves have been displaced -4dB).

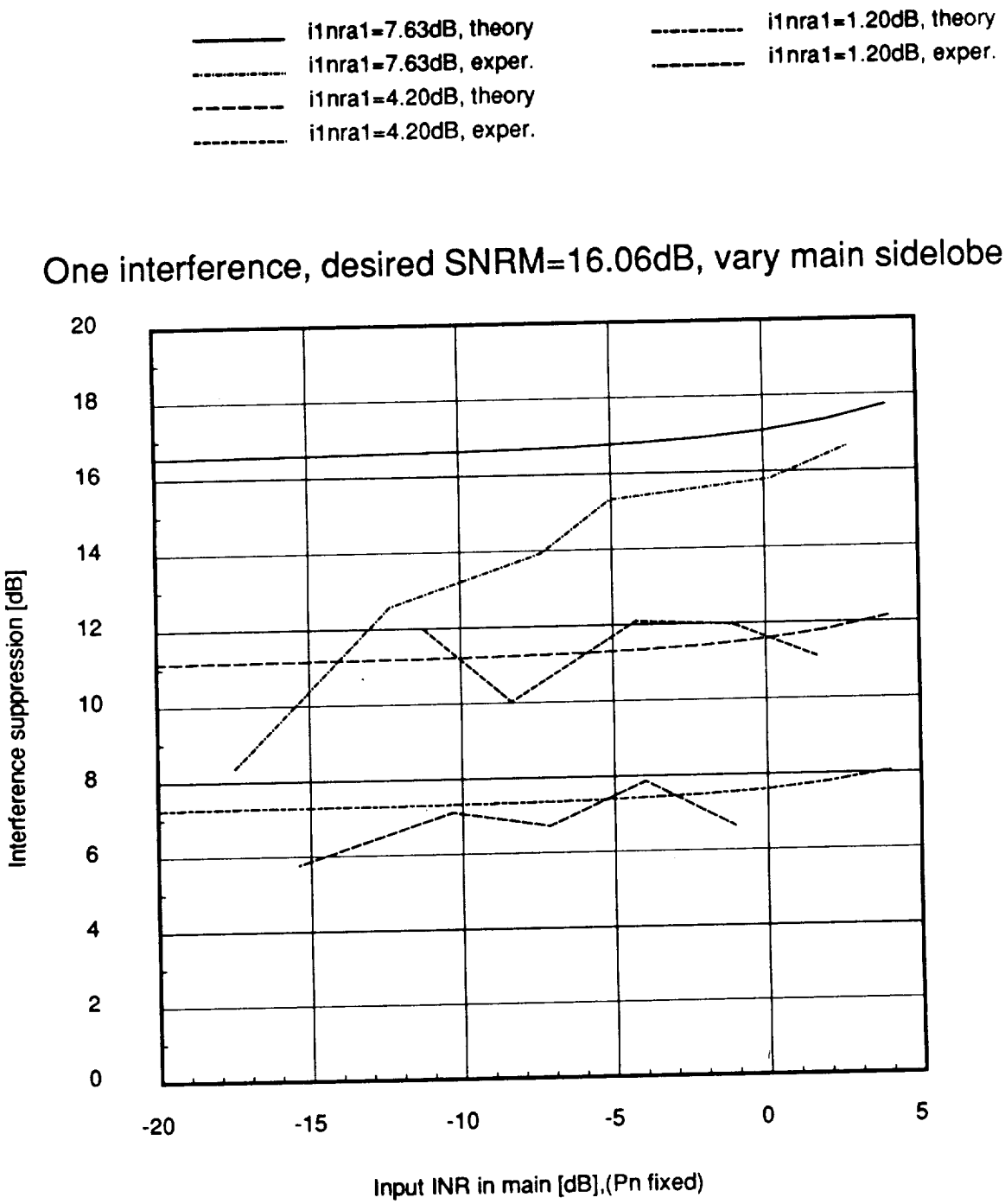


Figure 4.5: Experiment 4: Suppression of the first interference signal with a 3-element array, SNR(main)=16.06dB. The first auxiliary element gain is fixed and the main element sidelobe varies. The top pair of curves are theory and experiment for fixed INR(aux1)=7.63dB. The middle and bottom pairs are for INR(aux1)=4.20dB and 1.20dB,respectively.

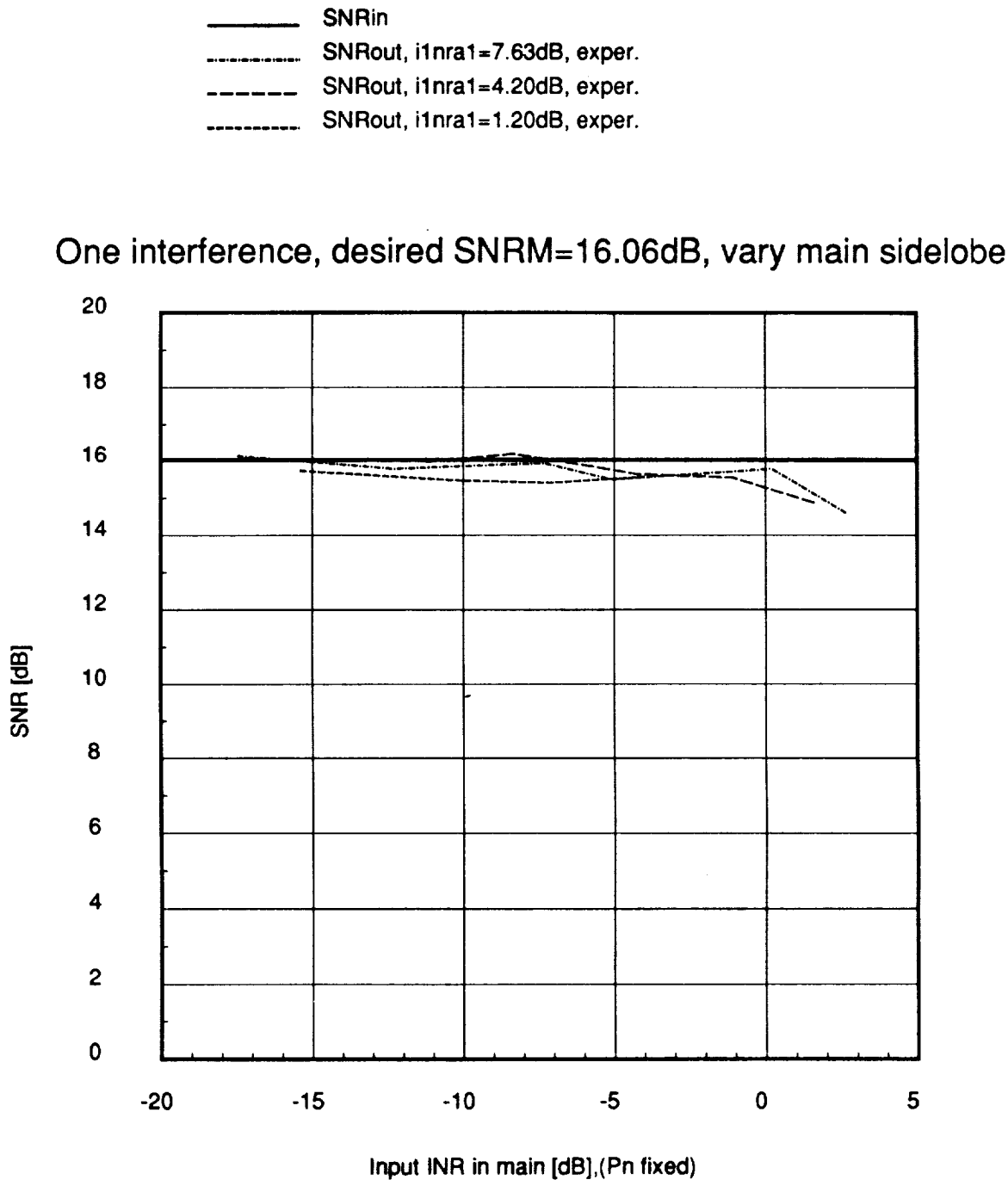


Figure 4.6: SNR(main) and SNR(out) curves for Experiment 4.

	INR(main)	SNR(main)	INR(aux1)	
			top pair of curves	bottom pair of curves
Exp. 5	-9.45	16.50	0.43	-3.57
Exp. 6	-2.68	16.50	4.72	0.53

Table 4.2: Signal levels for Experiments 5 and 6.

In all cases theory and experiment compare favorably. From Figures 4.7 and 4.9 we see that interference suppression increases as the fraction F used in the modified SMI algorithm is increased from zero towards one. For example, in Figure 4.7 with INR(aux1)=0.43dB we see that interference suppression is increased 9.5dB by changing from F=0.0 (standard SMI) to F=0.8. One can also make the observation from Figure 4.7 that a modified SMI array with low-gain auxiliary antennas (INR(aux1)=-3.57) and F=0.6 performs the same with respect to interference suppression as a standard SMI array with auxiliary antennas of 4dB (=0.43+3.57) higher gain. Similar observations can be made about Figure 4.9 of Experiment 6. From the SNR plots of Figures 4.8 and 4.10 we see that for F<0.9 there is less than a 2.5dB loss of SNR from input to output. Experiments 5 and 6 have demonstrated that modified SMI yields increased suppression of weak interference with only a small loss in desired signal power.

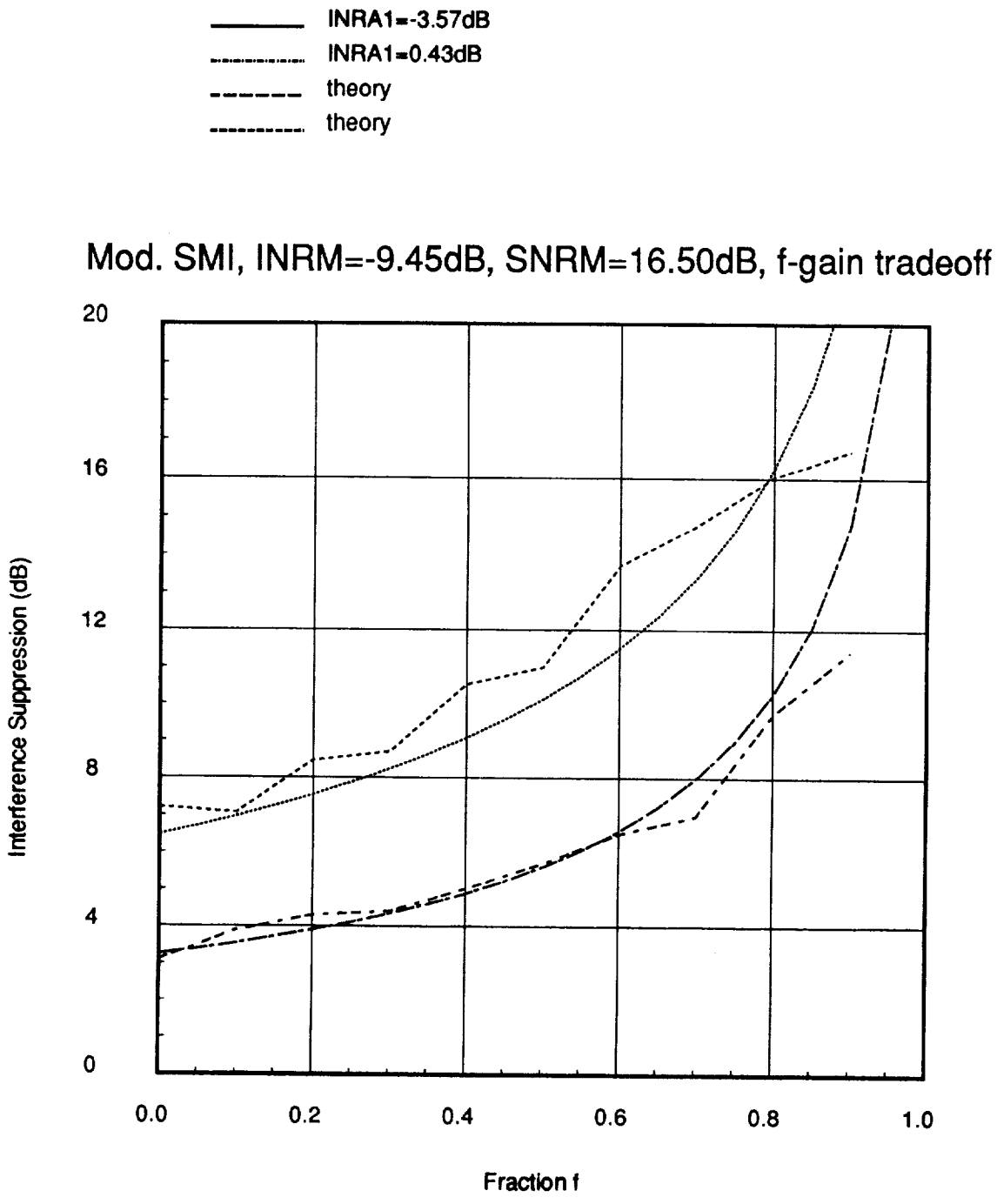


Figure 4.7: Experiment 5: Suppression of the first interference signal versus fraction F using a 3-element modified SMI array with desired SNR(main)= 16.50dB. INR(main)=-9.45dB. Bottom pair of curves has INR(aux1)= -3.57dB. Top pair has INR(aux1)=0.43dB. INR(aux2)=-20dB.

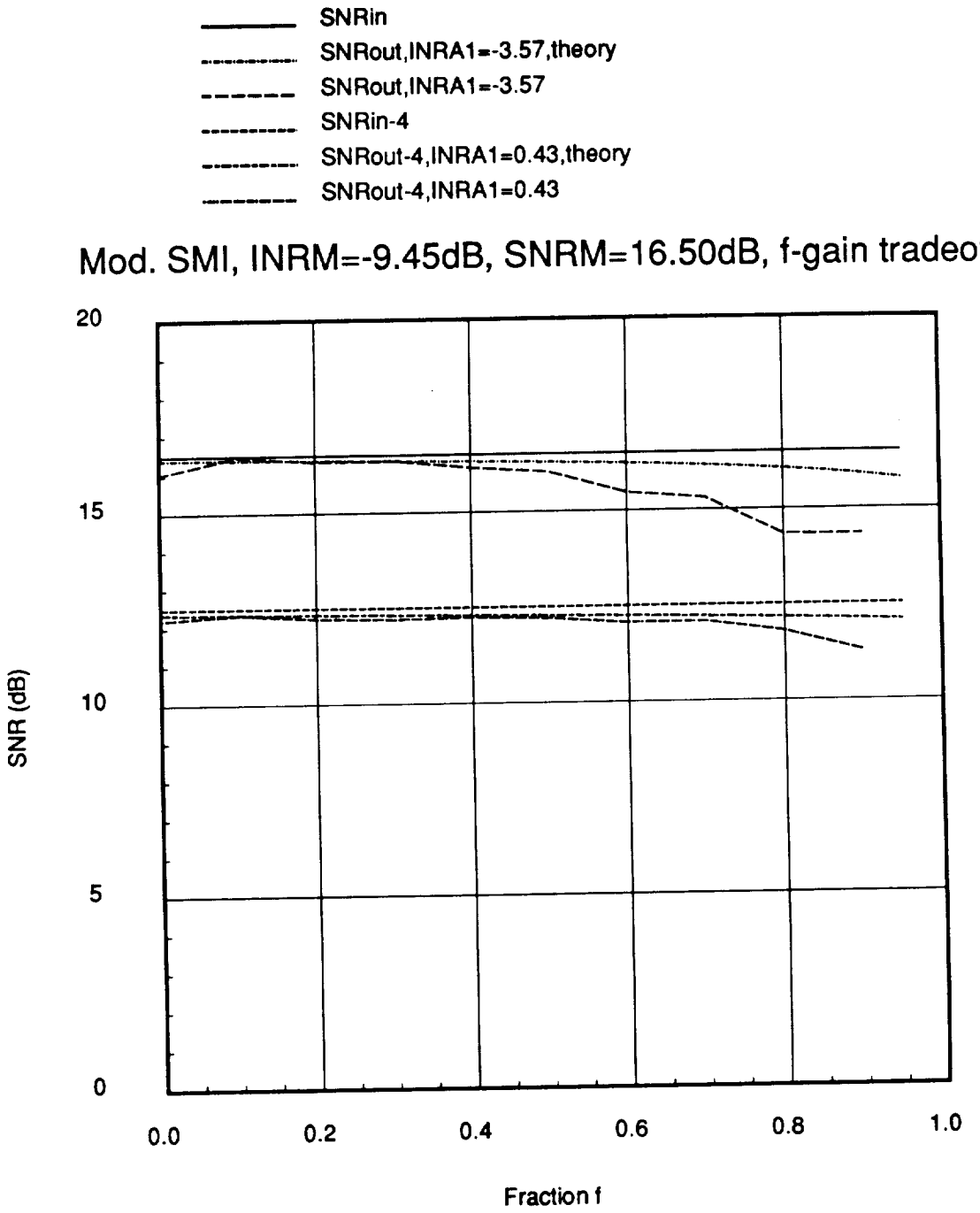


Figure 4.8: SNR curves for Experiment 5. Top three curves are input SNR(main), theoretical SNR(out), experimental SNR(out) for INR(aux1)= -3.57dB. Similarly, the bottom three curves are for INR(aux1)= 0.43dB and have been displaced -4dB for clarity.

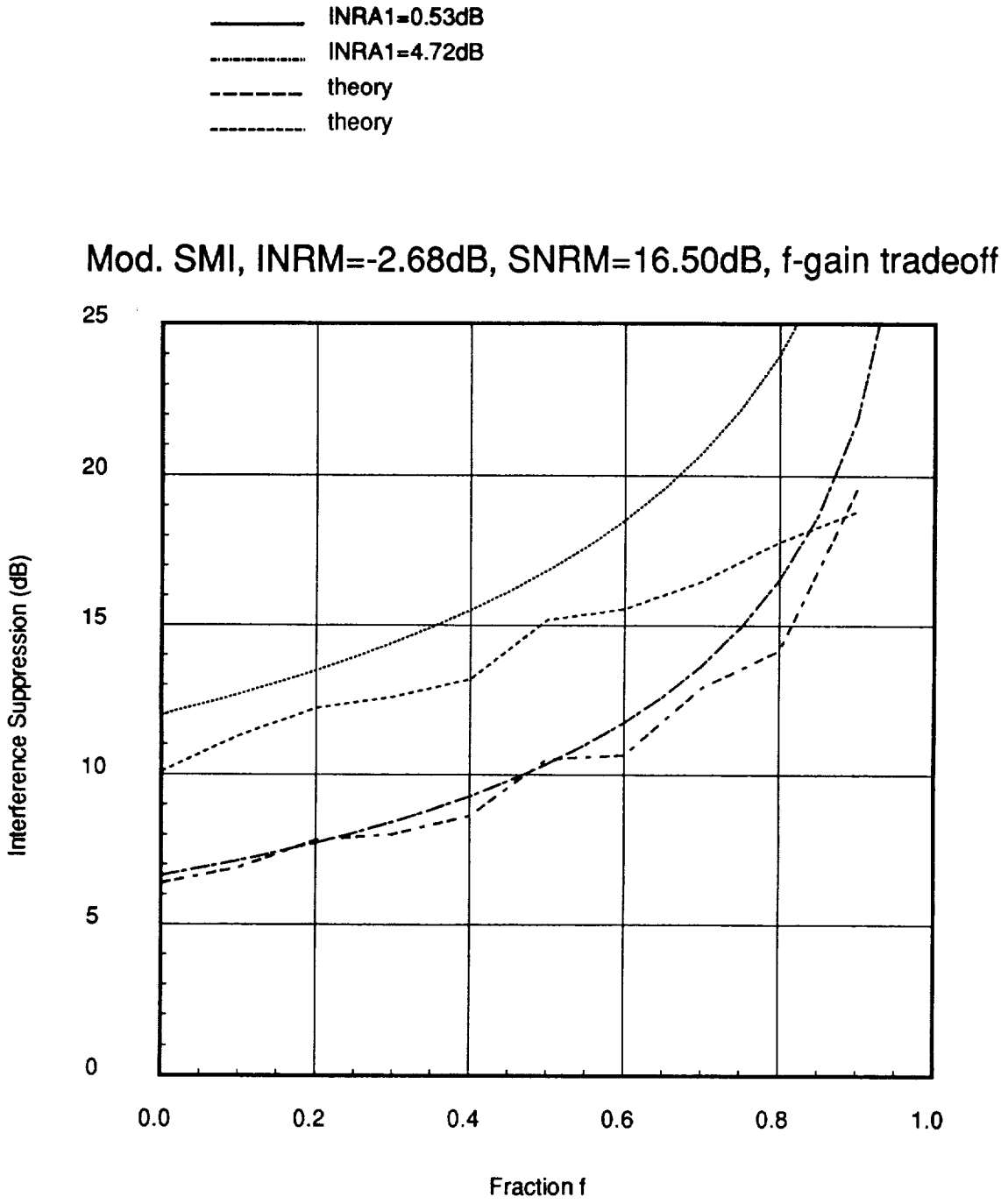


Figure 4.9: Experiment 6: Suppression of the first interference signal versus fraction F using a 3-element modified SMI array with desired SNR(main)= 16.50dB. INR(main)=-2.68dB. Bottom pair of curves has INR(aux1)= 0.53dB. Top pair has INR(aux1)=4.72dB. INR(aux2)=-17dB.

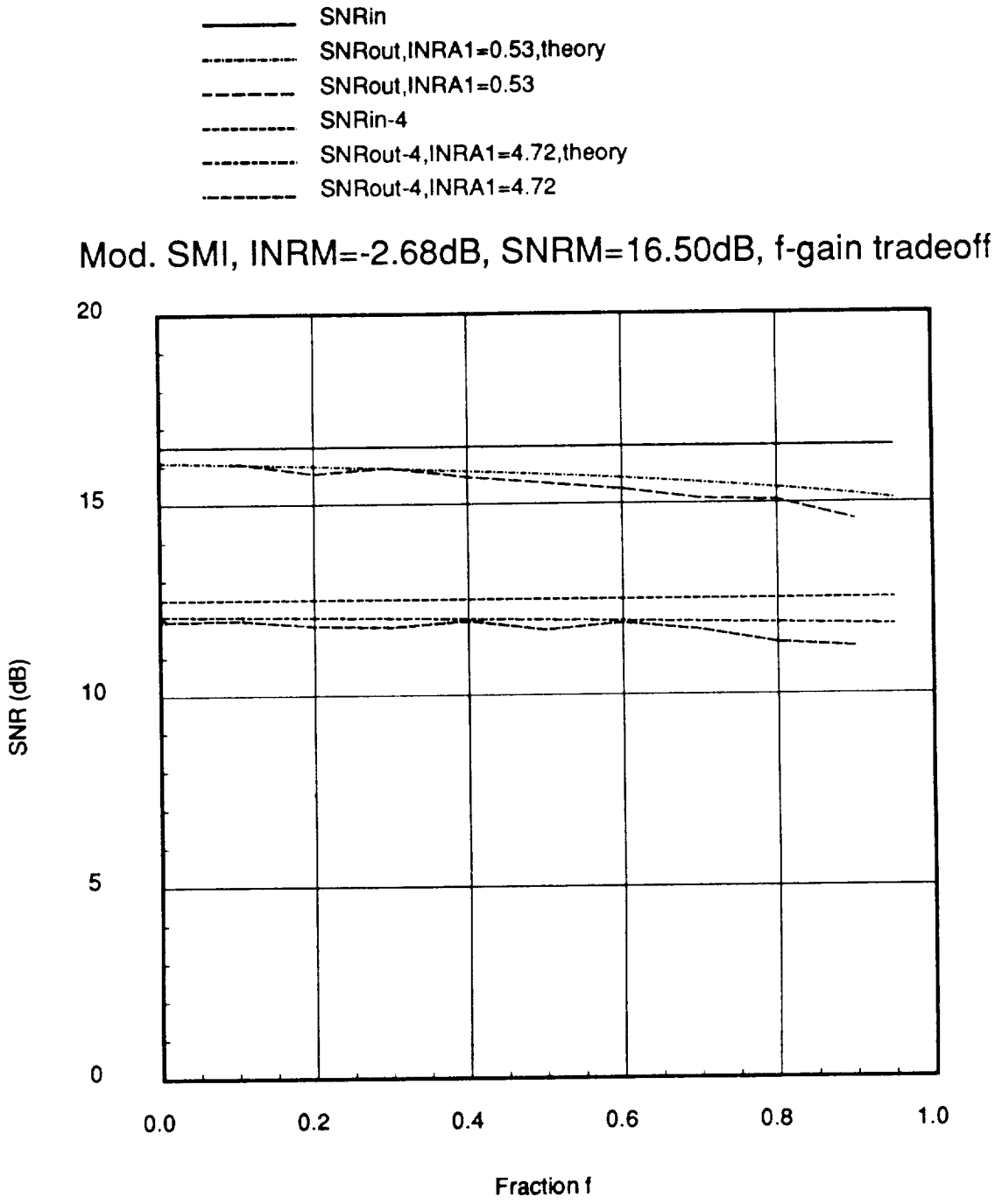


Figure 4.10: SNR curves for Experiment 6. Top three curves are input SNR(main), theoretical SNR(out), experimental SNR(out) for INR(aux1)= 0.53dB. Similarly, the bottom three curves are for INR(aux1)= 4.72dB and have been displaced -4dB for clarity.

Chapter 5

Conclusions

The problem of weak interference signals entering a satellite - ground station communication link through the sidelobes of the receive antenna has been studied experimentally. The receive antenna is made adaptive so that antenna pattern nulls can be formed in the directions of the interference signals. Standard adaptive antennas which are designed to maximize the output SINR fail to achieve adequate suppression of weak interference, especially when the auxiliary antenna elements have low gain. In a previous solution to this problem, the feedback loops used to control the array weights were modified so that the two inputs to the feedback loop correlators had uncorrelated noise components. These modified feedback loops yielded weights which were less influenced by noise and as a result adapted to null the interference. This solution had the drawback of requiring either two antennas per auxiliary element, two amplifiers per auxiliary element, or both.

In this report, an alternative solution using a modified SMI algorithm is studied experimentally. The modification involves subtracting a fraction of

the minimum eigenvalue of the estimated covariance matrix from its diagonal elements before inversion. The modified covariance matrix resembles a standard covariance matrix formed in a less noisy environment and thus "frees" the weights to further suppress the interference. The SMI modification is made in software and thus has the advantage of not requiring any additional antennas or amplifiers.

The modified SMI algorithm is implemented on a three-element experimental array system. The need for a weight (vector modulator) in the main channel is eliminated by normalizing the estimated weight vector so that the main channel weight is unity. Sampling of the main channel is accomplished by sampling the array output while the auxiliary element weights are set to zero. The signal snapshots are the input to the SMI algorithm and are formed from scaled samples. Such scaling provides a coarse adjustment for the losses present in the experimental system. The modified SMI algorithm is presented in detail. In order to quantify the performance of the experimental system the signal powers in each channel are estimated from samples. This is made possible by a special pulse modulation scheme in which the desired and interference signals are separated in time. Finally, a fully-automated, iterative procedure for system calibration is presented. In this procedure, amplitude and phase scale factors for the auxiliary signal samples are determined by comparing the weights that null the interference in a simple signal scenario with those that are estimated by the algorithm.

The experiments that are presented in this report focus on steady state performance (i.e. a large number of snapshots are used to form the weight estimate for any particular scenario). In all cases, the experimental results

compare favorably with theoretical predictions based on the ideal array model. As expected, the interference suppression increases as the power of the interference signal incident on the array increases. In particular, it is noted that interference suppression is relatively independent of the main antenna sidelobe level when that sidelobe level is small relative to the auxiliary element gain. In other words, the interference suppression is to a great extent determined by the gain of the auxiliary antennas in such situations. When low-gain auxiliary elements are used, standard SMI leads to only small amounts of interference suppression. When modified SMI is used the suppression is increased by as much as 13dB in the experiments. Thus, for large numbers of snapshots the experimental modified SMI array can provide the desired performance as predicted by theory.

Bibliography

[1] R.T. Compton, Jr., *Adaptive Antennas - Concepts and Performancd.* Englewood Clifs, NR: Prentice-Hall, 1988.

[2] I.J. Gupta and A.A. Ksienski, "Adaptive arrays for weak interfering signals," *IEEE Transactions on Antennas and Propagation*, vol. AP-34, no. 3, pp. 420–426, March 1986.

[3] J. Ward, E.K. Walton, I.J. Gupta, and A.A. Ksienski, "An experimental adaptive array to suppress weak interfering signals," *IEEE Transactions on Antennas and Propagation*," vol. 36, no. 11, pp. 1551–1559, November 1988.

[4] J. Ward, "Adaptive arrays for weak interfering signals –an experimental system," M.S. Thesis, The Ohio State University, Summer 1987.

[5] I.J. Gupta, "SMI adaptive antenna arrays for weak interfering signals," *IEEE Transaction on Antennas and Propagation*, vol. AP-34, no. 10, pp. 1237–1242, October 1986.

[6] R. Dilsavor and R. Moses, "Analysis of modified SMI method for adaptive array weight control," Technical Report 716111-6, The Ohio State University, Department of Electrical Engineering, ElectroScience Laboratory, February 1989.

[7] K. Steadman, "Adaptive array for weak interfering signals – Geostationary satellite experiments," M.S. Thesis, The Ohio State University, Summer 1989, prepared under Grant NAG 3-536, NASA Lewis Resarch Center.

[8] H.L. Van Trees, *Detection, Estimation, and Modulation Theory: Part I.* New York: Wiley, 1968.

[9] R.E. Ziemer and W.H. Tranter, *Principles of Communications — Systems, Modulation, and Noise*. Boston: Houghton Mifflin Company, 1985.

Appendix A

Experimental System Software (Fortran IV)

```
C       MASTER PROGRAM FOR RUNNING ADAPTIVE ARRAY SYSTEM
        COMMON/OUTDTA/X,AVE,AVCOR,OFF,
     X  LMAG,HMAG,LPH,HPH,NSC,CPHI
        COMMON/CTST/AVCD,AVIF,AVNSE,ST1
        COMMON/SCNRO/LATT,LPHI,PN
        COMMON/WTBLK/LDAO,WI,WQ,VMX,VMY,NW
        COMMON/SPWRS/PWR,PNSE,ITER,IFL,OFLAG,NFLAG,AIS,AIST
        COMMON/SMI/COV,NCOV,NCOV1,NCOV10,ICT
        COMPLEX X(3,64),AVCOR(2),OFF(3),AVCD(2),AVIF(2),AVNSE(2)
        COMPLEX COV(3,3),COVCOP(3,3),COVINV(3,3),STEER(3),ST1
        COMPLEX CW(3),EVEC(3,3),DET,ID(3,3),STR,CWTEST,WEIGHT(20,2)
        REAL VMX(40,4),VMY(40,4),POWERS(20,4),NCOV
        REAL EVAL(3),LVECT(3),MVECT(3),PIMIN,EIGVAL(20,3)
        REAL PWR(3,3),PNSE(3),PB(3),PA(3),AIS(3),AIST,CPHI(2)
        REAL AVE(6,3),LMAG(2),HMAG(2),LPH(2),HPH(2),PI,P(2),SIZE
        REAL LPHI(2,2),PN(4),WI(2),WQ(2),TEMPI(2),TEMPQ(2),WIO(2),WQO(2)
        INTEGER IDACO,LATT(3),LDAO,OFLAG,NW(4),ICT,ICF,IDIM
        INTEGER NCOV1,NCOV10
        LOGICAL*1 FLNME(6),FTYPE(4),FLSPEC(15),NME(9,4)
        DATA FTYPE/'.','D','A','T'/CPHI/2*0.0/
        DATA IDACO/"170440/OFF/3*(0.0,0.0)/
        DATA WI/2*0.0/WQ/2*0.0/
        DATA PI/3.1415927/NFLAG/0/
        DATA NSC/17/STEER/(0.0,0.0),(0.0,0.0),(1.0,0.0)/
C  Reading data for VMOD control
        CALL SCOPY('VM1I.DAT',NME(1,1))
        CALL SCOPY('VM1Q.DAT',NME(1,2))
        CALL SCOPY('VM2I.DAT',NME(1,3))
        CALL SCOPY('VM2Q.DAT',NME(1,4))
        DO 15 I=1,4
         OPEN(UNIT=10,NAME=NME(1,I),TYPE='OLD',FORM='UNFORMATTED')
         DO 10 J=1,40
10        READ(10,END=11) VMX(J,I),VMY(J,I)
11       NW(I)=J
```

```
        CLOSE(UNIT=10)
15      CONTINUE
C  Enable pulse generator and set THA's to TRACK mode
        CALL IPOKE(IDAC0,"40000)
25      WI(1)=0.0
        WI(2)=0.0
        WQ(1)=0.0
        WQ(2)=0.0
        CALL JWWT
C  Enter Array Simulator parameters
        CALL JWINIT
        TYPE *,' '
        TYPE *,'  Main and AUX1 elements only--------Enter 1'
        TYPE *,'  Main and AUX2 elements only--------Enter 2'
        TYPE *,'  All three elements-----------------Enter 3'
        TYPE 975
        ACCEPT 921,ICF
        IDIM=2
        IF(ICF.EQ.3)IDIM=3
C
        TYPE *,' '
        TYPE *,'ENTER NUMBER OF EIGENVECTORS TO KEEP:'
        ACCEPT 921, NEIG
C
        TYPE *,' '
        TYPE 978
        ACCEPT 922,FRAC
        TYPE*,' '
        TYPE*,' TRANSIENT RUN ? ENTER 7 IF YES, 0 IF NO:'
        ACCEPT 921, ICODE
C
        TYPE *,' '
        TYPE *,'  Enter 1 to output powers to PWRS.DAT else Enter 0 :'
        ACCEPT 921,IPOW
        IF(IPOW.EQ.1)OPEN(UNIT=10,NAME='PWRS.DAT',TYPE='NEW',
     +   FORM='FORMATTED')
C
40      TYPE 919
        PAUSE
C  Calculate beginning powers
        TYPE 879
        CALL JWISC
        DO 60 L=1,3
60       PB(L)=PWR(L,3)
        PINT=PWR(1,3)+PWR(2,3)

        TYPE 929
        PAUSE
        NFLAG=1
        TYPE 879
        CALL JWISC
        IFL=1
        CALL RDOTPT
        TYPE 928
        ACCEPT 921,IRC
        IF (IRC .EQ. 1) GOTO 40
```

```
      TYPE 965
      ACCEPT 970,NSC
      TYPE 977
      ACCEPT 921,ICT
      TYPE 979
      ACCEPT 923,LIMIT
      DO 380 I=1,3
      DO 390 J=1,3
      COV(I,J)=(0.0,0.0)
390   CONTINUE
380   CONTINUE
      NCOV1=0
      NCOV10=0
      ITER=1
C   Start adaptation loop
      TYPE 968
      TYPE 969
      TYPE 966
375   IF (ICODE .NE. 1) GOTO 490
       TYPE 967
       ICODE=0
490    IF(ICODE.NE.7)GOTO 491
       FRAC = 0.0
C--------Store present weights and apply zero weights
C       so that the main channel may be sampled as the array output.
491    DO 600 K=1,2
        TEMPI(K)=WI(K)
        TEMPQ(K)=WQ(K)
        WI(K)=0.0
        WQ(K)=0.0
600    CONTINUE
      CALL JWWT
C-------Update the covariance matrix by 12*NSC snapshots
C       then reapply adapted weights.
       CALL COVAR(NSC,CPHI,OFF)

       type*, cov(1,1),cov(1,2),cov(1,3)
       type*, cov(2,1),cov(2,2),cov(2,3)
       type*, cov(3,1),cov(3,2),cov(3,3)

        DO 610 I=1,IDIM
        IF(I.NE.3)WI(I)=TEMPI(I)
        IF(I.NE.3)WQ(I)=TEMPQ(I)
        DO 620 J=1,IDIM
        II=I
        JJ=J
        IF(ICF.EQ.1 .AND.I.EQ.2)II=3
        IF(ICF.EQ.1 .AND.J.EQ.2)JJ=3
        IF(ICF.EQ.2)II=I+1
        IF(ICF.EQ.2)JJ=J+1
        COVCOP(I,J)=COV(II,JJ)
620    CONTINUE
610    CONTINUE
       CALL JWWT
C-------Get eigvals and eigvecs of cov.
       CALL HEIGEN(COVCOP,EVEC,EVAL,IDIM)
```

```
        type *, (eval(i),i=1,idim)
C-------NORMALIZE EIGENVECTORS
        DO 623 I=1,IDIM
        SIZE=0.0
        DO 624 J=1,IDIM
        SIZE=SIZE+CABS(EVEC(J,I))**2
624     CONTINUE
        SIZE=SQRT(SIZE)
        DO 625 J=1,IDIM
        EVEC(J,I)=EVEC(J,I)/SIZE
625     CONTINUE
623     CONTINUE
C-------INVERT COVARIANCE MATRIX
802     IF(ICODE.EQ.6)FRAC=FRAC+0.1
        IF(ABS(1.0-FRAC).LT.0.05)GOTO 670
C       IF(ABS(1.0-FRAC) .LT. 0.05)FRAC=0.9
C       IF(1.1-FRAC .LT. 0.05)GOTO 670
804     DO 621 I=1,IDIM
        DO 622 J=1,IDIM
        COVINV(I,J)=(0.0,0.0)

        DO 626 K=1,NEIG
        COVINV(I,J)=COVINV(I,J)+EVEC(I,K)*CONJG(EVEC(J,K))/(EVAL(K)-FRAC
     +              *EVAL(IDIM))
626     CONTINUE
622     CONTINUE
621     CONTINUE
C-------CALCULATE COMPLEX WEIGHTS CW
        DO 630  I=1,IDIM
        CW(I)=(0.0,0.0)
        DO 631  J=1,IDIM
        STR=STEER(J)
        IF(ICF.EQ.1 .AND. J.EQ.2)STR=STEER(3)
        IF(ICF.EQ.2)STR=STEER(J+1)
        CW(I)=CW(I)+COVINV(I,J)*STR
631     CONTINUE
630     CONTINUE
C-------NORMALIZE WEIGHTS TO CW(IDIM)=main channel weight
        DO 632 I=1,IDIM
        CW(I)=CW(I)/CW(IDIM)
632     CONTINUE
C-------PICK OFF I AND Q COMPONENTS OF WEIGHTS
         DO 601 K=1,2
          TEMPI(K)=WI(K)
          TEMPQ(K)=WQ(K)
601      CONTINUE
602     DO 633 K=1,2
        WI(K)=      REAL(CW(K))
        WQ(K)=-1.0*AIMAG(CW(K))
633     CONTINUE
        IF(ICF.EQ.1)WI(2)=0.0
        IF(ICF.EQ.1)WQ(2)=0.0
        IF(ICF.EQ.2)WI(2)=WI(1)
        IF(ICF.EQ.2)WQ(2)=WQ(1)
        IF(ICF.EQ.2)WI(1)=0.0
        IF(ICF.EQ.2)WQ(1)=0.0
```

```
      DO 634 K=1,2
        IF (ABS(WI(K)).LE.1.0 .AND. ABS(WQ(K)).LE.1.0) GOTO 634
         TYPE 972,K,K
         WI(K)=TEMPI(K)
         WQ(K)=TEMPQ(K)
634   CONTINUE
        IF(ICODE.EQ.7)GOTO 637
        IF (MOD(ITER,5) .NE. 0) GOTO 635
637      DO 636 K=1,2
         IF (K .EQ. 1) TYPE 884,ITER
         TYPE 885,K,K
         TYPE 887,TEMPI(K),TEMPQ(K),WI(K),WQ(K)
         TYPE *,''
636      CONTINUE
635     CALL JWWT
        IF(ICODE.EQ.6)GOTO 710
        IF(IPOW.EQ.0) GOTO 456
        TYPE 919
        PAUSE
        TYPE 879
        NFLAG=0
        CALL JWISC
        TYPE 929
        PAUSE
        NFLAG=1
        TYPE 879
        CALL JWISC
       IF(ICODE.EQ.6)TYPE 891,FRAC,NCOV
       WRITE(10,*)NCOV,FRAC
       WRITE(10,*)(PWR(I,3),I=1,3),PNSE(3)
       WRITE(10,*)(CW(I),I=1,IDIM)
       WRITE(10,*)(EVAL(I),I=1,IDIM)
456     IF (IRSP .EQ. 0) GOTO 640
        IF (ITER .LE. 250) WRITE(10) (WI(I),WQ(I),I=1,2),
     X (AVCOR(I),I=1,2),(AVCD(I),I=1,2),(AVIF(I),I=1,2),(AVNSE(I),I=1,2)
     X,ITER
640     IMCH=ITTINR()
        ICRLF=ITTINR()
        ICRLF=ITTINR()
        IF (IMCH .LT. 0) GOTO  800
        IF (IMCH .EQ. 83) GOTO 650
        TYPE 878
        GOTO 800
650     IF(MOD(ITER,5) .EQ. 0) GOTO 670
        TYPE 884,ITER
        DO 660 K=1,2
         TYPE 885,K,K
         TYPE 887,TEMPI(K),TEMPQ(K),WI(K),WQ(K)
         TYPE *,''
660     CONTINUE
670     TYPE 888

        TYPE *,'   Continue adaptation-------------Enter 1'
        TYPE *,'   Exit adaptation loop------------Enter 2'
        TYPE *,'   Calculate jammer suppression-----Enter 3'
        TYPE *,'   Output intermediate results------Enter 4'
```

```
      TYPE *,'   Compute calibration factors------Enter 5'
      TYPE *,'   Vary frac-----------------------Enter 6'
      TYPE *,'   Number of eigenvectors----------Enter 8'
      TYPE *,'   Assign steering vector,use COV---Enter 9'
      TYPE *,'   Reset steering vector to 001-----Enter 10'
      TYPE 880
      ACCEPT 881,ICODE
      IF (ICODE .EQ. 1) GOTO 800
      IF (ICODE .EQ. 5) GOTO 801
      IF (ICODE .EQ. 6) FRAC=-0.1
      IF (ICODE .EQ. 6) GOTO 802
      IF (ICODE .EQ. 8) TYPE *,'ENTER NUMBER OF EIGENVECTORS TO KEEP:'
      IF (ICODE .EQ. 8) ACCEPT 921, NEIG
      IF (ICODE .EQ. 8) GOTO 670
      IF (ICODE .EQ. 9)STEER(1)=COV(1,1)
      IF (ICODE .EQ. 9)STEER(2)=COV(2,1)
      IF (ICODE .EQ. 9)STEER(3)=COV(3,1)
      IF (ICODE .EQ. 9) GOTO 670
      IF (ICODE .EQ. 10)STEER(1)=(0.0,0.0)
      IF (ICODE .EQ. 10)STEER(2)=(0.0,0.0)
      IF (ICODE .EQ. 10)STEER(3)=(1.0,0.0)
      IF (ICODE .EQ. 10) GOTO 670
      IF (ICODE-3) 850,710,720
710   TYPE 919
      PAUSE
      TYPE 879
      NFLAG=0
      CALL JWISC
      DO 715 I=1,3
715    AIS(I)=10.0*ALOG10(PB(I)/PWR(I,3))
      AIST=10.0*ALOG10(PINT/(PWR(1,3)+PWR(2,3)))
      TYPE 929
      PAUSE
      NFLAG=1
      TYPE 879
      CALL JWISC
      TYPE 890,AIST
      TYPE 889,(AIS(I),I=1,3)
      IF(ICODE.EQ.6)TYPE 891,FRAC,NCOV
891   FORMAT(' FRAC = ',F8.5,' NCOV = ',F7.0)
      IF(ICODE.EQ.6)GOTO 802
      GOTO 670
720   CALL RDOTPT
      GOTO 670
800   IF(LIMIT.LE.0)GOTO 803
      LIM=LIMIT
      IF(NCOV .GE. FLOAT(LIMIT)) LIMIT=LIMIT+12*NSC
      IF(NCOV.GE. FLOAT(LIM)) GOTO 670
803   IF(ICODE.NE.7) ITER=ITER+1
      IF(ICODE.NE.7) GOTO 375
      IF(FRAC.EQ.0.9)ITER=ITER+1
      IF(FRAC.EQ.0.9) GOTO 375
      IF(FRAC.EQ.0.7) FRAC=0.9
      IF(FRAC.EQ.0.0) FRAC=0.7
      GOTO 804
801   SCBEST=1.0
```

```
      DO 812 J=1,6
      PIMIN=999999.999
      DO 811 I=-10,10
      THETAD=FLOAT(I)
      THETA=PI/180.*THETAD
      CI=COS(THETA)
      CQ=SIN(THETA)
      CWTEST=CW(1)*CMPLX(CI,CQ)*SCBEST
      WI(ICF)=      REAL(CWTEST)
      WQ(ICF)=-1.0*AIMAG(CWTEST)
      CALL JWWT
      CALL JWISC
      IF(PWR(ICF,3).LT.PIMIN)PHBEST=THETAD
      IF(PWR(ICF,3).LT.PIMIN)PIMIN=PWR(ICF,3)
      TYPE*,THETAD,PHBEST
811   CONTINUE
      TYPE 462
462   FORMAT(' ENTER PHBEST BETWEEN -10. AND 10.')
      ACCEPT 924,PHBEST
      PIMIN=9999999.999
      SC=0.90
      DO 810 I=1,21
      IF( CABS( CW(1)*SC ) .GE. 1.0)GOTO 812
      CI=COS(PHBEST*PI/180.)
      CQ=SIN(PHBEST*PI/180.)
      CWTEST=CW(1)*SC*CMPLX(CI,CQ)

      WI(ICF)=      REAL(CWTEST)
      WQ(ICF)=-1.0*AIMAG(CWTEST)
      CALL JWWT
      CALL JWISC
      IF(PWR(ICF,3).LT.PIMIN)SCBEST=SC
      IF(PWR(ICF,3).LT.PIMIN)PIMIN=PWR(ICF,3)
      TYPE*,SC,SCBEST
      SC=SC+.01
810   CONTINUE
      TYPE 463
463   FORMAT(' ENTER SCBEST BETWEEN 0.9 AND 1.1')
      ACCEPT 924,SCBEST
812   CONTINUE
      GOTO 670
850   IF (IRSP .EQ.1) CLOSE(UNIT=10)
      IF(IPOW.EQ.1)CLOSE(UNIT=10)
      TYPE 877
      TYPE *,'   End program execution----------Enter 2'
      TYPE 880
      ACCEPT 881,IECD
      IF (IECD .EQ. 2) GOTO 990
      IF (IECD .NE. 1) GOTO 850
      GOTO 25
877   FORMAT(//'  Conduct another test-----------Enter 1')
878   FORMAT(/' INADVERTANT CHARACTER ENTERED'/)
879   FORMAT(/' Power levels being computed . . . .')
880   FORMAT(' Enter code for desired option: ',$)
881   FORMAT(I2)
884   FORMAT(//10X,'Iteration number ',I3)
```

```
885     FORMAT(/13X,'LAST W(',I1,')',17X,'CURRENT W(',I1,')')
887     FORMAT(5X,2(E10.3,2X),4X,2(E10.3,2X))
888     FORMAT(//' OPTIONS:')
889     FORMAT(/' Jammer-1: ',F8.4,'  Jammer-2: ',F8.4,'  Desired
     X Sig: ',F8.4/)
890     FORMAT(/' INTERFERENCE SUPPRESSION: ',F8.4/)
918     FORMAT(/' For phase calibration, set all leakage attenu
     Xators to 70dB.')
919     FORMAT(/' Remove noise and hit [RETURN]:')
920     FORMAT(/' Enter code for storing weight values: ',$)
921     FORMAT(I2)
922     FORMAT(F3.1)
923     FORMAT(I6)
924     FORMAT(F6.3)
925     FORMAT(/' Enter filename(<=6 characters): ',$)
926     FORMAT(Q,8A1)
927     FORMAT(//' To store(on floppy) weight vector and its
     X  progression')
928     FORMAT(/' Enter 1 to recalculate else enter 0: ',$)
929     FORMAT(/' Restore noise and hit [RETURN]:')
933     FORMAT(/' Remove interference and noise for steering vector')
934     FORMAT('  calculation and hit [RETURN]:')
936     FORMAT(/' Restore interference and noise and hit [RETURN]')
937     FORMAT(/' Enter 1 to compute and use steering vector
     X else enter 0: ',$)
964     FORMAT(//' Enter loop gain: ',$)
965     FORMAT(//' Enter number of scans to average over: ',$)
966     FORMAT(//' Beginning Adaptation...'/)
967     FORMAT(//' Continuing Adaptation...'/)
968     FORMAT(//' Type ''S'' and hit [RETURN] to suspend iteration')
969     FORMAT('  at any point and go to OPTION menu')
970     FORMAT(I3)
971     FORMAT(F10.6)
972     FORMAT(/' WEIGHT #',I1,' TOO LARGE FOR VMOD-',I1,' TO HANDLE'/)
973     FORMAT('    AUX-1= ',F7.2,3X,' AUX-2= ',F7.2/)
974     FORMAT(/' Feedback loop differential phase shifts:')
975     FORMAT(' Enter array configuration code: ',$)
976     FORMAT(/' Steering vector being computed...'/)
977     FORMAT(/' Do not include crossterms------------ENTER 0'/
     X          ' Include d-j and j-j crossterms-------ENTER 1'/
     X          ' Enter covariance calculation method: ',$)
978     FORMAT(' Enter value for FRAC : ',$)
979     FORMAT(' Enter # of snapshots to take (<=0 for no limit): ',$)
990     TYPE *,' THE END'
999     STOP
        END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

C  Subprogram JWINIT--set array simulator parameters
C  Program to test control of programmable attenuators of array
C  simulator using parallel output port(DRV11)
        SUBROUTINE JWINIT
        COMMON/SCNRO/LATT,LPHI,PN
        INTEGER DRBUF,IVAL,LATT(3),IM(3),IP(3)
        REAL LPHI(2,2),PN(4)
```

```
      DATA DRBUF/"167772/IBN0/15*0/

      DATA IM,IP/3*0,3*0/K2/-1/
      IVAL=0
C  Desired signal parameters:  array index 1
C  Jammer 1 signal parameters: array index 2
C  Jammer 2 signal parameters: array index 3
20    FORMAT(//' Enter Main signal leakage attenuation(dB): ',$)
21    FORMAT(' Enter Jammer-',I1,' leakage attenuation(dB):  ',$)
25    FORMAT(I3)
30    FORMAT(' Set Main signal leakage rotary attenuator to ',I2,'dB')
31    FORMAT(' Set Jammer-',I1,' leakage rotary attenuator to ',I2,'dB')
32    FORMAT('   and hit [RETURN]')
40    FORMAT(' Enter phase shift from Jammer-',I1,
     X ' to MAIN(degrees):',$)
41    FORMAT(' Enter phase shift from Jammer-',I1,' to AUX-',I1,': ',$)
45    FORMAT(F8.2)
50    FORMAT(' Enter Main channel noise power(dBm): ',$)
51    FORMAT(' Enter Correlator channel noise power(dBm): ',$)
52    FORMAT(' Enter AUX-',I1,' signal channel noise power(dBm): ',$)
55    FORMAT(F7.2)
      DO 500 J=1,3
       K=J-1
       IF (J .EQ. 1) TYPE 20
       IF (J .NE. 1) TYPE 21,K
       ACCEPT 25,LATT(J)
        IM(J)=INT(0.1*LATT(J)-2.15)*10
        IF( IM(J) .LT. 0) IM(J)=0
        IP(J)=LATT(J)-IM(J)
        IF (J .EQ. 1) TYPE 30,IM(J)
        IF (J .NE. 1) TYPE 31,K,IM(J)
        TYPE 32
        PAUSE
       IVAL=IVAL+IP(J)*2**(K*5)
500   CONTINUE
      CALL IPOKE(DRBUF,IVAL)
C   Enter in leakage phase shifts, noise powers that are manually set:
      DO 600 J=1,2
       K2=-1.0*K2
       TYPE 40,J
       ACCEPT 45,LPHI(J,1)
       TYPE 41,J,J+K2
       ACCEPT 45,LPHI(J,2)
600   CONTINUE
      TYPE *,'   '
      DO 700 J=1,4
       IF (J .EQ. 1) TYPE 50
       IF (J .EQ. 4) TYPE 51
       IF (J .NE. 1 .AND. J .NE.4) TYPE 52,J-1
       ACCEPT 55,PN(J)
700   CONTINUE
      RETURN
      END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
C       SUBPROGRAM TO SCAN AND A/D CONVERT SAMPLES FROM A WAVEFORM
        SUBROUTINE JWSCAN
        COMMON/OUTDTA/X,AVE
        COMMON/WTBLK/LDAO
        COMPLEX X(3,64)
        REAL D(6,64),AVE(6,3)
        INTEGER ADCSR,CLCSR,ADDBR,CLBPR,KCSR,IAD(6,64),IDACO,LDAO
        DATA ADCSR,ADDBR,CLCSR,CLBPR/"170400,"170402,"170420,"170422/
        DATA IDACO/"170440/
        DO 10 IFX=1,6
         DO 9 JFX=1,3
9         AVE(IFX,JFX)=0.0
10      CONTINUE
11      FORMAT (I5)
C  Configure clock for external start(ST2) and single interval mode
        KCSR=8200
        ISTO="40000+LDAO
        ICNT2=-63
        CALL IPOKE(CLBPR,ICNT2)
C   Using TTL bit from DACO holding register to start pulse generator(ACT.LOW)
C     and put THA's to track mode
        CALL IPOKE(IDACO,ISTO)
        CALL IPOKE(CLCSR,KCSR)
C   Loop to check A/D done bit and read converted values
       DO 100 J=1,64
50      IC=IPEEKB(CLCSR)
C   Check clock overflow bit, then flag overrun bit
        IF (IC .LT. 128) GOTO 50
        IF (IPEEKB("170421) .AND. "20) GOTO 900
        CALL IPOKE(CLCSR,KCSR)
        CALL IPOKE(CLBPR,ICNT2-J)
        DO 95 K=1,6

         IASR=K*256+1
         CALL IPOKE(ADCSR,IASR)
C   Check for A/D done bit before reading converted data
60       IF (IPEEKB(ADCSR) .LT. 128) GOTO 60
C        IF (IPEEKB("170401) .GE. 128) GOTO 910
         IAD(K,J)=IPEEK(ADDBR)
95      CONTINUE
C   Reset THA's back to TRACK
        CALL IPOKE(IDACO,LDAO)
        CALL IPOKE(IDACO,ISTO)
100    CONTINUE
C   Disable clock starts, leaving pulse gen running
       CALL IPOKE(CLCSR,0)
       CALL IPOKE(IDACO,LDAO)
C   Convert data from offset binary
       DO 200 J2=1,64
        DO 195 K2=1,6
C   SCALE ACCOUNTS FOR LOSSES THROUGH VMODS ,POWER DIVIDERS
C   AND DIFFERENT AMPLIFIER SETTINGS AT THE A/D CONVERTERS.
        IF(K2.LE.4)SCALE=0.35481
        IF(K2.GT.4)SCALE=1.4142
        D(K2,J2)=(IAD(K2,J2)-"4000)*0.0025*SCALE
C       TYPE *,K2,J2,D(K2,J2)
```

```
195     CONTINUE
C--CALIBRATE (FINE-TUNE) THE SAMPLES
        ALPHA=D(1,J2)
        BETA =D(2,J2)
        D(1,J2)=(-.87618)*ALPHA-(.33075)*BETA
        D(2,J2)=(.33075)*ALPHA +(-.87618)*BETA
        ALPHA=D(3,J2)
        BETA =D(4,J2)
        D(3,J2)=(-.43634)*ALPHA-(.87702)*BETA
        D(4,J2)=(.87702)*ALPHA +(-.43634)*BETA
        CINV=-1.0
        DO 199 J=1,3
         IF (J .EQ. 3) CINV=1.0
         X(J,J2)=CINV*CMPLX(D(2*J-1,J2),D(2*J,J2))
199     CONTINUE
200     CONTINUE
C   Averaging pulse levels for comparison
C   MAIN--from samples 2-13
        DO 450 I=1,6
        DO 300 J2=2,13
300      AVE(I,3)=AVE(I,3)+D(I,J2)
C   J1--from samples 17,28
        DO 350 J2=17,28
350      AVE(I,1)=AVE(I,1)+D(I,J2)
C   J2--from samples 33,44
        DO 400 J2=33,44
400      AVE(I,2)=AVE(I,2)+D(I,J2)
        AVE(I,1)=AVE(I,1)/12.
        AVE(I,2)=AVE(I,2)/12.
        AVE(I,3)=AVE(I,3)/12.
450     CONTINUE
        GOTO 999
900     TYPE *,' FLAG OVERRUN BIT SET IN KWCSR','J=',J
999     RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C   Program to calculate signal and noise power at array output,
C    from samples taken with JWSCAN, then calculate J/N ratios
C    and the interference suppresion
        SUBROUTINE JWISC
        COMMON/OUTDTA/X,AVE,AVCOR,OFF
        COMMON/SPWRS/PWR,PNSE,ITER,IFL,OFLAG,NFLAG
        COMPLEX X(3,64),OFF(3),Y,AVCOR(2)
        REAL PWR(3,3),PNSE(3),AVE(6,3),S(3,3,2),OFFR(3),OFFI(3)
        INTEGER IFL,OFLAG,NFLAG
C   Enter number of samples
20      FORMAT(/' Enter number of scans to average over ')
21      FORMAT ('    for power calculations: ',$)
25      FORMAT(I3)
        TYPE 20
        TYPE 21
        ACCEPT 25,NUM
C       NUM=100
        NSMP=NUM*12.0
```

```
        NSMP2=NUM*16.0
        IF (NFLAG .EQ. 1) GOTO 853
        DO 15 J=1,3
         DO 10 K=1,3
          PWR(J,K)=0.0
          S(J,K,1)=0.0
          S(J,K,2)=0.0

10       CONTINUE
         OFF(J)=(0.0,0.0)
15      CONTINUE
         DO 810 J=1,NUM
          CALL JWSCAN
          DO 800 KCH=1,3
           DO 90 K=47,62
90          OFF(KCH)=OFF(KCH)+X(KCH,K)
505        DO 650 IM=1,3
            DO 600 M=1,2
             L=2*KCH-2+M
             S(IM,KCH,M)=S(IM,KCH,M)+AVE(L,IM)
600         CONTINUE
650        CONTINUE
800       CONTINUE
810      CONTINUE
        DO 850 M=1,3
         OFF(M)=OFF(M)/NSMP2
         OFFR(M)=REAL(OFF(M))
         OFFI(M)=AIMAG(OFF(M))
         DO 840 N=1,3
          S(N,M,1)=S(N,M,1)/NUM
          S(N,M,2)=S(N,M,2)/NUM
          PWR(N,M)=(S(N,M,1)-OFFR(M))**2+(S(N,M,2)-OFFI(M))**2
840      CONTINUE
850     CONTINUE
        GOTO 900
C  Noise power calculations over portion of PRP when no signal present.
853     DO 854 M=1,3
854      PNSE(M)=0.0
855     DO 890 N=1,NUM
         CALL JWSCAN
         DO 888 M=1,3
          DO 870 K=47,62
           Y=X(M,K)-OFF(M)
           PNSE(M)=PNSE(M)+CABS(Y)**2
870       CONTINUE
888      CONTINUE
890     CONTINUE
891     DO 895 M=1,3
895      PNSE(M)=PNSE(M)/NSMP2 *3.0
C--------THE FACTOR OF 3.0 ABOVE TAKES DUTY CYCLE INTO ACCOUNT
900     TYPE 910,(PWR(N,1),N=1,3)
        TYPE 911,(PWR(N,2),N=1,3)
        TYPE 912,(PWR(N,3),N=1,3)
        TYPE 914,(OFF(M),M=1,3)
        IF (NFLAG .EQ. 1) TYPE 913,(PNSE(N),N=1,3)
910     FORMAT(/5X,'AUX-1 POWERS:',3(2X,E12.5))
```

```
911     FORMAT(/5X,'AUX-2 POWERS:',3(2X,E12.5))
912     FORMAT(/5X,'ARRAY OUT POWERS:',3(2X,E12.5))
913     FORMAT(/5X,'NOISE POWERS:',3(2X,E12.5))
914     FORMAT(/5X,'OFFSETS:',6E10.3)
        NFLAG=0
9999    RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        SUBROUTINE COVAR(NSC,CPHI,OFF)
        COMMON/SMI/COV,NCOV,NCOV1,NCOV10,ICT
        COMMON/OUTDTA/X,AVE
        COMMON/WTBLK/LDAO
        INTEGER NCOV1,NCOV10,LDAO,NSC,I,J,K,ICT
        REAL    CI,CQ,CPHI(2),AVE(6,3),RV,PI,NCOV
        COMPLEX COV(3,3),SS(3),X(3,64),OFF(3)
        COMPLEX PHRAN1,PHRAN2,PHRAN3
        DATA IRAN1/0/IRAN2/0/
        PI=4*ATAN(1.0)
C-------CHANGE COV FROM A COVARIANCE MATRIX TO A SUM OF
C       SNAPSHOT OUTER PRODUCTS
        DO 8 J=1,3
        DO 9 K=1,3
        COV(J,K)=COV(J,K)*(FLOAT(NCOV1)+FLOAT(NCOV10)*10000.)
 9      CONTINUE
 8      CONTINUE
        IF(ICT.EQ.0)JMAX=44
        IF(ICT.EQ.1)JMAX=12
        CI=1.0
        CQ=0.0
C-------THE 100 LOOP INCREASES THE NUMBER OF SNAPSHOTS OVER WHICH THE
C       COVARIANCE MATRIX COV IS AVERAGED BY NSC*12.
        DO 100 I=1,NSC     !SCAN NUMBER
        CALL JWSCAN        !GET SAMPLES ACROSS ONE PRP,STORE IN X ARRAY
C----------THE 10 LOOP ADDS 12 SNAPSHOT OUTER PRODUCTS TO COV.
           DO 10 J=1,JMAX   !'SNAPSHOT' NUMBER FOR SCAN I
C------------NEED 3 PHASE-AT-SAMPLE-TIME RV'S FOR EACH SNAPSHOT

C             (ONE FOR EACH SIGNAL COMPONENT)
              RV=2*PI*RAN(IRAN1,IRAN2)
              PHRAN1=CMPLX(COS(RV),SIN(RV))
              IF(ICT.EQ.0)GOTO 11
              RV=2*PI*RAN(IRAN1,IRAN2)
              PHRAN2=CMPLX(COS(RV),SIN(RV))
              RV=2*PI*RAN(IRAN1,IRAN2)
              PHRAN3=CMPLX(COS(RV),SIN(RV))
C-------------THE 20 LOOP CREATES A SNAPSHOT VECTOR
C             WHICH LEADS TO CROSSTERMS
              DO 20 K=1,3 !SNAPSHOT ELEMENT NUMBER
C-------------ADD IN THE DESIRED, J1, AND J2 COMPONENTS OF THE
C             SIGNAL SNAPSHOT VECTOR SS.
C-------------GIVE EACH SIGNAL COMPONENT A RANDOM PHASE AT SAMPLE TIME
C             WHILE MAINTAINING INTERELEMENT PHASE RELATIONSHIPS.
              SS(K)=(0.0,0.0)
              SS(K)=SS(K)+(X(K,1+J)-OFF(K))*CMPLX(CI,CQ)*PHRAN1
```

```
                 SS(K)=SS(K)+(X(K,16+J)-OFF(K))*CMPLX(CI,CQ)*PHRAN2
                 SS(K)=SS(K)+(X(K,32+J)-OFF(K))*CMPLX(CI,CQ)*PHRAN3
 20              CONTINUE
                 IF(ICT.EQ.1)GOTO 12
 11              IF(J.EQ.1)GOTO 10
                 IF(13.LT.J .AND. J.LT.17)GOTO 10
                 IF(28.LT.J .AND. J.LT.33)GOTO 10
C-------------THE 25 LOOP CREATES A SNAPSHOT VECTOR
C                WHICH LEADS TO NO S|Ji OR J1|J2 CROSSTERMS
                 DO 25 K=1,3 !SNAPSHOT ELEMENT NUMBER
                 SS(K)=( X(K,J)-OFF(K) )*CMPLX(CI,CQ)*PHRAN1
 25              CONTINUE
C-------------LOOPS 30 AND 40 ADD A SINGLE SNAPSHOT OUTER PRODUCT
C                TO COV.
 12              DO 30 L=1,3
                 DO 40 M=1,3
                 COV(L,M)=COV(L,M) + SS(L)*CONJG(SS(M))
 40              CONTINUE
 30              CONTINUE
 10           CONTINUE
 100       CONTINUE
C-------UPDATE THE NUMBER OF SNAPSHOTS USED IN THE COV MATRIX AVERAGE
           NCOV1 = NCOV1 + 12*NSC
           IF(NCOV1.GE.10000)NCOV10=NCOV10+1
           IF(NCOV1.GE.10000)NCOV1=NCOV1-10000
           NCOV=FLOAT(NCOV1)+FLOAT(NCOV10)*10000.
           TYPE *,NCOV,NSC
C-------DIVIDE THROUGH BY THE NUMBER OF SNAPSHOTS IN THE AVERAGE.
           DO 50 J=1,3
           DO 60 K=1,3
           COV(J,K)=COV(J,K)/(FLOAT(NCOV1) + FLOAT(NCOV10)*10000.)
60         CONTINUE
50         CONTINUE
           RETURN
           END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

           SUBROUTINE HEIGEN(H,EVEC,EVAL,N)
           COMPLEX H(3,3),EVEC(3,3),B(6,6),W(6)
           COMPLEX PUNI(6,6),ALAM,P1(6,6),A(6,6)
           REAL ID(6,6),X,C(36),R(36),EVAL(3),ALPHA
           DATA B/36*(0.,0.)/,W/6*(0.,0.)/
           DATA PUNI/36*(0.,0.)/,ID/36*0./
           DATA P1/36*(0.,0.)/,C/36*0./
           DO 10 I=1,N
           ID(I,I)=1.
           DO 10 J=1,N
           P1(I,J)=ID(I,J)
10         B(I,J)=H(I,J)
           DO 20 J=1,N-1
           J5=0
           DO 24 I=J+1,N
24         IF(B(I,J).NE.(0.,0.))J5=1
           IF(J5.EQ.0)GOTO 20
           DO 15 I=1,N
```

```
      DO 15 K1=1,N
15    PUNI(I,K1)=ID(I,K1)
      K=0.
      DO 30 I=J+1,N
30    K=CABS(B(I,J))**2+K
      K=SQRT(K)
      J5=0
      DO 150 I=J+2,N
150   IF(B(I,J).NE.(0.,0.))J5=1
      IF((J5.EQ.0).AND.ABS(K-REAL(B(J+1,J))).LT. 0.1E-10) GOTO 20
      ALPHA=2.*K**2-2.*K*REAL(B(J+1,J))
      ALAM=ALPHA/(K**2-K*B(J+1,J))
      W(J+1)=B(J+1,J)-K

      DO 50 I=J+2,N
50    W(I)=B(I,J)
      DO 60 I=J+1,N
      DO 60 K1=J+1,N
60    PUNI(I,K1)=ID(I,K1)-(ALAM/ALPHA)*W(I)*CONJG(W(K1))
      DO 40 I=1,N
      DO 40 K1=1,N
40    A(I,K1)=(0.,0.)
      DO 100 I=1,N
      DO 100 L=1,N
      DO 100 M=1,N
      DO 100 M1=1,N
100   A(I,M1)=PUNI(I,L)*B(L,M)*CONJG(PUNI(M1,M))+A(I,M1)
      DO 80 I=1,N
      DO 80 K1=1,N
      B(I,K1)=A(I,K1)
80    A(I,K1)=(0.,0.)
      DO 110 I=1,N
      DO 110 M1=1,N
      DO 110 M2=1,N
110   A(I,M2)=P1(I,M1)*CONJG(PUNI(M2,M1))+A(I,M2)
      DO 90 I=1,N
      DO 90 M1=1,N
90    P1(I,M1)=A(I,M1)
20    CONTINUE
      K1=1
      DO 70 I=1,N
      DO 70 J=1,I
      C(K1)=REAL(B(J,I))
70    K1=K1+1
      CALL EIGEN(C,R,N,0)
      K1=1
      DO 130 I=1,N
      DO 130 J=1,N
      EVEC(I,J)=CMPLX(0.00,0.00)
      PUNI(J,I)=R(K1)
130   K1=K1+1
      DO 120 I=1,N
      DO 120 J=1,N
      DO 120 M1=1,N
120   EVEC(I,M1)=P1(I,J)*PUNI(J,M1)+EVEC(I,M1)
      DO 140 I=1,N
```

```
      K1=I+(I*I-I)/2
140   EVAL(I)=C(K1)
      RETURN
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE EIGEN(A,R,N,MV)
      REAL A(36),R(36),RANGE,FN,ANORM,ANORMX,THR,X,Y
      REAL SINX,SINX2,COSX,COSX2,SINCS
5     RANGE=1.0E-6
      IF(MV-1)10,25,10
10    IQ=-N
      DO 20 J=1,N
      IQ=IQ+N
      DO 20 I=1,N
      IJ=IQ+I
      R(IJ)=0.0
      IF(I-J)20,15,20
15    R(IJ)=1.0
20    CONTINUE
25    ANORM=0.0
      DO 35 I=1,N
      DO 35 J=I,N
      IF(I-J)30,35,30
30    IA=I+(J*J-J)/2
      ANORM=ANORM+A(IA)*A(IA)
35    CONTINUE
      IF(ANORM)165,165,40
40    ANORM=1.414*SQRT(ANORM)
      FN=1.0*N
      ANRMX=ANORM*RANGE/FN
      IND=0
      THR= ANORM
45    THR=THR/FN
50    L=1
55    M=L+1
60    MQ=(M*M-M)/2
      LQ=(L*L-L)/2
      LM=L+MQ
62    IF(ABS(A(LM))-THR)130,65,65
65    IND=1
      LL=L+LQ
      MM=M+MQ

      X=.5*(A(LL)-A(MM))
68    Y=-1*A(LM)/SQRT(A(LM)*A(LM)+X*X)
      IF(X)70,75,75
70    Y=-Y
75    SINX=Y/SQRT(2.0*(1.0+(SQRT(1.0-Y*Y))))
      SINX2=SINX*SINX
78    COSX=SQRT(1.0-SINX2)
      COSX2=COSX*COSX
      SINCS=SINX*COSX
      ILQ=N*(L-1)
      IMQ=N*(M-1)
```

```
      DO 125 I=1,N
      IQ=(I*I-I)/2
      IF(I-L)80,115,80
80    IF(I-M) 85,115,90
85    IM=I+MQ
      GO TO 95
90    IM=M+IQ
95    IF(I-L)100,105,105
100   IL=I+LQ
      GO TO 110
105   IL=L+IQ
110   X=A(IL)*COSX-A(IM)*SINX
      A(IM)=A(IL)*SINX+A(IM)*COSX
      A(IL)=X
115   IF(MV-1)120,125,120
120   ILR=ILQ+I
      IMR=IMQ+I
      X=R(ILR)*COSX-R(IMR)*SINX
      R(IMR)=R(ILR)*SINX+R(IMR)*COSX
      R(ILR)=X
125   CONTINUE
      X=2.0*A(LM)*SINCS
      Y=A(LL)*COSX2+A(MM)*SINX2-X
      X=A(LL)*SINX2+A(MM)*COSX2+X
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
      A(LL)=Y
      A(MM)=X
130   IF(M-N)135,140,135
135   M=M+1
      GO TO 60
140   IF(L-(N-1))145,150,145
145   L=L+1
      GO TO 55
150   IF(IND-1)160,155,160
155   IND=0
      GO TO 50
160   IF(THR-ANRMX)165,165,45
165   IQ=-N
      DO 185 I=1,N
      IQ=IQ+N
      LL=I+(I*I-I)/2
      JQ=N*(I-2)
      DO 185 J=I,N
      JQ=JQ+N
      MM=J+(J*J-J)/2
      IF(A(LL)-A(MM)) 170,185,185
170   X=A(LL)
      A(LL)=A(MM)
      A(MM)=X
      IF(MV-1)175,185,175
175   DO 180 K=1,N
      ILR=IQ+K
      IMR=JQ+K
      X=R(ILR)
      R(ILR)=R(IMR)
180   R(IMR)=X
```

```
185     CONTINUE
        RETURN
        END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C  Subroutine to print/type data from a particular experiment(JWMSV2)
        SUBROUTINE RDOTPT
        COMMON/OUTDTA/X,AVE,AVCOR,OFF
     X  ,LMAG,HMAG,LPH,HPH,NSC
        COMMON/SCNRO/LATT,LPHI,PN
        COMMON/SPWRS/PWR,PNSE,ITER,IFL,OFLAG,NFLAG,AIS,AIST
        COMPLEX X(3,64),AVCOR(2),OFF(2)
        REAL AVE(6,3),LMAG(2),HMAG(2),LPH(2),HPH(2),PI,AIS(3),AIST
        INTEGER LATT(3),OFLAG,NFLAG
        REAL LPHI(2,2),PN(4),PWR(3,3),PNSE(3)
        KD=-1
        ICR=13
        ILF=10

15      TYPE 111
        TYPE *,' Select Output Media:'
        TYPE *,'   Monitor display------Enter 1'
        TYPE 20
        ACCEPT 41,OFLAG
20      FORMAT(' Enter output media code: ',$)
25      TYPE *,' '
        TYPE *,' Select Output:'
        TYPE *,'   Signal Scenario-Simulator parameters--Enter 1'
        TYPE *,'   Signal/Jammer and Signal/Noise ratios'
        TYPE *,'    at each detector----------------------Enter 2'
        IF (IFL .EQ. 1) GOTO 34
        TYPE *,'   Sampled data from last scan-----------Enter 3'
        TYPE *,'   DC Offsets,Correlation range for'
        TYPE *,'    last iteration(ave over NSC scans)---Enter 4'
        TYPE *,'   Interference suppression--------------Enter 5'
34      TYPE *,'   Change output media--------------------Enter 6'
35      TYPE *,'   Done-Return to Main Program-----------Enter 7'
40      FORMAT (' Enter code for desired output: ',$)
41      FORMAT (I1)
        TYPE 40
        ACCEPT 41,IPCODE
        IF (IPCODE .EQ. 7) GOTO 9999
        IF (IPCODE .EQ. 6) GOTO 15
        IF (IPCODE .EQ. 5) GOTO 700
        IF (IPCODE .EQ. 4) GOTO 500
        IF (IPCODE-2) 150,970,950
        TYPE 111
100     FORMAT(' Main leakage attenuation= ',I2,' dB',2A1)
101     FORMAT(' Jammer-',I1,' leakage atten.= ',I2,' dB ',2A1)
105     FORMAT(' J',I1,'/MAIN leakage phase shift=',F7.2,' degrees'
     X  ,2X,2A1)
106     FORMAT(' J',I1,'/Aux-',I1,' leakage phase shift=',F7.2,' degrees'
     X,1X,2A1)
108     FORMAT(' Main channel noise power=',F6.2,' dBm',2X,2A1)
109     FORMAT(' Corr. channel noise power=',F6.2,' dBm',1X,2A1)
```

```
110     FORMAT(' Aux-',I1,' channel noise power= ',F6.2,' dBm',2A1)
111     FORMAT(/)
150     IF (OFLAG .NE. 1) GOTO 15
        TYPE 100,LATT(1)
        DO 190 J=2,3
190      TYPE 101,J-1,LATT(J)
        TYPE 111
        DO 220 J=1,2
         KD=(-1)**(J-1)
         TYPE 105,J,LPHI(J,1)
         TYPE 106,J,J+KD,LPHI(J,2)
220     CONTINUE
        TYPE 111
        TYPE 108,PN(1)
        TYPE 110,1,PN(2)
        TYPE 110,2,PN(3)
        TYPE 109,PN(4)
        GOTO 25
C  Program section to output dc offset, correlation range,etc.
500     IF(OFLAG .NE. 1) GOTO 15
        TYPE 111
        TYPE 910,NSC
        TYPE 111
        DO 550 J=1,2
550      TYPE 906,J,OFF(J)
        TYPE 911,OFF(3)
        DO 600 J=1,2
         TYPE 905,J,AVCOR(J)
         TYPE 901,J,LPH(J)
         TYPE 902,J,HPH(J)
         TYPE 903,J,LMAG(J)
         TYPE 904,J,HMAG(J)
         TYPE 111
600     CONTINUE
        GOTO 25
C  Output interference calculations
700     IF(OFLAG .NE. 1) GOTO 15
        TYPE 111
        TYPE 878
        TYPE 111
        TYPE 880,AIST
        TYPE 882,(AIS(J),J=1,3)
        TYPE 111
        GOTO 25
878     FORMAT (' After ',I3,' iterations:  ',2A1)
880     FORMAT (' INTERFERENCE SUPPRESSION: ',F8.4,2A1)
882     FORMAT (' Jammer-1: ',F8.4,' dB',' Jammer-2: ',F8.4,' dB',
     X ' Desired Sig.: ',F8.4,' dB',2A1)
901     FORMAT (' SMALLEST AUX-',I1,' CORR. PHASE=',F8.3,2A1)
902     FORMAT (' LARGEST  AUX-',I1,' CORR. PHASE=',F8.3,2A1)

903     FORMAT (' SMALLEST AUX-',I1,' CORR. MAG.= ',E8.2,2A1)
904     FORMAT (' LARGEST  AUX-',I1,' CORR. MAG.= ',E8.2,2A1)
905     FORMAT (' AVE. AUX-',I1,' CORRELATION=',2(1X,E11.4),2A1)
906     FORMAT (' AVE. AUX-',I1,' DC OFFSET=',2(1X,E10.3),2A1)
910     FORMAT (' # SCANS FOR OFFSET= 10;  # SCANS FOR AVE. CORR.=',I3
```

```
     X ,2A1)
911      FORMAT (' ARRAY OUT DC OFFSET= ',2(1X,E10.3),2A1)
950      TYPE*,'I DISABLED THIS OPTION'
         GOTO 25
970      CALL RDPRAT
         IF (IFL .EQ. 1) GOTO 9999
         GOTO 25
9999     IFL=0
         RETURN
         END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C  Subroutine rdprat, to calculate and output the S/J,S/N ratios,
C   and simulated sidelobe levels associated with a given exper-
C   iment.
         SUBROUTINE RDPRAT
         COMMON/SPWRS/PWR,PNSE,ITER,IFL,OFLAG
         INTEGER IFL,OFLAG
         REAL PWR(3,3),PNSE(3),SLL(3),PRAT(3,2),SNR(3),SIR
         PRAT(1,1)=10.0*ALOG10(PWR(1,1)/PWR(2,1))
         PRAT(1,2)=10.0*ALOG10(PWR(1,1)/PWR(3,1))
         PRAT(2,1)=10.0*ALOG10(PWR(2,2)/PWR(1,2))
         PRAT(2,2)=10.0*ALOG10(PWR(2,2)/PWR(3,2))
         PRAT(3,1)=10.0*ALOG10(PWR(3,3)/PWR(1,3))
         PRAT(3,2)=10.0*ALOG10(PWR(3,3)/PWR(2,3))
         SIR=10.0*ALOG10(PWR(3,3)/(PWR(1,3)+PWR(2,3)))
         SLL(1)=AMIN1(PRAT(1,1),PRAT(1,2))
         SLL(2)=AMIN1(PRAT(2,1),PRAT(2,2))
         SLL(3)=AMIN1(PRAT(3,1),PRAT(3,2))
C   The factor -7.27 is due to the average power of our pulsed
C    sinusoids, Pave=(Ac**2)/2*(duty cycle). The quantity PWR(I,I)
C    represents (Ac**2)/2. The duty cycle cancels when taking signal to
C    signal ratios, but must be included for SNR computations.
C    UPDATE 8-9-88 BY DILSAVOR: THIS FACTOR IS TAKEN INTO ACCOUNT
C    BY SCALING PNSE BY 3.0 AS SOON AS IT IS CALCULATED IN JWISC
         DO 18 I=1,3
18        SNR(I)=10.0*ALOG10(PWR(I,I)/PNSE(I))
C18       SNR(I)=10.0*ALOG10(PWR(I,I)/PNSE(I))-7.27
C        IF (OFLAG .NE. 1) GOTO 990
         TYPE 899
         IF (IFL .EQ. 1) TYPE 898
49       DO 100 J=1,2
          K=(-1)**J
          TYPE 900,J
          TYPE 901,J,J-K,PRAT(J,1),J,PRAT(J,2),J,SNR(J)
100      CONTINUE
         TYPE 902
         TYPE 903,PRAT(3,1),PRAT(3,2),SNR(3)
         TYPE 904,SIR
         RI1NRM=SNR(3)-PRAT(3,1)
         RI2NRM=SNR(3)-PRAT(3,2)
         RLATT1=10.0*ALOG10( PWR(1,1)/PWR(1,3) )
         RLATT2=10.0*ALOG10( PWR(2,2)/PWR(2,3) )
         TYPE 905,RLATT1,RLATT2
         TYPE 906,RI1NRM,RI2NRM
```

```
      GOTO 990
898   FORMAT(' Before Adaptation: ',2A1)
899   FORMAT(//)
900   FORMAT('  AUXILIARY ELEMENT ',I1,' PARAMETERS: ',2A1)
901   FORMAT('   J',I1,'/J',I1,'= ',F8.4,' dB',' J',I1,'/D= ',F8.4,
     X ' dB',' J',I1,'/NOISE= ',F8.4,' dB',2A1)
902   FORMAT('  ARRAY OUTPUT(MAIN ANTENNA IF WEIGHTS=0) PARAMETERS:'
     X ,2A1)
903   FORMAT('    D/J1= ',F8.4,' dB','   D/J2= ',F8.4,' dB','   D/N=',
     X F8.4,' dB',2A1)
904   FORMAT('    D/I=D/(J1+J2)= ',F8.4,' dB',2A1)
905   FORMAT(' LEAKAGE ATTENS: AUX1-M=',F6.2,' AUX2-M=',F6.2)
906   FORMAT(' I1NR IN MAIN=',F6.2,' I2NR IN MAIN=',F6.2)
990   RETURN
      END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C     SUBPROGRAM TO APPLY I,Q WEIGHTS USING DAC-11 D/A CONVERTERS
C          PROGRAMMING BOTH MAGNITUDE AND PHASE
      SUBROUTINE JWWT
      COMMON/WTBLK/LDAO,WI,WQ,VMX,VMY,NW
      REAL WI(2),WQ(2),VMX(40,4),VMY(40,4)
      REAL A(4),XLSB,MAG,PHI
      INTEGER ID(4),NW(4),J,IADDRI,IADDRQ,LDAO

      XLSB=20./4095
      DO 150 J=1,2
       K=2*J
       L=K-1
50     TI=WCV(VMX(1,L),VMY(1,L),NW(L),ABS(WI(J)))
       TQ=WCV(VMX(1,K),VMY(1,K),NW(K),ABS(WQ(J)))
       VI=SIGN(TI,WI(J))
       VQ=SIGN(TQ,WQ(J))
C      TYPE *,WI(J),VI,WQ(J),VQ
80     IDI=INT(VI/XLSB+"4000+0.5)
        IF(J .EQ. 1) LDAO=IDI
       IDQ=INT(VQ/XLSB+"4000+0.5)
       IADDRI="170440+(J-1)*4
       IADDRQ=IADDRI+2
       CALL IPOKE(IADDRI,IDI)
       CALL IPOKE(IADDRQ,IDQ)
150    CONTINUE
999    RETURN
      END
      FUNCTION WCV(X,Y,N,Y1)
      REAL X(40),Y(40),Y1
      DO 100 J=1,N
       Q=Y(J)
       R=Y(J+1)
       IF (Q.LE.Y1 .AND. Y1.LE.R) GOTO 150
100    CONTINUE
150    SLPE=(R-Q)/(X(J+1)-X(J))
      WCV=(Y1-Q)/SLPE+X(J)
      RETURN
      END
```