

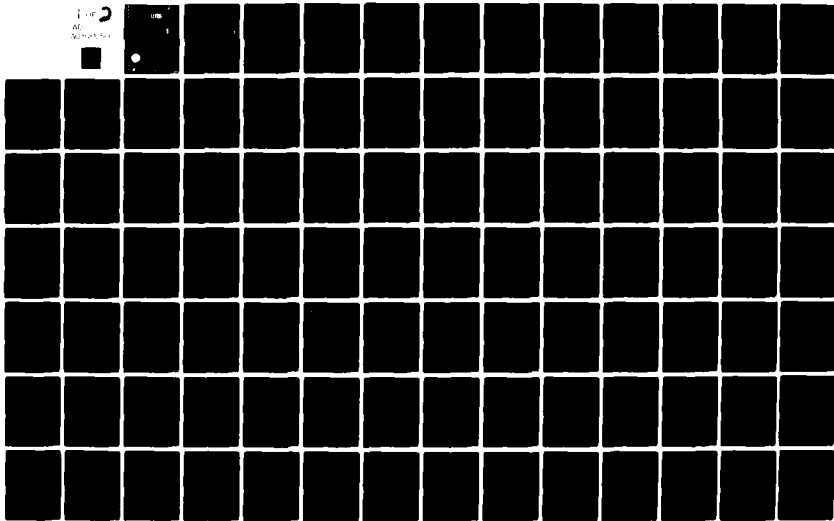
AD-A082 550 TECHNOLOGY SERVICE CORP SILVER SPRING MD

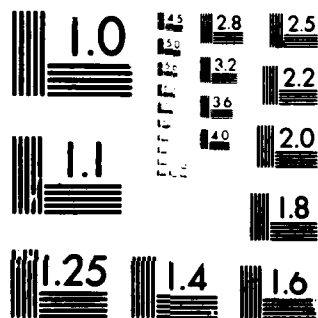
F/G 17/9

UNCLASSIFIED FEB 80 J N BUCKNAM
TSC-W7-78

N60921-79-C-A306
NL

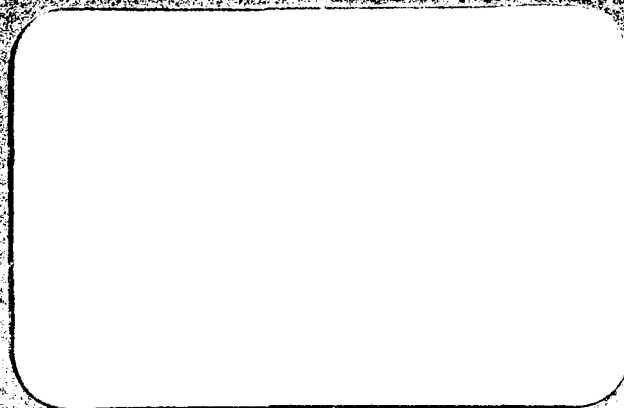
1-17-80
AD
200-100-100





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA082550



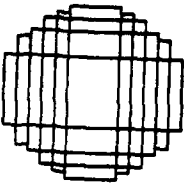
SPIC
C

THIS DOCUMENT CONTAINS
NO TECHNICAL DATA
AND IS NOT TO BE
REPRODUCED OR
TRANSMITTED IN ANY
FORM OR BY ANY
MEANS, ELECTRONIC OR
MECHANICAL, INCLUDING
PHOTOCOPYING, RECORDING,
OR BY ANY INFORMATION
STORAGE AND RETRIEVAL
SYSTEM.

FILE

Technology Service Corporation

80 3 31 040



Technology Service Corporation

Washington Operations: 8555 Sixteenth Street, Silver Spring, Maryland 20910 Phone: [301] 565-2970

Ref TSC-W7-78/rad
B6581111

⑨ Technical rept.,

⑥ DETECTION PERFORMANCE SIMULATION
OF A DICKE-TYPE CFAR
IN DISTRIBUTED CLUTTER.

⑪ 22 Feb 1980

⑫ 125

By:

⑩ J. N. Bucknam

CDRL Sequence Number A003

Contract/N60921-79-C-A306, N00024-78-C-7065

⑮

Prepared for:

Systems Compatibility Branch, Code F-42
Naval Surface Weapons Center
Dahlgren, Virginia 22448

This document has been approved
for public release and sale; its
distribution is unlimited.

390751

mtx

ACKNOWLEDGEMENTS

The work reported here was performed for the Naval Sea Systems Command's Shipboard Surveillance Radar Systems (SSURADS) program under contracts with the Command (N00024-78-C-7065) and the Naval Surface Weapons Center (N60921-79-C-A306). The Command's Program Manager is W. E. French, NAVSEA Code 62X12, and T. Henderson of NSWC/DL Code F-42 represented the Center.

This report is one of a series documenting radar modeling tools for use by Navy Laboratories.

Accession For	
NTIS	☒
DDC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

DETECTION PERFORMANCE SIMULATION
OF A DICKE-TYPE CFAR IN
DISTRIBUTED CLUTTER

by

J. N. Bucknam

ABSTRACT

↙ A Dicke-type CFAR processor is one which hard limits the radar IF or bipolar video signal prior to pulse compression. The constant false alarm rate (CFAR) properties of this processor in white receiver noise are well known. When the background against which targets must be detected is due to clutter returns, this CFAR property is destroyed.

This report summarizes results of computer simulation of a Dicke-type CFAR in a background consisting of distributed clutter returns. Clutter model parameters are chosen to approximate the statistics of sea, rain, and land clutter. The simulations show the increase in probability of false alarm (P_{FA}) in these backgrounds when the threshold settings are chosen to provide the desired P_{FA} in receiver noise. In addition, the probability of target detection as a function of signal-to-background ratio is determined. Values are determined for the coefficients in a parametric representation of these results.

A detailed documentation of the simulation software is included as an Appendix. ↗

P-1

TABLE OF CONTENTS

1.0	INTRODUCTION	1
2.0	PROCESSOR AND BACKGROUND DESCRIPTIONS	2
2.1	Waveform and Signal Processor	2
2.2	Background Models	2
3.0	SIMULATION RESULTS	8
3.1	Threshold Selections in Receiver Noise	8
3.2	Probability of False Alarm in Non-Gaussian Clutter	8
3.3	Probability of Detection, Non-Fluctuating Target	14
4.0	SIMULATION TECHNIQUES	31
4.1	Functional Description of Simulation Program	31
4.2	Simulation Simplifications	34
4.2.1	Waveform Selection	34
4.2.2	In-phase Signal Assumption	34
4.2.3	Beamshape, Range Bin, and Doppler Bin Scalping Losses Neglected	36
5.0	REFERENCES	37
Appendix A. Detailed Simulation Program Description		A-1
A.1	Module Descriptions	A-1
A.1.1	Driver (DETDFX)	A-3
A.1.2	File Open and Close (PNCLS)	A-10
A.1.3	Array Initialization (INIT)	A-16
A.1.4	Waveform Generation (PNCODE)	A-18
A.1.5	DB Conversion (DBCONV)	A-20
A.1.6	Gaussian Noise Generator (NOISE and SETUPN)	A-22
A.1.7	PIV Documentation (WRPIV)	A-26
A.1.8	Random Amplitude Generator (RANAMP)	A-28
A.1.9	Pseudo-Random Number Generator (NRAND)	A-31
A.1.10	Random Phase Generator (RANPHS)	A-33
A.1.11	Log-normal Background Generator (LGNORM and SETUPL)	A-37
A.1.12	Inverse Cumulative Gaussian Function (GAUSIV)	A-41
A.1.13	Weibull Background Generator (WEIBUL and SETUPW)	A-43
A.1.14	Gamma Function (GAMMA)	A-47
A.1.15	Correlated Background Generator (CORBKG)	A-49

Table of Contents (con't)

A.1.16	Radar IF Generator (IPGEN)	A-51
A.1.17	Target Generator (TARGET)	A-53
A.1.18	Inverse Cumulative Chi-Squared Function(CHISQ)	A-55
A.1.19	Threshold Crossing Accumulation (PDDFX)	A-58
A.1.20	Threshold Crossing Summary (SUMARY)	A-60
A.1.21	Edit, Fit, and Plot Data (PLOTDP)	A-62
A.1.22	Generate Plottable Files (PDPLLOT)	A-64
A.1.23	Parametric Curve Fit (QPDFIT)	A-66
A.1.24	SBR vs. PD (SBRVPD)	A-69
A.1.25	PD vs. SBR (PDVSBR)	A-71
A.2	<u>Example Run</u>	A-73

1.0 INTRODUCTION

A Dicke-type CFAR processor is one which hard limits the radar IF or bipolar video signal prior to pulse compression. The constant false alarm rate (CFAR) properties of this processor in white receiver noise are well known [1]. When the background against which targets must be detected is due to clutter returns, this CFAR property is destroyed.

This report summarizes results of computer simulation of a Dicke-type CFAR in a background consisting of distributed clutter returns. Clutter model parameters are chosen to approximate the statistics of sea, rain, and land clutter. The simulations show the increase in probability of false alarm (P_{FA}) in these backgrounds when the threshold settings are chosen to provide the desired P_{FA} in receiver noise. In addition, the probability of target detection as a function of signal-to-background ratio is determined. Values are determined for the coefficients in a parametric representation of these results.

2.0 PROCESSOR AND BACKGROUND DESCRIPTIONS

2.1 Waveform and Signal Processor

Figure 1 depicts the receive signal processing.

The transmitted waveform consists of a phase-coded pulse-compression waveform. It is formed by a sequence of contiguously transmitted, equal amplitude pulses (or "chips"), each of which has a phase of either 0° or 180° , determined by a maximal-length pseudo-random sequence generator. This waveform is transmitted sequentially on one or more frequency diversity channels.

On receive, the IF signal (on a particular frequency diversity channel) is first converted to I and Q bipolar video by filters matched to the individual chips. The bipolar video is hard-limited to ± 1 , and sampled at the chip rate. The hard-limited samples are then pulse compressed by cross-correlation against the stored pseudo-random sequence of ± 1 ones. The pulse-compressed I and Q are then square-law envelope detected, and added to the corresponding quantities from other diversity channels. This sum forms the test statistic for threshold comparison.

2.2 Background Models

Four types of background are of interest:

1. Receiver noise
2. Distributed Sea Clutter
3. Distributed Rain Clutter
4. Distributed Land Clutter

Receiver noise is easily (and accurately) described as Gaussian noise, which is spectrally flat (white) over the receiver bandwidth. Thus, samples of receiver noise are independent when separated in time by roughly the inverse of the receiver bandwidth.

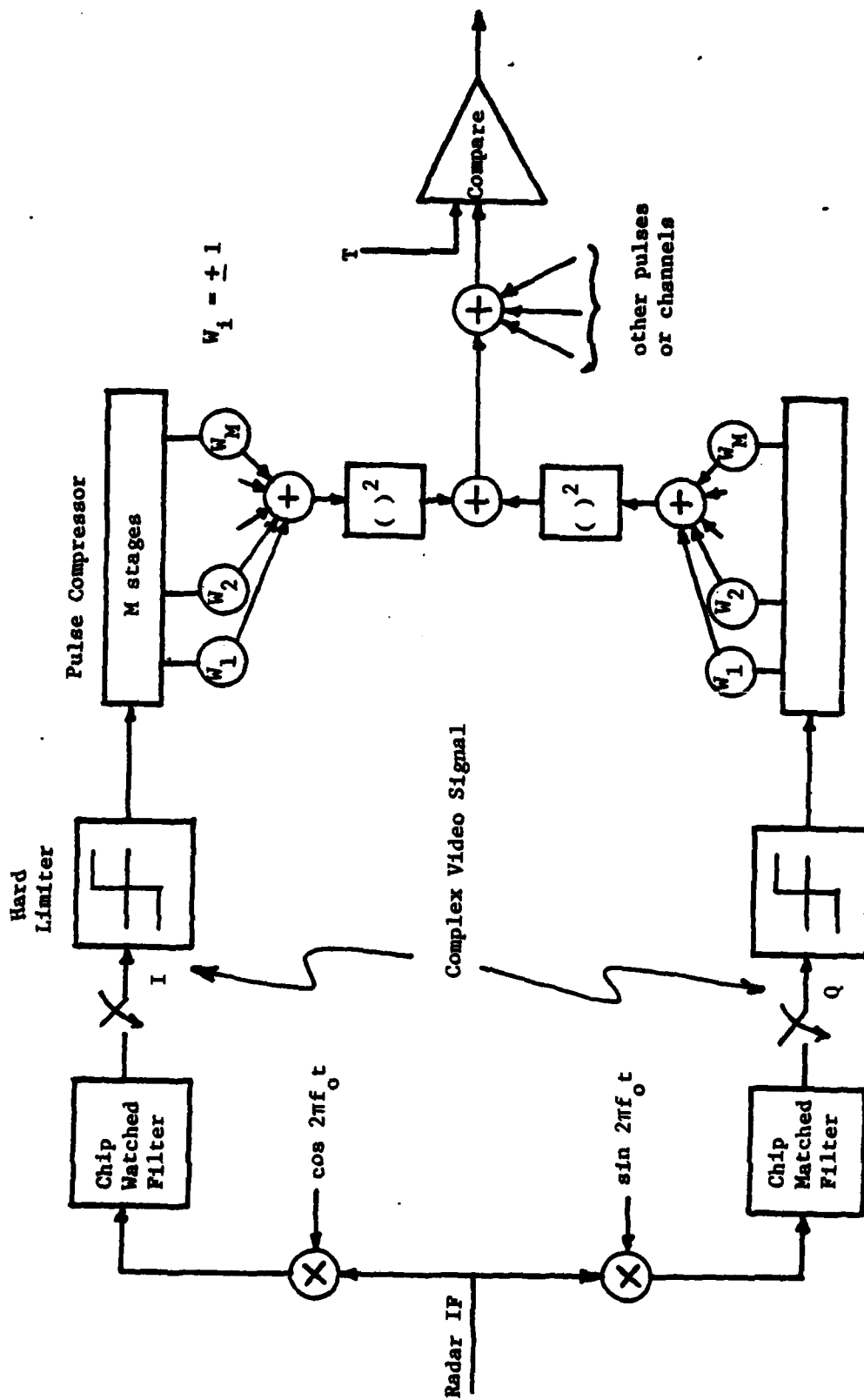


Figure 1 - Hard-Limiting Pulse Compression Receiver with Incoherent Pulse-to-Pulse Integration

The radar clutter response requires a relatively more complicated model. The clutter return from a given resolution cell is assumed to be the sum of contributions from a large number of independent "point" clutter scatterers within the cell. When the number of such scatterers is large, and when the sum is not dominated by a small subset of scatterers, samples of the radar clutter signal produced by the isolated resolution cell on successive pulses may be modeled as a Rayleigh random process, whose mean power is proportional to the number and strength of scatterers within the cell. On any given cell, however, the mean power of this Rayleigh process is apt to differ quite significantly from other cells within the local vicinity. This cell to cell variation of the local average cross section is more accurately described by the log-normal or Weibull families of distributions.

This leads to the following algorithm for simulating the clutter return from a group of range cells on a sequence of radar pulses. A random number Y is drawn for each cell from a log-normal or Weibull distribution to represent the average clutter cross-section occurring in that cell. For each cell, a sequence of clutter amplitudes is generated by sampling a narrowband complex Gaussian process whose mean power is the average cross-section from that cell, and whose sample-to-sample correlation corresponds to the desired pulse-to-pulse fluctuation spectrum of the clutter.

Mathematically, the complex clutter amplitude is obtained as the product of two random variables:

$$C_{ij} = \sqrt{Y_i} X_{ij}$$

where

C_{ij} is the complex (in-phase and quadrature) clutter amplitude on the i^{th} range bin, j^{th} pulse,

Y_i is the mean clutter cross-section on the i^{th} range cell, which follows a log-normal or Weibull distribution,

and

X_{ij} is a unit mean power, complex Gaussian random variable.

The complex clutter amplitude is a model for the complex bipolar video clutter response of a linear, simple pulse (no pulse compression) radar with pulse length equal to the chip length of the pseudo-noise code. That is, in Figure 1, C_{ij} represents the I, Q samples just prior to the hard limiters, when a unit compression ratio signal ($M=1$) is utilized:

$$\begin{aligned} I &= \text{real}(C_{ij}) \\ Q &= \text{imag}(C_{ij}) \end{aligned}$$

When a pulse compression waveform is used, the I, Q samples are obtained by convolution in the range dimension of C_{ij} with the transmit waveform samples W_l :

$$\begin{aligned} I &= \text{real} \left\{ \sum_{l=0}^{M-1} C_{i-l, j} W_l \right\} \\ Q &= \text{imag} \left\{ \sum_{l=0}^{M-1} C_{i-l, j} W_l \right\} \end{aligned}$$

Table 1 summarizes the specific distributions used for simulating the cell-to-cell variations (Y_i) in mean clutter cross-section, for each of three types of clutter. For sea and rain clutter, Y_i is drawn from a log-normal distribution. That is,

$$Y = e^z$$

where z is normally distributed. The distribution for Y can be written

$$P_Y(Y) = \frac{1}{SY\sqrt{2\pi}} \exp \left\{ -\frac{\ln^2(Y/Y_{50})}{2S^2} \right\}$$

where

Y_{50} is the median value of Y

and S is the standard deviation of $\ln(Y/Y_{50})$.

The average ($\bar{Y}$) and median (Y_{50}) values of Y are related by

$$\bar{Y} = Y_{50} \exp\{S^2/2\}$$

For purposes of simulation, the average value ($\bar{Y}$) of the mean clutter cross-section (Y) is constrained to unity, i.e., unit average mean clutter cross-section. The remaining parameter in the distribution is the log-standard deviation S . Expressed in dB units, this is

$$\begin{aligned} \text{SDB} &= \sqrt{E[10 \log_{10}(Y/Y_{50})]^2} \\ &= 4.34 S \end{aligned}$$

For sea clutter, SDB is 6 dB, and for rain clutter, 3 dB. The mean to median ratios are thus

$$\begin{aligned} \bar{Y}/Y_{50} &= \exp \left\{ 1/2 \left(\frac{\text{SDB}}{4.34} \right)^2 \right\} \\ &= \begin{cases} 2.60 & , \quad \text{sea clutter} \\ 1.27 & , \quad \text{rain clutter} \end{cases} \end{aligned}$$

For land clutter, Y_1 is drawn from a Weibull distribution. A unit mean Weibull probability density is

$$P_Y(Y) = C Y^{C-1} \left[\Gamma(1 + \frac{1}{C}) \right]^C \exp \{ -[Y \Gamma(1 + \frac{1}{C})]^C \}$$

The mean-to-median ratio for this distribution is

$$\bar{Y}/Y_{50} = \frac{\Gamma(1 + \frac{1}{C})}{(\ln 2)^{1/C}}$$

The value of C is 0.3, for which

$$\bar{Y}/Y_{50} = \frac{9.26}{.295} = 31.4$$

Table 1
Mean Clutter Cross-Section
Distribution Models

Clutter Type	Mean Cross-Section Distribution Type	Parameters
Sea	Log-normal	$\bar{Y} = 1$ SDB = 6 dB
Rain	Log-normal	$\bar{Y} = 1$ SDB = 3 dB
Land	Weibul	$\bar{Y} = 1$ C = 0.3

3.0 SIMULATION RESULTS

Detection performance of the processor described in section 2.1 is simulated in this section. The pulse compression ratio in all cases is 127:1. Results are presented for 1, 2, and 4 diversity channels. A non-fluctuating (Swerling 0) target is assumed.

The simulations entail three basic steps. First, the detection thresholds are selected to provide the desired P_{FA} in receiver noise. Using these thresholds, the actual P_{FA} 's are then obtained in each of the three types of clutter. Finally, the detection probability is determined as a function of signal-to-background ratio, in each clutter type. This functional dependence is then summarized by a four parameter smooth curve which is fit to the simulation results using the technique discussed in [2].

3.1 Threshold Selections in Receiver Noise

Figure 2 summarizes the false alarm behavior of the processor in white, Gaussian receiver noise, as determined by Monte Carlo simulation. The figure is based on 10^5 trials. It was found empirically that a nearly linear relationship existed between the threshold level in dB ($10 \log_{10} T$, where T is the threshold value in Figure 1) and $\log_{10} (-\log_{10} (P_{FA}))$, over a range of P_{FA} from .63 to $\approx 10^{-4}$. This law was assumed to continue for P_{FA} below 10^{-4} , so that the threshold settings in Table 2 could be extrapolated from the simulation data.

3.2 Probability of False Alarm in Non-Gaussian Clutter

When the threshold settings for receiver noise are used in other background such as distributed clutter, the probabilities of false alarm increase by orders of magnitude. Figures 3 to 5 summarize the false alarm performance of the processor for the distributed sea, rain, and land clutter models of section 2.2. The figures are based on Monte Carlo simulation with 10^5 trials.

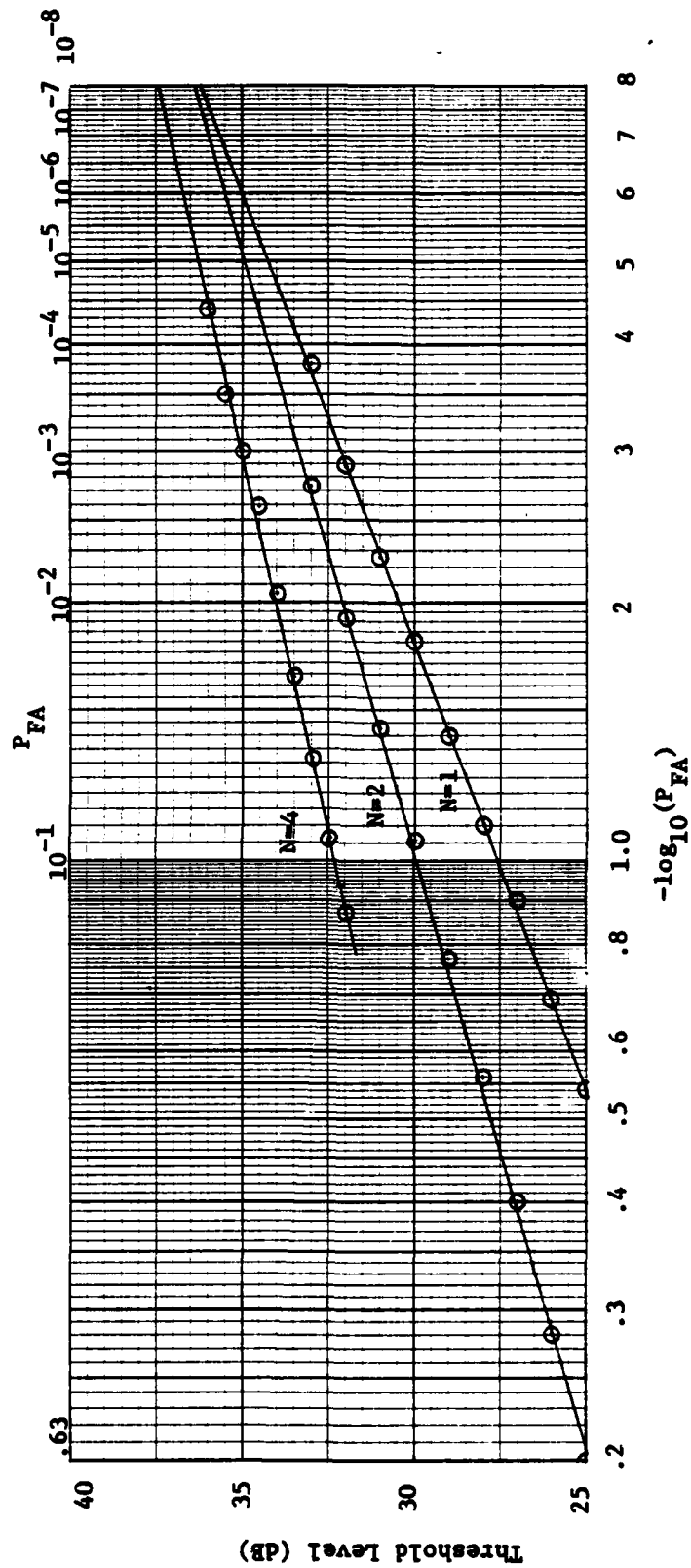


Figure 2 - False Alarm Performance in Receiver Noise; $M = 127$

Table 2 - Threshold Settings in Receiver Noise

<u>P_{FA}</u>	<u>N = 1</u>	<u>N = 2</u>	<u>N = 4</u>
10^{-3}	32.1 dB*	33.3	35.0
10^{-4}	33.3	34.2	35.7
10^{-5}	34.2	34.9	36.2
10^{-6}	35.0	35.4	36.7
10^{-7}	35.6	35.9	37.0
10^{-8}	36.2	36.3	37.4

*tabulated quantity is $10 \log_{10} T$, where T is the threshold setting.

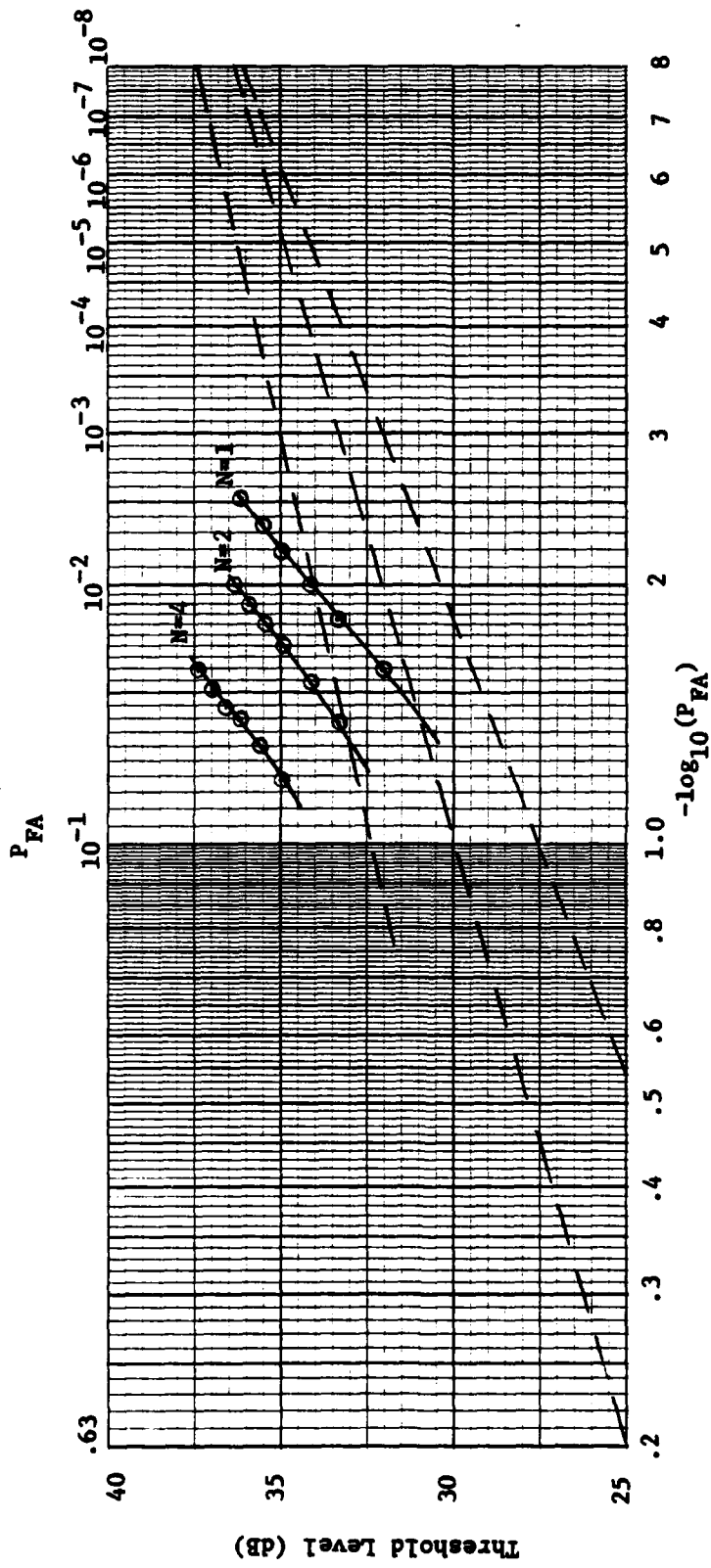


Figure 3 - False Alarm Performance in Sea Clutter; $M = 127$

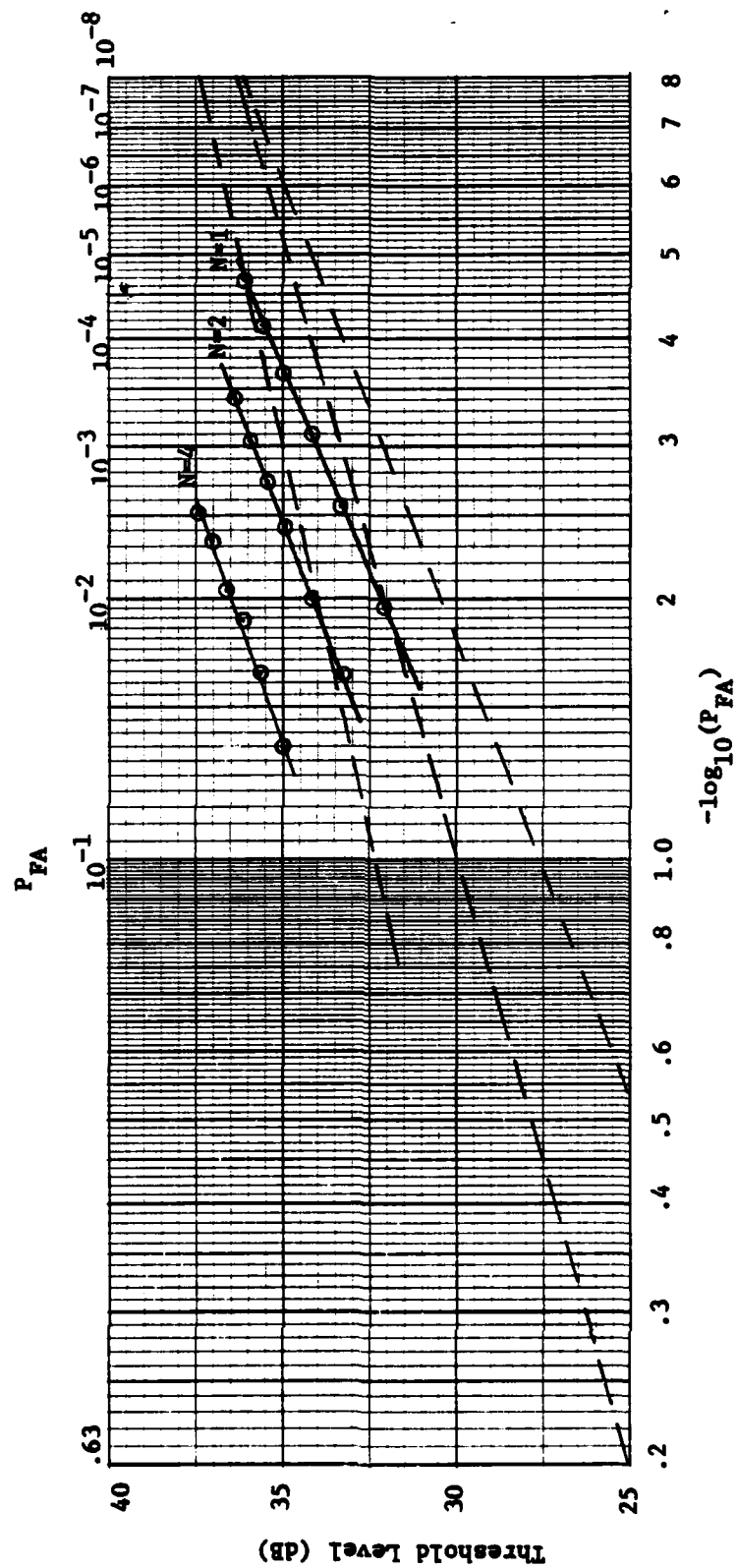


Figure 4 - False Alarm Performance in Rain Clutter; $M = 127$

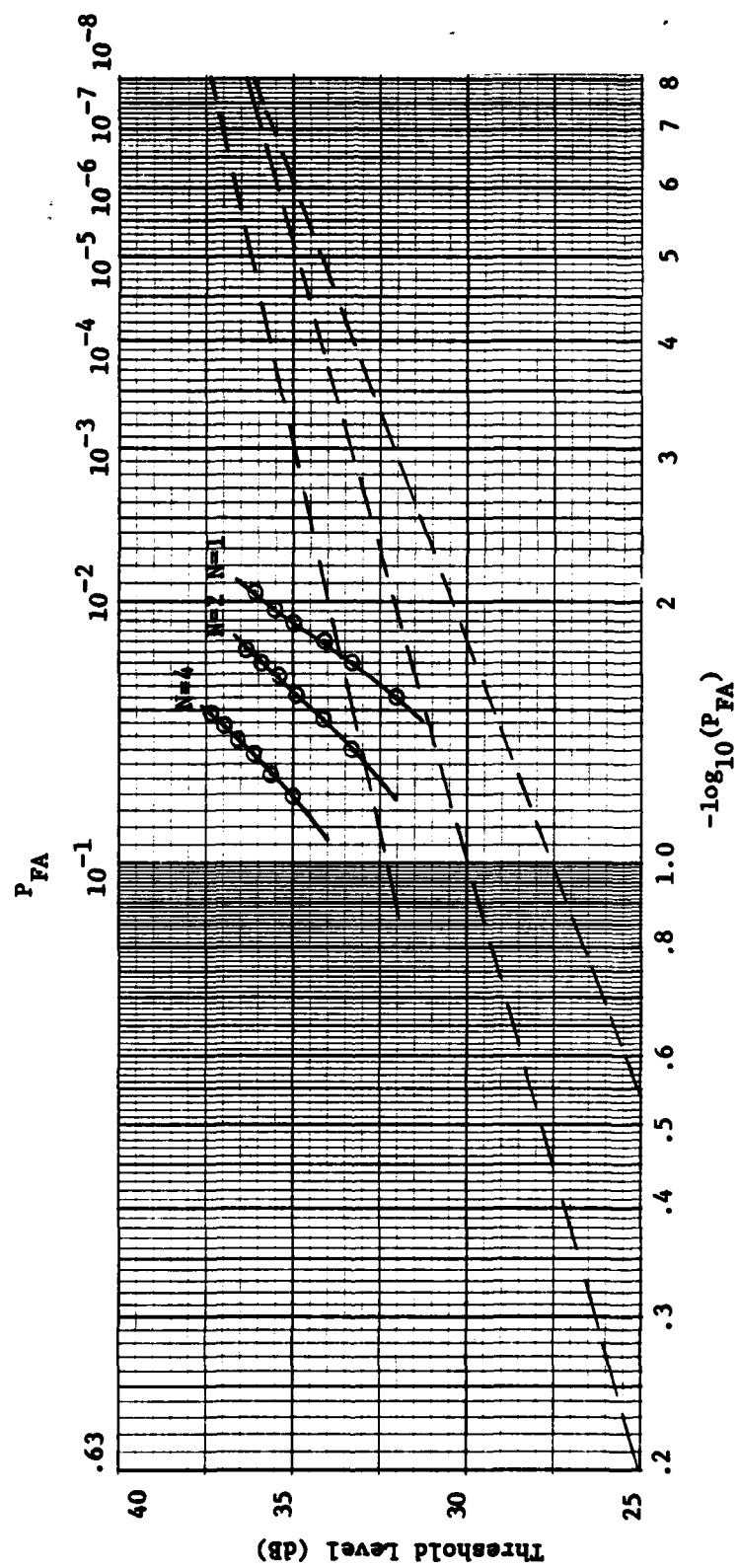


Figure 5 - False Alarm Performance in Land Clutter; $M = 127$

3.3 Probability of Detection, Non-Fluctuating Target

Detection performance for each background types was simulated using a non-fluctuating (Swerling 0) target model, and 200 trials. The threshold settings from Table 2 were used. The results are plotted in Figures 6 to 17, as curves of probability of detection (P_D) vs. signal-to-background ratio (SBR). When the background is receiver noise, signal-to-background ratio is defined as signal energy to noise spectral density ratio. When the background is distributed clutter, signal-to-background is defined as the ratio of average target cross-section to average mean clutter cross-section per range bin. In each figure, the curves are keyed to the false alarm probability obtained for each threshold in the particular background type.

The smooth curves connecting the simulation data points were obtained by application of the parametric curve fitting techniques of [2]. The coefficients for all cases in Figures 6 to 17 are summarized in Table 3.

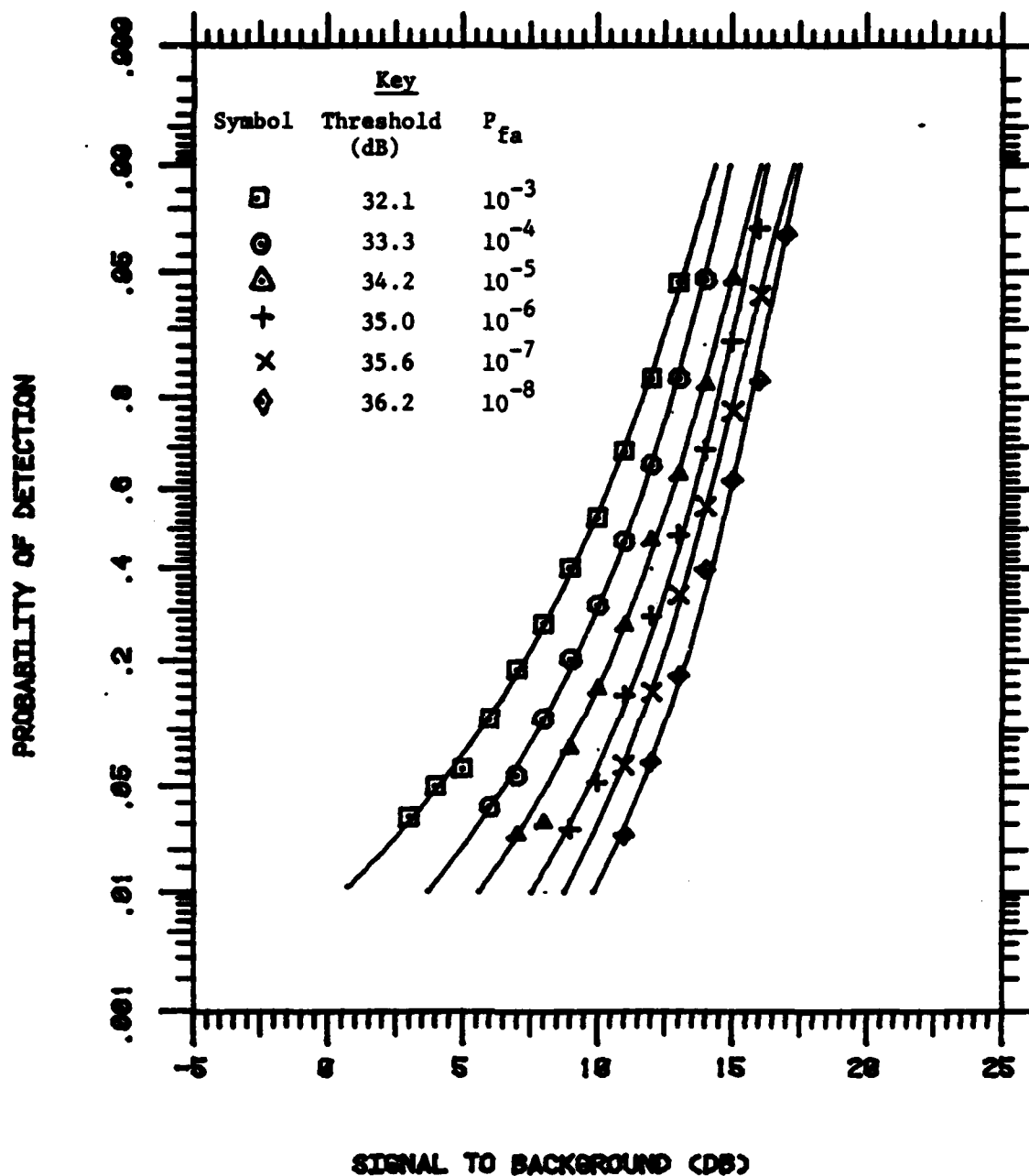


Figure 6

PERFORMANCE IN RECEIVER NOISE, $M=127$; $N=1$

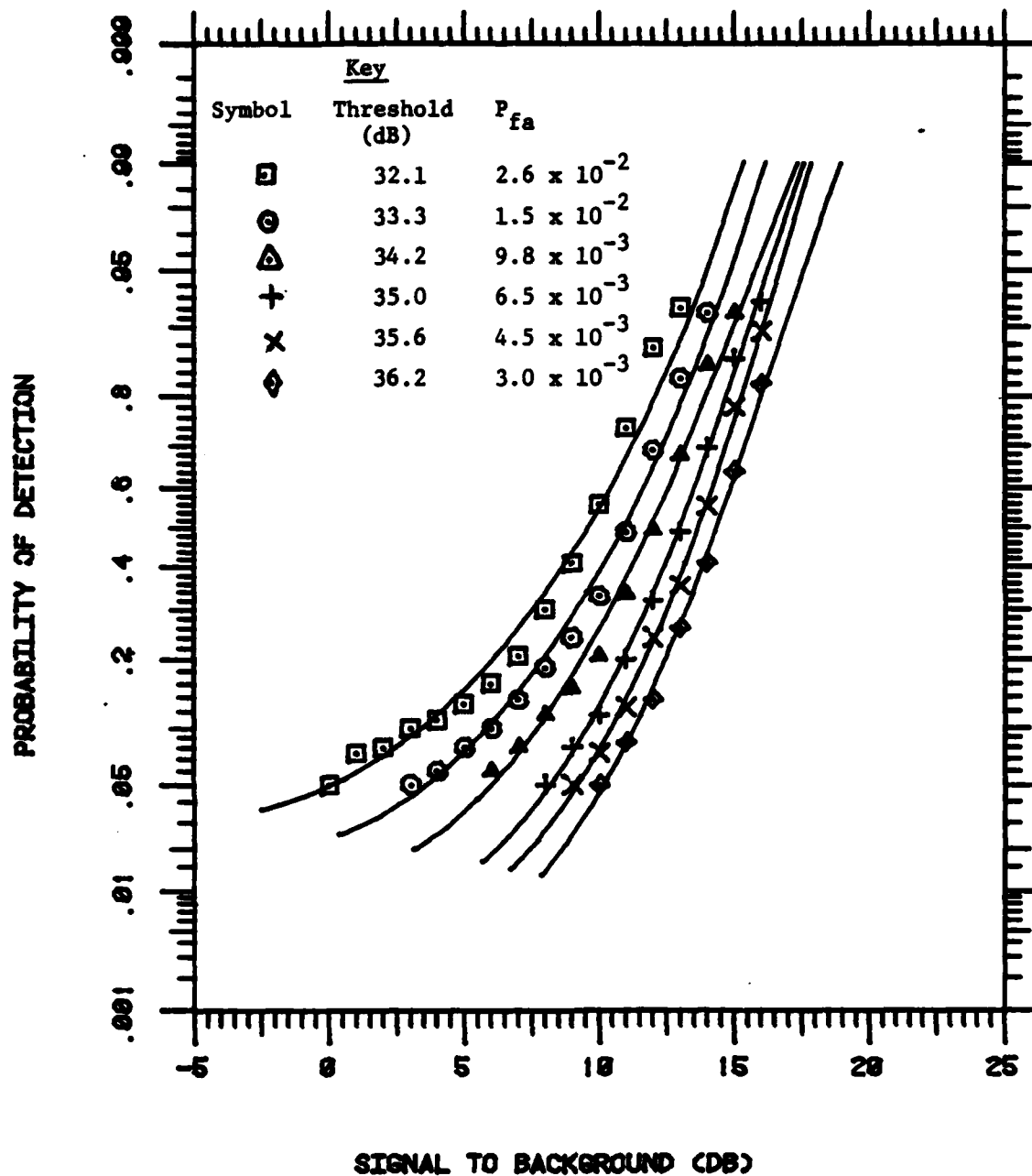


Figure 7

PERFORMANCE IN SEA CLUTTER, $M=127$; $N=1$

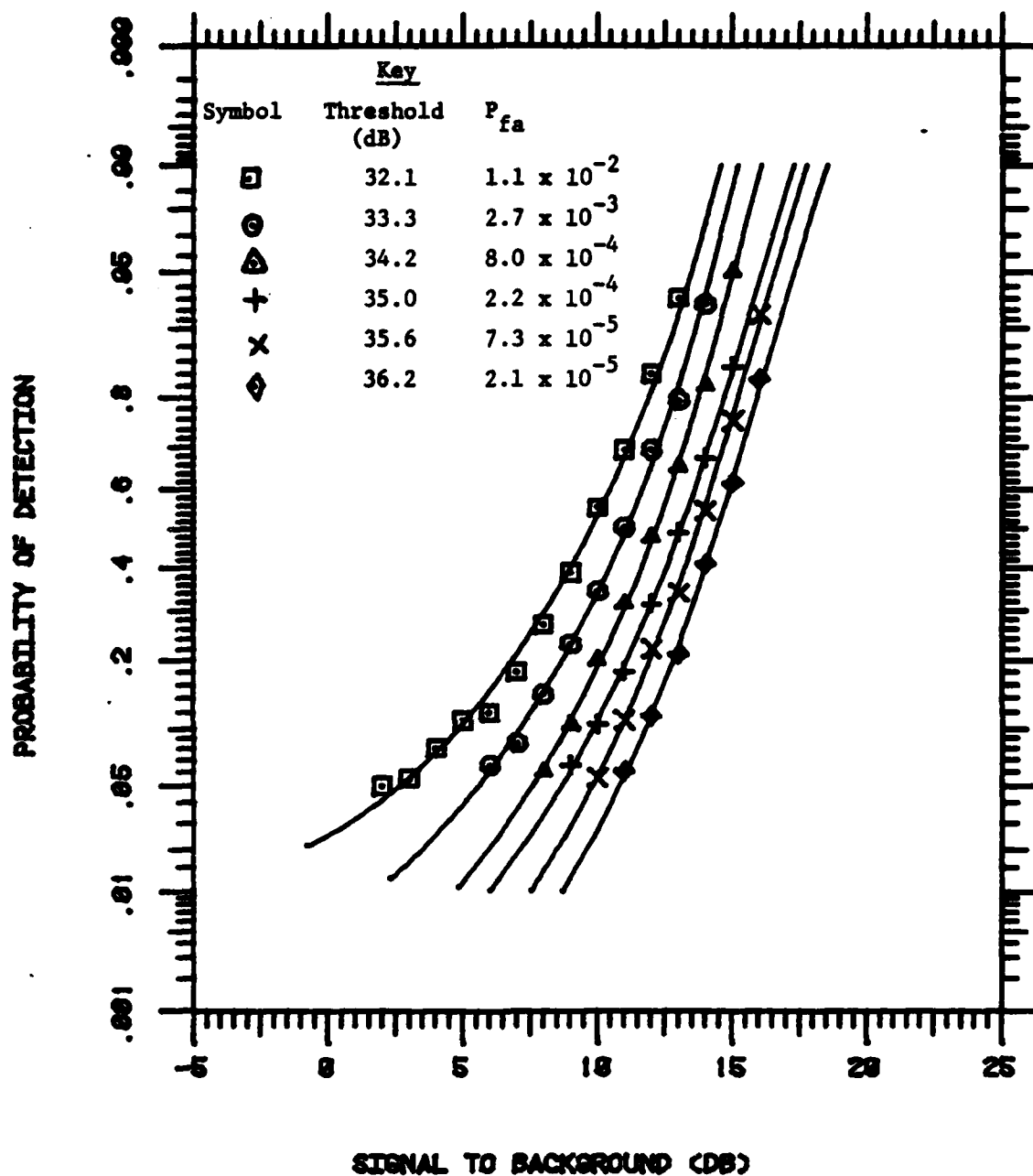


Figure 8

PERFORMANCE IN RAIN CLUTTER; $M=127$; $N=1$

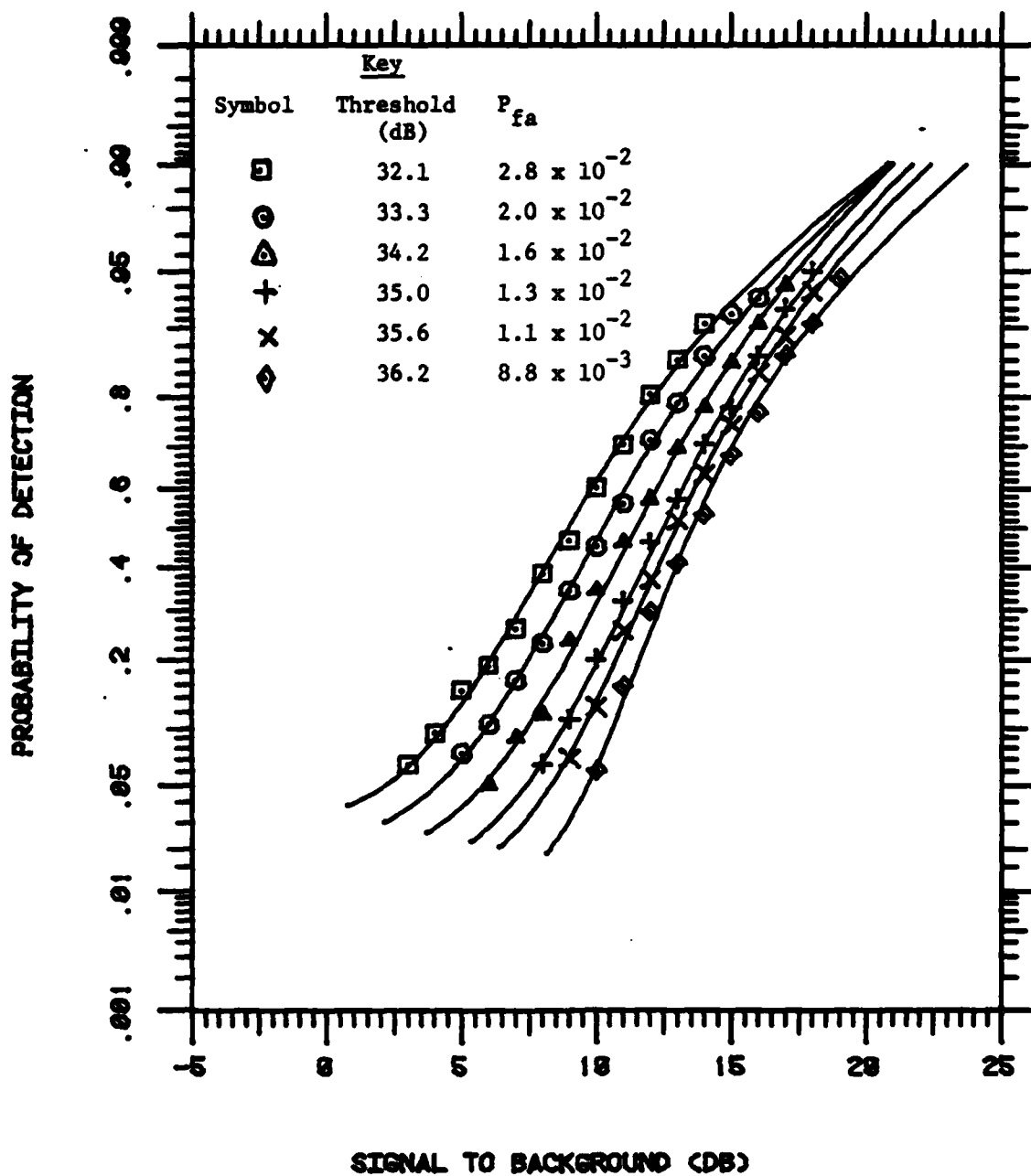


Figure 9

PERFORMANCE IN LAND CLUTTER, $M=127$, $N=1$

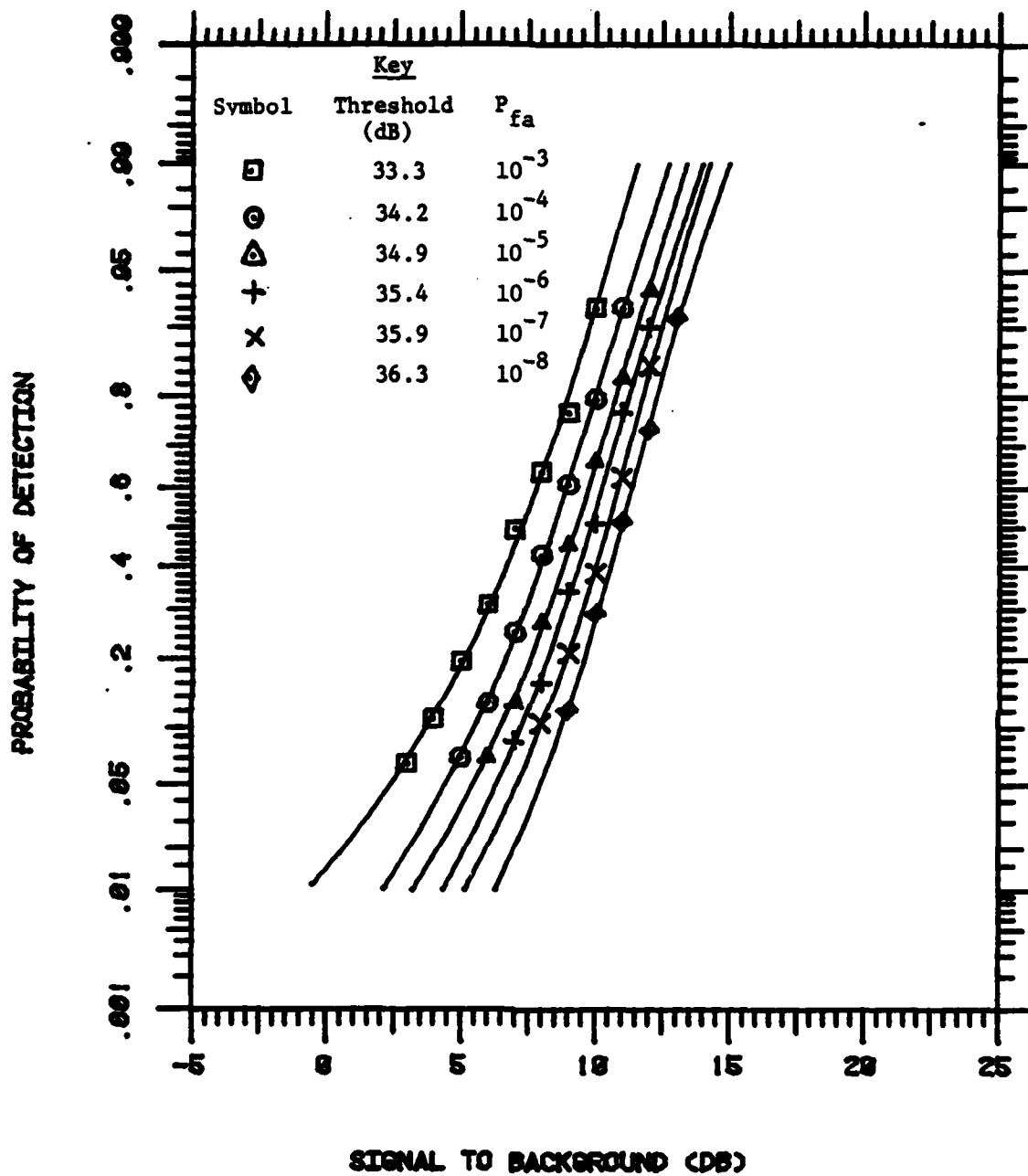


Figure 10

PERFORMANCE IN RECEIVER NOISE, $M=127$, $N=2$

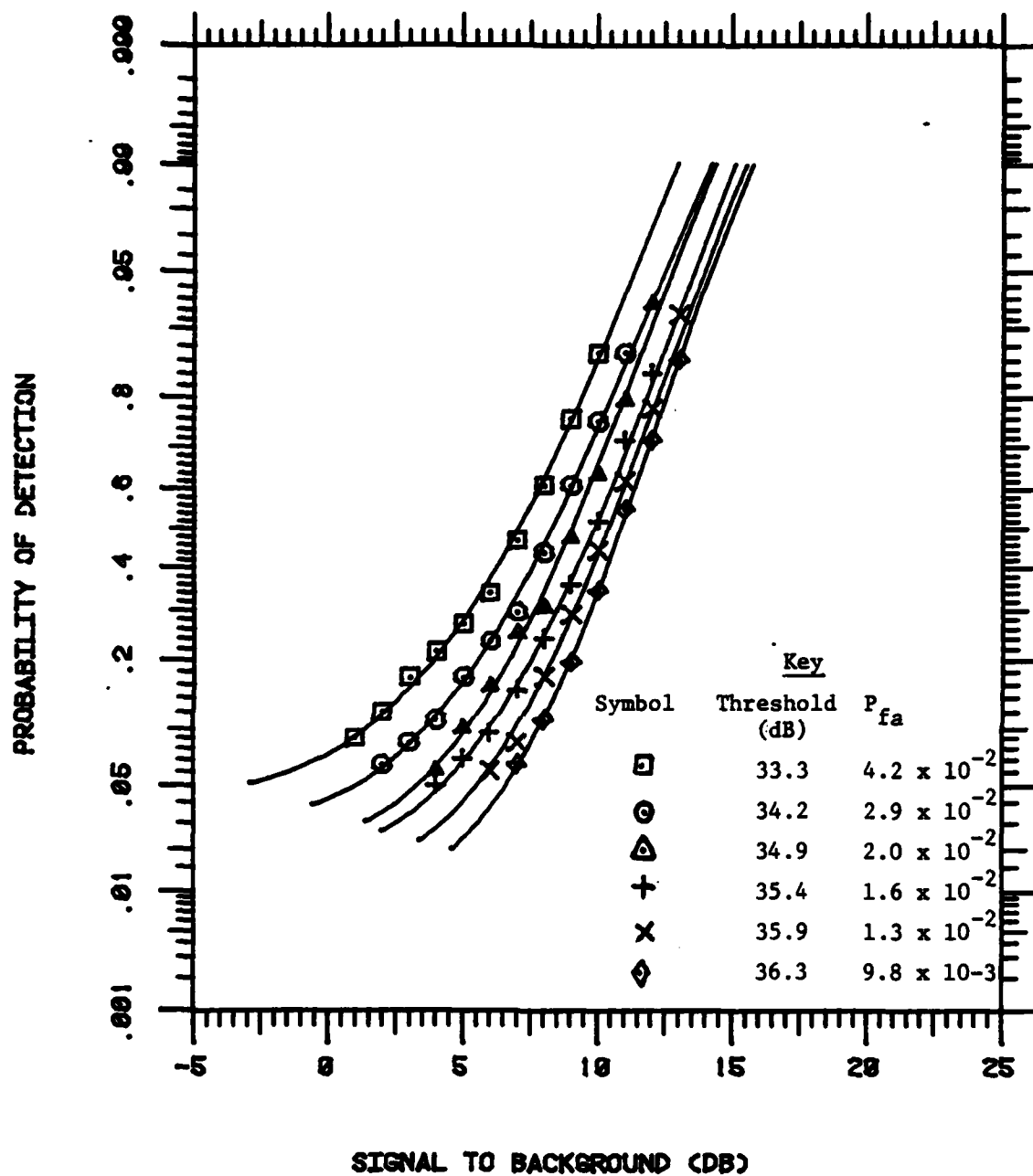


Figure 11

PERFORMANCE IN SEA CLUTTER; $M=127$; $N=2$

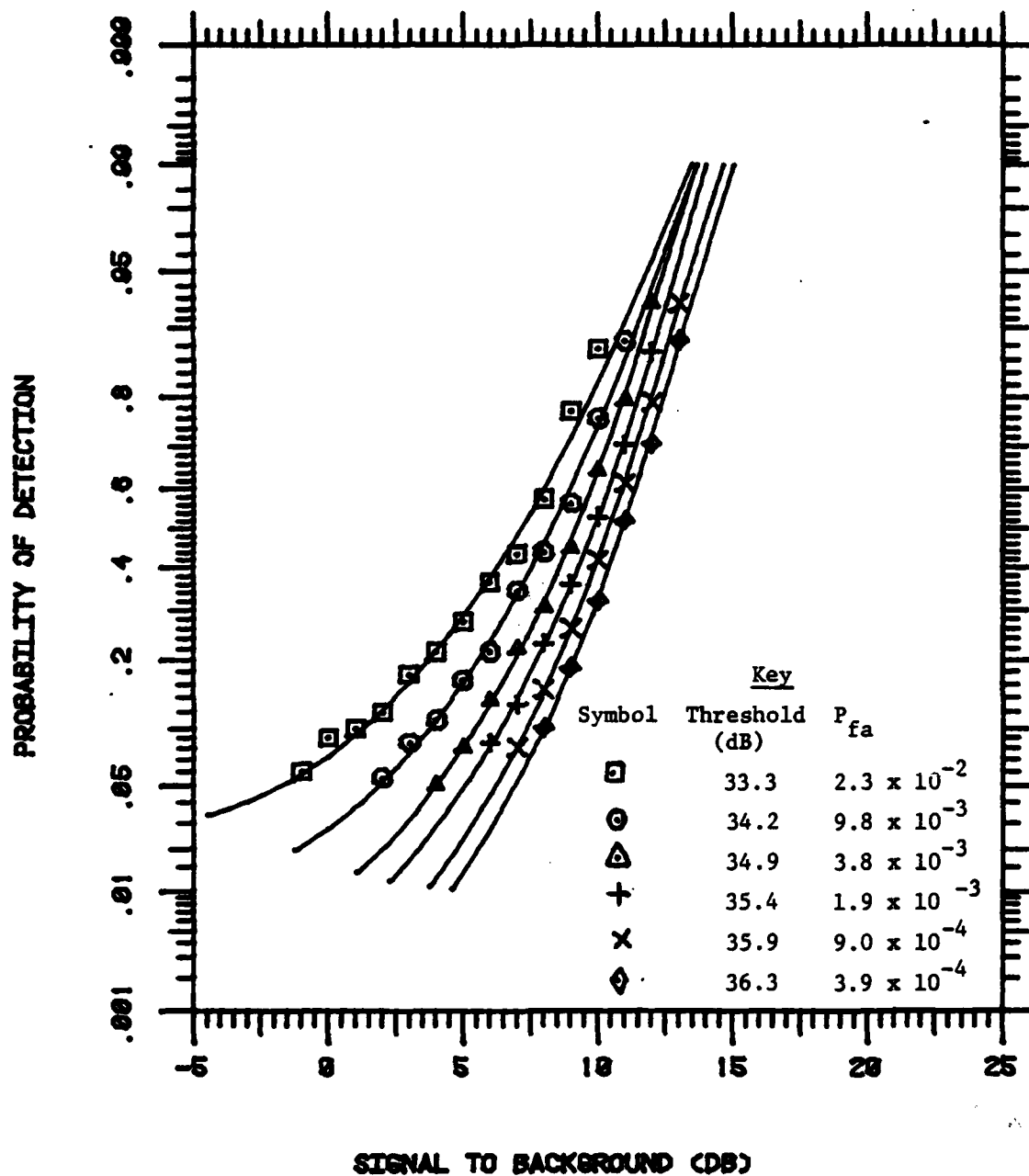


Figure 12

PERFORMANCE IN RAIN CLUTTER, $M=127$, $N=2$

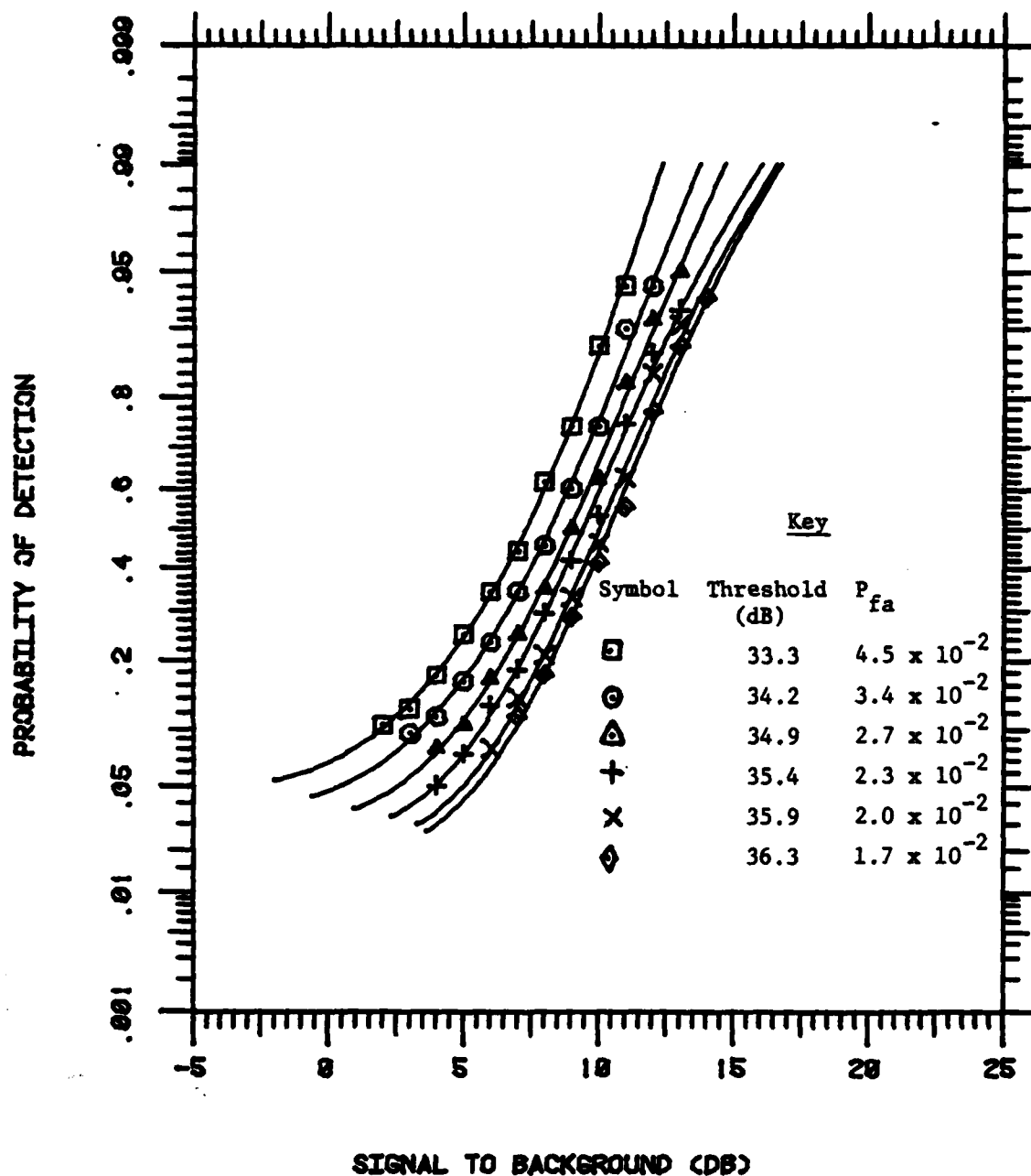


Figure 13
PERFORMANCE IN LAND CLUTTER, $M=127$; $N=2$

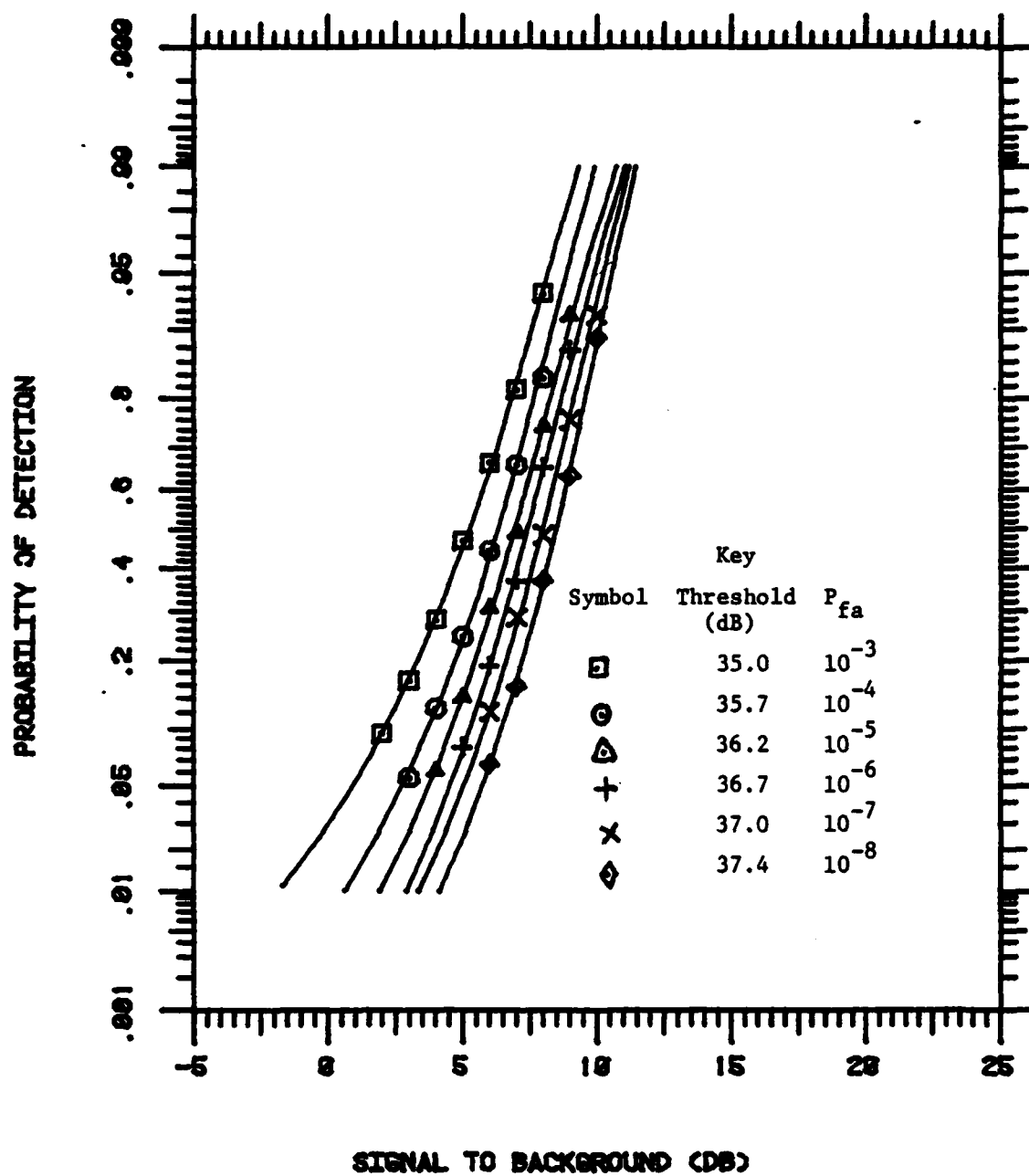


Figure 14

PERFORMANCE IN RECEIVER NOISE, $M=127$, $N=4$

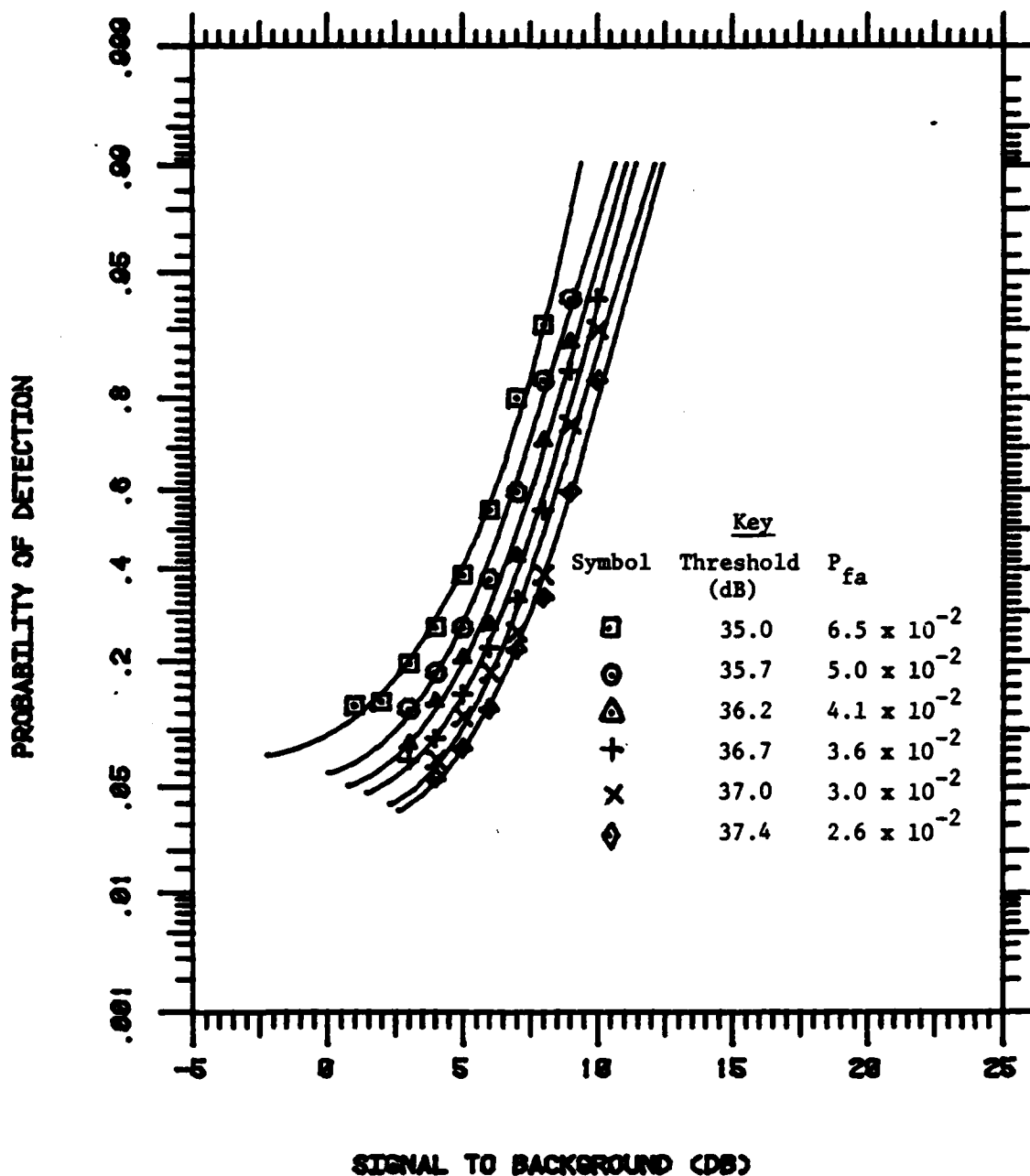


Figure 15

PERFORMANCE IN SEA CLUTTER, $M=127$, $N=4$

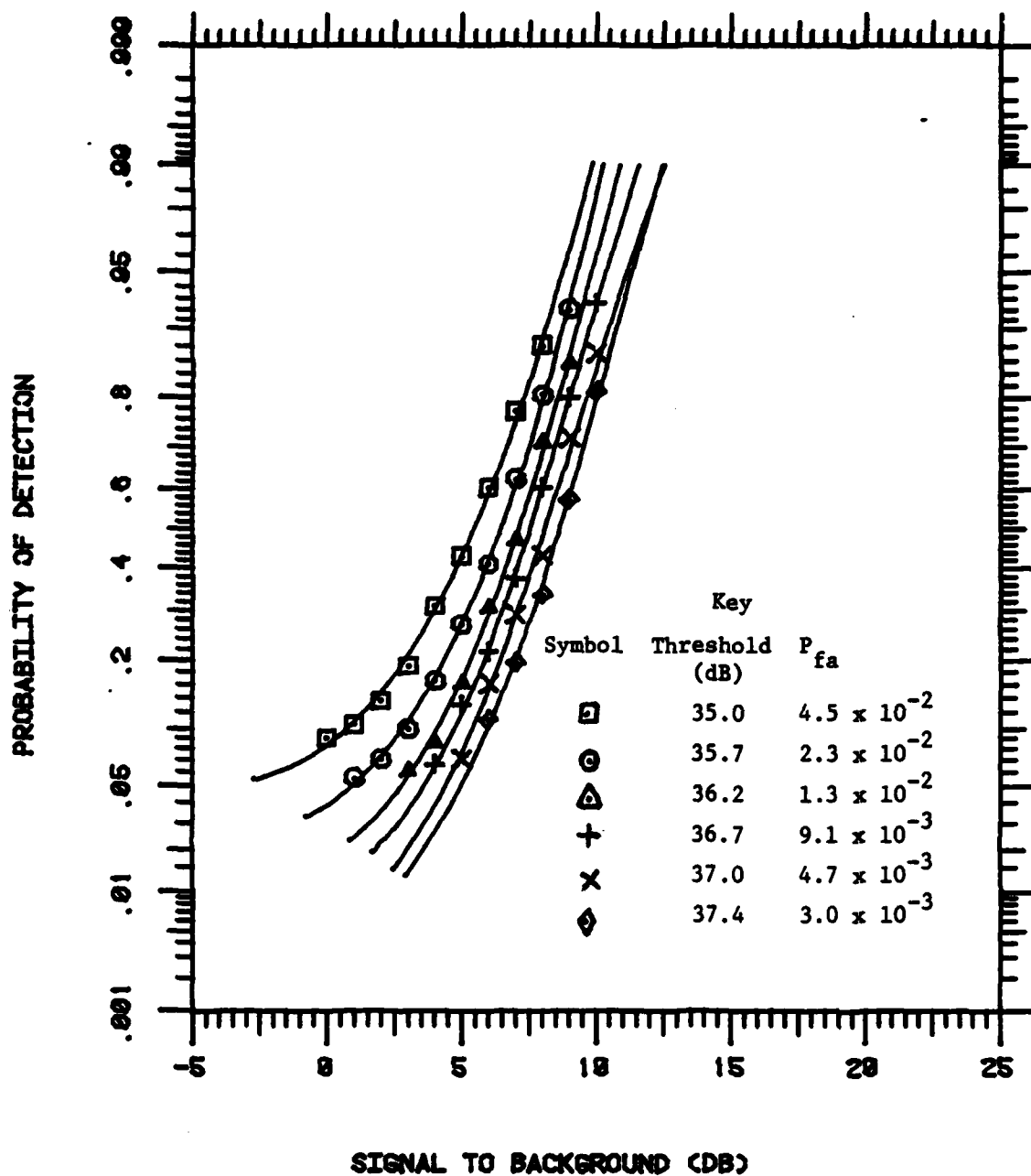


Figure 16

PERFORMANCE IN RAIN CLUTTER, $M=127$, $N=4$

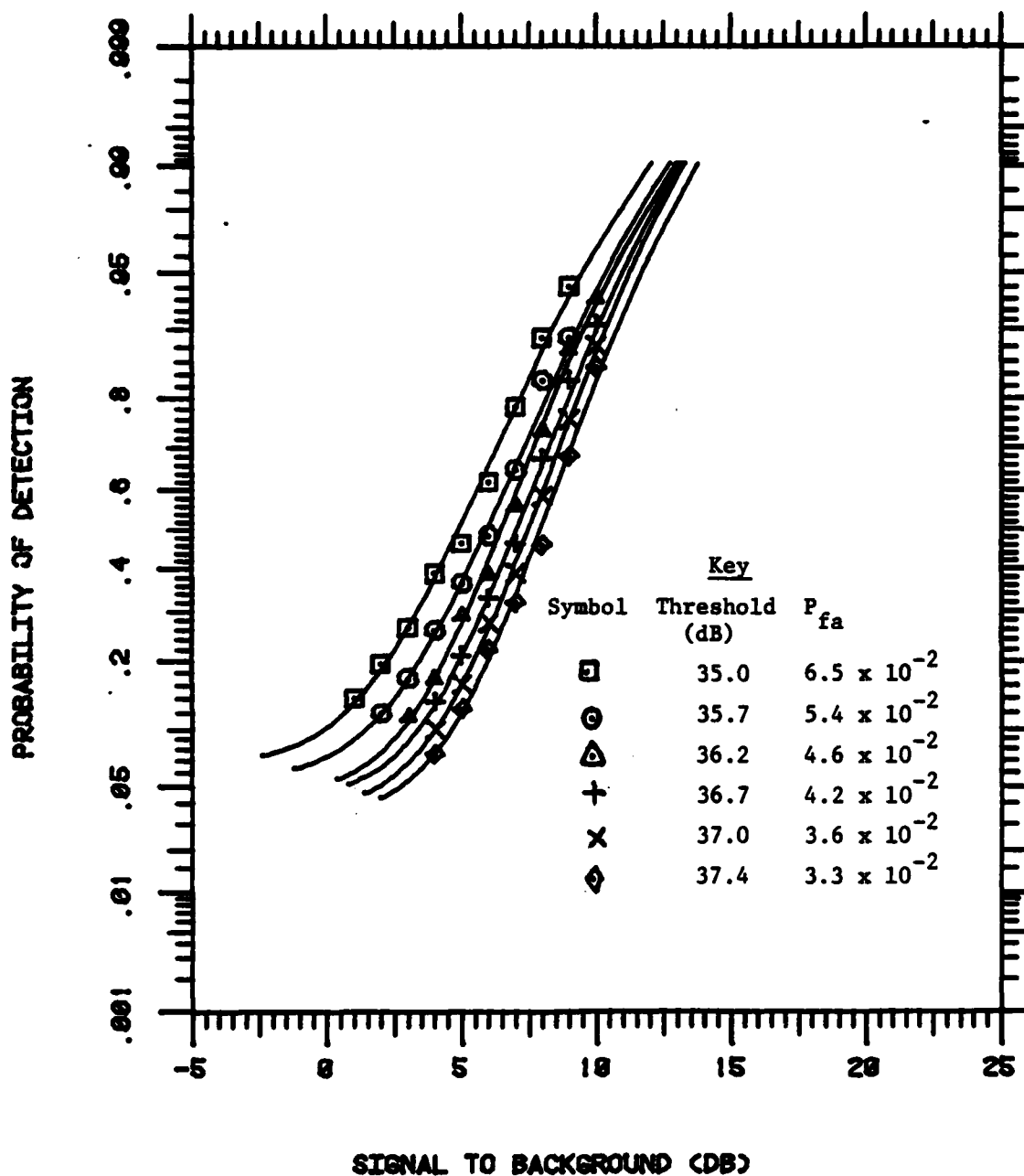


Figure 17

PERFORMANCE IN LAND CLUTTER, $M=127$, $N=4$

Table 3 - Coefficients in Parametric P_D vs. SBR Curve

$$SBR = \frac{1}{A} \left[\left(\frac{1 - P_{FA}}{1 - P_D} \right)^{1/C} - 1 \right]^{1/B}$$

(a) Receiver Noise

N	P_{FA}	A	B	C
1	10^{-3}	3.49×10^{-2}	2.05	7.09
	10^{-4}	2.82×10^{-2}	2.47	8.43
	10^{-5}	3.10×10^{-2}	2.78	4.41
	10^{-6}	2.70×10^{-2}	3.29	4.80
	10^{-7}	3.25×10^{-2}	3.78	2.09
	10^{-8}	2.47×10^{-2}	3.96	2.98
2	10^{-3}	8.93×10^{-2}	2.41	4.48
	10^{-4}	1.01×10^{-1}	2.99	2.27
	10^{-5}	8.46×10^{-2}	3.12	2.26
	10^{-6}	8.57×10^{-2}	3.50	1.69
	10^{-7}	7.22×10^{-2}	3.62	1.89
	10^{-8}	7.42×10^{-2}	4.21	1.29
4	10^{-3}	1.72×10^{-1}	2.71	3.40
	10^{-4}	1.60×10^{-1}	3.32	2.74
	10^{-5}	1.69×10^{-1}	3.83	1.69
	10^{-6}	1.63×10^{-1}	4.31	1.45
	10^{-7}	1.30×10^{-1}	4.16	1.99
	10^{-8}	1.14×10^{-1}	4.37	2.14

Table 3 - (continued)

(b) Sea Clutter

N	P_{FA}	A	B	C
1	2.63×10^{-2}	9.86×10^{-3}	1.51	25.85
	1.51×10^{-2}	1.69×10^{-2}	1.74	10.8
	9.8×10^{-3}	3.45×10^{-2}	2.14	2.95
	6.5×10^{-3}	3.11×10^{-2}	2.58	2.72
	4.5×10^{-3}	2.51×10^{-2}	2.70	3.26
	3.0×10^{-3}	2.74×10^{-2}	2.94	1.96
2	4.17×10^{-2}	8.15×10^{-2}	1.86	3.71
	2.88×10^{-2}	8.60×10^{-2}	2.12	2.43
	2.04×10^{-2}	8.36×10^{-2}	2.47	2.13
	1.62×10^{-2}	6.43×10^{-2}	2.39	2.42
	1.27×10^{-2}	6.86×10^{-2}	2.73	1.82
	9.8×10^{-3}	6.71×10^{-2}	3.07	1.59
4	6.46×10^{-2}	1.51×10^{-2}	2.30	492
	5.01×10^{-2}	1.52×10^{-1}	2.94	2.48
	4.07×10^{-2}	1.23×10^{-1}	2.95	2.93
	3.63×10^{-2}	1.09×10^{-1}	3.03	3.03
	2.95×10^{-2}	1.17×10^{-1}	3.29	2.06
	2.63×10^{-2}	9.23×10^{-2}	3.15	2.71

Table 3 - (continued)

(c) Rain Clutter

N	P_{FA}	A	B	C
1	1.14×10^{-2}	4.97×10^{-3}	1.73	136
	2.7×10^{-3}	1.50×10^{-2}	2.10	22.3
	8.0×10^{-3}	2.74×10^{-2}	2.54	5.60
	2.2×10^{-3}	2.87×10^{-2}	2.66	3.26
	7.3×10^{-4}	2.94×10^{-2}	3.06	2.45
	2.1×10^{-4}	2.79×10^{-2}	3.33	1.95
2	2.34×10^{-2}	5.35×10^{-2}	1.59	5.46
	9.8×10^{-3}	6.17×10^{-2}	1.96	4.14
	3.8×10^{-3}	5.43×10^{-2}	2.29	4.64
	1.9×10^{-3}	5.0×10^{-2}	2.46	4.50
	9.0×10^{-4}	6.01×10^{-2}	2.84	2.58
	3.9×10^{-4}	5.57×10^{-2}	2.99	2.44
2	4.47×10^{-2}	1.02×10^{-1}	2.24	6.69
	2.29×10^{-2}	9.10×10^{-2}	2.53	7.01
	1.32×10^{-2}	1.18×10^{-1}	2.99	3.33
	9.10×10^{-3}	1.30×10^{-1}	3.23	2.13
	4.70×10^{-3}	1.31×10^{-1}	3.41	1.55
	3.00×10^{-3}	9.94×10^{-2}	3.30	2.30

Table 3 - (continued)

(d) Land Clutter

N	P_{FA}	A	B	C
1	2.82×10^{-2}	1.71×10^{-1}	2.55	.596
	2.00×10^{-2}	1.16×10^{-1}	2.52	.688
	1.58×10^{-2}	8.29×10^{-2}	2.62	.749
	1.26×10^{-2}	6.88×10^{-2}	2.91	.683
	1.05×10^{-2}	6.44×10^{-2}	3.20	.598
	8.80×10^{-3}	6.31×10^{-2}	4.17	.410
2	4.47×10^{-2}	5.95×10^{-2}	1.97	6.46
	3.39×10^{-2}	7.75×10^{-2}	2.12	3.03
	2.69×10^{-2}	8.48×10^{-2}	2.36	2.05
	2.29×10^{-2}	1.05×10^{-1}	2.75	1.17
	1.96×10^{-2}	9.48×10^{-2}	2.91	1.08
	1.67×10^{-2}	8.6×10^{-2}	2.91	1.13
4	6.46×10^{-2}	2.94×10^{-1}	2.65	1.11
	5.37×10^{-2}	2.25×10^{-1}	2.69	1.18
	4.57×10^{-2}	2.24×10^{-1}	3.22	9.60×10^{-1}
	4.17×10^{-2}	1.85×10^{-1}	3.11	1.10
	3.63×10^{-2}	1.65×10^{-1}	3.17	1.15
	3.31×10^{-2}	1.51×10^{-1}	3.26	1.11

4.0 SIMULATION TECHNIQUES

The simulations were implemented in FORTRAN IV on a Prime 350 computer. This section summarizes the functional organization of the simulation program, and discusses a few simplifications utilized in the modeling. A more detailed description of the program modules is included as Appendix A.

4.1 Functional Description of Simulation Program

Figure 18 shows the functional organization of the simulation program, at the top level of module hierarchy. The first major operation of the simulation is the generation of a numerical representation of the transmit waveform. This is done with a call to the subroutine PNCODE, which returns a sequence of plus and minus ones generated from a maximal length shift register. This sequence represents the sequence of amplitudes (or equivalently phases) transmitted on each chip in the phase coded waveform.

The second step is the generation of the background signal for an ideal, impulse radar. There are four generic types of background which can be simulated. The first is white, Gaussian receiver noise, which is produced by a call to subroutine NOISE. The second and third are generalizations of NOISE, wherein the amplitude statistics of the receiver noise are changed from Rayleigh to either log-normal (LGNORM) or Weibull (WEIBUL). In all cases, samples of backgrounds produced by these three generators are independent from sample to sample. The fourth type of background generator, CORBKG, is an implementation of the clutter model of section 2.2. As such, it generates random samples of average clutter cross-section with either log-normal or Weibull statistics, and then applies uncorrelated Rayleigh amplitude and uniform phase modulations to each range bin from pulse-to-pulse.

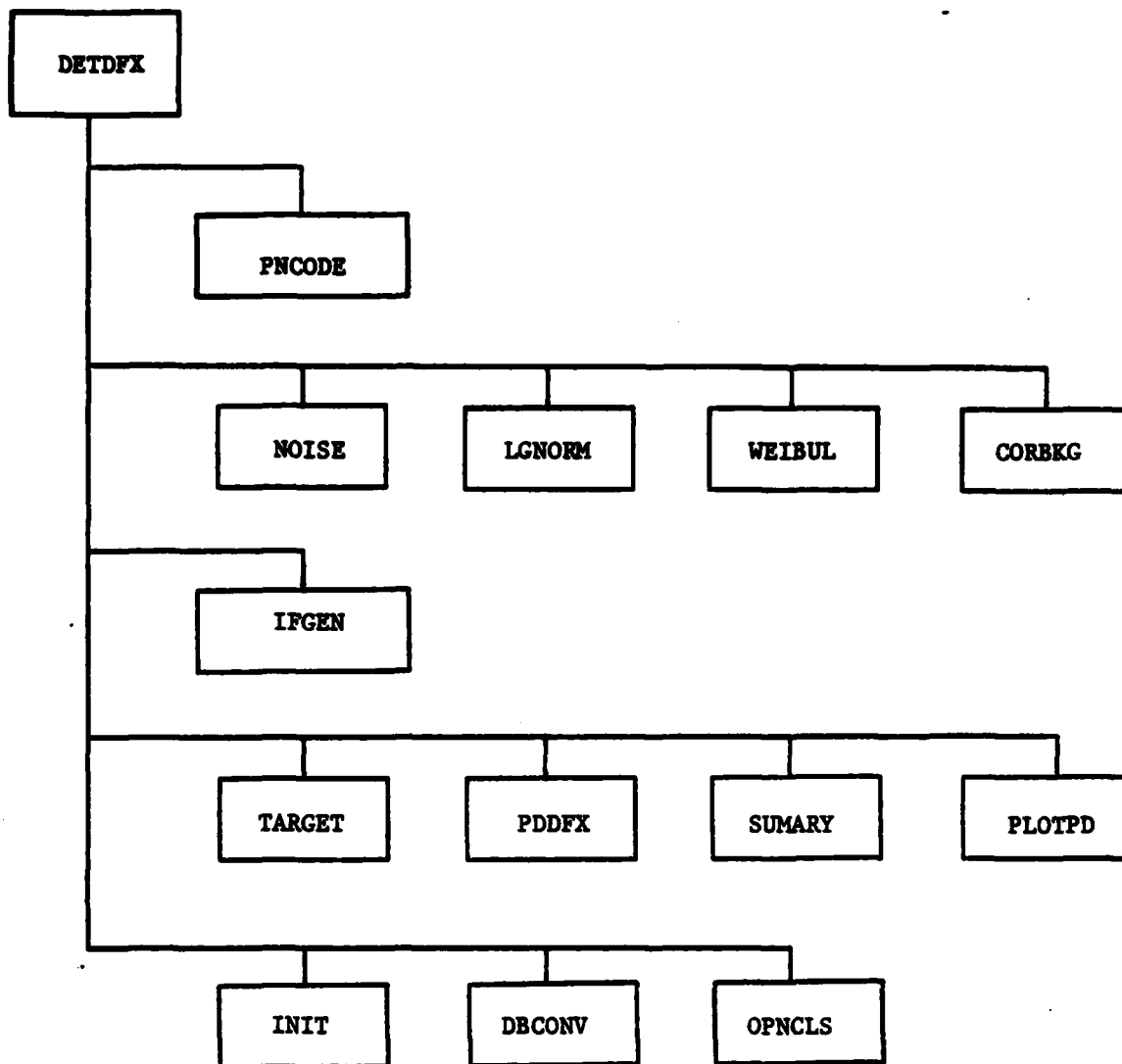


Figure 18 - Top Level Simulation Organization

The waveform and simple pulse radar backgrounds are combined to form a bipolar video signal for a pulse compression radar with a call to IFGEN. This subroutine convolves the transmit waveform produced by PNCODE with the backgrounds from the generators. The convolution is in the range dimension only. When the background is Gaussian noise, the call to IFGEN is by-passed.

Detection statistics are accumulated and summarized by the three modules TARGET, PDDFX, and SUMARY. TARGET is a target generator module, which returns samples of target amplitude which are either non-fluctuating (Swerling case 0), or one of the four Swerling cases 1-4. PDDFX is the module which implements the hard-limited pulse compression operations. It first adds the target signal, assembled from the waveform, the target amplitude generator, and an input value of signal-to-background ratio, to the linear signals generated by IFGEN. The sum is then hard-limited and pulse compressed. In-phase and quadrature channel outputs are squared and summed, and combined with the corresponding outputs from other diversity channels. The resultant is then subjected to a threshold test at each of several threshold levels. This process of target and background addition, pulse compression, diversity channel combination, and threshold comparison is repeated for all target-to-background ratios. A matrix of threshold crossing counts is accumulated by repeating for many independent trials. Module SUMARY summarizes these counts in a tabular format, expressed as probability of threshold crossing, as a function of threshold level and signal-to-background ratio. The PLOTDP module writes the threshold crossing probabilities to a file for off-line plotting. Additionally, it obtains a parametric fit to the data, writes the parameter values to the output file, and generates and writes to the plot file the smooth fit curve.

The three remaining modules are utilities for array initialization (INIT), linear to dB conversion (DBCONV), and output file opening and closing (OPNCLS).

4.2 Simulation Simplifications

As in any simulation, a number of simplifying assumptions have been invoked in the interest of efficiency. Some of the more significant simplifications are mentioned here.

4.2.1 Waveform Selection

As previously mentioned, the simulation is restricted to waveforms consisting of 0° and 180° phase code modulations of unity amplitude sinusoidal sub-pulses. The phase codes are generated from maximal length sequences with shift register generators. In general, there are several alternate shift register feedback connections which will generate a given length code. In addition, for each such set of connections, there is an arbitrary starting point in the resulting maximal length sequence, corresponding to the initial state of the shift register generator.

The classic approach to selecting from among all such possible resulting waveforms of a given length, is to examine the linear processing ambiguity function (point target matched filter response as a function of delay and Doppler mismatch) of each waveform, and select one which exhibits the "best" sidelobe behavior. In the non-linear processing schemes simulated here, this is still probably a reasonably good selection criterion. However, in the present simulation, an arbitrary selection of sequence has been made by using the shift register feedback connections suggested in Table 6 of Chapter 20 of the Radar Handbook [3], and an initial shift register state of all ones. The autocorrelation function sidelobes of this waveform are plotted in Figure 19. The peak sidelobe is about 19 dB below the mainlobe.

4.2.2 In-phase Signal Assumption

The target signal return is always assumed to have an initial RF phase of 0° , so that all target energy occurs in the in-phase arm of the I and Q channel processor. This differs from the actual situation in which target phase will be uniformly distributed over 360° , with target energy split between the two quadrature channels. For linear processing, this assumption of zero target phase has no effect on the performance of the unknown phase (I and Q channel) receiver, since the

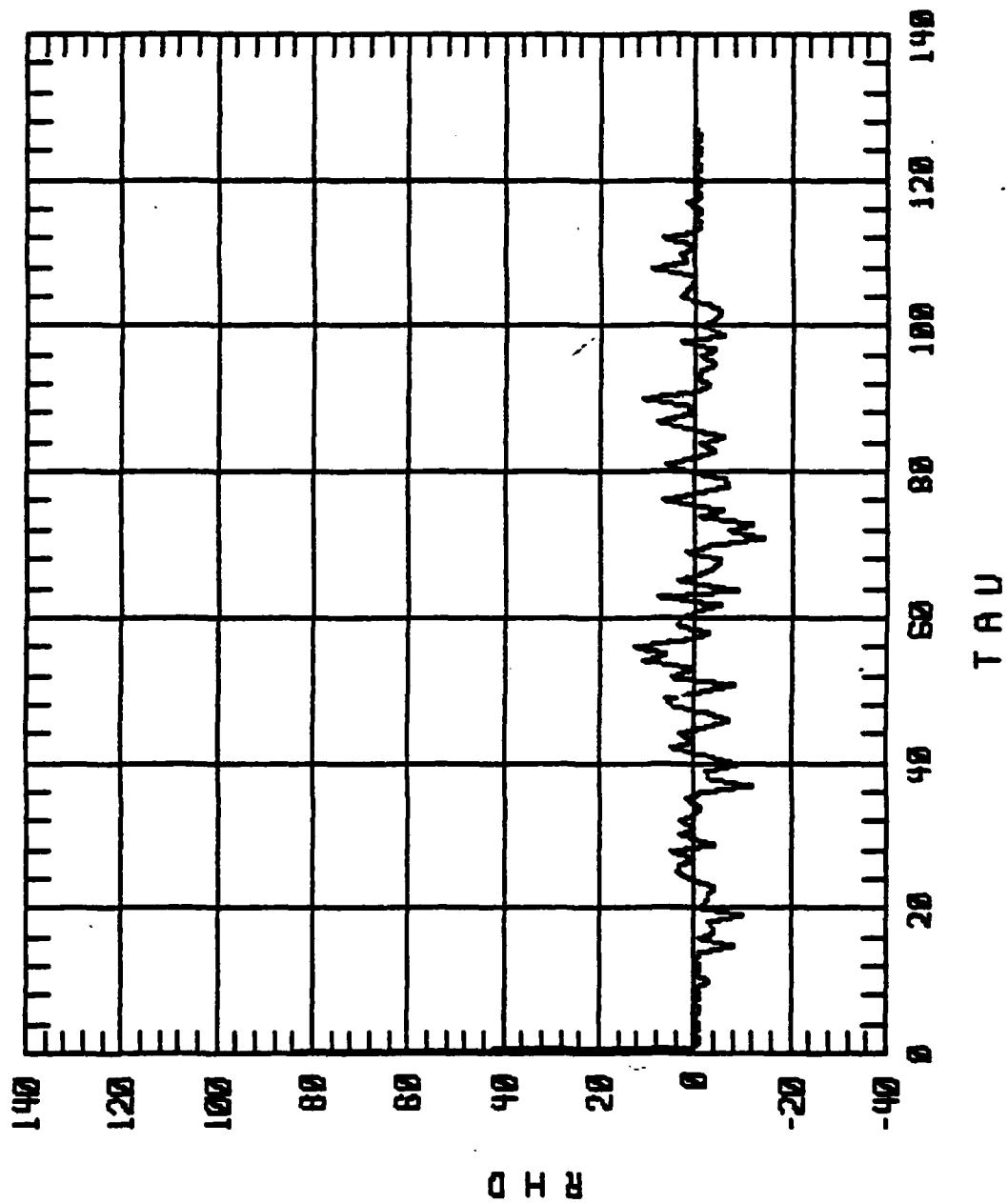


Figure 19 - Auto-Correlation Function of Transmit Waveform

receiver is equivalent to matching to an arbitrary phase followed by envelope detection [4].

The zero target phase assumption is believed to have little effect on performance of the hard-limiting receiver also, although this assumption has not been justified.

4.2.3 Beamshape, Range Bin, and Doppler Bin Scallop Losses Neglected

A target seldom falls on the main response axis of the radar beam, nor is it always aligned with the center of a range or Doppler bin. As the beam sweeps past the target, the sequence of target returns is modulated by the beamshape, with the result that the single pulse signal-to-noise ratio is not constant within the pulse sequence. A similar effect occurs due to mismatch in range and Doppler, often called scalloping.

These effects have been neglected in the simulation. Target amplitude is constant throughout the sequence of N pulses. Similarly, the target is presumed to fall in the center of a range bin, for all pulses. Finally, target Doppler is assumed to be zero.

REFERENCES

1. Hansen, V. G., and Zöttle, A. J., "The Detection Performance of the Siebert and Dicke-Fix CFAR Radar Detectors", IEEE Transactions on Aerospace and Electronic Systems, July 1971, pp 706-709.
2. Rivers, W., et. al., "A Review of Representation Functions for Probability of Detection", Technology Service Corporation Report No. TSC-W25-160, 11 January 1980.
3. Skolnik, Radar Handbook, McGraw Hill, 1970.
4. Whalen, Detection of Signals in Noise, Academic Press, 1971.
5. Mitchell, R. L., Radar Signal Simulation, Artech House, Inc. 1976.
6. Chambers, R. P., "Random Number Generation", IEEE Spectrum, February 1967, pp. 48-56.
7. Abramowitz and Stegun, Handbook of Mathematical Functions, Dover, 1970.

APPENDIX A

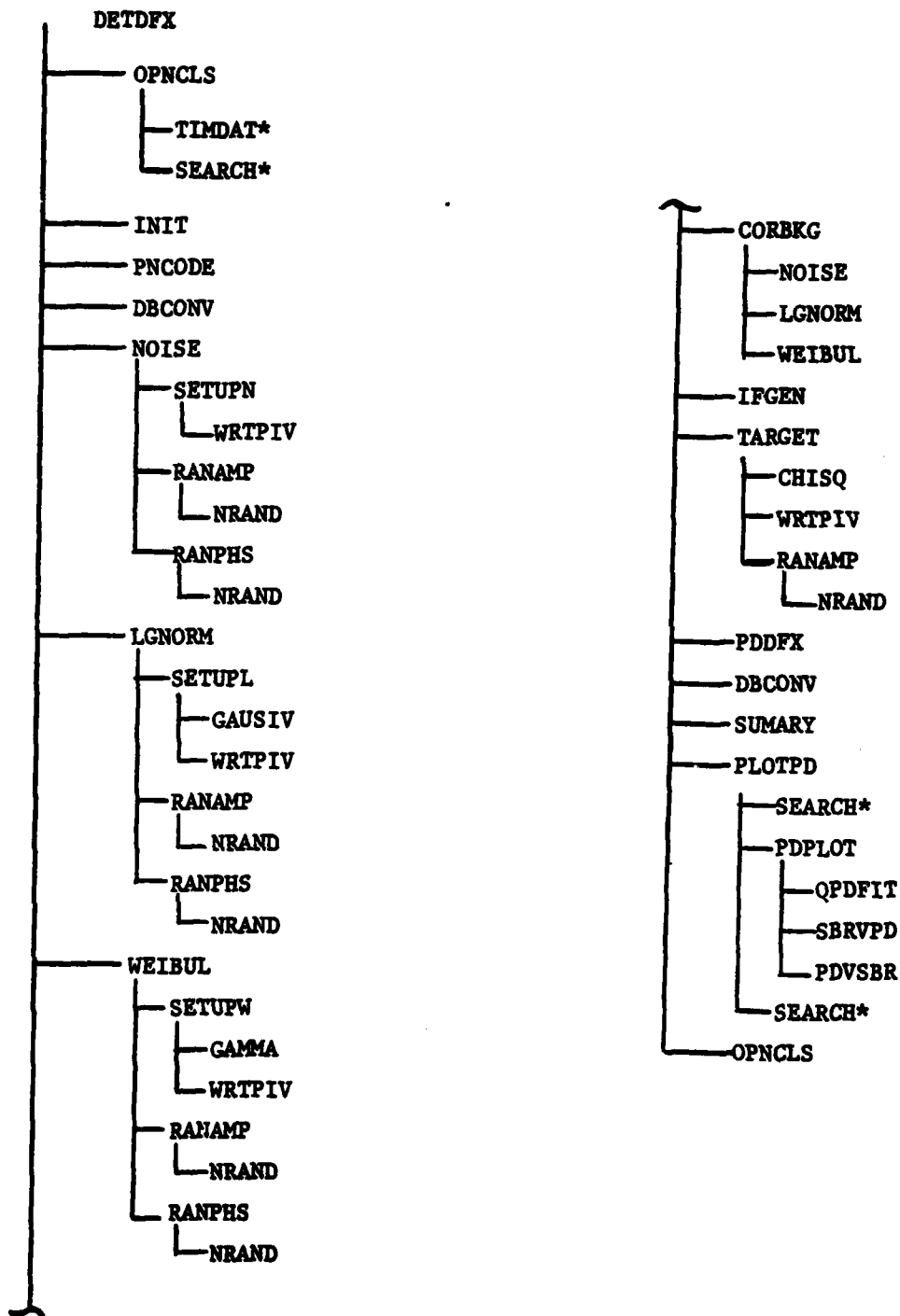
A.0 DETAILED SIMULATION PROGRAM DESCRIPTION

This appendix contains a more detailed description of the simulation program than is contained in the main body of the text. The description is done in two parts. The first part is module-by-module description of the entire program, in which a narrative discussion is given of the computation performed by each piece of the program. This is essentially an elaboration of the functional summary contained in Section 4.1. Following this, an example run is presented showing inputs and outputs.

A.1 Module Description

Figure A.1 presents the module hierarchy of the simulation program, showing the relationship of all subroutines. It is analagous to the organization chart shown in Figure 18, but contains greater detail, and presents the subroutines in the order of call. Each module in this chart is described in the paragraphs which follow. Section A.2 contains the example run.

Figure A.1 Module Hierarchy



* System Routine

A.1.1 Driver (DETD FX)

Calling Statement: driver module

Parameters Read from Input File*:

NTRIAL = number of Monte-Carlo range-bins for simulation run

NPDI = number of pulses (or frequency diversity channels)
for incoherent integration

M = pulse compression ratio

IPLOT = flag to specify generation of a plot file for off-
line plotting of results

IPLOT = 1 = generate plot file

IPLOT $\neq$ 1 = no plot file

NTRLPS = input number of range bins per scan.

ITYPE = background type code

ITYPE = 1 = Gaussian receiver noise.

ITYPE = 2 = Log-normal uncorrelated clutter

ITYPE = 3 = Weibull uncorrelated clutter

ITYPE = 4 = Correlated clutter

NTHR = number of threshold levels for accumulation of
threshold crossing counts

(THR(I), I=1, NTHR) = $10 \log_{10}$ (threshold levels)

NSBR = number of signal-to-background ratios

(SBR(I), I=1, NSBR) = signal-to-background ratios in dB.

Description:

The driver module (DETD FX) performs two functions. It serves as the executive routine which defines the sequence of functions needed for simulation of the signal processor described in Section 2.1. It also performs a limited amount of "book-keeping" calculations necessary for proper executive control.

The first operation is a call to subroutine OPNCLS, which opens input and output files on Fortran logical units LUIN and LUOUT. All program inputs are made on LUIN. With the exception of plot file outputs produced by subroutine PLOTDP, all program outputs are directed to LUOUT.

* Input parameter types and ranges of allowable values are summarized in Section A.2.

An unformatted read from LUIN sets values for parameters NTRIAL, NPDI, M, IPLOT, and NTRLPS. These parameters are defined briefly above. A value of zero for NTRIAL signals the end of a simulation run.

For non-zero NTRIAL, the next step is a call to subroutine INIT, which initializes all arrays in labelled common areas CNTRL, ARRAYS, and FLAGS. The total number of range bins (NTRIAL) will be simulated in blocks of NRBIN range bins at a time. This will be repeated for NSCAN blocks, until the specified total, NTRIAL, is reached. As a first cut, these two factors are set from the input variables NTRIAL and NTRLPS according to:

$$\text{NSCAN} = \text{NTRIAL} / \text{NTRLPS}$$

$$\text{NRBIN} = \text{NTRLPS}$$

This initial value of NRBIN is now checked to see if it requires NBIN, the number of complex sample pairs per call to the background generator NOISE, LGNORM, WEIBUL, and CORBKG, greater than 2048. If not, than these initial values are acceptable. Otherwise NRBIN is decremented and NSCAN recalculated:

$$\text{NSCAN} = \text{NTRIAL} / \text{NRBIN}$$

This is repeated until NBIN is less-than-or-equal-to 2048. When the constraint on NBIN is satisfied, NSCAN and NRBIN are checked to see if they produce at least the desired number of trials. If not, NSCAN is incremented, and NTRIAL re-calculated.

The remaining inputs, ITYPE, NTHR, THR, NSBR, and SBR are now read in from LUIN, and a summary of inputs is written to LUOUT.

A call to PNCODE generates an M-sample representation of the transmitted waveform, returning it in array WR. All samples of signal returned by PNCODE are either +1 or -1.

The next step is a conversion from dB units to linear units of THR and SBR. The threshold conversion is straightforward, according to

$$THR = 10^{(THR/10)}$$

The conversion of SBR is such that the converted value becomes the target amplitude which would produce the desired signal-to-background ratio at the output of a linear matched filter pulse compressor. This conversion depends on the type of background. For receiver noise, the conversion is

$$SBR = \frac{10^{(SBR/20)}}{\sqrt{M}}$$

For distributed clutter backgrounds, the conversion is simply

$$SBR = 10^{(SBR/20)}$$

Following conversion of SBR and THR, the main loop of the simulation begins. For each scan, a matrix of background samples is generated with a call to the appropriate background generator. The samples are returned in VUR and VUI by the receiver noise generator, and in XNR and XNI by the distributed clutter generators. The sample matrix is large enough such that NRBIN pulse-compressed range bin outputs can be simulated on each of NPDI pulses (or diversity channels). A call to IFGEN generates a simulated radar bipolar video signal by convolving the background samples with the waveform in the range dimension. The video samples are returned in the arrays VUR and VUI. If the background type is receiver noise, this convolution step is by-passed. A call to TARGET generates NRBIN x NPDI target samples, which are returned in array TGT.

Subroutine PDDFX combines target and background signals, hard-limits the in-phase and quadrature resultant signals, and pulse compresses. This is repeated on each range bin in a scan, on each of NPDI pulses (or diversity channels). The pulse compressed outputs are square-law detected ($I^2 + Q^2$) and a test statistic generated by summing over pulses. The test statistic is then compared to the vector of threshold levels THR, and crossing counts accumulated in NTC for each threshold and signal-to-background ratio combination.

Following the major loop, THR and SBR are re-converted to dB, and the threshold crossing counts are summarized as probability of threshold crossing (P_D) by a call to SUMMARY. An optional call to PLOTPD produces a parametric fit to each P_D vs SBR curve, and writes the simulated data points and curve fit to a file for off-line plotting.

The program then loops back to the beginning for another case. When all cases are finished (NTRIAL = 0), the program ends with a final call to OPNCLS to close LUIN and LUOUT.

```

C      PROGRAM TO SIMULATE DETECTION OF TARGETS IN
C      NOISE OR CLUTTER FOR DICKE-FIX CFAR.
C
C      INTEGER*4 NTRIAL, NSCAN
*INSERT JIMB JODI>SURADS>CNTROL
*INSERT JIMB JODI>SURADS>ARRAYS
*INSERT JIMB JODI>SURADS>IOLUS
C
C      DATA INEWPGE/:214/
C      CALL OPNCLS(LUIN, LUOUT, 1)
C
C      WRITE(LUOUT, 100)
100    FORMAT(////1X, 'DETECTION PERFORMANCE OF DICKE-FIX CFAR')
C
C      READ(LUIN, *) NTRIAL, NPDI, M, IPLOT, NTRLPS
C      IF(NTRIAL.EQ.0) GO TO 40
C
C      CALL INIT
C
C      MM1T2=(M-1)*2
C      NSCAN=NTRIAL/NTRLPS
C      NRBIN=NTRLPS
5      CONTINUE
C      NRM2=NRBIN+MM1T2
C      NBIN=NRM2*NPDI
C      IF(NBIN.LE.2048) GO TO 6
C      NRBIN=NRBIN-1
C      NSCAN=NTRIAL/NRBIN
C      GO TO 5
6      CONTINUE
C      IF((NSCAN*NRBIN).LT.NTRIAL) NSCAN=NSCAN+1
C      NTRIAL=NSCAN*NRBIN
C
C      READ(LUIN, *) ITYPE, NTHR, (THR(I), I=1, NTHR), NSBR, (SBR(I), I=1, NSBR)
C      WRITE(LUOUT, 200) NSCAN, NRBIN, IPLOT, ITYPE, NPDI, M
200    FORMAT(//6X, 'NSCANS = ', I5, 5X, 'NRBINS = ', I5, 5X, 'IPLOT = ', I5,
*//6X, 'BACKGROUND TYPE = ', I3, 5X, 'NPDI = ', I3, 5X, 'M = ', I3)
C      WRITE(LUOUT, 300) (THR(I), I=1, NTHR)
300    FORMAT(//6X, 'THRESHOLD TEST LEVELS : '//
*5(//6X, 4Q11.3//)
C      WRITE(LUOUT, 400) (SBR(I), I=1, NSBR)
400    FORMAT(//6X, 'SIGNAL-TO-BACKGROUND RATIOS (DB) : '//
*5(//6X, 4Q11.3//)
C
C      CALL PNCODE(WR, M)
C
C      CONVERT S/B RATIOS TO LINEAR UNITS
C
C      CALL DBCONV(SBR, NSBR, .05, 1)
C      CALL DBCONV(THR, NTHR, .1, 1)
C      IF(.NOT.(ITYPE.EQ.1)) GO TO 20
C      SAMP=1./SQRT(FLOAT(M))
C      DO 10 I=1, NSBR
C          SBR(I)=SAMP*SBR(I)
10      CONTINUE
20      CONTINUE
C
C      DO 30 I=1, NSCAN
C          IF(ITYPE.EQ.1) CALL NOISE(VUR, VUI, NBIN, -1)
C          IF(ITYPE.EQ.2) CALL LGNORM(XNR, XNI, NBIN, -1)
C          IF(ITYPE.EQ.3) CALL WEIBUL(XNR, XNI, NBIN, -1)
C          IF(ITYPE.EQ.4) CALL CORBKQ(XNR, XNI, CR, NRM2, NPDI, -1, 1)
C          IF(ITYPE.NE.1)
*      CALL IFGEN(VUR, VUI, XNR, XNI, NRM2, NPDI, WR, M)
C          CALL TARGET(TGT, NRBIN, NPDI)
C          CALL PDDFX(M)
30      CONTINUE
C
C      CONVERT S/B RATIOS TO DB
C
C      IF(.NOT.(ITYPE.EQ.1)) GO TO 36
C      DO 32 I=1, NSBR

```

```

      SBR(I)=SBR(I)/SAMP
32  CONTINUE
36  CONTINUE
    CALL DBCONV(SBR, NSBR, 20., -1)
    CALL DBCONV(THR, NTHR, 10., -1)
C
    WRITE(LUOUT, 460) INEWPQ
460  FORMAT(A2)
    CALL SUMARY(NTC, SBR, THR, NSBR, NTHR, NTRIAL)
    IF(IPLT, EQ. 1) CALL PLOTPD(NTC, SBR, NSBR, NTHR, NTRIAL)
    WRITE(LUOUT, 500) INEWPQ
500  FORMAT(A2/70(' '))
    GO TO 1
C
40  CALL OPNCLS(LUIN, LUOUT, 0)
    STOP
    END

```

INTEGER*4 NTC, NSCAN
COMMON/CNTROL/NTC(400), SBR(20), THR(20),
* NSBR, NTHR, NSCAN, NRBIN, NPDI, CNR

C

COMMON/ARRAYS/CR(2048), CI(2048), XNR(2048), XNI(2048),
* VUR(2048), VUI(2048), VCR(2048), VCI(2048),
* TGT(2048), WR(256), WI(256)

C

COMMON/IOLUS/LUIN, LUOUT

C

COMMON/FLAGS/IFLAGN, IFLAGL, IFLAGW, IFLAGT, IFLAG(100)

C

A.1.2 File Open and Close (OPNCLS)

Calling Statements: Call OPNCLS (LUIN, LUOUT, ICODE)

Argument List Definition:

Input Variables:	ICODE = open/close code (1 → open, 0 → close)
Returned Variables:	LUIN = input logical unit LUOUT = output logical unit

Other Modules Called: SEARCH, TIMDAT

Description:

OPNCLS opens/closes input/output files, using system subroutine SEARCH, if job is a batch job ($\text{NUSER} \geq 10$). Input file is 'CFARIN', output file is 'CFROUT'. LUIN and LUOUT are set to 5 and 6, respectively.

If job is conversational, LUIN and LUOUT are set to 1 and 2, and all I/O is to user terminal.

In addition, OPNCLS obtains time and date of run from system subroutine TIMDAT, and documents on LUOUT.

```

SUBROUTINE OPNCLS(LUIN, LUOUT, ICODE)
CCCC
SUBROUTINE TO SET FORTRAN LOGICAL UNITS
LUIN AND LUOUT, AND OPEN AND CLOSE DISK
INPUT AND OUTPUT FILES FOR PHANTOM JOBS.
CCCC
INTEGER OUTFIL, OUTDSK
DIMENSION INFO(15), INFILE(3), OUTFIL(3)
DATA INFILE, OUTFIL/'CF', 'AR', 'IN', 'CF', 'RO', 'UT'/
DATA INDISK, OUTDSK/1, 2/
DATA NPHAN/10/
C
IF(.NOT. (ICODE. EQ. 1)) GO TO 30
CCCC
OPEN FILES
CALL TIMDAT(INFO, 15)
NUSER=INFO(12)
IMIN=MOD(INFO(4), 60)
IHRS=INFO(4)/60
ISEC=INFO(5)
IF(.NOT. (NUSER. GE. NPHAN)) GO TO 10
C
PHANTOM JOB:
LUIN=INDISK+4
LUOUT=OUTDSK+4
CALL SEARCH(1, INFILE, INDISK, $100)
CALL SEARCH(2, OUTFIL, OUTDSK, $200)
GO TO 20
C
CONTINUE
C10
INTERACTIVE JOB:
LUIN=1
LUOUT=1
C
CONTINUE
C20
WRITE TIME AND DATE OF RUN
WRITE(LUOUT, 900) (INFO(I), I=1, 3), IHRS, IMIN, ISEC,
* (INFO(I), I=13, 15), INFO(12)
900 * FORMAT(///1X, 50('*')/8X, A2, 2(' ', A2), 2X, 2(I2, ': '), I2,
* 2X, 3A2, ' (USER #', I2, ')'/1X, 50('*')/)
GO TO 40
C
CONTINUE
C30
CLOSE FILES
IF(.NOT. (NUSER. GE. NPHAN)) GO TO 40
C
PHANTOM JOB:
CALL SEARCH(4, INFILE, INDISK, $300)
CALL SEARCH(4, OUTFIL, OUTDSK, $400)
C
CONTINUE
40 RETURN
100 STOP 1
200 STOP 2
300 STOP 3
400 STOP 4
END

```

A.1.2.1 System Routine TIMDAT

* TIMDAT *

The calling sequence is:

CALL TIMDAT (array, num)

TIMDAT returns the date, time, CPU time, and paging time used since LOGIN, the users unique number on the system, and his login UFD name in an array as follows:

- array (1) Two ASCII characters representing month. Example: 11
- (2) Two ASCII characters representing day. Example: 30
- (3) Two ASCII characters representing year
Example: 75
- (4) Integer time in minutes since midnight.
- (5) Integer time in seconds.
- (6) Integer time in ticks.
- (7) Integer CPU time used in seconds.
- (8) Integer CPU time used in ticks.
- (9) Integer disk I/O time used in seconds.
- (10) Integer disk I/O time used in ticks.
- (11) Integer number of ticks per second.
- (12) User number.
- (13) Six-character login name, left-justified.
- (14)
- (15) Example: MSMITH

num words of array are set.

A.1.2.2 System Routine SEARCH

* SEARCH *

Definition of SEARCH

SEARCH is used to connect a file to a file unit (open a file) or disconnect a file from a file unit (close a file). After a file is connected to a unit, PRWFIL and other routines may be called, either to position the current-position pointer of a file unit (file pointer) or to transfer information to or from the file (using the file unit to reference the file).

Opening a File

On opening a file, SEARCH specifies 1) allowable operations that may be performed by PRWFIL and other routines (these operations are read only, write only, or both read and write); 2) where to look for a file or where to add the file, if the file does not already exist; and 3) whether the file is to be opened for writing only or both reading and writing. SEARCH either specifies a filename in the currently attached user file directory or a file unit number on which a segment directory is open. In the segment directory reference, the file to be opened or closed has its beginning disk address given by the word at the current position pointer of the file unit.

SEARCH Actions

On creating a new file, the user specifies to SEARCH the file type of the new file.

The subroutine SEARCH may be used to perform actions other than opening and closing a file. SEARCH may delete a file, rewind a file unit, or truncate a file.

Upon opening a file, SEARCH sets the file pointer to the beginning of the file. Subroutines PRWFIL and others cause information to be transferred to or from the file unit, starting at the file pointer. After the transfer, the pointer is moved past the data transferred. A call to SEARCH to rewind a file causes the file pointer to be set to the beginning of the file. Subsequent calls to PRWFIL and other routines cause information transfer to occur as if the file had just been opened. A call to SEARCH to truncate a file causes all information beyond the file pointer to be removed from the file. This call is useful if one is overwriting a file with less information than was originally contained in the file.

Subroutine Call

SEARCH is used as in the following call:

Format:

CALL SEARCH (KEY, NAME, FUNIT, ALTRIN)

KEY is composed of three subkeys that are combined additively: ACTION, REFERENCE, and NEWFILE. Not all subkeys are required on every call, and subkeys with values of zero can be omitted. The SEARCH call may therefore be represented as:

CALL SEARCH (ACTION+REFERENCE+NEWFILE, NAME, FUNIT, ALTRIN)

All calls require an ACTION subkey. The ACTION subkeys are shown in the following table:

<u>ACTION</u>	<u>Octal Value</u>	<u>Meaning</u>
OPNRED	1	Open NAME for reading on FUNIT
OPNWRT	2	Open NAME for writing on FUNIT
OPNBTH	3	Open NAME for both reading and writing on FUNIT
CLOSE	4	Close file by NAME or by FUNIT
DELETE	5	Delete file NAME
EXIST	6	Check to see if file exists.
REWIND	7	Rewind file on FUNIT
TRNCAT	10	Truncate file on FUNIT
CNGACC	1000	Change access of file to FUNIT

The REFERENCE subkeys are shown in the following table:

<u>REFERENCE</u>	<u>Octal Value</u>	<u>Meaning</u>
UFDREF	0	Searches for file NAME in the current user file directory (UFD) (as defined by a previous ATTACH) and perform the action in the ACTION subkey on the specified file.
SEGREF	100	Performs the action specified in the ACTION subkey on the file with the location indicated by the file pointer designated within the array NAME(1). This file unit must be an open segment directory.

Only those calls to SEARCH that reference a file in a UFD or segment directory need the reference key. Calls that reference file units do not need this key.

The following table lists the NEWFIL subkeys:

<u>NEWFIL</u>	<u>Octal Value</u>	<u>Meaning</u>
NIFILE	0	New threaded (SAM) file
NDFILE	2000	New directed (DAM) file
NISEG	4000	New threaded (SAM) segment directory
NDSEG	6000	New directed (DAM) segment directory
NEWUFD	10000	New User File Directory (SAM)

Only those calls to SEARCH that generate a new file require a NEWFIL subkey. On other calls, this subkey is ignored.

The name of the remaining parameters in a call to SEARCH are as follows:

NAME If the reference subkey is UFDREF, NAME is either a six-character Hollerith expression or the name of a three-word array that specifies a filename (existing or not).

If the reference subkey is UFDREF and NAME(1) is -1, the current UFD is opened. NAME = -1 must be used only in configuration with ACTION subkeys 1, 2, or 3. Owner status of the current UFD is required.

If the reference subkey is SEGREF, NAME is a file unit(1-16; 1-15 under PRIMOS II) on which a segment directory is open.

On calls in which the ACTION key requires only a file unit to specify the file to be acted on, NAME is ignored and, usually, specified as 0.

FUNIT On calls that require a file unit number, FUNIT is a number 1 to 16 (1-15 under PRIMOS II). On calls that require no unit number, FUNIT is ignored and usually specified as 1.

ALTRIN ALTRIN is an integer variable assigned the value of a label return in the user's FORTRAN program to be used as an alternate in case of uncorrectable errors (e.g., attempting to open a file that is already open). If this argument is 0 or omitted, an error message is printed; control returns to PRIMOS if any error should occur while using SEARCH.

A.1.3 Array Initialization (INIT)

Calling Statement: Call INIT

Description:

 Initializes signal arrays and threshold crossing accumulators.
Also sets flags to initialize inverse probability tables in NOISE, WEIBUL,
LGNORM, TARGET, and CORBKG.

```

SUBROUTINE INIT
C
*INSERT CNTROL
*INSERT ARRAYS
*INSERT FLAGS
C
DO 10 I=1,2048
  CR(I)=0.0
  CI(I)=0.0
  XNR(I)=0.0
  XNI(I)=0.0
  VUR(I)=0.0
  VUI(I)=0.0
  VCR(I)=0.0
  VCI(I)=0.0
  TOT(I)=0.0
10 CONTINUE
C
DO 20 I=1,256
  WR(I)=0.0
  WI(I)=0.0
20 CONTINUE
C
DO 30 I=1,400
  NTC(I)=0
30 CONTINUE
C
DO 40 I=1,20
  SBR(I)=0.0
  THR(I)=0.0
40 CONTINUE
C
IFLAGN=0
IFLAGW=0
IFLAGL=0
IFLAGT=0
DO 50 I=1,100
  IFLAG(I)=0
50 CONTINUE
C
RETURN
END

```

A.1.4 Waveform Generation (PNCODE)

Calling Statement: CALL PNCODE (WR, M)

Argument List Definitions:

Input variable: M = length of pseudo-noise code
to be generated

Returned variables: WR = vector of M waveform samples,
all of which are either +1 or -1.

Other Modules Called: None

Description

PNCODE generates a pseudo-noise pulse compression waveform derived from a maximal length shift register generator. Such generators are described in Chapter 20 of the Radar Handbook [3]. The length M of the waveform (or, equivalently, the pulse-compression ratio) is an input to the subroutine. This length is then converted into the number of stages in the shift register using

$$\text{NSTAGE} = \log_2 (M + 1)$$

If $M + 1$ is not initially a power of two, it is rounded down to the next lower such power. The shift register is then initialized to all one's. A sequence of one's and zero's is generated by using feedback connections suggested in Table 6 of Chapter 20 of [3]. The one's and zero's are converted to +1 or -1, respectively, which are returned as waveform samples. Prior to returning to the calling program, the shift register generator and resulting waveform are summarized on LUOUT.

A.1.5 DB Conversion (DBCONV)

Calling Statement: CALL DBCONV (X,N,P,ICODE)

Argument List Definitions:

X = array of values to be converted

N = length of X array

P = power law to use in conversion (see below)

ICODE = direction of conversion

ICODE = 1 = DB to linear conversion

ICODE $\neq$ 1 = linear to DB conversion

Other Subroutines Called: None

Description:

Converts to or from dB, according to the following expressions:

If ICODE = 1:

$$X' = 10^{X \cdot P}$$

If ICODE $\neq$ 1:

$$X' = P \cdot \log_{10}(X)$$

```

C      SUBROUTINE DBCONV(X,N,P,ICODE)
C      CONVERSION TO AND FROM DB
C      DIMENSION X(1)
C      IF(.NOT.(ICODE.EQ.1)) GO TO 20
C      DO 10 I=1,N
C          X(I)=10.**(X(I)*P)
10     CONTINUE
C      GO TO 40
C      CONTINUE
20     DO 30 I=1,N
C          X(I)=P*ALOG10(X(I))
30     CONTINUE
C      CONTINUE
40     RETURN
      END

```

A.1.6 Gaussian Noise Generator (NOISE and SETUPN)

Calling Statement: CALL NOISE (XR, XI, N, IREP)

Parameters Read from Input File:

ISEEDN = starting seed for pseudo-random number generator

Argument List Definition:

Input Variables: N = number of complex samples generated
by each call

IREP = representation code

Returned Variables: XR and XI are defined according to
IREP, as discussed below.

Other Modules Called: SETUPN, RANAMP, RANPHS, WRTPIV

Description:

For IREP = -1 NOISE generates N complex samples of Gaussian random noise of unit mean power, and uniformly distributed phase. That is,

$$XR = \sqrt{Y} \sin \phi$$

$$XI = \sqrt{Y} \cos \phi$$

where Y is exponentially distributed with probability density

$$p(Y)dY = \exp\{-Y\} \quad , \quad Y \geq 0$$

and

$$\phi = \tan^{-1}(XR, XI) = \text{uniform on } (-\pi, \pi).$$

$$E[XR^2 + XI^2] = E[Y] = 1.0$$

It follows that XR and XI are uncorrelated samples from a Gaussian distribution, with density

$$p(X)dx = \frac{1}{\sqrt{2\pi} \sigma} \exp\left\{-\frac{X^2}{2\sigma^2}\right\}$$

where

X is XR or XI

and

$$\sigma^2 = 1/2.$$

In IREP = -2, NOISE generates sine and cosine of uniformly distributed phase. That is,

$$XR = \sin \phi$$

$$XI = \cos \phi$$

For IREP = -3, XR and XI are hard-limited:

$$XR = \text{sign}(\sin \phi)$$

$$XI = \text{sign}(\cos \phi)$$

For non-negative values of IREP, NOISE returns detected values for square-law, linear-law, and logarithmic detectors:

$$\text{IREP} = +2: \begin{cases} XR = Y \\ XI = 0 \end{cases}$$

$$\text{IREP} = +1: \begin{cases} XR = \sqrt{Y} \\ XI = 0 \end{cases}$$

$$\text{IREP} = 0: \begin{cases} XR = \ln \sqrt{Y} \\ XI = 0 \end{cases}$$

On the first call to NOISE, a tabulation of the inverse cumulative distribution for $\sqrt{Y}$, Y , or $\ln \sqrt{Y}$ is generated and stored in the array PIV. This is done with a call to SETUPN. The table is a three-level table, the first level evaluated at the center of 128 uniform increments of the 0 to 1 probability axis. The second level divides the last (nearest unity) such increment into 128 sub-increments, and the third level divides the last sub-increment into 128 sub-sub-increments. Thus the table is truncated at probabilities of $\frac{.5}{128} = .0039$ and $1 - \frac{.5}{(128)^3} = 1 - 2.38 \times 10^{-7}$.

Once the table is generated, random amplitude samples are drawn from the tabulated distribution by a call to RANAMP. Subroutine RANPHS converts to inphase and quadrature components. Subroutine WRTPIV writes out the PIV table for debugging purposes.

```

SUBROUTINE NOISE(XR,XI,N,IREP)
GAUSSIAN NOISE GENERATOR
IREP RESULT
0: RETURNS NATURAL LOGARITHM OF RAYLEIGH AMPLITUDES IN XR.
1: RETURNS N UNCORRELATED RAYLEIGH AMPLITUDES IN XR,
  OF UNIT MEAN POWER.
2: RETURNS N UNCORRELATED EXPONENTIAL VARIABLES IN XR,
  OF UNIT MEAN POWER.
-1: RETURNS N UNCORRELATED COMPLEX GAUSSIAN SAMPLE PAIRS,
  MEAN=0, VARIANCE=0.5. RETURNS N SAMPLES IN XR,
  N SAMPLES IN XI. USES 3-LEVEL TABLE LOOK-UP FOR
  AMPLITUDES, UNIT CIRCLE ALGORITHM FOR SIN & COS. /
-2: RETURNS SIN & COS (XR & XI) OF N UNCORRELATED
  UNIFORMLY DISTRIBUTED PHASES.
-3: RETURNS SQN(SIN) & SQN(COS) OF N UNCORRELATED PHASES.

INTEGER*4 ISEEDN, ISEED
*INSERT JIMB ANITA>SURADS>IOLUS
*INSERT JIMB ANITA>SURADS>FLAGS
DIMENSION XR(1), XI(1), PIV(384)
DATA PIV, LPIV, NPIV, ISEED/384*1.0, 3, 7, 1735483429/

C
C ON FIRST CALL INITIALIZE TABLE
C
IF(.NOT. (IFLAGN.EQ.0)) GO TO 25
WRITE(LUOUT,100)
100 FORMAT(////1X, 'GAUSSIAN NOISE GENERATOR')
READ(LUIN,*) ISEEDN
IF(ISEEDN.EQ.0) ISEEDN=ISEED
200 WRITE(LUOUT,200) ISEEDN
FORMAT(//6X, 'PRN GENERATOR SEED = ', I12)
300 WRITE(LUOUT,300) IREP
FORMAT(//6X, 'REPRESENTATION CODE= ', I3)
C
CALL SETUPN(PIV, NPIV, LPIV, IREP)
IFLAGN=1
C
C25 CONTINUE
C
IF(IREP.LT.-1) GO TO 28
C
C GENERATE RANDOM AMPLITUDES (OR POWERS)
C
CALL RANAMP(PIV, NPIV, LPIV, ISEEDN, XR, N)
C
IF IREP<0 GENERATE AND APPLY RANDOM PHASE
C
28 CONTINUE
IF(IREP.GE.0) GO TO 30
CALL RANPHS(XR, XI, N, ISEEDN, IREP)
C
30 RETURN
END

```

```

SUBROUTINE SETUPN(PIV,NPIV,LPIV,IREP)
C
C      SETUP INVERSE PROBABILITY TABLE FOR GAUSSIAN NOISE GENERATOR
C
      DOUBLE PRECISION PROB
*INSERT JIMB ANITA>SURADS>IOLUS
      DIMENSION PIV(1)
C
      EXPO=.5
      IF(IREP.EQ.2) EXPO=1.0
      NN=2**NPIV
      JLP=0
      DO 20 LP=1,LPIV
        DO 10 J=1,NN
          JLP=JLP+1
          PROB=1.DO+DBLE(FLOAT(J-NN)-.5)/(DBLE(FLOAT(NN))**LP)
          PIV(JLP)=SNGL(-DLOG(1.DO-PROB))*EXPO
          IF(IREP.EQ.0) PIV(JLP)=ALOG(PIV(JLP))
10      CONTINUE
20      CONTINUE
C
      CALL WRTPIV(PIV,NPIV,LPIV,LUOUT)
C
30      RETURN
      END

```

A.1.7 PIV Documentation (WRTPIV)

Calling Statement: Call WRTPIV (PIV, NPIV, LPIV, LUOUT)

Argument List Definition:

Input Variables:

PIV = table to be documented

LPIV = number of levels in table

NPIV = $\log_2$ (number of increments
per level)

LUOUT = output logical unit
documentation

Description:

A call to WRTPIV writes out the contents of PIV to LUOUT, using formatted writes. Useful for debugging.

```

C      SUBROUTINE WRTPIV(PIV,NPIV,LPIV,LUOUT)
C
C      WRITE OUT CONTENTS OF PIV,
C      THE INVERSE PROBABILITY TABLE.
C
      DOUBLE PRECISION P,DP,DP1,DPO
      DIMENSION PIV(2),P(5),DP(5),X(5),ILABL(10)
      DATA ILABL/'PR','OB','(P','CT',' ',' ','LE','VE','L ',' '//'
C
      NN=2*NPIV
      DPO=100. DO
      DO 10 L=1,LPIV
        DP1=DPO/DBLE(FLOAT(NN))
        P(L)=100. DO-DPO-DP1*.5DO
        DP(L)=DP1
        DPO=DP1
10     CONTINUE
C
      WRITE(LUOUT,100)((ILABL(I),I=1,10),L=1,LPIV)
100    FORMAT(///6X,'INVERSE PROBABILITY TABLE'
*//6X,5(10A2,2X)//)
      DO 30 N=1,NN
        INDX=N
        DO 20 L=1,LPIV
          X(L)=PIV(INDX)
          P(L)=P(L)+DP(L)
          INDX=INDX+NN
20     CONTINUE
      WRITE(LUOUT,200)((P(L),X(L)),L=1,LPIV)
200    FORMAT(6X,5(OPF9.5,1PG11.3,2X))
30     CONTINUE
C
      RETURN
      END

```

A.1.8 Random Amplitude Generator (RANAMP)

Calling Statement: Call RANAMP (PIV, NPIV, LPIV, ISEED, X, K)

Argument List Definition: PIV = inverse cumulative probability
tabulation

LPIV = number of levels in table (3 or less)

NPIV = $\log_2$ (number of increments per
level) (NPIV $\leq$ 10)

Input Variables: ISEED = pseudo - random number generator
seed. (double length integer)

K = number of samples generated per call

Output Variables: X = array of samples from tabulated
distribution

ISEED = modified seed

Other Modules Called: NRAND, SHFT, INTS

Description:

RANAMP implements the random number generation via multi-level table look-up. For example, for a three level table, with 128 increments per level, random amplitude samples are drawn from the distribution tabulated in PIV by generating integers uniformly distributed between 0 and 127, and using this as an index to level 1 of the table. When a value of 0 is drawn for the index, a second random integer is drawn to index level 2. When the second index comes up 0, a third integer is drawn to index level 3. This table look-up procedure follows one by Mitchell [5].

For each sample to be generated a 32-bit pseudo-random integer is generated with a call to NRAND. A 7-bit positive integer is obtained from this with an appropriate shift-left, shift-right sequence. If this integer is non-zero, it is used as the index to a level 1 table look-up, completing the random sample generation. If it is zero, 7 more non-overlapping bits are shifted out, and the process repeated for level 2 and 3 (if necessary) of the table.

```

C      SUBROUTINE RANAMP(PIV,NPIV,LPIV,ISEED,X,K)
C      SUBROUTINE WHICH GENERATES RANDOM VARIABLES
C      USING LPIV-LEVEL TABLE LOOK-UP. PIV IS
C      TABLE CONTAINING INVERSE CUMULATIVE
C      PROBABILITY DISTRIBUTION FUNCTION. TABLE
C      LOOK-UP ADDRESS IS GENERATED WITH CALL TO
C      PRN SEQUENCE GENERATOR NRAND. NPIV BITS
C      OF PRN ARE USED FOR EACH LOOK-UP.
C
C      INTEGER*4 ISEED,NRAND,ILOC,ILOC2
C      DIMENSION PIV(1),X(1)
C      DATA IBIT1/32/
C
C      NN=2*NPIV
C      ISHRT=32-NPIV
C      DO 40 I=1,K
C          ILOC=NRAND(ISEED)
C          LEVEL=1
C          IOFFST=0
C          ISHLFT=IBIT1-32
C      30      CONTINUE
C          ILOC2=SHFT(ILOC,ISHLFT,ISHRT)
C          ILOC3=INTS(ILOC2)
C          IF(ILOC3.NE.0) GO TO 36
C          IF(LEVEL.EQ.LPIV) GO TO 35
C          LEVEL = LEVEL+1
C          IOFFST=IOFFST+NN
C          ISHLFT=ISHLFT-NPIV
C          GO TO 30
C      35      CONTINUE
C          ILOC3=NN
C      36      CONTINUE
C          ILOC3=ILOC3+IOFFST
C          X(I)=PIV(ILOC3)
C      40      CONTINUE
C
C      RETURN
C      END

```

A.1.8.1 System Function SHFT

SHFT - Logical Shift Function

SHFT performs logical shift operations on integer variables.

Calling Sequence:

Format 1:

IS = SHFT (I, IP1)

performs a shift operation on the variable. If IP1>0, the shift is to the right; if IP1<0, the shift is to the left; if IP1=0, no shift occurs. This operation is equivalent to the RS function, and is provided for compatibility with other FORTRAN compilers.

Format 2:

IS = SHFT (I, IP1, IP2)

Format 2 performs two shift operations, first by IP1 (setting zeroes in vacated bits), then by IP2 (setting zeroes in vacated bits). The sign of IP1 and IP2 determine the direction of the shift while their magnitude determines the number of bits to be shifted.

A.1.8.2 System Function INTS

INTS converts its argument from a double length (32 bit) to a single length (16 bit) integer.

A.1.1.9 Pseudo-Random Number Generator (NRAND)

Calling Statement: ILOC = NRAND (ISEED)

Argument List Definition:

Input Variables:	ISEED = double precision integer pseudo-random number seed.
Output Variables:	NRAND = double precision integer pseudo-random number
	ISEED = NRAND

Other Modules Called: SHFT

Description:

NRAND generates a pseudo-random number sequence by the congruential method [6]. That is,

$$X_{i+1} = [a X_i]_{\text{mod } 2^{32}}$$

The period length of this sequence is maximized at

$$\text{period length} = 2^{29} \text{ samples}$$

by setting:

$$x_0 \text{ odd}$$

$$a = 8t + 3, t = 1, 2, 3 \dots$$

The value for t which has been used here is

$$t=8192$$

for which

$$\begin{aligned} a &= 8 \times 8192 + 3 \\ &= 65539 \\ &= 2^{16} + 2^1 + 2^0. \end{aligned}$$

Hence, the multiplication is implemented by two shifts and two adds.

C
C
C
C

INTEGER*4 FUNCTION NRAND(IS)
DOUBLE PRECISION PRN GENERATOR
SEQUENCE LENGTH = 2**29

C

INTEGER*4 IS1, IS2, IS
IS1=SHFT(IS, -16)
IS2=SHFT(IS, -1)
IS=IS+IS2
IS=IS+IS1
NRAND=IS

RETURN
END

A.1.10 Random Phase Generator (RANPHS)

Calling Statement: Call RANPHS (XR, XI, N, ISEED, IREP)

Argument List Definition:

Input Variables:

XR = input vector of amplitudes
(generated by RANAMP)

N = Number of amplitude samples

ISEED = Pseudo-random No. generator seed

IREP = representation code

Returned Variables:

XR and XI are defined according
to IREP, as discussed below.

ISEED = modified seed

Other Modules Called: NRAND, SHFT

Description:

RANPHS generates sine and cosine of a uniformly distributed random phase using method attributed to Von Newman by Mithcell [5]. The procedure is based on generating (A,B) coordinate pairs uniformly distributed within a unit circle, and computing sine and cosine terms from these coordinates using trigonometric identities.

The coordinates are generated from a 32-bit random integer obtained from NRAND, from which two 10-bit 2's complement integer words are obtained. Each such word, A or B, is uniformly distributed on [-511, 512]. The sum of the squares of these two words is compared to $(511)^2 = 261121$, to test if the coordinate pair is within the "unit" circle, where "unit" radius is 511. If so, the sine and cosine terms are generated using:

$$\cos \phi = \frac{2AB}{A^2 + B^2}$$

$$\sin \phi = \frac{A^2 - B^2}{A^2 + B^2}$$

If not, another coordinate pair is generated and the process repeated until a point in the unit circle is obtained.

The value of IREP determines the returned values of XR and XI.

That is:

$$\text{IREP} = -1: \begin{cases} \text{XR} = \text{XR}_{\text{in}} \sin\phi \\ \text{XI} = \text{XR}_{\text{in}} \cos\phi \end{cases}$$

$$\text{IREP} = -2: \begin{cases} \text{XR} = \sin\phi \\ \text{XI} = \cos\phi \end{cases}$$

$$\text{IREP} = -3: \begin{cases} \text{XR} = \text{sign}(\sin\phi) \\ \text{XI} = \text{sign}(\cos\phi) \end{cases}$$

SUBROUTINE RANPHS(XR, XI, N, ISEED, IREP)

GENERATES SIN AND COS OF UNIFORM PHASE,
USING UNIT CIRCLE ALGORITHM.

IREP RETURNS

-1: XI=XR*COS
XR=XR*SIN
-2: XI=COS
XR=SIN
-3: XI=+/-1
XR=+/-1

INTEGER*4 ISEED, INTGR, IRAND, NRAND, INTA, INTB, INTT, INTT2
DIMENSION XR(1), XI(1), INTGR(2), ISHL(2)
EQUIVALENCE (INTA, INTGR(1)), (INTB, INTGR(2))

ISHRT=32-10
ISHL(1)=32-32
ISHL(2)=22-32

DO 40 I=1, N

GENERATE DOUBLE PRECISION PRN INTEGER

IRAND=NRAND(ISEED)

SHIFT OUT TWO 10-BIT 2'S COMPLEMENT WORDS
(-512. LE. INT1. LE. 511)

DO 30 J=1, 2
ISHLF=ISHL(J)
INTT=SHFT(IRAND, ISHLF)

IF (INTT. GE. 0) GO TO 10

INTT=NOT(INTT)
INTT=SHFT(INTT, ISHRT)
INTT=NOT(INTT)
GO TO 20

INTT=SHFT(INTT, ISHRT)

INTGR(J)=INTT
CONTINUE

SQUARE, SUM, AND TEST UNIT CIRCLE
(511*511=261121)

INTT=INTA*INTA
INTT2=INTB*INTB
IRAND=INTT+INTT2
IF (IRAND. GT. 262121) GO TO 5

IF IREP=-3: COMPUTE SQN(SIN), SQN(COS)

IF (.NOT. (IREP. EQ. -3)) GO TO 32
XI(I)=FLOAT(ISIGN(1, INTA))
XR(I)=FLOAT(ISIGN(1, INTB))
GO TO 40

IF IREP=-2: COMPUTE SIN, COS

CONTINUE
IF (.NOT. (IREP. EQ. -2)) GO TO 33
R=1.0/FLOAT(IRAND)
GO TO 34

IF IREP=-1: COMPUTE X*SIN AND X*COS

CONTINUE
R=XR(I)/FLOAT(IRAND)

34 CONTINUE
 XI(1)=FLOAT(2*INTA*INTB)*R
 XR(1)=FLOAT(INIT-IN TT2)*R
40 CONTINUE
C
 RETURN
 END

A.1.11 Log-normal Background Generator (LGNORM and SETUPL)

Calling Statement: Call LGNORM (XR, XI, N, IREP)

Parameters Read from Input File:

ISEEDL = starting seed for pseudo-random number generator

SDB = log-std.-deviation of clutter power in dB.

Argument List Definition:

Input Variables: N = number of complex samples
generated by each call

IREP = representation code

Returned Variables: XR and XI are defined according to
IREP, as discussed below.

Other Modules Called: SETUPL, RANAMP, RANPHS

Description:

For IREP = -1, LGNORM generates N uncorrelated complex samples of log-normal video, with unit mean power and uniformly distributed phase.

That is:

$$XR = \sqrt{Y} \sin \phi$$

$$XI = \sqrt{Y} \cos \phi$$

where Y is log-normally distributed with probability density

$$p(Y)dY = \frac{1}{S_y Y \sqrt{2\pi}} \exp \left\{ -\frac{(\ln Y + \frac{S_y^2}{2})^2}{2S_y^2} \right\} dY, Y \geq 0$$

where

$$S_y = \text{log.-std.-dev. of } Y$$

$$= \sqrt{E[\ln Y - \overline{\ln Y}]^2}$$

$$= \text{SDB}/4.3429$$

$$E(Y) = E(XR^2 + XI^2) = 1.0$$

$$Y_{50} = \text{median of } Y = \exp \left(-\frac{S_y^2}{2} \right)$$

and where

$\phi = \tan^{-1} (XR, XI)$ is uniform on $(-\pi, \pi)$.

The method of generation of the complex samples is identical to that for NOISE. An inverse probability table is constructed for $\sqrt{Y}$, Y , or $\ln\sqrt{Y}$, according to IREP, with a call to SETUPL. A sample is drawn from this table using subroutine RANAMP. For negative IREP, this amplitude sample is then converted to in-phase and quadrature components by subroutine RANPHS, which generates sine and cosine of a uniformly distributed random phase.

```

SUBROUTINE LGNORM(XR,XI,N,IREP)
LOG-NORMAL CLUTTER GENERATOR

0:   RETURNS NATURAL LOGARITHM OF AMPLITUDES
      IF IREP WERE 1 (SEE BELOW).

1:   RETURNS N UNCORRELATED LOG-NORMAL AMPLITUDES
      IN XR, WITH UNIT MEAN POWER, AND LOG.-STD.-DEV.
      SDB(POWER UNITS).

2:   RETURNS N UNCORRELATED LOG-NORMAL POWERS IN XR.

-1:  RETURNS N UNCORRELATED LOG-NORMAL CLUTTER SAMPLES
      WITH RANDOM PHASE, UNIT MEAN POWER, AND
      LOG.-STD.-DEV. SDB. RETURNS NSAMPLES IN XR,
      AND N QUADRATURE SAMPLES IN XI. USES 3-LEVEL
      TABLE LOOK-UP FOR AMPLITUDES, UNIT CIRCLE ALGORITHM
      FOR SIN AND COS.

-2:  RETURNS N SIN & COS PAIRS ONLY, IN XR & XI

-3:  RETURNS N SGN(SIN) & SGN(COS) PAIRS ONLY.

*INSERT JIMB ANITA>SURADS>IOLUS
*INSERT JIMB ANITA>SURADS>FLAGS
      INTEGER*4 ISEEDL, ISEED
      DIMENSION XR(1), XI(1), PIV(384)
      DATA PIV, LPIV, NPIV, ISEED/384*1.0, 3, 7, 298809973/

C
C   ON FIRST CALL READ PARAMETERS AND INITIALIZE PIV TABLE
C   IF(.NOT.(IFLAGL.EQ.0)) GO TO 25
C
C   WRITE(LUOUT,100)
100  FORMAT(////1X,'LOG-NORMAL CLUTTER GENERATOR')
      READ(LUIN,*) ISEEDL
      IF(ISEEDL.EQ.0) ISEEDL=ISEED
      WRITE(LUOUT,200) ISEEDL
200  FORMAT(//6X,'PRN GENERATOR SEED = ',I12)
      READ(LUIN,*) SDB
      WRITE(LUOUT,300) SDB
300  FORMAT(//6X,'LOG.-STD.-DEV. = ',F6.1,' DB')
      WRITE(LUOUT,400) IREP
400  FORMAT(//6X,'REPRESENTATION CODE = ',I3)
C
      CALL SETUPL(PIV,NPIV,LPIV,IREP,SDB)
      IFLAGL=1

C
25  CONTINUE
      IF(IREP.LT.-1) GO TO 28
C
C   GENERATE RANDOM AMPLITUDES (OR POWERS)
C   CALL RANAMP(PIV,NPIV,LPIV,ISEEDL,XR,N)
C
C   IF IREP<0 GENERATE AND APPLY RANDOM PHASE
C
28  CONTINUE
      IF(IREP.GE.0) GO TO 30
      CALL RANPHS(XR,XI,N,ISEEDL,IREP)
C
30  RETURN
      END

```

```

SUBROUTINE SETUPL(PIV,NPIV,LPIV,IREF,SDB)
C
C
C      SETUP INVERSE PROBABILITY TABLE FOR LOG-NORMAL GENERATOR
      DOUBLE PRECISION PROB
*INSERT JIMB ANITA>SURADS>IOLUS
      DIMENSION PIV(1)
C
      EXPO=1.0
      IF(IREF.EQ.2) EXPO=2.0
      SZ=SDB/8.6858
      NN=2**NPIV
      JLP=0
      DO 20 LP=1,LPIV
        DO 10 J=1,NN
          JLP=JLP+1
          PROB=1.DO+DBLE(FLOAT(J-NN)-0.5)/(DBLE(FLOAT(NN))*LP)
          PIV(JLP)=EXP(EXPO*SZ*(GAUSIV(PROB)-SZ))
          IF(IREF.EQ.0) PIV(JLP)=ALOG(PIV(JLP))
10      CONTINUE
20      CONTINUE
C
      CALL WRTPIV(PIV,NPIV,LPIV,LUDOUT)
C
      RETURN
      END

```

A.1.12 Inverse Cumulative Gaussian Function (GAUSIV)

Calling Statement: X = GAUSIV(P)

Argument List Definition:

Input Variables:	P = probability level (double precision)
Output Variables:	GAUSIV = amplitude for probability level P

Description:

This function is a rational approximation for the inverse cumulative probability distribution function for zero-mean, unit variance Gaussian random variables. It is based on equation 26.2.23 of [7]. The argument P must be double precision, but the return value is single precision.

FUNCTION GAUSIV(Z)

INVERSE PDF FOR ZERO-MEAN, UNIT VARIANCE GAUSSIAN R. V.

DOUBLE PRECISION Z, X, Y, C, D, TEMP

DIMENSION C(3), D(3)

DATA C/2.515517,.802853,.010328/

DATA D/1.432788,.189269,.001308/

X=Z

IF(Z.GT.0.5D0) X=1.D0-Z

Y=DSQRT(DLOG(1.D0/(X*X)))

TEMP=C(1)+Y*(C(2)+Y*C(3))

TEMP=TEMP/(1.D0+Y*(D(1)+Y*(D(2)+Y*D(3))))

GAUSIV=SNGL(TEMP-Y)

IF(Z.GT.0.5D0) GAUSIV=-GAUSIV

RETURN

END

A.1.13 Weibull Background Generator (WEIBUL and SETUPW)

Calling Statement: Call WEIBUL (XR, XI, N, IREP)

Parameters Read from Input File:

ISEEDC = starting seed for pseudo-random number generator

C = exponent parameter of Weibull clutter power
distribution

Argument List Definition:

Input Variables:

N = number of complex samples
generated by each call

IREP = representation code

Returned Variables:

XR and XI are defined according
to IREP, as discussed below.

Other Modules Called: SETUPW, RANAMP, RANPHS

Description:

For IREP = -1, WEIBUL generates N uncorrelated complex samples
of Weibull video, with unit mean power and uniformly distributed phase,
That is

$$XR = \sqrt{Y} \sin \phi$$

$$XI = \sqrt{Y} \cos \phi$$

where Y has Weibull distribution with probability density

$$p(Y)dy = \frac{C}{Y_0} \left(\frac{Y}{Y_0}\right)^{C-1} \exp \left\{-\left(\frac{Y}{Y_0}\right)^C\right\} dy, \quad Y \geq 0$$

where

C = exponent parameter

$$Y_0 = 1.0/\Gamma(1 + 1/C)$$

$$E(Y) = E(XR^2 + XI^2) = Y_0 \Gamma(1 + \frac{1}{C}) = 1.0$$

and where

$\phi = \tan^{-1} (XR, XI)$ is uniform on $(-\pi, \pi)$.

The method of generation is identical to that used in the log-normal generator LGNORM, except that Y has a Weibull distribution. The inverse probability table is generated with a call to SETUPW.


```

C      SUBROUTINE SETUPW(PIV,NPIV,LPIV,IREP,C)
C      SET UP INVERSE PROBABILITY TABLE FOR WEIBULL GENERATOR.
C      DOUBLE PRECISION PROB
.. *INSERT JIMB ANITA>SURADS> IOLUS
C      DIMENSION PIV(1)
C      CINV=1.0/C
C      CONST=1.0+CINV
C      CONST=1./GAMMA(CONST)
C      IF(IREP.EQ.2) GO TO 5
C      CINV=CINV*.5
C      CONST=SQRT(CONST)
5      CONTINUE
C      NN=2*NPIV
C      JLP=0
C      DO 20 LP=1,LPIV
C      DO 10 J=1,NN
C      JLP=JLP+1
C      PROB=1.DO+DBLE(FLOAT(J-NN)-0.5)/(DBLE(FLOAT(NN))*LP)
C      PIV(JLP)=(SNGL(-DLOG(1.DO-PROB))*CINV)*CONST
C      IF(IREP.EQ.0) PIV(JLP)=ALOG(PIV(JLP))
10      CONTINUE
20      CONTINUE
C      CALL WRTPIV(PIV,NPIV,LPIV,LUOUT)
C      RETURN
C      END

```

A.1.14 Gamma Function (GAMMA)

Calling Statement: Y = GAMMA(X)

Description:

GAMMA computes the gamma function, using the fact that

$$\Gamma(x + 1) = x\Gamma(x)$$

and a polynomial approximation for $1.0 < x < 2.0$ from equation 6.1.35 of [7].

```

C      FUNCTION GAMMA(Y)
C      GAMMA FUNCTION POLYNOMIAL APPROXIMATION
C      REF.: ABRAMOWITZ & STEGUN P. 257, EQ. 6.1.35
C      DIMENSION A(5)
C      DATA A/-.5748646, .9512363, -.6998588, .4245549, -.1010678/
C      X=Y
C      GAMMA=1.0
10     CONTINUE
C      IF(X.LE.2.0) GO TO 20
C      X>2.0
C      X=X-1.0
C      GAMMA=GAMMA*X
C      GO TO 10
20     CONTINUE
C      IF(X.GE.1.0) GO TO 30
C      X<1.0
C      GAMMA=GAMMA/X
C      X=X+1.0
C      GO TO 20
30     CONTINUE
C      1.0<=X<=2.0
C      X=X-1.0
C      GAMMA=GAMMA*(1.0+(A(1)+(A(2)+(A(3)+(A(4)+A(5)*X)*X)*X)*X)*X)
C      RETURN
C      END

```

A.1.15 Correlated Background Generator (CORBKG)

Calling Statement: Call CORBKG (XR, XI, XMOD, N, K, IREP, ICMBIN)

Parameters Read from Input File:

ITYPE = type of fluctuations for average cross-section

ITYPE = 2 = log-normal

ITYPE = 3 = Weibull

Argument List Definition:

XR, XI, XMOD = are components of background samples returned by CORBKG, defined according to IREP and ICMBIN, as discussed below.

N = number of range cells

K = number of pulses per range cell

IREP = representation code

ICMBIN = combination code to determine whether components will be combined (see discussion below).

Other Modules Called: NOISE, LGNORM, WEIBUL

Discussion:

CORBKG generates log-normal or Weibull background samples which are Rayleigh modulated pulse-to-pulse. Samples are generated on each of N range bins, and uncorrelated modulations are generated for each bin for K pulses. The sample are generated with a call to either LGNORM or WEIBUL. The Rayleigh fluctuations are generated by NOISE.

Sample representations are determined by IREP, just as they are for the NOISE, LGNORM, and WEIBUL generators. Each sample is the product of two components, the log-normal or Weibull average cross-section, and the Rayleigh fluctuation about that average cross-section. If ICMBIN = 0, the components are returned separately. If ICMBIN = 1, they are combined prior to return.

A.1.16 Radar IF Generator (IFGEN)

Calling Statement: Call IFGEN (YR, YI, XR, XI, NXY, MXY, W, NW)

Argument List Definitions:

Input Variables:

XR, XI = matrix of input signals

MXY, NXY = row and column dimensions
of XR, XI, YR, and YI

W = vector of waveform sample

NW = length of W

Output Variables:

YR, YI = matrix of output signals

Description:

IFGEN computes the convolution of the rows of the complex signal X with the real function W. It is assumed that all samples of W are either +1 or -1. There is no wrap-around on the ends. Thus, only the first NXY - NW + 1 columns of each row of Y are computed. The output YR is equivalent to:

$$YR(IY+(JY-1)*NXY) = \sum_{IW=1}^{NW} W(IW)*IR(IY+NW-IW+(JY-1)*NXY)$$

for

IY = 1, 2, ..., NXY - NW + 1

JY = 1, 2, ..., MXY

YI is similarly defined in terms of XI.

This subroutine is used to generate the radar IF signal (actually bipolar video) by convolving the transmit waveform W with the distributed clutter samples XR, XI.

SUBROUTINE IFOEN (YR,YI,XR,XI,NXY,MXY,W,NW)

COMPUTES CONVOLUTION OF COMPLEX SIGNAL X WITH REAL
FUNCTION W, WHERE EACH SAMPLE OF W IS +/- 1.0.
RESULT IS RETURNED IN Y.

SHAPE OF X IS MXY ROWS BY NXY COLUMNS, STORED
AS A ONE DIMENSIONAL ARRAY, ROW BY ROW.
SHAPE OF Y IS THE SAME AS X.

THE CONVOLUTION IS WITHIN EACH ROW OF X. FOR THE
FIRST ROW, THE RESULTS ARE EQUIVALENT TO:

$$Y(IY) = \sum_{IW=1}^{NW} W(IW) * X(IY + NW - IW)$$

FOR IY=1,NY
WHERE NY=NX-NW+1

THUS, ONLY THE FIRST NY COLUMNS OF EACH ROW OF
Y CONTAIN RESULTS.

CAUTION: X&Y MUST BE DISTINCT ARRAYS IN THE CALLING
PROGRAM

DIMENSION YR(1), YI(1), XR(1), XI(1), W(1)

NY=NXY-NW+1

IK=1

DO 20 K=1,MXY

DO 10 I=1,NXY

YR(IK)=0.0

YI(IK)=0.0

IK=IK+1

CONTINUE

CONTINUE

DO 80 IW=1,NW

KOFFST=-NXY

IF (.NOT. (W(IW).GT.0.0)) GO TO 50

DO 40 K=1,MXY

KOFFST=KOFFST+NXY

IX=NW+1-IW+KOFFST

DO 30 I=1,NY

IY=KOFFST+I

YR(IY)=YR(IY)+XR(IX)

YI(IY)=YI(IY)+XI(IX)

IX=IX+1

CONTINUE

CONTINUE

CONTINUE

IF (.NOT. (W(IW).LT.0.0)) GO TO 80

DO 70 K=1,MXY

KOFFST=KOFFST+NXY

IX=NW+1-IW+KOFFST

DO 60 I=1,NY

IY=KOFFST+I

YR(IY)=YR(IY)-XR(IX)

YI(IY)=YI(IY)-XI(IX)

IX=IX+1

CONTINUE

CONTINUE

CONTINUE

RETURN

END

A.1.17 Target Generator (TARGET)

Calling Statement: Call TARGET (X, N, K)

Parameters Read from Input File:

ISEEDT = starting seed for pseudo-random number generator

ISWRL = Swerling Case number (0,1,2,3,4)

Argument List Definitions:

Input Variables:

N - number of independent target
samples per pulse

K = number of pulses

Returned Variables:

X = vector of N x K target samples

Other Modules Called: CHISQ, WRTPIV, RANAMP

Description:

TARGET generates N x K target amplitude samples of unit mean power. N is the number of target samples per pulse, and K is the number of pulses. Target distributions correspond to Swerling cases 0-4. For case 0, X contains N x K ones. For cases 2 and 4, X contains N x K uncorrelated samples. For cases 1 and 3, X contains N uncorrelated samples, repeated K times in blocks of N samples.

The target amplitudes are given by

$$X = \sqrt{Y}$$

where Y is a unit mean random variable whose probability density function is given by

$$P_Y(Y) = \begin{cases} \delta(Y-1.0)dy, & \text{where } \delta(Y) \text{ is the delta} \\ & \text{function, for Swerling} \\ & \text{case 0,} \\ \exp(-Y)dy, & \text{Swerling case 1 \& 2, } y \geq 0 \\ 4Y\exp(-2Y)dy, & \text{Swerling cases 3 \& 4, } y \geq 0 \end{cases}$$

The method of generation is by inverse probability table construction and table look-up.

F/6 17/9

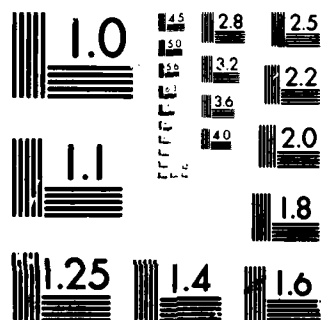
N60921-79-C-A306

NL

2.2

والله اعلم
بما فيه التوفيق

END
DATE
FILMED
5-80
RUC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

A.1.17 Target Generator (TARGET)

Calling Statement: Call TARGET (X, N, K)

Parameters Read from Input File:

ISEEDT = starting seed for pseudo-random number generator

ISWRL = Swerling Case number (0,1,2,3,4)

Argument List Definitions:

Input Variables:

N - number of independent target samples per pulse

K = number of pulses

Returned Variables:

X = vector of N x K target samples

Other Modules Called: CHISQ, WRTPIV, RANAMP

Description:

TARGET generates N x K target amplitude samples of unit mean power. N is the number of target samples per pulse, and K is the number of pulses. Target distributions correspond to Swerling cases 0-4. For case 0, X contains N x K ones. For cases 2 and 4, X contains N x K uncorrelated samples. For cases 1 and 3, X contains N uncorrelated samples, repeated K times in blocks of N samples.

The target amplitudes are given by

$$X = \sqrt{Y}$$

where Y is a unit mean random variable whose probability density function is given by

$$P_Y(Y) = \begin{cases} \delta(Y-1.0)dy, & \text{where } \delta(Y) \text{ is the delta function, for Swerling case 0,} \\ \exp(-Y)dy, & \text{Swerling case 1 \& 2, } y \geq 0 \\ 4Y\exp(-2Y)dy, & \text{Swerling cases 3 \& 4, } y \geq 0 \end{cases}$$

The method of generation is by inverse probability table construction and table look-up.

A.1.18 Inverse Cumulative Chi-Squared Function (CHISQ)

Calling Statement: Y = CHISQ(P, DP, M, YOLD)

Argument List Definition:

Input Variables: P = probability level (double precision)
DP = probability tolerance desired
(double precision)
M = number of duo-degrees of freedom
(i.e. number of degrees of
freedom = 2 x M)
YOLD = previous value returned by CHISQ
(double precision; initially
zero)

Returned Variables: CHISQ = amplitude for probability
level P (single precision)
YOLD = double precision replica
of CHISQ.

Other Modules Called: GAUSIV

Description:

CHISQ returns the inverse cumulative probability distribution function for a Chi-squared random variable with M duo-degrees of freedom, and max value M. For a given probability level P, the P-level amplitude is obtained by inverting the cumulative distribution function given by.

$$P(Y) = 1 - \exp(-Y) \sum_{k=0}^{M-1} \frac{Y^k}{k!}, M \geq 1$$

For M = 0, a value of CHISQ = 1.0 is always returned.

The inversion of P(Y) is obtained in the following manner.

For M = 1,

$$Y(P) = -\ln(1 - P).$$

For $1 < M \leq 50$, $Y(P)$ is found by making an initial guess Y_p , and computing $P(Y_p)$. If $P(Y_p)$ is within DP of P , the inversion is done, and

$$CHISQ = Y_p$$

is returned. If not, then a new guess is obtained as

$$Y_p' = Y_p + \frac{P - P(Y_p)}{\frac{\partial P(Y_p)}{\partial Y}}$$

and the process repeated until $P(Y_p')$ is close enough to P . The initial guess for Y_p is found by either of two techniques. On the first call to CHISQ (signified by $YOLD = 0$), Y_p is obtained by scaling the $M = 1$ root using a Gaussian approximation:

$$\frac{Y_p - M}{\sqrt{M}} = \frac{-\ln(1 - P) - 1}{\sqrt{1}}$$

On subsequent calls, the previous root is used:

$$Y_p = YOLD$$

For $M > 50$, the Chi-Square distribution is approximated as Gaussian, and the inversion is obtained by calling GAUSIV.

```

C      REAL FUNCTION CHISQ (PROB, DELPRB, M, XOLD)
C      INVERSE PDF FOR CHI-SQUARE
C      DEGREE 2*M, MEAN=M, RANDOM VARIABLES.
C      RETURNS CHISQ=1.0 FOR M=0.
C      USES GAUSSIAN APPROX. FOR M>50.
C
C      DOUBLE PRECISION PROB, DELPRB, XOLD, X, XM, PX, EXPO
C      DOUBLE PRECISION XTOK, SUM
C
C      CHISQ=1.0
C      IF (M.EQ.0) RETURN
C
C      IF (M.GE.51) GO TO 200
C      X = -DLOG(1.DO - PROB)
C
C      FOR M=1, THIS IS AN EXACT SOLUTION
C
C      IF (M.EQ.1) GO TO 100
C      IF (.NOT.(XOLD.EQ.0.0)) GO TO 5
C
C      SCALE M-1 ROOT ON GAUSSIAN APPROXIMATION
C
C      XM = DBLE(FLOAT(M))
C      X = XM - DSGRT(XM)*(1.DO - X)
C      GO TO 10
C
C      START AT PREVIOUS ROOT LOCATION
C
C      5      X = XOLD
C      10      CONTINUE
C      XTOK = 1.DO
C      SUM = 0.0
C      DO 30 I = 1, M
C          K = I - 1
C          IF (K.EQ.0) GO TO 20
C          XTOK = XTOK * X/DBLE(FLOAT(K))
C      20      SUM = SUM + XTOK
C      30      CONTINUE
C      EXPO = DEXP(-X)
C      PX = 1.DO - EXPO * SUM
C      IF (DABS(PROB - PX) .LE. DELPRB) GO TO 100
C      X = X + (PROB - PX)/(EXPO * XTOK)
C      GO TO 10
C
C      100      CONTINUE
C      CHISQ = SNGL(X)
C      XOLD = X
C      RETURN
C
C      200      CONTINUE
C      IF (PROB .GT. .5DO) GO TO 210
C      Y = GAUSIV(PROB)
C      CHISQ = FLOAT(M) + Y*SGRT(FLOAT(M))
C      RETURN
C
C      210      CONTINUE
C      Y = GAUSIV(1.DO - PROB)
C      CHISQ = FLOAT(M) - Y*SGRT(FLOAT(M))
C      IF (CHISQ .LT. 0.0) CHISQ = 0.0
C
C      RETURN
C
C      END

```

A.1.19 Threshold Crossing Accumulation (PDDFX)

Calling Statement: Call PDDFX(M)

Argument List Definition:

M = number of samples in waveform WR (in labelled common ARRAYS)

Description:

This is the module which actually implements the receiver signal processing. The IF background signals are presumed to be in array VUR and VUI. The target amplitudes are in the array TGT. The waveform used in generating VUR and VUI is in the array WR. It is assumed that WR is either +1 or -1.

PDDFX generates a target waveform and adds it to the background IF signal at the appropriate signal-to-background ratio. The in-phase and quadrature sum signal are then hard-limited to +/-1 and pulse compressed using the waveform WR. A test statistic (TEST) is formed by summing the pulse-compressed $I^2 + Q^2$ over NPDI pulses. The test statistic is compared to the array of thresholds, and the threshold crossing counts accumulated in the matrix NTC. The test statistic and threshold comparison is repeated on each range bin. The process is then repeated for all other signal-to-background ratios SBR. On return, the matrix NTC contains the cumulative count of threshold crossings as a function of threshold and signal-to-background ratio.

CCCCCCCC

SUBROUTINE PDDFX(M)

COMPUTE DETECTION PROBABILITY FOR
DICKE-FIX CFAR (HARD-LIMITED I & Q).

ADD TARGET PLUS BACKGROUND, PERFORM HARDLIMITING
AND PULSE COMPRESSION, SUM PULSES INCOHERENTLY,
AND TEST AGAINST THRESHOLDS.

```

*INSERT JIMB JODI>SURADS>CNTROL
*INSERT JIMB JODI>SURADS>IOLUS
*INSERT JIMB JODI>SURADS>ARRAYS
MM1T2=(M-1)*2
NRM2=NRBIN+MM1T2
DO 20 J=1,NSBR
  SB=SBR(J)
  ISBR=(J-1)+NTHR
  DO 15 K=1,NRBIN
    TEST=0.0
    KL=K-NRM2
    KLT=K-NRBIN
    DO 5 L=1,NPDI
      KL=KL+NRM2
      KLT=KLT+NRBIN
      KLM=KL
      TESTR=0.0
      TESTI=0.0
      TARG=SB*TGT(KLT)
      DO 4 MM=1,M
        IF (.NOT. (WR(MM).GT.0.0)) GO TO 31
        TESTR=TESTR+SIGN(1.0,(VUR(KLM)+TARG))
        TESTI=TESTI+SIGN(1.0,VUI(KLM))
        GO TO 41
      31 CONTINUE
        TESTR=TESTR-SIGN(1.0,(VUR(KLM)-TARG))
        TESTI=TESTI-SIGN(1.0,VUI(KLM))
      41 CONTINUE
        KLM=KLM+1
      4 CONTINUE
        TESTI=TESTR*TESTR+TESTI*TESTI
        TEST=TEST+TESTI
      5 CONTINUE
        INDX=ISBR
        DO 10 IT=1,NTHR
          INDX=INDX+1
          IF (TEST.GT.THR(IT)) NTC(INDX)=NTC(INDX)+1
        10 CONTINUE
      15 CONTINUE
    20 CONTINUE
  RETURN
END

```

...

•

•

```

C      SUBROUTINE SUMARY(NTC, SBR, THR, NSBR, NTHR, NTRIAL)
C      SUMMARIZES THRESHOLD CROSSING COUNTS
C      RECORDED IN NTC.
C      DOUBLE PRECISION XTRIAL, PROB
C      INTEGER*4 NTRIAL, NTC
C      *INSERT IOLUS
C      DIMENSION NTC(1), PROB(20), SBR(1), THR(1)
C      DATA NCOL/4/
C      WRITE(LUOUT, 100)
100    FORMAT(///1X, 'SUMMARY OF THRESHOLD CROSSING PROBABILITIES(PCT. )')
200    WRITE(LUOUT, 200) NTRIAL
200    FORMAT(/6X, 'NO. OF TRIALS = ', I7)
C      XTRIAL=100. DO/DBLE(FLOAT(NTRIAL))
C      NLOOP=NTHR/NCOL
C      NNLOOP=NCOL*NLOOP
C      IF(NNLOOP.LT. NTHR) NLOOP=NLOOP+1
C      J1=1
C      J2=J1+NCOL-1
C      DO 30 I=1, NLOOP
C          J2=MINO(J2, NTHR)
300    WRITE(LUOUT, 300) (THR(J), J=J1, J2)
300    FORMAT(///6X, 'THRESHOLD CONSTANT      ', 10G11.3/)
400    WRITE(LUOUT, 400)
400    FORMAT(/6X, 'S/B RATIO(DB) ')
C      DO 20 K=1, NSBR
C          DBSBR=SBR(K)
C          ISBR=(K-1)*NTHR
C          DO 10 J=J1, J2
C              INDX=ISBR+J
C              PROB(J)=DBLE(FLOAT(NTC(INDX)))*XTRIAL
10      CONTINUE
C          WRITE(LUOUT, 500) DBSBR, (PROB(J), J=J1, J2)
500    FORMAT(/6X, F10.1, 11X, 5F11.5)
20      CONTINUE
C          J1=J1+NCOL
C          J2=J2+NCOL
30      CONTINUE
C      RETURN
C      END

```

A.1.21 Edit, Fit, and Plot Data (PLOTDP)

Calling Statement: Call PLOTDP (NTC, SBR, NSBR, NTHR, NTRIAL)

Parameters Read from Input:

IFNAM = plot file name

PFAPCT = vector of threshold crossing probabilities for no signal.

Argument List Definition:

NTC = matrix of threshold crossing counts

SBR = vector of signal-to-background ratios in dB.

NSBR = length of SBR

NTHR = number of thresholds summarized in NTC

NTRIAL = number of trials

Other Modules Called: PDPLOT, SEARCH

Description:

PLOTDP First opens a plotfile for writing plottable curves of P_D vs. SBR. For each threshold level, PLOTDP then extracts the appropriate threshold crossing counts from NTC, edits them, and converts to percent probability of threshold crossing. These probabilities are then distorted using the value of PFAPCT. The distorted values are passed to PDPLOT for curve fitting and plot file writing.

The data editing rejects crossing counts of fewer than 10 events, or greater than 10 less than the total number of trials. In addition, it rejects crossing probabilities less than twice the false alarm probability. Finally, it rejects probabilities which are less than a probability obtained at a lower signal-to-background ratio, i.e. it assures that the probabilities passed to PDPLOT will be monotonically increasing with signal-to-background.

The probability distortions are to enable curve fit parameters to be obtained using the approximate procedure discussed in ref [2]. The distortion is:

$$P_D' = 100. \left(\frac{P_D - P_{FA}}{100 - P_{FA}} \right)$$

where all probabilities are in percent.

A.1.22 Generate Plottable Files (PDPLLOT)

Calling Statement: Call PDPLLOT (PDPCT, SDB, NPTS, PFAPCT, IWRIT)

Argument List Definitions:

PDPCT = distorted threshold crossing probabilities in percent

SDB = corresponding signal-to-background ratios in dB

NPTS = length of PDPCT and SDB

PFAPCT = threshold crossing probability for no signal

IWRIT = Fortran logical unit for writing plottable files.

Other Modules Called: QPDFIT, SBRVPD, PDVSBR

Description:

PDPLLOT generates plottable files of P_D vs signal-to-background data and parametric fit curves. The first step is the determination of the parameter values which fit the distorted P_D data. This is done by a call to QPDFIT. Next, these parameter values are used to determine the signal-to-background ratio required for distorted probabilities of 1% and 99%. This is done by a call to SBRVPD. A smooth curve is then generated by a call to PDVSBR, which returns smooth fit distorted probabilities at each of 25 signal-to-background ratios spaced evenly in dB between the 1% and 99% values. Finally, the distortion is removed from the P_D data and curve fits, and the data and smooth curve pairs written to a file for off-line plotting.

```

C      SUBROUTINE PDPLLOT (PDPCT,SDB,NPTS,PFAPCT,IWRIT)
C
C      GENERATE PLOTABLE FILES OF
C      PD VS. SBR DATA AND PARAMETRIC FIT
C
C      $INSERT JIMB JODI>SURADS>IOLUS
C      DIMENSION PDPCT(1),SDB(1),PDFIT(100),SDBFIT(100),
C      *          P(3),IFNAM(3),PD199(2)
C      DATA IOUT,XEOF,NFIT/10,1.E30,25/
C      DATA PD199/1.,99./
C
C      IF(NPTS.LT.2) GO TO 40
C
C      FIT THE DATA
C
C      CALL QPDFIT(PDPCT,SDB,NPTS,P)
C
C      IWRIT=IOUT+4
C      WRITE (LUOUT,115) (P(I),I=1,3),PFAPCT
115    FORMAT(/21X,4010.3)
C
C      FIND 1% AND 99% SBR)
C
C      CALL SBRVPD(PD199,SDBFIT,2,P)
C
C      GENERATE SMOOTH FIT
C
C      CALL PDVSBP(PDFIT,SDBFIT,-NFIT,P)
C
C      REMOVE PFA DISTORTION FROM DATA AND FIT
C
C      DO 10 I=1,NPTS
C          PDPCT(I)=PDPCT(I)*((100.-PFAPCT)/100.)+PFAPCT
10    CONTINUE
C      DO 20 I=1,NFIT
C          PDFIT(I)=PDFIT(I)*((100.-PFAPCT)/100.)+PFAPCT
20    CONTINUE
C
C      WRITE DATA AND FIT
C
C      WRITE(IWRIT,120)((SDB(I),PDPCT(I)),I=1,NPTS)
C      WRITE(IWRIT,120)(XEOF,XEOF)
C      WRITE(IWRIT,120)((SDBFIT(I),PDFIT(I)),I=1,NFIT)
120    WRITE(IWRIT,120) XEOF,XEOF
C      FORMAT(2E12.4)
C
C      40    RETURN
C      END

```

A.1.23 Parametric Curve Fit (QPDFIT)

Calling Statement: Call QPDFIT (PD, SDB, N, P)

Argument List Definitions:

PD = detection probability in percent

SDB = signal-to-background ratio in dB

N = length of PD and SDB

P = vector of 3 coefficients of parametric fit determined by QPDFIT.

Description:

QPDFIT computes approximate values for parameters P_1 , P_2 , and P_3 in the expression

$$P_D = 100 \times \left\{ 1 - [1 + (P_1 S)^{P_2}]^{-P_3} \right\}$$

where P_D is detection probability in percent and S = signal-to-background ratio = $\log_{10} \left(\frac{SDB}{10} \right)$. The method of calculation is discussed in Reference [2].


```

C      DO 250 I=1,N
        W(I)=ALOG10(((1. -. 01*PD(I))*(-1./P(3)))-1.)
        SUM=SUM+1.0
        SUMX=SUMX+SDB(I)
        SUMY=SUMY+W(I)
        SUMX2=SUMX2+SDB(I)*SDB(I)
        SUMXY=SUMXY+SDB(I)*W(I)
C 250  CONTINUE
C      D=SUM*SUMX2-SUMX*SUMX
        B=(SUMX2*SUMY-SUMX*SUMXY)/D
        A=(SUMXY*SUM-SUMX*SUMY)/D
C      COMPUTE K1 AND K2
C      P(1)=10.**(B/(10.*A))
        P(2)=10.*A
C      RETURN
      END

```

A.1.24 SBR vs. PD (SBRVPD)

Calling Statement: Call SBRVPD (PDPCT, SDB, N, P)

Argument List Definition:

PDPCT = detection probability in percent

SDB = signal-to-background in dB

N = length of PDPCT and SDB

P = vector of coefficients

Description:

Computes

$$SDB = 10 \log_{10} (S)$$

where

$$S = \left(\frac{1}{P_1}\right) \left[\left((1-P_D)^{-1/P_3} \right) - 1 \right]^{1/P_2}$$

$$P_D = .01 \times PDPCT$$

$$P_1, P_2, P_3 = P(1), P(2), P(3)$$

SDB is evaluated at each of N values of PDPCT, if N is positive. If N is negative, SDB is evaluated at each of |N| points uniformly spaced between PDPCT (1) and PDPCT (2).

A.1.25 PD vs. SBR (PDVSBP)

Calling Statement: Call PDVSBP (PDPCT, SDB, N, P)

Argument List Definition:

Same as for SBRVPD

Description:

Computes

$$PDPCT = 100. \times \left\{ 1 - \left[1 + (P_1 S)^{P_2} \right]^{-P_3} \right\}$$

where

$$S = 10^{(SDB/10)}$$

$$P_1, P_2, P_3 = P(1), P(2), P(3)$$

PDPCT is evaluated at each of N values of SDB, if N is positive. If N is negative, PDPCT is evaluated at each of |N| points uniformly spaced between SDB(1) and SDB(2).

CCCCC

C

10

20

```

SUBROUTINE PDVSR(PDPCT, SDB, N, P)
COMPUTES:
  PDPCT=100. *(1. -(1. +(P1*S)**P2)**(-P3))
WHERE:
  S=10. **SDB/10.
  P1, P2, P3=P(1), P(2), P(3)
DIMENSION P(3), PDPCT(1), SDB(1)
SP(X)=10. *(X/10.)
PD(S)=1. -(1. +(P(1)*S)**P(2))**(-P(3))
NI=IABS(N)
IF(.NOT. (N.LT. 0)) GO TO 10
SDBSTP=(SDB(2)-SDB(1))/FLOAT(NI-1)
SDBLOC=SDB(1)-SDBSTP
CONTINUE
DO 20 I=1, NI
  IF(N.GT. 0) SDBLOC=SDB(I)
  IF(N.LT. 0) SDBLOC=SDBLOC+SDBSTP
  SDB(I)=SDBLOC
  PDPCT(I)=100. *PD(SP(SDBLOC))
CONTINUE
RETURN
END

```

A.2

EXAMPLE RUN

An example run of the simulation program is included here illustrating the calculation of P_D vs. SBR in distributed rain clutter (Raleigh modulated log-normal clutter cross-section, 3 dB log-standard deviation), for a single pulse.

Figure A.2 lists all input parameters which must be supplied for the simulation. Figure A.3 lists those particular values used in the example run. Figure A.4 shows the output file CFROUT(LUOUT). The plot file is not shown. Table A.1 tabulates the allowable range of values for all program inputs.

INPUT PARAMETERREAD BY

NTRIAL, NPD1, M, IPLOT, NTRLPS	DETDFX
ITYPE, NTHR, (THR(I), I=1, NTHR), NSBR, (SBR(I), I=1, NSBR)	DETDFX
If ITYPE = 1: ISEEDN	NOISE
If ITYPE = 2: ISEEDL SDB	LGNORM LGNORM
If ITYPE = 3: ISEEDW C	WEIBUL WEIBUL
If ITYPE = 4: ITYPE (local variable in CORBKG) ISEEDN If ITYPE = 2: ISEEDL SDB If ITYPE = 3: ISEEDW C	CORBKG NOISE LGNORM LGNORM WEIBUL WEIBUL TARGET TARGET
ISEEDT	
ISWRL	
If IPLOT = 1: IFNAM(I), I = 1, 3 PFAPCT(I), I = 1, NTHR	PLOTPD PLOTPD

Figure A.2. General List of Inputs Required

```

NTRIAL, NPD1, M, IPLOT, NTRLPS:
200, 1, 127, 1, 1
ITYPE, NTHR, (THR(I), I = 1, NTHR):
4, 6, 32.1, 33.3, 34.2, 35, 35.6, 36.2
NSBR, (SBR(I), I = 1, NSBR):
20, -3, -2, -1, 0, 1, 2, ..., 16
ITYPE(local to CORBKG):
2
ISEEDN:
0
ISEEDL:
0
SDB:
3.0
ISEEDT:
0
ISWRL:
0
IFNAM(I), I = 1, 3
PDRI
PFAPCT(I), I=1, NTHR:
1.14, 0.27, 0.080, 0.022, 0.0073, 0.0021
NTRIAL, NPD1, M, IPLOT, NTRLPS:
0, 0, 0, 0, 0

```

Figure A.3. Example Run Inputs

Figure A.4 OUTPUT FILE (CFROUT)

 01/10/80 0: 0:49 SYSTEM (USER #16)

DETECTION PERFORMANCE OF DICKE-FIX CFAR

NSCANS = 200 NRBINS = 1 IPLOT = 1

BACKGROUND TYPE = 4 NPDI = 1 M = 127

THRESHOLD TEST LEVELS :

32.1	33.3	34.2	35.0
35.6	36.2		

SIGNAL-TO-BACKGROUND RATIOS (DB) :

-3.00	-2.00	-1.00	0.000
1.00	2.00	3.00	4.00
5.00	6.00	7.00	8.00
9.00	10.0	11.0	12.0
13.0	14.0	15.0	16.0

MAXIMAL LENGTH PN SEQUENCE

CODE LENGTH = 127

SHIFT REGISTER LENGTH = 7

FEEDBACK CONNECTIONS = 7 6

WAVEFORM:

1.	1.	1.	1.	1.	1.	1.	-1.	-1.	-1.	-1.	-1.	-1.	1.	-1.	-1.
-1.	-1.	-1.	1.	1.	-1.	-1.	-1.	-1.	1.	-1.	1.	-1.	-1.	-1.	1.
1.	1.	1.	-1.	-1.	1.	-1.	-1.	-1.	1.	-1.	1.	1.	-1.	-1.	1.
1.	1.	-1.	1.	-1.	1.	-1.	-1.	1.	1.	1.	1.	1.	-1.	1.	-1.
-1.	-1.	-1.	1.	1.	1.	-1.	-1.	-1.	1.	-1.	-1.	1.	-1.	-1.	1.
1.	-1.	1.	1.	-1.	1.	-1.	1.	1.	-1.	1.	1.	1.	1.	-1.	1.
1.	-1.	-1.	-1.	1.	1.	-1.	1.	-1.	-1.	1.	-1.	1.	1.	1.	-1.
1.	1.	1.	-1.	-1.	1.	1.	-1.	-1.	1.	-1.	1.	-1.	1.	1.	-1.

Figure A.4 (continued)

CORRELATED BACKGROUND GENERATOR

SPATIAL MODULATION TYPE = 2

GAUSSIAN NOISE GENERATOR

PRN GENERATOR SEED = 1735483429

REPRESENTATION CODE = -1

LOG-NORMAL CLUTTER GENERATOR

PRN GENERATOR SEED = 298809973

LOG. -STD. -DEV. = 3.0 DB

REPRESENTATION CODE = 1

TARGET AMPLITUDE GENERATOR

PRN GENERATOR SEED = 524922231

SWERLING CASE 0 TARGET MODEL

Figure A.4 (continued)

SUMMARY OF THRESHOLD CROSSING PROBABILITIES(PCT.)

NO. OF TRIALS = 200

THRESHOLD CONSTANT S/B RATIO(DB)	32.1	33.3	34.2	35.0
-3.0	4.00000	1.00000	0.00000	0.00000
-2.0	4.00000	1.00000	0.50000	0.00000
-1.0	4.00000	1.00000	0.50000	0.00000
-0.0	4.00000	2.00000	0.50000	0.00000
1.0	5.00000	2.00000	0.50000	0.00000
2.0	5.00000	2.50000	0.50000	0.00000
3.0	5.50000	3.50000	1.50000	0.00000
4.0	8.00000	3.50000	1.50000	0.50000
5.0	11.00000	3.50000	2.00000	0.50000
6.0	12.00000	6.50000	2.50000	1.00000
7.0	18.00000	8.50000	4.50000	1.50000
8.0	27.00000	14.50000	6.00000	3.00000
9.0	39.00000	23.00000	10.50000	6.50000
10.0	55.50000	34.50000	20.00000	10.50000
11.0	69.50000	50.50000	31.50000	18.00000
12.0	84.00000	69.50000	48.00000	31.50000
13.0	93.00000	79.50000	65.50000	49.00000
14.0	97.00000	92.50000	82.00000	67.50000
15.0	99.00000	97.50000	95.00000	85.00000
16.0	99.50000	99.50000	98.00000	96.00000

Figure A.4 (continued)

THRESHOLD CONSTANT	35.6	36.2
S/B RATIO(DB)		
-3.0	0.00000	0.00000
-2.0	0.00000	0.00000
-1.0	0.00000	0.00000
-0.0	0.00000	0.00000
1.0	0.00000	0.00000
2.0	0.00000	0.00000
3.0	0.00000	0.00000
4.0	0.00000	0.00000
5.0	0.00000	0.00000
6.0	0.00000	0.00000
7.0	0.00000	0.00000
8.0	1.50000	0.00000
9.0	2.50000	0.50000
10.0	5.50000	2.00000
11.0	11.00000	6.00000
12.0	22.00000	11.50000
13.0	34.00000	21.00000
14.0	54.50000	41.00000
15.0	75.50000	61.50000
16.0	91.50000	83.00000

PLOT FILE GENERATOR

PLOT FILE PDR1	OPENED			
FIT PARAMETERS:	K1	K2	K3	PFA(PCT)
	0.497E-02	1.73	136.	1.14
	0.150E-01	2.10	22.3	0.270
	0.274E-01	2.54	5.60	0.800E-01
	0.287E-01	2.66	3.26	0.220E-01
	0.294E-01	3.06	2.45	0.730E-02
	0.279E-01	3.33	1.95	0.210E-02

PLOT FILE PDR1 CLOSED

Table A.1. Allowable Ranges of Input Variables

Variable	Type	Allowable Range of Values
NTRIAL	double length integer	$1 \leq \text{NTRIAL} \leq$ maximum positive double integer for machine
NPDI	integer	$\text{NPDI} \geq 1$
M	integer	$2 \leq M \leq 256$ $((M-1)*2 + 1)\text{NPDI} \leq 2048$
IPLOT	integer	any single precision integer value
NTRLPS	integer	$1 \leq \text{NTRLPS} \leq \text{NTRIAL}$
ITYPE	integer	1, 2, 3, 4
NTHR	integer	$1 \leq \text{NTHR} \leq 20$
THR (20)	real	any single precision real value
NSBR	integer	$1 \leq \text{NSBR} \leq 20$
SBR (20)	real	any single precision real value
ISEEDN	double length integer	any non-negative double integer
ISEEDL	double length integer	any non-negative double integer
ISEEDW	double length integer	any non-negative double integer
ISEEDT	double length integer	any non-negative double integer
SDB	real	any single precision real value
C	real	any positive single precision real value
ISWRL	integer	0, 1, 2, 3, 4
IFNAM (3)	integer	any six-character file name (2 char./word)
PFAPCT (20)	real	$0.0 \leq \text{PFAPCT} \leq 100.0$

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Detection Performance Simulation of a Dicke-Type CFAR in Distributed Clutter		5. TYPE OF REPORT & PERIOD COVERED Technical
7. AUTHOR(s) James N. Bucknam		6. PERFORMING ORG. REPORT NUMBER TSC-W7-78
9. PERFORMING ORGANIZATION NAME AND ADDRESS Technology Service Corporation / 8555 16th Street, Suite 300 Silver Spring, Maryland 20910		8. CONTRACT OR GRANT NUMBER(s) N60921-79-C-A306
11. CONTROLLING OFFICE NAME AND ADDRESS Systems Compatibility Branch, Code F-42 Naval Surface Weapons Center Dahlgren, Virginia 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 22 February 1980
		13. NUMBER OF PAGES 117
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution is unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) CFAR, DICKE, RADAR CLUTTER, TARGET DETECTION, RADAR SIMULATION		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Abstract on reverse side.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

A Dicke-type CFAR processor is one which hard limits the radar IF or bipolar video signal prior to pulse compression. The constant false alarm rate (CFAR) properties of this processor in white receiver noise are well known. When the background against which targets must be detected is due to clutter returns, this CFAR property is destroyed.

This report summarizes results of computer simulation of a Dicke-type CFAR in a background consisting of distributed clutter returns. Clutter model parameters are chosen to approximate the statistics of sea, rain, and land clutter. The simulations show the increase in probability of false alarm (P_{FA}) in these backgrounds when the threshold settings are chosen to provide the desired P_{FA} in receiver noise. In addition, the probability of target detection as a function of signal-to-background ratio is determined. Values are determined for the coefficients in a parametric representation of these results.

A detailed documentation of the simulation software is included as an Appendix.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)