ARO 22848.2-EL-S

AD-A172 929

JC-2039-A

THE EFFECTIVENESS OF RANDOM MFSK FREQUENCY-HOPPING ECCM RADIOS AGAINST WORST-CASE PARTIAL-BAND NOISE JAMMING

FINAL REPORT

AUGUST 1986

PREPARED FOR

U. S. ARMY RESEARCH OFFICE

CONTRACT
DAAG29-85-C-0021

J. S. LEE ASSOCIATES, INC.
2001 JEFFERSON DAVIS HIGHWAY, SUITE 601
ARLINGTON, VIRGINIA 22202

JC-2039-A

THE EFFECTIVENESS OF RANDOM MFSK FREQUENCY-HOPPING ECCM RADIOS AGAINST WORST-CASE PARTIAL-BAND NOISE JAMMING

FINAL REPORT

AUGUST 1986

PREPARED FOR

U.S. ARMY RESEARCH OFFICE

CONTRACT

DAAG29-85-C-0021

J. S. LEE ASSOCIATES, INC.
2001 JEFFERSON DAVIS HIGHWAY, SUITE 601
ARLINGTON, VIRGINIA 22202

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARO 22848.2-EL-S	2. GOVT ACCESSION NO. N/A	3. RECIPIENT'S CATALOG NUMBER N/A
4. TITLE (and Subtitle) The Effectiveness of Random MFSK Frequency-Hopping ECCM Radios Against Worst-Case Partial-Band Noise Jamming		5. TYPE OF REPORT & PERIOD COVERED Final Report 1 Jul 1985--31 July 1986
		6. PERFORMING ORG. REPORT NUMBER JC-2039-A
7. AUTHOR(s) L.E. Miller, R.H. French, L.J. Cowles, and J.S. Lee		8. CONTRACT OR GRANT NUMBER(s) DAAG29-85-C-0021
9. PERFORMING ORGANIZATION NAME AND ADDRESS J.S. Lee Associates, Inc. Suite 601, 2001 Jefferson Davis Highway Arlington, Virginia 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS N/A
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE August 1986
		13. NUMBER OF PAGES 528
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Frequency-shift-keying (FSK), Frequency-hopping (FH), Jamming, Probability of error, Diversity, ECCM

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The system analyzed for bit error rate (BER) performance is frequency-hopped (FH) multiple frequency-shift-keying (MFSK) wherein the M symbol frequency assignments are independently and randomly chosen on each hop, and the information transmission is repeated L times (L-hop diversity) for a soft symbol decision. This FH/RMFSK system is intended to counter systematic follow-on jamming. The reported analysis is of the system's BER in worst case partial-band noise jamming (WCPBNJ). To be effective, the receiver must employ nonlinear combining of the L hops; several hop weighting schemes are evaluated with different assumptions about

Block 18 (cont.)

Further supplemental information:

a. Participating scientific personnel: authors (see list in Block 4) plus H.M. Kwon
b. Degrees earned while employed on project: none
c. Publications connected with this work (to date):
L.E. Miller, R.H. French, and J.S. Lee, "Performance of Frequency-Hopped Random MFSK in Follow-on and Partial-band Noise Jamming," to be presented at IEEE GLOBECOM '86 Conference and included in the proceedings of that conference.

Block 20 (cont.)

available real-time information on relative powers of signal, noise, and jamming. These include adaptive gain control, clipping, hard-decision, and self-normalizing (nonparametric) schemes. It is shown that a simple, self-normalizing receiver, using no jamming state information or measurements, can perform nearly as well as one using a priori values of received noise-plus-jamming powers for adaptive gain control. It is also demonstrated that a hard-decision receiver (majority logic decoding of the L repetitions) achieves an ECCM effect and is viable if the SNR is high. Although the BER varies with jammer power in much the same way as for conventional FH/MFSK (given the parameters M, L, and the unjammed SNR), including a diversity gain for high SNR, FH/RMFSK in general is more vulnerable to WCPBNJ for M greater than 2. Therefore, it is concluded that implementation of effective diversity schemes is feasible, and that for a binary system the additional complexity of random hopping can be assessed to the additional protection gained against follow-on jamming.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF TABLES

THE EFFECTIVENESS OF RANDOM MFSK FREQUENCY HOPPING ECCM RADIOS AGAINST WORST-CASE PARTIAL-BAND NOISE JAMMING

1.0 INTRODUCTION

The purpose of this study is to provide the Army a direct comparison of the uncoded bit error rate (BER) performance of several receiver anti-jam processing schemes with varying degrees of implementation complexity, under the same conditions of system noise and jamming. In this manner the engineering cost of complex anti-jam receiver designs can be weighed against their effectiveness, as illustrated in Figure 1.0-1. In what follows we discuss the issues surrounding the work and summarize our effort.

1.1 BACKGROUND

In the Electronic Warfare (EW) environment, where a "battle" is waged between the communicating party and the party that is engaged in the pursuit of disrupting the communicator's link, strategy plays an important and fundamental role for the opposing parties. To the communicating party, the opponent's Electronic Support Measures (ESM) and Electronic Countermeasures (ECM) pose as threats. ESM involves essentially activities for spectrum surveillance and direction finding by passive means, whereas ECM involves activities for the purpose of victimizing the communicator's link. Jamming is an active measure of accomplishing ECM objectives. It is, therefore,

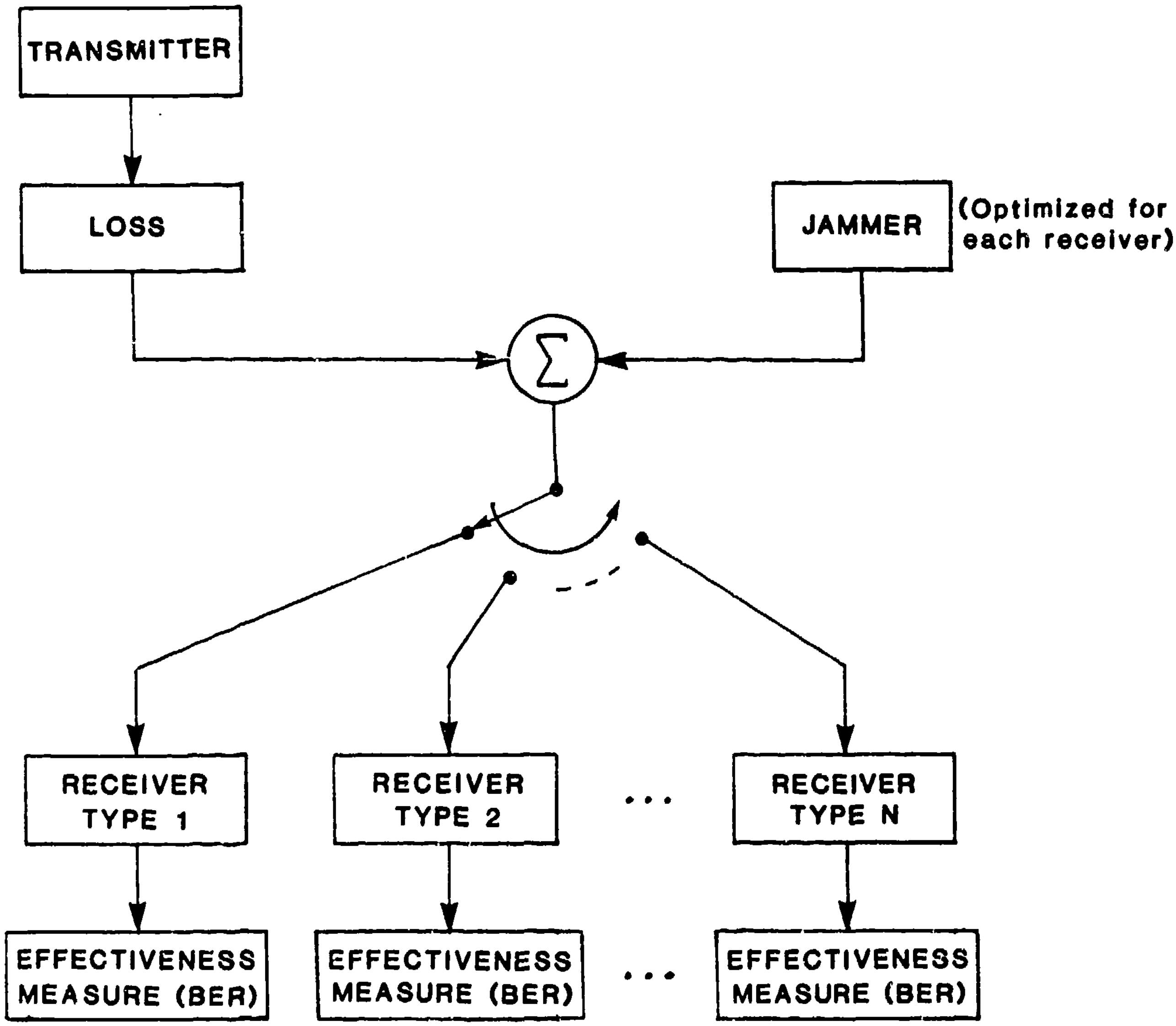


FIGURE 1.0-1 COMMUNICATIONS ECCM STRATEGY EVALUATION SCHEME

easy to recognize that fixed-frequency radios are very much vulnerable to ESM and ECM attacks. Communication systems that are designed to counter or mitigate the effects of ESM or ECM attacks are termed Electronic Counter-countermeasures (ECCM) radios, or jam-resistant communication systems.

In principle, there exist many different schemes which can provide the communicator with jam-resistant radio capabilities; Direct Sequence (DS) spread-spectrum and Frequency-Hopping (FH) spread-spectrum systems are two generic schemes. While the DS/SS system requires phase coherence over the system's wide operational bandwidth in its implementation, the FH/SS system does not. The fact that most of the tactical ECCM radios are of FH/SS type is based not only on this reason, but also on the fact that the attainable "processing gain" is achieved with less complexity and cost.

1.1.1 Jamming Strategy Against Frequency-Hopping Radios

ECCM radio designs are based on the desire to suppress the total jamming power by an amount equal to the processing gain, defined as the ratio of FH system bandwidth to the receiver noise bandwidth. The difference between the processing gain in dB and the SNR in dB required for traffic demodulation is the (anti-jam) margin that the communicator can use to tolerate an excess of jammer power over signal power at the system front end. The intelligent jammer, however, does not spread his power over the entire system bandwidth, so that the definition and effects of processing gain will not apply.

The jammer may employ a partial-band noise jamming strategy, in which the available jammer power is placed in a fraction (γ) of the radio system bandwidth, as illustrated in Figure 1.1-1. Assuming that total power is fixed, there is an optimum value of γ which achieves the most effective tradeoff of the probability of jamming and the probability of error when jammed, thus achieving a maximum overall error rate for the given amount of jammer power.

The jammer may, if it is feasible, concentrate his power further if he can intercept the hopping signal in real time and immediately broadcast a strong burst of noise in the frequencies near the signal (follow-on jamming). This type of jamming can be successful against FH systems in which a conventional narrowband communications signal is slowly hopped by simple translation in frequency.

1.1.2 ECCM Waveforms

Along with using hopped signals, the communicator can exercise an additional degree of freedom by employing a low energy density waveform to minimize interceptions by a potential jammer. Such a waveform is FH/MFSK using a number of hops per symbol (L), a kind of repetition code or diversity [5] to permit transmission at lower power and/or to combat fading. The conventional form of FH/MFSK is illustrated in Figure 1.1-2; once a symbol has been chosen for a given interval T_s, a conventional MFSK signal is generated and randomly hopped (translated) L times at a rate $R_H = L/T_s = 1/\tau$ before a new symbol is keyed. Because the M possible symbol frequencies are adjacent, this waveform is vulnerable to follow-on repeat jamming, unless the hopping rate can be made very high.

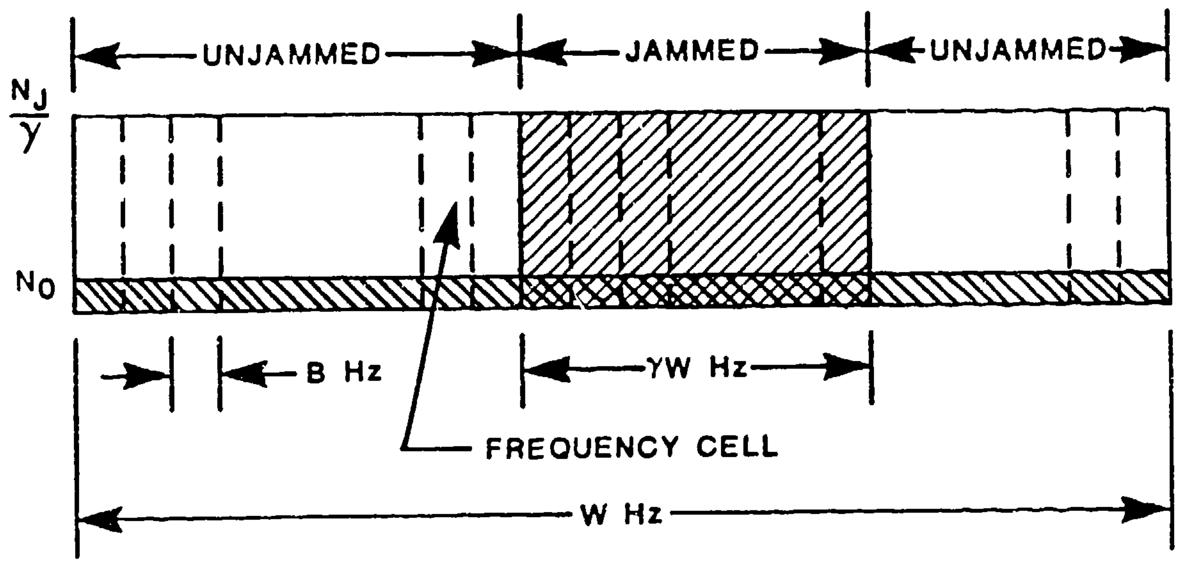


FIGURE 1.1-1 THERMAL NOISE AND PARTIAL-BAND NOISE JAMMING MODEL

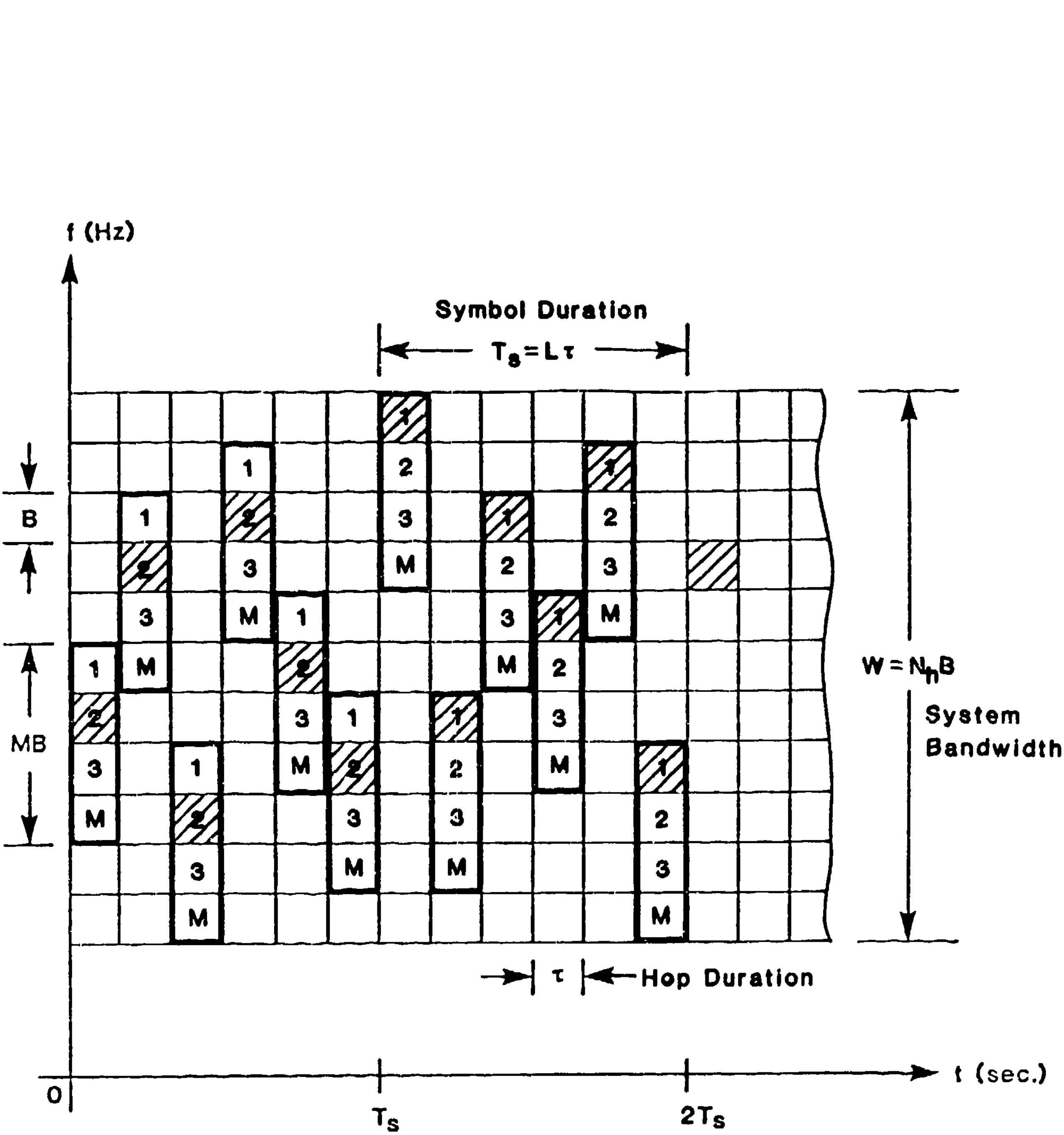


FIGURE 1.1-2 TYPICAL L HOPS/SYMBOL FH/MFSK WAVEFORM PATTERN

If the individual symbol frequencies of the MFSK symbol are assigned randomly on a per hop basis, as illustrated in Figure 1.1-3, then repeat jamming is less likely to produce an error since the symbol frequencies (at RF) are no longer adjacent [6,7]. (We shall refer to this waveform as FH/RMFSK with L hops per symbol.) It has been shown [6] that, for the special case of one hop/bit binary systems (M=2) and very little system or thermal noise (E_b/N_0 = 30 dB), the two forms of FH/BFSK achieve the same performance in optimum partial-band noise jamming. This suggests that the random MFSK waveform will perhaps be a better choice for $L > 1$ and $M > 2$, although it has not been established as a fact that its performance in partial-band noise is always equal to that of the conventional system, while offering additional protection against follow-on jamming.

The FH/RMFSK implementation is more complex, and the study in this report will permit the cost of this additional complexity to be weighed against its performance, compared to that of the more conventional FH/MFSK as calculated by LAI [1].

1.1.3 Motivation for the Proposed Random Hopping

As we have stated above, the proposed FH/RMFSK waveform is less vulnerable to follow-on or repeat jamming than is a conventional FH/MFSK systems with M contiguous signalling frequencies. In addition, the FH/RMFSK waveform is less vulnerable to tone jamming, since the randomized selection of the M frequencies reduces the amount of structure in the signal. This makes it

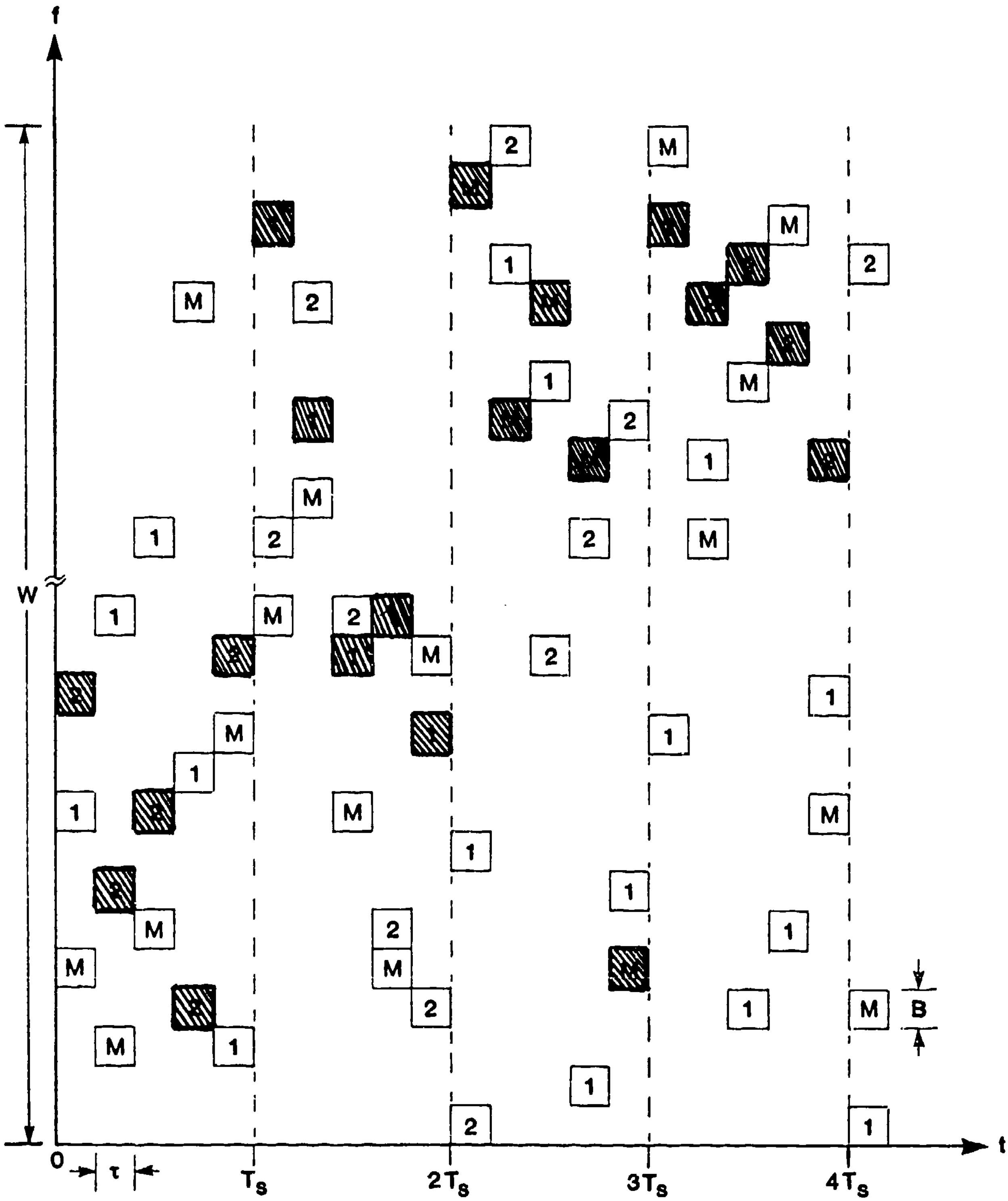


FIGURE 1.1-3 TYPICAL L HOPS/SYMBOL FH/RMFSK WAVEFORM PATTERN

much more difficult for a tone jammer to implement an optimum jamming strategy consisting of one jamming tone per M-ary symbol: the lack of structure in the hopping gives the jammer no features to exploit to insure a tone hits the symbol; thus the jammer is forced to divide his available power up into more tones, resulting in a lessened effect on the communications link when it hops into a jammed slot.

Therefore, the motivation for considering the use of FH/RMFSK as an LPI anti-jam communications system design is based upon a desire to lessen or reduce vulnerability to certain more sophisticated jamming threats, such as follow-on jamming and tone jamming. However, before the system can be considered a viable design candidate, its performance under the less sophisticated jamming, namely partial-band noise jamming, must be known.

1.1.4 Rationale for the Exact Analysis of FH/MFSK System Performance In Partial-Band Noise Jamming

It is known that the advantage of an M-ary orthogonal modulation system rests on the fact that the scheme requires less energy per data bit transmission than other available modulation schemes. Cost-effective implementation (efficient non-coherent detection) is another reason in selecting M-ary FSK waveforms by designers of ECCM radios. Recently, Hughes Aircraft Company has conducted studies for U.S. Army CECOM on feasibility of AJ/LPI ECCM techniques, employing L-hops per symbol FH/MFSK [8].

Exact knowledge of performance measures and vulnerability of L-hops per symbol FH/MFSK SS systems has not been available until recently, and that of FH/RMFSK is yet to be determined. Workers in this field previously held the view that M-ary system performance measures could be estimated once the performance measures of the binary systems are available. This view was based on the conventional wisdom of applying the "union bound". As is well known, once we know the binary system performance, that of the M-ary system can be approximated by the union bound given by

$$P_M(e;E_s) \leq (M-1)\, P_2(e;E_s) \tag{1.1-1}$$

where $P_2(e;E_s)$ is the probability of error for the binary system, using any pair of symbols from the set $\{s_1(t), s_2(t),\ldots,s_M(t)\}$, and $P_M(e;E_s)$ is the M-ary system probability of error, where E_s is the symbol energy. The workers have also invoked the well-known relationship between the bit error probability and the symbol (K-bit word) error probability for the M-ary orthogonal system. That is,

$$P_b(e;E_b) = \frac{M}{2(M-1)}\, P_M(e;E_s). \tag{1.1-2}$$

where $P_b(e;E_b)$ denotes bit error probability and E_b is the energy per bit. By putting equation (1.1-1) into equation (1.1-2), we obtain the "union bound"-based approximate-performance measure of the bit error probability of an MFSK systems, given by

$$P_b(e;E_b) \leq \frac{M}{2}\, P_2(e;E_s) = 2^{K-1}\, P_2(e;KE_b) \tag{1.1-3}$$

where

$$K = \log_2 M. \tag{1.1-4}$$

In principle, one can use equation (1.1-3) to assess the performance of an M-ary orthogonal system. Our discovery, however, did not support this generality when the communication channel is of the non-exponential type. In an attempt to obtain an approximate performance measure of 2-hops per symbol FH/MFSK system under partial-band noise jamming environment for M=4, 8, 16, and 32, we have used equation (1.1-3) in applying the binary results, as shown in Figures 1.1-4 to 1.1-6, to three different receiver schemes. A surprising result is that as M is increased, bit error probability as given by the union bound is worsened for all three receivers! Interpretation of this result is that one needs to expend more energy per bit in the higher-order-message-alphabet orthogonal system, a result that is not supportable even on the basis of intuition; and is, indeed, contrary to the exact results shown in the figures.

The above paragraph is to point out that the "union bound" cannot be used when one considers non-Gaussian channels such as partial-band noise, as experienced by a FH/MFSK system. These channels are inverse linear channels, and they do not allow the union bounding techniques to be applicable in assessing M-ary system performances. Thus, one can conclude that exact analysis is necessary.

1.1.5 Extension of Uncoded Error Analysis to Coded Performance

While error-control coding is quite likely to be used by the communicator to counter any jamming effects, the analysis of total system performance may be usefully divided into two parts: uncoded performance and

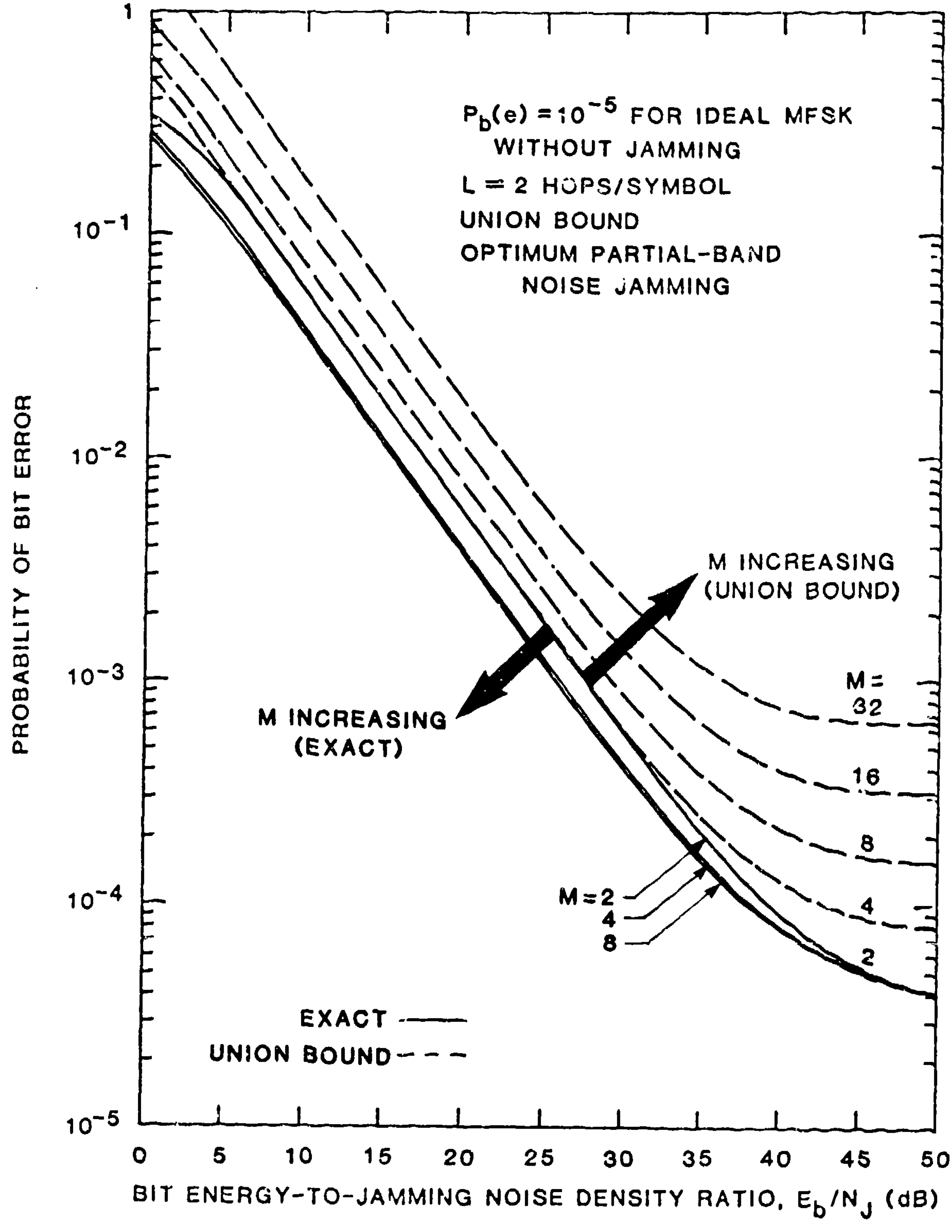


FIGURE 1.1-4 COMPARISON OF THE EXACT ANALYSES WITH THE UNION BOUNDS FOR THE SQUARE-LAW LINEAR COMBINING RECEIVER

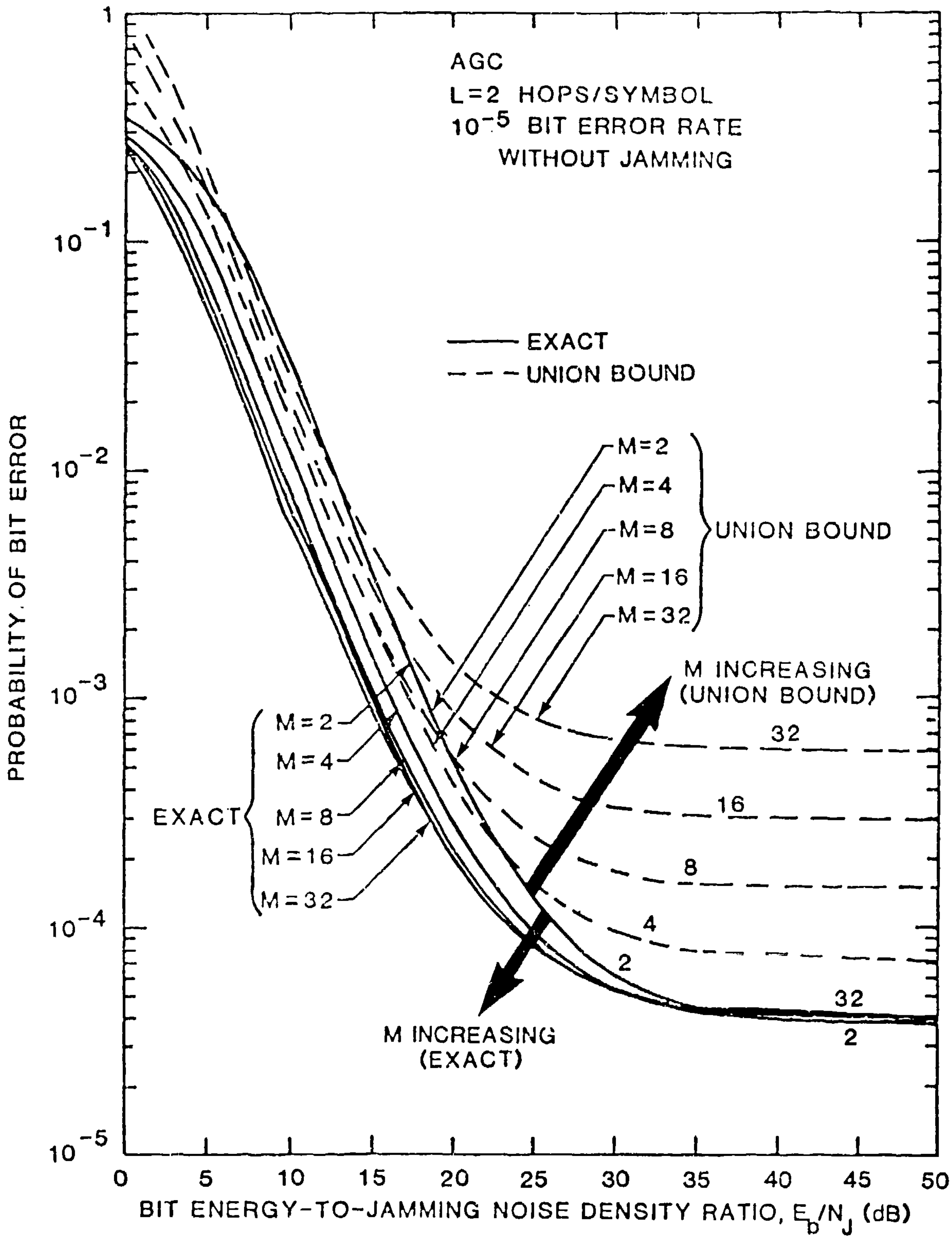


FIGURE 1.1-5 COMPARISON OF THE EXACT ANALYSES WITH THE UNION BOUNDS FOR THE AGC RECEIVER

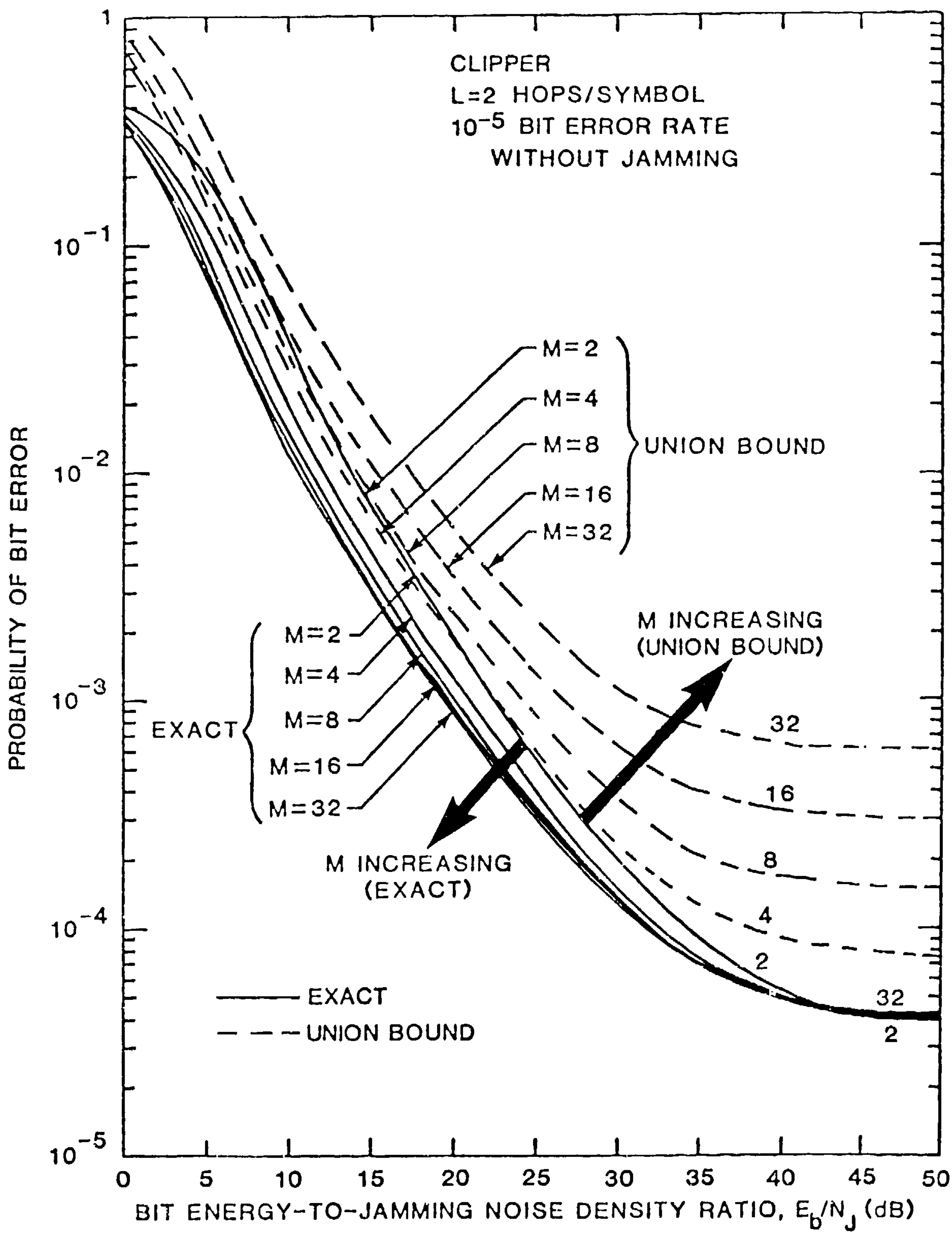


FIGURE 1.1-6 COMPARISON OF EXACT ANALYSES WITH THE UNION BOUNDS FOR THE CLIPPER RECEIVER

enhanced performance using coding. For comparison studies of anti-jam demodulation schemes such as we are proposing, it is sufficient to consider uncoded performance, since the coded system performance is proportional to the uncoded.

For example, for code words using n channel symbols the probability of word error for a bounded-distance decoding algorithm is [9]

$$P_W = \sum_{i=t+1}^{n} \binom{n}{i} P_s^i (1 - P_s)^{n-i} \qquad (1.1\text{-}5)$$

where P_s is the uncoded performance in terms of symbol errors and t is the number of correctable errors. This word error probability can be translated into an equivalent information bit error probability by a formula appropriate to the particular coding and decoding algorithms.

1.2 ECCM PROCESSING

Once the FH/MFSK or FH/RMFSK waveform has been dehopped at the intended receiver, the L hops constituting the MFSK symbol can be combined in several ways. It has been shown [10] that the conventional method of summing up the (non-coherent) L hop energies, although effective against fading, produces a BER which increases with L against optimum partial-band noise jamming. Therefore, a number of non-linear combining schemes have been studied, based on weighting the dehopped and envelope-detected hops in some fashion to discriminate against those hops which have been jammed [1, 11, 12].

Using these nonlinear combining schemes it has been shown for FH/MFSK that the use of $L > 1$ hops per symbol can be understood as providing a kind of diversity improvement against the jamming, depending on the system noise level.

It has not been determined how a FH/RMFSK waveform with L hops per symbol will perform against optimum partial-band noise, whether using conventional or nonlinear soft-decision combining of the hops.*

1.2.1 Examples of Receiver Effectiveness Computations for FH/MFSK

Under contract to the Office of Naval Research, LAI has studied in great detail the uncoded performances of frequency hopped BFSK and MFSK communication systems under optimum partial-band noise jamming [1, 10, 11, 13, 14, 15]. The focus of these efforts has been to determine both the optimum partial-band jamming strategy and the most effective anti-jam receiver processing schemes for this type of modulation, using exact analyses which include the system's thermal noise. One of the chief results of our work has been the discovery that conclusions drawn from previous, approximate studies neglecting thermal noise are not strictly valid. It had been commonly asserted that the use of multiple hops per symbol in FH/MFSK systems provides a diversity gain improvement against optimum partial-band jamming in much the same way that it does against the effect of fading on the signal. We have been able to show that this improvement does not exist for the conventional (linear combining) receiver, and we have demonstrated quantitatively

*In a recent paper [16], FH/RMFSK performance with L hops per symbol has been shown for a receiver using hard decisions. The binary hard-decision case was also analyzed in [17].

that a limited improvement holds for certain nonlinear hop combining receiver processing schemes ("metrics"), as a function of the system's thermal noise.

A generic model of FH/MFSK square-law receivers is given in Figure 1.2-1. Among the processing schemes, represented by the function $f_k(\cdot)$ in the figure prior to the accumulation of soft decision statistics $\{z_m\}$, are those listed in Table 1.2-1. The performance of the conventional, linear combining receiver in optimum partial-band noise jamming was calculated directly and compared to that of the three nonlinear combining receivers. For the calculation, the bit error probabilities were expressed by

$$P_b(e) = \sum_{\ell=0}^{L} p_\ell \; P_b(e|\ell), \tag{1.2-1}$$

where p_ℓ is the probability that ℓ out of L hops constituting a given symbol are jammed, and $P_b(e|\ell)$ is the bit error probability given that ℓ hops are jammed. For conventional FH/MFSK, we have assumed that

$$p_\ell = \binom{L}{\ell} \gamma^\ell \; (1-\gamma)^{L-\ell} \tag{1.2-2}$$

based on all of the M symbol frequency slots being jammed on a given hop, with probability γ (the fraction of the system bandwidth which is jammed), or none of them being jammed, with the probability $1-\gamma$.

For each receiver type and values of E_b/N_0 and E_b/N_J, the maximum bit error probability was found as a function of γ, the partial-band jamming

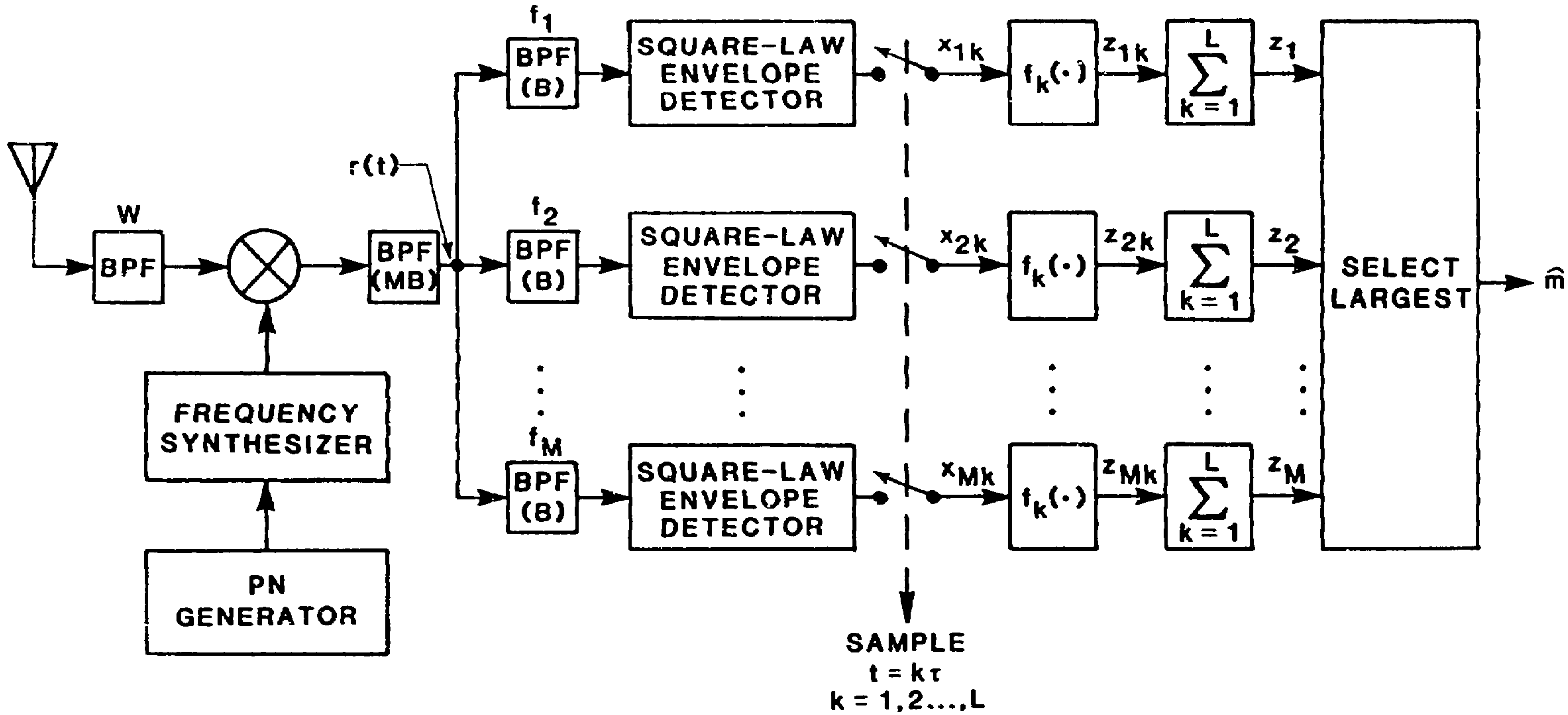


FIGURE 1.2-1 FH/MFSK SQUARE-LAW GENERIC RECEIVER

TABLE 1.2-1

DESCRIPTIONS OF THE RECEIVERS

RECEIVER TYPE	SPECIFICATION OF $z_{ik} = f_k(x_{ik}),\ i=1,2,\ldots,M$	REMARKS
LINEAR COMBINING RECEIVER	$z_{ik} = x_{ik}$	Direct Connection (Linear Combining)
CLIPPER RECEIVER	$z_{ik} = \begin{cases} x_{ik}, & x_{ik} \leq \eta \\ \eta, & x_{ik} > \eta \end{cases}$	Soft Limiter (Nonlinear Combining)
AGC RECEIVER	$z_{ik} = x_{ik}/\sigma_k^2$ $\sigma_k^2 = \begin{cases} \sigma_N^2, & \text{if not jammed} \\ \sigma_N^2 + \sigma_J^2, & \text{if jammed} \end{cases}$ (σ_k^2 = measured)	Adaptive Gain Control (Nonlinear Combining)
SELF-NORMALIZING RECEIVER	$z_{ik} = \dfrac{x_{ik}}{\sum_{i=1}^{M} x_{ik}}$	Practical Realization of AGC Using In-Band Measurements

fraction. These calculations revealed significant differences among the receiver types in the optimum value of γ as well as in the bit error probability. For example, in Figure 1.2-2, we show that for M=8 that the jammer's optimum γ is much more sensitive to the value of L, the number of hops per MFSK symbol, for the AGC receiver than for the clipper receiver. Therefore the jammer must have more accurate information on the modulation parameters in order to be as effective as possible against the AGC receiver.

Another typical result is the comparison shown in Figure 1.2-3, also for M=8, and for L=2 hops per symbol. We see that the (ideal) AGC form of ECCM receiver processing is significantly better at combatting the effects of the jamming, and that the clipper receiver also improves the BER, but not as much.

Figure 1.2-4 shows the effect of increasing E_b/N_0 so as to provide a lower bit error probability in the absence of jamming for the AGC receiver with M=4 and L as a paramter. We see that under these conditions, the optimum choice of L includes higher values of the number of hops per symbol before increased noncoherent combining loss dominates and forces a choice of a lower value of L.

Figure 1.2-5 illustrates the performance as $E_b/N_0 \to \infty$, i.e. no thermal noise, for FH/BFSK (i.e. M=2). We see that in the absence of thermal noise, the optimum value of L increases without limit as E_b/N_J increases. A similar result holds for the case of M>2.

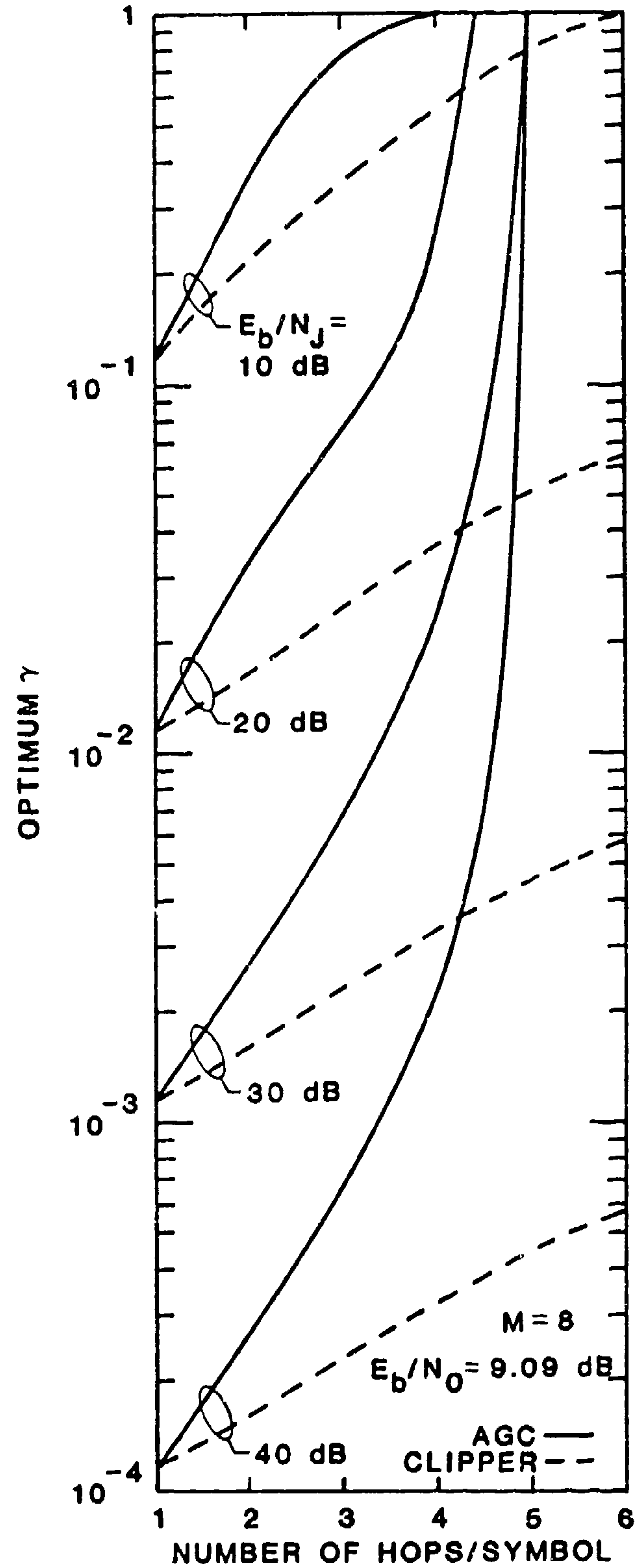


FIGURE 1.2-2 OPTIMUM JAMMING FRACTION (γ) VS. NUMBER OF HOPS/SYMBOL (L) FOR THE AGC AND CLIPPER FH/MFSK (M= 8) RECEIVERS WHEN E_b/N_0 = 9.09 dB WITH E_b/N_J AS A PARAMETER (FOR 10^{-5} ERROR RATE WITHOUT JAMMING)

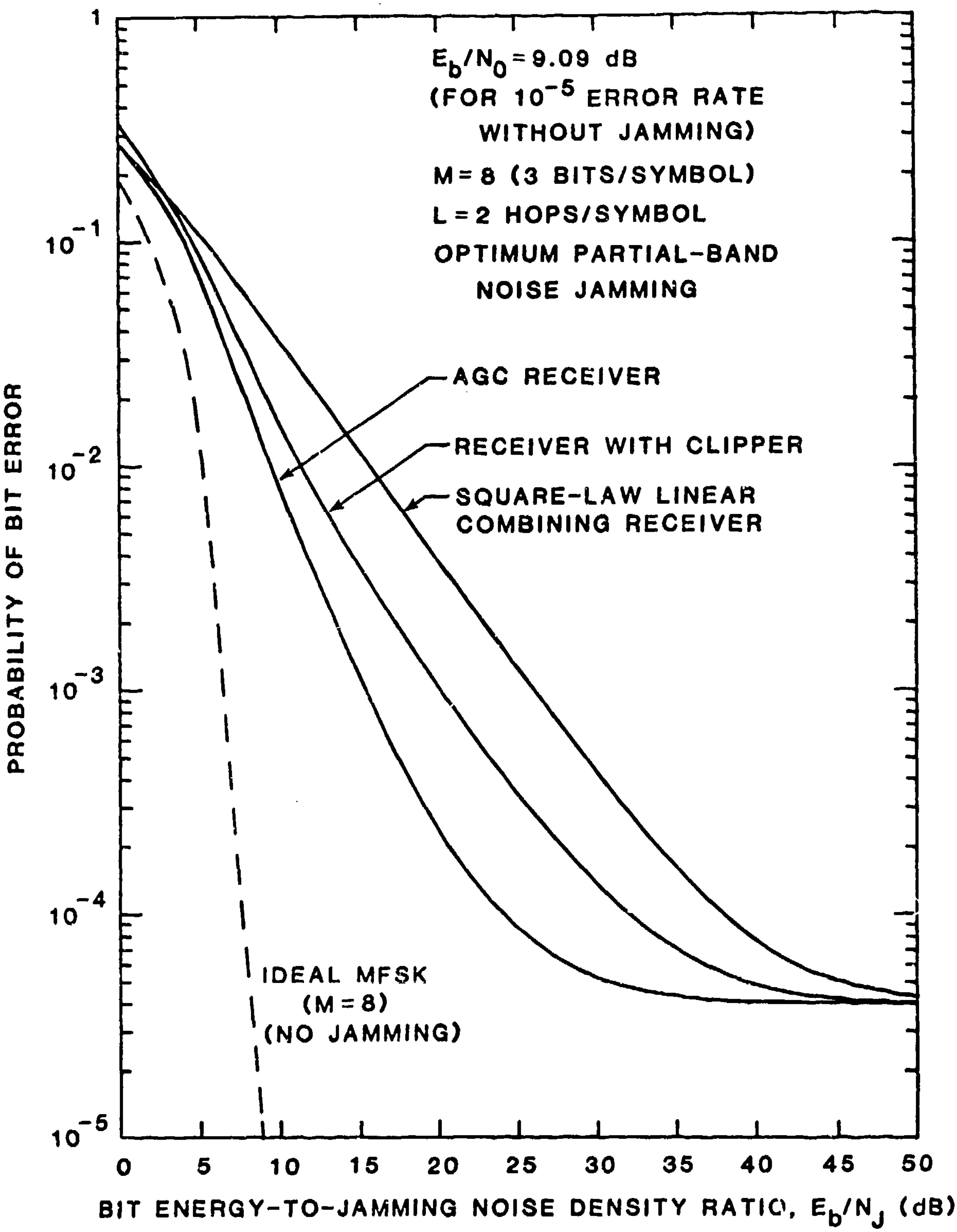


FIGURE 1.2-3 OPTIMUM PARTIAL-BAND NOISE JAMMING PERFORMANCE OF FH/MFSK (M= 8) SQUARE-LAW COMBINING RECEIVERS FOR L= 2 HOPS/SYMBOL WHEN E_b/N_0= 9.09 dB (FOR IDEAL MFSK (M = 8) CURVE THE ABSCISSA READS E_b/N_0)

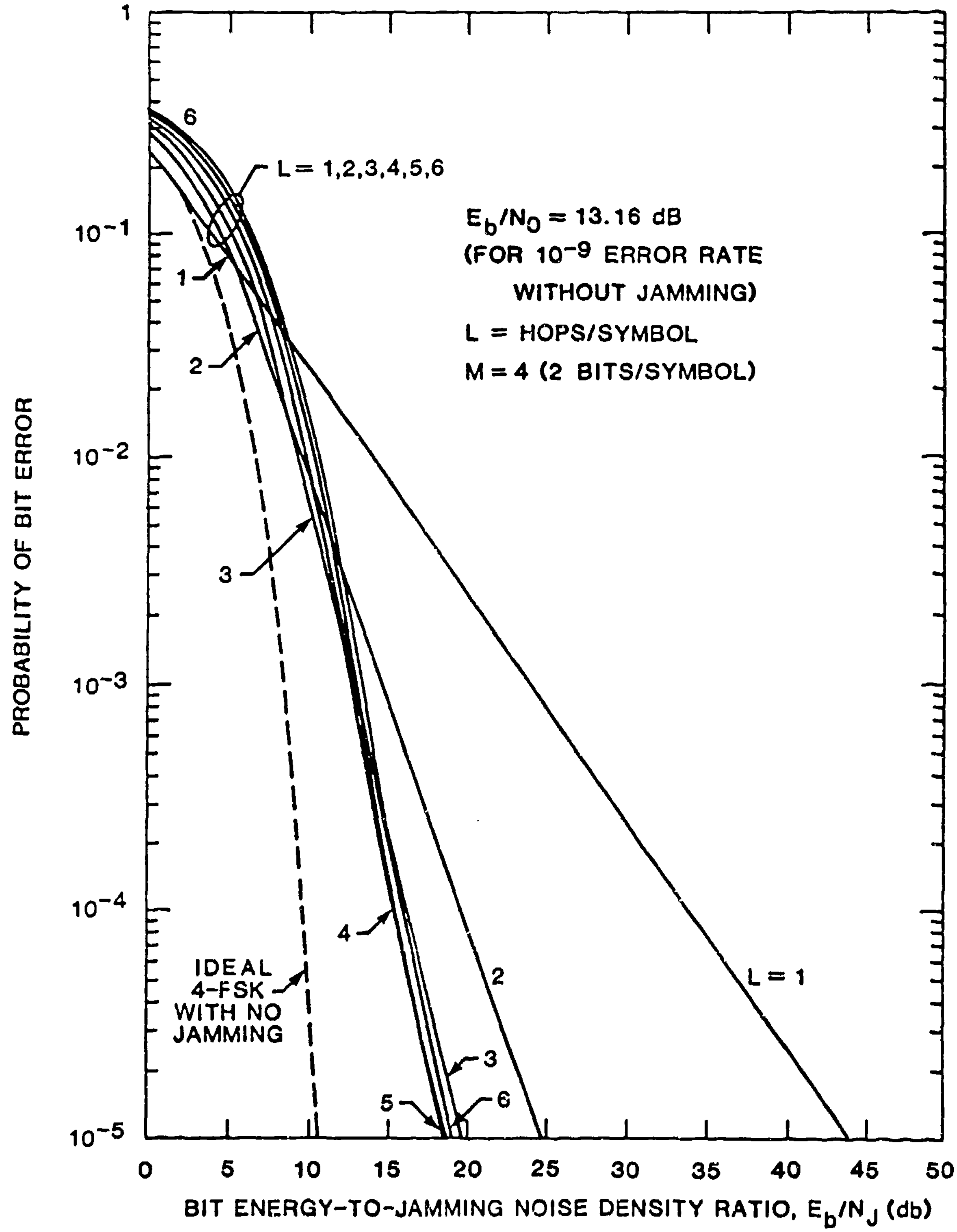


FIGURE 1.2-4 OPTIMUM JAMMING PERFORMANCE OF THE AGC FH/MFSK (M = 4) RECEIVER WHEN E_b/N_0 = 13.16 dB WITH THE NUMBER OF HOPS/SYMBOL (L) AS A PARAMETER (FOR IDEAL MFSK (M = 4) CURVE THE ABSCISSA READS E_b/N_0)

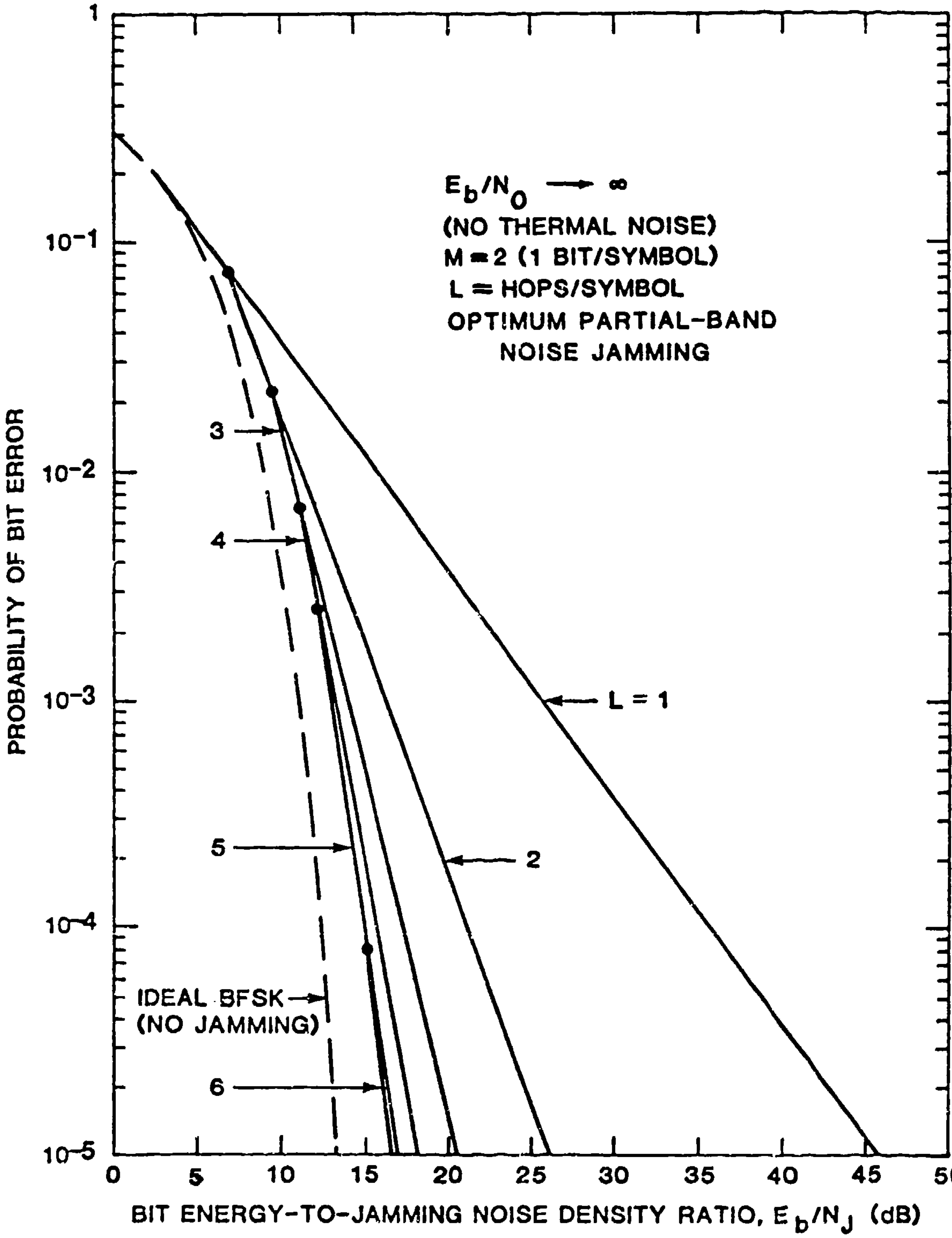


FIGURE 1.2-5 OPTIMUM JAMMING PERFORMANCE OF THE AGC RECEIVER FOR BFSK/FH WITH THE NUMBER OF HOPS/BIT (L) AS A PARAMETER WHEN THERMAL NOISE IS ABSENT (FOR IDEAL BFSK CURVE THE ABSCISSA READS E_b/N_0)

Our exact calculations permit the construction of composite curves such as illustrated in Figure 1.2-6, in which the performance of the AGC receiver processing scheme for FH/BFSK (M=2) is shown for the optimum L values at different thermal noise levels. It is seen that for $E_b/N_0 > 15$ dB, the use of the proper number of hops per bit enables the communication systems to recover the performance of unjammed BFSK to within 3 dB of SNR. The results for $E_b/N_0 < 15$ dB are very sensitive to thermal noise, and had not been predicted by other workers, who ignored thermal noise.

1.2.2 Impact of Random FH/MFSK (FH/RMFSK) on Analysis

Evaluation of the BER performance of ECCM receiver processing schemes becomes significantly more complex for M>2 and L>1 when the MFSK symbol frequency assignments are not contiguous but each randomly chosen to be anywhere in the hopping band. The complexity consists in there being many more jamming events than those reflected in equation (1.2-1), since now on each hop there can be from 0 <u>to</u> M of the dehopped symbol frequency slots jammed on a given hop (rather than 0 <u>or</u> M). The probability of bit error expression accordingly must be generalized, giving

$$P(e) = \sum_{\ell_1=0}^{L} \sum_{\ell_2=0}^{L} \cdots \sum_{\ell_M=0}^{L} \Pr(\ell_1,\ell_2,\ldots\ell_M)\, P(e|\ell_1,\ell_2\ldots\ell_M) \qquad (1.2\text{-}3)$$

in which the number of jammed hops in each symbol channel is explicitly enumerated and accounted for in the conditional probability of error calculations.

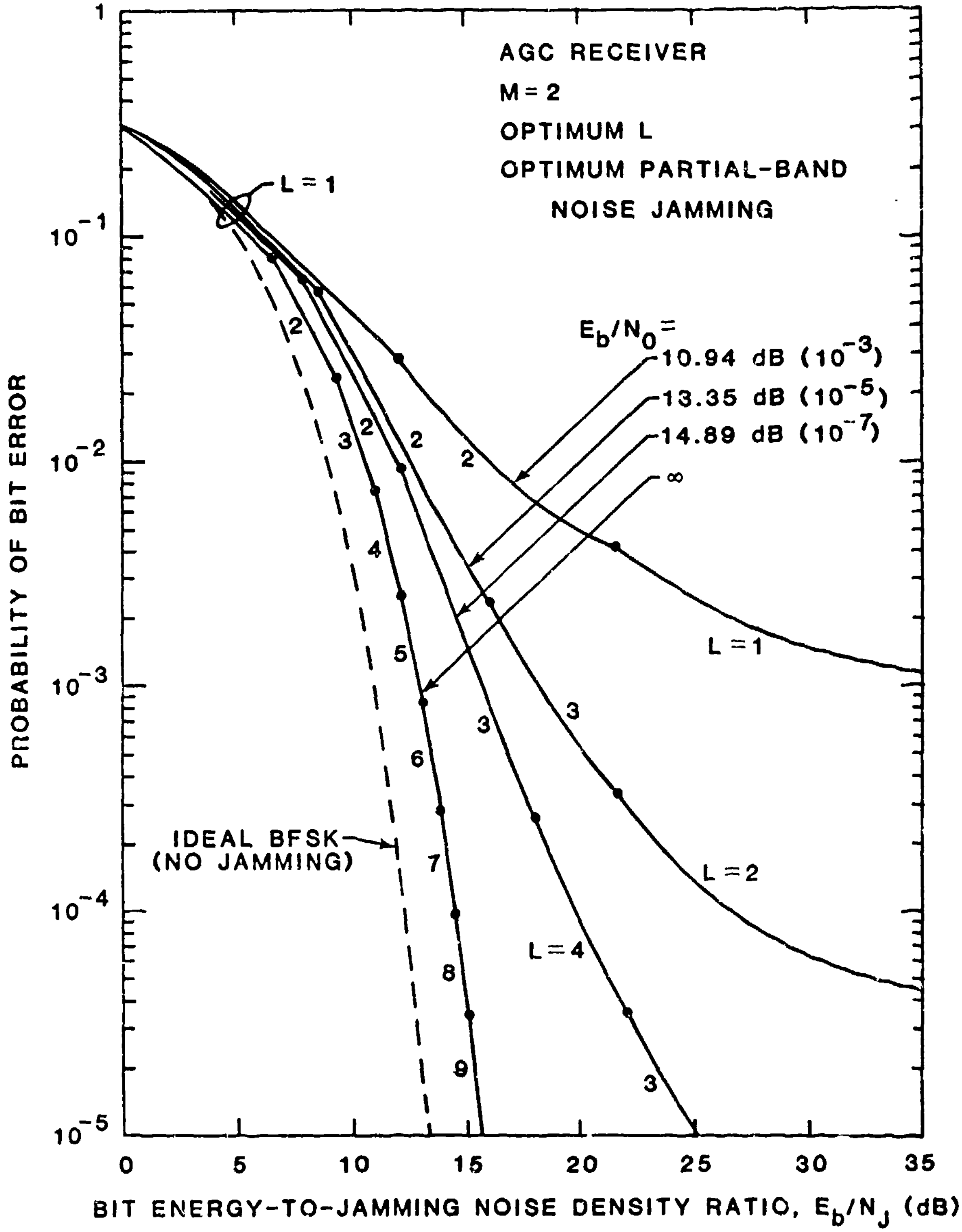


FIGURE 1.2-6 PROBABILITY OF BIT ERROR VS. E_b/N_J FOR AGC FH/BFSK RECEIVER USING OPTIMUM NUMBER (L) OF HOPS/BIT, FOR DIFFERENT VALUES OF E_b/N_0 IN WORST-CASE PARTIAL-BAND NOISE JAMMING

From equation (1.2-3) it is apparent that up to $(L + 1)^M$ jamming events may be distinguished, if it can be assumed that the symbol decision is affected only by the total numbers of jammed hops $\{\ell_m\}$ in the M dehopped channels, rather than by the individual hop patterns. The sheer number of events can therefore become the major factor influencing the magnitude of the receiver effectiveness evaluation task in terms of computational effort. LAI has had experience in the computation of similar expressions in the connection with the evaluation of tone jamming effects on FH/MFSK systems [1].

1.3 SUMMARY OF REPORT

In this section, we will first give a general description of the work. We then summarize the report organization and major findings.

1.3.1 General Description of Work and Approach.

In Sections 1.1 and 1.2 we discussed the fundamental issues concerning ECCM systems and ECCM processing. Now, we treat the more specific ECCM system which we have studied, namely FH/RMFSK in the presence of partial-band noise jamming.

1.3.1.1 Receiver models studied.

A generic soft-decision receiver structure for an FH/RMFSK waveform is shown in Figure 1.3-1. The incoming waveform is dehopped by mixing it separately with M hopping local oscillators controlled by replicas of the M possible hopping sequences available for transmission by the transmitter.

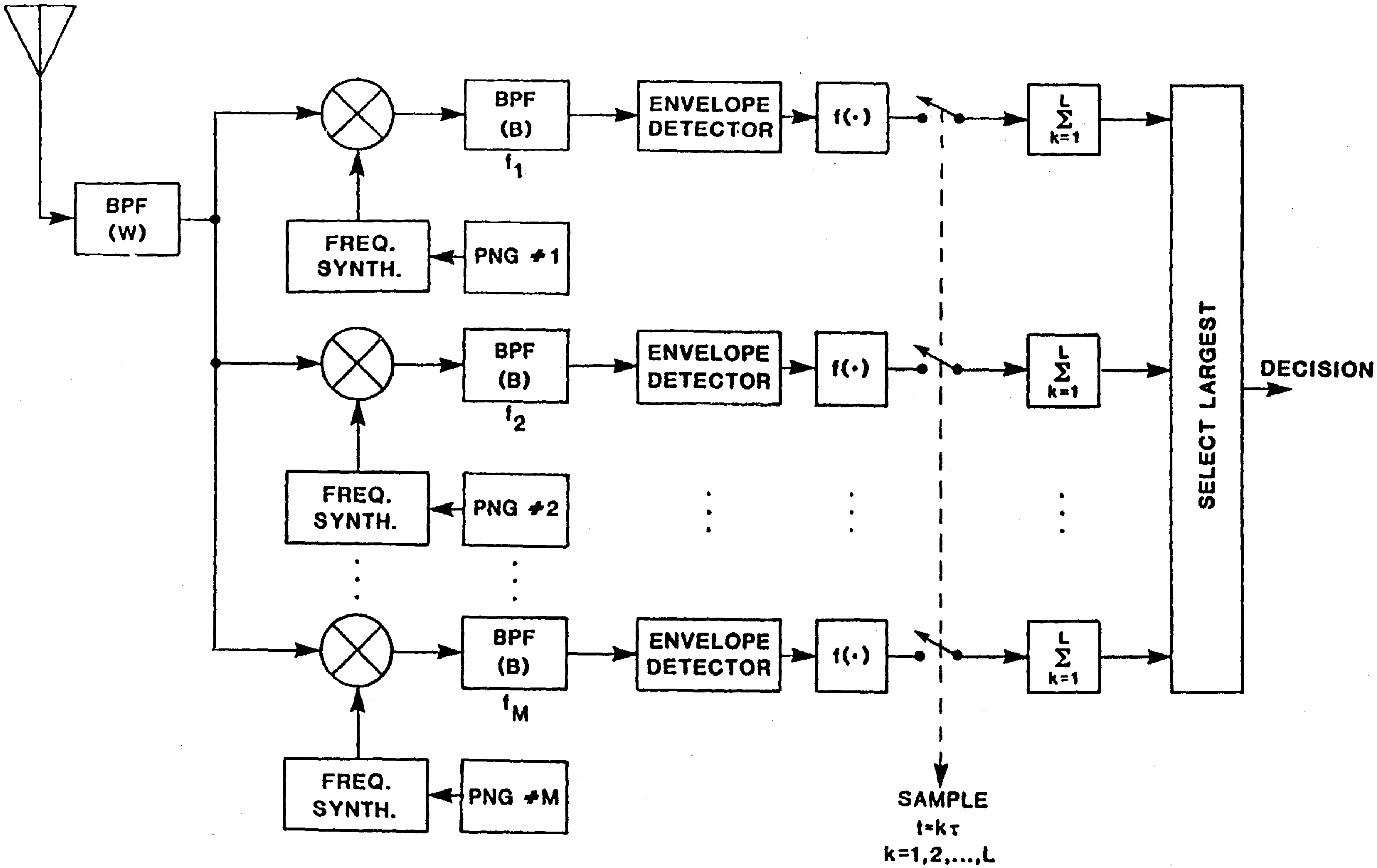


FIGURE 1.3-1 SOFT-DECISION RECEIVER FOR FH/RMFSK

Thermal noise with power spectral density N_0 is present over the entire bandwidth W. A fraction, γ, of the band is jammed by bandlimited white Gaussian noise of power spectral density N_J/γ, where $N_J \triangleq J/W$ with J being the total jammer power. The jamming fraction, γ, is constrained to the range $0 < \gamma \leq 1$.

The relation between the jammed bandwidth γW and the FH/RMFSK waveform is illustrated in Figure 1.3-2. On any given hop, anywhere from 0 to M of the possible signalling frequencies may have hopped into the jammed portion of the band; thus a multitude of jamming events may occur. Let the L hops for a given symbol be referred to individually by the index k (k = 1, 2,...,L). The jamming events for the kth hop can be described in terms of which of the M symbol frequencies are jammed, and which are not. In general there are 2^M possibilities for a given hop, which we may specify by the indicator vector

$$\underline{\nu}_k = (\nu_{1k}, \nu_{2k}, \ldots, \nu_{Mk}) \tag{1.3-1}$$

where

$$\nu_{mk} = \begin{cases} 1 \text{ if symbol slot m is jammed on hop k} \\ \\ 0 \text{ if not;} \end{cases}$$

$$m = 1, 2, \ldots, M; \quad k = 1, 2, \ldots, L. \tag{1.3-2}$$

For the L hops comprising a symbol, there are 2^{ML} possible jamming events, and these can be specified individually by the $M \times L$ indicator matrix $[\nu] \equiv [\nu_{mk}]$.

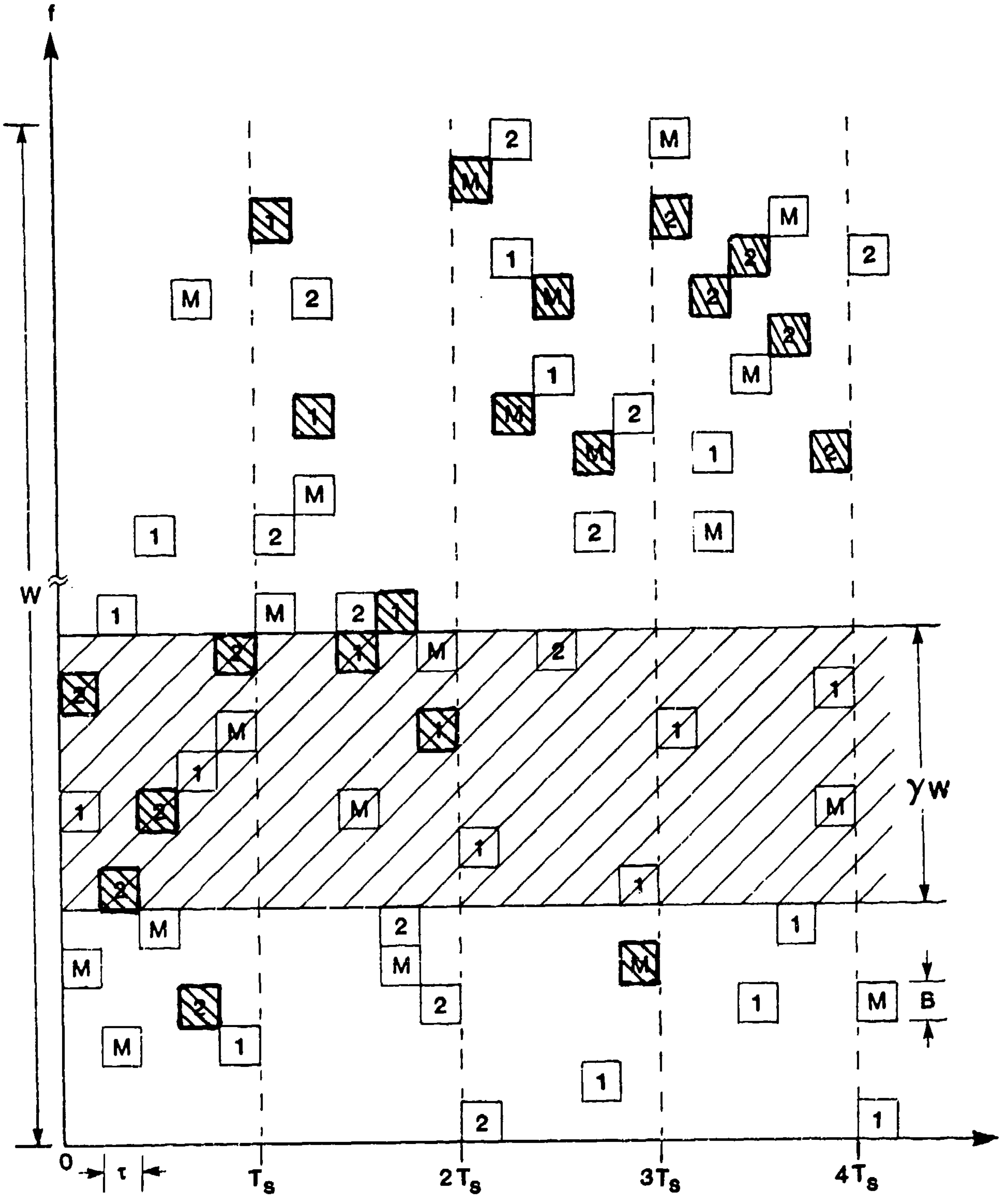


FIGURE 1.3-2 PARTIAL-BAND JAMMING OF FH/RMFSK

Each dehopped channel, corresponding to one of the M possible symbols, is then passed through a bandpass filter of width B Hz and the filter output is envelope detected. The output of each linear envelope detector is subjected to a function $f(\cdot)$; the form of this function defines the particular receiver structure.* Table 1.3-1 gives the forms of $f(\cdot)$ for the several receiver structures we include in the study. The modified envelopes are sampled once per hop and the samples in each channel are summed over the L hops comprising a symbol. The largest of these sums is selected and the index identifying the channel in which it occurred is outputted as the symbol decision.

As an alternative to the soft-decision receiver scheme described above, we may also consider the hard-decision receiver structure which is shown in Figure 1.3-3. The processing in this hard-decision receiver is identical with the soft-decision receiver up to the outputs of the samplers. In the hard-decision receiver, unlike the soft-decision receiver, the samples are not summed; rather, a symbol decision is made each hopping interval, giving a sequence of L decisions. These L decisions may be considered as a noise-corrupted received code-word in an M-ary repetition code wherein the transmitted symbol is repeated L times; thus the sequence is fed into an L-hop M-ary repetition code decoder which delivers the final decision as to which symbol was transmitted.

1.3.1.2 Jamming model and measure of effectiveness.

The partial-band noise jamming model was shown in Figure 1.1-2.

*As long as $f(\cdot)$ is a memoryless transformation, the order of applying $f(\cdot)$ and the sampling may be interchanged without altering the receiver's performance.

TABLE 1.3-1

RECEIVER PROCESSING FUNCTIONS STUDIED

RECEIVER TYPE	$f(\cdot)$
Square-Law Linear Combining	$f(x_i) = x_i^2$
Square-Law with Clipper	$f(x_i) = \begin{cases} x_i^2, & x_i < \sqrt{\eta} \\ \eta, & x_i \geq \sqrt{\eta} \end{cases}$
Square-Law AGC	$f(x_i) = x_i^2/\sigma_i^2$
Self-normalizing receiver	$f(x_i) = \dfrac{x_i^2}{\sum_{j=1}^{M} x_j^2}$

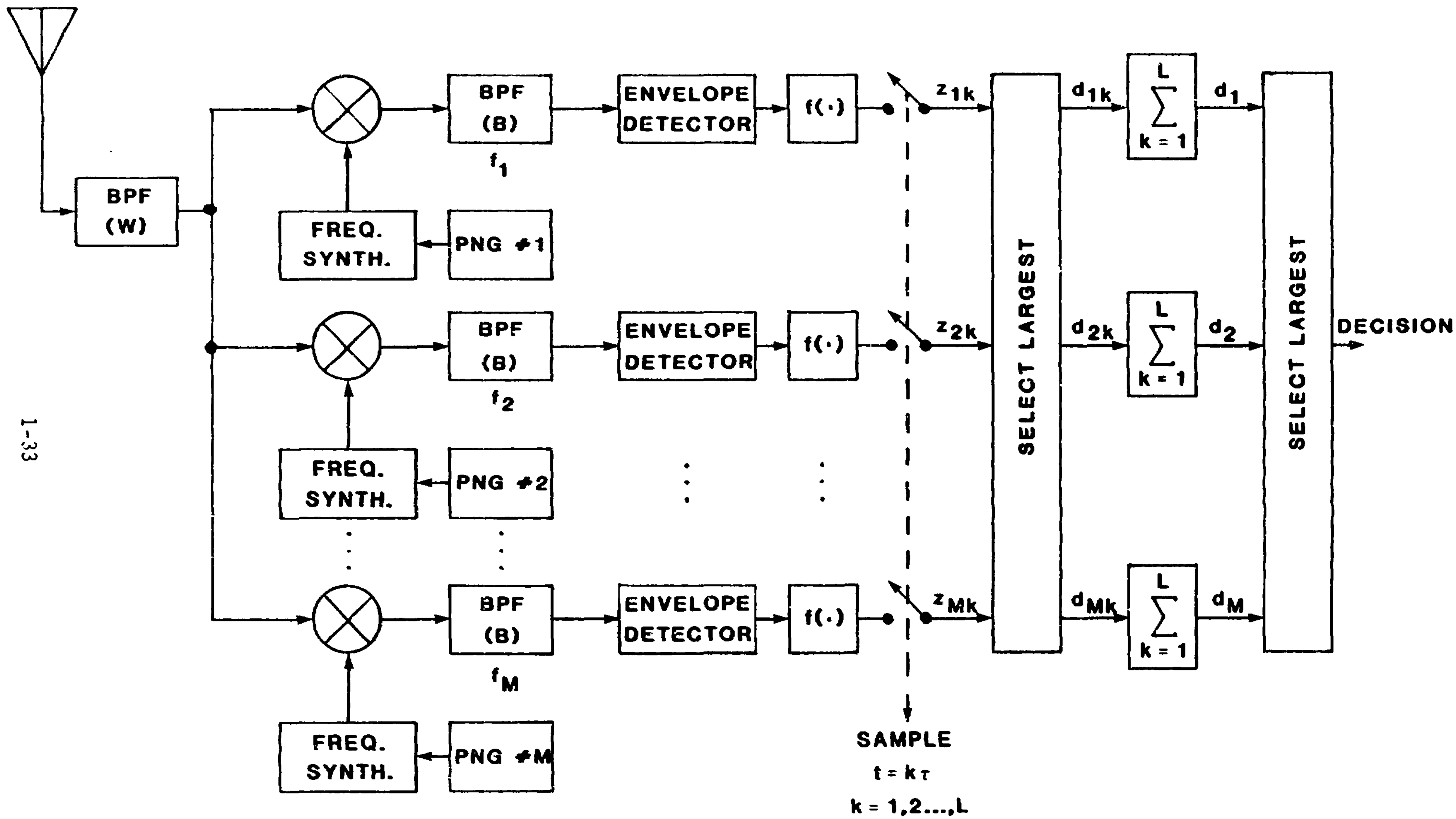


FIGURE 1.3-3 HARD-DECISION RECEIVER FOR FH/RMFSK

The measure of the effectiveness of the jammer is the degradation of the communicator's bit error probability inflicted by the presence of the jamming. Since the bit error probability $P_b(e)$ will depend upon the jamming event, we must average the error probability over the jamming events. Thus, the measure of effectiveness of the jamming is

$$P_b(e;\, E_b/N_0,\, E_b/N_J,\, \gamma,\, M,\, L) = \sum_{[\nu]} P_b(e;\, E_b/N_0,\, E_b/N_J,\, \gamma,\, M,\, L|[\nu])\Pi_L[\nu]$$

where $\Pi_L[\nu]$ is the probability of jamming event $[\nu]$ occurring over the L hops of the M-ary symbol. Thus the required analysis may be divided into two parts: determination of $\Pi_L[\nu]$ and determination of $P_b(e;\, E_b/N_0,\, E_b/N_J, \gamma\,,\, M,\, L|[\nu])$. These two parts can then be combined to perform the final optimization, namely finding the receiver performance under the optimum jamming fraction γ, $\max\limits_{\gamma} P_b(e;\gamma)$.

1.3.1.3 Organization of report.

In Section 2 we address parts of the analysis considered preliminary or containing aspects common to the several receiver types. This material includes enumeration of jamming events and analysis of their probabilities, as well as an analysis of the hard-decision receiver.

Sections 3 to 6 are devoted, respectively, to analysis and numerical results for the worst-case partial-band noise jamming error performances of FH/RMFSK using the linear combining receiver, the adaptive gain control (AGC) type receivers, the clipper receiver, and the self-normalizing receiver.

Section 7 first provides analysis and results for the performance of FH/RMFSK in follow-on noise jamming, then comparisons of RMFSK receivers with regard to their overall relative error performance, their performances in both RMFSK and MFSK, and their success in using diversity (multiple hops per symbol) to mitigate jamming effects.

Section 8 considers issues related to implementation of the FH/RMFSK receivers, including a discussion of possible measurement approaches to support the ECCM weighting schemes, and an assessment of the effect of using practical measurements (instead of *a priori* information) on the system performance. Conclusions and recommendations growing out of our study are included in Section 8 also.

1.3.2 Summary of Findings.

Here we only briefly cite the more significant findings from our study; more detailed information is contained throughout the report.

The overall significance of the work we have accomplished may be described as follows: For the first time, the expected performance of an FH/RMFSK system using L hops per symbol and soft decisions, in both thermal noise and worst-case partial-band jamming noise, has been derived and calculated. Moreover, we have demonstrated through direct analysis and calculation of bit error rate (BER) that the performances of certain practical soft-decision ECCM receivers (using no *a priori* or side information) are quite acceptable, being very close to those for idealized receivers (using *a priori* information on received noise and jamming conditions). While we have shown that the hard-decision receiver does implement a form of ECCM processing (not previously shown) against PBNJ in the most simple manner, it cannot be considered a viable alternative unless the system's unjammed SNR is quite high.

Specifically, we find that:

(a) Generally random frequency hopping MFSK is more vulnerable to partial-band noise jamming than is conventional FH/MFSK for $M>2$ and $L>1$. However, for certain diversity weighting schemes the increased vulnerability is small enough to justify saying that the two hopping systems achieve comparable performance for $M=2$ or 4. For one combining scheme studied, the self-normalizing receiver, FH/RMFSK performs better than FH/MFSK for $M=2$.

(b) A diversity effect for L hops/symbol is observed for RMFSK using nonlinear hop combining, in the same manner as for MFSK and subject to the same condition that thermal noise is relatively small.

(c) Using optimum diversity values, if thermal noise is negligible ($E_b/N_0 \geq 20$ dB), FH/RMFSK with ideal nonlinear combining can exhibit a nearly exponential dependence upon E_b/N_J, as opposed to an inverse linear one for no diversity; the jamming then is limited to inflicting about a 4 dB loss in system performance. However, this effect is very sensitive to the amount of thermal noise present, since the jammed BER cannot be better than the unjammed error, and the use of diversity tends to degrade the no-jamming performance due to noncoherent combining losses.

(d) Simple nonlinear combining receivers can duplicate the ideal receiver optimum diversity performance with about a one-dB loss when $E_b/N_0 > E_b/N_J$; the hard decision receiver can approach to within 2 dB with a sufficiently high E_b/N_0.

2.0 PRELIMINARY ANALYSES

The general approach to be followed in obtaining the probability of error for the multi-hops/symbol FH/RMFSK communications system under partial band noise jamming is to expand the total probability of error in terms of individual jamming events:

$$P(e) = \sum_{\substack{\text{jamming}\\ \text{events}}} P(e, \text{jamming event})$$

$$= \sum_{\substack{\text{jamming}\\ \text{events}}} \Pr(\text{jamming event})\, P(e|\text{jamming event}),$$

where $P(e|\text{jamming event})$ is the probability of error conditioned upon the occurrence of a particular jamming event. In Section 2.1, we consider a general formulation for this conditional probability, and in Section 2.2 the jamming events and their probabilities are developed. The computational procedures necessary for efficient evaluation of the error probability are discussed in Section 2.3. These analyses and procedures are applied to specific receiver structures beginning in Section 3.

2.1 CONDITIONAL PROBABILITY OF ERROR

The generic form of the receiver to be analyzed for reception of FH/RMFSK is shown in Figure 2.1-1. In effect it is an M-channel receiver with IF frequencies $f_1, f_2, \ldots, f_M$; M pseudorandom sequence generators, assumed to be in synchronism with the transmitter, command the frequency synthesizers used to tune the M channels to B-Hz wide frequency slots, one of which will be occupied by signal energy on a given hop. The message information is

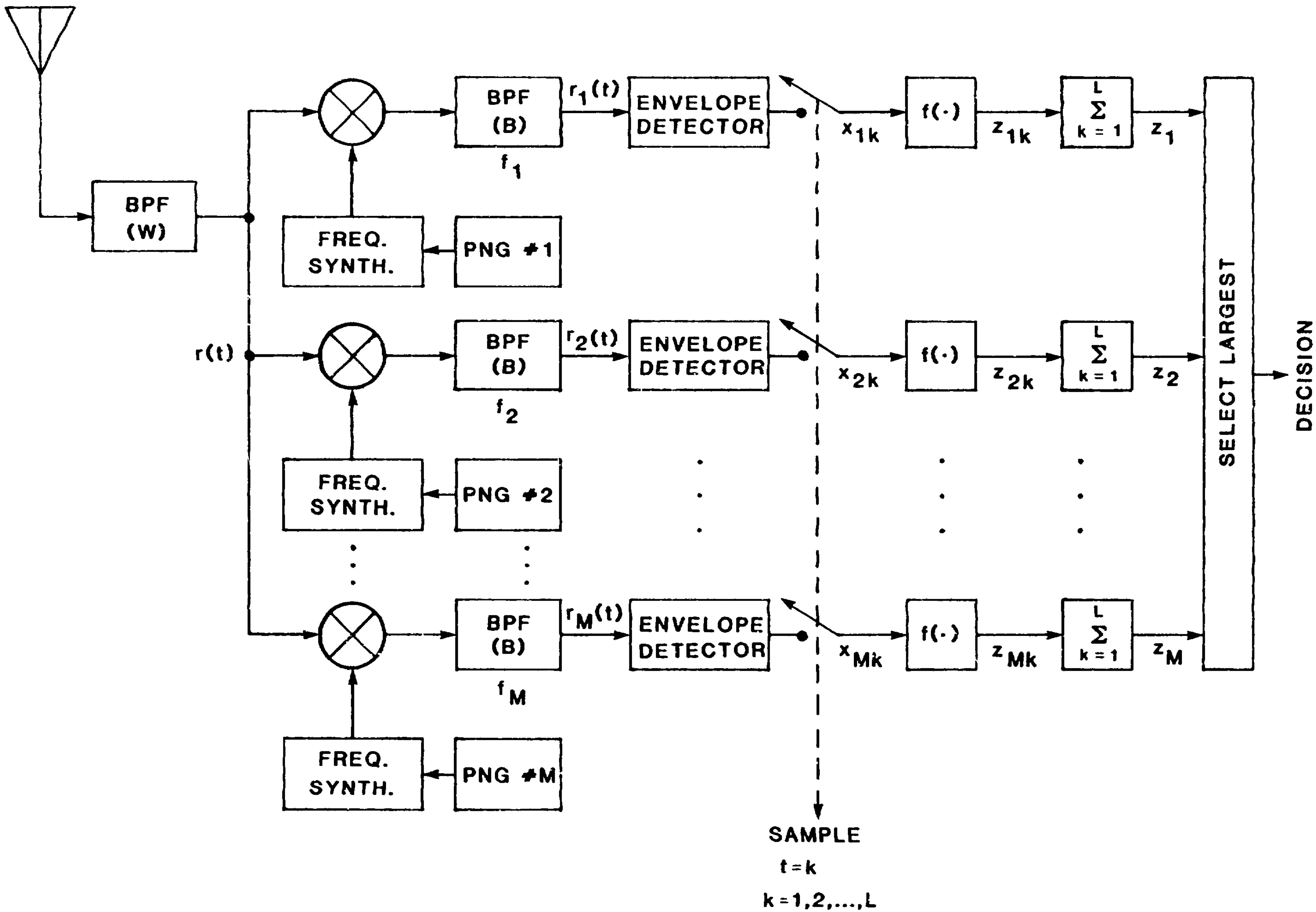


FIGURE 2.1-1 GENERIC RECEIVER FOR FH/RMFSK

conveyed through selection of the (randomly-shifted) IF (symbol) frequency for transmission of signal energy. In order to determine at which symbol frequency the signal is present, each of the IF waveforms $r_1(t), r_2(t), \ldots, r_M(t)$ is first subjected to envelope detection, then sampled before processing through memoryless, possibly nonlinear devices with transfer functions $f(\cdot)$. The outputs of these devices are the per-hop decision statistics $\{z_{mk}\}$, which are accumulated to form the final decision statistics

$$z_m = \sum_{k=1}^{L} z_{mk} \quad , \; m = 1,2,\ldots,M. \tag{2.1-1}$$

In the following subsections, we formulate the probability of error associated with the decision performed by the FH/RMFSK receiver, conditioned upon the possible jamming events.

2.1.1 Assumed Signals, Noise and Jamming.

After dehopping, the received signal is assumed equally likely to be present in any one of the M channels for the entire symbol period $T_s = L\tau$, where τ is the hop period and L is the number of hops per MFSK symbol. Without loss of generality, we assume that the signal with power S is in channel 1, or

$$s(t) = \sqrt{2S}\cos(\omega_1 t + \theta_k), \; (k-1)\tau < t \leq k\tau, \; k = 1, 2, \ldots, L, \tag{2.1-2}$$

where θ_k is an arbitrary carrier phase and $\omega_1 = 2\pi f_1$.

Thermal noise is considered also to be present in each channel, and is assumed to be zero-mean narrowband Gaussian noise with variance $\sigma_N^2 = N_0 B$, where $N_0/2$ is the (two-sided) noise power spectral density and B is the bandwidth of each channel. Thus for no jamming the samples of the M envelope detector outputs on the kth hop are the variables

$$x_{1k} = \left[\left(\sqrt{2S}\,\cos\theta_k + n_{c1k}\right)^2 + \left(\sqrt{2S}\,\sin\theta_k + n_{s1k}\right)^2\right]^{\frac{1}{2}} \tag{2.1-3a}$$

and

$$x_{mk} = \left(n_{cmk}^2 + n_{smk}^2\right)^{\frac{1}{2}},\ m = 2, 3, \ldots, M, \tag{2.1-3b}$$

where n_{cmk}, n_{smk}, $m = 1, 2, \ldots, M$; $k = 1, 2, \ldots, L$, are the independent noise quadrature components in the channels at the sample times $t_k = k\tau$, with

$$E\left(n_{cmk}^2\right) = E\left(n_{smk}^2\right) = \sigma_N^2 = N_0 B,\ \text{for all } m, k. \tag{2.1-4}$$

Because the MFSK symbol slots are hopped independently, none, some, or all of the dehopped channels can be jammed on an individual hop. The possible combinations of such events and their probabilities are discussed in Section 2.2.

When jamming noise is present in a channel, it is assumed to be zero-mean narrowband Gaussian noise with variance $\sigma_J^2 = N_J B/\gamma$, where $N_J/2$ is the (two-sided) noise power spectral density averaged over the system bandwidth; and γ is the fraction of this bandwidth which is jammed. That is,

$$N_J = \frac{J}{W}, \tag{2.1-5}$$

where J is the total jammer power and W is the system bandwidth. When the channels are jammed on the k-th hop, the combination of jamming and thermal noise produces the detector output samples

$$x_{1k} = \left[\left(\sqrt{2S}\,\cos\theta_k + n_{c1k} + j_{c1k}\right)^2 + \left(\sqrt{2S}\,\sin\theta_k + n_{s1k} + j_{s1k}\right)^2\right]^{\frac{1}{2}} \tag{2.1-6a}$$

$$x_{mk} = \left[\left(n_{cmk} + j_{cmk}\right)^2 + \left(n_{smk} + j_{smk}\right)^2\right]^{\frac{1}{2}},\quad m = 2, 3, \ldots, M, \tag{2.1-6b}$$

where j_{cmk}, j_{smk}, $i = 1, 2, \ldots, M$; $k = 1, 2, \ldots, L$, are the independent jamming noise quadrature components in the channels at the sample times, with

$$E\left(j_{cmk}^2\right) = E\left(j_{smk}^2\right) = \sigma_J^2 = N_J B/\gamma, \text{ for all } m, k, \tag{2.1-7}$$

and γ is the fraction of the system bandwidth which is jammed.

In a summary way, we can express the detector output samples by

$$x_{1k} = \sigma_{1k}\left[\left(\sqrt{\frac{2S}{\sigma_{1k}^2}}\cos\theta_k + \mu_{c1k}\right)^2 + \left(\sqrt{\frac{2S}{\sigma_{1k}^2}}\sin\theta_k + \mu_{s1k}\right)^2\right]^{\frac{1}{2}} \tag{2.1-8a}$$

$$x_{mk} = \sigma_{mk}\left(\mu_{cmk}^2 + \mu_{smk}^2\right)^{\frac{1}{2}}, \; m = 2, 3, \ldots, M, \tag{2.1-8b}$$

where μ_{cmk} and μ_{smk} are independent, unit-variance, zero-mean Gaussian random variables, and for channel m on hop k,

$$\sigma_{mk}^2 = \begin{cases} \sigma_N^2 = N_0 B, & \text{unjammed} \\ \sigma_T^2 = \sigma_N^2 + \sigma_J^2 = (N_0 + N_J/\gamma)B, & \text{jammed} \end{cases} \tag{2.1-9a}$$

or, more compactly,

$$\sigma_{mk}^2 = \sigma_N^2 + \nu_{mk}\,\sigma_J^2 . \tag{2.1-9b}$$

In this last equation $\nu_{mk} = 1$ if channel m is jammed on hop k, and $\nu_{mk} = 0$ if not. Thus x_1 is σ_{1k} times a Rician random variable with SNR

$$\rho_k = S/\sigma_{1k}^2 , \tag{2.1-10}$$

and x_{mk}, $m > 1$, is σ_{mk} times a Rayleigh random variable.

2.1.2 Conditional Error Probability Formulation

Assuming equally likely M-ary symbols, we may write the conditional symbol error probability as

$$P_s(e|[\nu]) = P_s(e|[\nu], m_1 \text{ transmitted}) \tag{2.1-11}$$

in which $[\nu]$ is a matrix describing the jamming event for L hops, with elements ν_{mk}.

For M a power of two ($M=2^K$), the conditional bit error probability is obtained from the conditional symbol error probability using the relation

$$P_b(e|[\nu]) = \frac{M/2}{M-1} P_s(e|[\nu]). \tag{2.1-12}$$

Since for M > 2 there are many error events but only one correct decision, it is convenient to write the conditional symbol error probability in terms of the probability of a correct decision as

$$\begin{aligned} P_s(e|[\nu]) &= 1 - P_s(c|m_1,[\nu]) \\ &= 1-\Pr\{z_2 < z_1,\ z_3 < z_1, \ldots, z_M < z_1\}. \end{aligned} \tag{2.1-13}$$

In terms of the pdf's for the statistics, this becomes

$$P_s(e|[\nu]) = 1 - \int_0^\infty d\beta_1 \int_0^{\beta_1} d\beta_2 \cdots \int_0^{\beta_1} d\beta_M\, p_{\underline{z}}(\beta_1,\beta_2,\ldots,\beta_M|[\nu]); \tag{2.1-14}$$

if the decision statistics are independent, then

$$P_s(e|[\nu]) = 1 - \int_0^\infty d\beta\, p_{z_1}(\beta|[\nu]) \prod_{m=2}^{M} \int_0^\beta d\alpha_m\, p_{z_m}(\alpha_m|[\nu]). \tag{2.1-15}$$

For certain receiver structures, the probability distributions of the individual channel statistics $\{z_m\}$ are mutually independent. This relationship causes the conditional probability of error to depend only on the number of hops jammed in each of the M channels, rather than on specific patterns, and greatly reduces the number of distinguishable jamming events.

2.2 ENUMERATION OF DISTINGUISHABLE JAMMING EVENTS

Examination of the conditional error probability expression reveals that the same conditional error will occur for several different values of the fundamental jamming event matrix, $[\nu]$. Therefore, in terms of error probability, there is a number of distinguishable jamming events which is smaller than the 2^{ML} possible values of the $[\nu]$ matrix. It is important to identify and enumerate these distinguishable jamming events in order to take advantage of the savings in computation which will result.

In this section we first identify and enumerate the distinguishable jamming events, then investigate methods for calculating their probabilities.

2.2.1 Definition of Distinguishable Jamming Events.

The conditional error probability, as shown in Section 2.1, is a function of given values of the $[\nu]$ matrix elements ν_{mk}, where $m=1$ to M (the number of MFSK channels) and $k=1$ to L (the number of hops per symbol). Often this function can be written

$$P(e|[\nu]) = P(e|\nu_{11},\nu_{12},\ldots\nu_{1L};\ldots;\nu_{M1},\ldots,\nu_{ML})$$
$$= f\left(\sum_{k=1}^{L} \nu_{1k}, \sum_{k=1}^{L}\nu_{2k},\ldots,\sum_{k=1}^{L} \nu_{mk}\right). \qquad (2.2\text{-}1)$$

Thus, if we define the row sums

$$\ell_m \triangleq \sum_{k=1}^{L} \nu_{mk}, \qquad (2.2\text{-}2)$$

the conditional P(e) is a function only of these sums*, which are to be interpreted as the number of hops jammed in the respective channels. This fact can be expressed by the relation

$$P(e|[\nu]) = f(\ell_1, \ell_2, \ell_3, \ldots, \ell_M)$$

$$\equiv f(\underline{\ell}) \ , \tag{2.2-3}$$

where $\underline{\ell}$ is the vector of ℓ_m components.

Since each ℓ_m can take integer values from 0 to L, there $(L+1)^M$ possible jamming events described by the vector $\underline{\ell}$. This is a considerable savings in numbers of jamming events, as illustrated by Table 2.2-1.

TABLE 2.2-1
NUMBER OF JAMMING EVENTS

M	L	#{[ν] events, 2^{ML}}	#{$\underline{\ell}$ events, $(L+1)^M$}
2	1	4	4
	2	16	9
	3	64	16
	4	256	25
4	1	16	16
	2	256	81
	3	4,096	256
	4	65,536	625
8	1	256	256
	2	65,536	6,561
	3	16,777,216	65,536
	4	4,294,967,296	390,625

2.2.2 Smallest Set of Distinguishable Jamming Events.

A further reduction in the number of distinguishable jamming events

*An exception to this condition results for an ECCM processing scheme studied in Section 4.

results from noticing that permutation of the non-signal channel quantities (ℓ_2 to ℓ_M) does not affect the conditional P(e). That is,

$$\begin{aligned} f(\underline{\ell}) &= f(\ell_1,\ell_2,\ell_3,\ldots,\ell_M) \\ &= f(\ell_1,\ell_M,\ell_3,\ell_4,\ldots,\ell_2) \qquad (2.2\text{-}4) \\ &= f(\ell_1,\ell_5,\ell_{M-1},\ell_2,\ldots,\ell_3) \\ &= \ldots \text{ etc.} \end{aligned}$$

Thus we can restrict our attention to just one permutation of the set of values $\underline{\ell} = (\ell_1,\ell_2,\ldots,\ell_M)$. A convenient way to represent the permutations is the ordered set of numbers

$$\underline{\ell}' \triangleq \{\ell_1,\ell_2,\ell_3,\ldots,\ell_M : \ell_2 \leq \ell_3 < \ldots \leq \ell_M\}. \qquad (2.2\text{-}5)$$

There are, from Appendix B.3,

$$\sum_{\ell_1=0}^{L} \sum_{\ell_M=0}^{M} \sum_{\ell_{M-1}=0}^{\ell_M} \cdots \sum_{\ell_3=0}^{\ell_4} \sum_{\ell_2=0}^{\ell_3} = (L+1)\binom{M-1+L}{M-1} \qquad (2.2\text{-}6)$$

such ordered $\underline{\ell}'$ vectors, which represent the minimum number of distinguishable events. Example values are given in Table 2.2-2.

TABLE 2.2-2
MINIMUM NUMBER OF DISTINGUISHABLE EVENTS

M	L	#{$\underline{\ell}'$ events}
2	1	4
	2	9
	3	16
	4	25
4	1	8
	2	30
	3	80
	4	175
8	1	16
	2	108
	3	480
	4	1,650

Each of the distinguishable jamming events represents a certain number of events with identical jamming effects. The number of $\underline{\ell}$ vectors thereby represented by a particular ordered vector $\underline{\ell}_1'$ is

$$\#(\underline{\ell} \leftrightarrow \underline{\ell}_1') = \binom{M-1}{n_0, n_1, \ldots, n_L} \tag{2.2-7a}$$

$$= \frac{(M-1)!}{n_0! n_1! \ldots n_L!} \tag{2.2-7b}$$

where

$$n_\ell = \text{number of } \ell_m \text{ which equal } \ell;\ \ell=0,1,\ldots,L;\ m > 1$$

and we have

$$\sum_{\ell=0}^{L} n_\ell = M-1. \tag{2.2-7d}$$

For example, for M=8 and L=6, the number of jamming event vectors $\underline{\ell}$ represented by the ordered vector $\underline{\ell}' = (\ell_1;0,0,2,3,3,4,5)$ is

$$\binom{7}{2,0,1,2,1,1,0} = \frac{7!}{2!0!1!2!1!1!0!} = 1260. \tag{2.2-8}$$

As a check on this enumeration, we find that the total number of $\underline{\ell}$ jamming events is given by

$$\#(\underline{\ell}) = \sum_{\ell_1=0}^{L} \sum_{\ell_M=0}^{L} \sum_{\ell_{M-1}=0}^{\ell_M} \sum_{\ell_3=0}^{\ell_4} \sum_{\ell_2=0}^{\ell_3} \binom{M-1}{n_0, n_1, \ldots, n_L}$$

$$= (L+1) \sum_{\ell_M=0}^{L} \cdots \sum_{\ell_2=0}^{\ell_3} \binom{M-1}{n_0, n_1, \ldots, n_L}. \tag{2.2-9}$$

It can be shown (see Appendix B) that the summation in (2.2-9) is equal to $(L+1)^{M-1}$. Thus the total number of $\underline{\ell}$ vectors computed by (2.2-9) is $(L+1)^M$, which agrees with our previous enumeration.

2.2.3 Jamming Event Probabilities, Single Hop.

Given a jamming event described by the vector $\underline{\ell}$, what is the probability of the event under the random hopping scheme and partial-band noise jamming? To find this answer, we first consider the case of one hop per MFSK symbol, or L=1.

The jammer spectrum is assumed to be flat, with one-sided power spectral density $J/\gamma W$, where γ is the fraction of the system bandwidth occupied by the jammer. There are $N=W/B$ possible symbol frequency slots, and it is assumed that M of these slots are assigned randomly to the MFSK symbol on each hop. At the same time the number of slots containing jamming power is

$$q \triangleq \gamma N, \tag{2.2-10}$$

assumed to be an integer. That is, $\gamma = q/N$, with q an integer.

The probability that n of the M symbol slots are jammed on a given hop is

$$\pi_n = \frac{q}{N} \cdot \frac{q-1}{N-1} \cdots \frac{q-n+1}{N-n+1} \cdot \frac{N-q}{N-n} \cdots \frac{N-q-M+n+1}{N-M+1}$$

$$= \frac{\binom{N-M}{q-n}}{\binom{N}{q}}, \quad n=0,1,2,\ldots,\min(q,M). \tag{2.2-11}$$

Note that the probability π_n is valid for several different jamming events since

$$n_k = \sum_{m=1}^{M} \nu_{mk} \tag{2.2-12}$$

on the k-th hop. In fact there are $\binom{M}{n}$ jamming events for L=1 which have probability π_n. Thus

$$\sum_{n=0}^{L} \binom{M}{n} \pi_n = 1, \tag{2.2-13}$$

as required.

In terms of distinguishable jamming events, we differentiate between whether the signal channel is jammed ($\nu_{1k} = 1$) or not ($\nu_{1k} = 0$), and describe single-hop jamming events by the pair of numbers (ν_{1k}, r_k) with

$$r_k \triangleq \sum_{m=2}^{M} \nu_{mk} , \tag{2.2-14}$$

the number of non-signal channels jammed. We have

$$\Pr\{\nu_{1k}, r_k\} = \pi(\nu_{1k} + r_k) \equiv \pi_n, \; n = \nu_{1k} + r_k \tag{2.2-15}$$

and
$$\sum_{\nu_{1k}=0}^{1} \sum_{r_k=0}^{M-1} \binom{M-1}{r_k} \pi(\nu_{1k} + r_k) = 1. \tag{2.2-16}$$

2.2.4 Jamming Event Probabilities For Multiple Hops, Characteristic Function Method.

For L > 1 hops per symbol, as discussed in Sections 2.2.1 and and 2.2.2, the distinguishable jamming events are described by the vectors $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$. We note that

$$\underline{\ell} = \sum_{k=1}^{L} \underline{\nu}_k \tag{2.2-17}$$

where $\underline{\nu}_k = (\nu_{1k}, \nu_{2k}, \ldots \nu_{mk})$ is the vector whose elements are the k-th column of the matrix $[\nu]$ of fundamental jamming events. Since the hopping pattern is assumed to be independent from hop to hop, we may treat $\underline{\ell}$ as the sum of identically distributed discrete random vectors, and find the probabilities of the $\underline{\ell}$ jamming events from the characteristic function of the $\underline{\nu}$ jamming events.

The characteristic function of any one of the random vectors $\underline{\nu}_k$ is given by

$$\begin{aligned}
\phi_{\underline{\nu}_k}(\underline{\mu};M) &= E\{\exp[j\underline{\nu}_k \cdot \underline{\mu}]\} \\
&= E\{\exp[j\mu_1\nu_{1k} + j\mu_2\nu_{2k} + \ldots + j\mu_M\nu_{Mk}]\} \\
&= \pi_0 + \pi_1(e^{j\mu_1} + e^{j\mu_2} + \ldots + e^{j\mu_M}) \\
&\quad + \pi_2(e^{j\mu_1 + j\mu_2} + \ldots + e^{j\mu_{M-1} + j\mu_M}) \\
&\quad + \ldots \\
&\quad + \pi_M e^{j\mu_1 + j\mu_2 + \cdots + j\mu_M} .
\end{aligned} \tag{2.2-18}$$

For example, for M=2 and 4 the characteristic functions for one hop are

$$\phi_{\underline{\nu}}(\underline{\mu};2) = \pi_0 + \pi_1 e^{j\mu_1} + \pi_1 e^{j\mu_2} + \pi_2 e^{j\mu_1+j\mu_2} \tag{2.2-19}$$

and

$$\begin{aligned}\phi_{\underline{\nu}}(\underline{\mu};4) = \pi_0 &+ \pi_1 e^{j\mu_1} + \pi_1 e^{j\mu_2} + \pi_1 e^{j\mu_3} + \pi_1 e^{j\mu_4} \\ &+ \pi_2 e^{j\mu_1+j\mu_2} + \pi_2 e^{j\mu_1+j\mu_3} + \pi_2 e^{j\mu_1+j\mu_4} \\ &+ \pi_2 e^{j\mu_2+j\mu_3} + \pi_2 e^{j\mu_2+j\mu_4} + \pi_2 e^{j\mu_3+j\mu_4} \\ &+ \pi_3 e^{j\mu_1+j\mu_2+j\mu_3} + \pi_3 e^{j\mu_1+j\mu_2+j\mu_4} \\ &+ \pi_3 e^{j\mu_1+j\mu_3+j\mu_4} + \pi_3 e^{j\mu_2+j\mu_3+j\mu_4} \\ &+ \pi_4 e^{j\mu_1+j\mu_2+j\mu_3+j\mu_4} \;.\end{aligned} \tag{2.2-20}$$

In this characteristic function, there are 2^M terms, one for each of the events described by $\underline{\nu}_k$.

The characteristic function for $\underline{\ell}$ is simply that of $\underline{\nu}_k$, raised to the L-th power:

$$\phi_{\underline{\ell}}(\underline{\mu};M) = \left[\phi_{\underline{\nu}}(\underline{\mu};M)\right]^L . \tag{2.2-21}$$

For example, for M=2,

$$\begin{aligned}\phi_{\underline{\ell}}(\underline{\mu};2) &= \left[\phi_{\underline{\nu}}(\underline{\mu};2)\right]^L \\ &= \sum \binom{L}{n_0,n_1,n_2,n_3} \pi_0^{n_0}\pi_1^{n_1+n_2}\pi_2^{n_3} \\ &\quad \times \exp\{j(n_1+n_3)\mu_1 + j(n_2+n_3)\mu_2\},\end{aligned} \tag{2.2-22a}$$

where the summation is over all (n_0,n_1,n_2,n_3) such that

$$\sum_{q=0}^{2^M-1} n_q = L \; . \tag{2.2-22b}$$

The discrete probability density function (pdf) for $\underline{\ell}$ is the inverse Fourier transform of $\phi_{\underline{\ell}}(\underline{\mu};M)$. Again, for M=2, the pdf $p(\underline{\ell};M,L)$ for $\underline{\ell}$ is

$$p(\underline{\ell};2,L) = \sum \binom{L}{n_0,n_1,n_2,n_3} \pi_0^{n_0} \pi_1^{n_1+n_2} \pi_2^{n_3} \times \delta(n_1+n_3-\ell_1)\;\delta(n_2+n_3-\ell_2), \tag{2.2-23}$$

which can be used to find the individual $\underline{\ell}$ vector probabilities

$$\Pr\{\underline{\ell};2,L\} = \sum_{n=0}^{L} \binom{L}{n,L-\ell_2-n,L-\ell_1-n,\ell_1+\ell_2+n-L} \times \pi_0^{n} \pi_1^{2L-\ell_1-\ell_2-2n} \pi_2^{\ell_1+\ell_2-L+n} \; . \tag{2.2-24}$$

In (2.2-24) it is realized that the combinatorial factor is zero if any of its parameters is negative. As an example for L=3 and M=2,

$$\begin{aligned} \Pr\{\ell_1 = 1,\ell_2 = 2;2,3\} &= \binom{3}{0,1,2,0} \pi_1^3 + \binom{3}{1,0,1,1} \pi_0\pi_1\pi_2 \\ &= 3\pi_1^3 + 6\pi_0\pi_1\pi_2 \; . \end{aligned} \tag{2.2-25}$$

An aid to checking our calculations is the fact that each $\underline{\ell}$ event corresponds to

$$\binom{L}{\ell_1}\binom{L}{\ell_2}\cdots\binom{L}{\ell_M} \tag{2.2-26}$$

$[\nu]$ matrix events. In (2.2-25) therefore, there are $\binom{3}{1}\binom{3}{2} = 9$ terms.

A complete table of M=2 jamming event probabilities for L=1 to 4 is given in Table 2.2-3, and an equation for computing these probabilities for M=4 is included as Table 2.2-4.

2.2.5 Jamming Event Probabilities For Multiple Hops, Convolution Method.

Since $\underline{\ell}$ is the sum of L independent random event vectors $\underline{\nu}_k$, the pdf for $\underline{\ell}$ is the L-fold convolution of the pdf for $\underline{\nu}_k$:

$$p(\underline{\ell};M,L) = p(\underline{\nu}_1;M)*p(\underline{\nu}_2;M)*\ldots*p(\underline{\nu}_L;M), \tag{2.2-27}$$

or

$$\Pr\{\underline{\ell};M,L\} = \sum_{\underline{\nu}_1}\sum_{\underline{\nu}_2}\cdots\sum_{\underline{\nu}_L} \Pr\{\underline{\nu}_1;M\}\ \ldots\ \Pr\{\underline{\nu}_L;M\}\ \delta(\underline{\ell} - \sum_{k=1}^{L}\underline{\nu}_k). \tag{2.2-28}$$

Figure 2.2-1 illustrates a programming approach for calculating this equation indirectly.

TABLE 2.2-3

JAMMING EVENT PROBABILITIES FOR M=2, L=1 TO 4

$\underline{\ell}$	L=1	L=2	L=3	L=4
0,0	π_0	π_0^2	π_0^3	π_0^4
0,1	π_1	$2\pi_0\pi_1$	$3\pi_0^2\pi_1$	$4\pi_0^3\pi_1$
1,0	π_1	$2\pi_0\pi_1$	$3\pi_0^2\pi_1$	$4\pi_0^3\pi_1$
1,1	π_2	$2\pi_0\pi_2 + 2\pi_1^2$	$3\pi_0^2\pi_2 + 6\pi_0\pi_1^2$	$4\pi_0^3\pi_2 + 12\pi_0^2\pi_1^2$
0,2		π_1^2	$3\pi_0\pi_1^2$	$6\pi_0^2\pi_1^2$
1,2		$2\pi_1\pi_2$	$6\pi_0\pi_1\pi_2 + 3\pi_1^3$	$12\pi_0^2\pi_1\pi_2 + 12\pi_0\pi_1^3$
2,0		π_1^2	$3\pi_0\pi_1^2$	$6\pi_0^2\pi_1^2$
2,1		$2\pi_1\pi_2$	$6\pi_0\pi_1\pi_2 + 3\pi_1^3$	$12\pi_0^2\pi_1\pi_2 + 12\pi_0\pi_1^3$
2,2		π_2^2	$3\pi_0\pi_2^2 + 6\pi_1^2\pi_2$	$6\pi_0^2\pi_2^2 + 24\pi_0\pi_1^2\pi_2 + 6\pi_1^4$
0,3			π_1^3	$4\pi_0\pi_1^3$
1,3			$3\pi_1^2\pi_2$	$12\pi_0\pi_1^2\pi_2 + 4\pi_1^4$
2,3			$3\pi_1\pi_2^2$	$12\pi_0\pi_1\pi_2^2 + 12\pi_1^3\pi_2$
3,0			π_1^3	$4\pi_0\pi_1^3$
3,1			$3\pi_1^2\pi_2$	$12\pi_0\pi_1^2\pi_2 + 4\pi_1^4$
3,2			$3\pi_1\pi_2^2$	$12\pi_0\pi_1\pi_2^2 + 12\pi_1^3\pi_2$
3,3			π_2^3	$4\pi_0\pi_2^3 + 12\pi_1^2\pi_2^2$
0,4				π_1^4
1,4				$4\pi_1^3\pi_2$
2,4				$6\pi_1^2\pi_2^2$
3,4				$4\pi_1\pi_2^3$
4,0				π_1^4
4.1				$4\pi_1^3\pi_2$
4,2				$6\pi_1^2\pi_2^2$
4,3				$4\pi_1\pi_2^3$
4,4				π_2^4

$$\Pr(\underline{\ell}) = \sum_{n=0}^{L} \binom{L}{n, L-\ell_2-n, L-\ell_1-n, \ell_1+\ell_2+n-L} \pi_0^n \, \pi_1^{2L-\ell_1-\ell_2-n} \, \pi_2^{\ell_1+\ell_2+n-L}$$

TABLE 2.2-4

PROBABILITIES OF $\underline{\ell}$ JAMMING EVENTS FOR M = 4

$$\Pr\{\underline{\ell}\,;\,4,\,L\} = \underbrace{\sum_{n_0=0}^{L}\sum_{n_1=1}^{L}\cdots\sum_{n_{15}=0}^{L}}_{\text{CONSTRAINTS}} \frac{L!}{n_0!n_1!\,\ldots\,n_{15}!}\,\pi_0^{n_0}\,\pi_1^{n_1+n_2+n_4+n_8}\,\pi_2^{n_3+n_5+n_6+n_9+n_{10}+n_{12}}\,\pi_3^{n_7+n_{11}+n_{13}+n_{14}}\,\pi_4^{n_{15}}$$

CONSTRAINTS : $n_1 + n_3 + n_5 + n_7 + n_9 + n_{11} + n_{13} + n_{15} = \ell_1$

$n_2 + n_3 + n_6 + n_7 + n_{10} + n_{11} + n_{14} + n_{15} = \ell_2$

$n_4 + n_5 + n_6 + n_7 + n_{12} + n_{13} + n_{14} + n_{15} = \ell_3$

$n_8 + n_9 + n_{10} + n_{11} + n_{12} + n_{13} + n_{14} + n_{15} = \ell_4$

$$\sum_{i=0}^{15} n_i = L$$

Input: M, L, π_0 to π_M

Initialize all $\Pr\{\underline{\ell}\} = 0$

$I_1 = 0$ to 2^M-1

ν_{m1} = mth bit of I_1, $m = 1$ to M

$w_1 = \sum_m \nu_{m1}$

$I_2 = 0$ to 2^M-1

ν_{m2} = mth bit of I_2, $m = 1$ to M

$w_2 = \sum_m \nu_{m2}$

.
.
.

$I_L = 0$ to 2^M-1

ν_{mL} = mth bit of I_L, $m = 1$ to M

$w_L = \sum_m \nu_{mL}$

$\ell_m = \sum_k \nu_{mk}$, $m = 1$ to M

$I(\underline{\ell}) = \sum_m \ell_m L^{m-1}$

Increment $\Pr\{\underline{\ell}\} = \Pr\{I(\underline{\ell})\}$ by $\pi(w_1)\,\pi(w_2)\ldots\pi(w_L)$

skip if any $\ell_m > \ell_{m+1}$, $m = 2$ to $M-1$ to get ordered vector output

FIGURE 2.2-1 PROGRAM STRUCTURE FOR CALCULATING JAMMING EVENT PROBABILITIES INDIRECTLY

This can also be done iteratively, using the fact that

$$\Pr\{\underline{\ell};M,L\} = \sum_{\underline{\alpha}} \sum_{\underline{\nu}_L} \Pr\{\underline{\alpha};M,L-1\} \Pr\{\underline{\nu}_L;M\}\, \delta(\underline{\ell}-\underline{\alpha}-\underline{\nu}_L)$$

$$= \sum_{\underline{\nu}_L} \Pr\{\underline{\ell}-\underline{\nu}_L;M,L-1\} \Pr\{\underline{\nu}_L;M\}. \tag{2.2-29}$$

To accomplish the vector additions needed for the convolution, we may encode the M x 1 vectors $\underline{\ell}$ and $\underline{\nu}_L$ into a number using the form

$$I(\underline{\ell}) = \ell_1 + b\ell_2 + b^2\ell_3 + .. + b^{M-1}\ell_M, \tag{2.2-30a}$$

where $b \geq L$ is an integer base number, supporting the relationship

$$I(\underline{\ell}_1 + \underline{\ell}_2) = I(\underline{\ell}_1) + I(\underline{\ell}_2). \tag{2.2-30b}$$

In this manner the convolution in (2.2-29) can be done using

$$\Pr\{\underline{\ell};M,L\} = \sum_{I(\underline{\alpha})} \sum_{I(\underline{\nu}_L)} \Pr\{I(\underline{\alpha});M,L-1\} \Pr\{I(\underline{\nu}_L);M\} \times \delta[I(\underline{\ell})-I(\underline{\alpha})-I(\underline{\nu}_L)].$$

$$= \sum_{I(\underline{\nu}_L)} \Pr\{I(\underline{\ell})-I(\underline{\nu}_L);M,L-1\}\Pr\{I(\underline{\nu}_L);M\}. \tag{2.2-31}$$

2.3 TOTAL PROBABILITY OF ERROR

In terms of the conditional probability of symbol error, given a jamming event defined by the vector $\underline{\ell}$, and the probabilities of the jamming events, we now can write the total probability of error as

$$P_s(e) = \sum_{\underline{\ell}} \Pr\{\underline{\ell}\}\, P_s(e|\underline{\ell})$$

$$= \sum_{\underline{\ell}'} \binom{M-1}{n_0, n_1, \ldots, n_L} \Pr\{\underline{\ell}'\}\, P_s(e|\underline{\ell}'). \tag{2.3-1}$$

This formulation utilizes the fact that $\underline{\ell}$ vectors formed by permutations of the nonsignal channel elements $\{\ell_m, m>1\}$ of the ordered $\underline{\ell}'$ vector are equiprobable, and that the conditional error probability is the same for each permutation.

2.4 FH/RMFSK HARD DECISION ANALYSIS

In addition to studying the performance of L hops/symbols soft decision receivers for various FH/RMFSK combining schemes, we shall calculate the performance when L M-ary hard decisions are combined to produce a final symbol decision. This configuration is in itself a form of ECCM processing, as will be shown in later sections.

2.4.1 Formulation of Error Probability.

Under an M-ary hard decision approach, shown previously as Figure 1.3-3, on the kth hop the decision variables $\{z_{mk}\}$ are compared to find the largest; the signal is assumed to be present in the channel with the largest decision variable. The per hop or "hard" symbol decision can be thought of as selecting one of M vectors $\{\underline{D}_1, \underline{D}_2, \ldots, \underline{D}_M\}$, where

$$\underline{D}_m = (D_{1m}, D_{2m}, \ldots, D_{Mm}) \tag{2.4-1a}$$

with

$$D_{im} = \begin{cases} 1, & i=m \\ 0, & i\neq m \end{cases} \qquad m=1,2,\ldots,M. \tag{2.4-1b}$$

The hard symbol decision on the kth hop then can be expressed as the vector

$$\underline{d}_k = (d_{1k}, d_{2k}, \ldots, d_{Mk}) \triangleq \underline{D}_{m^*} \tag{2.4-2a}$$

where m* is chosen such that

$$z_{m^*k} = \max_m (z_{mk}). \tag{2.4-2b}$$

The components of the per-hop decision vector $\underline{d}_k$ are accumulated over the L hops of the symbol to produce the final, discrete decision variables

$$d_m = \sum_{k=1}^{L} d_{mk} \; ; \; m = 1,2,\ldots,M; \tag{2.4-3a}$$

or, in vector notation, the final discrete-valued decision vector

$$\underline{d} = \sum_{k=1}^{L} \underline{d}_k \, . \tag{2.4-3b}$$

The error probability can be formulated as

$$\begin{aligned} P_s(e) &= 1 - P_s(\text{correct decision} \equiv C) \\ &= 1 - \sum_{\underline{\ell}} P_s(C|\underline{\ell}) \, Pr\{\underline{\ell}\} \\ &= 1 - \sum_{\underline{\ell}} Pr\{\underline{\ell}\} \sum_{\underline{n} \,\varepsilon\, \Omega_c} Pr\{\underline{d} = \underline{n}|\underline{\ell}\}, \end{aligned} \tag{2.4-4}$$

where $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$ describes the jamming events, and Ω_c is the set of decision vectors which produce a correct decision.

Since the components of $\underline{d}$ are discrete-valued, there exists the possibility of a tie between the signal channel's final decision variable value and that of one or more non-signal channels. Thus the error expression (2.4-4) should be modified to

$$P_s(e) = 1 - \sum_{\underline{\ell}} Pr\{\underline{\ell}\} \sum_{\underline{n} \varepsilon \Omega_c} h(\underline{n}) \, Pr\{\underline{d} = \underline{n}|\underline{\ell}\} \, , \tag{2.4-5a}$$

where $$h(\underline{n}) = (\#\text{channels equal to maximum})^{-1} \, , \tag{2.4-5b}$$

assuming that a randomized decision is made when there is a tie. For example, if three channels (including the signal channel) are equal to the maximum value, then $h(\underline{n}) = 1/3$.

2.4.2 Explicit Form of Error Probability.

Interchanging the summations in the probability of a correct symbol decision gives

$$P_s(c) = \sum_{\underline{n}\in\Omega_c} h(\underline{n}) \sum_{\underline{\ell}} \Pr\{\underline{\ell}\} \Pr\{\underline{d} = \underline{n} \mid \underline{\ell}\} . \tag{2.4-6}$$

Now, since the jamming event vector $\underline{\ell}$ is related to the jamming event matrix $[\nu]$ by

$$\underline{\ell} = \sum_{k=1}^{L} \underline{\nu}_k , \tag{2.4-7}$$

where $\underline{\nu}_k$ is the kth column of $[\nu]$, the summation over $\underline{\ell}$ can be replaced by

$$\begin{aligned} &\sum_{\underline{\nu}_1} \sum_{\underline{\nu}_2} \cdots \sum_{\underline{\nu}_L} \Pr\{[\nu]\} \Pr\{\underline{d} = \underline{n} \mid [\nu]\} \\ &= \sum_{\underline{\nu}_1} \Pr\{\underline{\nu}_1\} \sum_{\underline{\nu}_2} \Pr\{\underline{\nu}_2\} \cdots \sum_{\underline{\nu}_L} \Pr\{\underline{\nu}_L\} \Pr\{\underline{d}=\underline{n} \mid \underline{\nu}_1, \underline{\nu}_2, \ldots, \underline{\nu}_L\} . \end{aligned} \tag{2.4-8}$$

In this expression we use the fact that the $\{\underline{\nu}_k\}$ are statistically independent. It is also true that the individual hop decisions $\{\underline{d}_k\}$ are independent, so we can expand (2.4-8) further to obtain

$$\begin{aligned} &\sum_{\underline{\nu}_1} \Pr\{\underline{\nu}_1\} \Pr\{\underline{d}_1 \mid \underline{\nu}_1\} \sum_{\underline{\nu}_2} \Pr\{\underline{\nu}_2\} \Pr\{\underline{d}_2 \mid \underline{\nu}_2\} \\ &\cdots \sum_{\underline{\nu}_L} \Pr\{\underline{\nu}_L\} \Pr\{\underline{d}_L \mid \underline{\nu}_L\} \, \delta\Big(\sum_k \underline{d}_k, \underline{n}\Big) , \end{aligned} \tag{2.4-9}$$

where $\delta(\underline{a},\underline{b})$ is a vector version of the Kronecker delta function:

$$\delta(\underline{a},\underline{b}) \triangleq \begin{cases} 1 & \text{if } \underline{a} = \underline{b} \\ 0 & \text{if } \underline{a} \neq \underline{b} . \end{cases} \tag{2.4-10}$$

Recognizing that the sums over the individual $\{\underline{\nu}_k\}$ are simply averages, we can write

$$P_s(e) = 1 - \sum_{\underline{n} \in \Omega_c} h(n) \Pr\{\underline{d}_1\} \Pr\{\underline{d}_2\} \ldots \Pr\{\underline{d}_L\} \delta(\underline{d},\underline{n}), \tag{2.4-11}$$

where $\Pr\{\underline{d}_k\}$ for the kth hop is the average of the (discrete) probability distribution for the hop decision over the jamming events for the kth hop, $\underline{\nu}_k$. Assuming without loss of generality that the first channel is the signal channel, these averages are

$$\Pr\{\underline{d}_k = D_1\} \equiv p = 1 - P_s\left(e;\gamma,\frac{1}{L}\cdot\frac{E_s}{N_0},\frac{1}{L}\cdot\frac{E_s}{N_J}, L=1\right) \triangleq 1 - P_1 \tag{2.4-12a}$$

$$\begin{aligned} \Pr\{\underline{d}_k = \underline{D}_m\} &\equiv q = (1-p)/(M-1) \\ &= P_1/(M-1), \ m \geq 2. \end{aligned} \tag{2.4-12b}$$

Finally, using the function

$$H(\underline{n}) = \begin{cases} 1 & \text{if } \underline{n} \in \Omega_c \\ 0 & \text{if } \underline{n} \in \Omega_c' \end{cases} \tag{2.4-13}$$

as a "mask" for the correct decisions, we can write (2.4-11) more explicitly as

$$P_s(e) = 1 - \underbrace{\sum_{n_1=0}^{L} \sum_{n_2=0}^{L} \cdots \sum_{n_M=0}^{L}}_{\sum_i n_i = L} H(\underline{n})h(\underline{n}) \binom{L}{n_1,n_2,\ldots,n_M} p^{n_1} q^{L-n_1}$$

$$= 1 - \sum_{n_1=0}^{L} \sum_{n_2=0}^{n_1} \cdots \sum_{n_M=0}^{n_1} h(\underline{n}) \binom{L}{n_1,n_2,\ldots,n_M} p^{n_1} q^{L-n_1} \delta(\sum_i n_i, L). \tag{2.4-14}$$

2.4.2.1 Special case: $L = 2$.

For $L = 2$, (2.4-14) reduces to

$$\begin{aligned} P_s(e;\gamma,\frac{E_s}{N_0},\frac{E_s}{N_J}, L=2) &= 1 - p^2 - (M-1)pq \\ &= 1 - p^2 - p\,P_1 = P_1 \\ &= P_s(e;\gamma, \frac{1}{2}\frac{E_s}{N_0}, \frac{1}{2}\frac{E_s}{N_J}, L=1); \end{aligned} \tag{2.4-15}$$

that is, the hard decision receiver is uniformly 3 dB worse for $L = 2$ than for $L = 1$, for any value of M.

2.4.2.2 Special case: $L = 3$.

For $L = 3$, (2.4-14) reduces to

$$P_s(e;\gamma, \frac{E_s}{N_0}, \frac{E_s}{N_J}, L=3) = \frac{1}{M-1} P_1^2(2M - 1 - MP_1), \tag{2.4-16a}$$

with

$$P_1 = P_s(e;\gamma, \frac{1}{3}\frac{E_s}{N_0}, \frac{1}{3}\frac{E_s}{N_0}, L=1). \tag{2.4-16b}$$

2.4.2.3 Special case: L = 4.

For L = 4, (2.4-14) reduces to

$$P_s\left(e;\gamma,\frac{E_s}{N_0},\frac{E_s}{N_J}, L=3\right) = \frac{1}{M-1} P_1^2 \left[3 + \frac{P_1}{M-1}(3M^2-9M+4) - 2M \cdot \frac{M-2}{M-1} P_1^2\right] \quad (2.4\text{-}17a)$$

with

$$P_1 = P_s\left(e;\gamma, \frac{1}{4}\frac{E_s}{N_0}, \frac{1}{4}\frac{E_s}{N_J}, L=1\right). \quad (2.4\text{-}17b)$$

Note that for M = 2, (2.4-16a) and (2.4-17a) both give $P_s = P_1^2(3-2P_1)$; this implies that the L = 4 hard decision performance is uniformly $10 \log_{10} (\frac{4}{3})$ = 1.25 dB worse than that for L = 3 when M = 2.

2.4.2.4 Special case: M = 2.

For M = 2, (2.3-14) reduces to

$$P_b(e) = \begin{cases} 1 - \sum\limits_{n_1=\frac{L}{2}+1}^{L} \binom{L}{n_1} p^{n_1} q^{L-n_1} - \frac{1}{2}\binom{L}{L/2} p^{L/2} q^{L/2}, & L \text{ even} \\ 1 - \sum\limits_{n_1=(L+1)/2}^{L} \binom{L}{n_1} p^{n_1} q^{L-n_1}, & L \text{ odd.} \end{cases} \quad (2.4\text{-}18)$$

3.0 FH/RMFSK PERFORMANCE USING SQUARE-LAW LINEAR COMBINING RECEIVER

In this section we consider the case of the generic receiver shown in Figure 2.2-1 when the envelope samples are processed using the function

$$f(x_{mk}) = x_{mk}^2 \equiv z_{mk}. \tag{3.0-1}$$

That is, the decision statistics $\{z_m\}$ are the unweighted linear combinations or sums of samples of the squared envelopes in each channel over multiple (L) hops. For non-hopping systems, this receiver is known to give good performance when the signal is subject to Rayleigh fading, L being the order of diversity which can be chosen to optimize performance for a given SNR.

3.1 ERROR PROBABILITY ANALYSIS

In Section 2.2 it was shown that the envelope samples $\{x_{mk}\}$ are σ_{mk} times a Rician random variable for the signal channel (m=1) and σ_{mk} times a Rayleigh random variable for the non-signal channels (m>1), where the value of σ_{mk} depends upon whether the channel is jammed or not. Therefore, for the square-law linear combining FH/RMFSK receiver, the hop decision statistics, which are the squares of the envelope samples, are σ_{mk}^2 times chi-squared random variables with two degrees of freedom. For the signal channel the noncentrality parameter is

$$\lambda_k = 2\rho_k = 2S/\sigma_{1k}^2 . \tag{3.1-1}$$

The probability density function (pdf) for the signal channel samples is

$$p_{z_{1k}}\left(\alpha;\rho_k,\sigma_{1k}^2\right) = \frac{1}{2\sigma_{1k}^2} \exp\left\{-\rho_k - \frac{\alpha}{2\sigma_{1k}^2}\right\} I_0\left(\sqrt{2\rho_k\alpha/\sigma_{1k}^2}\right), \tag{3.1-2}$$

while that for the non-signal channel samples is

$$p_{z_{mk}}\left(\alpha;\sigma_{mk}^2\right) = \frac{1}{2\sigma_{mk}^2} \exp\left\{-\frac{\alpha}{2\sigma_{mk}^2}\right\}, \; m>1. \tag{3.1-3}$$

3.1.1 Distribution of the Decision Statistics.

The accumulated decision statistics $\{z_m\}$ can be expressed as

$$z_1 = \sum_{k=1}^{L} \sigma_{1k}^2 \chi^2(2;\lambda_k) \tag{3.1-4a}$$

and

$$z_m = \sum_{k=1}^{L} \sigma_{mk}^2 \chi^2(2) \tag{3.1-4b}$$

where $\chi^2(n)$ denotes a chi-squared random variable with n degrees of freedom and $\chi^2(n;\lambda)$ denotes a noncentral chi-squared random variable with n degrees of freedom and a noncentrality parameter λ. It is well known that sums of equally weighted chi-squared variables yield chi-squared variables:

$$\sum_{k=1}^{L} \chi^2(n_k;\lambda_k) = \chi^2\left(\sum_{k=1}^{L} n_k;\ \sum_{k=1}^{L}\lambda_k\right). \tag{3.1-5}$$

This fact can be applied to (3.1-4) by recalling that for a given jamming event, ℓ_m out of L hops in a given channel are jammed. Thus

$$z_1 = \sigma_N^2\chi^2[2(L-\ell_I);\ 2(L-\ell_I)S/\sigma_N^2] + \sigma_T^2\chi^2\left(2\ell_1;\ 2\ell_1 S/\sigma_T^2\right) \tag{3.1-6a}$$

and

$$z_m = \sigma_N^2\, \chi^2\,[2(L-\ell_m)] + \sigma_T^2\ \chi^2(2\,\ell_m), \quad m > 2. \tag{3.1-6b}$$

In Appendix A it is shown that the pdf for the normalized variable $u_1 = z_1/\sigma_N^2$ is

$$p_{u_1}(\alpha) = \begin{cases} p_{\chi^2}(\alpha;\ 2L,\ 2L\rho_N) & \ell_1 = 0; \quad (3.1\text{-}7a) \\ \frac{1}{K}\, p_{\chi^2}(\alpha/K;\ 2L,\ 2L\rho_T), & \ell_1 = L; \quad (3.1\text{-}7b) \\ \sum_{n=0}^{\infty} c_n p_{\chi^2}[\alpha;\ 2L + 2n,\ 2(L-\ell_1)\rho_N], & 0 < \ell_1 < L; \quad (3.1\text{-}7c) \end{cases}$$

using

$$\rho_N \equiv S/\sigma_N^2\ , \ \rho_T \equiv S/\sigma_T^2\ , \ K = \sigma_T^2/\sigma_N^2 \tag{3.1-7d}$$

and where

$$c_n = e^{-\ell_1\rho_T}\left(\frac{K-1}{K}\right)^n \left(\frac{1}{K}\right)^{\ell_1} \mathcal{L}_n^{\ell_1-1}\left[\frac{-\ell_1\ \rho_T}{K-1}\right],\ \ell_1 \geq 1. \tag{3.1-7e}$$

In (3.1-7e), the function $\mathcal{L}_n^a(x)$ is the generalized Laguerre polynomial.

For the non-signal channels, substitution of $\rho_N = \rho_T = 0$ in (3.1-7) yields, for $u_m = z_m/\sigma_N^2$ $(m > 1)$,

$$p_{u_m}(\alpha) = \begin{cases} p_{\chi^2}(\alpha;2L), \ \ \ell_m = 0; & (3.1\text{-}8a) \\ \frac{1}{K}\, p_{\chi^2}(\alpha/K;2L), \ \ \ell_m = L; & (3.1\text{-}8b) \\ \sum_{n=0}^{\infty} b_n\, p_{\chi^2}(\alpha;2L+2n), \ 0<\ell_m<L; & (3.1\text{-}8c) \end{cases}$$

where

$$b_n = \left(\frac{K-1}{K}\right)^n \left(\frac{1}{K}\right)^{\ell_m} \binom{n+\ell_m-1}{n}. \tag{3.1-8d}$$

In (3.1-7) and (3.1-8) the chi-squared pdf's are, for 2n degrees of freedom,

$$p_{\chi^2}(\alpha;2n,2\rho) = \frac{1}{2}\ \exp\left\{-\frac{\alpha}{2} - \rho\right\}\left(\frac{\alpha}{2\rho}\right)^{(n-1)/2} I_{n-1}\left(\sqrt{2\rho\alpha}\right) \tag{3.1-9a}$$

$$= \frac{1}{2}\, e^{-\alpha/2}\ (\alpha/2)^{n-1}\Big/\Gamma(n),\ \rho = 0 \tag{3.1-9b}$$

where $I_m(x)$ is the modified Bessel function of the first kind and order m and $\Gamma(\cdot)$ is the gamma function.

3.1.2 Formulation of the Conditional Error Probability.

From Section 2.2, the conditional symbol error probability is

$$\begin{aligned} P_s(e|[\nu]) &= P_s(e|\ell_1,\ell_2,\ldots,\ell_M) \\ &= 1 - \Pr\{z_2<z_1, z_3<z_1,\ldots,z_M<z_1|\underline{\ell}\} \\ &= 1 - \Pr\{u_2<u_1, u_3<u_1,\ldots,u_m<u_1|\underline{\ell}\} \\ &= 1 - \int_0^\infty d\alpha\, p_{u_1}(\alpha) \prod_{m=2}^{M} \int_0^\alpha d\beta_m\, p_{u_m}(\beta_m). \end{aligned} \tag{3.1-10}$$

From (3.1-8) and (3.1-9),

$$F_L(\alpha;\ell_m) \triangleq \int_0^\alpha d\beta_m\, p_{u_m}(\beta_m) = \begin{cases} 1 - \Gamma(L;\alpha/2)/\Gamma(L),\ \ell_m = 0; & (3.1\text{-}11a) \\ 1 - \Gamma(L;\alpha/2K)/\Gamma(L),\ \ell_m = L; & (3.1\text{-}11b) \\ 1 - \sum_{n=0}^{\infty} b_n \Gamma(L+n;\alpha/2)/\Gamma(L+n), & (3.1\text{-}11c) \\ \qquad 0<\ell_m<L; & \end{cases}$$

where $\Gamma(n;t)$ is the incomplete gamma function,

$$\Gamma(n;t) = \int_t^\infty dx\, e^{-x}\, x^{n-1} = \Gamma(n)\, e^{-t} \sum_{r=0}^{n-1} \frac{t^r}{r!} . \tag{3.1-12}$$

Formally, (3.1-10) can be written

$$P_s(e|\underline{\ell}) = 1 - \int_0^\infty d\alpha\, p_{u_1}(\alpha)\, [F_L(\alpha;0)]^{n_0}\, [F_L(\alpha;1)]^{n_1} \ldots [F_L(\alpha;L)]^{n_L}, \tag{3.1-13a}$$

where n_i is the number of non-signal channels with $\ell_m=i$, and it is true that

$$n_0 + n_1 + \ldots + n_L = M - 1 . \tag{3.1-13b}$$

3.1.3 Powers of Non-Signal Channel Probabilities.

We now show that the probabilities $F_L(\alpha;\ \ell_m)$ in the general expression (3.1-13a) can be written in terms of power series.

For $\ell_m = 0$, from (3.1-11) and (3.1-12)

$$[F_L(\alpha;\ 0)]^{n_0} = \left[1 - e^{-\alpha/2} \sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!}\right]^{n_0}$$

$$= \sum_{r_0=0}^{n_0} \binom{n_0}{r_0} (-1)^{r_0}\ e^{-r_0\alpha/2} \left[\sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!}\right]^{r_0}$$

$$= \sum_{r_0=0}^{n_0} \binom{n_0}{r_0} (-1)^{r_0}\ e^{-r_0\alpha/2} \sum_{k_0=0}^{r_0(L-1)} \frac{C(k_0,\ r_0)}{k_0!} \left(\frac{\alpha}{2}\right)^{k_0}, \quad (3.1\text{-}14)$$

where the coefficients $C(k_0,\ r_0)$ are [1, Appendix 4A] the functions

$$C(0,\ r) = 1 \quad (3.1\text{-}15a)$$

$$C(k,\ r) = \frac{1}{k} \sum_{n=1}^{\min(k,L-1)} \binom{k}{n} [(r+1)n - k]\ C(k-n,\ r), \quad (3.1\text{-}15b)$$

$$k > 0\ ,\quad L \geq 2.$$

For example, when L = 2 the coefficients are simply

$$C(k,\ r) = (r+1-k)\ C(k-1,\ r)$$
$$= r!/(r-k)! \quad (3.1\text{-}16a)$$

so that the series raised to the r_0 power is

$$\sum_{k_0=0}^{r_0} \frac{r_0!}{k_0!(r_0-k_0)!}\left(\frac{\alpha}{2}\right)^{k_0} = (1+\alpha/2)^{r_0} \tag{3.1-16b}$$

as required.

Similarly, for $\ell_m = L$, we find that

$$\left[F_L(\alpha;L)\right]^{n_L} = \sum_{r_L=0}^{n_L} \binom{n_L}{r_L}(-1)^{r_L} e^{-r_L\alpha/2K} \sum_{k_L=0}^{r_L(L-1)} \frac{C(k_L,r_L)}{k_L!}\left(\frac{\alpha}{2K}\right)^{k_L}, \tag{3.1-17}$$

using the same $C(k, r)$ function as for $\ell_m=0$.

For $0<\ell_M<L$, $L\geq 2$, the evaluation of the probability is more challenging, but does indeed reduce to a closed form. The development begins by recognizing that

$$\begin{aligned} F_L(\alpha;\ell_m) &= 1 - e^{-\alpha/2} \sum_{n=0}^{\infty} b_n \sum_{r=0}^{n+L-1} \frac{(\alpha/2)^r}{r!} \\ &= 1 - e^{-\alpha/2}\sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!} - e^{-\alpha/2}\sum_{n=0}^{\infty} b_{n+1} \sum_{r=0}^{n} \frac{(\alpha/2)^{r+L}}{(r+L)!} \end{aligned} \tag{3.1-18}$$

where b_n is defined by (3.1-8d). The double summation in the last term can be manipulated in the following way:

$$\begin{aligned} \sum_{n=0}^{\infty} b_{n+1} \sum_{r=0}^{n} \frac{(\alpha/2)^{r+L}}{(r+L)!} &= \sum_{n=0}^{\infty} \sum_{r=0}^{\infty} b_{n+r+1} \frac{(\alpha/2)^{r+L}}{(r+L)!} \\ &= K^{-\ell} \sum_{r=0}^{\infty} \left(\frac{K-1}{K}\right)^{r+1} \frac{(\alpha/2)^{r+L}}{(r+L)!}\binom{r+\ell}{\ell-1}\, {}_2F_1(r+\ell+1,1;r+2;\tfrac{K-1}{K}). \end{aligned} \tag{3.1-19}$$

The hypergeometric function can be transformed using [2, Eq. 15.3.5] to obtain on the summation were zero, the summation would be a Taylor's series:

$$\sum_{n=0}^{\infty} \frac{a^n}{n!} \frac{n!}{(n+L-\ell)!} \; {}_1F_1(n+1;n+L-\ell+1;b)$$

$$= \frac{1}{(L-\ell)!} \sum_{n=0}^{\infty} \frac{a^n}{n!} \left[\frac{d^n}{dx^n} \; {}_1F_1(1;L-\ell+1;x) \right] \Bigg|^{x\,=\,b} \tag{3.1-23a}$$

$$= \left[e^{a+b} - \sum_{n=0}^{L-\ell-1} \frac{(a+b)^n}{n!} \right] \frac{1}{(a+b)^{L-\ell}} ; \tag{3.1-23b}$$

therefore, (3.1-22) is seen to be

$$e^{\alpha/2} - \sum_{n=0}^{L-\ell-1} \frac{(\alpha/2)^n}{n!} - \sum_{n=0}^{\ell-1} \left(\frac{1}{K} \right)^n \frac{(\alpha/2)^{n+L-\ell}}{n!} \frac{d^n}{dx^n} x^{\ell-L} \left[e^x - \sum_{r=0}^{L-\ell-1} \frac{x^r}{r!} \right] \Bigg|^{x=(K-1)\alpha/2K} . \tag{3.1-24}$$

Now, substituting (3.1-24) into (3.1-21) and the resulting expression into (3.1-18) gives

$$F_L(\alpha;\ell) = 1 - e^{-\alpha/2} \left[\sum_{r=0}^{L-1} \frac{(\alpha/2)}{r!} + \sum_{r=0}^{\infty} \frac{(\alpha/2)^{r+L}}{(r+L)!} \right]$$

$$+ e^{-\alpha/2} \left[(3.1\text{-}24) \right]$$

$$= 0 + e^{-\alpha/2} \left[(3.1\text{-}24) \right]$$

$$= 1 - e^{-\alpha/2} \left\{ \sum_{n=0}^{L-\ell-1} \frac{(\alpha/2)^n}{n!} + \left(\frac{\alpha}{2} \right)^{L-\ell} \sum_{n=0}^{\ell-1} \frac{(\alpha/2K)^n}{n!} \frac{d^n}{dx^n} x^{\ell-L} \left[e^x - \sum_{r=0}^{L-\ell-1} \frac{x^r}{r!} \right] \Bigg|^{x=(K-1)\alpha/2K} \right\}$$

$$= 1 - e^{-\alpha/2} \left\{ \sum_{n=0}^{L-\ell-1} \frac{(\alpha/2)^n}{n!} \left[1 - \left(\frac{K}{K-1}\right)^{L-\ell-n} \sum_{r=0}^{\ell-1} \binom{L-\ell+r-n-1}{r} \frac{(-1)^r}{(K-1)^r} \right] \right\}$$

$$-e^{-\alpha/2K} \left(\frac{K}{K-1}\right)^{L-\ell} \sum_{n=0}^{\ell-1} \frac{(\alpha/2K)^n}{n!} \sum_{r=0}^{\ell-1-n} \binom{L-\ell+r-1}{r} \left(\frac{-1}{K-1}\right)^r . \qquad (3.1\text{-}25)$$

For example, if $L = 2$ and $\ell = 1$,

$$F_2(\alpha;\ 1) = 1 - e^{-\alpha/2} \left\{ 1 - \frac{K}{K-1} \right\} - e^{-\alpha/2K} \left(\frac{K}{K-1}\right)$$

$$= 1 - \frac{1}{K-1} \left\{ Ke^{-\alpha/2K} - e^{-\alpha/2} \right\} . \qquad (3.1\text{-}26)$$

By direct algebraic calculation of (3.1-25), it can be shown that $F_L(\alpha, \ell)$ is of the form

$$F_L(\alpha;\ \ell) = 1 - \frac{1}{(K-1)^{L-1}} \left\{ e^{-\alpha/2K} f_1(\alpha;\ \ell, L) + e^{-\alpha/2} f_2(\alpha;\ \ell, L) \right\} , \qquad (3.1\text{-}27)$$

where $f_1(\alpha;\ \ell, L)$ and $f_2(\alpha;\ \ell, L)$ are the polynomials given in Table 3.1-1. Therefore

$$[F_L(\alpha;\ \ell)]^{n_\ell} = \sum_{r_\ell=0}^{n_\ell} \binom{n}{r} \left[\frac{-1}{(K-1)^{L-1}} \right]^{r_\ell}$$

$$\cdot \left[e^{-\alpha/2K} f_1(\alpha;\ \ell, L) + e^{-\alpha/2} f_2(\alpha;\ \ell, L) \right]^{r_\ell}$$

TABLE 3.1-1 POLYNOMIALS FOR $F_L(\alpha;\ell)$

L	ℓ	$f_1(\alpha;\ell,L)$	$f_2(\alpha;\ell,L)$
1	0	0	1
	1	1	0
2	0	0	$(K-1)(1+\alpha/2)$
	1	K	-1
	2	$(K-1)(1+\alpha/2K)$	0
3	0	0	$(K-1)^2(1+\alpha/2+\alpha^2/8)$
	1	K^2	$-\left[2K-1+(K-1)\alpha/2\right]$
	2	$K(K-2)+(K-1)\alpha/2$	1
	3	$(K-1)^2\left[1+\alpha/2K+\alpha^2/8K^2\right]$	0
4	0	0	$(K-1)^3(1+\alpha/2+\alpha^2/8+\alpha^3/48)$
	1	K^3	$-\left[3K^2-3K+1+(2K^2-3K+1)\alpha/2+(K-1)^2\alpha^2/8\right]$
	2	$K^3-3K^2+K(K-1)\alpha/2$	$3K-1+(K-1)\alpha/2$
	3	K^3-3K^2+3K $+(K^2-3K+2)\alpha/2$ $+(K-1)^2\alpha^2/8K$	-1
	4	$(K-1)^3(1+\alpha/2K+\alpha^2/8K^2$ $+\alpha^3/48K^3)$	0

$$= \sum_{r_\ell=0}^{n_\ell} \binom{n_\ell}{r_\ell} \left[\frac{-1}{(K-1)^{L-1}}\right]^{r_\ell} \sum_{k_\ell=0}^{r_\ell} \binom{r_\ell}{k_\ell} e^{-(r_\ell - k_\ell)\alpha/2K - k_\ell \alpha/2}$$

$$\cdot \left[f_1(\alpha;\ \ell,\ L)\right]^{r_\ell - k_\ell} \left[f_2(\alpha;\ \ell,\ L)\right]^{k_\ell}, \quad 0 < \ell < L, \tag{3.1-28}$$

and the powers of the polynomials can be expressed as a higher order polynomial:

$$[f_1]^{r_\ell - k_\ell} [f_2]^{k_\ell} = \sum_{p_\ell=0}^{P} d(p_\ell)\ (\alpha/2)^{p_\ell}. \tag{3.1-29}$$

The coefficients d(p) are given in Table 3.1-2 for L up to 4.

3.1.4 Expectation Over Signal Channel PDF.

Substitution of the powers of the non-signal channel probabilities into the conditional probability of symbol error equation yields the lengthy expression given in Table 3.1-3. The remaining analysis requires obtaining the expectation

$$\int_0^\infty d\alpha\ p_{u_1}(\alpha)\ e^{-a_0 \alpha/2} (\alpha/2)^{b_0} = E_u \left\{ e^{-au_1/2} \left(\frac{u_1}{2}\right)^{b_0} \right\}, \tag{3.1-30}$$

where a_0 and b_0 are given in Table 3.1-3. From (3.1-7a), when the signal channel is not jammed ($\ell_1 = 0$), the pdf $p_{u_1}(\alpha)$ is a straightforward noncentral chi-squared pdf, and the expectation is

$$\int_0^\infty d\alpha\, p_{\chi^2}(\alpha;\ 2L,\ 2L\rho_N)\ e^{-a_0\alpha/2}\ (\alpha/2)^{b_0}$$

TABLE 3.1-2 COEFFICIENTS FOR EQUATION (3.1-29)

L	ℓ	$P = \max p$	$d(p)$
2	1	0	$K^{r_1-k_1}(-1)^{k_1}$
3	1	k_1	$(K^2)^{r_1-k_1}(-1)^{k_1}\binom{k_1}{p}(2K-1)^{k_1-p}(K-1)^p$
3	2	r_2-k_2	$\binom{r_2-k_2}{p}[K(K-2)]^{r_2-k_2-p}(K-1)^p$
4	1	$2k_1$	$(K^3)^{r_1-k_1}(-1)^{k_1}(3K^2-3K+1)^{k_1}\cdot g(p)$ where $g(0) = 1,\ g(1) = k_1 \cdot \frac{2K^2-3K+1}{3K^2-3K+1}$ $g(n) = \frac{1}{n}\left\{(k_1+1-n)\cdot\frac{2K^2-3K+1}{3K^2-3K+1}\cdot g(n-1) + [2(k_1+1)-n]\ \frac{(K-1)^2/2}{3K^2-3K+1}\cdot g(n-2)\right\},\ n \geq 2$
4	2	r_2	$K^{2r_2-2k_2-p}(K-1)^p \sum_{q=\max(0,\,p-r_2+k_2)}^{\min(p,\,k_2)} \binom{r_2-k_2}{p-q}\binom{k_2}{q}$ $\cdot (K-3)^{r_2-k_2-p+q}(3K-1)^{k_2-q}K^q$
4	3	$2(r_3-k_3)$	$(-1)^{k_3}(K^3-3K^2+3K)^{r_3-k_3}\ g(p)$ where $g(0) = 1,\ g(1) = (r_3-k_3)\cdot\frac{K^2-3K+2}{K^3-3K^2+3K}$ $g(n) = \frac{1}{n}\left\{(r_3-k_3+1-n)\cdot\frac{K^2-3K+2}{K^3-3K^2+3K}\cdot g(n-1) + [2(r_3-k_3+1)-n]\cdot\frac{(K-1)^2/2K}{K^3-3K^2+3K}\cdot g(n-2)\right\},\ n \geq 2$

TABLE 3.1-3 EXPRESSION FOR CONDITIONAL ERROR PROBABILITY

$$P_s(e|\underline{\ell}) = 1 - \int_0^\infty d\alpha \; p_{u_1}(\alpha) \, [F_L(\alpha;0)]^{n_0} \, [F_L(\alpha;1)]^{n_1} \cdots [F_L(\alpha;L-1)]^{n_{L-1}} \, [F_L(\alpha;L)]^{n_L}$$

$$= 1 - \sum_{r_0=0}^{n_0} \binom{n_0}{r_0} (-1)^{r_0} \sum_{k_0=0}^{r_0(L-1)} \frac{C(k_0,r_0)}{k_0!} \quad \sum_{r_L=0}^{n_L} \binom{n_L}{r_L} (-1)^{r_L} \sum_{k_L=0}^{r_L(L-1)} \frac{C(k_L,r_L)}{k_L!} \; K^{-k_L}$$

$$\times \sum_{r_1=0}^{n_1} \binom{n_1}{r_1} \left[\frac{-1}{(K-1)^{L-1}}\right]^{r_1} \sum_{k_1=0}^{r_1} \binom{r_1}{k_1} \sum_{p_1=0}^{p_1} d(p_1) \cdots$$

$$\sum_{r_{L-1}=0}^{n_{L-1}} \binom{n_{L-1}}{r_{L-1}} \left[\frac{-1}{(K-1)^{L-1}}\right]^{r_{L-1}} \sum_{k_{L-1}=0}^{r_{L-1}} \binom{r_{L-1}}{k_{L-1}} \sum_{p_{L-1}=0}^{P_{L-1}} d(p_{L-1})$$

$$\times \int_0^\infty d\alpha \; p_{u_1}(\alpha) \; e^{-a_0\alpha/2} \, (\alpha/2)^{b_0}$$

$$a_0 = r_0 + \frac{r_L}{K} + \sum_{\ell=1}^{L-1} \left[k_\ell + \frac{r_\ell - k_\ell}{K}\right]$$

$$b_0 = k_0 + k_L + \sum_{\ell=1}^{L-1} p_\ell$$

$$= \int_0^\infty d\alpha \; e^{-L\rho_N-(1+a_0)\alpha/2} \left(\frac{\alpha}{2L\rho_N}\right)^{(L-1)/2} I_{L-1}\left(\sqrt{2L\rho_N\alpha}\right)\left(\frac{\alpha}{2}\right)^{b_0}$$

$$= \exp\left(\frac{-L\rho_N a_0}{1+a_0}\right)(1+a_0)^{-b_0-L}$$

$$\cdot \int_0^\infty dx \; e^{-L\rho_N/(1+a_0)-x/2} \left[\frac{x}{2L\rho_N/(1+a_0)}\right]^{(L-1)/2} I_{L-1}\left(\sqrt{\frac{2L\rho_N x}{1+a_0}}\right)\left(\frac{x}{2}\right)^{b_0}$$

$$= \exp\left(\frac{-L\rho_N a_0}{1+a_0}\right)(1+a_0)^{-b_0-L} \cdot E\left[\left(\frac{x}{2}\right)^{b_0}\right], \qquad (3.1\text{-}31)$$

where x is distributed as a noncentral chi-squared random variable with 2L degrees of freedom and noncentral parameter $2L\rho_N/(1+a_0)$. The moments needed are

$$E\left[\left(\frac{x}{2}\right)^{b_0}\right] = b_0!\,\mathcal{L}_{b_0}^{L-1}\left(-\frac{L\rho_N}{1+a_0}\right) \qquad (3.1\text{-}32a)$$

$$= \sum_{r=0}^{b_0} \binom{b_0+L-1}{b_0-r} \left(\frac{L\rho_N}{1+a_0}\right)^r \frac{1}{r!}\, b_0! \qquad (3.1\text{-}32b)$$

where $\mathcal{L}_n^a(x)$ is the generalized Laguerre polynomial. Thus for the case of

$\ell_1 = 0$, the integral in (3.1-30) becomes the quantity

$$\exp\left(-\frac{L\rho_N a_0}{1+a_0}\right) \frac{b_0!}{(1+a_0)^{b_0+L}} \mathcal{L}_{b_0}^{L-1}\left(-\frac{L\rho_N}{1+a_0}\right). \tag{3.1-33}$$

In a similar way, when all the hops in the signal channel are jammed ($\ell_1 = L$), the integral in (3.1-30) becomes

$$\exp\left(-\frac{L\rho_T K a_0}{1+Ka_0}\right) \frac{K^{b_0} b_0!}{(1+Ka_0)^{b_0+L}} \mathcal{L}_{b_0}^{L-1}\left(-\frac{L\rho_T}{1+Ka_0}\right). \tag{3.1-34}$$

Now, when the signal channel is jammed, but not on every hop ($0<\ell_1<L$), the channel pdf is a series of weighted noncentral chi-squared pdf's, as shown previously in (3.1-7c). This expectation (3.1-30) yields

$$\exp\left[-\frac{(L-\ell_1)\rho_N a_0}{1+a_0}\right]\sum_{n=0}^{\infty} c_n \frac{b_0!}{(1+a_0)^{b_0+L+n}} \mathcal{L}_{b_0}^{L+n-1}\left[-\frac{(L-\ell_1)\rho_N}{1+a_0}\right], \tag{3.1-35}$$

where the weights $\{c_n\}$ are

$$c_n = e^{-\ell_1\rho_T}\left(\frac{K-1}{K}\right)^n K^{-\ell_1} \mathcal{L}_n^{\ell_1-1}\left[-\frac{\ell_1\rho_T}{K-1}\right]. \tag{3.1-36}$$

In order to reduce (3.1-35) to a finite summation, it is necessary to seek an identity for

$$\sum_{n=0}^{\infty} A^n \mathcal{L}_n^a(x) \mathcal{L}_r^{n+k}(y). \tag{3.1-37}$$

To accomplish this objective, we note that [3, eq. 8.970.1]

$$\mathcal{L}_r^{n+k}(y) \triangleq \frac{1}{r!} e^y\, y^{-n-k} \frac{d^r}{dy^r} \left(e^{-y}\, y^{r+n+k}\right); \tag{3.1-38}$$

substituting in (3.1-37) yields the development

$$\frac{1}{r!} e^y\, y^{-k} \sum_{n=0}^{\infty} \left(\frac{A}{y}\right)^n \mathcal{L}_n^a(x) \frac{d^r}{dy^r} \left(e^{-y}\, y^{r+n+k}\right)$$

$$= \frac{e^y\, y^{-k}}{r!} \frac{\partial^r}{\partial z^r} \left[e^{-z}\, z^{r+k} \sum_{n=0}^{\infty} \left(\frac{Az}{y}\right)^n \mathcal{L}_n^a(x) \right] \Bigg|^{z=y}$$

$$= \frac{e^y\, y^{-k}}{r!} \sum_{q=0}^{r} \binom{r}{q} \frac{\partial^{r-q}}{\partial z^{r-q}} \left(e^{-z}\, z^{r+k}\right) \frac{\partial^q}{\partial z^q} \left[\sum_{n=0}^{\infty} \left(\frac{Az}{y}\right)^n \mathcal{L}_n^a(x) \right] \Bigg|^{z=y}$$

$$= e^y\, y^{-k} \sum_{q=0}^{r} \frac{1}{q!}\, e^{-z}\, z^{k+q}\, \mathcal{L}_{r-q}^{k+q}(z) \sum_{n=q}^{\infty} \left(\frac{A}{y}\right)^n \frac{n!}{(n-q)!}\, z^{n-q}\, \mathcal{L}_n^a(x) \Bigg|^{z=y} . \tag{3.1-39}$$

The second summation, when manipulated, gives the result

$$\left(\frac{A}{y}\right)^q \sum_{n=0}^{\infty} \frac{(n+q)!}{n!} \left(\frac{Az}{y}\right)^n \mathcal{L}_{n+q}^a(x)$$

$$= \left(\frac{A}{y}\right)^q \sum_{n=0}^{\infty} \left(\frac{Az}{y}\right)^n \frac{(n+q+a)!}{n!\; a!}\, {}_1F_1(-n-q;\, a+1;\, x)$$

$$= \left(\frac{A}{y}\right)^q e^x \sum_{n=0}^{\infty} \left(\frac{Az}{y}\right)^n \frac{(n+q+a)!}{n!\ a!} \ {}_1F_1(n+q+a+1;\ a+1;\ -x)$$

$$= \left(\frac{A}{y}\right)^q e^x \sum_{n=0}^{\infty} \left(\frac{Az}{y}\right)^n \frac{1}{n!} \sum_{m=0}^{\infty} \frac{(-x)^m}{m!} \frac{(n+q+a+m)!}{(a+m)!}$$

$$= \left(\frac{A}{y}\right)^q e^x \sum_{m=0}^{\infty} \frac{(-x)^m}{m!} \frac{(q+a+m)!}{(a+m)!} (1 - Az/y)^{-q-a-m-1}$$

$$= \left(\frac{A}{z}\right)^q e^x \frac{(q+a)!}{a!} (1 - Az/y)^{-q-a-1} \ {}_1F_1\left(q+a+1;\ a+1;\ \frac{-x}{1 - Az/y}\right)$$

$$= \left(\frac{A}{y}\right)^q \frac{\exp\left(\frac{-Axz}{y-Az}\right)}{(1-Az/y)^{q+a+1}} \ q!\ \mathcal{L}_q^a\left(\frac{xy}{y-Az}\right) . \tag{3.1-40}$$

After substitution in (3.1-39), we find that

$$\sum_{n=0}^{\infty} A^n \mathcal{L}_n^a(x)\ \mathcal{L}_r^{n+k}(y)$$

$$= \frac{1}{(1-A)^{a+1}} \exp\left(\frac{-Ax}{1-A}\right) \sum_{q=0}^{r} \left(\frac{A}{1-A}\right)^q \mathcal{L}_{r-q}^{k+q}(y)\ \mathcal{L}_q^a\left(\frac{x}{1-A}\right). \tag{3.1-41}$$

And, substituting appropriately, the desired expectation (3.1-35) becomes

$$\exp\left[-\frac{(L-\ell_1)\rho_N a_0}{1+a_0} - \frac{\ell_1 \rho_T K a_0}{1+Ka_0}\right] \frac{b_0!}{(1+a_0)^{b_0+L-\ell_1}(1+Ka_0)^{\ell_1}}$$

$$\cdot \sum_{q=0}^{b_0} \left(\frac{K-1}{1+Ka_0}\right)^q \mathcal{L}_{b_0-q}^{q+L-1}\left[-\frac{(L-\ell_1)\rho_N}{1+a_0}\right] \mathcal{L}_q^{\ell_1-1}\left[-\frac{\ell_1\rho_T}{K-1}\cdot\frac{K(1+a_0)}{1+Ka_0}\right] . \quad (3.1\text{-}42)$$

3.1.5 Special case: L=1 and M=2.

For L=1 and M=2 the FH/RMFSK total error probability is

$$P(e) = \pi_0 \cdot \frac{1}{2} e^{-\rho_N/2} + \pi_1 \cdot e^{-\rho_N/(K+1)} + \pi_2 \cdot \frac{1}{2} e^{-\rho_T/2} \quad (3.1\text{-}43a)$$

where $\rho_N = E_b/N_0$, $\rho_T = E_b/N_T$; (3.1-43b)

and $K = \sigma_T^2/\sigma_N^2 = \rho_N/\rho_T$. (3.1-43c)

3.2 NUMERICAL RESULTS

3.2.1 Soft-Decision Receiver.

Numerical results were obtained using two computational methods. In regions where the series converge rapidly enough, the form given in Table 3.1-3 and equation (3.1-42) was used for the computations. However, the presence of the term $(K-1)^L$ in the denominator of several terms causes difficulty when K is nearly equal to 1. In these cases, and on other occasions when round-off errors became excessive, the computations were performed by direct numerical integration of (3.1-10) using the densities (3.1-7) and (3.1-8) and the identity

$$1 - \int_0^\infty d\alpha\ p(\alpha)\ g(\alpha) \equiv \int_0^\infty d\alpha\ p(\alpha)\left[1-g(\alpha)\right] \tag{3.2-1}$$

which holds for all density functions $p(\alpha)$ for which $p(\alpha) \equiv 0$ if $\alpha < 0$, and hence by the properties of a p.d.f.

$$\int_0^\infty d\alpha\ p(\alpha) = 1. \tag{3.2-2}$$

Then (3.2-1) follows immediately from the fact that integration is a linear operation. The form on the right-hand side of (3.2-1) has the computational advantage that only the integrand need be computed to high accuracy, rather than the integral. For example, if $P_s(e) \approx 10^{-5}$ and we desire 4 significant digits in the answer, than the integral on the left-hand side must be computed numerically to 8-digit accuracy (e.g. 0.9999xxxx) in order to leave 4 non-zero correct digits after subtracting from 1. But if we use the form on the right-hand side of (3.2-1) we could demand only 4-place relative accuracy from the numerical integration; the burden of many-place accuracy is placed only on the function $g(\cdot)$, which is usually much simpler and faster to compute then the overall integral. A listing of the computer program is given in Appendix D.

In Figures 3.2-1 through 3.2-4 we show the bit error probability as a function of bit energy-to-jamming density ratio with jamming fraction γ as a parameter for the case of M=2 (binary FSK) and L=1,2,3, and 4 hops per symbol (bit), respectively. In these figures the ratio of the bit energy to thermal noise density is set at 13.3525 dB, which corresponds to a bit error probability of 10^{-5} for one hop per bit in the absence of jamming. We note that for a given E_b/N_J ratio there is an optimum value of γ which maximizes the jammer's effectiveness. Furthermore, an incorrect choice of γ by the jammer can reduce the effectiveness (as measured by the communicator's bit error probability) by possibly as much as two orders of magnitude.

Figure 3.2-5 shows the envelopes of the curves in Figures 3.2-1 through 3.2-4, which represent the performance of the square-law combining receiver in worst-case partial-band noise jamming. We note that increasing L, the number of hops per bit, consistently degrades the performance. This implies that the noncoherent combining loss dominates over any diversity gain effects.

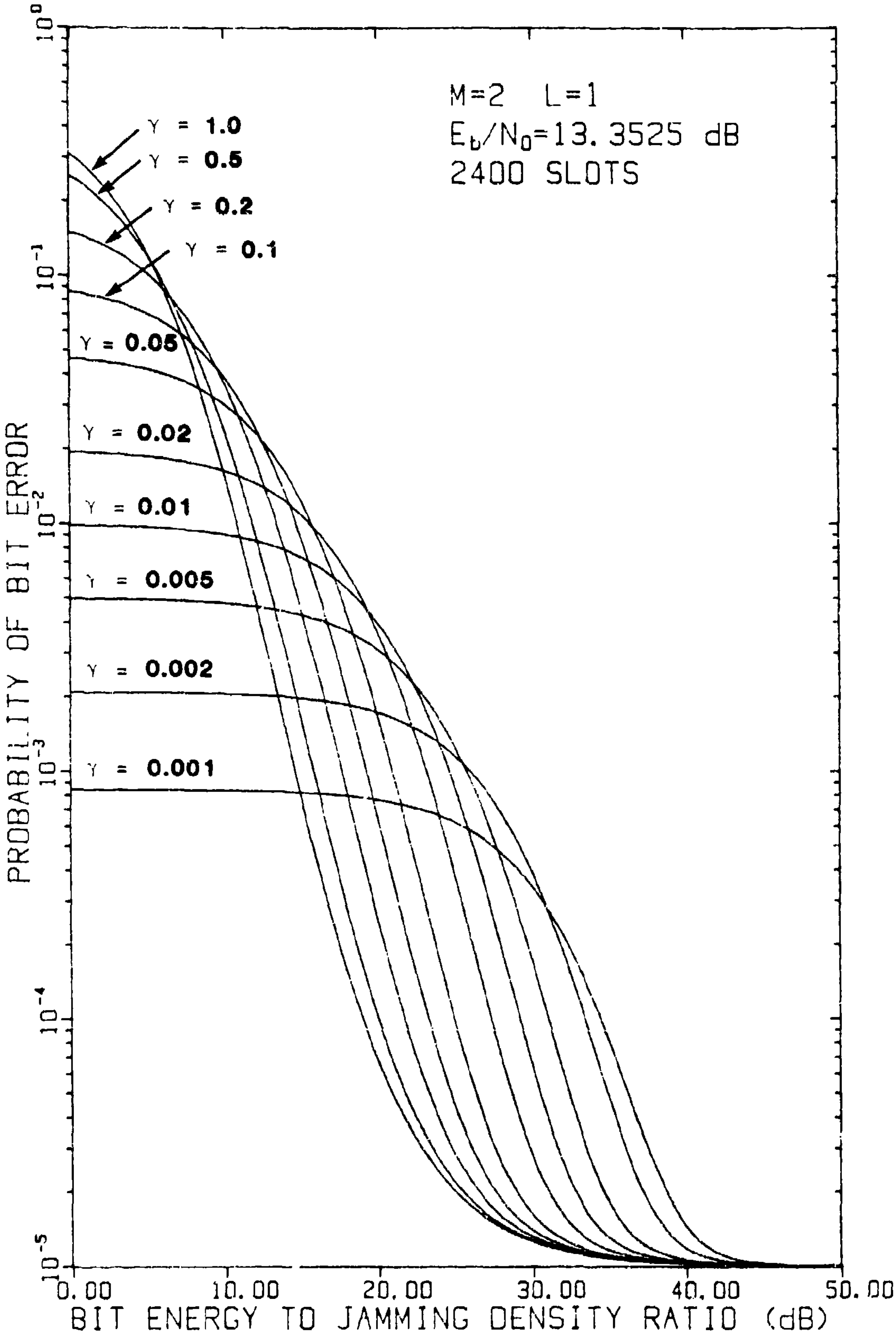


FIGURE 3.2-1 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M = 2 AND L = 1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.3525 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

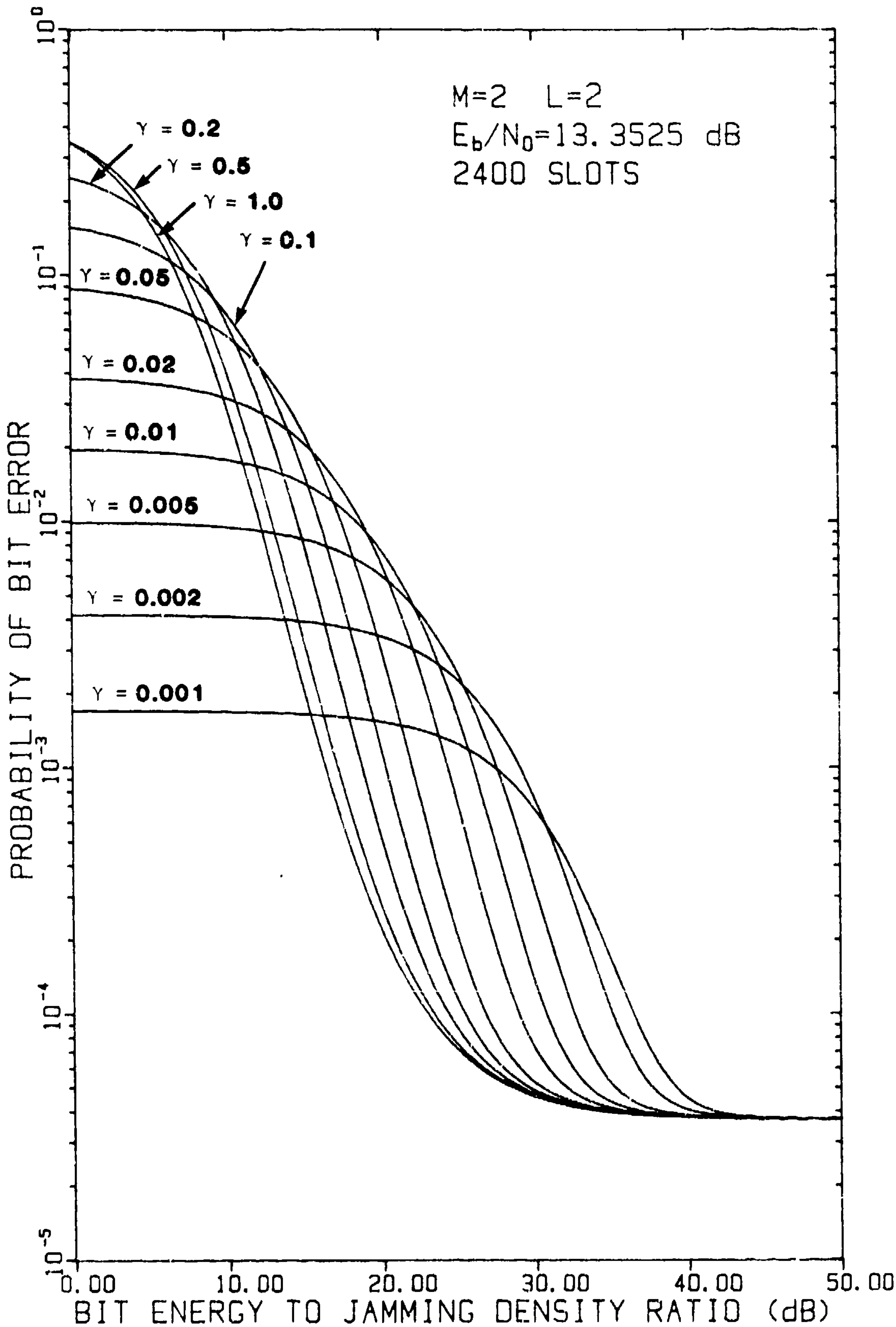


FIGURE 3.2-2 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M=2 AND L=2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.3525 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

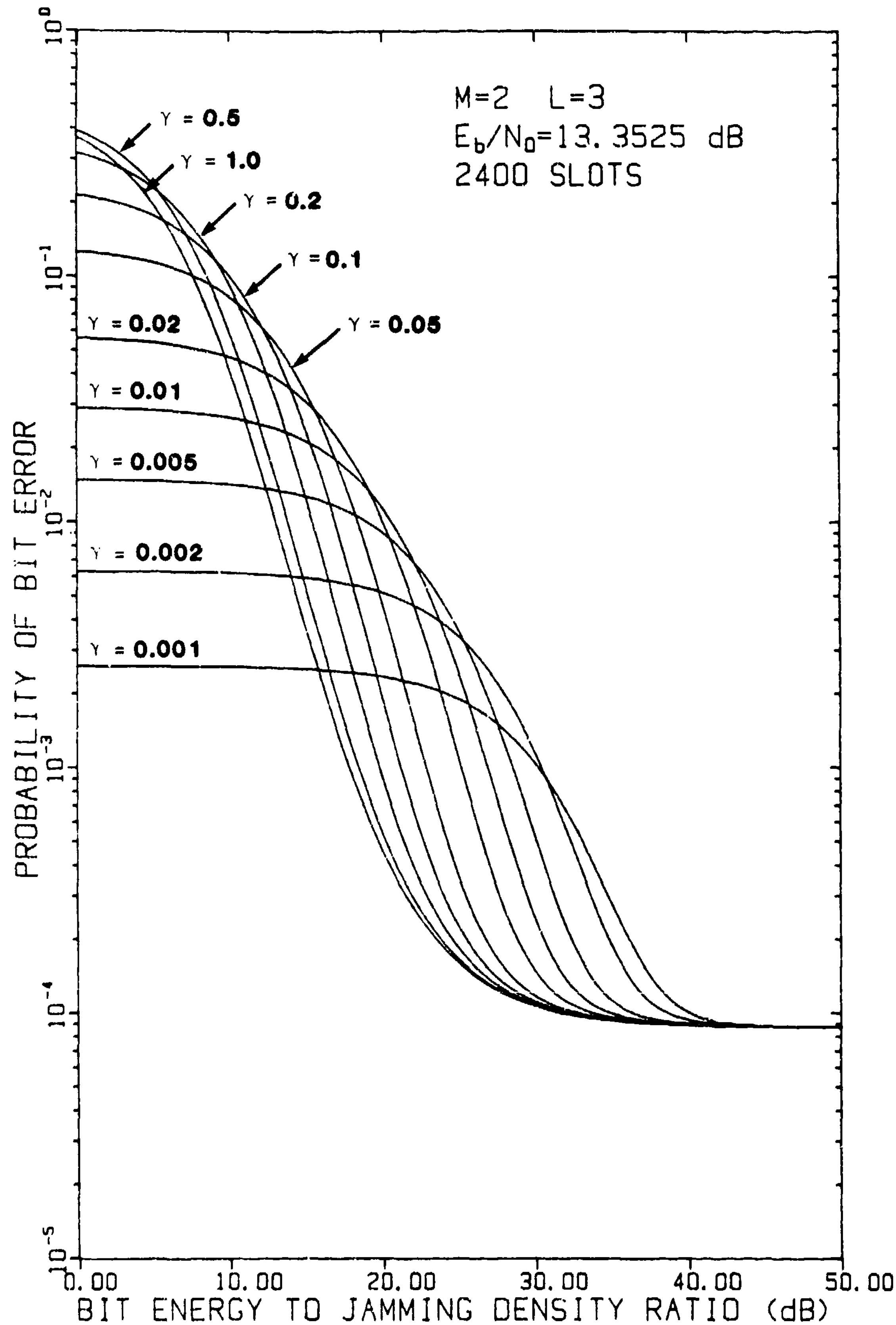


FIGURE 3.2-3 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M=2 AND L=3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.3525 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

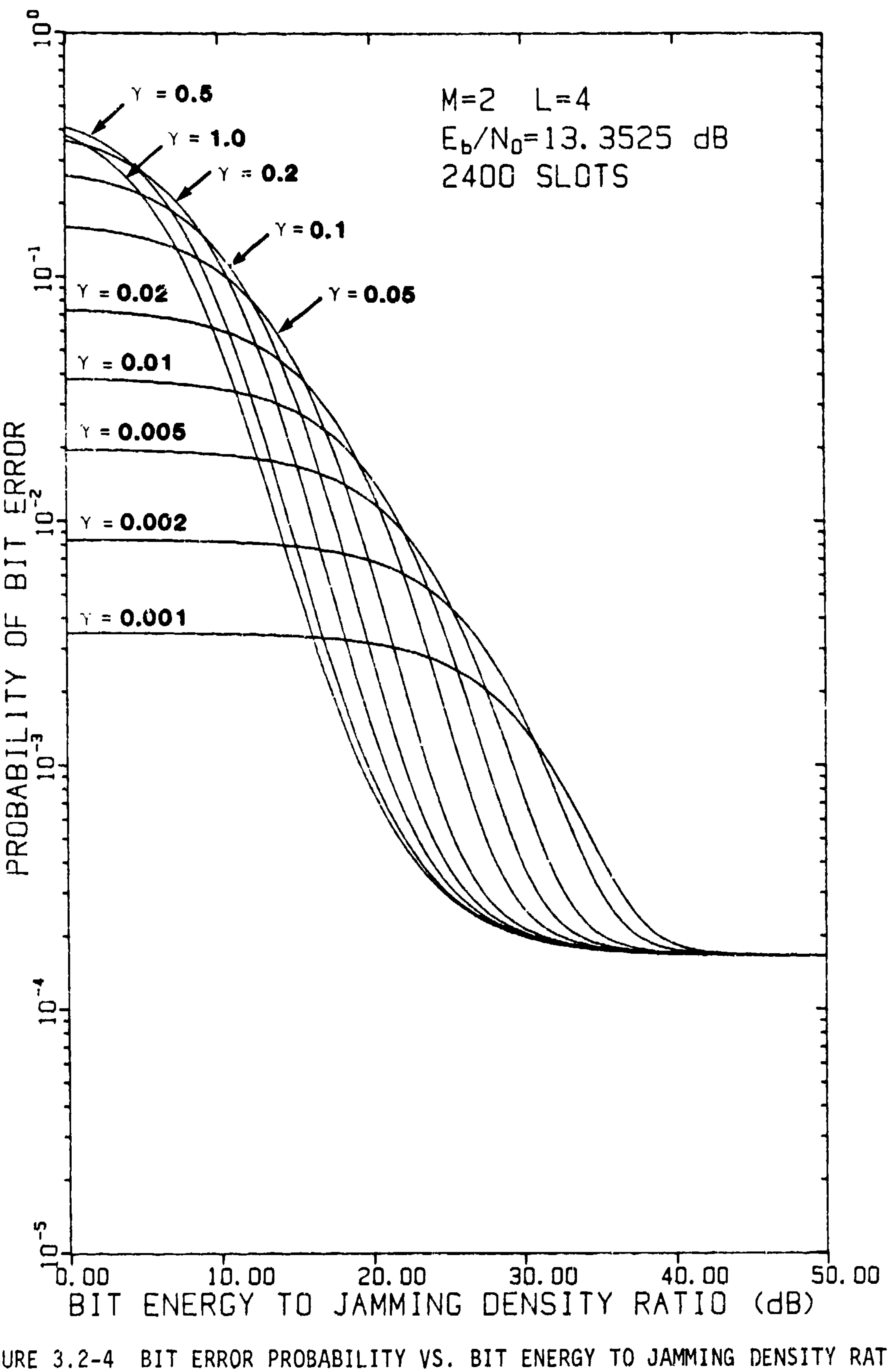


FIGURE 3.2-4 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M=2 AND L=4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.3525 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

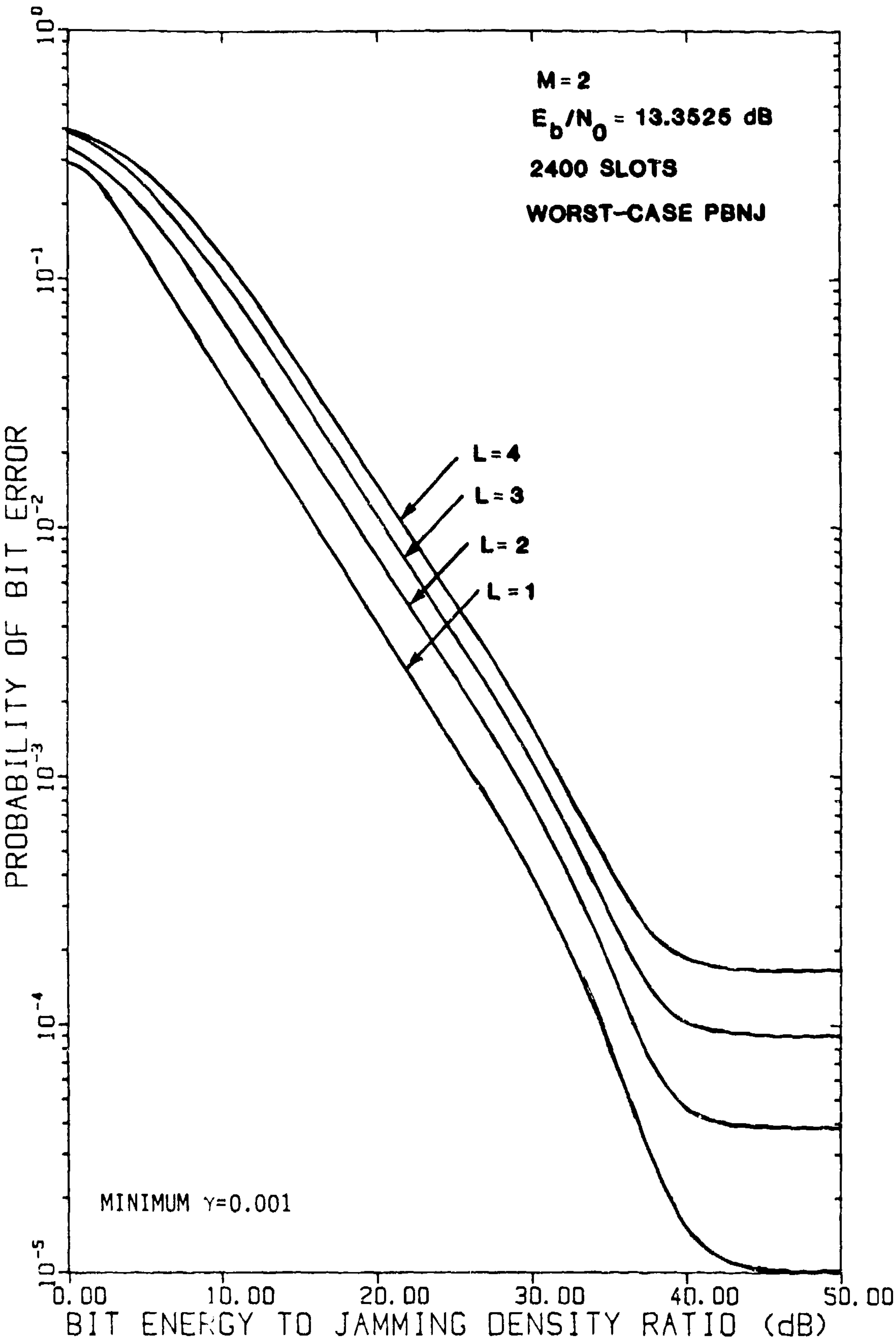


FIGURE 3.2-5 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER AND M = 2 WITH NUMBER OF HOPS/BIT AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

Similar results for the case of M=4 are shown in Figures 3.2-6 through 3.2-9 for L=1,2,3, and 4 hops/symbol, respectively. Again we note that the jammer must carefully choose the proper partial-band fraction or risk reducing his effectiveness by more than an order of magnitude. We also observe that full-band jamming (γ=1.0) is not optimum until the jamming becomes very strong, i.e. $E_b/N_J < 0$ dB.

Figure 3.2-10 shows the envelope of the curves in Figures 3.2-6 through 3.2-9, which gives the performance in worst-case partial-band noise jamming. We note that increasing the number of hops per symbol consistently degrades the performance of the 4-ary system, just as it does for the binary system.

Finally, Figure 3.2-11 shows the worst-case partial-band noise jamming performance of the square-law receiver for L=1 hop/symbol and M=2,4, and 8. We observe that for strong jamming increasing M from 2 to 4 provides a very small performance improvement; but further increase to M=8 degrades the performance. This behavior is similar to that of a block-hopping system in tone jamming [1, Section 8].

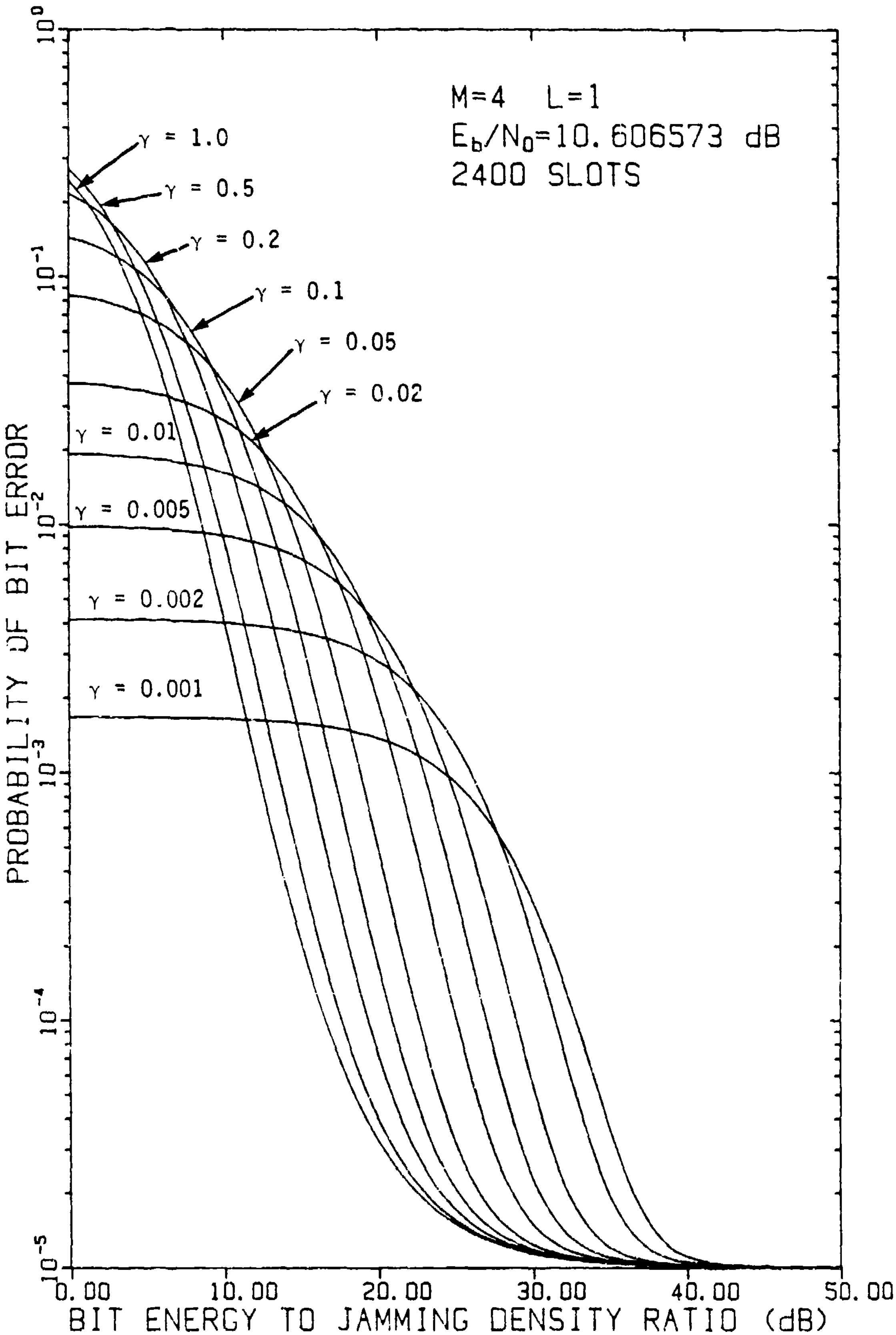


FIGURE 3.2-6 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M = 4 AND L = 1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

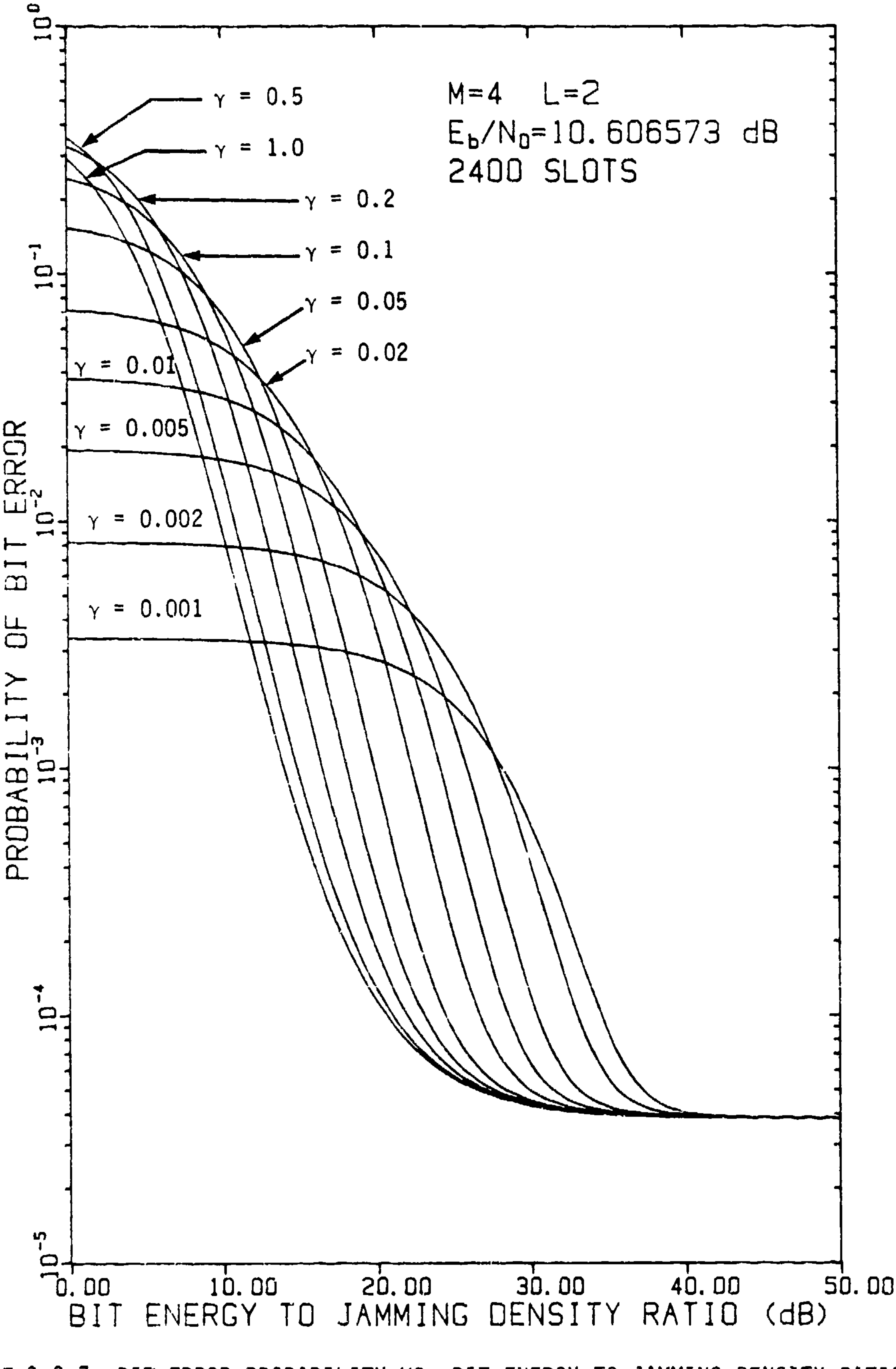


FIGURE 3.2-7 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M = 4 AND L = 2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITH-JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

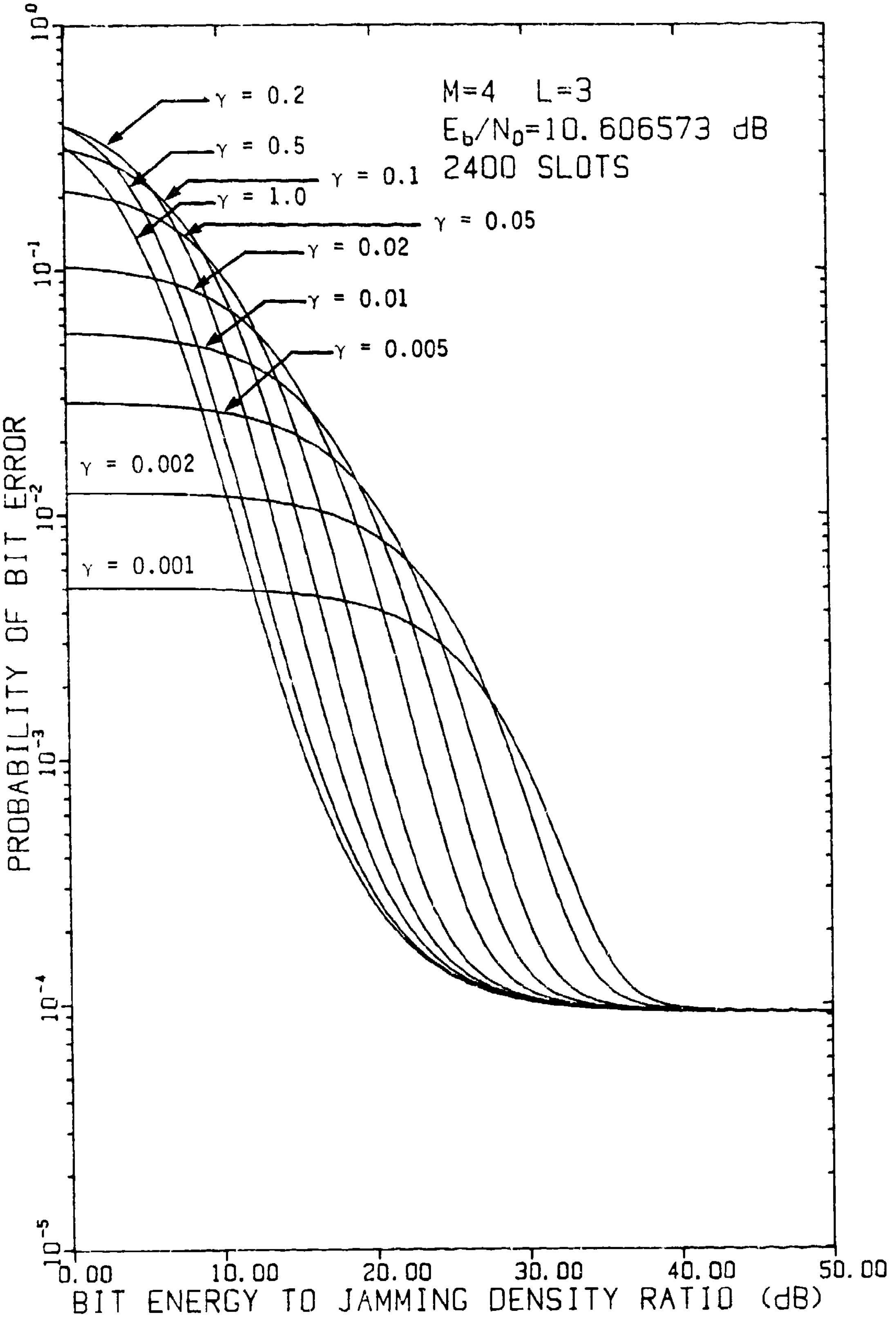


FIGURE 3.2-8 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M = 4 AND L = 3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

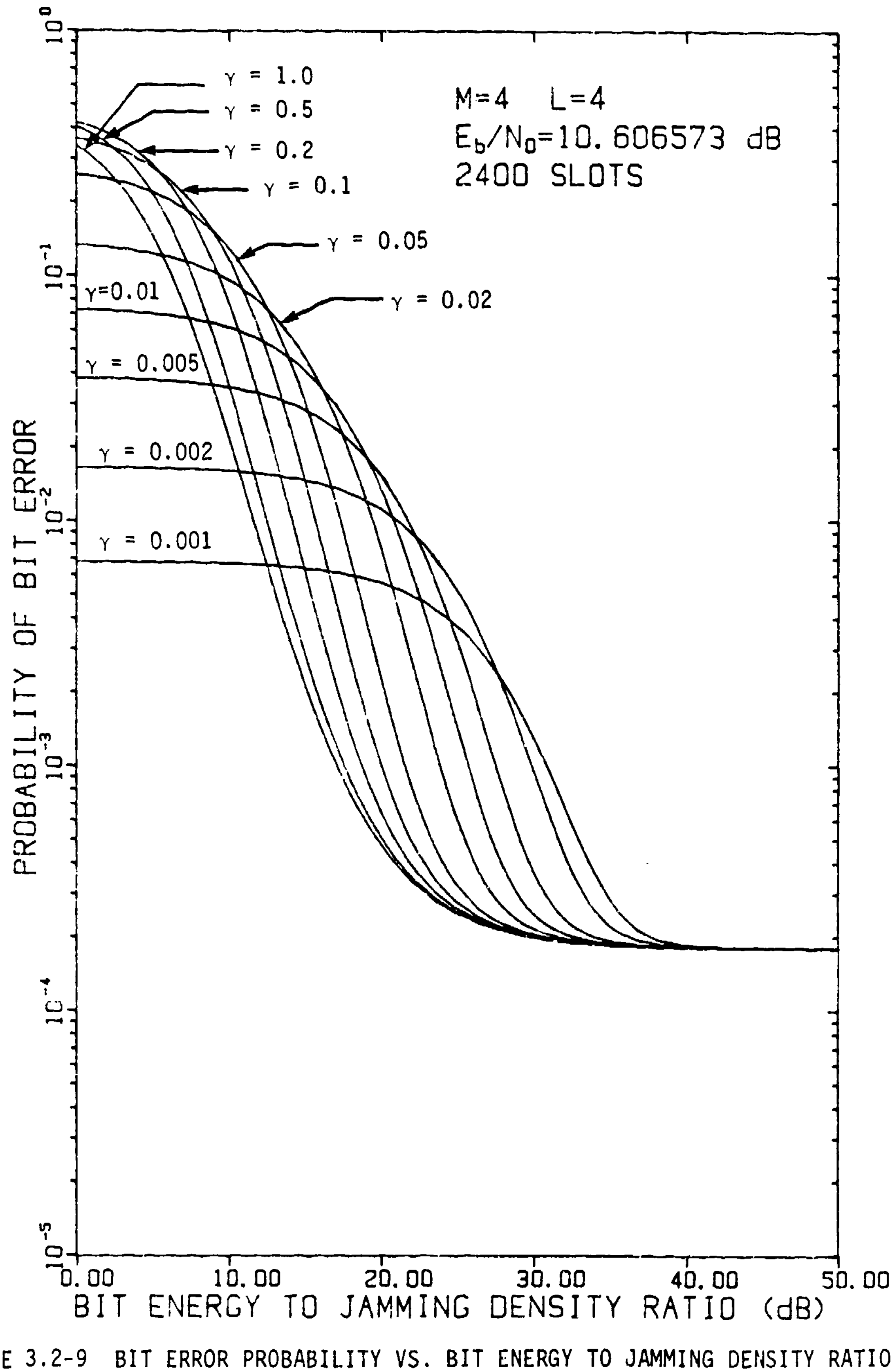


FIGURE 3.2-9 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH M = 4 AND L = 4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

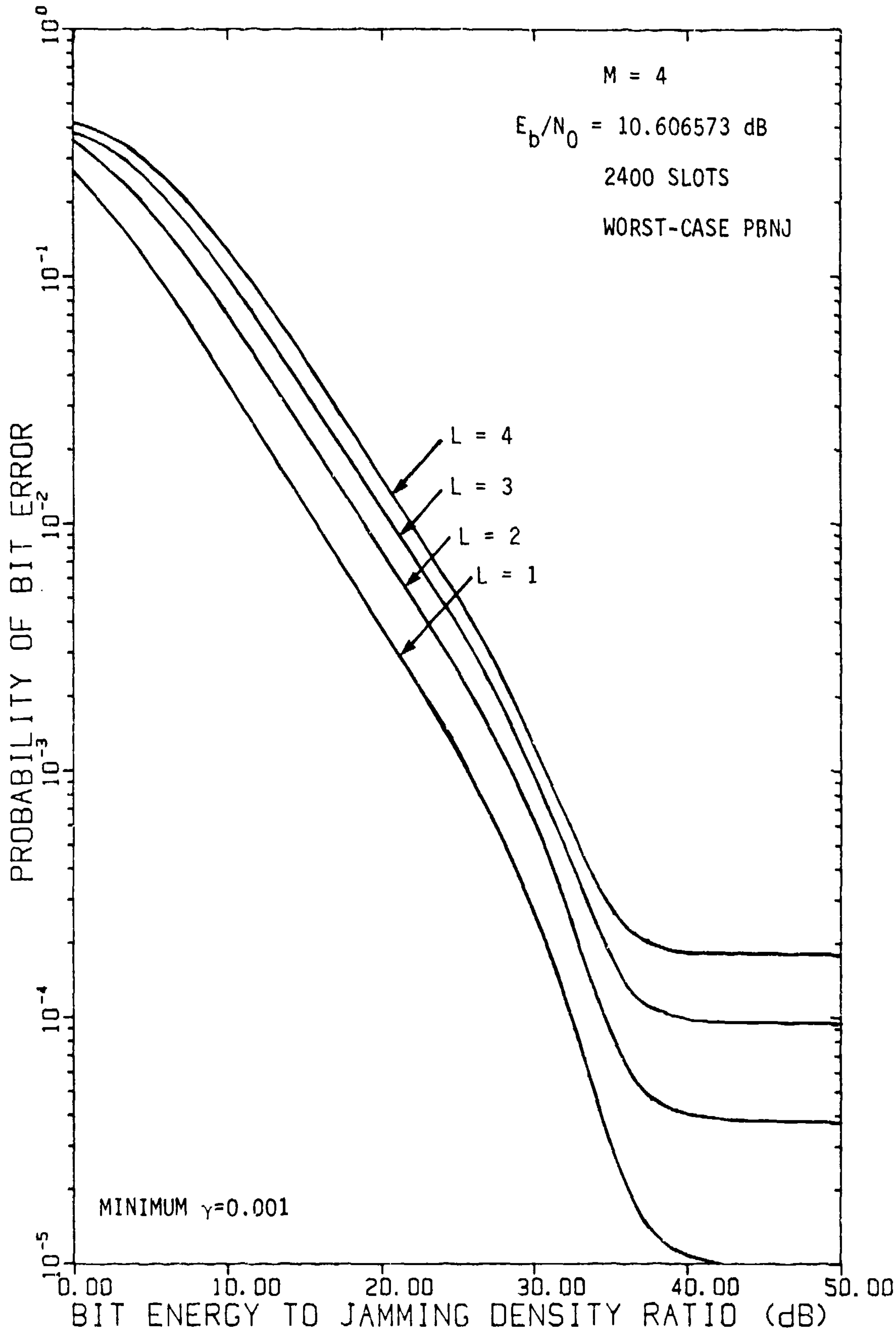


FIGURE 3.2-10 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER AND M = 4 WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

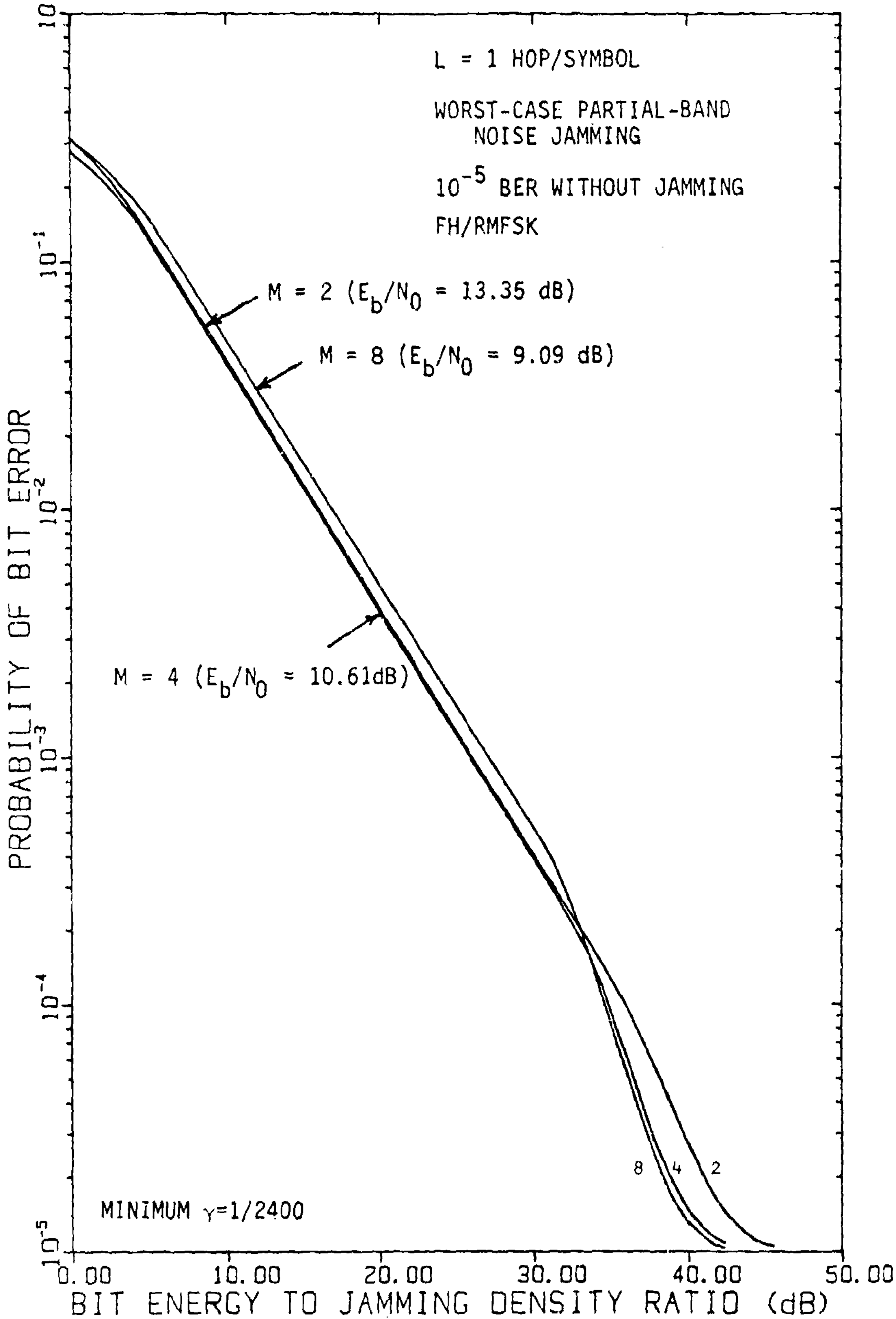


FIGURE 3.2-11 WORST-CASE PARTIAL-BAND NOISE JAMMING PERFORMANCE OF FH/RMFSK RECEIVERS FOR L = 1 HOP/SYMBOL AND M = 2, 4, 8 WHEN E_b/N_0 GIVES A 10^{-5} BER WITHOUT JAMMING

3.2.2 Hard-Decision Receiver

In this subsection we apply the explicit form of the error probability expression (2.4-14) to evaluate the symbol error probability, $P_s(e)$, for a square-law receiver with hard decisions. We consider M-ary cases of M=2,4, and 8 with L values (hops per symbol) ranging from one through five. The worst-case or maximum probability of error is obtained by computing $P_s(e)$ upon varying the number of noise jammed hopping slots q, for an FH/RMFSK system comprised of 2400 hopping slots; i.e. a partial-band noise jamming (PBNJ) model.

We first present plots of numerical results for P(e) versus the variable E_b/N_J with thermal noise (E_b/N_0) as a parameter. Practical values of E_b/N_0 were chosen for which the probability of error becomes 10^{-5} in the absence of jamming. These values are: 13.35247, 10.60657, and 9.09401 dB for M-ary signalling alphabets of M=2, 4, and 8 respectively. Corresponding performance plots are shown in Figures 3.2-12 to 3.2-14.

A comprehensive view of Figure 3.2-12 (M=2) reveals that all five of the L error curves could be grouped into three E_b/N_J regions of relative jamming strength: (1) below 11 dB (strong), (2) 11 dB to 28 dB (medium), and (3) beyond 28 dB (weak). Within the strong jamming region, we see a consistent P(e) L-curve ordering of 4,5,2,3,1 and a 4,2,5,3,1 ranking in the weak region. The region defined as medium strength jamming exhibits "cross-overs" of the various L P(e) curves. Similarly, for M=4 (Figure 3.2-13) and M=8 (Figure 3.2-14) this same three-region behavior exists as follows:

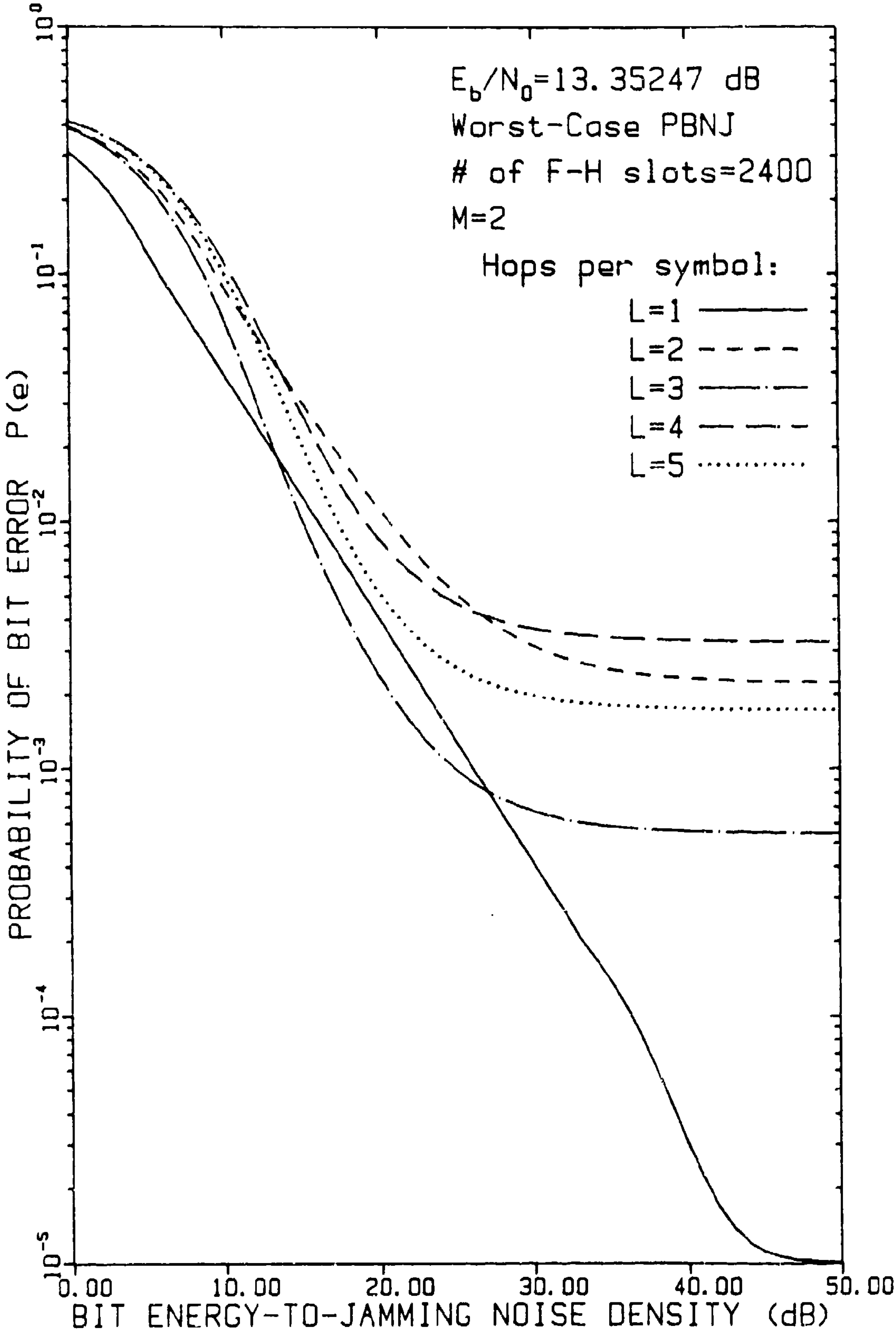


FIGURE 3.2-12 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=2 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=13.35247 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1)

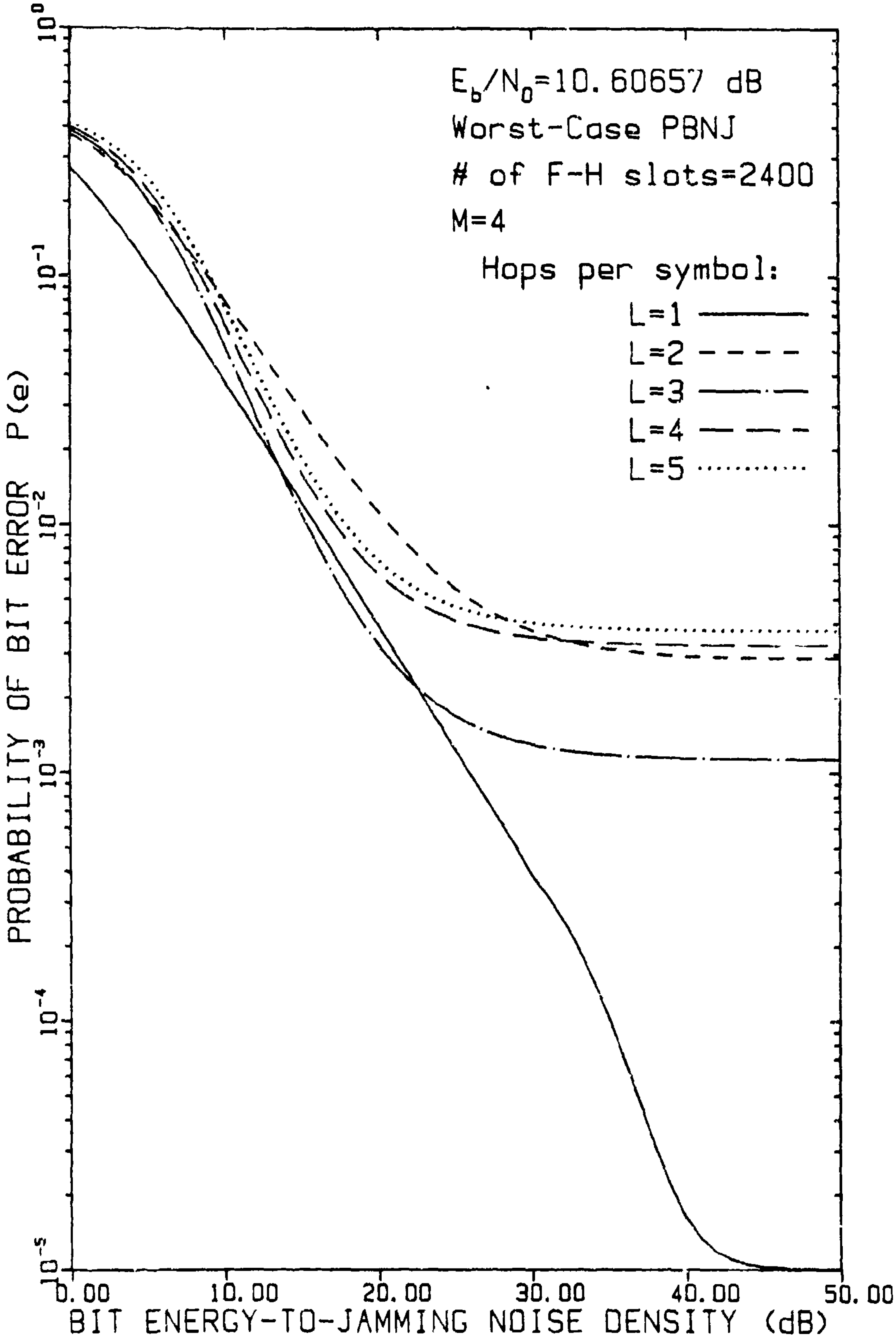


FIGURE 3.2-13 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=4 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=10.60657 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1)

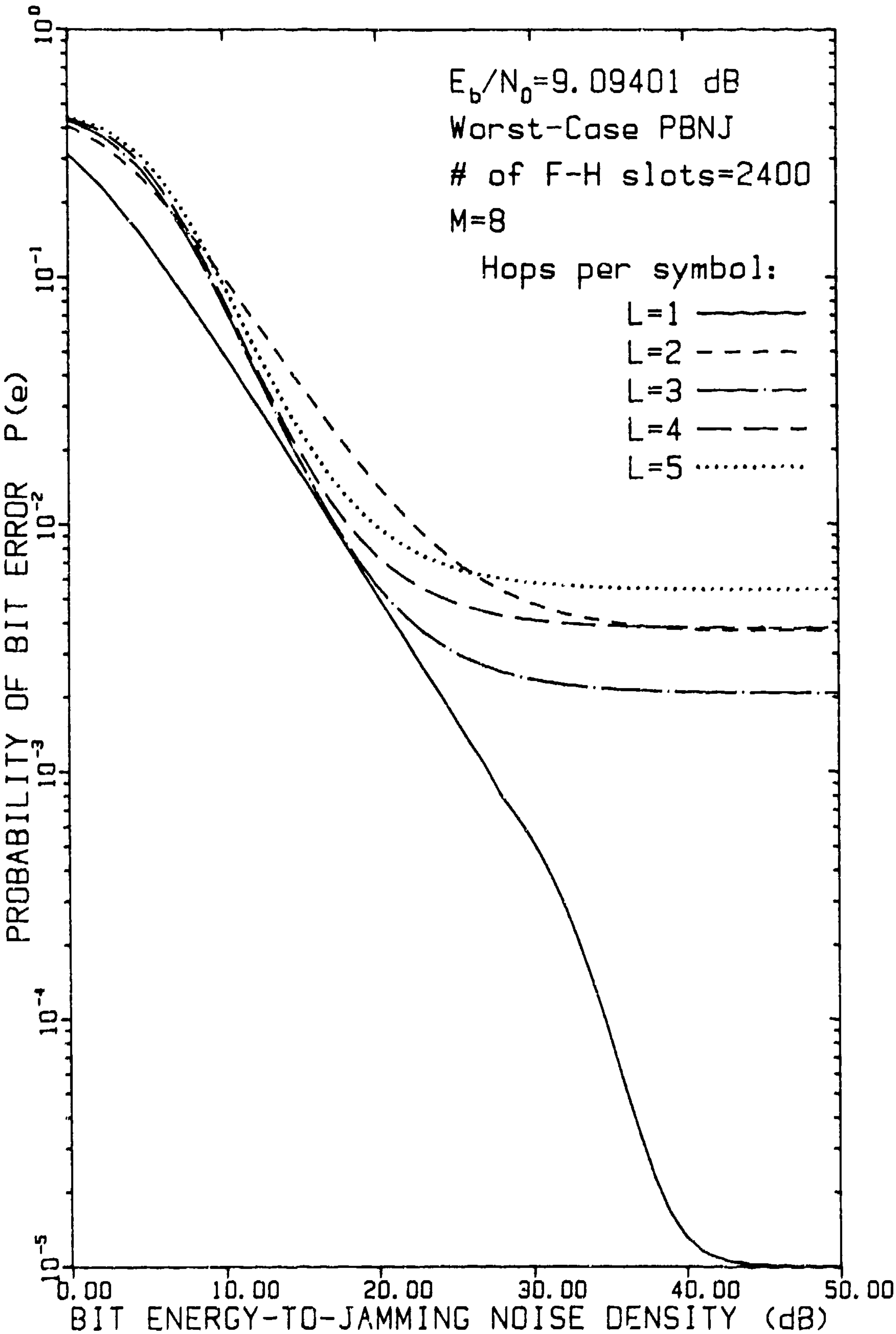


FIGURE 3.2-14 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=8 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=9.09401 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1)

Relative Jamming Region of E_b/N_J (dB) Values

M	Strong	Medium	Weak
4	<4	4 to 32	>32
8	<6	6 to 38	>38 .

The ranking of the L P(e) curves for both M=4 and M=8 in the strong jamming region is 5,4,3,2,1 while for weak jamming a 5,4,2,3,1 ordering is observed.

It is plain to see that for the most part no diversity improvement is realized by the communicator with the exception of a portion of the L=3 P(e) curve in medium jamming for M=2 and M=4. This general behavior can be attributed to the dominance of the noncoherent combining loss existing for the stated thermal noise levels.

A somewhat different trend is noticed when the effect of thermal noise is minimized. Figures 3.2-15 through 3.2-17 show P(e) results for M=2, 4, and 8 respectively at E_b/N_0 levels of 20 dB. Clearly, the regions of strong and weak jamming are now quite discernable with a smaller crossover region (medium jamming) existing among the L P(e) curves. However, we do notice a ranking in the weak jamming areas that differs from those of the 10^{-5} parameter E_b/N_0 curves previously presented. Here it is easily seen that the hard-decision receiver is uniformly 3 dB better for L=1 than for L=2 as described by (2.4-15) for any value of M. Also, a form of diversity improvement is realized as L becomes greater than 2 for all M-ary cases at E_b/N_J values of more than about 12 dB.

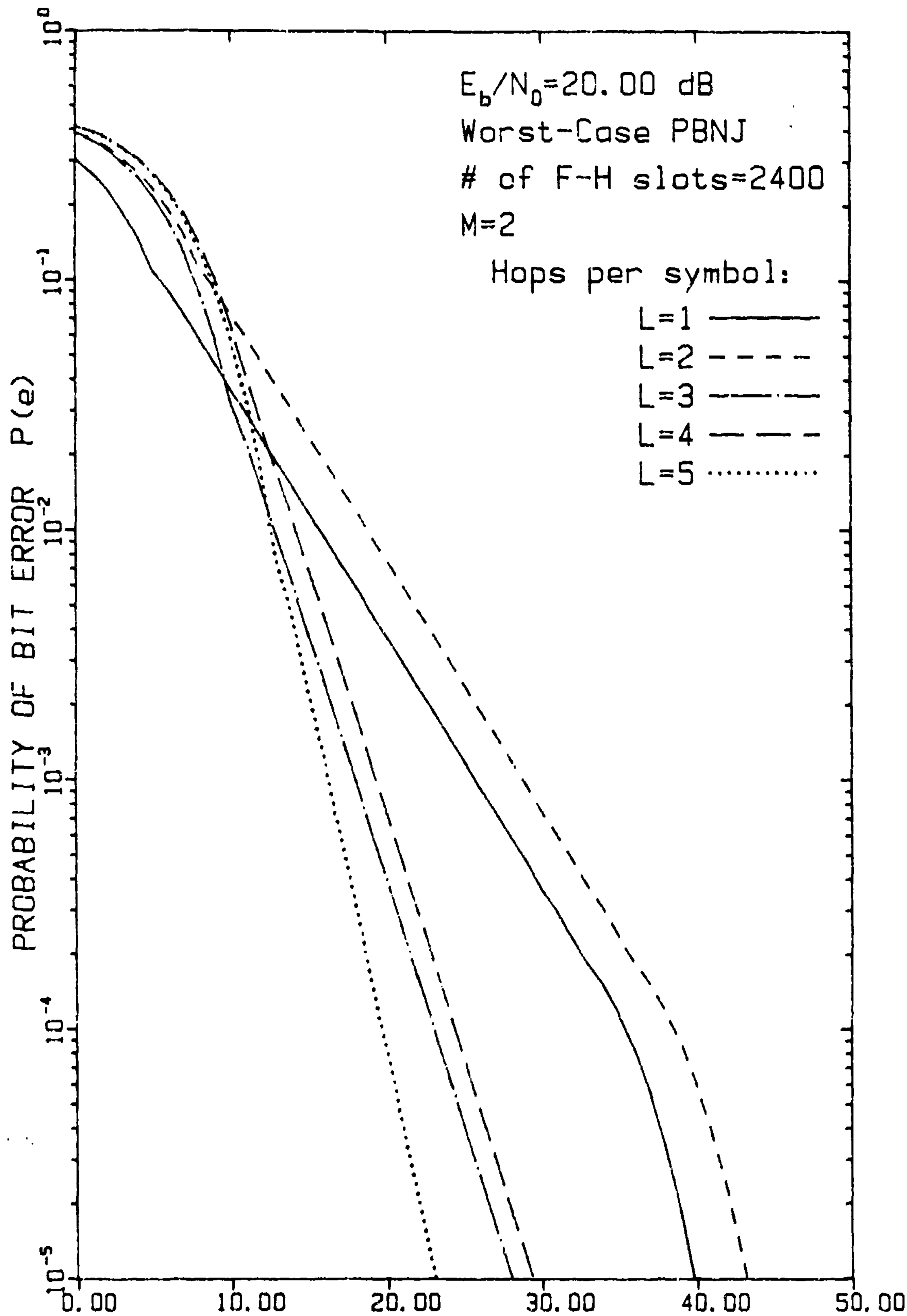


FIGURE 3.2-15 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=2 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=20 dB (FOR MINIMIZATION OF THERMAL NOISE EFFECT)

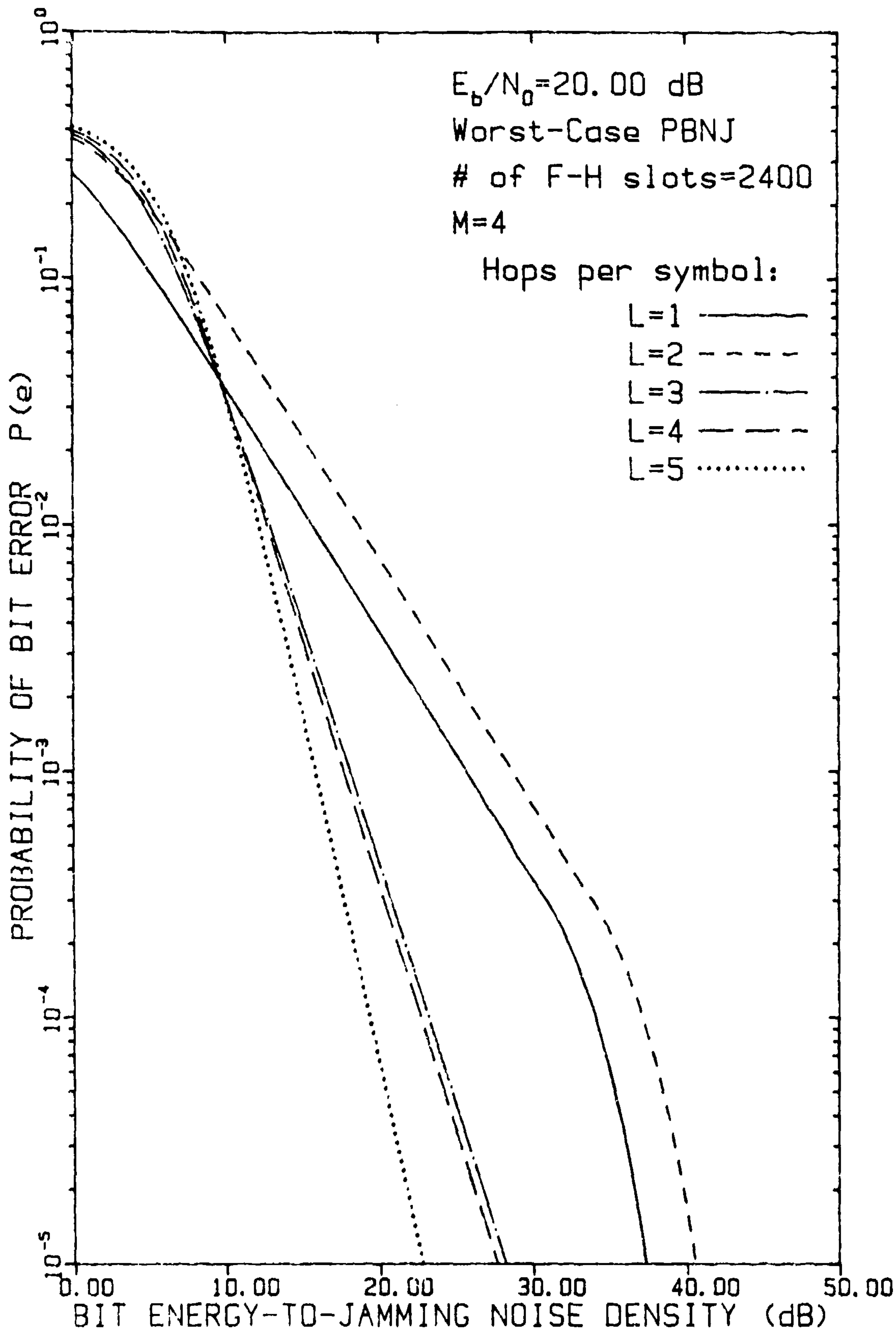


FIGURE 3.2-16 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=4 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=20 dB (FOR MINIMIZATION OF THERMAL NOISE EFFECT)

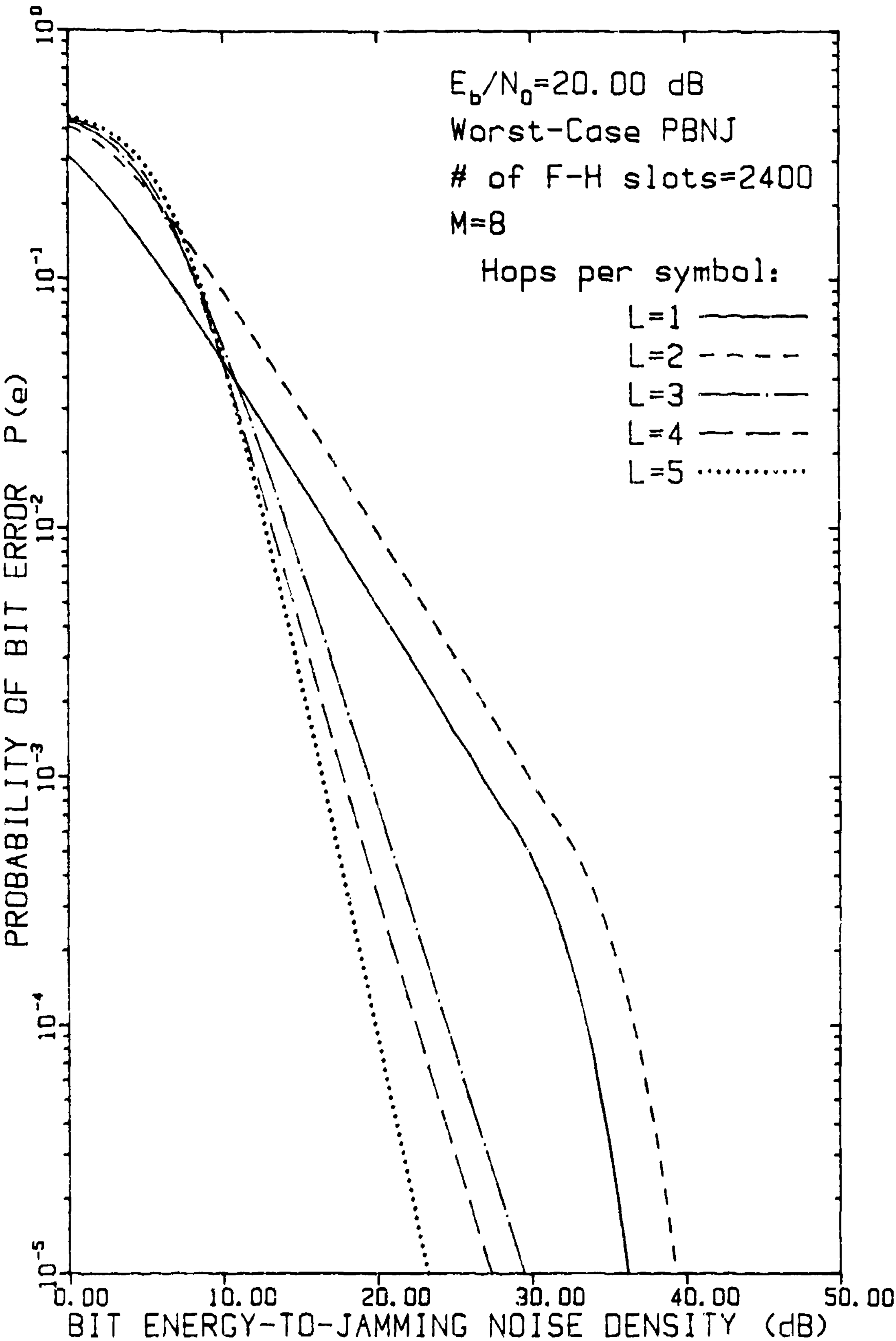


FIGURE 3.2-17 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=8 IN THE PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING AT E_b/N_0=20 dB (FOR MINIMIZATION OF THERMAL NOISE EFFECT)

The case of M=2 is an exception to this trend for L=3 and L=4 as explained in subsection 2.4.2.3 where L=4 is shown to be 1.25 dB worse than that for L=3.

Thus we conclude that when thermal noise is minimized, a form of diversity improvement does exist in all M cases for E_b/N_J values greater than around 12 dB for L>2 hops per symbol.

We now determine the optimum number of jammed slots (Q_{max}) which yields the maximum probability of error for a given value of E_b/N_J. Figures 3.2-18 to 3.2-20 show such plots for case of M=2,4, and 8 with L values ranging from one to five. It is seen that in all cases a definite ascending order of the L Q_{max} curves exists for increasing E_b/N_J values as is to be expected for worst-case jamming calculations. For example, in Figure 3.2-18 (M=2) we see that at a 30 dB E_b/N_J value, the Q_{max} value is 2 for L=1 and over 200 for L=5. Also in Figure 3.2-18 we note the "plateau" effect for all the curves at Q_{max} equal to 2400. Now the breakpoint at which each individual L-curve falls off from the "plateau" represents that E_b/N_J value for which full-band jamming (γ=1.0) will not cause maximum probability of error. We can also characterize each L-curve behavior of Figure 3.2-18 as per three definite regions with respect to the "slope" of each curve. These regions, in terms of Q_{max} values, are: (1) 2400 to about 900, (2) 900 down to approximately 20, and (3) below 20. Note that distinguishable breaks in the curves below 20 are due to the smaller quantized values of Q becoming more discernable for the lower values of the logarithmic Q_{max} scale.

With regard to Figures 3.2-19 (M=4) and 3.3-20 (M=8), we see an asymptotic merging of the L-curves within the region of approximately Q_{max} = 800 to 2400. Below a Q_{max} of about 800 we have two more noticeable regions

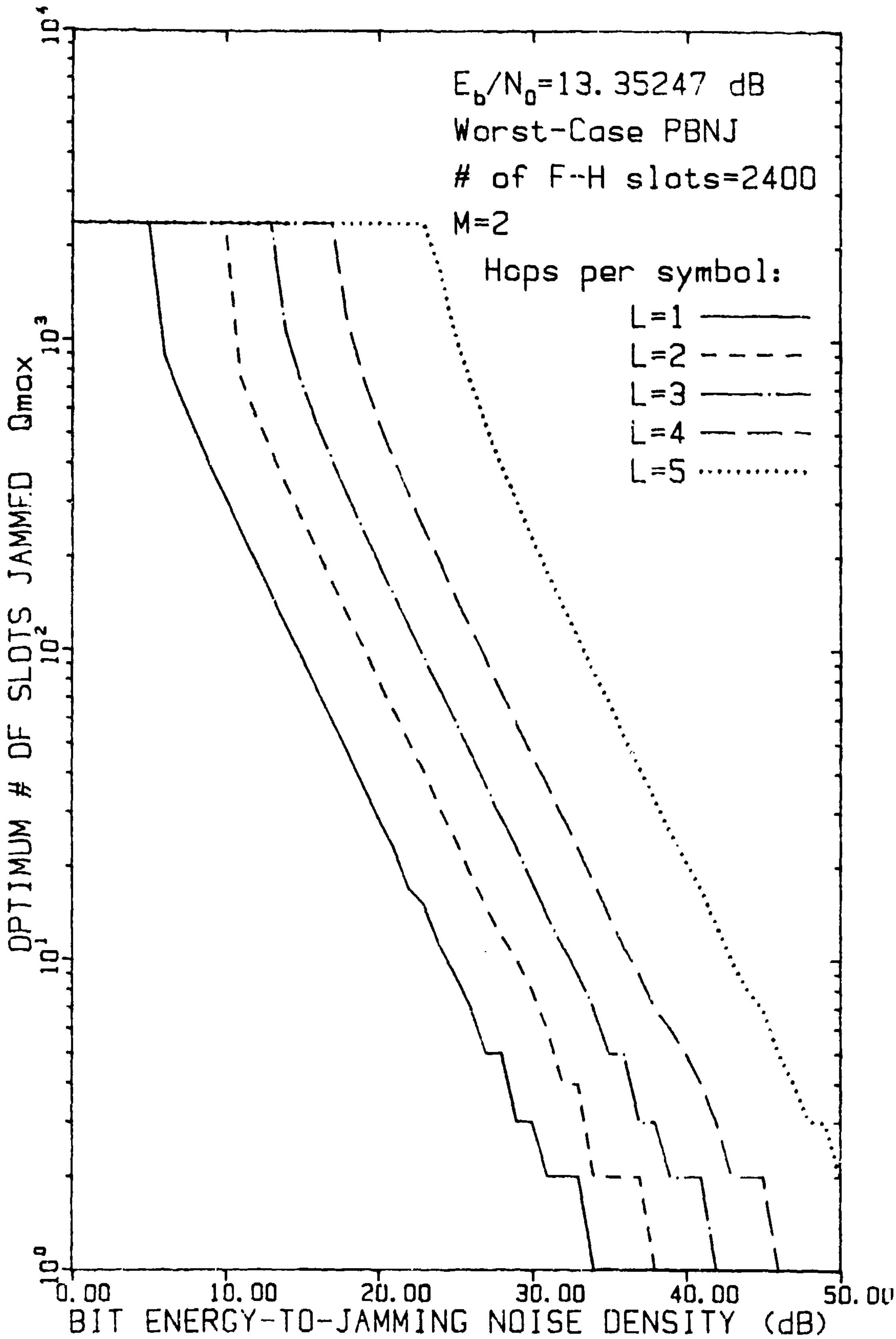


FIGURE 3.2-18 OPTIMUM NUMBER OF HOPPING SLOTS JAMMED (Q_{max} THAT PRODUCES $P(e)_{max}$) VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR THE SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=2 AT E_b/N_0=13.35246 dB (FOR 10^{-5} BER WHEN L=1)

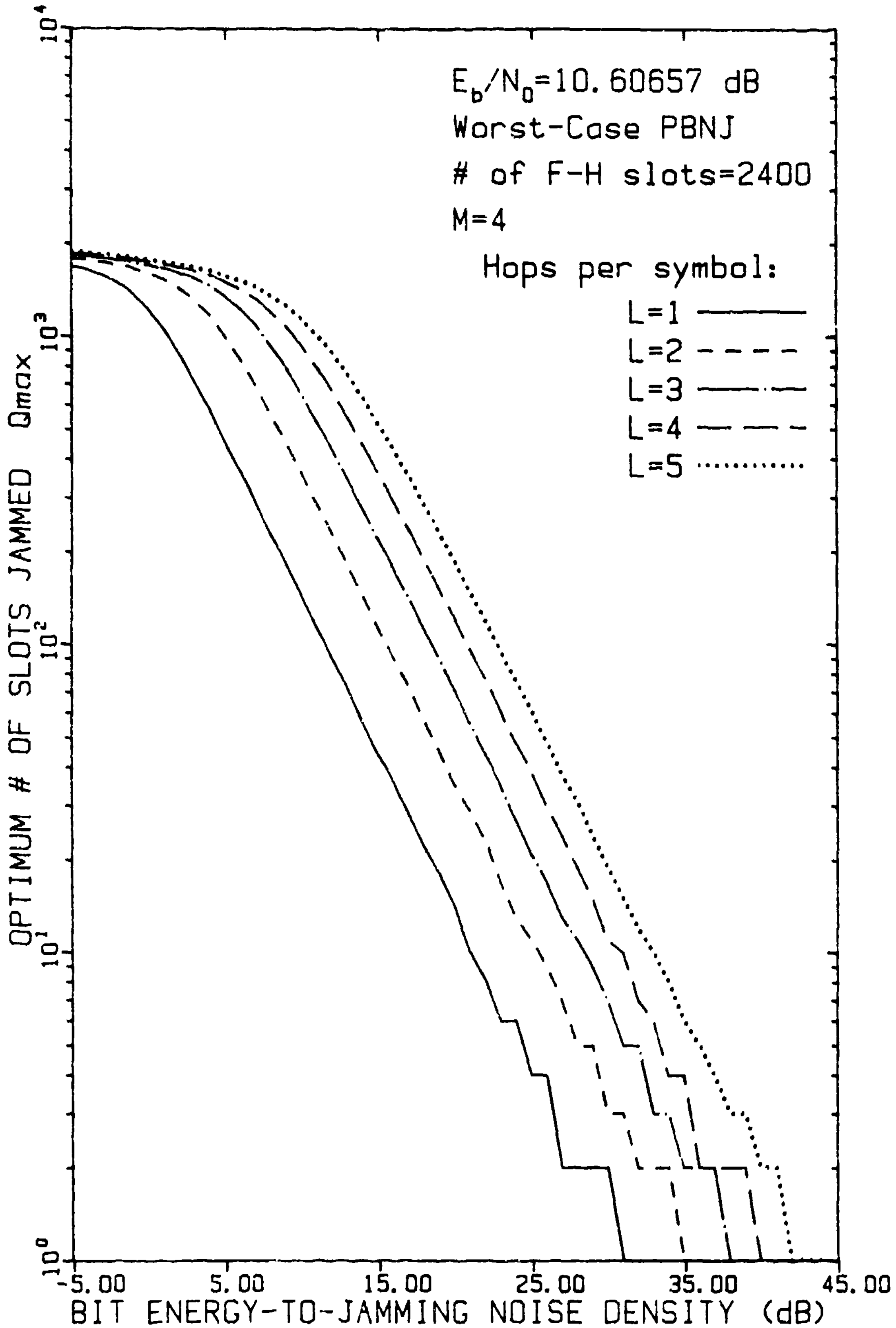


FIGURE 3.2-19 OPTIMUM NUMBER OF HOPPING SLOTS JAMMED (Q_{max} THAT PRODUCES $P(e)_{max}$) VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR THE SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=4 AT E_b/N_0=10.60657 dB (FOR 10^{-5} BER WHEN L=1)

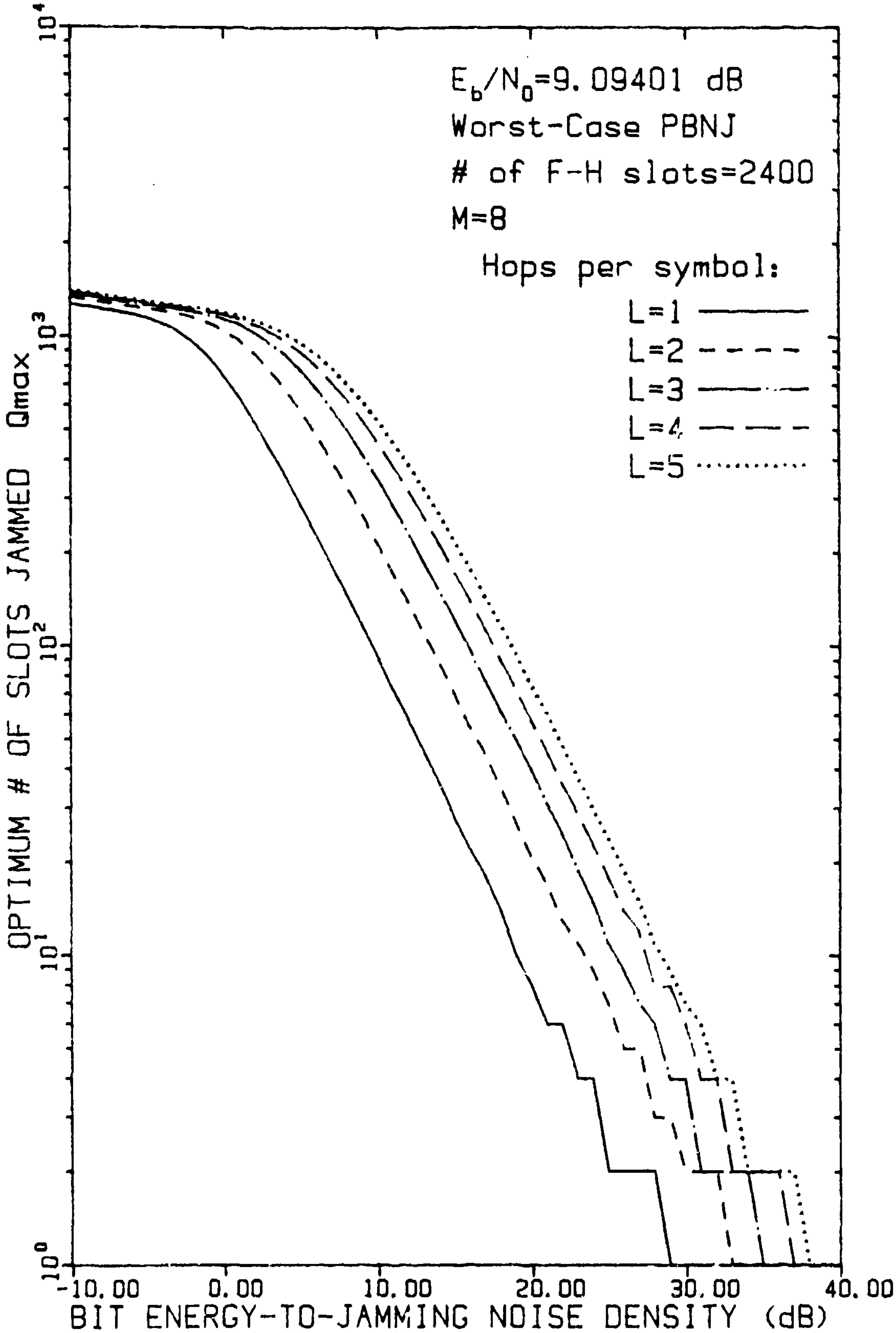


FIGURE 3.2-20 OPTIMUM NUMBER OF HOPPING SLOTS JAMMED (Q_{max} THAT PRODUCES $P(e)_{max}$) VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR THE SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=8 AT E_b/N_0=9.09401 dB (FOR 10^{-5} BER WHEN L=1)

similar to those in Figure 3.2-18. Additionally, we see that this asymptotic merging has yet to reach a full-band value of Q_{max}=2400 for the minimum E_b/N_J value utilized in the calculations. Further computations (not shown) for lower values of E_b/N_J reveal the following points at which the L=1 curve breaks from the Q_{max}=2400 (γ=1.0) value: M=4 at -149.0 dB, M=8 at -150.0 dB. Hence, in these cases, full-band jamming would only be optimum for a very large amount of available jamming power.

A final point of interest is shown in Figure 3.2-21 with respect to minimization of the thermal noise component. Here we see that for the binary (M=2) case, increasing the E_b/N_0 value over that used in Figure 3.2-18 causes the L=5 curve to move around 10 dB (E_b/N_J) lower while the L=1 curve decreases only about 1 dB.

The results indicate that the hard symbol decision receiver can be considered an ECCM receiver (for sufficiently high E_b/N_0), while the linear combining receiver cannot. Therefore, it is not diversity as such that yields an ECCM effect, but the combining technique. Hard decisions (a form of repetition coding) in effect limit the jamming effects on a given hop to that hop, whereas with linear combining a single, strongly jammed hop can dominate the soft decision.

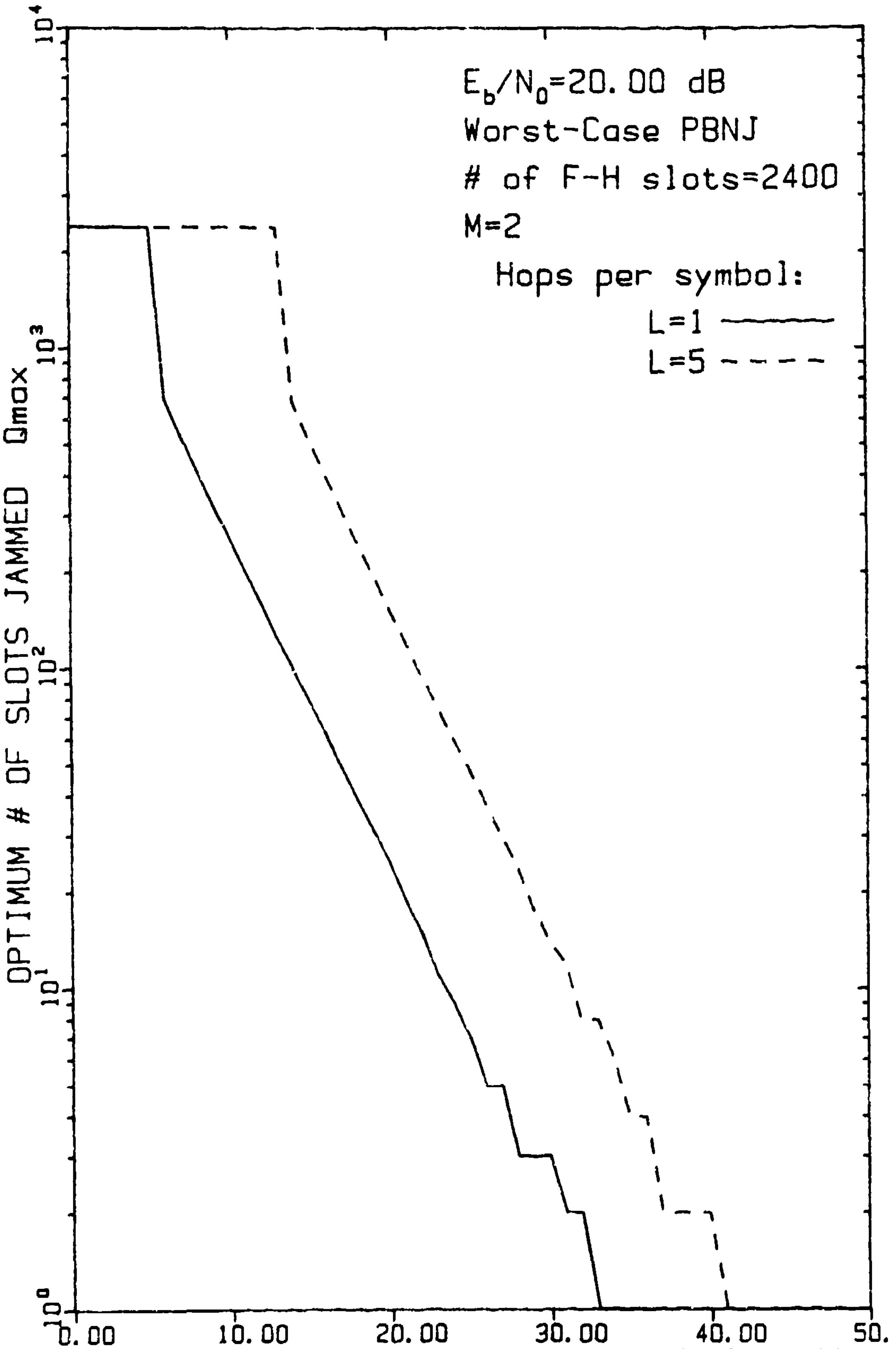


FIGURE 3.2-21 OPTIMUM NUMBER OF HOPPING SLOTS JAMMED (Q_{max} THAT PRODUCES $P(e)_{max}$) VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR THE SQUARE-LAW COMBINING RECEIVER WITH HARD DECISIONS FOR M=2 AND L=1,5 AT E_b/N_0=20 dB (FOR MINIMIZATION OF THERMAL NOISE)

4.0 FH/RMFSK PERFORMANCE USING SQUARE-LAW AGC RECEIVER

In this section we consider the case of the generic receiver shown in Figure 2.2-1 when the envelope samples are processed using the function

$$f(x_{mk}) = x_{mk}^2 \cdot w_{mk} = z_{mk} \quad . \tag{4.0-1}$$

That is, the decision statistics $\{z_m\}$ are weighted sums of samples of the squared envelope in each channel over multiple (L) hops.

For conventional FH/MFSK, where the symbol frequency slots are hopped together, it was assumed in [1] that all the slots are jammed or all the slots are not jammed on a given hop, and the weights were taken to be

$$w_{mk} = 1/\sigma_k^2 = \begin{cases} 1/\sigma_N^2 \text{ , hop not jammed} \\ 1/\sigma_T^2 \text{ , hop jammed.} \end{cases} \tag{4.0-2}$$

This weighting or normalization scheme was predicated on use of a separate channel, or perhaps a look-ahead scheme, to measure the noise power (perfectly) on each hop. The effect of the weighting is to de-emphasize the jammed hops in the summations

$$z_m = \sum_{k=1}^{L} z_{mk} \text{ ,} \tag{4.0-3}$$

and therefore to mitigate the effect of the jamming on the symbol decision.

For FH/RMFSK, in general the different symbol frequency slots are independently jammed or not jammed when the system bandwidth contains power from a partial-band noise jammer. In Section 4.1, we discuss several normalization schemes of the AGC (adaptive gain control) type. The performances of two of these schemes are analyzed in Sections 4.2 and 4.3. We also consider the effect of hard-limiting the hop statistics $\{z_{mk}\}$ prior to combining.

4.1 POSSIBLE AGC WEIGHTING SCHEMES FOR FH/RMFSK

The squared envelope samples in the M receiver channels on a given hop are weighted chi-squared random variables:

$$x_{1k}^2 \sim \sigma_{1k}^2 \; \chi^2(2;\ \lambda_{1k} = 2S/\sigma_{1k}^2)$$

$$= \ (\sigma_N^2 + \nu_{1k}\,\sigma_J^2)\ \chi^2[2;\ 2S/(\sigma_N^2 + \nu_{1k}\sigma_J^2)] \tag{4.1-1a}$$

in the signal channel, and

$$x_{mk}^2 \ \sim \sigma_{mk}^2\ \chi^2(2) = (\sigma_N^2 + \nu_{mk}\sigma_J^2)\ \chi^2(2),\ m > 1, \tag{4.1-1b}$$

in the non-signal channels, where

$$\nu_{mk} = \begin{cases} 1, \text{ channel m jammed on hop k} \\ 0, \text{ channel m not jammed on hop k.} \end{cases} \tag{4.1-2}$$

We shall consider three approaches to AGC normalization, as illustrated in Figure 4.1-1:

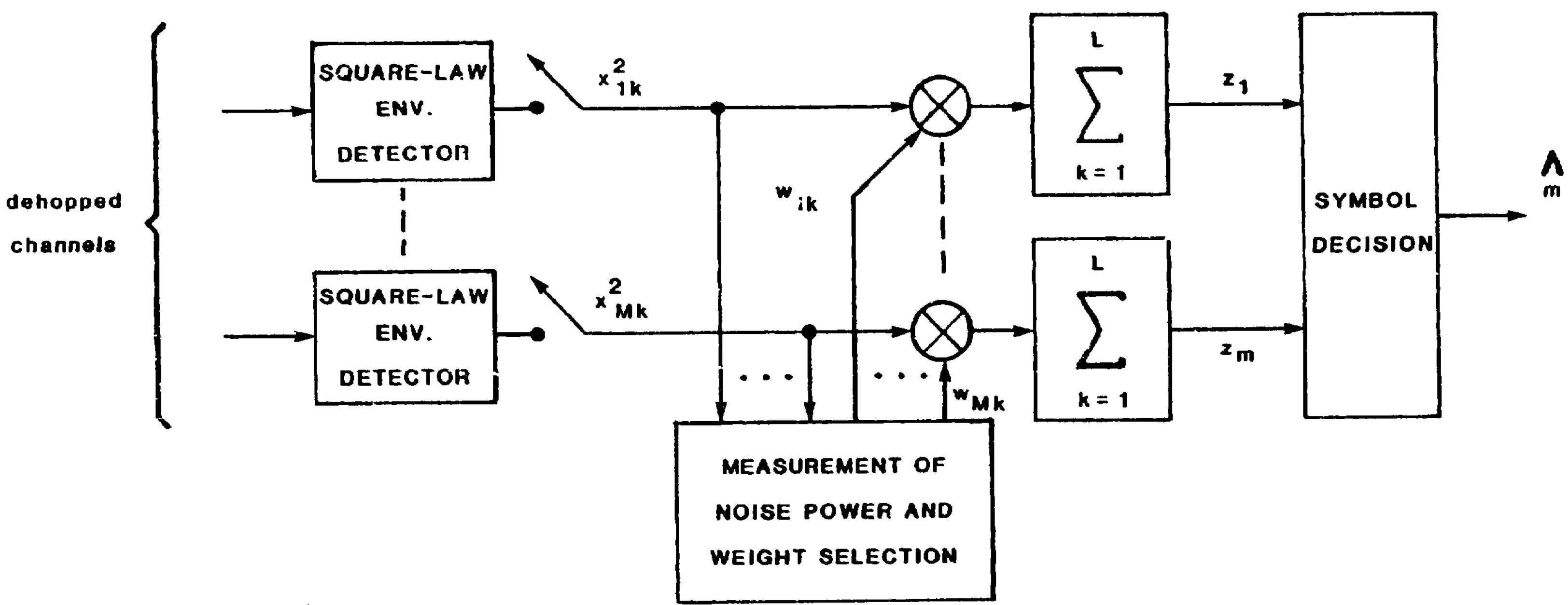
(a) Measurement of noise power in the M de-hopped channels on each hop and normalization (division) by the average received noise power (variance) in these channels.

(b) Individual measurement and normalization of each of the M channels on each hop.

(c) Normalization of the M channels by the same amount, depending on whether one or more of the channels are jammed, or none are jammed.

4.1.1 Average AGC Scheme.

An ideal measurement of average noise power in the M channels would yield the weights



Weighting schemes:

(a) Average AGC approach:

$$w_{mk} = \left[\frac{1}{2} E \left\{ \sum_{m=1}^{M} x^2_{Mk} - S \right\} \right]^{-1} = \left[\frac{1}{M} \sum_{m=1}^{M} \sigma^2_{mk} \right]^{-1} = w_k$$

(b) Individual-channel AGC approach: $w_{mk} = \left(\sigma^2_{mk} \right)^{-1}$

(c) Any-channel-jammed AGC approach: $w_{mk} = w_k = \begin{cases} \left(\sigma_N{}^2 \right)^{-1} & \text{no channels jammed} \\ \left(\sigma_T{}^2 \right)^{-1} & \text{one or more channels jammed} \end{cases}$

FIGURE 4.1-1 POSSIBLE AGC NORMALIZATION SCHEMES

$$w_{mk} \equiv w_k = \left(\frac{1}{M}\sum_{m=1}^{M} \sigma_{mk}^2\right)^{-1}$$

$$= \left(\sigma_N^2 + \sigma_J^2 \sum_{m=1}^{M} \nu_{mk}/M\right)^{-1}. \tag{4.1-3}$$

There are M possible values to these weights. After normalization, the $\{z_{mk}\}$ become

$$z_{1k} \sim w_k \sigma_{1k}^2 \; \chi^2(2;2S/\sigma_{1k}^2) \tag{4.1-4a}$$

$$z_{mk} \sim w_k \sigma_{mk}^2 \; \chi^2(2), \quad m > 1. \tag{4.1-4b}$$

The effective weights $W_k \equiv w_k \sigma_{mk}^2$ on the chi-squared variables in a given channel can take 2(M-1) values. Therefore the decision statistics for this normalization scheme have the form

$$z_m \sim \sum_{k=1}^{L} W_k \chi^2(2;\lambda_{mk}). \tag{4.1-5}$$

The distribution of sums of non-equally weighted chi-squared random variables is extremely difficult to compute. For this reason, it is not feasible to consider calculation of the error performance using such a weighting scheme.

4.1.2 Individual Channel AGC Scheme.

Ideal measurements of noise power in each of the M channels would yield the weights

$$w_{mk} = (\sigma_{mk}^2)^{-1} \tag{4.1-6}$$

and the decision statistics

$$z_1 = \sum_{k=1}^{L} \chi^2(2;\lambda_{1k}) = \chi^2(2L;\lambda_1) \tag{4.1-7a}$$

in the signal channel, where

$$\lambda_1 = \sum_{k=1}^{L} \lambda_{1k} = (L-\ell_1)\cdot 2S/\sigma_N^2 + \ell_1\cdot 2S/\sigma_T^2 , \tag{4.1-7b}$$

and in the non-signal channels,

$$z_m = \sum_{k=1}^{L} \chi^2(2) = \chi^2(2L). \tag{4.1-7c}$$

Thus whatever else the merits of this normalization scheme may be, it yields decision statistics which are purely chi-squared random variables with 2L degrees of freedom. In fact, from (4.1-7) we observe that the discernable jamming events are characterized solely by the number of hops jammed in the signal channel, ℓ_1.

4.1.3 Any-Channel-Jammed AGC Scheme.

This scheme takes the approach that if any of the channels on a given hop is jammed, then all the channels are normalized by $\sigma_T^2 = \sigma_N^2 + \sigma_J^2$; otherwise they are all normalized by σ_N^2. Expressed mathematically, the weights are

$$w_{mk} \equiv w_k = \begin{cases} (\sigma_N^2)^{-1}, & \sum_{m=1}^{M} \nu_{mk} = 0 \\ (\sigma_T^2)^{-1}, & \text{otherwise.} \end{cases} \tag{4.1-8}$$

The result of this approach is that the hop statistics fall into three categories:

(a) channel not jammed, normalized by σ_N^2

(b) channel not jammed, normalized by σ_T^2

(c) channel jammed, normalized by σ_T^2.

If we define ℓ_0 as the number of hops on which at least one of the channels is jammed, we find that in channel m (m=1,2,...,M) there would be

* $(L-\ell_0)$ hops with noise power σ_N^2 normalized by σ_N^2
* $(\ell_0-\ell_m)$ hops with noise power σ_N^2 normalized by σ_T^2
* ℓ_m hops with noise power σ_T^2 normalized by σ_T^2.

Therefore, for a given ℓ_0 and jamming event vector $\underline{\ell} = (\ell_1,\ell_2,\ldots,\ell_M)$, the decision statistics would have the following distributions:

$$z_1 \sim \chi^2[2(L-\ell_0);2(L-\ell_0)S/\sigma_N^2] + (\sigma_N^2/\sigma_T^2)\chi^2[2(\ell_0-\ell_1);2(\ell_0-\ell_1)S/\sigma_N^2]$$

$$+ \chi^2[2\ell_1;2\ell_1 S/\sigma_T^2]$$

$$= \chi^2[2(L+\ell_1-\ell_0);2(L-\ell_0)S/\sigma_N^2 + 2\ell_1 S/\sigma_T^2]$$
$$+ K^{-1}\chi^2[2(\ell_0-\ell_1);2(\ell_0-\ell_1)S/\sigma_N^2],\ \ell_1 \neq \ell_0; \qquad (4.1\text{-}9a)$$

and

$$z_m \sim \chi^2[2(L-\ell_0)] + (\sigma_N^2/\sigma_T^2)\chi^2[2(\ell_0-\ell_m)] + \chi^2(2\ell_m)$$

$$= \chi^2[2(L+\ell_m-\ell_0)] + K^{-1}\chi^2[2(\ell_0-\ell_m)],\ m > 1;\ \ell_m \neq \ell_0. \qquad (4.1\text{-}9b)$$

When $\ell_m = \ell_0$ the case of noise power σ_N^2 normalized by σ_T^2 does not occur and the distributions are:

$$z_1 \sim \chi^2[2L;2(L-\ell_0)S/\sigma_N^2 + 2\ell_0 S/\sigma_T^2],\ \ell_1 = \ell_0; \qquad (4.1\text{-}9c)$$

and

$$z_m \sim \chi^2(2L),\ \ell_m = \ell_0. \qquad (4.1\text{-}9d)$$

As in Section 3, we use $K \triangleq \sigma_T^2/\sigma_N^2 > 1$. We see from (4.1-9) that the decision statistics are in general sums of two unequally weighted chi-squared random variables. Analysis of this distribution is difficult but has been accomplished previously, in Section 3. There is the additional complexity, however, that the jamming events now must be described by an additional parameter: ℓ_0, the number of hops with at least one channel jammed. This task can be achieved as shown below.

4.2 ANALYSIS OF FH/RMFSK PERFORMANCE USING INDIVIDUAL CHANNEL AGC SCHEME

Now we obtain the probability of bit error for the FH/RMFSK receiver using the individual channel AGC normalization scheme.

4.2.1 Conditional Probability Of Error.

The probability of a symbol error, given a jamming event described by $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$, is

$$P_s(e|\underline{\ell}) = 1 - P_s(C|\underline{\ell})$$

$$= 1 - \int_0^\infty d\alpha \, p_{z_1}(\alpha) \prod_{m=2}^{M} \int_0^\alpha d\beta_m \, p_{z_m}(\beta_m). \quad (4.2\text{-}1)$$

From (4.1-7) we observe that the non-signal channel decision statistics $\{z_m, m>1\}$ are identically distributed as chi-squared random variables with 2L degrees of freedom. Thus

$$\int_0^\alpha d\beta_m p_{z_m}(\beta_m) = \int_0^\alpha d\beta p_{z_2}(\beta), \quad m = 2,3,\ldots,M$$

$$= 1 - \frac{\Gamma(L;\alpha/2)}{\Gamma(L)} \quad (4.2\text{-}2a)$$

$$= 1 - e^{-\alpha/2} \sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!}, \quad (4.2\text{-}2b)$$

and

$$P_s(C|\underline{\ell},\alpha) \triangleq \prod_{m=2}^{M} \int_0^\alpha d\beta_m \, p_{z_m}(\beta_m) = \left[1 - e^{-\alpha/2} \sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!} \right]^{M-1}$$

$$= \sum_{k=0}^{M-1} \binom{M-1}{k} (-1)^k e^{-k\alpha/2} \left[\sum_{r=0}^{L-1} \frac{(\alpha/2)^r}{r!} \right]^k. \quad (4.2\text{-}3)$$

From Section 3, equation (3.1-14), we find that

$$P_s(C|\underline{\ell},\alpha) = \sum_{k=0}^{M-1} \binom{M-1}{k} (-1)^k e^{-k\alpha/2} \sum_{r=0}^{k(L-1)} \frac{C(r,k)}{r!} \left(\frac{\alpha}{2}\right)^r, \tag{4.2-4a}$$

where

$$C(0,k) = 1$$
$$C(r,k) = \frac{1}{r} \sum_{n=1}^{\min(r,L-1)} \binom{r}{n} [(k+1)n - r] \; C(r-n,k), \quad r>0, L\geq 2. \tag{4.2-4b}$$

Substitution in (4.2-1) yields

$$P_s(e|\underline{\ell}) = 1 - \sum_{k=0}^{M-1} \binom{M-1}{k} (-1)^k \sum_{r=0}^{k(L-1)} \frac{C(r,k)}{r!} \int_0^\infty d\alpha \; p_{z_1}(\alpha) e^{-k\alpha/2} \; (\alpha/2)^r . \tag{4.2-5}$$

From (4.1-7) and (3.1-9) the pdf for z_1 is

$$p_{z_1}(\alpha) = \frac{1}{2} e^{-(\alpha+\lambda_1)/2} \left(\frac{\alpha}{\lambda_1}\right)^{(L-1)/2} I_{L-1}\left(\sqrt{\alpha\lambda_1}\right), \tag{4.2-6}$$

with $\lambda_1 = 2(L-\ell_1)\rho_N + 2\ell_1\rho_T$. Thus the required integral in (4.2-5) is

$$\begin{aligned} &\left(\frac{1}{1+k}\right)^{r+L} \int_0^\infty dx \left(\frac{x}{2}\right)^r \exp\left(-\frac{k\lambda_1/2}{k+1}\right) \; p_{\chi^2}\left(x;2L, \; \lambda = \frac{\lambda_1}{1+k}\right) \\ &\quad = \left(\frac{1}{1+k}\right)^{r+L} \exp\left(-\frac{k\lambda_1/2}{k+1}\right) r! \quad \mathcal{L}_r^{L-1}\left(\frac{\lambda_1/2}{k+1}\right), \end{aligned} \tag{4.2-7}$$

giving the error probability

$$P_s(e|\underline{\ell}) = \sum_{k=1}^{M-1} \binom{M-1}{k} \frac{(-1)^{k+1}}{(1+k)^L} \sum_{r=0}^{k(L-1)} \frac{C(r,k)}{(1+k)^r} \exp\left(-\frac{k\lambda_1/2}{k+1}\right) \mathcal{L}_r^{L-1}\left(\frac{-\lambda_1/2}{1+k}\right) \qquad (4.2\text{-}8)$$

$$\equiv P_s(e|\ell_1) \ .$$

4.2.2 Total Error Probability.

Since the symbol error probability depends only on whether the signal channel is jammed, the total bit error probability is

$$P_b(e) = \sum_{\ell_1=0}^{L} \binom{L}{\ell_1} \gamma^{\ell_1} (1-\gamma)^{L-\ell_1} \frac{M/2}{M-1} P_s(e|\ell_1), \qquad (4.2\text{-}9a)$$

where

$$\gamma = \Pr\{\text{channel 1 jammed on hop k}\} \ . \qquad (4.2\text{-}9b)$$

Noting that (4.2-8) and (4.2-9) are mathematically identical to equations (4-26) in [1], we observe that MFSK and RMFSK give equivalent PBNJ performances for individual channel AGC normalization.

4.3 ANALYSIS OF FH/RMFSK PERFORMANCE USING ANY-CHANNEL-JAMMED AGC SCHEME

In what follows we find the probability of bit error for the FH/RMFSK receiver using the any-channel-jammed AGC normalization scheme (ACJ).

4.3.1 Conditional Probability Of Error.

The probability of a symbol error, given the jamming event $(\ell_0, \underline{\ell})$, is

$$P_s(e|\ell_0,\underline{\ell}) = 1 - P_s(C|\ell_0,\underline{\ell})$$

$$= 1 - \int_0^\infty d\alpha \ \ p_{z_1}(\alpha;\ell_0,\ell_1) \prod_{m=2}^{M} \int_0^\alpha d\beta_m \ p_{z_m}(\beta_m;\ell_0,\ell_m). \qquad (4.3\text{-}1)$$

Since $P_s(C|\ell_0,\underline{\ell}) = \Pr\{z_2<z_1, z_3<z_1, \ldots, z_M<z_1\}$

$$= \Pr\{Kz_2<Kz_1,\ Kz_3<Kz_1, \ldots, Kz_M<Kz_1\}\ , \tag{4.3-2}$$

we may analyze the error probability using the statistics $\{u_m\}$ instead of $\{z_m\}$ where

$$u_1 \equiv Kz_1 = \chi^2[2(\ell_0-\ell_1);2(\ell_0-\ell_1)\rho_N] + K\chi^2[2(L+\ell_1-\ell_0);2(L-\ell_0)\rho_N + 2\ell_1\rho_T] \tag{4.3-3a}$$

and

$$u_m \equiv Kz_m = \chi^2[2(\ell_0-\ell_m)] + K\cdot\chi^2[2(L+\ell_m-\ell_0)] \quad m>1. \tag{4.3-3b}$$

From Appendix A, the pdf's of these random variables are as follows: for the signal channel,

$$p_{u_1}(\alpha) = \begin{cases} p_{\chi^2}(\alpha;2L,2L\rho_N), \quad \ell_0-\ell_1 = L & (4.3\text{-}4a) \\ \frac{1}{K}\ p_{\chi^2}[\alpha/K;2L,2(L-\ell_0)\rho_N + 2\ell\rho_T],\ \ell_0-\ell_1 = 0; & (4.3\text{-}4b) \\ \sum_{n=0}^{\infty} c_n\ p_{\chi^2}[\alpha;2L+2n,2(\ell_0-\ell_1)\rho_N],\ 0<\ell_0-\ell_1<L; & (4.3\text{-}4c) \end{cases}$$

where

$$c_n = e^{-(L-\ell_0)\rho_N-\ell_1\rho_T}\left(\frac{K-1}{K}\right)^n\left(\frac{1}{K}\right)^{L-\ell_0+\ell_1}\mathcal{L}_n^{L-\ell_0+\ell_1-1}\left[\frac{-(L-\ell_0)\rho_N\ell_1\rho_T}{K-1}\right]\cdot \tag{4.3-4d}$$

For the nonsignal channels (m>1),

$$p_{u_m}(\alpha) = \begin{cases} p_{\chi^2}(\alpha;2L), \quad \ell_0-\ell_m = L & (4.3\text{-}5a) \\ \frac{1}{K}\, p_{\chi^2}(\alpha/K;2L), \quad \ell_0-\ell_m = 0 & (4.3\text{-}5b) \\ \sum_{n=0}^{\infty} b_n\, p_{\chi^2}(\alpha;2L+2n), \quad 0 < \ell_0-\ell_m < L & (4.3\text{-}5c) \end{cases}$$

using the coefficients

$$b_n = \left(\frac{K-1}{K}\right)^n \left(\frac{1}{K}\right)^{L-\ell_0+\ell_m} \binom{n+L-\ell_0+\ell_m-1}{n} \qquad (4.3\text{-}5d)$$

The symbol error probability expression for the $\{u_m\}$ statistics is (4.3-1) with the subscripts $\{u_m\}$ instead of $\{z_m\}$, $m=1,2,\ldots,M$.

4.3.1.1 Formulations of nonsignal channel probabilities

Since the nonsignal channel pdf's are identical except for the parameters $\{\ell_m\}$, the number of hops jammed in the individual channels, we may express the product

$$\prod_{m=2}^{M} \int_0^{\alpha} d\beta_m\, p_{u_m}(\beta_m;\ell_0,\ell_m) \prod_{m=2}^{M} \Pr\{u_m<\alpha\} \qquad (4.3\text{-}6)$$

in terms of the numbers of channels with certain combinations of ℓ_0 and the $\{\ell_m\}$. The probabilities needed are

$$\Pr\{u_m<\alpha\} = \begin{cases} 1 - \Gamma(L;\alpha/2)/\Gamma(L), \quad \ell_0-\ell_m = L & (4.3\text{-}7a) \\ 1 - \Gamma(L;\alpha/2K)/\Gamma(L), \quad \ell_0-\ell_m = 0 & (4.3\text{-}7b) \\ 1 - \sum_{n=0}^{\infty} b_n \Gamma(L+n;\alpha/2)/\Gamma(L+n), \quad 0<\ell_0-\ell_m<L. & (4.3\text{-}7c) \end{cases}$$

Upon comparing (4.3-7) with (3.1-11), we observe that

$$\Pr\{u_m<\alpha\} \equiv F_L(\alpha;L-\ell_0+\ell_m). \tag{4.3-8}$$

Thus we can write the product in (4.3-6) as

$$\prod_{m=2}^{M} \Pr\{u_m<\alpha\} = \left[F_L(\alpha;0)\right]^{n_0} \left[F_L(\alpha;1)\right]^{n_1} \cdots \left[F_L(\alpha;L)\right]^{n_L} \tag{4.3-9}$$

where

$$n_i \triangleq \#\,(\text{channels with } L-\ell_0+\ell_m = i), \tag{4.3-10}$$

and $\left[F_L(\alpha;p)\right]^{n_p}$ is given by (3.1-14), (3.1-17), and (3.1-28).

4.3.1.2 Formulation of symbol error probability in terms of previous results (Section 3).

If we denote the conditional probability of symbol error for the square-law linear combining FH/RMFSK receiver studied in Section 3 by

$$P_s(e;\rho_N,\rho_T|\underline{\ell})_{LC}\ , \tag{4.3-11}$$

we can by analogy express the conditional probability of symbol error for the any-channel-jammed AGC receiver as

$$P_s(e;\rho_N,\rho_T|\ell_0,\underline{\ell})_{ACJ} = P_s(e;\rho_N,\rho_T'|\underline{v})_{LC} \tag{4.3-12a}$$

where

$$\rho_T' = \begin{cases} \dfrac{\ell_1\rho_T+(L-\ell_0)\rho_N}{L-\ell_0+\ell_1}\ , & \ell_0-\ell_1 \neq L \\ \rho_T\ , & \ell_0-\ell_1 \end{cases} \tag{4.3-12b}$$

and

$$\underline{v} \triangleq (L-\ell_0+\ell_1, L-\ell_0+\ell_2, \ldots, L-\ell_0+\ell_M). \qquad (4.3\text{-}13)$$

4.3.2 Enumeration and Probabilities for Jamming Events.

The enumeration of jamming events, and their probabilities, has already been accomplished in Section 2.2 for the situation in which the jamming event is sufficiently described by the vector $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$. Now our task is to develop for use in (4.3-12) the conditional probabilities $\Pr\{\ell_0|\underline{\ell}\}$ for the parameter ℓ_0, the number of hops on which at least one channel is jammed, given the vector $\underline{\ell}$.

The enumeration technique treated in Section 2.2 recognized the arbitrariness of the channel numbers m for m>1 (nonsignal channels) by assuming that the calculations will generate the partially ordered $\underline{\ell}$ vector

$$\underline{\ell}' = \{(\ell_1, \ell_2, \ldots, \ell_M) : \ell_2 \le \ell_3 \le \ldots \le \ell_M\}. \qquad (4.3\text{-}14)$$

Thus, with this ordering the range of ℓ_0 is

$$\ell_x \triangleq \max(\ell_1, \ldots, \ell_M) \le \ell_0 \le \min(L, \ell_1+\ell_2+\ldots+\ell_M). \qquad (4.3\text{-}15)$$

The number of elementary or $[\nu]$-matrix jamming events characterized by a given $\underline{\ell}$ or $\underline{\ell}'$ vector is

$$\#([\nu]\to\underline{\ell}) = \binom{L}{\ell_1}\binom{L}{\ell_2}\cdots\binom{L}{\ell_M} = \prod_{m=1}^{M}\binom{L}{\ell_m}, \qquad (4.3\text{-}16)$$

and each $\underline{\ell}'$ vector represents

$$\binom{M-1}{n_0, n_1, \ldots, n_L}, \qquad n_k = \#(\ell_m = k),\ m>1 \tag{4.3-17}$$

$\underline{\ell}$ vectors. Thus

$$\sum_{\underline{\ell}'} \binom{M-1}{n_0, n_1, \ldots, n_L} \Pr\{\underline{\ell}'; M, L\} = 1. \tag{4.3-18}$$

Now for jamming events specified by ℓ_0 as well as $\underline{\ell}$, the number of elementary jamming events thus specified can be shown to be

$$\#([\nu] \to \ell_0, \underline{\ell}) \quad \binom{L}{\ell_0} \sum_{r=0}^{\ell_0 - \ell_x} \binom{\ell_0}{r} (-1)^r \prod_{m=1}^{M} \binom{\ell_0 - r}{\ell_m}. \tag{4.3-19}$$

For example, if $\ell_0 = \ell_x = \max(\ell_m)$, there are

$$\binom{L}{\ell_0} \prod_{m=1}^{M} \binom{\ell_0}{\ell_m} \tag{4.3-20}$$

$[\nu]$ events. Summation of (4.3-19) over the values of ℓ_0 given by (4.3-15) can be shown numerically to give (4.3-16). (See Appendix B.4.)

What is needed for evaluation of the ACJ total probability of error are the probabilities of the jamming events $(\ell_0, \underline{\ell}')$ and the number of $(\ell_0, \underline{\ell}')$ events represented by the ordered version. Since $(\ell_0, \underline{\ell}')$ is a subset of $\underline{\ell}$ for any $\underline{\ell}$, and permutations of the nonsignal channel elements of $\underline{\ell}$ do not affect ℓ_0, it is reasonable that (4.3-17) gives the required number. This fact is confirmed by the consideration that

$$\Pr\{\underline{\ell}';M,L\} = \sum_{\ell_0} \Pr\{\underline{\ell}',\ell_0;M,L\} \qquad (4.3\text{-}21)$$

which can be substituted in (4.3-18) to show that $\Pr\{\ell_0,\underline{\ell}\}$ must be multiplied by (4.3-17).

The probability of the event $(\ell_0,\underline{\ell})$ is derived in the following manner:

$$\Pr\{\underline{\ell},\ell_0\} = \binom{L}{\ell_0} \Pr\{\underline{\nu}_1+\underline{\nu}_2+\ldots+\underline{\nu}_{\ell_0} = \underline{\ell}, \underline{\nu}_1 \neq \underline{0}, \underline{\nu}_2 \neq \underline{0}, \ldots,$$

$$\underline{\nu}_{\ell_0} \neq \underline{0}, \underline{\nu}_{\ell+1} = \underline{0}, \ldots, \underline{\nu}_L = \underline{0}\}$$

$$= \binom{L}{\ell_0} \pi_0^{L-\ell} \Pr\{\underline{\nu}_1+\underline{\nu}_2+\ldots+\underline{\nu}_{\ell_0} = \underline{\ell},\ \underline{\nu}_1 \neq 0, \ldots, \underline{\nu}_{\ell_0} \neq 0\} \qquad (4.3\text{-}22)$$

The probability required in (4.3-22) can be computed using the convolutional method described in Section 2.2.5, modified to give

$$\Pr\{\underline{\nu}_1+\underline{\nu}_2+\ldots+\underline{\nu}_{\ell_0} = \underline{\ell}, \underline{\nu}_1 \neq \underline{0}, \nu_2 \neq 0, \ldots, \underline{\nu}_{\ell_0} \neq 0\}$$

$$= \sum_{\underline{\nu}_1 > \underline{0}} \sum_{\underline{\nu}_2 > \underline{0}} \cdots \sum_{\underline{\nu}_{\ell_0} > \underline{0}} \Pr\{\underline{\nu}_1\}\Pr\{\underline{\nu}_2\} \ldots \Pr\{\underline{\nu}_{\ell_0}\}\delta\left[\underline{\ell} - \sum_{r=1}^{\ell_0} \underline{\nu}_r\right]. \qquad (4.3\text{-}23)$$

This method is useful for M tending to be large; for M=2, it is simpler to recognize that (2.2-24), repeated here as

$$\Pr\{\underline{\ell};2,L\} = \sum_{n=0}^{L} \binom{L}{n, L-\ell_2-n, L-\ell_1-n, \ell_1+\ell_2+n-L}$$

$$\times\ \pi_0^{n}\, \pi_1^{2L-\ell_1-\ell_2-2n}\, \pi_2^{\ell_1+\ell_2-L+n}, \qquad (4.3\text{-}24)$$

is the sum of $\Pr\{\underline{\ell}, \ell_0; 2, L\}$ over ℓ_0 with $n \equiv L-\ell_0$. Therefore,

$$\Pr\{\underline{\ell},\ell_0;2,L\} = \binom{L}{L-\ell_0,\ell_0-\ell_2,\ell_0-\ell_1,\ell_1+\ell_2-\ell_0} \pi_I^{\;2\ell_0-\ell_1-\ell_2} \pi_2^{\;\ell_1+\ell_2-\ell_0} . \quad (4.3\text{-}25)$$

For M=4, we recognize the same principle in the equation given in Table 2.2-4 for $\Pr\{\underline{\ell};4,L\}$ to give the $\Pr\{\ell_0,\underline{\ell};4,L\}$ equation shown in Table 4.3-1.

TABLE 4.3-1 PROBABILITIES OF $(\ell_0, \underline{\ell})$ JAMMING EVENTS FOR M = 4

$$\Pr\{\ell_0, \underline{\ell}; 4, L\} = \underbrace{\sum_{n_1=0}^{L} \sum_{n_2=0}^{L} \cdots \sum_{n_{15}=0}^{L}}_{\text{CONSTRAINTS}} \frac{L!}{(L-\ell_0)!\, n_1! \cdots n_{15}!} \pi_0^{L-\ell_0} \pi_1^{n_1+n_2+n_4+n_8} \pi_2^{n_3+n_5+n_6+n_9+n_{10}+n_{12}} \times \pi_3^{n_7+n_{11}+n_{13}+n_{14}} \pi_4^{n_{15}}$$

CONSTRAINTS:

$$n_1 + n_3 + n_5 + n_7 + n_9 + n_{11} + n_{13} + n_{15} = \ell_1$$

$$n_2 + n_3 + n_6 + n_7 + n_{10} + n_{11} + n_{14} + n_{15} = \ell_2$$

$$n_4 + n_5 + n_6 + n_7 + n_{12} + n_{13} + n_{14} + n_{15} = \ell_3$$

$$n_8 + n_9 + n_{10} + n_{11} + n_{12} + n_{13} + n_{14} + n_{15} = \ell_4$$

$$\sum_{i=1}^{15} n_i = \ell_0$$

4.4 NUMERICAL RESULTS

In this section we present numerical results for the performance of the individual-channel AGC receiver and the any channel-jammed (ACJ) AGC receiver.

4.4.1 Numerical Results for Individual-Channel AGC Receiver

The numerical computations for the performance of the individual-channel AGC receiver were performed using (4.2-8) and (4.2-9). In this particular case, no unusual computational difficulties are encountered in the computations. A listing of the computer program is given in Appendix E.

Figures 4.4-1 through 4.4-4 show the performance of binary (M=2) RMFSK/FH with L=1,2,3, and 4 hops/bit, respectively with the jamming fraction $\gamma=q/N$ as a parameter. We observe that the choice of jamming fraction is critical to the effective operation of the jammer. This is similar to the behavior of the square-law combining receiver. However, unlike the square-law combining receiver, the optimum jamming fraction against the individual-channel AGC receiver is $\gamma=1.0$ over a wider range of E_b/N_J, especially for higher values of L, the number of hops/bit.

Figure 4.4-5 compares the worst-case jamming performance of binary RMFSK/FH as L varies. We observe that over the range of about E_b/N_J=8 dB to E_b/N_J=39 dB, the optimum choice for the communicator is L=2 or 3 hops/bit. However, outside this range L=1 is optimum. In no case does increasing L beyond 3 hops/bit improve the performance.

Figures 4.4-6 through 4.4-9 show the performance of RMFSK/FH when M=4 and L=1,2,3, and 4, respectively. Again, the importance to the jammer of

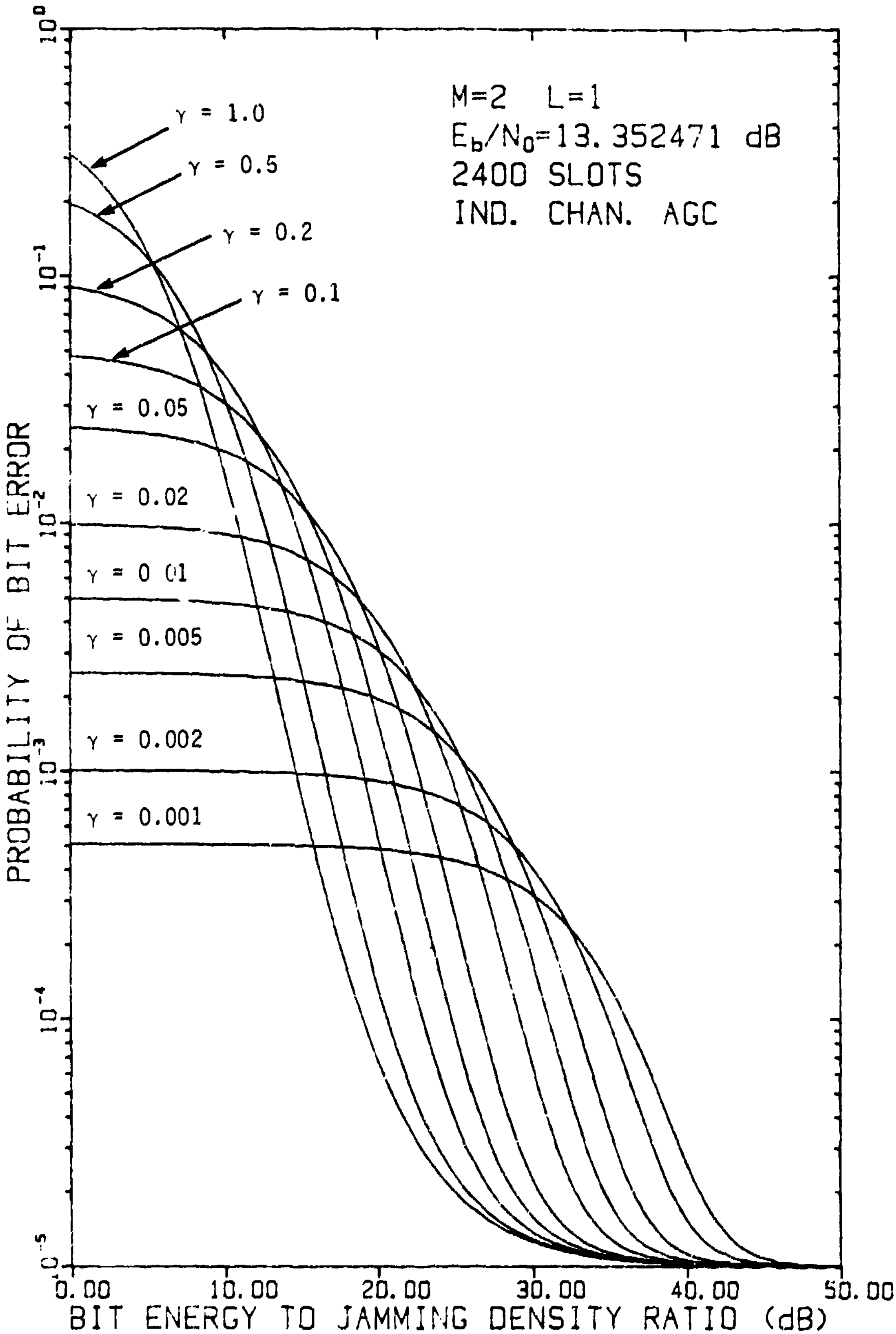


FIGURE 4.4-1 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 2 AND L = 1 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

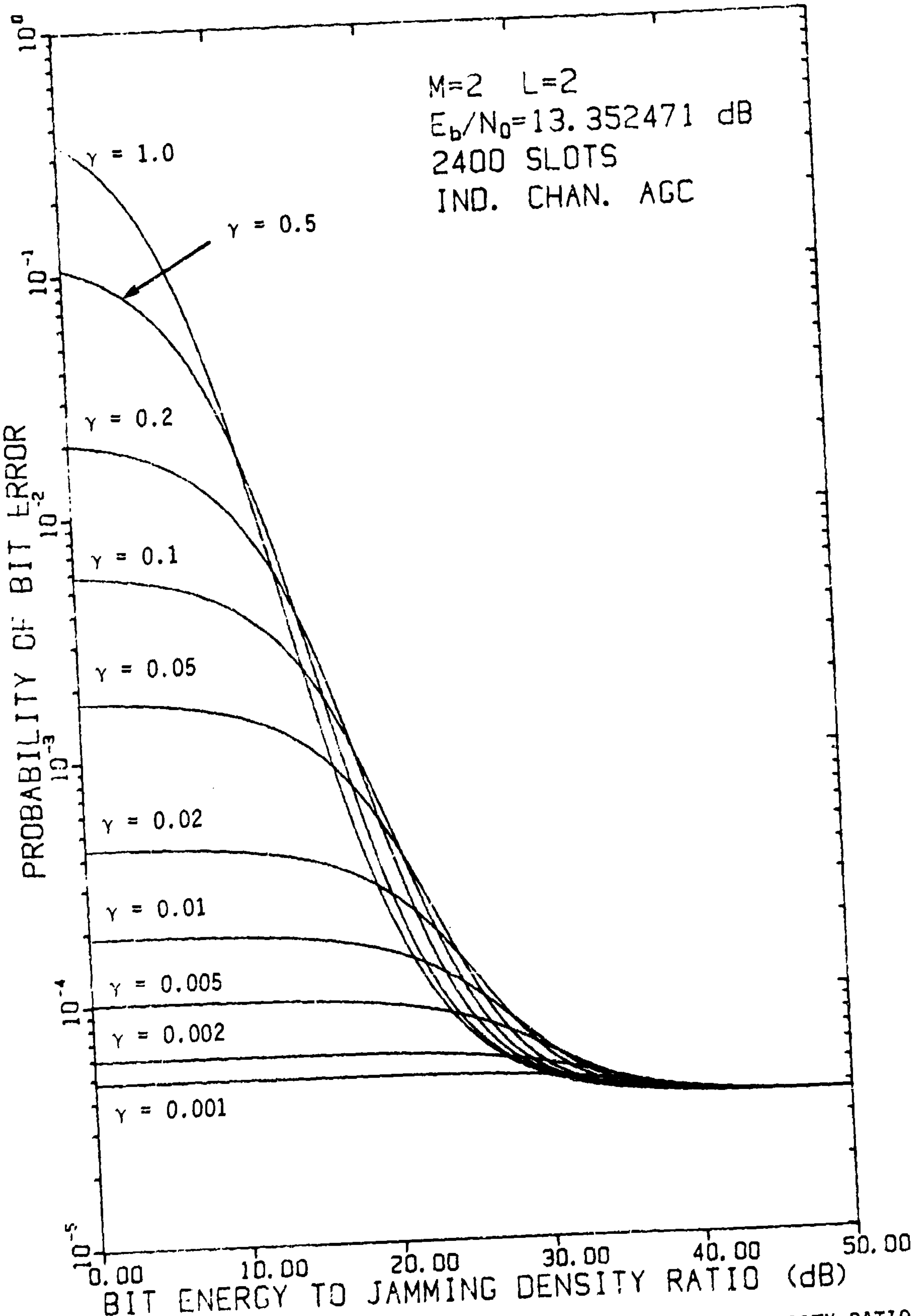


FIGURE 4.4-2 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 2 AND L = 2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

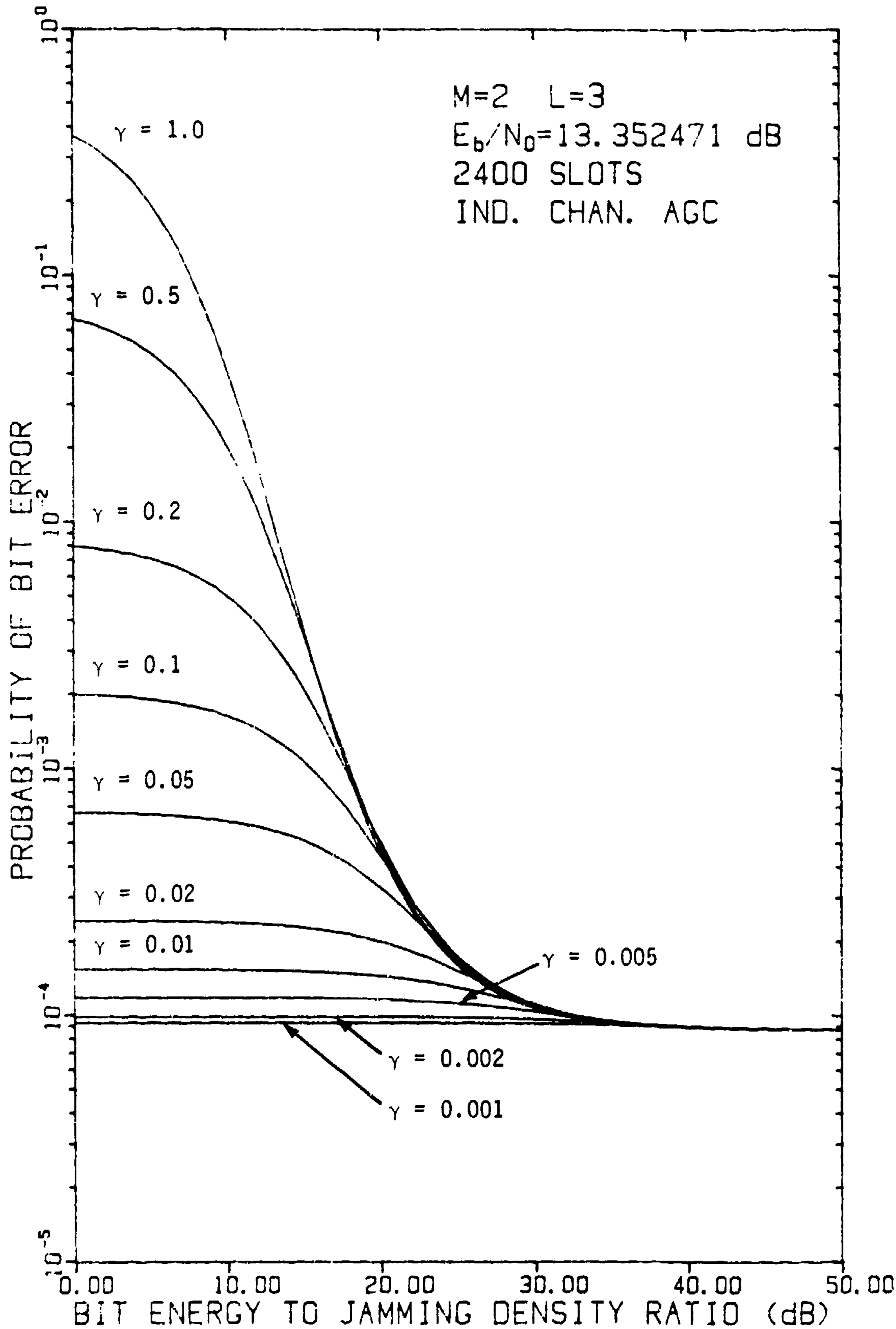


FIGURE 4.4-3 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 2 AND L = 3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

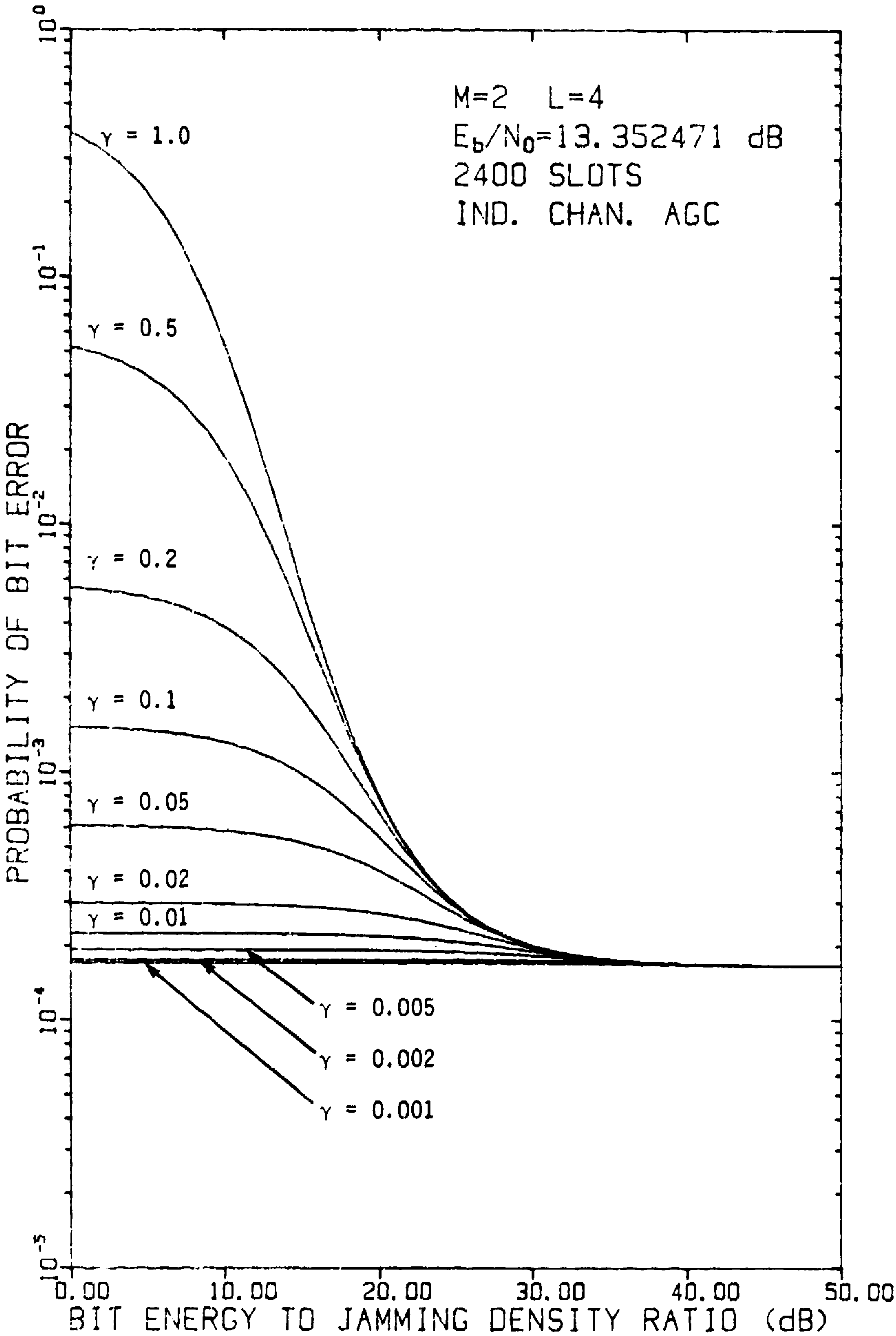


FIGURE 4.4-4 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 2 AND L = 4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

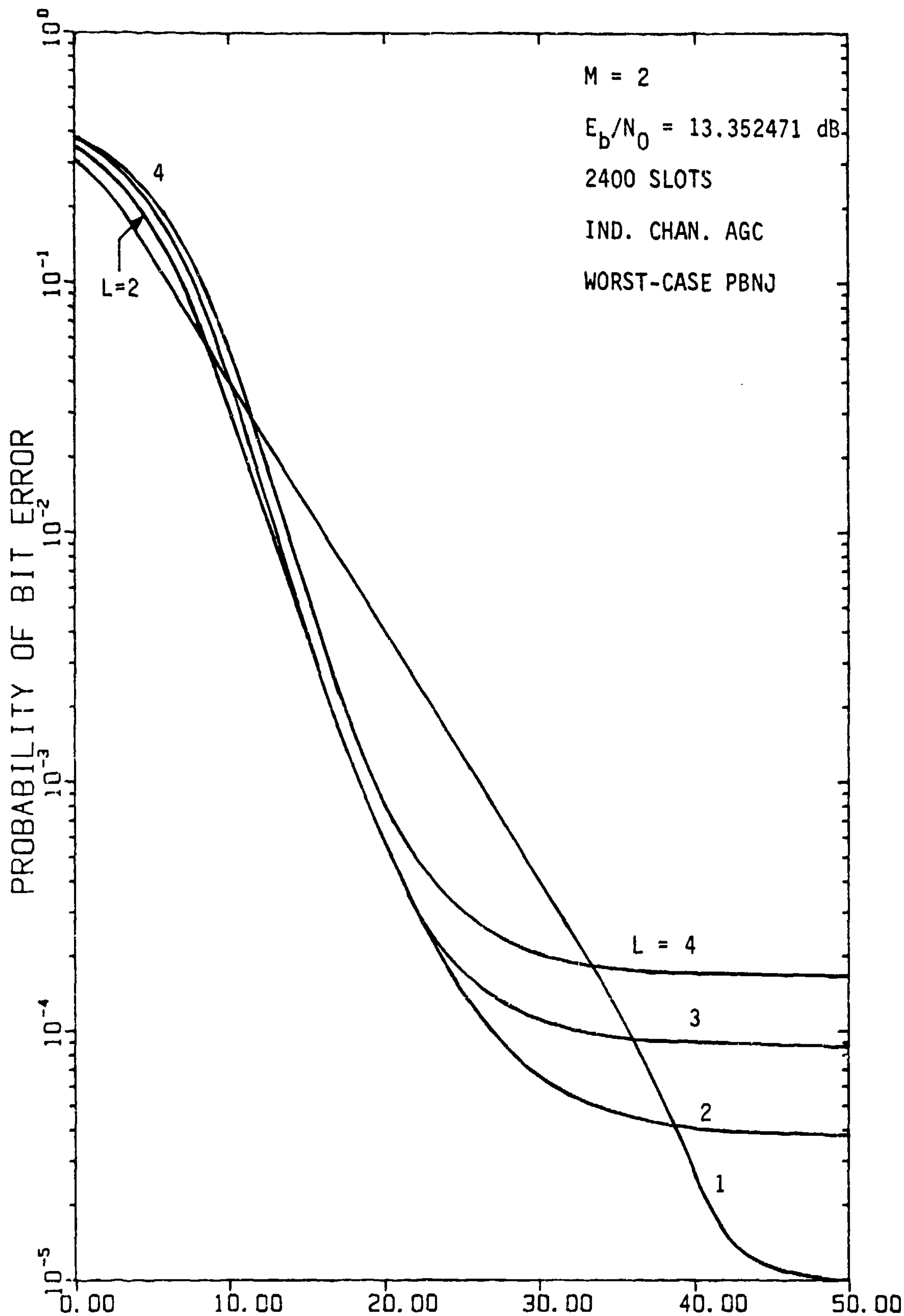


FIGURE 4.4-5 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER AND M = 2 WITH NUMBER OF HOPS/ SYMBOL L AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

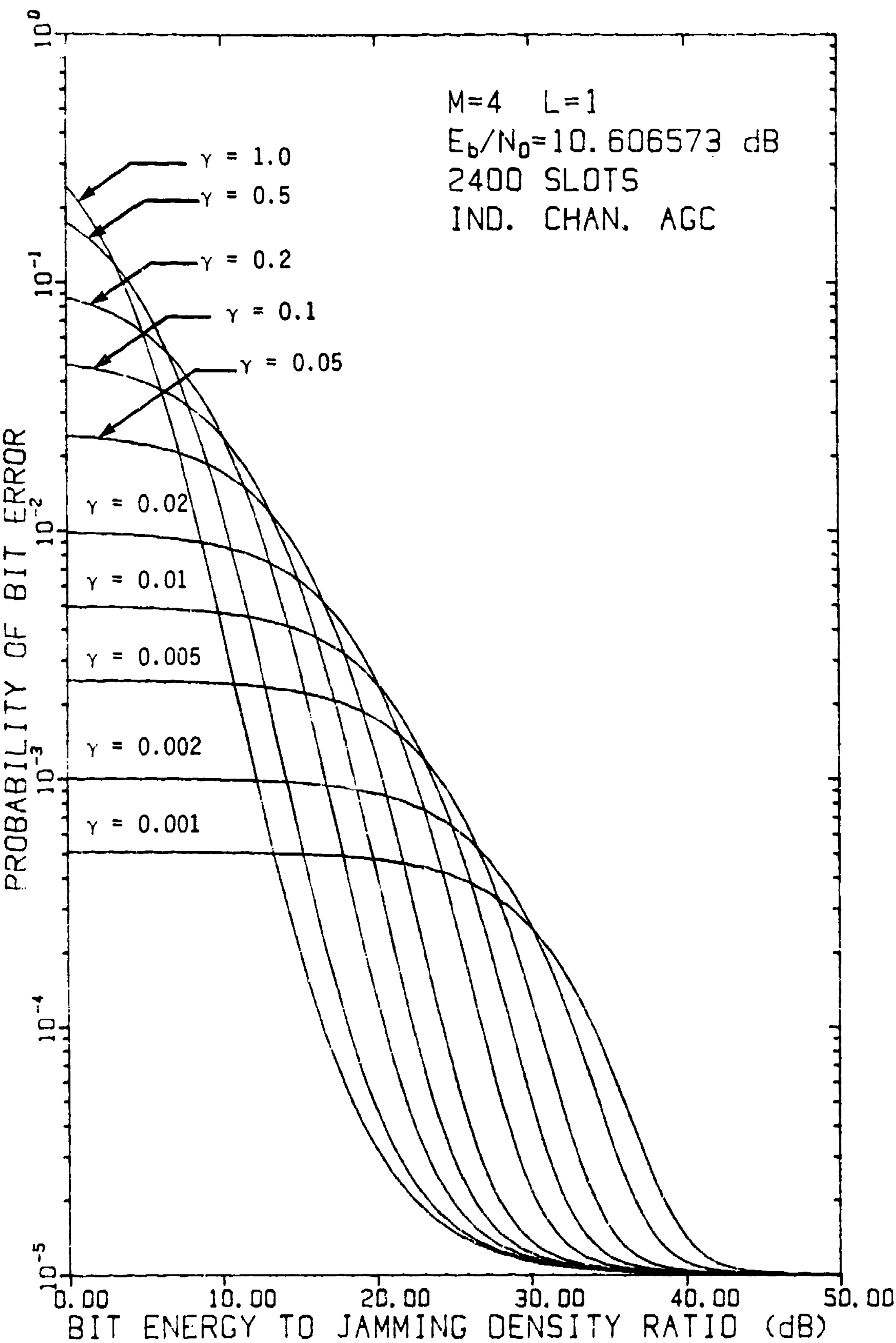


FIGURE 4.4-6 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 4 AND L = 2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

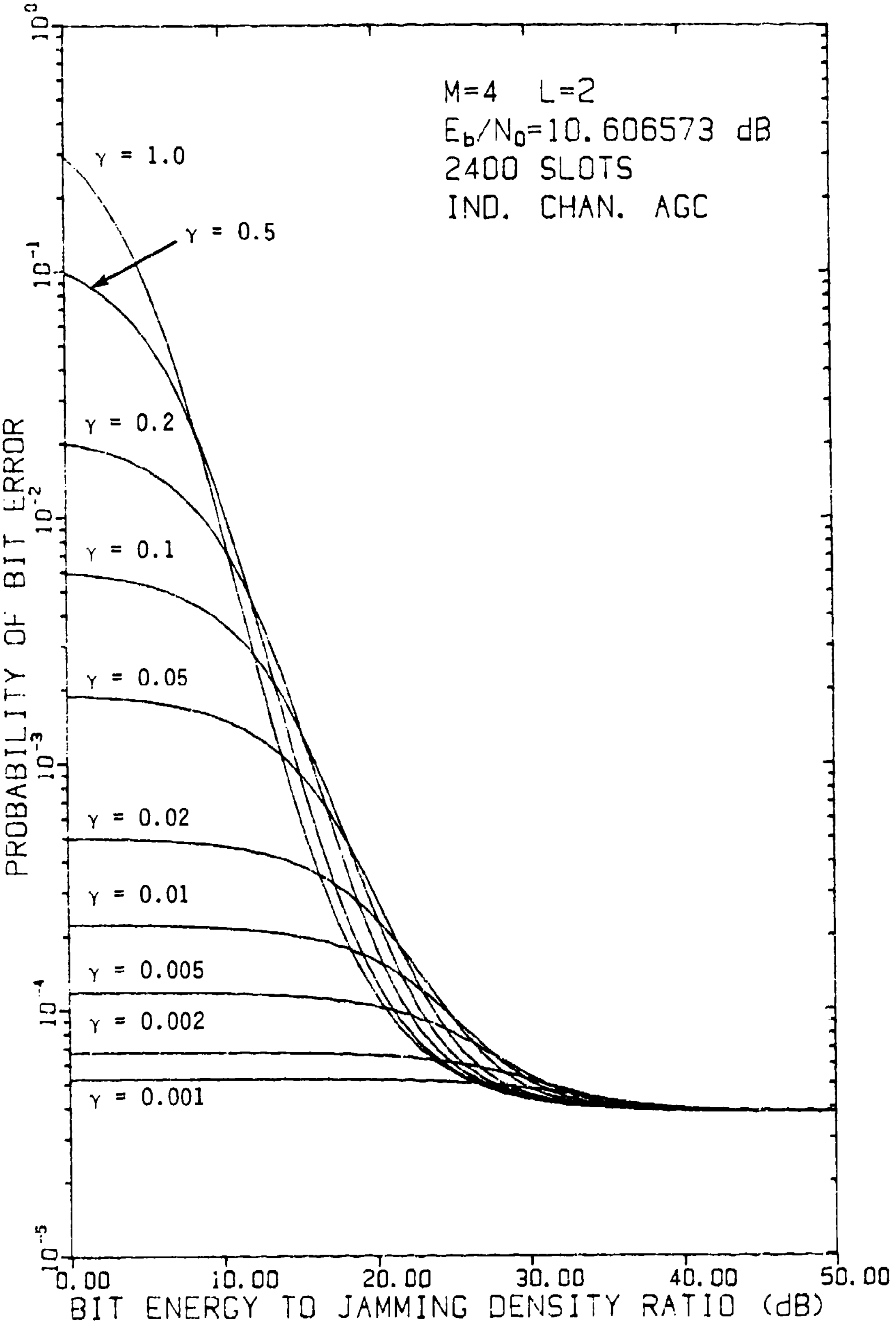


FIGURE 4.4-7 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 4 AND L = 2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

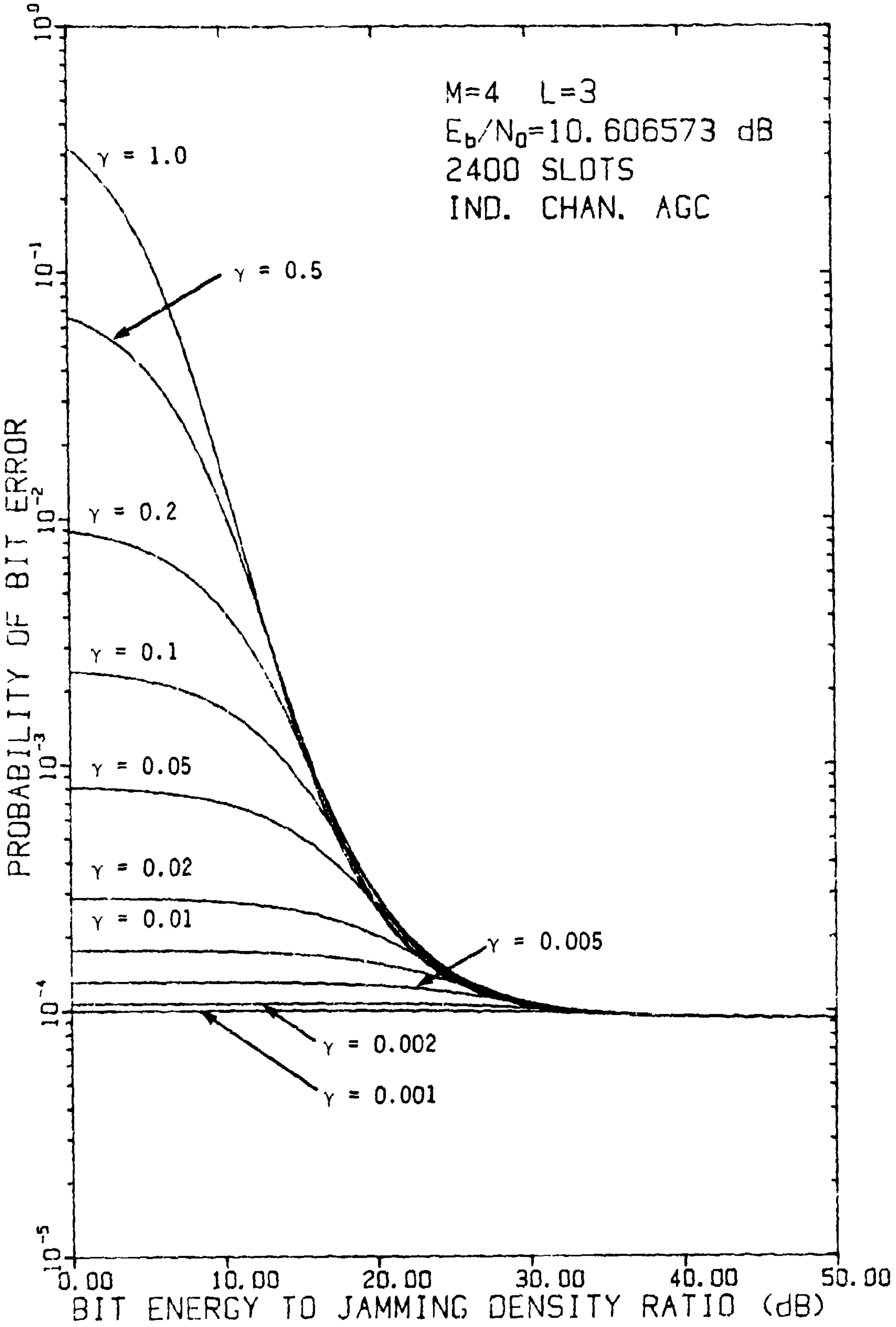


FIGURE 4.4-8 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 4 AND L = 3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

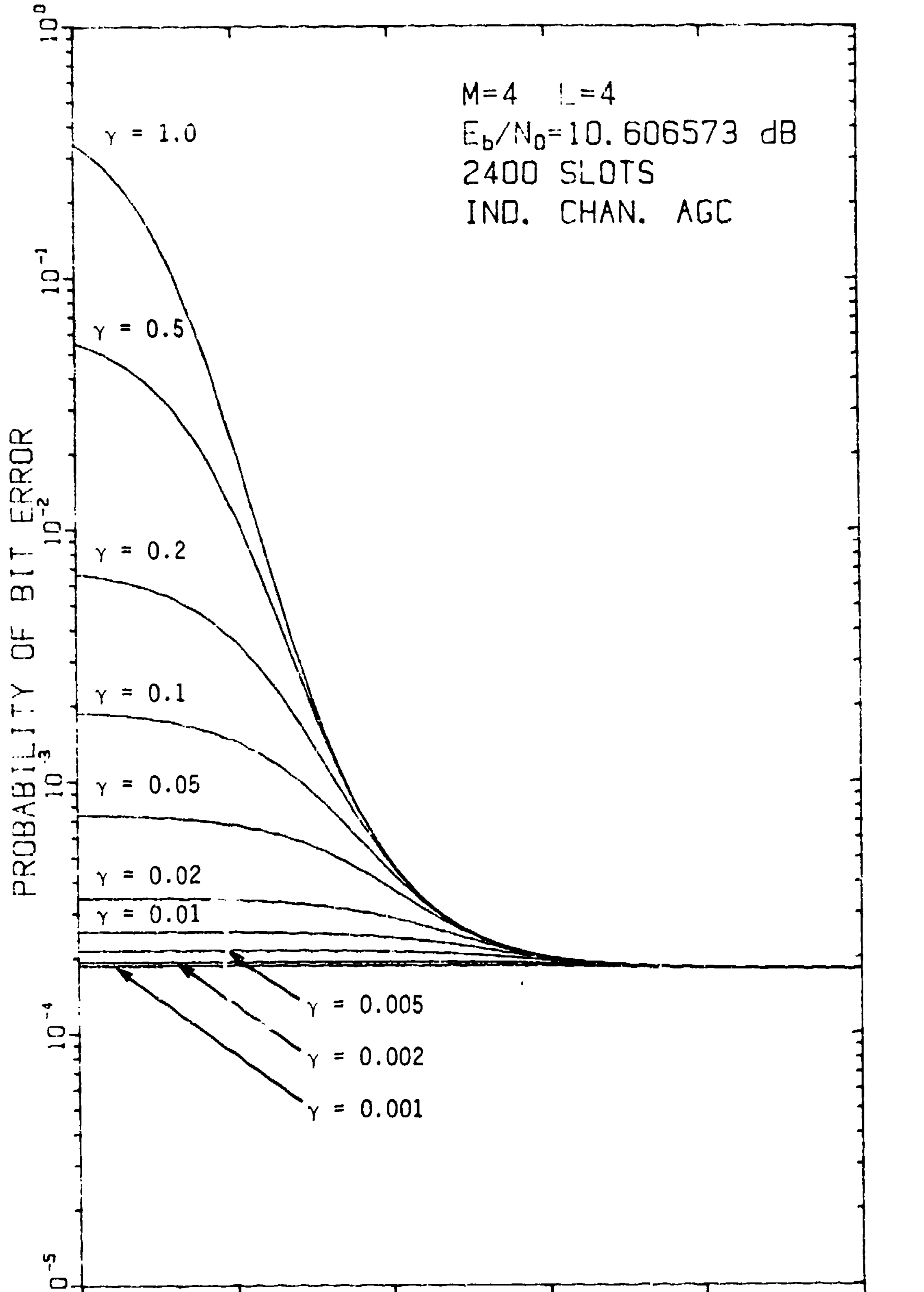


FIGURE 4.4-9 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 4 AND L = 4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

the correct selection of γ stands out clearly. Figure 4.4-10 compares the performance of 4-ary RMFSK/FH in worst-case partial-band noise jamming as L varies. We see that for E_b/N_J in the range of about 7 to 36 dB, L=2 or 3 is optimum; elsewhere, L=1 is optimum.

Figures 4.4-11 through 4.4-14 show the curves for M=8 with γ as a parameter and L=1,2,3, and 4, respectively. Figure 4.4-15 shows performance for M=8 with L as a parameter in worst-case partial-band noise jamming. Again, from about E_b/N_J=5 dB to 35 dB the optimum L is 2 or 3, but elsewhere L=1 is optimum.

Two important conclusions can be drawn from these curves:

- The correct choice of fraction γ is critical for the jammer;
- The communicator can obtain only a small benefit by using multiple hops/symbol, and then only over a limited range of jamming conditions.

4.4.2 Numerical Results for Any-Channel-Jammed AGC (ACJ-AGC) Receiver

The numerical computations for the performance of the ACJ-AGC receiver required the use of two alternative forms. We used (4.3-12) for the computations, in conjunction with the computational techniques discussed in Section 3.3. The switch-over criteria for choosing between series and numerical integration were determined empirically. A listing of the computer program for numerical computations is given in Appendix F and a listing of the program which produced the plots is given in Appendix G. This latter program is typical of the plotting programs for all receivers; for sake of brevity only this one version of the plot program is included in our report.

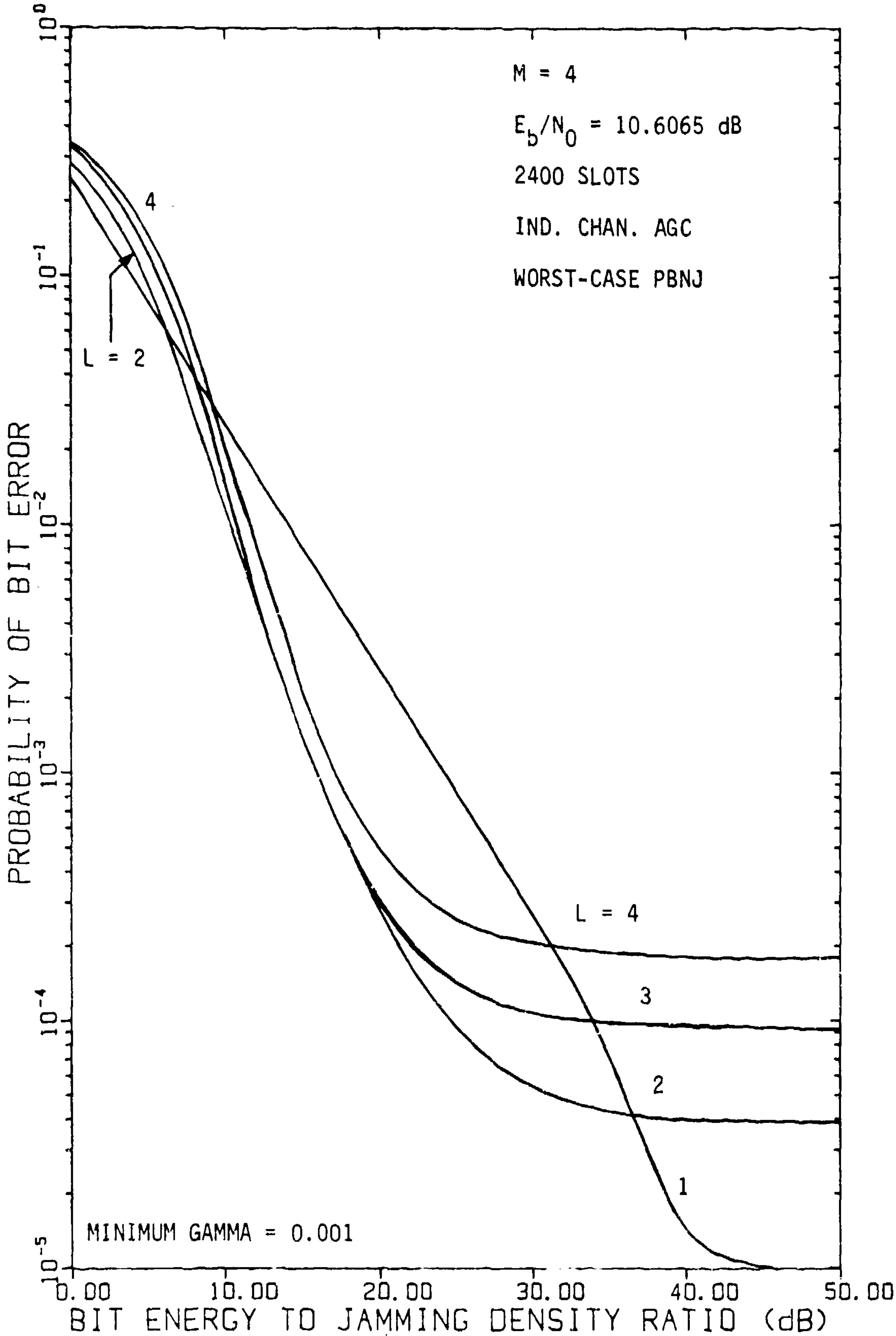


FIGURE 4.4-10 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER AND M = 4 WITH NUMBER OF HOPS/SYMBOL L AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

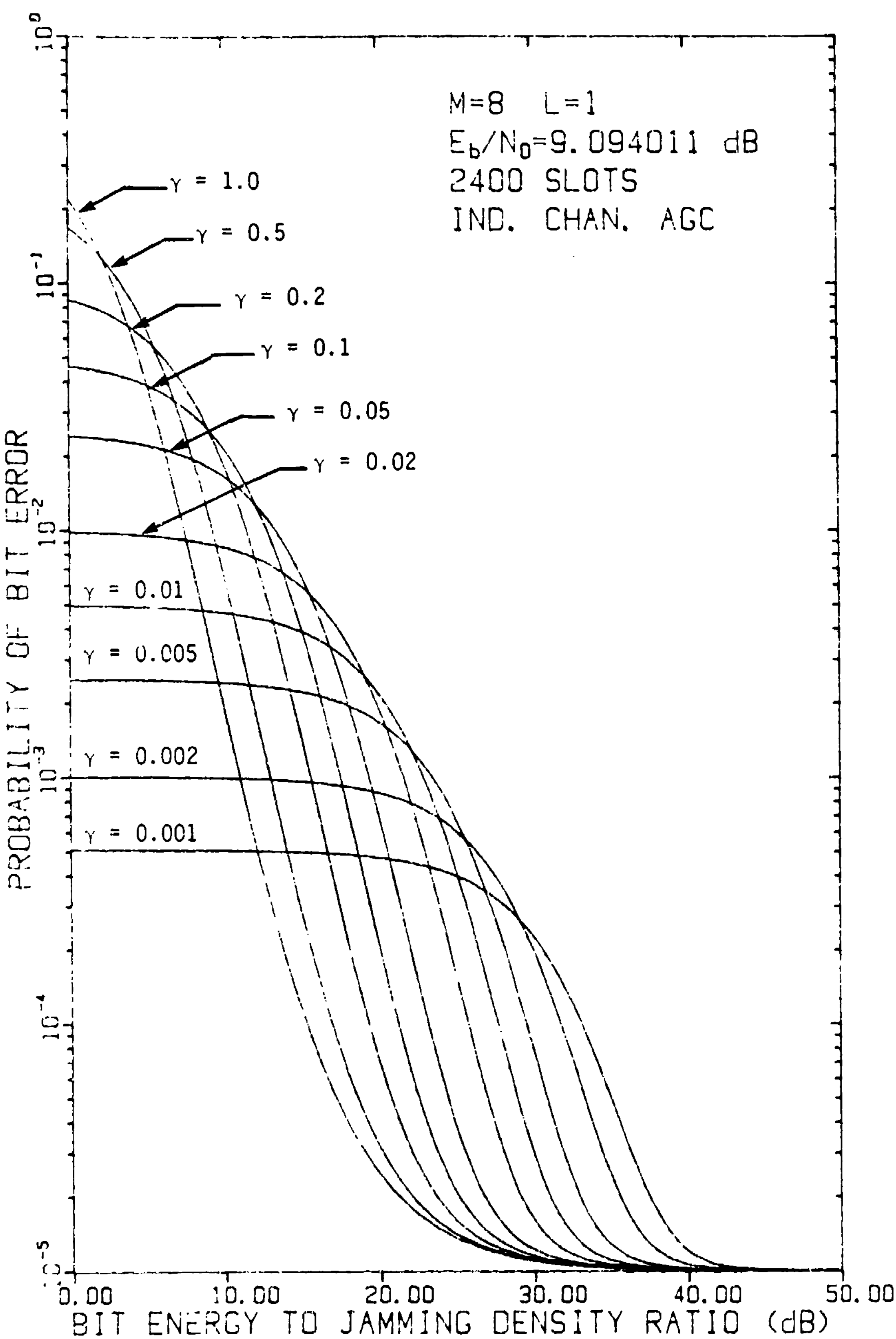


FIGURE 4.4-11 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 8 AND L = 1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

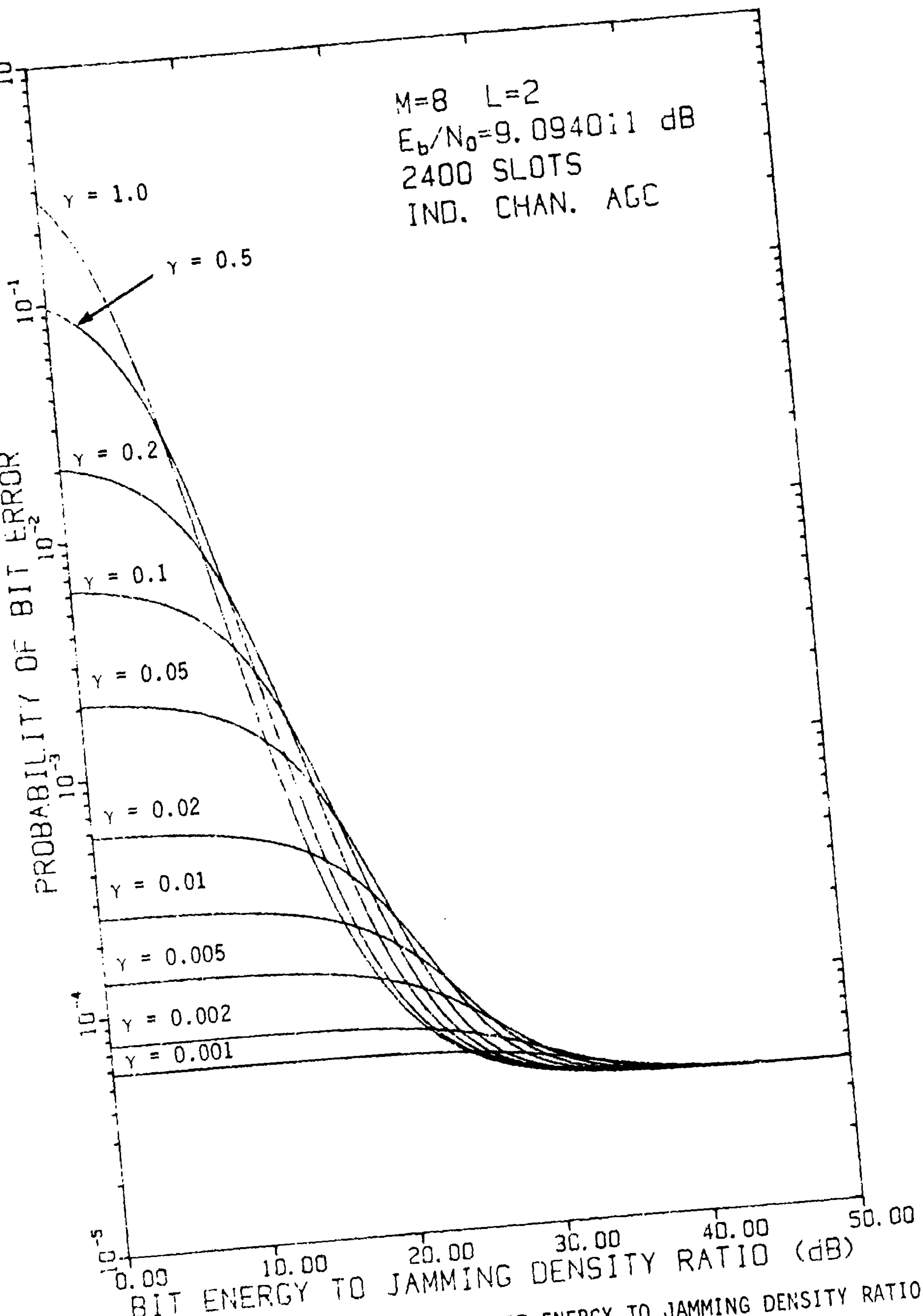


FIGURE 4.4-12 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 8 AND L = 2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

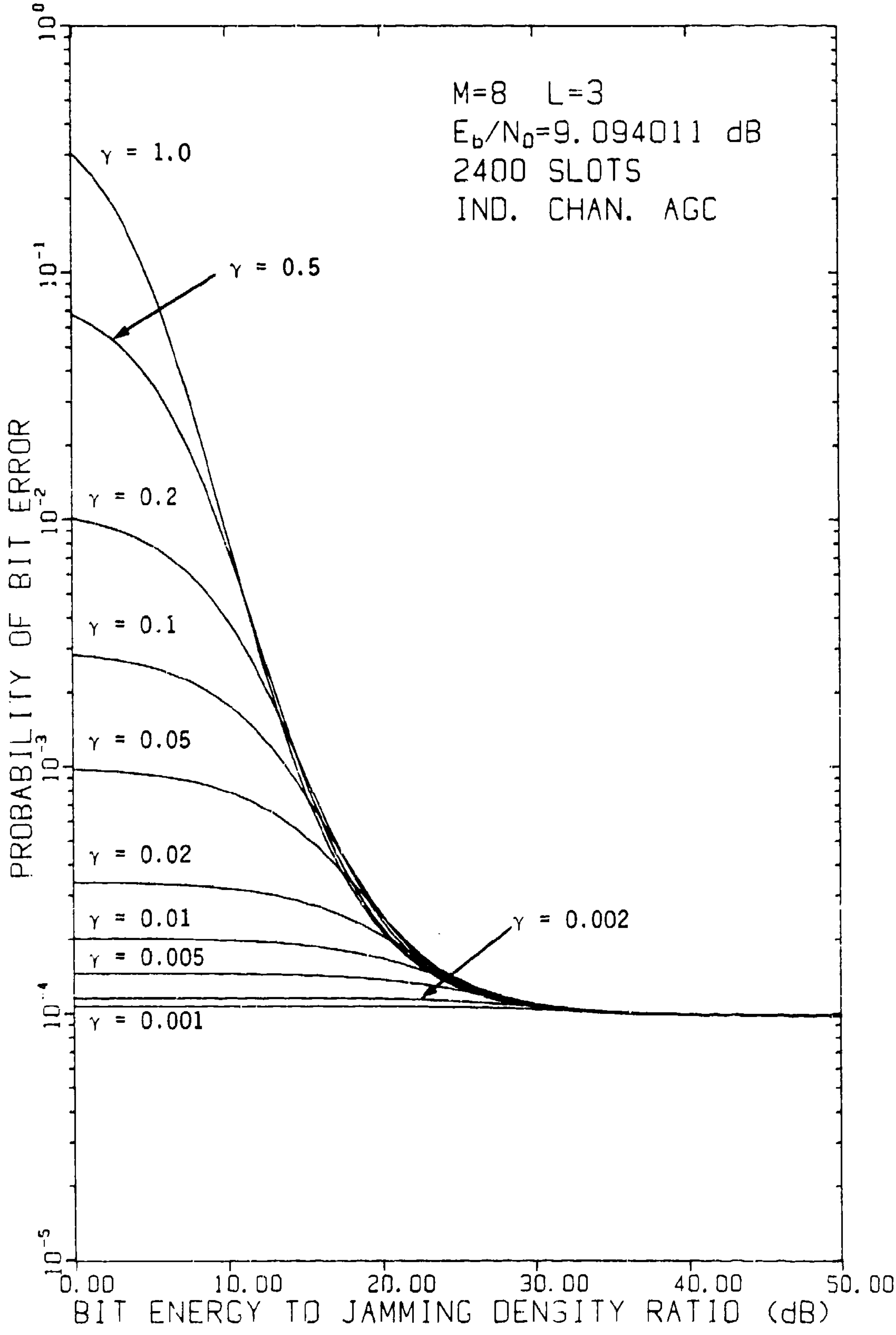


FIGURE 4.4-13 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 8 AND L = 3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

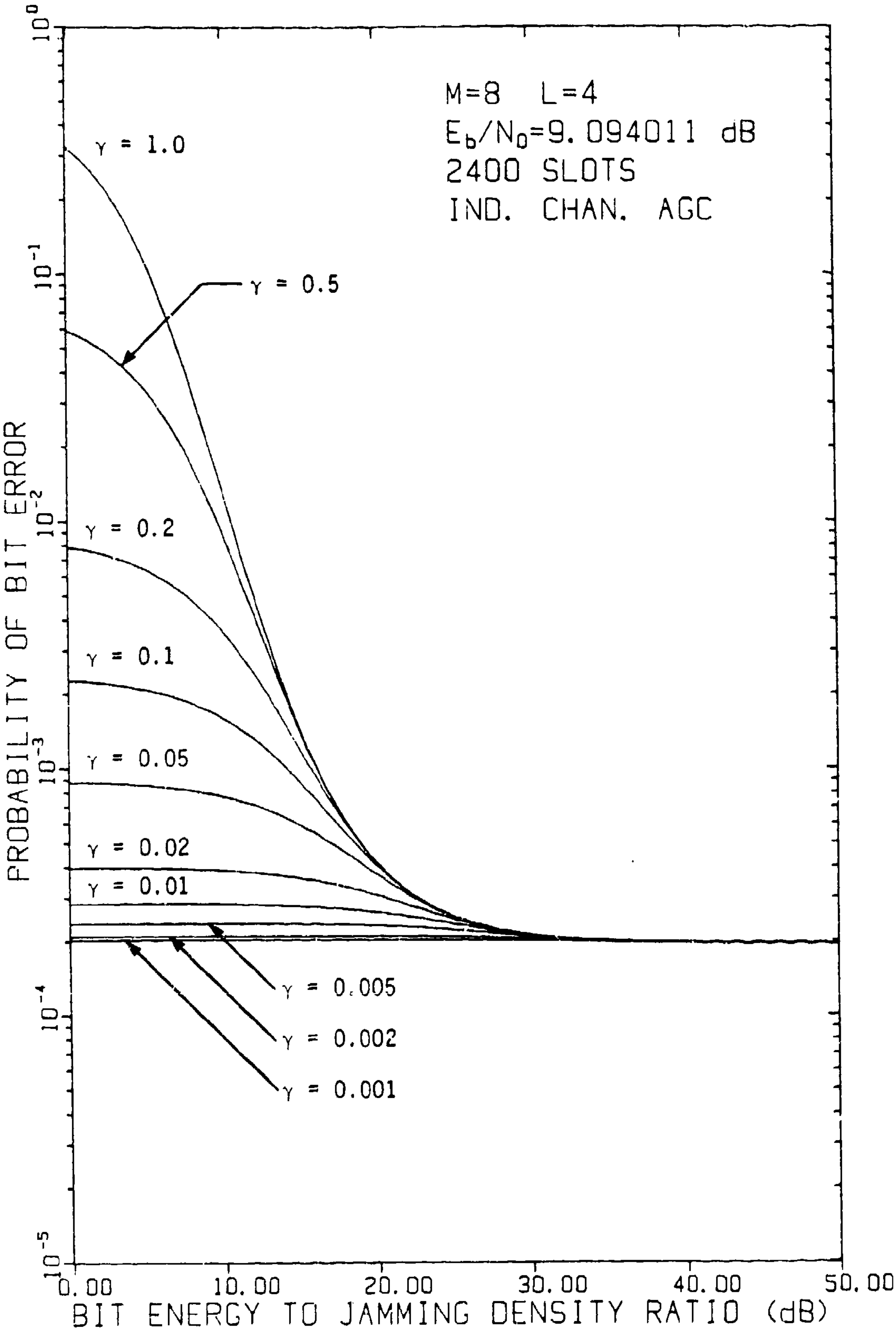


FIGURE 4.4-14 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER WITH M = 8 AND L = 4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L = 1) WITH JAMMING FRACTION γ AS A PARAMETER

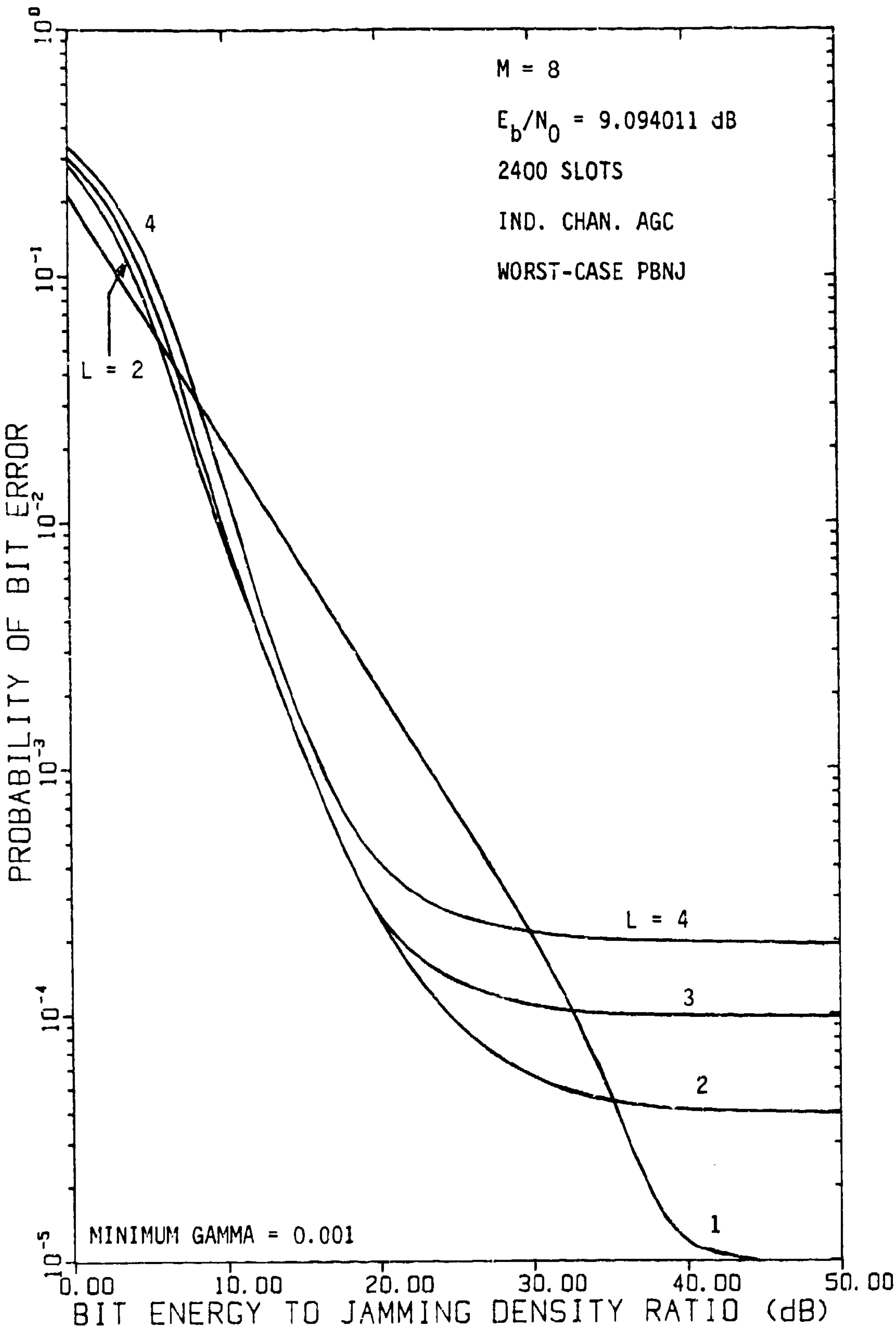


FIGURE 4.4-15 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER AND M=8 WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

Figures 4.4-16 through 4.4-19 show the performance of the ACJ-AGC receiver with M=2 for L=1,2,3, and 4 hops/bit, respectively. Figure 4.4-20 summarizes the performance in worst-case partial-band noise jamming with L as a parameter. Again, we find a limited range, roughly 10 dB $\leq E_b/N_J \leq$ 35 dB, where the optimum diversity is L=2 or 3 hops per bit; elsewhere L=1 is optimum.

Figures 4.4-21 through 4.4-24 show the performance of the ACJ-AGC receiver with M=4 for L=1,2,3, and 4 hops/symbol, respectively. These curves show the same general behavior as those for the other receivers. Figure 4.4-25 shows the performance in worst-case partial-band noise jamming with L as a parameter. Again, we find a limited range where L=2 or 3 is optimum, but elsewhere L=1 is optimum.

Finally, Figures 4.4-26 and 4.4-27 show the performance for M=8 and L=1 and 2, respectively. Because of the large computer time required, L>2 was not considered for M=8. Figure 4.4-28 summarizes performance in worst-case partial-band noise jamming. Again, there is a region where diversity (L=2) offers some advantage.

In summary, a small amount of diversity (L=2 or 3) is of meaningful benefit to RMFSK/FH over a limited range of signal-to-jamming ratios. However, outside this range no diversity (L=1) gives better performance. In some cases, e.g. Figure 4.4-20 for M=2, the penalty for using L=3 in the absence of jamming is nearly the same as the benefit of using L=3 when E_b/N_J=25 dB.

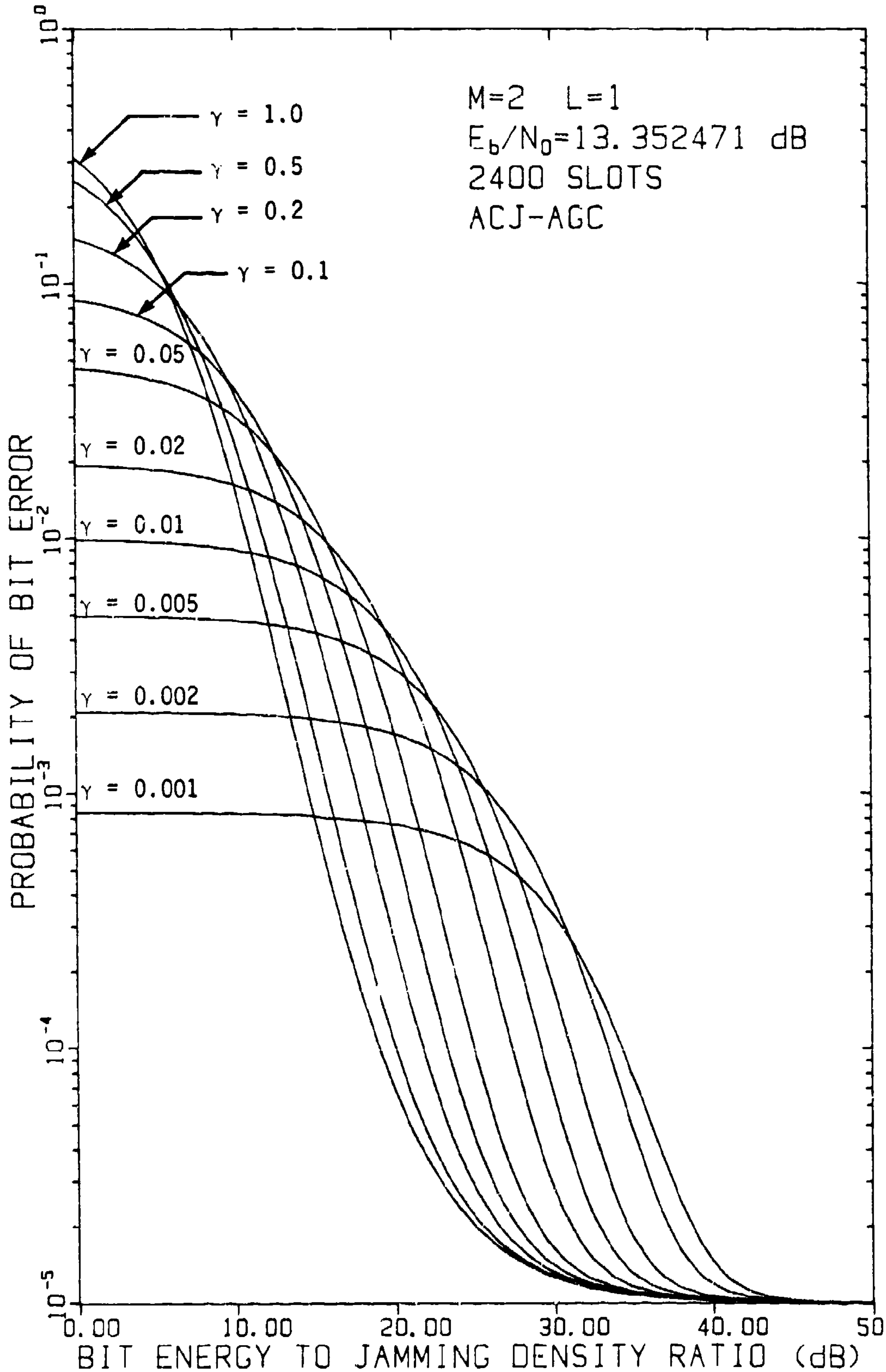


FIGURE 4.4-16 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=2 AND L=1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

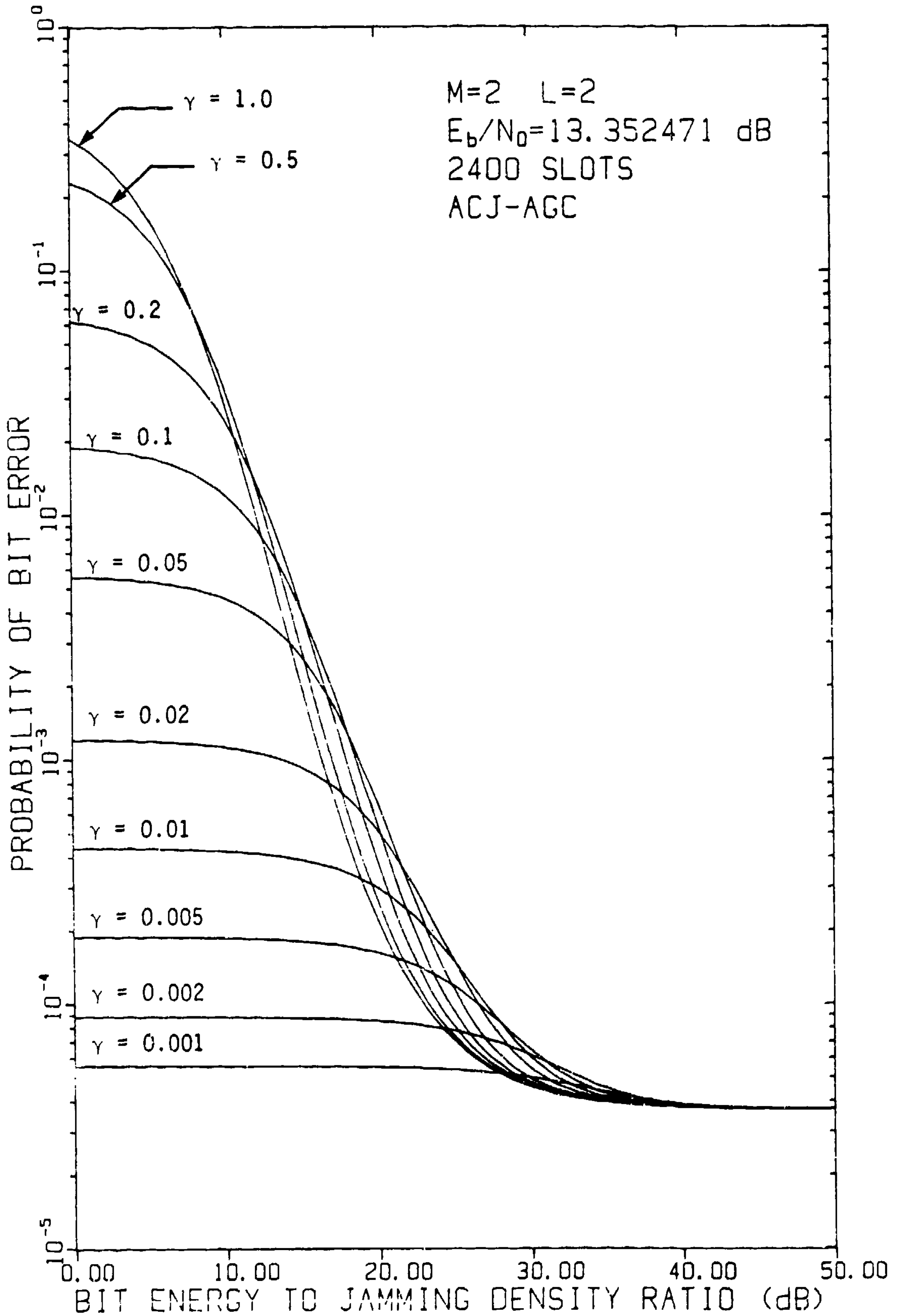


FIGURE 4.4-17 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=2 AND L=2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

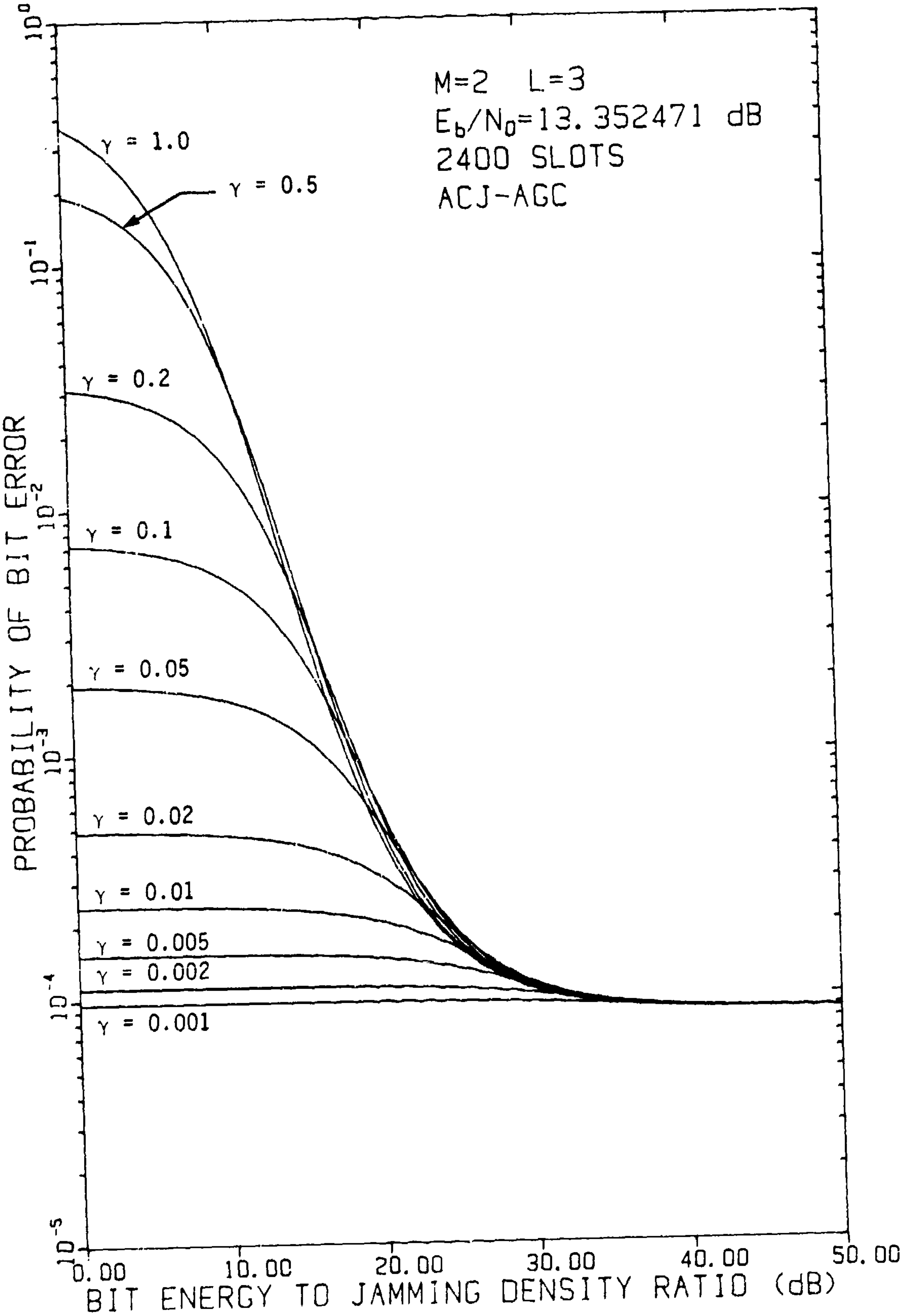


FIGURE 4.4-18 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL JAMMED AGC RECEIVER WITH M=2 AND L=3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

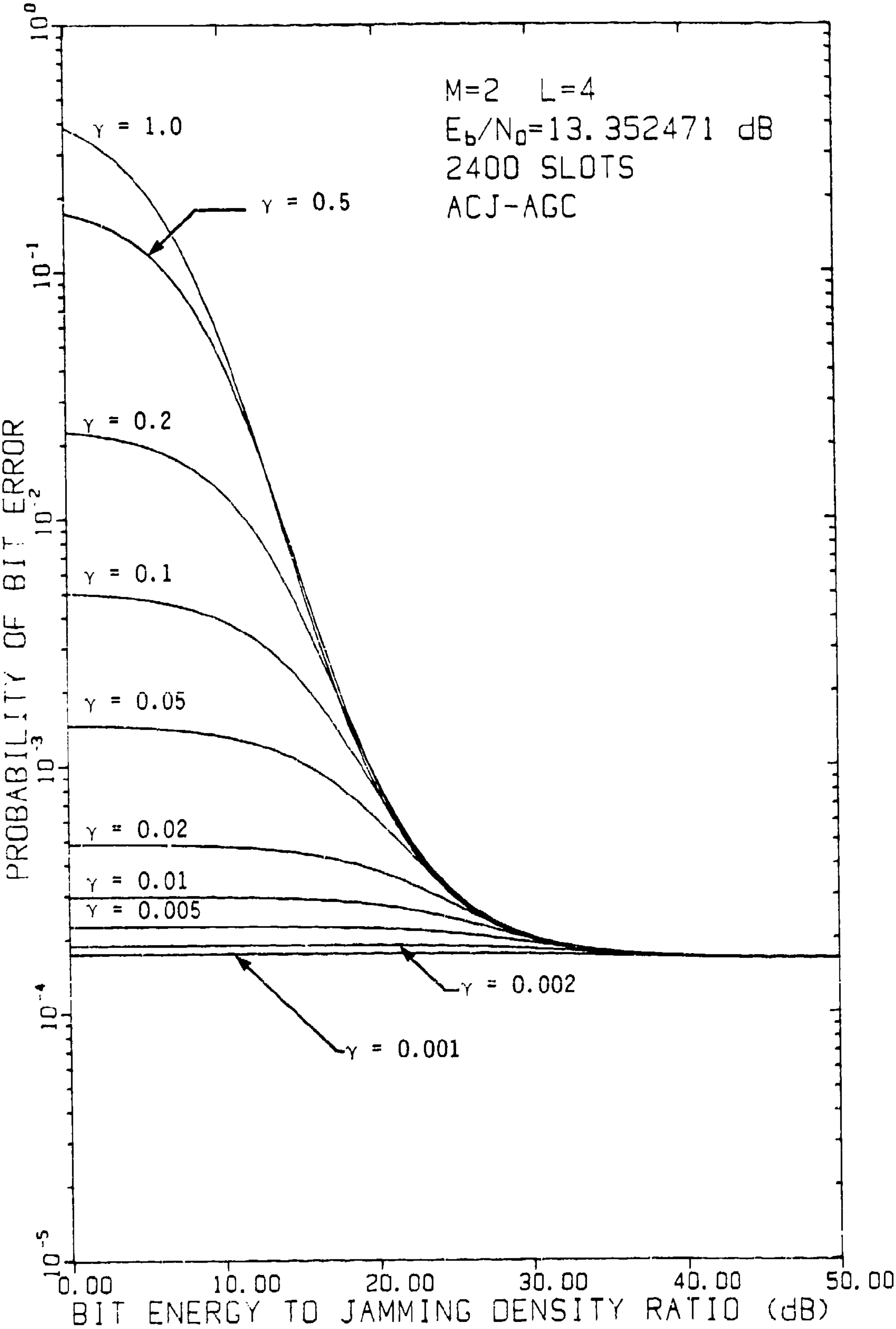


FIGURE 4.4-19 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=2 AND L=4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.352471 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

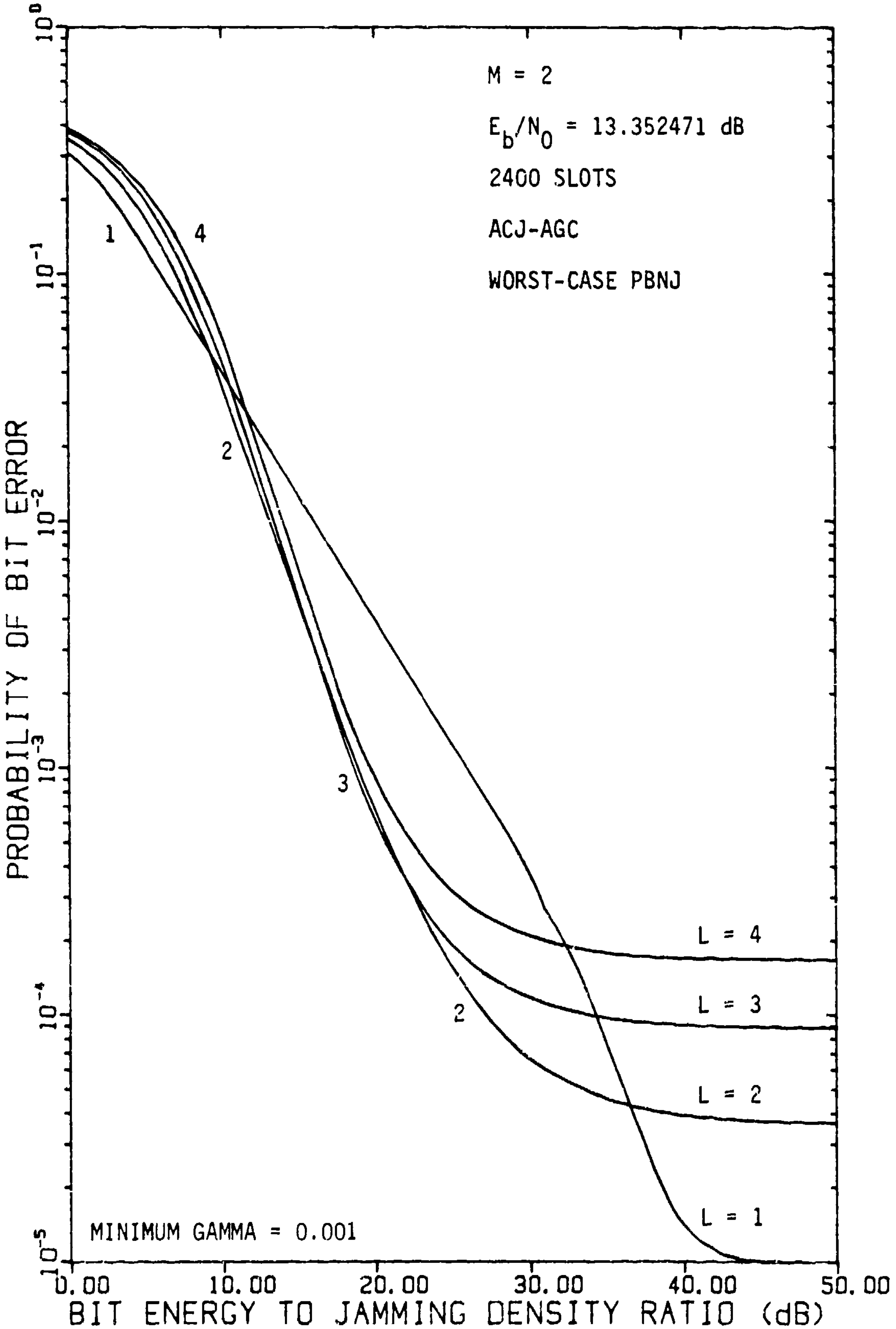


FIGURE 4.4-20 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER AND M=2 WITH NUMBER OF HOPS/BIT AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

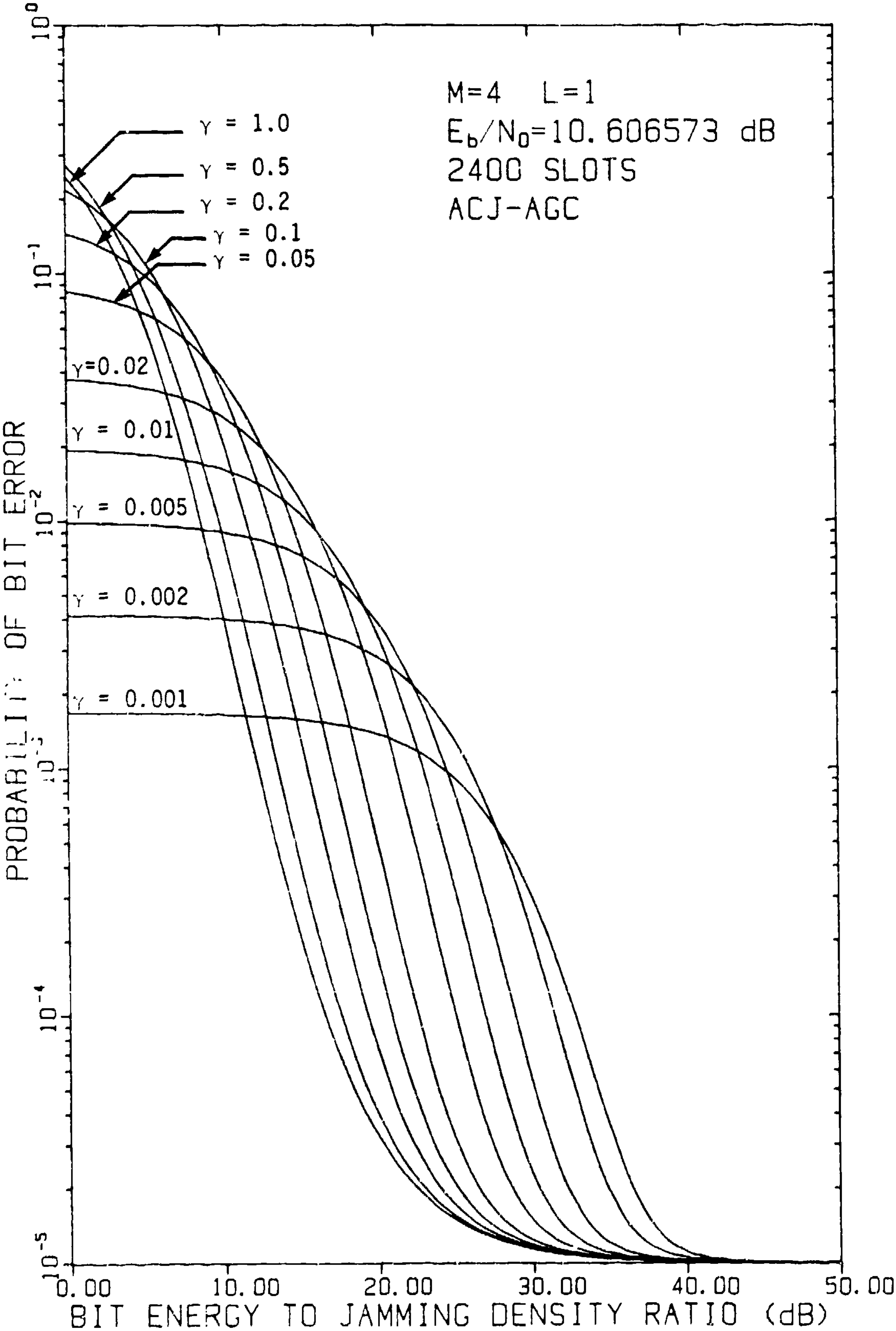


FIGURE 4.4-21 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=4 AND L=1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

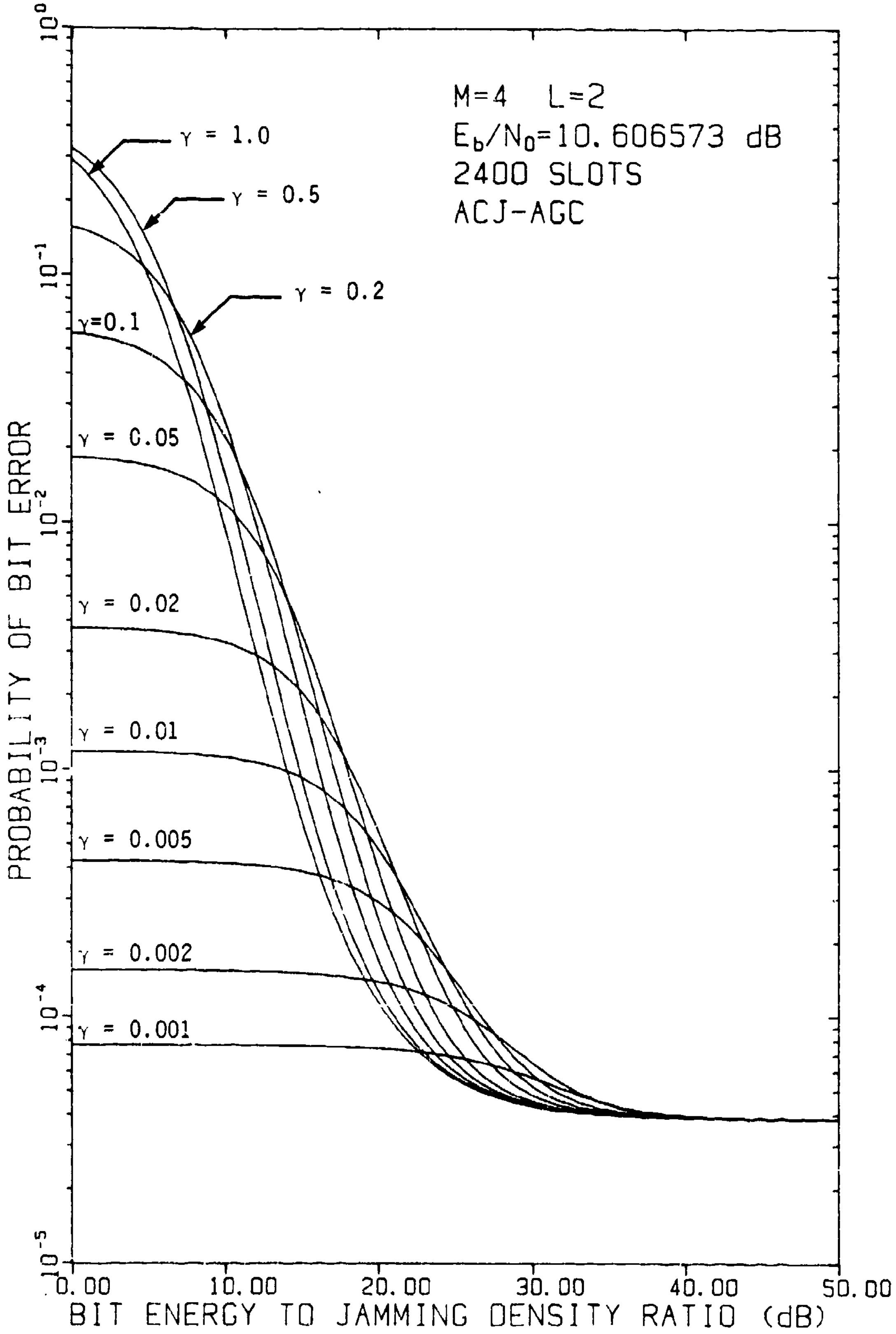


FIGURE 4.4-22 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=4 AND L=2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

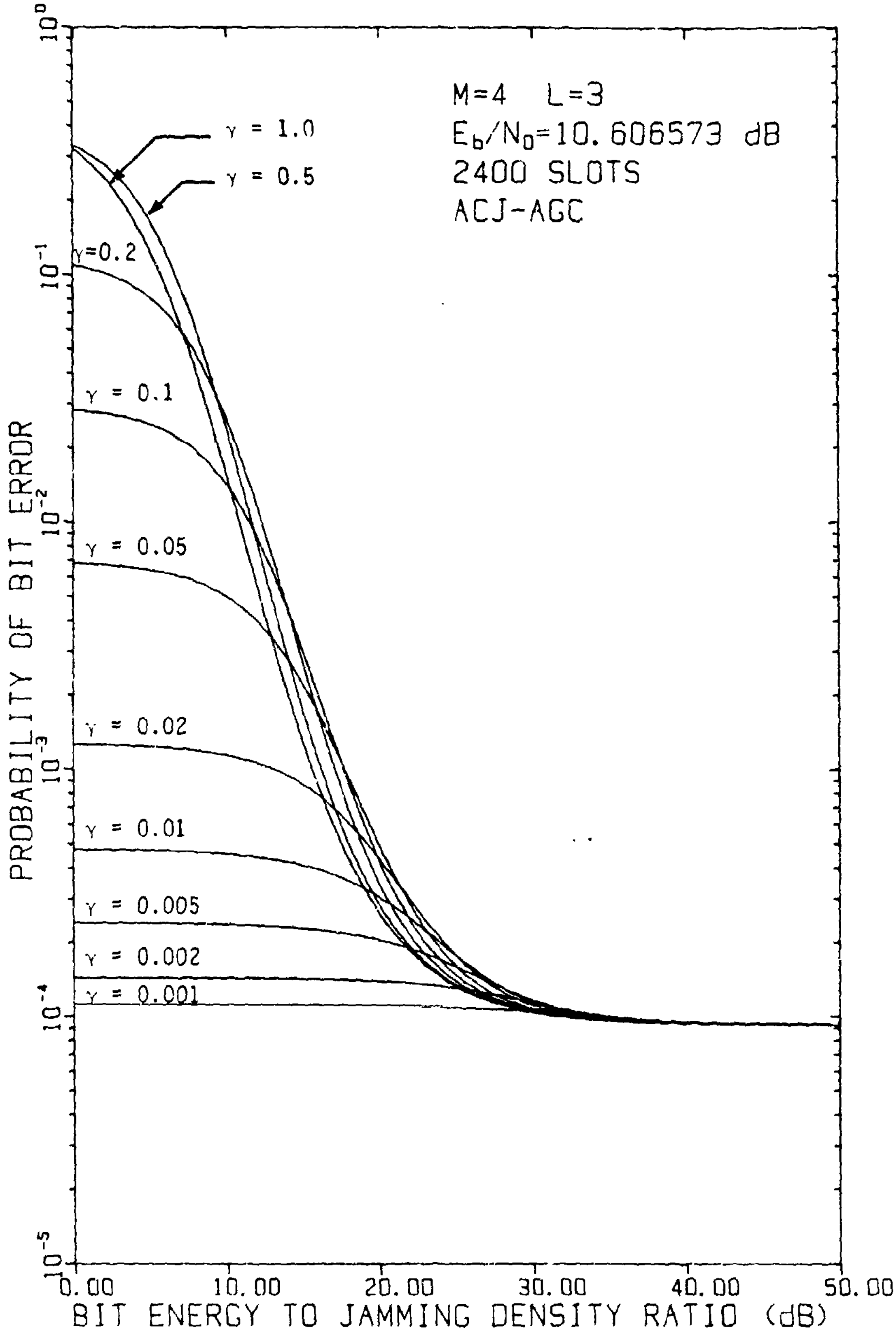


FIGURE 4.4-23 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=4 AND L=3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

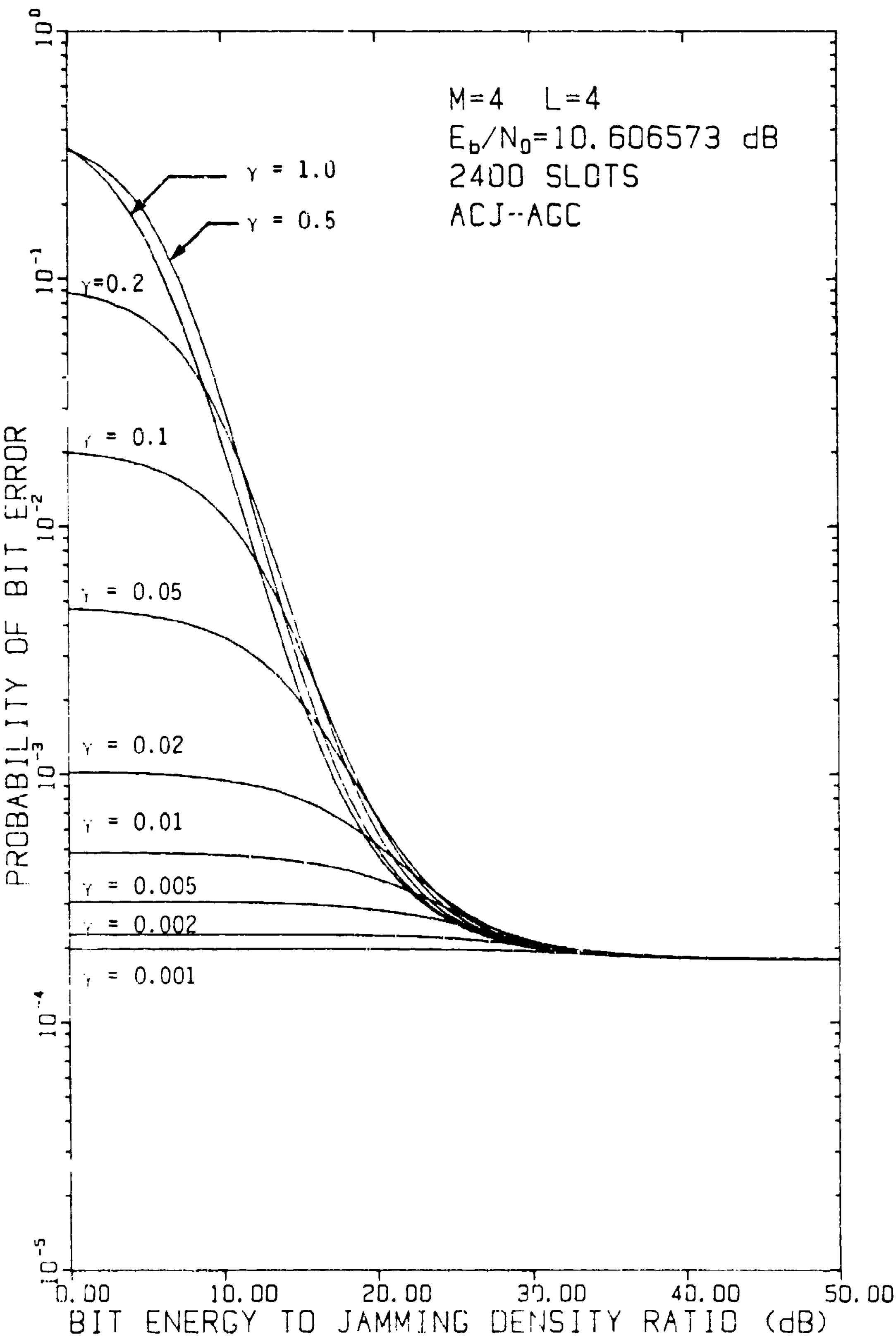


FIGURE 4.4-24 BIT ERROR PROBABILITY VS BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=4 AND L=4 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 10.606573 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

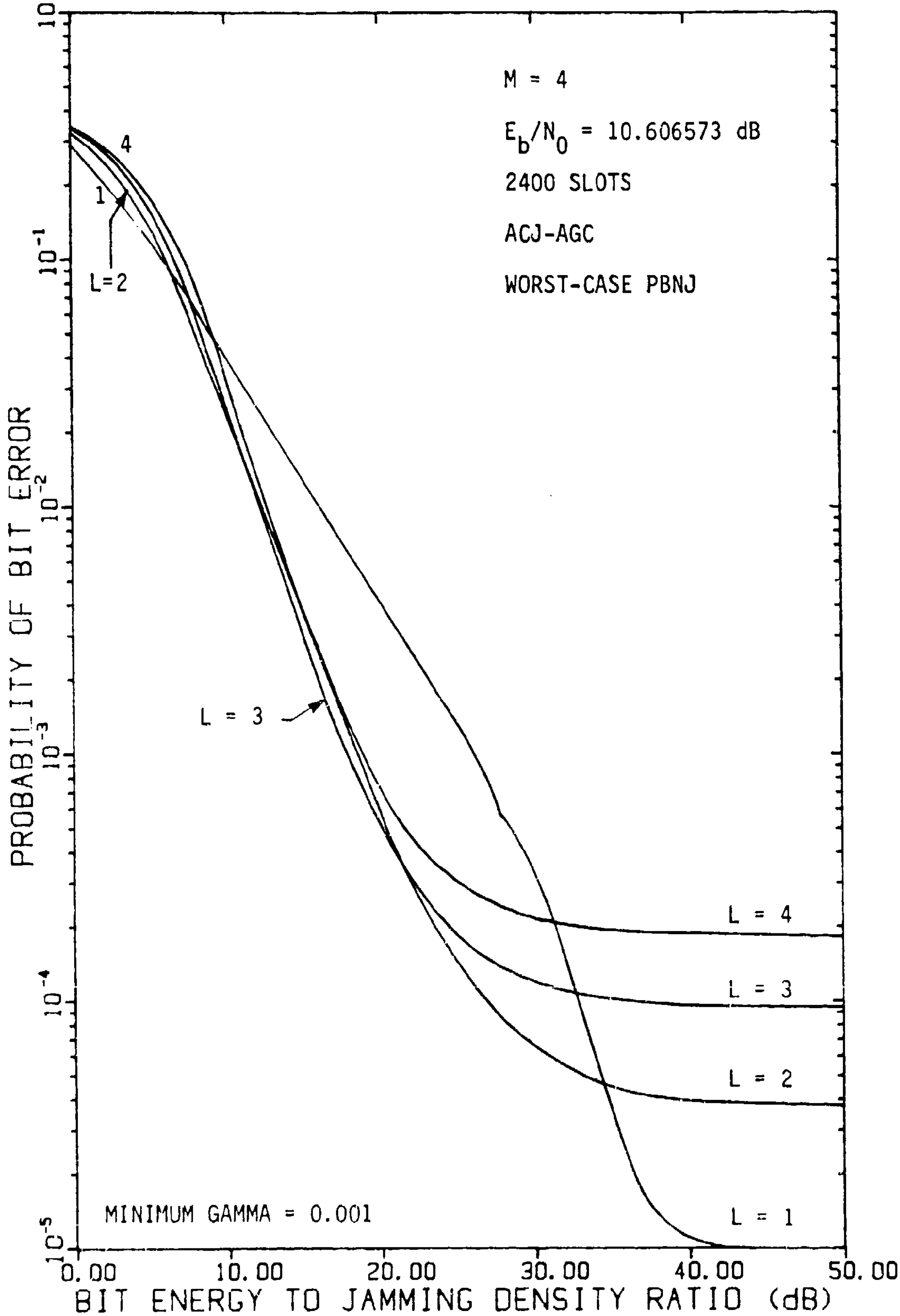


FIGURE 4.4-25 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER AND M=4 WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

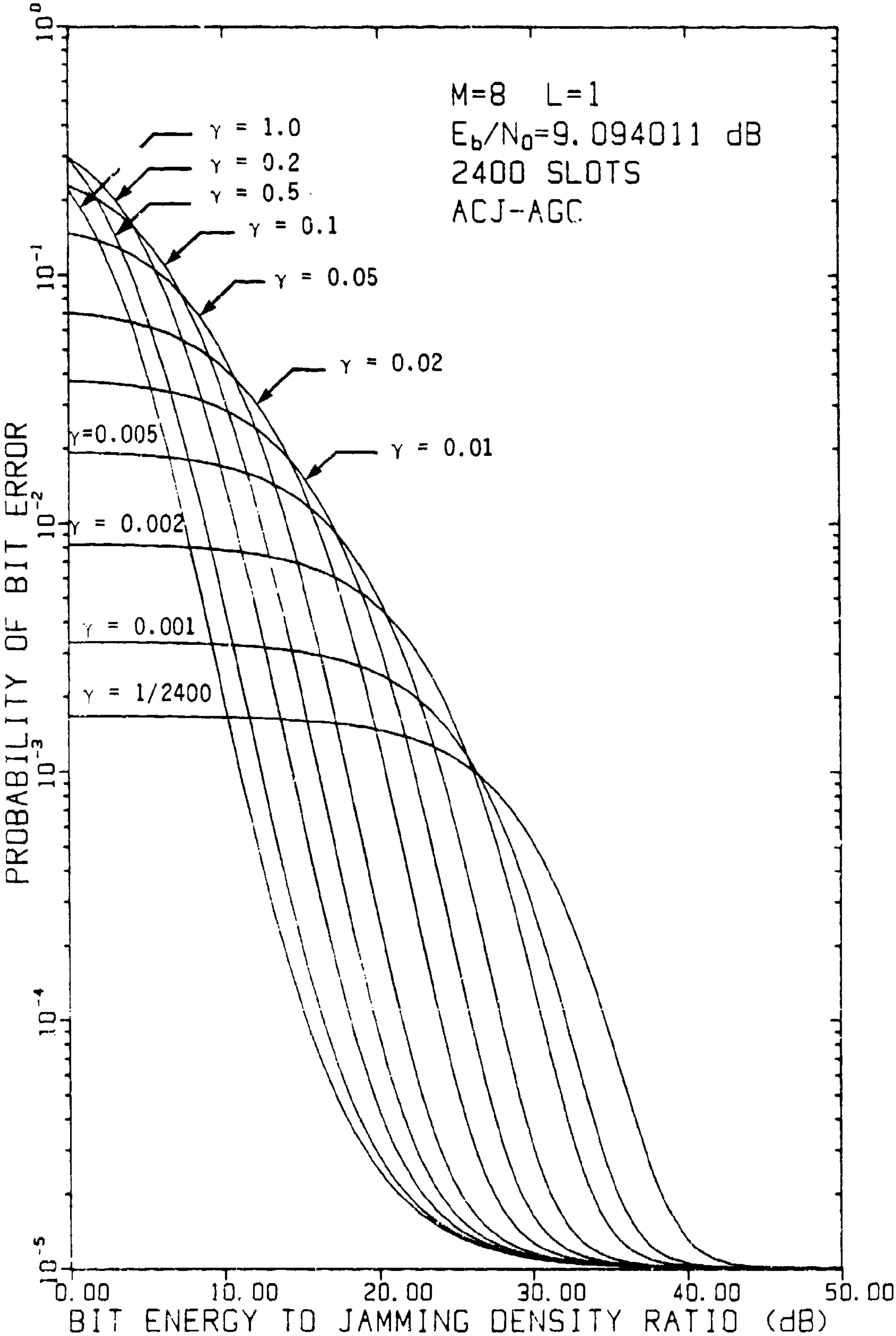


FIGURE 4.4-26 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=8 AND L=1 HOP/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

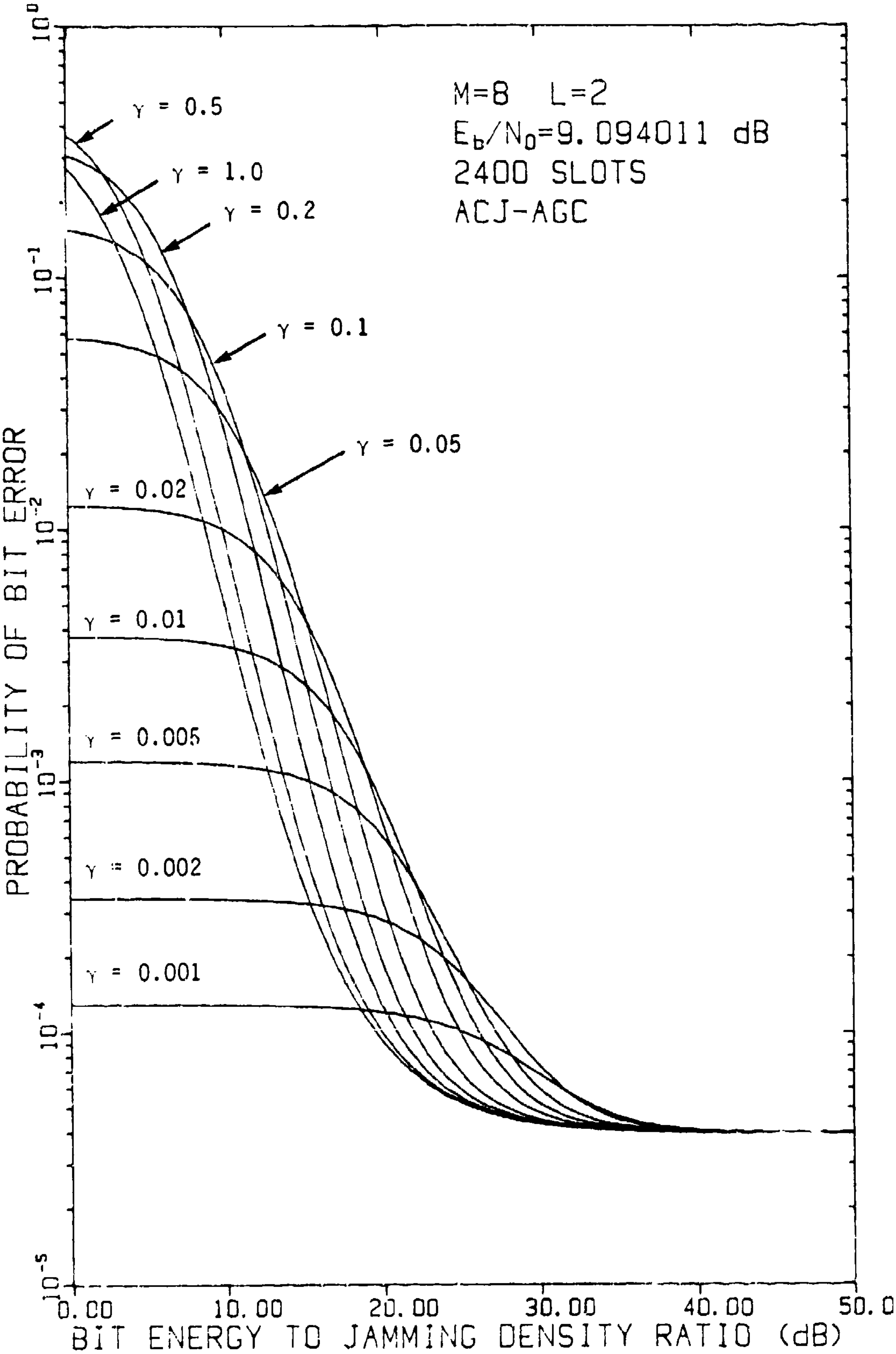


FIGURE 4.4-27 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER WITH M=8 AND L=2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 9.094011 dB (FOR 10^{-5} BER WITHOUT JAMMING WHEN L=1) WITH JAMMING FRACTION γ AS A PARAMETER

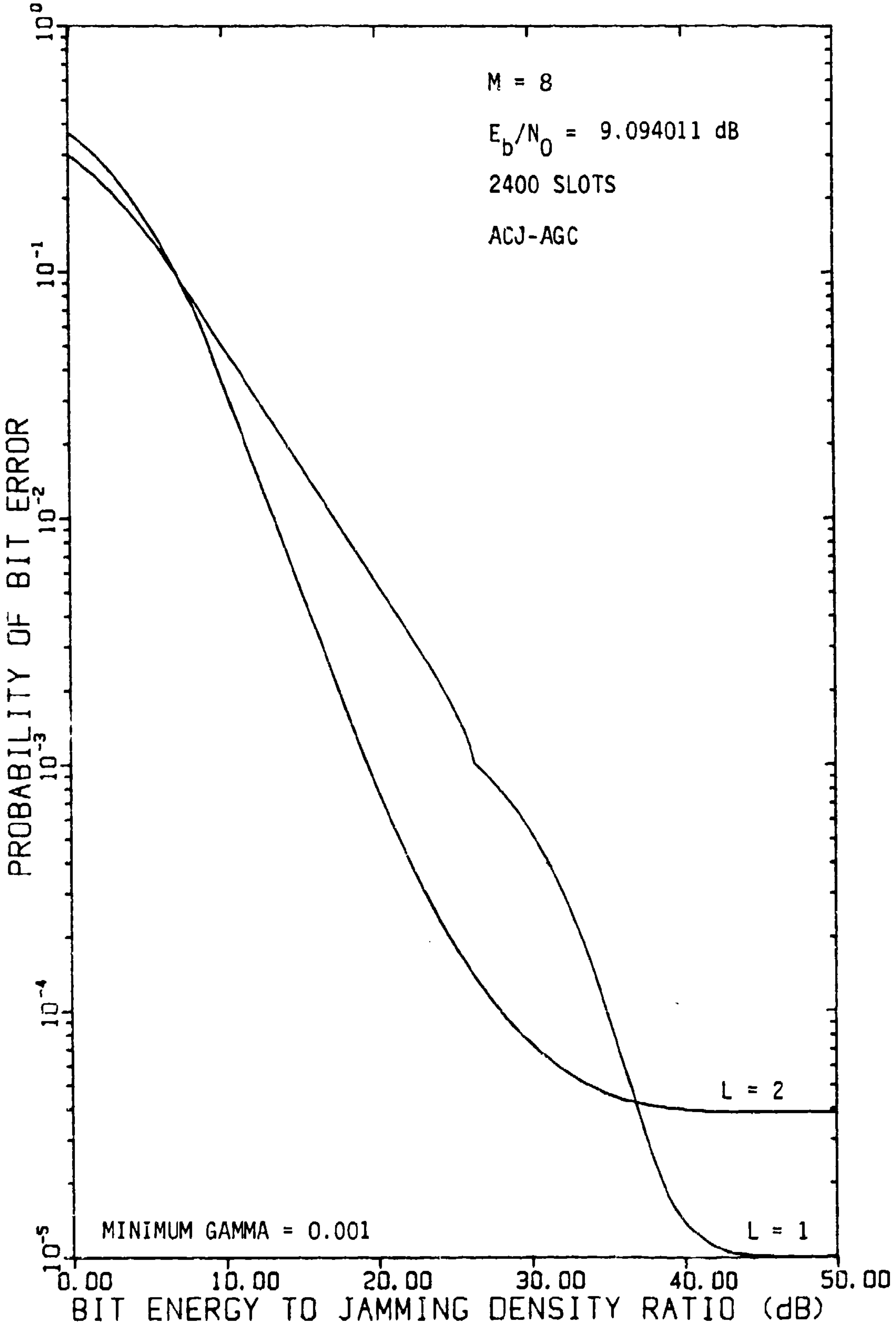


FIGURE 4.4-28 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER AND M=8 WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

5.0 FH/RMFSK PERFORMANCE USING CLIPPER RECEIVER

We now undertake analysis of a third type of ECCM receiver for FH/RMFSK, in which the effect of jamming on the symbol decision is reduced by soft-limiting or clipping the per-hop symbol decision variables $\{z_{mk};m=1,2,\ldots,M;k=1,2,\ldots,L\}$. The receiver structure is diagrammed in Figure 5.0-1. In each of the M dehopped symbol channels, the square-law envelope detector samples are clipped at some level η prior to summing to perform the symbol decision. Because the contribution of a jammed hop to the decision variables is at most η, no matter how strong the jammer noise power, it is expected that an improved performance will result. The clipping threshold η is to be chosen to minimize the error probability when there is no jamming.

In previous analyses of the clipper receiver (for conventional FH/MFSK) we had employed a numerical convolution technique to obtain the distributions of the decision variables. Here we shall obtain the needed probability density functions (pdf's) directly, through analysis.

5.1 DISTRIBUTIONS OF THE DECISION VARIABLES

We first discuss the general form for the pdf of the sum of clipped square-law envelope detector samples, then apply this form to non-signal and signal channels.

5.1.1 General Form of the pdf.

If the input to a clipper with clipping level η has the pdf $f_1(x), x\geq 0$, then the output has the pdf

$$p_1(x) = \begin{cases} f_1(x) + q\,\delta(x-\eta) & 0\leq x\leq \eta \\ 0, \text{ otherwise;} \end{cases} \qquad (5.1\text{-}1a)$$

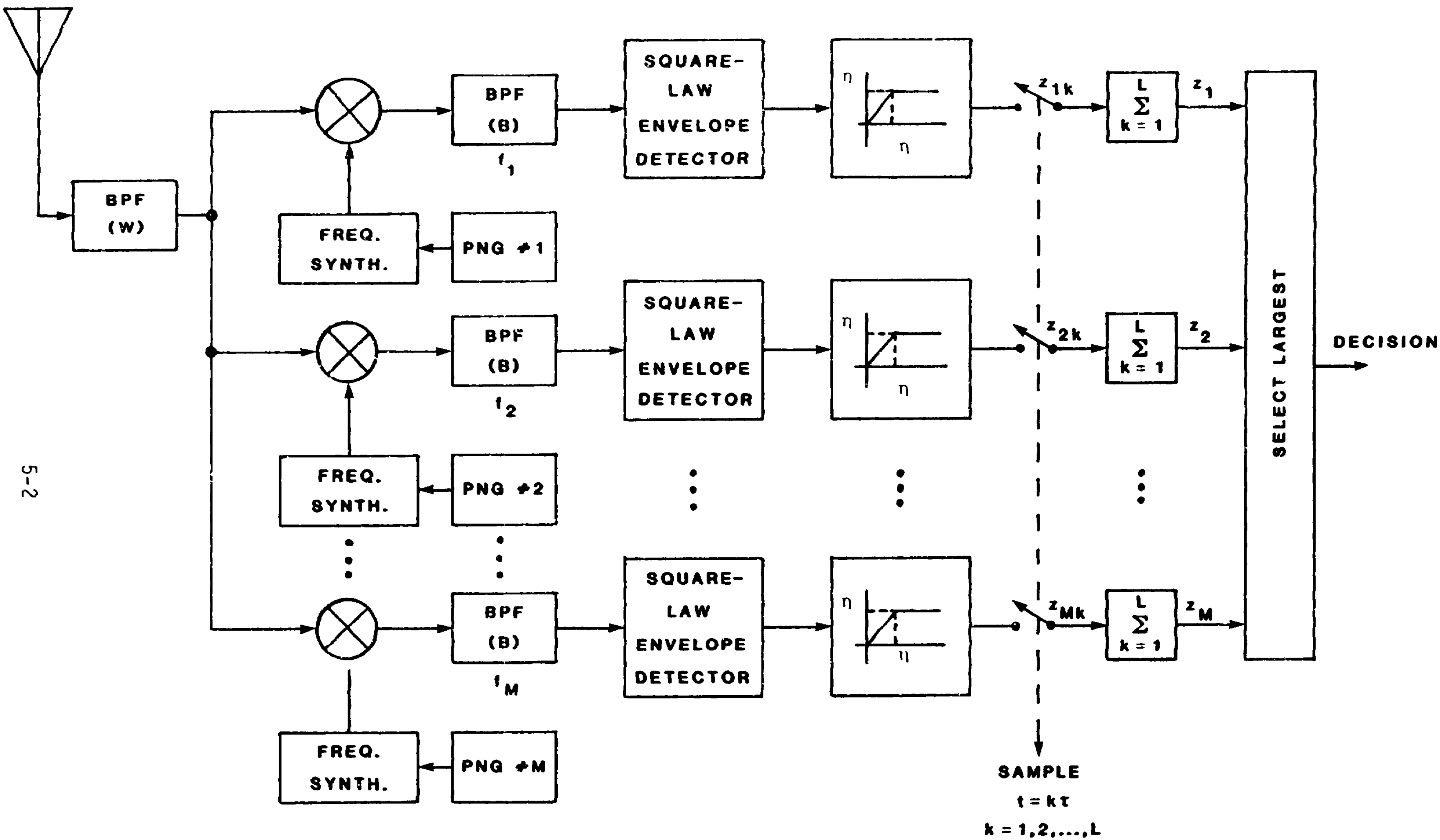


FIGURE 5.0-1 SOFT-DECISION FH/RMFSK RECEIVER WITH CLIPPERS

where

$$q = \Pr\{\text{input} > \eta\} = \int_{\eta}^{\infty} d\alpha \; f_1(\alpha) \tag{5.1-1b}$$

This fact is illustrated by figure 5.1-1.

Now, since individual hops are jammed independently and in any combination, we introduce the notations

$$f_L(x;\ell) \equiv \text{non-delta function part of the pdf for the sum of L clipped samples when } \ell \text{ hops in that channel are jammed.} \tag{5.1-2a}$$

$$q_0 = \Pr\{\text{one sample} > \eta \mid \text{not jammed}\} \tag{5.1-2b}$$

$$q_1 = \Pr\{\text{one sample} > \eta \mid \text{jammed}\}. \tag{5.1-2c}$$

Note that it is sufficient to specify only ℓ, the number of hops jammed; the order in which the jamming occurs does not affect the sum. Using this notation, the pdf for a single clipped envelope sample is

$$p_1(x;0) = f_1(x;0) + q_0\,\delta(x-\eta)\,, \text{ hop not jammed;} \tag{5.1-3a}$$

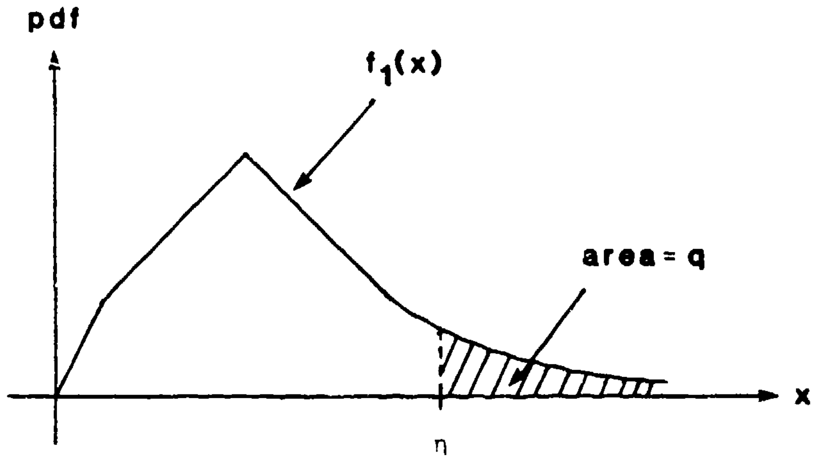
$$p_1(x;1) = f_1(x;1) + q_1\,\delta(x-\eta)\,, \text{ hop jammed;} \tag{5.1-3b}$$

and it is understood that the pdf is zero outside the interval $0 \leq x \leq \eta$.

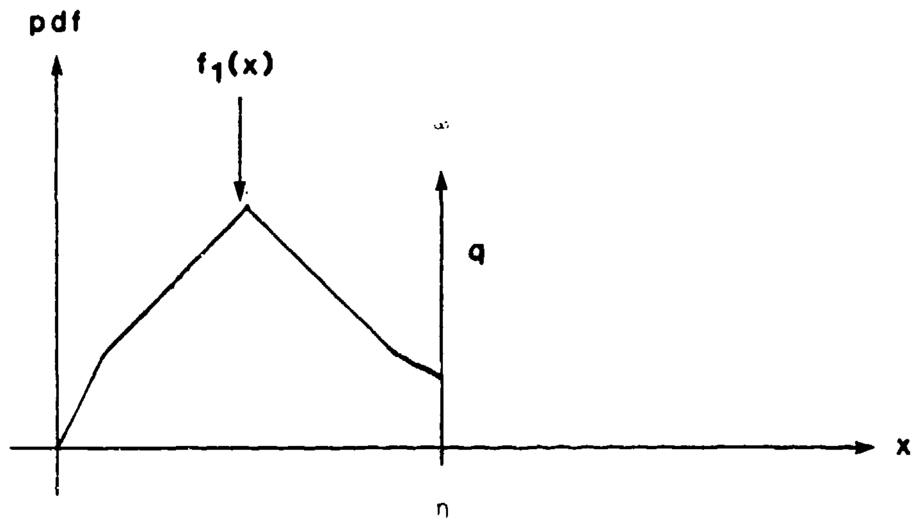
For ℓ hops jammed, the pdf of the sum of L clipped samples can be expressed as the convolution

$$\underbrace{p_1(x;0)*p_1(x;0)*\ldots*p_1(x;0)}_{L-\ell \text{ pdf's}}*\underbrace{p_1(x;1)*\ldots p_1(x;1)}_{\ell \text{ pdf's}}\,. \tag{5.1-4}$$

Thus we have the following general expressions for the sum's pdf for L=2 to 4:



(a) pdf before clipping



(b) pdf after clipping

FIGURE 5.1-1 EFFECTS OF CLIPPING ON pdf

$$p_2(x;0) = f_2(x;0) + q_0^2\,\delta(x-2\eta)\ ,\ 0\leq x\leq 2\eta; \tag{5.1-5a}$$

where

$$f_2(x;0) = \begin{cases} \int_0^x dw\ f_1(x-w;0)f_1(w;0), & 0\leq x<\eta; \\ \int_{x-\eta}^{\eta} dw\ f_1(x-w;0)f_1(w;0) \\ \qquad + 2q_0 f_1(x-\eta;0)\ , & \eta\leq x\leq 2\eta. \end{cases} \tag{5.1-5b}$$

$$p_2(x;1) = f_2(x;1) + q_0 q_1\ \delta(x-2\eta)\ ,\ 0\leq x\leq 2\eta; \tag{5.1-6a}$$

where

$$f_2(x;1) = \begin{cases} \int_0^x dw\ f_1(x-w;1)f_1(w;0)\ , & 0\leq x<\eta\ ; \\ \int_{x-\eta}^{\eta} dw\ f_1(x-w;1)f_1(w;0) \\ \qquad + q_0 f_1(x-\eta;1) + q_1 f_1\ (x-\eta;0), & \eta\leq x\leq 2\eta\ . \end{cases} \tag{5.1-6b}$$

$$p_2(x;2) = f_2(x;2) + q_1^2\delta(x-2\eta)\ ,\ \ 0\leq x\leq 2\eta; \tag{5.1-7a}$$

where

$$f_2(x;2) = \begin{cases} \int_0^x dw\ f_1(x-w;1)f_1(w;1)\ , & 0\leq x<\eta; \\ \int_{x-\eta}^{\eta} dw\ f_1(x-w;1)f_1(w;1) \\ \qquad + 2q_1 f_1(x-\eta;1)\ , & \eta\leq x\leq 2\eta. \end{cases} \tag{5.1-7b}$$

$$p_3(x;0) = f_3(x;0) + q_0^3\delta(x-3\eta)\ ,\ \ 0\leq x\leq 3\eta; \tag{5.1-8a}$$

where

$$f_3(x;0) = \begin{cases} \int_0^x dw\ f_1(x-w;0)\ f_2(w;0)\ , & 0\leq x<\eta; \\ \int_{x-\eta}^{x} dw\ f_1(x-w;0)\ f_2(w;0) \\ \qquad + q_0 f_2(x-\eta;0)\ , & \eta\leq x<2\eta; \end{cases} \tag{5.1-8b}$$

and

$$f_3(x;0) = \begin{cases} \displaystyle\int_{x-\eta}^{2\eta} dw\ f_1(x-w;0) f_2(w;0) \\ \quad +q_0 f_2(x-\eta;0) \\ \quad +q_0^2 f_1(x-2\eta;0)\ , \quad 2\eta \leq x \leq 3\eta\ . \end{cases} \tag{5.1-8c}$$

$$p_3(x;1) = f_3(x;1) + q_0^2 q_1 \delta(x-3\eta)\ , \quad 0 \leq x \leq 3\eta; \tag{5.1-9a}$$

where

$$f_3(x;1) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;1)\ f_2(w;0), \quad 0 \leq x < \eta \\ \displaystyle\int_{x-\eta}^{x} dw\ f_1(x-w;1) f_2(w;0) \\ \quad + q_1 f_2(x-\eta;0), \quad \eta \leq x < 2\eta \\ \displaystyle\int_{x-\eta}^{2\eta} dw\ f_1(x-w;1) f_2(w;0) \\ \quad + q_1 f_2(x-\eta;0) \\ \quad + q_0^2 f_1(x-2\eta;1)\ , \quad 2\eta \leq x \leq 3\eta\ . \end{cases} \tag{5.1-9b}$$

$$p_3(x;2) = f_3(x;2) + q_0 q_1^2 \delta(x-3\eta), \quad 0 \leq x \leq 3\eta; \tag{5.1-10a}$$

where

$$f_3(x;2) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;0)f_2(w;2)\ , & 0\le x<\eta; \\ \displaystyle\int_{x-\eta}^x dw\ f_1(x-w;0)f_2(w;2) + q_0 f_2(x-\eta;2)\ , & \eta\le x<2\eta; \\ \displaystyle\int_{x-\eta}^{2\eta} dw\ f_1(x-w;0)f_2(w;2) + q_0 f_2(x-\eta;2) + q_1^2 f_1(x-2\eta;0)\ , & 2\eta\le x\le 3\eta. \end{cases} \tag{5.1-10b}$$

$$p_3(x;3) = f_3(x;3) + q_1^3\delta(x-3\eta),\quad 0<x<3\eta\ ; \tag{5.1-11a}$$

where

$$f_3(x;3) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;1)f_2(w;2)\ , & 0\le x<\eta; \\ \displaystyle\int_{x-\eta}^x dw\ f_1(x-w;1)f_2(w;2) + q_1 f_2(x-\eta;2)\ , & \eta\le x<2\eta; \\ \displaystyle\int_{x-\eta}^{2\eta} dw\ f_1(x-w;1)f_2(w;2) + q_1 f_2(x-\eta;2) + q_1^2 f_1(x-2\eta;1)\ , & 2\eta\le x\le 3\eta. \end{cases} \tag{5.1-11b}$$

$$p_4(x;0) = f_4(x;0) + q_0^4\delta(x-4\eta), \quad 0 \le x \le 4\eta; \tag{5.1-12a}$$

where

$$f_4(x;0) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;0)f_3(w;0)\ , & 0 \le x < \eta; \\ \displaystyle\int_{x-\eta}^x dw\ f_1(x-w;0)f_3(w;0) \\ \qquad + q_0 f_3(x-\eta;0), & \eta \le x < 3\eta \\ \displaystyle\int_{x-\eta}^{3\eta} dw\ f_1(x-w;0)f_3(w;0) \\ \qquad + q_0 f_3(x-\eta;0) \\ \qquad + q_0^3 f_1(x-3\eta;0)\ , & 3\eta \le x \le 4\eta. \end{cases} \tag{5.1-12b}$$

$$p_4(x;1) = f_4(x;1) + q_0^3 q_1\delta(x-4\eta), \quad 0 \le x \le 4\eta; \tag{5.1-13a}$$

where

$$f_4(x;1) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;1)f_3(w;0)\ , & 0 \le x \le \eta; \\ \displaystyle\int_{x-\eta}^x dw\ f_1(x-w;1)f_3(w;0) \\ \qquad + q_1 f_3(x-\eta;0)\ , & \eta \le x < 3\eta; \\ \displaystyle\int_{x-\eta}^{3\eta} dw\ f_1(x-w;1)f_3(w;0) \\ \qquad + q_1 f_3(x-\eta;0) \\ \qquad + q_0^3 f_1(x-3\eta;1)\ , & \eta < x < 4\eta. \end{cases} \tag{5.1-13b}$$

$$p_4(x;2) = f_4(x;2) + q_0^2 q_1^2 \delta(x-4\eta), \quad 0 \le x \le 4\eta; \tag{5.1-14a}$$

where

$$f_4(x;2) = \begin{cases} \displaystyle\int_0^x dw\ f_2(x-w;2) f_2(w;0)\ , & 0 \le x < 2\eta; \\ \displaystyle\int_{x-2\eta}^{2\eta} dw\ f_2(x-w;2) f_2(w;0) \\ \quad + q_0^2 f_2(x-2\eta;2) \\ \quad + q_1^2 f_2(x-2\eta;0), & 2\eta \le x \le 4\eta. \end{cases} \tag{5.1-14b}$$

$$p_4(x;3) = f_4(x;3) + q_0 q_1^3 \delta(x-4\eta)\ , \quad 0 \le x \le 4\eta; \tag{5.1-15a}$$

where

$$f_4(x;3) = \begin{cases} \displaystyle\int_0^x dw\ f_1(x-w;0) f_3(w;3)\ , & 0 \le x < \eta; \\ \displaystyle\int_{x-\eta}^{x} dw\ f_1(x-w;0) f_3(w;3) \\ \quad + q_0 f_3(x-\eta;3), & \eta \le x < 3\eta; \\ \displaystyle\int_{x-\eta}^{3\eta} dw\ f_1(x-w;0) f_3(w;3) \\ \quad + q_0 f_3(x-\eta;3) \\ \quad + q_1^3 f_1(x-3\eta;0)\ , & 3\eta \le x \le 4\eta. \end{cases} \tag{5.1-15b}$$

$$p_4(x;4) = f_4(x;4) + q_1^4 \delta(x-4\eta)\ , \quad 0 \le x \le 4\eta; \tag{5.1-16a}$$

$$f_4(x;4) = \begin{cases} \displaystyle\int_0^x dw\, f_1(x-w;1) f_3(w;3)\ , & 0 \le x < \eta; \\ \displaystyle\int_{x-\eta}^x dw\, f_1(x-w;1) f_3(w;3) \\ \qquad + q_1 f_3(x-\eta;3)\ , & \eta \le x < 3\eta\ ; \\ \displaystyle\int_{x-\eta}^{3\eta} dw\, f_1(x-w;1) f_3(w;3) \\ \qquad + q_1 f_3(x-\eta;3) \\ \qquad + q_1^3 f_1(x-3\eta;1)\ , & 3\eta \le x \le 4\eta. \end{cases} \tag{5.1-16b}$$

5.1.2 Non-Signal Channel pdf.

Assuming without loss of generality that the received signal power S is present in the first (m=1) of M dehopped symbol frequency channels, the remaining channels (m=2,3,...,M) contain only background noise and possibly jamming noise. The samples of the square-law envelope detectors in these channels are independent chi-squared random variables with two degrees of freedom, multiplied by σ_{mk}^2, where

$$\sigma_{mk}^2 = \begin{cases} \sigma_N^2 = N_0 B, & \text{hop not jammed} \\ \sigma_T^2 = (N_0 + N_J/\gamma)B, & \text{hop jammed.} \end{cases} \tag{5.1-17}$$

Consequently, the pdf of the unclipped samples is

$$f_1(x) = \frac{1}{2\sigma^2_{mk}} e^{-x/2\sigma^2_{mk}} , \quad x \geq 0;$$

$$= \begin{cases} a\, e^{-ax} , & x \geq 0, \text{ hop not jammed;} \\ b\, e^{-bx} , & x \geq 0, \text{ hop jammed;} \end{cases} \tag{5.1-18a}$$

using

$$a \equiv 1/2\sigma^2_N \quad , \; b \equiv 1/2\sigma^2_T \; . \tag{5.1-18b}$$

Also, we have from (5.1-2b and c)

$$q_0 = e^{-a\eta} \; , \; q_1 = e^{-b\eta} \; . \tag{5.1-19}$$

In order to distinguish the non-signal channel pdf's from that of the signal channel, we adopt the notation

$$g_L(x;\ell) = f_L(x;\ell,S=0) \tag{5.1-20}$$

for the non-delta function part of the pdf of the sum of L clipped samples when ℓ hops in that channel are jammed. Thus we have for channels $\{m:m>2\}$, the sum pdf

$$p_{z_m}(x) = \begin{cases} g_L(x;\ell_m) + (q_0)^{L-\ell_m} (q_1)^{\ell m} \delta(x-L\eta), & 0 \leq x \leq L\eta; \\ 0, \text{ otherwise.} \end{cases} \tag{5.1-21}$$

Substituting (5.1-18) in the general convolutional formulas in Section 5.1.1 yields the pdf's listed in Table 5.1-1 for L=1 to 3.

TABLE 5.1-1 NON-SIGNAL CHANNEL PROBABILITY DENSITY FUNCTIONS, L = 1 TO 3

L	ℓ	$p_{zm}(x;\ \ell)$
1	0	$ae^{-ax} + e^{-a\eta}\ \delta(x-\eta),\ 0 \le x \le \eta;\ 0,$ otherwise. $\quad a \equiv 1/2\sigma_N^2$
	1	$be^{-ax} + e^{-b\eta}\ \delta(x-\eta),\ 0 \le x \le \eta;\ 0,$ otherwise. $\quad b \equiv 1/2\sigma_T^2$
2	0	$a^2xe^{-ax},\ 0 \le x < \eta;\ [2a + a^2(2\eta-x)]e^{-ax} + e^{-2a\eta}\ \delta(x-2\eta),\ \eta \le x \le 2\eta;\ 0,$ otherwise
	1	$\frac{ab}{a-b}(e^{-bx}-e^{-ax}),\ 0\le x<\eta;\ \frac{a^2}{a-b}e^{-b\eta-a(x-\eta)} - \frac{b^2}{a-b}e^{-a\eta-b(x-\eta)} + e^{-(a+b)\eta}\delta(x-2\eta),\ \eta\le x\le 2\eta; 0,$ otherwise
	2	same as for $\ell = 0$, but with a replaced by b
3	0	$\tfrac{1}{2}a^3x^2e^{-ax},\ 0 \le x < \eta; \quad 0,\ x > 3\eta,\ x < 0$ $a^2[\tfrac{1}{2}a\eta^2 + (3+a\eta)(x-\eta) - a(x-\eta)^2]\ e^{-ax},\ \eta \le x < 2\eta$ $a[3+3a\eta+\tfrac{1}{2}a^2\eta^2 - a(3+a\eta)(x-2\eta) + \tfrac{1}{2}a^2(x-2\eta)^2]\ e^{-ax} + e^{-3a\eta}\ \delta(x-3\eta),\ 2\eta \le x \le 3\eta.$
	1	$\frac{a^2b}{(a-b)^2}\left\{e^{-bx} - e^{-ax}[1+(a-b)x]\right\},\ 0 \le x < \eta; \quad 0,\ x > 3\eta,\ x < 0;$ $\frac{a}{a-b}e^{-a(x-\eta)}\left\{a\left[\frac{b}{a-b} + a(x-\eta)\right]e^{-b\eta} + b\left[\frac{a}{a-b} - (2+a\eta)+a(x-\eta)\right]e^{-a\eta}\right\} - \frac{2ab^2}{(a-b)^2}e^{-a\eta-b(x-\eta)},\ \eta\le x<2\eta$ $\frac{a^2}{a-b}e^{-(a+b)\eta-a(x-2\eta)}\left[\frac{2a-3b}{a-b} + a\eta-a(x-2\eta)\right] + \frac{b^3}{(a-b)^2}e^{-2a\eta-b(x-2\eta)} + e^{-(2a+b)\eta}\ \delta(x-3\eta),\ 2\eta\le x\le 3\eta.$
	2	same as for $\ell = 1$, but with a and b exchanged
	3	same as for $\ell = 0$, but with a replaced by b

In the conditional probability of error expression,

$$P_s(e|\ell,L) = 1 - E_{z_1}\left\{ \prod_{m=2}^{M} \Pr\{z_m<z_1|\ell,L\} \right\}, \qquad (5.1\text{-}22)$$

the cumulative distribution function for the non-signal channels is needed, written

$$G_L(x;\ell) \triangleq \Pr\{z_m \le x|\ell,L\}. \qquad (5.1\text{-}23)$$

This function is given in Table 5.1-2 for L=1 to 3.

5.1.3 Signal Channel pdf.

The samples of the square-law envelope detector in the signal channel are independent noncentral chi-squared random variables with two degrees of freedom, multiplied by σ_{1k}^2, and with noncentrality parameters

$$\lambda_k = 2S/\sigma_{1k}^2 = \begin{cases} 2S/\sigma_N^2 = 2\rho_N', & \text{hop not jammed} \\ 2S/\sigma_T^2 = 2\rho_T, & \text{hop jammed.} \end{cases} \qquad (5.1\text{-}24)$$

Consequently the pdf of the unclipped samples is

$$f_1(x) = \frac{1}{2\sigma_{1k}^2}\, e^{-(x+2S)/2\sigma_{1k}^2}\, I_0(\sqrt{2Sx}/\sigma_{1k}^2)$$

$$= \begin{cases} a\, e^{-a(x+2S)} I_0(2a\sqrt{2Sx}), & x\ge 0, \text{ hop not jammed;} \\ b\, e^{-b(x+2S)} I_0(2b\sqrt{2Sx}), & x\ge 0, \text{ hop jammed;} \end{cases} \qquad (5.1\text{-}25)$$

where a and b are given by (5.1-18b). To distinguish the signal case from the non-signal case, the q_0 and q_1 defined by (5.1-2) will be written in the upper case; the values are

TABLE 5.1-2 NON-SIGNAL CHANNEL CUMULATIVE DISTRIBUTION FUNCTIONS, L=1 TO 3

L	ℓ	$G_L(x;\ \ell) = \Pr\{z_m \leq x \mid \ell,\ L\},\ m \geq 2$
1	0	$1-e^{-ax},\ 0 \leq x < \eta;\ 1,\ x \geq \eta\ .$ $\quad a \equiv 1/2\sigma_N^2$
	1	$1-e^{-bx},\ 0 \leq x < \eta;\ 1,\ x \geq \eta\ .$ $\quad b \equiv 1/2\sigma_T^2$
2	0	$1-e^{-ax}(1+ax),\ 0 \leq x < \eta;\ 1-e^{-ax}[1+a(2\eta-x)],\ \eta \leq x < 2\eta;\ 1,\ x \geq 2\eta\ .$
	1	$1-\frac{1}{a-b}[ae^{-bx}-be^{-ax}],\ 0 \leq x < \eta;\ 1-\frac{1}{a-b}\left[ae^{-b\eta-a(x-\eta)}-be^{-a\eta-b(x-\eta)}\right],\ \eta \leq x < 2\eta;\ 1,\ x \geq 2\eta\ .$
	2	$1-e^{-bx}(1+bx),\ 0 \leq x < \eta;\ 1-e^{-bx}[1+b(2\eta-x)],\ \eta \leq x < 2\eta;\ 1,\ x \geq 2\eta$
3	0	$1-e^{-ax}(1+ax+\tfrac{1}{2}a^2x^2),\ 0 \leq x < \eta;\ 1-e^{-ax}[1+a\eta+\tfrac{1}{2}a^2\eta^2+a(1+a\eta)(x-\eta)-a^2(x-\eta)^2],\ \eta \leq x < 2\eta;$ $1-e^{-ax}[1+2a\eta+\tfrac{1}{2}a^2\eta^2-a(2+a\eta)(x-2\eta)+\tfrac{1}{2}a^2(x-2\eta)^2],\ 2\eta \leq x < 3\eta;\ 1,\ x \geq 3\eta\ .$
	1	$1-\frac{a^2}{(a-b)^2}e^{-bx}+\frac{b}{a-b}e^{-ax}\left[\frac{2a-b}{a-b}+ax\right],\ 0 \leq x < \eta;$ $1-e^{-a(x-\eta)}\left\{e^{-b\eta}\left[\frac{a^2}{(a-b)^2}+\frac{a^2}{a-b}(x-\eta)\right]+e^{-a\eta}\left[\frac{b^2}{(a-b)^2}+\frac{ab}{a-b}(x-2\eta)\right]\right\}+\frac{2ab}{(a-b)^2}e^{-a\eta-b(x-\eta)},\ \eta \leq x < 2\eta;$ $1-e^{-(a+b)\eta-a(x-2\eta)}\left[\frac{a(1+a\eta)}{a-b}-\frac{ab}{(a-b)^2}-\frac{a^2}{a-b}(x-2\eta)\right]-\frac{b^2}{(a-b)^2}e^{-2a\eta-b(x-2\eta)},\ 2\eta \leq x < 3\eta;\ 1,\ x \geq 3\eta\ .$
	2	same as for $\ell = 1$, but with a and b exchanged
	3	same as for $\ell = 0$, but with a replaced by b

$$Q_0 = \Pr\{z_{1k} > \eta \mid \text{not jammed}\}$$
$$= Q\left(2\sqrt{aS}, \sqrt{2a\eta}\right) \quad (5.1\text{-}26a)$$

and

$$Q_1 = \Pr\{z_{1k} > \eta \mid \text{jammed}\}$$
$$= Q\left(2\sqrt{bS}, \sqrt{2b\eta}\right), \quad (5.1\text{-}26b)$$

where $Q(x,y)$ is Marcum's Q-function.

For the sum of clipped samples, the pdf is

$$p_{z_1}(x) = \begin{cases} f_L(x;\ell_1) + (Q_0)^{L-\ell}\,(Q_1)^{\ell_1}\delta(x-L\eta)\ , & 0 \le x \le L\eta; \\ 0, \text{ otherwise.} & \end{cases} \quad (5.1\text{-}27)$$

Substituting (5.1-25) in the general convolutional formulas in Section 5.1.1 yields the pdf's listed in Table 5.1-3 for L=1, 2 and Table 5.1-4 for L=3.

5.2 ERROR PROBABILITY FORMULATION

Having the pdf's for the FH/RMFSK clip-and-sum decision variables $\{z_m\}$, we can formulate the probability of error.

5.2.1 Conditional Probability of Error.

The probability of symbol error, conditioned on the jamming event $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$ where ℓ_m is the number of hops jammed (out of L hops) in channel m, can be expressed as parametric in η, the clipping threshold, by

TABLE 5.1-3 SIGNAL CHANNEL PROBABILITY DENSITY FUNCTIONS, L = 1,2

L	ℓ	$p_{z_1}(x\|\ell, L)$
1	0	$ae^{-a(x+2S)}\, I_0\left(2a\sqrt{2xS}\right) + Q_0\delta(x-\eta),\ 0 \le x \le \eta.$ $\quad a \equiv 1/2\sigma_N^2,\ Q_0 = Q\left(2\sqrt{aS},\ \sqrt{2a\eta}\right)$
	1	$be^{-b(x+2S)}\, I_0\left(2b\sqrt{2xS}\right) + Q_1\delta(x-\eta),\ 0 \le x \le \eta.$ $\quad b \equiv 1/2\sigma_N^2,\ Q_1 = Q\left(2\sqrt{bS},\ \sqrt{2b\eta}\right)$
2	0	$\tfrac{1}{2}a\sqrt{x/S}\, e^{-a(x+4S)}\, I_1\left(4a\sqrt{Sx}\right),\ 0 \le x < \eta; \quad 0,\ x > 2\eta,\ x < 0;$ $2aQ_0e^{-a(x-\eta+2S)}\, I_0\left(2a\sqrt{2S(x-\eta)}\right) + a^2e^{-a(x+4S)}\int_{x-\eta}^{\eta} dw\, I_0\left(2a\sqrt{2Sw}\right) I_0\left(2a\sqrt{2S(x-w)}\right)$ $+ Q_0^2\delta(x-2\eta),\ \eta \le x \le 2\eta.$
	1	$abe^{-bx-2(a+b)S}\int_0^x dw\, e^{-(a-b)w}\, I_0\left(2a\sqrt{2Sw}\right) I_0\left(2b\sqrt{2S(x-w)}\right),\ 0 \le x < \eta;\ 0,\ x > 2\eta,\ x < 0;$ $bQ_0e^{-b(x-\eta+2S)}\, I_0\left(2b\sqrt{2S(x-\eta)}\right) + aQ_1e^{-a(x-\eta+2S)}\, I_0\left(2a\sqrt{2S(x-\eta)}\right)$ $+ abe^{-bx-2(a+b)S}\int_{x-\eta}^{\eta} dw\, e^{-(a-b)w}\, I_0\left(2a\sqrt{2Sw}\right) I_0\left(2b\sqrt{2S(x-w)}\right) + Q_0Q_1\delta(x-2\eta),\ \eta \le x \le 2\eta.$
	2	same as for $\ell = 0$, but with a replaced by b and Q_0 replaced by Q_1.

TABLE 5.1-4 SIGNAL CHANNEL PROBABILITY DENSITY FUNCTIONS, L = 3

L	ℓ	$p_{z1}(x\mid\ell, L)$
3	0	$(ax/6S)\, e^{-a(x+6S)} I_2(2a\sqrt{6Sx}),\ 0 \le x < \eta;\ 0,\ x > 3\eta,\ x < 0;$ $\frac{3}{2}\, aQ_0\sqrt{(x-\eta)/S}\; e^{-a(x-\eta+4S)} I_1(4a\sqrt{S(x-\eta)}) + \frac{1}{2}a^2 e^{-a(x+6S)} \int_{x-\eta}^{\eta} dw \sqrt{w/S}\; I_1(4a\sqrt{Sw}) I_0(2a\sqrt{2S(x-w)})$ $+\ a^3\, e^{-a(x+6S)} \int_{\eta}^{x} dw\; I_0(2a\sqrt{2S(x-w)}) \int_{w-\eta}^{\eta} dz\; I_0(2a\sqrt{2Sz}) I_0(2a\sqrt{2S(w-z)}),\ \eta \le x < 2\eta;$ $3aQ_0^2\, e^{-a(x-2\eta+2S)} I_0(2a\sqrt{2S(x-2\eta)})$ $+\ 3a^2Q_0\, e^{-a(x-\eta+4S)} \int_{x-\eta}^{2\eta} dw\; I_0(2a\sqrt{2S(x-w)}) I_0(2a\sqrt{2S(w-\eta)})\ +\ Q_0^3\, \delta(x-3\eta)$ $+\ a^3\, e^{-a(x+6S)} \int_{x-\eta}^{2\eta} dw\; I_0(2a\sqrt{2S(x-w)} \int_{w-\eta}^{\eta} dz I_0(2a\sqrt{2Sz})\; I_0(2a\sqrt{2S(w-z)},\ 2\eta \le x \le 3\eta.$
	1	$\frac{1}{2}ab\; e^{-bx-2(2a+b)S} \int_0^{x} dw\; e^{-(a-b)w} \sqrt{w/S}\; I_0(2b\sqrt{2S(x-w)}) I_1(4a\sqrt{Sw}),\ 0 \le x < \eta;\ \ 0,\ x > 3\eta,\ x < 0;$ $\frac{1}{2}aQ_1\sqrt{(x-\eta)/S}\; e^{-a(x-\eta+4S)} I_1(4a\sqrt{S(x-\eta)}) + \frac{1}{2}ab e^{-bx-(2a+b)2S} \int_{x-\eta}^{\eta} dw \sqrt{w/S}\; I_1(2b\sqrt{2S(x-w)})\; e^{-(a-b)w}$ $+\ 2abQ_0\, e^{-b(x-\eta)-(a+b)2S} \int_0^{x-\eta} dw\; I_0(2a\sqrt{2Sw}) I_0(2b\sqrt{2S(x-\eta-w)})\; e^{-(a-b)w}$ $+\ a^2b\; e^{-bx-(2a+b)2S} \int_{\eta}^{x} dw\; e^{-(a-b)w} I_0(2b\sqrt{2S(x-w)}) \int_{w-\eta}^{\eta} dz I_0(2a\sqrt{2Sz})\; I_0(2a\sqrt{2S(w-z)},\ \eta \le x \le 2\eta;$ $bQ_0^2\, e^{-b(x-2\eta+2S)} I_0(2b\sqrt{2S(x-2\eta)}) + 2aQ_0Q_1\; e^{-a(x-2\eta+2S)} I_0(2a\sqrt{2S(x-2\eta)})$ $+\ a^2Q_1\; e^{-a(x-\eta+4S)} \int_{x-\eta}^{2\eta} dw\; I_0(2a\sqrt{2S(x-w)}) I_0(2a\sqrt{2S(w-\eta)})$ $+\ 2abQ_0\; e^{-bx+a\eta-(a+b)2S} \int_{x-\eta}^{2\eta} dw\; I_0(2b\sqrt{2S(x-w)}) I_0(2a\sqrt{2S(w-\eta)})\; e^{-(a-b)w}$ $+\ a^2b\; e^{-bx-(2a+b)2S} \int_{x-\eta}^{2\eta} dw\; e^{-(a-b)w} I_0(2b\sqrt{2S(x-w)}) \int_{w-\eta}^{\eta} dz\; I_0(2a\sqrt{2Sz}) I_0(2a\sqrt{2S(w-z)}),$ $2\eta \le x \le 3\eta.$
	2	same as for $\ell = 1$, but with a and b exchanged and Q_0 and Q_1 exchanged
	3	same as for $\ell = 0$, but with a replaced by b and Q_0 replaced by Q_1

$$P_s(e;\eta|L,\underline{\ell}) = 1 - \Pr\{C \equiv \text{correct decision};\eta|L,\underline{\ell}\}. \tag{5.2-1}$$

Since there is clipping, there is a finite probability that one or more of the $\{z_m\}$ are equal to L . Thus an appropriate formulation, assuming a randomized decision rule, is

$$\begin{aligned}
\Pr\{C;\eta|L,\underline{\ell}\} &= \sum_{p=0}^{M} \Pr\{C \text{ and } (p \text{ channels } = L\eta);\eta|L,\underline{\ell}\} \\
&= \sum_{p=0}^{M} \Pr\{C \text{ and } (p \text{ channels}=L\eta, \text{ including signal channel});\eta|L,\underline{\ell}\} \\
&= \Pr\{C \text{ and no channels}=L\eta;\eta|L,\underline{\ell}\} \\
&\quad + \sum_{p=0}^{M-1} \Pr\{C \text{ and } (\text{signal channel}=L\eta) \text{ and } (p \text{ non-signal channels}=L\eta); \\
&\qquad\qquad \eta|L,\underline{\ell}\}.
\end{aligned} \tag{5.2-2}$$

The first term in (5.2-2) is

$$\int_0^{L\eta} dx\, f_L(x;\ell_1) \prod_{m=2}^{M} G_L(x;\ell_m) , \tag{5.2-3}$$

where $f_L(x;\ell_1)$ is the non-delta function part of the signal channel's pdf, and $G_L(x;\ell_m)$, $m\geq 2$, is the cumulative distribution function for the non-signal channels. (We assume without loss of generality that the signal channel is the first one, i.e., m=1.) The sum in (5.2-2) can be expanded as

$$\begin{aligned}
&\sum_{p=0}^{M-1} \Pr\{C;\eta|(z_1=L\eta) \text{ and } (p \text{ non-signal channels}=L\eta);\ L,\underline{\ell}\} \\
&\qquad\qquad \cdot \Pr\{p \text{ non-signal channels}=L\eta|L,\ell_2,\ldots,\ell_m\}. \\
&\qquad\qquad \cdot \Pr\{z_1= L\eta|L,\ell_1\} \\
&= \sum_{p=0}^{M-1} \frac{1}{p+1} \Pr\{p \text{ non-signal channels}=L\eta|L,\ell_2,\ldots\ell_M\} \cdot P_{1L}(\ell_1),
\end{aligned} \tag{5.2.4a}$$

and we use $P_{1L}(\ell_1) \triangleq Q_0^{L-\ell_1} Q_1^{\ell_1}$. (5.2-4b)

Using a similar notation, the probability of a non-signal channel's being equal to $L\eta$, i.e., $z_m = L\eta$ for $m \geq 2$, is

$$P_{2L}(\ell_m) \triangleq q_0^{L-\ell_m} q_1^{\ell_m} = e^{-(L-\ell_m)a\eta - \ell_m b\eta} . \tag{5.2-5}$$

Now, there are $\binom{M-1}{p}$ ways for p of the M-1 non-signal channels to be selected as either $z_m = L\eta$ or $z_m < L\eta$. However, it is necessary to account for the fact that these channels may have different numbers of hops jammed, ℓ_m. Let

$$\nu_m = \begin{cases} 1 & \text{if } z_m = L\eta \\ 0 & \text{if } z_m < L\eta \ ; \end{cases} \tag{5.2-6}$$

using this indicator variable, and the vector

$$\underline{\nu} = (\nu_2, \nu_3, \ldots, \nu_M), \tag{5.2-7}$$

we can write

$$\Pr\{p \text{ non-signal channels} = L\eta | L, \ell_2, \ldots, \ell_M\}$$

$$= \sum_{\underline{\nu}} \prod_{m=2}^{M} \left\{ \nu_m P_{2L}(\ell_m) + (1-\nu_m)\left[1 - P_{2L}(\ell_m)\right] \right\} \cdot \delta\left(\sum_m \nu_m, p\right) \tag{5.2-8a}$$

where

$$\delta(n,p) \triangleq \begin{cases} 1, & p = n \\ 0, & p \neq n. \end{cases} \tag{5.2-8b}$$

For example, if all the non-signal channels have the same number of jammed hops, $\ell_m=\ell$, then (5.2-8a) is evaluated as

$$\left[P_{2L}(\ell)\right]^p \left[1-P_{2L}(\ell)\right]^{m-1-p} \sum_{\underline{\nu}} \delta\left(\sum_m \nu_m, p\right)$$
$$= \binom{M-1}{p}\left[P_{2L}(\ell)\right]^p \left[1-P_{2L}(\ell)\right]^{-1-p}. \tag{5.2-9}$$

Substituting (5.2-8) and (5.2-3) into the error expression results in

$$P_s(e;\eta|L,\underline{\ell}) = 1 - \int_0^{L\eta} dx\, f_L(x;\ell_1) \prod_{m=2}^{M} G_L(x;\ell_m)$$
$$- P_{1L}(\ell_1) \sum_{p=0}^{M-1} \frac{1}{1+p} \sum_{\underline{\nu}} \prod_{m=2}^{M} \left\{\nu_m P_{2L}(\ell_m) + (1-\nu_m)\left[1-P_{2L}(\ell_m)\right]\right\} \delta\left(\sum_m \nu_m, p\right). \tag{5.2-10}$$

5.2.1.1 Special case: L=1 (one hop/symbol).

For L=1, we have

$$f_1(x;\ell_1) = c_1 e^{-c_1(x+2s)} I_0\left(2c_1\sqrt{2Sx}\right); \tag{5.2-11}$$

using

$$c_m = \begin{cases} a & \text{if } \ell_m = 0 \\ b & \text{if } \ell_m = 1; \end{cases} \tag{5.2-12}$$

$$P_{21}(\ell_m) = e^{-c_m\eta} \quad ; \tag{5.2-13}$$

and

$$\prod_{m=2}^{M} G_L(x;\ell_m) = \prod_{m=2}^{M} (1-e^{-c_m x})$$

$$= (1-e^{-ax})^{n_0} \, (1-e^{-bx})^{M-1-n_0}$$

$$= \sum_{k=0}^{n_0} \sum_{r=0}^{M-1-n_0} \binom{n_0}{k} \binom{M-1-n_0}{r} (-1)^{k+r} \, e^{-(ka+rb)x} \qquad (5.2\text{-}14a)$$

where

$$n_0 \triangleq \#\left(\ell_m = 0, \; m \geq 2\right), \qquad (5.2\text{-}14b)$$

that is, n_0 is the number of unjammed, non-signal channels. Substituting in the error expression (5.2-10) results in

$$P_s(e;\eta \mid 1,\underline{\ell}) = \sum_{\substack{k=0 \\ k+r>0}}^{n_0} \sum_{r=0}^{M-1-n_0} \binom{n_0}{k} \binom{M-1-n_0}{r} (-1)^{k+r+1}$$

$$\cdot \int_0^{\eta} dx \; c_1 \, e^{-2c_1 S-(c_1+ka+rb)x} \, I_0(2c_1\sqrt{Sx}) + Q_{\ell_1}$$

$$- Q_{\ell_1} \sum_{p=0}^{M-1} \frac{1}{1+p} \sum_{\underline{\nu}} \prod_{m=2}^{M} \left\{ (2\nu_m - 1) \, e^{-c_m \eta} + 1 - \nu_m \right\} \delta\left(\sum_m \nu_m, p\right). \qquad (5.2\text{-}15)$$

The integral equals

$$\frac{c_1}{c_1+ka+rb} \exp\left\{ \frac{-(ka+rb)2c_1 S}{c_1+ka+rb} \right\} \left[1 - Q\left(\sqrt{\frac{4c_1^2 S}{c_1+ka+rb}}, \sqrt{2(c_1+ka+rb)\eta}\right)\right]. \qquad (5.2\text{-}16)$$

Also, since $\ell_m = 0$ or 1 when $L = 1$, the last term can be written

$$- Q\left(2\sqrt{c_1 S},\sqrt{2c_1\eta}\right) \sum_{p=0}^{M-1} \frac{1}{1+p} \sum_{p_0=p_{min}}^{p_{max}} \binom{n_0}{p_0}\binom{M-1-n_0}{p-p_0} e^{-p_0(a-b)\eta-pb\eta}$$

$$\times \left(1-e^{-a\eta}\right)^{n_0-p_0} \left(1-e^{-b\eta}\right)^{M-1-n_0-(p-p_0)} , \qquad (5.2\text{-}17)$$

where

$$p_{min} = \max[0,p-(M-1-n_0)] , \; p_{max} = \min(p,n_0) . \qquad (5.2\text{-}17)$$

For example, if M=2, (5.2-15) becomes, since $n_0 = 0$ or 1 ($n_0 \equiv 1-\ell_2$),

$$P_b(e;\eta|1,\underline{\ell}) = \ell_2 \cdot \frac{c_1}{c_1+b} \exp\left\{\frac{-2bc_1 S}{c_1+b}\right\}\left[1 - Q\left(\sqrt{\frac{4c_1^2 S}{c_1+b}} , \sqrt{2(c_1+b)\eta}\right)\right]$$

$$+ (1-\ell_2) \cdot \frac{c_1}{c_1+a} \exp\left\{\frac{-2ac_1 S}{c_1+a}\right\}\left[1-Q\left(\sqrt{\frac{4c_1^2 S}{c_1+a}} , \sqrt{2(c_1+a)\eta}\right)\right]$$

$$- Q\left(2\sqrt{c_1 S},\sqrt{2c_1\eta}\right)\left\{\ell_2 \cdot \left[1-e^{-b\eta} + \frac{1}{2} e^{-b\eta}\right]\right.$$

$$\left. + (1-\ell_2) \cdot \left[1-e^{-a\eta} + \frac{1}{2} e^{-a\eta}\right]\right\}$$

$$+ Q\left(2\sqrt{c_1 S},\sqrt{2c_1\eta}\right) \qquad (5.2\text{-}18)$$

5.2.1.2 Special case: $\underline{\ell} = \underline{0}$ (no jamming).

For this case, we substitute $\underline{\ell} = \underline{0}$ in (5.2-10) to get

$$P_s(e;\eta|L,\underline{C}) = 1 - \int_0^{L\eta} dx\ f_L(x;0)\left[G_L(x;0)\right]^{M-1}$$

$$- Q_0^L \sum_{p=0}^{M-1} \frac{1}{1+p}\binom{M-1}{p} e^{-Lpa\eta}\left(1-e^{-La\eta}\right)^{M-1-p} \tag{5.2-19a}$$

$$= \sum_{k=1}^{M-1}\binom{M-1}{k}(-1)^{k+1}\int_0^{L\eta} dx\ f_L(x;0)\left[1-G_L(x;0)\right]^k$$

$$+ Q_0^L$$

$$- Q_0^L \cdot \frac{e^{La\eta}}{M}\left[1-\left(1-e^{-La\eta}\right)^M\right]. \tag{5.2-19b}$$

Now, $1-G_L(x;0)$ in the integral has the form (see Tables 5.1-3 and 5.1-4)

$$1-G_L(x;0) = \begin{cases} 1, & x<0; \\ e^{-ax}\ h_r(x-r\eta), & r\eta \le x < (r+1)\eta,\ r=0,1,2,\ldots,L-1; \\ 0, & x>L\eta; \end{cases} \tag{5.2-20}$$

where $h_r(x)$ is an $(L-1)$ degree polynomial. Using this form, the integral in (5.2-19) becomes

$$\sum_{r=0}^{L-1}\int_{r\eta}^{(r+1)\eta} dx\ f_L(x;0)\left[h_r(x-r\eta)\right]^k e^{-kax}$$

$$= \sum_{r=0}^{L-1}\int_0^{\eta} dx\ f_L(x+r\eta;0)\left[h_r(x)\right]^k e^{-ka(x+r\eta)}\ . \tag{5.2-21}$$

Noting also from Tables 5.1-5 and 5.1-6 that the signal channel pdf can be written

$$f_L(x;0) = e^{-ax}\, v_r(x-r\eta), \quad r=0,1,\ldots,L-1, \tag{5.2-22}$$

we further manipulate (5.2-21) to obtain

$$\sum_{r=0}^{L-1} e^{-(k+1)ra} \int_0^{\eta} dx\; e^{-(k+1)ax}\, v_r(x)\,[\,h_r(x)]^k. \tag{5.2-23}$$

For example, if L=1, then $h_0(x) \equiv 1$ and $v_0(x) = a\, e^{-2aS}\, I_0\!\left(2a\sqrt{2Sx}\right)$, giving for (5.2-23) the value

$$\frac{1}{1+k}\,\exp\left\{-\frac{2kaS}{k+1}\right\}\left[1-Q\left(\sqrt{\frac{4aS}{1+k}},\sqrt{2a(k+1)\eta}\right)\right] \tag{5.2-24}$$

and

$$P_s(e;\eta|1,\underline{0}) = \sum_{k=1}^{M-1}\binom{M-1}{k}\frac{(-1)^{k+1}}{k+1}\exp\left\{\frac{-2kaS}{1+k}\right\}\left[1-Q\left(\sqrt{\frac{4aS}{1+k}},\sqrt{2a(k+1)\eta}\right)\right].$$

$$+\,Q_0 - Q_0\,\frac{e^{a\eta}}{M}\left[1-\left(1-e^{-a\eta}\right)^M\right]\;. \tag{5.2-25}$$

5.2.2 Total Probability of Error.

For a given number of hops/symbol, L, the total symbol probability of error is

$$P_s(e;\eta,L) = \sum_{\underline{\ell}} \Pr\{\underline{\ell}\}\; P_s(e;\eta|L,\underline{\ell})\,; \tag{5.2-26}$$

the bit error probability is

$$P_b(e;\eta,L) = \frac{M/2}{M-1} P_s(e;\eta,L). \tag{5.2-27}$$

5.2.2.1 Choice of clipping threshold.

The procedure we have adopted for choosing η, the clipping threshold, is the following: choose η to minimize the error probability when there is no jamming. That is,

$$\eta^* : \min_{\eta} \; P_s(e;\eta|L,\underline{0}). \tag{5.2-28}$$

Differentiation of the error expression (5.2-19a) gives an equation for the optimum η thus defined. This equation may be written

$$\begin{aligned}
&- f_L(L\eta;0)\left(1 - e^{-La\eta}\right)^{M-1} \\
&+ \sum_{r=1}^{L-1} \int_0^{\eta} dx \, \frac{\partial}{\partial\eta} \left\{ f_L(x+r\eta;0) \left[G_L(x+r\eta;0)\right]^{M-1} \right\} \\
&+ L\, Q_0^{L-1} \; f_1(\eta;0) \; \frac{e^{La\eta}}{M} \left[1 - \left(1-e^{-La\eta}\right)^M\right] \\
&- Q_0^L \; \frac{La\, e^{La\eta}}{M} \left\{1 - \left(1-e^{-La\eta}\right)^M - Me^{-La\eta}\left(1-e^{-La\eta}\right)^{M-1}\right\}
\end{aligned} \tag{5.2-29}$$

For L=1, the second term is zero and the equation can be put in the form

$$\frac{e^{a\eta}}{M}\left[Q_0 a - f_1(\eta;0)\right]\left\{\left[1 + (M-1)e^{-a\eta}\right]\left(1-e^{-a\eta}\right)^{M-1} - 1\right\}; \tag{5.2-30}$$

this partial derivative with respect to η is negative, indicating the error decreases as η increases, indefinitely.* Thus for L=1 the optimum threshold is infinite (no clipping):

$$\eta^*(L=1) \to \infty \; . \tag{5.2-31}$$

For L>1, it is not feasible to find the optimum threshold by differentiation; it must be done numerically.

5.2.2.2 Total error for L=1.

Since the optimum threshold for L=1 is $\eta^* \to \infty$, we may express the total error probability by using (5.2-15) to obtain

$$P_b(e;L=1) = \frac{M/2}{M-1} \sum_{\underline{\ell}} \Pr\{\underline{\ell}\} \sum_{k=0}^{n_0} \sum_{\substack{r=0 \\ k+r>0}}^{M-1-n_0} \binom{n_0}{k}\binom{M-1-n_0}{k} (-1)^{k+r+1}$$

$$\cdot \frac{c_1}{c_1+ka+rb} \exp\left\{\frac{-(ka+rb)2c_1 S}{c_1+ka+rb}\right\}, \tag{5.2-32a}$$

where $$c_1 = (1-\ell_1)a + \ell_1 b \tag{5.2-32b}$$

and $$n_0 = M-1 - \sum_{m=2}^{M} \ell_m . \tag{5.2-32c}$$

This, of course, gives exactly the same performance as the other receiver processing schemes for L=1.

---

* The last factor in (5.2-30) can be recognized as the quantity $-\sum_{m=2}^{M} \binom{M}{m} (1-e^{-a\eta})^{n-m} e^{-ma\eta} < 0$. The second factor is always positive since $aQ_0 = f_1(\eta;0) + \exp\{-a(2S+\eta)\} \sum_{k>1} a(2S/\eta)^k I_k(2a\sqrt{S\eta})$.

5.3 NUMERICAL RESULTS

In this section we present some numerical results for the clipper receiver's performance. These results are less voluminous than those obtained for other receivers because of the extremely long computer run times for the clipper equations.*

Two stages of computation are required for the clipper receiver. First, the optimum clipping level in the absence of jamming, η_0/σ_N^2, must be found by a numerical search. Then this value must be used in computing the jammed performance. Whenever L, M, or E_b/N_0 changes the optimum clipping level must be recomputed.

The many numerical integrations required to evaluate (5.2-10) using the forms for $f_L(x;\ell_1)$ from Tables 5.1-3 and 5.1-4 and for $G_L(x;\ell)$ from Table 5.1-2 result in very lengthy computations. Consider, for example, the case of M=4, L=2, in which the numerical integrations which are required have the structure

$$1 - \left[\int f_1 g + \int (f_2 + \int f_3) g\right], \quad \ell_1 = 0 \text{ or } \ell_1 = 2 \tag{5.3-1a}$$

$$1 - \left[\int f_1 \int f_2) g + \int (f_3 + \int f_4) g\right], \quad \ell_1 = 1 \tag{5.3-1b}$$

where g is a function of ℓ_2. Each conditional error probability involves one or two double integrations which must be evaluated numerically to sufficient accuracy as to leave several significant digits after subtracting from 1. This subtractive cancellation problem is especially severe for high E_b/N_J when P(e) is small.

For L=3 the situation is even worse, for the numerical integrations take the forms

*It has been noted that for L=1 and any M value, the clipper receiver with optimum threshold is merely a conventional receiver, since that threshold is infinite for L>1. Thus the results computed previously for L=1 apply to this Section as well.

$$1 - \left\{ \int f_1 g + \int \left[f_1 + \int f_2 + \int (f_3 \int f_4) \right] g + \int \left[f_5 + \int f_6 + \int (f_7 \int f_8) \right] g \right\},$$

$$\ell_1 = 0 \text{ or } \ell_1 = 3 \qquad (5.3\text{-}2a)$$

$$1 - \left\{ \int (\int f_1) g + \int \left[f_2 + \int f_3 + \int f_4 + \int f_5 (\int f_6) \right] g \right.$$

$$\left. + \int \left[f_7 + \int f_8 + \int f_9 + \int f_{10} (\int f_{11}) \right] g \right\}, \ \ell_1 = 1 \text{ or } \ell_1 = 2 \quad (5.3\text{-}2b)$$

which results in a worst-case of 2 one-dimensional integrations, 5 two-dimensional integrations, and 2 three-dimensional integrations to be performed numerically. The inner-most integrals must be evaluated to very high precision in order to evaluate the outer integrals to sufficient precision so as to reduce subtractive cancellation to acceptable levels. The result is a very slow computer program.

Under these conditions, the available computational facilities (a PDP-11/44 minicomputer) restricted the number of performance curves we were able to generate.

5.3.1 The Optimum Threshold Setting.

The optimum clipping threshold η_0 is defined as the level η which minimizes the bit error probability in the absence of jamming. This is accomplished by the first part of the computer programs for calculating the performance in partial-band noise jamming. The thresholds are normalized by the thermal noise density; thus we actually find $\eta_0/2\sigma_N^2$. The optimum thresholds found by the computer programs given in appendices H (for M=2, L=2), I (for M=4, L=2), and J (for M=2 or 4, L=3) are given in Table 5.3-1.

TABLE 5.3-1

OPTIMUM NORMALIZED CLIPPING THRESHOLD $\eta_0/2\sigma_N^2$

L	M: 2 (E_b/N_0 = 13.35247 dB)	4 (E_b/N_0 = 10.60657 dB)	8 (E_b/N_0 = 9.09401 dB)
2	10.20	10.55	10.89
3	7.91	8.15	---

We note that in terms of signal power η_0 = (1.89S, 1.83S, 1.79S) for L=2 and M=(2,4,8); η_0 = (2.19S, 2.13S) for L=3 and M=(2,4). The threshold is almost a function only of S, L, and M.

5.3.2 Probability of Bit Error.

For M=2 and L=2, the computations using the program given in Appendix H were sufficiently rapid to permit obtaining a full set of curves for jamming fractions from γ = 0.001 through γ = 1.0, as shown in Figure 5.3-1.

For M=4 and L=2, the computations were much slower, due to the increased number of jamming events and the need to compute products of the function $G_L(x)$. Therefore, the program in Appendix I was used to search for the optimum value of γ for each value of E_b/N_J. To aid the speed of the search, we used the a priori knowledge that γ_{opt} = 1/N where N is the number of hopping slots when E_b/N_J is very high, and that γ_{opt} increases as E_b/N_J decreases. Thus the computations started at E_b/N_J = 50 dB and decreased (in rather large steps to conserve computer time) to 0 dB. The result is the curve of $P_b(e)$ vs. E_b/N_J in worst-case partial-band noise jamming as shown for M=4 in Figure 5.3-2. For comparison, the envelope of the curves from Figure 5.3-1 is shown in Figure 5.3-2. We see that M=4 FH/RMFSK is about 2 dB better than M=2 FH/RMFSK in strong jamming.

Selected runs for M=8 and L=2 with E_b/N_0=9.09 dB were made in order to examine the dependence of the worst-case jamming performance upon M. These runs yielded the threshold shown in Table 5.3-1 and the following points for γ=0.01: [E_b/N_J, P(e)] = [15, 8.963(-4)], [20, 4.680(-4)], [22.5, 2.604(-4)], [25, 1.286 (-4)], [∞, 3.91(-5)]. From these points, a curve for γ=0.01 was constructed, and the inflection point was taken to be a point on the worst-case jamming curve. This point, estimated as the 22.5 dB point given above, is shown on Figure 5.3-2.

Although the program given in Appendix J contains code to compute $P_b(e)$ as well as $\eta_0/2\sigma_N^2$, excessive run time (nearly 8 hours to obtain just $\eta_0/2\sigma_N^2$), prevented us from allowing it to run to completion to obtain performance curves for the case L=3 hops per symbol.

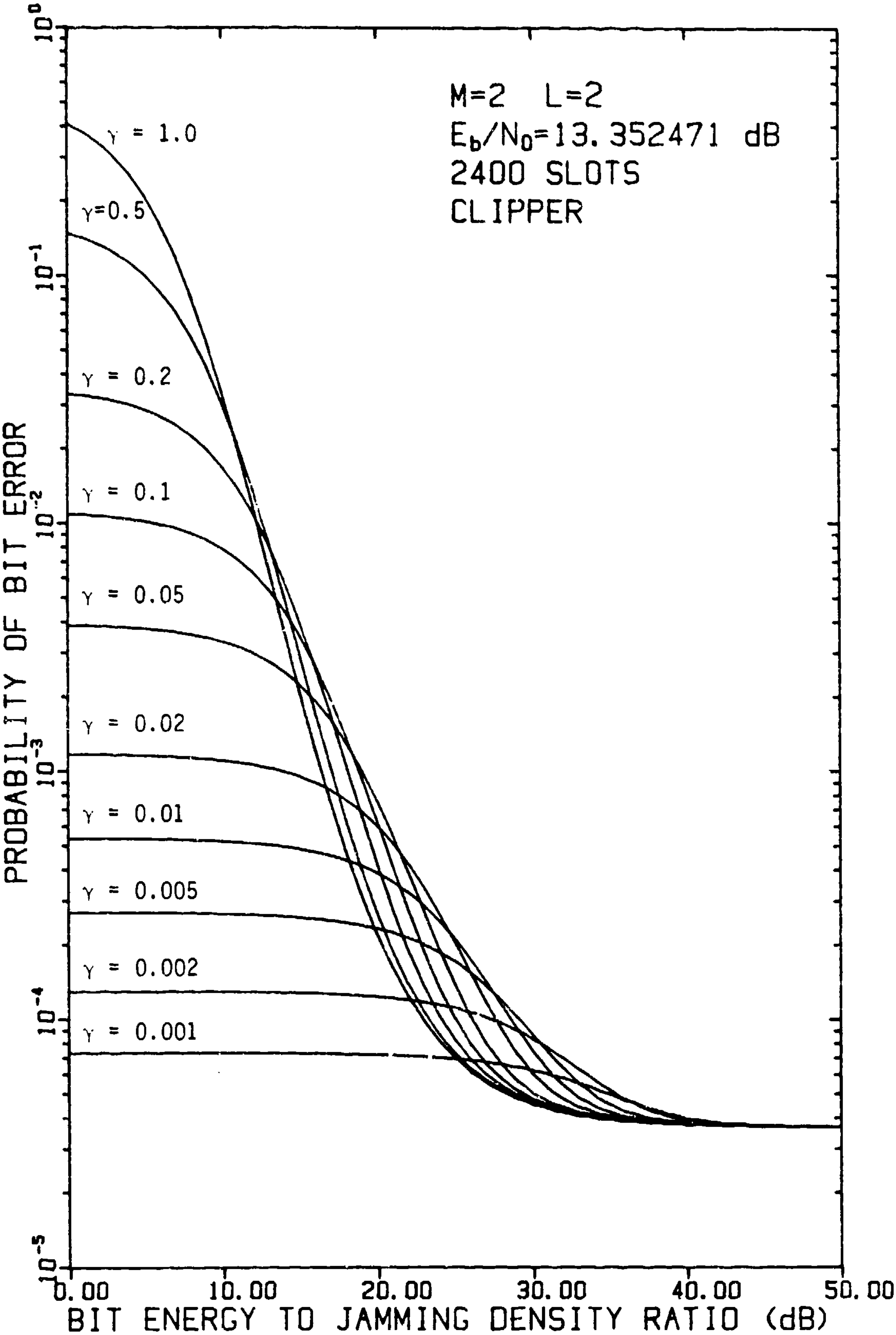


FIGURE 5.3-1 PERFORMANCE OF CLIPPER RECEIVER FOR FH/RMFSK WITH M=2, L=2 HOPS/SYMBOL, AND E_b/N_0 = 13.35247 dB (FOR $P_b(e) = 10^{-5}$ IN THE ABSENCE OF JAMMING WHEN L=1)

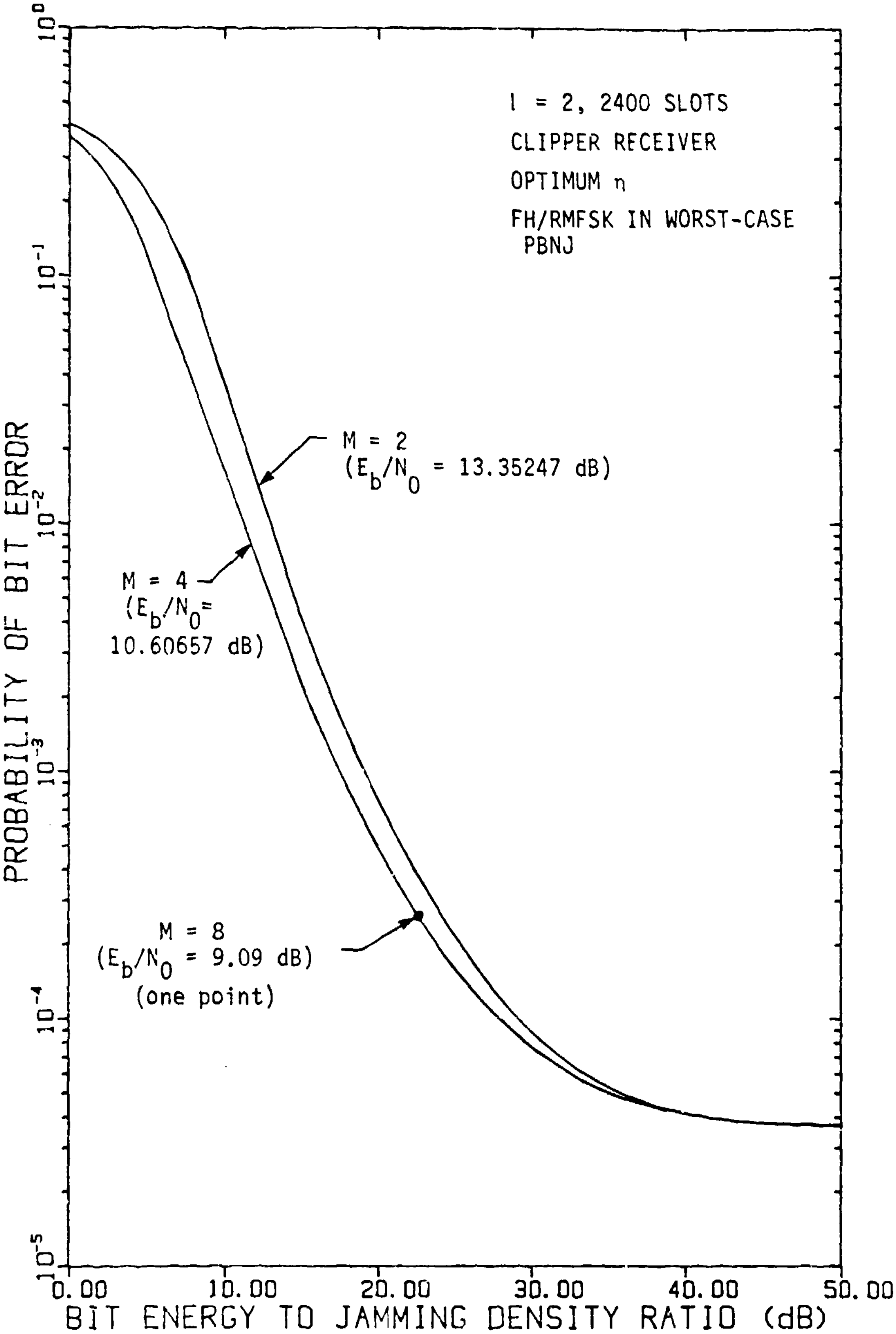


FIGURE 5.3-2 PERFORMANCE OF CLIPPER RECEIVER FOR FH/RMFSK WHEN L=2 HOPS/SYMBOL WITH M AS A PARAMETER AND E_b/N_0 CORRESPONDING TO $P_b(e) = 10^{-5}$ IN THE ABSENCE OF JAMMING (WHEN L=1 HOP/SYMBOL)

6.0 FH/RMFSK PERFORMANCE USING SQUARE-LAW SELF-NORMALIZING RECEIVER

The ECCM weighting schemes which make the (ideal) AGC and clipper soft-decision receivers for FH/MFSK and FH/RMFSK work depend upon *a priori* knowledge of system parameters, or else real-time measurements. (The feasibility of these measurements is discussed in a later section.) It is evident that the clipper strategy, which requires setting an SNR-dependent threshold, would be easier to implement than the AGC receiver, which requires detection of which hops are jammed and knowledge or measurement of thermal noise and jamming noise levels. Meanwhile we have seen that the hard-decision receiver accomplishes a form of ECCM protection, much in the manner of the clipper receiver - the jammed hops are prevented from dominating the decision. If there is sufficient SNR, one might well choose then to employ the hard-decision scheme, since it does not require any *a priori* knowledge or measurements.

In this section we consider soft-decision weighting schemes which are not predicated on using signal or noise parameters. In particular, we find the FH/RMFSK performance of a "self-normalizing" receiver in partial-band noise jamming.

6.1 THE SELF-NORMALIZATION SCHEME

The general FH/RMFSK soft-decision receiver shown in Figure 2.2-1 is rendered what we call the "self-normalizing" (SNORM) receiver by use of the weighting function

$$z_{mk} = f(x_{mk}) = \frac{x_{mk}^2}{x_{1k}^2 + x_{2k}^2 + \dots + x_{Mk}^2} . \qquad (6.1\text{-}1)$$

That is, on each hop (indexed by k, k=1,2,...,L), the squared envelope samples in each channel (m=1,2,..,M) are normalized (divided) by their sum. In this manner, hops which are jammed in one or more MFSK slots can be expected to be weighted less than unjammed hops.

Thus the decision variables for the SNORM receiver are

$$z_m = \sum_{k=1}^{L} z_{mk}$$

$$= \sum_{k=1}^{L} w_k x_{mk}^2 \tag{6.1-2a}$$

where

$$w_k = \left(\sum_{m=1}^{M} x_{mk}^2 \right)^{-1} . \tag{6.1-2b}$$

6.1.1 Single-hop Distribution of Decision Variables.

With the FH/RMFSK hopping scheme, any, none, or all of the channels can be jammed on a particular hop. Using $u_{mk} \equiv x_{mk}^2$ for the square-law envelope samples and

$$a \equiv 1/2\sigma_N^2$$

$$b \equiv 1/2\sigma_T^2, \tag{6.1-3}$$

for a general one-hop jamming event, we can write (assuming the signal is in channel 1)

$$p_{u_1}(\alpha;c_1) = c_1 e^{-c_1\alpha-\rho_1} I_0(2\sqrt{\rho_1 c_1 \alpha}), \; \alpha>0; \tag{6.1-4a}$$

and

$$p_{u_m}(\alpha;c_m) = c_m e^{-c_m\alpha}, \quad m=2,\ldots,M;\ \alpha > 0; \tag{6.1-4b}$$

where

$$c_m = \begin{cases} a, & \text{channel not jammed} \\ b, & \text{channel jammed}, \quad m=1,2,\ldots,M. \end{cases} \tag{6.1-4c}$$

Thus the joint pdf for the square-law envelope detector samples is, conditioned on the jamming,

$$p_{\underline{u}}(\alpha_1,\ldots,\alpha_M|c_1,\ldots,c_M) = c_1c_2\ldots c_M \exp\left\{-\rho_1 - \sum_{m=1}^{M} c_m\alpha_m\right\} I_0(2\sqrt{\rho_1 c_1 \alpha_1}),$$
$$\alpha_m \geq 0. \tag{6.1-5}$$

By a change of variables,

$$u_1 = \xi z_1$$

$$u_1+u_2 = \xi(z_1+z_2)$$

$$\sum_{i=1}^{k} u_i = \xi \sum_{i=1}^{k} z_i \tag{6.1-6}$$

$$u_1+u_2+\cdots+u_M = \xi,$$

we can express the joint pdf of $\{z_1,z_2,\ldots,z_{M-1}\}$ by, using $\underline{c} \equiv (c_1,c_2,\ldots,c_M)$,

$$p_{\underline{z}}(\alpha_1,\alpha_2,\ldots,\alpha_{M-1}|\underline{c}) = \int_0^\infty d\xi\ \xi^{M-1} p_{\underline{u}}\left[\xi\alpha_1,\xi\alpha_2,\ldots,\xi\alpha_{M-1};\xi - \sum_{m=1}^{M-1} \xi\alpha_m\right]$$

$$= c_1 c_2 \cdots c_M \, e^{-\rho_1} \int_0^\infty d\xi \, \xi^{M-1} \exp\left\{ -\xi \left[c_M + \sum_{m=1}^{M-1} (c_m - c_M)\alpha_m \right] \right\} \times I_0\left(2\sqrt{\rho_1 c_1 \xi \alpha_1}\right)$$

$$= \frac{(M-1)! \, e^{-\rho_1} \prod_m c_m}{\left[c_M + \sum_{m=1}^{M-1} (c_m - c_M)\alpha_m \right]^M} \; {}_1F_1\left[M; 1; \frac{c_1 \rho_1 \alpha_1}{c_M + \sum_{m=1}^{M-1} (c_m - c_M)\alpha_m} \right]. \tag{6.1-7}$$

In this development we used equations 6.643.2 and 9.220.2 from [3]; ${}_1F_1(a;b,x)$ is the confluent hypergeometric function.

Note that α_M does not appear in (6.1-7). This occurrence is due to the fact that there are now only M-1 dependent random variables; the value of the Mth channel variable is completely determined by the others. This fact can be made explicit by writing (using all variables)

$$p_{\underline{z}}(\alpha_1, \alpha_2, \ldots, \alpha_M | \underline{c}) = p_{\underline{z}}(\alpha_1, \alpha_2, \ldots, \alpha_{M-1} | \underline{c}) \quad \delta\left(\sum_{m=1}^{M} \alpha_m - 1 \right). \tag{6.1-8}$$

Also, we note that the domains of these variables are interdependent:

$$\begin{aligned} & 0 \le z_m \le 1 \\ & 0 \le z_i + z_j \le 1 \quad \text{(all pairs)} \\ & 0 \le z_i + z_j + z_k \le 1 \quad \text{(all triples)} \\ & \quad \vdots \\ & 0 \le z_1 + z_2 + \ldots + z_{M-1} \le 1 \\ & \sum_{m=1}^{M} z_m \equiv 1. \end{aligned} \tag{6.1-9}$$

This interdependence of finite domains makes analysis and computation difficult, as will be seen below.

6.1.2 Alternate Forms.

By using the identity

$$ {}_1F_1(M;1;x) = e^x \, {}_1F_1(1-M;1;-x) $$
$$ = e^x \mathcal{L}_{M-1}(-x) \;, \tag{6.1-10} $$

where $\mathcal{L}_n(x)$ is the Laguerre polynomial, we realize that the joint pdf given in (6.1-7) has the form of an exponential times an (M-1)-degree polynomial in $x(\underline{\alpha})$, divided by an M-degree polynomial in $y(\underline{\alpha})$:

$$ p_{\underline{z}}(\underline{\alpha}|\underline{c}) = \frac{\text{const}\cdot e^{x(\underline{\alpha})}\mathcal{L}_{M-1}[-x(\underline{\alpha})]}{\left[y(\underline{\alpha})\right]^M} \quad \delta\Big(\sum_m \alpha_m - 1\Big) \tag{6.1-11a} $$

where

$$ x(\underline{\alpha}) = c_1\rho_1\alpha_1/y(\underline{\alpha}) \tag{6.1-11b} $$

$$ y(\alpha) = c_M + \sum_{m=1}^{M-1} (c_m - c_M)\alpha_m \tag{6.1-11c} $$

and

$$ \text{const} = (M-1)!\,e^{-\rho_1} \prod_{m=1}^{M} c_m . \tag{6.1-11d} $$

A somewhat simpler form results from recognizing that [3, equation 8.970.1]

$$ \mathcal{L}_{M-1}(x) = \frac{1}{(M-1)!} e^x \frac{d^{M-1}}{dx^{M-1}} \left[e^{-x} x^{M-1} \right] . \tag{6.1-12} $$

Applying this relation results in

$$p_{\underline{z}}(\underline{\alpha}|\underline{c}) = \frac{\prod_m c_m}{\left[\sum_m c_m\alpha_m\right]^M} e^{-\rho_1} \frac{\partial^{M-1}}{\partial\rho_1^{M-1}} \left\{ \rho_1^{M-1} e^{\rho_1 x(\underline{\alpha})} \right\} \delta(\sum_m \alpha_m - 1). \tag{6.1-13}$$

6.1.2.1 Special case: M=2 (binary).

The various general expressions for the joint pdf reduce to the following ones for M=2:

$$p_{\underline{z}}(\alpha_1, \alpha_2|c_1,c_2) = \frac{c_1c_2e^{-\rho_1}}{(c_1\alpha_1 + c_2\alpha_2)^2} \; {}_1F_1(2;1; \frac{c_1\rho_1\alpha_1}{c_1\alpha_1 + c_2\alpha_2}) \, \delta(\alpha_1 + \alpha_2 - 1) \tag{6.1-14a}$$

$$= \frac{c_1c_2e^{-\rho_1}}{\left[c_2 + (c_1-c_2)\alpha_1\right]^2} \exp\left\{\frac{c_1\rho_1\alpha_1}{c_2 + (c_1-c_2)\alpha_1}\right\} \left[1 + \frac{c_1\rho_1\alpha_1}{c_2 + (c_1-c_2)\alpha_1}\right] \delta(\alpha_1+\alpha_2-1) \tag{6.1-14b}$$

$$= \frac{c_1c_2e^{-\rho_1}}{\left[c_2 + (c_1-c_2)\alpha_1\right]^2} \frac{\partial}{\partial\rho_1} \left[\rho_1 \exp\left\{\frac{c_1\rho_1\alpha_1}{c_2 + (c_1-c_2)\alpha_1}\right\}\right] \delta(\alpha_1 + \alpha_2 - 1) \tag{6.1-14c}$$

with

$$0 \le \alpha_1 \le 1, \quad 0 \le \alpha_2 = 1 - \alpha_1 \le 1. \tag{6.1-14d}$$

6.1.2.2 Special case: no jamming.

For no jamming, $c_1=c_2= \dots =c_M$ (also true for all channels jammed), the general pdf reduces to

$$p_{\underline{z}}(\underline{\alpha}|c_1=c_2= \dots =c_M) = (M-1)!e^{-\rho_1} {}_1F_1(M;1;\rho_1\alpha_1) \; \delta(\sum_m \alpha_m - 1) \tag{6.1-15a}$$

$$= (M-1)! \; e^{-\rho_1+\rho_1\alpha_1} \mathcal{L}_{M-1}(-\rho_1\alpha_1) \; \delta(\sum_m \alpha_m - 1) \tag{6.1-15b}$$

$$= e^{-\rho_1} \frac{\partial^{M-1}}{\partial\rho_1^{M-1}} \left\{ \rho_1^{M-1} e^{\rho_1\alpha_1} \right\} \delta(\sum_m \alpha_m - 1). \tag{6.1-15c}$$

6.1.3 Conditional pdf's for M=2.

We now show the explicit expressions for the decision variable for the binary case; there is only one decision variable $z \equiv z_1$ since $z_2 \equiv 1 - z_1$.

6.1.3.1 Single hop/bit case (L=1).

As far as computation of error probabilities is concerned, the L=1 case of the SNORM receiver pdf's is not needed since the normalization does not affect the outcome of the decision; we know in advance that the result will be the same as if no normalization were employed. However, to go on to the L=2 case, we need the L=1 pdf's.

Using $K \triangleq \sigma_T^2/\sigma_N^2$ as in previous analyses, the pdf's conditioned on the possible jamming events $\underline{\nu} = (\nu_1,\nu_2)$ are as follows:

$$p_1\left[z|\underline{\nu} = (0,0)\right] = e^{-\rho_N} {}_1F_1(2;1;\rho_N z) \quad (6.1\text{-}16a)$$

$$= e^{-\rho_N+\rho_N z}(1+\rho_N z) \; . \quad (6.1\text{-}16b)$$

$$p_1\left[z|\underline{\nu}=(0,1)\right] = \frac{Ke^{-\rho_N}}{\left[1 + (K-1)z\right]^2} \; {}_1F_1\left(2;1; \frac{K\rho_N z}{1 + (K-1)z}\right) \quad (6.1\text{-}17a)$$

$$= \frac{Ke^{-\rho_N}}{\left[1 + (K-1)z\right]^2} \exp\left[\frac{K\rho_N z}{1+(K-1)z}\right]\left(1 + \frac{K\rho_N z}{1 + (K-1)z}\right) \quad (6.1\text{-}17b)$$

$$p_1\left[z|\underline{\nu}=(1,0)\right] = \frac{Ke^{-\rho_T}}{\left[K - (K-1)z\right]^2} \; {}_1F_1\left(2;1; \frac{\rho_T z}{K - (K-1)z}\right) \quad (6.1\text{-}18a)$$

$$= \frac{Ke^{-\rho_T}}{\left[K - (K-1)z\right]^2} \exp\left[\frac{\rho_T z}{K - (K-1)z}\right]\left(1 + \frac{\rho_T z}{K - (K-1)z}\right) \quad (6.1\text{-}18b)$$

$$p_1\left[z|\underline{\nu} = (1,1)\right] = e^{-\rho_T} {}_1F_1(2;1;\rho_T z) \quad (6.1\text{-}19a)$$

$$= e^{-\rho_T+\rho_T z}(1 + \rho_T z) \; . \quad (6.1\text{-}19b)$$

For all of these expressions, the domain of z is $0 \leq z \leq 1$. Note that when there is no signal ($\rho_N = \rho_T = 0$), the variable z is uniformly distributed when both channels have the same noise power.

6.1.3.2 Two hops/bit case (L=2).

To obtain the pdf for L=2, it is necessary to convolve the expressions (6.1-16) to (6.1-19) with each other for the jamming events $\underline{\ell} = \underline{\nu}_1 + \underline{\nu}_2$. The general form of the convolution is

$$p_2(z|\underline{\ell} = \underline{\nu}_1 + \underline{\nu}_2) = \int_{\max(0,z-1)}^{\min(1,z)} dv\ p_1(z-v|\underline{\nu}_1)p_1(v|\underline{\nu}_2),\ 0 \leq z \leq 2. \qquad (6.1\text{-}20)$$

There are ten distinguishable jamming events, two for $\underline{\ell}=(1,1)$ and one each for other $\underline{\ell}$. For three of these ten cases, the convolution shown in (6.1-20) can be performed analytically without too much difficulty; the cases are the ones in which both channels are jammed or not jammed on a given hop: $\underline{\ell}=(0,0)$, $(1,1)^*$, $(2,2)$. For these cases, the pdf for one hop can be written

$$p_1(z) = e^{-\rho_1} \frac{\partial}{\partial \rho_1} \rho_1 e^{\rho_1 z} \quad , 0 \leq z \leq 1 \qquad (6.1\text{-}21)$$

The convolution then takes the form

$$p_2(z|\rho_1,\rho_2) = e^{-\rho_1-\rho_2} \frac{\partial^2}{\partial\rho_1\partial\rho_2} \rho_1\rho_2 \int_{\max(0,z-1)}^{\min(1,z)} dv\ e^{\rho_1(z-v)+\rho_2 v}$$

$$= e^{-\rho_1-\rho_2} \frac{\partial^2}{\partial\rho_1\partial\rho_2} \frac{\rho_2\rho_1}{\rho_2\ \rho_1} \cdot \begin{cases} e^{\rho_2 z} - e^{\rho_1 z} & , 0 \leq z \leq 1 \\ e^{\rho_2+\rho_1(z-1)} - e^{\rho_1+\rho_2(z-1)} & , 1 \leq z \leq 2. \end{cases} \qquad (6.1\text{-}22)$$

---

*i.e., the case of $\underline{\ell} = (1,1)$ where $\underline{\nu}_1 = (1,1)$ and $\underline{\nu}_2 = (0,0)$ or vice versa.

Carrying out the partial differentiations results in

$$p_2(z|\rho_1,\rho_2) = \frac{e^{-\rho_1-\rho_2}}{(\rho_1-\rho_2)^3}\left\{e^{\rho_1 z}\left[-2\rho_1\rho_2 + \rho_1^2(\rho_1-\rho_2)z\right]\right. \tag{6.1-23a}$$

$$\left. + e^{\rho_2 z}\left[2\rho_1\rho_2 + \rho_2^2(\rho_1-\rho_2)z\right]\right\},\ \rho_1\neq\rho_2, 0\leq z\leq 1;$$

$$= e^{-2\rho+\rho z}\, z(1 + \rho z + \rho^2 z^2/6),\ \rho_1=\rho_2=\rho,\ 0<z<1; \tag{6.1-23b}$$

$$= \frac{1}{(\rho_1-\rho_2)^3}\left\{e^{\rho_2(z-2)}\left[-2\rho_1\rho_2 + \rho_1^2(\rho_1-\rho_2) + \right.\right.$$

$$\left. \rho_2(\rho_1-\rho_2)(\rho_1^2-\rho_1\rho_2-\rho_2)(z-1)\right]$$

$$+ e^{\rho_1(z-2)}\left[2\rho_1\rho_2 + \rho_2^2(\rho_1-\rho_2)\right.$$

$$\left.\left. + \rho_1(\rho_1-\rho_2)(\rho_2^2-\rho_1\rho_2-\rho_1)(z-1)\right]\right\}$$

$$1 < z \leq 2,\ \rho_1\neq\rho_2 \tag{6.1-23c}$$

$$= e^{-2\rho+\rho z}\left[1 + \rho + \rho^2/6 - (1-\rho^2/2)(z-1) - \rho(1+\rho/2)(z-1)^2\right.$$

$$\left. -(\rho^2/6)(z-1)^3\right],\ \rho_1=\rho_2=\rho,\ 1<z\leq 2. \tag{6.1-23d}$$

This expression is applied to the pertinent jamming events using the following table:

ℓ_1	ρ_1	ρ_2
0	ρ_N	ρ_N
1	ρ_N	ρ_T
2	ρ_T	ρ_T.

(6.1-24)

The other seven cases must be handled by numerical convolution. (We have found an analytical expression, but it is no easier to compute than the convolutions.)

6.1.4 Conditional pdf's for M=4.

The system analysis for M>2 becomes very difficult, as we now demonstrate for M=4.

6.1.4.1 Single hop/symbol case (L=1).

For M=4 and L=1 there are sixteen possible jamming events, described by the vector $\underline{\nu} = (\nu_1,\nu_2,\nu_3,\nu_4)$, where $\nu_m = 1$ if the mth symbol frequency slot is jammed, and $\nu_m = 0$ if not. These events give rise to the conditional pdf

$$p(\underline{z}|\underline{\nu}) = \frac{6\mu_1}{\left[y(\underline{z})\right]^4}\; e^{-\rho_1+\mu_2\rho_1 z_1/y(z)}\; \mathcal{L}_3\left[-\mu_2\rho_1 z_1/y(\underline{z})\right] , \qquad (6.1\text{-}25a)$$

where

$$\mathcal{L}_3(-u) = 1 + 3u + \frac{3}{2}u^2 + \frac{1}{6}u^3 \qquad (6.1\text{-}25b)$$

and the parameters μ_1,μ_2, and ρ_1 and the polynomials $y(\underline{z})$ are listed in Table 6.1-1. Since $z_4 \equiv 1-z_1-z_2-z_3$, it does not appear in the pdf. We note that z_1 always appears in the conditional pdf, while z_2 and z_3 may or may not appear.

It is understood that the domain of values for the variables is $(z_1,z_2,z_3) \in \Omega_{4,1}$, where $\Omega_{4,1}$ is the volume

$$\Omega_{4,1}: \begin{cases} 0 \le z_i \le 1 \ , \ i = 1,2,3; \\ 0 \le z_i + z_j \le 1 \ , \ \text{all pairs}; \\ 0 \le z_1 + z_2 + z_3 \le 1. \end{cases} \qquad (6.1\text{-}26)$$

TABLE 6.1-1

L=1 PROBABILITY DENSITIES FOR M=4

	$\nu_1\nu_2\nu_3\nu_4$	ρ_1	μ_1	μ_2	$y(z)$
A	0 0 0 0		1	1	1
B	0 0 0 1		K^3		$1 + (K-1)\ (z_1+z_2+z_3)$
C	0 0 1 0		K^3		$K - (K-1)z_3$
D	0 0 1 1	ρ_N	K^2		$1 + (K-1)\ (z_1+z_2)$
E	0 1 0 0		K^3	K	$K - (K-1)z_2$
F	0 1 0 1		K^2		$1 + (K-1)\ (z_1+z_3)$
G	0 1 1 0		K^2		$K - (K-1)\ (z_2+z_3)$
H	0 1 1 1		K		$1 + (K-1)z_1$
I	1 0 0 0		K^3		$K - (K-1)z_1$
J	1 0 0 1		K^2		$1 + (K-1)\ (z_2+z_3)$
K	1 0 1 0		K^2		$K - (K-1)\ (z_1+z_3)$
L	1 0 1 1		K		$1 + (K-1)z_2$
M	1 1 0 0	ρ_T	K^2	1	$K - (K-1)\ (z_1+z_2)$
N	1 1 0 1		K		$1 + (K-1)z_3$
O	1 1 1 0		K		$K - (K-1)\ (z_1+z_2+z_3)$
P	1 1 1 1		1		1

Form:

$$p_1(\underline{z}|\underline{\nu}) = \frac{6\mu_1}{[y(\underline{z})]^4}\ e^{-\rho_1+\mu_2\rho_1 z_1/y(\underline{z})}\ \mathcal{L}_3(-\mu_2\rho_1 z_1/y(\underline{z}))$$

$$\mathcal{L}_3\ (-u) = 1 + 3u + \frac{3}{2}u^2 + \frac{1}{6}u^3$$

$$\underline{z} = (z_1, z_2, z_3) \ \epsilon\ \Omega_{4,1}$$

$$z_4 = 1 - z_1 - z_2 - z_3$$

The domain $\Omega_{4,1}$ may also be described as the volume included by the planes $z_1=0$, $z_2=0$, $z_3=0$, and $z_1+z_2+z_3=1$, as illustrated by Figure 6.1-1. It is obvious from the mutual constraints among the variables that they are statistically dependent.

6.1.4.2 Two hop/symbol case (L=2) .

Since the four SNORM variables are dependent, we cannot analyze the M=4, L=2 case by finding the two-sample distributions of the separate channels as we did for other receivers. The convolution must be done in three dimensions (M-1 dimensions for the general case). The concept for doing this is unusual, but can be visualized. Figure 6.1-2 illustrates the fact that multi-dimensional convolution of two pdf's involves integration over the volume which is the intersection of the domains of the pdf's. In Figure 6.1-2(b), the simple case when the point (z_1,z_2,z_3) lies inside the domain of $p_1(z)$ is shown; this yields a rectangular-sided volume. If (z_1,z_2,z_3) lies outside the domain of $p_1(z)$, the intersection is much more complicated.

By careful study we have determined that the pdf for the SNORM receiver's decision variables for M=4 and L=2 has the general form

$$p_2(\underline{z}|\underline{\ell} = \underline{\nu}_1+\underline{\nu}_2) = \int_{A_1}^{B_1} dv_1 \int_{A_2}^{B_2} dv_2 \int_{A_3}^{B_3} dv_3 \; p_1(\underline{v}|\underline{\nu}_1)p_1(\underline{z}-\underline{v}|\underline{\nu}_2), \qquad (6.1\text{-}27a)$$

where

$$\begin{aligned} A_1 &= \max\,(0,z_1-1) \\ B_1 &= \min\,(1,z_1) \\ A_2 &= \max\,(0,z_1+z_2-v_1-1) \\ B_2 &= \min\,(1-v_1,z_2) \\ A_3 &= \max\,(0,z_1+z_2+z_3-v_1-v_2-1) \\ B_3 &= \min\,(1-v_1-v_2,z_3). \end{aligned} \qquad (6.1\text{-}27b)$$

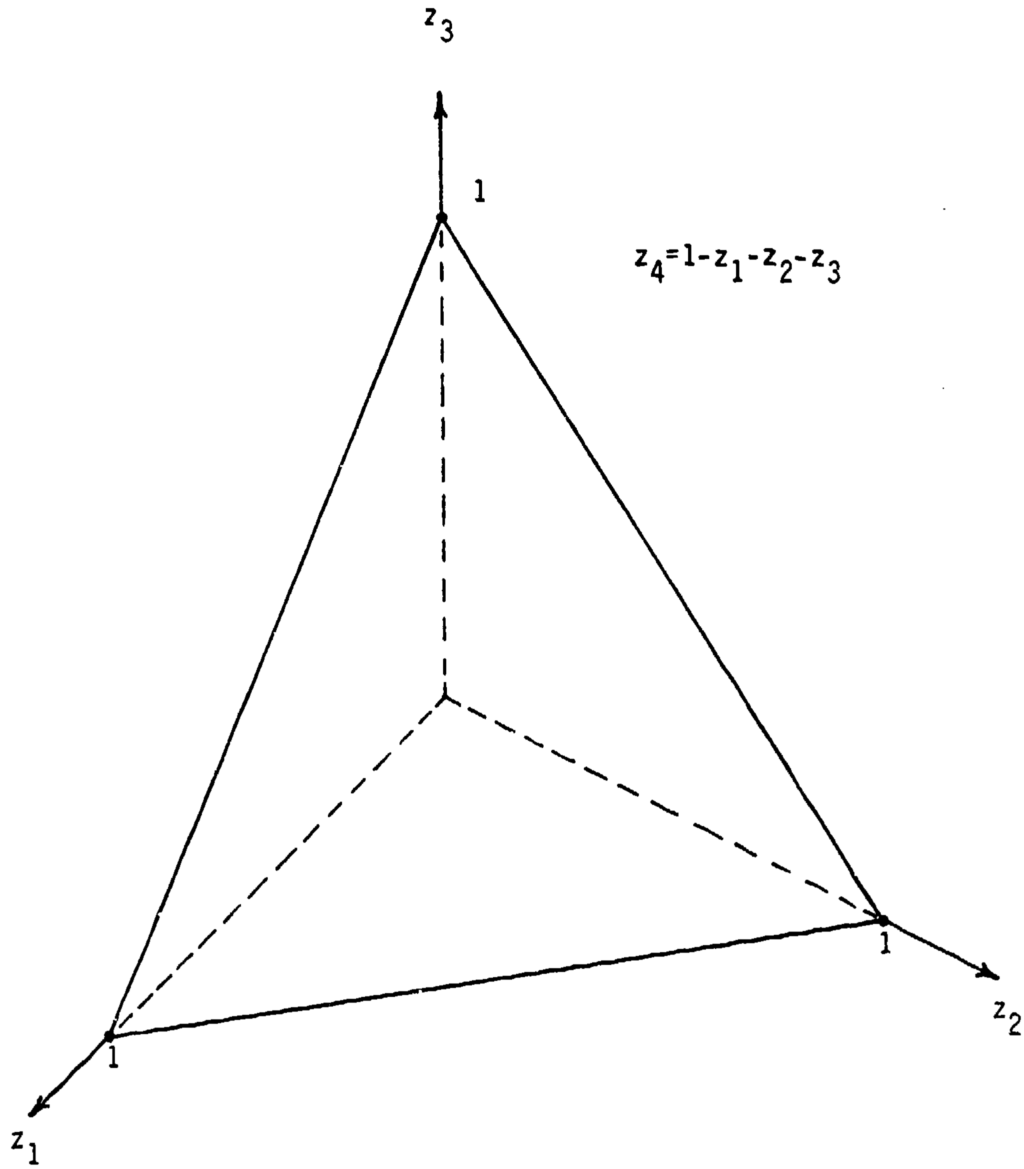
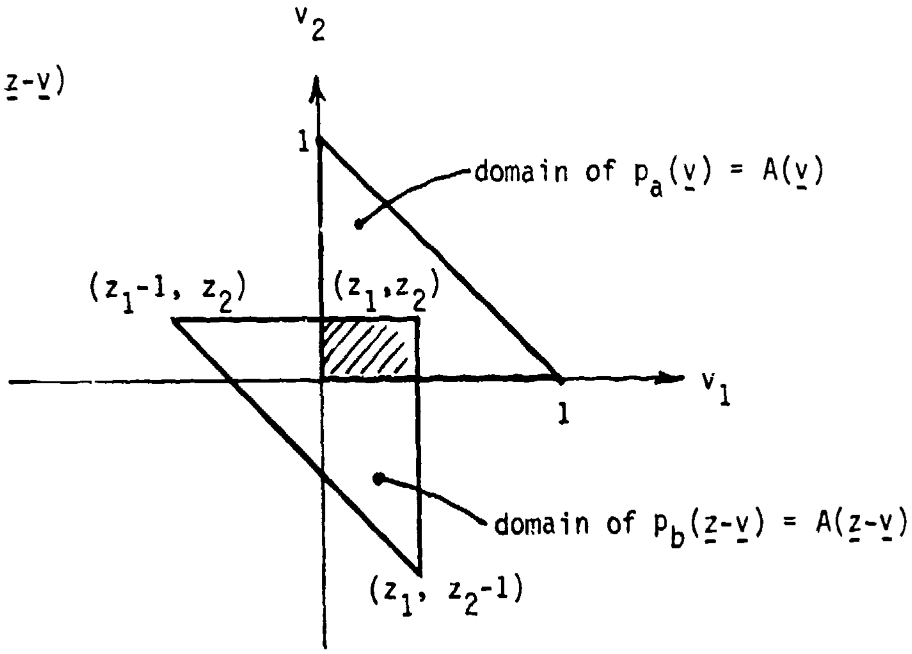
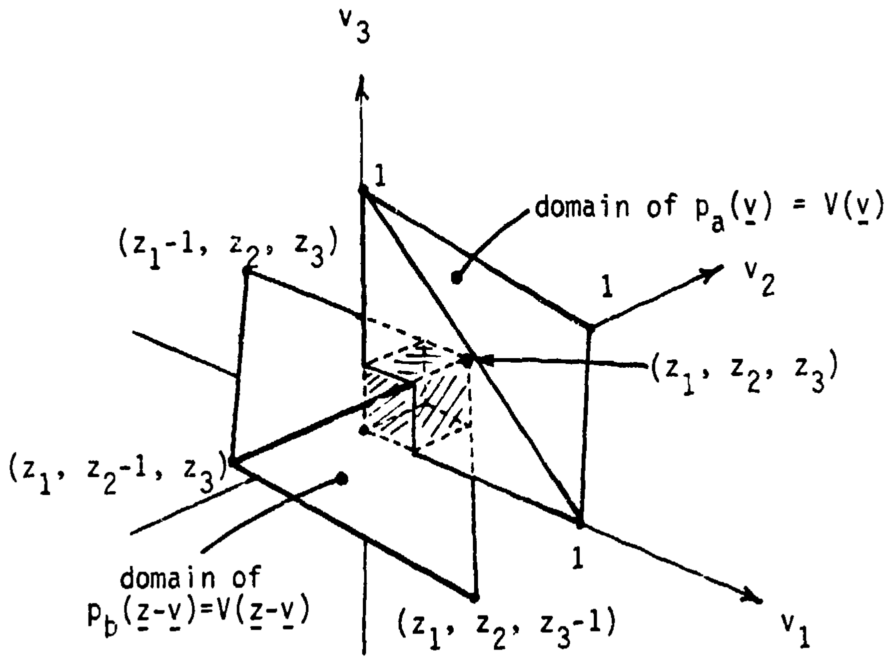


FIGURE 6.1-1 REGION CONTAINING DECISION VARIABLES

$$\iint\limits_{A(\underline{v}) \cap A(\underline{z}-\underline{v})} d\underline{v}\; p_a(\underline{v}) p_b(\underline{z}-\underline{v})$$



(a) two-dimensional convolution



$$\iiint\limits_{V(\underline{v}) \cap V(\underline{z}-\underline{v})} d\underline{v}\; p_a(\underline{v}) p_b(\underline{z}-\underline{v})$$

(b) three-dimensional convolution

FIGURE 6.1-2 TWO- AND THREE-DIMENSIONAL CONVOLUTIONS

Now, since there are sixteen cases of $p_1(\underline{z}|\underline{\nu})$ for L=1, there are $(16)^2 = 256$ cases for L=2. However since the numbering of symbol channels is arbitrary, there can be considered to be fewer, distinguishable jamming events. These are fully enumerated in Section 6.2. What we wish to note here is that if neither of the densities in (6.1-27) contains v_3, the integral can be simplified to

$$p_2(\underline{z}|\underline{\ell}) = \int_{A_1}^{B_1} dv_1 \int_{A_2}^{B_2} dv_2 \; p_1(\underline{v}|\nu_1)p_1(\underline{z}-\underline{v}|\underline{\nu}_2) \; (B_3-A_3) \tag{6.1-28a}$$

where

$$B_3 - A_3 = \min\,(1-v_1-v_2,v_3) - \max(0,z_1+z_2+z_3-v_1-v_2-1)$$

$$= \frac{1}{2}\left\{2-z_1-z_2-|1-v_1-v_2-z_3| - |z_1+z_2+z_3-v_1-v_2-1|\right\}. \tag{6.1-28b}$$

If neither pdf in (6.1-27) contains v_2 or v_3, the integral can be further simplified to

$$p_2(\underline{z}|\underline{\ell}) = \int_{A_1}^{B_1} dv_1 \; p_1(v_1|\underline{\nu}_1)p_1\;(z_1-v_1|\underline{\nu}_2) \int_{A_2}^{B_2} dv_2 \; (B_3-A_3), \tag{6.1-29}$$

where

$$\int_{A_2}^{B_2} dv_2 \; (B_3-A_3) = \frac{1}{2}\,(2-z_1-z_2)\;(B_2-A_2)$$

$$- \frac{1}{4}\left\{(B_2+v_1+z_3-1)|B_2+v_1+z_3-1|\right.$$

$$-(A_2+v_1+z_3-1)|A_2+v_1+z_3-1|$$

$$+ (B_2+v_1+1-z_1-z_2-z_3)|B_2+v_1+1-z_1-z_2-z_3|$$

$$\left.- (A_2+v_1+1-z_1-z_2-z_3)|A_2+v_1+1-z_1-z_2-z_3|\right\}, \tag{6.1-30}$$

since

$$\int_A^B dx\ |x-a| = \int_A^B dx\ |a-x| = \frac{1}{2}(B-a)|B-a| - \frac{1}{2}(A-a)|A-a|. \qquad (6.1\text{-}3$$

Now, if v_3 is in the integrand of (6.1-27) but v_2 is not, we simply "switch labels" on v_2 and v_3 to get (6.1-28) with v_3 replacing v_2.

6.2 JAMMING EVENTS AND ERROR PROBABILITY FOR L=2

We now extend the conditional distribution analysis in the last section to obtain the BER for the FH/RMFSK SNORM receiver under partial-band noise jamming. Since the jammed error for L=1 is the same for other receivers, we proceed to the case of L=2.

6.2.1 Jamming Events and Probabilities for M=2.

For L=2 and M=2 there are $2^{ML}=16$ elementary jamming events. As mentioned previously, for the SNORM receiver, only ten of these events are distinguishable in terms of jamming effects. These are listed in Table 6.2-1, along with the single-hop events which produce them and the probabilities of the L=2 events.

The error event, assuming the signal is in channel 1, is $z_1 < z_2 \equiv 2-z_1$. Thus

$$P_b(e;\gamma) = \Pr\{z_1<1\} = \sum_{\underline{\ell}} \Pr\{\underline{\ell}\} \Pr\{z_1<1|\underline{\ell}\}. \qquad (6.2\text{-}1)$$

The conditional error probabilities in (6.2-1) are calculated by integrating the pdf's shown previously in equation (6.1-23) or the convolution

TABLE 6.2-1

JAMMING EVENTS AND PROBABILITIES FOR M=2, L=2

$\underline{\ell}$	$\underline{\nu_1}$	$\underline{\nu_2}$	# events	Probability
0,0	0,0	0,0	1	$\pi_0{}^2$
0,1	0,1	0,0	2	$2\pi_0\pi_1$
0,2	0.1	0.1	1	$\pi_1{}^2$
1,0	1,0	0,0	2	$2\pi_0\pi_1$
1,1	0,0	1,1	2	$2\pi_0\pi_2$
	0,1	1,0	2	$2\pi_1{}^2$
1,2	0,1	1,1	2	$2\pi_1\pi_2$
2,0	1,0	1,0	1	$\pi_1{}^2$
2,1	1,0	1,1	2	$2\pi_1\pi_2$
2,2	1,1	1,1	1	$\pi_2{}^2$
		Totals:	16	1

$$\pi_0 = \frac{\binom{N-2}{q}}{\binom{N}{q}} = \frac{(N-q)(N-q-1)}{N\ (N-1)}\ ,\ \pi_1 = \frac{\binom{N-2}{q-1}}{\binom{N}{q}} = \frac{q}{N}\frac{(N-q)}{(N-1)}\ ,\ \pi_2 = \frac{\binom{N-2}{q-2}}{\binom{N}{q}} = \frac{q}{N}\frac{(q-1)}{(N-1)}$$

$$p_2(z\,|\,\underline{\ell}=\underline{\nu_2}+\underline{\nu_2}) = \int_{\max(0,z-1)}^{\min(1,z)} dv\ p_1(v\,|\,\underline{\nu_1})\ p_1(z-v\,|\,\underline{\nu_2})$$

of the pdf's in equations (6.1-16) to (6.1-19). The result is

$$
\begin{aligned}
P_b(e;\gamma) = {} & \pi_0^2 \cdot \frac{1}{2} e^{-\rho_N} (1+\rho_N/3) \\
& + 2\pi_0\pi_1 \int_0^1 dv\ (1-v) e^{-\rho_N v} p_1(v|0,1) \\
& + \pi_1^2 \int_0^1 dv\ p_1(v|0,1) \cdot \frac{K(1-v)}{v+K(1-v)} \exp\left\{ \frac{-\rho_N v}{v+K(1-v)} \right\} \\
& + 2\pi_0\pi_1 \int_0^1 dv\ (1-v) e^{-\rho_N v}\ p_1(v|1,0) \\
& + 2\pi_0\pi_2 \frac{1}{(\rho_N-\rho_T)^3} \left\{ e^{-\rho_T} \left[\rho_N(\rho_N-\rho_T) - (\rho_N+\rho_T) \right] \right. \\
& \qquad \left. + e^{-\rho_N} \left[\rho_T(\rho_N-\rho_T) + (\rho_N+\rho_T) \right] \right\} \\
& + 2\pi_1^2 \int_0^1 dv\ p_1(v|0,1) \cdot \frac{1-v}{Kv+1-v} \exp\left\{ \frac{-K\rho_T v}{Kv+1-v} \right\} \\
& + 2\pi_1\pi_2 \int_0^1 dv\ (1-v) e^{-\rho_T v} p_1(v|0,1) \\
& + \pi_1^2 \int_0^1 dv\ p_1\ (v|1,0) \cdot \frac{1-v}{Kv+1-v}\ \exp\left\{ \frac{-K\rho_T v}{Kv+1-v} \right\} \\
& + 2\pi_1\pi_2 \int_0^1 dv\ (1-v) e^{-\rho_T v}\ p_1(v|1,0) \\
& + \pi_2^2 \cdot \frac{1}{2} e^{-\rho_T} (1+\rho_T/3). \qquad (6.2\text{-}2)
\end{aligned}
$$

In this expression we have used the fact that

$$
\begin{aligned}
&\int_0^1 dz \int_{\max(0,z-1)}^{\min(1,z)} dv \;\; p_1(v|\underline{\nu}_1) \;\; p_1(z-v|\underline{\nu}_2) \\
&= \int_0^1 dz \int_0^z dv \;\; p_1(v|\underline{\nu}_1) \;\; p_1(z-v|\underline{\nu}_2) \\
&= \int_0^1 dv \;\; p_1(v|\underline{\nu}_1) \int_0^{1-v} dz \;\; p_1(z|\underline{\nu}_2) \; .
\end{aligned}
\tag{6.2-3}
$$

Also, the parameters K, ρ_N, and ρ_T are

$$
K = \frac{\sigma_T^2}{\sigma_N^2} \; , \; \rho_N = \frac{1}{2} \cdot \frac{E_b}{N_0} \, , \;\; \rho_T = \frac{1}{2} \cdot \frac{E_b}{N_T} \; . \tag{6.2-4}
$$

6.2.2 Jamming Events and Probabilities for M=4.

For L=2 and M=4, there are 256 elementary jamming events, which can be represented by 47 distinguishable events. These are listed in Table 6.2-2, along with their probabilities of occurrence. The joint pdf of the decision variables, given the representative jamming event shown in the table, is the convolution (6.1-27) with the single-hop pdf's selected from Table 6.1-1 as indicated.

The error event for M=4 and L=2 is the complement of the condition for a correct symbol decision, so that the conditional error probability is

$$
\begin{aligned}
P_s(e|\underline{\ell}) &= 1 - \Pr\{z_1 > z_2, \, z_1 > z_3, \, z_1 > z_4 = 2 - z_1 - z_2 - z_3 | \underline{\ell}\} \\
&= 1 - \Pr\{z_1 > z_2, \, z_1 > z_3, \, 2z_1 > 2 - z_2 - z_3 | \underline{\ell}\} \qquad (6.2\text{-}5a) \\
&\equiv 1 - P_s(C|\underline{\ell}). \qquad (6.2\text{-}5b)
\end{aligned}
$$

TABLE 6.2-2

JAMMING EVENTS AND PROBABILITIES FOR M=4, L=2

$\underline{\ell}$	cases*	# events	prob.
0 0 0 0	A+A	1	$\pi_0{}^2$
0 1 0 0	A+E	6	$6\pi_0\pi_1$
0 1 1 0	A+G	6	$6\pi_0\pi_2$
	C+E	6	$6\pi_1{}^2$
0 1 1 1	A+H	2	$2\pi_0\pi_3$
	C+F	6	$6\pi_1\pi_2$
0 2 0 0	E+E	3	$3\pi_1{}^2$
0 2 1 0	E+G	12	$12\pi_1\pi_2$
0 2 1 1	E+H	6	$6\pi_1\pi_3$
	F+G	6	$6\pi_2{}^2$
0 2 2 0	G+G	3	$3\pi_2{}^2$
0 2 2 1	G+H	6	$6\pi_2\pi_3$
0 2 2 0	H+H	1	$\pi_3{}^2$
1 0 0 0	A+I	2	$2\pi_0\pi_1$
1 1 0 0	A+M	6	$6\pi_0\pi_2$
	E+I	6	$6\pi_1{}^2$
1 1 1 0	A+N	6	$6\pi_0\pi_3$
	G+I	6	$6\pi_1\pi_2$
	C+M	12	$12\pi_1\pi_2$
1 1 1 1	A+P	2	$2\pi_0\pi_4$
	H+I	2	$2\pi_1\pi_3$
	C+N	6	$6\pi_1\pi_3$
	F+K	6	$6\pi_2{}^2$
1 2 0 0	E+M	6	$6\pi_1\pi_2$
1 2 1 0	E+N	12	$12\pi_1\pi_3$
	G+M	12	$12\pi_2{}^2$
1 2 1 1	G+N	12	$12\pi_{.}\pi_3$
	E+P	6	$6\pi_1\pi_4$
	H+M	6	$6\pi_2\pi_3$

$\underline{\ell}$	cases*	# events	prob.
1 2 2 0	F+N	6	$6\pi_2\pi_3$
1 2 2 1	H+N	6	$6\pi_3{}^2$
	G+P	6	$6\pi_2\pi_4$
1 2 2 2	H+P	2	$2\pi_3\pi_4$
2 0 0 0	I+I	1	$\pi_1{}^2$
2 1 0 0	I+M	6	$6\pi_1\pi_2$
2 1 1 0	I+N	6	$6\pi_1\pi_3$
	K+M	6	$6\pi_2{}^2$
2 1 1 1	I+P	2	$2\pi_1\pi_4$
	K+N	6	$6\pi_2\pi_3$
2 2 0 0	M+M	3	$3\pi_2{}^2$
2 2 1 0	M+N	12	$12\pi_2\pi_3$
2 2 1 1	M+P	6	$6\pi_2\pi_4$
	L+N	6	$6\pi_3{}^2$
2 2 2 0	N+N	3	$3\pi_3{}^2$
2 2 2 1	N+P	6	$6\pi_3\pi_4$
2 2 2 2	P+P	1	$\pi_4{}^2$
	Totals:	256	1

*cases (A-P): See Table 6.1-1

$$\pi_0 = \frac{(N-q)(N-q-1)(N-q-2)(N-q-3)}{N\ (N-1)(N-2)(N-3)}$$

$$\pi_1 = \frac{q\ (N-q)(N-q-1)(N-q-2)}{N(N-1)(N-2(N-3)}$$

$$\pi_2 = \frac{q(q-1)(N-q)(N-q-1)}{N\ (N-1)(N-2)(N-3)}$$

$$\pi_3 = \frac{q\ (q-1)(q-2)(N-q)}{N(N-1)(N-2)(N-3)}$$

$$\pi_4 = \frac{q\ (q-1)(q-2)(q-3)}{N(N-1)(N-2)(N-3)}$$

From this expression we observe that the probability of a correct symbol decision is obtained from the joint pdf of (z_1,z_2,z_3) by integrating it over the volume Ω_c implied in (6.2-5):

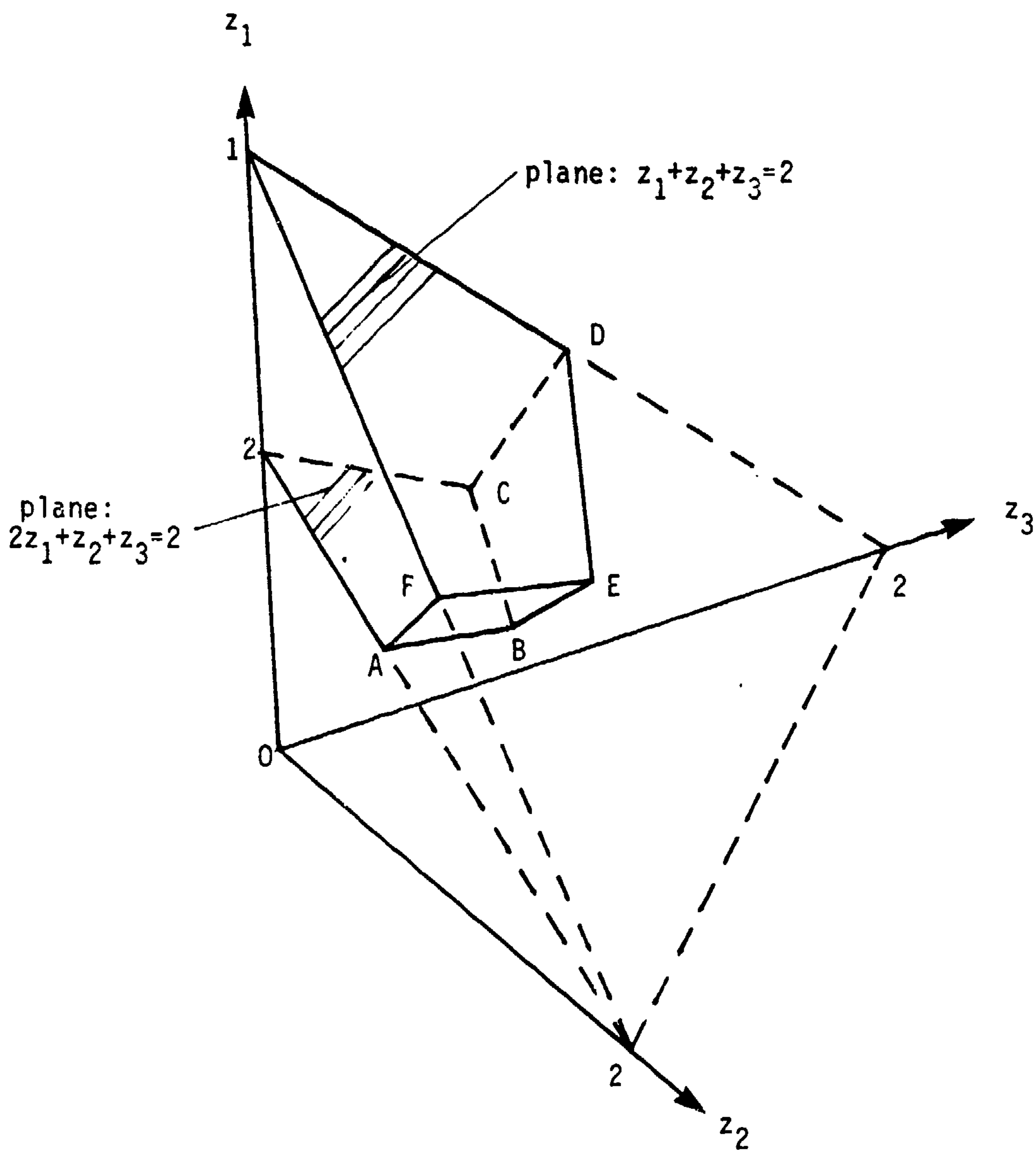
$$P_s(C|\underline{\ell}) = \iiint_{\Omega_c} dz_1 dz_2 dz_3 \;\; p_2(\underline{z} \mid \underline{\ell}). \tag{6.2-6}$$

As illustrated in Figure 6.2-1, the volume Ω_c may be described as that enclosed by the planes $z_2=0$, $z_3=0$, $2z_1+z_2+z_3=2$, $z_1+z_2+z_3=2$, $z_1=z_2$, and $z_1=z_3$. Thus

$$P_s(C|\underline{\ell}) = \int_{A_4}^{B_4} dz_1 \int_{A_5}^{B_5} dz_2 \int_{A_6}^{B_6} dz_3 \; p_2(\underline{z} \mid \underline{\ell}), \tag{6.2-7a}$$

where

$$\begin{aligned} A_4 &= 1/2,\; B_4 = 2 \\ A_5 &= \max(0, 2-3z_1) \\ B_5 &= \min(z_1,\, 2-z_1) \\ A_6 &= \max(0, 2-2z_1-z_2) \\ B_6 &= \min(z_1, 2-z_1-z_2). \end{aligned} \tag{6.2-7b}$$



points: (z_1, z_2, z_3)

A: (2/3, 2/3, 0)   D: (1, 0, 1)

B: (1/2, 1/2, 1/2) E: (2/3, 2/3, 2/3)

C: (2/3, 0, 2/3)   F: (1, 1, 0)

FIGURE 6.2-1 VOLUME OF INTEGRATION FOR CORRECT SYMBOL DECISION, M=4, L=2

6.3 AN ALTERNATE APPROACH FOR M=2 AND L=3 HOPS/SYMBOL

In order to obtain a more computationally tractable form for the performance of the self-normalizing receiver in partial-band noise jamming, we may proceed as follows. The probability of a symbol error is

$$\begin{aligned} \Pr(e) &= \Pr\{z<3/2\} \\ &= E_{\underline{\nu}}\{\Pr\{z<3/2\,|\,\underline{\nu}\}\} \\ &= E_{\underline{\nu}}\left\{\int_0^{3/2} p_3(\zeta\,|\,\underline{\nu})\,d\zeta\right\} \end{aligned} \tag{6.3-1}$$

where $p_3(\zeta|\underline{\nu})$ is the probability density function conditioned on jamming event $\underline{\nu}$.

If we interchange the order of integration with respect to ζ and expectation with respect to $\underline{\nu}$ in (6.3-1), we obtain

$$\Pr(e) = \int_0^{3/2} E_{\underline{\nu}}\{p_3(\zeta|\underline{\nu})\}d\zeta. \tag{6.3-2}$$

The expectation in (6.3-2) may be written as

$$\begin{aligned} p_3(\zeta) &\triangleq E_{\underline{\nu}}\{p_3(\zeta|\underline{\nu})\} \\ &= E_{\underline{\nu}}\{p_1(\zeta|\underline{\nu})*p_2(\zeta|\underline{\nu})\} \\ &= p_1(\zeta)*p_2(\zeta) \end{aligned} \tag{6.3-3}$$

where the operator * denotes convolution and

$$p_1(\zeta) \triangleq E_{\underline{\nu}_1}\{p_1(\zeta|\underline{\nu}_1)\} \tag{6.3-4}$$

with $\underline{\nu}_1$ being the first column of the event matrix $\underline{\nu}$ and

$$p_2(\zeta) \triangleq E_{\underline{\nu}_2}\{p_2(\zeta|\underline{\nu}_2)\} \tag{6.3-5}$$

with $\underline{\nu}_2$ being the last two columns of the matrix $\underline{\nu}$. The expectation in (6.3-4) is given by

$$p_1(\zeta) = \pi_0 p_{00}(\zeta) + \pi_1 p_{01}(\zeta) + \pi_1 p_{10}(\zeta) + \pi_2 p_{11}(\zeta) \tag{6.3-6}$$

where the $p_{ij}(\zeta) \triangleq p_1[\zeta|\underline{\nu} = (i,j)]$ are given by (6.1-16)-(6.1-19) and the event probabilities π_0, π_1, and π_2 are given by Table 6.2-1.

The density $p_2(\zeta)$ from (6.3-5) contains 10 terms, as discussed in Section 6.1.3.2. Analytical results for three of these cases are given by (6.1-23) and (6.1-24).

By performing the convolution (6.3-3) using (6.3-6) and (6.1-23), we obtain a form containing the sum of seven numerical convolutions. However, each convolution involves a reasonably well-behaved integrand. Overall, the computational effort is also lightened by the reduction in total terms due to splitting up the 3-hop jamming events into 1-hop events in (6.3-6) and 2-hop events in (6.3-5) for which, at least in part, analytical results are available. This is the method implemented by the computer program given in Appendix L.

6.4 NUMERICAL RESULTS

The numerical computations for the self-normalizing receiver suffer difficulties similar to those encountered for the clipper receiver, namely a multitude of multiple-dimensional numerical integrations. For the case of M=2 and L=2, from (6.2-2) and (6.1-16), we have the simplest case to compute, consisting of 7 one-dimensional integrals. For the case of M=2 and L=3, we must do 5 two-dimensional numerical integrations and 2 three-dimensional numerical integrations to obtain a value of $P_b(e)$ for given E_b/N_0, E_b/N_J, and γ. For the case of M=4 and L=2, we are faced with numerical integrations in five or six dimensions over non-standard regions (see, for example, Figure 6.2-1 for the region of integration of the outermost 3 dimensions).

At the computational throughput rate of the PDP-11/44 computer available for the computations, the CPU time to obtain results for even M=4 and L=2 are estimated to run to many months, or even years. Hence, we restrict our numerical computations to M=2 and L=2 and 3. For L=2, the speed of computation was sufficiently high to permit the full set of curves for $\gamma = 0.001$ to $\gamma = 1.0$, to be computed using the program in Appendix K. Figure 6.4-1 shows the performance as a function of E_b/N_J when $E_b/N_0 = 13.35247$ dB, which corresponds to $P_b(e) = 10^{-5}$ for ideal MFSK with M=2. We note in Figure 6.4-1 that there is a clustering of the cross-over around $E_b/N_J = 16$ dB.

If E_b/N_0 is increased to 20 dB, the performance curves shown in Figure 6.4-2 are obtained. In this figure we observe that many of the curves exhibit a breakpoint at which the direction of curvature changes. Clearly, the self-normalizing receiver departs considerably from the ideal receiver under certain ranges of operating conditions. The exact mechanisms which come into play to explain this behavior are not totally clear, but it appears to be the

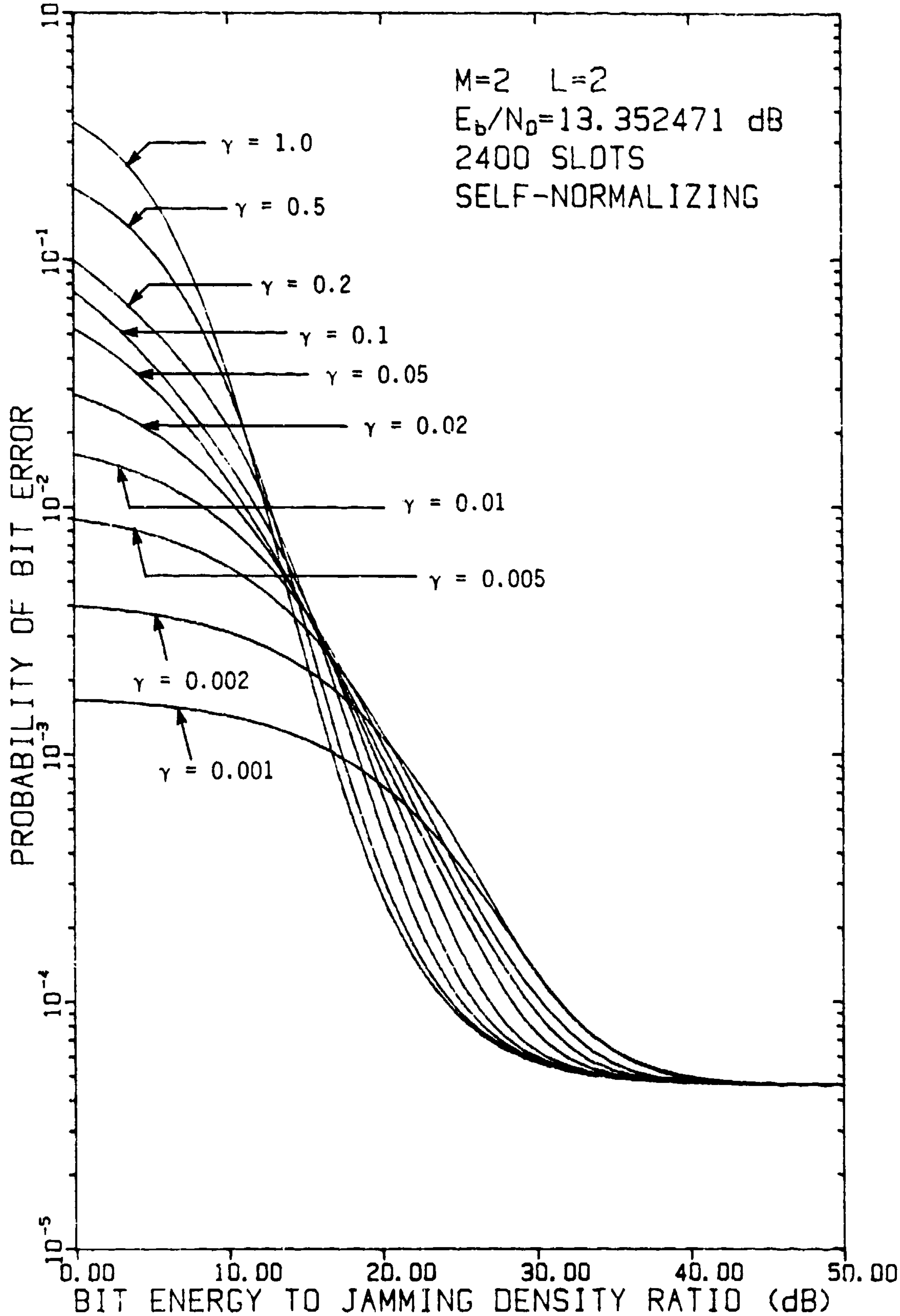


FIGURE 6.4-1 PERFORMANCE OF SELF-NORMALIZING RECEIVER FOR FH/RMFSK WITH M=2, L=2 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.35247 dB (FOR $P_b(e)=10^{-5}$ IN THE ABSENCE OF JAMMING WHEN L=1)

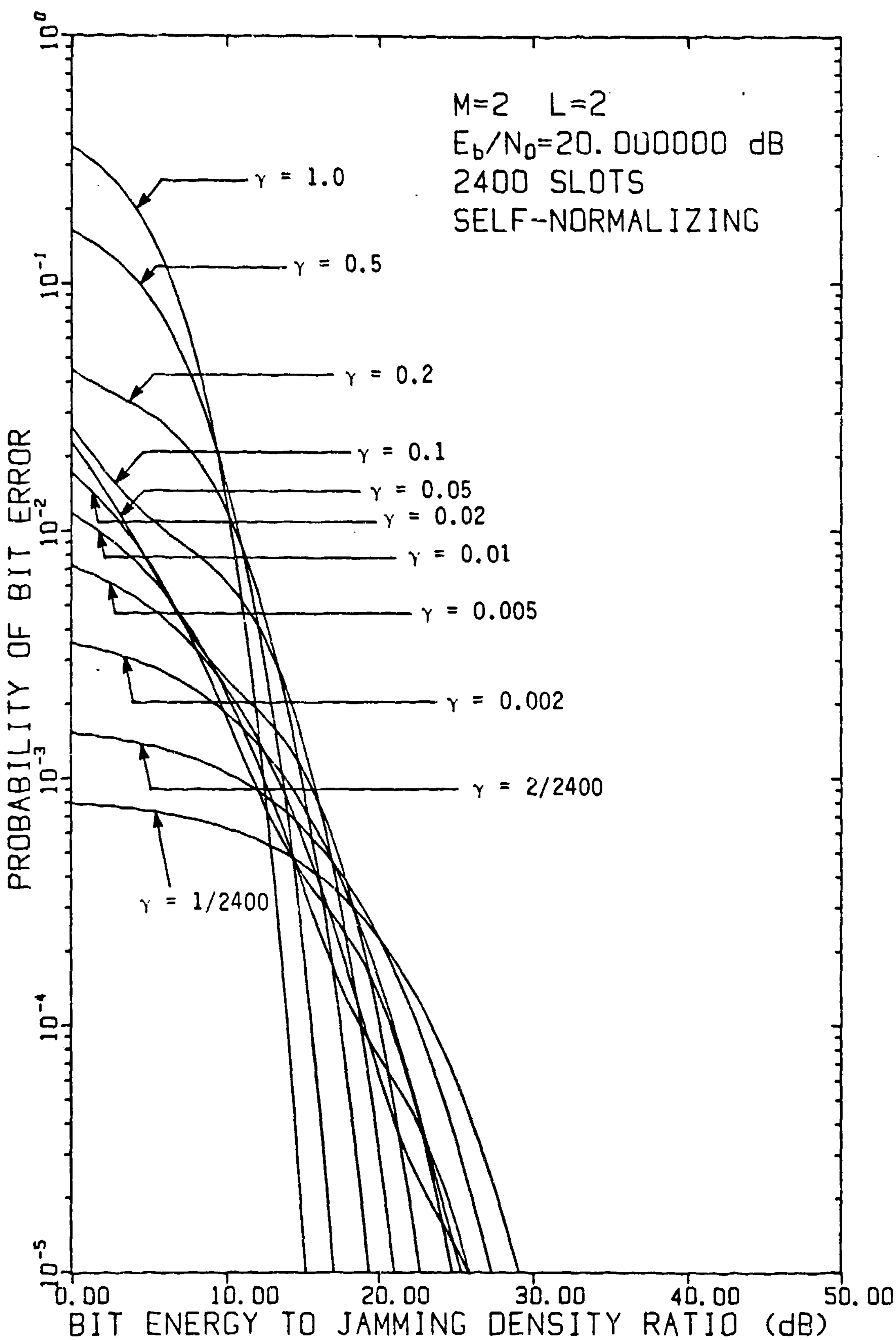


FIGURE 6.4-2 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR SELF-NORMALIZING RECEIVER IN WORST-CASE PARTIAL-BAND NOISE JAMMING FOR M=2, L=2 AT E_b/N_0 = 20 dB

interaction of several different effects, with the switch-over from thermal-noise-limited operation to partial-band-jamming-limited operation playing a significant role. The importance of this switch-over is supported by the lack of apparent breakpoints in the curves for very small γ (γ = 0.001, 0.002) and very large γ (γ = 0.5, 1.0); these are the cases in which the one-slot-jammed jamming event predominates and the shape of the curves reflects essentially the performance conditioned on the dominant event.*

For the case of L=3 hops, the reduced speed of computation dictated that we search for the optimum jamming fraction at each E_b/N_J rather than run full curves for the various values of γ. Again, we started at γ = 1/2400 for E_b/N_J = 50 dB to speed the search, and then stepped to a lower value of E_b/N_J. The computer program in Appendix L was used to obtain the results presented in Figure 6.4-3 for M=2, L=3.

Finally, Figure 6.4-4 compares the performances of the self-normalizing receiver as L, the number of hops per symbol, varies. As we have observed with the other receivers, there is a limited range of E_b/N_J over which a "diversity" effect is achieved. For example, L=3 outperforms L=2 for 17 dB < E_b/N_J < 29 dB. However, in the thermal-noise-limited region and in the strong-jamming region (where γ = 1.0 is the worst-case jamming), the noncoherent combining loss dominates and L=1 is optimum.

---

* For further discussion, see Section 7.3.3.5.

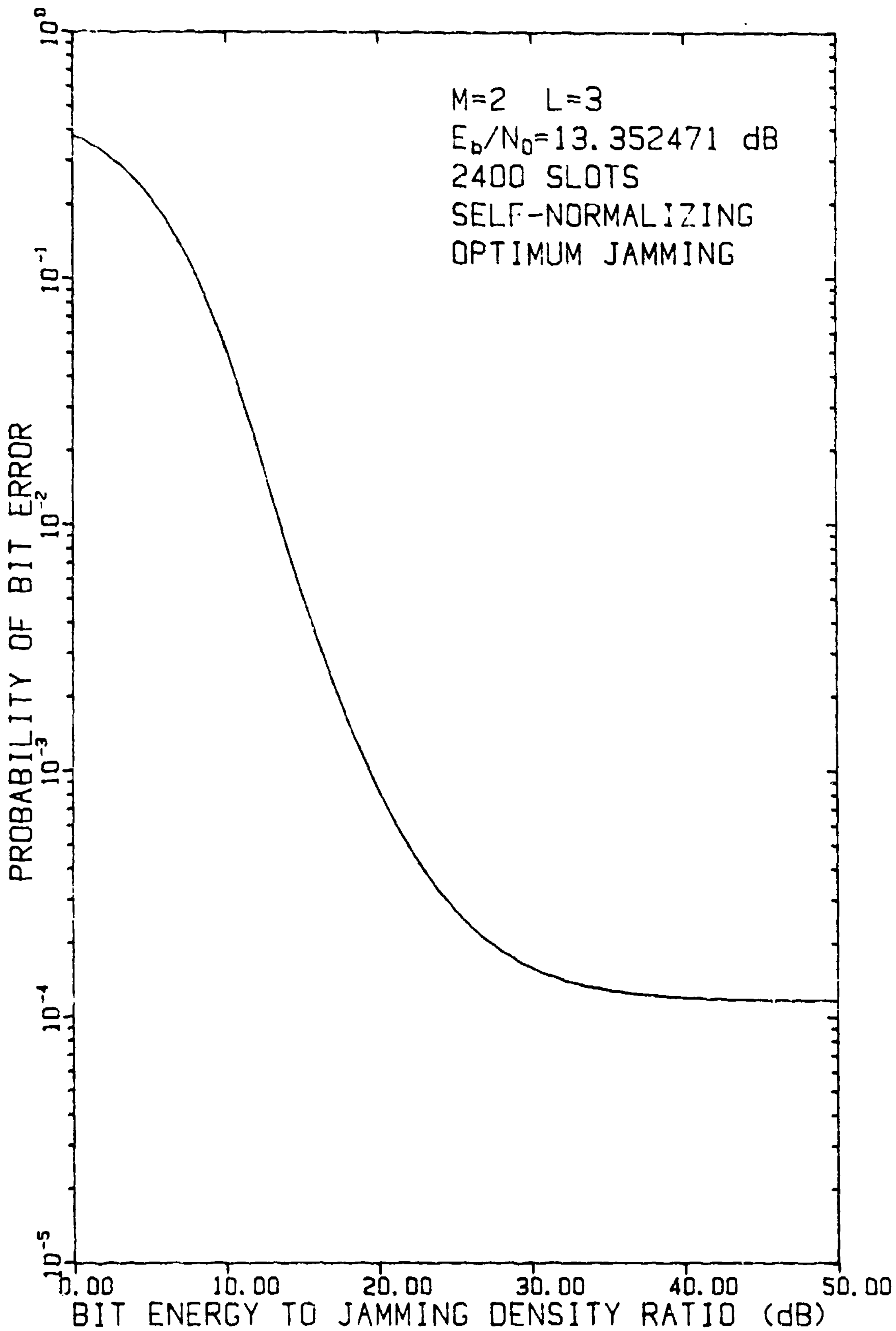


FIGURE 6.4-3 PERFORMANCE OF SELF-NORMALIZING RECEIVER FOR FH/RMFSK WITH M=2, L=3 HOPS/SYMBOL, 2400 HOPPING SLOTS, AND E_b/N_0 = 13.35247 dB (FOR $P_b(e) = 10^{-5}$ IN THE ABSENCE OF JAMMING WHEN L=1)

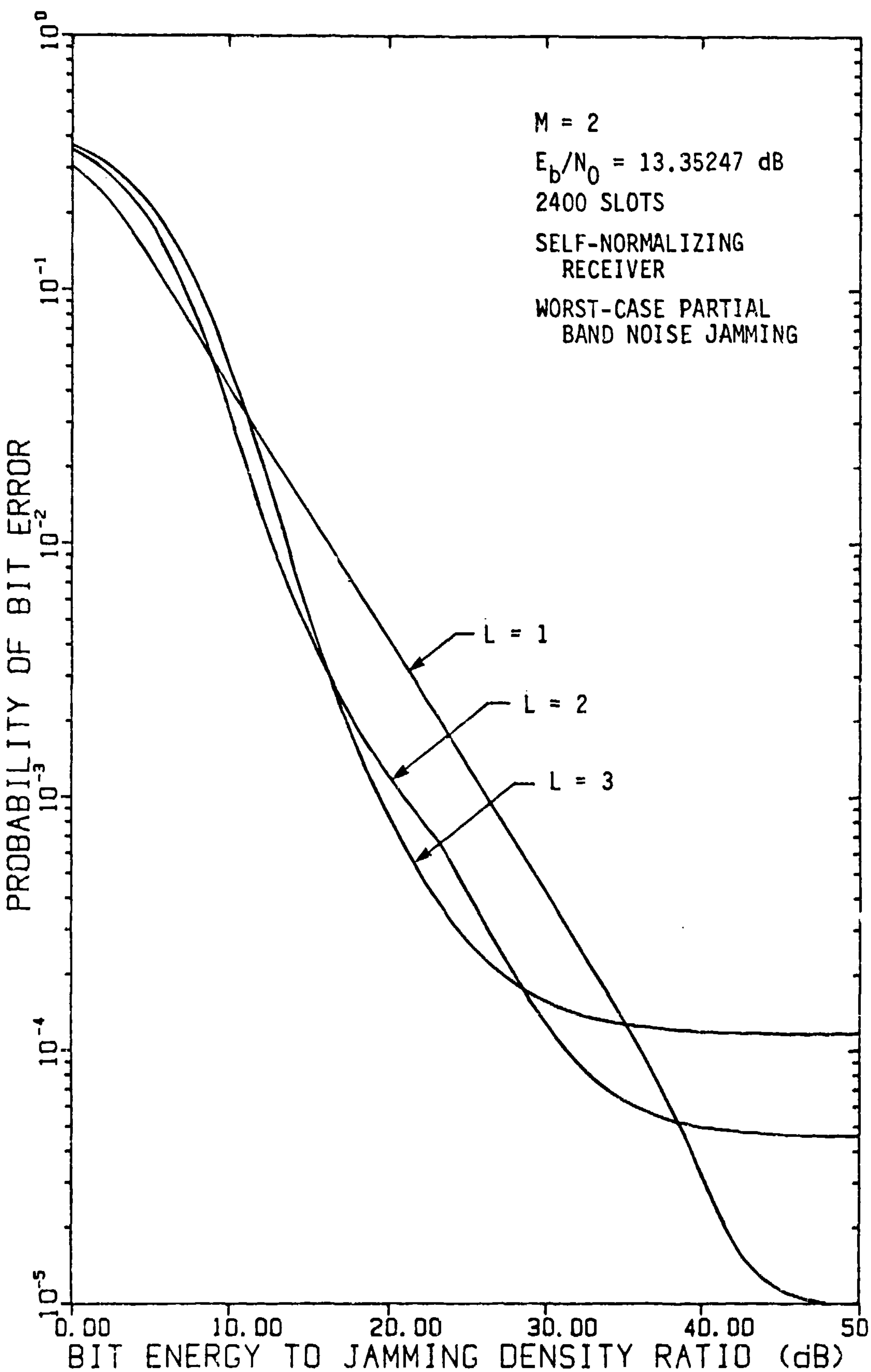


FIGURE 6.4-4 PERFORMANCE OF SELF-NORMALIZING RECEIVER FOR FH/RMFSK WITH M=2, AND E_b/N_0 = 13.35247 dB WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER

7.0 COMPARISONS OF RECEIVER PERFORMANCES

The information we have generated separately on the performances of the various FH/RMFSK receivers in Sections 3-6 can now be compared to learn which ECCM processing scheme is most effective in worst-case partial-band noise jamming. However, since the random MFSK waveform is specifically designed to counter follow-on jamming, we first develop the performances of these receiver processing schemes in follow-on jamming, both for conventional hopping and for random hopping.

7.1 RECEIVER PERFORMANCES IN FOLLOW-ON NOISE JAMMING

7.1.1 Formulation of Follow-on Jamming Analysis: Simple Jammer.

Under follow-on noise jamming (FNJ), it is assumed that on each hop, the jammer places his available power, J watts, in a relatively narrow band centered on the signal's hop frequency. If this band is at least 2(M-1) + 1 slots wide, then the jammer is guaranteed to jam all M slots of conventional FH/MFSK on every hop*. The hop SNR for FNJ therefore is

$$\rho_T = \frac{S}{\sigma_N^2 + \sigma_J^2} = \frac{E_h}{N_0 + N_J/\gamma_r} = \frac{\log_2 M}{L} \cdot \frac{E_b}{N_0 + N_J/\gamma_r} \qquad (7.1\text{-}1a)$$

where the jammer spectral density N_J is defined over the entire hopping system bandwidth, $N_J = J/W$, and therefore the effective jamming fraction is

$$\gamma_r = \frac{W_J}{W} \geq \frac{2M-1}{N} \quad . \qquad (7.1\text{-}1b)$$

In terms of the jamming events indexed by the vector $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$, where ℓ_m is the number of hops jammed in symbol frequency

*In Section 7.1.2, we consider an "advanced" FNJ which excludes jamming from the signal slot.

channel m, for repeat jamming we have

$$\ell_1 = L. \tag{7.1-2}$$

The values of the other $\{\ell_m\}$ depend on the hopping scheme.

7.1.2 Formulation of FNJ Analysis: Advanced Jammer.

The FNJ can avoid helping the communicator by not putting any jammer power in the signal slot. Assuming that this measure is taken, the hop SNR is ρ_N in the signal slot. For maximum effectiveness against FH/MFSK, we also assume that the jammer places half its power in each of the two slots on either side of the intercepted signal, as illustrated in Figure 7.1-1. Thus $\ell_1 = 0$, and for a single hop, for FH/MFSK,

$$\left.\begin{aligned} \Pr\{1 \text{ nonsignal slot jammed}\} &= \frac{2}{M} \\ \Pr\{2 \text{ nonsignal slots jammed}\} &= 1 - \frac{2}{M}\,. \end{aligned}\right\} \tag{7.1-3a}$$

For random hopping and advanced FNJ,

$$\left.\begin{aligned} \Pr\{0 \text{ nonsignal slots jammed}\} &= \frac{(N-M)(N-M-1)}{(N-1)(N-2)} \\ \Pr\{1 \text{ nonsignal slot jammed}\} &= 2\,\frac{(N-M)(M-1)}{(N-1)(N-2)} \\ \Pr\{2 \text{ nonsignal slots jammed}\} &= \frac{(M-1)(M-2)}{(N-1)(N-2)}\,. \end{aligned}\right\} \tag{7.1-3b}$$

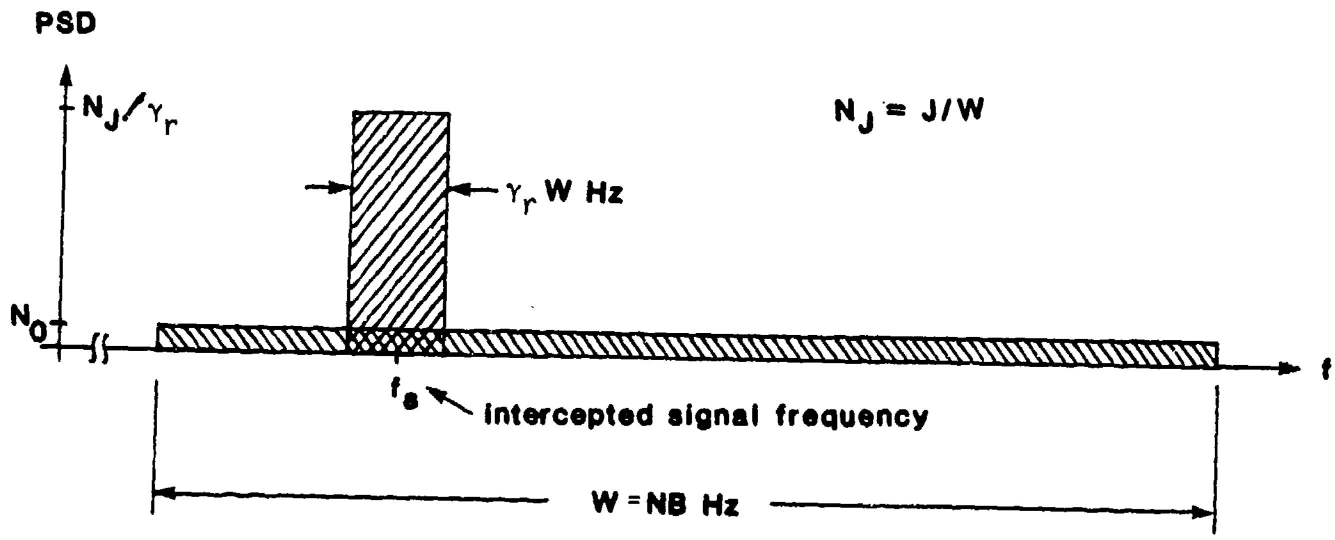
7.1.3 Performance of Conventional FH/MFSK in FNJ.

For conventional FH/MFSK, the M symbol frequency slots are contiguous at RF on each hop. Under simple FNJ, therefore, all M slots are jammed on each hop. That is,

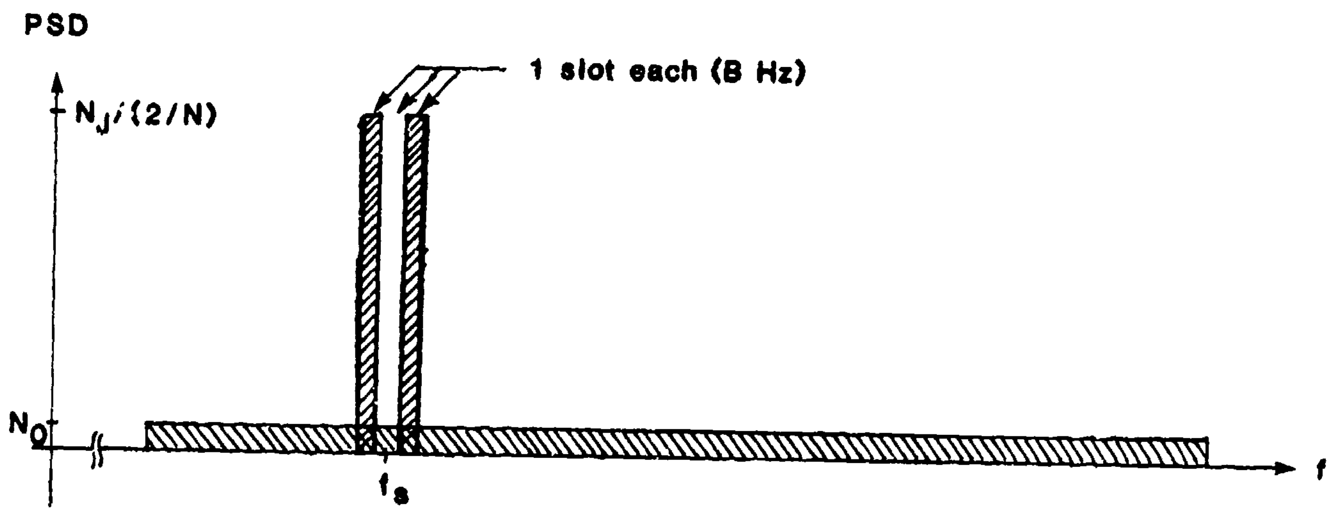
$$\ell_2 = \ell_3 = \cdots = \ell_M = L. \tag{7.1-4}$$

The error probability for the system then is

$$P_b(e;\gamma_r) = \frac{M/2}{M-1} P_s(e;\gamma_r \mid \ell_m = L, \text{ all } m). \tag{7.1-5}$$



(a) "Simple" follow-on jammer.



(b) "Advanced" follow-on jammer.

FIGURE 7.1-1 THERMAL NOISE AND FOLLOW-ON JAMMER MODELS

For the several receiver processing schemes we have the results using ρ_T given in (7.1-1),

$$P_b(e;\gamma_r) = \text{probability of error for broadband jamming } (\gamma=1), \text{ with } E_b/N_J \text{ replaced by } E_b/N_J + 10\log_{10}\gamma_r. \quad (7.1\text{-}6)$$

Thus the error curves for rollow-on jamming, $P_b(e)$ vs. E_b/N_J with E_b/N_0 = constant, are those for full-band jamming, but moved to the right by $-10 \log_{10} \gamma_r$ dB.

Figure 7.1-2 illustrates the performance of conventionally-hopped FH/MFSK in follow-on noise jamming, assuming the follow-on jammer's bandwidth guarantees jamming of all M slots of the symbol. For example, if there are 2400 hopping slots and the jammer occupies 100 slots, then γ_r = 100/2400 = -13.8 dB, so that the effective E_b/N_J shown in the figure runs from about 14 dB to 64 dB. Something like 25 dB E_b/N_J gives an error rate of 10^{-2} for L = 1.

For the advanced FNJ, the effect on FH/MFSK is to produce the symbol error rate

$$P_s = \frac{2}{M} P_s(e|\ell_2=L;\ell_m=0,m\neq2) + \left(1-\frac{2}{M}\right) P_s(e|\ell_2=\ell_3=L;\ell_m=0,m\neq2,3); \quad (7.1\text{-}7a)$$

with

$$\gamma_r = 2/N. \quad (7.1\text{-}7b)$$

Figure 7.1-3 illustrates the jammed BER for this case for M=2, 4, and 8. It is quite clear the the follow-on capability gives the jammer a tremendous advantage against FH/MFSK.

7.1.4 Performance of FH/RMFSK in Follow-on Jamming.

The FH/RMFSK hopping scheme is designed to defeat FNJ by making it difficult for the jammer to jam the nonsignal slots; the nonsignal slots are distributed randomly in the hopping band. The simple follow-on jammer

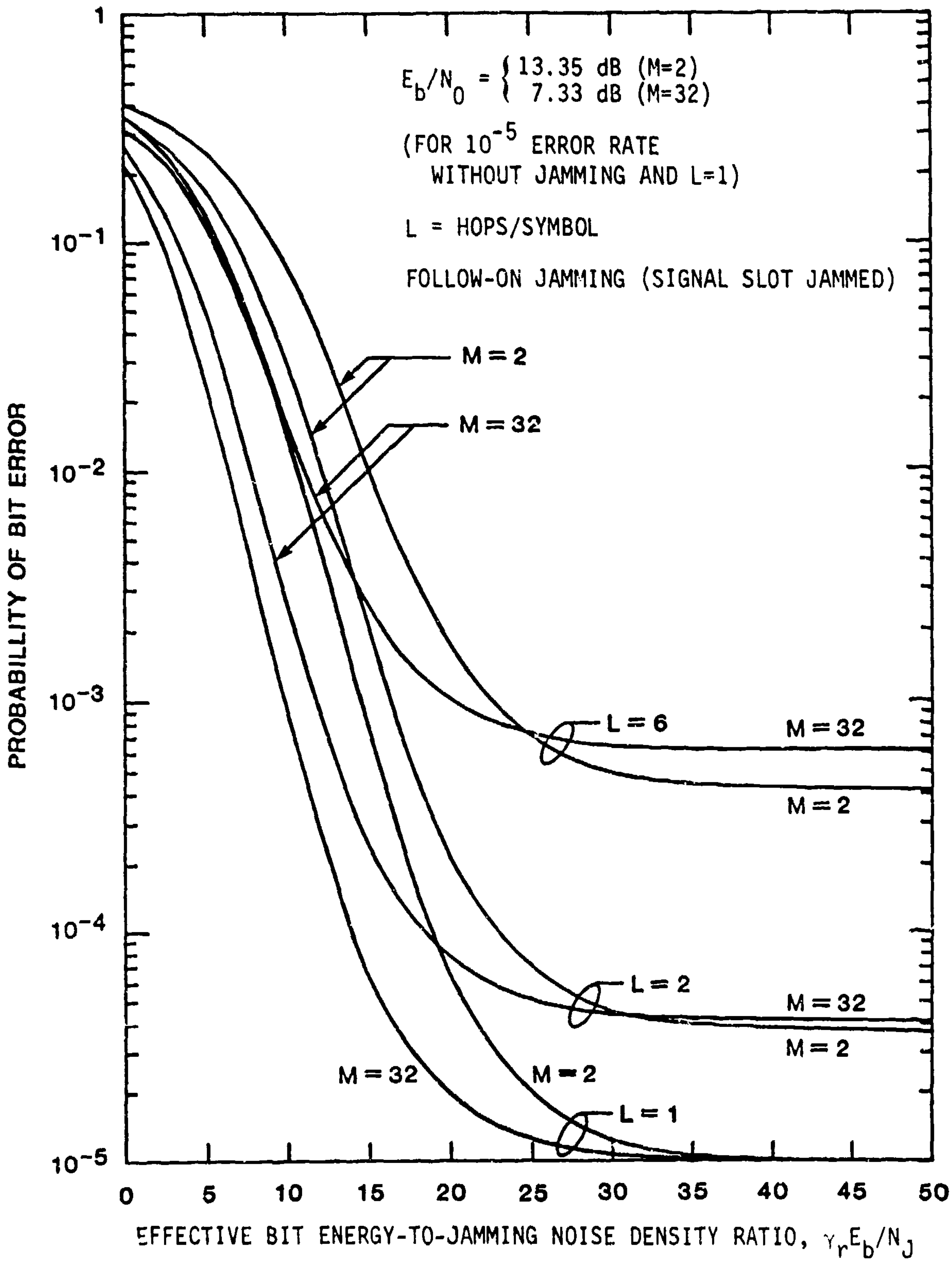


FIGURE 7.1-2 FOLLOW-ON JAMMING PERFORMANCE OF LCR AND ACJ-AGC RECEIVERS FOR FH/MFSK WITH L=1,2,6 AND M=2,32 WHEN E_b/N_0 YIELDS A 10^{-5} BER FOR L=1

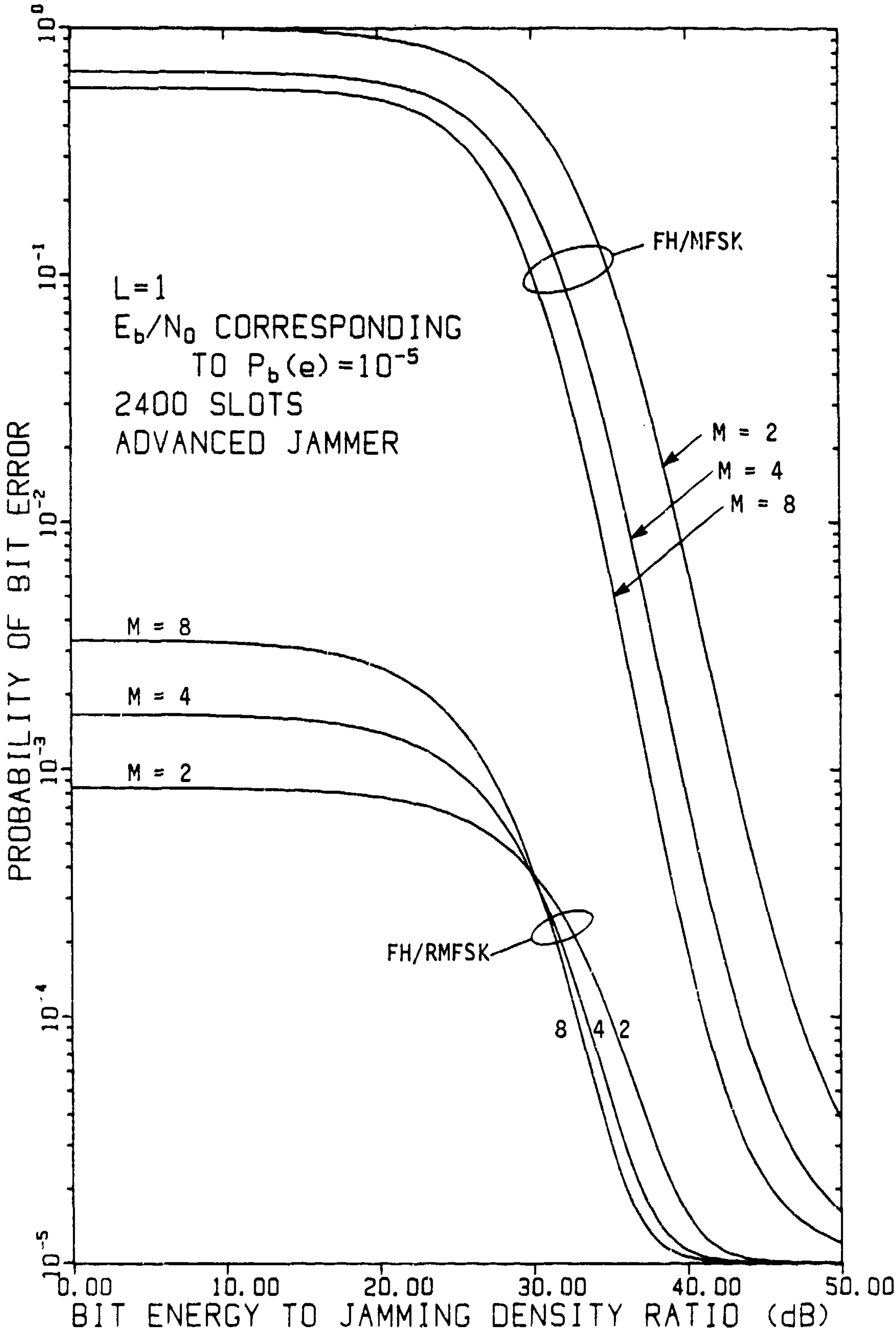


FIGURE 7.1-3 ADVANCED FOLLOW-ON NOISE JAMMING PERFORMANCE OF FH/RMFSK AND FH/MFSK FOR L = 1 HOP/SYMBOL AND UNJAMMED BER = 10^{-5}

can very likely help the receiver by placing more RF power into the signal's slot.

For M<<N, that is, for a very wide hopping band compared to the symbol bandwidth, the effect of FNJ on FH/RMFSK is approximately to guarantee that

$$\ell_1 = L$$
$$\ell_2 = \ell_3 = \cdots = \ell_M = 0. \qquad (7.1\text{-}8)$$

This has two effects. First, the per-hop SNR ρ_T is decreased since additional noise is inserted in the signal channel by the jammer. Second, the scaling of the square-law envelope samples increases for the same reason. Since the average value of a single sample is

$$E\{z_{1k}\} = 2(\sigma_T^2 + S),$$
$$E\{z_{mk}\} = 2\sigma_N^2 , \quad m \geq 2, \qquad (7.1\text{-}9)$$

on the whole we anticipate that repeat jamming will increase the probability that the receiver will make a correct decision.

7.1.4.1 Soft-decision receivers.

For the various soft-decision receivers studied, under simple FNJ the decision variables are described as follows:

Linear combining (conventional) receiver

$$z_1 = \sigma_T^2 \chi^2(2L;2L\rho_T)$$
$$z_m = \sigma_N^2 \chi^2(2L), \quad m \geq 2. \qquad (7.1\text{-}10)$$

AGC - individual channel normalization receiver (IC)

$$z_1 = \chi^2(2L;2L\rho_T)$$
$$z_m = \chi^2(2L), \quad m \geq 2. \qquad (7.1\text{-}11)$$

AGC - any channel jammed normalization receiver (ACJ)

$$z_1 = \chi^2(2L, 2L\rho_T)$$

$$z_m = \frac{\sigma_N^2}{\sigma_T^2}\chi^2(2L), \quad m \geq 2. \qquad (7.1\text{-}12)$$

Clipper receiver

$$z_1 = \sum_{k=1}^{L} [\sigma_T^2 \chi^2(2;2\rho_T)] \text{ clip at } \eta$$

$$z_m = \sum_{k=1}^{L} [\sigma_N^2 \chi^2(2)] \text{ clip at } \eta . \qquad (7.1\text{-}13)$$

Since multiplication of all channels by a constant factor does not affect the error probability, we observe that the conventional and AGC-ACJ receivers will achieve the same performance in FNJ:

$$P_b(e;\gamma_r)_{ACJ} = \frac{M/2}{M-1} \sum_{k=1}^{M-1} \binom{M-1}{k} \frac{(1)^{k+1}}{(1+kK)^L} \exp\left\{ - \frac{kKL\rho_T}{1+kK} \right\}$$

$$\times \sum_{r=0}^{k(L-1)} C(k,r) \left(\frac{K}{1+kK} \right)^r \mathcal{L}_r^{L-1} \left(\frac{-L\rho_T}{1+kK} \right), \qquad (7.1\text{-}14a)$$

where

$$K \triangleq \sigma_T^2/\sigma_N^2, \qquad (7.1\text{-}14b)$$

$$C(k,r) = \frac{1}{r} \sum_{n=1}^{\min(r,L-1)} \binom{r}{n} \left[(k+1)n - r \right] C(k,r-n), \qquad (7.1\text{-}14c)$$

$$C(k,0) = 1,$$

and $\mathcal{L}_r^{L-1}(\cdot)$ denotes the generalized Laguerre polynomial of degree r with parameter L-1.

The performance for the AGC receiver with individual channel normalization is the same as given by (7.1-14), but with K=1.

$$P_b(e;\gamma_r)_{IC} = P_b(e;\gamma_r)_{ACJ} \Big|^{K=1} . \qquad (7.1\text{-}15)$$

The performance of the clipper receiver in follow-on jamming and FH/RMFSK is

$$P_b(e;\gamma_r)_C = 1 - \int_0^{L\eta^*} dx\, f_L(x;L)\left[G_L(x;0)\right]^{M-1} - \left[Q\left(\sqrt{2\rho_T},\sqrt{\eta^*/\sigma_T^2}\right)\right]^L \frac{e^{L\eta^*/2\sigma_N^2}}{M} \left[1 - \left(1-e^{-L\eta^*/2\sigma_N^2}\right)^M\right], \qquad (7.1\text{-}16)$$

where η^* is the optimum clipping threshold. For L=1, this threshold is infinite, causing the clipper receiver to have the same performance as the ACJ receiver under follow-on jamming for that case.

Figure 7.1-4 illustrates the performance of the randomly-hopped FH/RMFSK receivers against simple FNJ for L=1. We observe that the error probability is maximized for a particular value of $\gamma_r E_b/N_J$; for the binary case, this value is slightly greater than 0 dB, while for M=4 and M=8, it is approximately -2.5 dB and -4.0 dB, respectively, for the assumed values of E_b/N_0 (chosen to achieve 10^{-5} error rate without jamming). It is interesting to note that for very strong jamming (to the left of the maximum error), the error rate increases with M as does the maximum error. Using the example of the last subsection, a 10^{-2} error rate is achieved for an E_b/N_J of about 17 dB for $\gamma_r \approx -14$ dB, an 8 dB improvement over conventional hopping.

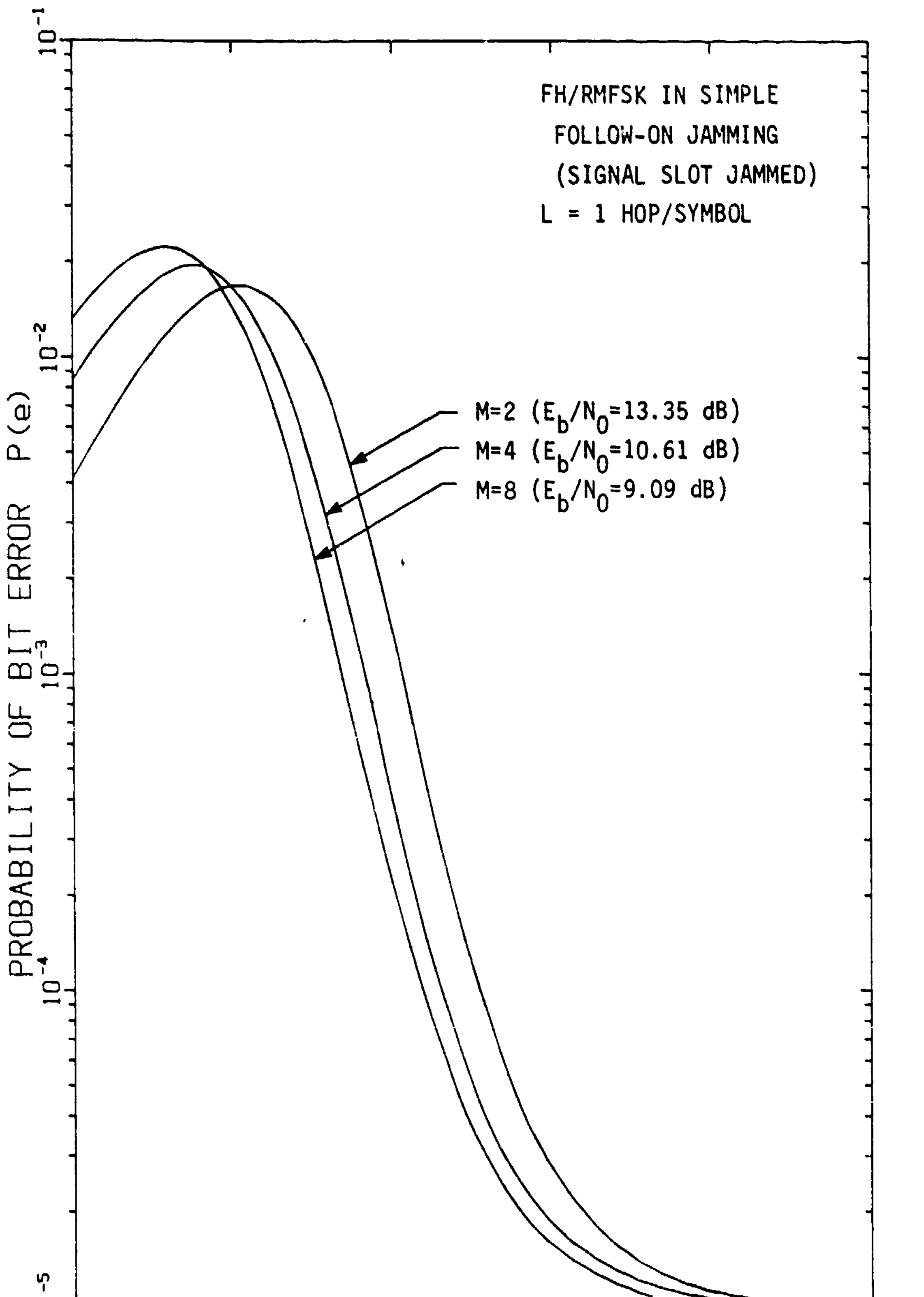


FIGURE 7.1-4 PERFORMANCE OF FH/RMFSK RECEIVERS IN SIMPLE FOLLOW-ON NOISE JAMMING VS. EFFECTIVE BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR L=1 HOP/SYMBOL AND M=2,4,8 WHEN E_b/N_0 YIELDS A 10^{-5} BER WITHOUT JAMMING

Figure 7.1-5 demonstrates that the same critical jammer power effect is observed for L=2, but at different values (approximately 3 dB higher), so that the jammer must know L in order to be effective. It is also evident from this figure that the maximum error rate is decreased by increasing L to 2.

For the advanced FNJ described above, using RMFSK hopping, there is only a slight chance of jamming the symbol on a given hop. The possible jamming events for a single hop are the vectors

$$\underline{\nu}_k = (0, \nu_{2k}, \nu_{3k}, \ldots, \nu_{MK}) \tag{7.1-17}$$

with probabilities

$$\Pr\{r \text{ nonsignal slots jammed}\}$$
$$= \pi_r = \frac{\binom{N-M}{2-r}}{\binom{N-1}{2}} \binom{M-1}{r}. \tag{7.1-18}$$

For M=2 the result is

$$P_b(e) = \sum_{k=0}^{L} \binom{L}{k} \left(\frac{2}{N-1}\right)^k \left(1 - \frac{2}{N-1}\right)^{L-k} P_b(e|\ell_1=0, \ell_2=k). \tag{7.1-19a}$$

$$\approx P_b(e|\ell_1=\ell_2=0) + \frac{2L}{N-1} P_b(e|\ell_1=0, \ell_2=1), \; N-1 >> 2. \tag{7.1-19b}$$

Thus the FH/RMFSK hopping scheme achieves very nearly the unjammed error performance of MFSK when the follow-on noise jammer is configured against FH/MFSK. As (7.1-19b) shows, for L=1 and M=2 the unjammed error rate is increased by, at most, 2/(N-1); this quantity equals 8.3×10^{-4} for N=2400, and 7.8×10^{-3} for N=256. For L=1, Figure 7.1-3 shows the performance of FH/RMFSK much improved over FH/MFSK in advanced FNJ.

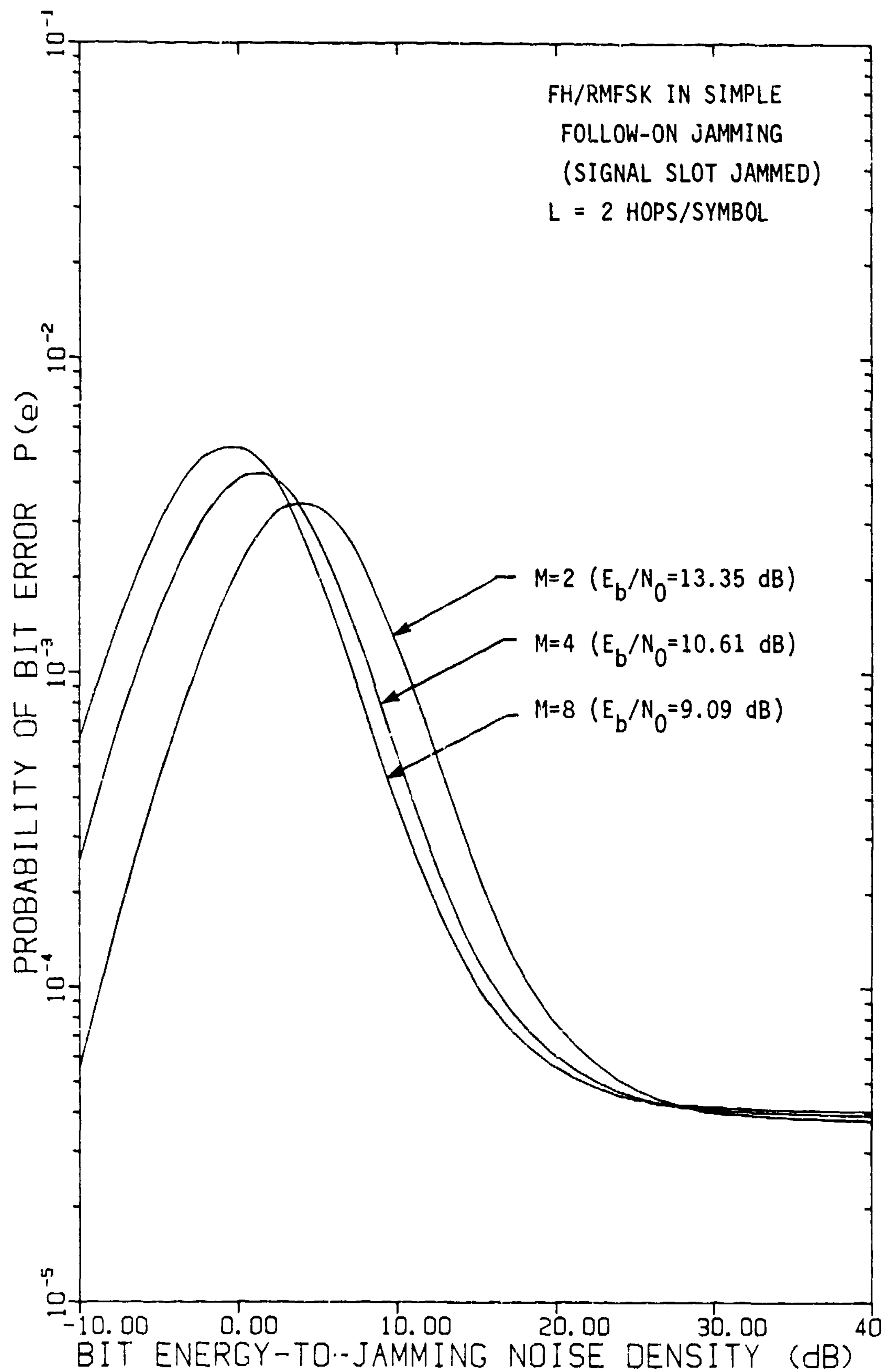


FIGURE 7.1-5 PERFORMANCE OF FH/RMFSK RECEIVERS IN SIMPLE FOLLOW-ON NOISE JAMMING VS. EFFECTIVE BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR L=2 HOPS/SYMBOL AND M=2,4,8 WHEN E_b/N_0 YIELDS A 10^{-5} BER WITHOUT JAMMING

An exception to Figure 7.1-3 for FH/RMFSK is the IC-AGC receiver, for which the "advanced" FNJ is completely nullified. The jammer in this case is "too smart," because the IC-AGC receiver is vulnerable to jamming only if the signal channel is jammed. This statement also holds for the case of conventional MFSK hopping if individual-channel normalization is employed.

7.1.4.2 Hard-decision receiver.

Performance of the hard-decision receiver in simple follow-on noise jamming is depected in Figures 7.1-6 through 7.1-8 for values of M=2, 4, and 8 respectively. The parameter E_b/N_0 was chosen to yield a 10^{-5} BER in the absence of jamming per respective M value for L=1 hop per symbol. It is clearly seen in each of the L P(e) curves that as the jammer power is increased below that E_b/N_J value to cause maximum P(e), a decrease in P(e) takes place. Hence, for strong jamming the jammer is actually aiding the communicator by the addition of energy to the non-coherent FSK signal slot. We also have a diversity improvement for $L \geq 3$ hops per symbol in the strong jamming regions. Conversely, for weak jamming (beyond $E_b/N_J \approx 20$ dB) no diversity improvement is realized for $L \geq 2$ hops per symbol due to the dominance of the noncoherent combining loss existing for the stated thermal noise (E_b/N_0) values. Therefore, in order to be effective (maximum P(e)), the jammer must maintain E_b/N_J to within small ranges. For example, to ensure a minimum P(e) of 10^{-2} for M=2 and L=1 (Figure 7.1-4), E_b/N_J must be held to values ranging from 4 to 14 dB.

The effects of decreased thermal noise levels (E_b/N_0 = 20 dB) for cases of M=2, 4, and 8 are illustrated in Figures 7.1-9 through 7.1-11 respectively. Here we observe all of the L P(e) curves exhibiting a

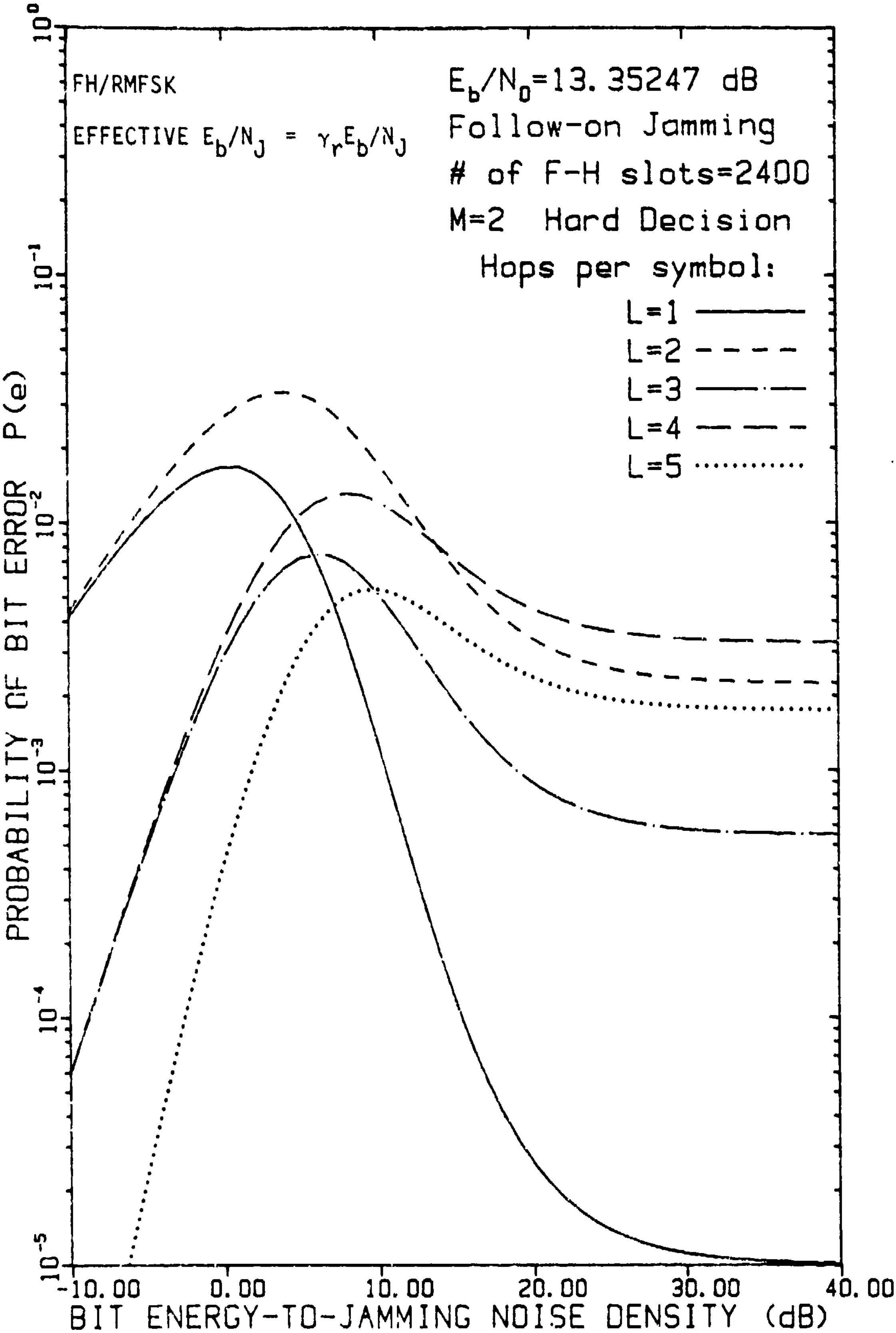


FIGURE 7.1-6 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J FOR M=2 AND E_b/N_0 SUCH THAT UNJAMMED BER IS 10^{-5} WHEN L=1

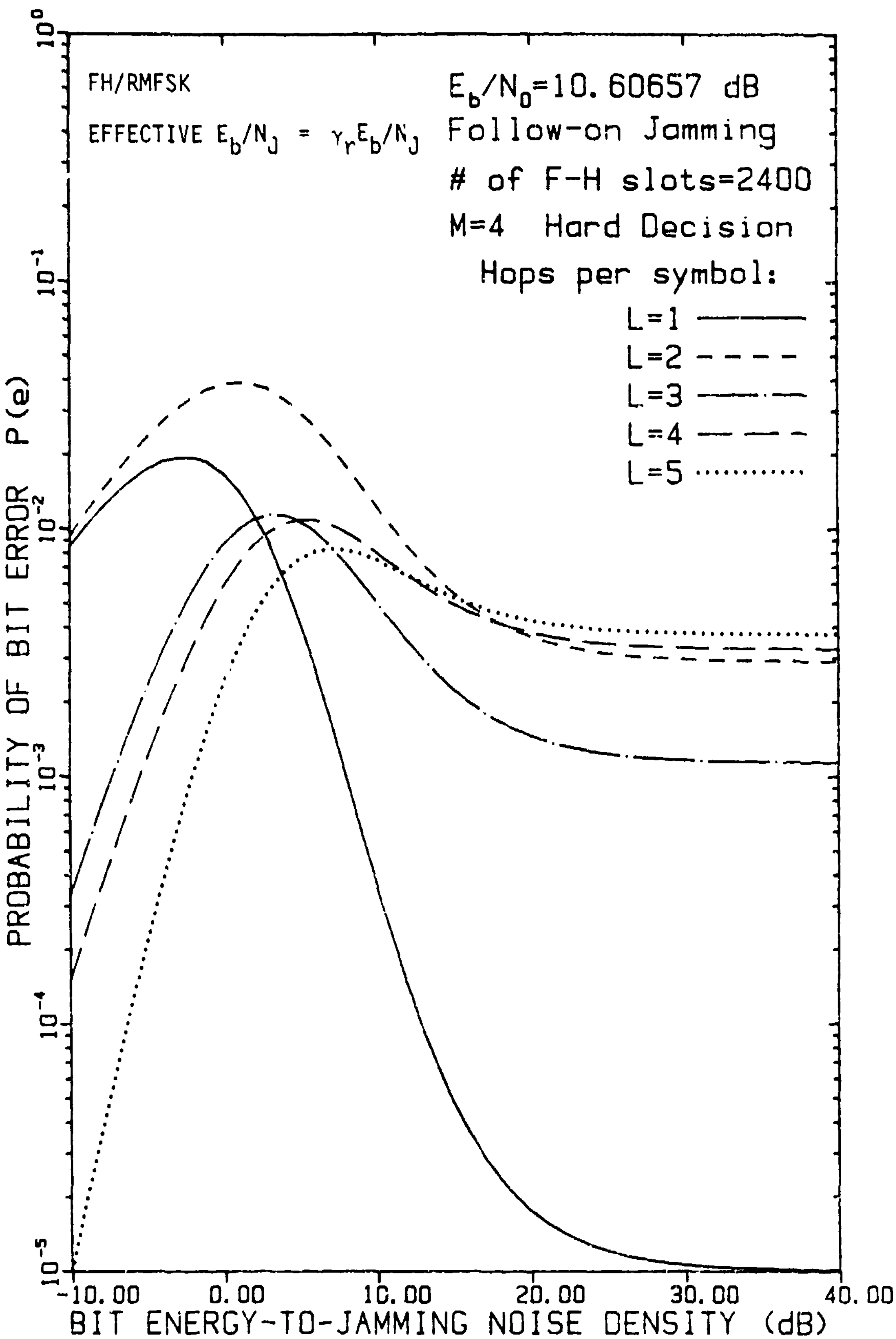


FIGURE 7.1-7 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J FOR M=4 AND E_b/N_0 SUCH THAT UNJAMMED BER IS 10^{-5} WHEN L=1

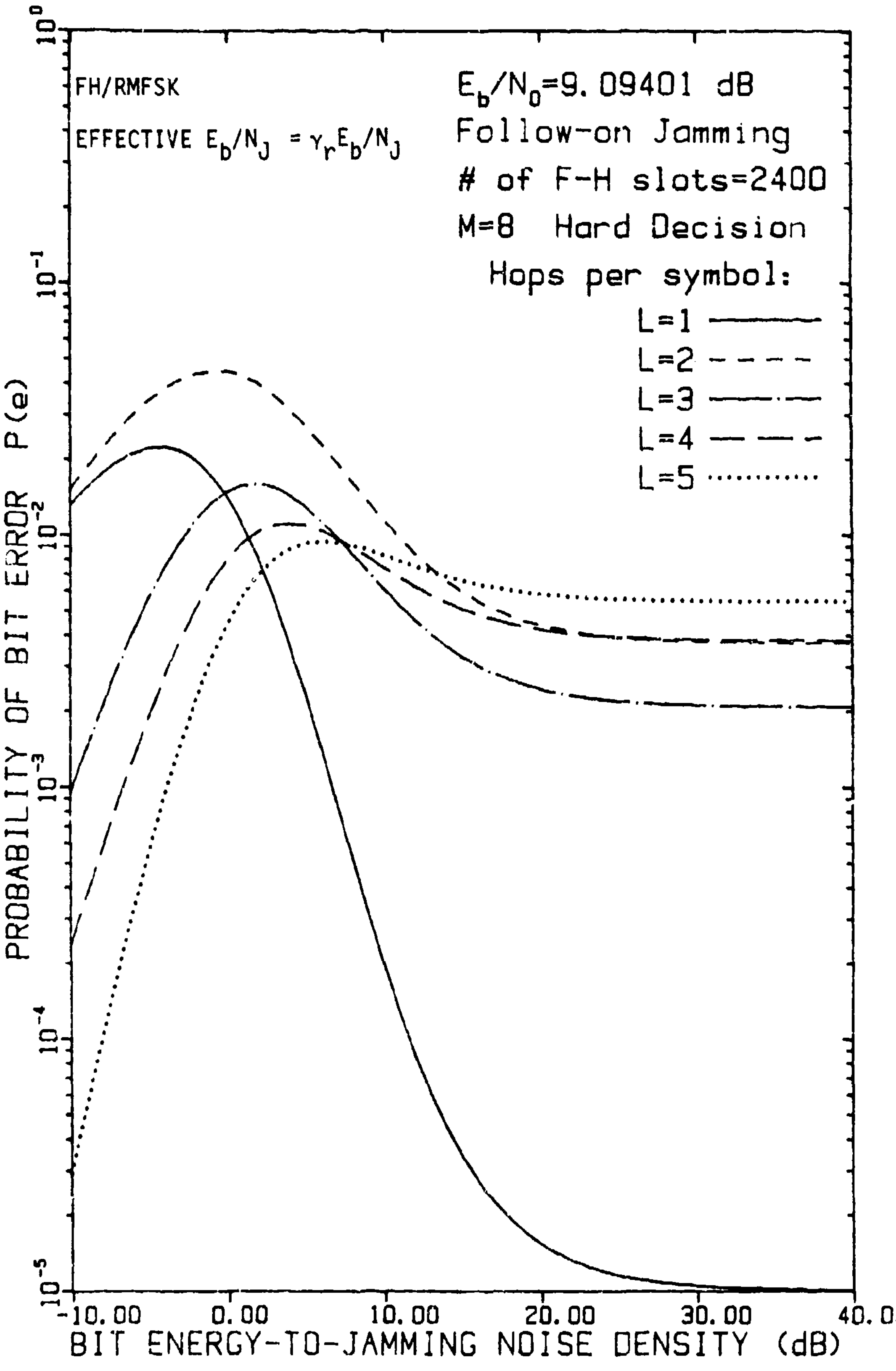


FIGURE 7.1-8 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J FOR M=8 AND E_b/N_0 SUCH THAT UNJAMMED BER IS 10^{-5} WHEN L=1

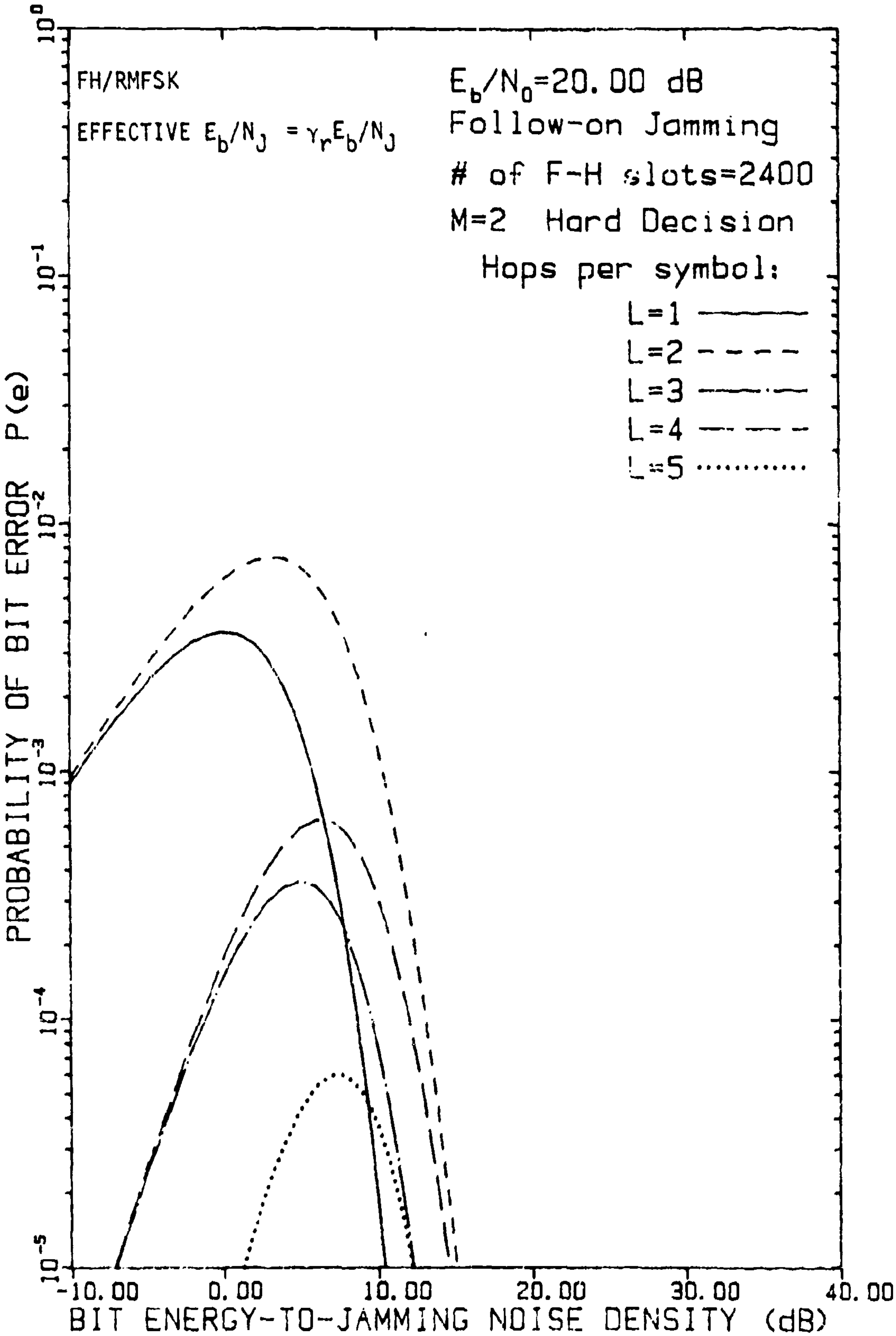


FIGURE 7.1-9 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J FOR M = 2 AND E_b/N_0 = 20 dB

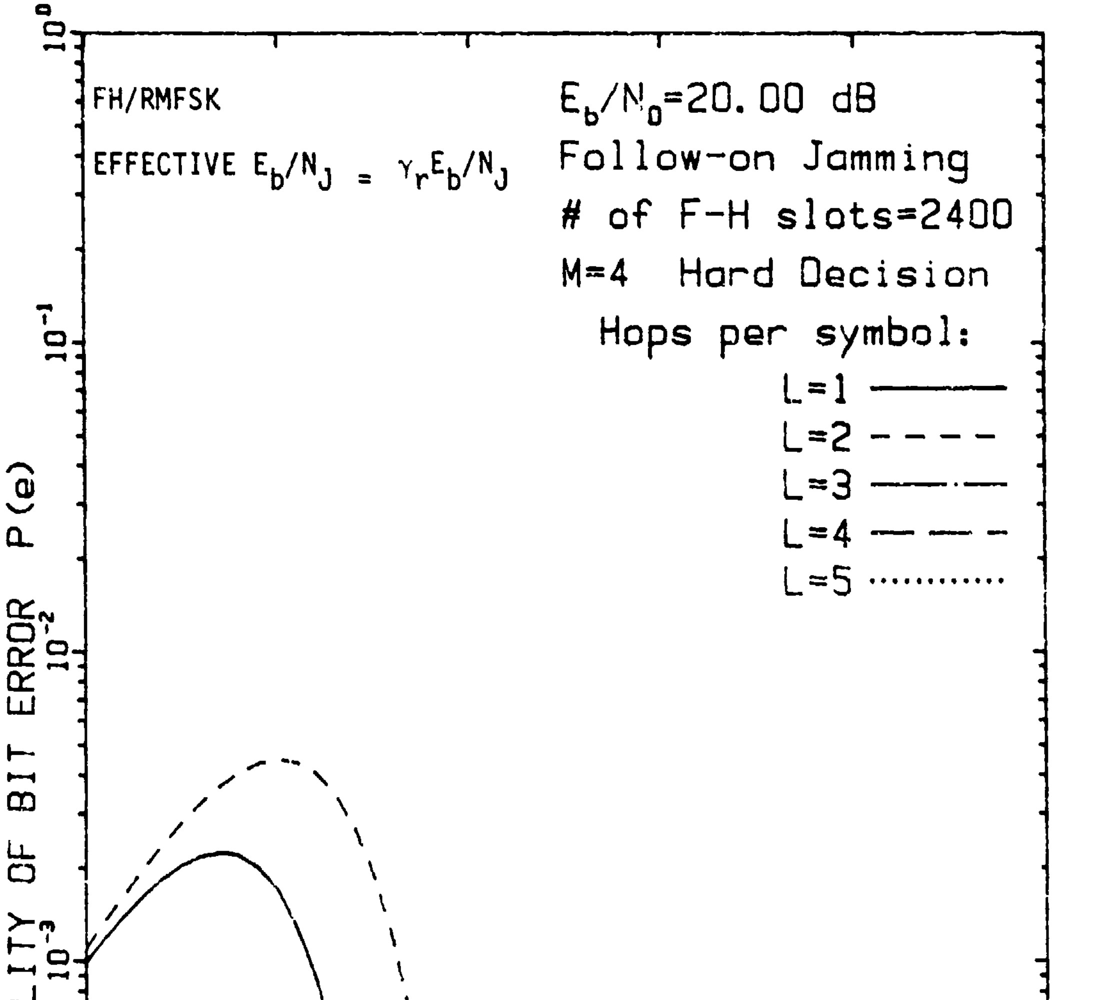


FIGURE 7.1-10 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J FOR M=4 AND E_b/N_0 = 20 dB

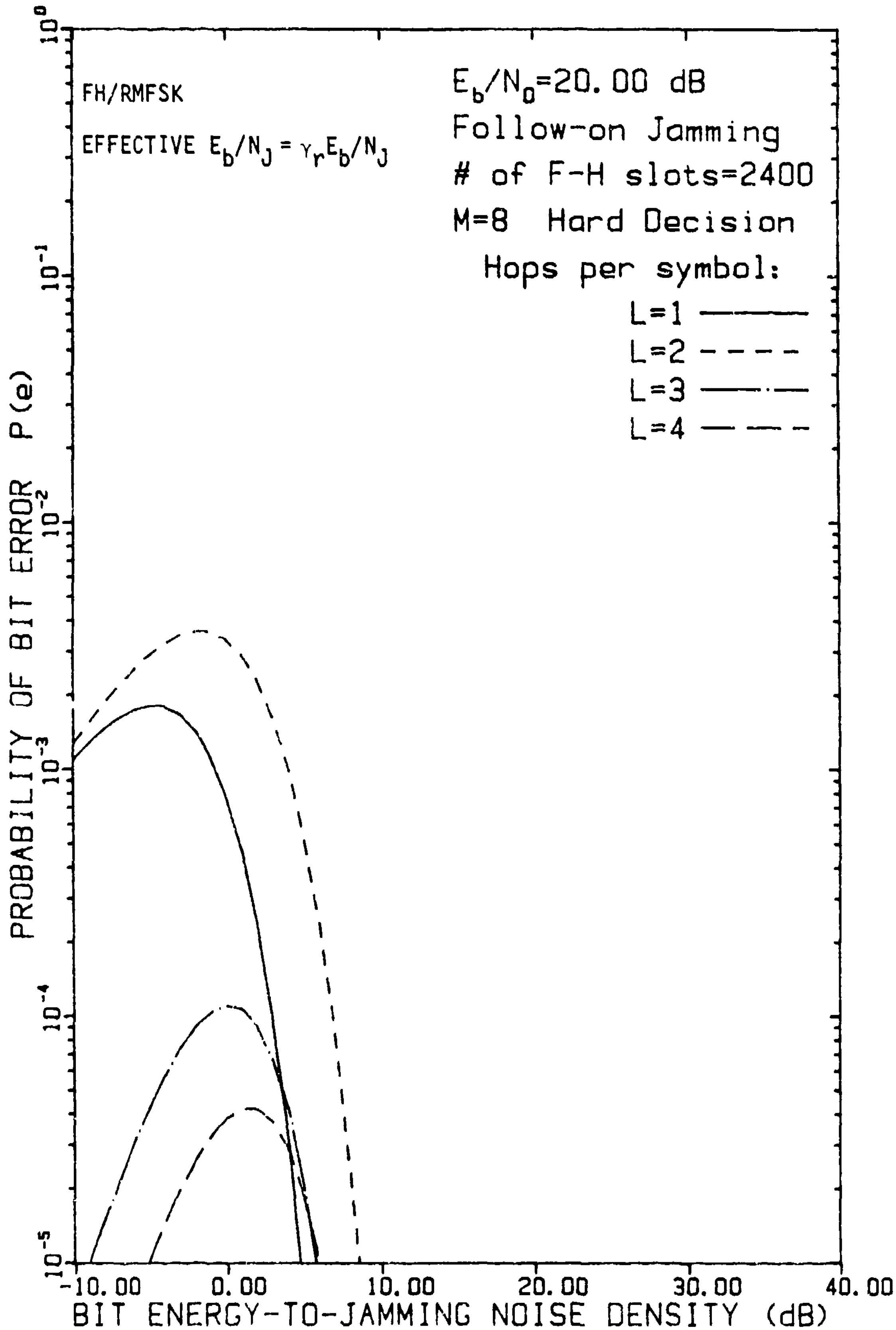


FIGURE 7.1-11 PERFORMANCE OF HARD-DECISION FH/RMFSK RECEIVER IN SIMPLE FNJ VS. EFFECTIVE E_b/N_J WHEN M=8 AND E_b/N_0 = 20 dB

"parabolic" type of behavior, i.e. steeply defined strong and weak jamming regions. Thus, the jammer appears to have quite a narrow window of E_b/N_J values to work within for attaining a maximum effect.

7.2 COMPARISONS OF FH/RMFSK RECEIVER PERFORMANCES IN WORST-CASE PARTIAL-BAND NOISE JAMMING (WCPBNJ)

It is understood that the motivation for using the proposed FH/RMFSK waveform is that it is less vulnerable to follow-on noise jamming (FNJ) or repeat noise jamming than is a conventional FH/MFSK block-hopping system. RMFSK is effective in that the FNJ is not able to place jamming power in the unused symbol frequency slots as is the case for MFSK where the M signalling frequencies are adjacent. At the FH/MFSK or FH/RMFSK receiver, the L hops comprising the MFSK symbol can be combined in a number of ways. Certain types of nonlinear combining soft-decision schemes, which weight the detected hops in some form to discriminate against jammed hops, are employed. Previous results [1] have shown that conventional FH/MFSK system performance with L-hop diversity in WCPBNJ is improved by nonlinear combining techniques. This study has addressed FH/RMFSK system performance in the less sophisticated, yet more pervasive, ECM tactic of PBNJ - a basic jamming threat which is inevitably encountered in an EW scenario. In Sections 3 through 6 it has been demonstrated that nonlinear combining yields improved RMFSK performance in this type of jamming.

In what follows, we compare performances of the different types of ECCM receivers for FH/RMFSK signals in the WCPBNJ environment. In Section 7.3 we also compare FH/MFSK and FH/RMFSK receiver performances in WCPBNJ, and in Section 7.4 consider the different effects of the RMFSK diversity combining techniques studied. Unless stated otherwise, we note that all E_b/N_0 values utilized in performance plots are chosen so as to yield a 10^{-5} BER in the absence of jamming when L=1 hop per symbol.

Comparisons among the different FH/RMFSK receivers analyzed are provided by Figures 7.2-1 and 7.2-2 for M=2 and E_b/N_0 = 13.35 dB, and by Figures 7.2-3 and 7.2-4 for M=4 at E_b/N_0 = 10.61 dB. The enormous amount of computer time required to obtain performance results for the clipper receiver at L=3 and the SNORM receiver for M=4 is beyond the scope of this study. Therefore, in some figures for comparison, these receivers are not represented. Explanations of the difficulties involved in such calculations were presented in the numerical results of Section 5 (clipper receiver) and Section 6 (SNORM receiver).

We can develop a performance ranking for these receivers with their respective parameter sets (M,L values) by assessing performances in the regions of strong, moderate, and weak jamming. These arbitrary regions are taken to mean the following: (1) strong jamming - usually full-band jamming with E_b/N_J values less than about 4 to 8 dB, (2) weak jamming - region of very small γ values with E_b/N_J usually greater than 35 to 40 dB, and (3) moderate jamming - area between strong and weak jamming.

For the case of M=2, L=2 (Figure 7.2-1), we find receiver performances asymptotically approaching two groups in the strong jamming region. These are (1) lower P(e): IC-AGC, ACJ-AGC, clipper, and SNORM; (2) higher P(e): hard-decision (HD), and square-law linear combining receiver (LCR). We observe that the first group is more effective due to their nonlinear weighting (normalization) schemes. In the second group, we have the LCR

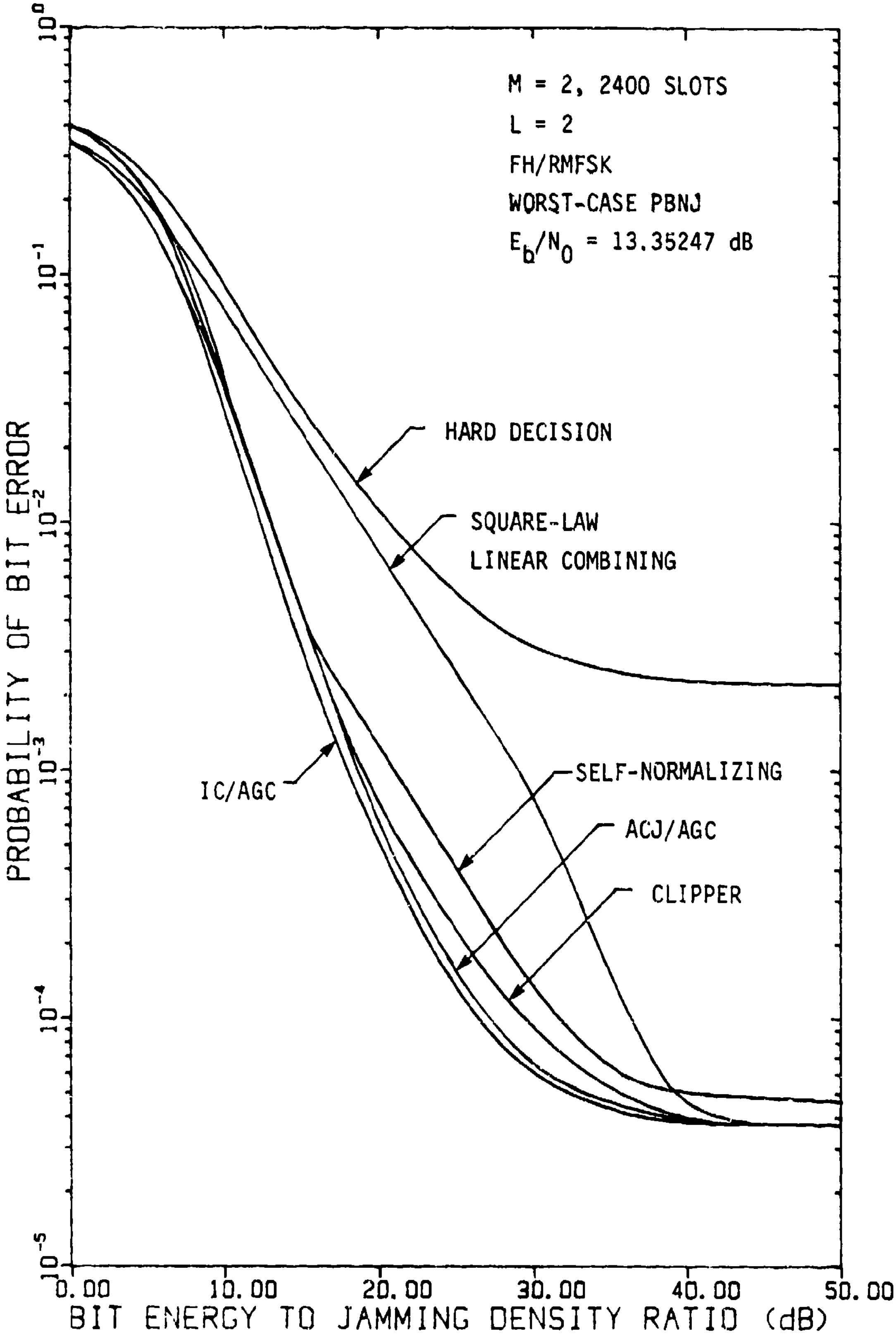


FIGURE 7.2-1 COMPARISON OF DIFFERENT RECEIVERS FOR FH/RMFSK WITH M=2 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 13.35247 dB

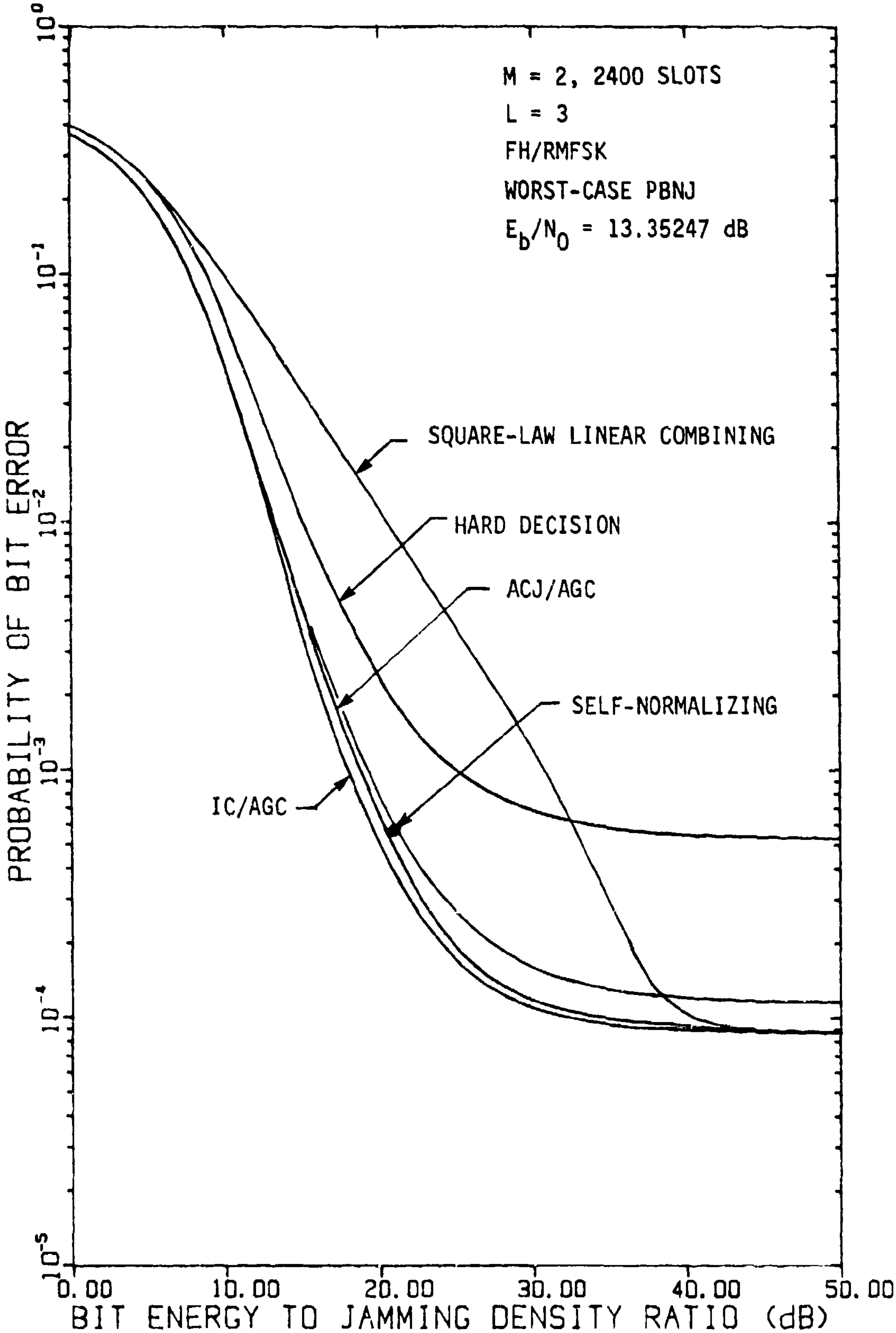


FIGURE 7.2-2 COMPARISON OF DIFFERENT RECEIVERS FOR FH/RMFSK WITH M=2 AND L=3 HOPS/SYMBOL WHEN E_b/N_0 = 13.35247 dB

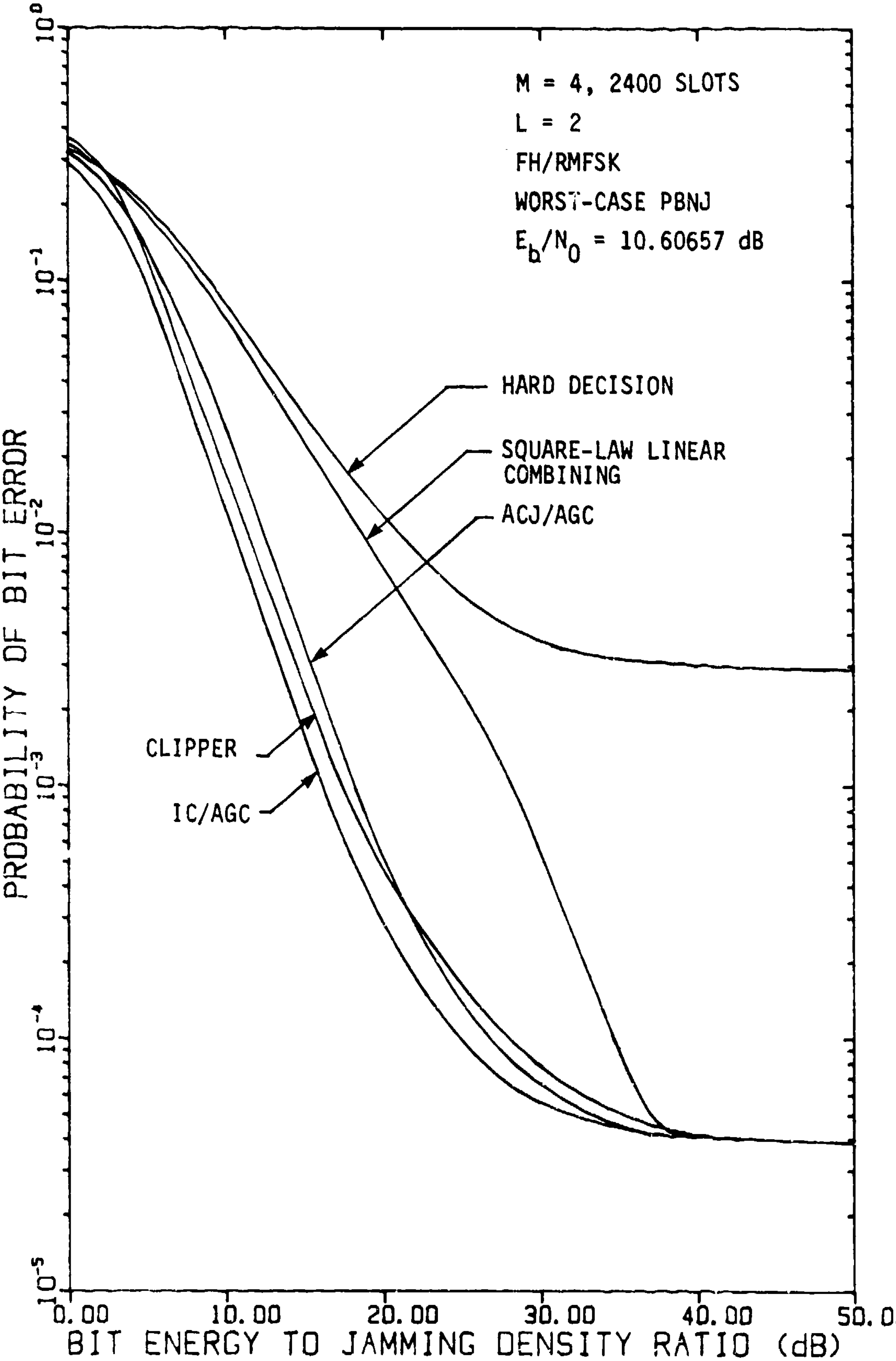


FIGURE 7.2-3 COMPARISON OF DIFFERENT RECEIVERS FOR FH/RMFSK WITH M=4 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 10.60657 dB

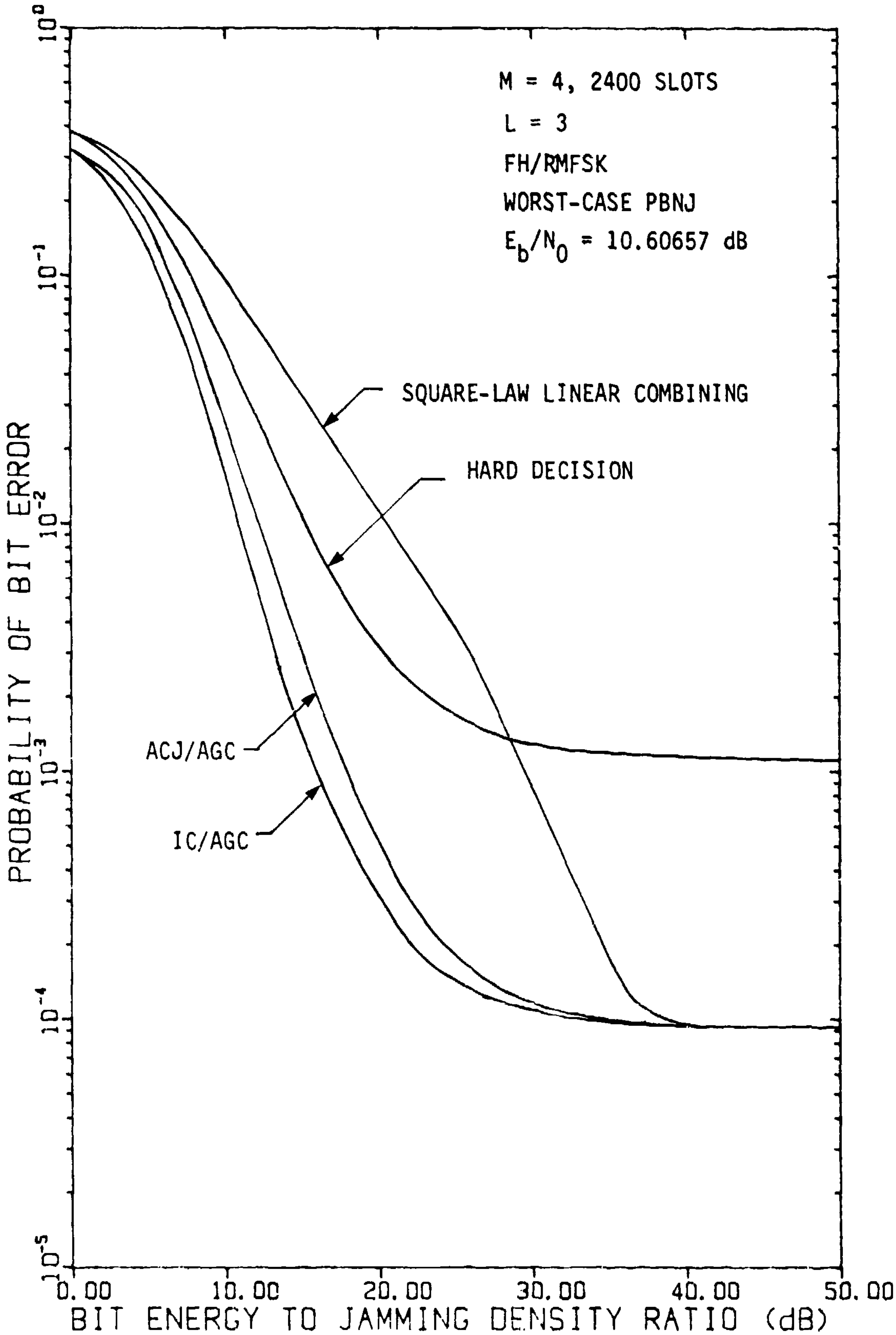


FIGURE 7.2-4 COMPARISON OF DIFFERENT RECEIVERS FOR FH/RMFSK WITH M=4 AND L=3 HOPS/SYMBOL WHEN E_b/N_0 = 10.60657 dB

which provides no weighting and the HD receiver which is ineffective for L=2 because it is subject to the possibility of a tie (for L even) in the final quantized decision variable values. In strong jamming, all receivers are experiencing full-band jamming at high power levels and discriminating against a jammed hop is not done by any receiver since all hops are jammed. It is only when the optimum γ values begin to fall below full-band jamming that we realize the performance improvement of the nonlinear combining techniques. In the moderate jamming region, we see a sub-division among the receivers which we termed "effective" for strong jamming: (a) "ideal" receivers (AGC) and (b) "practical" receivers (clipper, SNORM). In this region we notice performance results for the nonlinear combining types remaining within about 1 dB of each other up until around the point where $E_b/N_J > E_b/N_0$. That is, where thermal noise becomes more dominant than jamming noise. At these values, the SNORM and clipper receiver performances begin to degrade relative to the AGC types, yet still remain superior to the square-law LCR. We note the AGC receivers as maintaining a continuous and graceful transition in this moderate jamming region with the IC-AGC showing a slightly better performance. The worse performance for the ACJ-AGC is due to an imbalancing effect of the ACJ normalization scheme in which <u>all</u> receiver channels are inversely weighted by the largest of all the M channel noise powers. In contrast, the IC-AGC receiver normalizes each channel separately and thereby "balances" or "equalizes" the received noise powers.

The performance "breakaway" for $N_0 > N_J$ for the clipper and SNORM receivers reflects the dominance of the "unbalanced symbol" error mechanism as γ becomes smaller; jamming in more than one M-ary channel becomes unlikely. In the case of the clipper, the receiver tends to limit the unbalancing contribution to the sum without affecting the signal channel sum. But for the SNORM receiver, any input noise power unbalance due to one channel being jammed reduces the signal channel sum. This is because the SNORM normalization weight is inversely proportional to the total noise power (i.e. sum of all M channels) measurement on a given hop without recognition of which individual channels are jammed.

In the weak or no jamming region, all receivers suffer degradation due to the noncoherent combining loss (NCL) when L>1. As E_b/N_J approaches 50 dB (practically no jamming), the different performances of the receivers in the Gaussian channel are evident. All receivers suffer degradation relative to the L=1 result ($P(e) = 10^{-5}$) due to the NCL as Figure 7.2-1 demonstrates. We see that the SNORM and HD receivers are subject to higher NCL due, respectively, to inefficient combining and to the possibility of "tie votes" for M=2 and L=2, with the HD receiver being more severely affected because of its use of only two levels of quantization in the soft-decision.

Figure 7.2-2 compares receiver performances for the parameter set M=2, L=3. For strong jamming, we see a change in two groupings recognized for the case M=2, L=2. The HD receiver is now more effective than the square-law LCR, and provides a significant improvement in strong jamming. We attribute this to the fact that no ties exist in the majority logic decoding when L is odd. But in the weak or no jamming region the HD performance is worse than

LCR due to quantization noise effects. We likewise note that in moderate jamming, the SNORM performance has improved somewhat over the results for M=2, L=2. It is reasoned that for L=3, the multiple unbalancing effects predominant for decreasing γ values begin to become less probable. Performance rankings in the thermal-noise-limited region for SNORM remain unchanged from the case of M=2, L=2.

The effect of an increase in alphabet size (M=4) can be discerned from Figures 7.2-3 and 7.2-4. We notice that the parameter set M=4, L=2 (Figure 7.2-3) appears similar to the M=2, L=2 set where two distinct groupings are present in moderate jamming with the nonlinear combining soft-decision receiver group yielding superior performance. Considering the moderate jamming region to be from E_b/N_J = 5 to 39 dB, we observe that the clipper and ACJ-AGC receivers trade rankings around the regional midpoint of 22 dB; the clipper receiver showing $\leq$ 1 dB better performance over the range of 5 to 17 dB. However, in strong jamming we find the clipper's performance degrading to the point of being the overall worst performer at E_b/N_J = 0 dB. For the HD receiver, we find a worse performance than for M=2, L=2 because there are now two more channels allowing for the possibility of more tie decisions on the output decision variables.

In the case of M=4, L=3 (Figure 7.2-4), receiver performances appear similar to the behavior exhibited by the parameter set M=2, L=3 in that two distinguishable groups are presented. These are the AGC types (better performances) versus the square-law LCR and HD receivers. Throughout most of the strong and moderate jamming regions (2 to 35 dB), it is noticed that the ACJ-AGC performance is up to 2 dB worse than the IC-AGC; this again being due to the unbalancing effect of the normalization weighting scheme of the ACJ-AGC receiver. The difference between the two AGC performances for M=4 is larger than for M=2 (see Section 7.3 for more discussion of this phenomenon). As for the HD receiver, it proves to be better than the HD cases for L=2 yet exhibits poorer performance than HD for M=2, L=3. Although there are no output decision variable ties for L=3, the HD receiver with more channels will now suffer increased quantization effects in approximating the LCR.

With regard to receiver performances in little thermal noise, Figure 7.2-5 depicts performance results of three candidates (IC-AGC, ACJ-AGC, SNORM) for the parameter set M=2, L=2 at E_b/N_0 = 20 dB. These receivers represent the previously shown most ideal performers (AGC types) and the more realizable SNORM receiver. The HD receiver, although a relatively simple ECCM diversity technique in practice, is not included in this comparative set due to the "tie" decision factor when L=2.

We observe for full-band jamming that the IC-AGC and ACJ-AGC receivers yield equivalent performances with the SNORM showing a slightly higher BER. Such behavior for the AGC receivers is to be expected in full-band jamming where a Gaussian channel performance is realized and discrimination against a jammed hop is nonexistent. But as γ becomes less than full-band, it is seen that the AGC receiver curves maintain the same negative slope for increasing E_b/N_J with the ACJ-AGC staying about 1 dB worse than the IC-AGC, this inferior performance being due to the previously described imbalancing effect of the ACJ-AGC normalization mechanism. For the SNORM receiver, we see its performance with respect to the AGC receivers as being: (1) about 0.5 dB

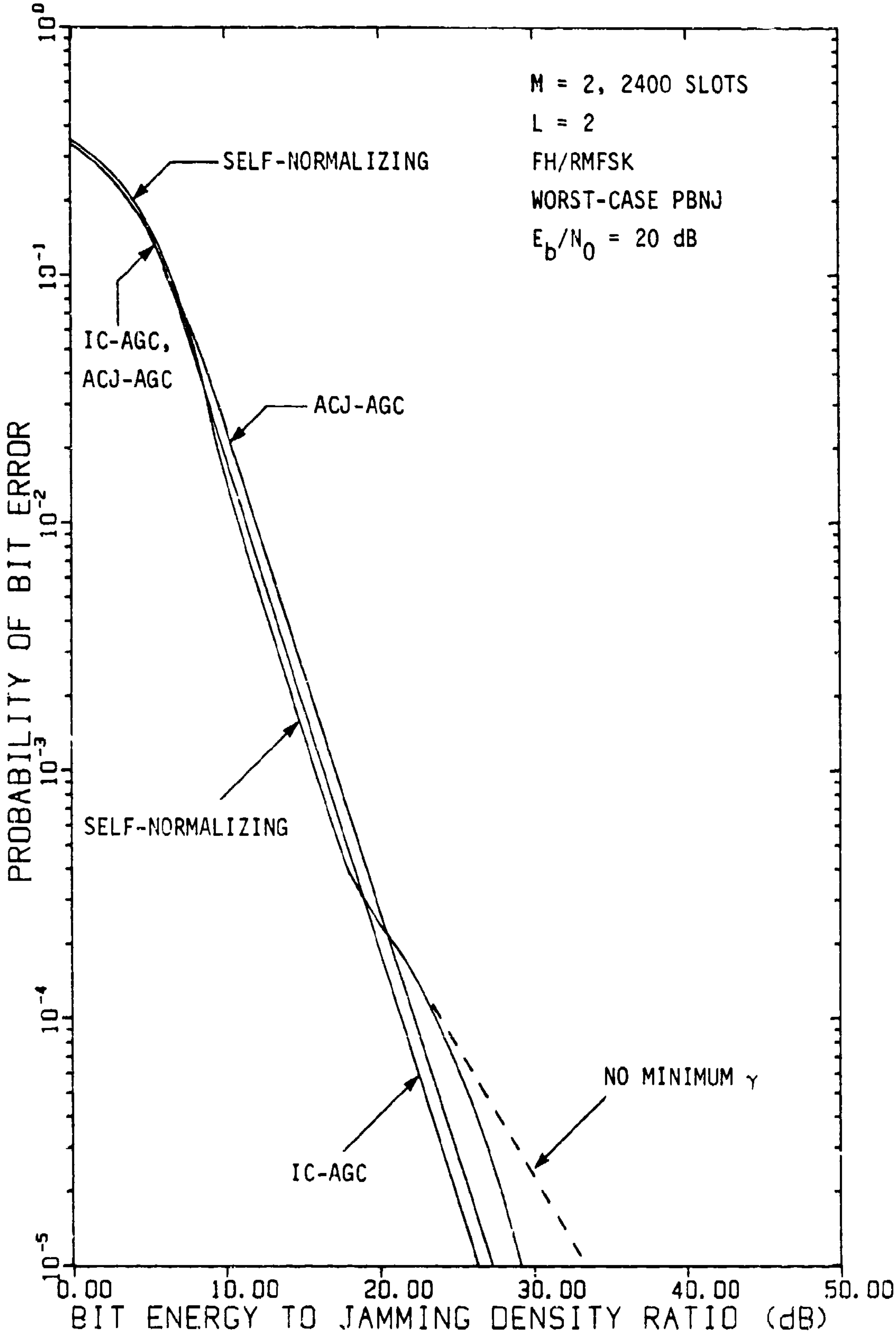


FIGURE 7.2-5 COMPARISON OF DIFFERENT RECEIVERS FOR FH/RMFSK WITH M=2 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 20 dB

worse in full-band jamming, (2) less than 1 dB better for $\gamma<1$ (at E_b/N_J ~ 10 dB) up until $E_b/N_J > E_b/N_0$, and (3) more than either ACJ-AGC or IC-AGC from about 20 dB to the point where $P(e) = 10^{-5}$ is reached.

An empirical explanation of this phenomenon is obtained by comparing Figure 7.2-6 (IC-AGC) with Figure 7.2-7 (SNORM). These figures show each receiver's individual performance at E_b/N_0 = 20 dB for ten different values of γ ranging from γ=0.001 to γ=1.0 or full-band jamming. We note for the AGC receiver (Figure 7.2-6) that γ=0.001 and 0.002 curves produce $P(e) < 10^{-5}$ and thus do not appear on the performance plots. Upon observing the eight remaining γ-curves in these AGC plots, we see each of these P(e) curves contributing the same smooth behavior toward producing an optimum γ-curve result which is a straight line for γ < 1.0, that is, a slope equal to $A/(E_b/N_J)^2$ where A is some constant defining the inverse-linear relationship existing between γ and available jamming power when E_b/N_0 = 20 dB.

However, for the SNORM case we find performance curves for γ=0.005 through γ=0.2 exhibiting behavior resulting in an optimum γ-curve which is not constant; over these γ ranges the SNORM performance is superior to the IC-AGC receiver. The upper envelope of the curves at first is proportional to $(E_b/N_J)^{-2}$, then transitions to a dependence on $(E_b/N_J)^{-1}$. For infinite E_b/N_0, from [21] we expect the SNORM and IC-AGC BER's to be proportional to $(E_b/N_J)^{-2}$ indefinitely; for L=3 and MFSK the SNORM in [21] is shown to be dependent on $(E_b/N_J)^{-2}$ for no thermal noise, but the AGC is dependent on $(E_b/N_J)^{-3}$, as shown in Figure 7.2-8, taken from [21]. Therefore, the better performance of SNORM for high SNR is not to be expected in all cases of M and L. (See Section 7.3.3.5 for further discussion on the SNORM performance.)

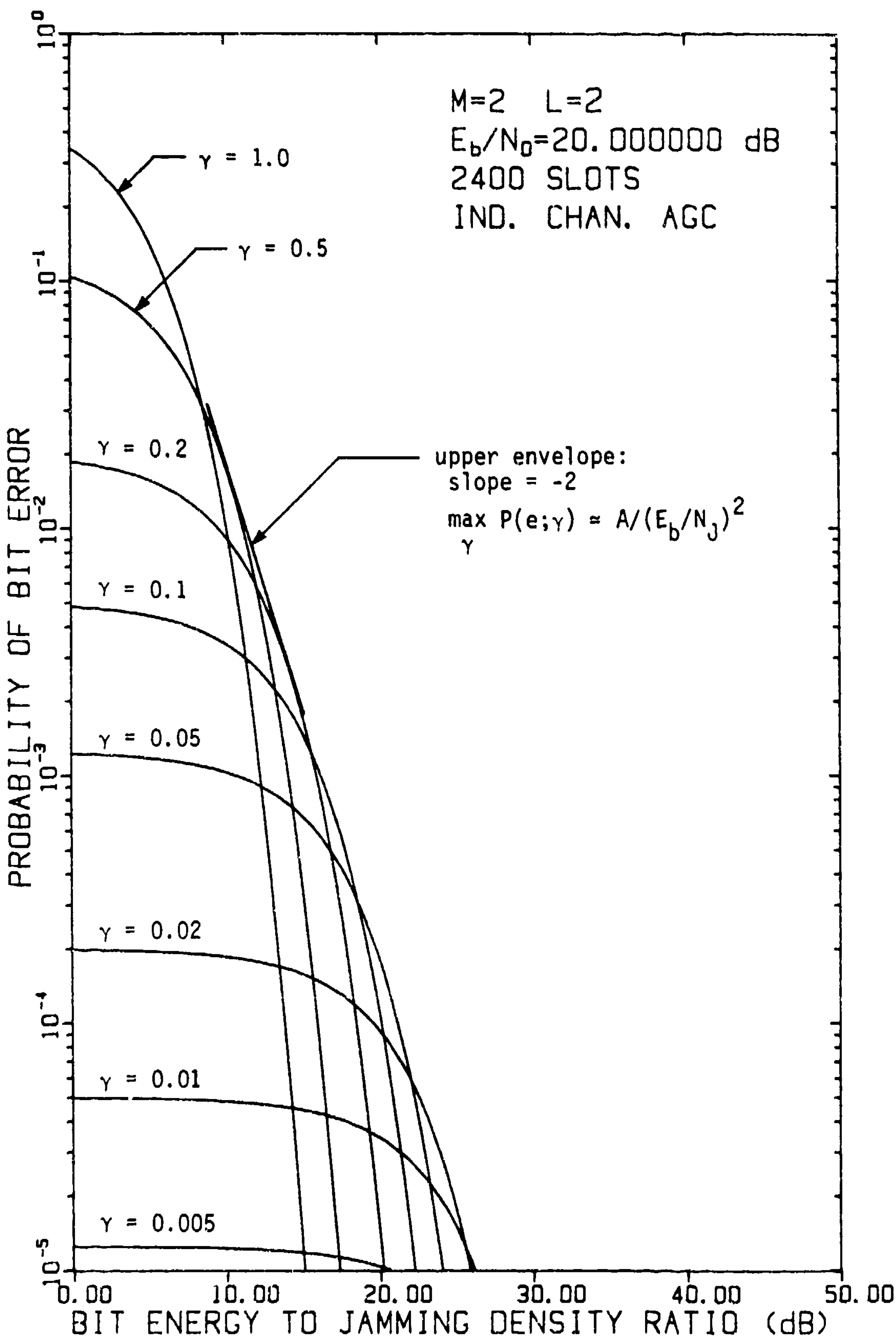


FIGURE 7.2-6 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR IC-AGC RECEIVER IN WORST-CASE PARTIAL-BAND NOISE FOR M=2, L=2 AT E_b/N_0 = 20 dB

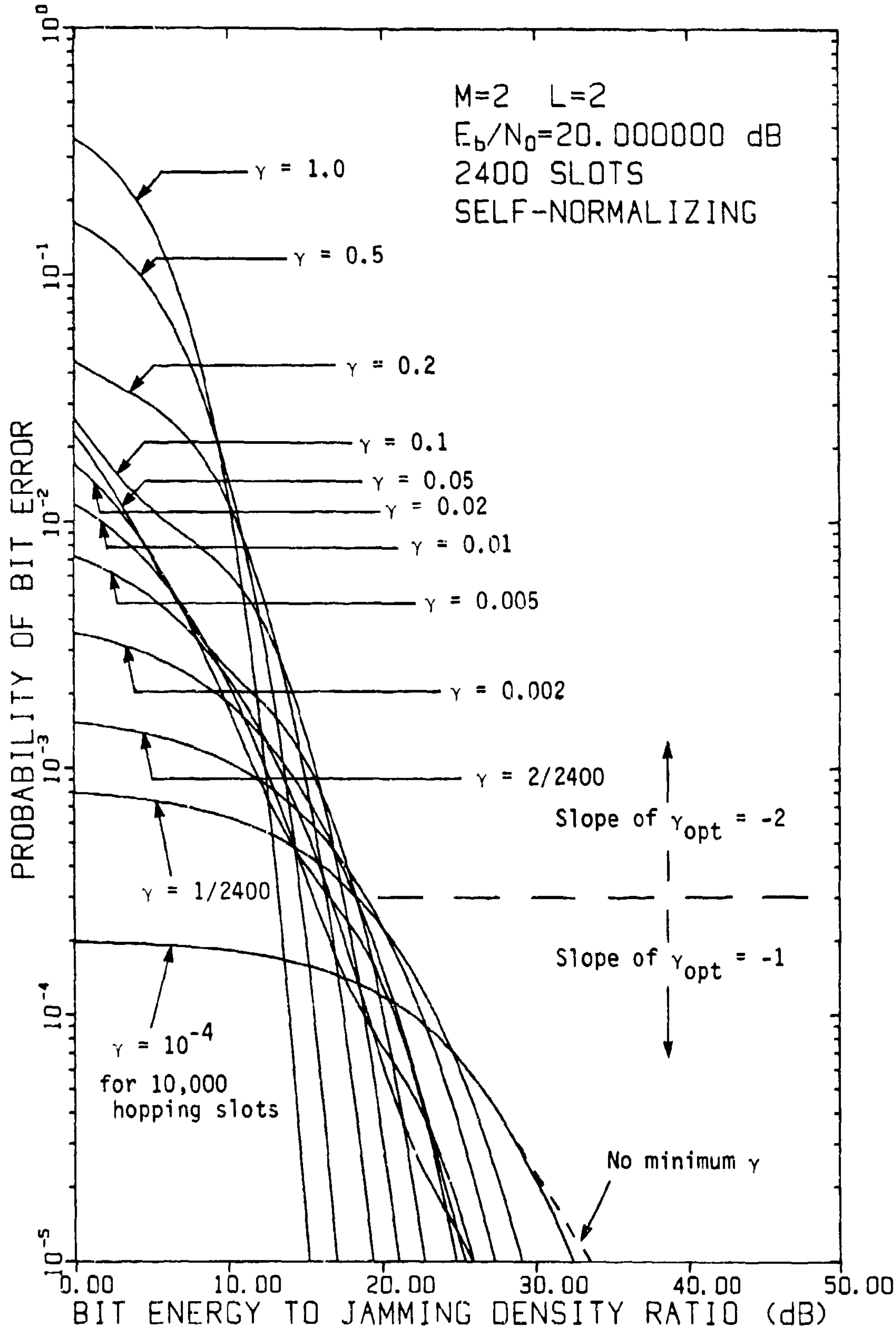


FIGURE 7.2-7 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR SELF-NORMALIZING RECEIVER IN WORST-CASE PARTIAL-BAND NOISE JAMMING FOR M=2, L=2 AT E_b/N_0 = 20 dB

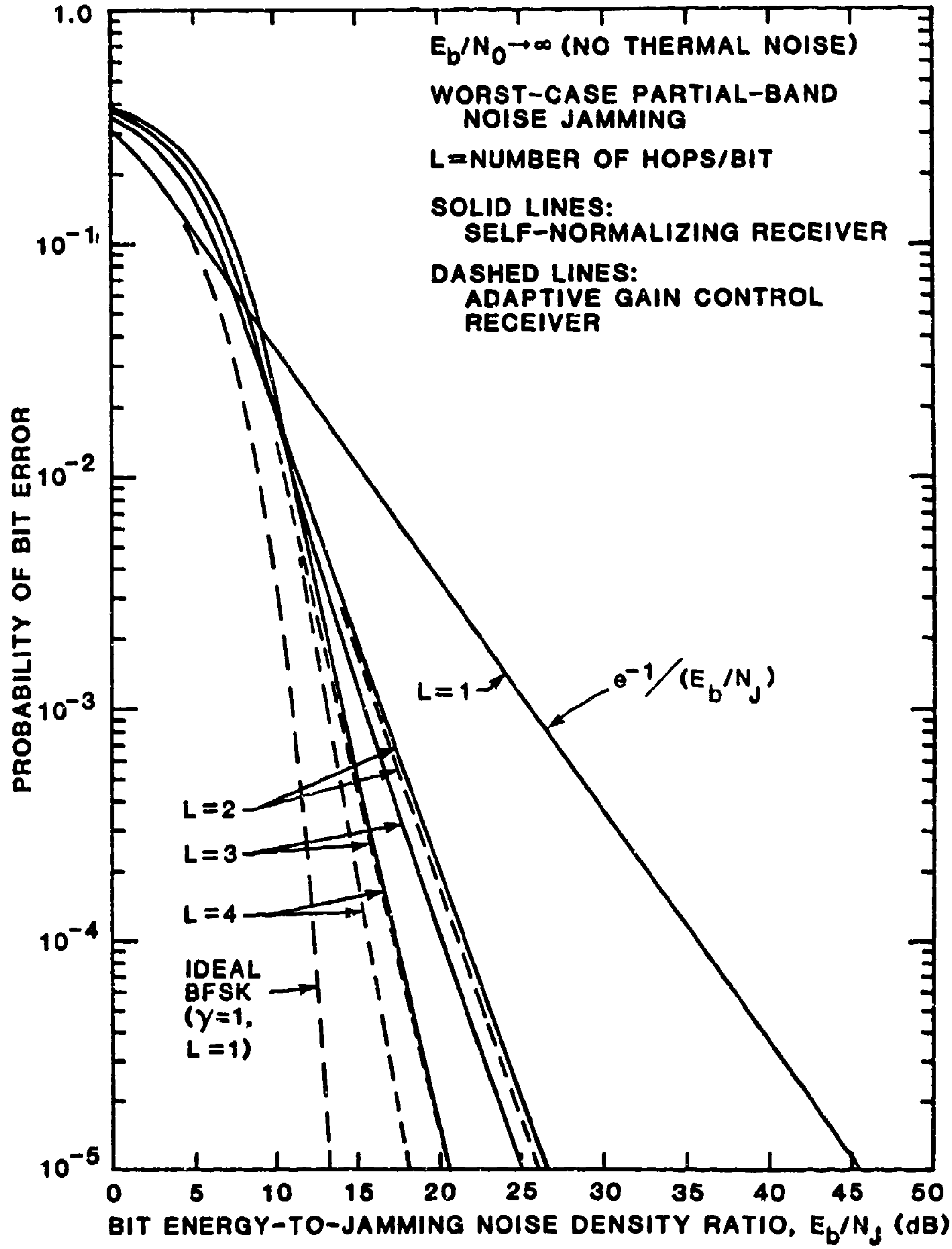


FIGURE 7.2-8 WORST-CASE PARTIAL-BAND NOISE JAMMING PERFORMANCE OF THE SELF-NORMALIZING FH/BFSK RECEIVER WHEN THERMAL NOISE IS ABSENT, WITH THE NUMBER OF HOPS/BIT (L) AS A PARAMETER

7.3 COMPARISONS OF FH/RMFSK AND FH/MFSK

Having compared the performances of the various FH/RMFSK receivers in worst-case partial-band noise jamming (WCPBNJ), we now consider the differences in performance to be expected between the random hopping MFSK system studied in this report (FH/RMFSK) and the conventional (adjacent or contiguous) hopping MFSK system (FH/MFSK) studied, for example, in [1].

7.3.1 The M=2, L=1 Case

It was found by Blanchard [6] that for M=2 and L=1 the two systems yield the same performance, at least for the E_b/N_0 = 30 dB case he studied, with the differences in possible jamming events accounted for by different optimum values of γ, the fraction of the system bandwidth which is jammed. (Typically, for high E_b/N_J the RMFSK γ_{opt} was found to be half that for MFSK.) Figure 7.3-1 displays the cases Blanchard considered, except that we use E_b/N_0 = 13.35 dB, corresponding to a 10^{-5} BER with no jamming. Our results indicate that the two systems do indeed perform the same for M=2 and L=1, except for certain differences for weak jamming (high E_b/N_J). What is the significance of the differences we observe in these computed results?

Curve A in Figure 7.3-1 is from [1] and represents the quantity

$$\max_{0<\gamma\leq 1}\left[\frac{1}{2}(1-\gamma)e^{-E_b/2N_0} + \frac{1}{2}\gamma\, e^{-E_b/2N_T}\right], \tag{7.3-1a}$$

where

$$\frac{E_b}{N_T} = \frac{\dfrac{E_b}{N_0}\cdot\dfrac{\gamma E_b}{N_J}}{\dfrac{E_b}{N_0}+\dfrac{\gamma E_b}{N_J}}\ . \tag{7.3-1b}$$

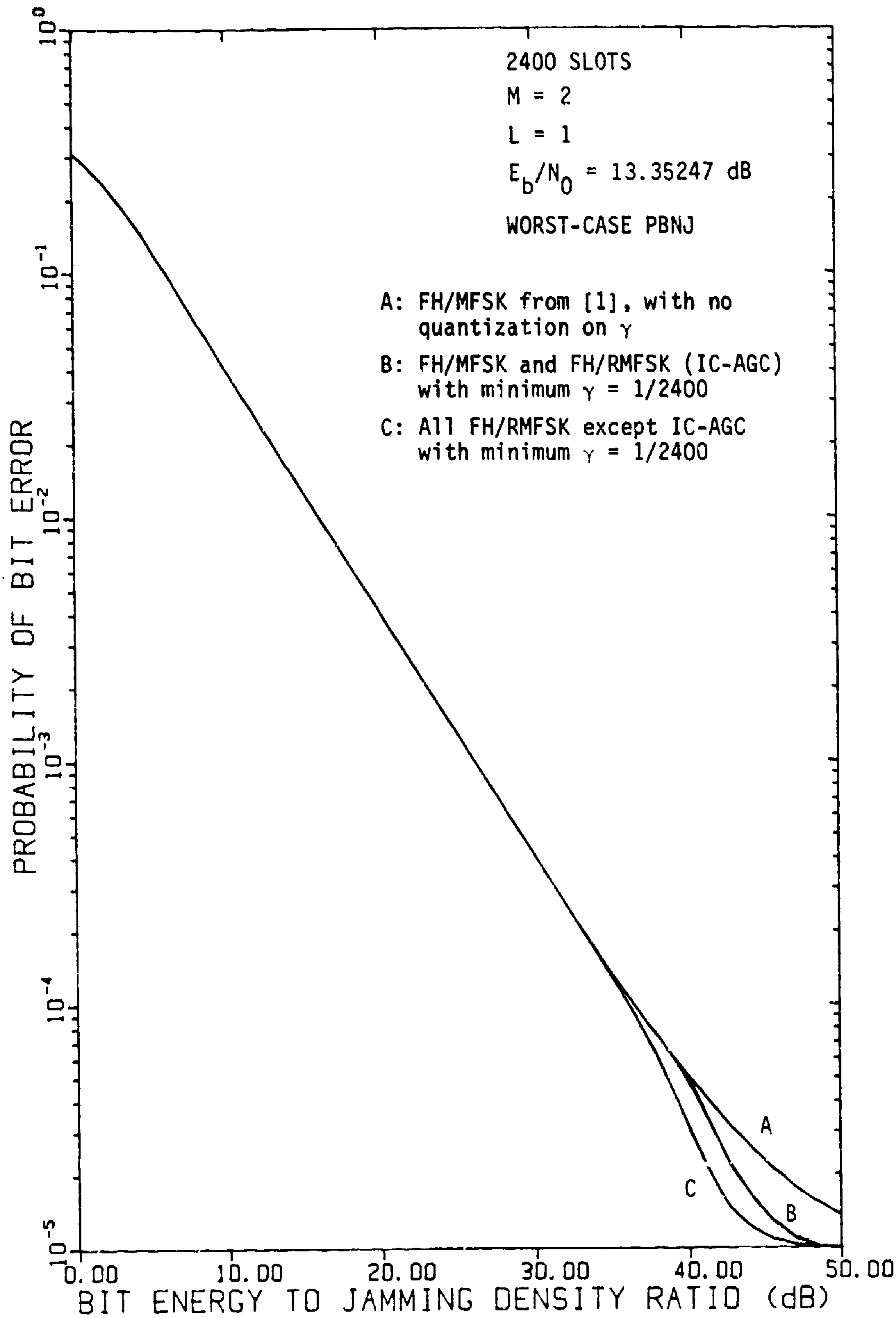


FIGURE 7.3-1 COMPARISON OF PERFORMANCE OF ALL RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=2 AND L=1 WHEN E_b/N_0 = 13.35247 dB

In this formulation for the binary case it is assumed that both slots are either jammed (with probability γ) or unjammed (with probability $1-\gamma$).

Curve B represents the quantity

$$\max_{1\le q\le N}\left[\frac{1}{2}\,p_0 e^{-E_b/2N_0} + \frac{1}{2}\,p_1 e^{-E_b/2N_T}\right], \tag{7.3-2a}$$

where

$$p_1 = 1 - q/N = 1 - p_0 \;. \tag{7.3-2b}$$

This formulation assumes that $\gamma = q/N$ is quantized - a discrete number (q) of the total number slots (N) are jammed, with the minimum γ equal to 1/N. It further assumes that the system is FH/RMFSK with IC-AGC processing; a more explicit form of the error expression is

$$P(e;q) = (\pi_0+\pi_1)\cdot\frac{1}{2}\,e^{-E_b/2N_0} + (\pi_1+\pi_2)\cdot\frac{1}{2}\,e^{-E_b/2N_T}\;, \tag{7.3-3}$$

where

$$\pi_r = \text{prob. that } r \text{ slots are jammed}, \tag{7.3-4a}$$

and

$$\pi_0 = \frac{N-q}{N}\cdot\frac{N-q-1}{N-1} \tag{7.3-4b}$$

$$\pi_1 = \frac{N-q}{N}\cdot\frac{q}{N-1} \tag{7.3-4c}$$

$$\pi_2 = \frac{q}{N}\cdot\frac{q-1}{N-1}\;. \tag{7.3-4d}$$

Because of the individual channel normalization, the conditional BER depends only on whether the signal channel is jammed. Thus here are only two terms, with weights $p_0 = \pi_0+\pi_1$ and $p_1 = \pi_1+\pi_2$.

Now, the only difference between (7.3-2) and (7.3-1) is the quantization and minimum value of $\gamma = \frac{q}{N}$. Therefore in Figure 7.3-1 we identify curve B also with binary FH/MFSK, even though q = 1 violates the assumption that both channels are together jammed or unjammed.

Curve C in Figure 7.3-1 represents the quantity

$$\max_{1 \leq q \leq N} \left[\pi_0 \cdot \frac{1}{2} e^{-E_b/2N_0} + \pi_1 \cdot e^{-(E_b/N_0)/(K+1)} + \pi_2 \cdot \frac{1}{2} e^{-E_b/2N_T}\right], \qquad (7.3\text{-}5a)$$

where

$$K = \sigma_T^2/\sigma_N^2 = (E_b/N_0)/(E_b/N_T). \qquad (7.3\text{-}5b)$$

This is the BER for all the FH/RMFSK receivers except the IC-AGC, and allows for only one of the two channels to be jammed.

Thus, in general, our results agree with Blanchard's conclusion that FH/RMFSK performs the same as FH/MFSK for M=2 and L=1, neglecting small asymptotic differences connected with assumptions on the quantization and minimum value of γ. Now we consider whether his conjecture that the two hopping systems perform the same for M>2 and L=1 is correct, and how the comparison is affected by L>1. In what follows, we shall use the fact that the IC-AGC FH/RMFSK receiver performs essentially the same as the AGC FH/MFSK receiver.

7.3.2 L=1 with Alphabet Size Varied.

In order to compare RMFSK and MFSK for L=1 and M>1, it is sufficient to consider Figures 7.3-2 and 7.3-3.

In Figure 7.3-2 the performances of the FH/MFSK and the IC-AGC FH/RMFSK receivers are shown for L=1 and M=2,4,8. The values of E_b/N_0 used were chosen

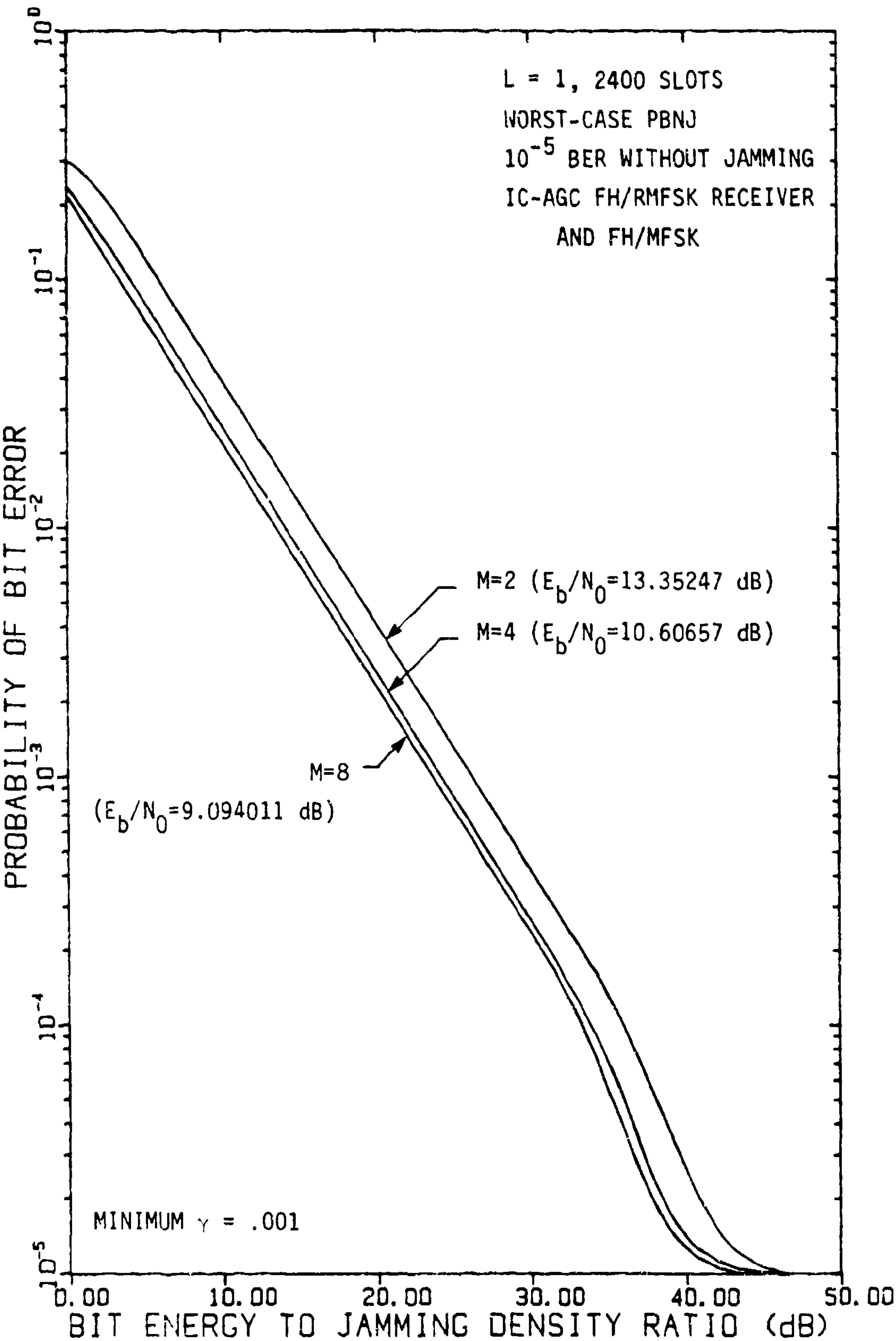


FIGURE 7.3-2 PERFORMANCE OF INDIVIDUAL CHANNEL AGC RECEIVER FOR FH/RMFSK WHEN L=1 HOP/SYMBOL WITH M AS A PARAMETER AND E_b/N_0 CORRESPONDING TO $P_b(e) = 10^{-5}$ IN THE ABSENCE OF JAMMING.

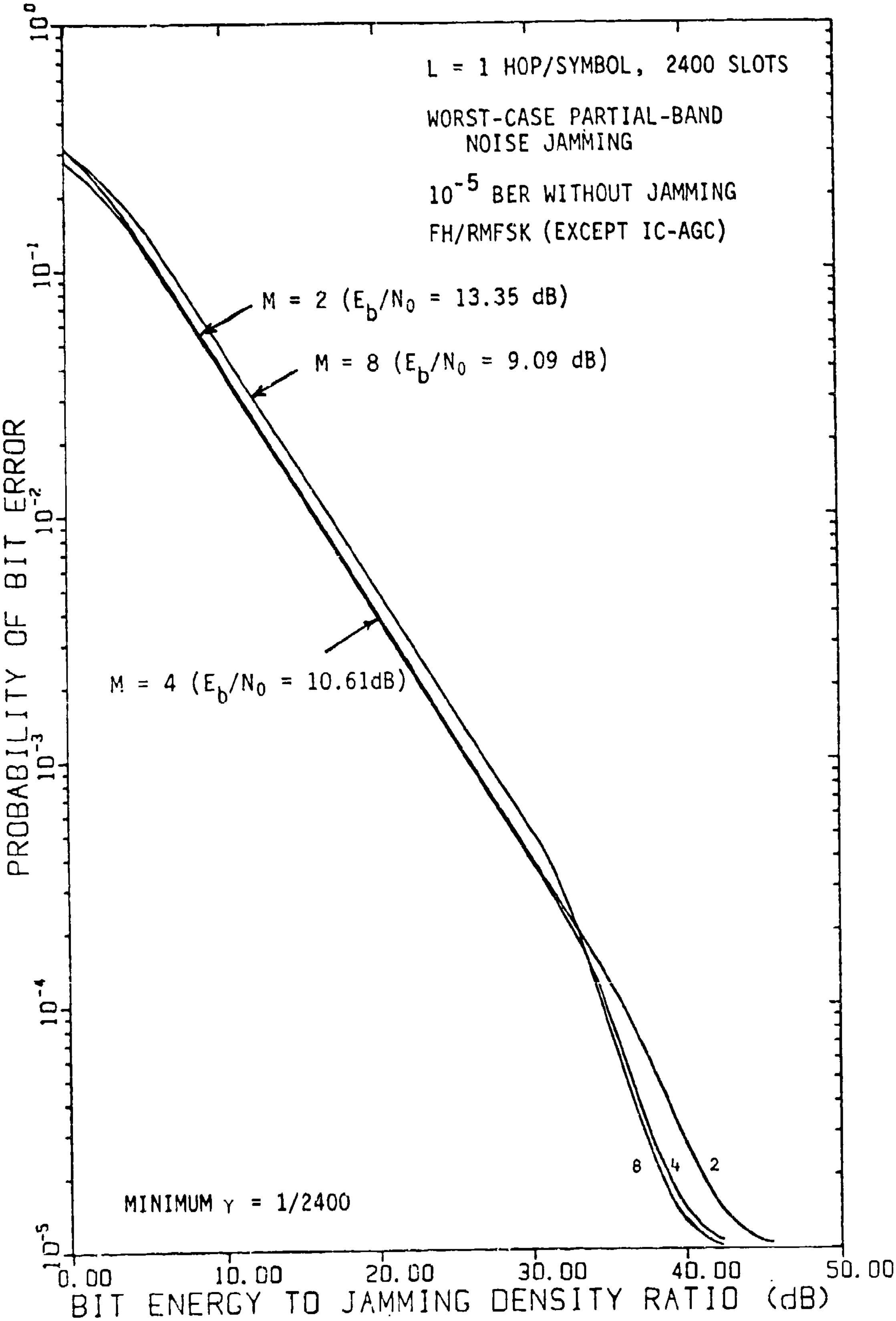


FIGURE 7.3-3 WORST-CASE PARTIAL-BAND NOISE JAMMING PERFORMANCE OF FH/RMFSK RECEIVERS FOR L = 1 HOP/SYMBOL AND M = 2, 4, 8 WHEN E_b/N_0 GIVES A 10^{-5} BER WITHOUT JAMMING

to give each example a 10^{-5} BER under no jamming. We observe from these results that for these systems the BER decreases as the alphabet size M increases, the conventional interpretation of which is that an "M-ary coding gain" is at work. This is the phenomenon usually observed for MFSK systems in the Gaussian interference channel.

In Figure 7.3-3 the same parameters are used as in Figure 7.3-2, but now the receivers are the FH/RMFSK receivers (except IC-AGC), which have identical performance for L=1. For these receivers we find that for strong jamming the system performance does not consistently improve as M increases, but instead improves very slightly for M=4 and degrades for M=8. Clearly this is the result of the increased probability, as M increases, of the most damaging jamming event: jamming power in a non-signal slot but not in the signal slot. Since the M=2 performances in the two figures are virtually the same, we conclude that FH/RMFSK is consistently more vulnerable to WCPBNJ than is FH/MFSK for M>2. The difference is about 3 dB for M=4 and 5 to 6 dB for M=8.

When the jamming is weak, we expect the relative performances for different M to approach the usual non-jammed behavior, and this is observable in Figure 7.3-3 for $E_b/N_J > 34$ dB.

7.3.3 Cases Where L>1 Hop/Symbol

Since the various FH/RMFSK receivers and their FH/MFSK counterparts begin to exhibit different performances when diversity is used (L>1), it is necessary to consider them separately. Most of the FH/MFSK results are taken from [1]; however, when convenient we shall continue to utilize the fact that IC-AGC FH/RMFSK receiver performs essentially the same as the AGC FH/MFSK receiver.

7.3.3.1 Linear Combining Receiver.

We begin by comparing the performance of the square-law linear combining receiver for both FH/RMFSK and RH/MFSK signalling strategies. Figures 7.3-4 and 7.3-5 show these performances for M=2, L=2 and M=4, L=2 respectively. In both figures, it is apparent that MFSK is superior to RMFSK, ignoring the effects of the different minimum γ value used in the computations. This vulnerability of RMFSK can be attributed to what we term the "unbalancing" error mechanism inherent in partial-band jamming of RMFSK. Specifically, the random placement of M-ary slots over the hopping bandwidth W allows more chance for a jamming hit than does a block-hopping MFSK signal where it is assumed that either all M slots will be jammed or unjammed. This probability increases for greater values of M as evidenced by Figure 7.3-5.

7.3.3.2 AGC receivers.

Comparisons for L>1 among the AGC-type nonlinear combining receiers are exhibited in Figures 7.3-6 to 7.3-9. Recalling that RMFSK and MFSK hopping systems perform virtually the same for M=2 and L=1, it is instructive to observe in Figures 7.3-6 and 7.3-7 that the system performances differ by about 1 dB when L=2 or L=3. The approximately 3 dB difference noted for M=4 and L=1 continues to hold for M=4 and L=2 or 3, as shown in Figures 7.3-8 and 7.3-9.

7.3.3.3 Clipper receiver.

A very interesting consideration is brought to light by Figure 7.3-1 which shows the clipper receiver's FH/RMFSK performance for L=2 and several values of M. It was found analytically in Section 5 that the optimum clipping

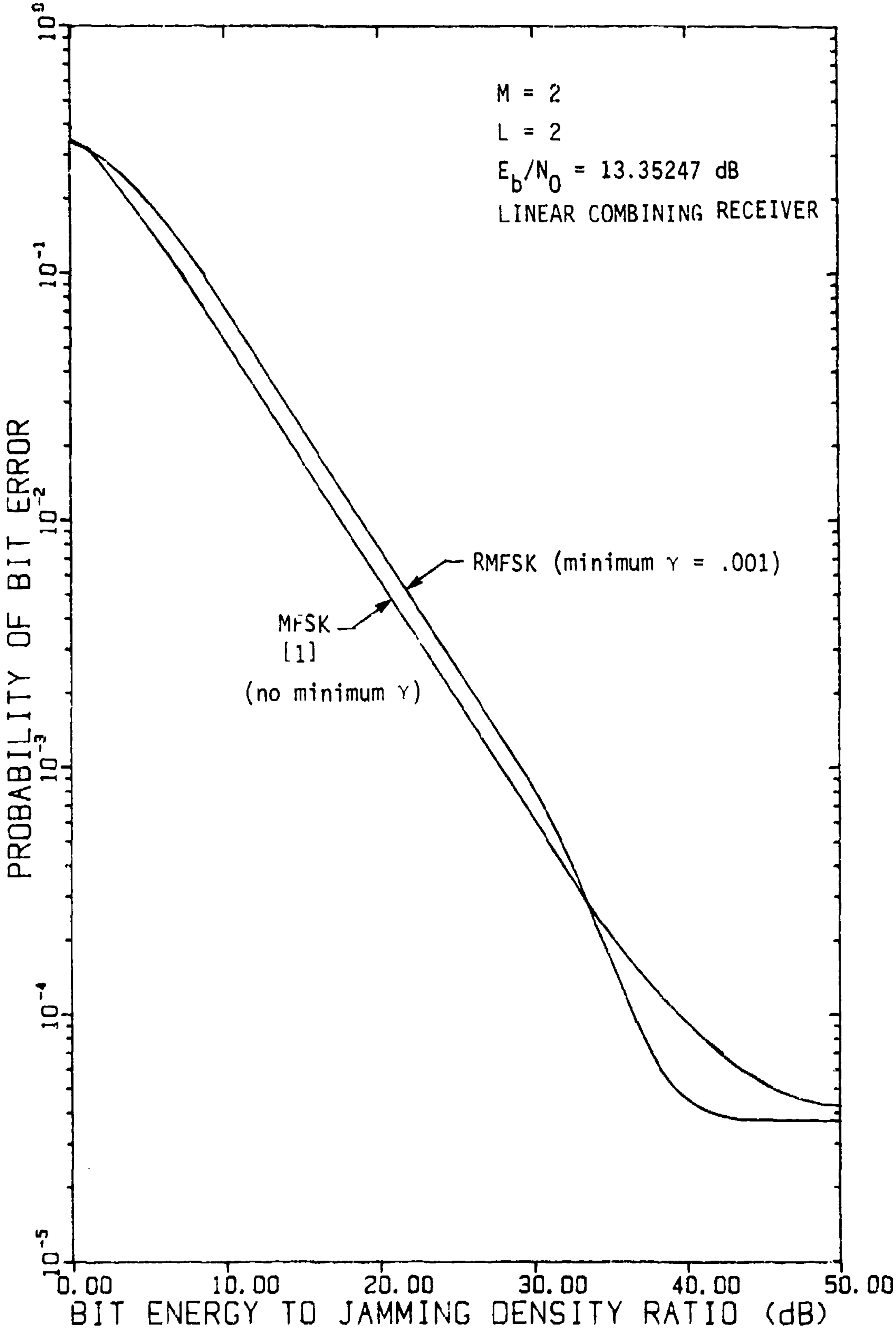


FIGURE 7.3-4 COMPARISON OF SQUARE-LAW LINEAR COMBINING RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=2 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 13.35247 dB

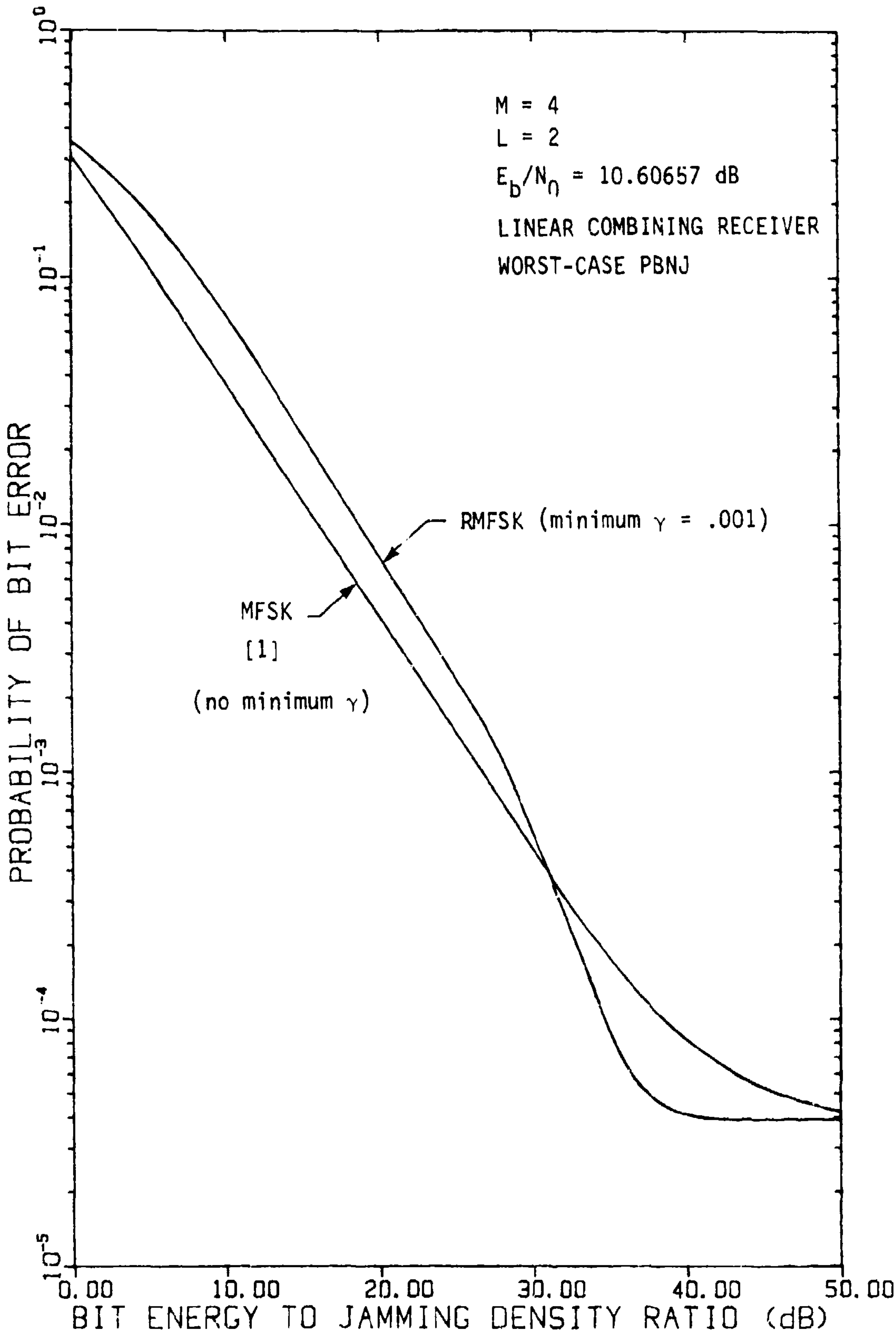


FIGURE 7.3-5 COMPARISON OF SQUARE-LAW LINEAR COMBINING RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=4 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 10.60657 dB

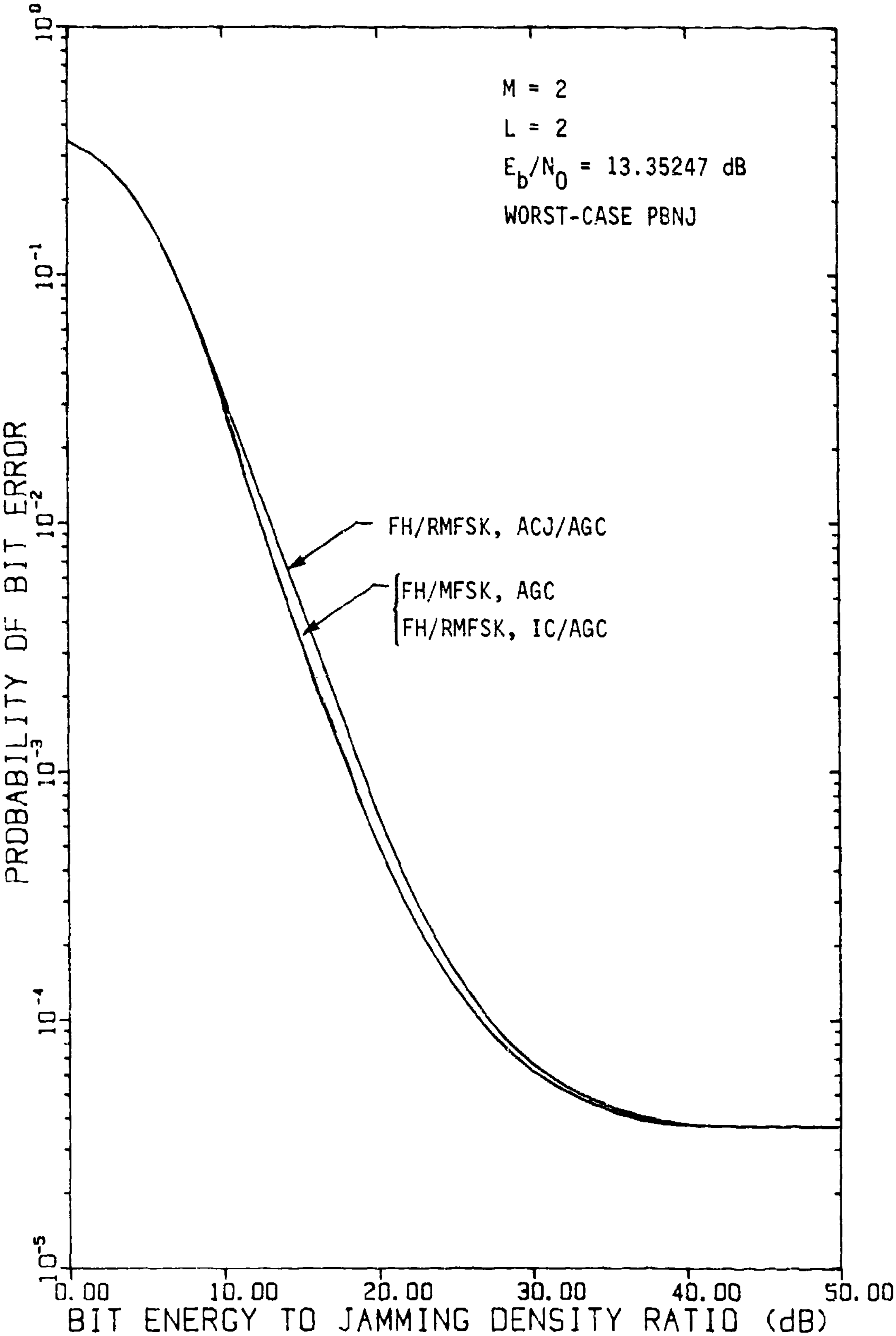


FIGURE 7.3-6 COMPARISON OF AGC RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=2 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 13.35247 dB

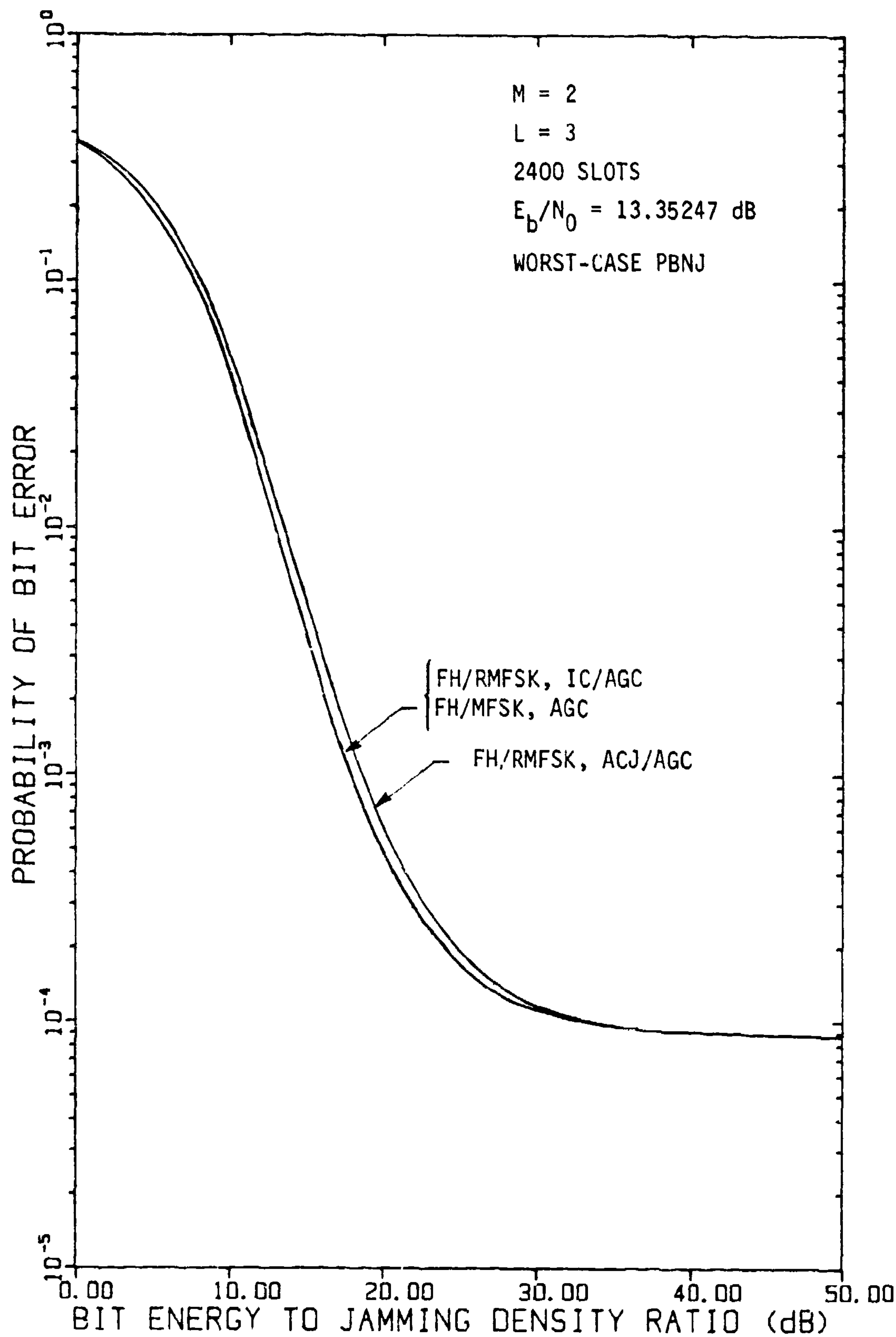


FIGURE 7.3-7 COMPARISON OF AGC RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=2 AND L=3 HOPS/SYMBOL WHEN E_b/N_0 = 13.35247 dB

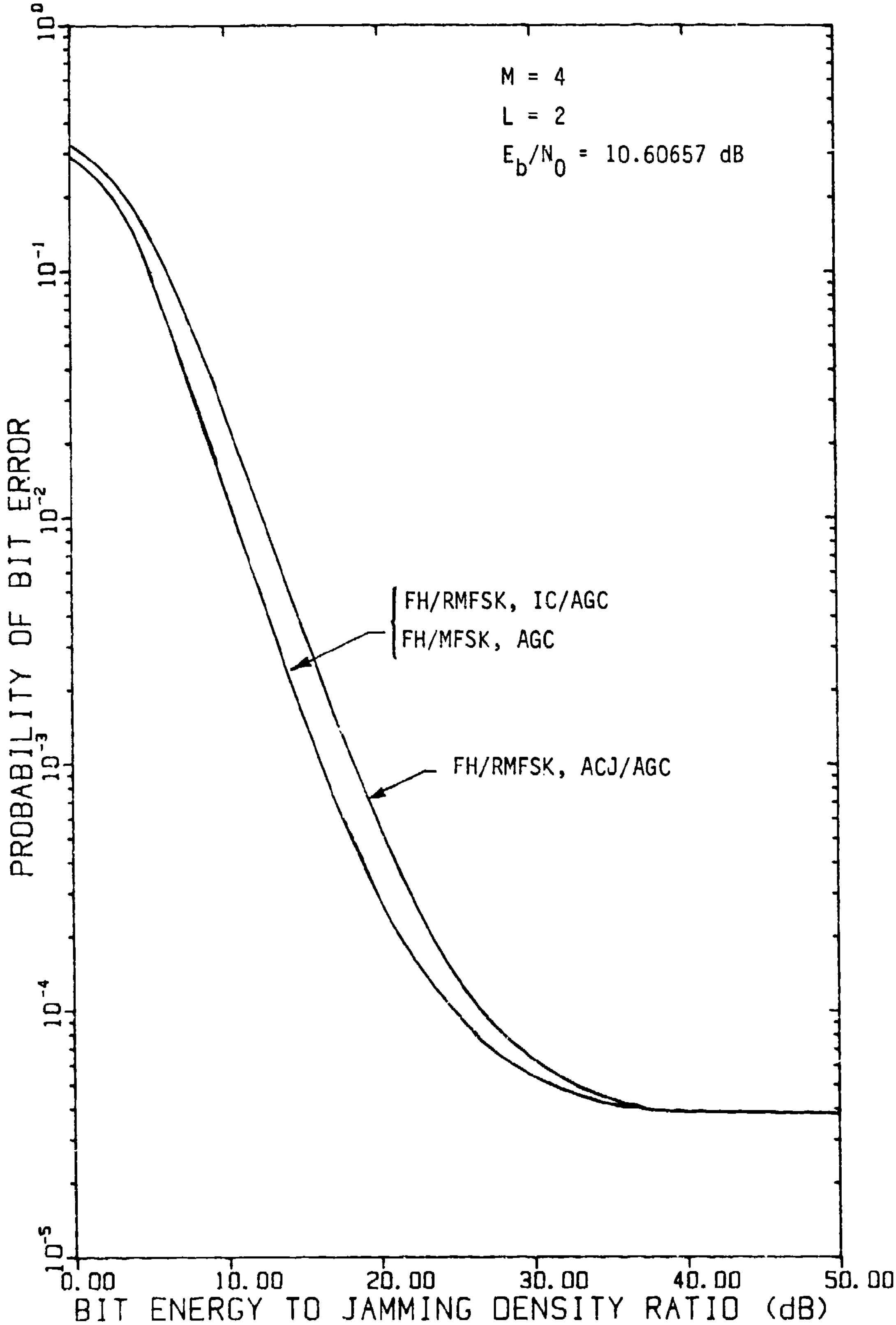


FIGURE 7.3-8 COMPARISON OF AGC RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=4 AND L=2 HOPS/SYMBOL WHEN E_b/N_0 = 10.60657 dB

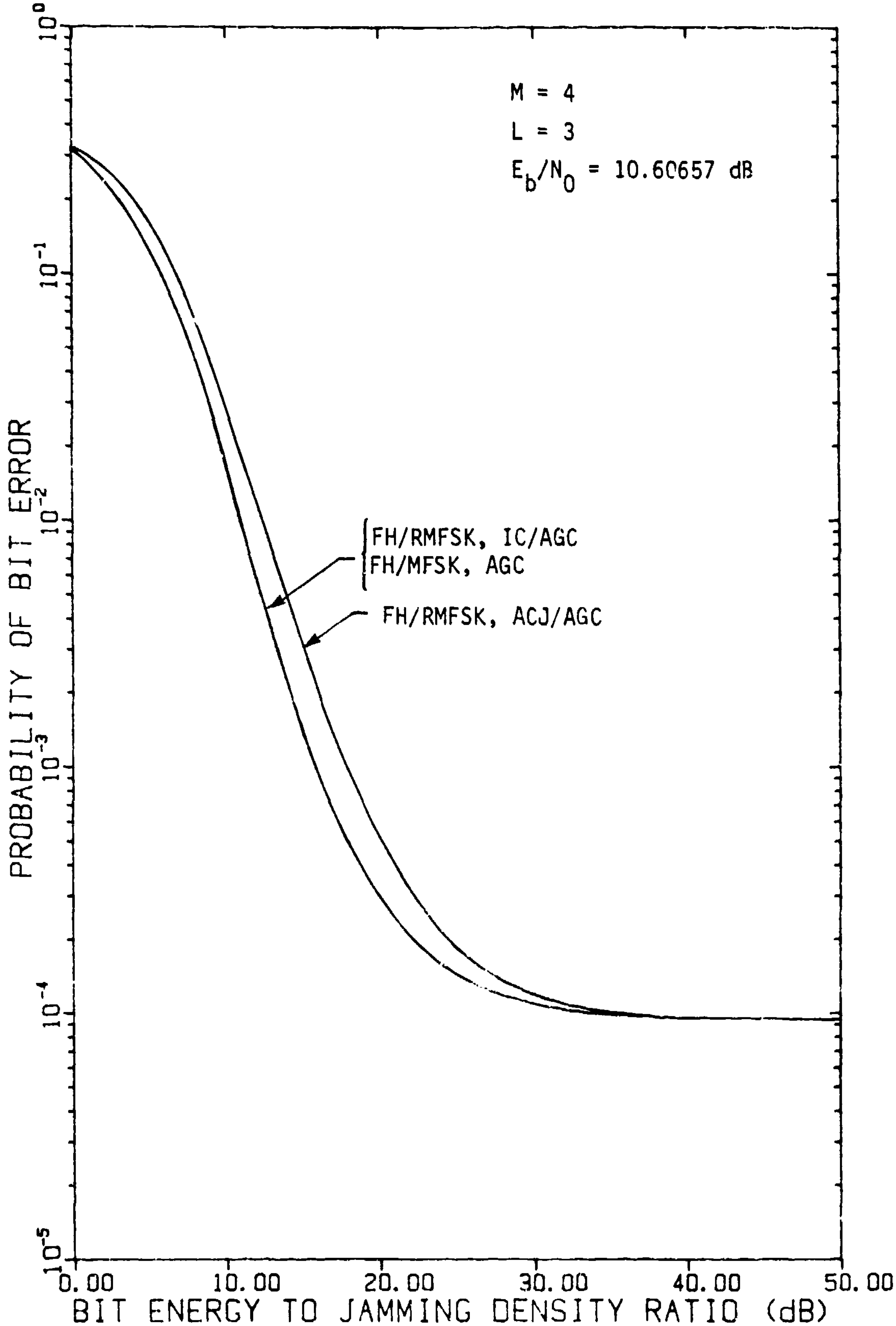


FIGURE 7.3-9 COMPARISON OF AGC RECEIVERS FOR FH/MFSK AND FH/RMFSK WITH M=4 AND L=3 HOPS/SYMBOL WHEN E_b/N_0 = 10.60657 dB

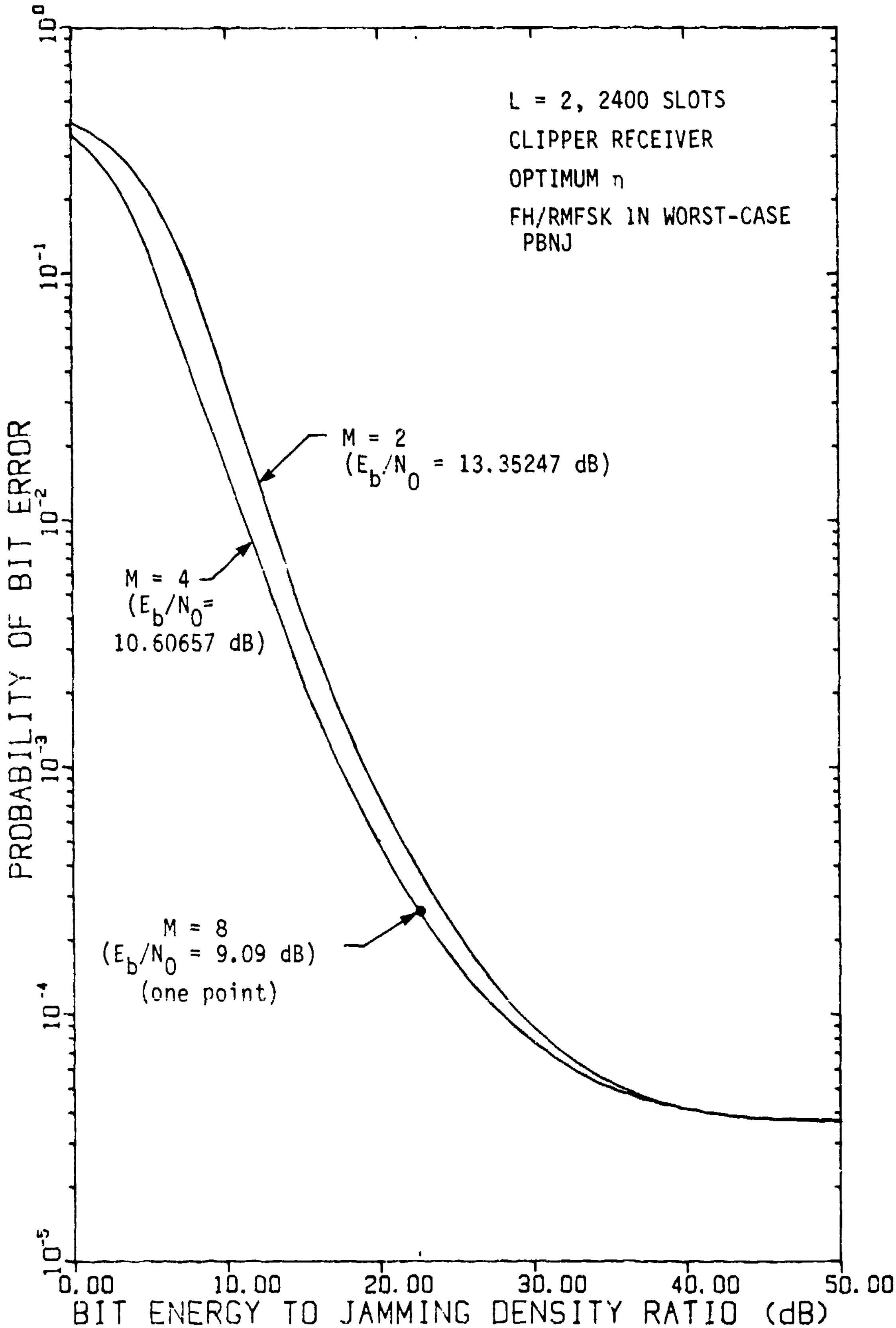


FIGURE 7.3-10 PERFORMANCE OF CLIPPER RECEIVER FOR FH/RMFSK WHEN L=2 HOPS/SYMBOL WITH M AS A PARAMETER AND E_b/N_0 CORRESPONDING TO $P_b(e) = 10^{-5}$ IN THE ABSENCE OF JAMMING (WHEN L=1 HOP/SYMBOL)

threshold for L=1 is infinite (no clipping), whereas numerically it was determined that a finite threshold is optimum for L>1. Consequently, the L=1 "clipper" receiver is not a clipping receiver at all but one identical in operation to all the other RMFSK receivers for L=1 except IC-AGC, and its FH/RMFSK performance tends to get worse for increasing M as demonstrated previously in Figure 7.3.3. However, for L=2 we observe from Figure 7.3-10 that the clipper is performing in a manner similar to the IC-AGC, in that increasing M from 2 to 4 reduces the BER; however further increase to M=8 degrades performance. The reason for this similarity in behavior is that the clipper receiver, like the IC-AGC, operates to limit jamming input to the soft decision <u>on an individual channel basis</u>. The clipper is in this sense a crude version of the IC-AGC; but for higher values of M the losses become significant and the performance trend resembles the other RMFSK receivers more than the IC-AGC receiver. We then would expect clipping to be advantageous against jamming for L=1 as well; but the threshold was optimized for no jamming in order to avoid requiring the receiver to know or measure jamming parameters. If the threshold were jamming-dependent, the clipper receiver might follow the IC-AGC more closely for higher M. The tendency of the clipper receiver to "emulate" the IC-AGC was observed earlier in Figure 7.2-3, where we see that this tendency is more pronounced for strong jamming.

7.3.3.4 Hard-decision receiver.

Now if the clipper receiver can be thought of as a crude version of the IC-AGC, the hard-decision (HD) receiver can be considered a crude version of the ACJ-AGC because both act to limit or de-emphasize the entire set of M channels on a jammed hop, rather than operating on the channels separately.

Thus we observe in Figure 7.3-11 the tendency for the HD receiver's BER to increase with M (after M>4) in strong jamming, just like the ACJ-AGC receiver's BER, and thereby to yield a worse performance for RMFSK than for MFSK. In weak or no jamming, the HD's BER for L>1 gets worse for increasing M (unlike the other, soft-decision receivers) because noncoherent combining losses are in effect amplified by the quantization the HD uses.

7.3.3.5 Self-normalizing receiver.

In the previous examples, we have observed a consistent trend for RMFSK hopping to yield no better--and sometimes worse--performance than conventional MFSK hopping. This was explained as being due to the possibility of jamming being present in a non-signal channel but not in the signal channel for RMFSK but not for MFSK. It is also true that using RMFSK there can be jamming only in the signal channel, which tends to favor a correct decision. Apparently, using the LCR, AGC, clipper, and HD receivers, the jamming of one channel has a net effect of degrading the system performance for L>1.

Now, we consider the comparison of RMFSK with MFSK using the self-normalizing receiver, and will see an exception to the trend previously observed. Figure 7.3-12 displays the SNORM error performances for FH/RMFSK and RH/MFSK in WCPBNJ for M=L=2 and E_b/N_0 = 13.35 dB and 20 dB. We see that the BER for RMFSK is better than for MFSK. This behavior seems to be connected with the jamming events in which both channels are either jammed or unjammed on a given hop, rather than those for which only one channel is jammed. This statement is supported by the fact that (1) the curves are roughly parallel for moderate-to-weak jamming (the portion of

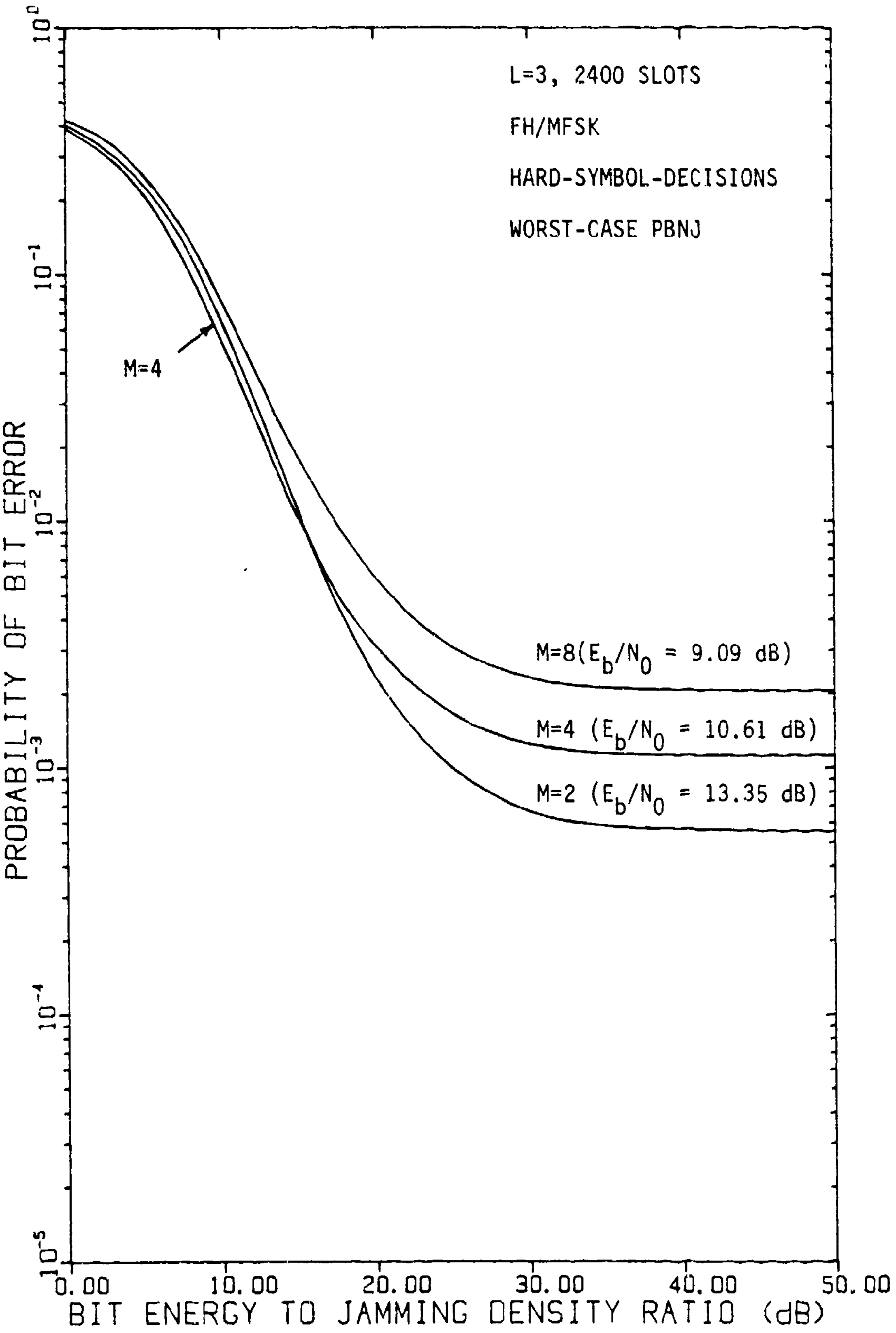


FIGURE 7.3-11 ERROR PERFORMANCE FOR HARD SYMBOL DECISION FH/RMFSK RECEIVER IN WORST-CA[SE] PARTIAL-BAND NOISE JAMMING WHEN L=3 HOPS/SYMBOL; M=2,4,8; AND E_b/N_0 GIVE[S] 10^{-5} ERROR WITHOUT JAMMING WHEN L=1

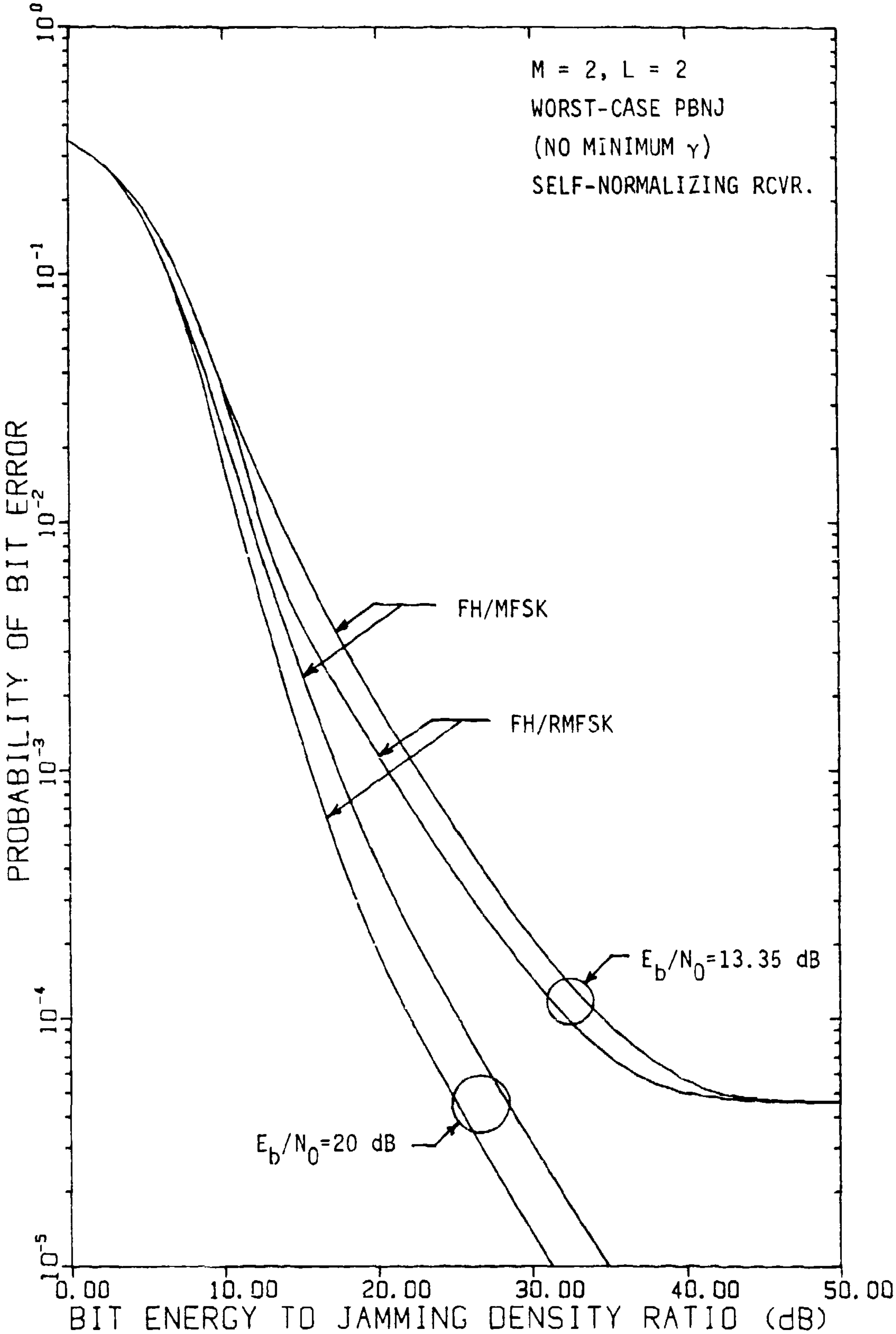


FIGURE 7.3-12 WORST-CASE PARTIAL-BAND NOISE JAMMING PERFORMANCES OF FH/RMFSK AND FH/MFSK USING THE SELF-NORMALIZING RECEIVER, FOR M=2 AND L=2

the curve proportional to e^{-E_b/N_J}), and (2) the MFSK receiver is subject only to those particular jamming events, by assumption. The advantage of RMFSK, according to this interpretation, then lies in the smaller probability of both channels being jammed on a given hop.

Now if M were increased to M=4 or M=8, it would be expected that the effects of jamming in one channel only would tend to increase the RMFSK error, since then a damaging effect would be M-1 times as likely as a helping effect.

On the other hand, the improvement of the RMFSK error over that of MFSK can be explained in terms of how the SNORM receiver processes the jamming events for which only one channel is jammed, in contrast to the way the other RMFSK receivers process the events. When only the non-signal channel is jammed the hop statistics are ($K = \sigma_T^2/\sigma_N^2 \gg 1$)

$$z_{1k} = \frac{\chi^2(2,2\rho_N)}{\chi^2(2,2\rho_N)+K\chi^2(2)} \to 0 \tag{7.3-6a}$$

$$z_{2k} = \frac{K\chi^2(2)}{\chi^2(2,2\rho_N)+K\chi^2(2)} \to 1. \tag{7.3-6b}$$

But when only the signal channel is jammed,

$$z_{1k} = \frac{K\chi^2(2,2\rho_T)}{K\chi^2(2,2\rho_T)+\chi^2(2)} \to 1 \tag{7.3-7a}$$

$$z_{2k} = \frac{\chi^2(2)}{K\chi^2(2,2\rho_T)+\chi^2(2)} \to 0. \tag{7.3-7b}$$

That is, the SNORM per-hop processing is nearly equivalent to a hard symbol decision; the signal channel suppresses the nonsignal channel when the signal is jammed, but if the nonsignal channel is jammed, it is awarded a value of at most 1. Thus the receiver de-emphasizes jammed hops while at the same time distinguishing between "good" and "bad" jammed hops. The

IC-AGC receiver, by contrast, penalizes the channel being jammed by normalizing its noise to the same variance as the other channel; this has the effect of suppressing the jammed channel when only one is jammed. This is a good thing to do when the non-signal channel is jammed; but it is not beneficial when the signal channel is jammed.

According to this second interpretation of the results the SNORM performance for RMFSK is better because it makes good use of the "favorable" jamming events, which do not occur for MFSK. However, we still would expect the RMFSK performance to degrade for higher M under this interpretation.

7.4 COMPARISON OF RECEIVER DIVERSITY EFFECTS

The FH/RMFSK receivers we have studied are distinguished by their methods of combining the L hops transmitted per MFSK symbol. The objective of the diversity transmission is to spread the signal on a symbol basis, making it less likely that the symbol is jammed for the entirety of its duration. The L pieces of the symbol transmission are then sequentially acquired noncoherently and accumulated after weighting or otherwise processing them individually. Since the combining is done noncoherently, the performance of the system without jamming or with full-band jamming (Gaussian channel) necessarily is degraded from that using one hop with same signal energy. However, when the system bandwidth is jammed partially, giving rise to a type of non-Gaussian interference channel, the system performance is improved using diversity, provided that the hop processing in some fashion limits or discriminates against those hops which are jammed.

The conventional diversity receiver for MFSK, which we have termed the linear combining square-law receiver (LCR), is known to be effective against signal fading, that is, when there exists a random-amplitude signal in a Gaussian channel. But against partial-band noise jamming (PBNJ), the LCR is not effective since jammed hops are not de-emphasized. Figure 7.4-1 illustrates for M=2 that LCR performance degrades in proportion to L.

One view of the individual-channel adaptive gain control receiver (IC-AGC), which normalizes each square-law detector sample by its *a priori* noise variance, is that it in effect renders the non-Gaussian PBNJ interference into a Gaussian interference with unit variance in each MFSK slot. The residual

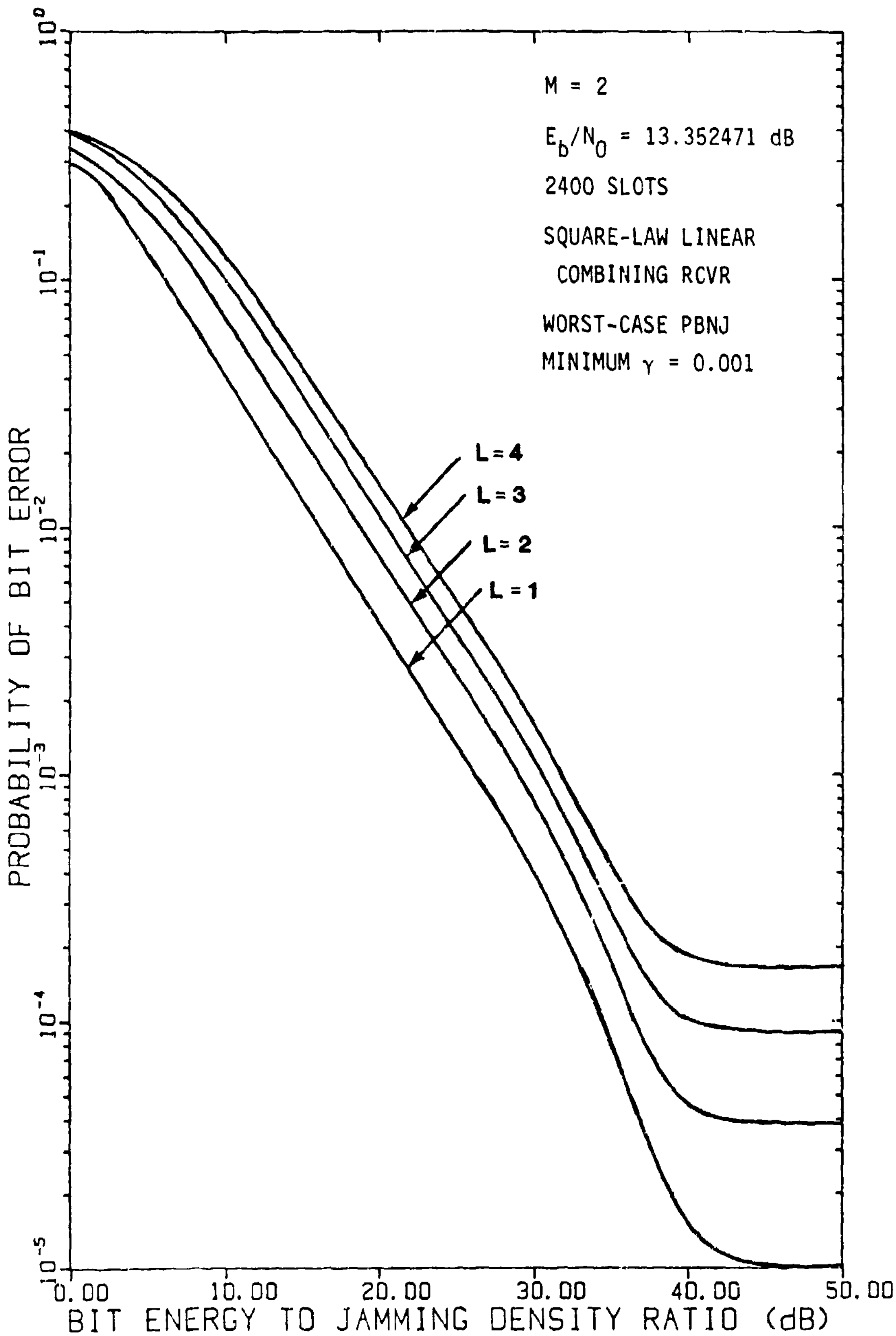


FIGURE 7.4-1 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR SQUARE-LAW COMBINING RECEIVER AND M = 2 WITH NUMBER OF HOPS/BIT AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

effect of the jamming after normalization is to reduce the SNR in the MFSK signal channel or slot by an amount which depends upon the random event of that channel's being jammed on ℓ_1 of the L hops. Because the amount of reduction is inversely proportional to γ, the fraction of the system bandwidth which is jammed, while the probability of jamming is directly proportional to γ, there exists an optimum value of γ which maximizes the system error probability as a function of available jamming power; generally $\gamma_{opt} = const_1/(E_b/N_J) = const_2 \cdot J$, that is, the optimum value of γ is directly proportional to jamming power, and for sufficient jammer power full-band jamming (γ=1) is optimum.

It is possible to reason without analysis that the IC-AGC receiver performs better than the LCR because, while the two receivers are subject to the same SNR degradation, the IC-AGC in effect "matches" the accumulator structure (soft-decision) to the channel. However, it is difficult to predict how any improvement would depend upon L, and whether an optimum value of L exists. Thus the analysis and computations of IC-AGC performance have been quite revealing. For example, Figure 7.4-2 shows that there is a tendency for increasing L to improve the IC-AGC performance for increasing E_b/N_J, but this tendency is by no means uniform for the value of E_b/N_0 shown. Figure 7.4-3 shows the IC-AGC performance which would be obtained if the best value of L were always used. This figure reveals that the effectiveness of the diversity depends on the degree of thermal noise present; when N_0 is not negligible, increasing L eventually gives rise to noncoherent combining losses which overcome the gains from diversity.

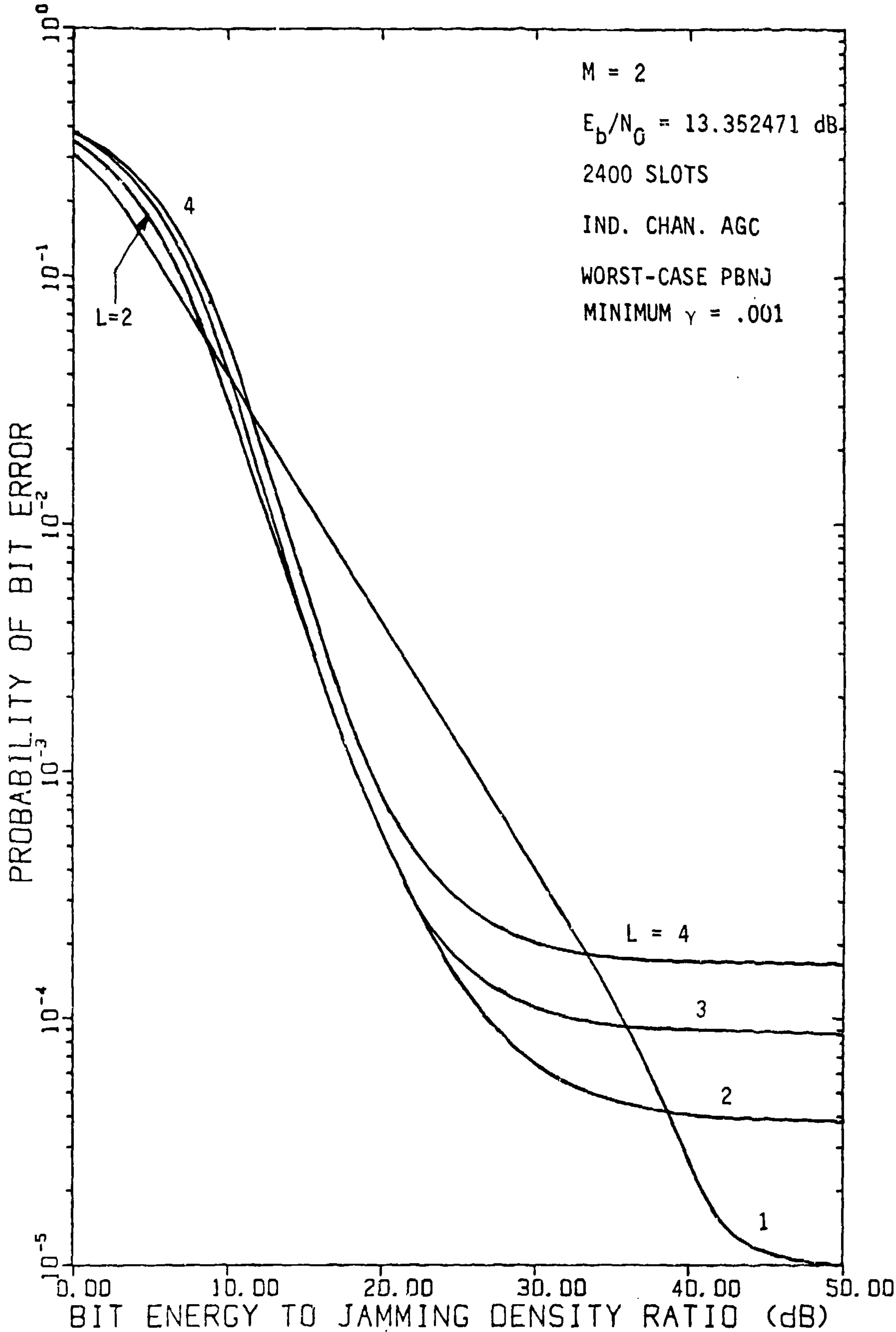


FIGURE 7.4-2 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR INDIVIDUAL CHANNEL AGC RECEIVER AND M=2 WITH NUMBER OF HOPS/SYMBOL AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

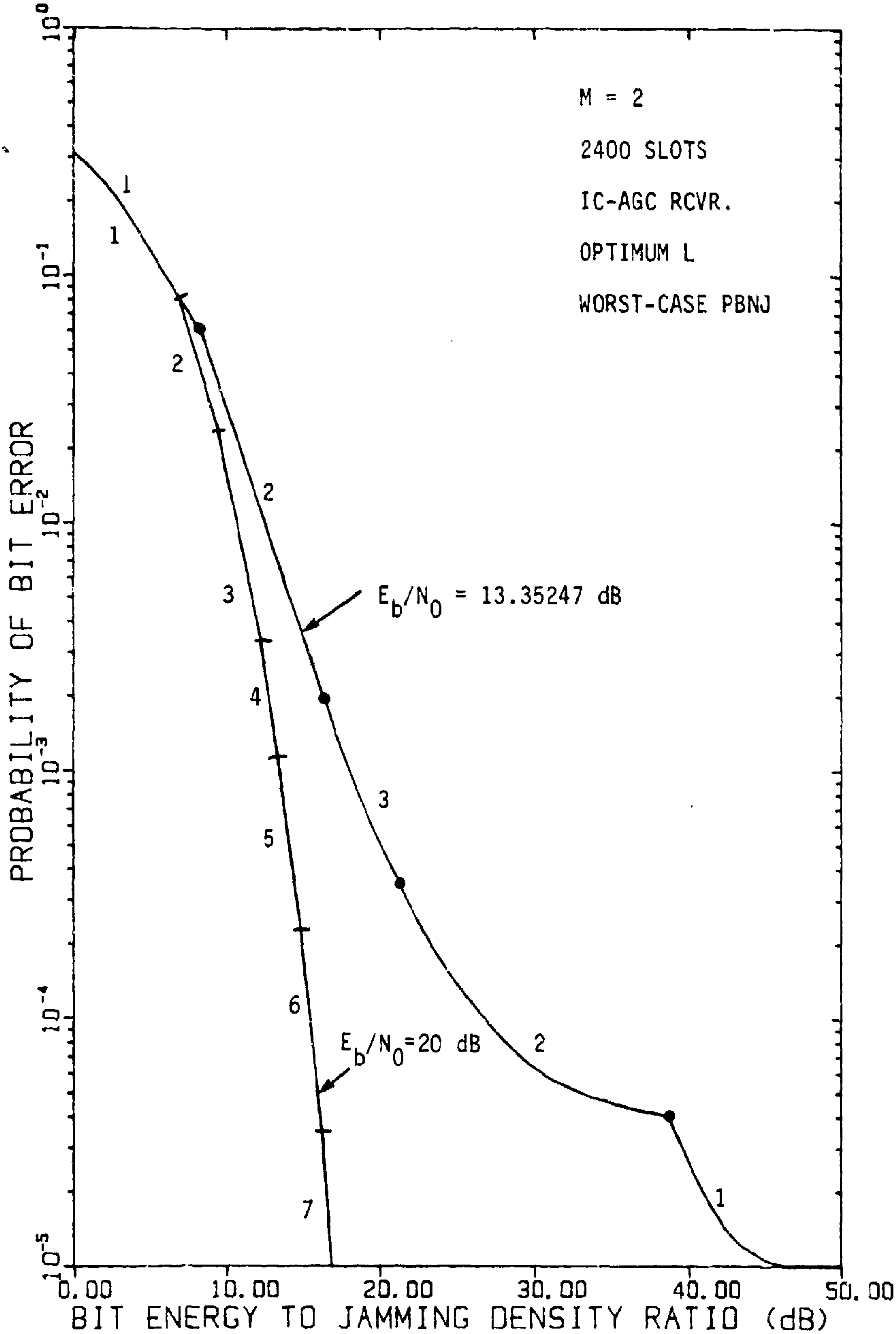


FIGURE 7.4-3 OPTIMUM DIVERSITY PERFORMANCE OF INDIVIDUAL CHANNEL AGC RECEIVER FOR FH/RMFSK WITH M=2 AND E_b/N_0 AS A PARAMETER

It is important to keep in mind that the performance in jamming can never be better than that without jamming, and that without jamming the best performance is for L=1. Therefore "optimum diversity" values may increase with E_b/N_J, but must eventually decrease again to L=1 as $E_b/N_J \to \infty$ (no jamming). However, as the figure demonstrates, for a desired performance of, say 10^{-5}, the optimum diversity value can be greater than one if the unjammed error is much smaller.

Examples of diversity effects for the "any channel jammed" receiver (ACJ) and the self-normalizing receiver are given by Figures 7.4-4 and 7.4-5. We observe that these receivers, in that their normalization techniques "approximate" that of the IC-AGC, achieve similar diversity gain effects.

Two of the receivers studied, the clipper and hard-symbol decision (HD) receivers, do not utilize normalization as such, yet accomplish a diversity gain effect by limiting the "contamination" that a jammed hop may bring to the symbol decision. It has noted previously that the clipper receiver's performance generally is close to that of the ACJ for stronger jamming, but that the HD receiver is considerably worse for the same amount of thermal noise. However, as Figure 7.4-6 shows, even this very simple HD approach can be considered effective in the diversity sense when thermal noise is low.

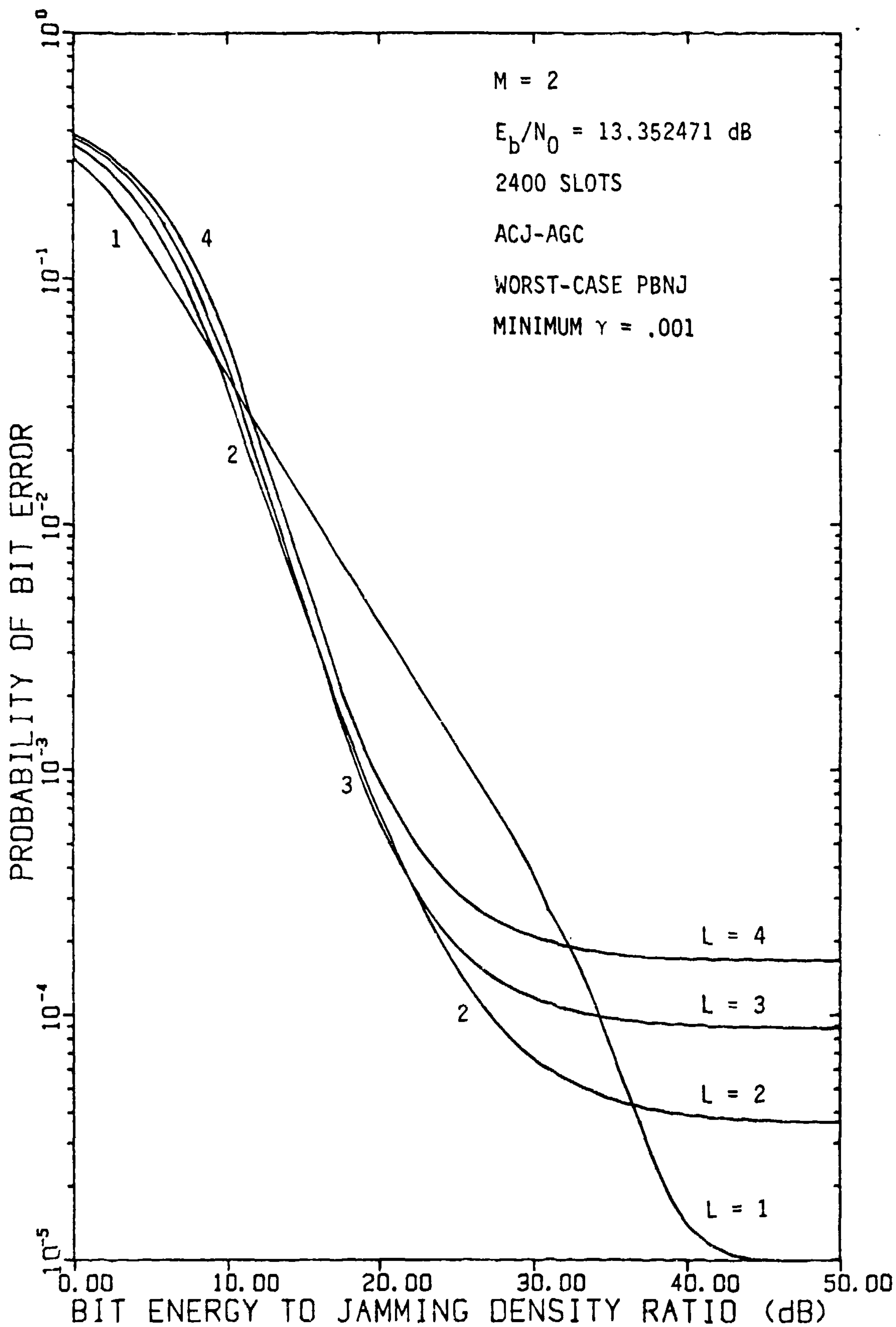


FIGURE 7.4-4 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING DENSITY RATIO FOR ANY-CHANNEL-JAMMED AGC RECEIVER AND M=2 WITH NUMBER OF HOPS/BIT AS A PARAMETER IN PRESENCE OF WORST-CASE PARTIAL-BAND NOISE JAMMING

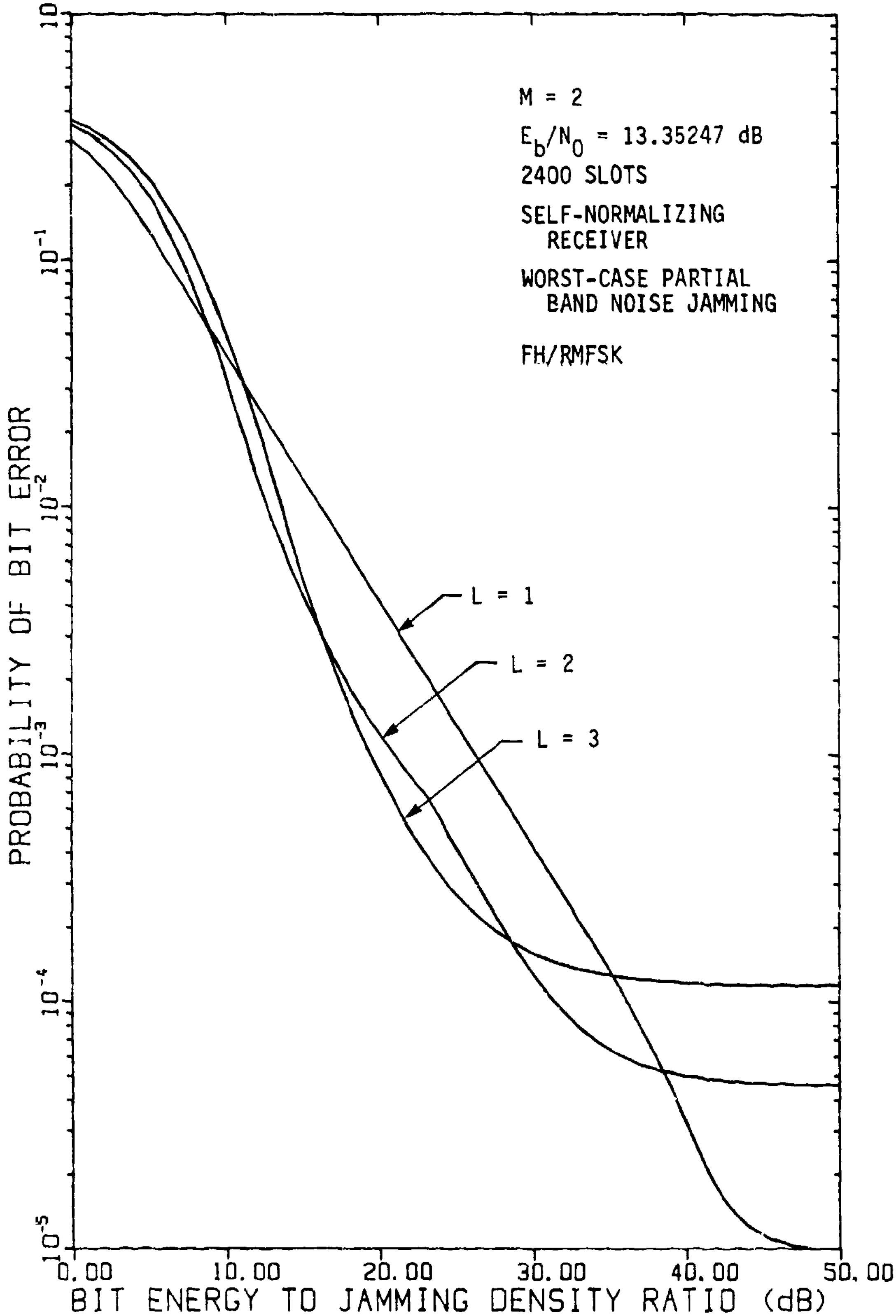


FIGURE 7.4-5 BIT ERROR PROBABILITY VS. BIT ENERGY TO JAMMING NOISE DENSITY RATIO FOR SELF-NORMALIZING RECEIVER IN WORST-CASE PARTIAL-BAND NOISE FOR M=2 AND E_b/N_0=13.35 dB, WITH THE NUMBER OF HOPS/BIT (L) VARIED

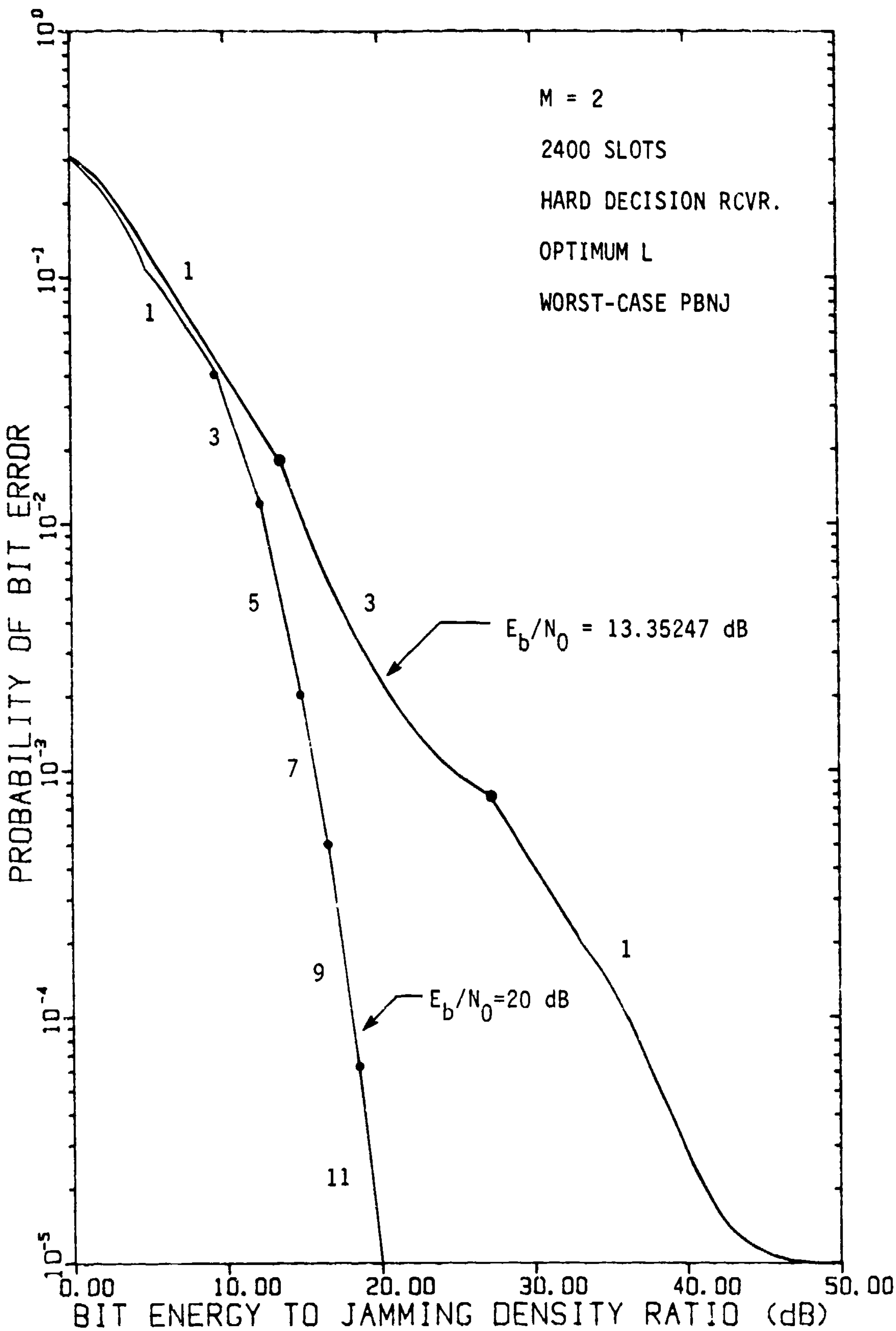


FIGURE 7.4-6 OPTIMUM DIVERSITY PERFORMANCE OF HARD DECISION RECEIVER FOR FH/RMFSK WITH M=2

8.0 ECCM RECEIVER IMPLEMENTATION STUDIES

In the previous sections, we analyzed the BER performance of various ECCM receiver processing schemes for uncoded FH/RMFSK radio systems in the presence of worst-case partial-band noise jamming (PBNJ).

Our objective has been to provide a comparison of these different systems, which vary extensively in their implementation complexities. These results will enable the ECCM system designer to weigh the engineering cost requirements of a particular receiver design versus the anti-jam effectiveness. Toward this end, we now explore practical issues related to the implementation of these different processing schemes along with an assessment of implementation effects.

8.1 IMPLEMENTATION ISSUES AND CONCEPTS

All receivers suppress the total noise jamming power by an amount equivalent to the system processing gain, defined as the ratio of the jammer bandwidth to the receiver bandwidth. Hopping the signal forces the jammer to spread its power over a wide bandwidth, but the jammer can maximize its effectiveness by selecting an optimum bandwidth, which is a certain fraction (γ) of the total hopping system bandwidth (W). This results in a BER which tends to be an inverse linear function of E_b/N_J, so that more than 40 dB of E_b/N_J is required to obtain BER's less than 10^{-4}. ECCM FH/MFSK and FH/MFSK systems counter this effect by using multiple hops per symbol, with the L hops per symbol combined at the receiver as in diversity transmission schemes.

We have demonstrated that effective jammer suppression is obtained by incorporating a nonlinear function in each M-ary channel prior to combining. The improvement in BER performance is realized by the fact that within a PBNJ environment, the nonlinear techniques (clipper, AGC, hard-decision, SNORM) mitigate the tendency of a jammed hop to dominate the symbol decision. Of these nonlinear techniques we have studied, it was assumed that certain a priori information or perfect measurements are available to the clipper (SNR threshold) and AGC (noise powers) schemes; no such measuring tactics are necessary for the hard-decision or SNORM receivers. In what follows, we investigate the practicalities concerning the implementation and impact of non-ideal noise power measurements and threshold settings.

8.1.1 ECCM Receiver Information Requirements

In Table 8.1-1 we summarize the ECCM techniques used by the various receivers we have studied, including the information necessary for their implementation. The square-law linear combining receiver is presented as a baseline for comparison with the other, nonlinear combining types. Our specific interest here is to address the feasibility of implementation.

In the table, the nonlinear combining receivers are classified according to whether their anti-jam measures operate on a per-symbol basis (across all M channels) or on a per-channel basis. The per-symbol ECCM receivers include the ACJ-AGC, the hard-decision, and the SNORM receivers. Of these, the ACJ-AGC is seen as the only type utilizing a priori information on the received noise (thermal plus jamming), since it weights all channels on a given hop by

$$w_{mk} = w_k = \begin{cases} (\sigma_N^2)^{-1}, & \text{no channels jammed on hop } k \\ (\sigma_T^2)^{-2}, & \text{any channel jammed on hop } k \end{cases}$$

$$= \left[\max_m (\sigma_{mk}^2) \right]^{-1}. \tag{8.1-1}$$

TABLE 8.1-1 SUMMARY OF ECCM TECHNIQUES FOR THE RECEIVERS STUDIED

EXTENT OF HOP WEIGHTING OR ANTI-JAM-MEASURE	RECEIVER(S)	SUPPRESSION TECHNIQUE	INFORMATION REQUIRED
None	Linear Combining	None	None
Per-Symbol	Any-Channel-Jammed AGC (ACJ-AGC)	Scale Down Jammed Hops (Normalize Max Variance to 1)	$\max_m \{\sigma^2_{mk}\}$
	Hard-Symbol Decision	Limit Hops to One Vote	None
	Self-Normalizing (SNORM)	Scale Down Jammed Hops (Normalize Channel Sum to 1)	None
Per-Channel	Individual Channel AGC (IC-AGC)	Scale Down Jammed Channels (Normalize All Variances to 1)	σ^2_{mk}
	Clipper	Limit Each Channel to Max Value of η	σ^2_N and E_b/N_0 (to Set Value of η)

*Notes: (1) *Hops numbered by* k ($k=1,2,\ldots,L$) *and channels numbered by* m ($m=1,2,\ldots,M$)

(2) σ^2_{mk} is the a priori variance of the total noise present in channel m at time k

(3) Since σ^2_{mk} is assumed to be either σ^2_N (unjammed) or σ^2_T (jammed), "information required" may be interpreted as knowing whether a channel is jammed (IC-AGC) or a symbol is jammed (ACJ-AGC)

Therefore, the information required is $w_k^{-1} = \max_m (\sigma^2_{mk})$, which involves knowing σ^2_N, σ^2_T, and whether any of the M channels is jammed. As a minimum, the ACJ receiver needs to know the ratio $\max_m(\sigma^2_{mk})/\sigma^2_N$, since the operation of the receiver is unaffected if unjammed hops are left alone (weight = 1) and jammed hops are reduced by the factor σ^2_N/σ^2_T.

The hard-decision receiver is classified as a per-symbol ECCM receiver because, as we have observed in previous sections, its operation in effect limits each symbol piece (hop) to one vote in the M-ary majority logic decision, no matter how strongly a hop may have been jammed. Its operation does not require any a priori information or measurement.

The SNORM receiver derives its per-symbol weights from the M square-law envelope detector samples themselves:

$$w_k = \left(\sum_m x^2_{mk} \right)^{-1} . \tag{8.1-2}$$

Therefore, it does not require a priori information or additional measurements in its operation.

The per-channel ECCM receivers include the IC-AGC and the clipper receivers, and both utilize a priori information. The IC-AGC weights each channel sample by the inverse of its a priori noise variance:

$$w_{mk} = (\sigma^2_{mk})^{-1} . \tag{8.1-3}$$

In this manner, all channels on all hops are normalized to have unit noise variance; any channels which are jammed ($\sigma^2_{mk} = \sigma^2_T$) are therefore suppressed. This technique involves knowing σ^2_N, σ^2_T, and the jamming state or condition of each channel. Alternately, the ratios σ_{mk}^2/σ_N^2 are needed, as a minimum.

The clipper receiver achieves an ECCM effect by "containing" any jammed channels; their contribution to the soft-decision sums cannot be any larger than the clipping threshold (η), no matter how strongly jammed. In order to set the threshold, both σ_N^2 and E_b/N_0 *a priori* values are needed since $\eta = \eta(\sigma_N^2, E_b/N_0)$ is chosen to minimize the error without jamming.

With respect to the additional receiver complexity required to develop information needed by the suppression technique, only the AGC schemes (jamming decision and normalization weights) and clipper (SNR levels) receivers need be addressed.

8.1.2 Measurement Approaches.

Implementation of the two AGC schemes requires differentiation between two zero-mean bandpass Gaussian noise processes with different variances which determine a jammed/unjammed channel state. In addition, to be useful as a quantity for a normalization weight in an AGC scheme, our measurement (noise-power estimate) must reflect as closely as possible in real-time the actual system state of the measured channel. This leads us to consider factors in both time and frequency domain representations of our measuring technique, i.e. the accuracy or quality of a band-limited channel noise-power measurement.

It is assumed throughout the measurement process that the data measures are sample records from a continuous stationary random process. Letting x(t) be a single sample time history record from a zero-mean stationary (ergodic) Gaussian random process $\{x(t)\}$, the mean-square value (variance) of $\{x(t)\}$ can be estimated by time-averaging over a finite time interval τ as

$$\hat{\sigma}_x^2 = \frac{1}{\tau} \int_0^{\tau} x^2(t)dt \qquad (8.1\text{-}4)$$

with the true mean-square value being

$$\sigma_x^2 = E\,[\,x^2(t)\,] \tag{8.1-5}$$

and is independent of t since $\{x(t)\}$ is stationary. Now the expected value of the estimate of σ_x^2 is

$$E[\hat{\sigma}_x^2] = \frac{1}{\tau}\int_0^\tau E[x^2(t)]\,dt = \frac{1}{\tau}\int_0^\tau \sigma_x^2\,dt = \sigma_x^2 . \tag{8.1-6}$$

Thus, $\hat{\sigma}_x^2$ is an unbiased estimate of σ_x^2, independent of τ.

Regarding the quality of measurement, it is shown [19 , p. 176] that the variance of a mean-square value estimate of band-limited white Gaussian noise with zero mean is

$$\mathrm{Var}[\hat{\sigma}_x^2] \simeq \frac{\sigma_x^2}{B\tau} . \tag{8.1-7}$$

Hence, we clearly see the inverse relationship of accuracy of our measurement to the measured channel time-bandwidth product; that is, for $B\tau \to \infty$ we would realize a "perfect" noise measurement.

8.1.2.1 A look-ahead measurement scheme.

Since the optimum power estimator uses a square-law detector [18] one could obtain a workable noise-power estimate by measuring the next slot to be hopped into by all M-ary channels; such a "look-ahead" scheme is illustrated in Figures 8.1-1 and 8.1-2.

Figure 8.1-1 shows the first part of the scheme in which we obtain the look-ahead receiver samples. Here we allow the receiver channel code-synchronized PN sequence generators to be delayed by one hop period (τ) in relationship to the respective measurement channel synchronization. In this manner we obtain the look-ahead samples at hop time k for use with

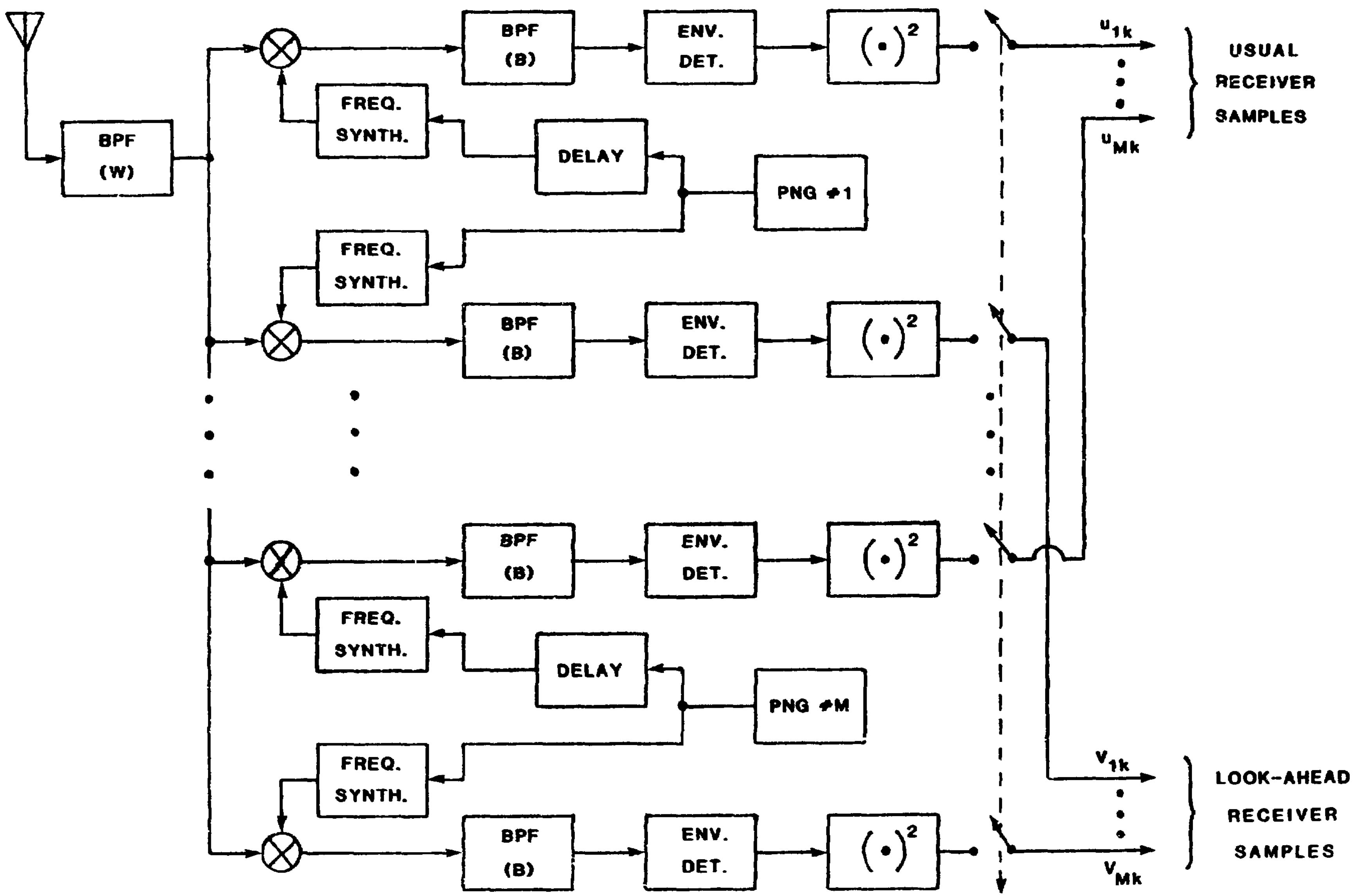


FIGURE 8.1-1 ACQUISITION OF JAMMING STATE DATA BY A "LOOK-AHEAD" SCHEME

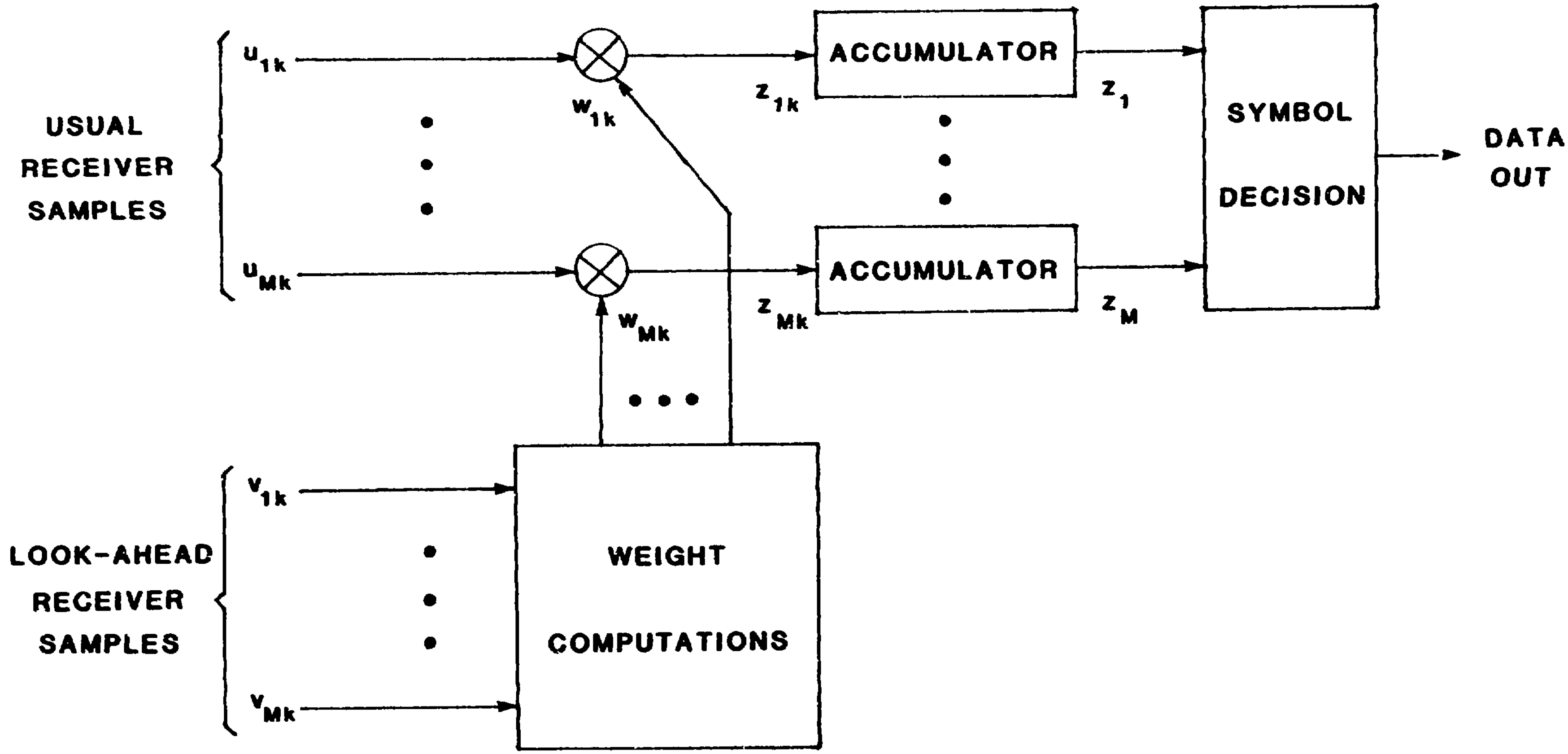


FIGURE 8.1-2 USE OF LOOK-AHEAD MEASUREMENTS TO DERIVE ECCM WEIGHTS

the usual communicator receiver samples to be gathered at time k+1. We note that the look-ahead samples are assumed to be values of either σ_N^2 or σ_T^2 and that any type of spectral interference from other hoppers in the network is nonexistent. Also, this first part (Figure 8.1-1) of the look-ahead operation is the same for both IC- and ACJ-AGC receiver types.

In Figure 8.1-2 we show the use of the M-channel look-ahead measurements to derive the proper normalization weights. Thus, at time k+1 we obtain the following variables for accumulating in the L-hop diversity combining state for each of the M channels:

$$z_{mk} = x_{mk}^2 \cdot \left[\max_m \left(\hat{\sigma}_{mk}^2\right)\right]^{-1}, \text{ for ACJ} \qquad (8.1\text{-}8)$$

$$z_{mk} = x_{mk}^2 \cdot \left(\hat{\sigma}_{mk}^2\right)^{-1}, \text{ for IC.} \qquad (8.1\text{-}9)$$

Additionally, the weight computation network could incorporate a multi-hop/multi-channel processing stage (see Figure 8.1-3) to determine jamming state information (JSI) based upon multiple look-ahead measurements. Should the channel be jammed, we then have a one-sample estimate of the noise plus jamming power $\hat{\sigma}_T^2$. If unjammed, we obtain $\hat{\sigma}_N^2$ which is a one-sample measure of the channel's thermal noise power. The additional processing envisions each channel estimate $\hat{\sigma}_N^2$ as going into a smoothing operation, which would then output the improved σ_N^2 and σ_T^2 estimates. These values in turn are fed back to the jammed/unjammed decision blocks in the form of the detection threshold. The use of multiple-hop/multiple-channel measurements improves the quality of the noise power estimates by increasing the $B\tau$ product in (8.1-7). Details involved in both the jamming state detection and data smoothing of estimates are discussed later in this section.

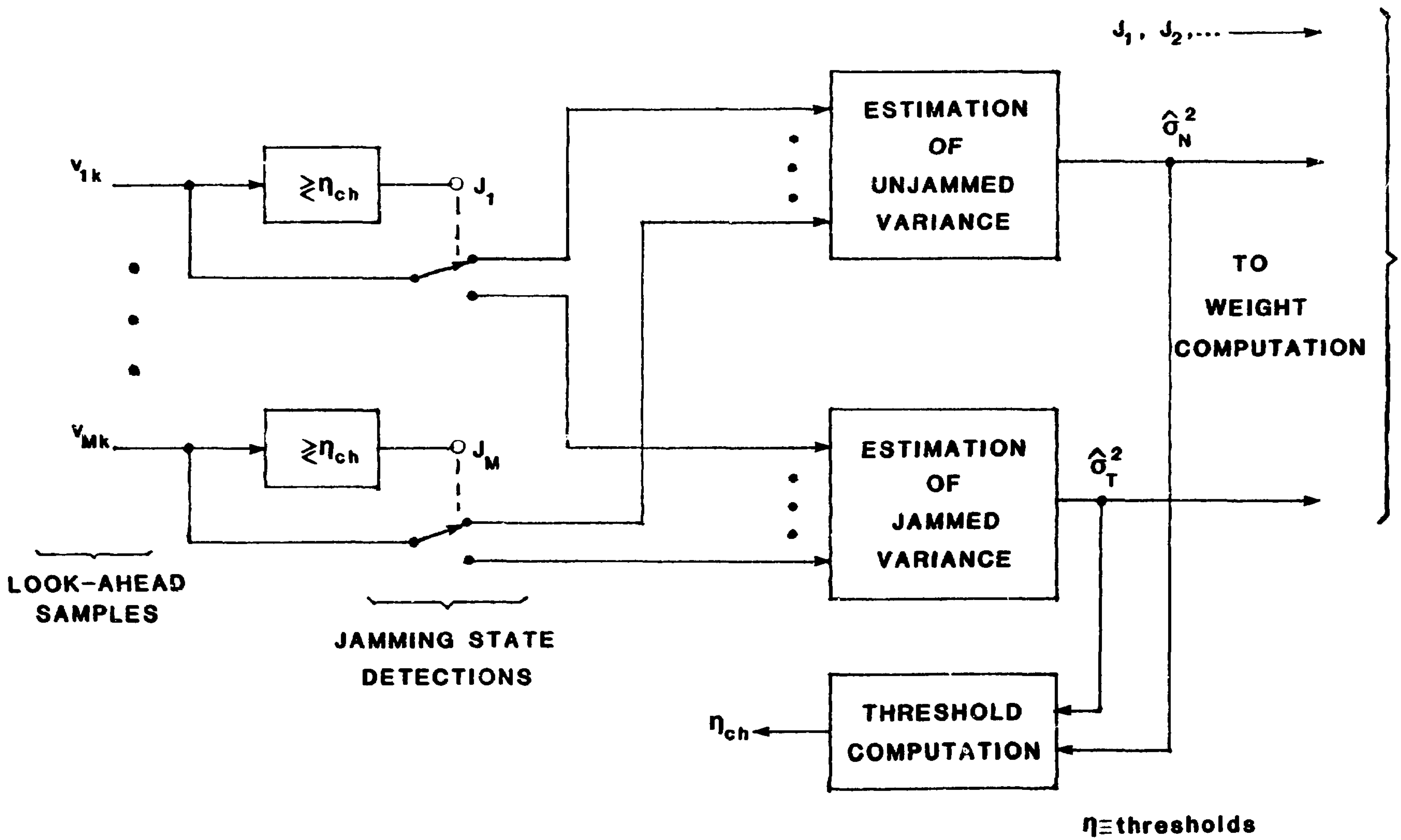


FIGURE 8.1-3 POSSIBLE USE OF MULTIPLE-HOP/MULTIPLE CHANNEL SAMPLES TO IMPROVE MEASUREMENTS

8.1.2.2 In-band measurement schemes.

One in-band measurement scheme assumes that $\hat{\sigma}_N^2$ and $\hat{\sigma}_T^2$ are to be measured between signal transmission times over the frequency-hopping bandwidth W. That is, a measurement process takes place during the communications "link idle" state, enabling the gathering of measurement samples over many hop periods. This procedure avoids the system complexity required by the look-ahead scheme, and is subject to the same caveats regarding stationarity of the noise (thermal, background, and jamming) environment across the hopping system bandwidth, as well as the corruption of purely noise estimates by the activity of other communications users during measurement intervals.

Receiver frame synchronization information or message preamble data could be used to decide when to cease or start taking measurement data samples. In concept, these measurements would enable high quality estimates of σ_N^2 and σ_T^2 to be developed, and/or perhaps even a stored estimate of the received noise power as a function of frequency.

Now, in order to detect the presence of jamming or to set the clipper's optimized threshold, we require knowledge of the received (average) thermal noise power $\sigma_N^2 = N_0B$, where N_0 is the noise density in watts per hertz, assumed to be independent of frequency, and B is the channel bandwidth. In concept, an independent channel noise-power measurement system could be used to take several simultaneous measurements uniformly distributed within the thermal-noise-only (unjammed) portion of the hopping band W. The arithmetic average of these measurements would then be the estimate $\hat{\sigma}_N^2$. Assuming that σ_N^2 varies slowly, if at all, we would continuously correct the long-term moving average of $\hat{\sigma}_N^2$;

that is, a type of smoothing operation in which the processing scheme uses all measurements between times 0 and T to estimate the state of a system at a time t, where $0<t<T$. This smoothed estimate of σ_N^2 over the time interval 0 and T can be denoted by $\underline{\sigma}_N^2(t|T)$. Specifically, we envision a fixed-lag smoother in which a running smoothing solution lags the most recent measurement by a constant time delay Δ and is denoted as $\hat{\sigma}_N^2(T-\Delta|T)$. A reasonable value of Δ would be equal to the time for one symbol transmission.

Figure 8.1-4 illustrates a conceptual in-band measurement approach for the AGC type FH/RMFSK receivers. There are two modes: (a) between signal transmissions and (b) during signal transmissions. Between transmissions, the receiver continues to operate its synthesizers, detectors, and samplers to gather data for estimates of σ_N^2 and σ_T^2, as mentioned above. During transmissions, jamming detection at the channel level (threshold η_{ch}) or symbol level (threshold η_{sym}) would furnish jamming state information (JSI) for selection of AGC weights, using thresholds based on the estimates of σ_N^2 and σ_T^2. Possibly the samples received during transmissions could be used also for the variance estimation by feeding back the symbol decision to identify the channel with the signal, as suggested in the diagram.

With respect to the clipper receiver, Figure 8.1-5 depicts a scheme for setting the optimized clipping threshold η_0. Toward this end, we need to obtain a current estimate of the clipper receiver's SNR. Similar to the previously described two-mode in-band noise-power measurement concept for the AGC receivers, an individual channel measurement system would likewise be used to estimate the received signal power S. Several measurements would be taken

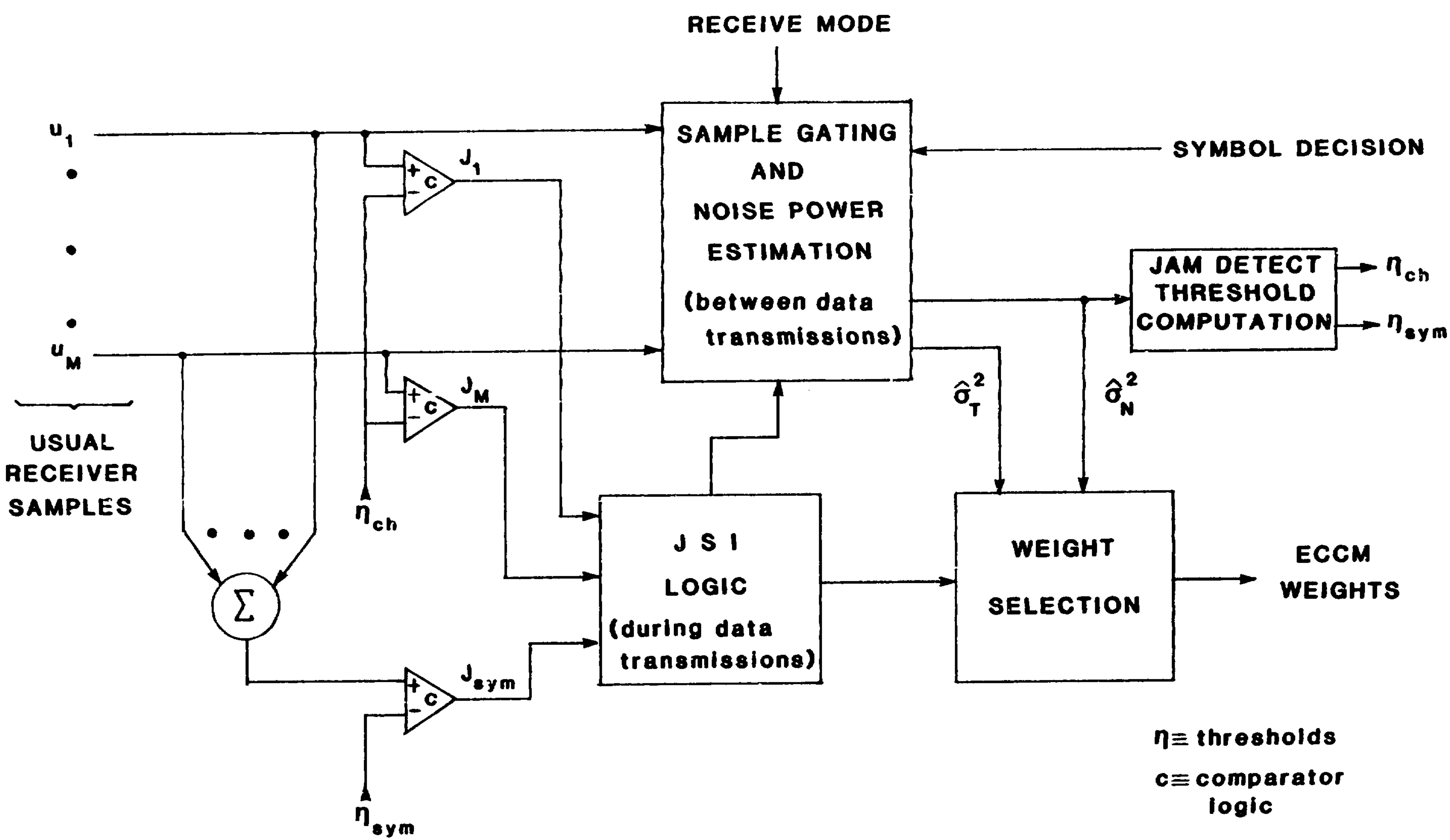


FIGURE 8.1-4 CONCEPTUAL PROCESSING OF IN-BAND MEASUREMENTS TO IMPLEMENT AGC RECEIVERS

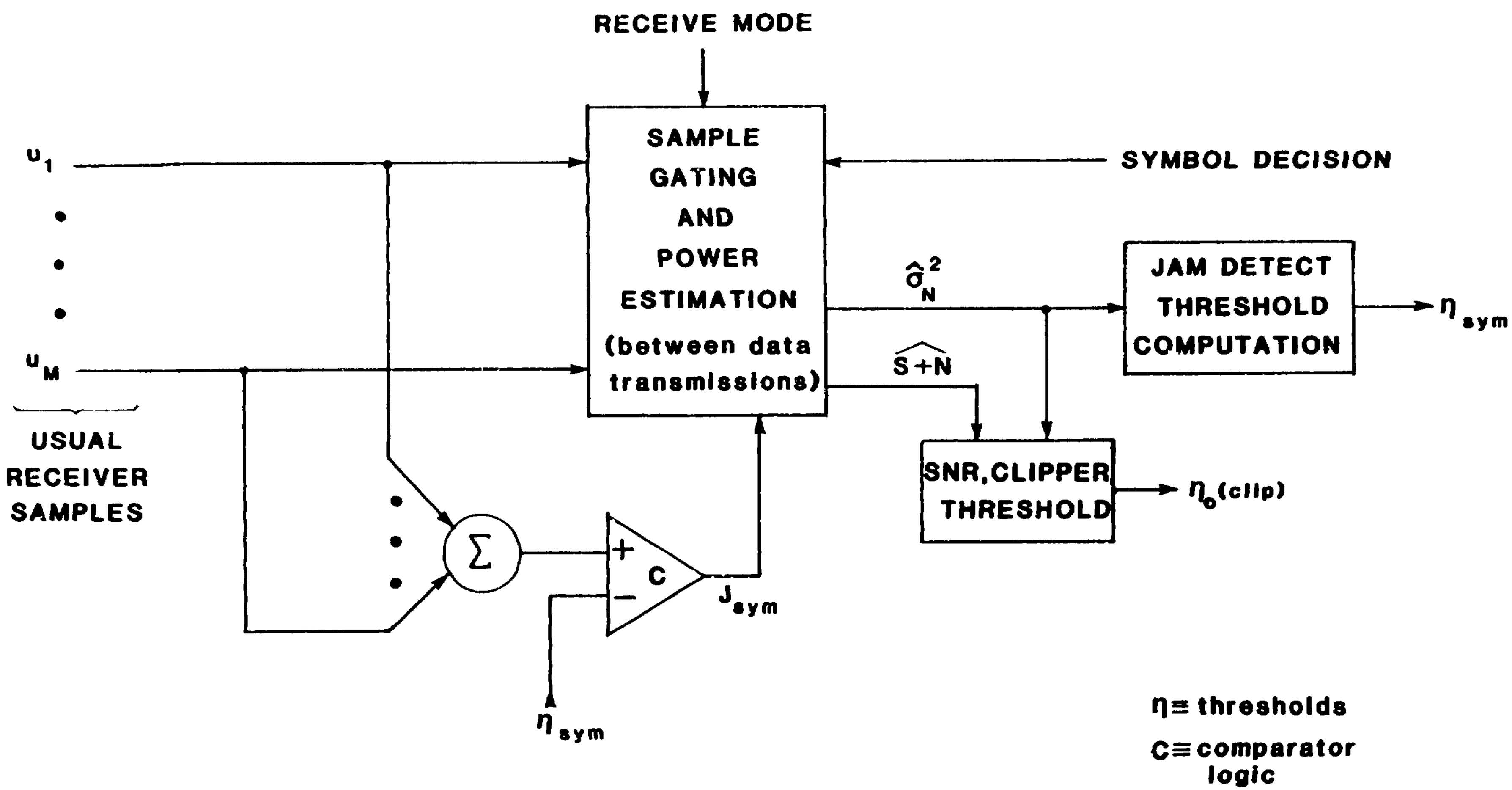


FIGURE 8.1-5 PROCESSING OF IN-BAND MEASUREMENTS TO IMPLEMENT CLIPPER RECEIVER

of the received signal plus thermal noise within the unjammed portion of the hopping band W, using symbol decision feedback to identify the signal channel. The arithmetic average of these measurements forms an estimate of signal plus noise power, $\hat{P}_{sn}$. This estimate could also be refined by a fixed-lag (per-symbol) smoothing operation. Hence, we would obtain

$$\hat{\rho}_N = \frac{\hat{S}}{\hat{\sigma}_N^2} = \frac{\hat{P}_{sn}}{\hat{\sigma}_N^2} - 1 = \frac{K}{L} \cdot \widehat{\frac{E_b}{N_0}} , \qquad (8.1\text{-}10)$$

the signal-to-noise ratio for a given hop dwell time, to be updated on a per-symbol basis.

Table 5.3-1 showed a summary of the clipper receiver values of η_0 for a given L, M, and E_b/N_0. Note that these numeric values were obtained only for values of E_b/N_0 such that $P_b(e) = 10^{-5}$ for two MFSK systems in the absence of jamming. We point out that, in practice, new values of η_0 need to be computed for each different value of E_b/N_0. Therefore the actual clipper receiver would require that matrices of η_0 values be stored in a read-only memory (ROM) look-up table.

8.1.3 Jamming State Decisions.

Implementation of a jamming state decision scheme would be based upon the following assumptions: (1) that a look-ahead or in-band measurement scheme is utilized, and (2) that the noise power estimates of $\hat{\sigma}_N^2$ and $\hat{\sigma}_T^2$ are readily available in time. Ultimately, the criterion for making a jammed/unjammed state detection is predicated upon a single observation (per channel) or a multiple observation (per symbol). The ACJ-AGC receiver requires detection of a jammed symbol hop whereas the IC-AGC requires detection of jamming in each of the M channels on each hop.

In both the symbol and the channel cases, the problem is one of deciding between Gaussian noise with variance σ_N^2 and Gaussian noise with variance σ_T^2. Due to the lack of a priori jamming probabilities (γ) or cost functions, we utilize the Neyman-Pearson criterion as our hypothesis testing technique. Its test objective is to maximize the probability of detection (P_D) for a given probability of false alarm (α) and is accomplished by employing a likelihood ratio test.

8.1.3.1 Jammed channel detection.

A basic problem in determining a jammed/unjammed state with an individual channel look-ahead scheme is in accounting for the signal itself. Recalling that the look-ahead measurement channel is one hop period "ahead" of the normal receiver channel, the situation can arise in which the measurement channel is actually the present signal channel. However, we first analyze the case for a measured channel without a signal present.

Using the narrowband Gaussian process representation for the channel samples, the hypotheses to be considered are

$$H_0:\ p(\underline{n}_c, \underline{n}_s) = \frac{1}{2\pi\sigma_N^2} \exp\left\{ - \frac{n_c^2+n_s^2}{2\sigma_N^2} \right\} \tag{8.1-11}$$

$$H_1:\ p(\underline{n}_c, \underline{n}_s) = \frac{1}{2\pi\sigma_T^2} \exp\left\{ - \frac{n_c^2+n_s^2}{2\sigma_T^2} \right\} \tag{8.1-12}$$

where H_0 is the unjammed case and H_1 is the jammed case. For the likelihood ratio we obtain

$$\Lambda = \frac{\sigma_N^2}{\sigma_T^2} \exp\left\{ -\frac{(n_c^2+n_s^2)}{2} \left(\frac{1}{\sigma_T^2} - \frac{1}{\sigma_N^2}\right)\right\} \tag{8.1-13}$$

and in comparing the log-likelihood function to a threshold we have

$$\log \Lambda = \log\left(\frac{\sigma_N^2}{\sigma_T^2}\right) + \frac{x^2}{2}\left(\frac{\sigma_T^2-\sigma_N^2}{\sigma_N^2\sigma_T^2}\right) \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{8.1-14}$$

where $x^2 = n_c^2+n_s^2$ is our estimated look-ahead noise-power measurement or sample test statistic for a single channel.

The value of x^2 to decide that jamming is present (H_1 true) is

$$x^2 \geq \frac{2\sigma_N^2\sigma_T^2}{\sigma_T^2-\sigma_N^2}\left[\eta - \log\left(\frac{\sigma_T^2}{\sigma_N^2}\right)\right] = \eta_{ch},\ \sigma_T^2>\sigma_N^2 . \tag{8.1-15}$$

For the false alarm probability we have

$$P_{FA} = \Pr\left\{x^2>\eta_{ch} | H_0\right\} = \alpha = e^{-\eta_{ch}/2\sigma_N^2} . \tag{8.1-16}$$

Similarly, the probability of detection is

$$P_D = e^{-\eta_{ch}/2\sigma_T^2} \tag{8.1-17}$$

with a receiver operating characteristic for our jamming detector given by

$$P_D = \left[\alpha\right]^{\sigma_N^2/\sigma_T^2} = \left[\alpha\right]^{1/K} . \tag{8.1-18}$$

Now in the case of the look-ahead channel sampling a signal channel slot, these samples (x^2) are independent noncentral chi-squared random variables with two degrees of freedom, multiplied by the total channel noise power σ_{ch}^2, and with noncentrality parameters

$$\lambda = 2S/\sigma_{ch}^2 = \begin{cases} 2S/\sigma_N^2 = 2\rho_N, \text{ hop unjammed} \\ 2S/\sigma_T^2 = 2\rho_T, \text{ hop jammed.} \end{cases} \tag{8.1-19}$$

Therefore, the pdf of a given sample $u=x^2$ is

$$f_{sig}(u) = \frac{1}{2\sigma_{ch}^2} e^{-(u+2S)/2\sigma_{ch}^2} I_0(\sqrt{2Su}/\sigma_{ch}^2) \tag{8.1-20a}$$

$$= \begin{cases} (1/2\sigma_N^2)\, e^{-(u+2S)/2\sigma_N^2} I_0(\sqrt{2Su}/\sigma_N^2), \quad u \geq 0, \text{ hop unjammed} \\ (1/2\sigma_T^2)\, e^{-(u+2S)/2\sigma_T^2} I_0(\sqrt{2Su}/\sigma_T^2), \quad u \geq 0, \text{ hop jammed} \end{cases} \tag{8.1-20b}$$

where S signifies power in the signal itself. Consequently, the probability of a false alarm and the probability of detection can be written as

$$P_{FA} = Q\left(\sqrt{2\rho_N}, \sqrt{\eta_{ch}/\sigma_N^2}\right) \tag{8.1-21}$$

$$P_D = Q\left(\sqrt{2\rho_T}, \sqrt{\eta_{ch}/\sigma_T^2}\right) \tag{8.1-22}$$

where $Q(x,y)$ is Marcum's Q-function.

8.1.3.2 Jammed symbol detection.

In the jammed M-ary symbol detection case, we have the two hypotheses of: (1) no channel is jammed, and (2) any of the M channels is jammed, i.e. ACJ. Hence, the situation is one of multiple alternative hypothesis testing given as

$$\begin{aligned} &H_0:\ \sigma_m^2 = \sigma_N^2,\quad m = 1,2,\ldots,M; \\ &H_1:\ \sigma_m^2 = \sigma_T^2,\ 2^M\text{-}1 \text{ possible combinations of jammed channels;} \end{aligned} \qquad (8.1\text{-}23)$$

where σ_m^2 is a parameter in the likelihood function of the measurement samples which is written as

$$p(\underline{n}_c,\underline{n}_s) = \prod_m \left(\frac{1}{2\pi\sigma_m^2}\right) e^{-x_m^2/2\sigma_m^2}. \qquad (8.1\text{-}24)$$

As in the jammed channel detection case, we require a current estimate of σ_N^2 obtained from a look-ahead or in-band noise-power measurement scheme. Furthermore, an estimate of SNR ($\hat{\rho}_N$) is also needed for threshold determination when accounting for the signal itself being detected in a look-ahead scheme.

Since there are many alternatives to H_0 as expressed by (8.1-23), we consider the simplified test of whether the sum of the channel samples exceeds a threshold.

For the case of the signal being present in one of the channels, the sum of the samples ($u_m = x_m^2$) is distributed as, with no jamming,

$$\sum_{m=1}^{M} u_m = \sigma_N^2 \chi^2(2M, 2\rho_N) \qquad (8.1\text{-}25)$$

i.e., a non-central chi-squared variable with 2M degrees of freedom, non-central parameter $2\rho_N$, and weighted by the current estimate of σ_N^2. In the noise-only case, the distribution of the summed decision variables is a central chi-squared distribution, written as

$$\sum_{m=1}^{M} u_m = \sigma_N^2 \chi^2(2M) \ . \tag{8.1-26}$$

A scheme that may provide a workable jammed symbol detection when a signal slot can be jammed is realized by discarding the maximum of the u_m variables which results in the distributional assumption

$$\sum_{m=1}^{M-1} u_m - u_{max} = \sigma_N^2 \chi^2(2M-2). \tag{8.1-27}$$

Note that the distribution of (8.1-27) is written as a central chi-square distribution which assumes that the signal itself is always stronger than the jamming. That is, the presence of weak jamming can easily be detected in the absence of the now discarded stronger signal. Should the jamming be strong, we have a situation in which the signal channel is included in the left-over sum. However, this is an acceptable scenario because the signal channel power will help in deciding if a jammed symbol condition exists.

The probability of a false alarm for this scheme is obtained by applying the methodology to formulate the non-signal channel probabilities for ACJ-AGC receivers as discussed in Section 4.3. Specifically, the probability of the sum of the non-signal channel measurement samples being less than the normalized symbol threshold on a given hop is written as

$$\Pr\left\{\sum u_m < \frac{n_{sym}}{\sigma_N^2}\right\} = 1 - \frac{\Gamma(2M-2; n_{sym}/2\sigma_N^2)}{\Gamma(2M-2)} \quad . \tag{8.1-28}$$

Hence, the probability of a false-alarm for jammed symbol detection is simply

$$P_{FA(sym)} = \frac{\Gamma(2M-2; n_{sym}/2\sigma_N^2)}{\Gamma(2M-2)} \quad . \tag{8.1-29}$$

Another method to implement jammed symbol detection in the presence of a signal would utilize a combination of the individual channel jammed detector outputs. For example, in the case of M=4 we can employ the combinatorial logic of any two individual channel detectors' outputs "ANDed" to produce a symbol jamming decision from the six possible combinations. The detection of the "any channel jammed" condition will be the logic variable

$$\begin{aligned} J_{symbol} &= (J_1 \cdot J_2) + (J_1 \cdot J_3) + (J_1 \cdot J_4) \\ &\quad + (J_2 \cdot J_3) + (J_2 \cdot J_4) + (J_3 \cdot J_4) \end{aligned} \tag{8.1-30}$$

where it is assumed that detection of the signal itself has triggered the decision of a jammed channel (J_i) for a particular channel. Thus, we are able to realize a jammed symbol detection when a signal is present.
For the case of any M we have the overall false alarm probability (when no signal is present)

$$P_{FA(sym)} = \sum_{n=2}^{M} \binom{M}{n} P_{FA(CH)}^{n} \left(1 - P_{FA(CH)}\right)^{M-n} \tag{8.1-31}$$

which simplifies to

$$P_{FA(sym)} \approx \binom{M}{2} P^2_{FA(CH)} \tag{8.1-32}$$

as the overall false-alarm rate for a jammed symbol detection.

8.2 ASSESSMENT OF IMPLEMENTATION EFFECTS

In this second part of our implementation studies, we investigate the effects of the previously discussed implementation schemes on the performance of the ECCM receivers. We demonstrate the necessary adjustments involved in reformulating the probability of error expressions which are now conditioned on estimated (measured) parameters instead of assumed "perfect" measurement quantities. Our objective is to assess the "return on the investment" realized by resorting to the complex measurement schemes needed to implement the AGC receivers, which for ideal measurements achieve the best ECCM receiver performances in worst-case PBNJ. That is, we seek to answer the question, "Will practical implementations of the AGC receivers continue to outperform the simple SNORM and hard-decision receivers, which require no measurement?"

8.2.1 Methodology for Direct Assessment.

Analyses of the error performance of the implementated AGC schemes are extremely difficult for the following reasons. First, we must account for the measured (estimated) quantities $\hat{\sigma}_N^2$ and $\hat{\sigma}_T^2$ as random variables imbedded in the probability of error expressions. Second, the effect of errors in the JSI decisions upon the P(e) expressions must also be considered. Previous analytical results in this report were derived on the assumption that perfect measurements were obtained for σ_N^2 and σ_T^2; this provided a lower bound on the error performance to be realized in practice. This ideal total error probability can be expressed parametrically by

$$P(e) \equiv P(e;\underbrace{\gamma,E_b/N_0,E_b/N_J,L,M}_{\substack{\text{actual}\\ \text{parameters}}};\underbrace{\sigma_N^2,\sigma_T^2,S}_{\substack{\text{parameters}\\ \text{assumed by}\\ \text{receivers}}}) \qquad (8.2\text{-}1)$$

and can be written as

$$P(e) = \sum \Pr\{[\nu]\}\, P_b(e|[\nu]), \qquad (8.2\text{-}2)$$

where $[\nu]$ is a matrix describing the jamming event (see Section 2.2) over the L-hop diversity. The above expressions must now be restated as,

$$P(e) \equiv P(e;\gamma,E_b/N_0\ ,\ E_b/N_J\ ,\ L,\ M;\ \hat{\sigma}_N^2,\hat{\sigma}_T^2,\hat{S}) \qquad (8.2\text{-}3)$$

$$P(e) = \sum \Pr\{[\nu],[\hat{\nu}]\}\ P(e|[\nu],[\hat{\nu}]) \qquad (8.2\text{-}4)$$

in accounting for $\hat{\sigma}_N^2$, $\hat{\sigma}_T^2$, $\hat{S}$, as well as estimates in JSI.

For example, in the ideal situation the IC-AGC receiver decision variables are

$$z_m = \sum_{k=1}^{L} z_{mk}\ ; \qquad m=1,2,\ldots,M; \qquad (8.2\text{-}5a)$$

where each z_{mk} is the square-law envelope detector sample x_{mk}^2 multiplied by the weight

$$w_{mk} = \begin{cases} 1/\sigma_N^2\ , & \text{channel m not jammed on hop k} \\ \\ 1/\sigma_T^2\ , & \text{channel m jammed on hop k.} \end{cases} \qquad (8.2\text{-}5b)$$

This ideal normalization results in the $\{z_{mk}\}$ being all unscaled chi-square random variables, as discussed in Section 4.

Now if the a priori quantities σ_N^2 and σ_T^2 are not available and the jamming condition of the channels is not known, we must use estimates $\hat{\sigma}_N^2$ and $\hat{\sigma}_T^2$, and also decide whether the channel is jammed. This results in the weights

$$\hat{w}_{mk} = \begin{cases} (\hat{\sigma}_N^2)^{-1}(1-P_{Fm}) + (\hat{\sigma}_T^2)^{-1} P_{Fm} = W_0, \text{ not jammed;} \\ (\hat{\sigma}_N^2)^{-1}(1-P_{Dm}) + (\hat{\sigma}_T^2)^{-1} P_{Dm} = W_1, \text{ jammed;} \end{cases} \quad (8.2\text{-}6)$$

where P_{Fm} and P_{Dm} are per-channel jamming false alarm and detection probabilities. In this description, it is assumed that the variance estimates are developed from look-ahead or in-band measurement data prior to the symbol being processed, and that the channel jamming decision is based on a one-hop look-ahead scheme. In the absence of signals in the look-ahead channels,

$$P_{Fm} = P_F(\hat{\sigma}_N^2, \hat{\sigma}_T^2; \sigma_N^2, \sigma_T^2) \text{ and } P_{Dm} = P_D(\hat{\sigma}_N^2, \hat{\sigma}_T^2; \sigma_N^2, \sigma_T^2);$$

that is, these probabilities are the same for each of the M symbol channels.

Thus, after accumulating the L hops, the decision statistics are, conditioned on jamming events and measurements,

$$z_1 = W_0 \sigma_N^2 \chi^2[2(L-\ell_1); 2(L-\ell_1)\rho_N] + W_1 \sigma_T^2 \chi^2[2\ell_1; 2\ell_1 \rho_T]; \quad (8.2\text{-}7a)$$

$$z_m = W_0 \sigma_N^2 \chi^2 [2(L-\ell_m)] + W_1 \sigma_T^2 \chi^2(2\ell_M),\ m>1; \quad (8.2\text{-}7b)$$

where $\underline{\ell} = (\ell_1, \ell_2, \ldots, \ell_M)$ is the jamming event vector of the number of hops jammed in each channel. Recall now that the linear combining receiver (LCR) has the normalized decision statistics

$$u_1 = z_1/\sigma_N^2 = \chi^2[2(L-\ell_1);2(L-\ell_1)\rho_N]$$

$$+ K\chi^2[2\ell_1;2\ell_1\rho_T]; \tag{8.2-8a}$$

$$u_m = z_m/\sigma_N^2 = \chi^2[2(L-\ell_m)] + K\chi^2(2\ell_m),\ m>1; \tag{8.2-8b}$$

where $K = \sigma_T^2/\sigma_N^2$. We therefore recognize that, conditioned on the measurements, the implemented IC-AGC receiver's BER will have the same functional form as the LCR's with the new K value

$$K' = \frac{\sigma_T^2 W_1}{\sigma_N^2 W_0} = \frac{\sigma_T^2}{\sigma_N^2} \cdot \frac{\hat{\sigma}_T^2(1-P_D) + \hat{\sigma}_N P_D}{\hat{\sigma}_T^2\ (1-P_F) + \hat{\sigma}_N^2 P_F}. \tag{8.2-9}$$

Evaluation of the effect of the measurements and JSI decisions then involves numerically averaging the LCR error probability (with K replaced by (8.2-9)) over the distributions of $\hat{\sigma}_N^2$ and $\hat{\sigma}_T^2$.

8.2.2 <u>In Search of an Upper Bound</u>: <u>Simplified Measurement Models</u>.

Equations (8.2-7) to (8.2-9) reveal that the implemented IC-AGC receiver statistics more or less tend toward those of the ineffective LCR, depending on the quality of the measurements. This fact underscores the important role of the *a priori* information utilized by the AGC receivers in their superior performance. We can take the position that the ideal AGC performances calculated in this report represent a lower bound on achievable BER, though perhaps not the lowest bound,* and then seek an upper bound instead of attempting the arduous and time-consuming direct analysis of implemented systems. Such an upper bound, if sufficiently tight, would be suitable for comparison with the BER results for the receivers not employing a priori information.

How shall we obtain an upper bound? Since the performance degradation associated with the receiver implementations is related to the quality of the measurements, we realize that any bound would be directly identified with a particular measurement approach, and parametric in the features of that approach (such as number of samples taken). We fully anticipate, for example, that an upper bound on the implemented system's BER would decrease as the number of samples used in the measurement increases. Therefore, it is reasonable to consider possible implementations utilizing one sample as candidates for systems whose performances represent an upper bound on what is achievable in the same manner as the ideal systems represent a lower bound.

For the ACJ-AGC FH/MFSK receiver, we consider the simplified version that we call the "practical ACJ" (PACJ) receiver. Since the ideal ACJ receiver uses the weights $w_k = (\max_m \sigma_{mk}^2)^{-1}$, we stipulate that the PACJ uses the weights

$$w_k = (\max_m x_{mk}^2). \qquad (8.2\text{-}10)$$

*In Section 7 it was observed that the SNORM receiver can outperform the AGC receivers in some, limited circumstances.

This approach in effect utilizes the received square-law envelope detector samples themselves as (one sample) measurements of the noise power in each channel, and thus is a form of in-band measurement which is very simple indeed compared to schemes discussed in Section 8.1.

We note in passing that this PACJ receiver is related to the hard-decision receiver in the following way: the HD decision statistics $\{z_{mk}\}$ are the PACJ decision statistics after being subjected to a two-level quantization. That is,

$$z_{HD} = \begin{cases} 1, & z_{PACJ} > 1 \\ 0, & z_{PACJ} < 1. \end{cases} \tag{8.2-11}$$

For the IC-AGC receiver, we postulate that a one-hop look-ahead implementation yields one detector sample in each of the M channels just prior to the actual symbol's occupancy of those channels.* Since the ideal IC-AGC uses the weights $w_{mk} = (\sigma_{mk}^2)^{-1}$, we can treat the look ahead samples $\{v_{mk}\}$ as one-sample estimates of noise power and specify the "practical IC" (PIC) weights

$$w_{mk} = (v_{mk})^{-1} . \tag{8.2-12}$$

In what follows, we evaluate each of these practical receivers for the case of M=2, L=2 in order to compare them with their respective ideal receivers. We also can consider the SNORM and hard-decision receivers as

*Alternatively, the hopping and symbol rates could be reduced by one-half in order to sample the channel first, then receive the transmission; look-ahead is avoided in this way, at the expense of data rates.

"practical" AGC implementations, and will continue to exhibit their BER results for comparison purposes.

8.2.3 Example Evaluations of Practical AGC Receivers.

We now find the error probabilities for the PACJ and PIC receivers for M=2 and L=2.

8.2.3.1 Analysis of the PACJ receiver.

The PACJ receiver for M=2 is diagrammed in Figure 8.2-1. The decision statistics are

$$z_m = \sum_{k=1}^{L} \frac{x_{mk}^2}{\max\limits_m (x_{mk}^2)}, \quad m = 1,2. \tag{8.2-13}$$

In Appendix C it is shown that this receiver for L=2 and FH/RMFSK has the bit error probability

$$P_b(e) = 2 \int_0^1 dx\, f(x)G(x) + G^2(1) \tag{8.2-14a}$$

where $K = \sigma_T^2/\sigma_N^2$ and

$$\begin{aligned} f(x) &= \pi_0 \frac{1}{(x+1)^2} \exp\left\{-\frac{\rho_N x}{(x+1)}\right\}\left[1+\frac{\rho_N}{x+1}\right] \\ &+ \pi_1 \frac{K}{(Kx+1)^2} \exp\left\{-\frac{K\rho_T x}{(Kx+1)}\right\}\left[1+\frac{\rho_T}{Kx+1}\right] \\ &+ \pi_1 \frac{K}{(x+K)^2} \exp\left\{-\frac{\rho_N x}{(x+K)}\right\}\left[1+\frac{K\rho_N}{x+K}\right] \\ &+ \pi_2 \frac{1}{(x+1)^2} \exp\left\{-\frac{\rho_T x}{(x+1)}\right\}\left[1+\frac{\rho_T}{x+1}\right]. \end{aligned} \tag{8.2-14b}$$

and

$$\begin{aligned} G(x) &= \pi_0 \frac{x}{x+1} e^{-\rho_N/(1+x)} + \pi_2 \frac{x}{x+1} e^{-\rho_T/(1+x)} \\ &+ \pi_1 \frac{Kx}{Kx+1} e^{-\rho_N/(1+Kx)} + \pi_1 \frac{x}{x+K} e^{-K\rho_T/(x+K)}. \end{aligned} \tag{8.2-14c}$$

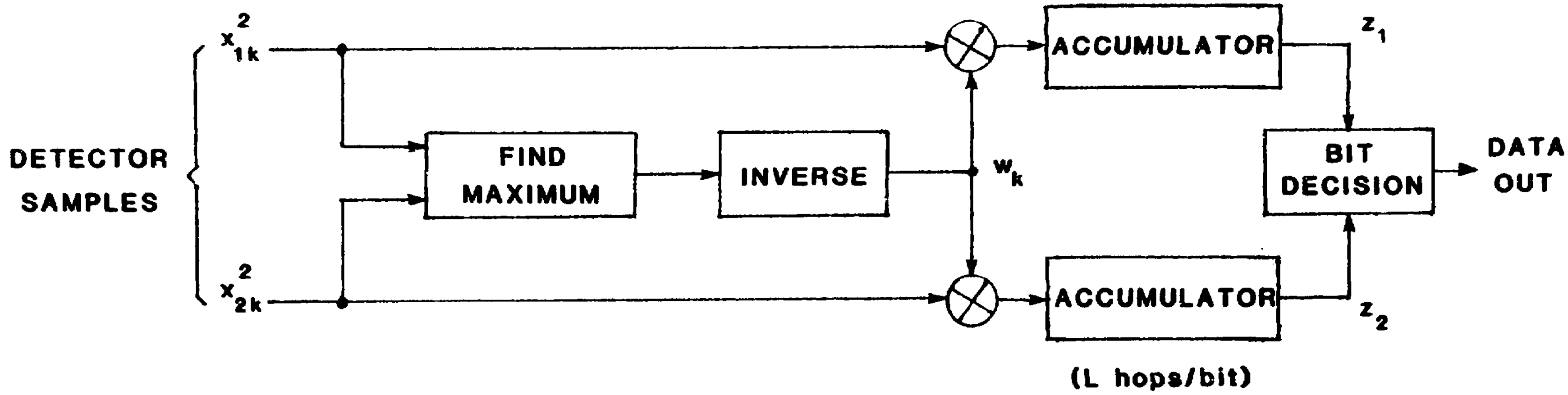


FIGURE 8.2-1 PRACTICAL ACJ-AGC(PACJ) RECEIVER FOR FH/RMFSK WHEN M=2

Numerical results for the L=2 PACJ binary receiver are shown for E_b/N_0 = 13.35 dB and 20 dB, respectively, in Figures 8.2-2 and 8.2-3. There is no difference between these results and those of the SNORM receiver that can be discerned from the figures - a close look at the data reveals a slight difference except for γ=1 (full-band jamming), for which analysis shows that the two receivers yield identical performance for the M=2, L=2 case.

8.2.3.2 Analysis of the PIC receiver.

The PIC receiver for M=2 is diagrammed in Figure 8.2-4. The decision statistics are

$$z_m = \sum_{k=1}^{L} \frac{u_{mk}}{v_{mk}} \quad , \quad m=1,2; \tag{8.2-15}$$

where the $\{u_{mk}\}$ are the usual receiver samples and the $\{v_{mk}\}$ are look-ahead samples. These look-ahead samples are assumed to have the same noise powers as their corresponding "usual" samples, and not to have a signal present.

In Appendix C it is shown that the L=2 FH/RMFSK or FH/MFSK PIC receiver error probability in partial-band noise jamming is given by

$$\begin{aligned} P_b(e) = {} & (1-\gamma)^2 \; P_b(e|\rho_1=\rho_2=E_b/2N_0) \\ & + 2\gamma(1-\gamma)P_b(e|\rho_1=E_b/2N_0,\rho_2=E_b/2N_T) \\ & + \gamma^2 P_b(e|\rho_1=\rho_2=E_b/2N_T), \end{aligned} \tag{8.2-16a}$$

where γ is the jamming fraction and

$$\begin{aligned} P_b(e|\rho_1,\rho_2) = {} & \int_0^1 du \int_0^1 dv \; e^{-\rho_1 u-\rho_2 v} (1+\rho_1-\rho_2 u)(1+\rho_2-\rho_2 v) \\ & \times \frac{uv}{u+v}\left[2 + \frac{uv}{u+v}\,\ell n\left(\frac{u+v}{uv} - 1\right)\right]. \end{aligned} \tag{8.2-16b}$$

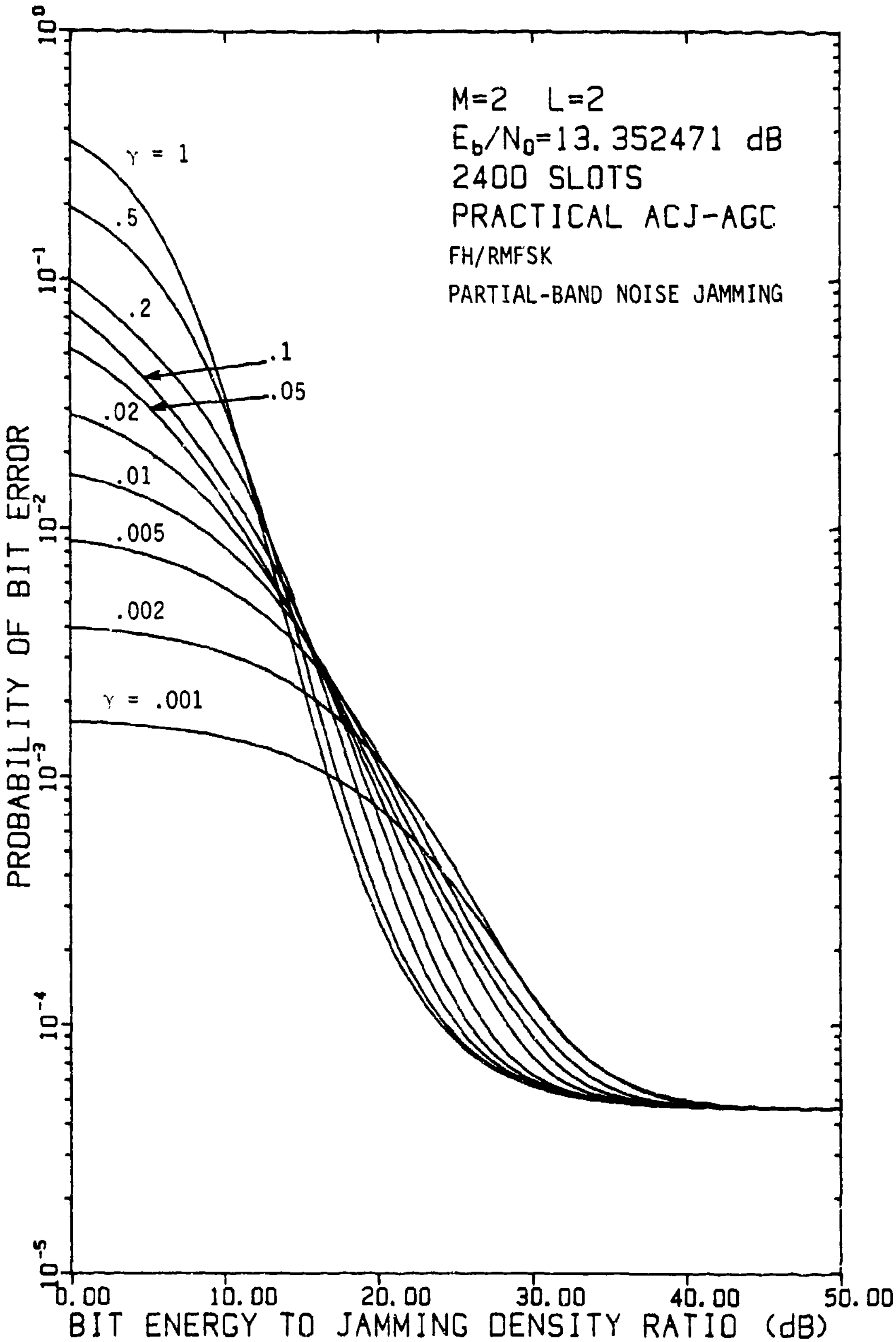


FIGURE 8.2-2 BER PERFORMANCE FOR PACJ FH/RMFSK RECEIVER IN PARTIAL-BAND NOISE JAMMING FOR M=2, L=2, AND E_b/N_0 = 13.35 dB

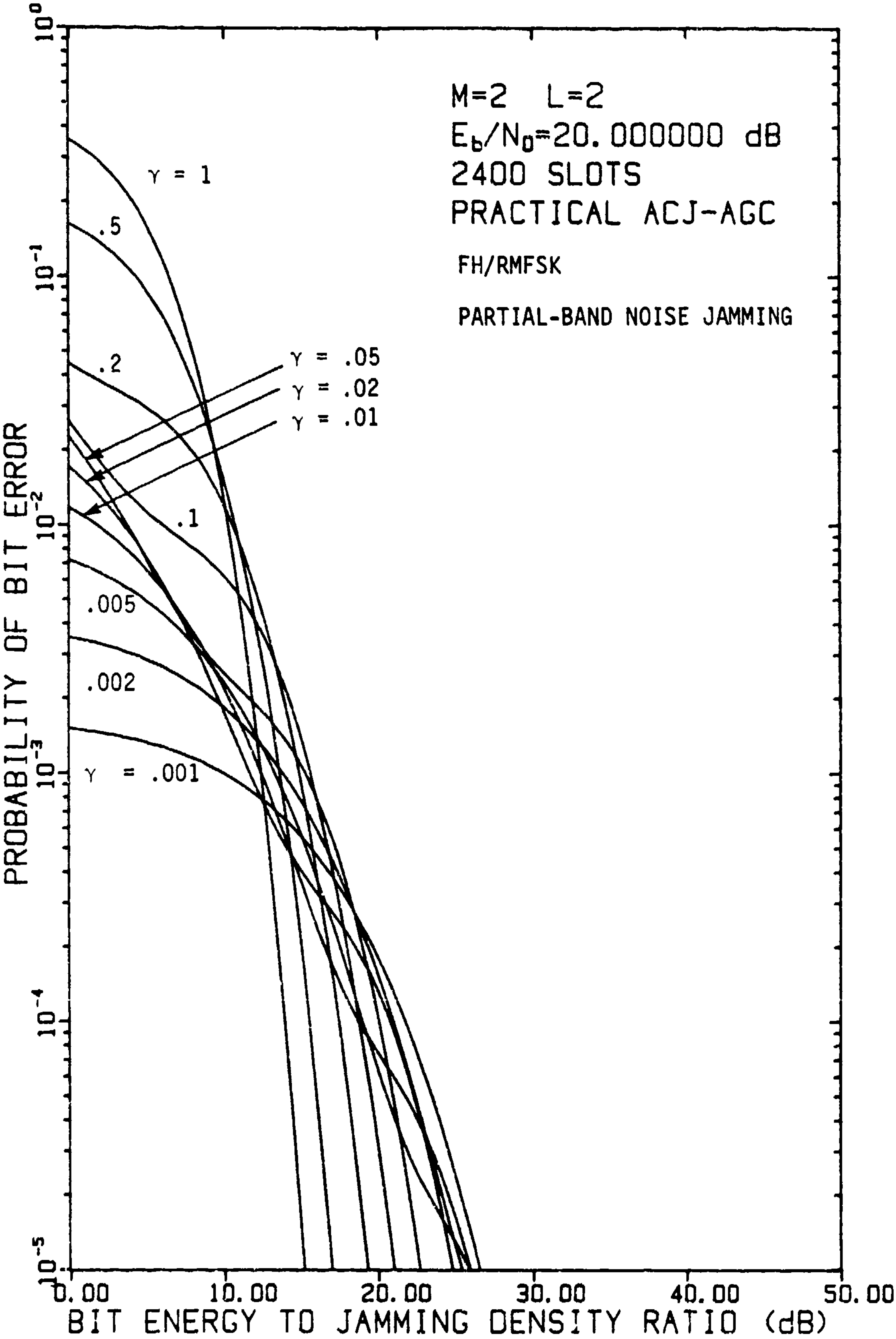


FIGURE 8.2-3 BER PERFORMANCE FOR PACJ FH/RMFSK RECEIVER IN PARTIAL-BAND NOISE JAMMING FOR M=2, L=2, AND E_b/N_0 = 20 dB

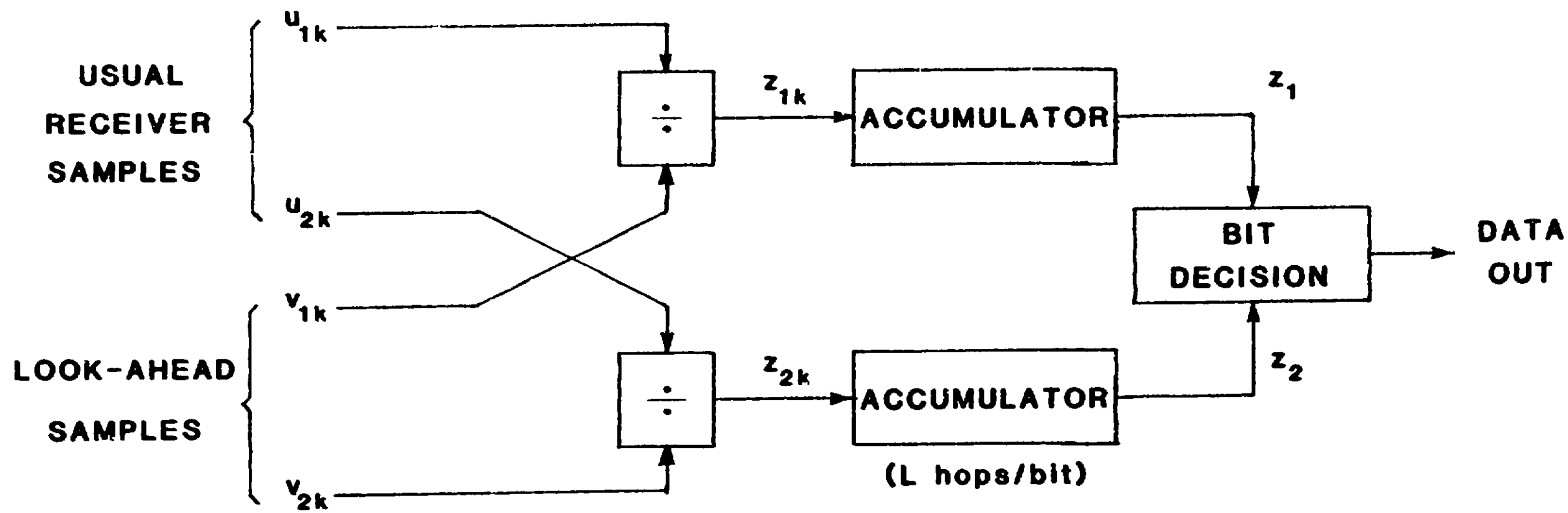


FIGURE 8.2-4 PRACTICAL IC-AGC(PIC) RECEIVER FOR FH/RMFSK WHEN M=2

Numerical results for the L=2 PIC binary receiver reveal that it performs very poorly. For example, for full-band noise jamming the BER varies with E_b/N_J as shown in Figure 8.2-5. We note that for high E_b/N_J (practically no jamming) the error probability is greater than 10^{-4} even for thermal noise so small that E_b/N_0 = 40 dB. Evidently the predictably poor quality of the one-sample estimate of noise variance is especially damaging when using it in the denominator of the ratios taken in (8.2-15).

With slightly more effort, we find that if the PIC receiver uses <u>two</u> look-ahead noise samples, summed to obtain a better variance estimate, the performance improves considerably. The decision statistics for this version of the PIC receiver are

$$z_m = \sum_{k=1}^{L} \frac{u_{mk}}{v_{mk1}+v_{mk2}} \quad , \ m=1,2. \qquad (8.2\text{-}17)$$

Using the same analytical approach as in Appendix C, but with the sum of the look-ahead variables being $\sigma_{mk}^2 \chi^2(4)$ distributed, we find that the conditional P(e) for M=2 and L=2 is

$$P_b(e|\rho_1,\rho_2) = 4\int_0^1 du \int_0^1 dv \ (1-u)(1-v) \ \exp\{-\rho_1(1-u)-\rho_2(1-v)\}$$

$$\times \ [1+2\rho_1^2 u + \rho_1^2 \ u^2/2] \ [1+2\rho_2^2 v + \rho_2 v^2/2]$$

$$\times \ g(\frac{u}{1-u} + \frac{v}{1-v}) \ , \qquad (8.2\text{-}18a)$$

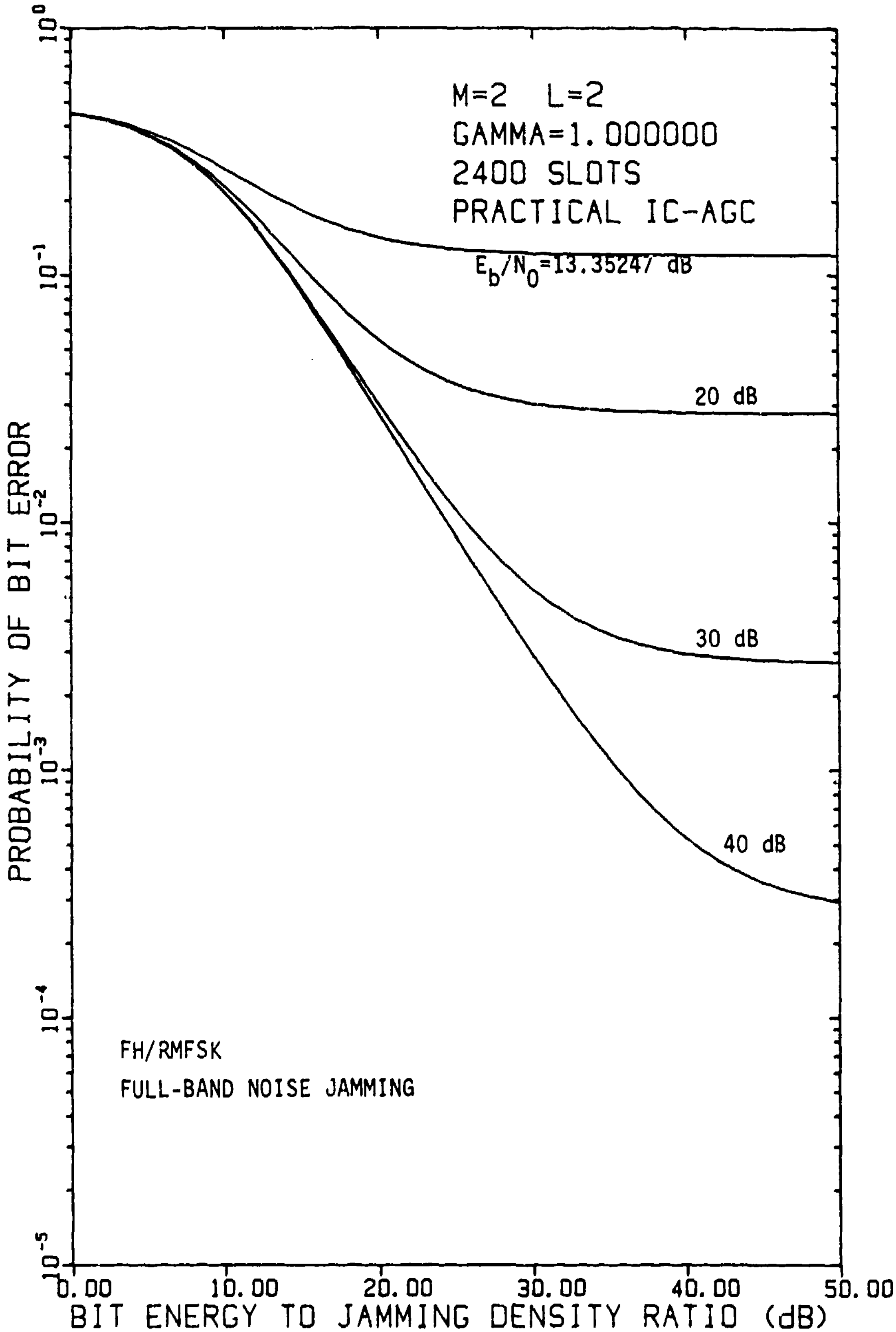


FIGURE 8.2-5 PERFORMANCE OF THE PRACTICAL IC-AGC FH/RMFSK RECEIVER IN FULL-BAND NOISE JAMMING FOR M=L=2.

where

$$g(x) = \frac{12}{(x+2)^4} \ \ln(x+1) + \frac{2(x^2+7x+4)}{(x+1)(x+2)^3} \ . \qquad (8.2\text{-}18b)$$

Numerical results for full-band jamming are shown in Figure 8.2-6. For E_b/N_0 = 13.35 dB, there is not much of an improvement, but for 20 dB and higher E_b/N_0, there is about a decade improvement over the asymptotic BER obtained using a one-sample noise estimate. Clearly as more look-ahead samples are used, the PIC receiver will act more like the IC-AGC. However, if more measurements are taken, the likelihood increases that the measurement will suffer from changes in the assumed noise environment over the measurement time. Perhaps with the RMFSK hopping scheme it is possible to gather noise samples from adjacent (unused) hopping slots in addition to (or in place of) using a look-ahead approach.

8.3 CONCLUSIONS AND RECOMMENDATIONS

Having considered practical implementations of ECCM receivers for FH/RMFSK in worst-case partial-band noise jamming (WCPBNJ), we are able to conclude our study with some "lessons learned," from which we also can recommend further studies.

8.3.1 Knowledge Gained from Study.

8.3.1.1 Ideal receiver performances.

The ideal receiver performances obtained in Sections 3-6 and compared in Section 7 show for the first time what the expected performance of random frequency-hopping MFSK is in WCPBNJ when L hops per symbol soft decisions are

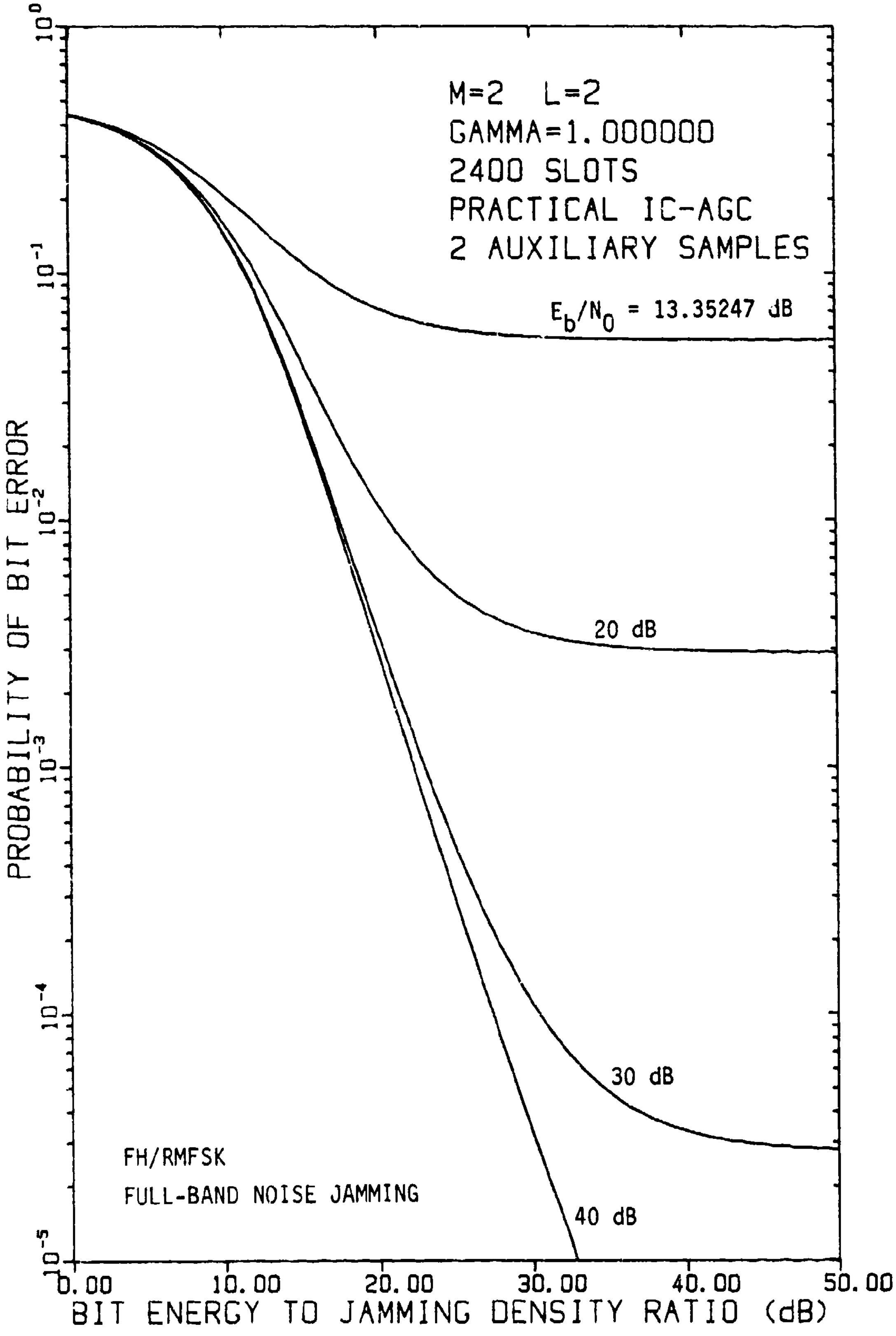


FIGURE 8.2-6 PERFORMANCE OF THE PRACTICAL IC-AGC FH/RMFSK RECEIVER USING A TWO-SAMPLE NOISE ESTIMATE IN FULL-BAND NOISE JAMMING FOR M=L=2.

used and thermal noise is not neglected. We have learned that per-channel adaptive normalization, such as envisioned using the IC-AGC receiver weighting scheme, is effective in countering the jamming effects which are most damaging to the RMFSK performance: jamming power in non-signal symbol frequency channels and not in the signal channel. The IG-AGC scheme controls the channel gains in such a way as to equalize the a priori noise powers in the channels, in effect forcing the non-Gaussian input noise process to be a Gaussian process. The jamming then only affects the error through the reduction of relative signal power when jammed signal hops are normalized, and RMFSK using this scheme performs the same as the conventional MFSK hopping.

We have learned also that per-symbol adaptive normalization, typified by the ACJ-AGC receiver's weighting scheme, is effective in countering WCPBNJ, though not as effective as per-channel normalization. The per-symbol operation equalizes the maximum of the M channels' a priori noise powers to a constant value, but does not affect the relative powers among the M channels, so that to a certain extent the RMFSK system, unlike MFSK, is still subject to noise power imbalances on each hop and therefore is more vulnerable than MFSK. However, the per-symbol normalization does prevent jammed hops from dominating the soft-decision, and therefore achieves an ECCM or anti-jam diversity effect.

The per-channel and per-symbol AGC schemes perform the same for no jamming, and the performance achieved is sensitive to the amount of thermal and/or background noise present at the receiver, expressed relatively in our study by the ratio E_b/N_0. For any finite E_b/N_0 value, the receiver performances for weak or no jamming degrade in proportion to L, the number of hops/symbol, due to noncoherent combining losses. It cannot be emphasized too strongly that thermal noise should always be included in any study, because different implementations of the ideal receiver combining schemes in general are subject to different noncoherent combining losses plus any losses due to quantization effects. The ideal AGC performances we have obtained provide a lower bound on achievable performance in the sense that errorless noise power measurements are assumed.

8.3.1.2 Practical receiver performances.

The several "practical" FH/RMFSK receiver combining schemes we have studied may be classified as implementations of either per-channel (IC-AGC) or per-symbol (ACJ-AGC) ideal schemes.

The clipper receiver implements a per-channel ECCM scheme and thus to a certain extent achieves a performance in WCPBNJ whose parametric behavior follows that of the IC-AGC. However, its best clipping threshold value is parametric in received signal and thermal noise powers and in L. In our study we have calculated performances assuming that these powers are known *a priori*; it is expected that estimation of the correct threshold will degrade its performance. But if we can assume that a reasonably good estimate of unjammed SNR is available, the clipper receiver appears to be a viable candidate for a practical ECCM receiver.

It was shown that direct implementation of the IC-AGC using auxiliary noise power measurements has the potential for approaching the IC-AGC performance, but only if certain assumptions are made: (a) the receiver complexity can be further advanced economically to include the required noise power measurements (as many as possible per channel); (b) the noise processes being measured are relatively stationary during the time of measurement and not subject to corruption from the signal or other signals. These assumptions are quite restrictive, so that we would choose to implement another, simpler scheme if its performance is satisfactory.

The per-symbol receiver ECCM schemes studied include the self-normalizing receiver (SNORM) and the so-called "practical ACJ" (PACJ). These receiver implementations are very simple, requiring only operations using the usual received envelope samples in the M symbol frequency channels. Somewhat surprisingly, these two schemes perform very well (nearly identically for M=2 and L=2), even better than the supposedly best ideal IC-AGC receiver under certain limited circumstances. Therefore, if the receivers using *a priori* noise and jamming information tend to represent a lower bound on system performance, and the small-sample size "practical" receivers, an upper bound--then we have observed a situation when lower and upper bounds converge to agree upon a predicted performance result. The implications are that we may regard the easily-calculated IC-AGC performance as representative of achievable system performance, with perhaps a slight implementation loss of a few dB when the simple practical receivers are employed, and when the system noise without jamming is small (high E_b/N_0).

8.3.1.3 RMFSK vs MFSK.

We have found that for smaller alphabet sizes (M=2 or 4), the error performance of FH/RMFSK in worst-case PBNJ is comparable to that of the conventional FH/MFSK, when appropriate receiver processing is employed, to the extent that we state that the price to be paid for the additional RMFSK system complexity can be assessed against the threat of follow-on noise jamming. That is, if follow-on jamming is not considered a threat, MFSK should be used; but if it is a threat, RMFSK is an effective counter-countermeasure that also works satisfactorily in the worst-case partial-band noise jamming environment.

8.3.2 Recommendations.

With the perspective gained from our study we make the following recommendations for further research.

(a) Derivation of system error performances using the PACJ and similar "nonparametric" ECCM receivers; it is conjectured that analysis would yield BER expressions for M>2 that are more feasible for computation than those for the clipper and self-normalizing receivers we have studied.

(b) Analysis of FH/RMFSK performance under multi-tone jamming; it has been asserted that FH/RMFSK "precludes" systematic tone jamming, but what, in quantitative terms, is its vulnerability to tone jamming, relative to FH/MFSK?

(c) Analysis of mutual interference effects in an FH/RMFSK system; these are considered to be more numerous than for FH/MFSK, and possibly more damaging - a more intricate setup for multiple users may be necessary to avoid mutual interference.

APPENDIX A

PROBABILITY DENSITY FUNCTIONS FOR SOFT DECISION RECEIVER STATISTICS

For soft decision receivers with multiple hops per symbol, the M decision statistics are of the form

$$z_m = \sigma_{1m}^2 \ \chi^2(\nu_{1m};\ \lambda_{1m}) + \sigma_{2m}^2 \ \chi^2(\nu_{2m};\ \lambda_{2m}), \tag{A-1}$$

where $\chi^2(\nu;\ \lambda)$ denotes a noncentral chi-squared random variable with ν degrees of freedom and noncentrality parameter λ, and σ_{1m}^2 and σ_{2m}^2 are different scalings. For the cases to be studied we can also write

$$u_m = z_m/\sigma_{1m}^2 = \chi^2(2L - 2\ell_m;\ \lambda_{1m}) + K_m \ \chi^2(2\ell_m;\ \lambda_{2m}), \tag{A-2}$$

since $\nu_{1m}+\nu_{2m} = 2L$, twice the number of hops per symbol, and $\nu_{2m} = 2\ell_m$, twice the number of jammed hops for symbol channel m. Also, without loss of generality we designate the first (m = 1) channel as the one containing the transmitted signal, and the others ($m \geq 2$) as containing noise only. Thus

$$\lambda_{11} = 2(L - \ell_1)S/\sigma_{11}^2, \ \lambda_{21} = 2\ell_1 S/\sigma_{21}^2 \tag{A-3a}$$

and

$$\lambda_{1m} = \lambda_{2m} = 0, \ m \geq 2. \tag{A-3b}$$

Previously it has been shown [1, Appendix 2E] that the probability density function (pdf) for u_m given by (A-2) is $p_u(\alpha;\ \ell_m,\ \lambda_{1m},\ \lambda_{2m},\ K_m)$, where

$$p_u(\alpha;\ \ell,\ \lambda_1,\ \lambda_2,\ K)$$

$$= \frac{K^{-\ell}}{2}\ \exp\left(-\frac{\lambda_1+\lambda_2+\alpha}{2}\right)\sum_{k=0}^{\infty}\ \sum_{r=0}^{\infty}\ \frac{\left(\frac{\lambda_1}{2}\right)^k\left(\frac{\lambda_2}{2K}\right)^r\left(\frac{\alpha}{2}\right)^{k+r+L-1}}{k!\ r!\ (k+r+L-1)!}$$

$$\times {}_1F_1\left(r+\ell;\ k+r+L;\ \frac{K-1}{K}\cdot\frac{\alpha}{2}\right), \tag{A-4}$$

where ${}_1F_1(a;\ b;\ x)$ is the confluent hypergeometric function. The computation of this expression is very time consuming; in this Appendix we consider alternative expressions that can be computed more quickly, or perhaps be amenable to approximations.

By expanding the confluent hypergeometric function in its series form,

$${}_1F_1\left(r+\ell;\ k+r+L;\ \frac{K-1}{K}\ \frac{\alpha}{2}\right) \tag{A-5}$$

$$= \sum_{n=0}^{\infty}\ \left(\frac{K-1}{K}\cdot\frac{\alpha}{2}\right)^n\ \frac{1}{n!}\ \frac{(r+\ell)_n}{(k+r+L)_n},$$

with

$$(a)_n = \Gamma(a+n)/\Gamma(a), \tag{A-6}$$

and summing over the index k, we obtain the expression

$$p_u(\alpha) = \frac{K^{-\ell}}{2}\exp\left(-\frac{\lambda_1+\lambda_2+\alpha}{2}\right)\sum_{n=0}^{\infty}\left(\frac{K-1}{K}\right)^n\ \frac{1}{n!}$$

$$\times\ \sum_{r=0}^{\infty}\ \frac{\left(\frac{\lambda_2}{2K}\right)^r}{r!}\ (r+\ell)_n\left(\frac{\alpha}{\lambda_1}\right)^{(r+n+L-1)/2}\ I_{r+m+L-1}\left(\sqrt{\alpha\lambda_1}\right) \tag{A-7}$$

This form is based on recognizing the series for $I_{r+n+L-1}(x)$, the modified Bessel function of the first kind of order r+n+L-1:

$$I_{r+n+L-1}(x) = \sum_{k=0}^{\infty} \frac{(x/2)^{2k+r+n+L-1}}{k!\,(k+r+n+L-1)!} \quad . \tag{A-8}$$

Further, we recognize that

$$\left(\frac{\alpha}{\lambda_1}\right)^{(q-1)/2} I_{q-1}(\sqrt{\alpha\lambda_1}) = 2e^{(\alpha+\lambda_1)/2} p_{\chi^2}(\alpha;\, 2q,\, \lambda_1), \tag{A-9}$$

where $p_{\chi^2}(\alpha; \nu, \lambda)$ is the pdf for a noncentral chi-squared random variable with ν degrees of freedom and noncentrality parameter λ. This allows us to write

$$p_u(\alpha) = K^{-\ell} e^{-\lambda_2/2} \sum_{n=0}^{\infty} \frac{\left(\frac{K-1}{K}\right)^n}{n!} \sum_{r=0}^{\infty} \frac{\left(\frac{\lambda_2}{2K}\right)^r}{r!} (r+\ell)_n$$

$$\times\, p_{\chi^2}(\alpha;\, 2r+2n+2L;\, \lambda_1) \tag{A-10}$$

Now we concentrate on the summation over the indices r and n. Since

$$\sum_{n=0}^{\infty} \sum_{r=0}^{\infty} f(n,\, r)\, g(n+r)$$

$$= \sum_{n=0}^{\infty} g(n) \sum_{r=0}^{n} f(n-r,\, r), \tag{A-11}$$

we can express the pdf in the form

$$\boxed{p_u(\alpha) = \sum_{n=0}^{\infty} c_n \, p_{\chi^2}(\alpha;\ 2L+2n,\ \lambda_1),} \tag{A-12}$$

where

$$c_n = e^{-\lambda_2/2} \left(\frac{K-1}{K}\right)^n \frac{1}{K^\ell} \sum_{r=0}^{n} \frac{\left(\frac{\lambda_2/2}{K-1}\right)^r}{r!} \frac{(r+\ell)_{n-r}}{(n-r)!}. \tag{A-13}$$

Now,

$$\frac{(r+\ell)_{n-r}}{(n-r)!} = \frac{(r+\ell+n-r-1)!}{(r+\ell-1)!(n-r)!}$$

$$= \binom{n+\ell-1}{n-r}; \tag{A-14}$$

and

$$\sum_{r=0}^{n} \binom{n+a}{n-r} \frac{(-x)^r}{r!} = \mathcal{L}_n^a(x), \tag{A-15}$$

with $\mathcal{L}_n^a(x)$ the generalized Laguerre polynomial. Thus,

$$\boxed{c_n = e^{-\lambda_2/2} \left(\frac{K-1}{K}\right)^n \frac{1}{K^\ell} \mathcal{L}_n^{\ell-1}\left[-\frac{\lambda_2/2}{K-1}\right]} . \tag{A-16}$$

Since (A-12) is a pdf, it must integrate over α to unity. This requires that

$$\sum_{n=0}^{\infty} c_n = 1. \tag{A-17}$$

In fact, this is so, since [3, eq. 8.975.1]

$$\sum_{n=0}^{\infty} b^n \mathcal{L}_n^a(x) = (1-b)^{-a-1} \exp\left\{\frac{bx}{b-1}\right\}$$

$$= K^{\ell} e^{\lambda_2/2}. \tag{A-18}$$

Approximation

The expression (A-12) for the pdf is in the form of a series of weighted chi-squared pdf's. This suggests an approximation based on truncating the series:

$$p_u(\alpha) \approx \sum_{n=0}^{N} c_n \, p_{\chi^2}(\alpha;\, 2L+2n,\, \lambda_1) \Big/ \sum_{n=0}^{N} c_n. \tag{A-19}$$

Since, even for $\lambda_2 = 0$,

$$\frac{c_{n+1}}{c_n} = \frac{1}{n}\,\frac{\left(\frac{K-1}{K}\right)}{1+1/n} \cdot \frac{1+\ell/n}{1+(\ell-1)/n} \longrightarrow \frac{\left(\frac{K-1}{K}\right)}{n} < \frac{1}{n}, \tag{A-20}$$

the truncation is feasible but, depending on the value of K, the convergence of the weights $\{c_n\}$ is slow.

APPENDIX B

COMBINATORIAL RELATIONS FOR JAMMING EVENT ENUMERATION

B.1 NUMBERS OF ORDERED VECTORS

We define

$$(L+1)S_M(L) \triangleq \#\{\underline{\ell}' : \underline{\ell}' = (\ell_1; \ell_2 \le \ell_3 \le \ldots \le \ell_M)\}$$
$$= (L-1)\sum_{\ell_M=0}^{L}\ \sum_{\ell_{M-1}=0}^{\ell_M} \cdots \sum_{\ell_2=0}^{\ell_3} (1)\ . \tag{B.1-1}$$

By direct manipulation,

$$S_2(L) = \sum_{\ell_2=0}^{L} 1 = L+1 = \binom{L+1}{1} \tag{B.1-2}$$

$$S_3(L) = \sum_{\ell_3=0}^{L}\ \sum_{\ell_2=0}^{\ell_3} 1 = \sum_{\ell_3=0}^{L} S_2(\ell_3)$$
$$= \sum_{\ell_3=0}^{L} \binom{\ell_3+1}{1} = \binom{L+2}{2}\ , \tag{B.1-3}$$

using [3, equation 0.151.1]. Assuming that

$$S_M(L) = \binom{L+M-1}{M-1}\ , \tag{B.1-4}$$

we find that

$$S_{M+1}(L) = \sum_{\ell_{M+1}=0}^{L} S_M(\ell_{M+1})$$

$$= \sum_{\ell_{M+1}=0}^{L} \binom{\ell_{M+1}+M-1}{M-1} = \binom{L+M}{M}\ . \tag{B.1-5}$$

Thus (B.1-4) is proved by induction.

B.2 NUMBERS OF PARTITIONED ORDERED VECTORS

We define $R_M(L;n)$ as $(L+1)^{-1}$ times the number of ordered vectors $\underline{\ell}'$ such that there are exactly n partitions of the components $(\ell_2 \le \ell_3 \le \ldots \le \ell_M)$ such that the ℓ_m in the partition are equal. For example,

$$R_M(L;1) = \sum_{\ell_M=0}^{L} \sum_{\ell_{M-1}=0}^{\ell_M} \cdots \sum_{\ell_2=0}^{\ell_3} U(\ell_2 = \ell_3 = \ell_4 = \cdots = \ell_M)$$

$$= \sum_{\ell_M=0}^{L} 1 = (L+1), \tag{B.2-1}$$

where $U(\cdot)$ is 1 if the relation in the argument is true and zero otherwise. $R_M(L;1)$ is the number of ordered vectors where the components are all equal. For two partitions, there are many cases, but they all produce the sum

$$R_M(L;2) = \sum_{r_2=1}^{L} \sum_{r_1=0}^{r_2-1} 1 = \sum_{r_2=0}^{L} r_2 = \binom{L+1}{2}, \tag{B.2-2}$$

from [3, equation 0.121.1]. We find that

$$R_M(L;n) = \sum_{r_n=0}^{L} R_{M-1}(r_n-1;\ n-1). \tag{B.2-3}$$

Asserting that

$$R_M(L;n) = \binom{L+1}{n} \equiv R(L;n) \tag{B.2-4}$$

and substituting (B.2-4) in (B.2-3) establishes this relation by inductive proof.

B.3 TOTAL NUMBER OF VECTORS REPRESENTED BY ORDERED VECTORS

Since there are

$$\binom{M-1}{n_0, n_1, \ldots, n_L} \tag{B.3-1}$$

$\underline{\ell}$ vectors represented by each $\underline{\ell}'$ vector (see Section 2.2), the total number of vectors represented is (L + 1) times the summation

$$\sum_{\ell_M=0}^{L} \sum_{\ell_{M-1}=0}^{\ell_M} \cdots \sum_{\ell_2=0}^{\ell_3} \binom{M-1}{n_0, n_1, \ldots, n_L} \triangleq T_M(L). \tag{B.3-2}$$

Using the partitioning relations, we can write

$$\begin{aligned} T_M(L) &= \sum \binom{M-1}{n_0, n_1, \ldots, n_L} \times \begin{pmatrix}\text{number of partitioned } \underline{\ell}' \\ \text{which produce } n_0, n_1, \ldots n_L\end{pmatrix} \\ &= \sum_{n=1}^{M-1} R(L;n) \sum_{\text{partitions}} \binom{M-1}{q_1, q_2, \ldots, q_n} \cdot \binom{n}{r_1 r_2, \ldots, r_{M-1}}, \end{aligned} \tag{B.3-3}$$

where

$$q_k \triangleq \text{number of equal } \ell\text{'s in partition } k \tag{B.3-4a}$$

$$r_s \triangleq \text{number of } q_k \text{ equal to } s. \tag{B.3-4b}$$

Thus

$$T_2(L) = \sum_{n=1}^{1} R(L;n) \sum_{\text{part.}} \binom{1}{q_1, q_2, \ldots, q_n} \binom{n}{r_1} = L+1, \tag{B.3-5}$$

and

$$T_3(L) = \sum_{n=1}^{2} R(L;n) \sum_{\text{part.}} \binom{2}{q_1, q_2, \ldots, q_n} \binom{n}{r_1, r_2}$$

$$= \binom{L+1}{1}\binom{2}{2}\binom{1}{1} + \binom{L+1}{2}\binom{2}{1,\ 1}\binom{2}{2,\ 0}$$

$$= L + 1 + 2\binom{L+1}{2} = (L+1)^2 \qquad \text{(B.3-6)}$$

Calculations show that

$$T_4(L) = (L+1)^3$$

$$T_5(L) = (L+1)^4$$

and $$T_6(L) = (L+1)^5.$$

It can be shown [4, p. 106] that

$$T_M(L) = (L+1)^{M-1}, \qquad \text{(B.3-7)}$$

giving the total of $(L+1)^M$ $\underline{\ell}$ vectors.

B.4 NUMBERS OF ACJ-AGC JAMMING EVENTS

From Section 4, the jamming events are described by the vector $\underline{\ell}$ and the number of hops with at least one channel jammed, ℓ_0.

B.4.1 Number of $\{\ell_0, \underline{\ell}\}$ events

For a given ℓ_0 and $\underline{\ell}$, the number of events may be counted directly using

$$\#(\ell_0, \underline{\ell}) = \binom{L}{\ell_0} \sum_{\underline{\nu}_1 > \underline{0}} \cdots \sum_{\underline{\nu}_{\ell_0} > \underline{0}} \delta\left(\sum_{k=1}^{\ell_0} \underline{\nu}_k, \underline{\ell}\right). \tag{B.4-1}$$

Previously we have established that the number of $\underline{\ell}$ vectors for n hops is

$$\mathcal{S}(n) \triangleq \sum_{\underline{\nu}_1} \cdots \sum_{\underline{\nu}_n} \delta\left(\sum_{k=1}^{n} \underline{\nu}_k, \underline{\ell}\right) \prod_{m=1}^{M} \binom{n}{\ell_m} \tag{B.4-2}$$

Note that this quantity is zero for $n < \ell_x = \max_m \ell_m$. For notational convenience, let the sum in (B.4-1) be represented as

$$S(\ell_0) = \sum_{[\nu:\ell_0] > 0} \delta(\cdot,\cdot) \equiv \text{(sum over all } [\nu] \text{ with } \ell_0 \text{ non-zero columns and M rows).} \tag{B.4-3}$$

Then we find that

$$\sum_{[\nu:\ell_0]>0} = \sum_{[\nu:\ell_0]} - \sum_{[\nu:\ell_0]} \text{(at least one zero column of } [\nu:\ell_0]), \tag{B.4-4a}$$

or

$$S(\ell_0) = \mathcal{S}(\ell_0) - \text{(sums with at least one zero column).} \tag{B.4-4b}$$

Now, a sum with exactly n non-zero columns of $[\nu]$ is equivalent to $S(n)$, so that

$$S(\ell_0) = \mathcal{S}(\ell_0) - \sum_{n=1}^{\ell_0} \binom{\ell_0}{n} S(\ell_0 - n). \tag{B.4-5}$$

For example,

$$S(1) = \mathcal{S}(1) - S(0) = \mathcal{S}(1) - S(0) \tag{B.4-6a}$$

$$S(2) = \mathcal{S}(2) - 2S(1) - \mathcal{S}(0)$$

$$= \mathcal{S}(2) - 2\mathcal{S}(1) + \mathcal{S}(0) \tag{B.4-6b}$$

$$S(3) = \mathcal{S}(3) - 3S(2) - 3S(1) - S(0)$$

$$= \mathcal{S}(3) - 3\mathcal{S}(2) - 3\mathcal{S}(1) - \mathcal{S}(0). \tag{B.4-6c}$$

From these examples we conjecture that

$$S(n) = \sum_{k=0}^{n} \binom{n}{k} (-1)^k \, \mathcal{S}(n-k); \tag{B.4-7}$$

substitution of (B.4-7) into (B.4-5) for $\ell_0 = n+1$ leads to an inductive proof. Therefore, we obtain the result

$$\#(\ell_0, \underline{\ell}) = S(\ell_0)$$

$$= \binom{L}{\ell_0} \sum_{r=0}^{\ell_0} \binom{\ell_0}{r} (-1)^r \prod_{m=1}^{M} \binom{\ell_0 - r}{\ell_m} \tag{B.4-8a}$$

$$= \binom{L}{\ell_0} \sum_{r=0}^{\ell_0 - \ell_x} \binom{\ell_0}{r} (-1)^r \prod_{m=1}^{M} \binom{\ell_0 - r}{\ell_m}, \tag{B.4-8b}$$

since terms of the sum are zero for $r > \ell_0 - \ell_x$.

B.4.2 Summation over ℓ_0 events

We now demonstrate that summation of (B.4-8a) over ℓ_0 gives the total number of $\underline{\ell}$ vectors. We use the fact that

$$\binom{\ell_0 - r}{k} = \frac{1}{k!} \frac{\partial^k}{\partial x^k} (1+x)^{\ell_0 - r} \Bigg|^{x=0} . \tag{B.4-9}$$

Substituting this M times (once for each $\ell_m = k$) yields

$$\sum_{\ell_0=0}^{L} S(\ell_0) = \left\{ \left(\prod_{m=1}^{M} \frac{1}{\ell_m!} \frac{\partial^{\ell_m}}{\partial x_m^{\ell_m}} \right) \sum_{\ell_0=0}^{L} \binom{L}{\ell_0} \sum_{r=0}^{\ell_0} \binom{\ell_0}{r} (-1)^r \right.$$

$$\left. \cdot \left[\prod (1+x_m) \right]^{\ell_0 - r} \right\}^{\underline{x=0}}$$

$$= \left\{ \prod_{m=1}^{M} \frac{1}{\ell_m!} \frac{\partial^{\ell_m}}{\partial x_m^{\ell_m}} (1+x_m)^L \right\}^{\underline{x=0}} = \prod_{m=1}^{M} \binom{L}{\ell_m} . \tag{B.4-10}$$

B.4.3 Summation over $\underline{\ell}$ events

The number of ℓ_0 events can be found by summing (B.4-8a) over all possible $\underline{\ell}$ vectors. This is found to be

$$\#(\ell_0) = \binom{L}{\ell_0} \sum_{r=0}^{\ell_0} \binom{\ell_0}{r} (-1)^r \sum_{\underline{\ell}} \prod_{m=1}^{M} \binom{\ell_0 - r}{\ell_m}$$

$$= \binom{L}{\ell_0} \sum_{r=0}^{\ell_0} \binom{\ell_0}{r} (-1)^r (\ell_0 - r + 1)^M . \tag{B.4-11}$$

For example,

$$\#(\ell_0=0) = 1 \tag{B.4-12}$$

$$\#(\ell_0=1) = L\ (2^M - 1) \tag{B.4-13}$$

$$\#(\ell_0=2) = \binom{L}{2}\ (3^M - 2\cdot 2^M + 1) \tag{B.4-14}$$

$$\#(\ell_0=L) = (L+1)^M - L\cdot L^M + \binom{L}{2}\ (L-1)^M + \ldots \tag{B.4-15}$$

APPENDIX C

DERIVATION OF ERROR RATE EXPRESSIONS FOR "PRACTICAL ACJ" AND "PRACTICAL IC" RECEIVERS

C.1 JOINT PDF FOR ONE PAIR OF SAMPLES (PACJ).

The error expression will be obtained for M=2 and L=2. For a single pair of square-law envelope detector samples the joint pdf is

$$p_0(x_1,x_2) = c_1c_2e^{-c_1x_1-c_2x_2-\rho_1} I_0(2\sqrt{\rho_1c_1x_1}), \tag{C.1-1a}$$

where

$$c_i = \begin{cases} 1/2\sigma_N^2, & \text{channel unjammed;} \\ 1/2\sigma_T^2, & \text{channel jammed,} \end{cases} \tag{C.1-1b}$$

and the signal is assumed to be in channel 1. The normalized variables (z_1,z_2) resulting from this pair have the pdf

$$p_1(z_1,z_2) = p_a(z_1,z_2;x_1>x_2) + p_b(z_1,z_2;x_1<x_2). \tag{C.1-2}$$

Now, when $x_1>x_2$, z_1 is made equal to 1 and $z_2 = x_2/x_1$; thus

$$p_a(z_1,z_2;x_1>x_2) = \delta(z_1-1)\int_0^\infty d\zeta\, \zeta\, p_0(\zeta,\zeta z_2), \quad 0\le z_2\le 1. \tag{C.1-3}$$

Similarly, when $x_2<x_1$,

$$p_b(z_1,z_2;x_1<x_2) = \delta(z_2-1)\int_0^\infty d\zeta\, \zeta p_0(\zeta z_1,\zeta), \quad 0\le z_1\le 1. \tag{C.1-4}$$

The result is that z_1 and z_2, a single pair of normalized variables, have the joint pdf (conditioned on the possible jamming events) given by

$$p_1(z_1,z_2;c_1,c_2) = \frac{c_1c_2}{(c_1z_1+c_2z_2)^2} \exp\left\{-\frac{c_2z_2\rho_1}{c_1z_1+c_2z_2}\right\}\left[1+\frac{\rho_1c_1z_1}{c_1z_1+c_2z_2}\right]$$

$$\times\left[\delta(z_1-1)+\delta(z_2-1)\right], \quad 0\leq z_1,z_2\leq 1. \qquad \text{(C.1-5)}$$

By direct integration it may be shown that

$$\Pr\{z_1<z_2\} = \frac{c_1}{c_1+c_2}\exp\left\{-\frac{\rho_1c_2}{c_1+c_2}\right\} \qquad \text{(C.1-6a)}$$

and that

$$\Pr\{z_1>z_2\} = 1 - \Pr\{z_1<z_2\}; \qquad \text{(C.1-6b)}$$

thus the pdf integrates to unity as required.

Taking into account the four possible jamming events, the unconditional pdf may be written using

$$f(z_1,z_2) = \pi_0\cdot\frac{1}{(z_1+z_2)^2}\exp\left\{-\frac{\rho_Nz_2}{z_1+z_2}\right\}\left[1+\frac{\rho_Nz_1}{z_1+z_2}\right]$$

$$+\pi_1\cdot\frac{K}{(z_1+Kz_2)^2}\exp\left\{-\frac{K\rho_Tz_2}{z_1+Kz_2}\right\}\left[1+\frac{\rho_Tz_1}{z_1+Kz_2}\right]$$

$$+\pi_1\cdot\frac{K}{(Kz_1+z_2)^2}\exp\left\{-\frac{\rho_Nz_2}{Kz_1+z_2}\right\}\left[1+\frac{\rho_NKz_1}{Kz_1+z_2}\right]$$

$$+\pi_2\cdot\frac{1}{(z_1+z_2)^2}\exp\left\{-\frac{\rho_Tz_2}{z_1+z_2}\right\}\left[1+\frac{\rho_Tz_1}{z_1+z_2}\right];$$

$$0\leq z_1,\ z_2\leq 1. \qquad \text{(C.1-7)}$$

With this function the unconditioned pdf becomes

$$p_1(z_1,z_2) = f(z_1,z_2)\left[\delta(z_1-1) + \delta(z_2-1)\right]. \tag{C.1-8}$$

C.2 JOINT PDF FOR SUMS OF TWO SAMPLES (PACJ).

Using convolution, the joint pdf for L=2 is

$$p_2(z_1,z_2) = \int_{\max(0,z_1-1)}^{\min(1,z_1)} dv_1 \int_{\max(0,z_2-1)}^{\min(1,z_2)} dv_2 \quad p_1(v_1,v_2)p_1(z_1-v_1,z_2-v_2), \tag{C.2-1}$$

which reduces to

$$\begin{aligned} p_2(z_1,z_2) = {} & \delta(z_1-2) \int_{\max(0,z_2-1)}^{\min(1,z_2)} dv_2 \quad f(1,v_2)f(1,z_2-v_2) \\ & + 2f(1,z_2-1)f(z_1-1,1)u(z_2-1)u(z_1-1) \\ & + \delta(z_2-2) \int_{\max(0,z_1-1)}^{\min(1,z_1)} dv_1 \quad f(v_1,1)f(z_1-v_1,1), \\ & \qquad 0 \leq z_1,z_2 \leq 2. \end{aligned} \tag{C.2-2}$$

C.3 ERROR PROBABILITY FOR L=2 (PACJ).

The error probability is, using (C.2-2),

$$\begin{aligned} P(e) = \Pr\{z_2 > z_1\} = {} & 2\int_1^2 dz_2 \int_1^{z_2} dz_1 f(1,z_2-1)f(z_1-1,1) \\ & + \int_0^1 dz_2 \int_0^{z_2} dv_1 f(v_1,1)f(z_1-v_1,1) \\ & + \int_1^2 dz_2 \int_{z_2-1}^1 dv_1 f(v_1,1)f(z_1-v_1,1). \end{aligned} \tag{C.3-1}$$

Manipulation of the integrals yields

$$P(e) = 2\int_0^1 dx\ f(1,x) \int_0^x dy\ f(y,1) + \int_0^1 dx \int_0^x dy\ f(y,1)\ f(x-y,1) + \int_0^1 dx \int_x^1 dy\ f(y,1)\ f(x+1-y,1) \tag{C.3-2}$$

$$= 2\int_0^1 dx\ f(1,x)\ G(x) + \int_0^1 dy\ f(y,1)\ G(1-y) + \int_0^1 dy\ f(y,1)\ [G(1) - G(1-y)] \tag{C.3-3}$$

$$= 2\int_0^1 dx\ f(1,x)\ G(x) + G^2(1), \tag{C.3-4}$$

where G(x) is defined as

$$G(x) \triangleq \int_0^x du\ f(u,1) \tag{C.3-5}$$

$$= \pi_0 \cdot \frac{x}{1+x} \exp\left\{ - \frac{\rho_N}{1+x} \right\} + \pi_1 \cdot \frac{x}{K+x} \exp\left\{ - \frac{K\rho_T}{K+x} \right\} + \pi_1 \cdot \frac{Kx}{1+Kx} \exp\left\{ - \frac{\rho_N}{1+Kx} \right\} + \pi_2 \cdot \frac{x}{1+x} \exp\left\{ - \frac{\rho_T}{1+x} \right\}. \tag{C.3-6}$$

Analytically it can be shown that, for $\pi_0 = 1$ (no jamming) or $\pi_2 = 1$ (full-band jamming), the P(e) equals

$$P(e) = \frac{1}{2} e^{-\rho} (1+\rho/3), \tag{C.3-7}$$

where $\rho = \frac{1}{2} E_b/N_0$ for $\pi_0 = 1$ and $\rho = \frac{1}{2} E_b/N_T$ for $\pi_2 = 1$. This is precisely the same performance obtained by the self-normalizing receiver in Section 5, for the same conditions.

C.4 JOINT PDF FOR ONE PAIR OF SAMPLES (PIC).

The joint pdf of the usual and the look-ahead receiver square-law envelope detector samples is

$$p_0(u_1,u_2,v_1,v_2) = c_1^2 c_2^2 e^{-c_1(u_1+v_1)-c_2(u_2+v_2)} I_0(2\sqrt{\rho_1 c_1 u_1}), \tag{C.4-1}$$

where the constants are defined in (C.1-1b). The normalized variables are $z_{1k} = u_1/v_1$ and $z_{2k} = u_2/v_2$, with the joint pdf

$$p_{z1k,z2k}(\alpha,\beta) = \int_0^\infty dv_1 \int_0^\infty dv_2\; v_1 v_2 p_0(v_1\alpha, v_2\beta, v_1, v_2)$$

$$= \frac{1}{(1+\beta)^2} \cdot \frac{1}{(1+\alpha)^2} \cdot \exp\left\{-\frac{\rho_1}{1+\alpha}\right\} \left[1 + \frac{\rho_1 \alpha}{1+\alpha}\right] \tag{C.4-2a}$$

$$= p_1(\beta;0)\; p_1(\alpha;\rho_1); \quad \alpha,\beta > 0. \tag{C.4-2b}$$

That is, the normalized variables are independent. Note that the jamming conditions are present only in the SNR, ρ_1.

C.5 CDF FOR SUM OF TWO NON-SIGNAL VARIABLES (PIC).

Since the two channels are independent, we may first derive the probability that the non-signal sum z_2 is greater than the signal channel sum z_1, given a specific value of z_1, then later average over z_1 to get the error probability. Formally,

$$\Pr\left\{z_2 > z_1 \mid z_1 = \alpha\right\} = 1 - F_2(\alpha), \tag{C.5-1}$$

where $F_2(\alpha)$ is the cumulative distribution function for z_2. This function is found to be

$$F_2(\alpha) = \Pr\left\{z_2 = z_{21} + z_{22} \leq \alpha\right\}$$

$$= \int_0^\alpha \frac{dx}{(1+x)^2} \int_0^{\alpha-x} \frac{dy}{(1+y)^2}$$

$$= \int_0^\alpha \frac{dx}{(1+x)^2}\left[1 - \frac{1}{\alpha-x+1}\right]$$

$$= \frac{\alpha}{1+\alpha} - \int_0^\alpha \frac{dx}{(1+x)^2} \cdot \frac{1}{(\alpha-x+1)} . \tag{C.5-2}$$

Using a partial-fraction expansion results in

$$\Pr\{z_2 > z_1 | z_1 = \alpha\} = \frac{1}{(\alpha+2)^2}\left\{\int_0^\alpha dx\, \frac{x+3+\alpha}{(1+x)^2} + \int_0^\alpha \frac{dx}{1+\alpha-x}\right\} + \frac{1}{1+\alpha}$$

$$= \frac{2}{(\alpha+2)^2}\left[\alpha + 2 + \ln(\alpha+1)\right]. \tag{C.5-3}$$

C.6 ERROR PROBABILITY FOR L=2 (PIC).

The pdf for z_1 is the convolution

$$p_{z1}(\alpha) = p_1(\alpha;\rho_1) * p_1(\alpha;\rho_2)$$

$$= \int_0^\alpha dx\, p_1(x;\rho_1)\, p_1(\alpha-x;\rho_2) , \quad \alpha > 0. \tag{C.6-1}$$

Thus the error probability is, conditioned on ρ_1 and ρ_2,

$$P_b(e|\rho_1,\rho_2) = 2\int_0^{\infty} d\alpha \, \frac{\alpha+2+\ln(\alpha+1)}{(\alpha+2)^2} \int_0^{\alpha} dx \; p_1(x;\rho_1)p_1(\alpha-x;\rho_2)$$

$$= 2\int_0^{\infty} d\alpha \int_0^{\infty} dx \; p_1(\alpha;\rho_1)p_1(x;\rho_2) \cdot \frac{\alpha+x+2+\ln(\alpha+x+1)}{(\alpha+x+2)^2} \qquad \text{(C.6-2)}$$

Transforming the integration variables by

$$u = \frac{1}{1+\alpha} \; , \; v = \frac{1}{1+x} \qquad \text{(C.6-3)}$$

results in the expression

$$P_b(e|\rho_1,\rho_2) = 2\int_0^{1} du \int_0^{1} dv \; e^{-\rho_1 u-\rho_2 v}(1+\rho_1-\rho_1 u)(1+\rho_2-\rho_2 v)$$

$$\times \left(\frac{uv}{u+v}\right)^2 \left[\frac{u+v}{uv} + \ln\left(\frac{u+v}{uv} - 1\right)\right]. \qquad \text{(C.6-4)}$$

Averaging over the jamming events (the number of hops jammed in the signal channel) yields the total error

$$P_b(e) = (1-\gamma)^2 \, P_b(e|\rho_1=\rho_2=E_b/2N_0)$$

$$+ \; 2\gamma(1-\gamma) \, P_b(e|\rho_1=E_b/2N_0,\rho_2=E_b/2N_T)$$

$$+ \; \gamma^2 P_b(e|\rho_1=\rho_2=E_b/2N_T). \qquad \text{(C.6-5)}$$

APPENDIX D

COMPUTER PROGRAM FOR SQUARE-LAW LINEAR COMBINING RECEIVER

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the square-law linear combining receiver for FH/RMFSK.

```
PDP-11 FORTRAN-77 V4.0-1         09:41:35    16-Jul-86            Page 1
RMFSKNIH8.FTN;13          /F77/TR:BLOCKS/WR

0001            PROGRAM CRNHOP
        C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM M-ARY
        C FSK/FH WITH MULTIPLE HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
        C NOISE JAMMING BY NUMERICAL INTEGRATION FOR THE SQUARE-LAW
        C LINEAR COMBINING RECEIVER
        C
        C ANALYSIS: L. E. MILLER, R. H. FRENCH
        C PROGRAM: R. H. FRENCH
        C
        C V 5.1.0 - COMPUTATIONS ONLY
        C
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            PARAMETER (LJ=125)
0004            CHARACTER*13 FNAME, GNAME
0005            LOGICAL TRASH1, TRASH2, HIGH, PRINTR
0006            LOGICAL*1 GOOD
0007            CHARACTER*1 REPLY, YES, NO
0008            REAL*4 PRLOG(LJ),DBSJR(LJ)
0009            DIMENSION MATRIX(4,8),MLOW(4,8),MUP(4,8),MINC(4,8),PIE(0:8)
0010            VIRTUAL D(625),IDSUB(625),PRERR(625),IPSUB(625)
0011            COMMON /RESET/ TRASH1, TRASH2
        C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
0012            COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
        C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
0013            COMMON /SIZE/ NO,NL,NG
0014            COMMON /REGION/ HIGH
0015            COMMON /PARMS/ BIGK, LL
0016            DATA YES, NO/'Y','N'/, PRINTR/.FALSE./
0017            CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0018            CALL GET(NJ,START,DBINC)
0019            WORBIT=0.5D0*MM/(MM-1.D0)
0020            DO 900 IL=1,NL
0021            LL=LLIST(IL)
0022            FLL=LL
0023            DO 800 IO=1,NO
0024            EBNO=10.D0**(DEBNOL(IO)/10.D0)
0025            RHON=K*EBNO/FLL
0026            IOOUT=DEBNOL(IO)
0027            DO 700 IG=1,NG
0028            GAMMA=GAMLST(IG)
0029            NQ=GAMMA*NSLOTS+0.5D0
        C
        C OPEN DATA FILE
        C
0030            IGOUT=GAMMA*1000.D0+0.5D0
0031            WRITE(FNAME,730) MM,LL,IOOUT,IGOUT
0032      730   FORMAT('P',I1,I1,I2.2,I4.4,'.DAT')
0033            IF(PRINTR) THEN
0034              OPEN(UNIT=6,FILE='FOR006.DAT',STATUS='OLD',FORM='FORMATTED',
          $            ACCESS='APPEND')
```

```
PDP-11 FORTRAN-77 V4.0-1         09:41:35    16-Jul-86            Page 2
RMFSKNIH8.FTN;13          /F77/TR:BLOCKS/WR

0035            ELSE
0036              OPEN(UNIT=6,FILE='FOR006.DAT',STATUS='NEW',FORM='FORMATTED')
0037              PRINTR=.TRUE.
0038            END IF
0039            WRITE(6,776) MM,LL,DEBNOL(IO),GAMMA
0040            CLOSE(UNIT=6)
0041      776   FORMAT('1SQUARE-LAW LINEAR COMBINING RECEIVER'/
          $     ' M=',I2,5X,'L=',I2,5X,'EB/NO=',F8.4,5X,'GAMMA=',1PD10.3/
          $     ' EB/NJ (dB)',5X,'P(e)')
0042            WRITE(5,733) FNAME
0043      733   FORMAT(' WORKING ON FILE ',A13)
0044            OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
          $         ERR=750)
        C
        C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
        C
0045            WRITE(5,1100)
0046     1100   FORMAT(' IS THIS A FIX-UP RE-RUN? [Y/N]: ',$)
0047            READ(5,1101) REPLY
0048     1101   FORMAT(A1)
0049            IF(REPLY.EQ.NO) GOTO 1300
        C
        C FIX UP AN EXISTING FILE WHICH HAS GARBAGE AT THE HIGH END
        C
        C   ... FIRST READ EVERYTHING
0050            READ(4) MMIN, LLIN, EBNOIN, NSLIN, GAMIN, DBSJR, PRLOG
0051            IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. EBNOIN.NE.DEBNOL(IO)
          $        .OR. GAMIN.NE.GAMMA .OR. NSLIN.NE.NSLOTS)
          $         STOP 'FILE SYNC ERROR OR CORRUPTED FILE'
0052            WRITE(5,1102)
0053     1102   FORMAT(' HOW MANY POINTS ARE GOOD? ',$)
0054            READ(5,1103) JGOOD
0055     1103   FORMAT(I3)
0056            CLOSE(UNIT=4)
0057            JJ=JGOOD+1
        C CREATE A NEW FILE WITH THE GOOD PART OF THE DATA IN IT
        C IN PROGRESS FORMAT
0058            OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
0059            WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA
0060            OPEN(UNIT=6,FILE='FOR006.DAT',STATUS='OLD',FORM='FORMATTED',
          $          ACCESS='APPEND')
0061            DO 1120 IGOOD=1,JGOOD
0062            WRITE(4) DBSJR(IGOOD), PRLOG(IGOOD)
0063            WRITE(6,666) DBSJR(IJ),PE
0064     1120   CONTINUE
0065            CLOSE(UNIT=6)
0066            CLOSE(UNIT=4)
        C AND GO FINISH UP THE CALCULATIONS
0067            GOTO 755
        C
        C NOT A FIX-UP RUN, MUST BE A CONTINUATION RUN
        C
```

```
1300  READ(4) MMIN, LLIN, EBNOIN, NSLIN, GAMIN
      IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. EBNOIN.NE.DEBNOL(IO)
     $   .OR. GAMIN.NE.GAMMA .OR. NSLIN.NE.NSLOTS)
     $   STOP 'FILE SYNC ERROR OR CORRUPTED FILE'
      JJ=0
      OPEN(UNIT=6,FILE='FOR006.DAT',STATUS='OLD',FORM='FORMATTED',
     $      ACCESS='APPEND')
 740  JJ=JJ+1
      READ(4,END=742) DBSJR(JJ), PRLOG(JJ)
      WRITE(6,666) DBSJR(IJ),PE
      GOTO 740
 742  CLOSE(UNIT=4)
      CLOSE(UNIT=6)
      GOTO 755
C
C NO EXISTING FILE, THIS IS THE FIRST TIME: CREATE FILE HEADER RECORD
C
 750  JJ=1
      OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA
      CLOSE(UNIT=4)
 755  WRITE(GNAME,735) MM,LL,IGOUT
 735  FORMAT('EV',I1,I1,I4.4,'.DAT')
      OPEN(UNIT=3,FILE=GNAME,STATUS='OLD',FORM='UNFORMATTED',
     $      READONLY,ERR=770)
      WRITE(5,3939)
3939  FORMAT(' READING EXISTING EVENT FILE')
      READ(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
      GOTO 777
C IF FILE FOR EVENT PROBABILITIES DOES NOT EXIST, CALCULATE THEM
C AND CREATE A FILE.
 770  CONTINUE
      WRITE(5,3938)
3938  FORMAT(' CREATING EVENT FILE')
      CALL GENPIE(LL,MM,NQ,NSLOTS,GOOD,MATRIX,MLOW,MINC,MUP,PIE,
     $             D,IDSUB,NUSED)
      OPEN(UNIT=3,FILE=GNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
 777  IF(.NOT.GOOD) GOTO 700
      DO 600 IJ=JJ,NJ
C GIVE PROGRESS MESSAGE TO TI:
      WRITE(5,601) IJ
 601  FORMAT(' IJ=',I3)
      TRASH2=.TRUE.
      DEBNJ=START+(IJ-1)*DBINC
      IF(MM.LE.4) THEN
        HIGH=DEBNJ.GE.15.D0
      ELSE
        HIGH=DEBNJ.GT.30.D0
      END IF
```



```
      DBSJR(IJ)=DEBNJ
      R=10.D0**(DEBNJ/10.D0)
      RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
      RHOT=K*RHOTS/FLL
C EVALUATE THE PROBABILITY
      CALL PSUBE(RHON,RHOT,LL,MM,PESYM,D,IDSUB,NUSED,PRERR,IPSUB)
      PE=WORBIT*PESYM
      OPEN(UNIT=6,FILE='FOR006.DAT',STATUS='OLD',FORM='FORMATTED',
     $      ACCESS='APPEND')
      WRITE(6,666) DBSJR(IJ),PE
      CLOSE(UNIT=6)
 666  FORMAT(1X,F7.3,5X,1PD12.5)
      PRLOG(IJ)=DLOG10(PE)
      OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $      FORM='UNFORMATTED')
      WRITE(4) DBSJR(IJ), PRLOG(IJ)
      CLOSE(UNIT=4)
 600  CONTINUE
      OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA,DBSJR,PRLOG
      CLOSE(UNIT=4)
 700  CONTINUE
 800  CONTINUE
 900  CONTINUE
      STOP 'PLEASE PURGE DATA FILES'
      END
```

```
              SUBROUTINE GET(NJ,START,DBINC)
      C
      C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
      C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              CHARACTER*9 FIELD,BLANK9
              COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
              COMMON /SIZE/ NO,NL,NG
      C DEFAULT LISTS TEMPORARILY NEEDED ARE IN SHARED STORAGE WITH
      C THE LARGE CONVOLUTION WORKING ARRAYS
              COMMCN /SHARE/ DG(31),DSNR(5,4)
              DATA BLANK9/'         '/
      32      WRITE(5,33)
      33      FORMAT(' BITS/SYMBOL (K) [2]: ',$)
              READ(5,3)K
              IF(K.EQ.0)K=2
              MM=2**K
      1       WRITE(5,2)
      2       FORMAT(' HOW MANY EB/NO? [1]: ',$)
              READ(5,3)NO
      3       FORMAT(I2)
              IF(NO.EQ.0)NO=1
              DO 7 IN=1,NO
              IF(K.LE.4) THEN
                DO=DSNR(IN,K)
              ELSE
                DO=0.DO
              END IF
      4       WRITE(5,5)IN,DO
      5       FORMAT('    EB/NO(',I2,') [',F9.6,']: ',$)
              READ(5,6)FIELD
      6       FORMAT(A9)
              IF(FIELD.EQ.BLANK9) THEN
                DEBNOL(IN)=DO
              ELSE
                DECODE(9,61,FIELD)DEBNOL(IN)
      61        FORMAT(F9.6)
              END IF
      7       CONTINUE
      15      WRITE(5,16)
      16      FORMAT(' HOW MANY L? [4]: ',$)
              READ(5,3)NL
              IF(NL.EQ.0)NL=4
              DO 21 IN=1,NL
      18      WRITE(5,19)IN,IN
      19      FORMAT('    L(',I1,') [',I1,']: ',$)
              READ(5,3)LLIST(IN)
              IF(LLIST(IN).EQ.0)LLIST(IN)=IN
      21      CONTINUE
      22      WRITE(5,23)
      23      FORMAT(' HOPPING SLOTS? [2400]: ',$)
              READ(5,24)NSLOTS
```



```
      24      FORMAT(I5)
              IF(NSLOTS.EQ.0)NSLOTS=2400
      25      WRITE(5,26)
      26      FORMAT(' HOW MANY GAMMA? [10]: ',$)
              READ(5,3)NG
              IF(NG.EQ.0)NG=10
              DO 31 IN=1,NG
      28      WRITE(5,29)IN,DG(IN)
      29      FORMAT('    GAMMA(',I2,') [',1PD8.1,']: ',$)
              READ(5,30)GAMLST(IN)
      30      FORMAT(D15.8)
              IF(GAMLST(IN).EQ.0.DO)GAMLST(IN)=DG(IN)
      31      CONTINUE
      38      WRITE(5,39)
      39      FORMAT(' HOW MANY EB/NJ? [1]: ',$)
              READ(5,34,ERR=38) NJ
      34      FORMAT(I3)
              IF(NJ.EQ.0) NJ=1
              IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 32
      40      WRITE(5,41)
      41      FORMAT(' STARTING VALUE FOR EB/NJ (DB) [0.]: ',$)
              READ(5,42,ERR=40) START
      42      FORMAT(F6.3)
              IF(NJ.EQ.1) RETURN
      35      WRITE(5,36)
      36      FORMAT(' DB INCREMENT FOR EB/NJ [5.]: ',$)
              READ(5,37,ERR=35) DBINC
      37      FORMAT(F6.3)
              IF(DBINC.EQ.0.) DBINC=5.
              RETURN
              END
```

```
0001            SUBROUTINE PSUBE(RHON,RHOT,LL,M,PE,D,IDSUB,NUSED,PRERR,IPSUB)
        C
        C COMPUTE UNCONDITIONAL ERROR PROBABILITY
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            INTEGER JAM(8),LUP(8),JSUB(8)
0004            LOGICAL*1 GO,NONE,STORE
0005            LOGICAL TRASH1, TRASH2, HIGH
        C WE DO WANT TO STORE ZERO ELEMENTS OF THE DENSITY FUNCTION,
        C SINCE IT SAVES TIME TO AVOID REPEATING THE UNDERFLOWS
0006            VIRTUAL PRERR(625),IPSUB(625)
0007            VIRTUAL D(625),IDSUB(625)
0008            COMMON /REGION/ HIGH
0009            COMMON /RESET/ TRASH1, TRASH2
0010            COMMON /SHARE2/ LOW(8),LINC(8)
0011            DATA STORE/.TRUE./
0012            PE=0.D0
0013            NPS=0
0014            JAM1=-1
0015            DO 199 I1=0,LL
0016            JAM(1)=I1
0017            ITER=1
0018      100   DO 101 I=ITER+1,M
        C
        C OUTERMOST NONSIGNAL LOOP ALWAYS STARTS FROM 0, BUT
        C THE OTHERS START FROM THE CURRENT VALUE OF THE
        C NEXT OUTER MORE LOOP TO PRODUCE THE SORTED EVENTS
        C
0019            IF(I.GT.2) THEN
0020              JAM(I)=JAM(I-1)
0021            ELSE
0022              JAM(I)=0
0023            END IF
0024      101   CONTINUE
0025            IF(JAM1.NE.JAM(1)) THEN
        C UPDATE TEST VALUE FOR NEXT TIME, AND ...
0026              JAM1=JAM(1)
0027              TRASH1=.TRUE.
0028            END IF
0029            CALL EVENT(LL,M,JAM,PIE,D,IDSUB,NUSED)
        C BYPASS CONDITIONAL ERROR PROBABILITY CALCULATION IF EVENT
        C PROBABILITY IS ZERO.  THIS SAVES MUCH TIME
0030            IF(PIE.EQ.0.D0)GOTO 198
        C STORE AS ELEMENT OF A SPARSE ARRAY TO SAVE MEMORY
        C EVEN THOUGH WE STORE ZEROS, THE SORTING OF SUBSCRIPTS
        C CUTS OUT MANY ELEMENTS.
0031            CALL LOCN(M,LOW,LUP,JAM,ISUB)
        C TRY TO FIND CONDITIONAL ERROR PROBABILITY IN STORED ARRAY
0032            CALL LOOKUP(PROB,PRERR,IPSUB,NPS,625,ISUB,STORE,NONE)
        C IF IT IS NOT THERE, WE MUST COMPUTE IT
```



```
0033            IF(NONE) THEN
0034              IF(HIGH) THEN
0035                CALL PSEL1(JAM,LL,M,RHON,RHOT,PROB)
0036              ELSE
0037                CALL PSEL2(JAM,LL,M,RHON,RHOT,PROB)
0038              END IF
        C ... AND SAVE IT FOR POSSIBLE FUTURE RE-USE
0039              CALL PUTIN(PROB,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
0040              IF(KODE.NE.0) STOP 2
0041            END IF
        C SUM UP UNCONDITIONAL ERROR PROBABILITY
0042            PE=PE+PIE*PROB
0043      198   DO 197 I=2,M
0044            ITER=M+2-I
0045            JAM(ITER)=JAM(ITER)+1
0046            IF(JAM(ITER).LE.LL) GOTO 100
0047      197   CONTINUE
0048      199   CONTINUE
0049            RETURN
0050            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:09    16-Jul-86          Page 9
RMFSKNIH8.FTN;13        /F77/TR:BLOCKS/WR

              SUBROUTINE EVENT(LL,M,JAM,PIE,D,IDSUB,NUSED)
        C
        C SUBROUTINE TO LOOK UP EVENT PROBABILITY FROM STORED ARRAY
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              LOGICAL*1 STORE,NONE
              DIMENSION JAM(8),LUP(8)
              VIRTUAL D(625),IDSUB(625)
              COMMON /SHARE2/ LOW(8),LINC(8)
              DATA STORE/.FALSE./
        C SET UP ARRAY DESCRIPTION D(O:LL,...,O:LL) WITH M DIMENSIONS
              DO 1 I=1,M
              LUP(I)=LL
        1     CONTINUE
        C COMPUTE LINEAR EQUIVALENT SUBSCRIPT FOR JAMMING EVENT
              CALL LOCN(M,LOW,LUP,JAM,ISUB)
        C LOOK UP THE VALUE, GET 0.DO IF NOT THERE
              CALL LOOKUP(PIE,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:12    16-Jul-86          Page 10
RMFSKNIH8.FTN;13        /F77/TR:BLOCKS/WR

              SUBROUTINE GENPIE(LL,MM,NQ,NSLOTS,GOOD,MATRIX,MLOW,MINC,
            $                    MUP,PIE,D,IDSUB,NUSED)
        C
        C SUBROUTINE TO GENERATE EVENT PROBABILITIES
        C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              LOGICAL*1 GO,STORE,NONE,GOOD
              DIMENSION MATRIX(LL,MM),MLOW(LL,MM),MINC(LL,MM),MUP(LL,MM),
            $      PIE(O:MM),IWORK(8),LOWWRK(8),LUPWRK(8)
              VIRTUAL D(625),IDSUB(625)
        C STORE=.FALSE. => DON'T STORE ZERO ELEMENTS OF SPARSE ARRAY.
              STORE=.FALSE.
              GOOD=.TRUE.
        9999  IF(NQ.LE.O) THEN
                GOOD=.FALSE.
                RETURN
              END IF
              DO 90 I=0,MM
              CALL PRIHOP(I,MM,NQ,NSLOTS,A)
              PIE(I)=A
          90  CONTINUE
              DO 95 J=1,MM
              DO 95 I=1,LL
              MLOW(I,J)=0
              MUP(I,J)=1
              MINC(I,J)=1
          95  CONTINUE
              NUSED=0
              DO 98 I=1,MM
              LOWWRK(I)=0
              LUPWRK(I)=LL
          98  CONTINUE
        C INITIALIZE MATRIX-INDEX LOOP
              CALL MLINIT(MATRIX,MLOW,LL,MM)
        999   CONTINUE
        C FORM COLUMN SUMS AND COMPUTE P(EVENT)
              P=1.DO
              DO 101 I=1,LL
              K=0
              DO 100 J=1,MM
              K=K+MATRIX(I,J)
         100  CONTINUE
              P=P*PIE(K)
         101  CONTINUE
        C FORM JAMMING EVENT VECTOR
              DO 102 J=1,MM
              IWORK(J)=0
              DO 102 I=1,LL
              IWORK(J)=IWORK(J)+MATRIX(I,J)
         102  CONTINUE
```

```
C SORT NONSIGNAL CHANNELS
        DO 103 I=2,MM-1
        DO 103 J=I+1,MM
        IF(IWORK(J).LT.IWORK(I)) THEN
          ITEMP=IWORK(I)
          IWORK(I)=IWORK(J)
          IWORK(J)=ITEMP
        END IF
  103   CONTINUE
        CALL LOCN(MM,LOWWRK,LUPWRK,IWORK,ISUB)
        CALL LOOKUP(DOUT,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
        DOUT=DOUT+P
        CALL PUTIN(DOUT,D,IDSUB,NUSED,625,ISUB,IERR,STORE)
        IF(IERR.NE.0) STOP 'TOO MANY EVENTS'
C ITERATE MATRIX-INDEX LOOP
        CALL MLITER(MATRIX,MLOW,MUP,MINC,LL,MM,GO)
        IF(GO) GOTO 999
        RETURN
        END
```



```
        SUBROUTINE PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C
C THIS SUBROUTINE INSERTS AN ELEMENT INTO A SPARSE ARRAY FOR
C WHICH ONLY THE NON-ZERO ELEMENTS ARE KEPT IN STORAGE IF
C THE SWITCH STORE IS .TRUE.
C
C THE DOUBLE PRECISION VALUE CIN IS STORED AS C(K) WHERE
C THE AUXILIARY ARRAY ICSUB IS USED TO KEEP TRACK OF THE
C SUBSCRIPTS OF THE CORRESPONDING ENTRIES IN C.
C LONG (4-BYTE) INTEGERS ARE USED TO ACCOMODATE LARGE
C SUBSCRIPT VALUES FOR THE SPARSE ARRAY C.
C
C USAGE:
C       LOGICAL*1 STORE
C       DOUBLE PRECISION C,CIN
C       VIRTUAL ICSUB(NMAX),C(NMAX)
C       CALL PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C WHERE
C  CIN = VALUE OF ELEMENT TO STORE
C  C = ARRAY IN WHICH NON-ZERO VALUES ARE ACTUALLY STORED
C  ICSUB = AUXILIARY ARRAY FOR ACTUAL SUBSCRIPT VALUES
C  NUSE = NUMBER OF ELEMENTS OF C CURRENTLY OCCUPIED
C  NMAX = SIZE OF ARRAY C
C  IERR = ERROR RETURN CODE, 0 IF NO ERROR OR 1 IF THERE IS
C      NO ROOM AVAILABLE IN C
C  STORE = .TRUE. TO STORE ZEROES EXPLICITLY, ELSE .FALSE.
C NOTE: IF CIN=0 AND THE SUBSCRIPT K IS FOUND IN ICSUB, THEN
C       THE ELEMENT IS DELETED BY SHIFTING DOWNWARD ALL
C       FOLLOWING ELEMENTS OF THE ARRAY
C
C PROGRAMMER: ROBERT H. FRENCH         DATE: 11 JANUARY 1984
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        VIRTUAL ICSUB(NMAX),C(NMAX)
        LOGICAL*1 STORE
        IERR=0
        IF(STORE) GOTO 5
        IF(CIN.EQ.0.D0) GOTO30
    5   IF(NUSE.EQ.0)GOTO 20
        DO 10 I=1,NUSE
        IF(ICSUB(I).NE.K) GOTO 10
        C(I)=CIN
        RETURN
   10   CONTINUE
        IF(NUSE.LT.NMAX) GOTO 20
        IERR=1
        RETURN
   20   NUSE=NUSE+1
        ICSUB(NUSE)=K
        C(NUSE)=CIN
        RETURN
```

```
0021     30          DO 40 I=1,NUSE
0022                 J=1
0023                 IF(ICSUB(I).EQ.K) GOTO 50
0024     40          CONTINUE
0025                 RETURN
               C
               C REMOVE THE ZEROED ELEMENT AND BUMP COUNT OF ENTRIES USED
               C
0026     50          DO 60 I=J,NUSE-1
0027                 ICSUB(I)=ICSUB(I+1)
0028                 C(I)=C(I+1)
0029     60          CONTINUE
0030                 NUSE=NUSE-1
0031                 RETURN
0032                 END
```



```
0001                 SUBROUTINE LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
               C
               C THIS SUBROUTINE RETRIEVES AN ELEMENT OF A SPARSE ARRAY WHICH
               C HAS BEEN STORED COMPACTLY BY STORING ONLY NON-ZERO ELEMENTS.
               C
               C THE ARRAY IS DOUBLE PRECISION.
               C
               C USAGE:
               C       VIRTUAL ICSUB(NMAX), C(NMAX)
               C       LOGICAL*1 STORE, NONE
               C       DOUBLE PRECISION COUT
               C       CALL LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
               C WHERE
               C       COUT = VALUE OF C(K) (OUTPUT FROM SUBROUTINE)
               C          C = ARRAY USED TO STORE NON-ZERO ELEMENTS
               C      ICSUB = AUXILIARY ARRAY TO STORE ACTUAL SUBSCRIPTS
               C          N = NUMBER OF ELEMENTS OF C CURRENTLY IN USE
               C       NMAX = SIZE OF C
               C          K = SUBSCRIPT OF SPARSE ARRAY TO LOOK UP
               C      STORE = .TRUE. IF ZEROES STORED EXPLICITLY, ELSE .FALSE.
               C       NONE = .FALSE. IF ZEROES NOT STORED OR ZEROES STORED AND
               C                    ELEMENT IS FOUND IN THE STORED ARRAY
               C              .TRUE.  IF ZEROES ARE STORED AND THE ELEMENT IS
               C              NOT FOUND (OUTPUT QUANTITY)
               C
               C PROGRAMMER: ROBERT H. FRENCH
               C       DATE: 11 JANUARY 1984
               C
0002                 IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003                 VIRTUAL ICSUB(NMAX),C(NMAX)
0004                 LOGICAL*1 STORE, NONE
0005                 NONE=.FALSE.
0006                 DO 10 I=1,N
0007                 IF(ICSUB(I).NE.K)GOTO 10
0008                 COUT=C(I)
0009                 RETURN
0010     10          CONTINUE
0011                 IF(STORE) THEN
0012                   NONE=.TRUE.
0013                 ELSE
0014                   COUT=0.
0015                 END IF
0016                 RETURN
0017                 END
```

```
0001            SUBROUTINE LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C
        C THIS SUBROUTINE COMPUTES THE EQUIVALENT LINEAR SUBSCRIPT FOR
        C A MULTIDIMENSIONAL ARRAY OF NDIM DIMENSIONS
        C
        C IF THE ARRAY A IS DEFINED AS
        C       DIMENSION A(ILOW(1):IUP(1),....,ILOW(NDIM):IUP(NDIM))
        C AND ISUB(1),....,ISUB(NDIM) IS A SET OF SUBSCRIPTS FOR A,
        C THEN THIS SUBROUTINE RETURNS IN LINEAR THE OFFSET FROM THE
        C ORIGIN OF A TO THE ELEMENT A(ISUB(1),....,ISUB(NDIM)), ASSUMING
        C THE FIRST SUBSCRIPT VARIES MOST RAPIDLY.
        C
        C USAGE:
        C       DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
        C       DATA ILOW/lower limits of defined subscripts of array/
        C       DATA IUP/upper limits of defined subscripts of array/
        C       ...SET ISUB TO DESIRED SEQUENCE OF SUBSCRIPTS...
        C       CALL LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C WHERE
        C       NDIM = NUMBER OF DIMENSIONS THE ARRAY HAS
        C       ILOW = ARRAY OF LOWER SUBSCRIPT BOUNDS
        C        IUP = ARRAY OF UPPER SUBSCRIPT BOUNDS
        C       ISUB = ARRAY CONTAINING SUBSCRIPT FOR WHICH LOCATION IS
        C              TO BE COMPUTED
        C     LINEAR = RETURNED VALUE OF OFFSET INTO ARRAY IN MEMORY
        C
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 11 JANUARY 1984
        C
0002            DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
0003            LINEAR=0
0004            DO 10 I=1,NDIM-1
0005            J=NDIM-I+1
0006            LINEAR=(LINEAR+(ISUB(J)-ILOW(J)))*(IUP(J-1)-ILOW(J-1)+1)
0007    10      CONTINUE
0008            LINEAR=LINEAR+ISUB(1)-ILOW(1)
0009            RETURN
0010            END
```



```
0001            SUBROUTINE MLINIT(LMAT,LLOW,LMAXC,LMAXR)
        C
        C THIS SUBROUTINE INITIALIZES A "MATRIX DO-LOOP" STRUCTURE
        C DEFINED BY THE FOLLOWING PSEUDO-FORTRAN CODE:
        C       DO 100 LMAT(1,1)=LLOW(1,1),LUP(1,1),LINC(1,1)
        C                   :
        C                   :
        C       DO 100 LMAT(LMAXC,1)=LLOW(LMAXC,1),LUP(LMAXC,1),LINC(LMAXC,1)
        C       DO 100 LMAT(1,2)=LLOW(1,2),LUP(1,2),LINC(1,2)
        C                   :
        C                   :
        C       DO 100 LMAT(LMAXC,2)=LLOW(LMAXC,2),LUP(LMAXC,2),LINC(LMAXC,2)
        C       DO 100 LMAT(1,LMAXR)=LLOW(1,LMAXR),LUP(1,LMAXR),LINC(1,LMAXR)
        C                   :
        C                   :
        C       DO 100 LMAT(LMAXC,LMAXR)=LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR),
        C     $                         LINC(LMAXC,LMAXR)
        C                   :
        C       (STATEMENTS IN RANGE OF LOOP)
        C                   :
        C 100   CONTINUE
        C
        C THE COMPANION ROUTINE MLITER HANDLES THE LOOP CONTROL AT THE
        C CONTINUE STATEMENT IN THE ABOVE STRUCTURE
        C
        C USAGE:
        C       LOGICAL*1 GO
        C       DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR)
        C       DIMENSION LINC(LMAXC,LMAXR)
        C (INITIALIZE MATRIX LLOW TO STARTING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE MATRIX LUP TO STOPING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE MATRIX LINC TO INCREMENTS OF THE LOOPS)
        C       CALL MLINIT(LMAT,LLOW,LMAXC,LMAXR)
        C   100 CONTINUE
        C         .
        C         . (STATEMENTS IN RANGE OF LOOPS)
        C         .
        C       CALL MLITER(LMAT,LLOW,LUP,LINC,LMAXC,LMAXR,GO)
        C       IF(GO)GOTO 100
        C WHERE
        C    LMAT = ARRAY FOR STORAGE OF LOOP INDICES. LMAT(1,1) IS THE
        C           OUTER-MOST LOOP; LMAT(LMAXC,LMAXR), THE INNER-MOST LOOP.
        C    LLOW = ARRAY FOR STORAGE OF LOOP STARTING VALUES, IN SAME
        C           SEQUENCE AS LMAT
        C    LUP = ARRAY FOR STORAGE OF LOOP FINAL VALUES, IN SAME
        C          SEQUENCE AS LMAT
        C    LINC = ARRAY FOR STORAGE OF LOOP INCREMENTS, IN SAME
        C           SEQUENCE AS LMAT
        C    LMAX = NUMBER OF LOOPS NESTED
        C    GO = LOGICAL VARIABLE, .TRUE. IF JUMP BACK TO BEGINNING OF
        C         STATEMENTS IN THE RANGE OF THE LOOP SHOULD OCCUR,
        C         .FALSE. OTHERWISE (I.E. OUTER-MOST LOOP TERMINATED)
```

```
        C
        C PROGRAMMER: ROBERT H. FRENCH          DATE: 10 MARCH 1986
        C
0002          DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR)
0003          DO 1 N=1,LMAXR
0004          DO 1 M=1,LMAXC
0005          LMAT(M,N)=LLOW(M,N)
0006    1     CONTINUE
0007          RETURN
0008          END
```



```
0001          SUBROUTINE MLITER(LMAT,LLOW,LUP,LINC,LMAXC,LMAXR,GO)
        C
        C LOOP ITERATION LOGIC FOR A "MATRIX DO-LOOP"
        C
        C SEE DETAILED COMMENTS IN SUBROUTINE MLINIT FOR USAGE AND
        C PARAMETER DEFINITIONS
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 10 MARCH 1986
        C
0002          LOGICAL*1 GO
0003          DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR)
0004          DIMENSION LINC(LMAXC,LMAXR)
0005          GO=.TRUE.
0006          DO 100 NDX=1,LMAXR
0007          NSUB=LMAXR+1-NDX
0008          DO 100 MDX=1,LMAXC
0009          MSUB=LMAXC+1-MDX
0010          LMAT(MSUB,NSUB)=LMAT(MSUB,NSUB)+LINC(MSUB,NSUB)
0011          IF((LINC(MSUB,NSUB).GE.O.AND.LMAT(MSUB,NSUB).LE.LUP(MSUB,NSUB))
             $                               .OR.
             $ (LINC(MSUB,NSUB).LT.O.AND.LMAT(MSUB,NSUB).GE.LUP(MSUB,NSUB)))
             $ RETURN
0012          LMAT(MSUB,NSUB)=LLOW(MSUB,NSUB)
0013    100   CONTINUE
0014          GO=.FALSE.
0015          RETURN
0016          END
```

```
0001            SUBROUTINE PRIHOP(KJAM,KM,KQ,KN,AIN)
         C
         C THIS SUBROUTINE COMPUTES THE EVENT PROBABILITY FOR ALL
         C POSSIBLE JAMMING PATTERNS WITH NON-ZERO PROBABILITY FOR
         C L=1 HOP/SYMBOL FOR RMFSK/FH IN PBNJ
         C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            AIN=0.D0
         C IF THIS IS AN IMPOSSIBLE CASE, RETURN WITH RESULT = 0.0
0004            IF(KJAM.GT.MIN0(KQ,KM)) RETURN
0005            KPMAX=KJAM-1
0006            LPMAX=KM-KJAM-1
0007            JPMAX=KM-1
0008            IMAX=MAX0(KPMAX,LPMAX,JPMAX)
0009            PROD=1.D0
0010            Q=KQ
0011            DIFFNQ=KN-KQ
0012            EN=KN
0013            DO 100 LOOP=0,IMAX
0014            F=LOOP
0015            IF(LOOP.LE.KPMAX) PROD=PROD*(Q-F)
0016            IF(LOOP.LE.JPMAX) PROD=PROD/(EN-F)
0017            IF(LOOP.LE.LPMAX) PROD=PROD*(DIFFNQ-F)
0018      100   CONTINUE
0019            AIN=PROD
0020            RETURN
0021            END
```



```
0001            BLOCK DATA
         C INITIALIZE SHARED CONSTANTS
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            COMMON /SHARE/ DG(31),DSNR(5,4)
0004            COMMON /SHARE2/ LOW(8),LINC(8)
         C DEFAULT LISTS FOR INTERACTIVE PARAMETER INPUTS
         C ARE SHARED WITH LARGE WORKING STORAGE ARRAYS SINCE THEY
         C MAY BE DESTROYED ONCE THE INPUT PARAMETERS ARE SET UP
0005            DATA DG / .001D0, .002D0, .005D0,
              $           .01D0,  .02D0,  .05D0,
              $           .1D0,   .2D0,   .5D0, 1.D0, 21*0.D0/
0006            DATA DSNR /13.35247D0, 12.3133D0, 10.94443D0, 0.D0, 0.D0,
              $           10.606572D0,  9.6284D0,  8.35248D0, 0.D0, 0.D0,
              $            9.09401D0,  8.1690D0,  6.971995D0, 0.D0, 0.D0,
              $            8.07835D0,  7.1996D0,  6.069646D0, 0.D0, 0.D0/
         C FREQUENTLY NEEDED CONSTANT ARRAYS AND SCALARS
0007            DATA LOW/8*0/,LINC/8*1/
0008            END
```

```
      SUBROUTINE PSEL1(JSUB,LLL,MM,RHON,RHOT,PROB)
C
C RANDOM MFSK/FH IN PARTIAL BAND TONE JAMMING,
C GIVEN A JAMMING EVENT
C
C       JSUB - JAMMING EVENT VECTOR
C         LL - NUMBER OF HOPS/SYMBOL
C         MM - ALPHABET SIZE
C       PROB - RESULTING CONDITIONAL ERROR PROBABILITY
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL PGRAND, DGAU20
      DIMENSION WORK(75), STACK(75), SAVE(75)
      INTEGER JSUB(8)
      INTEGER NCHAN(0:6)
      COMMON /PARMS/ BIGK, LL
C        AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C       ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C        NJ = NUMBER OF JAMMED HOPS
C       NNJ = NUMBER OF NON-JAMMED HOPS
      COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
      COMMON /JAMCNT/ NCHAN
      LL=LLL
      BIGK=RHON/RHOT
      NJ=JSUB(1)
      NNJ=LL-NJ
      AJ=2.D0*NJ*RHOT
      ANJ=2.D0*NNJ*RHON
      ASHIFT=DMAX1(AJ,ANJ)
C SUBTRACT 1 FOR BENEFIT OF RAPID CHI-SQUARE DENSITY CALCULATION
      NJ=NJ-1
      NNJ=NNJ-1
C
C  COUNT NUMBER OF NON-SIGNAL CHANNELS WITH L HOPS JAMMED
C
      DO 1 I=0,LL
      NCHAN(I)=0
    1 CONTINUE
      DO 2 I=2,MM
      ISUB=JSUB(I)
      NCHAN(ISUB)=NCHAN(ISUB)+1
    2 CONTINUE
      CALL DLGS15(PGRAND,ASHIFT,TAIL)
      CALL ADQUAD(0.,ASHIFT,BODY,DGAU20,PGRAND,1.D-10,
     $            WORK,STACK,75,KODE)
      IF(KODE.NE.0) THEN
        WRITE(5,3) KODE
    3   FORMAT(' ADQUAD ERROR CODE = ',I1)
        STOP 'FATAL ERROR'
      END IF
   50 PROB=TAIL+BODY
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION PGRAND(BETA)
C
C INTEGRAND FUNCTION
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      LOGICAL TRASH1, TRASH2
      VIRTUAL PU1V(2053), PU1T(2053), IPU1T(2053),
     $        FLV(8191),  FLT(8191),  IFLT(8191)
      INTEGER NCHAN(0:6)
      COMMON /JAMCNT/ NCHAN
      COMMON /RESET/ TRASH1, TRASH2
      COMMON /PARMS/ BIGK, LL
C        AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C       ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C        NJ = NUMBER OF JAMMED HOPS
C       NNJ = NUMBER OF NON-JAMMED HOPS
      COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
      IF(TRASH1) THEN
C     TRASH THE SIGNAL CHANNEL DENSITY TABLES
        DO 1 I=1,2053
    1   PU1V(I)=0.D0
        DO 2 I=1,2053
    2   PU1T(I)=0.D0
        DO 3 I=1,2053
    3   IPU1T(I)=0
        TRASH1=.FALSE.
      END IF
      IF(TRASH2) THEN
C     TRASH THE NON-SIGNAL CHANNEL DENSITY TABLES
        DO 4 I=1,8191
    4   FLV(I)=0.D0
        DO 5 I=1,8191
    5   FLT(I)=0.D0
        DO 6 I=1,8191
    6   IFLT(I)=0
        TRASH2=.FALSE.
      END IF
      PROD=1.D0
      DO 10 I=0,LL
      IF(NCHAN(I).NE.0) THEN
        ISUB=IHASH(BETA,8191)
        JSUB=ISUB
   20   T=FLT(ISUB)
        IT=IFLT(ISUB)
        IF(T.EQ.0.D0 .AND. IT.EQ.0) THEN
C       NOT FOUND, COMPUTE IT AND ENTER INTO TABLE
          X=FL(BETA,I)
          FLV(ISUB)=X
          FLT(ISUB)=BETA
          IFLT(ISUB)=I+1
```

```
               ELSE IF(T.NE.BETA .OR. IT.NE.I+1) THEN
        C      NOT THIS ENTRY, TRY NEXT ONE
                 ISUB=ISUB+1
                 IF(ISUB.GT.8191) ISUB=ISUB-8191
                 IF(ISUB.NE.JSUB) THEN
                   GOTO 20
                 ELSE
        C        HASH TABLE OVERFLOW, MUST COMPUTE, CAN NOT STORE
                   X=FL(BETA,I)
                 END IF
               ELSE IF(T.EQ.BETA .AND. IT.EQ.I+1) THEN
        C      GOT IT!
                 X=FLV(ISUB)
               END IF
               PROD=PROD*DXI(X,NCHAN(I))
             END IF
          10 CONTINUE
             ISUB=IHASH(BETA,2053)
             JSUB=ISUB
          30 T=PUIT(ISUB)
             IT=IPUIT(ISUB)
             IF(T.EQ.0.DO .AND. IT.EQ.0) THEN
        C    NOT FOUND, COMPUTE IT AND ENTER INTO TABLE
               Y=PUI(BETA)
               PUIV(ISUB)=Y
               PUIT(ISUB)=BETA
               IPUIT(ISUB)=NJ+2
             ELSE IF(T.NE.BETA .OR. IT.NE.NJ+2) THEN
        C    NOT THIS ENTRY, TRY NEXT
               ISUB=ISUB+1
               IF(ISUB.GT.2053) ISUB=ISUB-2053
               IF(ISUB.NE.JSUB) THEN
                 GOTO 30
               ELSE
        C      HASH TABLE OVERFLOWED
                 Y=PUI(BETA)
               END IF
             ELSE IF(T.EQ.BETA .AND. IT.EQ.NJ+2) THEN
        C    GOT IT!
               Y=PUIV(ISUB)
             END IF
             PGRAND=Y*(1.DO-PROD)
             RETURN
             END
```



```
             DOUBLE PRECISION FUNCTION PUI(Y)
             IMPLICIT DOUBLE PRECISION (A-H,O-Z)
             DIMENSION WORK(75), STACK(75), SAVE(75)
             EXTERNAL DGXVI, PUIG
             COMMON /PUICOM/ YY
        C      AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
        C     ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
        C      NJ = NUMBER OF JAMMED HOPS
        C     NNJ = NUMBER OF NON-JAMMED HOPS
             COMMON /PUIPAR/ AJ, ANJ, NJ, NNJ
             COMMON /PARMS/ BIGK, LL
             IF(Y.LE.0.DO) THEN
               PUI=0.DO
               RETURN
             END IF
        C NJ HAS ALREADY HAD 1 SUBTRACTED FOR RAPID CHI-SQUARE EVALUATION...
             IF(NJ.NE.-1 .AND. NJ.NE.LL-1) THEN
        C    WE MUST CONVOLVE TWO NONCENTRAL CHI-SQUARE DENSITIES
               YY=Y
               CALL ADQUA2(0.,Y,VALUE,DGXVI,PUIG,1.D-10,WORK,STACK,
            $              SAVE,75,KODE)
               IF(KODE.NE.0) THEN
                 WRITE(5,1) KODE
           1     FORMAT(' PUI ADQUA2 ERROR: KODE=',I2)
                 STOP 'FATAL ERROR'
               END IF
               PUI=VALUE/BIGK
             ELSE
        C    WE ONLY HAVE ONE NONCENTRAL CHI-SQUARE DENSITY
        C    WITH 2*LL DEGREES OF FREEDOM
               IF(NJ.EQ.-1) THEN
        C            ALL HOPS UNJAMMED
                 CALL CHISQE(Y,LL-1,ANJ,F,KODE)
               ELSE IF(NJ.EQ.LL-1) THEN
        C            ALL HOPS JAMMED
                 CALL CHISQE(Y/BIGK,LL-1,AJ,F,KODE)
                 F=F/BIGK
               END IF
               IF(KODE.NE.0) THEN
                 WRITE(5,111) KODE
         111     FORMAT(' BESSEL FUNCTION ERROR CODE = ',I1)
                 STOP 'FATAL ERROR'
               END IF
               PUI=F
             END IF
             RETURN
             END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:49    16-Jul-86           Page 25
RMFSKNIHB.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION PU1G(X)
C
C INNER INTEGRAND FUNCTION
C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /PU1COM/ YY
C             AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C            ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C             NJ = NUMBER OF JAMMED HOPS
C            NNJ = NUMBER OF NON-JAMMED HOPS
            COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
            COMMON /PARMS/ BIGK, LL
            CALL CHISQE(X/BIGK,NJ,AJ,F1,KODE)
            IF(KODE.NE.0) THEN
              WRITE(5,1) KODE
    1         FORMAT(' BESSEL FUNCTION ERROR CODE: ',I2)
              STOP 'FATAL IN JAMMED HOP DENSITY'
            END IF
            CALL CHISQE(YY-X,NNJ,ANJ,F2,KODE)
            IF(KODE.NE.0) THEN
              WRITE(5,1) KODE
              STOP 'FATAL IN UNJAMMED HOP DENSITY'
            END IF
            PU1G=F1*F2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:51    16-Jul-86           Page 26
RMFSKNIHB.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION FL(ALPHA,L)
C
C NONSIGNAL CHANNEL CUMULATIVE DISTRIBUTION
C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /PARMS/ BIGK, LL
            IF(L.NE.0.AND.L.NE.LL) THEN
              R=(BIGK-1.)/BIGK
              BIGLOG=DLOG(BIGK)
              ARG=L*BIGLOG+ALPHA/2.
              START=DEXP(-ARG)
              FL1=L-1
              DO 100 N=0,150
              EN=N
              IF(N.EQ.0) THEN
                PART=1.D0
                TERM=DIEXP(ALPHA/2.D0,START,LL-1)
                SUM=TERM
              ELSE
                PART=PART*R*(EN+FL1)/EN
                TERM=PART*DIEXP(ALPHA/2.D0,START,LL+N-1)
                DUMMY=SUM+TERM
                SUM=DUMMY
              END IF
              IF(DABS(TERM).LE.1.D-11*DABS(SUM)) GOTO 125
  100         CONTINUE
              STOP 'FL SUM DID NOT CONVERGE'
            ELSE
              IF(L.EQ.0) THEN
                A=ALPHA/2.D0
              ELSE IF(L.EQ.LL) THEN
                A=ALPHA/(2.D0*BIGK)
              END IF
              START=DEXP(-A)
              SUM=DIEXP(A,START,LL-1)
            END IF
  125       FL=1.D0-SUM
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:55    16-Jul-86            Page 27
RMFSKNIH8.FTN;13          /F77/TR:BLOCKS/WR

0001            DOUBLE PRECISION FUNCTION DIEXP(X,START,IUP)
      C
      C INCOMPLETE EXPONENTIAL FUNCTION (DOUBLE PRECISION)
      C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            TERM=START
0004            DIEXP=TERM
0005            IF(IUP.EQ.0) RETURN
0006            DO 100 I=1,IUP
0007            F=I
0008            TERM=TERM*X/F
0009            DIEXP=DIEXP+TERM
0010      100   CONTINUE
0011            RETURN
0012            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:42:56    16-Jul-86            Page 28
RMFSKNIH8.FTN;13          /F77/TR:BLOCKS/WR

0001            SUBROUTINE CHISQE(X,N,A,DEN,KODE)
      C
      C  NON-CENTRAL CHI-SQUARE DENSITY FOR EVEN DEGREES OF FREEDOM
      C
      C         DEGREES OF FREEDOM (M) IS M=2*N+2
      C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            B=DSQRT(X*A)
0004            CALL DXBESI(B,N,BESSEL,KODE)
0005            IF(KODE.NE.0) RETURN
0006            R=X/A
0007            IF(R.NE.0.D0) THEN
0008              POWER=R**(N/2.D0)
0009            ELSE
0010              IF(N.NE.0) THEN
0011                POWER=0.D0
0012              ELSE
0013                POWER=1.D0
0014              END IF
0015            END IF
0016            DEN=0.5D0*POWER*DEXP(B-0.5D0*(X+A))*BESSEL
0017            RETURN
0018            END
```

```
      SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,WORK,STACK,SAVE,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C            WITH CALLING SEQUENCE
C            CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C        SAME ARRAY AS WORK (IN)
C  SAVE- THIRD WORK ARRAY OF SIZE N, MUST NOT BE
C        SAME ARRAY AS WORK NOR SAME AS STACK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),SAVE(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
      CALL QR(XL,XU,F,T)
      SAVE(1)=T
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      EPS=EPS/2.D0
```



```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      T=P1
      SAVE(NPTS)=P2
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=SAVE(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```

```
      FUNCTION IHASH(BETA,ISIZE)
C
C AD HOC HASHING FUNCTION
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IF(BETA.LT.1.DO) THEN
        IF(BETA.GT.0.5DO) THEN
          B=1.DO/(1.01DO-BETA)
        ELSE
          B=50.DO+1.5DO/(BETA+0.011DO)
        END IF
      ELSE
        B=100000.DO*(BETA-DINT(BETA*1000.DO)/1000.DO)+187.DO
      END IF
      B=B*23.DO
      SIZE=ISIZE
      I=DMOD(B+0.5DO,SIZE)+0.5DO
      IHASH=I+1
      RETURN
      END
```



```
      SUBROUTINE DG16(A,B,F,ANSWER)
C
C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
C
C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
C
C R. H. FRENCH, 28 FEBRUARY 1986
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(8),W(8)
      DATA X/ 0.095012509837637440185DO,
     $        0.281603550779258913230DO,
     $        0.458016777657227386342DO,
     $        0.617876244402643748447DO,
     $        0.755404408355003033895DO,
     $        0.865631202387831743880DO,
     $        0.944575023073232576078DO,
     $        0.989400934991649932596DO /
      DATA W/ 0.189450610455068496285DO,
     $        0.182603415044923588867DO,
     $        0.169156519395002538189DO,
     $        0.149595988816576732081DO,
     $        0.124628971255533872052DO,
     $        0.095158511682492784810DO,
     $        0.062253523938647892863DO,
     $        0.027152459411754094852DO /
      ANSWER=0.DO
      BMAO2=(B-A)/2.DO
      BPAO2=(B+A)/2.DO
      DO 10 I=1,8
      C=X(I)*BMAO2
      Y1=BPAO2+C
      Y2=BPAO2-C
      ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
   10 CONTINUE
      ANSWER=ANSWER*BMAO2
      RETURN
      END
```

```
0001            SUBROUTINE DGXVI(A,B,F,ANSWER)
        C
        C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
        C
        C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
        C
        C R. H. FRENCH, 28 FEBRUARY 1986
        C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            DIMENSION X(8),W(8)
0004            DATA X/ 0.0950125098376374401850D0,
           $            0.28160355077925891323D0,
           $            0.45801677765722738634200,
           $            0.61787624440264374844700,
           $            0.75540440835500303389500,
           $            0.86563120238783174388000,
           $            0.94457502307323257607800,
           $            0.98940093499164993259600 /
0005            DATA W/ 0.18945061045506849628500,
           $            0.18260341504492358886700,
           $            0.16915651939500253818900,
           $            0.14959598881657673208100,
           $            0.12462897125553387205200,
           $            0.09515851168249278481000,
           $            0.06225352393864789286300,
           $            0.02715245941175409485200 /
0006            ANSWER=0.D0
0007            BMA02=(B-A)/2.D0
0008            BPA02=(B+A)/2.D0
0009            DO 10 I=1,8
0010            C=X(I)*BMA02
0011            Y1=BPA02+C
0012            Y2=BPA02-C
0013            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0014      10    CONTINUE
0015            ANSWER=ANSWER*BMA02
0016            RETURN
0017            END
```



```
0001            SUBROUTINE DLGS15(F,A,RESULT)
        C
        C  DOUBLE PRECISION 15-POINT LAGUERRE INTEGRATION
        C  FOR THE SHIFTED SEMI-INFINITE INTERVAL:
        C
        C      INFINITY
        C         -
        C        /
        C       /    F(X) DX  =  RESULT
        C      /
        C     -
        C    A
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            DIMENSION X(15), W(15)
0004            DATA X/ 0.09330781201700,
           $            0.49269174030200,
           $            1.21559541207100,
           $            2.26994952620400,
           $            3.66762272175100,
           $            5.42533662741400,
           $            7.56591622661300,
           $           10.12022856801900,
           $           13.13028248217600,
           $           16.65440770833000,
           $           20.77647889944900,
           $           25.62389422672900,
           $           31.40751916975400,
           $           38.53068330648600,
           $           48.02608557268600 /
0005            DATA W/ 0.23957817031100,
           $            0.56010084279300,
           $            0.88700826291900,
           $            1.22366440215D0,
           $            1.57444872163D0,
           $            1.94475197653D0,
           $            2.34150205664D0,
           $            2.77404192683D0,
           $            3.25564334640D0,
           $            3.80631171423D0,
           $            4.45847775384D0,
           $            5.27001778443D0,
           $            6.35956346973D0,
           $            8.03178763212D0,
           $           11.52777210090D0 /
0006            RESULT=0.D0
0007            DO 10 I=1,15
0008            RESULT=RESULT+W(I)*F(A+X(I))
0009      10    CONTINUE
0010            RETURN
0011            END
```

```
              SUBROUTINE PSEL2(JSUB,LLL,MM,RHON1,RHOT1,PROB)
        C
        C RANDOM MFSK/FH IN PARTIAL BAND TONE JAMMING,
        C GIVEN A JAMMING EVENT
        C
        C        JSUB - JAMMING EVENT VECTOR
        C          LL - NUMBER OF HOPS/SYMBOL
        C          MM - ALPHABET SIZE
        C        PROB - RESULTING CONDITIONAL ERROR PROBABILITY
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              INTEGER RO,RL
              INTEGER JSUB(8)
              DIMENSION N(0:8),CO(0:24),CL(0:24),
             $            COEFR(10),COEFK(10),COEFP(10),
             $            INDEX(20),INDUP(20)
        C SYMBOLIC NAMES FOR INDEX-IDENTIFICATION SUBSCRIPTS
        C  (LOOP NUMBER MODULO 3)
              PARAMETER(LOOPR=1, LOOPK=2, LOOPP=0)
              COMMON /PARMS/ BIGK, LL
              COMMON /RRRCOM/ RHOT,RHON
        C
        C COMPUTE PARAMETERS
        C
              RHON=RHON1
              RHOT=RHOT1
              LL=LLL
              BIGK=RHON/RHOT
              LM1=LL-1
              FK1L1 = DXI(BIGK-1.DO,LM1)
        C
        C COMPUTE POWERS OF FL
        C
              DO 10 I=0,LL
              N(I)=0
           10 CONTINUE
              DO 12 I=2,MM
              N(JSUB(I))=N(JSUB(I))+1
           12 CONTINUE
        C
        C OVER-ALL SUMMATION INITIALIZATION
        C
              SUM=0.DO
        C
        C --- START LOOP ON RO ---
        C
              DO 9000 RO=0,N(0)
              FRO=RO
              IF(RO.EQ.0) THEN
                COEFO=1.DO
              ELSE
                COEFO=-COEFO*((N(0)-FRO+1.DO)/FRO)
              END IF
```



```
        C
        C --- START LOOP ON KO ---
        C
              KOMAX=RO*LM1
        C
        C     PRE-COMPUTE THE J.C.P. MILLER COEFFICIENTS OVER KO!
        C
              CALL JCPMF(CO,KOMAX,RO,LM1)
              DO 8000 KO=0,KOMAX
        C
        C --- START LOOP ON RL ---
        C
              DO 7000 RL=0,N(LL)
              FRL=RL
              IF(RL.EQ.0) THEN
                COEFL=1.DO
              ELSE
                COEFL=-COEFL*((N(LL)-FRL+1.DO)/FRL)
              END IF
        C
        C --- START LOOP ON KL ---
        C
              KLMAX=RL*LM1
        C
        C     PRE-COMPUTE THE J.C.P. MILLER COEFFICIENTS OVER KL!
        C
              CALL JCPMF(CL,KLMAX,RL,LM1)
              DO 6000 KL=0,KLMAX
              IF(KL.EQ.0) THEN
                POWKL=1.DO
              ELSE
                POWKL=POWKL/BIGK
              END IF
              COEFKL=CL(KL)*POWKL
        C
        C --- START THE VARIABLE-LEVEL NESTED LOOPS
        C
              LOOPS=3*LM1
              IF(LOOPS.EQ.0) GOTO 555
        C  SET UP INDEX OF OUTERMOST LOOP
              INDEX(1)=0
        C  SET UP THE NON-VARYING UPPER LIMITS
              DO 22 I=1,LM1
              INDUP(3*I-2)=N(I)
           22 CONTINUE
        C
        C  MARK OUTER-MOST LOOP AS JUST ITERATED
        C
              NSUB=1
        C
        C  PERFORM INITIALIZATION CODE FOR LOOPS FROM JUST-ITERATED LOOP
        C  INWARD
```

```
             C
             C   FIRST, RESET INDICES OF ALL LOOPS WITHIN ONE JUST ITERATED
             C
0055      1000      DO 25 I=NSUB+1,LOOPS
0056                INDEX(I)=0
0057        25      CONTINUE
             C
             C   SECOND, PERFORM FRONT-OF-LOOP CODE FOR EACH LOOP FROM ITERATED
             C          LOOP ON INWARD
             C
0058                DO 30 I=NSUB,LOOPS
0059                IRKP=MOD(I,3)
             C   MAP 1,2,3 INTO 1; 4,5,6 INTO 2; ETC. FOR NESTING LEVEL
0060                NEST=(I+2)/3
0061                IF(IRKP.EQ.LOOPR) THEN
             C      WE ARE ITERATING AN R-LOOP
0062                  IF(INDEX(I).EQ.0) THEN
             C          FIRST TIME THROUGH:
             C            (A) INITIALIZE THE COEFFICIENT
0063                    COEFR(NEST)=1.D0
0064                  ELSE
             C          NOT FIRST TIME THROUGH:
             C            UPDATE THE COEFFICIENT
0065                    COEFR(NEST)=-(COEFR(NEST)/FKILI)*
          $                         ((N(NEST)-INDEX(I)+1.D0)/INDEX(I))
0066                  END IF
             C          SET UP VARIABLE UPPER LIMITS FOR THE ASSOCIATED
             C          K LOOP
0067                  INDUP(I+1)=INDEX(I)
0068                ELSE IF(IRKP.EQ.LOOPK) THEN
             C      WE ARE ITERATING A K LOOP
0069                  IF(INDEX(I).EQ.0) THEN
             C          FIRST TIME THROUGH:
             C            INITIALIZE THE COEFFICIENT
0070                    COEFK(NEST)=1.D0
0071                  ELSE
             C          NOT FIRST TIME THROUGH:
             C            UPDATE THE COEFFICIENT
0072                    COEFK(NEST)=COEFK(NEST)*
          $                         ((INDEX(I-1)-INDEX(I)+1.D0)/INDEX(I))
0073                  END IF
             C          SET UP THE UPPER LIMIT FOR THE ASSOCIATED P LOOP
0074                  INDUP(I+1)=ICAPP(LL,NEST,INDEX(I-1),INDEX(I))
0075                ELSE IF(IRKP.EQ.LOOPP) THEN
             C      WE ARE ITERATING A P LOOP
0076                  COEFP(NEST)=DEE(LL,NEST,INDEX(I-2),INDEX(I-1),INDEX(I),BIGK)
0077                END IF
0078        30      CONTINUE
             C
             C --- DO THE INTEGRAL ---
             C
```



```
0079       555      A0=R0+RL/BIGK
0080                IB0=K0+KL
0081                IF(LOOPS.GT.0) THEN
0082                  DO 60 I=1,LM1
0083                  A0=A0+INDEX(3*I-1)+(INDEX(3*I-2)-INDEX(3*I-1))/BIGK
0084                  IB0=IB0+INDEX(3*I)
0085        60        CONTINUE
0086                END IF
0087                PART=GRAL(A0,IB0,JSUB(1))
             C
             C --- ADD A TERM TO THE OVERALL SUM ---
             C
0088                PROD=COEF0*C0(K0)*COEFL*COEFKL
0089                IF(LOOPS.GT.0) THEN
0090                  DO 70 I=1,LM1
0091                  PROD=PROD*COEFR(I)*COEFK(I)*COEFP(I)
0092        70        CONTINUE
0093                END IF
0094                PROD=PROD*PART
0095                SUM=SUM+PROD
             C
             C --- ITERATE THE LOOPS ---
             C
0096                IF(LOOPS.GT.0) THEN
0097                  DO 80 NSUB=LOOPS,1,-1
0098                  INDEX(NSUB)=INDEX(NSUB)+1
0099                  IF(INDEX(NSUB).LE.INDUP(NSUB)) THEN
0100                    GOTO 1000
0101                  END IF
0102        80        CONTINUE
             C
             C --- OUTERMOST VARIABLE-NEST LOOP REACHED ITS UPPER LIMIT, ---
             C --- DONE WITH THE VARIABLE-NEST LOOPS                     ---
             C
0103                END IF
             C
             C --- ITERATE THE NORMAL DO LOOPS ---
             C
0104      6000      CONTINUE
0105      7000      CONTINUE
0106      8000      CONTINUE
0107      9000      CONTINUE
0108                PROB=1.D0-SUM
0109                RETURN
0110                END
```

```
      DOUBLE PRECISION FUNCTION DEE(LL,NEST,IR,K,IP,BIGK)
C
C  THE FUNCTION D(P) FOR 'BNJ/RMFSK
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION G(0:8)
      IF(LL.EQ.2) THEN
        IF(NEST.EQ.1) THEN
          DEE=DXI(BIGK,IR-K)
          IF(MOD(K,2).EQ.1) DEE=-DEE
        END IF
      ELSE IF(LL.EQ.3) THEN
        IF(NEST.EQ.1) THEN
          DEE=DXI(BIGK*BIGK,IR-K)*DBINCO(K,IP)*
     $          DXI(2.D0*BIGK-1.D0,K-IP)*DXI(BIGK-1.D0,IP)
          IF(MOD(K,2).EQ.1) DEE=-DEE
        ELSE IF(NEST.EQ.2) THEN
          DEE=DBINCO(IR-K,IP)*DXI(BIGK*(BIGK-2.D0),IR-K-IP)*
     $          DXI(BIGK-1.D0,IP)
        END IF
      ELSE IF(LL.EQ.4) THEN
        IF(NEST.EQ.1) THEN
          DENOM=3.D0*BIGK*BIGK-3.D0*BIGK+1.D0
          COEF1=(2.D0*BIGK*BIGK-3.D0*BIGK+1.D0)/DENOM
          G(0)=1.D0
          G(1)=K*COEF1
          BK1=BIGK-1.D0
          COEF2=0.5D0*BK1*BK1/DENOM
          DO 41 N=2,IP
          G(N)=((K+1-N)*COEF1*G(N-1) + (2*K+2-N)*COEF2*G(N-2))/N
   41     CONTINUE
          DEE=DXI(BIGK,3*(IR-K))*DXI(DENOM,K)*G(IP)
          IF(MOD(K,2).EQ.1) DEE=-DEE
        ELSE IF(NEST.EQ 2) THEN
          IUP=MINO(IP,K)
          ILOW=MAXO(0,IP-IR+K)
          BK3=BIGK-3.D0
          TK1=3.D0*BIGK-1.D0
          BASE=BK3*BIGK/TK1
          SUMQ=0.D0
          DO 42 IQ=ILOW,IUP
          SUMQ=SUMQ+DBINCO(IR-K,IP-IQ)*DBINCO(K,IQ)
     $             *DXI(BK3,IR-K-IP)*DXI(TK1,K)*DXI(BASE,IQ)
   42     CONTINUE
          DEE=SUMQ*DXI(BIGK,IR+IR-K-K-IP)*DXI(BIGK-1.D0,IP)
        ELSE IF(NEST.EQ.3) THEN
          G(0)=1.D0
          DENOM=DXI(BIGK,3)-3.D0*BIGK*BIGK+3.D0*BIGK
          COEF1=(BIGK*BIGK-3.D0*BIGK+2.D0)/DENOM
          G(1)=(IR-K)*COEF1
          BK1=BIGK-1.D0
          COEF2=0.5D0*BK1*BK1/(BIGK*DENOM)
```



```
          DO 43 N=2,IP
          G(N)=((IR-K+1-N)*COEF1*G(N-1)+
     $          (2*(IR-K+1)-N)*COEF2*G(N-2))/N
   43     CONTINUE
          DEE=DXI(DENOM,IR-K)*G(IP)
          IF(MOD(K,2).EQ.1) DEE=-DEE
        END IF
      END IF
      RETURN
      END
```

```
              FUNCTION ICAPP(LL,NEST,IR,K)
      C
      C COMPUTE UPPER SUMMATION LIMIT PL
      C
              IF(LL.LE.2) THEN
                ICAPP=0
              ELSE IF(LL.EQ.3) THEN
                IF(NEST.EQ.1) THEN
                  ICAPP=K
                ELSE IF(NEST.EQ.2) THEN
                  ICAPP=IR-K
                END IF
              ELSE IF(LL.EQ.4) THEN
                IF(NEST.EQ.1) THEN
                  ICAPP=2*K
                ELSE IF(NEST.EQ.2) THEN
                  ICAPP=IR
                ELSE IF(NEST.EQ.3) THEN
                  ICAPP=2*(IR-K)
                END IF
              END IF
              RETURN
              END
```



```
              DOUBLE PRECISION FUNCTION GRAL(A0,IB0,L1)
      C
      C  THE INNER INTEGRAL FOR CONDITIONAL ERROR PROBABILITY
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /RRRCOM/ RHOT,RHON
              COMMON /PARMS/ BIGK, LL
              A1=1.D0+A0
              IF(L1.GT.0 .AND. L1.LT.LL) THEN
                BK1=BIGK-1.D0
                BASE=BK1/(1.D0+BIGK*A0)
                ARG1=(L1-LL)*RHON/A1
                AR2=-L1*RHOT*BIGK/(1.D0+BIGK*A0)
                ARG2=AR2*A1/BK1
                POWER=1.D0
                SUM=0.D0
                DO 100 IQ=0,IB0
                IF(IQ.GT.0) POWER=POWER*BASE
                PM=DSQRT(POWER)
                SUM=GNLGPM(IQ+LL-1,IB0-IQ,ARG1,PM)*GNLGPM(L1-1,IQ,ARG2,PM)+SUM
       100      CONTINUE
                F=1.D0
                DO 110 I=1,IB0
                F=F*(I/A1)
       110      CONTINUE
                F=(F/DXI(A1,LL-L1))/DXI(1.D0+BIGK*A0,L1)
                GRAL=SUM*F*DEXP((ARG1+AR2)*A0)
              ELSE IF(L1.EQ.0) THEN
                ARG1=-LL*RHON/A1
                START=DEXP(ARG1*A0-LL*DLOG(A1))
                DO 200 I=1,IB0
                START=START*(I/A1)
       200      CONTINUE
                GRAL=GNLGPM(LL-1,IB0,ARG1,START)
              ELSE IF(L1.EQ.LL) THEN
                A2=1.D0+BIGK*A0
                ARG1=-LL*RHOT/A2
                BASE=BIGK/A2
                START=DEXP(ARG1*BIGK*A0-LL*DLOG(A2))
                DO 300 I=1,IB0
                START=START*(I*BASE)
       300      CONTINUE
                GRAL=GNLGPM(LL-1,IB0,ARG1,START)
              END IF
              RETURN
              END
```

```
      DOUBLE PRECISION FUNCTION GNLGPM(IALFA,N,X,PREMUL)
C
C GENERALIZED LAGUERRE POLYNOMIALS WITH PREMULTIPLICATION BY
C A WEIGHTING FACTOR (TO DEFER THE INEVITABLE OVERFLOWS)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      TERM=DBINCO(N+IALFA,N)*PREMUL
      SUM=TERM
      IF(N.EQ.0) GOTO 200
      DO 100 M=1,N
      TERM=-TERM*((X/M)*((N-M+1.DO)/(IALFA+M)))
      SUM=SUM+TERM
100   CONTINUE
200   GNLGPM=SUM
      RETURN
      END
```



```
      SUBROUTINE JCPMF(C,KMAX,R,LM1)
C
C COMPUTE J.C.P. MILLER COEFFICIENTS DIVIDED BY K!
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER R
      DIMENSION C(0:KMAX)
      C(0)=1.DO
      IF(LM1.EQ.0) RETURN
      DO 100 K=1,KMAX
      SUM=0.DO
      NUP=MIN0(K,LM1)
      BC=1.DO
      DO 90 N=1,NUP
      BC=BC*((K-N+1.DO)/N)
      SUM=SUM+BC*((R+1.DO)*N-K)*C(K-N)
90    CONTINUE
      C(K)=SUM/K
100   CONTINUE
      FAC=1.DO
      DO 110 K=1,KMAX
      FAC=FAC*K
      C(K)=C(K)/FAC
110   CONTINUE
      RETURN
      END
```

APPENDIX E

COMPUTER PROGRAM FOR INDIVIDUAL CHANNEL ADAPTIVE GAIN CONTROL RECEIVER

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the individual channel adaptive gain control receiver for FH/RMFSK.

```
0001            PROGRAM AGCIC
        C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM M-ARY
        C FSK/FH WITH MULTIPLE HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
        C NOISE JAMMING WITH INDIVICUAL CHANNEL AGC RECEIVER
        C BY NUMERICAL INTEGRATION
        C
        C ANALYSIS: L. E. MILLER, R. H. FRENCH
        C PROGRAM: R. H. FRENCH
        C
        C V 1.0.0 - COMPUTATIONS ONLY
        C
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            PARAMETER (LJ=126)
0004            CHARACTER*13 FNAME, GNAME
0005            CHARACTER*1 REPLY, YES, NO
0006            REAL*4 PRLOG(LJ),DBSJR(LJ)
        C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
0007            COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
        C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
0008            COMMON /SIZE/ NO,NL,NG
0009            DATA YES, NO/'Y','N'/
0010            CALL GET(NJ,START,DBINC)
0011            CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0012            WORBIT=0.5D0*MM/(MM-1.D0)
0013            DO 900 IL=1,NL
0014            LL=LLIST(IL)
0015            FLL=LL
0016            DO 800 IO=1,NO
0017            EBNO=10.D0**(DEBNOL(IO)/10.D0)
0018            RHON=K*EBNO/FLL
0019            IOOUT=DEBNOL(IO)
0020            DO 700 IG=1,NG
0021            GAMMA=GAMLST(IG)
0022            NQ=GAMMA*NSLOTS+0.5D0
        C
        C OPEN DATA FILE
        C
0023            IGOUT=GAMMA*1000.D0+0.5D0
0024            WRITE(FNAME,730) MM,LL,IOOUT,IGOUT
0025      730   FORMAT('I',I1,I1,I2.2,I4.4,'.DAT')
0026            WRITE(6,776) MM,LL,DEBNOL(IO),GAMMA
0027      776   FORMAT(/' INDEPENDENT CHANNEL AGC RECEIVER'//
         $      ' M=',I2,5X,'L=',I2,5X,'EB/NO=',F8.4,5X,'GAMMA=',1PD10.3/
         $      ' SNR(DB)',10X,'P(E)')
0028            WRITE(5,733) FNAME
0029      733   FORMAT(' WORKING ON FILE ',A13)
0030            OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
         $           ERR=750)
        C
        C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
        C
```



```
0031            WRITE(5,1100)
0032      1100  FORMAT(' IS THIS A FIX-UP RE-RUN? [Y/N]: ',$)
0033            READ(5,1101) REPLY
0034      1101  FORMAT(A1)
0035            IF(REPLY.EQ.NO) GOTO 1300
        C
        C WE ARE TO FIX UP AN EXISTING FILE WHICH HAS GARBAGE AT THE HIGH END
        C
        C   ... FIRST READ EVERYTHING
0036            READ(4) MMIN, LLIN, EBNOIN, NSLIN, GAMIN, DBSJR, PRLOG
0037            IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. EBNOIN.NE.DEBNOL(IO)
         $          .OR. GAMIN.NE.GAMMA .OR. NSLIN.NE.NSLOTS)
         $          STOP 'FILE SYNC ERROR OR CORRUPTED FILE'
0038            WRITE(5,1102)
0039      1102  FORMAT(' HOW MANY POINTS ARE GOOD? ',$)
0040            READ(5,1103) JGOOD
0041      1103  FORMAT(I3)
0042            CLOSE(UNIT=4)
0043            JJ=JGOOD+1
        C CREATE A NEW FILE WITH THE GOOD PART OF THE DATA IN IT
        C IN PROGRESS FORMAT
0044            OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
0045            WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA
0046            DO 1120 IGOOD=1,JGOOD
0047            WRITE(4) DBSJR(IGOOD), PRLOG(IGOOD)
0048      1120  CONTINUE
0049            CLOSE(UNIT=4)
        C AND GO FINISH UP THE CALCULATIONS
0050            GOTO 755
        C
        C NOT A FIX-UP RUN, MUST BE A CONTINUATION RUN
        C
0051      1300  READ(4) MMIN, LLIN, EBNOIN, NSLIN, GAMIN
0052            IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. EBNOIN.NE.DEBNOL(IO)
         $          .OR. GAMIN.NE.GAMMA .OR. NSLIN.NE.NSLOTS)
         $          STOP 'FILE SYNC ERROR OR CORRUPTED FILE'
0053            JJ=0
0054      740   JJ=JJ+1
0055            READ(4,END=742) DBSJR(JJ), PRLOG(JJ)
0056            GOTO 740
0057      742   CLOSE(UNIT=4)
0058            GOTO 755
        C
        C NO EXISTING FILE, THIS IS THE FIRST TIME: CREATE FILE HEADER RECORD
        C
0059      750   JJ=1
0060            OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
0061            WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA
0062            CLOSE(UNIT=4)
0063      755   DO 600 IJ=JJ,NJ
        C GIVE PROGRESS MESSAGE TO TI:
0064            WRITE(5,601) IJ
0065      601   FORMAT(' IJ=',I3)
```

```
          TRASH2=.TRUE.
          DEBNJ=START+(IJ-1)*DBINC
          HIGH=DEBNJ.GE.15.D0
          DBSJR(IJ)=DEBNJ
          R=10.D0**(DEBNJ/10.D0)
          RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
          RHOT=K*RHOTS/FLL
C EVALUATE THE PROBABILITY
          CALL PSUBE(RHON,RHOT,LL,MM,GAMMA,PESYM)
          PE=WORBIT*PESYM
          WRITE(6,666) DBSJR(IJ),PE
  666     FORMAT(1X,F7.3,5X,1PD12.5)
          PRLOG(IJ)=DLOG10(PE)
          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $        FORM='UNFORMATTED')
          WRITE(4) DBSJR(IJ), PRLOG(IJ)
          CLOSE(UNIT=4)
  600     CONTINUE
          OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
          WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA,DBSJR,PRLOG
          CLOSE(UNIT=4)
  700     CONTINUE
  800     CONTINUE
  900     CONTINUE
          STOP 'PLEASE PURGE DATA FILES'
          END
```



```
          SUBROUTINE GET(NJ,START,DBINC)
C
C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
C
          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
          CHARACTER*9 FIELD,BLANK9
          COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
          COMMON /SIZE/ NO,NL,NG
          DIMENSION DG(31), DSNR(5,4)
          DATA DG / .001D0, .002D0, .005D0,
     $              .01D0,  .02D0,  .05D0,
     $              .1D0,   .2D0,   .5D0, 1.D0, 21*0.D0/
          DATA DSNR /13.35247D0, 12.3133D0, 10.94443D0, 0.D0, 0.D0,
     $               10.60657200,  9.6284D0,  8.35248D0, 0.D0, 0.D0,
     $                9.09401D0,  8.169000,  6.9719950D0, 0.D0, 0.D0,
     $                8.07835D0,  7.1996D0,  6.0696460D0, 0.D0, 0.D0/
          DATA BLANK9/'         '/
   32     WRITE(5,33)
   33     FORMAT(' BITS/SYMBOL (K) [2]: ',$)
          READ(5,3)K
          IF(K.EQ.0)K=2
          MM=2**K
    1     WRITE(5,2)
    2     FORMAT(' HOW MANY EB/NO? [1]: ',$)
          READ(5,3)NO
    3     FORMAT(I2)
          IF(NO.EQ.0)NO=1
          DO 7 IN=1,NO
          IF(K.LE.4) THEN
            DO=DSNR(IN,K)
          ELSE
            DO=0.D0
          END IF
    4     WRITE(5,5)IN,DO
    5     FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
          READ(5,6)FIELD
    6     FORMAT(A9)
          IF(FIELD.EQ.BLANK9) THEN
            DEBNOL(IN)=DO
          ELSE
            DECODE(9,61,FIELD)DEBNOL(IN)
   61       FORMAT(F9.6)
          END IF
    7     CONTINUE
   15     WRITE(5,16)
   16     FORMAT(' HOW MANY L? [4]: ',$)
          READ(5,3)NL
          IF(NL.EQ.0)NL=4
          DO 21 IN=1,NL
   18     WRITE(5,19)IN,IN
   19     FORMAT('   L(',I1,') [',I1,']: ',$)
          READ(5,3)LLIST(IN)
```

```
        IF(LLIST(IN).EQ.0)LLIST(IN)=IN
 21     CONTINUE
 22     WRITE(5,23)
 23     FORMAT(' HOPPING SLOTS? [2400]: ',$)
        READ(5,24)NSLOTS
 24     FORMAT(I5)
        IF(NSLOTS.EQ.0)NSLOTS=2400
 25     WRITE(5,26)
 26     FORMAT(' HOW MANY GAMMA? [10]: ',$)
        READ(5,3)NG
        IF(NG.EQ.0)NG=10
        DO 31 IN=1,NG
 28     WRITE(5,29)IN,DG(IN)
 29     FORMAT('    GAMMA(',I2,') [',1PD8.1,']: ',$)
        READ(5,30)GAMLST(IN)
 30     FORMAT(D15.8)
        IF(GAMLST(IN).EQ.0.D0)GAMLST(IN)=DG(IN)
 31     CONTINUE
 38     WRITE(5,39)
 39     FORMAT(' HOW MANY EB/NJ? [1]: ',$)
        READ(5,34,ERR=38) NJ
 34     FORMAT(I3)
        IF(NJ.EQ.0) NJ=1
        IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 32
 40     WRITE(5,41)
 41     FORMAT(' STARTING VALUE FOR EB/NJ (DB) [0.]: ',$)
        READ(5,42,ERR=40) START
 42     FORMAT(F6.3)
        IF(NJ.EQ.1) RETURN
 35     WRITE(5,36)
 36     FORMAT(' DB INCREMENT FOR EB/NJ [5.]: ',$)
        READ(5,37,ERR=35) DBINC
 37     FORMAT(F6.3)
        IF(DBINC.EQ.0.) DBINC=5.
        RETURN
        END
```



```
        SUBROUTINE PSUBE(RHON,RHGT,LL,M,GAMMA,PE)
C
C COMPUTE UNCONDITIONAL ERROR PROBABILITY
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        PE=0.D0
        OMG=1.D0-GAMMA
        DO 100 L1=0,LL
        ALAM=2.D0*(LL-L1)*RHON+2.D0*L1*RHOT
        IF(OMG.NE.0.D0 .OR. (OMG.EQ.0.D0 .AND. L1.EQ.LL)) THEN
          CALL PSL1(ALAM,M,LL,PECON)
          PE=PE+DBINCO(LL,L1)*DXI(GAMMA,L1)*DXI(OMG,LL-L1)*PECON
        END IF
 100    CONTINUE
        RETURN
        END
```

```
0001            SUBROUTINE PSL1(ALAM,MM,LL,PE)
          C
          C COMPUTE CONDITIONAL ERROR PROBABILITY
          C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            DIMENSION C(0:50)
0004            PE=0.D0
0005            POWER1=-1.D0
0006            DO 900 K=1,MM-1
0007            FK=K
0008            POWER1=-POWER1*(MM-FK)/FK
0009            COEFK=POWER1/DXI(FK+1.D0,LL)
0010            IRU=K*(LL-1)
0011            IF(IRU.GT.50) STOP 'C ARRAY TOO SMALL'
0012            FK1=K+1
0013            AX=-FK*ALAM*0.5D0/FK1
0014            ARG=-ALAM*0.5D0/FK1
0015            SUMR=0.D0
0016            DO 800 IR=0,IRU
0017            R=IR
          C
          C BUILD UP THE JCP MILLER COEFFICIENT
          C
0018            IF(IR.EQ.0) THEN
0019              C(0)=1.D0
0020            ELSE
0021              NUP=MIN0(IR,MAX0(1,LL-1))
0022              SEE=0.D0
0023              DO 100 N=1,NUP
0024              SEE=SEE+DBINCO(IR,N)*((K+1)*N-IR)*C(IR-N)
0025      100     CONTINUE
0026              C(IR)=SEE/R
0027            END IF
0028            GL=GENLAG(LL-1,IR,ARG)
0029            Y=DEXP(AX+DLOG(GL))
0030            TERM=C(IR)*Y/DXI(FK1,IR)
0031            SUMR=SUMR+TERM
0032      800   CONTINUE
0033            PE=PE+COEFK*SUMR
0034      900   CONTINUE
0035            RETURN
0036            END
```



```
0001            DOUBLE PRECISION FUNCTION GENLAG(IALFA,N,X)
          C
          C GENERALIZED LAGUERRE POLYNOMIALS
          C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            TERM=DBINCO(N+IALFA,N)
0004            SUM=TERM
0005            IF(N.EQ.0) GOTO 200
0006            DO 100 M=1,N
0007            TERM=-TERM*((X/M)*((N-M+1.D0)/(IALFA+M)))
0008            SUM=SUM+TERM
0009      100   CONTINUE
0010      200   GENLAG=SUM
0011            RETURN
0012            END
```

APPENDIX F

COMPUTER PROGRAM FOR NUMERICAL COMPUTATIONS FOR THE ANY-CHANNEL-JAMMED ADAPTIVE GAIN CONTROL RECEIVER

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the any-channel-jammed adaptive gain control receiver for FH/RMFSK.

```
      PROGRAM CRNHOP
C
C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM M-ARY
C FSK/FH WITH MULTIPLE HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
C NOISE JAMMING USING THE ANY-CHANNEL-JAMMED AGC RECEIVER
C
C ANALYSIS: L. E. MILLER, R. H. FRENCH
C PROGRAM: R. H. FRENCH
C
C V 5.1.0 - COMPUTATIONS ONLY
C
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (LJ=126)
      CHARACTER*13 FNAME, GNAME
      LOGICAL TRASH1, TRASH2
      LOGICAL*1 GOOD
      DIMENSION MATRIX(4,8),MLOW(4,8),MUP(4,8),MINC(4,8),
     $          PIE(0:8)
      REAL*4 PRLOG(LJ),DBSJR(LJ)
      VIRTUAL A(100),IASUB(100),C(625),ICSUB(625)
      VIRTUAL D(625),IDSUB(625),PRERR(625),IPSUB(625)
C SAVED DENSITY VALIDITY FLAGS
      COMMON /RESET/ TRASH1, TRASH2
C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
      COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
      COMMON /SIZE/ NO,NL,NG
      COMMON /PARMS/ BIGK, LL
C DISABLE ERROR MESSAGES FOR 'FILE NOT FOUND' CONDITION
      CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
      CALL GET(NJ,START,DBINC)
      BITS=K
      WORBIT=0.5D0*MM/(MM-1.D0)
      DO 900 IL=1,NL
      LL=LLIST(IL)
      FLL=LL
      DO 800 IO=1,NO
      EBNO=10.D0**(DEBNOL(IO)/10.D0)
      RHON=BITS*EBNO/FLL
      IOOUT=DEBNOL(IO)
      DO 700 IG=1,NG
      GAMMA=GAMLST(IG)
      NQ=GAMMA*NSLOTS+0.5D0
C
C OPEN DATA FILE
C
      IGOUT=GAMMA*1000.D0+0.5D0
      WRITE(FNAME,730) MM,LL,IOOUT,IGOUT
  730 FORMAT('A',I1,I1,I2.2,I4.4,'.DAT')
      WRITE(6,776) MM,LL,DEBNOL(IO),GAMMA
  776 FORMAT(' M=',I2,5X,'L=',I2,5X,'EB/NO=',F8.4,5X,
     $          'GAMMA=',1PD10.3)
```



```
      WRITE(5 733) FNAME
  733 FORMAT(' WORKING ON FILE ',A13)
      OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
     $     ERR=750)
C
C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
C
C
 1300 READ(4) MMIN, LLIN, EBNOIN, NSLIN, GAMIN
      IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. EBNOIN.NE.DEBNOL(IO)
     $   .OR. GAMIN.NE.GAMMA .OR. NSLIN.NE.NSLOTS)
     $   STOP 'FILE SYNC ERROR OR CORRUPTED FILE'
      JJ=0
  740 JJ=JJ+1
      READ(4,END=742) DBSJR(JJ), PRLOG(JJ)
      GOTO 740
  742 CLOSE(UNIT=4)
      GOTO 755
C
C NO EXISTING FILE, THIS IS THE FIRST TIME:
C     CREATE FILE HEADER RECORD
C
  750 JJ=1
      OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA
      CLOSE(UNIT=4)
  755 WRITE(GNAME,735) MM,LL,IGOUT
  735 FORMAT('JV',I1,I1,I4.4,'.DAT')
      OPEN(UNIT=3,FILE=GNAME,STATUS='OLD',FORM='UNFORMATTED',
     $     READONLY,ERR=770)
      WRITE(5,3939)
 3939 FORMAT(' READING EXISTING EVENT FILE')
      READ(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
      GOTO 777
C
C IF FILE FOR EVENT PROBABILITIES DOES NOT EXIST,
C CALCULATE THEM AND CREATE A FILE.
C
  770 CONTINUE
      WRITE(5,3938)
 3938 FORMAT(' CREATING EVENT FILE')
      CALL GENPIE(LL,MM,NQ,NSLOTS,GOOD,MATRIX,MLOW,MINC,MUP,PIE,
     $          D,IDSUB,NUSED)
      OPEN(UNIT=3,FILE=GNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
  777 IF(.NOT.GOOD) GOTO 700
      DO 600 IJ=JJ,NJ
C GIVE PROGRESS MESSAGE TO TI:
      WRITE(5,601) IJ
  601 FORMAT(' IJ=',I3)
```

```
              TRASH2=.TRUE.
              DEBNJ=START+(IJ-1)*DBINC
              DBSJR(IJ)=DEBNJ
              R=10.D0**(DEBNJ/10.D0)
              RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
              RHOT=BITS*RHOTS/FLL
C EVALUATE THE PROBABILITY
              CALL PSUBE(RHON,RHOT,LL,MM,BITS,PESYM,D,IDSUB,NUSED,PRERR,
     $              IPSUB,DEBNJ)
              PE=WORBIT*PESYM
              WRITE(6,666) DBSJR(IJ),PE
 666          FORMAT(1X,F7.3,5X,1PD12.5)
              PRLOG(IJ)=DLOG10(PE)
              OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $           FORM='UNFORMATTED')
              WRITE(4) DBSJR(IJ), PRLOG(IJ)
              CLOSE(UNIT=4)
 600          CONTINUE
              OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
              WRITE(4) MM,LL,DEBNOL(IO),NSLOTS,GAMMA,DBSJR,PRLOG
              CLOSE(UNIT=4)
 700          CONTINUE
 800          CONTINUE
 900          CONTINUE
              STOP 'PLEASE PURGE DATA FILES'
              END
```



```
              SUBROUTINE GET(NJ,START,DBINC)
C
C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              CHARACTER*9 FIELD,BLANK9
              COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,GAMLST(31),K,MM
              COMMON /SIZE/ NO,NL,NG
C DEFAULT LISTS TEMPORARILY NEEDED ARE IN SHARED STORAGE WITH
C THE LARGE CONVOLUTION WORKING ARRAYS
              COMMON /SHARE/ DG(31),DSNR(5,4)
              DATA BLANK9/'         '/
 32           WRITE(5,33)
 33           FORMAT(' BITS/SYMBOL (K) [2]: ',$)
              READ(5,3)K
              IF(K.EQ.0)K=2
              MM=2**K
 1            WRITE(5,2)
 2            FORMAT(' HOW MANY EB/NO? [1]: ',$)
              READ(5,3)NO
 3            FORMAT(I2)
              IF(NO.EQ.0)NO=1
              DO 7 IN=1,NO
              IF(K.LE.4) THEN
                DO=DSNR(IN,K)
              ELSE
                DO=0.D0
              END IF
 4            WRITE(5,5)IN,DO
 5            FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
              READ(5,6)FIELD
 6            FORMAT(A9)
              IF(FIELD.EQ.BLANK9) THEN
                DEBNOL(IN)=DO
              ELSE
                DECODE(9,61,FIELD)DEBNOL(IN)
 61             FORMAT(F9.6)
              END IF
 7            CONTINUE
 15           WRITE(5,16)
 16           FORMAT(' HOW MANY L? [1]: ',$)
              READ(5,3)NL
              IF(NL.EQ.0)NL=1
              DO 21 IN=1,NL
 18           WRITE(5,19)IN
 19           FORMAT('   L(',I1,') [4]: ',$)
              READ(5,3)LLIST(IN)
              IF(LLIST(IN).EQ.0)LLIST(IN)=4
 21           CONTINUE
 22           WRITE(5,23)
 23           FORMAT(' HOPPING SLOTS? [2400]: ',$)
              READ(5,24)NSLOTS
```

```
 24     FORMAT(I5)
        IF(NSLOTS.EQ.0)NSLOTS=2400
 25     WRITE(5,26)
 26     FORMAT(' HOW MANY GAMMA? [3]: ',$)
        READ(5,3)NG
        IF(NG.EQ.0)NG=3
        DO 31 IN=1,NG
 28     WRITE(5,29)IN,DG(IN)
 29     FORMAT('   GAMMA(',I2,') [',1PD8.1,']: ',$)
        READ(5,30,ERR=28)GAMLST(IN)
 30     FORMAT(D15.8)
        IF(GAMLST(IN).EQ.0.D0)GAMLST(IN)=DG(IN)
 31     CONTINUE
 38     WRITE(5,39)
 39     FORMAT(' HOW MANY EB/NJ? [126]: ',$)
        READ(5,34,ERR=38) NJ
 34     FORMAT(I3)
        IF(NJ.EQ.0) NJ=126
        IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 32
 40     WRITE(5,41)
 41     FORMAT(' STARTING VALUE FOR EB/NJ (DB) [0.]: ',$)
        READ(5,42,ERR=40) START
 42     FORMAT(F6.3)
        IF(NJ.EQ.1) RETURN
 35     WRITE(5,36)
 36     FORMAT(' DB INCREMENT FOR EB/NJ [.4]: ',$)
        READ(5,37,ERR=35) DBINC
 37     FORMAT(F6.3)
        IF(DBINC.EQ.0.) DBINC=0.400
        RETURN
        END
```



```
        SUBROUTINE PSUBE(RHON,RHOT,LL,M,BITS,
     $                   PE,D,IDSUB,NUSED,PRERR,IPSUB,EBNJ)
C
C COMPUTE UNCONDITIONAL ERROR PROBABILITY
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        INTEGER JAM(8),JAML(0:8)
        LOGICAL TRASH1, TRASH2
        VIRTUAL PRERR(625),IPSUB(625)
        VIRTUAL D(625),IDSUB(625)
        COMMON /RESET/ TRASH1, TRASH2
        PE=0.D0
        DO 199 LTHJ=0,LL
        JAML(0)=LTHJ
C FLAG TO MAKE SURE TRASH1 IS TRUE FOR FIRST TIME AROUND
        JAM(1)=-1
        DO 199 I1=0,LTHJ
        JAML(1)=I1
        ITER=1
 100    DO 101 I=ITER+1,M
C
C OUTERMOST NONSIGNAL LOOP ALWAYS STARTS FROM 0,
C OTHERS FROM THE CURRENT VALUE OF NEXT OUTER MORE LOOP INDEX
C
        IF(I.GT.2) THEN
          JAML(I)=JAML(I-1)
        ELSE
          JAML(I)=0
        END IF
 101    CONTINUE
        CALL EVENT(LL,M,JAML,PIE,D,IDSUB,NUSED)
        IF(PIE.EQ.0.D0) GOTO 190
C
C TRANSFORM TO EQUIVALENT LINEAR RECEIVER CONDITIONAL PROBABILITY
C
        RHOT10=RHOT1
        IF(JAML(0)-JAML(1).NE.LL) THEN
          RHOT1=RHON+(JAML(1)*(RHOT-RHON))/(LL-JAML(0)+JAML(1))
        ELSE
          RHOT1=RHOT
        END IF
        IF(RHOT10.NE.RHOT1) TRASH2=.TRUE.
        JAM1=JAM(1)
        DO 105 I=1,M
        JAM(I)=LL-JAML(0)+JAML(I)
 105    CONTINUE
        IF(JAM1.NE.JAM(1)) TRASH1=.TRUE.
        UCK=RHON/RHOT
        IF(RHOT1.NE.RHON) THEN
          EBNJ1=LL*RHON*RHOT1/(BITS*(RHOT1-RHON))
        ELSE
          EBNJ1=LL*RHON/BITS
        END IF
```

```
        C EMPIRICAL SWITCH-OVER POINT FOR COMPUTATIONAL METHOD
        C TO AVOID ROUND-OFF AND TRUNCATION ERRORS
0042            IF(EBNJ1.GT.31.6 .OR. EBNJ.GE.40.DO .OR.
             $     (RHON.GE.30.DO .AND. EBNJ.GE.25.DO)
             $     .OR. UCK.LT.1.01DO) THEN
0043              CALL PSEL1(JAM,LL,M,RHON,RHOT1,UCK,PROB)
0044            ELSE
0045              CALL PSEL2(JAM,LL,M,RHON,RHOT1,UCK,PROB)
0046              TRASH1=.TRUE.
0047              TRASH2=.TRUE.
0048            END IF
0049            PE=PE+PROB*PIE
0050      190   DO 195 I=2,M
0051            ITER=M+2-I
0052            JAML(ITER)=JAML(ITER)+1
0053            IF(JAML(ITER).LE.LTHJ) GOTO 100
0054      195   CONTINUE
0055      199   CONTINUE
0056            RETURN
0057            END
```



```
0001            SUBROUTINE EVENT(LL,M,JAML,PIE,D,IDSUB,NUSED)
        C
        C SUBROUTINE TO LOOK UP EVENT PROBABILITY FROM STORED ARRAY
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            LOGICAL*1 STORE,NONE
0004            DIMENSION JAML(9),LUP(9)
0005            VIRTUAL D(625),IDSUB(625)
0006            DIMENSION LOW(9)
0007            DATA STORE/.FALSE./, LOW/9*0/
        C SET UP ARRAY DESCRIPTION D(O:LL,...,O:LL) WITH M+1 DIMENSIONS
0008            DO 1 I=1,M+1
0009            LUP(I)=LL
0010      1     CONTINUE
        C COMPUTE LINEAR EQUIVALENT SUBSCRIPT FOR JAMMING EVENT
0011            CALL LOCN(M+1,LOW,LUP,JAML,ISUB)
        C LOOK UP THE VALUE, GET 0.DO IF NOT THERE
0012            CALL LOOKUP(PIE,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
0013            RETURN
0014            END
```

```
              SUBROUTINE GENPIE(LL,MM,NQ,NSLOTS,GOOD,MATRIX,MLOW,MINC,
         $                         MUP,PIE,D,IDSUB,NUSED)
      C
      C SUBROUTINE TO GENERATE EVENT PROBABILITIES
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              LOGICAL*1 GO,STORE,NONE,GOOD
              DIMENSION MATRIX(LL,MM),MLOW(LL,MM),MINC(LL,MM),MUP(LL,MM),
         $                 PIE(0:MM),IWORK(0:8),LOWWRK(0:8),LUPWRK(0:8)
              VIRTUAL D(625),IDSUB(625)
      C STORE=.FALSE. => DON'T STORE ZERO ELEMENTS OF SPARSE ARRAY.
              STORE=.FALSE.
              GOOD=.TRUE.
      9999    IF(NQ.LE.0) THEN
                GOOD=.FALSE.
                RETURN
              END IF
              DO 90 I=0,MM
              CALL PRIHOP(I,MM,NQ,NSLOTS,A)
              PIE(I)=A
        90    CONTINUE
              DO 95 J=1,MM
              DO 95 I=1,LL
              MLOW(I,J)=0
              MUP(I,J)=1
              MINC(I,J)=1
        95    CONTINUE
              NUSED=0
              DO 98 I=0,MM
              LOWWRK(I)=0
              LUPWRK(I)=LL
        98    CONTINUE
      C INITIALIZE MATRIX-INDEX LOOP
              CALL MLINIT(MATRIX,MLOW,LL,MM)
      999     CONTINUE
      C FORM COLUMN SUMS AND COUNT JAMMED HOPS AND COMPUTE P(EVENT)
              LTHJ=0
              P=1.D0
              DO 101 I=1,LL
              K=0
              DO 100 J=1,MM
              K=K+MATRIX(I,J)
       100    CONTINUE
              IF(K.NE.0) LTHJ=LTHJ+1
              P=P*PIE(K)
       101    CONTINUE
      C FORM JAMMING EVENT VECTOR
              IWORK(0)=LTHJ
              DO 102 J=1,MM
              IWORK(J)=0
              DO 102 I=1,LL
              IWORK(J)=IWORK(J)+MATRIX(I,J)
       102    CONTINUE
```



```
      C SORT NONSIGNAL CHANNELS
              DO 103 I=2,MM-1
              DO 103 J=I+1,MM
              IF(IWORK(J).LT.IWORK(I)) THEN
                ITEMP=IWORK(I)
                IWORK(I)=IWORK(J)
                IWORK(J)=ITEMP
              END IF
       103    CONTINUE
              CALL LOCN(MM+1,LOWWRK,LUPWRK,IWORK,ISUB)
              CALL LOOKUP(DOUT,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
              DOUT=DOUT+P
              CALL PUTIN(DOUT,D,IDSUB,NUSED,625,ISUB,IERR,STORE)
              IF(IERR.NE.0) STOP 'TOO MANY EVENTS'
      C ITERATE MATRIX-INDEX LOOP
              CALL MLITER(MATRIX,MLOW,MUP,MINC,LL,MM,GO)
              IF(GO) GOTO 999
              RETURN
              END
```

```
      SUBROUTINE PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C
C THIS SUBROUTINE INSERTS AN ELEMENT INTO A SPARSE ARRAY FOR
C WHICH ONLY THE NON-ZERO ELEMENTS ARE KEPT IN STORAGE IF
C THE SWITCH STORE IS .TRUE.
C
C THE DOUBLE PRECISION VALUE CIN IS STORED AS C(K) WHERE
C THE AUXILIARY ARRAY ICSUB IS USED TO KEEP TRACK OF THE
C SUBSCRIPTS OF THE CORRESPONDING ENTRIES IN C.
C LONG (4-BYTE) INTEGERS ARE USED TO ACCOMODATE LARGE
C SUBSCRIPT VALUES FOR THE SPARSE ARRAY C.
C
C USAGE:
C      LOGICAL*1 STORE
C      DOUBLE PRECISION C,CIN
C      VIRTUAL ICSUB(NMAX),C(NMAX)
C      CALL PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C WHERE
C  CIN = VALUE OF ELEMENT TO STORE
C  C = ARRAY IN WHICH NON-ZERO VALUES ARE ACTUALLY STORED
C  ICSUB = AUXILIARY ARRAY FOR ACTUAL SUBSCRIPT VALUES
C  NUSE = NUMBER OF ELEMENTS OF C CURRENTLY OCCUPIED
C  NMAX = SIZE OF ARRAY C
C  IERR = ERROR RETURN CODE, 0 IF NO ERROR OR 1 IF THERE IS
C         NO ROOM AVAILABLE IN C
C  STORE = .TRUE. TO STORE ZEROES EXPLICITLY, ELSE .FALSE.
C NOTE: IF CIN=0 AND THE SUBSCRIPT K IS FOUND IN ICSUB, THEN
C      THE ELEMENT IS DELETED BY SHIFTING DOWNWARD ALL
C      FOLLOWING ELEMENTS OF THE ARRAY
C
C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      VIRTUAL ICSUB(NMAX),C(NMAX)
      LOGICAL*1 STORE
      IERR=0
      IF(STORE) GOTO 5
      IF(CIN.EQ.0.D0) GOTO30
5     IF(NUSE.EQ.0)GOTO 20
      DO 10 I=1,NUSE
      IF(ICSUB(I).NE.K) GOTO 10
      C(I)=CIN
      RETURN
10    CONTINUE
      IF(NUSE.LT.NMAX) GOTO 20
      IERR=1
      RETURN
20    NUSE=NUSE+1
      ICSUB(NUSE)=K
      C(NUSE)=CIN
      RETURN
```



```
30    DO 40 I=1,NUSE
      J=I
      IF(ICSUB(I).EQ.K) GOTO 50
40    CONTINUE
      RETURN
C
C REMOVE THE ZEROED ELEMENT AND BUMP COUNT OF ENTRIES USED
C
50    DO 60 I=J,NUSE-1
      ICSUB(I)=ICSUB(I+1)
      C(I)=C(I+1)
60    CONTINUE
      NUSE=NUSE-1
      RETURN
      END
```

```
0001            SUBROUTINE LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
        C
        C THIS SUBROUTINE RETRIEVES AN ELEMENT OF A SPARSE ARRAY WHICH
        C HAS BEEN STORED COMPACTLY BY STORING ONLY NON-ZERO ELEMENTS.
        C
        C THE ARRAY IS DOUBLE PRECISION.
        C
        C USAGE:
        C       VIRTUAL ICSUB(NMAX), C(NMAX)
        C       LOGICAL*1 STORE, NONE
        C       DOUBLE PRECISION COUT
        C       CALL LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
        C WHERE
        C       COUT = VALUE OF C(K) (OUTPUT FROM SUBROUTINE)
        C          C = ARRAY USED TO STORE NON-ZERO ELEMENTS
        C      ICSUB = AUXILIARY ARRAY TO STORE ACTUAL SUBSCRIPTS
        C          N = NUMBER OF ELEMENTS OF C CURRENTLY IN USE
        C       NMAX = SIZE OF C
        C          K = SUBSCRIPT OF SPARSE ARRAY TO LOOK UP
        C      STORE = .TRUE. IF ZEROES STORED EXPLICITLY, ELSE .FALSE.
        C       NONE = .FALSE. IF ZEROES NOT STORED OR ZEROES STORED AND
        C                      ELEMENT IS FOUND IN THE STORED ARRAY
        C              .TRUE.  IF ZEROES ARE STORED AND THE ELEMENT IS
        C              NOT FOUND (OUTPUT QUANTITY)
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 11 JANUARY 1984
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            VIRTUAL ICSUB(NMAX),C(NMAX)
0004            LOGICAL*1 STORE, NONE
0005            NONE=.FALSE.
0006            DO 10 I=1,N
0007            IF(ICSUB(I).NE.K)GOTO 10
0008            COUT=C(I)
0009            RETURN
0010    10      CONTINUE
0011            IF(STORE) THEN
0012              NONE=.TRUE.
0013            ELSE
0014              COUT=0.
0015            END IF
0016            RETURN
0017            END
```



```
0001            SUBROUTINE LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C
        C THIS SUBROUTINE COMPUTES THE EQUIVALENT LINEAR SUBSCRIPT FOR
        C A MULTIDIMENSIONAL ARRAY OF NDIM DIMENSIONS
        C
        C IF THE ARRAY A IS DEFINED AS
        C       DIMENSION A(ILOW(1):IUP(1),...,ILOW(NDIM):IUP(NDIM))
        C AND ISUB(1),...,ISUB(NDIM) IS A SET OF SUBSCRIPTS FOR A,
        C THEN THIS SUBROUTINE RETURNS IN LINEAR THE OFFSET FROM THE
        C ORIGIN OF A TO THE ELEMENT A(ISUB(1),...,ISUB(NDIM)), ASSUMING
        C THE FIRST SUBSCRIPT VARIES MOST RAPIDLY.
        C
        C USAGE:
        C       DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
        C       DATA ILOW/lower limits of defined subscripts of array/
        C       DATA IUP/upper limits of defined subscripts of array/
        C       ...SET ISUB TO DESIRED SEQUENCE OF SUBSCRIPTS...
        C       CALL LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C WHERE
        C       NDIM = NUMBER OF DIMENSIONS THE ARRAY HAS
        C       ILOW = ARRAY OF LOWER SUBSCRIPT BOUNDS
        C        IUP = ARRAY OF UPPER SUBSCRIPT BOUNDS
        C       ISUB = ARRAY CONTAINING SUBSCRIPT FOR WHICH LOCATION IS
        C              TO BE COMPUTED
        C     LINEAR = RETURNED VALUE OF OFFSET INTO ARRAY IN MEMORY
        C
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 11 JANUARY 1984
        C
0002            DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
0003            LINEAR=0
0004            DO 10 I=1,NDIM-1
0005            J=NDIM-I+1
0006            LINEAR=(LINEAR+(ISUB(J)-ILOW(J)))*(IUP(J-1)-ILOW(J-1)+1)
0007    10      CONTINUE
0008            LINEAR=LINEAR+ISUB(1)-ILOW(1)
0009            RETURN
0010            END
```

```
      SUBROUTINE PRIHOP(KJAM,KM,KQ,KN,AIN)
C
C THIS SUBROUTINE COMPUTES THE EVENT PROBABILITY FOR ALL
C POSSIBLE JAMMING PATTERNS WITH NON-ZERO PROBABILITY FOR
C L=1 HOP/SYMBOL FOR RMFSK/FH IN PBNJ
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      AIN=0.D0
C IF THIS IS AN IMPOSSIBLE CASE, RETURN WITH RESULT = 0.0
      IF(KJAM.GT.MIN0(KQ,KM)) RETURN
      KPMAX=KJAM-1
      LPMAX=KM-KJAM-1
      JPMAX=KM-1
      IMAX=MAX0(KPMAX,LPMAX,JPMAX)
      PROD=1.D0
      Q=KQ
      DIFFNQ=KN-KQ
      EN=KN
      DO 100 LOOP=0,IMAX
      F=LOOP
      IF(LOOP.LE.KPMAX) PROD=PROD*(Q-F)
      IF(LOOP.LE.JPMAX) PROD=PROD/(EN-F)
      IF(LOOP.LE.LPMAX) PROD=PROD*(DIFFNQ-F)
  100 CONTINUE
      AIN=PROD
      RETURN
      END
```



```
      BLOCK DATA
C INITIALIZE SHARED CONSTANTS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /SHARE/ DG(31),DSNR(5,4)
      COMMON /SHARE2/ LOW(4),LINC(4)
C DEFAULT LISTS FOR INTERACTIVE PARAMETER INPUTS
C ARE SHARED WITH LARGE WORKING STORAGE ARRAYS SINCE THEY
C MAY BE DESTROYED ONCE THE INPUT PARAMETERS ARE SET UP
      DATA DG / .001D0, .002D0, .005D0,
     $          .01D0,  .02D0,  .05D0,
     $          .1D0,   .2D0,   .5D0, 1.D0, 21*0.D0/
      DATA DSNR /13.35247D0, 12.31330D0, 10.94443D0, 0.D0, 0.D0,
     $           10.606572D0,  9.6284D0,  8.35248D0, 0.D0, 0.D0,
     $            9.094010D0,  8.16900D0,  6.971995D0, 0.D0, 0.D0,
     $            8.078350D0,  7.19960D0,  6.069646D0, 0.D0, 0.D0/
C FREQUENTLY NEEDED CONSTANT ARRAYS AND SCALARS
      DATA LOW/4*0/,LINC/4*1/
      END
```

```
      SUBROUTINE PSEL1(JSUB,LLL,MM,RHON,RHOT,UCK,PROB)
C
C RANDOM MFSK/FH IN PARTIAL BAND TONE JAMMING,
C GIVEN A JAMMING EVENT
C
C        JSUB - JAMMING EVENT VECTOR
C          LL - NUMBER OF HOPS/SYMBOL
C          MM - ALPHABET SIZE
C        PROB - RESULTING CONDITIONAL ERROR PROBABILITY
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL PGRAND, DGAU20
      DIMENSION WORK(75), STACK(75), SAVE(75)
      INTEGER JSUB(4)
      INTEGER NCHAN(0:6)
      COMMON /PARMS/ BIGK, LL
C       AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C      ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C       NJ = NUMBER OF JAMMED HOPS
C      NNJ = NUMBER OF NON-JAMMED HOPS
      COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
      COMMON /JAMCNT/ NCHAN
      LL=LLL
      BIGK=UCK
      NJ=JSUB(1)
      NNJ=LL-NJ
      AJ=2.DO*NJ*RHOT
      ANJ=2.DO*NNJ*RHON
      ASHIFT=DMAX1(AJ,ANJ)
C SUBTRACT 1 FOR BENEFIT OF RAPID CHI-SQUARE DENSITY CALCULATION
      NJ=NJ-1
      NNJ=NNJ-1
C
C  COUNT NUMBER OF NON-SIGNAL CHANNELS WITH L HOPS JAMMED
C
      DO 1 I=0,LL
      NCHAN(I)=0
    1 CONTINUE
      DO 2 I=2,MM
      ISUB=JSUB(I)
      NCHAN(ISUB)=NCHAN(ISUB)+1
    2 CONTINUE
      CALL DLGS15(PGRAND,ASHIFT,TAIL)
      CALL ADQUAD(O.,ASHIFT,BODY,DGAU20,PGRAND,1.D-10,
     $            WORK,STACK,75,KODE)
      IF(KODE.NE.0) THEN
        WRITE(5,3) KODE
    3   FORMAT(' ADQUAD ERROR CODE = ',I1)
        STOP 'FATAL ERROR'
      END IF
   50 PROB=TAIL+BODY
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION PGRAND(BETA)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      LOGICAL TRASH1, TRASH2
      VIRTUAL PU1V(2053), PU1T(2053), IPU1T(2053),
     $        FLV(8191),  FLT(8191),  IFLT(8191)
      INTEGER NCHAN(0:6)
      COMMON /JAMCNT/ NCHAN
      COMMON /RESET/ TRASH1, TRASH2
      COMMON /PARMS/ BIGK, LL
C       AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C      ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C       NJ = NUMBER OF JAMMED HOPS
C      NNJ = NUMBER OF NON-JAMMED HOPS
      COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
      IF(TRASH1) THEN
C     TRASH THE SIGNAL CHANNEL DENSITY TABLES
        DO 1 I=1,2053
    1   PU1V(I)=0.DO
        DO 2 I=1,2053
    2   PU1T(I)=0.DO
        DO 3 I=1,2053
    3   IPU1T(I)=0
        TRASH1=.FALSE.
      END IF
      IF(TRASH2) THEN
C     TRASH THE NON-SIGNAL CHANNEL DENSITY TABLES
        DO 4 I=1,8191
    4   FLV(I)=0.DO
        DO 5 I=1,8191
    5   FLT(I)=0.DO
        DO 6 I=1,8191
    6   IFLT(I)=0
        TRASH2=.FALSE.
      END IF
      PROD=1.DO
      DO 10 I=0,LL
      IF(NCHAN(I).NE.0) THEN
        ISUB=IHASH(BETA,8191)
        JSUB=ISUB
   20   T=FLT(ISUB)
        IT=IFLT(ISUB)
        IF(T.EQ.0.DO .AND. IT.EQ.0) THEN
C       NOT FOUND, COMPUTE IT AND ENTER INTO TABLE
          X=FL(BETA,I)
          FLV(ISUB)=X
          FLT(ISUB)=BETA
          IFLT(ISUB)=I+1
        ELSE IF(T.NE.BETA .OR. IT.NE.I+1) THEN
C       NOT THIS ENTRY, TRY NEXT ONE
          ISUB=ISUB+1
          IF(ISUB.GT.8191) ISUB=ISUB-8191
```

```
                IF(ISUB.NE.JSUB) THEN
                  GOTO 20
                ELSE
C               HASH TABLE OVERFLOW, MUST COMPUTE, CAN NOT STORE
                  X=FL(BETA,I)
                END IF
              ELSE IF(T.EQ.BETA .AND. IT.EQ.I+1) THEN
C             GOT IT!
                X=FLV(ISUB)
              END IF
              PROD=PROD*DXI(X,NCHAN(I))
            END IF
   10     CONTINUE
          ISUB=IHASH(BETA,2053)
          JSUB=ISUB
   30     T=PUIT(ISUB)
          IT=IPUIT(ISUB)
          IF(T.EQ.0.D0 .AND. IT.EQ.0) THEN
C         NOT FOUND, COMPUTE IT AND ENTER INTO TABLE
            Y=PU1(BETA)
            PU1V(ISUB)=Y
            PUIT(ISUB)=BETA
            IPUIT(ISUB)=NJ+2
          ELSE IF(T.NE.BETA .OR. IT.NE.NJ+2) THEN
C         NOT THIS ENTRY, TRY NEXT
            ISUB=ISUB+1
            IF(ISUB.GT.2053) ISUB=ISUB-2053
            IF(ISUB.NE.JSUB) THEN
              GOTO 30
            ELSE
C           HASH TABLE OVERFLOWED
              Y=PU1(BETA)
            END IF
          ELSE IF(T.EQ.BETA .AND. IT.EQ.NJ+2) THEN
C         GOT IT!
            Y=PU1V(ISUB)
          END IF
          PGRAND=Y*(1.D0-PROD)
          RETURN
          END
```



```
          DOUBLE PRECISION FUNCTION PU1(Y)
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
          DIMENSION WORK(75), STACK(75), SAVE(75)
          EXTERNAL DGXVI, PU1G
          COMMON /PU1COM/ YY
C           AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C          ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C           NJ = NUMBER OF JAMMED HOPS
C          NNJ = NUMBER OF NON-JAMMED HOPS
          COMMON /PU1PAR/ AJ, ANJ, NJ, NNJ
          COMMON /PARMS/ BIGK, LL
          IF(Y.LE.0.D0) THEN
            PU1=0.D0
            RETURN
          END IF
C NJ HAS ALREADY HAD 1 SUBTRACTED FOR RAPID CHI-SQUARE EVALUATION...
          IF(NJ.NE.-1 .AND. NJ.NE.LL-1) THEN
C         WE MUST CONVOLVE TWO NONCENTRAL CHI-SQUARE DENSITIES
            YY=Y
            CALL ADQUA2(0.,Y,VALUE,DGXVI,PU1G,1.D-10,WORK,STACK,
     $                SAVE,75,KODE)
            IF(KODE.NE.0) THEN
              WRITE(5,1) KODE
    1         FORMAT(' PU1 ADQUA2 ERROR: KODE=',I2)
              STOP 'FATAL ERROR'
            END IF
            PU1=VALUE/BIGK
          ELSE
C         WE ONLY HAVE ONE NONCENTRAL CHI-SQUARE DENSITY WITH 2*LL
C         DEGREES OF FREEDOM
            IF(NJ.EQ.-1) THEN
C                 ALL HOPS UNJAMMED
              CALL CHISQE(Y,LL-1,ANJ,F,KODE)
            ELSE IF(NJ.EQ.LL-1) THEN
C                 ALL HOPS JAMMED
              CALL CHISQE(Y/BIGK,LL-1,AJ,F,KODE)
              F=F/BIGK
            END IF
            IF(KODE.NE.0) THEN
              WRITE(5,111) KODE
  111         FORMAT(' BESSEL FUNCTION ERROR CODE = ',I1)
              STOP 'FATAL ERROR'
            END IF
            PU1=F
          END IF
          RETURN
          END
```

PDP-11 FORTRAN-77 V4.0-1 16:58:24 15-Jul-86 Page 21
ACJAGC.FTN;36 /F77/TR:BLOCKS/WR

```
              DOUBLE PRECISION FUNCTION PU1G(X)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /PU1COM/ YY
C               AJ = NONCENTRAL PARAMETER FOR JAMMED HOPS
C              ANJ = NONCENTRAL PARAMETER FOR NON-JAMMED HOPS
C               NJ = NUMBER OF JAMMED HOPS
C              NNJ = NUMBER OF NON-JAMMED HOPS
              COMMCN /PU1PAR/ AJ, ANJ, NJ, NNJ
              COMMON /PARMS/ BIGK, LL
              CALL CHISQE(X/BIGK,NJ,AJ,F1,KODE)
              IF(KODE.NE.0) THEN
                WRITE(5,1) KODE
    1           FORMAT(' BESSEL FUNCTION ERROR CODE: ',I2)
                STOP 'FATAL IN JAMMED HOP DENSITY'
              END IF
              CALL CHISQE(YY-X,NNJ,ANJ,F2,KODE)
              IF(KODE.NE.0) THEN
                WRITE(5,1) KODE
                STOP 'FATAL IN UNJAMMED HOP DENSITY'
              END IF
              PU1G=F1*F2
              RETURN
              END
```

PDP-11 FORTRAN-77 V4.0-1 16:58:27 15-Jul-86 Page 22
ACJAGC.FTN;36 /F77/TR:BLOCKS/WR

```
              DOUBLE PRECISION FUNCTION FL(ALPHA,L)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /PARMS/ BIGK, LL
              IF(L.NE.0.AND.L.NE.LL) THEN
                R=(BIGK-1.)/BIGK
                BIGLOG=DLOG(BIGK)
                ARG=L*BIGLOG+ALPHA/2.
                START=DEXP(-ARG)
                FL1=L-1
                DO 100 N=0,100
                EN=N
                IF(N.EQ.0) THEN
                  PART=1.D0
                  TERM=DIEXP(ALPHA/2.D0,START,LL-1)
                  SUM=TERM
                ELSE
                  PART=PART*R*(EN+FL1)/EN
                  TERM=PART*DIEXP(ALPHA/2.D0,START,LL+N-1)
                  DUMMY=SUM+TERM
                  SUM=DUMMY
                END IF
                IF(DABS(TERM).LE.1.D-11*DABS(SUM)) GOTO 125
  100           CONTINUE
                STOP 'FL SUM DID NOT CONVERGE'
              ELSE
                IF(L.EQ.0) THEN
                  A=ALPHA/2.D0
                ELSE IF(L.EQ.LL) THEN
                  A=ALPHA/(2.D0*BIGK)
                END IF
                START=DEXP(-A)
                SUM=DIEXP(A,START,LL-1)
              END IF
  125         FL=1.D0-SUM
              RETURN
              END
```

```
0001            DOUBLE PRECISION FUNCTION DIEXP(X,START,IUP)
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            TERM=START
0004            DIEXP=TERM
0005            IF(IUP.EQ.0) RETURN
0006            DO 100 I=1,IUP
0007            F=I
0008            TERM=TERM*X/F
0009            DIEXP=DIEXP+TERM
0010      100   CONTINUE
0011            RETURN
0012            END
```



```
0001            SUBROUTINE CHISQE(X,N,A,DEN,KODE)
          C
          C  NON-CENTRAL CHI-SQUARE DENSITY FOR EVEN DEGREES OF FREEDOM
          C       DEGREES OF FREEDOM (M) IS M=2*N+2
          C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            B=DSQRT(X*A)
0004            CALL DXBESI(B,N,BESSEL,KODE)
0005            IF(KODE.NE.0) RETURN
0006            R=X/A
0007            IF(R.NE.0.D0) THEN
0008              POWER=R**(N/2.D0)
0009            ELSE
0010              IF(N.NE.0) THEN
0011                POWER=0.D0
0012              ELSE
0013                POWER=1.D0
0014              END IF
0015            END IF
0016            DEN=0.5D0*POWER*DEXP(B-0.5D0*(X+A))*BESSEL
0017            RETURN
0018            END
```

```
0001            SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,WORK,STACK,SAVE,N,KODE)
        C
        C ADAPTIVE QUADRATURE ALGORITHM
        C   XL - LOWER LIMIT OF INTEGRAL (IN)
        C   XU - UPPER LIMIT OF INTEGRAL (IN)
        C    Y - VALUE OF INTEGRAL (OUT)
        C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
        C              WITH CALLING SEQUENCE
        C              CALL QR(XL,XU,F,Y)
        C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
        C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
        C WORK - WORK ARRAY OF SIZE N (IN)
        C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
        C         SAME ARRAY AS WORK (IN)
        C  SAVE- THIRD WORK ARRAY OF SIZE N, MUST NOT BE
        C         SAME ARRAY AS WORK NOR SAME AS STACK (IN)
        C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
        C KODE - ERROR INDICATOR (OUT)
        C          0 -- NO ERROR
        C          1 -- WORK ARRAYS TOO SMALL
        C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
        C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
        C               ATTAINING REQUIRED ACCURACY
        C R. H. FRENCH, 14 AUGUST 1984
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL F
0004            DIMENSION WORK(N),STACK(N),SAVE(N)
0005            KODE=0
0006            Y=0.D0
0007            WORK(1)=XU
0008            A=XL
0009            NPTS=1
0010            EPS=TOL
0011            STACK(1)=EPS
0012            CALL QR(XL,XU,F,T)
0013            SAVE(1)=T
0014    10      B=WORK(NPTS)
0015            XM=(A+B)*0.5D0
0016            CALL QR(A,XM,F,P1)
0017            CALL QR(XM,B,F,P2)
0018            IF(DABS(T-P1-P2).LE.EPS) GOTO 20
        C SPLIT IT
0019            NPTS=NPTS+1
0020            IF(NPTS.GT.N) THEN
0021              KODE=1
0022              RETURN
0023            END IF
0024            WORK(NPTS)=XM
0025            EPS=EPS/2.D0
```



```
0026            IF(EPS.EQ.0.D0) THEN
0027              KODE=2
0028              RETURN
0029            END IF
0030            STACK(NPTS)=EPS
0031            T=P1
0032            SAVE(NPTS)=P2
0033            GOTO 10
        C FINISHED A PIECE
0034    20      Y=Y+P1+P2
0035            EPS=STACK(NPTS)
0036            T=SAVE(NPTS)
0037            NPTS=NPTS-1
0038            A=B
0039            IF(NPTS.EQ.0) RETURN
0040            GOTO 10
0041            END
```

```
0001          FUNCTION IHASH(BETA,ISIZE)
        C AD HOC HASHING FUNCTION
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          IF(BETA.LT.1.D0) THEN
0004            IF(BETA.GT.0.5D0) THEN
0005              B=1.D0/(1.01D0-BETA)
0006            ELSE
0007              B=50.D0+1.5D0/(BETA+0.011D0)
0008            END IF
0009          ELSE
0010            B=100000.D0*(BETA-DINT(BETA*1000.D0)/1000.D0)+187.D0
0011          END IF
0012          B=B*23.D0
0013          SIZE=ISIZE
0014          I=DMOD(B+0.5D0,SIZE)+0.5D0
0015          IHASH=I+1
0016          RETURN
0017          END
```



```
0001          SUBROUTINE DG16(A,B,F,ANSWER)
        C
        C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
        C
        C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
        C
        C R. H. FRENCH, 28 FEBRUARY 1986
        C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          DIMENSION X(8),W(8)
0004          DATA X/ 0.09501250983763744018 5D0,
     $                0.28160355077925891323 0D0,
     $                0.45801677765722738634 2D0,
     $                0.61787624440264374844 7D0,
     $                0.75540440835500303389 5D0,
     $                0.86563120238783174388 0D0,
     $                0.94457502307323257607 8D0,
     $                0.98940093499164993259 6D0 /
0005          DATA W/ 0.18945061045506849628 5D0,
     $                0.18260341504492358886 7D0,
     $                0.16915651939500253818 9D0,
     $                0.14959598881657673208 1D0,
     $                0.12462897125553387205 2D0,
     $                0.09515851168249278481 0D0,
     $                0.06225352393864789286 3D0,
     $                0.02715245941175409485 2D0 /
0006          ANSWER=0.D0
0007          BMA02=(B-A)/2.D0
0008          BPA02=(B+A)/2.D0
0009          DO 10 I=1,8
0010          C=X(I)*BMA02
0011          Y1=BPA02+C
0012          Y2=BPA02-C
0013          ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0014    10    CONTINUE
0015          ANSWER=ANSWER*BMA02
0016          RETURN
0017          END
```

```
0001            SUBROUTINE DGXVI(A,B,F,ANSWER)
          C
          C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
          C
          C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
          C
          C R. H. FRENCH, 28 FEBRUARY 1986
          C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            DIMENSION X(8),W(8)
0004            DATA X/ 0.0950125098376374401850D0,
            $           0.2816035507792589132300D0,
            $           0.4580167776572273863420D0,
            $           0.6178762444026437484470D0,
            $           0.7554044083550030338950D0,
            $           0.8656312023878317438800D0,
            $           0.9445750230732325760780D0,
            $           0.9894009349916499325960D0 /
0005            DATA W/ 0.1894506104550684962850D0,
            $           0.1826034150449235888670D0,
            $           0.1691565193950025381890D0,
            $           0.1495959888165767320810D0,
            $           0.1246289712555338720520D0,
            $           0.0951585116824927848100D0,
            $           0.0622535239386478928630D0,
            $           0.0271524594117540948520D0 /
0006            ANSWER=0.D0
0007            BMAO2=(B-A)/2.D0
0008            BPAO2=(B+A)/2.D0
0009            DO 10 I=1,8
0010            C=X(I)*BMAO2
0011            Y1=BPAO2+C
0012            Y2=BPAO2-C
0013            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0014      10    CONTINUE
0015            ANSWER=ANSWER*BMAO2
0016            RETURN
0017            END
```



```
0001            SUBROUTINE DLGS15(F,A,RESULT)
          C
          C DOUBLE PRECISION 15-POINT LAGUERRE INTEGRATION
          C FOR THE SHIFTED SEMI-INFINITE INTERVAL:
          C
          C      INFINITY
          C         -
          C        /
          C       /    F(X) DX  =  RESULT
          C      /
          C     -
          C    A
          C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            DIMENSION X(15), W(15)
0004            DATA X/ 0.093307812017D0,
            $           0.49269174030 2D0,
            $           1.21559541207 1D0,
            $           2.26994952620 4D0,
            $           3.66762272175 1D0,
            $           5.42533662741 4D0,
            $           7.56591622661 3D0,
            $          10.12022856801 9D0,
            $          13.13028248217 6D0,
            $          16.65440770833 0D0,
            $          20.77647889944 9D0,
            $          25.62389422672 9D0,
            $          31.40751916975 4D0,
            $          38.53068330648 6D0,
            $          48.02608557268 6D0 /
0005            DATA W/ 0.23957817031 1D0,
            $           0.56010084279 3D0,
            $           0.88700826291 9D0,
            $           1.22366440215 D0,
            $           1.57444872163 D0,
            $           1.94475197653 D0,
            $           2.34150205664 D0,
            $           2.77404192683 D0,
            $           3.25564334640 D0,
            $           3.80631171423 D0,
            $           4.45847775384 D0,
            $           5.27001778443 D0,
            $           6.35956346973 D0,
            $           8.03178763212 D0,
            $          11.52777210090 D0 /
0006            RESULT=0.D0
0007            DO 10 I=1,15
0008            RESULT=RESULT+W(I)*F(A+X(I))
0009      10    CONTINUE
0010            RETURN
0011            END
```

```
0001          SUBROUTINE PSEL2(JSUB,LLL,MM,RHON1,RHOT1,UCK,PROB)
        C
        C RANDOM MFSK/FH IN PARTIAL BAND TONE JAMMING,
        C GIVEN A JAMMING EVENT
        C
        C        JSUB - JAMMING EVENT VECTOR
        C          LL - NUMBER OF HOPS/SYMBOL
        C          MM - ALPHABET SIZE
        C        PROB - RESULTING CONDITIONAL ERROR PROBABILITY
        C
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          INTEGER RO,RL
0004          INTEGER JSUB(4)
0005          DIMENSION N(0:4),CO(0:12),CL(0:12),
      $                       COEFR(3),COEFK(3),COEFP(3),
      $                       INDEX(9),INDUP(9)
        C SYMBOLIC NAMES FOR INDEX-IDENTIFICATION SUBSCRIPTS
        C  (LOOP NUMBER MODULO 3)
0006          PARAMETER(LOOPR=1, LOOPK=2, LOOPP=0)
0007          COMMON /PARMS/ BIGK, LL
0008          COMMON /RRRCOM/ RHOT,RHON
        C
        C COMPUTE PARAMETERS
        C
0009          RHON=RHON1
0010          RHOT=RHOT1
0011          LL=LLL
0012          BIGK=UCK
0013          LM1=LL-1
0014          FKILI = DXI(BIGK-1.D0,LM1)
        C
        C COMPUTE POWERS OF FL
        C
0015          DO 10 I=0,LL
0016          N(I)=0
0017     10   CONTINUE
0018          DO 12 I=2,MM
0019          N(JSUB(I))=N(JSUB(I))+1
0020     12   CONTINUE
        C
        C OVER-ALL SUMMATION INITIALIZATION
        C
0021          SUM=0.D0
        C
        C --- START LOOP ON RO ---
        C
0022          DO 9000 RO=0,N(0)
0023          FRO=RO
0024          IF(RO.EQ.0) THEN
0025            COEFO=1.D0
0026          ELSE
0027            COEFO=-COEFO*((N(0)-FRO+1.D0)/FRO)
0028          END IF
```



```
        C
        C --- START LOOP ON KO ---
        C
0029          KOMAX=RO*LM1
        C
        C     PRE-COMPUTE THE J.C.P. MILLER COEFFICIENTS OVER KO!
        C
0030          CALL JCPMF(CO,KOMAX,RO,LM1)
0031          DO 8000 KO=0,KOMAX
        C
        C --- START LOOP ON RL ---
        C
0032          DO 7000 RL=0,N(LL)
0033          FRL=RL
0034          IF(RL.EQ.0) THEN
0035            COEFL=1.D0
0036          ELSE
0037            COEFL=-COEFL*((N(LL)-FRL+1.D0)/FRL)
0038          END IF
        C
        C --- START LOOP ON KL ---
        C
0039          KLMAX=RL*LM1
        C
        C     PRE-COMPUTE THE J.C.P. MILLER COEFFICIENTS OVER KL!
        C
0040          CALL JCPMF(CL,KLMAX,RL,LM1)
0041          DO 6000 KL=0,KLMAX
0042          IF(KL.EQ.0) THEN
0043            POWKL=1.D0
0044          ELSE
0045            POWKL=POWKL/BIGK
0046          END IF
0047          COEFKL=CL(KL)*POWKL
        C
        C --- START THE VARIABLE-LEVEL NESTED LOOPS
        C
0048          LOOPS=3*LM1
0049          IF(LOOPS.EQ.0) GOTO 555
        C  SET UP INDEX OF OUTERMOST LOOP
0050          INDEX(1)=0
        C  SET UP THE NON-VARYING UPPER LIMITS
0051          DO 22 I=1,LM1
0052          INDUP(3*I-2)=N(I)
0053     22   CONTINUE
        C
        C   MARK OUTER-MOST LOOP AS JUST ITERATED
        C
0054          NSUB=1
        C
        C   PERFORM INITIALIZATION CODE FOR LOOPS FROM JUST-ITERATED LOOP
        C   INWARD
```

```
      C
      C  FIRST. RESET INDICES OF ALL LOOPS WITHIN ONE JUST ITERATED
      C
       1000  DO 25 I=NSUB+1,LOOPS
             INDEX(I)=0
         25  CONTINUE
      C
      C  SECOND, PERFORM FRONT-OF-LOOP CODE FOR EACH LOOP FROM ITERATED
      C         LOOP ON INWARD
      C

             DO 30 I=NSUB,LOOPS
             IRKP=MOD(I,3)
      C   MAP 1,2,3 INTO 1; 4,5,6 INTO 2; ETC. FOR NESTING LEVEL
             NEST=(I+2)/3
             IF(IRKP.EQ.LOOPR) THEN
      C      WE ARE ITERATING AN R-LOOP
               IF(INDEX(I).EQ.0) THEN
      C        FIRST TIME THROUGH:
      C          (A) INITIALIZE THE COEFFICIENT
                 COEFR(NEST)=1.DO
               ELSE
      C        NOT FIRST TIME THROUGH:
      C          UPDATE THE COEFFICIENT
                 COEFR(NEST)=-(COEFR(NEST)/FK1L1)*
          $                 ((N(NEST)-INDEX(I)+1.DO)/INDEX(I))
               END IF
      C        SET UP VARIABLE UPPER LIMITS FOR THE ASSOCIATED
      C        K LOOP
               INDUP(I+1)=INDEX(I)
             ELSE IF(IRKP.EQ.LOOPK) THEN
      C      WE ARE ITERATING A K LOOP
               IF(INDEX(I).EQ.0) THEN
      C        FIRST TIME THROUGH:
      C          INITIALIZE THE COEFFICIENT
                 COEFK(NEST)=1.DO
               ELSE
      C        NOT FIRST TIME THROUGH:
      C          UPDATE THE COEFFICIENT
                 COEFK(NEST)=COEFK(NEST)*
          $                 ((INDEX(I-1)-INDEX(I)+1.DO)/INDEX(I))
               END IF
      C        SET UP THE UPPER LIMIT FOR THE ASSOCIATED P LOOP
               INDUP(I+1)=ICAPP(LL,NEST,INDEX(I-1),INDEX(I))
             ELSE IF(IRKP.EQ.LOOPP) THEN
      C      WE ARE ITERATING A P LOOP
               COEFP(NEST)=DEE(LL,NEST,INDEX(I-2),INDEX(I-1),INDEX(I),BIGK)
             END IF
         30  CONTINUE
      C
      C --- DO THE INTEGRAL ---
      C
```



```
        555  AO=RC+RL/BIGK
             IBO=KO+KL
             IF(LOOPS.GT.0) THEN
               DO 60 I=1,LM1
               AO=AO+INDEX(3*I-1)+(INDEX(3*I-2)-INDEX(3*I-1))/BIGK
               IBO=IBO+INDEX(3*I)
         60    CONTINUE
             END IF
             PART=GRAL(AO,IBO,JSUB(1))
      C
      C --- ADD A TERM TO THE OVERALL SUM ---
      C
             PROD=COEFO*CO(KO)*COEFL*COEFKL
             IF(LOOPS.GT.0) THEN
               DO 70 I=1,LM1
               PROD=PROD*COEFR(I)*COEFK(I)*COEFP(I)
         70    CONTINUE
             END IF
             PROD=PROD*PART
             SUM=SUM+PROD
      C
      C --- ITERATE THE LOOPS ---
      C
             IF(LOOPS.GT.0) THEN
               DO 80 NSUB=LOOPS,1,-1
               INDEX(NSUB)=INDEX(NSUB)+1
               IF(INDEX(NSUB).LE.INDUP(NSUB)) THEN
                 GOTO 1000
               END IF
         80    CONTINUE
      C
      C --- OUTERMOST VARIABLE-NEST LOOP REACHED ITS UPPER LIMIT, ---
      C --- DONE WITH THE VARIABLE-NEST LOOPS                     ---
      C
             END IF
      C
      C --- ITERATE THE NORMAL DO LOOPS ---
      C
       6000  CONTINUE
       7000  CONTINUE
       8000  CONTINUE
       9000  CONTINUE
             PROB=1.DO-SUM
             RETURN
             END
```

```
            DOUBLE PRECISION FUNCTION DEE(LL,NEST,IR,K,IP,BIGK)
      C
      C  THE FUNCTION D(P) FOR PBNJ/RMFSK
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            DIMENSION G(0:8)
            IF(LL.EQ.2) THEN
              IF(NEST.EQ.1) THEN
                DEE=DXI(BIGK,IR-K)
                IF(MOD(K,2).EQ.1) DEE=-DEE
              END IF
            ELSE IF(LL.EQ.3) THEN
              IF(NEST.EQ.1) THEN
                DEE=DXI(BIGK*BIGK,IR-K)*DBINCO(K,IP)*
     $              DXI(2.DO*BIGK-1.DO,K-IP)*DXI(BIGK-1.DO,IP)
                IF(MOD(K,2).EQ.1) DEE=-DEE
              ELSE IF(NEST.EQ.2) THEN
                DEE=DBINCO(IR-K,IP)*DXI(BIGK*(BIGK-2.DO),IR-K-IP)*
     $              DXI(BIGK-1.DO,IP)
              END IF
            ELSE IF(LL.EQ.4) THEN
              IF(NEST.EQ.1) THEN
                DENOM=3.DO*BIGK*BIGK-3.DO*BIGK+1.DO
                COEF1=(2.DO*BIGK*BIGK-3.DO*BIGK+1.DO)/DENOM
                G(0)=1.DO
                G(1)=K*COEF1
                BK1=BIGK-1.DO
                COEF2=0.5DO*BK1*BK1/DENOM
                DO 41 N=2,IP
                G(N)=((K+1-N)*COEF1*G(N-1) + (2*K+2-N)*COEF2*G(N-2))/N
   41           CONTINUE
                DEE=DXI(BIGK,3*(IR-K))*DXI(DENOM,K)*G(IP)
                IF(MOD(K,2).EQ.1) DEE=-DEE
              ELSE IF(NEST.EQ.2) THEN
                IUP=MINO(IP,K)
                ILOW=MAXO(0,IP-IR+K)
                BK3=BIGK-3.DO
                TK1=3.DO*BIGK-1.DO
                BASE=BK3*BIGK/TK1
                SUMQ=0.DO
                DO 42 IQ=ILOW,IUP
                SUMQ=SUMQ+DBINCO(IR-K,IP-IQ)*DBINCO(K,IQ)
     $                 *DXI(BK3,IR-K-IP)*DXI(TK1,K)*DXI(BASE,IQ)
   42           CONTINUE
                DEE=SUMQ*DXI(BIGK,IR+IR-K-K-IP)*DXI(BIGK-1.DO,IP)
              ELSE IF(NEST.EQ.3) THEN
                G(0)=1.DO
                DENOM=DXI(BIGK,3)-3.DO*BIGK*BIGK+3.DO*BIGK
                COEF1=(BIGK*BIGK-3.DO*BIGK+2.DO)/DENOM
                G(1)=(IR-K)*COEF1
                BK1=BIGK-1.DO
                COEF2=0.5DO*BK1*BK1/(BIGK*DENOM)
```



```
                DO 43 N=2,IP
                G(N)=((IR-K+1-N)*COEF1*G(N-1)+
     $               (2*(IR-K+1)-N)*COEF2*G(N-2))/N
   43           CONTINUE
                DEE=DXI(DENOM,IR-K)*G(IP)
                IF(MOD(K,2).EQ.1) DEE=-DEE
              END IF
            END IF
            RETURN
            END
```

```
      FUNCTION ICAPP(LL,NEST,IR,K)
C
C COMPUTE UPPER SUMMATION LIMIT PL
C
      IF(LL.LE.2) THEN
        ICAPP=0
      ELSE IF(LL.EQ.3) THEN
        IF(NEST.EQ.1) THEN
          ICAPP=K
        ELSE IF(NEST.EQ.2) THEN
          ICAPP=IR-K
        END IF
      ELSE IF(LL.EQ.4) THEN
        IF(NEST.EQ.1) THEN
          ICAPP=2*K
        ELSE IF(NEST.EQ.2) THEN
          ICAPP=IR
        ELSE IF(NEST.EQ.3) THEN
          ICAPP=2*(IR-K)
        END IF
      END IF
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION GRAL(A0,IBO,L1)
C
C THE INNER INTEGRAL FOR CONDITIONAL ERROR PROBABILITY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RRRCOM/ RHOT,RHON
      COMMON /PARMS/ BIGK, LL
      A1=1.D0+A0
      IF(L1.GT.0 .AND. L1.LT.LL) THEN
        BK1=BIGK-1.D0
        BASE=BK1/(1.D0+BIGK*A0)
        ARG1=(L1-LL)*RHON/A1
        AR2=-L1*RHOT*BIGK/(1.D0+BIGK*A0)
        ARG2=AR2*A1/BK1
        POWER=1.D0
        SUM=0.D0
        DO 100 IQ=0,IBO
        IF(IQ.GT.0) POWER=POWER*BASE
        PM=DSQRT(POWER)
        SUM=GNLGPM(IQ+LL-1,IBO-IQ,ARG1,PM)*GNLGPM(L1-1,IQ,ARG2,PM)+SUM
100     CONTINUE
        F=1.D0
        DO 110 I=1,IBO
        F=F*(I/A1)
110     CONTINUE
        F=(F/DXI(A1,LL-L1))/DXI(1.D0+BIGK*A0,L1)
        GRAL=SUM*F*DEXP((ARG1+AR2)*A0)
      ELSE IF(L1.EQ.0) THEN
        ARG1=-LL*RHON/A1
        START=DEXP(ARG1*A0-LL*DLOG(A1))
        DO 200 I=1,IBO
        START=START*(I/A1)
200     CONTINUE
        GRAL=GNLGPM(LL-1,IBO,ARG1,START)
      ELSE IF(L1.EQ.LL) THEN
        A2=1.D0+BIGK*A0
        ARG1=-LL*RHOT/A2
        BASE=BIGK/A2
        START=DEXP(ARG1*BIGK*A0-LL*DLOG(A2))
        DO 300 I=1,IBO
        START=START*(I*BASE)
300     CONTINUE
        GRAL=GNLGPM(LL-1,IBO,ARG1,START)
      END IF
      RETURN
      END
```

```
        DOUBLE PRECISION FUNCTION GNLGPM(IALFA,N,X,PREMUL)
C
C GENERALIZED LAGUERRE POLYNOMIALS WITH PREMULTIPLICATION BY
C A WEIGHTING FACTOR (TO DEFER THE INEVITABLE OVERFLOWS)
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        TERM=DBINCO(N+IALFA,N)*PREMUL
        SUM=TERM
        IF(N.EQ.0) GOTO 200
        DO 100 M=1,N
        TERM=-TERM*((X/M)*((N-M+1.DO)/(IALFA+M)))
        SUM=SUM+TERM
  100   CONTINUE
  200   GNLGPM=SUM
        RETURN
        END
```



```
        SUBROUTINE JCPMF(C,KMAX,R,LM1)
C
C COMPUTE J.C.P. MILLER COEFFICIENTS DIVIDED BY K!
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        INTEGER R
        DIMENSION C(0:KMAX)
        C(0)=1.DO
        IF(LM1.EQ.0) RETURN
        DO 100 K=1,KMAX
        SUM=0.DO
        NUP=MINO(K,LM1)
        BC=1.DO
        DO 90 N=1,NUP
        BC=BC*((K-N+1.DO)/N)
        SUM=SUM+BC*((R+1.DO)*N-K)*C(K-N)
   90   CONTINUE
        C(K)=SUM/K
  100   CONTINUE
        FAC=1.DO
        DO 110 K=1,KMAX
        FAC=FAC*K
        C(K)=C(K)/FAC
  110   CONTINUE
        RETURN
        END
```

```
PDP-11 FORTRAN-77 V4.0-1        16:59:20    15-Jul-86           Page 41
ACJAGC.FTN;36           /F77/TR:BLOCKS/WR

            SUBROUTINE MLINIT(LMAT,LLOW,LMAXC,LMAXR)
      C
      C THIS SUBROUTINE INITIALIZES A "MATRIX DO-LOOP" STRUCTURE
      C DEFINED BY THE FOLLOWING PSEUDO-FORTRAN CODE:
      C       DO 100 LMAT(1,1)=LLOW(1,1),LUP(1,1),LINC(1,1)
      C                        :
      C                        :
      C       DO 100 LMAT(LMAXC,1)=LLOW(LMAXC,1),LUP(LMAXC,1),LINC(LMAXC,1)
      C       DO 100 LMAT(1,2)=LLOW(1,2),LUP(1,2),LINC(1,2)
      C                        :
      C                        :
      C       DO 100 LMAT(LMAXC,2)=LLOW(LMAXC,2),LUP(LMAXC,2),LINC(LMAXC,2)
      C       DO 100 LMAT(1,LMAXR)=LLOW(1,LMAXR),LUP(1,LMAXR),LINC(1,LMAXR)
      C                        :
      C                        :
      C       DO 100 LMAT(LMAXC,LMAXR)=LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR),
      C     $                          LINC(LMAXC,LMAXR)
      C                        :
      C       (STATEMENTS IN RANGE OF LOOP)
      C                        :
      C 100   CONTINUE
      C
      C THE COMPANION ROUTINE MLITER HANDLES THE LOOP CONTROL AT THE
      C CONTINUE STATEMENT IN THE ABOVE STRUCTURE
      C
      C USAGE:
      C       LOGICAL*1 GO
      C       DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR)
      C       DIMENSION LINC(LMAXC,LMAXR)
      C (INITIALIZE MATRIX LLOW TO STARTING VALUES OF THE NESTED LOOPS)
      C (INITIALIZE MATRIX LUP TO STOPING VALUES OF THE NESTED LOOPS)
      C (INITIALIZE MATRIX LINC TO INCREMENTS OF THE LOOPS)
      C       CALL MLINIT(LMAT,LLOW,LMAXC,LMAXR)
      C   100 CONTINUE
      C         .
      C         . (STATEMENTS IN RANGE OF LOOPS)
      C         .
      C       CALL MLITER(LMAT,LLOW,LUP,LINC,LMAXC,LMAXR,GO)
      C       IF(GO)GOTO 100
      C WHERE
      C    LMAT = ARRAY FOR STORAGE OF LOOP INDICES. LMAT(1,1) IS THE
      C           OUTER-MOST LOOP; LMAT(LMAXC,LMAXR), THE INNER-MOST LOOP.
      C    LLOW = ARRAY FOR STORAGE OF LOOP STARTING VALUES, IN SAME
      C           SEQUENCE AS LMAT
      C    LUP = ARRAY FOR STORAGE OF LOOP FINAL VALUES, IN SAME
      C          SEQUENCE AS LMAT
      C    LINC = ARRAY FOR STORAGE OF LOOP INCREMENTS, IN SAME
      C           SEQUENCE AS LMAT
      C    LMAX = NUMBER OF LOOPS NESTED
      C    GO = LOGICAL VARIABLE, .TRUE. IF JUMP BACK TO BEGINNING OF
      C         STATEMENTS IN THE RANGE OF THE LOOP SHOULD OCCUR,
      C         .FALSE. OTHERWISE (I.E. OUTER-MOST LOOP TERMINATED)
```

```
PDP-11 FORTRAN-77 V4.0-1        16:59:20    15-Jul-86           Page 42
ACJAGC.FTN;36           /F77/TR:BLOCKS/WR

      C
      C PROGRAMMER: ROBERT H. FRENCH          DATE: 10 MARCH 1986
      C
            DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR)
            DO 1 N=1,LMAXR
            DO 1 M=1,LMAXC
            LMAT(M,N)=LLOW(M,N)
      1     CONTINUE
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        16:59:22    15-Jul-86           Page 43
ACJAGC.FTN;36           /F77/TR:BLOCKS/WR

            SUBROUTINE MLITER(LMAT,LLOW,LUP,LINC,LMAXC,LMAXR,GO)
      C
      C LOOP ITERATION LOGIC FOR A "MATRIX DO-LOOP"
      C
      C SEE DETAILED COMMENTS IN SUBROUTINE MLINIT FOR USAGE AND
      C PARAMETER DEFINITIONS
      C
      C PROGRAMMER: ROBERT H. FRENCH
      C       DATE: 10 MARCH 1986
      C
            LOGICAL*1 GO
            DIMENSION LMAT(LMAXC,LMAXR),LLOW(LMAXC,LMAXR),LUP(LMAXC,LMAXR)
            DIMENSION LINC(LMAXC,LMAXR)
            GO=.TRUE.
            DO 100 NDX=1,LMAXR
            NSUB=LMAXR+1-NDX
            DO 100 MDX=1,LMAXC
            MSUB=LMAXC+1-MDX
            LMAT(MSUB,NSUB)=LMAT(MSUB,NSUB)+LINC(MSUB,NSUB)
            IF((LINC(MSUB,NSUB).GE.0.AND.LMAT(MSUB,NSUB).LE.LUP(MSUB,NSUB))
          $                          .OR.
          $  (LINC(MSUB,NSUB).LT.0.AND.LMAT(MSUB,NSUB).GE.LUP(MSUB,NSUB)))
          $  RETURN
            LMAT(MSUB,NSUB)=LLOW(MSUB,NSUB)
      100   CONTINUE
            GO=.FALSE.
            RETURN
            END
```

APPENDIX G

COMPUTER PROGRAM FOR PLOTTING GRAPHICAL RESULTS FOR THE ANY-CHANNEL-JAMMED ADAPTIVE GAIN CONTROL RECEIVER

The following pages contain the source code listing for the FORTRAN-77 program used to produce the plotted graphical results for the any-channel-jammed adaptive gain control receiver for FH/RMFSK. The program makes use of the Hewlett-Packard Industry Standard Plotting Package (ISPP) to drive an HP-7470A plotter.

With minor modifications in annotations and file names, this program will serve to plot results for all other receivers. For brevity, the other versions of the plotting program are not included in this report.

```
0001          PROGRAM RNHOPP
      C THIS PROGRAM PLOTS THE ERROR PROBABILITY FOR RANDOM M-ARY
      C FSK/FH WITH MULTIPLE HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
      C NOISE JAMMING
      C
      C ANALYSIS: L. E. MILLER, R. H. FRENCH
      C PROGRAM: R. H. FRENCH
      C
      C
      C V 3.0.0 - PLOTS
      C
      C
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          PARAMETER (NJ=126)
0004          PARAMETER (NJ2=NJ+2)
0005          REAL*4 PRLOG(NJ2),DBSJR(NJ2),RTEMP
0006          CHARACTER*13 FNAME
      C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
0007          COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,
     $                        GAMLST(31),K,MM
      C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
0008          COMMON /SIZE/ NO,NL,NG
0009          CALL JSLGGO
0010          CALL GET
0011          PRLOG(NJ+1)=-5.
0012          PRLOG(NJ+2)=0.625
0013          DBSJR(NJ+1)=0.
0014          DBSJR(NJ+2)=-10.
0015          CALL SETUP(KODE)
0016          IF(KODE.NE.0) STOP 1
0017          DO 900 IL=1,NL
0018          LL=LLIST(IL)
0019          DO 800 IO=1,NO
0020          IOOUT=DEBNOL(IO)
      C
      C DRAW THE BOX
      C
0021          CALL SLOW
0022          CALL PLOT(1.25,6.0,-3)
0023          CALL LOGAX1(0.,0.,'PROBABILITY OF BIT ERROR',
     $                LEN('PROBABILITY OF BIT ERROR'),8.,0.,-5.,5)
0024          CALL AXIS1(0.,0.,'BIT ENERGY TO JAMMING DENSITY RATIO (dB)',
     $               -LEN('BIT ENERGY TO JAMMING DENSITY RATIO (dB)'),
     $               5.,270.,0.,10.)
      C COMPLETE THE BOX
0025          CALL AXIS1(8.,0.,-1,0,5.,270.,0.,10.)
0026          CALL LOGAX1(0.,-5.,1,0,8.,0.,-5.,5)
      C ANNOTATE M, L, EB/NO
0027          CALL SYMBOL(7.5,-2.5,0.14,'M=',270.,2)
0028          RTEMP=MM
0029          CALL NUMBER(999.,999.,0.14,RTEMP,270.,-1)
0030          CALL SYMBOL(999.,999.,0.14,'  L=',270.,4)
```



```
0031          RTEMP=LL
0032          CALL NUMBER(999.,999.,0.15,RTEMP,270.,-1)
0033          CALL SYMBOL(7.25,-2.5,0.14,'E',270.,1)
0034          CALL SYMBOL(7.215,999.,0.09,'b',270.,1)
0035          CALL SYMBOL(7.25,999.,0.14,'/N',270.,2)
0036          CALL SYMBOL(7.215,999.,0.09,'O',270.,1)
0037          CALL SYMBOL(7.25,999.,0.14,'=',270.,1)
0038          RTEMP=DEBNOL(IO)
0039          CALL NUMBER(999.,999.,0.14,RTEMP,270.,6)
0040          CALL SYMBOL(999.,999.,0.14,' dB',270.,3)
0041          RTEMP=NSLOTS
0042          CALL NUMBER(7.0,-2.5,0.14,RTEMP,270.,-1)
0043          CALL SYMBOL(999.,999.,0.14,' SLOTS',270.,6)
0044          CALL SYMBOL(6.75,-2.5,0.14,'ACJ-AGC',270.,7)
0045          CALL PENUP
0046          DO 700 IG=1,NG
0047          GAMMA=GAMLST(IG)
0048          IGOUT=GAMMA*1000.D0+0.5D0
      C
      C OPEN DATA FILE
      C
0049          WRITE(FNAME,730) MM,LL,IOOUT,IGOUT
0050   730    FORMAT('A',I1,I1,I2.2,I4.4,'.DAT')
0051          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
     $             ERR=740)
0052          GOTO 750
0053   740    WRITE(5,741) FNAME
0054   741    FORMAT(1X,A13)
0055          STOP 'FILE NOT FOUND'
0056   750    READ(4) MMIN,LLIN,DEBNOI,ISLOTS,GAMMAI,
     $              (DBSJR(IN),IN=1,NJ),
     $              (PRLOG(IN),IN=1,NJ)
0057          IF(MMIN.NE.MM .OR. LLIN.NE.LL .OR. DEBNOI.NE.DEBNOL(IO)
     $        .OR. ISLOTS.NE.NSLOTS .OR. GAMMAI.NE.GAMMA) THEN
0058            WRITE(5,741) FNAME
0059            STOP 'FILE CONTENTS ERROR'
0060          END IF
      C
      C INTERPOLATE TO EDGE OF THE GRAPH
      C
0061          DO 790 I=1,NJ
0062          NPTS=1
0063          IF(PRLOG(I).GE.-5.) GOTO 790
0064          DY=PRLOG(I)-PRLOG(I-1)
0065          DX=DBSJR(I)-DBSJR(I-1)
0066          PART=(-5.)-PRLOG(I-1)
0067          SLOPE=DX/DY
0068          DBSJR(I)=DBSJR(I-1)+SLOPE*PART
0069          PRLOG(I)=-5.
0070          GOTO 791
0071   790    CONTINUE
```

```
 791   PRLOG(NPTS+1)=-5.
       PRLOG(NPTS+2)=0.625
       DBSJR(NPTS+1)=0.
       DBSJR(NPTS+2)=-10.
C
C PLOT THE DATA
C
       CALL LINE(PRLOG,DBSJR,NPTS,1,0,0)
       CALL PENUP
       CLOSE(UNIT=4)
 700   CONTINUE
C
C CLOSE AND FLUSH IF DONE, ELSE NEW PAGE
C
       IF(IL. EQ. NL  .AND.  IO .EQ. NO) THEN
         CALL NEWPEN(0)
         CALL PLOT(0.,0.,999)
       ELSE
         CALL NEWPLT
       END IF
 800   CONTINUE
 900   CONTINUE
       STOP 0
       END
```



```
       SUBROUTINE GET
C
C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
       CHARACTER*9 FIELD,BLANK9
       DIMENSION DG(31),DSNR(5,4)
       COMMON /INPUTS/ DEBNOL(5),LLIST(4),NSLOTS,
     $               GAMLST(31),K,MM
       COMMON /SIZE/ NO,NL,NG
       DATA DG / .001D0, .002D0, .005D0,
     $           .01D0,  .02D0,  .05D0,
     $           .1D0,   .2D0,   .5D0, 1.D0, 21*0.D0/
       DATA DSNR /13.35247D0, 12.31330D0, 10.94443D0,  0.D0, 0.D0,
     $           10.60657200, 9.6284D0,  8.35248D0,  0.D0, 0.D0,
     $            9.09401D0,  8.1690D0,  6.971995D0, 0.D0, 0.D0,
     $            8.07835D0,  7.1996D0,  6.069646D0, 0.D0, 0.D0/
       DATA BLANK9/'         '/
 32    WRITE(5,33)
 33    FORMAT(' BITS/SYMBOL (K) [2]: ',$)
       READ(5,3)K
       IF(K.EQ.0)K=2
       MM=2**K
 1     WRITE(5,2)
 2     FORMAT(' HOW MANY EB/NO? [1]: ',$)
       READ(5,3)NO
 3     FORMAT(I2)
       IF(NO.EQ.0)NO=1
       DO 7 IN=1,NO
       IF(K.LE.4) THEN
         DO=DSNR(IN,K)
       ELSE
         DO=0.D0
       END IF
 4     WRITE(5,5)IN,DO
 5     FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
       READ(5,6)FIELD
 6     FORMAT(A9)
       IF(FIELD.EQ.BLANK9) THEN
         DEBNOL(IN)=DO
       ELSE
         DECODE(9,61,FIELD)DEBNOL(IN)
 61      FORMAT(F9.6)
       END IF
 7     CONTINUE
 15    WRITE(5,16)
 16    FORMAT(' HOW MANY L? [1]: ',$)
       READ(5,3)NL
       IF(NL.EQ.0)NL=1
       DO 21 IN=1,NL
 18    WRITE(5,19)IN
 19    FORMAT('   L(',I1,') [4]: ',$)
```

```
            READ(5,3)LLIST(IN)
            IF(LLIST(IN).EQ.0)LLIST(IN)=4
      21    CONTINUE
      22    WRITE(5,23)
      23    FORMAT(' HOPPING SLOTS? [2400]: ',$)
            READ(5,24)NSLOTS
      24    FORMAT(I5)
            IF(NSLOTS.EQ.0)NSLOTS=2400
      25    WRITE(5,26)
      26    FORMAT(' HOW MANY GAMMA? [10]: ',$)
            READ(5,3)NG
            IF(NG.EQ.0)NG=10
            DO 31 IN=1,NG
      28    WRITE(5,29)IN,DG(IN)
      29    FORMAT('    GAMMA(',I2,') [',1PD8.1,']: ',$)
            READ(5,30)GAMLST(IN)
      30    FORMAT(D15.8)
            IF(GAMLST(IN).EQ.0.D0)GAMLST(IN)=DG(IN)
      31    CONTINUE
            RETURN
            END
```

APPENDIX H

COMPUTER PROGRAM FOR

CLIPPER RECEIVER WITH M=2 AND L=2

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the clipper receiver for the case of M=2 and L=2 with the jamming fraction γ=q/N as a parameter.

```
0001            PROGRAM CLPRAN
        C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM M-ARY
        C FSK/FH WITH 2 HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
        C NOISE JAMMING BY NUMERICAL INTEGRATION FOR THE CLIPPER RECEIVER
        C
        C ANALYSIS: L. E. MILLER, R. H. FRENCH
        C PROGRAM: R. H. FRENCH
        C
        C V 2.1.0 - COMPUTATIONS ONLY
        C
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            PARAMETER (LJ=51)
0004            CHARACTER*1 YES, NO, REPLY, BLANK
0005            CHARACTER*13 FNAME, GNAME
0006            LOGICAL DOTAU
0007            LOGICAL*1 GOOD
0008            REAL*4 PRLOG(LJ),DBSJR(LJ)
0009            VIRTUAL A(100),IASUB(100),C(625),ICSUB(625)
0010            VIRTUAL D(625),IDSUB(625),PRERR(625),IPSUB(625)
        C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
0011            COMMON /INPUTS/ DEBNOL(3),NSLOTS,GAMLST(10),K,MM
        C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
0012            COMMON /SIZE/ NO,NG
0013            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
             $                  TAU, TAU2, TAUK, TAUK2
0014            COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
             $                  RHON8, RHOT8, RHONH, RHOTH
0015            DATA YES, NO, BLANK /'Y', 'N', ' '/
0016            CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0017            CALL GET(NJ,START,DBINC)
0018            WORBIT=0.5D0*MM/(MM-1.D0)
0019            DO 800 IO=1,NO
0020            DOTAU=.TRUE.
0021            EBNO=10.D0**(DEBNOL(IO)/10.D0)
0022            RHON=K*EBNO/2.D0
0023            RHON2=2.D0*RHON
0024            RHON4=4.D0*RHON
0025            RHON8=8.D0*RHON
0026            RHONH=RHON/2.D0
0027            IOOUT=DEBNOL(IO)
0028            DO 700 IG=1,NG
0029            GAMMA=GAMLST(IG)
0030            NQ=GAMMA*NSLOTS+0.5D0
        C
        C OPEN DATA FILE
        C
0031            IGOUT=GAMMA*1000.D0+0.5D0
0032            WRITE(FNAME,730) MM,IOOUT,IGOUT
0033      730   FORMAT('C',I1,'2',I2.2,I4.4,'.DAT')
0034            WRITE(6,776) MM,DEBNOL(IO),GAMMA
0035      776   FORMAT(' M=',I2,5X,'L=2',5X,'EB/NO=',F8.4,5X,'GAMMA=',1PD10.3)
```



```
0036            WRITE(5,733) FNAME
0037      733   FORMAT(' WORKING ON ',A13)
0038            OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
             $       ERR=750)
        C
        C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
        C
0039     1300   READ(4) MMIN, EBNOIN, NSLIN, GAMIN, TAU, TAU2, PEOO
        C
        C WE HAVE READ A VALUE OF TAU, SO WE WON'T NEED TO RECOMPUTE
        C IT UNTIL EITHER EB/NO, MM, OR LL CHANGES
        C
0040            DOTAU=.FALSE.
0041            JJ=0
0042      740   JJ=JJ+1
0043            READ(4,END=742) DBSJR(JJ), PRLOG(JJ)
0044            GOTO 740
0045      742   CLOSE(UNIT=4)
0046            WRITE(5,7420)
0047     7420   FORMAT(' FIX-UP RUN? [N] ',$)
0048            READ(5,7421) REPLY
0049     7421   FORMAT(A1)
0050            IF(REPLY.EQ.BLANK) REPLY=NO
0051            IF(REPLY.EQ.NO) GOTO 755
0052            WRITE(5,7422)
0053     7422   FORMAT(' HOW MANY GOOD POINTS? ',$)
0054            READ(5,7423) JGOOD
0055     7423   FORMAT(I3)
0056            IF(JGOOD.GT.JJ) JGOOD=JJ
        C CREATE FILE CONTAINING ONLY GOOD DATA IN PROGRESS FORMAT
0057            OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
0058            WRITE(4) MM,DEBNOL(IO),NSLOTS,GAMMA,TAU,TAU2,PEOO
0059            DO 7424 IJ=1,JGOOD
0060            WRITE(4) DBSJR(IJ), PRLOG(IJ)
0061     7424   CONTINUE
0062            CLOSE(UNIT=4)
0063            JJ=JGOOD+1
0064            GOTO 755
        C NO EXISTING FILE, THIS IS THE FIRST TIME:
        C CREATE FILE HEADER RECORD
0065      750   JJ=1
        C    ... WE CAN'T READ TAU FROM A FILE, SO IF THE FIRST TIME
        C       THROUGH THE LOOP ON EB/NO WE MUST COMPUTE IT.  BUT IF
        C       EB/NO HASN'T CHANGED, WE DON'T NEED TO RECOMPUTE IT SINCE
        C       THE THRESHOLD IS NOT A FUNCTION OF GAMMA NOR OF EB/NJ.
0066            IF(DOTAU) THEN
0067              CALL SETTAU(MM,PEOO)
0068              DOTAU=.FALSE.
0069              WRITE(5,1991) MM, TAU
0070              WRITE(6,1991) MM, TAU
0071     1991     FORMAT(' M=',I2,'   L=2    OPT THRES = ',1PD15.8)
0072            END IF
```

```
        OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
        WRITE(4) MM,DEBNOL(10),NSLOTS,GAMMA,TAU,TAU2,PEOO
        CLOSE(UNIT=4)
  755   WRITE(GNAME,735) MM,IGOUT
  735   FORMAT('EV',I1,'2',I4.4,'.DAT')
        OPEN(UNIT=3,FILE=GNAME,STATUS='OLD',FORM='UNFORMATTED',
     $        READONLY,ERR=770)
        WRITE(5,3939)
 3939   FORMAT(' READING EVENT FILE')
        READ(3) D,IDSUB,NUSED,GOOD
        CLOSE(UNIT=3)
        GOTO 777
C MUST CREATE EVENT PROBABILITIES
  770   CONTINUE
        WRITE(5,3938)
 3938   FORMAT(' CREATING EVENT FILE')
        CALL GENPIE(MM,NO,NSLOTS,GOOD,A,IASUB,C,ICSUB,
     $               D,IDSUB,NUSED)
        OPEN(UNIT=3,FILE=GNAME,STATUS='NEW',FORM='UNFORMATTED')
        WRITE(3) D,IDSUB,NUSED,GOOD
        CLOSE(UNIT=3)
  777   IF(.NOT.GOOD) GOTO 700
        DO 600 IJ=JJ,NJ
        WRITE(5,601) IJ
  601   FORMAT(' IJ=',I3)
        DEBNJ=START+(IJ-1)*DBINC
        DBSJR(IJ)=DEBNJ
        R=10.D0**(DEBNJ/10.D0)
        RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
        RHOT=K*RHOTS/2.D0
        RHOT2=RHOT*2.D0
        RHOT4=4.D0*RHOT
        RHOT8=RHOT*8
        RHOTH=RHOT/2.D0
        CALL PSUBE(MM,PESYM,D,IDSUB,NUSED,PRERR,IPSUB,PEOO)
        PE=WORBIT*PESYM
        WRITE(6,666) DBSJR(IJ),PE
  666   FORMAT(1X,F7.3,5X,1PD12.5)
        PRLOG(IJ)=DLOG10(PE)
        OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $        FORM='UNFORMATTED')
        WRITE(4) DBSJR(IJ), PRLOG(IJ)
        CLOSE(UNIT=4)
  600   CONTINUE
        OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
        WRITE(4) MM,2,DEBNOL(10),NSLOTS,GAMMA,DBSJR,PRLOG
        CLOSE(UNIT=4)
  700   CONTINUE
  800   CONTINUE
  900   CONTINUE
        STOP 0
        END
```



```
        SUBROUTINE GET(NJ,START,DBINC)
C
C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        CHARACTER*9 FIELD,BLANK9
        COMMON /INPUTS/ DEBNOL(3),NSLOTS,GAMLST(10),K,MM
        COMMON /SIZE/ NO,NG
C DEFAULT LISTS TEMPORARILY NEEDED ARE IN SHARED STORAGE WITH
C THE LARGE CONVOLUTION WORKING ARRAYS
        COMMON /SHARE/ DG(10),DSNR(3,4)
        DATA BLANK9/'         '/
   32   WRITE(5,33)
   33   FORMAT(' BITS/SYMBOL (K) [2]: ',$)
        READ(5,3)K
        IF(K.EQ.0)K=2
        MM=2**K
    1   WRITE(5,2)
    2   FORMAT(' HOW MANY EB/NO? [1]: ',$)
        READ(5,3)NO
    3   FORMAT(I2)
        IF(NO.EQ.0)NO=1
        DO 7 IN=1,NO
        DO=DSNR(IN,K)
    4   WRITE(5,5)IN,DO
    5   FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
        READ(5,6)FIELD
    6   FORMAT(A9)
        IF(FIELD.EQ.BLANK9) THEN
          DEBNOL(IN)=DO
        ELSE
          DECODE(9,61,FIELD)DEBNOL(IN)
   61     FORMAT(F9.6)
        END IF
    7   CONTINUE
C AUTOMATICALLY TAKE 2400 SLOTS
        NSLOTS=2400
   25   WRITE(5,26)
   26   FORMAT(' HOW MANY GAMMA? [10]: ',$)
        READ(5,3)NG
        IF(NG.EQ.0)NG=10
        DO 31 IN=1,NG
   28   WRITE(5,29)IN,DG(IN)
   29   FORMAT('   GAMMA(',I2,') [',1PD8.1,']: ',$)
        READ(5,30)GAMLST(IN)
   30   FORMAT(D15.8)
        IF(GAMLST(IN).EQ.0.D0)GAMLST(IN)=DG(IN)
   31   CONTINUE
   38   WRITE(5,39)
   39   FORMAT(' HOW MANY EB/NJ? [51]: ',$)
        READ(5,34,ERR=38) NJ
   34   FORMAT(I3)
```

```
      IF(NJ.EQ.0) NJ=51
      IF(NJ.LT.0 .OR. NJ.GT.51) GOTO 32
40    WRITE(5,41)
41    FORMAT(' STARTING VALUE FOR EB/NJ (DB) [0.]: ',$)
      READ(5,42,ERR=40) START
42    FORMAT(F6.3)
      IF(NJ.EQ.1) RETURN
35    WRITE(5,36)
36    FORMAT(' DB INCREMENT FOR EB/NJ [1.0]: ',$)
      READ(5,37,ERR=35) DBINC
37    FORMAT(F6.3)
      IF(DBINC.EQ.0.D0) DBINC=1.D0
      RETURN
      END
```



```
      SUBROUTINE PSUBE(M,PE,D,IDSUB,NUSED,PRERR,IPSUB,PE00)
C COMPUTE UNCONDITIONAL ERROR PROBABILITY
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INTEGER JAM(4),LUP(4),JSUB(4)
      LOGICAL*1 GO,NONE,STORE
      VIRTUAL PRERR(625),IPSUB(625)
      VIRTUAL D(625),IDSUB(625)
      COMMON /SHARE2/ LOW(4),LINC(4)
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
     $                RHON8, RHOT8, RHONH, RHOTH
      DATA STORE/.TRUE./
      PE=0.D0
      NPS=0
      DO 10 I=1,M
      LUP(I)=2
      JSUB(I)=0
10    CONTINUE
      CALL LOCN(M,LOW,LUP,JSUB,ISUB)
      CALL PUTIN(PE00,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
      IF(KODE.NE.0) STOP 'PRERR FULL'
      JAM1=-1
C START VECTOR-INDEXED LOOP ON JAMMING EVENTS
      CALL VLINIT(JAM,LOW,M)
100   CONTINUE
      IF(JAM1.NE.JAM(1)) THEN
        JAM1=JAM(1)
      END IF
      CALL EVENT(M,JAM,PIE,D,IDSUB,NUSED)
C BYPASS CONDITIONAL ERROR PROBABILITY CALCULATION IF EVENT
C PROBABILITY IS ZERO.  THIS SAVES MUCH TIME.
      IF(PIE.EQ.0.D0)GOTO 101
C SINCE JAMMING PROBABILITIES DEPEND ONLY ON NO. OF CHANNELS
C JAMMED AND NOT THE ARRANGEMENT OF THE CHANNELS, WE CAN SORT
C THE NON-SIGNAL CHANNELS INTO ASCENDING NUMBERS OF HOPS JAMMED.
      DO 111 I=1,M
      JSUB(I)=JAM(I)
111   CONTINUE
      IF(M.EQ.2) GOTO 199
      DO 110 I=2,M-1
      DO 120 J=I+1,M
      IF(JSUB(J).LT.JSUB(I)) THEN
        JTEMP=JSUB(I)
        JSUB(I)=JSUB(J)
        JSUB(J)=JTEMP
      END IF
120   CONTINUE
110   CONTINUE
199   CONTINUE
```

```
PDP-11 FORTRAN-77 V4.0-1        13:12:50     16-Jul-86            Page 7
CLIP2HOP.FTN;14          /F77/WR

        C STORE AS ELEMENT OF A SPARSE ARRAY TO SAVE MEMORY
        C EVEN THOUGH WE STORE ZEROS, THE SORTING OF SUBSCRIPTS
        C CUTS OUT MANY ELEMENTS.
0042            CALL LOCN(M,LOW,LUP,JSUB,ISUB)
        C TRY TO FIND CONDITIONAL ERROR PROBABILITY IN STORED ARRAY
0043            CALL LOOKUP(PROB,PRERR,IPSUB,NPS,625,ISUB,STORE,NONE)
        C IF IT IS NOT THERE, WE MUST COMPUTE IT
0044            IF(NONE) THEN
0045              CALL PSEL(JSUB,M,PROB)
        C  ... AND SAVE IT FOR POSSIBLE FUTURE RE-USE
0046              CALL PUTIN(PROB,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
0047              IF(KODE.NE.0) STOP 2
0048            END IF
        C SUM UP UNCONDITIONAL ERROR PROBABILITY
0049            PE=PE+PIE*PROB
        C ITERATE THE VECTOR-INDEX LOOP
0050    101     CALL VLITER(JAM,LOW,LUP,LINC,M,GO)
0051            IF(GO) GOTO 100
0052            RETURN
0053            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:12:56     16-Jul-86            Page 8
CLIP2HOP.FTN;14          /F77/WR

0001            SUBROUTINE EVENT(M,JAM,PIE,D,IDSUB,NUSED)
        C
        C SUBROUTINE TO LOOK UP EVENT PROBABILITY FROM STORED ARRAY
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            LOGICAL*1 STORE,NONE
0004            DIMENSION JAM(4),LUP(4)
0005            VIRTUAL D(625),IDSUB(625)
0006            COMMON /SHARE2/ LOW(4),LINC(4)
0007            DATA STORE/.FALSE./
        C SET UP ARRAY DESCRIPTION D(0:LL,...,0:LL) WITH M DIMENSIONS
0008            DO 1 I=1,M
0009            LUP(I)=2
0010    1       CONTINUE
        C COMPUTE LINEAR EQUIVALENT SUBSCRIPT FOR JAMMING EVENT
0011            CALL LOCN(M,LOW,LUP,JAM,ISUB)
        C LOOK UP THE VALUE, GET 0.D0 IF NOT THERE
0012            CALL LOOKUP(PIE,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
0013            RETURN
0014            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:12:58     16-Jul-86            Page 9
CLIP2HOP.FTN;14          /F77/WR

0001            SUBROUTINE GENPIE(MM,NQ,NSLOTS,GOOD,A,IASUB,C,ICSUB,
             $                    D,IDSUB,NUSED)
        C
        C SUBROUTINE TO GENERATE EVENT PROBABILITIES
        C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            LOGICAL*1 GO,GO2,STORE,NONE,GOOD
0004            DIMENSION LUP2(4),LUP3(4)
0005            DIMENSION IUPA(4)
0006            DIMENSION IUPD(4)
0007            DIMENSION LUP1(4)
0008            VIRTUAL A(100),IASUB(100),C(625),ICSUB(625),
             $          D(625),IDSUB(625)
0009            DIMENSION I(4),II(4),III(4)
        C SHARED STORAGE FOR COMMONLY NEEDED CONSTANT ARRAYS
0010            COMMON /SHARE2/ LOW(4),LINC(4)
        C SHARED STORATE FOR: (1) INPUT DEFAULT LISTS,
        C (2) CONDITIONAL PROB GEN., AND
        C (3) EVENT PROB. GEN.   THESE ARE NON-OVERLAPPING USAGES.
0011            COMMON /SHARE/ LUP2,LUP3,IUPD,L1,NUSEA,NUSEC,IERR,
             $                 ISUB,ISUB1,ISUB2,AIN,I,II,III,NN,AOUT,
             $                 CIN,COUT,DOUT,DIN
0012            DATA I100/100/
0013            DATA IUPA/4*1/
0014            DATA LUP1/4*1/
        C STORE=.FALSE. => DON'T STORE ZERO ELEMENTS OF SPARSE ARRAY.
0015            STORE=.FALSE.
0016            GOOD=.TRUE.
0017    9999    IF(NQ.LE.0) THEN
0018              GOOD=.FALSE.
0019              RETURN
0020            END IF
0021            DO 80 L1=1,MM
0022            IUPD(L1)=2
0023    80      CONTINUE
        C JAMMING PATTERN W/NON-ZERO PROBABILITY ON PER-HOP BASIS
0024            NUSEA=0
        C INITIALIZE VECTOR-INDEX LOOP
0025            CALL VLINIT(I,LOW,MM)
0026    90      CONTINUE
0027            CALL LOCN(MM,LOW,IUPA,1,ISUB)
0028            CALL PR1HOP(I,MM,NQ,NSLOTS,AIN)
0029            CALL PUTIN(AIN,A,IASUB,NUSEA,I100,ISUB,IERR,STORE)
0030            IF(IERR.NE.0)STOP 3
        C ITERATE VECTOR-INDEX LOOP
0031            CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
0032            IF(GO) GOTO 90
        C COMPUTATION STARTS HERE. FIRST COPY A INTO D.
        C SINCE ARRAYS ARE A(0:1,0:1,...,0:1) AND D(0:L,0:L,...,0:L)
        C THE COPYING MUST BE DONE ON BASIS OF EQUIVALENT LINEAR
        C SUBSCRIPTS RATHER THAN A SIMPLE MOVE OPERATION.
0033            NUSED=0
```

```
C INITIALIZE VECTOR-INDEX LOOP
        CALL VLINIT(I,LOW,MM)
99      CONTINUE
        CALL LOCN(MM,LOW,IUPA,I,ISUB1)
        CALL LOCN(MM,LOW,IUPD,I,ISUB2)
        CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
        CALL PUTIN(AOUT,D,IDSUB,NUSED,625,ISUB2,IERR,STORE)
C ITERATE VECTOR-INDEX LOOP
        CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
        IF(GO)GOTO 99
C ... L-1 CONVOLUTIONS ARE NEEDED ...
        DO 9998 L1=1,1
C SET UP VECTOR-LOOP UPPER LIMITS FOR THIS CONVOLUTION
9       DO 125 NN=1,MM
        LUP2(NN)=L1
125     LUP3(NN)=L1+1
        NUSEC=0
        CALL VLINIT(I,LOW,MM)
98      CONTINUE
        CALL VLINIT(II,LOW,MM)
97      CONTINUE
C LOOK UP ELEMENTS AND PERFORM ONE TERM OF THE CONVOLUTION
        CALL LOCN(MM,LOW,IUPA,I,ISUB1)
        CALL LOCN(MM,LOW,IUPD,II,ISUB2)
        DO 21 NN=1,MM
        III(NN)=I(NN)+II(NN)
21      CONTINUE
        CALL LOCN(MM,LOW,IUPD,III,ISUB3)
        CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
        CALL LOOKUP(DOUT,D,IDSUB,NUSED,625,ISUB2,STORE,NONE)
        CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB3,STORE,NONE)
        CIN=COUT+AOUT*DOUT
        CALL PUTIN(CIN,C,ICSUB,NUSEC,625,ISUB3,IERR,STORE)
        IF(IERR.NE.0) STOP 4
        CALL VLITER(II,LOW,LUP2,LINC,MM,GO2)
        IF(GO2) GOTO 97
        CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
        IF(GO) GOTO 98
C COPY C TO D IN SORTED ORDER FOR NEXT ITERATION
        NUSED=0
        CALL VLINIT(II,LOW,MM)
96      CONTINUE
        CALL LOCN(MM,LOW,IUPD,II,ISUB)
        CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB,STORE,NONE)
        DIN=COUT
        CALL PUTIN(DIN,D,IDSUB,NUSED,625,ISUB,IERR,STORE)
        IF(IERR.NE.0) STOP 5
        CALL VLITER(II,LOW,LUP3,LINC,MM,GO)
        IF(GO) GOTO 96
9998    CONTINUE
        RETURN
        END
```



```
        SUBROUTINE PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C
C THIS SUBROUTINE INSERTS AN ELEMENT INTO A SPARSE ARRAY FOR
C WHICH ONLY THE NON-ZERO ELEMENTS ARE KEPT IN STORAGE IF
C THE SWITCH STORE IS .TRUE.
C
C THE DOUBLE PRECISION VALUE CIN IS STORED AS C(k) WHERE
C THE AUXILIARY ARRAY ICSUB IS USED TO KEEP TRACK OF THE
C SUBSCRIPTS OF THE CORRESPONDING ENTRIES IN C.
C
C USAGE:
C       LOGICAL*1 STORE
C       DOUBLE PRECISION C,CIN
C       VIRTUAL ICSUB(NMAX),C(NMAX)
C       CALL PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
C WHERE
C  CIN = VALUE OF ELEMENT TO STORE
C  C = ARRAY IN WHICH NON-ZERO VALUES ARE ACTUALLY STORED
C  ICSUB = AUXILIARY ARRAY FOR ACTUAL SUBSCRIPT VALUES
C  NUSE = NUMBER OF ELEMENTS OF C CURRENTLY OCCUPIED
C  NMAX = SIZE OF ARRAY C
C  IERR = ERROR RETURN CODE, 0 IF NO ERROR OR 1 IF THERE IS
C       NO ROOM AVAILABLE IN C
C  STORE = .TRUE. TO STORE ZEROES EXPLICITLY, ELSE .FALSE.
C NOTE: IF CIN=0 AND THE SUBSCRIPT K IS FOUND IN ICSUB, THEN
C       THE ELEMENT IS DELETED BY SHIFTING DOWNWARD ALL
C       FOLLOWING ELEMENTS OF THE ARRAY
C
C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        VIRTUAL ICSUB(NMAX),C(NMAX)
        LOGICAL*1 STORE
        IERR=0
        IF(STORE) GOTO 5
        IF(CIN.EQ.0.D0) GOTO 30
5       IF(NUSE.EQ.0)GOTO 20
        DO 10 I=1,NUSE
        IF(ICSUB(I).NE.K) GOTO 10
        C(I)=CIN
        RETURN
10      CONTINUE
        IF(NUSE.LT.NMAX) GOTO 20
        IERR=1
        RETURN
20      NUSE=NUSE+1
        ICSUB(NUSE)=K
        C(NUSE)=CIN
        RETURN
```

```
30      DO 40 I=1,NUSE
        J=I
        IF(ICSUB(I).EQ.K) GOTO 50
40      CONTINUE
        RETURN
C
C REMOVE THE ZEROED ELEMENT AND BUMP COUNT OF ENTRIES USED
C
50      DO 60 I=J,NUSE-1
        ICSUB(I)=ICSUB(I+1)
        C(I)=C(I+1)
60      CONTINUE
        NUSE=NUSE-1
        RETURN
        END
```



```
        SUBROUTINE LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
C
C THIS SUBROUTINE RETRIEVES AN ELEMENT OF A SPARSE ARRAY WHICH
C HAS BEEN STORED COMPACTLY BY STORING ONLY NON-ZERO ELEMENTS.
C
C THE ARRAY IS DOUBLE PRECISION.
C
C USAGE:
C       VIRTUAL ICSUB(NMAX), C(NMAX)
C       LOGICAL*1 STORE, NONE
C       DOUBLE PRECISION COUT
C       CALL LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
C WHERE
C       COUT = VALUE OF C(K) (OUTPUT FROM SUBROUTINE)
C          C = ARRAY USED TO STORE NON-ZERO ELEMENTS
C      ICSUB = AUXILIARY ARRAY TO STORE ACTUAL SUBSCRIPTS
C          N = NUMBER OF ELEMENTS OF C CURRENTLY IN USE
C       NMAX = SIZE OF C
C          K = SUBSCRIPT OF SPARSE ARRAY TO LOOK UP
C      STORE = .TRUE. IF ZEROES STORED EXPLICITLY, ELSE .FALSE.
C       NONE = .FALSE. IF ZEROES NOT STORED OR ZEROES STORED AND
C                      ELEMENT IS FOUND IN THE STORED ARRAY
C              .TRUE.  IF ZEROES ARE STORED AND THE ELEMENT IS
C              NOT FOUND (OUTPUT QUANTITY)
C
C PROGRAMMER: ROBERT H. FRENCH
C       DATE: 11 JANUARY 1984
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        VIRTUAL ICSUB(NMAX),C(NMAX)
        LOGICAL*1 STORE, NONE
        NONE=.FALSE.
        DO 10 I=1,N
        IF(ICSUB(I).NE.K)GOTO 10
        COUT=C(I)
        RETURN
10      CONTINUE
        IF(STORE) THEN
          NONE=.TRUE.
        ELSE
          COUT=0.
        END IF
        RETURN
        END
```

```
0001            SUBROUTINE LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C
        C THIS SUBROUTINE COMPUTES THE EQUIVALENT LINEAR SUBSCRIPT FOR
        C A MULTIDIMENSIONAL ARRAY OF NDIM DIMENSIONS
        C
        C IF THE ARRAY A IS DEFINED AS
        C       DIMENSION A(ILOW(1):IUP(1),...,ILOW(NDIM):IUP(NDIM))
        C AND ISUB(1),...,ISUB(NDIM) IS A SET OF SUBSCRIPTS FOR A,
        C THEN THIS SUBROUTINE RETURNS IN LINEAR THE OFFSET FROM THE
        C ORIGIN OF A TO THE ELEMENT A(ISUB(1),...,ISUB(NDIM)), ASSUMING
        C THE FIRST SUBSCRIPT VARIES MOST RAPIDLY.
        C
        C USAGE:
        C       DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
        C       DATA ILOW/lower limits of defined subscripts of array/
        C       DATA IUP/upper limits of defined subscripts of array/
        C       ...SET ISUB TO DESIRED SEQUENCE OF SUBSCRIPTS...
        C       CALL LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C WHERE
        C       NDIM = NUMBER OF DIMENSIONS THE ARRAY HAS
        C       ILOW = ARRAY OF LOWER SUBSCRIPT BOUNDS
        C        IUP = ARRAY OF UPPER SUBSCRIPT BOUNDS
        C       ISUB = ARRAY CONTAINING SUBSCRIPT FOR WHICH LOCATION IS
        C              TO BE COMPUTED
        C     LINEAR = RETURNED VALUE OF OFFSET INTO ARRAY IN MEMORY
        C
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 11 JANUARY 1984
        C
0002            DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
0003            LINEAR=0
0004            DO 10 I=1,NDIM-1
0005            J=NDIM-I+1
0006            LINEAR=(LINEAR+(ISUB(J)-ILOW(J)))*(IUP(J-1)-ILOW(J-1)+1)
0007    10      CONTINUE
0008            LINEAR=LINEAR+ISUB(1)-ILOW(1)
0009            RETURN
0010            END
```



```
0001            SUBROUTINE VLINIT(LVEC,LLOW,LMAX)
        C
        C THIS SUBROUTINE INITIALIZES A "VECTOR DO-LOOP" STRUCTURE
        C DEFINED BY THE FOLLOWING PSEUDO-FORTRAN CODE:
        C       DO 100 LVEC(1)=LLOW(1),LUP(1),LINC(1)
        C       DO 100 LVEC(2)=LLOW(2),LUP(2),LINC(2)
        C                   :
        C                   :
        C       DO 100 LVEC(LMAX)=LLOW(LMAX),LUP(LMAX),LINC(LMAX)
        C                   :
        C                   :
        C       (STATEMENTS IN RANGE OF LOOP)
        C 100   CONTINUE
        C
        C THE COMPANION ROUTINE VLITER HANDLES THE LOOP CONTROL AT THE
        C CONTINUE STATEMENT IN THE ABOVE STRUCTURE
        C
        C USAGE:
        C       LOGICAL*1 GO
        C       DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
        C (INITIALIZE ARRAY LLOW TO STARTING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE ARRAY LUP TO STOPING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE ARRAY LINC TO INCREMENTS OF THE LOOPS)
        C       CALL VLINIT(LVEC,LLOW,LMAX)
        C   100 CONTINUE
        C         .
        C         . (STATEMENTS IN RANGE OF LOOPS)
        C         .
        C       CALL VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
        C       IF(GO)GOTO 100
        C WHERE
        C    LVEC = ARRAY FOR STORAGE OF LOOP INDICES. LVEC(1) IS THE
        C           OUTER-MOST LOOP; LVEC(LMAX), THE INNER-MOST LOOP.
        C    LLOW = ARRAY FOR STORAGE OF LOOP STARTING VALUES, IN SAME
        C           SEQUENCE AS LVEC
        C    LUP = ARRAY FOR STORAGE OF LOOP FINAL VALUES, IN SAME
        C          SEQUENCE AS LVEC
        C    LINC = ARRAY FOR STORAGE OF LOOP INCREMENTS, IN SAME
        C           SEQUENCE AS LVEC
        C    LMAX = NUMBER OF LOOPS NESTED
        C    GO = LOGICAL VARIABLE, .TRUE. IF JUMP BACK TO BEGINNING OF
        C         STATEMENTS IN THE RANGE OF THE LOOP SHOULD OCCUR,
        C         .FALSE. OTHERWISE (I.E. OUTER-MOST LOOP TERMINATED)
        C
        C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
        C
0002            DIMENSION LVEC(LMAX),LLOW(LMAX)
0003            DO 1 N=1,LMAX
0004            LVEC(N)=LLOW(N)
0005    1       CONTINUE
0006            RETURN
0007            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:16    16-Jul-86            Page 16
CLIP2HOP.FTN;14          /F77/WR

            SUBROUTINE VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
      C
      C LOOP ITERATION LOGIC FOR A "VECTOR DO-LOOP"
      C
      C SEE DETAILED COMMENTS IN SUBROUTINE VLINIT FOR USAGE AND
      C PARAMETER DEFINITIONS
      C
      C PROGRAMMER: ROBERT H. FRENCH
      C       DATE: 11 JANUARY 1984
      C
            LOGICAL*1 GO
            DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
            GO=.TRUE.
            DO 100 NDX=1,LMAX
            NSUB=LMAX+1-NDX
            LVEC(NSUB)=LVEC(NSUB)+LINC(NSUB)
            IF((LINC(NSUB).GE.0.AND.LVEC(NSUB).LE.LUP(NSUB))
         $    .OR.(LINC(NSUB).LT.0.AND.LVEC(NSUB).GE.LUP(NSUB))) RETURN
            LVEC(NSUB)=LLOW(NSUB)
      100   CONTINUE
            GO=.FALSE.
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:18    16-Jul-86            Page 17
CLIP2HOP.FTN;14          /F77/WR

            SUBROUTINE PRIHOP(I,KM,KQ,KN,AIN)
      C
      C THIS SUBROUTINE COMPUTES THE EVENT PROBABILITY FOR ALL
      C POSSIBLE JAMMING PATTERNS WITH NON-ZERO PROBABILITY FOR
      C L=1 HOP/SYMBOL FOR RMFSK/FH IN PBNJ
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            DIMENSION I(4)
            AIN=0.D0
            KJAM=0
            DO 1 K=1,KM
            KJAM=KJAM+I(K)
        1   CONTINUE
      C IF THIS IS AN IMPOSSIBLE CASE, RETURN WITH RESULT = 0.0
            IF(KJAM.GT.MIN0(KQ,KM)) RETURN
            KPMAX=KJAM-1
            LPMAX=KM-KJAM-1
            JPMAX=KM-1
            IMAX=MAX0(KPMAX,LPMAX,JPMAX)
            PROD=1.D0
            Q=KQ
            DIFFNQ=KN-KQ
            EN=KN
            DO 100 LOOP=0,IMAX
            F=LOOP
            IF(LOOP.LE.KPMAX) PROD=PROD*(Q-F)
            IF(LOOP.LE.JPMAX) PROD=PROD/(EN-F)
            IF(LOOP.LE.LPMAX) PROD=PROD*(DIFFNQ-F)
      100   CONTINUE
            AIN=PROD
            RETURN
            END
```

```
0001          BLOCK DATA
        C INITIALIZE SHARED CONSTANTS
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          COMMON /SHARE/ DG(10),DSNR(3,4)
0004          COMMON /SHARE2/ LOW(4),LINC(4)
0005          COMMON /WTS/ X(8),W(8)
        C
        C WEIGHTS AND ABSCISSAS FOR 16-POINT GAUSSIAN QUADRATURE
        C
0006          DATA X/ 0.09501250983763744018500,
     $                0.28160355077925891323000,
     $                0.45801677765722738634200,
     $                0.61787624440264374844700,
     $                0.75540440835500303389500,
     $                0.86563120238783174388000,
     $                0.94457502307323257607800,
     $                0.98940093499164993259600 /
0007          DATA W/ 0.18945061045506849628500,
     $                0.18260341504492358886700,
     $                0.16915651939500253818900,
     $                0.14959598881657673208100,
     $                0.12462897125553387205200,
     $                0.09515851168249278481000,
     $                0.06225352393864789286300,
     $                0.02715245941175409485200 /
        C DEFAULT LISTS FOR INTERACTIVE PARAMETER INPUTS
        C ARE SHARED WITH LARGE WORKING STORAGE ARRAYS SINCE THEY
        C MAY BE DESTROYED ONCE THE INPUT PARAMETERS ARE SET UP
0008          DATA DG / .001D0, .002D0, .005D0,
     $                 .01D0,  .02D0,  .05D0,
     $                 .1D0,   .2D0,   .5D0,
     $                1.D0 /
0009          DATA DSNR /13.35247D0, 12.3133D0, 10.94443D0,
     $                 10.606572D0, 9.6284D0,  8.35248D0,
     $                  9.094010D0, 8.16900D0, 6.971995D0,
     $                  8.07835D0,  7.1996D0,  6.069646D0/
        C FREQUENTLY NEEDED CONSTANT ARRAYS AND SCALARS
0010          DATA LOW/4*0/,LINC/4*1/
0011          END
```



```
0001          SUBROUTINE PSEL(JSUB,MM,PROB)
        C
        C RANDOM MFSK/FH IN PARTIAL BAND NOISE JAMMING,
        C GIVEN A JAMMING EVENT, WITH CLIPPER RECEIVER
        C
        C        JSUB - JAMMING EVENT VECTOR
        C          MM - ALPHABET SIZE
        C        PROB - RESULTING CONDITIONAL ERROR PROBABILITY
        C
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          DIMENSION JSUB(MM), WORK(30), STACK(30), HEAP(30)
0004          EXTERNAL DG16, PGRAND
0005          INTEGER NCHAN(0:3)
0006          COMMON /JAMCNT/ NCHAN
0007          COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                        TAU, TAU2, TAUK, TAUK2
0008          COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
     $                        RHON8, RHOT8, RHONH, RHOTH
0009          COMMON /QUES/ Q0, Q1
0010          COMMON /SCJAM/ JAMSC
0011          KSUB=0
0012          DO 6 I=1,MM
0013          KSUB=KSUB+JSUB(I)
0014        6 CONTINUE
        C SET UP VALUES WHICH WILL REMAIN IF THIS IS THE NOTHING-JAMMED CASE
0015          Q0=Q(2.D0*DSQRT(RHONH),DSQRT(2.D0*TAU))
0016          Q1=1.D0
0017          TAUK=0.D0
        C IF ANYTHING IS JAMMED, SET UP JAMMING-RELATED QUANTITIES
0018          IF(KSUB.NE.0) THEN
0019            BIGK=RHON/RHOT
0020            BK1=BIGK-1.D0
0021            AAB=BIGK/BK1
0022            BAB=1.D0/BK1
0023            TAUK=TAU/BIGK
0024            TAUK2=TAU2/BIGK
0025            Q1=Q(2.D0*DSQRT(RHOTH),DSQRT(2.D0*TAUK))
0026          END IF
        C
        C COUNT NUMBER OF NONSIGNAL CHANNELS WITH Lm HOPS JAMMED
        C
0027          DO 10 I=0,2
0028          NCHAN(I)=0
0029       10 CONTINUE
0030          DO 11 I=2,MM
0031          KSUB=JSUB(I)
0032          NCHAN(KSUB)=NCHAN(KSUB)+1
0033       11 CONTINUE
0034          JAMSC=JSUB(1)
        C
        C DO THE CONTINUOUS PART OF THE DENSITY IN SECTIONS
        C
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:23    16-Jul-86            Page 20
CLIP2HOP.FTN;14          /F77/WR

            PCONT=0.DO
            DO 13 ISECT=1,2
            XL=(ISECT-1)*TAU
            XU=ISECT*TAU
            CALL ADQUAD(XL,XU,CHUNK,DG16,PGRAND,1.D-8,WORK,
     $                  STACK,HEAP,30,KODE)
            CALL TEST2(KODE,10)
            PCONT=PCONT+CHUNK
   13       CONTINUE
C
C DO THE TIE PART OF THE DENSITY
C
            CALL TIES(JSUB,MM,PTIE)
C
C PUT THEM TOGETHER
C
            PROB=1.DO-PCONT-PTIE
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:27    16-Jul-86            Page 21
CLIP2HOP.FTN;14          /F77/WR

            DOUBLE PRECISION FUNCTION PGRAND(BETA)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            INTEGER NCHAN(0:3)
            COMMON /SCJAM/ JAMSC
            COMMON /JAMCNT/ NCHAN
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                      TAU, TAU2, TAUK, TAUK2
            COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
     $                      RHON8, RHOT8, RHONH, RHOTH
            COMMON /QUES/ QO, Q1
            PROD=1.DO
            DO 10 I=0,2
            IF(NCHAN(I).NE.0) THEN
              LJAM=I
              X=GL(BETA)
              PROD=PROD*DXI(X,NCHAN(I))
            END IF
   10       CONTINUE
            LJAM=JAMSC
            Y=PZ1(BETA)
            PGRAND=Y*PROD
            RETURN
            END
```



```
PDP-11 FORTRAN-77 V4.0-1        13:13:30    16-Jul-86            Page 22
CLIP2HOP.FTN;14          /F77/WR

            SUBROUTINE ADQUAD(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C              WITH CALLING SEQUENCE
C              CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C        SAME ARRAY AS WORK (IN)
C  HEAP- THIRD WORK ARRAY OF SIZE N, DISTINCT FROM BOTH WORK AND STACK
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            EXTERNAL F
            DIMENSION WORK(N),STACK(N),HEAP(N)
            KODE=0
            Y=0.DO
            WORK(1)=XU
            CALL QR(XL,XU,F,T)
            HEAP(1)=T
            A=XL
            NPTS=1
            EPS=TOL
            STACK(1)=EPS
   10       B=WORK(NPTS)
            XM=(A+B)*0.5DO
            CALL QR(A,XM,F,P1)
            CALL QR(XM,B,F,P2)
            IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
            NPTS=NPTS+1
            IF(NPTS.GT.N) THEN
              KODE=1
              RETURN
            END IF
            WORK(NPTS)=XM
            HEAP(NPTS)=P2
            T=P1
            EPS=EPS/2.DO
```

```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```



```
      SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C              WITH CALLING SEQUENCE
C              CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C        SAME ARRAY AS WORK (IN)
C  HEAP- THIRD WORK ARRAY OF SIZE N, DISTINCT FROM BOTH WORK AND STACK
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=EPS/2.D0
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:34    16-Jul-86          Page 25
CLIP2HOP.FTN;14          /F77/WR

            IF(EPS.EQ.0.D0) THEN
              KODE=2
              RETURN
            END IF
            STACK(NPTS)=EPS
            GOTO 10
      C FINISHED A PIECE
      20    Y=Y+P1+P2
            EPS=STACK(NPTS)
            T=HEAP(NPTS)
            NPTS=NPTS-1
            A=B
            IF(NPTS.EQ.0) RETURN
            GOTO 10
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:37    16-Jul-86          Page 26
CLIP2HOP.FTN;14          /F77/WR

            SUBROUTINE DG16(A,B,F,ANSWER)
      C
      C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
      C
      C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
      C
      C R. H. FRENCH, 28 FEBRUARY 1986
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON /WTS/ X(8),W(8)
            ANSWER=0.D0
            BMAO2=(B-A)/2.D0
            BPAO2=(B+A)/2.D0
            DO 10 I=1,8
            C=X(I)*BMAO2
            Y1=BPAO2+C
            Y2=BPAO2-C
            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
      10    CONTINUE
            ANSWER=ANSWER*BMAO2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:38    16-Jul-86          Page 27
CLIP2HOP.FTN;14          /F77/WR

            SUBROUTINE DGXVI(A,B,F,ANSWER)
      C
      C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
      C
      C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
      C
      C R. H. FRENCH, 28 FEBRUARY 1986
      C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON /WTS/ X(8),W(8)
            ANSWER=0.D0
            BMAO2=(B-A)/2.D0
            BPAO2=(B+A)/2.D0
            DO 10 I=1,8
            C=X(I)*BMAO2
            Y1=BPAO2+C
            Y2=BPAO2-C
            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
      10    CONTINUE
            ANSWER=ANSWER*BMAO2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:40    16-Jul-86            Page 28
CLIP2HOP.FTN;14          /F77/WR

0001            SUBROUTINE TEST(ID)
        C
        C TEST RETURN CODE FROM BESSEL FUNCTION
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            IF(KODE.EQ.0) RETURN
0005            WRITE(5,1) KODE, ID
0006        1   FORMAT(' BESSEL FUNCTION CODE = ',I2,' FROM CALL NUMBER ',I5)
0007            STOP 'FATAL ERROR'
0008            END
```

```
PDP-11 FORTRAN-77 V4.0-1        13:13:42    16-Jul-86            Page 29
CLIP2HOP.FTN;14          /F77/WR

0001            SUBROUTINE TEST2(KODE,ID)
        C
        C TEST RETURN CODE FROM ADQUAD/ADQUA2
        C
0002            IF(KODE.EQ.0) RETURN
0003            WRITE(5,1) KODE, ID
0004        1   FORMAT(' ADAPTIVE INTEGRATOR CODE = ',I2,' FROM CALL NUMBER ',I5)
0005            STOP 'FATAL ERROR'
0006            END
```



```
PDP-11 FORTRAN-77 V4.0-1        13:13:43    16-Jul-86            Page 30
CLIP2HOP.FTN;14          /F77/WR

0001            DOUBLE PRECISION FUNCTION PZ1(Y)
        C
        C       SIGNAL CHANNEL P.D.F. WITH CHANGE OF VARIABLE Y=AX
        C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            DIMENSION WORK(30), STACK(30), HEAP(30)
0004            LOGICAL*1 REG1, REG2
0005            EXTERNAL DGXVI, F20, F21, F22
0006            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0007            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
            $                   TAU, TAU2, TAUK, TAUK2
0008            COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
            $                   RHON8, RHOT8, RHONH, RHOTH
0009            COMMON /QUES/ Q0, Q1
0010            COMMON /XCOM/ XCON
0011            COMMON /OUTER/ XXX, XXXK
0012            XXX=Y
0013            IF(LJAM.GE.1) THEN
0014              XXXK=Y/BIGK
0015              YK=XXXK
0016              YTK=(Y-TAU)/BIGK
0017              YTK2=(Y-TAU2)/BIGK
0018            END IF
0019            REG1=Y.GE.0.D0 .AND. Y.LT.TAU
0020            REG2=Y.GE.TAU .AND. Y.LT.TAU2
        C
        C       TWO HOPS PER SYMBOL
        C
0021     2000   GOTO (2100, 2200, 2300), LJAM+1
        C
        C               NO HOPS JAMMED
        C
0022     2100   IF(REG1) THEN
0023              BARG1=DSQRT(RHON8*Y)
0024              IO1=1
0025              CALL DXBT(2100)
0026              PZ1=0.5D0*DSQRT(Y/RHONH)*DEXP(BARG1-Y-RHON2)*B1
0027            ELSE IF(REG2) THEN
0028              BARG1=DSQRT(RHON4*(Y-TAU))
0029              IO1=0
0030              CALL DXBT(2101)
0031              PART=2.D0*Q0*DEXP(BARG1-Y+TAU-RHON)*B1
0032              XCON=Y+RHON2
0033              CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F20,1.D-9,WORK,STACK,
            $                 HEAP,30,KOD)
0034              CALL TEST2(KOD,2100)
0035              PZ1=PART+ANSWER
0036            ELSE
0037              PZ1=0.D0
0038            END IF
0039            GOTO 9000
```

```
      C
      C             ONE HOP JAMMED
      C
        2200  IF(REG1) THEN
                XCON=YK+RHON+RHOT
                CALL ADQUA2(0.DO,Y,ANSWER,DGXVI,F21,1.D-9,WORK,STACK,
           $                HEAP,30,KOD)
                CALL TEST2(KOD,2200)
                PZ1=ANSWER/BIGK
              ELSE IF(REG2) THEN
                BARG1=DSQRT(RHOT4*YTK)
                IO1=0
                CALL DXBT(2201)
                PART=Q0*DEXP(BARG1-YK+TAUK-RHOT)*B1/BIGK
                BARG1=DSQRT(RHON4*(Y-TAU))
                IO1=0
                CALL DXBT(2202)
                PART=PART+Q1*DEXP(BARG1-Y+TAU-RHON)*B1
                XCON=YK+RHON+RHOT
                CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F21,1.D-9,WORK,STACK,
           $                HEAP,30,KOD)
                CALL TEST2(KOD,2202)
                PZ1=PART+ANSWER/BIGK
              ELSE
                PZ1=0.DO
              END IF
              GOTO 9000
      C
      C             TWO HOPS JAMMED
      C
        2300  IF(REG1) THEN
                BARG1=DSQRT(RHOT8*YK)
                IO1=1
                CALL DXBT(2300)
                PZ1=0.5DO*DSQRT(Y/RHONH)*DEXP(BARG1-YK-RHOT2)
           $             *B1/BIGK
              ELSE IF(REG2) THEN
                BARG1=DSQRT(RHOT4*YTK)
                IO1=0
                CALL DXBT(2301)
                PART=2.DO*Q1*DEXP(BARG1-YTK-RHOT)*B1/BIGK
                XCON=YK+RHOT2
                CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F22,1.D-9,WORK,STACK,
           $                HEAP,30,KOD)
                CALL TEST2(KOD,2300)
                PZ1=PART+ANSWER/(BIGK*BIGK)
              ELSE
                PZ1=0.DO
              END IF
        9000  CONTINUE
              RETURN
              END
```



```
              DOUBLE PRECISION FUNCTION GL(Y)
      C
      C  NON-SIGNAL CHANNEL CUMULATIVE DISTRIBUTION FUNCTION
      C       WITH CHANGE OF VARIABLE Y=AX
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              LOGICAL*1 REG1, REG2, REG3
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
           $                  TAU, TAU2, TAUK, TAUK2
              REG1=Y.GE.0.DO .AND. Y.LT.TAU
              REG2=Y.GE.TAU  .AND. Y.LT.TAU2
              IF(LJAM.GT.0) THEN
                YK=Y/BIGK
                YTK=(Y-TAU)/BIGK
                YTK2=(Y-TAU2)/BIGK
              END IF
      C
      C       TWO HOPS PER SYMBOL
      C
        2000  GOTO (2100, 2200, 2300), LJAM+1
      C
      C             NO HOPS JAMMED
      C
        2100  IF(REG1) THEN
                GL=1.DO-(1.DO+Y)*DEXP(-Y)
              ELSE IF(REG2) THEN
                GL=1.DO-(1.DO+TAU2-Y)*DEXP(-Y)
              ELSE IF(Y.GE.TAU2) THEN
                GL=1.DO
              ELSE
                GL=0.DO
              END IF
              GOTO 9000
      C
      C             ONE HOP JAMMED
      C
        2200  IF(REG1) THEN
                GL=1.DO-AAB*DEXP(-YK)+BAB*DEXP(-Y)
              ELSE IF(REG2) THEN
                GL=1.DO-AAB*DEXP(TAU-TAUK-Y)+BAB*DEXP(TAUK-TAU-YK)
              ELSE IF(Y.GE.TAU2) THEN
                GL=1.DO
              ELSE
                GL=0.DO
              END IF
              GOTO 9000
```

```
PDP-11 FORTRAN-77 V4.0-1          13:13:50    16-Jul-86            Page 33
CLIP2HOP.FTN;14           /F77/WR

        C
        C              TWO HOPS JAMMED
        C
0033     2300    IF(REG1) THEN
0034               GL=1.DO-(1.DO+YK)*DEXP(-YK)
0035             ELSE IF(REG2) THEN
0036               GL=1.DO-(1.DO+TAUK2-YK)*DEXP(-YK)
0037             ELSE IF(Y.GE.TAU2) THEN
0038               GL=1.DO
0039             ELSE
0040               GL=0.DO
0041             END IF
0042             GOTO 9000
0043     9000    CONTINUE
0044             RETURN
0045             END

PDP-11 FORTRAN-77 V4.0-1          13:13:55    16-Jul-86            Page 34
CLIP2HOP.FTN;14           /F77/WR

0001             DOUBLE PRECISION FUNCTION F20(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=0
        C
0002             IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003             COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
            $                    TAU, TAU2, TAUK, TAUK2
0004             COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
            $                    RHON8, RHOT8, RHONH, RHOTH
0005             COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0006             COMMON /QUES/ Q0, Q1
0007             COMMON /XCOM/ XCON
0008             COMMON /OUTER/ XXX, XXXK
0009             BARG1=DSQRT(RHON4*U)
0010             BARG2=DSQRT(RHON4*(XXX-U))
0011             I01=0
0012             I02=0
0013             CALL BPROD(2110)
0014             F20=DEXP(BARG1+BARG2-XCON)*B1*B2
0015             RETURN
0016             END

PDP-11 FORTRAN-77 V4.0-1          13:13:56    16-Jul-86            Page 35
CLIP2HOP.FTN;14           /F77/WR

0001             DOUBLE PRECISION FUNCTION F21(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=1
        C
0002             IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003             COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0004             COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
            $                    TAU, TAU2, TAUK, TAUK2
0005             COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
            $                    RHON8, RHOT8, RHONH, RHOTH
0006             COMMON /QUES/ Q0, Q1
0007             COMMON /XCOM/ XCON
0008             COMMON /OUTER/ XXX, XXXK
0009             BARG1=DSQRT(RHON4*U)
0010             BARG2=DSQRT(RHOT4*(XXXK-U/BIGK))
0011             I01=0
0012             I02=0
0013             CALL BPROD(2210)
0014             F21=DEXP(BARG1+BARG2-XCON-U+U/BIGK)*B1*B2
0015             RETURN
0016             END

PDP-11 FORTRAN-77 V4.0-1          13:13:58    16-Jul-86            Page 36
CLIP2HOP.FTN;14           /F77/WR

0001             SUBROUTINE BPROD(IDENT)
        C
        C COMPUTE TWO BESSEL FUNCTIONS, ARGUMENTS AND RESULTS IN COMMON
        C
0002             IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003             COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0004             CALL DXBT(IDENT)
0005             CALL DXBESI(BARG2,I02,B2,KODE)
0006             CALL TEST(IDENT+1)
0007             RETURN
0008             END
```

```
            DOUBLE PRECISION FUNCTION F22(U)
      C
      C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=2
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
           $                TAU, TAU2, TAUK, TAUK2
            COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
           $                RHON8, RHOT8, RHONH, RHOTH
            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
            COMMON /QUES/ Q0, Q1
            COMMON /XCOM/ XCON
            COMMON /OUTER/ XXX, XXXK
            BARG1=DSQRT(RHOT4*U/BIGK)
            BARG2=DSQRT(RHOT4*(XXXK-U/BIGK))
            I01=0
            I02=0
            CALL BPROD(2310)
            F22=DEXP(BARG1+BARG2-XCON)*B1*B2
            RETURN
            END
```



```
            SUBROUTINE DXBT(ID)
      C
      C CALL DXBESI AND TEST RETURN CODE
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
            CALL DXBESI(BARG1,I01,B1,KODE)
            CALL TEST(ID)
            RETURN
            END
```



```
            SUBROUTINE TIES(JSUB,MM,PTIE)
      C
      C  COMPUTE PROBABILITY OF CORRECT DECISION GIVEN THAT
      C  SEVERAL SATURATED CHANNELS ARE TIED
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            DIMENSION JSUB(MM), LLOW(7), LINC(7), LUP(7), NU(7),
           $          P2LM(2:8)
            LOGICAL*1 GO
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
           $                TAU, TAU2, TAUK, TAUK2
            COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
           $                RHON8, RHOT8, RHONH, RHOTH
            COMMON /QUES/ Q0, Q1
      C NUMBER OF NON-SIGNAL CHANNELS
            MNU=MM-1
            PTIE=0.D0
            CUE0=DEXP(-TAU)
            CUE1=DEXP(-TAUK)
            P1L=DXI(Q0,2-JSUB(1))*DXI(Q1,JSUB(1))
            DO 10 I=2,MM
            P2LM(I)=DXI(CUE0,2-JSUB(I))*DXI(CUE1,JSUB(I))
         10 CONTINUE
      C SET UP VECTOR LOOP PARAMETERS
            DO 20 I=1,MM-1
            LLOW(I)=0
            LINC(I)=1
            LUP(I)=1
         20 CONTINUE
            PTIE=0.D0
      C START LOOP ON THE TIE EVENTS
            CALL VLINIT(NU,LLOW,MM-1)
         30 NUSUM=0
            DO 40 I=1,MM-1
            NUSUM=NUSUM+NU(I)
         40 CONTINUE
            FRAC=1.D0/(1.D0+NUSUM)
            PROD=1.D0
            DO 50 M=2,MM
            IF(NU(M-1).EQ.1) THEN
              PROD=PROD*P2LM(M)
            ELSE
              PROD=PROD*(1.D0-P2LM(M))
            END IF
         50 CONTINUE
            PTIE=FRAC*PROD+PTIE
            CALL VLITER(NU,LLOW,LUP,LINC,MM-1,GO)
            IF(GO) GOTO 30
            PTIE=PTIE*P1L
            RETURN
            END
```

```
0001              DOUBLE PRECISION FUNCTION PUNJAM(ETA)
          C
          C FUNCTION FOR UNJAMMED P(E) FOR OPT. THRESHOLD SEARCH
          C
          C       NOTE: WHEN JAMMING EVENT IS (0,0,...,0), THE VARIABLES
          C             BIGK, AAB, BAB, TAUK, AND TAUK2 ARE NOT
          C             USED IN THE COMPUTATIONS, AND HENCE DO NOT NEED
          C             TO BE SET UP BEFORE CALLING PSEL FROM THIS FUNCTION
          C
0002              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003              DIMENSION NOJAM(4)
0004              COMMON /INPUTS/ DEBNOL(3),NSLOTS,GAMLST(10),K,MM
0005              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
              $                   TAU, TAU2, TAUK, TAUK2
0006              DATA NOJAM/0,0,0,0/
0007              LJAM=0
0008              TAU=ETA
0009              TAU2=TAU+ETA
0010              CALL PSEL(NOJAM,MM,P)
0011              PUNJAM=P
0012              RETURN
0013              END
```



```
0001              SUBROUTINE SETTAU(MM,PEOO)
          C
          C  SEARCH FOR OPTIMUM THRESHOLD IN ABSENCE OF JAMMING
          C
0002              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003              EXTERNAL PUNJAM
0004              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
              $                   TAU, TAU2, TAUK, TAUK2
0005              COMMON /PARDEN/ RHON, RHOT, RHON2, RHOT2, RHON4, RHOT4,
              $                   RHON8, RHOT8, RHONH, RHOTH
0006              COMMON /QUES/ Q0, Q1
0007              LJAM=0
          C
          C GUESS BASED ON QUADRATIC CURVE FIT
          C
0008              IF(MM.EQ.2) THEN
0009                GUESS=0.925D0*4-8.475D0*2+32.45D0
0010              ELSE IF(MM.EQ.4) THEN
0011                GUESS=1.05D0*4-9.35D0*2+34.5D0
0012              ELSE IF(MM.EQ.8) THEN
0013                GUESS=1.1D0*4-9.9D0*2+36.3D0
0014              ELSE
0015                GUESS=15.D0
0016              END IF
0017              CALL MINSER(PUNJAM,PEMIN,TAUOPT,1.D0,GUESS,0.D0,
              $                 50.D0,0.01D0)
0018              TAU=TAUOPT
0019              TAU2=TAU+TAU
0020              PEOO=PEMIN
0021              RETURN
0022              END
```

```
0001          SUBROUTINE MINSER(F,FMIN,XMIN,STEP,GUESS,BLIM,ULIM,TOL)
        C
        C SEARCH FOR MINIMUM OF F(X) OVER THE INTERVAL BLIM <= X <= ULIM
        C
        C TROUBLE MAY OCCUR IF F(X) HAS MULTIPLE LOCAL MINIMA WITHIN THE
        C SEARCH INTERVAL OR IF THE FUNCTION IS VERY STEEP AND STEP IS
        C TOO BIG.
        C
        C             F = NAME OF FUNCTION TO BE MINIMIZED
        C          FMIN = MINIMUM VALUE OF F(X) OVER INTERVAL
        C          XMIN = VALUE OF X FOR WHICH FMIN OCCURS
        C          STEP = INITIAL STEP SIZE FOR SEARCH
        C         GUESS = INITIAL GUESS AT XMIN, BLIM <= GUESS <= ULIM
        C          BLIM = LOWER LIMIT OF SEARCH INTERVAL
        C          ULIM = UPPER LIMIT OF SEARCH INTERVAL
        C           TOL = TOLERANCE ON XMIN; SEARCH STOPS WHEN DX < TOL
        C
        C NOTE: F MUST BE A DOUBLE PRECISION FUNCTION OF ONE
        C       DOUBLE PRECISION ARGUMENT.  ANY PARAMETERS CAN BE PASSED
        C       FROM THE CALLER VIA A COMMON BLOCK.
        C
        C PROGRAMMER: ROBERT H. FRENCH         DATE: 17 MARCH 1986
        C
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          X=GUESS
0004          SLMAX=DABS(X-BLIM)
0005          SUMAX=DABS(ULIM-X)
0006          DX=DMIN1(STEP,SLMAX,SUMAX)
0007          TEST=TOL
0008     10   F0=F(X)
0009          F1=F(X+DX)
        C ARE WE GOING IN THE RIGHT DIRECTION?
0010          IF(F1.LE.F0) GOTO 100
        C ... NO, SWITCH DIRECTION
0011          DX=-DX
0012          F1=F(X+DX)
0013          IF(F1.LE.F0) GOTO 100
        C ELSE WE MUST BE CLOSE TO A MIN. AT X=GUESS, SO CUT
        C STEP SIZE AND TRY AGAIN
0014          DX=DX/10.D0
0015          IF(DABS(DX).GE.TEST) GOTO 10
        C CLOSE ENOUGH AT GUESS
0016     12   XMIN=X
0017          FMIN=F0
0018          RETURN
        C
        C NOW GOING RIGHT DIRECTION.  KEEP GOING UNTIL PAST MINIMUM
        C BY ONE STEP.
        C
0019    100   X2=X+DX+DX
        C HAVE WE REACHED END POINT?
0020          IF(X2.GT.ULIM .OR. X2.LT.BLIM) GOTO 110
```



```
        C ALL OK
0021    105   F2=F(X2)
        C PAST MIN?
0022          IF(F2.GE.F1) GOTO 200
        C ...NO, STEP AGAIN
0023          F0=F1
0024          F1=F2
0025          X=X+DX
0026          GOTO 100
        C MIN MAY BE AT AN ENDPOINT.  CUT STEP SIZE AND TRY AGAIN
        C IF INCREMENT NOT TOO SMALL.
0027    110   IF(DABS(DX).LE.TEST) GOTO 120
0028          X=X+DX
0029          F0=F1
0030    115   DX=DX/10.D0
0031          F1=F(X+DX)
0032          GOTO 100
        C MIN MUST BE AT THE ENDPOINT (OR WITHIN MINIMUM DX THEREOF)
0033    120   IF(X2.LE.BLIM) GOTO 122
        C MIN AT X=ULIM
0034          XMIN=ULIM
0035    121   FMIN=F(XMIN)
0036          RETURN
        C MIN AT BLIM
0037    122   XMIN=BLIM
0038          GOTO 121
        C HAVE PASSED MIN.  IS IT LOCATED CLOSELY ENOUGH YET?
0039    200   IF(DABS(DX).LE.TEST) GOTO 300
        C ... NO, CUT STEP SIZE AND TRY AGAIN
0040          GOTO 115
        C DONE!
        C SINCE F0 >= F1 & F2>= F1 AND ABS(DX)<MIN. DX, CALL F1 THE MIN.
0041    300   FMIN=F1
0042          XMIN=X+DX
0043          RETURN
0044          END
```

APPENDIX I

COMPUTER PROGRAM FOR

CLIPPER RECEIVER FOR M=4 AND L=2

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the clipper receiver for the case of M=4 and L=2 with a numerical search for the worst-case jamming fraction. By increasing the array A to 256 elements and the arrays C and D to 6561 elements each, and changing the array size parameters to calls to PUTIN and LOOKUP, the program may also be used for the case M=8, L=2.

```
0001             PROGRAM CLPRAN
           C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM 4-ARY
           C FSK/FH WITH 2 HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
           C NOISE JAMMING BY NUMERICAL INTEGRATION FOR THE CLIPPER RECEIVER
           C
           C ANALYSIS: L. E. MILLER, R. H. FRENCH
           C PROGRAM: R. H. FRENCH
           C
           C V 2.1.0 - COMPUTATIONS ONLY
           C
           C
0002             IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003             PARAMETER (LJ=51)
0004             CHARACTER*1 YES, NO, REPLY, BLANK
0005             CHARACTER*13 FNAME, GNAME
0006             LOGICAL DOTAU, TEST
0007             LOGICAL*1 GOOD
0008             REAL*4 PRLOG(LJ),DBSJR(LJ),QOPT(LJ)
0009             VIRTUAL A(100),IASUB(100),C(625),ICSUB(625)
0010             VIRTUAL D(625),IDSUB(625),PRERR(625),IPSUB(625)
0011             VIRTUAL PESAV(2400)
           C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
0012             COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
           C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
0013             COMMON /SIZE/ NO
0014             COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
            $                   TAU, TAU2, TAUK, TAUK2
0015             COMMON /PARGEN/ RHON, RHOT
0016             DATA YES, NO, BLANK /'Y', 'N', ' '/
0017             CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0018             CALL GET(NJ,START,DBINC)
0019             SLOTS=NSLOTS
0020             WORBIT=0.5D0*MM/(MM-1.D0)
0021             DO 800 IO=1,NO
0022             DOTAU=.TRUE.
0023             EBNO=10.D0**(DEBNOL(IO)/10.D0)
0024             RHON=K*EBNO/2.D0
0025             IOOUT=DEBNOL(IO)
           C
           C OPEN DATA FILE
           C
0026             IGOUT=GAMMA*1000.D0+0.5D0
0027             WRITE(FNAME,730) MM,IOOUT
0028       730   FORMAT('COJ',I1,'2',I2.2,'.DAT')
0029             WRITE(6,776) MM,DEBNOL(IO)
0030       776   FORMAT('1CLIPPER RECEIVER, OPTIMUM GAMMA RESULTS'/
            $          ' M=',I1,5X,'L=2',5X,'EB/NO=',F8.4//' EB/NJ (dB)',
            $          5X,'P(e)',15X,'Oopt')
0031             WRITE(5,733) FNAME
0032       733   FORMAT(' WORKING ON ',A13)
0033             OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
            $         ERR=750)
```



```
           C
           C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
           C
0034      1300   READ(4) MMIN, EBNOIN, NSLIN, TAU, TAU2, PEOO
           C
           C WE HAVE READ A VALUE OF TAU, SO WE WON'T NEED TO RECOMPUTE
           C IT UNTIL EITHER EB/NO, MM, OR LL CHANGES
           C
0035             DOTAU=.FALSE.
0036             JJ=0
0037       740   JJ=JJ+1
0038             READ(4,END=742) DBSJR(JJ), PRLOG(JJ), QOPT(JJ)
0039             GOTO 740
0040       742   CLOSE(UNIT=4)
0041             GOTO 755
           C
           C NO EXISTING FILE, THIS IS THE FIRST TIME:
           C CREATE FILE HEADER RECORD
           C
0042       750   JJ=1
           C  ... WE CAN'T READ TAU FROM A FILE, SO IF THE FIRST TIME
           C      THROUGH THE LOOP ON EB/NO WE MUST COMPUTE IT.  BUT IF
           C      EB/NO HASN'T CHANGED, WE DON'T NEED TO RECOMPUTE IT SINCE
           C      THE THRESHOLD IS NOT A FUNCTION OF GAMMA NOR OF EB/NJ.
0043             IF(DOTAU) THEN
0044               WRITE(5,757)
0045       757     FORMAT(' SETTING THRESHOLD')
0046               CALL SETTAU(MM,PEOO)
0047               DOTAU=.FALSE.
0048               WRITE(5,1991) MM, TAU
0049               WRITE(6,1991) MM, TAU
0050      1991     FORMAT(' M=',I2,'   L=2     OPT THRES = ',1PD15.8)
0051             END IF
0052             OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
0053             WRITE(4) MM,DEBNOL(IO),NSLOTS,TAU,TAU2,PEOO
0054             CLOSE(UNIT=4)
0055       755   DO 600 IJ=JJ,NJ
0056             IF(IJ.GE.3) THEN
0057               IDQ=QOPT(IJ-1)-QOPT(IJ-2)+0.5D0
0058               IF(IDQ.EQ.0) IDQ=1
0059               Q=QOPT(IJ-1)
0060               IQ=Q
0061             ELSE
0062               IDQ=1
0063               Q=1.DC
0064               IQ=1
0065             END IF
0066             DQ=IDQ
           C GIVE PROGRESS MESSAGE TO TI:
0067             WRITE(5,601) IJ
0068       601   FORMAT(' IJ=',I3)
0069             DEBNJ=START+(IJ-1)*DBINC
```

```
      DBSJR(IJ)=DEBNJ
      DO 602 IJO=1,2400
602   PESAV(IJO)=0.DO
      R=10.DO**(DEBNJ/10.DO)
C PRIME THE ALGORITHM WITH DUMMY OLD VALUES OF P(E)
      P1=0.DO
      P2=0.DO
709   GAMMA=O/SLOTS
      WRITE(GNAME,735) MM,IQ
735   FORMAT('EQ',I1,'2',I4.4,'.DAT')
      OPEN(UNIT=3,FILE=GNAME,STATUS='OLD',FORM='UNFORMATTED',
     $     READONLY,ERR=770)
      WRITE(5,3939)
3939  FORMAT(' READING EVENT FILE')
      READ(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
      GOTO 777
C IF FILE FOR EVENT PROBABILITIES DOES NOT EXIST, CALCULATE THEM
C AND CREATE A FILE.
770   CONTINUE
      WRITE(5,3938)
3938  FORMAT(' CREATING EVENT FILE')
      CALL GENPIE(MM,IQ,NSLOTS,GOOD,A,IASUB,C,ICSUB,
     $            D,IDSUB,NUSED)
      OPEN(UNIT=3,FILE=GNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(3) D,IDSUB,NUSED,GOOD
      CLOSE(UNIT=3)
777   IF(.NOT.GOOD) GOTO 700
      RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
      RHOT=K*RHOTS/2.DO
C EVALUATE THE PROBABILITY
      CALL PSUBE(MM,PESYM,D,IDSUB,NUSED,PRERR,IPSUB,PEOO,
     $           PESAV,IQ)
      P3=PESYM
      IF(P3.GT.P2 .AND. IQ.LT.NSLOTS) THEN
C KEEP ON GOING, WE ARE NOT PAST THE MAXIMUM
        P1=P2
        P2=P3
        IQ=MINO(IQ+IDQ,NSLOTS)
        Q=DMIN1(Q+DQ,SLOTS)
        GOTO 709
      ELSE
        PMAX=DMAX1(P1,P2,P3)
        EPS=0.001DO*PMAX
        TEST=(DABS(P1-P2).LE.EPS .AND. DABS(P1-P3).LE.EPS .AND.
     $     DABS(P2-P3).LE.EPS)
        IF( TEST .OR. IDQ.EQ.1
     $           .OR. ((.NOT.TEST) .AND. IQ.EQ.NSLOTS)) THEN
C         WE ARE DONE WHEN ALL 3 ARE CLOSE TOGETHER OR WHEN DQ=1
C         OR WHEN WE REACHED FULL-BAND JAMMING AND P(E) IS STILL
C         INCREASING
          POPT=PMAX
```



```
          IF(P2.GT.P3) THEN
C           THE OPTIMUM MUST BE THE MIDDLE POINT OF THE 3
            QOPT(IJ)=Q-DQ
            IF(QOPT(IJ).EQ.0.DO) QOPT(IJ)=1.DO
C PREVENT ROUND-OFF FROM MAKING QOPT VS. EB/NJ NON-MONOTONIC
            IF(IJ.GT.1) THEN
              IF(QOPT(IJ).LT.QOPT(IJ-1)) QOPT(IJ)=QOPT(IJ-1)
            END IF
          ELSE
C           THE OPTIMUM IS FULL-BAND JAMMING
            QOPT(IJ)=NSLOTS
          END IF
          GOTO 665
        ELSE
C         NOT LOCATED SUFFICIENTLY ACCURATELY, CUT DQ AND TRY AGAIN
          Q=Q-DQ-DQ
          IQ=IQ-IDQ-IDQ
          IDQ=IDQ/2
          DQ=IDQ
          P2=P1
          P1=0.DO
          Q=Q+DQ
          IQ=IQ+IDQ
          GOTO 709
        END IF
      END IF
665   PE=WORBIT*POPT
      WRITE(6,666) DBSJR(IJ),PE,QOPT(IJ)
666   FORMAT(1X,F7.3,5X,1PD12.5,5X,1PD12.5)
      PRLOG(IJ)=DLOG10(PE)
      OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $     FORM='UNFORMATTED')
      WRITE(4) DBSJR(IJ), PRLOG(IJ), QOPT(IJ)
      CLOSE(UNIT=4)
600   CONTINUE
      OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
      WRITE(4) MM,2,DEBNOL(IO),NSLOTS,DBSJR,PRLOG,QOPT
      CLOSE(UNIT=4)
      WRITE(6,776) MM,DEBNOL(IO)
      DO 689 IJ=1,NJ
      WRITE(6,666) DBSJR(IJ),10.**PRLOG(IJ),QOPT(IJ)
689   CONTINUE
      WRITE(6,688) TAU
688   FORMAT(////' OPTIMUM THRESHOLD FOR ABOVE IS ETA/SIGMA**2 =',
     $           F7.3)
700   CONTINUE
800   CONTINUE
900   CONTINUE
      STOP 0
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:03:46    16-Jul-86           Page 5
CLIPL2M4A.FTN;2            /F77/WR

            SUBROUTINE GET(NJ,START,DBINC)
      C
      C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            CHARACTER*9 FIELD,BLANK9
            COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
            COMMON /SIZE/ NO
            COMMON /SHARE/ DSNR(3,4)
            DATA BLANK9/'         '/
 32         WRITE(5,33)
 33         FORMAT(' BITS/SYMBOL (K) [2]: ',$)
            READ(5,3)K
            IF(K.EQ.0)K=2
            MM=2**K
 1          WRITE(5,2)
 2          FORMAT(' HOW MANY EB/NO? [1]: ',$)
            READ(5,3)NO
 3          FORMAT(I2)
            IF(NO.EQ.0)NO=1
            DO 7 IN=1,NO
            DO=DSNR(IN,K)
 4          WRITE(5,5)IN,DO
 5          FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
            READ(5,6)FIELD
 6          FORMAT(A9)
            IF(FIELD.EQ.BLANK9) THEN
              DEBNOL(IN)=DO
            ELSE
              DECODE(9,61,FIELD)DEBNOL(IN)
 61           FORMAT(F9.6)
            END IF
 7          CONTINUE
            NSLOTS=2400
 38         WRITE(5,39)
 39         FORMAT(' HOW MANY EB/NJ? [51]: ',$)
            READ(5,34,ERR=38) NJ
 34         FORMAT(I3)
            IF(NJ.EQ.0) NJ=51
            IF(NJ.LT.0 .OR. NJ.GT.51) GOTO 32
 40         WRITE(5,41)
 41         FORMAT(' STARTING VALUE FOR EB/NJ (DB) [50.]: ',$)
            READ(5,42,ERR=40) START
            IF(START.EQ.0.D0) START=50.D0
 42         FORMAT(F6.3)
            IF(NJ.EQ.1) RETURN
 35         WRITE(5,36)
 36         FORMAT(' DB INCREMENT FOR EB/NJ [-1.0]: ',$)
            READ(5,37,ERR=35) DBINC
 37         FORMAT(F6.3)
            IF(DBINC.EQ.0.D0) DBINC=-1.D0
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:03:51    16-Jul-86           Page 6
CLIPL2M4A.FTN;2            /F77/WR

            SUBROUTINE PSUBE(M,PE,D,IDSUB,NUSED,PRERR,IPSUB,PEOO,
     $                       PESAV,IQ)
      C
      C COMPUTE UNCONDITIONAL ERROR PROBABILITY
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            INTEGER JAM(4),LUP(4),JSUB(4)
            LOGICAL*1 GO,NONE,STORE
            VIRTUAL PRERR(625),IPSUB(625)
            VIRTUAL D(625),IDSUB(625)
            VIRTUAL PESAV(2400)
            COMMON /SHARE2/ LOW(4),LINC(4)
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                      TAU, TAU2, TAUK, TAUK2
            COMMON /PARDEN/ RHON, RHOT
            DATA STORE/.TRUE./
            IF(PESAV(IQ).NE.0.D0) THEN
              PE=PESAV(IQ)
              RETURN
            END IF
            PE=0.D0
            NPS=0
            DO 10 I=1,M
            LUP(I)=2
            JSUB(I)=0
 10         CONTINUE
      C THE ALL-ZERO JAMMING EVENT P(E) IS AVAILABLE FROM THE SEARCH FOR
      C THE OPTIMUM THRESHOLD, SO PUT IT INTO THE ARRAY OF SAVED VALUES
            CALL LOCN(M,LOW,LUP,JSUB,ISUB)
            CALL PUTIN(PEOO,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
            IF(KODE.NE.0) STOP 'PRERR FULL'
            JAM1=-1
      C START VECTOR-INDEXED LOOP ON JAMMING EVENTS
            CALL VLINIT(JAM,LOW,M)
 100        CONTINUE
            IF(JAM1.NE.JAM(1)) THEN
      C UPDATE TEST VALUE FOR NEXT TIME, AND ...
              JAM1=JAM(1)
            END IF
            CALL EVENT(M,JAM,PIE,D,IDSUB,NUSED)
      C BYPASS CONDITIONAL ERROR PROBABILITY CALCULATION IF EVENT
      C PROBABILITY IS ZERO.  THIS SAVES MUCH TIME.
            IF(PIE.EQ.0.D0)GOTO 101
      C SINCE JAMMING PROBABILITIES DEPEND ONLY ON NO. OF CHANNELS
      C HAVING JAM(I) HOPS JAMMED AND NOT THE ARRANGEMENT OF THE
      C CHANNELS, WE CAN SORT THE NON-SIGNAL CHANNELS INTO ASCENDING
      C NUMBERS OF HOPS JAMMED.  THIS REDUCES NUMBER OF DISTINCT
      C CONDITIONAL ERROR PROBABILITIES WHICH MUST BE SAVED TO AVOID
      C RECOMPUTING THEM UNNECESSARILY.
            DO 111 I=1,M
            JSUB(I)=JAM(I)
 111        CONTINUE
```

```
          IF(M.EQ.2) GOTO 199
          DO 110 I=2,M-1
          DO 120 J=I+1,M
          IF(JSUB(J).LT.JSUB(I)) THEN
            JTEMP=JSUB(I)
            JSUB(I)=JSUB(J)
            JSUB(J)=JTEMP
          END IF
120       CONTINUE
110       CONTINUE
199       CONTINUE
C STORE AS ELEMENT OF A SPARSE ARRAY TO SAVE MEMORY
C EVEN THOUGH WE STORE ZEROS, THE SORTING OF SUBSCRIPTS
C CUTS OUT MANY ELEMENTS.
          CALL LOCN(M,LOW,LUP,JSUB,ISUB)
C TRY TO FIND CONDITIONAL ERROR PROBABILITY IN STORED ARRAY
          CALL LOOKUP(PROB,PRERR,IPSUB,NPS,625,ISUB,STORE,NONE)
C IF IT IS NOT THERE, WE MUST COMPUTE IT
          IF(NONE) THEN
            CALL PSEL(JSUB,M,PROB)
C  ... AND SAVE IT FOR POSSIBLE FUTURE RE-USE
            CALL PUTIN(PROB,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
            IF(KODE.NE.0) STOP 2
          END IF
C SUM UP UNCONDITIONAL ERROR PROBABILITY
          PE=PE+PIE*PROB
C ITERATE THE VECTOR-INDEX LOOP
101       CALL VLITER(JAM,LOW,LUP,LINC,M,GO)
          IF(GO) GOTO 100
          RETURN
          END
```



```
          SUBROUTINE EVENT(M,JAM,PIE,D,IDSUB,NUSED)
C
C SUBROUTINE TO LOOK UP EVENT PROBABILITY FROM STORED ARRAY
C
          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
          LOGICAL*1 STORE,NONE
          DIMENSION JAM(4),LUP(4)
          VIRTUAL D(625),IDSUB(625)
          COMMON /SHARE2/ LOW(4),LINC(4)
          DATA STORE/.FALSE./
C SET UP ARRAY DESCRIPTION D(0:LL,...,0:LL) WITH M DIMENSIONS
          DO 1 I=1,M
          LUP(I)=2
1         CONTINUE
C COMPUTE LINEAR EQUIVALENT SUBSCRIPT FOR JAMMING EVENT
          CALL LOCN(M,LOW,LUP,JAM,ISUB)
C LOOK UP THE VALUE, GET 0.D0 IF NOT THERE
          CALL LOOKUP(PIE,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
          RETURN
          END
```

```
              SUBROUTINE GENPIE(MM,NQ,NSLOTS,GOOD,A,IASUB,C,ICSUB,
          $                        D,IDSUB,NUSED)
       C
       C SUBROUTINE TO GENERATE EVENT PROBABILITIES
       C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              LOGICAL*1 GO,GO2,STORE,NONE,GOOD
              DIMENSION LUP2(4),LUP3(4)
              DIMENSION IUPA(4)
              DIMENSION IUPD(4)
              DIMENSION LUP1(4)
              VIRTUAL A(100),IASUB(100),C(625),ICSUB(625),
          $           D(625),IDSUB(625)
              DIMENSION I(4),II(4),III(4)
       C SHARED STORAGE FOR COMMONLY NEEDED CONSTANT ARRAYS
              COMMON /SHARE2/ LOW(4),LINC(4)
       C SHARED STORATE FOR:
       C (1) INPUT DEFAULT LISTS,
       C (2) CONDITIONAL PROB GEN., AND
       C (3) EVENT PROB. GEN.   THESE ARE NON-OVERLAPPING USAGES.
              COMMON /SHARE/ LUP2,LUP3,IUPD,L1,NUSEA,NUSEC,IERR,
          $                  ISUB,ISUB1,ISUB2,AIN,I,II,III,NN,AOUT,
          $                  CIN,COUT,DOUT,DIN
              DATA I100/100/
              DATA IUPA/4*1/
              DATA LUP1/4*1/
       C STORE=.FALSE. => DON'T STORE ZERO ELEMENTS OF SPARSE ARRAY.
              STORE=.FALSE.
              GOOD=.TRUE.
       9999   IF(NQ.LE.0) THEN
                GOOD=.FALSE.
                RETURN
              END IF
              DO 80 L1=1,MM
              IUPD(L1)=2
       80     CONTINUE
       C JAMMING PATTERN W/NON-ZERO PROBABILITY ON PER-HOP BASIS
              NUSEA=0
       C INITIALIZE VECTOR-INDEX LOOP
              CALL VLINIT(I,LOW,MM)
       90     CONTINUE
              CALL LOCN(MM,LOW,IUPA,I,ISUB)
              CALL PR1HOP(I,MM,NQ,NSLOTS,AIN)
              CALL PUTIN(AIN,A,IASUB,NUSEA,I100,ISUB,IERR,STORE)
              IF(IERR.NE.0)STOP 3
       C ITERATE VECTOR-INDEX LOOP
              CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
              IF(GO) GOTO 90
       C COMPUTATION STARTS HERE. FIRST COPY A INTO D.
       C SINCE ARRAYS ARE A(0:1,0:1,...,0:1) AND D(0:L,0:L,...,0:L)
       C THE COPYING MUST BE DONE ON BASIS OF EQUIVALENT LINEAR
       C SUBSCRIPTS RATHER THAN A SIMPLE MOVE OPERATION.
```



```
              NUSED=0
       C INITIALIZE VECTOR-INDEX LOOP
              CALL VLINIT(I,LOW,MM)
       99     CONTINUE
              CALL LOCN(MM,LOW,IUPA,I,ISUB1)
              CALL LOCN(MM,LOW,IUPD,I,ISUB2)
              CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
              CALL PUTIN(AOUT,D,IDSUB,NUSED,625,ISUB2,IERR,STORE)
              CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
              IF(GO)GOTO 99
       C ... L-1 CONVOLUTIONS ARE NEEDED ...
              DO 9998 L1=1,1
       C SET UP VECTOR-LOOP UPPER LIMITS FOR THIS CONVOLUTION
       9      DO 125 NN=1,MM
              LUP2(NN)=L1
       125    LUP3(NN)=L1+1
              NUSEC=0
              CALL VLINIT(I,LOW,MM)
       98     CONTINUE
              CALL VLINIT(II,LOW,MM)
       97     CONTINUE
       C LOOK UP ELEMENTS AND PERFORM ONE TERM OF THE CONVOLUTION
              CALL LOCN(MM,LOW,IUPA,I,ISUB1)
              CALL LOCN(MM,LOW,IUPD,II,ISUB2)
              DO 21 NN=1,MM
              III(NN)=I(NN)+II(NN)
       21     CONTINUE
              CALL LOCN(MM,LOW,IUPD,III,ISUB3)
              CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
              CALL LOOKUP(DOUT,D,IDSUB,NUSED,625,ISUB2,STORE,NONE)
              CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB3,STORE,NONE)
              CIN=COUT+AOUT*DOUT
              CALL PUTIN(CIN,C,ICSUB,NUSEC,625,ISUB3,IERR,STORE)
              IF(IERR.NE.0) STOP 4
              CALL VLITER(II,LOW,LUP2,LINC,MM,GO2)
              IF(GO2) GOTO 97
              CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
              IF(GO) GOTO 98
       C COPY C TO D IN SORTED ORDER FOR NEXT ITERATION
              NUSED=0
              CALL VLINIT(II,LOW,MM)
       96     CONTINUE
              CALL LOCN(MM,LOW,IUPD,II,ISUB)
              CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB,STORE,NONE)
              DIN=COUT
              CALL PUTIN(DIN,D,IDSUB,NUSED,625,ISUB,IERR,STORE)
              IF(IERR.NE.0) STOP 5
              CALL VLITER(II,LOW,LUP3,LINC,MM,GO)
              IF(GO) GOTO 96
       9998   CONTINUE
              RETURN
              END
```

```
              SUBROUTINE PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
        C
        C THIS SUBROUTINE INSERTS AN ELEMENT INTO A SPARSE ARRAY FOR
        C WHICH ONLY THE NON-ZERO ELEMENTS ARE KEPT IN STORAGE IF
        C THE SWITCH STORE IS .TRUE.
        C
        C THE DOUBLE PRECISION VALUE CIN IS STORED AS C(K) WHERE
        C THE AUXILIARY ARRAY ICSUB IS USED TO KEEP TRACK OF THE
        C SUBSCRIPTS OF THE CORRESPONDING ENTRIES IN C.
        C
        C USAGE:
        C       LOGICAL*1 STORE
        C       DOUBLE PRECISION C,CIN
        C       VIRTUAL ICSUB(NMAX),C(NMAX)
        C       CALL PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
        C WHERE
        C  CIN = VALUE OF ELEMENT TO STORE
        C  C = ARRAY IN WHICH NON-ZERO VALUES ARE ACTUALLY STORED
        C  ICSUB = AUXILIARY ARRAY FOR ACTUAL SUBSCRIPT VALUES
        C  NUSE = NUMBER OF ELEMENTS OF C CURRENTLY OCCUPIED
        C  NMAX = SIZE OF ARRAY C
        C  IERR = ERROR RETURN CODE, 0 IF NO ERROR OR 1 IF THERE IS
        C      NO ROOM AVAILABLE IN C
        C  STORE = .TRUE. TO STORE ZEROES EXPLICITLY, ELSE .FALSE.
        C NOTE: IF CIN=0 AND THE SUBSCRIPT K IS FOUND IN ICSUB, THEN
        C       THE ELEMENT IS DELETED BY SHIFTING DOWNWARD ALL
        C       FOLLOWING ELEMENTS OF THE ARRAY
        C
        C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              VIRTUAL ICSUB(NMAX),C(NMAX)
              LOGICAL*1 STORE
              IERR=0
              IF(STORE) GOTO 5
              IF(CIN.EQ.0.D0) GOTO 30
        5     IF(NUSE.EQ.0)GOTO 20
              DO 10 I=1,NUSE
              IF(ICSUB(I).NE.K) GOTO 10
              C(I)=CIN
              RETURN
        10    CONTINUE
              IF(NUSE.LT.NMAX) GOTO 20
              IERR=1
              RETURN
        20    NUSE=NUSE+1
              ICSUB(NUSE)=K
              C(NUSE)=CIN
              RETURN
        30    DO 40 I=1,NUSE
              J=I
              IF(ICSUB(I).EQ.K) GOTO 50
        40    CONTINUE
```



```
              RETURN
        C
        C REMOVE THE ZEROED ELEMENT AND BUMP COUNT OF ENTRIES USED
        C
        50    DO 60 I=J,NUSE-1
              ICSUB(I)=ICSUB(I+1)
              C(I)=C(I+1)
        60    CONTINUE
              NUSE=NUSE-1
              RETURN
              END
```



```
              SUBROUTINE LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
        C
        C THIS SUBROUTINE RETRIEVES AN ELEMENT OF A SPARSE ARRAY WHICH
        C HAS BEEN STORED COMPACTLY BY STORING ONLY NON-ZERO ELEMENTS.
        C
        C USAGE:
        C       VIRTUAL ICSUB(NMAX), C(NMAX)
        C       LOGICAL*1 STORE, NONE
        C       DOUBLE PRECISION COUT
        C       CALL LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
        C WHERE
        C       COUT = VALUE OF C(K) (OUTPUT FROM SUBROUTINE)
        C          C = ARRAY USED TO STORE NON-ZERO ELEMENTS
        C      ICSUB = AUXILIARY ARRAY TO STORE ACTUAL SUBSCRIPTS
        C          N = NUMBER OF ELEMENTS OF C CURRENTLY IN USE
        C       NMAX = SIZE OF C
        C          K = SUBSCRIPT OF SPARSE ARRAY TO LOOK UP
        C      STORE = .TRUE. IF ZEROES STORED EXPLICITLY, ELSE .FALSE.
        C       NONE = .FALSE. IF ZEROES NOT STORED
        C              .TRUE.  IF ZEROES ARE STORED
        C PROGRAMMER: ROBERT H. FRENCH        DATE: 11 JANUARY 1984
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              VIRTUAL ICSUB(NMAX),C(NMAX)
              LOGICAL*1 STORE, NONE
              NONE=.FALSE.
              DO 10 I=1,N
              IF(ICSUB(I).NE.K)GOTO 10
              COUT=C(I)
              RETURN
        10    CONTINUE
              IF(STORE) THEN
                NONE=.TRUE.
              ELSE
                COUT=0.
              END IF
              RETURN
              END
```

```
0001            SUBROUTINE LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C
        C THIS SUBROUTINE COMPUTES THE EQUIVALENT LINEAR SUBSCRIPT FOR
        C A MULTIDIMENSIONAL ARRAY OF NDIM DIMENSIONS
        C
        C IF THE ARRAY A IS DEFINED AS
        C       DIMENSION A(ILOW(1):IUP(1),...,ILOW(NDIM):IUP(NDIM))
        C AND ISUB(1),...,ISUB(NDIM) IS A SET OF SUBSCRIPTS FOR A,
        C THEN THIS SUBROUTINE RETURNS IN LINEAR THE OFFSET FROM THE
        C ORIGIN OF A TO THE ELEMENT A(ISUB(1),...,ISUB(NDIM)), ASSUMING
        C THE FIRST SUBSCRIPT VARIES MOST RAPIDLY.
        C
        C USAGE:
        C       DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
        C       DATA ILOW/lower limits of defined subscripts of array/
        C       DATA IUP/upper limits of defined subscripts of array/
        C       ...SET ISUB TO DESIRED SEQUENCE OF SUBSCRIPTS...
        C       CALL LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
        C WHERE
        C       NDIM = NUMBER OF DIMENSIONS THE ARRAY HAS
        C       ILOW = ARRAY OF LOWER SUBSCRIPT BOUNDS
        C        IUP = ARRAY OF UPPER SUBSCRIPT BOUNDS
        C       ISUB = ARRAY CONTAINING SUBSCRIPT FOR WHICH LOCATION IS
        C              TO BE COMPUTED
        C     LINEAR = RETURNED VALUE OF OFFSET INTO ARRAY IN MEMORY
        C
        C
        C PROGRAMMER: ROBERT H. FRENCH
        C       DATE: 11 JANUARY 1984
        C
0002            DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
0003            LINEAR=0
0004            DO 10 I=1,NDIM-1
0005            J=NDIM-I+1
0006            LINEAR=(LINEAR+(ISUB(J)-ILOW(J)))*(IUP(J-1)-ILOW(J-1)+1)
0007    10      CONTINUE
0008            LINEAR=LINEAR+ISUB(1)-ILOW(1)
0009            RETURN
0010            END
```



```
0001            SUBROUTINE VLINIT(LVEC,LLOW,LMAX)
        C
        C THIS SUBROUTINE INITIALIZES A "VECTOR DO-LOOP" STRUCTURE
        C DEFINED BY THE FOLLOWING PSEUDO-FORTRAN CODE:
        C       DO 100 LVEC(1)=LLOW(1),LUP(1),LINC(1)
        C       DO 100 LVEC(2)=LLOW(2),LUP(2),LINC(2)
        C                    :
        C                    :
        C       DO 100 LVEC(LMAX)=LLOW(LMAX),LUP(LMAX),LINC(LMAX)
        C                    :
        C                    :
        C       (STATEMENTS IN RANGE OF LOOP)
        C 100   CONTINUE
        C
        C THE COMPANION ROUTINE VLITER HANDLES THE LOOP CONTROL AT THE
        C CONTINUE STATEMENT IN THE ABOVE STRUCTURE
        C
        C USAGE:
        C       LOGICAL*1 GO
        C       DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
        C (INITIALIZE ARRAY LLOW TO STARTING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE ARRAY LUP TO STOPING VALUES OF THE NESTED LOOPS)
        C (INITIALIZE ARRAY LINC TO INCREMENTS OF THE LOOPS)
        C       CALL VLINIT(LVEC,LLOW,LMAX)
        C  100 CONTINUE
        C         .
        C         . (STATEMENTS IN RANGE OF LOOPS)
        C         .
        C       CALL VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
        C       IF(GO)GOTO 100
        C WHERE
        C    LVEC = ARRAY FOR STORAGE OF LOOP INDICES. LVEC(1) IS THE
        C           OUTER-MOST LOOP; LVEC(LMAX), THE INNER-MOST LOOP.
        C    LLOW = ARRAY FOR STORAGE OF LOOP STARTING VALUES, IN SAME
        C           SEQUENCE AS LVEC
        C    LUP = ARRAY FOR STORAGE OF LOOP FINAL VALUES, IN SAME
        C          SEQUENCE AS LVEC
        C    LINC = ARRAY FOR STORAGE OF LOOP INCREMENTS, IN SAME
        C           SEQUENCE AS LVEC
        C    LMAX = NUMBER OF LOOPS NESTED
        C    GO = LOGICAL VARIABLE, .TRUE. IF JUMP BACK TO BEGINNING OF
        C         STATEMENTS IN THE RANGE OF THE LOOP SHOULD OCCUR,
        C         .FALSE. OTHERWISE (I.E. OUTER-MOST LOOP TERMINATED)
        C
        C PROGRAMMER: ROBERT H. FRENCH         DATE: 11 JANUARY 1984
        C
0002            DIMENSION LVEC(LMAX),LLOW(LMAX)
0003            DO 1 N=1,LMAX
0004            LVEC(N)=LLOW(N)
0005    1       CONTINUE
0006            RETURN
0007            END
```

```
      SUBROUTINE VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
C
C LOOP ITERATION LOGIC FOR A "VECTOR DO-LOOP"
C
C SEE DETAILED COMMENTS IN SUBROUTINE VLINIT FOR USAGE AND
C PARAMETER DEFINITIONS
C
C PROGRAMMER: ROBERT H. FRENCH
C       DATE: 11 JANUARY 1984
C
      LOGICAL*1 GO
      DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
      GO=.TRUE.
      DO 100 NDX=1,LMAX
      NSUB=LMAX+1-NDX
      LVEC(NSUB)=LVEC(NSUB)+LINC(NSUB)
      IF((LINC(NSUB).GE.O.AND.LVEC(NSUB).LE.LUP(NSUB))
     $  .OR.(LINC(NSUB).LT.O.AND.LVEC(NSUB).GE.LUP(NSUB))) RETURN
      LVEC(NSUB)=LLOW(NSUB)
100   CONTINUE
      GO=.FALSE.
      RETURN
      END
```



```
      SUBROUTINE PR1HOP(I,KM,KQ,KN,AIN)
C
C THIS SUBROUTINE COMPUTES THE EVENT PROBABILITY FOR ALL
C POSSIBLE JAMMING PATTERNS WITH NON-ZERO PROBABILITY FOR
C L=1 HOP/SYMBOL FOR RMFSK/FH IN PBNJ
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION I(4)
      AIN=0.DO
      KJAM=0
      DO 1 K=1,KM
      KJAM=KJAM+I(K)
    1 CONTINUE
C IF THIS IS AN IMPOSSIBLE CASE, RETURN WITH RESULT = 0.0
      IF(KJAM.GT.MINO(KQ,KM)) RETURN
      KPMAX=KJAM-1
      LPMAX=KM-KJAM-1
      JPMAX=KM-1
      IMAX=MAXO(KPMAX,LPMAX,JPMAX)
      PROD=1.DO
      Q=KQ
      DIFFNQ=KN-KQ
      EN=KN
      DO 100 LOOP=0,IMAX
      F=LOOP
      IF(LOOP.LE.KPMAX) PROD=PROD*(Q-F)
      IF(LOOP.LE.JPMAX) PROD=PROD/(EN-F)
      IF(LOOP.LE.LPMAX) PROD=PROD*(DIFFNQ-F)
  100 CONTINUE
      AIN=PROD
      RETURN
      END
```

```
      BLOCK DATA
C INITIALIZE SHARED CONSTANTS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /SHARE/ DSNR(3,4)
      COMMON /SHARE2/ LOW(4),LINC(4)
      COMMON /WTS/ X(8),W(8)
C
C WEIGHTS AND ABSCISSAS FOR 16-POINT GAUSSIAN QUADRATURE
C
      DATA X/ 0.0950125098376374401850D0,
     $        0.2816035507792589132300D0,
     $        0.4580167776572273863420D0,
     $        0.6178762444026437484470D0,
     $        0.7554044083550030338950D0,
     $        0.8656312023878317438800D0,
     $        0.9445750230732325760780D0,
     $        0.9894009349916499325960D0 /
      DATA W/ 0.1894506104550684962850D0,
     $        0.1826034150449235888670D0,
     $        0.1691565193950025381890D0,
     $        0.1495959888165767320610D0,
     $        0.1246289712555338720520D0,
     $        0.0951585116824927848100D0,
     $        0.0622535239386478928630D0,
     $        0.0271524594117540948520D0 /
C DEFAULT LISTS FOR INTERACTIVE PARAMETER INPUTS
C ARE SHARED WITH LARGE WORKING STORAGE ARRAYS SINCE THEY
C MAY BE DESTROYED ONCE THE INPUT PARAMETERS ARE SET UP
      DATA DSNR /13.3524700, 12.313300, 10.9444300,
     $           10.6065720D0, 9.6284D0,  8.35248D0,
     $            9.09401D0,  8.1690D0,  6.971995D0,
     $            8.07835D0,  7.1996D0,  6.0696460D0/
C FREQUENTLY NEEDED CONSTANT ARRAYS AND SCALARS
      DATA LOW/4*0/,LINC/4*1/
      END
```



```
      SUBROUTINE FSEL(JSUB,MM,PROB)
C
C RANDOM MFSK/FH IN PARTIAL BAND NOISF JAMMING,
C GIVEN A JAMMING EVENT, WITH CLIPPER RECEIVER
C
C       JSUB - JAMMING EVENT VECTOR
C         MM - ALPHABET SIZE
C       PROB - RESULTING CONDITIONAL ERROR PROBABILITY
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION JSUB(MM), WORK(30), STACK(30), HEAP(30)
      EXTERNAL DG16, PGRAND
      INTEGER NCHAN(0:3)
      COMMON /JAMCNT/ NCHAN
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
      COMMON /SCJAM/ JAMSC
      KSUB=0
      DO 6 I=1,MM
      KSUB=KSUB+JSUB(I)
    6 CONTINUE
C SET UP VALUES WHICH WILL REMAIN IF THIS IS THE NOTHING-JAMMED CASE
      Q0=Q(2.D0*DSQRT(0.5D0*RHON),DSQRT(2.D0*TAU))
      Q1=1.D0
      TAUK=0.D0
C IF ANYTHING IS JAMMED, SET UP JAMMING-RELATED QUANTITIES
      IF(KSUB.NE.0) THEN
        BIGK=RHON/RHOT
        BK1=BIGK-1.D0
        AAB=BIGK/BK1
        BAB=1.D0/BK1
        TAUK=TAU/BIGK
        TAUK2=TAU2/BIGK
        Q1=Q(2.D0*DSQRT(0.5D0*RHOT),DSQRT(2.D0*TAUK))
      END IF
C
C COUNT NUMBER OF NONSIGNAL CHANNELS WITH Lm HOPS JAMMED
C
      DO 10 I=0,2
      NCHAN(I)=0
   10 CONTINUE
      DO 11 I=2,MM
      KSUB=JSUB(I)
      NCHAN(KSUB)=NCHAN(KSUB)+1
   11 CONTINUE
      JAMSC=JSUB(1)
C
C DO THE CONTINUOUS PART OF THE DENSITY IN SECTIONS
C
      PCONT=0.D0
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:23    16-Jul-86              Page 20
CLIPL2M4A.FTN;2          /F77/WR

              DO 13 ISECT=1,2
              XL=(ISECT-1)*TAU
              XU=ISECT*TAU
              CALL ADQUAD(XL,XU,CHUNK,DG16,PGRAND,1.D-8,WORK,
         $                STACK,HEAP,30,KODE)
              CALL TEST2(KODE,10)
              PCONT=PCONT+CHUNK
        13    CONTINUE
      C
      C DO THE TIE PART OF THE DENSITY
      C
              CALL TIES(JSUB,MM,PTIE)
      C
      C PUT THEM TOGETHER
      C
              PROB=1.DO-PCONT-PTIE
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:27    16-Jul-86              Page 21
CLIPL2M4A.FTN;2          /F77/WR

              DOUBLE PRECISION FUNCTION PGRAND(BETA)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              INTEGER NCHAN(0:3)
              COMMON /SCJAM/ JAMSC
              COMMON /JAMCNT/ NCHAN
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
         $                    TAU, TAU2, TAUK, TAUK2
              COMMON /PARDEN/ RHON, RHOT
              COMMON /QUES/ Q0, Q1
              PROD=1.DO
              DO 10 I=0,2
              IF(NCHAN(I).NE.0) THEN
                LJAM=I
                X=GL(BETA)
                PROD=PROD*DXI(X,NCHAN(I))
              END IF
        10    CONTINUE
              LJAM=JAMSC
              Y=PZ1(BETA)
              PGRAND=Y*PROD
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:30    16-Jul-86              Page 22
CLIPL2M4A.FTN;2          /F77/WR

              SUBROUTINE ADQUAD(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
      C
      C ADAPTIVE QUADRATURE ALGORITHM
      C   XL - LOWER LIMIT OF INTEGRAL (IN)
      C   XU - UPPER LIMIT OF INTEGRAL (IN)
      C    Y - VALUE OF INTEGRAL (OUT)
      C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
      C               WITH CALLING SEQUENCE
      C               CALL QR(XL,XU,F,Y)
      C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
      C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
      C WORK - WORK ARRAY OF SIZE N (IN)
      C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
      C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
      C        SAME ARRAY AS WORK (IN)
      C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
      C KODE - ERROR INDICATOR (OUT)
      C         0 -- NO ERROR
      C         1 -- WORK ARRAYS TOO SMALL
      C         2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
      C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
      C               ATTAINING REQUIRED ACCURACY
      C R. H. FRENCH, 14 AUGUST 1984
      C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              EXTERNAL F
              DIMENSION WORK(N),STACK(N),HEAP(N)
              KODE=0
              Y=0.D0
              WORK(1)=XU
              CALL QR(XL,XU,F,T)
              HEAP(1)=T
              A=XL
              NPTS=1
              EPS=TOL
              STACK(1)=EPS
        10    B=WORK(NPTS)
              XM=(A+B)*0.5D0
              CALL QR(A,XM,F,P1)
              CALL QR(XM,B,F,P2)
              IF(DABS(T-P1-P2).LE.EPS) GOTO 20
      C SPLIT IT
              NPTS=NPTS+1
              IF(NPTS.GT.N) THEN
                KODE=1
                RETURN
              END IF
              WORK(NPTS)=XM
              HEAP(NPTS)=P2
              T=P1
              EPS=EPS/2.D0
```

```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```



```
      SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C               WITH CALLING SEQUENCE
C               CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
C        SAME ARRAY AS WORK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=EPS/2.D0
```

```
PDP-11 FORTRAN-77 V4.0-1          14:04:33    16-Jul-86          Page 25
CLIPL2M4A.FTN;2          /F77/WR

            IF(EPS.EQ.0.D0) THEN
              KODE=2
              RETURN
            END IF
            STACK(NPTS)=EPS
            GOTO 10
C FINISHED A PIECE
20          Y=Y+P1+P2
            EPS=STACK(NPTS)
            T=HEAP(NPTS)
            NPTS=NPTS-1
            A=B
            IF(NPTS.EQ.0) RETURN
            GOTO 10
            END
```

```
PDP-11 FORTRAN-77 V4.0-1          14:04:36    16-Jul-86          Page 26
CLIPL2M4A.FTN;2          /F77/WR

            SUBROUTINE DG16(A,B,F,ANSWER)
C
C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
C
C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
C
C R. H. FRENCH, 28 FEBRUARY 1986
C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON /WTS/ X(8),W(8)
            ANSWER=0.D0
            BMAO2=(B-A)/2.D0
            BPAO2=(B+A)/2.D0
            DO 10 I=1,8
            C=X(I)*BMAO2
            Y1=BPAO2+C
            Y2=BPAO2-C
            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
10          CONTINUE
            ANSWER=ANSWER*BMAO2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1          14:04:38    16-Jul-86          Page 27
CLIPL2M4A.FTN;2          /F77/WR

            SUBROUTINE DGXVI(A,B,F,ANSWER)
C
C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
C
C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
C
C R. H. FRENCH, 28 FEBRUARY 1986
C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            COMMON /WTS/ X(8),W(8)
            ANSWER=0.D0
            BMAO2=(B-A)/2.D0
            BPAO2=(B+A)/2.D0
            DO 10 I=1,8
            C=X(I)*BMAO2
            Y1=BPAO2+C
            Y2=BPAO2-C
            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
10          CONTINUE
            ANSWER=ANSWER*BMAO2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:40    16-Jul-86          Page 28
CLIPL2M4A.FTN;2           /F77/WR

0001            SUBROUTINE TEST(ID)
        C
        C TEST RETURN CODE FROM BESSEL FUNCTION
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            IF(KODE.EQ.0) RETURN
0005            WRITE(5,1) KODE, ID
0006        1   FORMAT(' BESSEL FUNCTION CODE = ',I2,
           $          ' FROM CALL NUMBER ',I5)
0007            STOP 'FATAL ERROR'
0008            END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:41    16-Jul-86          Page 29
CLIPL2M4A.FTN;2           /F77/WR

0001            SUBROUTINE TEST2(KODE,ID)
        C
        C TEST RETURN CODE FROM ADQUAD/ADQUA2
        C
0002            IF(KODE.EQ.0) RETURN
0003            WRITE(5,1) KODE, ID
0004        1   FORMAT(' ADAPTIVE INTEGRATOR CODE = ',I2,
           $          ' FROM CALL NUMBER ',I5)
0005            STOP 'FATAL ERROR'
0006            END
```

```
      DOUBLE PRECISION FUNCTION PZ1(Y)
C
C     SIGNAL CHANNEL P.D.F. WITH CHANGE OF VARIABLE Y=AX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION WORK(30), STACK(30), HEAP(30)
      LOGICAL*1 REG1, REG2
      EXTERNAL DGXVI, F20, F21, F22
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
      COMMON /XCOM/ XCON
      COMMON /OUTER/ XXX, XXXK
      XXX=Y
      IF(LJAM.GE.1) THEN
        XXXK=Y/BIGK
        YK=XXXK
        YTK=(Y-TAU)/BIGK
        YTK2=(Y-TAU2)/BIGK
      END IF
      REG1=Y.GE.0.D0 .AND. Y.LT.TAU
      REG2=Y.GE.TAU .AND. Y.LT.TAU2
C
C     TWO HOPS PER SYMBOL
C
 2000 GOTO (2100, 2200, 2300), LJAM+1
C
C             NO HOPS JAMMED
C
 2100 IF(REG1) THEN
        BARG1=DSQRT(8.D0*RHON*Y)
        IO1=1
        CALL DXBT(2100)
        PZ1=0.5D0*DSQRT(2.D0*Y/RHON)*DEXP(BARG1-Y-2.D0*RHON)*B1
      ELSE IF(REG2) THEN
        BARG1=DSQRT(4.D0*RHON*(Y-TAU))
        IO1=0
        CALL DXBT(2101)
        PART=2.D0*Q0*DEXP(BARG1-Y+TAU-RHON)*B1
        XCON=Y+2.D0*RHON
        CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F20,1.D-9,WORK,STACK,
     $              HEAP,30,KOD)
        CALL TEST2(KOD,2100)
        PZ1=PART+ANSWER
      ELSE
        PZ1=0.D0
      END IF
      GOTO 9000
```



```
C
C             ONE HOP JAMMED
C
 2200 IF(REG1) THEN
        XCON=YK+RHON+RHOT
        CALL ADQUA2(0.D0,Y,ANSWER,DGXVI,F21,1.D-9,WORK,STACK,
     $              HEAP,30,KOD)
        CALL TEST2(KOD,2200)
        PZ1=ANSWER/BIGK
      ELSE IF(REG2) THEN
        BARG1=DSQRT(4.D0*RHOT*YTK)
        IO1=0
        CALL DXBT(2201)
        PART=Q0*DEXP(BARG1-YK+TAUK-RHOT)*B1/BIGK
        BARG1=DSQRT(4.D0*RHON*(Y-TAU))
        IO1=0
        CALL DXBT(2202)
        PART=PART+Q1*DEXP(BARG1-Y+TAU-RHON)*B1
        XCON=YK+RHON+RHOT
        CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F21,1.D-9,WORK,STACK,
     $              HEAP,30,KOD)
        CALL TEST2(KOD,2202)
        PZ1=PART+ANSWER/BIGK
      ELSE
        PZ1=0.D0
      END IF
      GOTO 9000
C
C             TWO HOPS JAMMED
C
 2300 IF(REG1) THEN
        BARG1=DSQRT(8.D0*RHOT*YK)
        IO1=1
        CALL DXBT(2300)
        PZ1=0.5D0*DSQRT(2.D0*Y/RHON)*DEXP(BARG1-YK-2.D0*RHOT)
     $      *B1/BIGK
      ELSE IF(REG2) THEN
        BARG1=DSQRT(4.D0*RHOT*YTK)
        IO1=0
        CALL DXBT(2301)
        PART=2.D0*Q1*DEXP(BARG1-YTK-RHOT)*B1/BIGK
        XCON=YK+2.D0*RHOT
        CALL ADQUA2(Y-TAU,TAU,ANSWER,DGXVI,F22,1.D-9,WORK,STACK,
     $              HEAP,30,KOD)
        CALL TEST2(KOD,2300)
        PZ1=PART+ANSWER/(BIGK*BIGK)
      ELSE
        PZ1=0.D0
      END IF
 9000 CONTINUE
      RETURN
      END
```

```
            DOUBLE PRECISION FUNCTION GL(Y)
C
C    NON-SIGNAL CHANNEL CUMULATIVE DISTRIBUTION FUNCTION
C           WITH CHANGE OF VARIABLE Y=AX
C
            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
            LOGICAL*1 REG1, REG2, REG3
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                      TAU, TAU2, TAUK, TAUK2
            REG1=Y.GE.0.D0 .AND. Y.LT.TAU
            REG2=Y.GE.TAU  .AND. Y.LT.TAU2
            IF(LJAM.GT.0) THEN
              YK=Y/BIGK
              YTK=(Y-TAU)/BIGK
              YTK2=(Y-TAU2)/BIGK
            END IF
C
C           TWO HOPS PER SYMBOL
C
 2000       GOTO (2100, 2200, 2300), LJAM+1
C
C                   NO HOPS JAMMED
C
 2100       IF(REG1) THEN
              GL=1.D0-(1.D0+Y)*DEXP(-Y)
            ELSE IF(REG2) THEN
              GL=1.D0-(1.D0+TAU2-Y)*DEXP(-Y)
            ELSE IF(Y.GE.TAU2) THEN
              GL=1.D0
            ELSE
              GL=0.D0
            END IF
            GOTO 9000
C
C                   ONE HOP JAMMED
C
 2200       IF(REG1) THEN
              GL=1.D0-AAB*DEXP(-YK)+BAB*DEXP(-Y)
            ELSE IF(REG2) THEN
              GL=1.D0-AAB*DEXP(TAU-TAUK-Y)+BAB*DEXP(TAUK-TAU-YK)
            ELSE IF(Y.GE.TAU2) THEN
              GL=1.D0
            ELSE
              GL=0.D0
            END IF
            GOTO 9000
C
C                   TWO HOPS JAMMED
C
 2300       IF(REG1) THEN
              GL=1.D0-(1.D0+YK)*DEXP(-YK)
```



```
            ELSE IF(REG2) THEN
              GL=1.D0-(1.D0+TAUK2-YK)*DEXP(-YK)
            ELSE IF(Y.GE.TAU2) THEN
              GL=1.D0
            ELSE
              GL=0.D0
            END IF
            GOTO 9000
 9000       CONTINUE
            RETURN
            END
```



```
            DOUBLE PRECISION FUNCTION F20(U)
C
C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=0
C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                      TAU, TAU2, TAUK, TAUK2
            COMMON /PARDEN/ RHON, RHOT
            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
            COMMON /QUES/ Q0, Q1
            COMMON /XCOM/ XCON
            COMMON /OUTER/ XXX, XXXK
            BARG1=DSQRT(4.D0*RHON*U)
            BARG2=DSQRT(4.D0*RHON*(XXX-U))
            IO1=0
            IO2=0
            CALL BPROD(2110)
            F20=DEXP(BARG1+BARG2-XCON)*B1*B2
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:55    16-Jul-86          Page 35
CLIPL2M4A.FTN;2          /F77/WR

      DOUBLE PRECISION FUNCTION F21(U)
C
C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=1
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
      COMMON /XCOM/ XCON
      COMMON /OUTER/ XXX, XXXK
      BARG1=DSQRT(4.D0*RHON*U)
      BARG2=DSQRT(4.D0*RHOT*(XXXK-U/BIGK))
      I01=0
      I02=0
      CALL BPROD(2210)
      F21=DEXP(BARG1+BARG2-XCON-U+U/BIGK)*B1*B2
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:04:57    16-Jul-86          Page 36
CLIPL2M4A.FTN;2          /F77/WR

      SUBROUTINE BPROD(IDENT)
C
C COMPUTE TWO BESSEL FUNCTIONS, ARGUMENTS AND RESULTS IN COMMON
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
      CALL DXBT(IDENT)
      CALL DXBESI(BARG2,I02,B2,KODE)
      CALL TEST(IDENT+1)
      RETURN
      END
```



```
PDP-11 FORTRAN-77 V4.0-1        14:04:58    16-Jul-86          Page 37
CLIPL2M4A.FTN;2          /F77/WR

      DOUBLE PRECISION FUNCTION F22(U)
C
C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=2, LJAM=2
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
      COMMON /QUES/ Q0, Q1
      COMMON /XCOM/ XCON
      COMMON /OUTER/ XXX, XXXK
      BARG1=DSQRT(4.D0*RHOT*U/BIGK)
      BARG2=DSQRT(4.D0*RHOT*(XXXK-U/BIGK))
      I01=0
      I02=0
      CALL BPROD(2310)
      F22=DEXP(BARG1+BARG2-XCON)*B1*B2
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        14:05:00    16-Jul-86          Page 38
CLIPL2M4A.FTN;2          /F77/WR

      SUBROUTINE DXBT(ID)
C
C CALL DXBESI AND TEST RETURN CODE
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
      CALL DXBESI(BARG1,I01,B1,KODE)
      CALL TEST(ID)
      RETURN
      END
```

```
      SUBROUTINE TIES(JSUB,MM,PTIE)
C
C  COMPUTE PROBABILITY OF CORRECT DECISION GIVEN THAT
C  SEVERAL SATURATED CHANNELS ARE TIED
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION JSUB(MM), LLOW(7), LINC(7), LUP(7), NU(7),
     $          P2LM(2:8)
      LOGICAL*1 GO
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
C NUMBER OF NON-SIGNAL CHANNELS
      MNU=MM-1
      PTIE=0.D0
      CUE0=DEXP(-TAU)
      CUE1=DEXP(-TAUK)
      P1L=DXI(Q0,2-JSUB(1))*DXI(Q1,JSUB(1))
      DO 10 I=2,MM
      P2LM(I)=DXI(CUE0,2-JSUB(I))*DXI(CUE1,JSUB(I))
   10 CONTINUE
C
C SET UP VECTOR LOOP PARAMETERS
C
      DO 20 I=1,MM-1
      LLOW(I)=0
      LINC(I)=1
      LUP(I)=1
   20 CONTINUE
      PTIE=0.D0
C START LOOP ON THE TIE EVENTS
      CALL VLINIT(NU,LLOW,MM-1)
   30 NUSUM=0
      DO 40 I=1,MM-1
      NUSUM=NUSUM+NU(I)
   40 CONTINUE
      FRAC=1.D0/(1.D0+NUSUM)
      PROD=1.D0
      DO 50 M=2,MM
      IF(NU(M-1).EQ.1) THEN
        PROD=PROD*P2LM(M)
      ELSE
        PROD=PROD*(1.D0-P2LM(M))
      END IF
   50 CONTINUE
      PTIE=FRAC*PROD+PTIE
      CALL VLITER(NU,LLOW,LUP,LINC,MM-1,GO)
      IF(GO) GOTO 30
      PTIE=PTIE*P1L
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION PUNJAM(ETA)
C
C FUNCTION FOR UNJAMMED P(E) FOR OPT. THRESHOLD SEARCH
C     NOTE: WHEN JAMMING EVENT IS (0,0,...,0), THE VARIABLES
C           BIGK, AAB, BAB, TAUK, AND TAUK2 ARE NOT
C           USED IN THE COMPUTATIONS.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION NOJAM(4)
      COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      DATA NOJAM/0,0,0,0/
      LJAM=0
      TAU=ETA
      TAU2=TAU+ETA
      CALL PSEL(NOJAM,MM,P)
      PUNJAM=P
      RETURN
      END
```



```
      SUBROUTINE SETTAU(MM,PE00)
C
C  SEARCH FOR OPTIMUM THRESHOLD IN ABSENCE OF JAMMING
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL PUNJAM
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM,
     $                TAU, TAU2, TAUK, TAUK2
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
      LJAM=0
C GUESS BASED ON QUADRATIC CURVE FIT
      IF(MM.EQ.2) THEN
        GUESS=0.925D0*4-8.475D0*2+32.45D0
      ELSE IF(MM.EQ.4) THEN
        GUESS=1.05D0*4-9.35D0*2+34.5D0
      ELSE IF(MM.EQ.8) THEN
        GUESS=1.1D0*4-9.9D0*2+36.3D0
      ELSE
        GUESS=15.D0
      END IF
      CALL MINSER(PUNJAM,PEMIN,TAUOPT,1.D0,GUESS,0.D0,
     $            50.D0,0.01D0)
      TAU=TAUOPT
      TAU2=TAU+TAU
      PE00=PEMIN
      RETURN
      END
```

```
            SUBROUTINE MINSER(F,FMIN,XMIN,STEP,GUESS,BLIM,ULIM,TOL)
C
C SEARCH FOR MINIMUM OF F(X) OVER THE INTERVAL BLIM <= X <= ULIM
C
C TROUBLE MAY OCCUR IF F(X) HAS MULTIPLE LOCAL MINIMA WITHIN THE
C SEARCH INTERVAL OR IF THE FUNCTION IS VERY STEEP AND STEP IS
C TOO BIG.
C
C              F = NAME OF FUNCTION TO BE MINIMIZED
C           FMIN = MINIMUM VALUE OF F(X) OVER INTERVAL
C           XMIN = VALUE OF X FOR WHICH FMIN OCCURS
C           STEP = INITIAL STEP SIZE FOR SEARCH
C          GUESS = INITIAL GUESS AT XMIN, BLIM <= GUESS <= ULIM
C           BLIM = LOWER LIMIT OF SEARCH INTERVAL
C           ULIM = UPPER LIMIT OF SEARCH INTERVAL
C            TOL = TOLERANCE ON XMIN; SEARCH STOPS WHEN DX < TOL
C
C NOTE: F MUST BE A DOUBLE PRECISION FUNCTION OF ONE DOUBLE PRECISION
C       ARGUMENT.  ANY PARAMETERS CAN BE PASSED FROM THE CALLER VIA
C       A COMMON BLOCK.
C
C PROGRAMMER: ROBERT H. FRENCH        DATE: 17 MARCH 1986
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        X=GUESS
        SLMAX=DABS(X-BLIM)
        SUMAX=DABS(ULIM-X)
        DX=DMIN1(STEP,SLMAX,SUMAX)
        TEST=TOL
   10   F0=F(X)
        F1=F(X+DX)
C ARE WE GOING IN THE RIGHT DIRECTION?
        IF(F1.LE.F0) GOTO 100
C ... NO, SWITCH DIRECTION
        DX=-DX
        F1=F(X+DX)
        IF(F1.LE.F0) GOTO 100
C ELSE WE MUST BE CLOSE TO A MIN. AT X=GUESS, SO CUT
C STEP SIZE AND TRY AGAIN
        DX=DX/10.D0
        IF(DABS(DX).GE.TEST) GOTO 10
C CLOSE ENOUGH AT GUESS
   12   XMIN=X
        FMIN=F0
        RETURN
C
C NOW GOING RIGHT DIRECTION.
C KEEP GOING UNTIL PAST MINIMUM BY ONE STEP.
C
  100   X2=X+DX+DX
C HAVE WE REACHED END POINT?
        IF(X2.GT.ULIM .OR. X2.LT.BLIM) GOTO 110
```



```
C ALL OK
  105   F2=F(X2)
C PAST MIN?
        IF(F2.GE.F1) GOTO 200
C ...NO, STEP AGAIN
        F0=F1
        F1=F2
        X=X+DX
        GOTO 100
C MIN MAY BE AT AN ENDPOINT.  CUT STEP SIZE AND TRY AGAIN
C IF INCREMENT NOT TOO SMALL.
  110   IF(DABS(DX).LE.TEST) GOTO 120
        X=X+DX
        F0=F1
  115   DX=DX/10.D0
        F1=F(X+DX)
        GOTO 100
C MIN MUST BE AT THE ENDPOINT (OR WITHIN MINIMUM DX THEREOF)
  120   IF(X2.LE.BLIM) GOTO 122
C MIN AT X=ULIM
        XMIN=ULIM
  121   FMIN=F(XMIN)
        RETURN
C MIN AT BLIM
  122   XMIN=BLIM
        GOTO 121
C HAVE PASSED MIN.  IS IT LOCATED CLOSELY ENOUGH YET?
  200   IF(DABS(DX).LE.TEST) GOTO 300
C ... NO, CUT STEP SIZE AND TRY AGAIN
        GOTO 115
C DONE!
C SINCE F0 >= F1 & F2>= F1 AND ABS(DX)<MIN. DX, CALL F1 THE MIN.
  300   FMIN=F1
        XMIN=X+DX
        RETURN
        END
```

APPENDIX J

COMPUTER PROGRAM FOR

CLIPPER RECEIVER WITH L=3 HOPS/SYMBOL

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the clipper receiver when L=3 hops/symbol, using a numerical search for the worst-case jamming fractions. If M>4 the sizes of the arrays used in computing event probabilities must be increased and the corresponding array-size parameters in calls to PUTIN and LOOKUP changed accordingly.

```
              PROGRAM CLIP3R
        C THIS PROGRAM COMPUTES THE ERROR PROBABILITY FOR RANDOM M-ARY
        C FSK/FH WITH 3 HOPS/BIT IN THE PRESENCE OF PARTIAL-BAND
        C NOISE JAMMING BY NUMERICAL INTEGRATION FOR THE CLIPPER RECEIVER
        C AND BETWEEN-CONDITIONAL-PROB. RESTART CAPABILITY
        C
        C ANALYSIS: L. E. MILLER, R. H. FRENCH
        C PROGRAM: R. H. FRENCH
        C
        C V 3.1.0 - COMPUTATIONS ONLY
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              PARAMETER (LJ=11)
              CHARACTER*1 YES, NO, REPLY, BLANK
              CHARACTER*13 FNAME, GNAME
              LOGICAL DOTAU, TEST
              LOGICAL*1 GOOD, RESTRT
              DIMENSION POFQ(50), IQL(50)
              REAL*4 PRLOG(LJ),DBSJR(LJ),QOPT(LJ)
              VIRTUAL A(100),IASUB(100),C(625),ICSUB(625)
              VIRTUAL D(625),IDSUB(625),PRERR(625),IPSUB(625)
        C COMMON /INPUTS/ PASSES PARAMETERS OF THE RUN
              COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
        C COMMON /SIZE/ PASSES NUMBERS OF PARAMETERS
              COMMON /SIZE/ NO
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3,TAUK, TAUK2, TAUK3
              COMMON /PARDEN/ RHON, RHOT
              COMMON /RSPAR/ JAM(4),JAM1,NPS,IJ,RESTRT
              DATA YES, NO, BLANK /'Y', 'N', ' '/
              CALL JSLGO
              CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
              CALL GET(NJ,START,DBINC)
              SLOTS=NSLOTS
              WORBIT=0.5D0*MM/(MM-1.D0)
              DO 800 IO=1,NO
              DOTAU=.TRUE.
              EBNO=10.D0**(DEBNOL(IO)/10.D0)
              RHON=K*EBNO/3.D0
              IOOUT=DEBNOL(IO)
        C
        C OPEN DATA FILE
        C
              WRITE(FNAME,730) MM,IOOUT
          730 FORMAT('COJ',I1,'3',I2.2,'.DAT')
              WRITE(6,776) MM,DEBNOL(IO)
          776 FORMAT('1CLIPPER RECEIVER, OPTIMUM GAMMA RESULTS'/
           $        ' M=',I2,5X,'L=3',5X,'EB/NO=',F8.4//' EB/NJ (dB)',
           $        5X,'P(e)',15X,'Qopt')
              WRITE(5,733) FNAME
          733 FORMAT(' WORKING ON ',A13)
              OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',FORM='UNFORMATTED',
           $       ERR=750)
```



```
        C
        C HAVE AN EXISTING FILE, READ TO SEE HOW FAR WE GOT BEFORE
        C
         1300 READ(4) MMIN, EBNOIN, NSLIN, TAU, TAU2, TAU3, PEOO
        C
        C WE HAVE READ A VALUE OF TAU, SO WE WON'T NEED TO RECOMPUTE IT UNTIL
        C EITHER EB/NO, MM, OR LL CHANGES
        C
              DOTAU=.FALSE.
              JJ=0
          740 JJ=JJ+1
              READ(4,END=742) DBSJR(JJ), PRLOG(JJ), QOPT(JJ)
              GOTO 740
          742 CLOSE(UNIT=4)
              OPEN(UNIT=4,FILE='RESUME.DAT',STATUS='UNKNOWN',
           $       READONLY,ERR=756,FORM='UNFORMATTED')
              READ(4) JAM,JAM1,PRERR,NPS,IPSUB,JJIN
              CLOSE(UNIT=4)
              RESTRT=JJ.EQ.JJIN
              GOTO 755
        C
        C NO RESUME DATA FILE, SO WE MUST BE STARTING
        C A POINT FROM THE BEGINNING
        C
          756 RESTRT=.FALSE.
              GOTO 755
        C
        C NO EXISTING FILE, THIS IS THE FIRST TIME:
        C CREATE FILE HEADER RECORD
        C
          750 JJ=1
        C ... WE CAN'T READ TAU FROM A FILE, SO IF THE FIRST TIME
        C     THROUGH THE LOOP ON EB/NO WE MUST COMPUTE IT.  BUT IF
        C     EB/NO HASN'T CHANGED, WE DON'T NEED TO RECOMPUTE IT SINCE
        C     THE THRESHOLD IS NOT A FUNCTION OF GAMMA NOR OF EB/NJ.
              RESTRT=.FALSE.
              IF(DOTAU) THEN
                WRITE(5,757)
          757   FORMAT(' SETTING THRESHOLD')
                CALL SETTAU(MM,PEOO)
                DOTAU=.FALSE.
                WRITE(5,1991) MM, TAU
                WRITE(6,1991) MM, TAU
         1991   FORMAT(' M=',I2,'   L=3    OPT THRES = ',1PD15.8)
              END IF
              OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
              WRITE(4) MM,DEBNOL(IO),NSLOTS,TAU,TAU2,TAU3,PEOO
              CLOSE(UNIT=4)
          755 DO 600 IJ=JJ,NJ
              IF(IJ.GE.3) THEN
                IDQ=QOPT(IJ-1)-QOPT(IJ-2)+0.5D0
                IF(IDQ.EQ.0) IDQ=1
```

```
             Q=QOPT(IJ-1)
              IQ=Q
            ELSE
              IDQ=1
              Q=1.D0
              IQ=1
            END IF
            DQ=IDQ
C GIVE PROGRESS MESSAGE TO TI:
            WRITE(5,601) IJ
 601        FORMAT(' IJ=',I3)
            DEBNJ=START+(IJ-1)*DBINC
            DBSJR(IJ)=DEBNJ
            R=10.D0**(DEBNJ/10.D0)
            DO 602 IQS=1,50
            POFQ(IQS)=0.D0
            IQL(IQS)=0
 602        CONTINUE
            IQS=0
C PRIME THE ALGORITHM WITH DUMMY OLD VALUES OF P(E)
            P1=0.D0
            P2=0.D0
 709        GAMMA=Q/SLOTS
            WRITE(GNAME,735) MM,IQ
 735        FORMAT('EQ',I1,'3',I4.4,'.DAT')
            OPEN(UNIT=3,FILE=GNAME,STATUS='OLD',FORM='UNFORMATTED',
     $           READONLY,ERR=770)
            WRITE(5,3939)
 3939       FORMAT(' READING EVENT FILE')
            READ(3) D,IDSUB,NUSED,GOOD
            CLOSE(UNIT=3)
            GOTO 777
C IF FILE FOR EVENT PROBABILITIES DOES NOT EXIST, CALCULATE THEM
C AND CREATE A FILE.
 770        CONTINUE
            WRITE(5,3938)
 3938       FORMAT(' CREATING EVENT FILE')
            CALL GENPIE(MM,IQ,NSLOTS,GOOD,A,IASUB,
     $                  C,ICSUB,D,IDSUB,NUSED)
            OPEN(UNIT=3,FILE=GNAME,STATUS='NEW',FORM='UNFORMATTED')
            WRITE(3) D,IDSUB,NUSED,GOOD
            CLOSE(UNIT=3)
 777        IF(.NOT.GOOD) GOTO 700
            RHOTS=GAMMA*R*EBNO/(GAMMA*R+EBNO)
            RHOT=K*RHOTS/3.D0
C EVALUATE THE PROBABILITY
            DO 780 NQS=1,IQS
            IF(IQL(NQS).EQ.IQ) THEN
              PESYM=POFQ(NQS)
              GOTO 781
            END IF
 780        CONTINUE
```



```
C NOT IN STORED LIST, COMPUTE IT
            CALL PSUBE(MM,PESYM,D,IDSUB,NUSED,PRERR,IPSUB,PEOO)
            IQS=IQS+1
            IF(IQS.GT.50) THEN
              IQS=50
            ELSE
              POFQ(IQS)=PESYM
              IQL(IQS)=IQ
            END IF
 781        P3=PESYM
            IF(P3.GT.P2 .AND. IQ.LT.NSLOTS) THEN
C KEEP ON GOING, WE ARE NOT PAST THE MAXIMUM
              P1=P2
              P2=P3
              IQ=MINO(IQ+IDQ,NSLOTS)
              Q=DMIN1(Q+DQ,SLOTS)
              GOTO 709
            ELSE
              PMAX=DMAX1(P1,P2,P3)
              EPS=0.001D0*PMAX
              TEST=(DABS(P1-P2).LE.EPS .AND. DABS(P1-P3).LE.EPS .AND.
     $           DABS(P2-P3).LE.EPS)
              IF( TEST .OR. IDQ.EQ.1
     $                 .OR. ((.NOT.TEST) .AND. IQ.EQ.NSLOTS)) THEN
C               WE ARE DONE WHEN ALL 3 ARE CLOSE TOGETHER OR WHEN DQ=1
C               OR WHEN WE REACHED FULL-BAND JAMMING AND P(E) IS STILL
C               INCREASING
                POPT=PMAX
                IF(P2.GT.P3) THEN
C                 THE OPTIMUM MUST BE THE MIDDLE POINT OF THE 3
                  QOPT(IJ)=Q-DQ
                  IF(QOPT(IJ).EQ.0.D0) QOPT(IJ)=1.D0
C PREVENT ROUND-OFF FROM MAKING QOPT VS. EB/NJ NON-MONOTONIC
                  IF(IJ.GT.1) THEN
                    IF(QOPT(IJ).LT.QOPT(IJ-1)) QOPT(IJ)=QOPT(IJ-1)
                  END IF
                ELSE
C                 THE OPTIMUM IS FULL-BAND JAMMING
                  QOPT(IJ)=NSLOTS
                END IF
                GOTO 665
              ELSE
C               NOT LOCATED SUFFICIENTLY ACCURATELY, CUT DQ AND TRY AGAIN
                Q=Q-DQ-DQ
                IQ=IQ-IDQ-IDQ
                IDQ=IDQ/2
                DQ=IDQ
                P2=P1
                P1=0.D0
                Q=Q+DQ
                IQ=IQ+IDQ
```

```
                   GOTO 709
                 END IF
               END IF
      665      PE=WORBIT*POPT
               WRITE(6,666) DBSJR(IJ),PE,QOPT(IJ)
      666      FORMAT(1X,F7.3,5X,1PD12.5,5X,1PD12.5)
               PRLOG(IJ)=DLOG10(PE)
               OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
          $         FORM='UNFORMATTED')
               WRITE(4) DBSJR(IJ), PRLOG(IJ), QOPT(IJ)
               CLOSE(UNIT=4)
      600      CONTINUE
               OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED')
               WRITE(4) MM,3,DEBNOL(IO),NSLOTS,DBSJR,PRLOG,QOPT
               CLOSE(UNIT=4)
               WRITE(6,776) MM,DEBNOL(IO)
               DO 689 IJ=1,NJ
               WRITE(6,666) DBSJR(IJ),10.**PRLOG(IJ),QOPT(IJ)
      689      CONTINUE
               WRITE(6,688) TAU
      688      FORMAT(////' OPTIMUM THRESHOLD FOR ABOVE IS ETA/SIGMA**2 =',
          $             F7.3)
      700      CONTINUE
      800      CONTINUE
      900      CONTINUE
               STOP 0
               END
```



```
               SUBROUTINE GET(NJ,START,DBINC)
      C  INTERACTIVE INPUT OF PARAMETERS FOR RUN
               IMPLICIT DOUBLE PRECISION(A-H,O-Z)
               CHARACTER*9 FIELD,BLANK9
               COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
               COMMON /SIZE/ NO
      C DEFAULT LISTS TEMPORARILY NEEDED ARE IN SHARED STORAGE WITH
      C THE LARGE CONVOLUTION WORKING ARRAYS
               COMMON /SHARE/ DSNR(3,4)
               DATA BLANK9/'         '/
      32       WRITE(5,33)
      33       FORMAT(' BITS/SYMBOL (K) [2]: ',$)
               READ(5,3)K
               IF(K.EQ.0)K=2
               MM=2**K
      1        WRITE(5,2)
      2        FORMAT(' HOW MANY EB/NO? [1]: ',$)
               READ(5,3)NO
      3        FORMAT(I2)
               IF(NO.EQ.0)NO=1
               DO 7 IN=1,NO
               DO=DSNR(IN,K)
      4        WRITE(5,5)IN,DO
      5        FORMAT('   EB/NO(',I2,') [',F9.6,']: ',$)
               READ(5,6)FIELD
      6        FORMAT(A9)
               IF(FIELD.EQ.BLANK9) THEN
                 DEBNOL(IN)=DO
               ELSE
                 DECODE(9,61,FIELD)DEBNOL(IN)
      61         FORMAT(F9.6)
               END IF
      7        CONTINUE
               NSLOTS=2400
      38       WRITE(5,39)
      39       FORMAT(' HOW MANY EB/NJ? [11]: ',$)
               READ(5,34,ERR=38) NJ
      34       FORMAT(I3)
               IF(NJ.EQ.0) NJ=11
               IF(NJ.LT.0 .OR. NJ.GT.11) GOTO 32
      40       WRITE(5,41)
      41       FORMAT(' STARTING VALUE FOR EB/NJ (DB) [50.]: ',$)
               READ(5,42,ERR=40) START
               IF(START.EQ.0.D0) START=50.D0
      42       FORMAT(F6.3)
               IF(NJ.EQ.1) RETURN
      35       WRITE(5,36)
      36       FORMAT(' DB INCREMENT FOR EB/NJ [-5.0]: ',$)
               READ(5,37,ERR=35) DBINC
      37       FORMAT(F6.3)
               IF(DBINC.EQ.0.D0) DBINC=-5.D0
               RETURN
               END
```

```
0001          SUBROUTINE PSUBE(M,PE,D,IDSUB,NUSED,PRERR,IPSUB,PEOO)
        C
        C COMPUTE UNCONDITIONAL ERROR PROBABILITY
        C
0002          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003          INTEGER LUP(4),JSUB(4)
0004          LOGICAL*1 GO,NONE,STORE,RESTRT
        C WE DO WANT TO STORE ZERO ELEMENTS OF THE DENSITY FUNCTION,
        C SINCE IT SAVES TIME TO AVOID REPEATING THE UNDERFLOWS
0005          VIRTUAL PRERR(625),IPSUB(625)
0006          VIRTUAL D(625),IDSUB(625)
0007          COMMON /RSPAR/ JAM(4),JAM1,NPS,IJ,RESTRT
0008          COMMON /SHARE2/ LOW(4),LINC(4)
0009          COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0010          COMMON /PARDEN/ RHON, RHOT
0011          DATA STORE/.TRUE./
0012          PE=0.D0
0013          DO 10 I=1,M
0014          LUP(I)=3
0015          JSUB(I)=0
0016    10    CONTINUE
0017          IF(.NOT.RESTRT) THEN
0018            NPS=0
        C       THE ALL-ZERO JAMMING EVENT P(E) IS AVAILABLE FROM THE
        C       SEARCH FOR THE OPTIMUM THRESHOLD, SO PUT IT INTO THE
        C       ARRAY OF SAVED VALUES
0019            CALL LOCN(M,LOW,LUP,JSUB,ISUB)
0020            CALL PUTIN(PEOO,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
0021            IF(KODE.NE.0) STOP 'PRERR FULL'
0022            JAM1=-1
0023          ELSE
        C       RESTART BYPASSES VECTOR LOOP INITIALIZATION CALL
0024            GOTO 100
0025          END IF
        C START VECTOR-INDEXED LOOP ON JAMMING EVENTS
0026          CALL VLINIT(JAM,LOW,M)
0027    100   CONTINUE
0028          IF(JAM1.NE.JAM(1)) THEN
        C UPDATE TEST VALUE FOR NEXT TIME, AND ...
0029            JAM1=JAM(1)
0030          END IF
0031          CALL EVENT(M,JAM,PIE,D,IDSUB,NUSED)
        C BYPASS CONDITIONAL ERROR PROBABILITY CALCULATION IF EVENT
        C PROBABILITY IS ZERO.  THIS SAVES MUCH TIME.
0032          IF(PIE.EQ.0.D0)GOTO 101
        C SINCE JAMMING PROBABILITIES DEPEND ONLY ON NO. OF CHANNELS
        C HAVING JAM(I) HOPS JAMMED AND NOT THE ARRANGEMENT OF THE
        C CHANNELS, WE CAN SORT THE NON-SIGNAL CHANNELS INTO ASCENDING
        C NUMBERS OF HOPS JAMMED.  THIS REDUCES NUMBER OF DISTINCT
        C CONDITIONAL ERROR PROBABILITIES WHICH MUST BE SAVED TO AVOID
        C RECOMPUTING THEM UNNECESSARILY.
```



```
0033          DO 111 I=1,M
0034          JSUB(I)=JAM(I)
0035    111   CONTINUE
0036          IF(M.EQ.2) GOTO 199
0037          DO 110 I=2,M-1
0038          DO 120 J=I+1,M
0039          IF(JSUB(J).LT.JSUB(I)) THEN
0040            JTEMP=JSUB(I)
0041            JSUB(I)=JSUB(J)
0042            JSUB(J)=JTEMP
0043          END IF
0044    120   CONTINUE
0045    110   CONTINUE
0046    199   CONTINUE
        C STORE AS ELEMENT OF A SPARSE ARRAY TO SAVE MEMORY
        C EVEN THOUGH WE STORE ZEROS, THE SORTING OF SUBSCRIPTS
        C CUTS OUT MANY ELEMENTS.
0047          CALL LOCN(M,LOW,LUP,JSUB,ISUB)
        C TRY TO FIND CONDITIONAL ERROR PROBABILITY IN STORED ARRAY
0048          CALL LOOKUP(PROB,PRERR,IPSUB,NPS,625,ISUB,STORE,NONE)
        C IF IT IS NOT THERE, WE MUST COMPUTE IT
0049          IF(NONE) THEN
0050            CALL PSEL(JSUB,M,PROB)
        C  ... AND SAVE IT FOR POSSIBLE FUTURE RE-USE
0051            CALL PUTIN(PROB,PRERR,IPSUB,NPS,625,ISUB,KODE,STORE)
0052            IF(KODE.NE.0) STOP 2
0053          END IF
        C SUM UP UNCONDITIONAL ERROR PROBABILITY
0054          PE=PE+PIE*PROB
        C
        C SAVE WHAT WE HAVE SO FAR FOR RESTART CAPABILITY
        C
0055          OPEN(UNIT=4,FILE='RESUME.DAT',STATUS='UNKNOWN',
           $      FORM='UNFORMATTED')
0056          WRITE(4) JAM,JAM1,PRERR,NPS,IPSUB,IJ
0057          CLOSE(UNIT=4)
        C
        C ONCE WE HAVE UPDATED RESTART INFORMATION, THE RESTART
        C FLAG BECOMES FALSE.
        C
0058          RESTRT=.FALSE.
        C ITERATE THE VECTOR-INDEX LOOP
0059    101   CALL VLITER(JAM,LOW,LUP,LINC,M,GO)
0060          IF(GO) GOTO 100
0061          RETURN
0062          END
```

```
      SUBROUTINE EVENT(M,JAM,PIE,D,IDSUB,NUSED)
C
C SUBROUTINE TO LOOK UP EVENT PROBABILITY FROM STORED ARRAY
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      LOGICAL*1 STORE,NONE
      DIMENSION JAM(4),LUP(4)
      VIRTUAL D(625),IDSUB(625)
      COMMON /SHARE2/ LOW(4),LINC(4)
      DATA STORE/.FALSE./
C SET UP ARRAY DESCRIPTION D(0:LL,...,0:LL) WITH M DIMENSIONS
      DO 1 I=1,M
      LUP(I)=3
1     CONTINUE
C COMPUTE LINEAR EQUIVALENT SUBSCRIPT FOR JAMMING EVENT
      CALL LOCN(M,LOW,LUP,JAM,ISUB)
C LOOK UP THE VALUE, GET 0.D0 IF NOT THERE
      CALL LOOKUP(PIE,D,IDSUB,NUSED,625,ISUB,STORE,NONE)
      RETURN
      END
```



```
      SUBROUTINE GENPIE(MM,NQ,NSLOTS,GOOD,A,IASUB,C,ICSUB,
     $                  D,IDSUB,NUSED)
C
C SUBROUTINE TO GENERATE EVENT PROBABILITIES
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL*1 GO,GO2,STORE,NONE,GOOD
      DIMENSION LUP2(4),LUP3(4)
      DIMENSION IUPA(4)
      DIMENSION IUPD(4)
      DIMENSION LUP1(4)
      VIRTUAL A(100),IASUB(100),C(625),ICSUB(625),
     $        D(625),IDSUB(625)
      DIMENSION I(4),II(4),III(4)
C SHARED STORAGE FOR COMMONLY NEEDED CONSTANT ARRAYS
      COMMON /SHARE2/ LOW(4),LINC(4)
C SHARED STORATE FOR: (1) INPUT DEFAULT LISTS, (2) CONDITIONAL PROB GEN.,
C AND (3) EVENT PROB. GEN.   THESE ARE NON-OVERLAPPING USAGES.
      COMMON /SHARE/ LUP2,LUP3,IUPD,L1,NUSEA,NUSEC,IERR,
     $               ISUB,ISUB1,ISUB2,AIN,I,II,III,NN,AOUT,CIN,COUT,
     $               DOUT,DIN
      DATA I100/100/
      DATA IUPA/4*1/
      DATA LUP1/4*1/
C STORE=.FALSE. => DON'T STORE ZERO ELEMENTS OF SPARSE ARRAY.
      STORE=.FALSE.
      GOOD=.TRUE.
9999  IF(NQ.LE.0) THEN
        GOOD=.FALSE.
        RETURN
      END IF
      DO 80 L1=1,MM
      IUPD(L1)=3
80    CONTINUE
C JAMMING PATTERN W/NON-ZERO PROBABILITY ON PER-HOP BASIS
      NUSEA=0
C INITIALIZE VECTOR-INDEX LOOP
      CALL VLINIT(I,LOW,MM)
90    CONTINUE
      CALL LOCN(MM,LOW,IUPA,I,ISUB)
      CALL PRIHOP(I,MM,NQ,NSLOTS,AIN)
      CALL PUTIN(AIN,A,IASUB,NUSEA,I100,ISUB,IERR,STORE)
      IF(IERR.NE.0)STOP 3
C ITERATE VECTOR-INDEX LOOP
      CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
      IF(GO) GOTO 90
C COMPUTATION STARTS HERE. FIRST COPY A INTO D.
C SINCE ARRAYS ARE A(0:1,0:1,...,0:1) AND D(0:L,0:L,...,0:L)
C THE COPYING MUST BE DONE ON BASIS OF EQUIVALENT LINEAR
C SUBSCRIPTS RATHER THAN A SIMPLE MOVE OPERATION.
      NUSED=0
```

```
      C INITIALIZE VECTOR-INDEX LOOP
            CALL VLINIT(I,LOW,MM)
   99       CONTINUE
            CALL LOCN(MM,LOW,IUPA,I,ISUB1)
            CALL LOCN(MM,LOW,IUPD,I,ISUB2)
            CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
            CALL PUTIN(AOUT,D,IDSUB,NUSED,625,ISUB2,IERR,STORE)
            CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
            IF(GO)GOTO 99
      C ... L-1 CONVOLUTIONS ARE NEEDED ...
            DO 9998 L1=1,2
    9       DO I25 NN=1,MM
            LUP2(NN)=L1
  125       LUP3(NN)=L1+1
            NUSEC=0
            CALL VLINIT(I,LOW,MM)
   98       CONTINUE
            CALL VLINIT(II,LOW,MM)
   97       CONTINUE
            CALL LOCN(MM,LOW,IUPA,I,ISUB1)
            CALL LOCN(MM,LOW,IUPD,II,ISUB2)
            DO 21 NN=1,MM
            III(NN)=I(NN)+II(NN)
   21       CONTINUE
            CALL LOCN(MM,LOW,IUPD,III,ISUB3)
            CALL LOOKUP(AOUT,A,IASUB,NUSEA,I100,ISUB1,STORE,NONE)
            CALL LOOKUP(DOUT,D,IDSUB,NUSED,625,ISUB2,STORE,NONE)
            CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB3,STORE,NONE)
            CIN=COUT+AOUT*DOUT
            CALL PUTIN(CIN,C,ICSUB,NUSEC,625,ISUB3,IERR,STORE)
            IF(IERR.NE.0) STOP 4
      C ITERATE VECTOR-LOOP FOR ARRAY D
            CALL VLITER(II,LOW,LUP2,LINC,MM,GO2)
            IF(GO2) GOTO 97
      C ITERATE VECTOR-LOOP FOR ARRAY A
            CALL VLITER(I,LOW,LUP1,LINC,MM,GO)
            IF(GO) GOTO 98
      C COPY C TO D IN SORTED ORDER FOR NEXT ITERATION
            NUSED=0
            CALL VLINIT(II,LOW,MM)
   96       CONTINUE
            CALL LOCN(MM,LOW,IUPD,II,ISUB)
            CALL LOOKUP(COUT,C,ICSUB,NUSEC,625,ISUB,STORE,NONE)
            DIN=COUT
            CALL PUTIN(DIN,D,IDSUB,NUSED,625,ISUB,IERR,STORE)
            IF(IERR.NE.0) STOP 5
            CALL VLITER(II,LOW,LUP3,LINC,MM,GO)
            IF(GO) GOTO 96
 9998       CONTINUE
            RETURN
            END
```



```
            SUBROUTINE PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
      C
      C THIS SUBROUTINE INSERTS AN ELEMENT INTO A SPARSE ARRAY FOR
      C WHICH ONLY THE NON-ZERO ELEMENTS ARE KEPT IN STORAGE IF
      C THE SWITCH STORE IS .TRUE.
      C
      C THE DOUBLE PRECISION VALUE CIN IS STORED AS C(K) WHERE
      C THE AUXILIARY ARRAY ICSUB IS USED TO KEEP TRACK OF THE
      C SUBSCRIPTS OF THE CORRESPONDING ENTRIES IN C.
      C
      C USAGE:
      C       LOGICAL*1 STORE
      C       DOUBLE PRECISION C,CIN
      C       VIRTUAL ICSUB(NMAX),C(NMAX)
      C       CALL PUTIN(CIN,C,ICSUB,NUSE,NMAX,K,IERR,STORE)
      C WHERE
      C  CIN = VALUE OF ELEMENT TO STORE
      C  C = ARRAY IN WHICH NON-ZERO VALUES ARE ACTUALLY STORED
      C  ICSUB = AUXILIARY ARRAY FOR ACTUAL SUBSCRIPT VALUES
      C  NUSE = NUMBER OF ELEMENTS OF C CURRENTLY OCCUPIED
      C  NMAX = SIZE OF ARRAY C
      C  IERR = ERROR RETURN CODE, 0 IF NO ERROR OR 1 IF THERE IS
      C      NO ROOM AVAILABLE IN C
      C  STORE = .TRUE. TO STORE ZEROES EXPLICITLY, ELSE .FALSE.
      C NOTE: IF CIN=0 AND THE SUBSCRIPT K IS FOUND IN ICSUB, THEN
      C       THE ELEMENT IS DELETED BY SHIFTING DOWNWARD ALL
      C       FOLLOWING ELEMENTS OF THE ARRAY
      C
      C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            VIRTUAL ICSUB(NMAX),C(NMAX)
            LOGICAL*1 STORE
            IERR=0
            IF(STORE) GOTO 5
            IF(CIN.EQ.0.D0) GOTO 30
    5       IF(NUSE.EQ.0)GOTO 20
            DO 10 I=1,NUSE
            IF(ICSUB(I).NE.K) GOTO 10
            C(I)=CIN
            RETURN
   10       CONTINUE
            IF(NUSE.LT.NMAX) GOTO 20
            IERR=1
            RETURN
   20       NUSE=NUSE+1
            ICSUB(NUSE)=K
            C(NUSE)=CIN
            RETURN
   30       DO 40 I=1,NUSE
            J=I
            IF(ICSUB(I).EQ.K) GOTO 50
   40       CONTINUE
```

```
0025             RETURN
           C
           C REMOVE THE ZEROED ELEMENT AND BUMP COUNT OF ENTRIES USED
           C
0026       50    DO 60 I=J,NUSE-1
0027             ICSUB(I)=ICSUB(I+1)
0028             C(I)=C(I+1)
0029       60    CONTINUE
0030             NUSE=NUSE-1
0031             RETURN
0032             END
```



```
0001             SUBROUTINE LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
           C
           C THIS SUBROUTINE RETRIEVES AN ELEMENT OF A SPARSE ARRAY WHICH
           C HAS BEEN STORED COMPACTLY BY STORING ONLY NON-ZERO ELEMENTS.
           C
           C THE ARRAY IS DOUBLE PRECISION.
           C
           C USAGE:
           C       VIRTUAL ICSUB(NMAX), C(NMAX)
           C       LOGICAL*1 STORE, NONE
           C       DOUBLE PRECISION COUT
           C       CALL LOOKUP(COUT,C,ICSUB,N,NMAX,K,STORE,NONE)
           C WHERE
           C       COUT = VALUE OF C(K) (OUTPUT FROM SUBROUTINE)
           C          C = ARRAY USED TO STORE NON-ZERO ELEMENTS
           C      ICSUB = AUXILIARY ARRAY TO STORE ACTUAL SUBSCRIPTS
           C          N = NUMBER OF ELEMENTS OF C CURRENTLY IN USE
           C       NMAX = SIZE OF C
           C          K = SUBSCRIPT OF SPARSE ARRAY TO LOOK UP
           C      STORE = .TRUE. IF ZEROES STORED EXPLICITLY, ELSE .FALSE.
           C       NONE = .FALSE. IF ZEROES NOT STORED OR ZEROES STORED AND
           C                      ELEMENT IS FOUND IN THE STORED ARRAY
           C              .TRUE.  IF ZEROES ARE STORED AND THE ELEMENT IS
           C              NOT FOUND (OUTPUT QUANTITY)
           C
           C PROGRAMMER: ROBERT H. FRENCH
           C       DATE: 11 JANUARY 1984
           C
0002             IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003             VIRTUAL ICSUB(NMAX),C(NMAX)
0004             LOGICAL*1 STORE, NONE
0005             NONE=.FALSE.
0006             DO 10 I=1,N
0007             IF(ICSUB(I).NE.K)GOTO 10
0008             COUT=C(I)
0009             RETURN
0010       10    CONTINUE
0011             IF(STORE) THEN
0012               NONE=.TRUE.
0013             ELSE
0014               COUT=0.
0015             END IF
0016             RETURN
0017             END
```

```
0001            SUBROUTINE LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
          C
          C THIS SUBROUTINE COMPUTES THE EQUIVALENT LINEAR SUBSCRIPT FOR
          C A MULTIDIMENSIONAL ARRAY OF NDIM DIMENSIONS
          C
          C IF THE ARRAY A IS DEFINED AS
          C       DIMENSION A(ILOW(1):IUP(1),...,ILOW(NDIM):IUP(NDIM))
          C AND ISUB(1),...,ISUB(NDIM) IS A SET OF SUBSCRIPTS FOR A,
          C THEN THIS SUBROUTINE RETURNS IN LINEAR THE OFFSET FROM THE
          C ORIGIN OF A TO THE ELEMENT A(ISUB(1),...,ISUB(NDIM)), ASSUMING
          C THE FIRST SUBSCRIPT VARIES MOST RAPIDLY.
          C
          C USAGE:
          C       DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
          C       DATA ILOW/lower limits of defined subscripts of array/
          C       DATA IUP/upper limits of defined subscripts of array/
          C       ...SET ISUB TO DESIRED SEQUENCE OF SUBSCRIPTS...
          C       CALL LOCN(NDIM,ILOW,IUP,ISUB,LINEAR)
          C WHERE
          C       NDIM = NUMBER OF DIMENSIONS THE ARRAY HAS
          C       ILOW = ARRAY OF LOWER SUBSCRIPT BOUNDS
          C        IUP = ARRAY OF UPPER SUBSCRIPT BOUNDS
          C       ISUB = ARRAY CONTAINING SUBSCRIPT FOR WHICH LOCATION IS
          C              TO BE COMPUTED
          C     LINEAR = RETURNED VALUE OF OFFSET INTO ARRAY IN MEMORY
          C
          C
          C PROGRAMMER: ROBERT H. FRENCH
          C       DATE: 11 JANUARY 1984
          C
0002            DIMENSION ILOW(NDIM),IUP(NDIM),ISUB(NDIM)
0003            LINEAR=0
0004            DO 10 I=1,NDIM-1
0005            J=NDIM-I+1
0006            LINEAR=(LINEAR+(ISUB(J)-ILOW(J)))*(IUP(J-1)-ILOW(J-1)+1)
0007      10    CONTINUE
0008            LINEAR=LINEAR+ISUB(1)-ILOW(1)
0009            RETURN
0010            END
```



```
0001            SUBROUTINE VLINIT(LVEC,LLOW,LMAX)
          C
          C THIS SUBROUTINE INITIALIZES A "VECTOR DO-LOOP" STRUCTURE
          C DEFINED BY THE FOLLOWING PSEUDO-FORTRAN CODE:
          C       DO 100 LVEC(1)=LLOW(1),LUP(1),LINC(1)
          C       DO 100 LVEC(2)=LLOW(2),LUP(2),LINC(2)
          C                  :
          C                  :
          C       DO 100 LVEC(LMAX)=LLOW(LMAX),LUP(LMAX),LINC(LMAX)
          C                  :
          C                  :
          C       (STATEMENTS IN RANGE OF LOOP)
          C 100   CONTINUE
          C
          C THE COMPANION ROUTINE VLITER HANDLES THE LOOP CONTROL AT THE
          C CONTINUE STATEMENT IN THE ABOVE STRUCTURE
          C
          C USAGE:
          C       LOGICAL*1 GO
          C       DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
          C (INITIALIZE ARRAY LLOW TO STARTING VALUES OF THE NESTED LOOPS)
          C (INITIALIZE ARRAY LUP TO STOPING VALUES OF THE NESTED LOOPS)
          C (INITIALIZE ARRAY LINC TO INCREMENTS OF THE LOOPS)
          C       CALL VLINIT(LVEC,LLOW,LMAX)
          C   100 CONTINUE
          C
          C         . (STATEMENTS IN RANGE OF LOOPS)
          C
          C       CALL VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
          C       IF(GO)GOTO 100
          C WHERE
          C    LVEC = ARRAY FOR STORAGE OF LOOP INDICES. LVEC(1) IS THE
          C           OUTER-MOST LOOP; LVEC(LMAX), THE INNER-MOST LOOP.
          C    LLOW = ARRAY FOR STORAGE OF LOOP STARTING VALUES, IN SAME
          C           SEQUENCE AS LVEC
          C    LUP = ARRAY FOR STORAGE OF LOOP FINAL VALUES, IN SAME
          C          SEQUENCE AS LVEC
          C    LINC = ARRAY FOR STORAGE OF LOOP INCREMENTS, IN SAME
          C           SEQUENCE AS LVEC
          C    LMAX = NUMBER OF LOOPS NESTED
          C    GO = LOGICAL VARIABLE, .TRUE. IF JUMP BACK TO BEGINNING OF
          C         STATEMENTS IN THE RANGE OF THE LOOP SHOULD OCCUR,
          C         .FALSE. OTHERWISE (I.E. OUTER-MOST LOOP TERMINATED)
          C
          C PROGRAMMER: ROBERT H. FRENCH          DATE: 11 JANUARY 1984
          C
0002            DIMENSION LVEC(LMAX),LLOW(LMAX)
0003            DO 1 N=1,LMAX
0004            LVEC(N)=LLOW(N)
0005      1     CONTINUE
0006            RETURN
0007            END
```

```
      SUBROUTINE VLITER(LVEC,LLOW,LUP,LINC,LMAX,GO)
C
C LOOP ITERATION LOGIC FOR A "VECTOR DO-LOOP"
C
C SEE DETAILED COMMENTS IN SUBROUTINE VLINIT FOR USAGE AND
C PARAMETER DEFINITIONS
C
C PROGRAMMER: ROBERT H. FRENCH
C       DATE: 11 JANUARY 1984
C
C
      LOGICAL*1 GO
      DIMENSION LVEC(LMAX),LLOW(LMAX),LUP(LMAX),LINC(LMAX)
      GO=.TRUE.
      DO 100 NDX=1,LMAX
      NSUB=LMAX+1-NDX
      LVEC(NSUB)=LVEC(NSUB)+LINC(NSUB)
      IF((LINC(NSUB).GE.0.AND.LVEC(NSUB).LE.LUP(NSUB))
     $  .OR.(LINC(NSUB).LT.0.AND.LVEC(NSUB).GE.LUP(NSUB))) RETURN
      LVEC(NSUB)=LLOW(NSUB)
100   CONTINUE
      GO=.FALSE.
      RETURN
      END
```



```
      SUBROUTINE PR1HOP(I,KM,KQ,KN,AIN)
C
C THIS SUBROUTINE COMPUTES THE EVENT PROBABILITY FOR ALL
C POSSIBLE JAMMING PATTERNS WITH NON-ZERO PROBABILITY FOR
C L=1 HOP/SYMBOL FOR RMFSK/FH IN PBNJ
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION I(4)
      AIN=0.D0
      KJAM=0
      DO 1 K=1,KM
      KJAM=KJAM+I(K)
    1 CONTINUE
C IF THIS IS AN IMPOSSIBLE CASE, RETURN WITH RESULT = 0.0
      IF(KJAM.GT.MIN0(KQ,KM)) RETURN
      KPMAX=KJAM-1
      LPMAX=KM-KJAM-1
      JPMAX=KM-1
      IMAX=MAX0(KPMAX,LPMAX,JPMAX)
      PROD=1.D0
      Q=KQ
      DIFFNQ=KN-KQ
      EN=KN
      DO 100 LOOP=0,IMAX
      F=LOOP
      IF(LOOP.LE.KPMAX) PROD=PROD*(Q-F)
      IF(LOOP.LE.JPMAX) PROD=PROD/(EN-F)
      IF(LOOP.LE.LPMAX) PROD=PROD*(DIFFNQ-F)
  100 CONTINUE
      AIN=PROD
      RETURN
      END
```

```
0001            BLOCK DATA
          C INITIALIZE SHARED CONSTANTS
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            COMMON /SHARE/ DSNR(3,4)
0004            COMMON /SHARE2/ LOW(4),LINC(4)
0005            COMMON /WTS/ X(5),W(5)
          C
          C WEIGHTS AND ABSCISSAS FOR 10-POINT GAUSSIAN QUADRATURE
          C
0006            DATA X/ 0.148874338981631D0,
             $          0.433395394129247D0,
             $          0.679409568299024D0,
             $          0.865063366688985D0,
             $          0.973906528517172D0 /
0007            DATA W/ 0.295524224714753D0,
             $          0.269266719309996D0,
             $          0.219086362515982D0,
             $          0.149451349150581D0,
             $          0.066671344308688D0 /
          C DEFAULT LISTS FOR INTERACTIVE PARAMETER INPUTS
          C ARE SHARED WITH LARGE WORKING STORAGE ARRAYS SINCE THEY
          C MAY BE DESTROYED ONCE THE INPUT PARAMETERS ARE SET UP
0008            DATA DSNR /13.35247D0, 12.3133D0, 10.94443D0,
             $            10.606572D0, 9.6284D0, 8.35245D0,
             $             9.09401D0, 8.1690D0, 6.971995D0,
             $             8.07835D0, 7.1996D0, 6.069646D0/
          C FREQUENTLY NEEDED CONSTANT ARRAYS AND SCALARS
0009            DATA LOW/4*0/,LINC/4*1/
0010            END
```



```
0001            SUBROUTINE ADQUAD(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
          C
          C ADAPTIVE QUADRATURE ALGORITHM
          C   XL - LOWER LIMIT OF INTEGRAL (IN)
          C   XU - UPPER LIMIT OF INTEGRAL (IN)
          C    Y - VALUE OF INTEGRAL (OUT)
          C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
          C               WITH CALLING SEQUENCE
          C               CALL QR(XL,XU,F,Y)
          C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
          C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
          C WORK - WORK ARRAY OF SIZE N (IN)
          C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
          C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
          C        SAME ARRAY AS WORK (IN)
          C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
          C KODE - ERROR INDICATOR (OUT)
          C          0 -- NO ERROR
          C          1 -- WORK ARRAYS TOO SMALL
          C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
          C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
          C               ATTAINING REQUIRED ACCURACY
          C R. H. FRENCH, 14 AUGUST 1984
          C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL F
0004            DIMENSION WORK(N),STACK(N),HEAP(N)
0005            KODE=0
0006            Y=0.D0
0007            WORK(1)=XU
0008            CALL QR(XL,XU,F,T)
0009            HEAP(1)=T
0010            A=XL
0011            NPTS=1
0012            EPS=TOL
0013            STACK(1)=EPS
0014      10    B=WORK(NPTS)
0015            XM=(A+B)*0.5D0
0016            CALL QR(A,XM,F,P1)
0017            CALL QR(XM,B,F,P2)
0018            IF(DABS(T-P1-P2).LE.EPS) GOTO 20
          C SPLIT IT
0019            NPTS=NPTS+1
0020            IF(NPTS.GT.N) THEN
0021              KODE=1
0022              RETURN
0023            END IF
0024            WORK(NPTS)=XM
0025            HEAP(NPTS)=P2
0026            T=P1
0027            EPS=EPS/2.D0
```

```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```



```
      SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C               WITH CALLING SEQUENCE
C               CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
C        SAME ARRAY AS WORK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=EPS/2.D0
```

```
        IF(EPS.EQ.0.D0) THEN
          KODE=2
          RETURN
        END IF
        STACK(NPTS)=EPS
        GOTO 10
C FINISHED A PIECE
20      Y=Y+P1+P2
        EPS=STACK(NPTS)
        T=HEAP(NPTS)
        NPTS=NPTS-1
        A=B
        IF(NPTS.EQ.0) RETURN
        GOTO 10
        END
```



```
        SUBROUTINE DG10(A,B,F,ANSWER)
C
C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
C
C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
C
C R. H. FRENCH, 28 FEBRUARY 1986
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        COMMON /WTS/ X(5),W(5)
        ANSWER=0.D0
        BMA02=(B-A)/2.D0
        BPA02=(B+A)/2.D0
        DO 10 I=1,5
        C=X(I)*BMA02
        Y1=BPA02+C
        Y2=BPA02-C
        ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
10      CONTINUE
        ANSWER=ANSWER*BMA02
        RETURN
        END
```

```
      SUBROUTINE ADQUA3(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPT'VE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C              WITH CALLING SEQUENCE
C              CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WOR* ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C        SAME ARRAY AS WORK (IN)
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
C *** MAKE IT A RELATIVE TEST FOR THIS INTEGRAL ***
      IF(DABS(T-P1-P2).LE.DABS(T*EPS)) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=EPS/2.D0
```



```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```

```
0001          SUBROUTINE DGTEN(A,B,F,ANSWER)
        C
        C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
        C
        C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
        C
        C R. H. FRENCH, 28 FEBRUARY 1986
        C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          COMMON /WTS/ X(5),W(5)
0004          ANSWER=0.D0
0005          BMAO2=(B-A)/2.D0
0006          BPAO2=(B+A)/2.D0
0007          DO 10 I=1,5
0008          C=X(I)*BMAO2
0009          Y1=BPAO2+C
0010          Y2=BPAO2-C
0011          ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0012    10    CONTINUE
0013          ANSWER=ANSWER*BMAO2
0014          RETURN
0015          END
```



```
0001          SUBROUTINE DGX(A,B,F,ANSWER)
        C
        C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
        C
        C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
        C
        C R. H. FRENCH, 28 FEBRUARY 1986
        C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          COMMON /WTS/ X(5),W(5)
0004          ANSWER=0.D0
0005          BMAO2=(B-A)/2.D0
0006          BPAO2=(B+A)/2.D0
0007          DO 10 I=1,5
0008          C=X(I)*BMAO2
0009          Y1=BPAO2+C
0010          Y2=BPAO2-C
0011          ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0012    10    CONTINUE
0013          ANSWER=ANSWER*BMAO2
0014          RETURN
0015          END
```

```
            SUBROUTINE MINSER(F,FMIN,XMIN,STEP,GUESS,BLIM,ULIM,TOL)
      C
      C SEARCH FOR MINIMUM OF F(X) OVER THE INTERVAL BLIM <= X <= ULIM
      C
      C TROUBLE MAY OCCUR IF F(X) HAS MULTIPLE LOCAL MINIMA WITHIN THE
      C SEARCH INTERVAL OR IF THE FUNCTION IS VERY STEEP AND STEP IS
      C TOO BIG.
      C
      C            F = NAME OF FUNCTION TO BE MINIMIZED
      C         FMIN = MINIMUM VALUE OF F(X) OVER INTERVAL
      C         XMIN = VALUE OF X FOR WHICH FMIN OCCURS
      C         STEP = INITIAL STEP SIZE FOR SEARCH
      C        GUESS = INITIAL GUESS AT XMIN, BLIM <= GUESS <= ULIM
      C         BLIM = LOWER LIMIT OF SEARCH INTERVAL
      C         ULIM = UPPER LIMIT OF SEARCH INTERVAL
      C          TOL = TOLERANCE ON XMIN; SEARCH STOPS WHEN DX < TOL
      C
      C NOTE: F MUST BE A DOUBLE PRECISION FUNCTION OF ONE DOUBLE PRECISION
      C       ARGUMENT.  ANY PARAMETERS CAN BE PASSED FROM THE CALLER VIA
      C       A COMMON BLOCK.
      C
      C PROGRAMMER: ROBERT H. FRENCH         DATE: 17 MARCH 1986
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            X=GUESS
            SLMAX=DABS(X-BLIM)
            SUMAX=DABS(ULIM-X)
            DX=DMIN1(STEP,SLMAX,SUMAX)
            TEST=TOL
         10 FO=F(X)
            F1=F(X+DX)
      C ARE WE GOING IN THE RIGHT DIRECTION?
            IF(F1.LE.FO) GOTO 100
      C ... NO, SWITCH DIRECTION
            DX=-DX
            F1=F(X+DX)
            IF(F1.LE.FO) GOTO 100
      C ELSE WE MUST BE CLOSE TO A MIN. AT X=GUESS, SO CUT
      C STEP SIZE AND TRY AGAIN
            DX=DX/10.DO
            IF(DABS(DX).GE.TEST) GOTO 10
      C CLOSE ENOUGH AT GUESS
         12 XMIN=X
            FMIN=FO
            RETURN
      C
      C NOW GOING RIGHT DIRECTION.
      C KEEP GOING UNTIL PAST MINIMUM BY ONE STEP.
      C
        100 X2=X+DX+DX
      C HAVE WE REACHED END POINT?
            IF(X2.GT.ULIM .OR. X2.LT.BLIM) GOTO 110
```



```
      C ALL OK
        105 F2=F(X2)
      C PAST MIN?
            IF(F2.GE.F1) GOTO 200
      C ...NO, STEP AGAIN
            FO=F1
            F1=F2
            X=X+DX
            GOTO 100
      C MIN MAY BE AT AN ENDPOINT.  CUT STEP SIZE AND TRY AGAIN
      C IF INCREMENT NOT TOO SMALL.
        110 IF(DABS(DX).LE.TEST) GOTO 120
            X=X+DX
            FO=F1
        115 DX=DX/10.DO
            F1=F(X+DX)
            GOTO 100
      C MIN MUST BE AT THE ENDPOINT (OR WITHIN MINIMUM DX THEREOF)
        120 IF(X2.LE.BLIM) GOTO 122
      C MIN AT X=ULIM
            XMIN=ULIM
        121 FMIN=F(XMIN)
            RETURN
      C MIN AT BLIM
        122 XMIN=BLIM
            GOTO 121
      C HAVE PASSED MIN.  IS IT LOCATED CLOSELY ENOUGH YET?
        200 IF(DABS(DX).LE.TEST) GOTO 300
      C ... NO, CUT STEP SIZE AND TRY AGAIN
            GOTO 115
      C DONE!
      C SINCE FO >= F1 & F2>= F1 AND ABS(DX)<MIN. DX, CALL F1 THE MIN.
        300 FMIN=F1
            XMIN=X+DX
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:45:25    16-Jul-86            Page 31
CLIPL3SGR.FTN;6          /F77/WR

            SUBROUTINE PSEL(JSUB,MM,PROB)
      C
      C RANDOM MFSK/FH IN PARTIAL BAND NOISE JAMMING,
      C GIVEN A JAMMING EVENT, WITH CLIPPER RECEIVER
      C
      C        JSUB - JAMMING EVENT VECTOR
      C          LL - NUMBER OF HOPS/SYMBOL
      C          MM - ALPHABET SIZE
      C        PROB - RESULTING CONDITIONAL ERROR PROBABILITY
      C
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            DIMENSION JSUB(MM), WORK(30), STACK(30), HEAP(30)
            EXTERNAL DG10, PGRAND
            INTEGER NCHAN(0:3)
            COMMON /JAMCNT/ NCHAN
            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
         $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
            COMMON /PARDEN/ RHON, RHOT
            COMMON /QUES/ Q0, Q1
            COMMON /SCJAM/ JAMSC
            KSUB=0
            DO 6 I=1,MM
            KSUB=KSUB+JSUB(I)
        6   CONTINUE
      C SET UP VALUES WHICH WILL REMAIN IF THIS IS THE NOTHING-JAMMED CASE
            Q0=Q(2.D0*DSQRT(0.5D0*RHON),DSQRT(2.D0*TAU))
            Q1=1.D0
            TAUK=0.D0
      C IF ANYTHING IS JAMMED, SET UP JAMMING-RELATED QUANTITIES
            IF(KSUB.NE.0) THEN
              BIGK=RHON/RHOT
              BK1=BIGK-1.D0
              AAB=BIGK/BK1
              AAB2=AAB*AAB
              BAB=1.D0/BK1
              BAB2=BAB*BAB
              AABBAB=AAB*BAB
              TAUK=TAU/BIGK
              TAUK2=TAU2/BIGK
              TAUK3=TAU3/BIGK
              Q1=Q(2.D0*DSQRT(0.5D0*RHOT),DSQRT(2.D0*TAUK))
            END IF
      C
      C COUNT NUMBER OF NONSIGNAL CHANNELS WITH Lm HOPS JAMMED
      C
            DO 10 I=0,3
            NCHAN(I)=0
       10   CONTINUE
            DO 11 I=2,MM
            KSUB=JSUB(I)
            NCHAN(KSUB)=NCHAN(KSUB)+1
       11   CONTINUE
            JAMSC=JSUB(1)
```

```
PDP-11 FORTRAN-77 V4.0-1        10:45:25    16-Jul-86            Page 32
CLIPL3SGR.FTN;6          /F77/WR

      C
      C DO THE TIE PART OF THE DENSITY
      C
            CALL TIES(JSUB,MM,PTIE)
      C
      C DO THE CONTINUOUS PART OF THE DENSITY IN SECTIONS
      C
            PCONT=0.D0
            DO 13 ISECT=1,3
            XL=(ISECT-1)*TAU
            XU=ISECT*TAU
            CALL ADQUAD(XL,XU,CHUNK,DG10,PGRAND,1.D-8,WORK,
         $              STACK,HEAP,30,KODE)
            CALL TEST2(KODE,10)
            PCONT=PCONT+CHUNK
       13   CONTINUE
      C
      C PUT THEM TOGETHER
      C
            PROB=1.D0-PCONT-PTIE
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:45:30    16-Jul-86            Page 33
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION PGRAND(BETA)
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            INTEGER NCHAN(0:3)
0004            COMMON /SCJAM/ JAMSC
0005            COMMON /JAMCNT/ NCHAN
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ QO, Q1
0009            PROD=1.DO
0010            DO 10 I=0,3
0011            IF(NCHAN(I).NE.0) THEN
0012              LJAM=I
0013              X=GL(BETA)
0014              PROD=PROD*DXI(X,NCHAN(I))
0015            END IF
0016       10   CONTINUE
0017            LJAM=JAMSC
0018            Y=PZ1(BETA)
0019            PGRAND=Y*PROD
0020            RETURN
0021            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:45:33    16-Jul-86            Page 34
CLIPL3SGR.FTN;6          /F77/WR

0001            SUBROUTINE TEST(ID)
        C
        C TEST RETURN CODE FROM BESSEL FUNCTION
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            IF(KODE.EQ.0) RETURN
0005            WRITE(5,1) KODE, ID
0006        1   FORMAT(' BESSEL FUNCTION CODE = ',I2,' FROM CALL NUMBER ',I5)
0007            STOP 'FATAL ERROR'
0008            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:45:35    16-Jul-86            Page 35
CLIPL3SGR.FTN;6          /F77/WR

0001            SUBROUTINE TEST2(KODE,ID)
        C
        C TEST RETURN CODE FROM ADQUAD/ADQUA2/ADQUA3
        C
0002            IF(KODE.EQ.0) RETURN
0003            WRITE(5,1) KODE, ID
0004        1   FORMAT(' ADAPTIVE INTEGRATOR CODE = ',I2,' FROM CALL NUMBER ',I5)
0005            STOP 'FATAL ERROR'
0006            END
```



```
PDP-11 FORTRAN-77 V4.0-1        10:45:36    16-Jul-86            Page 36
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION PZ1(Y)
        C
        C       SIGNAL CHANNEL P.D.F. WITH CHANGE OF VARIABLE Y=AX
        C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            DIMENSION WORK1(30), STACK1(30), HEAP1(30)
0004            LOGICAL*1 REG1, REG2, REG3
0005            EXTERNAL DGX, F30A1, F30A2, F30B1,
          $             F30B2, F31A, F31B1, F31B2, F31B3, F31C2,
          $             F32A, F32B1, F32B2, F32B3, F32C2,
          $             F33A1, F33A2, F33B1, F33B2
0006            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0007            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0008            COMMON /PARDEN/ RHON, RHOT
0009            COMMON /QUES/ QO, Q1
0010            COMMON /XCOM/ XCON
0011            COMMON /OUTER/ XXX, XXXK
0012            XXX=Y
0013            IF(LJAM.GE.1) THEN
0014              XXXK=Y/BIGK
0015              YK=XXXK
0016              YTK=(Y-TAU)/BIGK
0017              YTK2=(Y-TAU2)/BIGK
0018            END IF
0019            REG1=Y.GE.0.DO .AND. Y.LT.TAU
0020            REG2=Y.GE.TAU .AND. Y.LT.TAU2
0021            REG3=Y.GE.TAU2 .AND. Y.LT.TAU3
        C
        C       THREE HOPS PER SYMBOL
        C
0022     3000   GOTO (3100, 3200, 3300, 3400), LJAM+1
        C
        C               NO HOPS JAMMED
        C
0023     3100   IF(REG1) THEN
0024              BARG1=DSQRT(12.DO*RHON*Y)
0025              IO1=2
0026              CALL DXBT(3100)
0027              PZ1=(Y/(3.DO*RHON))*DEXP(BARG1-Y-3.DO*RHON)*B1
0028            ELSE IF(REG2) THEN
0029              BARG1=DSQRT(8.DO*RHON*(Y-TAU))
0030              IO1=1
0031              CALL DXBT(3101)
0032              PART=1.5DO*QO*DSQRT(2.DO*(Y-TAU)/RHON)
          $                *DEXP(BARG1-Y+TAU-2.DO*RHON)*B1
0033              XCON=Y+3.DO*RHON
0034              CALL ADQUA2(Y-TAU,TAU,ANSWER,DGX,F30A1,1.D-9,WORK1,STACK1,
          $                   HEAP1,30,KOD)
0035              CALL TEST2(KOD,3100)
0036              PART=PART+ANSWER/2.DO
```

```
            CALL ADQUA2(TAU,Y,ANSWER,DGX,F30A2,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3101)
            PZ1=PART+ANSWER
          ELSE IF(REG3) THEN
            BARG1=DSQRT(4.DO*RHON*(Y-TAU2))
            IO1=0
            CALL DXBT(3103)
            PART=3.DO*QO*QO*DEXP(BARG1-Y+TAU2-RHON)*B1
            XCON=Y-TAU+2.DO*RHON
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F30B1,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3103)
            PART=PART+3.DO*QO*ANSWER
            XCON=Y+3.DO*RHON
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F30B2,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3104)
            PZ1=PART+ANSWER
          ELSE
            PZ1=0.DO
          END IF
          GOTO 9000
C
C                ONE HOP JAMMED
C
 3200     IF(REG1) THEN
            XCON=YK+2.DO*RHON+RHOT
            CALL ADQUA2(0.DO,Y,ANSWER,DGX,F31A,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3200)
            PZ1=ANSWER*0.5DO/BIGK
          ELSE IF(REG2) THEN
            BARG1=DSQRT(8.DO*RHON*(Y-TAU))
            IO1=1
            CALL DXBT(3200)
            PART=0.5DO*Q1*DSQRT(2.DO*(Y-TAU)/RHON)
     $                 *DEXP(BARG1-Y+TAU-2.DO*RHON)*B1
            XCON=YK+2.DO*RHON+RHOT
            CALL ADQUA2(Y-TAU,TAU,ANSWER,DGX,F31B1,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3201)
            PART=PART+0.5DO*ANSWER/BIGK
            XCON=YTK+RHON+RHOT
            CALL ADQUA2(0.DO,Y-TAU,ANSWER,DGX,F31B2,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3202)
            PART=PART+2.DO*QO*ANSWER/BIGK
            XCON=YK+2.DO*RHON+RHOT
            CALL ADQUA2(TAU,Y,ANSWER,DGX,F31B3,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3203)
            PZ1=PART+ANSWER/BIGK
```



```
          ELSE IF(REG3) THEN
            BARG1=DSQRT(4.DO*RHOT*(YK-TAU2))
            IO1=0
            CALL DXBT(3204)
            PART=QO*QO*DEXP(BARG1-YK+TAUK2-RHOT)*B1/BIGK
            BARG1=DSQRT(4.DO*RHON*(Y-TAU2))
            IO1=0
            CALL DXBT(3205)
            PART=PART+1.DO*QO*Q1*DEXP(BARG1-Y+TAU2-RHON)*B1
            XCON=Y-TAU+2.DO*RHON
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F30B1,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3204)
            PART=PART+Q1*ANSWER
            XCON=YK-TAU+RHON+RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F31C2,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3205)
            PART=PART+2.DO*QO*ANSWER/BIGK
            XCON=YK+2.DO*RHON+RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F31B3,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3206)
            PZ1=PART+ANSWER/BIGK
          ELSE
            PZ1=0.DO
          END IF
          GOTO 9000
C
C                TWO HOPS JAMMED
C
 3300     IF(REG1) THEN
            XCON=Y+2.DO*RHOT+RHON
            CALL ADQUA2(0.DO,Y,ANSWER,DGX,F32A,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3301)
            PZ1=0.5DO*ANSWER/BIGK
          ELSE IF(REG2) THEN
            BARG1=DSQRT(8.DO*RHOT*YTK)
            IO1=1
            CALL DXBT(3302)
            PART=0.5DO*QO*DSQRT(2.DO*(Y-TAU)/RHON)
     $                 *DEXP(BARG1-YK+TAUK-2.DO*RHOT)*B1/BIGK
            XCON=Y+2.DO*RHOT+RHON
            CALL ADQUA2(Y-TAU,TAU,ANSWER,DGX,F32B1,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3302)
            PART=PART+ANSWER*0.5DO/BIGK
            XCON=Y-TAU+RHON+RHOT
            CALL ADQUA2(0.DO,Y-TAU,ANSWER,DGX,F32B2,1.D-9,WORK1,STACK1,
     $                 HEAP1,30,KOD)
            CALL TEST2(KOD,3303)
```

```
            PART=PART+2.D0*Q1*ANSWER/BIGK
            XCON=Y+2.D0*RHOT+RHON
            CALL ADQUA2(TAU,Y,ANSWER,DGX,F32B3,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3304)
            PZ1=PART+ANSWER/(BIGK*BIGK)
          ELSE IF(REG3) THEN
            BARG1=DSQRT(4.D0*RHON*(Y-TAU2))
            I01=0
            CALL DXBT(3305)
            PART=Q1*Q1*DEXP(BARG1-Y+TAU2-RHON)*B1
            BARG1=DSQRT(4.D0*RHOT*YTK2)
            I01=1
            CALL DXBT(3306)
            PART=PART
     $          +2.D0*Q1*Q0*DEXP(BARG1-YTK2-RHOT)*B1/BIGK
            XCON=YK+TAUK-2.D0*RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F33B1,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3305)
            PART=PART+Q0*ANSWER/(BIGK*BIGK)
            XCON=YTK+RHON+RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F32C2,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3306)
            PART=PART+2.D0*Q1*ANSWER/BIGK
            XCON=Y+2.D0*RHOT+RHON
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F32B3,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3307)
            PZ1=PART+ANSWER/(BIGK*BIGK)
          ELSE
            PZ1=0.D0
          END IF
          GOTO 9000
C
C                THREE HOPS JAMMED
C
 3400     IF(REG1) THEN
            BARG1=DSQRT(12.D0*RHOT*YK)
            I01=2
            CALL DXBT(3400)
            PZ1=(Y/(3.D0*RHON))*DEXP(BARG1-YK-3.D0*RHOT)*B1/BIGK
          ELSE IF(REG2) THEN
            BARG1=DSQRT(8.D0*RHOT*YTK)
            I01=1
            CALL DXBT(3401)
            PART=1.5D0*Q1*DSQRT(2.D0*(Y-TAU)/RHON)
     $          *DEXP(BARG1-YTK-2.D0*RHOT)*B1/BIGK
            XCON=YK+3.D0*RHOT
            CALL ADQUA2(Y-TAU,TAU,ANSWER,DGX,F33A1,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
```



```
            CALL TEST2(KOD,3400)
            PART=PART+0.5D0*ANSWER/(BIGK*BIGK)
            CALL ADQUA2(TAU,Y,ANSWER,DGX,F33A2,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3401)
            PZ1=PART+ANSWER/DXI(BIGK,3)
          ELSE IF(REG3) THEN
            BARG1=DSQRT(4.D0*RHOT*YTK2)
            I01=0
            CALL DXBT(3402)
            PART=3.D0*Q1*Q1*DEXP(BARG1-YTK2-RHOT)*B1/BIGK
            XCON=YTK+2.D0*RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F33B1,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3402)
            PART=PART+3.D0*Q1*ANSWER/(BIGK*BIGK)
            XCON=YK+3.D0*RHOT
            CALL ADQUA2(Y-TAU,TAU2,ANSWER,DGX,F33B2,1.D-9,WORK1,STACK1,
     $                HEAP1,30,KOD)
            CALL TEST2(KOD,3403)
            PZ1=PART+ANSWER/DXI(BIGK,3)
          ELSE
            PZ1=0.D0
          END IF
 9000     CONTINUE
          RETURN
          END
```

```
      DOUBLE PRECISION FUNCTION GL(Y)
C
C NON-SIGNAL CHANNEL CUMULATIVE DISTRIBUTION FUNCTION
C     WITH CHANGE OF VARIABLE Y=AX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL*1 REG1, REG2, REG3
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
     $                TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
      REG1=Y.GE.0.D0 .AND. Y.LT.TAU
      REG2=Y.GE.TAU  .AND. Y.LT.TAU2
      REG3=Y.GE.TAU2 .AND. Y.LT.TAU3
      IF(LJAM.GT.0) THEN
        YK=Y/BIGK
        YTK=(Y-TAU)/BIGK
        YTK2=(Y-TAU2)/BIGK
      END IF
C
C     THREE HOPS PER SYMBOL
C
 3000 GOTO (3100, 3200, 3300, 3400), LJAM+1
C
C             NO HOPS JAMMED
C
 3100 IF(REG1) THEN
        GL=1.D0-(1.D0+Y+0.5D0*Y*Y)*DEXP(-Y)
      ELSE IF(REG2) THEN
        YMT=Y-TAU
        GL=1.D0-(1.D0+TAU+0.5D0*TAU*TAU
     $            +(1.D0+TAU)*YMT-YMT*YMT)*DEXP(-Y)
      ELSE IF(REG3) THEN
        YM2T=Y-TAU2
        GL=1.D0-(1.D0+TAU+TAU+0.5D0*TAU*TAU
     $            -(2.D0+TAU)*YM2T+0.5D0*YM2T*YM2T)*DEXP(-Y)
      ELSE IF(Y.GE.TAU3) THEN
        GL=1.D0
      ELSE
        GL=0.D0
      END IF
      GOTO 9000
C
C             ONE HOP JAMMED
C
 3200 IF(REG1) THEN
        GL=1.D0-AAB2*DEXP(-YK)+BAB*(1.D0+AAB+Y)*DEXP(-Y)
      ELSE IF(REG2) THEN
        YMT=Y-TAU
        GL=1.D0-DEXP(-YMT)*(DEXP(-YK)*(AAB*(AAB+YMT))+
     $                      DEXP(-Y)*(BAB*(BAB+Y-TAU2)))
     $           + 2.D0*AABBAB*DEXP(TAUK-TAU-YK)
      ELSE IF(REG3) THEN
        YM2T=Y-TAU2
        GL=1.D0-DEXP(TAU-TAUK-Y)*(AAB*(1.D0+Y)-AABBAB-AAB*YM2T)
     $           -BAB2*DEXP(TAUK2-TAU2-YK)
      ELSE IF(Y.GE.TAU3) THEN
        GL=1.D0
      ELSE
        GL=0.D0
      END IF
      GOTO 9000
C
C             TWO HOPS JAMMED
C
 3300 IF(REG1) THEN
        GL=1.D0-BAB2*DEXP(-Y)-AAB*DEXP(-YK)
     $         *(1.D0-BAB+YK)
      ELSE IF(REG2) THEN
        GL=1.D0-DEXP(TAUK-YK)
     $         *(DEXP(-Y)*(BAB2-BAB*YTK)
     $          +DEXP(-YK)*(AAB2-BAB*(Y-TAU2)))
     $          +2.D0+AABBAB*DEXP(TAU-TAUK-Y)
      ELSE IF(REG3) THEN
        GL=1.D0-DEXP(TAUK-TAU-YK)*(BAB*YTK2
     $          -BAB*(1.D0+YK)-AABBAB)
     $          -AAB2*DEXP(TAU2-TAUK2-Y)
      ELSE IF(Y.GE.TAU3) THEN
        GL=1.D0
      ELSE
        GL=0.D0
      END IF
      GOTO 9000
C
C             THREE HOPS JAMMED
C
 3400 IF(Y.GE.0.D0 .AND. Y.LT.TAU) THEN
        GL=1.D0-DEXP(-YK)*(1.D0+YK+0.5D0*YK*YK)
      ELSE IF(Y.GE.TAU .AND. Y.LT.TAU2) THEN
        GL=1.D0-DEXP(-YK)*(1.D0+TAUK+0.5D0*TAUK*TAUK
     $                    +(1.D0+TAUK)*YTK-YTK*YTK)
      ELSE IF(Y.GE.TAU2 .AND. Y.LT.TAU3) THEN
        GL=1.D0-DEXP(-YK)*(1.D0+2.D0*TAUK
     $     +0.5D0*TAUK*TAUK-(2.D0+TAUK)*YTK2+0.5D0*YTK2*YTK2)
      ELSE IF(Y.GE.TAU3) THEN
        GL=1.D0
      ELSE
        GL=0.D0
      END IF
 9000 CONTINUE
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:11    16-Jul-86           Page 43
CLIPL3SGR.FTN;6         /F77/WR

0001            SUBROUTINE BPROD(IDENT)
        C
        C COMPUTE TWO BESSEL FUNCTIONS, ARGUMENTS AND RESULTS IN COMMON
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            CALL DXBT(IDENT)
0005            CALL DXBESI(BARG2,IO2,B2,KODE)
0006            CALL TEST(IDENT+1)
0007            RETURN
0008            END

PDP-11 FORTRAN-77 V4.0-1        10:46:12    16-Jul-86           Page 44
CLIPL3SGR.FTN;6         /F77/WR

0001            DOUBLE PRECISION FUNCTION F30A1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=0,
        C FIRST INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            BARG1=DSQRT(8.D0*RHON*U)
0010            BARG2=DSQRT(4.D0*RHON*(XXX-U))
0011            IO1=1
0012            IO2=0
0013            CALL BPROD(3110)
0014            S=DSQRT(2.D0*U/RHON)
0015            F30A1=S*((DEXP(BARG1+BARG2-XCON)*B1)*B2)
0016            RETURN
0017            END

PDP-11 FORTRAN-77 V4.0-1        10:46:14    16-Jul-86           Page 45
CLIPL3SGR.FTN;6         /F77/WR

0001            SUBROUTINE DXBT(ID)
        C
        C CALL DXBESI AND TEST RETURN CODE
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            CALL DXBESI(BARG1,IO1,B1,KODE)
0005            CALL TEST(ID)
0006            RETURN
0007            END

PDP-11 FORTRAN-77 V4.0-1        10:46:15    16-Jul-86           Page 46
CLIPL3SGR.FTN;6         /F77/WR

0001            DOUBLE PRECISION FUNCTION F30A2(U)
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=0,
        C SECOND INTEGRAL
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL DGTEN, F30ABA
0004            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0005            COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ Q0, Q1
0009            COMMON /XCOM/ XCON
0010            COMMON /OUTER/ XXX, XXXK
0011            COMMON /INNER/ WWW, WWWK
0012            WWW=U
0013            CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F30ABA,1.D-10,
          $                 WORK,STACK,HEAP,30,KOD)
0014            CALL TEST2(KOD,3112)
0015            BARG1=DSQRT(4.D0*RHON*(XXX-U))
0016            IO1=0
0017            CALL DXBT(3112)
0018            F30A2=DEXP(BARG1-XCON)*B1*ANSWER
0019            RETURN
0020            END

PDP-11 FORTRAN-77 V4.0-1        10:46:17    16-Jul-86           Page 47
CLIPL3SGR.FTN;6         /F77/WR

0001            DOUBLE PRECISION FUNCTION F30ABA(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=0,
        C SECOND INTEGRAL'S INNER INTEGRAL; ALSO L=3, LJAM=1, SECOND
        C REGION, THIRD INTEGRAL, INNER INTEGRAL.
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            COMMON /INNER/ WWW, WWWK
0010            BARG1=DSQRT(4.D0*RHON*U)
0011            CALL DBESI(BARG1,0,B1,KODE)
0012            CALL TEST(13110)
0013            BARG2=DSQRT(4.D0*RHON*(WWW-U))
0014            CALL DBESI(BARG2,0,B2,KODE)
0015            CALL TEST(13111)
0016            F30ABA=B1*B2
0017            RETURN
0018            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:20    16-Jul-86            Page 48
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F30B1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=0,
        C THIRD REGION, FIRST INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0  Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            BARG1=DSQRT(4.D0*RHON*(XXX-U))
0010            BARG2=DSQRT(4.D0*RHON*(U-TAU))
0011            I01=0
0012            I02=0
0013            CALL BPROD(3120)
0014            F30B1=DEXP(BARG1+BARG2-XCON)*B1*B2
0015            RETURN
0016            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:21    16-Jul-86            Page 49
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F30B2(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=0,
        C SECOND REGION, SECOND INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL DGTEN, F30ABA
0004            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0005            COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ Q0, Q1
0009            COMMON /XCOM/ XCON
0010            COMMON /OUTER/ XXX, XXXK
0011            COMMON /INNER/ WWW, WWWK
0012            WWW=U
0013            CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F30ABA,1.D-10,
          $                 WORK,STACK,HEAP,30,KOD)
0014            CALL TEST2(KOD,3120)
0015            BARG1=DSQRT(4.D0*RHON*(XXX-U))
0016            I01=0
0017            CALL OXBT(3121)
0018            F30B2=DEXP(BARG1-XCON)*B1*ANSWER
0019            RETURN
0020            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:23    16-Jul-86            Page 50
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F31A(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=1,
        C FIRST REGION
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0004            COMMON /PARDEN/ RHON, RHOT
0005            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            BARG1=DSQRT(4.D0*RHOT*(XXXK-U/BIGK))
0010            BARG2=DSQRT(2.D0*RHON*U)
0011            I01=0
0012            I02=1
0013            CALL BPROD(3210)
0014            F31A=DEXP(BARG1+BARG2-XCON)*B1*B2*DSQRT(2.D0*U/RHON)
0015            RETURN
0016            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:25    16-Jul-86            Page 51
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F31B1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=1,
        C SECOND REGION, FIRST INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
          $                     TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0004            COMMON /PARDEN/ RHON, RHOT
0005            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            BARG1=DSQRT(8.D0*RHON*U)
0010            UK=U/BIGK
0011            BARG2=DSQRT(4.D0*RHOT*(XXXK-UK))
0012            I01=1
0013            I02=0
0014            CALL BPROD(3220)
0015            S=DSQRT(2.D0*U/RHON)
0016            F31B1=DEXP(BARG1+BARG2-XCON-U+UK)*B1*B2*S
0017            RETURN
0018            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:27    16-Jul-86           Page 52
CLIPL3SGR.FTN;6          /F77/WR

              DOUBLE PRECISION FUNCTION F31B2(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=1,
        C SECOND REGION, SECOND INTEGRAL
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
              COMMON /PARDEN/ RHON, RHOT
              COMMON /QUES/ Q0, Q1
              COMMON /XCOM/ XCON
              COMMON /OUTER/ XXX, XXXK
              BARG1=DSQRT(4.D0*RHON*U)
              UK=U/BIGK
              BARG2=DSQRT(4.D0*RHOT*(XXXK-TAUK-UK))
              IO1=0
              IO2=0
              CALL BPROD(3222)
              F31B2=DEXP(BARG1+BARG2-XCON-U+UK)*B1*B2
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:29    16-Jul-86           Page 53
CLIPL3SGR.FTN;6          /F77/WR

              DOUBLE PRECISION FUNCTION F31B3(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=1,
        C SECOND REGION, THIRD INTEGRAL
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              EXTERNAL DGTEN, F30ABA
              COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
              COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
              COMMON /PARDEN/ RHON, RHOT
              COMMON /QUES/ Q0, Q1
              COMMON /XCOM/ XCON
              COMMON /OUTER/ XXX, XXXK
              COMMON /INNER/ WWW, WWWK
              WWW=U
              CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F30ABA,1.D-10,
           $              WORK,STACK,HEAP,30,KOD)
              CALL TEST2(KOD,3222)
              UK=U/BIGK
              BARG1=DSQRT(4.D0*RHOT*(XXXK-UK))
              IO1=0
              CALL DXBT(3222)
              F31B3=DEXP(BARG1-XCON-U+UK)*B1*ANSWER
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:31    16-Jul-86           Page 54
CLIPL3SGR.FTN;6          /F77/WR

              DOUBLE PRECISION FUNCTION F31C2(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=1,
        C THIRD REGION, SECOND INTEGRAL
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
              COMMON /PARDEN/ RHON, RHOT
              COMMON /QUES/ Q0, Q1
              COMMON /XCOM/ XCON
              COMMON /OUTER/ XXX, XXXK
              UK=U/BIGK
              BARG1=DSQRT(4.D0*RHOT*(XXXK-UK))
              BARG2=DSQRT(4.D0*RHON*(U-TAU))
              IO1=0
              IO2=0
              CALL BPROD(3130)
              F31C2=DEXP(BARG1+BARG2-XCON-U+UK)*B1*B2
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:33    16-Jul-86           Page 55
CLIPL3SGR.FTN;6          /F77/WR

              DOUBLE PRECISION FUNCTION F33A1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=3,
        C FIRST INTEGRAL
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
              COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
              COMMON /PARDEN/ RHON, RHOT
              COMMON /QUES/ Q0, QI
              COMMON /XCOM/ XCON
              COMMON /OUTER/ XXX, XXXK
              UK=U/BIGK
              BARG1=DSQRT(8.D0*RHOT*UK)
              BARG2=DSQRT(4.D0*RHOT*(XXXK-UK))
              IO1=1
              IO2=0
              CALL BPROD(3410)
              S=DSQRT(2.D0*U/RHON)
              F33A1=S*((DEXP(BARG1+BARG2-XCON)*B1)*B2)
              RETUPN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:35    16-Jul-86           Page 56
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F33A2(U)
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=3,
        C SECOND INTEGRAL
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL DGTEN, F33ABA
0004            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0005            COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
            $                   TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ Q0, Q1
0009            COMMON /XCOM/ XCON
0010            COMMON /OUTER/ XXX, XXXK
0011            COMMON /INNER/ WWW, WWWK
0012            UK=U/BIGK
0013            WWWK=UK
0014            CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F33ABA,1.D-10,
            $                WORK,STACK,HEAP,30,KOD)
0015            CALL TEST2(KOD,3412)
0016            BARG1=DSQRT(4.D0*RHON*(XXXK-UK))
0017            IO1=0
0018            CALL DXBT(3112)
0019            F33A2=DEXP(BARG1-XCON)*B1*ANSWER
0020            RETURN
0021            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:38    16-Jul-86           Page 57
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F33ABA(U)
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=3,
        C SECOND INTEGRAL'S INNER INTEGRAL; ALSO L=3, LJAM=2, SECOND
        C REGION, THIRD INTEGRAL, INNER INTEGRAL.
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
            $                   TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            COMMON /INNER/ WWW, WWWK
0010            UK=U/BIGK
0011            BARG1=DSQRT(4.D0*RHOT*UK)
0012            CALL DBESI(BARG1,0,B1,KODE)
0013            CALL TEST(13410)
0014            BARG2=DSQRT(4.D0*RHOT*(WWWK-UK))
0015            CALL DBESI(BARG2,0,B2,KODE)
0016            CALL TEST(13411)
0017            F33ABA=B1*B2
0018            RETURN
0019            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:39    16-Jul-86           Page 58
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F33B1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=3,
        C THIRD REGION, FIRST INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
            $                   TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            UK=U/BIGK
0010            BARG1=DSQRT(4.D0*RHOT*(XXXK-UK))
0011            BARG2=DSQRT(4.D0*RHOT*(UK-TAUK))
0012            IO1=0
0013            IO2=0
0014            CALL BPROD(3420)
0015            F33B1=DEXP(BARG1+BARG2-XCON)*B1*B2
0016            RETURN
0017            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:41    16-Jul-36           Page 59
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F33B2(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=3,
        C SECOND REGION, SECOND INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL DGTEN, F33ABA
0004            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0005            COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
            $                   TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ Q0, Q1
0009            COMMON /XCOM/ XCON
0010            COMMON /OUTER/ XXX, XXXK
0011            COMMON /INNER/ WWW, WWWK
0012            UK=U/BIGK
0013            WWWK=UK
0014            CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F33ABA,1.D-10,
            $                WORK,STACK,HEAP,30,KOD)
0015            CALL TEST2(KOD,3420)
0016            BARG1=DSQRT(4.D0*RHOT*(XXXK-UK))
0017            IO1=0
0018            CALL DXBT(3421)
0019            F33B2=DEXP(BARG1-XCON)*B1*ANSWER
0020            RETURN
0021            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:44    16-Jul-86           Page 60
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F32A(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=2,
        C FIRST REGION
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                    TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            UK=U/BIGK
0010            BARG1=DSQRT(4.D0*RHON*(XXX-U))
0011            BARG2=DSQRT(2.D0*RHOT*UK)
0012            IO1=0
0013            IO2=1
0014            CALL BPROD(3310)
0015            F32A=DEXP(BARG1+BARG2-XCON)*B1*B2*DSQRT(2.D0*U/RHON)
0016            RETURN
0017            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:46    16-Jul-86           Page 61
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F32B1(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=2,
        C SECOND REGION, FIRST INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                    TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            UK=U/BIGK
0010            BARG1=DSQRT(8.D0*RHOT*UK)
0011            BARG2=DSQRT(4.D0*RHON*(XXX-U))
0012            IO1=1
0013            IO2=0
0014            CALL BPROD(3320)
0015            S=DSQRT(2.D0*U/RHON)
0016            F32B1=DEXP(BARG1+BARG2-XCON-UK+U)*B1*B2*S
0017            RETURN
0018            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:48    16-Jul-86           Page 62
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F32B2(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=2,
        C SECOND REGION, SECOND INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0004            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                    TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0005            COMMON /PARDEN/ RHON, RHOT
0006            COMMON /QUES/ Q0, Q1
0007            COMMON /XCOM/ XCON
0008            COMMON /OUTER/ XXX, XXXK
0009            UK=U/BIGK
0010            BARG1=DSQRT(4.D0*RHOT*UK)
0011            IO1=0
0012            BARG2=DSQRT(4.D0*RHON*(XXX-TAU-U))
0013            IO2=0
0014            CALL BPROD(3322)
0015            F32B2=DEXP(BARG1+BARG2-XCON-UK+U)*B1*B2
0016            RETURN
0017            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:46:50    16-Jul-86           Page 63
CLIPL3SGR.FTN;6          /F77/WR

0001            DOUBLE PRECISION FUNCTION F32B3(U)
        C
        C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=2,
        C SECOND REGION, THIRD INTEGRAL
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL DGTEN, F33ABA
0004            COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, IO1, IO2
0005            COMMON /INWORK/ WORK(30), STACK(30), HEAP(30)
0006            COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
           $                    TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
0007            COMMON /PARDEN/ RHON, RHOT
0008            COMMON /QUES/ Q0, Q1
0009            COMMON /XCOM/ XCON
0010            COMMON /OUTER/ XXX, XXXK
0011            COMMON /INNER/ WWW, WWWK
0012            UK=U/BIGK
0013            WWWK=UK
0014            CALL ADQUA3(U-TAU,TAU,ANSWER,DGTEN,F33ABA,1.D-10,
           $                WORK,STACK,HEAP,30,KOD)
0015            CALL TEST2(KOD,3322)
0016            BARG1=DSQRT(4.D0*RHON*(XXX-U))
0017            IO1=0
0018            CALL DXBT(3324)
0019            F32B3=DEXP(BARG1-XCON-UK+U)*B1*ANSWER
0020            RETURN
0021            END
```

```
      DOUBLE PRECISION FUNCTION F32C2(U)
C
C INTEGRAND FUNCTION FOR SIGNAL CHANNEL DENSITY, L=3, LJAM=2,
C THIRD REGION, SECOND INTEGRAL
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /LOCALS/ BARG1, BARG2, B1, B2, KODE, I01, I02
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
     $                TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
      COMMON /XCOM/ XCON
      COMMON /OUTER/ XXX, XXXK
      UK=U/BIGK
      BARG1=DSQRT(4.D0*RHON*(XXX-U))
      I01=0
      BARG2=DSQRT(4.D0*RHOT*(UK-TAUK))
      I02=0
      CALL BPROD(3330)
      F32C2=DEXP(BARG1+BARG2-XCON-UK+U)*B1*B2
      RETURN
      END
```



```
      SUBROUTINE TIES(JSUB,MM,PTIE)
C  COMPUTE PROBABILITY OF CORRECT DECISION GIVEN THAT
C  SEVERAL SATURATED CHANNELS ARE TIED
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION JSUB(MM), LLOW(7), LINC(7), LUP(7), NU(7),
     $        P2LM(2:8)
      LOGICAL*1 GO
      COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
     $                TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
      COMMON /PARDEN/ RHON, RHOT
      COMMON /QUES/ Q0, Q1
C NUMBER OF NON-SIGNAL CHANNELS
      MNU=MM-1
      PTIE=0.D0
      CUE0=DEXP(-TAU)
      CUE1=DEXP(-TAUK)
      P1L=DXI(Q0,3-JSUB(1))*DXI(Q1,JSUB(1))
      DO 10 I=2,MM
      P2LM(I)=DXI(CUE0,3-JSUB(I))*DXI(CUE1,JSUB(I))
   10 CONTINUE
C
C SET UP VECTOR LOOP PARAMETERS
C
      DO 20 I=1,MM-1
      LLOW(I)=0
      LINC(I)=1
      LUP(I)=1
   20 CONTINUE
      PTIE=0.D0
C
C START LOOP ON THE TIE EVENTS
C
      CALL VLINIT(NU,LLOW,MM-1)
   30 NUSUM=0
      DO 40 I=1,MM-1
      NUSUM=NUSUM+NU(I)
   40 CONTINUE
      FRAC=1.D0/(1.D0+NUSUM)
      PROD=1.D0
      DO 50 M=2,MM
      IF(NU(M-1).EQ.1) THEN
        PROD=PROD*P2LM(M)
      ELSE
        PROD=PROD*(1.D0-P2LM(M))
      END IF
   50 CONTINUE
      PTIE=FRAC*PROD+PTIE
      CALL VLITER(NU,LLOW,LUP,LINC,MM-1,GO)
      IF(GO) GOTO 30
      PTIE=PTIE*P1L
      RETURN
      END
```

```
        DOUBLE PRECISION FUNCTION PUNJAM(ETA)
C
C FUNCTION FOR UNJAMMED P(E) FOR OPT. THRESHOLD SEARCH
C
C       NOTE: WHEN JAMMING EVENT IS (0,0,...,0), THE VARIABLES
C             BIGK, AAB, BAB, TAUK, TAUK2, AND TAUK3 ARE NOT
C             USED IN THE COMPUTATIONS, AND HENCE DO NOT NEED
C             TO BE SET UP BEFORE CALLING PSEL FROM THIS FUNCTION
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION NOJAM(4)
        COMMON /INPUTS/ DEBNOL(3),NSLOTS,K,MM
        COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
     $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
        DATA NOJAM/0,0,0,0/
        LJAM=0
        TAU=ETA
        TAU2=TAU+ETA
        TAU3=TAU2+ETA
        CALL PSEL(NOJAM,MM,P)
        PUNJAM=P
        RETURN
        END
```



```
        SUBROUTINE SETTAU(MM,PEOO)
C
C  SEARCH FOR OPTIMUM THRESHOLD IN ABSENCE OF JAMMING
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        EXTERNAL PUNJAM
        COMMON /DENPAR/ BIGK, AAB, BAB, LJAM, AAB2, BAB2, AABBAB,
     $                  TAU, TAU2, TAU3, TAUK, TAUK2, TAUK3
        COMMON /PARDEN/ RHON, RHOT
        COMMON /QUES/ Q0, Q1
        LJAM=0
C
C GUESS BASED ON PREVIOUS RESULTS
C
        GUESS=8.D0
        STEP=1.D0
C FOR M=4, L=3 THE OPTIMUM THRESHOLD IS 8.15
        CALL MINSER(PUNJAM,PEMIN,TAUOPT,STEP,GUESS,0.D0,
     $              50.D0,0.01D0)
        TAU=TAUOPT
        TAU2=TAU+TAU
        TAU3=TAU2+TAU
        PEOO=PEMIN
        RETURN
        END
```

APPENDIX K

COMPUTER PROGRAM FOR

SELF-NORMALIZING RECEIVER WITH M=2, L=2

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the self-normalizing receiver when M=2 and L=2, with jamming fraction γ as a parameter.

```
0001          PROGRAM SELF22
      C
      C SELF NORMALIZING RECEIVER, M=2, L=2
      C
      C ANALYSIS: L.E. MILLER
      C PROGRAM:  R.H. FRENCH
      C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          PARAMETER(M=2, L=2, NSLOTS=2400)
0004          CHARACTER*13 FNAME
0005          REAL DEBNJ(126), PELOG(126)
0006          COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
0007          COMMON /PARMS/ NO, NJ, NG, DEBNOL(5), GAMLST(10),
            $              DEBNJL(126), DJ, DBJO
0008          CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0009          CALL GET
0010          DO 900 IO=1,NO
0011          EBNO=10.DO**(DEBNOL(IO)/10.DO)
0012          RHON=EBNO/2.DO
0013          IOOUT=DEBNOL(IO)
0014          DO 800 IG=1,NG
0015          GAMMA=GAMLST(IG)
0016          IQ=GAMMA*NSLOTS+0.5DO
0017          IGOUT=1000.DO*GAMMA+0.5DO
      C
      C PROGRESS FILE
      C
0018          WRITE(FNAME,1) IOOUT,IGOUT
0019        1 FORMAT('S22',I2.2,I4.4,'.DAT')
0020          WRITE(5,2) FNAME
0021        2 FORMAT(' WORKING ON FILE ',A13)
0022          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ERR=810,
            $     FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
      C
      C HAVE A PROGRESS FILE, READ IT
      C
0023          READ(4) EBNOIN, GAMMIN, DBJOIN, DJIN
0024          IF(EBNOIN.NE.DEBNOL(IO) .OR. GAMMIN.NE.GAMMA .OR.
            $   DBJOIN.NE.DBJO .OR. DJIN.NE.DJ) STOP 'FILE CORRUPT'
0025          JJ=0
0026      801 JJ=JJ+1
0027          READ(4,END=802) DEBNJ(JJ), PELOG(JJ)
0028          GOTO 801
0029      802 CLOSE(UNIT=4)
0030          GOTO 820
      C
      C NO FILE, MUST CREATE IT
      C
0031      810 OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED',
            $     ACCESS='SEQUENTIAL')
0032          WRITE(4) DEBNOL(IO), GAMMA, DBJO, DJ
0033          CLOSE(UNIT=4)
```



```
0034          JJ=1
      C
      C KEEP ON GOING
      C
0035      820 DO 700 IJ=JJ,NJ
0036          WRITE(5,821) IJ
0037      821 FORMAT(' IJ = ',I3)
0038          DBEBNJ=DBJO+(IJ-1)*DJ
0039          DEBNJ(IJ)=DBEBNJ
0040          EBNJ=10.DO**(DBEBNJ/10.DO)
0041          RHOJ=EBNJ/2.DO
0042          RHOJ1=GAMMA*RHOJ
0043          RHOT=RHOJ1*RHON/(RHOJ1+RHON)
0044          BIGK=RHON/RHOT
0045          CALL PSUBE(IQ,PE)
0046          PELOG(IJ)=DLOG10(PE)
0047          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
            $     FORM='UNFORMATTED')
0048          WRITE(4) DEBNJ(IJ), PELOG(IJ)
0049          CLOSE(UNIT=4)
0050      700 CONTINUE
0051          OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',ACCESS='SEQUENTIAL',
            $     FORM='UNFORMATTED')
0052          WRITE(4) M, L, DEBNOL(IO), GAMMA, NSLOTS, NJ, DEBNJ, PELOG
0053          CLOSE(UNIT=4)
0054          WRITE(6,710) M, L, DEBNOL(IO), GAMMA, NSLOTS,
            $              (DEBNJ(IJ),10.DO**PELOG(IJ),IJ=1,NJ)
0055      710 FORMAT('1SELF-NORMALIZING RECEIVER FOR M=',I1,' AND L=',I1/
            $ 5X,'EB/NO = ',F8.5,' dB',5X,'GAMMA=',1PD10.3,5X,'NSLOTS=',I4/
            $ ' EB/NJ (dB)',8X,'P(E)'/<NJ>(4X,0PF4.1,8X,1PD10.3/))
0056      800 CONTINUE
0057      900 CONTINUE
0058          STOP 'PLEASE PURGE S22*.DAT'
0059          END
```

```
      SUBROUTINE GET
C
C INTERACTIVE RUN PARAMETER INPUTS
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION DGAM(10), DOLST(5)
      CHARACTER*8 REPLY, BLANKS
      COMMON /PARMS/ NO, NJ, NG, DEBNOL(5), GAMLST(10),
     $               DEBNJL(126), DJ, DBJO
      DATA DGAM/1.D-3, 2.D-3, 5.D-3,
     $          1.D-2, 2.D-2, 5.D-2,
     $          1.D-1, 2.D-1, 5.D-1, 1.D0/
      DATA DOLST/ 13.35247D0, 12.3133D0, 10.94443D0, 14.89253D0,
     $           16.027135D0/
      DATA BLANKS/'        '/
    1 WRITE(5,2)
    2 FORMAT(' HOW MANY EB/NO? [1] ',$)
      READ(5,3,ERR=1) NO
    3 FORMAT(I1)
      IF(NO.EQ.0) NO=1
      IF(NO.LT.0 .OR. NO.GT.5) GOTO 1
      DO 8 IN=1,NO
    4 WRITE(5,5) IN, DOLST(IN)
    5 FORMAT(3X,'EB/NO(',I1,') [',F8.5,' dB]: ',$)
      READ(5,6,ERR=4) REPLY
    6 FORMAT(A8)
      IF(REPLY.EQ.BLANKS) THEN
        DEBNOL(IN)=DOLST(IN)
      ELSE
        READ(REPLY,7,ERR=4) DEBNOL(IN)
    7   FORMAT(F8.5)
      END IF
    8 CONTINUE
    9 WRITE(5,10)
   10 FORMAT(' HOW MANY GAMMA? [10] ',$)
      READ(5,11,ERR=9) NG
   11 FORMAT(I2)
      IF(NG.EQ.0) NG=10
      IF(NG.LT.0 .OR. NG.GT.10) GOTO 9
      DO 15 IN=1,NG
   12 WRITE(5,13) IN,DGAM(IN)
   13 FORMAT(3X,'GAMMA(',I2,') [',F5.3,']: ',$)
      READ(5,14,ERR=12) GAMLST(IN)
   14 FORMAT(F8.6)
      IF(GAMLST(IN).EQ.0.D0) GAMLST(IN)=DGAM(IN)
      IF(GAMLST(IN).LE.0.D0 .OR. GAMLST(IN).GT.1.D0) GOTO 12
   15 CONTINUE
   16 WRITE(5,17)
   17 FORMAT(' HOW MANY EB/NJ? [126] ',$)
      READ(5,18,ERR=16) NJ
   18 FORMAT(I3)
      IF(NJ.EQ.0) NJ=126
```



```
      IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 16
   19 WRITE(5,20)
   20 FORMAT(' STARTING VALUE FOR EB/NJ [0 dB]: ',$)
      READ(5,21,ERR=19) DBJO
   21 FORMAT(F5.2)
      DJ=0.D0
      IF(NJ.GT.1) THEN
   22   WRITE(5,23)
   23   FORMAT(' INCREMENT FOR EB/NJ [0.4 dB]: ',$)
        READ(5,24,ERR=22) DJ
   24   FORMAT(F5.0)
        IF(DJ.EQ.0.D0) DJ=0.4D0
      END IF
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:17    17-Jul-86          Page 5
SELFL2M2.FTN;13          /F77/TR:BLOCKS/WR

0001            SUBROUTINE PSUBE(IQ,PE)
        C
        C COMPUTE ERROR PROBABILITY
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            PARAMETER (SLOTS=2400.D0)
0004            PARAMETER (SLOTS1=2399.D0)
0005            PARAMETER (SLOTPR=5757600.D0)
0006            DIMENSION WORK(20), STACK(20), HEAP(20)
0007            EXTERNAL DG16, F01A, F01B, F10A, F01C, F01D, F10B, F10C
0008            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
        C
        C COMPUTE ELEMENTAL EVENT PROBABILITIES
        C
0009            Q=IQ
0010            PI2=Q*(Q-1.D0)/SLOTPR
0011            PI1=Q*(SLOTS-Q)/SLOTPR
0012            PI0=(SLOTS-Q)*(SLOTS1-Q)/SLOTPR
0013            PART=PI0*PI0*0.5D0*DEXP(-RHON)*(1.D0+RHON/3.D0)
0014            PB=PART
0015            CALL ADQUAD(0.D0,1.D0,PART,DG16,F01A,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0016            IF(KODE.NE.0) STOP 'ADQUAD ERROR FIRST CALL'
0017            PART=2.D0*PI0*PI1*PART
0018            PB=PB+PART
0019            CALL ADQUAD(0.D0,1.D0,PART,DG16,F01B,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0020            IF(KODE.NE.0) STOP 'ADQUAD ERROR SECOND CALL'
0021            PART=PI1*PI1*PART
0022            PB=PB+PART
0023            CALL ADQUAD(0.D0,1.D0,PART,DG16,F10A,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0024            IF(KODE.NE.0) STOP 'ADQUAD ERROR THIRD CALL'
0025            PART=2.D0*PI0*PI1*PART
0026            PB=PB+PART
0027            RDIFF=RHON-RHOT
0028            RSUM=RHON+RHOT
0029            PART=DEXP(-RHOT)*(RHON*RDIFF-RSUM)
0030            PART=PART+DEXP(-RHON)*(RHOT*RDIFF+RSUM)
0031            PART=PART/DXI(RDIFF,3)
0032            PART=2.D0*PI0*PI2*PART
0033            PB=PB+PART
0034            CALL ADQUAD(0.D0,1.D0,PART,DG16,F01C,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0035            IF(KODE.NE.0) STOP 'ADQUAD ERROR FOURTH CALL'
0036            PART=2.D0*PI1*PI1*PART
0037            PB=PB+PART
0038            CALL ADQUAD(0.D0,1.D0,PART,DG16,F01D,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0039            IF(KODE.NE.0) STOP 'ADQUAD ERROR FIFTH CALL'
0040            PART=2.D0*PI1*PI2*PART
0041            PB=PB+PART
```



```
PDP-11 FORTRAN-77 V4.0-1        09:50:17    17-Jul-86          Page 6
SELFL2M2.FTN;13          /F77/TR:BLOCKS/WR

0042            CALL ADQUAD(0.D0,1.D0,PART,DG16,F10B,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0043            IF(KODE.NE.0) STOP 'ADQUAD ERROR SIXTH CALL'
0044            PART=PI1*PI1*PART
0045            PB=PB+PART
0046            CALL ADQUAD(0.D0,1.D0,PART,DG16,F10C,1.D-9,WORK,STACK,HEAP,
             $              20,KODE)
0047            IF(KODE.NE.0) STOP 'ADQUAD ERROR SEVENTH CALL'
0048            PART=2.D0*PI1*PI2*PART
0049            PB=PB+PART
0050            PART=0.5D0*DEXP(-RHOT)*(1.D0+RHOT/3.D0)
0051            PART=PI2*PI2*PART
0052            PB=PB+PART
0053            PE=PB
0054            RETURN
0055            END

PDP-11 FORTRAN-77 V4.0-1        09:50:23    17-Jul-86          Page 7
SELFL2M2.FTN;13          /F77/TR:BLOCKS/WR

0001            DOUBLE PRECISION FUNCTION P101(Z)
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
0004            D=1.D0+(BIGK-1.D0)*Z
0005            X=BIGK*RHON*Z/D
0006            P101=BIGK*DEXP(X-RHON)*(1.D0+X)/(D*D)
0007            RETURN
0008            END

PDP-11 FORTRAN-77 V4.0-1        09:50:25    17-Jul-86          Page 8
SELFL2M2.FTN;13          /F77/TR:BLOCKS/WR

0001            DOUBLE PRECISION FUNCTION P110(Z)
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
0004            D=BIGK-(BIGK-1.D0)*Z
0005            X=RHOT*Z/D
0006            P110=BIGK*DEXP(X-RHOT)*(1.D0+X)/(D*D)
0007            RETURN
0008            END

PDP-11 FORTRAN-77 V4.0-1        09:50:26    17-Jul-86          Page 9
SELFL2M2.FTN;13          /F77/TR:BLOCKS/WR

0001            DOUBLE PRECISION FUNCTION F01A(V)
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
0004            F01A=(1.D0-V)*DEXP(-RHON*V)*P101(V)
0005            RETURN
0006            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:27    17-Jul-86        Page 10
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F01B(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            OMV=1.D0-V
            D=V+BIGK*OMV
            F01B=P101(V)*BIGK*DEXP(-V*RHON/D)*OMV/D
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:29    17-Jul-86        Page 11
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F10A(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            F10A=(1.D0-V)*DEXP(-RHON*V)*P110(V)
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:31    17-Jul-86        Page 12
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION P110I(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            OMV=1.D0-V
            D=BIGK*V+OMV
            P110I=OMV*DEXP(-BIGK*RHOT*V/D)/D
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:32    17-Jul-86        Page 13
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F01C(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            F01C=P101(V)*P110I(V)
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:34    17-Jul-86        Page 14
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F01D(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            F01D=(1.D0-V)*DEXP(-RHOT*V)*P101(V)
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:35    17-Jul-86        Page 15
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F10B(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            F10B=P110(V)*P101I(V)
            RETURN
            END
```

```
PDP-11 FORTRAN-77 V4.0-1        09:50:37    17-Jul-86        Page 16
SELFL2M2.FTN;13         /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION F10C(V)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            F10C=(1.D0-V)*DEXP(-RHOT*V)*P110(V)
            RETURN
            END
```

```
      SUBROUTINE ADQUAD(XL,XU,Y,QR,F,TOL,WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C             WITH CALLING SEQUENCE
C             CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
C         SAME ARRAY AS WORK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C          2 -- EPS DIVIDED TO ZERO, EITHER ASKING FOR TOO
C               TIGHT A TOLERANCE OR ROUND-OFF PREVENTS
C               ATTAINING REQUIRED ACCURACY
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      IF(DABS(T-P1-P2).LE.EPS) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=EPS/2.D0
```



```
      IF(EPS.EQ.0.D0) THEN
        KODE=2
        RETURN
      END IF
      STACK(NPTS)=EPS
      GOTO 10
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```

```
0001            SUBROUTINE DG16(A,B,F,ANSWER)
        C
        C 16-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
        C
        C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
        C
        C R. H. FRENCH, 28 FEBRUARY 1986
        C
0002            IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003            DIMENSION X(8),W(8)
        C
        C WEIGHTS AND ABSCISSAS FOR 16-POINT GAUSSIAN QUADRATURE
        C
0004            DATA X/ 0.09501250983763744018500,
             $          0.28160355077925891323000,
             $          0.45801677765722738634200,
             $          0.61787624440264374844700,
             $          0.75540440835500303389500,
             $          0.86563120238783174388000,
             $          0.94457502307323257607800,
             $          0.98940093499164993259600 /
0005            DATA W/ 0.18945061045506849628500,
             $          0.18260341504492358886700,
             $          0.16915651939500253818900,
             $          0.14959598881657673208100,
             $          0.12462897125553387205200,
             $          0.09515851168249278481000,
             $          0.06225352393864789286300,
             $          0.02715245941175409485200 /
0006            ANSWER=0.D0
0007            BMA02=(B-A)/2.D0
0008            BPA02=(B+A)/2.D0
0009            DO 10 I=1,8
0010            C=X(I)*BMA02
0011            Y1=BPA02+C
0012            Y2=BPA02-C
0013            ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
0014    10      CONTINUE
0015            ANSWER=ANSWER*BMA02
0016            RETURN
0017            END
```

APPENDIX L

COMPUTER PROGRAM FOR

SELF-NORMALIZING RECEIVER WITH M=2, L=3

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the self-normalizing receiver when M=2 and L=3. The program searches numerically for the worst-case jamming fraction.

```
0001          PROGRAM SELF23
        C
        C SELF NORMALIZING RECEIVER, M=2, L=3
        C
        C ANALYSIS: L.E. MILLER
        C PROGRAM:  R.H. FRENCH
        C
        C METHOD: SELF-CONVOLUTION OF THE MARGINAL DENSITY
        C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          PARAMETER(M=2, L=3, NSLOTS=2400, SLOTS=2400.D0)
0004          PARAMETER (SLOTPR=5757600.D0)
0005          LOGICAL TEST
0006          CHARACTER*13 FNAME
0007          REAL DEBNJ(126), PELOG(126), QOPT(126)
0008          VIRTUAL PEOLD(2400)
0009          COMMON /PIES/ PI0, PI1, PI2
0010          COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
0011          COMMON /ROWCON/ A0N, A1N, A2N, A3N, A0T, A1T, A2T, A3T,
          $                 B0, B1, C0, C1
0012          COMMON /PARMS/ NO, NJ, DEBNOL(5),
          $                 DEBNJL(126), DJ, DBJ0
0013          CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0014          CALL GET
0015          DO 900 IO=1,NO
0016          EBNO=10.D0**(DEBNOL(IO)/10.D0)
0017          RHON=EBNO/L
0018          A0N=1.D0+RHON+RHON*RHON/6.D0
0019          A1N=RHON*RHON/2.D0-1.D0
0020          A2N=-RHON*(1.D0+RHON/2.D0)
0021          A3N=-RHON*RHON/6.D0
0022          IOOUT=DEBNOL(IO)
        C
        C PROGRESS FILE
        C
0023          WRITE(FNAME,1) IOOUT
0024       1  FORMAT('S23',I2.2,'GOPT.DAT')
0025          WRITE(5,2) FNAME
0026       2  FORMAT(' WORKING ON FILE ',A13)
0027          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ERR=810,
          $       FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
        C
        C HAVE A PROGRESS FILE, READ IT
        C
0028          READ(4) EBNOIN, DBJOIN, DJIN
0029          IF(EBNOIN.NE.DEBNOL(IO) .OR.
          $      DBJOIN.NE.DBJ0 .OR. DJIN.NE.DJ) STOP 'FILE CORRUPT'
0030          JJ=0
0031     801  JJ=JJ+1
0032          READ(4,END=802) DEBNJ(JJ), PELOG(JJ), QOPT(JJ)
0033          GOTO 801
0034     802  CLOSE(UNIT=4)
0035          GOTO 820
```



```
        C
        C NO FILE, MUST CREATE IT
        C
0036     810  OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED',
          $       ACCESS='SEQUENTIAL')
0037          WRITE(4) DEBNOL(IO), DBJ0, DJ
0038          CLOSE(UNIT=4)
0039          JJ=1
        C
        C KEEP ON GOING
        C
0040     820  DO 700 IJ=JJ,NJ
        C INVALIDATE SAVED PE ARRAY
0041          DO 737 IJK=1,2400
0042          PEOLD(IJK)=-1.D0
0043     737  CONTINUE
0044          IF(IJ.GE.3) THEN
0045            IDQ=QOPT(IJ-1)-QOPT(IJ-2)+0.5D0
0046            IF(IDQ.EQ.0) IDQ=1
0047            Q=QOPT(IJ-1)
0048            IQ=Q
0049          ELSE
0050            IDQ=1
0051            Q=1.D0
0052            IQ=1
0053          END IF
0054          DQ=IDQ
0055          WRITE(5,821) IJ
0056     821  FORMAT(' IJ = ',I3)
0057          DBEBNJ=DBJ0+(IJ-1)*DJ
0058          DEBNJ(IJ)=DBEBNJ
0059          EBNJ=10.D0**(DBEBNJ/10.D0)
        C PRIME THE ALGORITHM WITH DUMMY OLD VALUES OF P(E)
0060          P1=0.D0
0061          P2=0.D0
0062     709  GAMMA=Q/SLOTS
0063          WRITE(5,828) IQ
0064     828  FORMAT(5X,'Q=',I5)
0065          PI0=(SLOTS-Q)*(SLOTS-1.D0-Q)/SLOTPR
0066          PI1=Q*(SLOTS-Q)/SLOTPR
0067          PI2=Q*(Q-1.D0)/SLOTPR
0068          RHOJ=EBNJ/L
0069          RHOJ1=GAMMA*RHOJ
0070          RHOT=RHOJ1*RHON/(RHOJ1+RHON)
0071          BIGK=RHON/RHOT
0072          BIGK1=BIGK-1.D0
0073          A0T=1.D0+RHOT+RHOT*RHOT/6.D0
0074          A1T=0.5D0*RHOT*RHOT-1.D0
0075          A2T=-RHOT*(1.D0+0.5D0*RHOT)
0076          A3T=-RHOT*RHOT/6.D0
0077          B0=RHON*RHON*(RHON-RHOT)-2.D0*RHON*RHOT
0078          B1=RHOT*(RHON-RHOT)*(RHON*RHON-RHON*RHOT-RHOT)
```

```
            C0=RHOT*RHOT*(RHON-RHOT)+2.D0*RHON*RHOT
            C1=RHON*(RHON-RHOT)*(RHOT*RHOT-RHON*RHOT-RHON)
            CALL PSUBE(PE,PEOLD,IQ)
            P3=PE
            IF(P3.GT.P2 .AND. IQ.LT.NSLOTS) THEN
C KEEP ON GOING, WE ARE NOT PAST THE MAXIMUM
              P1=P2
              P2=P3
              IQ=MIN0(IQ+IDQ,NSLOTS)
              Q=DMIN1(Q+DQ,SLOTS)
              GOTO 709
            ELSE
              PMAX=DMAX1(P1,P2,P3)
              EPS=0.001D0*PMAX
              TEST=(DABS(P1-P2).LE.EPS .AND. DABS(P1-P3).LE.EPS .AND.
     $              DABS(P2-P3).LE.EPS)
              IF(TEST .OR. IDQ.EQ.1
     $              .OR. ((.NOT.TEST) .AND. IQ.EQ.NSLOTS)) THEN
C             WE ARE DONE WHEN ALL 3 ARE CLOSE TOGETHER OR WHEN DQ=1
C             OR WHEN WE REACHED FULL-BAND JAMMING AND P(E) IS STILL
C             INCREASING
              POPT=PMAX
                IF(P2.GT.P3) THEN
C               THE OPTIMUM MUST BE THE MIDDLE POINT OF THE 3
                  QOPT(IJ)=Q-DQ
                  IF(QOPT(IJ).EQ.0.D0) QOPT(IJ)=1.D0
C PREVENT ROUND-OFF FROM MAKING QOPT VS. EB/NJ NON-MONOTONIC
                  IF(IJ.GT.1) THEN
                    IF(QOPT(IJ).LT.QOPT(IJ-1)) QOPT(IJ)=QOPT(IJ-1)
                  END IF
                ELSE
C                 THE OPTIMUM IS FULL-BAND JAMMING
                  QOPT(IJ)=NSLOTS
                END IF
                GOTO 665
              ELSE
C               NOT LOCATED SUFFICIENTLY ACCURATELY, CUT DQ AND TRY AGAIN
                Q=Q-DQ-DQ
                IQ=IQ-IDQ-IDQ
                IDQ=IDQ/2
                DQ=IDQ
                P2=P1
                P1=0.D0
                Q=Q+DQ
                IQ=IQ+IDQ
                GOTO 709
              END IF
            END IF
  665       PELOG(IJ)=DLOG10(POPT)
            OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
     $           FORM='UNFORMATTED')
            WRITE(4) DEBNJ(IJ), PELOG(IJ), QOPT(IJ)
```



```
            CLOSE(UNIT=4)
  700       CONTINUE
            OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',ACCESS='SEQUENTIAL',
     $           FORM='UNFORMATTED')
            WRITE(4) M, L, DEBNOL(IO), NSLOTS, NJ, DEBNJ, PELOG, QOPT
            CLOSE(UNIT=4)
            WRITE(6,710) M, L, DEBNOL(IO), NSLOTS,
     $                   (DEBNJ(IJ),10.D0**PELOG(IJ),QOPT(IJ),IJ=1,NJ)
  710       FORMAT('1SELF-NORMALIZING RECEIVER FOR M=',I1,' AND L=',I1/
     $      5X,'EB/NO = ',F8.5,' dB',5X,'NSLOTS=',I4/
     $      ' EB/NJ (dB)',8X,'P(E)',8X,'QOPT'/<NJ>(4X,0PF4.1,8X,1PD10.3,
     $       3X,0PF5.0/))
  800       CONTINUE
  900       CONTINUE
            STOP 'PLEASE PURGE S23*.DAT'
            END
```

```
      SUBROUTINE GET
C
C INTERACTIVE RUN PARAMETER INPUTS
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION DOLST(5)
      CHARACTER*8 REPLY, BLANKS
      COMMON /PARMS/ NO, NJ, DEBNOL(5),
     $               DEBNJL(126), DJ, DBJO
      DATA DOLST/ 13.35247D0, 12.31330D0, 10.94443D0, 14.89253D0,
     $            16.027135D0/
      DATA BLANKS/'        '/
    1 WRITE(5,2)
    2 FORMAT(' HOW MANY EB/NO? [1] ',$)
      READ(5,3,ERR=1) NO
    3 FORMAT(I1)
      IF(NO.EQ.0) NO=1
      IF(NO.LT.0 .OR. NO.GT.5) GOTO 1
      DO 8 IN=1,NO
    4 WRITE(5,5) IN, DOLST(IN)
    5 FORMAT(3X,'EB/NO(',I1,') [',F8.5,' dB]: ',$)
      READ(5,6,ERR=4) REPLY
    6 FORMAT(A8)
      IF(REPLY.EQ.BLANKS) THEN
        DEBNOL(IN)=DOLST(IN)
      ELSE
        READ(REPLY,7,ERR=4) DEBNOL(IN)
    7   FORMAT(F8.5)
      END IF
    8 CONTINUE
   16 WRITE(5,17)
   17 FORMAT(' HOW MANY EB/NJ? [51] ',$)
      READ(5,18,ERR=16) NJ
   18 FORMAT(I3)
      IF(NJ.EQ.0) NJ=51
      IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 16
   19 WRITE(5,20)
   20 FORMAT(' STARTING VALUE FOR EB/NJ [50 dB]: ',$)
      READ(5,21,ERR=19) DBJO
   21 FORMAT(F5.2)
      IF(DBJO.EQ.0.D0) DBJO=50.D0
      DJ=0.D0
      IF(NJ.GT.1) THEN
   22   WRITE(5,23)
   23   FORMAT(' INCREMENT FOR EB/NJ [-1 dB]: ',$
        READ(5,24,ERR=22) DJ
   24   FORMAT(F5.0)
        IF(DJ.EQ.0.D0) DJ=-1.D0
      END IF
      RETURN
      END
```



```
      SUBROUTINE PSUBE(PE,PEOLD,IQ)
C
C COMPUTE ERROR PROBABILITY
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      VIRTUAL PEOLD(2400)
      DIMENSION WORK(20), STACK(20), HEAP(20)
      EXTERNAL DG10A, PDF
      IF(PEOLD(IQ).NE.-1.D0) THEN
C
C WE HAVE A SAVED VALUE FROM A LARGER STEP SIZE
C
        PE=PEOLD(IQ)
      ELSE
C
C COMPUTE ELEMENTAL EVENT PROBABILITIES
C
        CALL ADQUA1(0.D0,1.5D0,PE,DG10A,PDF,1.D-3,1.D-1,
     $              WORK,STACK,HEAP,20,KODE)
        IF(KODE.NE.0) STOP 'ADQUA1 ERROR'
        PEOLD(IQ)=PE
      END IF
      RETURN
      END
```

```
          SUBROUTINE ADQUA1(XL,XU,Y,QR,F,TOL,ABSTOL,
     $                      WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C               WITH CALLING SEQUENCE
C               CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C ABSTOL-ABSOLUTE ERROR TOLERANCE (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
C        SAME ARRAY AS WORK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C R. H. FRENCH, 14 AUGUST 1984
C
          IMPLICIT DOUBLE PRECISION(A-H,O-Z)
          EXTERNAL F
          DIMENSION WORK(N),STACK(N),HEAP(N)
          KODE=0
          Y=0.D0
          WORK(1)=XU
          CALL QR(XL,XU,F,T)
          HEAP(1)=T
          A=XL
          NPTS=1
          EPS=TOL
          STACK(1)=EPS
 10       B=WORK(NPTS)
          XM=(A+B)*0.5D0
          CALL QR(A,XM,F,P1)
          CALL QR(XM,B,F,P2)
          TEST=DMAX1(EPS*DABS(T),ABSTOL)
          IF(DABS(T-P1-P2).LE.TEST .OR. DABS(T).LE.ABSTOL) GOTO 20
C SPLIT IT
          NPTS=NPTS+1
          IF(NPTS.GT.N) THEN
            KODE=1
            RETURN
          END IF
          WORK(NPTS)=XM
          HEAP(NPTS)=P2
          T=P1
          EPS=DMAX1(EPS/2.D0,5.D-16)
```



```
          STACK(NPTS)=EPS
          GOTO 10
C FINISHED A PIECE
 20       Y=Y+P1+P2
          EPS=STACK(NPTS)
          T=HEAP(NPTS)
          NPTS=NPTS-1
          A=B
          IF(NPTS.EQ.0) RETURN
          GOTO 10
          END
```

```
0001            SUBROUTINE ADQUA2(XL,XU,Y,QR,F,TOL,ABSTOL,
          $                       WORK,STACK,HEAP,N,KODE)
        C
        C ADAPTIVE QUADRATURE ALGORITHM
        C   XL - LOWER LIMIT OF INTEGRAL (IN)
        C   XU - UPPER LIMIT OF INTEGRAL (IN)
        C    Y - VALUE OF INTEGRAL (OUT)
        C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
        C              WITH CALLING SEQUENCE
        C              CALL QR(XL,XU,F,Y)
        C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
        C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
        C ABSTOL-ABSOLUTE ERROR TOLERANCE (IN)
        C WORK - WORK ARRAY OF SIZE N (IN)
        C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
        C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
        C        SAME ARRAY AS WORK (IN)
        C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
        C KODE - ERROR INDICATOR (OUT)
        C          0 -- NO ERROR
        C          1 -- WORK ARRAYS TOO SMALL
        C R. H. FRENCH, 14 AUGUST 1984
        C
0002            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0003            EXTERNAL F
0004            DIMENSION WORK(N),STACK(N),HEAP(N)
0005            KODE=0
0006            Y=0.D0
0007            WORK(1)=XU
0008            CALL QR(XL,XU,F,T)
0009            HEAP(1)=T
0010            A=XL
0011            NPTS=1
0012            EPS=TOL
0013            STACK(1)=EPS
0014    10      B=WORK(NPTS)
0015            XM=(A+B)*0.5D0
0016            CALL QR(A,XM,F,P1)
0017            CALL QR(XM,B,F,P2)
0018            TEST=DMAX1(EPS*DABS(T),ABSTOL)
0019            IF(DABS(T-P1-P2).LE.TEST .OR. DABS(T).LE.ABSTOL) GOTO 20
        C SPLIT IT
0020            NPTS=NPTS+1
0021            IF(NPTS.GT.N) THEN
0022              KODE=1
0023              RETURN
0024            END IF
0025            WORK(NPTS)=XM
0026            HEAP(NPTS)=P2
0027            T=P1
0028            EPS=DMAX1(EPS/2.D0,5.D-16)
```



```
0029            STACK(NPTS)=EPS
0030            GOTO 10
        C FINISHED A PIECE
0031    20      Y=Y+P1+P2
0032            EPS=STACK(NPTS)
0033            T=HEAP(NPTS)
0034            NPTS=NPTS-1
0035            A=B
0036            IF(NPTS.EQ.0) RETURN
0037            GOTO 10
0038            END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:10    17-Jul-86              Page 11
SELFL3M2B.FTN;16         /F77/WR

                SUBROUTINE ADQUA3(XL,XU,Y,QR,F,TOL,ABSTOL,
             $                    WORK,STACK,HEAP,N,KODE)
          C
          C ADAPTIVE QUADRATURE ALGORITHM
          C   XL - LOWER LIMIT OF INTEGRAL (IN)
          C   XU - UPPER LIMIT OF INTEGRAL (IN)
          C    Y - VALUE OF INTEGRAL (OUT)
          C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
          C             WITH CALLING SEQUENCE
          C             CALL QR(XL,XU,F,Y)
          C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
          C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
          C ABSTOL-ABSOLUTE ERROR TOLERANCE (IN)
          C WORK - WORK ARRAY OF SIZE N (IN)
          C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
          C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK
          C        SAME ARRAY AS WORK (IN)
          C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
          C KODE - ERROR INDICATOR (OUT)
          C          0 -- NO ERROR
          C          1 -- WORK ARRAYS TOO SMALL
          C R. H. FRENCH, 14 AUGUST 1984
          C
                IMPLICIT DOUBLE PRECISION(A-H,O-Z)
                EXTERNAL F
                DIMENSION WORK(N),STACK(N),HEAP(N)
                KODE=0
                Y=0.D0
                WORK(1)=XU
                CALL QR(XL,XU,F,T)
                HEAP(1)=T
                A=XL
                NPTS=1
                EPS=TOL
                STACK(1)=EPS
          10    B=WORK(NPTS)
                XM=(A+B)*0.5D0
                CALL QR(A,XM,F,P1)
                CALL QR(XM,B,F,P2)
                TEST=DMAX1(EPS*DABS(T),ABSTOL)
                IF(DABS(T-P1-P2).LE.TEST .OR. DABS(T).LE.ABSTOL) GOTO 20
          C SPLIT IT
                NPTS=NPTS+1
                IF(NPTS.GT.N) THEN
                  KODE=1
                  RETURN
                END IF
                WORK(NPTS)=XM
                HEAP(NPTS)=P2
                T=P1
                EPS=DMAX1(EPS/2.D0,5.D-16)
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:10    17-Jul-86              Page 12
SELFL3M2B.FTN;16         /F77/WR

                STACK(NPTS)=EPS
                GOTO 10
          C FINISHED A PIECE
          20    Y=Y+P1+P2
                EPS=STACK(NPTS)
                T=HEAP(NPTS)
                NPTS=NPTS-1
                A=B
                IF(NPTS.EQ.0) RETURN
                GOTO 10
                END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:13    17-Jul-86              Page 13
SELFL3M2B.FTN;16         /F77/WR

                BLOCK DATA
                IMPLICIT DOUBLE PRECISION(A-H,O-Z)
                COMMON /GQWTS/ X(5),W(5)
          C
          C WEIGHTS AND ABSCISSAS FOR 10-POINT GAUSSIAN QUADRATURE
          C
                DATA X/ 0.1488743389816310D0,
             $          0.4333953941292470D0,
             $          0.6794095682990240D0,
             $          0.8650633666889850D0,
             $          0.9739065285171720D0 /
                DATA W/ 0.2955242247147530D0,
             $          0.2692667193099960D0,
             $          0.2190863625159820D0,
             $          0.1494513491505810D0,
             $          0.0666713443086880D0 /
                END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:14    17-Jul-86             Page 14
SELFL3M2B.FTN;16         /F77/WR

              SUBROUTINE DG10A(A,B,F,ANSWER)
      C
      C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
      C
      C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
      C
      C R. H. FRENCH, 30 MAY 1986
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /GQWTS/ X(5),W(5)
              ANSWER=0.D0
              BMAO2=(B-A)/2.D0
              BPAO2=(B+A)/2.D0
              DO 10 I=1,5
              C=X(I)*BMAO2
              Y1=BPAO2+C
              Y2=BPAO2-C
              ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
      10      CONTINUE
              ANSWER=ANSWER*BMAO2
              RETURN
              END
```



```
PDP-11 FORTRAN-77 V4.0-1        10:02:16    17-Jul-86             Page 15
SELFL3M2B.FTN;16         /F77/WR

              SUBROUTINE DG10B(A,B,F,ANSWER)
      C
      C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
      C
      C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
      C
      C R. H. FRENCH, 30 MAY 1986
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /GQWTS/ X(5),W(5)
              ANSWER=0.D0
              BMAO2=(B-A)/2.D0
              BPAO2=(B+A)/2.D0
              DO 10 I=1,5
              C=X(I)*BMAO2
              Y1=BPAO2+C
              Y2=BPAO2-C
              ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
      10      CONTINUE
              ANSWER=ANSWER*BMAO2
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:18    17-Jul-86             Page 16
SELFL3M2B.FTN;16         /F77/WR

              SUBROUTINE DG10C(A,B,F,ANSWER)
      C
      C 10-POINT GAUSSIAN INTEGRATION OVER ARBITRARY INTERVAL
      C
      C REF.: ABRAMOWITZ & STEGUN, EQ. 25.4.30 AND TABLE 25.4
      C
      C R. H. FRENCH, 30 MAY 1986
      C
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /GQWTS/ X(5),W(5)
              ANSWER=0.D0
              BMAO2=(B-A)/2.D0
              BPAO2=(B+A)/2.D0
              DO 10 I=1,5
              C=X(I)*BMAO2
              Y1=BPAO2+C
              Y2=BPAO2-C
              ANSWER=ANSWER+W(I)*(F(Y1)+F(Y2))
      10      CONTINUE
              ANSWER=ANSWER*BMAO2
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:02:20    17-Jul-86             Page 17
SELFL3M2B.FTN;16         /F77/WR

              DOUBLE PRECISION FUNCTION PDF(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              PARAMETER (CUSP=1.D0)
              EXTERNAL DG10B, CONV02
              DIMENSION WORK(20), STACK(20), HEAP(20)
              COMMON /PASSZ/ ZEE
              ZEE=Z
              XL=DMAX1(0.D0,Z-1.D0)
              XU=DMIN1(2.D0,Z)
              IF(XL.GE.CUSP .OR. XU.LE.CUSP) THEN
                CALL ADQUA2(XL,XU,PDF,DG10B,CONV02,1.D-4,1.D-10,
           $                WORK, STACK, HEAP, 20, KODE)
                IF(KODE.NE.0) STOP 'ADQUA2 ERROR'
              ELSE
                CALL ADQUA2(XL,CUSP,PX,DG10B,CONV02,1.D-4,1.D-10,
           $                WORK,STACK,HEAP,20,KODE)
                IF(KODE.NE.0) STOP 'ADQUA2-1 ERROR'
                CALL ADQUA2(CUSP,XU,PY,DG10B,CONV02,1.D-4,1.D-10,
           $                WORK,STACK,HEAP,20,KODE)
                IF(KODE.NE.0) STOP 'ADQUA2-2 ERROR'
                PDF=PX+PY
              END IF
              RETURN
              END
```

PDP-11 FORTRAN-77 V4.0-1 10:02:22 17-Jul-86 Page 18
SELFL3M2B.FTN;16 /F77/WR

```
      DOUBLE PRECISION FUNCTION CONVO2(W)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL DG10C, F0001, F0010, F0101, F0110, F0111, F1010,
     $         F1011
      DIMENSION WORK(20), STACK(20), HEAP(20)
      COMMON /PASSW/ DUBEYU
      COMMON /PASSZ/ ZEE
      COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
      COMMON /ROWCON/ A0N, A1N, A2N, A3N, A0T, A1T, A2T, A3T,
     $                B0, B1, C0, C1
      COMMON /PIES/ PI0, PI1, PI2
      DUBEYU=W
      RZN=RHON*W
      RZT=RHOT*W
      R12=RHON-RHOT
      RSUM=RHOT+RHON
      TRNT=2.D0*RHON*RHOT
      IF(W.LT.1.D0) THEN
        P2=PI0*PI0*DEXP(RZN-2.D0*RHON)*W*(1.D0+RZN+RZN*RZN/6.D0)
     $    +PI2*PI2*DEXP(RZT-2.D0*RHOT)*W*(1.D0+RZT+RZT*RZT/6.D0)
     $    +2.D0*PI0*PI2*(DEXP(RZN-RSUM)*(RHON*RHON*R12*W-TRNT)
     $                  +DEXP(RZT-RSUM)*(RHOT*RHOT*R12*W+TRNT))/
     $                  DXI(R12,3)
      ELSE
        ZM1=W-1.D0
        ZM2=W-2.D0
        P2=PI0*PI0*DEXP(RZN-2.D0*RHON)
     $       *(A0N+(A1N+(A2N+A3N*ZM1)*ZM1)*ZM1)
     $    +PI2*PI2*DEXP(RZT-2.D0*RHOT)
     $       *(A0T+(A1T+(A2T+A3T*ZM1)*ZM1)*ZM1)
     $    +2.D0*PI0*PI2*(DEXP(RHOT*ZM2)*(B0+B1*ZM1)
     $                  +DEXP(RHON*ZM2)*(C0+C1*ZM1))/DXI(R12,3)
      END IF
      YL=DMAX1(0.D0,W-1.D0)
      YU=DMIN1(1.D0,W)
      IF(PI0.NE.0.D0 .AND. PI1.NE.0.D0) THEN
        CALL ADQUA3(YL,YU,PART,DG10C,F0001,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-1'
        P2=P2+2.D0*PI0*PI1*PART
        CALL ADQUA3(YL,YU,PART,DG10C,F0010,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-2'
        P2=P2+2.D0*PI0*PI1*PART
      END IF
      IF(PI1.NE.0.D0) THEN
        CALL ADQUA3(YL,YU,PART,DG10C,F0101,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-3'
        P2=P2+2.D0*PI1*PI1*PART
        CALL ADQUA3(YL,YU,PART,DG10C,F0110,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
```

PDP-11 FORTRAN-77 V4.0-1 10:02:22 17-Jul-86 Page 19
SELFL3M2B.FTN;16 /F77/WR

```
        IF(KODE.NE.0) STOP 'ADQUA3-4'
        P2=P2+PI1*PI1*PART
        CALL ADQUA3(YL,YU,PART,DG10C,F1010,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-6'
        P2=P2+PI1*PI1*PART
      END IF
      IF(PI1.NE.0.D0 .AND. PI2.NE.0.D0) THEN
        CALL ADQUA3(YL,YU,PART,DG10C,F0111,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-5'
        P2=P2+2.D0*PI1*PI2*PART
        CALL ADQUA3(YL,YU,PART,DG10C,F1011,1.D-6,1.D-11,
     $              WORK, STACK, HEAP, 20, KODE)
        IF(KODE.NE.0) STOP 'ADQUA3-7'
        P2=P2+2.D0*PI1*PI2*PART
      END IF
      CONVO2=P2*P1(ZEE-W)
      RETURN
      END
```

PDP-11 FORTRAN-77 V4.0-1 10:02:30 17-Jul-86 Page 20
SELFL3M2B.FTN;16 /F77/WR

```
      DOUBLE PRECISION FUNCTION P1(Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
      COMMON /PIES/ PI0, PI1, PI2
      RNZ=RHON*Z
      RTZ=RHOT*Z
      P1=PI0*DEXP(RNZ-RHON)*(1.D0+RNZ)
      BK1Z=BIGK1*Z
      D=1.D0+BK1Z
      A=BIGK*RNZ/D
      P01=BIGK*DEXP(A-RHON)*((1.D0+A)/D)/D
      D=BIGK-BK1Z
      A=RTZ/D
      P1=P1+PI1*(P01+BIGK*DEXP(A-RHOT)*((1.D0+A)/D)/D)
      P1=P1+PI2*DEXP(RTZ-RHOT)*(1.D0+RTZ)
      RETURN
      END
```

PDP-11 FORTRAN-77 V4.0-1 10:02:32 17-Jul-86 Page 21
SELFL3M2B.FTN;16 /F77/WR

```
      DOUBLE PRECISION FUNCTION F0001(Z)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /PASSW/ DUBEYU
      F0001=P00(Z)*P01(DUBEYU-Z)
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:34    17-Jul-86          Page 22
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F0010(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F0010=P00(Z)*P10(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:35    17-Jul-86          Page 23
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F0101(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F0101=P01(Z)*P01(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:37    17-Jul-86          Page 24
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F0110(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F0110=P01(Z)*P10(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:36    17-Jul-86          Page 25
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F0111(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F0111=P01(Z)*P11(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:40    17-Jul-86          Page 26
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F1010(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F1010=P10(Z)*P10(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:41    17-Jul-86          Page 27
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION F1011(Z)
              IMPLICIT DOUBLE PRECISION (A-H,O-Z)
              COMMON /PASSW/ DUBEYU
              F1011=P10(Z)*P11(DUBEYU-Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:42    17-Jul-86          Page 28
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION P00(Z)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
              P00=DEXP(RHON*Z-RHON)*(1.D0+RHON*Z)
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:44    17-Jul-86          Page 29
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION P01(Z)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
              RNZ=RHON*Z
              BK1Z=BIGK1*Z
              D=1.D0+BK1Z
              A=BIGK*RNZ/D
              P01=BIGK*DEXP(A-RHON)*((1.D0+A)/D)/D
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1          10:02:45    17-Jul-86          Page 30
SELFL3M2B.FTN;16          /F77/WR

              DOUBLE PRECISION FUNCTION P10(Z)
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
              RTZ=RHOT*Z
              BK1Z=BIGK1*Z
              D=BIGK-BK1Z
              A=RTZ/D
              P10=BIGK*DEXP(A-RHOT)*((1.D0+A)/D)/D
              RETURN
              END
```

```
      DOUBLE PRECISION FUNCTION P11(Z)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK, BIGK1
      P11=DEXP(RHOT*Z-RHOT)*(1.D0+RHOT*Z)
      RETURN
      END
```

APPENDIX M

COMPUTER PROGRAM FOR PRACTICAL ADAPTIVE GAIN CONTROL RECEIVER

The following pages contain the source code listing for the FORTRAN-77 program used for numerical computations of the performance of the practical adaptive gain control receiver for FH/RMFSK.

```
0001          PROGRAM PRAC22
        C
        C PRACTICAL ACJ/AGC RECEIVER, M=2, L=2
        C
        C ANALYSIS: L.E. MILLER
        C PROGRAM:  R.H. FRENCH
        C
0002          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
0003          PARAMETER(M=2, L=2, NSLOTS=2400)
0004          CHARACTER*13 FNAME
0005          REAL DEBNJ(126), PELOG(126)
0006          COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
0007          COMMON /PARMS/ NO, NJ, NG, DEBNOL(5), GAMLST(10),
           $                 DEBNJL(126), DJ, DBJO
0008          CALL ERRSET(29,.TRUE.,.FALSE.,.TRUE.,.FALSE.,15)
0009          CALL GET
0010          DO 900 IO=1,NO
0011          EBNO=10.DO**(DEBNOL(IO)/10.DO)
0012          RHON=EBNO/L
0013          IOOUT=DEBNOL(IO)
0014          DO 800 IG=1,NG
0015          GAMMA=GAMLST(IG)
0016          IQ=GAMMA*NSLOTS+0.5DO
0017          IGOUT=1000.DO*GAMMA+0.5DO
        C
        C PROGRESS FILE
        C
0018          WRITE(FNAME,1) IOOUT,IGOUT
0019        1 FORMAT('B22',I2.2,I4.4,'.DAT')
0020          WRITE(5,2) FNAME
0021        2 FORMAT(' WORKING ON FILE ',A13)
0022          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ERR=810,
           $     FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
        C
        C HAVE A PROGRESS FILE, READ IT
        C
0023          READ(4) EBNOIN, GAMMIN, DBJOIN, DJIN
0024          IF(EBNOIN.NE.DEBNOL(IO) .OR. GAMMIN.NE.GAMMA .OR.
           $     DBJOIN.NE.DBJO .OR. DJIN.NE.DJ) STOP 'FILE CORRUPT'
0025          JJ=0
0026      801 JJ=JJ+1
0027          READ(4,END=802) DEBNJ(JJ), PELOG(JJ)
0028          GOTO 801
0029      802 CLOSE(UNIT=4)
0030          GOTO 820
        C
        C NO FILE, MUST CREATE IT
        C
0031      810 OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',FORM='UNFORMATTED',
           $     ACCESS='SEQUENTIAL')
0032          WRITE(4) DEBNOL(IO), GAMMA, DBJO, DJ
0033          CLOSE(UNIT=4)
```



```
0034          JJ=1
        C
        C KEEP ON GOING
        C
0035      820 DO 700 IJ=JJ,NJ
0036          WRITE(5,821) IJ
0037      821 FORMAT(' IJ = ',I3)
0038          DBEBNJ=DBJO+(IJ-1)*DJ
0039          DEBNJ(IJ)=DBEBNJ
0040          EBNJ=10.DO**(DBEBNJ/10.DO)
0041          RHOJ=EBNJ/L
0042          RHOJ1=GAMMA*RHOJ
0043          RHOT=RHOJ1*RHON/(RHOJ1+RHON)
0044          BIGK=RHON/RHOT
0045          CALL PSUBE(IQ,PE)
0046          PELOG(IJ)=DLOG10(PE)
0047          OPEN(UNIT=4,FILE=FNAME,STATUS='OLD',ACCESS='APPEND',
           $     FORM='UNFORMATTED')
0048          WRITE(4) DEBNJ(IJ), PELOG(IJ)
0049          CLOSE(UNIT=4)
0050      700 CONTINUE
0051          OPEN(UNIT=4,FILE=FNAME,STATUS='NEW',ACCESS='SEQUENTIAL',
           $     FORM='UNFORMATTED')
0052          WRITE(4) M, L, DEBNOL(IO), GAMMA, NSLOTS, NJ,
           $         DEBNJ, PELOG
0053          CLOSE(UNIT=4)
0054          WRITE(6,710) M, L, DEBNOL(IO), GAMMA, NSLOTS,
           $           (DEBNJ(IJ),10.DO**PELOG(IJ),IJ=1,NJ)
0055      710 FORMAT('1PRACTICAL ACJ-AGC RECEIVER FOR M=',I1,
           $ ' AND L=',I1/5X,'EB/NO = ',F8.5,' dB',5X,'GAMMA=',
           $ 1PD10.3,5X,'NSLOTS=',I4/' EB/NJ (dB)',8X,'P(E)'/
           $ <NJ>(4X,OPF4.1,8X,1PD10.3/))
0056      800 CONTINUE
0057      900 CONTINUE
0058          STOP 'PLEASE PURGE B22*.DAT'
0059          END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:38:34    17-Jul-86          Page 3
PACJAGC.FTN;7             /F77/TR:BLOCKS/WR

              SUBROUTINE GET
        C
        C INTERACTIVE RUN PARAMETER INPUTS
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              DIMENSION DGAM(10), DOLST(5)
              CHARACTER*8 REPLY, BLANKS
              COMMON /PARMS/ NO, NJ, NG, DEBNOL(5), GAMLST(10),
             $              DEBNJL(126), DJ, DBJO
              DATA DGAM/1.D-3, 2.D-3, 5.D-3,
             $          1.D-2, 2.D-2, 5.D-2,
             $          1.D-1, 2.D-1, 5.D-1, 1.D0/
              DATA DOLST/ 13.35247D0, 12.3133D0, 10.94443D0,
             $            14.89253D0, 16.027135D0/
              DATA BLANKS/'        '/
            1 WRITE(5,2)
            2 FORMAT(' HOW MANY EB/NO? [1] ',$)
              READ(5,3,ERR=1) NO
            3 FORMAT(I1)
              IF(NO.EQ.0) NO=1
              IF(NO.LT.0 .OR. NO.GT.5) GOTO 1
              DO 8 IN=1,NO
            4 WRITE(5,5) IN, DOLST(IN)
            5 FORMAT(3X,'EB/NO(',I1,') [',F8.5,' dB]: ',$)
              READ(5,6,ERR=4) REPLY
            6 FORMAT(A8)
              IF(REPLY.EQ.BLANKS) THEN
                DEBNOL(IN)=DOLST(IN)
              ELSE
                READ(REPLY,7,ERR=4) DEBNOL(IN)
            7   FORMAT(F8.5)
              END IF
            8 CONTINUE
            9 WRITE(5,10)
           10 FORMAT(' HOW MANY GAMMA? [10] ',$)
              READ(5,11,ERR=9) NG
           11 FORMAT(I2)
              IF(NG.EQ.0) NG=10
              IF(NG.LT.0 .OR. NG.GT.10) GOTO 9
              DO 15 IN=1,NG
           12 WRITE(5,13) IN,DGAM(IN)
           13 FORMAT(3X,'GAMMA(',I2,') [',F5.3,']: ',$)
              READ(5,14,ERR=12) GAMLST(IN)
           14 FORMAT(F8.6)
              IF(GAMLST(IN).EQ.0.D0) GAMLST(IN)=DGAM(IN)
              IF(GAMLST(IN).LE.0.D0 .OR. GAMLST(IN).GT.1.D0) GOTO 12
           15 CONTINUE
           16 WRITE(5,17)
           17 FORMAT(' HOW MANY EB/NJ? [126] ',$)
              READ(5,18,ERR=16) NJ
           18 FORMAT(I3)
              IF(NJ.EQ.0) NJ=126
```

```
PDP-11 FORTRAN-77 V4.0-1        10:38:34    17-Jul-86          Page 4
PACJAGC.FTN;7             /F77/TR:BLOCKS/WR

              IF(NJ.LT.0 .OR. NJ.GT.126) GOTO 16
           19 WRITE(5,20)
           20 FORMAT(' STARTING VALUE FOR EB/NJ [0 dB]: ',$)
              READ(5,21,ERR=19) DBJO
           21 FORMAT(F5.2)
              DJ=0.D0
              IF(NJ.GT.1) THEN
           22   WRITE(5,23)
           23   FORMAT(' INCREMENT FOR EB/NJ [0.4 dB]: ',$)
                READ(5,24,ERR=22) DJ
           24   FORMAT(F5.0)
                IF(DJ.EQ.0.D0) DJ=0.4D0
              END IF
              RETURN
              END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:38:42    17-Jul-86          Page 5
PACJAGC.FTN;7             /F77/TR:BLOCKS/WR

              SUBROUTINE PSUBE(IQ,PE)
        C
        C COMPUTE ERROR PROBABILITY
        C
              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
              PARAMETER (SLOTS=2400.D0)
              PARAMETER (SLOTS1=2399.D0)
              PARAMETER (SLOTPR=5757600.D0)
              DIMENSION WORK(20), STACK(20), HEAP(20)
              EXTERNAL DGAU20, PDF
              COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
              COMMON /PIES/ PI0, PI1, PI2
        C
        C COMPUTE ELEMENTAL EVENT PROBABILITIES
        C
              Q=IQ
              PI0=(SLOTS-Q)*(SLOTS1-Q)/SLOTPR
              PI1=Q*(SLOTS-Q)/SLOTPR
              PI2=Q*(Q-1.D0)/SLOTPR
              CALL ADQUA1(0.D0,1.D0,PE,DGAU20,PDF,1.D-4,1.D-15,
             $            WORK,STACK,HEAP,20,KODE)
              IF(KODE.NE.0) STOP 'ADQUA1 ERROR'
              G1=G(1.D0)
              PE=2.D0*PE+G1*G1
              RETURN
              END
```

```
      SUBROUTINE ADQUA1(XL,XU,Y,QR,F,TOL,ABSTOL,
     $                  WORK,STACK,HEAP,N,KODE)
C
C ADAPTIVE QUADRATURE ALGORITHM
C   XL - LOWER LIMIT OF INTEGRAL (IN)
C   XU - UPPER LIMIT OF INTEGRAL (IN)
C    Y - VALUE OF INTEGRAL (OUT)
C   QR - NAME OF A QUADRATURE RULE SUBROUTINE (IN)
C             WITH CALLING SEQUENCE
C             CALL QR(XL,XU,F,Y)
C    F - NAME OF FUNCTION TO BE INTEGRATED (IN)
C  TOL - ERROR TOLERANCE FOR FINAL ANSWER (IN)
C ABSTOL-ABSOLUTE ERROR TOLERANCE (IN)
C WORK - WORK ARRAY OF SIZE N (IN)
C STACK- SECOND WORK ARRAY OF SIZE N, MUST NOT BE
C  HEAP- THIRD WORK ARRAY, SIZE N, DISTINCT FROM WORK AND STACK (IN)
C    N - SIZE OF WORK AND STACK; MAX. NO. OF BISECTIONS (IN)
C KODE - ERROR INDICATOR (OUT)
C          0 -- NO ERROR
C          1 -- WORK ARRAYS TOO SMALL
C R. H. FRENCH, 14 AUGUST 1984
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL F
      DIMENSION WORK(N),STACK(N),HEAP(N)
      KODE=0
      Y=0.D0
      WORK(1)=XU
      CALL QR(XL,XU,F,T)
      HEAP(1)=T
      A=XL
      NPTS=1
      EPS=TOL
      STACK(1)=EPS
10    B=WORK(NPTS)
      XM=(A+B)*0.5D0
      CALL QR(A,XM,F,P1)
      CALL QR(XM,B,F,P2)
      TEST=DMAX1(EPS*DABS(T),ABSTOL)
      IF(DABS(T-P1-P2).LE.TEST) GOTO 20
C SPLIT IT
      NPTS=NPTS+1
      IF(NPTS.GT.N) THEN
        KODE=1
        RETURN
      END IF
      WORK(NPTS)=XM
      HEAP(NPTS)=P2
      T=P1
      EPS=DMAX1(EPS/2.D0,5.D-16)
      STACK(NPTS)=EPS
      GOTO 10
```



```
C FINISHED A PIECE
20    Y=Y+P1+P2
      EPS=STACK(NPTS)
      T=HEAP(NPTS)
      NPTS=NPTS-1
      A=B
      IF(NPTS.EQ.0) RETURN
      GOTO 10
      END
```



```
      DOUBLE PRECISION FUNCTION PDF(Z)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PDF=F(Z)*G(Z)
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION F(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON /PIES/ PI0, PI1, PI2
      COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
      X1=X+1.D0
      RNX=RHON*X
      RTX=RHOT*X
      PART1=DEXP(-RNX/X1)*(1.D0+RHON/X1)/(X1*X1)
      D=BIGK*X+1.D0
      PART2=BIGK*DEXP(-BIGK*RTX/D)*(1.D0+RHOT/D)/(D*D)
      D=X+BIGK
      PART3=BIGK*DEXP(-RNX/D)*(1.D0+BIGK*RHON/D)/(D*D)
      PART4=DEXP(-RTX/X1)*(1.D0+RHOT/X1)/(X1*X1)
      F=PI0*PART1+PI1*(PART2+PART3)+PI2*PART4
      RETURN
      END
```

```
PDP-11 FORTRAN-77 V4.0-1        10:38:51    17-Jul-86           Page 10
PACJAGC.FTN;7           /F77/TR:BLOCKS/WR

            DOUBLE PRECISION FUNCTION G(X)
            IMPLICIT DOUBLE PRECISION(A-H,O-Z)
            COMMON /PIES/ PIO, PI1, PI2
            COMMON /ROSE/ RHON, RHOT, GAMMA, BIGK
            X1=X+1.D0
            PART1=DEXP(-RHON/X1)/X1
            PART2=DEXP(-RHOT/X1)/X1
            XK1=1.D0+BIGK*X
            PART3=BIGK*DEXP(-RHON/XK1)/XK1
            XK=X+BIGK
            PART4=DEXP(-BIGK*RHOT/XK)/XK
            G=X*(PIO*PART1+PI1*(PART4+PART3)+PI2*PART2)
            RETURN
            END
```

REFERENCES

1. J.S. Lee, L.E. Miller, R.H. French, Y.K. Kim, and A.P. Kadrichu, "The optimum jamming effects on frequency-hopping M-ary FSK systems under certain ECCM receiver design strategies, Final report, Office of Naval Research Contract N00014-83-C-0312, October 1984. (AD-A147766)

2. Milton Abramowitz and Irene A. Stegun (eds.), Handbook of Mathematical Functions, National Bureau of Standards Applied Mathematics Series 55. Washington, D.C.: Government Printing Office, June 1964, Ninth printing, November 1970.

3. I.S. Gradshteyn and I.M. Ryzhik, Table of Integrals, Series, and Products, Corrected and Enlarged Edition. New York: Academic Press, 1980.

4. Daniel I.A. Cohen, Basic Techniques of Combinatorial Theory. New York: Wiley and Sons, 1978.

5. A.J. Viterbi and I.M. Jacobs, "Advances in Coding and Modulation for Non-Coherent Channels Affected by Fading, Partial Band, and Multiple Access Interference", in Advances in Comm. Syst., A.J. Viterbi (ed.), New York: Academic Press, 1975, pp. 279-308.

6. J.E. Blanchard, "A Slow Frequency Hopping Technique That is Robust to Repeat Jamming," IEEE Military Electronics Conf. Record, October 1982, paper 14.1.

7. D.J. Torrieri, Principles of Military Communications. Dedham, MA: Artech, 1981.

8. H. Hite, et al., "High Frequency Counter-Countermeasures Study", August 5, 1983, Hughes Aircraft Co., Fullerton, CA, under Contract DAAK80-81-C-0119, AD-C032993.

9. D.J. Torrieri, "Information-Bit Error Rate for Block Codes", IEEE Trans. on Comm., Vol. COM-32, April 1984, pp. 474-476.

10. J.S. Lee, et al., "Probability of Error Analyses of a BFSK Frequency Hopping System With Diversity Under Partial-Band Jamming Interference - Part I: Performance of Square-Law Linear Combining Soft Decision Receiver", IEEE Trans. on Commun., Vol. COM-32, June 1984, pp. 645-653.

11. J.S. Lee, et al., "Probability of Error Analyses of a BFSK Frequency Hopping System With Diversity Under Partial-Band Jamming Interference - Part II: Performance of Square-Law Nonlinear Combining Soft Decision Receivers", IEEE Trans. on Commun., Vol. COM-32, December 1984, pp. 1243-1250.

12. K.S. Gong, "Performance of Diversity Combining Techniques for FH/MFSK in Worst Case Partial Band Noise and Multi-Tone Jamming", Proc. 1983 IEEE Military Commun. Conf., pp. 17-21.

13. Y.K. Kim, et al., "The Exact Performance Analyses of Two Types of Adaptive Receivers for Multi-Hops Per Symbol FH/MFSK Systems in Partial-Band Noise Jamming and System Thermal Noise", Proc. IEEE Global Telecom. Conf. 1983, pp. 1309-1314.

14. J.S. Lee, et al., "Signal Design and Detection Strategies for LPI Communications in Electronic Warfare Environments", J.S. Lee Associates, Inc., May 1983, Final report on Contract N00014-81-C-0534, AD-B073026L.

15. J.S. Lee, et al., "Error Performance Analyses of Linear and Nonlinear Combining Square-Law Receivers for L-Hops Per Bit FH/BFSK Waveforms in Worst-Case Partial-Band Jamming", Proc. 1983 IEEE Military Comm. Conf., pp. 22-28.

16. J.S. Bird and E.B. Felstead, "Antijam Performance of Fast Frequency-Hopped M-ary NCFSK an Overview," IEEE Journal on Selected Areas in Communications, Vol. SAC-4, March 1986, pp. 216-233.

17. D.J. Torrieri, "Frequency Hopping with Multiple Frequency-Shift Keying and Hard Decisions," IEEE Trans. on Commun., Vol. COM-32, May 1984, pp. 574-582.

18. E.J. Kelly, et al., "The Sensitivity of Radiometric Measurements", J. Soc. Indust. Appl. Math., Vol. 11, pp. 235-257, June 1963.

19. J.S. Bendat and A.G. Piersol, Random Data: Analysis and Measurement Procedures, New York: Wiley-Interscience, 1971.

20. A. Gelb, ed., Applied Optimal Estimation. Cambridge: MIT Press, 1974.

21. L.E. Miller, J.S. Lee and A.P. Kadrichu, "Probability of Error Analyses of a BFSK Frequency-Hopping System with Diversity under Partial-Band Jamming Interference--Part III: Performance of a Square-Law Self-Normalizing Soft Decision Receiver," IEEE Trans. on Commun., Vol. COM-34, No. 7, July 1986, pp. 669-675.