FINAL REPORT

A STUDY OF HYBRID SEQUENTIAL AND ALGEBRAIC DECODING TECHNIQUES

Contract NAS2 - 4877

Submitted to:

NASA AMES RESEARCH CENTER

MOFFETT FIELD, CALIFORNIA

6 January 1969

Submitted by:

CODEX

CODEX CORPORATION • 222 ARSENAL STREET, WATERTOWN, MASSACHUSETTS 02172

Final Report

Contract NAS2-4877

A Study of Hybrid Sequential and Algebraic Decoding Techniques

Submitted to:

NASA Ames Research Center

Moffett Field, California

6 January 1969

Submitted by:

Codex Corporation

222 Arsenal Street

Watertown, Mass. 02172

TABLE OF CONTENTS

Introduction

This is the Final Report on Contract NAS2-4877, "A Study of Hybrid Sequential and Algebraic Decoding Techniques." In the main body of the report we summarize the objectives of the study, the work performed, and the principal conclusions which can be drawn from experiment (simulation) and from analysis; finally, we evaluate the potentialities of several hybrid techniques. Most of the detailed results which were submitted in bimonthly progress reports have been collected in five appendices.

Background and Objectives

In the last few years, the potential efficiencies in deep space telemetry data transmission which can be obtained by the introduction of coding have begun to be widely recognized. In the first operational system using coding, for the Pioneer program (now flying), Ames chose to use sequential decoding because of its unmatched efficiency and relative ease of implementation. We feel that sequential decoding should and will be the standard scheme for deep space telemetry involving moderate data rates over the next few years.

In many respects, sequential decoding is ideally suited to the deep space telemetry application. The on-board equipment required is merely a convolutional encoder, which is easy to build with standard digital components. The complexity is concentrated on the ground, but even here, for data rates of the order of 1000 bps or less, general purpose computers may be commandeered, as in Pioneer, to handle the decoding. By the use of a moderately long code, the probability of undetected errors may be made negligible, thus permitting efficient data compression schemes without compromising the ability to recognize the interesting rare event. The probability of detectable loss of a data frame or two is governed by the amount of computing time available; it is possible to set up the system so that 99% of the data is available for real-time look-ins without excessive real-time computing or buffering, and then to retrieve the remainder of the data through off-line processing, possibly con centrating on particularly interesting frames.

The efficiency of sequential decoding is outstanding, though not the maximum permitted by theory. On a channel such as the space channel, on which the noise is adequately characterized as white and gaussian, the theoretical Shannon limit (capacity) occurs at a signal-to-noise ratio per information bit E_b/N_o of ln 2, or -1.6 db; arbitrarily reliable communication down to this limit is possible in principle with sufficient complexity. With sequential decoding, the nominal E_b/N_o is taken as that value below which the average decoding computation is infinite, and this appears to be an accurate estimate of the practical limit as well. This nominal E_b/N_o depends on the rate of the code and the amount of output quantization; for very low rates and very fine quantization it is 1.4 db, exactly 3 db above the Shannon limit. Quantization of the receive modem decision variable to three bits involves about a .15 db loss, and restriction to a practical rate like 1/6, another .2 db, so that one might say that practical schemes can approach to within a few tenths of a db of the nominal 1.4 db limit. The lowness of the code rate is principally limited by the signal-to-noise ratio per transmitted bit E/N_o permissible in the receive modem/bit synchronizer; the number of useful quantization levels depends on the accuracy of the decision variable automatic gain control. To these losses must in practice be added the redundant bits required for resynchronization; however, this loss can be made small, and superfluous bits are normally required anyway for frame synchronization.

Although signalling within 3-4 db of the Shannon limit with adequate reliability was not dreamed of until recently, the exigencies of deep space missions require examination of whether still more efficient communication can be obtained. It is hardly necessary to point out again that when information is the sole product of an expensive mission and when power is severely constrained, each last decibel has immense value. The route to still more efficient communication has been generally seen to be to build on the existing efficiency of sequential decoding with some kind of auxiliary apparatus for curbing the variability of decoding computation which is sequential decoding's fundamental limitation. In Appendix E of our Phase II Report for Contract NAS2-2874, we discussed how the addition of more redundancy in the form of an erasure correcting algebraic code might work out in terms of achievable performance, code parameters, and system complexity. We singled out the configuration of a (64,58)

Reed-Solomon code and a rate 1/8 sequential decoding scheme for special attention, and suggested that implementation appeared not too outrageous; a later study of an on-board data processor (Sylvania) confirmed that the encoding function, at least, would not be excessively burdensome as part of a data processor structure. Concurrently, Falconer undertook a serious study of the whole class of hybrid sequential and algebraic schemes in his doctoral thesis at MIT (1967), obtaining both theoretical and simulation results consistent with the earlier rough predictions. As a result of Falconer's work, the feeling became general among workers in the field that his was the most likely route to the most efficient communication on memoryless channels. Finally, as the concluding part of our recent study work for Ames (Final Report, 1967), we attempted without notable success to find more simply implemented schemes of the same class and power, and with substantial success to develop methods of predicting performance for schemes in this class, based upon a detailed understanding of the mechanisms dominating the behavior of sequential decoders. One small success was to confirm through simulations the efficiency of a backwards-forwards decoding scheme which had been suggested by Lumb at Ames, a major virtue of which was the absence of any requirement for additional redundant bits.

Our objectives in this study were to find a scheme in the Falconer class which would be not unreasonable to implement and would perform significantly better than ordinary sequential decoding. Quantitatively, what does this mean? First of all, for a scheme of any complexity at all, the performance improvement ought to be at least 1 db below $R_{comp.}$ We make this statement, first, because we do not believe project people will even consider a new system if the potential gain is less than a decibel, and in fact will really have to be desperate before 1 dB looks attractive; second, because with large computational resources even ordinary sequential decoding can be pushed several tenths of a dB below $R_{comp.}$ Second, since all these schemes are based on the assumption of coherence, we can assume a bit rate of at least 10 bps or so; we may also assume that even with special purpose hardware no more than one sequential decoding 'computation' can be done per microsecond; we conclude that the average number of computations per bit must be less than 10^5 or so. This seems liberal enough, yet we shall see that we approach even this figure very rapidly as we move from R_{comp} toward capacity.

Code Structure

We have assumed throughout a concatenated coding structure like Falconer's, with an (n,k) algebraic outer code and a rate 1/m convolutional inner code. Information bits are separated into k distinct streams; an algebraic (n,k) encoder generates n-k additional check streams by encoding across the separate streams. Each separate stream is then encoded by a binary convolutional encoder of rate 1/m, and sent through a white gaussian channel at an energy-to-noise ratio per information bit of $E_b/N_o = mE/N_o$. The overall energy-to-noise ratio per original information bit $(E_b/N_o)_o$ is then E_b/N_o (in dB) plus an outer coding rate loss given by the outer code rate in dB; a rate 4/5 code gives about a 1 dB loss, a rate 9/10 .5 db, and so forth.

The convolutional encoders may periodically insert fixed sequences in the transmitted streams for purposes of resynchronization, in which case an additional resynchronization loss is involved; we also experiment below with automatic resynchronization strategies.

At the receiver each of the n streams is initially decoded independently by ordinary sequential decoding. The central idea is that if most are progressing well, but a few are in trouble, these few can be decoded by an algebraic decoder from the outer code constraints. In Falconer's original scheme, the algebraic decoder was activated whenever all but d-1 of the streams had been sequentially decoded far enough beyond a particular symbol location that they were assuredly correct, so that the remaining d-1 symbols could be computed by an erasure-correction algorithm, d-1 being the erasure-correction capability of the algebraic decoder. In our work here, we also contemplate allowing the algebraic decoder to intervene earlier, when some of the streams may still be incorrect. We also consider briefly the possibility of using quite short frames, or of using a Viterbi-type decoding algorithm rather than sequential decoding.

Computer Programs

The principal computer program was one simulating sequential decoding of a rate 1/m code at energy-to-noise ratios between R_{comp} and capacity. The program was capable of operation in three modes:

1. (Normal) Starting from a known initial state, the decoder decodes

a continuous run of N branches (=Nm bits), stopping when it first reaches the last bit. The decoder is inhibited from ever searching back past the initial bit or subsequently more than B branches back from its furthest advance, where B is called the backsearch limit or search depth.

2. (Genie) Same as normal, except that any decoding error passing beyond the backsearch limit is immediately corrected, and its influence removed from subsequent decoding.

3. (Resynch) The decoder is started from a random initial state, and decodes until at some point it has hypothesized an entire constraint length of correct data.

We also used a sequential decoding simulator for a binary symmetric channel largely written under a previous contract (NAS2-3637) to cross check some of the results.

Finally, we generated a program to compute the exact distribution of the d th largest of n independent identically distributed Pareto random variables.

Computer Results

The computer results obtained may be divided into those concerning computational distributions, probabilities of decoding error, and resynchronization.

The computational distributions obtained depended greatly on the backsearch limit B, and of course the signal-to-noise ratio, but very little on anything else. We began by establishing distributions of computation to advance one branch for B large (equal to 500), for codes of rate 1/2 and 1/4, and for signal-to-noise ratios equivalent to R_{comp}, 1dB below R_{comp}, and 2dB below R_{comp} (Appendix C). The results were approximately of the expected Pareto form, with approximately the theoretically expected Pareto exponent at R_{comp} and 1 dB below R_{comp}. At 2dB below R_{comp}, the observed Pareto exponent was significantly superior to the theoretical prediction, indicating perhaps that in the interesting range of computation the conventional sequential decoding analyses are unreliable near capacity. We determined that neither the code constraint length nor its composition (systematic or nonsystematic) affected the computational distribution. As we reduced the backsearch limit B, we found that a very sharp dropoff ('knee') in the computational distributions

was induced at some number of computations C(B). The location of this knee was seen to be completely independent of constraint length, code composition (systematic or nonsystematic), or whether the decoder was in normal, genie, or resynch mode, and largely independent of signal-to-noise ratio (Appendix D). Figures 1 and 2 show characteristic curves obtained for a rate 1/4 code of constraint length 20 at 2 dB below R_{comp} with B=20,50, and 100, for normal and for genie mode. Very similar results were obtained on the binary symmetric channel (Appendix D), even at an error probability of 50%.

An attempt to obtain a knee by decreasing the constraint length to 6 and increasing the bias with B=500 was unsuccessful (Appendix C).

Two types of error event occur with sequential decoding. The first is the ordinary sort of undetected error which occurs when the sequential decoder accepts a string of wrong hypotheses and then continues normally. The second occurs where an error escapes beyond the backsearch boundary. In normal decoding, this seems to have the effect of unsynchronizing the decoder, which then bumbles on as though receiving random data until becoming resynchronized, making many errors. We consider such behavior under resynchronization below. With a genie, no such propagation occurs; we simply tabulate an irreducible error rate, which seems to be independent of constraint length when the undetected error probability is negligible. With rate-1/4 codes at 2 dB below R_{comp}, this irreducible error probability is obtained with a constraint length of 20 and is about 2% at B=100, 5% at B=50, and 10% at B=20. At a constraint length of 10 a few percent of additional undetected errors are also observed. These bit errors are not bunched, but rather are in general fairly scattered (half a dozen or a dozen bits apart). With a genie, the irreducible error rate at B=100 was reduced to 1% at 1.5 dB below R_{comp} and 6.7×10^{-4} at .5 dB below R_{comp} (Appendix E).

Success at resynchronization was highly dependent on constraint length and backsearch limit. Histograms of number of branches to resynchronize for a constraint length 20 code at 2dB below R_{comp} are shown in Figure 3 for B=20, 50, and 100. The curves are acceptably fitted by a distribution in which the probability of resynchronization after N branches is given by $(1-p)^N$, where 1/p, the average time to resynch, is about 2200, 1200, and 200 for the three cases. For a constraint length of 10 the average resynch time was 40 branches for both B=50 and 100, while for 30 only twice in 9 trials was resynchronization achieved in 1000 branches with B=100. We conclude

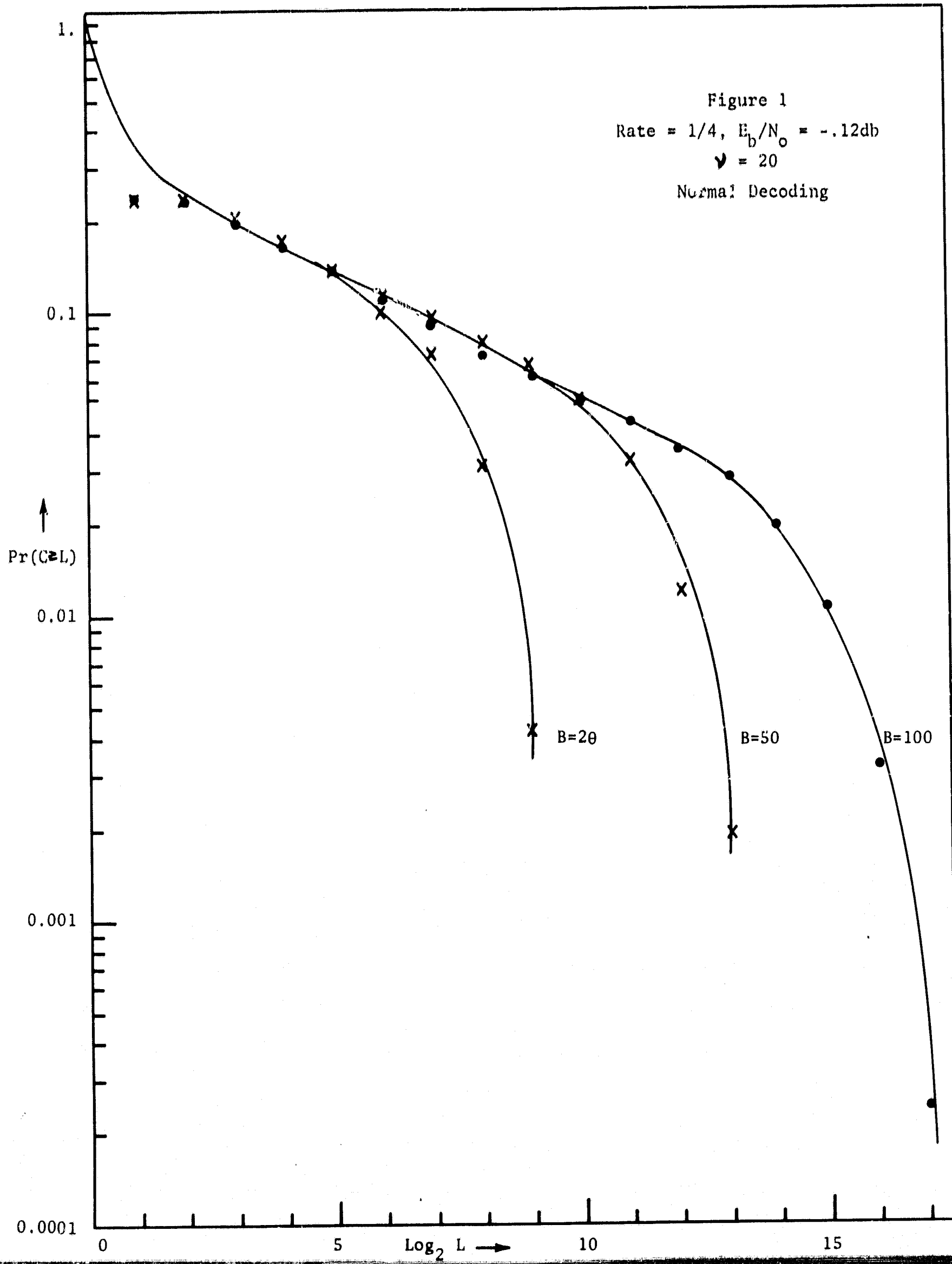


Figure 1

Rate = 1/4, E_b/N_o = -.12db

ν = 20

Normal Decoding

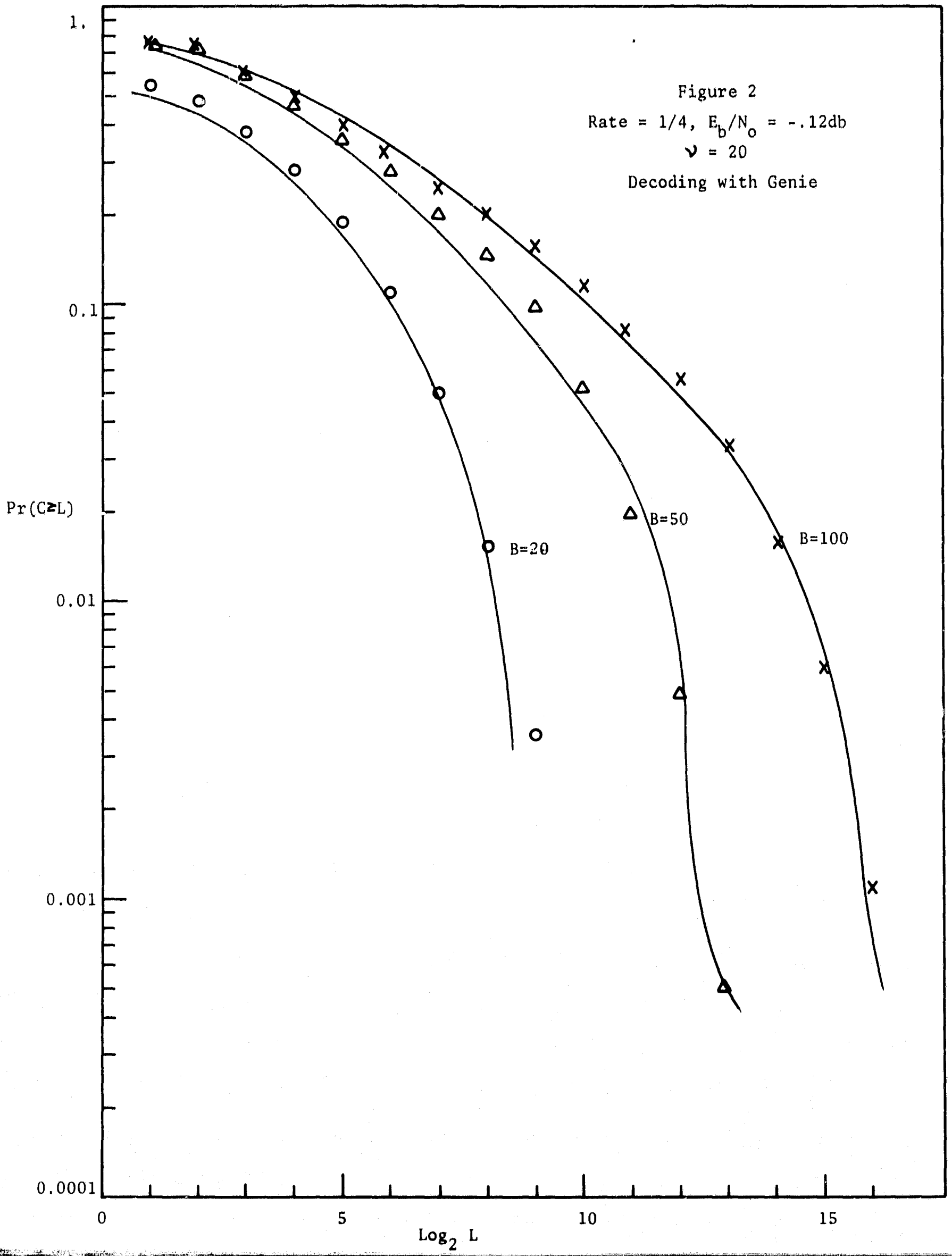


Figure 2

Rate = 1/4, E_b/N_o = -.12db

ν = 20

Decoding with Genie

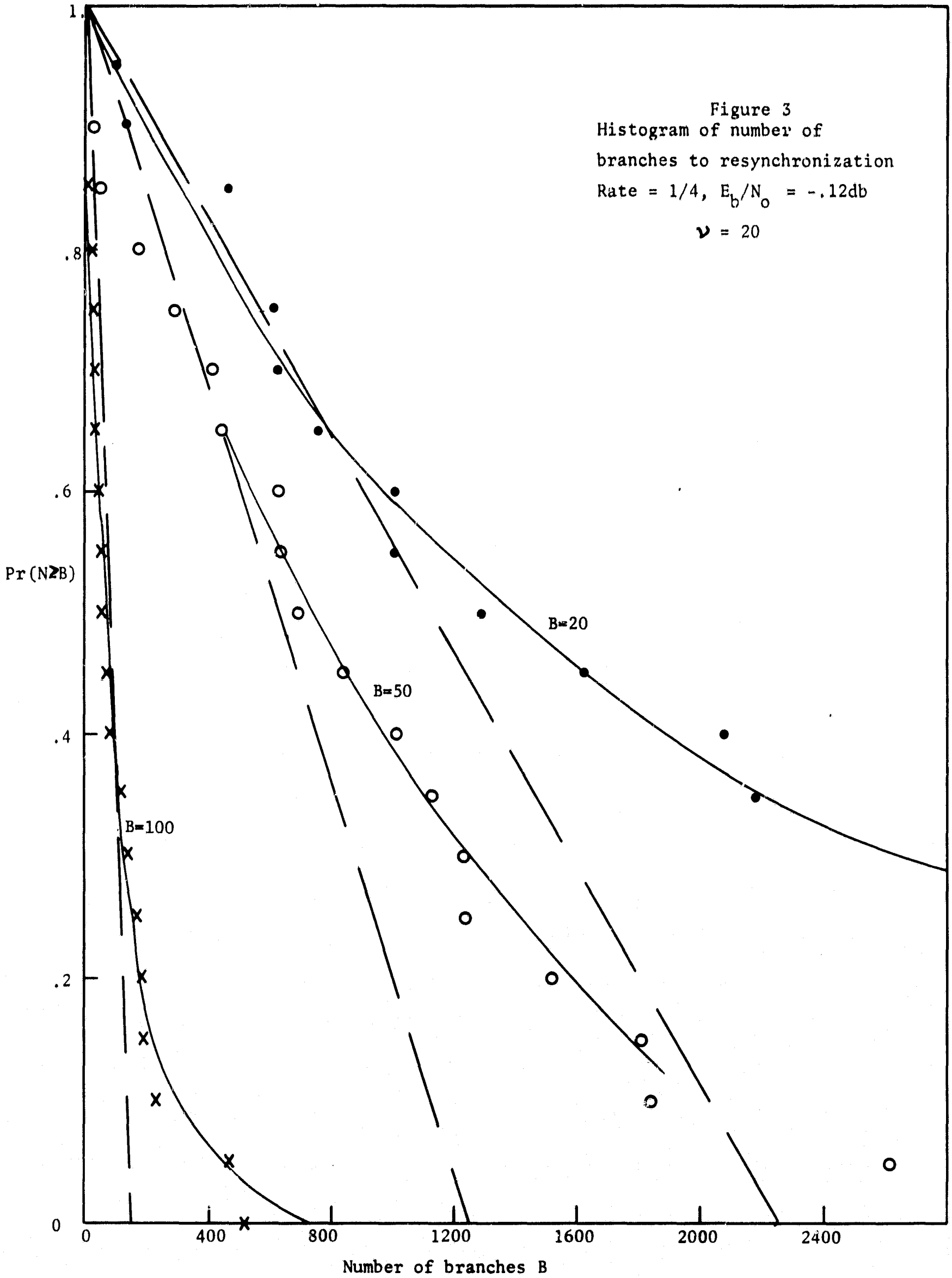


Figure 3
Histogram of number of
branches to resynchronization
Rate = 1/4, E_b/N_o = -.12db
ν = 20

is feasible for small constraint lengths. It seems likely that just starting a sequential decoder in a random state with an immovable backsearch limit at the starting point would resynchronize with as few computations as any method, with the loss of the fewest number of bits.

Finally, exact calculation showed that the distribution of the dth largest of n independent identically distributed Pareto variables, each with distribution

$$\Pr (C \geq L) = KL^{-\alpha},$$

which is bounded by the union bound by

$$\Pr (C_d \geq L) \leq \binom{n}{d} K^d L^{-\alpha d},$$

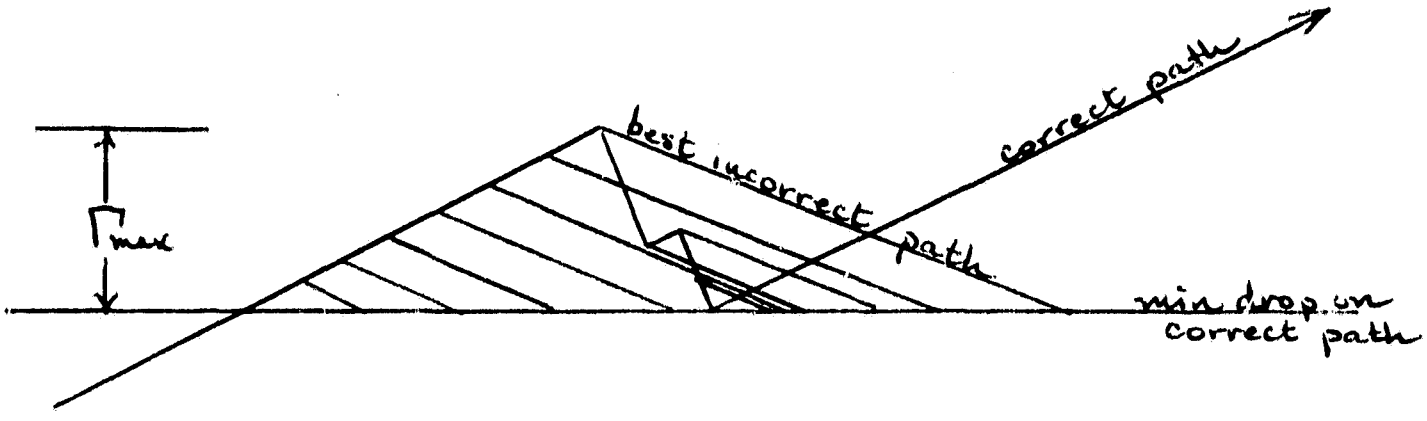
is actually very nearly equal to the above over the complete range for d=2, α=1/2 and d=5, α=1/5 (Appendix A).

Outline of Analysis of Effect of Limiting Backsearch

The unexpected phenomenon which most engaged our attention throughout this study was the sharp dropoff in computational distribution due to imposition of a backsearch limit on the sequential decoder. We now give our present understanding of this phenomenon, and its connection to the event of errors escaping beyond the backsearch limit.

A sequential decoder is customarily thought of as searching through the tree of all possible transmitted sequences. The sequence actually transmitted is called the correct path. From each branch on the correct path there extends a subtree of all sequences correct to that point, differing in the next branch, and anything thereafter, known as the incorrect subset associated with that branch. While the decoder is searching in an incorrect subset, theory assumes, and experiment tends to confirm, that the received data are totally uncorrelated with the decoder's hypotheses, i.e., that the decoder might as well be trying to decode assuming a code totally different from the encoder's code. Under these circumstances, it is overwhelmingly probable that the best path in the incorrect subset of length B branches will have from the point of its divergence a metric of $-\lambda B$, where λ is a fixed constant depending on the metric statistics (which on the binary symmetric channel can be derived from the Gilbert bound). Therefore a diagram of the metric

history of the correct path and the best of each of the incorrect paths will look about like this:



Thus if there is a dip in the correct path of size Γ_{max}, then to get by the dip the threshold must be dropped to the point where the best path from the high point extends $B = \Gamma_{max}/\lambda$ branches before crossing the threshold. If the backsearch limit is less than B, an error will then pass beyond it and the decoder will be stuck in the incorrect subset until resynchronization. If the backsearch limit is greater than B, no error will occur. Finally, the amount of computation C(B) in an incorrect subset is, because of the statistical regularity in the incorrect subset, a not-very-random function of the depth B, or equally of the height of the origin above the minimum threshold, increasing asymptotically exponentially with the latter. The computation due to a drop of Γ_{max} is therefore dominated by the computation in the incorrect subset extending from the high point, so is largely determined also by Γ_{max}. We have thus agreed that the following

three events may be considered nearly identical:

1. The correct path drops by Γ_{max};
2. An error persists at a depth of $B = \Gamma_{max}/\lambda$;
3. The decoder has a computational search of length C(B).

This analysis leads to the following recipe for determing the tradeoff between computational limitation and error due to limited backsearch. Take the distribution of computation to advance one branch which applies when the backsearch limit is large. Let p be some acceptable probability of bit error due to limited backsearch, and let L_o be such that $Pr\ (C \geq L_o) = \beta$. Then it is possible to choose B such that the tail of computation comes at L_o and the probability of error is p.

Application of this method to Figure 1 of this report or Figure 2 of the Third Bimonthly leads to reasonably accurate predictions of the actually observed error probabilities for a given B and C(B).

One implication of this analysis is that application of either an overflow limit on computation or a drop limit on the metric path would be very similar in effect to imposition of the backsearch limit; in any of the three cases computation would be sharply limited, but whenever the limit was hit the decoder would become unsynchronized, until the bad section was algebraically decoded or the decoder automatically resynchronized.

Evaluation of Different Algorithms

First, let us return to Falconer's original scheme, where the algebraic decoder was interposed after N branches, N being large enough that the probability of an error at that depth is negligible. Without evaluating what N must be, we see that our results above show that at 2 dB below R_{comp} N must be much larger than 100, perhaps more than 1000. If the distribution of the number of computations C to advance a single branch is

$$Pr\ (C \geq L) = KL^{-\alpha},$$

then the distribution of the number of computations C_N to advance N branches is approximately

$$Pr\ (C_N \geq L) = KNL^{-\alpha}.$$

The distribution of $C_{n,d}$, the dth largest of n independent variables C_N is, as we have seen, accurately approximated by

$$\Pr\ (C_{n,d} \geq L) \simeq \binom{n}{d}\ (KN)^d\ L^{-\alpha d}.$$

Finally, the total computations for all but d-1 of n independent decoders to advance N branches is greater than $C_{n,d}$ and less than $n\,C_{n,d}$, so

$$\Pr\ (C_t \geq L) \geq \Pr\ (C_{n,d} \geq L)$$
$$\Pr\ (C_t \geq L) \leq \Pr\ (C_{n,d} \geq L/n),$$

so

$$\binom{n}{d}\ (KN)^d\ L^{-\alpha d} \leq \Pr\ (C_t \geq L) \leq \binom{n}{d}\ (KN)^d\ n^{\alpha d}\ L^{-\alpha d}.$$

The computation per bit is at least C_t/n N.

From Figure 1 we draw the parameters $K \simeq .3$, $\alpha \simeq .27$ at 2dB below R_{comp} for that rate-1/4 code. Letting d=4 (to get $\alpha d \simeq 1$), and letting N be of the order of 500, probably an optimistic estimate, we compute that

$$\binom{n}{d}\ (KN)^d \simeq n^4 \times 2 \times 10^7$$

It is clear that this coefficient is simply much too huge; even with n=20, $\Pr(C_{n,d} \geq L) \simeq 1$ at $L=3\times10^{12}$ computations.

In his thesis, Falconer reports simulations for N=10 and d=2 which show that with a rate-1/7 code at E_b/N_o=1.38 db (or an overall $(E_b/N_o)_o$ of about 1.8 dB, with a rate 9/10 code) the probability that the number of computations C for an individual decoder to advance one bit exceeds $L=10^5$ is about 10^{-4}, and by extrapolation we can estimate $\Pr\ (C \geq 10^6) \simeq 10^{-5}$, $\Pr\ (C \geq 10^7) \simeq 10^{-6}$. On the other hand, the average number of computations per bit was only about 50. These parameters are certainly practical, if a moderate amount of buffer storage can be provided. However, an $(E_b/N_o)_o$ of 1.8 dB can be more easily obtained with ordinary sequential decoding. What these results suggest is that when Falconer's scheme is pushed into the region of even moderate gains like 1 dB, the numbers quickly become insupportable.

The most efficient scheme of this class apparent to us now is the following. Let all symbols in all n streams have been decoded correctly and irrevocably up to some point. Let the algebraic decoder be capable of decoding up to d-1 erasures. Let us further make the somewhat optimistic assumption that we can tell as soon as any stream is correctly decoded by the

sequential decoder for one further symbol of b bits. (If our previous analysis was correct, this is probably not too unrealistic, since as long as the decoder is searching in the incorrect subset a constant functional relationship between the number of computations, the metric drop, and the search depth on the best path should hold; correct decoding could thus be signalled by a breakout from these relationships.) We wait until all but d-1 symbols at that point have been correctly decoded by the sequential decoders, and then pick up the rest with the algebraic erasure-corrector. It is hard to see how we could do much better than this; as long as d decoding errors remain the algebraic decoder is helpless. Now we interject our conclusions that the event that an error persists after B branches is highly correlated with the event of a search of C(B) or more computations, the probability of which we can read from our graphs of Pr $(C \geq L)$. The probability of having to undergo a search of size $C_b \geq L$ before decoding b branches correctly is then

$$\Pr\ (C_b \geq L) \simeq b \Pr\ (C \geq L) \simeq bKL^{-\alpha}.$$

The probability that $C_{n,d}$, the d^{th} largest computation of n, is greater than or equal to L is then approximately

$$\Pr\ (C_{n,d} \geq L) \simeq \binom{n}{d} (bK)^d L^{-\alpha d}.$$

The following table gives the value of p for which $\binom{n}{d} p^d = 10^{-6}$ and 10^{-9} for d=2, 5, and 11, and n=5(d-1) and 10(d-1). (With a Reed-Solomon code the former would give a rate loss of 1dB, the latter of 1/2 dB.)

d	n	p (10^{-6})	p (10^{-9})
2	5	3.2×10^{-4}	10^{-5}
2	10	1.5×10^{-4}	4.7×10^{-6}
5	20	9.2×10^{-3}	2.3×10^{-3}
5	40	4.3×10^{-3}	1.1×10^{-3}
11	50	3.1×10^{-2}	1.7×10^{-2}
11	100	1.5×10^{-2}	7.9×10^{-3}

This shows that b Pr $(C \geq L) \simeq bKL^{-\alpha}$ must be of the order of one or two percent for an algebraic erasure-corrector to work effectively. Unfortunately, with a Reed-Solomon code $b=\lceil \log_2 n \rceil$, so Pr $(C \geq L)$ itself must be down to a few parts in 10^{-3} for $n \simeq 100$. We see by extrapolation from Figure 1, for example,

that at 2 dB below R_{comp} this means that L must be of the order of 2^{25} or $3x10^7$. The average of a random variable which is Pareto [$Pr\ (C \geq L) = KL^{-\alpha}$] out to some point L_o and then is limited to L_o is approximately $KL_o^{1-\alpha}$ for $\alpha \leq 1$, so the average computation per bit if we truncated the distribution at $L_o = 3x10^7$, by imposing either a backsearch limit or a limit directly on computation, would be $.3x(3x10^7)^{.73} \simeq 10^5$, at the outside limit of what we could tolerate with special purpose decoding equipment. Dropping back to 1 dB below R_{comp} and extrapolating from Figure 2 of the Appendix C, we find that we get to $Pr\ (C \geq L_o) = 3x10^{-3}$ at $L_o \simeq 2^{17.5} = 2x10^5$; since $K \simeq .5$, $\alpha \simeq .56$ here, the average computation per bit would be more like 100, which begins to seem reasonable.

In conclusion, although the one curve we have at 1.5 dB below R_{comp} is difficult to extrapolate from, we can estimate that our minimum target of a 1dB gain at an average of less than 10^5 computations per bit might be just about met by operating a sequential decoder at about 1.5 dB below R_{comp}, limiting computations per bit to somewhere in the range 10^6-10^7, and using something like a (100, 90) or (200, 180) Reed-Solomon erasure-correcting outer code to pick up undecoded 7- or 8-bit symbols. With a probability of something like 10^{-6} to 10^{-9} per symbol, erasure-correction would fail, at which point the d or more undecoded streams would have to be automatically resynchronized, with a loss of something like 100 to 1000 bits in the process. It seems very questionable whether this is worth the effort.

Finally, let us mention two other possibilities for similar schemes which we did not investigate. The first would involve use of a rather short frame with a resynchronization tail of length equal to the constraint length. We have seen that a short frame without a tail has a computational dropoff due to the limitation of backsearches to the start of the frame, and presumably the tail would improve things still more. However, even for a constraint length of 20 the frame length would have to be of the order of 200 to keep the additional rate loss down to .5 dB; when added to the outer code rate loss, this would probably make this scheme too inefficient.

A second possibility, discussed in Appendix B, would be to use the Viterbi (1967) decoding algorithm rather than sequential decoding. The theoretical advantage, as with the concatenation schemes, is that the error probability can be made to decrease exponentally with total

code length N while decoding complexity per bit increases only linearly with N. The difficulties are, first, that, unlike with sequential decoding, part of the complexity is the requirement for storage of 2^{ν} past histories, which makes the algorithm impractical for constraint lengths greater than 10 or so. Even if this disadvantage could be circumvented by some modification of the Viterbi algorithm, the practicality and performance of the scheme would depend critically on the actually achievable error probabilities at rates between R_{comp} and C. Work on the Viterbi algorithm is now in progress at MIT, JPL, and UCLA, and perhaps elsewhere.

Recommendations for Future Work

None; at this point one more decibel just doesn't seem worth it.

References

Codex Corp., Final Report on a Coding System Design for Advanced Solar Missions, Contract NAS2-3637, Watertown, Mass., Dec. 18, 1967.

Falconer, D.D., "A Hybrid Sequential and Algebraic Decoding Scheme," Ph. D. Thesis, M.I.T. Dept. of Electrical Engineering, February 1967.

Viterbi, A. J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Info. Thy., IT-13, 260-269 (1967).

Appendix A. The Distribution of the d-th Largest of N Independent Pareto Random Variables

Consider N random variables, chosen independently from the Pareto distribution,

$$F(z) = 1 - \left(\frac{z}{z_0}\right)^{-\alpha},$$

$$f(z) = \frac{\alpha}{z_0}\left(\frac{z}{z_0}\right)^{-\alpha-1},$$

$$z \geq z_0, \quad \alpha > 0.$$

The density funciton, gd(y), of the d-th largest of these random variables is given by [1]

$$g_d(y) = \frac{N!}{(N-d)!\,(d-1)!}\left[1 - F(y)\right]^{d-1}\left[F(y)\right]^{N-d} f(y)$$

$$= \binom{N}{d} d \left(\frac{y}{z_0}\right)^{-\alpha(d-1)}\left[1-\left(\frac{y}{z_0}\right)^{-\alpha}\right]^{N-d}\left(\frac{\alpha}{z_0}\right)\left(\frac{y}{z_0}\right)^{-\alpha-1}$$

$$= \binom{N}{d}\frac{d\alpha}{z_0}\left(\frac{y}{z_0}\right)^{-\alpha d-1}\left[1-\left(\frac{y}{z_0}\right)^{-\alpha}\right]^{N-d}.$$

The probability, Pd(y), that the d-th largest random variable exceeds $y \geq Z_0$ is given by

$$P_d(y) = 1 - G_d(y) = \int_y^{\infty} g_d(x)\,dx$$

Substituting the above expression for gd(x) and expanding its last factor by the binomial expansion yields

$$P_d(y) = \int_y^\infty \binom{N}{d} \frac{d\alpha}{z_o} \left(\frac{x}{z_o}\right)^{-\alpha d - 1} \sum_{i=0}^{N-d} \binom{N-d}{i} \left[-\left(\frac{x}{z_o}\right)^{-\alpha}\right]^{N-d-i} dx.$$

$$= \binom{N}{d} \frac{d\alpha}{z_o} \sum_{i=0}^{N-d} (-1)^{N-d-i} \binom{N-d}{i} \int_y^\infty \left(\frac{x}{z_o}\right)^{\alpha i - \alpha N - 1} dx$$

$$= \binom{N}{d} d\alpha \sum_{i=0}^{N-d} (-1)^{N-d-i} \binom{N-d}{i} \left[\frac{-1}{\alpha i - \alpha N} \left(\frac{y}{z_o}\right)^{\alpha i - \alpha N}\right]$$

$$= \binom{N}{d} d \sum_{i=0}^{N-d} (-1)^{N-d-i} \binom{N-d}{i} \left(\frac{1}{N-i}\right) \left(\frac{y}{z_o}\right)^{-\alpha(N-i)}.$$

This sum may be rearranged to place its asymptotically largest term first by making the substitution j = N-d- i.

$$P_d(y) = \binom{N}{d} \left(\frac{y}{z_o}\right)^{-\alpha d} \sum_{j=0}^{N-d} (-1)^j \binom{N-d}{j} \left(\frac{d}{d+j}\right) \left(\frac{y}{z_o}\right)^{-j\alpha}.$$

Since we have been unable to reduce this expression to a closed form, a Fortran program has been written to evaluate Pd(y) for a given set of N, d, x, Z_o, and y.

The asymptotic behavior of the distribution can be determined from the above expression. As y becomes large, the first (j=0) term of the summation becomes dominant. This term has value 1, so the asymptotic distribution is

$$P_d(y) \doteq \binom{N}{d} \left(\frac{y}{z_o}\right)^{-\alpha d} = \binom{N}{d} z_o^{\alpha d}\, y^{-\alpha d}.$$

This confirms Falconer's [2] result that the distribution is asymptotically Paretian, with exponent αd. The coefficient is seen to

be $\binom{N}{d}$, the value which Falconer obtained as an upper bound. The proposal recommended (20, 2) and (100, 5) as values of (N, d); the corresponding values of $\binom{N}{d}$ are 190 and approximately 7.5×10^7, respectively.

Figure 1 shows the distributions and their asymptotes. In both cases, α was chosen such that $d\alpha = 1$. These curves show that the distribution resembles a Pareto distribution with exponent $d\alpha$ over its entire range.

1. Gumbel, E.J., Statistics of Extremes, Columbia University Press, New York, 1966.

2. Falconer, D.D., "A Hybrid Sequential and Algebraic Decoding Scheme", unpublished Ph.D. dissertation, Department of Electrical Engineering, M.I.T., Feb. 1967.

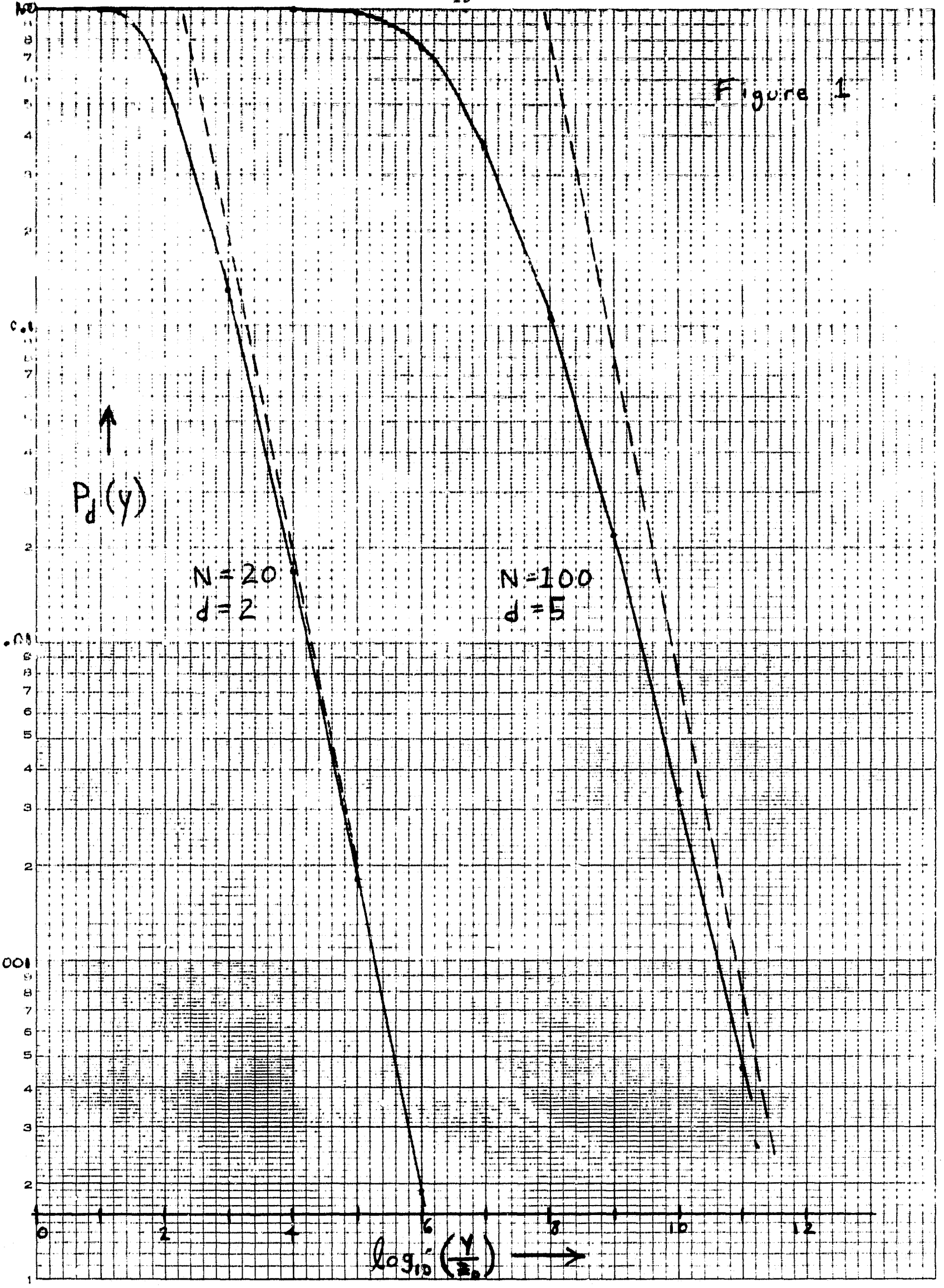


Figure 1
$P_d(y)$
N = 20
d = 2
N = 100
d = 5
$\log_{10}\left(\frac{Y}{\alpha_0}\right)$

Appendix B. On the Possibility of a New Hybrid Scheme

The Viterbi algorithm is an optimum decoding algorithm for convolutional codes on a white gaussian channel. Unfortunately, it requires too much memory to implement. In this appendix, we indicate how, if one could find an algorithm with essentially the same computational requirements and error probability as the Viterbi algorithm, but less memory requirements, a new type of hybrid scheme would become possible with attractive error probability and computational requirements.

Let us review the Viterbi algorithm, a complete discussion of which appears in Appendix A of our Final Report on a Coding System Design for Advanced Solar Missions, Contract NAS2-3637. With randomly chosen binary (q = 2) tree codes, the Viterbi algorithm gives an error probability equal asymptotically to

$$P_E \doteq 2^{-\nu E(r)}, \quad R \geq R_{comp}, \tag{1}$$

where $\doteq$ indicates asymptotic equality (though the actual coefficient is of the order of magnitude of 1), ν is the code constraint length in information bits, r is the code rate in bits per information bit, and $E(r)$ is the solution to

$$\frac{E_o(\rho)}{\rho} = r, \tag{2}$$

where $E_o(\rho)$ is the Gallager function for the channel under consideration. On the very noisy channel, which is what a white gaussian channel becomes in the limit of low rates and fine quantization,

$$E_o(\rho) = \frac{\rho}{1+\rho} C, \tag{3}$$

where C is channel capacity; thus we have

$$r/C = \frac{1}{1+\rho}\,; \quad \rho(r/C) = C/r - 1\,. \tag{4}$$

Thus at half of capacity (3 dB away) $\rho = 1$; at 5/8 of capacity (2 dB away) $\rho = 3/5$; and at 4/5 of capacity (1 dB away) $\rho = 1/4$.

There are two elements involved in evaluating the complexity of the Viterbi algorithm. First, there is the computational load: to advance a single bit, the algorithm must compute $2^{\nu+1}$ likelihoods and make 2^{ν} pairwise comparisons, so that the total computation per bit is asymptotically 2^{ν}, again with a coefficient of the order of unity. The algorithm also requires that each of 2^{ν} possible information streams be stored back to the point where a hard decision can be made, which, for a given rate r, is a length equal to some fixed constant $\lambda(r)$ times the constraint length. This requirement of $\lambda(r)\,\nu\,2^{\nu}$ memory cells tends to make the Viterbi algorithm impractical for moderately large ν.

Let us ignore this for the moment, and suppose that we have an implementable algorithm with

$$\begin{aligned} r/C &= \frac{1}{1+\rho}\,; \\ P_r(E)/\text{bit} &\doteq 2^{-\nu\rho}; \\ C/\text{bit} &\doteq 2^{\nu}. \end{aligned} \tag{5}$$

Let us now concatenate the convolutional code with an algebraic Reed-Solomon code on GF(q) with length N and rate $(1-2\tau)$, capable of correcting $N\tau$ errors. Previous work in concatenation suggests that good choices for these parameters will be

$$\begin{aligned} q = N &= 2^{\nu}; \\ (1-2\tau) &= r/C = \frac{1}{1+\rho}\,. \end{aligned} \tag{6}$$

Explicitly, then, what we have is 2^{ν} parallel convolutional codes of rate $r=\frac{C}{1+\rho}$, and an algebraic code across them of length 2^{ν} and rate $R = 1/1+\rho$, operating on ν-bit symbols.

For this scheme, the overall rate as a fraction of capacity will be

$$R_o/C = \left(\frac{1}{1+\rho}\right)^2. \tag{7}$$

The overall probability of error will be approximately

$$P_{e_o}(\varepsilon) \simeq \binom{N}{N\tau}\left[\nu P(\varepsilon)/b.t\right]^{N\tau}$$

$$\doteq 2^{N\mathcal{H}(\tau)}\, 2^{N\tau\log_2(\log_2 N)}\, 2^{-N\log_2 N\,\rho\tau}$$

$$= 2^{N}\left\{\mathcal{H}\left[\frac{\rho}{2(1+\rho)}\right] + \frac{\rho}{2(1+\rho)}\log_2(\log_2 N) - \log_2 N\left[\frac{\rho^2}{2(1+\rho)}\right]\right\}; \tag{8}$$

this quantity is the probability of error in decoding a whole block of $\nu x 2^{\nu}$ bits. Finally, the number of computations to decode this whole block is $\nu x 2^{\nu}$ $x 2^{\nu}$ convolutional code computations, and (with the use of Berlekamp's algorithm) something like $K(N\tau)^2$ algebraic computations, where K is a small constant, for a total of

$$(\nu + K\tau^2)\,2^{2\nu} = K' N^2 \tag{9}$$

computations in all, or

$$K' N^2/\nu N = K'' N \tag{10}$$

per decoded bit, where K'' is a constant of the order of unity. Combining (8) and (10), we see that we can obtain an error probability decreasing exponentially with N with a number of computations linear in N for all rates less than capacity.

The actual numbers we obtain are not quite as good as this asymptotic analysis would suggest. For example, if we use a (100, 80) 10-error-correcting outer code, we need a convolutional decoding error probability of about .01 over any 7-bit symbol to get an overall error probability of 10^{-6}; the bounds suggest that with $\rho = 1/4$ this will take a constraint length of 25-30; with these parameters the overall scheme will be 2 dB from capacity, 1 dB for the convolutional code and 1 dB for the algebraic. With a (1000, 800) 100-error-correcting outer code, the inner decoding error probability need only be brought down to .06, which again with $\rho = 1/4$ would require a constraint length of 16-20. Clearly these calculations of constraint length depend crucially on the accuracy of (1) at short constraint lengths.

We feel that this approach is promising enough that we ought at least to evaluate convolutional code error probabilities for short constraint length codes at rates very near capacity. A sequential decoder is an asymptotically optimum decoder of nonsystematic codes when the bias is set at the theoretical optimum, and becomes a true (not just asymptotic) maximum likelihood decoder if the bias is increased enough (to the point where all metric increments are nonpositive). We can therefore use our sequential decoding program to determine optimum convolutional code error probabilities. It would be a pleasant surprise if, as we increased the bias, the computational distribution developed a very sharp knee and thus became essentially bounded, as does the Viterbi algorithm. It is also possible that the necessary constraint lengths will be found short enough that the Viterbi algorithm itself can be implemented.

Appendix C. Computational Distributions with Normal Decoding

Figures 1 through 7 are plots of log $Pr(C \geq L)$ versus log L, where C is the measured number of computations per branch and L is a dummy variable. The following conclusions are based upon these distributions alone, and do not reflect the values of any of the other measured variables.

Figure 1 shows three pairs of curves, produced by codes of rate 1/2 and constraint length 99. Each pair corresponds to a different signal to noise ratio per information bit (E_b/N_o or SNR), namely 2.46 db, 1.47 db, and 0.48 db. (2.46 db is approximately R_{comp} at rate 1/2). Each pair consists of a systematic code and a nonsystematic code. The principal conclusion to be drawn from this figure is that there is no important difference in computation distribution between systematic and nonsystematic codes. The remaining simulations mostly use nonsystematic codes.

Since the curves of Figure 1 are nearly straight lines over much of their range, they may be approximated by Pareto distributions, of the form

$$Pr(C \geq L) = k\, L^{-\alpha},$$

where k is known as the coefficient and α is the Pareto exponent. The simulation program computes a theoretical Pareto exponent, while an experimental exponent may be estimated from the curves. The following table compares these quantities.

E_b/N_o	Theoretical	Experimental
2.46	0.97	0.95
1.47	0.48	0.54-0.60
0.48	0.087	0.38-0.41

The discrepancies in this table have not been completely explained. Since a large α implies fewer computations, it is encouraging to note that the experimental values are usually somewhat greater than the corresponding theoretical values.

Figure 2 shows the same sort of curves, resulting from the simulation of systematic codes of rate 1/4, with E_b/N_o of 1.86 db, 0.88 db, and

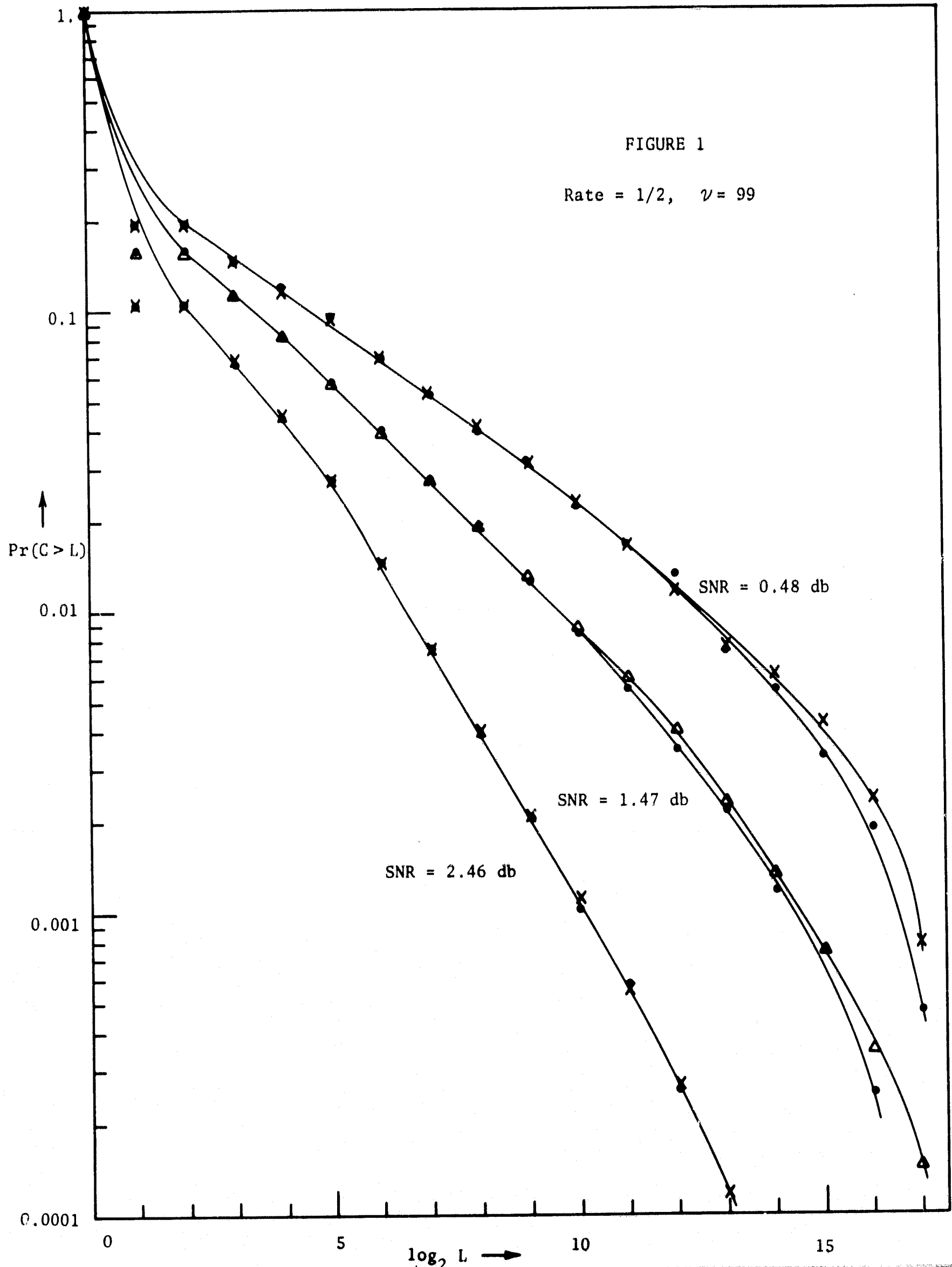


FIGURE 1

Rate = 1/2, ν = 99

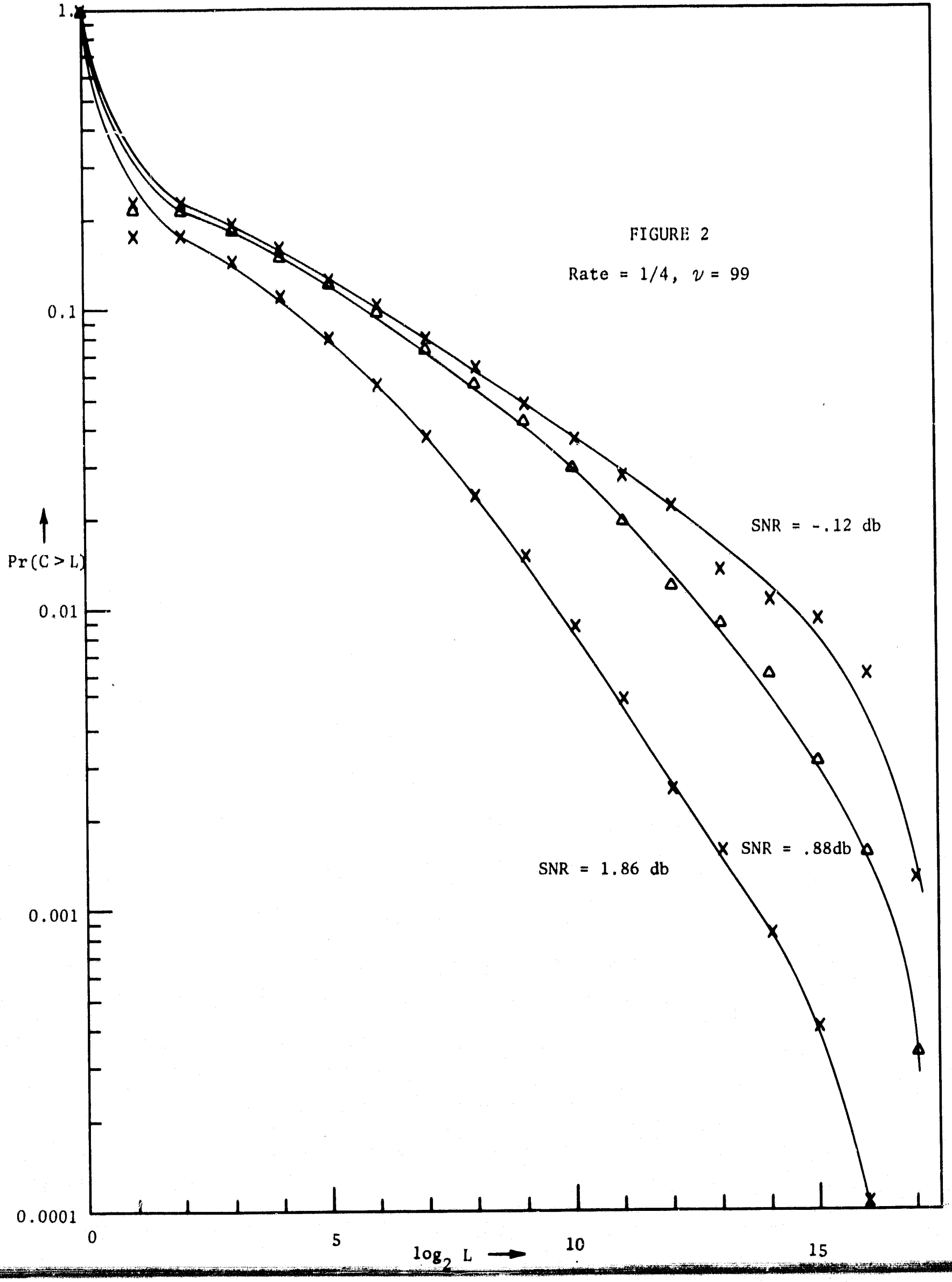


FIGURE 2

Rate = 1/4, ν = 99

-0.12 db. (In this case 1.86 db is about R_{comp}). The theoretical and experimental values of α are found in the following table, along with the experimental value from the curves of Figure 7.

SNR	Theoretical	Experimental
0.86	0.98	0.83
0.88	0.54	0.56
-0.12	0.185	0.27*-0.36

*from Figure 7

Figure 3 shows four curves, one of which is repeated from Figure 1 for comparison. The other three show the effect of changing the bias, a quantity used by the decoder algorithm to determine the metric increments. The bias experiment was conducted at rate 1/2 with E_b/N_o = 0.48 db, using a code having constraint length 6; it is seen that changing the constraint length from 99 to 6 results in fewer computations, but that the exponent, which is determined by the slope of the curve, is virtually unchanged. It is also seen that increasing the bias causes more computations and a smaller exponent, without enhancing the "knee", or rapid fall off at the right side of the curve. The apparent knees of the curves for nonzero bias are thought to be caused by the small size of the sample of measurements. Since the desired knee failed to appear, this approach was discarded.

Figure 4 shows five curves (three of which are virtually superimposed) resulting from codes of rate 1/4 with an E_b/N_o of -0.12 db. The three merged curves have frame lengths of 100 and constraint lengths of 20, 50, and 99 branches. This shows that the computation distribution is essentially independent of constraint length under these conditions. Of the remaining (unmerged) curves, the one showing the least number of computations resulted from a frame length of 50, while the curve showing a large number of computations resulted from a frame length of 200. This figure shows that a pronounced knee may be obtained by using a short frame length. The number of computations required to cause the curve to reach any given small probability is seen to be roughly proportional to a small power of the frame length.

Figure 5 shows two curves, corresponding to codes of rate 1/2 and 1/4 under comparable conditions, namely 2 db below the nominal R_{comp}. The

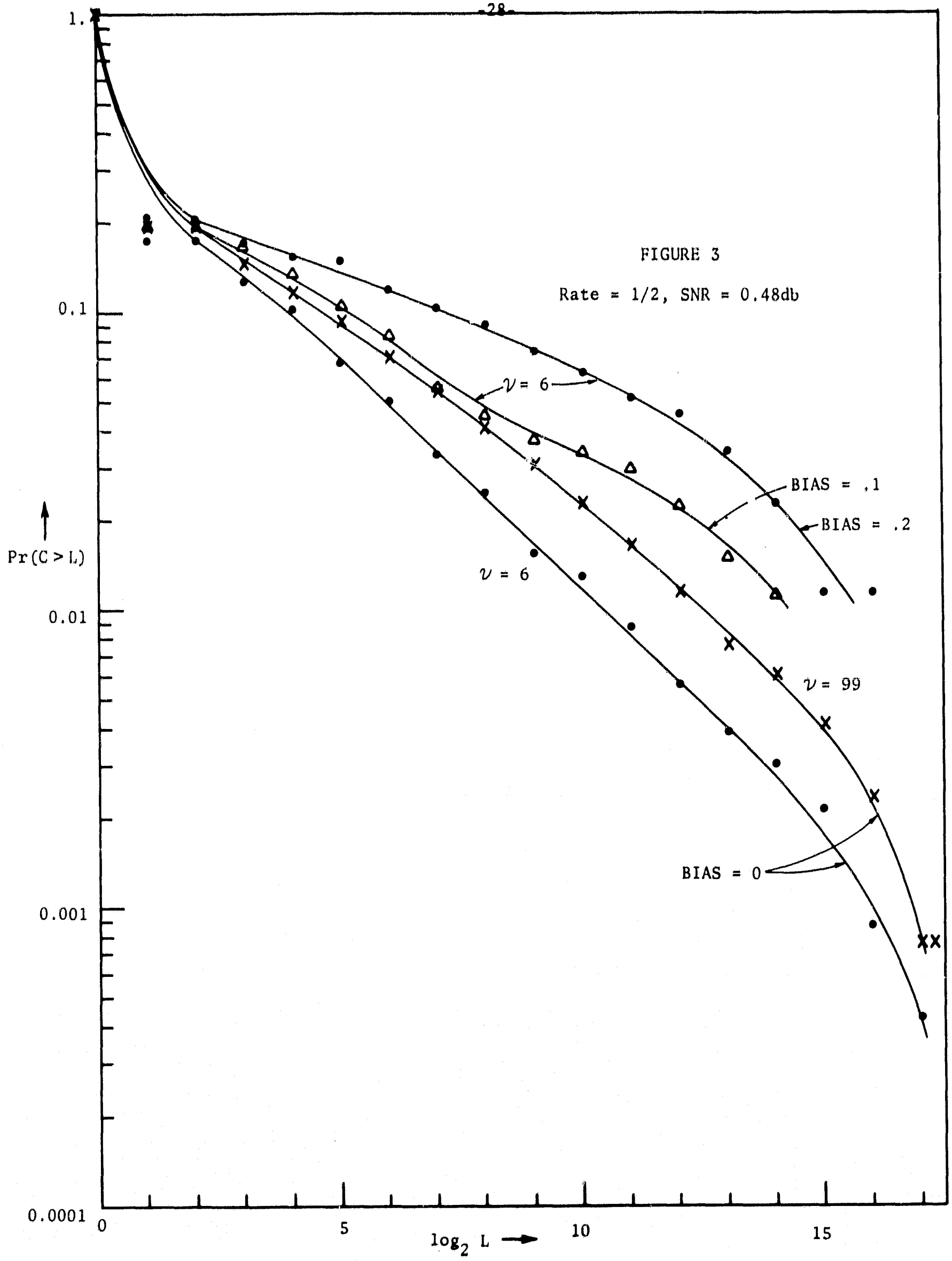
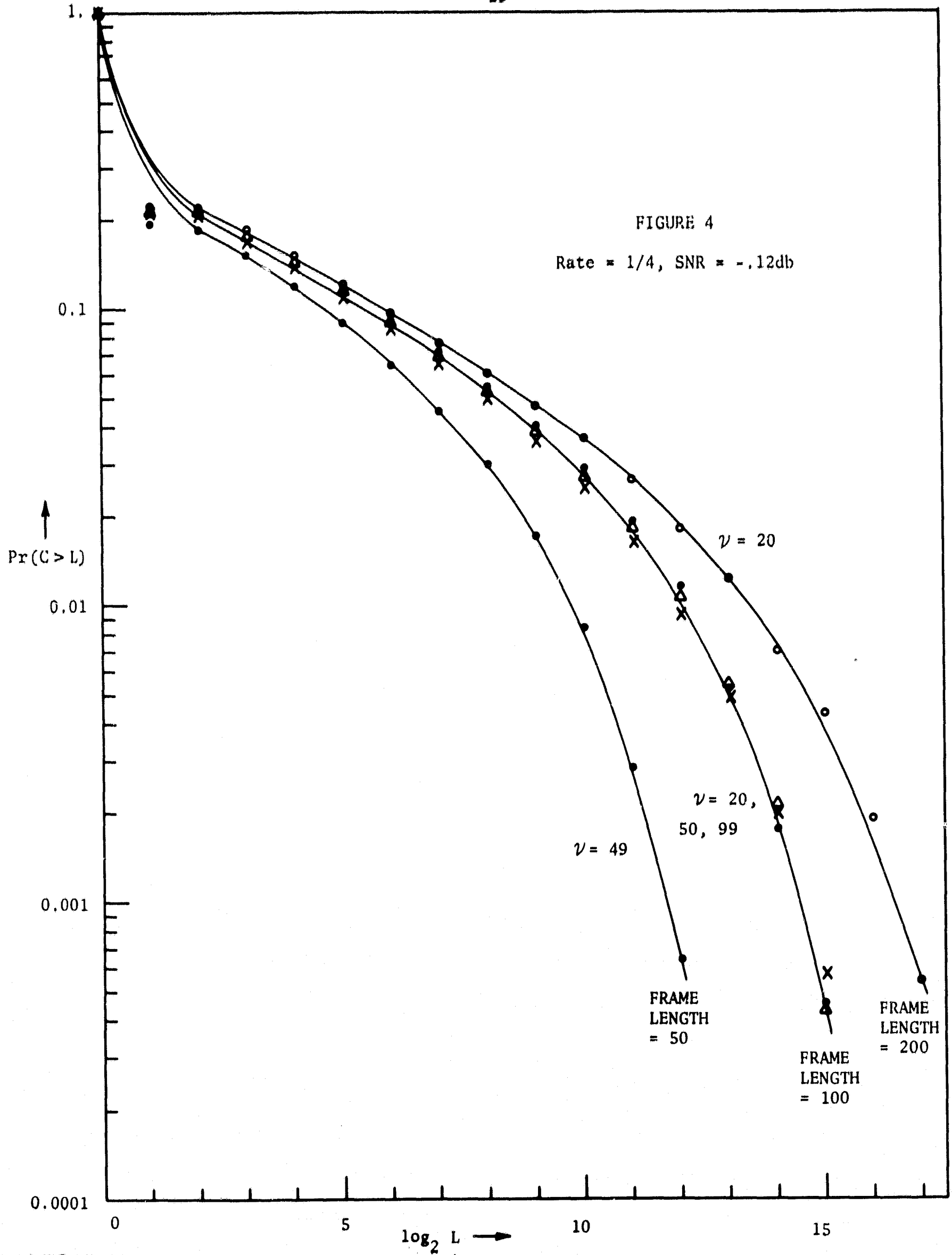


FIGURE 3

Rate = 1/2, SNR = 0.48db


1.
0.1
0.01
0.001
0.0001
Pr(C > L)
0
5
10
15
log2 L
FIGURE 4
Rate = 1/4, SNR = -.12db
ν = 20
ν = 20, 50, 99
ν = 49
FRAME LENGTH = 50
FRAME LENGTH = 100
FRAME LENGTH = 200

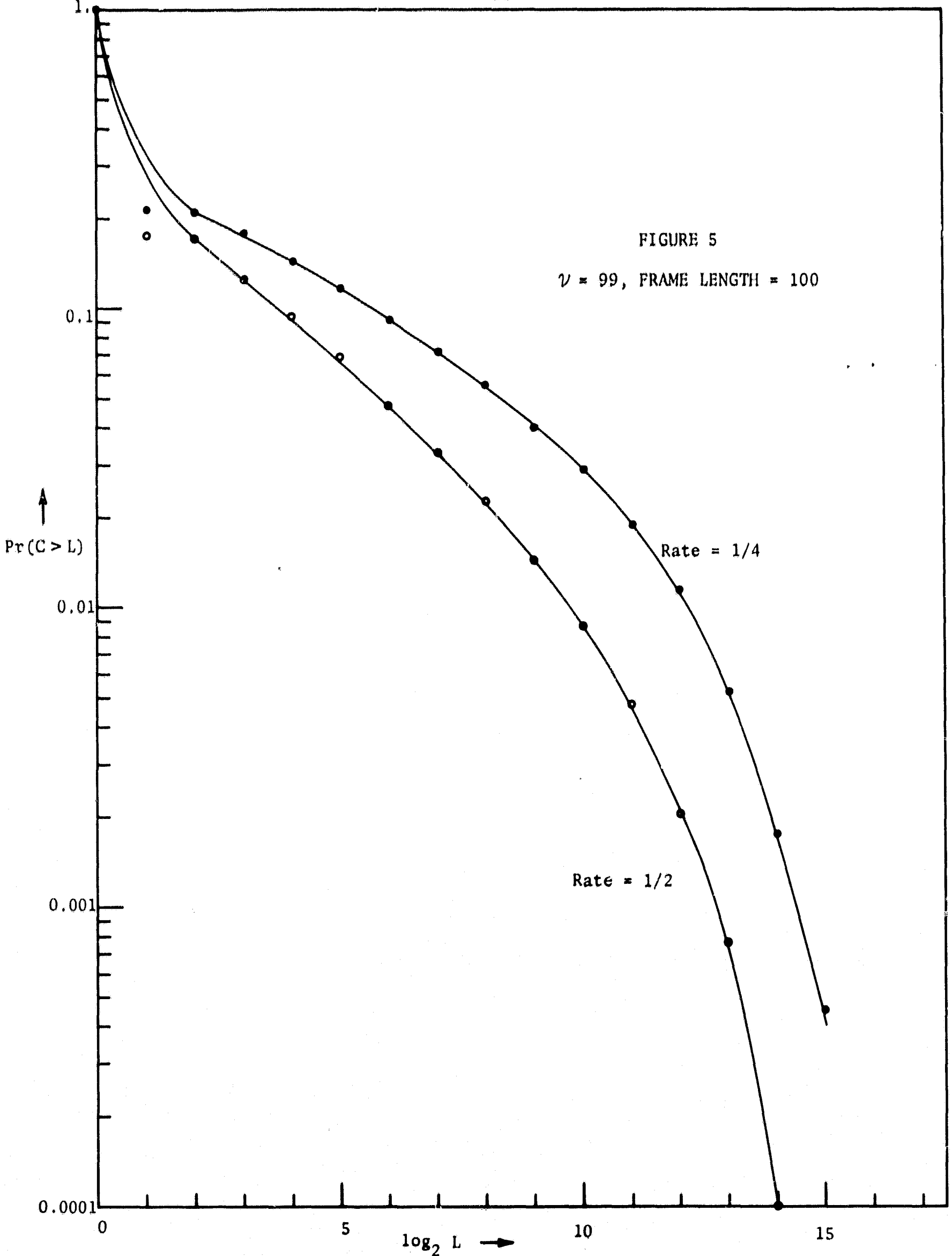


FIGURE 5

ν = 99, FRAME LENGTH = 100

code of rate 1/2 has somewhat fewer computations, although its curve does not seem to fall off as rapidly as that of the rate 1/4 code.

On the basis of the results shown in Figure 4, it was decided to try some simulations with a very short backsearch limit, in an attempt to produce a sharp knee in the computation distribution. Again, a rate 1/4 code was used at -0.12 db. A backsearch limit of 50 branches was imposed. Constraint lengths of 20, 30, and 40 branches were used. Little difference appears in the resulting curves, except that the constraint length 40 code required somewhat more computations than the others (here a particularly bad code was used by mistake). The curves all have pronounced knees at about the same place. Again, constraint length seems relatively unimportant as far as the computation distribution is concerned. We believe that in the constraint length 30 and 40 runs resynchronization was never successfully accomplished, in contrast to the run with constraint length 20 where typically resynchronization was established after a few hundred bits or less.

The last figure, Figure 7, shows the most dramatic and intriguing result. Again the parameters are rate 1/4, and E_b/N_o = -0.12 db, but now the constraint length is held constant at 20 while the backsearch limit is variable, with a large frame length (1000). The effect of the limitation on backsearches is clearly revealed as a sharp drop in the computational distribution at a number of computations C(B) dependent on the backsearch limit B. If we approximate C(B) by the number of computations at which $\Pr(C(B) \geq L) = 10^{-3}$, then $C(20) = 2^{9.3}$, $C(50) = 2^{13.2}$, $C(100) = 2^{16.5}$, which leads to the very rough approximation

$$C(B) \simeq .1B^3,$$

However, the observed probability of error with this limited backsearch is of the order of 10%, with typically 100 errors per error event.

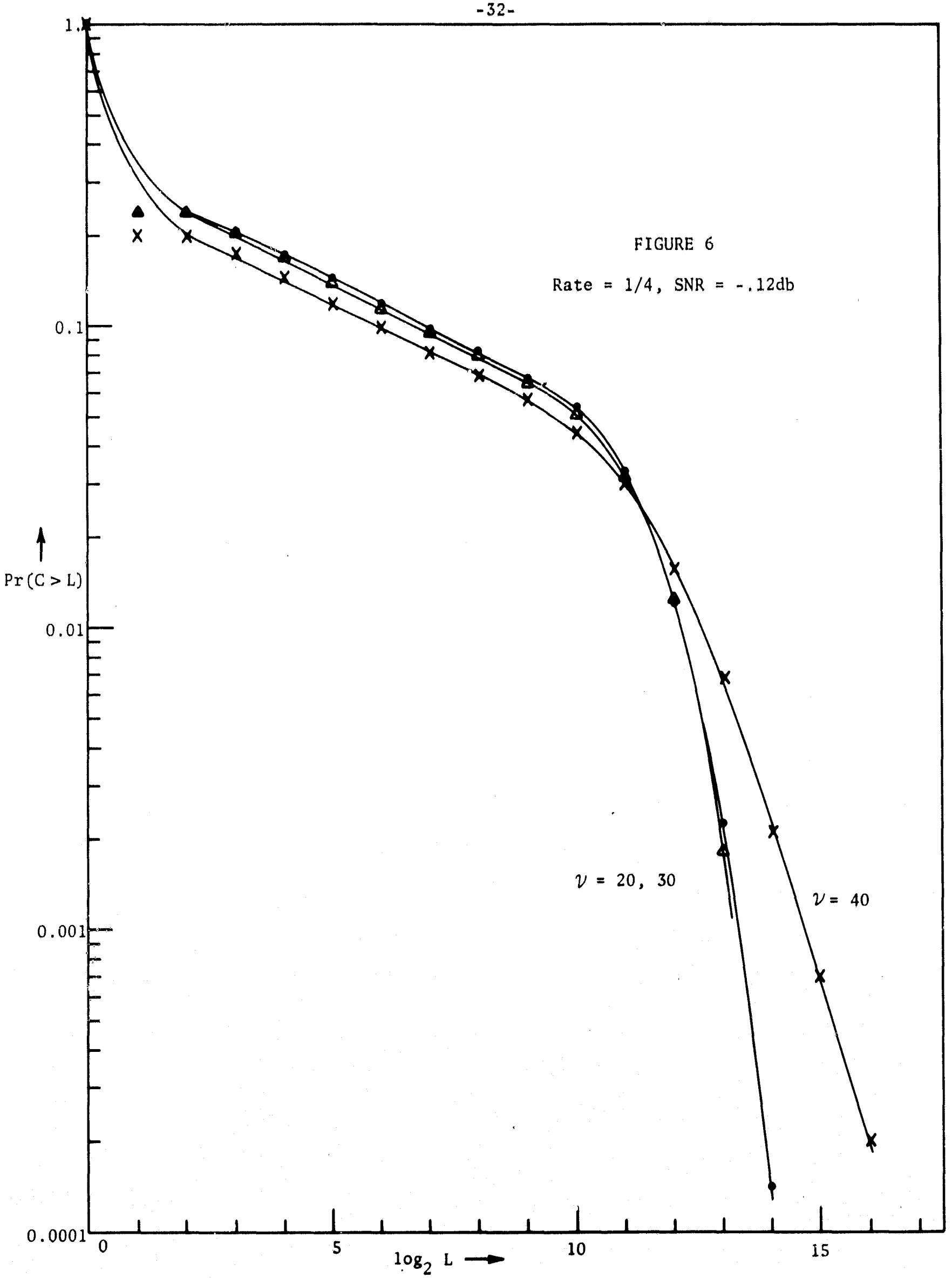


FIGURE 6

Rate = 1/4, SNR = -.12db

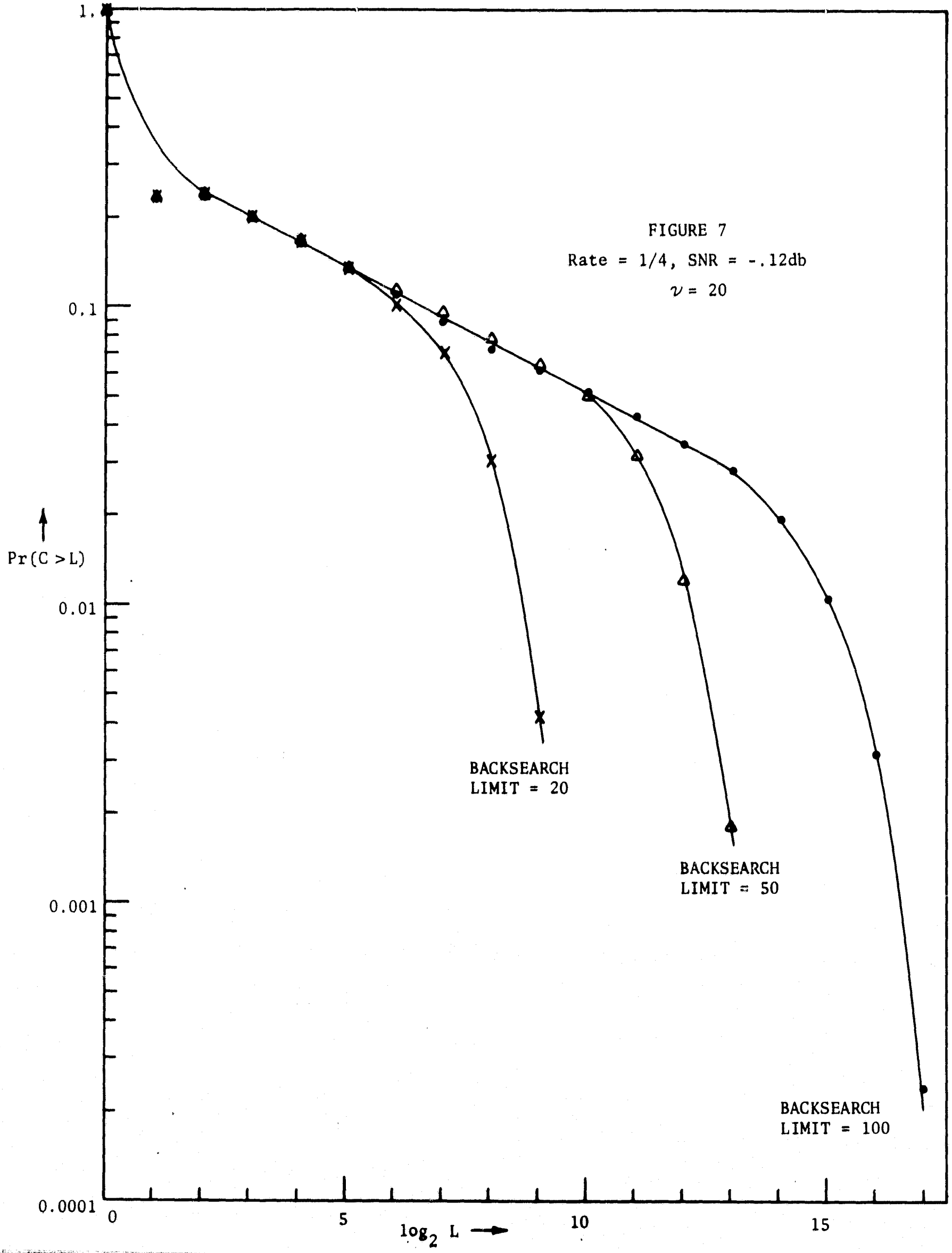


FIGURE 7

Rate = 1/4, SNR = -.12db

ν = 20

Appendix D. Computational Distributions with Genie Decoding During Resynchronization, and on the BSC

Figure 1 shows the computational distribution at 2 db below R_{comp} with a genie, for rate-1/4 codes of various constraint lengths ν with backsearch limits B=SD of 20, 50, and 100 branches. It is to the same scale as and should be compared with Figure 7 of Appendix C, where a genie was not used. It is clear that to the accuracy of the simulation the number C(B) of computations at which the distribution tails away rapidly depends only on B and not on constraint length or the presence or absence of a genie. Taking C(B) to be the C for which $Pr(C \geq L) \approx 10^{-3}$, we now estimate

TABLE I

B	C(B)
20	$2^{9.5} \simeq 700$
50	$2^{13} \simeq 8000$
100	$2^{16} \simeq 64000$

As in Appendix C, we note that these results fit the expression $C(B) = KB^3$ to within the accuracy of the simulation, where $K \approx .07$.

The much higher $Pr(C \geq L)$ for the lower values of L is explained by the fact that after the genie restarts the decoder B branches back, it causes it to traverse B extra branches, most with small L since the backsearch barrier is so close. The discrepancy in the non-tail regions of the $\nu = 10$, B = 100 curve is probably related to the significant number of ordinary undetected errors which occur when $\nu = 10$, whereas very few or none such occur with $\nu = 30$ or 99; the tail C(B) is still at the same place, however.

Figure 2 shows the results of varying the signal-to-noise ratio with a rate-1/4 code of constraint length 30 and a backsearch limit of 100 branches. The E_b/N_o used were -.124 db (2 db below R_{comp}), .382 db (1.5 db below R_{comp}), and 1.386 db (.5 db below R_{comp}). Although for better E_b/N_o the lower part of the curve is improved in magnitude (due to fewer errors being made and the genie being invoked less frequently) and slope (due to the theoretical increase in magnitude of Pareto exponent), the tail location C(B) is steady as a rock, perhaps even increasing slightly; another indication that C(B) is insensitive to all parameters but the backsearch limit B.

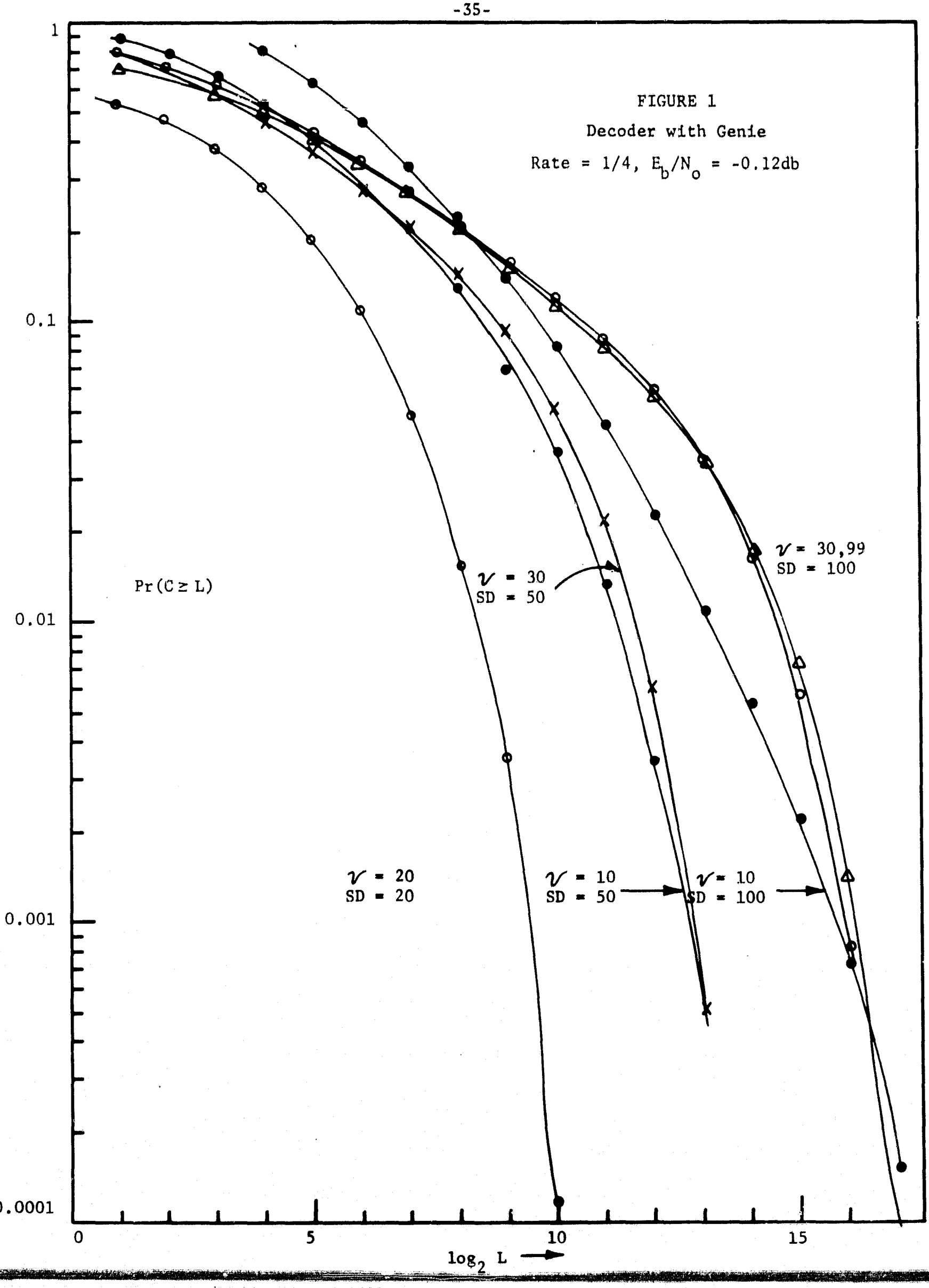


FIGURE 1

Decoder with Genie

Rate = 1/4, E_b/N_o = -0.12db

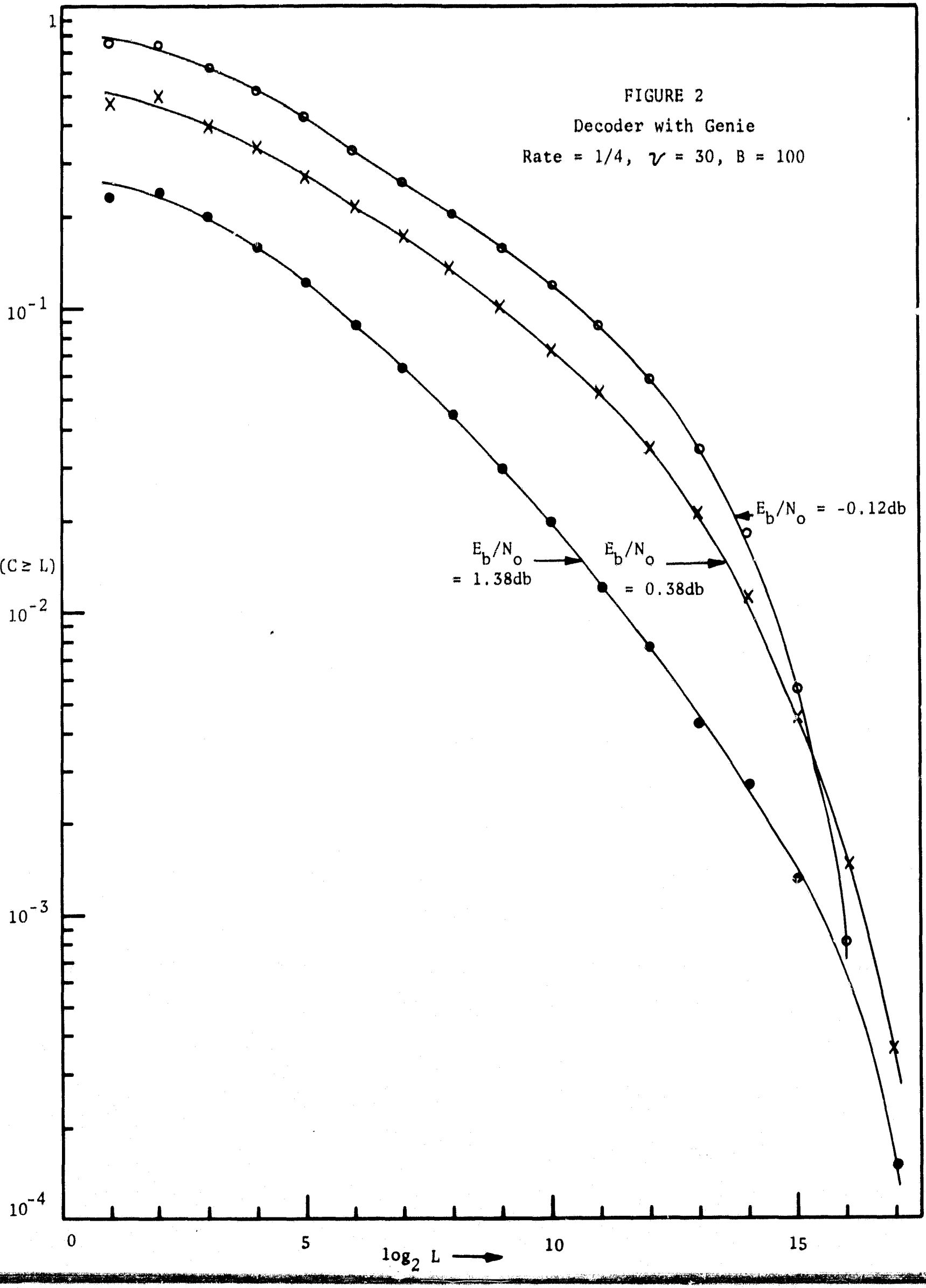


FIGURE 2

Decoder with Genie

Rate = 1/4, ν = 30, B = 100

Figure 3 compares the computational distributions obtained during resynchronization trials with those described above. The results are not conclusive, but are consistent with the hypothesis that the tail C(B) occurs at the same place as before, with the evidence being very strong for $\nu = 20$ and $\nu = 30$ with B = 50. The discrepancies at low L are due to the absence of the genie during resynchronization.

Figure 4 shows the effect of increasing the bias in the decoding metric from its nominal value, equal to the rate R in bits/bit, (used in all other simulations), to .02 and .05 bits/bit (.08 and .2 bits/branch) above nominal, during resynchronization trials with a rate-1/4 code of constraint length 20 with B = 100 at 2 db below R_{comp}. The results are very fragmentary, with a large number of incomplete frames due to computational overflow (declared at $L = 10^5$). It seems likely, however, that even small increases in the bias increase the tail markedly. The following table gives the number of computational overflows observed in 20 resynchronization trials:

TABLE II

Bias above nominal	Overflows ($L = 10^5$) in 20 trials
0.0	0
0.02	11
0.05	15

Figures 5-7 show computational distributions obtained on the binary symmetric channel with the DDP-116 program. Figure 5 gives distributions obtained for 10^5 bits at a probability of error p = 3/64 = 4.7%, slightly above the p for which $R_{comp} = 1/2$, and for p = 50%, data totally unrelated to the code. The code used was a rate-1/2 systematic code of length 47. The low L, high $Pr(C \geq L)$ parts of these curves are without significance, the take-off being due to the method of tabulation. The tails of these curves, however, follow the pattern observed with the other program. The striking feature is that the tails occur in the same place regardless of whether p = 4.7% or 50%. Comparing the curves in detail, we find that the two for B = 32 are nearly identical, the two for B = 64 have the same character but differ in magnitude by about a factor of two, while the B = 128 there is nearly two codes of magnitude difference (the p = 4.7% curve has been moved up an order of magnitude with respect to the other curves). This correlates precisely with the facts that at p = 4.7%

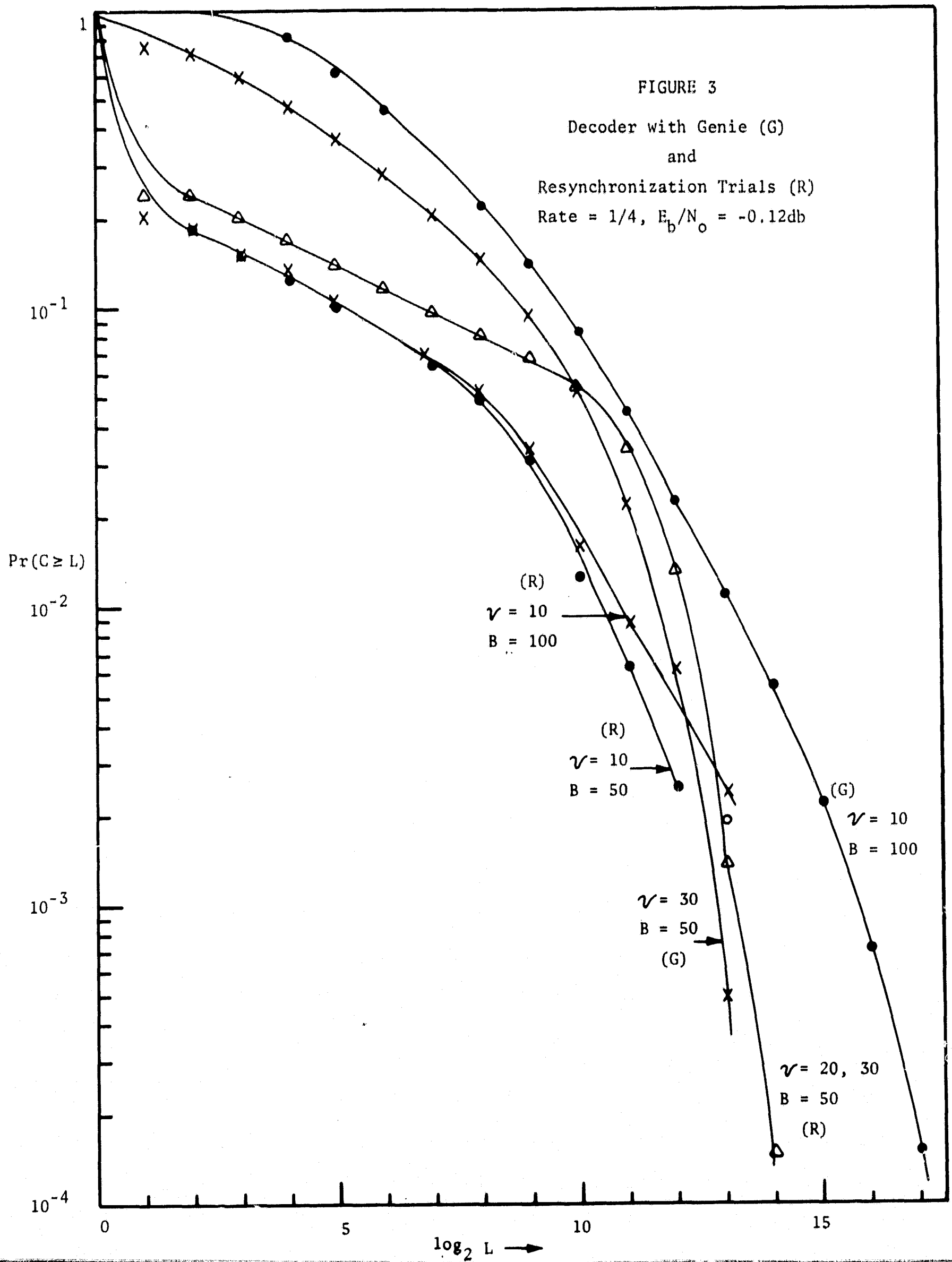


FIGURE 3

Decoder with Genie (G)
and
Resynchronization Trials (R)
Rate = 1/4, E_b/N_o = -0.12db

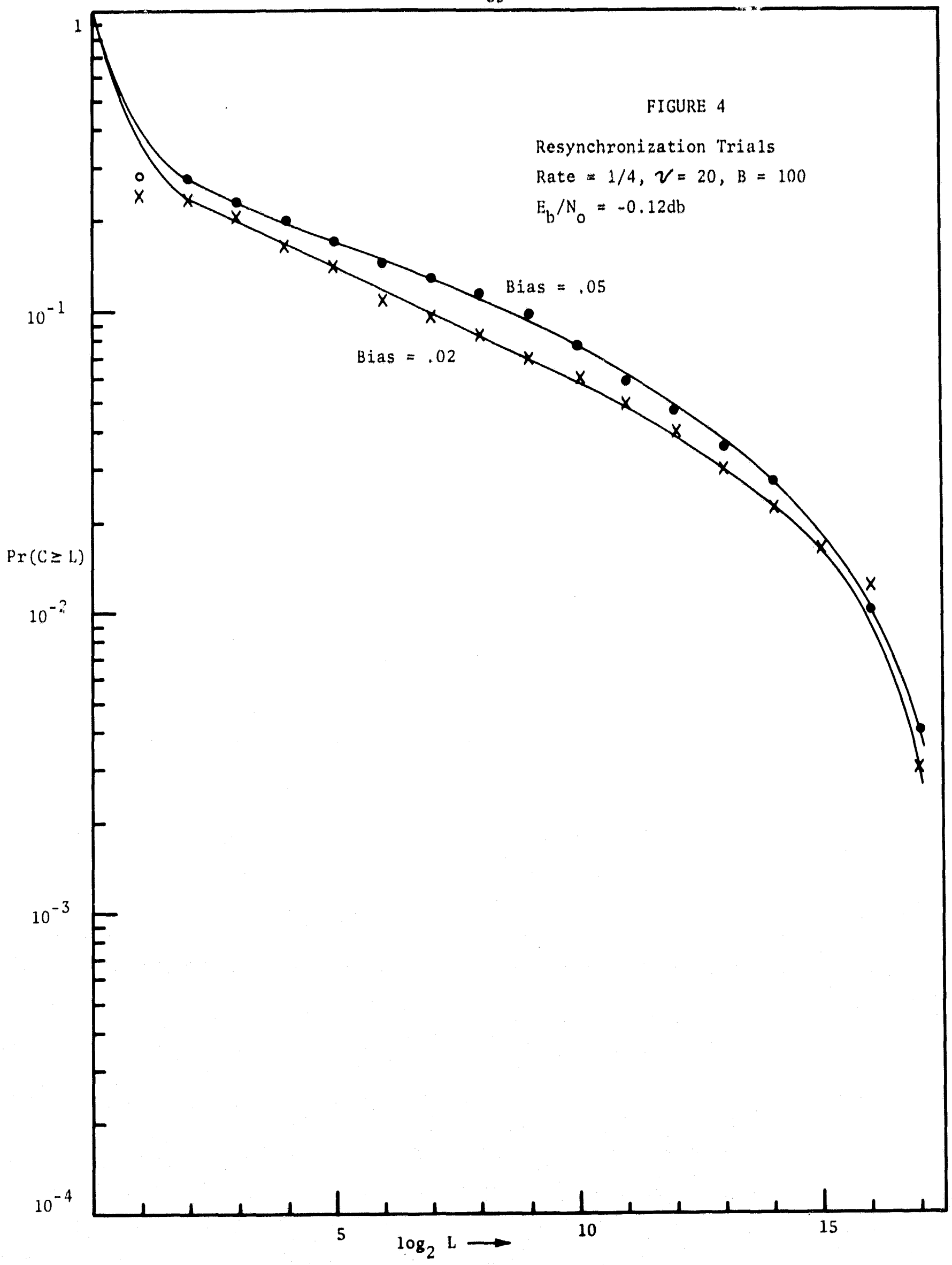


FIGURE 4

Resynchronization Trials

Rate = 1/4, ν = 20, B = 100

E_b/N_o = -0.12db

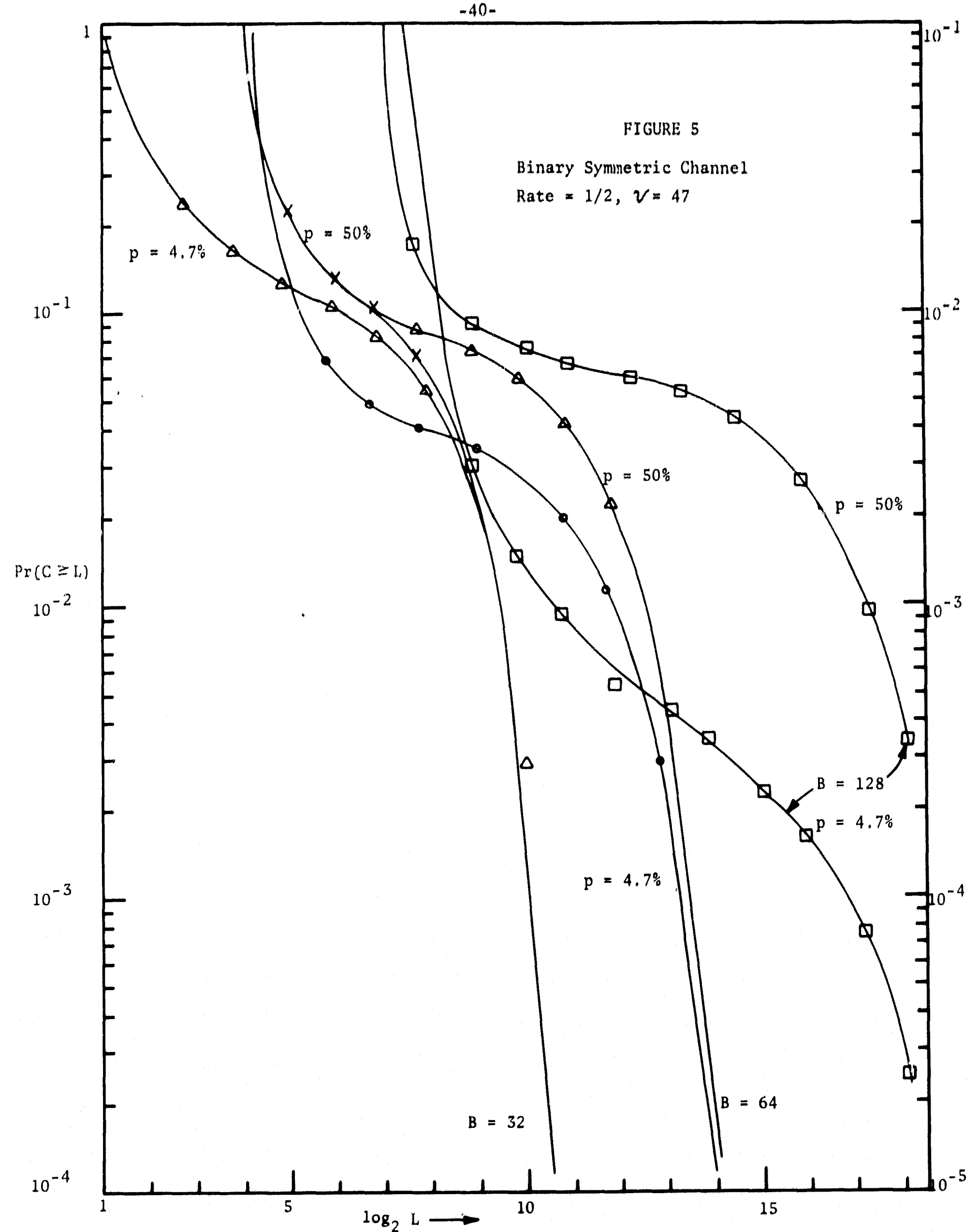


FIGURE 5
Binary Symmetric Channel
Rate = 1/2, $\mathcal{V}$ = 47
p = 4.7%
p = 50%
p = 50%
p = 50%
B = 128
p = 4.7%
p = 4.7%
B = 64
B = 32
Pr(C ≥ L)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
1
5
10
15
$\log_2$ L

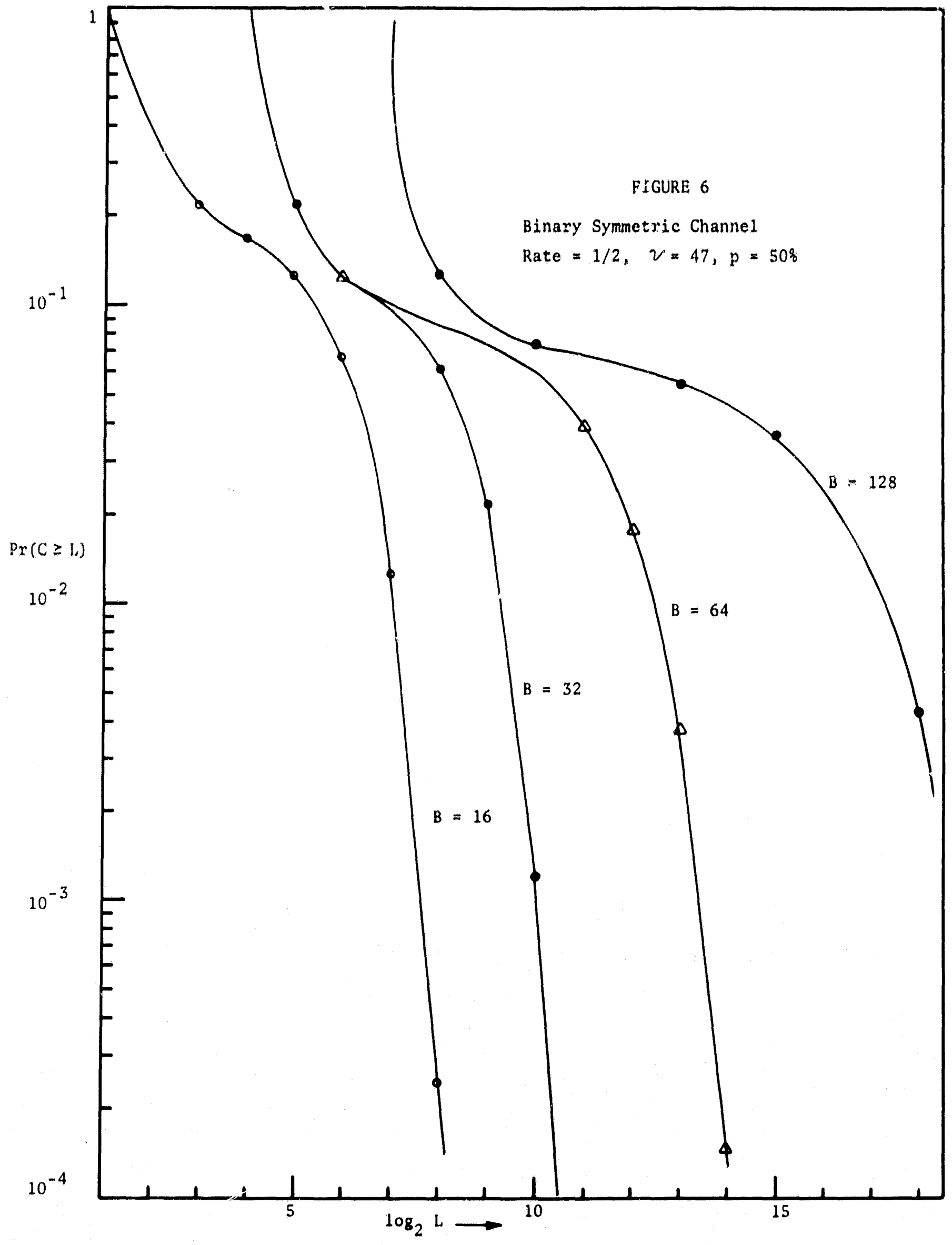


FIGURE 6

Binary Symmetric Channel

Rate = 1/2, ν = 47, p = 50%

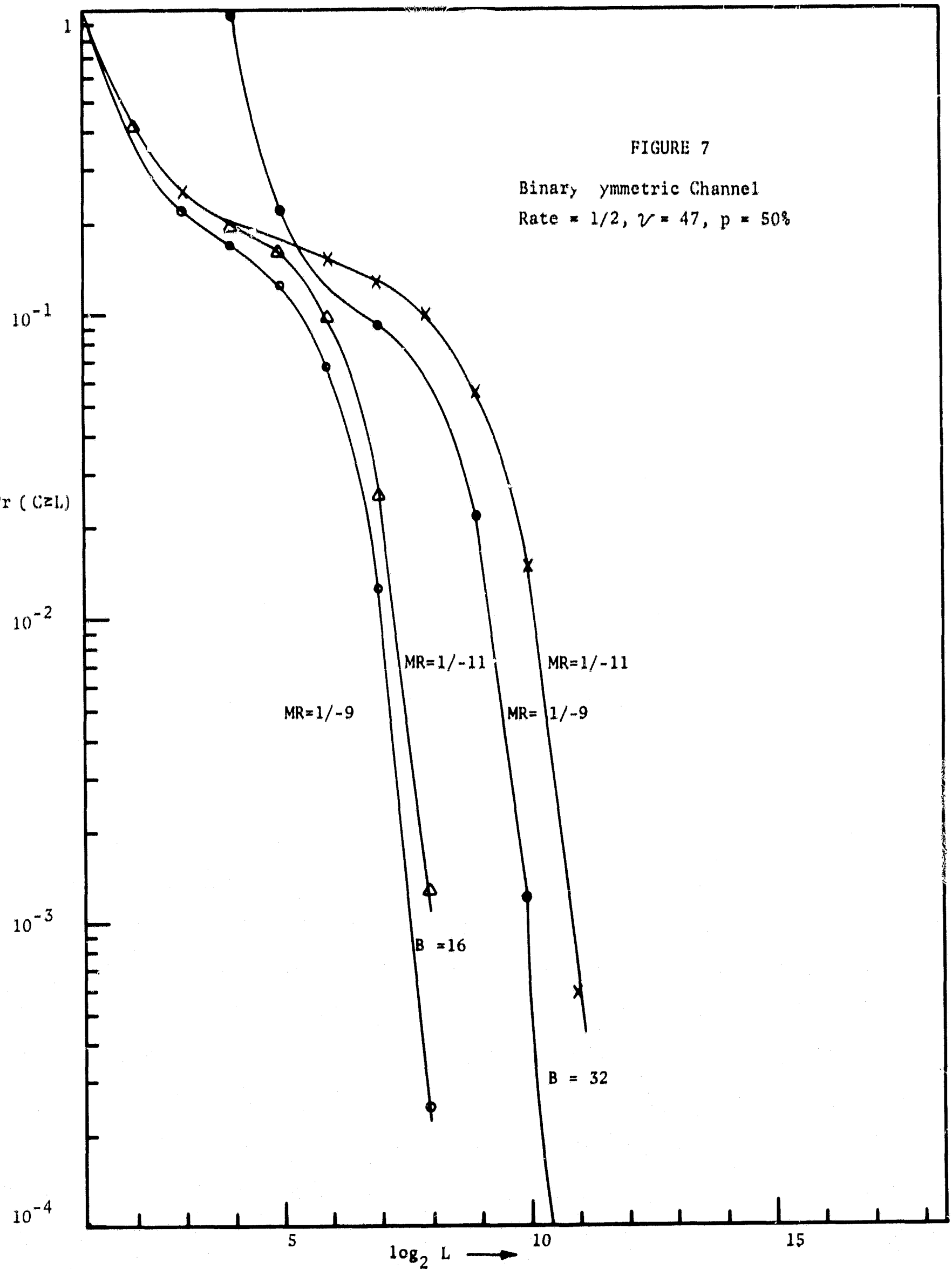


FIGURE 7

Binary symmetric Channel

Rate = 1/2, ν = 47, p = 50%

the decoder was making errors and trying to resynchronize 93% of the time with B = 32, 45% of the time with B = 64, and 1% of the time with B = 128. We are led to the conclusion, upon which we expand elsewhere, that when the decoder is trying to resynchronize, it is seeing essentially random data, and that its behavior during these times dominates the computational distribution for large L.

Figure 6 therefore shows the computational distribution with p = 50%, for backsearch limits of 16, 32, 64, and 128. The number of bits in these runs were 10^5, 10^5, 2×10^4, and 3.2×10^3. If we take the tail C(B) as the 10^{-3} point again, we obtain Table III:

TABLE III

B	C(B)	$\frac{2^{-.5} B^{1.8} 2^{.05B}}{2^{7.5}}$
16	$2^{7.6} \simeq 200$	
32	$2^{10} \simeq 1000$	$2^{10.1}$
64	$2^{13.4} \simeq 10000$	$2^{13.5}$
128	$2^{18.5} \simeq 320000$	$2^{18.5}$

The steps are quantitively similar to those of Table I, but it is no longer clear that a curve of the form $C(B) = KB^3$ best fits the data. In fact the points fit neither the simple algebraic curve $C(B) = KB^{\alpha}$ nor the exponential curve $C(B) = K2^{\beta B}$. For these four points, however, we find a rather good fit of the form $C(B) = KB^{\alpha} 2^{\beta B}$, with $K = 2^{-.5}$, $\alpha = 1.8$, and $\beta = .05$, which good fit is to be expected with three free parameters to fit four points.

Finally, Figure 7 shows the effect of changing the bias, for B = 16 and 32. In terms of the ratio MR between the metric increment given a correct bit and that given an incorrect, the two values are MR = +1/-9 (used in all other curves) and MR = 1/-11. For p = .045, where the nominal metric ratio is 1/-9.1, this would correspond to bias terms [above nominal bias = 1/2 bits/bit (1 bit/branch)] of -.0054 bits/bit (-.01 bits/branch) and +.074 bits/bit (.15 bits/branch). The tail moves out in each case, but by less than a factor of two, indicating not much effect of bias on tail, counter to what was previously suggested. We note that the effect does tend to increase with B, however, and recall that our earlier observations were with B = 100.

Appendix E. Error Probability Results

We tabulated three quantities: the probability of error event Pr(EE), the probability of bit error Pr(E), and the probability of being in an error event Pr(in EE). An error event begins when the decoder is not in an error event and an error passes the backsearch limit; it ends when a complete constraint length of correct data passes the backsearch limit; and its length is defined as the number of branches from the first to the last error. Pr(EE) is then the number of error events divided by the total branches decoded, while Pr(in EE) is the total length of error events divided by the number of branches decoded. Pr(E) is just the total bit errors divided by total branches decoded.

Table IV tabulates Pr(EE), Pr(E), and Pr(in EE) for rate 1/4 codes of various constraint lengths at 2 db below R_{comp}, for backsearch limits 20, 50, and 100. In all cases a genie was used.

TABLE IV

ν	B	Pr(EE)	Pr(in EE)	Pr(E)
20	20	.016625	.424375	.10525
10	50	.02195	.1819	.06345
30	50	.01075	.307	.04935
10	100	.02113	.1391	.05106
30	100	.0100	.1269	.0222
99	100	.0034	.5126	.0207

Starting from the bottom, at B = 100 Pr(E) is about the same for ν = 30 and ν = 99, indicating that in either case the event which dominates the error probability is that in which a normally correctable error has not been picked up 100 branches later, rather than the undetected error event. Pr(in EE) is much higher for ν = 99 because the definition of error events changes with ν, and with Pr(E) = 2% the probability of an error in a run of 100 branches is high. With ν = 10, we are clearly seeing many undetected errors as well, giving another 3% of Pr(E). At B = 50, the discrepancy between ν = 30 and ν = 10 is less marked, probably since there is more overlap between the undetected error and backsearch violation events. The most discouraging thing

is the slow falloff of Pr(E) with B; if we assume that undetected errors are negligible for B = 20 and 30, then we have an irreducible Pr(E) of 10% at B = 20, 5% at B = 50, and 2% at B = 100, due entirely to backsearch violations.

We also recorded the error probability as a function of E_b/N_o, for a rate-1/4 code of constraint length 30 with B = 100, shown in Table V.

TABLE V

E_b/N_o	Pr(EE)	Pr(in EE)	Pr(E)
-.124 db	.0100	.1269	.0222
.382 db	.00535	.0450	.0103
1.386 db	.00067	.00123	.00067

We see that varying the signal-to-noise ratio is effective in reducing the error rate; however, this is precisely the route we wish to avoid.

Finally, the recorded error rates on the DDP-116 were universally Pr(E) = 50% when p = 50%, thank goodness; when p = 4.7%, the error rate was 13.8% at B = 32, 6.5% at B = 64, and .125% at B = 128. This has a very nice interpretation: while the decoder is trying to resynchronize, it will decode to a sequence which differs from thy received sequence in about 11% of the places, since the Gilbert bound says that a sequence picked at random will with high probability be distance .11n from some code word in a rate-1/2 code over a length n, since $1 - \mathcal{H}(.11) = 1/2 = R$. With p = 4.7%, therefore, the decoded sequence will have a density of errors of about (.953)x(.11) + (.047)x(.89) = .14666. Thus, if the decoder were hung up all the time, we would expect an error probability of 14 2/3%. The percentages of hangup of 93%, 45% and 1% square very closely with the error rates observed. This interpretation therefore gives additional evidence that while the decoder is hung up it behaves as though the input data were random.

Discussion

The fact that even with the genie and a backsearch limit of 100 branches, the error probability is about 2% at 2 db below R_{comp} is disheartening. This means that every fiftieth bit requires a search extending over more than 100 branches to be corrected; i.e., that it takes more than 100

branches for all paths leading from the incorrect hypothesis to fall below the minimum point on the correct path. Furthermore, these errors are not closely bunched, but rather diffuse, as can be seen from the fact that the probability of being in an error event exceeds 10%, and as is also seen in the detailed printouts of the statistics of each error event. Looking now to the use of such a code in hybrid schemes, we find that this hurts us in two situations:

1) If we use a more efficient symbol-error-correcting code, such as a Reed-Solomon, rather than a binary code, the probability of error for an n-bit symbol is nearly n times the bit error probability. Possibly this effect could be avoided if we used a nonbinary convolutional code with the sequential decoder.

2) If we wish to correct erasures as well as errors, and are able to detect substantially all error events, the erasure probability will be at least of the order of the probability of being in an error event, again much higher than the bit error probability.

A few quick calculations show how far we are from an acceptable scheme. Let us suppose that the inner code parameters are $E_b/N_o = R_{comp}$ -2 db, $\nu = 30$, $B = 100$, so that the bit error probability is 2%. Suppose first a binary algebraic code as the outer code. The capacity of a binary symmetric channel with $p = 2\%$ is about .85 bits; thus if we could signal at capacity we would incur a rate loss of at least .7 db. Actually, to get a low error probability of error with the known binary codes would require much greater rate losses. Suppose then that we go to a non-binary code, and that the symbol error probability can be held to 10%; the most efficient error correcting code would have to have 20% check bits, for a 1 db rate loss. Even if we used erasure-correction and could hold the erasure probability to 10% and the error probability to zero, with a (100, 80) code the decoding error probability would be about 10^{-3}, much too large in view of the fact that every decoding error must be followed by resynchronization. With a (1000, 860) code the decoding error probability would be greater than 10^{-5}, while with a (1000, 800), (200-erasure-correcting) code the decoding error probability would be more like 10^{-20}; thus with codes of length 1000 the rate loss would be between .7 and 1 db.

We conclude that under the most favorable assumptions and with very complicated algebraic decoding the rate loss in the algebraic code will be of the order of 1 db, perhaps more, thus reducing the net gain of the concatenated scheme to 1 db below R_{comp}.

Whether or not anything can be done about this depends on how closely error events are associated with large search events. The object of all our schemes has been to obtain an acceptably small error probability with a limited amount of sequential decoder computation. The most direct approach would be to set a limit C_o and to invoke the algebraic decoder whenever C_o computations were required to advance a single bit. Now if the algebraic decoder can correct e errors, and e+1 streams still have errors at symbol b after C_o computations on each, there is nothing to be done. Thus if each of our observed errors is accompanied by $C(100) \approx 64{,}000$ computations, then we cannot improve on the discouraging results obtained above. If, however, there is little correlation between long searches and errors, then we might be able to do better. The data we have suggests, but does not prove, that the correlation is strong. The fact that in Figure 7 of Appendix C the distributions of computation break away from nominal at about 10%, 5% and 2% at B = 20, 50, and 100 invites the hypothesis that the searches artificially terminated by the backsearch limit are indeed the ones in which errors persist at a depth of B or more branches. In summary, the hope of obtaining more than about 1 db gain through hybrid techniques now appears bleak.