2

CR 114450

STUDY OF SEQUENTIAL DECODING

FEB. 1972

FINAL REPORT TO:

NATIONAL AERONAUTICS

AND

SPACE ADMINISTRATION

CONTRACT NAS 2-5643

INFORMATION AND DECISION THEORY GROUP

SCHOOL OF ELECTRICAL ENGINEERING

CORNELL UNIVERSITY

Ithaca, New York

241 pg.

NASA CR 114450

Available to the Public

NASA CR

STUDY OF SEQUENTIAL DECODING

By Dr. Frederick Jelinek

February 1972

FINAL REPORT

Prepared under Contract No. NAS 2-5643

School of Electrical Engineering
Cornell University
Ithaca, New York

for

AMES RESEARCH CENTER

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

I. INTRODUCTION

This final report on project NAS 2-5643, Study of Sequential Decoding consists of two main portions: results of Phase I and II of our research. Covered are results obtained in the period September 1970 through January 1972. Earlier work concerning September 1969 through August 1970 is contained in the Annual Report of September 1970.

Phase I deals with problems of reliable transmission through noisy space channels and is subdivided into nine areas reported on in Chapter II. (see Table of Contents).

Phase II of the project deals with problems of encoding of space sources for the purpose of data compression. It is subdivided into four areas that are reported in Chapter III.

Chapter IV lists the theses, publications, and talks that were based on work supported by this project.

A substantial portion of this report has already been presented in Quarterly Progress Reports 5 through 8.

II. REPORT ON PHASE I

II-A. Theoretical Performance Curves for Bootstrap Sequential Decoding

We have evaluated $R^{L}_{BOOT}(1)$ and $R^{U}_{BOOT}(1)$ vs. 10 log E_b/N_o performance curves of quaternary and octal quantized Gaussian channels with binary antipodal inputs. E_b denotes the energy per information bit. As previously, the rates given do not include the degradation factor $\frac{m-1}{m}$ corresponding to the single parity algebraic code. Each of the curves includes parameter values K denoting the least number of streams for which the former are valid (for m <K, better performance is obtainable). The performance curves were obtained for uniform quantization at the receiver, whose intervals were optimized with the help of Figures 1 through 7. The latter are parametric curves (with respect to a fixed SNR) showing the performance as a function of varying quantization size. It is interesting to note that in each figure, the optimal quantization size (in fractions of E_b/N_o) is almost invariant to any changes in the value of E_b/N_o.

Figures 1 through 5 deal with $R^{L}_{BOOT}(1)$ and correspond to the following cases: Quaternary channel with a binary state stream (1) and with a full (quaternary) state stream (2), Octal channel with a binary state stream (3), with a quaternary state stream (4), and with a full (octal) state stream (5). In case (4) the quaternary state stream was obtained by lumping together the three neighboring output digits that correspond to the extreme quantization values

on each side of the 0 point (this is the optimal lumping procedure).

Figures 6 and 7 deal with $R^{U}_{BOOT}(1)$ for the quaternary (6) and octal (7) channels with a binary state stream.

Figures 8 through 12 give then the $R^{L}_{BOOT}(1)$ vs. 10 log E_b/N_o relationship for optimal uniform quantization at the receiver. All these curves contain parametric indications of E_s/N_o(dB) performance, where E_s is the energy per transmitted bit. Also shown are the previously mentioned K-limits. Figure 8 compares the performance of Bootstrap Hybrid Decoding for binary, quaternary, and octal quantization with full channel state streams. It can be seen that in the limit of low rates, quaternary quantization constitutes an improvement of about 1.35 dB over binary, and octal quantization constitutes a 0.35 dB improvement over quaternary. Figure 9 shows the same relationships for a binary state stream. There, quaternary quantization is 1.4 dB better than binary, and octal is 0.4 dB better than quaternary.

Figure 10 contains R_{comp}, $R^{L}_{BOOT}(1)$, and capacity curves for binary quantization. In the limit of low rates, bootstrap decoding has a 1.7 dB advantage over sequential decoding (the degradation factor $\frac{m-1}{m}$ is not included). Figure 11 contains R_{comp}, $R^{L}_{BOOT}(1)$ and capacity curves for quaternary quantization. It is seen that a full state stream enjoys a noticeable advantage over a binary one only for rates $R > 1/4$. Furthermore, this advantage is always small (at most 0.15 dB). Again, in the limit of small rates, bootstrap decoding is about 1.7 dB better than sequential decoding. Figure 12 concerns octal quantization. An octal state stream is nowhere noticeably better than a quaternary one, and a binary state stream is worse than the latter only for $R > 1/4$. The 1.7 dB advantage over sequential decoding is again evident.

Next, Figure 13 compares $R^{U}_{BOOT}(1)$ performances of binary, quaternary, and octal quantization with a binary channel state stream. The curves have a slight upward slope for low rates, indicating that the upper bound tightens as the rates decrease. In fact, comparison with Figure 9 shows that the $R^{U}_{BOOT}(1)$ and $R^{L}_{BOOT}(1)$ limits are the same!

It should be noted that Figures 10 through 12 show a consistent 1.1 dB capacity over $R^{L}_{BOOT}(1)$ advantage. This shows that worthwhile improvement might be obtainable from use of more sophisticated algebraic "outer" codes.

The final six curves (Figures 14 through 19) pertaining to this section show the Pareto exponent as a function of SNR per transmitted bit (in dB) for Bootstrap and straight sequential decoding at fixed track rate $R = \frac{1}{2}$ (the degradation factor is not included). γ_{upper} denotes the exponent obtainable from the upper bound and γ_{lower} that from the lower bound on bootstrap decoding. Finally, $\sigma(\infty)$ denotes the exponent for sequential decoding.

All the curves show that γ_{upper} and γ_{lower} approach each other with increasing SNR, and pull away from $\sigma(\infty)$. It is interesting to note (compare Figures 14, 15, and 17 and Figures 16 and 18) that performance is not improved too much as the output quantization increases, provided the alphabet of the channel state stream stays constant. However, for a large signal-to-noise ratio, the performance of the quaternary bootstrap scheme with a quaternary state stream is better than that achievable for an octal bootstrap scheme with a binary state stream! In general, the improvement obtainable from an increase in the state stream alphabet increases with the SNR.

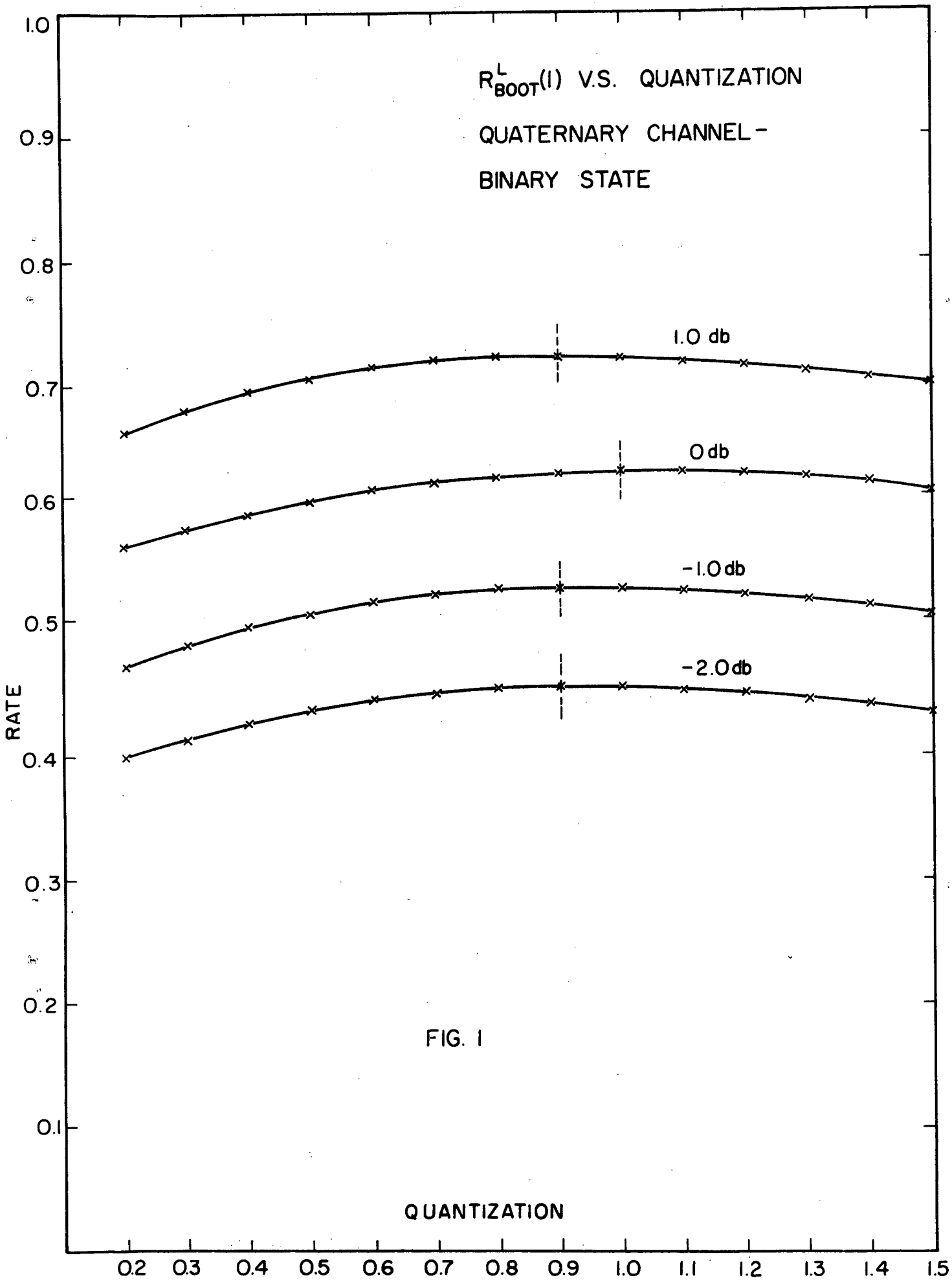

$R^{L}_{BOOT}(I)$ V.S. QUANTIZATION
QUATERNARY CHANNEL-
BINARY STATE
1.0 db
0 db
-1.0 db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5

FIG. 1

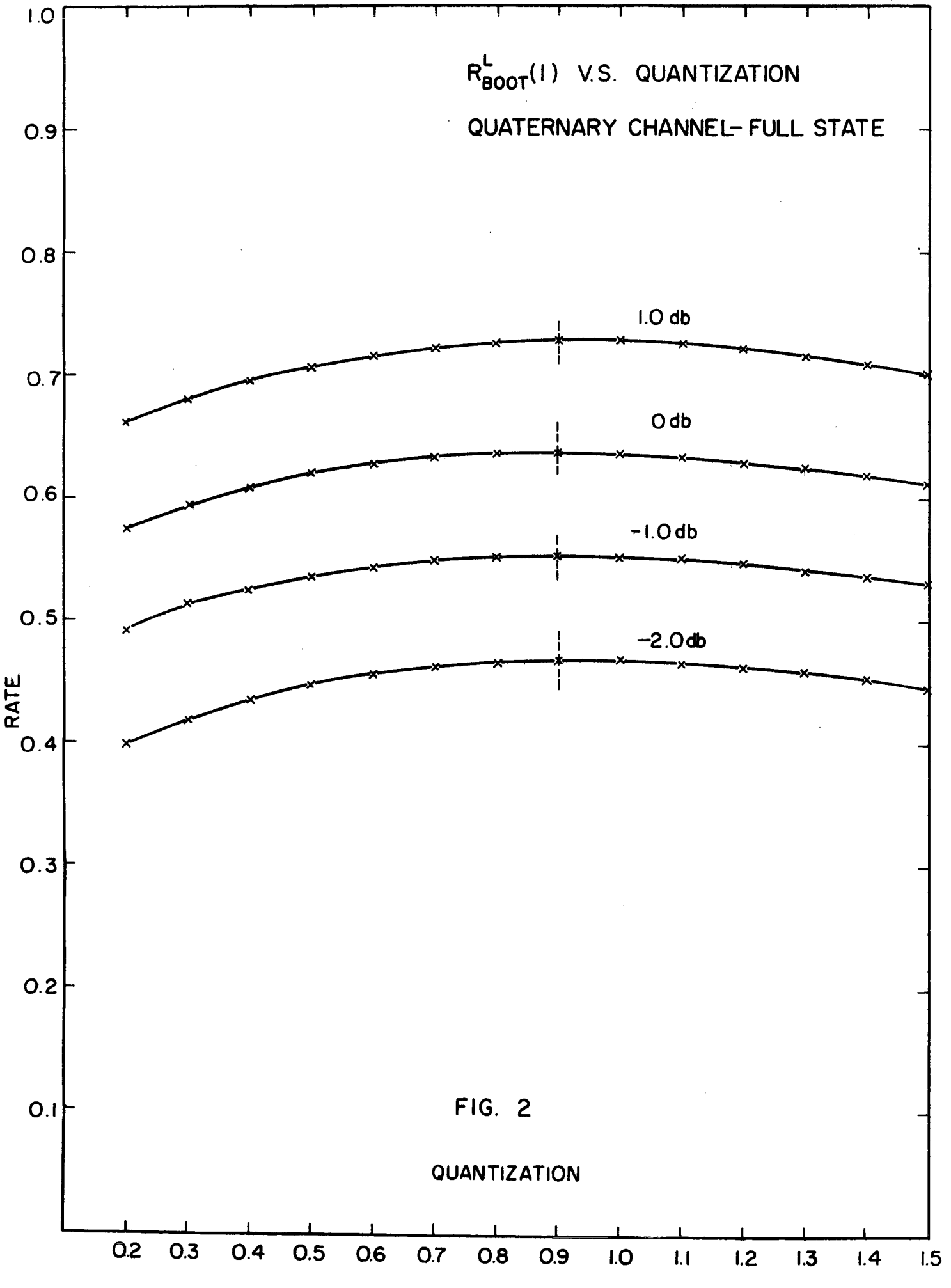

$R^{L}_{BOOT}(1)$ V.S. QUANTIZATION
QUATERNARY CHANNEL- FULL STATE
1.0 db
0 db
-1.0 db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

FIG. 2

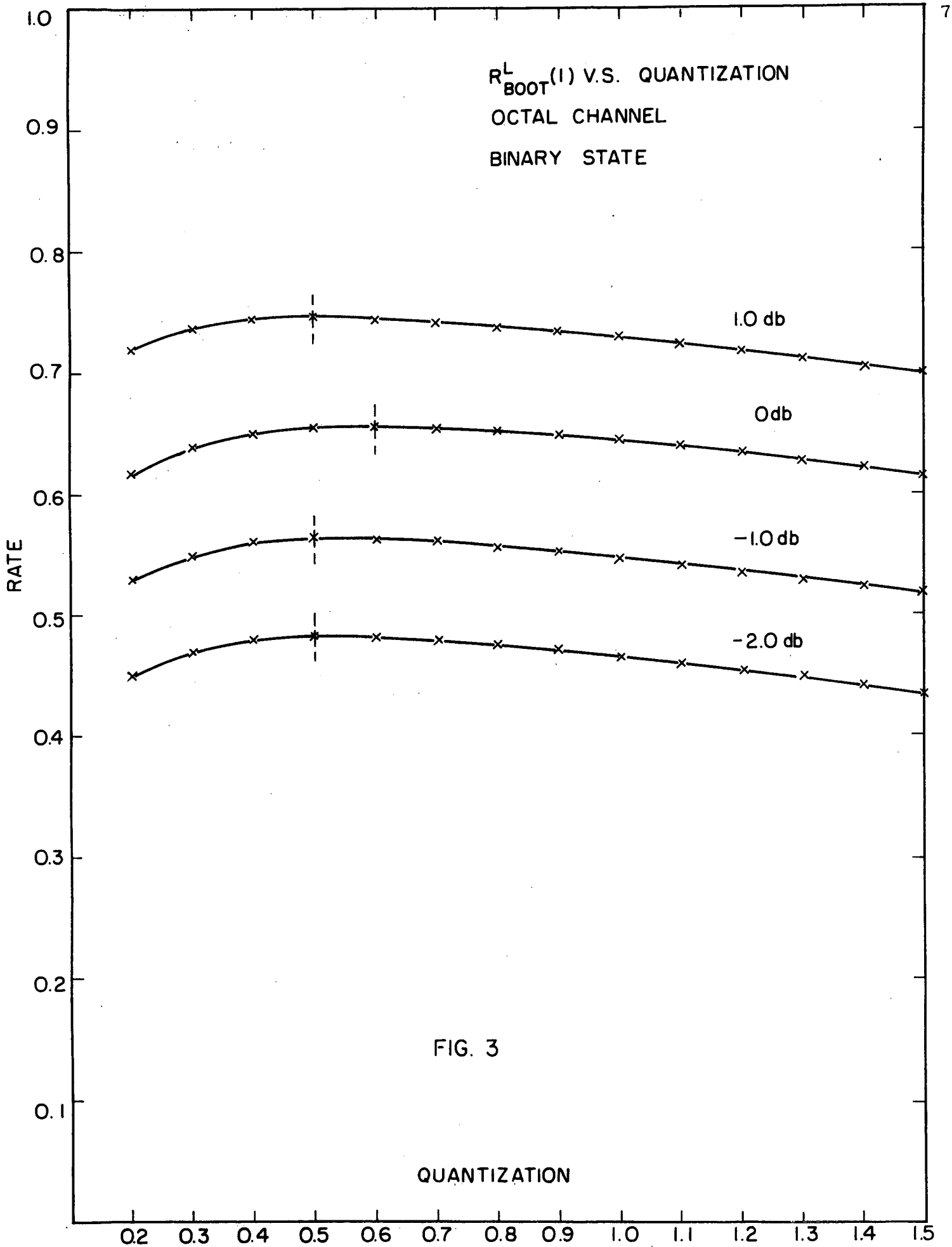

$R^{L}_{BOOT}(1)$ V.S. QUANTIZATION
OCTAL CHANNEL
BINARY STATE
1.0 db
0db
−1.0 db
−2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

FIG. 3

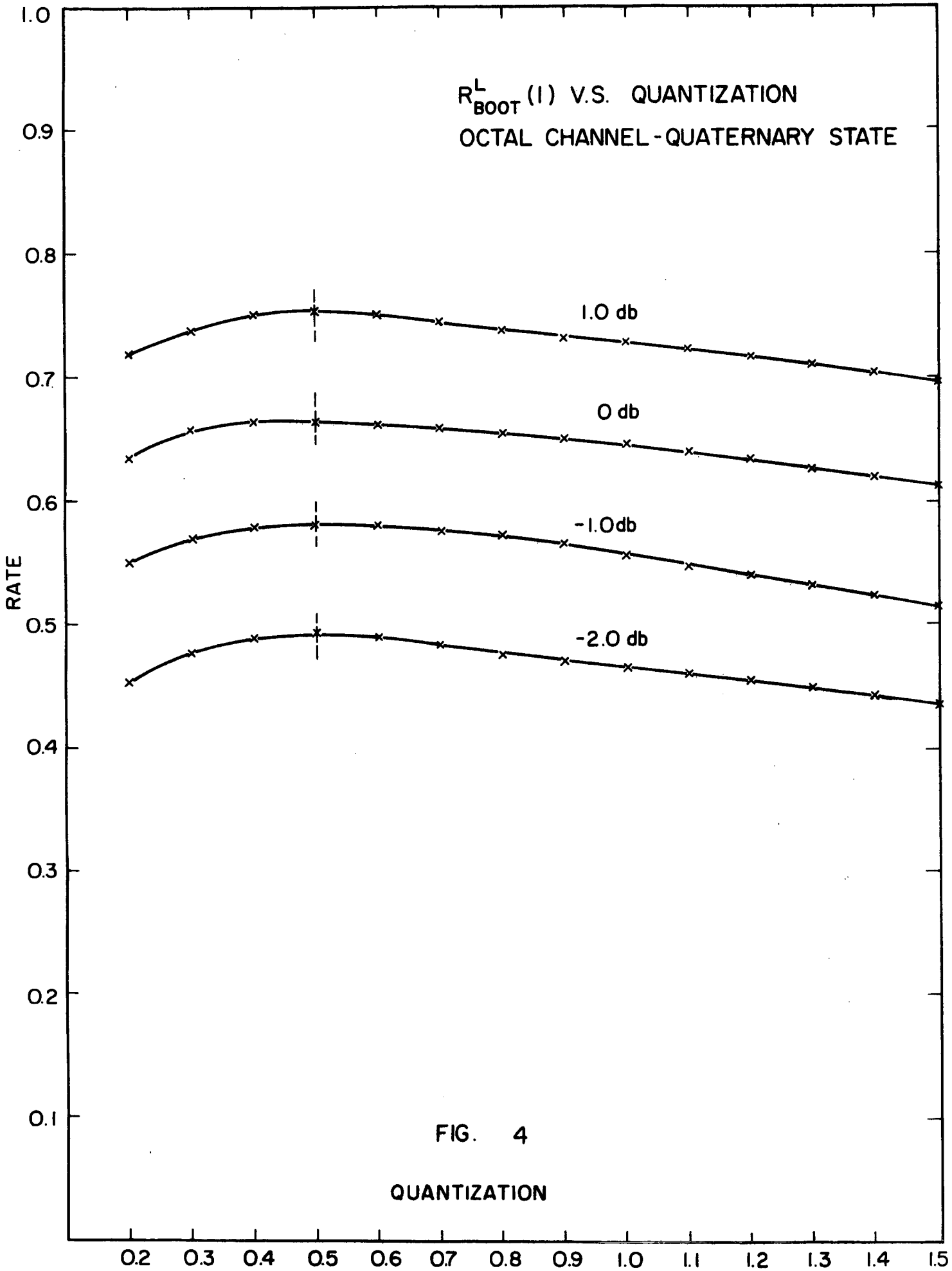

R^{L}_{BOOT} (1) V.S. QUANTIZATION
OCTAL CHANNEL - QUATERNARY STATE
1.0 db
0 db
-1.0db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5

FIG. 4

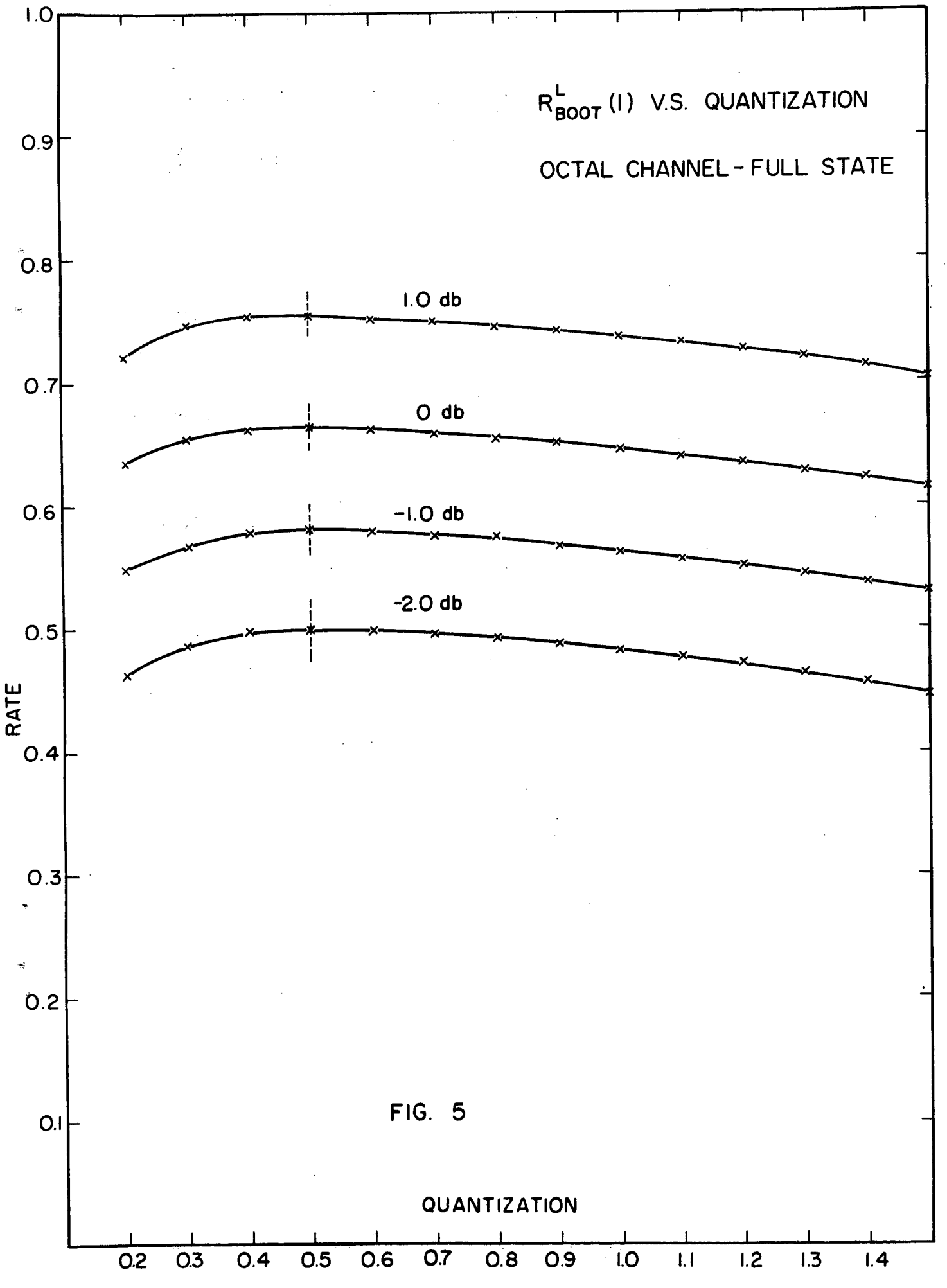

R^{L}_{BOOT} (1) V.S. QUANTIZATION
OCTAL CHANNEL - FULL STATE
1.0 db
0 db
-1.0 db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3
1.4

FIG. 5

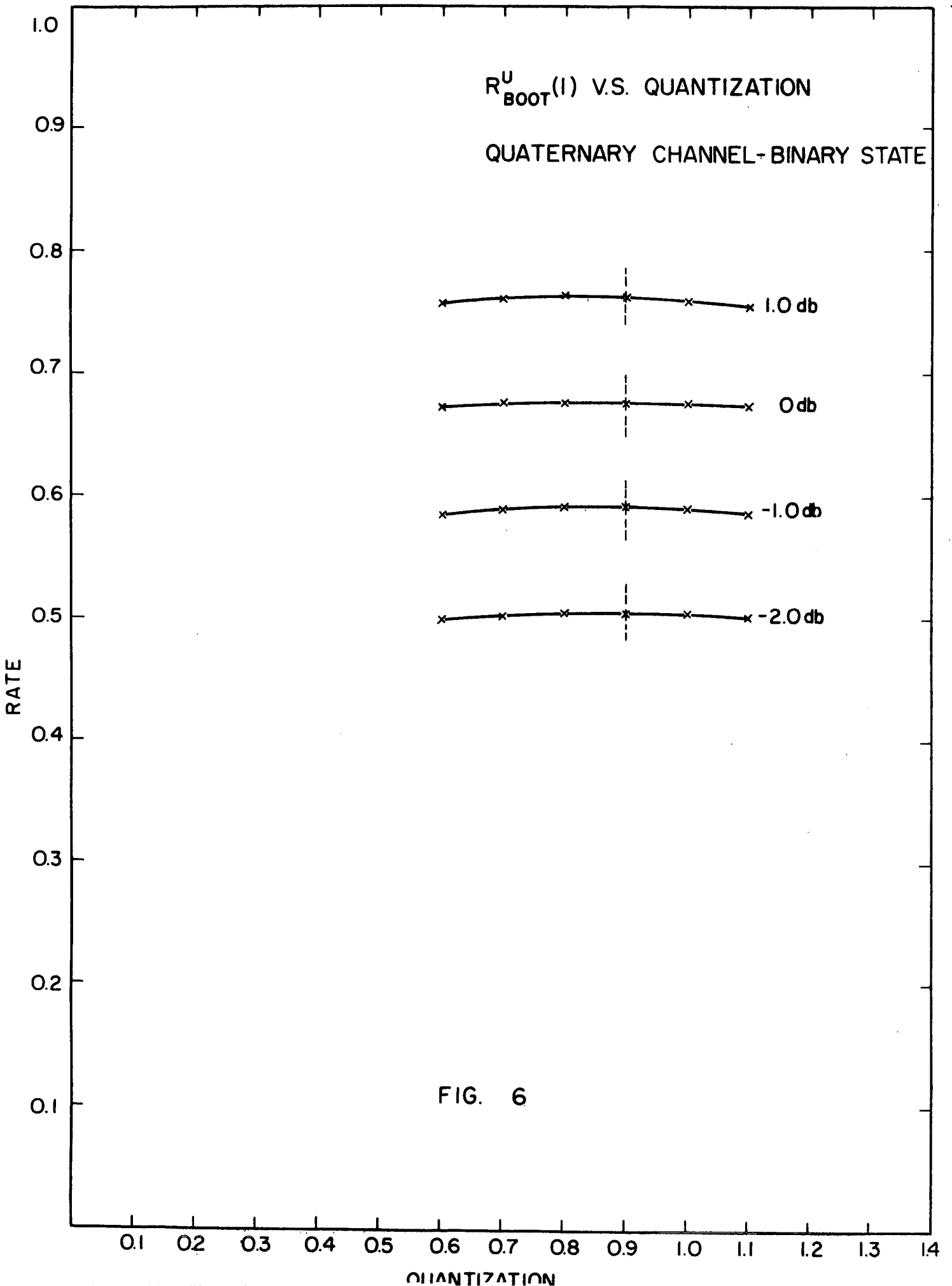

$R^{U}_{BOOT}(I)$ V.S. QUANTIZATION
QUATERNARY CHANNEL-BINARY STATE
1.0 db
0 db
-1.0 db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4

FIG. 6

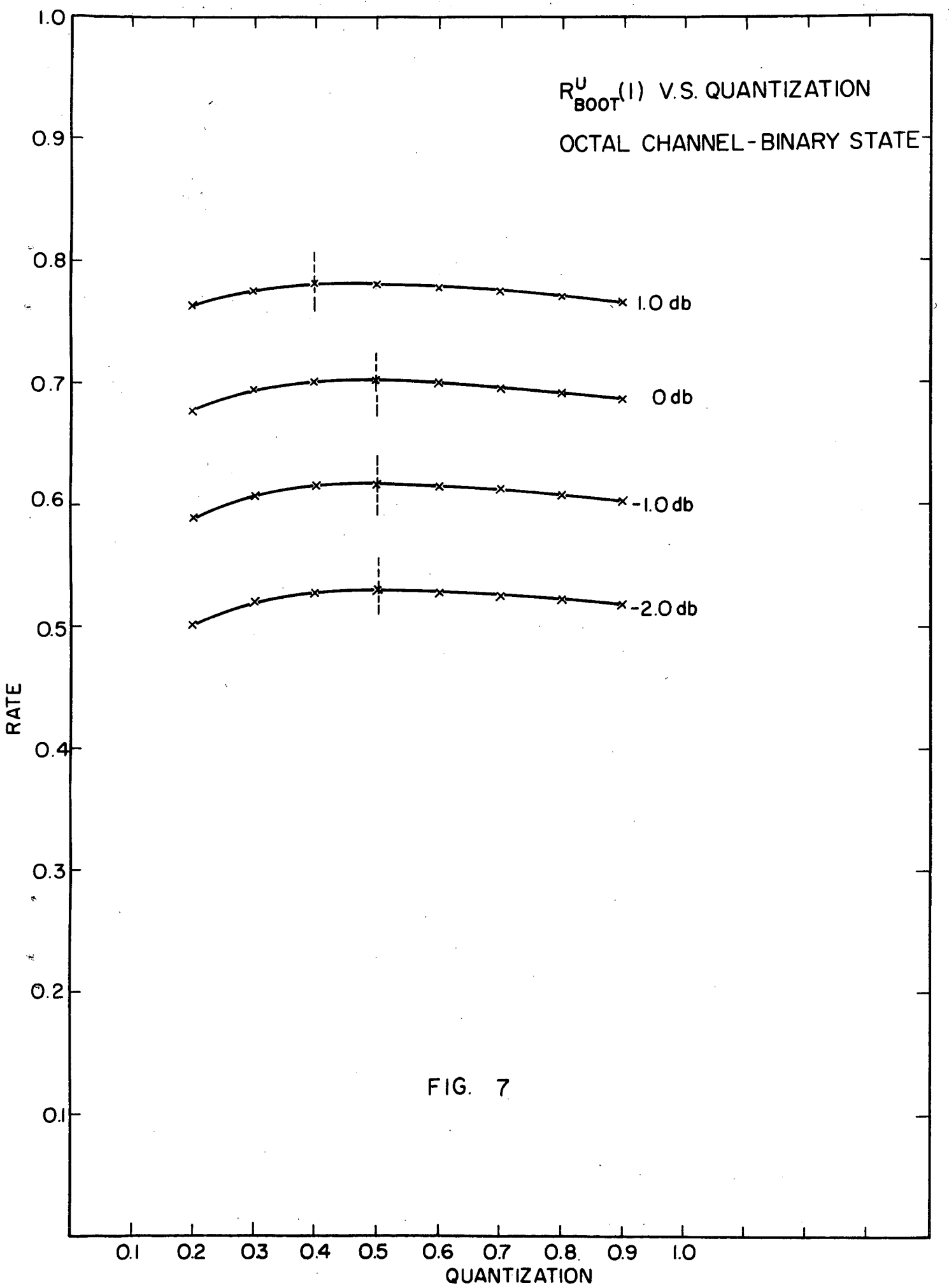

$R^{U}_{BOOT}(1)$ V.S. QUANTIZATION
OCTAL CHANNEL-BINARY STATE
1.0 db
0 db
-1.0 db
-2.0 db
RATE
QUANTIZATION
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

FIG. 7

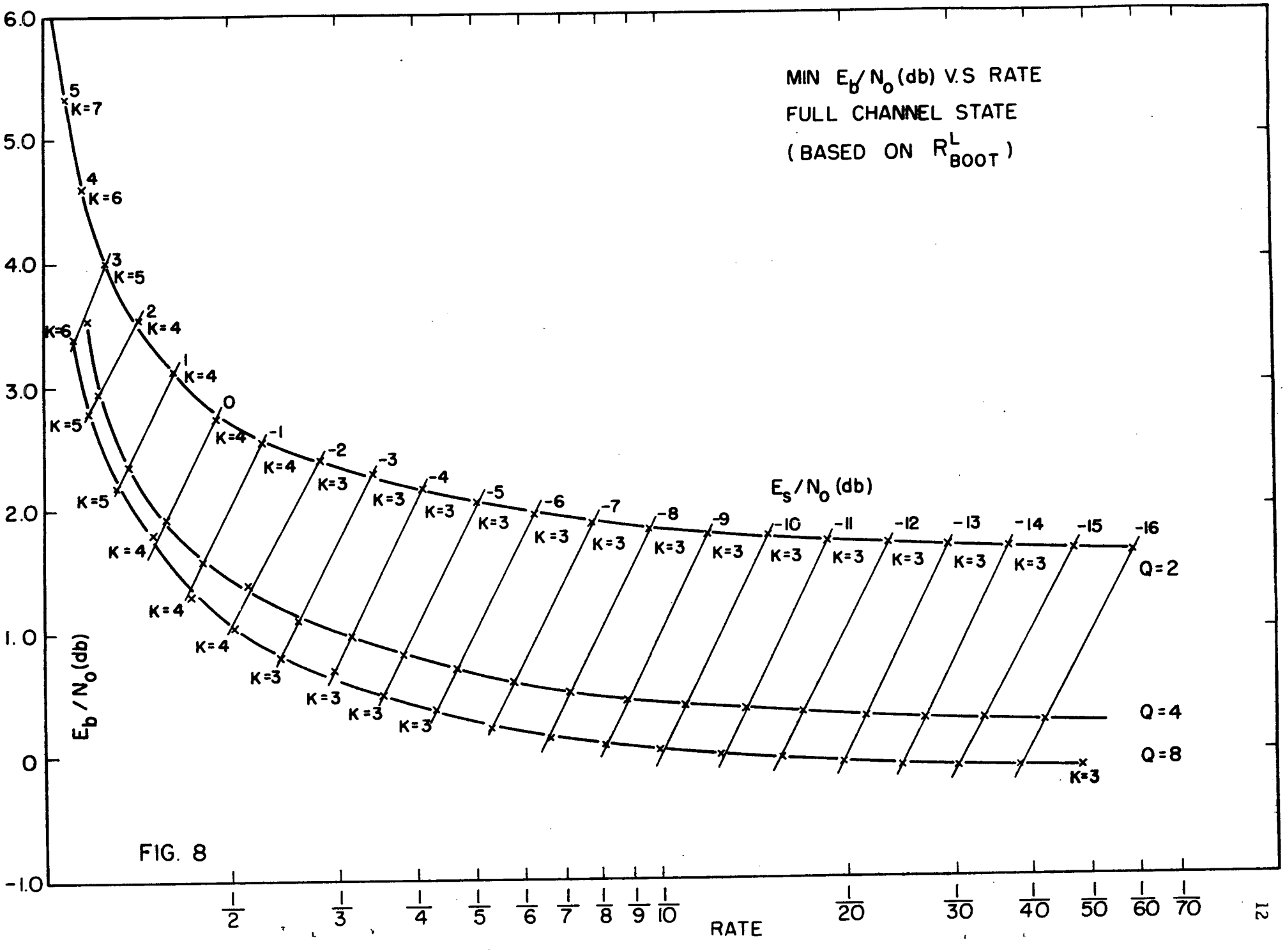

MIN E_b/N_o (db) V.S RATE
FULL CHANNEL STATE
(BASED ON R^L_{BOOT})
E_b/N_o (db)
E_s/N_o (db)
RATE
Q=2
Q=4
Q=8
6.0
5.0
4.0
3.0
2.0
1.0
0
-1.0
1/2
1/3
1/4
1/5
1/6
1/7
1/8
1/9
1/10
1/20
1/30
1/40
1/50
1/60
1/70
5 K=7
4 K=6
3 K=5
2 K=4
1 K=4
0 K=4
-1 K=4
-2 K=3
-3 K=3
-4 K=3
-5 K=3
-6 K=3
-7 K=3
-8 K=3
-9 K=3
-10 K=3
-11 K=3
-12 K=3
-13 K=3
-14 K=3
-15
-16
K=6
K=5
K=5
K=4
K=4
K=4
K=3
K=3
K=3
K=3
K=3
K=3

FIG. 8

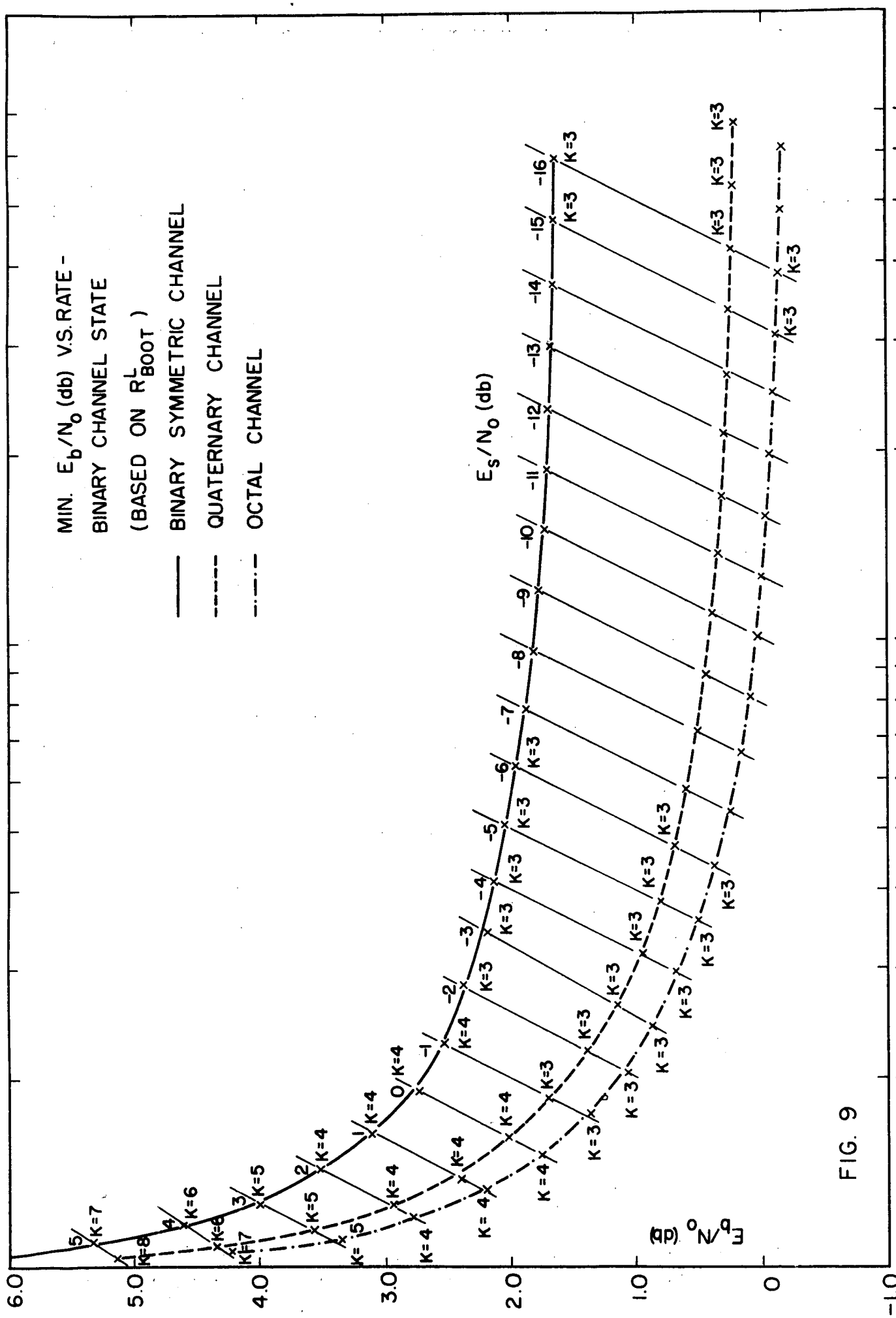

MIN. E_b/N_o (db) V.S.RATE -
BINARY CHANNEL STATE
(BASED ON R^L_{BOOT})
BINARY SYMMETRIC CHANNEL
QUATERNARY CHANNEL
OCTAL CHANNEL
E_s/N_o (db)
E_b/N_o (db)
RATE
6.0
5.0
4.0
3.0
2.0
1.0
0
-1.0
1/2
1/3
1/4
1/5
1/6
1/7
1/8
1/9
1/10
1/20
1/30
1/40
1/50
1/60
1/70

FIG. 9

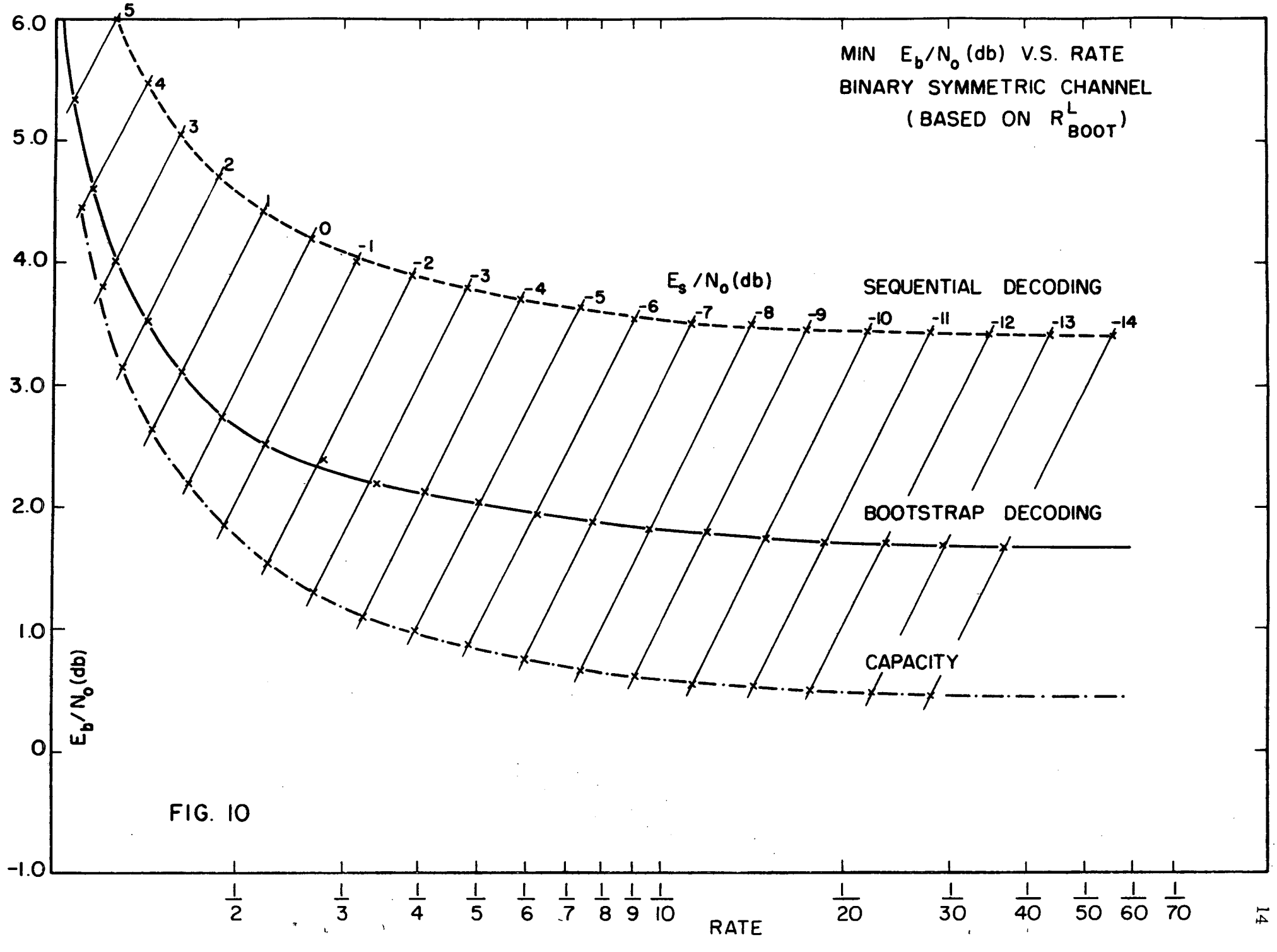

MIN E_b/N_0 (db) V.S. RATE
BINARY SYMMETRIC CHANNEL
(BASED ON R^L_{BOOT})
E_s/N_0 (db)
SEQUENTIAL DECODING
BOOTSTRAP DECODING
CAPACITY
E_b/N_0 (db)
RATE
6.0
5.0
4.0
3.0
2.0
1.0
0
-1.0
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
1/2
1/3
1/4
1/5
1/6
1/7
1/8
1/9
1/10
1/20
1/30
1/40
1/50
1/60
1/70

FIG. 10

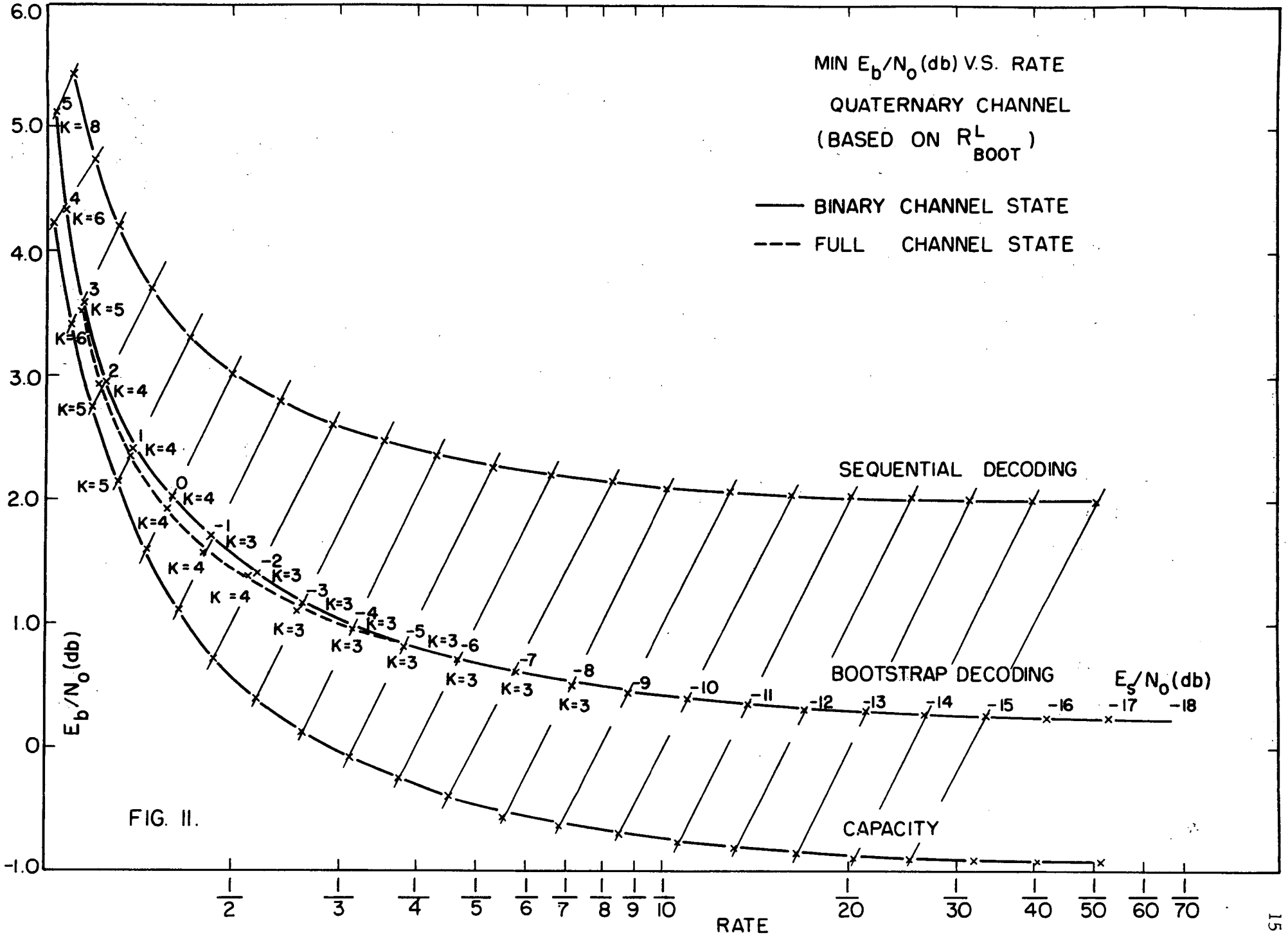

MIN E_b/N_0 (db) V.S. RATE
QUATERNARY CHANNEL
(BASED ON R^L_{BOOT})
BINARY CHANNEL STATE
FULL CHANNEL STATE
SEQUENTIAL DECODING
BOOTSTRAP DECODING
CAPACITY
E_b/N_0 (db)
E_s/N_0 (db)
RATE
6.0
5.0
4.0
3.0
2.0
1.0
0
-1.0
1/2
1/3
1/4
1/5
1/6
1/7
1/8
1/9
1/10
1/20
1/30
1/40
1/50
1/60
1/70
5 K=8
4 K=6
3 K=5
K=6
2 K=4
K=5
1 K=4
K=5
0 K=4
K=4
-1 K=3
K=4
-2 K=3
K=4
-3 K=3
K=3
-4 K=3
K=3
-5 K=3
K=3
-6 K=3
K=3
-7 K=3
K=3
-8 K=3
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18

FIG. 11.

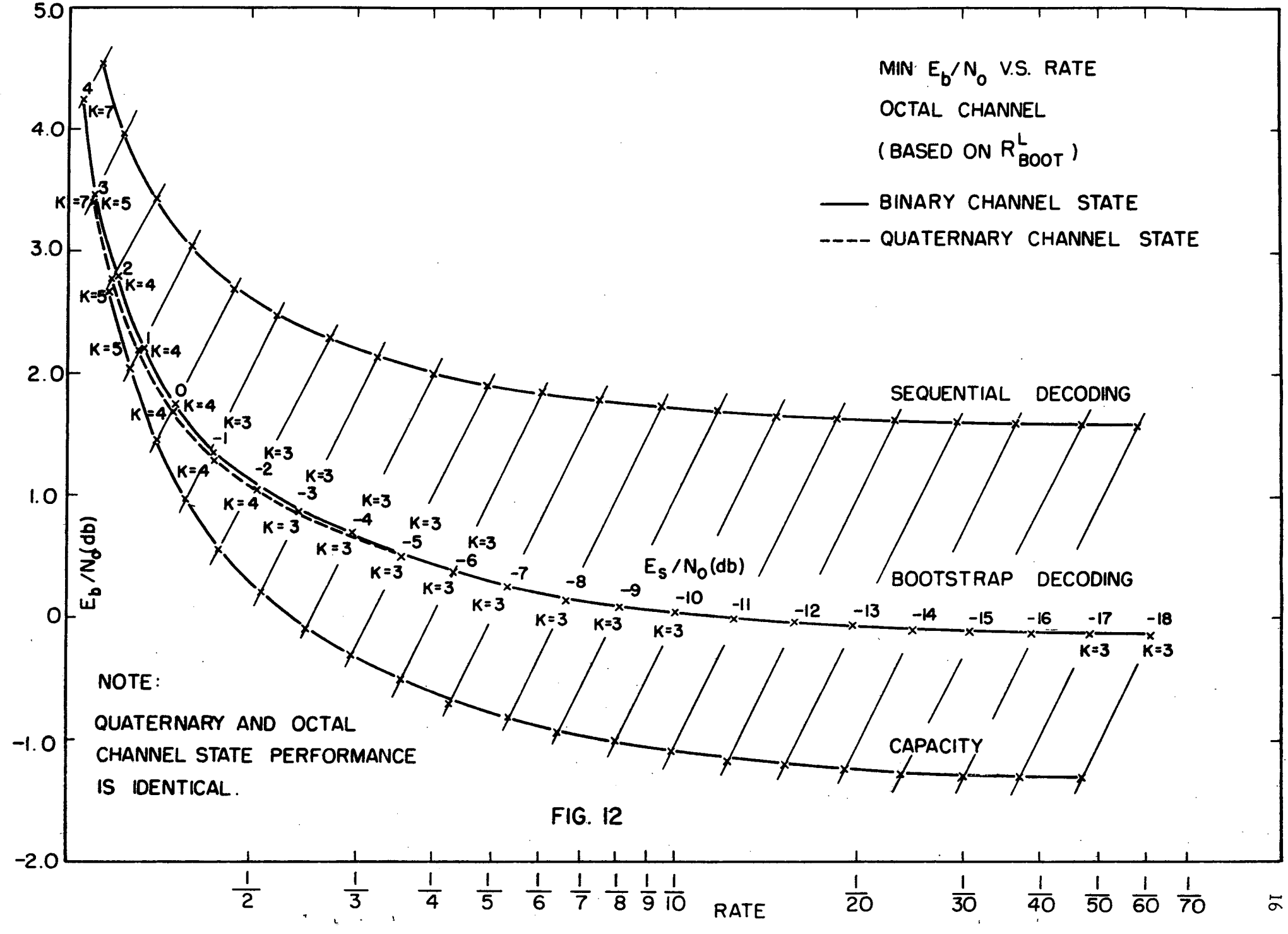

MIN E_b/N_o V.S. RATE
OCTAL CHANNEL
(BASED ON R^L_{BOOT})
BINARY CHANNEL STATE
QUATERNARY CHANNEL STATE
SEQUENTIAL DECODING
BOOTSTRAP DECODING
CAPACITY
E_b/N_o(db)
E_s/N_o(db)
RATE
NOTE:
QUATERNARY AND OCTAL
CHANNEL STATE PERFORMANCE
IS IDENTICAL.

FIG. 12

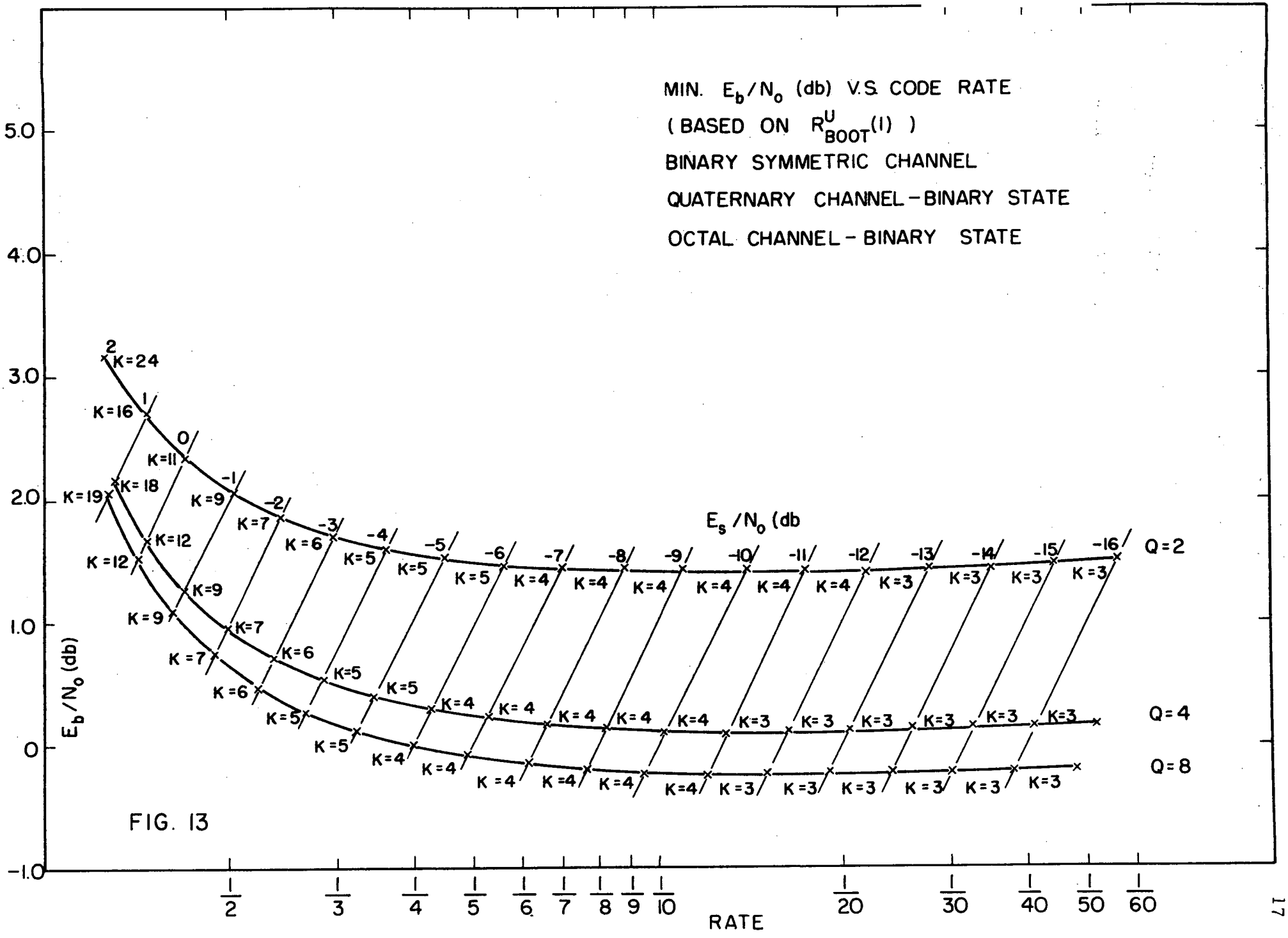

MIN. E_b/N_o (db) V.S. CODE RATE
(BASED ON $R^U_{BOOT}(1)$)
BINARY SYMMETRIC CHANNEL
QUATERNARY CHANNEL - BINARY STATE
OCTAL CHANNEL - BINARY STATE
E_b/N_o (db)
E_s/N_o (db
RATE
Q=2
Q=4
Q=8
5.0
4.0
3.0
2.0
1.0
0
-1.0
1/2
1/3
1/4
1/5
1/6
1/7
1/8
1/9
1/10
1/20
1/30
1/40
1/50
1/60

FIG. 13

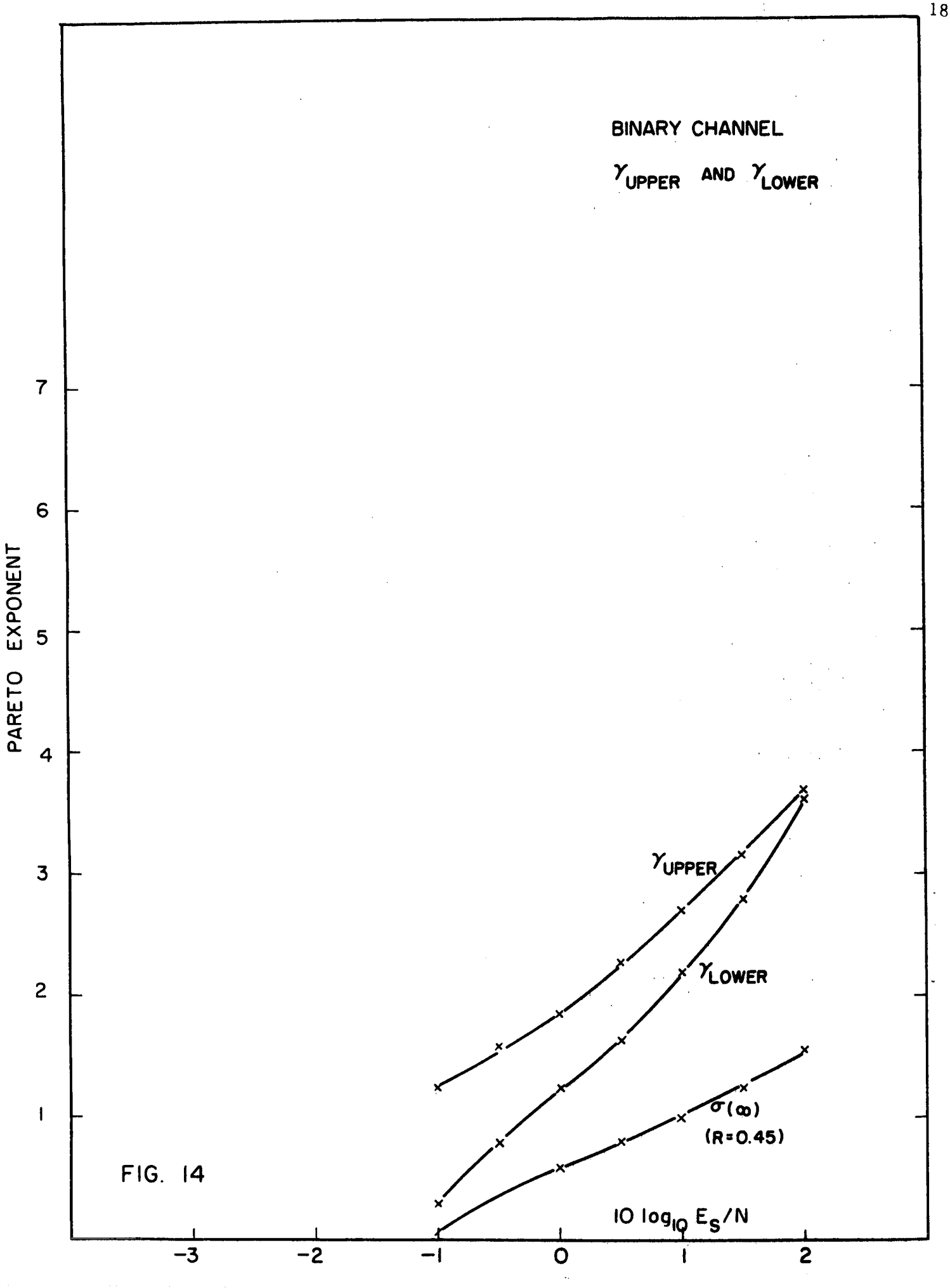

BINARY CHANNEL
γ_{UPPER} AND γ_{LOWER}
PARETO EXPONENT
7
6
5
4
3
2
1
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$
(R=0.45)
10 $\log_{10}$ E_S/N
-3
-2
-1
0
1
2

FIG. 14

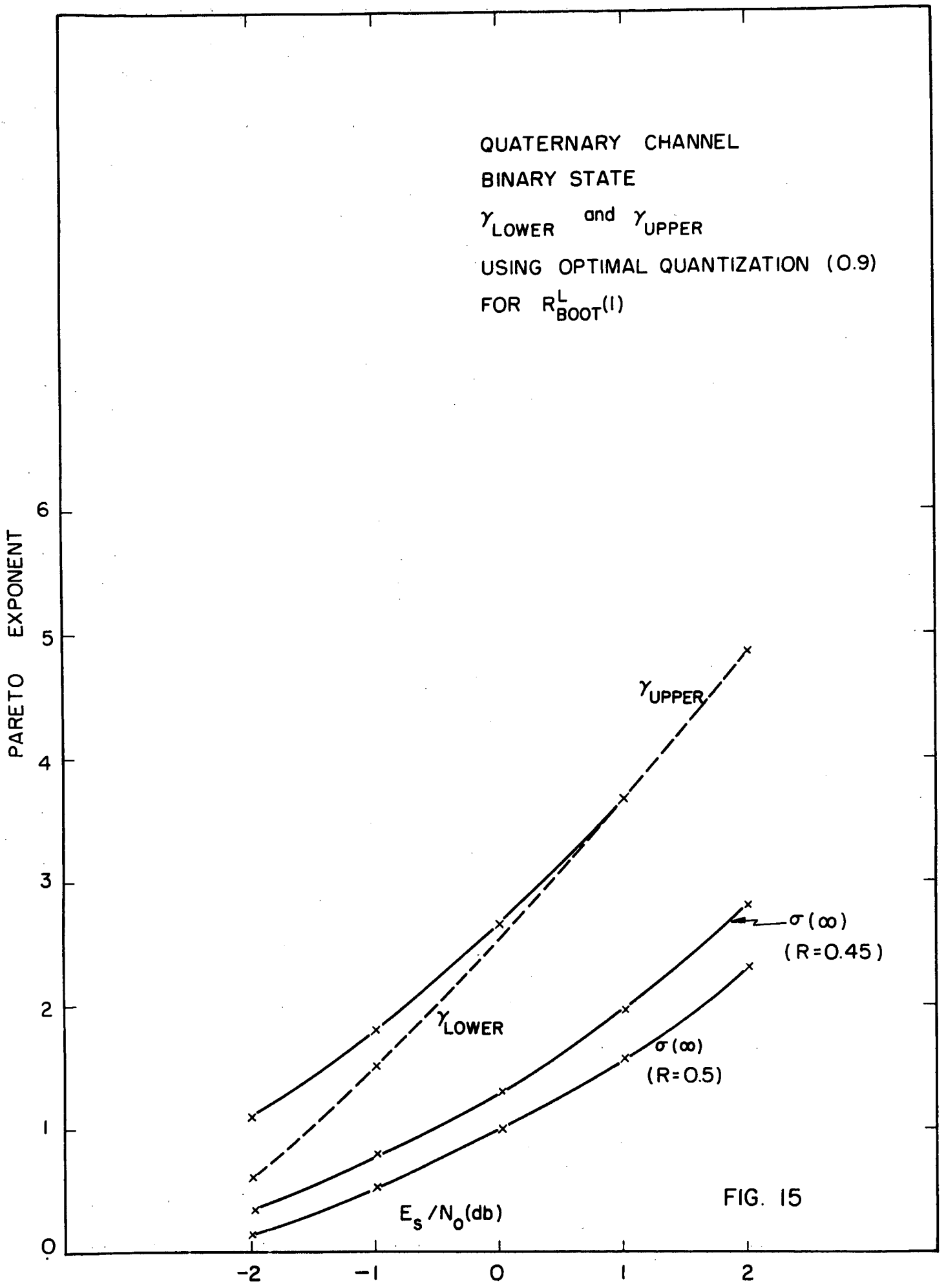

QUATERNARY CHANNEL
BINARY STATE
γ_{LOWER} and γ_{UPPER}
USING OPTIMAL QUANTIZATION (0.9)
FOR $R^{L}_{BOOT}(1)$
PARETO EXPONENT
0
1
2
3
4
5
6
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$
(R=0.45)
$\sigma(\infty)$
(R=0.5)
E_s/N_o(db)
-2
-1
0
1
2

FIG. 15

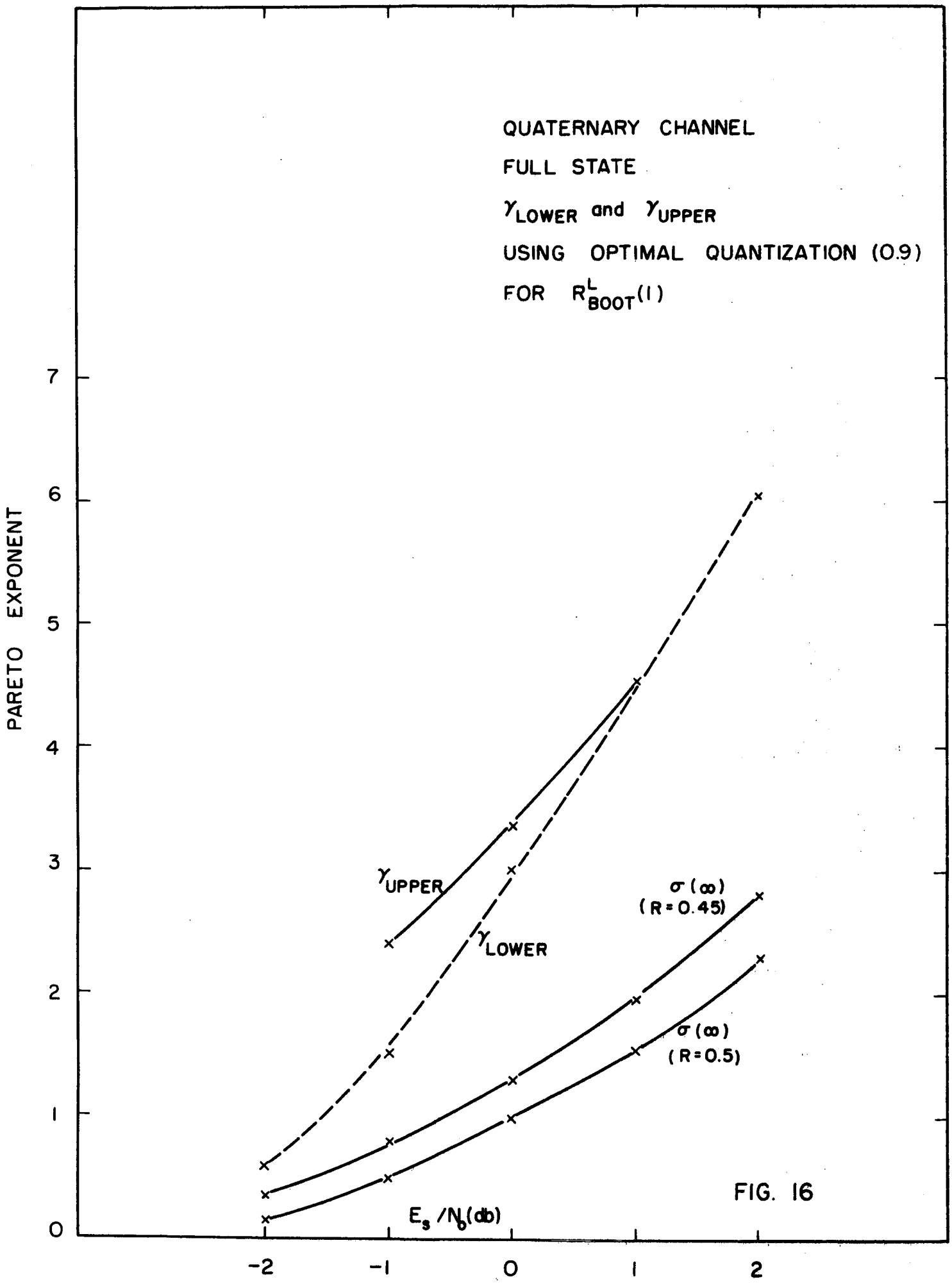

QUATERNARY CHANNEL
FULL STATE
γ_{LOWER} and γ_{UPPER}
USING OPTIMAL QUANTIZATION (0.9)
FOR $R^{L}_{BOOT}(1)$
PARETO EXPONENT
7
6
5
4
3
2
1
0
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$ (R = 0.45)
$\sigma(\infty)$ (R = 0.5)
E_s/N_0(db)
-2
-1
0
1
2

FIG. 16

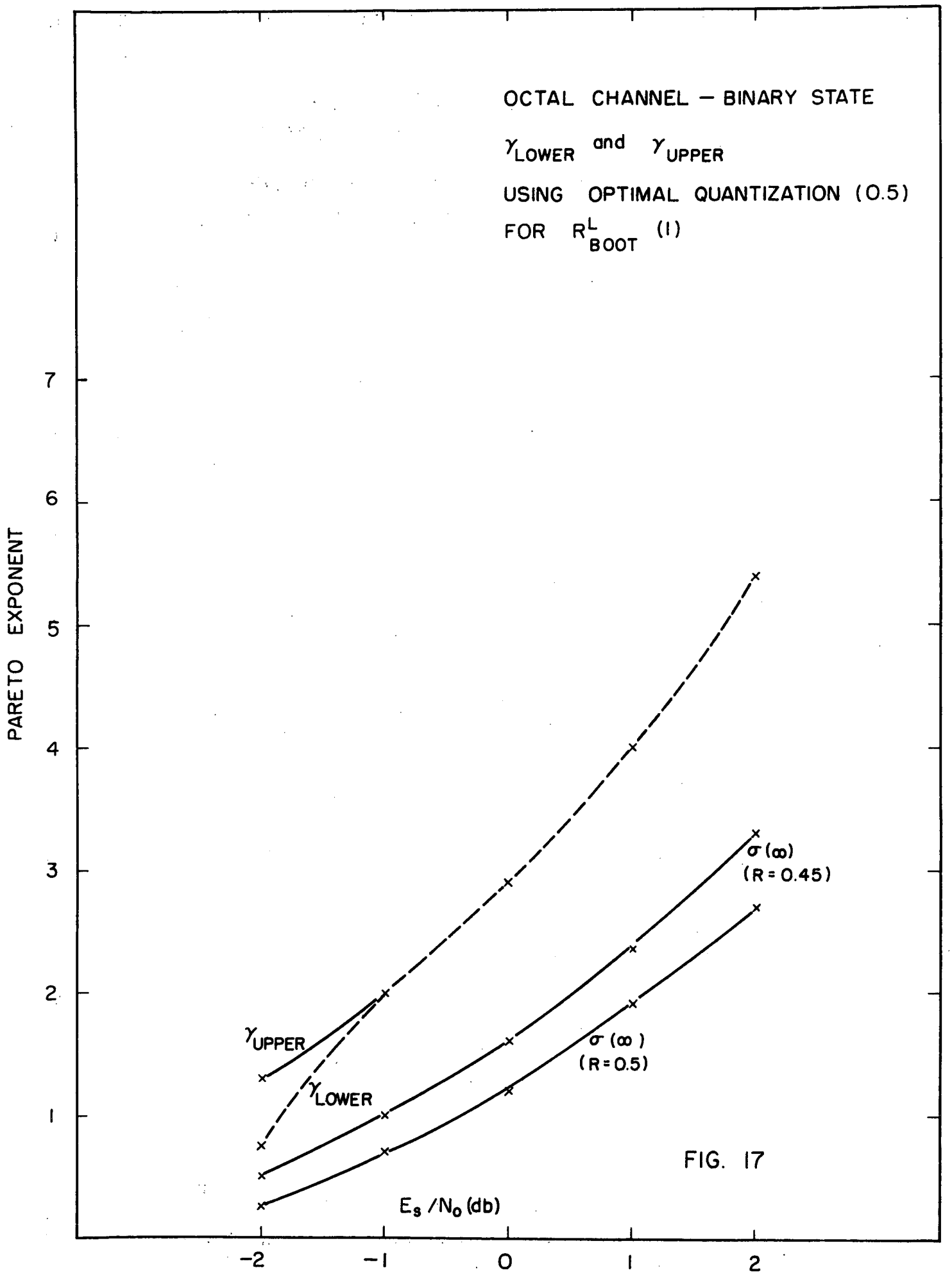

OCTAL CHANNEL – BINARY STATE
γ_{LOWER} and γ_{UPPER}
USING OPTIMAL QUANTIZATION (0.5)
FOR R^{L}_{BOOT} (1)
PARETO EXPONENT
7
6
5
4
3
2
1
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$
(R = 0.45)
$\sigma(\infty)$
(R = 0.5)
E_s/N_0 (db)
-2
-1
0
1
2

FIG. 17

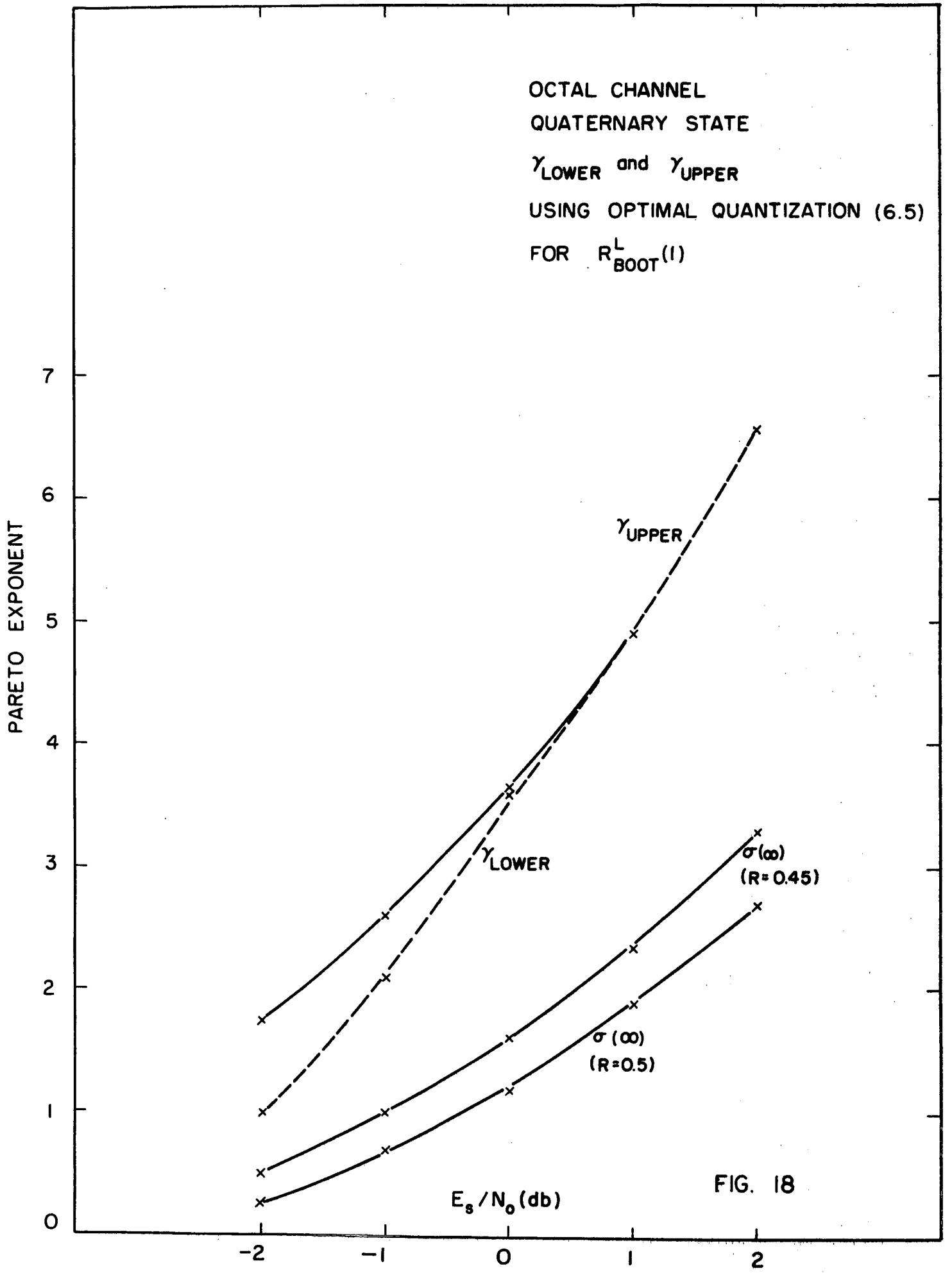

OCTAL CHANNEL
QUATERNARY STATE
γ_{LOWER} and γ_{UPPER}
USING OPTIMAL QUANTIZATION (6.5)
FOR $R^{L}_{BOOT}(1)$
PARETO EXPONENT
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$
(R=0.45)
$\sigma(\infty)$
(R=0.5)
E_s/N_0 (db)
0
1
2
3
4
5
6
7
-2
-1
0
1
2

FIG. 18

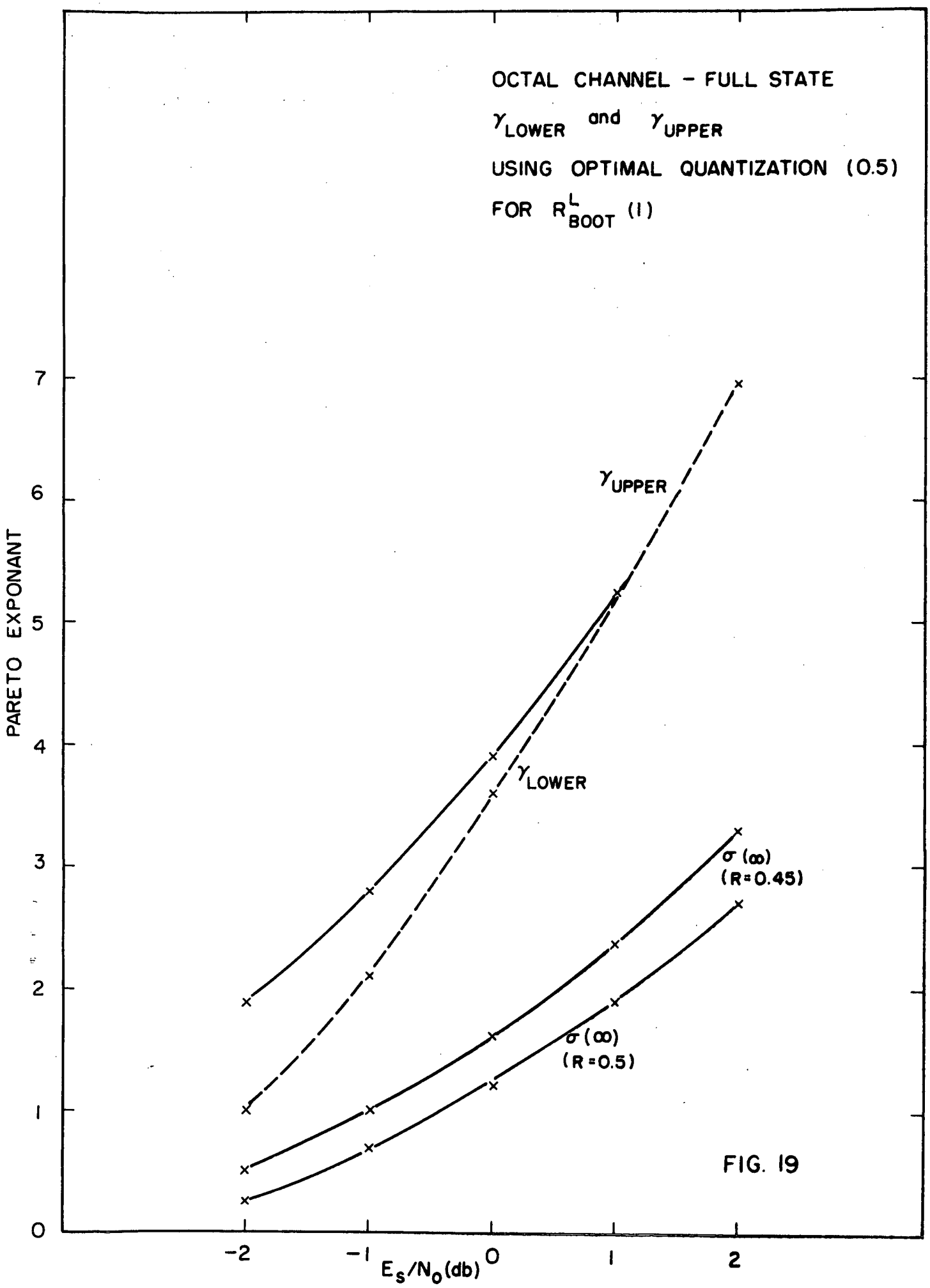

OCTAL CHANNEL - FULL STATE
γ_{LOWER} and γ_{UPPER}
USING OPTIMAL QUANTIZATION (0.5)
FOR R^{L}_{BOOT} (1)
PARETO EXPONANT
γ_{UPPER}
γ_{LOWER}
$\sigma(\infty)$ (R=0.45)
$\sigma(\infty)$ (R=0.5)
E_s/N_0(db)
0
1
2
3
4
5
6
7
-2
-1
0
1
2

FIG. 19

II-B. Development of Programs for Simulation of Bootstrap Sequential Decoding

1. While inspecting some simulation results, we noticed that with the systematic convolutional code being used, the bootstrap decoder commits a considerable number of decoding errors. We have therefore adjusted both the rudimentary and the pull-up decoders so that they insert these errors into the state stream and continue decoding (instead of stopping as previously). Selective simulations suggested the following conclusions:

The errors committed by the rudimentary scheme occur mostly in the tails (hence longer tail length than 25 seems definitely indicated). When these are inserted into the state stream, the decoder is able to finish the entire block at the price of inserting into some decoded stream those errors that are forced by the parity relationship.

The pull-up scheme works at a lower SNR. When the errors committed on stream J are inserted into the state stream, the parity forces them into some stream K. The decoder is then capable of decoding all but the J^{th} and K^{th} streams, and is not able to continue the decoding of the latter within a reasonable number of steps. This suggests that the errors can again be eliminated at the end simply by re-decoding the J^{th} and K^{th} streams from their beginning. The simulation results reported in the next section are based on programs that incorporate the above changes.

2. Fortran versions of the rudimentary and pull-up bootstrap hybrid decoders based on the Fano algorithm were debugged. Simulation indicates that these decoders examine on the average four times the number of nodes examined by the corresponding stack-based algorithm. The results reported in the next section are based on Fano sequential decoders.

3. A bootstrap algorithm was constructed that is suitable for decoding of channels with binary inputs and quaternary or octal outputs. These channels arise from optimal equal level quantization of Gaussian additive noise channels. The program has a preamble that computes the channel transition probabilities corresponding to that quantization, as a function of a supplied SNR in dB. The bootstrapping algorithm utilizes a binary channel state stream. The next section reports simulation results based on this program.

4. A generalization of our original bootstrap algorithm was constructed that is suitable for decoding of channels with binary inputs and quarternary or octal outputs. The algorithm has a full output alphabet state stream. The program has a preamble that computes the channel transition probabilities corresponding to optimal uniform 4 to 8 level output quantitization of a Gaussian additive noise channel as a function of a supplied SNR value in dB. Theoretical curves of Section II-A indicate that the dB gain arising from this refinement will be only a moderate one. Nevertheless, a strategy employing the refinement in case the binary state stream algorithm runs into trouble might be well worth considering.

5. An algorithm was de-bugged which uses a three-group algebraic outer code with a convolutional inner code. The operation of the algebraic part of the algorithm is described in Section II-F.

II-C. Simulated Performance of Bootstrap Sequential Decoding

L. B. Hofman used the various algorithmic techniques developed under this contract to construct programs simulating the performance of the Bootstrap Sequential Decoding Algorithm. He summarized his results in the paper "Performance Results for a Hybrid Coding System" that he presented at the 1971 International Telemetering Conference. This work is reproduced below:

Summary.– Computer simulation studies of the hybrid pull-up bootstrap decoding algorithm have been conducted using a constraint length 24, nonsystematic, rate 1/2 convolutional code for the symmetric channel with both binary and 8-level quantized outputs. Computational performance was used to measure the effect of several decoder parameters and determine practical operating constraints. Results reveal that the track length may be reduced to 500 information bits with small degradation in performance. The optimum number of tracks per block was found to be in the range of 7 to 11. An effective technique was devised to efficiently allocate computational effort and identify reliably decoded data sections. Long simulations indicate that a practical bootstrap decoding configuration has a computational performance about 1.0 dB better than sequential decoding and an output bit error rate about 2.5×10^{-6} near the R_{comp} point.

Introduction.– The basic coding dilemma is one of exponentially increasing decoding complexity as the theoretical capacity of a communications channel is approached. Hybrid coding is a cascade or concatenation of block and/or convolutional codes in an attempt to operate near capacity while maintaining a complexity less than that possible with either code type alone. This paper presents the results of a study of the hybrid bootstrap coding system of Jelinek.[1] This technique is similar to a simple case of the Falconer scheme[2] in that a parity relationship between a set of convolutionally encoded data tracks is used to aid in the decoding of those portions that are difficult. (An even parity is assumed throughout.) It differs from the Falconer scheme, which uses an algebraic relationship to derive directly the most difficult portions after a sufficient number of others are decoded, by making use of additional probabilistic information contained in the parity relationship. In so doing, each bit of data decoded helps to "bootstrap" those remaining.

After reviewing briefly the functioning of bootstrap decoding, this paper examines the computational effect of several decoder parameters and determines a practical range of operating values. Detailed performance behavior of such an optimized system is presented and compared to simple sequential decoding and Falconer decoding.

Encoding.– The encoding function is the same for all variations of bootstrap decoding described in this paper. (The decoders differ only in the manner in which they utilize information that is always available at the receiver.) Basically, m–1 independent, convolutionally encoded "data tracks" are linked together into one "decoding block" by the addition of an m-th "parity track." That is, each bit of the parity track is the modulo-two sum of the corresponding bits in the data tracks. Because of the linearity of convolutional codes, this parity track is also decodable and, as will be shown, may actually be generated by a convolutional encoder. The reader will note that this encoding function is identical to that required by the Falconer system for a simple parity check code.

Actual mechanization of the encoder depends upon several operational considerations. One method, which requires m–1 convolutional encoders, provides natural interleaving of the tracks. Data are routed to the encoders for coding and transmission in a "round robin" fashion, with the parity bit inserted in its turn by a modulo-two adder. Decoder synchronization for such a scheme will be difficult; synchronization and tail-forcing bits must be independent of data

formating, possibly causing a small data buffering problem at the end of each block. Failure of the decoder to complete the decoding of a block results in the loss of a large amount of data, if not the entire block.

An alternative way of mechanizing the encoder requires one convolutional encoder and a storage register having the length of a track. The data are encoded and transmitted, one track at a time, while the parity track is formed in the storage register. The contents of the storage register are then encoded, transmitted, and reset following the last data track of each block. Although this scheme does not provide interleaving and causes an even larger buffering problem while the parity track is transmitted, it does offer several advantages. It is possible to let data formating correspond to individual data tracks. Code synchronization can be performed easily on a track basis, with block synchronization derived from identification bits embedded in the data tracks. In addition, a decoder failure will not necessarily result in the loss of a full block of data. Finally, since data formating, synchronization, and tail-forcing can be related, the rate loss for these functions can be reduced.

Rudimentary Bootstrap Decoding.– Bootstrap decoding is applicable to all symmetrical binary input channels. For the purposes of this paper, a simplified description of the "rudimentary" algorithm for the binary symmetric channel (BSC) is given following the outline used by Jelinek.[1]

After the encoded data have been received and are synchronized, the bits of a block are grouped into m tracks, and an additional track, the "channel state stream," is formed by the decoder. Each channel state stream bit is the modulo-two sum of corresponding bits in the parity and data tracks. The channel state stream differs from the parity track because it includes the parity track and is formed after the transmitted sequence is corrupted by noise. Therefore, a "zero" in this track indicates that an even number of errors was received at a given position, and a "one" indicates an odd number.

The probabilities that k bits which are independently transmitted through a BSC of crossover probability p will be received with an even or odd number of errors are given by

$$q_k(0) = [1 + (1 - 2p)^k]/2$$
$$q_k(1) = [1 - (1 - 2p)^k]/2$$

The information is used to form an augmented transition probability matrix $w_m(y,z/x)$ where y is the received bit and z is the channel state bit associated with y and formed over m tracks, given that x was transmitted. Thus:

$$w_m(0,0/0) = w_m(1,0/1) = (1 - p)q_{m-1}(0)$$
$$w_m(0,1/0) = w_m(1,1/1) = (1 - p)q_{m-1}(1)$$
$$w_m(1,0/0) = w_m(0,0/1) = p\ q_{m-1}(1)$$
$$w_m(1,1/0) = w_m(0,1/1) = p\ q_{m-1}(0)$$

It is natural to use these augmented transition probabilities in forming the bit likelihood function for sequential decoding. The function is

$$\lambda_m \triangleq \log_2 [w_m(y,z/x)/w_m(y,z)] - R$$

where

$$w_m(y,z) = [w_m(y,z/0) + w_m(y,z/1)]/2 = [q_m(z)]/2$$

and R is the bias factor.

From this starting point, the development of the rudimentary bootstrap decoding algorithm follows directly. The first of the m tracks is sequentially decoded using the channel state stream and likelihood values defined above. If, after a preassigned amount of effort, decoding of this track is not completed, restart values are saved. This step is repeated on successive tracks,

looping back to the first track if necessary, until decoding of one is completed. At this time, the received sequence for the completely decoded track is replaced by the newly estimated sequence, and the channel state stream is recomputed. If the decoding was error free, then the new channel state stream values represent an even or odd number of errors in the m−1 remaining tracks, as before. The entire process is repeated, excluding the decoded track, now using likelihood values for m−1 tracks. When a second track is completed, its received sequence is replaced, and the channel state stream is again updated.

The pattern is now obvious, and the process is repeated until all tracks have been decoded or the total work exceeds a maximum amount. It would be possible to derive the last remaining track, on the basis of the parity relationship, when m−1 tracks have been decoded. Indeed, this is the principle of Falconer decoding; but it is actually simpler to decode this track, too, since the decoding requires exactly one computation per bit. This fact, and the general effect of using the channel state stream, may be seen in the sample likelihood table shown in figure 1. When many tracks are undecoded, the channel state bit gives little additional information about the probability of error in a single received bit. Therefore, for large k, the likelihood values for bootstrap decoding approach the usual values for sequential decoding, depending mainly upon agreement or disagreement between the received bit and the hypothesis. At the other extreme, for small k, the channel state bit has a large influence. For example, if two tracks remain undecoded (k = 2) and the channel state bit is "one," neither hypothesis is reliable because the probability is 0.5 that the received bit is in error. On the other hand, great reliance is placed on the correctness of the received bit when the channel state bit is "zero" since the probability of a double error is small. When k = 1, the knowledge that the received bit is in error for a channel state bit "one" and correct for a "zero" is reflected in the table by a $-\infty$ likelihood value for the impossible hypothesis and 1.0 for the correct hypothesis.

Pull-Up Algorithm.– The primary worth of the rudimentary algorithm is the description of the bootstrapping process and simplification of its analysis. Practical use of the rudimentary algorithm is probably limited because one rather simple modification substantially increases the power of the decoder. In the modified algorithm, called the "pull-up" algorithm, the decoder does not wait until a track is decoded completely before updating the channel state stream. It operates instead on a single track until the track is completed or a difficult-to-decode section is sensed, at which time decoding is stopped. The completed track is handled as in the rudimentary algorithm. Before proceeding with the next track after a track is terminated, however, the decoder declares that portion which it deems reliable to be "definitely decoded." In doing so, it updates the channel state stream and prepares restart values so that the next decoding attempt on the track will begin immediately after the definitely decoded section.

Since it is possible to have all tracks in varying stages of completion, to obtain the most effective use of the channel state stream it is necessary to indicate how many tracks remain undecoded at a given node. This is done with a vector, KLEFT, the length of a track, which the decoder references to determine the likelihood values to use at a given node. At the outset, all KLEFT values are set to m, the number of tracks in a block, and are adjusted accordingly as individual tracks are "pulled up." Note that it is necessary each time to start decoding from the "origin" because the state stream may change from the time the decoder terminates a track to the time the decoder restarts it.

Computer Simulations.– Many variables affect the performance and practicality of a system as complex as bootstrap decoding. Unfortunately, analysis can give only bounds on performance for simplified and idealized conditions. Therefore, simulations have been performed to determine the gross effect of a number of parameters for the pull-up version and to obtain performance figures for a quasi-optimized system that could be considered for possible deep space application.

The simulation program was written in FORTRAN for a 24-bit, 1.75 μs/cycle computer with in-line assembly language used to optimize the critical loops. The convolutional code was restricted to the rate 1/2, constraint length 24 complementary code (taps 51202215 and

66575563) found by Bahl and Jelinek.[3] This code was selected because it could be simulated within a single computer word and is sufficiently powerful (free distance 24, minimum distance 10) that decoder errors do not limit the system. The Fano algorithm was used for sequential decoding with a simulation speed in excess of 3000 computations per second. All simulations were run with the bias factor R = 0.5 and the threshold spacing = 3.0. One-dimensional parameter studies of this system using the BSC concern track length, tracks per block, stopping rule, and reliability criterion. These tests were run at low signal energy per information bit per noise power spectral density (E_b/N_0) values so that the effects of the parameters could be observed near threshold-of-operation conditions.

Track Length.– It is possible (perhaps desirable from a theoretical point of view) that the track length be very long for the pull-up algorithm. Other practical considerations, such as synchronization, formating, and buffering, require that the track length be reasonably short. Figure 2 shows the effect of track length on computation performance, with the number of tracks fixed at 7. All tracks for all simulations are terminated by a one-constraint-length tail that is included in the rate loss for the code. The value of E_b/N_0 was fixed at 3.43 dB, and for direct comparison, the computation distributions per block were normalized per information bit before being plotted. Average computations are shown in the legend. It can be seen that the computation performance is degraded for a track length of 300 information bits, but that little improvement is actually obtained for a length beyond 500. The track length was fixed at 500 information bits for all other simulations.

Tracks per Block.– The rate loss of bootstrap decoding, determined by the number of tracks per block, m, is significant for small m but it decreases rapidly and then changes relatively little as m becomes large. This fact, and the fact that the effect of the channel state stream is predominant when the number of tracks is small, suggests that an optimum value for m can be found. In addition, the value of m has a direct influence on formating, encoder complexity, and decoder buffering. Simulations were conducted to determine the effect of m on the computation distributions. The track length was fixed at 500 information bits (plus the one-constraint-length tail), and all simulations were carried out for a value of E_b/N_0 held constant at 3.43 dB. Figure 3 is a summary of the results of these simulations, with distributions of normalized computations per information bit plotted for selected values of m and a more complete table of average computations per information bit given in the legend. The irregular variation in computations between values of m is probably due to small sample size (3.675×10^6 bits per value of m), but a broad minimum is indicated between m = 7 and 11. Values of m in this range would be practical for operational use. The number of tracks per block was fixed at 7 for all other simulations.

Stopping Rule.– An effective stopping rule must be devised in order to obtain the maximum efficiency of pull-up bootstrap decoding. The sequential decoder should be allowed to operate as far as it can go easily. Unnecessary time is wasted in restarting when a track is stopped too soon, or in computing, when it is not stopped soon enough. In addition, the stopping rule should provide information about the reliability of the path on which the decoder is operating. Several rules based upon limiting the number of computations per track were devised and tested, but none proved very useful because of the large variation in the number of computations for each track. When the computations limit is set low and increased when no progress has been made on any track, many decoding attempts are required to complete each block. Setting the initial limit high to reduce the number of attempts caused long unnecessary searches. In addition, computations alone do not provide reliability information.

The final and most effective rule devised is based solely on observation of the path likelihood value. Since the likelihood of the correct path tends to increase with depth in the code tree, the rule allows the decoder to operate as long as a drop in the value of the likelihood does not exceed a specified value, D. Mechanization of this rule also gives the needed reliability information. The decoder keeps track of the maximum likelihood value, L_{max}, of any path

visited. Operation is stopped if the decoder attempts to lower the threshold more than D below L_{max}. At this time, the decoder is pointing to a node before the L_{max} node which has a path likelihood approximately D below it. The probability that this node is on the correct path increases with increasing values of D. The definitely decoded section is declared to extend from the starting point up to LBACK nodes from the stopping point, where LBACK is another variable in the stopping rule.

In order to sense stagnation in the decoding process, it is necessary to count the times the definitely decoded section is not increased by NPULL nodes for a single decoding attempt. For all simulations, NPULL was set to 15. The counter, KROUND, is initially set to 0 and reset each time that decoding results in more than NPULL definitely decoded nodes. If the KROUND count becomes equal to the number of undecoded tracks, thus indicating that no progress can be made on any track, the value of D is increased and KROUND reset. At this time, the channel state stream is recomputed and decoding is begun from the first node of each uncompleted track. This procedure allows for correction of possible errors included in definitely decoded sections of the incomplete tracks which may be causing the decoding difficulty. The value of D is reset to its initial value each time a track is completely decoded.

Figures 4 and 5 show the results of simulations for the above scheme. All simulations are for 500 information bits per track, 7 tracks per block, and $E_b/N_0 = 3.43$ dB. For these simulations, D is determined by multiplying the indicated stop factor by the "disagree, 0 state bit" likelihood value for the number of existing uncompleted tracks. Figure 4 shows the effect of several stop factor sequences with LBACK = 50, and figure 5 shows the effect of LBACK using only the 4, 5, 6, 7 stop factor sequence. It can be seen that an initial stop factor of 3 or 4 is optimum with an increase of 1 each time stagnation occurs. For these values the stopping point does not usually contain errors, and LBACK may be small.

Performance of Optimum System.– Figure 6 shows the performance of a pull-up bootstrap decoder for the BSC. System parameters, chosen near the optimum values determined in the previous simulations, were held fixed over the E_b/N_0 range. Although no further attempt was made to optimize the system, these curves provide a good measure for comparison with other systems. The Pareto slope, α, is plotted as a function of E_b/N_0 in figure 7. The R_{comp} point is interpolated to be 3.1 dB. During these simulations 62 decoder errors were observed for the 3.43 dB case. The resulting output bit error rate was about 5×10^{-6}.

It is worthwhile to note here that the power of the code and stopping rule worked very effectively in eliminating decoder errors. Numerous errors were inserted in partially completed tracks but were removed when the tracks were eventually restarted. The 62 errors occurred in one block; 31 were decoded into the second track to be completed and the other 31 were forced by parity into the last track. (Weaker codes have been observed to permit more frequent errors, which were also duplicated in a second track with no significant effect on the computation performance.)

Figure 8 shows pertinent information about decoder operation for one block of the $E_b/N_0 = 3.43$ dB sample. This block was selected because it shows the decoder trying to commit errors (step 7), a change in stopping rule (step 18), the effect of pull-up, and the general reduction in computations per track as the quantity of definitely decoded data is increased (when there are no errors). The step number is KTRY; JNOW is the track being operated on; ITCT is the number of computations for the step; IT is the stopping threshold value; ITMX is the maximum threshold value; DFAC is the stopping rule likelihood drop factor; NSTART is the starting node; N is the stopping node; NMAX is the maximum node depth; KLEFT is the number of uncompleted tracks after the decode step; and KROUND is the number of steps since pull-up.

Figure 9 is a plot of the probability that the total number of decoder steps per block will exceed a given number for the optimum system with E_b/N_0 as a parameter. Note that these curves exhibit a Pareto-type distribution with a sharp change in slope near the R_{comp} point of the system.

Comparison With Other Systems.– It is interesting to compare bootstrap decoding with two other decoding techniques because of their similarities. The first is simple sequential decoding. To provide a means for direct comparison, simulations were performed for the same E_b/N_o values as were used for the bootstrap decoder. The same Fano algorithm, track size, and rate 1/2 convolutional code were used. The results are shown in the normalized computations curves of figure 10 with the Pareto slope values plotted in figure 7. R_{comp} is at approximately 4.6 dB. Bootstrap decoding has a gain of about 1.5 dB over simple sequential decoding.

In order to determine the exact effect of the channel state stream, the pull-up decoder was modified to use standard likelihood values when k ranges from 2 to 7 so the channel state stream is useless, except to pull up the track which is farthest behind the others. Consequently, the algorithm actually behaves like the Falconer algorithm for a 7-bit parity check code, with the exception that the decoder is restarted from the first undecoded node at each decoding attempt. The computation results of these simulations are shown in figure 11 with the Pareto slope values plotted in figure 7. This algorithm has an R_{comp} of about 4.1 dB which is only 0.5 dB better than simple sequential decoding. The use of the channel state stream therefore yields a rather inexpensive 1.0 dB gain.

Extension to Quantized Channel.– Bootstrap decoding would be of little use if it were applicable only to binary output channels since nearly 2 dB can be gained for simple sequential decoding if the output is quantized to eight levels. Jelinek has provided such an extension for the bootstrap decoder.[1] Unfortunately, to make full use of the information provided by the quantized symbols, a large amount of time is required to compute channel state values, which are no longer binary. Excessive computing time, coupled with the large likelihood tables required (15,280 entries for 8-level quantization and 7 tracks), probably makes such a scheme impractical.

Fortunately, there is a compromise – to use the quantized values of the track symbols and maintain only a binary channel state stream. If the receiver outputs are broken into sign and quality bits, u and v, then the channel state values, z, are modulo-two sums of u, as before. Then,

$$\lambda_m \triangleq \log_2 [w_m(u,v,z/x)/w_m(u,v,z)] - R$$

where

$$w_m(u,v,z) = [w_m(u,v,z/0) + w_m(u,v,z/1)]/2$$

and

$$\begin{aligned} w_m(0,v,0/0) &= w_m(1,v,0,1) = w(0,v/0)q_{m-1}(0) \\ w_m(0,v,1/0) &= w_m(1,v,1/1) = w(0,v/0)q_{m-1}(1) \\ w_m(1,v,0/0) &= w_m(0,v,0/1) = w(1,v/0)q_{m-1}(1) \\ w_m(1,v,1/0) &= w_m(0,v,1/1) = w(1,v/0)q_{m-1}(0) \end{aligned}$$

$q_k(z)$ is defined as before, and

$$p = \sum_v w(1,v/0)$$

According to theoretical bounds derived by Jelinek,[1] full use of the 8-level channel gives an additional gain of about 1.7 dB over the BSC for rate 1/2 bootstrap decoding. Using a binary state stream for this channel causes a theoretical degradation of only 0.1 dB, which is a small price to pay since the channel state computation and likelihood look-up are direct and the table size is only four times larger than for the BSC.

The simulation program was modified for the quantized channel with binary state stream with no significant change in speed. Simulations were performed for eight levels of output with

quantization spacing of 0.5 σ for all E_b/N_o. Tests were conducted which determined the optimum values for the stop factor sequence to be 2.0, 2.5, 3.0, 3.5 times the "strongest disagree, 0 state bit" likelihood with LBACK = 10, 7 tracks, 500 information bits per track, and E_b/N_o = 1.91 dB. Extensive computer runs were made under these conditions for a range of E_b/N_o values. The resulting computation performance curves are shown in figure 12. The observed Pareto slopes are plotted in figure 7 for comparison with the other simulations. The interpolated R_{comp} point is at 1.7 dB, a gain of 1.4 dB over the BSC and 1.0 dB better than rate 1/2 sequential decoding using the octal channel. Figure 7 also shows an interesting thresholding effect for the codes plotted -- the threshold is approached more sharply as code power increases. Over 27,000 blocks were run for the 1.91 dB case (near the threshold of operation) in order to look for any peculiar deviation in computations performance for low probabilities of $C > T$. The Pareto slope remained constant over the significant range. For this case, 190 bit errors were observed in 4 blocks for a probability of bit error less than 2.5×10^{-6}.

Conclusions.– Simulations have provided a great deal of experience with the bootstrap decoding algorithm. Although a number of questions remain unanswered (e.g., effects of channel memory and likelihood/channel mismatch), it is clear that this technique offers a gain of about 1.0 dB over that obtainable from sequential decoding alone. Bootstrap decoding has been shown to operate under the constraints imposed by digital communication systems, such as those typical of deep space. A bootstrap decoding system would be relatively complex, but appears suitable for low-to-moderate data rates where the value of 1.0 dB is worth the cost of implementation.

Acknowledgement.– The research for this paper was performed in conjunction with a study contract with Cornell University, Prof. F. Jelinek, principal investigator.

References

1. F. Jelinek, and J. Cocke, "Bootstrap hybrid decoding for symmetrical binary input channels," Information and Control, March 1971.
2. D. D. Falconer, "A hybrid coding scheme for discrete memoryless channels," Bell System Technical Journal, vol. 48, pp. 691-728, March 1969.
3. L. R. Bahl, and F. Jelinek, "Rate 1/2 convolutional codes with complementary generators," to appear in the IEEE Transactions on Information Theory.

Figure Captions

Fig. 1 – Likelihood values λ_k for $p = 0.09$ and $R = 0.0$.

Fig. 2 – Pull-up decoder computations performance as a function of track length.

Fig. 3 – Pull-up decoder computations performance as a function of tracks per block.

Fig. 4 – Pull-up decoder computations performance as a function of stop factor.

Fig. 5 – Pull-up decoder computations performance as a function of LBACK.

Fig. 6 – Optimized pull-up decoder computations performance for the BSC as a function of E_b/N_o.

Fig. 7 – Pareto exponent vs. E_b/N_o for several decoding techniques.

Fig. 8 – Sample program output.

Fig. 9 – Probability that the number of decode steps will exceed K for the optimized pull-up decoder as a function of E_b/N_o.

Fig. 10 –Simple sequential decoder computations performance for the BSC as a function of E_b/N_o.

Fig. 11 – Pseudo Falconer decoder computations performance for the BSC as a function of E_b/N_o.

Fig. 12 – Pull-up decoder computations performance for the octal channel as a function of E_b/N_o.

RECEIVED BIT/ HYPOTHESIS	CHANNEL STATE BIT, z	NUMBER OF UNDECODED TRACKS, k														
		1	2	3	4	5	6	7	8	9	10	15	20	25	30	35
AGREE	0	1.000	0.986	0.972	0.959	0.947	0.936	0.926	0.917	0.909	0.902	0.879	0.870	0.866	0.865	0.864
	1	$-\infty$	.000	.410	.576	.664	.719	.755	.780	.799	.813	.847	.858	.862	.863	.864
DISAGREE	0	$-\infty$	−5.690	−4.717	−4.618	−3.797	−3.525	−3.318	−3.156	−3.027	−2.924	−2.638	−2.535	−2.496	−2.482	−2.477
	1	1.000	.000	−.576	−.972	−1.267	−1.496	−1.677	−1.823	−1.941	−2.037	−2.311	−2.414	−2.452	−2.466	−2.471

Fig. 1

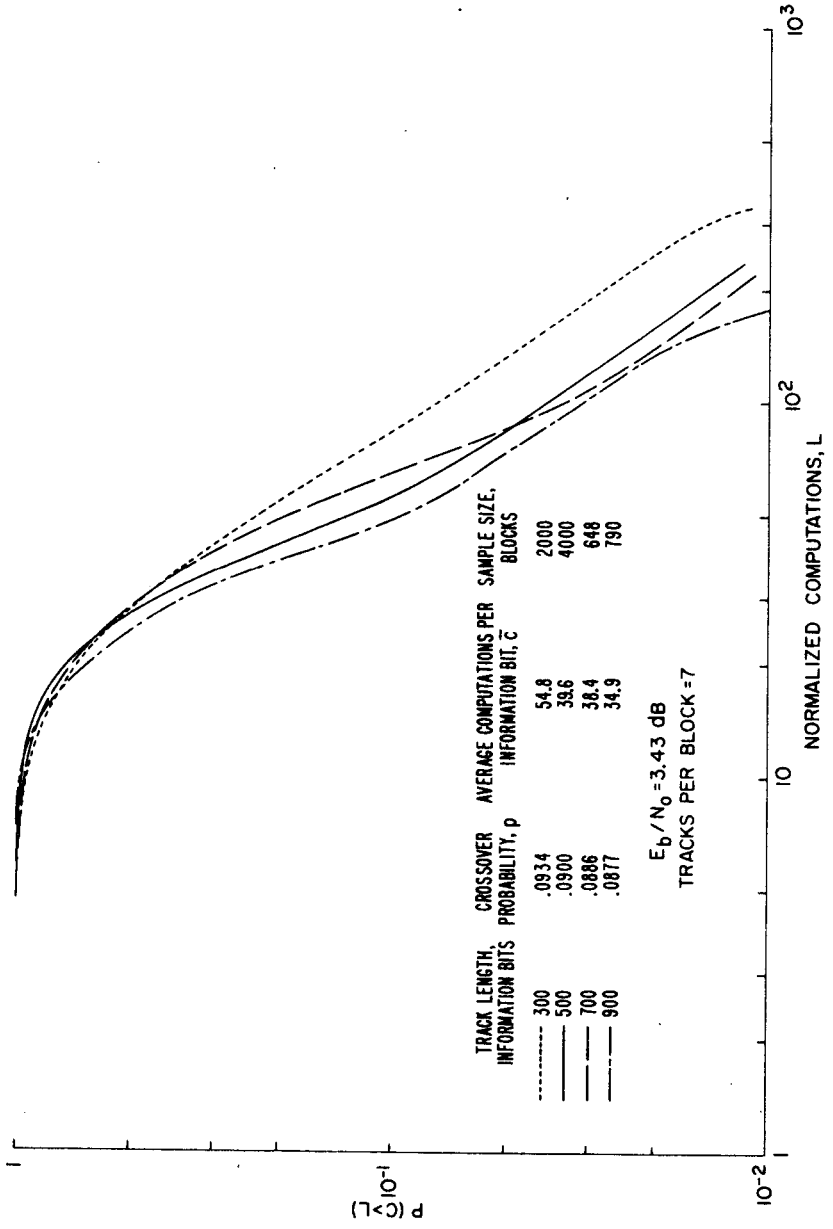

$P(\tilde{C} \geq L)$
10^{-1}
1
10^{-2}
10
10^2
10^3
NORMALIZED COMPUTATIONS, L
TRACK LENGTH, INFORMATION BITS
CROSSOVER PROBABILITY, p
AVERAGE COMPUTATIONS PER INFORMATION BIT, $\bar{C}$
SAMPLE SIZE, BLOCKS
300 .0934 54.8 2000
500 .0900 39.6 4000
700 .0886 38.4 648
900 .0877 34.9 790
E_b/N_0 =3.43 dB
TRACKS PER BLOCK=7

Fig. 2

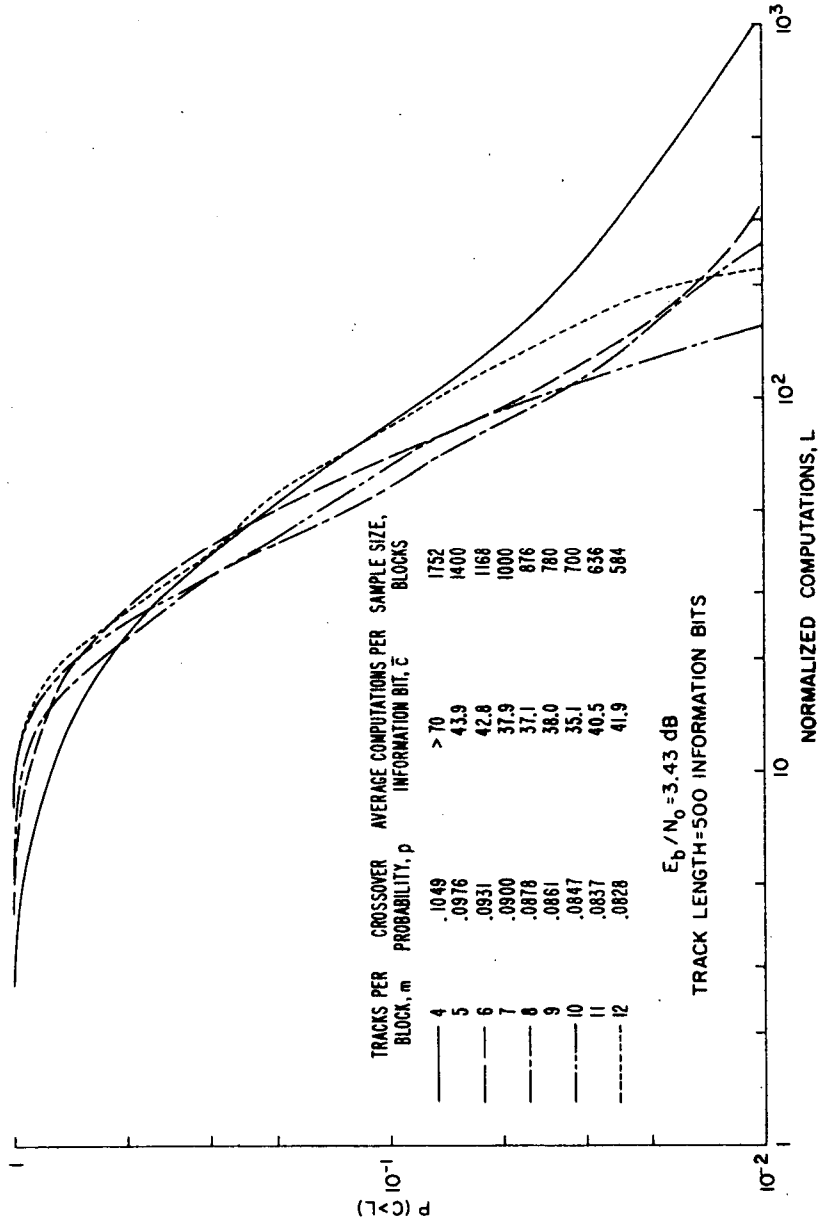

P (C > L)
1
10^{-1}
10^{-2}
1
10
10^2
10^3
NORMALIZED COMPUTATIONS, L
E_b/N_0 =3.43 dB
TRACK LENGTH=500 INFORMATION BITS
TRACKS PER BLOCK, m
CROSSOVER PROBABILITY, p
AVERAGE COMPUTATIONS PER INFORMATION BIT, C̄
SAMPLE SIZE, BLOCKS
4 .1049 > 70 1752
5 .0976 43.9 1400
6 .0931 42.8 1168
7 .0900 37.9 1000
8 .0878 37.1 876
9 .0861 38.0 780
10 .0847 35.1 700
11 .0837 40.5 636
12 .0828 41.9 584

Fig. 3

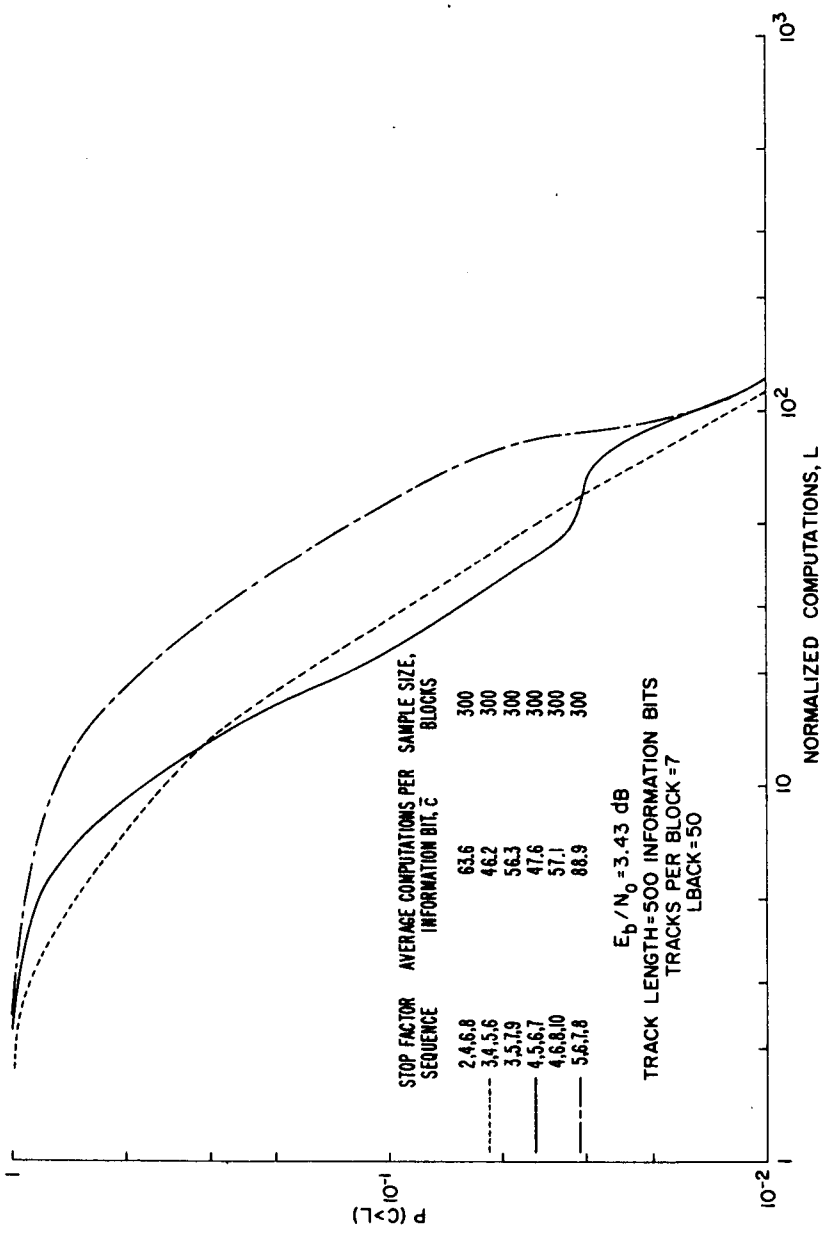

P (C>L)
NORMALIZED COMPUTATIONS, L
STOP FACTOR SEQUENCE
AVERAGE COMPUTATIONS PER INFORMATION BIT, C
SAMPLE SIZE, BLOCKS
2,4,6,8
63.6
300
3,4,5,6
46.2
300
3,5,7,9
56.3
300
4,5,6,7
47.6
300
4,6,8,10
57.1
300
5,6,7,8
88.9
300
E_b/N_0 = 3.43 dB
TRACK LENGTH=500 INFORMATION BITS
TRACKS PER BLOCK=7
LBACK=50

Fig. 4

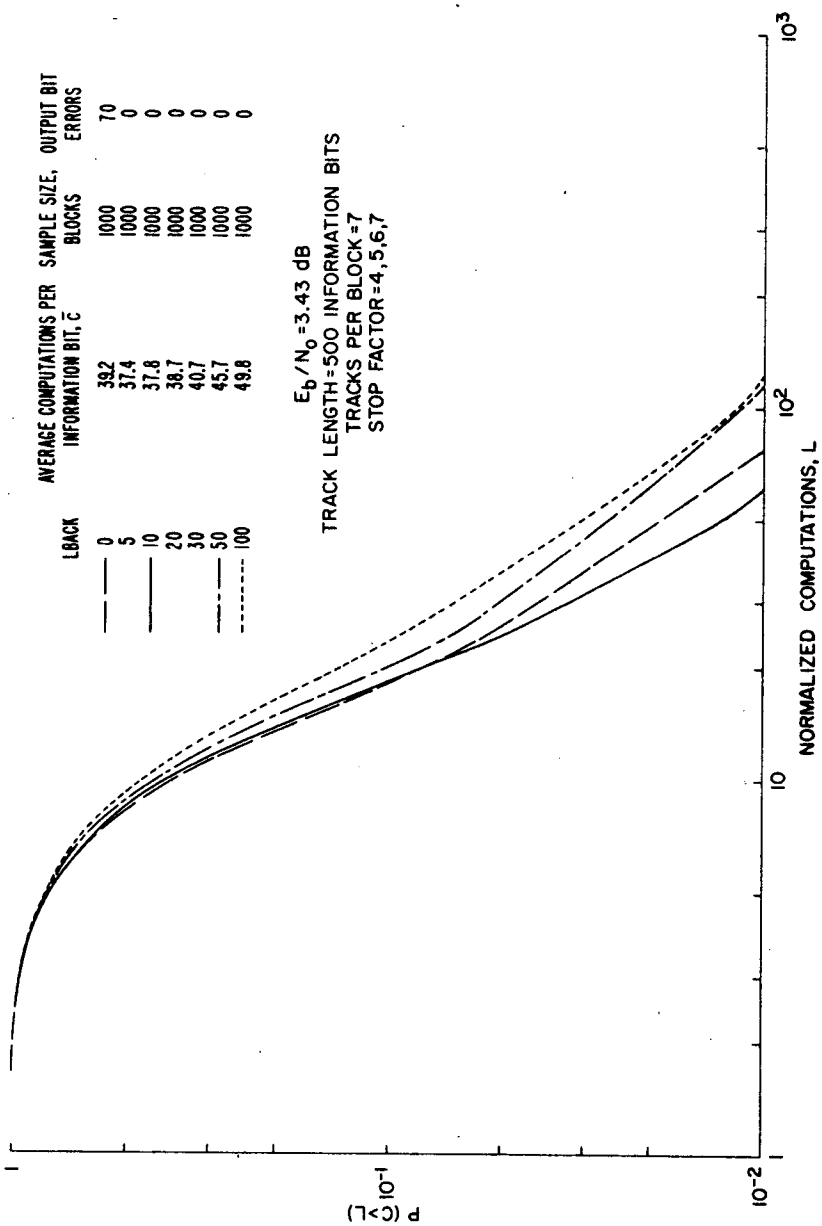
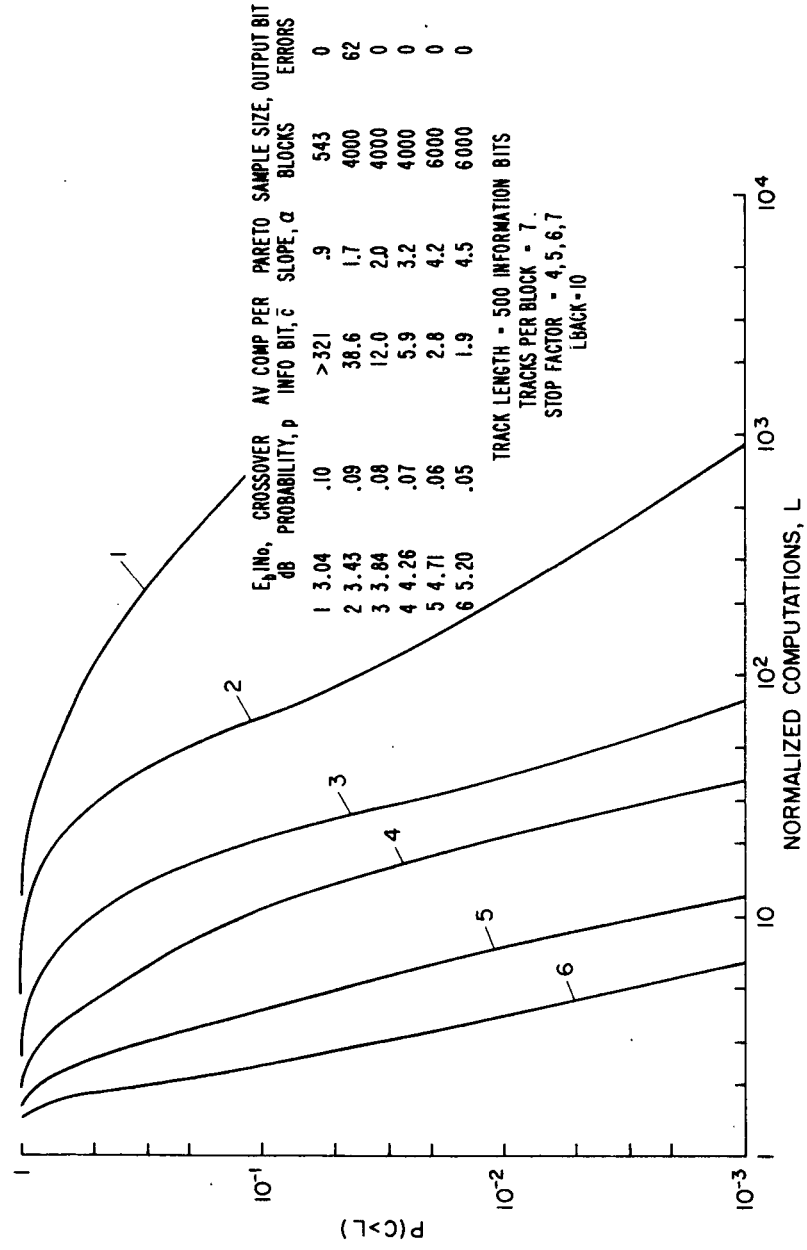

AVERAGE COMPUTATIONS PER INFORMATION BIT, $\bar{C}$
SAMPLE SIZE, BLOCKS
OUTPUT BIT ERRORS
LBACK
0 39.2 1000 70
5 37.4 1000 0
10 37.8 1000 0
20 38.7 1000 0
30 40.7 1000 0
50 45.7 1000 0
100 49.8 1000 0
E_b/N_0 =3.43 dB
TRACK LENGTH=500 INFORMATION BITS
TRACKS PER BLOCK=7
STOP FACTOR=4,5,6,7
P (C>L)
NORMALIZED COMPUTATIONS, L
1
10^{-1}
10^{-2}
10
10^2
10^3

Fig. 5



	E_b/N_o, dB	CROSSOVER PROBABILITY, p	AV COMP PER INFO BIT, $\bar{c}$	PARETO SLOPE, α	SAMPLE SIZE, BLOCKS	OUTPUT BIT ERRORS
1	3.04	.10	>321	.9	543	0
2	3.43	.09	38.6	1.7	4000	62
3	3.84	.08	12.0	2.0	4000	0
4	4.26	.07	5.9	3.2	4000	0
5	4.71	.06	2.8	4.2	6000	0
6	5.20	.05	1.9	4.5	6000	0

Fig. 6

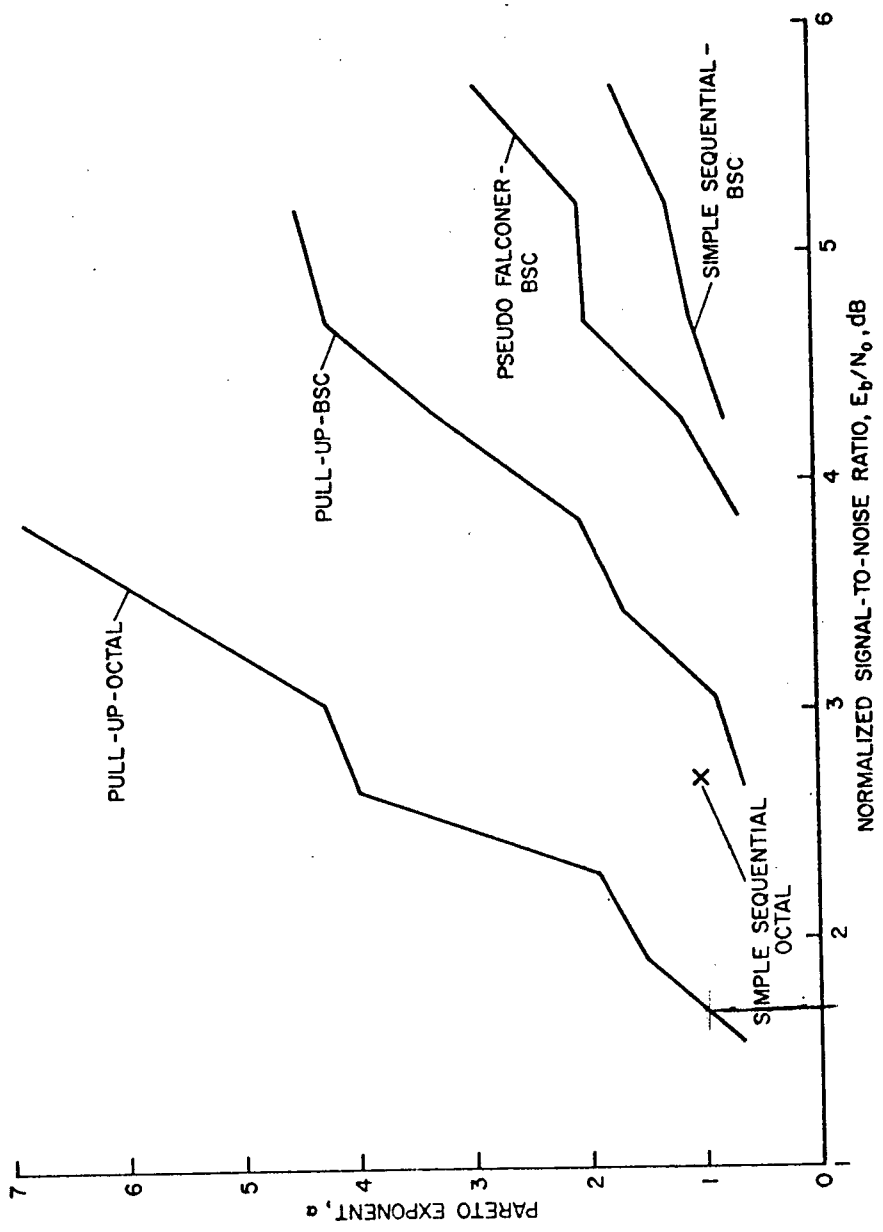

PULL-UP-OCTAL
PULL-UP-BSC
PSEUDO FALCONER-BSC
SIMPLE SEQUENTIAL-BSC
SIMPLE SEQUENTIAL OCTAL
PARETO EXPONENT, α
NORMALIZED SIGNAL-TO-NOISE RATIO, E_b/N_o, dB
0
1
2
3
4
5
6
7

Fig. 7

KTRY	JNOW	ITCT	IT	ITMX	DFAC	NSTART	N	NMAX	KLEFT	KROUND
1	1	49326	15	33	4	1	94	253	7	0
2	2	46987	30	48	4	1	198	395	7	0
3	3	28994	63	81	4	1	310	503	7	0
4	4	16665	18	36	4	1	117	267	7	0
5	5	8011	114	114	4	1	524	524	6	0
6	6	15276	81	99	4	1	275	377	6	0
7	7	23223	141	159	4	1	353	504	6	0
7	7	ERRORS DECODED IN POSITIONS 328, 329, 330, 332, 333, 336, 339, 340								
8	1	1301	147	150	4	85	524	524	5	0
9	2	13710	33	51	4	189	348	465	5	0
10	3	9262	3	21	4	301	311	423	5	1
11	4	2023	174	192	4	108	295	351	5	0
12	6	4717	21	39	4	266	292	387	5	0
13	7	11577	−3	15	4	344	354	467	5	1
14	2	13562	−6	12	4	339	348	465	5	2
15	3	7828	3	21	4	302	311	423	5	3
16	4	2199	3	21	4	286	295	351	5	4
17	6	7220	3	21	4	283	291	405	5	5
18	7	213150	81	105	5	1	214	502	5	0
19	2	235821	36	60	5	1	195	403	5	0
20	3	15744	120	123	5	1	524	524	4	0
21	4	10008	159	159	4	1	524	524	3	0
22	6	4868	198	219	4	1	222	303	3	0
23	7	4777	78	78	4	205	524	524	2	0
24	2	1104	144	144	4	186	524	524	1	0
25	6	312	312	312	4	213	524	524	0	0

ERROR RATE BY TRACKS 0.0914, 0.1105, 0.0924, 0.1010, 0.0857, 0.0924, 0.0819
TOTAL COMPUTATIONS THIS BLOCK 747655

Fig. 8

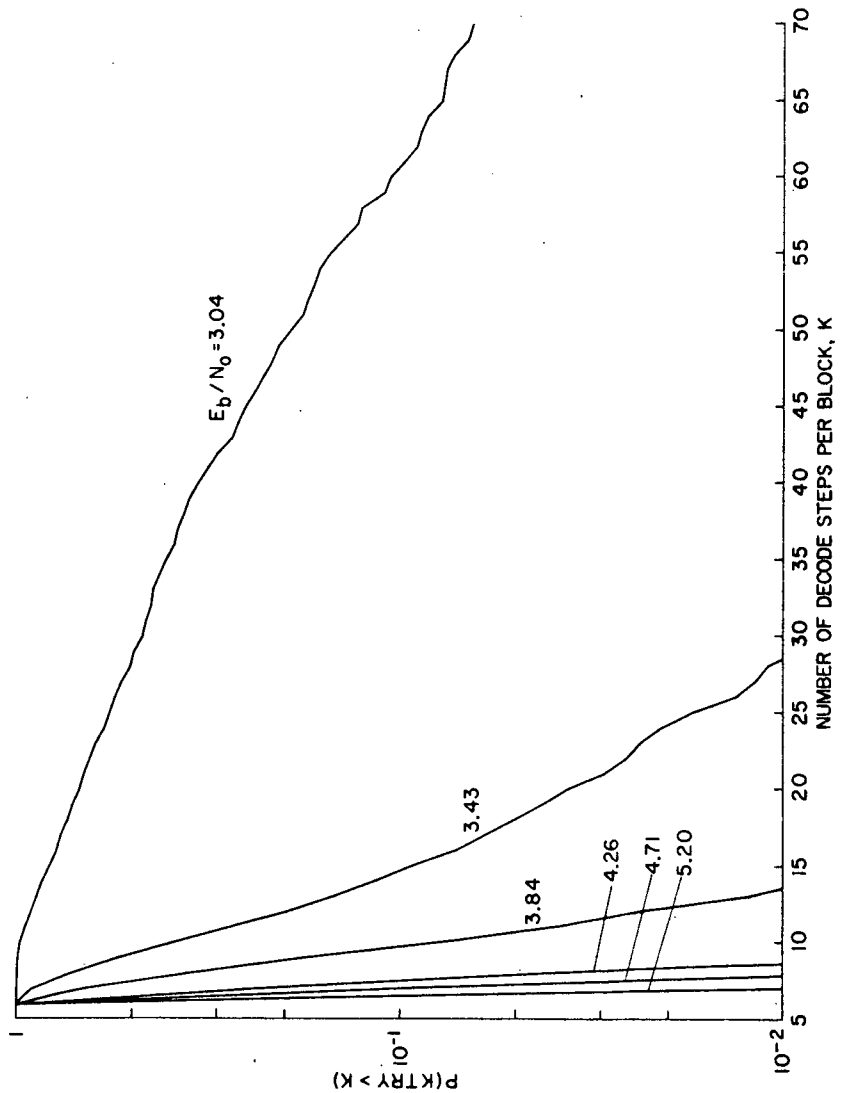

P(KTRY > K)
1
10^{-1}
10^{-2}
5
10
15
20
25
30
35
40
45
50
55
60
65
70
NUMBER OF DECODE STEPS PER BLOCK, K
E_b/N_o=3.04
3.43
3.84
4.26
4.71
5.20

Fig. 9

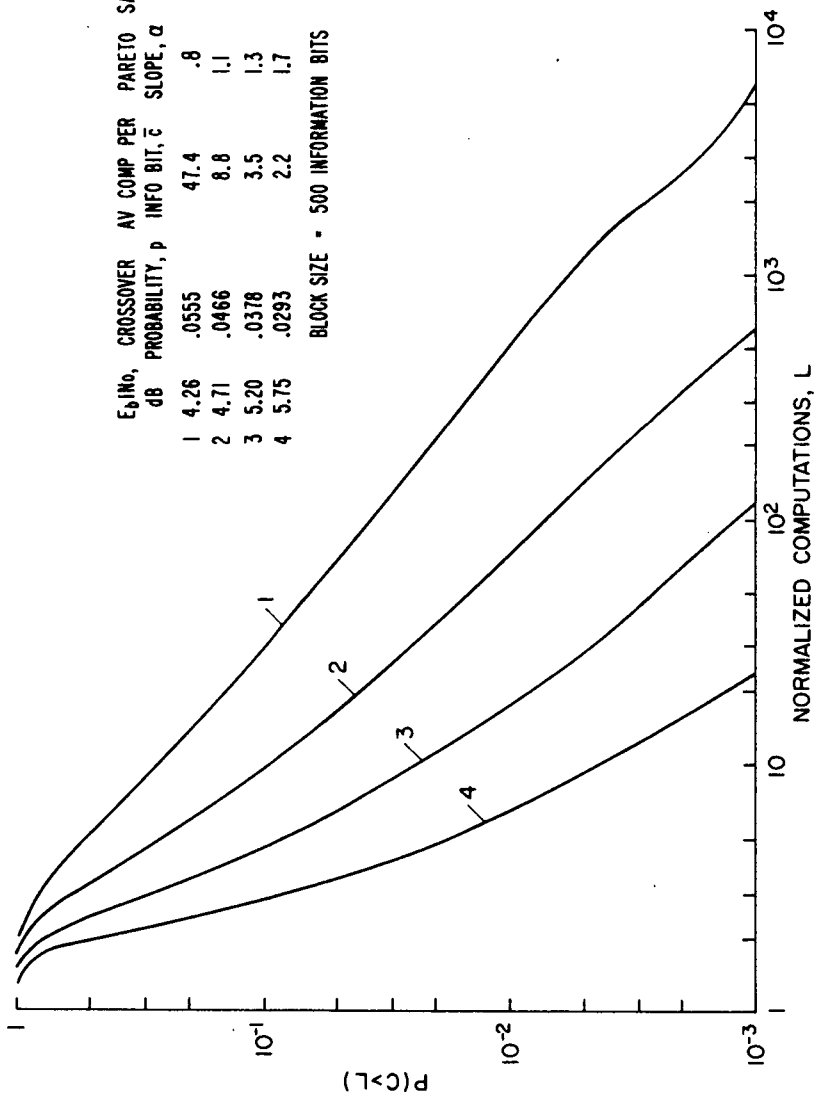

E_b/N_o, dB
CROSSOVER PROBABILITY, p
AV COMP PER INFO BIT, $\bar{c}$
PARETO SLOPE, α
SAMPLE SIZE, BLOCKS
1 4.26 .0555 47.4 .8 12190
2 4.71 .0466 8.8 1.1 30000
3 5.20 .0378 3.5 1.3 30000
4 5.75 .0293 2.2 1.7 30000
BLOCK SIZE = 500 INFORMATION BITS
P(C>L)
NORMALIZED COMPUTATIONS, L
1
10^{-1}
10^{-2}
10^{-3}
10
10^2
10^3
10^4

Fig. 10

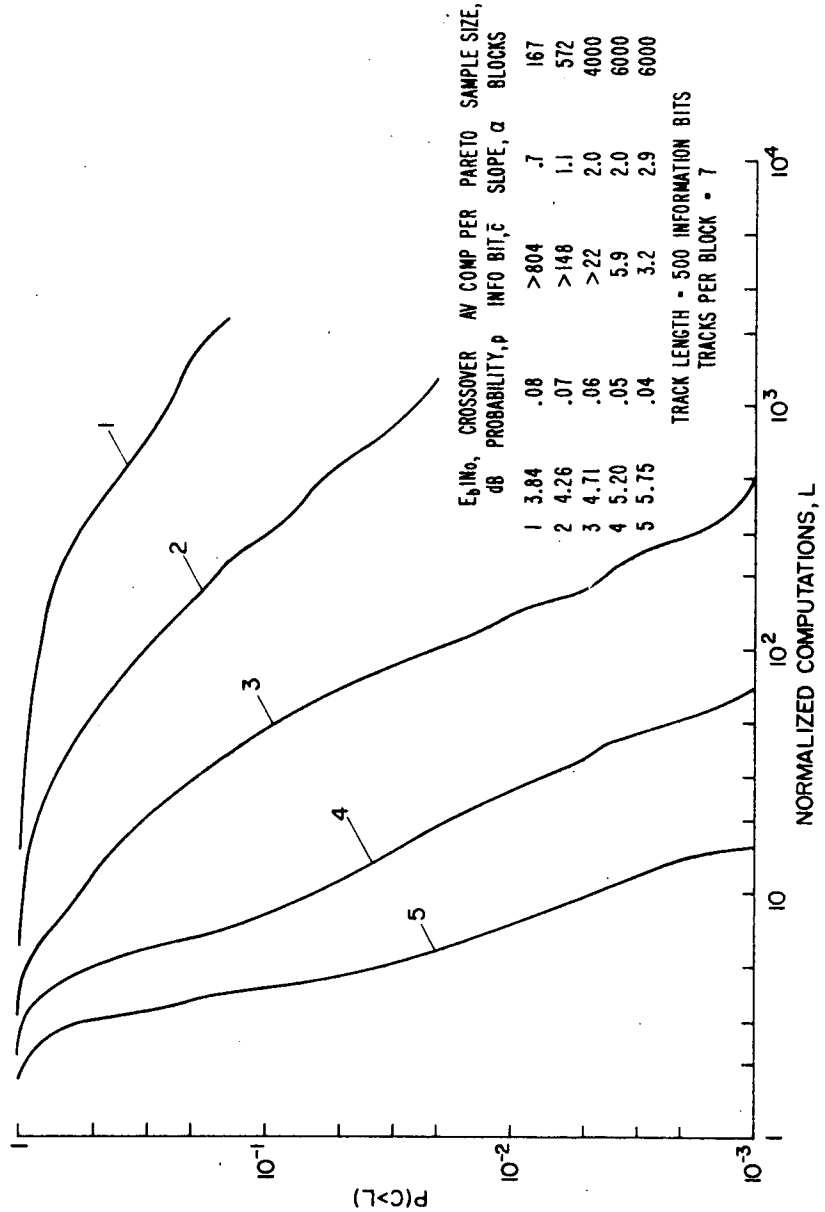

P(C>L)
NORMALIZED COMPUTATIONS, L
TRACK LENGTH - 500 INFORMATION BITS
TRACKS PER BLOCK - 7
Eb/No, dB
CROSSOVER PROBABILITY, p
AV COMP PER INFO BIT, C̄
PARETO SLOPE, α
SAMPLE SIZE, BLOCKS
1 3.84 .08 >804 .7 167
2 4.26 .07 >148 1.1 572
3 4.71 .06 >22 2.0 4000
4 5.20 .05 5.9 2.0 6000
5 5.75 .04 3.2 2.9 6000

Fig. 11

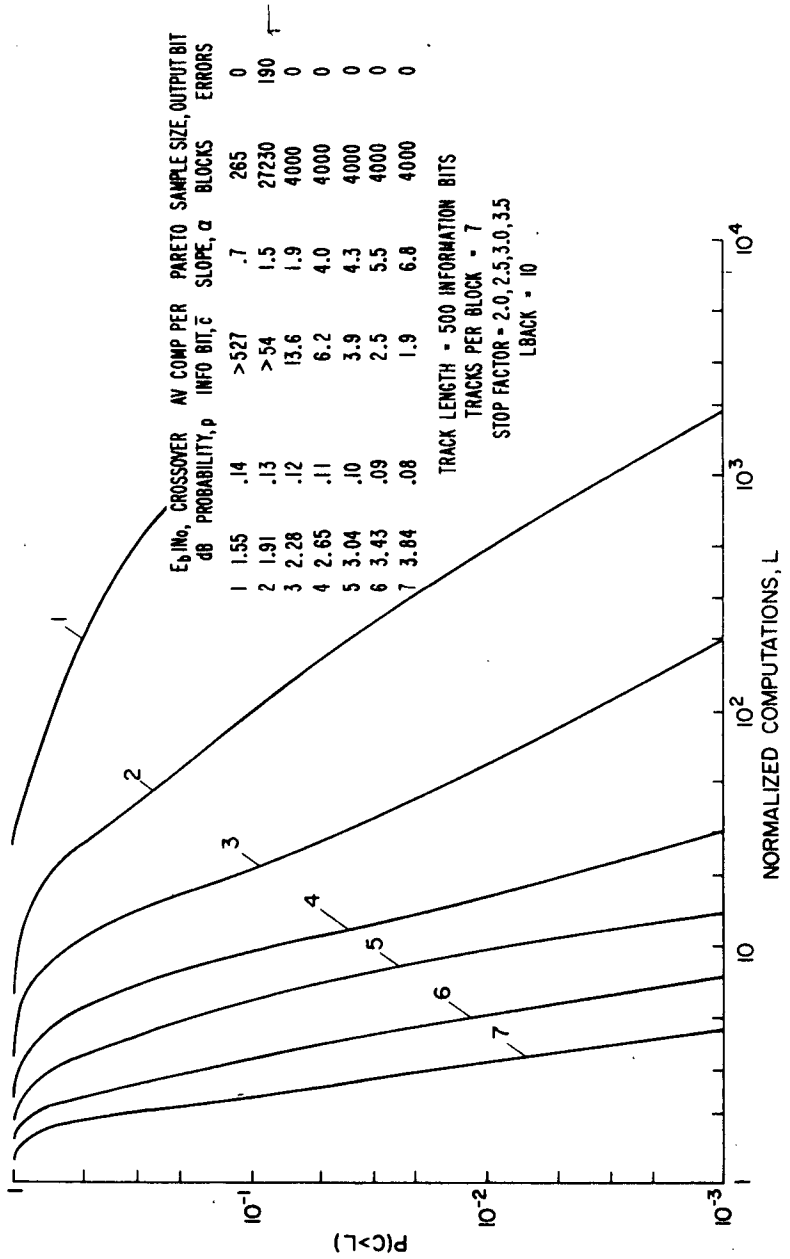

E_b/N_o, dB
CROSSOVER PROBABILITY, p
AV COMP PER INFO BIT, $\bar{c}$
PARETO SLOPE, α
SAMPLE SIZE, BLOCKS
OUTPUT BIT ERRORS
1 1.55 .14 >527 .7 265 0
2 1.91 .13 >54 1.5 27230 190
3 2.28 .12 13.6 1.9 4000 0
4 2.65 .11 6.2 4.0 4000 0
5 3.04 .10 3.9 4.3 4000 0
6 3.43 .09 2.5 5.5 4000 0
7 3.84 .08 1.9 6.8 4000 0
TRACK LENGTH - 500 INFORMATION BITS
TRACKS PER BLOCK - 7
STOP FACTOR - 2.0, 2.5, 3.0, 3.5
LBACK - 10
P(C>L)
NORMALIZED COMPUTATIONS, L
1
10^{-1}
10^{-2}
10^{-3}
10
10^2
10^3
10^4

Fig. 12

II-D. Effect of Likelihood Bias on Sequential Decoding Parameters

1. Introduction

The performance of sequential decoding has traditionally been evaluated in terms of three characteristics: the probability of undetectable error ([1], p. 349), the probability of failure of order t([1], p. 349), and the Pareto exponent associated with the decoding effort ([1], p. 349). Most published bounds on these quantities assume that the decoder uses the likelihood metric

$$\log \frac{w(y/x)}{w(y)} - R \tag{1}$$

where R is the rate of the code used, w(y/x) is the channel transmission probability function, and w(y) is the marginal probability distribution of received digits based on the optimal code ensemble. It is generally known [1] that the three quantities of interest are optimized by the metric form

$$\log \frac{w(v/x)}{w(y)} - G \tag{2}$$

where the optimal value of G may be different for each of the three cases. For instance, Zigangirov [2] manipulates G to minimize the probability of failure, and Stiglitz and Yudkin explore some effects of G-variation in an unpublished memorandum [3]. However, their use of simplifying inequalities at certain critical points of their development prevents them from obtaining the strongest achievable results.

The trade-off between the three performance parameters is interesting from the point of view of Bootstrap Hybrid Decoding [4]. In one mode of the pull-up version of the algorithm, digits of branch depth J-t and less are definitely decoded if the deepest penetration of the decoder was to branch level J.

Making the retreat length t as short as possible will tend to decrease the decoding effort as long as no error at depth J-t or less was committed. Otherwise the definite decision will have possibly catastrophic consequences. Hence all other things being equal, G should be adjusted so as to minimize the probability of failure. We will see below that usually such setting will lower somewhat the Pareto exponent of the sequential decoding component of the scheme, and will increase the probability of undetectable error. The latter difficulty may be cheaply remedied by an increase in constraint length, but what the best compromise is between the failure and Pareto exponent parameters remains an open question.

A second mode of the pull-up version of Bootstrap Decoding definitely decodes digits by the following rule: Let the decoder be located at some node whose likelihood is L and let the path leading to that node contain some node n* at depth τ whose cumulative likelihood does not exceed L-a. Then the decoder will definitely decide to release to the user all τ branches of the path leading to n*. How to set the value of the likelihood drop a depends on Q(a), the probability that with zero likelihood value assigned to a root node, there exists a node in the incorrect subset whose cumulative likelihood exceeds a. Q(a) is thus a fourth performance parameter of interest.

This paper attempts to determine the effects of G-variations on the four performance characteristics. In sections 2 through 5 we deal with random coding upper bounds. In sections 6 and 7 we develop expurgated

bounds for the probabilities of failure and undetected error. We show that the former is identical to the one developed by Viterbi and Odenwalder [10] for maximum likelihood decoding, and that the latter leads to the block coding expurgated exponent. In section 8 we present some curves that apply our results to quantized Gaussian additive noise channels with binary inputs.

2. Definitions and Basic Upper Bounds

As is usual, we will work with the random coding ensemble and we will not bother to argue that the obtained bounds are simultaneously valid for particular codes as well. To save space, we will use the notation and some of the intermediate results from Chapter 10 of Jelinek [1]. However, to simplify matters further, we will adopt the stack sequential decoding algorithm [5] that leads asymptotically to the same results as the Fano algorithm. The reader will be assumed familiar with both. Our random codes of rate $\frac{k}{n}$ will have the trellis structure of Figure 1 (see also p. 336 of [1]) with 2^k branches leaving each node, each branch associated with a block of n channel input digits x (in Figure 1, k = 1 and n = 2, and the channel input alphabet is binary). Each level of the trellis will contain $2^{k(\upsilon-1)}$ states, where υ is called the branch constraint length of the code. The information digits that determine the path that the encoder takes through the trellis are binary, the state being determined uniquely by $k(\upsilon-1)$ most recent bits (by convention, the information preceding time t = 0 is assumed to consist of 0's). In the random ensemble, each digit of each branch of the trellis is selected independently, at random, with some probability distribution r(x) over the channel inputs. The coding trellis generates a coding tree whose root node corresponds to the initial all-zero trellis state. In this paper we will consider infinite depth trees and trellises.

An undetectable error is committed at depth i by a sequential decoder if, after it operated without any restriction on the number and depth of

returns, the i^{th} branch on the finally decoded trellis path differs from the one actually taken by the encoder. We will be interested in $U(\upsilon)$, the average number of undetectable errors per decoded digits when a random code of constraint length υ was used.

A failure of order t takes place if the decoder advances by t branches or more into the incorrect subset of the coding tree. We will be interested in the probability of failure $P_f(t)$.

Let N_i be the number of times the sequential decoder is located at some node of the incorrect subset stemming from a correct node on level i. Then $\overline{N_i^{\gamma}}$ is the γ^{th} moment of the decoding effort at depth i. Let α be the supremum of the values γ for which $\overline{N_i^{\gamma}}$ is bounded. α is then called the Pareto exponent of the decoding effort.

In the preceding section we have defined $Q(a)$, the probability of an a-likelihood advance in the incorrect subset.

Let $\underset{\sim}{s} = (s_1, s_2, \ldots)$ denote some path in the tree determined by information digits s_i, $i = 1, 2, \ldots$, let $\underset{\sim}{s}^*$ be the correct path and let $\underset{\sim}{x}^t(\underset{\sim}{s})$ denote the code digits corresponding to the initial t branches of $\underset{\sim}{s}$. Let $\mathcal{D}^t$ denote the set of nodes at depth t of the incorrect subset stemming from the root node, and let $G^{t+\upsilon}$ be the subset of nodes of $\mathcal{D}^{t+\upsilon}$ that corresponds to trellis paths whose first branch is incorrect and which rejoin the correct path for the first time at depth $\upsilon+t$ (i.e., these are paths containing at most t incorrect information digits).

The following upper bounds have been proved in Jelinek [1], pp. 354-359 (we have made some adjustments to assure applicability to the stack algorithm) where $0 \leq \sigma, \gamma$:

$$U(\upsilon) \leq \sum_{t=1}^{\infty} \sum_{m=0}^{\infty} t2^{(m-t-\upsilon)G} \cdot \underset{\sim}{E} \left\{ \sum_{\underset{\sim}{s} \in G^{t+\upsilon}} \left[\frac{w(\underset{\sim}{\chi}^t/\underset{\sim}{x}^t(\underset{\sim}{s}))\ w(\underset{\sim}{\chi}^m)}{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m(\underset{\sim}{s}^*))\ w(\underset{\sim}{\chi}^t)} \right]^{\sigma} \right\}^{\delta} \tag{3}$$

$$P_f(t) \leq \sum_{m=0}^{t} 2^{\delta\sigma(m-t)nG}\ \underset{\sim}{E} \left\{ \sum_{\underset{\sim}{s} \in \mathcal{D}^t} \left[\frac{w(\underset{\sim}{\chi}^t/\underset{\sim}{x}^t(\underset{\sim}{s}))\ w(\underset{\sim}{\chi}^m)}{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m(\underset{\sim}{s}^*))\ w(\underset{\sim}{\chi}^t)} \right]^{\sigma} \right\}^{\delta} \tag{4}$$

and for $\gamma \leq 1$,

$$\overline{N^{\gamma}} \leq \sum_{t=0}^{\infty} \sum_{m=0}^{\infty} 2^{\gamma\sigma(m-t)nG}\ \underset{\sim}{E} \left\{ \sum_{\mathcal{D}} \left[\frac{w(\underset{\sim}{\chi}^t/\underset{\sim}{x}^t(\underset{\sim}{s}))\ w(\underset{\sim}{\chi}^m)}{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m(\underset{\sim}{s}^*))\ w(\underset{\sim}{\chi}^y)} \right]^{\sigma} \right\}^{\gamma} \tag{5}$$

An upper bound on $\overline{N^{\gamma}}$ for $\gamma > 1$ has also been derived by Jelinek [6]. However, the purpose of this paper is to investigate the effect of not taking certain usual bounding shortcuts which alone make the bound on $\overline{N^{\gamma}}$ for $\gamma > 1$ tractable. We will therefore restrict ourselves to the case $\gamma \leq 1$ which is the one for which optimal choice of G is crucial. An adventurous reader may in any case decide to use our conclusions as a guide for action in the region $\gamma > 1$.

Before bounding Q(a) let us observe that (4) and (5) have the expectation term in common and that the expectation term in (3) is similar. In Appendix I we have bounded these as follows.

Define the exponent functions

$$f_1(\lambda) = \log \sum_y w(y) \sum_x \left[\frac{w(y/x)}{w(y)}\right]^{1-\lambda} r(x) \tag{6}$$

$$f_2(\sigma,\gamma) = \log \sum_y w(y) \left\{\sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\sigma} r(x)\right\}^{\gamma} \left\{\sum_z \left[\frac{w(y/z)}{w(y)}\right]^{1-\sigma\gamma} r(z)\right\} \tag{7}$$

$$f_3(\sigma,\gamma) = \log \sum_y w(y) \left\{\sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\sigma} r(x)\right\}^{\gamma} \tag{8}$$

then if $\gamma \in [0, 1]$,

$$\underset{\sim}{E}\left\{\sum_{\underset{\sim}{s}\in B} \left[\frac{w(\underset{\sim}{y}^t/\underset{\sim}{x}^t(\underset{\sim}{s}))\, w(\underset{\sim}{y}^m)}{w(\underset{\sim}{y}^m/\underset{\sim}{x}^m(\underset{\sim}{s}))\, w(\underset{\sim}{y}^t)}\right]^{\sigma}\right\}^{\gamma} \leq \begin{cases} 2^{n[(m-t)f_1(\sigma\gamma) + t f_2(\sigma,\gamma) + \ell(B)\gamma R]} & \text{if } m \geq t \\ 2^{n[(t-m)f_3(\sigma,\gamma) + m f_2(\sigma,\gamma) + \ell(B)\gamma R]} & \text{if } m < t \end{cases} \tag{9}$$

where B is either equal to $\mathcal{D}^t$ or to G^t and $\ell(\mathcal{D}^t) = t$, $\ell(G^t) = t-\upsilon$.

3. Random Coding Upper Bounds on Performance Parameters

In this section we use (9) to obtain upper bounds on $U(\upsilon)$, $P_f(t)$, and $\overline{N^\delta}$ and develop an upper bound on $Q(a)$. Substituting (9) into (3) we get

$$U(\upsilon) \leq \sum_{t=1}^{\infty} \sum_{m=t+\upsilon}^{\infty} t \exp_2 n \left\{ \delta\sigma(m-t-\upsilon)G + t\delta R + (t+\upsilon) f_2(\sigma, \delta) + (m-t-\upsilon) f_1(\sigma\delta) \right\}$$

$$+ \sum_{t=1}^{\infty} \sum_{m=0}^{t+\upsilon-1} t \exp_2 n \left\{ \delta\sigma(m-t-\upsilon)G + t\,\delta R + m f_2(\sigma, \delta) + (\upsilon+t-m) f_3(\sigma, \delta) \right\} \tag{10}$$

where $\delta \in [0, 1]$, $\sigma \geq 0$. Using the geometrical sum formula, the first term in (10) is bounded by $K_1 2^{\upsilon n f_2(\sigma, \delta)}$ where K_1 is finite provided

$$\delta\sigma G + f_1(\sigma\delta) < 0$$

$$\delta R + f_2(\sigma, \delta) < 0 \ . \tag{11}$$

It is best to break up the second sum in (10) into two parts, the first for $m \in [0, \upsilon-1]$ and the second for $m \in [\upsilon, \upsilon+t-1]$. The first part is then equal to

$$\exp_2 n\upsilon \left\{ f_3(\sigma, \gamma) - \delta G \right\} \cdot \sum_{m=0}^{\upsilon-1} \exp_2 nm \left\{ \delta\sigma G + f_2(\sigma, \delta) - f_3(\sigma, \delta) \right\} \cdot$$

$$\cdot \sum_{t=1}^{\infty} t \exp_2 nt \left\{ f_3(\sigma, \delta) + \delta R - \sigma\delta G \right\} \tag{12}$$

The result then depends on whether the exponent in the second summation is positive or negative. Thus the bound is $K_2 2^{n\upsilon f_2(\sigma, \delta)}$ where K_2 is finite provided

$$\delta\sigma G + f_2(\sigma, \delta) - f_3(\sigma, \delta) \geq 0$$

$$f_3(\sigma, \delta) + \delta R - \sigma\delta G < 0 \tag{13}$$

C.2

and it is $K_3 2^{n\upsilon[f_3(\sigma,\delta)-\sigma\delta G]}$ where K_3 is finite provided

$$\sigma\delta G + f_2(\sigma,\delta) - f_3(\sigma,\delta) < 0$$

$$f_3(\sigma,\delta) + \delta R - \sigma\delta G < 0 \,. \tag{14}$$

The second part of the second sum in (10) is equal to

$$\exp_2 n\upsilon \left\{f_3(\sigma,\delta) - \sigma\delta G\right\} \sum_{m=\upsilon}^{\infty} \exp_2 nm \left\{\sigma\delta G + f_2(\sigma,\delta) - f_3(\sigma,\delta)\right\} \cdot$$

$$\cdot \sum_{t=m-\upsilon+1}^{\infty} t \exp_2 nt \left\{f_3(\sigma,\delta) + \sigma R - \sigma\delta G\right\} \tag{15}$$

which is bounded by $K_4 2^{n\upsilon f_2(\sigma,\delta)}$ provided

$$f_3(\sigma,\delta) + \delta R - \sigma\delta G < 0$$

$$\delta R + f_2(\sigma,\delta) < 0 \,. \tag{16}$$

Now the last two constraints of the set

$$\sigma\delta G + f_1(\sigma\delta) < 0 \tag{17a}$$

$$\delta R + f_2(\sigma,\delta) < 0 \tag{17b}$$

$$f_3(\sigma,\delta) - f_2(\sigma,\delta) - \sigma\delta G \leq 0 \tag{17c}$$

imply the second constraint in (13), so that (17) is equivalent to (11), (13), and (16). Similarly, the last two constraints of the set

$$\sigma\delta G + f_1(\sigma\delta) < 0 \tag{18a}$$

$$\sigma\delta G + f_2(\sigma,\delta) - f_3(\sigma,\delta) < 0 \tag{18b}$$

$$f_3(\sigma,\delta) + \delta R - \sigma\delta G < 0 \tag{18c}$$

imply the second constraint of (11), so that (18) is equivalent to (11), (14), and (16). We thus get the bound

$$U(\upsilon) \leq \begin{cases} K_5 2^{n\upsilon f_2(\sigma,\,\delta)} & \text{if (17) holds} \qquad (19a) \\ K_6 2^{n\upsilon[f_3(\sigma,\,\delta) - \sigma\delta G]} & \text{if (18) holds} \qquad (19b) \end{cases}$$

if $\sigma \geq 0$, $\delta \in [0, 1]$, where the second exponent was obtained with the help of the inequality of (18b).

Substituting (9) into (4) we get that if $\sigma \geq 0$, $\delta \in [0, 1]$ then

$$P_f(t) \leq \exp_2 nt\,[f_3(\sigma, \delta) + \delta R - \delta\sigma G] \cdot$$

$$\cdot \sum_{m=0}^{t} \exp_2 nm\,[\sigma\delta G - f_3(\sigma, \delta) + f_2(\sigma, \delta)] .$$

Therefore

$$P_f(t) \leq \begin{cases} K_7 2^{nt[f_2(\sigma,\,\delta) + \delta R]} & \text{if (21) holds} \qquad (20a) \\ K_8 2^{nt[f_3(\sigma,\,\delta) + \delta R - \delta\sigma G]} & \text{if (21) does not hold} \qquad (20b) \end{cases}$$

where

$$\sigma\delta G - f_3(\sigma, \delta) + f_2(\sigma, \delta) > 0 . \qquad (21)$$

Substituting finally (9) into (5) we get for $\sigma \geq 0$, $\gamma \in [0, 1]$

$$\overline{N^\gamma} \leq \sum_{t=0}^{\infty} \sum_{m=t}^{\infty} \exp_2 n \left\{ \gamma\sigma(m-t)G + (m-t) f_1(\sigma\delta) + t f_2(\sigma, \gamma) + \gamma t R \right\} +$$

$$+ \sum_{m=0}^{\infty} \sum_{t=m+1}^{\infty} \exp_2 n \left\{ \gamma\sigma(m-t)G + (t-m) f_3(\sigma, \gamma) + m f_2(\sigma, \gamma) + \gamma t R \right\} \qquad (22)$$

The first sum in (22) converges provided

$$\gamma\sigma G + f_1(\sigma\gamma) < 0$$

$$\gamma R + f_2(\sigma, \gamma) < 0$$

while the second sum converges provided

$$f_3(\sigma, \gamma) + \gamma R - \gamma\sigma G < 0$$

$$\gamma R + f_2(\sigma, \gamma) < 0 \ .$$

We therefore conclude that

$$\overline{N^{\gamma}} \leq K_9 \tag{23}$$

where K_9 is finite if

$$\gamma\sigma G + f_1(\sigma\gamma) < 0 \tag{24a}$$

$$\gamma R + f_2(\sigma, \gamma) < 0 \tag{24b}$$

$$f_3(\sigma, \gamma) + \gamma R - \gamma\sigma G < 0 \ . \tag{24c}$$

We conclude this section by upper bounding Q(a). We do so using a difference equation method pioneered by Zigangirov [7].

Consider the partial tree of Figure 2 all of whose branches are in the incorrect subset, with $d = 2^k$ branches leaving all but the first node (in Figure 2, $k = 2$). Let β be the cumulative likelihood value of the first node and Δ the likelihood of the branch emanating from it. Let $F_a(\beta)$ be the probability that at least one of the nodes of the tree of Figure 2 has a cumulative likelihood that exceeds the value a, given that the initial node had likelihood β. $F_a(\beta)$ then satisfies the difference equation

$$1 - F_a(\beta) = \sum_{\Delta} P(\Delta) \left[1 - F_a(\beta + \Delta)\right]^d \tag{25}$$

where $P(\Delta)$ denotes the probability that a branch has likelihood Δ, and by definition

$$F_a(\beta) = 1 \text{ for } \beta \geq a \ . \tag{26}$$

Because $\sum P(\Delta) = 1$, it follows from (25) that

$$F_a(\beta) \leq d \sum_{\Delta} P(\Delta)\, F_a(\beta + \Delta) . \tag{27}$$

Let $F^*(\beta)$ be any function satisfying (27) such that

$$F^*(\beta) \geq 1 \qquad \text{for } \beta \geq a \tag{28}$$

then it is well known that [see [8], pp. 281-282]

$$F_a(\beta) \leq F^*(\beta) . \tag{29}$$

We will try $F^*(\beta) = 2^{s[\beta - a]}$ with s chosen so that (27) is satisfied with equality. Thus we desire

$$2^{s[\beta-a]} = d \sum P(\Delta)\, a^{s[\beta + \Delta - a]}$$

or

$$1 = d \sum_{\Delta} P(\Delta)\, 2^{s\Delta} . \tag{30}$$

Using the metric formula (2) and the fact that $d = 2^{nR}$, (30) becomes

$$2^{n[sG-R]} = \left\{ \sum_{x,\,y} w(y)\, r(x) \left[\frac{w(y/x)}{w(y)} \right]^{s} \right\}^{n}$$

or

$$sG - R - f_1(1-s) = 0 . \tag{31}$$

The relation between $Q(a)$ and $F_a(\beta)$ is, of course,

$$Q(a) = 1 - [1 - F_a(0)]^{d-1} \leq (d-1)\, F_a(0) \tag{32}$$

so that

$$Q(a) \leq (d-1)\, 2^{-sa} \tag{33}$$

where s is the maximum value satisfying (31).

4. Optimization of the Random Coding Bounds

In this section we will choose the various values of G that optimize the bounds on $U(\upsilon)$, $P_f(t)$, $\overline{N^\gamma}$, and $Q(a)$. These should be expected to be different for the four cases. In the next section we will choose the best values of σ and δ for fixed G.

Our analysis will presuppose a constant value of the source distribution $r(x)$. Most channels of interest are symmetrical and for them the best $r(x)$ is uniform. For other channels $r(x)$ should be optimized, but we will not concern ourselves with this problem (see Chapter 7 of [1]). In fact, in general different distributions $r(x)$ would optimize the bounds on $P_f(t)$, $U(\upsilon)$, $Q(a)$, and $\overline{N^\gamma}$!

First, consider the bound (19a). Our approach to its optimization is to choose for a fixed δ values of σ and G that will allow satisfaction of (17) by the maximum value of R. In this way a parametric relation (in δ) between R and the exponent $-f_2(\sigma(\delta), \delta)$ will be obtained. If an increase in R will lead to a decrease in $-f_2(\sigma(\delta), \delta)$ the bound will be optimized.

Now R is maximized (see (17b)) by maximizing $-f_2(\sigma, \delta)$ and then choosing G that would satisfy (17a) and (17c). Straightforward calculus shows that the desired value is

$$\sigma = \frac{1}{1+\delta} \tag{34}$$

so that the choice of G is

$$\frac{1+\delta}{\delta}\left[f_3\left(\frac{1}{1+\delta}, \delta\right) - f_2\left(\frac{1}{1+\delta}, \delta\right)\right] \leq G \leq -\frac{1+\delta}{\delta} f_1\left(\frac{\delta}{1+\delta}\right) \tag{35}$$

We show in Appendix II that indeed the righthand side of (35) is at least as large as the lefthand side. It is interesting to note from (7) that

$$-f_2\left(\frac{1}{1+\delta},\ \delta\right) = -\log \sum_y \left[\sum_x w(y/x)^{\frac{1}{1+\delta}} r(x)\right]^{1+\delta} = E_o(\delta) \qquad (36)$$

where $E_o(\delta)$ is the well known exponent function of Gallager [9]. The desired maximal value of R is then $\frac{1}{\delta} E_o(\delta)$. Since δ is restricted to the range [0, 1], it remains to treat the case of $R < E_o(1)$.

Since the maximum of $E_o(\delta)$ for $\delta \in [0, 1]$ is $E_o(1)$, then the exponent will be $E_o(1)$ provided G satisfies (35) with $\delta = 1$.

We must next check if better results cannot be obtained with bound (19b). It follows from (18b) and (18c) that choosing σ to maximize $-f_2(\sigma, \delta)$ will allow simultaneous maximization of R and of the exponent $\sigma\delta G - f_3(\sigma, \delta)$ provided (18a) can be satisfied. However, (18b) will in any case force the exponent of (19b) not to exceed that of (19a). We state our result as a theorem.

Theorem 1

For $R \in [E_o(1), C]$ and a code of branch constraint length υ the probability of undetectable error is bounded by

$$U(\upsilon) \leq K_5 2^{-n\upsilon E_o(\delta)} \qquad (37)$$

where $\delta \in [0, 1]$ is the solution of

$$R = \frac{1}{\delta} E_o(\delta) \qquad (38)$$

and K_5 is finite if

$$\frac{1+\delta}{\delta}\left[E_o(\delta) + f_3\left(\frac{1}{1+\delta},\ \delta\right)\right] \leq G \leq -\frac{1+\delta}{\delta} f_1\left(\frac{\delta}{1+\delta}\right) \qquad (39)$$

For $R \in [0, E_o(1)]$, (37) holds if $\delta = 1$ and (39) is satisfied.

It follows from Appendix II that the two extreme sides of (39) are equal if and only if

$$\sum_{x} \left(\frac{w(y/x)}{w(y)}\right)^{\frac{1}{1+\delta}} r(x) = \text{const} \quad \text{for all } y \tag{40}$$

This is actually the case for the BSC when r(x) is uniform, but is not true in general. If (40) holds, then (39) reduces to the "usual" choice ([1], p. 360)

$$G = \frac{1}{\delta} E_o(\delta) . \tag{41}$$

We show in Appendix II that

$$\frac{1+\delta}{\delta}\left[f_3\left(\frac{1}{1+\delta}, \delta\right) + E_o(\delta)\right] < \frac{1}{\delta} E_o(\delta) < -\frac{1+\delta}{\delta} f_1\left(\frac{\delta}{1+\delta}\right) \tag{42}$$

so that Theorem 1 constitutes a real strengthening of the previous results that provides us with a welcome leeway for choosing G.

We next turn to the optimization of the bound (20). In (20a), for a fixed η and R (for reasons that will become apparent in the next section, we are using the parameter η instead of δ) one wishes to select σ so as to maximize $-f_2(\sigma, \eta)$ and then choose G sufficiently large to satisfy (21). This implies that $\sigma = \frac{1}{1+\eta}$ and

$$G \geq \frac{1+\eta}{\eta}\left[f_3\left(\frac{1}{1+\eta}, \eta\right) - f_2\left(\frac{1}{1+\eta}, \eta\right)\right] .$$

As a result of this choice,

$$P_f(t) \leq K_7 \, 2^{-nt[E_o(\eta) - \eta R]} \tag{43}$$

As is well known, the exponent of (43) is maximized by the value of η satisfying

$$R = E_o'(\eta) . \tag{44}$$

It is immediately obvious that (20b) is optimized by the same value of σ and by G satisfying (42) with equality. This choice gives the same exponent. We then get the following

Theorem 2

For $R \in [E_o'(1), C)$, the probability of failure of branch order $t \leq \upsilon$ is bounded by

$$P_f(t) \leq K_7 \, 2^{-nt[E_o(\eta) - \eta R]} \tag{45}$$

where η satisfies (44). K_7 is finite provided

$$G \geq \frac{1+\eta}{\eta} \left[E_o(\eta) + f_3\left(\frac{1}{1+\eta}, \eta\right)\right] . \tag{46}$$

For $R \in (0, E_o'(1))$, we choose $\eta = 1$ in both (45) and (46).

The above theorem shows that if G satisfies (46) then the so called random block coding exponent applies to the probability of failure. Again, if (40) holds, the righthand side of (46) reduces to the usual choice of $G = \frac{1}{\eta} E_o(\eta)$ (see [1] p. 361). Because of the left inequality in (42), Theorem 2 strengthens the previously published results.

Our next topic is to maximize the value of R for which $\overline{N^\gamma}$ is finite where $\gamma \in (0, 1]$. It follows from (24c) that G must be made as large as (24a) allows. Hence R must satisfy

$$\gamma R < \max_{\sigma > 0} \min \left\{-f_2(\sigma, \gamma), -f_1(\sigma\gamma) - f_3(\sigma, \gamma)\right\} . \tag{47}$$

But, as already pointed out, $-f_2(\sigma, \gamma)$ is maximized by $\sigma = \frac{1}{1+\gamma}$ and

$$-f_2\left(\frac{1}{1+\gamma}, \gamma\right) \leq -f_1\left(\frac{\gamma}{1+\gamma}\right) - f_3\left(\frac{1}{1+\gamma}, \gamma\right) . \tag{48}$$

We thus have the following

Theorem 3(*)

For $\gamma \in (0, 1]$, $\overline{N^{\gamma}}$ is finite provided

$$R < \frac{1}{\gamma} E_o(\gamma) \tag{49}$$

and

$$\frac{1+\gamma}{\gamma}\left[f_3\left(\frac{1}{1+\gamma},\ \gamma\right) + E_o(\gamma)\right] \leq G \leq -\frac{1+\gamma}{\gamma} f_1\left(\frac{\gamma}{1+\gamma}\right). \tag{50}$$

We see that Theorem 3 represents the same strengthening of the usual bound (see [1], p. 363) as Theorem 1 did. In particular the usual choice $G = \frac{1}{\gamma} E_o(\gamma)$ is within the range of the interval (50) that has non-zero length whenever (40) does not hold.

Let us finally consider the bound (33). In Appendix III we have shown that $f_1(\lambda)$ is a convex function with

$$f_1(0) = 0,\ f_1(1) \leq 0.$$

Thus Figure 3 represents the graphical solution to the problem of maximizing s* that satisfies (31). As is intuitively obvious, s* is a monotonically increasing function of G. s* = 1 for G = R. We summarize our conclusions in

Theorem 4

The probability Q(a) that the likelihood of some path in the incorrect subset exceeds a is bounded by

$$Q(a) \leq (2^{nR} - 1)\, 2^{-s^* a} \tag{51}$$

where s* is the maximum of at most two solutions of the equation

$$G = \frac{1}{s}\left[R + f_1(1-s)\right]. \tag{52}$$

* We call the reader's attention to the fact that Theorem 3 does not imply that if (49) holds then the upper bound on N^{γ} is finite only if (50) holds as well. When G = R, the conditions (24) reduce with the help of (42) to the usual condition (49).

Let

$$G^+ = \lim_{s \to \infty} \frac{1}{s} f_1(1-s) .$$

If $R \geq -f_1(1)$ there is a unique value G^- such that (53) has two positive solutions for $G \in (G^-, G^+)$ and no solution for $G < G^-$. If $R < -f_1(1)$ then exactly one positive solution exists for all $G < G^+$. s^* is a monotonically increasing function of $G \in (G^-, G^+)$.

The reader should note that Theorems 1, 2, and 3 have a somewhat different status than Theorem 4. There is definite practical value in setting G so as to minimize $P_f(t)$ and $U(\upsilon)$ for fixed t and υ, and to maximize the Pareto exponent for a given rate R. On the other hand, it would be foolish to blindly increase G just to minimize the bound on Q(a) for fixed $\underline{a}$. The latter is an arbitrary parameter which is used to determine a back-stop before which decoding information can safely be released to the user. One might therefore wish to answer the following question.

> Given a prescribed average lag of released information behind maximum tree penetration by the decoder (information is assumed here to be released in accordance with the rule of the next to last paragraph of Section 1), how shall G and a be chosen so as to minimize the probability of error Q(a)?

To answer the above question, note that the expected penetration depth in branches necessary to achieve the likelihood increase a is given by

$$\overline{m} = \frac{a}{nE\left[\log \frac{w(y/x)}{w(y)} - G\right]} = \frac{a}{n\left[I(X;Y) - G\right]} \qquad (53)$$

since the denominator is the expected likelihood increase per branch. Now from (51),

$$-\log Q(a) \geq -nR + s^*a = -nR + \overline{m}n\left[s^*I(X;Y) - R - f_1(1-s^*)\right]$$

where the value of a was given by (53) and that of G by (52).

It follow therefore, that we wish to choose s* so as to maximize

$$s^*I(X;Y) - f_1(1-s^*)$$

which is equivalent to choosing the largest s* such that

$$I(X;Y) = -f_1'(1-s^*)$$

But of Theorem III.1, the desired s*=1, so the best choice is G = R. We then get

Theorem 5

To minimize the random coding bound on the probability of released information error, $Q^*(\overline{m})$, for a prescribed average lag $\overline{m}$ of released information behind maximum tree penetration by the decoder, the bias G should be chosen to equal R. Then

$$Q^*(\overline{m}) \leq (2^{nR} - 1)\, 2^{-\overline{m}n\left[I(x;y) - R\right]}$$

provided the likelihood decision threshold is set to

$$a = \overline{m}n\left[I(\ ;\ ) - R\right]$$

It is worth noting that because of (38), (39), (42), (49), and (50), the choice G = R allows for simultaneous optimization of the bounds on the Pareto exponent, $U(\upsilon)$, and $Q^*(\overline{m})$.

5. The Random Coding Bounds for Arbitrary Values of G

In sequential decoding one ordinarily wishes to choose all parameters so as to maximize the Pareto exponent α which determines the amount of decoding effort. This is especially true in the range $\alpha \in (0, 1]$. Comparing Theorems 1 and 3 we see that the bounds on $U(\upsilon)$ and $\overline{N^{\gamma}}$ require the same optimal choice of G, namely in the range (50). From Theorem 5 we see that to minimize $Q^*(m)$, G ought to be selected equal to R. To minimize $P_f(t)$, G ought to be selected within the range (46) whose lower limit is formally identical with that of (39) which minimizes $U(\upsilon)$. However, the values of the parameters η and δ appropriate for (39) and (46) are different. For (39), η satisfies $R = E_o'(\eta)$, while for (46), $R = \frac{1}{\delta} E_o(\delta)$. Because of the well-known concave nature of $E_o(\lambda)$,

$$E_o'(\lambda) \leq \frac{1}{\lambda} E_o(\lambda) \tag{55}$$

since $E_o(0) = 0$. Hence

$$\eta \leq \delta . \tag{56}$$

Thus for some channels at least [certainly for all channels satisfying (40), such as the BSC]

$$\frac{1+\eta}{\eta}\left[E_o(\eta) + f_3\left(\frac{1}{1+\eta},\ \eta\right)\right] > -\frac{1+\gamma}{\gamma} f_1\left(\frac{\gamma}{1+\gamma}\right) \tag{57}$$

so that there is no value of G that would simultaneously optimize the bounds on $U(\upsilon)$ and $P_f(t)$.

We already remarked in the preceding section that for non-symmetrical channels a different input distribution r(x) optimizes the different performance parameters. As a consequence, the algorithms of the present section will not be optimal for such channels.

A. Bounds on U(υ) when (39) not satisfied

Let δ_R satisfy $R = \frac{1}{\delta_R} E_o(\delta_R)$ where $R \geq E_o(1)$, otherwise $\delta_R \triangleq 1$.

Because of (57), the more interesting case of violation of (39) is that

$$I(X;Y) \geq G > -\frac{1+\delta_R}{\delta_R} f_1\left(\frac{\delta_R}{1+\delta_R}\right). \tag{58}$$

We will now minimize the bound (19) for this case. (It is shown in Appendix III, Theorem III-1 that $-f'(0) = I(X;Y)$. Therefore, unless the lefthand side of (59) holds, neither (17a) nor (18a) can be satisfied.)

Lemma 1

The exponent of the upper bound on U(υ) is minimized by some $\delta \in (\delta^*, \delta_R)$ where δ^* is the unique solution of

$$\frac{\delta^*}{1+\delta^*} G + f_1\left(\frac{\delta^*}{1+\delta^*}\right) = 0 . \tag{59}$$

Proof

Because of the convex nature of $f_1(\lambda)$ and inequality (58),

$$\rho^* \triangleq \frac{\delta^*}{1+\delta^*} < \frac{\delta_R}{1+\delta_R} \tag{60}$$

so that $\delta^* < \delta_R$ as asserted. Because of (59), inequalities (17a) and (18a) can be satisfied for a fixed δ only if $\sigma \in (0, \frac{\rho^*}{\delta})$. Since for $\delta \in (\delta^*, \delta_R)$

$$\rho^* < \frac{\delta}{1+\delta} \tag{61}$$

then because of the concave nature of $-f_2(\sigma, \delta)$ as a function of σ, the former is maximized over $\sigma \in (0, \frac{\rho^*}{\delta}]$ by the value $\sigma = \frac{\rho^*}{\delta}$. Therefore, from (17), (18), and (19) the maximum achievable exponent cannot exceed $-f_2(\frac{\rho^*}{\delta}, \delta)$. But for $\delta < \delta^*$, $-f_2(\frac{\rho^*}{\delta}, \delta) < -f_2(\frac{\rho^*}{\delta^*}, \delta^*)$ [see Appendix III, Theorem III-3],

and exponent $-f_2(\frac{\rho^*}{\delta^*}, \delta^*)$ is achievable since the assignment $\sigma = \frac{\rho^*}{\delta^*}$, $\delta = \delta^*$ satisfies (17a) because of (59), (17c) because of (35), and (17b) because by the concavity of $E_o(\delta)$,

$$R < \frac{1}{\delta^*} E_o(\delta^*) = -\frac{1}{\delta^*} f_2\left(\frac{\rho^*}{\delta^*}, \delta^*\right). \tag{62}$$

Therefore only $\delta \geq \delta^*$ need be considered.

Adding (18b) and (18c) results in (17b), and for $\delta \in (\delta_R, 1)$

$$-\frac{1}{\delta} f_2(\sigma, \delta) \leq -\frac{1}{\delta} f_2\left(\frac{1}{1+\delta}, \delta\right) < -\frac{1}{\delta_R} f_2\left(\frac{1}{1+\delta_R}, \delta_R\right) = R$$

so that neither (17) nor (18) can be satisfied. We thus conclude that $\delta < \delta_R$. Q.E.D.

Let us now pick $\delta \in (\delta^*, \delta_R)$ and try to find the value of σ maximizing the exponent. If

$$R \leq -\frac{1}{\delta} f_2\left(\frac{\rho^*}{\delta}, \delta\right) \tag{63}$$

does not hold then that value of δ is inadmissible since neither (17b) nor (18b) and (18c) can be satisfied for any σ in the allowed range $(0, \frac{\rho^*}{\delta})$ [see the proof of the preceding Lemma]. Assume therefore that (63) does hold, and suppose that

$$-f_1(\rho^*) \geq f_3\left(\frac{\rho^*}{\delta}\right) - f_2\left(\frac{\rho^*}{\delta}, \delta\right). \tag{64}$$

In this case the choice $\sigma = \frac{\rho^*}{\delta}$ satisfies (17a) and (17c). Since (17b) is also satisfied and any smaller value of σ decreases $-f_2(\sigma, \delta)$, the exponent is equal to $-f_2\left(\frac{\rho^*}{\delta}, \delta\right)$.

Next, suppose that (64) does not hold and let σ_1 be the largest value in $(0, \rho/\delta^*)$ such that

$$\sigma_1 \delta G = f_3(\sigma_1, \delta) - f_2(\sigma_1, \delta) . \tag{65}$$

If (17b) holds with $\sigma = \sigma_1$, then the largest conceivable exponent obtainable from bound (19a) is $-f_2(\sigma_1, \delta)$ which is at most as large as the exponent from (19b) obtainable for some $\sigma \in (\sigma_1, \rho^*/\delta)$.

Thus if (64) does not hold, we need consider only the bound (19b).

Let $\sigma_G(\delta)$ be the unique value satisfying

$$\delta G = f_3'(\sigma, \delta) \tag{66}$$

that exists provided $G < f_3'(\infty, \delta)$ [see Appendix III, Theorem III-4]. If (66) cannot be satisfied, we set $\sigma_G(\delta) = \infty$. Suppose

$$\frac{\rho^*}{\delta} \leq \sigma_G(\delta) . \tag{67}$$

Then with $\sigma = \frac{\rho^*}{\delta}$, (18a) and (18b) are satisfied and if

$$R \leq \frac{1}{\delta} [\rho^* G - f_3(\rho^*/\delta, \delta)] \tag{68}$$

then $\rho^* G - f_3(\rho^*/\delta, \delta)$ is the best obtainable exponent for that value of δ. If (68) is not satisfied, δ is not admissible. If (67) does not hold, and $\sigma_1 \geq \sigma_G(\delta)$, then the best attainable exponent is $-f_2(\sigma_1, \delta)$ provided (18c) holds, while if $\sigma_G(\delta) \in (\sigma_1, \rho^*/\delta)$ then the best exponent is $\sigma_G(\delta)\, \delta G - f_3(\sigma_G(\delta), \delta)$, provided (18c) holds. If (18c) does not hold, δ is not admissible.

We can now state an algorithm that will obtain the best exponent for the upperbound on $U(\upsilon)$ for a fixed R and G satisfying (59).

1. Find the interval (δ^*, δ_R) and ρ^*.
2. Pick $\delta \in (\delta^*, \delta_R)$ and check if (63) holds.

If it does not, δ is not admissible. Otherwise continue.

3. If (64) holds, let

$$E_u(\delta) = -f_2(\rho^*/\delta, \delta)$$

4. If (64) does not hold, check if

$$\delta G - f_3'(\rho^*/\delta, \delta) \geq 0 \tag{69}$$

If (69) holds and (18c) is not satisfied with $\sigma = \rho^*/\delta$ then δ is not admissible. Otherwise

$$E_u(\delta) = \rho^* G - f_3(\rho^*/\delta, \delta)$$

5. If (69) does not hold, find the largest $\sigma_1 \in (0, \rho^*/\delta)$ satisfying (65) and check whether

$$\delta G - f_3'(\sigma_1, \delta) < 0 \tag{70}$$

If (70) holds and (18c) is not satisfied with $\sigma = \sigma_1$ then δ is not admissible. Otherwise,

$$E_u(\delta) = -f_2(\sigma_1, \delta)$$

6. If (70) does not hold, find $\sigma_G(\delta)$ satisfying (66) [necessarily $\sigma_G(\delta) \in (\sigma_1, \rho^*/\delta)$]. If (18c) does not hold with $\sigma = \sigma_G(\delta)$ then δ is not admissible, otherwise

$$E_u(\delta) = \sigma_G(\delta)\delta G - f_3(\sigma_G(\delta), \delta)$$

7. Repeat from step 2 on, so as to obtain a plot of $E_u(\delta)$ for all admissible values $\delta \in (\delta^*, \delta_R)$. The maximum of this plot is the desired exponent.

We expect that (δ^*, δ_R) will contain only one sub-interval of admissible values of δ, and that over that sub-interval $E_u(\delta)$ will be unimodal.

Let us next consider the case

$$0 \leq G < \frac{1+\delta_R}{\delta_R}\left[f_3\left(\frac{1}{1+\delta_R}, \delta_R\right) - f_2\left(\frac{1}{1+\delta_R}, \delta_R\right)\right]. \tag{71}$$

Lemma 2

The exponent of the upper bound on $U(\upsilon)$ is minimized by some $\delta \in (\delta_1, \delta_R)$ where δ_1 is either the largest $\delta \in (0, \delta_R)$ such that

$$\frac{\delta_1}{1+\delta_1} G = f_3\left(\frac{1}{1+\delta_1}, \delta_1\right) - f_2\left(\frac{1}{1+\delta_1}, \delta_1\right) \tag{72}$$

or is 0 if (72) cannot be satisfied.

Proof

Let σ_1 be the "best" value of σ for some $\delta \in (0, 1)$. If δ is admissible, then

$$R \leq -\frac{1}{\delta} f_2(\sigma_1, \delta) < -\frac{1}{\delta} f_2\left(\frac{1}{1+\delta}, \delta\right). \tag{73}$$

But the righhand side of (73) is a decreasing function of δ, so if $\delta_R < 1$, the lefthand inequality in (73) can hold only if $\delta < \delta_R$.

Next, let δ_1 be as defined in the Lemma. Because (35) holds, then for $\delta = \delta_1$ and $\sigma = \frac{1}{1+\delta_1}$, all the conditions (17) are satisfied so that the exponent for this value of R and G is at least $-f_2\left(\frac{1}{1+\delta_1}, \delta_1\right)$. Because $-f_2\left(\frac{1}{1+\delta}, \delta\right)$ is an increasing function of δ, then for all $\delta < \delta_1$ the exponent is smaller than $-f_2\left(\frac{1}{1+\delta_1}, \delta_1\right)$ so only $\delta > \delta_1$ need be considered.

Q.E.D.

It follows from the definition of δ_1, and from (71) that for all $\delta \in (\delta_1, \delta_R)$,

$$\frac{\delta}{1+\delta} G < f_3\left(\frac{1}{1+\delta}, \delta\right) - f_2\left(\frac{1}{1+\delta}, \delta\right) \leq -f_1\left(\frac{\delta}{1+\delta}\right). \tag{74}$$

Let σ_2 be the largest value of $\sigma \in \left[0, \frac{1}{1+\delta}\right]$ such that

$$\sigma\delta G \leq f_3(\sigma, \delta) - f_2(\sigma, \delta) \tag{18b}$$

holds with equality and let σ_1 be the smallest value of $\sigma \in \left[\frac{1}{1+\delta}, \infty\right)$ for which (18b) holds with equality. From (74) it follows that (18b) holds for all $\sigma \in (\sigma_2, \sigma_1)$ and that $\frac{1}{1+\delta} \leq \rho^*/\delta$. Therefore (18a) and (18b) both hold for $\sigma \in (\sigma_2, \sigma_3)$ where

$$\sigma_3 = \min \{\sigma_1, \rho^*/\delta\} . \tag{75}$$

Let $\sigma_G(\delta)$ be as defined in (66). If $\sigma_G(\delta) \in (\sigma_2, \sigma_3)$ and

$$\delta R \leq \sigma_G(\delta)\, \delta G - f_3(\sigma_G(\delta), \delta) \tag{76}$$

then the righthand side of (76) is the exponent. If $\sigma_G(\delta) \leq \sigma_2$ and

$$\delta R \leq \sigma_2\, \delta G - f_3(\sigma_2, \delta) \tag{77}$$

then the righthand side of (77) is the exponent.

If $\sigma_G(\delta) \geq \sigma_3$ and

$$\delta R \leq \sigma_3\, \delta G - f_3(\sigma_3, \delta) \tag{77a}$$

then the righthand side of (77a) is the exponent. If neither of the three cases holds, δ is inadmissible. We therefore get the following algorithm.

1. Find the interval (δ_1, δ_2) and ρ^*.
2. Pick $\delta \in (\delta_1, \delta_R)$ and compute σ_1, σ_2, σ_3.
3. If $\delta G - f_3'(\sigma_3, \delta) \geq 0$ (78)

check whether (77a) holds. If it does not, δ is inadmissible, if it does, the exponent is

$$E_u(\delta) = \sigma_3\, \delta G - f_3(\sigma_3, \delta)$$

4. If (78) does not hold and

$$\delta G - f_3'(\sigma_2, \delta) \leq 0 \tag{79}$$

check whether (77a) holds. If it does not, δ is inadmissible, if it does, the exponent is

$$E_u(\delta) = \sigma_2 \delta G - f_3(\sigma_3, \delta)$$

5. If neither (78) nor (79) hold, determine $\sigma_G(\delta)$ satisfying (66). If (76) does not hold, δ is inadmissible, if it does hold then the exponent is

$$E_u(\delta) = \sigma_G(\delta)\, \delta G - f_3(\sigma_G(\delta), \delta)$$

6. Repeat from step 2 on so as to obtain a plot of $E_u(\delta)$ for all admissible values $\delta \in (\delta_1, \delta_R)$. The maximum is the desired exponent.

B. Bound on $P_f(t)$ when (46) not satisfied

We will now see how to optimize bound (20) for a fixed G less than the righthand side of (46).

Lemma 3

If when η satisfies (44), the inequality (46) does not hold, then the value of δ optimizing the bound (20) on $P_f(t)$ is within the interval (δ_m, δ_M) where $\delta_m(\delta_M)$ is the largest $\delta < \eta$ (smallest $\delta \geq \eta$) such that

$$\frac{\delta}{1+\delta} G - f_3\left(\frac{1}{1+\delta}, \delta\right) + f_2\left(\frac{1}{1+\delta}, \delta\right) = 0 \tag{77}$$

or is equal to 0 (equal to 1) whichever is larger (smaller).

Proof

First note that only $\delta \in [0, 1]$ are admissible by the bound. If $\delta \in [0, \delta_m)$ $\left(\delta \in (\delta_M, 1]\right)$ then because of the concave nature of $-f_2\left(\frac{1}{1+\delta}, \delta\right) - \delta R$ the largest value of the exponent cannot exceed

$$-f_2\left(\frac{1}{1+\delta_m}, \delta_m\right) - \delta_m R \qquad \left(-f_2\left(\frac{1}{1+\delta_M}, \delta_M\right) - \delta_M R\right) \tag{78}$$

otherwise the optimal exponent for any G would not be achieved at η. However, because ot (77) the values (78) are achievable with the given G and so the optimizing $\delta \in [\delta_m, \delta_M]$. Q.E.D.

Consider now $\delta \in [\delta_m, \delta_M]$ fixed. We will see how to find the value of $\sigma \geq 0$ that optimizes the exponent in (20). We will use the fact that $-f_2(\sigma, \delta)$ and $-f_3(\sigma, \delta)$ are both concave functions of σ that are positive for some interval $(0, \sigma_M)$ [see Appendix III, Theorems III-2 and III-4], and that $-f_2(\sigma, \delta)$ is maximized at $\sigma = \frac{1}{1+\delta}$. For $\delta \in (\delta_m, \delta_M)$, the left-hand side of (77) is negative. If $\sigma_G(\delta)$ maximizes $\sigma\, \delta\, G - f_3(\sigma, \delta)$, then there are two cases. If

$$\sigma_G(\delta)\, \delta G - f_3(\sigma_G(\delta), \delta) \leq -f_2(\sigma_G(\delta), \delta) \tag{79}$$

then because of the concave nature of $-f_2$ and $-f_3$, the best exponent $E_f(\delta)$ is given by

$$E_f(\delta) = \sigma_G(\delta)\, \delta\, G - f_3(\sigma_G(\delta), \delta) - \delta R . \tag{80}$$

On the other hand, if (79) does not hold, and $\sigma_G(\delta) < \frac{1}{1+\delta}$, then there exists a unique $\sigma_4 \in \left(\sigma_G(\delta), \frac{1}{1+\delta}\right)$ such that

$$\sigma_4\, \delta G - f_3(\sigma_4, \delta) = -f_2(\sigma_4, \delta) \tag{81}$$

and the best exponent is

$$E_f(\delta) = -f_2(\sigma_4, \delta) - \delta R . \tag{82}$$

Of course, if $\sigma_G(\delta) > \frac{1}{1+\delta}$, then $\sigma_4 \in \left(\frac{1}{1+\delta}, \sigma_G(\delta)\right)$ satisfying (81) is desired.

In finding the best exponent for the upper bound on $P_f(t)$ when (46) is not satisfied, one proceeds as follows:

1. Find the interval (δ_m, δ_M)
2. Pick $\delta \in (\delta_m, \delta_M)$ and find $\sigma_G(\delta)$ satisfying (66).
3. If (79) is satisfied, $E_f(\delta)$ is given by (80). Go to step 5.
4. If (79) is not satisfied, find σ_4. $E_f(\delta)$ is given by (82).
5. Repeat from step 2 on so as obtain a plot of $E_f(\delta)$ vs. δ. The maximum of this plot is the desired exponent.

C. Pareto Exponent for Arbitrary G

Let $R \geq E_o(1)$ and let δ_R be as defined previously. We will first find the lower bound on the Pareto exponent when

$$I(X;Y) \geq G > -\frac{1+\delta_R}{\delta_R} f_1\left(\frac{\delta_R}{1+\delta_R}\right). \tag{58}$$

We wish to find the largest possible value of γ such that (24) can be satisfied for some $\gamma \geq 0$.

Lemma 4

If (59) is satisfied then the best lower bound on the Pareto exponent γ falls within the interval (δ^*, δ_R) where δ^* satisfies (59).

Proof

Since G is not chosen optimally, $\delta < \delta_R$. Since $\delta^* < \delta_R$ (see Lemma 1) we need only to show that $\gamma = \delta^*$ satisfies (24) for $\sigma = \rho^*/\delta^*$, where ρ^* was defined in (60). But that choice satisfies (59) and therefore (24a). Furthermore, by concavity of $E_o(\delta)$,

$$R + \frac{1}{\delta^*} f_2\left(\frac{\rho^*}{\delta^*}, \delta^*\right) = R + \frac{1}{\delta^*} E_o(\delta^*) < R + \frac{1}{\delta_R} E_o(\delta_R) = 0$$

so that (24b) is satisfied as well. Finally, from (59) and (48),

$$\delta^* R + f_3\left(\frac{\rho^*}{\delta^*},\ \delta^*\right) - \rho^* G = \delta^* R + f_3\left(\frac{1}{1+\delta^*},\ \delta^*\right) + f_1\left(\frac{\delta^*}{1+\delta^*}\right)$$

$$\leq \delta^* R + f_2\left(\frac{1}{1+\delta^*},\ \delta^*\right) < 0$$

so (24c) is satisfied as well. Q.E.D.

Let $\gamma \in (\delta^*, \delta_R)$. (24a) can be satisfied only with $\sigma \leq \rho^*/\gamma$. Also since $\delta^* < \gamma$, then

$$\rho^* = \frac{\delta^*}{1+\delta^*} < \frac{\gamma}{1+\gamma}$$

so that $-f_2(\sigma, \gamma)$ is maximized over $(0,\ \rho^*/\gamma]$ by $\sigma = \frac{\rho^*}{\gamma}$. Thus, if

$$R > \frac{1}{\gamma} f_2\left(\frac{\rho^*}{\gamma},\ \gamma\right) \tag{83}$$

then the Pareto exponent is less than γ. If (83) does not hold and $\sigma_G(\delta) \geq \rho^*/\gamma$ then the Pareto exponent is less than γ if (24c) is not satisfied with $\sigma = \rho^*/\gamma$, and it exceeds γ otherwise. If $\sigma_G(\delta) < \rho^*/\gamma$, let σ_5 be the unique value of $\sigma \in (\sigma_G(\delta),\ \rho^*/\gamma)$ such that

$$f_2(\sigma_5, \gamma) = f_3(\sigma_5, \gamma) - \sigma_5 \gamma G \ . \tag{84}$$

The Pareto exponent then exceeds γ if (24b) holds with $\sigma = \sigma_5$, and is less than γ otherwise.

If (59) holds, the best lower bound on the Pareto exponent is found by the following method:

1. Find (δ^*, δ_R) and ρ^*. Let $a_1 = \delta^*$, $a_2 = \delta_R$.
2. If $a_2 - a_1 < \epsilon$ exponent is at least a_1. Stop. Otherwise pick $\gamma \in (a_1, a_2)$.

If $R > -\frac{1}{\gamma} f_2(\rho^*/\gamma, \gamma)$, set $a_2 = \gamma$ and go to step 2. Otherwise continue.

3. If $\gamma G - f_3'(\rho^*/\gamma, \gamma) < 0$

go to step 4. Otherwise if (24c) is satisfied with $\sigma = \rho^*/\gamma$ set $a_1 = \gamma$. If (24c) is not satisfied, set $a_2 = \gamma$. Go to step 2.

4. Find σ_5 satisfying (84). If (24b) holds with $\sigma = \sigma_5$, set $a_1 = \gamma$. Otherwise set $a_2 = \gamma$. Go to step 2.

We will conclude this section by treating the case

$$0 < G < \frac{1+\delta_R}{\delta_R}\left[f_3\left(\frac{1}{1+\delta_R}, \delta_R\right) + E_o(\delta_R)\right] . \tag{71}$$

Lemma 5

If (71) is satisfied, then the best lower bound on the Pareto exponent γ falls within the interval (δ_1, δ_R) where δ_1 is either the largest $\delta \in (0, \delta_R)$ for which (72) holds, or is 0 if (72) cannot be satisfied.

Proof

We omit the proof which is similar to that of Lemma 2. Q.E.D.

Let $\gamma \in (\delta_1, \delta_R)$. Then (24a) is satisfied for all $\sigma \leq \rho^*/\gamma$. Furthermore, since

$$\frac{\gamma}{1+\gamma} G < f_3\left(\frac{1}{1+\gamma}, \gamma\right) - f_2\left(\frac{1}{1+\gamma}, \gamma\right) < -f_1\left(\frac{\gamma}{1+\gamma}\right)$$

then

$$\frac{\rho^*}{\gamma} > \frac{1}{1+\gamma} . \tag{85}$$

For the sake of brevity, we shall immediately describe the algorithm that obtains the best lower bound on the Pareto exponent.

1. Find (δ_1, δ_R) and ρ^*. Let $a_1 = \delta_1$, $a_2 = \delta_R$.

2. If $a_2 - a_1 < \epsilon$, the exponent is at least a_1. Stop. Otherwise, pick $\gamma \in (a_1, a_2)$. If

$$R < \frac{1}{1+\gamma} G - f_3\left(\frac{1}{1+\gamma}, \gamma\right)$$

set $a_1 = \gamma$ and go to step 2. Otherwise, continue.

3. If $\rho^* G - f_3(\rho^*/\gamma, \gamma) < -f_2(\rho^*/\gamma, \gamma)$ (86)

go to step 4. Otherwise there is a unique $\sigma_6 \in \left(\frac{1}{1+\gamma}, \rho^*/\gamma\right)$ such that

$$\gamma\sigma_6 G - f_3(\sigma_6, \gamma) = -f_2(\sigma_6, \gamma) \quad (87)$$

If (24b) is satisfied with $\sigma = \sigma_6$, set $a_1 = \gamma$. If it is not satisfied, set $a_2 = \gamma$. Go to step 2.

4. If $\gamma G - f_3'(\rho^*/\gamma, \gamma) < 0$

go to step 5. Otherwise, if (24c) is satisfied with $\sigma = \rho^*/\gamma$, set $a_1 = \gamma$. If it is not satisfied, set $a_2 = \gamma$. Go to step 2.

5. Find $\sigma_G(\gamma)$ satisfying (66). If

$$\sigma_G(\gamma)\gamma G - f_3(\sigma_G(\gamma), \gamma) > -f_2(\sigma_G(\gamma), \gamma)$$

then go to step 6. Otherwise, if (24c) is satisfied with $\sigma = \sigma_G(\gamma)$ set $a_1 = \gamma$. If it is not satisfied, set $a_2 = \gamma$. Go to step 2.

6. If $\sigma_G(\gamma) > \frac{1}{1+\gamma}$ $\left[\sigma_G(\gamma) < \frac{1}{1+\gamma}\right]$, there is a unique $\sigma_6 \in \left(\frac{1}{1+\gamma}, \sigma_G(\gamma)\right)$ $\left[\sigma_6 \in \left(\sigma_G(\gamma), \frac{1}{1+\gamma}\right)\right]$

for which (87) holds. If (24b) is satisfied with $\sigma = \sigma_6$, set $a_1 = \gamma$. If it is not satisfied, set $a_2 = \gamma$. Go to step 2.

6. Optimal Expurgated Bounds

In this section we will develop expurgated upper bounds to the probabilities of undetected error and of failure. We will use the notation of Chapter 10 of Jelinek [1]. We will limit our attention to convolutional codes and channels symmetrical from the input, so that for any given code the probability that any information sequence be incorrectly decoded is the same for all sequences. We will therefore always assume that the all-zero sequence was transmitted.

If $\underset{\sim}{\chi}$ is received, an undetected error will take place at depth 0 only if

$$L(\underset{\sim}{s}) - L^m > 0 \text{ for some } \underset{\sim}{s} \in G_o^{t+\upsilon},\ t \geq 0,\ m \geq 0. \tag{88}$$

Hence if an undetected error takes place, then

$$\sum_{t=0}^{\infty} \sum_{m=0}^{\infty} \sum_{\underset{\sim}{s} \in G_o^{t+\upsilon}} 2^{\sigma[L(\underset{\sim}{s})-L^m]} > 1 \tag{89}$$

for all $\sigma \geq 0$. Let $\theta(\underset{\sim}{\chi})$ be the undetected error indicator function for a fixed convolutional code C of constraint length ν and a received sequence $\underset{\sim}{\chi}$. Then the probability of undetected error at depth 0 is given by

$$P_C^U(e) = \underset{\sim}{E}_{\underset{\sim}{\chi}} [\theta(\underset{\sim}{\chi})] \tag{90}$$

and the probability $P\{C: P_C^U(e) > B\}$ of selecting a code from the ensemble whose undetected error probability exceeds some number B is bounded by

$$P\{C : P_C^U(e) > B\} \leq B^{-1/\rho}\, \underset{\sim}{E}_C [\underset{\sim}{E}_{\underset{\sim}{\chi}} \theta(\underset{\sim}{\chi})]^{1/\rho} \tag{91}$$

where $\underset{\sim}{E}_C$ denotes averaging over the ensemble. Thus the probability is at most 1/2 that a code will be selected whose probability of undetected error exceeds

$$B = 2^{\rho}\left\{\underset{\sim}{E}_{C}\left[\underset{\sim}{E}_{\chi}\left(\sum_{t=0}^{\infty}\sum_{m=0}^{\infty}\sum_{\underset{\sim}{s}\in G_o^{t+\upsilon}} 2^{\sigma[L(\underset{\sim}{s})-L^{m}]}\right)\right]^{1/\rho}\right\}^{\rho} \tag{92}$$

where we took into account the fact that the lefthand side of (89) exceeds $\theta(\chi)$. Let $\underset{\sim}{u}_o$ stand for the all-zero sequence. Then we can re-write (92) as

$$B = 2^{\rho}\left\{\underset{\sim}{E}\left(\sum_{t}\sum_{m} 2^{\frac{\sigma}{\rho}G(m-t-\upsilon)}\cdot\right.\right.$$

$$\left.\left.\cdot\sum_{\underset{\sim}{s}\in G_o^{t+\upsilon}}\sum_{\chi} w(\chi/\underset{\sim}{u}_o)\left[\frac{w\left(\chi^{t+\upsilon}/\underset{\sim}{x}^{t+\upsilon}(\underset{\sim}{s})\right)w(\chi)}{w(\chi^{m}/\underset{\sim}{u}_o)\,w(\chi^{t+\upsilon})}\right]^{\sigma}\right)^{1/\rho}\right\}^{\rho}$$

If $\rho \geq 1$, then Jensen's inequality yields

$$B \leq 2^{\rho}\left\{\sum_{t}\sum_{m} 2^{\frac{\sigma}{\rho}G(m-t-\upsilon)}\cdot\right.$$

$$\left.\cdot\sum_{\underset{\sim}{s}\in G_o^{t+\upsilon}}\underset{\sim}{E}_{C}\left(\sum_{\chi} w(\chi/\underset{\sim}{u}_o)\left[\frac{w(\chi^{t+\upsilon}/\underset{\sim}{x}^{t+\upsilon}(\underset{\sim}{s}))\,w(\chi)}{w(\chi^{m}/u_o)\,w(\chi^{t+\upsilon})}\right]^{\sigma}\right)^{1/\rho}\right\}^{\rho} \tag{93}$$

We now define the exponent functions

$$g_1(\sigma) = \log\sum_{y} w(y/0)^{1-\sigma}\,w(y)^{\sigma} \tag{94}$$

$$g_2(\sigma,\rho) = \rho\log\frac{1}{a}\sum_{x}\left(\sum_{y} w(y/0)^{\sigma}\,w(y/x)^{1-\sigma}\right)^{1/\rho} \tag{95}$$

$$g_3(\sigma,\rho) = \rho\log\frac{1}{a}\sum_{x}\left(\sum_{y} w(y/0)\left[\frac{w(y/x)}{w(y)}\right]^{\sigma}\right)^{1/\rho} \tag{96}$$

with whose help we can bound the expectation in (93). Denoting the latter by $F(m,t)$, we get

$$F(m,t) \leq \begin{cases} \exp_2 \frac{1}{\rho}[m\, g_2(\sigma,\rho) + (t+\upsilon-m)\, g_3(\sigma,\rho)] & \text{if } m \leq t+\upsilon \\ \exp_2 \frac{1}{\rho}[(t+\upsilon)\, g_2(\sigma,\rho) + (m-t-\upsilon)\, g_1(\sigma)] & \text{if } m \leq t+\upsilon \end{cases} \tag{97}$$

After some algebra that is identical to that used to derive (19) we finally get the bound

$$B \leq \begin{cases} K\, 2^{\upsilon g_2(\sigma,\rho)} & \text{where K is finite if (99) holds} \\ K\, 2^{\upsilon[g_3(\sigma,\rho) - \sigma G]} & \text{where K is finite if (100) holds} \end{cases} \tag{98}$$

$$\sigma G + g_1(\sigma) < 0$$

$$\rho R + g_2(\sigma,\rho) < 0 \tag{99}$$

$$g_3(\sigma,\rho) - g_2(\sigma,\rho) - \sigma G < 0$$

$$\sigma G + g_1(\sigma) < 0$$

$$\sigma G + g_2(\sigma,\rho) - g_3(\sigma,\rho) < 0 \tag{100}$$

$$g_3(\sigma,\rho) + \rho R - \sigma G < 0$$

In the bound (98) the restrictions $\sigma \geq 0$, $\rho \geq 1$ are assumed. Comparing (98) through (100) with (17) through (19) we see that both bounds have the same formal structure. We will take advantage of this when optimizing the expurgated bound.

We show in Lemma IV-1 of Appendix IV that for channels symmetric from the input, $g_2(\sigma,\rho)$ is minimized by the choice $\sigma = 1/2$. Since at least half of the codes in the ensemble have a probability of error that does not exceed B, we may conclude that

Theorem 5

For $R \in [0, -g_2(1/2, 1) = E_o(1)]$ and channels symmetric from the input there exist convolutional codes whose probability of undetected error is bounded by

$$P_u(e) \leq K\, 2^{-\upsilon[-g_2(\frac{1}{2}, \rho)]} \tag{101}$$

where $\rho \geq 1$ satisfies

$$R = -\frac{1}{\rho} g_2(1/2, \rho) \tag{102}$$

and K is finite provided G is chosen so that

$$2[g_3(1/2, \rho) - g_2(1/2, \rho)] < G < -2\, g_1(1/2) \tag{103}$$

Of course, it is necessary to show that the righthand side of (103) exceeds the lefthand side, which we do for equidistant channels in Theorem IV-1 of Appendix IV. It is interesting to point out that the expurgated exponent of Theorem 5 is the same as that obtained by Viterbi and Odenwalder [10] for maximum likelihood decoding of convolutional codes.

We next turn to the probability of failure. If $\underset{\sim}{\chi}$ is received, a failure of order t will take place at depth 0 only if

$$L(\underset{\sim}{s}) - L^m > 0 \qquad \text{for some } \underset{\sim}{s} \in \mathcal{D}_o^t \text{ and } 0 \leq m \leq t.$$

Hence if a failure takes place then

$$\sum_{m=0}^{t} \sum_{\underset{\sim}{s} \in \mathcal{D}_o^t} 2^{\sigma[L(\underset{\sim}{s}) - L^m]} > 1 \tag{104}$$

for all $\sigma \geq 0$. Letting $\varphi(\underset{\sim}{y})$ be the failure indicator function for a fixed convolutional code C, and denoting the failure probability by $P_C^f(e)$, we can conclude that (c.f. (91)) over the ensemble,

$$P\{C : P_C^f(e) > D\} \leq D^{-1/\rho} \underset{\sim}{E}_C \left[E_{\underset{\sim}{\chi}} (\underset{\sim}{\chi}) \right]^{-1/\rho} \tag{105}$$

Hence the probability is at most 1/2 that a code (of constraint length $\upsilon \geq t$) will be selected whose probability of t-order failure exceeds

$$D = 2^{\rho} \left\{ \underset{\sim}{E} \left[E_{\underset{\sim}{\chi}} \left(\sum_{m=0}^{t} \sum_{\underset{\sim}{s} \in \mathcal{V}_o^t} 2^{\sigma[L(\underset{\sim}{s}) - L^m]} \right) \right]^{1/\rho} \right\}^{\rho} \tag{106}$$

The same algebra that led from (92) to (93) leads from (106) to

$$D \leq 2^{\rho - t\sigma G} \left\{ \sum_{m=0}^{t} 2^{m\frac{\sigma}{\rho} G} \cdot \right.$$

$$\left. \cdot \sum_{\underset{\sim}{s} \in \mathcal{V}_o^t} \underset{\sim}{E}_C \left(\sum_{\underset{\sim}{\chi}} w(\underset{\sim}{\chi} / \underset{\sim}{u}_o) \left[\frac{w(\underset{\sim}{\chi}^t / \underset{\sim}{x}^t(\underset{\sim}{s}))\, w(\underset{\sim}{\chi}^m)}{w(\underset{\sim}{\chi}^m / \underset{\sim}{u}_o)\, w(\underset{\sim}{\chi}^t)} \right]^{\sigma} \right)^{1/\rho} \right\}^{\rho} \tag{107}$$

Using the functions $g_i(\sigma, \rho)$, the righthand side of (107) can be evaluated so as to yield the bound

$$D \leq 2^{\rho - t[\sigma G - \rho R - g_3(\sigma, \rho)]} \cdot$$

$$\cdot \left(\sum_{m=0}^{t} 2^{\frac{m}{\rho}[\sigma G + g_2(\sigma, \rho) - g_3(\sigma, \rho)]} \right)^{\rho} \tag{108}$$

It follows directly that if for $\sigma > 0$, $\rho \geq 1$

$$\sigma G + g_2(\sigma, \rho) - g_3(\sigma, \rho) > 0 \tag{109}$$

then

$$D \leq K\, 2^{t[\rho R + g_2(\sigma, \rho)]} \tag{110a}$$

and otherwise

$$D \leq K\, 2^{t[\rho R - \sigma G + g_3(\sigma, \rho)]} \tag{110b}$$

Again we see that the obtained bounds (110) have the same formal structure as the bounds (20) had. Since, as clearly remarked, $g_2(\sigma, \rho)$ is minimized by $\sigma = 1/2$, and $g_2(1/2, \rho)$ is convex in ρ, we can conclude with [$g_2'(1/2, \rho)$ denotes $\frac{\partial}{\partial \rho} g_2(1/2, \rho)$]

Theorem 6

For $R \in [0, -g_2'(1/2, 1)]$ and channels symmetric from the input there exist convolutional codes whose probability of failure of order t is bounded by

$$P_f(e) \leq K\, 2^{t[\mu R + g_2(1/2, \mu)]} \tag{111}$$

where $\mu \geq 1$ is the unique solution of

$$R = - g_2'(1/2, \mu) \tag{112}$$

and K is finite provided

$$G > 2\,[g_3(1/2, \mu) - g_2(1/2, \mu)] \tag{113}$$

For $R \in (-g_2'(1/2, 1),\ -g_2(1/2, 1) = E_o(1)]$, bound (111) holds with $\mu = 1$ provided G satisfies (113) with $\mu = 1$.

It should be noted that the exponent of the bound (111) is identical to the expurgated exponent obtained previously for block codes (see Jelinek [1], p. 217).

It is further interesting to note that since

$$g_3(1/2, \mu) - g_2(1/2, \mu) \leq -g_1(1/2)$$

then the choice

$$G = -2\, g_1(1/2) = -f_1(1/2) = -2 \log \sum_y \sqrt{w(y)\, w(y/0)} \tag{114}$$

optimizes <u>simultaneously</u> both the undetected error and failure bounds for all $R \in [0, -g_2(1/2, 1)]$.

7. Expurgated Bounds for Arbitrary Values of G

In this section we describe algorithms that optimize the bounds (98) and (110) for arbitrary values of G. This we do in spite of the last assertion of the previous section, because in the range of rates of interest the G-value maximizing the Pareto exponent differs from (114). Moreover, the rate points below which optimal expurgated exponents exceed the corresponding random coding exponents for probabilities of undetected error and failure, respectively, are also in general different, so that, e.g., the random coding failure and the expurgated undetected error exponents might apply simultaneously for some rate interval [this is shown in Section 8].

Since the bounds (98) and (110) are formally identical to the bounds (19) and (20), the optimization problem ahead of us is almost identical to that of Section 5. We will therefore simply state the exponent optimization algorithms without providing a detailed justification.

Let ρ_R be the solution of

$$R = -\frac{1}{\rho} g_2(1/2, \rho) \tag{115}$$

and let us attempt to optimize the undetected error bound when

$$I(X;Y) \geq G > -2\, g_1(1/2) \tag{116}$$

The upper bound in (116) is due to the fact that $-g_1(\sigma)$ is a concave function with $-g_1(0) = 0$ and that

$$-g_1'(0) = \sum_y w(y/0) \log \frac{w(y/0)}{w(y)} = I(X;Y) \tag{117}$$

Let σ_G ($< 1/2$) be the solution of

$$G = -\frac{1}{\sigma} g_1(\sigma) \tag{118}$$

Then clearly both (99) and (100) can only be satisfied by $\sigma \in [0, \sigma_G]$. Since by Lemma IV-3, $-\frac{1}{\rho} g_2(\sigma, \rho)$ is a decreasing function of ρ, and is a concave function of σ with a maximum at $\sigma = 1/2$, then in the range $\sigma \in (0, \sigma_G]$, the inequality

$$R \leq -\frac{1}{\rho} g_2(\sigma, \rho) \tag{119}$$

can be satisfied only for some $\rho < \rho_R$. In (119) let $\sigma = \sigma_G$, and let ρ_{GR} be the value of ρ that satisfies (119) with equality. If $\rho_{GR} < 1$ then for that R-G combination an expurgated bound cannot be developed. Otherwise, we know that in any case we must choose $\rho \in (1, \rho_{GR})$ and $\sigma \in (0, \sigma_G)$ to satisfy either (99) or (100). The algorithm to find the best exponent for the case (116) is as follows:

1. Find σ_G satisfying (118), and ρ_{GR} satisfying (119) with $\sigma = \sigma_G$. If $\rho_{GR} < 1$, the exponent $E_u^{exp} = 0$, and stop.

2. If $\rho_{GR} > 1$, see whether with $\rho = \rho_{GR}$

$$-g_1(\sigma_G) \geq g_3(\sigma_G, \rho) - g_2(\sigma_G, \rho) \tag{120}$$

If so then (99) are satisfied and the exponent is

$$E_u^{exp} = -g_2(\sigma_G, \rho_{GR})$$

Stop.

3. If (120) does not hold, neither (99) nor (110) hold with $\sigma = \sigma_G$, $\rho = \rho_{GR}$. Select $\rho \in (0, \rho_{GR})$ and see whether (120) holds. If so the best exponent for that value of ρ is

$$E_u^{exp}(\rho) = -g_2(\sigma_G, \rho)$$

4. If (120) does not hold for the chosen value of ρ, check if

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \rho) \Big|_{\sigma = \sigma_G} \geq 0 \tag{121}$$

If (120) holds and (100c) is not satisfied with $\sigma = \sigma_n$, then ρ

is not admissible. Otherwise

$$E_u^{exp}(\rho) = \sigma_G G - g_3(\sigma_G, \rho)$$

5. If (121) does not hold, find the largest $\sigma_1 \in (0, \sigma_G)$ satisfying

$$\sigma_1 G + g_2(\sigma_1, \rho) - g_3(\sigma_1, \rho) = 0$$

and check whether

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \rho)\Big|_{\sigma = \sigma_1} \leq 0 \tag{122}$$

If (122) holds and (100c) is not satisfied with $\sigma = \sigma_1$, then ρ is not admissible. Otherwise

$$E_u^{exp}(\rho) = -g_2(\sigma_1, \rho)$$

6. If (122) does not hold, find σ_2 satisfying

$$G = \frac{\partial}{\partial \sigma} g_3(\sigma, \rho)$$

If (100c) does not hold with $\sigma = \sigma_2$, then ρ is not admissible. Otherwise

$$E_u^{exp}(\rho) = \sigma_2 G - g_3(\sigma_2, \rho)$$

7. Repeat from step 3 so as to obtain a plot of $E_u^{exp}(\rho)$ for all admissible values $\rho \in (1, \rho_{GR})$. The maximum of this plot is the desired exponent E_u^{exp}.

We wish next to find the exponent for undetected error when

$$G < 2\left[g_3\left(\tfrac{1}{2}, \rho_R\right) - g_2\left(\tfrac{1}{2}, \rho_R\right)\right] \tag{123}$$

In this case $\sigma_G > 1/2$. Since (119) must be satisfied, ρ can be admissible only if $\rho < \rho_R$. Let ρ_1 be the largest $\rho \in [1, \rho_R]$, if it exists, such that

$$G = 2[g_3(1/2, \rho) - g_2(1/2, \rho)] \tag{124}$$

Conditions (99) are satisfied by $\sigma = 1/2$, $\rho = \rho_1$ and so the exponent is at least $-g_2(\frac{1}{2}, \rho_1)$. For $\rho < \rho_1$, the exponent would have to be smaller, and so if $\rho_1 \geq 1$ exists, we need only consider the interval $[\rho_1, \rho_R]$. Our algorithm for finding the best exponent is as follows:

1. Find ρ_1 if it exists. If $\rho_1 \in [1, \rho_R]$ does not exist for which (124) holds, let $\rho_1 = 1$.

2. Select $\rho \in [\rho_1, \rho_R]$. Let $\sigma_2(\sigma_1)$ be the largest value of $\sigma \in [0, 1/2]$ (smallest value of $\sigma \in [1/2, \infty]$) for which

$$\sigma G = g_3(\sigma, \rho) - g_2(\sigma, \rho) \tag{125}$$

and define

$$\sigma_3 = \min(\sigma_1, \sigma_G)$$

3. If

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \rho) \Big|_{\sigma = \sigma_3} \geq 0 \tag{126}$$

check if (127) holds with $\sigma = \sigma_3$

$$\rho R \leq \sigma G - g_3(\sigma, \rho) \tag{127}$$

If (127) does not hold, ρ is inadmissible. If it does hold, then

$$E_u^{exp}(\rho) = \sigma_3 G - g_3(\sigma_3, \rho)$$

4. If (126) does not hold and

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \rho) \Big|_{\sigma = \sigma_2} \leq 0 \tag{128}$$

check if (127) holds with $\sigma = \sigma_2$. If it does not, ρ is inadmissible. If it holds, then

$$E_u^{exp}(\rho) = \sigma_2 G - g_3(\sigma_2, \rho)$$

5. If neither (126) nor (128) hold, let σ_4 be the unique value of σ satisfying

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \rho) = 0$$

If (127) holds with $\sigma = \sigma_4$ then

$$E_u^{exp}(\rho) = \sigma_4 G - g_3(\sigma_3, \rho)$$

otherwise ρ is inadmissible.

6. Repeat from step 2 on so as to obtain a plot of $E_u^{exp}(\rho)$ for all admissible values $\rho \in [\rho_1, \rho_R]$. The maximum is the desired exponent E_u^{exp}.

Finally, we wish to find the best expurgated exponent for the probability of failure when

$$G < 2[g_3(1/2, \mu_R) - g_2(1/2, \mu_R)] \tag{129}$$

where μ_R satisfies

$$R = -g_3'(1/2, \mu) \tag{130}$$

Our search algorithm is as follows

1. Find $\mu_m(\mu_M)$ the largest $\mu \in [1, \mu_R)$ (the smallest $\mu \geq \mu_R$) such that

$$G = 2[g_3(1/2, \mu) - g_2(1/2, \mu)]$$

If μ_m does not exist, set $\mu_m = 1$.

2. Choose $\mu \in (\mu_m, \mu_M)$ and find σ_1 satisfying

$$G - \frac{\partial}{\partial \sigma} g_3(\sigma, \mu) = 0$$

If

$$\sigma_1 G + g_2(\sigma_1, \mu) - g_3(\sigma_1, \mu) \leq 0 \qquad (131)$$

then

$$E_f^{ex}(\mu) = - [\mu R - \sigma_1 G + g_3(\sigma_1, \mu)]$$

3. If (131) does not hold, and $\sigma_1 < 1/2$ [$\sigma_1 > 1/2$] find unique $\sigma_2 \in (\sigma_1, 1/2)$ [$\sigma_2 \in (1/2, \sigma_1)$] such that

$$\sigma_2 G = g_3(\sigma_2, \mu) - g_2(\sigma_2, \mu)$$

Then

$$E_f^{ex}(\mu) = - [\mu R + g_2(\sigma_2, \mu)]$$

4. Repeat from step 2 on so as to obtain a plot of $E_f^{ex}(\mu)$ for $\mu \in (\mu_m, \mu_M)$. The maximum is the desired exponent E_f^{ex}.

8. Performance Curves for Gaussian Channels with Binary Inputs

In this section we first apply our exponent optimization procedures to quantized Guassian additive noise channels with binary inputs. Figure 4 concerns binary output quantization applied to a channel whose SNR is 1.5 dB per transmitted bit (this channel has $R_{comp} = .485$). In Figure 5 the quantization is optimal uniform octal and its SNR is -.3 dB per transmitted bit (here $R_{comp} = .51$). Finally, in Figure 6 the quantization is again octal, but the SNR = -2.0 dB ($R_{comp} = .375$).

Each of the figures contains curves of the failure and undetected error exponents as a function of the rate R. There are three curves of each type: the first curve corresponds to the usual choice G=R. The second curve corresponds to the choice

$$G = -\frac{1+\sigma}{\sigma} f_1\left(\frac{\sigma}{1+\sigma}\right) \tag{132a}$$

for

$$E_o(1) \leq R = \frac{1}{\sigma} E_o(\sigma) \leq C \tag{132b}$$

and

$$G = -2 f_1(1/2) = -2 g_1(1/2) \tag{133a}$$

for

$$0 \leq R \leq E_o(1), \tag{133b}$$

which is the largest possible G optimizing the undetected error exponent. The third curve corresponds to the choice

$$G = \frac{1+\eta}{\eta}\left[E_o(\eta) + f_3\left(\frac{1}{1+\eta}, \eta\right)\right] \tag{134a}$$

for

$$E_o'(1) \leq R = E_o'(\eta) \leq C, \tag{134b}$$

$$G = 2\left[E_o(1) + f_3(1/2, 1)\right] = 2\left[g_3(1/2, 1) - g_2(1/2, 1)\right] \tag{135a}$$

for

$$-g_2'(1/2, 1) \leq R < E_o'(1), \tag{135b}$$

and

$$G = 2\left[g_3(1/2, \mu) - g_2(1/2, \mu)\right] \tag{136a}$$

for

$$0 \leq R = -g_2'(1/2, \mu) < -g_2'(1/2, 1), \tag{136b}$$

which is the smallest G value possible that optimizes the failure exponent. The three figures show the performance degradations incurred incurred by a non-optimal bias assignment. Interesting is especially the substantial failure exponent degradation that results from the customary assignment G=R. The corresponding weakening of the undetected error exponents at low rates should also be noted. It is hard to say whether this phenomenon is real or simply reflects the inadequacy of the bounds.

Figure 7, the last presented in this paper, gives the Pareto exponents for the three kinds of channels (see above) when G is selected so as to optimize the Pareto or failure exponents, respectively.

Appendix I

Derivation of the Fundamental Bound

In this appendix we prove the validity of the bound (9). Let the set $\mathcal{B}$ denote either $\mathcal{D}^t$ or $\mathcal{G}^t$ (see definitions preceding (3) in Section 2) and let $\underset{\sim}{s}^*$ be the path taken by the encoder. Then assuming $\delta \in (0, 1]$,

$$E_{\underset{\sim}{Y}}\left\{\sum_{\underset{\sim}{s}\in\mathcal{B}}\left[\frac{w(\underset{\sim}{Y}^t/\underset{\sim}{X}^t(\underset{\sim}{s}))\, w(\underset{\sim}{Y}^m)}{w(\underset{\sim}{Y}^m/\underset{\sim}{X}^m(\underset{\sim}{s}^*))\, w(\underset{\sim}{Y}^t)}\right]^{\sigma}\right\} =$$

$$= E_{\underset{\sim}{Y}}\left\{\left[\frac{w(\underset{\sim}{Y}^m)}{w(\underset{\sim}{Y}^m/\underset{\sim}{X}^m(\underset{\sim}{s}^*))}\right]^{\sigma\delta}\left(\sum_{s\in\mathcal{B}}\left[\frac{w(\underset{\sim}{Y}^t/\underset{\sim}{X}^t(\underset{\sim}{s}))}{w(\underset{\sim}{Y}^t)}\right]^{\sigma}\right)^{\delta}\right\}$$

$$\le E_{\underset{\sim}{Y}}\left\{E_{\underset{\sim}{X}^m/\underset{\sim}{Y}^m}\left[\frac{w(\underset{\sim}{Y}^m)}{w(\underset{\sim}{Y}^m/\underset{\sim}{X}^m(\underset{\sim}{s}^*))}\right]^{\sigma\delta}\left(\sum_{\underset{\sim}{s}\in\mathcal{B}}E_{\underset{\sim}{X}^t}\left[\frac{w(\underset{\sim}{Y}^t/\underset{\sim}{X}^t(\underset{\sim}{s}))}{w(\underset{\sim}{Y}^t)}\right]^{\sigma}\right)^{\delta}\right\} \qquad \text{(I-1)}$$

where $E_{\underset{\sim}{Y}}$, $E_{\underset{\sim}{X}^m/\underset{\sim}{Y}^m}$, $E_{\underset{\sim}{X}^t}$ denote expectations with respect to the random vectors $\underset{\sim}{Y}$, $\underset{\sim}{X}^m$ given a fixed $\underset{\sim}{Y}^m$, and $\underset{\sim}{X}^t$ (which due to the code ensemble structure is independent of $\underset{\sim}{Y}$). Let $||\mathcal{B}||_t$ denote the number of sequences $\underset{\sim}{s}$ of length t in the set . Then if $\ell(\mathcal{D}^t) = t$ and $\ell(\mathcal{G}^t) = t-\upsilon$,

$$||\mathcal{B}||_t \le 2^{\ell(\mathcal{B})\, n\, R} \qquad \text{(I-2)}$$

where R is the rate of the code. We now have two cases.

Case I: $m \geq t$

The righthand side of (I-1) is equal to ($\underset{\sim}{y}_t^m$ denotes the sequence $y_{t+1}, \ldots, y_m$)

$$\sum_{\underset{\sim}{y}^m} w(\underset{\sim}{y}_t^m)^{\sigma\delta} \left(\sum_{\underset{\sim}{x}^m} \frac{w(\underset{\sim}{y}^m/\underset{\sim}{x}^m)\, r(\underset{\sim}{x}^m)}{w(\underset{\sim}{y}^m)} \; w(\underset{\sim}{y}^m/\underset{\sim}{x}^m)^{-\sigma\delta} \right) \cdot$$

$$\cdot \left(2^{n\ell(\beta)R} \sum_{\underset{\sim}{x}^t} r(\underset{\sim}{x}^t)\; w(\underset{\sim}{y}^t/\underset{\sim}{x}^t)^{\sigma} \right)^{\delta} =$$

$$= 2^{\delta n \ell(\beta) R} \left\{ \sum_{\underset{\sim}{y}_t^m} w(\underset{\sim}{y}_t^m) \left(\sum_{\underset{\sim}{x}_t^m} r(\underset{\sim}{x}_t^m) \right) \left(\frac{w(\underset{\sim}{y}_t^m/\underset{\sim}{x}_t^m)}{w(\underset{\sim}{y}_t^m)} \right)^{1-\sigma\delta} \right\} \cdot$$

$$\cdot \left\{ \sum_{\underset{\sim}{y}^t} w(\underset{\sim}{y}^t) \left[\sum_{\underset{\sim}{x}^t} r(\underset{\sim}{x}^t) \left(\frac{w(\underset{\sim}{y}^t/\underset{\sim}{x}^t)}{w(\underset{\sim}{y}^t)} \right)^{1-\sigma\delta} \right] \left[\sum_{\underset{\sim}{x}^t} r(\underset{\sim}{x}^t) \left(\frac{w(\underset{\sim}{y}^t/\underset{\sim}{x}^t)}{w(\underset{\sim}{y}^t)} \right)^{\sigma} \right]^{\delta} \right\}$$

$$= 2^{\delta n \ell(\beta) R + (m-t) f_1(\sigma\delta) + t\, f_2(\sigma, \delta)} \qquad \text{(I-3)}$$

Case II: $m < t$

The righthand side of (I-1) is equal to

$$\sum_{\underset{\sim}{\chi}^t} w(\underset{\sim}{\chi}^t)\, w(\underset{\sim}{\chi}_m^t)^{-\sigma\delta} \left(\sum_{\underset{\sim}{x}^m} \frac{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m)\, r(\underset{\sim}{x}^m)}{w(\underset{\sim}{\chi}^m)}\; w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m)^{-\sigma\delta} \right) \cdot$$

$$\cdot \left(2^{\delta n \ell(B)}\, R \sum_{\underset{\sim}{x}^t} r(\underset{\sim}{x}^t)\; w(\underset{\sim}{\chi}^t/\underset{\sim}{x}^t)^{\sigma} \right)^{\delta} =$$

$$= 2^{\delta n \ell(B)}\, R \left\{ \sum_{\underset{\sim}{\chi}_m^t} w(\underset{\sim}{\chi}_m^t) \left[\sum_{\underset{\sim}{x}_m^t} r(\underset{\sim}{x}_m^t) \left(\frac{w(\underset{\sim}{\chi}_m^t/\underset{\sim}{x}_m^t)}{w(\underset{\sim}{\chi}_m^t)} \right)^{\sigma} \right]^{\delta} \right\} \cdot$$

$$\cdot \left\{ \sum_{\underset{\sim}{\chi}^m} w(\underset{\sim}{\chi}^m) \left[\sum_{\underset{\sim}{x}^m} r(\underset{\sim}{x}^m) \left(\frac{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m)}{w(\underset{\sim}{\chi}^m)} \right)^{1-\sigma\delta} \right] \left[\sum_{\underset{\sim}{x}^m} r(\underset{\sim}{x}^m) \left(\frac{w(\underset{\sim}{\chi}^m/\underset{\sim}{x}^m)}{w(\underset{\sim}{\chi}^m)} \right)^{\sigma} \right]^{\delta} \right\} =$$

$$= 2^{\delta n \ell(B)}\, R + (t-m)\, f_3(\sigma, \delta) + m\, f_2(\sigma, \delta) \qquad \text{(I-4)}$$

Relations (I-3) and (I-4) substantiate the top and bottom bounds of (9), respectively.

Appendix II

Relations Between Functions $f_i\left(\frac{1}{1+\delta}, \delta\right)$

Theorem II-1

$$\frac{1+\gamma}{\gamma}\left[f_3\left(\frac{1}{1+\gamma}, \gamma\right) + E_o(\gamma)\right] \leq \frac{1}{\gamma} E_o(\gamma) \leq -\frac{1+\gamma}{\gamma} f_1\left(\frac{\gamma}{1+\gamma}\right) \tag{II-1}$$

with equality on either side if and only if

$$\sum_x \left(\frac{w(y/x)}{w(y)}\right)^{\frac{1}{1+\gamma}} r(x) = \text{const} \quad \text{for all } y \tag{II-2}$$

Proof

Using Holder's inequality,

$$\exp_2 \frac{1+\gamma}{\gamma} f_1\left(\frac{\gamma}{1+\gamma}\right) = \left(\sum_y w(y) \sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\frac{1}{1+\gamma}} r(x)\right)^{\frac{1+\gamma}{\gamma}}$$

$$\leq \left(\sum_y w(y) \left\{\sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\frac{1}{1+\gamma}} r(x)\right\}^{1+\gamma}\right)^{\frac{1}{\gamma}} = \exp_2 \frac{1}{\gamma} f_2\left(\frac{1}{1+\gamma}, \gamma\right)$$

with equality if and only if (II-2) holds. This establishes the righthand side of (II-1). Similarly,

$$\exp_2 f_3\left(\frac{1}{1+\gamma},\ \gamma\right) = \left(\sum_y w(y)\left\{\sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\frac{1}{1+\gamma}} r(x)\right\}^{\gamma}\right.$$

$$\leq \left(\sum_y w(y)\left\{\sum_x \left[\frac{w(y/x)}{w(y)}\right]^{\frac{1}{1+\gamma}} r(x)\right\}^{1+\gamma}\right)^{\frac{1}{1+\gamma}} =$$

$$= \exp_2 \frac{\gamma}{1+\gamma}\ f_2\left(\frac{1}{1+\gamma},\ \gamma\right) \qquad \text{(II-3)}$$

with equality if and only if (II-2) holds. As a consequence

$$\frac{1+\gamma}{\gamma}\left[f_3\left(\frac{1}{1+\gamma},\ \gamma\right) + E_o(\gamma)\right] \leq \frac{1+\gamma}{\gamma}\left[-\frac{\gamma}{1+\gamma} E_o(\gamma) + E_o(\gamma)\right] = \frac{1}{\gamma} E_o(\gamma)$$

so that the lefthand inequality of (II-1) holds as well. Q.E.D.

Since $E_o(\gamma) = -f_2\left(\frac{1}{1+\gamma},\ \gamma\right)$ the relation (II-1) establishes that for every $\delta \in (0, 1)$, G can be chosen so as to satisfy (35).

Appendix III

Properties of $f_i(\sigma, \delta)$ Functions

Theorem III-1

The function $f_1(\lambda)$ is convex. $f_1(0) \equiv 0$, $f_1(1) \leq 0$ with equality if and only if $w(y/x) > 0$ whenever $r(x) > 0$. Finally, $f_1'(0) = -I(X; Y)$, the mutual information between X and Y.

Proof

Let

$$X_y \triangleq \frac{w(y/x)}{w(y)} \tag{III-1}$$

Then

$$f_1(1-\lambda) = \log \underset{\sim}{E}_y \left[\underset{\sim}{E} (X_y)^{\lambda}\right] \tag{III-2}$$

If $\lambda = \theta\lambda_1 + (1-\theta)\lambda_2$ with $\theta \in (0, 1)$, then

$$f_1(1-\lambda) = \log \underset{\sim}{E}_y \left[\underset{\sim}{E}\, X_y^{\theta\lambda_1 + (1-\theta)\lambda_2}\right] \leq$$

$$\leq \log \underset{\sim}{E}_y \left[\underset{\sim}{E}\, X_y^{\lambda_1}\right]^{\theta} \left[\underset{\sim}{E} X_y^{\lambda_2}\right]^{1-\theta} \leq$$

$$\leq \log \left\{\underset{\sim}{E}_y\left[\underset{\sim}{E} X_y^{\lambda_1}\right]\right\}^{\theta} \left\{\underset{\sim}{E}_y\left[\underset{\sim}{E} X_y^{\lambda_2}\right]\right\}^{1-\theta} =$$

$$= \theta\, f_1(1-\lambda_1) + (1-\theta)\, f_1(1-\lambda_2) \tag{III-3}$$

(III-3) proves the convexity of $f_1(\lambda)$.

Since $\underset{\sim}{E} X_y = 1$ then $f_1(0) = 0$. Next,

$$f_1(1) = \lim_{\lambda \downarrow 0} \log \sum_y w(y) \sum_x \left(\frac{w(y/x)}{w(y)}\right)^{\lambda} r(x) \leq 0$$

with equality if and only if

$$\lim_{\lambda \downarrow 0} \left(\frac{w(y/x)}{w(y)}\right)^{\lambda} = 1 \quad \text{whenever } r(x) > 0$$

Finally,

$$\frac{\partial}{\partial \lambda} f_1(1-\lambda) = \frac{\underset{\sim}{E}_y \underset{\sim}{E}\left[X_y^{\lambda} \log X_y\right]}{\underset{\sim}{E}_y \underset{\sim}{E}\left[X_y^{\lambda}\right]}$$

so

$$\lim_{\lambda \downarrow 0} f_1'(\lambda) = -\lim_{\lambda \uparrow 1} f_1'(1-\lambda) = -\underset{\sim}{E}_y \underset{\sim}{E}\, X_y \log X_y =$$

$$= -\sum_y w(y) \sum_x r(x) \frac{w(y/x)}{w(y)} \log \frac{w(y/x)}{w(y)} = -I(X;Y) \qquad \text{Q.E.D.}$$

Theorem III-2

$f_2(\sigma, \delta)$ is convex in σ. $f_2(0, \delta) \leq 0$ with equality if and only if $w(y/x) > 0$ whenever $r(x) > 0$. $f_2(1, \delta) = f_1(\delta)$. Thus for $\delta \leq 1$, $f_2(1, \delta) \leq 0$.

Proof

Using (III-1)

$$f_2(\sigma, \delta) = \log \underset{\sim}{E}_y \left(\underset{\sim}{E}\, X_y^{\sigma}\right)^{\delta} \left(\underset{\sim}{E}\, X_y^{1-\sigma\delta}\right) \qquad \text{(III-4)}$$

Let $\theta \in (0, 1)$. Then

$$f_2(\theta\sigma_1 + (1-\theta)\sigma_2, \delta) =$$

$$= \log \underset{\sim}{E}_y \left(\underset{\sim}{E} X_y^{\theta\sigma_1 + (1-\theta)\sigma_2} \right)^{\delta} \left(\underset{\sim}{E} X_y^{\theta(1-\sigma_1\delta) + (1-\theta)(1-\sigma_2\delta)} \right)$$

$$\leq \log \underset{\sim}{E}_y \left[\left(\underset{\sim}{E} X_y^{\sigma_1} \right)^{\delta} \left(\underset{\sim}{E} X_y^{1-\sigma_1\delta} \right) \right]^{\theta} \left[\underset{\sim}{E} X_t^{\sigma_2} \right)^{\delta} \left(\underset{\sim}{E} X_y^{1-\sigma_2\delta} \right) \Big]^{1-\theta}$$

$$\leq \theta f_2(\sigma_1, \delta) + (1-\theta) f_2(\sigma_2, \delta) \qquad \text{(III-5)}$$

so that $f_2(\sigma, \delta)$ is indeed convex. Since $\underset{\sim}{E} X_y = 1$,

$$f_2(0, \delta) = \lim_{\sigma \downarrow 0} \log \underset{\sim}{E}_y (\underset{\sim}{E} X_y^{\sigma})^{\delta} \leq 0$$

with equality if and only if

$$\lim_{\sigma \downarrow 0} \underset{\sim}{E} X_y^{\sigma} = 1 \quad \text{for all } y$$

i.e., if and only if $w(y/x) > 0$ whenever $r(x) > 0$.

The fact that $f_2(1, \delta) = f_1(\delta)$ follows directly from (III-2) and (III-4).

Q.E.D.

Theorem III-3

For $\rho > 0$, $f_2(\rho/\delta, \delta)$ is a convex decreasing function of δ.

Proof

We first prove convexity. Let $\delta = \theta\delta_1 + (1-\theta)\delta_2$ where $\theta \in (0, 1)$.

Also, let

$$a = \frac{\theta \delta_1}{\delta} , \quad 1-a = \frac{(1-\theta)\,\delta_2}{\delta} \tag{III-6}$$

so that

$$\frac{\rho}{\delta} = a \frac{\rho}{\delta_1} + (1-a) \frac{\rho}{\delta_2} \tag{III-7}$$

Then

$$f_2\left(\frac{\rho}{\delta}, \delta\right) = \log \underset{\sim}{E}_y \left(\underset{\sim}{E} X_y^{a\rho/\delta_1 + (1-a)\rho/\delta_2}\right)^{\delta} \left(\underset{\sim}{E} X_y^{1-\rho}\right) \leq$$

$$\leq \log \underset{\sim}{E}_y \left(\underset{\sim}{E} X_y^{a\rho/\delta_1}\right)^{\theta\delta_1} \left(\underset{\sim}{E} X_y^{a\rho/\delta_1}\right)^{(1-\theta)\delta_2} \left(\underset{\sim}{E} X_y^{1-\rho}\right) \leq$$

$$\leq \theta\, f_2(\rho/\delta_1, \delta_1) + (1-\theta)\, f_2(\rho/\delta_2, \delta_2) \tag{III-8}$$

which proves convexity.

Next, after some algebra,

$$\frac{d}{d\delta} f_2\left(\frac{\rho}{\delta}, \delta\right) = \left[\exp_2 -f_2(\rho/\delta, \delta)\right] \cdot$$

$$\cdot \underset{\sim}{E}_y \left\{ \left(\underset{\sim}{E} X_y^{1-\rho}\right) \left(\underset{\sim}{E} X_y^{\rho/\delta}\right)^{\delta-1} \left(\underset{\sim}{E} X_y^{\rho/\delta} \log \frac{E X_y^{\rho/\delta}}{X_y^{\rho/\delta}}\right) \right\} \leq 0$$

where we made use of the $\log x \leq x-1$ inequality. Q.E.D.

Theorem III-4

The function $f_3(\sigma, \delta)$ is convex in σ. $f_3(1, \delta) = 0$ and $f_3(0, \delta) \leq 0$ with equality if and only if $w(y/x) > 0$ whenever $r(x) > 0$.

Proof

Using (III-1)

$$f_3(\sigma, \delta) = \log \underset{\sim}{E}_y \left(\underset{\sim}{E}\, X_y^{\sigma}\right)^{\delta}$$

Thus if $\theta \in (0, 1)$ then

$$f_3(\theta\sigma_1 + (1-\theta)\sigma_2, \delta) \leq \log \underset{\sim}{E}_y (\underset{\sim}{E}\, X_y^{\sigma_1})^{\theta\delta} (\underset{\sim}{E}\, X_y^{\sigma_2})^{(1-\theta)\delta} \leq$$

$$\leq \theta f_3(\sigma_1, \delta) + (1-\theta) f_3(\sigma_2, \delta)$$

Next, $f_3(1, \delta) = 0$ because $\underset{\sim}{E}\, X_y = 1$. Finally,

$$\lim_{\sigma \downarrow 0} f_3(\sigma, \delta) = \lim_{\sigma \downarrow 0} f_2(\sigma, \delta) \leq 0$$

with equality if and only if $w(y/x) > 0$ whenever $r(x) > 0$. Q.E.D.

Theorem III-5

$\frac{1+\delta}{\delta}\left[E_o(\delta) + f_3\left(\frac{1}{1+\delta}, \delta\right)\right]$ is a non-negative function of $\delta \geq 0$.

Proof

Since by Holder's inequality

$$\underset{\sim}{E}\, X_y^{\frac{1}{1+\delta}} \leq \left(\underset{\sim}{E}\, X_y\right)^{\frac{1}{1+\delta}} = 1$$

then

$$f_3\left(\frac{1}{1+\delta}, \delta\right) = \log \underset{\sim}{E}_y \left(\underset{\sim}{E}\, X_y^{\frac{1}{1+\delta}}\right)^{\delta} \geq$$

$$\geq \log \underset{\sim}{E}_y \left(\underset{\sim}{E}\, X_y^{\frac{1}{1+\delta}}\right)^{1+\delta} = f_2\left(\frac{1}{1+\delta}, \delta\right) = -E_o(\delta)$$

Therefore the function is indeed non-negative for all $\delta \geq 0$. Q.E.D.

Appendix IV

Properties of $g_i(\sigma, \delta)$ Functions

Lemma IV-1

For all $\rho > 0$, $g_2(\sigma, \rho)$ is minimized by the choice $\sigma = 1/2$.

Proof

Consider any input letter $x' \neq 0$. By definition of channels symmetrical from the input [see Jelinek [1], p. 201], there exists a permutation π of outputs y such that

$$w(y/0) = w(\pi(y)\,|\,x') \quad \text{for all } y \tag{IV-1}$$

and a permutation π^* of inputs x such that

$$w(y\,|\,x) = w(\pi'(y)\,|\,\pi^*(x)) \quad \text{for all } x \tag{IV-2}$$

Therefore,

$$\sum_x \left(\sum_y w(y/0)^\sigma \, w(y\,|\,x)^{1-\sigma} \right)^{1/\rho} = \sum_x \left(\sum_y w(\pi(y)\,|\,x')^\sigma \; w(\pi(y)\,|\,\pi^*(x))^{1-\sigma} \right)^{1/\rho} =$$

$$= \sum_x \left(\sum_y w(y\,|\,x')^\sigma \, w(y\,|\,x)^{1-\sigma} \right)^{1/\rho} \tag{IV-3}$$

and IV-3 holds for all x'. Thus we can write

$$g_2(\sigma, \rho) = \rho \log \frac{1}{a^2} \sum_{x, x'} \left(\sum_y w(y/x')^\sigma \; w(y/x)^{1-\sigma} \right)^{1/\rho} \tag{IV-4}$$

It is well known that the righthand side of (IV-4) is minimized by the choice $\sigma = 1/2$ (see Jelinek [1], p. 246, problem 7.28). Q.E.D.

Define an equidistant symmetrical channel (c.f. Jelinek [1], p. 230) as a channel symmetrical from the input that also satisfies

$$\sum_y \sqrt{w(y/0)\, w(y/x)} = \alpha \qquad \text{for all } x \neq 0$$

$$\sum_y w(y/0) \sqrt{\frac{w(y/x)}{w(y)}} = \gamma \qquad \text{for all } x \neq 0 \tag{IV-5}$$

Theorem IV-1

For equidistant symmetrical channels and all $\rho \geq 1$,

$$g_3(1/2, \rho) + g_1(1/2) \leq g_2(1/2, \rho) \tag{IV-6}$$

Proof

Instead of (IV-6) we will prove that

$$\exp \frac{1}{\rho}\left[g_3(1/2, \rho) + g_1(1/2)\right] \leq \exp \frac{1}{\rho}\left[g_2(1/2, \rho)\right] \tag{IV-7}$$

Let

$$\phi(a_x) = a_x^{1/\rho} \tag{IV-8}$$

Then ϕ is an increasing, concave function for $\rho \geq 1$. If we let

$$a_x = \left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{1/2}\right]\left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)\left(\frac{w(y/x)}{w(y)}\right)^{1/2}\right] \tag{IV-9}$$

and

$$a'_x = \left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{1/2} \left(\frac{w(y/x)}{w(y)}\right)^{1/2}\right] \tag{IV-10}$$

then our task is to prove that

$$\sum_x \phi(a_x) \leq \sum_x \phi(a'_x) \tag{IV-11}$$

or, utilizing condition (IV-5), that

$$\phi(a_o) + (a-1)\,\phi(a_1) \leq \phi(a_o') + (a-1)\,\phi(a_1') \tag{IV-12}$$

It follows from a trivial modification of Theorem 108 on p. 89 of Hardy, Littlewood, and Pola [11] that (IV-12) holds if

$$\begin{aligned} &a_o \geq a_1,\ a_o' \geq a_1' \\ &a_o \geq a_o',\ a_o + (a-1)\,a_1 \leq a_o' + (a-1)\,a_1' \end{aligned} \tag{IV-13}$$

We must therefore prove that (IV-13) is indeed satisfied. Now, by Holder's inequality,

$$\gamma = \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right) \left(\frac{w(y/x)}{w(y)}\right)^{1/2} \leq$$

$$\left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{3/2}\right]^{2/3} \left[\sum_y w(y) \left(\frac{w(y/x)}{w(y)}\right)^{3/2}\right]^{1/3} =$$

$$= \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{3/2} \tag{IV-14}$$

where the last step is due to the symmetricity conditions (IV-1) and (IV-2). (IV-14) proves $a_o \geq a_1$. Next,

$$a_1' = \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{1/2} \left(\frac{w(y/1)}{w(y)}\right)^{1/2} \leq \left[\sum_y w(y)\, \frac{w(y/0)}{w(y)}\right]^{1/2} \cdot$$

$$\cdot \left[\sum_y w(y) \left(\frac{w(y/1)}{w(y)}\right)\right]^{1/2} = a_o' \tag{IV-15}$$

and

$$a_o = \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{3/4} \left(\frac{w(y/0)}{w(y)}\right)^{1/4} \leq$$

$$\leq \left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{3/2}\right]^{1/2} \left[\sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{1/2}\right]^{1/2} = [a'_o]^{1/2}$$

However, $a_o = 1$ so that $a_o^2 = a_o \leq a'_o$. Finally, we must substantiate the last inequality in (IV-13). But because of the symmetricity of the channel,

$$\sum_x \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right) \left(\frac{w(y/x)}{w(y)}\right)^{1/2} = \sum_x \sum_y w\left(\pi(y)\right) \frac{w(\pi(y)/x')}{w(\pi(y))} \left[\frac{w(\pi(y)/\pi^*(x))}{w(\pi(y))}\right]^{1/2} =$$

$$= \sum_x \sum_y w(y) \frac{w(y/x')}{w(y)} \left[\frac{w(y/x)}{w(y)}\right]^{1/2} \qquad \text{(IV-16)}$$

where the permutations π and π^* are those referred to in the proof of Lemma IV-1. Since (IV-16) holds for all x', we get

$$a_o + (a-1)\, a_1 = \sum_x a_x = \left\{\frac{1}{a} \sum_{x'} \sum_y w(y) \left[\frac{w(y/x')}{w(y)}\right]^{1/2}\right\}$$

$$\cdot \left\{\frac{1}{a} \sum_{x'} \sum_x \sum_y w(y) \frac{w(y/x')}{w(y)} \left(\frac{w(y/x)}{w(y)}\right)^{1/2}\right\} =$$

$$= \frac{1}{a} \left\{\sum_y w(y) \sum_x \left(\frac{w(y/x)}{w(y)}\right)^{1/2}\right\} \left\{\sum_y w(y) \sum_x \left(\frac{w(y/x)}{w(y)}\right)^{1/2}\right\} \qquad \text{(IV-17)}$$

On the other hand,

$$a_o' + (a-1)a_1' = \sum_x a_x' = \sum_x \sum_y w(y) \left(\frac{w(y/0)}{w(y)}\right)^{1/2} \left(\frac{w(y/x)}{w(y)}\right)^{1/2} \Big\} =$$

$$= \frac{1}{a} \sum_{x'} \sum_x \sum_y w(y) \left(\frac{w(y/x')}{w(y)}\right)^{1/2} \left(\frac{w(y/x)}{w(y)}\right)^{1/2} =$$

$$= \frac{1}{a} \sum_y w(y) \left[\sum_x \left(\frac{w(y/x')}{w(y)}\right)^{1/2}\right]^2 \qquad \text{(IV-18)}$$

Since (IV-17) has the form $\frac{1}{a}(\underset{\sim}{E}\ Z)^2$ and (IV-18) has the form $\frac{1}{a}\ \underset{\sim}{E}\ (Z^2)$, the last relation of (IV-13) holds, and the theorem is proven.

Q.E.D.

Lemma IV-2

For any $\rho \geq 0$, the functions $g_1(\sigma)$, $g_2(\sigma, \rho)$, and $g_3(\sigma, \rho)$ are convex with σ.

Proof

By Holder's inequality,

$$g_1(\theta\sigma_1 + (1-\theta)\sigma_2) = \log \sum_y w(y/0) \left[\frac{w(y)}{w(y/0)}\right]^{\theta\sigma_1 + (1-\theta)\sigma_2}$$

$$\leq \log \left[\sum_y w(y/0) \left(\frac{w(y)}{w(y/0)}\right)^{\sigma_1}\right]^{\theta} \left[\sum_y w(y/0) \left[\frac{w(y)}{w(y/0)}\right]^{\sigma_2}\right]^{1-\theta} =$$

$$= \theta\, g_1(\sigma_1) + (1-\theta)\, g_1(\sigma_2)$$

so $g_1(\sigma)$ is convex. Similarly,

$$g_2(\theta\sigma_1 + (1-\theta)\sigma_2, \rho) = \rho \log \frac{1}{a} \sum_x \left[\sum_y w(y/x) \left(\frac{w(y/0)}{w(y/x)} \right)^{\theta\sigma_1 + (1-\theta)\sigma_2} \right]^{1/\rho}$$

$$\leq \rho \log \frac{1}{a} \sum_x \left[\sum_y w(y/x) \left(\frac{w(y/0)}{w(y/x)} \right)^{\sigma_1} \right]^{\frac{\theta}{\rho}} \left[\sum_y w(y/x) \left(\frac{w(y/0)}{w(y/x)} \right)^{\sigma_2} \right]^{\frac{1-\theta}{\rho}}$$

$$\leq \theta\, g_2(\sigma_1, \rho) + (1-\theta)\, g_2(\sigma_2, \rho)$$

so $g_2(\sigma, \rho)$ is convex as well. The convexity of $g_3(\sigma, \rho)$ is proven in the same way.

Q.E.D.

Lemma IV-3

For any fixed $\sigma \in (0, 1)$, $\frac{1}{\rho} g_2(\sigma, \rho)$ is an increasing function of $\rho > 0$.

Proof

$$2^{\frac{1}{\rho} g_2(\sigma, \rho)} = \frac{1}{a} \sum_x \left(\sum_y w(y/0)^\sigma\, w(y/x)^{1-\sigma} \right)^{1/\rho} \triangleq h(\tfrac{1}{\rho})$$

But

$$\frac{d}{d\lambda} h(\lambda) = \frac{1}{a} \sum_x \left(\sum_y w(y/0)^\sigma\, w(y/x)^{1-\sigma} \right)^{\lambda} \log \left(\sum_y w(y/0)^\sigma\, w(y/x)^{1-\sigma} \right)$$

and for $\sigma \in (0, 1)$

$$\sum_y w(y/0)^\sigma\, w(y/x)^{1-\sigma} \leq \left(\sum_y w(y/0) \right)^\sigma \left(\sum_y w(y/x) \right)^{1-\sigma} = 1$$

so that $h'(\lambda) \leq 0$. Therefore

$$\frac{d}{d\rho} \left[2^{\frac{1}{\rho} g_2(\sigma, \rho)} \right] = -\frac{1}{\rho^2} h'(\tfrac{1}{\rho}) \geq 0$$

for $\rho > 0$.

Q.E.D.

Lemma IV-4

$$-g_2'(1/2, \rho) = \sum_x \frac{h(x)^{1/\rho}}{\sum_z h(z)^{1/\rho}} \log \frac{h(x)^{1/\rho}}{\sum_z h(z)^{1/\rho}} + \log a$$

where

$$h(x) = \sum_y \sqrt{w(y/0)\, w(y/x)}$$

Proof

Involves simple algebra and is omitted. Q.E.D.

References

1. Jelinek, Frederick, Probabilistic Information Theory, McGraw-Hill; New York, 1968.

2. Zigangirov, K. S., "Sequential Decoding Procedure with Error Probability Exponent Given by Random Coding," Problems of Information Transmission, Vol. 4, No. 2, pp. 93-95, 1968.

3. Stiglitz, I. and H. Yudkin, Personal communication.

4. Jelinek, Frederick and J. Cocke, "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels," Information and Control, 18, No. 3, April 1971.

5. Jelinek, Frederick, "A Fast Sequential Decoding Algorithm Utilizing a Stack," IBM Journal of Research and Development, 13, No. 6, November, 1969.

6. Jelinek, Frederick, "An Upper Bound on Moments of Sequential Decoding Effort," IEEE Transactions on Information Theory, IT-15, No. 1, January, 1969.

7. Zigangirov, K. S., "Some Sequential Decoding Procedures," Problems of Information Transmission, Vol. 2, No. 4, pp. 13-25, 1966.

8. Schneider, K. S., Reliable Data Compression of Constant Rate Markov Sources for Fixed Rate Channels, Ph.D. Thesis, School of Electrical Engineering, Cornell University, 1966.

9. Gallager, R. G., "A Simple Derivation of the Coding Theorem and Some Applications," IEEE Transactions on Information Theory, IT-11, No. 1, January, 1965.

10. Viterbi, A. J. and J. P. Odenwalder, "Further Results on Optimal Decoding of Convolutional Codes," IEEE Transactions on Information Theory, IT-15, No. 6, pp. 732-734, Nov. 1969.

11. Hardy, G. H. and J. E. Littlewood and G. Polya, Inequalities, Cambridge University Press, Cambridge, 1952.

Figure Captions

Figure 1: A trellis for a rate R = 1/2 code.

Figure 2: Partial tree of a code of rate R = 2/n .

Figure 3: Graphical maximization of s* leading to an asymptotically optimal upper bound to Q(a).

Figure 4: Undetected error (top curves) and failure exponents (bottom curves) for a binary output quantized Gaussian channel with SNR equal to 1.5 dB per transmitted bit when different bias values are used.

Figure 5: Undetected error (top curves) and failure exponents (bottom curves) for an octal output quantized Gaussian channel with SNR equal to -0.3 dB per transmitted bit when different bias values are used.

Figure 6: Undetected error (top curves) and failure exponents (bottom curves) for an octal output quantized Gaussian channel with SNR equal to -2.0 dB per transmitted bit when different bias values are used.

Figure 7: Pareto exponent pairs for the binary and octal quantized channels of Figures 4, 5, and 6 when the bias G optimizes the Pareto (better curve) or failure (worse curve) exponents.

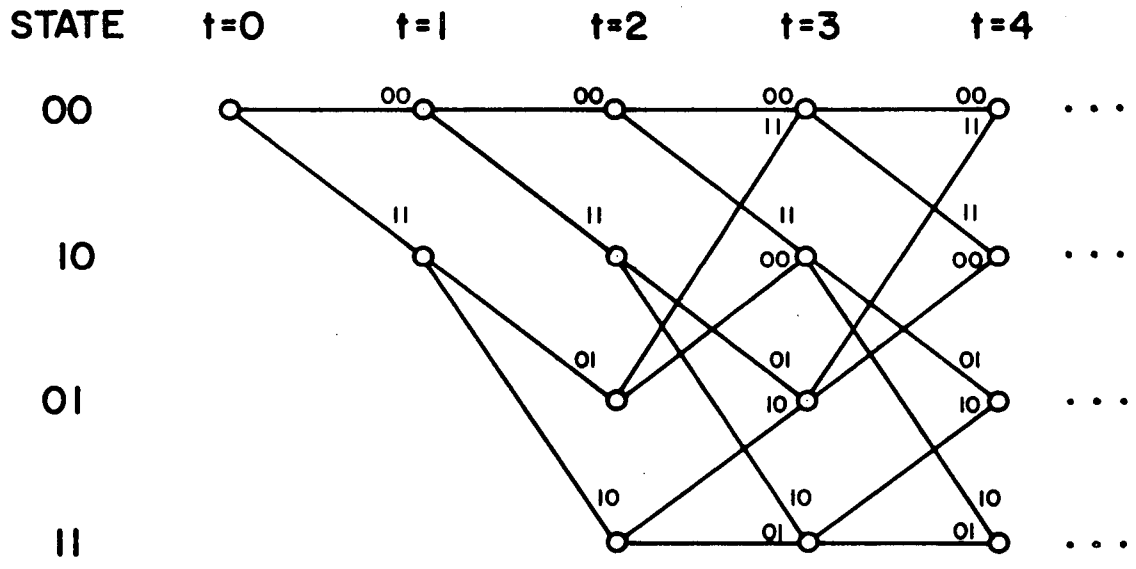

STATE
t=0
t=1
t=2
t=3
t=4
00
10
01
11
00
11
11
01
10
00
11
01
10
01
10
00
11
01
10
01
10
00
11
01
10
01
10
. . .
. . .
. . .
. . .

FIG. 1

C.3.

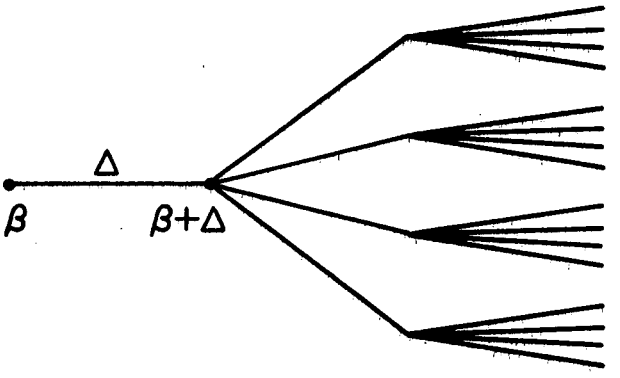


FIG. 2

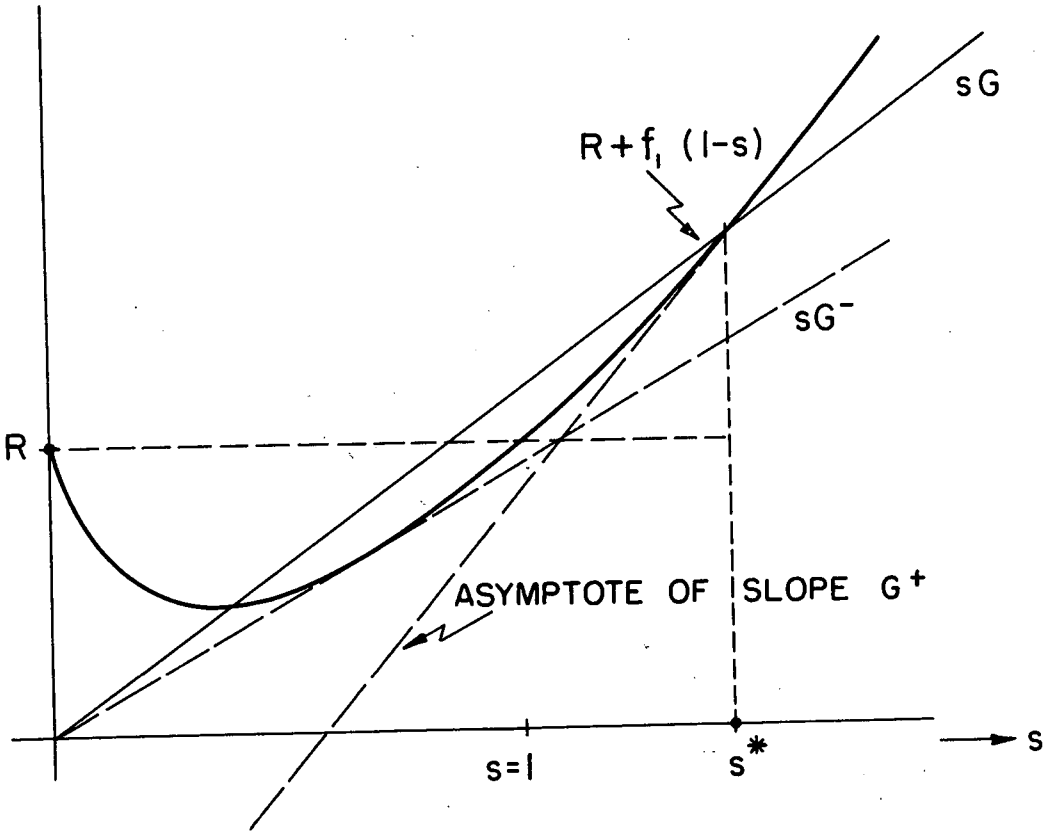


FIG. 3

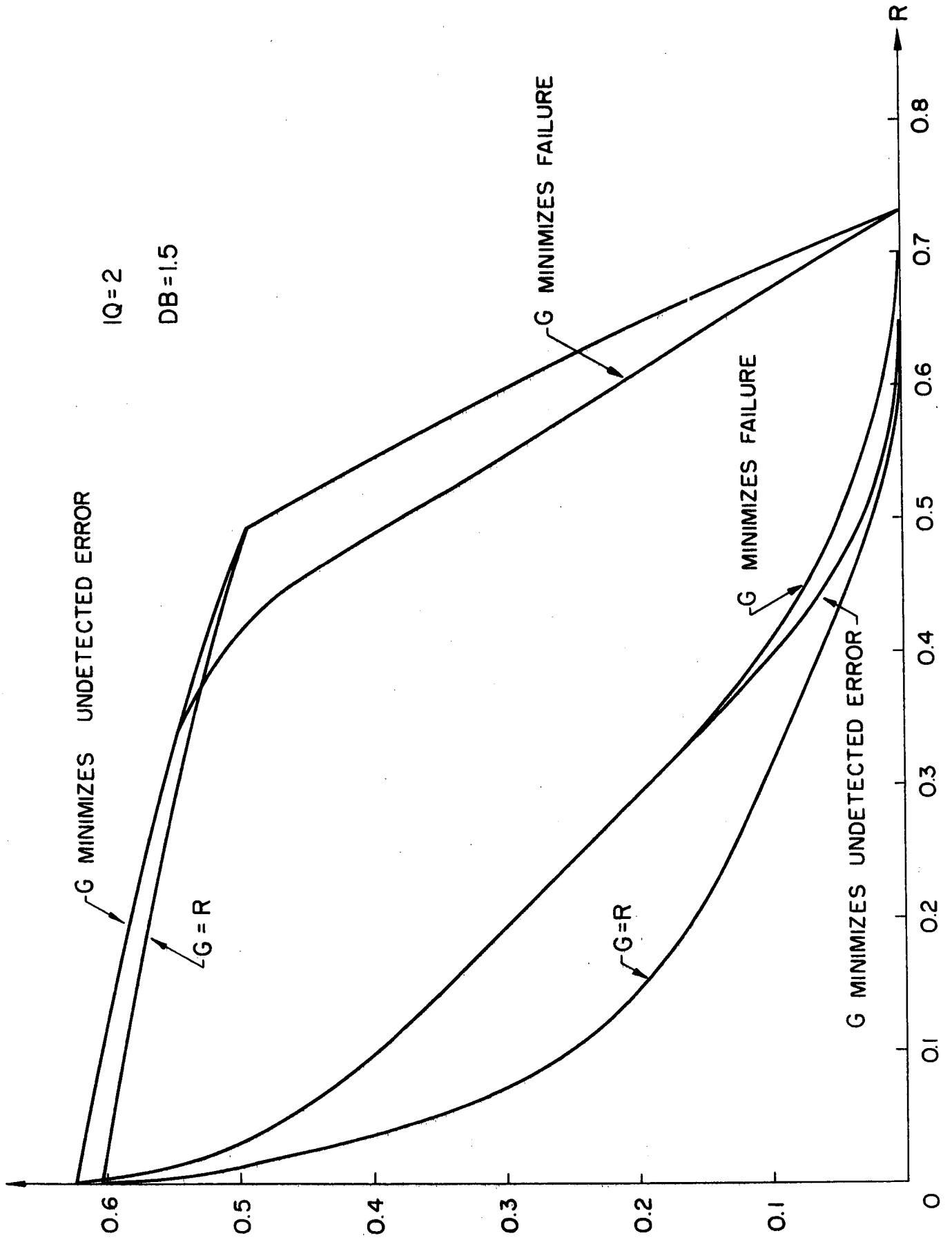

IQ=2
DB=1.5
G MINIMIZES UNDETECTED ERROR
G=R
G MINIMIZES FAILURE
G MINIMIZES FAILURE
G=R
G MINIMIZES UNDETECTED ERROR
R
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

FIG. 4

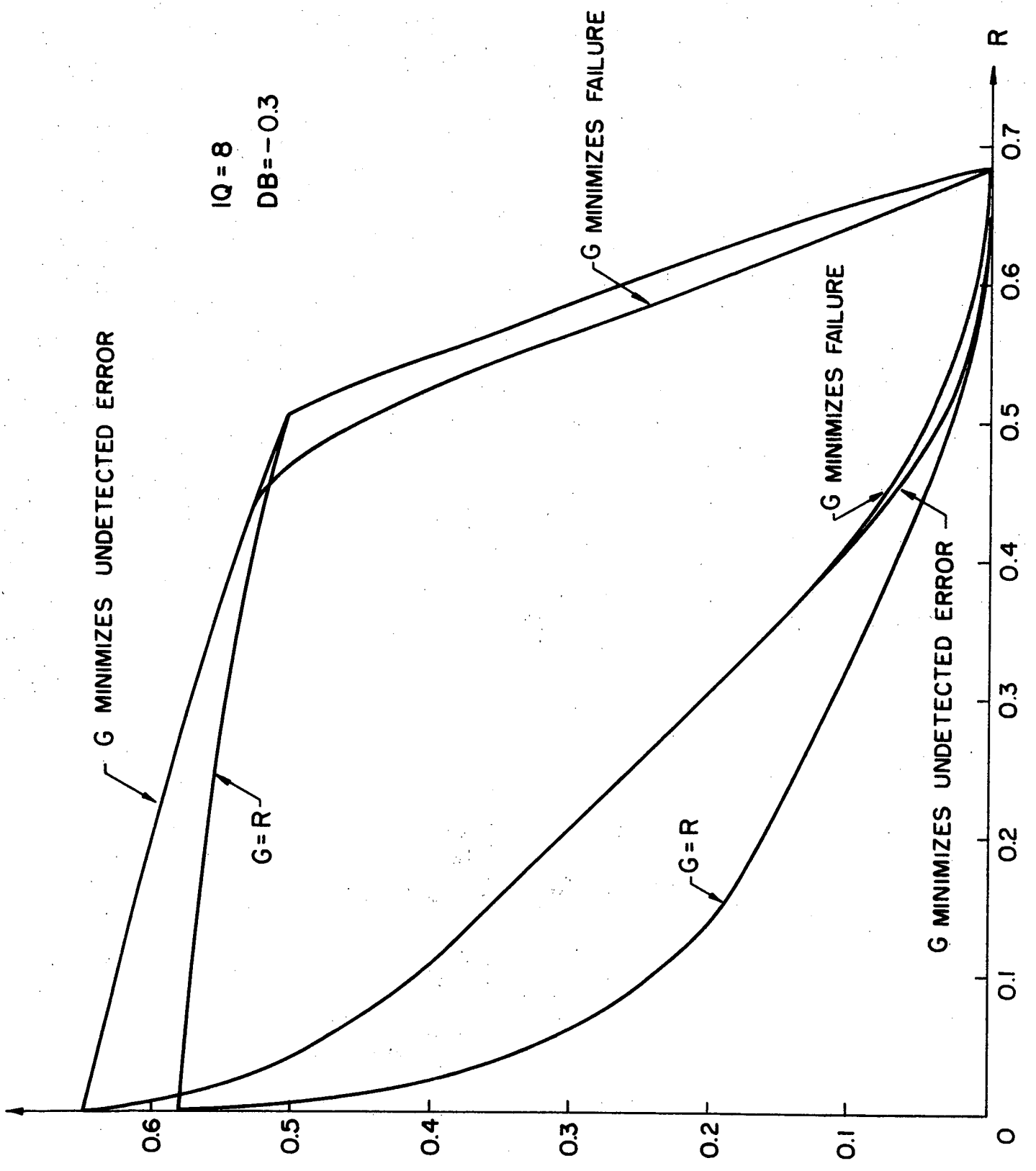

IQ = 8
DB = −0.3
G MINIMIZES UNDETECTED ERROR
G MINIMIZES FAILURE
G MINIMIZES FAILURE
G MINIMIZES UNDETECTED ERROR
G=R
G=R
R
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

FIG. 5

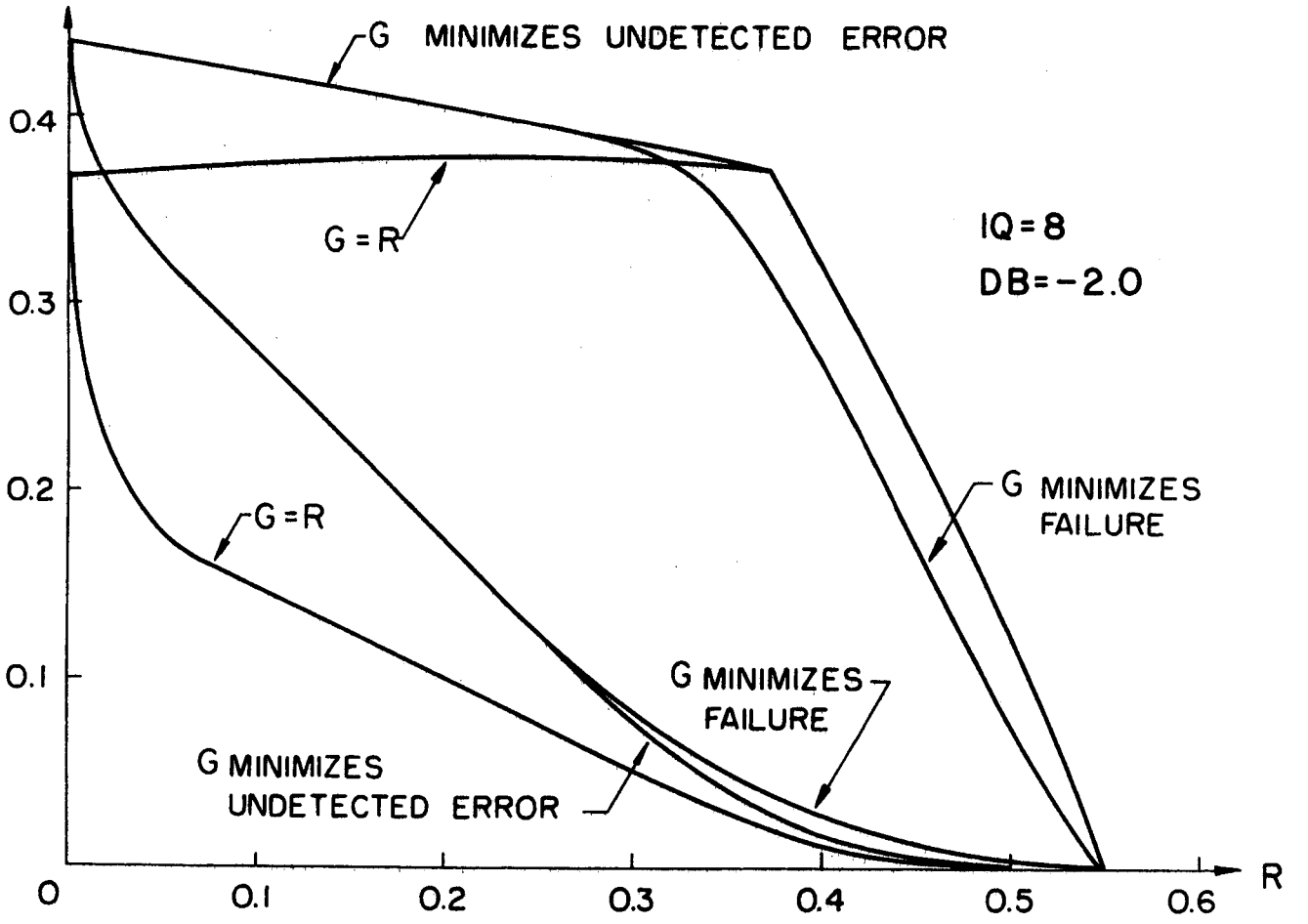

G MINIMIZES UNDETECTED ERROR
G = R
IQ = 8
DB = −2.0
G MINIMIZES FAILURE
G = R
G MINIMIZES FAILURE
G MINIMIZES UNDETECTED ERROR
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
R

FIG. 6

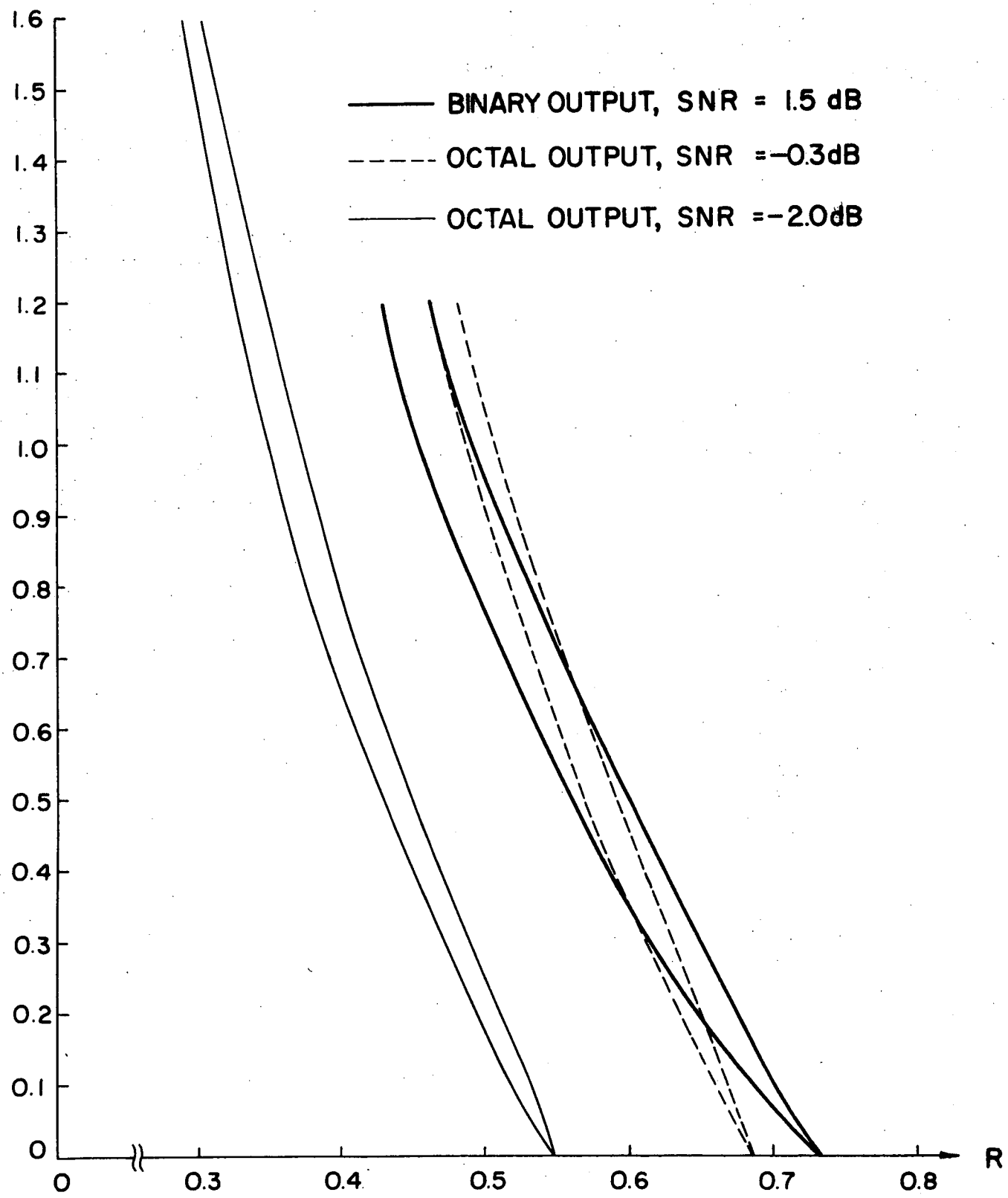

BINARY OUTPUT, SNR = 1.5 dB
OCTAL OUTPUT, SNR =-0.3dB
OCTAL OUTPUT, SNR =-2.0dB
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.3
0.4
0.5
0.6
0.7
0.8
R

FIG. 7

II-E. Bootstrap Trellis Decoding

1. Description of the Rudimentary Decoder

Bootstrap trellis decoding is based on a convolutional code of contraint length ν_b (in branches) and its truncated version that is obtained by eliminating all but the first $\mu_b < \nu_b$ digits of each generator defining the original code. The truncated code has therefore $2^{\mu_b - 1}$ trellis states per level. We will assume ν_b to be so large that at the SNR used, the probability of error of the corresponding maximum likelihood (Viterbi) decoding would be negligible compared to the probability of error resulting from the scheme described below (see Section 3).

The rudimentary binary bootstrap trellis decoding algorithm is as follows:

1) m-1 streams of binary data of length N are encoded using the same ν_b-constraint length code, and an mth stream is created using mod 2 position by position addition of the m-1 streams.

2) The m streams are transmitted through the channel, and the receiver creates an appropriate state stream as in Bootstrap Sequential Decoding [3].

3) A μ_b-truncated trellis decoder is used to decode the first stream, its metrics at depth i,

$$\log \frac{w_m(y_i, z_i/x_i)}{w_m(y_i, z_i)} - R \qquad (1)$$

being based on m, the number of streams in a block, on the transmitted and received digit x_i and y_i, and on the state stream digits z_i. The bias R corresponds to the convolutional rate. To each depth i of the N-branches long codeword there correspond $2^{\mu_b - 1}$ likelihoods, the maximum of these at depth n being denoted by L_n. Let

$$L_n^M = \max_{1 \le i \le n} L_i$$

so that L_n^M is a monotone increasing function of $n \in \{1, \ldots, N\}$ (N is the stream length in branches). Let θ be some suitably chosen threshold. If $L_n^M - L_n < \theta$ for all n, the decoder <u>accepts</u> the decoded first stream information sequence, otherwise it <u>rejects</u> it (in fact, it will stop decoding whenever a depth n is reached for which $L_n^M - L_n \ge \theta$).

4) If the 1st stream was accepted, it is replaced by the estimated transmitted stream, the state stream is accordingly recalculated, and the decoder proceeds to decode the 2nd stream as in step 3, using a metric table appropriate to m-1 undecoded streams (the subscript m in (1) is replaced by m-1).

5) If the 1st stream was rejected, 2nd stream decoding proceeds exactly as in (3) with no change to either metric or state stream.

6) Steps 3 through 5 establish a patte rn that is adhered to in general: after every acceptance the state stream and metrics are recalculated, and decoding of the "round robin" next stream begins.

7) Decoding terminates in either of 2 ways:

(a) SUCCESS: all m streams get finally accepted.

(b) FAILURE: when ℓ streams ($\ell \leq m$) remain undecoded, ℓ successive attempts at stream decoding end with rejection.

2. Bounds on the Probability of Failure

In this section we will obtain upper and lower bounds on the probability of failure or error. Let $A_k(\ell)$ and $F_k(\ell)$ denote the events that when m-k streams have been correctly decoded, the ℓth of k remaining streams has been decoded in error and has failed the threshold test, respectively. Let $\overline{A_k}(\ell)$ and $\overline{F_k}(\ell)$ denote the complements of these events when m-k streams have been correctly decoded. Then the probability of failure or error is bounded by

$$P(F \cup E) \leq P\left\{\left[\prod_{i=1}^{m} F_m(i)\right] \cup \left[\bigcup_{i=1}^{m} A_m(i)\,\overline{F}_m(i)\right]\right\} +$$

$$+ P\left\{\bigcup_{i=1}^{m} \overline{A}_m(i)\overline{F}_m(i)\left(\left[\prod_{j=1} F_{m-1}(j)\right] \cup \left[\bigcup_{j\neq i} A_{m-1}(j)\overline{F}_{m-1}(j)\right]\right)\right\} +$$

$$+ P\left\{\bigcup_{i=1}^{m}\bigcup_{\substack{j=1\\ j\neq i}}^{m} \overline{A}_m(i)\overline{F}_m(i)\overline{A}_{m-1}(j)\,\overline{F}_{m-1}(j)\left(\left[\prod_{\substack{\ell\neq i\\ \ell\neq j}} F_{m-2}(\ell)\right] \cup \left[\bigcup_{\substack{\ell\neq i\\ \ell\neq j}} A_{m-2}(j)\overline{F}_{m-2}(j)\right]\right)\right\} +$$

$$+ \ldots + P\left\{\bigcup_{i_j} \overline{A}_m(i_1)\,\overline{F}_m(i_1)\ldots\overline{A}_3(i_{m-2})\overline{F}_3(i_{m-2})\left(\left[F_2(i_{m-1})F_2(i_m)\right]\right.\right.$$

$$\left.\left. \cup A_2(i_{m-1})\overline{F}_2(i_{m-1}) \cup A_2(i_m)\overline{F}_2(i_m)\right)\right\}$$

where the union with the subscript i_j is over all permutations of m-1

digits taken from the set $\{1,2,\ldots,m\}$. Realizing that every term in each union is equally probable, we can upper bound $P(F \cup E)$ further by

$$P(F \cup E) \leq P\left\{\prod_{i=1}^{m} F_m(i)\right\} + \binom{m}{1} P\left\{\prod_{i=1}^{m-1} F_{m-1}(i)\right\} +$$

$$+ \binom{m}{2} P\left\{\prod_{i=1}^{m-2} F_{m-2}(i)\right\} + \ldots + \binom{m}{m-2} P\left\{F_2(1)F_2(2)\right\} +$$

$$+ m\, P\left\{A_m(1)\overline{F}_m(1)\right\} + (m-1)\binom{m}{1} P\left\{A_{m-1}(1)\overline{F}_{m-1}(1)\right\} +$$

$$+ (m-2)\binom{m}{2} P\left\{A_{m-2}(1)\overline{F}_{m-2}(1)\right\} + \ldots + 2\binom{m}{m-2} P\left\{A_2(1)\overline{F}_2(1)\right\} . \quad (2)$$

Since not using the state information increases the probability of not being able to decode, then

$$P\left\{\prod_{j=1}^{\ell} F_\ell(j)\right\} \leq P\left\{\prod_{j=1}^{\ell} F_\infty(j)\right\} = P\left\{F_\infty(1)\right\}^{\ell} \quad (3)$$

where $F_\infty(j)$ denotes the event of failing the threshold test on the jth of a block of $m = \infty$ streams (in such a case state information is worthless). The last equality in (3) follows from the fact that if state information is not used, decoding of any set of streams is independent and identically distributed. Another valid upper bound is

$$P\left\{\prod_{j=1}^{\ell} F_\ell(j)\right\} \leq P\left\{F_\ell(1)\right\} . \quad (4)$$

Collecting the results (2) through (4) we get

$$P(F \cup E) \leq \sum_{\ell=0}^{m-2} \binom{m}{\ell} \left[\min\left\{ P\{F_\infty(1)\}^{m-\ell}, P\{F_{m-\ell}(1)\} \right\} + (m-\ell) P\{A_{m-\ell}(1) \overline{F}_{m-\ell}(1)\} \right]. \tag{5}$$

To lower bound P(F), let B_ℓ be the event that some set of $\ell \leq m$ streams have been correctly decoded and passed the threshold test, and let $C_{m-\ell}$ be the event that after ℓ streams have been decoded, none of the remaining $m-\ell$ streams can be correctly decoded. Then

$$P(F \cup E) \geq P\left\{ \left[B_\ell \cdot C_{m-\ell} \right] \cup B_\ell \right\} \geq P\left\{ C_{m-\ell} \right\}. \tag{6}$$

However, since the probability of decoding at least one of remaining $m-\ell$ streams is smaller than or equal to the probability of decoding at least one of a <u>given</u> set of $m-\ell$ streams that satisfy the parity constraint (because the first ℓ streams to be decoded will in general be the least noisy ones), we have

$$P\left\{ C_{m-\ell} \right\} \geq P\left\{ \prod_{j=1}^{m-\ell} A_{m-\ell}(j) \right\} .$$

Since certainly

$$P\{A_k(1), A_k(2), \ldots, A_k(k-1)/A_k(k)\} \geq P\{A_k(1), A_k(2), \ldots, A_k(k-1)/\overline{A}_k(k)\} ,$$

then

$$P\left\{ C_{m-\ell} \right\} \geq P\left\{ A_{m-\ell}(1) \right\}^{m-\ell} \tag{7}$$

where $P\left\{ A_{m-\ell}(1) \right\}$ denotes the probability that the first of a given set of $m-\ell$ streams cannot be correctly decoded. Furthermore, because of the parity constraint, if two streams remain, then either both or neither

will be correctly decoded. Hence

$$P\{C_2\} \geq P\{A_2(1) \cdot A_2(2)\} = P\{A_2(1)\} \quad . \tag{8}$$

We therefore get from (6), (7), and (8) that

$$P(F \cup E) \geq \max \left\{ P\{A_2(1)\}, \max_{m \geq k \geq 3} P\{A_k(1)\}^k \right\} . \tag{9}$$

3. Estimates on Exponents

In this section we use the bounds (5) and (9) to estimate the limiting behavior of $(1/\mu)\log P(F \cup E)$. We get

$$P\{F_k(1)\} \leq P\{A_k(1)\} + P\{F_k(1)\bar{A}_k(1)\} . \tag{10}$$

Now

$$P\{A_k(1)\} \cong N \; L_k(N,\mu) 2^{-\mu E_k(R)} \tag{11}$$

where $\mu = \mu_b \lambda$ is the truncated constraint length in bits (λ is the number of transmitted digits per branch) $L_k(n,\mu)$ is a slowly varing function of its parameters whose value does not exceed 1, and $E_k(R)$ is the undetected error exponent that corresponds to maximum likelihood decoding of the first of k parity constrained streams (see step (1) of Section 1) that utilizes the received as well as state stream digits when the convolutional transmission rate is R (the net rate that takes into account parity as well as stream tail degradations is

$$\frac{m-1}{m} \cdot \frac{N}{N+\nu_b} R \;) \; .$$

The probability $P\{F_k(1)\bar{A}_k(1)\}$ is upper bounded by the probability that the likelihood on the correct path ever drops by θ . It has been shown in [1] that the bound

$$P\{F_k(1)\bar{A}_k(1)\} \leq K_1 N 2^{-h_k\theta} \tag{12}$$

holds where $K_1 \in (0, 1]$. For channels symmetric from the input h_k is the solution of

$$F = \frac{1}{h_k} f_1^k(-h_k) \tag{13}$$

where

$$f_1^k(\zeta) \triangleq \log \sum_{y,z} w_k(y,z) \left[\frac{w_k(y,z/0)}{w_k(y,z)} \right]^{1-\zeta} . \tag{14}$$

Finally, $P\{A_k(1)F_k(1)\}$ is the probability that some incorrect path passes the threshold test at all depths. It is upper bounded by the probability that the likelihoods of all initially incorrect paths exceed $-\theta$ at the earliest point at which they rejoin the correct path (all paths are joined with correct path at depth $N + \nu$). It is then easy to show that

$$P\{A_k(1)\bar{F}_k(1)\} \leq K_2 N 2^{\sigma_k\theta - \nu[\sigma_k - f_1^k(1-\sigma_k)]} \tag{15}$$

where $\nu = \nu_b\lambda$ is the constraint length in transmitted digits, and K_2 is a finite constant provided

$$R < \sigma_k R - f_1^k(1-\sigma_k) \qquad \sigma_k \geq 0 . \tag{16}$$

Since $f_1^k(\zeta)$ is a convex function of ζ , and

$$\lim_{\zeta \to 0} \frac{1}{\zeta} f_1^k(\zeta) = \frac{d}{d_5} f_1^k(\zeta) \Big|_{\zeta=0} = I_k(X;Y) ,$$

then relations (13) and (16) can be satisfied simultaneously provided

$$R < I_k(X; Y) = \sum_{y,z} w_k(y, z/0) \log \frac{w_k(y, z/0)}{w_k(y, z)} . \tag{17}$$

Plugging (10), (11), (12), and (15) into (5) we get

$$P(F \cup E) \leq \sum_{\ell=0}^{m-2} \binom{m}{\ell} \min \left\{ \left(N\left[K_3 2^{-\mu E_\infty(R)} + K_1 2^{-h_\infty \theta} \right] \right)^{m-\ell} , \right.$$

$$\left. N\left[K_3 2^{-\mu E_{m-\ell}(R)} + K_1 2^{-h_{m-\ell}\theta} \right] \right\} +$$

$$+ \sum_{k=2}^{m} \binom{m}{(m-k)} k K_2 N 2^{\sigma_k \theta - \nu[\sigma_k R - f_1^k(1-\sigma_k)]} . \tag{18}$$

Let

$$\theta = \mu\gamma$$

where

$$\gamma = \max_{2 \leq k \leq m} \frac{1}{h_k} E_k(R) \tag{19}$$

and note that $(m-\ell)\, E_\infty(R)$ and $E_{m-\ell}(R)$ decrease and increase with ℓ, respectively. Also, let $\ell \in (2, m)$ be the index maximizing $\sigma_k \theta - \nu\,[\sigma_k R - f_1^k(1 - \sigma_k)]$, let $\sigma = \sigma_\ell$, and define

$$a \triangleq \sigma R - f_1^k(1 - \sigma) . \tag{20}$$

Then

$$P(F \cup E) \leq K_4 N k^* \, 2^{-\mu\beta_U(R)} + K_5 \, 2^{\mu\sigma\gamma - \nu a} \tag{21}$$

where

$$\beta_U(R) = \min \left\{ k^* E_\infty(R), E_{k^*-1}(R) \right\} \tag{22}$$

and

$$k^* = \min\left\{k : k \, E_\infty(R) \geq E_k(R)\right\} \quad . \tag{23}$$

We see from (21) that

$$\lim_{\mu \to \infty} -\frac{1}{\mu} \log P(F \cup E) \geq \beta_U(R) \tag{24}$$

provided

$$\nu \geq \mu[\beta_U(R) + \sigma\gamma] \quad . \tag{25}$$

Finally, using (9) and (11) we get that

$$P(F \cup E) \geq \max \left\{ N K_4 \, 2^{-\mu E_2(R)}, \max_{m \geq k \geq 3} (NK_4)^k \, 2^{-\mu k E_k(R)} \right\} . \tag{26}$$

Let k^+ be the integer minimizing $k \, E_k(R)$ over $k = 3, 4, \ldots, m$ and define

$$\beta_L(R) = \min \left\{E_2(R), k^+ E_{k^+}(R)\right\} \quad . \tag{27}$$

Then

$$P(F \cup E) \geq K_6 N \, 2^{-\mu\beta_L(R)} \tag{28}$$

and

$$\lim_{\mu \to \infty} -\frac{1}{\mu} \log P(F \cup E) \leq \beta_L(R) \quad . \tag{29}$$

We will summarize our results in the following theorem.

Theorem 1

Let $E_k(R)$ be the exponent of the probability of undetected error corresponding to Viterbi decoding of the first of a block of k received streams when the transmitted codewords satisfy the parity constraint. Let h_k, $k = 1, \ldots, m$ be the solutions of (13), let σ_k maximize the righthand sides of (16), and let γ and α and σ be as defined in (19) and (20). Let the bootstrap trellis decoder be based on the μ-truncated prefix of a convolutional code of constraint length ν. If the stopping threshold has value $\theta = \mu\gamma$ then there are codes whose probability of failure or error satisfies

$$\beta_U(R) \leq - \lim_{\mu \to \infty} \frac{1}{\mu} \log P(F \cup E) \leq \beta_L(R) \tag{30}$$

provided $\nu \geq \mu [\beta_U(R) + \sigma\gamma]$. The bounds $\beta_U(R)$ and $\beta_L(R)$ are given by (22) and (27), respectively.

4. Exponent Evaluation

The preceding theorem gives bounds on the error exponent for Bootstrap Trellis Decoding in terms of the undetected error exponent $E_k(R)$. In this section we show how the bounds can be evaluated.

First note that the exponent $E_k(R)$ is known only for $R \in (R_{comp}, C)$, but that upper and lower bounds to it exist for $R \in (0, R_{comp})$. Since what is wanted in practice is an estimate of the behavior of $P(F \cup E)$, we will take the point of view that for $R \in (0, R_{comp})$, $E_k(R)$ is given by its expurgated lower bound [2].

Let $w_k(y, z/x)$ be the probability that when x is transmitted y is received and z is the state digit, when the block of k transmitted streams satisfies the parity constraint (see Jelinek and Cocke [3]). Assuming a symmetric binary input channel, define the exponent functions

$$E_k^o(\sigma) = (1+\sigma) - \log \sum_{y,z} \left[w_k(y, z/0)^{\frac{1}{1+\delta}} + w_k(y, z/1)^{\frac{1}{1+\delta}} \right]^{1+\delta} \tag{31}$$

$$E_k^x(\sigma) = \sigma - \log \left[1 + \left(\sum_{y,z} \sqrt{w_k(y, z/0) w_k(y, z/1)} \right)^{1/\sigma} \right] . \tag{32}$$

It can be shown that $E_k^o(1) = E_k^x(1)$. Define further

$$E_k^1(\sigma) = \begin{cases} E_k^o(\sigma) & \sigma \in (0,1) \\ E_k^x(\sigma) & \sigma > 1 \end{cases} . \tag{33}$$

Then having assumed the expurgated exponent as the true one, we get for $0 \leq R < C_k$ [C_k is the capacity of the channel $w_k(y, z/x)$]

$$E_k(R) = R\sigma$$

where σ is the solution of

$$R = \frac{1}{\sigma} E_k^1(\sigma) . \tag{34}$$

(34) thus allows us to evaluate both $\beta_U(R)$ and $\beta_L(R)$ provided we solve the equations $R = E_k^1(\sigma)/\sigma$. This is impractical if the β-exponents are wanted for all R. In that case it is best to proceed parametrically with the help of the following theorem.

Theorem 2

Let $\gamma \geq 0$ be arbitrary.

I. The ratio $\beta_U(R)/R$ attains the value γ at the rate

$$R = \max\left\{\frac{1}{\gamma} E_m^1(\gamma),\ \min\left\{\frac{1}{\gamma} E_{k^+-1}^1(\gamma),\ \frac{k^+}{\gamma} E_\infty^1\left(\frac{\gamma}{k^+}\right)\right\}\right\}$$

where

$$k^+ = \min\left\{k : k \geq 2,\ \frac{1}{\gamma} E_k^1(\gamma) \leq \frac{k}{\gamma} E_\infty^1\left(\frac{\gamma}{k}\right)\right\} .$$

II. The ratio $\beta_L(R)/R$ attains the value γ at the rate

$$R = \min\left\{\frac{1}{\gamma} E_2^1(\gamma),\ \frac{\ell^+}{\gamma} E_{\ell^+}^1\left(\frac{\gamma}{\ell^+}\right)\right\}$$

where

$$\ell^+ = \min\left\{m,\ \min\left\{\ell : \ell \geq 3,\ \frac{\ell}{\gamma} E_\ell^1\left(\frac{\gamma}{\ell}\right) \leq \frac{\ell+1}{\gamma} E_{\ell+1}^1\left(\frac{\gamma}{\ell+1}\right)\right\}\right\} .$$

The proof is similar to that of Theorems 3 and 4 of [3] and is omitted. Figures 1 through 4 evaluate $\beta_U(R)$ and $\beta_L(R)$ vs R for $m = \infty$ and compare these to the exponent $E_\infty(R)$ appropriate to straight Viterbi decoding. The four figures apply to the BSC with crossover probabilities $p = 0.045,\ 0.056,\ 0.07$, and 0.09, respectively. It should again be stressed that R is the convolutional rate and not the net rate. For every combination of m, N, and ν_b the latter curves can be obtained by replotting the present ones, taking into account the relationship

$$R_{NET} = \frac{m-1}{m} \frac{N}{N+\nu_b} R .$$

5. Simulation

The simulated bootstrap decoding algorithm(BTDA) operates as follows. First, the truncated trellis algorithm is employed to decode each of the streams. While decoding a stream, if L_n does not exceed its previous maximum within some number of time intervals THRSH, the decoded path will be computed by tracing back from the position of the maximum. The digits on the decoded path will be declared reliable up to the position which is located KBACK intervals earlier than the position of the previous maximum.

Once a portion in a stream is declared reliable, the channel state modifications will be made over that portion, and the algorithm will go on to decode the next stream. When the m-th stream is encountered, first the parity relationship will be used to decode digits above which all the (m-1) streams are declared reliable, and then the truncated Viterbi decoder will be operated over the undecoded digits of the m-th stream.

After decoding the m-th stream, the parity relationship will be used again to decode the portions where (m-1) streams are decoded and declared reliable. These procedures constitute the first pass of the algorithm. For the second pass, the last stream decoded in the first pass will be the first stream to be tried, and, in addition, the decoder will operate backwards starting from the opposite end of the stream.

After decoding of a stream stops, the channel state symbols are modified over the reliable portion according to the definitely decoded digits in that stream. The encoder will go on to decode the next to last stream of the previous pass, and so on. Passes will continue until no further improvement in the length of the reliable stream portion can be achieved.

Using optimization methods described in his Ph.D. thesis [4], H. S. Park selected THRSH=40 and KBACK=50 for m = 10. He simulated the algorithm on a BSC with crossover $p_c = 0.056$ whose $R_{comp} = 0.45$, which is the net value of the transmission rate ($R_{NET} = 9/10\ R$) of the convolutional code of rate 1/2. This allows comparison with the straight maximum likelihood decoding (MLDA) performance of R = 1/2 codes over a BSC with p = 0.045. The following results are obtained:

Hybrid BTDA				Straight MLDA	MLDA Equivalent
p_c	THRSH	KBACK	p_e	p_e	
.056	40	50	.00018	.0034	$\nu \gtrsim 11.5$

Table 1.

The above table lists the constraint length ν necessary for the MLDA algorithm to achieve the error performance $p_e = .00018$.

For meaningful statistical data on p_e for the BTDA, the running time of the simulation program should be large so that the simulated value of p_e be reliable. Due to limited computer time, only 1200

blocks of 10 streams were run to count decoding errors. The BTDA has achieved the error probability 0.00018 for those 1200 blocks. In all, 240 bit errors were responsible for this figure, and these were spread over 40 of the 1200 blocks. As many as 45 of the 240 bit errors occurred in a single block. To achieve more firm support for the value of p_e, additional computer time is needed to view more of these occasional "large error" blocks.

6. Computational Complexity of the BTDA

We shall assume that the computational complexity of the MLDA is determined by

$$E_c = (N + \mu - 1)2^{\mu} \tag{35}$$

where N is the length of the information sequence and $(\mu - 1)$ is the number of digits defining the binary trellis states in the trellis diagram.

In the BTDA, if we let T denote the average number of trials to decode m streams of the hybrid scheme, then the average number of trials M per decoded information stream is given by

$$M = \frac{T}{m-1} \tag{36}$$

where (m-1) takes account of the rate reduction due to the extra parity stream of the hybrid scheme.

If we assume that whenever the BTDA returns to decode a stream that has already been tried, decoding starts at the beginning of that stream, then the average number of computations E_{ch} per decoded

information stream of the hybrid scheme is upper-bounded by

$$E_{ch} \leq M(N + \mu - 1)2^{\mu}. \tag{37}$$

From the simulation program of the BTDA ($\nu = 10$, $\mu = 5$), the number M is shown in Table 2 below. Thus

$$E_{ch} \leq M \cdot (N + \mu - 1) \cdot 2^{\mu} \Big|_{\mu=5} = 1.5 \cdot (104) \cdot 2^{5} < 104 \cdot 2^{6} . \tag{38}$$

However, as shown in the previous section, the performance achieved by the BTDA ($\nu = 10$, $\mu = 5$) is almost equivalent to the performance for the straight MLDA with $\nu \gtrsim 11$, whose E_{ch} is given by

$$E_{ch} = (N + \mu - 1) \cdot 2^{\mu} \Big|_{\substack{N=100 \\ \mu=11}} = 104 \cdot 2^{11} . \tag{39}$$

From Eqs. (38) and (39), the computational complexity of the BTDA compared to the straight MLDA is smaller by almost a factor of $2^5 = 32$.

p_c	ν	μ	THRSH	KBACK	M	p_e	MLDA Equivalent
.056	10	5	40	50	1.5	.00018	$\nu > 11.5$

Table 2.

References

1. F. Jelinek: Final Report on "Study of Sequential Decoding", NASA contract NAS-2-5643, February 1972, Section II-D.

2. A. J. Viterbi and J. P. Odenwalder: "Further Results on Optimal Decoding of Convolutional Codes", IEEE Trans. on Information Theory, IT-15, No. 6, pp. 752-754. November 1969.

3. F. Jelinek and J. Cocke: "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels", Information and Control, 18, No. 3, April 1971.

4. H. S. Park: "Bootstrap Hybrid Trellis Decoding", Ph.D. Thesis, Cornell University, May 1972.

Figure Captions

Figure 1: Comparison of $\beta_L(R)$, $\beta_U(R)$, and $E_\infty(R)$ exponents for the BSC with crossover probability $p_c = 0.045$.

Figure 2: Comparison of $\beta_L(R)$, $\beta_U(R)$, and $E_\infty(R)$ exponents for the BSC with crossover probability $p_c = 0.056$.

Figure 3: Comparison of $\beta_L(R)$, $\beta_U(R)$, and $E_\infty(R)$ exponents for the BSC with crossover probability $p_c = 0.07$.

Figure 4: Comparison of $\beta_L(R)$, $\beta_U(R)$, and $E_\infty(R)$ exponents for the BSC with crossover probability $p_c = 0.09$.

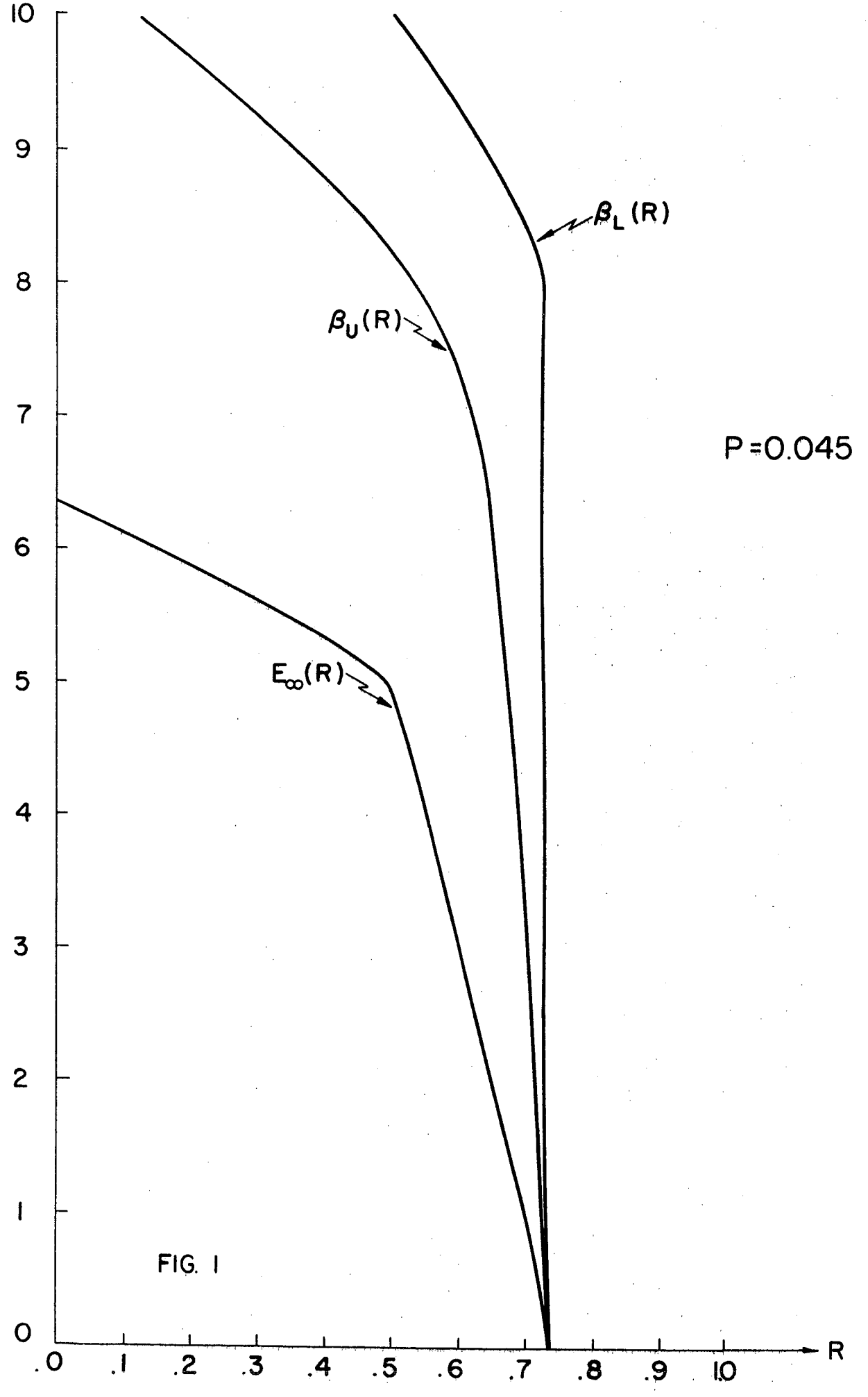


FIG. 1

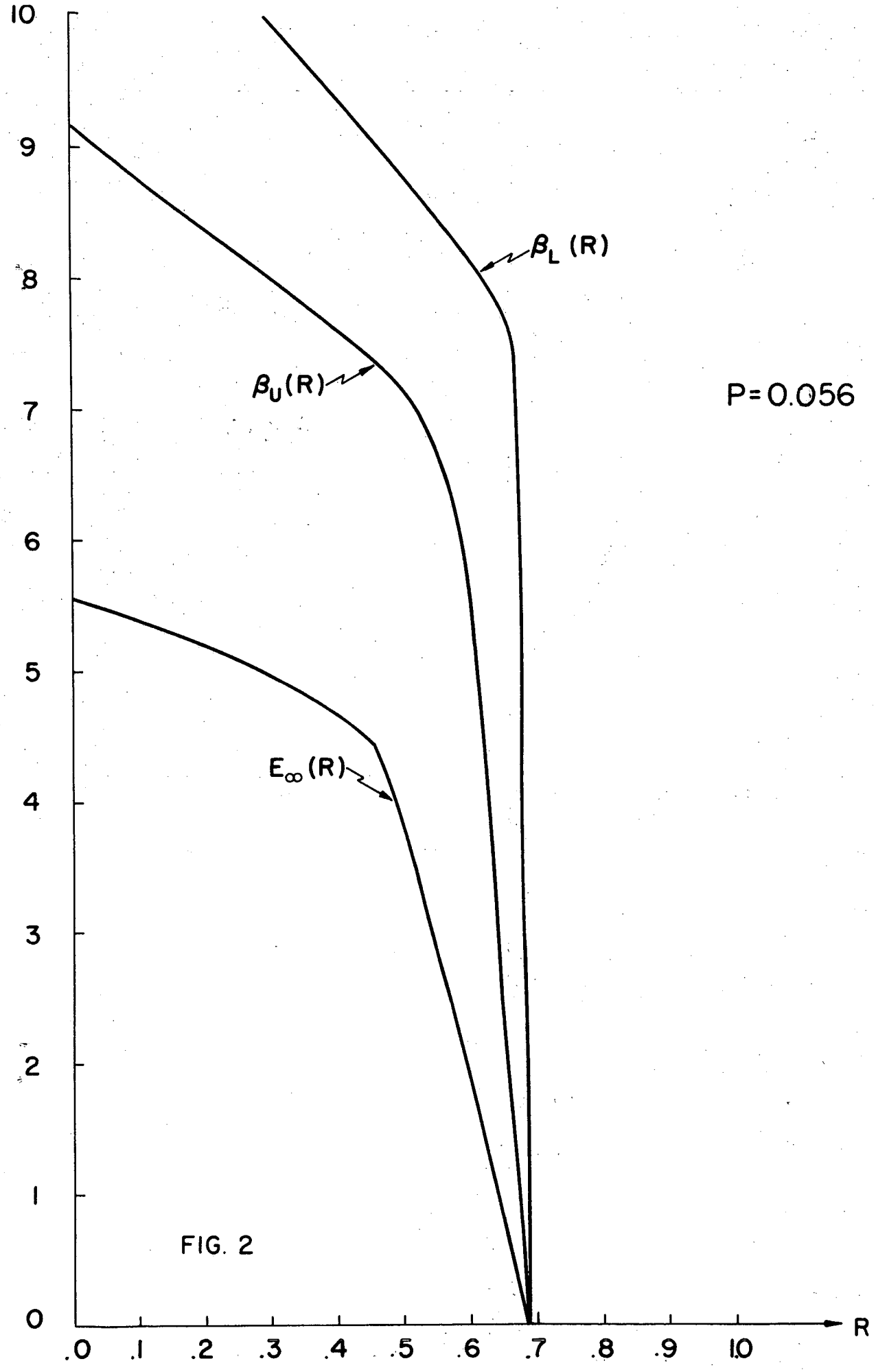


FIG. 2

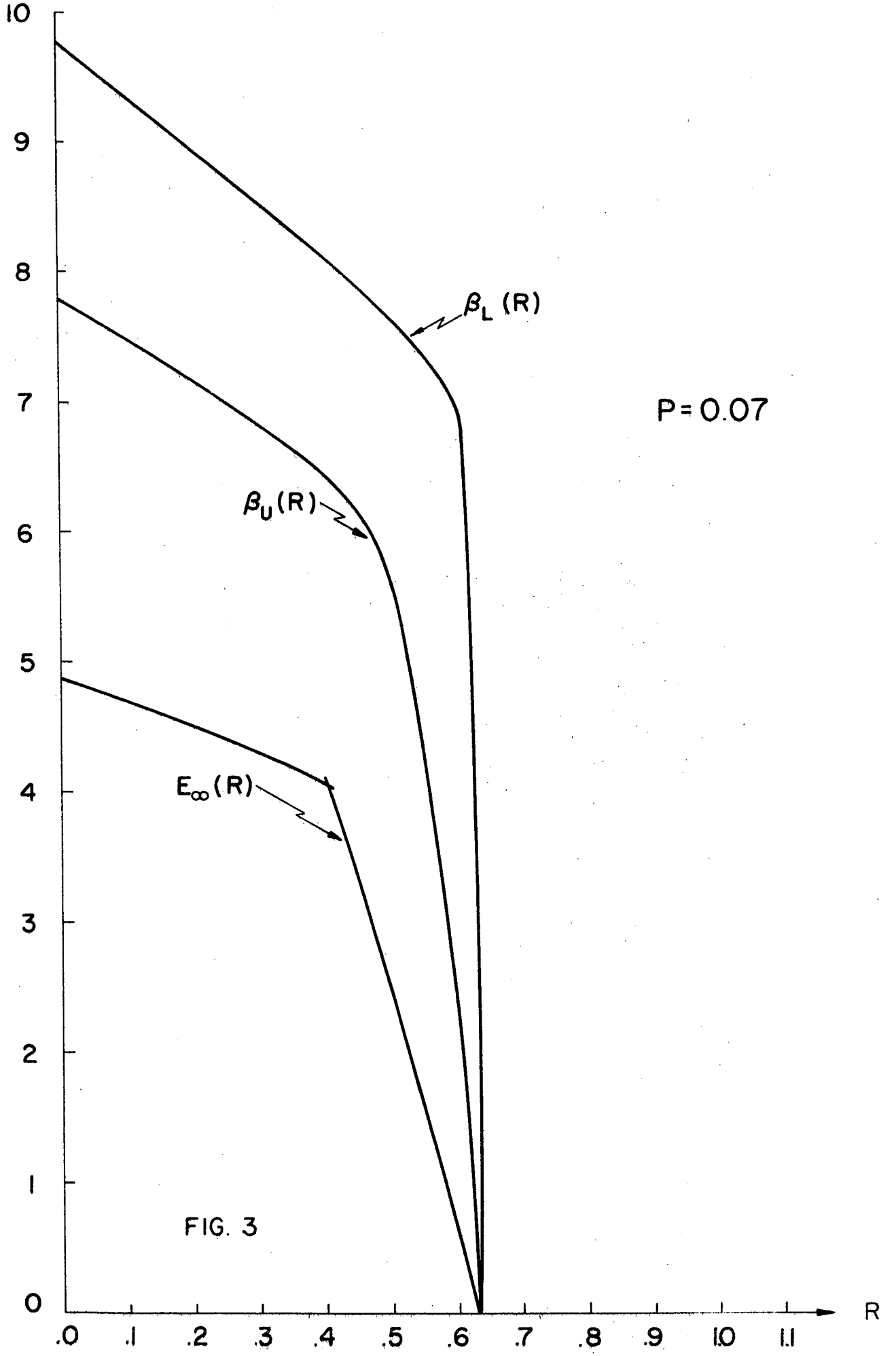

10
9
8
7
6
5
4
3
2
1
0
β_L (R)
β_U(R)
E_∞(R)
P = 0.07
R
.0
.1
.2
.3
.4
.5
.6
.7
.8
.9
1.0
1.1

FIG. 3

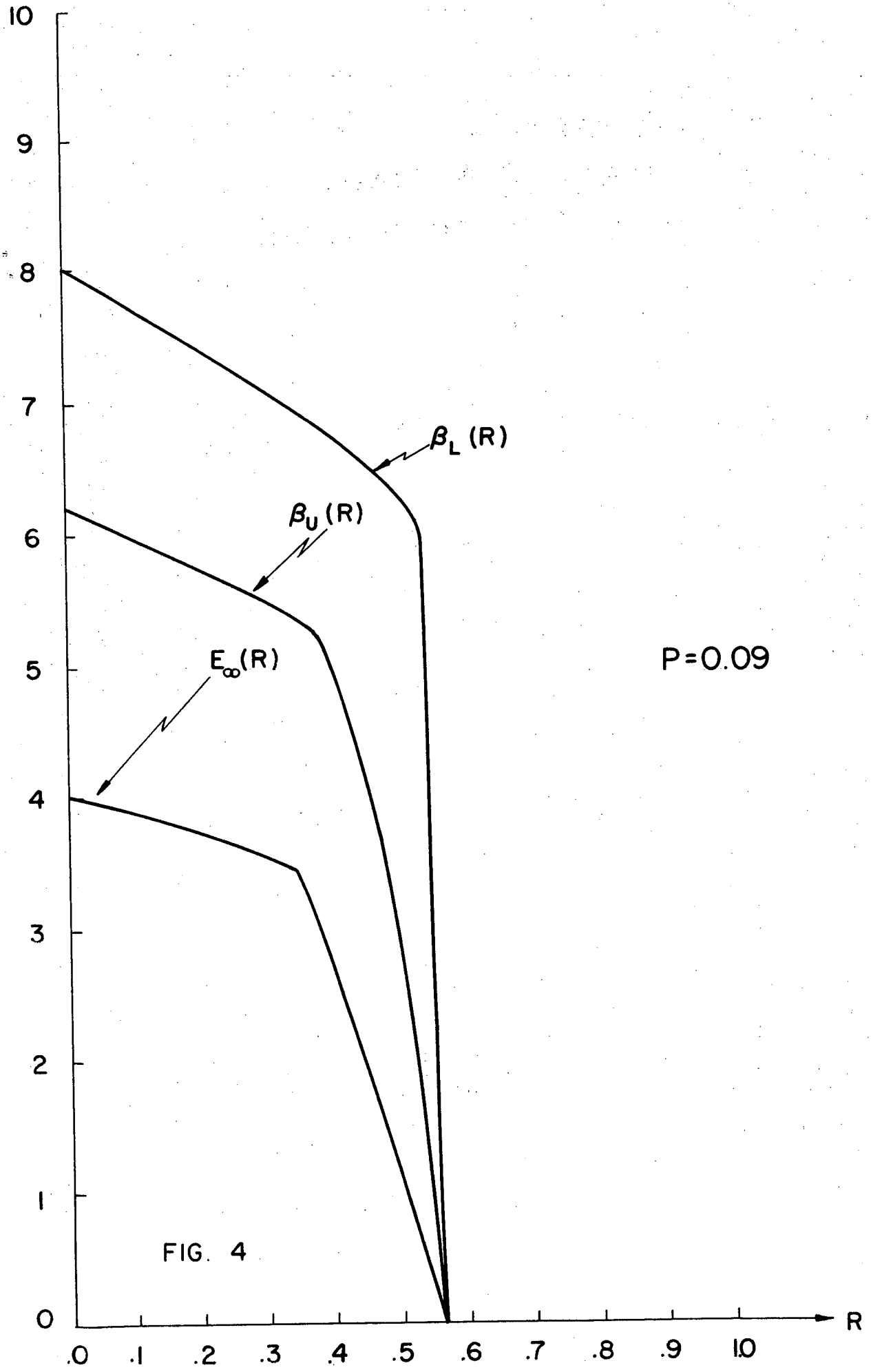


FIG. 4

II-F. Three Group Bootstrap Decoding

1. Description of Code and Its Use in Bootstrapping

It is desirable to generalize bootstrap decoding to encode transmitted streams by use of an algebraic code that has more than one parity check. The three-goup code has two parity check digits v_1, v_2 and k information digits $m_1, \ldots, m_k$. Every information digit is checked by at least one parity check digit. Without loss of generality let

$$v_1 = \sum_{i=1}^{\ell-1} m_i \qquad 1 \leq \ell - 1 \leq k$$

$$v_2 = \sum_{i=h}^{k} m_i \qquad 1 \leq h \leq k \tag{1}$$

For the code to be non-trivial, $1 \leq h \leq \ell-1 \leq k$ and at least one of the outside inequalities is strict. It is convenient to define the codeword digits, $x_1, \ldots, x_{k+2}$ as follows:

$$x_1 = v_1$$

$$x_i = m_{i-1} \qquad i = 2, \ldots, k+1 \tag{2}$$

$$x_{k+2} = v_2$$

The codeword digits may then be divided into three groups

$$\mathcal{G}_1 = \{x_1, \ldots, x_h\}, \quad \mathcal{G}_2 = \{x_{h+1}, \ldots, x_\ell\}, \quad \mathcal{G}_3 = \{x_{\ell+1}, \ldots, x_n\} \tag{3}$$

where $n=k+2$. Let $y = y_1, \ldots, y_n$ be the received digits, and define

$$t_1 \triangleq \sum_{i=1}^{h} y_i \qquad u_1 \triangleq \sum_{i=1}^{h} x_i$$

$$t_2 \triangleq \sum_{i=h+1}^{\ell} y_i \qquad u_2 \triangleq \sum_{i=h+1}^{\ell} x_i \qquad (4)$$

$$t_3 \triangleq \sum_{i=\ell+1}^{n} y_i \qquad u_3 \triangleq \sum_{i=\ell+1}^{n} x_i$$

Then the syndrome digits of $\underset{\sim}{y}$ are

$$s_1 = t_1 \oplus t_2 \quad \text{and} \quad s_2 = t_2 \oplus t_3 \qquad (5)$$

Let $\underset{\sim}{u} = (u_1,u_2,u_3)$, $\underset{\sim}{t} = (t_1,t_2,t_3)$, $n_1 = h$, $n_2 = \ell - h$, $n_3 = n - \ell$. Assuming that the information digits $m_1,\ldots,m_k$ are i.i.d. with $P(m_i = 0) = P\{m_i = 1\} = 1/2$, then

$$P\{\underset{\sim}{u} = 0,0,0\} = P\{\underset{\sim}{u} = 1,\ 1,1\} = 1/2 \qquad (6)$$

Now for $n_i \geq 1$,

$$q_{n_i}(0) \triangleq P\{t_i = u_i \mid u_i\} = P = \left\{ \begin{array}{l} \text{an even number of } n_i \\ \text{digits were received} \\ \text{incorrectly through} \\ \text{the channel} \end{array} \right\} = \frac{1+(1-2p)^{n_i}}{2} \qquad (7a)$$

where p is the channel crossover probability. As a consequence,

$$q_{n_i}(1) \triangleq P\{t_1 \neq u_i \mid u_i\} = \frac{1-(1-2p)^{n_i}}{2} \qquad (7b)$$

It will prove convenient to also define

$$q_o(0) \triangleq 1 \quad , \quad q_o(1) \triangleq 0 \qquad (8)$$

From the above, we get the relation

$$P\{t_1,t_2,t_3\} = \frac{1}{2}\left[q_{n_1}(t_1)\ q_{n_2}(t_2)\ q_{n_3}(t_3) + q_{n_1}(t_1 \oplus 1)\ q_{n_2}(t_2 \oplus 1)\ q_{n_3}(t_3 \oplus 1)\right] \qquad (9)$$

where $\oplus$ denotes mod 2 summation, and $n_j \geq 1$.

We will be able to show below that for the three group code,

$$\frac{P\{\underset{\sim}{y} / x_i\}}{P\{\underset{\sim}{y}\}} = \frac{P\{y_i, t_1, t_2, t_3 / x_i\}}{P\{y_i, t_1, t_2, t_3\}} \quad \text{if } n_j \geq 1 \tag{10}$$

Since the left-hand ratio is the one that enters into the likelihood calculation for bootstrap decoding, the receiver will be interested in probabilities $P\{y_i, t_1, t_2, t_3 / x_i\}$. Suppose $i \in [1, \ldots, n_1]$. Then

$$P\{y_i, \underset{\sim}{t} / x_i\} = \sum_{\underset{\sim}{u}} P\{y_i, \underset{\sim}{t}, \underset{\sim}{u} / x_i\} = \sum_{\underset{\sim}{u}} P\{y_i, \underset{\sim}{t} / \underset{\sim}{u}, x_i\} P\{\underset{\sim}{u} / x_i\}$$

$$= \sum_{\underset{\sim}{u}} P\{y_i, t_1 / x_i, u_1\} P\{t_2 / u_2\} P\{t_3 / u_3\} P\{\underset{\sim}{u} / x_i\} \tag{11}$$

But

$$P\{\underset{\sim}{u} / x_i\} = P\{u_1 / x_i\} P\{u_2, u_3 / u_1\} = P\{u_1 / x_i\} \delta(u_2, u_1) \delta(u_3, u_1) \tag{12}$$

[where $\delta(\ ,\ )$ is the Kronecker delta function], and

$$P\{u_1 / x_i\} = \begin{cases} \frac{1}{2} & n_1 > 1 \\ \delta(u_1, x_i) & n_1 = 1 \end{cases} \tag{13}$$

Furthermore, for $n_1 > 1$ (t_1' and u_1' are the sums over the first group excluding the i^{th} variable),

$$P\{y_i, t_1 / x_i, u_1\} = w(y_i / x_i) P\{t_1' = t_1 \oplus y_i / u_1' = u_1 \oplus x_i\} =$$

$$= w(y_i / x_i) \; q_{n_1 - 1}(t_1 \oplus u_1 \oplus y_i \oplus x_i) \tag{14}$$

Thus it follows from (11) through (14) that as long as $n_1 > 1$, $n_2 \geq 1$, $n_3 \geq 1$, then

$$P\{y_i, \underset{\sim}{t} / x_i\} = \sum_{\underset{\sim}{u}} \Big[\frac{1}{2} \; q_{n_1-1}(t_1 \oplus u_1 \oplus y_i \oplus x_i) \; q_{n_2}(t_2 \oplus u_2) \; q_{n_3}(t_3 \oplus u_3)$$

$$w(y_i/x_i) \; \delta(u_2, u_1) \; \delta(u_3, u_1) \Big] =$$

$$= \frac{1}{2} \; w(y_i/x_i) \Big[q_{n_1-1}(t_1 \oplus y_i \oplus x_i) \; q_{n_2}(t_2) \; q_{n_3}(t_3) +$$

$$+ \; q_{n_1-1}(t_1 \oplus y_i \oplus x_i \oplus 1) \; q_{n_2}(t_2 \oplus 1) \; q_{n_3}(t_3 \oplus 1) \Big] \qquad (15)$$

As a consequence,

$$P\{y_i, \underset{\sim}{t}\} = \frac{1}{2} \Big[P\{y_i, \underset{\sim}{t}/0\} + P\{y_i, \underset{\sim}{t}/1\} \Big] =$$

$$\frac{1}{4} \Big(\Big[w(y_i/0) \; q_{n_1-1}(t_1 + y_i) + w(y_i/1) \; q_{n_1-1}(t_1 \oplus y_i \oplus 1) \Big]$$

$$q_{n_2}(t_2) \qquad q_{n_3}(t_3) \qquad +$$

$$+ \Big[w(y_i/0) \; q_{n_1-1}(t_1 \oplus y_i \oplus 1) + w(y_i/1) \; q_{n_1-1}(t_1 \oplus y_i) \Big] \; q_{n_2}(t_2 \oplus 1)$$

$$q_{n_3}(t_3 \oplus 1) \Big)$$

$$= \frac{1}{4} \Big[q_{n_1}(t_1) \; q_{n_2}(t_2) \; q_{n_3}(t_3) + q_{n_1}(t_1 \oplus 1) \; q_{n_2}(t_2 \oplus 1) \; q_{n_3}(t_3 \oplus 1) \Big] \qquad (16)$$

so that for $n_1 > 1$, $n_2 \geq 1$, $n_3 \geq 1$,

$$\frac{P\{y_i, \underset{\sim}{t}/x_i\}}{P\{y_i, \underset{\sim}{t}\}} =$$

$$2w(y_i/x_i) \; \frac{q_{n_1-1}(t_1 \oplus y_i \oplus x_i) \; q_{n_2}(t_2) \; q_{n_3}(t_3) + q_{n_1-1}(t_1 \oplus y_i \oplus x_i \oplus 1) \; q_{n_2}(t_2 \oplus 1) \; q_{n_3}(t_3 \oplus 1)}{q_{n_1}(t_1) \; q_{n_2}(t_2) \; q_{n_3}(t_3) + q_{n_1}(t_1 \oplus 1) \; q_{n_2}(t_2 \oplus 1) \; q_{n_3}(t_3 \oplus 1)} \qquad (17)$$

Since for $n_1 = 1$,

$$P\{y_i, t_1 / x_i, u_1\} = \begin{cases} w(y_i/x_i) & \text{if } y_i = t_1 \\ 0 & \text{if } y_i \neq t_1 \end{cases}$$

then

$$P\{y_i, \underset{\sim}{t}/x_i\} = \begin{cases} w(y_i/x_i)\, q_{n_2}(t_2 \oplus x_i)\, q_{n_3}(t_3 \oplus x_i) & \text{if } y_i = t_1 \\ 0 & \text{if } y_i \neq t_1 \end{cases} \tag{18}$$

Assuming the case $t_1 \oplus y_i = 0$, then

$$q_1(t_1 \oplus x_i) = w(y_i/x_i)$$

and $t_1 \oplus y_i \oplus x_i = x_i$. Thus (17) is valid if $n_i \geq 1$, provided definition (8) is used.

Relation (15) was obtained under the assumption that $i \epsilon [1, \ldots, n_1]$. If $n_1 + 1 \leq i \leq n_1 + n_2$, we need only interchange n_1 and t_1 with n_2 and t_2 in (15). The interchange of n_1 and t_1 with n_3 and t_3 preserves the validity of (15) fo $n_1 + n_2 + 1 \leq i \leq n$.

It follows from (9), (10), and (15) that if $n_i \geq 1$. the likelihood used in bootstrap decoding with a three group algebraic code is a function of y_i, x_i, and the state variables $(t_1, t_2, t_3, n_1, n_2, n_3)$. We will see that these variables will also be sufficient if all the digits of one or two of the three groups have been decoded. The needed adjustment of the state variable values as the decoding proceeds is as follows:

At the beginning, when no digit in a column has yet been decoded, (15) and (16) are used directly. Suppose, w.l.o.g., that y_1 is decoded

as x_1. Then a new $t_1' = t_1 \oplus x_1 \oplus y_1$ is obtained and used in (15) and (16) with n_1 replaced by n_1-1. This process continues until all digits of some set $\mathcal{J}_i$ have been decoded. W.l.o.g. assume that such a set is $\mathcal{J}_2$, that the new t-values are t_1, t_2, t_3 and that n_1 and n_3 digits remain undecoded in $\mathcal{J}_1$ and $\mathcal{J}_3$. Assuming that no error was committed, $t_2 = u_2$, and when decoding y_i for $1 \leq i \leq n_1$ the value of t_3 becomes irrelevant and only those of t_1 and t_2 count. Thus the numerator in (10) is replaced by

$$P\{y_i, \underset{\sim}{t} \,/\, x_i, u_2\} = P\{y_i, t_1 \,/\, x_i, u_2 = t_2\} = \tag{18a}$$

$$= P\{y_i, t_1 \,/\, x_i, u_1 = t_2\} = w(y_i \,/\, x_i)\, q_{n_1-1}(t_2 \oplus t_1 \oplus x_i \oplus y_i)$$

for $n_1 > 1$. Similarly, the denominator of (10) is replaced by

$$P\{\underset{\sim}{t}, y_i \,/\, u_2 = t_2\} = P\{t_1, y_i \,/\, u_1 = t_2\} = \tfrac{1}{2}\, q_{n_1}(t_2 \oplus t_1) \tag{18b}$$

When $n_1 = 1$, the remaining y_i can be decoded algebraically from the relation

$$x_i = t_1 \oplus t_2 \oplus y_i = t_2 \tag{19}$$

We now observe from formulas (15) and (18a) that if in the former we set $n_2 = 0$ and use definition (8), we get the relation

$$P\{y_i, \underset{\sim}{t}/x_i\} = \tfrac{1}{2}\, w(y_i/x_i)\, q_{n_1-1}(t_1 \oplus t_2 \oplus y_i \oplus x_i)\, q_{n_3}(t_3 \oplus t_2)$$

$$= \tfrac{1}{2}\, q_{n_3}(t_3 \oplus t_2)\, P\{y_i, \underset{\sim}{t}/\, x_i, u_2\} \tag{20}$$

Similarly, setting $n_2 = 0$ in formula (16), we obtain

$$P\{y_i, \underset{\sim}{t}\} = \tfrac{1}{4}\, q_{n_1}(t_1 \oplus t_2)\, q_{n_3}(t_3 \oplus t_2) =$$

$$= \tfrac{1}{2}\, q_{n_3}(t_3 \oplus t_2)\, P\{y_i, \underset{\sim}{t}/\, u_2 = t_2\} \tag{21}$$

The ratio of probabilities (20) to (21) is thus equal to the ratio (17) and the latter formula thus remains valid even if one of the groups is completely decoded. In fact, if $n_1 = 1$ and $n_2 = 0$, (17) becomes [note that $t_1 = y_i$ if $n_1 = 1$]

$$\frac{P\{y_i, \underset{\sim}{t}/x_i\}}{P\{y_i, \underset{\sim}{t}\}} = 2w(y_i/x_i) \quad \frac{q_0(t_1 \oplus t_2 \oplus y_i \oplus x_i)}{q_1(t_1 \oplus t_2)} = \begin{cases} 2 & \text{if } x_i = t_2 \\ 0 & \text{if } x_i \neq t_2 \end{cases} \tag{22}$$

so that straight-forward sequential decoding using the likelihood function based on (17) will force the decoder to select the path on which (19) is satisfied at each depth.

As seen from above, the value of t_3 is irrelevant once all digits of $\mathcal{S}_2$ were decoded and those of $\mathcal{S}_1$ are being decoded. Of course when the latter task is complete, decoding of $\mathcal{S}_3$ starts that will depend on t_3 and t_2 [note that since $u_1 \oplus u_2 = 0$ then $t_1 = t_2$ when $\mathcal{S}_1$ and $\mathcal{S}_2$ have been decoded] in the same way that the just described decoding was dependent on t_1 and t_2.

2. Proof of Formula (10)

Because of the symmetry of the situation, it is obviously sufficient to prove formula (10) for $i = n$. Let us define the set

$$\mathcal{V}(u_1,u_2,u_3) = \{x_1, \ldots, x_{n-1}: \sum_{i=1}^{n_1} x_i = u_1, \sum_{i=n_1+1}^{n_2} x_i = u_2, \sum_{i=n_1+n_2+1}^{n-1} x_i = u_3\} \tag{23}$$

Then

$$P\{\underset{\sim}{y}/x_n\} = w(y_n/x_n)\, 2^{-(n-1)} \left[\sum_{\underset{\sim}{x} \in \mathcal{V}(0,0,x_n)} \prod_{j=1}^{n-1} w(y_j/x_j) + \right.$$

$$+ \sum_{\underset{\sim}{x} \in \mathcal{V}(1,1,x_n \oplus 1)} \prod_{j=1}^{n-1} w(y_j/x_j) \Big] \qquad (24)$$

Using formula (9) of Jelinek and Cocke [1] and defining

$$f^+(y) = w(y/0) + w(y/1) = 1$$

$$f^-(y) = w(y/0) - w(y/1) = \begin{cases} 1 - 2p & \text{if } y = 0 \\ 2p - 1 & \text{if } y = 1 \end{cases} \qquad (25)$$

we get that

$$8 \sum_{\underset{\sim}{x} \in \mathcal{V}(0,0,x_n)} \prod_{j=1}^{n-1} w(y_j/x_j) = \left\{ \prod_{j=1}^{n_1} f^+(y_j) + \prod_{j=1}^{n_1} f^-(y_j) \right\} \cdot$$

$$\cdot \left\{ \prod_{j=n_1+1}^{n_1+n_2} f^+(y_j) + \prod_{j=n_1+1}^{n_1+n_2} f^-(y_j) \right\} \cdot \left\{ \prod_{j=n_1+n_2+1}^{n-1} f^+(y_j) + \right.$$

$$\left. + (-1)^{x_n} \prod_{j=n_1+n_2+1}^{n-1} f^-(y_j) \right\} =$$

$$= \left\{1+(-1)^{t_1} (1-2p)^{n_1}\right\}\left\{1+(-1)^{t_2} (1-2p)^{n_2}\right\}\left\{1+(-1)^{t_3+y_n+x_n} (1-2p)^{n_3-1}\right\}$$

$$= 8\, q_{n_1}(t_1)\, q_{n_2}(t_2)\, q_{n_3-1}(t_3 \oplus y_n \oplus x_n) \qquad (26)$$

where t_1, t_2, t_3 are given by (4). Similarly,

$$\sum_{\underset{\sim}{x} \in \mathcal{V}(1,1,x_n \oplus 1)} \prod_{j=1}^{n-1} w(y_j/x_j) = q_{n_1}(t_1 \oplus 1)\, q_{n_2}(t_2 \oplus 1)\, q_{n_3-1}(t_3 \oplus y_n \oplus x_n \oplus 1) \qquad (27)$$

It follows therefore from (24) , (26), (27), and (15) that

$$P\{\underset{\sim}{y} / x_n\} = w(y_n / x_n)\, 2^{-(n-1)} \left[q_{n_1}(t_1)\, q_{n_2}(t_2)\, q_{n_3-1}(t_3 \oplus y_n \oplus t_n) \right.$$

$$\left. + q_{n_1}(t_1 \oplus 1)\, q_{n_2}(t_2 \oplus 1)\, q_{n_3-1} \oplus y_n \oplus t_n \oplus 1) \right]$$

$$= 2^{-(n-2)}\, P\{y_n, \underset{\sim}{t} / x_n\} \qquad (28)$$

Averaging (28) over x_n results in

$$P\{\underset{\sim}{y}\} = 2^{-(n-2)}\, P\{y_n, \underset{\sim}{t}\} \qquad (29)$$

Formula (10) then follows from (28) and (29).

3. Description of Likelihood Table

Obviously, the likelihood

$$\log \frac{P\{\underset{\sim}{y} / x_i, \underset{\sim}{x}'\}}{P\{\underset{\sim}{y} / \underset{\sim}{x}'\}} - R \qquad (30)$$

(where $\underset{\sim}{x}'$ is the vector of digits already decoded) would not actually be computed from scratch during the process of bootstrap decoding based on the three group code. Rather, the values of (30) would be stored in a table whose arguments would be the parameters

$$x_i \oplus y_i,\ t_1, t_2, t_3\ n_1, n_2, n_3, h \qquad (31)$$

where h denotes the group membership of x_i (i.e., $x_i \in \mathcal{S}_h$), n_j denotes the number of digits in the j^{th} group still left to be decoded, and t_j denotes the adjusted mod 2 sum of the j^{th} received group (i.e., if the digits $x_{i_1}, \ldots, x_{i_\ell}$ of $\mathcal{S}_j$ have been decoded and $y_{m_1}, \ldots, y_{m_r}$ are yet to be decoded then $t_j = \sum_{s=1}^{\ell} x_{i_s} \oplus \sum_{s=1}^{r} y_{m_s}$).

The table would be computed with the help of formula (17). Obviously, it would contain a lot of symmetries which could be eliminated if storage was a factor. For instance, the parameter h of (31) is not needed if by convention y_i and x_i are always members of the first group. The likelihood would then be of the form

$$\lambda(x_i \oplus y_i, t_1, t_2, t_3\ n_1, n_2, n_3) \qquad (32)$$

with the first four parameters binary. A further reduction in storage size is attainable by noting that (32) is invariant to an interchange of (t_2, n_2) with (t_3, n_3).

4. Decoding Strategy of the Bootstrap Algorithm

The convolutionally encoded streams belong to three groups. By convention $n_1 \leq n_2 \leq n_3$. There is a parameter KRANK (J) which ranks the groups in "desirability" of decoding. At the start KRANK (J) = J. The general idea is to work on all streams of KRANK(1) until they have either been all successfully decoded or until everyone of those streams of KRANK(1) that have not been decoded has been attempted (in sequence) without success. In the latter case streams of KRANK(2) are tried, and if this fails then streams of KRANK(3). In case of such a "complete" failure, another decoding attempt is made with increased values of the ISTOP and KSTACK parameters. As soon as any stream of some group LNOW is decoded, KRANK(1) is set equal to LNOW, and KRANK(2) is set equal to that remaining group that has the smallest number of undecoded streams. The last group is then labeled KRANK(3).

Originally, the parameter KPHASE is set equal to 1. When a group has been completely decoded, KPHASE is set equal to 2, KRANK(3) = LNOW, and KRANK(1) is set equal to that remaining group that has the smallest number of undecoded streams. When two groups have been completely decoded, KPHASE is set equal to 3, KRANK(1) = KRANK(2), KRANK(2) = KRANK(3), KRANK(3) = LNOW.

A decoding attempt on a stream is "successful" if depth LTRACK was reached by the decoder. In this case all digits of that stream are considered definitely decoded. Otherwise the attempt is "unsuccessful" and digits up to depth IMAX - LBACK are considered definitely decoded. If a decoding error takes place the algorithm halts and an UNSUCCESSFUL CONCLUSION is declared.

To aid in the understanding of the Fortran listing of the algorithm we give a glossary of some key parameters that are peeuliar to the three-group bootstrap scheme.

LNOW - current group being decoded.

LNOW 2, LNOW 3 - the other two groups

KPHASE = 1 + number of completely decoded groups

KRANK(J) - The J^{th} most "desirable" group. Originally LNOW = KRANK(1). Also if a stream is completely decoded, the group to which it belongs, LNOW, becomes the most desirable one, i.e., KRANK(1) = LNOW. The remaining order is that of group size if KPHASE = 1. If KPHASE = 2, then KRANK(2) is equal to the other undecoded group.

KLEFT(I) - number of undecoded streams within the $I^{\underline{th}}$ group.

KNEXT - the order of the stream within the group LNOW which is to be decoded next.

LGRP - is the order of the group currently decoded, i.e., LNOW = KRANK(LGRP) ($1 \leq$ LGRP $\leq$ 4-KPHASE)

KROUND - number of streams within the group that the decoder attempted to decode without success since the last change of LGRP.

LROUND - number of times LGRP attained its maximal value without the decoding of any of the attempted streams advancing by more than LBACK + 40 branches.

5. An Upper Bound on the Moments of the Decoding Effort for Three Group Bootstrap Decoding

The analysis of this section was developed by D. Costello while he was a research associate employed by the contract.

Jelinek and Cocke[1] have developed an upper bound on the moments of the decoding effort for bootstrap decoding using a single parity stream. We will extend that analysis to the case of three group bootstrap decoding. Emphasis will be placed on those portions of the argument which differ from the original argument. In addition, for simplicity's sake we will restrict attention to the BSC.

First of all, assume there are n_i streams left to be decoded in group i, i = 1, 2, 3. Then let $N_i(\underline{n})$ be the number of steps necessary to decode any given stream in group i when the step allocation is M = 1 and $\underline{n} = (n_1, n_2, n_3)$. Applying well known results about ordinary sequential decoding, we can conclude that

$$P\left\{N_i(\underline{n}) > \ell\right\} \leq K(R,\nu)(\Gamma + t)\ell^{-\sigma_i} \qquad (1)$$

where $\Gamma + t$ is the length of the information sequence and $K(R,\nu)$ is a function of rate R and constraint length ν which is finite if ν is finite and σ_i satisfies

$$R < \frac{E_{\underline{n}}(\sigma_i)}{\sigma_i} \qquad \text{for } R \geq \frac{E_{\underline{n}}(2)}{2}$$

or (2)

$$R < \frac{E_{\underline{n}}(2)}{\sigma_i} \qquad \text{for } R \leq \frac{E_{\underline{n}}(2)}{2} \quad .$$

In (2), $E_{\underline{n}}(\sigma_3)$ is the concave, positive, increasing function of σ_3 defined as follows:

Let $k = n_1 + n_2 + n_3$ and label the k received digits left to be decoded as $y_1, y_2, \ldots, y_{n_1}, y_{n_1+1}, \ldots, y_{n_1+n_2}, y_{n_1+n_2+1}, \ldots, y_{n_1+n_2+n_3} = y_k$. Define $\underline{y} = (y_1, \ldots, y_k)$ and $\tilde{\underline{y}} = (y_1, \ldots, y_{k-1})$ and assume that the k^{th} stream is in group 3 to be decoded. Then

$$E_{\underline{n}}(\sigma_3) = 1 + \sigma - \log \left[\sum_{\underline{y}} \left[\sum_{x_k=o} P(\underline{y}|x_k)^{1/1+\sigma} \right]^{1+\sigma} \right]$$

$$= 1+\sigma-\log 2^{k-3} \sum_{y_k,\underline{s}} \left[\sum_{x_k=o}^{1} \left\langle P(y_k,\underline{s}|x_k) P(\tilde{\underline{y}}|x_k y_k,\underline{s}) \right\rangle^{1/1+\sigma} \right]^{1+\sigma} \tag{3}$$

since $P(\underline{y}|x_k)$ depends only on y_k and the pair of syndrome (state) digits $\underline{s} = (s_1, s_2)$. Noting that $P(\tilde{\underline{y}}|x_k, y_k, \underline{s}) = P(\tilde{\underline{y}}|y_k, \underline{s}) = 2^{-(k-3)}$, and substituting for $P(y_k, \underline{s}|x_k)$ from the analysis of the three group code, we obtain the rather long but straightforward expression

$$\begin{aligned}
E_{\underline{n}}(\sigma_3) = \sigma_3 - \log \Bigg[& \Bigg(\Big\{ (1-p) \Big[q_{n_1}(0) q_{n_2}(0) q_{n_3-1}(0) + q_{n_1}(1) q_{n_2}(1) q_{n_3-1}(1) \Big] \Big\}^{1/1+\sigma_3} \\
& + \Big\{ p \Big[q_{n_1}(0) q_{n_2}(0) q_{n_3-1}(1) + q_{n_1}(1) q_{n_2}(1) q_{n_3-1}(0) \Big] \Big\}^{1/1+\sigma_3} \Bigg)^{1+\sigma_3} \\
& + \Bigg(\Big\{ (1-p) \Big[q_{n_1}(0) q_{n_2}(0) q_{n_3-1}(1) + q_{n_1}(1) q_{n_2}(1) q_{n_3-1}(0) \Big] \Big\}^{1/1+\sigma_3} \\
& + \Big\{ p \Big[q_{n_1}(0) q_{n_2}(0) q_{n_3-1}(0) + q_{n_1}(1) q_{n_2}(1) q_{n_3-1}(1) \Big] \Big\}^{1/1+\sigma_3} \Bigg)^{1+\sigma_3} \\
& + \Bigg(\Big\{ (1-p) \Big[q_{n_1}(1) q_{n_2}(0) q_{n_3-1}(0) + q_{n_1}(0) q_{n_2}(1) q_{n_3-1}(1) \Big] \Big\}^{1/1+\sigma_3} \\
& + \Big\{ p \Big[q_{n_1}(1) q_{n_2}(0) q_{n_3-1}(1) + q_{n_1}(0) q_{n_2}(1) q_{n_3-1}(0) \Big] \Big\}^{1/1+\sigma_3} \Bigg)^{1+\sigma_3} \\
& + \Bigg(\Big\{ (1-p) \Big[q_{n_1}(1) q_{n_2}(0) q_{n_3-1}(1) + q_{n_1}(0) q_{n_2}(1) q_{n_3-1}(0) \Big] \Big\}^{1/1+\sigma_3} \\
& + \Big\{ p \Big[q_{n_1}(1) q_{n_2}(0) q_{n_3-1}(0) + q_{n_1}(0) q_{n_2}(1) q_{n_3-1}(1) \Big] \Big\}^{1/1+\sigma_3} \Bigg)^{1+\sigma_3} \Bigg]
\end{aligned} \tag{4}$$

where p is the channel crossover probability and

$$q_{n_i}(0) = \frac{1+(1-2p)^{n_i}}{2} \qquad q_{n_1}(1) = \frac{1-(1-2p)^{n_i}}{2} \quad . \tag{5}$$

Clearly, $E_{\underline{n}}(\sigma_1)$ and $E_{\underline{n}}(\sigma_2)$ are defined in a similar way.

Next let $\sigma_i(\underline{n})$ be the least upper bound on the numbers σ_i satisfying (2), i.e., $\sigma_i(n)$ is the solution of

$$\begin{aligned} R &= \frac{E_{\underline{n}}(\sigma_i(\underline{n}))}{\sigma_i(\underline{n})} && \text{for } \frac{E_{\underline{n}}(2)}{2} \leqq R < C \\ \sigma_i(\underline{n}) &= \frac{E_{\underline{n}}(2)}{R} && \text{for } 0 < R < \frac{E_{\underline{n}}(2)}{2} \quad . \end{aligned} \tag{6}$$

Now choose k(R) to be the largest positive integer such that

$$k(R)\ \sigma(\infty) < \min_{\mathcal{S}} \left\{ \max \left[\sigma_1(\underline{n}),\ \sigma_2(\underline{n}),\ \sigma_3(\underline{n}) \right] \right\} \tag{7}$$

where $\mathcal{S} = \left\{ \underline{n} = (n_1, n_2, n_3) \mid n_1+n_2+n_3 = k(R) \right\}$ and $\sigma(\infty)$ is the Pareto exponent which would be obtained with ordinary sequential decoding, i.e., $\sigma(\infty)$ is the solution of

$$\begin{aligned} R &= \frac{E_\infty(\sigma(\infty))}{\sigma(\infty)} && \text{for } \frac{E_\infty(2)}{2} \leqq R < C \\ \sigma(\infty) &= \frac{E_\infty(2)}{R} && \text{for } 0 < R < \frac{E_\infty(2)}{2} \end{aligned} \tag{8}$$

where

$$E_\infty(\sigma) = \sigma - \log \left[(1-p)^{1/1+\sigma} + p^{1/1+\sigma} \right]^{1+\sigma} \quad . \tag{9}$$

If there are originally m streams of digits to decode, we wish to modify the three group bootstrap decoding algorithm as follows:

(1) Decode m-k(R) streams by ordinary sequential decoding without the help of the two parity streams and with the step allocation M = 1.

(2) Decode the remaining k(R) streams with the help of the parity streams using the three group bootstrap decoding algorithm.

We now briefly highlight the arguments leading to the desired bound. The details will not be pursued since they closely follow the development in Jelinek and Cocke[1]. In part (1) of the modified algorithm, the easiest m-k(R) streams are decoded by ordinary sequential decoding. If L^* is the number of steps needed to decode the hardest of the decoded streams, then $P(L^* > \ell)$ is upper bounded by the probability that there is a set of k(R) + 1 streams that need more than ℓ steps each to decode by ordinary sequential decoding. Since the decoding of the first m-k(R) streams is independent, the γth computational moment of the decoding effort in part (1) is bounded if $(k(R)+1)\,\sigma(\infty) > \gamma$.

In part (2) of the modified algorithm, we compute the three Pareto exponents $\sigma_1(\underline{n})$, $\sigma_2(\underline{n})$, and $\sigma_3(\underline{n})$ given that decoding starts in group 1, group 2, or group 3. We then begin decoding in the group with the largest exponent. After decoding each stream, this procedure is repeated, thereby assuring that each successive stream is easier to decode than the previous one. If L(k(R)) is the number of steps needed to decode at least one of the k(R) remaining streams, then $P\left[L(k(R)) > \ell\right]$ is upper bounded by the probability that there is a set of k(R) streams that need more than ℓ steps each to decode by the three group bootstrap

decoding algorithm. Since the decoding of the last k(R) streams is not independent, the γth computational moment of the decoding effort in part (2) is bounded if $\max\left[\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n})\right] > \gamma$.

In bounding the decoding effort for the complete modified algorithm, we must consider the fact that after the first m-k(R) streams have been decoded any of the situations in the set $\mathcal{S}$ may describe the distribution of the remaining k(R) streams. Since in part (1), we decode the m-k(R) easiest streams, we are not free to choose the situation which would give us the best Pareto exponent for part (2). Hence the worst case must be assumed, and the bounding condition in part (2) minimized over all situations in $\mathcal{S}$.

Finally, since the decoding effort must be bounded for both part (1) and part (2), the γth computational moment of the decoding effort is bounded if $\min\left\{(k(R)+1)\sigma(\infty), \min_{\mathcal{S}}\left[\max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))\right]\right\} > \gamma$. We can now summarize as follows:

Theorem: The modified three group bootstrap decoding algorithm leads to a finite γth moment of computation per decoded digit if

$$\min\left\{(k(R)+1)\sigma(\infty), \min_{\mathcal{S}}\left[\max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))\right]\right\} > \gamma \quad (10)$$

where k(R) is the unique integer satisfying (7), $\sigma(\infty)$ is the unique solution of (8), and $\sigma_i(\underline{n})$ is the unique solution of (6), i = 1, 2, 3.

It is necessary to derive the above bound in terms of a modified decoding algorithm due to the difficulties involved in taking the dependencies of the bootstrap algorithm into account. It should also be noted that this is the essential difference between the bounding technique in part (2) of the modified

algorithm and that used in part (1). In the latter case the decoding is independent and we were able to obtain a tight bound on the decoding effort. However in part (2), the decoding is dependent, and we were forced to upper bound the probability that there is a set of k(R) streams that need more than ℓ steps to decode.

Now define $R^L_{boot}(\gamma)$ as the supremem of rates for which (10) is satisfied. Since the average computation will be bounded for the three group bootstrap decoding algorithm if $R < R^L_{boot}$ (1), $R^L_{boot}(1)$ is a lower bound on the R_{comp} of this decoding scheme.

We can evaluate $R^L_{boot}(\gamma)$ by computing the differences

$$\frac{1}{\gamma} \min_{\mathcal{S}} \left[\max\left(E_{\underline{n}}(\sigma_1)\Big|_{\sigma_1=\gamma}, E_{\underline{n}}(\sigma_2)\Big|_{\sigma_2=\gamma}, E_{\underline{n}}(\sigma_3)\Big|_{\sigma_3=\gamma} \right) \right] - kE_\infty\left(\frac{\gamma}{k}\right) \quad (11)$$

for k = 3,4, ... until their value becomes negative, where $\mathcal{S} = \left\{ \underline{n} = (n_1, n_2, n_3) \mid n_1 + n_2 + n_3 = k \right\}$. If this takes place for $k = k^+$, then

$$R^L_{boot}(\gamma) = \min\left\{ \frac{1}{\gamma} \min_{\mathcal{S}} \left[\max\left(E_{\underline{n}}(\sigma_1)\Big|_{\sigma_1=\gamma}, E_{\underline{n}}(\sigma_2)\Big|_{\sigma_2=\gamma}, E_{\underline{n}}(\sigma_3)\Big|_{\sigma_3=\gamma} \right) \right], \frac{k^+}{\gamma} E_\infty\left(\frac{\gamma}{k^+}\right) \right\} \quad (12)$$

where $\mathcal{S} = \left\{ \underline{n} = (n_1, n_2, n_3) \mid n_1 + n_2 + n_3 = k^+ \right\}$.

It remains to specify the elements of the set $\mathcal{S}$. Assume that m is a multiple of 3 and that the original distribution of the streams is $n_1 = n_2 = n_3 = m/3$. The problem is to specify the number of ways of arranging k(R) streams into 3 groups of size n_1, n_2, and n_3 respectively such that $n_1 + n_2 + n_3 = k(R)$ and n_1, n_2, and n_3 are always less than or equal to m/3. We

will not consider a relabeling of the groups to constitute an additional member of $\mathcal{S}$, since the labeling of the groups in the bootstrap decoding algorithm is immaterial.

First consider the number of ways $\mathcal{S}^*$ of arranging n_1, n_2, and n_3 such that $n_1 + n_2 + n_3 = k(R)$ without any restrictions on the size of the groups. We can easily deduce that

$$\mathcal{S}^* = \frac{\sum_{j=1}^{k(R)+1} j - 3\left\lceil \frac{k(R)}{2} \right\rceil - 1}{6} + \left\lceil \frac{k(R)+2}{2} \right\rceil \quad \text{if } 3 \mid k(R)$$

$$\mathcal{S}^* = \frac{\sum_{j=1}^{k(R)+1} j - 3\left\lfloor \frac{k(R)+2}{2} \right\rfloor}{6} + \left\lceil \frac{k(R)+2}{2} \right\rceil \quad \text{if } 3 \nmid k(R) \tag{13}$$

where $3 \mid k(R)$ means "3 divides k(R)", $3 \nmid k(R)$ means "3 does not divided k(R)", and $\lceil I \rceil$ is the largest integer less than or equal to I.

Now consider the limits placed on n_1, n_2, and n_3, viz., that they cannot exceed m/3. Letting $\#\mathcal{S}$ be the size of the set $\mathcal{S}$, we arrive at the following formulas:

Case 1. For $1 \leqq k(R) \leqq m/3$,

$$\#\mathcal{S} = \mathcal{S}^* .$$

Case 2. For $m/3 < k(R) \leqq \lfloor m/2 \rfloor$,

$$\#\mathcal{S} = \mathcal{S}^* - 2\left(1+2+\ldots+\left\lfloor \frac{k(R)-m/3}{2} \right\rfloor\right) \text{ if } k(R)-m/3 \text{ is even}$$

$$\#\mathcal{S} = \mathcal{S}^* - 2\left(1+2+\ldots+\left\lfloor \frac{k(R)-m/3}{2} \right\rfloor\right) - \frac{k(R)-m/3+1}{2} \text{ if } k(R)-m/3 \text{ is odd} \tag{15}$$

where $\lfloor I \rfloor$ is the least integer greater than or equal to I.

Case 3. For $\lceil m/2 \rceil < k(R) \leqq m$, we can use the fact that $\#\mathcal{S}$ is symmetric about m/2 since specifying the distribution of the

streams left to be decoded is equivalent to specifying the distribution of the streams already decoded.

We will now illustrate the use of these formulas by considering an example with m = 21 and $\underline{n}$ = (7,7,7) as the original distribution of groups.

k(R)	#$\mathcal{S}$	$\mathcal{S}$
1	1	{(0,0,1)}
2	2	{(0,0,2);(0,1,1)}
3	3	{(0,0,3);(0,1,2); (1,1,1)}
4	4	{(0,0,4);(0,1,3);(0,2,2);(1,1,2)}
5	5	{(0,0,5);(0,1,4);(0,2,3);(1,1,3);(1,2,2)}
6	7	{(0,0,6);(0,1,5);(0,2,4);(0,3,3);(1,1,4);(1,2,3);(2,2,2)}
7	8	{(0,0,7);(0,1,6);(0,2,5);(0,3,4);{(1,1,5);(1,3,3)};{(1,2,4);(2,2,3)}}
8	9	{(0,1,7);(0,2,6);(0,3,5);{(0,4,4);(1,3,4)};{(1,1,6);(2,2,4)};{(1,2,5);(2,3,3)}}
9	10	{(0,2,7);(0,3,6);{(0,4,5);(1,4,4)};{(1,1,7);(2,2,5)};{(1,2,6);(2,3,4)};{(1,3,5);(3,3,3)}}
10	10	{(0,3,7);(0,4,6);{(0,5,5);(2,2,6)};{(1,2,7);(2,3,5)};{(1,3,6);(2,4,4)};1,4,5);(3,3,4)}
11	10	{(0,4,7);(0,5,6);{(1,3,7);(2,3,6)};{(1,4,6);(2,4,5)};{(1,5,5);(3,3,5)};{(2,2,7);(3,4,4)}}
12	10	{(0,5,7);(0,6,6);{(1,4,7);(2,5,5)};{(1,5,6);(3,3,6)};{(2,3,7);(3,4,5)};{(2,4,6);(4,4,4)}}
13	9	{(0,6,7);(1,5,7);(1,6,6);{(2,4,7);(3,4,6)};{(2,5,6);(3,5,5)};{(3,3,7);(4,4,5)}}
14	8	{(0,7,7);(1,6,7);(2,5,7);(2,6,6);{(3,4,7);(4,4,6)};{(3,5,6);(4,5,5)}}
15	7	{(1,7,7);(2,6,7);(3,5,7);(3,6,6);(4,4,7);{(4,5,6);(5,5,5)}}
16	5	{(2,7,7);(3,6,7),(4,5,7);(4,6,6);(5,5,6)}
17	4	{(3,7,7);(4,6,7);(5,5,7);(5,6,6)}
18	3	{(4,7,7);(5,6,7);(6,6,6)}
19	2	{(5,7,7);(6,6,7)}
20	1	{(6,7,7)}

Clearly, if m is not a multiple of 3 or if the original group distribution is not symmetric, these formulas get more complicated.

It is also helpful to have an algorithm for generating the members of the set $\mathcal{S}$ for a given k(R). Such an algorithm follows:

(1) $n_1 = \max \left[0, k(R) - 2m/3\right]$

(2) $n_2 = \max \left[n_1, k(R) - m/3 - n_1\right]$

(3) $n_3 = k(R) - n_1 - n_2$

(4) WRITE (n_1, n_2, n_3)

(5) IF $n_3 \leqq n_2 + 1$, GO TO (9)

(6) $n_2 = n_2 + 1$

(7) $n_3 = n_3 - 1$

(8) GO TO (4)

(9) $n_1 = n_1 + 1$

(10) IF $n_1 \leqq k(R)/3$, GO TO (2)

(11) STOP

As an aside to the above discussion, let us consider an alternate way of deriving a lower bound on $R_{boot}(\gamma)$ for three group bootstrap decoding. We will proceed as follows:

(1) Compute the best Pareto exponent $\max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))$ that can be obtained using the three group bootstrap decoding

algorithm starting from all possible situations $\underline{n}$, i.e., all $n \in \mathcal{J} = \{\underline{n} = (n_1, n_2, n_3): n_1 + n_2 + n_3 = k(R),\ 1 \leq k(R) \leq m\}$. (Note that $\mathcal{J}$ is the set of all $\underline{n}$ with a fixed $k(R)$ whereas $\mathcal{J}$ is the set of all $\underline{n}$ with any $k(R)$.)

(2) Let $\mathcal{J}^* = \{\underline{n} \in \mathcal{J}: \max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(n)) > \gamma\}$ (Note that if $\underline{n}' = (n_1', n_2', n_3') \in \mathcal{J}^*$, then any $\underline{n}'' = (n_1'', n_2'', n_3'')$ which can be obtained from $\underline{n}'$, i.e., $n_1'' \leq n_1'$, $n_2'' \leq n_2'$, and $n_3'' \leq n_3'$, also belongs to $\mathcal{J}^*$. This saves us the task of computing $\max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))$ for all $\underline{n} \in \mathcal{J}$. Also note that an $\underline{n}'$ with a large $k(R)$ will in general have a smaller Pareto exponent than an $\underline{n}''$ with a smaller $k(R)$ which cannot be obtained from $\underline{n}'$ since we would expect the parity information to speed up decoding more in the latter case.)

(3) Compute the exponent $\tilde{k}(R)\sigma(\infty)$ for ordinary sequential decoding which leaves the decoder in a situation $\underline{n} \in \mathcal{J}^*$. (Note that $\tilde{k}(R)$ need not be an integer.)

(4) $R_{boot}^{L}(\gamma)$ is then defined as the supremum of rates for which

$$\min\left\{\tilde{k}(R)\ \sigma(\infty),\ \min_{\mathcal{J}^*}\left[\max(\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))\right]\right\} > \gamma \qquad (16)$$

is satisfied.

The main difficulty in computing this bound is in finding the exponent for the ordinary sequential decoding portion of the algorithm. Let $k_{max}(R)$ be the largest value of $k(R)$ for any $\underline{n} \in \mathcal{J}^*$ and let $k_{min}(R)$ be the smallest value of $k(R)$ for any $\underline{n} \in \mathcal{J}^*$ which cannot be obtained from another member of $\mathcal{J}^*$ with a larger value of $k(R)$. Then it may appear that by suitable combinatorial arguments, $\tilde{k}(R)$ could be shown to be in the range

$k_{min}(R) < \tilde{k}(R) < k_{max}(R)$. However in the limit of large ℓ, terms with smaller values of k(R) dominate terms with larger values of k(R), and hence $\tilde{k}(R) = k_{min}(R)$. Therefore the bound obtained using this method is the same as the original bound.

Finally, we will say a few words about extending the results of this bound to other parity-check schemes. In particular, consider the following (n-1) x n array ($n \geq 3$):

$$\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} .$$

We can then form a parity-check matrix $\underline{H}$ for an n-group code by repeating each column of the above array m/n times, resulting in an R = m-n+1/m block code. For example, the $\underline{H}$ matrix for the R = 25/28 4-group code is

$$\underline{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (17)$$

Columns 1-7 constitute group 1, columns 8-14 constitute group 2, etc. Note that for any given codeword, the parity of each group must be the same. Hence once one group is decoded correctly the parity of each of the other groups is known, which is a significant aid to finishing the decoding of the other groups. Also note that the row space of $\underline{H}$(the set of all parity checks) is

completely symmetric with respect to the labeling of the groups. Therefore the labeling of the groups is immaterial, as was mentioned before in specifying the members of the set .

It should be evident that the arguments used in finding an upper bound on the moments of decoding effort for the 3-group code can be extended directly to group codes of higher order. The formulas for specifying the size of the set $\mathcal{S}$ and the algorithm for generating the members of $\mathcal{S}$, however, must be restated for each particular case. This will be carried out upon successful completion of the computer calculations necessary to plot the bounds for the 3-group code.

Reference

1. F. Jelinek and J. Cocke, "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels," Information and Control, April 1971.

II-G. Group Code Results Applicable to Bootstrap Decoding

The results of this section were obtained by D. Costello while he was a research associate of the project.

1. Extending the Upper Bound on the Moments of the Decoding Effort to n-Group Codes

The characteristic feature of all n-group codes is that once the parity of any one group is decoded, the parity of all the other groups is immediately known to be the same. An n-group code contains n-1 parity checks, i.e., the $\underline{H}$ matrix has n-1 rows. The columns of $\underline{H}$ consist of the following set of n vectors of length n-1, each of which may appear more than once:

1	1	0				0	0	0
0	1	1				0	0	0
0	0	1				0	0	0
0	0	0				0	0	0
.	.	.				.	.	.
.	.	.	.	.	.	.	.	.
.	.	.				.	.	.
0	0	0				0	0	0
0	0	0				1	0	0
0	0	0				1	1	0
0	0	0				0	1	1

The number of columns in which each of these vectors appears determines the size of each of the n groups. For convenience we will assume that all groups are of the same size.

Example

$$\underline{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

is the parity check matrix for an R = 9/12 4-group code.

Note that the first row of the $\underline{H}$ matrix forces the parity of the first group to be the same as the parity of the second group, the second row of $\underline{H}$ forces the parity of the second group to be the same as the parity of the third group, and so on. Thus we get the property of group codes mentioned previously. Also note that all n-group codes are very high rate codes with minimum distance 2, i.e., they only detect single errors in an algebraic sense. However, this does not militate against their use as algebraic codes in the bootstrap hybrid decoding scheme. In fact, their simple structure makes them especially attractive for calculating the error exponent function. (NOTE: The word "group" here should not be confused with the usual notion of a group (linear) code.)

When using group codes, once we have decoded a single group, the parity of the other groups is known and they can be decoded independently as in the single parity check case. Hence if we desire high rates, it is also advantagneous to keep the group sizes as small as possible.

EXAMPLE

Assume that we wish to use an algebraic code of rate about 9/10. With a single parity check the group size is 10.

With two parity checks, a three group code with R = 19/21 has group size 7. However, we cannot continue to decrease the group size by increasing the number of groups. With three parity checks, a four group code with R = 29/32 has group size 8. In general, we require that R = gn-(n-1)/gn = 9/10, where g is the group size and n is the number of groups. This imples that gn = (n-1)10 or $\lim_{n\to\infty} g = 10$, the same group size required by a single parity check. Clearly, for a given rate R, there is an optimum group number n which yields the smallest possible group size g.

The derivation of the upper bound on the moments of the decoding effort given for the three group code can be extended to higher order group codes. The only difference is that a new algorithm is needed to generate the set S of possible situations in each case and the formula for the error exponent function $E_k(\sigma)$ must be generalized.

2. A Lower Bound on the Moments of the Decoding Effort for Group Codes

Proceeding analogously to the derivation of the lower bound on the moments of the decoding effort for the single parity check case, we can derive a similar lower bound for all group codes. In particular, for the three group code, $R^{U}_{boot}(\gamma)$ is the infimum (greatest lower bound) of rates for which $\min \{\min_{S_3} [\max (\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))], k \min_{S_k} [\max (\sigma_1(\underline{n}), \sigma_2(\underline{n}), \sigma_3(\underline{n}))]\} < \gamma$, where

$$S_k = \{\underline{n}: n_1+n_2+n_3 = k\} \ , \quad S_3 = \{\underline{n}: n_1+n_2+n_3 = 3\} \ ,$$

$\sigma_i(\underline{n})$ is the solution of $R = E_k^i(\sigma)/\sigma$, and $E_k^i(\sigma)$ is the error exponent function for a given situation when decoding begins in group i. In order to compute $R_{boot}^{U}(\gamma)$, we must compute the differences

$$\frac{k}{\gamma} \min_{S_k} \left[\max \left(E_k^1(\tfrac{\gamma}{k}), E_k^2(\tfrac{\gamma}{k}), E_k^3(\tfrac{\gamma}{k})\right)\right] -$$

$$\frac{k+1}{\gamma} \min_{S_{k+1}} \left[\max \left(E_{k+1}^1(\tfrac{\gamma}{k+1}), E_{k+1}^2(\tfrac{\gamma}{k+1}), E_{k+1}^3(\tfrac{\gamma}{k+1})\right)\right]$$

for k = 4,5,..., until their value becomes negative. If this takes place at $k = k^+$, then

$$R_{boot}^{U}(\gamma) = \min \left[\frac{1}{\gamma}E_3(\gamma), \frac{k^+}{\gamma} E_{k^+}(\tfrac{\gamma}{k^+})\right] ,$$

where $E_{k^+}(\gamma/ k^+)$ is

$$\min_{S_{k^+}} \left[\max \left(E_{k^+}^1(\tfrac{\gamma}{k^+}), E_{k^+}^2(\tfrac{\gamma}{k^+}), E_{k^+}^3(\tfrac{\gamma}{k^+})\right)\right]$$

And $E_3(\gamma)$ is

$$\min_{S_3} \left[\max \left(E_3^1(\gamma), E_3^2(\gamma), E_3^3(\gamma)\right)\right] .$$

Again the extension of the lower bound to all group codes depends only on the generalization of the function $E_k(\sigma)$ and on a new algorithm to generate the set S.

3. Proof That Knowledge of the Syndrome is Equivalent to Knowledge of the Group Parity

We wish to show that the ratio that enters into the calculation of the likelihood function, viz. $P(\underset{\sim}{y},x_i)/P(\underset{\sim}{y})$, is equivalent to the ratio $P(\underset{\sim}{s},y_i/x_i)/P(\underset{\sim}{s},y_i)$, where $\underset{\sim}{s}$ is the syndrome sequence and the ith digit is being decoded.

First we compute

$$P(\underset{\sim}{y}) = \sum_{C} P(\underset{\sim}{y},\underset{\sim}{x}) = \sum_{C} P(\underset{\sim}{y}/\underset{\sim}{x})\ P(\underset{\sim}{x}) = 2^{-k} \sum_{C} \prod_{j=1}^{n} P(y_j/x_j)$$

where the rate of the algebraic code being used is k/n and C is the set of all codewords. Similarly,

$$P(\underset{\sim}{y}/x_i) = P(y_i/x_i)P(y_1,\ldots,y_{i+1};\ldots,y_n/x_i)$$

$$= P(y_i/x_i) \sum_{C_i} 2^{-(k-1)} \prod_{\substack{j=1 \\ j\neq i}}^{n} P(y_j/x_j)$$

where C_i is the set of all codewords whose ith component is x_i (half of the codewords in C for a linear code). Hence we obtain the ratio

$$\frac{P(\underset{\sim}{y}/x_i)}{P(\underset{\sim}{y})} = 2P(y_i/x_i)\ \frac{\sum_{C_i} \prod_{\substack{j=1 \\ j\neq i}}^{n} P(y_j/x_j)}{\sum_{C} \prod_{j=1}^{n} P(y_j/x_j)} \quad .$$

Now the ratio $P(\underset{\sim}{s},y_i/x_i)/P(\underset{\sim}{s},y_i)$ must be determined. Beginning with the denominator, we find that $P(\underset{\sim}{s},y_i) = \sum_{Y_i(\underset{\sim}{s})} P(\underset{\sim}{y})$ where $Y_i(\underset{\sim}{s})$ is the set of all possible received sequences $\underset{\sim}{y}$ which have syndome $\underset{\sim}{s}$ and whose ith component is y_i. Since there are 2^k equally likely received sequences corresponding to each syndrome and half of these have

an ith component equal to y_i,

$$P(\underset{\sim}{s},y_i) = 2^{(k-1)}\, P(\underset{\sim}{y}:\ \underset{\sim}{y}\,\underset{\sim}{H} = \underset{\sim}{s}) ,$$

where $\underset{\sim}{H}$ is the parity check matrix. (Note that in general $P(\underset{\sim}{y})$ depends upon $\underset{\sim}{y}$, but that those particular received sequences which result in a given syndrome are all equally likely. For example, the set of all codewords result in the syndrome $\underset{\sim}{s} = \underset{\sim}{0}$, and they are clearly equally likely.) But

$$P(\underset{\sim}{y}:\ \underset{\sim}{y}\,\underset{\sim}{H} = \underset{\sim}{s}) = 2^{-k} \sum_{C} \prod_{j=1}^{n} P(y_j/x_j),$$

where the evaluation is the same for all $\underset{\sim}{y}$ that result in a given syndrome. Hence,

$$P(\underset{\sim}{s},y_i) = \frac{1}{2} \sum_{C} \prod_{j=1}^{n} P(y_j/x_j),$$

where the products are taken for any $\underset{\sim}{y}$ such that $\underset{\sim}{y}\underset{\sim}{H} = \underset{\sim}{s}$. Since

$$P(\underset{\sim}{s}/y_i) = \frac{P(\underset{\sim}{s},y_i)}{P(y_i)} = \sum_{C} \prod_{j=1}^{n} P(y_j/x_j),$$

$$P(\underset{\sim}{s},y_i/x_i) = P(y_i/x_i)\, P(\underset{\sim}{s}/y_i,x_i)$$

$$= P(y_i/x_i) \sum_{C_i} \prod_{\substack{j=1 \\ j\neq i}}^{n} P(y_j/x_j),$$

where the products are taken for any $\underset{\sim}{y}$ such that $\underset{\sim}{y}\underset{\sim}{H} = \underset{\sim}{s}$. Hence we obtain the ratio

$$\frac{P(\underset{\sim}{s},y_i/x_i)}{P(\underset{\sim}{s},y_i)} = 2P(y_i/x_i)\,\frac{\sum_{C_i}\ \prod_{j=1,\,j\neq i}^{n} P(y_j/x_j)}{\sum_{C}\ \prod_{j=1}^{n} P(y_j/x_j)} = \frac{P(\underset{\sim}{y}/x_i)}{P(\underset{\sim}{y})} \quad .$$

In the case of three group bootstrap decoding, this result states that knowledge of the two syndrome digits is equivalent to knowledge of the three group parity digits. However, the simplest way to calculate $P(\underset{\sim}{y},x_i)/P(\underset{\sim}{y})$ is to use the formulas based on the three group parity digits, since these formulas take advantage of the independence among the three groups.

C. 9.

II-H. Optimal Decoding of Convolutional Codes for Finite State Channels and its Application to Bootstrap Decoding

1. Introduction

In this section we describe a method of decoding of convolutional codes which minimizes the per bit probability of error (Viterbi decoding minimizes the probability of erroneous codeword decoding). This method applies to all linear codes (see Sections 5 and 6) and can be used in conjunction with arbitrary discrete finite state channels. The complexity of the method grows as $K2^{\upsilon}$ where υ is either the constraint length of the convolutional code or the syndrome size of the linear code. This work was done jointly with L. Bahl, J. Cocke, and J. Raviv of IBM.

While it is doubtful that one would actually wish to build decoders operating according to these methods, they can be effectively used to allow computation of optimal likelihood functions for the sequential decoding phase of a bootstrap scheme whose algebraic component is based on an arbitrary convolutional or linear code (see Section 7). Moreover, we believe that our method will make possible the application of bootstrapping methods to finite state channels such as the Gilbert burst noise channel.

The per-bit probability of error will be minimized by finding the probabilities that the encoder was in a particular state at any time i. As a consequence, a posteriori probabilities that a particular digit was sent through the channel at some given time i will also be obtainable.

Our method will apply to finite state channels whose transmission probabilities are

$$Q^*(y_i, v_i \mid v_{i-1}, x_i) \tag{1}$$

where $y_i \in \mathcal{Y}$ $x_i \in \mathcal{X}$ are the i^{th} received and transmitted digits ($\mathcal{X}$ and $\mathcal{Y}$ are finite alphabets), and v_i, $v_{i-1} \in \mathcal{V}$ are the i^{th} and $(i-1)^{\text{th}}$ channel states and is a finite state alphabet). The channel operates by the rule:

$$P\{y_1, \ldots, y_n, v_1, \ldots, v_n \mid v_o, x_1, \ldots, x_n\} = \prod_{i=1}^{n} Q^*(y_i, v_i \mid v_{i-1}, x_i). \tag{2}$$

Obviously, discrete memoryless channels are special cases of finite state channels, as is, for instance, the well-known Gilbert Channel which has a "good" and a "bad" state with transitions that are independent of channel inputs.

Since the natural transmission units of convolutional codes

are branches (i.e. blocks of n digits), it will be convenient to define special notation for these. We will let capitals refer to branches, i.e.,

$$X_t = x_{tn+1}, x_{tn+2}, \cdots, x_{(t+1)n}$$

$$Y_t = y_{tn+1}, y_{tn+2}, \cdots, y_{(t+1)n} \tag{3}$$

Also, we will define a new branch transmission probability

$$Q(Y_t, v_t \mid v_{t-1}, X_t) = \sum_{\mathcal{S}} Q^*\left(y_{(t+1)n}, v_t \mid v_{n-1}, x_{(t+1)n}\right) Q^*\left(y_{tn+1}, v_1 \mid v_{t-1}, x_{tn+1}\right) \cdot \prod_{i=2}^{n-1} Q^*\left(y_{tn+i}, v_i \mid v_{i-1}, x_{tn+i}\right) \tag{4}$$

where $\mathcal{S}$ is the set of all vectors $(v_1, v_2, \ldots, v_{n-1})$. As a result,

$$P\left\{Y_1, \ldots, Y_k, v_1, \ldots, v_k \mid v_o, X_1, \ldots, X_k\right\} = \prod_{i=1}^{k} Q\left(Y_i, v_i \mid v_{i-1}, X_i\right) \tag{5}$$

2. Optimal Determination of Message Digits.

Let the information blocks determining the coder output branches be $I_1, I_2, \ldots$ (e.g. for a binary convolutional code of rate $R = k/n$, I_i corresponds to a block of k bits), and let the $i^{\underline{th}}$ state of the encoder, S_i, be given by the vector

$$S_i = (I_i, I_{i-1}, \ldots, I_{i-\upsilon-2}) \tag{6}$$

where υ is the constraint length of the code. Suppose a codeword is determined by T true information digits, and thus consists of $T+\upsilon-1$ branches (the usual termination by $\upsilon-1$ dummy information $\underline{0}$-blocks is assumed). The encoder state sequence of interest the is

$$S_o = \underline{0},\ S_1, \ldots,\ S_T, \ldots,\ S_{T+\upsilon-1} = \underline{0} \tag{7}$$

If f is the code output function, then

$$X_t = f(I_t, S_{t-1}) \tag{8}$$

Let

$$\varphi_i = \begin{cases} 1 & \text{if the decoder determines the } i^{\text{th}} \text{ message bit incorrectly} \\ 0 & \text{otherwise} \end{cases}$$

Then the per-input block probability of error is

$$P_e = \frac{1}{T}\ E\left[\sum_{i=1}^{Tn} \varphi_i\right] = \frac{1}{T} \sum_{i=1}^{Tn} E\,\varphi_i \tag{10}$$

and so we wish to minimize $E\varphi_i$ for all i. But for $1 \le j \le t$,

$$E\left[\varphi_{tn+j}\right] = \sum_{\ell \notin a_{t,j}} P\left\{S_{t+1} = \ell \mid Y_1, \ldots, Y_{T+\upsilon-1}\right\} \tag{11}$$

where $a_{t,j}$ denotes the set of states S_{t+1} with first block I_{t+1} (see (6)) whose j^{th} digit agrees with the one actually sent. It follows that to minimize P_e we ought to minimize the sums on the righthand side of (11) over all the possible sets $a_{t,j}$. To be able to do so, we will find the probability terms of the sum of (11).

3. Determination of A Posteriori Encoder State Probabilities

Let us define super-states

$$U_i = (S_i, v_i) \tag{12}$$

and the probability functions

$$\alpha_t(i,\ell) = P\{U_t = (i,\ell),\ Y_1,\ldots,Y_t\} \tag{13}$$

$$\beta_t(i,\ell) = P\{Y_{t+1},\ldots,Y_{T+\upsilon-1} \mid U_t = (i,\ell)\} \tag{14}$$

$$\lambda_t(i,\ell) = P\{U_t = (i,\ell),\ Y_1,\ldots,Y_{T+\upsilon-1}\} \tag{15}$$

Now for $t\in[1,T+\upsilon-2]$

$$\lambda_t(i,\ell) = P\{U_t = (i,\ell),\ Y_1,\ldots,Y_t\} \cdot$$

$$\cdot P\{Y_{t+1},\ldots,Y_{T+\upsilon-1} \mid U_t = (i,\ell),\ Y_1,\ldots,Y_t\} =$$

$$= \alpha_t(i,\ell)\ \beta_t(i,\ell) \tag{16}$$

and

$$\lambda_{T+\upsilon-1}(i,\ell) = \alpha_{T+\upsilon-1}(i,\ell) \tag{17}$$

(If U_t is known, events after time t do not depend on $Y_1,\ldots,Y_t$).

We will show below that it is easy to compute α_t and β_t recursively.

In any case, it follows from (15) and (17) that

$$P\{S_t = i \mid Y_1,\ldots,Y_{T+\upsilon-1}\} = \frac{\sum_{\ell} \lambda_t(i,\ell)}{\sum_{i,\ell} \alpha_{T+\upsilon-1}(i,\ell)} \tag{18}$$

and so our task is to find $\lambda_t(i,\ell)$.

Let the initial distribution of the channel state v_o be given by

$$P\{v_o = q\} = w(q) \tag{19}$$

Then

$$\alpha_1(i,\ell) = \sum_q P\{U_1 = (i,\ell),\ Y_1 \mid U_o = (\underline{0},\ q)\}w(q) \tag{20}$$

and for $t = 2, 3, \ldots, T + \upsilon - 1$,

$$\alpha_t(i,\ell,q) = \sum_{j,m,q} P\{U_{t-1} = (j,m),\ U_t = (i,\ell),\ Y_1, \ldots, Y_t\}$$

$$= \sum_{j,m,} P\{U_t = (i,\ell),\ Y_t \mid U_{t-1} = (j,m)\}\ P\{U_{t-1} = (j,m),\ Y_1, \ldots, Y_{t-1}\}$$

$$= \sum_{j,m} P\{U_t = (i,\ell),\ Y_t \mid U_{t-1} = (j,m)\}\ \alpha_{t-1}(j,m) \tag{21}$$

where the middle equality follows from the fact that all events after time $t-1$ are independent of $Y_1, \ldots, Y_{t-1}$ once the superstate U_{t-1} is known. Similarly,

$$\beta_{T+\upsilon-2}(i,\ell) = \sum_m P\{U_{T+\upsilon-1} = (\underline{0},m),\ Y_{T+\upsilon-2} \mid U_{T+\upsilon-2} = (i,\ell)\} \tag{22}$$

and for $t = 1, 2, \ldots, T + \upsilon - 3$,

$$\beta_t(i,\ell) = \sum_{j,m} P\{U_{t+1} = (j,m),\ Y_{t+1}, \ldots,\ Y_{T+\upsilon-1} \mid U_t = (i,\ell)\} =$$

$$= \sum_{j,m} P\{Y_{t+2}, \ldots, Y_{T+\upsilon-1} \mid U_{t+1} = (j,m)\}\ P\{U_{t+1} = (j,m),\ Y_{t+1} \mid U_t = (i,\ell)\}$$

$$= \sum_{j,m} \beta_{t+1}(j,m)\ \ P\{U_{t+1} = (j,m),\ Y_{t+1} \mid U_t = (i,\ell)\} \tag{23}$$

Relations (21) and (22) bear out our earlier contention that α_t and β_t are recursively obtainable. It remains to specify the probabilities $P\{U_{t+1}=(j,m),\ Y_{t+1} \mid U_t=(i,\ell)\}$ that appear on the righthand sides of (20) through (23).

Let

$$\mu(i,j) = \begin{cases} 1 & \text{if a one step transition from state } i \text{ to state } j \text{ is possible} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

and let $g(j)$ be the initial information block of the state j. Then

$$P\{U_{t+1}=(j,m),\ Y_{t+1} \mid U_t=(i,\ell)\} =$$

$$= \mu(i,j)\ Q(Y_{t+1},\ m \mid \ell,\ f(g(j),\ i)\ P\{I_{t+1} = g(j)\} \tag{25}$$

In the usual situation in which all sequences are equally likely, $P\{I_{t+1} = g(j)\} = 2^{-k}$. It will, however, be useful later on to have the general expression (25).

We conclude this section by outlining the algorithm that will minimize the probability of bit error:

1) While the sequence $Y_1, \ldots, Y_{T+\upsilon-1}$ is being received, the decoder computes recursively the probabilities $\alpha_t(i,\ell)$ [see (13)], using the relations (21) and (25). The obtained values are stored for all $t = 1, \ldots, T+\upsilon-1$ and i, ℓ. The amount of work involved is roughly that for forward Viterbi decoding.

2) The decoder then starts computing recursively the probabilities $\beta_{T+\upsilon-2}(i,\ell), \beta_{T+\upsilon-3}(i,\ell), \ldots, \beta_1(i,\ell)$, using relations (23) and (25). When $\beta_{T+\upsilon-2}(i,\ell)$ are computed, they and the stored $\alpha_{T+\upsilon-2}(i,\ell)$ are used

to obtain $\lambda_{T+\upsilon-2}(i,\ell)$ [see (16)]. The latter then replace $\alpha_{T+\upsilon-2}(i,\ell)$ in storage. This is done in general, $\lambda_t(i,\ell)$ replacing $\alpha_t(i,\ell)$ for $t = T + \upsilon-3, T + \upsilon -4, \ldots, 1$. The work involved in this stage of the algorithm is roughly equivalent to that of backward Viterbi decoding.

3) Finally, the stored $\lambda_t(i,\ell)$ are used to calculate $P\{S_t = i \mid Y_1, \ldots, Y_{T+\upsilon-1}\}$ [see (18)] and the quantities

$$\gamma_{t,i}(z) = \sum_{\ell \in \mathcal{A}_i(z)} P\{S_t = \ell \mid Y_1, \ldots Y_{T+\upsilon-1}\} \qquad (26)$$

where $\mathcal{A}_i(z)$ is the set of states whose initial block I_t has its i^{th} digit $(i = 1, \ldots, k)$ equal to z. If

$$\gamma_{t,i}(z^*) = \max_z \gamma_{t,i}(z) \qquad (27)$$

the decoder decides that the $[(t-1)\, n + i]^{\text{th}}$ information digit was z^*.

Unfortunately, this algorithm requires quite a large storage. Its size grows linearly with block length T. It is not clear with what accuracy it is necessary to store the values $\alpha_t(i,\ell)$ and $\lambda_t(i,\ell)$.

In conclusion, let us observe that the computation of the probabilities $\alpha_t(i,\ell)$ [see (20) and (21)] was based on the initial channel state distribution w(q). At the beginning of the communication process, w( ) would normally be the stationary distribution of the states. However, it follows from (13) that

$$P\{v_{T+\upsilon-1} = \ell \mid Y_1, \ldots, Y_{T+\upsilon-1}\} = \frac{\alpha_{T+\upsilon-1}(0,\ell)}{\sum_{\ell} \alpha_{T+\upsilon-1}(0,\ell)} \qquad (28)$$

and thus the w-function for the decoding of the second block would naturally be given by the relation

$$w(q) = P\left\{v_{T+\upsilon-1} = \ell \,\middle|\, Y_1, \ldots, Y_{T+\upsilon-1}\right\} \tag{29}$$

where the conditioning random variables are those received in the first block. The definition of the w( ) -function for the third and following blocks is similar. The important point is that no information about the starting state of any block gained through the decoding of previous blocks is ever lost.

4. Probabilities of Transmitted Digits.

Sometimes it is of interest to determine the probabilities $P\{X_t \mid Y_1,\ldots,Y_{T+\upsilon-1}\}$ that the $t^{\underline{th}}$ transmitted branch was X_t, given that the branches $Y_1,\ldots,Y_{T+\upsilon-1}$ were received (an application is given in the next section). We now proceed to do so.

X_t is fully determined by S_{t-1} and S_t (see (81)), so that

$$X_t = F(S_{t-1}, S_t) \tag{30}$$

Let $\mathcal{F}(X_t)$ be the set of all pairs S_{t-1}, S_t for which (30) holds. Then

$$P\{X_t \mid Y_1,\ldots,Y_{T+\upsilon-1}\} = \sum_{(i,j)\,\in\,\mathcal{F}(X_t)} P\{S_{t-1} = i,\ S_t = j \mid Y_1,\ldots,Y_{T+\upsilon-1}\} =$$

$$= \sum_{(i,j)\in\mathcal{F}(X_t)} \sum_{\ell,m} P\{U_{t-1} = (i,\ell),\ U_t = (j,m) \mid Y_1,\ldots,Y_{T+\upsilon-1}\} \tag{31}$$

Therefore, it is desirable to determine the probability terms on the righthand side of (31). But

$$P\{U_{t-1} = (i,\ell),\ U_t = (j.m),\ Y_1,\ldots,Y_{T+\upsilon-1}\} =$$

$$= P\{Y_{t+1},\ldots,Y_{T+\upsilon-1} \mid U_t = (j,m)\}\ P\{U_t = (j,m), Y_t \mid U_{t-1} = (i,\ell)\}\cdot$$

$$\cdot\ P\{U_{t-1} = (i,\ell),\ Y_1,\ldots,\ Y_{t-1}\} =$$

$$= \beta_t(j,m)\ P\{U_t = (j,m),\ Y_t \mid U_{t-1} = (i,\ell)\}\alpha_{t-1}(i,\ell) \tag{32}$$

and from (15) and (17),

$$P\{Y_1,\ldots,\ Y_{T+\upsilon-1}\} = \sum_{i,\ell} \alpha_{T+\upsilon-1}(i,\ell) \tag{33}$$

Combining (31) through (33) we thus get the formula

$$P\{X_t \mid Y_1, \ldots, Y_{T+U-1}\} = \left[\sum_{i,\ell} \alpha_{T+U-1}(i,\ell) \right]^{-1} \cdot$$

$$\cdot \sum_{(i,j)\in \mathcal{F}(X_t)} \sum_{(\ell,m)} \beta_t(j,m)\, P\{U_t = (j,m),\ Y_t \mid U_{t-1} = (i,\ell)\} \alpha_{t-1}(i,\ell) \tag{34}$$

5. Generalization to All Linear Codes

The preceding results depend on the existence of the super-states U_t whose knowledge allows the separation of past (events before time t) from the future (events after t). As seen from (12), U_t presupposes the existence of S_t, the encoder state. Our results would thus be generalizeable to all codes for which a state could be defined, and therefore a coding trellis drawn.

Let H be the parity check matrix of a given linear (n,n-r) code, and let $\underline{h}_i$, $i = 1,\ldots,n$ be the column vectors of H. Let $\underline{c}$ be a codeword. We will then define the states S_t, $t = 0,1,\ldots,n$ pertaining to $\underline{c}$ as follows:

$$\underline{S}_o = \underline{0}$$

$$\underline{S}_t = \underline{S}_{t-1} + c_t \underline{h}_t = \sum_{i=1}^{t} c_i \underline{h}_i \qquad t = 1,\ldots,n \qquad (35)$$

Obviously, $\underline{S}_n = \underline{0}$ and the current state $\underline{S}_t$ is a function of the preceding state $\underline{S}_{t-1}$ and the current input digit c_t(the relationship is time varying!). Relation (35) can thus be used to draw a trellis with at most 2^r states $\underline{S}_t$ per level. The appearance of the trellis will be similar to that for convolutional codes provided the vector set $\{\underline{h}_n,\underline{h}_{n-1},\ldots,\underline{h}_{n-r+1}\}$ has rank r (which can always be arranged). For binary codes, there will exist two transitions out of every state S_t, t= 0,1,...,k-1, and one transition out of every state S_t, t = k, k+1,...,n-1. If it turns out that $\{h_1,\ldots,h_\ell\}$ are linearly independent, then there will be one transition leading into every state S_t, $t = 1, 2,\ldots,\ell$. An example of the trellis for the Hamming code

$$H = \begin{bmatrix} 1\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,1\,1\,1 \end{bmatrix}$$

is given in Figure 1. Unfortunately, the irregularity of the trellis is typical for the general case of block codes. Obviously, every transition corresponds to a single channel input digit only. Horizontal transitions (those to an identically indexed state) correspond to 0's, the remaining transitions to 1's.

Viterbi decoding, as well as the methods of the preceding sections are clearly applicable to the trellises of linear block codes (it is even conceivable that sequential decoding can also be used). Since high-rate codes have relatively fewer states, the methods might even prove attractive in practice.

6. A "Time-Invariant" Trellis Diagram for Cyclic Codes

The trellis diagram of Figure 1 is time dependent. This unfortunate feature can be eliminated when the code is cyclic by defining the state in terms of the shift register realization of the encoder rather than in terms of the parity check matrix. This leads to a piecewise time-invariant trellis diagram, as illustrated by the following example.

EXAMPLE

Consider the 3-stage shift register encoder shown in Figure 2 for the (7,4) Hamming code. The switches are in positions A for four time units and then switch to positions B for three time units. Taking the state to be the outputs of the three register stages, we can draw the trellis diagrams as in Figure 3.

In part A of the diagram, for the states (000,110,010, 100) up branches correspond to input 0's and down branches to input 1's, whereas for the states (011,101,001,111) up branches correspond to input 1's and down branches to input 0's. In part B all branches correspond to input 0's. Note that part A and part B of the diagram, when considered separately, are both time-invariant, i.e., each state has exactly the same successors independent of time. This trellis diagram can be reduced to a state diagram whose transitions are labeled either A/B (where A is the input when the transition occurs in part B) or just A (where A is the input when the transition occurs in part A and the

transition does not occur in part B). For the (7,4) Hamming code under consideration, the state diagram is in Figure 4. When all the information digits have been read into the encoder (at the end of part A), the path back to the all-zero state can be determined directly from the state diagram for part A by merely following the path indicated by the digits of the present state read in reverse order. For example, if we are in state 100 at the end of part A, then following the path indicated by the digits 001 returns us to the state 000.

This form of the encoder results in relatively simple state diagrams for high rate codes and relatively complex state diagrams for low rate codes (since the number of states is 2^r, where r is the number of redundant digits in the code).

7. Application to Bootstrap Decoding

In this section we will state a particular application of the decoding methods of this paper to bootstrap decoding, but others are equally possible. Our example will be restricted to symmetrical, binary input channels. Consider two convolutional codes C_1 and C_2. Use C_1 to encode T_2 blocks of $K_1 = T_1 k_1$ information digits into T_2 blocks of $N_1 = (T_1 + \upsilon_1 - 1)\, n_1$ channel digits (the rate of C_i is $R_i = k_i/n_i$ and its constraint length is υ_i, $i = 1,2$), and lay the resulting code words next to each other (as indicated in Figure 5), obtaining a binary array of N_1 rows and T_2 columns. Next, take each row in the array of Figure 5 and use C_2 to encode it into a codeword of $N_2 = (T_2 + \upsilon_2 - 1)\, n_2$ channel digits, and lay the resulting codewords below each other, as indicated in Figure 6. The obtained binary array has N_1 rows and N_2 columns. Because of linearity, every column in this array is a codeword of the code C_1.

If the digits of Figure 6 are transmitted, the received digits can be used to form another $N_1 \times N_2$ array whose appearance is that of Figure 3. It is then possible to decode the array either row-wise (using code C_2) or column-wise (using code C_1) on both, and to do so, any convenient decoder may be used. If both constraint lengths υ_1 and υ_2 are relatively short, the methods of this paper may be used in both directions (see below), if υ_1 is short and υ_2 long, horizontal decoding may be carried out with the help of a sequential decoder.

In either case, the following interactive approach is suggested. The array of Figure 6 is transmitted by columns, i.e., first the digits of the first column in sequence, then those of the second column, etc.

We will assume that the state process is irreducible, and that N_1 is large enough relative to the memory of the state process so that the channel is virtually memoryless along the horizontal direction of the array of Figure 6 (in case this assumption is not satisfied, it is in principle easy to modify the following approach appropriately). The receiver works on the columns first, using the relations (29) to determine initial state distributions. The aim is to obtain the distributions (see Section 4)

$$P\{X_t \mid Y_1, \ldots, Y_{T+\upsilon_1-1}\} \qquad t = 1,2,\ldots, T_1+\upsilon_1-1 \tag{36}$$

and

$$P\{v_{T_1+\upsilon_1-1} \mid Y_1, \ldots, Y_{T_1+\upsilon_1-1}\} \tag{37}$$

the latter in order to decode the next column. The probabilities (36) may be used to find the probabilities of transmission of individual digits in the various rows of the columns,

$$P\{x_{(t-1)n+j} \mid Y_1, \ldots, Y_{T_1+\upsilon_1-1}\} = \sum P\{X_t \mid Y_1, \ldots, Y_{T_1+\upsilon_1-1}\} \tag{38}$$

where the sum is over all X_t whose $j^{\underline{th}}$ digit is $x_{(t-1)n+j}$.

When the work on the columns is completed, row decoding starts. The decoding of the r^{th} row will utilize the probabilities $P\{x_r \mid Y_1, \ldots, Y_{T_1+\upsilon_1-1}\}$ obtained for each of the N_2 columns. First, consider the case where row decoding utilizes the methods of this paper. Let $q_1(\ ), \ldots, q_{n_2}(\ )$ be the distributions (38) applicable to the n_2 digits on the branch at depth (t+1) of the r^{th} row. Because of our virtual independence assumptions, superstates U_t can be replaced by encoder states S_t, so that the probabilities $\lambda_t(i)$ [the second variable is eliminated] will be based

on the transition probabilities (compare with (25))

$$P\{S_{t+1} = j,\ Y_{t+1} \mid S_t = i\} =$$

$$= \mu(i,j)\, w\left(Y_{t+1} \mid f(g(j), i)\right)\ P\{I_{t+1} = g(j) / S_t = i\} \quad (39)$$

where $w(\ /\ )$ is the transmission probability of the virtually memoryless row channel. The probability $P\{I_{t+1} = g(j) \ / \ S_t = i\}$ is obtained with the help of the probabilities $q_1(\ ),\ldots,q_{n_2}(\ )$ determined by column decoding. In fact, let the branch digits corresponding to the transition $g(j)$ out of state j be $x_1^*,\ldots,x_{n_2}^*$. Then

$$P\{I_{t+1} = g(j) \ / \ S_t = i\} = \frac{\prod_{\ell=1}^{n_2} q_\ell(x_\ell^*)}{\sum \prod_{\ell=1}^{n_2} q_\ell(x_\ell^*)} \quad (40)$$

where the sum in the denominator is over the sequences $x_1,\ldots,x_{n_2}$ associated with the 2^k branches leaving state i.

The aim of row decoding is to obtain probabilities $P\{x_r \mid Y_1,\ldots, Y_{T_2+\upsilon_2-1}\}$, $r = 1,\ldots,N_2$ to be used next in column decoding based again on the transition probabilities $P\{U_{t+1} = (j,m),\ Y_{t+1} \mid U_t = (i,\ell)\}$ [see (25)] where formula (40) enables utilization of information gained in row decoding. The process may be iterated any number of times. The last iteration performs the final decoding according to the three-step algorithm described in Section 3.

Let us next consider the case where the row constraint length υ_2 is large so that sequential decoding must be used. When the first column decoding cycle is completed, the row decoder is in possession

of probabilities $P\{x_r \mid Y_1,\ldots,Y_{T_1+\upsilon_1-1}\}$ obtained by formula (38). Since row-memory is assumed to be practically non-existent, the usual sequential algorithm is carried out. The difference is that the likelihood functions used on the $i^{\underline{th}}$ branch digit are given by the formula

$$\log \frac{w(y_i \mid x_i)}{w_i(y_i)} - R \tag{41}$$

where

$$w_i(y_i) = \sum_x w(y_i \mid x)\ q_i(x) \tag{42}$$

It is, of course, through formula (42) that the sequential decoder utilizes information gained in column decoding. Sequential decoding on a given row continues until that row is decoded, or until the likelihood drops by so much that further advance is "hopeless" (this is similar to the original Bootstrap Decoding Algorithm). If the decoder advanced to depth J, it is assumed that all digits from depth 1 through J-t [for some judiciously chosen t] have been definitely decoded. This means that for the purpose of future column decoding, the probabilities $P\{I_{t+1} = g(j) \mid S_t = i\}$ are changed, some becoming zero [we assume that the sequential decoding involved row $tk_1 + r$, $r\epsilon\{1,2,\ldots,k_1\}$]. After row decoding has been completed, column decoding whose aim is to obtain new probabilities (38) is performed on those columns where change in some probabilities $P\{I_{t+1} = g(j) \mid S_t = i\}$ took place. This process is iterated until all rows have been completely sequentially decoded.

Obviously, the above two applications to bootstrapping are very tentative. The precise algorithms must be determined by experimentation.

In conclusion we wish to point out, that the column code need not be a convolutional one. As shown in Section 5, any linear code is amenable to the methods of Sections 3 and 4, provided its rate is high enough so that the number of trellis states is not excessive.

Figure Captions:

Fig. 1: Trellis diagram for the (7, 4) Hamming code.

Fig. 2: Shift register encoder for the (7, 4) Hamming code.

Fig. 3: Time-invariant trellis diagram for the (7, 4) Hamming code.

Fig. 4: State diagram for the (7, 4) Hamming code.

Fig. 5: Initial convolutional encoding of T_2 information digit sequences.

Fig. 6: The final code block resulting from convolutional encoding of N_1 sequences of binary code digits.

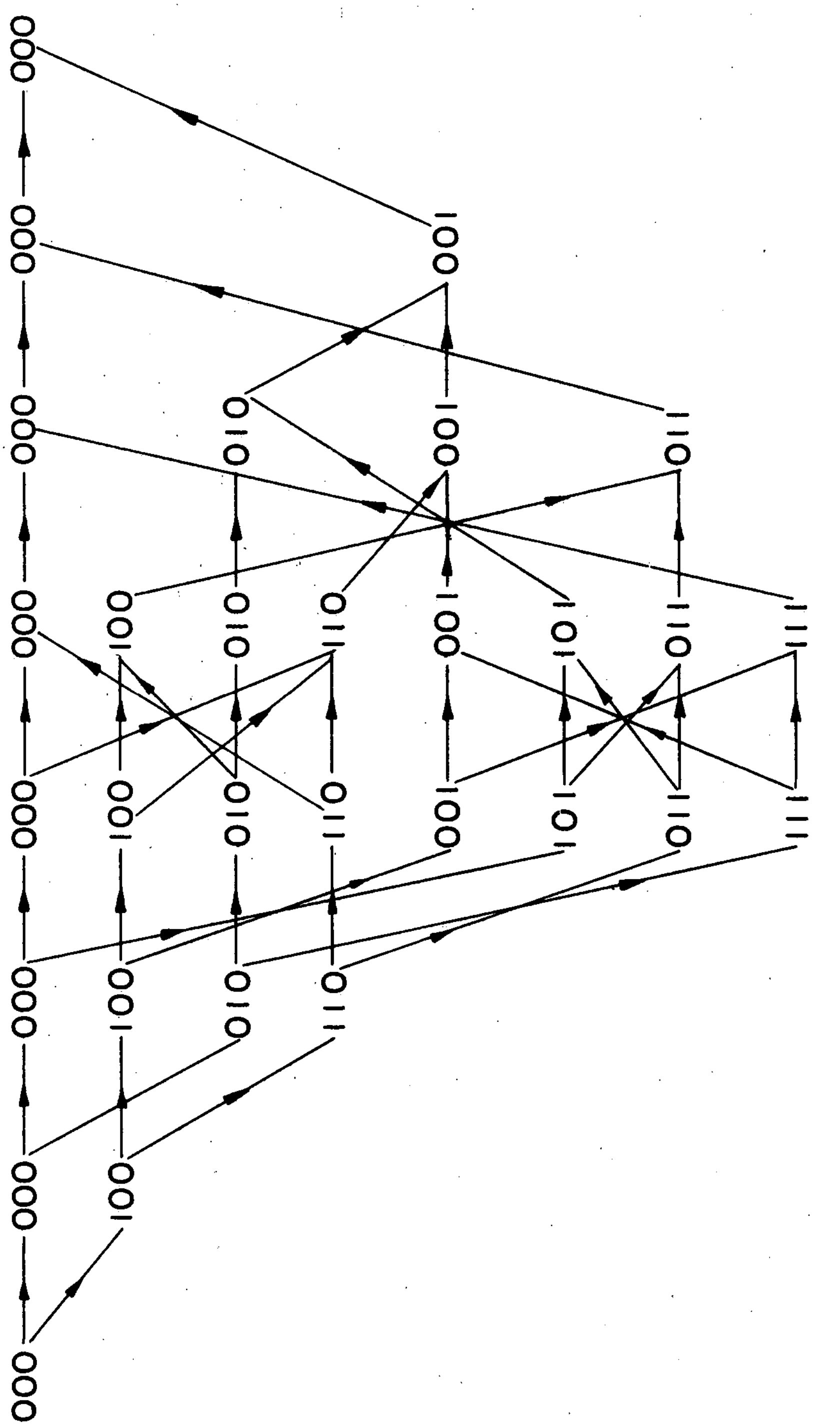


FIG I.

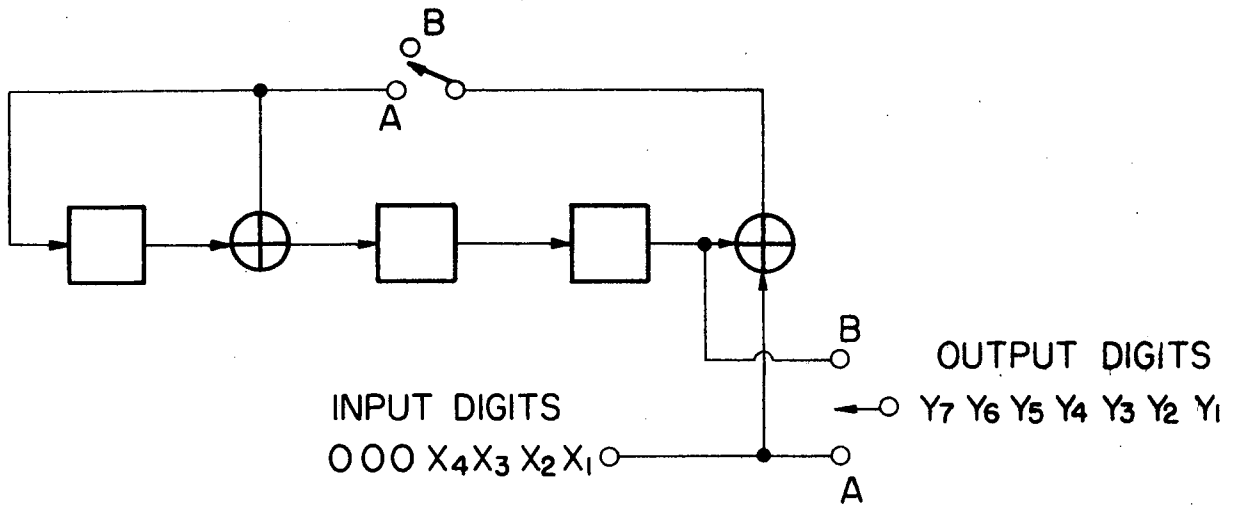

B
A
B
OUTPUT DIGITS
Y7 Y6 Y5 Y4 Y3 Y2 Y1
INPUT DIGITS
000 X4X3 X2X1
A

FIG 2

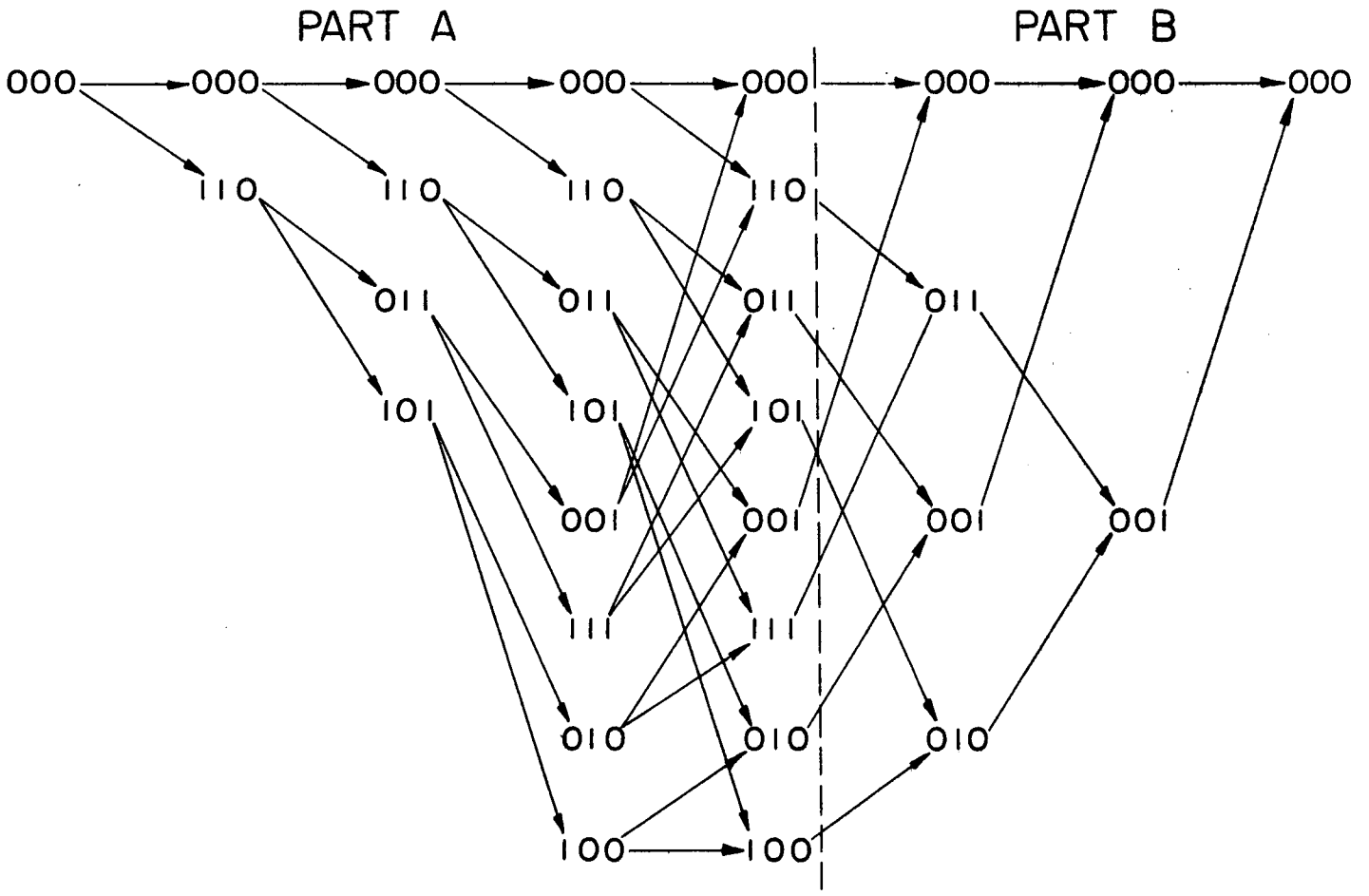

PART A
PART B
000
110
011
101
001
111
010
100

FIG. 3

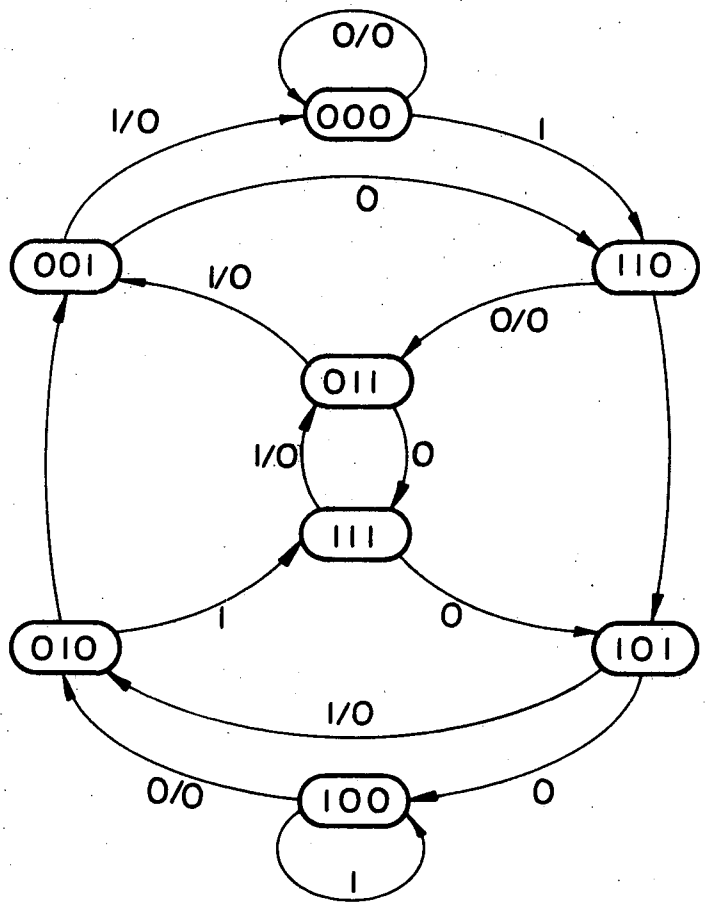


FIG. 4

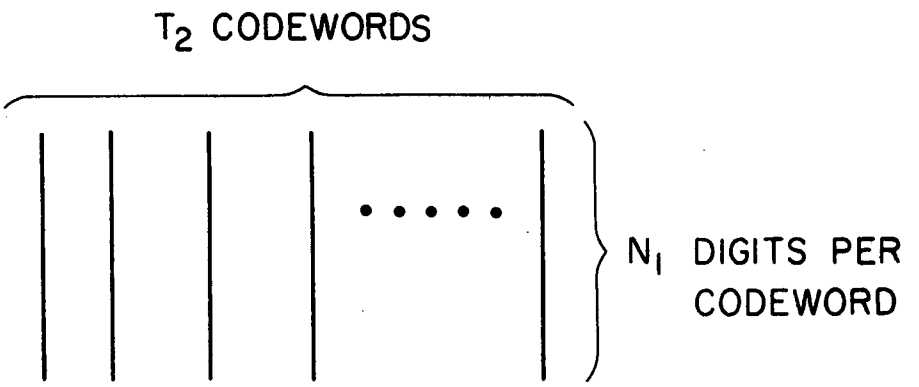

T_2 CODEWORDS
N_1 DIGITS PER CODEWORD

FIG. 5

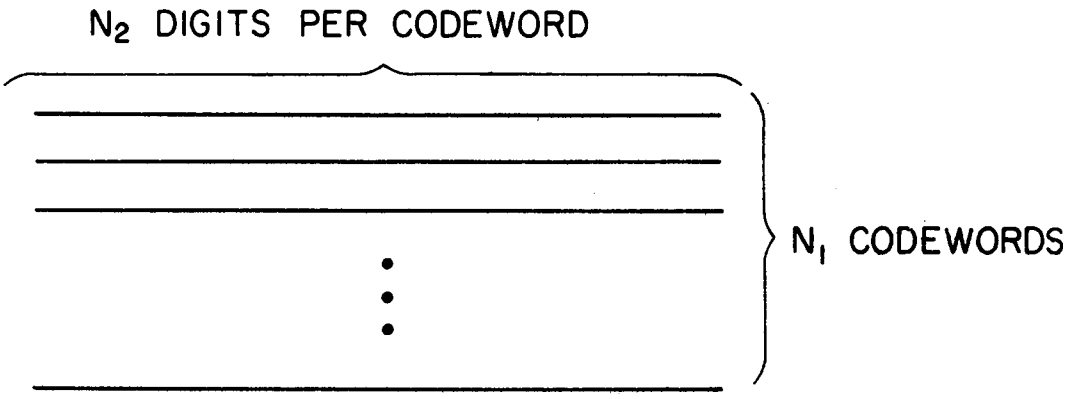

N_2 DIGITS PER CODEWORD
N_1 CODEWORDS

FIG. 6

II-I An Algorithm Determining Free Distance of Convolutional Codes

The algorithm to be described here works for convolutional codes of all rates $R = \frac{k}{n}$. However, for simplicity of exposition we will confine ourselves to rate 1/n binary codes.

It will be useful to take the old-fashioned point of view that the state S(t) of a convolutional encoder at time t is defined by υ immediately preceeding information digits

$$S(t) = \left[i_t, i_{t-1}, \ldots, i_{t-\upsilon+1} \right] \tag{1}$$

and that the encoder output block $\underset{\sim}{x}^n = x_1, \ldots x_n$ at time t is a function of S(t) only.

If the code is non-catastrophic then the free distance d_f is equal to the minimal weight of a codeword that corresponds to some information sequence of the form

$$(1, i_2, i_3, \ldots, i_{m-1}, 1, 0, 0, \ldots) \tag{2}$$

where m = 1, 2, 3,... . We will, of course, restrict our attention to non-catastrophic codes only (tests for possible catastrophic character of codes are simple).

It follows for (2) that free distance will be achieved on a path defined by a state succession S(1), S(2),..., S(m+υ-1), S(m+υ),... where

$$\begin{aligned} S(1) &= (1,0,\ldots,0) \\ S(m+\upsilon-1) &= (0,\ldots,0,1) \\ S(m+\upsilon) &= S(m+\upsilon+1) = \ldots = (0,0,\ldots,0) \end{aligned} \tag{3}$$

Furthermore, $S(t+1)$ is obtainable from $S(t)$ by a right-shift followed by insertion of i_{t+1} into the leftmost state position ($t=1,2,\ldots,m+\upsilon-2$) and $S(k-1)$ is obtainable from $S(k)$ by a left-shift followed by insertion of $i_{k-\upsilon+1}$ into the rightmost state position ($k=m+\upsilon-1, m+\upsilon-2,\ldots,2$). Assume for the time being that we have the following two machines:

a) A right-shifting machine whose starting state is $(1,0,\ldots,0)$ which searches the trellis in the forward direction: computing outputs, recording their weight, adding the latter to the cumulative weight that corresponds to the path from the root code $(1,0,\ldots,0)$ to the state in question, and keeping track of the states (regardless of depth) already visited.

b) A left-shifting machine whose starting state is $(0,\ldots,0,1)$ which searches the trellis in the backward direction (again recording the states visited).

If one of the machines ever reaches a state already reached by the other machine, then a path connection is established whose information digit form is that of (2) and which therefore possibly achieves free distance. This is the main idea of the bi-directional search for d_f being proposed here.

For obvious reasons of economy, both machines should extend low weight paths first. As a consequence, for a rate $R = 1/n$ code, the memory of each machine will contain at any given time only extendible paths whose weights are $w, w+1,\ldots, w+n$.

Both 0 and 1 extensions, π^0 and π^1, of a path π ending in state $S(t) = (i_t, i_{t-1},\ldots,i_{t-\upsilon+1})$ will be generated simultaneously. Let $S^0(t+1) = (0,i_t,\ldots,i_{t-\upsilon+2})$ and $S^1(t+1) = (1,i_t,\ldots,i_{t-\upsilon+2})$ be the

last states of π^0 and π^1 respectively, and suppose (w.ℓ.o.g.) that the right-shifting machine already generated some other path π^* whose end state was $S^0(t+1)$. If that path was previously extended, then its cumulative weight at that time could not have exceeded the weight of path π. Hence the path π^0 can be eliminated from consideration. If, on the other hand, π^* was not extended by the time π^0 is generated, then either π^0 or π^* can be eliminated depending on which has the larger cumulative weight. In fact, suppose $w_H(\pi^0) < w_H(\pi^*)$, and the left shifting machine generates a path π^+ whose last state is $S^0(t+1)$. Then, obviously, the concatenation π^0,π^+ may correspond to a sequence (2) of least weight, but π^*,π^+ cannot. We therefore conclude that at any given time the memory of the right-shifting machine need contain only paths ending in (live paths) or leading through (dead paths) distinct states. Same remarks, of course, apply to the left-shifting machine.

As a matter of fact, when the search for d_f is carried out by a digital computer, no left or right-shifting machines need be simulated. All that is necessary is to attach a three-valued flag to each state ever reached from left or right. The flag's value is 'D' if the state was already extended, and it is 'R' if the state is to be extended by a right-shift and it is 'L' otherwise (e.g., the flag value of S(t) when it was generated was 'R'. When the extensions $S^0(t+1)$ and $S^1(t+1)$ were generated, their flag values became 'R', and the flag value of S(t) changed from 'R' to 'D').

We are now ready to describe the algorithm. The storage consists of three arrays: The first, S, gives the state, the second, F, the flag value, and the third W, gives the cumulative weight of the path leading to the state S: W^* will denote the current upper bound on d_f.

It will originally be set equal to nu. If T is a state, $\Delta_T W$ will denote the weight of the output branch corresponding to T.

1. Place (1,0,...,0) into the first S-location, 'R' into the first F-location, and the weight of the output of (1,0,...,0) into the first W-location.

2. Place (0,...,0,1) into the second S-location, 'L' into the second F-location, and the weight of the output of (0,...,0,1) into the second W-location.

3. Search through memory for a non-'D' location whose W-value is least. Let it be found at location J. If $2W(J) \geq W^*$, go to 19.

4. Set T = S(J) and K = 0 (K is an indication whose values are 0 and 1). If F(J) = 'L', go to 6.

5. Shift T right and place a 0 into the leftmost position of T. Go to 7.

6. Shift T left and place a 0 into the rightmost position of T.

7. Search through memory for some location I such that S(I) = T. If such I exists, go to 13.

8. Find M, the first non-occupied location. Then set S(M) = T, $W(M) = W(J) + \Delta_T W$, F(M) = F(J)

9. If K = 1, set F(J) = 'D' and go to 3.

10. If F(J) = 'L', go to 12.

11. Place a 1 into the leftmost position of T. Let K = 1. Go to 7.

12. Place a 1 into rightmost position of T. Let K = 1. Go to 7.

13. If $F(I) \neq$ 'D' go to 15.

14. Go to 9.

15. If $F(I) \neq F(J)$ go to 18.

16. If $W(J) + \Delta_T W \geq W(I)$, go to 9.

17. Purge location I, and make it available. Go to 8.

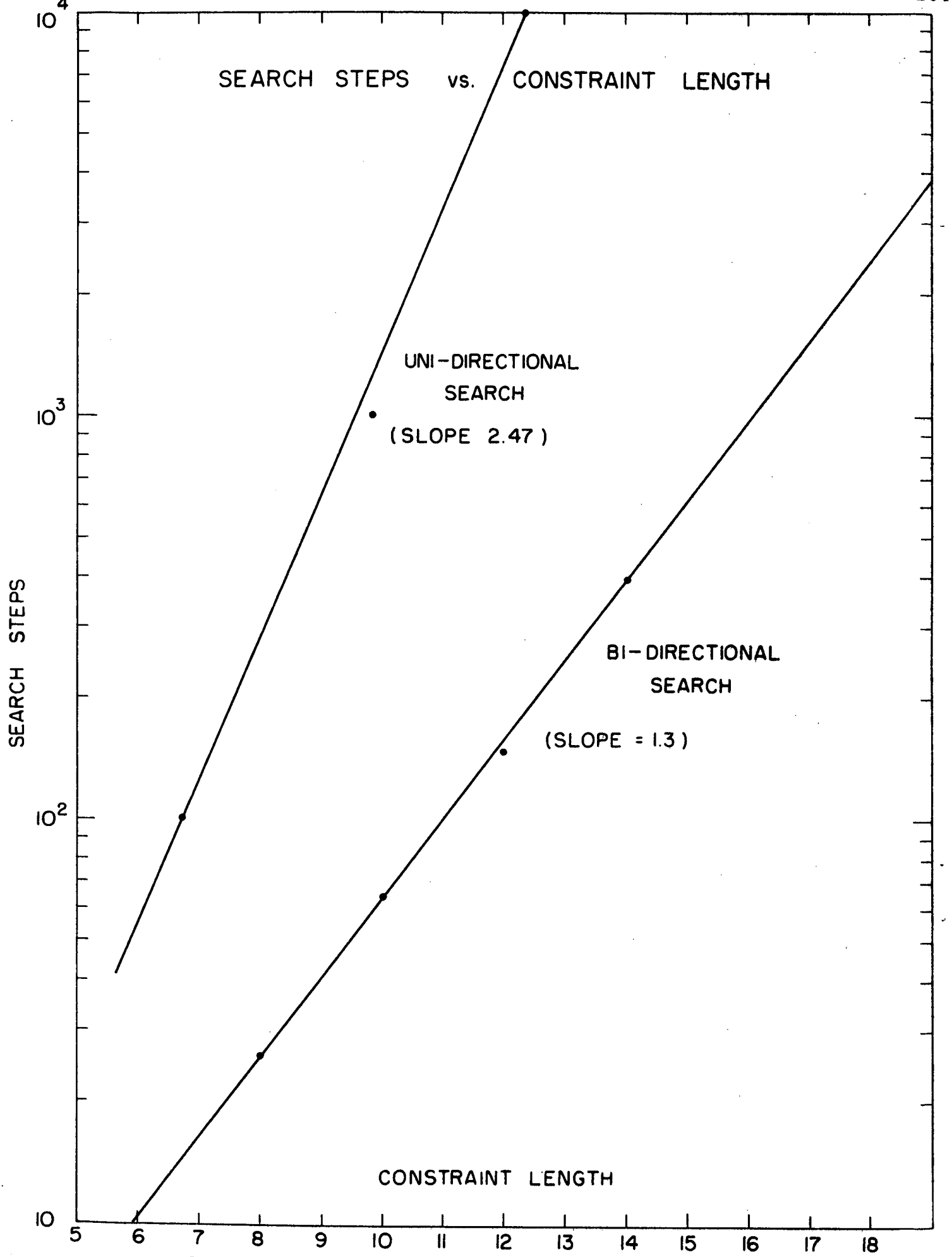
18. If $W^* > W(J) + \Delta_T W + W(I)$, set $W^* = W(J) + \Delta_T W + W(I)$. Go to 9.

19. The free distance is W^*. Stop. Figure 1 shows the number of search steps as a function of constraint length υ, and compares them with the number of steps involved in the conventional stack-type search. It is seen that on a semi-log plot, the slope of the latter is approximately twice that of the former.

This is just as one would expect: each direction of search need now be carried out only to about half of the depth as formerly, and an exponentially growing tree arrangement exists in both directions.

There is, of course, one obvious difficulty connected with this algorithm: the size of the storage and the search through it. To reduce the former would mean to change the algorithm, but an efficient storage organization to minimize the search is essential. If there are 2^{υ} storage locations available, then there is no problem: each possible state is assigned a definite address, and the algorithm simply checks at the appropriate address if the state in question has already been generated, etc. If the available storage is smaller (its minimal order of magnitude is a direct function of the number of search steps) a more efficient organization is necessary. We have tried some simple hashing schemes which seem to work excellently as long as the occupancy stays below 60%, and we will experiment with tree arrangements involving pointers.

The algorithm applies to rate $R = \frac{k}{n}$ codes as well. There are now $2(2^k-1)$ initial states, (10...00...0) through (11...10...0) and (0...00...01) through (0...01...11), and every path is extended into 2^k paths, one for each possible outgoing branch. Otherwise the algorithm stays the same.


SEARCH STEPS vs. CONSTRAINT LENGTH
UNI-DIRECTIONAL SEARCH
(SLOPE 2.47)
BI-DIRECTIONAL SEARCH
(SLOPE = 1.3)
SEARCH STEPS
CONSTRAINT LENGTH
10^4
10^3
10^2
10
5
6
7
8
9
10
11
12
13
14
15
16
17
18

III. REPORT ON PHASE 2

III-A. The Two-Cycle Algorithm

1. Introduction

In this section we will describe the two-cycle algorithm and summarize our analytical results for it. A long paper by J.B. Anderson and F. Jelinek entitled "A Two Cycle Algorithm for Source Coding with a Fidelity Criterion" going into the details was presented at the 1972 International Symposium on Information Theory and will be published in the IEEE Transactions on Information Theory.

In the 2-cycle algorithm, the encoder will work in two fundamental modes, called cycles, one embedded within the other. In the first mode a search is made among tree paths to find feasible candidates for encoding of the generated information. In the second mode, the candidates are concatenated with the help of a push-down stack. The operation is, in a way, not too different from that suggested in Jelinek's original proof of the three coding theorem. What makes analytical evaluation possible and the algorithm desirable (from an encoding effort standpoint) are the kinds of stopping rules introduced to limit the amount of work in each mode.

Assume that code words for encoding of a binary digit IID source

have been arranged in a tree structure. The tree has rate $R = \log_2 d/n$, with d branches stemming from each node and n source approximating binary digits on each branch. The object of an encoder is to find a path of branches through the tree, the digits of which approximate the source sufficiently closely. To measure distance between the source output and various paths, we use the Hamming measure

$$d(\underset{\sim}{z}^{\ell}, \underset{\sim}{\hat{z}}^{\ell}) = \sum_{i=1}^{\ell} \left[1 - \delta(z_i, \hat{z}_i)\right] \tag{1}$$

where $\underset{\sim}{z}^{\ell}$ is a source sequence, $\underset{\sim}{\hat{z}}^{\ell}$ is an hypothesized path, (both of length ℓ) and δ is the Kronecker delta function. It should be stressed that our encoder works for other measures and sources as well.

Goodness of individual paths depends on path length as well as distortion and is compared by the algorithm with the help of a path metric,

$$\mu(\underset{\sim}{\hat{z}}^{\ell}) = \ell D^* - d(\underset{\sim}{z}^{\ell}, \underset{\sim}{\hat{z}}^{\ell}) \tag{2}$$

Since a path involves an integral multiple of branches to be of interest, ℓ is assumed to be a multiple of n. D^* is the target distortion per encoded source digit desired at the end of encoding, and $D^* > \Delta(R)$, the inverse rate distortion function relative to (1) and the source.

With this path metric in mind, we define two freezing barriers (in the terminology of Gallager), one at metric $a > 0$, the other at $b < 0$. Further extension of paths whose metrics rise above a will be frozen temporarily and the paths removed to the push-down stack, (these are the live links) while paths falling below b will be dropped entirely.

A precise description of the algorithm follows:

Step (1) Starting at the code tree root node (which is assigned the metric zero), a freezing cycle is performed: Paths are extended in an exhaustive search until all root node descendants crash a freezing barrier and are frozen. Those paths that rise above the a barrier are placed at the top of a push-down stack.

Step (2) When a freezing cycle terminates, attention turns to the push-down stack. The final node of the path at the top of the stack now becomes a root node (metric value 0 assigned) for a new freezing cycle, and the encoder executes again Step (1). As described in Step (3), the top stack path may occasionally be saved. If the stack is either empty, or its top contains a path made up of a concatenation of L links from Step (1), the encoder passes to Step (3).*

---

* The push-down stack requires no sorting effort, since paths are inserted as they come and are removed at the top. The resulting stack of paths is thus naturally ordered by the number of live links each path consists of, the longest (in terms of links, but not necessarily branches) being on top. To order paths according to branch lengths is another possibility that may involve extra sorting work. We do not know how to take proper analytical advantage of such an improvement. The fastest way to carry out the freezing cycle would seem to be a Fano-type search that would take the 0-branch extension first until freezing is achieved, and then backstack. In this way the ordering of live links within each freezing cycle would be lexical. If, on the other hand, all extensions were to be carried out by depth, then the links would be inserted into the stack in the desirable branch-length order.

Step (3) When the push-down cycle defined by Steps (1) and (2) terminates, the encoder releases the output to the user. If an L-concatenation has appeared, it is released directly, and an L-termination is said to have occurred. In the event of an empty stack, the push-down cycle has terminated by extinction. To defend against this, the encoder keeps track of the longest concatenation found by the push-down cycle and returns to it if extinction occurs. Step (1) is performed for the second time beginning on the last node of this path. The first frozen path encountered (it must be at barrier b!!!) is then concatenated with the saved path and released as the codeword to the user.

Step (4) When an encoding takes place, the push-down stack is purged and the last node of the obtained codeword is inserted into the stack. The latter then constitutes a new root node for further operation of the encoding algorithm.

Step (1) constitutes the freezing cycle, and Steps (1) and (2) together are the push-down cycle. Step (3) implies release of accumulated output, and the time between successive executions of this step is the delay in encoding. The analysis of our algorithm is an interesting one in itself, but the scheme has several practical advantages. The freezing cycle need not be extensive, and far less time is spent scrutinizing codewords than with the Jelinek stack algorithm. In general, efficiency and simplicity are well combined.

Before proceeding with an analysis, we pause to develop further terminology and identify quantities of interest. The language of tree

structures is well suited to our discussion, except that the two-cycle algorithm contains two tree structures, one "within" the other, which are easily confused. Accordingly, let the code tree paths be made up of branches, of which d stem from each node, but let the tree structure diagramming the push-down stack development consist of links. In this tree, sons of a node are formed by a freezing cycle, their number being a random variable, and paths of links represent concatenations of the "good paths" alluded to above. Corresponding to each link is a link length in branches of the code tree, and a stack tree node has sons equal in number to the code tree paths frozen at a during some freezing cycle. The subject of code trees is well known, and the growth of the stack tree, a process we call a push-down stack searched branching process, will be estimated in Section 3. The process terminates either by extinction, or by E-termination.

2. Quantities of Interest in the Two-Cycle Algorithm

We now discuss quantities of interest in the operation of the two-cycle algorithm: Computation per source digit encoded, computation per freezing cycle, freezing cycles per push-down cycle, probability of termination by extinction, concentration of work in one or the other cycles, and of course, the distortion attained. All of these eventually must depend on the three parameters of the algorithm, a,b, and L.

Let the term live link refer to an a-frozen link, and dead link to the occasional b-frozen link (recall the push-down process involves a-frozen links only). Let the path that constitutes the codeword released to the user be referred to as the chosen link path. Let the latter be of length ℓ, and let X_i be the branch length of the i^{th} link. Let Y be the branch length of the last (and only!) dead link, if any,

of the chosen path. Then the chosen path branch length M is given by (ℓ is a random variable not exceeding L)

$$M = \sum_{i=1}^{\ell} X_i + Y\,[1-\delta(\ell,L)] \tag{3}$$

and the total distortion incurred in encoding is

$$D_{Tot} = D^*M - \ell a - [1-\delta(\ell,L)]b \quad , \quad b < 0 \tag{4}$$

Let W_i be the computation performed in the code tree during the i^{th} freezing cycle, and let V be the number of freezing cycles necessary to complete a push-down cycle. Then U, the total computation expended in a push-down cycle, is

$$U = \sum_{i=1}^{V} W_i \tag{5}$$

Among our interests is the relation between the average distortion per encoded source digit $E\left[\frac{D_{Tot}}{M}\right]$, and the average work per encoded source digit $E\left[\frac{U}{M}\right]$. Under suitable conditions, satisfied in this case,

$$E\left[\frac{D_{Tot}}{M}\right] = \frac{E[D_{Tot}]}{E[M]} \tag{6}$$

$$E\left[\frac{U}{M}\right] = \frac{E[U]}{E[M]} \tag{7}$$

Let q_i be the probability that the push-down cycle terminates by extinction before any link on tree level i has been generated. Clearly $q_i \leq q_{i+1}$ Let

$$q = \lim_{i \to \infty} q_i$$

It can be shown that a proper choice of $a > o$ and $b < 0$ results in $q < 1$. Assuming that to be the case, let us choose L to satisfy

$$L = \frac{q}{1-q} \frac{|b|}{a} \tag{8}$$

Then from (4) and (6)

$$E\left[\frac{D_{Tot}}{M}\right] = D^* - \frac{aE[\ell] + bE[1-\delta(\ell,L)]}{E[M]} \tag{9}$$

But

$$E[1-\delta(\ell,L)] = q_L \tag{10}$$

and

$$E[\ell] = \sum_{\ell=1}^{L-1} (q_{\ell+1} - q_\ell)\,\ell + (1-q_L)L \geq (1-q_L)L \tag{11}$$

Hence, using (8), (10), and (11)

$$\begin{aligned} -aE[\ell] - bE[1-\delta(\ell,L)] \leq \\ \leq -a(1-q_L)L + aL\frac{(1-q)}{q} q_L \\ = aL\left(\frac{q_L - q}{q}\right) \leq 0 \end{aligned} \tag{12}$$

It follows that L chosen as in (8) causes

$$E\left[\frac{D_{Tot}}{M}\right] \leq D^* \tag{13}$$

Next, the computation in successive freezing cycles is independent

under our assumptions, so by Wald's Lemma,

$$E[U] = E[V]\, E[W] \tag{14}$$

and

$$\begin{aligned} E[M] &= E[\ell]\, E[X] + q_L\, E[Y] \\ &\geq (1 - q)\, L\, E[X] \end{aligned} \tag{15}$$

where we have made use of (11).

Hence

$$E\left[\frac{U}{M}\right] \leq \frac{E[V]\, E[W]}{L(1-q)\, E[X]} \tag{16}$$

A characteristic of push-down stack searched branching processes is that the underbound of (11) is quite tight, so that the bounds (13) and (16) are also tight. Thus, (16) gives the computation required to produce distortion D*.

Since q is a function of $\underline{a}$ and $\underline{b}$ only, then $\underline{a}$, $\underline{b}$, and L are all implicitly present in (16). It turns out that certain choices of $\underline{a}$, $\underline{b}$, and L decrease the computation in one cycle at the expense of the other (e.g., smaller freezing cycles, but more of them, or vice versa). Obviously, some combination minimizes the bound (16) while preserving the validity of (13). To complete our analysis, we must study

i) E[W], the expected number of computations in a freezing cycle

ii) E[V], the expected number of freezing cycles in a push-down cycle

iii) q, the probability of extinction in a push-down cycle that has $L = \infty$

iv) E[X], the expected branch length of a live link

v) Choices of a, b, and L

3. Summary of Analytical Results for the Two-Cycle Algorithm

For this progress report, we summarize briefly the analytical results that have been obtained to this date. Only the simplest equations and no proofs will be given. A full length report on the two-cycle algorithm will be forthcoming.

i) Expected freezing cycle computations

In the code tree, let

N_a = Number of paths frozen at a-barrier (i.e., live links)

N_b = Number of paths frozen at b-barrier (i.e., dead links)

N_∞ = Number of paths remaining forever unfrozen

Then the following theorem is true:

Theorem 1 For a tree with rate $R = \log_2 d/n$ used to encode binary IID sources with respect to the Hamming distortion measure,

$$EN_a \sim \frac{r^{-a}}{\text{sinwa}} \Big/ \left(\frac{\text{coswa}}{\text{sinwa}} - \frac{\text{coswb}}{\text{sinwb}} \right) \tag{17a}$$

$$EN_b \sim \frac{r^{-b}}{-\text{sinwb}} \Big/ \left(\frac{\text{coswa}}{\text{sinwa}} - \frac{\text{coswb}}{\text{sinwb}} \right) \tag{17b}$$

whenever $b - a < \pi/w$. $s = r\, e^{iw}$ is the possibly complex solution to

$$2^{1-R} = S^{D^*-1} + S^{D^*} \tag{18}$$

w and r are functions of D* and R only.

$w \searrow 0$ as $D^* \searrow \Delta(R)$ and r is typically near $(1-D^*)/D^*$. A careful look at (17) reveals that as $|\, b-a \,|$ tends to π/w, both EN_a and EN_b tend to infinity. In fact, given an a one may choose b to make the right hand side of (17a) precisely unity. In this way, R, D*, and a specify a minimal b necessary to achieve $EN_a > 1$. We can state this as a

Corollary 1 For any given $a < \pi/w$, there exists b* such that if $|\, b-a \,| < \eta/w$ and $b < b^*$, then $EN_a > 1$

As a rule, b* is very near π/w. A second corollary will give us the desired result for E[W]. As is customary, let one computation include the generation and scrutiny of d branches stemming from their common parent node. Then an exercise in tree branch topology yields

Corollary 2 $$E[W] = \frac{EN_a + EN_b - 1}{d-1} \tag{18}$$

The significance of $EN_a > 1$ is given by Theorem 2, which amounts to a coding theorem proved by the device of a two-cycle algorithm:

Theorem 2 Under the hypotheses of Theorem 1, whenever $EN_a > 1$ and $D^* > \Delta(R)$, the two-cycle algorithm along with some source code will perform arbitrarily close to D* for some L.

ii) Expected freezing cycles per push-down cycle

An effective means of analysis has been found for the push-down

stack, that shows, among other things, the surprising theorem to follow. Let the distribution $\{ p_k \}$ be defined by

$$p_k = \Pr \{N_a = k\} = \text{probability of k sons of a stack tree node}$$

Theorem 3 For any distribution $\{p_k\}$ such that $Ek > 1$ (i.e., $EN_a > 1$), the expected number of son formations, $E[V]$, necessary to terminate a push-down cycle is overbounded by

$$E[V] \leq L \tag{19}$$

(Recall that L is the termination depth of the cycle when extinction does not occur, but the expectation is over either termination).
We conjecture that (19) is a tight overbound.

iii) Probability of push-down cycle extinction

In the event of extinction, the push-down cycle behaves identically to an ordinary branching process. Exploiting this relationship gives q. In particular, whenever $EN_a < 1$, the monotone increasing sequence $\{q_i\}$ has limit 1, so that for large L, extinction occurs with probability 1. When $EN_a > 1$, q is the solution of the polynomial equation

$$q = \sum_{k=0}^{\infty} p_k q^k \tag{20}$$

It remains only to find the distribution $\{p_k\}$, and it turns out that each $p_k(a,b)$ is the solution of a linear difference equation with non-constant coefficients. These equations are easy to solve numerically, although much more complicated analytical methods are available also.

iv) Expected length of live links

Recursions are now available to find the expected length of a live path searched out by the freezing cycle. These recursions allow also the study of freezing cycles with a length restriction on searching in the code tree. Such a feature is important as a practical matter to insure the steady operation of the encoder. For lack of time, numerical analysis of these recursions has not as yet been undertaken.

v) Choices of a, b, L

Intensive work on this problem is awaiting further numerical analysis. Increasing a will increase q and increasing $|b|$ will have the opposite effect. Simultaneous increase in a and $|b|$ will increase E[W] but might conceivably decrease L (see (8)). The point is that the amount of work in the push-down cycle might be traded for work in the freezing cycle, and there will exist some optimal balance that we shall seek to discover.

III-B. The Stack Algorithm for Source Coding

The stack algorithm is a scheme that uses tree codes to encode source data with respect to a fidelity criterion. It stems directly from the Jelinek stack algorithm [1] for sequential channel decoding, but differs radically in its analysis. In terms of code tree branches searched per digit output, it is the most efficient algorithm known to the authors (see [2], [3], [4]) . The algorithm suffers, however, from clumsy data handling and large storage.

The stack algorithm is simple to describe and consists of one repeated basic operation, the stack augmentation. Hypothetical code tree paths $\hat{\underset{\sim}{z}}^k$ of varying lengths k, ordered by the usual metric

$$\underset{\sim}{\mu}(\hat{\underset{\sim}{z}}^k) \triangleq kD^* - d(\underset{\sim}{z}^k, \hat{\underset{\sim}{z}}^k) \qquad (1)$$

reside in a stack. From the top path in the stack, the d branches stemming from its final node are extended to form d new paths. Stacking these in order of metric, the algorithm completes an augmentation. Repetition continues until a stopping rule intervenes.

Suppose the algorithm stops and releases output when a

path exceeds metric $A>0$ for the first time, that is, when the "top" of the ordered stack exceeds A. We can imagine a bottom limit $B<0$ below which all paths are dropped from the stack, and a limit t on the length of tree paths stored in the stack. Our analysis is sufficient for this generality, but for simplicity consider a stack of infinite capacity to store nodes, with $B = -\infty$ and $t = \infty$. With these assumptions, the average stack storage in branches is identical to the expected number of nodes scrutinized by the algorithm, since no paths are ever dropped. Furthermore, if this expectation is EN(A,B) -- with $B = -\infty$ -- then the number of nodes searched per branch released as output, over many stack searches, is

$$E[\text{Nodes per branch}] = EN(A,-\infty)/EL \qquad (2)$$

where EL is the expected length of a released path. The expected distortion of this path will depend on A as well as D*, and is

$$E[\text{Dist. per branch}] = nD^* - \frac{A}{EL} \qquad (3)$$

Similar, but more complicated, equations hold if B and t are not indefinitely large.

Our analytical method is to identify the tree search with linear and non-linear difference equations, and then approximate these. The non-linear equations predominate, unfortunately, and the stack sorting will require a careful mathematical model. Quantities needed will be the average nodes searched EN(A,B), the average length releases EL, and the probability distribution of the top-of-stack minimum (TSM). The latter describes how low

the metric of the best stack path drops before some path is finally released.

Define the function G(y) by means of its d^{th} power to be

$$G^d(y) = P\left\{\text{Forward of some node } n_o \text{ in TSM} \leq B \;\middle|\; \underset{\sim}{\mu}(n_o) = y,\ B < y < A\right\} \quad (4)$$

n_o can be any node encountered during the stack search, and $\underset{\sim}{\mu}(n_o)$ represents the value of (1) at that node. Then one can show that G( ) satisfies the non-linear difference equation with constant coefficients,

$$G(y) = \sum_m p(\mu_m)\, G^d(y+\mu_m), \quad B < y < A \quad (5)$$

$\{\mu_m\}$ is the set of (IID) metric increments that can appear in the tree code, and p( ) is their distribution. G( ) gives the distribution of the stack top, but turns out to be far more important than that. As we shall now see, every stack quantity is directly related to G( ), and the study of the algorithm consists almost entirely of manipulating this function.

After a careful derivation, taking into account the stack sorting, one gets that

$$EN(A,B) = \sum_{j=1}^{B} M(j/A+j) \quad (6)$$

where the family of functions $\{M(\cdot/i)\}$ are solutions of linear difference equations with non-constant coefficients of the form

$$M(y/i) = d\, G^{d-1}(y/i) \sum_m p(\mu_m)\, M(y+\mu_m/i) + C(y/i) \quad (7)$$

An equation (7) exists for each i, i=1,...,I(A,B). I(A,B) is a

finite integer function of A and B. All I(A,B) solutions are needed to compute (6). $C(\cdot/i)$ is calculated from G( ) functions ; $G(\cdot/i)$ is the solution of (5) for certain boundaries specified by i.

A final derivation yields that

$$EL = \sum_{\ell=0}^{\infty} G^{d}(0) \tag{8}$$

assuming the stack search begins at a root node with metric 0. The $\{G_\ell(\ )\}$ are obtained from iterations of the recursion

$$G_\ell(y) = \sum_{m} p(\mu_m)\, G_{\ell-1}^{d}(y+\mu_m) \quad , \quad B < y < A \tag{9}$$

$$G_o = 1 \qquad \text{(Boundaries as in (5))}$$

which provides incidentally a numerical means to solve (5), since it can be shown $G_\ell(y) \downarrow G(y)$.

Using these equations (4)-(9), extensive numerical studies have been conducted for a stack algorithm using a randomly chosen tree code to encode the binary IID source with Hamming fidelity criterion $d(z,\hat{z}) = 1 - \delta(z,\hat{z})$. In addition, a FORTRAN stack encoder has simulated the same situation. To summarize these results, observe that distortion is a function of both D* and A. If one optimizes A and D* for smallest storage, A will be as small as possible, with D* as a consequence very near the distortion desired from the algorithm. On the other hand, optimizing with respect to branch computation requires a larger A and a D* somewhat above the final distortion.

It turns out that the stack search involves by far fewer

tree branches per digit released as output than any other scheme studied by the authors [2], [3]. But this strong advantage is balanced by several disadvantages. Both computation and length released vary widely from search to search, and storage is large. A difficulty of another sort, encountered during simulation, is sorting effort. After each augmentation, d new paths must be sorted into the stack in order of metric, and among paths of the same metric, in order of length. In general, this is not easily done. New paths typically are inserted far down into the stack, particularly if some of the branch increments are reasonably negative, since many other paths usually have metrics nearer the best.

Overall, it appears that the efficiency in branches studied is overbalanced by this clumsy sorting. Algorithms such as the M-algorithm [2] and the 2-cycle algorithm [3] have proved faster in simulation thus far, and simpler to implement. But improvements in all the algorithms are always a possibility, and the subject is not closed.

References

1. F. Jelinek, "A Stack Algorithm for Faster Sequential Decoding of Transmitted Information," IBM J. Res. Develop., 13, #6, pp. 675-85, Nov. 1969.

2. F. Jelinek and J.B. Anderson, " Instrumentable Tree Encoding of Information Sources," IEEE Trans. Inform. Theory, IT-17, pp. 118-119, Jan. 1971.

3. J.B. Anderson and F. Jelinek, "A 2-Cycle Algorithm for Source Coding with a Fidelity Criterion," to be published in IEEE Trans. Inform. Theory.

4. J.B. Anderson, "Algorithm for Tree Source Coding with a Fidelity Criterion," Ph.D. Thesis, School of Elec. Engrg., Cornell University, May 1972.

III-C. Development of a Stack Algorithm for Tree Encoding of a Gaussian Source with a Mean Square Fidelity Criterion

1. Introduction

The most general theoretical formulation of the data compression problem was provided by Shannon in 1959 in his paper "Coding Theorems for a Discrete Source with a Fidelity Criterion" [1]. He enlarged there on his 1949 source coding ideas [2] referred to in the literature as variable length source coding and block source coding. Concisely stated, Shannon's results are as follows: let a memoryless source of alphabet $A = (0,1,\ldots,a\text{-}1)$ governed by the probability distribution $Q(z)$, $z \epsilon A$ be given. Let an approximation of the source outputs in the reproducer alphabet $B = (0,1,\ldots,b\text{-}1)$ be desired (in practice $b \leq a$) with an attached additive per letter distortion criterion $d(z,\hat{z})$ defined for all pairs $z \epsilon A$, $\hat{z} \epsilon B$. (i.e., the distortion between sequences $\hat{\underset{\sim}{z}}^n = \hat{z}_1,\ldots,\hat{z}_n$ and $\underset{\sim}{z}^n = z_1,\ldots,z_n$ is defined to be $d(\underset{\sim}{z}^n;\hat{\underset{\sim}{z}}^n) = \sum_{i=1}^{n} d(z_i,z_i)$). Let $\psi_n(\underset{\sim}{z}^n)$ be an encoding function that assigns some reproducer sequence $\hat{\underset{\sim}{z}}^n$ to each possible source sequence $\underset{\sim}{z}^n$. The rate of the resultant code is defined to be $R = \log \psi_n/n$ where ψ_n denotes the number of sequences in the range of $\psi_n(\ )$. Shannon shows the existence of a rate distortion function $R(D)$ [whose shape depends on $Q(\ )$ and $d(\ ,\ )$ only] that has the following properties:

a) for all n and all codes ψ_n, if $R<R(D)$ then the expected distortion $E[\frac{1}{n} d(\underset{\sim}{z}^n;\ \psi_n(\underset{\sim}{z}^n))] > D$.

b) for $R \geq R(D)$ there exists a sequence of codes ψ_n^* of rate $\log \psi_n^* / n \leq R(D)$ such that $E[\frac{1}{n} d(\underset{\sim}{z}^n; \psi_n^*(\underset{\sim}{z}^n))] \to D$.

In recent years much work has been done generalizing the above results to a broader class of sources, evaluating the performance of existing systems relative to the achievable optimum, and developing methods for evaluation of the R(D) function. The first consideration of the actual coding problem was undertaken by Jelinek [3] who showed that the sequence of coding functions ψ_n^* can possess the above desirable properties even if it is restricted to generate tree codes (instead of block codes to which Shannon's theorem applies). It was hoped that a tree code structure would facilitate the development of computationally feasible encoding algorithms.

The present report concerns the performance of two such algorithms as applied to the restricted case of the time discrete Gaussian memoryless source [with probability density

$$Q(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}, \quad x \text{ real }]$$

and the squared error criterion $[d(z, \hat{z}) = (z - \hat{z})^2]$.

For this case the R(D) function is $R = -\frac{1}{2} \log D$. Furthermore, for this case it can be shown that any sequence of codes ψ_n^* with rates $\log \psi_n^* / n \to R(D)$ and distortions $E[d(\underset{\sim}{z}^n, \psi_n^*(\underset{\sim}{z}^n))/n] \to D$ must have the average conditional distortions

$$\frac{1}{n}\sum_{k=1}^{n} E[d(z_k, \psi_n^*(z^n)) \,|\, z_k = x] \to D(1 - D) + D^2x^2$$

almost everywhere in x where z_k is the kth element of $\underset{\sim}{z}^n$ and $\psi^*_{n,k}(\underset{\sim}{z}^n)$ is the kth element of $\psi^*_n(\underset{\sim}{z}^n)$.

An example of a tree code with 4 branches per node and two initial states is given in Figure 1. The various codewords are the sequences associated with the $2 \times 4^2 = 32$ different paths of the tree. For a tree with b_o initial states and b branches per node a path of length ℓ is specified by a map sequence $\underset{\sim}{s}^{\ell} = (s_0, s_1, \ldots, s_\ell)$ where the s_i's are non-negative integers, $s_o \le b_o - 1$, $s_i \le b - 1$ for $i = 1, 2, \ldots, \ell$. This map sequence determines which initial state was taken and at each node level determines if the first (0), second (1), ..., or b^{th} (b-1) branch was taken. Thus for the tree of Figure 1 the map sequence $\underset{\sim}{s}^2 = 11$ corresponds to the codeword $\underset{\sim}{z}^2 = (-0.87, 0.60)$. The rate of the code of Figure 1 is $R = \frac{1}{2}\log 32 = 2.5$ bits.

A convenient method of filling the tree is by means of a finite state tree encoder. In this method each branch in the tree is associated with a state as follows: branch s_j of path $\underset{\sim}{s}^j = (s_0, s_1, \ldots, s_j)$ is assigned state $(\ell(j), t(\underset{\sim}{s}^j))$, where time state $\ell(j) = j$ (modulo r) and branch state

$$t(\underset{\sim}{s}^j) = (s_o b^j + \sum_{i=1}^{j} s_i b^{j-i}) \text{(modulo m)}$$

and the period r and number of branch states m are positive integers.

Then each state is given an element of the reproducer alphabet and each branch is given the element assigned to its state. An example of a finite state tree encoder with r = 2 and m = 8 is shown in Table 1. This code gives the tree of Figure 1 when used to fill a tree with $b_o = 2$ and b = 4. For example, path 112 has states

(1 (modulo 2), (1 x 4 + 1) (modulo 8)) = (1, 5) and

(2 (modulo 2), (1 x 4^2 + 1 x 4 + 2) (modulo 8)) = (0, 6)

and therefore has the codeword (-0.87, 0.60).

It is not known how to find the best code given R, D, r, m. However, it can be shown that for a tree with b = R(D) branches per node, if the states are assigned real numbers independently at random with probability density $P(z) = \frac{1}{\sqrt{2\pi(1-D)}} \exp[-\frac{z^2}{2\sqrt{1-D}}]$, then with probability one in the limit of large r, m and large tree depth the resulting code is optimal in the following sense: the expectation over all source output sequences of the average distortion along the best path for each source sequence is arbitrarily close to D.

A question still remaining is how to search the tree efficiently to find good paths. Two algorithms for doing this will now be described.

Since $t(\underset{\sim}{s}^j) = (t(\underset{\sim}{s}^{j-1}) \times b + s_j)$ (modulo m), the state of a branch determines the states of all branches deriving from it. Consequently, branches at the same level with the same state are identical for coding purposes. Thus for example in the tree code of Figure 1,

State	Representation	State	Representation
(0,0)	-0.72	(1,0)	0.38
(0,1)	0.30	(1,1)	-0.69
(0,2)	1.38	(1,2)	-0.97
(0,3)	-0.32	(1,3)	0.76
(0,4)	1.32	(1,4)	1.32
(0,5)	-0.92	(1,5)	-0.87
(0,6)	0.60	(1,6)	0.37
(0,7)	-1.28	(1.7)	0.10

Table 1.

An example of a finite state tree code with period r = 2 and number of branch states m = 8.

paths 012, 032, 112, and 132 all have state (0,6) at level 2 and are therefore equivalent there. Thus for a memoryless source a choice from any set of paths in encoding a given source output should depend only on their distortions up until the time they reach the same state.

This property is used by an exhaustive search algorithm known as the Viterbi algorithm: encoder states are grouped into equivalence classes T_i defined by $T_i = \{t:bt = i \text{ (modulo m)}\}$, $i = 0,1,\ldots,m-1$. The algorithm proceeds by successive elimination and operates with all paths of the same length.

All one branch extensions of all paths still being considered are found and their distortions are computed. For each i, all paths ending in states in class T_i are compared and all but the one with the smallest total distortion are eliminated from further consideration. This process is repeated for each level until a given stopping level is reached. Then all remaining paths are compared and the one with the smallest total distortion is chosen to be the encoder output.

Another search algorithm, known as the stack encoding algorithm [4], operates as follows:

Let D* be the per letter distortion desired by the user. To be realistic (see the previously quoted results) we must have $R > R(D^*)$. Define a metric distortion function $d^*(z,\hat{z}) = d(z,\hat{z}) - (A + Bz^2)$ where $A + B = D^*$ are parameters to be adjusted. For example, a choice of metric matched to the limit of the performance of the best possible

codes would be $R = R(D^*)$, $A = D^*(1 - D^*)$, $B = (D^*)^2$. Then $\underset{\sim}{z}^i$ will be an acceptable approximation of a source sequence $\underset{\sim}{z}^i$ if and only if

$$\sum_{j=1}^{i} d^*(z_j, \hat{z}_j) \leq 0$$

(we assume that the code is indefinitely extensible, i. e., that the number of levels in the tree is practically infinite). Suppose the sequence $\underset{\sim}{z}^n$ (n large) was generated by the source, let $d^*(\underset{\sim}{s}^j)$ denote the metric relative to $\underset{\sim}{z}^n$ corresponding to the last branch of the path $\underset{\sim}{s}^j$ [e.g., $d^*(112) = d^*(z_2, 0.60)$ and $d^*(113) = d^*(z_2, -1.28)$ for the code of Figure 1], and let $D(\underset{\sim}{s}^j)$ be the cumulative metric along the path $\underset{\sim}{s}^j$. $D(\underset{\sim}{s}^j) = \sum_{i=1}^{j} d^*(\underset{\sim}{s}^i)$ where $\underset{\sim}{s}^i$ are the initial subsequences of length i of $\underset{\sim}{s}^j (i \leq j)$. The <u>stack</u> will contain different paths $\underset{\sim}{s}^j$ and their cumulative metrics $D(\underset{\sim}{s}^j)$, and will be arranged in ascending order of the latter (i.e., at the top of the stack there will be that path $\underset{\sim}{s}^j$ whose $D(\underset{\sim}{s}^j)$ is least).

1. At the beginning of the encoding process, the paths $0, 1, \ldots, b_o - 1$ are assigned zero cumulative distortion and arranged in the stack in any order (e.g., numerical order).

2. The encoder checks whether the path $\underset{\sim}{s}^j$ on top of the stack is such that j is greater than some stopping value. If so, go to step 4, if not, go to step 3.

3. The top entry $[\underset{\sim}{s}^j, D(\underset{\sim}{s}^j)]$ is eliminated from the stack, the branch metrics $d^*(\underset{\sim}{s}^j 0), d^*(\underset{\sim}{s}^j 1), \ldots, d^*(\underset{\sim}{s}^j (b - 1))$ are computed, and b

new entries $[\underset{\sim}{s}^j k,\ D(\underset{\sim}{s}^j k) = D(\underset{\sim}{s}^j) + d^*(\underset{\sim}{s}^j k)]$, $k = 0, 1, \ldots, b - 1$ are inserted in the proper locations in the stack. Go to 2.

4. The sequence $\underset{\sim}{z}^j$ is encoded into the codeword $\underset{\sim}{z}^j$ that corresponds to the path $\underset{\sim}{s}^j$. Stop.

2. Results

The basic algorithms were modified in several ways in the computer programs to simulate the encoding. A modification applying to both the Viterbi and stack algorithms was that data (source outputs) of magnitude greater than a certain cutoff c were encoded separately, using one quantization region for each tail of the Gaussian distribution.

The additional coding needed to code extreme data separately requires on the average rate $R_c = H\{\Phi(c) - \Phi(-c),\ 1 - \Phi(c),\ 1 - \Phi(c)\}$ where H is the entropy function defined by

$$H\{p_i\} = \sum_i (-\, p_i \log p_i).$$

Overall rate R is then

$$R = R_c + [\Phi(c) - \Phi(-c)]R_t$$

where R_t is the tree coding rate.

For D_c the expected distortion of the extreme source values and D_t the average distortion of tree coded source values, overall distortion D is given by

$$D = 2[1 - \Phi(c)]\, D_c + [\Phi(c) - \Phi(-c)]D_t\ .$$

It was determined experimentally that for both Viterbi and stack algorithms the cutoff c should be in the region of 3.5 to 4 source standard deviations.

The Viterbi algorithm with data cutoff 3.5 was simulated in IBM System 360 assembler language. It was run on 60 blocks of length 250 source outputs each, with period $r \geq 250$, that is, with branches of the code tree at different depths being assigned numbers independently. m was 16,384, b_o and b were 32. Overall rate R was thus about 5 bits per source output.

As given above, the lower limit of possible rate R versus distortion D performance is given by

$$R = -\frac{1}{2}\log_2 D \quad \text{or} \quad D = 2^{-2R}.$$

The Viterbi algorithm simulation just described was found to operate at an overall distortion $D = 1.31\,(2^{-2R})$. Because doing this required a search of about 16 thousand branches per datum encoded, the simulation could process only about 2 data per second.

Stack algorithm modifications were as follows:

(a) The branches coming out of a node were grouped together and put as a group into the stack according to the best cumulative distortion metric of the group. When the group arrives at the top of the stack its best branch is removed and extended and the group is re-entered in the stack according to the best cumulative distortion metric of the paths remaining in it.

(b) Whenever the stack contained more than 3,000 path groups, the group at the bottom of the stack (i.e., the group with the largest distortion metric) was eliminated from further consideration. This modification was required by the finiteness of the memory of the computer.

(c) Whenever step 3 of the stack algorithm was executed any multiple of 100,000 times, all path groups except the 32 deepest into the tree were eliminated from further consideration. This modification speeds search through the tree in the event that the encoding is taking too long.

The stack algorithm simulation was found to give performance of the same order of magnitude as did the Viterbi algorithm simulation. It was run on the same 60 blocks of data of length 250 each which the Viterbi algorithm used. Parameters were b = 32 branches per node, period r = 1, m = 2^{29} branch states, and b_o = 32 initial states. Thus overall rate was again about 5 bits per source output. Distortion metric parameters A, B given by the limit of performance of the best possible coding were found to give the most efficient results. That is, a D^* is chosen and A, B are set at $A = D^*(1-D^*)$, $B = (D^*)^2$. Varying D^* varies the distortion obtained and also the amount of search performed.

The stack algorithm simulation just described was found to give overall distortions of $D = 1.28\,(2^{-2R})$ and $D = 1.25\,(2^{-2R})$ with searches

of about 14 thousand and 23 thousand branches per datum respectively. It required about 7% longer to search each branch than required in the Viterbi algorithm.

References

1. C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," IRE National Convention Record, Part 4, pp. 142-163, 1959.

2. C. E. Shannon, "A Mathematical Theory of Communication," Bell System Tel. J., 27, p. 379, and p. 623, 1948.

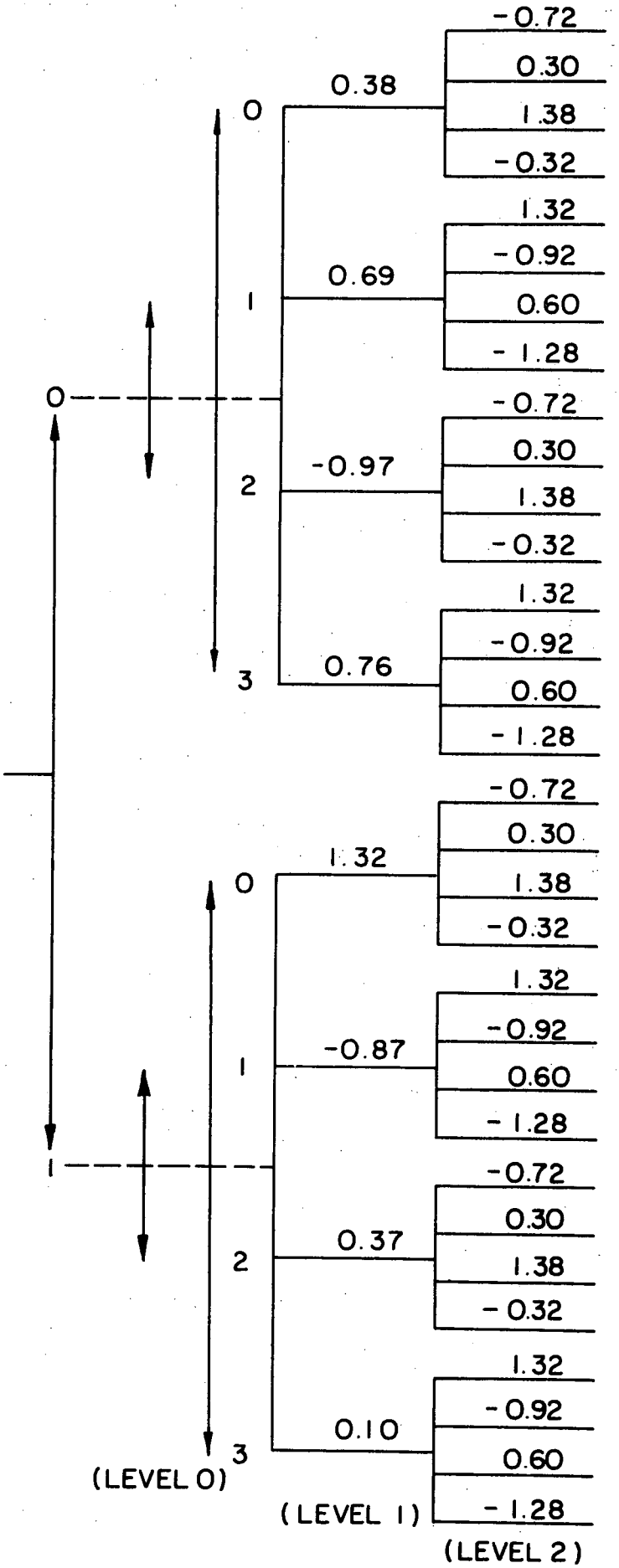
3. F. Jelinek, "Tree Encoding of Memoryless Time-Discrete Sources with a Fidelity Criterion," IEEE Transactions on Information Theory, IT-15, No. 5, Sept. 1969.

4. F. Jelinek, "A Fast Sequential Decoding Algorithm Utilizing a Stack," IBM Journal of Research and Development, 13, No. 6, Nov. 1969.

5. F. Jelinek, Probabilistic Information Theory, McGraw-Hill, New York, 1968.

Figure Caption

Fig. 1: Example of a partial coding tree of rate R = 2 for a Gaussian source with a square error fidelity criterion.


-0.72
0.30
1.38
-0.32
0.38
0
1.32
-0.92
0.60
-1.28
0.69
1
0
-0.72
0.30
1.38
-0.32
-0.97
2
1.32
-0.92
0.60
-1.28
0.76
3
-0.72
0.30
1.38
-0.32
1.32
0
1.32
-0.92
0.60
-1.28
-0.87
1
-0.72
0.30
1.38
-0.32
0.37
2
1.32
-0.92
0.60
-1.28
0.10
3
(LEVEL 0)
(LEVEL 1)
(LEVEL 2)

C.5

III-D Variable Length-to-Block Coding of Fixed Rate Sources

There are two practical problems associated with noiseless source coding: (a) optimal codes require a codebook table look-up, (b) real-time variable length coding and real-time decoding data retrieval are both subject to buffer overflow. A partial answer to problem (a) is Elias source coding as described in Appendix A of Jelinek: Probabilistic Information Theory. Problem (b) for block-to-variable length coding has also been analyzed there. It is, however, of interest to analyze the buffer over-flow problem of variable length-to-block coding that assigns constant length codewords to variable length source output sequences. (It is thus a generalization of run length coding.) The reason is the word-like character of computer storage that makes retrieval of constant length codewords much easier. In a paper to be published in IEEE Transactions on Information Theory (the abstract can be found below) Schneider and Jelinek derive tight bounds on buffer overflow probabilities. For binary sources that are more skew than (0.8, 0.2), variable length-to-block coding leads to lower probabilities of buffer overflow than does the usual block-to-variable length coding.

ON VARIABLE LENGTH-TO-BLOCK CODING*

by

K. Schneider,[1] Member IEEE

F. Jelinek,[2] Senior Member IEEE

ABSTRACT

Variable length-to-block codes are a generalization of run length codes. A coding theorem is first proven. When the codes are used to transmit information from fixed rate sources through fixed rate noiseless channels, buffer overflow results. The latter phenomenon is an important consideration in the retrieval of compressed data from storage. The probability of buffer overflow decreases exponentially with buffer length and we determine the relation between rate and exponent size for memoryless sources. We obtain codes that maximize the overflow exponent for any given transmission rate exceeding entropy, and present asymptotically optimal coding algorithms whose complexity grows linearly with codeword length. We compare error exponents corresponding to variable length-to-block, block-to-variable length, and block coding.

IV. PUBLICATIONS SUPPORTED BY THE CONTRACT

IV-A. Journal Articles by F. Jelinek

1. "Instrumentable Tree Encoding of Information Sources," IEEE Transactions on Information Theory, IT-17, January, 1971, (with J.B. Anderson).

2. "Channel Coding with a Sequential-Algebraic Hybrid Coding Scheme," IEEE Transactions on Communication Technology, COM-19, February, 1971, (with F.L. Huband).

3. "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels," Information and Control, 18, No. 3, pp. 261-298, April,1971 (with J. Cocke).

4. "Rate 1/2 Convolutional Codes with Complementary Generators," IEEE Transactions on Information Theory, IT-17, No. 6, November, 1971 (L.R. Bahl).

5. "On the Structure of Rate 1/n Convolutional Codes," IEEE Transactions on Information Theory, IT-18, No. 1, January, 1972.

6. "Permutation Codes for Sources," IEEE Transactions on Information Theory, IT-18, No. 1, January, 1972, (with T. Berger and J.K. Wolf).

7. "Variable Length Encoding of Fixed Rate Markov Sources for Fixed Rate Channels," accepted by IEEE Transactions on Information Theory, (with K. Schneider).

8. "On Variable Length-to-Block Coding," IEEE Transactions on Information Theory, accepted (with K. Schneider).

9. "A Two-Cycle Algorithm for Source Coding with a Fidelity Criterion," submitted to IEEE Transactions on Information Theory, (with J.B. Anderson).

IV-B. Conference Talks by F. Jelinek

1. Yale Conference on System Science, December, 1969, "The Tree Structure in Information Processing and Transmission."

2. Hawaii Conference on System Science, January 1970, (with J.B. Anderson), "Instrumentable Tree Encoding of Information Sources."

3. Nasa-wide Coded Communications Conference, JPL, February, 1970, "Adaptive Hybrid Sequential Decoding."

4. Southeastern Symposium on System Theory, Gainsville, Fla., March 1970, "An Algorithm for Encoding of Sources with a Fidelity Criterion."

5. International Symposium on Information Theory (with J. Cocke), June 1970, Noordwijk, The Netherlands, long paper, "Adaptive Hybrid Sequential Decoding".

6. International Symposium on Information Theory (with L.R. Bahl), June 1970, Noordwijk, The Netherlands, "A Class of Rate 1/2 Convolutional Codes".

7. Symposium on Picture Processing (with T. Berger and R. Dick), September 1970, North Carolina State University, "Tree Encoding of Analog Sources with Memory".

8. Princeton Conference on System Science (with K. Schneider), March 25, 1971, "Variable-to-Block Coding of Fixed Rate Memoryless Sources for Fixed Rate Channels."

9. Princeton Conference on System Science (with L. Bahl, C.D. Cullum, and W.D. Frazer), March 25, 1971, " "Algorithms for Computing the Free Distance of Convolutional Codes."

10. September 3, 1971, Second Soviet International Symposium on Information Theory, Tsakhadsor, Armenia, "On a Stack-Searched Branching Process."

11. IEEE Symposium on Information Theory (with J. B. Anderson), Feb. 1, 1972, Asilomar, California, long paper, "A Two-Cycle Algorithm for Source Coding with a Fidelity Criterion".

12. IEEE Symposium on Information Theory (with L.R. Bahl, J. Cocke, and J. Raviv), Feb. 3, 1972, Asilomar, California, "Optimal Decoding of Linear Block and Convolutional Codes for Minimizing Symbol Error Rate".

13. Also: talk of L.B. Hofman based on progress developed for this project: "Performance Results for a Hybrid Coding System," International Telemetering Conference, Washington, D.C., Sept. 30, 1971.

IV-C. Seminar Talks by F.Jelinek

1. Nov. 5, 1969, Dept. of Electrical Engineering, Mc Master University, Hamilton, Ontario, Canada, "New Approaches to Sequential Decoding."

2. Nov. 12, 1969, Division of Engineering, Brown University, Providence, R.I., "New Approaches to Sequential Decoding."

3. October 26, 1970, Jet Propulsion Laboratory, Pasadena, Calif., "On the Structure of Rate 1/n Convolutional Codes."

4. January 29, 1971, IBM-T.J. Watson Research Center, Yorktown Heights, New York; "Bootstrap Decoding Based on the Viterbi Algorithm."

5. April 1, 1971, Department of System Science, UCLA, Los Angeles, California; "A Two-cycle Algorithm for Tree Encoding of Sources with a Fidelity Criterion."

6. Oct. 20, 1971, Dept. of Electrical Engineering, Illinois Institute of Technology, "Bootstrap Sequential Decoding."

7. Dec. 6, 1971, Dept. of Electrical Engineering, University of Mass., "Survey of Sequential Decoding."

IV-D. Theses

1. J. B. Anderson, "Algorithms for Tree Source Coding with a Fidelity Criterion," Ph.D. Thesis, Cornell University, May 1972.

2. H.S. Park, "Bootstrap Hybrid Trellis Decoding," Ph.D. Thesis, Cornell University, 1972.

3. R.J. Dick, "Source Encoding of Gaussian Data," Ph.D. Thesis in preparation, Cornell University.