Performance of an Optimum Buffer Management Strategy for Sequential Decoding

J. W. Layland
Communications Systems Research Section

Sequential decoding has been found to be an efficient means of communicating at low undetected error rates from deep space probes, but a failure mechanism known as erasure or computational overflow remains a significant problem. The erasure of a block occurs when the decoder has not finished decoding that block at the time that it must be output.

A recent article developed a technique for scheduling the operations of a sequential decoder which has the potential for significant reduction in erasure probability relative to a decoder with the same parameters and using the conventional method of scheduling. Performance results reported previously depend upon the accuracy of an accepted model for the number of computations needed to decode a block of data. This article presents a reevaluation of decoder performance using actual sequential decoding data. Results are generally unchanged: a decoder with a 10-block buffer will achieve less than a 10^{-4} erasure probability with the new scheduling technique whenever a similar decoder had achieved less than a 10^{-2} erasure probability in conventional operation.

I. Introduction

A recent article (Ref. 1) described the characteristics of a buffer memory management strategy for sequential decoding which is capable of significantly reducing erasure probability for fixed values of speed advantage, buffer size, etc. Both the analytic work and the simulation results presented there relied upon a hypothesized theoretical distribution for the number of decoder computations needed to decode a fixed-length block of data. Subsequent work, described herein, has focused upon the behavior of the memory management algorithm with real data from a sequential decoder, and in determining the distribution of per-block computations for sequential decoding. Although the hypothesized distribution of decoding is not, in fact, completely correct, the results obtained previously for sequential decoding with feedback queuing memory management (FBQM) are essentially unchanged; i.e., for a decoder with a speed advantage of 10 and buffer size of 10 blocks operating with an erasure probability of 10^{-2}, use of the FBQM technique reduces the erasure probability to less than 10^{-4}.

II. The Distribution of Computations-per-Block for Sequential Decoding

It is well established that the number of computations c performed by a sequential decoder in incrementing by one the number of bits for which a local estimate has been made has a Pareto distribution, i.e.,

$$Pr\{c > N\} \sim k N^{-\alpha} \tag{1}$$

The exponent α is a positive increasing function of the bit signal-to-noise ratio E_b/N_0. The constant k has been estimated by Heller (Ref. 2, p. 41) to be 1.9.

The distribution of the number of computations C_L required to decode a block of length L has not been extensively studied, owing in large part to the amount of time required to decode enough blocks to obtain a statistically accurate estimate of this parameter. Since it is C_L which determines the performance of a decoder using FBQM, the performance evaluation of FBQM, and accurately determining the distribution of C_L are essentially the same problem. Previous work on determining the distribution of C_L has established that it is lower bounded by a linear function of L (Refs. 3 and 4); i.e.,

$$Pr\{C_L > N\} \gtrsim k L N^{-\alpha} \tag{2}$$

The parameters k and α are as before. Additional arguments have tended to show that Eq. (2) is both a lower bound and a reasonable approximation to the distribution of C_L. The simulations to be presented here, performed with block lengths of 256 and 1024 bits, show that the best first-order approximation to the distribution of C_L is instead

$$Pr\{C_L > N\} \sim k(N/L)^{-\alpha} \tag{3}$$

The parameters k and α are as before. A more accurate approximation with a wide range of applicability was used in the FBQM simulations, discussed in *Section IV*. This approximation is shown in the solid lines in Fig. 1.

The dotted lines in Fig. 1 represent an experimental distribution of computation-per-block for the decoding of blocks of length 1024 bits. Each line resulted from approximately 15 h of decoding on a Sigma 5, using a program which emulates the operations of the high-speed multi-mission sequential decoder (Ref. 5). The software simulation was used because of the existence of instrumentation within it to measure the desired parameter. Sample-sizes range from 3×10^2 blocks at the lowest signal-to-noise ratios (SNRs) up to 10^4 blocks at the highest. The list of the number of computations required to decode each block in the sample set was sorted to define the experimental distribution function. The dots appearing in Fig. 1 correspond to samples from this list spaced approximately uniformly in the computation variable.

At low signal-to-noise ratio, the experimental distribution is closely approximated by the simple Pareto distribution of Eq. (3) with $k \approx 3$ and α as defined from theory. This approximation is poor at high signal-to-noise ratios where the distribution exhibits two distinct behavior patterns dependent upon the range of the computation variable being considered. At relatively large values the distribution is roughly Paretian, as in Eq. (3), with approximately the correct exponent, but with a very small value of k. At small values of the computation variable, if the distribution is assumed to be Paretian, the apparent exponent is much higher than the per-bit computation exponent. This seeming anomaly can be explained on the basis of the following model for per-block computation. (A similar model was used in Ref. 4.) Let us divide the block into a number of sub-blocks, each one of which is entered and exited on the correct path, and suppose that the distribution of computations for each sub-block is given by Eq. (3), with L replaced by L_s, the sub-block length. Both the exponent α and the lengths of the sub-blocks L_s depend on the signal-to-noise ratio. Each sub-block is terminated by a sequence of bits for which no searching is required; for which the initial best-bit estimate is correct; and which is never changed by back-tracking. As such, they represent independent regions of decoding computation. The number of computations needed to decode a block is thus the sum of several independent variables: the number of computations needed to decode each of the component sub-blocks. At low signal-to-noise ratios, the sub-block length L_s is large—so large, in fact, that there is only one sub-block in the complete block, and the computation distribution for the entire block is well approximated by Eq. (3). At higher signal-to-noise ratios, the sub-block length decreases, so that several sub-blocks exist within a block of 1024 bits. When a number of independent Pareto random variables are added, large values of the sum occur most likely as a result of one large component value. Contrarily, small values would arise as sums of small values on each of the sub-blocks. The result is that for small values of the computation variable, the distribution falls off rapidly; more in the manner of a gaussian distribution than Pareto, as indeed the true distribution does.

If the model just proposed is valid, the distribution of computation for other block lengths will differ in a specific fashion from the distribution in Fig. 1. For shorter block lengths, the SNR above which the two-part distribution exists is higher than the SNR threshold for the same effect with 1024 bit blocks, because the typical sub-block length must be less than the block length for the averaging of shorter computations to take place. This appears to have happened in Fig. 2, which shows the experimental computation distribution for a block length of 256 bits. The averaging effect has only begun to be apparent at $E_b/N_0 = +3.5$ dB with the shorter block length, whereas the appearance is at about 2.25 dB with 1024 bits/block.

The solid lines in Fig. 1 represent a constructive approximately least-square error fit to the experimental distributions on the same figure. Since the primary purpose in determining an analytical formulation for the distribution was to generate pseudo-random variables with that distribution, the function fitted determines the computation variable from the probability value, rather than vice versa. The computation variable is segmented into two parts, $C_L = C_1 + C_2$. Each part, C_i, has a distribution given by

$$Pr\{C_i > N_0\} = 1/(1 + k_i(R)(N/L)^{\alpha_i(R)}) \tag{4}$$

The values, $k_i(R)$ and $\alpha_i(R)$, are slowly varying functions of the bit SNR, R(in dB), and L is the block length = 1024. The C_i are treated as if completely dependent random variables, so that determining either determines the other, and hence their sum. The constants have been computed so that, in general, C_1 fits well to the computation distribution for large values of the computation variable; C_2 fits the distribution for small values; and their sum fits quite well over the entire range. The parameter values which resulted from a computer-aided search are

$$\left.\begin{aligned} k_1 &= 0.1069 + 0.7651 \cdot \exp\,[2.98 \cdot (E_s/N_0)] \\ \alpha_1 &= 1.327 + 0.583 \cdot (E_s/N_0) \\ k_2 &= 1.07 \cdot 10^{-6} + 2.84 \cdot 10^{-6} \cdot \exp\,[3.776 \cdot (E_s/N_0)] \\ \alpha_2 &= 13.1 + 6.67 \cdot (E_s/N_0) \end{aligned}\right\} \tag{5}$$

for the 1024 bit blocks. No attempt has been made to fit a similar distribution to the experimental data for the 256 bit blocks. The intent here was not to exhaustively study the computation behavior of sequential decoding, but to assist in the evaluation of the performance of sequential decoding with FBQM.

III. Decoding With a Magic Genie

Two hypothetical magic genie-aided decoders were analyzed in Ref. 1. Before proceeding further, we will present revised versions of those analyses using as a basis the first-order approximation to the computation distribution in Eq. (3). Both decoders are able to perform μ computations during the time one bit is being received and have an infinitely large buffer to hold data waiting to be decoded.

The magic genie which aids the first decoder is benevolent and informs the decoder before starting decoding each block whether it should or should not decode that block in order to minimize the fraction of blocks erased. That is, the genie identifies to the decoder those blocks for which the total computation requirement is above a threshold, T_0. On the average, the number of computations available to decode each block is μ^*L while the number actually expended is

$$\int_0^{T_0} N\, dP(N)$$

where $P(N) = Pr\{C_L > N\}$. Since the number of computations expended cannot exceed the supply, and the decoder is doing as well as possible, T_0 is the largest value for which the following relation holds.

$$\int_0^{T_0} N\, dP(N) \leq \mu^*L \tag{6}$$

The parameter of interest is $P(T_0)$, the fraction of blocks erased by this decoder. Using Eq. (3) for $P(N)$, we find

$$\left.\begin{aligned} p(T_0) &= \exp\,(-\mu/k) && \alpha = 1 \\ P(T_0) &= \left[1 + \mu k^{-\frac{1}{\alpha}}\left(\frac{1-\alpha}{\alpha}\right)\right]^{-\frac{\alpha}{1-\alpha}} && \alpha \neq 1 \end{aligned}\right\} \tag{7}$$

For $\alpha > 1$, $P(T_0)$ can be made zero by making μ sufficiently large.

The magic genie which aids the second decoder is whimsical and only informs the decoder that it should erase a block after it has expended as much computational work toward decoding an erased block as in decoding the most difficult block which does get decoded. In this case all blocks requiring more than T_1 units of computation are erased after having received T_1 units. Using the requirements that computation ex-

pended cannot exceed the supply, T_1 is the largest value for which the following inequality holds.

$$T_1 P(T_1) + \int_0^{T_1} N dP(N) \leq \mu^* L \tag{8}$$

The erasure probability is $P(T_1)$.

$$\left.\begin{aligned} P(T_1) &= \exp\left[-\{(\mu/k) - 1\}\right] && \alpha = 1 \\ P(T_1) &= [\mu k^{-1/\alpha}(1-\alpha) + \alpha]^{-\alpha/(1-\alpha)} && \alpha \neq 1 \end{aligned}\right\} \tag{9}$$

For $\alpha > 1$, $P(T_1)$ can be made arbitrarily small by making μ sufficiently large.

IV. Simulated Performance of a Practical Decoder

A practical decoder which performs similarly to the hypothetical decoder aided by the whimsical genie can be developed by treating the problem of allocating the decoder's efforts as time-sharing processor allocation problem (Ref. 6). We observe that for a Pareto distribution of computation, the block which is most likely to be completely decoded by applying to it Δc computations is that block which has received the least amount of computation among all of those undecoded blocks in the buffer.

Decoder operation is as follows: The decoder begins decoding on each block immediately after it is received. If it is not completely decoded by the time the next block becomes available, it is labeled as having received $\mu^* L$ computations and is stored in the buffer. If decoding is completed before the next block is available, the block in storage which has received the least amount of computation is brought out and decoding performed upon it until either the next new block is available, or until this block too has been decoded. A flow-chart of this process was given in Ref. 1, Fig. 1.

This memory management scheme was recently simulated in two ways using the decoding data set discussed in *Section II*. Initially, this data was fed directly into the memory management model with various values for speed advantage and buffer size. Since the sample-sizes obtained from the decoder simulations were too small to adequately mask the "end-effects" due to the starting and stopping of the memory management process, another sequence of simulations were performed using pseudo-random data generated to conform to the distribution families described by Eq. (4) and (5). The result of this latter simulation is the family of performance curves in Fig. 3. At all values of speed advantage considered, FBQM provides a considerable improvement over the linear buffer strategy at erasure rates of 10^{-2} and below.

The curves in Fig. 4 represent a comparison between performance data obtained using pseudo-random data with the distribution of Eqs. (4) and (5), and using the actual sequential decoding data. The solid lines represent 10^5 blocks of pseudo-random data at $E_b/N_0 = 3.0$ dB. The circles represent approximately 5×10^3 blocks of simulated sequential decoder data at $E_b/N_0 = 3.0$ dB. Most of the difference can be explained on the basis of the small sample size: Either buffer management scheme is capable of abandoning a buffer-full (in this case 10 blocks) of data when the simulation is abruptly terminated, and the FBQM process has a relatively high probability of doing so. A partial verification of this assertion can be found in the triangular marks in Fig. 4 which represent the result of treating the 5×10^3 actual decoding samples as a finite population from which 10^5 samples are drawn with replacement. Even with this device for extending the apparent sample size, the tails of the distribution are only represented as well as is done by the smaller sample size. However, if the fitted distribution function has been built correctly, the tails should be accurately represented at any sample size, and hence the performance curves of Fig. 3 more accurately reflect the performance to be achieved in practice than performance curves derived directly from the samples of the sequential decoder computation variable.

Another view of the performance difference between the FBQM process and linear buffer management can be gained by considering the SNR required for a decoder with a given speed advantage to achieve some fixed erasure rate, e.g., 5×10^{-3} or 5×10^{-4}. This is shown in Fig. 5. A performance advantage in excess of 0.5 dB is evident for a wide range of speed advantage.

V. Time-Sequence Un-Scrambling

It should be apparent from the behavior of the FBQM algorithm that it possesses one drawback which could seriously detract from its usefulness. That is, that decoded blocks do not emanate from the decoder in strict time-sequence but are reordered and delayed an amount which depends upon the difficulty encountered in decoding. If the data is spacecraft video, the reordering should pose little difficulty, as it would be a relatively small increment to the already extensive picture processing. Other potential users might well be deterred by the

reordering, so it is worthwhile to consider a combination FBQM decoder and unscrambler which emits decoded data blocks in the same order as the corresponding undecoded data is received.

Suppose a decoder has been implemented using FBQM which performs with essentially zero erasure probability, and suppose that the output of this machine is the input to a buffering machine capable of holding N blocks of decoded data. An erasure occurs in this combination whenever the buffer is filled with blocks $m,m+1,\cdots,$ $m+k$, and block $m-1$ has not yet been output from the decoder. Let us assume that the decoder and unscrambler buffers were empty when block $m-1$ was received, and that blocks $m,m+1,\cdots,m+N-1$ required negligible effort to decode. Under these circumstances, the probability that block $m-1$ is erased is the probability that the number of computations needed to decode block $m-1$ exceeds $N*\mu*L$, the number of computations that can be performed while blocks $m,m+1,\cdots,m+N-1$ are being received. This is, in effect, a lower bound on the probability that block $m-1$ is erased. Using Eq. (3):

$$P_d^u \lesssim k(N*\mu)^{-\alpha} \tag{10}$$

Suppose instead of the FBQM decoder/unscrambler combination we substitute a decoder with linear buffer management which has the same number of bits of storage as the unscrambler had. If the code rate is $1/\nu$ and the received symbols are quantized to 3-bits each, that memory is now able to store $N/(3*\nu)$ blocks of data. Assuming that this buffer is empty when block $m-1$ is received, the probability of erasing it is the probability that it requires more than $N*\mu*L/(3*\nu)$ computations.

$$P_d^l \lesssim k\left(N*\mu/(3*\nu)\right)^{-\alpha} \tag{11}$$

The assumptions in both cases are approximately equivalent: both saying that long computations are isolated events.

It remains to estimate or eliminate the buffer in the FBQM decoder itself. If it is negligible, the FBQM decoder/unscrambler pair will be equivalent to a normal sequential decoder with $3*\nu$ times as much memory! We can, in fact, force a dynamic trade-off between storage utilized by the FBQM decoder and storage utilized by the unscrambler, by fixing their total size in bits, and erasing the oldest block or blocks whenever needed storage for an incoming block is not available. The memory, e.g., would be capable of holding 10 blocks of raw partially decoded data, or 70 blocks of fully decoded data, or any integer solution in between. This is a relatively simple modification to the FBQM algorithm, requiring the addition of an information chain which identifies the age of each block in storage, and the storage of decoded data from completely decoded blocks until the undecoded blocks which are older are either decoded or erased. This modified FBQM algorithm has been simulated using pseudo-random data for several SNR values. The results for $E_b/N_0 = 3.0$ dB are shown in Fig. 6 where they are compared to the full FBQM and to a linearly buffered decoder. With the rate 1/2 code simulated, the FBQM/unscrambler combination is equivalent to a linearly managed decoder with approximately a threefold increase in buffer-size at higher values of μ. Since a sixfold increase is conceptually possible, a different algorithm could perhaps be found for decoder scheduling which would show significant improvement over the FBQM/unscrambler combination, and still satisfy the time-sequence constraint.

VI. Summary

This article has presented the results of a sequence of simulations intended to evaluate the performance of a sequential decoder employing the FBQM memory management algorithm. A model for the number of computations required to decode a fixed-length block of data was also developed to aid in the evaluation. Performance curves for the FBQM decoder have been presented which show that significant performance improvements are possible: i.e., reduction by 0.5 dB in the value of E_b/N_0 required to achieve an erasure rate of 10^{-3} or a reduction in erasure rate of from 10^{-2} to 10^{-4} by the addition of the feedback queuing memory management strategy, with all other decoding parameters held constant.

A brief discussion was also presented of the problems associated with the reordering into strict time-sequence of the FBQM decoder output data. It was shown that even if this unscrambler must share memory with the FBQM decoder itself, a performance improvement equivalent to a better than threefold increase in buffer size is possible, relative to a strictly linear-management decoder.

For JPL's Deep Space Network, however, a preferable solution would seem to be a FBQM decoder in the DSSs cascaded with a separate unscrambler in the net-

work control center. The data rate out of the FBQM decoder is as low as can be achieved—the true data rate from the spacecraft, and for a fixed dollar outlay, a considerably larger descrambler buffer could be acquired for the central facility than could be deployed at each of the affected DSSs.

References

1. Layland, J. W., "An Optimum Buffer Management Strategy for Sequential Decoding," in *The Deep Space Network Progress Report,* Technical Report 32-1526, Vol. VI, pp. 106–111. Jet Propulsion Laboratory, Pasadena, Calif., Dec. 15, 1971.

2. Heller, J. A., "Decoding and Synchronization Research: Description and Operation of a Sequential Decoder Simulation Program," in *Supporting Research and Advanced Development,* Space Programs Summary 37-58, Vol. III, p. 42. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 31, 1969.

3. Savage, J. E., "The Distribution of Sequential Decoding Computation Time," *IEEE Trans. Inform. Theory,* Vol. IT-11, pp. 143–147, April 1966.

4. Jacobs, I. M., and Berlekamp, E. R., "A Lower Bound to the Distribution of Computation for Sequential Decoding," *IEEE Trans. Inform. Theory,* Vol. IT-13, pp. 167–174, April 1967. (See also Jacobs, I. M., and Berlekamp, E., "A Lower Bound to the Distribution of Computation for Sequential Decoding," in *Supporting Research and Advanced Development,* Space Programs Summary 37-34, Vol. IV, pp. 270–276. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 31, 1965.)

5. Lushbaugh, W. A., and Layland, J. W., "A Flexible High-Speed Sequential Decoder for Deep Space Channels," *IEEE Trans. Commun. Tech.,* Vol. COM-19, pp. 813–820, Oct. '71. (See also Lushbaugh, W. A., and Layland, J. W., "System Design of a Sequential Decoding Machine," in *The Deep Space Network,* Space Programs Summary 37-50, Vol. II, pp. 71–78. Jet Propulsion Laboratory, Pasadena, Calif.)

6. Kleinrock, L., "A Continuum of Time-Sharing Scheduling Algorithms," in *AFIPS Conference Proceedings,* Vol. 36, 1970, pp. 453–458, Spring Joint Computer Conference, Atlantic City, N. J., 1970.

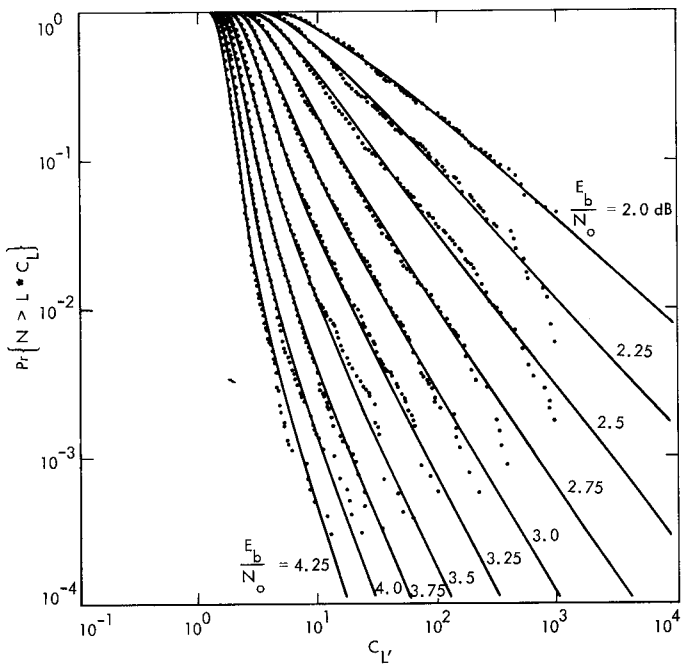

Fig. 1. Distribution of per-block computation with 1024 bits/block

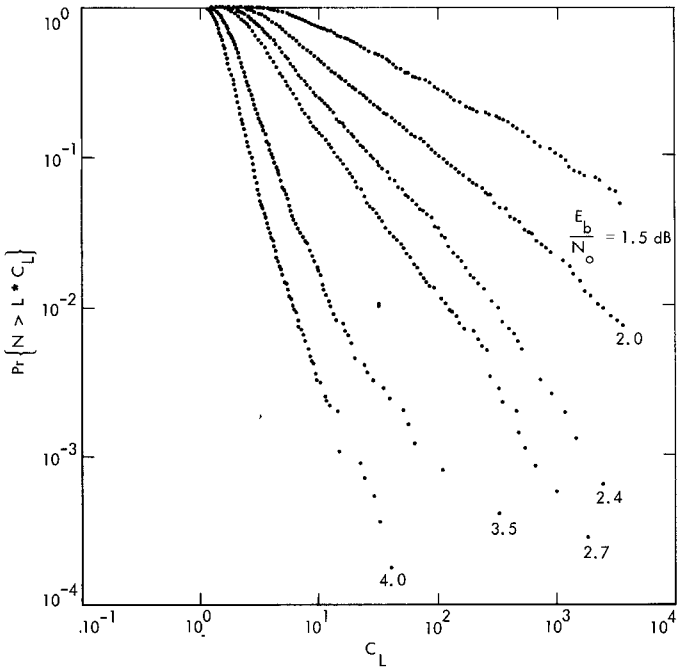

Fig. 2. Distribution of per-block computation with 256 bits/block

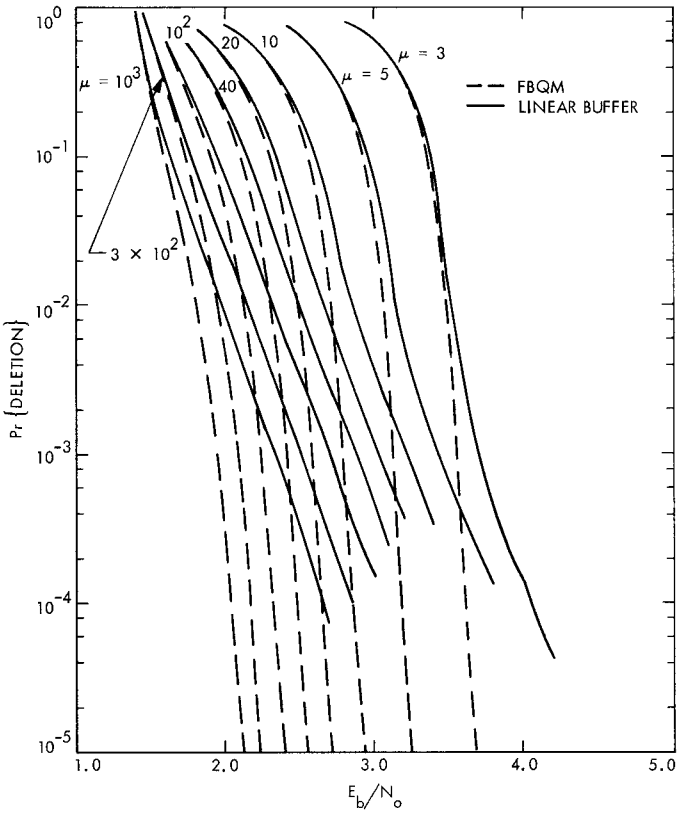

Fig. 3. Comparison of erasure performance for sequential decoding with FBQM and linear buffer management

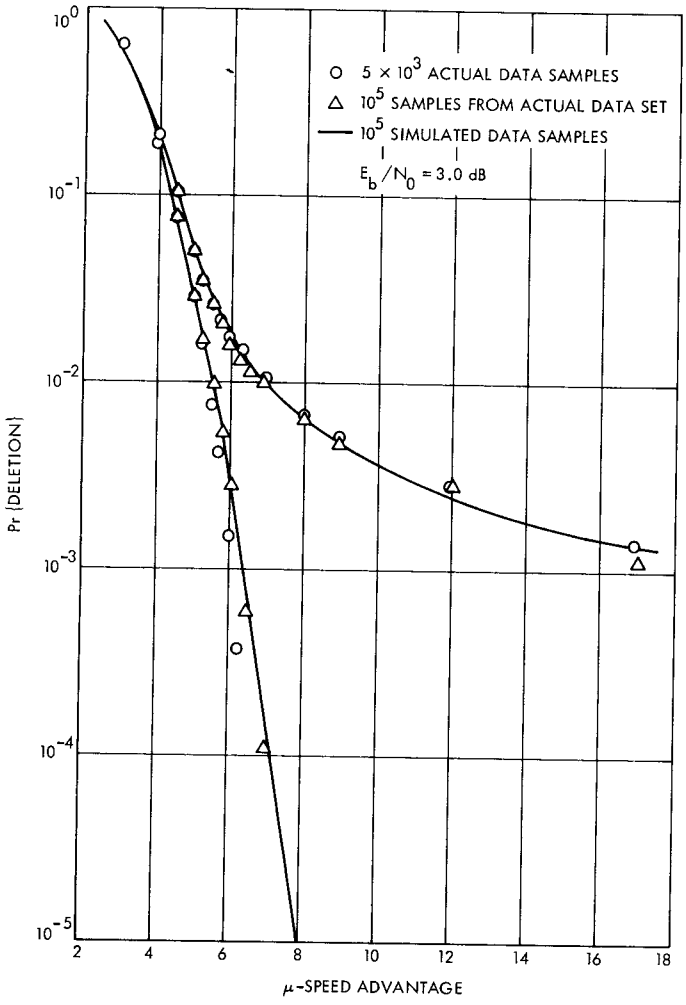

Fig. 4. Comparison between simulation and actual data

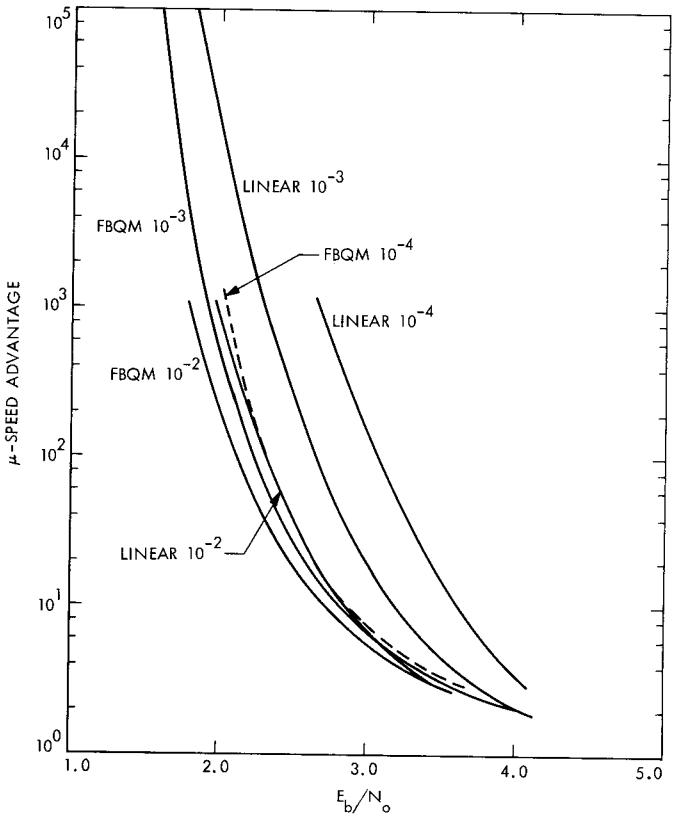

Fig. 5. Signal-to-noise ratio required to achieve desired erasure probability for varying decoder speed

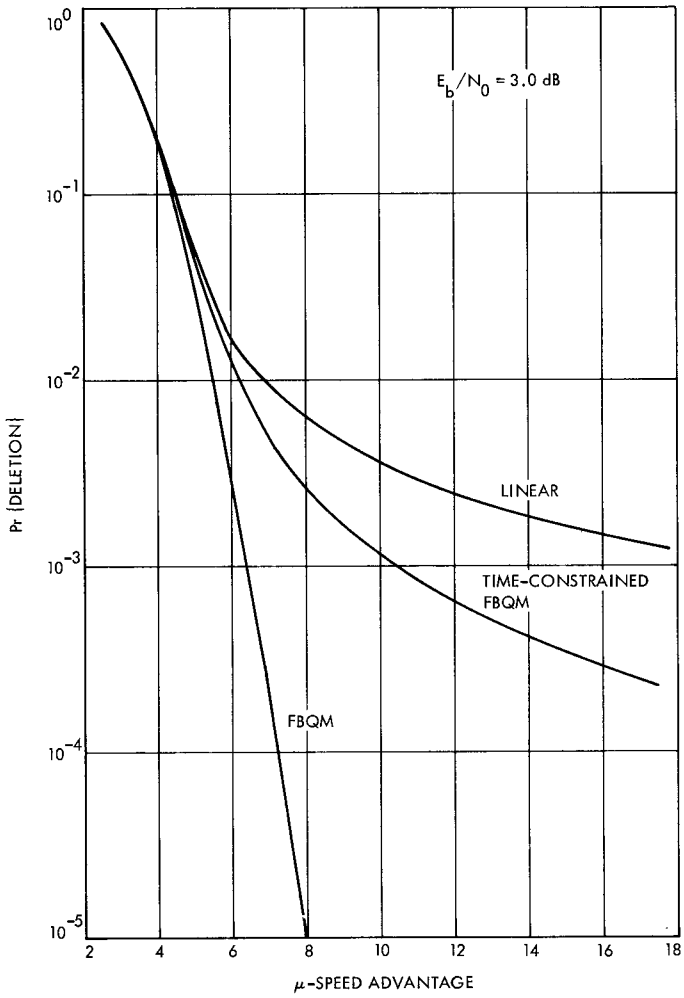

Fig. 6. Performance of FBQM as modified to observe a time-ordering constraint on emitted data