[51] A. Yano and T. Matsumoto, "Projection-type image holography and its information quantity," in *Proc. 34th Fall Meeting Japan Soc. Appl. Phys.*, p. 94, Oct. 1973.
[52] T. Matsumoto and A. Yano, "Tricolor holographic stereogram," in *Proc. 34th Fall Meeting Japan Soc. Appl. Phys.* p. 95, Oct. 1973.
[53] T. Okoshi, K. Aikawa, and K. Oshima, "Projection-type image holography using an anamorphic lens system," in *Proc. 6th Annu. Conf. Imaging Techniques* (Tokyo, Japan), Nov. 1975.
[54] T. Okoshi, "Recent progress in projection-type holography," in *Proc. 1976 4th Int. Optical Computing Conf.* (Capri, Italy), Aug. 31-Sept. 2, 1976.
[55] ——, "Projection-type holographic displays," in *Proc. 1977 SPIE Annu. Tech. Conf.* (San Diego, CA), seminar 10, no 120-13, pp. Tech. Digest, pp. 102-108, Aug. 1977.
[56] D. Gabor, "Three-dimensional picture projection," U.S. Patent 3 479 111 (filed Aug. 24, 1967; patented Nov. 18, 1969).
[57] V. G. Komar, "Progress on the holographic movie process in USSR," in 1977 *SPIE Annu. Tech. Conf.* (San Diego, CA) seminar 10, no. 120-17, Tech. Digest, pp. 127-144, Aug. 1977.
[58] See [1, section 4.2, and table 4.1].
[59] E. N. Leith, J. Upatnieks, B. P. Hildebrand, and K. Haines, "Requirements for wavefront-reconstruction television facsimile systems," *J. Soc. Motion Pict. Telev. Eng.*, vol. 74, no. 10, pp. 893-896, Oct. 1965.
[60] F. T. S. Yu, "Information channel capacity of a photographic film," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 477-480, July 1970.
[61] N. J. Bershad, "Resolution, optical-channel capacity and information theory," *J. Opt. Soc. Amer.*, vol. 59, no. 2, pp. 157-163, Feb. 1969.
[62] L. H. Lin, "A method of hologram information reduction by spatial frequency sampling," *Appl. Opt.*, vol. 7, no. 3, pp. 545-548, Mar. 1968.
[63] K. Oshima and T. Okoshi, "Synthesis of an autostereoscopic three-dimensional image from binocular stereoscopic images," *Appl. Opt.*, vol. 18, no. 4, pp. 469-476, Feb. 1979.
[64] A. C. Traub, "Stereoscopic display using varifocal mirror oscillations," *Appl. Opt.*, vol. 6, no. 6, pp. 1085-1087, June 1967.
[65] E. G. Rawson, "Three-dimensional computer-generated movies using a varifocal mirror," *Appl. Opt.*, vol. 7, no. 8, pp. 1505-1511, Aug. 1968.
[66] M. C. King and D. H. Berry, "Varifocal mirror technique for video transmission of three-dimensional images," *Appl. Opt.*, vol. 9, no. 9, pp. 2035-2039, Sept. 1970.
[67] J. Hamasaki, Y. Nagata, H. Higuchi, and M. Okada, "Real-time transmission of a 3-D image using volume scanning and spatial modulation," *Appl. Opt.* vol. 16, no. 6, pp. 1675-1685, June 1977.
[68] H. J. Gerritsen and B. Horwitz, "Experimental demonstration of optical sectioning by spatial filtering and its possible use in transmission of three-dimensional pictures," *Appl. Opt.*, vol. 10, no. 4, pp. 862-867, Apr. 1971.
[69] H. Higuchi, J. Hamasaki, and M. Okada, "Real-time optical sectioning having high SNR by using frequency interleaving," *Appl. Opt.*, vol. 16, no. 7, pp. 1777-1779, July 1977.
[70] H. Higuchi and J. Hamasaki, "Real-time transmission of three-dimensional images formed by parallax panoramagrams," *Appl. Opt.*, vol. 17, no. 24, pp. 3895-3902, Dec. 1978.
[71] T. Okoshi, "Video signal displays," in *Electronic Imaging*. New York, Academic Press, 1979 (Proc. Int. Symp. Electronic Imaging, Sept. 11-13, 1978, London).

The Technology of Error-Correcting Codes

ELWYN R. BERLEKAMP, FELLOW, IEEE

Invited Paper

Abstract—This paper is a survey of error-correcting codes, with emphasis on the costs of encoders and decoders, and the relationship of these costs to various important system parameters such as speed and delay.

Following an introductory overview, the remainder of this paper is divided into three sections corresponding to the three major types of channel noise: white Gaussian noise, interference, and digital errors of the sort which occur in secondary memories such as disks. Appendix A presents some of the more important facts about modern implementations of decoders for long high-rate Reed-Solomon codes, which play an important role throughout the paper. Appendix B investigates some important aspects of the tradeoffs between error correction and error detection.

Manuscript received June 25, 1979; revised January 11, 1980.
The author is with Cyclotomics, Inc., 2140 Shattuck Ave., Berkeley, CA 94704.

I. Introductory Overview of Coding Technology in 1980

COMMUNICATION links transmit information from here to there. Computer memories transmit information from now to then. In either case, noise causes the received data to differ slightly from the original data. As Shannon [1] showed in 1948, the noise need not cause any degradation in reliability. The noise does impose some limiting capacity on the throughput rate, although that limit is typically well above the throughput rate at which real systems operate. Error-correcting codes enable a system to achieve a high degree of reliability despite the presence of noise. In addition to the data bits one wishes to transmit, one also transmits some additional redundant check bits. Even though the noise causes

0018-9219/80/0500-0564$00.75

some errors in both the transmitted data bits and the transmitted check bits, there is usually still enough information available to the receiver to allow a sophisticated decoder to correct all of the errors unless the noise is extremely severe. On many interesting channels only a modest amount of redundancy is needed to ensure that the probability of decoding error is negligibly small.

In the 1950's and 1960's, the costs of sophisticated coding systems were measured in terms of the relatively large amount of digital hardware needed to implement encoders and decoders. In some communication systems, the simplest alternative to coding is increased signal power. However, that has always been expensive, and the cost of energy is inflating. A few years ago, the rule-of-thumb figure for the costs of power on certain satellite communication links was one million dollars per decibel. In space communications the costs can be much higher. Not surprisingly, space communication links were among the first to employ coding.

Since 1960 the costs of digital electronics have been decreasing at a phenomenal rate. The relative functional value of a single integrated circuit (IC) has been doubling every 14 to 16 months, while the corresponding prices increase very slowly, even when measured in deflating dollar value. Almost anything that was built on a single card of hardware a decade ago is now available in a single IC, and it is widely believed that this trend is destined to continue. In addition, there have also been significant improvements in decoding algorithms and in the design of special-purpose digital hardware to implement them. Meanwhile, the costs of analog components such as power, antennas, etc., have either been increasing or decreasing at a much slower rate. Hence sophisticated coding systems have become increasingly attractive from an economic viewpoint, and this trend seems destined to continue.

Several types of decoders have already been manufactured in relatively large volumes. Various models of Viterbi decoders have been used in various communication links. Much larger numbers of block decoders have been manufactured and sold with various computer memory systems. Most such decoders are elementary devices which correct one bit per block or one burst per block. Limited numbers of many other kinds of decoders have also been built, and some of them now appear firmly ensconced in markets with very large future potential. The Joint Tactical Information Distribution System (JTIDS) program has adopted the Reed-Solomon (RS) (31, 15) code, which may now be destined to become standard for all tactical military communciation links of the U.S. Armed Forces, and possibly several or all of the NATO countries as well.

Most of the coding and decoding systems in use today are minor refinements of designs which were first introduced a decade or more ago. Many of those designs were based on technological assumptions that are no longer valid. For example, in one of the most influential papers ever written on coding technology, R. T. Chien [2] introduced a very clever technique for correcting single bursts quickly. The obvious next step is the correction of multiple bursts of errors. But in the technology of the 1960's, this option was too expensive; as Chien wrote, "In many systems use of powerful codes such as suggested by . . . Reed and Solomon may be ruled out on the basis of economics." One major goal of the present papers is to convince the reader that this quotation is now obsolete. Since 1975, it has been quite practical to build decoders which correct multiple bursts of errors, on line, and at high speeds. The amount of hardware needed to implement such decoders is already rather modest in comparison to the amount of hardware now used in adjacent parts of the system for purposes such as timing, synchronizing, deskewing, buffering, and controlling various routing and protocol requirements.

The cost of manufacturing sophisticated high-performance decoders is now dwarfed by other considerations. In order to attain the maximum benefit from sophisticated decoders, it is typically necessary to modify or redesign adjacent parts of the system. In some cases this may force the new system to be incompatible with previous systems. Furthermore, the Galois field arithmetic and the novel architectures used in the best modern decoders are quite unconventional. Following unsuccessful efforts to design efficient modern decoders in-house, some organizations have opted for considerably more expensive designs with substantially weaker performance specifications primarily because the latter could be comprehended, developed, and maintained by people already on their payrolls.

Accurate cost/performance comparisons of different coding systems are not easy to obtain. In addition to the usual management problem of giving appropriate weights to design and development costs, manufacturing costs, sales costs, and maintenance costs, one must also face the engineering problem of modeling the noise environment in which the coding system is to be embedded. Many studies which have attempted to predict performance of various proposed coding systems have ended with erroneous and misleading conclusions. The number of serious mistakes is a large fraction of the number of such studies. One common error is to accept an overly simplistic model of the noise environment. Another common error is to accept as "state-of-the-art" cost/performance figures based on poor decoder designs. It is not uncommon to find different designs of the *same* decoder differing in cost/performance by a factor of ten or more. The major factor limiting the spread of coding technology in the 1970's has been ignorance rather then cost. This paper is a small attempt to dispel some of that ignorance.

Table I presents a summary of most of the major coding options now available on a few small cards of digital electronics.

The total gate count of most of these decoders is somewhat less than some existing IC's, such as the 64K RAM. Hence, with sufficient investment, it would be possible to develop almost any of these decoders as a single IC within 3 or 4 years. However, the market now appears to be divided among many different types of requirements, and so it might easily be more than 4 years before sophisticated single IC decoders become commonplace.

Each of the coding schemes listed in Table I includes a wide range of possible codes which can be obtained by suitable choices of the parameters which distinguish the various members of each family from each other. For many choices, one or two medium-sized cards are sufficient. There are also a large number of *ad hoc* coding schemes which have been custom designed to various applications. These schemes are so numerous, and so specialized, that they have not been included in Table I, even though some of them, such as the IBM magnetic tape codes, are now in widespread use.

The coding systems listed in Table I are broadly classified into three sets, depending on the amount of analog soft-decision information which the decoder uses. For transmitting information at slow speeds against white Gaussian noise, the most effective coding schemes utilize soft-decision real-number analog information obtained by correlating the received signal with each of the possible transmitted signals. For transmitting information against strong interference, only one bit of soft-decision information is meaningful. This bit, called the

TABLE I
SUMMARY OF MAJOR CODING OPTIONS AVAILABLE ON A FEW CARDS

VERSUS DIGITAL ERROR

Coding Scheme	Type of Noise It Handles Best	Recommended Applications
≥3RS (interleaved)	Multiple very long bursts	?
2RS	Multiple long bursts	Tapes
RS	Multiple short bursts	Disks
Fire-like	Single bursts	Disks with compatibility requirements
≥3BCH	None at high rates; it is uniformly inferior to 2RS	
2BCH	Multiple bursts of length 2	?
BCH	Multiple independent bit errors	Large semiconductor memories
Hamming-like	Single bits	Small semiconductor memories

VERSUS INTERFERENCE

Coding Scheme	Bandwidth Requirements	Power Requirements	Performance vs. Strong RFI	Maximum Speed
RS (Orthogonal)	Modest	Low	Excellent	Moderate
n-Viterbi (interleaved)	Low	Modest	Good	Moderate
Vector Crunchers	Low	Modest	Excellent	Slow (32 kHz)
Chase + BCH	Low	Modest	Good+	Slow

VERSUS WHITE GAUSSIAN NOISE

Coding Scheme	Bandwidth Requirements	Power Requirements	Typical Performance Specifications	Maximum Speed
Orthogonal Codes	Very Large	Very Low	10^{-1} to 10^{-13}	Slow
Concatenated RS (Orthogonal)	Modest (10 to 1)	Low	10^{-5} to 10^{-13}	Moderate
Short Binary, e.g., (18,9,6)	Low (2.5 to 1)	Modest	10^{-1} to 10^{-3}	Fast (40 MHz)
Concatenated RS (Short Binary)	Low	Modest	10^{-5} to 10^{-13}	Fast
Viterbi Decoders	Low	Modest	10^{-3} to 10^{-6}	Moderate
Concatenated RS (Viterbi)	Low	Modest	10^{-5} to 10^{-13}	Moderate

"erasure indicator" distinguishes between a high level of interference (when the signal is not detectable) and the absence of interference (when the signal-to-noise level is high). Finally, there are systems (such as semiconductor memories) in which the noise is intrinsically digital, because it is uneconomical or even impossible to obtain any additional soft-decision information about the validity of any particular bit.

In practice, the borderlines between these three types of noise environments are fuzzy. Most errors in magnetic memories arise not because the signal is wrong, but because it is very weak and this could, in principal, be detected by analog means. Memory systems rarely do this, however, partly because the highly non-Gaussian nature of the noise makes such information less useful than it is on the deep space communication channel, and partly because the relatively higher speeds on the disk channel make the acquisition of analog information more difficult. Similarly, there are some very high-speed communication links on which the acquisition of good analog soft-decision information is sufficiently difficult that the system designers have opted instead to use more power, hard decisions, and an error-correcting code which is well matched to the hard-decision environment.

The other sections of these notes consider the performance of codes in each of the three major types of noise environments. Additional sections discuss the amounts of hardware required for various modern decoders and how a small amount of additional redundancy can allow the coding engineer to convert most "catastrophic" errors into "detected" errors in systems which have a high reliability requirement.

II. Coding Versus White Gaussian Noise

A. Historical Review and Perspective

Shannon [3] inaugurated the study of the performance of coding schemes on a channel in which the input is a real signal of limited power, and the noise is additive, Gaussian, and white. His 1959 paper contained many numerical results concerning the performance of various ensembles of codes on this channel.

In Shannon's view, the channel was regarded as fixed and the codes as variable. Ignoring the question of decoding complexity, he considered a sequence of codes of increasing lengths, and a whole continuum of rates. Typically, the codes within this (vast) ensemble are selected according to one of the following criteria: 1) randomly, 2) randomly, with expurgation (meaning that a code is selected at random and then improved by discarding codewords according to some standard which typically disqualifies the worst half of the randomly generated codewords), or 3) optimally (according to an involuted criterion which optimizes whatever it is that is being studied). Shannon showed that if the code rate is fixed at any value below capacity, then a sequence of random codes at that rate has the property that their average (for each length) block-error probabilities go to zero exponentially with increasing block length,

$$P_e \approx \exp\,[-nE(R)]$$

where n is the block length and R is the rate. The function $E(R)$, called the channel reliability function, provides some information concerning the code length needed to attain a certain specified very small error probability against a well-specified channel with a known stationary signal-to-noise level.

There was considerable research in the early-to-mid 1960's on the determination of the reliability function $E(R)$. The restrictions on the types of channels for which $E(R)$ could be bounded were generalized. New analytic techniques introduced in Gallager's 1963 paper [4], which become a cornerstone of his subsequent 1968 book [5], led to a vast reduction in the length of the derivations needed to obtain the various bounds, as well as to improvements in some of the results. Various theoretical modifications of the decoding rules (such as threshold decoding, list decoding and strategies using noiseless feedback from the decoder to the encoder) were proposed and analyzed. In some sense, the lengthy two-part 1967 *Information and Control* paper by Shannon, Gallager, and Berlekamp [6], marked the climax of research on the reliability function, although several key questions about reliability functions remain unanswered to this day.

In the past decade, Shannon's formulation of the problem of analyzing the performance of various coding and decoding schemes has fallen out of favor. It is now fashionable to reverse the roles of the parameters, and to consider the code fixed and the channel variable rather than vice versa. The importance of this change of roles of the various parameters is readily appreciated by anyone who has ever tried to sell decoders. Naively paraphrased, Shannon's formulation says to the customer, "you specify the channel parameters, including the time-invariant average signal-to-noise level, as well as the prob-

ability of error which your users require, and we will plot a curve showing the tradeoffs between code length and code rate needed to meet your specification." The customer's likely response is not to buy or design any particular decoders (as nothing specific has yet been offered for sale). At best, he may embark on experimental studies trying to determine more about the channel noise environments and the level of accuracy which the users require. If these studies are ever completed, the results may typically reveal that: 1) the channel varies greatly from time to time, depending on numerous unpredictable factors, such as the weather or the interference from various sources, which may be friendly, disinterested, or hostile, and that 2) the system might be used by various classes or users who have varying requirements, ranging from bit error rates from 10^{-3} to 10^{-13}.

The modern engineering approach begins with a particular *ad hoc* choice (or a limited range of choices) of code and decoding system, and provides a curve or set of curves showing the results of some sort of analysis or simulation of how well the proposed coding/decoding system will function against a variable level of average channel noise. The big marketing advantage of this approach is that now, having been offered a specific product and a set of data curves, there is a positive probability that the prospective customer may decide to buy. The fact that he knows neither his noise environment nor his end-users' requirement with any real precision is no longer an insurmountable obstacle to the sale; the calculated performance curves provide a means of relating these two unknown constraints.

B. Deficiencies of the Orthodox Model

The modern formulation of the problem of performance evaluation has proven its value in the marketplace, and this success has led to a number of a *de facto* standards of terminology which are sometimes misapplied to situations where they are quite inappropriate.

The horizontal axis of the orthodox performance curve is measured in decibels, which presumes that the noise is Gaussian and white. These assumptions are sometimes experimentally verifiable *when no interference is present*. In order to predict performance against interference, one must have an experimentally measured (or at least a plausibly postulated) model of the interference source. The interference is typically at least somewhat non-Gaussian and almost surely it is very nonwhite. Orthodox analysis offers a strong temptation to model the interference as some additional additive level of white Gaussian noise, but this seldom, if ever, yields meaningful results. If the demodulator is typically able to detect the presence of interference, and to blank out the corresponding portion of the received waveform, then a much more appropriate model of the interference source is the *erasure burst*. If there is a significant danger that the interference level may be just weak enough to avoid being blanked out by the demodulator, then the actual performance may be so sensitive to details of the demodulator's strategy for setting the blank-out threshold that there may be no known effective way to calculate performance, even approximately, without elaborate and carefully tuned simulations. In general, whenever a significant amount of non-Gaussian interference is expected, it is wise to avoid the decibel axis entirely and substitute some less irrelevant measure of the interference level, such as *erasure rate*. When the interference is so strong that it is necessary to assume a high ratio of signal-to-background Gaussian noise, then it is more appropriate to measure performance against the interference level rather than against the (negligible) level of background Gaussian noise.

The vertical axis of the orthodox performance curve shows the decoded information-bit-error rate, plotted on a logarithmic scale. The appropriateness of this measure should be questioned in each application. Sometimes a character-error rate or a block-error rate is a more meaningful measure than a bit-error rate. Particularly in the case of Teletype text messages, burstiness of the user errors is highly desirable. A very badly garbled word or set of continguous words may actually be preferable to an isolated bit error, because the former will surely attract the reader's attention whereas the latter may create an inadvertent change of meaning. When long error-free stretches are highly desirable, it may be preferable to replace the "decoded bit-error rate" with an alternative performance measure such as the reciprocal "mean error-free run length," which might be defined as the largest value of N for which half of the decoded bits in an infinite sequence belong to error-free stretches of length N or more. Although this modified performance measure is superficially similar to the orthodox performance measure, the modified performance measure is biased against short block codes rather than against long block codes. Thus, although a long binary code of length 10 000, distance 100, and block error probability of 10^{-2} and a short code of length 20, distance 5, and block error probability of 4×10^{-4} both have a decoded bit-error rate of 10^{-4}, the mean error-free run length of the former is about 20 times as large as the latter.

A more serious objection to the "bit-error" rate measure of performance is that it implicitly treats all bits alike. If the data is numerical, then some of the bits are more significant than others. Fortunately, if the wordlength is not too long, then this difference in significance may not be too important. However, greater difference in performance specifications are sometimes inherited from higher system-wide design constraints. For example, protocol, routing information, passwords, signatures, etc., may require a much different level of reliability than any of the more conventional data. The coding system may also be required to detect and honor certain abnormal emergency or alarm messages with certain abnormal performance specifications. The orthodox approach is to add on special hardware or special codes to handle each of these specific situations. However, in many cases it is possible to handle the special situations by relatively modest changes or refinements of the "mainline" decoding system. One cannot assume that any particular performance curve applies uniformly to all subsets of messages; it is often possible to attain several levels of performance with the same system.

As a trivial example, consider a "code" which assigns 10^7 "phone numbers" to 10^7 different users, with no redundancy at all. If each digit is misdialed with some given probability, then one can compute the probability of wrong numbers in a straightforward way, and derive projections of the frequency with which each user will be bothered by wrong numbers. Now suppose we are faced with an additional constraint which requires that a certain small number of "high-status" users must see a significant reduction in the rate with which they are bothered by wrong numbers. The orthodox solution might be to introduce additional subsystems to meet the tougher performance constraints. However, if most of the system traffic goes *to and from* the small set of high-status users, then a much simpler solution might be simply to reassign the phone numbers of the high-status users in such a way that they form a subcode with high distance. This leaves the average wrong-

number rate unchanged. However, it greatly reduces the rate at which high-status users receive wrong-number calls, and it does this by ensuring that virtually all wrong numbers from high-status users lead to low-status destinations. By using more sophisticated variations on this theme, it is often possible to provide high performance for a small set of special messages within a coding system whose overall performance is much lower.

Despite all of these *caveats*, the most effective way to deal with the limitations of the orthodox curves is not to wage an intransigent fight against orthodoxy, but to extend and supplement it with additional curves showing whatever alternative measures of noise levels and performances may be relevant to the actual situation.

C. Orthogonal Codes

The best-known codes for the Gaussian channel are the codes in which the waveform representing any codeword is orthogonal to the waveform representing any other. These are the so-called "orthogonal signal" codes, which differ in relatively trivial respects from biorthogonal codes and simplex codes. The codewords can be represented by any orthogonal set, set as the classical sines and cosines, or by a set of discrete waveforms, such as the Walsh functions, or by the rows of a Hadamard matrix. With the increasing prevalence of digital logic, digital signals have enjoyed an increasing popularity. One popular embodiment of the biorthogonal codes are the first-order Reed–Muller codes, which can be decoded by fast-Fourier transform (FFT) techniques. On the other hand, advances in certain analog (or hybrid) technologies, such as surface acoustic waves, have kept the analog methods dominant in certain situations.

All of the orthogonal, biorthogonal, and simplex codes have very low code rates, and in digital implementations this means that many redundant bits must be transmitted for every transmitted information bit. In other words, long orthogonal codes require a rather high bandwidth. However, they require relatively low signal power. Furthermore, their performance curves can be calculated explicitly by evaluating certain integrals which we shall now derive.

The classical white Gaussian noise channel may be assumed to have unit power in every dimension. We consider a communication system which transmits one of N orthogonal waveforms across this channel. Each of these waveforms has signal power Δ^2. The maximum-likelihood receiver computes the dot product of the received waveform with each of the N possible transmitted waveforms. Each such dot product, or "score," is an independent univariate normal variable. The mean of this random variable is Δ for the correct waveform; it is 0 for each of the $N-1$ incorrect waveforms. The maximum-likelihood decoder is correct if the correct score is x and every incorrect score is less than x. Evidently, this is given by the integral

$$\Pr(\text{correct}) = \int_{-\infty}^{\infty} ndc^{N-1}(x)\, ndc'(x-\Delta)\, dx.$$

Here ndc represents the cumulative normal distribution function, defined as

$$ndc(x) = \int_{-\infty}^{x} \frac{\exp(-t^2/2)}{\sqrt{2\pi}}\, dt.$$

The preceding integral for the probability of correct decoding of orthogonal signals, Pr(correct), converges to one very rapidly

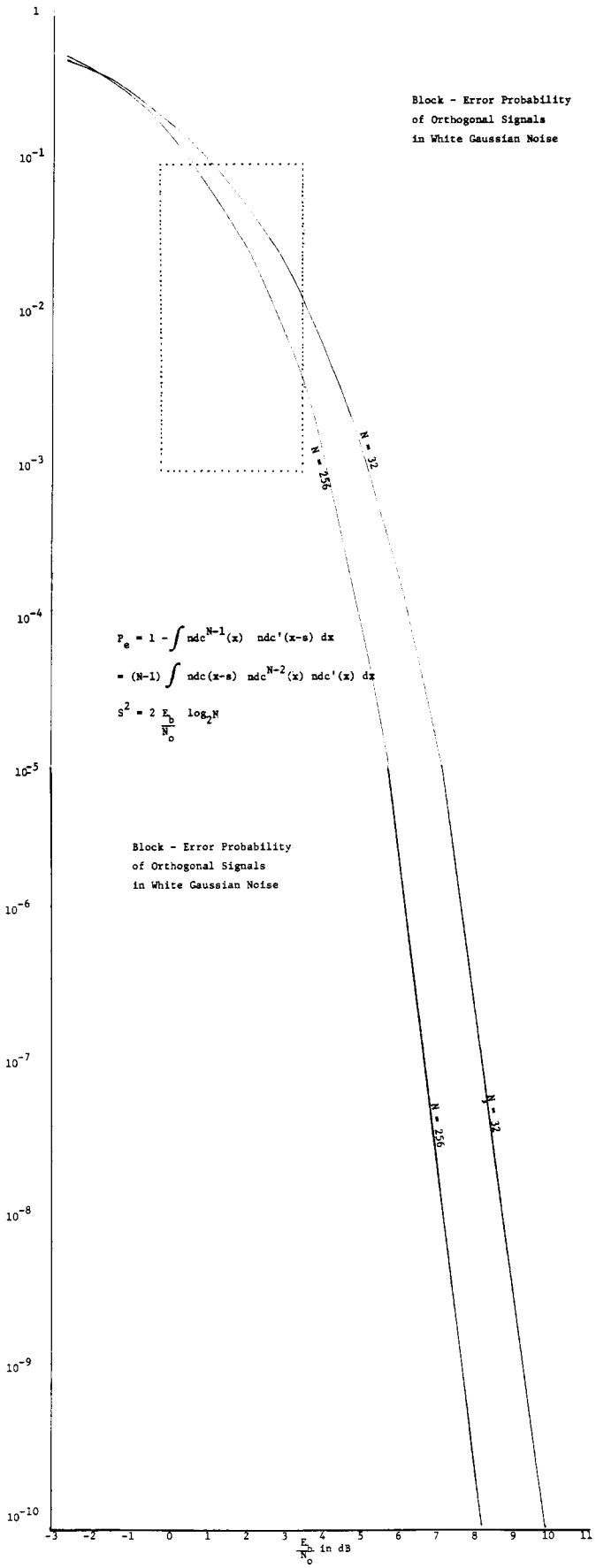


Fig. 1. Performance Curves for 32 and 256 orthogonal signals.

with increasing N. Therefore, if one seeks to determine the probability of error by subtracting this integral from one, extreme numerical accuracy may be required. An effective way to bypass this difficulty is to integrate by parts, obtaining

$$\Pr(\text{incorrect}) = (N-1)\int_{-\infty}^{\infty} ndc(x-\Delta)\,ndc^{N-2}(x)\,ndc'(x)\,dx.$$

Evidently, a decoding error occurs if any one of the $(N-1)$ incorrect signals achieves a score of x, while the other $N-2$ incorrect signals and the correct signal all score below x. We may generalize this expression for the probability of error by inserting two more parameters

$$I(\Delta,\delta_1,\delta_2) = (N-1)\int ndc(x+\delta_1-\Delta)\,ndc^{N-2}\cdot(x)\,ndc'(x+\delta_2)\,dx.$$

If the receiver refuses to decode whenever the difference between the two highest scores is less than δ, then the probability of decoding failure is given by $I(\Delta,\delta,0)$. This may be regarded as the sum of the probabilities of two failure modes: "decoding erasure," which happens whenever the decoder refuses to decode, and "decoding error," which occurs whenever a wrong signal outscores all others by at least δ. The probability of decoding error is given by $I(\Delta,0,\delta)$. By evaluating this integral numerically for the cases when $N=32$ and $N=256$, and making the appropriate corrections to convert block-error rates to bit-error rates, we obtain the performance curves shown in Figs. 1 and 2. The portion of the $N=32$ curve shown in the dotted box of Fig. 1 is expanded and redrawn in Fig. 2, along with the modifications obtained by introducing small null zones of values $\delta=0.05$ and $\delta=0.15$.

Many other authors have tabulated these curves in the case of $\delta=0$. Numerical tables may be found in Viterbi's Appendix 4 to *Digital Communications* by Golomb *et al.* [7], or in Table 5-1 of *Telecommunication Systems Engineering* by Lindsey and Simon [8]. But a morass of raw numbers alone yields no insight, and it is more instructive to seek asymptotic approximations to this integral when N is very large. We will suppress the analytic calculations, but describe the phenomena which are discovered when this is done.

If N is sufficiently large, and if Δ^2 is bigger than $8\ln N$, then the largest values of the integrand occur in the vicinity of the point $x=\Delta/2$. Since $nd(\Delta/2) \ll 1/(N-2)$, the integral for $I(\Delta,0,0)$ is closely approximated by the well-known "union bound"

$$I(\Delta,0,0) \approx (N-1)\int_{-\infty}^{\infty} ndc(x-\Delta)\,ndc'(x)\,dx.$$

Errors evidently occur when the noise has magnitude of about $\Delta/2$ and it is oriented directly at some particular wrong codeword. When wrong, the decoder typically sees a two-way tie, with the third, fourth, ... choices far behind. In this case, when the signal level is above the *critical* signal level of 4.4 dB, the code has rate less than Shannon's classical R_{crit}.

However, if $\Delta/2$ is less than μ, which is defined implicitly as the solution of

$$nd(\mu) = \frac{1}{N-2}$$

then, for large N, the maximum value of the integrand of

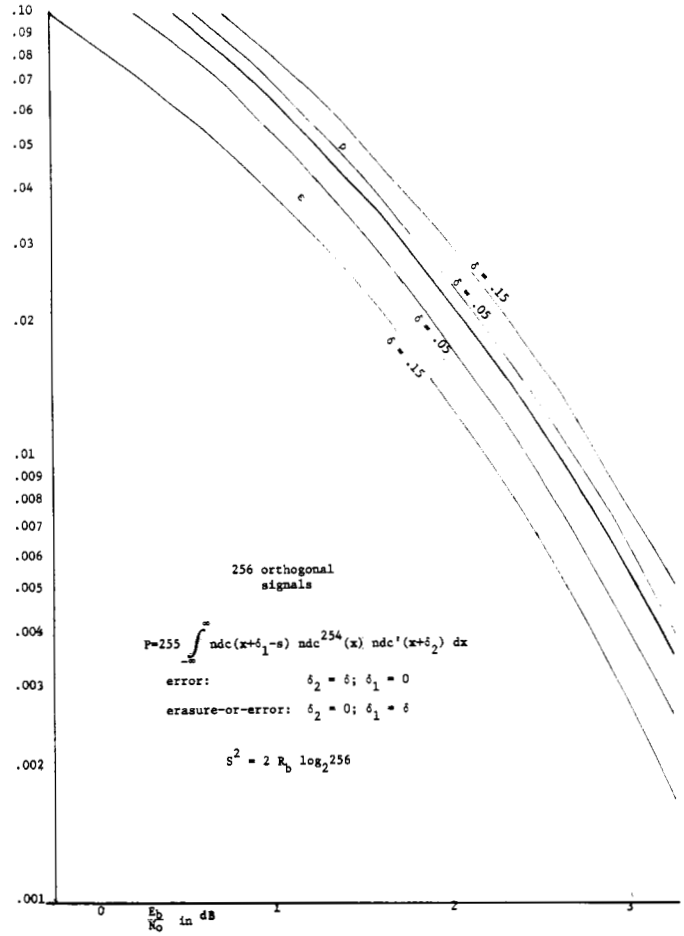


Fig. 2. Enlargement of part of Fig. 1.

$I(\Delta,0,0)$ occurs not in the vicinity of $x=\Delta/2$, but in the vicinity of $x=\mu$. This corresponds to the code rate exceeding the classical rate R_{crit}. In this case, when the decoder is wrong, it sees a list of choices virtually identical to the choices it would typically see if no signal had been transmitted! The first, second, third, ... choices all have scores similar to those which would be expected from the largest N independent samples of pure noise. The scores of many choices are relatively close. The correct signal is "lost in the pack." In some sense, the troublesome noise should be interpreted as oriented from the correct codeword to the mean of the other codewords rather than towards any particular one of them.

If we allow the decoder to decode only when its first choice exceeds its second choice by some fixed amount $\delta>0$, then the original decoding error probability P splits into two different probabilities: a "decoding failure" probability ρ and a decoding error probability ϵ.

The decoding failure probability may be regarded as the sum of ϵ and a probability that the decoder is unable to decode because its first and second choices are too close. At signal-to-noise ratios (SNR) above the critical 4.4 dB, the ratio by which ϵ is reduced below P is the same as the ratio by which ρ rises above P. However, at lower SNR's, the introduction of a null zone δ suppresses ϵ by more than it raises ρ. In the extreme limit of very, very long codes operating very near the channel capacity (i.e., SNR of -1.6 dB), the introduction of the null zone of amount δ decreases the decoding error probability significantly with essentially *no* increase in decoding failure probability. Intermediate SNR's and modest block

lengths yield intermediate results, as revealed in the example shown in Fig. 2.

D. Reed-Solomon Codes

Although it is possible to achieve arbitrarily small output error rates at all SNR's above channel capacity (i.e., -1.6 dB) by using sufficiently long orthogonal codes, the bandwidth requirements grow very rapidly. Even if a large bandwidth is available, the memory and processing constraints rapidly become formidable. Very large FFT machines have been seriously considered in other contexts. One of the more grandiose such designs is a proposed billion-point/10 s FFT machine which might someday be used in conjunction with the CYCLOPS program to listen for intelligent signals from other worlds. By using spiral track disks as delay-lines, the cost of that FFT machine could be kept below the cost of many large scientific computers. But as the decoder for an orthogonal code, the billion-point FFT machine would emit decoded information at the disappointingly low rate of 31 bits/10 s.

A much more effective strategy for attaining high performance at low or moderate SNR's on wide-band Gaussian channels is to combine an orthogonal code of short to moderate length (e.g., 32 or 256) with a RS [9] code. In contrast to the one or two roomfuls of expensive hardware comprising the proposed FFT machine, a powerful 100 Mb/s RS decoder can now be constructed on one or two inexpensive cards. Appendix A presents a detailed itemization of the amount of hardware needed to implement various RS decoders in 1975. In fact, if the rate of the RS code is sufficiently high, the overall implementation costs are dominated more by the devices needed to decode the (inner) orthogonal code rather than by the costs of the (outer) RS code. In addition, the bandwidth used by the concatenated code is much less than the bandwidth used by a single long orthogonal code.

The RS code treats each codeword of the orthogonal code as a single character in its large alphabet. If the RS code has an alphabet of size $q = 2^m$, then its block length may be taken as $N = q$. (Actually, this is an "extended" RS code; the conventional RS codes have lengths $N = q - 1$, but they can be easily extended to lengths q and $q + 1$, although not any further. The length $N = q$ simplifies the analysis, and the adjacent lengths lead to only trivial differences in the results.) If the RS code has r redundant characters, then its distance is $d = r + 1$, and the code is able to correct any pattern of t character errors and s character erasures for which

$$2t + s < d.$$

Of course, any linear code is able to correct r character erasures if the r erased characters all happen to lie on the redundant bits. Under these circumstances, the decoding can be accomplished simply by reencoding. However, the RS codes have the remarkable property that they are able to correct *any* set of r character erasures. Codes with this property are called "maximum distance separable codes." Maximum distance separable codes of block lengths greater than three are necessarily nonbinary, and their erasure-correcting capabilities are much better than any binary code. For example, consider the (1023, 1013, 3) binary Hamming [10] code, which has 1023 bits in each codeword, of which 1013 are information bits and 10 are check bits. The distance of this code is 3. Since the code has 10 redundant bits, it is able to correct some patterns of 10 bit erasures (e.g., the 10 redundant bits), but it is unable to correct a few patterns of only 3 erasures (namely, the 3 bits corresponding to any Hamming codeword of weight 3). This dichotomy between error-correction capability and erasure-correction capability of binary codes was investigated by Heller [10].

TABLE II
SOME VALUES OF p WHICH SATISFY THE EQUATION

$$P_e = \sum_{j>r} \frac{j}{2(n-1)} \binom{n}{j} p^j (1-p)^{n-j}$$

	$P_e = 10^{-3}$	$P_e = 10^{-5}$	$P_e = 10^{-7}$	$P_e = 10^{-9}$	$P_e = 10^{-11}$
			$n = 8$		
$r = 2$	4.60E-2	9.51E-3	2.03E-3	4.37E-4	9.41E-5
$r = 4$	1.49E-1	5.65E-2	2.21E-2	8.74E-3	3.47E-3
			$n = 16$		
$r = 2$	2.84E-2	5.73E-3	1.22E-3	2.62E-4	5.63E-5
$r = 4$	7.72E-2	2.81E-2	1.09E-2	4.27E-3	1.69E-3
$r = 8$	2.23E-1	1.23E-1	7.10E-2	4.17E-2	2.47E-2
			$n = 32$		
$r = 2$	1.80E-2	3.55E-3	7.51E-4	1.61E-4	3.47E-5
$r = 4$	4.36E-2	1.54E-2	5.87E-3	2.30E-3	9.11E-4
$r = 8$	1.11E-1	5.88E-2	3.32E-2	1.93E-2	1.13E-2
$r = 16$	2.90E-1	2.01E-1	1.45E-1	1.07E-1	7.96E-2
			$n = 64$		
$r = 2$	1.17E-2	2.23E-3	4.68E-4	1.00E-4	2.16E-5
$r = 4$	2.55E-2	8.70E-3	3.29E-3	1.29E-3	5.08E-4
$r = 8$	5.96E-2	3.05E-2	1.70E-2	9.78E-3	5.74E-3
$r = 16$	1.43E-1	9.50E-2	6.71E-2	4.87E-2	3.59E-2
$r = 32$	3.45E-1	2.74E-1	2.25E-1	1.98E-1	1.58E-1
			$n = 128$		
$r = 2$	7.72E-3	1.41E-3	2.94E-4	6.29E-5	1.35E-5
$r = 4$	1.52E-2	5.00E-3	1.87E-3	7.29E-4	2.88E-4
$r = 8$	3.30E-2	1.63E-2	8.99E-3	5.15E-3	3.01E-3
$r = 16$	7.44E-2	4.81E-2	3.35E-2	2.41E-2	1.76E-2
$r = 32$	1.69E-1	1.30E-1	1.05E-1	8.59E-2	7.16E-2
$r = 64$	3.88E-1	3.34E-1	2.95E-1	2.64E-1	2.38E-1
			$n = 256$		
$r = 2$	5.20E-3	8.94E-4	1.85E-4	3.96E-5	8.51E-6
$r = 4$	9.29E-3	2.90E-3	1.08E-3	4.17E-4	1.64E-4
$r = 8$	1.86E-2	8.89E-3	4.83E-3	2.76E-3	1.61E-3
$r = 16$	3.98E-2	2.50E-2	1.72E-2	1.23E-2	8.97E-3
$r = 32$	8.68E-2	6.51E-2	5.17E-2	4.21E-2	3.49E-2
$r = 64$	1.91E-1	1.60E-1	1.39E-1	1.23E-1	1.10E-1
			$n = 512$		
$r = 2$	3.62E-3	5.70E-4	1.17E-4	2.49E-5	5.35E-6
$r = 4$	5.80E-3	1.70E-3	6.20E-4	2.39E-4	9.41E-5
$r = 8$	1.07E-2	4.88E-3	2.62E-3	1.49E-3	8.64E-4
$r = 16$	2.16E-2	1.32E-2	8.97E-3	6.37E-3	4.64E-3
$r = 32$	4.54E-2	3.33E-2	2.62E-2	2.12E-2	1.75E-2
$r = 64$	9.68E-2	7.97E-2	6.85E-2	6.01E-2	5.34E-2
			$n = 1024$		
$r = 2$	2.65E-3	3.66E-4	7.38E-5	1.57E-5	3.37E-6
$r = 4$	3.75E-3	9.95E-4	3.58E-4	1.38E-4	5.40E-5
$r = 8$	6.29E-3	2.70E-3	1.43E-3	8.05E-4	4.67E-4
$r = 16$	1.19E-2	6.99E-3	4.71E-3	3.33E-3	2.41E-3
$r = 32$	2.40E-2	1.72E-2	1.34E-2	1.08E-2	8.90E-3
$r = 64$	4.99E-2	4.04E-2	3.45E-2	3.01E-2	2.67E-2
			$n = 2048$		
$r = 2$	2.14E-3	2.36E-4	4.67E-5	9.89E-6	2.12E-6
$r = 4$	2.58E-3	5.88E-4	2.07E-4	7.92E-5	3.10E-5
$r = 8$	3.82E-3	1.50E-3	7.81E-4	4.37E-4	2.53E-4
$r = 16$	6.68E-3	3.73E-3	2.48E-3	1.74E-3	1.26E-3
$r = 32$	1.29E-2	8.95E-3	6.91E-3	5.55E-3	4.55E-3
$r = 64$	2.59E-2	2.06E-2	1.75E-2	1.52E-2	1.34E-2
			$n = 4096$		
$r = 2$	2.00E-3	1.53E-4	2.96E-5	6.24E-6	1.34E-6
$r = 4$	2.06E-3	3.49E-4	1.20E-4	4.57E-5	1.78E-5
$r = 8$	2.50E-3	8.36E-4	4.28E-4	2.38E-4	1.37E-4
$r = 16$	3.87E-3	2.00E-3	1.31E-3	9.17E-4	6.60E-4
$r = 32$	6.99E-3	4.67E-3	3.57E-3	2.86E-3	2.34E-3
$r = 64$	1.36E-2	1.06E-2	8.92E-3	7.73E-3	6.81E-3

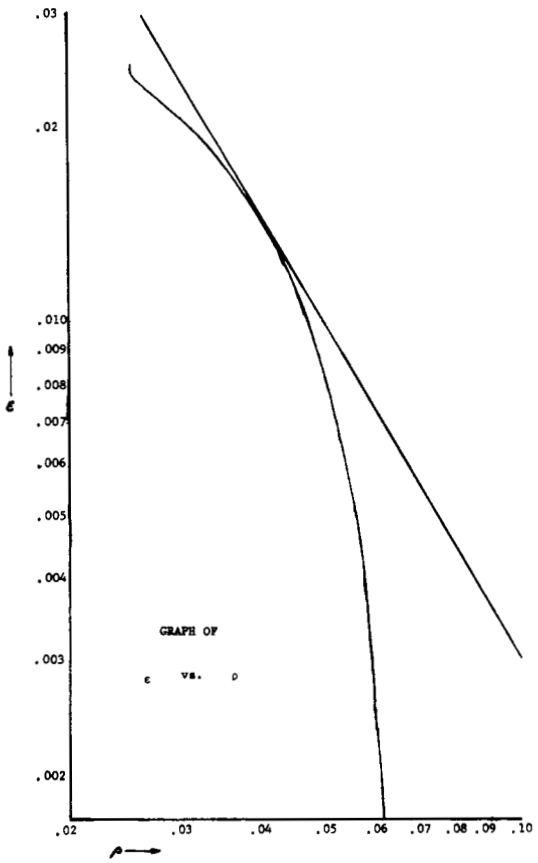


Fig. 3. Solutions of $10^{-5} = \sum_{s+2t>32} \frac{(s+t)\,256!\,\epsilon^t(\rho-\epsilon)^s(1-\rho)^{256-s-t}}{510\,s!\,t!\,(256-s-t)!}$

Although RS codes can be designed to have any redundancy, the complexity of high-speed implementations increases with increasing redundancy. For this reason, the most attractive RS codes are often those with relatively high code rates (and hence relatively low redundancies). In order to provide some specific points of reference, we have calculated the performance of many (extended) RS codes having lengths $n = q = 2^m$, $m = 3, 4, 5, \ldots, 12$ and redundancy $r = 2^i$. The input probabilities needed for each such RS code to achieve a decoded output bit-error rate of 10^{-3}, 10^{-5}, 10^{-7}, or 10^{-11} are shown in Table II.

The error-correcting capability of an RS code is equal to one half of its erasure-correcting capability. Since the redundancies of the codes listed in Table II have been chosen to be powers of 2, we may use the numbers in the table to evaluate the performance either against character errors alone or against character erasures alone. As an illustrative example, consider the RS code whose characters are 8-bit bytes, whose length is 256, and whose redundancy is 32. Suppose that this code must attain an output bit-error rate of 10^{-5}. If the noise environment consists entirely of character erasures (presumably caused by interference rather than by Gaussian noise), then according to Table II, the maximum input character-erasure rate is 6.51E-2 = 0.0651. On the other hand, if we suppose that the noise consists entirely of character-errors (presumably caused by an unsophisticated inner decoder which is incapable of declaring erasures, no matter how unconfident it is of its decisions) then we can determine the maximum input character-error rate from the preceding line of Table II; it is 2.50E-2 = 0.025.

The problem of determining the input specifications when both character errors and character erasures are present is considerably more complicated, as it necessarily depends on the possible tradeoffs between the two. As an illustrative example, we continue the consideration of this same RS code, with block length, dimension, distance given by $(n, k, d) = (256,224,33$. If we suppose that the characters which are input to this code arise from some decoder for an orthogonal code on the classical white-Gaussian channel, then Fig. 2 becomes applicable. By examining this figure, we may deduce a relationship between the original error probability P and the pair of probabilities ρ and ϵ. The values of the various functions at 1.5 dB are 0.051 = 0.038 X exp (0.29), 0.042 = 0.038 X exp (0.10), 0.038, 0.032 = 0.038 X exp (-17), 0.0233 = 0.038 X exp (-0.49), from which we empirically infer that at 1.5 dB,

$$\epsilon \approx P \exp(-1.7\mu)$$

$$\rho \approx P \exp(\mu)$$

where μ is related to the null-zone width δ.

Having determined the attainable tradeoffs between ϵ and ρ in this model, we must next perform an independent calculation to determine the range of ϵ and ρ which will satisfy a 10^{-5} output constraint. This was done by iterative numerical methods, and the results are plotted on the graph of Fig. 3. The curve of this figure begins at the point $\epsilon = \rho = 0.0250$, and continues to the asympotic point at which $\epsilon = 0$, $\rho = 0.0651$.

After determining this curve, we then match the values of ϵ and ρ attainable from a fixed E_b/N_0 with the values which are sufficient to achieve the specified performance. On the logarithmic scale of Fig. 3, the former values fall on a straight line of known slope, and the value of P (and hence the maximum tolerable average noise level, in decibels), is attained by selecting the line which is tangent to the curve. This yields the operating point

$$\rho = 0.040, \quad \epsilon = 0.0185, \quad P = 0.030,$$

$$E_b/N_0 = 1.72, \quad \text{and} \quad \delta \approx 0.15.$$

In other words, by carefully selecting the null-zone threshold δ, we acquire only 0.15 dB improvement over the original $E_b/N_0 = 1.87$, $P = 0.025$ from which the RS decoder can achieve the 10^{-5} bit-error rate by correcting errors only, which obviates the additional threshold-setting problems involved in deciding when to declare erasures.

Similar calculations done by others have led to similar conclusions: Distinguishing between character-errors and character-erasures on a classical white-Gaussian noise channel leads to very little improvement in performance. In the above example, we can save only 0.15 dB. At higher SNR's and lower bit-error rate requirements, the improvement is even less.

However, these disappointing conclusions do *not* imply anything at all about the performance of character-erasure-based strategies when the noise arises from interference sources rather than from the classical white Gaussian noise. In many such situations, character-erasure-based strategies are the natural approach, and they often outperform character-error-based strategies by a substantial margin.

The combination of orthogonal signals and RS codes is a very effective way to gain high performance with low signal-to-noise margins on classical white Gaussian noise channels which have wide bandwidths and relatively slow speed requirements. However, when the white Gaussian noise channel has stringent bandwidth constraints as well as stringent signal power constraints, then the use of a large alphabet of orthogonal

signals becomes infeasible. One of several methods of dealing with this difficulty is to replace the inner orthogonal code with a short binary code of much higher rate. For example, if the outer code is a RS code of length 512 9-bit characters, then the inner code might be any code with 9 information bits. If the transmission speed and bandwidth requirements preclude the choice of the (256, 9, 128) first-order binary Reed-Muller code, then we might instead employ another inner code such as the (18, 9, 6) binary quadratic residue code. In order to evaluate the performance of such options, we need to investigate effective methods to decode short binary codes on the Gaussian channel, and to evaluate their performance under relatively weak SNR's. These topics are considered in the next two sections.

E. The Union Bound and the Tangential Union Bound

One conventional technique for obtaining estimates of the performance of any particular code is the so-called *union bound*, which is based on the fact that the probability of the union of any set of events is no greater than the sum of their probabilities. In particular, suppose we have a code which consists of one word of weight 0, and A_j codewords of weight j, for $j = 1, 2, 3, \cdots, n$. (If the code is relatively good, then we should have $A_j = 0$ for sufficiently small nonzero values of j.) If a maximum-likelihood decoder makes an error, then it must be due to the fact that the received signal lies closer to one of the wrong codewords than it does to the correct codeword. Hence, a maximum-likelihood decoding error may be regarded as the union of many events, each event being that the noise moved the received signal closer to a particular wrong codeword than to the correct codeword. Thus the union bound allows us to estimate the probability of maximum-likelihood decoding error in terms of the error probabilities for relatively trivial codes consisting of only two codewords. To be specific, suppose that we have a binary code in which each 0 is transmitted as a positive S and each 1 is transmitted as a negative S, where S is the signal amplitude, and every orthogonal component of the noise has unit variance. Then the Euclidean distance between two codewords which differ in w bits is $2S\sqrt{w}$, and so the probability that the noise causes the received signal to lie closer to a particular codeword at distance w from the transmitted codeword is precisely $nd(S\sqrt{w})$. Applying the union bound then leads to the following estimate of the error probability of the entire code:

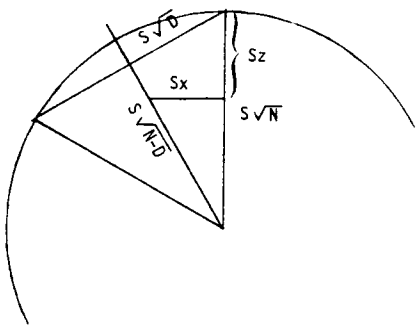
$$\Pr(\text{Maximum-Likelihood Error}) \leq \sum_{w=1}^{n} A_w nd(S\sqrt{w}).$$

By invoking the classical reliability results of the 1960's, it can be shown that this bound is actually quite tight for sufficiently large SNR's and this has led to the widespread popularity of the union bound. Unfortunately, however, the union bound is quite weak at low SNR's ratios. The point of demarcation between high SNR's and low SNR's depends on the code, and it is often difficult to calculate. For many codes, it occurs at a lower SNR than any point of operating interest, and so the widespread unquestioning acceptance of the general tightness of the union bound has not led to as many engineering mistakes as a rigorously oriented academic might fear. Nevertheless, examples of real problems which lead to weak union bounds are not hard to find. Very long orthogonal codes, for example, have a critical SNR of 4.4 dB; these codes achieve a wide range of interesting bit-error probabilities at significantly lower SNR's, at which the union bound is definitely weak. Luckily, this causes no serious difficulties because the performance of orthogonal codes has been evaluated explicitly by definite integrals, so no one uses the (weak) union bounds for them anyway.

The union bounds for higher rate binary codes become weak only when the decoded bit-error rate is relatively bad (e.g., 10^{-1} or 10^{-2}), and while this is generally not a viable region for overall operation, it can become important in concatenated coding systems in which the high-rate binary code is to be used as the inner code in conjunction with an outer RS code which will reduce the overall user bit-error rate to a much smaller value.

Although techniques for obtaining improvements on the union bound for high-rate random codes have long been known, these methods are not easily applied to specific codes with given weight enumerators. We have recently developed a simple technique to do that, and it is (to our knowledge) presented here for the first time.

We assume that we are considering a binary code on the white Gaussian noise channel. In this case, we can improve the union bound by separating the noise into a radial component and a tangential component. All of the signal points are on the surface of a Euclidean sphere of radius $S\sqrt{N}$. The square of the difference between any two codewords at Hamming distance D is $D(S - (-S))^2 = 4DS^2$, so the Euclidean distance is $2S\sqrt{D}$. Thus the minimum value of noise amplitude needed to cause an error to a minimum weight neighbor is $S\sqrt{D}$, as shown in the following figure:



If we suppose that the radial component of the noise has value Sz, then the minimum value of the orthogonal component needed to cause an error to a nearby codeword is

$$Sx = (S\sqrt{N} - Sz)\tan\left(\arcsin\sqrt{\frac{D}{N}}\right)$$
$$= (S\sqrt{N} - Sz)\sqrt{\frac{D}{N-D}}.$$

Thus

$$P_{MLE} \leq \sum_{D} A_D \int nd\left((S\sqrt{N} - Sz)\sqrt{\frac{D}{N-D}}\right) nd'(Sz)\, d(Sz).$$

If one makes the substitutions

$$y = z\sqrt{\frac{D}{N}} + x\sqrt{\frac{N-D}{N}}$$
$$w = z\sqrt{\frac{N-D}{N}} - x\sqrt{\frac{D}{N}}$$

then $x^2 + z^2 = y^2 + w^2$ and it follows by straightforward

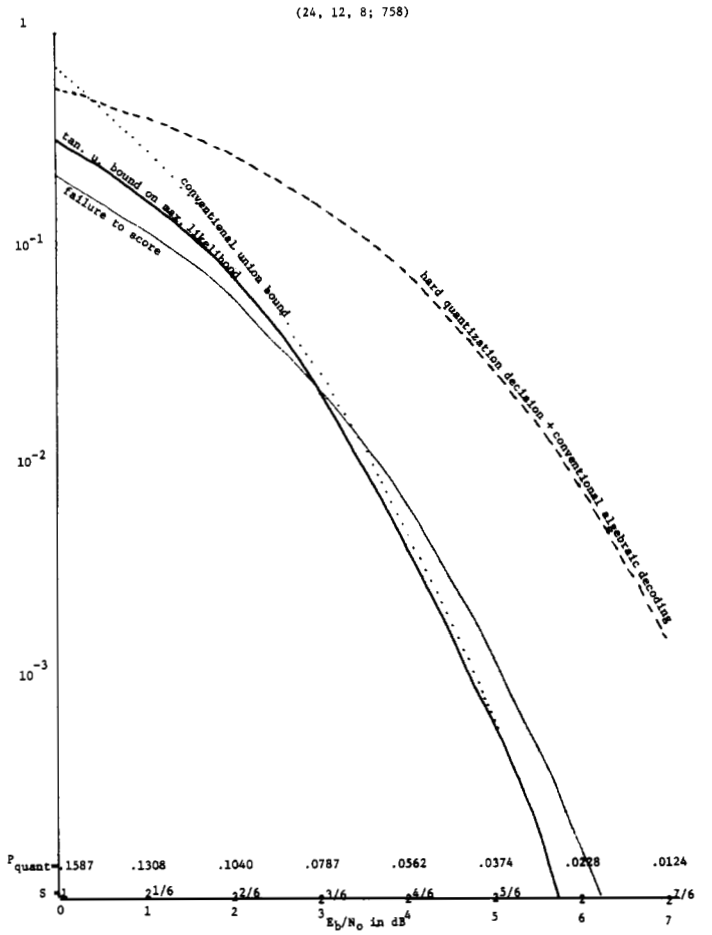


Fig. 4.

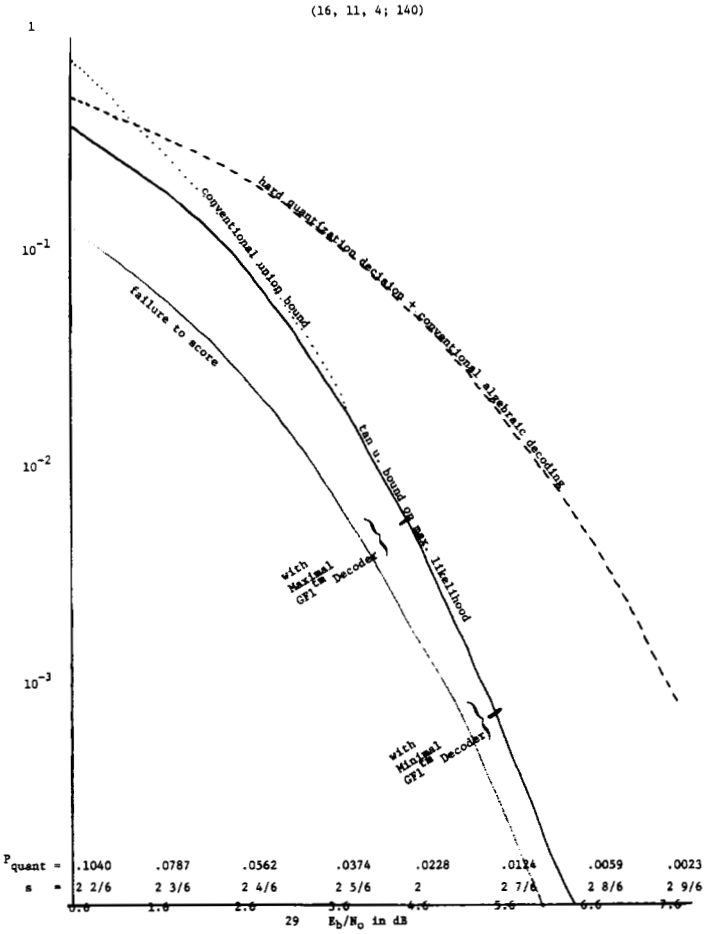


Fig. 5.

manipulation that

$$\int A_D \; nd(Sx)\, nd'(Sz)\, dz = A_D \; nd(S\sqrt{D}).$$

Although the right side of this equation is simpler than the left side, the left side is more easily modified to yield a tighter result. Any noise vector may be projected onto radial and tangential components, and since the tangential hyperplane of the N-dimensional sphere is orthogonal to the radial vector, the radial component of the noise is a Gaussian random variable which is orthogonal to all tangential noise components. For any given value of the radial noise component, a decoding failure occurs if the tangential noise lies outside of some appropriately defined convex region within the tangential hyperplane. Since the probability of large tangential noise cannot exceed 1, we may replace the factor

$$A_D \; nd\left((S\sqrt{N} - Sz)\sqrt{\frac{D}{N-D}}\right)$$

by 1 whenever z is sufficiently large for this bound to yield an improvement. Thus

$$P_{MLE} \leq \sum_D \int_{-\infty}^{S\hat{z}} A_D \; nd\left((S\sqrt{N} - t)\sqrt{\frac{D}{N-D}}\right) nd'(t)\, dt + nd(S\hat{z}).$$

This bound is valid for any value of $\hat{z}$, but the bound is optimized by the value of $\hat{z}$ which is defined implicitly by the equation

$$nd\left((S\sqrt{N} - S\hat{z})\sqrt{\frac{d}{N-d}}\right) = \frac{1}{A_d}$$

where d is the code's minimum distance. This refined bound on P_{MLE} is called the *tangential union bound*. The tangential union bound is significantly better than the conventional union bound at low signal levels S, but the ratio of the two bounds approaches 1 rapidly with increasing signal level S.

The tangential union bounds for the binary codes with $(n,k,d;\, A_d)$ = (18,9,6; 102), (16,11,4; 140), (24,12,8; 758), and (32,21,6; 992) are shown in Figs. 4–7. In each case, the tangential union bound is a solid dark line, and the conventional union bound is a dashed line which joins the tangential union bound at high signal-to-noise level. In accordance with the properties of a particular decoding algorithm which is discussed in the following section, both the union bound and the tangential union bound shown in each of these figures actually includes only the dominant term proportional to A_d (where d is the minimum distance of the code) rather than the sum over all D.

It is known that at low SNR's, the bounds on the classical reliability function which are obtained by random coding arguments are "correct" in an appropriate asympotic sense because they yield results which coincide with opposing bounds obtained by other methods. We would like to conjecture that the tangential union bound is similarly "correct," but we have

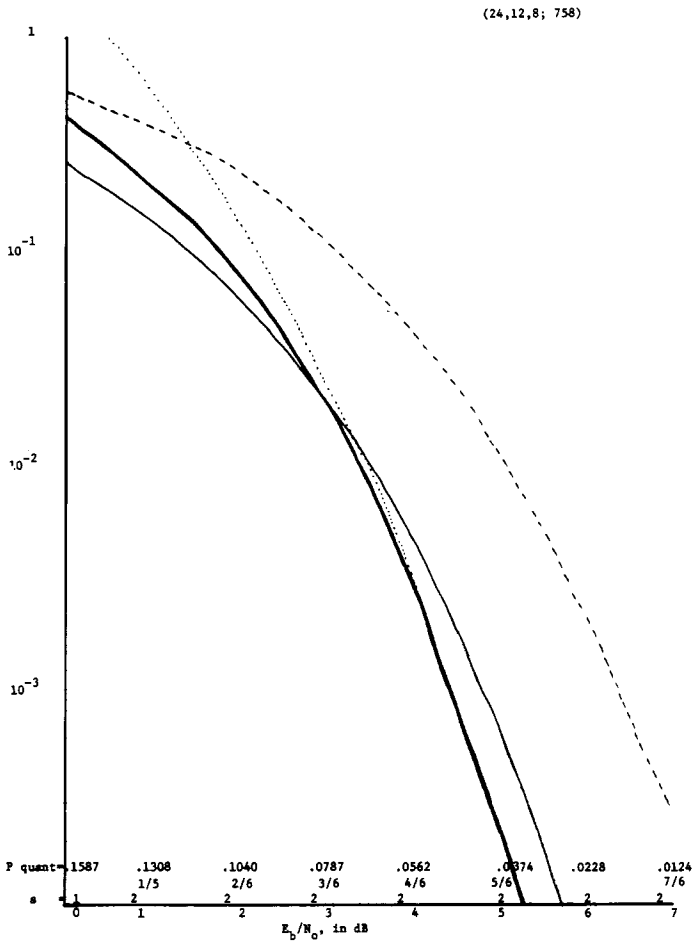


Fig. 6.

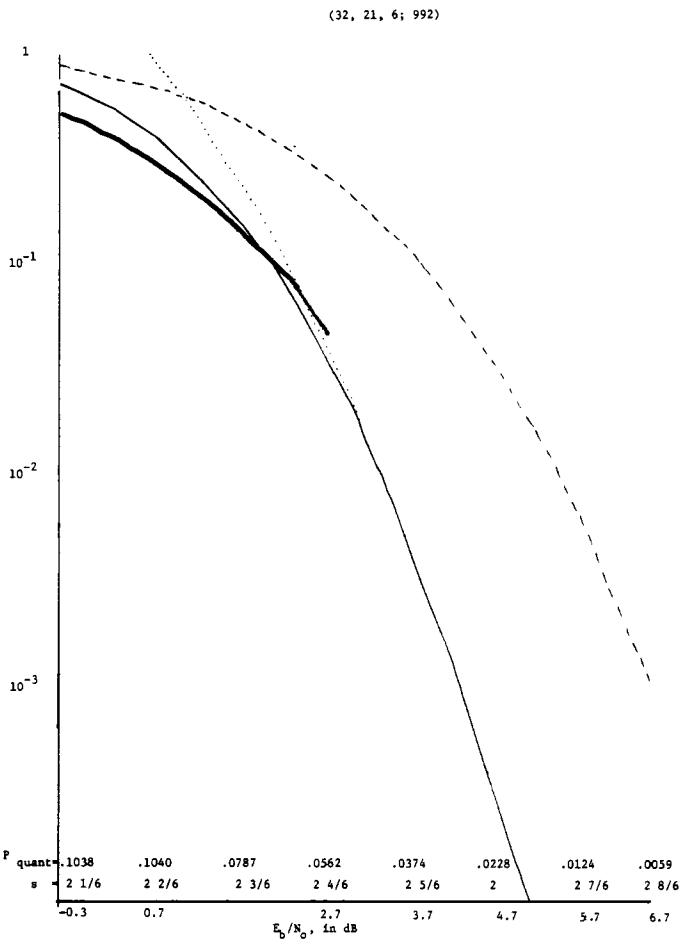


Fig. 7.

no precise formulation of what is meant by this conjecture, nor any real supporting evidence other than some rather straightforward calculations which show that the tangential union bound lies close to the known answer in the special case of long orthogonal codes.

F. Performance of Gaussian Decoders Which Try to Correct d/2 Errors Plus a Single Erasure

We consider relatively short binary block codes of moderately high rates, such as those with $(n,k,d) = (16,11,4)$, $(18,9,6)$, $(24,12,8)$, or $(32,21,6)$. We further assume that the code satisfies an overall parity check. We assume the simplest model, in which the channel consists solely of additive white Gaussian noise. Since the short code we consider here might be used as the inner code in a concatenated system which includes a powerful RS outer code, we shall consider the case of poor SNR's (and relatively high error probability for the short block code) as well as the more traditional case of moderate SNR's. Although we might reduce the probability of decoding error by allowing the decoder to refuse to decode whenever its decision is too close, the preliminary analysis which we will present in Figs. 4–7 neglects that refinement.

High-rate codes have a relatively large number of codewords and a relatively small number of syndromes, and so syndrome-based decoding strategies (such as those conventionally used to decode Hamming codes) potentially require much less computation than codeword-based search strategies (such as the FFT methods conventionally used to decode first-order Reed-Muller codes).

The decoding strategy which we shall evaluate is the following:

1) examine the received number corresponding to each bit of the block, and determine the least reliable bit; depending on the value of the overall parity check, this least reliable bit may be changed in order to ensure that the total number of bit errors in the received block has the same parity as $d/2$;
2) recompute the syndrome;
3) compute the sum of the received amplitudes corresponding to each coset leader;
4) the coset leader having minimum sum is the presumed error pattern, although it may be rejected if the sum corresponding to any other coset leader is too close.

All words of weight $\leqslant d/2$ are coset leaders. Most cosets contain many coset leaders each having weight $> d/2$. All coset leaders in any particular coset are disjoint, and this property ensures that no received number is added into more than one score, thereby facilitating very-high-speed implementations of this algorithm. In fact, this algorithm for decoding short high-rate binary codes on the Gaussian channel requires only about $1/(\log_2 N)$ times as many real additions as the well-known fast Fourier algorithm for maximum-likelihood decoding of low-rate orthogonal codes. Although the detailed distribution of bits among coset leaders is a function of the syndrome that is not easily comprehended by humans, many

codes (including all of those mentioned above) have sufficient structure to facilitate a very efficient implementation [12, sec. 16.48].

This decoding algorithm must fail whenever the number of bit-quantization errors is $d/2 + 2$ or larger, and this happens with probability

$$P_1 = \sum_{j=d/2+2}^{n} \binom{n}{j} ndc(S)\, nd^{n-j}(S)$$

where S is the signal amplitude per transmitted bit. The decoder must also fail whenever the number of quantization errors is exactly $d/2 + 1$ *and* another received estimate lies closer to the quantization threshold than any of the correct numbers. This probability, which is denoted by P_2, has been calculated numerically. The sum $P_1 + P_2$ gives the probability that the actual error pattern is not among the coset leaders which are scored.

If the actual error pattern is among the coset leaders which are scored, then the decoded word either agrees with the transmitted codeword or differs from it by a codeword of minimum weight d. Thus every decoding error or failure is the result of one (or both) of two causes: 1) the decoding algorithm fails to score the actual error pattern, or 2) the actual error pattern is outscored by an incorrect error pattern which differs from it by a codeword of minimum weight. This probability may be regarded as the "maximum-likelihood" error probability. Both of these probabilities are plotted as solid curves in Figs. 4–7. The probability of failure to score is typically below the max-likelihood error probability at low SNR's, but lies above it at high SNR's. The overall probability of decoding error is larger than the maximum of these two probabilities, but less than their sum.

For comparative purposes, Figs. 4–7 also show the disappointing performance of the decoding scheme which quantizes each received bit and then applies conventional algebraic decoding techniques for binary block codes. The lesson taught by these curves is that the cost of ignoring likelihood information which is available at the output of the Gaussian channel is several decibels. Reasonable performance on the Gaussian channel requires some additions of real numbers with carries and the relatively slower speed which that implies. The much more easily implemented Galois field arithmetic alone is insufficient.

However, the point of this section is that a relatively few additions can yield a big improvement. With each of the four codes shown in Figs. 4–7, maximum-likelihood performance is closely approximated by an algorithm which requires only one real addition per received number.

The performance of the algorithm which tries to correct $d/2$ errors plus one erasure deteriorates as the codes become longer or as the SNR increases. The generalization of this algorithm to longer codes has not yet been thoroughly investigated, partly because of probable implementation difficulties related to the general lack of understanding of the structure of cosets which have weight larger than half the minimum distance.

At higher SNR's, this strategy can be improved by correcting more erasures and fewer bit errors. Since this change of emphasis coincides with the requirements of many channels which have serious problems with interference noise sources, erasure-based decoding algorithms are more properly evaluated on non-Gaussian channels, even though many of them perform quite respectably on Gaussian channels as well.

G. An Example with Illustrative System Constraints

As an illustrative example, we now consider a white Gaussian noise channel with the following system constraints:

1) the modulation must be binary, and the channel transmission rate must be at least 60 MHz;
2) due to bandwidth limitations, the code rate must be at least 2/3, preferably higher;
3) the decoded bit-error rate must be 10^{-9} or better; no retransmission is allowed; there is no distinction between decoding failures and miscorrections;
4) the block length may not exceed 30 000 bits;
5) the decoder must operate continuously, without falling behind the channel; however, as long as the delay is constant, there is no objection to it being as large as tens of block lengths;
6) total hardware shall not exceed 600 IC's, preferably much less; encoding and decoding hardware costs receive equal weight (i.e., both will be situated on the ground); hardware power consumption is unimportant (i.e., emitter-coupled logic is acceptable).

Transmitter power, decoder hardware costs, and code rate are the parameters among which tradeoffs are possible.

Because of the constraint on block length, and the compatibility with the (16,11) Hamming code, we select RS codes with an 11-bit character size as the baseline choice. Such codes have a natural block length of 11×2047 bits. We select 64 characters as the baseline redundancy. This gives an RS coderate of $R = 1983/2047$, well above the specification. The redundancy is sufficiently large that the performance is limited by the capabilities of the decoding hardware rather than by the capabilities of the RS code. An encoder cost for this code is not a significant part of the total cost, particularly if one applies modern principles of algebraic encoder design [13].

In order to utilize soft-decision information (if any is available), we might wish to combine this RS code with a short inner code. One possibility would be a Viterbi decoder for a convolutional code with constraint length sufficiently short to be economically feasible at the required speed (e.g., rate $\frac{3}{4}$ and constraint length $k = 3$). The analysis of such a concatenated system proves difficult, because of the difficulties of modelling the burstiness of the error patterns coming out of the Viterbi decoder when it is operating in a very poor noise environment. Hence, we propose using a (16,11) Hamming inner code instead. This facilitates a much easier performance analysis, and provides a baseline against which one might make cost/performance comparisons with other possible inner codes.

The results of a detailed performance analysis for this particular illustrative system are shown in Table III. In order to facilitate analysis of simpler systems, we have itemized the decibels of coding gain among the various components of the encoder/decoder system.

The RS encoder loses 0.14 dB, due to the redundancy which it inserts into the channel bit stream. However, an ultra-high-speed RS decoder could regain more than 5 dB by correcting up to 16 character errors in each block. Unfortunately, no present-day decoder can do that much correction continuously at the 60-MHz channel rate. However, such performance is unnecessary as well as presently unattainable. Since a required decoded bit-error rate is 10^{-9}, most blocks must have only one or two character errors rather than 16 character errors, and so if the decoder has a modest-sized buffer which is able to hold tens of blocks, it can decode blocks containing many character

TABLE III
PROPOSED HIGH-PERFORMANCE RS DECODERS (AT THE COST OF 2 BLOCKS ADDITIONAL LATENCY TIME)

	Gross Throughput = 10 to 15 megabits/sec using low-power Schottky or 40 - 50 megabits/sec using ECL (Net Throughput = gross throughput - redundancy - rerending overhead)			Current Madrid
Elementary Block Length (might be interleaved to depth 2 for increased protection)	$8x2^8$	$10x2^{10}$	$12x2^{12}$	156400
Character Size	8	10?	12	1
Maximum Number of Correctable character errors (limited by decoding time)	8	16	32	4
Redundancy†	$\frac{16+4?}{2^8}$	$\frac{32^+}{2^{10}}$	$\frac{64}{2^{12}}$	$\frac{48}{156400}$
Based on Poisson statistics* Raw error rate against which system attains 1% pause rate	$1.5x10^{-3}$/bit		$6.2x10^{-4}$/bit	$6.4x10^{-8}$
Based on Poisson statistics* Raw error rate against which system attains 0.1% pause rate	$1.0x10^{-3}$/bit		$5.8x10^{-4}$/bit	$6.4x10^{-9}$
Probability of unreadable Data Block (conditional on 1% pause rate and assumptions that media noise, writing noise, and reading noise are of equal severity)	$<10^{-10}$	$<10^{-10}$	$<10^{-10}$	?
Probability of misreading block* (conditional on terrible noise level)	$<10^{-12}$	$<10^{-12}$	$<10^{-12}$	$<4.4x10^{-9}$

† May be increased to provide lower probability of misreading

* May be lowered either by increasing † or by simple system-level reject strategies (See Appendix A)

errors at a rate slower than the channel, then catch up by relying on its ability to decode blocks containing few character errors much faster than the channel rate. Estimates of proposed systems using this strategy have revealed a variety of tradeoffs between decoding cost and decibels gained. The first 3.9 dB can be obtained by a single GF1[tm] RS Decoder with a modest sized buffer. In order to achieve another 1.3-dB coding gain, the decoder costs more than double, as this now requires two GF1[tm] Decoders operating in parallel to meet the speed requirements. It also requires a larger buffer.

The 3.8- to 5.1-dB RS coding gain can be achieved without soft decisions and with only insignificant costs in bandwidth. Additional coding gains can be achieved by using an inner code which trades off transmitter power against those parameters. The (15,11) Hamming encoder costs 1.35 dB in redundancy, but recovers 2.6 dB via single-bit-error corrections, using no soft decisions. If soft decisions are available, the (16,11) Hamming code is preferable. This costs 1.63 dB in redundancy, but its soft-decision decoder, which was described in the previous section of this paper, achieves about 4.6-dB coding gain. In accordance with well-known folk theorems, this is about 2.0 dB better than the comparable code achieves using only hard decisions.

The operating points of the inner decoder for both concatenated systems (using either 1 outer RS Decoder or 2 outer RS Decoders in parallel) are shown in Fig. 5. The total coding gain of the entire system shown in Table III exceeds 8.0 dB. It is able to achieve a 10^{-9} output bit-error rate on a channel whose raw input bit-error rate is worse than 2×10^{-2}.

H. Convolutional Codes

Real arithmetic is the key to success in decoding binary codes on the Gaussian channel, and in many cases, the most effective procedure for doing this real arithmetic is the Viterbi [14] algorithm for decoding convolutional codes of short to moderate constraint lengths. Detailed discussions of the algorithm are widely available[15]-[18].

The Viterbi decoding algorithm is particularly well suited to channels for which

1) the bandwidth limitations suggest code rates between $\frac{1}{2}$ and $\frac{5}{6}$, and the channel input is constrained to be binary;
2) there is no appreciable interference from sources other than white Gaussian noise;
3) a user output bit-error rate of 10^{-5} is satisfactory;
4) the transmission speed requirement lies in the range between tens of kilobits/second and a few megabits/second;
5) the overall system is geared to the transmission of relatively long streams of data, in which traffic control, protocol, and synchronization messages occur relatively sparsely.

There are certain channels, including the space channel, which meet all of the above conditions. There are many other communication environments which fail to meet some or all of the above conditions. Whenever such environments still allow any flexibility of code design, it is wise to evaluate all plausible coding schemes, one of which may be an appropriate modification of the Viterbi algorithm.

We will list some of the tradeoffs which must be explored when each of the above constraints is violated.

1) If sufficient bandwidth is available, higher performance with lower power can be attained via the concatention of orthogonal codes and RS codes. On the other hand, if the bandwidth constraint is extremely tight, then multiple signal levels may be much better than binary inputs and an amplitude-based coding scheme, such as the negacylic codes of [12, ch 9], may be suitable.

2) The Viterbi decoders can be modified to deal with non-Gaussian noises, but so can all of the more sophisticated block decoding schemes. Erasure bursts constitute the form of interference which is the most disadvantageous for both sequentially oriented convolutional decoders and Viterbi decoders [19]. Some of the intrinsic difficulties which convolutional codes have with erasure bursts can be alleviated by interleaving, but when the interference level is high, the interleaved Viterbi decoders are outperformed by sophisticated long block decoding schemes designed to correct many erasures and a few additional quantization errors.

3) The relatively small free distances of convolutional codes implementable with present-day Viterbi decoders cause their performance curves to drop off relatively slowly at high SNR, so they cannot generally achieve the performance needed for the transmission of computer data files. The most widely advocated solution is to concatenate an inner Viterbi decoder with an outer RS code. This may win under some circumstances, but under other conditions it loses to the concatenation of a short inner binary block code and an outer RS code, or to a single unconcatenated RS code.

4) The fact that the complexity of the Viterbi algorithm is inherently exponential in the constraint length leads to a relatively narrow choice of plausible constraint lengths, which translate into the given range of transmission speeds. The fact that the complexity of most block decoding algorithms is a quadratic or cubic function of block length leads to a much wider range of feasible choices.

5) When the system protocols require the transmission of short message blocks, all convolutional systems require flush

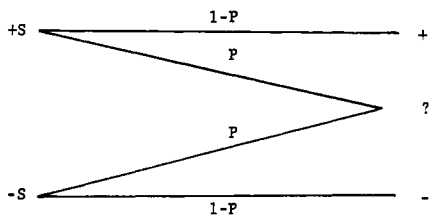
out and restarts. These overhead costs can cause significant deterioration from the nominal performance specifications which the convolutional code actually attains only on infinite-length messages. Block coding schemes encounter similar overhead costs when the available block lengths are mismatched to the system needs, but the infinite-length convolutional code is mismatched to all finite block lengths.

In packetized networks employing indirect routing, an important figure of merit is the total delay encountered at a particular intermediate node when it receives, decodes, reencodes, and retransmits a packet. If the channel noise is truly white, convolutional codes achieve a smaller delay than block codes, because both systems entail the same (negligible) encoding delay while the long block decoder entails a decoding delay somewhat in excess of full block, whereas the convolutional decoder entails a decoding delay consisting only of a few short constraint lengths. On the other hand, if the noise is sufficiently nonwhite that convolutional codes require scrambling, then the comparative delay changes in favor of the long block codes. This is because the scrambling imposes an additional one-packet delay time on both the encoder and the decoder. The descrambling delay delays the convolutional decoder about the same amount as the unscrambled long block decoder, and the scrambling delay creates an additional one-packet delay for the convolutional encoder which is not needed by any encoder for a long block code. Thus the total delay to receive, decode, reencode, and retransmit is significantly greater for the scrambled convolutional code than for a competitive unscrambled long block code.

III. Coding Against Fading or Interference Which Can Be Blanked

A. Classical Analysis of Performance on the Binary Erasure Channel

Many real communication channels have both Gaussian noise and non-Gaussian interference. By making the oversimplifying assumptions that the signal-to-noise level is so strong that the Gaussian noise can be neglected, and that all interference is detected and blanked, we may model the channel as a binary erasure channel (with memory):



The erasure probability may change from bit to bit. In practice, the erasures often occur in bursts. However, we shall first analyze the memoryless binary erasure channel for several reasons.

1) Much of the classical analysis of the memoryless case remains applicable to the case of erasure bursts.

2) The analysis of the memoryless binary erasure channel is much easier than the analysis of almost any other channel, and much of the insight provided by the binary erasure channel carries over to other channel models.

3) New decoding algorithms introduced in 1978–1979 now enable decoders for the binary erasure channel to correct moderate levels of quantization errors and Gaussian noise as well, so that these newer erasure-based decoding algorithms are robust with respect to changes in the channel model which replace the oversimplified binary erasure channel with more complicated and more realistic models.

The effectiveness of linear binary block codes for combating erasures was established by Elias [20] and Epstein [21] in the mid-1950's. Most of the combinatorial arguments date way back to Landsberg [22].

Landsberg computed the probability that e randomly chosen r-dimensional binary vectors are linearly independent. His argument is straightforward: the first vector must be nonzero, which happens with probability $(1 - 2^{-r})$. The second vector must lie outside of the one-dimensional subspace containing 0 and the first vector. This happens with probability $1 - 2^{-(r-1)} \cdots$. The $(j+1)st$ vector must lie outside the j-dimensional subspace spanned by the first j vectors, and this happens with probability $(1 - 2^{-(r-j)})$. Altogether then, the probability that e randomly chosen r-dimensional binary vectors are linearly independent is given by

$$\prod_{j=0}^{e-1} (1 - 2^{-(r-j)}).$$

When $r - e = s$, but $r \to \infty$ and $e \to \infty$, this product becomes

$$Q_s = \prod_{i=s+1}^{\infty} (1 - 2^{-i})$$

whence

$$\ln Q_s = \sum_{i=s+1}^{\infty} \ln (1 - 2^{-i})$$

$$= \sum_{j=1}^{\infty} \frac{-2^{-sj}}{j(2^j - 1)}.$$

Summing this rapidly converging series for $s = 8$ yields

$$Q_8 = 0.9961 \qquad P_8 = 0.00390$$

$$Q_7 = \frac{255}{256} \times Q_8$$

$$= 0.9922 \qquad P_7 = 0.007787$$

$$Q_6 = 0.9845 \qquad P_6 = 0.01554$$

$$Q_5 = 0.9691 \qquad P_5 = 0.03092$$

$$Q_4 = 0.9388 \qquad P_4 = 0.0612$$

$$Q_3 = 0.8801 \qquad P_3 = 0.1199$$

$$Q_2 = 0.7701 \qquad P_2 = 0.2299$$

$$Q_1 = 0.5776 \qquad P_1 = 0.4224$$

$$Q_0 = 0.288789 \qquad P_0 = 0.7112$$

$$Q_{-1} = 0 \qquad P_{-1} = 1.$$

It happens that $P_s < 2^{-s}$ for all s, and the approximation is tight in the sense that $\lim_{s \to \infty} 2^s P_s = 1$.

Q_s may be interpreted as the probability that many random binary equations in s fewer unknowns have a unique solution. If there are more unknowns than equations, then the probability of a unique solution is 0. If there are 5 more equations than unknowns, the probability of a unique solution is Q_5 = 97 percent. If there are the same number of equations and unknowns, the probability of a unique solution is Q_0 = 29 percent; the probability of multiple solutions is P_0 = 71 percent. Since the infinite product converges, any large upper limit would yield the same answer. In other words, a large randomly chosen square binary matrix is invertible with probability about 29 percent. The exact probability is relatively insen-

sitive to the size of the matrix, although it increases slowly with decreasing dimension, reaching $\frac{3}{8}$ for 2×2 matrices and $\frac{1}{2}$ for 1×1 matrices.

Using Landsberg's results, it is not difficult to determine the probability that a random linear block code for the binary erasure channel will fail to decode the block correctly. If the block length is n, the redundancy is r, and the noise creates e erasures, then the code's parity check equations will provide the decoder with r simultaneous linear binary equations in the e unknown binary bits. A decoder which is able to solve simultaneous linear binary equations will find the correct values of all erased bits unless the equations it is attempting to solve have multiple solutions. When there are multiple solutions, we conservatively assume that the decoder always guesses wrong. If the code is random, this occurs with probability P_{r-e}. Since the probability that the noise erases e bits is $\binom{n}{e}p^e q^{n-e}$, we have the following expression for the probability of block decoding error:

$$\text{Prob block error} = \sum_{e=0}^{n} \binom{n}{e} p^e q^{n-e} P_{r-e}.$$

Since $P_{r-e} = 1$ for $e > r$, it is convenient to partition this sum into two parts

$$\text{Prob block errors} = \begin{matrix}\text{Prob \{noise hits weak spot in code\}} \\ + \\ \text{Prob \{noise has large magnitude\}}\end{matrix}$$

where

$$\text{Prob \{noise hits weakspot\}} = \sum_{e=0}^{r} \binom{n}{e} p^e q^{n-e} P_{r-e}$$

$$\text{Prob \{noise magnitude severe\}} = \sum_{e=r+1}^{n} \binom{n}{e} p^e q^{n-e}.$$

Bounding P_{r-e} by the estimate $2^{-r}2^e$, gives

$$2^{-r} \sum_{e=0}^{r} \binom{n}{e} p^e q^{n-e} 2^e.$$

If $p \leqslant r/(2n-r)$, then the maximum summand occurs within the range $0 \leqslant e \leqslant r$. Similar terms with $e > r$ are small compared with terms already in the summations, so we may approximate the sum by

$$\text{Prob \{noise hits weakspot\}} \overset{(n)}{=} 2^{-r} \sum_{e=0}^{n} \binom{n}{e} p^e q^{n-e} 2^e.$$

(Here and subsequently, we use the notation

$$x \overset{(n)}{=} y$$

to mean that $y/n < x < ny$. It means "equal within a factor of n." Since the expression is usually used when both sides are exponential, the factor of n becomes relatively unimportant as n goes to infinity.) On the other hand, if $r/(2n-r) \leqslant p$, then the sum expressing the value of the probability that the noise hits a weakspot is dominated by the term $e = r$. Thus

$$\Pr\left\{\begin{matrix}\text{noise hits weakspot} \\ \text{of random code}\end{matrix}\right\} \overset{(n)}{=} \begin{cases} 2^{-r}(1+p)^n, & \text{if } p \leqslant \dfrac{r}{2n-r} \\ \binom{n}{r}p^r q^{n-r}, & \text{if } \dfrac{r}{2n-r} \leqslant p. \end{cases}$$

Similar arguments give

$$\Pr\{\text{severe noise magnitude}\} \overset{(n)}{=} \begin{cases} \binom{n}{r}p^r q^{n-r}, & \text{if } p < \dfrac{r}{n} \\ 1, & \text{if } \dfrac{r}{n} < p. \end{cases}$$

It is instructive to recompute the first of these probabilities from another point of view. If the noise pattern of e erasures causes the decoder to make an incorrect decision, then there must be a nonzero codeword of weight $w < e$, all of whose nonzero components lie among the e positions. Instead of itemizing by e, let us begin by itemizing with respect to w. Let A_w denote the number of codewords of weight w. The probability that the noise will cover any of these is p^w. Since a decoder error occurs within the union of these events, and since the probability of a union of events is less than the sum of the probabilities, we have the bound

$$\Pr\left\{\begin{matrix}\text{noise hits weakspot} \\ \text{of particular code}\end{matrix}\right\} \leqslant \sum_{w=1}^{r} A_w p^w.$$

For a random linear code, each of the $\binom{n}{w}$ words of weight w satisfies all r independent parity check equations with probability 2^{-r}, so $A_w = 2^{-r}\binom{n}{w}$ and

$$\Pr\left\{\begin{matrix}\text{noise hits weakspot} \\ \text{of random code}\end{matrix}\right\} \leqslant 2^{-r} \sum_{w=1}^{r} \binom{n}{w} p^w$$

$$\overset{(n)}{=} \begin{cases} 2^{-r}(1+p)^n, & \text{if } p \leqslant \dfrac{r}{n-r} \\ 2^{-r}\binom{n}{r}p^r, & \text{if } \dfrac{r}{n-r} < p. \end{cases}$$

For $r/(2n-r) < p$, this bound is weaker than our previous estimate. This is not too surprising, because the union bound ignores both the possibility that a single erasure pattern of weight e may cover several codewords of weights $< w$, and also the fact that codewords of weight w can be covered by erasure patterns of weight $e > r$, which are defined as severe noises rather than noises which hit weak spots in the code.

A more careful analysis which takes account of the latter possibility is as follows: of the $n - w$ positions outside a codeword of weight w, some set of $e - w$ can be covered by erasures and the rest cannot be covered. Hence

$$\text{Prob}\left\{\begin{matrix}\text{noise hits weakspot} \\ \text{of particular code}\end{matrix}\right\} \leqslant \sum_{w=1}^{r} A_w p^w \sum_{e=w}^{r} \binom{n-w}{e-w} \cdot p^{e-w} q^{n-e}.$$

The inner sum may be simplified via

$$\sum_{j=0}^{r-w} \binom{n-w}{j} p^j q^{n-w-j} \overset{(n)}{=} \begin{cases} 1, & \text{if } w \leqslant \dfrac{r-np}{q} \\ \binom{n-w}{r-w}p^{r-w}q^{n-r}, & \text{if } \dfrac{r-np}{q} \leqslant w \end{cases}$$

$$\text{Prob}\left\{\begin{matrix}\text{noise hits weakspot} \\ \text{of particular code}\end{matrix}\right\} \leqslant \sum^{\lfloor (r-np)/q \rfloor} A_w p^w + p^r q^{n-r} \sum_{w=\lfloor (r-np)/q \rfloor}^{r} \binom{n-w}{r-w} A_w.$$

Even for most highly structured (i.e., nonrandom) codes, the number of codewords of weight $w \approx r$ is usually not too different from $\binom{n}{w}2^{-r}$. If the weight distribution satisfies the substantially weaker assumption

$$A_w < \frac{\binom{n}{w}}{\binom{r}{w}}, \quad \text{for } \left\lceil \frac{r - np}{q} \right\rceil \leq w \leq r$$

then

$$\binom{n-w}{r-w} A_w < \binom{n}{r}$$

and

$$p^r q^{n-r} \sum_{w=\lceil (r-np)/q \rceil}^{r} \binom{n-w}{r-w} A_w \overset{(n)}{\leq} p^r q^{n-r} \binom{n}{r}.$$

Hence, the probability of maximum-likelihood decoding error for a given code for the binary erasure channel can generally be conveniently expressed as the sum of two terms. One term, which represents the probability that the noise has severe magnitude, is *independent of the code.* This term is approximately

$$\binom{n}{r} p^r q^{n-r}.$$

If p exceeds a critical threshold, $p_{\text{crit}} = r/(2n - r)$, then *almost every* randomly chosen code attains approximately this error probability.

However, when $p < r/(2n - r)$, then the probability of block decoding is dominated by another term, which depends on the weight distribution of the code

$$\sum_{w \geq 1} A_w p^w.$$

Only the numbers of low-weight codewords are relevant. In particular, the range of summation on w need never extend beyond $w = r$. Under reasonably weak hypotheses, $w \leq (r - np)/q$ may be adequate. In the case of randomly chosen codes, the even weaker limit $w \leq r/2$ yields

$$\sum_{w=1}^{r/2} A_w p^w = 2^{-r} \sum_{w=1}^{r/2} \binom{n}{w} p^w \overset{(n)}{=} 2^{-r}(1+p)^n, \quad \text{if } p \leq \frac{r}{2n - r}$$

which coincides with a previous result. Hence, for some codes, knowledge of $A_1, A_2, \cdots, A_{r/2}$ may be sufficient. For other codes, however, (including the extended Golay code with $r = 12$, $n = 24$, $A_1 = A_2 = \cdots = A_6 = A_7 = 0$), some A_w in the range $r/2 < w < r$ may be required to obtain a meaningful estimate of the probability of max-likelihood block decoding error when the channel is sufficiently quiet (i.e., $p < p_{\text{crit}}$).

Many people find it helpful to visualize the space of received words as a two-dimensional Euclidean plane. Since each decoding region is actually a convex polytope, the common example of decoding regions in two Euclidean dimensions is the lattice tiled with regular hexagons. Each codeword lies at the center of its hexagon, and each hexagon represents the set of points which are closer to one particular received word than to any other received word.

Unfortunately, the hexagonal example presents a misleading notion of the ways in which typical n-dimensional decoding polytopes differ from spheres. The distance from the center to the corner of an n-dimensional Euclidean unit cube is $\sqrt{n/2}$. Hence, when n is large, the smallest sphere which can be circumscribed about a unit cube (or other typical convex polytope) is often much larger than the largest sphere which can be inscribed within it.

Fig. 8 portrays the decoding regions of a particular quaternary code, which has length 60 and 40 information digits. A typical codeword is a point which lies at the center of the decoding region shown. It is surrounded by 20 concentric circles, having radii 1, 2, 3, $\cdots$, 20. Of all points on the circle of radius r, some fraction of them lie within the region of erasure patterns which are correctable, and some other fraction of them do not. At each radius, the sizes of the "stalactite" and the "stalagmite" shown in Fig. 8 have been selected so that the fraction of the circle which lies inside the cave shown in Fig. 8 is equal to the fraction of points which are decodable.

Now imagine that a bat takes a random excursion from the center of the cave shown in Fig. 8. His chances of avoiding collisions with the stalactites or with the cave walls obviously depend on the expected distance away from the center which he travels. If he travels a sufficiently large distance there is a considerable probability that he will hit the cave walls. In this case, which corresponds to a noise level so bad that the code is operating above the critical rate R_{crit}, the sizes of the stalactites and stalagmites are relatively unimportant, because the chances of hitting them are no greater than the chances of hitting the walls. If the bat travels a smaller distance from the center, then the greatest danger of collision will be with the sides of the stalactites and stalagmites. In this case, which corresponds to a noise level slightly below R_{crit}, the number and shapes of stalactites are important, but the distance from the center of the cave to the walls and the distance to the tips of the stalactites are both unimportant. This is the region where the union bound on random codes is tight. Finally, if the bat expects to remain very near the center of the cave, then the most significant collision threat is posed by the tips of the stalactites and stalagmites. This corresponds to a relatively high signal-to-noise level, so that the code is operating below the classical expurgated rate R_x. In this region, the minimum distance of the code is quite important.

In actual fact, the cave corresponding to any decoding region of the 60-dimensional quaternary space should have many, many stalactites and stalagmites. Poetic license is required in order to project 60 quaternary dimensions onto two Euclidean dimensions!

In general, the diameter of the cave depends only on the redundancy of the code, and not on the code's structure. The distance from the center of the code to the tips of the longest stalactites and stalagmites depends on the minimum distance of the code. The sizes and thicknesses of the stalactites and stalagmites depends on the detailed weight structure of the code.

RS codes have erasure decoding regions which are spheres. The stalactites and stalagmites do not exist. Codes which have this desirable property are called "maximum distance separable." Such a code has a minimum distance which exceeds its redundancy by one. Maximum distance separable codes are optimally suited for use on erasure channels in the same sense that "perfect" codes are suited for use on hard-decision channels which have no erasure indicator. Unfortunately, it is known that very few perfect codes exist. On the other hand, RS codes are plentiful.

Unfortunately, no binary code which is maximum distance separable has more than four codewords. Long nontrivial maximum distance separable codes require large alphabet

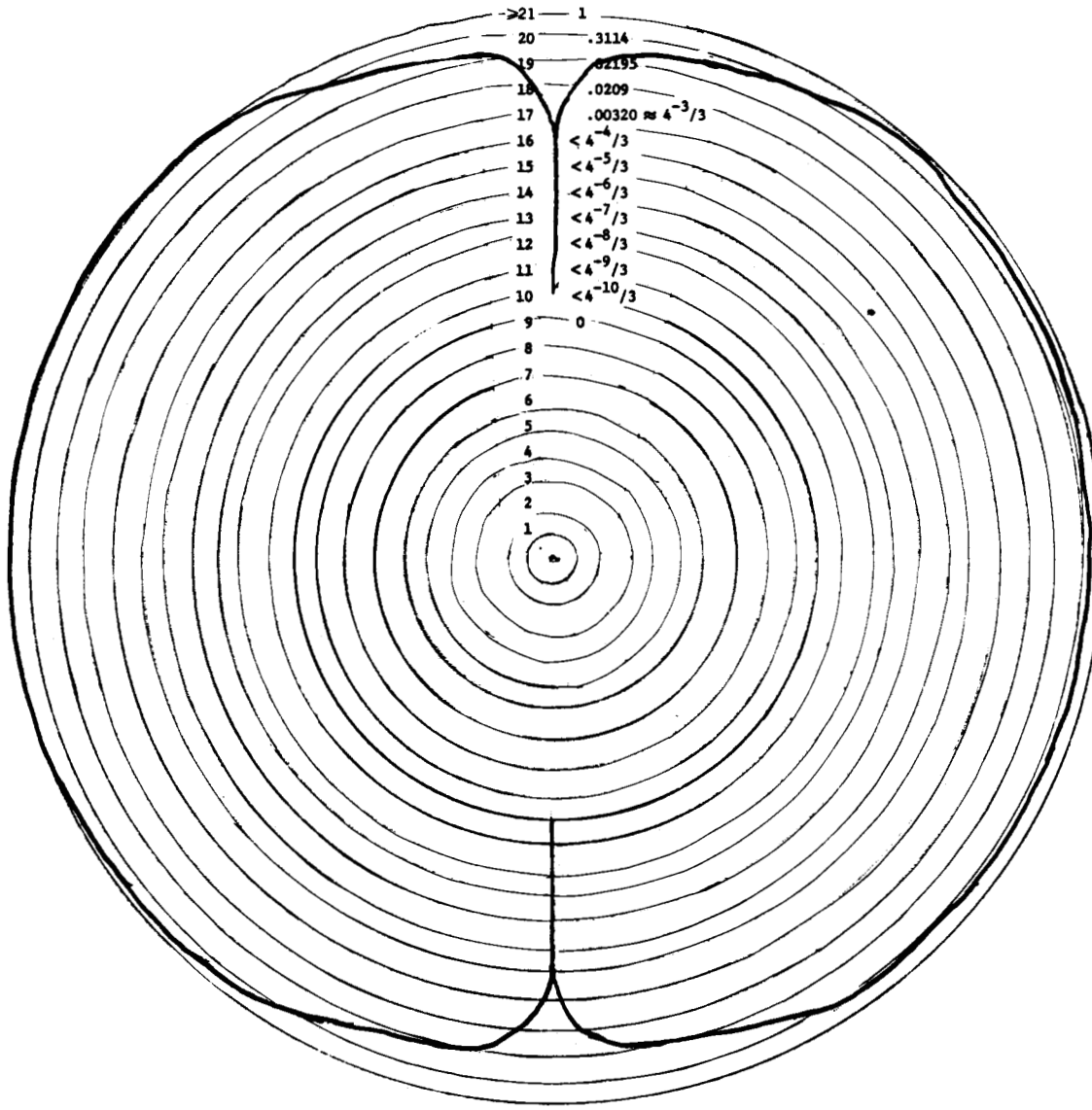


Fig. 8. A decoding region of a (60, 40, 10) code. The size of the stalactites/stalagmites indicates the fraction of erasure patterns of each weight which are not correctable by a (60, 40, 10) code.

sizes. If the channel has insufficient bandwidth to transmit the alphabet characters as orthogonal signals, then the RS code requires a short nonorthogonal inner code, and the performance of the concatenated combination may be less cost effective than other alternatives.

B. Interleaved Viterbi Decoders

Viterbi decoders are easily fooled by bursts. This is due to their relatively short constraint lengths. A block code of comparably short constraint length would be fooled with similar ease. However, in order to balance the hardware costs of the two competing systems, it is usually appropriate to compare Viterbi decoders of constraint lengths 6–10 with much, much longer binary block codes. Since the length of such a block code is typically long compared to the expected burstlengths, an average rate of noise bursts of average lengths creates only an average level of difficulty for the binary block code. Bursts pose no unusual difficulties, and in some cases they actually expedite the decoding algorithm.

Interleaving is often used to try to help Viterbi decoders overcome their burst weaknesses. Conceptually, the interleaver successively assigns each of n consecutive received bits to a different Viterbi decoder, so that no single Viterbi decoder sees more than one bit from any burst of length not exceeding n. In practice, it is easy to schedule things so that all n conceptually different Viterbi decoders are all implemented via a single physical one.

Interleaving increases the entropy of the noise. Philosophically, this is the wrong thing to do. Practically, it may nevertheless be correct in certain cases, because it reduces a new problem (bursts) to one which can be solved by a decoder which has already been extensively tested, and whose performance against white Gaussian noise is well understood.

When a code is interleaved to a depth of n, then each of the n conceptually different decoders sees every nth transmitted bit. If the interference contains not only bursts but periodically repetitive noise sources (e.g., radar transmitters, 60-Hz hum, or commutator arcing of a motor), then there remains the danger that the value of n and the transmission speed of the channel may be such that the interleaver effectively creates some unwanted new bursts in addition to its intended function of removing the original bursts. Some systems employ a complicated

"pseudorandom interleaving" (also known as "scrambling") strategy to avoid this possible danger. Most systems simply attempt to select the transmission speed and the value of n so that the weak spots of the interleaver occur at frequencies which differ from the repetition rates of all of the known noise sources.

Interleaving to a short depth may not be sufficient to destroy the burstiness of the effective channel as seen by an individual Viterbi decoder. Interleaving to a great depth aggravates the overhead problems associated with the short blocks often needed to transmit protocol, routing, and header information.

Any significant depth of interleaving requires a significant increase in the encoding delay, and this may be unacceptable in packet transmission networks which rely on decoding, reencoding, and retransmission at several intermediate nodes.

All of the above factors must be taken into account when the interleaving depth or scrambling plan is chosen. In some cases, the best solution is to avoid scrambling and/or interleaving entirely by using one of the block coding schemes discussed in the next sections.

C. Erasure-Based Decoders; General Notions

It has long been known that block codes used on the binary erasure channel can be decoded by a particularly simple decoding algorithm based on solving simultaneous linear binary equations. This idea was investigated by Elias and Epstein in the 1950's [20], [21]. The amount of computational effort needed to decode a random linear code grows as at most the cube of the number of erasures to be corrected. In some special cases, such as an arbitrary linear cyclic code employed against a single erasure burst, or a specially structured code such as a RS code employed against random patterns of erased characters, the work factor is even smaller. In some cases the work factor is effectively a quadratic function of the number of erasures. All of these cases contrast markedly with the work requirements of both sequential decoders and Viterbi decoders, each of which basically performs a search over all 2^e possibilities to correct a burst of e erased bits.

Since there are few, if any, real channels which can be adequately modeled as a binary erasure channel, erasure-based decoding algorithms are little used. Several studies which have considered this option have assumed that any noises which escaped blanking by the demodulator necessarily caused a block decoding error. In every interesting case, that assumption has implied an inadequate level of performance. In order to be viable on any real channel of current practical interest, a decoding algorithm must be able to correct some modest level of noise in addition to the noises which have been detected and erased by the demodulator. Some nonbinary erasure-correcting decoding algorithms, such as the algorithm for decoding RS codes, are easily augmented to include error-correcting capabilities as well. Historically, binary erasure-correcting algorithms which have been modified to provide additional error-correcting capabilities have offered relatively little additional capabilities at relatively large costs of hardware. The work factors of most of these algorithms grow exponentially with the increasing error-correction capability. However, recent embellishments and improvements in some of these algorithms have made them very attractive for use.

D. The Binary Vector Cruncher

Recent embellishments and improvements in the implementation of some erasure-based decoding algorithms have made them very attractive for use, particularly on relatively slow channels (e.g., 32 kbit/s which allow a relatively small amount of hardware to do a modest amount of computation per received bit. One such system, which we recently designed for Navelex, implements several options of binary codes of rates $\frac{1}{2}$ and $\frac{3}{4}$ and lengths up to 120 bits/block. Soft decision information is accepted on each bit. Analysis shows that the performance is expected to be very close to the performance of a maximum-likelihood decoder for these codes. This performance is comparable to Viterbi decoders against pure white Gaussian noise, and somewhat better than interleaved Viterbi decoders against combinations of Gaussian noise and interference, especially when the interference level is high.

This system will comprise two small cards of electronics. One of them, called a binary vector cruncher (BVC) is designed to perform high-speed manipulations on binary vectors which expedite the solution of simultaneous linear binary equations. The other card, called a "monitor" serves primarily to control the BVC and to provide the flexibility needed to interface the coding with the demodulators of a variety of possible communication systems.

The decoding algorithm which this system employs requires a relatively large amount of microcode, but relatively little running time. Effectively, the received bits are ordered according to increasing reliability, as indicated by the soft decision information. All of the bits which are blanked are erased and lie at the top of this ordered list. These bits, plus a few more relatively unreliable ones, are effectively erased, and the parity check matrix is transformed to eliminate these received bits from further consideration. The BVC then searches to find all error patterns which have at most four additional quantization errors, of which at most one lies beneath the top L_1 bits, at most two lie beneath the top L_2 bits, and at most three lie beneath the top L_3 bits. All such candidate error patterns are then scored by the monitor to determine the winner. The innovative organization of the BVC allows it to find all such candidate error patterns about 30 times as fast as more naive "direct" implementations.

Although the speed of the BVC is sufficient to attain virtual maximum-likelihood decoding performance of slightly longer binary codes at 32 kbits, the maximum length of 120 was chosen because the noise environment in which this system is designed to operate is known to contain a powerful interference generator which creates periodic "beeps." Each beep is an erasure burst, and the difference between the starting times of consecutive beeps is virtually guaranteed to be between 120 and 135 bits. Thus the choice of a block length of 120 bits ensures that there can never be more than on such beep per block. For this reason, the performance of this coding system against this particular interference source is decidedly better than any coding system can attain against a memoryless interference source which erases bits independently with the same probability. As the noise source becomes less structured, the performance of the BVC-based decoder degrades to the same level which any maximum-likelihood decoder attains against memoryless interference. This noise source provides a remarkable example of how interleaving degrades performance.

E. Chase Decoders

In 1973, David Chase [23] introduced a significant variation on the theme of decoding binary block codes with soft decision information by erasing the least reliable bits. As before, one starts by ordering the received bits according to their reliabilities, as indicated by the soft decision information. As before,

one next picks out a small set of the 5 or 10 least reliable bits for special treatment. However, instead of erasing them, Chase suggested that the values of these bits be guessed! As there are only 5 to 10 such bits, the number of guesses over which the decoder must search is only 2^5–2^{10}. For each such guess, a decoding of the entire block is attempted, and the decoded error patterns which emerge from any successes are scored.

The guessing or searching procedure is called "Chasing". Most investigations of Chasing have assumed the use of a binary Bose–Chaudhuri–Hocquenghem (BCH) code. It is generally assumed that the algebraic decoding procedure begins afresh for each new guess of the values of the bits in the Chased set. In fact, it is possible to implement Chase decoding in a way which moves the Chasing into an innermore loop, and thereby attains a modest speedup factor.

The most obvious weakness of the Chase algorithm is that, even though the proportionality constant can be reduced by algorithmic refinements, the work factor is inherently exponential in the size of the set of bits which is Chased. However, similar exponential growth with respect to some key parameter is a characteristic of all known decoding algorithms for modest rate binary codes on the Gaussian channel. In practice, this constrains all such parameters to relatively small values, typically less than 10. The less obvious weakness of the Chase algorithm is that, except for the final scoring, it uses the soft-decision analog information only to select the bits in the Chase subset.

Chasing provides a general method by which the performance of almost any decoding algorithm for block codes with soft-decision information can be improved at the cost of additional time. If a decoder runs 4 or 8 times as fast as needed, then its performance can be improved by Chasing over another 2 or 3 bits. In most present situations, there usually seem to be more desirable features which can be bought with the excess time, so Chasing is presently little used. However, as hardware speeds appear to be increasing faster than many communication channels, Chasing remains a strong candidate for inclusion in each new decoder design.

IV. Coding Against Digital Errors

As discussed in the introduction, noise is best considered to be digital whenever the acquisition of analog soft-decision information is uneconomical. With both speeds and economics changing rapidly, it is not yet possible to predict which channels will be properly considered as digital in the long-term future.

At present, most digital channels are memories rather than communication links. If it were not for the possibility of someone reading this paper 10 years or more hence, this section might have more appropriately been titled, "Coding for Secondary Memory Systems." Present-day memory channels differ from most communication channels in a number of respects besides the digital nature of the noise. Present memory speeds are typically at least 20 Mbit/s. Reliability requirements are usually much higher for memories than for communication systems. Typical traffic on a communication system might be digitized voice, for which a bit-error rate of 10^{-5} or even 10^{-4} may be acceptable. Typical data in a computer memory is likely to be a large program, for which an error rate of 10^{-13} or less may not be an unreasonable requirement.

Many decoders used in communication systems have only one type of error mode: the decoder delivers incorrect data to the user and alleges it to be correct. In computer memory systems, this type of failure is called a *catastrophic error*. In secondary memory systems, such as disks and tapes, this type of failure is so undesirable that there is an industry-wide reluctance to place any specification on how often it is allowed to occur. When the designer of a new system tactfully tries to broach this subject by asking, for comparative purposes, how often catastrophic failures occur in the present system, the most common answer is either the naive "never," or an unquantitative response like "We're not aware of any such problems in the present system."

The next most serious mode of decoding failure, *persistent failure*, occurs when a block of data cannot be decoded despite numerous attempts to reread it. At each attempt, the decoder avoids catastrophic error by observing that the noise is so severe, relative to the power of the code, that it refuses to decode.

The most benign type of decoding failure occurs when the first attempt to decode a block of noisy data fails, and the system must then attempt to reread the data. Since many reading errors are at least partly due to problems such as poor alignment of the reading head with the data track, there may be an excellent chance of success on a subsequent reading attempt.

An even more routine type of decoding "failure" occurs when the decoder is unable to keep up with the data channel in real time. Even though none of the data needs to be reread, an atypically noisy block of data may cause the decoder to ask the reading system to *pause* for a brief interval during which the decoder corrects an atypically noisy block.

The comparative evaluation of various error correction and detection (ECAD) schemes must take all of the above performance measures into account. As we shall show in Appendix B, the probability of catastrophic error can be reduced to any desired level simply by annexing more redundancy. The implementation is straightforward and inexpensive.

A more critical design specification is the frequency of persistent failures. How often persistent failures will occur depends both on the coding system and on certain details about the nature of the noise. Fortunately, some features of secondary memory noise are common to many different technologies, even though the appropriate quantitative values of the distributions and parameters vary from application to application.

The basic steps in the operation of secondary memories are as follows:

```
FABRICATE media;
RECORD tracking information;
WRITE data;
READ data.
```

Because of the large differences between latency times and bit throughput times, data is often written and read in relatively long records. Thus the "WRITE data" and "READ data" commands above may be expanded into the following do-loops:

```
      Do 20 i = 1, n
20    Write ith data bit
      Do 10 i = 1, n
10    Read ith data bit.
```

The parameter n, which is called the *record length* or *block length* may be either fixed or variable. However, even in systems which allow n to vary, there is often a small set of preferred values of n, and some loss of performance occurs when n differs from the preferred values. Common preferred values of n in currently popular disk storage devices include

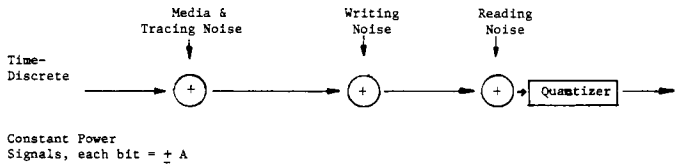


Fig. 9. A disk or tape storage "channel."

the "Winchester" $n = 70368$ and the "Madrid" $n = 156400$. These numbers correspond to the maximum number of bits which those devices can write or read contiguously before it becomes necessary to suspend reading or writing long enough to jump to a different track.

The obvious (but rarely seen) model of the "disk channel" is shown in Fig. 9.

Although any real system encounters noise from a plethora of sources, we find it fruitful to lump all such noise sources into three noise sources shown in Fig. 9.

A key feature of this model is that the effects of the various noises are additive. One might therefore be tempted either to insist on analog measurements of their individual properties, or to lump all three sources and the output quantizer together to consider the entire channel as a single entity. Both of these approaches are commonly pursued. But each provides only limited insights into the nature of the overall channel. Large amounts of analog data cannot be conveniently analyzed and recorded. Small amounts of analog data generally prove quite useful for fine-tuning analog components at adjacent places within the system, and indeed the collection and utilization of such data is now widespread. Despite its importance in optimizing the performance of subsystems, analog data yields relatively little useful insight into the nature of the system as a whole.

Digital data on the rates and statistics of errors in the overall system prove equally difficult to interpret. Some observers are convinced that the predominant cause of errors is "white Gaussian noise"; others are convinced that errors tend to cluster in bursts. Those who believe in bursts adhere to a wide range of feared burstlengths. Some believe that virtually all causes of error that strike more than a single bit cause at most a burst of length two, which affects two adjacent bits. Others believe that systems must be designed to cope with single bursts of lengths $\leqslant b$, where b is some rigid parameter agreed upon by the believer and his friends. Popular critical values of b include 2, 3, 4, 5, 10, 13, and 20.

From a sociological point of view, each of these contradictory viewpoints may be "correct." In any particular environment, the need to please the customer (or the boss) or to match the actual or perceived performance of some other system may indeed make any of the above mentioned criteria the overriding short-term engineering design goal.

Fortunately, the issue of the "critical" maximum burstlength plays only a minor role in the design of multiple-burst-correcting codes. This is because RS codes currently enjoy a large cost/performance advantage over all other competitors, and they include a "built-in" capability to correct moderately long bursts. For example, if one needs a multiple-burst-correcting code of length 2^{11} bits, it makes little difference whether one wishes to correct bursts of length 3 or 4 or 5 or 6 or 7 or 8 or 9, for in each of these cases, the two leading contenders are the RS code and the doubly interleaved RS code, both using an 8-bit character. These codes remain strong contenders at other burstlengths, although other candidates may then also need to be considered. BCH codes are appropriate for independent bit errors (i.e., bursts of length 1), and other RS codes have some advantages if longer bursts are common.

In the author's opinion, the reason that no single interpretation of the experimental data on disk errors has won wide acceptance is that the data has not been taken in a manner designed to expose the individual noise sources shown in Fig. 9. In order to isolate these separate sources, it is not necessary to resort to analog methods. In fact, it is possible to obtain a significant amount of statistical data about the behavior of each of the noise sources shown in Fig. 9 by performing only system level experiments on errors. The basic steps of such an experiment are identical to those listed above; one generates and records some pseudorandom data sequence, then rereads it, and compares the read-back version with a regenerated copy of the original in order to see where errors have occurred. However, one must be careful where to put the do-loops in the data-collecting program! In order to obtain statistical data about the behavior of media noise and writing noise in the absence of reading noise, we may simply reread the same record a large number of times, and thereby average out the effects of the reading noise. The proposed program to gather statistical information on each of the error sources shown in Fig. 9 is as follows:

```
      Do 10 n3 = 1, m3
      Fabricate media and record tracking information;
      Do 10 n2 = 1, m2
      Do 20 i = 1, n
20    Write ith data bit
      Do 10 n1 = 1, m1
      Do 10 i = 1, n,
      Read ith data bit,
10    { Compare with regenerated copy of original, and re-
      { cord all indices of each error.
```

We expect some interesting features of the data to escape notice if $m1 < 1000$. For best results, the values of $m1$, $m2$, and $m3$ should be chosen as large as feasible, and with $m1 > m2 \gg m3$.

Let us now imagine the raw data which might be obtained from such an experiment plotted in a large binary matrix whose ith row represents the error pattern observed on the ith reading of the same data. Most of the entries in this matrix are zero. Most conventional error-measuring experiments study each row of this matrix in isolation from the others. However, we expect the most interesting data to be obtained by summing along the columns of this matrix. Each column sum gives a count of the number of times that a particular bit was read incorrectly. Suppose, for example, that after reading the same record one thousand times, we observed the following subsequence of error counts: 0, 0, 0, 0, 2, 6, 8, 10, 9, 10, 9, 5, 0, 0, 0, 0. Such data would provide conclusive evidence of a "burst" of bits where the data was "weakly written." Yet no individual bit within this burst was misread more than 1 percent of the time. In fact, the entire burst will usually be read correctly, and a less persistent data collecting experiment might fail even to discover its existence. Whether this weak "burst" is due to the media noise or to the writing noise (or to some combination of both) would not yet be evident from the experimental data gathered above. However, such a distinction could probably be made by data obtained from the next level of the experiment. By writing many times in the same locations, one can distinguish between writing noise which varies

each time data is written, and the media noise which remains physically attached to the particular geographical locations on the disk.

Even after completing the entire experiment prescribed above, there would still be no statistical way to distinguish between media noise due to manufacturing irregularities and "media noise" due to weakly recorded tracking information. However, unless one contemplates rewriting the tracking information in some operational situation, there is no way to make use of such information anyway. R. M. Fano's [24] fundamental tenet of communications engineering asserts that the channel should be defined as those parts of the system which the designer is unable or unwilling to change. The decomposition of the disk channel shown in Fig. 9 provides an application of this tenet because the system can, in fact, be broken at each of the intermediate points shown. Rereading several times provides independent samples of the reading noise while the writing noise and the media noise remain fixed. Rewriting provides a new sample of the writing noise while leaving the media noise fixed. "Defect-skipping" tactics provide a direct means of modifying the media noise distribution.

Despite the limited amounts of available quantitative data, there is general industry-wide acceptance of the qualitative behavior of each of the noise sources shown in Fig. 9. Writing and reading noise are so fundamentally similar that it seems appropriate to use identical mathematical models for each of these two noise sources. The noise contains an electrical component of uncertain spectrum, quite possibly white, and a mechanical component of colored spectrum, which is strongest at frequencies corresponding to several bit-lengths duration. This component of the noise presumably results from the variations in the distance between the disk head and the media. If one were to magnify a modern disk head to the size of an airplane and the disk surface to a runway, the airplane would be designed to fly at a constant altitude only a few feet above the ground! Just as airplanes sometimes encounter turbulence, so do disk heads. While the disk technology "appropriately scaled" is more advanced than the commercial aircraft industry, it is still possible for the distance between the head and the media to vary by a significant relative fraction. When the disk head flies up to some abnormally high distance above the media, the reading or writing occurring during this excursion is relatively weaker than when the head maintains its usual close elevation.

Media noise may contain one "spike" component, due to impurities, which have the effect of introducing such a large noise that no data bits can ever be written at those geographic locations. However, the media noise distribution also contains a continuous component. In some manufacturing technologies, this may be due to the fact that the garnet is not always precisely evenly sprayed onto the disks. There are some spots where the garnet is thinner than usual, and while it is possible to write and read on those locations, the signal there will not be as strong as it is elsewhere, and such a location will be more prone to errors than most locations are.

According to this description and the model of Fig. 1, it is generally futile to attempt to distinguish between errors caused by misreading, errors caused by miswriting, and errors caused by media defects. In some rare cases, such as locations containing impurities, the noise may be of such a nature that this attempt to assign the blame succeeds. However, in most cases the legalistic attempt to determine the culpable noise source cannot succeed because no one or two components of the noise would be sufficient to push the signal across the threshold without the addition of the other components. In physical terms, most errors occur at locations where the garnet is abnormally thin on data which was written when the writing head was abnormally high. Even then, such data is usually read correctly, although it is occasionally misread because it lacks sufficient strength to withstand large reading noise as well.

The current "industry-standard" error-detection and correction strategy is shown in Fig. 10. As each block of 156 400 bits is read from the disk, its 48-bit syndrome is computed. Very shortly after the last bit of the block is received (i.e., within a few hundred nanoseconds), the decoder announces whether or not it is "finished" (i.e., whether or not it thinks the block is error free). With probability greater than 0.99, the decoder correctly announces that the received block is error free, and the system is then free to continue reading the next block without any pause.

It is also possible for the block to contain an unusually perverse error pattern, so that the decoder fails to detect that anything is wrong and bad data is passed on to the user with the certification that it is allegedly correct, but the probability of this is very small. Generally, however, when errors are present, they are detected at this point and the decoder forces the system to pause while it attempts to correct a single burst. The system is then shut down for a pause interval whose duration is very close to the time required to read a block of 156 400 bits. During the pause interval, the decoder attempts to locate and correct a single burst of length $\leqslant 4$ bits. With high probability it really succeeds. With probability about 10^{-14}, it decodes in error, inserting an additional burst of errors into a block already containing many more errors, and then passes this garbled data block on to the user with the allegation that it is corrected. With higher probability, perhaps 10^{-8}, the decoder fails to decode the block and the system pause is then extended while rereads are attempted. If decoding failures persist, then the rereading pause persists until after 27 unsuccessful rereading and decoding attempts fail, the block is declared "unreadable."

According to the model of Fig. 9, rereading the block gains a new sample of reading noise, but the same media noise and the same writing noise remain present in the reread block as were present in the original block. Hence, after each successive decoding failure, the *a posteriori* probability of severe media and writing noises increases, and the conditional probability of successful decoding decreases with each repetition. However, the dominant form of persistent failure is *not* because the media noise and the writing noise have jointly conspired to create an unreadable "double burst" of errors, or a long single burst of errors in which the specific error pattern repeats, but rather that the media noise and the writing noise have created a sufficiently large number of weaknesses in the block that each attempt to read it entirely without error (except for a single short burst) has very little chance of success.

Thus, when a block is reread persistently, there are some changes in the reading noise each time the block is reread. Since some of the weak bits are read differently from one rereading attempt to the next, the probability of persistent decoding failure can be considerably reduced by remembering the data obtained from each of several rereading attempts. If the block is read an odd number of times denoted by d, and a majority vote is taken among these d samples of each bit, then the majority vote will be wrong only if the bit was misread of $(d+1)/2$ or more different rereadings. If the reading noise were a unit Gaussian random variable, then the probability

that the magnitude of one of its samples exceeds x would be given by the normal distribution function, $ndf(x) = \int_x^\infty \exp(-t^2/2)\,dt/\sqrt{2\pi}$, and the probability that a majority of d samples exceeds x would be given by

$$\sum_{i \geq (d+1)/2} \binom{d}{i} ndf^i(x)\,(1 - ndf(x))^{d-i}.$$

For a large and moderate to large d, this sum is approximated by

$$\sim \frac{2^{d+1}}{\sqrt{2\pi d}}\, ndf\,(d + 1/2)\,(x)$$

$$\sim \frac{2(d + 3/2)}{\sqrt{2\pi d}} \left(ndf\, \sqrt{\frac{d+1}{2}}\, x\right).$$

In other words, rereading d times and taking a majority vote has an effect similar to reducing the power of the reading noise source by a factor of about $2/(d + 2)$; memoryless rereading merely obtains another sample of reading noise with the same power as the original sample. Thus, for example, if the media noise, the writing noise, and the reading noise were all Gaussian with the same power N, then the total noise of the channel shown in Fig. 9 would be $3N$. However, the channel obtained by rereading 3 times and taking a majority vote has noise power reduced to $2\frac{1}{2}N$; the channel obtained by rereading 5 times and taking a majority vote has noise power reduced to about $2\frac{1}{3}N$; etc. If the tails of the media noise distribution have been truncated by defect skipping, then the relative advantage obtained by rereading and taking a majority vote on each bit is even greater. In the limit of negligible media noise, a large number of rereads should succeed in reducing the overall channel noise power by almost a factor of 2.

Using wn bits of memory, it is possible to obtain exact results of majority votes among d samples of each of n bits whenever

$$d \leq n(2^{w+1} - 3).$$

In particular, a memory sufficient to buffer 2 blocks is sufficient to accumulate exact results of all majority votes among 5 readings; a memory of 3 blocks is sufficient to accumulate exact results of all majority votes among 13 readings. Even more readings can be accumulated if rare errors in vote counting are permitted. In practice, of course, we expect such a large number of readings to be required only very rarely.

Even though the strategy of majority voting offers a very powerful technique for combatting the potential problem of persistent decoding failure, it is not currently used in the industry-standard EDAC strategy. This is partly because buffer memory (RAM's or delay lines) were not so cheap when this industry standard was designed as they are now, and partly because the EDAC strategy shown in Fig. 10 is adequate for current disks, whose raw error rates are very low.

In some sense, current disks have been overengineered to make the raw error rate so low that they are able to attain adequate system-level performance using only the relatively primitive single-short-burst-correcting EDAC shown in Fig. 10. Most of these same system design goals can be achieved with much poorer raw error rate by using the more sophisticated EDAC strategy indicated in Fig. 11. As shown in Table III, current algebraic coding technology can handle a raw error rate of up to about 5×10^{-4} at interesting throughput rates using only modest amounts of decoding hardware and very

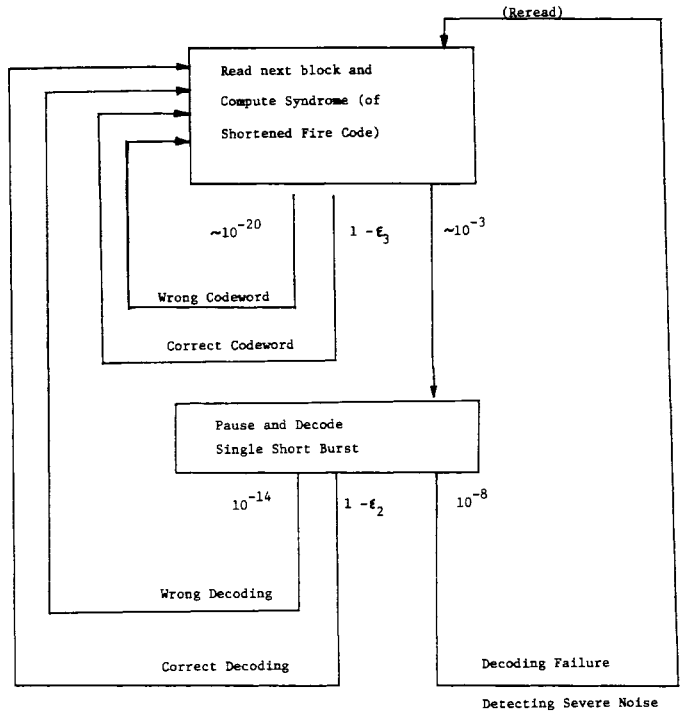


Fig. 10. 1977 disk industry-standard EDAC strategy.

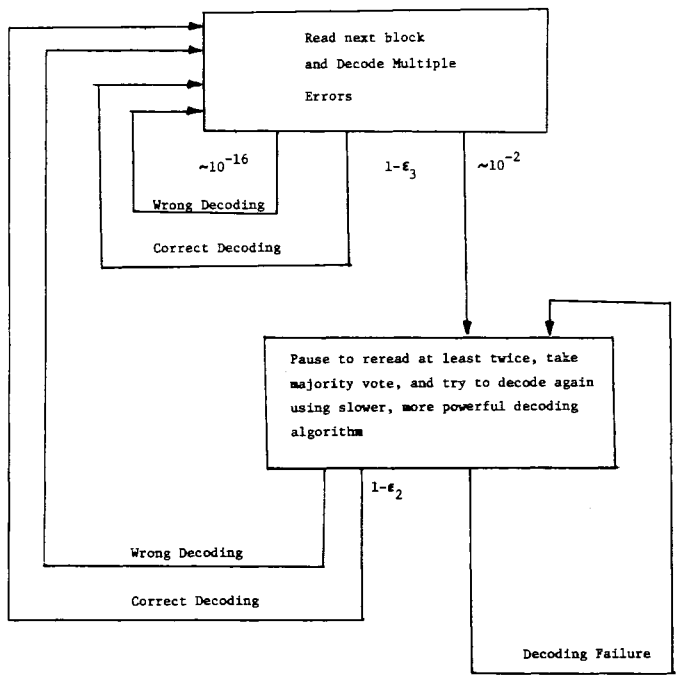


Fig. 11. Proposed high-performance EDAC strategy.

little redundancy. Although the relationship between raw error rates and packing densities is not precisely known, some projections have indicated that weakening the raw error rate requirement to 10^{-3} might allow an immediate increase in packing density of about 30 percent. A more conservative raw error rate specification, such as 10^{-4} yields a slightly lower increase in packing density. However, either of these increases can surely be further improved by redesign of the analog modulation portions of the recording system. In the past these subsystems have been designed to achieve raw

error rates of about 10^{-9}, and they are not fine tuned to operate at higher densities with an EDAC system that is so strong that only 10^{-4} or 10^{-3} raw error rate is necessary.

The major system change needed to implement to sophisticated EDAC strategy shown in Fig. 11 is an increase in *latency*. Instead of delivering the raw (uncorrected) data from the disk directly to the user, the sophisticated decoder would hold all data coming from the disk in an internal buffer. For most purposes, this buffer can be viewed as a delay line approximately two blocks long, so that the user would typically receive corrected data about two block times after the noisy data was first read from the disk. Since all correction would be done within the decoder, the user would never be bothered with the need to make corrections on incorrectly received data. However, the user might not get his data as quickly as he does with the 1977 industry standard EDAC, but the throughput would be improved.

In both Fig. 10 and Fig. 11, there are two operating costs which reduce the raw (gross) throughput rate to the net throughput rate: redundancy and pauses. The redundancy is a constant overhead which is encountered on every block independent of the noises; the pauses are additional delays caused by the decoding system's need for additional time to deal with a relatively severe block of noise. In both figures, at large block lengths and interesting raw error rates, the pauses cause more loss of throughput than the redundancy.

Both systems are robust in the sense that they fail gracefully. If operated in environments with increasingly severe noise levels, neither system will allow the probability of (catastrophic) decoding error to rise above some very small fixed level, but each will produce less and less throughput due to more and more pauses of longer and longer durations. Calculated levels of raw error-rates against which the proposed systems must pause on 10^{-2} or 10^{-3} of the blocks are shown in Table III. In fact, the degradation in throughput will be several times higher, because each pause may last for several block-times (the average number of which should be somewhere between 2 and 3).

Even if the increased packing density attainable via the increased tolerance of raw error rate were no greater than the increased overhead due to redundancy and decoding pauses, one might argue that the overall system is improved because one could attain the same average throughput rate with a new disk of larger logical size. However, the calculations suggest that the sophisticated EDAC should also allow increased throughput as well, because an increase of 20 percent, 30 percent, or more in packing density apparently requires less than 1 percent in redundancy and less than 5 percent in pauses. Since the sophisticated EDAC hardware is cheap to manufacture, the other significant costs are primarily developmental costs rather than production costs: detailed design of the decoding system and modification to the operating system to provide the additional latency time which sophisticated EDAC requires.

Appendix A—Notes on the Implementation of Long RS Codes and Long BCH Codes with High Speeds and High Rates

A. Redundancy and Burst-Correcting Capabilities

All of the codes we consider in these notes are based on fields of characteristic two. Each consecutive set of c bits may be grouped together to form a *character*. Since there are c bits/character, the total alphabet consists of $2^c = q$ different characters.

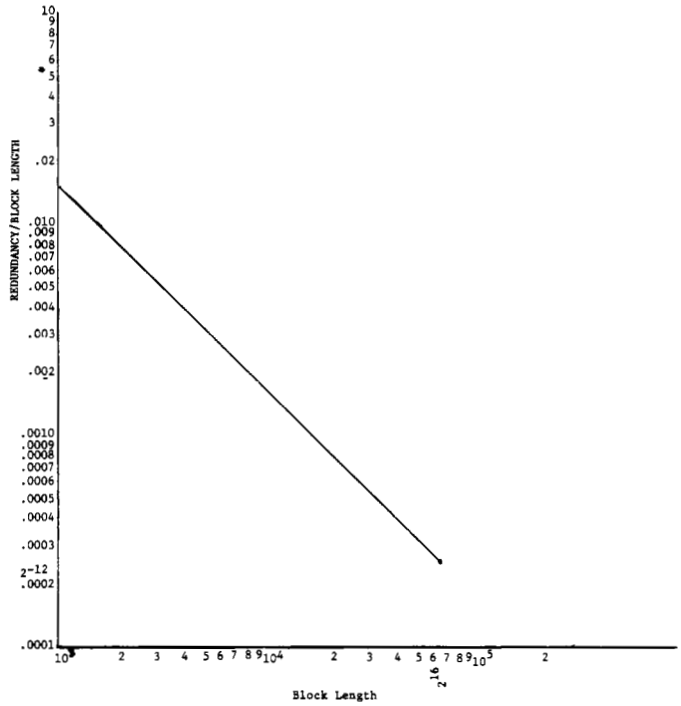


Fig. 12. Single error-correcting BCH code of length $2^{16} - 1$.

For any value of m, we may construct a primitive extended RS, BCH, [25]-[26] or Goppa code [27]-[31] which has length $N = q^m$ characters. Some of these characters are information characters; some are check characters. Altogether, such a code will have a length of $c \cdot q^m = c \cdot 2^{cm}$ bits. We denote the number of redundant bits by the letter r.

The primitive binary BCH codes all have 1 bit/character. RS codes all have several bits per character. The precise number of bits per RS character increases with the code length. For any given code length, the RS code generally has at least as many bits per character as any other code with comparable error-correction capability. Among the many codes which have character sizes intermediate between the primitive binary BCH codes and the RS codes are the quanternary and octal BCH codes.

For each of the primitive long code lengths considered here, at high code rates the number of redundant bits, r, is directly proportional to the error-correction capability, t. (This rule of proportionality fails to hold when t becomes sufficiently large, but for code lengths $\geqslant 10^3$, the proportionality ceases to hold only when t is considerably larger than 8).

Starting from any of the primitive codes, we may obtain another code of any shorter length by simply not using some of the information bits. The *shortened* code will have the same number of redundant bits and the same error-correction capability as the parent code.

Fig. 12 presents a plot of the redundancy divided by the product of the block length and the error-correction capability for all of the codes which can be obtained from the primitive binary BCH code of length 2^{16}. Since this quotient is inversely proportional to the block length, the graph forms a straight line of slope -1 on full logarithmic graph paper. The line terminates at the point $n = 2^{16}$, $r/Nt = 2^{-12}$. Since this termination point represents the parent code, it may be conventionally considered the beginning of the straight line rather than the end. Shortening bits from the parent BCH code of length 2^{16} then corresponds to moving leftward along the line

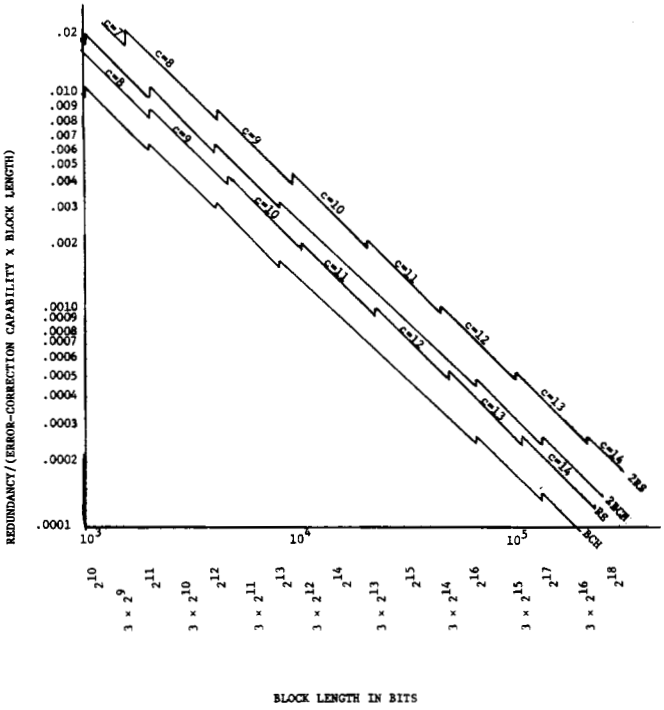


Fig. 13.

shown in Fig. 12. Even after half of the bits of the original code have been shortened, one may continue to shorten even more. However, if one needs a code of length less than 2^{15}, then one can obtain the same error-correction capability for lower redundancy by switching to a new parent BCH code.

By selecting the optimum parent BCH code for each possible length between 10^3 and 2×10^5, we obtain the lower curve shown in Fig. 13. This lightening like curve is piecewise linear. Each segment begins with a primitive binary BCH code and moves backward at slope -1 until it becomes possible to obtain a small decrease in redundancy by switching to a new parent. At such a length, the curve drops vertically to the next segment.

The curve labeled 2 RS in Fig. 13 is obtained by using RS parents instead of BCH parents. To a rough approximation, this RS curve lies above the BCH curve by a factor of about 1.4. This means that the best shortened RS code will have about 1.4 times as much redundancy as the best shortened primitive binary BCH code of the same length and same error-correction capability. However, the RS code is able to correct t character errors and the BCH code is able only to correct t bit errors. Thus the RS codes offer more protection against bursts than the BCH codes.

If the character size is c, and if we assume that burst errors have random phase, then over half of the bursts of length less than $c/2$ will lie within the same character. Only $1/c$ of the bursts of length 2 will strike more than one character. However, each burst of length 2 will be treated as two separate errors by the binary BCH code.

Interleaving provides a common method for increasing the immunity of codes to bursts. The interleaved code consists of several copies of the original code, which are multiplexed together. For example, binary BCH codewords $A_1A_2A_3A_4\cdots$, $B_1B_2B_3B_4\cdots$, $C_1C_2C_3C_4\cdots$, might be interleaved to form the codeword $A_1B_1C_1A_2B_2C_2A_3\cdots$ In this case, the depth of interleaving is 3. The length of the interleaved code is 3 times the redundancy of the component codes. However, since all errors might strike the same component codeword, the random-error-correction capability of the interleaved code is no greater than the random-error-correction capability of the component codes.

Nonbinary codes should be interleaved character by character rather than bit by bit. Thus, if $A_1A_2A_3A_4A_5$ and $B_1B_2B_3B_4B_5$ are the first characters of the two RS codewords, then the word formed by interleaving these two RS codewords would be

$$A_1A_2A_3A_4A_5B_1B_2B_3B_4B_5A_6A_7A_8A_9A_{10}B_6\cdots.$$

This interleaved code has the property that very short bursts are still likely to strike only a single character of the interleaved code.

The parameters for depth-2 interleaved binary BCH codes and depth-2 interleaved RS codes are the higher two curves plotted in Fig. 13. These curves may be obtained by shifting the lower curves to the right by a factor of 2.

One can draw similar plots for other families of codes, but the results are all disappointing. The curve of binary BCH codes interleaved to depth 3 lies very close to the curve for RS codes interleaved to depth 2. (In fact, many of the vertical lines on each of these curves cross the lines of slope -1 on the other curve.) However, the burst-correcting capabilities of the depth-2 interleaved RS code are clearly superior to the burst-correcting capabilities of the depth-3 interleaved binary BCH code with the same random-error-correction capabilities. Similarly, the curve for quaternary BCH codes lies close to the curve for the (uninterleaved) RS codes, and the RS codes have superior burst-correcting properties. Other families of codes, including octal codes and hexadecimal codes, have also been considered, but all of them except the binary BCH codes require even more redundancy than the RS code with the same length, and so we omit the details from these notes.

We conclude with a comparison of the burst-correcting capabilities of the four classes of codes whose parameters are plotted in Fig. 13:

Depth-2 Interleaved RS Codes (2 RS): Any burst of length $c + 1$ or less counts as a single error. These codes are therefore capable of correcting any pattern of up to t bursts, assuming that each of the t bursts have lengths no greater than $c + 1$.

Depth-2 Interleaved BCH Codes (2 BCH): The redundancy of such a code is only about 0.7 times as much as the redundancy of the depth-2 interleaved RS code of the same length and same random-error-correction capability, but these codes are able to correct all patterns of t bursts only when each burst has length 1 or 2. All bursts of length 3 or 4 are counted as two separate errors.

Uninterleaved RS Codes (RS): The redundancy is only about 0.5 times as much as the redundancy of the depth-2 interleaved RS code of the same length and same random-error-correction capability, but these codes are only able to correct t character errors. Any burst of length $c + 1$ counts as exactly two character errors. Most bursts of length less than $c/2$ count as only one error. The fraction $1/c$ of the possible bursts of length 2 count as two character errors; the fraction $(c - 1)/c$ of the possible bursts of length 2 count as single character errors.

Uninterleaved Binary BCH Codes (BCH): The redundancy is about 0.7 times as much as the redundancy of the RS code of the same length and same random-error-correction capability, but these codes have no burst capabilities whatsoever. Every burst of length two counts as two separate errors.

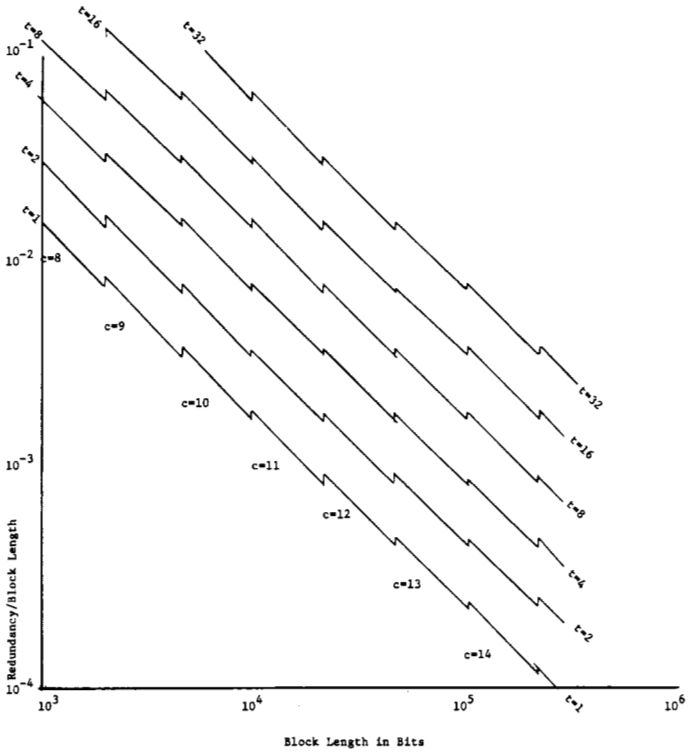


Fig. 14. Uninterleaved RS codes.

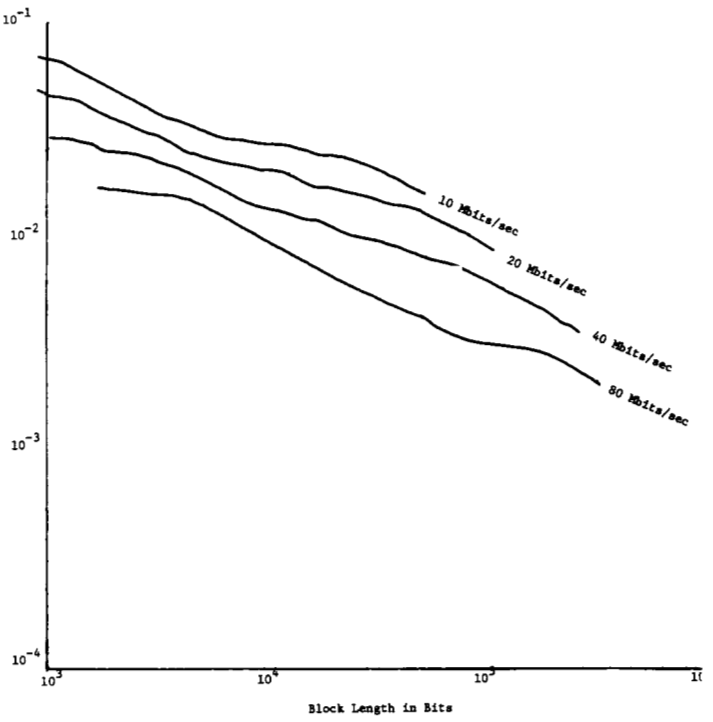


Fig. 15. Estimated maximum throughput speed of GF1[tm] Decoder (ECL) (Uninterleaved RS codes).

Decoding algorithms: Although the fundamental algorithms for decoding RS codes and BCH codes were all discovered in the early- to mid-1960's, [12], [32] there have been many recent expositions, variations, embellishments, and generalizations [33]-[35]. Novel algorithms [36]-[39] have reduced the number of Galois field multiplications needed to decode long algebraic codes of low to moderate rates, but these algorithms are not used in the best current implementations, partly because high-rate codes are more attractive and partly because the number of Galois field multiplications is a poor measure of either running time or hardware complexity. We have designed and built various models of a special-purpose microprogrammable Galois field computer, called the GF1[tm] Decoder. It decodes many RS codes more than twice as quickly as any of several competing decoders which employ three to five times as much hardware. The cost-performance advantages of the GF1[tm] Decoder appear to be due more to the way in which the architecture was carefully matched to the hardware needs of the decoding algorithms than to the particular embellishments in the decoding algorithms which have been implemented.

B. Running Times of the GF1[tm] Decoder

An ECL version of the GF1[tm] computer, which is designed to perform the Galois field arithmetic needed to decode algebraic codes and to control the peripheral hardware units which implement buffering and syndrome calculations, is projected to run at speeds up to about 40 ns/clock cycle. Assuming worst case specifications on the IC's, there are several paths which would operate with only marginal timing tolerances at 35 ns, no matter how careful the layout, and a clock cycle time of 50 ns appears quite ample even for a wire-wrapped prototype with unoptimized layout and relatively poor signal propagation times.

The other factor which determines running time is the number of cycles which the microprograms will need in order to finish decoding each block before the next block arrives. There is a T²L version of the GF1[tm] Decoder now in production, and considerable experimental data exists on the running times of its RS decoding programs. By making approximate allowances for the expected slight increases in running times due to larger field sizes, we have obtained estimates of the running times for most of the subroutines in the decoding programs. The running times of those few subroutines which are not implemented in the present version have been estimated by careful analysis of the algorithms.

Based on these careful analyses, we have obtained the throughput estimates shown in Fig. 15, which should be regarded as superimposed on Fig. 14. The wiggliness of the curves in Fig. 15 is indicative of a likely margin of error of plus or minus 20 to 30 percent. However, the estimates are certainly correct to within a factor of 2, and they are usually much better than that.

The accuracy of these running-time estimates depends somewhat on the details of the external specifications. For small t, particularly for $t = 1$ and 2, the running time is dominated by the overhead necessary to interface the decoder with the rest of the system, and this is strongly dependent on the details of the interface. For example, should the header to each decoded block contain a count of the number of errors which the decoder has corrected? Should it also contain information about the burstiness of those errors?

The running time estimates improve if it is not necessary to correct all error patterns within the allotted time. In some situations, well over 99 percent of the error patterns can be corrected in a small fraction of the worst case time. Fig. 15 is based on worst case times rather than on average times.

Another relevant parameter is the amount by which the parent code has been shortened. Notice that shortened codes

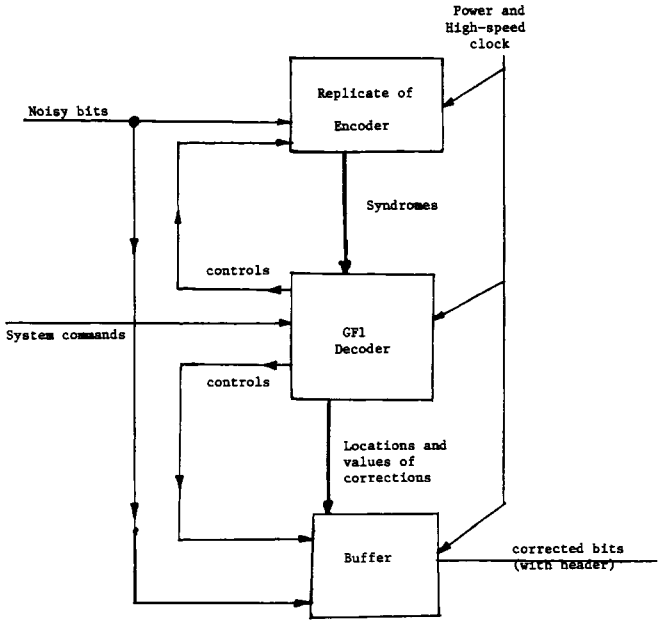


Fig. 16. Configuration of decoder submodules.

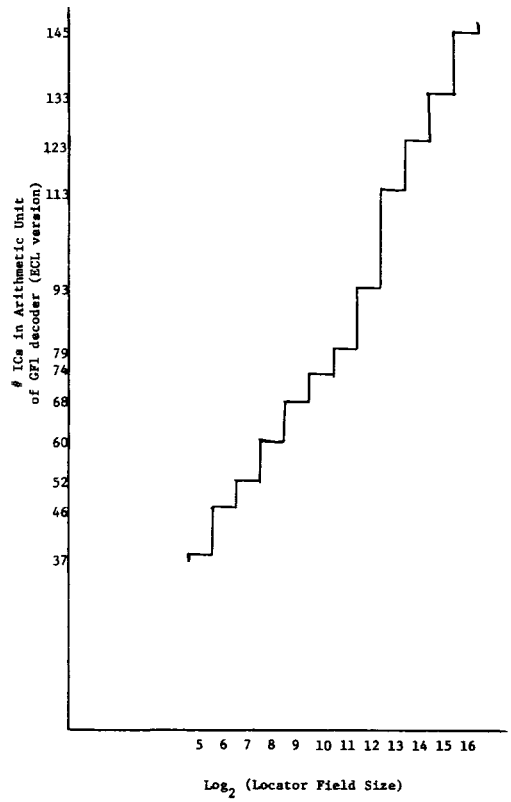


Fig. 17.

are *harder* to decode, because the decoder must do the same amount of computation as necessary for the parent code, whereas if the actual code is only half as long as the parent, then the decoder has only half as much time to do the calculations. This consideration favors the selection of block lengths which correspond to codes whose lengths are close to the lengths of their parents.

For any fixed value of n, the t-character-error-correcting RS code requires a smaller Galois field than the t-error-correcting binary BCH code of the same length. In many cases, this more than makes up for the time lost to compute the values of the erroneous characters, and so RS decoders can operate slightly faster than BCH decoders. It also turns out that many RS decoders also require slightly less hardware than comparable BCH decoders. Thus the only advantage of long BCH codes over long RS codes is the lower redundancy required to correct arbitrary patterns of t isolated single-bit errors. Even this advantage disappears when the noise comes in bursts.

C. Hardware Complexity

1) Encoding Hardware: There is a wide range of options available to the encoder designer, and some of them are well known. As with most hardware design problems, the major tradeoff is between speed and complexity.

Using the most conventional designs, a 30-Mbit/s encoder capable of encoding any of p different binary BCH codes can be constructed primarily from the following Motorola/Fairchild ECL 10K series of IC's: 10176, 10164, 10160, with some possibility of improvement by replacing some 10160 by 10163 or 10193. Such an encoder would have p different generator polynomials wired in, with the capability of changing the selection among them from block to block. If r_{max} is the maximum redundancy, then the total number of IC's in that encoder is approximately

$$\#\text{IC's} \approx \frac{r_{max}}{3} + \frac{8p}{7} + 12.$$

This number can be decreased by reducing the flexibility or by slowing down the speed requirement.

The same general approach may also be used in the design of RS encoders. The above estimate of the IC count is changed only in that the term $8p/7$ is replaced by $cp/2$, where c represents the character size.

When the character sizes are moderately large (i.e., 8 or more bits/character) and the speed requirements are sufficiently lax (i.e., not much more than 10 Mbit/s then the IC count of an RS encoder can be significantly reduced by switching to a quite different design philosophy [13] which uses more LSI (primarily the 10145). Interleaving can usually be included within such an encoder at negligible extra hardware costs.

2) Decoding Hardware: The fast decoder for any of the long BCH or RS codes considered here would consist of a replicate of the encoder, a buffer, and a GF1^{tm} computer, connected as shown in Fig. 16. Estimates of the amount of hardware in the encoder have been presented in the previous section.

3) The GF1^{tm} Decoder: For present purposes, we may consider the GF1 computer as consisting of three parts: an addressing/control unit, a PROM which holds the machine's microcode, and an arithmetic unit. The addressing/control unit comprises 33 IC's, independent of everything else. The PROM consists of 9 MCM 10150, which is expected to be barely sufficient to implement the decoding algorithms for a single code correcting up to 8 errors. A more flexible program may require 18 or 27 IC's instead.

The complexity of the arithmetic unit depends on the wordlength as shown in Fig. 17. The wordlength required for an RS code is identical to its character size; the wordlength required for a BCH code is the $\log_2$ of the blocklength of the parent code. If flexibility is necessary, it is possible to build a single arithmetic unit which would handle either of two or more different wordlengths, but more hardware is required.

Technology	Manufacturer	IC#	Size	Approximate Bandwidth	# Pins
ECL	Fairchild	100415 or 10415A	1K	20 Mbits	16
ECL	Fairchild	10470	4K	10 Mbits	18
MOS dynamic	Intel	2116-2	16K	2 Mbits	16

Fig. 18.

	20 Mbits/second c = 8 t = 2 Uninterleaved	40 Mbits/second c = 12 t = 8 Interleaved	Primary Sensitivity
Encoder Replicate	15 to 36		Redundancy
GF1	102 to 145		Character Size, Speed, Flexibility
Buffer	25 to 46		Block Length, Retry Strategy, IC Technology, Speed
Totals	142 to 227		

Fig. 19. IC counts (all ≤18 pins per IC).

All of the numbers shown in Fig. 17 are prototype numbers based on designs oriented toward very high speed. No LSI is used. If very high speeds are not required, then the number of IC's can be significantly reduced.

Also notice that all of these numbers are based on designs which use 16- or 18-pin devices exclusively. Designs using larger components must be compared on the basis of board area: one 40-pin IC costs 4.3 times as much area as one 14-pin IC;

4) The Buffer: The buffer must contain sufficient memory to hold at least two full codewords. It must also have sufficient bandwidth to allow bits to be streamed in and out simultaneously at the maximum throughput rate. These two requirements constrain the number and type of memory IC's which form the heart of the buffer.

The three leading candidates for the type of memory component on which the buffer will be based are shown in Fig. 18. The "bandwidth" of each chip shown in Fig. 18 is the reciprocal of the time which would be required to select an address, read data out of that address, and then write new data into the same address before changing to a new address and repeating the entire cycle. These bandwidths include a moderate allowance for timing skews and signal propagations on a wire-wrapped board having a memory wordlength of 8–16 bits. Even more conservative estimates might be appropriate to ensure very high hardware reliability, although it might also be possible to attain better bandwidths based on the improved signal propagations attainable on PC boards.

In order to take advantage of the 64 K RAM available only in the MOS technology, it may be advantageous to implement the buffer in T^2L rather than in ECL. By keeping the number of interconnections small, we anticipate that this could be done with only about 4 conversion IC's. However, it would also require a separate board and a different power supply, and these costs might exceed the costs of additional ECL memory. Fortunately, the fact that we anticipate so few connections between the buffer and the rest of the decoder would allow us to change the technology of the buffer at a relatively late stage of the development cycle.

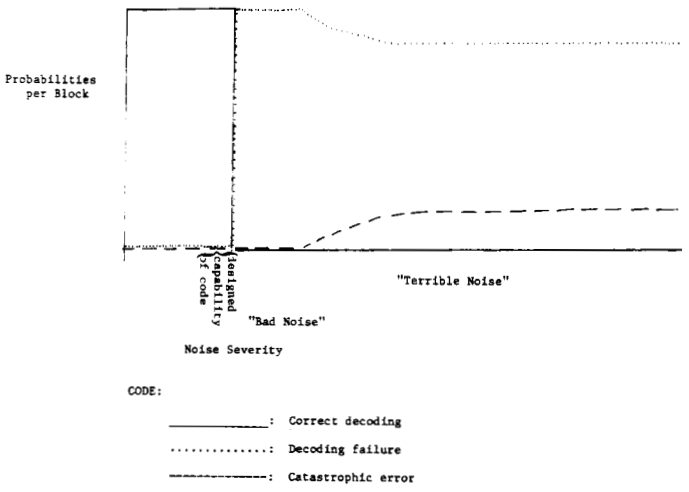


Fig. 20.

If the buffer is based on the MOS memory, then its size is determined by the bandwidth requirement. In order to be capable of operation at 20 Mbit/s, it would need at least 10 memory IC's with a bandwidth of 2 Mbit/s each. If speeds of 32 Mbit/s are desired, then at least 16 of the MOS memory IC's will be needed. In either case, size of the buffer memory is likely to be larger than needed. Conceivably the oversized buffer could be used to provide some flexibility in how quickly the rest of the system responds to a decoding failure. The refresh requirements of the dynamic MOS 64K memory do not appear to pose any significant restriction; these requirements would be automatically met if the buffer is operated in the manner in which we envision.

If the buffer is implemented in ECL circuitry, then the number of memory chips is determined by the required memory size rather than by the bandwidth requirement. An ECL buffer memory could easily be designed to have more than twice the bandwidth of the rest of the system.

D. Conclusion

Combining the above inputs leads us to the "bottom line," which is presented in Fig. 19.

There are many ways in which hardware savings can be achieved if less performance is required. At short to moderate block lengths, it may be possible for the GF1[tm] Decoder to swallow up the buffer and/or the replicate of the encoder. For example, the model which decodes the (31, 15) RS code has swallowed up both, yielding a complete decoder consisting of 73 IC's (all ≤16 pins) packaged onto one side of *a single card less about 15 cm × 15 cm.* The low-power Schottky version has a worst case decoding time of under 800-μs block. This version uses a 200-ns clock speed. A similar ECL version would have about the same number of IC's and run four times as fast, but it would require more power.

Appendix B—Prevention of Catastrophic Error

An attempt to decode any particular word with any particular decoding algorithm for any particular code will result in one of three possible outcomes

1) Correct decoding.

2) Decoding failure, which detects that the error pattern has greater severity than the decoder is prepared to handle, and then evokes some backup strategy such as a slower and more powerful decoding algorithm or an attempt to reread the data.

3) Catastrophic error, which occurs when an unusually malevolent error pattern fools the decoder into changing some bits and then transmitting incorrect data to the user with the claim that it has been corrected.

For any particular code, we may plot the probabilities of these three events as a function of the severity of the noise. For example, for uninterleaved binary BCH codes, the noise severity may be taken as the total number of bit errors per block. (A precise definition of a single parameter representing "noise severity" proves more difficult in other cases, but the following comments remain valid nevertheless.)

Fig. 20 shows the qualitative behavior of the three probabilities as a function of noise severity. The probability of correct decoding is one if the noise does not exceed the designed capability of the code; otherwise it is zero. If the noise severity is "bad" (i.e., only slightly above the code's capacity), then the probability of failure is one or nearly one and the probability of catastrophic error is zero or nearly zero. However, as the noise severity continues to increase well above the code's capability, the probability of catastrophic error also increases, approaching an asymptote which is the *probability of catastrophy conditioned on terrible noise.* In some (presumably typical) cases, any noise severity more than double the code's designed capability is large enough to be "terrible."

In well-designed systems, catastrophies occur too infrequently to facilitate the collection of valid statistics. Most theoretical noise models suggest that "terrible" noises are so unlikely that most catastrophies will result from bad noises rather than from terrible noises. Unfortunately, the behavior of the curves of Fig. 12 in the region of "bad" noise severity is very difficult to obtain for most codes, partly because it depends very much on very detailed information about the code (e.g., which information digits are omitted when the code is shortened). We are therefore generally forced to rely on bounds based on the probability of catastrophy conditioned on terrible noise, even though this typically provides an overly conservative estimate of the probability of catastrophy conditioned on bad noise.

Fortunately, the probability of catastrophy conditioned on terrible noise is relatively easy to calculate. Conceptually, we simply assume that no signal is present, so that the word entering the decoder is a completely random collection of bits and any codeword which the decoder may select is wrong. The probability of catastrophy conditioned on terrible noise is equal to the probability that the decoder does not fail to decode a block of completely random bits. Denoting the probability of catastrophy conditioned on terrible noise by $P(c|t)$, we have

$$P(c|t) = 2^{-r} \cdot (\#\text{ of error patterns which code will correct}).$$

For unshortened uninterleaved t-bit-error-correcting binary BCH codes,

$$\begin{aligned} P(c|t) &= 2^{-r} \cdot \sum_{i=0}^{t} \binom{n}{i} \\ &\approx 2^{-r} \binom{n}{t} \\ &\approx \frac{2^{-r} n^t}{t!} \\ &\approx \frac{1}{t!}. \end{aligned}$$

For unshortened uninterleaved t-character-error-correcting RS codes,

$$\begin{aligned} P(c|t) &= 2^{-ct} \cdot \sum_{i=0}^{t} \binom{n}{i} (2^c - 1)^i \\ &\approx 2^{-ct} \binom{n}{t} (2^c - 1)^t \\ &\approx 2^{-ct} \frac{2^{ct}}{t!} (2^c - 1)^t \\ &= \frac{(1 - 2^{-c})^t}{t!} \\ &\approx \frac{1}{t!} \text{ whenever } t \ll 2^c. \end{aligned}$$

Since a word in a doubly interleaved code is decodable only if both subcodes are decodable, for unshortened doubly interleaved BCH or RS codes we have

$$P(c|t) \approx \left(\frac{1}{t!}\right)^2.$$

In general, $P(c|t)$ can be decreased by increasing the redundancy or by setting a decoding threshold at a lower value than the code was designed to correct. (This follows the well-known bureaucratic maxim that if you can avoid the responsibility for all of the hard decisions, you need not take the blame for too many mistakes!) In general, a high value of $P(c|t)$ is one of the penalties we must expect to pay for using a "good" code. If we squeeze the redundancy down to the smallest possible value, then all of the redundancy is used for error correction, and since there is little or no excess redundancy left over for error detection, many (or all) terrible noises will cause a catastrophic decoding error.

Although we can generally lower the value of $P(c|t)$ by refusing to accept some noise patterns which the decoder thinks it has "decoded," this usually forces us to pay a significant price in terms of a reduced probability of correct decoding. However, RS codes provide an uncommon exception to this rule, because many error patterns which are "corrected" by the naive decoding algorithm are quite unlikely to be caused by multiple bursts of "natural" errors. This is because most patterns of t or fewer character errors consist of bursts of length slightly less than c bits, and these bursts rarely cross over the boundaries between characters of the code, whereas most "natural" error bursts occur at phases uncorrelated with the code's character boundaries.

The $P(c|t)$ for RS codes may be improved considerably by a *rejection strategy.* After the decoder has found the error pattern which contains no more than t character errors, this error pattern may be examined and partitioned into bursts. For these purposes, a character-error burst is defined as a consecutive sequence of erroneous characters, bounded on both sides by correct characters but having no correct characters within the burst. Within each character-error burst, we may then define the unique bit-error burst as the maximum-length sequence of bits which starts and ends on erroneous bits. In this way, we may partition any RS error pattern into a sum of bit-error bursts of lengths $b_1, b_2, b_3, \cdots$.

After the RS decoding algorithm has found a candidate error pattern of t or fewer character errors, we may then compute the lengths of the bit-error bursts. A typical "wrong" pattern will consist of about t bursts of lengths averaging about $c - 2$. A typical "natural" pattern will consist of fewer bursts, with lengths much shorter (or much longer!). We now propose a

criterion based on those lengths for deciding whether or not each candidate error pattern should be corrected or rejected. Although rejections force rereads, they buy a considerable amount of insurance against catastrophic error in return for an almost negligible decrease in the probability of correct decoding of any natural burst-noise source.

On the average, a randomly phased burst of length b will strike $1 + (b - 1)/c$ characters. Hence, any error pattern which is composed of the sum of I bursts of lengths $b_1, b_2, \cdots, b_I$ whose lengths satisfy

$$1 + \sum_{i=1}^{I} \left(\frac{b_i - 1}{c} \right) > t$$

cannot be corrected unless there are favorable phase relationships between the error bursts and the characters. If

$$1 + \sum_{i} \left(\frac{b_i - 1}{c} \right) > t + 1$$

then an even more fortuitous set of phase relationships is necessary. Hence, a decoder which accepts only those error patterns whose burst lengths satisfy

$$\sum_{i=1}^{I} b_i \leqslant c(t + 1) - (c - 1)I$$

has a probability of decoding failure against natural error patterns which is no less than perhaps twice as large as the probability of the complete algebraic algorithm which corrects all patterns of t or fewer character errors. However, the probability of catastrophy conditioned on terrible noise is greatly reduced by rejecting all error patterns except those for which

$$\sum_{i=1}^{I} b_i \leqslant c(t + 1) - (c - 1)I$$

We now compute a bound on the number of correctable error patterns which satisfy this criterion. For each value of I, there are less than n^I positions for the I bursts. (If all b_i are equal, then the actual number is even less than $\binom{n}{I} \approx n^I/I!$, but if all b_i are distinct and $\Sigma\, b_i \ll n$, the number differs from n^I only by an insignificant amount due to the possibility of overlapping bursts.) Once we have fixed the positions of all bursts, the number of bursts which occupy those positions is no greater than

$$2^{\Sigma(b_i - 2)}$$

where the sum runs over only those

$$b_i \geqslant 2.$$

This is overbounded by including

$$b_i = 1$$

in the sum, which is then in turn overbounded by

$$\sum_{i=1}^{I} (b_i - 2) \leqslant c(t + 1) - (c + 1)I.$$

Therefore, for each fixed I, the number of unrejected error patterns is less than

$$n^I\, 2^{c(t+1)-(c+1)I}.$$

Whence

$$\begin{aligned} P(c|t)\, 2^r &< \sum_{I=1}^{t} n^I\, 2^{c(t+1)-(c+1)I} \\ &= 2^{ct+c} \sum_{I=1}^{t} \left(\frac{n}{2^{(c+1)}} \right)^I \\ &< 2^{ct+c} \frac{n^t}{2^{(c+1)t}} \frac{1}{(1 - 2^{c+1}/n)} \\ &< 2^{c-t} n^t \frac{1}{(1 - 2^{c+1}/n)}. \end{aligned}$$

This bound is reasonable for all values of n in the region $2^{c+2} \leqslant n \leqslant c2c$. Other estimates prove tighter when the RS code is shortened so much that $n \approx 2^{c+1}$.

References

[1] C. E. Shannon, "A mathematical theory of communications," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, 623-656; and *Math. Rev.*, vol. 10, p. 133, 1948.

[2] R. T. Chien, "Burst-correcting codes with high-speed decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 109-113, Jan. 1969.

[3] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611-656; and *Math. Rev.*, vol. 21, p. 1920, 1959.

[4] R. G. Gallager, "A simple derivation of the coding theorem and some applications," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 3-18; and *Math. Rev.*, vol. 32, p. 3951, 1965.

[5] —, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[6] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, "Lower bounds to error probability for coding on discrete memoryless channels," *Inform. Contr.*, vol. 10, pp. 65-103, 522-552; and *Math. Rev.*, vol. 21, p. 1920, 1967.

[7] S. W. Golomb *et al.*, *Digital Communications with Space Applications.* Englewood Cliffs, NJ.: Prentice-Hall, 1964, Apendix 4, pp. 196-204.

[8] W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1973, Table 5-1, pp. 199-209.

[9] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300-304; and *Math. Rev.*, vol. 23B, p. 510, 1960.

[10] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147-160; and *Math. Rev.*, vol. 12, p. 35, 1950.

[11] R. M. Heller, "Forced-erasure decoding and the erasure reconstruction spectra of group codes," *IEEE Trans. Commun. Technol.*, vol. COM-15, pp. 390-397, June 1967.

[12] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968, ch. 9.

[13] —, "Better Reed-Solomon encoders," presented at California Institute Technology EE Seminar, Pasadena, CA, Dec. 12, 1979.

[14] A. J. Viterbi, "Error bounds for convolutional codes and an asympotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, 1967.

[15] G. D. Forney, Jr., "Coding and its application in space communications," *IEEE Spectrum*, vol. 7, no. 6, pp. 47-58, June 1970.

[16] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun.*, vol. COM-19, no. 5, Oct. 1971.

[17] I. M. Jacobs, "Practical applications of coding," *IEEE Trans. Inform. Theory*, pp. 305-310, May 1974.

[18] R. J. McEliece, "Theory of information and coding," in *Encyclopedia of Mathematics and Its Applications, Vol. 3.* Reading, MA: Addison-Wesley, 1977, ch. 9.

[19] I. M. Jacobs and E. R. Berlekamp, "A lower bound to the distribution of computation for sequential decoding," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 167-174, 1967.

[20] P. Elias, "Coding for noisy channels" in *IRE Conv. Rec.*, pt. 4, pp. 37-46, 1955.

[21] M. A. Epstein, "Algebraic decoding for a binary erasure channel," M.I.T. Res. Lab. Electron. Rep., vol. 340, 1955.

[22] G. Landsburg, "Uber eine Anzahlbestimmung und eine damit zusammenhangende Reihe," *J. Reine Angew. Math.*, vol. 111, pp. 87-88, 1893.

[23] D. Chase, "Class of algorithms for decoding block codes with

channel measurement information," *IEEE Trans. Inform. Theory*, pp. 170-182, Jan. 1972.
[24] R. M. Fano, *The Transmission of Information*. Cambridge, MA: M.I.T. Press and Wiley, 1961.
[25] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inform. Contr.*, vol. 3, pp. 68-79, 279-290; and *Math. Rev.*, vol. 22, p. 3619, 1960.
[26] A. Hocquenghem, "Codes correcteurs d'Erreurs," *Chiffres* (Paris), vol. 2, pp. 147-156; and *Math. Rev.*, vol. 22, p. 652, 1959.
[27] V. D. Goppa, "A new class of linear error-correcting codes," *Probl. Peredach. Inform.*, vol. 6, no. 3, pp. 24-30, Sept., 1970.
[28] ——, "Rational representation of codes and (L, g) codes," *Probl. Peredach. Inform.*, vol. 7, no. 3, pp. 41-49, Sept. 1971.
[29] E. R. Berlekamp, "Goppa codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 590-592, Sept. 1973.
[30] N. J. Patterson, "The Algebraic Decoding of Goppa Codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 203-207, Mar. 1975.
[31] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equations for decoding goppa codes," *Inform. Contr.*, vol. 27, pp. 87-99, 1975.
[32] W. W. Peterson, *Error-Correcting Codes*. Cambridge, MA: M.I.T. Press, 1961.

Crosscorrelation Properties of Pseudorandom and Related Sequences

DILIP V. SARWATE, SENIOR MEMBER, IEEE
AND MICHAEL B. PURSLEY, SENIOR MEMBER, IEEE

Invited Paper

Abstract—Binary maximal-length linear feedback shift register sequences (m-sequences) have been successfully employed in communications, navigation, and related systems over the past several years. For the early applications, m-sequences were used primarily because of their excellent periodic autocorrelation properties. For many of the recent systems applications, however, the crosscorrelation properties of such sequences are at least as important as the autocorrelation properties, and the system performance depends upon the aperiodic correlation in addition to the periodic correlation. This paper presents a survey of recent results and provides several new results on the periodic and aperiodic crosscorrelation functions for pairs of m-sequences and for pairs of related (but not maximal-length) binary shift register sequences. Also included are several recent results on correlation for complex-valued sequences as well as identities relating the crosscorrelation functions to autocorrelation functions. Examples of problems in spread-spectrum communications are employed to motivate the choice of correlation parameters that are considered in the paper.

Manuscript received June 25, 1979; revised January 11, 1980. This work was supported in part by the National Science Foundation under Grant ENG78-06630, the Army Research Office under Grant DAAG-29-78-G-0114, and the Joint Services Electronics Program under Contract N00014-79-C-0424.

The authors are with the Coordinated Science Laboratory and the Department of Engineering, University of Illinois, Urbana, IL 61801.

I. Introduction

THERE are a large number of problems in systems engineering that require sets of signals which have one or both of the following two properties:

i) each signal in the set is easy to distinguish from a time-shifted version of itself;
ii) each signal in the set is easy to distinguish from (a possibly time-shifted version of) every other signal in the set.

The first property is important for such applications as ranging systems, radar systems, and spread-spectrum communications systems. The second is important for simultaneous ranging to several targets, multiple-terminal system identification, and code-division multiple-access communications systems.

The signals employed in the kinds of applications mentioned above are usually required to be periodic. This is primarily because of the simplifications in system implementation that

0018-9219/80/0500-0593$00.75