INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the original text directly from the copy submitted. Thus, some dissertation copies are in typewriter face, while others may be from a computer printer.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyrighted material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is available as one exposure on a standard 35 mm slide or as a 17″ × 23″ black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. 35 mm slides or 6″ × 9″ black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Accessing the World's Information since 1938

300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

Order Number 8801733

A new class of forward error correcting codes for burst and random errors

Manela, David, Ph.D.

City University of New York, 1987

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ___√___.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy ___✓___
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ___✓___
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other _____

_____

_____

U·M·I

A NEW CLASS OF FORWARD ERROR CORRECTING CODES

FOR

BURST AND RANDOM ERRORS

by

DAVID MANELA

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1987

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

7/27/87
Date

Chair of Examining Committee

July 27, 1987
Date

Executive Officer

Prof. R. Pickholtz

Prof. J. Barba

Prof. L. Roytman

Prof. T. Saadawi

Supervisory Committee

The City University of New York

ABSTRACT

A NEW CLASS OF FORWARD ERROR CORRECTING CODES

FOR

BURST AND RANDOM ERRORS

by

David Manela

Adviser: Professor D.L. Schilling

A new class of error-correcting codes, with random and burst error correcting capability is described, (we will refer to it as the SM codes). A SM encoding method is provided comprising the steps of storing a block of data-bit in a memory, calculating parity-check symbols from parity-line symbols having p-bits per symbol along parity lines, and setting the parity-check symbols equal to the modulo p sum of the parity-line symbols. A SM decoding method is provided comprising the steps of storing an encoded data-bit sequence in memory. The encoded data-bit sequence includes a parity-check-symbol sequence which is stored in parity-memory cells, and data-bit sequence which

is blocked and stored in information-memory cells. The parity-check symbols and the parity-line symbols along the parity lines in the information-memory cells are found. The count of each composite cell on a composite-error graph traversed by the path of each of the parity lines having an error is incremented and the largest-number cell in the composite-error graph having the largest number is determined. The largest number is compared to a threshold, and a new data symbol is chosen to minimize the count in the largest-number cell and substituted into the stored data-bit sequence.

The main features of this code are:

(1) Extreme simplicity of design of the decoder as well as the encoder. For example, the decoding algorithm is of significantly less complexity than that of the Viterbi algorithm for the same error rate performance.

(2) The codec is less complex and provides a higher throughput for the same error rate thakn other codes, such as Chase's Code Combiner.

(3) The codec can be used for burst error correction and in this operation operates at the theoretical maximum efficiency as defined by the Rieger Bound.

ACKNOWLEDGEMENT

I wish to express my deep gratitude to professor Donald L. Schilling for the his interest, encouragement and valuable guidance during the progress of this research. The many technical discussions and his constructive suggestions are most appreciated.

I would like to thank all the members of my Doctoral Committee; Dean Paul Karmel, Professor Raymond Pickholtz, Professor Joseph Barba, Professor Leonard Roytman, and Professor Tarek Saadwi, for the time and effort each has taken to read and constructively criticize this dissertation. In addition, Professor David Newman Jr.'s interest in this work and his valuable comments are appreciated.

Special thanks to my parents and my brother for their support and encouragement. Finally, I would like to thank my wife Asnat for her constant assistance, her sense of humor and patience in this project.

TABLE OF CONTENTS

TABLE OF FIGURES

1. STATEMENT OF THE PROBLEM

1.1 INTRODUCTION

In recent years,there has been an increasing demand for efficient and reliable digital data transmission and storage systems. This demand has been accelerated by the emergence of large-scale, high-speed data networks for the exchange, processing, and storage of digital information in commercial and military sectors. A merging of communication and computer technology is required in the design of these systems. A major concern of the designer is the control of errors so that reliable reproduction of data can be obtained.

In 1948, Shannon [1] demonstrated in a landmark paper that, by proper encoding of information, errors induced by a noisy channel or storage medium can be reduced to any desired level without sacrificing the rate of information transmission or storage. Since Shannon's work, a great deal of effort has been expanded on the problem of devising efficient encoding and decoding methods for error control in a noisy environment. Recent developments have contributed toward achieving the reliability required by today's

high-speed digital systems, and the use of coding for error control has become an integral part in the design of modern communication systems and digital computers.

1.1.1 Coding Systems

The transmission of digital data and storage of digital information have much in common. They both transfer data from an information source to a destination (user). A block diagram which describes the digital communication process (or storage) system is shown in Fig. 1.1-1.

Analysis shows that communication channels (or storage media) can theoretically operate at maximum bit rate called the channel capacity for information transmission, so that, if the rate of the source is less than this capacity ,it is possible to choose a set of signals such that the probability of erroneous decoding is arbitrarily small. This theory does not indicate precisely how these signals are to be constructed nor does it guarantee that such a system can be built in the real world.

The use of error-correcting codes is an attempt to circumvent these problems. A system that employs the type of coding described, is shown in Fig. 1.1-1.

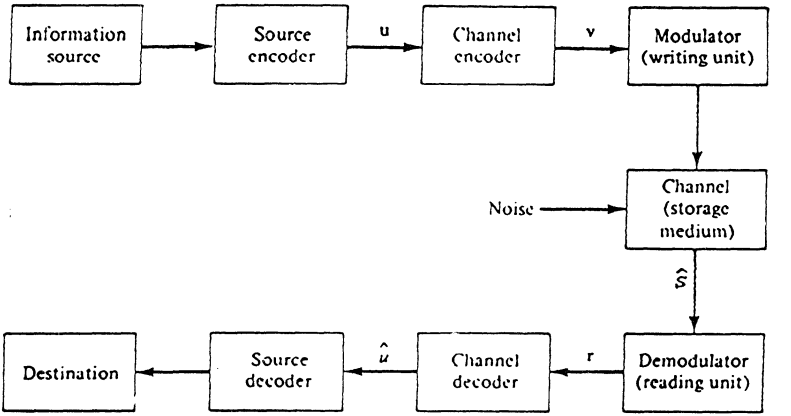


Figure 1.1-1 Block diagram of a typical data transmission or storage system.

Data (Information) source

The data source generates data in the form of symbols. The symbols may be generated by a digital system (computer), or by an analog to digital (A/D) conversion of a continuous signal. The sequence of symbols is called the information sequence u.

Channel Encoder

The channel encoder transforms the information sequence u into a discrete encoded sequence v called a code word. In most instances v is a binary sequence, although in some applications nonbinary codes have been used. The design and implementation of the channel encoders is to combat the noisy environment in which the code words must be transmitted or stored.

Modulator (writing unit)

Discrete symbols are not usually suitable for transmission over a physical channel or for recording on a digital storage medium. The modulator transforms each output symbol of the channel encoder into a waveform of duration T seconds which is suitable for transmission.

Channel (Storage Medium)

The waveform channel consists of all hardware and physical media that the waveform passes through in going from the output of the modulator to the input of the demodulator. $s(t)$, of the channel is a scaled replica of the input s(t), to which some random disturbance, n(t) has been added. Much more general situations are possible. Distortion may be present due heavy filtering or multiple signal paths. The disturbance may cause signal suppression which could in turn cause the amplitude of the received signal to vary. The channel itself may be time varying which again could cause random amplitude and phase to occur. The disturbance n(t) may be simple receiver noise which may be modeled as an additive Gaussian process, it may be urban noise of various kinds, or it may be intentional jamming by an unfriendly party. Although much of the the existing work in coding has been tailored to the simple case of additive Gaussian noise, it is often in the more complicated situations where coding concepts prove to be the most effective.

Demodulator (reading Unit)

The demodulator is a device which estimates which of the possible symbols was transmitted based upon an observation of the received signal $s(t)$. The probability that this estimate is correct depends upon the ratio of the signal power to the noise power in the data bandwidth, the amount of signal distortion due to filtering and nonlinear effects, and the detection scheme that is being used. In a coded system the demodulator sometimes performs a second function. That function is to supply information to the decoder as to the reliabilty of each individual symbol decision (soft decision).

This information may be obtained in a number of different ways, and the approach utilized depends strongly on the nature of the disturbance, n(t). One possibility occurs when there is a known interference such as a jammer or a radar signal whose presence can be determined independently. The reliability information would then be a single symbol that indicates whether the interference exists or not. This information results in an "erasure", and the channel decoder can use that information to improve significantly the probability of error. Another possibility

occurs when the detector is a sampled match filter, and n(t) is additive Gaussian noise. In this case, the magnitude of the sampled voltage relative to the decision threshold is a strong indication of the reliability of the decision, and that information can be furnished to the channel decoder to decrese the probability of error.

Channel Decoder

The channel decoder is the device that inverts the operation of the channel encoder. Because the sequence of symbols r that are generated by the demodulator may contain errors, the decoder must perform a significantly more complex mapping than the encoder. Although this could in principle be performed using an optimal techniques such as maximum likelihood detection, this approach rapidly becomes impractical as the size of the code grows. In practice, one must often resort to a suboptimal technique. To make the decoding operation feasible, one must devise computational procedures for realizing this process. The decoding algorithms can be categorized according whether the techniques apply to block codes or tree codes. The use of reliability information or so-called ("erasure" or "soft-decision") is applicable to both block codes and tree codes.

Destination

The destination accepts the sequence $\hat{u}$ from the channel decoder. When the source is continuous, this involves an additional digital to analog (D/A) conversion. In a well-designed system, the estimate will be a faithful reproduction of the source output except when the channel (or storage medium) is very noisy.

The essence of this work is to suggest a unique channel encoder/decoder pair such that information can be transmitted (or recorded) in a noisy environment as fast as possible, reliable reproduction of the information can be obtained at the output of the channel decoder, and the cost of the implementing the encoder and decoder falls within acceptable limits.

1.2 TYPES OF CHANNELS (ERRORS)

From the view point of the encoder and decoder, the segment of Fig. 1.1-1 enclosed in the dash lines is the most important. The channel is characterized by a set of input symbols, output symbols and, transition probabilities.

In the simplest case the transition probabilities are time invariant and independent from symbol to symbol. This is the so called discrete memoryless channel (DMC). The most commonly encounted case of the DMC is the so called binary symmetric channel (BSC), whose transition diagram is shown in Fig. 1.2-1. Each transmitted bit has a probability p of being incorrect and a probability 1-p of being received correctly, independent on the other bits. Hence transmission errors occur randomly in the received sequence. Good examples of such channels are the deep-space channel and many satellite channels, as well as many line of sight transmission facilities. The codes advised for correcting random errors are called random-error-correcting codes.

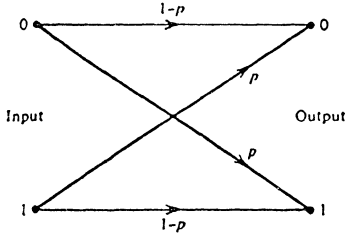


Figure 1.2-1 Binary symmetric channel.

So far we have discussed examples in which symbol errors are independent from one symbol to the next. Often in practical structures this will not be true, and closely spaced or "burst" errors may be more common than widely spaced errors. This could be caused by an almost periodic noise source such as a near by communication or radar system or rotating machine or by fading in the communication link. One can extend the modeling ideas discussed thus far by making the transition probability depend upon previous transmission or by making them a function of time. Typical solutions for this type have included either burst-error-correcting codes which exhibit good word separation properties based upon burst lengths rather then on the number of random errors, or interleaving several code words from random-error-correcting code.

Finaly, some channels contain a combination of both random and burst errors. These are called compound channels, and code devised for correcting errors of these channels are called burst-and-random-error-correcting codes.
The code suggested in this work belongs to this group.

1.3 THE CODING PROBLEM

For codes to be very effective they must be long, so as to average the effect of the noise over a large number of symbols. Such a code may have 10^{100} possible code words and many times this number of possible received words. While the code and decoding are still conceptually described by a table, it becomes impossible to construct such a table, or even to list all of the code words. A mathematical structure can enable us to determine the important properties of such codes.

Thus, there are three main aspects of the coding problems:

(1). To find codes that have the required error correcting ability.

(2). To find a practical method of encoding of the code.

(3). To find a practical method of making the decision at the receiver, that is, a method of error correcting.

The typical attack on the problem has been to find codes that could be proven mathematically to satisfy the required error-correcting ability. This mathematical structure is then exploited to meet the other two requirements, ability to encode and to decode.

1.4 TYPES OF CODES

The channel encoder accepts at its input a continuous sequence of information digits. At its output it produces another sequence with somewhat more digits, which fed to the modulator. Conversely, the decoder accepts a sequence of channel symbols from the demodulator and translates it into a somewhat shorter sequence of information digits. The rules under which the encoder and the decoder operate are specified by the particular code that is employed.

1.4.1 Block Codes

There are two fundamentally different types of codes, block codes and convolutional codes. The encoder for block code breaks the continuous sequence of information digits into k-symbol sections or blocks. It then operates on these blocks independently according to the particular code to be employed. With each possible information block is associated an n-tuple of channel symbols, where n>k. The result, now called a code word, is transmitted, corrupted by noise, and decoded independently of all other code words. The quantity n is referred to as the code length or block length.

Let q denote the number of distinct symbol employed on the channel; here q will be assumed as a power of a prime number, the binary case (q=2) is the most used case. A block code is a set of M sequences of channel symbols of length n. These q-ary n-tuples are called the code words of the code. The total number of code words is $M=q^k$.

1.4.1-1 Properties of Block Codes

Block codes are characterized by their algorithm for encoding and decoding and also by several other properties:

Code rate R:

The code rate is the ratio of information data symbols k, to the total number of symbols in a code word n, which is the sum of information data symbols and the parity check symbols r. Thus, the number of symbols in a codeword is n=k+r and the code rate R=k/n.

Minimum Hamming Distance d_{min} :

This parameter determines the random-error-detecting and random-error-correcting capabilities of the code. Let $v=(v_0,v_1,\ldots,,v_{n-1})$ be an n-tuple. The Hamming weight of v,

denote by w(v), is defined as the number of nonzero components of v. For example, the Hamming weight of v=(0,1,0,0,1,1,0) is 3. Let v and u be two n-tuples. The Hamming distance between u and v, denoted d(u,v), is defined as the number of places where they differ. For example, the Hamming distance between v=(0,1,0,0,1,1,0) and u=(1,1,1,0,1,1,1) is d(u,v)=3. Hamming distance is a metric function that satisfies the triangle inequality.

$$d(u,v)+d(u,w)\geq d(v,w) \qquad (1.4-1)$$

It follows from the definition of the Hamming distance and the definition of the modulo-2 addition that the Hamming distance between two n-tuples, v and u, is equal to the Hamming weight of the sum of v and u,

$$d(u,v)=w(u+v) \qquad (1.4-2)$$

For example the Hamming distance between u and v as above d(u,v)=3 and the weight of u+v=(1,0,1,0,0,0,1) w(u+v)=3.

Given a block code C, one can compute the Hamming distance between any two distinct code words. The minimum distance of C, denoted d_{min}, is defined as

$$d_{min} = \min\{d(u,v): u,v \in C, u \neq v\} \qquad (1.4-3)$$

If C is a linear block code, the sum of every two code words in C is another code word in C. Then because the all 0 n-tuple is a code word in C:

$$\begin{aligned} d_{min} &= \min\{w(u+v): u,v \in C, u \neq v\} \\ &= \min\{w(X): X \in C, X \neq 0\} \\ &= w_{min} \end{aligned} \qquad (1.4-4)$$

Summarizing the above result, we have the following theorem

Theorem- The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words.

1.4.2 Tree Codes

The other type of code, called a tree code or convolutional code, operate on the information sequence without breaking it up into independent blocks. Rather the encoder for the tree code processes the information continuously and associates each long (perhaps semi-infinite) information sequence with a code sequence containing an increased number of digits. The encoder breaks its input sequence into blocks, each of which contains k-symbols where

k is usually a small number. Then, on the basis of this k-tuple and the preceding m information symbols, the encoder emits an n -symbol section of the code sequence. Hence the encoder has a memory order of m. The set of encoded sequence produced by the k-input,n-output encoder of memory order m is called an (n,k,m) convolutional code.

1.4.2-1 Properties of Tree Codes

Tree codes are characterized by their algorithm of encoding and decoding and also by several other properties:

Code rate R:

A convolutional encoder generates n encoded symbols for each k information symbols and R=k/n is called the code rate. Note, however, that for an information sequence of finite length $k \cdot L$, (L-the number of blocks entered into the encoder). The corresponding code word has length n(L+m), where the final $n \cdot m$ outputs are generated after the last nonzero information symbols has entered the encoder. In other words, an information sequence is terminated with all-zero block in order to allow the encoder memory to clear. Viewing a convolutional code as a linear block code, the block rate is given by kL/n(L+m), the ratio of the

number of information symbols to the length of the code word. If L >> m, then L / (L+m)≈ 1, and the block code rate and tree code rate are approximately equal. On the other hand if L is small then the effective rate of information transmission. would be reduced below the code rate by a fractional amount

$$\frac{k/n - kL/n(L+m)}{k/n} = \frac{m}{L+m} \qquad (1.4-5)$$

called the fractional rate loss.

Memory order m:

A tree code encoder can be implemented by using shift registers Fig. 1.4-1. The encoder may contain k shift registers, not all of which must have the same length. If K_i is the length of the ith shift register, then the encoder memory order m is defined as

$$m = \max_{1 \le i \le k} K_i \qquad (1.4-6)$$

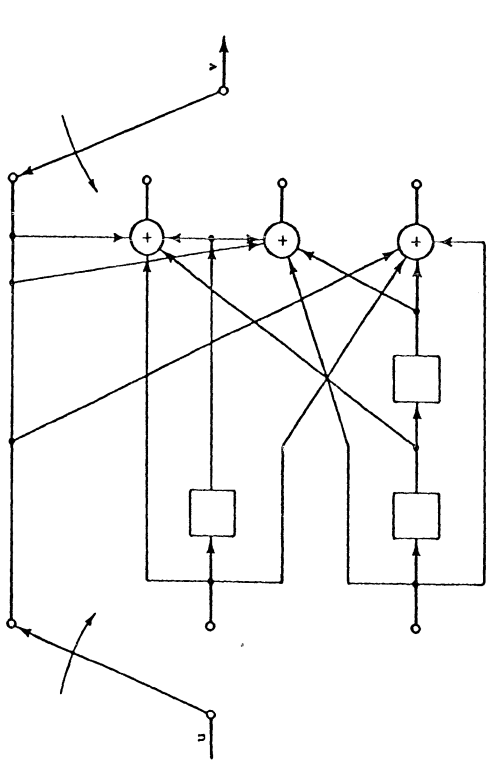


Figure 1.4-1 A (4, 3, 2) binary convolutional encoder.

Constraint length n_A:

The constraint length is defined as

$$n_A = n(m+1) \qquad (1.4-7)$$

since each information symbol remains in the encoder for up to m+1 units, and during each time unit can affect any of the n encoder outputs. n_A can be interpreted as the maximum number of encoder outputs that can be affected by the single a information symbol.

Distance Properties of Convolutional Codes:

The performance of a convolutional code depends on the decoding algorithm employed and the distance properties of the code. The most important distance measure for convolutional codes is the minimum free distance d_{free}, defined as

$$d_{free} = \min\{d(v', v''): u' \neq u''\} \qquad (1.4-8)$$

where v' and v'' are the code words corresponding to the

information sequence u' and u'',respectively. Hence d_{free} is the minimum distance between any two words i the code. Since a convolutional code is a linear code,

$$d_{free} = \min\{d(v', v''): u' \neq u''\}$$
$$= \min\{w(v): u \neq 0\} \qquad (1.4-9)$$

where v is the code word corresponding to the information sequence u. Hence, d_{free} is the minimum weight code word of any length produced by a nonzero information sequence.

Another important distance measure for tree codes is the column distance function (CDF). Letting

$$[v]_i = \left(v_0^1 v_0^2 \ldots v_0^n, v_1^1 v_1^2 \ldots v_1^n, v_i^1 v_i^2 \ldots v_i^n\right) \qquad (1.4-10)$$

denote the ith truncation of the code word v, and

$$[u]_i = \left(u_0^1 u_0^2 \ldots u_0^k, u_1^1 u_1^2 \ldots u_1^k, u_i^1 u_i^2 \ldots u_i^k\right) \qquad (1.4-11)$$

denote the ith truncation of the information sequence u, the column distance function of order i, d_i, is defined as

$$d_i = \min\{d([v']_i, [v'']_i) : [u']_0 \neq [u'']_0\}$$
$$= \min\{w[v]_i : [u]_i \neq 0\} \qquad (1.4-12)$$

where again v is the code word corresponding to the information sequence u. Hence, d_i is the minimum weight code over the first (i=1) time units whose initial information block is nonzero.

Two cases are of specific interest: i=m and $i \to \infty$. For i=m, d_m is called the minimum distance of a convolutional code and will also be denoted d_{min}. Much of early work has treated d_{min} as the distance parameter of most interest. This was due to the fact that the principal decoding at that time had a decoding memory equal to the constraint length. More recently, as Viterbi decoding has become more prominent, d_{free} and CDF have replaced d_{min} as the distance parameter of primary interest.

For $i \to \infty$, $\lim_{i\to\infty} d_i$ is the minimum-weight code word of any length whose first information block is nonzero. It can be shown that for noncatastrophic codes

$$\lim_{i\to\infty} d_i = d_{free} \qquad (1.4-13)$$

Hence, d_i eventually reaches d_{free}, and then increases no more. This usually happens within three to four constraint length.

The maximal achievable d_{free} for convolutional codes with a given rate and encoder memory has not been determined exactly. However, upper and lower bounds on d_{free} have been obtained using random coding approach.

1.5 SOME GENERAL REMARKS ON ERROR-DETECTING AND ERROR-CORRECTING CODES

In an ideal system the symbol that comes out of the channel-symbol-to-destination-symbol converter should match the symbol that entered the source-symbol-to-channel-symbol converter. In a practical system there are occasional errors, and it is the purpose of codes to detect and perhaps correct such errors. These codes cannot correct every conceivable pattern of errors but rather must be designed to

correct only the most likely patterns. Much of coding theory has been based on the assumption that each symbol is affected independently by the noise, so that the probabilty of a given error pattern depends only on the number of errors. Thus for example, codes have been developed that correct any pattern of t or fewer errors in a block of n symbols. While this may be an appropriate model for some channels, on telephone lines and on magnetic-tape storage systems error occur predominantly in bursts. Telephone-line disturbances, such as lightning or switching transients,last longer than the transmission time for one symbol. Symilary, magnetic-tape defects are typically larger than the space required to store one symbol. Consequently, codes for correcting bursts of errors are required,and some remarkably good codes have been developed for this purpose, including the code of this work.

The communication channel shown in Fig. 1.1-1 is a one way channel. Very frequently communication systems employ two-way channels, a fact that should be considered in designing codes. With the two way channel, for example, an error detecting code can be used. When an error is detected at one terminal, a request for a repeat can be given, and thus errors can be effectively be corrected. This kind of

system are called automatic repeat request (ARQ).

There are true examples of one way channels, in which error probabilities can be reduced with error correcting codes, but not by error detecting and retransmission. With a magnetic tape storage system, for example, it is too late to ask for a retransmission after the tape has been stored some times,and errors are detected when the record is read. Encoding for error-correcting and error detecting has the same complexity, it is the decoding that is likely to require complex equipment.

There are many good reasons for using error detection and retransmission when possible. Error detection is by nature a much simpler task than error correction and requires much simpler decoding equipment. Also, error detection with retransmission is adaptive - i.e. when errors occur and are detected an increase in redundant information is transmitted and received, making it possible to get better performance with a system of this kind than is theoretically possible on a one-way channel.

There is a definite limit to the efficiency of a system that uses simple error detection alone. Short error detecting codes cannot detect errors efficiently, while if extremely long codes are used, retransmission must often be

done. It can be shown that a combination of correction of the most frequent error patterns and detection, coupled with retransmission for less frequent patterns is not subject to the limitation and, in fact, is often more efficient then either error correction or error detection and retransmission alone.

1.6 CAPABILITIES of ERROR-DETECTING-CORRECTING CODES

1.6.1 Random Error-Detecting and Error-Correcting Capabilities of Block Code

1.6.1-1 Capabilities of Error-Detecting Codes

When a code vector v is transmitted over a noisy channel, an error having a pattern of ζ errors will result in a received vector r which differs from the transmitted vector v in ζ places. If the minimum distance of the block code C, is d_{min}, any two distinct code vectors of C differ in at least d_{min} places. For this code C, no error pattern of $d_{min}-1$ or fewer errors can change one code vector into another. Therefore, any error pattern of $d_{min}-1$ or fewer errors will result in a received vector r that is not a code word in C. When the receiver determines that the received vector is not a code word of C, we say that the errors are

detected. Hence, a block code with minimum distance d_{min} is capable of detecting all errors patterns of $d_{min}-1$ or fewer errors.

Even though a block code with minimum distance d_{min} guarantees detecting all the error patterns of $d_{min}-1$ or fewer errors, it is also capable of detecting a large fraction of error patterns with d_{min} or more errors. In fact, an (n,k) linear code is capable of detecting 2^n-2^k error patterns of length n. This can be shown as follows: Among the 2^n-1 possible nonzero error pattern, there are 2^k-1 error pattern that are identical to 2^k-1 code words. If any of there 2^k-1 error patterns occur, it alters the transmitted word v into another code word w. Thus ,w will be considered as a correct word and results in an incorrect decoding. Therefore, there are 2^k-1 undetectable error patterns. Hence, there are exactly 2^n-2^k errors patterns that are not identical to the code words of an (n,k) linear code. These 2^n-2^k error patterns are detectable error patterns.

Let C be an (n,k) linear code. Let A_i be the number of vectors of weight i in C.The numbers $A_0,A_1,\ldots A_n$ are called the weight distribution of C. If C is used only for error detection of a Binary Symetrical Channel (BSC), the

probability that the decoder fails to detect the presence of errors can be computed from the weight disribution of C. Let Pu(E) denote the probability of an undetected error. Since an undetected error occurs only when an error pattern is identical to a nonzero code vector of C,

$$Pu(E) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \qquad (1.6-1)$$

where p is the transition probability of the BSC. If the minimum distance of C is d_{min} then A_1 to A_{dmin-1} are zero.

An interesting asymptotic result is the error detection performance of an (n,k) code as $p \to 1/2$. In this case all received words are equally likely with probability 2^{-n}. Thus the probability of undetected sequence error is

$$Pu(E) = (2^k - 1)2^{-n} < 2^{-(n-k)} \qquad (1.6-2)$$

for n-k large. In addition, we see that

$$Pd(\text{prob. of detecting an error}) = 1 - 2^{-(n-k)} \qquad (1.6-3)$$

This means that while the throughput is very small, the

undetected sequence error rate can be made as small as desired by sufficiently increasing the number of redundant symbols in each word.

Consider the (7,4) Hamming code. The code contain k=4 information bits and n=7 coded bits, it has 2^4=16 codewords of a possible 2^7=128 words. It can be shown that the code words in the code differs by 3 bits. Thus the minimum distance d_{min}=3. The weight distribution of this code [8] is A_0=1, A_1=A_2=0, A_3=A_4=7, A_5=A_6=0, and A_7=1. The probability of an undetected error is

$$Pu(E) = 7p^3(1-p)^4 + 7p^4(1-p)^3 + p^7 \qquad (1.6-4)$$

If $p=10^{-2}$, this probability is approximately 7×10^{-4}. In other words, if 1 million code words are transmitted over a BSC with probability $p=10^{-2}$, there are on the average seven erroneous code words passing through the decoder without being detected.

1.6.1-2 Capability of Random Error-Correcting Codes

If a block code C with minimum distance d_{min} is used for random error correcting one would like to know how many

errors the code is able to correct. The minimum distance d_{min} is either odd or even. Let t be a positive integer such that

$$2t+1 < d_{min} < 2t+2 \qquad (1.6-5)$$

Next, we show that the code C is capable of correcting all the error patterns of t or fewer errors. Let v and r be the transmitted code vector and received vector,respectively. Let u be any other code vector in C. The Hamming distances among v,r,and u satisfy the triangle inequality:

$$d(v,r)+d(r,u) \geq d(u,v) \qquad (1.6-6)$$

Suppose that an error pattern of t' errors occurs during the transmission of v. Then the received vector r differs from v in t' places and therefore d(v,r)=t'. Since v and u are coded vectors in C, we have

$$d(u,v) \geq d_{min} \geq 2t+1 \qquad (1.6-7)$$

Combining the above and using the fact that d(v,r)=t', we obtain the following inequality:

$$d(u,r) \geq 2t+1-t' \qquad (1.6-8)$$

If $t' \leq t$, then

$$d(u,r) \geq t \qquad (1.6-9)$$

The inequality above says that if an error pattern of fewer errors occur, the received vector r is closer to the transmitted code vector v then to any other code vector u in C. For the BSC, these means that the conditional probability p(r|v) is greater than the conditional probabilty p(r|u) for u=v. Based on the maximum likelihood decoding scheme, r is decoded into v, which is the actual transmitted code vector.

On the other hand, the code is not capable of correcting all error patterns of ξ errors with $\xi > t$, for there is at least one case where an error pattern of ξ errors results in a received vector which is closer to an incorrect code vector than to the actual transmitted code vector. To show this, let u and v be two code vector in C such that

$$d(u,v) = d_{min} \qquad (1.6-10)$$

Let e_1 and e_2 be two error patterns that satisfy the

following conditions:

(i) $e_1+e_2=u+v$

(ii) e_1 and e_2 do not have nonzero components in common places.

Obviously, we have

$$W(e_1)+W(e_2)=W(u+v)=d(v,u)=d_{min} \qquad (1.6-11)$$

Now suppose that v is transmitted and is corrupted by the error e_1. Then the received vector is

$$r=v+e_1 \qquad (1.6-12)$$

The Hamming distance between v and r is

$$d(v,r)=W(v+r)=W(e_1) \qquad (1.6-13)$$

The Hamming distance between u and r is

$$d(u,r)=W(u+r)=W(u+v+e_1)=W(e_2) \qquad (1.6-14)$$

Now suppose that the error pattern e_1 contains more then t errors. Then since $2t+1 < d_{min} \leq 2t+2$, it follows that

$$w(e_2) \leq t+1 \qquad (1.6-15)$$

Combining the above and using the fact that $w(e_1) > t$ and $w(e_2) \leq t+1$, we obtain the following inequality :

$$d(v,r) \geq d(u,r) \qquad (1.6-16)$$

This inequality says that there exists an error pattern of l (l > t) errors which results in a received vector that is closer to an incorrect code vector than to the transmitted vector.

Summarizing the results above, a block code with minimum distance d_{min} guarantees correcting all errors of $t=[(d_{min}-1)/2]$ or fewer errors, where $[(d_{min}-1)/2]$ denotes the largest integer no greater than $(d_{min}-1)/2$. The parameter $t=[(d_{min}-1)/2]$ is called the random-error-correcting capability of the code. The code is referred as a t-error-correcting code. For example the (7,4) Hamming code has $d_{min}=3$ and thus t=1. It is capable of correcting any error pattern of single error over a block of seven digits.

If a t-error-correcting block code is used strictly for error correcting on a BSC channel (hard decision) with transition probability p, the block error probability,

P_{block} is the probability that more then t errors occured. since there are $\binom{n}{i}$ different ways of having i errors in n symbols, the block error probability is

$$P_{block} = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \qquad (1.6-17)$$

The bit probabilty depends on the particular code and decoder. The bit error probability for systematic codes can be estimated by assuming that the error rate of the corrected received sequence is equal to the error rate of the encoder input information symbol sequence. Then the bit error probability can be expressed [17] as

$$P_b = \left(\frac{1}{n}\right) \sum_{i=t+1}^{n} \beta_i \binom{n}{i} p^i (1-p)^{n-i} \qquad (1.6-18)$$

where i is the average number of symbol errors remaining in the corrected received sequence given that the channel caused i symbol errors. Of course $\beta_i = 0 \text{ for } i \leq t$. When i > t, β_i can be bounded by noting that when more than t errors occur a decoder which can correct at most t errors would at best correct t of the errors and at worse add t errors. So

$$i-t \le \beta_i \le i+t \quad , \quad i>t \qquad (1.6-19)$$

The decoder performance can be slightly improved by passing the received sequence unchanged when the corrected received sequence is not a valid code word. In either case for the majority of codes for which β_i has not been determined, $\beta_i = i$ is a good approximation.

The block code error probability formulas presented thus far have been for hard decision methods. Decoders capable of using soft decision technique are possible, but they are more difficult to implement. The simplest type of soft decision is the erasure decoding. Forney [6] has shown how to use the erasure information to improve the codes performance. This type of decoding is called erasure-and-errors decoding. The number of erasures that a code can correct in a codeword is

$$E \le d_{min} - 1 \qquad (1.6-20)$$

If errors and erasures can both occur in a codeword then a code with dmin can correct a combination of:

$$E + 2t < d_{min} \qquad (1.6-21)$$

For example, if dmin=8,then the code will be able to correct 2 errors and 3 erasures. So if the probability of an erasure is p_x and the probability of the channel error is p_e, then the block error probability is

$$P_{block} = \sum_{i=0}^{t} \sum_{j=d-2i}^{n-i} \binom{n}{j,i} p_e^i p_x^j (1 - p_x - p_e)^{(n-i-j)} + \sum_{i=t+1}^{n} \binom{n}{i} p_e^i (1 - p_i)^{n-i} \qquad (1.6-22)$$

where

$$\binom{n}{i,j} = \frac{n!}{i!\,j!\,(n-i-j)!} \qquad (1.6-23)$$

1.6.2 Random Error-Correcting Capabilities of Tree Codes

A Maximum-likelihood decoder (Viterbi decoder) is the optimal decoding technique available for convolutional codes. To be able to evaluate convolutional random error correcting capabilities we will analyze the performance of the Viterbi algorithm for a BSC channel with transition probability p.

We say that the first event error made at arbitrary time unit j is if the all-zero path (the correct path) is eliminated for the first time unit j in favor of a competitor path (the incorrect path). If the incorrect path has weight d, a first event error is made with probabilty

$$P_d = \begin{cases} \sum_{e=(d+1)/2}^{d} \binom{d}{e} p^e (1-p)^{d-e} & d \text{ odd} \\ \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} + \sum_{e=(d/2)+1}^{d} \binom{d}{e} p^e (1-p)^{d-e} & d \text{ even} \end{cases} \qquad (1.6-24)$$

Since all incorrect paths of length j or less can cause an error at time unit j, P(E) can be over bounded, using the union bound, by the sum of the error probabilities of each of this paths.

$$P(E) < \sum_{d=d_{free}}^{\infty} A_d P_d \qquad (1.6-25)$$

where P(E)is the event error probability at any time unit, and A_d is the number of code words of weight d.

The above bound can be simplified by noting that for d odd,

$$
\begin{aligned}
P_d &= \sum_{e=(d+1)/2}^{d} \binom{d}{e} p^e (1-p)^{d-e} \\
&< \sum_{e=(d+1)/2}^{d} \binom{d}{e} p^{d/2} (1-p)^{d/2} \\
&= p^{d/2}(1-p)^{d/2} \sum_{e=(d+1)/2}^{d} \binom{d}{e} \\
&< p^{d/2}(1-p)^{d/2} \sum_{e=0}^{d} \binom{d}{e} \\
&= 2^d p^{d/2}(1-p)^{d/2}
\end{aligned}
\tag{1.6-26}
$$

For small p the bound is dominated by its first term (i.e., the free distance term) and the event error probability can be approximated as

$$
P(E) \approx A_{d_{free}} \left(2\sqrt{p(1-p)}\right)^{d_{free}} \approx A_{d_{free}} 2^{d_{free}} p^{d_{free}/2} \tag{1.6-27}
$$

The event error probability bound above can be modified to provide a bound on the bit error probability, $P_b(E)$. Each event error causes a number of information bit errors equal to the number of nonzero information bits on the incorrect path. Hence if each event error probability term P_d is

weighted by the number of nonzero information bits on the weight d path, or, if there is more than one weight d path,by the total number of nonzero informationbits on all weight d paths, a bound on the expected number of information bit decoding errors made at time unit results. This can then be divided by k, the number of information bits per unit time, to obtain a bound on $P_b(E)$. In other word the bit error probability is bounded by

$$P_b(E) < \frac{1}{k} \sum_{d=d_{free}}^{\infty} B_d P_d \qquad (1.6-28)$$

where B_d is the total number of nonzero information bits on all weight d paths. For small p we can assume only the first term in the summation and the bit probability can be bounded by

$$P_b(E) \approx \frac{1}{k} B_{d_{free}} \left(2\sqrt{p(1-p)}\right)^{d_{free}} \approx \frac{1}{k} B_{d_{free}} 2^{d_{free}} p^{d_{free}/2} \qquad (1.6-29)$$

1.7 CODING BOUNDS

1.7.1 Bounds on Random-Error Correcting Codes

1.7.1-1 General Comments on Bounds

There are essentially two kinds of coding bounds. These are bounds on the minimum distance and bounds on the performance. The bounds on the minimum distance are obtained examining certain of the structural aspect of codes. The principal useful bounds are the the Hamming or sphere bound, the Plotkin bound, the Elias bound, the Gilbert-Varshamov bound and the new, lowest upper bound by McElice . These bounds indicate the maximum possible minimum distance for a given code length and code rate. However, the Gilbert-Varshamov bound is an achievable bound and thus provides a lower bound on the minimum distance of the best code. These bounds on the minimum distance are often used when attempting to find new codes to indicate how close one is to the best that is possible.

The bounds on performance are all based on random coding arguments. They indicate that the average performance of all block codes exhibits a probability of errors that decreases exponentially with code length. This, of course,implies the

existence of specific codes which are better than the average. Although random coding bounds show that it is possible to drive the error rate to zero in a manner that decreases exponentially with n they do not provide useful information in selecting such codes. Also, these bounds are not very useful for estimating the absolute performance of a code since good code exhibit a probability of error that is considerably better than predicted by the bound.

Before proceeding, there is an interesting digression that is of little practical importance but have frustrated researchers in the coding area for many years. We would assume that if one had a scheme for correcting a fixed fraction, t/n, of erroneous symbols per block, then the error rate could be made arbitrarily small by simply choosing the block to be long enough. Unfortunately, this turns out to be a very difficult task. Most constructive procedures encounter the problem that they only maintain a constant ratio t/n at the expense of an increasing percentage of redundant symbols (or equivalently, $R \to 0$ as $n \to \infty$). Thus the loss of efficiency occurs because the relative number of useful messages that these schemes convey becomes vanishingly small for long block length. A partial solution to this problem was given by Justesen [2] in 1972.

Justesen has shown that it is possible to construct a class of codes which is asymptotically good (in the sense describe above) and for which decoding procedure can be specified.

Now consider codes for a BSC channel. By Shannon's fundamental theorem for the noisy channel, it is known that for any rate R=k/n which is less than the channel capacity C there exist codes for which the probability of error is arbitrarily small. Yet only two nonrandom coding systems, Elias's error-free coding and Forney's concatenated BCH codes, are known to achieve an error probability which approaches zero and also maintain a rate that does not approach zero as the code length n is increased.

There is one additional noteworthy fact. The number of errors occurring in a block of length n symbols in a BSC has a binomial distribution with mean nP and variance nP(1-P). The fraction of position in error then has mean P and variance P(1-P)/n. As n approaches infinity the variance approaches zero, and for a fixed value of P_0>P, the probability that P_0n or more errors will occur approaches zero. On the other hand if P_0 < P, the probability that P_0n or more errors will occur certainly does not approach zero. A code that has minimum distance d, corrects all errors

patterns of weight t=[(d-1)/2] or less and no others. If $t=nP_0$, the probability of error approaches zero with increasing n if and only if $P_0 > P$. The rate for such a code is bounded by the Elias bound to a value below channel capacity. In other words, a coding system which achieves channel capacity must do so by correcting most error patterns of weight somewhat greater than [(d-1)/2] even though it is possible to correct all patterns up to the weight [(d-1)/2].

1.7.1-2 Specific Bounds

The coding bound on the minimum distance of codes are equally applied for block and convolutional codes [12]. The only different is the way we measure the distances in the codes. For block codes the distance is considered to be d_{min}, for convolutional codes it is d_{free}, as ,defined above.

Hamming Bound

For a fixed n the two upper bounds are the Hamming and the Plotkin bound. For the Hamming bound it can be stated that the maximum number of code words that can be provided for a given n and t is:

$$n-k \geq \log_q \left[\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \right] \qquad (1.7-1)$$

Plotkin Bound

The Plotkin bound is also an upper bound on the minimum distance that can be achieved for a given n and k, and is:

$$k \leq n - \frac{q d_{min} - 1}{q-1} + 1 + \log_q d_{min} \qquad (1.7-2)$$

The Hamming bound tend to provide the tightest bound for high rate codes R > 0.4, while the Plotkin bound is more appropriate for low rate codes. See Fig 1.7-1 for the binary case.

Gilbert-Varshamov Bound

The Gilbert-Varshamov bound states that it is always possible to find an (n,k) code with minimum distance at least equal to d where n,k, and d satisfy the inequality :

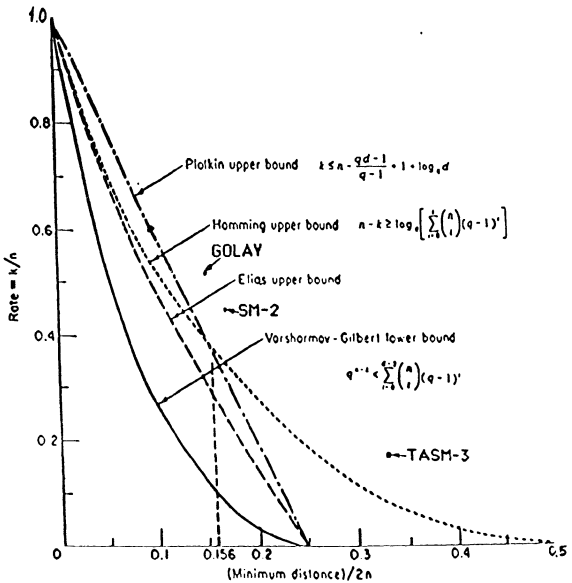


Figure 1.7-1 Bounds on minimum distance for the best binary bloc (The code length n is assumed to be very large.)

$$q^{n-k} < \sum_{i=0}^{q-2} \binom{n}{i} (q-1)^i \qquad (1.7-3)$$

The asymptotic bounds for a BSC is shown in Fig 1.7-1. Note that these coding bounds are for very long codes ($n \to \infty$). We have shown the Golay code and an SM-2 and TASM-3 as examples. Note that the Golay code, a perfect code, exceeded the bounds. In this case, so did the SM codes since n is small and not infinite.

1.7.1-3 Codes and Bounds.

Whenever a bound is obtained, it is natural to examine the cases, if any, when the bound is tight.

Perfect codes- A perfect code is one for which there are equal radius spheres about the codewords that are disjoint and that completely fill the space. Codes which satisfy the Hamming bound with equality were perfect codes. Example of such codes are the binary repetition code of odd block length n, the single Hamming error-correcting code, and the (23,12) Golay code.

Perfect codes, when they exist, have nice properties and are aesthetically pleasant,but they are rare.

Quasi-Perfect codes- A quasi-perfect code is one with spheres of radius t about each codeword are disjoint and all

words not in such a sphere are at a distance t+1 from at least one codeword.

Quasi-perfect codes are more common than perfect codes. An example of such codes are the double-error-corrective binary BCH codes. It can be shown that very long codes cannot be quasi-perfect unless they are very high rates.

The minimum distance of any linear (n,k)code satisfies Singelton's bound :

$$d_{min} \leq n - k + 1 \qquad (1.7-4)$$

Maximum distance separable codes- Any code that satisfies Singelton's bound with equality is a maximum distance separable code.

It can be shown that:

$$\text{If} \quad d = n + 1 - k > 2 \quad \text{then} \quad q \geq k + 1$$

and that

$$\text{If} \quad d = n + 1 - k < n - 1 \quad \text{then} \quad q \geq n - k + 1$$

According to the above equations no binary codes can be a maximum distance separable code unless k=1 (the binary repetition code) or d=2 (the single parity check code). Maximum distance separable codes exist over higher alphbets and the Reed-Solomon is such a code.

The minimum distance of any code which has K code-words of block length n over the alphabet of q letters is bounded by :

$$d_{min} \leq \frac{\bar{D}n}{1-K^{-1}} \quad \text{where} \quad \bar{D} = \frac{q-1}{q} \qquad (1.7-5)$$

In order for a code to satisfy the above equation with equality, its minimum distance must equal its average distance. that can happen only if the distance between every pair of codewords is the same.

Equidistant codes- Codes that satisfies the above equation with equality is a equidistant code.

A maximal-length shift register code is an equidistant code.

1.7.2 Bounds for Burst-Error Correcting and Detecing Codes

A burst of length ζ is defined as a vector whose only nonzero components are among ζ successive components, the first and the last of which are nonzero. For example, the error vector e=(0 0 0 1 0 0 1 1 0 0 1 0 0 0) is a burst of length 8. A linear code that is capable of correcting all burst errors of length ζ or less but not all error burst of

length $\zeta+1$ is called an ζ-burst -error-correcting-code.

It is clear that for a given code of length n and burst error correcting capability of ζ, we desired to construct an (n,k) code with as small a redundancy r=n-k as possible.

Theorem-(on the bound of burst error detecting capability of a code). For detecting all burst errors of length ζ r less with a linear block code of length n, ζ parity check symbols are necessary and sufficient.

Theorem-(Rieger 1960) [7]. In order to correct all burst errors of length b or less, a linear block code must have at least 2b parity check symbols. In order to correct all burst of length b or less and simultaneously detect all burst of length $\zeta \geq b$ or less, the code must have at least $b+\zeta$ parity check symbols.

Similary, every burst of length $b+\zeta$ or less can be written as the difference of burst of length ζ or less and a burst of length b or less. if the code is simultaneously to correct bursts of length b or less and to detect all bursts of length ζ or less, the number of parity check symbols of the code must be at least $b+\zeta$.

For the Theorem above it can be shown, that if erasure information is known we can replace the detecting bound on codes with erasure bounds on block codes.

For a given n,k code, Reiger's Bound implies that the burst-error-correcting capability of an (n,k) code is at most [(n-k)/2],that is,

$$b \leq \left\lfloor \frac{(n-k)}{2} \right\rfloor \qquad (1.7-6)$$

This is an upper bound on the burst-error-correcting capability of an (n,k) code. Code that meets the Reiger bound are said to be optimal.The ratio

$$z = 2\frac{b}{(n-k)} \qquad (1.7-7)$$

is used as a measure of the burst-error-correcting efficiency of a code. An optimal code has burst-correcting efficiency equal to 1.

1.8 STATE OF THE ART ON CODES.

1.8.1 State-of-The-Art of Random Codes.

Performance of random-error-correcting codes are bounded by the McEliece-Rodemich-Rumesey-Welch upper bound and by the Gilbert-Varshamov lower bound. A list of the best block codes known is given in Appendix A of MacWilliams and Sloane [3]. The best convolutional codes based by rate, constraint length and d_{free} is given in Proakis [4].

The most often implemented families of block codes are the:

1.8.1-1. The Hamming codes (n,k)

The Hamming codes have the following parameters:

$n=(q^m-1)/(q-1)$, $k=(q^m-1)/(q-1)-m$

$R=k/n=1-m/[(q^m-1)/(q-1)]$.

The Hamming is a single error-correcting code where the symbols are been taken from GF(q), and m is any integer. The symbol error probability is [17]:

$$p_{sym} = p - p(1-p)^{n-1} \qquad (1.8-1)$$

1.8.1-2. The optimal Golay code (23,12) and the extended Golay code (24,12).

The minimum distances of the codes are 7 and 8 respectively. The code is capable of correcting every 3 errors in a block. The bit error probability is :

$$\frac{1}{24}\sum_{i=4}^{24}\beta_i\binom{24}{i}p^i(1-p)^{24-i} \qquad (1.8-2)$$

where β_i is given in tables [12].

1.8.1-3. The binary BCH codes (n,k)

The binary BCH have the following parameters:

$n=(2^m-1)$, $k=(2^m-1)-Em$, and $R=k/n=1-Em/(2^m-1)$.

The BCH codes are multiple error correcting codes with the capability to correct E errors. The bit error probability is:

$$p_b=\frac{1}{n}\sum_{i=E+1}^{n}\beta_i\binom{n}{i}p^i(1-p)^{n-i} \qquad (1.8-3)$$

with β_i equal approximately to i.

1.8.1-4. The Reed-Solomon codes (n,k)

The Reed-Solomon code is a special nonbinary BCH code with maximal separable distance properties. The code is an E error correcting code with symbols taken from GF(q). It has the following parameters:

The block length:	n=q-1 and ($q=2^m$)
Number of parity check symbols:	n-k=2E
Minimum distance:	$d_{min}=2E+1$

The symbol error probability is:

$$P_{sym} = \frac{1}{n} \sum_{i=E+1}^{n} i \binom{n}{i} p_s^i (1-p_s)^{n-i} \qquad (1.8-4)$$

p_s is the channel symbol probability.
The bit error probability is:

$$P_b = \frac{2^{m-1}}{2^m - 1} P_{sym} \qquad (1.8-5)$$

In addition to correcting random symbols, the R-S codes can correct bursts of errors. Indeed, the R-S codes are

usually exploited to correct such error bursts. The R-S codes satisfies the Regier bound. Other burst codes are presented in Sec. 1.8.2.

1.8.2 State of The Art of Burst Codes.

1.8.2-1 Fire Codes

The first class of known burst error correcting codes are the Fire codes. The Fire codes are cyclic systematic codes generated by a polynomial g(x)

$$g(x)=\left(x^{2\zeta-1}+1\right)P(x) \qquad (1.8-6)$$

where P(x) is an irreducible polynomial of degree m over GF(2), and ζ is a positive integer such that $\zeta \leq m$ and $2\zeta-1$ is not divisible by ρ. where ρ is the smallest integer such that P(x) divides $x^{\rho}+1$. The length n of this code is the least common multiple of $2\zeta-1$ and ρ. that is,

$$n=LCM(2\zeta-1,\rho) \qquad (1.8-7)$$

The number of parity check bits of this code is $m+2\zeta-1$.

The efficiency of a Fire code is :

$$z = 2\zeta/(m + 2\zeta - 1) \qquad (1.8-8)$$

If ζ is chosen to be equal m then z=2m/(3m-1). For large m, z is approximately 2/3. Thus, Fire codes are not very efficient with respect to the Reiger bound.

In addition to Fire codes, some very efficient cyclic codes and shortened cyclic codes for correcting single bursts have been found either analytically or by computer search. A list of these codes is given in the [8]

1.8.2-2 Array Codes

In the November 1986 Issue of the Transactions on Information theory [5], a paper on burst error correcting codes introduced an array code, which is arranged as shown in Fig. 1.8-1 with the following parameters:

$$n = (k_2 + 1) \cdot (k_1 + 1) \qquad (1.8-9)$$

and

$$k = k_2 \cdot k_1 \qquad (1.8-10)$$

The code is capable of correcting a single burst of k_1

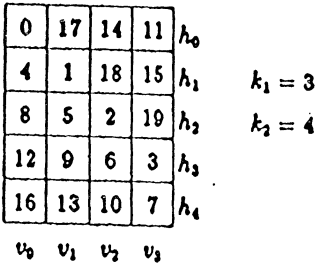


Figure 1.8-1 (20,12), $b = 3$ array code. 16,13,10,11,15,19,3,7: checks, readout order: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow \cdots \rightarrow 19$.

errors, Iff $k_2 \geq 2(k_1-1)$. The efficiency of this code is:

$$z = 2k_1/(k_2+k_1+1) \qquad (1.8-11)$$

If k_2 is chosen to be equal to $2(k_1-1)$ then the efficiency of the code is:

$$z = 2k_1/2(k_1-1)+k_1+1 = 2k_1/(3k_1-1) \qquad (1.8-12)$$

For large k_1 z is approximately 2/3.

1.8.2-3 Interleaved Codes

A technique for generating burst error correcting codes is the Interleaved code technique. Given an (n,k) code, it is possible to construct a $(\lambda n, \lambda k)$ code (i.e.,a code λ times as long as with λ times as many information digits) by interleaving. This is done simply by arranging λ code vectors in the original code into λ rows of a rectangular array and then transmitting them column by column as in Fig. 1.8-2. This is called an interleaved code, and the parameter λ is referred to as the interleaving dgree.

Obviously, a pattern of errors can be corrected for the whole array if and only if the pattern of errors in each row

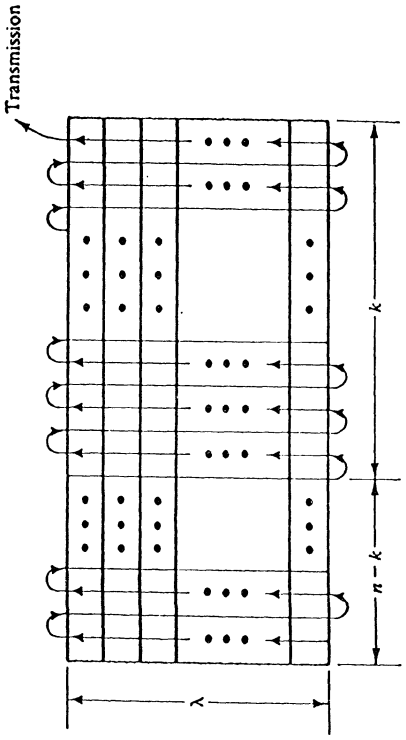


Figure 1.8-2 Transmission of an interleaved code.

is a correctable pattern for the original code. No matter where it starts, a burst of length λ will affect no more than one digit in each row. Thus, if the original code corrects single errors, the interleaved code will correct a burst of length λ or less. If the original code can correct any single burst of length ζ or less, then the interleaved code will correct any burst of length $\lambda \cdot \zeta$ or less. If an (n,k) code is a perfect burst error correcting code then an interleaved code made of that code is also perfect. Thus, by interleaving short codes with maximum possible burst error correcting capability, it is possible to construct codes of any length with maximum burst error correcting ability. This technique reduces the need to search long efficient burst error correcting codes.

The interleaving technique is effective not only for deriving long powerful single-burst-error-correcting codes, but also for deriving long powerful burst-and-random-error-correcting codes from short codes.

1.8.3 State-of-The-Art of Burst and Random Error Correcting Codes

On many communication channels, errors occur neither independently at random nor in a well defined single burst,

but occur in a mixed manner. To combat these kind of errors, the random error correcting codes or the single burst error correcting codes will be either inefficient or inadequate. Thus, it is desirable to design codes that are capable of correcting random errors and/or single or multiple bursts. There are several methods of constructing codes for those purposes. The most effective method is the interleaving technique described above. By interleaving a t error correcting code to a degree (λ, we obtain a $\lambda n, \lambda k$) code which is capable of correcting any combination of t bursts of length λ or less. Other methods that handle the problem are described below.

1.8.3-1 Product Codes

It is possible to combine the use of two or more codes so as to obtain a more powerful code. Let C_1 be an (n_1,k_1) linear code and let C_2 be an (n_2,k_2) linear code. Then the (n_1n_2,k_1k_2) linear code can be formed such that each code word is a rectangular array of n_1 columns and n_2 rows in which every row is a code vector in C_1 and every column is a code vector in C_2, as shown in Fig. 1.8-3. This two dimensional code is called a product code of C_1 and C_2. The check on check bits $(n_1-k_1)\times(n_2-k_2)$ shown in Fig 1.8-3 can be

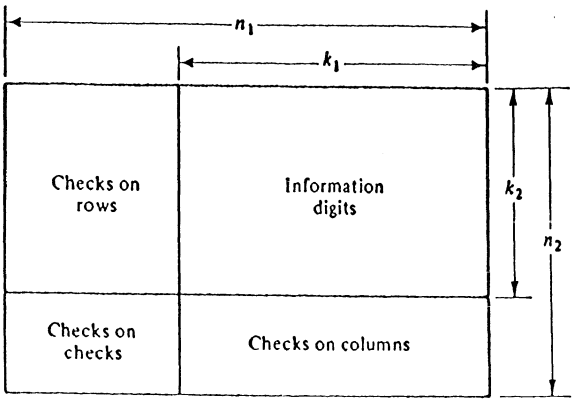


Figure 1.8-3 Code array for the product code $C_1 \times C_2$.

calculated either by using the parity check rules for C_2 on columns or by using the parity check rules for c_1 on rows, and it is possible to prove that either way will yield the same result. Thus, it is possible to have all row code vectors in C_1 and all column vectors in C_2 simultaneously. Therefore, for encoding the product code $C_1 \times C_2$, we may first encode the k_2 rows of the information array based on the parity check rules for C_1 and then encode the n_1 resulting columns based on the rules for C_2, or vice versa. Product codes can also be generalized to three or higher dimensional arrays.

Several observations can be made about the error correcting ability of these codes:

Theorem (Elias).-The minimum weight for the product of two codes is the product of the minimum weights for the codes.

Corollary -The minimum weight for the product of m codes is the product of the minimum weights for the codes.

Theorem (Burton and Weldon).-The product of an (n_1,k_1) code with error-correcting ability $t_1 = [(d_1-1)/2]$ and an (n_2,k_2) code with $t_2 = [(d_2-1)/2]$ can correct all patterns of weight

$t = [(d_1 d_2 - 1)/2]$ or less and all bursts of length up to;

$$b = \max(n_1 \cdot t_1, n_2 \cdot t_2) \qquad (1.8-13)$$

Corollary -If code C_1 has random-error-correcting capability t_1 and burst-error-correcting capability b_1 and code C_2 has random-error-correcting capability of t_2 and burst error-correcting capability b_2. Then the burst-error-correcting capability b of their product code C_1 and C_2 is at least equal to $\max(n_1 t_2 + b_1, n_2 t_1 + b_2)$.

$$b \geq \max(n_1 t_2 + b_1, n_2 t_1 + b_2) \qquad (1.8-14)$$

The method of correcting one code and then the other, will not achieve the random error correcting performance, because of the limitation of each code. For example, consider the product of two Hamming codes. The minimum distance of each of them is 3, so the minimum distance of their product is 9. A pattern of 4 errors at the corners of the array, gives two errors in each of the two rows and the two columns, and therefore this pattern will not be corrected by simple correction on rows and columns.

Theorem-For a binary symmetric channel, if one code has

probability of error $f_1(P)$ and another $f_2(P)$, their product is capable of decoding with a probability of error no greater than $f_2[f_1(P)]$.

1.8.3-2 Concatenated Codes

Forney [5] (1966) devised a method of combining two codes to form a larger code in a manner that resembles product codes somewhat. Fig. 1.8-4 shows a general concatenated coding system. One code called the inner code, is an (n,k) code with symbols from GF(q). The other code, called the outer code, is an (N,K) code with symbols from $GF(q^k)$. Encoding is done as follow:

1. A block of kK information symbols is divided into K "units" k-bits, symbols.

2. Each of the k-bits symbols is considered to be a symbol in $GF(q^k)$. The K bit symbols are then encoded into an N symbol code word in the outer code.

3. Each of the N, k-bits symbols is encoded as information symbols into an n-symbol code word in the inner code.

The set of N code words of (n,k) inner code then is a code word in the concatenated (Nn,Kk) code.

Decoding is accomplished in the obvious way: First, decoding is done on the inner code. resulting in reducing N

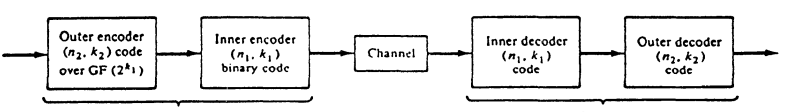


Figure 1.8-4 Communication system using a concatenated code.

inner-code words to N, k-bits symbols. Then these N "units" are decoded into K "units" by the decoding of the outer code.

It can be shown that the minimum distance of the concatenated code is $d_{min} = d_{min1} \times d_{min2}$, where d_{min1} is the minimum distance of the outer code, and d_{min2} is the minimum distance of the inner code. Further more the rate of the concatenated code is R=Kk/Nn.

A broader viewpoint of concatenated codes can be achieved by considering the concatenation of block codes and convolutional code or the concatenation of two convolutional codes.

As described above the outer code is usually chosen to be a nonbinary with each symbol selected from an alphabet of $q=2^k$ symbols. This code may be a block code, such as a Reed-Solomon code, or a convolutional code, such as a dual-k code. The inner code may be either binary or nonbinary and either block or a convolutional code.

1.9 AUTOMATIC-REPEAT-REQUEST (ARQ)

As pointed out earlier, there are two categories of techniques for controlling transmission errors in data transmission systems: the forward-error control (FEC) scheme

and automatic-repeat-request (ARQ) scheme. In the FEC system, an error-correcting code is used. When the receiver detects the presence of errors in a received vector, it attempts to determine the error locations and then corrects the errors. If the exact locations of errors are determined, the received vector will be correctly decoded; if the receiver fails to determine the exact locations of errors, the received vector will be decoded incorrectly and erroneous data will be delivered to the user. In an ARQ system, a code with good error-detecting capabilty is used. At the receiver, the syndrome of the received vector is computed. If the syndrome is zero, the received vector is assumed to be error-free and is accepted by the receiver. At the same time the receiver notifies the transmitter, via a return channel, that the transmitted vector has been successfully received. If the syndrome is not zero, errors are detected in the received vector. Then the transmitter is instructed, through the return channel, to retransmit the same code word. Retransmission continues until the code vector is successfully received. With this system, erroneous data is delivered to the data sink only if the receiver fails to detect the presence of errors. Using a proper code, the probability of an undetected error can be made as small as possible.

1.9.1 Basic ARQ Systems [15]

There are three basic types of ARQ schemes: the stop-and-wait ARQ, the go-back-N ARQ, and the selective-repeat ARQ. In a stop-and-wait ARQ data transmission system, the transmitter sends a code vector to the receiver and waits for the acknowledgment from the receiver as shown in Fig. 1.9-1. A positive acknowledgment (ACK) from the receiver signals that the code vector has been successfully received; and the transmitter sends the next code vector. A negative acknowledgment (NAK) from the receiver indicates that the received vector has been detected in error; the transmitter resends the code vector. Retransmissions continue until an ACK is received by the transmitter. This kind of system is simple and used in many data communicatin systems; IBM's widely used Binary Synchronous Communication (BISYNC) protocol is an example. However, the stop-and-wait ARQ scheme is inherently inefficient because of the idle time spent waiting for an acknowledgment for each transmitted code vector.

In a go-back-N ARQ system, code vectors are transmitted continuously. The transmitter does not wait for the acknowledgment after sending a code vector; as soon as he

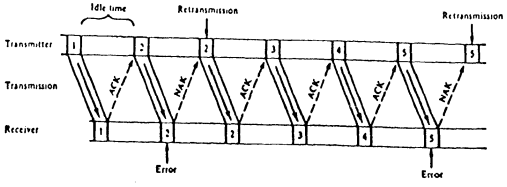


FIGURE 1.9-1 STOP-AND-WAIT ARQ.

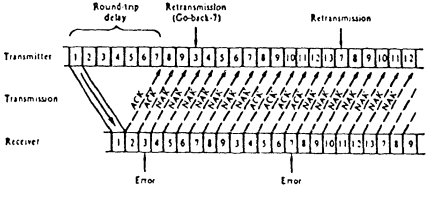


FIGURE 1.9-2 GO-BACK-N ARQ WITH N = 7

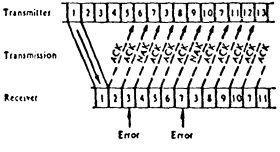


FIGURE 1.9-3 SELECTIVE-REPEAT ARQ

has completed sending one, it begins sending the next code vector as shown in Fig. 1.9-2. The acknowledgment for a code vector arrives after a round-trip delay. The round-trip delay is defined as the time interval between the transmission of a code vector and the receipt of an acknowledgment for that code vector. During this time interval, N-1 other code vectors have also been transmitted. When a NAK is received, the transmitter backs up to the code vector that is negatively acknowledged and resends that code vector and N-1 succeeding code vectors were transmitted during the round-trip delay. Of course, buffer must be provided at the transmitter for these code vectors. At the receiver, the N-1 received vectors erroneously received vector are discarded regardless of wether they are error free or not. Therefore the receiver needs to store only one received vector at a time. Because of the continuous transmission and retransmission of code vectors, the go-back-N ARQ scheme is more effective than the stope-and-wait ARQ. Communication protocols such as SDLC (Synchronous Data Link Control) and others employ the go-back-N ARQ scheme. The go-back-N ARQ scheme becomes ineffective when the round-trip delay is large and data transmission rate is high.

In a selective-repeat ARQ system, code vectors are also transmitted continuously. However, the transmitter only resends those code vectors that are negatively acknowledged as shown in Fig. 1.9-3. Since, ordinarily, code vectors must be delivered to the user in correct order, a buffers must be provided at the receiver to store the error-free received vectors following a received vector detected in error. When the first negatively acknowledged code vector is successfully received, the receiver than releases the error-free received vectors in consecutive order untill the next erroneously received vector is encountered. Sufficient receiver buffers must be provided; otherwise, buffer overflow may occur and data may be lost. The selective-repeat ARQ is the most efficient one among the three basic ARQ schemes described.

1.9.2 Performance of ARQ Schemes

In ARQ systems, the receiver commits a decoding error whenever it accepts a received vector with undetected errors. Such event is called an error event. Let P(E) denote the probability of an error event. Clearly for an ARQ system to be reliable, P(E) should be very small. Therefore, the reliability of an ARQ system is measured by its probability

P(E). Suppose that an (n,k) code C is used foe error detection in an ARQ system. Let us define the following probabilities:

P_c - probability that a received vector contains no error;

P_d - probability that a received vector contains a detectable error pattern;

P_e - probability that a received vector contains an undetectable error pattern.

These probabilities add to 1. The probability P_c depends on the channel error statistics, the probabilities P_d and P_e depends on both the channel error statistics and the choice of the (n,k) error detecting code C. The probability P(E) that a receiver commits an error is given by

$$P(E) = P_e / (P_c + P_e) \qquad (1.9-1)$$

The probabilty P_e can be made very small relative to P_c by choosing the code C properly. Consequently, the error probability P(E) can be made very small. For a BSC with transition probability p, we have

$$P_c = (1-p)^n \qquad (1.9-2)$$

In Section 1.6 we have shown that, for an average (n,k) code,

$$P_e < 2^{-(n-k)} \qquad (1.9-3)$$

by choosing n and k we can obtain an upper bound on P(E).

Another measure of performance of an ARQ system is its throughput efficiency. The throughput is defined as the ratio of the average number of information digits successfully accepted by the receiver per unit time to the total number of digits that could be transmitted per unit time. All three basic ARQ schemes achieve the same reliability; however they have different throughput efficiencies. The throughput efficiency is based on the following;

The probability that the received vector will be accepted by the receiver is

$$P = P_c + P_e \qquad (1.9-4)$$

For the usual situation where P_e << P_c then $P \approx P_c$. The probability that a code vector will be transmitted is simply

$$P_d = 1 - P \approx 1 - P_c \qquad (1.9-5)$$

Thus, for stop-and-wait ARQ, the average number of digits (including the idling effect) that the transmitter could have transmitted is

$$\begin{aligned} T_{sw} &= (n + D\tau)P + 2(n + D\tau)P(1 - P) + 3(n + D\tau)P(1 - P)^2 + \ldots \\ &= \frac{n + D\tau}{P} \end{aligned} \qquad (1.9-6)$$

Therefore, the throughput efficiency of the stop-and-wait ARQ system is

$$\eta_{SW} = \frac{k}{T_{SW}} = \frac{P}{1 + D\tau/n}\left(\frac{k}{n}\right) \qquad (1.9-7)$$

where D is the idle time from the end of the transmission of one code word to the beginning of transmission of the next. and τ the signaling rate of the transmition in bits per second.

In the go-back-N ARQ system, the average number of retransmissions required is

$$T_{GBN} = P + (N+1)P(1-P) + (2N+1)P(1-P)^2 + \ldots + (lN+1)P(1-P)^l + \ldots$$
$$= 1 + \frac{N(1-P)}{P} \qquad (1.9-8)$$

Consequently, the throughput efficiency of the go-back-N ARQ system is

$$\eta_{GBN} = \frac{1}{T_{GBN}}\left(\frac{k}{n}\right) = \frac{P}{P + (1-P)N}\left(\frac{k}{n}\right) \qquad (1.9-9)$$

For the selective-repeat ARQ system, the average number of retransmission required is

$$T_{SR} = P + 2P(1-P) + 3P(1-P)^2 + \ldots + lP(1-P)^{l-1} + \ldots$$
$$= \frac{1}{P} \qquad (1.9-10)$$

Then the throughput of the selective-repeat ARQ system is

$$\eta_{SR} = \frac{1}{T_{SR}}\left(\frac{k}{n}\right) = \left(\frac{k}{n}\right)P \qquad (1.9-11)$$

From the throughput performance just given, we see that the selective-repeat ARQ system is the most efficient scheme. The throughput of the selective-repeat ARQ does not

depend on the round-trip delay. In communication systems where the round-trip delay is large and data rate is high, the parameter N for go-back-N ARQ and the parameter D_T/n for stop-and-wait ARQ becomes very large. In this case, the throughput of the go-back-N ARQ drops rapidly as the channel error rate increases, while the throughput of the stop-and-wait ARQ becomes unacceptable.

1.9.3 Hybrid ARQ Schemes

Comparing the two error control schemes, we see that ARQ is simple and provides high system reliability. However ARQ systems have a severe drawback: their throughput fall rapidly with increasing channel error rate. Systems using FEC maintain constant throughput (equal to the code rate R=k/n) regardless of the channel error rate. However, FEC systems have two drawbacks: First, when a received vector is detected in error, it must be decoded and the decoded message must be delivered to the user regardless of whether it is correct or incorrect. Since the probability of a decoding error is much greater than the probability of an undetected error, it is hard to achieve high system reliability with FEC. Second, to obtain a high system reliability, a long powerfull code must be used and a large

collection of error patterns must be corrected. This makes decoding hard to implement and expensive. For these reasons, ARQ is often preferred over FEC for error control in data communication systems, such as packet-switching data networks and computer communication network. However, in communication systems where returns channels are not available or retransmission is not possible for some reason, FEC is the only choice.

The drawbacks in both ARQ and FEC could be overcome if two error control schemes are properly combined. Such a combination of the two basic error control schemes is referred to as hybrid ARQ. A hybrid ARQ consists of an FEC subsystem contained in an ARQ system. The function of the FEC portion is to reduce the frequency of retransmission by correcting the error patterns that occur most frequently. This idea increases the system throughput. When a less frequent error pattern occurs and detected, the receiver request a retransmission rather than passing the unreliably decoded message to the user. This increases the system reliability than an FEC system alone and a higher throughput than an ARQ system alone.

A straightforward hybrid ARQ scheme is to use a code, say, an (n,k) code, which is designed for simultaneous error

correction and error detection. When a received vector is detected in error, the receiver first attempts to locate and correct the errors. If the number of errors is within the codes capability, the errors will be corrected and the message will be passed to the user. If an uncorrectable error pattern is detected, the receiver rejects the received message and request a retransmission. When the retransmitted vector is received, the receiver again attempts to correct the errors (if any). If the decoding is not successful, the receiver again rejects the received vector and asks for another retransmission. This process continues until the vector is successfully received or decoded.

The hybrid ARQ scheme described above is referred to as type I hybrid ARQ schemes. Since a code used in this scheme must be able to correct a certain collection of errors and simultaneously detect other error patterns, more parity-check bits are needed. This increases the overhead for each transmission. As a result, when the channel error rate is low, it has a lower throughput than its corresponding ARQ scheme. However, when the channel error rate increases, the throughput of the ARQ system drops rapidly and the hybrid-ARQ cheme provides higher throughput. The type I

hybrid ARQ scheme is capable of maintaining significant high a over a wide range of channel error rates if the design of the FEC is appropriate.

2. DESCRIPTION OF CODES

2.1 INTRODUCTION

The object of the work is to provide a forward error correction (FEC) and detection code (EDC) method.

Another object of the work is to provide an FEC and a EDC method that is very efficient for burst and random channels.

A further object of the work is to provide an FEC for operating in an ARQ system using efficient "code combining" method.

A still further object of the work is to provide an FEC method and apparatus that easier to implement than many prior art codes.

An additional object of the work is to provide an FEC and detection code method and apparatus that is more powerful and less complex than many prior art codes.

According to the present work, as embodied and broadly described herein, a Schilling-Manela (SM) encoding method is provided comprising the steps of storing a block of data-symbol sequence in memory means having g rows by h columns of information-memory cells, calculating pari-

ty-check symbols from parity-line symbols having p-bits per symbol, along at least a first and a second set of parity lines. Each of the first set of parity lines can have a straight diagonal path with first slope through the g rows by h columns of the information-memory cells and each of the second set of parity lines can have a straight diagonal path with a slope through the g rows and h columns of the information-memory cells. Alternatively, each of the first and second set of parity lines can have curved paths through the g rows and h columns of the information-memory cells. The parity-check symbols are calculated by adding modulo p the parity-line symbols along each of the parity lines, respectively, and setting the parity-check symbol for each parity line equal to the modulo p sum of the parity-line symbols along each parity line. The parity-check symbols are stored in w parity-memory cells of the memory means, and an encoded-data-bit sequence is outputted comprising the data-bit sequence and the parity-check symbols.

The present work also includes a SM decoding method comprising the steps of storing an encoded data symbol sequence in memory means having at least g rows by h columns of information-memory cells and w parity-memory cells, wherein the encoded-data-symbol sequence includes a

parity-check-symbol sequence having w parity-check symbols stored in the w parity-memory cells, and a data-symbol sequence blocked and stored in the g rows by h columns of information-memory cells. The parity-check symbols and the parity-line symbols along the parity lines in the g rows by h columns of information-memory cells, having an error, are found. The count of each composite cell on a composite-error graph traversed by the path of each of the parity lines having an error is incremented and the largest-number cell in the composite-error graph having the largest number is determined. The largest number is compared to a threshold, and provided the largest number exceeds the threshold, a new data-symbol is determined for the memory cell in the information memory cells corresponding to the largest number cell in the composite-error graph having the largest number. The new-data symbol is chosen to minimize the count in the largest-number cell, and the new-data symbol is substituted into the stored data-bit sequence.

Without departing from the spirit or scope of the present work, the method of the present work may include the data-symbol sequence blocked and stored in a λ- dimensional memory. In this more general case, there would be sufficient rows and columns to correspond to the λ- dimensional system.

Further, parity-check symbols can be calculated from parity-line symbols along parity lines having curved paths passing through the λ- dimensional system.

2.2 DETAILED DESCRIPTION OF THE SM CODE.

Reference will now be made to the present preferred embodiments of the work, examples of which are illustrated in the accompanying drawings.

2.2.1 Encoding of the SM Code.

Referring to Fig. 2.2-1, a preferred embodiment of the SM error correcting and detecting code encoding method is shown comprising the steps of entering and storing [110] a block of data-bit sequence in memory means having g rows by h columns of information-memory cells. The memory means may be embodied as a memory including a random access memory, or any other memory wherein data readily may be accessed. The memory may include w parity-memory cells for storing w parity symbols. The methods includes calculating [112] parity-check symbols from parity-line symbols having p-bits per symbol. The plurality of parity-line symbols typically are along a plurality of parity paths. The parity paths may be parity lines, L_i. The parity check symbols S_i are stored

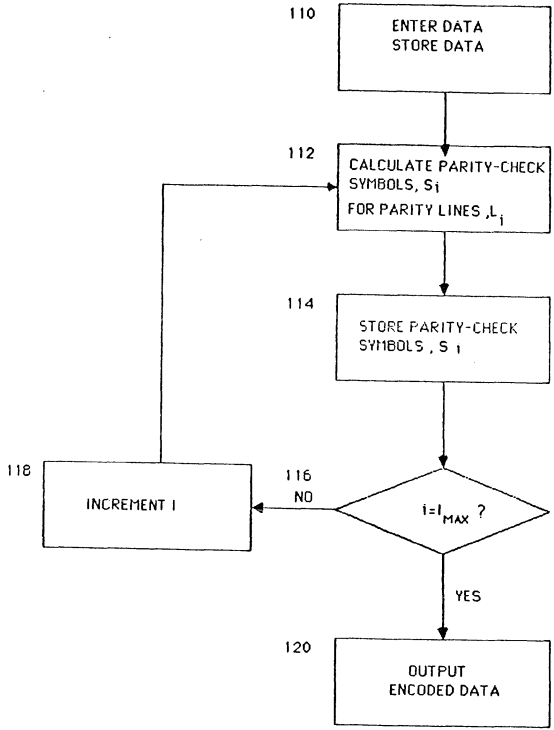


FIGURE 2.2-1 A FLOW DIAGRAM OF THE SM ENCODING METHOD

[114] in the w parity-memory cells. The method determines [116] whether all the parity-check symbols, I_{max}, have been calculated, and outputs [120] an encoded-data-symbol sequence comprising the data-symbol sequence and the parity-check symbols. If all the parity-check symbols, S_i, have not been calculated, then the process increments [118] to the next parity line and calculates the next parity symbol, S_{i+1}.

In a first preferred embodiment of the encoding method according to the present proposal, the parity-check symbols may be parity-check bit, and the parity-line symbols may be parity-line bits. In this first embodiment, the method may comprise calculating parity-check bits from parity-line bits, along a first and a second set of parity lines. Each of the first set of the parity lines can have a straight diagonal path with a first slope through the g rows by h columns of information-memory cells. Each of the second set of parity lines can have a straight diagonal path with a second slope through g rows by h columns of information-memory cells. Alternatively, each of the first and second set of parity lines can have curved paths through g rows by h columns of information-memory cells. The

parity-check bits are calculated by adding modulo 2 the parity-line bits along each of the parity lines, and setting the parity check bit for each parity line equal to the modulo 2 sum of the parity-line bits along each parity line, respectively. Accordingly, the parity-check bits are stored in the w parity-memory cells of the memory.

A second preferred embodiment of the SM error correcting and detecting encoding method using the flow chart of Fig. 2.2-1, includes entering the data to form a g rows by h columns data block in the information-memory cell. In an optimal mode the parity-check bits are calculated by starting with the parity lines having either all positive or all negative slopes. The first parity line starts with the bit (1,1) and the last parity line includes bit (g,h). The slope i is then incremented and the process is repeated. If negative slope lines are used then the same process is employed to calculate all parity check bits of each line of slope i. Now, however, the process starts with bit (1,h) and ends with bit (g,1). When $i=I_{max}$ the process is completed and all data bits and all parity check bits are transmitted. It is in the spirit of this work that each parity line includes each bit (or symbol) and it is not imported which order the parity lines are obtained.

The information data bits are first collected by the encoder from a g rows by h columns block of data in the information memory cells, as shown in Fig. 2.2-2a and 2.2-2b. Note, that the data can be entered into the encoder row-by-row, as illustrated in Fig. 2.2-2a or column-by-column, as illustrated in Fig. 2.2-2b. After the block of data is entered into the encoder, parity-check bits are added. To illustrate the algorithm for adding parity-check bits consider that the first h parity-check bits are obtained by the modulo 2 adding a column of data-bits and setting the parity-check bit so that parity-check bit is equal to the modulo 2 sum of the data bits. Thus, in Fig. 2.2-2b, for the i^{th} column, the i^{th} parity-check bits is :

$$r_i = \sum_{j=1 \, modulo \, 2}^{g} a_{ig+j} \qquad (2.2-1)$$

where i= 0, 1, 2,...,h-1.

Let us further assume that the second set of paricy-check bits are formed by modulo 2 adding the data bits along a 45^0 diagonal (slope=+1). Thus, in Fig. 2.2-2b,

$$r_i = \sum_{j=1 \, modulo \, 2}^{g} a_{(i+j)g-(j-1)} \qquad (2.2-2)$$

where i=-g+1, ... 0, 1, 2, ...,h-1.

a_1	a_2	a_3	. . .	a_h
a_{h+1}	a_{h+2}	a_{h+3}	. . .	a_{2h}
.	.	.		.
.	.	.		.
.	.	.		.
$a_{(g-1)h+1}$	$a_{(g-1)h+2}$	$a_{(g-1)h+3}$	. . .	a_{gh}

FIGURE 2.2-2a A G*H BLOCK OF DATA IN WHICH THE DATA WAS ENTERED ROW BY ROW.

a_1	a_{g+1}	a_{2g+1}	. . .	$a_{(h-1)g+1}$
a_2	a_{g+2}	a_{2g+2}	. . .	$a_{(h-1)g+2}$
.	.	.		.
.	.	.		.
.	.	.		.
a_g	a_{2g}	a_{3g}	. . .	a_{gh}

FIGURE 2.2-2b A G*H BLOCK OF DATA IN WHICH THE DATA WAS ENTERED COLUMN BY COLUMN.

According to the present proposal,parity-check bits can be formed from different diagonals and the diagonals can have a positive or a negative slope. The spirit of the work is that any selection of data bits, be it on a straight diagonal line or a curved line can be used to generate parity-check bits. each such line is called a parity line.

Augmentation by Zeros

Referring to the above equation, there are values of $a_{(i+j)g-(j-1)}$ which are non-existent . For example, consider i=h-1 and j=2. In this case one has $a_{(h+1)g-1}$. Since $g \cdot h$ is the largest subscript possible, $a_{(h+1)g-1}$. is non-existent. Whenever such a non-existent data bit is required, it is assumed to be a zero-bit. Assuming such data bits to be a 1-bit does not alter the spirit of the work but merely its implementation. Conceptually one can imagine that the data block shown in Fig. 2.2-2a, and 2.2-2b has 0-bits in all columns to the left and to the right of the data block. These 0-bits are not transmitted and therefore do not affect the code rate.

Code rate R:

The rate of the SM error-correcting and detecting code is a function of the number of rows g, columns h, and the number and the slope of parity lines, i.e., different sloped lines, r, used. The code rate R is approximately given (for h>>g) by :

$$R = \frac{g}{g+r} \qquad (2.2-3)$$

2.2.2 Decoding Method of the SM Code.

according to the present proposal a SM decoding method is provided, as illustrated in Fig. 2.2-3, includes the steps of entering, blocking and storing [210] an encoded-data-symbol sequence in memory means having at least g rows by h columns of information-memory cells, and w parity-memory cells with r parity lines. The memory means may be embodied as a memory including a random access memory, or any other memory wherein data readily may be accessed. The encoded-data-sombol sequence includes w parity-check symbols stored in the w parity-memory cells, and a data-symbol sequence blocked and stored in the g rows by h columns of information-memory cells.

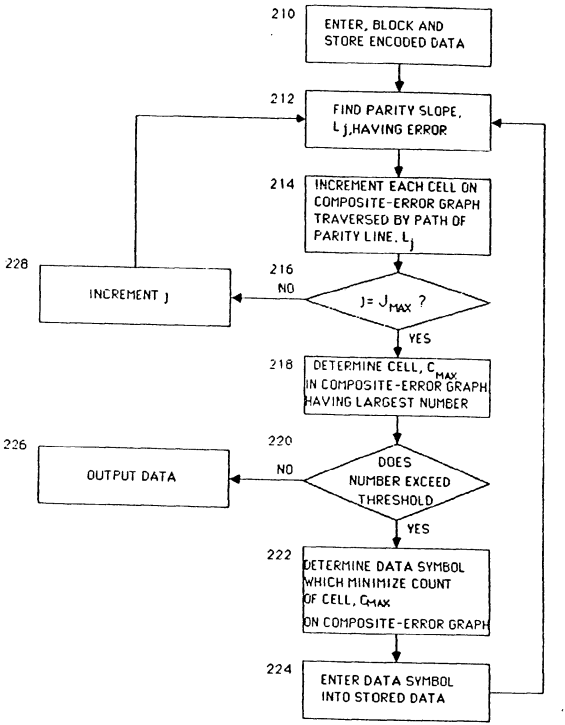


FIGURE 2 2-3 A FLOW DIAGRAM OF THE SM DECODING METHOD

The decoding method of the SM code, includes finding [212] the parity-check symbol and the parity-line symbol along the parity lines L_i in the g rows by h columns of information-memory cells,having an error. At the same time, the method uses a graph, and enters a 1-bit on each cell in the graph traversed by the path of each parity line L_i having an error. At the end of generating a graph for each parity line L_i having an error, the graphs generated from each parity line L_i are added together to make a composite-error graph. Alternatively, the method can employ a composite-error graph directly, and increment [214] the count of each composite cell on the composite-error graph traversed by the path of each of the parity lines having an error.

The next step in the SM decoding method according to the present work, determines [216] whether all the parity lines having an error have been entered onto the composite-error graph, and determines [218] the largest number cell in the composite-error graph having the largest number. Accordingly, the largest number is compared [220] to a threshold and, provided the largest number exceeds the threshold, a new-data symbol for the memory cell in the information-memory cells which corresponds to the largest number

cell in the composite-error graph having the largest number, is determined [222]. For optimum performance of burst error correction, the threshold might be set to be greater than or equal to 2. For optimum performance of random error correction, the thresholds might be set to be equal to r/2+1 if r is even or t (r+1)/2 for r odd. The new- data symbol is chosen so as to minimize the cell count in the largest number cell on the composite-error graph. The new-data symbol is substituted into the information-memory cells. In the event the largest number does not exceed the threshold then the SM decoding method outputs [226] the data.

A Random Error Correcting Example

To illustrate the SM decoding method, consider the encoded data block as shown in Fig. 2.2-4a which may be transmitted by row as shown in Fig. 2.2-4b. Note that the code rate is:

$$R = \frac{18}{18+6+8+8} = 0.45 \qquad (2.2-4)$$

The received codeword with errors is shown in Fig. 2.2-4c. In this particular embodiment of the SM decoding method, the parity-check symbols are parity-check bits and the parity-line symbols are parity-line bits. The modulo 2 sum of each and every parity line is computed for each slope. In

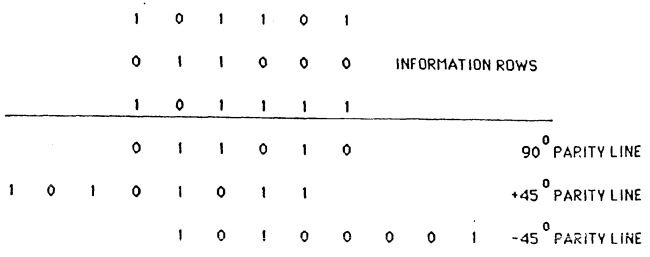


FIGURE 2.2-4a EXAMPLE OF AN ENCODED DATA BLOCK 3 PARITY CHECK LINES

ROW 1	ROW 2	ROW 3	ROW 4	ROW 5	ROW 6
101101	011000	101111	011010	10101011	10100001

FIGURE 2.2-4b AN ILLUSTRATION OF TRANSMISSION BY ROWS

this example, there are six vertical parity lines, eight 45^0 parity lines and eight -45^0 parity lines. If any parity line is in error a 1-bit is entered in the cells traversed by the path of the parity line, of a graph. If the parity line is not in error a 0-bit is entered in the cells traversed by the path of the parity line, of a graph. Fig. 2.2-4d shows the graphs after a vertical check, a $+45^0$ check and a -45^0 check was taken. Note that the vertical check "sees" an error in columns 3 and 5, and that 2 diagonals indicate errors in the $+45^0$ diagonal check and that 2 diagonals indicate errors in the -45^0 diagonal check.

A composite-error graph, shown in Fig. 2.2-4e, is formed. The graph includes composite cells showing the sum of the errors. The maximum number in a composite cell is equal to the number of parity check rows employed. A typical threshold might be t-1, indicating that peak values of 2 or 3 are used as error indicators. To correct an error each cell is searched to find the largest number on the composite error graph. If only a single composite cell contains this largest number, then the bit in that row and column is inverted and the procedure is repeated. In Fig. 2.2-4e row 2, column 3, is selected and that bit, (2,3) is inverted. The process is repeated using the partially decoded word

FIGURE 2.2-4c ILLUSTRATION OF A RECEIVED CODEWORD WITH TWO ERRORS

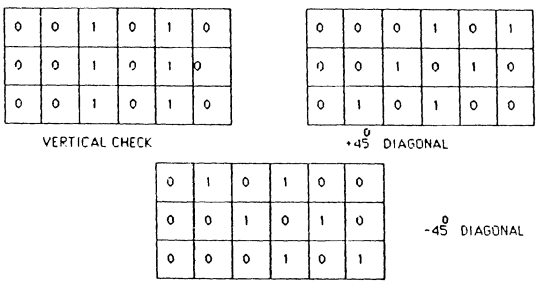


FIG. 2.2-4d ALL SLOPE CHECK GRAPH

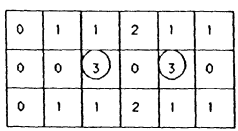


FIG. 2.2-4e ILLUSTRATION OF COMPOSITE-ERROR GRAPH

shown in Fig. 2.2-5.

Proceeding as above the next composite map is as shown in Fig. 2.2-6a and the next partial decoding is shown in Fig. 2.2-6b. Figure 2.2-7 shows that all errors have been corrected.

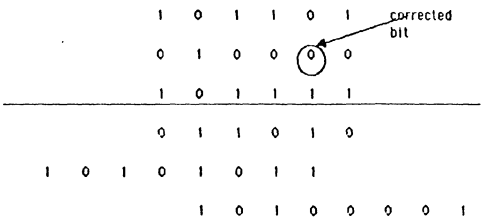
A Burst Error Correcting Example

To illustrate the burst error correction capability of the SM code, consider the data block and parity-check bits shown in Fig. 2.2-8a. Note that for optimal performance all parity lines must be in the same direction and should be interleaved with data rows. These bits are transmitted, row by row as shown in Fig. 2.2-8b. Interleaving is not shown in this example.

Figure 2.2-8c shows that the bits are received with row one in error.

Figure 2.2-8d shows a composite-error graph indicating a burst of errors. The procedure is to select the bit in row 1, column 1, for inversion when the parity line are positive.

Figure 2.2-8e shows the partially decoded word and Fig. 2.2-8f shows the resulting error graph. Next invert the bit adjacent to the 0 providing the threshold is exceeded. In



FIGURER 2.2-5 A PARTIALLY DECODED WORD

1	1	1	1	0	0
0	0	3	0	0	0
0	1	1	1	0	0

FIGURE 2.2-6a A RESULTING COMPOSITE-ERROR GRAPH

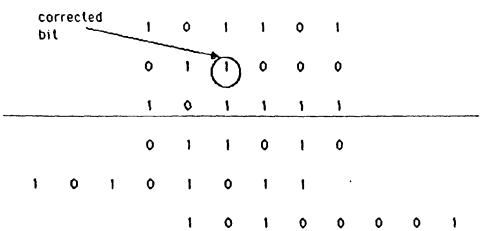


FIGURE 2.2-6b A RESULTING PARTIALLY DECODED WORD

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

FIGURE 2.2-7 A RESULTING COMPOSITE-ERROR GRAPH OF DECODED BLOCK

```
                                  1  0  1  1  0  1
                                  0  1  1  0  0  0
                                  1  0  1  1  1  1
SLOPE=+1 PARITY LINE    1  0  1  0  1  0  1  1
      1  0  1  0  0  1  1  1  1  1  SLOPE=+1/2 PARITY LINE
```

FIGURE 2.28a AN EXAMPLE OF AN ENCODED DATA BLOCK (TWO PARITY CHECK LINES)

```
ROW 1      ROW 2      ROW 3      ROW 4      ROW 5
1011101    0111000    1011111    1010 1011  1010011111
```

FIGURE 2.2-8b AN ILLUSTRATION OF TRANSMITTED DATA BY ROW

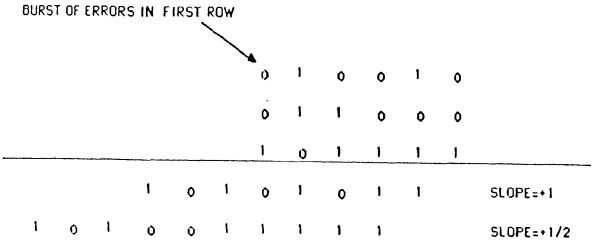


FIGURE 2.2-8c AN ILLUSTATION OF A RECEIVED CODEWORD WITH BURST ERROR

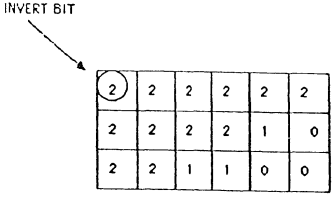


FIGURE 2.2-8d AN ILLUSTRATION OF THE COMPOSIT-ERROR GRAPH

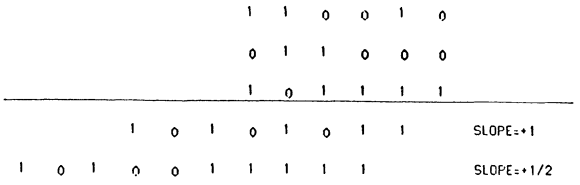


FIGURE 2 2-8e AN ILLUSTATION OF A PARTIALLY CORRECTED BIT PATTERN

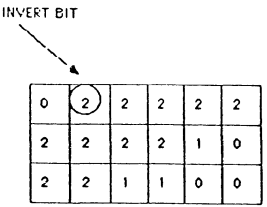


FIGURE 2 2-8f AN ILLUSTRATION OF THE PARTIALLY CORRECTED ERROR GRAPH

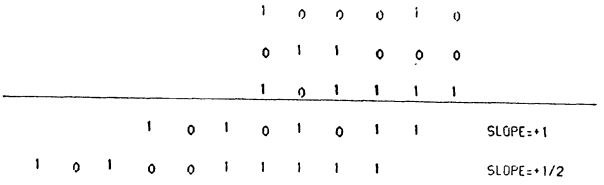


FIGURE 2.2-8g AN ILLUSTATION OF ADDITIONAL CORRECTION

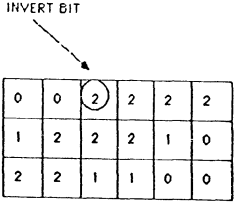


FIGURE 2.2-8h AN ILLUSTRATION OF ADDITIONAL CORRECTION

this case t=1. Figure 2.2-8g shows the partially decoded word and Fig. 2.2-8h indicates the cell exceeding the threshold adjacent to the 0. This procedure continues until the maximum number in a cell in less than the threshold. For this example the process continues until the number in the cells decreases below 2.

2.2.3 An SM Convolutional Encoder Implementation.

In Fig. 2.2-9 a (4,2) systematic convolutional encoder is shown. The encoder accept every unit T two bits of information and encode them into four code bits. The rate of the code R=1/2. The encoder described in Fig 2.2-9 is also an SM code with two information lines and two parity check lines, where the slopes of the parity check bits are 1 and 1/2. The encoding process is perform on the information bits. It consists delaying the data using shift registers and, modulo 2 addition which is equivalent to a logic XOR operation. Another example of an SM encoder of rate R=1/2 having 4 information bits and four parity check bits is shown in Fig. 2.2-10. The slopes of this implementation are $\infty, 1, 1/2, 1/3$. In this example every unit T 4 bits are entered into the encoder and 8 bits are delivered to the output. Every encoding of an SM code can be done base on the

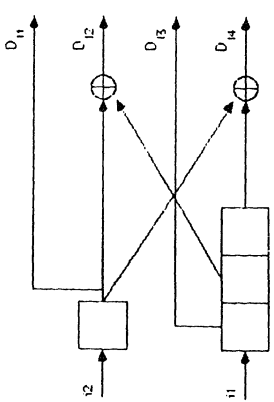


Figure 2.2-9 RATE HALF ENCODER 2 PARITY CHECK BITS (D=1 ,D=2)

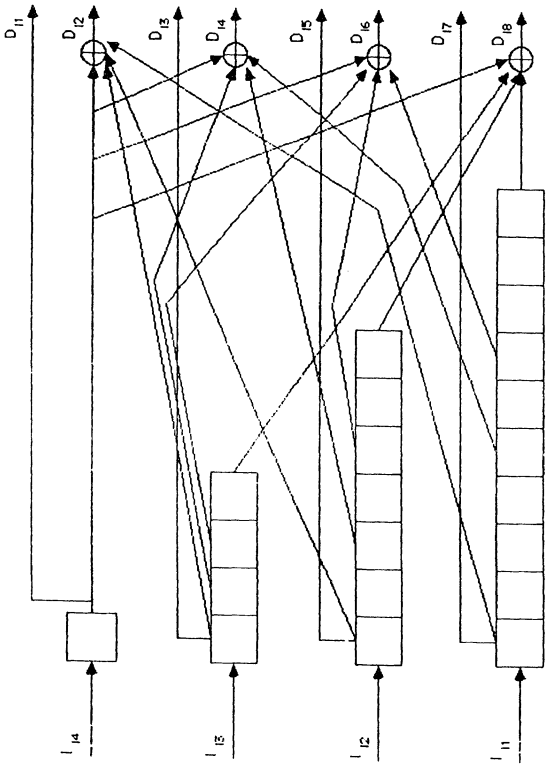


FIGURE 2.2-10 RATE HALF ENCODER 4 PARITY CHECK BITS (V, D1, D2, D3)

convolutional encoding procedure. For a code with g data lines and r parity check lines the convolutional encoder will accept every unit time T g bits and will produce g+r bits at the output. The output bits consist of the g input bits and their corresponding r parity check bits calculated by the encoder. The slopes of the code can be arbitrary, but as described earlier we would choose the slopes with the values of 1/(integer). This way of choosing the slopes guarantees that every bit in the data is been checked exactly r times, which create a symmetry configuration. The shift registers length is a function of g, and the slopes. The number of stages in each shift register can be calculated by the following formula:

$$l_k = \frac{1}{S_k} \cdot (g-1) + 1 \qquad (2.2\text{-}5)$$

where S_k is the value of the slope of parity check row k.

The encoding process starts when the content of the shift registers is "0". For a code of size h as in Fig. 2.2-2a the data is entered into the encoder h times to produce the $h \times (g+r)$ encoded message. To complete the encoding process "0" are entered to the delay lines and the

parity check bits are augment to the encoded message. The augmentation is performed l_r times. Where l_r is the number of stages in the longest shift register.

2.3 PARTIAL AUTOCONCATENATION SM CODE (PASM)

By looking at the SM code it is clear that the most vulnerable bits in the code are the parity check bits. Special error constellations in the parity check bits may generate errors in the information bits. Thus, a way to improve the codes performance is to protect the parity check bits as well as the information bits.

2.3.1 Encoding of the PASM Code.

The partially autoconcatenated SM code (PASM) is encoded using the flow diagram shown in Fig. 2.3-1. The input data is first stored in k rows, each row having M symbols, where there are p bits per symbol, 100.

A parity check row is formed as in the basic SM code, by adding symbols modulo 2^P along a line of slope S_i, 300. The result is a symbol in the parity check row. This procedure is repeated using line S_i until every symbol stored in 200 is intersected by the line of slope S_i.

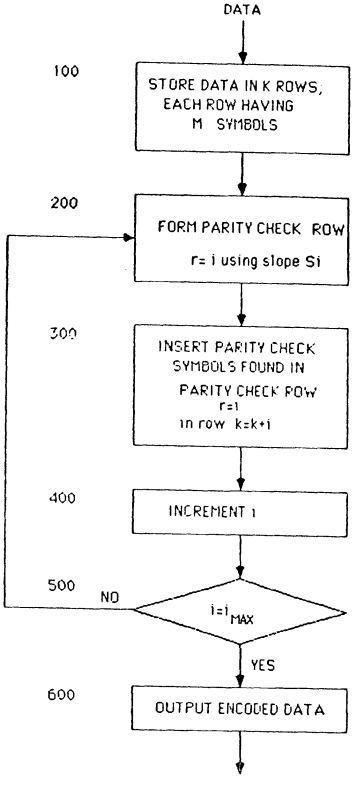


FIGURE 2.3-1 PASM ENCODER FLOW CHART

The parity check row then formed is stored in row k=k+1, 400.

Slope S_i is now incremented to slope S_{i+1}, 500 and the process is repeated until $i=i_{max}$. Thus, the number of parity check rows formed is $r=i_{max}$. The composite data block consisting of data symbols and parity check symbols are then transmitted at will.

Although any set of slopes S_i will result in satisfactory performance, the node distance can be maximized by using slopes S_i which do not yield multiple intersections in the sense described below and illustrated in Fig. 2.3-2.

The selection of slopes S_i is optimized by maximizing the minimum Hamming distance of the code. The slopes have been selected correctly if the minimum Hamming distance of the code is $d_{min} = 2^{imax}$, where i_{max} is the number of parity check rows.

In Fig. 2.3-2 we see that slope S_2 less then S_1. Also slope S_3 should be chosen so that it is less than the slope of a line passing through cells A and B. Similarily for slope S_4 we guarantee our performance if slope S_4 is less than the slope of a line passing through cells C and D, etc.

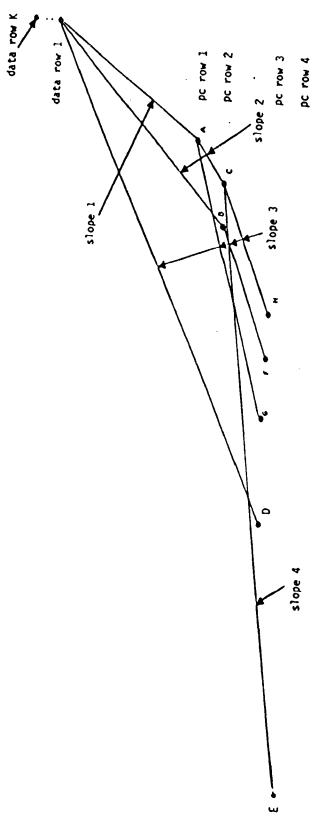


FIGURE 2.3-2 A CONSTRUCTION OF SLOPES FOR PASM CODE

It is readily seen that if only 3 parity rows are used and the slopes S_1 , S_2 , S_3 are chosen correctly, $d_{min} = 2^3 = 8$, which means that 8 errors must occur before the errors are nondetectable. One correct choice is: $S_1 = 1$, $S_2 = 1/2$, $S_3 = 1/4$.

2.3.2 Decoding of the PASM Code.

Decoding the PASM is similar to the decoding of the basic SM code concept except that a variable threshold exists for the parity check rows, according to what was calculated in the SM code as a function of the number of intersection of that row.

To explain, we see that all data bits are checked by r rows of parity check bits so that the maximum intersections is r. Thus, the threshold T_D is set at the value [r/2]. However, the bits in parity check row 1 are checked by r parity rows and threfore the maximum number of intersections possible are r. Hence the threshold T_{PC1} is the same as T_D, $T_{PC1} = [r/2]$. The bits in parity check row 2 are intersected by r-1 parity check rows and therefore $T_{PC2} = [(r-1)/2\ ]$. Similarly $T_{PCi} = [(r-i)/2\ ]$. Note that errors in parity check row r-1 can have two intersections and $T_{PC(r-1)} = 1$. Thus, errors in this row can be corrected. Further, Errors

in parity check row r can only detected, $T_{PCr} = 0$.

A Random Error Correcting Examples in PASM

Fig. 2.3-3 shows an example of 3 errors. Fig. 2.3-4a shows the error graph . Since r=3 the data threshold and the threshold of parity row 1 is T=1. We see one error in cell (4,2) which is above threshold. Inverting the bit in this cell yields an error graph Fig. 2.3-4b where cell (3,1) shows two intersections. Inverting the bit in cell (3,1) yield an error graph Fig. 2.3-4c where cell (5,5) shows two intersections. Inverting the bit in cell (5,5) clears the message from errors.

This three error pattern could not be corrected in the basic SM code. Because its composite error graph shown in Fig. 2.3-5 contains values of 1 at most. This kind of composite error graph will not allow any correction, and this lives the message with an uncorrectable error.

A general comment about the PASM code is, that the correction is "weaker" as we approach parity check row r.

2.4 TOTAL AUTOCONCATENATION SM CODE (TASM)

The SM code suffered from the fact that the parity check bits checked the data but did not check other parity check

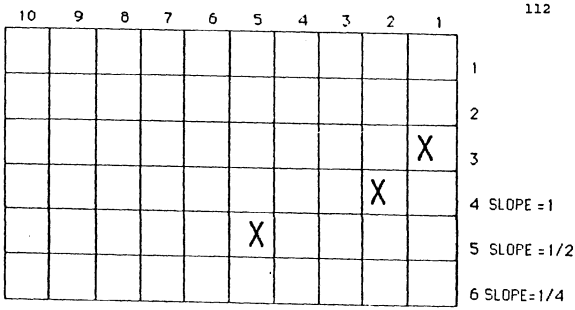


FIGURE 2.3-3 A MESSAGE WITH 3 ERRORS

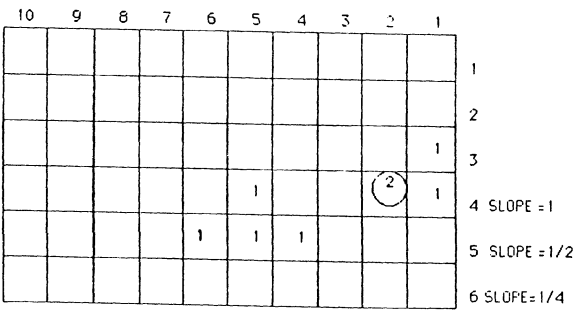


FIGURE 2.3-4a COMPOSITE ERROR GRAPH

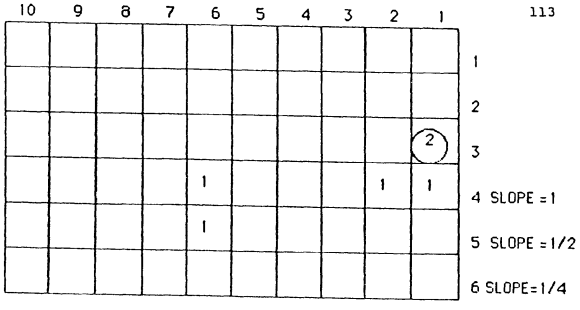


FIGURE 2 3-4b COMPOSITE ERROR GRAPH AFTER FIRST CORRECTION

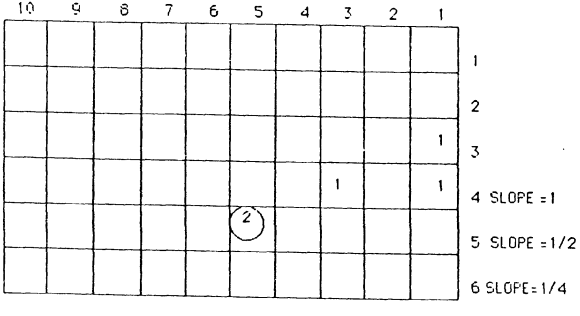


FIGURE 2 3-4c COMPOSITE ERROR GRAPH AFTER SECOND CORRECTION

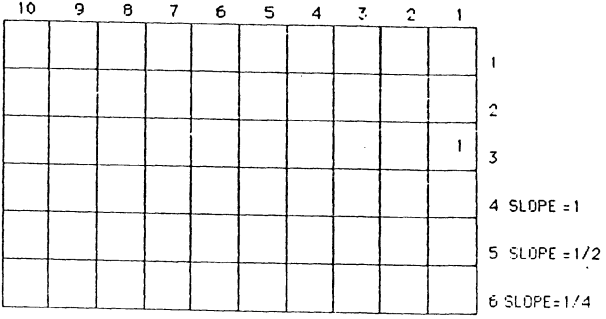


FIGURE 2 3-5 COMPOSITE ERROR GRAPH FOR THE SM CODE BASE ON FIGURE 2,3-3

bits. The PASM partially corrected this deficiency since parity check rows checked the data and the parity check rows preceding it.

The TASM completely solves the problem by letting each parity row check all data rows and all other parity check rows.

2.4.1 Encoding of the TASM Code.

The total autoconcatenation SM code (TASM) is encoded using the flow diagram shown in Fig 2.4-1.

The data is stored in K rows, each row having M symbols and each symbol having p bits, [100]. Refer to column j. Initially j=0, [200]. Starting with parity check row i=1 and using slope S_1 we determine parity check cell i=1 and j=0. We next determine parity check cell i=2, j=0 using slope S_2, etc., [300] and [400]. Any selection of slopes is appropriate so long as $S_{i+j} > S_i$. When $i=i_{max}$ we increment j from j=0 to j=1 and repeat, [500]. When $j=j_{max}$ we store and forward data and parity check cells for each parity check row and each column.

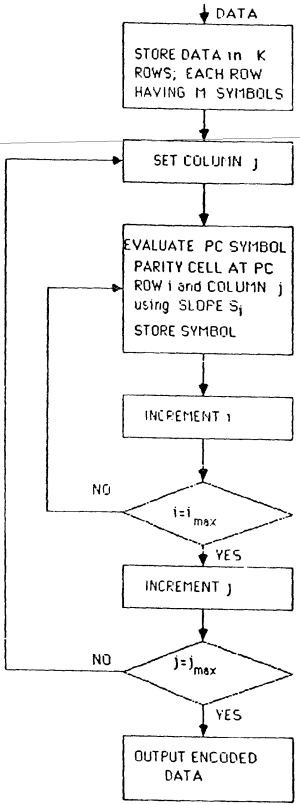


FIGURE 2.4-1 TASM ENCODER FLOW CHART

ALTERNATE ENCODING

Fig. 2.4-2 illustrates the TASM encoder for special case of a (7,4) code. The figure also shows a method of implementing the encoder. To begin, all of the registers are cleared so that a 0 is in each cell. Data is inserted into the four, data row registers d_1, d_2, d_3 and d_4. In our example, slopes of 1/4 for pc1, 1/2 for pc2 and 1 for pc3 are employed. The registers are initially cleared so until data or a pc bit reaches a given register, there is a 0 stored in it.

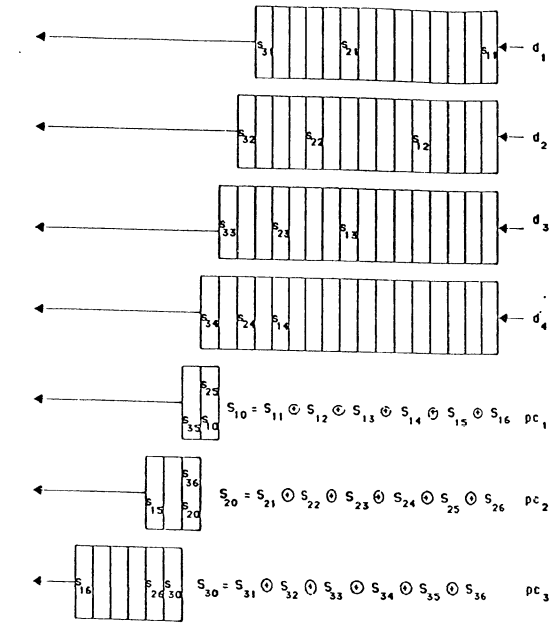
The first parity bit entered is

$$S_{10} = \sum_{\substack{i=1 \\ (\mathrm{mod}\,2)}}^{6} S_{1i} \qquad (2.4-1)$$

Next we calculate S_{20} where

$$S_{20} = \sum_{\substack{i=1 \\ (\mathrm{mod}\,2)}}^{6} S_{2i} \qquad (2.4-2)$$

and finally



STEPS: (1) CALUCLATE: $S_{10} = S_{25}$

(2) CALCULATE: $S_{20} = S_{36}$

(3) CALCULATE: S_{30}

FIGURE 2.4-2 EXAMPLE OF THE TASM ENCODER FOR A (7,4) CODE

$$S_{30} = \sum_{\substack{i=1 \\ (\mathrm{mod}-2)}}^{6} S_{3i} \qquad (2.4-3)$$

At each clock pulse four data bits, d1, d2, d3 and d4 are entered into the register and 7 bits are shifted out: S31, S32, S33, S34, S35, S15 and S16. Transmission ends when the last data bits clear the registers.
The decoding algorithm is the same as the PASM and the SM.

In general we can design using K data rows and R pc rows. The first pc row uses the slope S1=1/m1, the second S2=1/m2, etc., where s1<S2<...<SR. Typically SR=1 to insure compactness, and S1, S2, ...,SR are chosen using the construction described in the PASM.

PREFERRED ENCODING

Construction can be performed using shift registers or RAMS or any other memory device. It is convenient to consider the memory device to consist of K+R rows as shown if Fig. 2.4-3. Initially each memory cell (column, row) is cleared, i.e., a bit b(c,r)=0 is set to each cell. The data is entered into the K data bits in column 0. S_{10} is first calculated, then S_{20} is calculated and finally S_{R0} is calculated. The entire

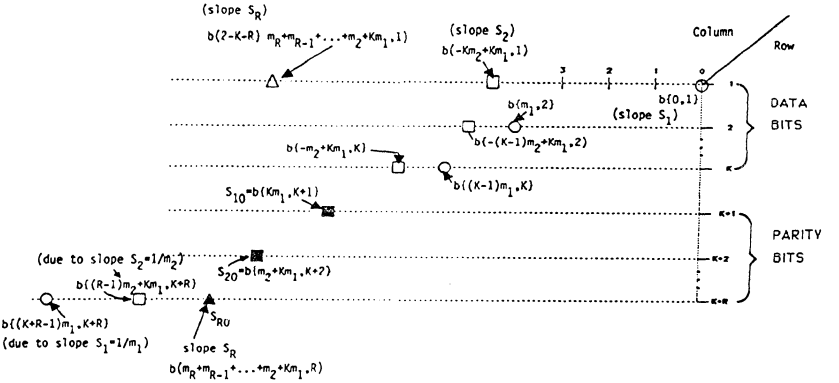


FIGURE 2.4-3 GENERAL STRUCTURE OF THE TASM SHOWING TWO SLOPES S_{10} AND S_{20} OF THE R SLOPES

contents of the register is now shifted to the left and a second column of k data bits is entered. S_{10}, S_{20}, ..., S_{R0} is again calculated and the process is repeated.

The equations used to calculate the parity check bits S_{10}, S_{20}, S_{30}, S_{40} and an arbitrary parity check bit S_{P0} where $1 \leq P \leq R$ is given below. Here S_{10} is calculated for the slope $S_1 = 1/m_1$ and S_{P0} for the slope $S_p = 1/m_p$.

$$S_{10} = \sum_{\substack{r=1 \\ r \neq k+1}}^{K+R} b\{(r-1)m_1, r\} \tag{2.4-5}$$

$$S_{20} = \sum_{\substack{r=1 \\ r \neq k+2}}^{K+R} b\{(r-K-1)m_2 + Km_1, r\} \tag{2.4-6}$$

$$S_{30} = \sum_{\substack{r=1 \\ r \neq k+3}}^{K+R} b\{(r-K-2)m_3 + m_2 + Km_1, r\} \tag{2.4-7}$$

$$S_{40} = \sum_{\substack{r=1 \\ r \neq k+4}}^{K+R} b\{(r-K-3)m_4 + m_3 + m_2 + kKm_1, r\} \tag{2.4-8}$$

$$S_{p0} = \sum_{\substack{r=1 \\ r \neq k-p, p=2,\ldots,R}}^{K+R} b\{(r-K-p+1)m_p + m_{p-1} + \ldots + m_2 + Km_1, r\} \qquad (2.4-9)$$

(all additions above are modulo-2 addition)

The minimum length of each register depends on the R slopes employed. For example for row 1, the register length is

$$L_1 = (2-K-R)m_r + m_{R-1} + \ldots + m_2 + Km_1 \qquad (2.4-10)$$

for row r the length is

$$L_r = (r-K-R+1)m_r + m_{R-1} + \ldots + m_2 + (K-r+1)m_1 \qquad (2.4-11)$$

where r = 1, 2, ..., K+R

2.4.2 Decoding of the TASM Code.

Decoding the TASM is similar to the decoding of the basic SM code concept. Except that a constant threshold exists for all the bits in the code. We first calculate the composite graph of the received word. If there are cells in the graph which have values above the threshold value their corresponding bits are invereted. The decoding is terminated

when there is no cells in the composite graph with a value above the threshold. Because of the symmetry of the system all bit are treated equally.

A Random Error Correcting Examples in TASM

Fig. 2.4-4 shows an example of 3 errors. Fig. 2.4-5a shows the error graph of that message. From the composite error graph we see 2 errors in cells (5,5) and (6,13) which are above threshold. Inverting the bits in these cells yields an error graph Fig. 2.4-5c where cell (3,1) shows 3 intersections. Inverting bit in cell (3,1) yields to a corrected message.

This three error pattern could not be corrected in the basic SM code, and the PASM code.

Fig. 2.4-6 and Fig. 2.4-7 shows the composite error graphs of the basic SM code and the PASM code respectively. In those figures we can observe that the highest values of the cells in the graph is 1, which does not allow any correction procedure. This lives the messages in both codes with uncorrectable errors.

The TASM code is the most powerful code in the family of SM codes.

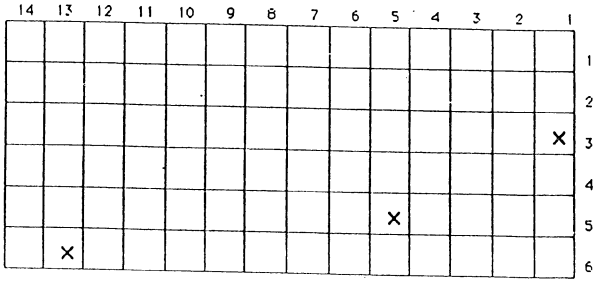


FIGURE 2 4-4 A MESSAGE WITH 3 ERRORS

SLOPE 1=1 SLOPE 2=1/2 SLOPE 3=1/4

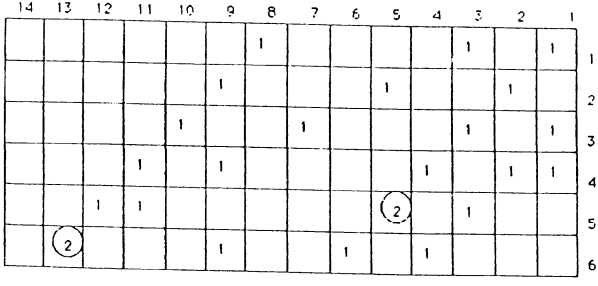


FIGURE 2 4-5a COMPOSITE ERROR GRAPH

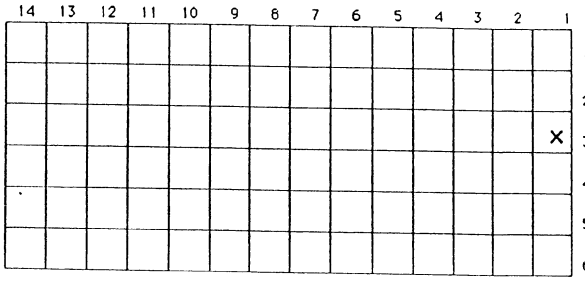


FIGURE 2.4-5-b MESSAGE AFTER FIRST CORRECTION

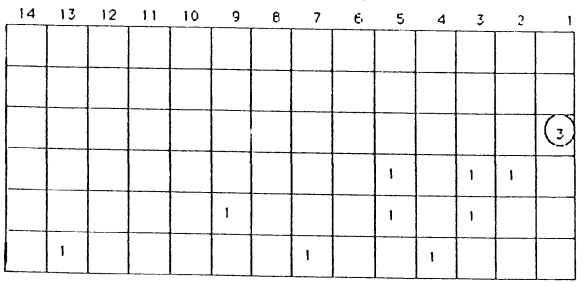


FIGURE 2.4-5c COMPOSITE ERROR GRAPHAFTER FIRST CORRECTION

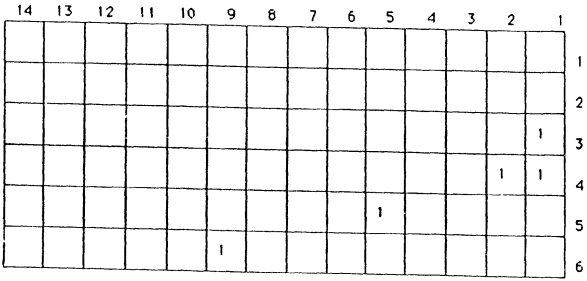


FIGURE 2.4-6 COMPOSITE ERROR GRAPH CODE PASM
BASE ON FIGURE 2.4-4

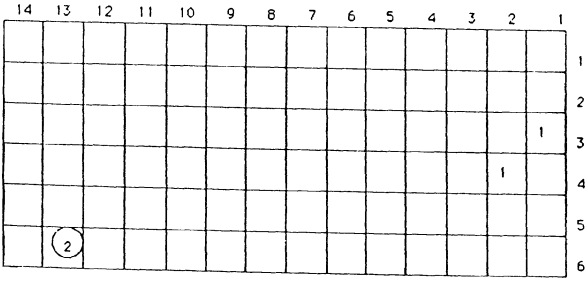


FIGURE 2.4-7 COMPOSITE ERROR GRAPH CODE SM
BASE ON FIGURE 2.4-4

3. ANALYSIS AND RESULTS

3.1 ANALYSIS OF THE SM CODES

3.1.1 Minimum Hamming Distance of the SM Codes

The minimum Hamming distance is a parameter of block codes as defined in Section 1.4. It determine the random-error-detecting and the random-error-correcting capabilities of a code. The minimum distance of a linear block code is equal to the minimum weight of its nonzero code word. When a code is transmitted over a noisy channel, and errors are induced, the performance of the code with respect to the minimum distance are:

(i) a code with minimum Hamming distance d_{min} will guarantee detection of any constellation of (d_{min} -1) errors.

(ii) If the decoding algorithm is optimal, than the code is capable of correcting all t errors or less, where

$$2t+1 < d_{min} < 2t+2 \qquad (3.1-1)$$

The above discussion emphasizes the importance of the

minimum Hamming distance. In Section 1.7 we discussed coding bounds, those bounds are calculated with the minimum Hamming distance as a variable. So by being able to determine the minimum distance of a code, it will give us a better understanding and a way to compare the codes capabilities.

3.1.1-1 Minimum Hamming distance of the basic SM code.

In Fig. 3.1-4 a general construction of an SM code is shown. If we assume that all the data is zero, we now induce errors into the code in such a way that at the end we will get an error constellation that will allow us to accept the error message as a legitimate code word. The number of errors induced which transfers one codeword into another codeword is the minimum Hamming distance.

Figure 3.1-4 shows such a configuration which puts one error in an information bit and one error along every one of the slopes in the parity check area. In this case the error in the information bit is not detected. In other words one legitimate codeword has been converted into another codeword. To do this conversion we need r+1 errors. Thus, the minimum Hamming distance is:

$$d_{min} = r + 1 \qquad (3.1-2)$$

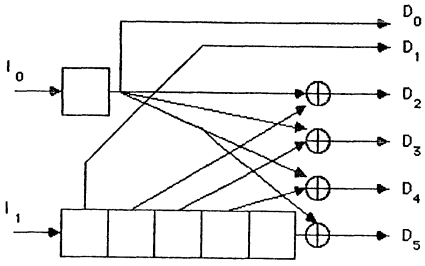
Another way of solving the problem of determining the minimum Hamming distance is, to look at the trellis of the convolutional version of the code. Observing the trellis we are looking for the number of ones in the path that merge with the all zero path, after diverting from it. An example of a (6,2) code is shown in Fig. 3.1-1. from the example and the codes structure we can show that:

$$d_{free} = r + 1 \qquad (3.1-3)$$

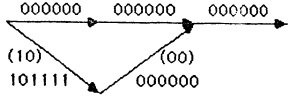
which is equal to the minimum Hamming distance found earlier. When dealing with the convolutional configuration of the code, it is interesting to try to calculate another important parameter which is the constraint length of the SM codes. It can be shown that the constraint length of the basic SM code is given by the by:

$$K = (9/2)2 + (r-1)(g-1) \qquad (3.1-4)$$

DISTANCE PROPERTIES FOR SM (6,2) CODE



ON THE TRELLIS



minimum free distance of the code

$d_{free} = 5$ $\quad$ $n_{free} = 12$

FIGURE 3.1-1 d_{free} CONVOLUTIONAL SM CODE

if the slopes are chosen in the following order

$\infty, 1, 1/2, 1/3, \ldots, 1/(r-1)$.

where

g- number of information lines

r- number of parity check lines

3.1.1-2 Minimum Hamming Distance of the PASM Code

The PASM code was designed to overcome some of the problems identified in the basic SM code. The main advantage of the PASM code is the increase of the minimum Hamming distance. By choosing the appropriate slopes as shown in Fig. 2.3-2 we discover that in order to generate a legitimate codeword by inducing errors into the all zero codeword, we need that one error will be in the information area and that every parity check line will have $2^{(i-1)}$ errors that checks the errors in all the upper lines including the data line that is in error. Calculating the minimum Hamming distance using the above observation we get:

$$d_{min} = \sum_{i=0}^{r} 2^{(i-1)} = 2^r \qquad (3.1-5)$$

The d_{min} we got in the above calculation is a huge improvement over the basic SM code, and it will be seen in the error-correcting capability of the code.

3.1.1-3 Minimum Hamming Distance of the TASM Code

The TASM code was designed to improve the performance of the basic SM and the PASM codes. The same observation that was made to calculate d_{min} of the PASM code can be done for the TASM code. By the same arguments $d_{min}=2^r$ for the TASM code.

Another interesting observation that allows us to calculate the d_{min} is shown in Fig. 3.1-2 a,b. In these figures we show a two and a three parity check row codes. We draw a set of minimum errors that will be considered by the code as a legitimate codeword. Figure 3.1-2a shows a two dimensional parallelogram. Figure 3.1-2b shows a projection of a three dimensional cube. Continuing this method we can see that for r parity check lines an r-dimensional hyper cube is needed which has 2^r points.

It is worthwhile to mention that what we have here is a projection of an r-dimensional space code into the two

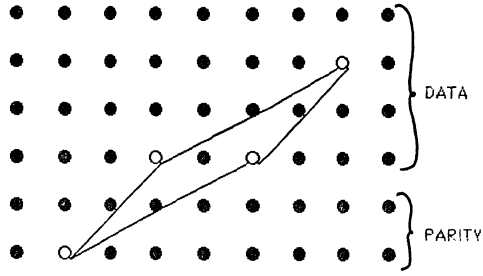


FIGURE 3.1-2 a TASM- MINIMUM HAMMING DISTANCE
2 PARITY CHECK LINES

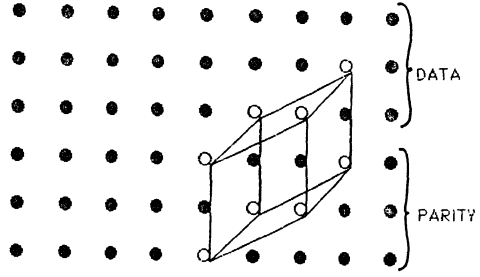


FIGURE 3.1-2 b TASM- MINIMUM HAMMING DISTANCE
3 PARITY CHECK LINES

dimension space. Another conclusion that comes up is that the TASM code has the same performance as an r-dimensional parity check product code.

3.1.2 Random Error Probability Calculation

To calculate the error probability of the SM codes, we initially used the results of the previous section, in order to evaluate what kind of errors can we expect to be corrected and what kind of errors cannot be corrected.

The second step was to identify the error configurations that cannot be corrected according to the initial step, and the decoding algorithm. For those errors we tried to calculate their probability of occurrence.

The third step was to try to identify error configurations, that are not predicted by the minimum Hamming distance. If those errors occur (as in the PASM code) repeat the the second step.

Because of our threshold decoding algorithm, there are special error configurations that generate errors into the data area. These events, when they do occur, increase the output error rate of the code.

The event of error generation is a drawback to the codes performances, and it enters into the probability of error calculation a whole new dimension of complexity. The two different events that leave the message with uncorrectable errors, also effects the way we choose our threshold. By searching for an optimal threshold we found that the two events have opposite direction in determining the value of the threshold. In order to increase information error correction we would like to lower the threshold as much as possible. On the other hand to reduce the error generation process we have to increase the threshold value to the maximum. These two contradictory routes led to an optimum threshold choice that will minimize the output probability of error.

Another approach to the calculation was to find upper and lower bounds for the probability of error. The results we obtained were loose bounds and are not of significant use. Indeed, a theoretical study to determine these bounds represents future work.

This problem led us to calculate the codes performances, by using computer calculations and computer simulation.

Only for the basic SM code were we able to come out with

a close analytical formula for the bound. This formula allows us to calculate the codes probability of error. The formula is described in the next section.

3.1.2-1 Basic SM Random Error Probability Calculation

To calculate the random error capability of the SM code, consider the data block and the parity-check bits shown in Fig. 3.1-3. The block has h data columns, g data lines and r parity check lines. To calculate the exact bit error probability of the decoded data, we should consider all possible error patterns with their probability of occurrence. To perform such calculation is of no theoretical and practical uses. Instead a lower bound on the probability of error is calculated, this calculation has general results applied to all the SM code configurations.

The lower bound of the bit error probability is based on the optimization of two error events. We will consider in the calculation the error events with the highest probability to occur. The two events are:

1. An error event that have errors in the data block that can not be corrected (uncorrectable random error patterns).

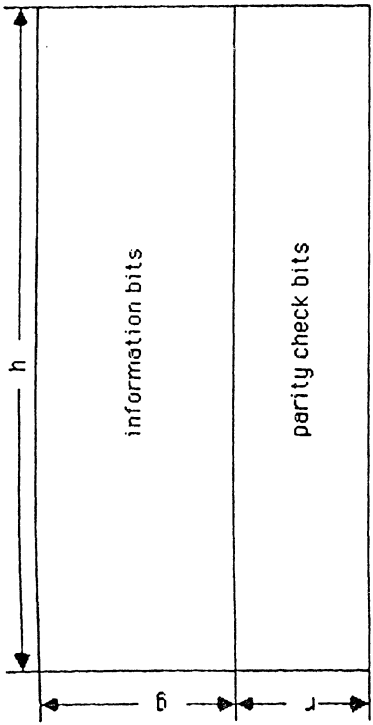


FIGURE 3.1-3 CODEWORD DIMENSION

2. An error event that have errors only in the parity check block and by the decoding method induced errors into the data block (random error generation).

Uncorrectable Random Error Patterns

An error pattern that is uncorrectable is shown in Fig. 3.1-4. Let the decoding threshold be equal to T.

Then if there are r parity check lines and only T-1 lines intersect, we will not correct the data bit that is in error. This requires r-(T-1) parity bits to be in error. The number of ways to choose r-(T-1) errors that will cause the error is:

$$\binom{r}{r-(T-1)} g \cdot h \qquad (3.1-6)$$

first error can occur in h slots.
second error can occur in only g slots.
the rest of the errors must occur only in specific slots.

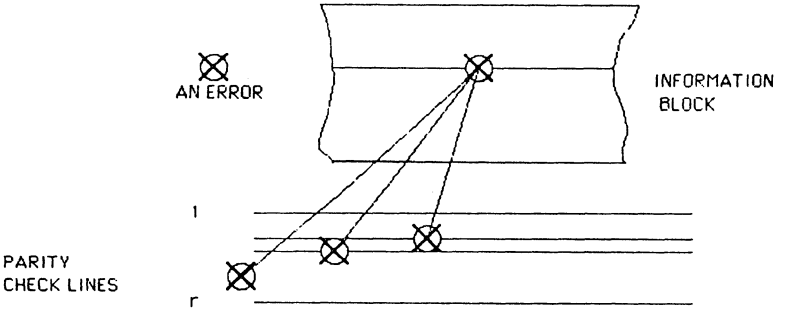


FIGURE 3.1-4 ERROR PATTERN OF AN UNCORRECTED BIT

Thus, the event probability is :

$$P_{e1} = g \cdot h \binom{r}{r-(T-1)} p^{r-T+1}(1-p)^{n-(r-T+1)} p(1-p)$$

$$= g \cdot h \binom{r}{r-(T-1)} p^{r-T+2}(1-p)^{n-(r-T+2)} \qquad (3.1-7)$$

where n-is the total number of bits in the block word.
The above event causes one data bit in error, out of $g \cdot h$ data bits in the block. Thus the bit error probability is :

$$P_{b1} = \binom{r}{r-(T-1)} p^{r-T+2}(1-p)^{n-(r-T+2)} \qquad (3.1-8)$$

Random Error Generation

To generate an error with a threshold T, T parity lines must be in error, as shown in Fig. 3.1-5.
The number of ways of choosing T errors in r lines that will intersect in the data block is:

FIGURE 3.1-5 ERROR PATTERN GENERATING AN ERROR IN THE INFORMATION

$$\binom{r}{T} g \cdot h \qquad (3.1-9)$$

Thus, the event probability is :

$$P_{e2} = g \cdot h \binom{r}{T} p^T (1-p)^{n-T} \qquad (3.1-10)$$

The above event causes one data bit in error, out of $g \cdot h$ data bits in the block. Thus the bit error probability is :

$$P_{b2} = \binom{r}{T} p^T (1-p)^{n-T} \qquad (3.1-11)$$

Threshold Optimization

The above events place opposite demands on T. In order to minimize the bit error probability, the above two events must be equal. This will allow us to choose T so, that the error correcting performance of the code are optimized.

$$\binom{r}{T} p^{T} \approx \binom{r}{r-T+1} p^{r-T+2} \qquad (3.1-12)$$

The solution of the above equation gives us a range for choosing T:

$$\frac{r+1}{2} < T < \frac{r}{2} + 1 \qquad (3.1-13)$$

Thus,

$$T = \left\{ \begin{array}{l} \frac{r}{2} + 1 \quad \text{for r even} \\ \frac{r+1}{2} \quad \text{for r odd} \end{array} \right\} \qquad (3.1-14)$$

Bit Error Probability

To calculate the bit error probability we have to consider the above two events. These events are mutually exclusive, which allow us to sum the two probabilities and get a lower bound on the bit error probability.
With the optimization on T we get,
1. for the probability of not correcting an error:

$$P_{b1} \approx \left\{ \begin{array}{ll} \binom{r}{r/2} p^{r/2+1} & r \text{ even} \\ \binom{r}{\frac{r+1}{2}} p^{\frac{r+3}{2}} & r \text{ odd} \end{array} \right\} \qquad (3.1-15)$$

2. for the probability of generating an error:

$$P_{b2} \approx \left\{ \begin{array}{ll} \binom{r}{r/2+1} p^{r/2+1} & r \text{ even} \\ \binom{r}{\frac{r+1}{2}} p^{\frac{r+1}{2}} & r \text{ odd} \end{array} \right\} \qquad (3.1-16)$$

and the lower bound is given by:

$$P_b > \left\{ \begin{array}{ll} \binom{r+1}{r/2+1} p^{r/2+1} & r \text{ even} \\ \binom{r}{\frac{r+1}{2}} p^{\frac{r+1}{2}} & r \text{ odd} \end{array} \right\} \qquad (3.1-17)$$

A correction factor F in the calculation has to be added due the the following argument: in both error events we have calculated the number of ways of choosing the error bit in

the parity check lines as:

$$\left(\binom{r}{r-(T-1)} \text{ or } \binom{r}{T}\right) g \cdot h \qquad (3.1-18)$$

The factor h in that multiplication corresponds to the number of ways choosing the first slot, but because the party check bits are calculated using diagonal modulo 2 summation, there are more then h parity check bits in every parity check lines. Thus the multiplication $g \cdot h$ in the equation should be the average length of a parity check line. If the slopes of the diagonals are chosen to be (1/i) where i= 1, 2, ..., r. Then the average length L of a parity check line is:

$$L = g + \frac{(g-1)(r+1)}{2} \qquad (3.1-19)$$

and the correction factor F in the lower bound equation is:

$$F = 1 + \frac{(g-1)(r+1)}{2g} \qquad (3.1-20)$$

Thus, the lower bound on the bit error probability is

$$P_b > \left(1 + \frac{(g-1)(r+1)}{2g}\right) \left\{ \begin{array}{ll} \binom{r+1}{r/2+1} p^{r/2+1} & r \text{ even} \\ \binom{r}{\frac{r+1}{2}} p^{\frac{r+1}{2}} & r \text{ odd} \end{array} \right\} \quad (3.1-21)$$

We see that for the basic SM code the minimum Hamming distance and the codes correction capabilities are equal. From the minimum distance rule we predicted that

$$t = \left\lfloor \frac{r-1}{2} \right\rfloor \quad (3.1-22)$$

which is exactly what we get from the above calculation.

3.1.2-2 Computer simulation

One of the approaches employed to compare the analytical calculations, was to perform a simulation of the performance of the SM codes. The parameters in the simulation were as follow:

i) Type of code (basic SM, PASM, TASM)

ii) Size and rate of the code
   - Number of data lines
   - Number of parity check lines
   - Number of columns

iii) Seed for the random number generator

iiii) Input error rate.

The simulation process included the following steps:

i) Generate the codeword

ii) Use the uniform random number generator to induce errors in the codeword, based on the input error rate entered the the beginning

iii) Perform the decoding process on the "received" data

iiii) Count the number of errors left in the information bits

iiiii) Add the number of errors found in (iiii) to the total number of errors calculated in the previous codewords.

iiiiii) When the number of errors exceeded 10 stop the process. Calculate the average probability of error of the simulation.

The simulation was completed when 10 or more errors were left uncorrected in the data. (The Chernoff bound guarantees that with the above amount of errors detected, the accuracy of the simulation is above 99%). This number is sufficient enough for all purposes of data communications.

The results obtained were:

i) The actual rate of the code

ii) The output bit error rate

iii) The total number of runs.

The results obtained from the various simulations required a significant amount of running time, particulary at low error rates. Thus, to reduce the computer time, we performed an alternate approach: to perform a computer calculation for the above codes. This is described in the next section.

3.1.2-3 Computer calculations

In order to overcome the complexity of the calculation an analytical formula for the probability of error, we performed a computer calculation using a VAX-780 computer. The computer calculation involved the following steps:

i) Generate the codeword

ii) Induce all ζ possible errors into the codeword.

iii) Performe a decoding process on the "received" data, i.e.,the data with errors.

iiii) Count the number of errors left after the decoding process is completed.

The following variables have been used in the calculation:

i) ζ - number of errors induced

ii) tr- the threshold value

iii) code size

g - number of information lines

r - number of parity check lines

h - number of columns in the code

iiii) encoding slopes

We performed the calculation for $1 \leq g \leq 4$, $1 \leq r \leq 4$ and $1 \leq h \leq 20$. The calculation have been performed for each of the SM codes. The results of the calculations is presented in the next sections.

As a result of the calculations we performed we found that the SM code could correct $t=(r-1)/2$ errors and the TASM code could correct $t=(2^r/2) -1$ as expected from the formula $t=[d/2]-1$. However, the PASM code could correct only $t=r$ errors which is less than what is predicted $(2^r/2) -1$.

In the computer calculation process we found that because of the different number of intersections that every parity check line has. We were unable achieve the above prediction for the codes correction capability. This result is a degradation in the codes performance. The explanation of this result is that the decoding algorithm for the PASM code is a suboptimal one, and that is the price for its simplicity implementation. It is left for future research to determine an optimum decoding procedure.

3.1.3 Burst Error Capability of the SM Codes

An example of the burst error capability of the SM code was given in Chapter 2. That example was dealing with correcting one line in error. In this section we will explain the burst error capability of the codes capable of correcting bursts having several lines in error. The burst error capability of the code is based on the idea that the transmitted bits are outputed line by line and that the parity check lines are interleaved with the data lines. In Fig. 3.1-6 we show a burst of errors with the length of two

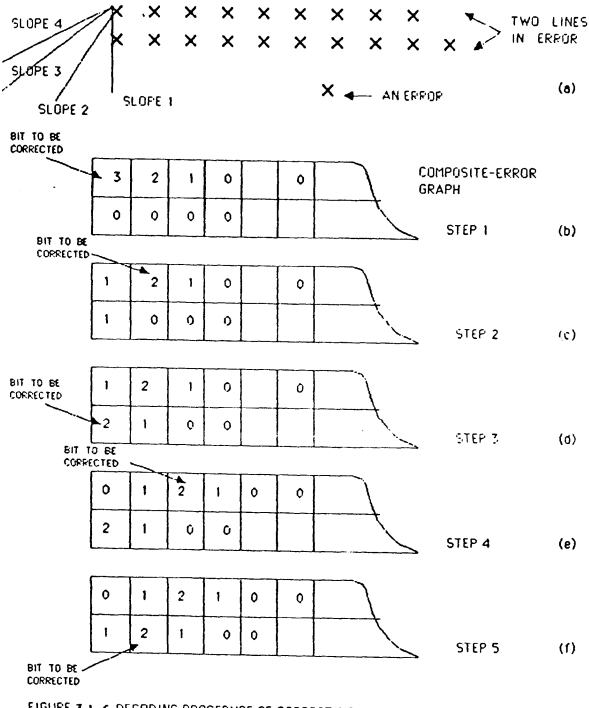


FIGURE 3.1-6 DECODING PROCEDURE OF CORRECTING A BURST OF TWO LINES

lines. If we assume that four parity check lines are used then we see first that the upper left bit that is in error is corrected. The second bit to be corrected is the one to the right of the first one. The next step is correcting one from the lower line and one from the upper line alternately. Every correction at this stage is allowing another bit to be corrected. The burst correction is a dependency correction. If we use the composite-error graph decoding we should set the threshold to be equal to 2. This value of the threshold guarantees that the burst correction performance are optimal. The same idea can be expanded into more parity check lines. Thus, for every even number of parity check lines (2k), the code is capable of correcting k lines that are in error.

For the basic SM code in order to correct more than one line in error we were required to set the threshold to the value 2. This value of the threshold does not fit the optimum value of (r/2)+1 found in the previous section. Thus for a basic SM code to operate in the burst mode, we will set the threshold to 2 and by doing it there will be a degradation of the random error correcting capability of the code.

Next we will calculate the efficiency of the SM code with respect to bursts of errors. For a code with 2k parity check lines, and slopes of the value 1/i where (i= 1, 2, ..., 2k). The total number of parity check bits is equal to:

$$w = 2kh + (g-1)(2k+1)k \qquad (3.1-23)$$

The efficiency of the code is:

$$z = \frac{2kh}{w}$$

$$= \frac{2kh}{2kh + (g-1)(2k+1)k} \qquad (3.1-24)$$

If the code is made longer, h increases and the efficiency of the code is approaches 1. In the limit

$$\lim_{h\to\infty} z = \lim_{h\to\infty} \left\{ \frac{2kh}{2kh + (g-1)(2k+1)k} \right\} = 1 \qquad (3.1-25)$$

From the above we see that for a given number of parity check lines the SM codes are asymptotically approaching the

Reiger bound and they have efficiency Z which approaches unity. Thus, the SM codes are asymptotically optimum burst error correcting code.

3.2 RESULTS

This section summarizes the results obtained from the calculations performed for the SM codes family.

The following table presents upper bounds on the probability of error obtained in the calculations based on m columns as m becomes arbitrarily large. The probability of a bit being in error p.

r=2

g	SM	PASM	TASM
1	$3p^2$	$5p^2$	$5p^2$
2	$8p^2$	$10p^2$	$10p^2$
3	$13p^2$	$14p^2$	$14p^2$
4	$15p^2$	$16p^2$	$16p^2$

r=3

g	SM	PASM	TASM
1	$2p^2$	$9p^3$	$220p^4$
2	$3p^2$	$25p^3$	$1000p^4$
3	$5p^2$	$50p^3$	$2000p^4$
4	$6p^2$	$100p^3$	$3000p^4$

r=4

g	SM	PASM
1	$10p^3$	$20p^4$
2	$15p^3$	$60p^4$
3	$20p^3$	$80p^4$
4	$29p^3$	$100p^4$

Figure 3.2-1 to 3.2-5 describe the codes relative performances error rate.

Figure 3.2-1 shows the output error probability versus the input error probability for one data line and several different number of parity check lines $1 \leq r \leq 4$.

Figure 3.2-2 shows the output error probability versus the input error probability for two data lines and a different number of parity check lines $1 \leq r \leq 4$.

Figure 3.2-3 shows the output error probability versus the input error probability for three data line and different number of parity check lines $1 \leq r \leq 4$.

Figure 3.2-4 shows the output error probability versus the input error probability for four data line and different number of parity check lines $1 \leq r \leq 4$. The results are compared to the (7,4) Hamming code. Such a code has a rate

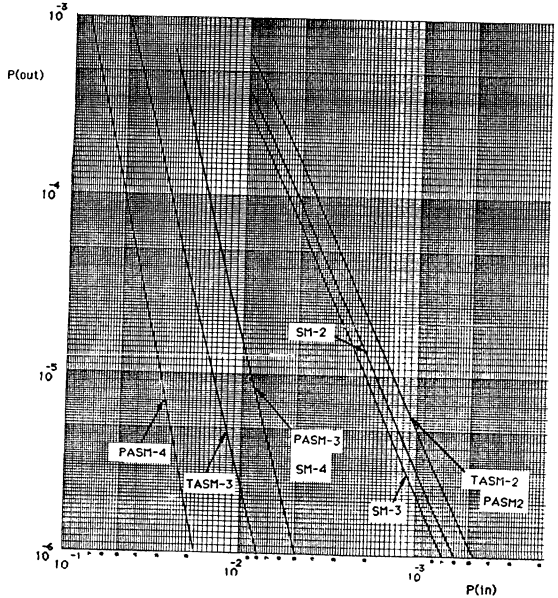


FIGURE 3.2-1 P(out) versus P(in) 1 DATA LINE

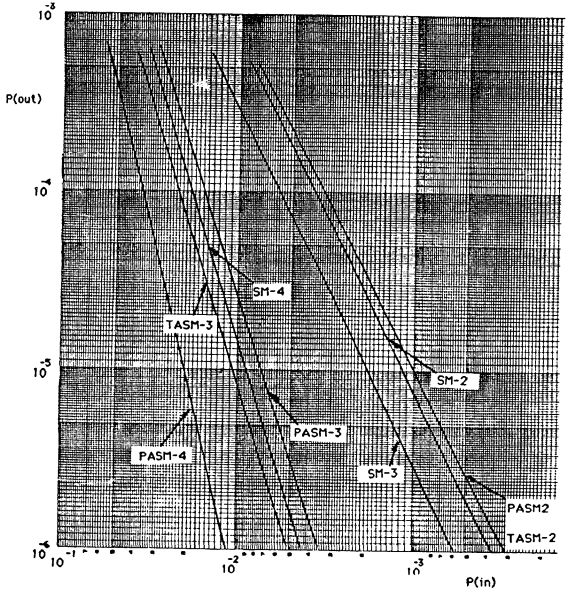


FIGURE 3.2-2 P(out) versus P(in) 2 DATA LINES

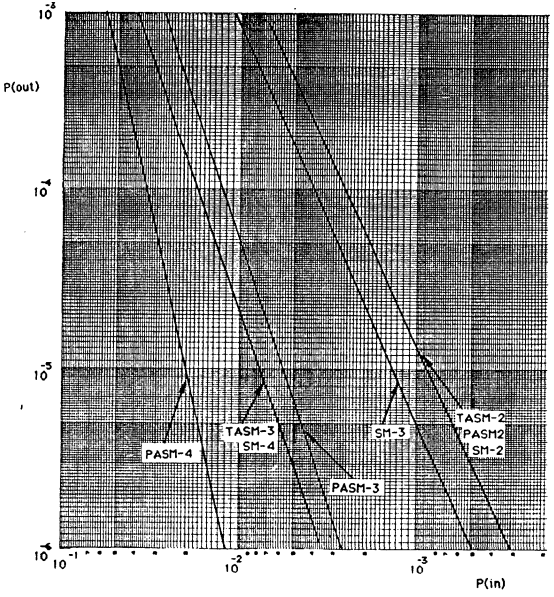


FIGURE 3.2-3 P(out) versus P(in) 3 DATA LINES

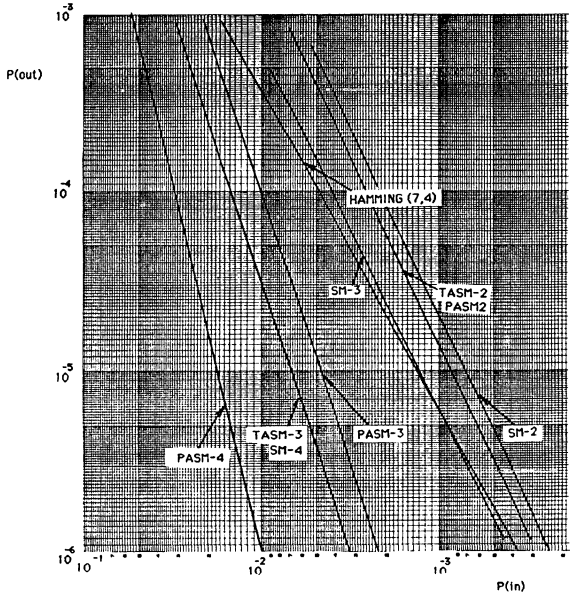


FIGURE 3.2-4 P(out) versus P(in) 4 DATA LINES

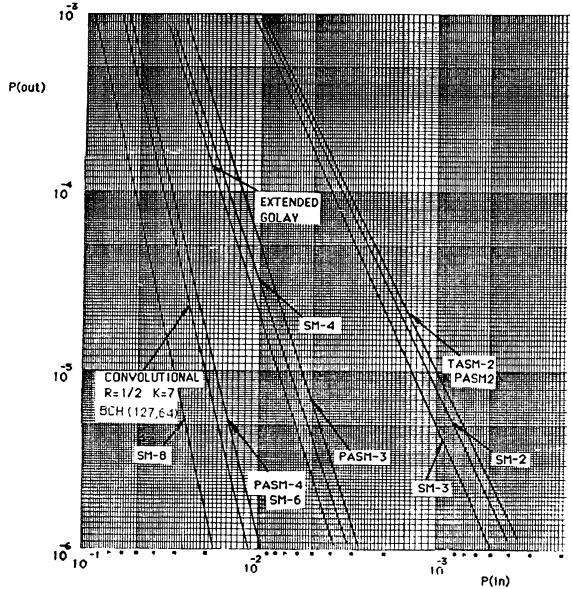


FIGURE 3.2-5 P(out) versus P(in) R=1/2

similar to the SM-3, PASM-3 or TASM-3. Note that the performance of the SM-3 is similar to the Hamming code, but the PASM-3 code results in an error reduction of more than 10. However, it was pointed out by Dr. R.L. Pickholtz that it is not "fair" to compare a relatively short code such as the Hamming or Golay (see Fig. 3.2-5) with the SM class of codes which are convolutional codes and therefore are very long codes.

From the above figures we can see that as r increases the code is capable of handling higher a input probability of error. Another observation is that the TASM code outperforms the other two codes, and that the PASM is the second in the SM family.

For r=2 all three codes perform almost equally, the basic SM code have a slight advantage over the other two codes. The reason for it is that the minimum Hamming distance in all three codes guarantees only single error correcting capability. The coefficient in the basic SM code is lower than in the others code because of less bits are been checked on every slope.

Another point is that for r an odd number for the basic SM code there is not much of an improvement over the next

lower integer. The reason for it is that for r odd the code will still correct the same number of errors as the previous r even. While using the PASM or the TASM we can see a noticeable improvement, and that is because of the change in the number of errors that can be corrected in those codes.

Another interesting observation is that the PASM-i (the number of parity check lines) has almost the same performance as an SM-(2i-2). The explanation is, that because this two codes have the same number of error correcting capabilities the difference lies in the coefficient which is almost equal because of the same structure of the code.

In Fig. 3.2-4 we draw the curve of the Hamming (7,4) code, for comparison reasons. We can see that the Hamming code has almost the same performance as the SM-3 with 4 data lines. (the two codes have also the same code rate).

Figure 3.2-5 presents the performances of different codes with the same code rate r=1/2. From the figure we can see that as r increases for all the SM family the codes performances are improved. In this graph we added two very useful codes, the "extended Golay" code, a convolutional

code with R=1/2, and constraint length K=7 and maximum likelihood decoding and a BCH(127,64) block code. From the figure we can see that the PASM-3 and the SM-4 have almost the same performances as the extended GOLAY code. The PASM-4 and the SM-6 have the same performances as the convolutional code and BCH code. For larger r the SM family codes will outperform those codes. As an example an SM-8 was added to the figure to show that even the weakest code in the family with r large enough will perform better than other famous codes.

It should be mentioned that the PASM-5 and the TASM-4 will each outperform the R=1/2 K=7 convolutional code and BCH codes. In addition as mentioned earlier construction of the PASM and TASM decoders is far less complex than the Viterbi decoder. Further the BCH coder is not used due to the complexity of the implementation.

4. ARQ APPLICATION OF THE SM'S CODES

4.1 STATE OF THE ART ON ARQ SYSTEMS

As described in Section 1.9.3 the hybrid type I ARQ system, consists of an FEC subsystem contained in an ARQ system. The function of the FEC portion is to reduce the frequency of retransmission by correcting the error patterns that occur most frequently. The purpose of that system is to provide higher reliability than an FEC system alone and a higher throughput than a system with ARQ only. The disadvantage of the type I ARQ systems is that it has constant overhead that must be included in each transmission and retransmission regardless of the channel error rate. When the channel is quiet, this represents a waste, and the throughput of type I hybrid ARQ systems is lower than the corresponding ARQ scheme.

Various hybrid ARQ schemes have been proposed to remove the disadvantage of the type I hydrid ARQ scheme by using some adaptive ideas.

4.1.1 Type II Hybrid ARQ Systems [8]

The type II hybrid ARQ system is based on the concept that the parity-check bits for error correction are sent to the receiver only when they are needed. Two linear codes are used in this type of scheme; one is a high-rate (n,k) code C_0, which is designed for error detection only, the other is an invertible half-rate code (2k,k) code C_1, which is designed for simultaneous error correction and error detection. A code is said to be invertible if, knowing only the parity-check bits of a code vector, the corresponding information bits can be uniquely determined by an inversion process (providing, of course, that there are no errors in the parity-check bits).

To explain the operation of this system, consider that a message u contains k information digits is ready for transmission. It is first encoded into a code vector v of n digits based on the error-detecting code C_0. The code vector v is then transmitted. At the same time the transmitter computes the k parity-check bits based on the message u and the half-rate invertible code C_1. The k parity bits of C_1

are not transmitter but stored in the retransmission buffer of the transmitter for later use. At the receiver the first n bits are received, the receiver computes the syndrome of the vector based on C_0. If the syndrome is zero, it is assumed that the data is error free and the message is accepted. If the syndrome is not zero, errors are detected in the message. The erroneous message is stored and a NAK is sent to the transmitter. Upon receiving the NAK, the transmitter encodes the k parity-check bits of C_1 into n bits based on C_0. Those n bits are now transmitted as vector v'. At the receiver the syndrome of that vector is calculated. If the syndrome is zero, the vector is assumed to be error-free and the message u is recovered by the inversion operation. If the syndrome is not zero, a decoding process take place on the 2k bits that were received based on the half-rate code C_1, i.e., k bits of data sent during the first transmission and k bits of parity-check bits sent during the retransmission. If the errors in the combined message can be corrected the corrected message will be accepted by the receiver. If the errors can't be corrected but will be detected, a NAK will be sent to the transmitter and the receiver will store the last k bits that were

received, discarding the previous k bits. Upon receiving the second NAK the transmitter will send the data u and the n-k error-detection parity-check bits based on C_0. At the receiver the process is repeated, and the result is an ACK or a NAK message sent back to the transmitter. The transmitter retransmits alternating repetitons of the parity code vector v' and the and the information code vector v. The receiver stores the received message u and the received parity block alternately. The retransmissions continues until u is finally recovered.

Figure 4.1-1 shows a qualitative comparison of the throughput versus input error probability, for selective-repeat ARQ systems and type I selective repeat hybrid ARQ system. Since in the hybrid I system a FEC code is used, more parity check bits are needed. This increases the overhead for each transmission and retransmission. As a result, when the channel error rate is low, it has a lower throughput than its corresponding ARQ scheme. However when the channel error rate increases, the throughput of the ARQ scheme drops rapidly and the hybrid-ARQ scheme is capable of maintaining a significant high throughput over a wide range of channel error rates if the design error-correcting

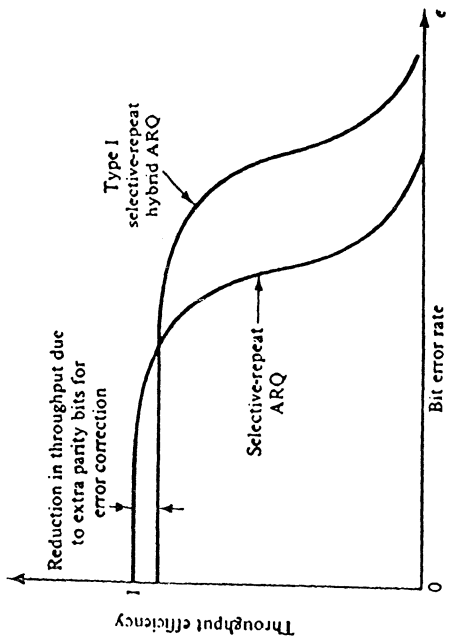


FIGURE 4.1-1 THROUGHPUT PERFORMANCE OF THE TYPE I SELECTIVE-REPEAT ARQ

capability of the code is sufficiently large.

The hybrid ARQ type II scheme has the advantages of the two systems with respect to the throughput efficiency. At low error rate the system operates in a selective-repeat ARQ scheme, and at high error rate it operates in a hybrid type I [8] selective-repeat ARQ scheme.

4.1.2 Code Combining [16]

Code Combining represents a technique for combining a number of repeated packets encoded with a code of rate R to obtain an effective lower rate code, and thus a more powerful error-correcting code. The Code Combining technique transmits the repeated vectors of the encoded word. In the receiver a maximum likelihood decoding is accomplished by generating a weighting value for each received symbol.

The Code Combining technique can be readily applied to ARQ systems. Data is first transmitted in blocks. If the received data is detected in error, a NAK is sent to the transmitter. The transmitter repeats the message that was NAK'd. At the receiver the messages are combined such that they can be decoded using a maximum likelihood decoding technique. If after the error correcting process the

message is still in error, another NAK is sent to the transmitter. At the transmitter the process is repeated, i.e., the encoded message is repeated again. At the receiver the retransmitted vector is combined to the older data and a new decoding process occurs. At the end of this process an ACK or NAK is sent back to the transmitter. The retransmissions continue until the message is finally recovered.

One way of soft-decision Code Combining is to represent the received voltage representing any bit b_i by V_i and the retransmitted b_i by V_i^*. Then combining is merely the average value $(V_i+V_i^*)/2$. This average value is used in the final convolutional decoder.

The Code Combining technique, is designed to work over noisy channels, and thus, an arbitrary number of noisy packets may be combined until the code rate is low enough to allow correct decoding of the message. A graph that describes code combining performance is given in Fig. 4.1-2, for a rate 1/2 convolutional code.

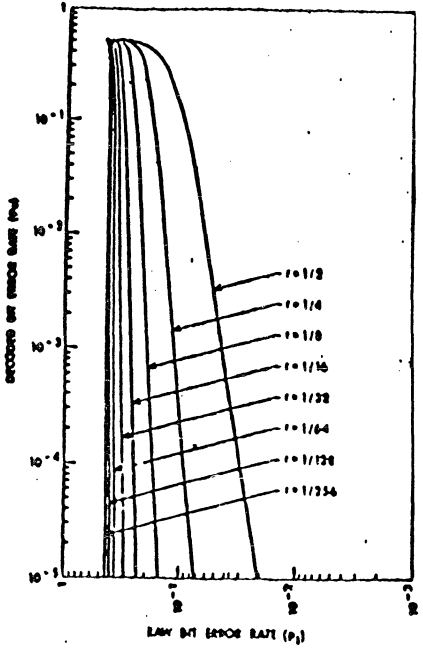


FIGURE 4.1-2 **Simulated Performance of K = 7 Convolutional Codes with Hard Decoding and Independent Bit Errors**

4.1.3 The SM ARQ System

Consider communication using the SM error correcting and detecting code when a feedback channel is available as shown in Fig. 4.1-3. If errors are detected and are too numerous to be corrected, the feedback channel is used to notify the encoder via a NAK. The encoder then sends one or more additional rows of parity check bits which significantly increases the power of the code.

Using this system, the average code rate remains high and only those messages which were initially received incorrectly, need have additional parity check bits sent.

This technique is more efficient than other ARQ systems since only a new set of parity check bits are retransmitted, while other techniques retransmit the entire data block. Also the power of the code increases as the sum of the parity lines received. Other codes do not have this capability. If we try to compare the throughput of this system with respect to the hybrid type II ARQ scheme (the most efficient system described in the literature), we can understand that at low error rate the two systems behave equally, at high error rates the SM scheme will outperform

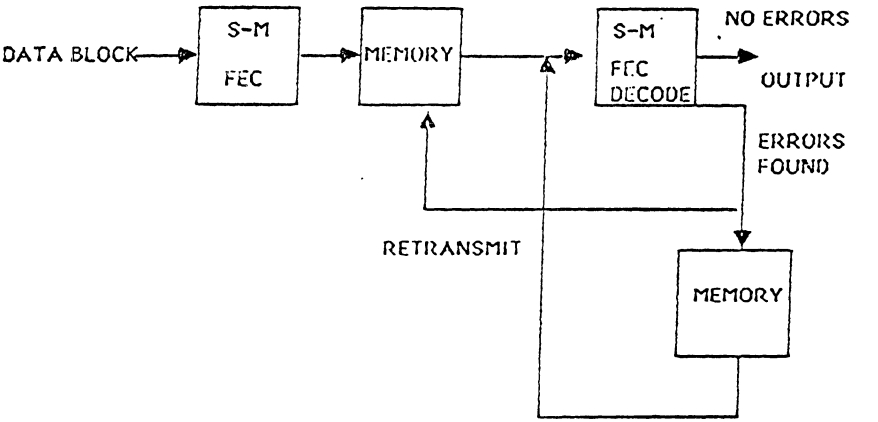


FIGURE 4.1-3 SM FEC/ARQ SYSTEM

the hybrid type II scheme, because of less redundancy of the retransmissions and improvements of the codes correcting capabilities.

EXAMPLE

To elaborate more on the significance of this system we consider the following example:

A block of data $D_1 = g \cdot h$ bits is encoded with r parity check lines. For example, let r=12. The block of data D_1 is transmitted with $r_{11} < r$ parity check lines. For example, let $r_{11}=2$. At the receiver, error correction occurs. If the errors are all corrected, the data is outputted. If some errors are not correctable, the received data block D_1 and the associated r_{11} parity check lines are stored. A retransmit request, i.e., a NAK, is returned to the sender.

Depending on the system delays when the data block D_i (i>1) is transmitted, it contains r_{1i} parity check lines and $r_{21}=4$ parity check lines. The r_{21} parity check lines belongs to block data D_1 and were stored in the transmitters memory in the decoding process. At the receiver the FEC code decodes data block D_2 and again decodes data block D_1 now using $r_{11}+r_{21}$ parity check lines. For example in the basic SM code for the $r_{11}=r_{21}=4$, then the probability of bit error

in D_1 after the first decoding is of the order of $p_{11} \propto p^3$, while since $r_{11}+r_{21}=8$, the probability of bit error in D_1 after second decoding is $p_{21} \propto p^5$.

This technique can be applied to the basic SM code and the PASM code, but not to the TASM code.

To calculate the throughput we will use the following notation:

p - bit error probability in channel
P_b - bit error probability after SM correction
P_{mi} - probability of the i^{th} message being corrected
n_i - number of bits in message i
T_b - time period of a bit
h - number of columns in a codeword

$$P_m = (1 - P_b)^n \qquad (4.1-1)$$

$$P_b \propto p^{r/2+1} \quad \text{for basic SM code} \qquad (4.1-2)$$

$$P_m \propto \left(1 - p^{r/2+1}\right)^n \approx \left(1 - p^{r/2}\right)^n \qquad (4.1-3)$$

The message size for each transmission is:

$n_1 = h \cdot (k + r_1)$

$n_2 = h \cdot r_2$

$n_3 = h \cdot r_3$

$n_l = h \cdot r_l$

the average time of transmission is:

$$\frac{T}{hT_b} = (k + r_1)P_{m1} + (k + r_1 + r_2)(1 - P_{m1})P_{m2} + \cdots + (k + r_1 + r_2 + \cdots + r_l)(1 - P_{m1})(1 - P_{m2})\cdots(1 - P_{m(l-1)})P_{ml} + \cdots \quad (4.1-4)$$

for $r_1 = r_2 = r_3 = \cdots = r_l = r$

$$\frac{T}{hT_b} = (k + r)P_{m1} + (k + 2r)(1 - P_{m1})P_{m2} + \cdots + (k + lr)(1 - P_{m1})(1 - P_{m2})\cdots(1 - P_{m(l-1)})P_{ml} + \cdots \quad (4.1-5)$$

Assuming that $P_m = 1$

$$\frac{T}{hT_b} \leq (k + r)P_{m1} + (k + 2r)(1 - P_{m1}) + \cdots + (k + lr)(1 - P_{m1})(1 - P_{m2})\cdots(1 - P_{m(l-1)}) + \cdots \quad (4.1-6)$$

using the following approximation that

$\mathrm{Max}(1-P_{mi})=x$

$$\frac{T}{hT_b}\leq(k+r)P_{m1}+(k+2r)x+\ldots+(k+lr)x^{l-1}+\ldots \qquad (4.1-7)$$

$$\frac{T}{hT_b}\leq(k+r)P_{m1}+\frac{1}{x}\left\{\frac{k}{1-x}+xr(1-x)^2\right\}-\frac{1}{x}\{k+(k+r)x\} \qquad (4.1-8)$$

after some algebra we get

$$\frac{T}{hT_b}\leq(k+r)P_{m1}+\frac{(1-P_{mi})^2}{P_{mi}^2}\{(k+r)P_{mi}+r\} \qquad (4.1-9)$$

The last part of this equation is the added throughput to the scheme and this term is neglible.

In the case of a very noisy channel, (channel error rate is above .02) one often employs Code Combining. To compare the performance of the basic SM code with the code combining technique using, a convolutional rate half code with constraint 7, with Viterbi Decoding algorithm (performance shown in Fig. 4.1-2) was employed. In order to perform a

comparison we found an SM code that matched the convolutional code performance (note an SM code is readily found which is superior to that of the rate 1/2 convolutional code). An SM code that satisfies those requirements is an SM-(6,6) (6 data lines and 6 parity check lines). For $P_i = 2\times10^{-2}$ the SM-(6,6) will deliver an output bit probability of 2×10^{-5} which is a factor of 2 worse than the convolutional code. Using the Code Combining method with two packet sent (R=1/4), to obtain an output of 1×10^{-5} the input error rate is 7×10^{-2} . For a rate 1/4 SM-(6,18) (6 data lines 18 parity lines) with $P_i = 7\times10^{-2}$ the output error probability will be 2.5×10^{-6} which outperforms the code combining scheme. Another way of comparison using the flexibility of the SM, ARQ scheme is to find an SM such code that for $P_i = 7\times10^{-2}$ the output error probability will be 1×10^{-5}. Such a code is an SM(6,16). The rate of this code is R=3/11 which is a factor of 1.1 more efficient than the code combining scheme.

The advantage of the SM-ARQ technique over the Code Combining scheme is in the flexibility of choosing the code rate, the significant lack of complexity of the decoding algorithm, and in the better handling of high error rates.

In the SM code the code rate can be changed gradually until the message is corrected, while the code combining scheme requires quantum steps in processing from stage to stage, which cause a very rapid decrease in code rate or equally decreases efficiency of the system.

5. CONCLUSION AND IDENTIFICATION OF FUTURE WORK

The research present herein was concerned with a new class of error-correcting codes. The codes in this category are capable of handling random and burst errors. There were three subfamilies considered in our research; the basic SM, the PASM and the TASM codes. The codes performances are improved from one family to the next.

The encoding and the decoding of these codes are very simple, which makes these codes very easy for implementation. The decoding can be made in two ways:

i) The block encoding method. In this method the data is arranged in a rectangular block and the parity check bits are arranged in lines. The generation is performed using modulo 2 addition of the data bits and the parity check bits along different slopes of the two dimensional space.

ii) The convolutional encoding technique. In this implementation we use shift registers and modulo 2 adders to calculate the parity check bits. An interesting point in this implementation is that we essentially perform a time domain filtering of the data. For the basic SM code the encoding is done by an FIR (finite impulse response) filter.

The encoding of the PASM and the TASM codes requires feedback operation and it can be viewed as an IIR (infinite impulse response) filtering. An important issue in this implementation is to chose the slopes or feedback points such that the encoding process will not oscillate .

The decoding technique used in this work is a threshold decoding method. The first step is to generate a composite error graph using modulo 2 addition along the encoding slopes. After the graph generation is completed all bits that corresponds to cells with values that exceeds the threshold are inverted.

The codes described in this work can be considered as a class of 'projection' codes. The projection is from an n-dimensional product code into a two dimensional space. The projection is performed by using different slopes for every one of the dimensions. This perspective of the above codes structure allows simple implementation of complex high dimensional product codes.

Another unique feature of the codes is its flexibility of construction. The parameters of the code that can be easily manipulated are the rate, length and error-correcting capabilities.

The random error correcting performance of the code exceeds some well known codes such as BCH codes and convolutional codes. Moreover, our codes are very efficient burst error correcting codes, and are asymptotically Reiger bound optimal codes.

We have shown our code to possess higher throughput when applied to an ARQ system compared with other techniques such as the hybrid type II ARQ system and code-combining systems.

Our work thus far enables us to identify challenging questions that need be answered. Among them are:

1) The utilization of higher alphabets in construction of our code.
2) An investigation of our code in higher dimensions.
3) Use of other codes in the construction of a conceptually related family of codes.
4) The application of soft decision in the decoding of our code.

In summary, our work has presented and analyzed an extremely efficient, simple and powerful family of codes. More importantly, it provides us with many exciting new areas of research and potential benefits.

6. REFERENCES

[1] C.E. Shannon, "A mathematical theory of communication," Bell Syst. Tech.J.,27 pp 379-423 (Part I), 623-656 (Part II), July 1948.

[2] J.I. Justessen, "A class of constructive asymptotically good algebraic codes," IEEE Trans. Inf. Theory IT-18, pp 652-656 September 1972.

[3] F.J. MacWilliams N.J.A. Sloane, "The theory of error-correcting codes," North-Holland Mathematical Library 1983.

[4] J.G. Proakis, "Digital communications," Mc Graw Hill Pub. 1983.

[5] M. Blaum, P.G. Farrell, H. van Tilborg "A class of burst error-correcting array codes, " IEEE Tran. Inf. Theory IT-32, pp 836-839 November 1986.

[6] G.D. Forney, "Concatenated Codes, " MIT Press, Cambridge, Mass., 1966.

[7] S.H. Reiger, "Codes for the correction of 'Clustered' errors, " IRE Trans. Inf. Theory, IT-6, pp 16-21, March 1960.

[8] S. Lin D.J. Costello, "Error control coding: Fundamental and applications, " Prentice-Hall Pub. 1983.

[9] W.W. Peterson E.J. Weldon, "Error-Correcting codes, " second edition, MIT, Press, Cambridge, Mass., 1972.

[10] E.R. Berlekamp, "Algebraic coding theory, " Mc Graw Hill Pub. 1968.

[11] G.C. Clark J.B. Cain, "Error-Correcting coding for Digital Communication, "

[12] R.E. Blahut, "Theory and practice of error control codes, " Addison Wesley Pub. 1983.

[13] A.J. Viterbi J.K. Omura "Principles of Digital Communication, " Mc Grow Hill Pub. 1979.

[14] A.J. Viterbi "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, " IEEE Trans. Inf. Theory, IT-13 pp 260-269, April 1967.

[15] H. Taub D.L.Schilling, "Principles of Communication Systems" second-edition Mc graw Hill Pub. 1986.

[16] D. Chase "Code Combining - a Maximum-Likelihood Decoding Approach for Combining an Arbitrary Number of Noisy Packets" IEEE Trans. on Communication Vol. COM-33 NO. 5, pp 385-393, MAY 1985.

[17] J.P. Odenwalder, "Error Control Coding Handbook" (Final Report) LINKABIT 1976.