VARIABLE REDUNDANCY CODING

FOR

ADAPTIVE ERROR CONTROL

by

R. M. F. Goodman

A thesis submitted for the Degree of Doctor of Philosophy in the Faculty of Natural Science at the University of Kent at Canterbury.

Electronics Laboratories June 1975

ABSTRACT

This thesis is concerned with variable redundancy (VR) error control coding. VR coding is proposed as one method of providing efficient adaptive error control for time-varying digital data transmission links. The VR technique involves using a set of short, easy to implement, block codes; rather than the one code of a fixed redundancy system which is usually inefficient and complex to decode. With a VR system, efficient, low-power codes are used when data-rate channel conditions are good, and very high-power inefficient codes are used when the channel is noisy. The decoder decides which code is required to cope with current conditions, and communicates this decision to the encoder by means of a feedback link. This thesis presents a theoretical and practical investigation of the VR technique, and aims to show that when compared with a fixed redundancy system one or more of the advantages of increased average data throughput, decreased maximum probability of erroneous decoding, and decreased complexity can be realised. This is confirmed by the practical results presented in the thesis, which were obtained from field trials of an experimental VR system operating over the HF radio channel, and from computer simulations. One consequence of the research has been the inception of a study of codes with disjoint code books and mutual Hamming distance (initially considered for combatting feedback errors), and this topic is introduced in the thesis.

ACKNOWLEDGEMENTS

I should like to thank A.S.W.E. and the S.R.C. for their support in terms of finance and equipment, and Professor R. C. Jennison for all the many facilities of the Electronics Laboratories that I have used. Thanks are also due to the technicians of the Laboratories for their practical help, and in particular to D. E. Pantony for all his help on the field trials. Special thanks go to my wife and parents, and to the many friends and colleagues who have made it possible for me to complete this work. Finally, and most importantly, I want to thank Dr. P. G. Farrell for being such a good friend and an excellent supervisor from the start of our research association.

TABLE OF CONTENTS

LIST OF PRINCIPAL SYMBOLS

a_m	probability of erroneously decoding a block containing m errors.
a_l	probability of erroneously decoding a block containing a burst of l errors.
b	burst correcting ability.
C	Channel capacity.
d	minimum Hamming distance
d'	distance of dual code
e	probability of an erasure
E	efficiency(also error exponent)
F	VR throughput improvement factor
f_c	critical frequency
k	number of message digits
m	a sample value of block error rate
n	block length
n_d	number of detectable error patterns
N	number of samples
N_c	proportion of correctable burst error patterns
p	bit error probability
p'	mean block error rate (mean of many p_{BLK} samples)
p_{BLK}	block error rate
p_{UNDET}	undetected block error rate
P	probability
P_c	probability of correct decoding
P_d	probability of detecting an erroneous block
P_e	probability of erroneously decoding a block
P_L	long term average bit error rate
P(m,n)	probability of m errors in n bits
R	code rate
S	sink bit error rate
t	error correcting capability
U	sink block error rate
x	proportion of time P_e exceeded
z	Reiger efficiency

CHAPTER 1

Introduction

This thesis is concerned with variable redundancy (VR) error control coding, for use on noisy data transmission links. VR coding is proposed as one method by which error detecting and/or correcting (EDC) codes may be used to provide automatically adaptive error control for a time-varying channel (Farrell 1969).

The results presented in this thesis aim to show that by using VR coding, reliable and efficient data transmission can be achieved with *simple* EDC codes; and that when compared to a fixed redundancy (FR) coding system operating on a time-varying channel, the advantages of increased overall throughput*, or decreased complexity, or both, can be realised.

The genesis of the study of coding theory is contained in Shannon's fundamental theorem (Shannon 1948). This theorem states that every noisy channel has a definite channel capacity (C), and that provided information is transmitted through the channel at a rate less than C, there always exists some coding scheme which will give as near error-free performance as is desired. More specifically, for any rate $R<C$, and any code length (n), there exists a code for which the probability of erroneous decoding (P_e) is given by

$$P_e \leq e^{-nE(R)}, \qquad 1.1$$

where E(R) is specified by the channel transition probabilities.

*here, and subsequently, average throughput is implied.

Shannon's theorem gives no indication of how to construct these good, long, codes; and it has been the ever-increasing need for extremely reliable data communication that has provided the incentive for coding theory to find the good coding schemes promised by Shannon.

In order to detect and/or correct transmission errors, redundant digits are added to the message data in a controlled, hopefully efficient, way. The rules for doing this, and for detecting and locating the errors in the received sequence, are given by the particular code. Equation 1.1 shows that for a code to be effective n must be large, and finding codes that are both efficient and long is theoretically difficult. More seriously, large n implies long decoding delays and complex decoding procedures, which may render impractical a theoretically good coding scheme.

The fundamental problem facing coding theory is therefore one of finding long, powerful and efficient codes that can be easily implemented.

The probability of erroneous decoding for a particular code, depends on the channel error statistics, and if these are fixed it may be possible to design an optimum code for the channel. For many practical digital data transmission links, however, the error statistics vary considerably with time. Consequently, for the system to be reliable, the coding scheme used must be powerful enough to cope with the worst channel error conditions. Since powerful codes are invariably inefficient or complex or both, throughput efficiency and P_e are unnecessarily low, and/or

the complexity is high, when the system is operating under average or low channel error conditions.

Variable redundancy coding offers one possible solution to this problem by using a set of codes of varying redundancy, and hence EDC power, instead of just one powerful code. The set of codes is chosen to span the range of channel error conditions expected, and the system automatically uses the most suitable code at any particular instant. In this way, the EDC power of the system is varied in sympathy with the channel noise conditions, in order to *match* the EDC power required by the channel at any particular instant. A VR system therefore has the effect of increasing throughput and maintaining P_e relatively constant under varying channel conditions. The penalty paid for this is a variable rate of data extraction from the source.

Compared to an FR system, the demands on the codes used by a VR system are not so stringent; only the codes for low or average error rates need be efficient (although all should be optimum and easy to implement) because the system will operate in this condition for most of the time. Short, known, codes with low complexity can therefore be used.

The receiving terminal of the system must be capable of extracting channel statistics from the incoming transmission and using them to form a decision, based on a suitable criterion, as to which code in the set is most appropriate to the current channel conditions. This decision must be reliably passed to the encoder via a feedback link so that the appropriate code can be selected. In a simplex (one-way)

transmission system the penalty of having to have this feedback link is offset because a retransmission method of error correction can be used, thereby decreasing decoder complexity even further. In addition, the feedback bandwidth (or time for duplex) can be much smaller than that of the forward channel.

During the course of the research an experimental VR system was built. This was operated over a forward HF channel, and used a G.P.O. line feedback link. Results from this system together with results obtained from computer simulations of other VR systems, are presented in this thesis; and aim to show that simple and efficient adaptive data transmission can be realised by using the VR technique. The problem of feedback errors, encountered on the experimental system, has led to the study of codes with disjoint code books and mutual Hamming distance (Goodman 1974). This study is introduced in this thesis together with theoretical work on the generation of sets of such codes.

The thesis develops in the following way. Firstly chapter 2 outlines methods of providing error control for different types of channel by using EDC codes, and introduces the elementary properties of codes and channels. Chapter 3 develops the mathematical theory of block codes, and outlines the limitations, structure, implementation, and general properties of codes, giving particular attention to the important class of cyclic codes. This chapter is essentially a review of coding theory required for a full

understanding of subsequent chapters, and as such may be omitted by the suitably informed reader. The concept of adaptive communication, together with the problems of implementing such systems are outlined in chapter 4. Also considered there are the characteristics of time-varying channels (with which the adaptive system must cope) particularly the HF channel. Chapter 5 presents an investigation of the VR approach to adaptive coding, and both theoretical and practical aspects of the performance and implementation of VR systems are considered there. Also introduced in chapter 5 is the topic of codes with disjoint code books, and this is further developed in appendix A. Chapter 6 describes the design and operation of the experimental system, and gives details of the system's field trials. Chapter 7 presents and discusses results on the behaviour of the HF link, the performance of the experimental VR system, and the performance of other computer-simulated VR systems. Finally, in chapter 8, the thesis is summarised and conclusions on the work are drawn, together with suggestions for further research.

CHAPTER 2

Error Control by means of Coding

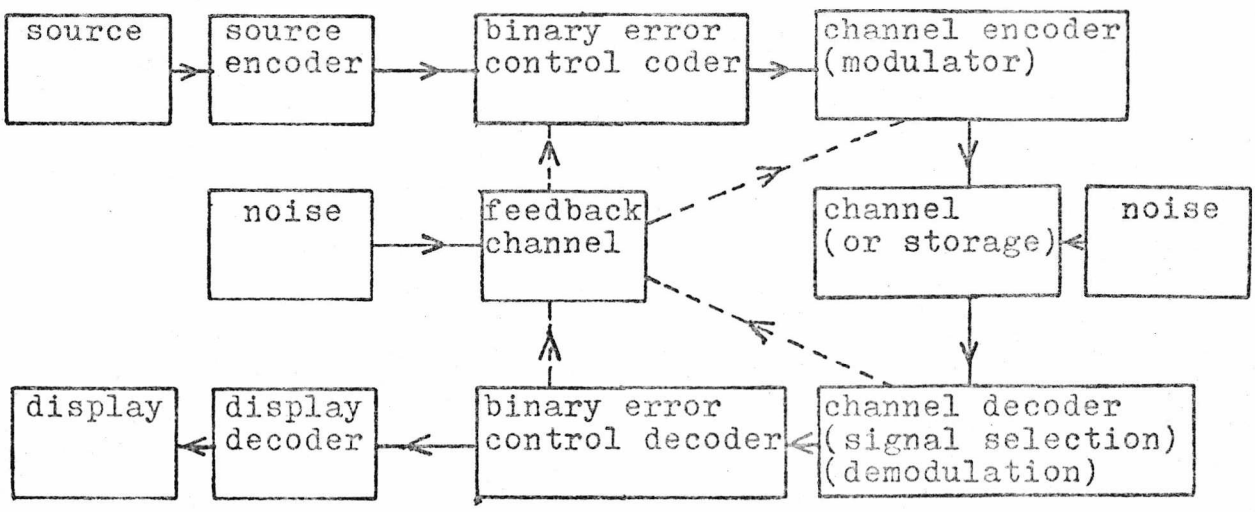
This chapter outlines the basic methods of providing error control for noisy data communication links by using EDC coding. The types of code available for error control (EC) are classified, and some of the elementary properties and limitations of codes are introduced. Attention in this and subsequent chapters is, however, restricted to binary codes used over binary channels, and in particular to binary linear block codes. The basic types of channel, their properties, and their effects on decoding are next considered; and finally the fundamental coding problem is outlined.

2.1 *The data communication system*

A basic data communication (or storage and retrieval) system is shown in figure 2.1.

Figure 2.1

The Basic System



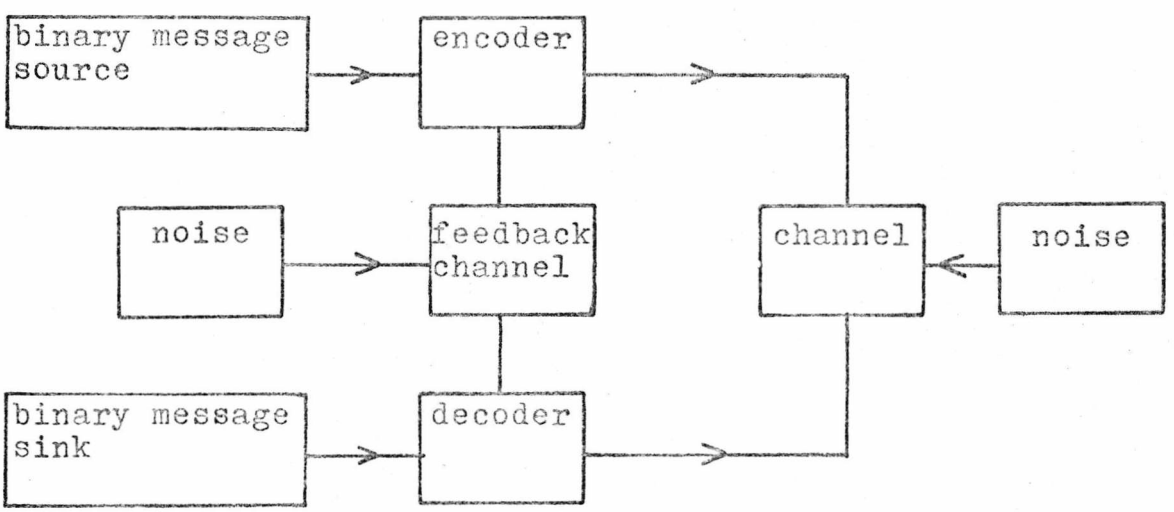
The first element of the system is an information source, which may be a person or a machine, that outputs source messages which may be continuous waveforms or discrete symbols. The source encoder takes the source information and performs a one-to-one translation from the source language to binary, thereby producing a serial sequence of binary digits which is the binary message presented to the error control coder. In order for the transmission to be efficient it is generally assumed that the message presented to the EC coder is not in itself redundant and hence in possession of some inherent EDC power. This is obviously not true for information sources which operate in languages such as English; and so it is assumed that as well as being a transducer, the source encoder may perform some symbol encoding (Huffman 1952) in order to produce binary data with minimum redundancy. The error control coder takes the data sequence and, according to the coding rules, transforms this on a one-to-one basis into another, longer, binary sequence that is the code sequence. The code sequence is used by the modulator to produce a unique signal suitable for transmission over the channel. Signals pass through the channel and are perturbed by noise to varying degrees. The receiver then selects the required signal and the demodulator makes a decision as to which code sequence corresponds to which received signal. The output of the channel decoder is a received binary sequence that may contain errors and hence may or may not exactly match the transmitted code sequence. The error control decoder, on the basis of the received sequence, the EDC capability

of the coding rules, the modulation used, and the channel characteristics, does two things. Firstly, an attempt is made to detect and/or correct errors in the received sequence, in order to produce an estimate of the transmitted code sequence. Secondly, if the code is non-linear or non-systematic (section 2.3), the code word is transformed to produce an estimate of the original binary data sequence. If the channel is very noisy the estimate and the original may differ considerably, thereby giving unreliable communication. Finally, the data sequence is delivered to the user and can be displayed in any convenient form. Also shown in the diagram are possible feedback paths whereby control information may be passed from the receiving side to the transmitting side in order to help the modulator and EC coder form more reliable estimates, and/or vary transmission to cope with a time varying channel.

If attention is to be restricted to error control only, the system of figure 2.1 can be reduced to that shown in figure 2.2.

Figure 2.2

The Restricted System



In this case the encoder - decoder pair are the error control subsystems with binary inputs and outputs. Binary message sequences are coded using an EDC code and the binary code sequence is passed through the channel where it may be corrupted by noise. The decoder then attempts to produce a reliable estimate of the original message sequence from the corrupted code sequence and may use a feedback link to facilitate this by means of coder control. The design of the encoder - decoder pair must ensure firstly that data throughput is as fast as possible, and secondly that the decoded message is as accurate a replica of the original message as possible.

The channel now includes the modulator and demodulator (modems), and can therefore be modelled by a binary process. As the code used is chosen to cope with the type of error patterns produced by the channel, it is advantageous to have full channel error statistics, and to be able to model the channel in some mathematical way. Any model, however, should ideally include not only the effect of errors due to the characteristics of the physical channel, but also the effect of the modulation used, which may give rise to particular types of errors.

2.2 *Methods of error control*

An EDC code can be used to provide error control in any one of three main modes: error detection (ED); error detection with correction by automatic repeat request (ED-ARQ); and forward error correction (FEC).

Error detection involves using the code redundancy to check whether or not a *received* sequence is a *code* sequence. If not, then errors are known to lie somewhere in the sequence; but cannot be located and corrected. The decoder in an ED system therefore delivers an erroneous sequence to the data sink together with a flag symbol to indicate that the sequence contains errors. If correction of the detected errors is required then a retransmission system with a feedback link is needed, so that the decoder can instruct the coder to repeat the erroneous sequence as many times as is needed for the sequence to be correctly received. This system is known as automatic repeat request (ARQ).

ARQ can be implemented in two main ways. Firstly, with recursive ARQ, the correct reception of each code sequence is acknowledged before another sequence is transmitted. This reduces the data throughput rate, due to the bandwidth of the return channel which may be much lower than the forward channel, delaying the acknowledgment signal. Secondly, with non-recursive ARQ, forward transmission takes place continuously and is only interrupted for retransmission. Interruption, due to the return delay, therefore takes place several sequences after the detected error; and so although transmission is faster for non-recursive ARQ, more complex buffering is required than in the recursive case. Both types of ARQ system accept data at a variable rate, dependent on channel conditions, and if it is required to extract data from the

source at a constant rate, then buffering is required. This leads to the possibility of buffer overflow, with consequent complete data loss, if channel conditions are bad enough to affect many adjacent sequences.

Error control can also be used to correct errors in a sequence by using an EDC code in the forward error correction (FEC) mode. Data extraction and throughput takes place at a constant rate and the coding rules are used to detect, locate, and hence correct errors.

Hybrid error control methods are possible by combining two or more of the above methods. For example, a code can be used to *correct* up to a certain number of errors in a sequence, and then to *detect* if a greater number occur. These detected but non-correctable sequences may then the corrected by ARQ, if a retransmission system is available.

The choice of an error control method for a particular application depends on a number of factors. Firstly, codes used for forward error correction are more redundant than error detecting codes and data throughput is consequently less. The channel conditions also affect the choice of method: for example, a channel with a mainly random error distribution which does not vary greatly with time may be an ideal case for FEC; whilst a channel with varying impulse or burst type noise may dictate an adaptive method incorporating ARQ. If operation at a constant source data extraction rate is required then ARQ and similar rate-adaptive methods are

ruled out unless sufficient buffer storage is available to cope with the worst channel error conditions. FEC and ARQ systems are more complex than detection systems; and this complexity may not be justified if the data has inherent redundancy which can be utilised by the user to correct errors intuitively, for example in the transmission of English text. If a return link is not available then decoder controlled adaptive systems are ruled out. Data that is reproducable or storable on a temporary basis would indicate an ED or ARQ system whilst non-reproducable data would demand an FEC system, for example, prime data stored on magnetic tape or disc for later replay.

The choice of an error control method therefore depends on the individual communication system requirements, and the engineering problem involves a compromise between the user's data reproduction reliability demands, the system cost in of terms of data throughput and equipment complexity, and the restrictions imposed by an already existing communication system that may have to be modified to incorporate error control.

2.3 *Codes for error control*

In order to code for error control, redundant digits are added to the source data in a controlled way, so as to form checks on the data. This results in a code with a fixed EDC power. The redundancy is used by the decoder to detect and/or correct errors in the received code sequence.

Although codes have been devised on an heuristic basis, it is very desirable for codes to have high mathematical structure. This structure enables classes of codes rather than individual codes to be devised and also simplifies implementation. In this thesis the emphasis is on codes with an algebraic structure, but codes with other structures such as finite geometries are briefly considered. Most of the codes devised so far are linear codes, and attention is here restricted to the more practical subclass, the binary linear codes.

In general a binary linear (n_0, k_0) code is one in which the coder accepts a string of k_0 information digits from the source, calculates a set of $n_0 > k_0$ modulo - 2 sums on various information digits, and delivers the n_0 digit code sequence to the channel. The ratio k_0/n_0 is called the code rate (R), and for efficient coding we require codes with as large a rate as possible for a given EDC power. This usually implies that n_0 must be long with high implementation complexity.

It is possible to divide the binary linear subclass into two fundamentally different types of codes: block codes, and tree codes. In a block code information digits are split into k digit sections, or blocks, and these are processed independently to produce an n digit code word. The code word passes through the channel and is decoded independently of any other blocks. A systematic block code (as implied in this thesis) is one in which the first k digits of the code word are the same as the

k information digits. As every binary linear block code is equivalent to some systematic code, it is therefore usual to use mainly systematic block codes in order to realise the advantage of reduced coder - decoder (codec) complexity. Coding with a systematic block code therefore involves taking n-k modulo-2 sums on the information digits and serially adding these n-k parity checks to the k digit information block, to produce the n digit code word.

The second major class of codes are tree codes, and these have as a more useful subclass the convolutional or recurrent codes. Tree codes, unlike block codes, do not operate on information sequences independently. Instead, information is processed continuously, each information sequence being associated with a longer (possibly semi-infinite) code sequence. A binary convolutional code encoder operates by firstly splitting the information sequence into k_0 digit blocks, where k_0 is usually small. Parity checks are then calculated by taking mod-2 sums not only on the present k_0 information block, but also on many previous blocks. These checks, together with the k_0 information digits form a basic n_0 digit section of the code sequence. For encoding and decoding to be practical the number of basic blocks (m) over which the code calculates checks must be restricted; and the number (mn_0) of digits which the decoder has to handle is called the constraint length (n) of the code. Convolutional codes originated with Elias (1954) and Wozencraft (1957); and the most important decoding

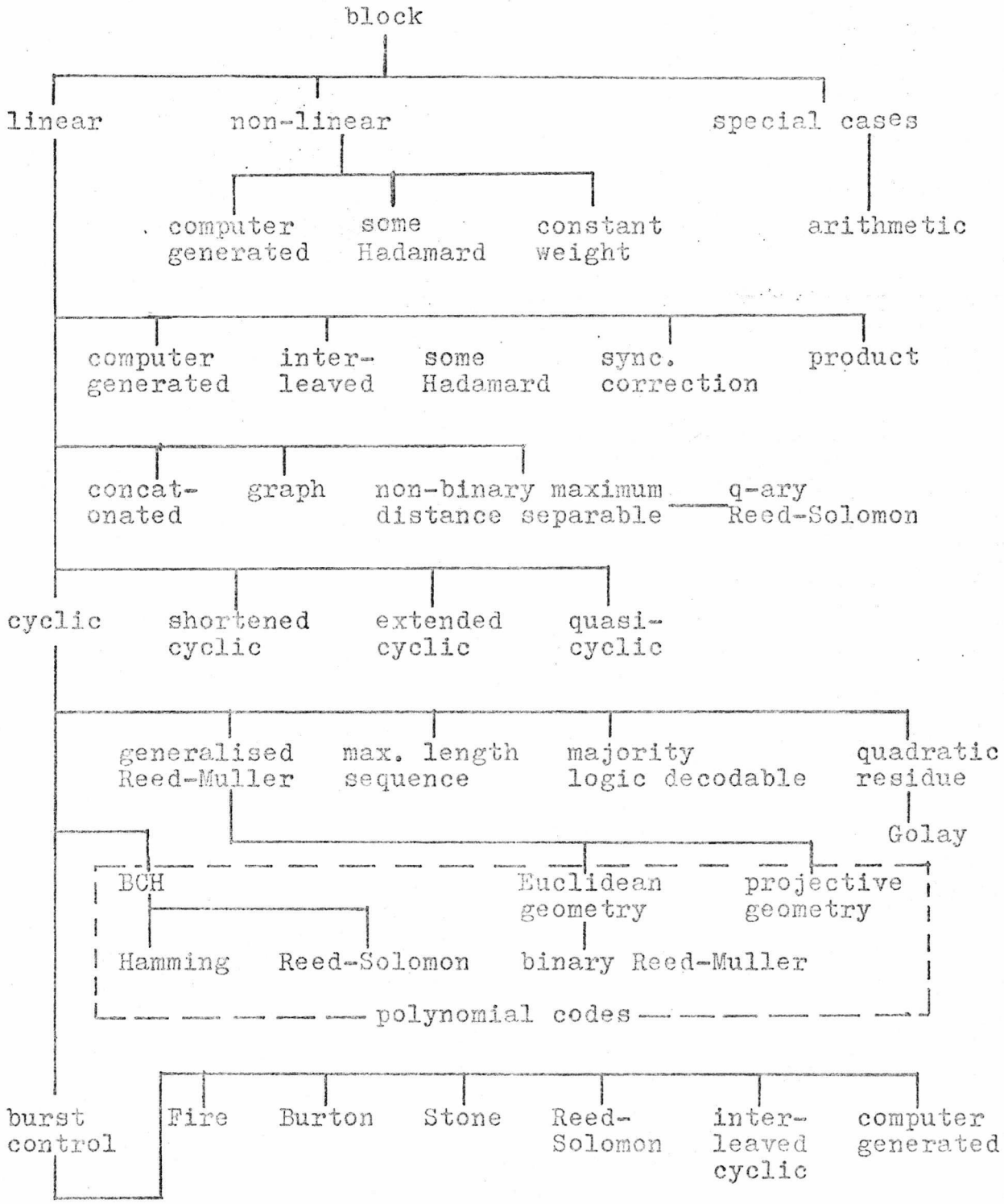
procedures are described by Massey (1963), Wozencraft (1957) and Viterbi (1967). Although, as yet, convolutional codes are not as thoroughly investigated as block codes, they do have similar, and in some cases superior, error control capabilities. Shannon's fundamental theorem holds for both types of code. Convolutional codes are not considered further in this thesis.

The first block codes were devised by Hamming (1950), and Golay (1949), and since then many new classes of code have been devised. Binary linear block codes have reasonably high mathematical structure in that the 2^k code vectors (n-tuples) have the properties of a group under mod-2 addition, and are vectors chosen from the vector space of all binary n-tuples. By imposing further structure in an attempt to generate powerful easily implemented codes, many subclasses of linear block codes have been devised, and attention will now be restricted to these codes. Figure 2.3 outlines the relationships between the main classes of block code. Individual codes, however, may belong to more than one subclass.

The study of cyclic codes started with Prange (1957, 1959), and reference to figure 2.3 shows that these codes form by far the largest collection of practical codes, including the powerful random error correcting BCH codes and the powerful burst correcting Reed-Solomon codes. Much work has gone into the generation of codes that are either genuine subclasses of cyclic codes, or strongly related to cyclic codes, in an attempt to realise

Figure 2.3

Code Classification



the advantages of easy implementation inherent in this large class of codes. Most of the codes in the above diagram are dealt with in chapter 3, and appropriate references appear there. Information on codes that appear in the diagram but which are not dealt with further may be found, with references, in Peterson (1972), Lin (1970), and Franco (1965).

2.4. *Hamming distance*

The concept of Hamming distance (Hamming 1950) is essential in discussing the EDC power of a code as it has the properties of a metric. The weight of a binary code word with block length n is defined as the number of 1's it contains; and the Hamming distance between two n-tuples is the number of positions in which their elements differ, that is, the weight of their mod-2 sum. Thus the Hamming distance between two n-tuples is a measure of their separation in the n-dimensional vector space of all n-tuples. When considering a linear block code the code vectors may not all be at equal Hamming distance from each other. In this case the minimum Hamming distance that occurs in the collection of code n-tuples determines the EDC power of the code, and is called the code distance (d). The minimum distance of the code is therefore equal to the minimum number of errors per block that will change one code word into another, thereby producing a situation that overloads the code and is neither detectable nor correctable.

If a code is used for error detection, then a detectable error pattern is one that does not change a code word into another valid code word; and therefore to guarantee the

detection of all patterns of d-1 or fewer errors in the block, an (n,k) code must have a minimum distance of at least d. The number of detectable error patterns (n_d) guaranteed by this lower bound is therefore

$$n_d \geq \binom{n}{d-1} + \binom{n}{d-2} + \ldots\ldots \binom{n}{1}. \qquad 2.1.$$

Many linear block codes, however, have an error detecting power much greater than that guaranteed by the bound, due to the group property of such codes. For these group codes the digit by digit mod-2 sum of any two code words generates another valid code word and therefore the only error patterns that can change one code word into another, the undetectable patterns, are patterns identical to the code words. The true number of detectable error patterns is therefore given by:

$$n_{td} = 2^n - 2^k , \qquad 2.2$$

and in general, $n_{td} >> n_d$.

The error correcting power of a code can also be related to distance by considering the requirement that all patterns of t or fewer errors must be corrected. One method of correction is to look for the code word that is the shortest distance away from the received word. Taking the worst case, a received n-tuple may be t units away from the transmitted code word, and in order to correct this we must require that the n-tuple is at least t+1 units of distance away from any other code word. Therefore, to guarantee the correction of all patterns of t or fewer errors a code must have distance 2t+1. Furthermore, considering that an error must be detected in order to be corrected, the simultaneous correction of all patterns of t or fewer errors

and detection of all patterns of $l \geq t$ errors requires a distance of $t+l+1$.

In order to increase the EDC power of a code the distance must increase. This means that either the number of code words must decrease while n is held constant, or that n must increase while k is held constant. In both cases this implies that further redundancy is added, thereby reducing the data throughput rate. If n is increased with the rate held constant then distance also increases and, unfortunately, so does codec complexity.

The structure of linear block codes enables the distance of such a code to be easily found because, due to the group property, the minimum distance of the code equals the weight of the minimum weight non-zero code word. Therefore, to find the distance of an arbitrary code, the code words are generated to find the minimum weight code word. This procedure can also be used to generate codes on a trial and error basis using a computer. For large n, however, this is impractical and accounts for the need to utilise further mathematical structure, in order to generate classes of codes for which the minimum distance can be explicitly stated.

2.5 *Decoding block codes, and the effect of the channel*

Decoding involves associating a received n-tuple (r) with a transmitted code word (c); and

this decision must take into account the effects of the particular channel.

The probability of erroneous decoding then depends on the code used, the channel error characteristics, and the decoding strategy employed. If all code words are equi-probable then the best decoding scheme is known as maximum likelihood decoding. To implement this scheme after receiving a vector r, the decoder computes the conditional probability $P(r/c)$, which depends on the channel transition probabilities, for all the 2^k code words. The code word c_t is identified as the one transmitted if $P(r/c_t)$ is largest. There are two major disadvantages to this scheme when taken at face value; firstly, 2^k computations are involved and this can render the scheme impracticable for long codes; secondly, precise knowledge of the channel is implied at all times, and this is difficult to achieve on real channels.

To assess the probability of erroneous decoding it is therefore essential to model the channel, including modems, in some reasonably simple way, in order to see if maximum likelihood or some other near-optimum decoding scheme can be implemented in a practical way, that is, one not involving the direct computation of probabilities.

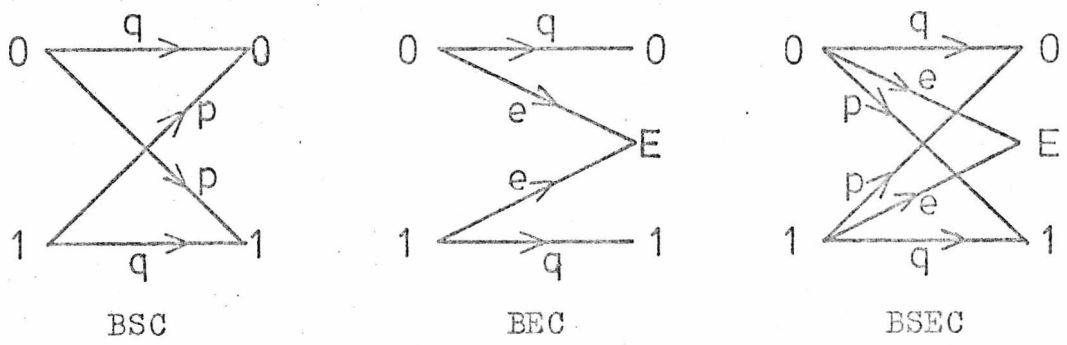
Channels may be roughly split into two types: memoryless channels, where errors occur independently at random; and channels with memory, where errors occur in bursts separated by relatively error free intervals. All real channels are a mixture of the two; but if a channel is predominantly of one type, a code designed specifically to cope with either burst or random errors can be chosen.

2.5.1 *Memoryless channels*

Memoryless channels are generally modelled by the binary symmetric channel (BSC), or the binary erasure channel (BEC), or both (BSEC). These channels are shown in Figure 2.4.

Figure 2.4

Memoryless Channels



(i) The BSC

The BSC is characterised by a single parameter p, the probability of a bit error; and the performance of a code used on this channel can be assessed by comparing the probability P(m,n) of m errors in a block of n digits, with the EDC power of the code. P(m,n) is given by the binomial distribution:

$$P(m,n) = \binom{n}{m} p^m q^{n-m} , \qquad 2.3$$

which shows that equal weight error patterns are equi-probable; and that one error is more probable than two errors, and so on.

It can be shown (Gallager 1968) that maximum likelihood decoding for the BSC is equivalent to choosing a code

vector c that is the shortest Hamming distance away from the received vector r.* If the received vector is equi-distant from two valid code words, then detection has occurred but the code is overloaded as far as correction is concerned, as it cannot uniquely decode the received vector.

Choosing the 'nearest' vector is equivalent to finding the minimum weight error pattern (e) that will change the received vector into a valid code word. That is, the minimum weight solution (e) to the equation

$$c = r \oplus e \ , \qquad 2.4$$

where $\oplus$ signifies modulo-2 addition. The distance properties of a linear code guarantee at most one solution having (d-1)/2 or fewer 1's, and so provided the number of errors per block does not exceed (d-1)/2, equation 2.4 can be solved uniquely for e and hence c. The complexity of such a solution is still, however, exponentially dependent on code length.

The term Algebraic Decoding refers to a class of decoding algorithms designed to systematically determine the minimum weight solution to equation 2.4 by exploiting the algebraic structure of the code; and are generally only guaranteed to work when there exists a solution of weight (d-1)/2 or less. These algorithms are designed to have a complexity that is not exponentially dependent on code length (as opposed to basic maximum likelihood decoding), and some of the correcting power of the code, as guaranteed by the minimum distance, may be sacrificed to achieve this.

*assuming $p < \frac{1}{2}$.

(ii) The BEC and BSEC

The erasure channel was introduced by Elias (1954), and corresponds to the case where the demodulator, lumped into the channel as in Figure 2.2, outputs an erasure symbol instead of an 0 or 1 in doubtful cases. This erasure symbol therefore corresponds to an error whose position is known; but whose magnitude is not. For the BEC a code with distance e+1 can correct all combinations of e or fewer erasures; and for the more practical BSEC model the simultaneous correction of t errors and e erasures requires a minimum distance of at least 2t+e+1. Due to the random nature of the BSEC, maximum likelihood decoding is again optimum, and general decoding for the erasure channel has been studied by Epstein (1958). By exploiting code structure, algebraic decoding algorithms that correct errors and erasures can be evolved, with only a slight increase in complexity over the pure BSC case (Gallager 1968, Peterson 1972).

2.5.2 *Channels with memory*

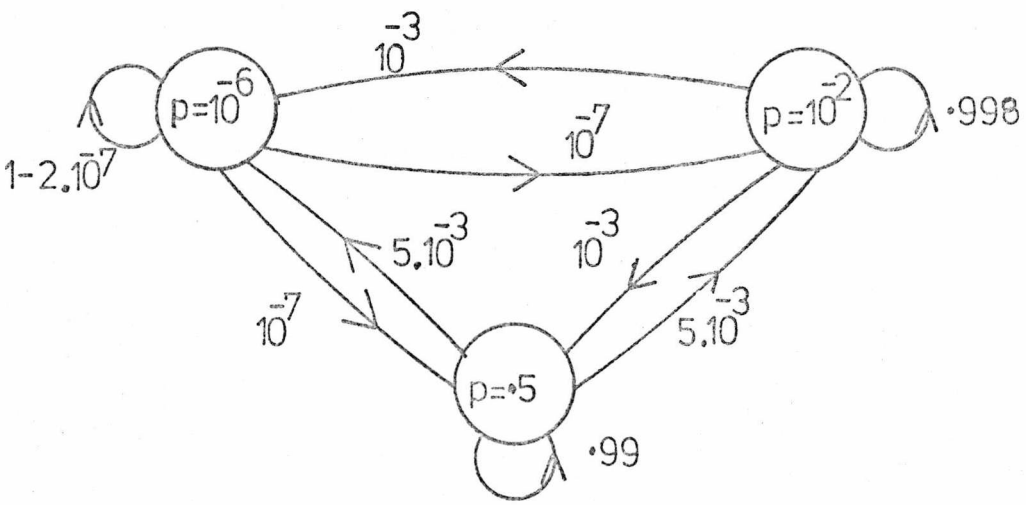
In general, real channels (such as HF and telephone) have memory, and are prone to bursts of errors, that is, a pattern of error digits starting and ending with an error. These channels can be characterised by their bit error rate and their P(m,n) distribution in the same way as the BSC; but the shape of the P(m,n) distribution is generally very different from that of the BSC. Further distributions such as the distribution of error-free intervals, and the distribution of burst lengths, are then

needed to fully characterise these channels.

A model of the burst channel originally proposed by Gilbert (1960) considers the channel to be composed of a small number of BSC's, with the state of the model, that is, the BSC in use at any particular time, being controlled by a Markov process. As the probability of remaining in a given state is greater than the probability of leaving and also because one state is usually chosen to have a bit error rate of 0.5, errors tend to occur in bursts. Generally, two or three states are sufficient to define a model whose P(m,n) or error free interval distribution is similar to that of a real channel. A three state Gilbert model, with transition and bit error probabilities, for a typical telephone channel is shown in Figure 2.5.

Figure 2.5

A Gilbert Model



Exact expressions for the P(m,n) and error-free interval distributions are given by Elliott (1963) using the two state Gilbert model, and therefore the performance of a code when used on this channel can be explicitly calculated. A modified Gilbert model that gives an excellent match to telephone data is given in Elliott (1965).

A partitioned Markov chain model proposed by Fritchman (1967) contains only one error state and no transitions between error free states, and is shown to be uniquely determined by the error free interval distribution. This model is therefore easier to fit to real channel statistics than a generalised Gilbert model, and Tsai (1969) shows a good fit can be obtained using error statistics from an HF channel.

Burst error correcting codes are generally designed to correct any burst of length b in a block of n, and so emphasis is on the length of the error pattern rather than the weight. Algebraic decoding methods that guarantee to correct bursts of up to a certain length also exist, and if it can be assumed that for the particular channel a burst of length b is less probable than one of length b-1, these decoding schemes are maximum likelihood and hence optimum.

Most real channels corrupt signals with both Gaussian and impulse noise, and are also time varying. Therefore, ideally, to cope with this situation a code capable of correcting both random and burst errors, as well as adapting to changing channel conditions, is required. Also real channels may insert or delete bits from the code sequence giving rise to timing errors which, hopefully, the decoder should be able to correct. Under these comprehensive error conditions it is difficult, if not impossible, to explicitly postulate an optimum decoding strategy. Good algebraic decoding methods can, however, be formulated, given that comprehensive channel models are

available; but in general, system performance over real channels must be assessed by comprehensive simulation or field trials.

2.5.3 *Probabilistic decoding*

A fundamentally different approach to algebraic decoding, is that of probabilistic decoding (e.g. sequential or soft decision decoding). Probabilistic decoding attempts to reduce the decoding effort by utilising the probabilistic nature of noise to replace one difficult decision (as to which code word was sent) with a number of simpler ones. Thus a tentative decision as to which code word was received is made, and on the basis of more incoming sections of the code sequence, the estimate is gradually iterated toward a hopefully correct decision. The amount of decoding effort is therefore dependent on the channel error conditions and may exceed the time available before a new word must be decoded.

Although sequential decoding techniques have been applied to the decoding of block codes (Gallager 1963, Massey 1963, Forney 1966, Fano 1963), they are most effective for decoding convolutional codes (Massey 1963, Wozencraft 1957, and 1961, Viterbi 1967), and will not be considered further. Soft decision techniques can be used with both block and non-block codes, and are capable of greatly improving overall system performance (Weldon 1971).

2.6 *The coding problem*

For codes to be powerful they must be long; but it is difficult to "randomly" generate good, long, codes even with

computer aid. This means that mathematical structure is required to generate classes of codes with guaranteed minimum distances. If codes are long it also becomes impossible to store all code words at the encoder and decoder, and structure must be further exploited to achieve practical implementation. In order to correct errors it is desirable to achieve an optimum decoding scheme, but this depends on the channel statistics which may vary if the channel varies. The coding problem is therefore essentially threefold: finding powerful codes with the highest possible rate that can cope with errors that are likely to arise on the particular channel; devising efficient encoding schemes based on a code's inherent mathematical structure; and finding practical methods of decoding that are optimum for the particular channels in use.

The engineering problem of setting up an error control system is essentially one of compromise and trade-off. An optimum solution to a particular problem cannot be found analytically; rather, out of many possible systems, a decision is formed. This decision must take account of: powerful coding, data throughput, final undetected or uncorrected error rate, equipment complexity, channel error statistics, channel availability, decoding delays, and, by no means least, user requirements.

CHAPTER 3

Linear Block Codes - Theory, Implementation and Capabilities

This chapter describes the properties of various classes of linear binary block codes, giving particular emphasis to the properties and implementation of the most important class, the cyclic codes. Due to the structure of linear codes this description is essentially mathematical and a knowledge of modern algebra is assumed.

The chapter is essentially a review of the relevant coding theory required for a full understanding of chapter 5, and as such may be omitted at a first reading by the suitably informed reader.

3.1 Linear binary block codes

The set of all *n-tuples* with binary elements, that is, elements from the Galois field of two elements, GF(2), form an n-dimensional vector space. A subset containing 2^k of these vectors is an (n,k) *linear block code* iff* they form a k-dimensional subspace of the vector space of all n-tuples. The code may therefore be described by any set of k linearly independent vectors which span the code space. The vectors in the code form a group under mod-2 addition: hence the sum of any two code vectors is itself a code vector, and the minimum distance of the code equals the minimum weight code vector. The set of all n-tuples orthogonal to a subspace V_1 of dimension k, forms a n-k dimenional subspace V_2, which is called the *null space* of V_1. Therefore, if

*if and only if.

a vector is orthogonal to every code vector, it is in the null space of the code, and it is possible to describe a linear code in terms of the null space of some vector space.

3.1.1. *Generator matrix description of linear block codes*

Any set of k linearly independent vectors which span a k-dimensional vector space is called a *basis* of the space, and any set of k basis vectors for a linear block code can be considered as rows of a matrix G, called a *generator matrix* of the code. The row space of G is the (n,k) linear code, and each of the 2^k distinct linear combinations of rows gives one of the 2^k distinct code vectors. The matrix description is therefore much more compact than a list of code vectors, and code words (u) can be considered to be generated by the matrix multiplication: u=dG, where d is the k digit message block.

There are as many generator matrices for a given code as there are sets of basis vectors. These matrices all generate the same row space, are called *row equivalent*, and can be obtained, one from the other, by a succession of *elementary row operations*. All generator matrices for a particular code, however, are row equivalent to only one matrix in *canonical echelon form*, and this matrix is therefore a unique description of the code. Every linear code is therefore described by one, and only one, generator matrix in canonical echelon form.

Two codes may differ only in the arrangement of their symbols within the code words. These codes will therefore have different code words and different canonical generator matrices, but will have identical weight structures and equal minimum distance. These codes are called *equivalent*, and their performance on a channel with *random* errors will be identical. It is important to note, however, that because the code words have different binary patterns, their burst correcting ability may not be the same.

If we consider a generator matrix G, then every permutation of columns results in the generator matrix of an equivalent code. Matrices that are obtained from one another by a combination of elementary row operations and column permutations are called *combinatorially equivalent*, and it is therefore possible to express all the generator matrices of all equivalent codes with just one combinatorially equivalent matrix in a standard form, called *reduced echelon form*. This form is shown in equation 3.1 below and has the leading 1's of each row arranged into a k × k identity matrix.

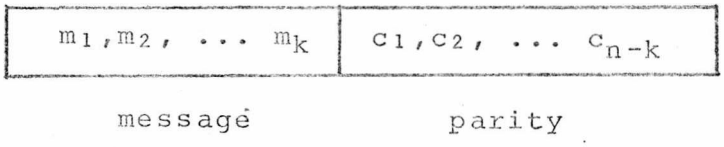
$$G = \begin{bmatrix} 1 0 \ldots 0 & p_{11} \ldots p_{1,n-k} \\ 0 1 \ldots 0 & p_{21} \ldots p_{2,n-k} \\ \cdot\cdot \quad \cdot \quad \cdot & \quad \cdot \\ \cdot\cdot \quad \cdot \quad \cdot & \quad \cdot \\ \cdot\cdot \quad \cdot \quad \cdot & \quad \cdot \\ 0 0 \ldots 1 & p_{k1} \ldots p_{k,n-k} \end{bmatrix} = \left[I_k P \right] \qquad 3.1$$

The code generated by this matrix is called a *systematic code* because the first k digits of each code word are identical to the k message digits; the encoding operation, u=dG, is therefore greatly simplified. Each of the last (n-k) digits is a linear combination of the k message digits

and those digits are called the redundant, or *parity check digits*. The matrix P therefore represents the parity check equations, the value of the ith parity check digit in a code word being given by the mod-2 sum of message symbols whose positions correspond to 1's in the ith column of P. The code words from a systematic code can therefore be considered as a block comprising of two sections: a k digit message section, followed by an (n-k) digit parity check section as shown in figure 3.1 below.

Figure 3.1

A Systematic Code Word



Because every generator matrix is combinatorially equivalent to some matrix in reduced echelon form, every linear code is equivalent to a systematic code. It is therefore possible to restrict the consideration of random error-correcting codes to systematic codes only, without loss of generality.

3.1.2 *Parity check matrix description of linear block codes*

An alternative matrix description of a linear code can be formed by considering the parity check matrix (H) of the code. A linear code V, generated by G, has a null space V' of dimension (n-k), and this null space can be considered to be generated by a matrix H that uses basis vectors for V' as rows. The code V is therefore the null space of V'

and a vector u is a code word iff,

$$uH^T = O \qquad 3.2$$

The (n-k) rows of H are essentially (n-k) independent equations which an n-tuple must satisfy in order to be a code word. These equations are called *generalised parity checks* because, in the binary case, they check certain digits in the code word for even parity.

If h_{ij} is an element of H, and if $u = (u_1, u_2, \ldots u_n)$, then equation 3.2 implies that u is a vector in the code generated by G iff, for each i (row):

$$\sum_j u_j h_{ij} = O \qquad 3.3$$

Also, as equation 3.2 holds for every code vector, it must hold for the k basis vectors of G, and therefore

$$GH^T = \left[O\right] \qquad 3.4$$

where $\left[O\right]$ denotes the k by (n-k) all-zero matrix.

For a systematic code with a generator matrix of the form of equation 3.1, the parity check matrix is given by

$$H = \left[P^T I_{n-k}\right] \qquad 3.5$$

and therefore

$$H^T = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} \qquad 3.6$$

Equation 3.3 then implies that, for each parity digit, the mod-2 sum of that digit and the message digits checked by it, as given by the appropriate column of P, must be zero.

If $\left[h_i\right]$ denotes the ith column vector of H, then equation 3.3 gives

$$\sum_{i=1}^{n} u_i \left[h_i\right] = 0 \qquad 3.7$$

and this is a linear dependence relation showing that w columns of H add to zero, where w is the number of non-zero components of u. If u is considered to be the minimum weight code vector, then the code that is the null space of H has minimum distance at least w, iff no combination of (w-1) or fewer colums of H add to zero. This indicates an heuristic method of constructing a code of weight w by progressively choosing columns of H such that no combination of (w-1) columns are linearly dependent.

The row space (V') of H, which is the null space of V, is also a subspace of the space of all n-tuples and is hence also a linear code. The code formed by taking rows of H as a basis is called the *dual code* of G, and every (n,k) linear code has an (n,n-k) dual code, and vice versa.

3.1.3 The syndrome

Consider a received vector r consisting of a code vector u that has been corrupted by an error vector e, that is,

$$r = u \oplus e \qquad 3.8$$

The function of the decoder is to recover u from r. Because r is not a code word,

$$s = rH^T \neq 0 \qquad 3.9$$

where the (n-k) digit vector s is called the *syndrome* of r.

Substituting equation 3.8 into equation 3.9 gives

$$s = (u \oplus e)H^T$$

$$= eH^T \qquad 3.10$$

because the syndrome of a code word (uH^T) is zero. The syndrome of a received n-tuple therefore only depends on the error vector corrupting the code word, and not on the actual code word sent. For a systematic code the syndrome may be found by recalculating the parity digits, by what is effectively encoding the message section of the received word, and then adding these recalculated checks to the received checks.

When using a linear code for error detection it is only necessary to test for a non-zero syndrome; for error correction, the value of the syndrome can be used to identify a particular error pattern, which can then be added to the received code word, thereby correcting it.

3.1.4 Coset decomposition and the standard array

Given a group G with elements (g_i) and a subgroup (H) of G, with elements (h_j), it is possible to express all the elements of G in an array called the coset decomposition of the group. The first row in the array is the subgroup, with the identity element (h_1) at the left. The first element in the second row is any element not in the first row, and the remaining elements in the row are obtained by multiplying each subgroup element by this left-most element. Further rows are formed in a similar manner, always placing a previously unused group element in the first column, until all the group elements appear in the array. The set of elements

in a row of the array is called a *coset*, and the element in the first column of a row is the *coset leader* of that row.

The code words (u) of a linear code form a group (under mod-2 addition) which is a subgroup of the group of all n-tuples. This group may therefore be decomposed into cosets starting with the code subgroup, and the array so formed is called the *standard array* (Slepian 1956). By choosing the 2^{n-k} coset leaders to be error patterns (e), it is then possible to use the array as a decoding table. A standard array for an (n,k) code is shown in figure 3.2.

Figure 3.2

The Standard Array

$u_1 = 0$	u_2	u_3 ...	u_{2^k}
e_2	$e_2 + u_2$	$e_2 + u_3$ ...	$e_2 + u_{2^k}$
.	.	.	.
.	.	.	.
.	.	.	.
$e_{2^{(n-k)}}$	$e_{2^{(n-k)}} + u_2$	$e_{2^{(n-k)}} + u_3$ ...	$e_{2^{(n-k)}} + u_{2^k}$

Every group element appears once and once only in the array and there are therefore exactly 2^{n-k} rows, or cosets, in the array.

The standard array when used as a decoding table gives correct decoding iff the error pattern caused by the channel is a coset leader. The 2^{n-k} coset leaders are called *correctable* error patterns, and for these patterns the required code vector heads the column in which the received vector lies. If the error pattern is not a coset leader, incorrect decoding results. Therefore, in order to minimise

the probability of erroneous decoding, the coset leaders are chosen to be the error patterns which are most likely to occur on the particular channel in use. For the BSC, increasingly weighty error patterns are decreasingly probable, and so coset leaders for this channel should be chosen to have minimum weight from amongst the remaining vectors (assuming $p<\frac{1}{2}$).

From figure 3.2 it can be seen that each row consists of code words plus a specific error vector, therefore all the 2^k n-tuples of a coset have the same syndrome, and the syndromes for each coset are different. This means that there is a one-to-one correspondence between a correctable error pattern (a coset leader) and a syndrome. It is therefore possible to have a decoding table that does not involve storing the entire standard array. This is called *look-up* decoding and is accomplished as follows.

(i) calculate the syndrome $s = rH^T$ of r.

(ii) locate the coset leader e_i whose syndrome is s, this is assumed to be the error pattern.

(iii) add the coset leader and received vector to give the transmitted code vector $u = (u+e_i)+e_i$.

This scheme, although simpler than using the whole array, still means that 2^{n-k} coset leaders and syndromes have to be stored, which may be impractical for large n-k. Alternatively, the decoder may store only the coset leaders, and calculate the corresponding syndromes each time a match with the received syndrome is required. The storage required is halved, but the computation can be prohibitive.

Generally, in an (n,k) code with minimum distance (2t+1) it is possible to construct a standard array such that all error patterns of weight t or less, and possibly some of weight t+1, can be used as coset leaders. If no patterns of weight t+1 are possible, the code is called a *perfect* code.

For a perfect code each coset leader can be chosen to be the minimum weight word in the coset; if some patterns of weight t+1 are correctable, then these cosets do not have a unique minimum weight word. A decoding algorithm that is maximum likelihood for the BSC is then:

(i) attempt to correct all patterns of weight t

(ii) only detect patterns of weight t+1.

Decoding by using the standard array has a complexity that is exponentially dependent on length, and to reduce this rate of complexity increase usually implies that the code has to have more restrictive properties than just linearity.

3.1.5 Step-by-step decoding

This decoding algorithm assumes that, for any received n-tuple (v), the weight of the coset leader $w_c(v)$ in which the n-tuple is located, can be easily determined. It is also assumed that the coset leaders (correctable error patterns) are chosen in groups of increasing weight. The most obvious way to determine $w_c(v)$ is to have available a table of syndromes and coset leader weights. If however, $w_c(v)$ can be computed from the syndrome in some simple manner, the decoding effort need not increase exponentially

with n.

The step-by-step decoding algorithm is then as follows.

(i) calculate $w_c(v)$.

(ii) invert the first digit of v to form a new vector v'

(iii) calculate $w_c(v')$

(iv) if $w_c(v')<w_c(v)$, the first digit was in error and has been corrected. If $w_c(v')>w_c(v)$, the first digit was correct and must be inverted back to its original value.

(v) repeat steps (iii) and (iv) for the other digit positions until $w_c(v')=0$, that is, v' is a code word.

The decoding effort in this procedure is $\leq$ n times the effort involved in calculating $w_c(v)$, and if the $w_c(v)$ computation is not exponentially dependent on n, then neither is the total decoding effort.

Step-by-step decoding always results in a code word that is the shortest distance away from the received n-tuple, and is therefore maximum-likelihood for the BSC.

3.1.6 Conclusions

This section has outlined methods by which any linear binary block code can be described. Also indicated were general methods of encoding, finding minimum distance, and decoding. It is important to note several points about these operations.

(i) Encoding does not require the storage of all the 2^k code words. Instead the $k \times (n-k)$ P matrix is stored, and code words can be generated as they are needed.

(ii) The minimum distance of the code can be determined by finding the minimum weight code word, an operation requiring 2^k steps.

(iii) A general decoding method involves the use of the standard array, but the disadvantage of this method is that complexity grows exponentially with block length. Step by step decoding is useful for a restricted sub class of linear codes, and has a complexity that is linearly dependent on block length.

As far as general code construction is concerned, this section has implied two possible methods.

(i) A code can be heuristically constructed by generating all possible P matrices for a given n and k. The minimum distance of the resulting (n,k) code is then determined by testing each code word for minimum weight.

(ii) A code with a guaranteed minimum distance of at least w can be generated by systematically choosing columns of the H matrix, such that no combination of (w-1) columns or less is linearly dependent.

The points raised above on implementation and construction, indicate that these general methods quickly become

impractical as n increases. The following sections of this chapter describe restricted classes of codes that can be constructed, even for large n, by the utilisation of further mathematical structure. This structure is then exploited to ensure that the codes are efficient, have an explicitly guaranteed minimum distance, and can be implemented (particularly decoded) in a practical way.

3.2 *Capabilities and limitations of linear codes*

In order to design new good codes and comparatively assess their error control performance, it is necessary to have a knowledge of the ultimate theoretical capabilities of codes, as well as a knowledge of what is achievable from known codes. Given this information it is then theoretically possible to say for each (n,k) which codes are 'best' and which codes fall significantly short of theoretical expectations.

To calculate the values of these capabilities exactly is difficult, if not impossible. Instead, upper and lower bounds have been derived in order to restrict these capabilities to a range of values. Bounds generally fall into two main classes: bounds on minimum distance, and bounds on error control performance. Performance bounds are generally more complex than distance bounds as they depend on the type of errors introduced by the channel.

This section looks at bounds on minimum distance, performance over the BSC, and burst error control capability.

3.2.1 Minimum distance bounds

The parameter of interest here is $d_{MAX}(n,k)$, that is, the maximum minimum distance attainable with a linear (n,k) code. Despite a great deal of research, $d_{MAX}(n,k)$ is only known exactly for $k \leq 5$, all n and k for which $d_{MAX}(n,k) \leq 3$ and a few other isolated values. There is no theoretical approach to a general solution, and computing d(n,k) for all (n,k) codes is impractical if n is at all large. Upper and lower bounds have, however, been derived and can be evaluated in two ways. Firstly, the bounds can be calculated for exact values of n and k. This involves a large amount of computation but is necessary for small n. Secondly, these bounds usually approach a simple asymptotic form for large n, the value of which can be easily calculated. Bounds for exact values of n up to 127 have been calculated by Helgert and Stinaff (1973); for larger n the asymptotic bounds are fairly sharp, and attention will now be restricted to these.

Upper bounds are the most interesting as they indicate what is theoretically achievable; lower bounds merely state what is possible. Most bounds are based on the sphere packing argument proposed by Hamming (Hamming 1950, Johnson 1971, Gilbert 1952, Wax 1959) the 'average distance' approach of Plotkin (Plotkin 1960, Griesmer 1960) or a combination of these (Wyner 1965). Lower bounds on distance are provided by a number of well known codes, such as the Hamming and BCH codes.

The Plotkin (1960) upper bound is a relatively crude bound on d that relies on the fact that the minimum weight code word in a linear code is at most equal to the average word weight. The asymptotic form for large n is:

$$1 - \frac{k}{n} \geq \frac{2d}{n} \qquad 3.11$$

The Hamming (1950) upper bound on d is derived by noting that if a code is to correct all combinations of t or fewer errors, then all errors of weight t or less must be coset leaders. Therefore the number of error patterns must be no greater than the number of cosets (2^{n-k}). For large n, the asymptotic form is:

$$1-\frac{k}{n} \geq H\left(\frac{d}{2n}\right) \qquad 3.12$$

where $H(x) = -x\log x-(1-x)\log(1-x)$.

The Elias upper bound employs concepts that are used in both the Plotkin and Hamming bounds, and for large n is tighter than either (Peterson 1972).

Other upper bounds on d, which are tighter in certain cases than the above three bounds, have also been derived (Johnson 1962, 1963, Wax 1959). These bounds are generally more complex for small n than the Hamming or Plotkin bounds, and for large n none improve on the asymptotic Elias bound.

Upper bounds indicate that it is impossible to construct codes with a given n and k that have greater distance than that given by the bounds. It is therefore important to know what distance is achievable. Specific answers to this problem are provided by the multitude of known codes; a

general solution is given by the Varsharmov-Gilbert-Sacks (VGS) lower bound on d. This bound is a refinement of Gilbert's (1952) bound and was found independently by Varsharmov (1957) and Sacks (1958).

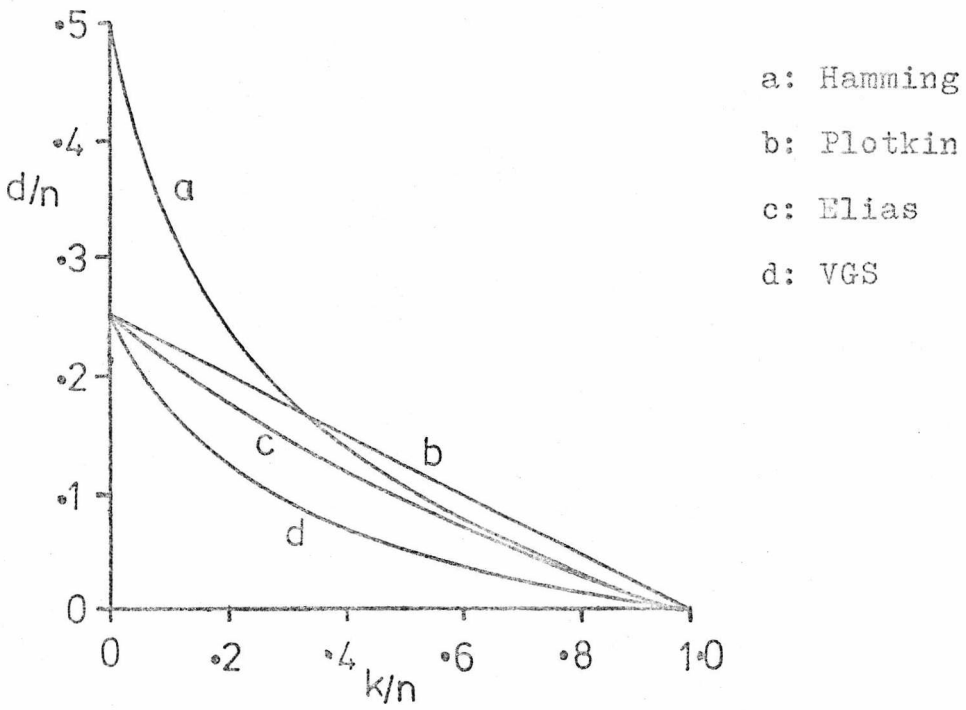
The VGS bound employs the fact that, in an (n,k) linear code with distance d, no combination of d-1 or fewer columns of the parity check matrix (H) are linearly dependent. The asymptotic form of the bound is

$$H\left(\frac{d}{n}\right) \geq 1-\frac{k}{n} \qquad 3.13$$

The four asymptotic bounds mentioned so far are shown plotted in figure 3.3 below.

Figure 3.3

Asymptotic Distance Bounds



The bounds of figure 3.3 indicate the theoretical limits on minimum distance possible with an (n,k) code. Known codes, however, fall significantly short of these limits when n is large. For small n the Hamming codes,

and the equivalent highest rate BCH and Reed-Muller codes, meet the Hamming bound. The lowest rate Reed-Muller and BCH codes meet the Plotkin bound. These codes therefore have maximum minimum distance. For large n , however, their rates all approach 0 or 1. In the middle rate range, with the rate fixed, both BCH and majority-logic codes have the ratio d/n tending to zero when n tends to infinity. The BCH codes, however, still lie approximately on the VGS bound for n=1023. In fact there is no known coding system for which it has been proved that d/n remains non-zero as n tends to infinity with the rate fixed, although the VGS bound shows that such codes exist. It would be of great theoretical interest to find codes that achieve what the bounds say is possible, although their practical use would depend on ease of implementation.

3.2.2 *Performance bounds for random error correction on the BSC*

The performance of a code when used over a particular channel can be measured in various ways. In order to simplify matters, it is usual to assume that the channel can be modelled by the BSC, and that 'performance' can be measured by P_e, the probability of erroneously decoding a block. A code can then be called *optimum* for the BSC if its value of P_e is as small as for any other code with the same n and k. This section gives an explicit expression for P_e which is applicable to any code, and then goes on to consider some upper and lower bounds on P_e.

For the BSC, equal weight error patterns are equiprobable, and the channel is described by the binomial distribution P(m,n), the probability of m errors occurring in a block of length n. The probability of a decoding error is then given by,

$$P_e = \sum_{m=0}^{n} a_m . P(m,n) \qquad 3.14$$

where a_m is the probability of erroneously decoding an error pattern of weight m, using the particular code. The parameter a_m is purely a function of the code and the decoding algorithm used, and, for certain cases, there may not exist an analytic method for its determination. For example a perfect code being maximum-likelihood decoded would have $a_m=0$ for $m \leq t$, and $a_m=1$ for $m>t$. A quasi-perfect code, however, would have $a_m \leq 1$ for $m=t+1$, and these a_m may not be easily determinable. Equation 3.14 therefore gives rise to an upper bound on P_e.

$$P_e \leq \sum_{i=t+1}^{n} P(m,n) \qquad 3.15$$

In some cases, particularly when n is large and soft decision decoding is employed, a_m may be very much less than 1 for m slightly greater than t, and equation 3.15 may only provide a crude upper bound on P_e. For the BSC, P(m,n) is the binomial distribution and P_e in equation 3.15 is simply the tail of this distribution. Application of Sterling's approximation to equation 3.15 shows that if the ratio t/n is kept fixed, then P_e decreases exponentially with increasing n. This behaviour is exhibited in BCH codes provided n is not too large, thereby indicating that very low probability of error is possible on truly memoryless

channels whilst using known codes. Equation 3.15 can also be combined with the VGS lower bound to provide an upper bound on P_e which is reasonably good for low rates. Much better bounds have however been derived, and the two most important of these are the sphere-packing bound and the random coding bound. These are described in Peterson (1972) and Gallager (1968), and even tighter bounds are given in Elias (1955, 1956), Fano (1961), Shannon (1957), and Shannon, Gallager and Berlekamp (1967).

Bounds on P_e again indicate that known codes fall short of theoretical expectations when n is at all large. Shannon's fundamental theorem proves that for any rate less than channel capacity, codes exist that will give an arbitrarily small probability of error. However, only two non-random coding schemes, Elias's error free coding and Forney's concatonated codes, are known to have a probability of error that approaches zero while maintaining a non-zero rate as n increases. The bounds indicate that good codes can be found by choosing a modest number at random and keeping the best, and that in order to approach channel capacity a code must correct most error patterns of weight slightly greater than (d-1)/2, although it is only possible to correct all patterns of weight up to (d-1)/2.

3.2.3 *Performance of error detecting codes on the BSC*

For error control systems that are wholly or partially error detecting, there are three probabilities of interest. These are: P_e, the probability of erroneous decoding;

P_d, the probability of detecting an error; and P_c, the probability of correct decoding. These probabilities are related by the expression,

$$P_c + P_d + P_e = 1 \qquad 3.16$$

If limited error correction is employed then P_c is the probability of getting t or fewer errors. If no error correction is employed, however, P_c is the probability that the error pattern is the all zero n-tuple and is given by,

$$P_c = P(o,n) \qquad 3.17$$

The probability of erroneous decoding, on the other hand, is the probability that the error pattern is one of the 2^k-1 non-zero code words. If the weight distribution of the code is W(m), and assuming that equal weight error patterns are equiprobable on the BSC, then,

$$P_e = \sum_{m=d}^{n} \frac{W(m)}{\binom{n}{m}} P(m,n) \qquad 3.18$$

where W(m) is the number of code words with weight m. Considerable effort has been directed towards finding the exact weight structure of various codes (Berlekamp 1968), but often this is impossible and in such cases an approximation to W(m) is useful. Many random error correcting codes, such as the BCH codes, have been found to have a weight distribution that approximates to the distribution of binomial coefficients (Peterson 1967). That is,

$$W(m) \approx \frac{\binom{n}{m}}{2^{n-k}} \qquad 3.19$$

so that equation 3.18 then reduces to

$$P_e \approx \frac{1}{2^{n-k}} \sum_{m=d}^{n} P(m,n) \qquad 3.20$$

This result is easy to evaluate and gives a reasonable approximation to P_e even when the weight structure of a particular code is substantially different from the binomial.* Using this result together with equations 3.16 and 3.17 gives an expression for the probability of detecting an error:

$$P_d \approx \sum_{m=1}^{n} P(m,n) - \frac{1}{2^{n-k}} \sum_{m=d}^{n} P(m,n) \qquad 3.21$$

where the second term is often considerably smaller than the first and can therefore be ignored.

By comparing equations 3.14, 3.15 and 3.20, it can be seen that because a_m is never as small as $1/2^{n-k}$, considerably lower error rates can be obtained by using error detection rather than FEC. In addition, this advantage is still realised if a small amount of error correction is introduced (Berlekamp 1968).

3.2.4 *Burst error control capabilities of codes*

This section considers, firstly, bounds on the burst detecting/correcting power of a code, and secondly, means of assessing the performance of a linear code when used over a burst noise channel.

A burst of length l is defined as a pattern of digits, the first and last of which are non-zero. A code with burst error detecting and/or correcting power b, is then defined

*(Lucky 1968)

as a code which is capable of detecting and/or correcting all bursts of length b or less.

It is possible to explicitly state burst *detecting* capability by noting that if no code word is a burst of length l or less, then no burst of length l can change a code word into another code word. In order to have no code word that is a burst of length l or less the code must have l parity check digits, and the burst detecting capability (b') of the code is then,

$$b' = n-k \qquad 3.22$$

An upper bound on the burst *correcting* ability (b) of a code is due to Reiger (1960) and states that

$$b \leq \frac{n-k}{2} \qquad 3.23$$

In addition, if a code is to correct all bursts of length b or less and simultaneously detect all bursts of length $l \geq b$ or less, the code must have at least b + l parity checks

Another upper bound on b can be found by considering that each correctable burst must have a unique syndrome. The number of syndromes is 2^{n-k}, which must at least equal the number of burst patterns of length b or less. This gives:

$$n-k \geq b-1 + \log n \qquad 3.24$$

If end-around bursts are discounted a similar result to that of equation 3.24, due to Fire (1959) gives:

$$n-k \geq b-1 + \log(n-b+2) \qquad 3.25$$

A lower bound on b that is analagous to the VGS bound was found by Compopiano, and is described in Peterson (1972). This bound states that it is possible to construct an (n,k) code that is $b<n/2$ burst-correcting, if

$$n-k \leq 2b + \log (n-2b) \qquad 3.26$$

For large n, equation 3.26 states that all values of $b<(n-k)/2$ are attainable, while equation 3.23 states that no higher values can be attained. Therefore when n is large, a practically attainable maximum for burst-correcting power is $(n-k)/2$.

In order to assess the performance of a burst-correcting code, it is necessary to have a knowledge of the burst error statistics of the channel. These may be obtained from measurements taken on a real channel, or by considering any of the burst channel models mentioned in section 2.5. It is usual to make the simplifying assumption that equal length bursts are equiprobable so that the channel can be characterised by its P(l,n) distribution, that is, the probability of a burst of length l in a block of length n. The probability of erroneous decoding is then given by

$$P_e = \sum_{l=0}^{n} a_l P(l,n) \qquad 3.27$$

where a_l is the probability of erroneously decoding a burst of length l. The decoding algorithm determines the values of a_l, and the exact values of a_l, for $b<l\leq n-k$, have to be found by analysis of specific systems. General bounds have, however, been derived by Gallager (Lucky 1968), and are tight for large (n-k). If bursts of length not greater than b are the only ones corrected, then $a_l=1$ for

$b<l\leq n-k$, and equation 3.27 gives,

$$P_e = \sum_{l=b+1}^{n} P(l,n) \qquad 3.28$$

which can be evaluated from either models or real measurements.

3.3 *Some specific block codes*

This section briefly mentions several important classes of block codes. Some of these codes are equivalent to cyclic codes and can therefore be easily implemented.

(i) Hamming codes - these are a class of single-error-correcting (SEC) codes that are perfect for all $n=2^{n-k}-1$, quasi-perfect for all n, and are therefore optimum for the BSC. In Hamming's original arrangement the codes were non-systematic in order to make the syndrome equal the binary position of the error, and therefore simplify decoding. A quasi-perfect distance 4 Hamming code can be formed by adding a single overall parity check to the distance 3 code. The weight structures of these codes are known (MacWilliams 1963, Peterson 1972) and their performance can therefore be calculated (section 3.2).

(ii) The Golay (23,12) code - this is a perfect triple-error-correcting code (Golay 1949) and is equivalent to a cyclic code. A (24,12) distance 8 code can be formed, and weight structures are again known. The Golay, Hamming, and trivial $n=2m+1$ repeated-bit codes are the only known perfect binary codes, and it seems likely that there are no others (Lloyd 1957).

(iii) Reed-Muller (RM) codes - the Reed (1954)-Muller (1954) codes were first discovered by Mitani (1951), and are a subclass of Euclidian geometry codes, and are in fact equivalent to cyclic codes with an overall parity check added. The codes can be easily decoded by the 'majority testing' procedure of Reed ; the forerunner of majority-logic decoding.

(iv) Product (Iterative) codes - In this method two or more codes are combined (iterated) to form a more powerful code that has both random and burst-correcting power. Taking row and column parity checks on an array of data is an example, and the method generalises to 3 and higher dimensional arrays. Elias' 'error-free' coding scheme uses the Hamming distance 4 code of length 2^m. This code is successively iterated with itself thereby doubling the length of the product code at each iteration. Probability of error (on the BSC) tends to zero whilst the rate tends to a non-zero limit (error free coding), as the iterations increase. Unfortunately the ratio of distance to length tends to zero, thereby falling short of what the VGS bound indicates is possible. Decoding product codes by using decoders for the individual codes is dealt with by Reddy and Robinson (1970); but in some cases product codes are equivalent to cyclic codes (Burton and Weldon 1965), thereby easing the implementation problem. Distance and performance bounds are given in Peterson (1972).

(v) Low-density codes - these codes were originally defined by Gallager (1963), and have a low density of ones in the parity check matrix. Decoding can be done by a straight majority procedure, by a variable threshold majority procedure, or by a more complex algorithm that utilises the demodulator's *a posteriori* probabilities (Gallager 1963).

(vi) Concatonated codes - this system of coding was proposed by Fornay (1966) and consists of combining two codes in tandem to form a more powerful code in a way that resembles product coding. An 'inner' (n,k) binary code is *concatonated* with an 'outer' (N,K) code (usually a Reed-Solomon code using symbols, or bytes, from $GF(2^k)$), to form an (Nn,Kk) concatonated binary code.

3.4 *Cyclic codes*

Cyclic codes are a subclass of linear codes that possess considerable mathematical structure, and due to this structure it is possible to find efficient decoding algorithms for these codes. In addition, encoding and syndrome calculation are easily achieved by using linear feedback shift registers, making these codes ideal for error detection systems. Cyclic codes were first studied by Prange (1957) and have received considerable attention since then, particularly in finding sub-classes of cyclic codes that are easily decodable.

This section presents the general properties of cyclic codes, including implementation, before specific sub-classes of these codes are considered later on in this chapter.

3.4.1 *Polynomial description of cyclic codes*

A cyclic (n,k) code is a cyclic subspace of the vector space of all n-tuples with the property that every cyclic shift of every code vector is another vector in the code space. That is, if an n-tuple

$$u = (u_{n-1}, u_{n-2}, \ldots u_o)$$

is a code vector, then the n-tuple

$$u^i = (u_{i-1}, u_{i-2}, \ldots u_o, u_{n-1}, \ldots u_{i+1}, u_i),$$

obtained by shifting u cyclicly i places to the right, is also a code vector.

There is a one-to-one correspondence between n-tuples and elements in the algebra (A_n) of polynomials modulo X^n-1 over a finite field F. These elements are *residue classes* of polynomials, and every residue class equals a polynomial of degree less than n in S, where S denotes the residue class that contains the polynomial X. It is convenient to represent the residue class that *contains* X as x, and in this case every n-tuple corresponds to a polynomial of degree less than n in x, whose coefficients are elements of a finite field F.

The elements of the algebra A_n can therefore be interchangeably denoted by the n-tuple

$$u = (u_{n-1}, u_{n-2}, \ldots u_o),$$

or by the polynomial

$$u(x) = u_{n-1}x^{n-1} + u_{n-2}x^{n-2} + \ldots + u_1x + u_o,$$

whose coefficients are binary digits if the algebra A_n is over GF(2). The sum of two n-tuples corresponds to

the sum of two polynomials, and multiplication by a field element carries over similarly. As far as orthogonality is concerned, however, n-tuples and polynomials differ. For two n-tuples to be orthogonal, the inner product of the vectors they represent must be zero. For two polynomials b(x) and d(x) to be orthogonal, however, implies $b(x)d(x)=0$. By considering such a multiplication it can be seen that orthogonality of polynomials implies that the vector corresponding to b(x) is orthogonal to the vector corresponding to d(x) with the order of components reversed, and to every cyclic shift of this vector. Bearing these points in mind, the terms code vector, code word and code polynomial can be used interchangeably.

It is now possible to arrive at a definition of a cyclic code: a binary cyclic code is a cyclic subspace made up of elements from A_n, and is in fact an *ideal* in the algebra A_n of polynomials modulo x^n-1 over GF(2). The structure of ideals in A_n states that each distinct ideal of dimension k, is generated by a distinct divisior of x^n-1. This divisior has degree (n-k) and is called the *generator* g(x) of the ideal. Therefore, every cyclic (n,k) code is completely specified by a monic polynomial of degree (n-k) that divides x^n-1, called the *generator polynomial* g(x) of the code, and every distinct divisior of x^n-1 generates a distinct (n,k) code. Alternatively the code (C) can be completely specified by saying that it is in the null space of the ideal generated by $h(x)=(x^n-1)/g(x)$. The polynomial h(x) is called the *parity check polynomial* of C, and because it divides

(x^n-1) can be used to generate a code which is *equivalent* to the *dual* code of C. Every element in the ideal is a multiple of g(x), and, conversely, an element is in the ideal iff it is divisible by g(x). The ideal contains 2^k elements, and is spanned by any k linearly independent elements. A binary cyclic code is therefore spanned by any set of k linearly independent vectors, all of which are multiples of g(x), and these vectors can be used as basis vectors of the cyclic subspace, and arranged into a generator matrix whose row space is the cyclic code. In particular, the code is spanned by the vectors corresponding to g(x), $xg(x)$, $x^2g(x)$, ... $x^{k-1}g(x)$, that is, g(x) together with every left cyclic shift of g(x), up to the (k-1)th shift.

The Euclidean division algorithm for polynomials states that for every pair of polynomials s(x) and d(x), there is a unique pair of polynomials, q(x), the quotient, and r(x), the remainder, such that

$$s(x) = d(x)q(x)+r(x) \qquad 3.29$$

where the degree of r(x) is less than the degree of d(x). If f(x) is a code polynomial, and g(x) is the generator polynomial of the code, then

$$f(x) = d(x)g(x)+r(x) \qquad 3.30$$

and the remainder r(x) is identically equal to zero because f(x) is a multiple of g(x). If f(x) is not a code polynomial then r(x) is non-zero and has degree less than d(x). Thus in order to test whether or not a polynomial f(x) is a code polynomial, it is only necessary to divide f(x) by g(x) and test for a non-zero remainder. This division can be accomplished using linear feedback shift registers and is

obviously much easier than the general requirement for linear codes: that a vector (v) is a code vector iff $vH^T=0$, which is a matrix multiplication.

3.4.2 *Description of cyclic codes in terms of generator polynomial roots*

It is possible to specify a cyclic code in terms of the roots of the generator g(x) of the ideal. The coefficients for a binary generator polynomial are in the *ground field*, that is GF(2), while the roots are in an *extension field* of the ground field. Firstly, consider the result obtained by assuming a first degree divisior $(x-\alpha)$ of g(x). The Euclidean division algorithm (equation 3.29) then gives,

$$g(x) = (x-\alpha)q(x)+r \qquad 3.31$$

where r must be a ground field element because it must have degree less than the divisior, that is, it must have degree zero. Then if $g(\alpha)=0$, that is, α is a root of g(x), r=0 and $(x-\alpha)$ is a factor of g(x). Therefore each distinct root of g(x) gives rise to a distinct first degree factor and g(x) can be completely specified by its first degree roots. Since the degree of g(x) must equal the sum of the degrees of its factors, the degree of g(x) is at least as great as the number of roots of g(x), and is equal to the number of roots if there are no repeated roots.

The polynomial x^n-1, with n odd, can also be factorised into n distinct linear factors, the n roots of unity:

$$(x-\alpha_1)(x-\alpha_2)\ \ldots\ (x-\alpha_n) \qquad 3.32$$

and because every generator divides x^n-1 the roots of the generator are a subset of these roots. If a is a *primitive* nth root of unity then a is a solution of x^n-1, that is, $a^n=1$. It is possible to express these n roots as powers of a, and the multiplicative group of these roots is cyclic, that is,

$$a^1, a^2, \ldots a^{n-1}, a^n=1=a^0 \qquad 3.33$$

where n is the smallest power such that $a^n=1$, and is called the order of the group.

It is also possible to factorise x^n-1 into a set of irreducible polynomials each of which has m unique roots where m is the degree of the irreducible factor. Any generator polynomial is a product of these irreducible factors and the roots of the generator are the roots of all its constituent irreducible factors. Any generator polynomial is therefore specified by a list of roots that are roots of unity, and these roots are the roots of one or more of the irreducible factors of x^n-1.

The roots of x^n-1 are elements in the extension field $GF(2^m)$ where $m<n$, and, if $n=2^m-1$, then the polynomial $x^{(2^m-1)}-1$ has as roots *all* the (2^m-1) roots of $GF(2^m)$. If $n \neq 2^p-1$ then the roots of x^n-1 are a (cyclic) subset of the elements of $GF(2^m)$. It is therefore important to investigate $GF(2^m)$.

Formally, $GF(2^m)$ is the field of polynomials over GF(2) modulo an irreducible polynomial of degree m. The elements of $GF(2^m)$ can be formed by taking all the powers

of the primitive element α modulo any irreducible polynomial p(x) of degree m. If the primitive element is also a root of p(x) then p(x) is called a *primitive polynomial*, and for every m there is at least one primitive polynomial.

For example, the Galois field of 2^4 elements, GF(2^4) may be formed as the field of polynomials over GF(2) modulo x^4+x+1. If the element α is a root of x^4+x+1 then $\alpha^4=\alpha+1$. The element α is also a primitve element and its powers generate all the elements of GF(2^4). Table 3.1 below shows the representation of GF(2^4) modulo x^4+x+1.

Table 3.1

GF(2^4)

α^0	=					1	=	0001
α^1	=			α			=	0010
α^2	=		α^2				=	0100
α^3	=	α^3					=	1000
α^4	=			α	+	1	=	0011
α^5	=		α^2	+ α			=	0110
α^6	=	α^3	+ α^2				=	1100
α^7	=	α^3		+ α	+	1	=	1011
α^8	=		α^2		+	1	=	0101
α^9	=	α^3		+ α			=	1010
α^{10}	=		α^2	+ α	+	1	=	0111
α^{11}	=	α^3	+ α^2	+ α			=	1110
α^{12}	=	α^3	+ α^2	+ α	+	1	=	1111
α^{13}	=	α^3	+ α^2		+	1	=	1101
α^{14}	=	α^3			+	1	=	1001
α^{15}	=				+	1	=	α^0

It can be shown, by forming a table similar to table 3.1, that α^2, α^4 and α^8 are also primitive elements of $GF(2^4)$ and that their powers generate all the elements of $GF(2^4)$. This is because α is a root of x^4+x+1, and in general if b_i is a root of p(x) then so is $b_i^{2^m}$.

Consider $x^4+x^3+x^2+x+1$ which is also irreducible over GF(2) and let b be a root of this. Therefore, $b^4=b^3+b^2+b+1$ and table 3.2 shows the field of polynomials modulo $x^4+x^3+x^2+x+1$.

Table 3.2

		GF(2)
b^0 =		b^0 = 0001
b^1 =		b^1 = 0010
b^2 =	b^2	= 0100
b^3 =	b^3	= 1000
b^4 =	$b^3 + b^2 + b + 1$	= 1111
b^5 =		b^0 = 0001

Therefore b is not a primitve element of $GF(2^4)$ and $x^4+x^3+x^2+x+1$ is not a primitive polynomial. The roots of $x^4+x^3+x^2+x+1$ are therefore a subset of the elements of $GF(2^4)$ and form a cyclic group of order 5, that is they are the roots of x^5-1.

Returning to the representation of $GF(2^m)$ in table 3.1, consider an element b_i; then the smallest degree monic polynomial $m_i(x)$ with coefficients in the ground field GF(2) such that $m_i(b_i)=0$ is called the *minimum polynomial* of b_i and is irreducible over GF(2). It is possible to express any polynomial in the ground field,

and in particular to express x^n-1 and $g(x)$, as the product of one or more minimum polynomials; or by a list of the roots of these polynomials. The generator polynomial is completely specified by a list of the powers of α, the primitive element of $GF(2^m)$, which correspond to roots of its constituent minimum polynomials. In addition, $f(x)$ is a code polynomial iff $f(x)$ is divisible by $m_1(x)$, $m_2(x)$, ..., $m_r(x)$.

3.4.3 *Matrix description of cyclic codes*

In section 3.4.1. the generator matrix resulting from the basis vectors chosen gives rise to a non-systematic code. It is, however, possible to put this matrix in reduced echelon form by using *row* combinations only, and the resulting systematic code is the same code as before, not just an equivalent code. The basis vectors can be formed successively from the generator polynomial as follows. The kth row of the matrix is the generator polynomial. The (k-1)th row is formed by shifting $g(x)$ one place left and adding $g(x)$ to this if the kth element in the row is one. This process is repeated until all k rows are formed, and ensures that a k-by-k identity sub-matrix appears on the left of the G matrix.

3.4.4 *Some binary cyclic codes*

(i) Codes that have $g(x)=(x-1)$ have minimum distance 2 and are single-parity-check codes. The dual code of this has $g(x)=(x^n-1)/(x-1)$, consists of an information digit repeated (n-1) times, and has distance n. Generator polynomials

which divide x^m-1, where $m<n$, have x^m-1 as a weight two code vector, and hence the code has distance two. If n is even, then (x^2-1) is a repeated factor and therefore cyclic codes of even length have distance 2.

The binary $(2^m-1, 2^m-m-1)$ Hamming codes of section 3.3 are equivalent to cyclic codes, and the generator polynomial of these codes is primitive. Therefore every primitive polynomial generates an (n,k) Hamming code. A cyclic code with distance 4 can be generated by the polynomial $(x+1)p(x)$ where $p(x)$ is primitive, and this is the single-error-correcting, double-error-detecting Hamming code of length 2^m.

(ii) A *maximum-length-sequence* code is one in which all the non-zero code words are all n shifts of a generator polynomial that is identical to a *Pseudo-Noise* (PN) sequence. These codes exist for all values of $(n,k)=(2^m-1,m)$, and the parity check polynomial of such a code is primitive. The code is therefore the dual of the cyclic Hamming code mentioned above. The generator polynomial being a PN sequence, has 2^{m-1} ones and $2^{m-1}-1$ zeros, and the distance of the code is therefore 2^{m-1}.

(iii) *Quadratic residue* codes were first considered by Prange (1958) and give rise to a class of cyclic codes of length n, where n is a prime number of the form $p=8m\pm1$. In this case x^p-1 factors into $(x-1)f_1(x)f_2(x)$, where the roots of $f_1(x)$ are quadratic residues of p and the roots of $f_2(x)$ are nonresidues. The codes generated by $f_1(x)$,

$(x-1)f_1(x)$, $f_2(x)$, and $(x-1)f_2(x)$ are then quadratic residue codes. Peterson 1972 gives a list of some $(n,(n+1)/2)$ quadratic residue codes for which d is known exactly (e.g. the Golay code), and Berlekamp (1968) give some additional codes for which only an upper bound on d is known.

(iv) The product of two cyclic codes can give rise to a cyclic product code (Burton and Weldon 1965) if the length n_1, n_2 of the constituent codes are relatively prime. The generator polynomial of the product code g(x) can be characterised in terms of the roots (α_1,α_2) of the generators of the constituent codes $g_1(x)$ and $g_2(x)$ (Lin 1970).

(v) Peterson (1972) gives a list of binary cyclic codes of odd length less than 65, in which n, k, d, and exponents of the roots of the generator polynomial are tabulated.

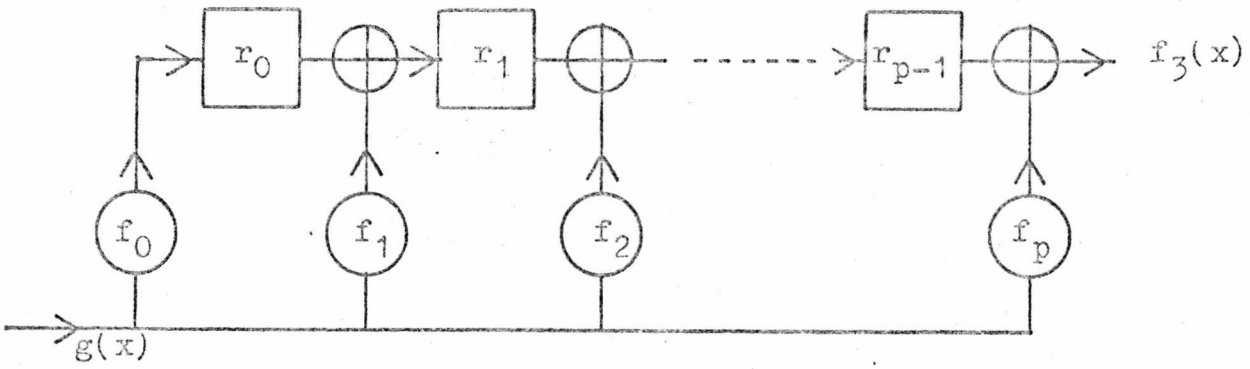
3.4.5 *Implementation, and polynomial arithmetic circuits*

Cyclic codes are implemented with feedback shift registers made up of mod-2 adders (exclusive-OR gates) and bistables. These circuits are used to perform multiplication and division of polynomials in GF(2), and these operations are briefly described in this section.

Multiplication of two polynomials $f_1(x)$ and $f_2(x)$ basically involves repeated addition of shifted versions of $f_1(x)$. The number of additions corresponds to the number of non-zero coefficients of $f_2(x)$, and the number of shifts required for each term in the addition is given by the powers of x in $f_2(x)$. Figure 3.4 below shows a circuit for multiplying g(x) by f(x).

Figure 3.4

Polynomial Multiplication Circuit



where:

r_i denotes a bistable (flip/flop), that is synchronously clocked

$\oplus$ denotes mod-2 addition (exclusive-OR gate)

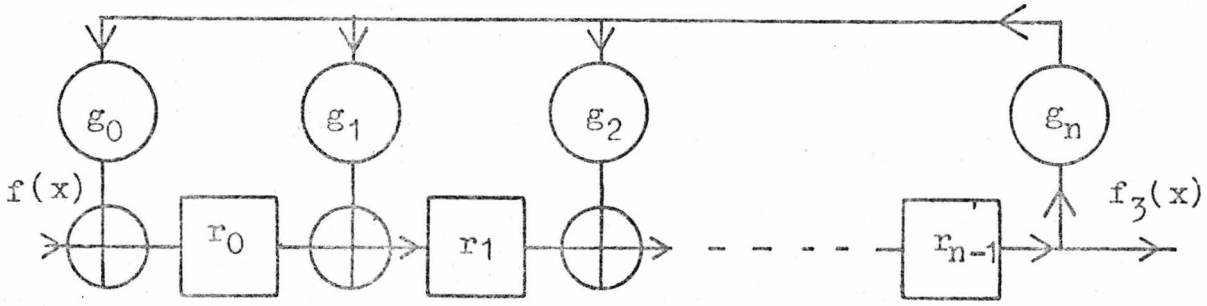
f_i denotes connection if $f_i=1$, and no connection if $f_i=0$; where f_i is the coefficient of x^i in f(x).

The circuit processes bits from left to right (high order bits first) and produces an output bit (of $f_3(x)$) at every shift. A total of n+p shifts are required: where n is the degree of g(x), and p is the degree of f(x).

Division of polynomials takes the form $f(x)/g(x)=f_3(x)+r(x)$, and corresponds to repeated subtraction of g(x) from the dividend. Such a division circuit is shown in figure 3.5 below.

Figure 3.5

Polynomial Division Circuit



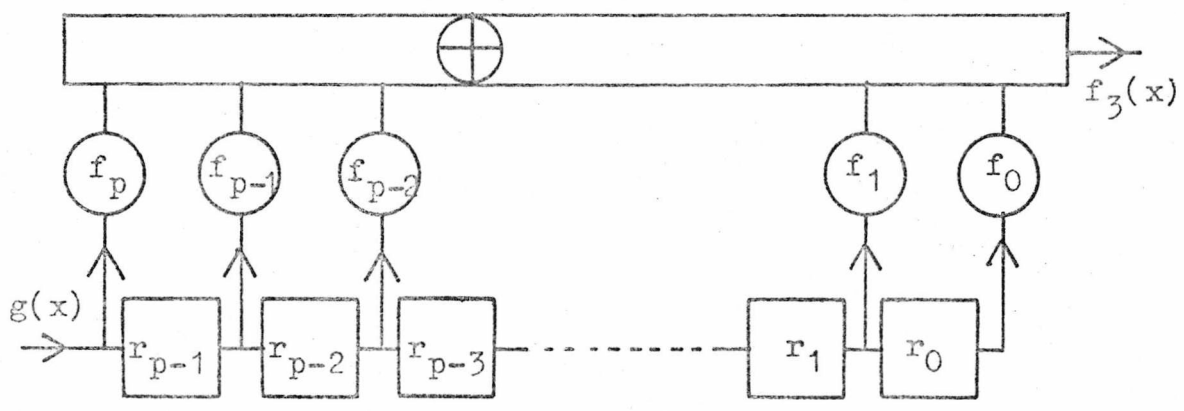
The register is initially cleared and f(x) is shifted in. After p shifts the entire quotient, $f_3(x)$, has appeared at the output, and the remainder, r(x), is in the register.

Multiplication and division may be simultaneously accomplished in one circuit by suitably combining the circuits of figures 3.4 and 3.5.

The circuits so far described have a serious disadvantage as far as implementation by using modern integrated circuits is concerned. The disadvantage is that exclusive-OR gates are in series with the shift register stages; whilst integrated circuit shift registers, in order to achieve high packing density, do not generally have the facility to break links between successive stages. Rather, it is only possible to tap the output of each stage. A multiplication circuit that multiplies g(x) by the fixed polynomial f(x) in n+p shifts, and overcomes this problem is shown by figure 3.6 below.

Figure 3.6

Alternative Multiplication Circuit



where ⊕ denotes a multi-input exclusive-OR gate.

Division circuits of this type (only one multi-input gate) are also possible: a circuit of the figure 3.5 type is first designed; and then a linear transformation process (Peterson 1972) is applied to convert the circuit into the figure 3.6 type.

3.4.6 Encoding cyclic codes

Given a k digit message sequence of the form $m(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \ldots + m_o$, the encoding operation involves:

(i) pre-multiplying m(x) by x^{n-k}. This shifts the message sequence n-k places towards the high order end of the word, as required for a systematic code.

(ii) dividing $x^{n-k}m(x)$ by the generator polynomial g(x), in order to calculate the parity digits.

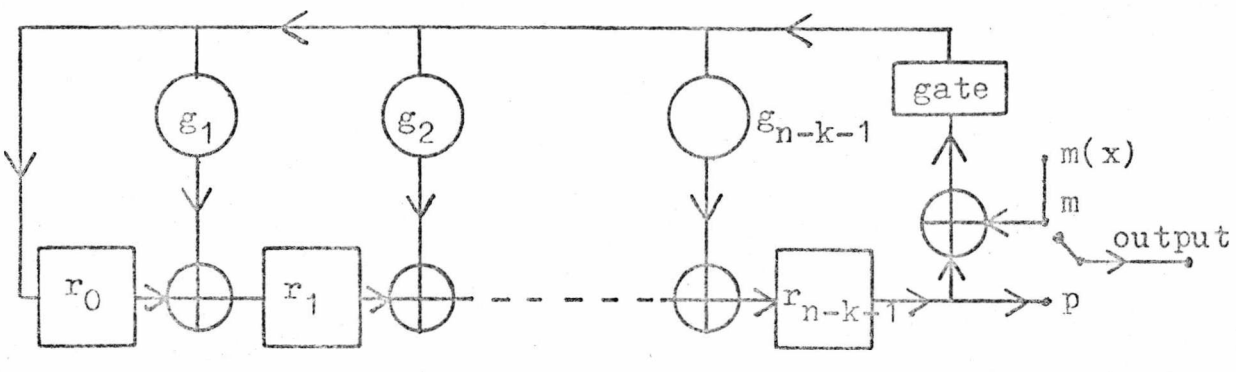
(iii) adding the remainder (the parity digits) of operation (ii) to the results of (i).

The circuits that perform this are based on those of section 3.4.5, and can either have n-k or k stages; the shortest usually being chosen.

An (n-k) stage encoder for a cyclic (n,k) code is shown in figure 3.7 below, and is based on the division circuit of figure 3.5.

Figure 3.7

n-k Stage Encoder



In this case the feedback connections are determined by the coefficients of g(x), and the position of the input is such that m(x) is multiplied by x^{n-k} prior to division by g(x). The circuit operates as follows.

(i) With the gate turned on, the k message digits of m(x) are shifted into the register and simultaneously into the channel via the switch in position m. Because of the input position, $x^{n-k}m(x)$ is actually being loaded into the register.

(ii) After k shifts the division is complete and the n-k digits in the register are the remainder of the division, and hence the parity check digits.

(iii) The feedback is disconnected by turning off the gate, the switch is moved to position p, and the parity digits are shifted out to the channel in (n-k) shifts.

The complete operation takes n shifts to produce a code word and therefore encoding successive code words can take place serially and without interruption. It must be noticed however that data is extracted from the source on a start stop basis; if this is not possible as in the case of a source producing information continuously, an input buffer is required. If (n-k) is large, but still much smaller than k, it may be necessary to transform this circuit into one that has uninterrupted register stages and a single multi-input exclusive-OR gate.

A k-stage encoder can be evolved based on computations involving h(x), the parity check polynomial, rather than g(x).

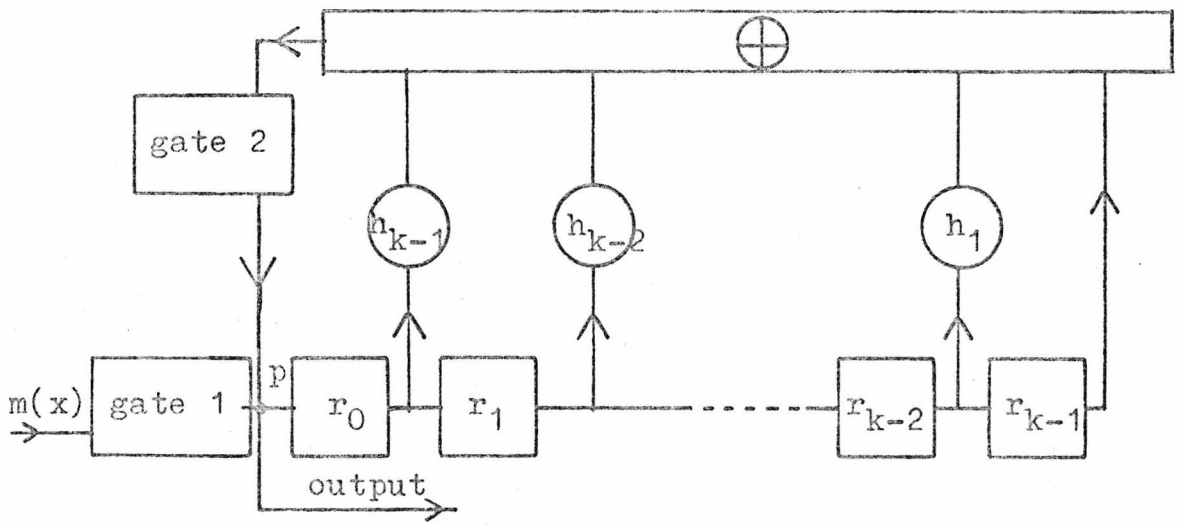
A recurrence relation involving coefficients (h_i) of h(x), and bit positions in the code word (u_i), is given by:

$$u_{n-k-j} = \sum_{i=0}^{k-1} h_i u_{n-i-j} \qquad 3.34$$

(Peterson 1972), and is in fact a rule for calculating the n-k parity checks (u_1 to u_{n-k}) from the k information digits, and the previously calculated checks. Figure 3.8 below shows a k-stage encoder based on equation 3.34

Figure 3.8

k-Stage Encoder



The encoder operates as follows:

(i) Initially gate 1 is on and gate 2 is off. The k information digits are loaded into the register and simultaneously sent to the channel in k shifts.

(ii) Gate 1 is turned off, gate 2 is turned on and the first parity check appears at p.

(iii) The register is shifted once and the first parity check is shifted into the channel and also into the leftmost register stage. The second parity digit is then formed and appears at p.

(iv) The operation continues in a similar manner, and encoding is completed in a total of n shifts.

The encoding circuits described above are simple, and therefore account for the attractiveness of binary cyclic codes.

3.4.7 *Syndrome calculation and error detection*

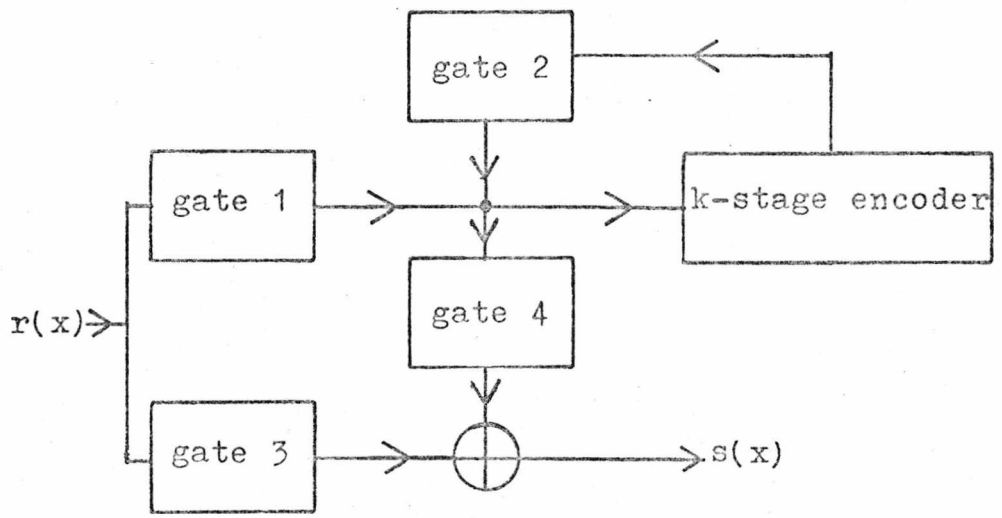
The syndrome s(x) of a cyclic code is the remainder resulting from dividing a received vector r(x) by the generator polynomial g(x), that is,

$$r(x)=p(x)g(x)+s(x) \qquad 3.35$$

The syndrome can be obtained by recalculating the parity checks using the message section of the received vector, and adding the recalculated checks to the received checks. This can be accomplished by using the encoding circuits of section 3.4.6. Figure 3.9 below shows a syndrome calculation circuit based on the k-stage encoder described earlier.

Figure 3.9

Syndrome Calculator



The circuit operates as follows:

(i) Initially gate 1 is on, gates 2,3,4 are off, and the received message digits are shifted into the register.

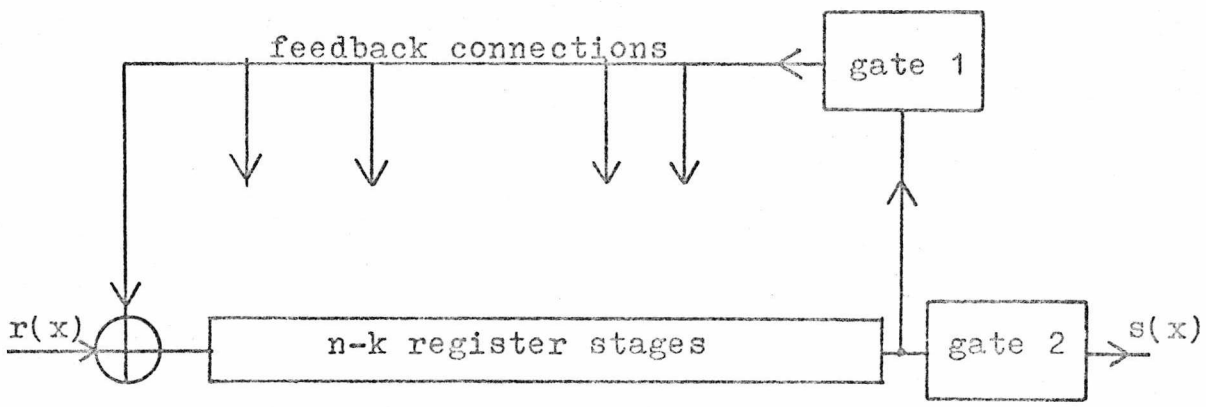
(ii) After k shifts the first parity digit is ready and gates 2,3,4 are turned on, gate 1 is turned off.

(iii) This causes the calculated checks to be added to the received checks and the syndrome is read out in (n-k) shifts.

A circuit based on the (n-k) stage encoder is shown in figure 3.10 below and is generally called a syndrome generator.

Figure 3.10

Syndrome Generator



This circuit operates as follows:

(i) Initially, the register is cleared, gate 1 is on and gate 2 is off.

(ii) After k shifts the register stages contain the syndrome, which can be output serially, if desired, by turning gate 1 off, gate 2 on and shifting (n-k) times.

In addition to being uniquely dependent on the added error polynomial e(x), the syndrome of a cyclic code has a further interesting property. That is, the syndrome of an i place cyclic shift of a given n-tuple is simply an i place cyclic shift of the syndrome in its syndrome generator, with feedback operating.

In order to detect errors it is only necessary to test for a non-zero syndrome and use this to set a bistable indicating a word in error. Because of this ease of syndrome calculation cyclic codes are ideally suited to error detection. A cyclic code with distance d will detect all combinations of d-1 or fewer errors, as will any linear code, but due to the cyclic property it is also possible to place the following bounds on the burst detecting ability of these codes.

(i) Because every code vector is a multiple of g(x), which is a burst of n-k+1 digits, an (n,k) cyclic code can detect all bursts of length n-k or less.

(ii) A fraction of bursts of length b>n-k can also be detected. If b=n-k+1 then the burst has the same length as the generator, and is undetectable if it has the same pattern as the generator or a cyclic shift of the generator. Remembering that there are n starting positions for a burst and there are 2^{n-k-1} burst patterns of length n-k+1, the fraction of undetected bursts is $n/n(2^{n-k-1})=2^{-(n-k-1)}$.

(iii) If $b>n-k+1$ then a burst pattern is undetectable if it has $g(x)$ as a factor, that is, $e(x)=g(x)q(x)$. As $g(x)$ has degree $n-k$, and the burst $e(x)$ has degree $b-1$, $q(x)$ has degree $b-1-(n-k)$, and there are $2^{b-1-(n-k)-1}$ polynomials of this degree. There are 2^{b-2} bursts of length b and the proportion of undetected bursts of length $b>n-k+1$ is therefore $2^{b-1-(n-k)-1}/2^{b-2}=2^{-(n-k)}$.

The results of (i) to (iii) above indicate that the probability of erroneous decoding for a cyclic error detecting code is determined by the number of parity check digits, in particular, the probability of undetected error is $\propto 2^{-(n-k)}$ regardless of how long the code is or how noisy the channel is. If $(n-k)=30$ parity checks are used the probability of undetected error is approximately 10^{-9} which should be sufficient for most applications.

3.4.8 Error correction

The decoding process consists of three basic steps.

(i) calculate the syndrome of the received n-tuple

(ii) determine the coset leader (correctable error pattern) that corresponds to the syndrome

(iii) add the received n-tuple to the coset leader, thereby correcting the errors.

It has been shown that steps (i) and (iii) are easy for cyclic codes; step (ii) is generally much more complex. A general decoder for cyclic codes was first described by Meggitt (1960, 1961) and is suitable for decoding burst

correcting and short random error correcting codes. A Meggitt decoder is schematically shown in figure 3.11 below and consists of three major parts.

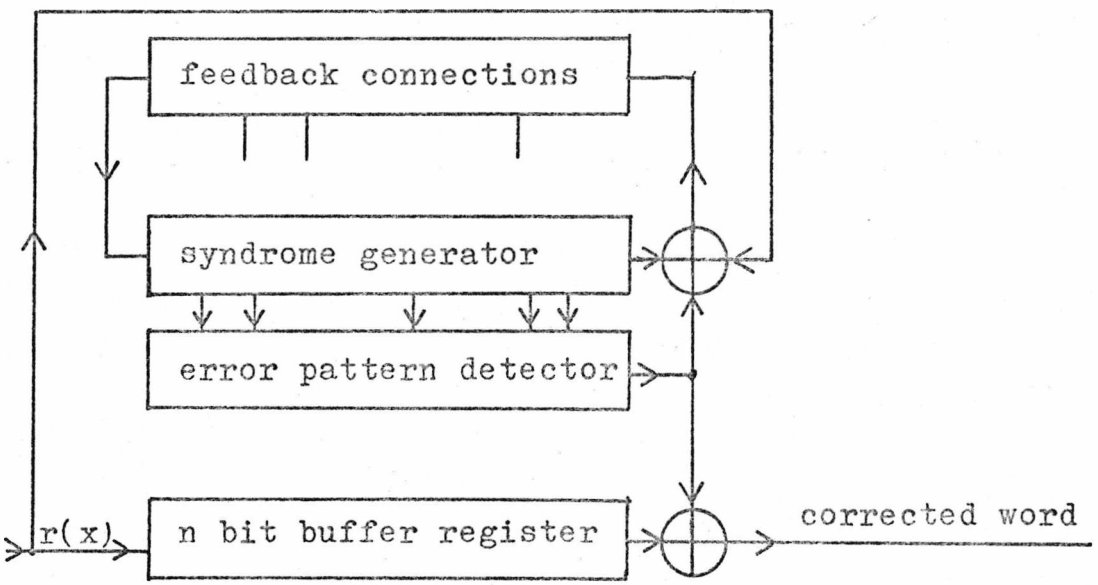
(i) a syndrome generator of the type described in section 3.4.7

(ii) an error pattern detector which is a combinational logic circuit

(iii) an n bit buffer register

Figure 3.11

Meggitt Decoder



The Meggitt decoder uses the principle that: because of the cyclic property of the syndrome, mentioned in section 3.4.7, the determination of the error pattern from the syndrome is considerably simplified. This simplification means that it is possible to have only one syndrome-error pattern correspondence for a particular pattern *and* every cyclic shift of that pattern, rather than n such correspondences. The Meggitt decoder operates as follows:

(i) It is assumed that the most likely error patterns are chosen as coset leaders, and every cyclic shift of a coset leader is also a coset leader. The pattern obtained by changing a single one to a zero is called a descendant of an n-tuple, and the decoder assumes that every descendant of a coset leader is also a coset leader. The combinational logic circuit is designed to output a 1 if there is an error in the highest order bit of a pattern, that is, the bit about to be read out of the buffer.

(ii) After n shifts the complete word is in the buffer and the syndrome is in the generator. The combinational logic circuit outputs a 1 if the bit about to be read out is in error.

(iii) On the next shift the first bit is corrected if it is in error. The syndrome generator contains the syndrome of the shifted word, because of the cyclic property, and the effect of the error (if present) is also removed from the syndrome by feeding back the detector output.

(iv) Operation continues in a similar manner until the entire word is shifted out of the buffer. The errors will have been corrected if they correspond to a pattern built into the detector, and the syndrome generator will contain all zeros. If the syndrome generator contents are non-zero at the end of the process, an uncorrectable error pattern has been detected.

A total of 2n shifts are required to decode each word and therefore the decoder cannot operate at 'line speed', rather alternate words are decoded. This can be overcome by either: providing duplicate decoders; or, shifting the received word into the buffer at line speed and then performing the n shifts required to read out the word in one digit time.

The Meggitt decoder can in principle be used to decode any cyclic code; whether or not the decoder is practical for a particular code, however, generally depends on the complexity of the combinational logic circuit required. The Hamming codes for example can be decoded very easily with a Meggitt decoder because the detector circuit is simply an (n-k) input AND gate. The BCH codes on the other hand require a very complex detector circuit.

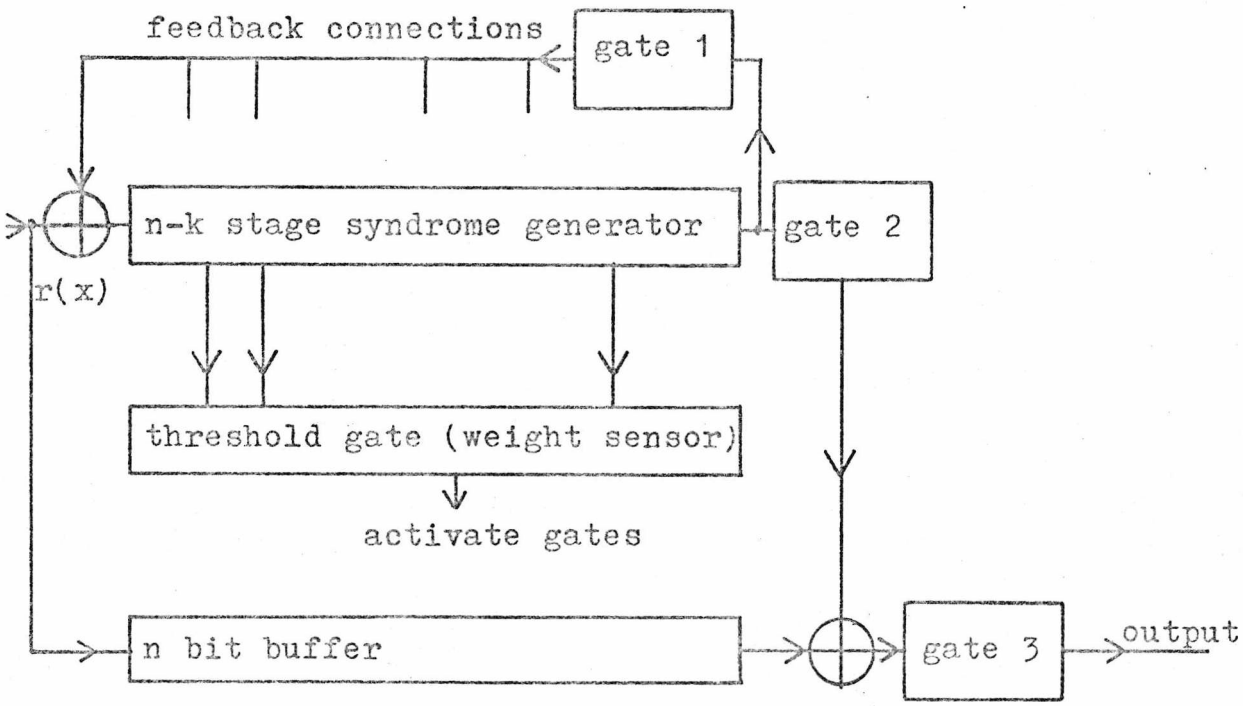
3.4.9 Error trapping decoding

Error trapping decoding (Rudolph and Mitchell 1964) is a variation of Meggitt decoding that relies on it being possible to confine the errors in a word to n-k consecutive positions. The method is therefore most suitable for single-error-correcting and burst correcting codes. Several modifications of the method have been proposed by Kasami (1964) and MacWilliams (1964) in an effort to extend its application to multiple error correcting codes; in general however the method sacrifices much correcting ability if applied to long and high rate codes with large distance.

It can be seen that if t errors are confined to the n-k parity positions, then the error pattern equals the syndrome, and the message section of the word does not require correction. Error trapping uses the fact that if the errors are confined to *any* n-k consecutive positions, including the end around case, then by cyclicly shifting the received vector, the errors can be confined (or trapped) in the n-k parity check positions of the shifted vector, and therefore corrected. An error trapping decoder is shown in figure 3.12 below.

Figure 3.12

Error-Trapping Decoder



The decoder operates as follows:

(i) Initially gate 1 is on, gates 2 and 3 are off. After n shifts the received word is in the buffer and the syndrome is in the generator.

(ii) The weight of the syndrome is tested by the (n-k) input

threshold gate. If t or fewer inputs are 1 the output is 1, otherwise it is zero.

(iii) If the syndrome weight is t or less the errors are in the parity section and the information digits are correct, and can be shifted out of the buffer. If the weight of the syndrome is greater than t, the generator is shifted once.

(iva) If the weight of the new syndrome is t or less the errors are in the end around positions x^{n-1}, ... , x^{n-k-2}, ... , x_1, x_0, and the left most digit of the syndrome matches the error in the right most postion of the buffer. The syndrome is therefore shifted to the right, with gate 2 off, in order to place the error corresponding to x^{n-1} at the right of the generator; this digit then corrects the first message digit as it comes out of the buffer.

(ivb) If the weight of the new syndrome is greater than t, the generator is again shifted.

(v) Step (ivb) repeats until the weight goes down to t or less, and then the generator is shifted to match the right most positions in the buffer, thereby correcting the errors as the message digits emerge.

(vi) If the weight of the syndrome does not go down to t or less in (n-k) shifts, gate 3 is turned on and the information digits are read out one at a time while the generator is cycled with gate 1 on. As soon as the weight of the generator contents goes down to t or less, the syndrome in the generator exactly matches the

the errors in the next (n-k) digits to come out of the buffer. These errors are then corrected as digits emerge from the buffer.

(vii) If the weight of the generator contents never goes down to t or less by the time the k message digits have been read out of the buffer, the error pattern is either uncorrectable or not confined to (n-k) consecutive positions.

Error trapping decoding is advantageous in that the complexity of the pattern detector is reduced. In particular for a b-burst correcting code, the pattern detector is an (n-k-b+1)-input AND gate. Decoders for the Hamming codes are simple; but generalisations to multiple error correcting codes have to make use of the proposed modifications, with a consequent increase in complexity.

3.4.10 Shortened cyclic codes

Given an (n,k) cyclic code, it is always possible to form an (n-i,k-i) code by making all the i leading message digits zero and omitting them from the code word. Such a code has minimum distance at least equal to that of the original code and has the advantage that although the code is not cyclic, encoding and decoding can be implemented by using cyclic code circuits. Shortened cyclic codes are in fact the same as pseudo-cyclic codes, that is, they are ideals in the algebra of polynomials modulo f(x), where f(x) is some polynomial other than x^n-1.

3.5 *Bose-Chaudhuri-Hocquenghem (BCH) codes*

BCH cyclic codes were discovered by Hocquenghem (1959) and independently by Bose and Chaudhuri (1960), and, as a class are the most powerful random-error-correcting codes known. The first efficient decoding algorithm for binary BCH codes was devised by Peterson (1960) and has since been refined by Gorenstein and Zierler (1960), Chien (1964), Forney (1965), and Massey (1965, 1969).

3.5.1 *The BCH bound*

The BCH lower bound on distance applies to all cydic codes; and the BCH codes are a class of codes whose generator polynomials are chosen in such a way as to maximise the distance guaranteed by the bound. The BCH bound states that given a generator polynomial with roots α^{e_1}, α^{e_2}, ... , $\alpha^{e_{n-k}}$, the distance of the code is greater than the largest number of consecutive integers modulo n in the set (e_1, e_2, ... , e_{n-k}). The bound is proved in Peterson (1972) and for BCH codes is particularly tight.

3.5.2 *Description of BCH codes*

Consider an element α in GF(2^m). Then for any specified m_0 and d_0, a BCH code with distance at least d_0 is generated by the smallest degree polynomial (least parity checks) g(x) that has α^{m_0}, α^{m_0+1}, ... , $\alpha^{m_0+d_0-2}$ as roots. The length of the codes is the least common multiple (LCM) of the orders of the roots, and d_0 is called the *designed distance*. The

most important sub-class of BCH codes are obtained by letting α be a primitive element of $GF(2^m)$, $m_0=1$, and $d_0=2t_0+1$. These codes are called *narrow-sense* BCH codes and f(x) is then a code vector iff

$$\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t_0} \qquad 3.36$$

are roots of g(x). This is equivalent to saying that g(x) is the least common multiple of the minimum polynomials of the roots, that is,

$$g(x)=LCM\left[m_1(x), m_2(x), \ldots, m_{2t}(x)\right] \qquad 3.37$$

From section 3.4.2 it can be seen, however, that every even power of α has the same minimum function as some previous odd power of α. For example α^1, α^2, α^4, α^8 all have $m_1(x)$ as their minimum polynomial; α^3, α^6 have $m_3(x)$, and so on. An equivalent statement to equation 3.37 is therefore that g(x) is given by:

$$g(x)=LCM\left[m_1(x), m_3(x), m_5(x), \ldots, m_{2t_0-1}(x)\right] \qquad 3.38$$

and since the degree of each minimum polynomial is m or less, the degree of g(x) is at most mt. The codes can therefore be defined by saying that for any positive integers m and $t_0<n/2$, there exists a narrow sense BCH code of length $n=2^m-1$ which corrects all combinations of t_0 or fewer errors and has at most mt_0 parity checks.

The actual value of (n-k) can be determined by an algebraic procedure due to Berlekamp (1967) and for small t_0, (n-k) is generally exactly equal to mt. Peterson (1972) shows that for many of the codes the designed distance equals the actual distance and in addition tabulates all narrow sense (primitive) BCH codes of length up to 1023.

The codes of length 15 or less are perfect and therefore optimum for the BSC. All the double-error-correcting codes of length up to 1023 are quasi-perfect and hence also optimum. If the rate k/n is kept constant while n is increased, the lower bound on t/n approaches zero. The BCH codes are therefore very weak for large n but do, however, still lie approximately on the VGS bound for n=1023.

3.5.3 Decoding BCH codes

Encoding BCH codes is done by using the cyclic code circuits of section 3.4.6. A decoding algorithm for multiple error-correcting BCH codes that is more efficient than Meggitt decoding was originated by Peterson (1960) and will now be briefly described.

Consider a transmitted vector u(x) that is corrupted by noise e(x) to form a received vector r(x), that is,

$$r(x)=u(x)+e(x) \qquad 3.39$$

If α^i is substituted into r(x) then $u(\alpha^i)=0$ because u(x) is a code word and the result is

$$s_i=r(\alpha^i)=e(\alpha^i) \qquad 3.40$$

$$s_i=e_{n-1}(\alpha^i)^{n-1}+e_{n-2}(\alpha^i)^{n-2}+ \ldots +e_1(\alpha^i)+e_0 \qquad 3.41$$

The s_i are called *partial syndromes* and the complete syndrome for a BCH code is defined as a set of partial syndromes,

$$s=s_1,s_2, \ldots , s_{2t} \qquad 3.42$$

one for each root of the generator polynomial. The partial syndrome s_i is in fact the remainder after dividing r(x) by $m_i(x)$, the minimum polynomial of α_i, and therefore amounts

to a parity check calculation. If e(x) is an error pattern with v errors then

$$e(x)=x^{j_1}+x^{j_2}+ \ldots +x^{j_v} \qquad 3.43$$

where x^{j_1} is the lowest order error. And from equations 3.41 and 3.42

$$\begin{aligned} s_1&=\alpha^{j_1}+\alpha^{j_2}+ \ldots +\alpha^{j_v} \\ s_2&=(\alpha^{j_1})^2+(\alpha^{j_2})^2+ \ldots +(\alpha^{j_v})^2 \\ &\vdots \\ s_{2t}&=(\alpha^{j_1})^{2t}+(\alpha^{j_2})^{2t}+ \ldots +(\alpha^{j_v})^{2t} \end{aligned} \qquad 3.44$$

Any error correction algorithm is thus a method of solving equations (3.44) for α^{j_1}, α^{j_2}, ... , α^{j_v}. The powers j_1, j_2, ... , j_v then give the locations of errors in e(x) as in equation 3.43.

Algorithms for determining the error location numbers ($b_i=\alpha^{j_i}$, for $1\leq i\leq v$) are dealt with extensively in the literature (Peterson 1972, Lin 1970, Berlekamp 1968, Massey 1969, Chien 1964) and will not be dealt with further. Implementation of these decoding algorithms can be achieved by using digital hardware, by programming a general-purpose computer (software implementation), or by a hybrid combination of the two. In addition the correction of erasures and errors can be achieved with little extra complexity (Peterson 1972).

3.5.4 *Non-binary BCH codes and Reed-Solomon codes*

Non-binary BCH codes of length $n=q^s-1$. where q is a power of a prime, are called q-ary codes and have code symbols from a q symbol alphabet. Reed-Solomon (1960) codes are a subclass of these codes and have s=1. A t-error-correcting Reed-Solomon code has 2t parity symbols and a

minimum distance of (2t+1), these codes are therefore maximum distance separable and hence their weight structures are known exactly (Peterson 1972). If $q=2^m$ then each q-ary symbol can be expressed as an m-tuple over GF(2) and a t-error-correcting (n,n-2t) Reed-Solomon code over GF(2^m) can be regarded as a (mn,m(n-2t)) code over GF(2) capable of correcting any error pattern whose non-zero digits are confined to t m-symbol blocks. Some of the most powerful burst-and-random correcting codes can be constructed using this technique.

Decoding a Reed-Solomon code is effected in a similar way to a binary BCH code, that is, by determining error location numbers, but in addition an error value number corresponding to each error location must also be calculated (Zierler 1969).

3.6 Majority logic decoding for cyclic codes

Majority logic decoding is a decoding method that is suitable for certain classes of block code and most codes in this class have been found to be cyclic. Although these codes are generally not as powerful as BCH codes their ease of decoding makes them attractive. The first majority logic procedure was that of Reed's, for decoding Muller's codes (section 3.3), Massey (1963) has unified many majority logic decoding algorithms.

3.6.1 One-step majority logic decoding

Each digit in the syndrome of a received vector r(x) can be considered as a parity check sum of various digits in

the error vector e(x) that was added to the transmitted word. The positions covered by the ith sum correspond to the non-zero positions in the ith column of H^T. Majority logic decoding is based on the concept of forming check sums that are *orthogonal* on a set of positions. One-step majority logic decoding restricts this to the formation of check sums that are orthogonal on one position in the error vector.

A set of check sums $A_1, A_2, \ldots, A_j$ are said to be orthogonal on a particular bit position if:

(i) that bit is involved in every sum

(ii) no other bit is involved in more than one sum.

For example the set of sums:

$$\begin{aligned} A_1 &= e_6 + e_3 + e_0 \\ A_2 &= e_5 + e_2 + e_0 \\ A_3 &= e_4 + e_1 + e_0 \end{aligned} \qquad 3.45$$

are orthogonal on the error position e_0. An error correction procedure based on the majority value of these sums then becomes possible. Suppose firstly that it is possible to form j check sums orthogonal on a particular position, say e_0, and assume that the error vector contains j/2 or fewer errors. Then if $e_0=0$ the j/2 errors distribute amongst at the most j/2 check sums. Therefore in the worst case when j/2 errors affect j/2 sums, half the sums are zero and half are one. If, however, $e_0=1$, the (j/2)-1 other errors can distribute amongst at most (j/2)-1 sums, and therefore more than half the sums equal 1. The value of the error digit e_0

is therefore 1 if a clear majority of the sums orthogonal on e_0 are 1. If it is possible to form j check sums orthogonal on e_0, then because of the cyclic symmetry of cyclic codes it is possible to form j check sums orthogonal on any digit in the word by cyclicly shifting the error vector. In addition, orthogonal check sums provide a lower bound on minimum distance which states that if j=d-1 check sums orthogonal on any digit in a cyclic code can be formed, then the code has minimum distance d and is called completely orthogonisable in one step. For a completely one-step orthogonisable code j/2 equals the error correcting ability (d-1)/2 of the code. If j/2 is much less than the error correcting ability of the code then the use of one-step decoding will sacrifice much of the power of the code. An upper bound on the number of errors that can be corrected by one-step majority logic decoding (Peterson 1972) is given by

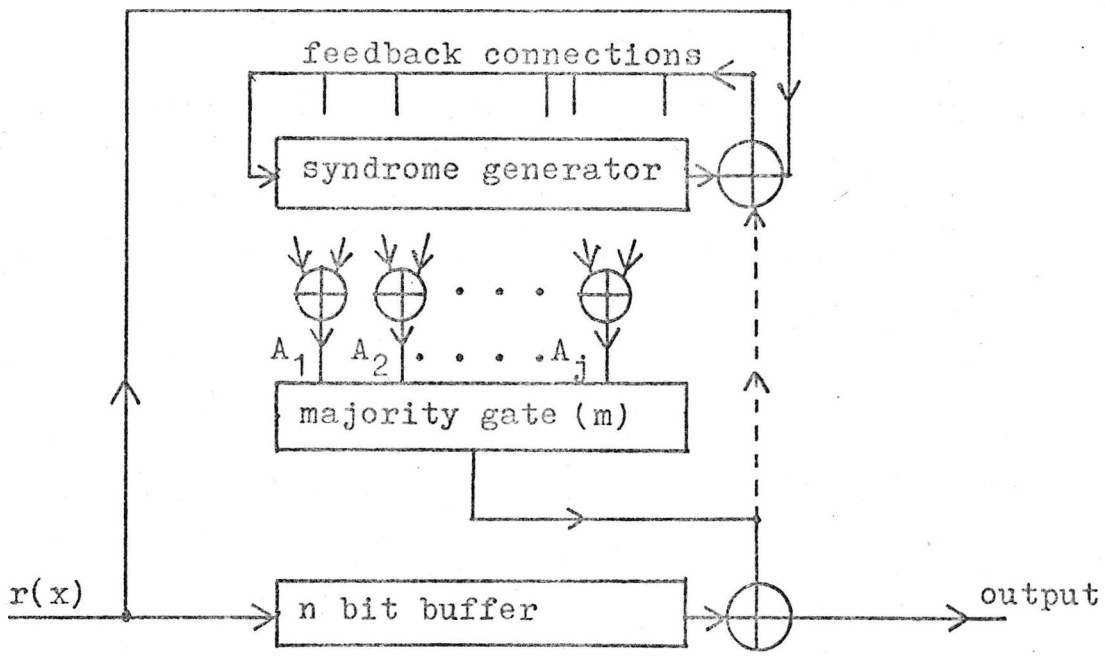
$$t_1 \leq \frac{n-1}{2(d'-1)} \qquad 3.46$$

where d' is the distance of the dual code of the (n,k) code being considered. Equation 3.46 severely restricts the usefulness of one-step majority logic decoding for many classes of code. The only known BCH code that is completely orthogonisable in one step is the (15,7) code and therefore using this technique to decode BCH codes usually sacrifices much of their power. Two small classes of codes that are completely orthogonisable in one step are the maximum-length-sequence codes mentioned in section 3.4 and the difference-set codes discovered by Weldon (1966).

There are two main types of majority logic decoder, (Massey 1963), based on the two main ways of encoding cyclic codes. A Type I decoder is based on the (n-k) stage syndrome generator circuit and is shown in figure 3.13 below.

Figure 3.13

Type I Decoder



The decoder operates as follows:

(i) The syndrome is calculated as usual

(ii) The j check sums orthogonal on the first digit (e_{n-1}) are formed by taking linear combinations of syndrome digits that give the required sums of error digits.

(iii) If the output of the majority gate is 1 the first digit is erroneous and is corrected as it is read out of the buffer. The syndrome generator is shifted simultaneously and the new syndrome corresponds to the shifted received vector. The previously calculated error

digit is then fed back to alter the syndrome so that it corresponds to the altered shifted received vector. (See section 3.6.4.)

(iv) Subsequent digits are similarly decoded in a total of n shifts and decoding is correct if fewer than j/2 errors occurred in the received vector.

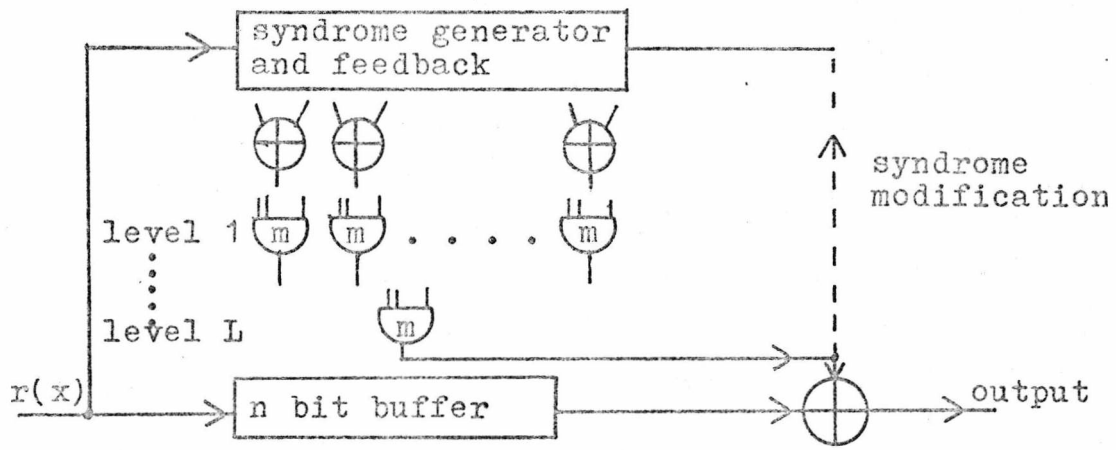
3.6.2 *L-step majority logic decoding*

The concept of one-step majority logic decoding can be generalised to L-step decoding in which a set of check sums are orthogonal on a *set* (E) of error digits rather than just one error digit. In a similar manner to one-step decoding, the sum of digits in E, that is, $(e_{i_1} + e_{i_2} + \ldots + e_{i_e})$ can be correctly determined from the j check sums provided j/2 or fewer errors occur in the received word. This sum can be considered as an additional check sum and used in the next level of decoding. The results of the j original set of sums are used to estimate the value of a second set E' which contains fewer error digits than E, and j sums or more are required to do this. The process of estimating sums from sums of a larger size is known as *orthogonalization* and continues until a set of j or more sums orthogonal on a single error digit is obtained. The value of this error digit can then be estimated by using a single majority gate and because of the cylic symmetry of the code, so can every other error digit. A code is called L-step decodable if L steps of orthogonalization are required before a decoding decision on a particular error digit can be made, and a

code is completely L-step orthogonisable if j=d-1. The decoder based on this process uses majority gates to estimate sums at each orthogonalisation, and therefore L levels of majority gates are required. An L-step type I majority logic decoder is shown in figure 3.14 below.

Figure 3.14

L-Step Type I Majority Logic Decoder



Such a decoder requires L levels of majority gates and in general the ith level requires $(j)^{L-i}$ gates. The complexity of the gating is therefore an exponential function of L and if L is large the decoder may be impractical.

L-step decoding is most efficient when used with a completely L-step orthogonalisable code, and is roughly twice as powerful as one step decoding (equation 3.46).

3.6.3 Majority logic decodable codes

Massey (1963) has shown that the $(2^m, 2^m-m-1)$ Hamming codes are completely orthogonalisable in m-1 steps.

The BCH (15,7) and (31,16) are 2 and 3 step orthogonalisable respectively and so is the subclass of $(2^m-1, m+1)$ codes which

are $(2^{m-2}-1)$ error correcting, and orthogonalisable in two steps for m>2. Other classes of completely orthogonalisable codes are based on the study of finite geometries and were first studied by Rudolph (1967). The construction and rules for orthogonalisation are too involved to be dealt with here, but are extensively dealt with in the literature. (Goethals 1968, Kasami 1968, Peterson 1972, Weldon 1968.) The two main classes of these codes are projective geometry codes and Euclidean geometry codes, a particular code being denoted either $PG(m,2^s)$ or $EG(m,2^s)$. The class of projective geometry codes contains a subclass of codes which itself contains the subclass of difference-set cyclic codes mentioned in section 3.6.1. Also contained in this class are maximum-length-sequence codes, and the equivalents of the even-distance duals of Reed-Muller codes with one digit omitted. Euclidean geometry codes with s=1 are equivalent to the odd-distance Reed-Muller codes with one digit omitted. Lists of some completely orthogonalisable codes are given in Peterson (1972), Lin (1970) and Lucky (1968).

For moderate n, geometry codes compete with BCH codes because of their ease of implementation, for large n, however, the number of majority gates required is excessive.

3.6.4 Modifications to the basic procedure

The decoding procedures for BCH codes are bounded-distance procedures, that is, no patterns of weight greater than t are corrected. Majority logic decoding, however, is capable of decoding many patterns of weight greater than t.

Two main modifications have been proposed in order to extend the usefulness of the procedure:

(i) the use of feedback. If t+1 errors occur and one of them is in the first digit then it is possible for this error to be corrected. In the case of one step decoding this implies that one of the check sums orthogonal on the first bit contains two additional errors. The effect of this error can be removed leaving t errors which can be corrected. The effect of this error on the syndrome is also removed via the dashed lines in figures 3.13 and 3.14.

(ii) Variable threshold decoding (Townsend 1967). This modification to one-step decoding involves replacing the majority gate by a threshold gate that outputs a 1 if T or more of its input are 1. Decoding commences with the threshold set at its highest value, d-1, and the decoder attempts to correct each digit of the received word. If this decoding is unsuccessful the threshold is lowered by one and another attempt is made. When a correction is made the syndrome is modified and the threshold is increased by one. Decoding continues in this manner until the threshold reaches (d+1)/2, when decoding stops. Many error patterns of weight greater than (d-1)/2 which are uncorrectable with basic one-step decoding can be corrected with this modification. The penalty paid, however, is a considerable increase in decoding time.

3.7 *Burst-error control codes*

The coding techniques described in previous sections have been biased toward the correction of random errors; most real channels, however, exhibit burst-error behaviour and codes with burst-correcting ability are in practice required. Codes for burst-error-correction have been constructed both analytically and by trial and error, and a simple decoding method has been devised. The design of a practical and efficient burst-error-correcting system is therefore much easier than the design of a random-error-correction system. The Reiger bound (equation 3.23) gives a measure of the efficiency (z) of an (n,k) code with burst-correcting ability b, that is,

$$z=\frac{2b}{n-k} \qquad 3.47$$

Codes which meet the Reiger bound have an efficiency of 1 and are optimum.

3.7.1 *Fire codes*

The first cyclic burst-correcting code was found by Abramson (1959); and Fire (1959) generalised this work to produce the first class of analytically constructed cyclic codes designed to correct single bursts.

A b-burst correcting Fire code is generated by:

$$g(x)=p(x)(x^{2b-1}-1) \qquad 3.48$$

where $p(x)$ is an irreducible polynomial of degree $c \geq b$, and $2b-1$ is not divisible by e, the exponent of $p(x)$. The length is $n=e(2b-1)$, and the resulting $(n,n-2b-c+1)$ code has burst correcting ability b. The efficiency of a Fire

code is

$$z = \frac{2b}{c+2b-1} \qquad 3.49$$

and if b=c then

$$z = \frac{2c}{3c-1} \qquad 3.50$$

Equation 3.50 shows that Fire codes are not very efficient, and in particular for large c, z is approximately 2/3.

3.7.2 Interleaving

Given an (n,k) code, it is possible to form a longer code with a large amount of burst correcting ability by interleaving digits of the original code. This can be accomplished as follows. Firstly, i code words are stacked vertically to form an i by n array. The array is then transmitted column by column, that is, the first digits of each code word are transmitted consecutively, followed by all the second digits and so on. At the receiver the array is reformed and the rows, which are code words in the original code, can then be decoded. It can be seen that a burst of i errors will affect no more than one digit in each original code word. Therefore in general, if the original code corrects all combinations of t or fewer errors, the code formed by interleaving to degree i will correct any combination of t bursts or less of length i or less. Similarly if the original code corrects a single burst of length b or less the interleaved code corrects any single burst of length bi or less.

If the original (n,k) code meets the Reiger bound then the interleaved code also has maximum possible burst correcting

ability, and it is therfore possible to construct codes of almost any length with optimum burst correcting ability. In addition, interleaving reduces the problem of finding long efficient burst correcting codes to one of finding short good codes, a considerably easier task.

Important simplifications occur when the original code is cyclic. In this case the interleaved code is also cyclic and if the generator polynomial of the original code is g(x) the generator of the interleaved code is $g(x^i)$. Similarly, interleaving a shortened (n,k) cyclic code produces a shortened (ni,ki) cyclic code, and a considerable number of good codes can be formed by this technique.

Finally, because none of the random-correcting ability of the original code is destroyed by the interleaving process, it is possible to derive powerful, long, burst-and-random correcting codes by interleaving *short*, powerful, random-correcting codes.

3.7.3 *Phased-burst-error-correcting codes*

Consider an (n,k) code whose length n is a multiple of an integer m, say n=rm. A code word therefore consists of r sub-blocks of length m. A burst of length b^1=im is called a *phased burst* if the errors are confined to i consecutive blocks. A code that can correct all phased-error-bursts confined to i or fewer sub-blocks is called an im-phased-burst-error-correcting (PBEC) code. A burst of length (i-1)m+1 affects no more than i sub-blocks and therefore an im PBEC code can be used as a (i-1)m+1 single-burst-error-correcting code.

Burton (1969) discovered a class of PBEC codes that are superior to Fire codes in most cases. An (em,(e-2)m) Burton code is generated by:

$$g(x)=p(x)(x^m+1) \qquad 3.51$$

where p(x) is an irreducible polynomial of degree m and exponent e. The code can correct all bursts of length m or less which are confined to the positions $x^{jb}, x^{jb+1}, \ldots, x^{jb+b-1}$, for $0 \leq j \leq e-1$.

The Burton codes can be sub-block interleaved to form a class of code that is asymptotically optimal with increasing interleaving degree.

3.7.4 *Computer generated codes*

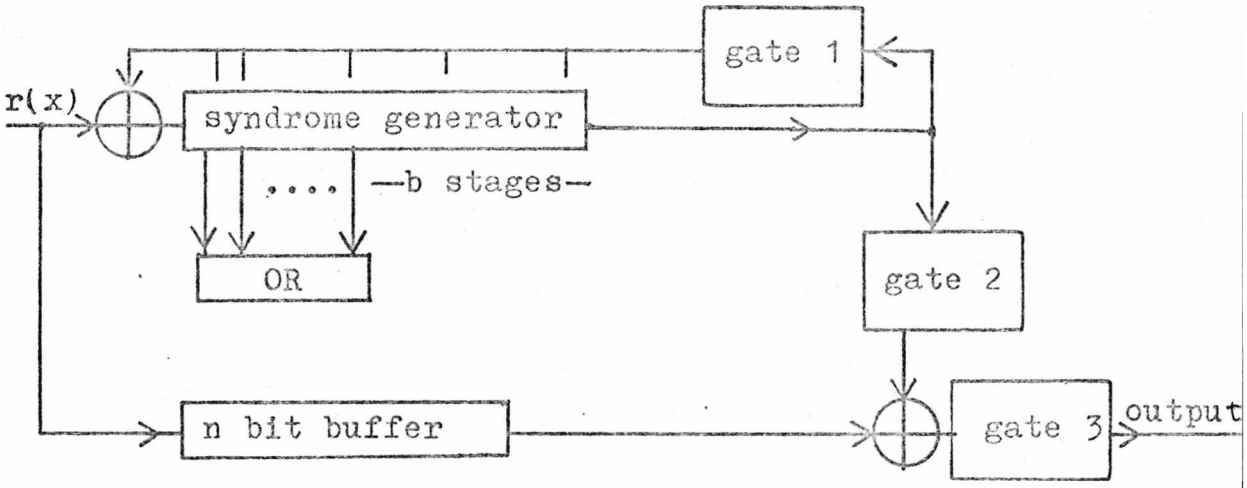
Lists of computer generated burst correcting cyclic codes appear in Lucky (1968), Kasami and Mabola (1964), Zetterburg (1962) and Peterson (1972). These codes are useful because it is possible to construct a wide range of codes by either interleaving or shortening these published codes.

3.7.5 *Decoding single-burst-error-correcting codes*

Decoding for these codes can be accomplished by using the error trapping method. Figure 3.15 below shows such a decoder. If the errors are confined to b positions then it is possible to confine them to the b high order positions of the syndrome. The low order n-k-b positions then all contain zeroes and this condition is detected by the OR gate.

Figure 3.15

Burst Correcting Decoder



By suitable gating and shifting (as in section 3.4.9) the errors can then be lined up to correct digits as they come out of the buffer register. If the n-k-b left most stages of the syndrome generator never contain zeroes by the time the k information digits have been read out of the buffer, then a burst of length longer than b or an uncorrectable burst has occurred.

The above method of decoding only corrects bursts of length b or less, of which there are $n2^{b-1}$. If n is at all large this will be but a small fraction of the 2^{n-k} correctable patterns (coset leaders). It is however possible, with a slight complexity increase, to correct all correctable bursts of n-k or less by using a modification due to Gallager (1968).

3.7.6 Burst-and-Random error correcting codes

Of the codes so far considered; product, interleaved, and concatonated codes have both burst and random-correcting

capability. For a code to be t random-correcting and b burst-correcting the cosets containing patterns of weight t or less must be disjoint from those containing the burst patterns. Several codes have this property and of these the binary codes derived from q-ary Reed-Solomon codes are the most powerful.

A Reed-Solomon code with symbols from $GF(2^m)$ has parameters

$$n=m(2^m-1)$$

$$n-k=2mt$$

and is capable of correcting any error pattern that affects t or fewer m-bit bytes. This t-byte correcting code is therefore an example of a multiple-phased-burst correcting code. The random-correcting capability of these codes is t, and the single burst correcting ability is $m(t-1)+1$. Reed-Solomon codes can also correct multiple bursts, and in general a t-byte correcting code is capable of correcting any combination of

$$i=\frac{t}{1+\frac{b+m-2}{m}}$$

bursts of length b.

Computer generated burst-and-random codes have been described by Hsu (1968), and seem to show that good random error correcting codes should be able to correct fairly long bursts.

Interleaving the codes mentioned in this section can again be used to extend the useful range of burst-and-random correcting codes, as well as providing multiple burst-correcting ability.

CHAPTER 4

Adaptive Communications and Time Varying Channels

This chapter introduces the concept of an adaptive communication system in a mainly qualitative way. Initially, the term *adaptive* is broadly used to mean the variation of some system parameter in an attempt to 'adapt' the system to currently prevailing conditions, so that a certain degree of 'reliable' communication is maintained as these conditions change. More specifically, we consider the adaption of a data transmission system to *match* changing channel conditions, in order to achieve lower average sink error rates with higher average data throughput than that obtainable with a *fixed* system.

The choice of a suitable parameter to vary is a system design choice and in this chapter the variation of several individual system parameters is considered. The following chapters, however, direct attention towards adaptive error control coding, and in particular to variable redundancy coding.

In order to vary a system parameter in sympathy with changing channel conditions, some controlling device is necessary. This device must firstly assess the *state* of the channel, and secondly, make a decision based on some criterion (or Algorithm) as to how the chosen system parameter must be varied in order to counteract the effects of the original channel change. Various methods of performing this function are considered, together with the

problem of communicating the decision (or control signal) to the parameter which is to be varied.

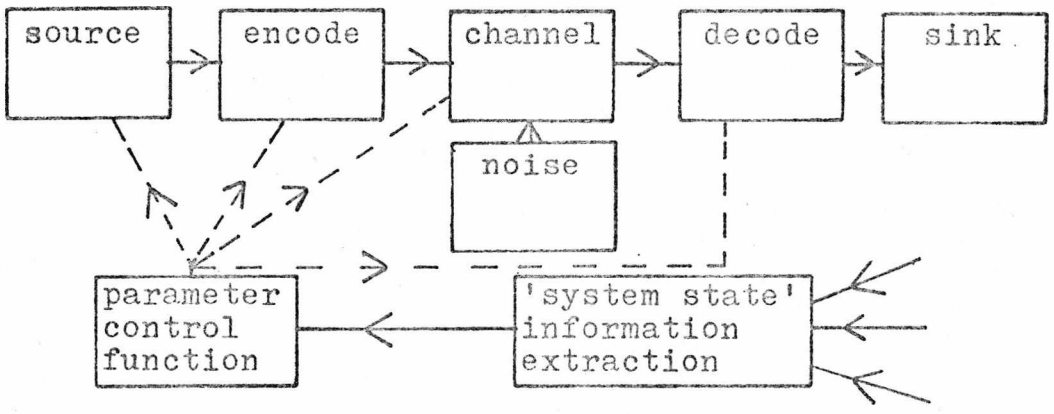
Adaptive systems result from attempts to cope with the non-stationary nature of real channels, and therefore the choice of a particular channel-adaptive system is influenced by the way in which these channels vary. For this reason, and in order to more fully appreciate the effects with which an adaptive system has to cope, several time-varying fading channels are described. Particular attention is given to the HF radio channel, which was the channel used for field tests of the experimental variable redundancy system.

4.1 Adaptive data transmission

A general adaptive data transmission system is shown in figure 4.1 below.

Figure 4.1

The General Adaptive System



The model of figure 4.1 is very general as it includes modulation, demodulation, and error control under the terms *encode* and *decode*. The source, channel, encoder, and decoder can be considered as parameters of the system and therefore available for adaptive control. The channel is assumed to be controlled by nature, and therefore unalterable, but the *choice* of channel is available for adaptive control (e.g. diversity systems).

The control of the adaptive process can be considered in three parts. Firstly, information about the state of the system must be extracted. This involves the measurement of some statistic which is considered to give a reliable estimate of system performance, and usually occurs at the receiving end of the system. Secondly, this information is presented to the parameter control function which makes a decision, based on some criterion, as to whether or not the required degree of reliable communication is being achieved. If conditions are not satisfactory some system parameter is varied, again according to some algorithm, in order to achieve 'reliable' communication. Thirdly, the parameter control function generates control signals which must be communicated to the selected parameter, and if this involves transmission between opposite ends of the system, it is important that these control signals are communicated reliably.

The design of an adaptive system therefore involves consideration of the following problems: the choice of one or more parameters to vary, the formulation and measurement of a 'system-state' statistic, the formulation of a

parameter control algorithm, the reliable communication of control signals, and the effect of control errors on the operation of the system.

A further major point to consider is the speed with which 'conditions' change. It is **desirable** that such conditions should change relatively slowly compared to the time taken for the system to adapt to these changes, if any meaningful improvement is to be gained from the adaptive technique.

In response to the question "to what should the system adapt?", it is possible, whilst still considering general principles, to split adaptive systems into two main types. These types arise from the two different user requirements that the system should adapt to: (i) the source data, and (ii) the channel. These will be called data-adaptive and channel-adaptive systems respectively.

4.1.1 Data-adaptive systems

In this method the user states that some parts of the source data have a greater importance (priority) than other parts, and must consequently be guaranteed a higher probability of correct reception. For example, in the transmission of monetary data, high order decimal digits should have a higher probability of correct reception than lower order digits.

The 'system state' information extraction process in this case simply involves examining sections of data to see what priority is required, and then the appropriate system

parameter is varied on a one-to-one basis. The decision to vary the system parameter is therefore non-statistical (deterministic), and the control process is located at the transmitter. If the decoder meeds to know how the parameter has been varied in order to decode correctly, control information (suitably protected against errors) can be prefixed to the relevant message section and transmitted through the main forward channel. Data-adaptive systems will not be considered further in this chapter.

4.1.2 *Channel-adaptive systems*

This type of adaptive communication system arises in an attempt to cope with the time-varying nature of real channels. The user requirement here is that transmission should be as efficient as possible whilst simultaneously being as reliable as possible for any particular channel condition, and adapt to remain so as channel conditions change. This requirement effectively means that the **average** probability of erroneous data being delivered to the sink is restricted to a range of values smaller than the range resulting from a *fixed* system, for a given range of channel conditions. The variations in sink error rate, which reflect channel variations, are therefore smoothed out, whilst data throughput is kept as fast as possible, concurrent with keeping the sink error rate below some maximum value.

The channel information extraction process is now one of building up an accurate statistical picture of the state of the channel upon which parameter variation decisions can be based. Implicit in this statement, however, is that

the channel must have some degree of 'memory' in order that past statistics can be assumed to predict the short-term future state of the channel. In particular, the *information gathering* time must be significantly shorter than the channel variation (memory) time if a meaningful statistical picture is to be built up.

If the channel is not predictable with time then the channel state must be assessed by real-time operations on received signals. This means that the adaptive control processes are located at the receiver, whereas control signals must usually be passed to transmitter parameters. In the case of a one-way system, this requires that control signals are passed from receiver to transmitter via a feedback link; or in the case of a full duplex link, control signals are multiplexed with the main forward data stream . The amount of control signalling required affects the efficiency of the system and is dependent on the speed with which channel conditions vary relative to the main forward data throughput rate. If the channel is varying rapidly a large amount of control signalling is required, and this implies a feedback bandwidth comparable to the main channel in the one-way case, or inefficient data-to-control channel usage in the duplex case. The situation is further aggrevated by the requirement for reliable control signalling, which in turn implies control signals must be transmitted in a redundant form requiring even more bandwidth or time.

It can therefore be seen that the design of a channel-adaptive system is strongly dependent on the characteristics

of the channel in use and in particular, for the system to be efficient and reliable, the channel must vary slowly compared to the total system adaption time.

4.2 *Parameter Variation*

This section briefly outlines various methods of adapting a data transmission system by the variation of one or more system parameters. Some of the parameters which can be varied are presented below, without reference to the rest of the system. It must be noted, however, that the choice of a particular channel-adaptive system involves an engineering compromise between all parts of the system.

The model of figure 4.1 can be refined so that the error control system is separated from the modulation-demodulation process. This gives rise to two broad sections of parameters that can be varied: parameters in the error control system, and parameters in the modem section (including the channel). It is also possible to vary parameters from both of these sections and for both sections to contribute to the channel statistics gathering operation.

Some penalty must usually be paid in order to achieve an adaptive system, regardless of which parameters are varied. These penalties (or trade-offs) indicate two basic types of adaptive system: those that vary parameters which effectively result in varying data throughput rate (rate-adaptive) - these pay the penalty of requiring buffers or a start-stop source; and systems that do not vary the throughput rate but pay the penalty of requiring high equipment complexity with possible channel and modem duplication.

The main system parameters which can be varied are now considered below.

(i) power and gain variation - with this method the transmitter power and the receiver gain is increased if the received signal is weak. The variation of transmitter power raises practical difficulties if large powers are involved and is only useful if it is not possible to transmit at maximum power all the time. Variable transmitter power would be useful in cutting down the high powers needed by tropospheric scatter links, thereby increasing system life and reducing potential interference to other users; as well as reducing the average transmitter power required for space vehicles where power is at a premium (Lieberman 1963). An additional point in the case of space transmissions is that the feedback link can be made relatively noise-free by the use of the high powers which are available at the ground station. In the case of a Rayleigh channel (section 4.4.2), Hayes (1968) has shown that variable power is remarkably effective in combatting the effects of fading. Finally, variable power can be considered as a non rate-adaptive method of increasing the amount of signal energy per bit, and requires control of the modem processes rather than the error control system.

Variation of receiver gain is the oldest method of coping with a time-varying channel, and any practical receiving system would already include some form of automatic gain control.

(ii) adaptive equalisation - an equaliser is basically a filter, located at the receiving end of the system, and designed to compensate for the time and frequency distortions introduced by the channel. Equalisation was traditionally introduced on voice telephone lines in order to flatten the amplitude characteristic and linearise the phase. Time spread in the channel gives rise to intersymbol influence, and much work has gone into providing adaptive equalisers for land lines so that once a data circuit is established, the equaliser automatically provides the 'best' overall characteristic for data transmission. Lucky (1968) describes the principles of baseband equalisation, and O'Neill (1966) has investigated the equalisation of quadrature carrier systems. The equalisation of a channel such as the HF radio channel, which has both time and frequency spread, has been investigated by Di Toro (1968). Di Toro describes a serial data transmission system called ADAPTICOM which periodically, and in real time, rejuvenates the time-spread correction networks, and is capable of coping with a time-frequency spread product of up to 1/200, provided the signal to noise ratio (SNR) is at least 25 db. This spread product is of the order met with in HF transmission below the MUF (see section 4.5), indicating that ADAPTICOM can achieve significant improvements on these circuits, provided the SNR remains high.

Under conditions of high SNR, equalisation can significantly reduce the effects of time and frequency distortion caused by the channel, and can therefore provide

a 'best' possible circuit. Equalisation cannot, however, cope with errors caused by noise, interference and wide variations in SNR. Under these conditions even an optimally-equalised 'best' circuit will be subject to time varying errors.

(iii) channel choice variation - the parameter under control here is the utilisation of a number of different channels with uncorrelated noise characteristics, and includes diversity techniques. Two types of operation are possible, one involving feedback and one not. In the first case the receiver periodically monitors conditions on all available channels and chooses the best one at any particular time. The system therefore 'hops' from channel to channel and control information may have to pass from receiver to transmitter. Alternatively, several channels are used simultaneously and the receiver reconstructs the message after combining all the received signals according to some linear or statistical weighting (diversity reception). Diversity techniques have traditionally been used to improve fading radio circuits, and rely on the principle that if all the channels are fading independently it is unlikely that the diversity branches will all simultaneously receive a poor signal. There are three main ways of achieving diversity signals: aerial diversity, frequency diversity, and time diversity. Aerial diversity, as used with radio systems, involves the use of a number of aerials that are either spaced apart, have a different angle of arrival, have different polarizations, or all three.

Frequency diversity involves the use of several bandwidth slots for the same message. Time diversity involves the repetition of messages, spaced for enough in time to escape the effects of channel memory, and is closely related to multipath diversity as used in the Rake technique (section 4.5.4) Diversity techniques can reduce the probability of error on time-varying channels by orders of magnitudes, but as the number of diversity branches increase, the improvement is on a diminishing returns basis. Aerial and time diversity techniques are appropriate for a single fixed-bandwidth slot, although aerial diversity is only applicable to radio systems. Aerial diversity is, however, complex due to the physical separations involved, whilst time diversity requires both transmitter and receiver storage, and is wasteful of forward channel time. Diversity techniques, including optimal adaptive signal combining, are dealt with in Stein (1966).

(iv) variations of baud rate - this method of adaption, in common with method (i), is an alternative way of varying the signal energy per bit arriving at the receiver. Signal elements are made long during poor channel conditions and short during good conditions. The method is rate-adaptive and can be considered as one form of variable redundancy coding, where the codes in use are variable-length repeated-bit codes. In common with variable redundancy coding, the method does not involve the variation of the modem-channel arrangement, and is attractive when presented with an unalterable existing data transmission system which has to be regarded as a 'black-box' with digital input and output. The variation of baud rate on a

continuous basis is impractical because of synchronisation problems, and any practical system would use a number of discrete rates suitably related to one synchronising frequency. Bello and Cowan (1962) have analysed the performance of on/off transmission, which may be considered as an extreme case of two-rate transmission, and Cavers (1972) has analysed continuous and discrete variable-rate transmission for Rayleigh fading channels. Cavers considers a system whereby the time-varying gain and phase of the channel are tracked and used as the reference with which to control the rate via a noiseless feedback channel. In addition, Cavers shows that ideal variable-rate control can almost eliminate the effects of flat Rayleigh fading, and that when constraints on feedback delay, and bandwidth, and number of allowable rates are introduced, significant reductions in transmitter power (of the order of 14db for full duplex) can still be obtained. A disadvantage of variable-rate control is the variable bandwidth used; but Cavers shows that even when bandwidth expansion is constrained, significant reductions in power are obtained when compared with the equivalent maximal-ratio predetection combining diversity system. Srinivasan and Brewster (1974) have expanded Cavers' work to include both rate and power control, and show that the performance of such a system more closely approaches that of a system operating in a non-fading channel.

(v) automatic repeat request (ARQ) - this is a rate-adaptive method in which error control coding is required in order to detect erroneous blocks prior to requesting a repeat.

ARQ has been recommeded by the CCITT for HF duplex transmission, and several practical HF links have been successfully operated using constant-ratio codes with a four character repetition cycle, although efficiency may be very low during high noise periods due to continued repeat cycles. A feedback link is required for ARQ and serious mis-operation can result if this channel is noisy.

(vi) code variation - this method has been called variable redundancy coding, because the power (and hence the redundancy) of the error control coding scheme is varied according to the channel conditions. The method is rate-adaptive and can be used on any digital data transmission system without varying the particular modem-channel system provided. Variable redundancy coding is more fully dealt with in the following chapters.

(vii) other methods - these include changing the type of error-control, i.e. decoding or interleaving to suit the channel, and are generally not rate-adaptive.

4.3 *Extraction of channel statistics*

The state of any practical channel is subject to both long term and short term variations. It is usually possible to predict long-term variations with some degree of accuracy, by the use of established forecasting measures which are based on the past long-term behaviour of the channel. The knowledge of long-term variations gives some indication of the overall conditions with which the adaptive system must cope. As the time scale of channel variations

becomes shorter, however, and in particular when variations are of the order of the system baud rate, continuous monitoring of some system variable is necessary, in order to predict short-term channel variations. The choice of which parameter should be measured in order to reliably reflect short-term channel conditions depends on both the practical transmission system and the physical channel in use. If it is possible to have an explicit mathematical model of the channel, it may be possible to form a one-to-one relationship between channel model parameters and receiver parameters which can be measured. This then makes it theoretically possible to have an exact description of the channel at any particular time. If no explicit channel model is possible, an heuristic approach to the problem of which receiver parameter (or combination of parameters) to measure in order to reliably approximate the channel state, must be taken.

The real-time measurement of parameters that are assumed to predict the short-term behaviour of the channel must usually be done at the receiver, and depends on which parts of the system are available for measurement. Two rough divisions may be made. Firstly, if the system is considered as a 'digital black box', then measurements must be taken on the digital output of the receiving modem, or on the error control decoder. If, on the other hand, the receiving system can be split into the receiver-demodulator-decoder sub-systems, measurements can be taken on both the outputs and internal behaviour of these sub-systems. It is therefore possible to consider measurements

taken before, during, and after demodulation.

The simplest measurement to make before demodulation, that is, on received signals, is the amount of power received in the bandwidth slot being used. The disadvantage here is that signal-plus-noise is being measured, and if the noise component is high (for example from interference), the estimate of channel gain derived from this measurement will be inaccurate. The sending of one or more pilot tones in an attempt to assess channel gain and phase variation is another commonly used method, as is the sending of known sounding signals. More complex decision-directed statistical analyses of received signals are also possible, but will not be dealt with here (Stein 1966, Wainstein 1962).

The demodulation process can be considered to be the conversion of received signals into a binary digit train. If the 'difficulty' of this conversion process can be monitored, this information can be used as an estimate of the channel state. More specifically, soft decision decoding methods (Chase 1972) in which the error control decoder uses information supplied by the demodulator on the probability of a given bit having been correctly demodulated, can also be used to indicate the state of the channel.

Finally, only the digital demodulator output may be available and any channel state information must be extracted from this binary output. The variable redundancy coding schemes presented in the next chapter assume this condition. Two basic methods are possible. Firstly, a known sounding binary sequence can be time

multiplexed with the main data sequence, and channel statistics are calculated by operations on this sequence. Although the available data throughput is reduced by this technique, it is possible to use the known sequence to also transmit synchronising information. If channel variations are fast, however, the sequence must be transmitted frequently, and this entails considerable waste of forward channel time. Alternatively, sounding sequences may be multiplexed with, or superimposed upon the data in some other fashion to provide continuous channel assessment. The second method possible is to measure quantities relevant to the error control system such as bit error rate, block error rate, burst length, etc. The advantage of this method is that the only redundancy required is that used for error control, and in addition some channel information can be obtained after decoding each block.

4.4 Time-varying channels

Most physical channels are subject to varying degrees of time-dependent behaviour and therefore require some form of adaptive system if transmission is to be optimised. Man-made channels, on the other hand, are essentially steady state processes which, once optimised do not vary greatly with time, and do not require a continuously adaptive system.

The performance of digital transmission links is often analysed by assuming steady received signals contaminated

by additive stationary Gaussian noise. Unfortunately, most real channels dependent on natural media are prone to noise that may not be additive, stationary or Gaussian, and it then becomes very difficult to formulate theoretical models for these channels. Under these conditions the receiver experiences *signal fading* conditions and it is these fading effects that are combatted by the adaptive system. A non-adaptive system, on the other hand, is normally designed with sufficient *margins* to cover any such variations. The time scale of channel variations may range from those smaller than the shortest transmission pulse length, which are not inherently of interest as they are usually experienced by the receiver in some averaged form, to long term variations of the order of hours or years. It is with variations in between these two extremes that an adaptive system must cope.

In order to appreciate the fading conditions met with in practical transmission systems, this section gives a qualitative characterisation of fading channels as a preliminary to the discussion of the HF channel given in section 4.5.

4.4.1 Fading radio channels

Historically, radio communication has been dominated by the HF band (3 to 30 MHz) in which communication is achieved by ionospheric reflection, and indeed this medium remains one of the worst as far as interference and severe fading is concerned. The need for further bandwidth has extended the radio communication spectrum into the VHF,

UHF, and microwave bands in which the majority of operation is on a line-of-sight basis, with long distance communication being achieved by using a chain of point-to-point links between repeater stations. Some satellite relay links are prone to severe fading conditions. Studies of the use of millimeter and smaller wavelengths on active satellite links indicate severe fading due to rainfall absorption (Intelsat/IEE conference 1969). The use of the moon as a passive reflector gives rise to severe selective fading (Ingalls 1961, Anderson 1961), whilst the orbital dipole belt experiment (West Ford experiment 1964) involves fading more severe than that encountered in HF skywave propagation. The majority of line-of-sight systems, including most satellite relay links, are however basically steady state systems and are not prone to severe fading.

Transhorizon propagation at frequencies above which ionospheric reflection fails (approximately 30 MHz) is possible by using scattering modes. These modes use high power transmitters and give rise to weak but persistent fading fields. Transhorizon propagation at UHF and above by tropospheric scatter enables reliable communication out to distances of several hundred miles beyond the radio horizon, and ionospheric scatter of VHF frequencies can extend this range. Ionospheric scatter is, however, only efficient in the 35 to 50 MHz band and this, together with the high power required, has precluded any large scale use of this mode.

Some of the ionisation at ionospheric altitudes is due to meteor trails which last for fractions of seconds and

can act as wideband reflectors suitable for moderate power communication. Such trails are present for as much as ten per cent of the time and have led to the 'meteor-burst' mode of digital communication. In this mode, transmitter and receiver continually search for the existance of a suitable trail, and when one is found a 'burst' of data is sent for as long as possible.

The medium and low frequency bands are generally used for broadcast and radio-navigation purposes rather than communications, and are not considered further here.

4.4.2 *Multipath and fading*

The use of natural media for radio communication implies a dependence on the random variations inherent in natural processes. The two main variations experienced by natural channels are relatively long term variations in attenuation, and short-term variations in electrical path length. More unfortunately, several different paths may exist simultaneously when only one is required (multipath) giving rise to what may be generally called destructive interference fading. These short-term fading effects, due to both attenuation and interference variation, are the effects with which an adaptive system must cope.

Multipath propagation gives rise to signals that arrive at the receiver displaced sequentially in time after having been transmitted simultaneously. The observable effects of such path length variations depend on the magnitude of the time-differentials between the various paths.

The effect of path changes with time differentials smaller than the reciprocal bandwidth, and in particular of the order of the carrier wavelength, gives rise to sequentially displaced sine waves at the receiver which sees only the result of their superposition. The several different signals therefore interfere constructively or destructively depending on the relative phase shifts introduced by the several different paths. If the path lengths then change continuously and randomly, so do the relative phases, and the envelope of a received unmodulated carrier will be observed to be governed by a Rayleigh distribution. The effect of these rapid variations in path length therefore gives rise to the type of interference fading called Rayleigh fading, and experimental observations confirm that many natural channels experience fading that can be described by the Rayleigh model, or by modifications of the model.

The effect of a fading medium on a band of frequencies, rather than a single tone, is important because fading can be frequency selective. If a band of frequencies fades uniformly, then this condition is known as flat fading. More usually, however, two spaced frequencies will fade in a highly uncorrelated manner, thereby giving rise to time-varying relative distortion within the spectrum of the modulated signal. This condition is known as frequency selective fading.

In many situations a natural medium is characterised, for at least part of the time, by having one strong path as well as several additional weak paths. The received signal

can then be considered to comprise a steady (specular) signal plus a Rayleigh fading signal. This condition is called Rice fading.

Multipath signals with time differentials longer than the reciprocal bandwidth are observable as 'echoes' in pulsed systems and generally have the effect of producing time delayed or 'smeared' versions of the digital data which interfere causing errors. Particularly severe multipath can occur if two equal strength paths exist, with time differentials of the order of the reciprocal bandwidth. The baud rate of serial data transmission systems is therefore fundamentally limited by the multipath structure of the medium, and is generally chosen so that individual bit widths do not vary more than ±25% due to the varying multipath interference experienced on the particular channel in use. The resolution of multipath structure is a feature of several anti-multipath systems and requires high bandwidth (narrow pulses) if fine detail is to be observed (Stein 1966).

A crude measure of the rapidity of fading due to the superposition of all channel fading effects is the fading rate. This is usually defined for a single unmodulated frequency as the average rate of excursions of the received envelope level downwards across its median level. A fading bandwidth can then be defined as the average bandwidth occupied by the time varying channel gain function. It is important to note that fading rate is most meaningful under flat fading conditions, as equal-power antiphase selective fading

of components in a bandpass signal can give the illusion of a 'steady' signal as far as average levels are concerned.

4.5 *High-frequency ionospheric skywave communication*

This section considers several aspects of HF radio communication. Firstly, the characteristics of the ionospheric layers are considered together with the basic limitations they place on HF propagation and data transmission. Secondly, sounding systems are considered, and finally several modulation methods, used on the HF band, are described.

4.5.1 *Ionospheric layers and propagation*

Four significant ionospheric layers have been identified, the D, E, F_1, and F_2-layers. Propagation conditions depend on the heights and electron densities of these layers which vary according to solar radiation conditions. HF communication is therefore subject to strong diurnal and seasonal variations, as well as more 'drastic' effects such as magnetic storms and sudden ionospheric disturbances (SIDs) which are related to solar flares and sunspot activity. In addition, man-made interference from atmospheric nuclear tests can create large scale disruptions in ionospheric communications.

The D-layer at about 70-100 km altitude, is the lowest layer, and acts mainly as an absorbing layer due to the high molecular density at this altitude. Absorption is greatest at local noon when ionisation is greatest and propagation blackouts due to solar activity

are usually related to intense ionisation of the D-layer. The higher the frequency of radiation the lower the D-layer absorption is, and it is therefore possible to calculate the absorption loss over a particular HF circuit for several different frequencies. In particular, a lowest usable frequency (LUF) which guarantees a minimum received signal-to-noise ratio may be calculated, and methods of performing these calculations are given in the appropriate CCIR and CCITT documents.

The other ionospheric layers act predominantly as reflectors. Radiation incident on these layers is reflected (or more correctly, gradually refracted) down to earth at a distant point, thereby enabling long distance transhorizon point-to-point communications. For a wave launched vertically at a point in the reflecting layer to be completely refracted back to earth, its frequency must be less than the critical frequency of the layer which is dependent on the prevailing electron density. Similarly, an obliquely incident ray must not exceed a certain maximum frequency for refraction back to earth, as refractive index is inversely proportional to frequency. Thus for any two stations it is possible to define a maximum usable frequency (MUF), above which the transmitted ray will not be bent sufficiently, and will therefore miss (or skip) the receiver which is then said to be in a skip zone. Operation at, or just below, the MUF is desirable for two reasons. Firstly, absorption decreases with increasing frequency, and secondly multipath is reduced due to many paths skipping the receiver. Several international organisations issue monthly median MUF prediction

charts in order to enable operators to optimise their HF circuits. In general, however, it is customary to adopt an optimum working frequency (OWF) that ensures that the operating frequency is below the actual MUF for 90% of the time (Greenburg 1962). The spectrum of available frequencies (those between LUF and MUF) is an important limitation imposed by the ionosphere, and this range is greatest during sunspot maximum, as is the accuracy of frequency prediction. Unfortunately, however, SIDs are also at their worst during sunspot maximum.

The lowest reflecting region is the E-region (100-150 km) which is capable of providing only short distance communication because of its height and its low critical frequency (4 MHz maximum). The F-region at 160-450 km exists as two separate layers (F_1 and F_2) in daytime, and exhibits higher critical frequencies which enable much longer distance communication.

The propagation of waves at the MUF occurs in the 'single-hop' mode, but at lower frequencies several paths are possible including 'multiple-hop' propagation in which the wave repeatedly bounces between earth and reflecting layer, and/or between E and F-layers, thus enabling very long distance propagation.

Multihop propagation via several paths, and single hop propagation well below the MUF, accounts for the severe multipath effects observed on HF circuits where path-time differences of 0.5-5ms are experienced. This seriously limits the baud rate of serial data transmission

systems, so that pulse lengths commonly in use are in the 10 to 20ms range, and generally not shorter than 3ms. The use of antimultipath measures such as operation at the MUF, path discrimination with vertically steerable aerials, and 'Rake' may reduce this minimum to 0.5-1ms, but there still remains a fundamental limitation because of the 0.1ms order of multipath due to the roughness of the ionospheric layers.

The earth's magnetic field also affects ionospheric propagation by giving rise to 'polarisation fading' and wave splitting which contributes to the multipath nature of the medium.

4.5.2 Ionospheric sounding

The state of the ionosphere may be probed by the use of sounding systems.

Vertical sounders transmit short pulses of 30 to 100 µS duration at 30-120 pulses per second whilst sweeping the operating frequency across the HF band. By this method a dynamic plot of frequency versus propagation delay is produced, which can be used to show the number of existing layers, their height, and their critical frequencies. A large number of vertical sounders distributed around the earth could then give useful up-to-date information to operators attempting to set up or optimise a particular circuit. Vertical sounding in itself, however, corresponds to sounding a zero-range path.

Oblique incidence sounding between separated stations using swept-frequency modes is generally impractical due to synchronisation problems, and therefore operation usually occurs on spot frequencies. Oblique soundings can be used to analyse both the fading and multipath characteristics of the medium.

A third method of ionospheric probing involves the reception of sounding signals that are back-scattered by the earth. A receiver in the vicinity of the transmitter can then analyse pulse travel times and deduce which areas of the earth can be reached by waves of the given frequency. This information can then be used to immediately set up a working HF circuit.

4.5.3 *Fading characteristics*

Oblique sounding has been used to determine the short-term fading characteristics of single path HF circuits. These soundings reveal that over periods of about five minutes envelope distributions obey a Rayleigh or a Rayleigh plus specular law. In addition these types of fading are generally observed with nonstationary intensity or nonstationary specular to Rayleigh component over such periods. For longer periods of 15 to 60 minutes, over which the ionosphere is generally highly nonstationary, fading appears to obey a log-normal law with fading rates of the order of 0.1 per second for single-path flat-fading transmission.

Interference fading of HF links is generally frequency selective, and selective effects set in at frequency spacings

well below 1 kHz with the result that frequencies within a nominal 3 kHz bandwidth can behave in a highly uncorrelated manner. Fading rates under these conditions are then generally higher, of the order of 1 to 10 per second.

Long term fading is related to solar effects and ranges from the diurnal variation of ionisation due to illumination, up to the eleven year sunspot cycle.

4.5.4 *Modulation systems for HF data transmission*

In order to cope with the irregularities and limitations of the HF medium, several complex modulation methods have been proposed. These methods, however, can be split into two main types, methods that occupy a nominal 3 kHz bandwidth, and wideband antimultipath methods. Given a 3 kHz bandwidth, signals can either be transmitted in a single high-rate serial form that occupies the whole bandwidth, or as a number of parallel low rate signals, each occupying a sub-channel in the 3 kHz slot. Regardless of whether the system is serial full-band or multichannel narrowband, the main modulation methods are phase modulation, frequency modulation, or modulated continuous wave (tone transmission), and it is possible to consider single channel (or sub-channel) modulation without loss of generality. On-off keying (OOK) of the carrier is rarely used because of its susceptibility to selective fading and interference.

In general, signal detection can be coherent (carrier or sub-carrier phase available at the receiver) or non-coherent (envelope detection). It is well known that coherent systems

have a lower probability of error for a given signal to noise ratio, due to out of phase noise rejection, but this assumes a noise-free phase reference at the receiver. Under conditions which tend to destroy the coherence of received signals, such as rapid selective fading, it is possible, in practice, for a non-coherent system to outperform a coherent one.

(i) Frequency shift keying (FSK)

FSK signals can be generated by allowing a binary pulse train to frequency modulate a carrier or sub-carrier. This gives continuous-phase FSK. If the pulse train has sharp transitions, however, this effectively results in the transmission of two distinct carrier frequencies (or tones), one for '0' and one for '1'. This FSK signal can therefore be generated by selectively gating one of a pair of independent tone oscillators, and this generally gives rise to discontinuous-phase at the transitions. For reasonable separation of the two frequencies, it becomes advantageous to demodulate the received FSK signal not by the use of a conventional FM discriminator, but by the use of a pair of filters, one centred on each tone frequency. This is supported by statistical decision theory which indicates that the optimum detector for an FSK signal in Gaussian noise involves a pair of cross-correlators, which for certain signals can be replaced by a pair of realisable matched filters.

Under high SNR conditions, noncoherent FSK achieves the same error rate as ideal optimised-threshold noncoherent OOK, for the same average power. OOK, however, requires the detection threshold to be optimised at each SNR whereas

FSK does not, thereby making FSK much superior under fading conditions.

Coherent (synchronous) detection of FSK requires that reference signals of both tone frequencies, including accurate phase, are available at the receiver. Coherent detection has the effect of rejecting the portion of noise that is in phase quadrature with the desired signal, thereby achieving the theoretical supremacy over non-coherent systems. Performance is equal to that of optimal-threshold coherent OOK with the same average power, and 3 db worse than ideal phase shift keying (PSK). Coherent FSK out-performs non-coherent FSK, but this supremacy is vanishingly small at high SNR. Coherent FSK is not generally used because of the double reference required, and also because if a coherent system is to be used, PSK is better.

An important problem in FSK systems is the choice of an optimum tone spacing. The assumption of no 'crosstalk' in FSK systems implies orthogonal waveforms, which in turn requires that the two tone frequencies are multiples of 1/T, the pulse modulation rate. If a condition of 'zero start-phase' can be assumed, the minimum tone spacing for othogonality is 0.5/T. By increasing the tone spacing, however, the waveforms can be made partially 'antiparallel', and it can be shown (Stein 1966) that the 'best' spacing for tone discrimination is 0.7/T.

(ii) Phase shift keying (PSK)

Simple binary PSK encodes information in the *algebraic sign* of the carrier, that is, two phase states separated by

180 degrees. More complex 'carrier' waveforms such as PN sequences can be used, to facilitate synchronisation or multipath resolution. Coherent detection of PSK involves a synchronous detector which requires an error-free reference waveform, accurate in both frequency and phase. Ideal PSK also requires long-term stability of this reference signal, and any phase error in the reference reduces the theoretical 3 db performance advantage over coherent FSK. An alternative method that does not require this long-term stability and gives a 3 db advantage over a non-coherent FSK, is differential phase shift keying (DPSK). With this method, information is differentially encoded as the phase change between successive waveforms. Coherent detection is then possible by using the 'previous' waveform as a reference for the 'current' waveform. Coherency is now only required over a period of 2T seconds, but performance is inferior to ideal PSK because the reference is contaminated by noise to the same extent as the information waveform. In addition, a high probability of double adjacent errors exists due to the fact that a single DPSK waveform is involved in two successive binary decisions.

(iii) FSK and PSK in non-Gaussian noise conditions

The analysis of single-channel PSK and FSK systems under conditions of slow non-selective Rayleigh fading again shows an exact 3 db advantage for ideal PSK over coherent FSK, and for DPSK over non-coherent FSK, at all mean SNRs. For large SNRs all these systems have linearly decreasing error probabilities with increasing SNR, as opposed to the

exponential decrease in steady noise situations. Therefore higher SNRs are required in the fading case than in the non-fading case in order to achieve the same error probabilities. The magnitude of these *system margins* is approximately 10 db for error rates in the 10^{-2} to 10^{-3} range, with an additional 10 db for every factor of ten further decrease in allowed error rate.

Bello and Nellin (1963) have done considerable analysis on FSK and PSK systems under conditions of fast flat fading and slow selective fading, including allowing for the use of diversity. These results indicate that non-coherent FSK is less sensitve to these extreme variations than DPSK, and that phase-continuous FSK is better than discontinuous FSK. The superior performance of non-coherent FSK under these extreme coherence-destroying conditions is because orthogonality is based on *energy* concentration around each tone frequency, whereas DPSK requires *phase coherence* over two pulse lengths for the preservation of symbol orthogonality.

(iv) Multichannel systems

Multichannel 3 kHz data transmission systems utilising low-rate parallel transmission haye been constructed with most modulation methods.

Amplitude modulation with differential-coherent detection has been employed on the two-tone predicted wave system (Doelz 1954) and the multi-tone Piccolo system (Robin 1963).

Narrowband FSK with frequency shifts of the order of

80 Hz on each subcarrier, with tone interleaving to combat selective fading, has also been used, as has wideband FSK with shifts of 850 Hz.

Phase modulation has been used on many systems including Kineplex (Mosier and Clabaugh 1958) which has orthogonally spaced subcarriers that are 4-phase differentially keyed, to provide a total rate of 3000 bits per second when all subchannels are keyed at 75 bauds.

Variable rate adaptive transmission may be provided by duplicating information on several subcarriers, and this is the principle used in the manually adaptive "Katheryn" system (Zimmerman 1967, Kirch 1969). The choice and performance of a particular multichannel system is strongly dependent on the channel conditions under which the system must operate. This makes the theoretical comparison of multichannel systems for HF operation very difficult, rather, one is left to compare the results of field tests of various equipments. Bello (1969) reviews various multichannel systems for tropospheric scatter transmission.

(v) Wideband systems

Wideband transmission originated with the 'Rake' system (Price and Green 1958). The original Rake system used orthogonal PN sequences 1023 bits long, repeating after 8.5ms, to represent the binary digits. At the receiver the delayed versions of the PN sequences, due to multipath, are brought into alignment and added, thereby giving the

impression of a single strong path. The system operates best under heavy multipath conditions, but is wasteful of bandwidth, as the bauds are 10ms long for the 10 kHz bandwidth used.

CHAPTER 5

Variable Redundancy Coding

This chapter describes the variable redundancy (VR) coding technique of adaptive error control for time-varying channels, and assesses the theoretical performance advantages of VR coding, as well as examining the practical problems involved in the design of VR systems. A general view of VR coding is therefore presented in this chapter, as a preliminary to the investigations of the next two chapters, which are concerned with specific VR systems.

The VR technique is first outlined, and this is followed by a general analysis of the advantages of the technique when used on time varying channels. The basic VR system is next described, together with its operation under different error control modes, and the problem areas in the design of such a system are defined. The choice of suitable sets of codes for use in VR systems is then considered, and each type of fundamentally different code set is analysed with respect to performance, implementation, and relative advantage. Section 5.7 describes methods of extracting channel information from the incoming digit stream, as well as the formulation of criteria for initiating code changes. Finally, feedback code-control signalling and the effect of feedback errors are considered, showing how code sets with mutual distance (Appendix A) can be advantageous for VR systems with noisy feedback links.

5.1 *The basic VR technique*

Variable redundancy coding for a data transmission system involves the use of a *set* of codes as opposed to the *single* code of a fixed redundancy (FR) system. Each code in the set has a different redundancy, and hence EDC power, and can therefore cope with a different level of channel noise before exceeding some specified (fixed) probability of erroneous decoding (P_e). This set of codes is used by the VR system to provide a variable amount of EDC power that automatically adapts to changing channel conditions, so as to 'match' the code in use at any particular instant to the amount of coding power required by the channel. Therefore: when the channel is in a highly noisy state, high-power, low-rate codes are used; whilst under low noise conditions, high-rate (efficient) low-power codes are selected. The VR system and the channel are considered to be 'matched' if no higher-rate code can be used without exceeding a certain specified maximum probability of erroneous decoding.

If it can be assumed that the VR system is capable of accurately assessing the channel so as to use the correct code for the prevailing conditions, and in addition, if the channel changes relatively slowly compared to the time taken to assess the channel and physically effect a code change, it is theoretically possible for a VR system to realise two important advantages over an FR system: increased information throughput *and* decreased probability of erroneous decoding.

Firstly, an FR system must employ a code which is capable of coping with the worst possible channel conditions, that is, given some maximum value of channel noise, and the proportion of time that this maximum is exceeded, a high redundancy code must be employed if the user is to be guaranteed that P_e remains below some desired maximum (P_e max) for a large proportion of the time. Reducing the code power on the fixed system therefore increases the proportion of time $P_{e\ max}$ is exceeded and vice-versa. Under conditions of low channel noise, therefore, the FR system is unnecessarily inefficient because it is providing an unnecessarily low value of P_e (the user's guarantee is fixed by the high noise channel conditions). A VR system operating under low noise conditions, however, sacrifices the low value of P_e and uses a higher rate code with a P_e comparable to the high-noise P_e, thereby realising the advantage of increased data throughput, whilst still maintaining the user guarantee that P_e is below $P_{e\ max}$.

Secondly, given that a VR system can achieve higher average throughput than an FR system, all of this advantage could be 'traded-off' by employing very highly redundant codes during periods of very high channel noise. This would increase the user guarantee whilst maintaining the same average throughput.

From the preceeding arguments it can therefore be seen that, ideally, a VR system can simultaneously realise the advantages (over an FR system) of increased overall data throughput *with* increased user guarantees. In addition,

P_e, which varies widely with channel conditions on an FR system, thereby making the overall *average* transmission P_e (P_{eAV}) a somewhat meaningless statistic, tends to remain relatively constant on a VR system; and because the range of P_e variations is greatly reduced, P_{eAV} becomes much more meaningful to the user. Indeed, if an infinite number of codes were employed the VR user could then be guaranteed that P_e remains constant over all channel conditions, up to a certain maximum, instead of being guaranteed that P_e remains less than $P_{e\ max}$ and (depending on the number of codes) does not vary much about P_{eAV}.

5.2 *Performance analysis of the VR technique*

In this section the VR technique is applied to a time varying channel, and its performance is analysed, in order to obtain some quantitative measure of the improvement in throughput and error-rate guarantees that that can be obtained by using VR coding. The analysis is intended to be as general as possible and does not, therefore, refer to specific codes or specific VR systems. The technique is applied subject to certain idealisations about the VR system; that is, noiseless feedback, accurate channel knowledge, slowly varying channel, and so on, and although in practice these assumptions may be very far from the truth, it is useful to have a rough idea of VR capability, before presenting specific results in later chapters.

In order to calculate the performance of an FR or VR system it is necessary to have a model of the time varying channel being considered. For simplicity, it will therefore

be assumed that the 'short term' behaviour of the channel is modelled by a BSC (section 2.5) with average bit error rate (BER) p, and that the 'long term' behaviour is obtained by making p vary with time, according to some rule or distribution.

The next stage in the analysis would be to choose a set of codes and obtain BSC performance figures for each of these codes. In order to continue in general terms, however, specific codes are not considered; instead, BSC performance versus rate figures can be calculated from the bounds described in section 3.2. In addition, the decoding scheme will be assumed to be of the 'bounded distance' type, regardless of the type of error control being provided.

5.2.1 Error correction

For error correction on the BSC with bounded distance decoding, that is, the correction of t or fewer errors only, the probability of erroneously decoding a block (P_e) is given by:

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} p^i q^{n-i} \qquad 5.1$$

Equation 5.1 represents the 'tail' of a binomial distribution, which can be estimated by using the formulae given in Appendix A of Peterson (1972) to give:

$$P_e \leq \lambda^{-\lambda n} \mu^{-\mu n} p^{\lambda n} q^{\mu n} \qquad 5.2$$

where $\lambda=(t+1)/n$, and $\mu=1-\lambda$.

In order to retain generality it is necessary to remove the dependance on n exhibited by equation 5.2. This can be done by evaluating the asymptotic result obtained by letting $n \to \infty$. It is then convenient to work in terms of E: the *reliability* or error exponent; where,

$$E = - \lim_{n\to\infty} \frac{1}{n} \log P_e \qquad 5.3$$

(Reliability may be thought of as the exponent with which P_e may be made to vanish with increasing n.) For large n equation 5.2 and 5.3 give,

$$-\frac{1}{n} \log (\lambda^{-\lambda n} \mu^{-\mu n} p^{\lambda n} q^{\mu n}) = E(\lambda,p) \qquad 5.4$$

where

$$E(\lambda,p) = H(p) + (\lambda-p)H'(p) - H(\lambda) \qquad 5.5$$

and $H'(x)$ is the first derivative of the entropy function: $H(x) = -x \log x - (1-x) \log (1-x)$. If t and n are both $>>1$ then $\lambda = t/n$, and the bounded distance asymptotic lower bound on reliability (E_{BD}) is:

$$E_{BD} \geq E(t/n, p). \qquad 5.6$$

In order to plot reliability versus rate we require values of t/n for given rates; these can be obtained by using the minimum distance bounds of section 3.2. Using the asymptotic VGS lower bound on d (equation 3.13) gives values of t/n versus rate which are guaranteed achievable and, although reasonably pessimistic for short codes, are met in practice by random error correcting codes of moderate length ($n \leq 1000$, e.g. BCH codes, section 3.5). For large n the VGS bound gives:

$$1 - \frac{k}{n} \leq H(\frac{d}{n}) \quad , \quad t/n = d/2n \; ; \qquad 5.7$$

so that the asymptotic lower bound on reliability for

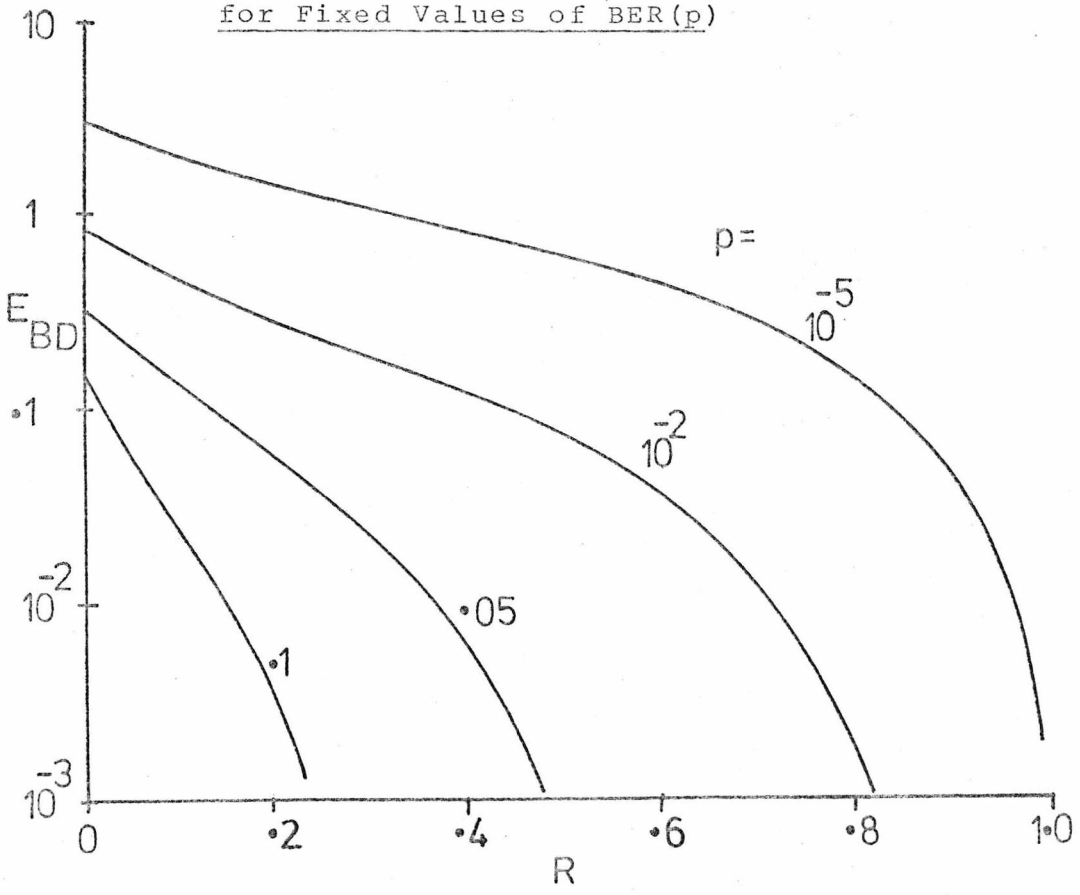
bounded distance decoding becomes

$$E_{BD} \geq E(^{d}/2n, p). \qquad 5.8$$

Equations 5.7 and 5.8 can now be used to plot rate versus reliability for several different BER's, as shown in figure 5.1, and it is these curves that are used for an initial comparison of VR and FR systems.

Figure 5.1

Reliability (E_{BD}) versus Rate (R), for Fixed Values of BER(p)



As a first approximation it is convenient to make two simplifying assumptions. Firstly, the channel BER is assumed to take a finite number of discrete values, each of which is in effect for an equal proportion of the total transmission time, and secondly, the codes used by the VR and FR systems exactly meet the E_{BD} bound so that the

rate versus reliability curve for each state is a curve similar to those of figure 5.1.

The following example shows how the VR and FR systems may be compared. Consider a four state channel with p values of 10^{-5}, 10^{-4}, 10^{-3} and 10^{-2}. The user requires that the reliability is ≥ 0.1 under all conditions. An FR system must therefore have a reliability of 0.1 under the worst conditions ($p=10^{-2}$) and from figure 5.1 the fastest rate that can be used whilst still satisfying these requirements is $R_F=0.44$. We now assume a VR system that adjusts the code rate so that for each equal duration channel state (value of p) the reliability is maintained at 0.1, and the user is therefore still guaranteed an average reliability of ≥ 0.1. For the same values of p the corresponding code rates are: 0.85, 0.79, 0.685, 0.44, giving an average rate of $R_V=0.69$, which leads to an *improvement factor* (R_V/R_F) of 1.57 over the fixed redundancy case. Consider now a similar four-code VR system that maintains an average reliability of ≥ 0.2 instead of 0.1. In this case the corresponding rates are: 0.75, 0.675, 0.55 and 0.295, giving an average rate of 0.57. Therefore, compared to the FR system, this VR system has achieved an increase in *both* throughput and reliability.

Table 5.1 below presents similar VR and FR comparisons for different reliabilities, channel error rate range, and number of codes.

Table 5.1

VR Performance

Result Number	Reliability ≥	Channel error rate range	FR system rate (R_F)	No. of codes in VR system	VR system rate (R_V)	Improvement factor R_V/R_F
1	10^{-2}	10^{-1} to 5×10^{-5}	0.15	8	0.72	4.8
2				4	0.68	4.5
3				2	0.51	3.4
4	10^{-1}	10^{-1} to 5×10^{-5}	0.02	8	0.52	26
5				4	0.49	24
6				2	0.35	18
7		10^{-2} to 5×10^{-6}	0.44	8	0.71	1.6
8				4	0.69	1.6
9				2	0.62	1.4
10		10^{-1} to 5×10^{-4}	0.02	8	0.44	22
11				4	0.39	19
12				2	0.29	15
13		10^{-2} to 5×10^{-5}	0.44	8	0.68	1.55
14				4	0.65	1.47
15				2	0.58	1.32
16	0.2	10^{-2} to 5×10^{-6}	0.30	8	0.59	2.0
17				4	0.57	1.9
18				2	0.49	1.6
19	0.5	10^{-2} to 5×10^{-6}	0.08	8	0.37	4.6
20				4	0.35	4.4
21				2	0.27	3.4

By inspecting table 5.1 it is possible to draw the following conclusions about VR performance behaviour.

(i) For a given 'worst error rate' VR improvement increases only slightly with increasing range of error rate. (Results 10 and 4, 13 and 7). That is, very low noise periods do not significantly affect VR performance if high error rates also occur.

(ii) For a given range of error rate the improvement increases substantially with increasing 'worst error rate'. (Results 7 and 4, 13 and 10). Thus, high noise periods have a major affect on VR performance.

(iii) As the required reliability increases, so does the improvement. (Results 1 and 4; 7, 16 and 19)

(iv) Improvement increases with the number of codes but on a rapidly diminishing returns basis. For example, the 2 code VR rates in table 5.1 are about 70% of the 8 code rates. Increasing the number of codes beyond 8 would not provide enough rate increase to justify the rapid increase in complexity, and in fact most of the 8 code rate results of table 5.1 are the same (to two decimal places) as the results for an infinite number of slots and codes.

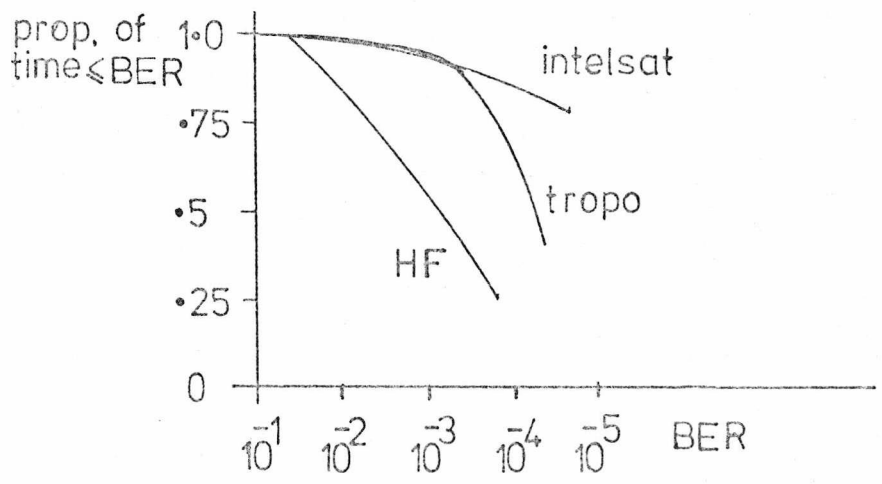
In general it is therefore possible to say that a VR system, whilst using only a few codes, is capable of significantly increasing throughput when compared to an FR system, and that the system's relative performance is best under conditions of high required reliability *and* high worst-noise.

We now consider a modified channel model which provides a better approximation to real channels in that the channel slots can be of unequal duration. This has a large effect on VR performance, and may make VR improvement minimal if, say, the worst BER's are in operation for most of the time. The distribution of short term BER versus time for different channels has been investigated and reported in a number of papers. This distribution often turns out to be log-normal about the long term mean (Stein 1966, Balkovic 1971) with a wide range of standard deviations (s.d.) if a very large number of runs are considered. If the distribution within any one 'transmission time' is plotted however, this may be highly non-log-normal, particularly for the HF channel. It is therefore important to bear in mind that, in the following analysis, VR performance attributed to a particular channel is based on long-term BER distributions.

Figure 5.2 is from Brayer (1971) and shows BER distributions for the HF, Tropospheric scatter, and Intelsat II channels.

Figure 5.2

BER Distributions



Of the three curves the Intelsat channel is the only one that cannot be reasonably well approximated by a log-normal distribution. The HF and Tropo channels exhibit strong log-normal behaviour with long term means of 10^{-3} and 5×10^{-5}, and standard deviations (s.d.) of 1 and 0.7 respectively.

An 8 state channel model based on the curves of figure 5.2 can be constructed, and table 5.2 shows the proportion of time spent in each state.

Table 5.2

	State	1	2	3	4	5	6	7	8
	p	2×10^{-2}	10^{-2}	5×10^{-3}	2×10^{-3}	10^{-3}	5×10^{-4}	2×10^{-4}	10^{-}
Proportion of time	(HF	.16	.08	.11	.10	.09	.13	.13	.2
	(TROPO	0	0	.03	.04	.05	.1	.13	.65
	(INTELSAT	.01	.02	.02	.01	.01	.03	.03	.87

The performance of VR and FR systems when used on these channels is calculated as before and table 5.3 presents several results.

Table 5.3

Reliability ≥	Channel	FR system rate R_F	No. of codes in VR system	VR system rate R_V	Improvement factor R_V/R_F
0.1	HF	.3	8	0.62	2.07
			4	0.59	1.97
			2	0.51	1.71
0.2		.18	8	0.49	2.74
			4	0.46	2.57
			2	0.38	2.13
0.1	TROPO	.55	8	0.76	1.39
			4	0.74	2.56
			2	0.68	1.23

Table 5.3 contd.

Reliability ≥	Channel	FR system rate R_F	No. of codes in VR system	VR system rate R_V	Improvement factor R_V/R_F
0.1	INTELSAT	.3	8	0.77	2.56
			4	0.74	2.47
			2	0.51	1.71

The following points arise from considering table 5.3 and also comparing it with table 5.1.

(i) Compared with table 5.1 there are no large improvement figures, because the worst value of p is lower in this case.

(ii) The largest improvements occur in the Intelsat case, because of the high proportion of time spent in state 8, and because state 1 conditions still have to be coped with.

(iii) The lowest improvements occur in the Tropo case because states 1 and 2 never occur, so that the FR system rates are correspondingly higher.

In order to have a still more realistic assessment of VR performance it is necessary to change and standardise the analysis in the following ways.

(i) The channel is the BSC with the value of p governed by a continuous log-normal distribution about the long term mean, instead of the discrete p variation so far used.

(ii) The VR system uses eight or less codes of differing rates all of which meet the bound of equation 5.8. Given a certain level of reliability, the system automatically selects the highest rate code which still equals or exceeds

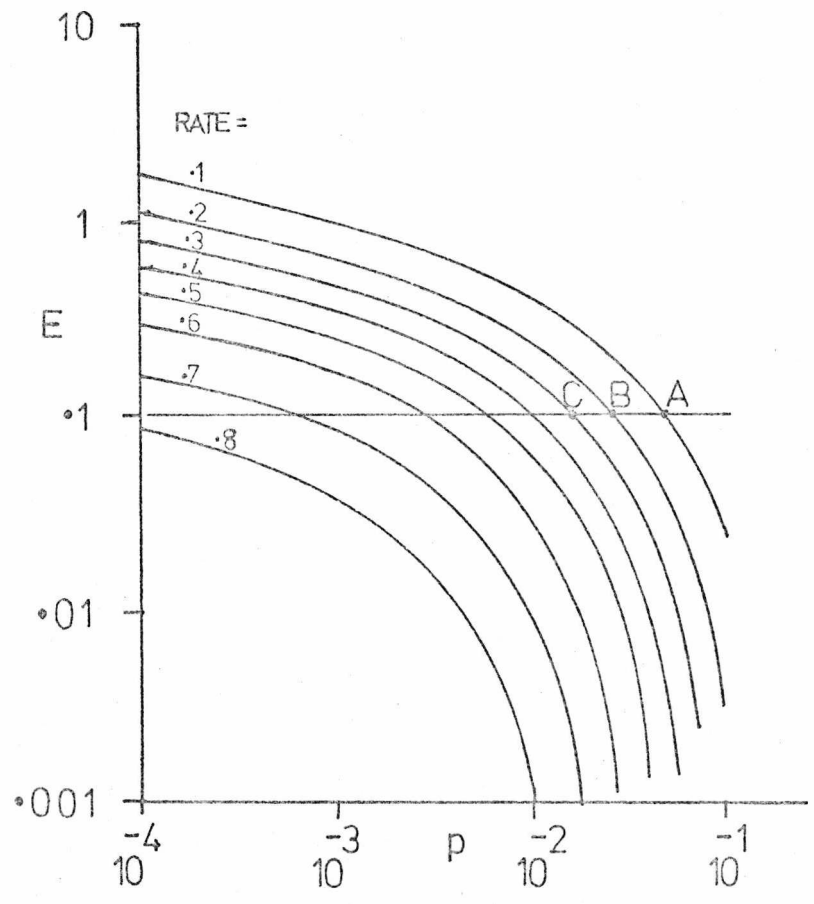
this level. The proportion of time for which a particular code is in use is calculated by measuring the proportion of time between the p values at which the code switches in and out.

(iii) Because all values of p are possible with a continuous distribution there is a certain probability, with the highest power code in use, of exceeding any specified level of reliability. The guarantee to the user must therefore take the more realistic form of being greater than or equal to some reliability level for a certain proportion of the time.

Figure 5.3 shows 8 codes that meet the bound of equation 5.8, and plots reliability versus p for each.

Figure 5.3

FEC Code Performance assuming the VGS Bound



VR performance can then be calculated as in the following example.

(i) Assume a required reliability level of 0.1, and a long term mean BER of 10^{-3} with the short term mean (p) distributed about this value log-normally with s.d.=1.

(ii) The proportion of time that p is to the right of point A (the intersection of most powerful code and E-level) determines the reliability guarantee. This value (from tables of the normal distribution) is 0.041, and the system therefore guarantees that $E \geq 0.1$ for 95.9% of the time.

(iii) The proportion of time each code is in operation is then determined from tables. For instance, the 0.1 rate code is in use for the amount of time for which p is between A and B; at B the 0.2 rate code switches in and is in use between B and C; and so on.

(iv) The long term VR system rate is then calculated by summing the proportional rate contributions from each code.

(v) The equivalent FR system, that is, the one that provides the same guarantee would use the 0.1 rate code all the time. An FR system could use one of the other codes but would provide a weaker guarantee, for example, the 0.2 rate code, used all the time, would guarantee the 0.1 E-level for only 94% of the time.

Figures 5.4, 5.5 and 5.6 present the results of several such performance calculations. Each graph shows:

(i) VR system rate (R_V) versus long term BER (P_L).

Figure 5.4

VR Performance for Different Numbers of Codes

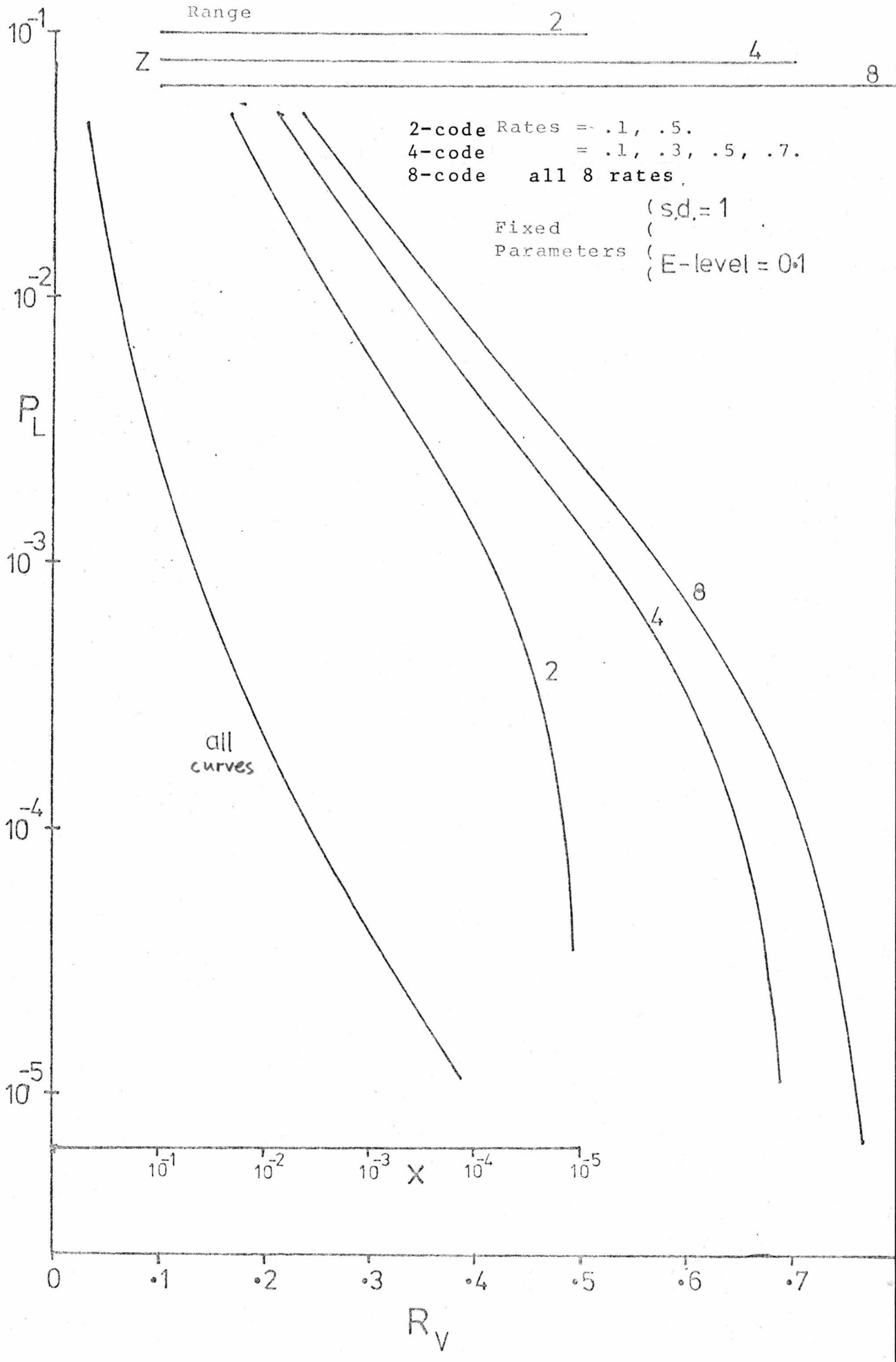


Figure 5.5

VR Performance for Different Standard Deviations

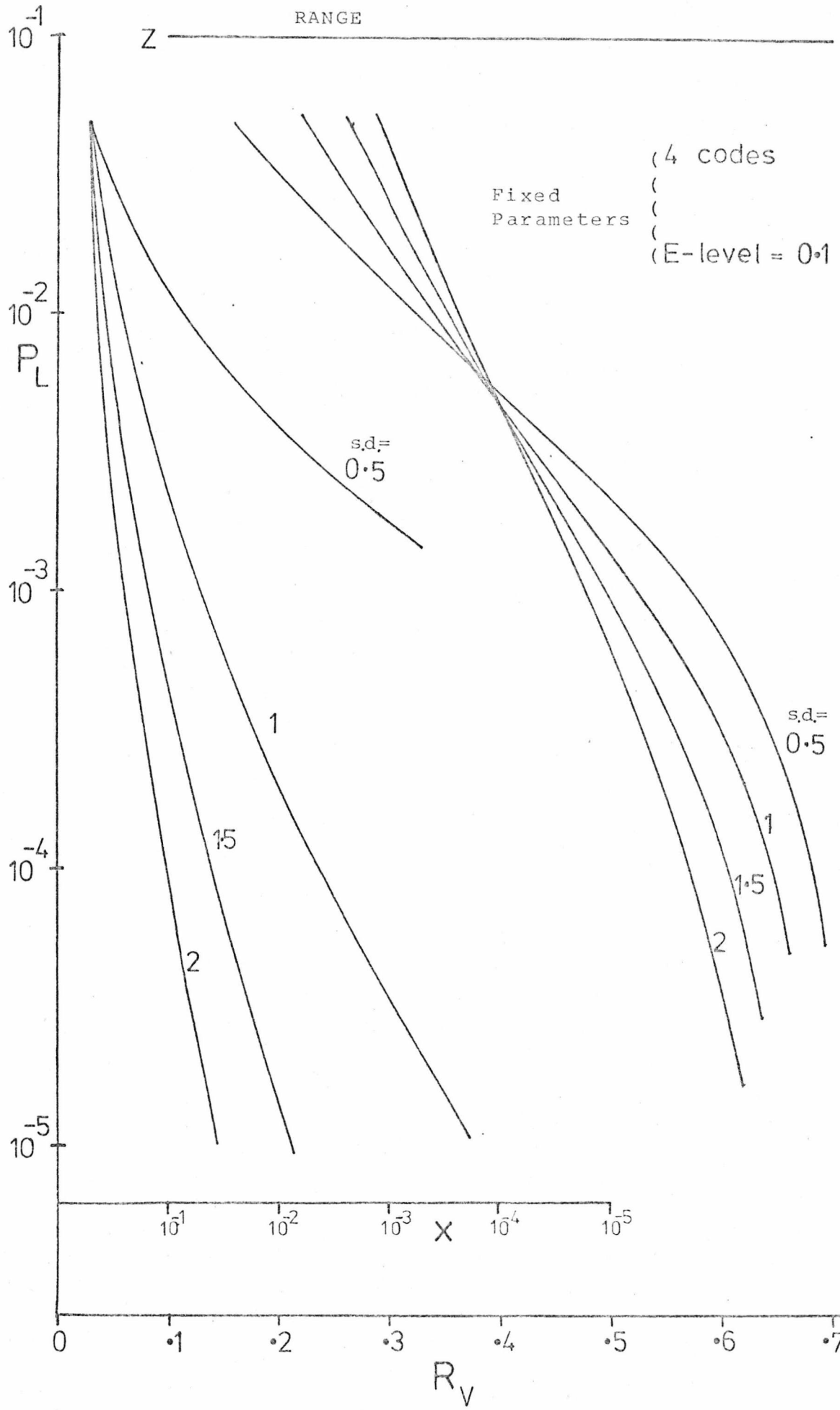
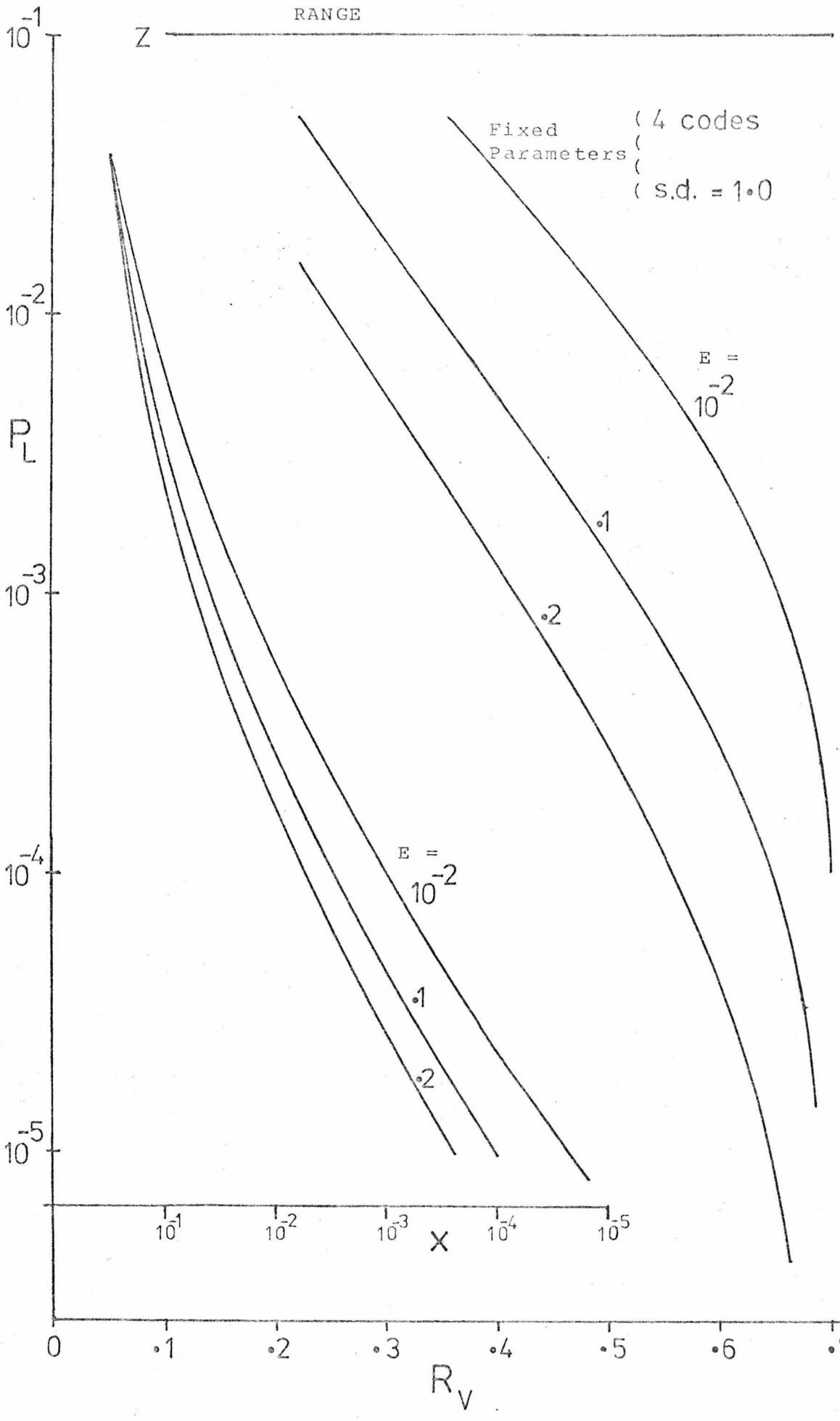


Figure 5.6

VR Performance for Different E Levels



(ii) The proportion of time for which the guaranteed E-level is exceeded (x) versus P_L (left hand curves).

(iii) The range (Z) of VR system rate (lines at top); the lowest is 0.1 corresponding to the FR system with the same guarantee, and the highest is equal to the highest rate VR code.

(iv) The equivalent FR system rate, which is 0.1 for all values of P_L.

We note the following points about these graphs.

(i) Figure 5.4. No matter how many codes (numbers beside curves) are used in the VR system the guarantee curve is the same. This is because they all use the same lowest-rate code (.1). The difference between 4-code (rates .1, .3, .5, .7) and 8-code performance is less than that between 2-code (rates .1, .5) and 4-code, particularly at low values of P_L; at high values of P_L the low rate codes are in use most of the time and the difference is small. The higher the number of codes the lower P_L must be for R_V to reach its maximum value.

(ii) Figure 5.5. In this case varying the s.d. (numbers beside curves) varies the guarantee substantially but does not affect the performance curves so much. The more peaked the distribution about P_L (s.d.=.5) the more rapidly high guarantee levels are reached. We also note that the curves cross over at about $P_L=5\times10^{-3}$, so that the lowest s.d. performs worst below this value, and best above. All the curves perform approximately the same at the crossover point. This effect is due to the spacing of the code curves shown in figure 5.3, and the crossover point corresponds to

the value of p for which the two highest rate and the two lowest rate codes are in use for an equal amount of time (giving R_V=0.4). Below (worse P_L) this point the low s.d. values tail away rapidly so that the high rate codes are used for a small proportion of the time while the high s.d. values still use these codes for a significant proportion of the time. The contrary argument applies above the crossover point.

(iii) Figure 5.6 shows that varying the E-level has a significant effect on the performance curves whilst the guarantee curves do not vary greatly. The lowest E-levels are the quickest to reach maximum R_V with decreasing P_L, and also perform best, whilst suffering only slightly degraded guarantees.

These performance curves show that, for error correction considerable improvements in throughput are theoretically achievable by using VR coding on log-normal BSC channels that have long term error rates similar to those found on real channels. In addition, these improvements are achieved whilst retaining the same guarantee as the equivalent FR system, and the absolute value of the guarantee can easily be increased by using a more powerful 'highest power' code. An important practical point that arises is that these improvements can be achieved by using only a few, and certainly no more than 8, codes in the VR system, thereby relieving the implementation problem.

5.2.2 *Error detection*

The probability of erroneous decoding for error detection on the BSC is given by equation 3.20. This can be expressed in terms of reliability, and in a manner entirely analagous to the error correction case, VR-ED performance can be computed and plotted. The performance graphs are not reproduced here but are similar to the error correction case, and show that approximately the same VR performance (rates and guarantees) is obtained for E-levels which, in the case of ED, are about an order of magnitude greater.

5.2.3 *Burst error correction*

It is first necessary to consider what is meant by a burst VR system. A loose definition would be to assume that the system automatically adjusts the code redundancy to cope with the prevailing 'burstyness' of the channel. Increased 'burstyness' is most often taken to mean increased length of burst, and the VR system would therefore vary the burst correcting power to suit the prevailing length of burst.

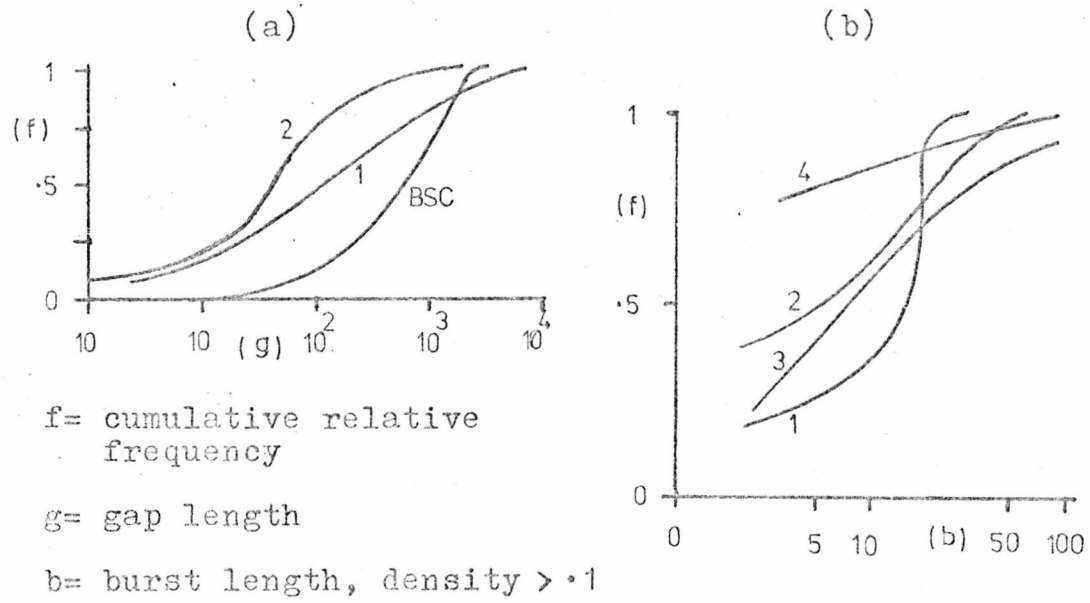
Unfortunately, the burst distributions of most real channels, and particularly the HF channel, do not have a simple dependence on bit error rate, and are extremely variable as to how 'burst-like' or 'random-like' particular runs are. This rules out a meaningful general analysis of the type carried out for error correction.

Assuming that the VR system adjusts redundancy to cope with different burst lengths it is possible to obtain a

crude measure of VR performance. Figure 5.7 shows data taken from Brayer and Cardinale (1967) for the burst and gap distributions of several runs over the HF channel.

Figure 5.7

HF Error Statistics



f= cumulative relative frequency

g= gap length

b= burst length, density > ·1

Runs 2,3 and 4 are classified as 'burst-like', whilst run 1 is 'random-like'. This is decided by measuring the disparity between the gap distribution of the run and the gap distribution of a BSC with an equal error rate (Figure 5.7(a)).

To calculate performance the following assumptions are made.

(i) The VR system uses 4 codes with burst correcting powers b=10, 20, 50 and 100 bits.

(ii) The codes have length n=400 and meet the Reiger bound, that is b=(n-k)/2, and therefore have rates R=0.95, 0.9, 0.75, 0.5.

(iii) The burst length on the channel changes slowly, and the system knows which burst length prevails and can therefore select the correct code at any particular time.

Table 5.4 below shows the proportion of time each code is in operation and the resulting VR rate.

Table 5.4

Run No.	% of bursts corrected	FR rate (R_F)	Proportion of time codes in use .5	.75	.9	.95	VR rate (R_V)	Improvement factor (F)
1	100	0.5	.01	.14	.65	.20	0.89	1.77
2	98	0.5	.03	.15	.18	.62	0.88	1.76
3	95	0.5	.07	.18	.25	.55	0.92	1.84
4	99	0.5	.04	.05	.05	.85	0.91	1.82

These results show that throughput improvements are possible; but the shakiness of assumption (iii) renders dubious any further analysis along these lines. Rather, the results indicate that the more 'burst-like' a particular run is the more the VR system should tend to a 2-code system that detects only 'good' and 'bad' conditions, and then uses the appropriate 'low' or 'high' redundancy code.

Rather than continue with any general analysis, therefore, specific VR results under burst conditions are presented later (Ch.7).

5.3 *The basic VR system and modes of operation*

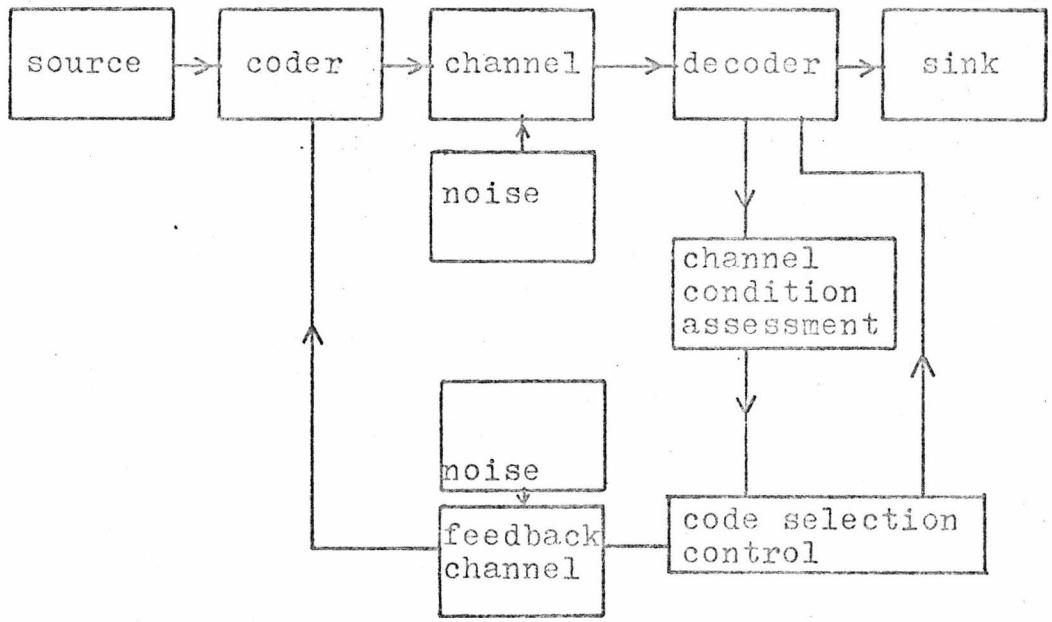
In this section the basic VR system is described, together with the ways of operating such a system under various error control modes. The problems that arise in the design of a VR system are brought out, so that they can be considered in later sections.

A basic simplex (one-way) VR system is shown in

figure 5.8 below.

Figure 5.8

A Simplex VR System



In the diagram, all modems are lumped into the 'channels' and the adaptive process is solely concerned with the coding scheme. In addition, a start-stop (off-line) data source is assumed, and the data transmission system in use may be synchronous or asynchronous.

The system operates as follows. Initially, after system start-up procedures, the code in use should be the highest redundancy one, for safety. The prevailing channel conditions, or the suitability of the present code to the prevailing channel conditions is then continuously monitored by operations on the decoder, and received and decoded bit streams only. If the present code does not match these conditions, according to some predetermined criterion, the code selection control initiates a request for a code change which passes to the coder via the feedback link. The

system continues automatically selecting the (hopefully) most suitable code for the prevailing conditions.

The basic VR system of figure 5.8 can be considered to operate under any of the three main modes of error control, ED, ARQ or FEC, to provide three main modes of adaptive error control: VR-ED, VR-ARQ and VR-FEC.

(i) VR-ED uses error detection codes thereby considerably simplifying decoder equipment. In addition, the feedback link only conveys code control signals and can therefore have a smaller bandwidth than the forward channel.

(ii) VR-ARQ uses the feedback link to request repeats of erroneous blocks as well as to convey code control signals. This would seem to require significantly more feedback bandwidth (or time for duplex) than FR-ARQ, but it may be possible to reduce this requirement by combining RQ and code control signals. For example, if recursive ARQ is used (section 2.2), an acknowledgement of 'incorrect' (i.e. repeat) may also be used to mean 'increase redundancy' to the code control equipment. Although decoder complexity is low for VR-ARQ, as detection codes only are used, the overall system complexity may be high, because of both code changes and repeats occuring simultaneously, particularly in the areas of timing and data bufferage.

(iii) VR-FEC control uses forward error correcting codes with consequent high decoder complexity, but the feedback bandwidth requirement is again low as only code control signals are conveyed. An extension of this mode would be to have

VR-FEC with no feedback link. This would be possible (and desirable for military systems) if the coder could have available reasonably accurate and speedily updated channel information (e.g. from its own soundings). The coder would then be in control of code selection and the receiver would have to decide from the incoming bit stream which code was in use.

(iv) Hybrid VR systems are also possible. In this case operation would be mainly VR-FEC with overload error detection. Detectable but uncorrectable errors could then be corrected by ARQ if required, and the number of times the overload redundancy was used could give an indication of the degree of 'match' between code and channel.

The design of a *particular* VR system is influenced at every stage by the mode of error control that must be provided, and by the type of forward channel in use. The general areas to be considered in the design of any VR system can, however, be considered separately, and are as follows:

(i) Choice of a set of codes. The problem under consideration here is, given a particular channel and mode of error control, what is the best set of codes for the situation. This choice must be taken on grounds of not only efficiency and power but also on ease of implementation, particularly in the VR-FEC mode.

(ii) Channel information extraction. For a VR system to operate correctly, a reliable picture of the short-term state of the channel must be extracted by measuring one or more quantities. What these quantities are and how reliable

the resulting picture is have a large influence on the overall reliability of the system.

(iii) Code selection control. Given a picture of the channel state, to a certain reliability, part of the design of the VR system must be to establish criteria upon which the code change command can be based. Different criteria may produce different overall system efficiencies, and the establishment of these criteria must be considered in conjunction with (ii) above.

(iv) Feedback signalling. The choice of a particular method of feedback signalling must be considered from opposing points of view: the signalling should be fast enough not to hold up forward channel code changes; and also slow enough (or redundant enough) to be transmitted reliably. The effect of feedback errors on a VR system is to reduce the overall system reliability, and different code selection command structures can cause different amounts of misoperation if commands are received in error.

The remainder of this chapter considers possible solutions to points (i)-(iv) above in more detail, so that the reasons for choosing the particular VR experimental system (and simulations) described later, can be more fully appreciated.

5.4 *Code sets for VR systems*

This section examines the possible ways in which a set of codes of varying rate and redundancy can be chosen. The performances of different code sets, and the VR systems

that use these sets, are determined in a manner similar to section 5.2, and compared *without* reference to their ease of implementation. The values of n, k, and d used in the subsequent analyses are from Helgert and Stinaff (1973) unless otherwise stated, and where the best distance for an (n,k) code is not known the lower bound is taken.

5.4.1 *General considerations*

It is possible to make several generalisations about how the choice of codes for VR is constrained by the VR technique.

Firstly, and possibly most important of all, the codes should be relatively simple to implement. This may rule out the use of an unrelated collection of codes because of the undesirable duplication of encoders and decoders. Rather, the set of codes should be related and possibly all belong to one class, so that implementation would require only one basic encoding or decoding circuit. A particular code in the set would then be encoded or decoded by only slight modifications to the basic circuit. In the case of VR-FEC, decoder complexity may override all other considerations.

Secondly, the block length should not be long (say $n \leq 50$) for the following reasons:

(i) Short codes are easier to implement.

(ii) Channel conditions must not change appreciably over several blocks.

(iii) Channel information is available after each block decoding, and should be updated as fast as possible.

(iv) The delay in changing to another code in the set should be as small as possible, so that waiting for long blocks to finish is not necessary.

(v) Optimum short codes, both burst and random, are easier to find.

(vi) If interleaved codes are used in order to improve the burst capability of the system, the effective block length is increased; therefore requiring a short original code.

(vii) ARQ is inefficient if long blocks are used.

The maximum size of n is therefore taken as 127 in subsequent examples.

Finally, the set of codes should be as efficient as possible, particularly the high rate codes, and have a wide range of minimum distance.

Types of code set can be split into two main classes: those with variable block length (variable n); and those with constant block length. Inaddition, code sets with variable block length can be further sub-divided into sets with k constant, d constant or (n-k) constant.

Before examining the theoretical capabilities of these sets it is necessary to point out that variable block length sets have several important system disadvantages.

Firstly, any VR system with variable n codes will experience block synchronisation difficulties. This is further aggrevated by the (perhaps high) possibility of feedback link errors causing the encoder and decoder to operate with

different codes, and hence different block lengths. Detecting and rectifying such a situation would certainly require further complexity and/or redundancy, and may not be feasible.

Secondly, if channel information is updated after every block decoding, the time taken to assess the performance of the code in use is variable, as is the delay before changing to a new code.

Finally, the range of block size, and hence EDC power, is limited by the requirement for relatively short blocks.

5.4.2 *Constant k code sets*

A set of variable n, constant k codes is characterised by having the shortest codes as the least powerful (highest rate). Because the longest codes are the most powerful, decoding complexity would be high for FEC. The range of EDC power is limited by the requirement for short blocks at the high power end, and by the value of k at the low power end.

Table 5.5 shows the extreme end codes of several constant k code sets, and gives values of rate (k/n), relative random EDC power (d/n) and range of rate and d/n.

An examination of table 5.5 verifies that the widest ranges of both d/n and k/n occur when k is in the region of 10 to 20. If k is low, eg. 4, it is possible to have very high power codes (thereby ensuring high guarantees) but at the expense of having poor efficiencies for the highest rate codes. If k is large, on the other hand, high-efficiency

Table 5.5

Constant k Codes

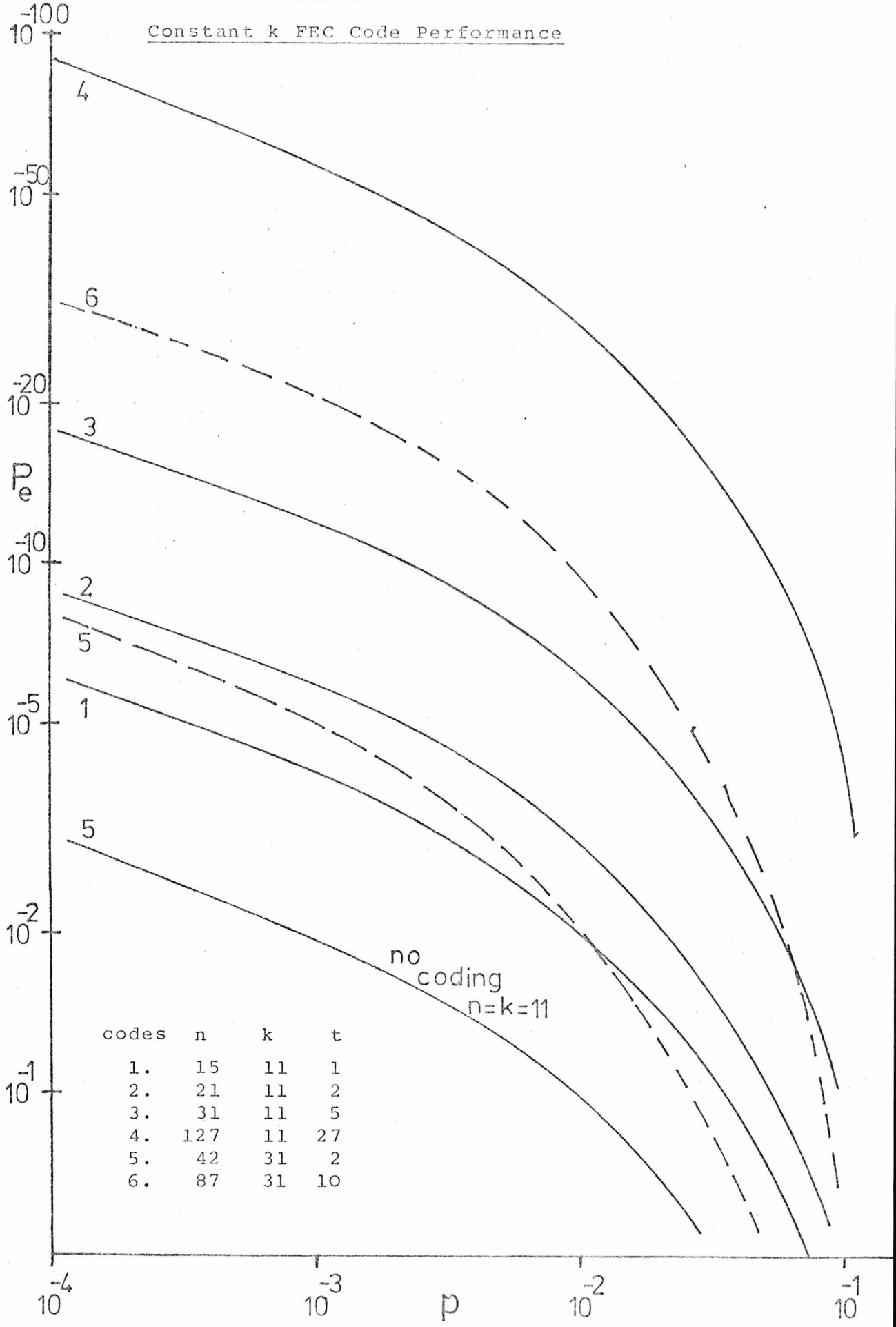
n	k	d	d/n	k/n	range
5	4	2	.4	.8	d/n:0.13
127	4	67	.53	.031	k/n:0.77
12	11	2	.17	.92	d/n:0.27
127	11	56	.44	.087	k/n:0.83
16	15	2	.13	.94	d/n:0.3
127	15	55	.43	.12	k/n:0.82
32	31	2	.063	.97	d/n:0.19
127	31	32	.25	.24	k/n:0.73

high-rate codes are possible but the highest power codes are restricted by the limitation $n \leq 127$.

(a) error correction: figure 5.9 plots P_e versus p for four of the k=11 codes. Also shown are the curves for two k=31 codes (curves 5 and 6) which have approximately the same rates as codes 1 and 3 respectively. Figure 5.9 shows that, for high efficiency codes of the same rate, the longer codes of the k=31 set perform better at low BER, and worst at moderately high BER, than the k=11 codes (1 and 5). For low efficiency codes, however, the longer codes out-perform the shorter ones until very high p values are reached (3 and 6). The use of code sets with high k would therefore be advantageous for VR systems if it were not for the fact that the highest power codes are then severely limited by the $n \leq 127$ requirement, so that guarantees are worse for high k sets than low k sets. Assuming that in a VR system the performance fall-off of the high rate codes at high BER is not important because these codes are not used at high BER, the best method of

Figure 5.9

Constant k FEC Code Performance



codes	n	k	t
1.	15	11	1
2.	21	11	2
3.	31	11	5
4.	127	11	27
5.	42	31	2
6.	87	31	10

choosing a constant k code set for FEC would seem to be to choose the highest value of k for which some specified guarantee (performance of the highest power code) is still satisfied.

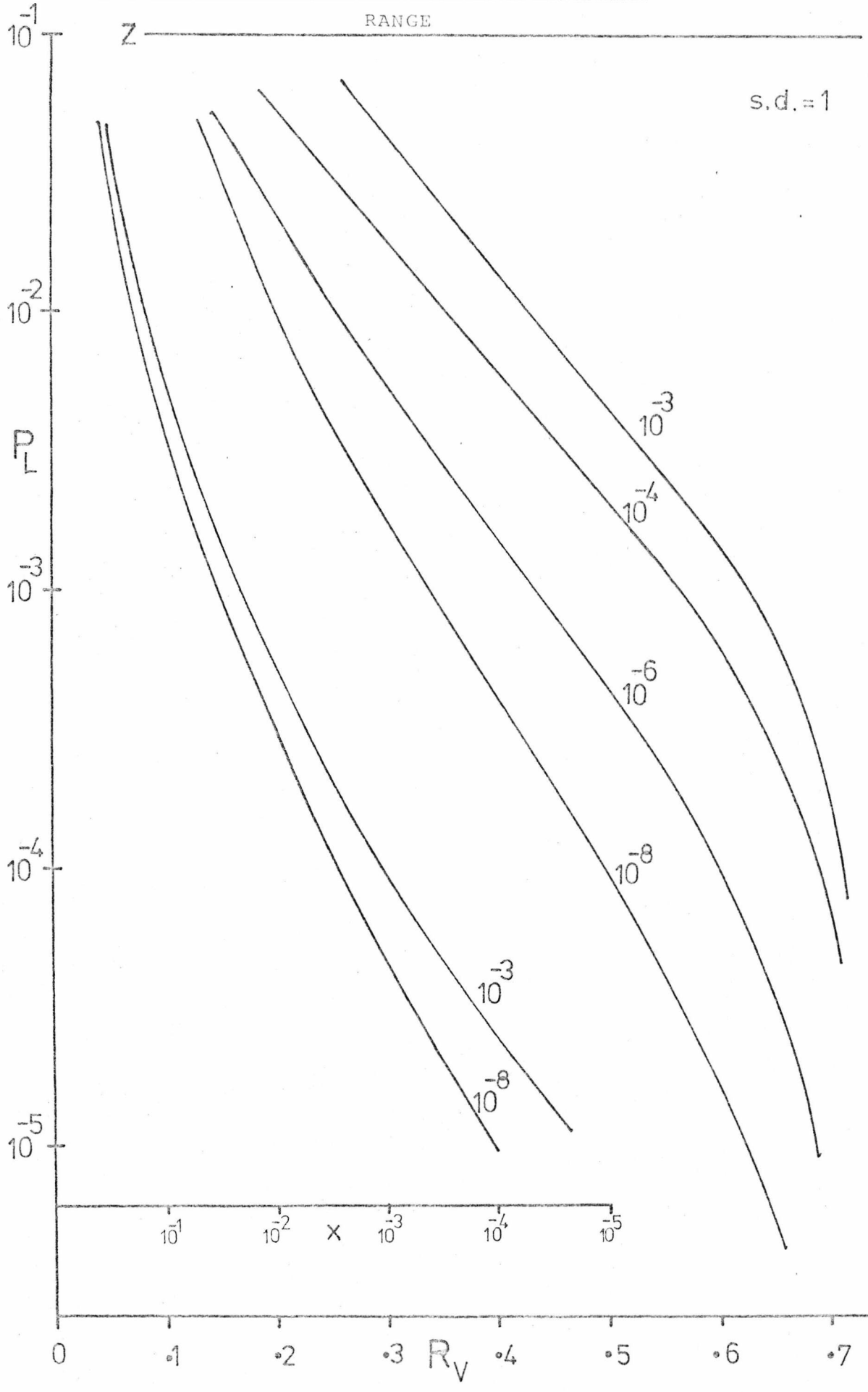
The performance of a VR system that uses the four k=11 codes of figure 5.9 on a BSC with log-normal p distribution can be calculated by the same method used in section 5.2, and is shown in figure 5.10 below.

Figure 5.10 shows VR system rate (R_V) versus long term BER (P_L) for different levels of maximum desired probability of block error (P_e) (numbers beside curves). Also shown is the proportion of time that the P_e level is exceeded (x), and VR rate range (Z). The results show that significant improvements in throughput are achievable from around $P_L=10^{-3}$ (the equivalent FR system rate is 0.087) and that the guarantee does not vary greatly with P_e level for P_e levels of interest. Comparison with figure 5.6 shows that guarantees are similar, and that the $P_e=10^{-3}$ curve corresponds to the $E=10^{-2}$ curve. Given a code set with higher k one would expect better VR performance but with reduced guarantees (increased x).

(b) error detection: when used for ED, constant k codes can have a much higher rate than when used for FEC, whilst still achieving the same P_e levels and guarantees. In particular, the 'most powerful' code in the VR set (which is also the equivalent FR system code) can have a much higher rate than in the FEC case, thereby reducing the range of R_V,

Figure 5.10

Constant k=11 4-code VR-FEC Performance



and the relative improvement of VR over FR at any particular P_L value. VR-ED is therefore characterised by high rates and guarantees, with low VR/FR improvement.

(c) burst error correction: burst *performance* is difficult to assess analytically, and is not very meaningful because of the variables involved in choosing a burst channel model. Performance will therefore not be considered here. Rather, we consider the code sets from the point of view of burst correcting *ability* (b), and the proportion of correctable burst patterns (N_c).

For constant k codes the most powerful are also the longest, this goes against 'common sense' in that if a channel becomes very 'bursty' the most effective way to deal with it is to shorten the block length. In addition, good optimum short burst codes are known, and optimum long codes can be constructed from these by interleaving.

Table 5.6 below compares the proportion of correctable patterns: $N_c = \left(n + \sum_{i=2}^{b} (n-i+1)2^{i-2}\right)/2^n$ 5.9

for several code sets. The value of b, if not exactly known, is taken as the mean of b' and b": where $b' \leq (n-k)/2$ is the Reiger upper bound on b, and $b'' \geq (3d-8)/4$ is a lower bound on b for cyclic codes given by Peterson (1972). Table 5.6 verifies that the most powerful codes have a very poor burst performance, particularly for large k. In general we may therefore conclude that constant k codes would exhibit poor performance if used on real channels.

Table 5.6

Constant k Burst Capability

n	k	k/n	b	Proportion N_c
15	11	.73	1	4.6×10^{-4}
31	11	.36	9	2.9×10^{-6}
63	11	.18	22	9.7×10^{-12}
37	31	.84	1	2.7×10^{-10}
42	31	.74	4	7.3×10^{-11}
56	31	.55	9	1.7×10^{-13}
7	4	.57	1	5.5×10^{-2}
20	4	.2	7	9.2×10^{-4}

5.4.3 *Code sets with d, t, or b constant*

These code sets are characterised by having powerful short codes and weak but efficient long codes. The advantages of this scheme over the constant k scheme are that short powerful codes are easy to find, and that the amount of decoding effort increase only slowly with n (because n-k increases slowly with n). The problem of finding long efficient codes for the scheme can be circumvented if the maximum code length is again restricted. In addition, for a burst VR system, it makes good sense to have the shortest codes as the most powerful. Table 5.7 below shows several constant d code sets.

Table 5.7 shows that the range of d/n values are much larger than those of table 5.5 and that they decrease with increasing k/n range. Also, the powerful k=1 codes can be

Table 5.7

Constant d Codes

n	k	d	d/n	k/n	range
3	1	3	1	.33	
15	11	3	.2	.73	d/n:0.98
127	120	3	.024	.95	k/n:0.62
5	1	5	1	.2	
31	21	5	.16	.68	d/n:0.96
127	113	5	.039	.89	k/n:0.69
7	1	7	1	.14	
63	46	7	.11	.73	d/n:0.94
127	109	7	.056	.86	k/n:0.72

used for any scheme, enabling correction of approximately n/2 errors in a short block, instead of n/4 errors in a long block, as with the constant k schemes.

(a) error correction: figure 5.11 shows code performance for several constant t FEC codes, and illustrates the following points.

(i) High t codes outperform equal rate lower t codes except at high BER (curves 2 and 5).

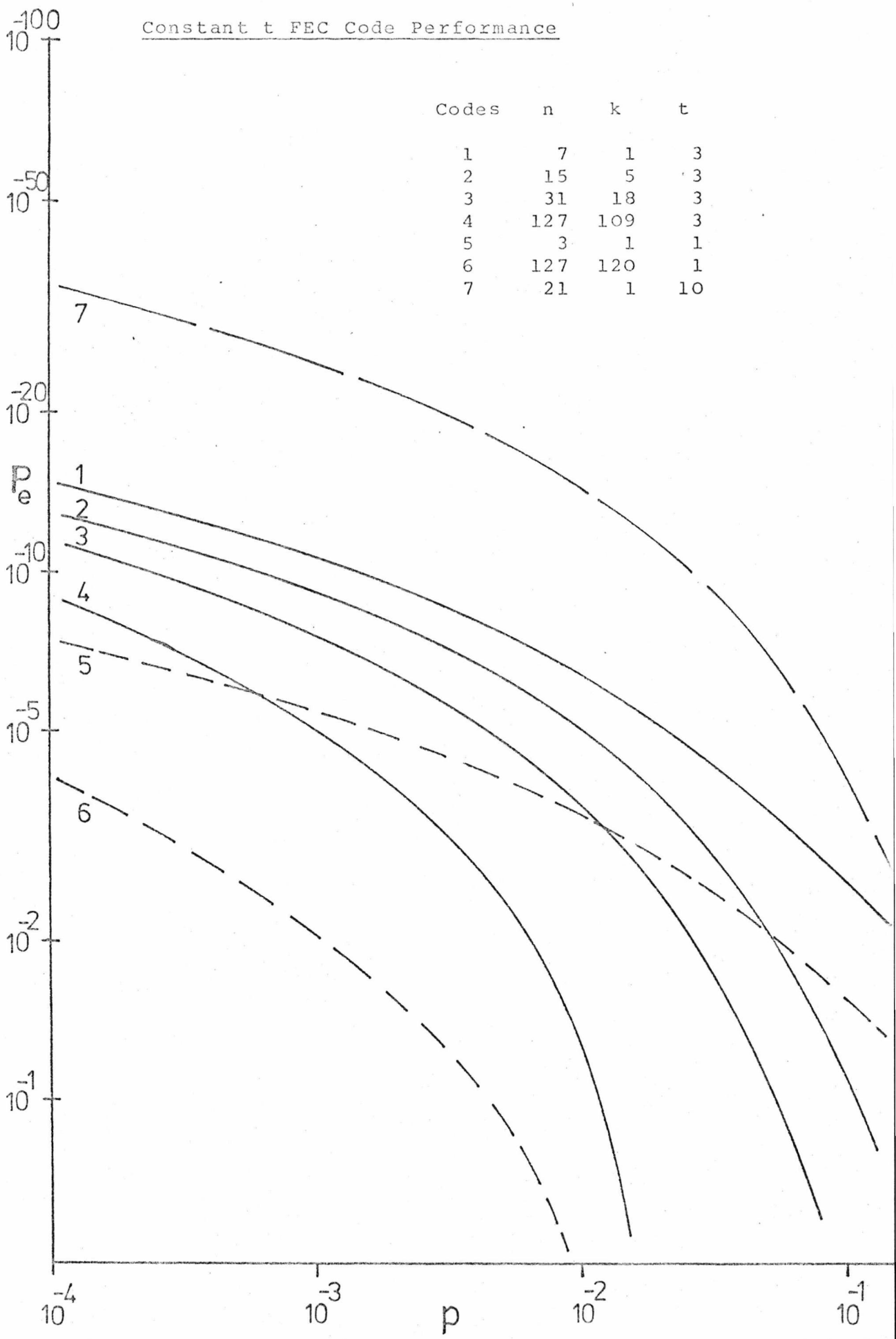
(ii) The convergence of the curves at low BER means that, for a VR system operating at a particular P_e level, the codes are all used up over a small range of p. This range diminishes with decreasing p and P_e, thereby bringing high efficiency codes into operation more quickly.

(iii) The most powerful codes (k=1) do not give good guarantees if t is low (curves 5 and 1). Improving the guarantee by increasing t (curve 7) decreases the efficiency of the maximum efficiency code because of $n \leq 127$, and also increases the decoding complexity.

Figure 5.11

Constant t FEC Code Performance

Codes	n	k	t
1	7	1	3
2	15	5	3
3	31	18	3
4	127	109	3
5	3	1	1
6	127	120	1
7	21	1	10



It can be concluded that constant t codes on a VR system can only be effective if t is large, in order to provide good guarantees ; and if the block length of the least powerful code is considerably larger than 127, in order to provide high efficiency.

Figure 5.12 plots the performance of a VR system using the four t=3 codes of figure 5.11 on a BSC with log-normal p distribution, for different P_e levels (numbers beside curves).

A comparison of figures 5.10 and 5.12 shows:

(i) The proportion of time that P_e levels are exceeded (x) is much greater in the constant t case (i.e. lower guarantees), and the spread of these curves is greater.

(ii) VR rates are much higher in the constant t case.

These constant t schemes will therefore provide higher rates but at lower guarantees, or higher rates with the same guarantees but at lower P_L values. They can therefore only be considered to outperform constant k codes at low P_e values.

(b) error detection: Figure 5.13 below plots ED code performance versus p for several constant d codes. Again it can be seen that the curves bunch together and do not provide a wide variation of P_e versus p.

The performance of a VR-ED system will therefore be similar to that of the FEC system, with low guarantees,

Figure 5.12

Constant t=3 4-code VR-FEC Performance

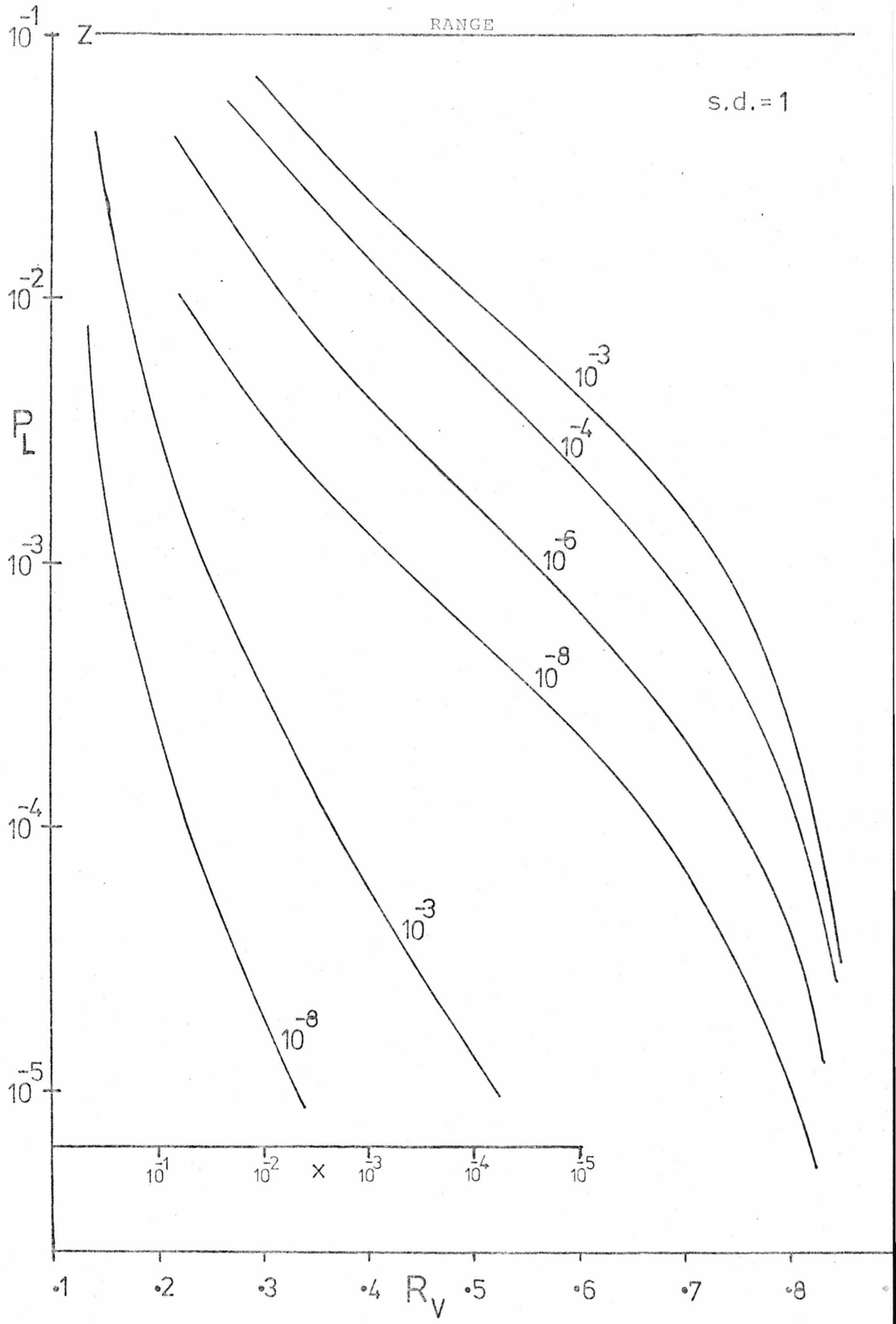
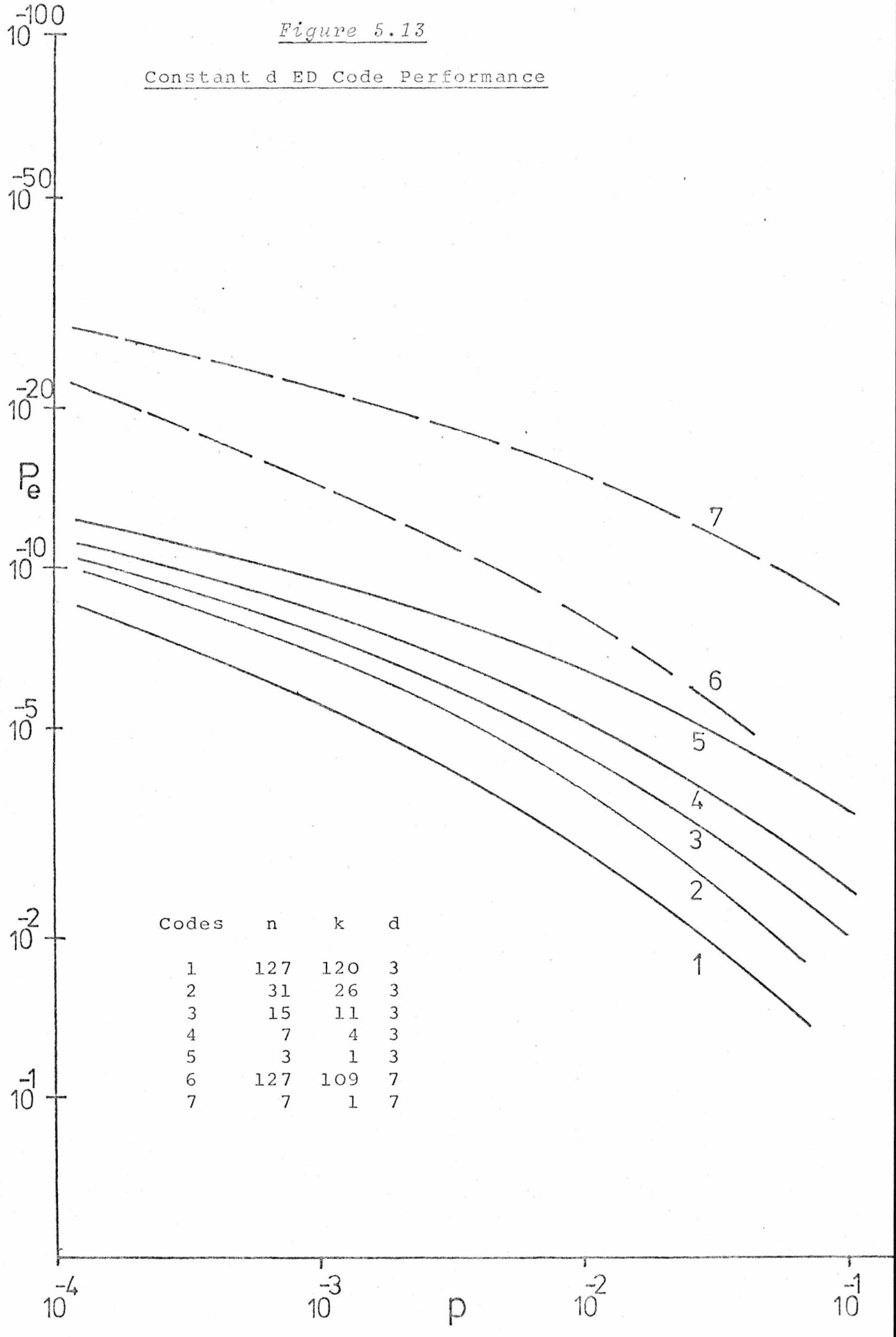


Figure 5.13

Constant d ED Code Performance



Codes	n	k	d
1	127	120	3
2	31	26	3
3	15	11	3
4	7	4	3
5	3	1	3
6	127	109	7
7	7	1	7

and high VR rates at low BER. A further consequence of the bunching of the curves is that there is very little difference between two-code VR and VR using a greater number of codes.

(c) burst error correction: for burst-VR it is appropriate to consider the system as using constant b (rather than d) codes. The code set would now be operating appropriately (compared to constant k) in that the shortest codes are the least efficient and have to cope with the worst burst conditions. Table 5.8 below lists some burst correcting codes (Lin 1970).

Table 5.8

Constant b Code Sets

n	k	k/n	b	N_c
7	3	.43	2	0.102
15	10	.667	2	8.85×10^{-4}
31	25	.81	2	2.84×10^{-8}
63	56	.89	2	1.36×10^{-17}
15	9	.6	3	1.68×10^{-3}
27	20	.74	3	7.67×10^{-7}
63	55	.87	3	2.68×10^{-17}
121	112	.93	3	1.8×10^{-34}
19	11	.58	4	2.58×10^{-4}
38	29	.76	4	1.04×10^{-9}
85	75	.88	4	1.71×10^{-23}
164	133	.93	4	5.54×10^{-47}

Table 5.8 indicates that low b sets would perform best under the worst burst conditions (because of the relatively shorter block length) and that the high b sets would be most efficient when conditions are good.

5.4.4 *Code sets with (n-k) constant*

These code sets are similar to the sets of the last section in that the most powerful codes are the shortest. The efficiency of the most efficient code for a given value of n-k is limited by $t \geq 1$ and $n \leq 127$, as shown in table 5.9 below.

Table 5.9

Constant n-k Codes

n	k	n-k	t	d	d/n	k/n	range
12	1	11	5	12	1	.083	d/n:0.9
40	29	11	1	4	.1	.73	k/n:0.65
22	1	21	10	22	1	.046	d/n:0.92
85	64	21	3	7	.082	.75	k/n:0.7
36	1	35	17	35	1	.028	d/n:0.91
127	92	35	5	11	.086	.72	k/n:0.69

It is again possible to use (n,1) codes, and because values of n can be larger in this case without restricting the range of rates available, guarantees can be higher than in the constant d or t cases.

(a) error correction: when compared with constant t sets, code performance curves in this case do not converge as much. This gives rise to similar VR-FEC performance but with higher guarantees and lower VR rates.

(b) error detection: code performance curves again do not converge as much, and occupy a wider range of P_e than in the constant d case. VR-ED performance for a given guarantee is generally superior in this case, in terms of P_e levels and rates.

(c) burst error correction: for codes that meet the Reiger bound, constant n-k implies constant b; however, for a practical set of codes, b decreases as n increases with n-k constant, implying slightly inferior burst capability in this case.

5.4.5 *Constant n code sets*

Constant n codes are advantageous for reasons of timing and ARQ simplicity, and were used in the experimental system. Table 5.10 below lists several constant n codes.

Table 5.10

Constant n Codes

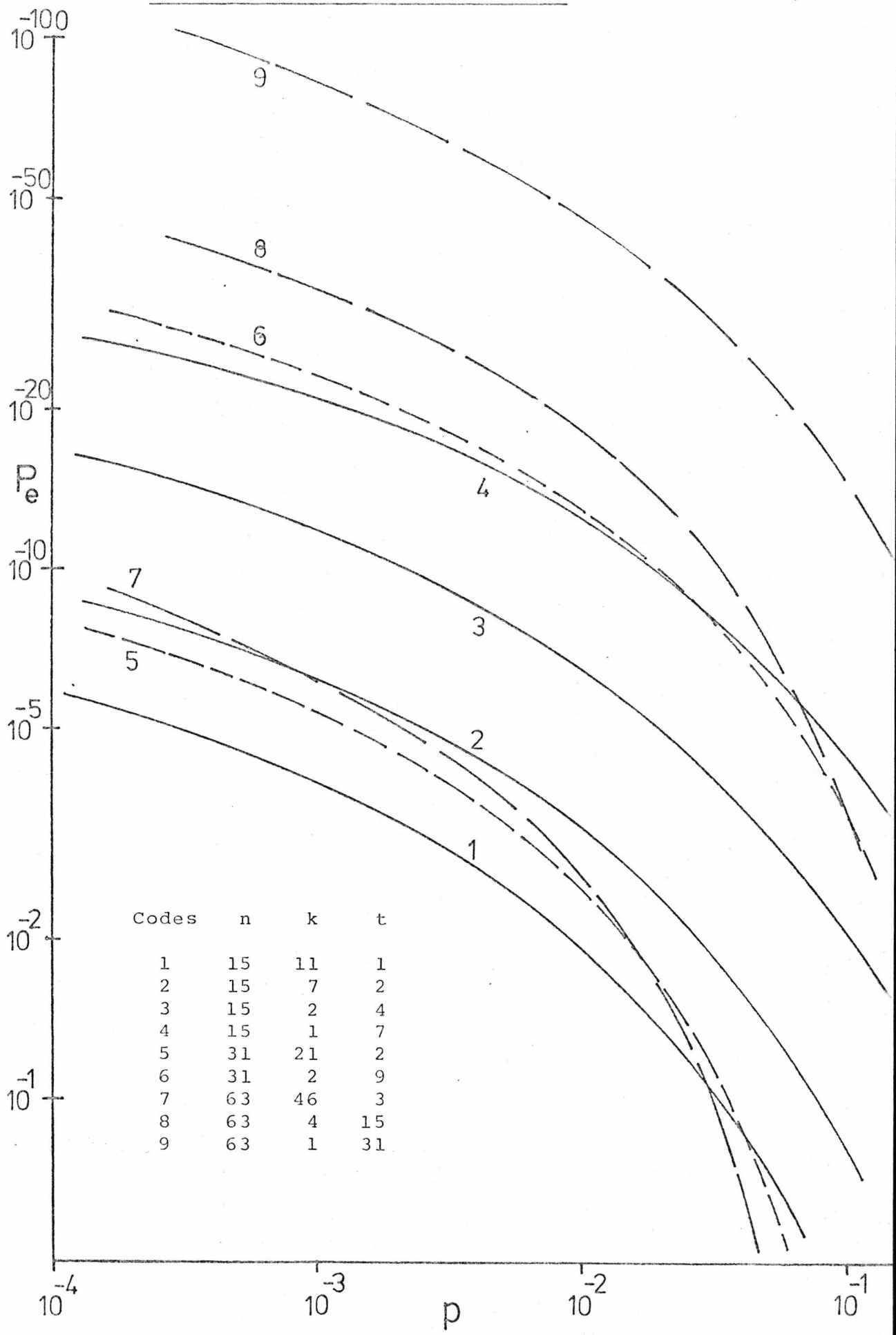
n	k	d	d/n	k/n	range
15	14	2	.13	.93	d/n:0.87
15	5	7	.47	.33	k/n:0.86
15	1	15	1	.067	
21	20	2	.095	.95	d/n:0.91
21	11	6	.29	.52	k/n:0.9
21	1	21	1	.048	
63	62	2	.032	.98	d/n:0.97
63	46	8	.127	.73	k/n:0.96
63	1	63	1	.016	

Table 5.10 shows that the constant n code sets cover a wide range of both k/n and d/n without being limited by $n \leq 127$.

(a) error correction: figure 5.14 plots the BSC performance of several constant n FEC codes. It can be seen that equal-rate higher n codes outperform lower n codes except at high BER (curves 4, 6 and 8, and 1, 5 and 7). High guarantees can be provided by using high n (curve 9)

Figure 5.14

Constant n FEC Code Performance



Codes	n	k	t
1	15	11	1
2	15	7	2
3	15	2	4
4	15	1	7
5	31	21	2
6	31	2	9
7	63	46	3
8	63	4	15
9	63	1	31

which also means that higher performance codes can be used at the high rate end of the set (c.f curves 1 and 7). Block length should therefore be as high as possible, whilst bearing in mind other system requirements; but would not need to be greater than 100 in order to provide much better guarantees than the other VR schemes considered.

Figure 5.15 plots VR performance for 4-code and 2-code (one curve only) VR using n=15. P_e levels are shown beside the curves. In comparison with the other code sets considered, constant n can be seen to provide lower VR rates with higher guarantees. This is as expected because of the high guarantees provided by the k=1 codes, and the efficiency limit of the (15,11) code. Higher n would certainly provide even better guarantees *and* VR rates.

Figure 5.16 shows 2-code VR for n=15 and n=63, when approximately equal rate codes are used (figure 5.14, codes 1 and 4, and codes 7 and 8). Guarantees are approximately the same (crossover of curves 4 and 8, figure 5.14) for these two sets when $P_e=10^{-6}$, and the n=63 case is worse above this level and better below. Curves 3 and 4 show the performance of this scheme at the $P_e=10^{-6}$ level, showing the rate advantage of high n. At the $P_e=10^{-3}$ level the rate advantage is smaller and the n=15 code provides better guarantees (curves 1 and 2). The use of the (63,1) code provides very high guarantees (curve 5), and if used with the (63,46) code (or one of even higher efficiency) would enable higher VR rates to be achieved (at higher P_e levels) than that obtainable with previous VR sets.

Figure 5.15

Constant n=15 VR-FEC Performance

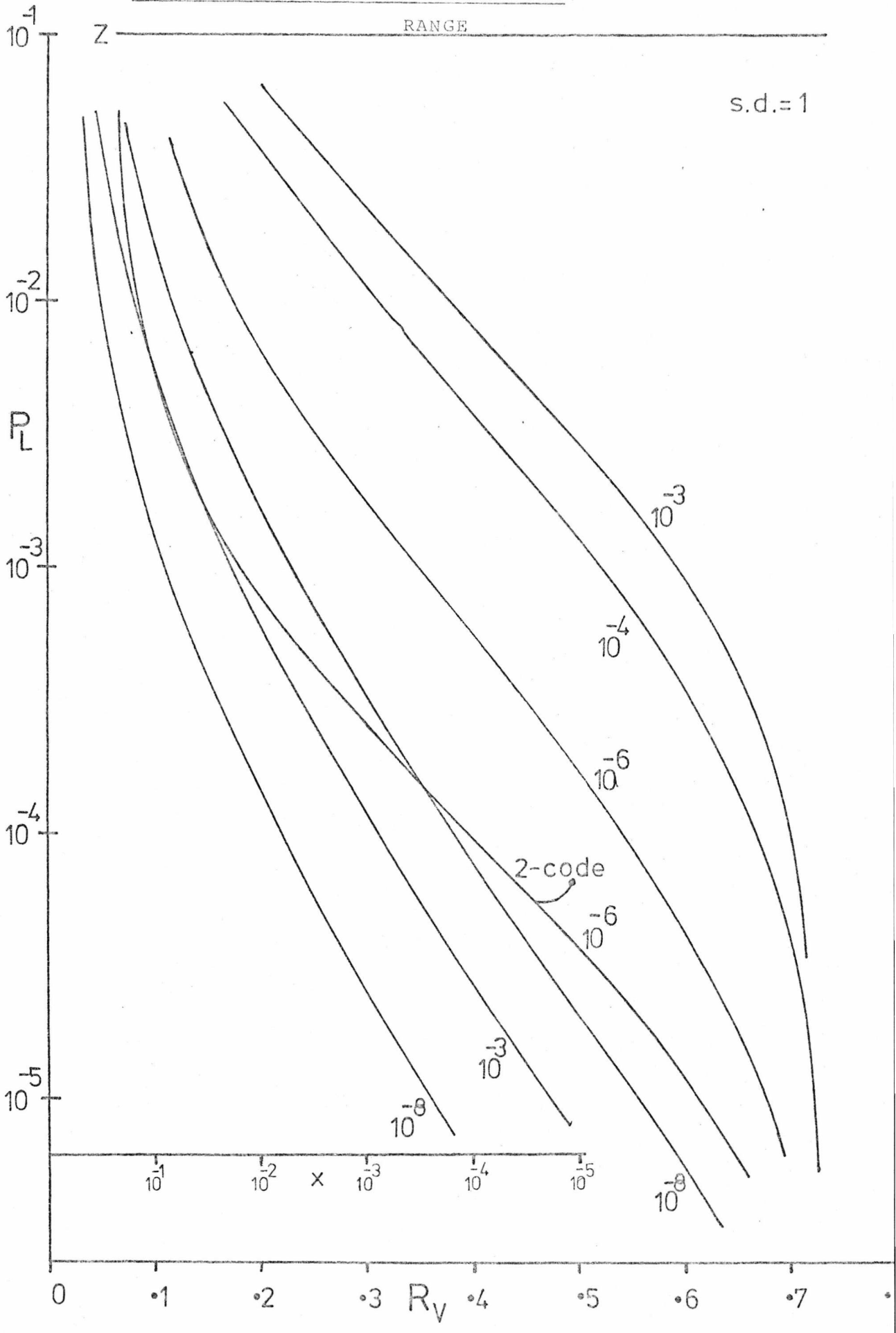
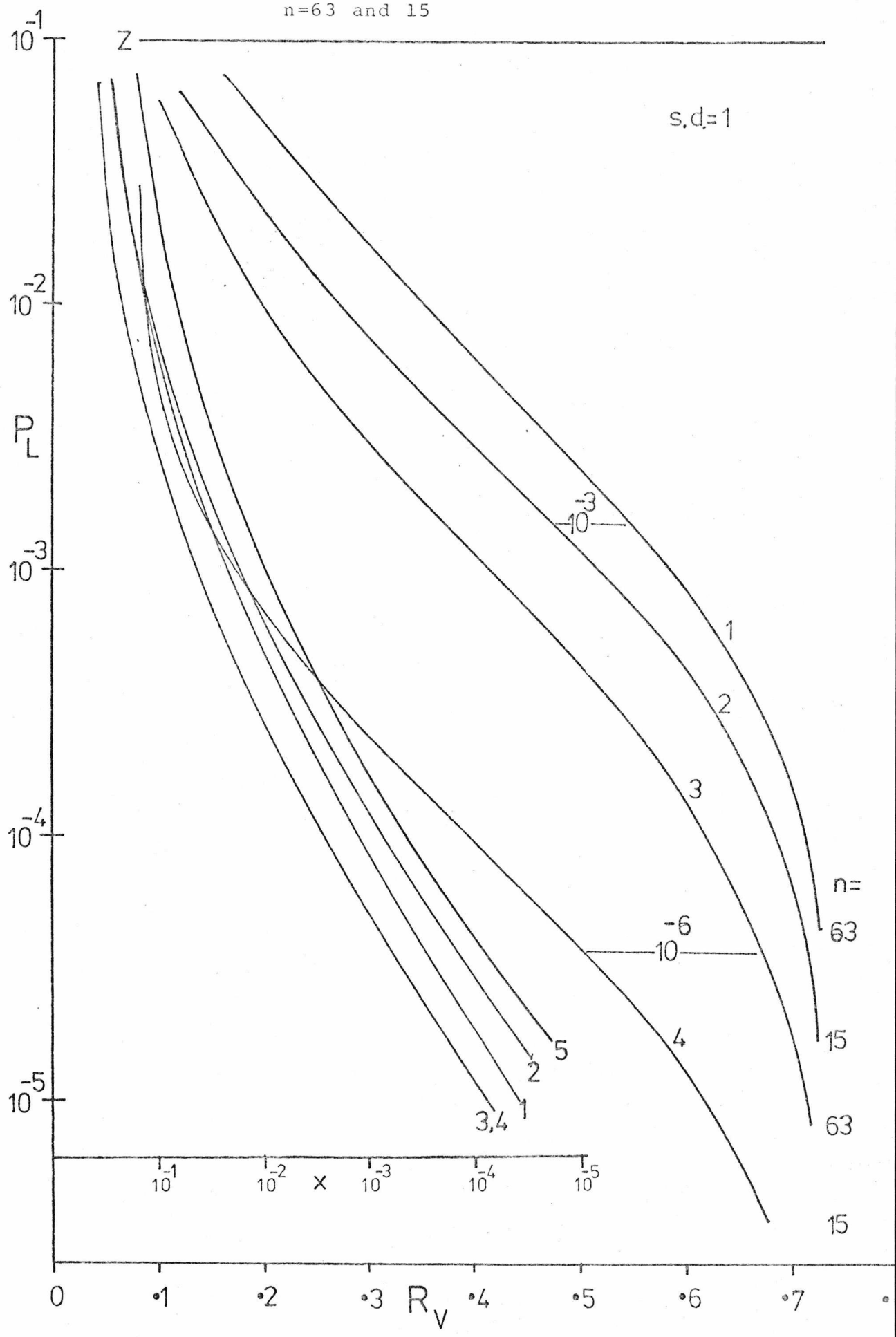


Figure 5.16

Constant n VR-FEC Performance

n=63 and 15



(b) error detection: ED code performance is again similar to FEC, except that higher P_e levels can be attained, and VR-ED performance again improves with increasing n.

(c) burst error correction: table 5.11 lists the burst capability of several constant n codes.

Table 5.11

n	k	k/n	b	N_c
15	1	.067	7	1.95×10^{-2}
15	5	.33	5	5.83×10^{-3}
15	7	.47	3	1.68×10^{-3}
15	11	.73	1	4.6×10^{-4}
31	2	.065	13	7.25×10^{-5}
31	11	.36	9	1.49×10^{-6}
31	21	.68	4	1.08×10^{-7}
63	4	25	.064	2.7×10^{-10}
63	46	6	.73	2.05×10^{-16}

Comparing the values of N_c in table 5.11 with those of previous code sets it can be seen that the range is small because n does not vary. The range widens when high n sets are taken. Because the k=1 codes meet the Reiger bound and because the high efficiency codes need not have long blocks, it is to be expected that constant n codes give good burst VR performance. Under very bad conditions, however, codes with constant b or n-k may prove superior because of their ability to have *very* short blocks at the most powerful end of the scale.

Finally it can be seen that if n is high (desirable for random error control) the burst performance of the set will suffer, and this limits the choice of n in practice.

5.4.6 *Conclusions*

On the basis of the performance calculations and system considerations so far presented, it was decided to choose constant n codes for further investigation. Some of the main considerations in this choice are as follows.

(i) Constant k codes have long blocks for the most powerful code and these perform badly when error rates are high thereby giving low guarantees.

(ii) Burst capability is poor on the most powerful constant k codes.

(iii) For a given n that must not be exceeded, a constant n system will provide better guarantees than a constant d, or constant n-k system.

(iv) Given a k=1 code which defines the guarantee for both a constant n and a constant d or (n-k) system, the constant n VR rate will be inferior because of the inferior performance of its high efficiency codes at low BER. However, the longer codes at the high efficiency end of a constant d, b or n-k system make these sets more susceptable to burst errors than the equivalent equal rate constant n codes.

The performance of code sets with n,k and d all variable can be inferred from the performance curves so far presented; but will not be considered further. This is because although such a scheme may be theoretically attractive because of flexibility in being able to choose short powerful codes, and long efficient codes, there are the practical considerations

of implementing unrelated codes to be taken into account. The practical difficulties, however, may be small if the code set contains only a few codes (say 3 or less).

5.5 *Implementation*

This section considers encoding/decoding implementation for the VR schemes described in section 5.4. The criterion of simplicity dictates that attention should be mainly restricted to the implementation of sets of cyclic codes. In addition, important coding refinements such as interleaving, soft-decision decoding, burst/random mode decoding; although applicable to any VR system, will not be dealt with here.

5.5.1 *Constant k and constant n-k codes*

Sets of these codes can be easily encoded by using only one (constant) k-stage or (n-k)-stage circuit of the form of figure 3.8 and 3.7 respectively. The only operations that must be modified by the *code control circuitry* in order to generate an individual code, are the number of shifts, and the feedback connections. The additional circuitry required to encode the whole set rather than just one code, is therefore small.

Syndrome calculation can also be simply accomplished by the use of the syndrome calculator circuit of figure 3.9 (for constant k), or the syndrome generator circuit of figure 3.10 (for constant n-k). Error detection is therefore quite easy to implement.

Error correction, on the other hand, is much more complex. Systematic search decoding can be used if k or

n-k are not too large (say <16) and would involve storage (or generation) and comparison of either the 2^k code words (constant k), or the 2^{n-k} syndromes (constant n-k). By utilising the cyclic property of the syndrome (section 3.4.7) it should be possible to use greatly increased k or (n-k). The disadvantage of this method is that a virtually different decoder is being set up for each code.

Meggitt decoding (figure 3.11) requires a variable n-stage buffer, and an (n-k)-stage syndrome generator, which for constant k, is variable. Complexity and feasibility depends on the combinatorial pattern detector network, and this would be large for constant k. Minimisation could be used to ensure that the detector for VR was less complex than the sum of the complexities of each individual code's detector, but the circuit would still be much more complex than that required to implement the most complex code in the set.

Error trapping decoding (section 3.4.8) is easier to implement in the constant (n-k) case than in the constant k case (because of the constant (n-k)-stage syndrome generator); but much more of the power of the high rate codes is lost because of the limitation that only $t' \leq t < 1/R$ errors can be corrected by this method.

The choice of a set of suitable BCH codes is severely restricted for both constant k and constant n-k. Decoding requires computations in $GF(2^m)$, and therefore each BCH code requires a virtually different decoder to be set up.

A similar problem of choice exists for majority logic

decodable codes (section 3.6). For example, if k=11 we find the (15,11) and (1023,11) EG codes, the (21,11) PG code, and the (2047,11) maximum-length-sequence code. Implementation is simpler in the constant n-k case, but is complex in both cases because of different numbers of orthogonalisation steps, which requires setting up virtually different majority gate networks for each code in the set.

Burst decoding is easily accomplished by using a decoder of the form of figure 3.15, which for constant n-k has a constant number of stages. It is also possible to choose a set of optimum or nearly optimum burst codes for n-k constant (but not for k constant); but this involves spanning a large range of n.

Constant n-k VR is therefore generally easier to implement than constant k VR, particularly in the burst case, but the implementation of a set of high-power random-error-correcting codes is complex in both cases.

5.5.2 *Constant d, t or b*

For a set of constant d codes, n-k increases only slowly with increasing n. Encoding is therefore best done by an n-k stage circuit of the form of figure 3.7. Code control complexity is somewhat increased because each code will require a different input/output point in the register, as well as different g(x)'s and numbers of shifts.

VR-FEC is simple if t=1 (the Hamming codes can all be easily error-trapping decoded with one basic circuit) and becomes more complex as t increases.

Meggitt decoding is feasible if t=2 or 3 and n is reasonably short; but the complexity of the pattern detector rapidly becomes impractical for n>50, even given that some minimisation is possible.

Systematic search decoding is limited by the largest value of n-k in the set which in turn limits its application to cases with t=2 or 3.

BCH hardware decoding is somewhat more feasible than in the previous cases because although the feedback shift registers (FSRs) used in the decoding process have variable length (m) and variable feedback connections, the multiplicity of these FSRs (t) is fixed. Software implementation is again complex because of different $GF(2^m)$ arithmetics. In general, constant t BCH VR codes sets are easy to find and feasible, but complexity is considerably more than that required for the most powerful code.

Majority logic decoding for constant t codes is best implemented by a type I decoder but the choice of a suitable code set is restricted by the majority circuits needed. Each code in a set would require a different number of majority gates, have a different number of orthogonalisation steps (L), and a varying set of check sums (approximately j^L of them), which would check different syndrome bits. Construction of the decoder control circuitry is, however, considerably simplified by noting that each gate requires exactly j inputs, because each code corrects up to j/2 errors.

A suitable set of EG codes is available for j=6 (t=3)

and is shown in table 5.12 below. These codes are the Reed-Muller eqivalents with one digit omitted.

Table 5.12

Constant t=3 Codes

n	k	t	Type	steps (L)	Majority gates
7	1	3	EG(3,2)	1	1
15	5	3	EG(4,2)	2	7
31	16	3	EG(5,2)	3	43
63	42	3	EG(6,2)	4	259
127	99	3	EG(7,2)	5	1555

Sets of constant b burst-correcting-codes can be chosen from available Reiger-efficient codes and simply decoded by error trapping. Such a decoder requires a variable (n-k)-stage syndrome generator, a variable length n-bit buffer, and a variable (n-k-b)-input OR-gate. Table 5.13 below shows such a set for b=4.

Table 5.13

Constant b=4 Codes

n	k	n-k	n-k-4	g(x)(Octal)	Reiger efficiency
19	11	8	4	1151	1.0
38	29	9	5	2151	0.89
85	75	10	6	2651	0.8
164	153	11	7	6255	0.73
511	499	12	8	10451	0.67
1023	1010	13	9	22365	0.62

Constant power code sets can be considered to be better than the previous cases when providing a set of multiple error-correcting codes, particularly with respect to the range of codes available with this restriction; but again complexity tends to be much more than that required for a single code.

5.5.3 *Constant n*

A set of constant n cyclic codes can be encoded in n shifts by using either the (n-k)-stage or the k-stage circuits previously mentioned. If k varies widely (say from 1 to n-1), approximately n stages must be provided in both cases, and any advantage of one circuit over the other would depend on the individual codes used.

Syndrome calculation can also be done with either circuit. The k-stage circuit would be advantageous for ED-ARQ because any non-zero syndrome is serially detected as quickly as possible. If a stored syndrome is required the full n shifts must be done and the (n-k)-stage circuit is to be preferred.

Meggitt decoding requires a variable (n-k)-stage syndrome generator, and a constant n bit buffer. Because n is constant the error patterns that have to be dealt with by each code are the same, even though each code corrects a different number of patterns (eg. all single errors, all single plus all doubles, etc.). This fact enables the combinational error pattern detector to be considerably simplified with the result that the complexity of this circuit for VR decoding is not very much greater than the complexity required for the most powerful code. For example, consider the (n, k, t) code set: (15, 11, 1); (15, 7, 2); (15, 5, 3). For a single error there is now only one pattern with an error in the highest position (x^{n-1}). The syndromes of all three codes when presented with this error are the same: n-k-1 zeroes and a '1' in the lowest order

position (x^0). A single AND gate will detect this, and must have a number of inputs determined by the largest n-k, that is, 10. If the unused high order stages of the variable syndrome generator are automatically held at '0' for the lower power codes, no extra code control circuitry is needed and this one gate will correct a single error in any of the three codes.

In this case the complexity required to decode a single error in any code is equal to the complexity required for the most powerful code to decode a single error. For multiple errors this is generally not true; the increase in complexity (over the most powerful code) is, however, small as we require only the extra 'code in use' inputs from the code controller, whilst having the list of syndromes from all codes makes minimisation even more effective. Meggitt decoding is therefore clearly feasible for VR if it is feasible to Meggitt decode the most powerful code in the set.

Systematic search decoding can be done by either successive generation of code words, or error patterns. If the range of k/n and t is large ($0 \rightarrow 1$, and $1 \rightarrow n/2$ respectively) the maximum value of n is limited by currently available logic to about 15 for either method. This can be approximately doubled if two decoders operate; one using code word generation decoding on the low rate codes, and the other using error pattern generation on the high rate codes.

Error trapping decoding for constant n code sets is inefficient unless n is short because much of the power of the medium to high power codes is wasted. For an error

trapping decoder to correct t' errors: t'<1/R; because every error pattern of t' or fewer errors must contain k successive zeroes. Therefore we see that although the (15,11,1) and (15,7,2) codes can be properly decoded, the (15,5,3) code has t=1/R and there is one error pattern of three errors which does not have k successive zeroes, and can not be decoded. In this case it is possible to correct this pattern with a little extra equipment. As n increases the additional equipment (or time) required to correct these 'extra' errors (over and above the guaranteed t') increases rapidly, with the result that error-trapping for even the n=31 primitive BCH codes is not really feasible.

The scope of error trapping can be extended by using permutation decoding (MacWilliams 1964). This uses code preserving permutations (other than the purely cyclic one) in order to trap the errors. Unfortunately, for a VR system, different codes would probably require different permutations thereby unacceptably increasing the complexity of the decoder. One method of permutation decoding that would be suitable for VR is that of randomly choosing a permutation, and then attempting to error trap decode. Omura (1969) has shown that a small number of selections will yield a high probability of correct decoding. If t=(d-1)/2 or fewer errors are to be corrected then

$$N \geq \frac{\sum_{i=1}^{t} \binom{n}{i}}{\sum_{i=1}^{t} \binom{n-k}{i}} \approx \left(\frac{1}{1-R}\right)^{t} \text{ for n large,} \qquad 5.10$$

permutations are required. Table 5.14 below shows that the

n=31 codes could be feasibly decoded by this method.

Table 5.14

Permutations Required to Decode the n=31 BCH Codes

n	k	t	t'	N
31	26	1	1	-
31	21	2	1	5
31	16	3	1	9
31	11	5	2	10
31	6	7	5	5

The value of N increases rapidly with increasing t (and hence n) but even so, for the n=63 BCH codes the maximum number of permutations required is only 69 (for the (63,36,5) code).

In general it is possible to conclude that error trapping decoding, or modifications thereof, or hybrid techniques involving error trapping and some other decoding method, can be effectively and efficiently used to decode short ($\sim$50) constant n VR codes, for a wide range of multiple errors.

For $n \leq 127$, sets of constant n BCH codes provide a wide range of powerful random error correcting codes. In addition, it is certainly feasible to implement these codes in a VR system if it is feasible to implement the most powerful code in the set. This is because each code in the set works in the same $GF(2^m)$ arithmetic, and this means that equipment duplication is avoided.

We now consider the three steps in decoding a binary BCH code in more detail. These steps are:

Step (i) Calculate the t partial syndromes (s_1, s_3, ... s_{2t}) from the received vector r(x).

Step (ii) Determine the error location polynomial $\sigma(x)$ from (s_1, s_3, ... s_{2t}) in t steps using the iterative procedure due to Berlecamp (1968).

Step (iii) Determine the error location numbers (which are roots of $\sigma(x)$, and are equal to α^i for an error in the (n-i)th position), and correct the received vector by software or by Chien's procedure (1964).

Step i. The s_i can be computed by hardware FSRs. Each requires an m-stage register and the whole operation can be performed in n shifts as r(x) is read in. The simplification in the case of VR occurs because each code in a set of primitive constant n BCH codes has a generator made up of various minimum polynomials from just *one* set. That is, the minimum polynomials of α^i, where α is a primitive element in $GF(2^m)$, and $0 \leq i \leq n-1$. This means that the t registers used to calculate the partial syndromes for the most powerful code contain the t-1 registers that can calculate the partial syndromes for the next most powerful code, and so on, so that no extra registers (over that required by the most powerful code) are needed to decode all lower power codes. Code control circuitry therefore leaves all these registers running and merely selects the approriate partial syndromes after n shifts.

Software implementation of this step is simplified because of constant $GF(2^m)$ working. Addition is exclusive-OR addition of bytes which is available in most computers and

therefore requires one instruction execution; multiplication can be programmed by using shift and add instructions, and would require roughly 2m executions; division can be done by multiplication. The process requires a total of (n-1)t additions and nt multiplications, that is, (2nm+n-1)t instruction executions. In this case code control computes the required number of partial syndromes, the most powerful code taking the longest time to compute.

Step ii. Each of the t stages in the iterative algorithm requires less than 2t additions and 2t multiplications, giving a maximum total of roughly $2t^2$ additions and $2t^2$ multiplications. In software this amounts to $2t^2(1+2m)$ instruction executions. Hardware implementation requires the same amount of computation, but as arithmetic is $GF(2^m)$, one such unit can decode all the codes in the VR set. The speed of a hardware implementation would depend on how much parallel working could be performed. In the most extreme case t $GF(2^m)$ arithmetic units could be used, would be expensive, but would operate very fast. Roughly 4t m-stage registers would have to be dedicated to this step, but would only require 2mt clock cycles.

Step iii. Software implementation of this step requires n(t-1) multiplications and nt additions, that is, nt(2m+1) executions. Chien's procedure in hardware uses t of the step ii registers; t $GF(2^m)$ multipliers, which are m-stage FSRs; and in addition a combinational detection circuit which requires m t-input exclusive-OR gates, an m-input OR gate, and an inverter.

From the above considerations it can be seen that the amount of equipment and/or maximum decoding time for the VR code set is determined by the most powerful code. If steps i and iii are hardware implemented they can be performed during line input and message output, thereby contributing negligably to the total processing time. Total software decoding time is then only step ii time, which represents an increase in operating speed of one to two orders of magnitude over a full software implementation. With a 1μs computer instruction time, achievable bit rates would be approximately 75,000 bits/sec for the (15,5) triple error correcting code, and 5000 bits/sec for the (127,15) 27-error-correcting code.

In conclusion it can be said that a VR system using constant n BCH codes offers two major advantages. Firstly, a suitable range of short codes is easy to find, and secondly, one decoder, equal in complexity to the most powerful code's decoder, will decode all codes in the set with only slight modification.

Majority logic decoding for constant n VR sets is advantageous because there are a reasonable number of codes to choose from, and a type II decoder would require a constant number of stages. The main disadvantage is again the complexity and variability of the majority-gate network from code to code. Table 5.15 below lists several sets of suitable short codes.

Table 5.15

Constant n Majority Logic Codes

Code (n,k,t)	Type	Orthogonalisation Steps (L)	Gates	Inputs
15,11,1	EG(4,2)	3	7	2
15,7,2	EG(2,2)	1	1	4
15,5,3	EG(4,2)	2	7	6
15,1,7	EG(4,2)	1	1	15
31,26,1	EG(5,2)	4	15	2
31,16,3	EG(5,2)	3	43	6
31,6,7	EG(5,2)	2	15	14
31,1,15	EG(5,2)	1	1	31
63,37,4	$EG(2,2^3)$	1	1	8
63,13,10	$EG(3,2^2)$	1	1	20
63,6,15	PG(5,2)	1	1	30
63,1,31	EG(6,2)	1	1	61

It can be seen from table 5.15 that, in general a VR code set would require a different number of gates, with differing numbers of inputs, to be set up in configurations that were different for each code in the set. In addition, the exclusive-OR gates feeding these gates would have different numbers of inputs, and would sum different bit positions in the word. This would be feasible if the maximum number of gates required was not too large, but code control circuitry would rapidly become cumbersome with increasing n and L. One possible solution to this problem would be to choose a set of one-step codes so that only one variable-input majority gate would be required. The disadvantage of this method is the restricted choice of one-step codes for a given constant n.

Majority logic decoding for constant n VR code sets is therefore certainly feasible, and its scope can be greatly extended by combining with another decoding method. For

example, majority logic decoding and error-trapping would form a powerful hybrid VR decoding scheme.

Good burst performance is to be expected from constant n code sets by interleaving BCH or Reed-Solomon codes. In addition, interleaving is much simpler to implement than in the variable n case.

Finally, it can be concluded that constant n VR code sets are very attractive because: there is a wide choice of codes (even when restricted to one class); BCH codes are easily decoded; hybrid methods are easy to set up; and timing is simpler than for variable n methods.

5.6 *Other VR methods*

A VR code set can be constructed so that n,k and t are all variable. If decoding is not to be too difficult then the codes must be chosen so that a simple hybrid scheme, consisting of perhaps two or three methods, can decode all the codes. One way of doing this, and as an alternative to choosing relatively unrelated codes is to construct VR sets from product codes, or concatonated codes (section 3.3).

One method of using product codes to form a variable n, k, t set is to successively iterate a single code, as in Elias' error free coding. If the code used is simple to decode then so is the ith iteration of the code. Product block length, and therefore decoding delay, increase rapidly, however, thereby severely limiting the number of iterations.

VR sets with powerful random *and* burst correcting ability can be formed by concatonating various cyclic codes. If the outer code is chosen as a Reed-Solomon (N,K) code with symbols from GF(2^m), it is best to keep $N=2^m-1$ constant for ease of decoding. The inner code must then have k=m constant but its block length (n) can be made to vary; if a variable Nn scheme is allowed. With Nn constant the VR system adjusts the number of n bit 'bytes' that the outer code can cope with, while keeping the power of the inner code constant. With Nn variable the inner code is also varied, and this may be used to offset some of the load on the outer code. This scheme also raises the possibility of changing the coding to suit the 'burstyness' or 'randomness' of the channel. Under bursty conditions power should be directed to the outer code, whilst under random conditions the inner code should be more powerful.

VR code sets can also be constructed with unrelated codes, and under certain circumstances a separate (different) decoder for each code may be the most feasible solution.

5.7 *Channel statistics, code change criteria, and feedback signalling*

In this section the problems of determining the state (error statistics) of the channel, and establishing the criteria for making code changes are considered. Feedback signalling schemes are also outlined and code sets with disjoint code books are introduced.

5.7.1 *Extraction of channel statistics*

The 'state' of the channel can be monitored in the various ways previously outlined. In this section the methods considered do not involve the multiplexing of known sounding sequences in with the main bit stream. If such sequences can be used, it is possible to achieve a clearer picture of the channel particularly as regards the 'burstyness' or 'randomness' of the channel.

With these restrictions the obvious measurement to make is one of block error rate. The first scheme considered is therefore a constant n VR system operating on a slowly varying BSC, and measuring block error rate in order to obtain an indication of bit error rate. We also assume that by using sufficient interleaving such a scheme will be valid on a moderately bursty channel.

The criterion for changing a code will then be to examine the estimated prevailing block error rate, and compare this to thresholds determined from the code in use and the P_e required. If the comparison indicates that the block error rate is not within these thresholds, the code is changed so that the redundancy is either increased or decreased.

The threshold points, that is, the points at which each code is about to exceed the required maximum P_e, can be determined from the performance curves of the codes in the VR set (P_e versus p). Table 5.16 below shows the threshold points for a system using n=15 codes and operating to keep $P_e \leq 10^{-6}$. Also shown is the probability of undetected error

at each threshold point, assuming the lower redundancy code is in use at that point.

Table 5.16

Threshold point	Code in use between points	Bit Error rate p	Block Error rate p_{BLK}	Undetected Block Error Rate p_{UNDET}
-		0	0	0
	(15,11,1)			
1		10^{-4}	1.5×10^{-3}	3.24×10^{-14}
	(15,7,2)			
2		1.26×10^{-3}	1.87×10^{-2}	1.04×10^{-17}
	(15,5,3)			
3		5.01×10^{-3}	7.26×10^{-2}	2.39×10^{-19}
	(15,2,4)			
4		1.32×10^{-2}	0.181	1.33×10^{-22}
	(15,1,7)			
-		1	1	-

The VR system will therefore keep $P_e \leq 10^{-6}$ by using the appropriate code between the appropriate threshold points, assuming p_{BLK} is accurately known.

The correct operation of the VR system depends, to a certain extent, on the accuracy of the p_{BLK} determination and this raises two questions:

(i) what is the best way of sampling the block error rate, and how many blocks must be observed in order to determine p_{BLK} to a certain degree of confidence?

(ii) what is the effect of undetected block errors on the determination?

Considering question (ii) first, it can be seen from table 5.16 that for each threshold point, the undetected error rate is many orders of magnitude lower than the block error rate that is being determined. It can therefore be concluded that undetected errors will not effect the p_{BLK}

determination, provided the correct code is in use for the prevailing p_{BLK}. It is, however, the prevailing value of p_{BLK} that is being determined, and it is therefore quite possible that the wrong code is in operation. There is a wide margin of error, however, which can beshown as follows. From table 5.16 it can be seen that the lowest value of $r=p_{UNDET}/p_{BLK}$ occurs at threshold 1, and has a value of 2.16×10^{-11}. If it is now assumed that the system is operating incorrectly in that progressively weaker codes are being used instead of the required code, table 5.17 below gives the corresponding values of r.

Table 5.17

Values of r for Incorrect Operation

Number of codes lower than required	Lowest $r=p_{UNDET}/p_{BLK}$
0 (correct)	2.16×10^{-11}
1	5.35×10^{-9}
2	1.52×10^{-7}
3	5.52×10^{-7}

From table 5.17 it can be seen that even though a much weaker code than required may be in use while p_{BLK} is being determined, the undetected error rate is still negligible in comparison to p_{BLK}, and will not significantly affect the determination.

Question (i) is a statistical question and involves notions of sampling inspection schemes. The remainder of this section considers the measurement of block error rate by sampling; the derivation of code change criteria based on the sampling scheme used is considered in the next section.

Block error rate p_{BLK} is a binomially distributed variable with mean $p'=1-q'$ and standard deviation (s.d.)$=\sqrt{p'q'/n}$, where n is the number of blocks over which the observation (or sample) is taken. As $n\to\infty$ the s.d. of the distribution tends to zero and the error rate observed in the sample tends to the true mean of the distribution. If n is small then the error rate sample may differ significantly from the true mean.

We first consider a single sample of block error rate. This can be taken in two ways.

(i) Fix the number of blocks in the sample (n), and count the block errors that occur.

(ii) Fix the number of block errors (x) and count blocks until that number of erroneous blocks occur.

In method (i) the sample size is fixed and in (ii) it is variable. Method (ii) is preferable for several reasons: firstly, with method (i), n must be large if low error rates are to be determined, and this means that delay is large all the time; secondly, with method (ii) the number of blocks counted depends on the prevailing error rate and is low at high error rates (therefore short delay) and long at low error rates; thirdly, the s.d. is unnecessarily low at high error rates with method (i), whilst with method (ii) the s.d. tends to be a fixed proportion of the mean. This can be shown as follows: if x errors occur, the expected sample size $n=x/p'$. The expected s.d. is therefore $=\sqrt{p'^2q'/x}$, and as $p'\to 0$ the s.d. $\to p'/\sqrt{x}$. This means that regardless of the actual mean block error rate only the minimum number of

blocks necessary to give a certain s.d. are counted, which in turn means that the sampling delay is always as short as possible.

It is important to investigate the accuracy of the p_{BLK} determination when a single sample is taken. Defining m=x/n as the sample mean, that is, errors divided by total blocks, the probability (P) of a sample mean deviating from the actual mean (p') by more than ε is given by the weak law of large numbers:

$$P\left(|m-p'| \geq \varepsilon\right) \leq \frac{p'q'}{n\varepsilon^2} \qquad 5.11$$

where $p'q'/n$ is the variance of the error rate. This result is easy to evaluate but is a very weak bound on P. A much more powerful, though less easy to evaluate, bound is the Chernoff bound (Wozencraft and Jacobs 1965):

$$e^{-nX} \geq \begin{cases} P\left(m \geq d''\right); & 1 \geq d'' > p' \\ P\left(m \leq d''\right); & 0 \leq d'' < p' \end{cases} \qquad 5.12$$

where $X=-ln\left(\left(\frac{p'}{d''}\right)^{d''}\left(\frac{1-p'}{1-d''}\right)^{1-d''}\right)$

and d'' is a threshold value of the block error rate.

The Chernoff bound can be used to calculate the probability that, given a certain observed sample mean (m) which lies within two threshold points, the actual block error rate lies outside these two points, thereby indicating that a different code should be in operation. If m lies roughly midway between two threshold points then it can be seen that there is a high probability (how high depends on the sample size) that the correct code is in operation. As m nears a threshold point the probability that the actual block error rate is past that threshold increases. A single

sample therefore suffers from the disadvantage that there is a high probability of initiating an incorrect code change if the actual error rate is near a threshold. Increasing the size of the sample serves to narrow this *band of uncertainty* about a threshold point, but means that for most of the time the sample size (and hence code change delay) is unnecessarily large. A much better solution is to accept a wide band of uncertainty about a threshold point, and change to a multiple sampling scheme, as described later.

In order to indicate sample sizes (n) for the VR system so far considered, table 5.18 shows results obtained by the Chernoff bound to calculate the worst case number of blocks in a single sample that will provide a certain confidence that the actual error rate is greater than a certain threshold point; given that the *observed* value of m is less than or equal to the next lowest threshold point. That is, the worst case confidence that the code in use is not ≥ 2 codes too weak. Table 5.18 shows that high confidence can be obtained from reasonably small sample sizes. Also shown is the expected average confidence when method (ii) sampling is used, that is, when the sample size is variable. Two values are shown corresponding to stopping the block counter after 7 errors, and after 10 errors.

Because table 5.18 considers error rate values exactly one threshold apart, the figures also give a rough indication of the average confidence (method (ii)) that: when a mid-threshold m value is observed, the actual error rate is *not*

Table 5.18

Required Sample Sizes

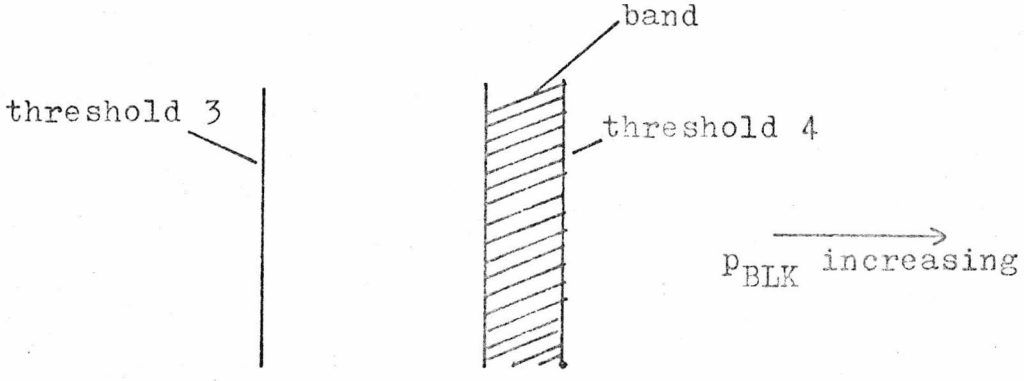
Observed sample m near which error threshold?	rate	Actual p′ near which error threshold?	rate	Confidence %	Required n	Average Confidence % 7 errors	10 errors
<3	$<7.26\times10^{-2}$	>4	>.181	99 90 80 70	94 47 33 25	85	93
<2	$<1.87\times10^{-2}$	>3	$>7.26\times10^{-2}$	99 90 80 70	153 77 54 40	95	98
<1	$<1.5\times10^{-3}$	>2	$>1.87\times10^{-2}$	99 90 80 70	278 139 97 73	$\tilde{}$100%	$\tilde{}$100%

≥ the next highest mid-threshold value.

The Chernoff bound can also be used to calculate similar confidence values when bands of uncertainty are introduced around the threshold points. The size of a single sample rises rapidly, however, as the band narrows. For example, consider figure 5.17.

Figure 5.17

Band of Uncertainty



b

We require to determine the number of blocks in the single sample necessary to give a confidence of $\geq$80%, that when an observed sample m has value $\leq$b, the actual error rate (p') is $\leq p_{BLK}$ at threshold 4. Application of the Chernoff bound when b is such that the interval from b to threshold 4 is ¼ of the inter-threshold interval 3 to 4, gives $n \geq 3629$; corresponding to about a minute delay at 1000 bits/sec.

A simple code change scheme based on the result of a single sample can therefore be built up in this way using samples as large as the rest of the system allows. The VR system would then use the criterion that if the error rate sample indicates that the threshold has been passed, the code is changed. This may or may not be a correct decision, depending on the prevailing confidence. Low values of confidence are still of value, however, because the VR system can be made to operate so that there is a bias towards choosing too high a redundancy rather than too low a redundancy.

A single sample yields a value of m, that is, the mean of a sample. If N such samples are taken a distribution of *sample means* may be plotted. This distribution tends to a normal distribution regardless of the original block error rate distribution. The mean of this *distribution of sample means* is the mean of the original distribution, that is, the required average block error rate, and the s.d. is given by:

$$\sigma_m = \frac{\sigma}{\sqrt{N-1}} \qquad 5.13$$

where σ is the s.d. of the samples. With method (ii) sampling this gives

$$\sigma_m = p' \sqrt{\frac{q'}{x(N-1)}} \qquad 5.14$$

The VR scheme can therefore base its estimate of block error rate on the *mean of sample means* rather than on the mean of a single sample. Table 5.19 below shows the number of samples (N) required to give a certain confidence that the mean of sample means is with $\pm yp'$ of the true mean.

Table 5.19

Required Number of Samples (N)

±y	Confidence	x	N ($q' \backsim 1$)
.1	80	7	25
	90	7	40
	99	7	96
.05	80	7	95
	90	7	156
	99	7	380

5.7.2 *Code change criteria based on sampling inspection schemes*

The simplest code control scheme consists of inspecting one sample and instructing the coder to use the code corresponding to the estimated error rate. This generally requires a large number of blocks in the sample, with a large decision delay.

An alternative method of code control is to base the code change criterion on the result of a sampling inspection scheme. The simplest inspection scheme, called single sampling, would involve taking N estimates (samples) of the error rate. A decision is then made whether to increase, decrease or hold

redundancy. Such a scheme could operate in the following way:

(i) If more than ½ the samples are above the upper threshold point, change to the next most powerful code.

(ii) If more than $^2/_3$ of the samples are below the lower threshold change to the next lowest powerful code.

(iii) Otherwise stay with the present code.

This scheme has a bias towards not reducing redundancy. Many such schemes can be implemented and their operating characteristics can be determined by simulation rather than lengthy calculation. The common factor in all these schemes is that there is a constant N sample delay before a decision can be taken, although if sampling method (ii) is used the decision time is not constant, but is shorter at high error rates.

An extension of single sampling is double sampling. With this method a first sample is not necessarily decisive but may lead to the taking of a second sample which, in conjunction with the first, is then to be decisive. The advantage of double sampling (over single sampling with double the sample size) is that the first sample offers the opportunity of taking a quick decision if it is reasonably certain that thresholds have been exceeded, while the second sample enables accuracy to be increased if the decision is not so clear cut.

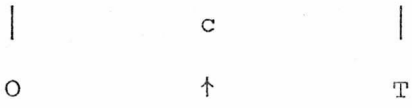
A logical extension of double sampling is that of sequential sampling in which a decision is made after each sample either to (i) increase redundancy, or (ii) decrease redundancy, or (iii) hold the present redundancy

but carry on sampling. This method is specially advantageous because: (i) gross changes in error rate are quickly detected, (ii) the past history of the sample values contributes to every decision, (iii) the difficult near-threshold decisions are automatically allotted more time, and can therefore be determined with greater accuracy than a non-sequential method.

The method can be implemented with a reversable counter, and is shown diagramatically in figure 5.18 below.

Figure 5.18

Sequential Sampling



Two counts or thresholds are defined, zero and T. The pointer initially starts midway between these two and then moves according to the following rules:

(i) if the sample is below the lower threshold move the pointer x places to the left

(ii) if the sample is above the upper threshold move the pointer y places to the right

(iii) if the sample is within the current threshold points move the pointer 1 place towards the centre, or stay at the centre if currently at the centre.

Sampling continues, using the present code, until either the zero or T threshold is crossed, and redundancy is then decreased or increased respectively. The values of x, y and T fix the characteristics of the scheme about which the following points can be noted:

(i) If x and y are equal there is no inherent bias to moving up or down.

(ii) The value of y (relative to T) determines the length of time taken to change up when confronted with a gross and sudden upward change in error rate.

(iii) The value of y relative to 1 (and T) determines the amount of time taken to change up (and the decision accuracy), when the error rate is near the threshold. For example, if the error rate is at the upper threshold we may expect half the samples to be above the threshold and half to be within the current threshold points. If y=1 the pointer will remain relatively stationary but if y=2 there will be a steady drift towards T. Therefore, as y increases, the rate of drift towards T increases, and the equilibrium point moves progressively towards T/2. This means that the larger y is the quicker a code change decision is taken, once the equilibrium point is passed.

(iv) Similar arguments apply to the value of x.

Sequential sampling therefore offers a versatile method of making code change decisions for a VR system in a way that adapts to the difficulty of the decision.

5.7.3 *Other code change schemes*

Two other basic types of code scheme can be considered: (i) schemes that attempt to monitor the operation of the code itself (eg. at the decoder), and (ii) schemes that make simple but quick assumptions (i.e. not calculated) about the behaviour of the channel. The channel itself is

assumed to have some degree of memory for both types of scheme to operate successfully.

One method of monitoring the progress of the code is to try to decide if the code is 'overloaded' or 'underloaded' with errors. An overload can be considered to be detected in several ways. For example:

(i) If a detectable but uncorrectable error occurs in a bounded distance decoding scheme.

(ii) If an error pattern of t+1 is observed when a t-error correcting quasi-perfect code is being fully decoded.

(iii) If the code is correcting up to its full capability (i.e. t errors) for a disproportionate amount of time. The appropriate action to take on detecting an overload is then to increase redundancy. Underload detection is more difficult to detect. Rather, what is detected is the absence of overload over a suitable period of time after which redundancy is reduced. If conditions have not changed for the better an overload should quickly occur, indicating that the attempted reduction of redundancy was not justified, and that the higher redundancy code was more suitable.

Another possible method of monitoring code progress is to assess decoding 'difficulty'. If the decoder is having great difficulty in decoding, the redundancy is increased whereas redundancy is reduced if decoding is too easy. Such a procedure can be applied to a system using variable threshold majority logic decoding. In this case the number of attempts at decoding with different thresholds, or the total decoding time, can be taken as an indication of decoding difficulty and used to initiate code changes. Randomised permutation decoding

also offers a way of monitoring decoder difficulty by noting the number of permutations required to decode a particular code, and comparing this to the expected number for the codes correcting capability. If these numbers are disproportionate a code change can be initiated.

When the channel behaves in a predominantly bursty way it becomes possible to use simpler, but cruder, code change criterion. In the simplest case the channel can be considered as a two state Markov process, and the VR system would correspondingly use two codes only, one for the 'good' (low error rate) state and one for the 'bad' state. The system would operate with the low redundancy code curing the good state, and then change to the high redundancy code on detection of an erroneous block on the assumption that either more errors will follow or a 'bad' state is about to begin. Redundancy would then be tentatively reduced after a number of error free blocks had elapsed. Such a system can be extended to three code operation so that a middle redundancy code can be tried before committing the system to a very inefficient code. In this case the three codes could correspond to error-free, low error density, and high error density states. Extension to more than three codes is not justified because it is not really valid to attempt to even more finely divide channel states in this way. Even a three code system, however, would be expected to increase throughput efficiency.

Other code change schemes can be designed by considering particular decoding methods, and in particular the scope of such

schemes can be greatly increased if demodulation information is available. In general the operational characteristics of a code change scheme are not easily calculated, but can be determined by simulation

5.7.4 *Feedback signalling and errors*

There are basically two different types of signalling schemes that can be used on a VR system regardless of whether feedback is via a dedicated feedback link or via a multiplexed return bit stream. In scheme (i) the only signals are 'up', 'down' and 'stay'. In scheme (ii) the code required is signalled by name.

Considering scheme (i) first it can be seen that regardless of the number of codes in use (m), only 3 signals are required, and if the feedback channel is a dedicated analogue channel this can be achieved with two waveforms (or 1 bit) for up and down and no signal for stay. If the return channel is a multiplexed bit stream then two bits are required, but economies may be affected by utilising, say, three orientations of an existing synchronising sequence. This scheme uses the minimum amount of signalling, but it does assume that the coder and decoder are operating in the same code, and it does mean that codes can only be changed one at a time in sequence.

Scheme (ii) on the other hand, requires an analogue waveform or $\log_2 m$ bits. Given that m need not be more than 8 it can be seen that 3 bits are required for this scheme as opposed to two bits for scheme (i). In addition, it

would be difficult to find an existing sequence in the return bit stream that could represent 8 different waveforms, without having to introduce extra dedicated feedback signalling bits. The scheme can, however, choose any code in any sequence.

The bandwidth required and the delay introduced by the schemes must be considered in relation to whether operation is full duplex, or simplex with dedicated feedback.

With dedicated feedback there is not much difference between the schemes: signals can be transmitted at any time, and the feedback link is used only rarely if 'change' signals only are sent. The bandwidth required is not greatly different and depends on the number of blocks delay that is acceptable. For example, if a maximum of 3 signal bits are required, and it is assumed that this is (7,3) error coded for protection, then 7 bits must be return transmitted in xn forward bit periods (where x is the number of blocks delay) giving a return bandwidth of roughly 7/xn of the forward bandwidth.

With full duplex control, feedback information can be transmitted via dedicated feedback bits, or via timing sequence alteration if possible. The amount of delay depends on how frequently these dedicated bits or sequences are inserted into the main bit stream. The shortest delay (and largest forward time wastage) occurs when feedback bits are included in every block. Scheme (i) is certainly advantageous in this case as it uses the minimum number of bits, and in fact can also provide an ARQ request with the

one remaining combination of the two bits. Error control is provided by the forward coding scheme in use.

The effect of feedback errors on the VR system depends on which signalling scheme is in operation. Such errors can be reduced by adequate coding, at the price of increasing bandwidth, delay or the proportion of signalling bits to total bits. If scheme (ii) is in operation and an error causes the coder-decoder to become mismatched, then the next set of feedback information should direct the coder back to the correct code. With scheme (i) however the next feedback signal will not necessarily rectify the error. For both schemes there is a period during which mismatch occurs and if this is not detectable then erroneous data is delivered to the sink. This problem has led to the investigation of codes with disjoint code books and mutual Hamming distance (Chapter 8; Appendix A).

The use of codes with disjoint code books for VR means that the decoder can detect when a mismatch occurs, because even if a reasonable number of forward channel errors are occuring the block error rate will rise dramatically (with no forward channel errors, consecutive block errors would occur). With scheme (ii) this means that incoming blocks are not undetectably erroneous but can be stored and later decoded in the code that the decoder believes the coder was using. With scheme (i) the decoder detects the mismatch situation, and as well as storing incoming words for later decoding, sends 'change up' redundancy signals until the coder and decoder reach the highest redundancy code, when they are in code match again.

CHAPTER 6

Experimental Work on Variable Redundancy Coding

This chapter describes experimental work on VR coding; which may be split into two parts:

(i) the designing, building and operation of an experimental automatic VR system;

(ii) computer analysis of the forward HF channel and the experimental system, and computer simulation of various other VR systems.

The initial considerations that lead to the proposition of the particular experimental system are first outlined, so that the way in which the design and testing of the system developed can be appreciated. The modems and feedback link are next described and this is followed by a description of the organisation of the field tests together with the equipment used on each test. The experimental VR system is then outlined. Finally, the analysis and simulation work performed on the computer is described.

In the descriptions of the hardware that follow, it is not intended to explain the design and construction of the circuits in great detail; rather the circuits will be described at block diagram level wherever possible, and it is to be assumed that modern integrated circuits, both analogue and digital, were used to full advantage.

6.1 Initial considerations

This section outlines the initial decisions taken as to the development of the experimental work, and the

construction of the experimental VR system, and can be considered as the original specification for the research.

6.1.1 *The Forward Channel*

This was chosen to be a nominal 3kHz simplex HF channel operating between West Wellow (Hants) and Canterbury (Kent), a distance of 124 miles (200 km), with the transmitter at West Wellow and the receiver at Canterbury. The type of propagation aimed for was preferably single hop via the F_2-layer with no interfering E-layer reflection. At this range E-layer reflection ceases at approximately $\sqrt{2}f_c$, and taking an absolute maximum E-layer critical frequency of f_c=4MHz, this indicates that frequencies above about 5MHz should be used. Frequency predictions for the path indicated that optimum working frequencies (OWFs) would be in the region of 6 to 8MHz over the period August '70 to May '71 for single hop F_2 daytime propagation. Seven spot frequencies in the range 3.860MHz to 8.994MHz were available for use during the field trials, and after initial propagation tests the operating frequency was fixed at 7.375MHz.

6.1.2 *Modulation and bit rate*

A simple modem arrangement capable of cheap and quick construction was required. For this reason sub-channel systems with parallel low speed bit streams were rejected, and single channel high-speed FSK with a nominal ±425Hz frequency deviation was chosen.

The choice of single channel working dictates that the bit rate must be reasonably high to be comparable to practical

systems, so this was chosen to be 1000 bits/sec. This bit rate is readily achievable for ranges of 1000-4000 km, but is certainly fast for a 200 km range, and implies severe multipath distortion. It was therefore decided that the bit rate should be variable, with the coder controlled bya master clock at the receiver, in case a bit rate reduction was absolutely unavoidable. Such a reduction was not, however, expected for the following reasons:

(i) The operating frequency was expected to be close to the MUF for most of the time, thereby reducing multipath.

(ii) A reasonably 'bad' channel was required, to severely test the coding system.

(iii) Conditions on HF are so variable that if multipath is making a test run meaningless, one need only wait for a reasonably short time before the multipath structure changes for the better.

6.1.3 Feedback signalling

An ideally error-free feedback link was required for the experiment and for this reason a Post Office private leased line was chosen. Feedback signalling was chosen to be two audio tones, one to indicate 'up' and one to indicate 'down'- changes in redundancy, which could be transmitted at any time. The absence of a control tone was to be taken as a 'stay' signal. In addition, it was hoped to bit-synchronise the Tx and Rx stations by sending a constant 1kHz tone via the line. This later proved to be impossible because the P.O. SSB circuits involved in the line (including the London-Southampton microwave link) do not have locked carriers. A slightly drifting frequency displacement occurred

when this method was tried, resulting in a sync slip every 15 minutes which corresponds to an oscillator accuracy of $\sim$ 1 part in 10^6.

6.1.4 *Synchronisation*

Bit synchronisation was eventually achieved by using AIRMEC receivers locked to the 200kHz Droitwich transmissions, in conjunction with phase locked loop (PLL) control. Frame synchronisation and start-up procedures were to be manual for simplicity, and controllable from the receiving station.

6.1.5 *System and field trial organisation*

The main criterion here was that the transmitting station should be robust (for safe transportation to and from Canterbury) reliable, and capable of operating unattended for several weeks. All adjustments, after an initial set-up, were therefore required to be controllable from Canterbury via the feedback wire. This was achieved; apart from the variable bit rate requirement.

Four field trials were planned for the period August 1970 to May 1971, during which the channel analysis recordings and the experimental system were to develop, culminating in fully automatic VR tests.

6.1.6 *Error control and coding*

It was decided to use up to eight length 15 binary cyclic codes for the VR code set, with provision for varying the number of codes in use by receiver operations via the feedback link. The error control method and therefore the

decoding procedure were deliberately left unspecified in the initial stages, with the aim of making as much use of the computer as possible, thereby minimising hardware requirements and realising a very flexible system. It was therefore decided to have the system operating in the VR-ED mode and to on-line record both decoder operations and the incoming bit stream, so that offline ARQ and FEC decoding could be performed by computer and analysed at leisure. All three types of error control could therefore be analysed without complex hardware requirements at the decoder.

6.1.7 Code change criterion

In order to achieve the decoding flexibility mentioned above it is necessary to have a code change scheme that is not particular to one error control method. For this and other reasons the block error rate sequential sampling scheme was chosen, and it was decided to simulate other schemes if needed. It was also decided that the system should operate on reasonably low confidence levels in order to minimise change delay.

6.1.8 Recording

It was decided that recordings of incoming bit streams and decoder operations should be capable of being both: recorded on-line onto computer magnetic tape; or recorded on standard ¼" magnetic tape for later off-line transcription onto computer magnetic tape. Provision was also made for bit and block error counting, and pen recording of received signal strength (S-level recording).

6.2 *The experimental system*

The initial requirements outlined in section 6.1 were nearly all met satisfactorily. The equipment used in the four tests was gradually built up until the full VR system was operational for the fourth test, and it is appropriate at this point to give the final system diagram before describing the sub-units in more detail. This is shown in figure 6.1.

6.3 *HF and modem equipment*

6.3.1 *Aerials*

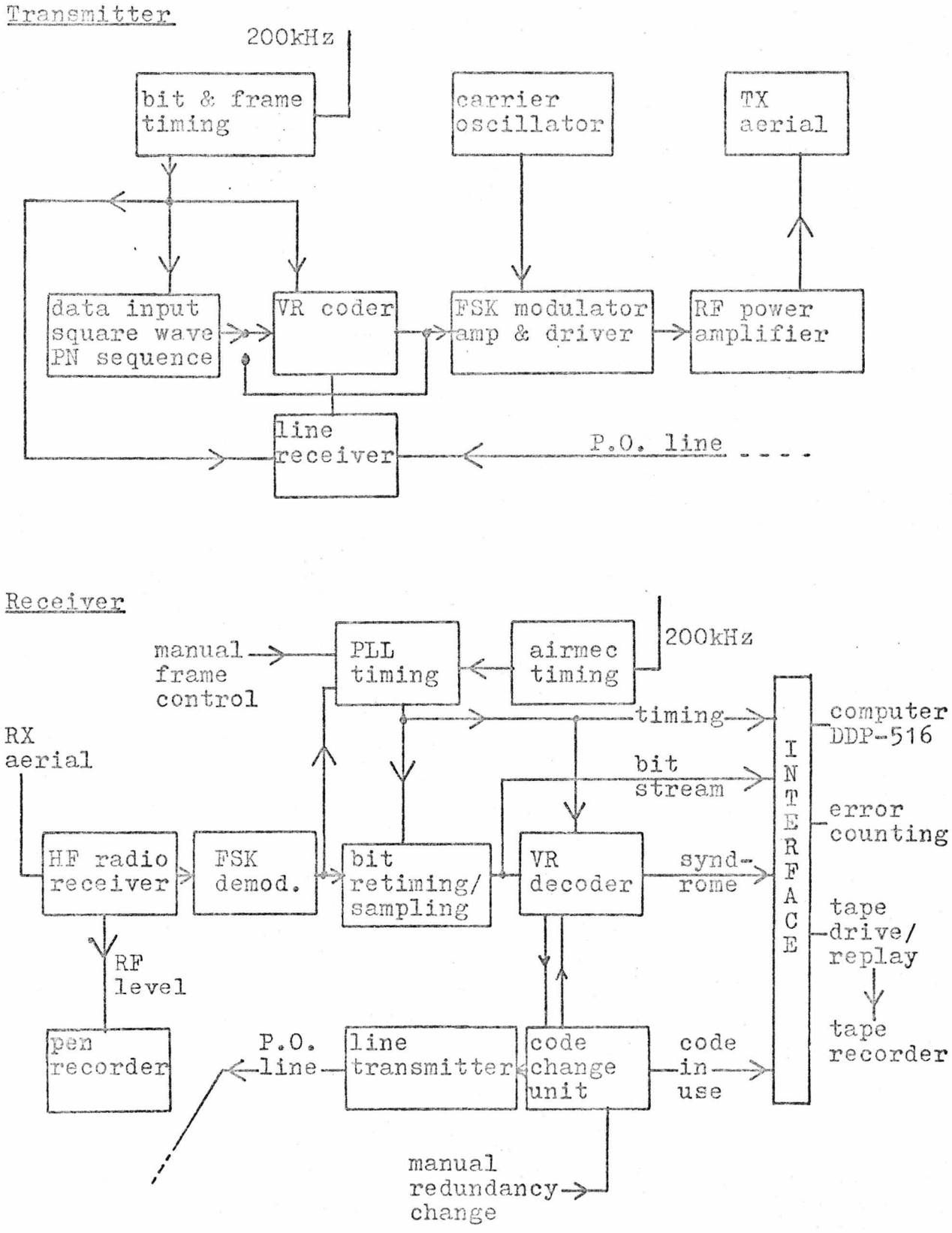
Several types of aerial, including verticals and dipoles, were tested during the field trials, and the final outcome was that halfwave dipoles were used at both stations. The transmitter aerial employed a balanced feeder, and was supported above relatively open ground by 75' high masts. The receiving aerial was supported by 20' high masts on the roof of the Electronics Laboratories at the University of Kent, and the feeder contained a balun to match into the 75 ohm unbalanced receiver input.

6.3.2 *Power amplifiers*

The site transmitter at the West Wellow station was initially used, and is a Redifon cabinet-mounted valve transmitter capable of about 100 watts output. A much more compact and portable 150 watt valve transmitter was built by D. E. Pantony at Canterbury, and was used for the later trials. The approximate radiated power used throughout the trials was 75 watts.

Figure 6.1

The Experimental System

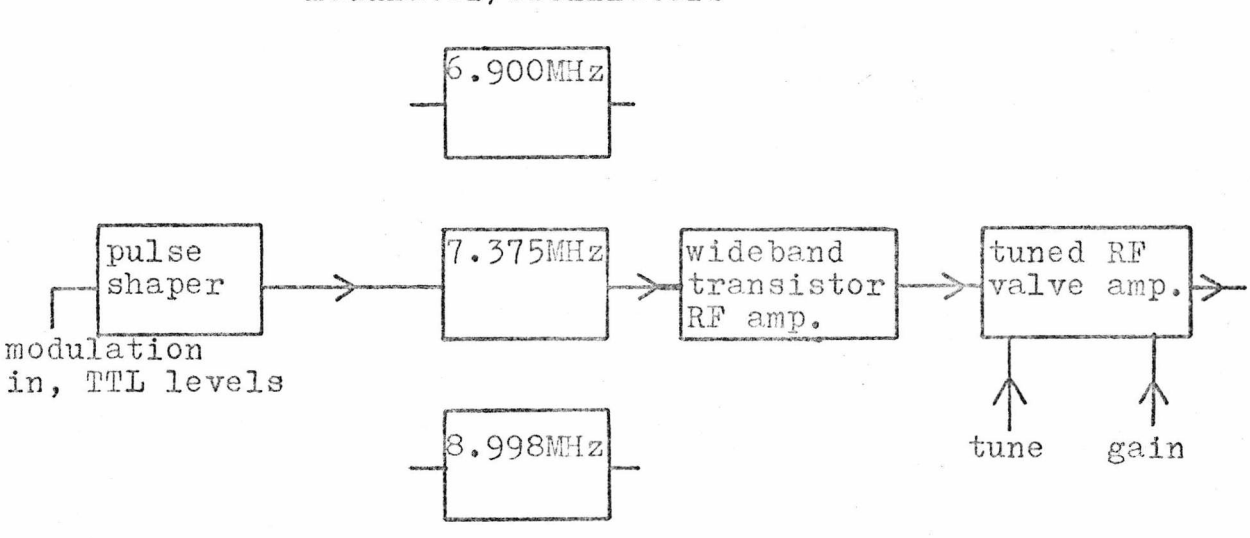


6.3.3 *FSK modulator and power amp driver*

This equipment occupied one level of a 19" rack and was designed to produce a continuous-phase FSK signal capable of driving the main power amplifier from a TTL logic digital input signal. The units in this rack are shown in Figure 6.2 below.

Figure 6.2

Modulator/Driver Unit



The function and operation of the units are as follows.

The pulse shaper is a low pass filter/amplifier whose input can be switched to accept a constant '0', a constant '1', or a TTL data input signal. This serves to remove the sharp TTL pulse transitions and thereby band-limit the input to the modulator.

The band limited signal passes to any one of three modulator/oscillators enabling operation at carrier frequencies of 6.900MHz, 7.375MHz and 8.998MHz. Each unit

is basically an FET crystal oscillator whose frequency can be 'pulled' by applying the data signal to a varactor diode in the frequency-defining circuit. The resulting FSK signal has an 850Hz shift, and the '0' and '1' frequencies are f_c-450Hz and f_c+450Hz respectively.

This low power FSK signal is then amplified in two stages as shown in figure 6.2 to give an 80 volt (maximum) signal capable of providing 5 watts of drive for the power amplifiers.

6.3.4 The receiver

The receiver used was a Racal RA 1217 solid state HF communications receiver, which features high stability and a wide range of controls. Aerial input is 75 ohms unbalanced, and the output taken was the final 100kHz I.F. capable of providing 1mW into 75 ohms.

The AGC line from the receiver drives the internal 'S' meter and was used to drive a pen recorder so as to produce a permanent record of the input R.F. level. Each roll lasted approximately 52 hours and the paper speed was ¼" per minute enabling fading effect to be resolved to about 5 seconds.

6.3.5 FSK demodulation

The demodulation of the FSK signal can be done by two filters, one for the '1' frequency, and one for the '0' frequency. The choice of a final I.F. at which to filter these signals out is governed by two opposing factors:

(i) If the I.F. is high (100kHz) the filters must have very high Q in order to discriminate a shift of 850Hz with low crosstalk.

(ii) If the I.F. is low (10kHz) the choice of bit rate (500Hz squarewave) means that the filter must respond to a possible minimum of 20 cycles; a very fast response is therefore required in this case.

Initial experiments with high-Q active filters proved unsatisfactory because a compromise betweenthe above two factors could not be achieved.

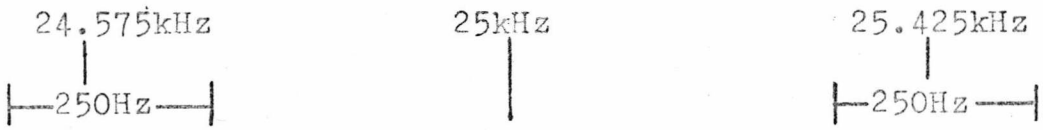
The final circuit was designed around the FX-101 Z-trip MOS integrated circuit. This I.C. is a frequency sensitive switch capable of very accurate (and high Q) switching at externally set threshold points. The device uses a form of digital frequency discrimination which samples the periods of input signal waveforms and compares them with predetermined reference values on a statistical basis. This technique allows the input frequency to be resolved to a high degree of accuracy within a very short period of time (typically 9 signal cycles), and also renders the device relatively immune to false switching by noise and input amplitude variation. The effect of noise is to increase the number of cycles needed for the sample.

The final demodulator employed three FX 101's: two as band filters tuned to the '0' and '1' frequencies, and one as an above/below threshold switch, set to the centre (I.F.) frequency. A majority decision was then taken on these outputs to provide the final output. The frequency

points which were chosen after considering the FX 101 performance limits and the previously mentioned factors; were nominally as shown in figure 6.3 below, and were also manually adjustable.

Figure 6.3

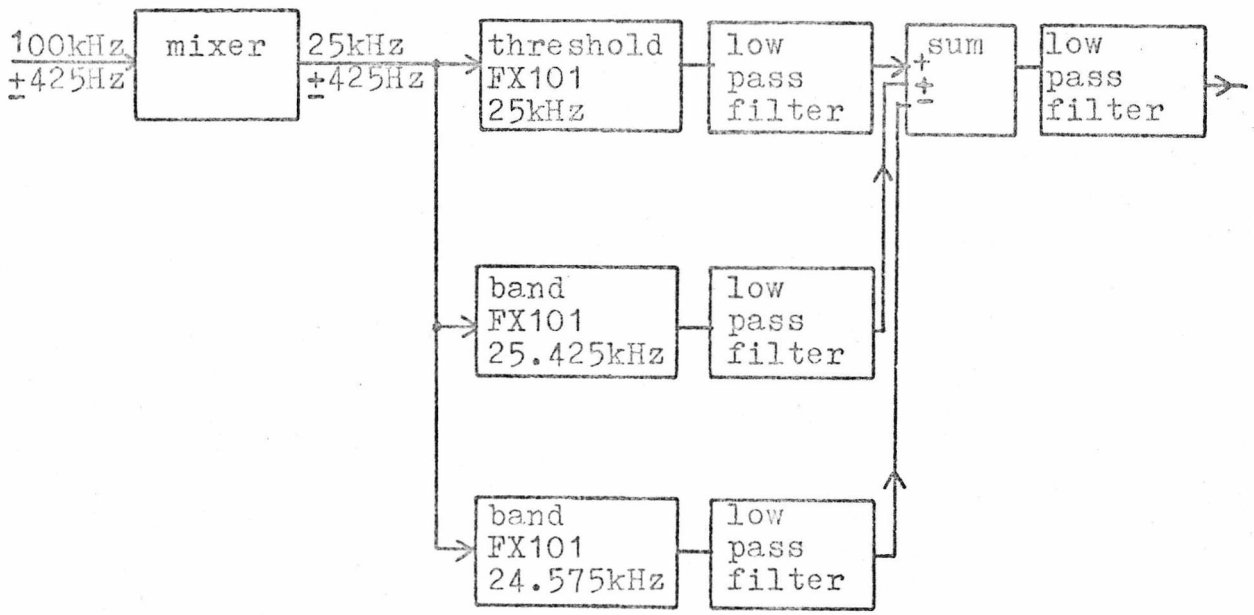
Frequency Thresholds



The demodulator system diagram is shown in figure 6.4 below.

Figure 6.4

The Demodulator



The 100kHz receiver I.F. was first mixed down to provide a 25kHz centre frequency which was then passed to the frequency selective switches. The switch outputs are then low pass filtered to remove spurious switch transients of duration much less than 1ms (the bit period). The summing

amplifier then produces a composite waveform from the three outputs that is later threshold detected to provide the majority decision. This waveform is further low pass filtered to remove short switch transients resulting from the summation of three waveforms with small relative phase differences.

The output of the summer is therefore a smoothed digital signal of amplitude 0, 1, 2 or 3 units, depending on which switches are 'on' and thereby contributing 1 unit. This is later threshold detected at 1½ units (section 6.5) with the result that the following modified majority decision scheme is automatically implemented:

(i) If the two band filters agree on a particular digit (they should not) the output is taken as the *threshold* FX-101 output.

(ii) If the two band filters disagree (that is, one has an inband signal, the other has not), and the threshold filter agreeds with the inband filter, their output is taken as the true output.

(iii) If the two band filters disagree and the threshold filter disagrees with the inband filter, the inband filter indication is assumed the more important and overrides the threshold filter to provide the output.

The jitter on the composite waveform under usable signal strength conditions was observed to be a maximum of 0.12ms per edge, corresponding to a variation of 3 cycles in the number of input cycles required by the FX-101 to make a decision. Stability with regard to temperature and

supply variations was excellent and the circuit required little retuning after initial setting-up.

6.4 *Feedback signalling*

Feedback signalling was conducted via the P.O. line and consisted of tone bursts of different frequencies, one for 'UP', and one for 'DOWN' redundancy changes. These bursts were initiated at the receiver, either manually or by the code control unit. A third tone of different frequency was manually operated to control either 3-code or 8-code VR operation.

The system diagram is shown in figure 6.5 below.

The timing sequence of the code change operation which operates with a 1 block delay is shown in figure 6.6 below.

The availability of the dedicated feedback link made the use of the relatively simple line transmitter circuit of figure 6.5 possible because only one tone at a time need be transmitted. The signal circuit used, however, also mixed a 1kHz continuous synchronisation tone with the output, and provided a spare tone for possible ARQ. The choice of frequencies for these tones reflects the need to avoid false triggering due to intermodulation products.

At the line receiver FX-101 microcircuits were again used to recognise the tone frequencies. the 'UP' and 'DOWN' switches were set for a bandwidth of approximately 40Hz, and operated in the latch-reset mode. In this mode the switch latches to 'ON' when an inband frequency is

Figure 6.5

Feedback Signalling

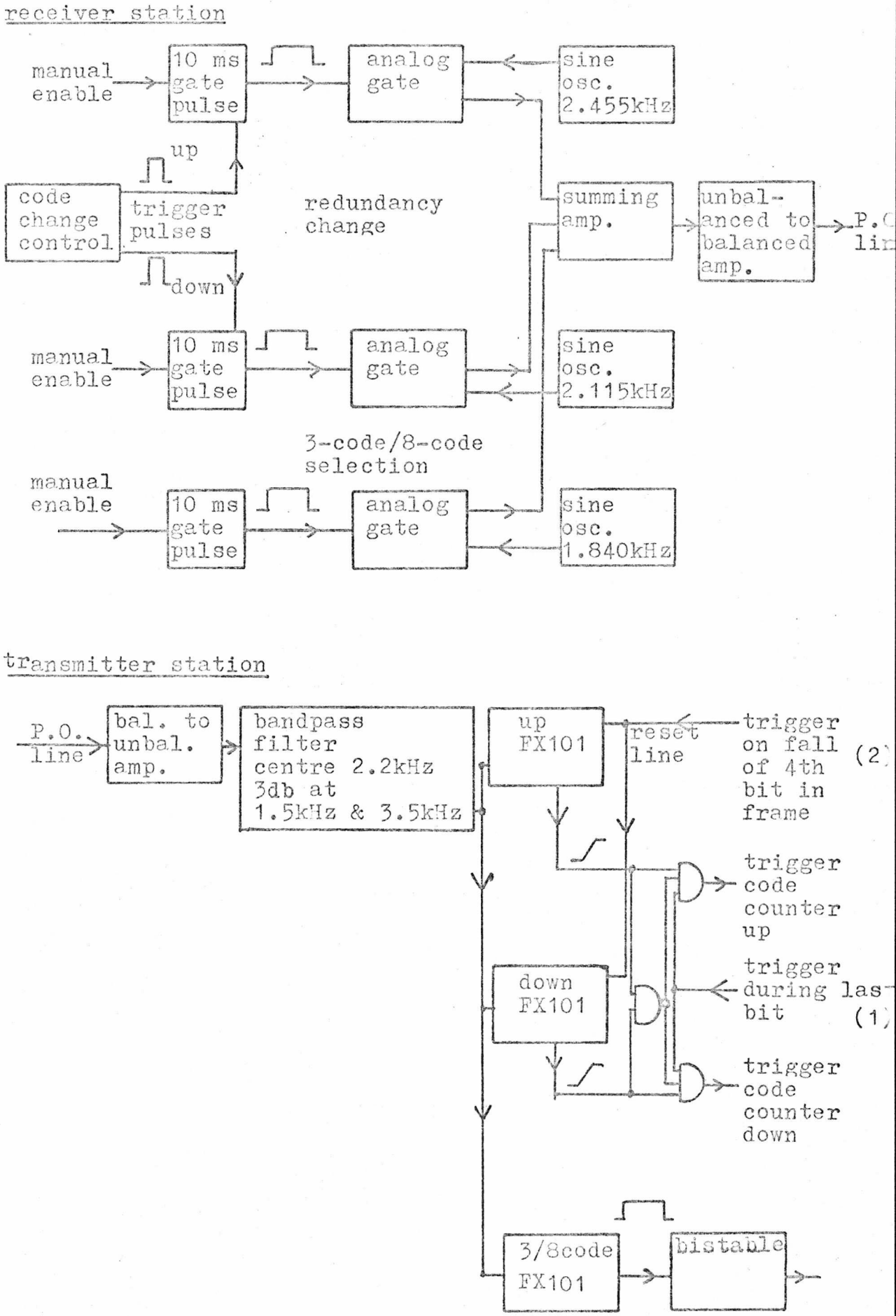
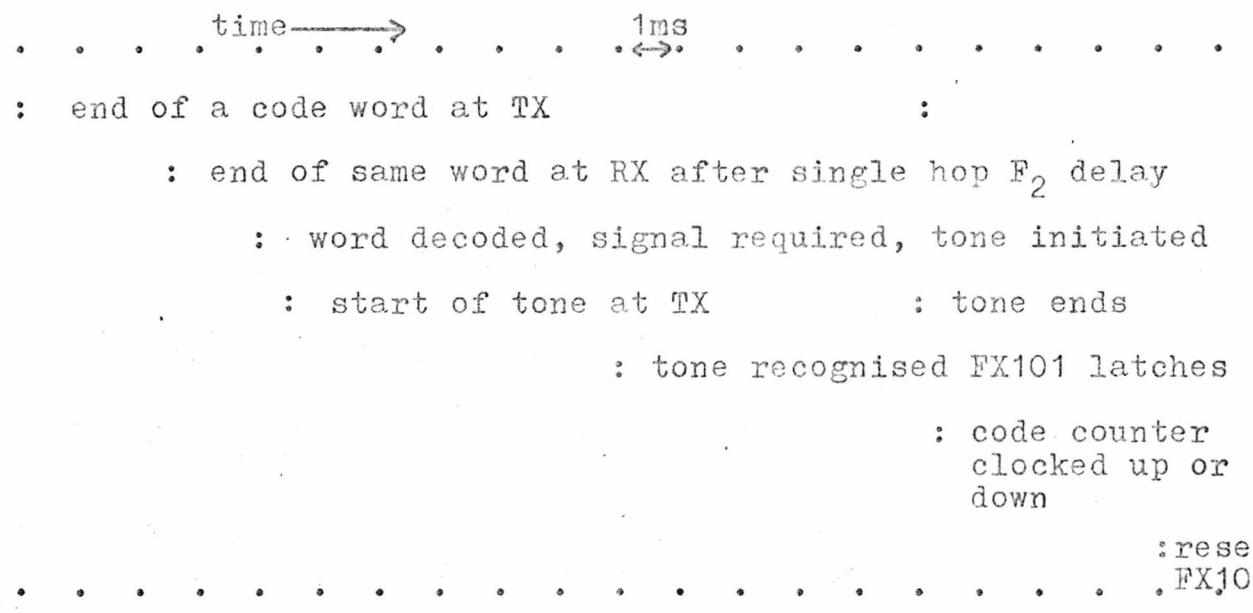


Figure 6.6

Feedback Timing



detected and must be externally reset. Figure 6.5 (Tx) shows the latched outputs which are then used to direct a trigger pulse to the up or down line of the reversible code control counter, via the three-input AND gates shown. If both circuits are latched, an error condition exists and the NAND gate disables the AND gates, thereby stopping any code change. The circuits are reset on the fall of fourth bit of each word and are then ready for further signals. The timing diagram (figure 6.6) shows that there is a margin of about 4ms for any additional delay, and that delay variations of this amount will not affect the correct operation of the circuit. The third FX-101 shown operates in the auto-reset inband mode (as used in the FSK demodulator) the output of which is filtered to remove short transients, and used to toggle the 3-code/8-code bistable.

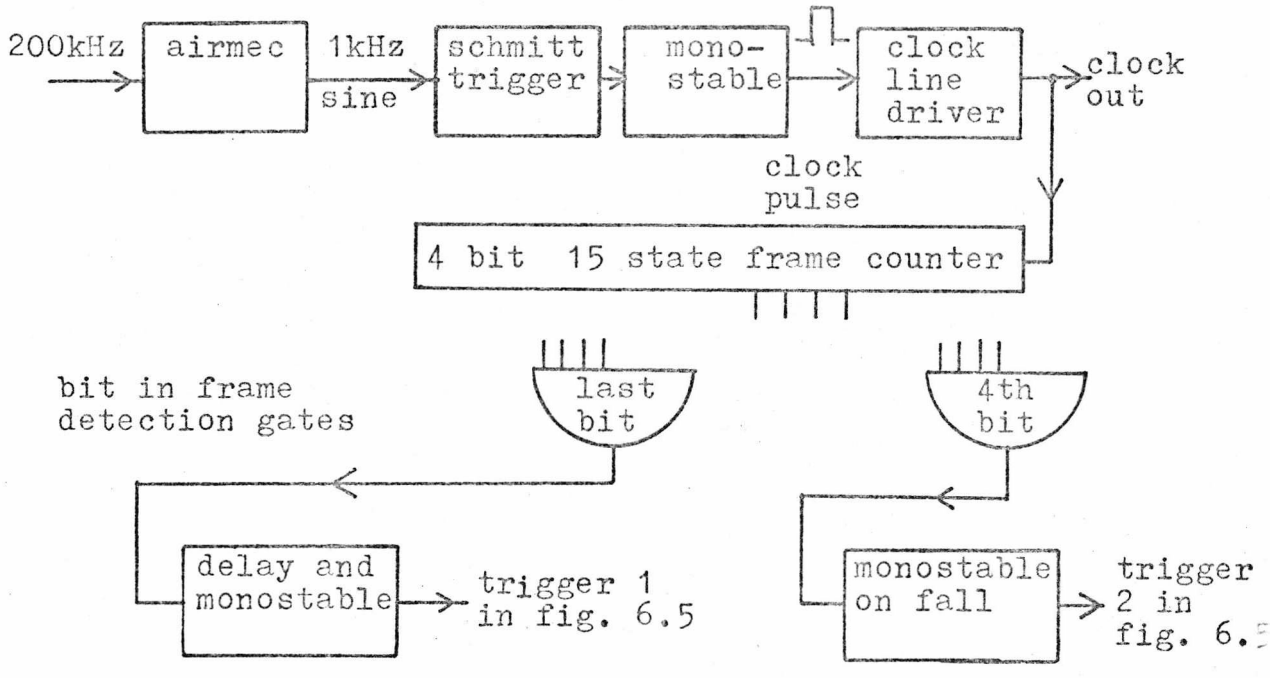
The feedback signalling system operated satisfactorily with one block delay operation, but was subject to relatively infrequent incorrect code changing due to both the non-recognition of transmitted tones, and the recognition of spurious signals when no tones were sent.

6.5 *Synchronisation and digit sampling/retiming*

Digit timing at both the transmitter and receiver was derived from two Airmec Type 311 frequency standards. These units contain very high stability crystal oscillators that are oven controlled and can be locked to the 200kHz Droitwich transmissions. The transmitter timing equipment is shown in figure 6.7.

Figure 6.7

Transmitter Timing

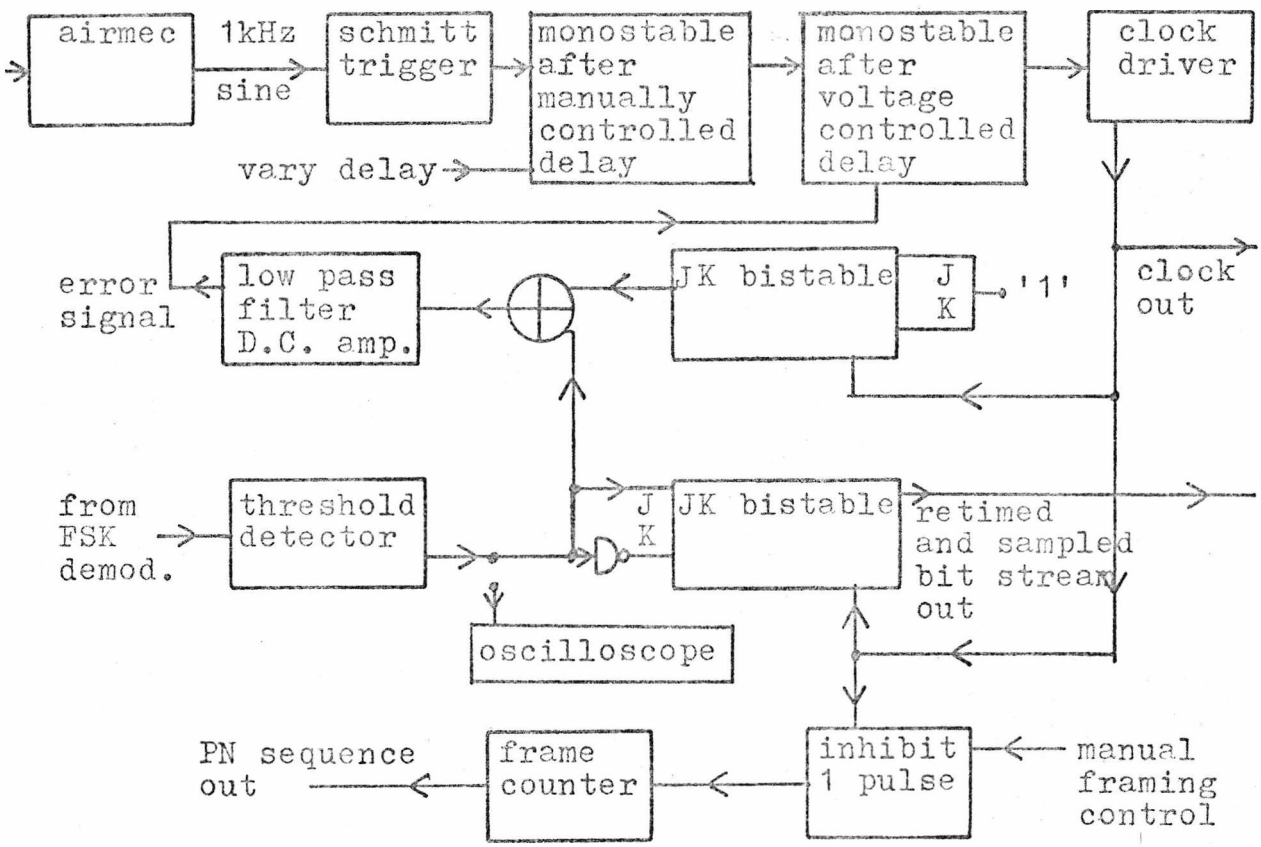


The frame counter is clocked at every digit so that each bit position in the word has a unique state allocated to it. These states can be detected with AND gates, thereby detecting position in the frame, and used to provide control pulses at appropriate points in the frame.

At the receiver the Airmec standard is used in conjunction with a phase lock loop (PLL) which tracks slow drifts in the phasing of the clock relative to the incoming bit stream. The receiver timing equipment is shown in figure 6.8 below.

Figure 6.8

Receiver Timing Equipment



The setting-up and operation of the equipment is as follows. The threshold detector output is disconnected and viewed on an oscilloscope together with the clock waveform. The negative edge of the clock (sampling point) is then visually aligned with the centre of each detected digit by using the manual delay control on the first monostable. This monostable is used to trigger the voltage controlled monostable which is then in its rest state with zero average error signal and ½ms delay. Under normal operating conditions the threshold detector output is connected. If the relative position of the sample point and the average position of bit centres then changes, the exclusive-OR gate and filtering system produces a positive or negative error signal which is used to counteract the original phase change, up to a maximum of ±½ms.

A four bit frame counter is also included in the equipment, for keeping track of position in the frame. The output of one particular bit of this counter is a 15 bit PN sequence which is displayed on an oscilloscope together with the retimed and sampled bit stream. If a known code word is then transmitted a visual comparison indicates whether or not the receiver is correctly framed. In order to initially set up (or recover) correct framing a manual control is used to 'slip' the framing by one bit each time a button is pushed, until the observed waveforms coincide.

6.6 *The VR encoder, and transmitter code control*

In order to simplify both framing and the subsequent analysis of the bit stream for off-line decoding, it was

decided that the VR encoder should be a simulating encoder. That is, for each code only one code word in the code is used, and continuous operation in a particular code is characterised by repeated transmission of the corresponding code word. The choice of code words was influenced by the following factors.

(i) Each code should have a different word, for ease of 'code in use' recognition.

(ii) Each word should have a roughly equal number of ones and zeros, in order to have the same kind of resistance to frequency selective fading as a random-data code word stream would have.

(iii) Each word should not be cyclic in less than 15 shifts, in order to easily recognise misframing.

Conditions (ii) and (iii) cannot be met for the (15,1) and the (15,2) codes so that when these codes were in operation each used two code words transmitted alternately. Table 6.1 below lists the transmitted code words.

Table 6.1

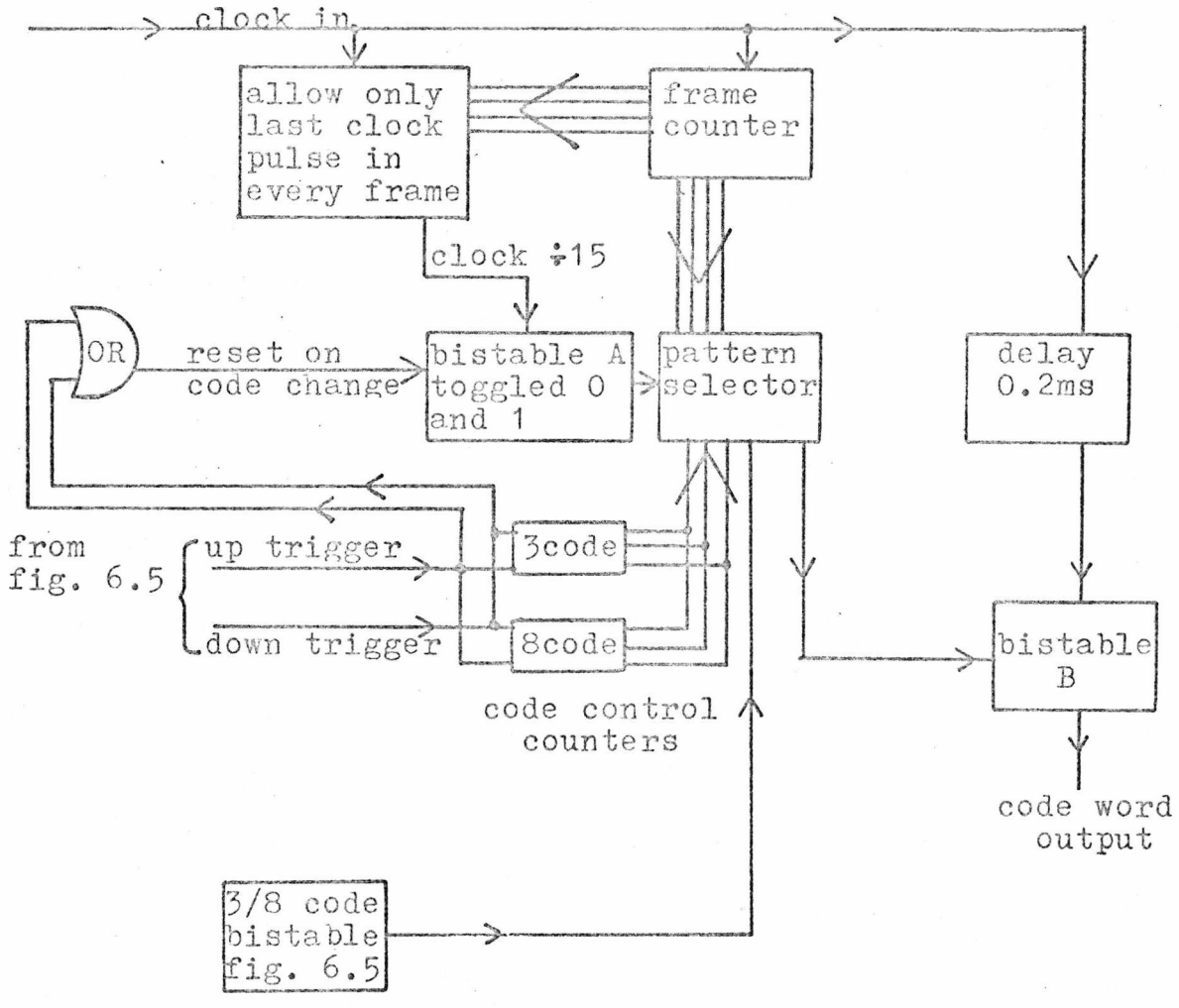
Transmitted Code Words

Code Control Counter	3 code VR	8 code VR	No of 1's	Word
111	(15,1)	(15,1)	0 and 15	000000000000000 and 111111111111111
110		(15.2)	10 and 10	0110110110110110 and 1101101101101101
3/8=1, 101		(15,4)	8	1111010110010000
3/8=0, 101	(15,5)		7	0000101001101111
100		(15,6)	6	0000010011100111
011		(15,7)	9	0111100101101010
010		(15,10)	8	1111010000111000
001	(15,11)	(15,11)	7	1110101000010010
000		(15,14)	8	1110010100101100

The code word generator and code control equipment is shown in figure 6.9 below.

Figure 6.9

Code Word Generator



The code word generator is based on a combinational pattern selecting network. The frame counter indicates the current bit in the word, and the pattern selector chooses the value of that bit according to its other inputs. These inputs come from the code control counters, the 3-code/8-code selector bistable, and a bistable that generates the alternate all-zero and all-one words required for controlling the

(15,1) and the (15,2) codes. This bistable is reset at every code change to ensure that when these codes are selected the first word is always the same.

The code control counters are reversible three-bit counters with limit stops at each end. That is, when the state corresponding to the most powerful code is in operation a further "UP" trigger will not change state. The converse procedure works for the least powerful code. When 8-code VR is selected the inputs from the 3-code counter are ignored, and the 8-code counter operates in straight binary with 111 for the most powerful code and 000 for the least powerful. If the (15,1) code is in operation the pattern output is simply the output of bistable A. When the (15,2) code is required the output code word is either of the two set patterns, depending on the value of bistable A's output. When 3-code VR is selected a similar procedure operates, with the code control counting sequence: 001, 101, and 111.

The pattern selector output is clocked into the output bistable B, with a slightly delayed clock. This ensures that the code change operation, which is triggered during the last bit of the word and after this delayed clock, will not cause out-of-sequence level changes at the final output due to the combinational network changing.

Because the codes chosen do not have disjoint code books, it is possible for the transmitter and receiver to operate with different codes, whilst still displaying a zero syndrome. Table 6.2 indicates which transmitted words are code words in more than one code. Coder-decoder mismatch

Table 6.2

Code Words and Corresponding Codes

k =	Transmitted Word in Code									
	1	2	4	5	6	7	10	11	14	
1	✓									
2		✓								
4			✓							Also a Code
5	✓		✓	✓						Word in this
6		✓	✓		✓					Code
7	✓	✓	✓	✓	✓	✓				
10		✓	✓		✓		✓			
11	✓	✓	✓	✓	✓	✓	✓	✓		
14		✓	✓		✓		✓		✓	

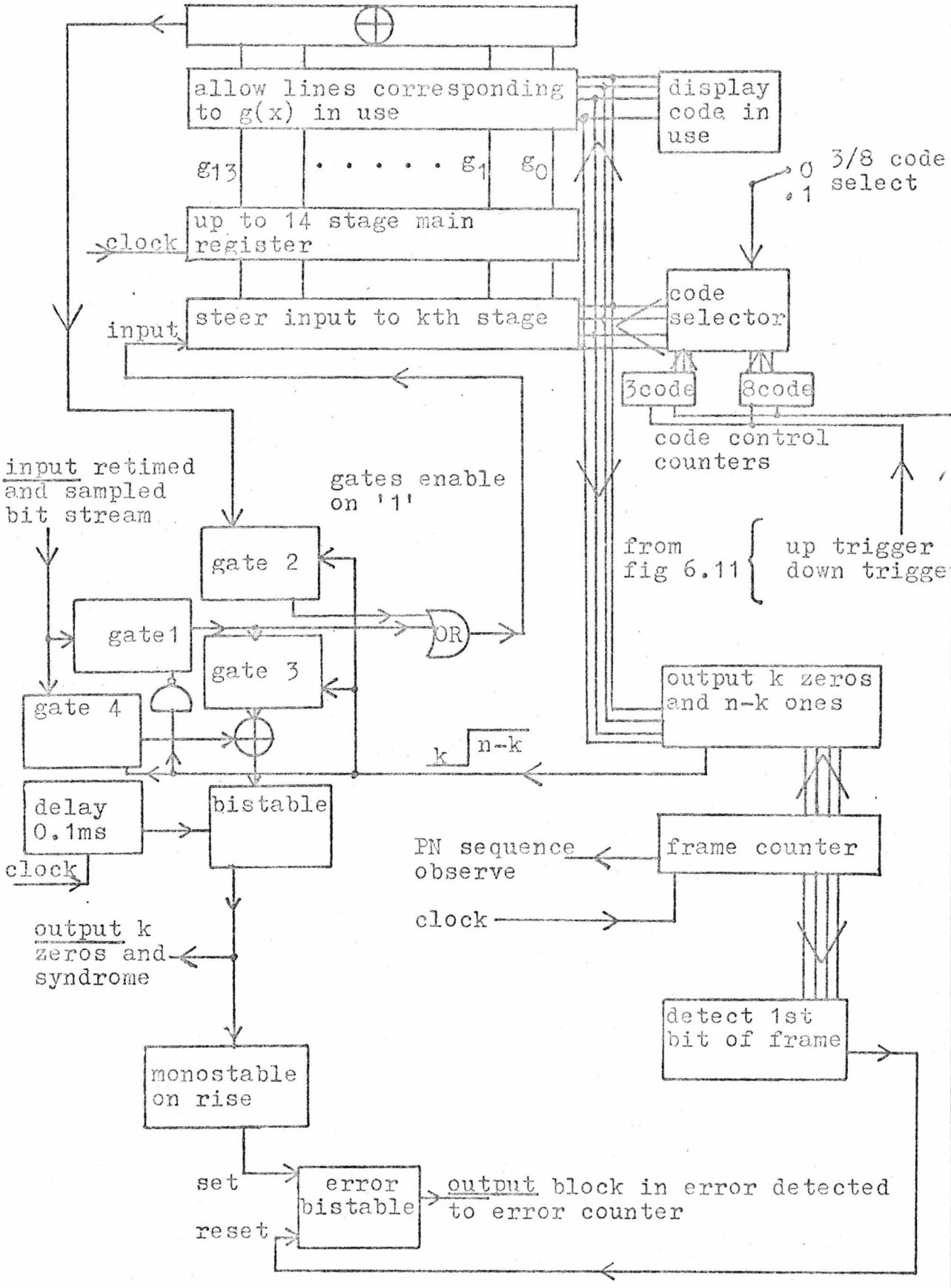
can occur because of a feedback signalling error, and although undetectable unless disjoint code books are used, can easily be detected with the experimental system because the use of single different words for each code enables the incoming word and the receiver code to be visually compared.

6.7 *The VR decoder and receiver code control*

The decoder is basically a (variable) k-stage syndrome calculation circuit of the figure 3.9 type, and is shown in figure 6.10 below.

The output of this circuit is again delay clocked into a bistable to isolate the syndrome from changes in the combinational networks. The error detection bistable is set during the last bit of the word at the latest (or not at all) and its

Figure 6.10 The VR Decoder



output is fed to the error counting unit. It is reset by a monostable pulse during the first bit of the next frame. Gates 1 to 4 are enabled in the correct sequence by a combinational circuit which produces k zeroes followed by (n-k) ones for every frame. The various combinational circuits are changed during the last bit and just before the start of a new frame. The code selection is governed by the code control counters which are either operated manually or automatically from the code change unit.

The length 15 PN sequence generated from the frame counter is displayed on an oscilloscope together with the retimed bit stream in order to check framing. The output on the code selector lines is displayed on illuminated bulbs and compared with the retimed bit stream to check for coder-decoder mismatch.

6.8 *Code change unit*

A single sampling code change philosophy was adopted initially and it was hoped to extend this to a sequential sampling scheme. This extension could not be achieved in the field trial time available and therefore single sampling only was tried.

The block error rate sample was obtained by running a 'total number of blocks' counter and a 'blocks in error' counter. The total blocks counter was stopped after the error counter indicated seven errors, and the resulting total count was compared to a set of threshold values which were pre-set by manual patching. If the lower

or higher threshold for the present code was exceeded a 'DOWN' or 'UP' signal was initiated.

The code change unit system diagram, together with its control flow diagram is shown in figure 6.11.

The threshold points were calculated from the performance curves for the length 15 codes (chapter 5), and consisted of seven 10 bit numbers for 8-code VR, and two 10 bit numbers for 3-code VR. If seven errors are counted the largest number of total blocks that must be counted, for P_e values similar to those of chapter 5, is considerably in excess of 1000. It was therefore decided to implement the largest threshold (the point at which the change to the lowest redundancy code occurs) by stopping the block counter at 1000 and examining the error counter for one or no errors. This enabled a large saving in register and comparator stages to be made whilst maintaining a flexibility in threshold choice for both 8-code and 3-code VR.

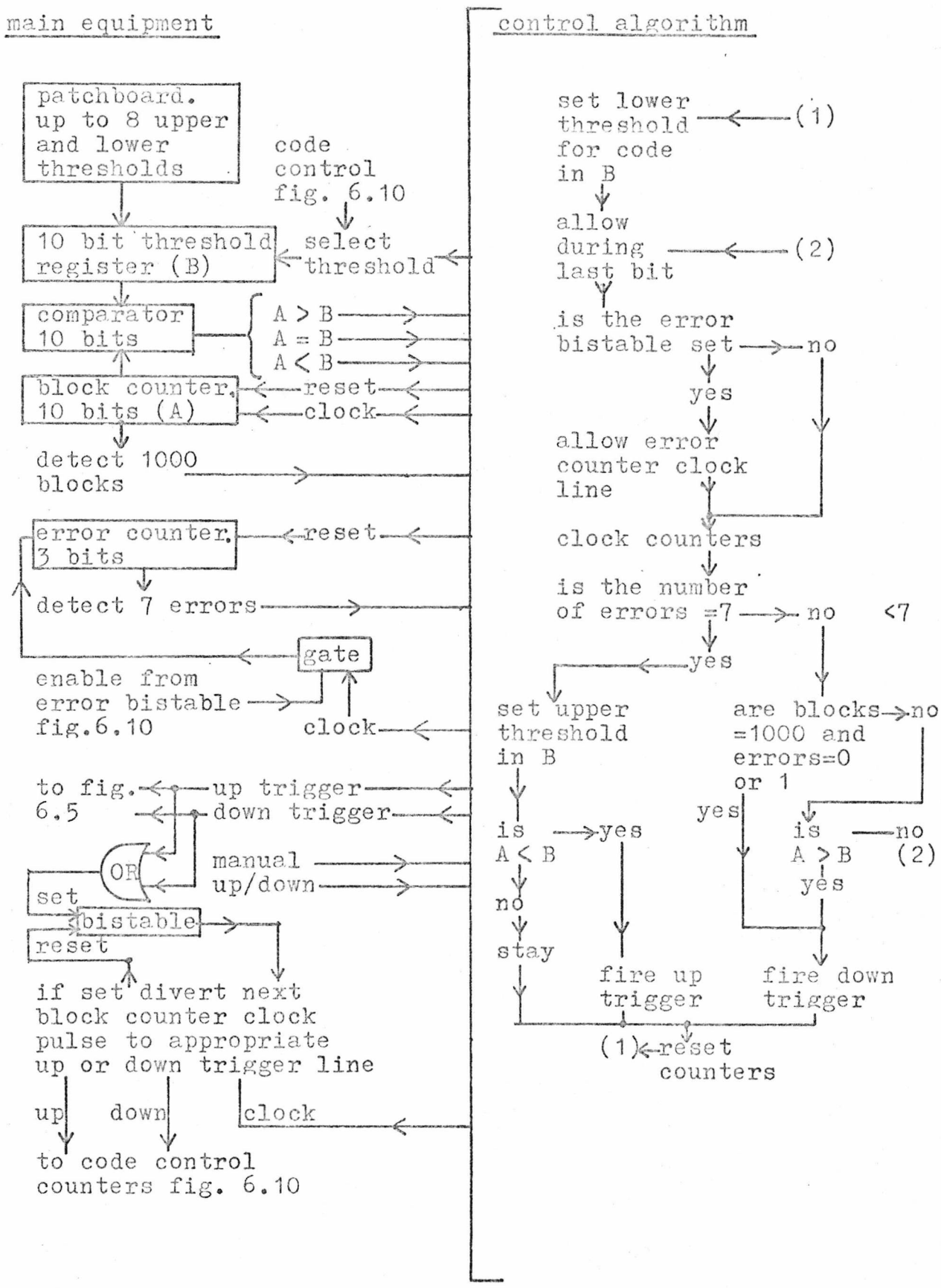
The unit also ensures that there is a one block delay between a 'change' initiation and actually changing the decoder circuits.

6.9 *Recordings and off-line decoding*

Results were collected by:simple error counts, PN sequence recordings, and automatic VR operation recordings. The equipment used was a Honeywell DDP-516 computer, a National 2-track tape recorder, two Racal 1 M bit counters, and suitable signal processing circuits for all of these.

Figure 6.11

Code Change Unit



6.9.1 *Error counting*

For these experiments either the PN sequence used for the (15,4) code or a 0,1 square wave was continuously transmitted and exclusive-OR compared with replica sequences at the receiver. The clock line was used to increment a total bits counter and also to sample the mod-2 sum at the output of the exclusive-OR, so that errors (ONES) allowed the clock to increment the error counter. The operation was manually stopped and started.

6.9.2 *Direct computer recordings*

PN sequence recordings were required for channel analysis and simulation of other VR systems. Depending on the availability of the computer the incoming PN sequence bit stream was fed on-line to the computer and recorded. The DDP-516 input interface was capable of inputting a 16 bit word at a rate of 5000 words per second with the program used. For these recordings, however, only one bit per word was used so that the bit stream was input serially in real time, with the decoder clock providing the required strobe pulses. The bit stream was then recorded serially on to magnetic tape as 15 bit words, and these tapes were later processed (by exclusive-OR comparison with a replica PN sequence) to provide compact paper tape and magnetic tape error sequence recordings.

6.9.3 *Tape recordings*

The majority of recordings were made directly onto ¼" magnetic tape using the National tape recorder, and later

transferred to the computer. A drive and replay unit was constructed that converted the TTL logic bit streams into a series of positive and negative pulses suitable for tape recording at 7 inches per second. On replay these pulses were detected and processed back into TTL level bit streams suitable for computer input. The errors introduced by this processing were found to be negligible. Two independent bit streams plus individual clocks (if required) could be recorded by this method.

PN sequence recordings were taken using one track for the incoming sequence and the other for the clock. These were later input to the computer and processed as in section 6.9.2.

VR recordings used one track for the incoming bit stream and clock, and the other for the 'code in use' signal. This signal was in fact the decoder framing sequence and consisted of k zeroes followed by (n-k) ones. On replay the clock was extracted and used as the input strobe for the computer. The VR bit stream occupied one bit per computer input word, and the framing signal occupied one other. These were later processed and recorded as a 15 bit incoming word followed by a 15 bit syndrome plus a 4-bit 'code in use' tag.

6.9.4 Off-line decoding

The experimental system basically operates in the VR-ED mode; by suitable programming, however, this can be easily extended to VR-ARQ and VR-FEC once the syndrome and 'code in use' signals are available to the computer.

In the case of VR-ARQ the computer inspects the syndromes and simulates an ARQ if a syndrome is non-zero. The 'information digits delivered to sink' counter is not incremented and the next word is dumped to allow for feedback delay. The code word after this is then assumed to contain the repeated information digits. The ARQ cycle then continues in the normal manner and is independent of the VR system. Because known words are being transmitted, each all-zero syndrome can be checked against its corresponding incoming word, thereby enabling decoding errors, and 'erroneous information digits delivered to the sink' to be counted.

For off-line FEC decoding with 3-code VR, each code was simulation-decoded in a different manner for ease of implementation. The (15,1) code was majority decoded. The (15,5) code was decoded by comparing the received word with an internally stored table of the 32 code words and choosing the nearest. The (15,11) code was decoded by comparing the syndrome to an internal list of the 15 single-error syndromes, and inverting the corresponding bit. In each case the (supposedly) corrected word can be compared to the known transmitted word to count erroneous decodings and sink digits.

6.10 Field trials

Four field trials were undertaken during the period August, 1970 to May, 1971. The equipment used was developed over this period and table 6.3 below summarises equipment used and results taken for each test.

Table 6.3

Field Trial Programme

Trial No.	1	2	3	4
Dates	3.8.70- 7.8.70	31.8.70- 3.9.70	15.1.71- 1.2.71	15.3.71- 10.5.71
Modulation	OOK	FSK	FSK	FSK
Feedback link	-	-	✓	✓
Encoder	-	-	✓	✓
Decoder	-	-	-	✓
Pen recordings	-	✓	✓	✓
Error counts	-	✓	✓	-
PN seq. recording	-	-	✓	✓
Automatic VR	-	-	-	✓

Comments: Trial 1:- Initial setting up and observation of pulsed carrier at various frequencies.

Trial 2:- FSK tested on square wave modulation.

Trial 3:- Manual code change tested.

Trial 4:- Propagation very variable over test periods, several complete HF blackouts. Full system operating.

6.11 *Computer analysis and simulation*

The error sequence runs obtained from the PN sequence recordings were analysed to provide channel statistics. Some of these statistics were used as parameters with which several random channels and burst channels were modelled. These channels were then processed to provide further error sequence runs.

The real and modelled error sequence runs were then used in the simulation of several fixed and VR coding schemes, so that the performance of these schemes over real

and simulated channels could be assessed.

The automatic VR recordings were separately processed to provide performance results for the experimental system.

CHAPTER 7

Results and Discussion

This chapter presents and discusses the results of field trials and computations on VR coding systems. The experimental results collected are outlined first and their development noted. Observed propagation conditions are then described, and this is followed by quantitative results that characterise the channel. The performance of various fixed and VR coding systems on the observed and simulated channels is given next. Finally, the performance of the experimental VR system is presented, together with those of other simulated schemes for comparison.

For the sake of clarity in the following tables, numbers containing powers of ten will be presented as mantissa, sign of the exponent, exponent. For example: 3.23×10^{-3} is shown as 3.23-3.

7.1 Experimental results and their development

The basic results collected during the field trial periods were as follows:-

(i) Photographs of OOK soundings to show multipath effects (Test 1, qualitative).

(ii) Tape recordings of receiver BFO output to show frequency selective fading effects on incoming FSK (Test 2, qualitative).

(iii) Pen roll recordings of received power (Tests 2, 3 and 4).

(iv) Error rate counts (Tests 2 and 3).

(v) PN sequence recordings (Tests 2, 3 and 4).

(vi) Automatic VR system recordings (Test 4).

The quantitative results were then developed in the following way.

(i) Error counts only give a rough idea of channel conditions, but were correlated with received power and fading rate results, obtained from the pen roll recordings.

(ii) The PN sequence recordings were processed to provide channel statistics. Twenty-one runs (each lasting about three quarters of an hour) were recorded, and 11 of these were selected for detailed analysis. The channel statistics so obtained were used to model three random and eleven burst-error channels. The 11 observed runs and 14 modelled runs were then used as digital channels in the simulation of both fixed and VR coding schemes.

(iii) The performance of the experimental VR system (operating in VR-ED) was obtained for 19 runs, and VR-ARQ, and VR-FEC results were computed by means of off-line decoding. The 19 runs were then used as digital channels in the simulation of several fixed and VR coding schemes, whose performance is then compared to that of the experimental system.

7.2 *Propagation conditions*

The choice of 7.375 MHz for the carrier frequency turned out to be a good one given that the tests spanned 9 months during which the MUF varied from about 7 to 10 MHz. The LUF did not affect the trials.

Night reception (7 pm to 2 am) at this frequency was normally impossible due to broadcast interference and low MUF. During the early hours of the morning (2-7 am), the MUF limited propagation. The signal usually appeared from out of the noise at about 9 am, and daytime reception was generally adequate (received signal strength ≈ 0.15μV/m). Flat and selective fading was continually present to varying degrees, with occasional interference from ground-to-air and teleprinter traffic. The usable signal time (bit probability of error $<10^{-1}$) averaged out at about 30% of the 24 hours during the trials. Several complete propagation blackouts occurred during test 4, and could have been connected with large sunspots that were observed at the time.

Multipath effects were observed when transmitting OOK at 1:10 mark/space and 1000 bauds. Strong echoes were observed at about 1 ms delay, with weaker echoes at up to 5 ms. Fading of the main pulse and echoes varied, but was often periodic with periods of between 0.5 and 5.0 s. The main pulse and strong echoes were generally observed to fade in an uncorrelated manner.

Frequency selective fading with the tones fading independently, and in antiphase, was observed. In the latter case, exact antiphase fading produced a steady trace on the pen recorder (steady *average* received power) which lasted for up to 10 minutes before getting out of phase and producing normal periodic fades on the recorder. Periods of about 0.5 to 60 s were observed for antiphase fading.

7.3 Error rate counts

Fifty-three runs of FSK reversals transmission were observed. The lowest error rate (apart from two error-free tests) is 6.7×10^{-6}, and the highest (excluding runs with error rates $>10^{-1}$) is 2.33×10^{-2}. The average error rate over all usuable runs is approximately 1×10^{-3}.

Table 7.1 shows bit error rates for 16 of these runs, together with power and fading data obtained from simultaneous pen recordings.

Table 7.1

Bit Error Rates (Reversals Transmission)

Run No.	Duration (mins)	Bit error rate	Average Receiver power (db above 1μV)	FADING RATE across average (/min.)	FADING RATE across 10db down from average (/min) (e)
1a	37	1.66-3	27	0.46	0.2
2b	33	6.70-6	31	0.06	0
3b	31	2.20-5	31	0.03	0
4c	30	1.10-3	30	0.26	0.17
5	39	9.70-3	43	0.3	0
6	10	1.40-4	33	0.2	0
7	10	2.33-2	33	0.2	0
8	10	9.00-4	26	0.6	0.3
9	11	4.90-3	26	5.2	1.5
10d	11	8.00-3	27	11.8	2.45
11d	10	1.10-2	29	4.7	3.1
12	12	1.52-3	26	3.58	1.16
13	12	6.80-6	32	0.46	0.17
14	10	1.80-4	32	0.1	0.1
15	11	2.10-3	28	2.0	0.27
16	11	5.40-3	28	5.8	2.6

a - *flat fading*
b - *steady received power*
c - *some voice interference*
d - *selective fading*
e - *gives an indication of very deep fading*

The correlations between received signal power and error rate, and fading rate are poor; even though the runs selected were chosen to be relatively free of interference, so that propagation effects were the main cause of errors.

The error counts reveal that error rates vary rapidly and are generally greater under selective fading conditions than under flat fading conditions, and that interference can have a worse effect than deep fading.

7.4 *Channel statistics from PN sequence recordings*

The PN sequence recordings were analysed to provide both channel statistics and digital channels for simulation. Twenty-one 45-minute runs, with error rates as low as 10^{-5}, and as high as 0.5 were recorded. The former are of little use in assessing code performance, and the latter make the received data meaningless; these extreme runs were therefore rejected. In addition, runs with very bad sync slip (quite common because of the high baud rate used and the multipath experienced) were also rejected. The eleven runs selected for further analysis (nos. 17-27) correspond to conditions of slight to moderate multipath (with no strong first echo), and are in the middle of the error rate range of all runs.

The error statistics of these channels were also used to model three random (BSC), and eleven burst (two-state Gilbert) channels. These channels are designated runs 28r-30r, and 31b-41b respectively.

7.4.1 Bit and block error rates

Table 7.2 shows bit and block error rates for the selected runs, in order of increasing block error rate.

Table 7.2

Error Rates for Selected Runs

Run	Bit error rate	Block error rate (n=15)
17	6.23-3	3.48-2
18	8.02-3	3.68-2
19	1.93-2	5.99-2
20	1.01-2	6.90-2
21	1.66-2	7.72-2
22	2.15-2	0.101
23	1.87-2	0.102
24	3.47-2	0.118
25	3.62-2	0.162
26	5.63-2	0.172
27	5.38-2	0.281

The details for the selected tests are:

lowest BER = 6.23-3 (1 in 161), run 17

Average BER = 2.56-2 (1 in 39)

highest BER = 5.62-2 (1 in 18), run 26.

The generally high bit error rates observed show that the high baud rate used is very susceptable to intersymbol interference caused by multipath.

Table 7.3 shows bit and block error rates for the computer generated channels, in order of increasing bit error rates.

Table 7.3

Error Rates for the Model Channels

Run	Bit error rate	Block error rate
28r	1.56-2	0.210
29r	3.12-2	0.379
30r	6.24-2	0.619
31b	1.15-3	1.07-2
32b	8.80-3	3.56-2
33b	9.31-3	8.91-2
34b	1.22-2	5.99-2
35b	1.23-2	0.103
36b	3.01-2	0.111
37b	3.16-2	0.129
38b	3.50-2	0.119
39b	6.84-2	2.51-2
40b	7.24-2	0.341
41b	9.87-2	0.28

The burst channels with an unusually low block error rate (39b, 34b) for the given value of bit error rate, are very 'bursty' channels in which the errors are very bunched and therefore only affect a relatively few blocks.

7.4.2 The distribution of consecutive errors

Figure 7.1 shows consecutive error distributions for some of the real and simulated runs. Figure 7.1 shows that the channels have a burst nature, because the probability of consecutive ones is much greater than that of a BSC with an equivalent error rate. Run 20 behaves most like a BSC and all other runs have ordinates above those for this run, showing that they are more 'bursty'. Run 17 displays about average burstyness whilst runs 19, 24 and 27 are very bursty.

Figure 7.1

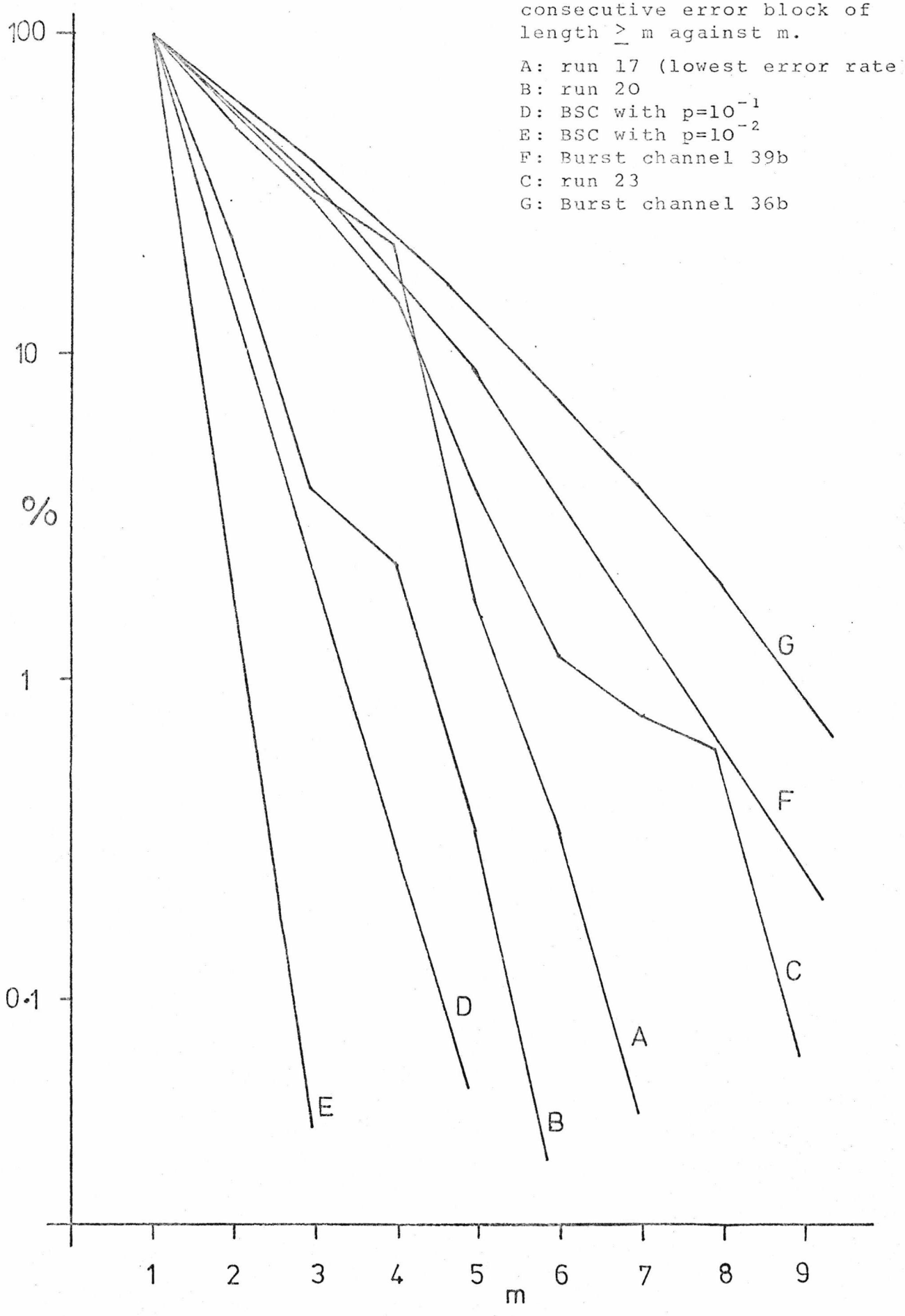

Percentage of errors in a consecutive error block of length ≥ m against m.
A: run 17 (lowest error rate
B: run 20
D: BSC with p=10^-1
E: BSC with p=10^-2
F: Burst channel 39b
C: run 23
G: Burst channel 36b
100
10
%
1
0·1
1
2
3
4
5
6
7
8
9
m
G
F
C
A
B
D
E

Figure 7.1 also shows that the proportion of values for consecutive runs of length 2, 3 and 4 are relatively higher than those for m=5 to 10; this is due to the effect of multipath on the PN sequence. Multipath causes delayed digit trains to be summed thereby producing error sequences that are replicas of the PN sequence, and each replica contains 2 bursts of length 1, 1 of length 2, and 1 of length 4. The values for these lengths are therefore inflated, and the runs most affected by multipath have a large ordinate difference between m=4 and m=5 (runs 18, 19, 22, 26).

7.4.3 Gap distributions

The error-free gap distributions shown in figure 7.2 further reveals the burst nature of the channel. In comparison to the equivalent BSC curve, it can be seen that the runs have more frequent long gaps and less frequent short ones. This flattening of the gap curve is a characteristic of burst channels. The crossover point of the BSC curve and the pooled results curve (A) occurs at 200 bits. This indicates that the runs are characterised by having short periods (<200) of high error density, separated by relatively frequent long (>200) error-free lengths.

7.4.4 P(m,n) distributions

Several P(m,n) distributions are shown in figure 7.3. The curves show that, over the given range of block length, probabilities do not change as much as in the BSC case. This is again due to the burst nature of the channel.

Figure 7.2

Gap Distributions

A: All 11 runs (17-27) pooled
B: run 18
C: run 20
D: BSC with p=average BER of runs (17-27)

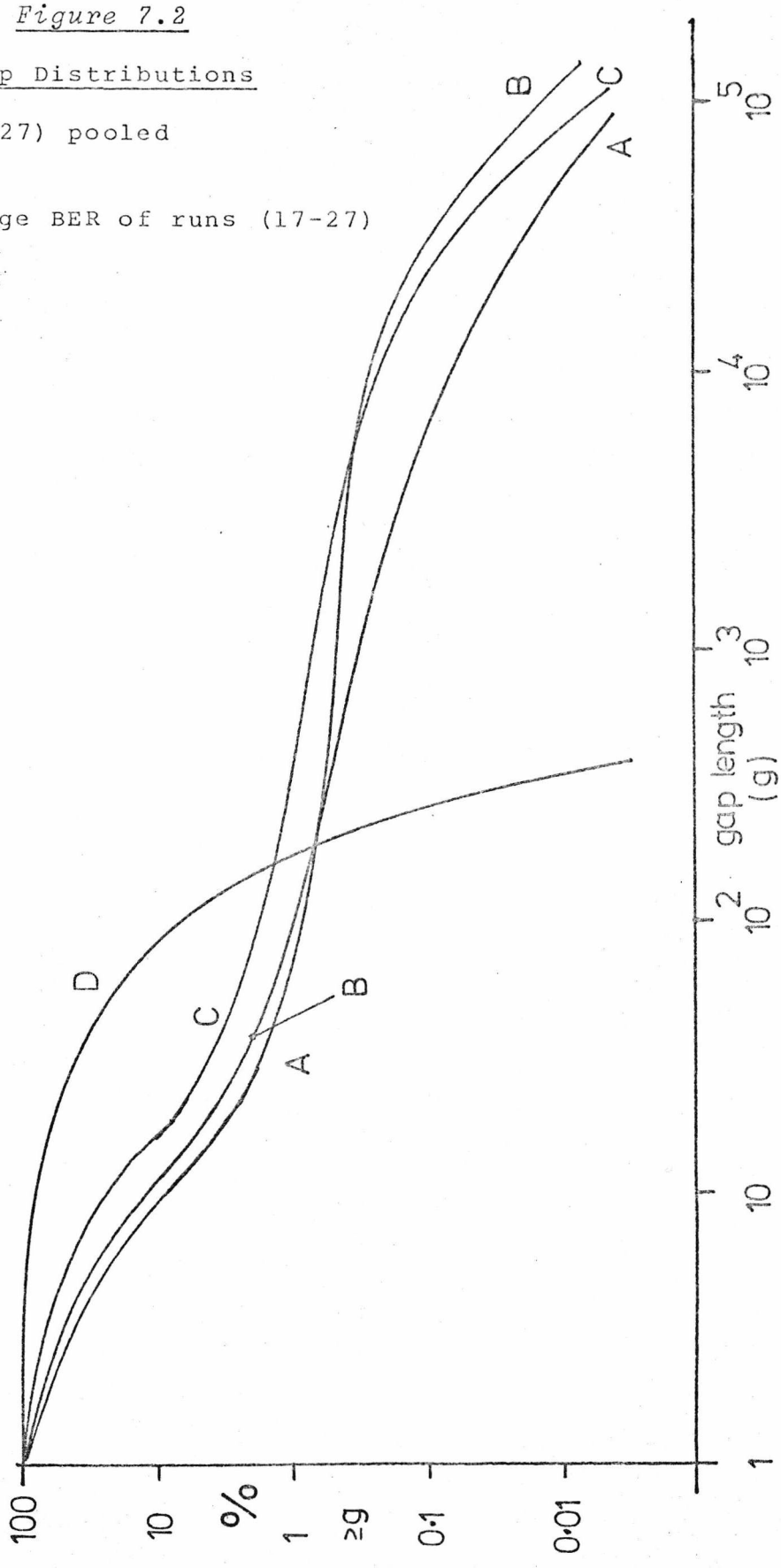
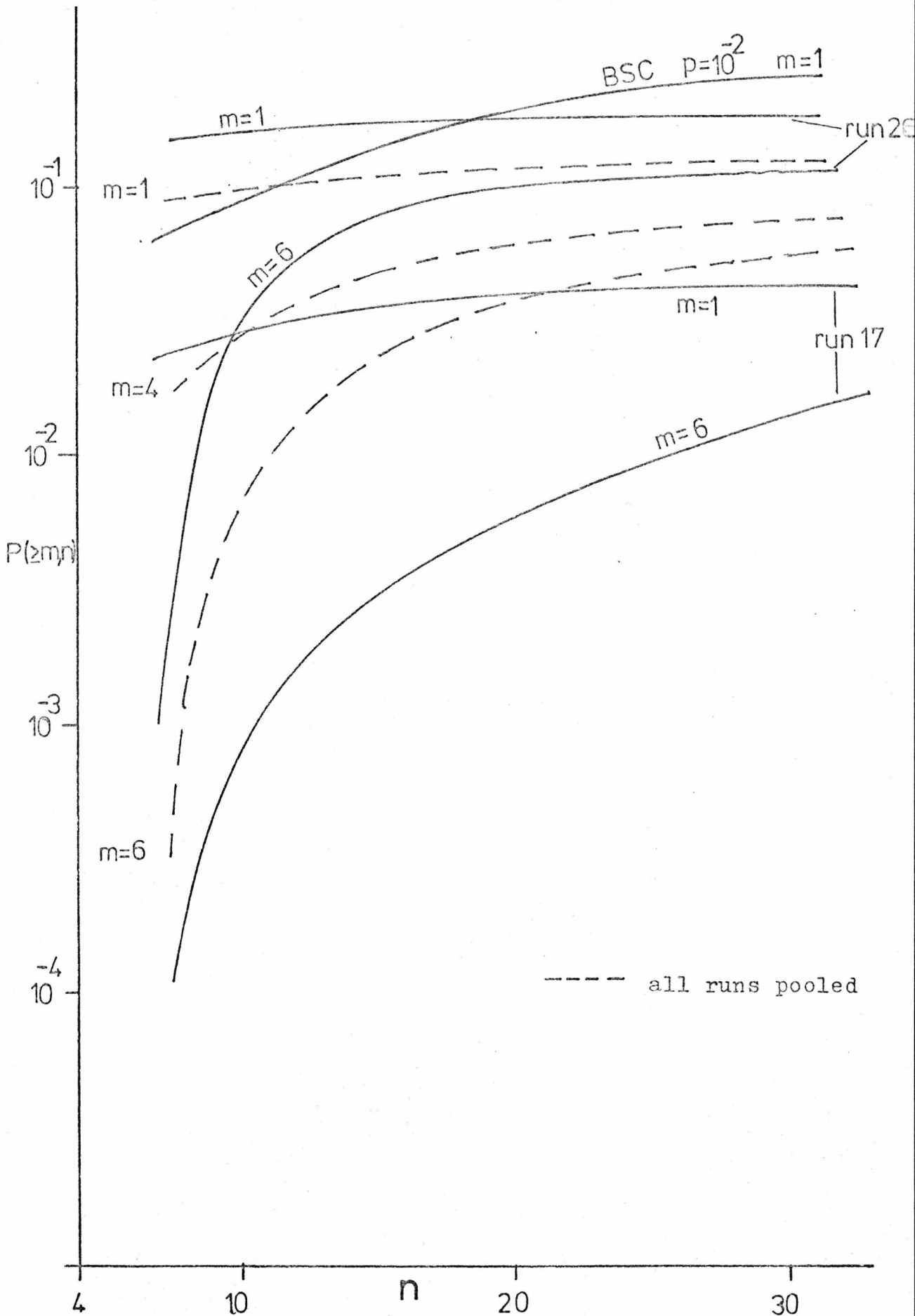


Figure 7.3

Probability of m Errors in a Block of Length n.



7.4.5 *Block error rates*

Block error rate versus block length curves are shown in figure 7.4. The BSC curves have a greater slope than any of the runs, showing that error rate increases rapidly with block length. The test runs, on the other hand are bursty, so that error rate is high at low block length, and therefore increases only slowly with increasing block length.

7.5 *Performance of simulated coding systems*

The 11 observed runs and 14 modelled runs were used as digital channels in the simulation of various fixed and VR systems. The performance of these systems when operating in various error control modes is presented in this section.

7.5.1 *Error detection using fixed redundancy codes*

Table 7.4 gives computed sink block error rates for the nine n=15 cyclic codes (efficiency E) when these codes are simulation-used over the observed runs. The results indicate:

(i) performance is best under random channel conditions (run 20).

(ii) good performance is possible under moderate burst conditions (runs 17, 23).

(iii) performance is worst under heavy burst and multipath conditions (runs 19, 26).

In general the cydic codes used are very susceptible to the sync-slip and multipath errors experienced during the field trials. Table 7.4 also shows the average improvement

Figure 7.4

Block Error Rates

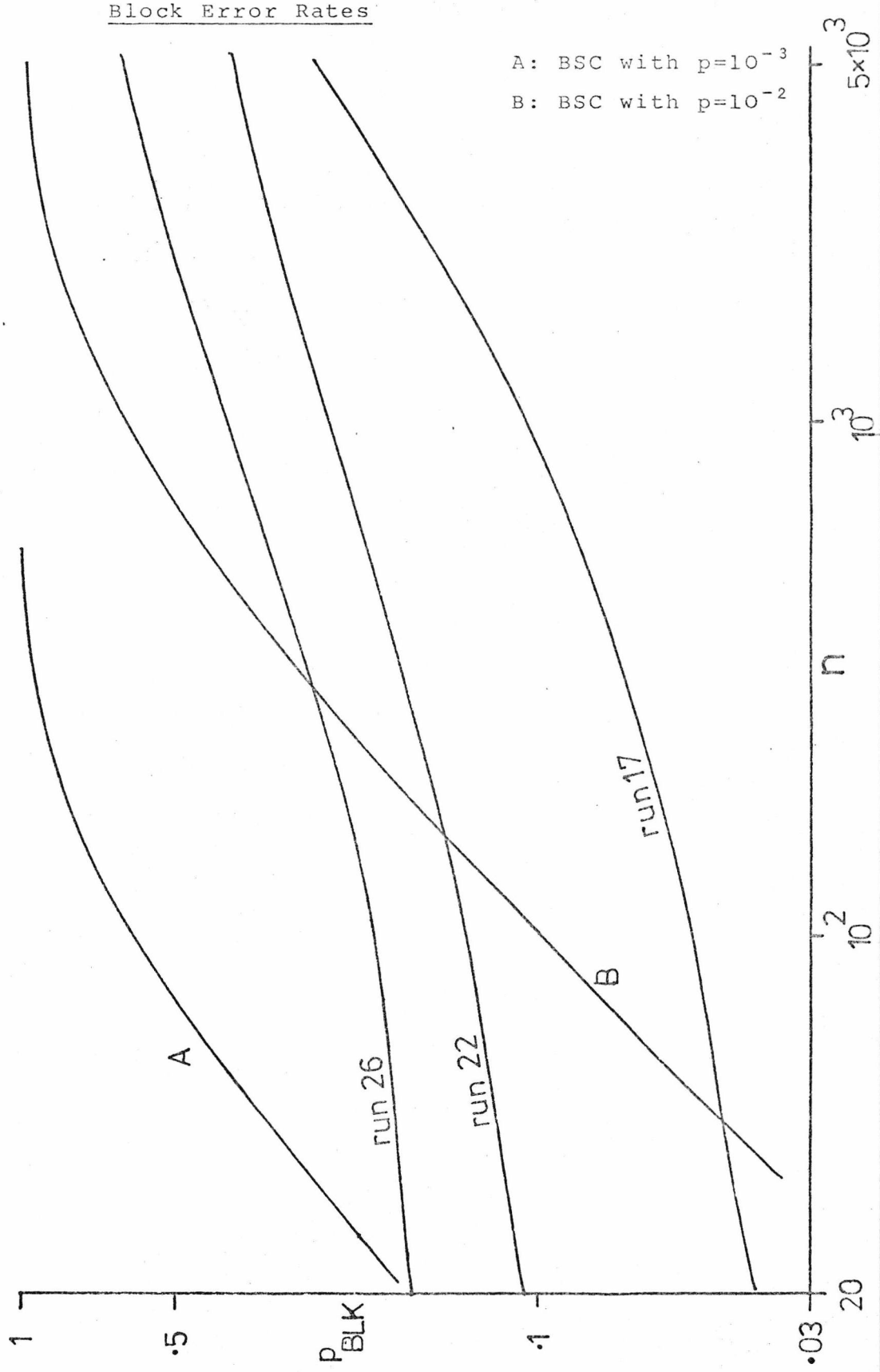


Table 7.4

Fixed Redundancy Error Detection

Run	(15,14)	(15,11)	(15,10)	(15,7)	(15,6)	(15,5)	(15,4)	(15,2)	(15,1)
17	1.6-2	5.8-4	2.1-4	5.4-5	5.4-6	4.8-5	0	0	0
18	1.6-2	3.5-3	3.4-3	3.2-3	3.2-3	3.2-3	3.2-3	0	0
19	4.0-2	9.5-3	9.5-3	9.1-3	9.1-3	9.1-3	9.1-3	0	0
20	2.5-2	3.4-4	2.2-4	0	0	0	0	0	0
21	3.5-2	2.6-3	2.1-3	1.7-3	1.6-3	1.6-3	1.6-3	0	0
22	4.9-2	4.0-3	3.7-3	3.6-4	3.5-4	3.6-4	3.5-4	0	0
23	4.0-2	3.4-3	2.4-3	7.4-5	1.7-5	5.7-5	1.1-5	0	0
24	5.7-2	2.9-2	2.2-2	2.1-2	1.5-2	2.1-2	1.5-2	0	0
25	7.5-2	2.2-2	2.1-2	4.2-3	4.2-3	4.2-3	4.1-3	0	0
26	9.8-2	5.0-2	4.9-2	4.5-2	4.5-2	4.5-2	4.5-2	0	0
27	1.0-1	1.1-2	3.4-3	3.6-3	1.7-4	3.3-3	1.0-4	0	0
F	2.2	8.9	11	14	15	14	16	∞	∞
E(%)	93	73	67	47	40	33	27	13	6.7
Interleaving degree 2									
17	1.4-2	8.0-4	3.5-4	0	0	0	0	0	0
19	1.9-2	7.2-4	5.2-4	1.1-5	0	0	0	0	0
20	2.5-2	1.0-3	2.1-4	5.8-6	0	0	0	0	0
22	3.7-2	1.2-3	7.1-4	4.1-4	4.1-4	0	0	0	0
26	5.6-2	4.4-3	2.3-3	4.3-5	2.1-5	1.0-5	0	0	0
27	1.0-1	7.4-3	3.4-3	2.6-4	7.5-5	4.0-5	1.0-5	0	0
Interleaving degree 3									
17	1.1-3	5.8-4	3.0-4	5.4-6	5.4-6	0	0	0	0
22	3.6-2	1.7-3	8.0-4	9.7-5	0	9.2-5	0	0	0
26	9.1-2	2.8-3	1.3-3	5.8-5	1.6-5	5.3-6	0	0	0

(over uncoded transmission) in block error rate (F) given by these codes over all 11 runs. That is, F=average sink block error rate/average block error rate.

Also shown in table 7.4 is the effect of interleaving to degrees 2 and 3. A general improvement in the case of the burst runs is noted; but not in the case of the random-like run 20. Interleaving to higher degrees (up to 10) did not give correspondingly better results, because of multipath affecting many successive blocks, and the nature of the transmitted sequence.

7.5.2 *Error detection using 2-code VR*

The VR scheme used consists of changing to a more powerful code when an erroneous block is detected, and back to the less powerful (but more efficient) code when a block is error free. A one block feedback delay is affected before making the change.

Table 7.5 shows sink block error rates (U) and efficiencies (E) for the two code sets used. A substantial improvement in both efficiency and sink error rate, over the fixed code case (table 7.4) is noted. This is particularly so for the high error rate cases. The effect that VR has of reducing the range over which U varies, for given bit error rate variations (chapter 5), can also be seen.

A comparison simulation using the 3-out-of 7 ARQ code was tried, and this achieved an average block improvement of 20 over all 11 runs, but at the low average efficiency of 43%.

Table 7.5

Performance of 2-code VR-ED

Run	(15,14)(15,1)		(15,11)(15,1)	
	U	E(%)	U	E(%)
17	4.0-3	91	1.6-5	71
18	3.8-4	90	0	71
19	4.0-3	88	1.6-5	69
20	1.7-3	87	1.7-5	69
21	2.6-3	87	1.2-5	68
22	6.1-3	85	2.7-5	67
23	4.4-3	85	2.8-5	67
24	5.3-3	83	1.6-3	66
25	2.2-3	79	1.1-5	62
26	6.3-3	79	1.2-4	62
27	2.0-2	69	5.5-4	54
Average of the Above				
	5.3-3 (F=21)	84	2.2-4 (F=506)	66
28r	1.5-2(14)	70	1.2-4(1718)	55
36b	3.0-3(37)	81	1.1-4(971)	63
39b	7.2-3(3.5)	91	4.9-4(51)	71
Interleaving degree 2				
17	6.3-3	91		
22	8.4-3	85		
26	1.1-2	80		
Average of the Above				
	8.6-3 (F=20)	86		

Also shown in table 7.5 are sink block error rates with improvement factors (bracketed) for one random, and two burst channel models. For the burst channels, improvement is best for the very bursty run 36b, performance on the random channel improves when the (15,11) code is used; but at the expense of greatly reduced efficiency.

Interleaving is not generally useful because this type of VR scheme relies on channel memory for correct operation.

7.5.3 *Error correction by ED-ARQ with fixed and variable redundancy*

In this case the scheme consists of requesting a repeat (and increasing redundancy for VR-ARQ) every time an erroneous block is detected. Redundancy is reduced on reception of a correct block. A four-block ARQ and code change delay was assumed, during which no repeat or code change signals are generated.

Table 7.6 shows sink bit error rate(S) and sink throughput efficiency (E) for various code combinations when used on runs 17, 22 and 26. In almost every case, the use of variable redundancy improves the sink bit error rate substantially at the expense of only a few percent drop in efficiency.

The codes marked (*) were calculated assuming a one block ARQ/code change delay. This has the effect of increasing efficiency and sink error rate slightly, indicating that

Table 7.6

Performance with ARQ

odes	Run 17		Runn 22		Run 26	
	S	E(%)	S	E(%)	S	E(%)
15,14)	1.9-3	90	9.0-3	85	3.4-2	81
15,14)(15,11)	5.2-4	89	1.2-3	83	2.7-2	79
15,14)(15,1)	4.6-4	89	5.1-4	80	6.0-4	73
15,14)(15,11)(15,1)	4.4-4	89	4.8-4	80	6.2-4	73
code	4.1-4	88	3.4-4	79	5.8-4	72
15,11)	7.9-5	70	6.7-4	64	2.6-2	62
15,11)(15,10)	1.8-5	70	5.7-4	64	2.6-2	62
15,11)(15,1)	0	69	0	63	1.2-6	57
15,10)	2.0-5	64	5.0-4	59	2.9-2	57
15,7)	8.8-6	45	1.1-4	41	2.7-2	40
15,14)(15,1) *	5.9-5	70	7.7-4	82	8.2-4	76
15,11)(15,1) *	0	70	1.7-6	64	1.2-6	59
code *	4.9-4	89	5.3-4	81	6.4-4	74

odes	Average over all 11 runs (17-27)			Average over all random models (28r-30r)			Average over all burst models (31b-41		
	S	F	E(%)	S	F	E(%)	S	F	E(%)
15,14)	9.8-3	(2.6)	83	2.1-2	(1.7)	30	8.8-3	(3.3)	71
15,14)(15,11)	4.6-3	(5.5)	81	2.5-3	(15)	21	3.9-3	(7.5)	66
15,14)(15,7)	3.9-3	(6.6)	80	2.0-3	(18)	15	3.6-3	(8.1)	62
15,14)(15,1)	5.1-4	(71)	79	6.7-3	(5.5)	7	5.2-3	(5.6)	55
15,14)(15,11)(15,1)	2.8-4	(91)	79	1.1-3	(35)	7	3.5-3	(8.3)	59
code	5.0-4	(51)	78	1.8-5	(2068)	3	2.0-3	(15)	58
15,11)	4.4-3	(5.8)	64	1.1-3	(35)	19	6.8-4	(43)	53
15,11)(15,10)	4.0-3	(6.4)	64	2.4-4	(150)	18	5.9-4	(49)	52
15,11)(15,1)	3.9-6	(6591)	62	2.7-4	(136)	5	2.6-4	(110)	42
15,10)	4.1-3	(6.2)	58	2.0-4	(180)	18	3.0-4	(96)	49
15,7)	3.7-3	(7)	41	2.8-6	(13118)	12	2.6-5	(1119)	34

channel 'memory' is generally greater than one block.

Also shown in table 7.6 are sink error rates, and sink bit improvement factors(F), averaged over all runs in each of the three types of channel (real runs, random models, and burst models). In the case of the real runs (17-27) the use of VR is clearly advantageous, and this is also true to a lesser extent for the burst model channels. VR (with this scheme) is not useful on the random channels, as expected.

7.5.4 *Forward error correction with fixed and variable redundancy*

In this case the VR-FEC scheme again consists of increasing redundancy on detection of an erroneous block. A three block feedback signalling delay is assumed. A 'down redundancy' signal is sent on reception of a correct block, but the encoder only acts on reception of two consecutive 'down' signals, in order to slightly favour the use of high redundancy codes.

Table 7.7 shows sink bit error rates(S) and efficiencies(E) for various code sets on channels 17, 22 and 26. Variable redundancy can be seen to improve the sink error rates and efficiencies in most cases.

Simulations with different feedback delays, and 'down' delays, were also run. In general these indicate that increased 'down' delay slightly decreases both error rate and efficiency, whilst increased feedback delay increases sink error rate but does not really affect efficiency.

Table 7.7

Performance with FEC

Codes	Run 17		Run 22		Run 26	
	S	E(%)	S	E(%)	S	E(%)
(15,11)	7.1-3	73	2.5-2	73	5.9-2	73
(15,7)	6.2-3	47	1.5-2	47	5.7-2	47
(15,1)	6.4-4	6.7	5.7-4	6.7	5.1-2	6.7
(15,11)(15,7)	4.3-3	72	1.1-2	70	4.8-2	70
(15,11)(15,5)	3.2-3	72	5.1-3	69	3.9-2	68
(15,11)(15,1)	6.6-4	71	1.5-3	66	8.5-3	61
(15,7)(15,5)	4.6-3	46	6.1-3	45	4.5-2	46
(15,5)(15,1)	6.5-4	32	3.0-4	30	1.3-2	28
4 code	6.9-4	71	1.5-3	66	9.1-3	61

Codes	Average over all 11 runs(17-27)			Average over all random models (28r-30r)			Average over all burst models (31b-41b)		
	S	F	E(%)	S	F	E(%)	S	F	E(%)
(15,11)	2.6-2	1.02	73	2.4-2	1.5	73	3.1-2	1.03	73
(15,7)	2.0-2	1.3	47	6.7-3	5.5	47	2.8-2	1.1	47
(15,1)	7.0-3	3.7	6.7	0	-	6.7	1.2-2	2.7	6.7
(15,11)(15,7)	1.5-2	1.7	70	1.2-2	2.9	57	2.8-2	1.1	68
(15,11)(15,5)	1.2-2	2.1	69	1.0-2	3.6	49	2.5-2	1.3	66
(15,11)(15,1)	2.1-3	12	65	1.7-2	2.1	32	2.1-2	1.5	61
(15,7)(15,5)	1.5-2	1.8	45	2.9-3	13	39	2.5-2	1.3	44
(15,5)(15,1)	2.3-3	11	30	7.8-4	47	17	1.7-2	1.9	28
4 code	2.1-3	12	65	3.0-3	12	29	2.2-2	1.4	62

Table 7.7 also shows average results for the three types of channel. In general VR performance is superior, but improvement factors (F) are low due to the high error rates prevailing, and the poor performance of cyclic codes when operating in the FEC mode over bursty channels. This is particularly shown in the burst model case, and is due to the fact that burst densities for these channels are in the region of 0.3 - 0.5, which is high compared to the real channels.

7.6 Performance of the automatic VR system

This section first presents results for the 8 code VR system which was operated in the ED mode. A total of 19 runs were recorded, and these are designated runs (42-60). Performance results for VR-ARQ, and VR-FEC (obtained by off-line decoding each of the recorded runs) are also shown.

The 19 recorded runs were processed to provide a set of digital channels, which were then used in the simulation of various fixed and VR systems. The results of these simulations are presented for comparison with the experimental system.

7.6.1 Error detection

The results obtained from the automatic 8 code VR system are shown in table 7.8, together with results for three simulated fixed redundancy systems for comparison. Sink block error rates (U), efficiencies (E), and block improvement factors (F) are shown. The performance of the automatic VR system is not quite as good as might have been

Table 7.8

Performance of the Experimental System, with some Simulated Fixed Redundancy Results

Run	Bit Error Rate	Block Error Rate n=15	Experimental System		(15,14) E=93%	(15,11) E=73%	(15,7) E=47%
			U	E(%)	U	U	U
42	7.2-4	2.3-3	8.5-4	40(88)	6.9-4	1.9-3	1.8-3
43	2.5-4	3.2-3	1.2-4	76(86)	2.8-4	0	0
44	5.9-4	4.4-3	3.6-4	90	1.7-3	1.6-4	1.3-5
45	8.7-4	9.2-3	1.3-3	70(85)	1.8-3	5.3-4	0
46	1.3-3	1.1-2	5.9-5	76	3.3-3	1.8-4	0
47	1.8-3	1.4-2	2.2-3	91	7.8-3	3.8-3	3.7-4
48	3.9-3	2.4-2	0	36(83)	1.1-2	8.0-4	8.8-5
49	5.1-3	2.8-2	1.6-3	37(64)	7.1-3	5.9-3	3.1-3
50	4.6-3	2.8-2	3.4-3	34(72)	1.6-2	0	0
51	4.2-3	4.3-2	0	33(66)	9.9-3	0	0
52	3.1-3	4.6-2	0	34(91)	1.0-3	0	0
53	1.1-2	5.3-2	1.2-3	37(67)	2.1-2	2.5-3	9.2-4
54	8.4-3	5.7-2	3.0-3	26(71)	1.9-2	3.6-3	1.4-3
55	1.2-2	6.8-2	5.0-3	46(53)	2.3-2	6.9-3	4.3-3
56	7.8-3	7.0-2	1.7-3	49(72)	1.2-2	6.9-3	0
57	1.7-2	1.2-1	2.4-2	85	3.9-2	1.9-3	4.8-4
58	3.1-2	1.5-1	1.5-2	40(67)	6.0-2	1.2-2	7.6-3
59	3.0-2	2.6-1	3.3-2	84	8.1-2	3.9-3	0
60	6.6-2	4.9-1	1.3-1	81	1.9-1	1.9-2	7.1-3
Average of the above:							
	1.1-2	7.8-2	1.2-2	56(78)	2.4-2	3.7-3	1.4-3
F=			6.7		3.3	21	55

Average results for 2 code VR-ED:

Code	U	F	E(%)
(15,14)(15,11)	1.2-2	6.4	92
(15,14)(15,7)	1.1-2	7.5	90
(15,14)(15,1)	9.3-3	8.4	87

expected. This can be partially attributed to the following:

(i) Sink block error rate is increased because of encoder-decoder mismatch. This was quite frequent and undoubtedly had a large effect on sink error rate. All 19 runs have some mismatch, and many other runs had to be discarded because of 'oscillation' between two mismatched states. The mismatch situation quite often righted itself quickly; but the use of codes with disjoint code books would have improved matters considerably.

(ii) Efficiencies were decreased because of a fault in the code change unit. This meant that if *no* error occurred it took 1000 blocks to change 'down' to a more efficient code. The fault was not detected until after the tests, but its effect was removed by offline processing. The effect of this fault varies from run to run, depending on the error statistics; and adjusted efficiencies are shown (bracketed) in table 7.8 for those runs most affected. The adjustment gives a 20% increase in average efficiency over all 19 runs, and shows that the experimental VR system performed reasonably well.

(iii) The code change criterion is not particularly suitable for the burst conditions prevailing.

The simulated ED performance of the (15,14), (15,11), and (15,7) fixed redundancy codes are also shown in table 7.8, and are generally better than the VR-ED system.

A comparison simulation using the 3-out-of 7 ARQ code gave a sink block improvement factor of 10, at an efficiency of 43%, when averaged over all 19 runs.

Finally, table 7.8 shows the performance of 2-code VR-ED, using the same simulation with the same code change criterion, as that used in section 7.5.2. This system out-performs the fixed redundancy simulations; but when comparing the results to those of the auto-VR experimental system, it must be remembered that noiseless feedback is assumed.

7.6.2 Error correction by ED-ARQ

Table 7.9 shows the performance of the experimental system in the VR-ARQ mode; the efficiencies quoted have been adjusted to cancel out the effect of the 1000 block delay. The system performs reasonably well when compared with the fixed redundancy simulations also shown in table 7.9.

Table 7.10 shows ARQ performance averaged over all 19 runs for several fixed and VR schemes, as well as for the experimental system. The advantage of the VR schemes (which are the same as those of section 7.5.3) is clear; but the performance of the experimental system is poor in comparison. However, noiseless feedback in the simulations must be considered when comparing practical and simulated results.

Table 7.9

ED-ARQ Performance

Run	Experimental VR system		(15,14)		(15,11)		(15,7)	
	S	E(%)	S	E(%)	S	E(%)	S	E(%)
42	2.7-4	87	3.0-4	93	6.7-4	73	2.6-4	47
43	9.3-6	86	1.7-5	92	0	73	0	46
44	4.4-5	89	1.6-4	93	8.6-6	73	0	46
45	1.3-4	84	1.7-4	92	3.8-5	72	0	46
46	7.1-6	74	2.3-4	92	2.9-5	72	0	46
47	2.9-4	87	1.3-4	92	4.6-4	72	8.6-5	46
48	0	79	1.6-3	90	1.4-4	70	5.3-5	44
49	8.1-4	61	8.9-4	89	1.1-3	70	6.5-4	45
50	5.8-4	69	9.3-4	90	0	68	0	43
51	0	60	5.7-4	86	0	67	0	43
52	0	83	1.2-4	88	0	69	0	44
53	2.6-4	59	2.2-3	87	5.7-4	67	1.4-4	43
54	4.9-4	64	1.6-3	84	717-4	66	2.8-4	42
55	1.0-3	46	2.9-3	84	1.4-3	66	6.0-4	42
56	2.2-4	63	7.1-4	83	4.8-4	65	0	41
57	4.2-3	68	3.4-3	77	1.1-4	59	0	37
58	4.3-3	48	9.8-3	78	3.3-3	59	2.5-3	37
59	7.0-3	46	6.8-3	61	1.7-4	45	0	29
60	6.0-2	24	2.4-2	44	3.4-3	26	2.1-3	16

Table 7.10

ED-ARQ Performance Averaged over Runs (42-60)

Codes	S	F	E(%)
(15,14)	3.0-3	3.7	84
(15,14)(15,1)	6.4-4	17	79
(15,14)(15,11)(15,1)	7.1-4	16	80
(15,14)(15,7)(15,1)	6.9-4	16	79
8 code	6.4-4	17	79
(15,11)	6.6-4	17	70
(15,11)(15,7)	4.2-4	26	64
(15,11)(15,1)	1.9-4	58	62
(15,7)	3.5-4	31	41
(15,5)	1.1-4	100	29
Experimental system	4.2-3	2.6	67

7.6.3 *Forward error correction*

Table 7.11 gives the results obtained from off-line decoding the experimental system in the 3-code FEC mode. Again, the efficiencies have been adjusted.

The results here are much better than previously shown results, when compared to the simulated fixed redundancy schemes. Possible reasons for this are that: 3-code VR is better than 8-code VR under the type of conditions experienced; the higher redundancy (15,11) code is a better choice of 'least powerful' code than the (15,14) code; the code change criterion used operates more accurately with 3-code VR.

Table 7.12 shows averaged results for several simulated schemes (section 7.5.4) and the experimental system. In this case the advantage of VR is again shown, and the experimental system also compares favourably to the simulated results.

7.7 *Comments on the results*

The propagation results are as expected. They correspond well with the available predicitions, and confirm the work of other investigators (Betts 1967). The results emphasise that in order to avoid multipath effects, operation as close to the MUF is desirable, particularly at high baud rates.

The channel statistics measurements confirm the severity of propagation and interference effects; the error rate is generally high, and varies rapidly. Fast selective fading,

Table 7.11

FEC Performance

Run	Experimental System		(15,11)	(15,7) E=47%	(15,1) E=7%
	S	E(%)	E=73% S	S	S
42	3.2-4	67	7.3-4	3.3-4	0
43	6.5-5	71	7.5-5	3.4-5	0
44	1.1-4	73	5.2-4	2.7-4	4.0-5
45	3.4-4	70	3.6-4	1.8-4	0
46	1.7-4	59	1.1-3	5.3-4	0
47	4.1-4	72	8.0-4	3.2-4	2.6-4
48	2.3-4	66	4.3-3	3.6-3	8.8-5
49	1.3-3	48	4.1-3	2.6-3	1.7-3
50	1.3-3	53	5.1-3	3.1-3	0
51	6.0-4	52	2.9-3	1.4-3	0
52	4.6-5	72	9.4-5	0	0
53	8.1-4	58	1.1-2	9.0-3	1.9-3
54	1.6-3	54	7.6-3	5.1-3	5.4-4
55	4.5-3	42	1.0-2	6.3-3	2.3-3
56	1.7-3	55	4.7-3	1.8-3	0
57	4.6-3	71	1.2-2	8.6-3	7.7-3
58	6.4-3	51	2.9-2	2.3-2	8.5-3
59	1.8-2	68	2.5-2	1.2-2	8.4-4
60	2.2-2	67	4.5-2	2.6-2	5.7-3

Table 7.12

FEC Performance Averaged over all 19 Runs (42-60)

Codes	S	F	E(%)
(15,11)	8.7-3	1.3	73
(15,11)(15,1)	2.2-3	5.0	67
(15,11)(15,7)(15,1)	2.0-3	5.5	67
(15,7)	5.5-3	2.0	47
(15,7)(15,5)(15,1)	1.2-3	9.2	43
(15,5)	4.0-3	2.8	33
(15,5)(15,1)	1.0-3	11	31
(15,1)	1.6-3	6.9	7
Experimental system	3.4-3	3.2	62

the high baud rate used, multipath, and synchronisation drift all contributed to make the forward link generally very bursty.

The simulation results show that error rate is improved by fixed redundancy coding; but that efficiency is poor for the more powerful codes (particularly for FEC). Simulations of variable redundancy schemes show a distinct improvement in efficiency, with no loss of error control. Interleaving offers some degree of improvement, but is only worthwhile for small degrees of interleave.

The improvement due to the use of VR is generally confirmed by the results for the experimental system. The results would have been even better if:

(i) codes with some burst control ability had been used

(ii) codes less sensitive to synchronisation slip (and drift) had been used.

(iii) the serious effect of feedback errors had been reduced by using: codes with disjoint code books; and feedback error control.

The simulated results also indicate that simple 2 or 3-code VR systems, with simple code change criteria, offer advantages of realisation and performance, as compared with the more sophisticated system, under the heavy burst conditions experienced. Laboratory tests of the experimental system using noiseless feedback, and slowly varying white noise on the forward path, confirmed that *these* are the conditions under which maximum performance is obtained.

CHAPTER 8

Summary, Conclusions and Suggestions for Further Research

8.1 Summary

In this thesis, a theoretical and practical investigation of the variable redundancy coding technique has been presented. Chapter 1 introduced the technique, and outlined the reasons for thinking that VR systems could prove superior to fixed redundancy systems, in terms of performance and implementation.

This was followed in chapters 2 and 3 by a review of fixed redundancy error control systems, and a review of coding theory and implementation. This was included in order to make the thesis as self-contained as possible, and may be omitted by the informed reader. The review was also intended, however, to bring out the problems involved in finding and implementing (particularly decoding) long powerful codes. Such codes are needed if efficient fixed redundancy transmission over practical channels (which tend to be non-random and time-varying) is to be possible.

An alternative to having impossibly complex or highly inefficient fixed redundancy systems is to use an adaptive system. Chapter 4 looked at adaptive systems generally, of which VR coding was only one part, and was intended to classify different possible adaptive systems, and discuss the problems and advantages that might occur by their use.

Chapter 5 was solely concerned with VR coding and a broad view of the theoretical and practical aspects of the technique were presented there. The theoretical advantages to be gained by the use of VR coding on time-varying channels was confirmed, and the practical advantages of using constant n code sets, and codes with disjoint code books were brought out.

The experimental VR system was then described in chapter 6, together with the reasons for the particular sub-system design choices that were made. These choices reflected the investigations of chapter 5. Finally, in chapter 7, results obtained from the experimental system, and simulated systems, were presented. These results, in general, confirmed the performance advantages of VR coding that were postulated in chapter 1.

8.2 Conclusions

It is possible to draw the following conclusions from the research:

(i) There is a performance and implementation advantage to be gained by the use of a VR system which uses short block codes. This was shown theoretically for the time-varying BSC in chapter 5, and by experimentation and simulation for the burst channels of chapter 7.

(ii) The performance and implementation advantage of VR is greatest for FEC error control, because this is when an FR system is most inefficient and/or complex.

(iii) The performance advantage to be gained depends to a great extent on the code change criterion, and its accurate operation, which in turn must be based on the error characteristics of the channel. The relatively poor performance of the experimental system may be partially attributed to this: the code change criterion assumed a relatively slowly varying, mainly random channel, but was presented with a fast changing bursty one (the HF channel).

(iv) Simple code change criteria such as those used in the simulations of chapter 7 are very effective for burst channels; but a complex channel such as the HF channel would probably benefit from a code change criterion based on both random and burst high-order statistics, possibly with soft decision information.

(v) The number of codes needed in the code set is not excessive, and indeed little advantage is gained by using more than 4 codes, regardless of whether the channel is bursty or random.

(vi) Because of the low number of codes needed, the choice of a suitable set of codes need not be unduly restrictive. If synchronisation problems could be overcome, so that the constant n restriction could be removed, considerable optimisation of a VR system could probably be achieved, particularly by providing both burst and random error control.

(vii) Data transmission at 1000 bauds (serial) on the HF channel is not reliable without antimultipath

measures, or complex sounding systems which would aim to keep the carrier frequency just below the MUF. Rather, parallel low-baud sub-channel transmission seems the only reliable method for narrow-band high speed HF data transmission at present.

(viii) The problem of buffering the variable rate data input does not seem too severe for VR over the HF channel, given modern disc and tape storage facilities whose performance compares favourably with possible HF baud rates. For faster systems a many-user multiplex system, with users assigned different degrees of error control, depending on the volume and priority of the traffic, as well as channel conditions, might go some way to solving the problem.

(ix) The effect of feedback errors on a VR system can be greatly reduced, thereby improving reliability and performance, by the use of codes with disjoint code books.

8.3 *Suggestions for further research*

Several questions and research topics have arisen as a result of the research. These may be divided into three main areas: development of the binary VR system, other VR and FR systems, and codes with disjoint code books.

8.3.1 *Development of the system*

The VR technique could be applied to a duplex HF data transmission system with low baud rate parallel channels.

Questions and problems that arise are:

(i) the provision of burst and random error correcting codes

(ii) development of the code change criterion to include classification of the channel as 'bursty' or 'random-like' so that the requisite burst or random power is used.

(iii) The utilisation of soft-decision methods in oder to help in (ii) above, and to combat the effects of selective fading on sub-channels in close (frequency) proximity.

(iv) the use of FEC with ARQ for uncorrectable errors.

8.3.2 Other VR and FR systems

When thinking of VR and FR systems generally, the following points arise.

(i) convolutional codes for VR systems

(ii) automatically adaptive decoders that can decode in a 'random-like' or a 'burst-like' way depending on the assessed state of the channel.

(iii) codes that are comma-free, in order to ease the sync problem for variable n code sets.

(iv) a generalised code change criterion, that itself adapts to prevailing channel conditions.

(v) non-binary VR codes, for multi sub-channel systems.

(vi) non-linear VR codes for easier construction of disjoint code sets.

(vii) can the feedback link be eliminated for VR coding over the HF channel, by the use of sophisticated transmitter sounding (eg backscatter sounding).

(viii) VR systems that adapt to the *type* of errors, as well as to the *number* of errors.

(ix) the design of a single decoder that can decode all codes in the VR set, and basically exchanges decoding delay for code power (eg permutation decoding).

(x) VR with a variable number of codes, dependent on channel conditions.

(xi) can cybernetic learning techniques be applied to FR or VR coding in order to provide non-bounded-distance decoding schemes in which the decoder adaptively learns the 'best' decoding scheme in order to increase the probability of correct decoding for current conditions.

(xii) the application of pattern *prediction* techniques to the code change and channel assessment problems.

(xiii) adaptive soft-decision decoding, in which demodulation information from *past* decodings, as well as the current decoding, is used to adapt the decoding algorithm (particularly for multi sub-channel systems).

8.3.3 *Codes with disjoint code books and mutual Hamming distance*

Although this topic arose as a means of combatting feedback errors on the experimental system, disjoint codes are of interest in their own right.

Investigation of these codes has begun (Goodman, 1974) and a reprint of this paper is given in Appendix A.

Further questions that arise are:-

(i) can the common code generator matrix be easily partitioned to give two useful disjoint codes, in the non-cyclic case?

(ii) can bounds on mutual distance be found?

(iii) can disjoint codes be used to produce new good codes?

(iv) would non-linear codes facilitate finding disjoint code sets?

(v) the application of disjoint code sets to 'no-feedback' VR.

APPENDIX A

BINARY CODES WITH DISJOINT CODEBOOKS AND MUTUAL HAMMING DISTANCE

Indexing term: Error-correction codes

Equal-length linear binary block error-control codes with disjoint codebooks and mutual Hamming distance are considered. A method of constructing pairs of these disjoint codes from known cyclic codes, and determining their mutual distance, is described. Some sets of length-15 cyclic codes are tabulated.

The author is concerned with the construction of sets of equal-length linear binary block error-control codes with disjoint codebooks for use in several coding schemes.[1] In addition, for any pair of disjoint codes, it is required to find the minimum distance that separates the words of one code from the words of the other. This distance is called the minimum mutual Hamming distance d_m of the disjoint code pair. This letter establishes the conditions which a pair of codes must fulfil if they are to have disjoint codebooks, and gives a general method for calculating d_m. In particular, a practical method of testing pairs of known cyclic codes for disjoint codebooks, and determining their mutual distance, is described.

An (n, k, d) binary linear block code has length n, a codebook of 2^k codewords (including the all-zero word) and minimum distance d equal to the weight of the minimum-weight nonzero codeword. The code is completely specified by a $k \times n$ generator matrix G whose rows are linearly independent basis vectors that span the codespace. Each of the 2^k distinct linear combinations of rows of G generates a distinct codeword.

Consider two codes C_1 and C_2 with parameters (n, k_1, d_1) and (n, k_2, d_2) and generator matrices G_1 and G_2. For C_1 and C_2 to have disjoint codebooks, apart from the all-zero word, no codeword in C_1 must equal a codeword in C_2. That is, no linear combination of rows of G_1 equals a linear combination of rows of G_2, and therefore the rows of G_1 and G_2 must be mutually linearly independent. A matrix G_c that has as rows all the basis vectors of C_1, all the basis vectors of C_2 and no others, must therefore have k_1+k_2 linearly independent rows if C_1 and C_2 are to be disjoint, and can therefore be considered as the generator matrix of an (n, k_1+k_2, d_c) code, which will be called the common code C_c. A necessary condition for the rows of G_c to be linearly independent, and hence for C_1 and C_2 to be disjoint, is

$$n \geqslant k_1+k_2 \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (1)$$

Given that eqn. 1 is satisfied and that the rows of G_c are linearly independent, it can be shown that the minimum distance d_c of the common code equals the minimum mutual distance d_m of the code pair. The sum of any C_1 word u_1 with any C_2 word u_2 equals a third word u_{12}, and the weight of this 'mutual' word equals the mutual distance between u_1 and u_2. The minimum mutual distance between the codes then equals either the minimum-weight nonzero C_1 word, the minimum-weight nonzero C_2 word, or the minimum-weight mutual word, whichever is smaller. The common code contains all these words, plus the all-zero word, and no other. The minimum distance of the common code therefore equals the minimum mutual distance, which can never exceed d_1 or d_2, whichever is smaller. A necessary and sufficient condition for disjoint codebooks is therefore that the common code should have a distance $\geqslant 1$.

To test two codes for disjoint codebooks, the individual codes are tested for mutually linearly independent basis vectors, and d_m is determined by finding the distance of the common code. These procedures are, however, impracticably lengthy, even with computer aid, if k_1 and k_2 are large. The converse procedure, that of partitioning a known common code-generator matrix to yield two useful disjoint codes is attractive, and is at present under investigation. The problems involved in testing for disjoint codes and determining d_m are simplified if C_1 and C_2 are nontrivial $(k_1, k_2 \neq 0, \neq n)$ cyclic codes. In this case, we consider the generator polynomials $g_1(x)$ and $g_2(x)$ of C_1 and C_2, with degrees $n-k_1$ and $n-k_2$, respectively. A generator polynomial can be characterised by a list of the exponents of its roots, or written as a polynomial whose irreducible factors are minimum functions of its constituent roots.[2] The 2^k codewords in a cyclic code consist of all multiples $u(x) = m(x)\,g(x)$, where $m(x)$ is a message polynomial of degree $\leqslant k-1$. For C_1 and C_2 to have disjoint codebooks, a word u_1 in C_1 must not be exactly divisible by $g_2(x)$. That is,

$$\frac{m_1(x)\,g_1(x)}{g_2(x)} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

must have a nonzero remainder. Eqn. 2 therefore requires that the degree of $m_1(x)$ is less than the degree of $g_2(x)$ giving

$$n-k_2 > k_1-1 \quad \text{and} \quad n > k_1+k_2-1 \quad . \quad . \quad (3)$$

as a necessary condition for disjoint codebooks.

Because the factors of $g_1(x)$ and $g_2(x)$ are factors of x^n+1, $g_1(x)$ and $g_2(x)$ may have common factors that will cancel in eqn. 2. A further necessary condition is then that, after cancellation of common factors, the denominator of eqn. 2 must still have a higher degree than $m_1(x)$. The largest denominator after cancellation occurs when $g_2(x)$ contains all the factors of x^n+1 that do not appear in $g_1(x)$. The degree of the resultant denominator is then at least k_1, which is the minimum degree required for the denominator to have degree greater than k_1-1, the degree of $m_1(x)$. The conditions required for two cyclic codes to have disjoint code books are therefore (*a*) $k_1, k_2 \neq 0, \neq n$, (*b*) $n > k_1+k_2-1$, and (*c*) $g_1(x)$ and $g_2(x)$ between them contain all the irreducible factors of x^n+1.

The generator of the common code, $g_c(x)$, must divide $g_1(x)$, $g_2(x)$ and all mutual words only, and is therefore the greatest common divisor of $g_1(x)$ and $g_2(x)$. The common

code is also cyclic, so that its distance, and hence the mutual distance of the disjoint code pair, is easily determined if the code is tabulated.[2] If the distance of the common code is not known exactly, a lower bound on mutual distance can be obtained by the Bose–Chaudhuri–Hocquenghem (BCH) bound for cyclic codes.[2] The converse procedure, that of constructing a disjoint code pair, is also considerably simplified: a common code is first selected, defining d_m, and factors are added to $g_1(x)$ and $g_2(x)$, subject to conditions (*a*) to (*c*), to produce the required codes. Similarly, a code disjoint to an existing code can be easily constructed. Sets of more than two disjoint codes can also be formed by repeated construction of disjoint pairs. These procedures for the testing and construction of disjoint cyclic code pairs are practical, and also suitable for computer implementation.

Example: It is required to test the (15, 6, 6) code and the (15, 4, 8) code for disjoint codebooks. The exponents of the roots of the generator polynomial are tabulated[2] as (0, 1, 7) and (0, 1, 3, 5), respectively. Condition (*a*) is satisfied; condition (*b*) is satisfied: $15 > 6+4-1$; and condition (*c*) is satisfied because all the roots of $x^{15}+1$ are contained in $g_1(x)$ and $g_2(x)$. The codebooks are therefore disjoint. The roots (0, 1) are common and cancel, leaving (3, 5) as the roots of the denominator. The minimum functions of 3 and 5 have degrees 4 and 2, respectively, giving a denominator of degree 6, which is greater than 5, the degree of $m_1(x)$. The roots of $g_c(x)$ are (0, 1), and the common code is therefore the (15, 10, 4) code, giving $d_m = 4$. Table 1 gives some of the length-15 disjoint code pairs with mutual Hamming distances.

R. M. F. GOODMAN *5th August 1974*

School of Electronic Engineering
Kingston Polytechnic
Penrhyn Road, Kingston upon Thames, Surrey, England

References

1 GOODMAN, R. M. F.: 'Variable redundancy coding'. Ph.D thesis' University of Kent at Canterbury (in preparation)
2 PETERSON, W. W., and WELDON, E. J., JUN.: 'Error-correcting codes' (MIT Press, 1972)

Table 1

k_1	k_2	d_1	d_2	Roots $g_1(x)/g_2(x)$	Roots $g_c(x)$	k_c	d_m
9	6	4	6	3, 5/0, 1, 7	–	15	1
9	4	4	6	3, 5/0, 1, 5, 7	5	13	2
9	2	3	10	1, 5/0, 1, 3, 7	1	11	3
8	6	4	6	0, 3, 5/0, 1, 7	0	14	2
8	2	4	10	0, 3, 5/0, 1, 3, 7	0, 3	10	2
8	2	4	10	0, 1, 5/0, 1, 3, 7	0, 1	10	4
7	4	3	8	1, 7/0, 1, 3, 5	1	11	3
7	4	5	6	1, 3/0, 1, 5, 7	1	11	3
6	5	6	7	0, 1, 3/1, 5, 7	1	11	3
6	4	6	8	0, 1, 7/0, 1, 3, 5	0, 1	10	4
5	4	3	8	1, 5, 7/0, 1, 3, 5	1, 5	9	3
5	2	7	10	1, 3, 5/0, 1, 3, 7	1, 3	7	5
4	4	6	8	0, 1, 5, 7/0, 1, 3, 5	0, 1, 5	8	4
4	3	8	10	0, 1, 3, 5/1, 3, 7	1, 3	7	5
4	2	8	10	0, 1, 3, 5/0, 1, 3, 7	0, 1, 3	6	6

APPENDIX B

References

ABRAMSON, N. 1959. "A Class of Systematic Codes for Non-Independent Errors." IRE Trans., IT-5.

ANDERSON, R. E. 1961. "Sideband Correlation of lunar and Echo Satellite Reflection Signals in the 900 mc Range." Proc. IRE, vol 49.

BALKOVIC, M. D., H. W. Klancer, S. W. Klare, and W. G. McGruther. 1971. "High Speed Voiceband data transmission on the Switched Telecommunications network." BSTJ, Vol.50, No.4.

BELLO, P. A. 1969. "Selection of Multichannel digital data systems for Troposcatter Channels." IEEE Trans., Com-17, 2.

BELLO, P. A., and W. M. Cowan. 1962. "Theoretical Study of on/off Transmission over Gaussian Multiplicative Circuits." Proc. IRE 8th Nat. Comms. Symp., Utica, N.Y., U.S.A.

BELLO, P. A., and B. D. Nelin. 1963. "The Effect of Frequency Selective Fading on the Binary Error Probabilities of Incoherent and Differentially Coherent Matched filter Receivers." IRE Trans. Comm. Syst., CS-11.

BERLEKAMP, E. R. 1967. "Nonbinary BCH Decoding." IEEE Int. Symp. on Info. Th., San Remo, Italy.

BERLEKAMP, E. R. 1968. Algebraic Coding Theory. McGraw-Hill, New York.

BETTS, J. A. 1967. High Frequency Communications. English Universities Press.

BOSE, R. C., and D. K. Ray-Chaudhuri. 1960. "On a Class of Error Correcting Binary Group Codes." Inf. and Control, 3.

BRAYER, K. 1971. "Error Correction Code Performance on HF, Troposcatter, and Satellite Channels." IEEE Trans., COM-19, 5.

BRAYER, K., and O Cardinale. 1967. "Evaluation of Error Correction Block Encoding for High-Speed HF Data." IEEE Trans., Com-15, 3.

BURTON, H. O. 1969. "A Class of Asymptotically Optimal Burst Correcting Block Codes." Presented at the ICCC, Boulder, Colorado, U.S.A.

BURTON, H. O., and E. J. Weldon, Jr. 1965. "Cyclic Product Codes." IEEE Trans., IT-11.

CAVERS, J. K. 1972. "Variable-Rate Transmission for Rayleigh Fading Channels." IEEE Trans., Com-20, 1.

CHIEN, R. T. 1964. "Cyclic Decoding Procedure for BCH Codes." IEEE Trans., IT-10.

CHASE, D. 1972. "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information." IEEE Trans., IT-18, 1.

DI TORO, M. J. 1968. "Communication in Time - Frequency Spread Media Using Adaptive Equalisation." Proc. IEEE, Vol 56, No. 10.

DOELZ, M. L. 1954. "Predicted Wave Radio Teleprinter." Electronics, 27.

ELIAS, P. 1954. "Error-free Coding." IRE Trans., PGIT-4.

ELIAS, P. 1955. "Coding for Noisy Channels." IRE Convention Record, Part 4.

ELIAS, P. 1956. "Coding for two noisy Channels." Information Theory, ed., Colin Cherry. London, Butterworth.

ELLIOTT, E. O. 1963. "Estimates of Error Rates for Codes on Burst—Noise Channels." BSTJ, 42.

ELLIOTT, E. O. 1965. "A Model of the Switched Telephone Network for Data Communications." BSTJ, 44.

EPSTEIN, M. A. 1958. Algebraic Decoding for a Binary Erasure Channel. Research Laboratory of Electronics Report 340, M.I.T., Cambridge, Mass., U.S.A.

FANO, R. M. 1961. Transmission of Information. New York. John Wiley & Sons.

FANO, R. M. 1963. "A Heuristic Discussion of Probabilistic Decoding." IEEE Trans., IT-9, 2.

FARRELL, P. G. 1969. "Coding for Noisy Data Links." Ph.D. dissertation, University of Cambridge.

FIRE, P. 1959. "A Class of Multiple-Error-Correcting Binary Codes for Non-Indpendent Errors." Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Laboratory, Mountain View, California, U.S.A.

FORNEY, G. D., Jr. 1965. "On Decoding BCH codes." IEEE Trans., IT-11.

FORNEY, G. D., Jr. 1966. Concatonated Codes. M.I.T. Press, Cambridge, Mass., U.S.A.

FRANCO, A. G., and M. E. Wall. 1965. "Coding for Error Control." Electronics. Dec.

FRITCHMAN, B. D. 1967. "A Binary Channel Characterization Using Partitioned Markov Chains." IEEE Trans., IT-13.

GALLAGER, R. G. 1963. Low-Density Parity-Check Codes. M.I.T. Press Research Monograph 21, Cambridge, Mass., U.S.A.

GALLAGER, R. G. 1968. Information Theory and Reliable Communication. New York. John Wiley & Sons.

GILBERT, E. N. 1952. "A Comparison of Signalling Alphabets." BSTJ, 31.

GILBERT, E. N. 1960. "Capacity of a Burst-Noise Channel." BSTJ, 39.

GOETHALS, J. M., and P. Delsarte. 1968. "On a Class of Majority-Logic Decodable Cyclic Codes." IEEE Trans., IT-14, 2.

GOLAY, M. J. E. 1949. "Notes on Digital Coding." Proc. IRE, 37, Corresp.

GOODMAN, R. M. F. 1974. "Binary Codes with Disjoint Codebooks and Mutual Hamming Distance." Electronics Letters, Vol.10, No.18.

GORENSTEIN, D., W. W. Peterson, and N. Zierler. 1960. "Two Error Correcting BCH Codes are Quasi Perfect." Inf. and Control, 3.

GREENBURG, H., et al. 1962. "Optimum H. F. Prediction." Trans. IRE, AP-10, 3.

GREISMER, J. H. 1960. "A Bound for Error Correcting Codes." IBMJ. Res. Dev., 4.

HAMMING, R. W. 1950. "Error Detecting and Error Correcting Codes." BSTJ, 29.

HAYES, J. F. 1968. "Adaptive Feedback Communications." IEEE Trans., Com-16, 1.

HELGERT, H. J., and R. D. Stinaff. 1973. "Minimum Distance Bounds for Binary Linear Codes." IEEE Trans., IT-19, 3.

HOCQUENGHEM, A. 1959. "Codes correcteurs d'erreurs." Chiffres, 2.

HSU, H. T., T. Kasami, and R. T. Chien. 1968. "Error Correcting Codes for a Compound Channel." IEEE Trans., IT-14, 1.

HUFFMAN, D. A. 1952. "A Method for the Construction of Minimum Redundancy Codes." Proc. IRE, 40.

INGALLS, R. P., L. E. Bird, and J. W. B. Day. 1961. "Bandpass Measurements of a Lunar Reflection Circuit." Proc. IRE, 49.

INTELSAT IV International Conference. 1969. Held at the IEE, Savoy Place, London.

JOHNSON, S. M. 1962. "A New Upper Bound for Error-Correcting Codes." IEEE Trans., LT-8.

JOHNSON, S. M. 1963. "Improved Asymptotic Bounds for Error-Correcting Codes." IEEE Trans., IT-9.

JOHNSON, S. M. 1971. "On Upper Bounds for Unrestricted Binary Error-Correcting Codes." IEEE Trans., IT-17.

KASAMI, T. 1964. "A Decoding Procedure for Multiple Error-Correcting Cyclic Codes." IEEE Trans., IT-10.

KASAMI, T., S. Lin, and W. W. Peterson. 1968. "New Generalisations of the Reed-Muller Codes - Part I: Primitive Codes." IEEE Trans., IT-14, 2., and "Polynomial Codes." IT-14, 6.

KASAMI, T., and S. Matoba. 1964. "Some Efficient Shortened Cyclic Codes for Burst-Error Correction." IEEE Trans., IT-1

KIRCH, A. L., P. R. Gray, D. W. Hanna, Jr. 1969. "Field-Test Results of the AN/GSC-10 (KATHRYN) Digital Data Terminal." IEEE Trans., Com-17, 2.

LLOYD., S. P. 1957. "Binary Block Coding." BSTJ, 36.

LIEBERMAN, G. 1963. "Adaptive Digital Communication for a Slowly Varying Channel." IEEE Trans., No 65, March.

LIN, S. 1970. An Introduction to Error-Correcting Codes. New Jersey. Prentice-Hall.

LUCKY, R. W., J. Salz, and E. J. Weldon, Jr. 1968. Principles of Data Communication. McGraw-Hill.

MacWILLIAMS, F. J. 1963. "A Theorem on the Distribution of Weights in a Systematic Code." BSTJ, 42.

MacWILLIAMS, F. T. 1964. "Permutation Decoding of Systematic Codes." BSTJ, 43.

MASSEY, J. L. 1963. Threshold Decoding. M.I.T. Press Research Monograph 20, Cambridge, Mass., U.S.A.

MASSEY, J. L. 1965. "Step-by-Step Decoding of the BCH Codes." IEEE Trans., IT-11.

MASSEY, J. L. 1969. "Shift Register Synthesis and BCH Decoding." IEEE Trans., IT-15, 1.

MITANI, N. 1951. "On the Transmission of Numbers in a Sequential Computer." Nat. Conv. of the IECE of Japan.

MEGGITT, J. E. 1960. "Error Correcting Codes for Correcting Bursts of Errors." IBMJ, Research Develop., 4.

MEGGITT, J. E. 1961. "Error Correcting Codes and Their Implementation." IRE Trans., IT-7.

MOSIER, R. R., and R. G. Clabaugh. 1958. "Kineplex, A Bandwidth-Efficient Binary Transmission System." Trans. AIEE (Comm. and Elect.), 34.

MULLER, D. E. 1954. "Application of Boolean Algebra to Switching Circuit Design and Error Detection." IRE Trans., EC-3.

OMURA, J. K. 1969. "A Probabilistic Decoding Algorithm for Binary Group Codes." presented at IEEE Internat. Symp. on Info. Th.

O'NEILL, J. R., and B. R. Saltzberg. 1966. "An Automatic Equalizer for Coherent Quadrature Carrier Data Transmission Systems." IEEE Internat. Comm. Conf., Philadelphia, U.S.A.

PETERSON, W. W. 1960. "Encoding and Error-Correction Procedures for the BCH codes." IRE Trans., IT-6.

PETERSON, W. W. 1967. "On the Weight Structure and Symmetry of BCH Codes." Journal of the IECE, Japan, 50.

PETERSON, W. W., and E. J. Weldon, Jr. 1972. Error Correcting Codes. M.I.T. Press.

PLOTKIN, M. 1960. "Binary Codes with Specified Minimum Distance. IRE Trans., IT-6.

PRANGE, E. 1957. Cyclic Error-Correcting Codes in Two Symbols. AFCRC-TN-57-103, Air Force Cambridge Research Centre, Cambridge, Mass., U.S.A.

PRANGE, E. 1958. Some Cyclic Error-Correcting Codes with Simple Decoding Algorithms. AFCRC-TN-58-156, Air Force Cambridge Research Centre, Cambridge, Mass., U.S.A.

PRANGE, E. 1959. The Use of Coset Equivalence in the Analysis and Decoding of Group Codes. AFCRC-TR-59-164, Air Force Cambridge Research Centre, Cambridge, Mass., U.S.A.

PRICE, R., and P. E. Green, Jr. 1958. "A Communication Technique for Multipath Channels." Proc. IRE, 46.

REED, I. S. 1954. "A Class of Multiple Error Correcting Codes and the Decoding Scheme." IRE Trans., PGIT-4.

REED, I. S., and G. Solomon. 1960. "Polynomial Codes over Certain Finite Fields." J. Soc. Indust. Appl. Math., 8.

REDDY, S. M., and J. P. Robinson. 1970. "Random Error and Burst Error Correction by Iterated Codes." IEEE Trans., IT-18, 1.

REIGER, S. H. 1960. "Codes for the Correction of Clustered Errors." IRE Trans., IT-6.

ROBIN, H. K., et al. 1963. "Multitone Signalling System Employing Quenched Resonators for use on Noisy Radio-Teleprinter Circuits." Proc. IEE, Vol.110, No.9.

RUDOLPH, L. D. 1967. "A Class of Majority-Logic Decodable Codes." IEEE Trans., IT-13, 2.

RUDOLPH, L. D., and M. E. Mitchell. 1964. "Implementation of Decoders for Cyclic Codes." IEEE Trans., IT-10, 3.

SACKS, G. E. 1958. "Multiple Error Correction by Means of Parity Checks." IRE Trans., IT-4.

SHANNON, C. E. 1948. "A Mathematical Theory of Communication." BSTJ, 27.

SHANNON, C. E. 1957. "Certain Results in Coding Theory for Noisy Channels." Inf. and Control, 1.

SHANNON, C. E., R. G. Gallager, and E. R. Berlekamp. 1967. "Lower Bounds to Error Probability for Coding on Discrete Memoryless Channels." Inf. and Control, 10.

SLEPIAN, D. 1956. "A Class of Binary Signalling Alphabets." BSTJ, 35.

SRINIVASAN, R., and R. L. Brewster. 1974. "Feedback Communicatic in Fading Channels." IEEE Trans. Comm. January.

STEIN, S., with M. Schwartz, and W. R. Bennett. 1966. Communication Systems and Techniques. McGraw-Hill.

TOWNSEND, R. L., and E. J. Weldon, Jr. 1967. "Self-Orthogonal Quasi-Cyclic Codes." IEEE Trans., IT-13.

TSAI, S. 1969. "Markov Characterization of the HF Channel." IEEE Trans., Com-17, 1.

VARSHARMOV, R. R. 1957. "Estimate of the Number of Signals in Error Correcting Codes." Doklady A.N.S.S.S.R., 117, No 5.

VITERBI, A. J. 1967. "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." IEEE Trans., IT-13.

WAINSTEIN, L. A., and V. D. Zubakov. 1962. Extraction of Signals from Noise. Prentice-Hall.

WAX, N. 1959. "On Upper Bounds for Error Detecting and Error Correcting Codes of Finite Length." IRE Trans., IT-5.

WELDON, E. J., Jr. 1966. "Difference-Set Cyclic Codes." BSTJ, 45.

WELDON, E. J., Jr. 1968. "New Generalisations of the Reed-Mulle Codes - Part II: Non-Primitive Codes." IEEE Trans., IT-14, 2.

WELDON, E. J., Jr. 1971. "Decoding Binary Block Codes on Q-ary OUtput Channels." IEEE Trans., IT-17.

WEST FORD EXPERIMENT. I. L. Lebow et al. 1964. "The West Ford Belt as a Communications Medium." Proc. IEEE, 52.

WOZENCRAFT, J. M. 1957. "Sequential Decoding for Reliable Communication." M.I.T. Research Laboratory of Electronics Report 325, Cambridge, Mass., U. S. A.

WOZENCRAFT, J. M., and I. M. Jacobs. 1965. Principles of Communication Engineering. John Wiley & Sons.

WYNER, A. D., and R. B. Ash. 1963. "Analysis of Recurrent Codes." IEEE Trans., IT-9.

ZETTERBURG, L. H. 1962. "Cyclic Codes for Irreducible Polynomials for Correction of Multiple Errors." IEEE Trans., IT-8.

ZIERLER, N. 1969. "A Complete Theory for Generalised BCH Codes." Proc. 1968 Syp. on Error Correcting Codes, Ed. H. B. Mann, John Wiley & Sons.

ZIMMERMAN, M. S., and A. L. Kirch. 1967. "The AN/GSC-10 (KATHRYN) Variable Rate Data Modem for HF Radio." IEEE Trans., Com-15, 2.