Coding For Frequency Hopped Spread Spectrum Satellite Communications

Final Report

Period Covered: April 1 1989 to March 31 1990

Prepared for
The Department of Communications of Canada
under DSS Contract No. 36001-8-3529/01-SS

Department of Electrical and Computer Engineering

THE UNIVERSITY OF VICTORIA
P.O. BOX 1700, VICTORIA, B.C. CANADA V8W 2Y2

Coding For Frequency Hopped Spread Spectrum Satellite Communications

Final Report
Period Covered: April 1 1989 to March 31 1990

Prepared for
The Department of Communications of Canada
under DSS Contract No. 36001-8-3529/01-SS

by

V.K. Bhargava, Q. Wang, I.F. Blake[1], G. Li, T.A. Gulliver[2], and O. Dravnieks
Department of Electrical and Computer Engineering
University of Victoria
P.O. Box 1700
Victoria, B.C.
Canada V8W 2Y2

Scientific Authority: Dr. L.J. Mason

Technical Report ECE-90-1

April 15, 1990

[1]Professor Ian Blake is with the University of Waterloo, and is a consultant to the University of Victoria on this project.

[2]Now with Defence Research Establishment, Ottawa.

Abstract

The performance of Reed-Solomon (RS) error correcting codes with slow frequency hopped (SHF) differential phase shift keying (DPSK) signalling is analyzed and evaluated under worst case partial band noise and worst case multitone jamming. A representative set of the performance curves is shown. Based on these results, recommendation on the choice of RS code parameters is given.

Two in-hop jamming cancellation schemes for SFH/DPSK systems are proposed. One scheme is based on balanced coding; the other one uses notch filter to cancel jamming tone. The performance of both schemes are illustrated. It is shown that both schemes can work well under certain conditions.

Basic principles and techniques for designing interleavers are presented. Block, convolutional and the more recent helical interleavers are considered. Certain questions are considered on the trade-off between diversity and coding for spread spectrum systems, where a low code rate is anticipated.

An error correction scheme is presented for an M-ary symmetric channel (MSC) characterized by a large error probability p_e. The value of p_e can be near, but smaller than, 1-1/M, for which the channel capacity is zero. Such a large p_e may occur for example, in a jamming environment. Monte-Carlo simulation results are presented. For the binary symmetric channel (BSC), it is shown that the overall code rate is larger than $0.6R_0$, where R_0 is the cutoff rate of the channel. For BSC and a large m, a method is presented for BER approximation based on the central limit theorem.

Logic-cell array implementation of a (31,k) "programmable" Reed-Solomon CODEC is presented as an Appendix.

Suggestions for future work include investigation of coding and detection for slow frequency hop systems using DPSK, robust techniques for generation of erasures, use of constrained sequences to cancel interference and perform error correction, analysis of coded systems using finite interleavers, trade-offs between coding and diversity and implementation aspects of CODEC.

Contents

List of Figures

List of Tables

Acknowledgement

We wish to thank Drs. Lloyd J. Mason and E. Barry Felstead for their useful comments related to material presented in this report.

One of the authors (Dr. Ian F. Blake) is grateful to Drs. Elwyn Berlekamp and G. David Forney, Jr. who made available material for Chapter 4 that would have otherwise been difficult to obtain.

Chapter 1

Introduction

1.1 Background

In previous contracts, the use of various types of channel coding were studied to improve the jamming resistance of satellite communications systems using fast frequency hopping. Systems with fixed data rate as well as systems with fixed hop rate were examined under worst case jamming. A modified self-normalizing combiner was analyzed and compared with other non-linear combining schemes. To study coding for Slow Frequency Hopped Differential Phase Shift Keying (SFH/DPSK), the probability distribution of DPSK in tone interference was derived[1].

In this annual report, we present the work performed during the year 1989-90. We first consider coding for slow frequency-hopping DPSK systems. Repeated convolutional codes for high error rate channels are analyzed. The effects of interleaving and certain questions on the trade-offs between diversity and coding are considered. We also present a tone jamming cancellation scheme for SFH/DPSK systems.

1.2 Objective

The broad objectives of the work carried out during 1989-90 are described below.

1. Consideration of coding for SFH/DPSK systems using Reed-Solomon codes under tone and partial band noise jamming.

2. Examination of low rate codes with large minimum distance for high error-rate systems.

3. Consideration of interleaving and diversity versus coding for communications over intentional interference channels.

4. CODEC implementation using current technology, e.g. Xilinx.

1.3 Plan and Scope of the Report

The plan and scope of the report is as follows. In Chapter 2, coding for slow frequency hopped differential phase shift keying systems is presented.

Chapter 3 presents an analysis of two in-hop jamming cancellation schemes for SFH/DPSK systems.

Basic principles and techniques for designing interleavers are provided in Chapter 4.

Chapter 5 presents trade-off between diversity and coding for systems, such as spread spectrum systems, where a low code rate is anticipated.

Repeated convolutional codes are examined in Chapter 6 for channels characterized by large error probabilities. Emphasis is placed on using a binary convolutional code due to the consideration that there exist commercial CODECs for such a code.

Chapter 7 contains suggestions for future work.

As an adjunct to the report, a logic-cell array implementation of a (31,k) Reed-Solomon CODEC is presented as an Appendix.

Chapter 2

Coding for Slow Frequency Hopped Differential Phase Shift Keying

2.1 Introduction

Spectrum spreading via frequency hopping is commonly utilized in satellite communications systems to provide some protection against jamming. An intelligent jammer can, however, drastically reduce the effectiveness of such a system. This effectiveness can be regained through the use of error correcting (EC) codes. This chapter presents the results of a study of the performance of EC codes in a slow frequency hopping (SFH) system with binary differential phase shift keying (DPSK). By slow we mean one or more symbols per transmitted hop. SFH is employed because the differential signalling requires the phase of the previous received symbol as a reference. Using DPSK eliminates the need to establish a phase reference for the hop.

Transmitted signals hop over a total spread spectrum bandwidth W_{ss}. If the total jamming power is J (referenced to the receiver input), the effective jamming power spectral density is

$$J_O = J/W_{ss}.$$

The objective is to minimize the bit error rate (BER) for a given signal to jamming ratio, E_s/J_O. E_s is the energy in a DPSK symbol. An EC code is used to improve upon the severely degraded performance of uncoded DPSK when it is jammed. The analysis of the

coded channel is based on [2].

Under strong jamming, the receiver thermal or non-hostile background noise is usually small compared to the jamming, so receiver noise is neglected here. We consider two types of worst case (WC) intelligent but non-repeat-back jamming, namely partial band noise (PBN) and multitone (MT) interference. For partial band noise (PBN) jamming, J is restricted to a fraction ρ $(0 < \rho \leq 1)$ of the full spread spectrum bandwidth, but in this band the power spectral density is increased to J_O/ρ. Multitone jamming (MT) occurs when the jammer distributes J as continuous wave tones across W_{ss}. There are $N = W_{ss}T_s$ possible tone positions, where T_s is the signal symbol duration. If the jammer can place tones in N_t of these positions, then the fraction of the spread spectrum band which is jammed is $\rho = N_t/N$. In anti-jam communications, a good code should perform well regardless of the type of jamming. Thus good codes are those with the best performance for the most effective type of jamming, WC MT jamming or WC PBN jamming, at a given low BER. The WC BER performance of an RS code with a specific set of parameters is a function of E_s/J_O and ρ, where ρ is optimised to determine ρ_{wc}.

We consider (n,k) Q-ary Reed-Solomon (RS) block codes with symbol size $q = \log_2 Q$ bits, block length $Q - 1$, k information symbols per block, and minimum distance $d = n - k + 1$. RS codes are maximum distance separable, that is they achieve the highest possible distance for their code parameters. Since jamming tends to cause burst channel errors, RS codes are well suited for this system. In addition, RS codes have a low probability of decoding error [3].

2.2 Analysis of the Coded System

To find the BER for WC jamming, we proceed as follows. Suppose the DPSK symbol energy is E_s, and a hop has L_b coded bits, (note that each hop should in addition contain a phase reference bit). With an RS code, L_b bits can affect $\lceil \frac{L_b}{q} \rceil$ symbols, if L_b is chosen so that code symbols are aligned to a particular hop. $\lceil x \rceil$ is the smallest integer greater than or equal to x. Here we assume $L_b \geq q$. Then a codeword can be affected by about

$$H = \lceil \frac{nq}{L_b} \rceil$$

hops. When a hop is jammed, L_b bits in this hop will be in error independently with probability P_J. Although this independence assumption is not true in general, it has been shown to provide a good approximation in PBN jamming, and seems reasonable for MT jamming based on previous experience and results. P_J is derived from [4] and [2], with ρ chosen to be the worst possible, ρ_{wc}. For PBN jamming, ρ represents the fraction of the spread spectrum bandwidth which is jammed, while for MT jamming ρ represents the fraction of the total number of available frequency slots which are jammed. P_J is then given by

$$P_J = \begin{cases} \frac{1}{2}\exp(-\rho E_s/J_O), & \text{PBN jamming;} \\ P_{s1}(\frac{1.0}{\sqrt{\rho E_s/J_O}}, E_s/J_O), & \text{MT jamming,} \end{cases}$$

with the probability of error under MT jamming, P_{s1}, defined by Eq. (52) in [2]. The number of erroneous q-ary symbols in these $\lceil \frac{L_b}{q} \rceil$ symbols corresponding to a jammed hop is denoted as z. Then the probability that l symbols are in error is given by

$$P_r(z=l) = \begin{pmatrix} \lceil L_b/q \rceil \\ l \end{pmatrix} (1-(1-P_J)^q)^l (1-P_J)^{q(\lceil L_b/q \rceil - l)}.$$

The RS code output symbol error probability is then

$$P_s \approx \sum_{i=1}^{H} \begin{pmatrix} H \\ i \end{pmatrix} \left[\sum_{\sum_{j=1}^{i} z_j > t} P_r(\sum_{j=1}^{i} z_j > t) \left(\frac{\sum_{j=1}^{i} z_j}{n} \right) \rho^i (1-\rho)^{H-i} \right],$$

where t is the number of symbol errors which can be decoded by an RS code,

$$t = \lfloor \frac{d-1}{2} \rfloor,$$

and $\lfloor x \rfloor$ is the largest integer less than or equal to x. z_j is the number of erroneous q-ary symbols in the j-th jammed hop, given that i hops are jammed in a codeword. The event for a specific set of z_j's is denoted by

$$\sum_{j=1}^{i} z_j > t.$$

The summation is over all sets that cause a decoding failure. The probability of these events occurring is

$$P_r(\sum_{j=1}^{i} z_j > t)$$

which is computed using $P_r(z = l)$.

The final BER is given by

$$P_b = \frac{Q}{2(Q-1)} P_s.$$

ρ_{wc} is the value of ρ which maximises P_b, for a given L_b, RS EC code, (n, k), and type of jamming. The objective is then to determine the effects of these parameters on the BER and develop a set of guidelines for proper EC code selection. In the next section, various RS codes are evaluated and the results compiled.

2.3 Computation Results

To provide a benchmark and a check on the optimisation algorithms, uncoded DPSK was first evaluated. These results are given in Fig. 2.1. They are identical to the worst case results given in [4]. The performance of Reed-Solomon error correcting codes with $q = 3, 4, 5$ and 6, and $L_b \leq 2^q - 1$, was determined under worst case jamming via optimisation with respect to ρ. Since error probability evaluation for PBN jamming is much less computationally intensive than for MT jamming, candidate good codes were first found for this type of jamming and then the performance determined for MT jamming.

For $q = 3$, the $(7, k)$ RS codes require $L_b \leq 21$. Figs. 2.2 to 2.7 present the performance results of these codes for $k = 1, 3, 5$ and $L_b = 6, 11, 21$. From these figures, it can be seen that the $(7, 5)$ code is unable to improve upon uncoded DPSK. For the $(7, 1)$ and $(7, 3)$ codes, and $L_b = 6$, there is a dramatic improvement. In this case, performance with WC MT jamming is 3 to 5 dB worse than with WC PBN jamming. The next set of curves shows the $(15, k)$ RS codes. We examined the $(15, 5)$ and $(15, 9)$ codes, with $L_b = 10, 15, 20, 30, 60$, with results given in Figs. 2.8 to 2.11. The $(15, 5)$ code shows improvement when $L_b \leq 20$ and the $(15, 9)$ when $L_b = 10$. From this we can conclude that substantial performance improvements over uncoded DPSK can be achieved only when $t \geq L_b/q$. Otherwise, the RS code cannot correct the erroneous bits on a jammed hop and decoding will not succeed, resulting in a performance near that of uncoded DPSK. Finally, we evaluated the block length 31 and 63 RS codes. Results for the $(31, 11)$ and $(63, 31)$ codes are presented here in Figs. 2.12 to 2.15. From these figures, we again see the role

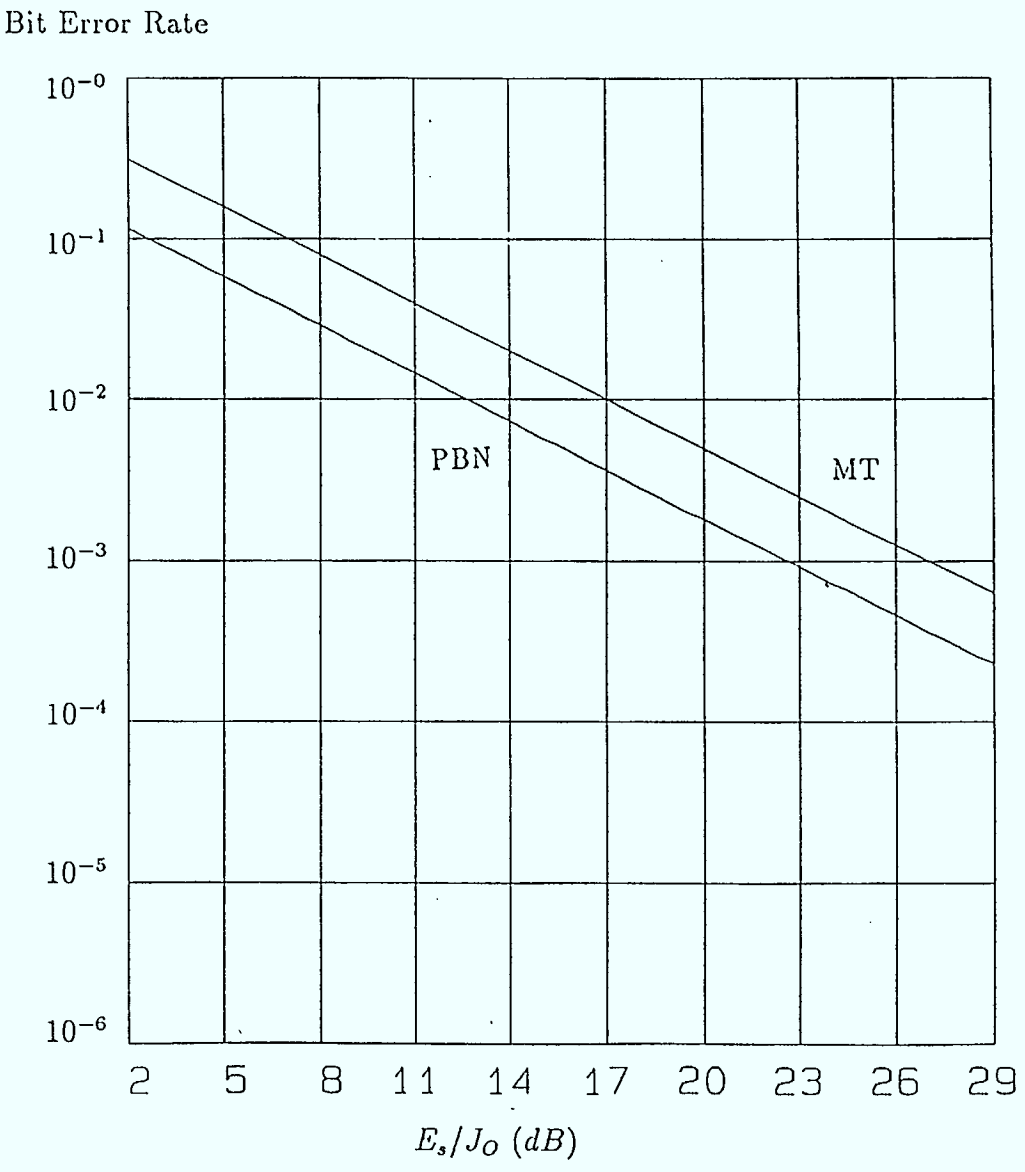


Figure 2.1: Uncoded SFH/DPSK Performance under worst case PBN and MT jamming.

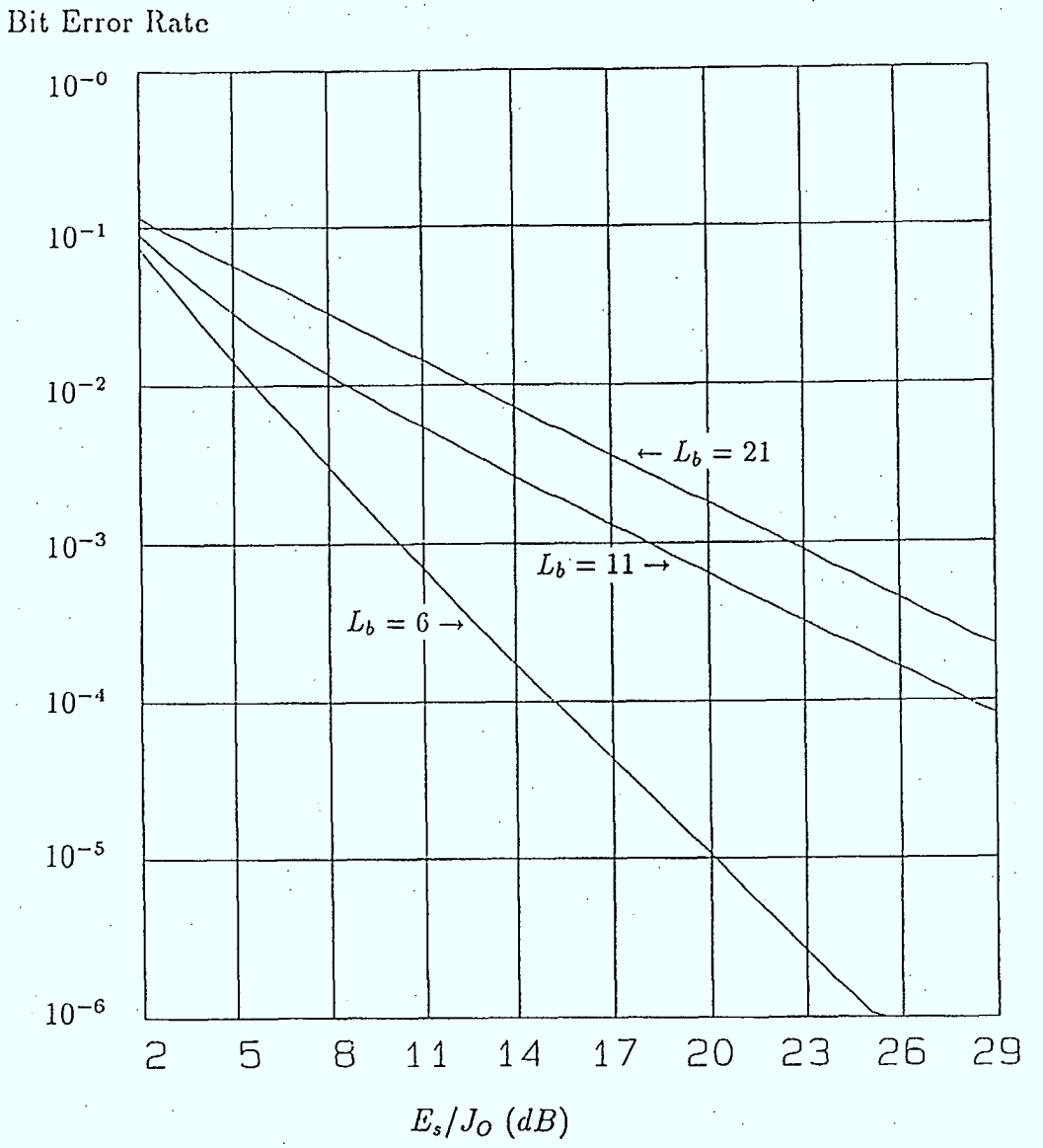


Figure 2.2: SFH/DPSK Performance of (7,1) RS code under worst case PBN jamming.

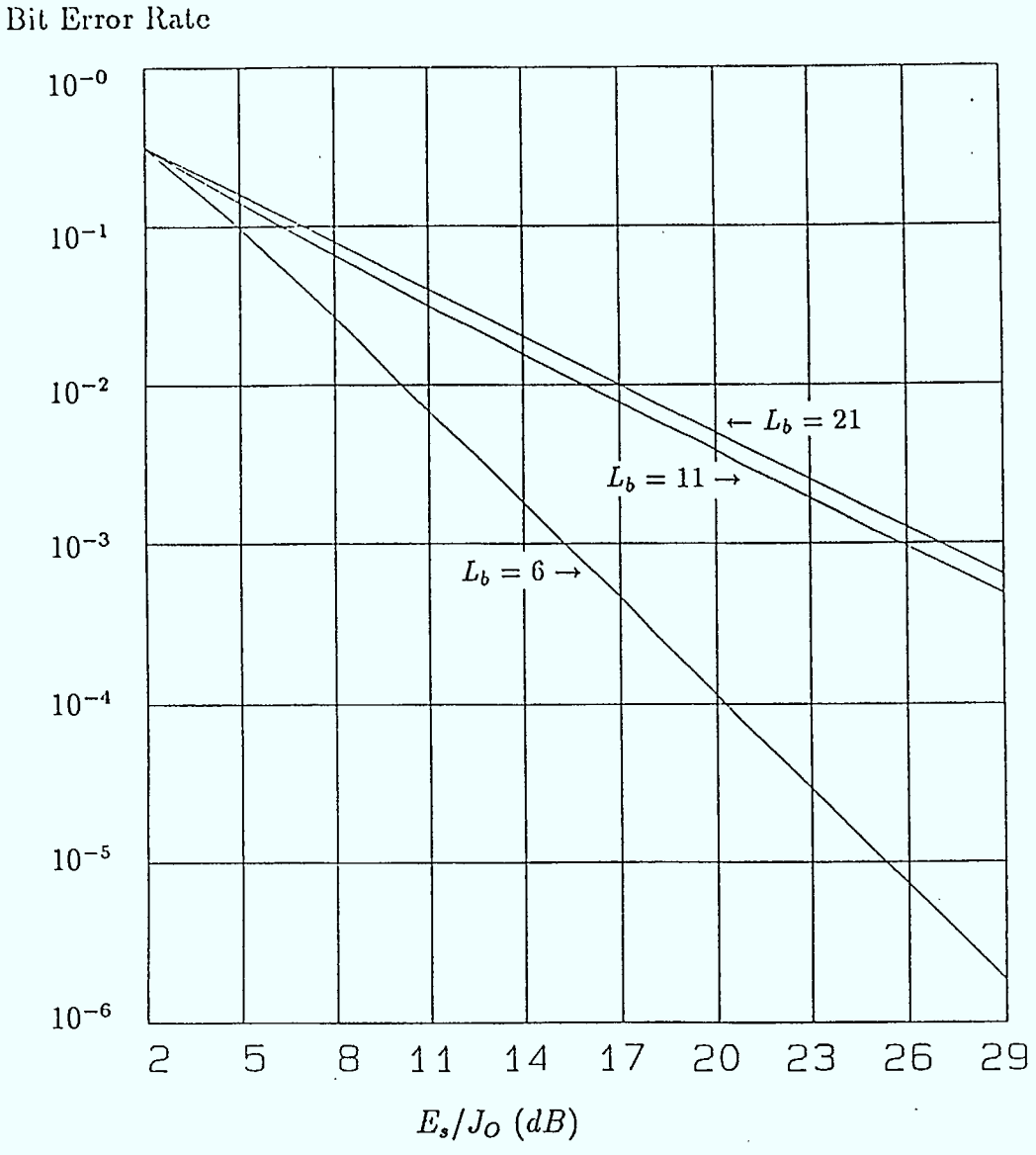


Figure 2.3: SFH/DPSK Performance of (7,1) RS code under worst case MT jamming.

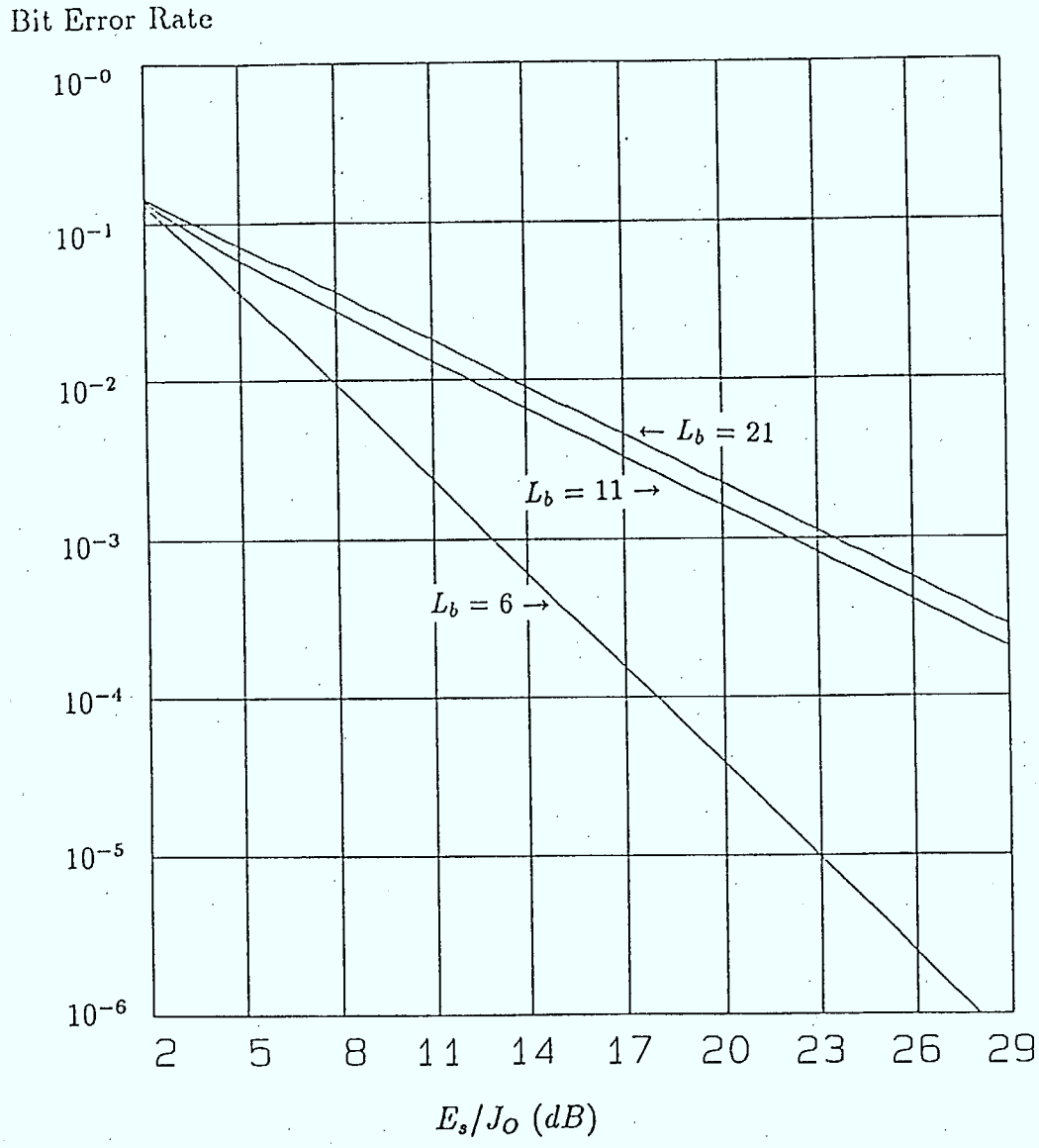


Figure 2.4: SFH/DPSK Performance of (7,3) RS code under worst case PBN jamming.

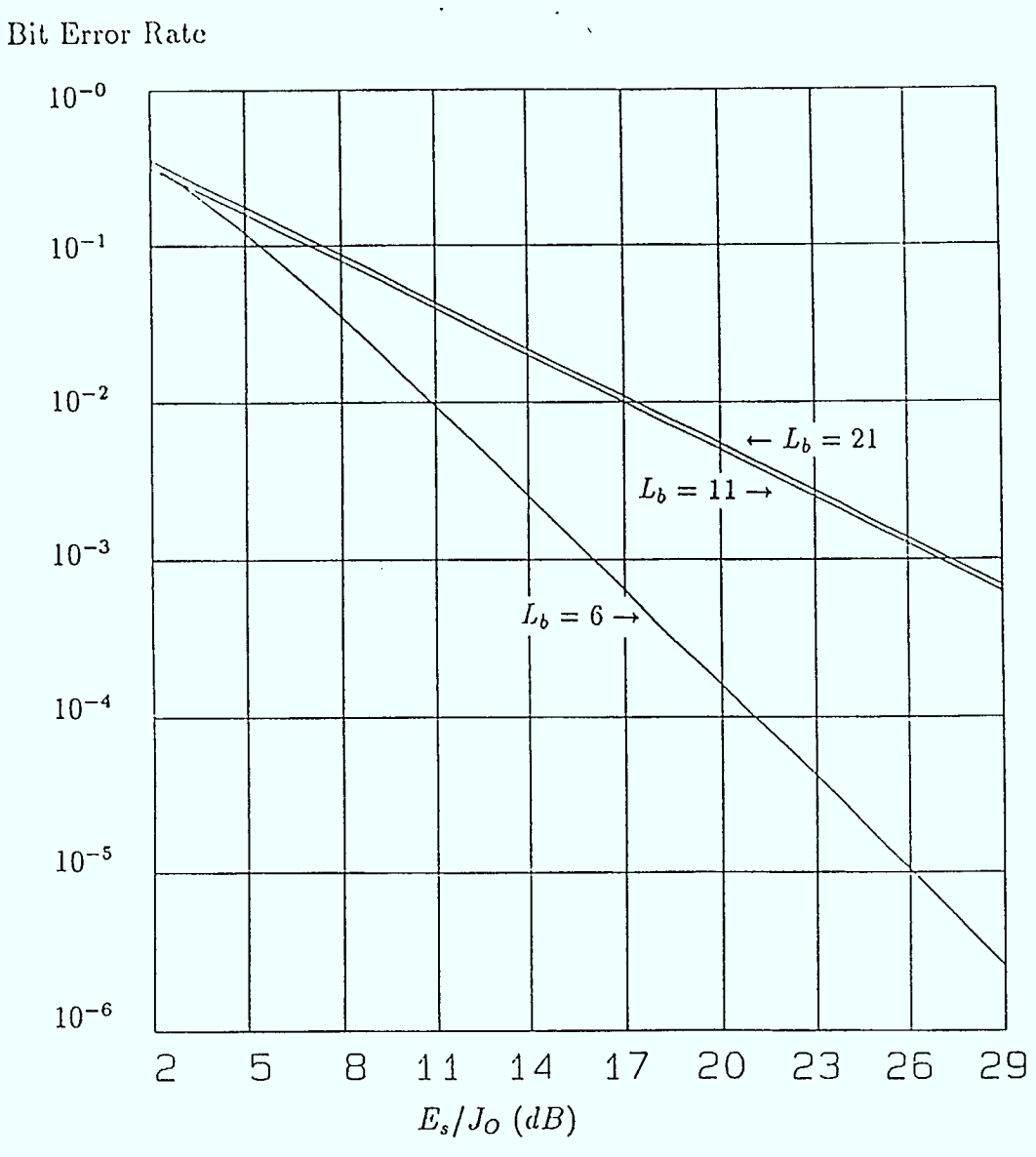


Figure 2.5: SFH/DPSK Performance of (7,3) RS code under worst case MT jamming.

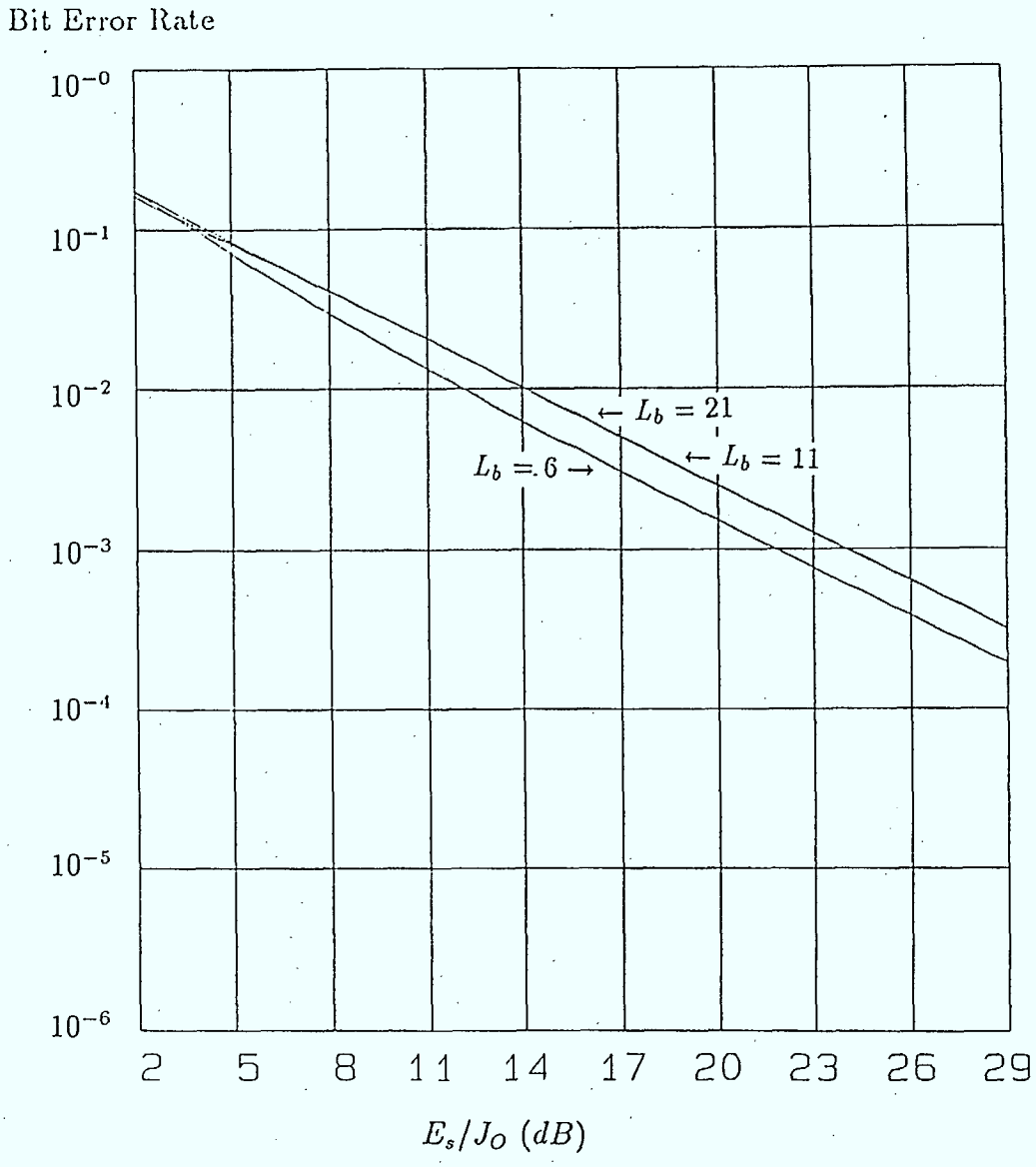


Figure 2.6: SFH/DPSK Performance of (7,5) RS code under worst case PBN jamming.

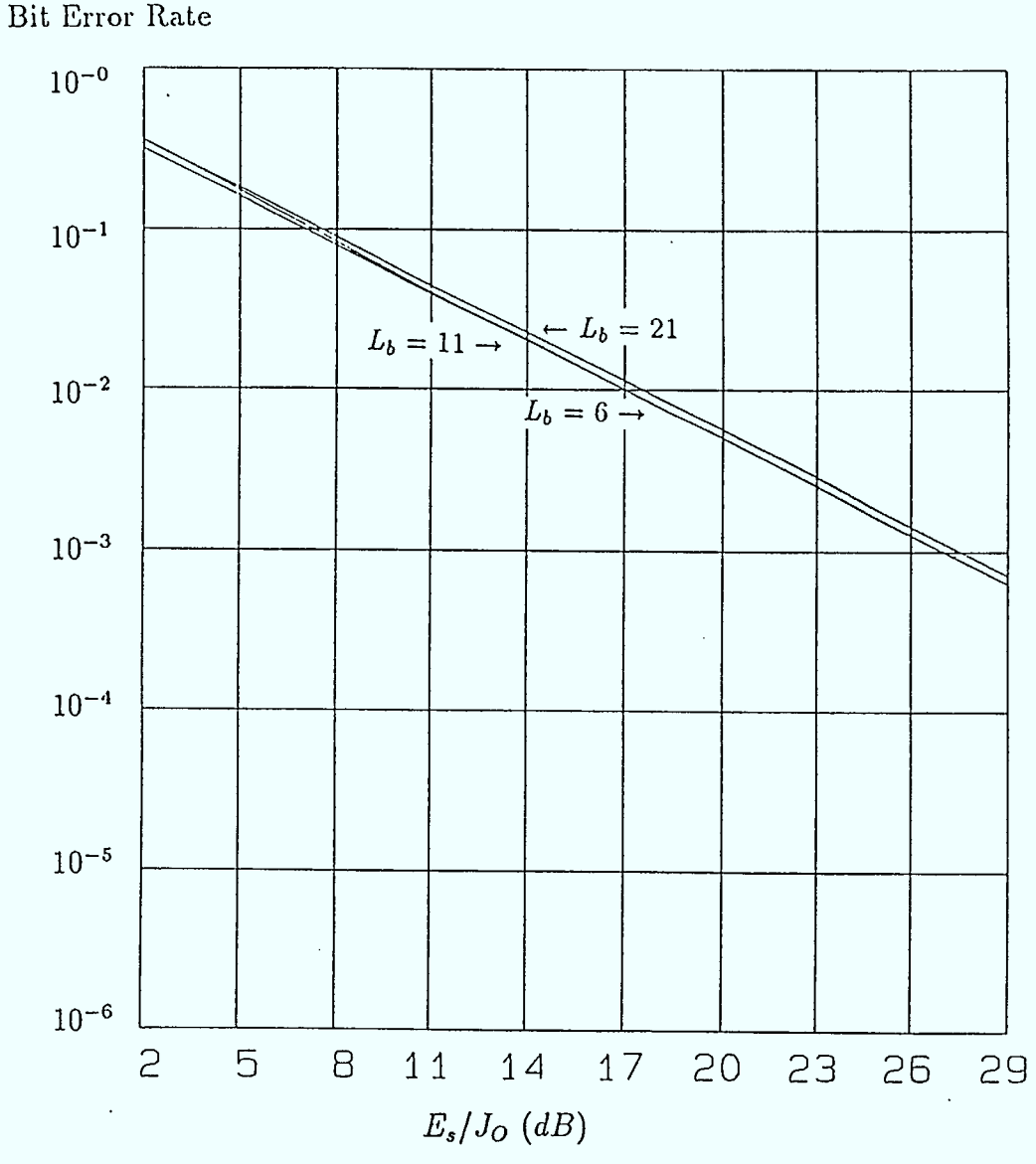


Figure 2.7: SFH/DPSK Performance of (7,5) RS code under worst case MT jamming.

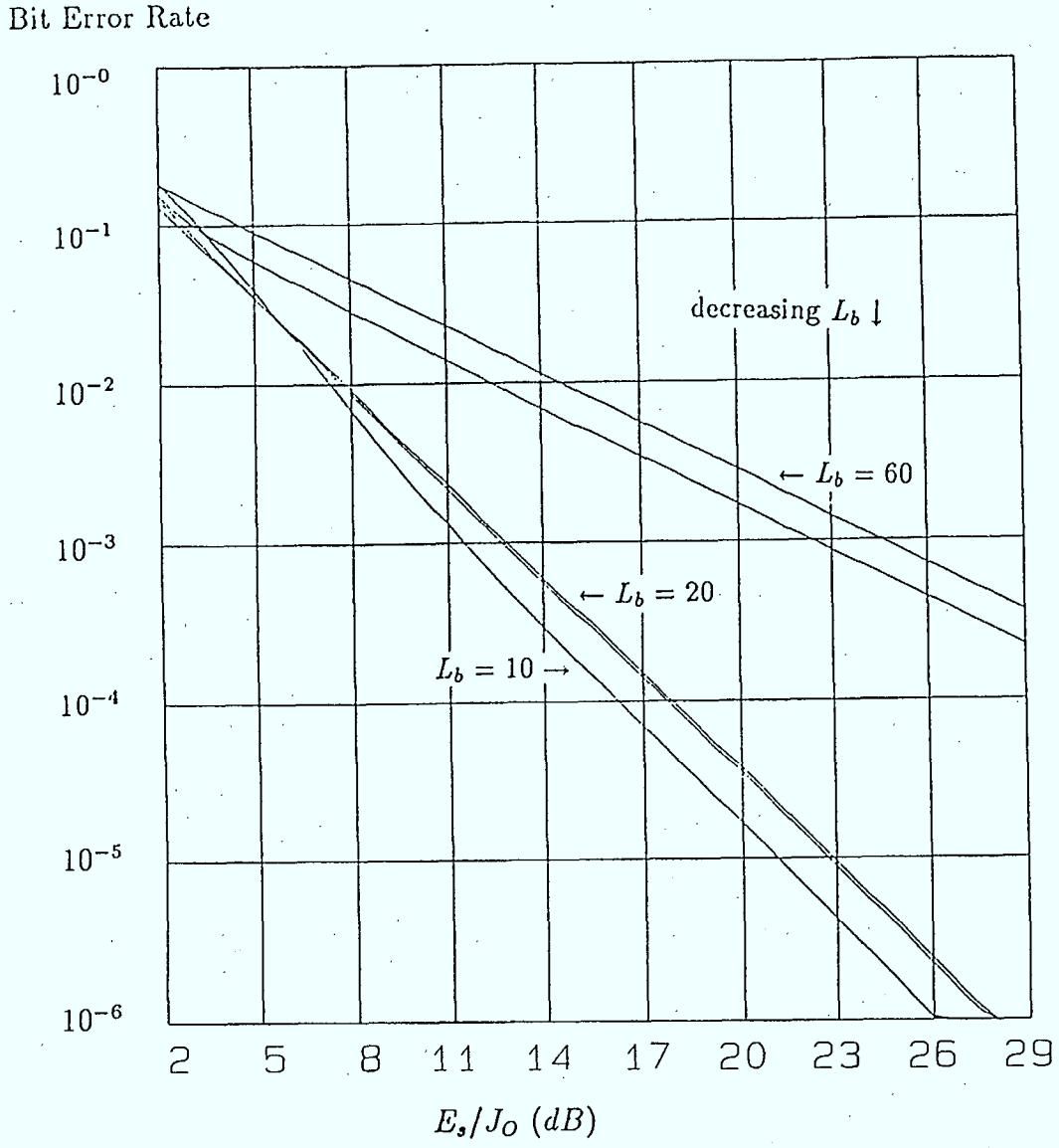


Figure 2.8: SFH/DPSK Performance of (15,5) RS code under worst case PBN jamming. $L_b = 10, 15, 20, 30, 60$.

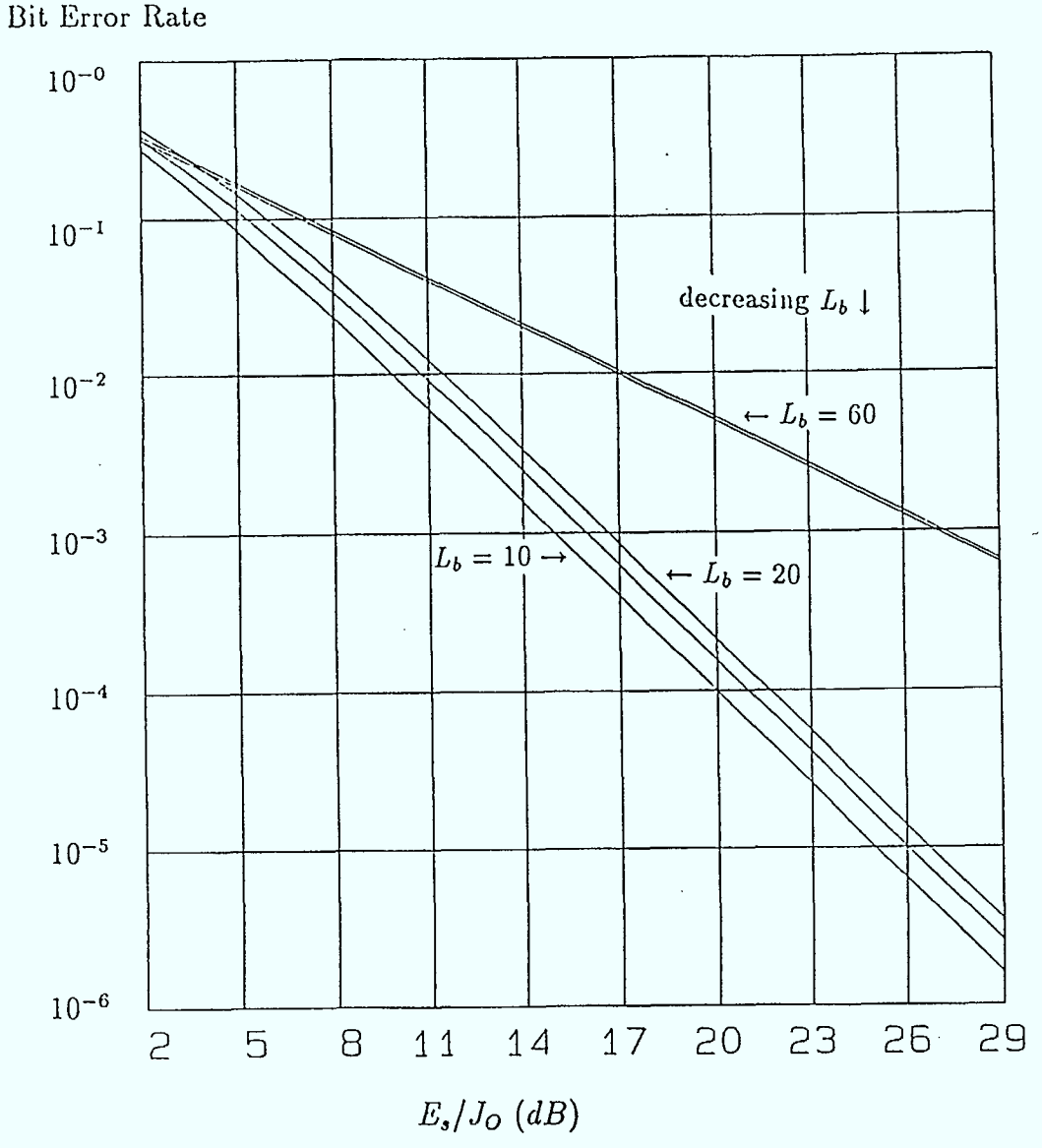


Figure 2.9: SFH/DPSK Performance of (15,5) RS code under worst case MT jamming. $L_b = 10, 15, 20, 30, 60$.

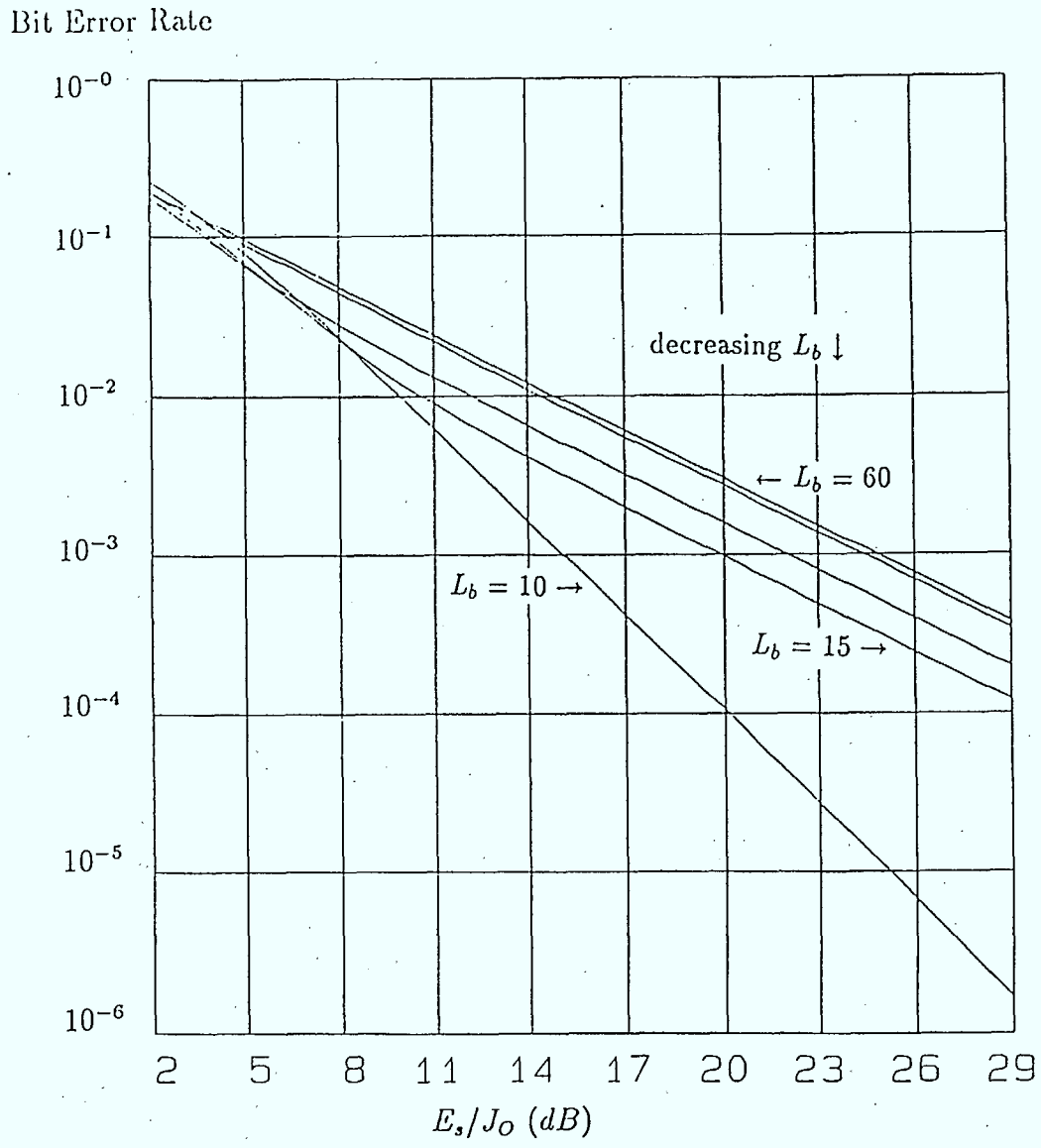


Figure 2.10: SFH/DPSK Performance of (15,9) RS code under worst case PBN jamming. $L_b = 10, 15, 20, 30, 60$.

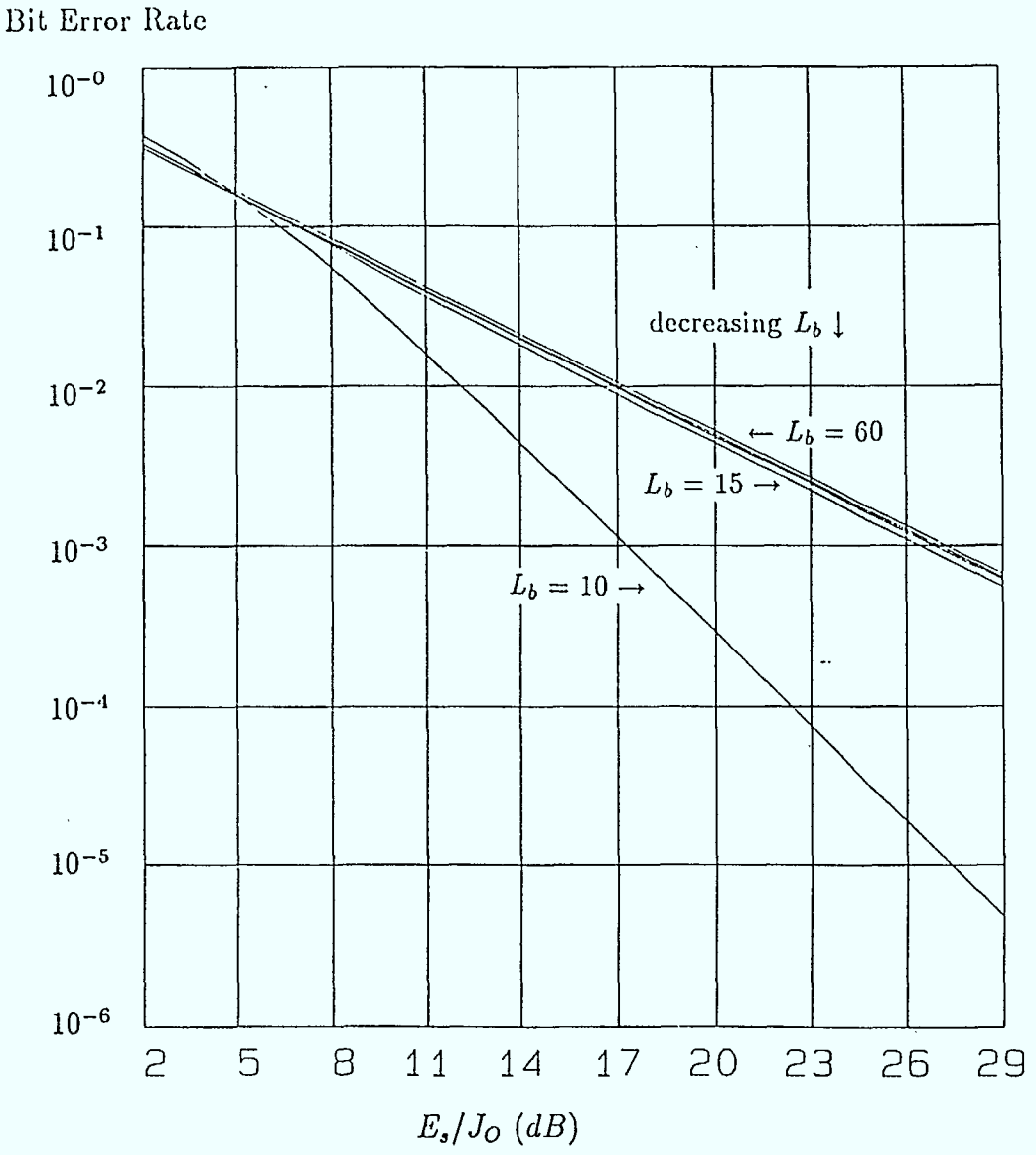


Figure 2.11: SFH/DPSK Performance of (15,9) RS code under worst case MT jamming. $L_b = 10, 15, 20, 30, 60$.

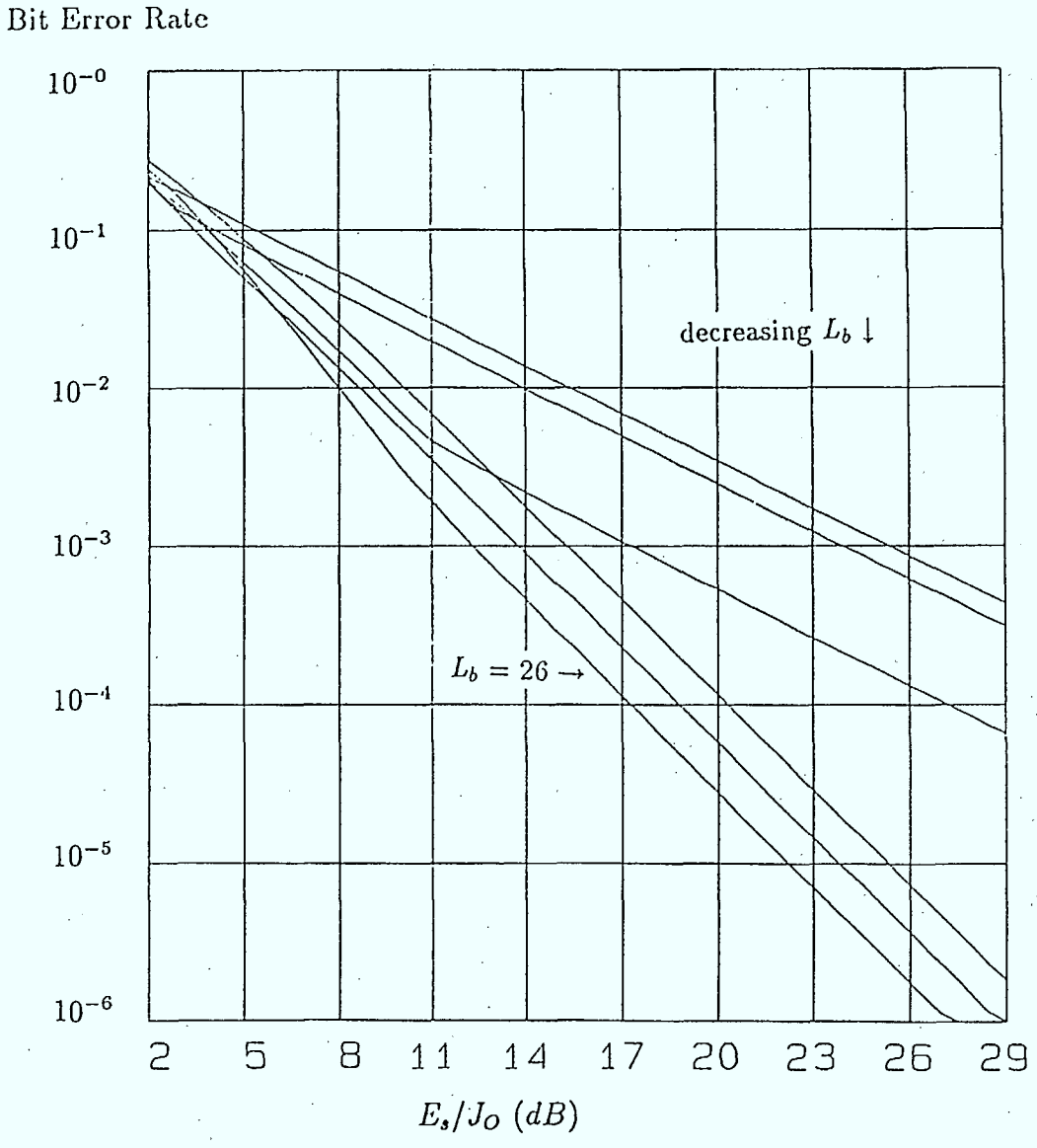


Figure 2.12: SFH/DPSK Performance of (31,11) RS code under worst case PBN jamming. $L_b = 26, 39, 50, 52, 78, 155$.

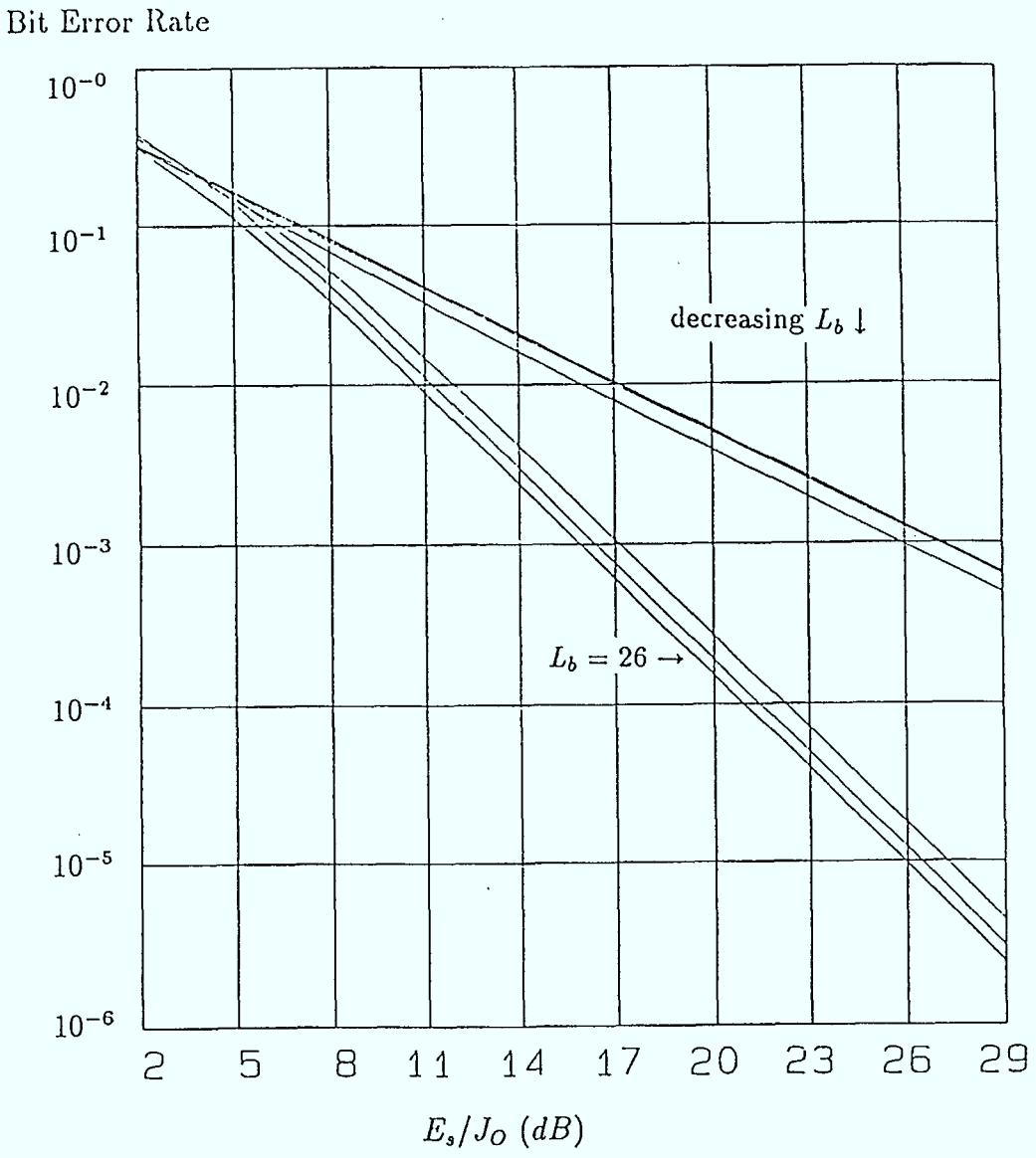


Figure 2.13: SFH/DPSK Performance of (31,11) RS code under worst case MT jamming. $L_b = 26, 39, 50, 52, 78, 155$.

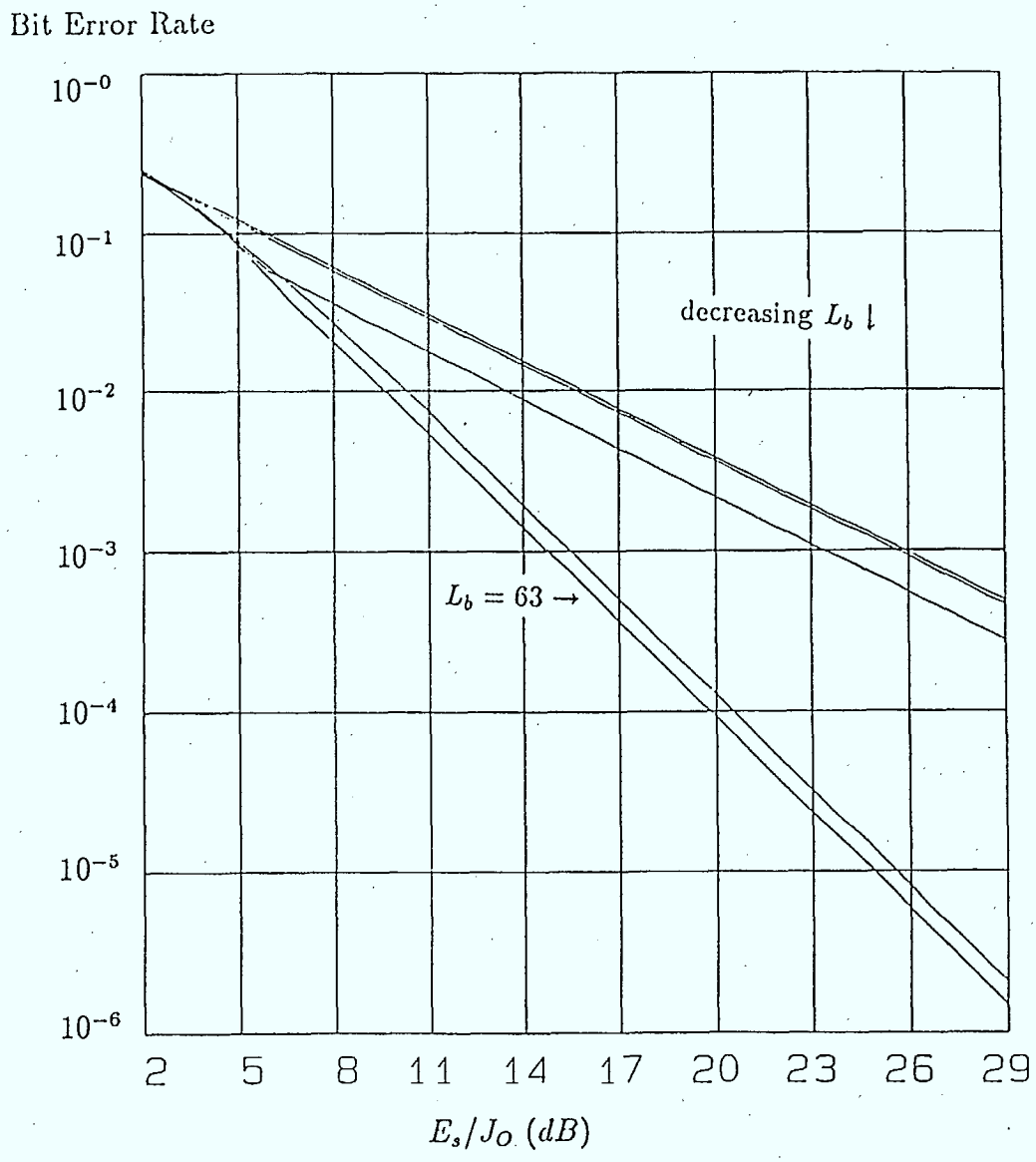


Figure 2.14: SFH/DPSK Performance of (63,31) RS code under worst case PBN jamming. $L_b = 63, 95, 126, 189, 378$.

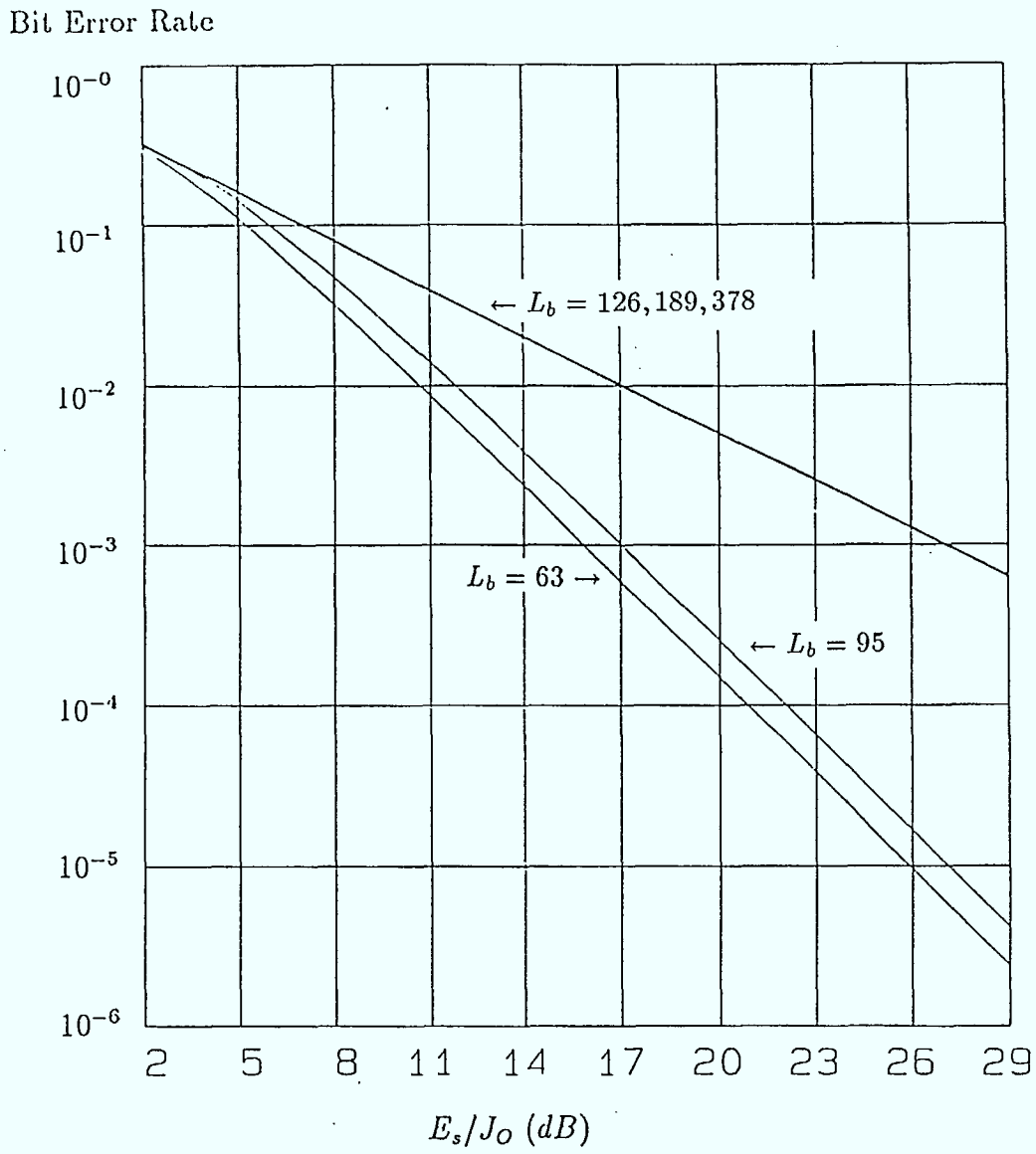


Figure 2.15: SFH/DPSK Performance of (63,31) RS code under worst case MT jamming. $L_b = 63, 95, 126, 189, 378$.

L_b/q plays in the performance. With the $(31, 11)$ code, $t = 10$ symbols, or $tq = 50$ bits can be corrected. Thus only for $L_b \leq 50$ is performance improved. This is most dramatically shown in Figs. 2.12 and 2.13 when L_b is dropped from 52 to 50. Similarly for the $(63, 31)$ code, when $L_b \leq 96$ performance is dramatically improved. WC MT jamming is always worse than WC PBN jamming, and for small L_b the difference is 2 to 5 dB.

2.4 Concluding Remarks

From the results of the previous section it is clear that the number of codeword symbols per hop must be small in order for the RS code to provide protection against jamming. Otherwise, no improvement over uncoded DPSK is gained. Lowering the symbols per hop can be achieved either by reducing the number of bits per hop, as was done here, or interleaving the RS codewords to a depth determined by the hop length and RS code parameters.

It remains to evaluate the performance when diversity is also employed and when nonbinary DPSK is used. Both of these directions should provide performance improvements over the results found in this chapter. As well, the use of interleaving will reduce the number of erroneous symbols in a given RS codeword when a hop is jammed. It also allows a long hop length, which reduces the amount of lost data due to the phase reference bit.

Chapter 3

Tone Jamming Cancellation in SFH/DPSK Systems

3.1 Introduction

In a slow frequency hopped (SFH) differential phase shift keying (DPSK) system, there are typically at least a few tens of bits in a hop. We can not afford the loss of even one hop. One way to improve the system anti-jam capability is to employ a long error correction code such as, a Reed-Solomon code. Error correction codes with long codewords spanned over several hops can be designed to correct burst errors as well as random errors. Random error correcting codes with deep interleaving can also be used. This type of method is based on multiple-hop information.

For tone jamming, however, it is possible to employ some signal processing techniques to combat jamming in a single hop. This can improve system performance significantly. If the system in-hop anti-jam capability is increased, the jammer must put more jamming power in a frequency slot to achieve the same jamming effect. Thus the total number of frequency slots jammed will be reduced when total jamming power is constant. Therefore the whole system anti-jam capability is increased.

In a previous report, we have analyzed the probability distribution of signals in DPSK systems in tone interference [1]. In this chapter, we propose two in-hop jamming cancellation schemes for SFH/DPSK systems. One is based on using balanced coding, while the other one uses adaptive filtering to cancel the jamming tone. The performance of these

schemes is analyzed as well.

3.2 Cancelling Tone Jamming by Balanced Coding

3.2.1 Assumptions

We consider a SFH spread spectrum system with binary DPSK modulation. We assume that the amplitude of the transmitted signal is constant, that initial phase of the signal does not vary in a hop, and that the amplitude information of the received signals is available. This means that there are no envelope limiting circuits in the receivers. We also assume that the frequency of the jamming tone is the same as the carrier frequency, so that the amplitude and initial phase of the jamming tone are constant in a hop.

3.2.2 Problem Description

In a SFH/DPSK system, when a hop is hit by a multitone jammer, the received signals in the hop contains highly correlated interference. Assume that the transmitted DPSK signals in a hop are: $\vec{E}_1, \vec{E}_2, \cdots, \vec{E}_m$,

$$\vec{E}_i = Ee^{j\theta_i} \qquad i = 1, 2, \cdots, m$$

where m is the number of channel symbols in a hop, and E and θ_i are the amplitude and phase of the ith transmitted signal, respectively. Then the received signals under tone jamming are: $\vec{R}_1, \vec{R}_2, \cdots, \vec{R}_m$,

$$\vec{R}_i = Ee^{j\theta_i} + Ie^{j\theta_J} \qquad i = 1, 2, \cdots, m$$

where $Ie^{j\theta_J}$ is the tone jamming with amplitude I and phase θ_J which is uniformly distributed in $[0, 2\pi)$. In DPSK modulation information is carried by phase change; therefore detection is usually based on phase change of two consecutive signals.

When signal to thermal noise ratio (SNR) is high, the influence of tone jamming on DPSK signal can be illustrated by a geometric relation, as depicted in Fig. 3.1 (a) and (b).

In Fig. 3.1 (a), there is no phase change between two consecutive transmitted signals, $\vec{E}_1$ and $\vec{E}_2$, i.e. $\angle(\vec{E}_1, \vec{E}_2) = 0$, where $\angle(\cdot, \cdot)$ is the angle between two vectors. It is

easy to see that there is also no phase change between two consecutive received signals, $\vec{R}_1$ and $\vec{R}_2$. So a correct decision can always be made no matter how large the tone jamming is (when neglecting errors due to thermal noise, and noting that the probability of the jamming vector exactly cancelling the signal vector is zero). Therefore tone jamming does not have direct influence on detection in this situation. However, an indirect influence is the change in the probability of erroneous decision due to thermal noise as a result of the amplitude change in the received signal caused by the jamming component.

In Fig. 3.1 (b), there is π radian phase change between two consecutive transmitted signals, i.e., $\angle(\vec{E}_1, \vec{E}_2) = \pi$. We can see that due to strong tone jamming the phase change of two consecutive received signals, $\vec{R}_1$ and $\vec{R}_2$, can be less than $\frac{\pi}{2}$. Therefore an erroneous decision can be made. When jamming is strong enough the phase difference between $\vec{R}_1$ and $\vec{R}_2$ can always be less than $\frac{\pi}{2}$, hence a zero phase change will be incorrectly detected.

In summary, the main influence of tone jamming on DPSK system is that the probability of the receiver not detecting phase change between two signals is much higher. This conclusion is directly based on the assumption that jamming tone hits the carrier frequency exactly.

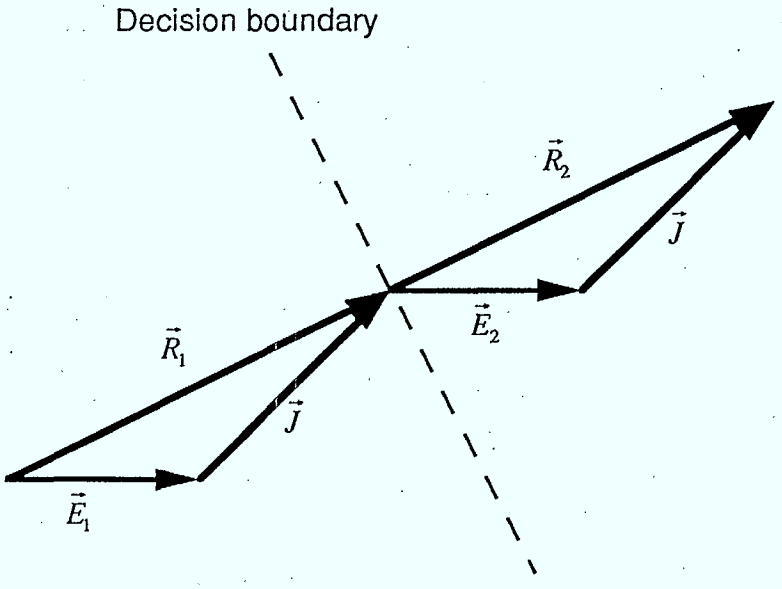
One way to combat tone jamming is to cancel the jamming tone before a decision is made. This can be carried out by employing the so called balanced code.

3.2.3 Cancelling Scheme

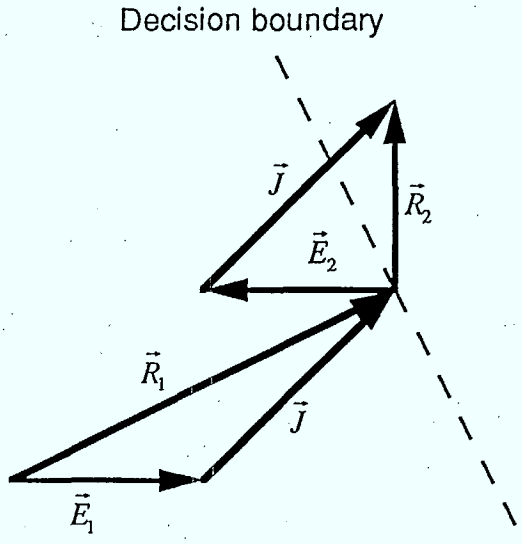
Consider a block of transmitted signal vectors such that the sum of vectors is zero, and suppose the whole block is transmitted in the same hop. Then neglecting the influence of thermal noise, the vector sum of the corresponding received symbols divided by the number of symbols in the block is the tone jamming component. Thus the amplitude and phase of the jamming tone can be estimated from the sum of the recieved signals, and then the tone jamming can be cancelled from the received signals.

Suppose the block length is n signals, and the transmitted signals are s_i, $i = 1, 2, \cdots, n$. The requirement on the transmitted signals is

$$\sum_{i=1}^{n} s_i = 0. \tag{3.1}$$



(a)



(b)

Figure 3.1: The effect of tone jamming on phase change detection when noise is small, and the jamming tone hits the carrier exactly.

In the binary case, Equation (3.1) means that in the sequence there must be the same number of signals with a phase θ and signals with a phase $\theta + \pi$. This can be achieved by making the codeword balanced before PSK modulation.

A balanced code has codewords with equal numbers of zeros and ones. One important feature of a balanced code is that it is dc-free. They have been used in magnetic storage and optical communication systems where a dc-free signal is required. Here we can use the dc-free feature to estimate jamming tone.

For a coherent binary PSK system, the balanced code can be used by simply placing the balanced code encoder before PSK modulation and the balanced code decoder after PSK demodulation. But for binary DPSK systems it is more complex to use a balanced code. Two possible schemes are considered in this report.

One scheme (Scheme I) is shown in Fig. 3.2. Differential encoded information data is encoded with a balanced code before PSK modulation. At the receiver the received signal is first downshifted to the baseband to generate in-phase and quadratic components. The balanced encoded signals at the baseband have a property that the vector sum of signals in a codeword is zero. Then the average of received signals in a balanced codeword is computed. The average is an estimation of the jamming tone. Thus by subtracting the average from each codeword the tone jamming is cancelled. The data is then passed to what we call a semi-differential PSK demodulator followed by a balanced decoder and a differential decoder.

The semi-differential PSK demodulator is a PSK demodulator based on differential phases of consecutive received signals. It works in the following way: assume the first received signal represents 1 (or 0 arbitrarily), then compare the phase difference between the second and the first received signals. If the phase difference is greater than $\pi/2$, then declare the second signal to represent 1; otherwise it represents 0. This procedure is repeated for the following signals.

The output of the semi-differential PSK demodulator is a received balanced coded sequence or its inversion. For instance, if the balanced sequence is 1010 ..., the received sequence may be 1010 ..., or 0101 .... The ambiguity comes from the differential PSK demodulator. Therefore, in order to recover the original information data with a differ-

ential decoder following a balanced decoder, the balanced code should be transparent[8]. Transparent code is a code in which an inversion of a codeword is also a codeword, and the corresponding information bits of a codeword and its inversion also have an inverse relation. Thus, by using transparent balanced coding, the tone jamming can be cancelled, and the original information sequence can be obtained from outputs of the differential decoder.

The other scheme (Scheme II) is shown in Fig. 3.3. The information data is encoded with a balanced code before PSK modulation. The cancellation algorithm is the same as that in the first scheme. Following the cancellation circuits is a differential demodulator in a DPSK receiver. The output of the decoder is a differentially decoded balanced sequence. A mapping between a differentially decoded balanced codeword and its corresponding information bits is used to decode the differentially decoded balanced codeword sequence.

Specifically, suppose we use a (n,k) balanced code. k information bits are encoded into a balanced codeword n bits long. At the receiver the differential demodulator is operated on every balanced codeword, and generates a $n-1$ bit long output, which is a differentially decoded balanced codeword and corresponds to the original k bits of information.

Because a codeword and its inversion are identical after being differentially decoded, the inversion of a balanced codeword should either correspond to the same information bits or not be a valid codeword.

3.3 Performance Analysis

We first analyze the tone jamming cancellation performance, which is the same for both schemes. The symbol error probability at the output of the differential demodulator in scheme II is derived. This symbol error probability is the same as the transition error probability at the output of the semi-differential PSK demodulator, where the transition of signals carries information. Next we discuss different balanced coding methods and performances in the two schemes separately.

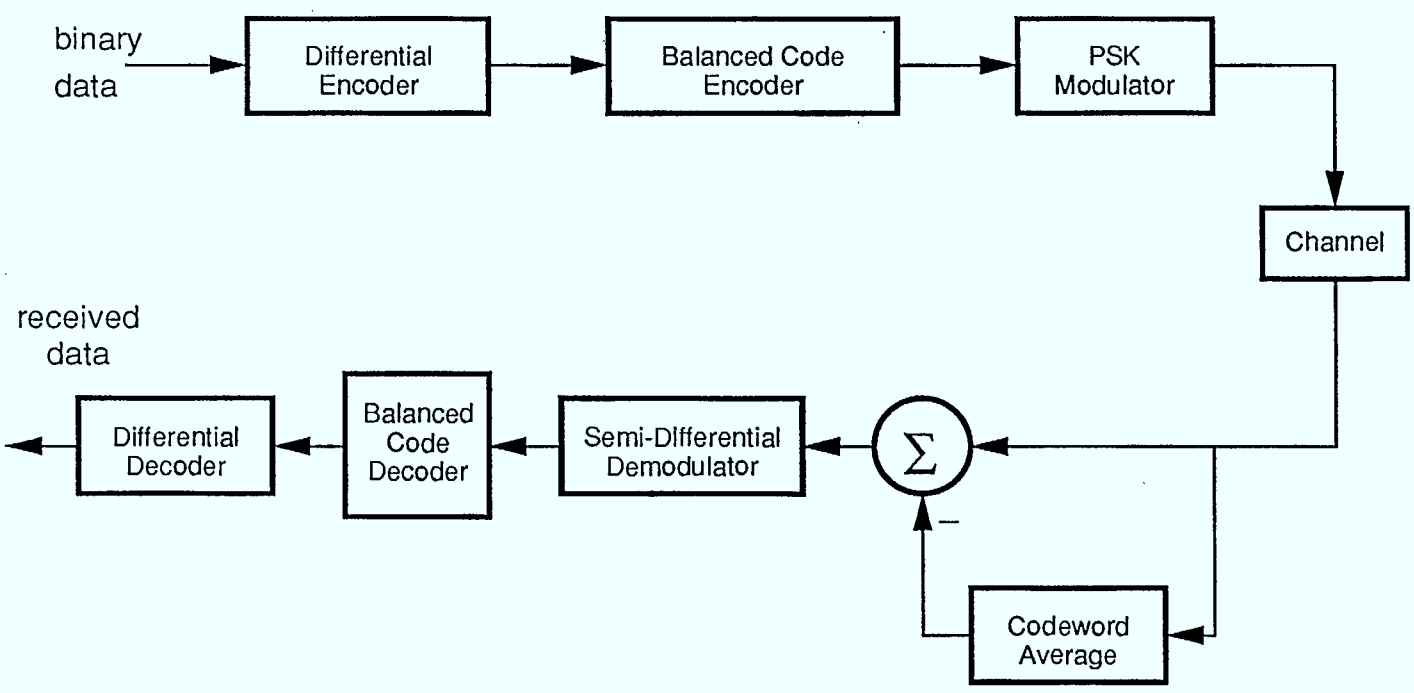


Figure 3.2: Balanced coding tone jamming cancellation scheme I.

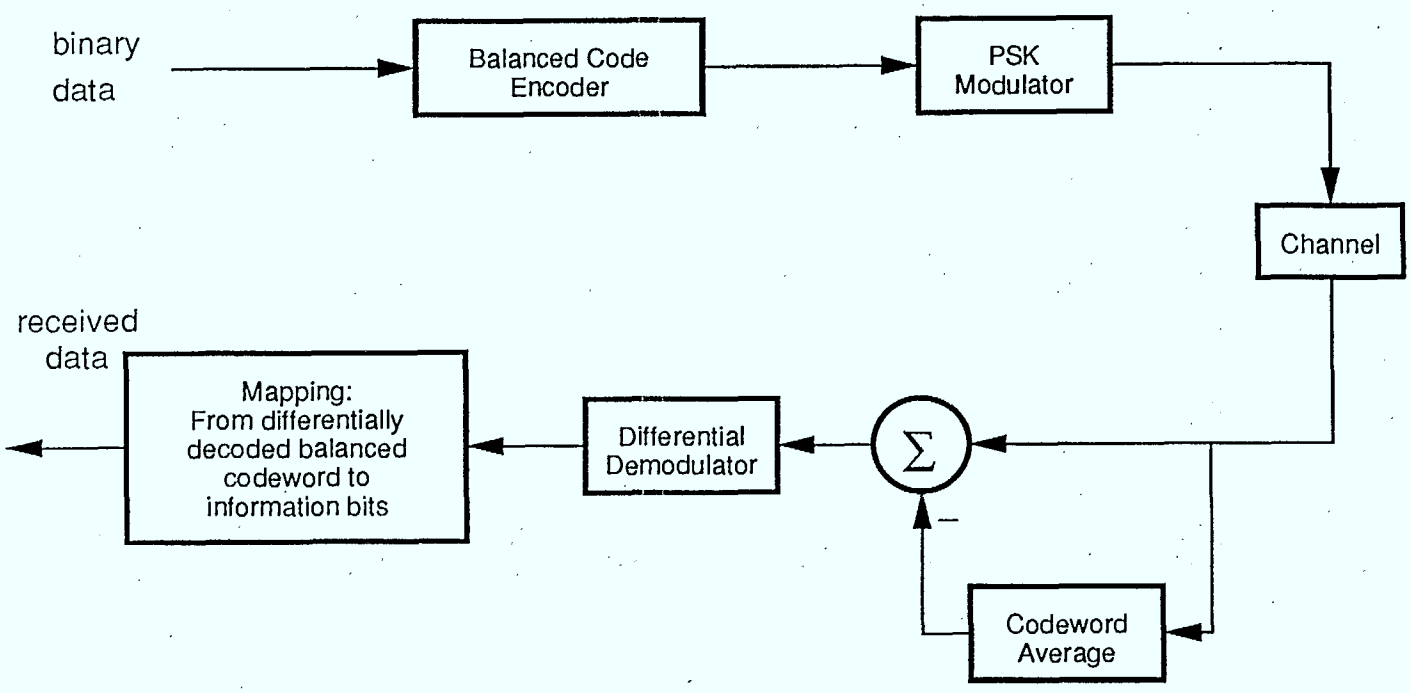


Figure 3.3: Balanced coding tone jamming cancellation scheme II.

3.3.1 Symbol Error Probability at Differential Demodulator Output in Scheme II

Let us observe a received data sequence corresponding to a balanced codeword, r_1, r_2, ... , r_n, and where

$$r_i = s_i + Ie^{j\theta_J} + n_i \qquad i = 1, 2, ..., n$$

where n is the length of balanced code; s_i is the transmitted signal with energy E_s; $Ie^{j\theta_J}$ is the tone jamming; and

$$n_i = x_i + jy_i$$

where x_i and y_i are zero mean additive white Gaussian noise (AWGN) with variance $\sigma^2 = N_0/2$ respectively. N_0 is the AWGN spectral density.

The arithmetic average of the sequence is

$$a = \frac{1}{n}\sum_{i=1}^{n} r_i = \frac{1}{n}\sum_{i=1}^{n} s_i + Ie^{j\theta_J} + \frac{1}{n}\sum_{i=1}^{n} n_i.$$

If the transmitted signal sequences s_i, which is encoded by a balanced code, has the property:

$$\sum_{i=1}^{n} s_i = 0,$$

then

$$a = Ie^{j\theta_J} + \frac{1}{n}\sum_{i=1}^{n} n_i.$$

We can see that a is a good estimation of the jamming tone when n is large enough. Subtracting a from r_i, we have

$$r_i' = r_i - a = s_i - \frac{1}{n}\sum_{k=1}^{n} n_k + n_i = s_i + \tilde{n}_i \qquad i = 1, 2, ..., n$$

where

$$\tilde{n}_i = n_i - \frac{1}{n}\sum_{k=1}^{n} n_k.$$

Therefore the tone jamming component has been cancelled. Since cancellation involves more than one sample, after cancellation the signal is imbeded in noise $\tilde{n}_i$ which is correlated with others in a codeword. $\tilde{n}_i$ can be rewitten as

$$\tilde{n}_i = \tilde{x}_i + j\tilde{y}_i$$

and

$$\tilde{x}_i = x_i - \frac{1}{n}\sum_{k=1}^{n} x_k = (1-\frac{1}{n})x_i - \frac{1}{n}\sum_{k=1,k\neq i}^{n} x_k,$$

$$\tilde{y}_i = y_i - \frac{1}{n}\sum_{k=1}^{n} y_k = (1-\frac{1}{n})y_i - \frac{1}{n}\sum_{k=1,k\neq i}^{n} y_k.$$

$\tilde{x}_i$ and $\tilde{y}_i$ are linear combination of Gaussian variables, and thus are still Gaussian. It is easy to see that the mean of $\tilde{x}_i$ and $\tilde{y}_i$ are zero, and that they have the same variance $\tilde{\sigma}^2$, which is given by

$$\begin{aligned}\tilde{\sigma}^2 &= \left(1-\frac{1}{n}\right)^2\sigma^2 + \frac{n-1}{n^2}\sigma^2 \\ &= \frac{n-1}{n}\sigma^2. \end{aligned} \tag{3.2}$$

The correlation between adjacent $\tilde{n}_i$ can be described with the auto-correlation coefficient r and the cross-correlation coefficient λ. Because for arbitrary i and j, x_i and y_j are uncorrelated, $\tilde{x}_i$ and $\tilde{y}_j$ are also uncorrelated. So

$$\lambda = \frac{\overline{\tilde{x}_i\tilde{y}_{i+1}}}{\tilde{\sigma}^2} = 0,$$

and r is defined by

$$r = \frac{\overline{\tilde{x}_i\tilde{x}_{i+1}}}{\tilde{\sigma}^2} = \frac{\overline{\tilde{y}_i\tilde{y}_{i+1}}}{\tilde{\sigma}^2}.$$

Therefore after tone cancellation, the signal is perturbed by coloured Gaussian noise with a correlation coefficient r.

Pawula *et al.* derived the distribution of the phase angle between two vectors perturbed by correlated Gaussian noise[5]. By using their results we can derive the bit error probability of binary DPSK signalling in coloured Gaussian noise. The bit error probability of binary DPSK signalling is given by:

$$P_e = \frac{1}{2}P\left(\frac{\pi}{2} \le \psi \le \frac{3\pi}{2} | \Delta\Psi = 0\right) + \frac{1}{2}P\left(-\frac{\pi}{2} \le \psi \le \frac{\pi}{2} | \Delta\Psi = \pi\right)$$

where $\Delta\Psi$ is the phase change between two consecutive transmitted signals; ψ is the phase change between corresponding received signals, and we have assumed the two transmitted signals to be equalprobable. According to [5, Eq.(9)],

$$P\left(\frac{\pi}{2} \le \psi \le \frac{3\pi}{2} | \Delta\Psi = 0\right) = F_0\left(\frac{3\pi}{2}\right) - F_0\left(\frac{\pi}{2}\right),$$

$$P\left(-\frac{\pi}{2} \leq \psi \leq \frac{\pi}{2} | \Delta\Psi = \pi\right) = F_\pi\left(\frac{\pi}{2}\right) - F_\pi\left(-\frac{\pi}{2}\right)$$

where

$$F_{\Delta\Psi}(\psi) = \frac{1}{4\pi}\int_{-\pi/2}^{\pi/2}[\alpha(t) + \beta(t)]e^{-E(t)}dt \tag{3.3}$$

and

$$E(t) = \frac{U - V\sin t - W\cos(\Delta\Psi - \psi)\cos t}{1 - (r\cos\psi + \lambda\sin\psi)\cos t}, \tag{3.4}$$

$$\alpha(t) = \frac{W\sin(\Delta\Psi - \psi)}{U - V\sin t - W\cos(\Delta\Psi - \psi)\cos t}, \tag{3.5}$$

$$\beta(t) = \frac{r\sin\psi - \lambda\cos\psi}{1 - (r\cos\psi + \lambda\sin\psi)\cos t}. \tag{3.6}$$

In our case $U = W =$ signal to coloured noise ratio, $V = 0$, and $\lambda = 0$. Substituting these relations into Equations (3.4), (3.5), and (3.6), we have

$$E(t) = \frac{1 - \cos(\Delta\Psi - \psi)\cos t}{1 - r\cos\psi\cos t}U,$$

$$\alpha(t) = \frac{\sin(\Delta\Psi - \psi)}{1 - \cos(\Delta\Psi - \psi)\cos t},$$

$$\beta(t) = \frac{r\sin\psi}{1 - r\cos\psi\cos t}.$$

Note that when $E(t)$, $\alpha(t)$, $\beta(t)$ are constants, Equation (3.3) can be written as

$$F_{\Delta\Psi}(\psi) = \frac{1}{4}(\alpha + \beta)e^{-E}.$$

When $\Delta\Psi = 0$,

$$E(t) = \frac{1 - \cos\psi\cos t}{1 - r\cos\psi\cos t}U = U \quad \text{for} \quad \psi = \pm\frac{\pi}{2},$$

$$\alpha(t) = \frac{-\sin\psi}{1 - \cos\psi\cos t} = \mp 1 \quad \text{for} \quad \psi = \pm\frac{\pi}{2},$$

$$\beta(t) = \frac{r\sin\psi}{1 - r\cos\psi\cos t} = \pm r \quad \text{for} \quad \psi = \pm\frac{\pi}{2}.$$

Thus

$$F_0\left(\frac{\pi}{2}\right) = \frac{1}{4}(r - 1)e^{-U},$$

$$F_0\left(\frac{3\pi}{2}\right) = F_0\left(\frac{\pi}{2}\right) = \frac{1}{4}(1 - r)e^{-U}.$$

Therefore

$$P\left(\frac{\pi}{2} \le \psi \le \frac{3\pi}{2} | \Delta\Psi = 0\right) = \frac{1}{2}(1-r)e^{-U}. \tag{3.7}$$

When $\Delta\Psi = \pi$,

$$\begin{aligned} E(t) &= \frac{1+\cos\psi\cos t}{1-r\cos\psi\cos t}U = U \quad \text{for} \quad \psi = \pm\frac{\pi}{2}, \\ \alpha(t) &= \frac{\sin\psi}{1-r\cos\psi\cos t} = \pm 1 \quad \text{for} \quad \psi = \pm\frac{\pi}{2}, \\ \beta(t) &= \frac{r\sin\psi}{1-r\cos\psi\cos t} = \pm r \quad \text{for} \quad \psi = \pm\frac{\pi}{2}. \end{aligned}$$

Thus

$$\begin{aligned} F_\pi\left(\frac{\pi}{2}\right) &= \frac{1}{4}(1+r)e^{-U}, \\ F_\pi\left(-\frac{\pi}{2}\right) &= \frac{1}{4}(-1-r)e^{-U}. \end{aligned}$$

Therefore

$$P\left(-\frac{\pi}{2} \le \psi \le \frac{\pi}{2} | \Delta\Psi = \pi\right) = F_\pi\left(\frac{\pi}{2}\right) - F_\pi\left(-\frac{\pi}{2}\right) = \frac{1}{2}(1+r)e^{-U}. \tag{3.8}$$

And thus the bit error probability is

$$P_e = \frac{1}{4}(1-r)e^{-U} + \frac{1}{4}(1+r)e^{-U} = \frac{1}{2}e^{-U}$$

where U is signal to coloured noise ratio. From Equation (3.2) we know that the coloured noise variance is $(n-1)/n$ times of original thermal noise variance. Thus

$$U = \frac{n}{n-1}\frac{E_s}{N_0} = \frac{n}{n-1}\frac{r_b E_b}{N_0}$$

where r_b is the code rate of the balanced code. And finally, we have

$$P_e = \frac{1}{2}\exp\left(-\frac{n}{n-1}\frac{r_b E_b}{N_0}\right). \tag{3.9}$$

This expression does not contain the amplitude of jamming tone I, and other parameters related to jamming tone. Therefore the jamming tone has clearly been cancelled.

Two types of error probability are given in Equations (3.7) and (3.8), respectively. Obviously they are not the same for a fixed r. Especially when $|r|$ is large ($|r|$ is close to

one), the difference of two types of error probability can be quite large. However, it can be shown that in our case

$$\begin{aligned} r &= \frac{1}{\tilde{\sigma}^2}\overline{\left[(1-\frac{1}{n})x_i - \frac{1}{n}\sum_{k=1,k\neq i}^{n} x_k\right]\left[(1-\frac{1}{n})x_{i+1} - \frac{1}{n}\sum_{k=1,k\neq i+1}^{n} x_k\right]} \\ &= -\frac{1}{n-1} \end{aligned}$$

where we have used the following relations:

$$\overline{x_i^2} = \sigma^2 \qquad \text{and} \qquad \overline{x_i x_k} = 0 \qquad i \neq k.$$

So when n is not very small r is much less than one, and thus the two types of error probability are almost the same.

Recall that the bit error probability of binary DPSK system with thermal noise only is

$$P_{DPSK} = \frac{1}{2}\exp\left(-\frac{E_b}{N_0}\right). \tag{3.10}$$

Comparing with Equation (3.9), we can see that after cancellation the bit error probability has a form similar to that of binary DPSK system with noise only. But we can not compare the two expressions directly because Equation (3.9) gives the coded symbol error probability, and Equation (3.10) gives the bit error rate without coding. The bit error probability after decoding has to be evaluated with the specified balanced code in coloured Gaussian noise. This problem needs further investigation.

However, we can compare the coded symbol error rate given in Equation (3.9) with that of a binary DPSK system with a code of the same code rate r_b in thermal noise. The later is given by

$$P_e = \frac{1}{2}\exp\left(-\frac{E_s}{N_0}\right) = \frac{1}{2}\exp\left(-\frac{r_b E_b}{N_0}\right).$$

In this case, the coded symbol error rate of the DPSK system with balanced coding tone jamming cancellation is a little bit less than that of the DPSK system without cancellation, but employing a code with code rate r_b. But if n is very large, the difference is very small. The coded symbols error rate of the binary DPSK sytem and that of the DPSK sytem with balanced coding tone jamming cancellation for $n = 8$, and 16 are plotted in Fig. 3.4.

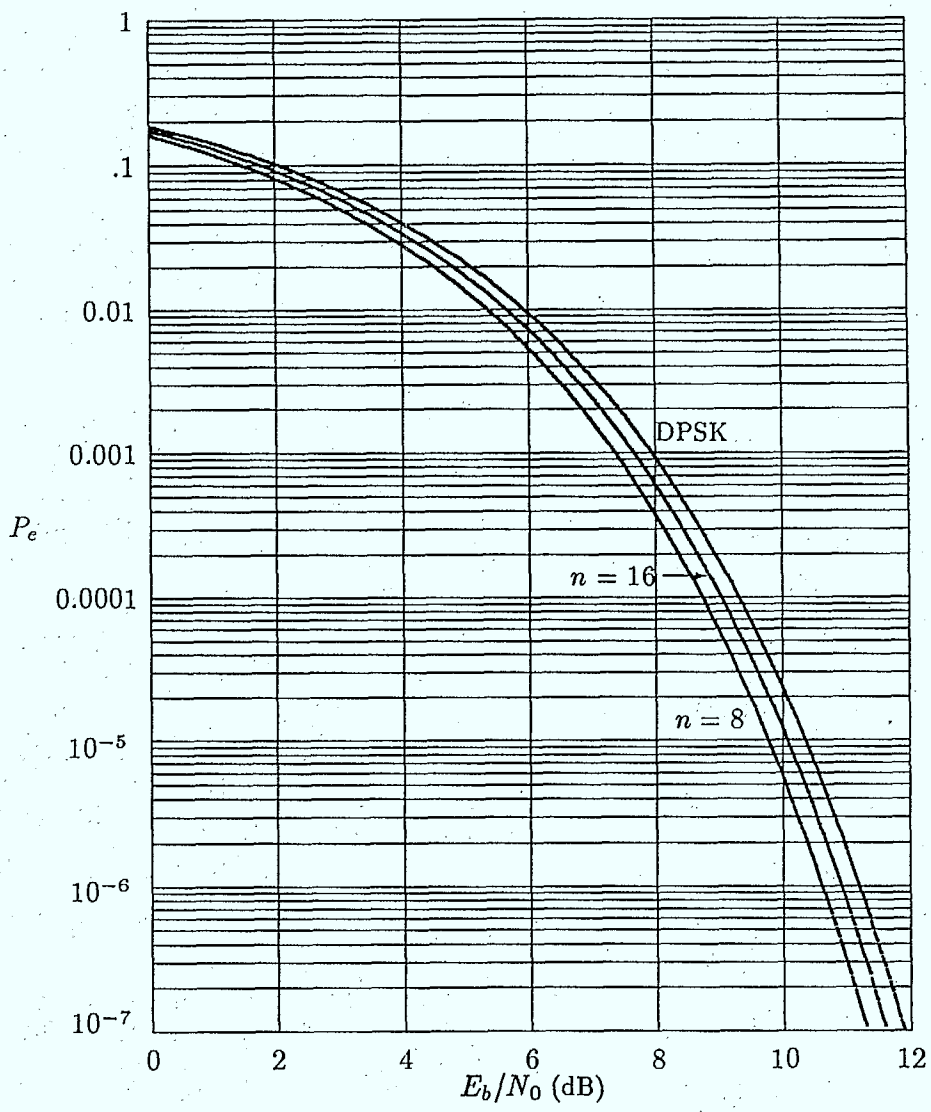


Figure 3.4: The coded symbol error rate of DPSK system with balanced coding tone jamming cancellation scheme and that of DPSK system employing a code with the same code rate in thermal noise.

3.3.2 Application of Balanced Coding in Scheme I

As mentioned early, the balanced code used in Scheme I should be transparent. The balanced codec operates on differentially encoded data with ambiguity on the data's absolute values.

Several papers have discussed balanced codes with high code rate, i.e. a code rate larger than 0.5 without considering the transparent property. D.E. Knuth [6] proposed a kind of balanced code which is very simple in encoding and decoding.

One trivial case of balanced transparent coding is simply adding the 1's complement of the information bits. This results in a rate $\frac{1}{2}$ balanced code with the transparent property. For example, if an information sequence is w, then the corresponding codeword is $w\bar{w}$, where $\bar{w}$ is 1's complment of w.

Recently several papers have discussed error-correcting balanced codes [7]. Usually we want to use a linear error correcting code to avoid a complex decoder. Thus we wish to have transparent linear balanced error-correcting code. However, no such codes exist because linear codes must have the all-zero codeword and linear transparent code must have the all-one codeword. These two codewords are not balanced codewords (in general, a balanced code is not necessarily a linear code; and there is only a one to one correspondence between information bits and codewords). But we can construct a code having similar properties. The code has the following structure:

1. (n, k) block code (n is even);
2. linear code;
3. transparent code;
4. all codewords are balanced codewords, except the all-one and all-zero codeword.

We call this code pseudo balanced code, and the code has the following properties:

1. The minimum distance $d_{\min} \geq \frac{n}{2}$; otherwise the sum of two balanced codewords would result in non-balanced codeword.
2. There are only three possible weights: 0, $\frac{n}{2}$, and n.

3. If w is a codeword, $\bar{w}$ is also a codeword; this is a property of a transparent code.

In communications, when data words happen to be the same as control words they are modified in certain ways to remove confusion. By using similar techniques, we can avoid encoding the all-zero and the all-one codewords at the transmission end. Then the code described above can be used in our tone jamming cancelling scheme as if it were a linear transparent balanced code. This code can help us eliminate the tone jamming and can correct random errors as well. Following are examples of two of such codes.

Example 3.1

The (7,4) Hamming code has weight enumerator:

$$A(z) = 1 + 7z^3 + 7z^4 + z^7.$$

This code can be modified to be a pseudo balanced code by adding a parity bit. This results in a (8,4) code with $d_{\min} = 4$. The code can correct 1 bit error and detect 2 bit errors at the same time.

Example 3.2

1st-order Reed-Muller codes are pseudo balanced codes. When $m = 4$ and $n = 16$, the generator matrix of the 1st-order Reed-Muller code is:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

This is a (16,5) code with $d_{\min} = 8$.

In the output of the differential demodulator in Scheme I, a one bit error may cause all of the following bits to be inverted. For a specific balanced codeword, if a bit error happens at the first bit or in a previous codeword the whole codeword will be inverted. Because the code is transparent, it can be correctly decoded. But if a bit error happens at a bit after the first bit in a codeword, all bits in the codeword after the error bit will be inverted, causing a burst of errors.

Code rate bound of error correcting balanced codes

For any (n, k, d) linear code there is a bound on minimum distance d:

$$n - k \geq d - 1.$$

When $d = \frac{n}{2}$ we have

$$n - k \geq \frac{n}{2} - 1$$

and therefore the code rate r is bounded by:

$$r = \frac{k}{n} \leq \frac{1}{2} + \frac{1}{n}.$$

Hence it is not possible for the code rate of a pseudo linear transparent balanced code to be much larger than 0.5.

According to the Plotkin bound,

$$k \leq n - 2d + 2 + \log_2 d.$$

When $d = \frac{n}{2}$ we have

$$k \leq 0 + 2 + \log_2 \frac{n}{2} = \log_2 n + 1$$

and hence

$$r = \frac{k}{n} \leq \frac{\log_2 n}{n} + \frac{1}{n}.$$

This upper bound is tighter than the first one.

Note that the code rate of the 1st-order Reed-Muller code is

$$r = \frac{m+1}{2^m} = \frac{\log_2 n + 1}{n}$$

where $n = 2^m$. So the code rate of the 1st-order Reed-Muller code achieves the Plotkin bound on code rate.

3.3.3 Application of Balanced Coding in Scheme II

The balanced decoder in Scheme II does not work on balanced encoded data, but on differentially decoded balanced encoded data. This is because the output of a differential

Information	Codewords	Diff. Decoded Codewords
0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0
1 0 0 0	1 1 0 1 0 0 0 1	0 1 1 1 0 0 1
0 1 0 0	0 1 1 0 1 0 0 1	1 0 1 1 1 0 1
1 1 0 0	1 0 1 1 1 0 0 0	1 1 0 0 1 0 0
0 0 1 0	1 1 1 0 0 1 0 0	0 0 1 0 1 1 0
1 0 1 0	0 0 1 1 0 1 0 1	0 1 0 1 1 1 1
0 1 1 0	1 0 0 0 1 1 0 1	1 0 0 1 0 1 1
1 1 1 0	0 1 0 1 1 1 0 0	1 1 1 0 0 1 0
0 0 0 1	1 0 1 0 0 0 1 1	1 1 1 0 0 1 0
1 0 0 1	0 1 1 1 0 0 1 0	1 0 0 1 0 1 1
0 1 0 1	1 1 0 0 1 0 1 0	0 1 0 1 1 1 1
1 1 0 1	0 0 0 1 1 0 1 1	0 0 1 0 1 1 0
0 0 1 1	0 1 0 0 0 1 1 1	1 1 0 0 1 0 0
1 0 1 1	1 0 0 1 0 1 1 0	1 0 1 1 1 0 1
0 1 1 1	0 0 1 0 1 1 1 0	0 1 1 1 0 0 1
1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0

Table 3.1: Extended Hamming (8,4) code and its differentially decoded codewords.

demodulator is differential decoded data. For a decoder, a mapping between the information bits and the differentially decoded balanced codeword is established. The mapping can be obtained easily by differentially decoding all balanced codewords.

The balanced coding can also provide some error correction capability in Scheme II. This can be implemented by choosing a balanced code whose codewords after differentially decoding have a large minimum Hamming distance.

Example 3.3

Consider the extended Hamming (8,4) balanced code discussed in *Example 3.1*. All codewords are listed in Table 3.1. This is a transparent code. We can only use the upper half codewords with the exception of the all-zero one. Thus in fact we use it as a (8,3) code. Note that the code is linear in the differential decoded domain, and has a minimum weight 3. Therefore the code can correct one error in a codeword at the output of differential demodulator.

3.4 Frequency Offset Problem

The analysis in the previous sections is based on the assumption that jamming tone hits the carrier tone exactly. However, in practice there is a frequency offset between carrier and jamming tone.

Frequency offset causes the tone jamming component in a baseband signal to be no longer constant. Instead the jamming has a sinusoidal form.

Suppose the jamming tone ω_J is:

$$\omega_J = \omega_C + \Delta\omega,$$

where ω_C is the carrier frequency, and $\Delta\theta$ is the frequency offset. Then the tone jamming component in the baseband signal is:

$$Ie^{j(\omega_J t+\theta_J)}e^{-j\omega_C t} = Ie^{j(\Delta\omega t+\theta_J)}.$$

Note that it is a sinusoid with frequency $\Delta\omega$, which is the frequency offset. The balanced code tone jamming cancelling scheme is based on the assumption that a tone jamming component is constant. Thus we need to analyze the cancelling performance when there is a frequency offset.

First we will study how much the phase change of consecutive jamming components is caused by frequency offset. Consider that the bandpass filter has a bandwidth B, so that

$$B = \frac{2}{T_b}$$

and

$$\Delta\omega_{MAX} = 2\pi\frac{B}{2} = \frac{2\pi}{T_b}.$$

The phase change between two consecutive jamming components is

$$\Delta\theta = \Delta\omega T_b$$

and the maximum phase change is

$$\Delta\theta_{MAX} = \Delta\omega_{MAX}T_b = 2\pi.$$

Thus the frequency offset can cause $\Delta\theta$ to be as large as 2π.

We then analyze the sensitivity of the cancelling performance to $\Delta\theta$. When there is a frequency offset $\Delta\omega$, the received baseband signal is

$$r_i = s_i + n_i + Ie^{j[\theta_J+(i-1)\Delta\theta]}.$$

The average over a codeword of length n is

$$\begin{aligned} a &= \frac{1}{n}\sum_{i=1}^{n} r_i \\ &= \frac{1}{n}\sum_{i=1}^{n} n_i + \frac{I}{n}\sum_{i=1}^{n} e^{j[\theta_J+(i-1)\Delta\theta]} \\ &= \sum_{i=1}^{n} n_i + CIe^{j\theta_J} \end{aligned}$$

where

$$\begin{aligned} C &= \frac{1}{n}\sum_{i=1}^{n} e^{(i-1)\Delta\theta} \\ &= \frac{\sin\left(\frac{n\Delta\theta}{2}\right)}{n\sin\left(\frac{\Delta\theta}{2}\right)} e^{j(n-1)\frac{\Delta\theta}{2}}. \end{aligned}$$

And after cancelling we have

$$r_i' = r_i - a = s_i + n_i - \tilde{n} + (1-C)Ie^{j\theta_J}.$$

$|1-C|$ can be used as an indication of the cancelling effects, which is a function of $\Delta\theta$ and the codeword length n. $|1-C|$ versus $\Delta\theta$ with n as parameter is plotted in Fig. 3.5.

It is shown that the cancelling region (where $|1-C|$ is near zero) is relatively wider for small n. We also see that when $\Delta\theta > 0.2\pi$, $|1-C|$ is almost unity, indicating no cancelling at all. When $\Delta\theta < 0.05\pi$, about 3 dB cancellation may be obtained for $n \leq 10$. Thus, in principle this tone jamming cancelling scheme can work well when the frequency offset is less than 2.5% of the total bandwidth in either direction.

3.5 Tone Jamming Cancelling by Adaptive Notch Filter

As mentioned in section 3.4, when there is a frequency offset of the tone jamming to carrier tone the jamming tone component in the baseband signal is a sinusoid. Its frequency is the

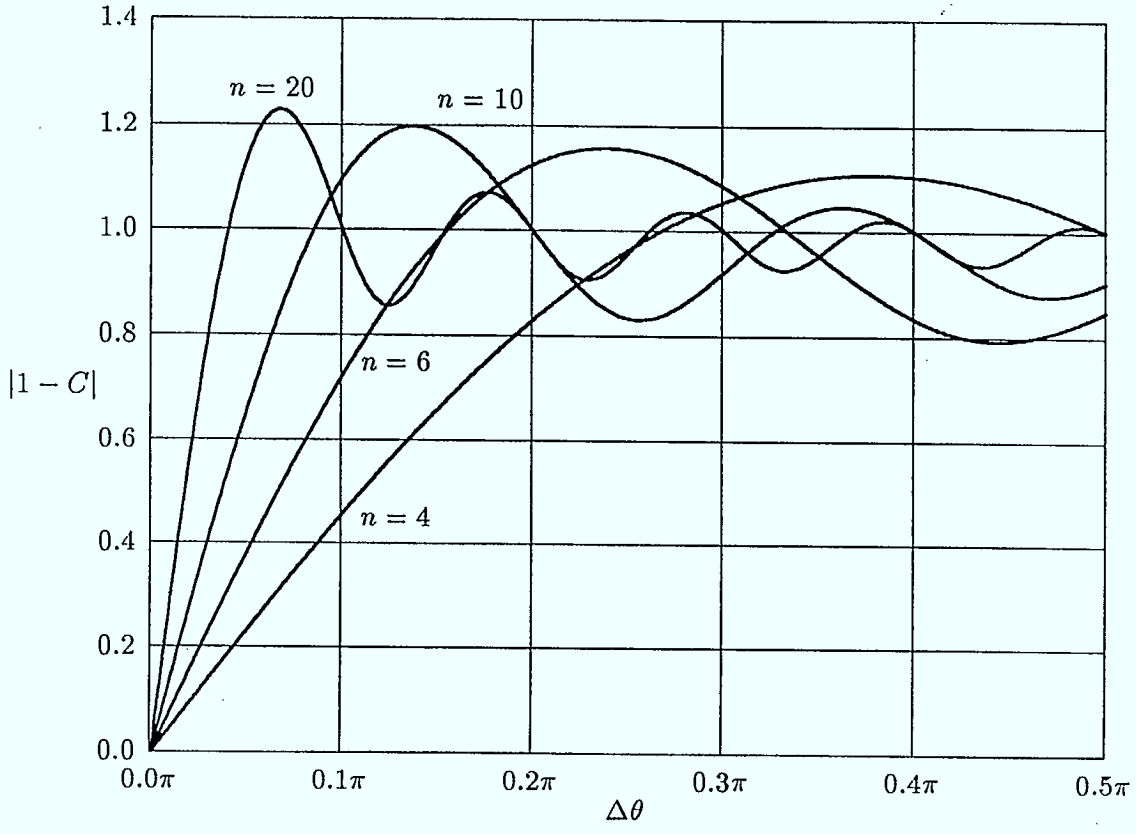


Figure 3.5: $|1 - C|$ versus $\Delta\theta$ with n as parameter.

difference between the jamming tone and the carrier frequency. Thus the received signal at baseband can be modeled as:

$$r_i = s_i + n_i + Ie^{j(\theta_J + (i-1)\Delta\theta)} \qquad i = 1, 2, \cdots$$

where $\Delta\theta = (\omega_J - \omega_C)T_b$. $\Delta\theta$ can be as large as 2π. Because the s_i's are uncorrelated with each other and n_i is AWGN, the spectrum of $s_i + n_i$ is much wider than that of the tone jamming component. Hence $s_i + n_i$ and the jamming component can be separated by signal processing techniques. One method is to use an adaptive notch filter based on an estimation of $\Delta\theta$.

Because there are at least a few tens of bits in a hop, the amount of data which can be used in estimation of $\Delta\theta$ is large. The simple FFT method can provide a good quality estimation[9].

A notch filter which has a zero on the unit circle with an angle $\Delta\theta$ can cancel the jamming tone clearly if the estimation of $\Delta\theta$ is accurate. There are two problems associated with the use of a notch filter. One is sensitivity to estimation errors, the other one is the distortion of transmitted signals. To reduce the sensitivity of estimation errors the stopband of the notch filter should be wider. However, to reduce the distortion of useful signals the stopband should be as narrow as possible. The basic solution to both problem is to improve the estimation of $\Delta\theta$.

The performance of the notch filter cancelling scheme is illustrated by simulations. The notch filter used in the simulations is a simple one-zero one-pole IIR filter. The zero is on the unit circle with an angle of $\Delta\theta$, and the pole is at $re^{j\Delta\theta}$ and $r < 1$, i.e. the pole is within the unit circle with the same angle. The transfer function of the filter is

$$H(z) = \frac{1 - e^{j\Delta\theta}z^{-1}}{1 - re^{j\Delta\theta}z^{-1}}.$$

The system diagram is shown in Fig. 3.6. The bit error rate versus E_b/N_0 with r as a parameter, assuming no error in the estimation of $\Delta\theta$, is shown in Fig. 3.7. Because there are no estimation errors, the tone jamming is cancelled completely. Thus BER is not related to E_b/N_J. These curves can be viewed as the optimum performance that can be achieved by the first order IIR filter. For comparison, the BER without a notch filter under

tone jamming with E_b/N_J=0 dB and without jamming are also plotted in Fig. 3.7. We can see that when $r = 0.9$ there is about 5 dB performance loss for BER around 10^{-5}. When $r = 0.99$ the performance loss is very small. And the performance loss is very large for $r < 0.8$ because of large signal distortion.

The bit error rate versus E_b/N_J, with r as parameter and with estimation error 0.05 rad and 0.025 rad, are plotted in Fig. 3.8 and Fig. 3.9 respectively. E_b/N_0 is 10.34 dB (corresponding to BER = 10^{-5} without jamming). These figures show that when the signal is not much stronger than the jamming ($E_b/N_J < 10$ dB), there is more than one order of improvement over BER by using a notch filter, i.e. BER drops from 10^{-1} to 10^{-2} or 10^{-3}. But when the jamming is very much stronger than the signal, the notch filter can only cancel a small part of tone jamming, and the cancellation does not work because of errors in the estimation of $\Delta\theta$. We can see that the smaller the estimation error, the stronger the jamming that is needed to defeat notch filter cancellation. When a signal is much stronger than the jamming, the filter distortion to useful signals is dominant, and therefore the BER with notch filter is higher than BER without notch filter.

It should be noted that the notch filter used in simulations is a first order IIR filter, a very simple digital filter. Filters with higher order may have better performance. This needs further investigation.

According to our simulation results, the following conclusions can be made: (1) the notch filter can cancel tone jamming when jamming is not very strong; (2)it would be beneficial to have channel state information of the jamming condition to switch on or off the notch filter according to whether the hop is jammed or not (such side information need not be perfect and may be obtained by using FFT); (3) random error correcting codes may be needed to improve performance under jamming (it is also interesting to note that notch filter cancellation makes it possible for error correcting codes to be effective, because cancellation brings the BER from 10^{-1} to 10^{-3} and error correcting codes can make the BER drop to 10^{-5} with a reasonable code rate).

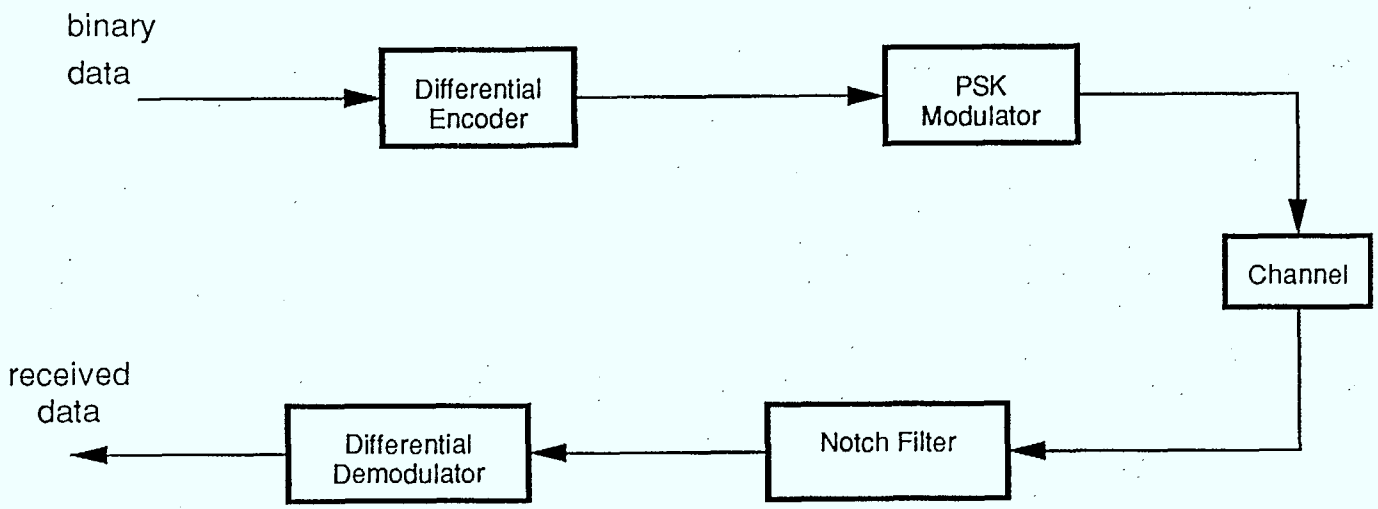


Figure 3.6: Notch filter tone jamming cancellation scheme.

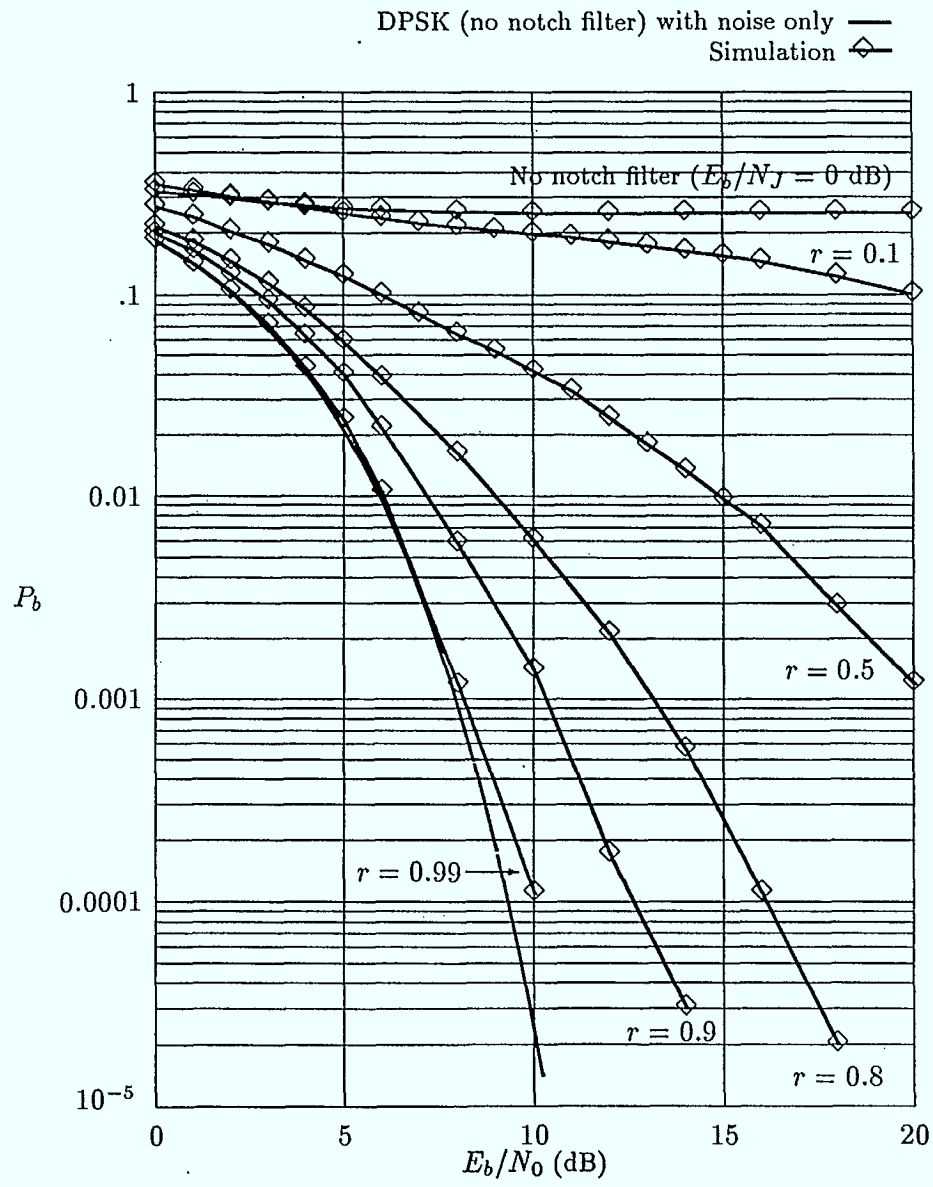


Figure 3.7: BER of binary DPSK system with notch filter tone jamming cancellation scheme versus E_b/N_0 with r as parameter without estimation errors. $\Delta\theta = \pi/3$.

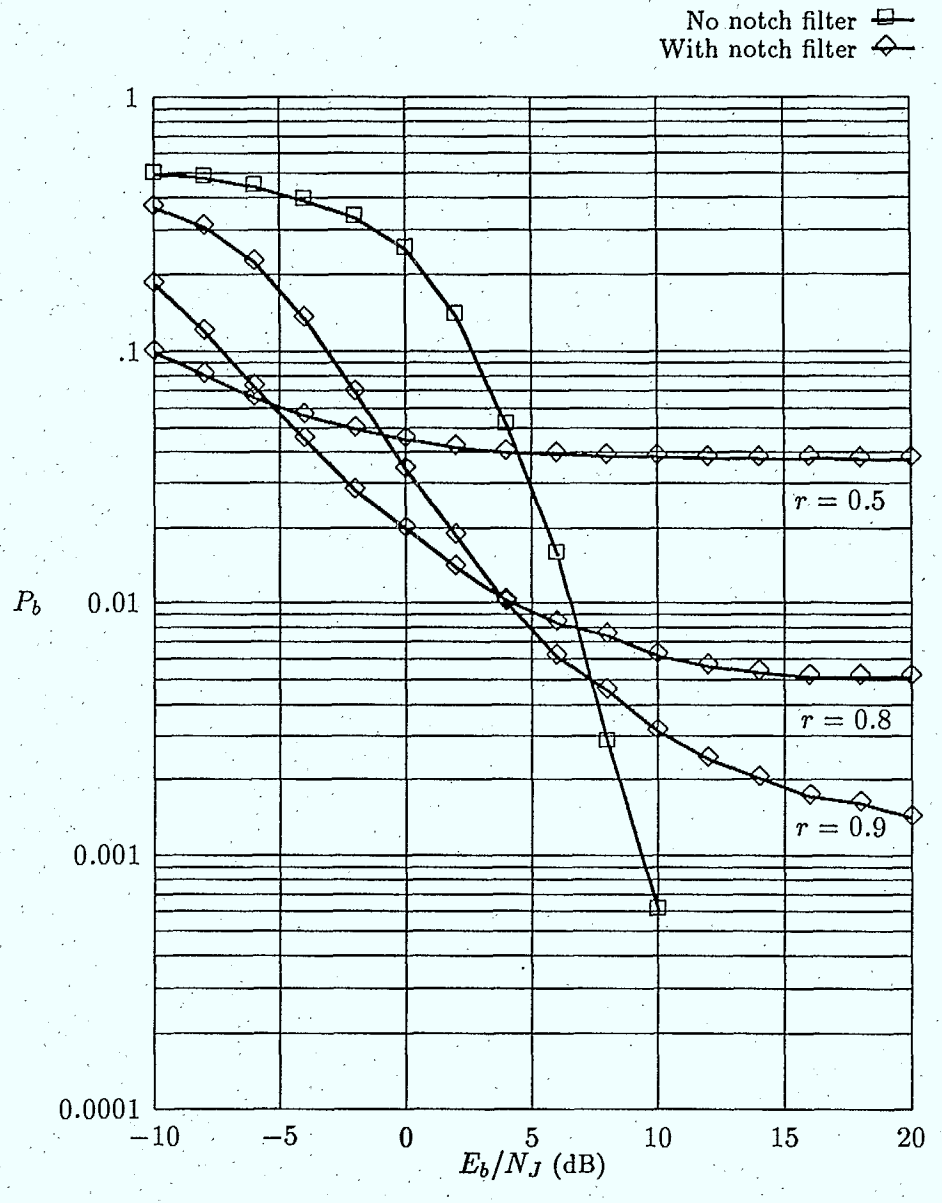


Figure 3.8: BER of binary DPSK system with notch filter tone jamming cancellation scheme versus E_b/N_J and with r as parameter with estimation error 0.05 rad. $\Delta\theta = 1.0$ rad, and filter notch at 1.05 rad.

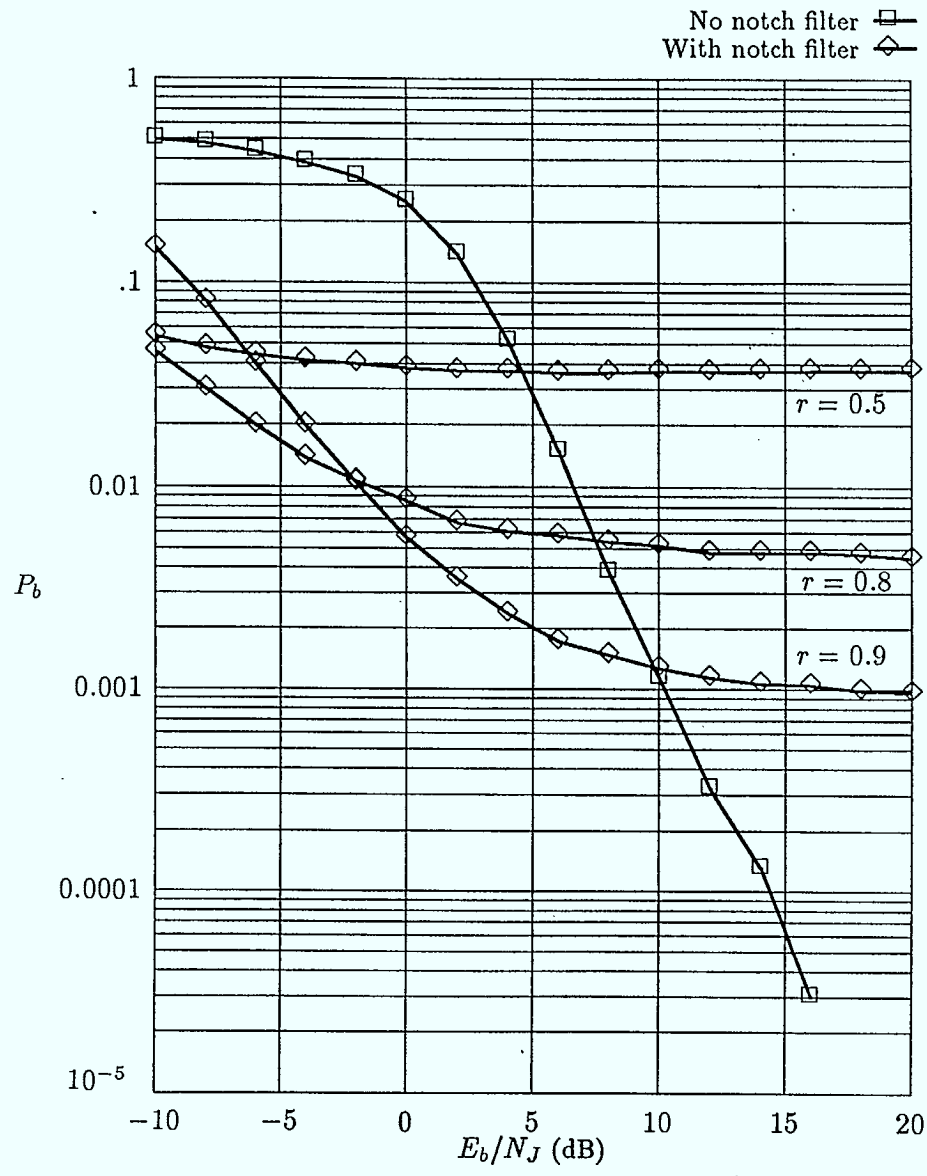


Figure 3.9: BER of binary DPSK system with notch filter tone jamming cancellation scheme versus E_b/N_J and with r as parameter with estimation error 0.025 rad. $\Delta\theta = 1.0$ rad, and fiter notch at 1.025 rad.

3.6 Conclusion

The balanced coding based tone jamming cancellation scheme can work well when the tone jamming frequency offset is very small. This scheme has little distortion on useful signals, and the corresponding performance loss is less than 2 dB. Balanced coding can be combined with error correction coding.

The notch filter tone jamming cancellation scheme can cancel tone jamming with arbitrary frequency offset in the passband provided there is accurate estimation of the frequency offset. The notch filter has some distortion to useful signals. Therefore, it is better to switch it off for hops without jamming. Notch filter cancellation can also make error correction coding effective.

Chapter 4

Interleaving

4.1 Introduction

Interleaving is the process of reordering a sequence of symbols in a one-to-one deterministic manner so that any two symbols within a given separation in the input sequence are separated by at least some minimum number of symbols in the output sequence. The concept finds application in burst and fading channels where bursts of channel noise occur and the action of the interleaver/deinterleaver is intended to randomize the resulting error patterns, hopefully resulting in improved effectiveness of the codi ng. It is also used on intentional interference (jamming) channels and a convenient assumption in the analysis of systems on all these channels is that the use of interleaving renders the channels memoryless. In practice the situation is not so simple and the analysis of coded systems employing finite interleavers can be difficult. For example the use of a periodic interleaver on a jamming channel, where the jammer hits only certain symbols in each period, might have a disastrous effect on communication performance. In such applications one might have the choice of using a random error correcting code with interleaving or a burst error correcting code without interleaving. A burst error correcting code will typically have greater efficiency than a random error correcting code for a given burst length, but might have limited ability ability to correct additional random errors. The behavior of either class of code when the burst length exceeds either the burst correcting capability or interleaver/code design capability is sometimes difficult to assess.

The amount of work on the design of interleavers appears to be limited and much

of the interesting work is not in the open literature. The scope of this chapter is narrow being limited to presenting the basic principles and techniques for designing interleavers. The more complicated problem of analyzing the performance of coded systems using a particular finite interleaver will be considered in later work. The section 4.2 includes some elementary properties and definitions of interleavers. The following sections consider, respectively, descriptions of block, convolutional and the more recent helical interleavers. The material for this chapter is drawn from the references listed and a brief review of their contents might be useful. The book of Clark and Cain [10] contains a discussion on interleavers and systems using them, although somewhat restricted in both the analysis and variety of interleavers it considers. The paper by Ramsey [11] discusses fundamental properties of interleavers in terms of delay and storage requirements, as well as implementations of some optimal convolutional interleavers. Forney [12] suggests a particular type of convolutional interleaver, a special case of a more general class considered in [13]. Richer [14] discusses a particular type of pseudo-random block interleaver. The report [15] is quite far ranging, introducing the notion of helical interleavers and analyzing and comparing their performance to standard interleavers. In addition it describes a block pseudo-random interleaver attributed to McEliece as well as many other aspects of the implementation and analysis of interleavers. A method of randomizing the rows of a block interleaver is discussed in [17] and in [18] an algorithm for the design of a pseudo-random interleaver with a certain distance property is considered.

4.2 Properties of Interleavers

A few properties that pertain to all interleavers are discussed, drawn mainly from the work of Ramsey [11]. Since the terminology is not quite standard we introduce our own. We define a (b, n) interleaver as one that reorders an input sequence so that any pair of n contiguous symbols of the input is separated by at least b symbols at the output (i.e. at least $b-1$ symbols between the pair). Equivalently, a (b, n) interleaver is such that any contiguous sequence of b symbols at the output contains no pair of symbols that lie in a contiguous block of n symbols at the input. Notice that, since the deinterleaver restores the order of

the original sequence with some delay, the deinterleaver of a (b,n) interleaver is an (n,b) interleaver. To express the notions more formally, if the input sequence is $...a_{i-1}a_ia_{i+1}...$ and the reordered sequence is $...a_{z_{i-1}}a_{z_i}a_{z_{i+1}}...$ (note that symbol a_{z_j} appeared in position j in the output and position z_j at the input and $j \geq z_j$) then a (b,n) interleaver has the property that

$$|z_i - z_j| \geq n \qquad \text{whenever} \qquad |i-j| < b.$$

The terminology reflects the situation that if n is chosen as the block length of a code and b as the maximum noise burst length on the channel, then for an isolated burst, no codeword at the input to the decoder is hit more than once by the burst. Typically then b might be chosen on the order of the maximum burst length, in symbols, expected on the channel and n as the block length or constraint length of the code used.

To consider the delay of an interleaver, we first assume without loss of generality that $\min_i(i-z_i) = 0$ where, as noted, since the interleaver is assumed to be realizable, $i \geq z_i$. Let the maximum delay experienced by a symbol through the interleaver be $d = \max_j(j-z_j)$. Since the output sequence of the deinterleaver is a delayed version of the input sequence to the interleaver, say by D symbols, the sum of the delays of a symbol in the interleaver and the deinterleaver is D. Since it is assumed there is at least one symbol with a delay of zero through the interleaver, and similarly through the deinterleaver, the overall delay D is at most d. Since there is at least one symbol with a delay of d in the interleaver the overall delay is at least d and so the delay in the interleaver is the same as the delay in the deinterleaver which is the same as the overall delay d. If the overall delay is d then when symbol a_i appears at the output of the deinterleaver, symbols $a_{i+1}, a_{i+2}, ..., a_{i+d}$ must all have entered the interleaver. It follows that the combined storage of the interleaver/deinterleaver is at least d. It is a simple matter to show that the maximum delay of a (b,n) interleaver is at least $(b-1)(n-1)$ and to see this consider the location of the n contiguous symbols $a_ia_{i+1}...a_{i+n-1}$ at the output of the interleaver. By definition there must be at least $b-1$ symbols between each of these symbols at the output. The span of these symbols in the output sequence is thus at least $(n-1)b$. In the "worst" case the last symbol in the input sequence is also the last symbol in this output sequence and the maximum delay must

therefore be at least $(n-1)b-(n-1)=(n-1)(b-1)$. It is also noted in [13] that the average delay of a (b,n) interleaver is at least $(n-1)(b-1)/2$. The period of a periodic interleaver is the minimum sequence length for which the entire pattern of delays repeats. It is noted, again in [13], that the period of a (b,n) interleaver is at least $\min(b,n)$. The depth [15] of an interleaver is defined as one less than the shortest burst length which can hit any codeword twice, where it is assumed that the interleaver input is divided into codewords.

These delay arguments are slightly refined in [11]. An interleaver is defined to be uniform if there is no set of $(b+1)$ contiguous symbols in the output sequence for which every pair of symbols is separated by at least n symbols in the input sequence. If the interleaver is not uniform it is referred to as nonuniform. It is shown in [11] that the encoding delay of a uniform (b,n) interleaver is at least $(b-1)(n+1)$ and for a nonuniform is at least $b(n+1)$. It is also shown that the deinterleaver for any (b,n) interleaver that achieves the minimum possible encoding delay is an (n,b) interleaver which also achieves the minimum possible encoding delay.

An interleaver/deinterleaver pair is called optimum if it achieves both the minimum encoding delay and the minimum combined storage requirements. Optimum interleavers are given in [11] for all pairs n,b that satisfy certain relative primeness conditions. The realizations of these interleavers are in terms of one long shift register. The work of Forney ([12],[13]) realizes the interleaving by means of shorter registers and commutators. These will be considered in the section 4.4 on convolutional interleavers.

4.3 Block Interleavers

The usual (b,n) block interleaver consists of an array of n rows and b columns. The symbols are read into the array by columns and out by rows and it is assumed the upper left symbol is the first read in and the first read out. Any burst of fewer than b errors on the channel results in errors separated by at least n symbols at the output of the deinterleaver. Labeling the rows of the array from 0 to $n-1$ and the columns from 0 to $b-1$, then a symbol in position (j,k) receives a delay of $nb+(b-1)j-(n-1)k$ at the interleaver and $nb+(n-1)k-(b-1)j$ at the deinterleaver. The characters at the top left and lower right each receive a delay of

nb symbols at both the interleaver and deinterleaver, assuming the array is filled before any symbols are read out. The minimum delay of the interleaver occurs for the symbol in the upper right hand corner and is $b + n - 1$. At the deinterleaver this element experiences a delay of $2nb - b - n + 1$ for a total delay of $2nb$. From previous considerations it is possible to arrange matters so that the delay at both the transmitter and receiver is reduced by $b + n - 1$ to give a total delay of $2(n-1)(b-1)$, twice the minimum possible established earlier. It is easy to see that it is not necessary to wait until the array is completely filled before beginning the read out process and a simple strategy achieves the reduction in the delay.

The obvious implementation of this block interleaver might use two $n \times b$ RAM's and ping-pong back and forth. In [15] the following one-RAM implementation is given. View the addresses of the RAM as the integers modulo $nb - 1$, running sequentially down the columns and moving from the bottom of one column to the top of the next. This is augmented by the special symbol ∞ corresponding to the lower right location. At any location the symbol is first read and then replaced by an incoming symbol. The procedure works by passes. On the zeroth pass, symbols are read in only with the address sequence

$$0, 1, 2, \cdots, nb - 2, \infty.$$

On the first pass, the above address sequence is multiplied by n, all addresses taken modulo $nb - 1$. On the kth pass the address sequence is

$$0, n^k, 2n^k, \cdots, (nb-2)n^k, \infty$$

modulo $nb - 1$. Clearly the period of the address sequence generator is the multiplicative order of n modulo $nb-1$. Similarly, the address sequence on the kth pass for the deinterleaver is

$$0, b^k, 2b^k, ..., (nb-2)b^k, \infty$$

and note that, since $nb \equiv 1$ modulo $(nb-1)$, the multiplicative orders of n and b modulo $nb-1$ are the same. Notice that the storage capacity of the interleaver and the deinterleaver are the same, nb, approximately one half of the total delay $d = 2(n-1)(b-1)$, and hence

close to optimum. It is not clear to the author how to achieve the optimum implementation, assuming it exists for this structure.

For some applications a periodic interleaver is unsuitable due to either intentional or unintentional periodicities in the interference. For such applications pseudo-random interleavers will be of interest. A scheme due to McEliece, reported in [15] is described. A RAM of size 2^{m-1} is used. The address sequence is taken from the lower $(m-1)$ bits of an m bit maximum length linear feedback shift register, except for the state labeled β where the $(m-1)$ low order bits are zero and the mth bit is one. The period of the address generator is clearly $2^m - 1$ and the delays encountered by bits going through the interleaver are uniformly distributed among the integers $1, 2, ..., 2^m - 1$. To see this recall that the state of the register at time t may be assumed to be α^t where α is a root of the primitive polynomial of the shift register. If $\alpha^t = \beta$ then the character which enters the RAM at time t leaves the RAM at time τ where

$$\alpha^\tau = \alpha^t + \beta$$

for a delay of $\delta = \tau - t$. Consequently

$$\alpha^\delta = 1 + \beta\alpha^{-t}$$

or

$$\alpha^t = \frac{\beta}{1+\alpha^\delta}$$

As δ runs through $0 < \delta < 2^m - 1$, the values of α^t obtained are distinct, indicating that once in every period (length $2^m - 1$) one symbol experiences a delay of δ. Notice that every address of the RAM is generated twice every period of the shift register, except the all zero address, corresponding to β which is generated only once. Characters in this address experience the full delay of $2^m - 1$. To maintain the full delay of transmitter and receiver at $2^m - 1$ either the interleaver or deinterleaver must be modified so that the symbol destined for this zero address bypasses the RAM altogether - a delay of zero.

Notice that the depth of this interleaver is one, i.e. it is possible for a burst of length two on the channel to hit the same codeword, and all delays are realized.

A similar type of pseudo-random interleaver has been proposed ([10],[14]). Here a sequence of L symbols is read into a RAM sequentially and then read out according to

some permutation. It is suggested in [14] that the permutation used correspond to a linear congruence:

$$A_{n+1} = aA_n + c \qquad \text{mod} \quad L$$

which generates a sequence of integers from 0 to $L-1$. To obtain all the integers from 0 to $L-1$ i.e. obtain a maximum length sequence, a and c must be chosen as follows [16]:

i) $a, c < L$

ii) $(c, L) = 1$

iii) if $p|L$ then $p|(a-1)$

iv) if $4|L$ then $p|(a-1)$

Such an approach is convenient for implementation. For example, in an intentional interference environment it will be desirable to change the permutation frequently, necessitating storage of the permutations. The above approach requires only the storage of pairs (a, c) representing the complete permutation.

There have been two more recent contributions to pseudo-random interleaving. In [17] it is noted that in a periodic (b, n) block interleaver, for example, a burst of length exceeding b will manifest itself as a burst in the deinterleaver output. If such occasional long channel bursts are anticipated and a burst error-correcting code of sufficient strength is used, then, as noted previously, good system performance is expected. If a random error correcting code is used, however, then such bursts may lead to degraded performance. In an attempt to make such bursts at the output of the deinterleaver appear more random, it is suggested in [17] that the rows of a block interleaver be read out in a random manner rather periodic. There, a parameter a is chosen and q, r defined by $n = aq + r, 0 \leq r < a, q > 0$. If t_{ij} is the number of rows transmitted between the transmission of row i and j and $t = \min_{|i-j| \leq a-1} t_{ij}$, then t is the minimum number of rows transmitted between any two rows that are within a of each other in the array. Alternatively if each row of the (b, n) block interleaver is treated as one symbol, then what is required is a (t, a) interleaver operating on the rows. This is actually a concatenated interleaver, discussed briefly later in section 4.5 on

helical interleaving. In [17] it is shown that t cannot be greater than $q-1$ and an algorithm is given that achieves this upper bound.

Recall that for a periodic block interleaver, a certain minimum separation at the interleaver output of any pair of symbols within a given span at the input can be assured. For pseudo-random interleavers, this is not the case and, for example, for the implementation of McEliece [15] the depth is one, as noted and the delays are uniformly distributed. In [18] a RAM read out/read in implementation of a pseudo-random interleaver is considered and an algorithm is given that is both pseudo-random and attempts to guarantee a given depth. The approach is interesting but is not pursued here.

4.4 Convolutional Interleavers

In contrast to the block interleavers of the previous section, convolutional interleavers utilize shift registers or delay lines. Data is fed through on a continuous basis with various stages being tapped sequentially. It remains only to arrange the taps and the commutator sequence to ensure all data symbols are transmitted with an appropriate delay. Only the interleavers of Forney ([12],[13]) will be described here. The work of Ramsey [11] appears similar in spirit although no formal equivalence between the two approaches has been established to our knowledge. Essentially Ramsey utilizes one long shift register with certain stages tapped at certain times while Forney considers a horizontal bank of shift registers of varying lengths with the output of each register transmitted sequentially on the channel.

A simple version of Forney's scheme will be considered [12]. Symbols are first divided into blocks of length P and it is convenient to view this as a serial-to-parallel conversion. The ith symbol of each block is fed into a shift register of length iD, $i = 0, 1, ...P-1$. The outputs of the P shift registers are sampled sequentially (parallel-to-serial conversion) and transmitted on the channel. The structure of the deinterleaver is similar except that the ith element of each block enters a shift register of length $(P-1-i)D', i = 0, 1, ...P-1$. Each symbol receives a total delay of $(P-1)D$ time units (each unit corresponding to P symbols) or $(P-1)DP$ symbols and the total storage requirements of interleaver and deinterleaver is $(P-1)DP$. In our previous terminology we would identify n with DP and b with P and

refer to this as a (b, n) interleaver.

A more general version of this interleaver that will have implementation advantages in some situations is described in [13]. Retaining the same notation as above we describe a (P, D, m) modular interleaver as follows (here m is a new parameter with the property that $1 \leq m \leq P-1, (P, m) = 1$ and $(P, mD+1) = 1$). Let $r_i \equiv im \bmod P$, $i = 0, 1, ..., P-1$ and note that, by assumption, m is a unit in the set of integers mod P and so $\{r_0, r_1, \cdots, r_{P-1}\} = \{0, 1, ..., P-1\}$. The interleaver thus consists of a serial-to-parallel converter, converting to parallel blocks of size P, and the ith symbol of each block is fed into a shift register of length $d_i = r_i D, i = 0, 1, ..., P-1$. The outputs of the shift registers are then parallel to serial converted for transmission on the channel.

Note that this modular interleaver is a permuted version of the first one described i.e. all the shift registers of the varying lengths occur but in a different order. The previous one would then be an $(P, D, 1)$ interleaver. Thus, as before, the period of the interleaver is P, the maximum delay is $(P-1)DP$ and the average delay is $(P-1)DP/2$. If one defines m^{-1}, k and k^{-1} by the equations $mm^{-1} \equiv 1 \bmod P$, $k(mD+1) \equiv -m \bmod P$ and $kk^{-1} \equiv 1 \bmod P$ respectively (note that by assumption, k^{-1} exists), then it can be shown that the inverse of a (P, D, m) interleaver is a (P, D, k) interleaver, and if $k = m$ (i.e. $mD \equiv -2 \bmod P$ then the (P, D, m) interleaver is its own inverse.

It is sometimes possible to realize modular interleavers as a cascade of two interleavers. Suppose $P = P_1 P_2$ and $(P_1, P_2) = 1$. Then the first interleaver uses delays $iP_1 D, 0 \leq i \leq P_2 - 1$ and a second uses delays $jD, 0 \leq j \leq P_1 - 1$ ensuring that in a P symbol interval a delay of kD is experienced by one symbol, $k = 0, 1, ...P-1$. If $(P_1, P_2) > 1$ then let b be the smallest integer such that the least common multiple of P_1 and bP_2 is P. Then the first interleaver uses delays $iP_1 D$ but operates on b symbol groups i.e. the b symbols in the ith group of each bP_2 symbol block experiences a delay of $iP_1 D, i = 0, 1, ..., P_2-1$. The period of the second interleaver is still P_1 with delays of $jD, 0 \leq j \leq P_1 - 1$.

It has been observed that the periodic interleavers might not be suitable for jamming channels where a periodic jammer would be very effective. To alleviate this problem it is suggested in [13] that the ith delay path might be modified to have delay $iD + j_i$, $i = 0, 1, ..., P-1$, where the j_i are small integers.

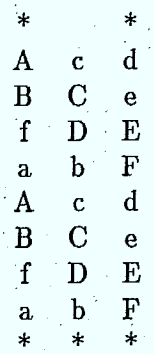


Figure 4.1: Helical interleaver, $n = 4$.

It is noted that it is only necessary for the deinterleavers to establish synchronism modulo P for these modular interleavers, an improvement from the block interleavers where synchronism modulo nb is required, usually necessitating the insertion of special synchronization sequences.

4.5 Helical Interleavers

Helical interleavers are very similar to block interleavers with successive columns slipped by one symbol. Figure 4.1, taken from [15], illustrates the technique.

Symbols are read in by columns of length four (say codewords) and read out by rows. In general the columns will be of length n and the number of columns will be $n-1$. Thus there is not the flexibility of choosing the "width" of the helical interleaver.

The capital and lower case letters represent the same physical location of memory and it is easily verified that such an assignment works as claimed. Figure 4.2 demonstrates the order of the symbols into and out of the interleaver/deinterleaver. It is clear that the interleaver of Figure 4.1 has a period of 12, a minimum delay of 3, a maximum delay of 9 and can be implemented with a RAM of size 6.

To extrapolate from Figure 4.1, consider the length 6 interleaver of Figure 4.3.

The inherent symmetries of the helical interleaver are clear and the RAM size required for the helical interleaver of length n is $n(n-1)/2$, the minimum delay is $n-1$, the

	time →											
into interleaver:	A	B	f	a	C	D	b	c	E	F	d	e
into channel:	a	b	F	A	c	d	B	C	e	f	D	E
out of deinterleaver:	A	B	f	a	C	D	b	c	E	F	d	e

Figure 4.2: Input/output of helical interleaver, $n = 4$, period 12 and overall delay 12.

*	*	*	*	*
A	d	e	f	O
B	D	g	h	i
C	E	G	j	k
l	F	H	J	m
n	o	I	K	M
a	b	c	L	N
A	d	e	f	O
B	D	g	h	i
C	E	G	j	k
l	F	H	J	m
*	*	*	*	*

Figure 4.3: Helical interleaver, $n = 6$.

*	*	*
A	B	C
A	A	D
B	A	A
C	D	A
A	B	C
A	A	D
B	A	A
C	D	A
*	*	*

Figure 4.4: Helical interleaver memory read-in sequence, $n = 4$.

*	*	*
A		C
↓	↙	↑
A	→	B
↓	↙	↑
A	→	D
↓	↙	↑
A	→	C
*	*	*

Figure 4.5: Helical interleaver memory address generation, $n = 4$.

maximum delay is $(n-1)^2$ and the total interleaving delay is $n(n-1)$. In our previous terminology this would be referred to as an $(n-1, n)$ interleaver.

The total interleaving delay of a helical interleaver can in fact be reduced to $(n-1)(n-2) + 2$, close to the minimum maximum delay established earlier of $(n-1)(n-2)$. An example [15] of the RAM organization and address assignment that achieves this delay, and also reduces the memory requirement to 4 (from 6) for the case of $n = 4$, are given in Figures 4.4 and 4.5. This address assignment technique is easy to extend for the case of n even and appears more difficult for n odd.

The advantages of helical interleavers appear to lie in the reduced synchronization requirements, compared to block interleavers, and their relatively simple memory implementations.

Figure 4.6: The effect of deep staggering for $n = 4, i = 3$.

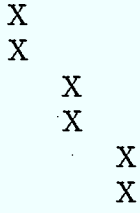


Figure 4.7: The effect of shallow staggering for $n = 4, i = 2$.

The concatenation of helical interleavers with block interleavers is also considered in [15]. If a block interleaver precedes a helical interleaver, the combination is referred to as deep staggered interleaving. If a code of length n symbols is used and an (i, n) block interleaver is used, then the depth of the deep staggered interleaver is $(n-1)i$. At the output of the block interleaver, each block of i symbols is treated as a single character to the helical interleaver. Thus the memory and total delay of the two helical interleavers is $n(n-1)i$ input symbols and the memory and total delay of the two block interleavers is $2ni$ symbols. The staircase effect of the deep staggering is depicted in Figure 4.6, for $n = 4, i = 3$, where the step size is elongated with respect to the usual helical interleaver. Interchanging the block and helical interleavers yields a shallow staggered interleaver. If the code has length ni symbols then each i consecutive input symbols is treated as a single character for the helical interleaver. The block interleaver uses a block length of i and is of depth $n-1$. The memory and total delay of the two helical interleavers is then $n(n-1)i$ symbols and the memory and total delay of the two block interleavers is $2(n-1)i$ symbols. The staircase effect of shallow staggering is depicted in Figure 4.7 for $n = 4, i = 2$ where the elongated step riser is observed.

4.6 Comments

This chapter has attempted to survey interleaver design techniques. Since the essence of interleaving is the implementation of variable delay, it is natural to realize them in terms of delay lines. It is commented in [15] that when the delay lines begin to exceed several tens of thousands of bits long then a RAM implementation may be more effective. At this point one is advised to abandon the delay line concept and design the interleaver from the beginning from a RAM point of view.

The actual performance of an interleaver in a system seems difficult to assess and depends on the actual distribution of delays rather than just the parameters b and n for example. One would like a graceful degradation of performance as the designed burst length is exceeded. It is hoped to further investigate interleaver properties and their impact on system performance in future studies.

Chapter 5

Coding and Diversity

5.1 Introduction

The purpose of this chapter is to consider certain questions on the trade-off between diversity and coding for systems, such as spread spectrum systems, where a low code rate is anticipated. Ultimately, the aim is to determine actual performance trade-offs in terms of signal-to-noise ratios and probability of error. For this report however, only distance properties of codes, both block and convolutional, will be examined.

The work finds its origin in the interesting paper of Chase[19] who showed that in some cases when a low rate code is to be used there is very little penalty in terms of code distance, if any, in using a higher rate code and replicating the code symbols (diversity) to achieve a lower rate. While this observation was not a focus of that work it appears to be an interesting one. The advantages of using a higher rate code with diversity is the lower complexity of the decoder. This theme is expanded upon here. Specifically, the trade-off between diversity and coding for codes over nonbinary alphabets is considered from the point of view of minimum distance. The section 5.2 investigates this question for convolutional codes. It begins with a closer look at the Heller bound for convolutional codes over F_q. Punctured convolutional codes are also considered to obtain good initial higher rate codes. Section 5.3 briefly considers the same problem for block codes where, for simplicity, only Reed-Solomon codes are considered. The section 5.4 considers future directions for this work.

5.2 Diversity and Convolutional Codes

A more general version of the Heller[20] bound for convolutional codes is first derived using a trivial extension of an argument in Ryan and Wilson[21]. Some implications of the bound are then considered and the question of the trade-off between diversity and coding is investigated using the bound. A subsidiary question of codes over alphabets of different sizes is also considered.

5.2.1 The Heller Bound

The Heller bound is proved in [21] using the Plotkin bound which, in its general form states that for a code (linear or nonlinear) with M codewords over F_q, the minimum distance between any two codewords, d, is upper bounded by

$$d \leq \frac{M(q-1)}{(M-1)q} n \ .$$

The bound is essentially a reflection of the fact that the minimum distance cannot be greater than the average distance between codewords.

To apply this to convolutional codes of rate b/n over F_q with memory $m = K - 1$, the code generator circuit is viewed as having b shift registers, each of length k with n output adders with connections to the bK register cells. The generator matrix of the code may be viewed in the form

$$\mathrm{G} = \begin{bmatrix} \mathrm{g}_0 & \mathrm{g}_1 & \cdots & \mathrm{g}_m & 0 & \cdots \\ 0 & \mathrm{g}_0 & \cdots & \mathrm{g}_{m-1} & \mathrm{g}_m & \cdots \\ & & \cdots & & & \cdots \end{bmatrix}$$

where each g_i is a $b \times n$ matrix over F_q. Consider input words of length $(m+L)b$ symbols comprising Lb data symbols followed by mb zeros. This gives rise to $q^{Lb} - 1$ nonzero codewords of length $(m+l)n$. Applying the Plotkin bound to this situation then yields

$$d_f \leq \frac{q^{Lb}(q-1)}{(q^{Lb}-1)q}(m+L)n$$

and since L can be chosen as an arbitrary positive integer and d_f is an integer, we have

$$d_f \leq \min_L \lfloor \frac{q^{Lb}(q-1)}{(q^{Lb}-1)q}(m+L)n \rfloor \ .$$

The bound appears to be the strongest for $b = 1$ i.e. for codes of rate $1/n$ for which it becomes

$$d_f \leq \min_L \lfloor \frac{q^{L-1}(q-1)}{(q^L-1)}(m+L)n \rfloor \ . \tag{5.1}$$

It is observed in [21] that for most cases the minimization of Equation (5.1) is obtained for values of L of 1 or 2 where L is the data sequence length. This situation is first investigated. For $L = 1$ the bound is (recall that $m = K - 1$)

$$d_f^{(1)} \leq nK$$

and in many cases that will be of interest here, this bound is actually achieved. For $L = 2$ Equation (5.1) becomes

$$d_f^{(2)} \leq \lfloor \frac{q(q-1)}{q^2-1}(K+1)n \rfloor = \lfloor \frac{q}{(q+1)}(K+1)n \rfloor$$

which can be manipulated to

$$d_f^{(2)} \leq \lfloor nK + n(\frac{q-k}{q+1}) \rfloor \ . \tag{5.2}$$

The argument in Equation (5.2) is less than nK iff $K > q$, requiring the consideration of paths of length at least 2.

When $L = 3$ the argument in Equation (5.1) is

$$\frac{q^2(q-1)}{q^3-1}(K+2)n = (K+2)n - \frac{(q^2-1)}{(q^3-1)}(K+2)n$$

and the bound can be manipulated to

$$d_f^{(3)} \leq \lfloor nK + (q-1)n\{\frac{2q^2 - K(q+1)}{q^3-1}\} \rfloor \ . \tag{5.3}$$

It is an easy matter to verify that $d_f^{(3)} < d_f^{(2)}$ iff $K > q^2 + q - 1$ and also that if $K > 2q^2/(q+1) = 2(q-1) + 2/(q+1)$ or $K > 2q - 1$, then $d_f^{(3)} < d_f^{(1)}$.

From such arguments a clear picture emerges as to when path lengths greater than one must be considered to achieve d_f. For the remainder of this section interest will be largely in convolutional codes over F_q that achieve $d_f = nK$ and so these observations are not of direct interest. It is likely however that they will become important for later investigations.

K	Rate 1/2		Rate 1/4	
	d_f	d_{bound}	d_f	d_{bound}
3	5	5	10	10
4	6	6	13	13
5	7	8	16	16
6	8	8	18	18
7	10	10	20	20
8	10	11	22	22
9	12	12	24	24
10	12	13	27	27
11	14	14	29	29
12	15	16	32	32
13	16	16	33	33
14	16	17	36	36

Table 5.1: Maximum free distance for rate 1/2 and 1/4 binary codes (Larsen[22]).

5.2.2 Diversity versus Coding

The question of the trade-off between diversity and coding is considered from an elementary point of view in that codes are compared for free distance. While simplistic, this approach is adequate for the present purpose. Denote by r the rate of a convolutional code and by l the order of diversity i.e. each symbol is repeated l times and by (r, l) a combination of coding and diversity to give an overall code rate of r/l. The purpose here is to determine under what conditions it is possible to achieve the same, or similar, d_f with an (r, l) combination as with coding alone. The complexity of decoding a convolutional code of rate $1/n$ over F_q with memory m (shift register length $K = m + 1$) is proportional to q^{K-1}, and this quantity will be independent of the code rate for the codes of interest here. Nonetheless when using a low code rate there may be advantages to implementing it as a high rate code with diversity. Although the Viterbi decoder will have the same number of states, the implementation can take advantage of the diversity to achieve simplifications. Consider first the work of Larsen[22] who found, by computer search, rate 1/2, 1/3 and 1/4 binary codes that achieve maximal free distance. Part of those results are reproduced in Table 5.1 where the bound is that of Equation (5.1) for the appropriate parameters.

l	d_f	d_{bound}
1	10	10
2	20	20
4	40	41
8	80	82
16	160	164
32	320	329
64	640	658

Table 5.2: The coding/diversity example of Chase[19].

It is seen that in most cases the bound is close to being achieved and that in some cases the best rate 1/4 code can be achieved by using a rate 1/2 code with diversity $l = 2$ diversity.

In a similar vein, Table 5.2 shows the observation of Chase[19]. Starting from the well known rate $1/2, K = 7, d_f = 10$ code, the minimum distance achieved using diversity $l = 2^i$ is compared with the bound of Equation (5.1).

It is noted that the relative difference between the distance achieved with diversity is small compared to the maximum possible by the bound of Equation of (5.1). It is not known if the bound is tight for the very low rates but it is clear from this information that it would be easier to implement the rate 1/2 code with diversity 2^i than the best rate $1/2^i$ code. Again the evidence indicates that it will involve very little loss to implement the high rate code with diversity compared to the corresponding low rate code.

A few comments on the q-ary case will support the same general conclusion. It appears from equations (5.1) to (5.3) that for $q \geq K$, rate $1/n$ codes exist over F_q with $d_f = nK$. In the following this will be assumed to be the case without specific mention of it.

For rate 1/2 codes, $d_f = 2K$ and for rate 1/4 codes $d_f = 4K$. Clearly the rate 1/4 codes are easily achieved by using a rate 1/2 code with 2 diversity i.e. a $(1/2, 2)$ code rather than designing a rate 1/4 code. It is also possible to consider starting with a rate 3/4 code and using a 3-diversity i.e. a $(3/4, 3)$ code. Since the free distance cannot be improved upon and the simplest way to achieve the rate 3/4 code would be to puncture a rate 1/2 code,

	(2/3,2)		1/3
K	d	$2d_f$	d_f
2	3	6	6
3	4	8	9
4	6	12	12
5	7	14	15
6	9	18	18
7	10	10	21
8	12	24	24

Table 5.3: Rate 1/3 q-ary codes.

there seems little point in pursuing this approach.

For a rate 1/3 code it appears they can be constructed with $d_f = 3K$ for all $q \geq K$. In fact they can be constructed by puncturing every fourth symbol of the rate 1/4 code (which we have observed can be constructed as a $(1/2, 2)$ code) and this does indeed yield a code with $d_f = 3K$. To illustrate another (inferior) approach consider the following. For the rate 1/2 code, puncture every fourth symbol to yield a rate 2/3 code. Use this code in a $(2/3, 2)$ code to give a rate 1/3 code. The free distances achievable by this approach are shown in Table 5.3.

As noted previously it is assumed that q is sufficiently large (≥ 8) to allow the existence of maximal d_f codes.

Consider next the rate 1/6 codes. Clearly the maximum free distance of $6K$ can be achieved by using either a maximum distance rate 1/2 code and 3 diversity or a maximum rate 1/3 code and diversity 2. The rate 2/3 code mentioned previously obtained by puncturing might also be used with 4 diversity to give the results of Table 5.4, in which only the codes for even values of K give the maximum free distance and for K odd have a free distance of two less than the maximum possible.

Rate 1/5 codes with maximum free distance can be obtained by puncturing every sixth symbol from a maximum free distance rate 1/6 code.

K	d	$4d$	$d_f = 6K$
2	3	12	12
3	4	16	18
4	6	24	24
5	7	28	30
6	9	36	36
7	10	40	42
8	12	48	48

Table 5.4: d_f for the (2/3,4) codes compared to the 1/6 codes.

5.2.3 The Role of the Alphabet Size

Implicit in much of the discussion found in the literature on q-ary convolutional codes is the notion that q is chosen so that some natural modulation scheme such as q-FSK or q-PSK is to be used. For q-ary block codes, however, in particular Reed-Solomon codes, it is not at all uncommon for q-ary symbols to be transmitted as bit streams by some binary modulation scheme. This section initiates a discussion on the problem by considering the construction of codes to be transmitted for one value of q but constructed from another value of q. The discussion is very preliminary and only one simple example is considered as an introduction to the kind of problems of interest.

Consider a rate 1/2 code over F_q for $K = 4$ and $d_f = 8$. The code could be decoded using a Viterbi decoder with 8^3 states (high by current practice). If pairs of symbols are grouped together and interpreted as symbols of F_{64}, the coded stream can be viewed as coming from a rate 1/2 code over $F_{64}, d_f = 4$ although some caution is required for this interpretation. There may be some advantage to reviewing the relationship between q and the modulation system used. Again the determining factor will be in the error performance.

5.3 Diversity and Reed-Solomon Codes

The arguments of the preceeding section can be repeated for block codes. It is easier and more instructive to deal only with Reed-Solomon codes. The extended Reed-Solomon codes over F_q have the parameters: length $= q$, dimension k and minimum distance $= q - k + 1$.

Assume that a low rate code $r = k/q$ is to be used with distance $q - k + 1$. The minimum distance is relatively high and the code is capable of correcting many errors, implying a relatively complex decoding algorithm. Suppose now that l-diversity is used i.e. a $(kl/q, l)$ code/diversity combination, $kl < q$. The minimum distance of this code (now of length lq) is $l(q - lk + 1)$. The structure of this code might allow novel uses of the diversity before using the lower error correction capability of the code. Notice that the minimum distance of the $(kl/q, l)$ code is greater than that of the coded case only when $lq - l^2k + l > q - k + 1$ or when $k < (q + 1)/(m + 1)$. For example, when $l = 3$ and $q = 256$ this gives $k < 257/4 = 64$ indicating that perhaps the code with diversity has an advantage over the pure code case for these parameters.

Notice that although the $(lk/q, l)$ decoder is less complex than the k/q code, since it is required to correct far fewer errors, it does in fact have an effective block length of l times that of the pure code case. It is argued however that this extra length does not significantly add to the decoder complexity and the comparison of the two systems is fair.

Again the final determination between the two systems will be in the error performance which in turn will depend on how the diversity information is used in the receiver. These are topics for future investigation.

5.4 Comments

This initial investigation has considered the trade-off between diversity and coding from the point of view of minimum distance for both block and convolutional codes. Of more importance is the translation of the trade-offs considered here into an understanding of how it affects system performance on a variety of channels, such as the additive white Gaussian noise, Rayleigh fading and interference channels. Future work will consider these questions and attempt to determine guidelines for this trade-off.

Chapter 6

Repeated Convolutional Codes for High Error Rate Channel

6.1 Introduction

In this chapter, we consider error correction schemes that can correct errors at the output of a high error rate channel. Such a large channel error rate may result from the presense of strong interference or jamming. Conventional error correction schemes, such as the widely used constraint length 7 and rate 1/2 binary convolutional code due to Odenwalder[23] which is an international standard[24], may fail in such situations. It is clear that a low rate code must be used for such a channel by considering the channel capacity or cutoff rate.

Recently, Kasami, *et al* have considered a cascaded coding scheme for a binary symmetric channel (BSC) with a large error probability p_e [25]. Their scheme consists of two linear block codes. The inner code (closer to the channel) is a binary code and the outer code is a Reed-Solomon (RS) code. The parameters of the inner and outer codes have to be properly chosen to match each other in order to obtain a good performance. It turns out that for a large p_e whether a coding scheme works or not is very sensitive to p_e. For example, in [25], a scheme that consists of (63,31) RS outer code and (32, 6) biorthogonal inner code works well at $p_e = 0.2$ but will not work at $p_e = 0.3$. The sensitivity to the values of p_e and the somewhat rigid structure of the cascaded scheme implies that we should know p_e before designing a coding scheme. Also, two encoder/decoder systems are needed for a cascaded scheme. In a jamming environment, however, it is hardly possible to predict p_e.

Thus a system that can easily adapt to the actual p_e would be desirable.

In 1977, Shaft searched low rate convolutional codes and considered their use to combat burst interference[26]. Use of repeated convolutional codes seemed to be favored. In 1985, Chase proposed the scheme again for BSC more clearly with well made arguments[19]. To show that repeated binary convolutional codes are near optimum, both Shaft and Chase compared their free distances with Heller's bound for binary convolutional codes[20]. Chase also made a comparison of code rates with the channel capacity of BSC.

Nevertheless, there are still some practical problems that need to be addressed. For instance, for the BSC, if each code symbol is repeated m times, maximum likelihood decoding requires $m+1$ levels of quantization. Since m can be very large for a high channel error rate and practical convolutional decoders have a finite, and likely a smaller number of quantization levels, what is the corresponding performance degradation? Further, can we use a repeated binary convolutional code for an M-ary symmetric channel (MSC) and what is the best way to generate a binary decoding metric for use in the binary decoder? This question is motivated by the fact there are commercially available binary codecs at high speed and considerable efforts are being made to further improve the speed and reduce the cost of such codecs.

In this chapter, we consider repeated convolutional codes for an MSC (with BSC as a special case) with a large error probability p_e. The value of p_e can be near, but smaller than, $1 - 1/M$ for which the channel capacity is zero. In Section 6.2, we focus on the BSC and begin with a conventional analysis based on the union bound for BSC. For a large m, the central limit theorem is applied to provide another analytical tool. In Section 6.3, Monte Carlo simulation results for BSC are provided and compared with theoretical analyses. The quantization effect is shown. Based on these results, we compare the code rates of repeated convolutional code with the channel cutoff rate. In Section 6.4 we consider the use of a binary code over an MSC. The emphasis is placed on the methods to generate binary decoding metrics and their performances.

6.2 Theoretical Analysis for BSC

In this paper, we are particularly interested in the above mentioned Odenwalder rate 1/2 constraint length 7 convolutional code. The single-chip encoder/Viterbi decoder is commercially available at a low price from several sources. The decoder normally has up to eight levels of quantization. We consider the Viterbi decoding which is maximum likelihood decoding when infinite quantization is assumed. Each M-ary channel symbol is repeated m times. We call m the repetition order.

In the case of BSC, for each encoded symbol, it is repeated m times over a BSC. The BSC is assumed to have a large error probability (transition probability) p_e which is in the neighborhood of 0.1 or higher, but of course, smaller than 0.5. BSC is a proper channel model for anti-jam communication systems with complex demultiplexing between demodulator and decoder. In such a situation, the decoder has to cope with a hard decision channel and explicit and/or implicit interleaving/deinterleaving makes the errors random. One example of possible implicit interleaving/deinterleaving is a multiplexed multi-user system where each user has a decoder after demultiplexing.

It is clear that for two trellis paths at Hamming distance d, the repetition of order m will increase the distance to $d \times m$. It is well known that the decoder output bit error rate (BER) P_b can be upperbounded by an exponentially tight union bound. Specifically, suppose the P_d is the pairwise error probability of two trellis paths with Hamming distance d, then

$$P_b < \sum_{d=d_{free}}^{\infty} C_d P_{dm} \tag{6.1}$$

where d_{free} is the free distance of the convolutional code and C_d is the total number of information bit errors when pairwise errors between paths with Hamming distance d occur. For the Odenwalder code, C_d is nonzero only for even d and $d \geq 10$, since $d_{free} = 10$. Over the BSC, for an even d,

$$P_d = \frac{1}{2}\begin{pmatrix} d \\ d/2 \end{pmatrix} p_e^{d/2}(1-p_e)^{d/2} + \sum_{i=d/2+1}^{d} \begin{pmatrix} d \\ i \end{pmatrix} p_e^i (1-p_e)^{d-i}. \tag{6.2}$$

C_d can be found by expanding the transfer function of the convolutional code or using computer search through the trellis of the code. For the Odenwalder code, the first nine

terms are[4]: $C_{10} = 36$, $C_{12} = 211$, $C_{14} = 1{,}404$, $C_{16} = 11{,}633$, $C_{18} = 77{,}433$, $C_{20} = 502{,}690$, $C_{22} = 3{,}322{,}763$, $C_{24} = 21{,}292{,}910$, $C_{26} = 134{,}365{,}911$.

Using Equations (6.1) and (6.2), for $m = 3$, 7 and 15, respectively, P_b is plotted versus p_e for the Odenwalder code in Fig. 6.1 using the first term, the first four terms and the first nine terms of C_d, respectively. It can be seen from the figure that nine terms of the transfer function provides a sufficiently accurate bound, especially at a low P_b. The results using the first nine terms are used in the rest of the paper. In fact, it has been known that the union bound provides an accurate approximation for a low P_b provided enough quantization levels are available to facilitate the maximum likelihood decoding (MLD). For MLD it is well known that the decoding metric should be

$$mt_k = \sum_{i=1}^{m} r_{ki} \tag{6.3}$$

where r_{ki} is the received i-th repeated symbol (0 or 1) over the BSC for k-th convolutional encoded symbol. Here $mt_k = 0$ represents the k-th encoded symbol to be most likely a 0 and $mt_k = m$ represents the k-th encoded symbol to be most likely a 1. For eight levels of quantization from 0 to 7 (where 0 represents the most reliable logic 0 and 7 represents the most reliable logic 1), uniform quantization is natural and reasonable. Then, the above metric is modified as

$$mt_k = \lfloor \frac{\sum_{i=1}^{i=m} r_{ki}}{m} \times 7 + 0.5 \rfloor \tag{6.4}$$

where $\lfloor x \rfloor$ is the largest integer not exceeding x.

As mentioned earlier, the repetition order m must be large in order to correct the errors with a large probability p_e. For a large m we may apply the central limit theorem or the Gaussian approximation of the binomial probability distribution. Consider the following metric which is equivalent to Equation (6.3)

$$\alpha_k = \frac{\sum_{i=1}^{i=m} (-1)^{r_{ki}}}{m}. \tag{6.5}$$

Note that $(-1)^{r_{ki}}$ has a mean $E = 1 - 2p_e$ $(= (1 - p_e) \times 1 + p_e \times (-1))$ if the k-th encoded symbol is 0, and a mean $E = -(1 - 2p_e)$ if the k-th encoded symbol is 1. The variance is the same and is given by

$$\sigma^2 = [1 - (1 - 2p_e)]^2(1 - p_e) + [-1 - (1 - 2p_e)]^2 p_e = 4p_e(1 - p_e). \tag{6.6}$$

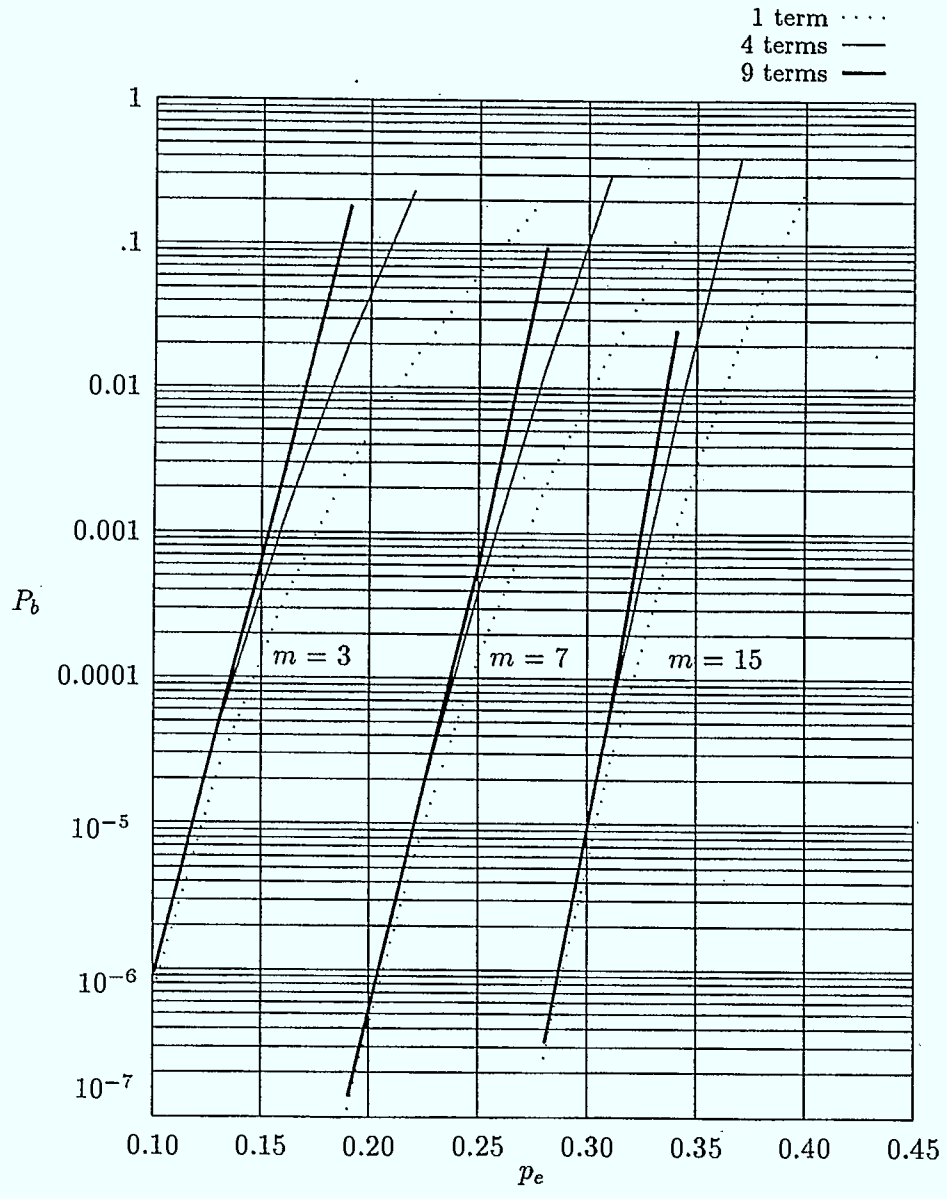


Figure 6.1: Union bounds for the repeated Odenwalder code over the BSC using the first term, the first four terms and the first nine terms of the transfer function, respectively. $m = 3$, 7 and 15.

By use of the central limit theorem (see, e.g. [27]), we know that

$$\beta_k = \frac{\sum_{i=1}^{i=m}[(-1)^{r_{ki}} - E]}{\sqrt{m}} \tag{6.7}$$

is a zero-mean Gaussian random variable with variance σ^2 when $m \to \infty$ (or m is very large). Since

$$\alpha_k = \frac{1}{\sqrt{m}}[\frac{\sum_{i=1}^{i=m}(-1)^{r_{ki}}}{\sqrt{m}}] = \frac{1}{\sqrt{m}}[\frac{[\sum_{i=1}^{i=m}[(-1)^{r_{ki}} - E)] + mE}{\sqrt{m}}] = \frac{\beta_k}{\sqrt{m}} + E \tag{6.8}$$

then, for a large m, α_k is also a Gaussian random variable with mean equal to E and variance

$$\sigma_\alpha^2 = \frac{4p_e(1-p_e)}{m}. \tag{6.9}$$

Thus for a sufficiently large m, the variance can be reduced to an arbitrarily small number. Compared with the coherently demodulated BPSK in additive white Gaussian noise (AWGN) with the noise spectral density N_0[10], the asymptotical Gaussian distribution of α_k implies an effective symbol energy (half of the bit energy for the rate 1/2 code) $E_s' = (1-2p_e)^2$ and an effective noise spectral density $N_0'/2 = \sigma_\alpha^2$, i.e.,

$$E_s'/N_0' = \frac{(1-2p_e)^2}{8p_e(1-p_e)}m. \tag{6.10}$$

Note the effective signal to noise ratio is proportional to the diversity order m. Since the simulated or measured BER curve for the Odenwalder code is well known (see, e.g. [8]), for a large m and a given p_e, we can use Equation (6.10) to determine the required E_s'/N_0', and thus m to sustain a required P_b. Even for a small m, the Gaussian approximation can be used to estimate the required m, and then adjust it therefrom.

Since the simulated or measured BER curves for the Odenwalder code take into account the finite levels of quantization and other practical constraints such as a finite trellis length, these factors are also included in the BER curves of the repeated convolutional code if the Gaussian approximation is used. In other words, for a large m performance degradation due to finite quantization, etc., will be about the same as what we have known for the convolutional coded coherent BPSK in AWGN.

In Fig. 6.2, P_b, obtained from Gaussian approximation, vs. p_e is given for $m = 3$, 7, 15, and 31, respectively. For $m > 7$, eight levels of quantization and trellis length 84 are

assumed. This trellis length is considered instead of five or six times the constraint length because 84 or so has been used in commercial realizations in order to accommodate the punctured rate 3/4 code[24]. For $m \leq 7$, MLD decoding is assumed which can be materialized with eight levels of quantization. In this case, the interest is to see the approximation error of the Gaussian approximation. For comparison, the results based on the union bound are also given in Fig. 6.2.

It can be seen from the figure that for a small m the Gaussian approximation results in a lower BER. For a reasonably large m (e.g. $m \geq 15$) the Gaussian approximation seems to be fairly accurate which needs to be verified by simulation.
It is also noted that for $m = 31$, the BER obtained from Gaussian approximation is slightly higher than the union bound. The basic reason for this difference is that for the union bound, ideal maximum likelihood decoding is assumed, i.e., no quantization and infinite trellis length, etc., while for the Gaussian approximation curve, practical constraints have been taken into account.

6.3 Computational Results for BSC

In order to verify the BER performance, Monte Carlo simulation has been performed. The trellis length is 84 and eight levels of quantization is assumed and Equation (6.4) is used to generate the metric for various m. Fig. 6.3 shows simulated P_b vs. p_e for $m = 3$, 5, 7, 15 and 31, respectively. Union bounds and the Gaussian approximation are also shown for comparison. For the latter only $m = 15$ and 31 are considered because m is supposed to be large for the Gaussian approximation.

It can be seen from the figure that for $m < 8$ the union bound, which assumes the MLD, is almost exact. Note now for $m = 3$, we have metric 0, 2, 5, 7 which means that we are not doing exactly MLD. This applies to $m = 5$ as well. But the performance degradation is insignificant. For a large m, the Gaussian approximation is fairly close to the simulation results. Note that finite quantization results in a higher BER that is not

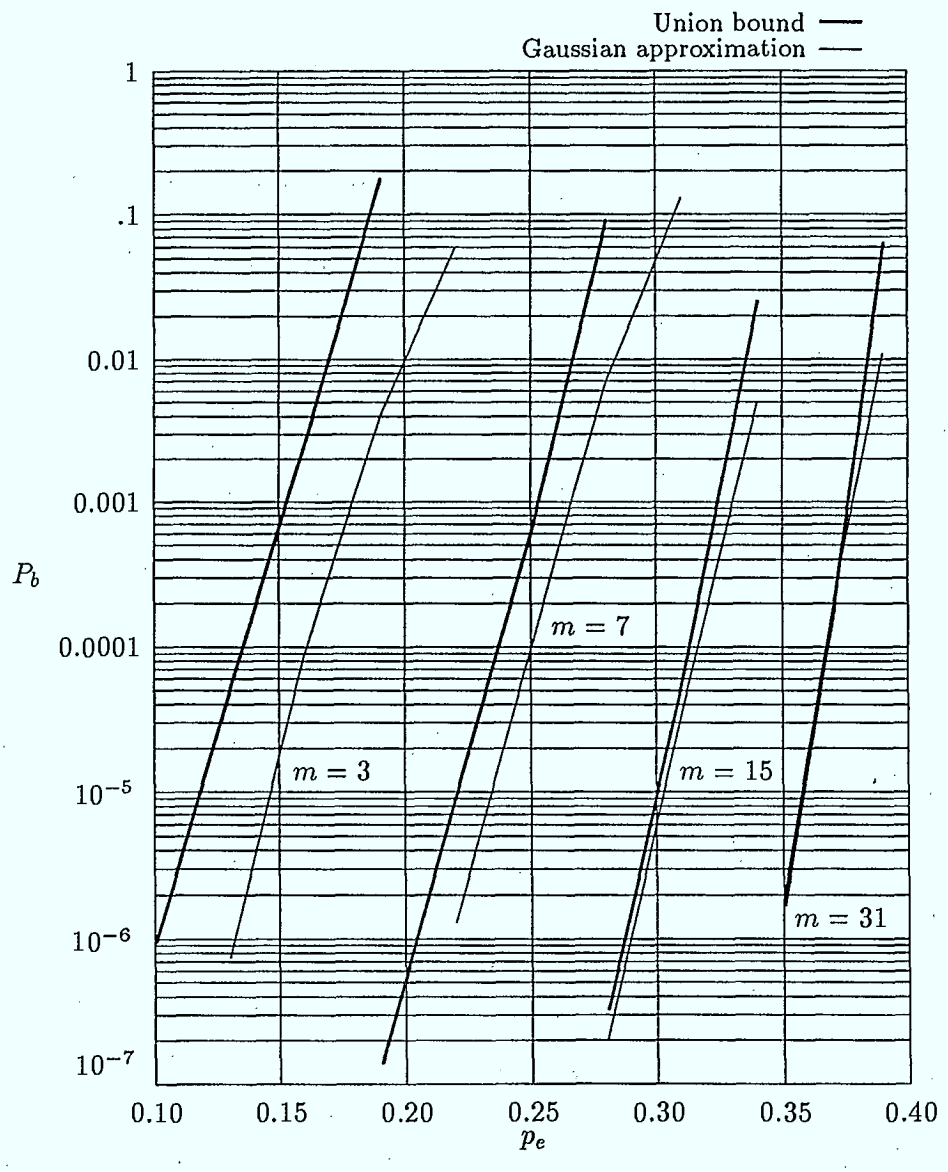


Figure 6.2: BER based on the Gaussian approximation and the union bound for the repeated Odenwalder code over the BSC. $m = 3$, 7, 15 and 31.

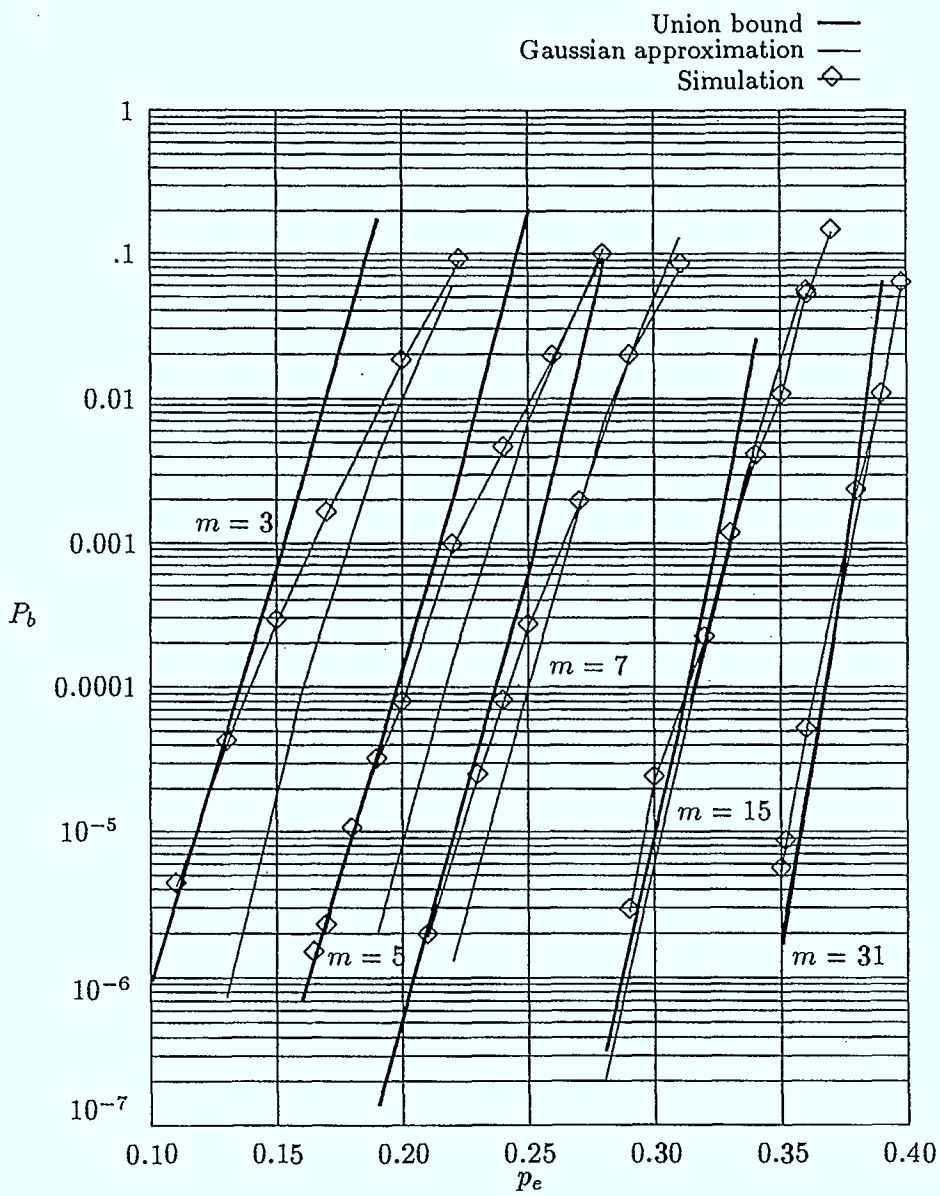


Figure 6.3: BER based on simulation, the union bound and the Gaussian approximation for the repeated Odenwalder code over the BSC. $m = 3$, 5, 7, 15, and 31.

upperbounded by the union bound, and in fact, Gaussian approximation is more accurate than the union bound at a high BER.

For $m = 5$ the overall code rate is 0.1 which is slightly higher than the rate, 0.092, of the abovementioned coding scheme considered in [25] with comparable BER performance. In consideration of its simplicity, the repeated Odenwalder code is favoured.

Fig. 6.4 shows the cutoff rate R_0 for the BSC[27] and the overall code rate $r = \frac{1}{2m}$ to sustain $P_b = 10^{-4}$ (based on the simulated BER) which was also used in [19]. From this figure, it seems that r moves closer to R_0 as p_e increases. But from r/R_0 vs. p_e as shown in Fig. 6.5, r decreases faster than R_0 as p_e increases. Nevertheless, it is interesting to note that the deviation between r and R_0 is bounded: as p_e approaches 0.5, r approaches $0.6R_0$. If compared to the channel capacity, r is near 30 percent of channel capacity. In conclusion, the repeated Odenwalder code can achieve more than one half of what is promised by the cutoff rate even for very large p_e, say, 0.3 to 0.5.

6.4 M-ary Symmetric Channel

In this section, we consider the M-ary symmetric channel (MSC) with high symbol error probability, which is illustrated in Fig. 6.6. Here p_e is the symbol error probability, which is near $1-\frac{1}{M}$, but smaller than it. Again, the coding scheme consists of an outer convolutional code and an inner repetition code where each M-ary channel symbol is repeated m times. This MSC model directly reflects the hard-decision demodulated fast frequency hopped MFSK (FFH/MFSK) where the repetition is inherent in the system. Here we are especially interested in $M = 4$ and 8. Previous work has shown that, under certain conditions, they represent best compromise in order to combat both partial band noise jamming and multitone jamming (see, e.g. [4]). As mentioned earlier, hard decision may be due to complex demultiplexing between the demodulator and decoder. We first consider a Trumpis code[30] as the outer code which is optimum for an M-ary orthogonal channel. 4-ary and 8-ary $R = 1$ bit/channel symbol Trumpis codes with constraint length 7 are considered. This code has the same implementation complexity as the Odenwalder code because of the same constraint length. In view of commercially available binary codecs, we then consider

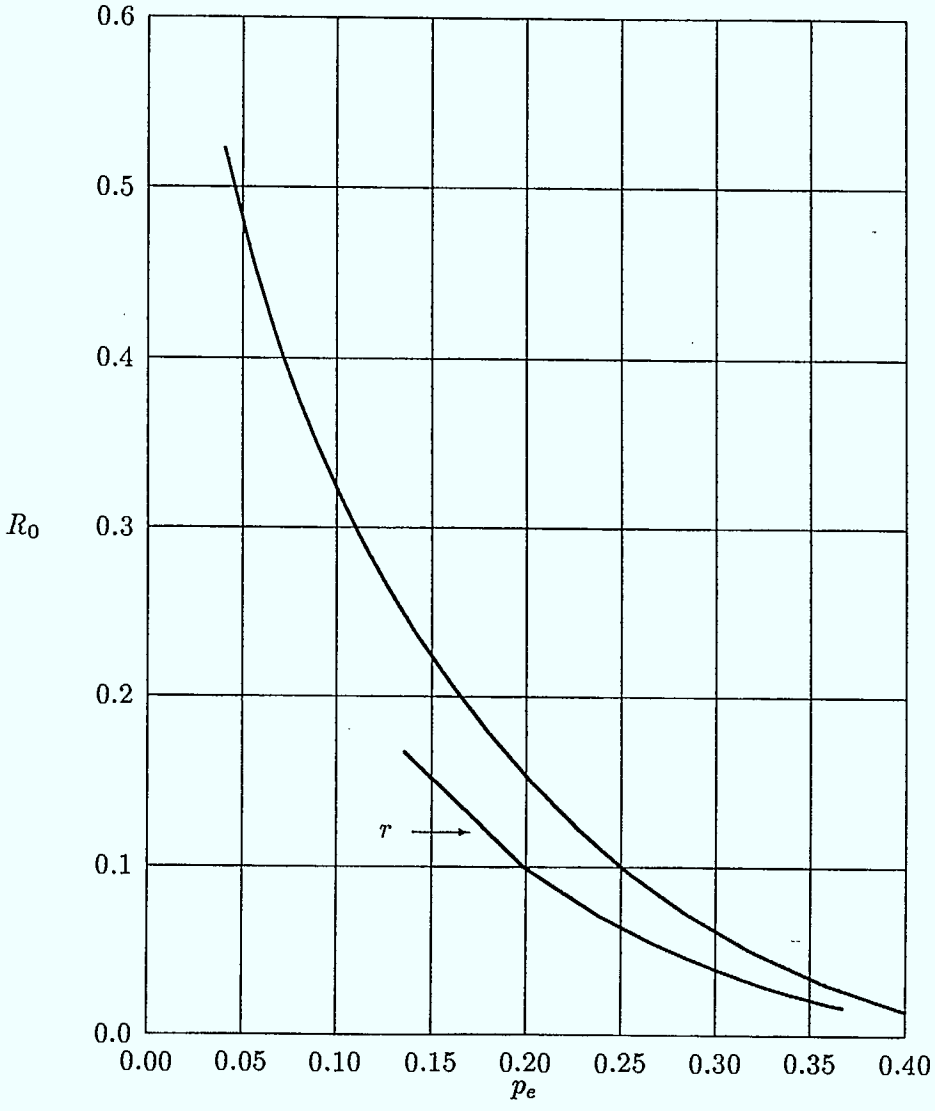


Figure 6.4: Comparison of the cutoff rate R_0 of the BSC and the overall code rate r of the repeated Odenwalder code over the BSC to sustain $P_b = 10^{-4}$.

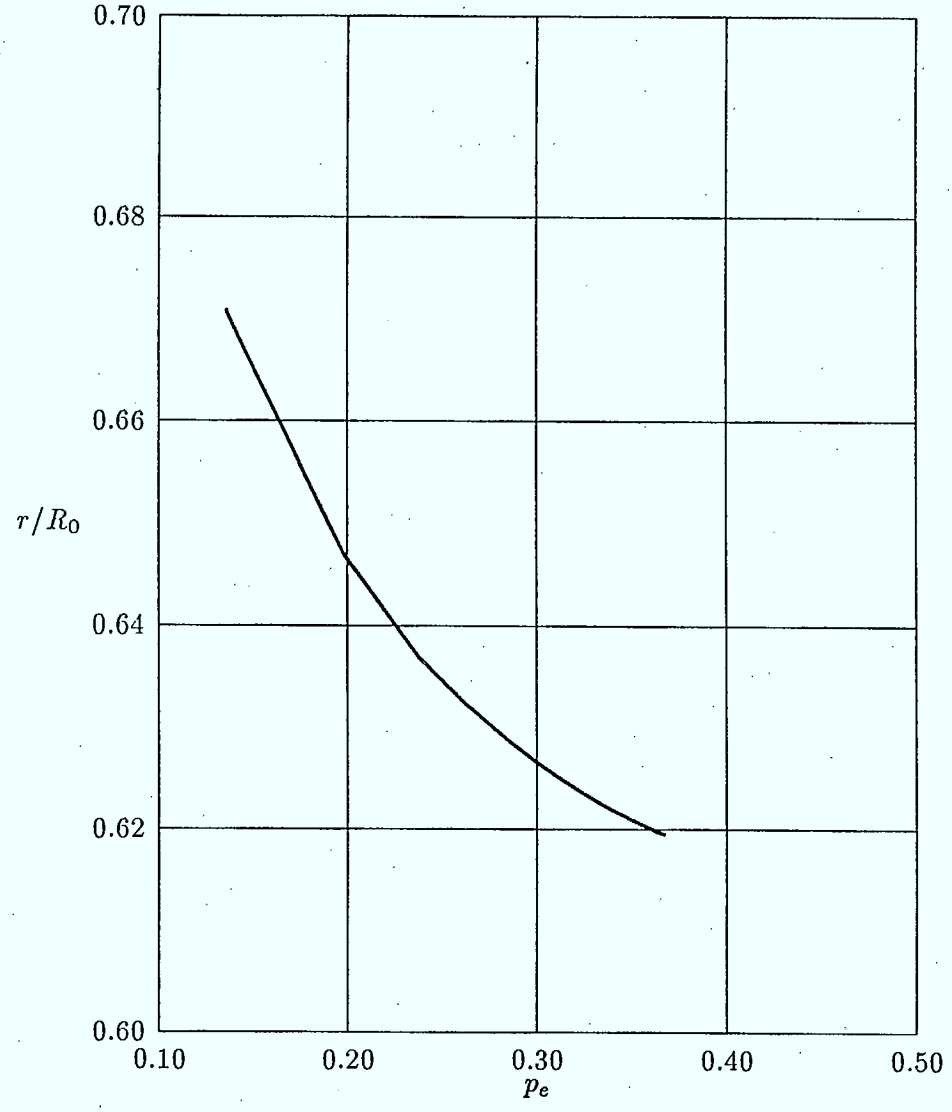


Figure 6.5: Ratio of the overall code rate r of the repeated Odenwalder code over the BSC to sustain $P_b = 10^{-4}$ to the cutoff rate R_0 of the BSC.

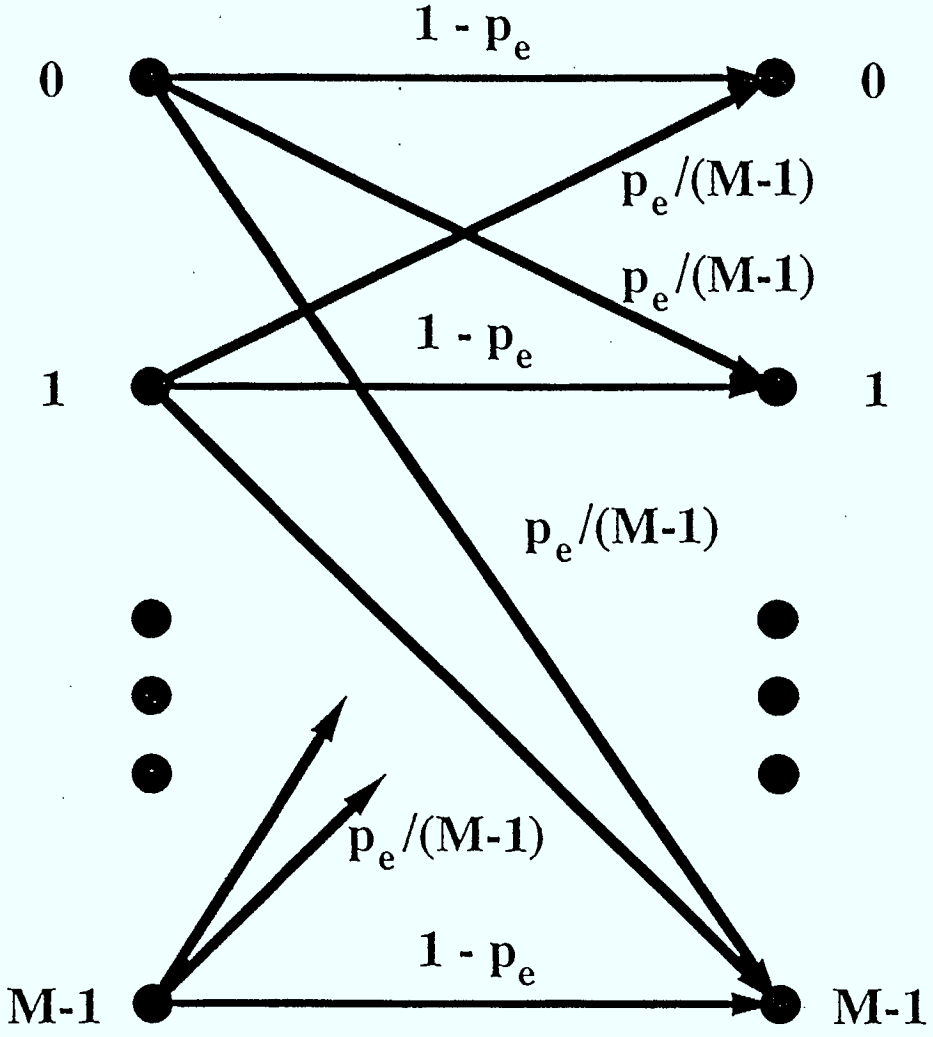


Figure 6.6: Model of M-ary Symmetric Channel

the possible use of the constraint length $K = 7$, rate $R = \frac{1}{2}$ Odenwalder code for the MSC. The emphasis is placed on how to generate binary decoding metrics.

6.4.1 M-ary Metric

For the Trumpis codes, it is known that the union bound of the decoder output BER P_b is [30]

$$P_b \leq 7P_{7m} + 39P_{8m} + 104P_{9m} + 352P_{10m} + 1187P_{11m} + \cdots \text{ for 4-ary channel} \quad (6.11)$$

and

$$P_b \leq P_{7m} + 4P_{8m} + 8P_{9m} + 49P_{10m} + 92P_{11m} + \cdots \text{ for 8-ary channel} \quad (6.12)$$

where P_d is the pairwise error probability between two trellis paths with Hamming distance d.

To consider the use of the Odenwalder code over a 4-ary symmetric channel (4SC), the most natural way is as follows. Recall that the encoder of the rate 1/2 code generates a pair of encoded bits at the encoder output for each incoming information bit. This pair of encoded bits can be considered as a 4-ary symbol and transmitted m times over the 4SC. In decoding, ideally, two encoded bits corresponding to one trellis branch will be assigned a 4-ary metric. This assumes that the decoder can accommodate 4-ary metrics. Using a trellis search algorithm, we found the union bound of the decoder output BER P_b as

$$\begin{aligned} P_b \leq \quad & P_{6m} + 10P_{7m} + 38P_{9m} + 92P_{9m} + 355P_{10m} + 1440P_{11m} \\ & +4684P_{12m} + 16043P_{13m} + 52240P_{14m} + 170679P_{15m} + \cdots \end{aligned} \quad (6.13)$$

Note the Trumpis codes are optimum over MSC in the sense that they have the largest M-ary free Hamming distance (7 for the 4-ary code) and fewest information bit errors due to path errors at the free distance. The Odenwalder code is not optimum for the 4SC. The free 4-ary Hamming distance is 6, which is one less than the optimum Trumpis code. But the number of information bit errors due to an incorrect trellis path at the free distance is only one. Thus we may expect that the Odenwalder code will have near optimum BER performance.

The use of the Odenwalder code over 8-ary symmetric channel (8SC) is similar. The encoder of the rate 1/2 code generates three pairs of encoded bit for every three incoming information bits. Then the first pair of encoded bits and one bit of the second pair of encoded bits are considered as an 8-ary symbol. The other bit of the second pair and the third pair are considered as another 8-ary symbol. Each of these 8-ary symbols is transmitted over 8SC m times. At the decoder, an 8-ary metric will be assigned to three encoded bits corresponding to one and a half trellis branches. Of course, it is assumed that the decoder can accommodate an 8-ary metric. We found the union bound of this kind of decoder output BER as

$$\begin{aligned} P_b \leq \quad & 3P_{5m} + 28P_{6m} + 83P_{7m} + 649P_{8m} + 2419P_{9m} + 10295P_{10m} \\ & +45175P_{11m} + 193378P_{12m} + \cdots \end{aligned} \tag{6.14}$$

By comparing with (6.12), we find the Odenwalder code is not bad over 8SC. The 8-ary free Hamming distance is 7 for the rate 1/3 Trumpis code, and 5 for the rate 1/2 Odenwalder code.

For the MSC with a repeated M-ary code, the maximum likelihood decoding metric for each M-ary symbol is the Hamming distance between the sequence of m repeated symbols and the corresponding received symbol sequence of length m. Here it is implied that for an M-ary symbol, a smaller metric is more favorable in that the M-ary symbol is more likely to be transmitted. This MLD metric is an M-ary metric in the sense that there are a total of M metrics for all M M-ary symbols.

For MLD, we can find the pairwise error probability between two paths with Hamming distance d, P_d. Recall that the exact meaning of P_d is, given a correct transmitted trellis path, the probability of a specific trellis path at Hamming distance d having a more favourable path metric than the correct path. Let us consider one symbol period where there is a symbol difference from the two paths. The correct symbol is called c and the symbol from the incorrect path is called e. Because $M > 2$, the MSC output can be neither c nor e. In fact the probability for the channel output to be c or e, denoted as p_{ce}, is given by

$$p_{ce} = 1 - p_e + \frac{p_e}{M-1}.$$

Then over d symbol periods where two paths have different symbols, there can be j ($0 \leq j \leq d$) periods where the channel output is neither c nor e and hence no contribution can be made to the metrics for either c or e. We can calculate a conditional pairwise error probability $P'_d(j)$ over the rest $d-j$ symbol periods where the channel output must be either c or e with probabilities (which are conditional probabilities under the condition that the channel output must be either c or e) of $\frac{1-p_e}{p_{ce}}$ and $\frac{p_e}{p_{ce}(M-1)}$, respectively. Specifically, we have

$$P'_d(j) = \sum_{i=\lceil \frac{d-j}{2} \rceil}^{d-j} (1 - 0.5\delta(i - \frac{d-j}{2})) \begin{pmatrix} d-j \\ i \end{pmatrix} \left(\frac{p_e}{(M-1)p_{ce}}\right)^i \left(\frac{1-p_e}{p_{ce}}\right)^{d-j-i} \tag{6.15}$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x, and

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0; \\ 0 & \text{otherwise.} \end{cases}$$

In fact, the δ function is equal to 1 (so that $1 - 0.5\delta = 0.5$) only if $d-j$ is even and $i = \frac{d-j}{2}$. Otherwise δ function is 0 and $1 - 0.5\delta = 1$. The probability that j of d periods where the channel output is neither c nor e is

$$P(j,d) = \begin{pmatrix} d \\ j \end{pmatrix} p_{ce}^{d-j}(1-p_{ce})^j. \tag{6.16}$$

Therefore, the pairwise error probability is

$$\begin{aligned} P_d &= \sum_{j=0}^{d} P(j,d) P'_d(j) \\ &= \sum_{j=0}^{d} \begin{pmatrix} d \\ j \end{pmatrix} p_{ce}^{d-j}(1-p_{ce})^j \times \\ &\quad \times \sum_{i=\lceil \frac{d-j}{2} \rceil}^{d-j} (1 - 0.5\delta(i - \frac{d-j}{2})) \begin{pmatrix} d-j \\ i \end{pmatrix} \left(\frac{p_e}{p_{ce}(M-1)}\right)^i \left(\frac{1-p_e}{p_{ce}}\right)^{d-j-i}. \end{aligned}$$

There is a factor P_{ce}^{d-j} in the inner summation and it can be canceled with the one at outside. So finally we get

$$\begin{aligned} P_d &= \sum_{j=0}^{d} \begin{pmatrix} d \\ j \end{pmatrix} \left(\frac{M-2}{M-1} p_e\right)^j \\ &\quad \times \sum_{i=\lceil \frac{d-j}{2} \rceil}^{d-j} (1 - 0.5\delta(i - \frac{d-j}{2})) \begin{pmatrix} d-j \\ i \end{pmatrix} \left(\frac{p_e}{M-1}\right)^i (1-p_e)^{d-j-i}. \end{aligned} \tag{6.17}$$

Using Equations (6.11), (6.13) and (6.17) for m = 3,7,15, and 31, the bound of P_b versus p_e is plotted for the Trumpis code and Odenwalder code with M-ary MLD decoding metric in 4-ary channel in Fig. 6.7 (curve a and b). From this figure, we can see that the Trumpis code is indeed better. However, it is interesting to note that the performance of the Odenwalder code is only slightly inferior to that of the Trumpis code. This is the basis for considering the use of the Odenwalder code over the 4-ary channel.

Using Equations (6.12), (6.14) and (6.17) for m = 3,7,15, and 31, the bound of P_b in an 8-ary channel is plotted in Fig. 6.8 (curve a and b). It appears that the performance of the Trumpis code is much better than that of the Odenwalder code in the 8-ary case. But recall that the 8-ary Trumpis code is a rate 1/3 code. So the code rate of this Trumpis code is only two thirds of the code rate of the Odenwalder code. The direct comparison in Fig. 6.8 is not fair.

Since the whole code rate of the repeated Trumpis and Odenwalder code are $\frac{1}{3m}$ and $\frac{1}{2m}$, respectively, if the repetition order m for the Odenwalder code is chosen to be 50 percent larger than that for the Trumpis code, the code rate for both repeated codes are the same, and then comparison can be made. So the union bound of P_b for the Odenwalder code with m = 5, 11, 23, and 47, and with m = 4, 10, 22, and 46 are plotted in Fig. 6.9. The bound of P_b for the Trumpis code with m = 3,7,15, and 31 are also plotted in Fig. 6.9. The corresponding code rate of the three groups are almost the same, but code rates of the Odenwalder code with m = 5,11,23, and 47 are a little bit lower than that of the Trumpis code, and code rates of the Odenwalder code with m=4,10, 22 and 46 are a little bit higher.

From Fig. 6.9, we can see that for BER less than 10^{-4}, the curve for the Trumpis code is in the middle of the space between the two curves for the Odenwalder code. Considering two curves corresponding to the Odenwalder code with a higher and a lower code rate, respectively, we can see that the repeated Odenwalder code would have almost the same performance as the repeated Trumpis code at the same overall code rate. Therefore, the same conclusion as in 4-ary channel can be drawn that the performance of the Odenwalder code is only slightly inferior to that of the Trumpis code. Because the comparison is based on the union bounds which are quite loose at high BER area, and the number of terms used in computing those union bounds are different, our comparison is only made at a low BER.

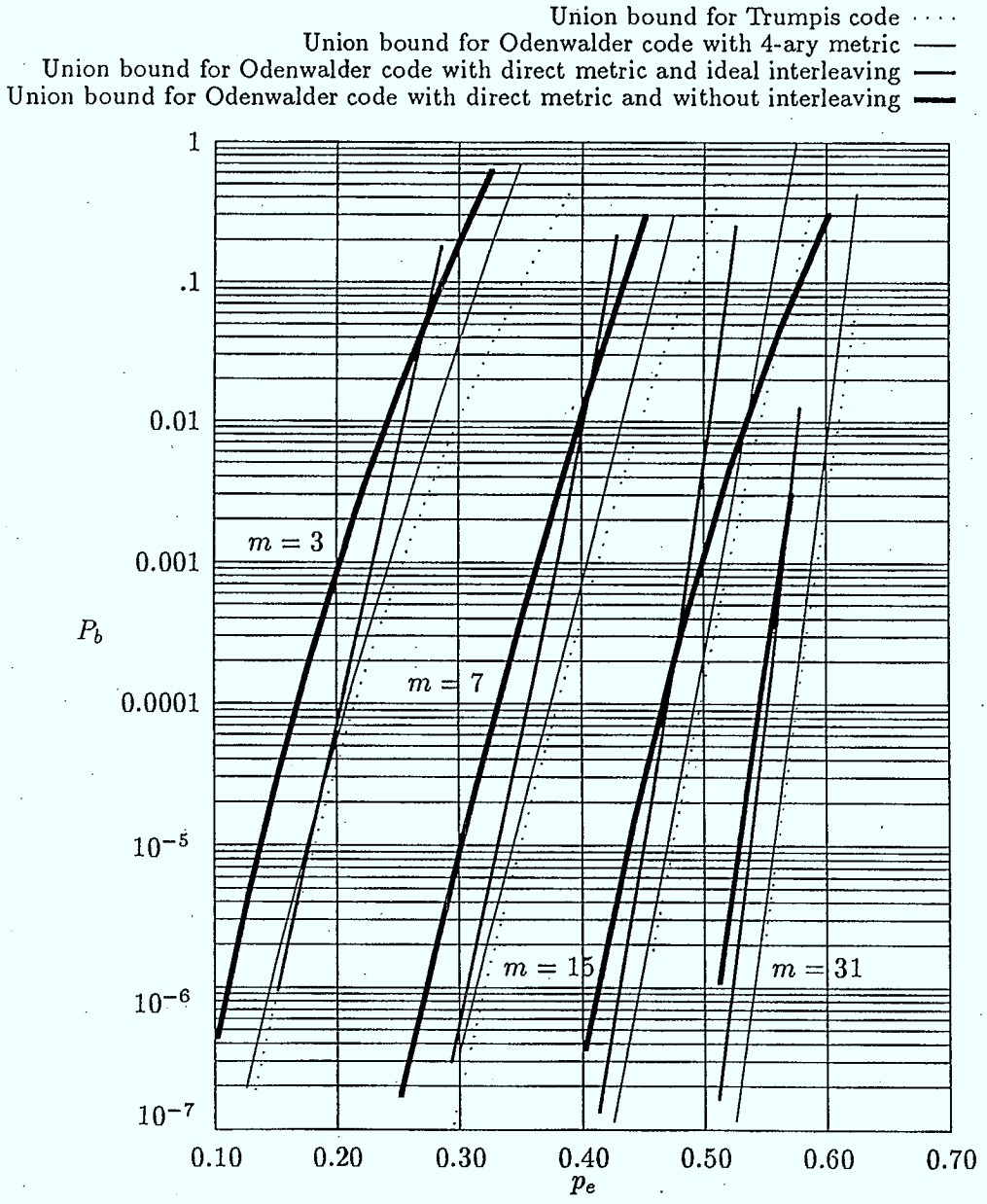


Figure 6.7: The union bound for the repeated Trumpis code and Odenwalder code with three kinds of metrics over 4-ary symmetric channel. m=3,7,15, and 31.

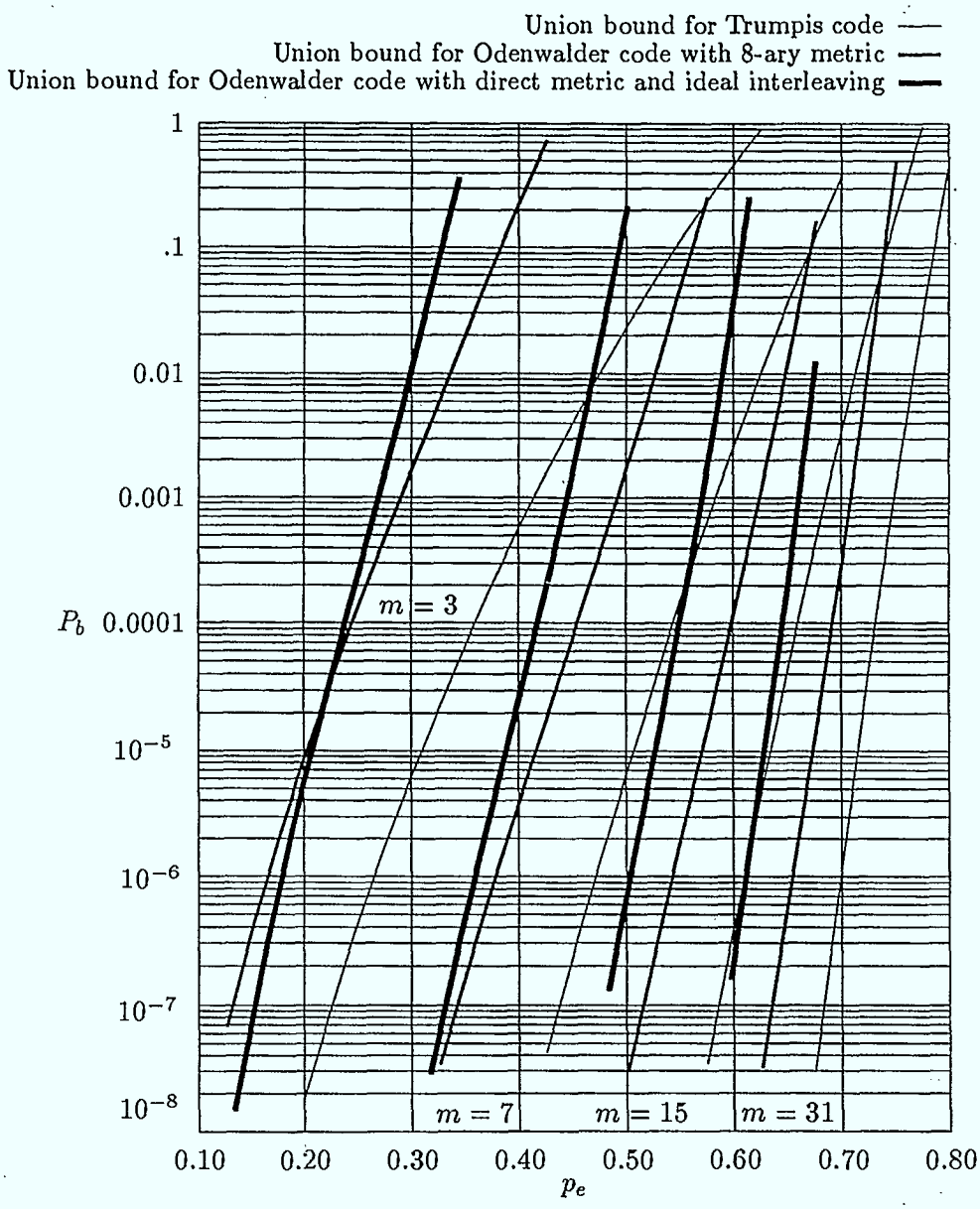


Figure 6.8: The union bound for the repeated Trumpis code and Odenwalder code with two kinds of metrics over 8-ary symmetric channel. m=3,7,15, and 31.

Note that the union bound for the Odenwalder code is based on the assumption that the decoder can accommodate M-ary metrics for MLD. This is generally not the case if we want to use commercially available decoding chips directly. In this case, the decoder is designed to accommodate binary metrics only. With this constraint, the above union bound should be understood as an upperbound for the BER performance of the decoder using binary metrics. But the free M-ary Hamming distance of the binary code shown in the union bound of Equation (6.13) and the corresponding error coefficient provide a basic indicator on whether or how well the binary code can work over the M-ary channel at all.

Since the performance degradation of the Odenwalder code is small relative to the optimum Trumpis code using M-ary metrics, the code is a good candidate for the M-ary system from a practical point of view. The practicality is that we can use commercially available codec chips provided we can properly generate binary decoding metrics. Further performance degradation will be introduced by using binary metrics because a binary metric is not a MLD decoding metric in an M-ary channel. How much the degradation will be depends on how binary metrics are generated. In the following sections, we consider several possible methods of generating binary decoding metrics and their performances.

Binary Metric Approximation of M-ary Metrics

Since the use of a binary decoder requires binary metrics, M-ary metrics can not be used directly in binary decoder. The most natural attempt would be to approximate M-ary metrics with $\log_2 M$ binary metrics. This method avoids interleaving.

Since the trellis decoding is based on comparing the metrics of different trellis paths, adding a number to all M metrics in one symbol period will not affect the decoding performance. Therefore, we only need to be concerned about differences between metrics for different M-ary symbols. There are $M-1$ M-ary metric differences.

Let m_{00}, m_{01}, m_{10} and m_{11} be the 4-ary metrics, respectively. The maximum likelihood decision decoding requires that branch metric for symbol ij is m_{ij} and the survivor has the smallest path metric. Thus MLD can be implemented by considering three differences between four m_{ij}s. Unfortunately, they can not be represented by two binary metrics

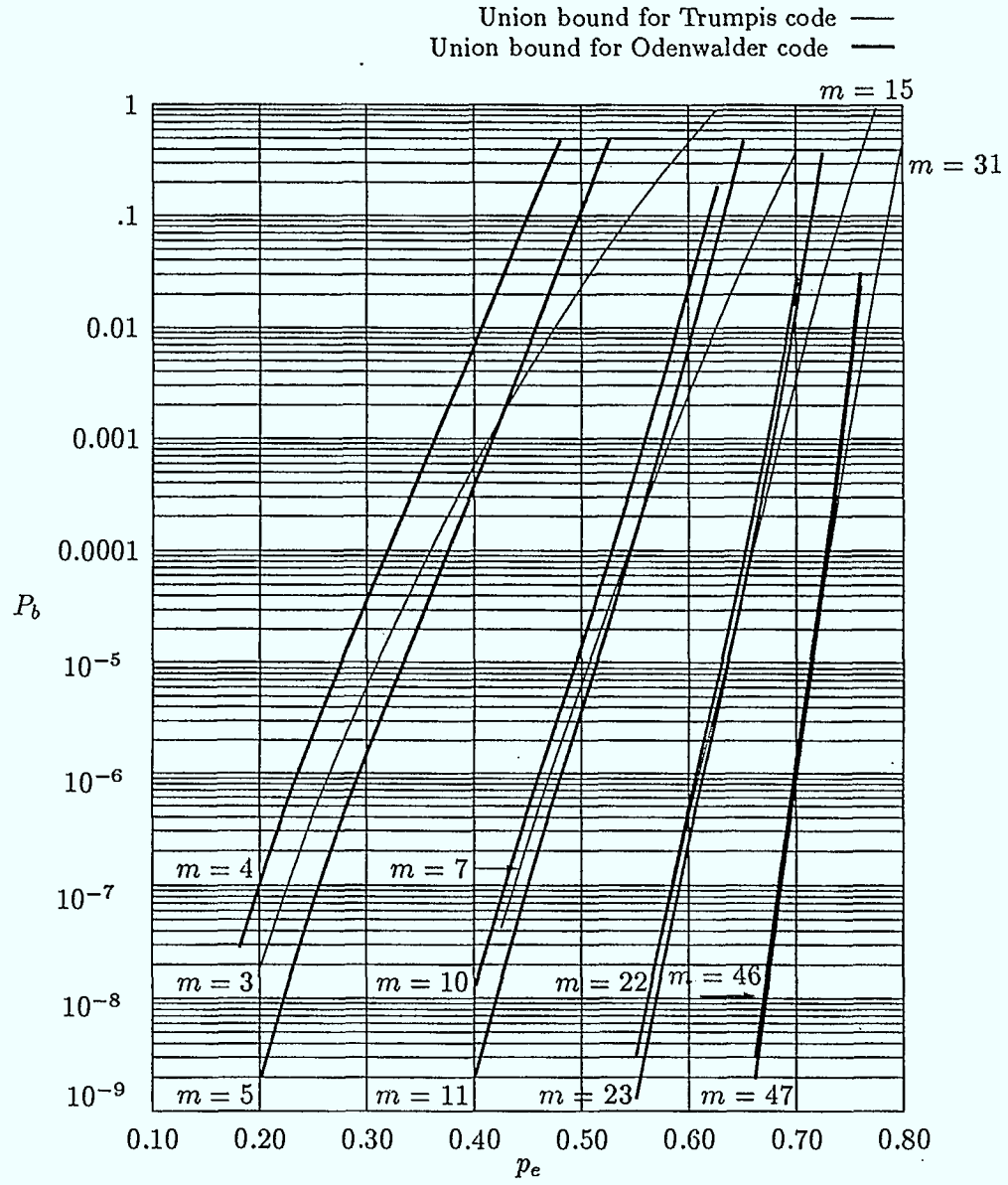


Figure 6.9: The union bound for the repeated Trumpis code and Odenwalder code with 8-ary metric over 8-ary symmetric channel. m=3,7,15, and 31 for Trumpis code; m=4,10,22, and 46 and m=5,11,23, and 47 for Odenwalder code.

exactly.

Assume the metric range is [0,7]. Without loss of generality, we assume m_{00} to be the smallest. Suppose the branch metric is $B_{00} = a + b$ corresponding to two binary metrics for symbol 00, a and b. Then for symbol 01, the branch metric is $B_{01} = a + 7 - b$. And we have

$$B_{00} - B_{01} = 2b - 7.$$

Because $m_{00} \leq m_{01}$, it is natural to require $B_{00} - B_{01} \leq 0$, i.e., $b \leq 3.5$. If we require

$$B_{00} - B_{01} = m_{00} - m_{01}$$

then

$$2b - 7 = m_{00} - m_{01}$$

hence

$$b = \frac{7 - (m_{01} - m_{00})}{2}.$$

For symbol 10, the branch metric $B_{10} = 7 - a + b$. Similarly, because $m_{00} \leq m_{10}$, $a \leq 3.5$. And if we let

$$B_{00} - B_{10} = m_{00} - m_{10}$$

then

$$a = \frac{7 - (m_{10} - m_{00})}{2}.$$

For symbol 11, the branch metric B_{11} is

$$B_{11} = 7 - a + 7 - b.$$

Because $a \leq 3.5$ and $b \leq 3.5$,

$$B_{11} - B_{01} = 7 - 2a \geq 0$$

and

$$B_{11} - B_{10} = 7 - 2b \geq 0.$$

This means that no matter what $m_{11} - m_{00}$ is, the term B_{11} always gives the least favorable metric. This problem is inveitable as long as there are only two binary metrics used. This is simply because if 00 is the most favorable symbol, 11, which is the farthest symbol to 00

in binary Hamming distance, should be the least favorable in binary metric representation. Thus the 4-ary metric m_{11} is not always preserved depending on its value, but the metrics m_{10} and m_{01} are genuinely preserved.

In summary, the proposed method to generate binary metrics to approximate 4-ary metrics is as follows:

1. Find $m_{ij} = \min(m_{00}, m_{01}, m_{10}, m_{11})$, where $i, j \in (0,1)$. Denote 1's complement of i as $\bar{i}$ and j as $\bar{j}$.

2. Then compute
$$a = \frac{7 - (m_{\bar{i}j} - m_{ij})}{2},$$
$$b = \frac{7 - (m_{i\bar{j}} - m_{ij})}{2}.$$

3. The actual two binary metric sent into the decoder, b_1 and b_2 are
$$b_1 = a(1-i) + (7-a)i,$$
$$b_2 = b(1-j) + (7-b)j.$$

For a larger M, for example, $M = 8$, there are seven 8-ary metric differences but only three binary metrics. The method given above can only accurately represent three out of seven differences. Thus it does not appear to be proper to extend the method to a larger M. In the following sections, we consider more general methods. The basic principle of these methods is to generate "sensible " binary metrics directly from M-ary metrics without attempting to approximate them.

6.4.2 Binary Metric Generation

When a binary code is used over an M-ary channel, the $\log_2 M$ encoded bits at the output of encoder are mapped into M-ary symbols through a one-to-one mapping. At the receiver, the received M-ary symbol is mapped back to the group of binary bits, and the corresponding metric for each binary bit is generated accordingly. The optimum binary metric generation method is the one which has the BER performance closest to that of Odenwalder codes with M-ary metrics, i.e., the curve b in Fig. 6.7. Here we propose a binary metric

generation method called the direct binary metric generation method.

Direct Binary Metric Generation Method

At the receiver, after receiving m M-ary symbols, we can generate the binary metric in the following way:

1. Mapping m received M-ary symbols back to m binary bits groups, respectively.
2. For each of $\log_2 M$ bits, accumulate over m repetitions the number of 0 or 1 received and form a binary metric like the one discussed in Section 2 for BSC.
3. Feed these binary metrics to the decoder in a certain order.

For example, for 4-ary symbols 0,1,2, and 3, we can map them to four groups of two binary bits, say, 00, 01,10, and 11, respectively. If $m = 3$ and the three received symbols are 0, 1, and 3, the corresponding two binary metrics are 1 and 2.

"Certain order" in step 3 depends on whether interleaving is used or not. Here we analyze two extreme cases, i.e., ideal interleaving and no interleaving at all.

Direct Binary Metric Generation With Ideal Interleaving

Obviously, the binary metrics generated using this direct method bear some dependence. Ideal interleaving makes that incoming metrics to the decoder are all statistically independent of each other over one decoder trellis length. This would require a block interleaver with an interleaving depth $\log_2 M$ and a span of at least of 5 to 6 times that of the constraint length. In this case, the M-ary symmetric channel can be simplified to the BSC model with the transition probability of BSC $p_e^{(B)}$

$$p_e^{(B)} = \frac{M}{2(M-1)} p_e. \tag{6.18}$$

Then analysis can be carried out easily in the same way as for the BSC model. Specifically, the analytical results in binary channel given in Equations (6.1) and (6.2) can be applied directly by substituting transition probability p_e by $\frac{M}{2(M-1)} p_e$. The bounds of P_b versus p_e are plotted for the Odenwalder code, with a direct binary metrics with ideal interleaving

for the 4-ary channel in Fig. 6.7 (curve c) and for the 8-ary channel in Fig. 6.8 (curve c), respectively.

Direct Binary Metric Generation Without Interleaving

No interleaving means that $\log_2 M$ consecutive binary decoding metrics are generated from one M-ary symbol. Here we consider the 4-ary case. Generalization for a larger M involves a higher level of sophistication but no more ingenuity.

For 4SC, the probability of receiving one of three wrong 4-ary symbols is $p_e/3$. However, two of three wrong symbols result in only one binary bit error, and the other one leads to two binary bit errors.

Consider two trellis pathes which differ in d bit positions, and where each branch in the paths contains one 4-ary symbol, or two binary bits. One path is considered as the correct path, while the other one is considered as the incorrect path. Assume d different positions reside in Z branches in the incorrect path. Among Z branches, there are two kinds of branches. One kind of branches are those with only one bit different from the corresponding branch in the correct path. The other kind are those with both bits different from the branch in the correct path. We call the first kind as one-bit-error branches, and the second kind as two-bit-error branches. For a received symbol, the metrics are different for these two kinds of branches.

Suppose the branch in the correct path is 00. Then the one-bit-error branch is either 01 or 10; and the two-bit-error branch is 11. We consider 01 as an example of one-bit-error branch. With the binary Hamming distance used as the metric, we have:

symbol received	probability	metric for correct branch (00)	metric for error branch (01)
00	$1\text{-}p_e$	0	1
01	$p_e/3$	1	0
10	$p_e/3$	1	2
11	$p_e/3$	2	1

For a two-bit-error branch, similarly, we have:

symbol received	probability	metric for correct branch (00)	metric for error branch (11)
00	1-p_e	0	2
01	$p_e/3$	1	1
10	$p_e/3$	1	1
11	$p_e/3$	2	0

Here we can see that the metrics are different for the two kinds of error branches, and therefore, for different combination of the two kinds of branches the pairwise error probabilities are different even for the same d, the total number of different position in bits.

To compute the union bound of the BER at the output of a decoder, we need to know how many one-bit-error branches and how many two-bit-error branches exist for each d, and the corresponding contributions of each combination to information bit errors.

Suppose there are X one-bit-error branches, and Y two-bit-error branches, and $X+Y=Z$. Then the union bound of BER at the output of the decoder is

$$P_b \leq \sum_{d=d_{free}}^{\infty} \sum_{X,Y \in \Gamma_d} C_d(X,Y) P_d(X,Y) \tag{6.19}$$

where Γ_d is the set of all possible X and Y combinations which are determined by the code. $C_d(X,Y)$ is the information bit error contribution for a trellis path with X one-bit-error branches, Y two-bit-error branches, and total d different positions from the correct path. $P_d(X,Y)$ is the pairwise probability of two trellis pathes with the binary Hamming distance d, and X one-bit-error branches and Y two-bit-error branches.

$C_d(X,Y)$ can be obtained by computer search. We have found $C_d(X,Y)$ for the Odenwalder code, and those for a small d are given in Table 6.1.

Now we compute the pairwise error probability $P_d(X,Y)$. Suppose that the correct path is the all zero path. If during $Z=X+Y$ transmission, symbol 00 is received k_0 times, symbol 01 k_1 times, symbol 10 k_2 times, and symbol 11 k_3 times, then the metric for the correct path corresponding the Z branches is

$$m_c = k_1 + k_2 + 2k_3. \tag{6.20}$$

To compute the metric of the error path, we have to consider how received symbols match the branches in the error path.

d	X	Y	$C_d(X,Y)$
10	2	4	1
	4	3	10
	6	2	25
12	4	4	13
	6	3	61
	8	2	137
14	4	5	29
	6	4	176
	8	3	792
	10	2	407
16	2	7	2
	4	6	42
	6	5	597
	8	4	3019
	10	3	5177
	12	2	2796

Table 6.1: $C_d(X,Y)$ of constraint length 7 Odenwalder code.

Let k_{0x} be the number of 00 received corresponding one-bit-error branches in the incorrect path, and k_{0y} be the number of 00 received corresponding two-bit-error branches in the incorrect path, and so on. Obviously,

$$k_0 = k_{0x} + k_{0y}$$

$$k_1 = k_{1x} + k_{1y}$$

$$k_2 = k_{2x} + k_{2y}$$

$$k_3 = k_{3x} + k_{3y}$$

and

$$k_{0x} + k_{1x} + k_{2x} + k_{3x} = X$$

$$k_{0y} + k_{1y} + k_{2y} + k_{3y} = Y.$$

As discussed in section 6.6, we can assume, without loss of generality, that all one-bit-error branches are 01. Then the metric for the incorrect path corresponding to the Z branches is

$$m_e = k_{0x} + 2k_{2x} + k_{3x} + 2k_{0y} + k_{1y} + k_{2y}. \tag{6.21}$$

The pairwise error probability is

$$P_d(X,Y) = \sum_{\Omega} u(m_c - m_e) \frac{X!}{k_{0x}!k_{1x}!k_{2x}!k_{3x}!} \frac{Y!}{k_{0y}!k_{1y}!k_{2y}!k_{3y}!} \left(\frac{p_e}{3}\right)^{k_1+k_2+k_3} (1-p_e)^{k_0} \quad (6.22)$$

where

$$\Omega = \{k_{ix}, k_{iy}, i = 0,1,2,3 \mid \quad 0 \le k_{ix}, 0 \le k_{iy}, i = 0,1,2,3, \sum_{i=0}^{3} k_{ix} = X, \sum_{i=0}^{3} k_{iy} = Y\}$$

and

$$u(x) = \begin{cases} 0 & x < 0; \\ 0.5 & x = 0; \\ 1 & x > 0. \end{cases}$$

By using (6.19), the union bound for the Odenwalder code in an 4-ary channel with direct generation metric without interleaving is computed and is also plotted in Fig. 6.7 (curve d).

Binary Metric Generation Based on M-ary Metric Without Interleaving

In order to generate binary metrics, Gong proposed a conversion scheme which converts the M-ary metrics into binary metrics [31]. For the i-th bit in $\log_2 M$ bits corresponding to an M-ary symbol, the binary metric is given by

$$\begin{aligned} b_i \quad = \quad & \max\{M\text{-ary metrics for symbols with } i\text{-th bit to be ``1''}\} \\ & - \max\{M\text{-ary metrics for symbols with } i\text{-th bit to be ``0''}\} \\ & i = 1,2,\ldots,\log_2 M. \end{aligned}$$

Here we use Gong's conversion scheme in the following way: first we find M-ary MLD metrics; then binary metrics are generated using the above equation. Since interleaving can cause a substantial delay in addition to its implementation cost, which sometimes is not desirable or tolerable, it is always interesting to know the trade-off between interleaving and the BER performance. Thus we consider both Gong's conversion scheme and our direct scheme without interleaving. It is interesting to compare the performance of these two schemes. Further, we note that the use of Trumpis codes does not require interleaving. Thus comparison based on no interleaving is fair to all cases.

6.4.3 Simulation Results

A Monte Carlo simulation is carried out to obtain the bit error rate at the decoder output for the Odenwalder code in a 4-ary symmetric channel with three metrics, our approximation of M-ary metrics, directly generated metric and Gong's conversion metric, and for the Odenwalder code in an 8-ary symmetric channel with two metrics, our directly generated metric and Gong's conversion metric, both without interleaving. The results are plotted in Fig. 6.10 for the 4-ary case and in Fig. 6.11 for the 8-ary case. For comparison, the union bound for the Odenwalder code with the M-ary metric, and the directly generated metric with ideal interleaving are also depicted in Fig. 6.10 and Fig. 6.11. The union bound for the Odenwalder code in a 4-ary channel with the direct generation metric without interleaving is plotted in Fig. 6.10 as well.

From simulation results in a 4-ary channel (Fig. 6.10), we can see that the directly generated metric gives the best performance among three metrics. For small m ($m \leq 7$), the approximation metric and the conversion metric have almost the same performance. But for large m, the conversion metric has a better performance. In Fig. 6.11, the simulation results in an 8-ary channel are similar. The direct generated metric has better performance. All three metrics are considered to work without interleaving, therefore the direct binary metric generation method is recommended when no interleaving is preferred.

It is also noted that the union bound for direct generation metric without interleaving is quite tight for BER less than 10^{-3}. Comparing the union bounds and simulation results with ideal interleaving and those for direct generation metric without interleaving, we can see that the difference of two cases gets smaller when m becomes larger. So when m is large ($m \geq 15$), interleaving may not improve the performance significantly, and therefore may not be necessary.

6.5 Concluding Remark

We have considered a repeated convolutional coding scheme for the MSC with a large p_e. BER performance has been both analyzed and simulated. We first considered BSC. A BER approximation method is proposed for a large m based on the central limit theorem. The

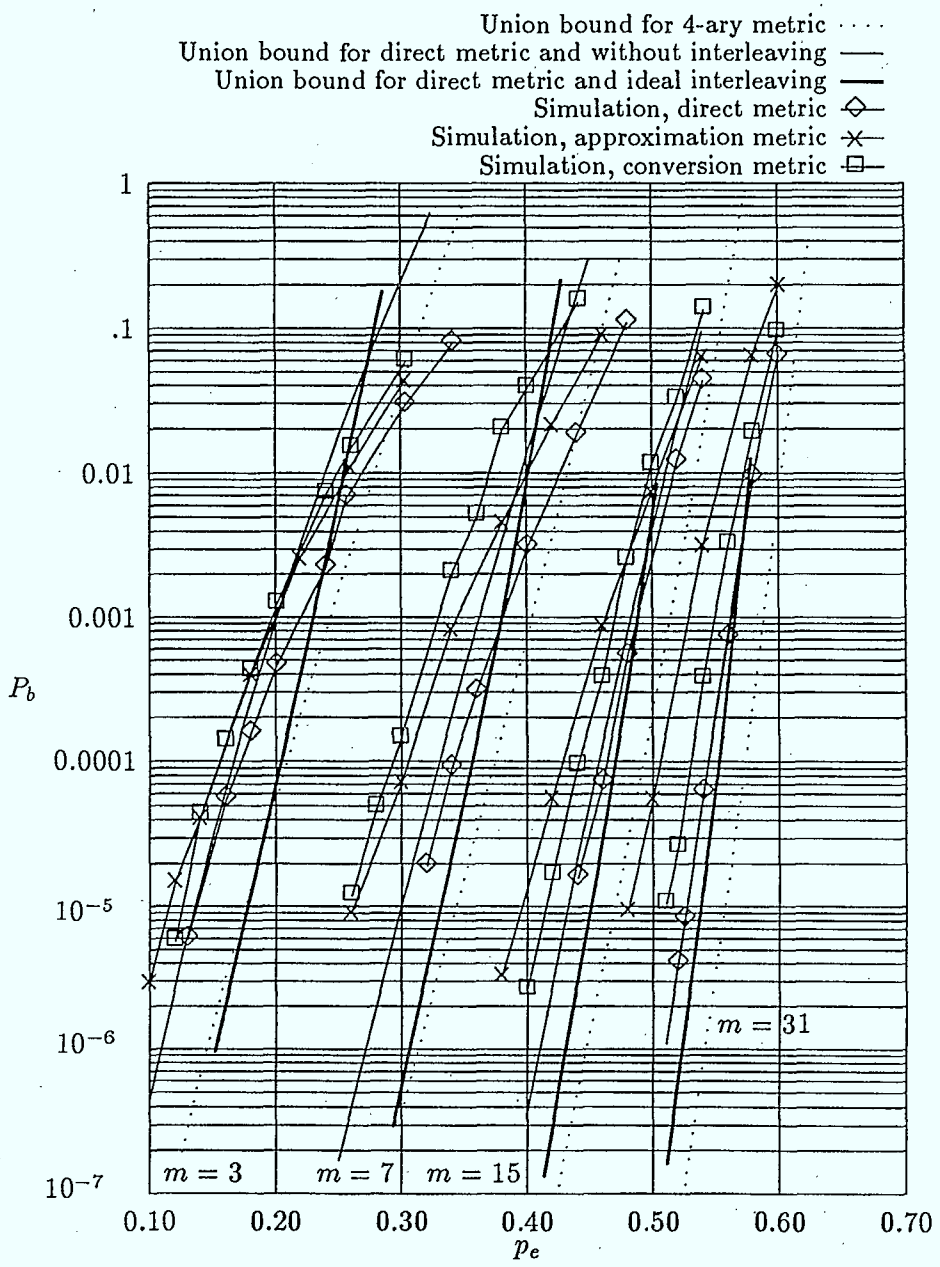


Figure 6.10: The Monte-Carlo simulation in 4-ary symmetric channel for BER of repeated Odenwalder code without interleaving and with direct generation metric, approximation metric, and conversion metric.

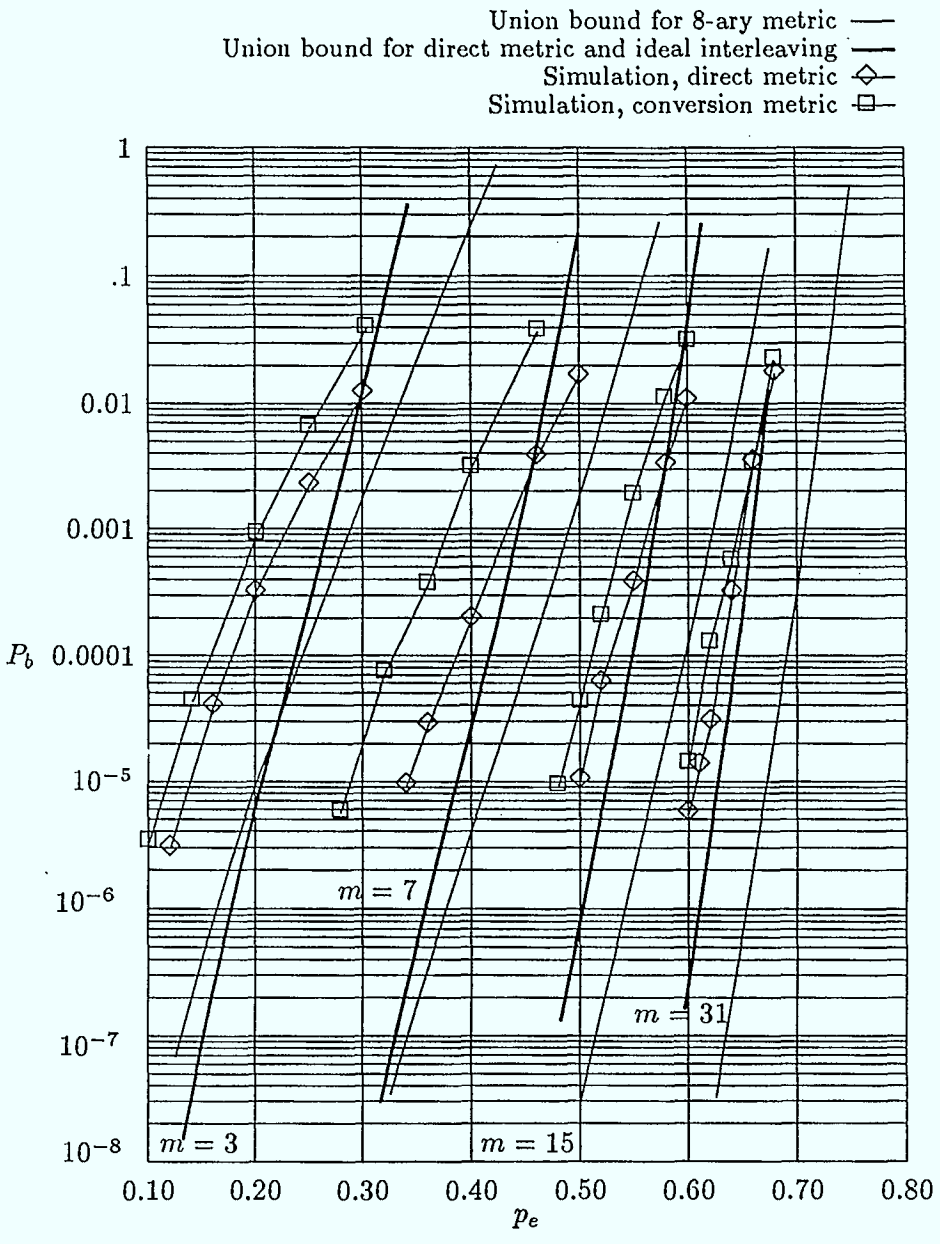


Figure 6.11: The Monte-Carlo simulation in 8-ary symmetric channel for BER of repeated Odenwalder code without interleaving and with direct generation metric and conversion metric.

overall code rate r is considered relative to the channel cutoff rate R_0. It has been shown that r is larger than $0.6R_0$. In comparison with a cascaded coding scheme proposed in [25], the repeated coding scheme has some clear advantages. One of them is that we can vary m to match the unknown p_e without changing the decoding procedure.

We then extended the repeated convolutional coding scheme to the M-ary symmetric channel. We have investigated the influence of various decoding metrics in the MSC model. If ideal interleaving is available, and m is not very large, then the repeated Odenwalder code with binary metric is almost as good as the one with M-ary metric. Further, the performance of the Odenwalder code over 4-ary and 8-ary channels is quite close to that of the optimum Trumpis code. When interleaving is not available, three methods of generating binary metrics from M-ary channel are proposed. The first is based on approximation of the differences of M-ary metrics with binary metrics. In the second, binary metrics are generated directly from M-ary metrics. The third method is a conversion method. Simulation results indicate that the direct binary generation method is the best for our coding scheme among all these binary metric generation methods. Therefore, the direct binary metric generation method is recommended if no interleaving is preferred. Union bound and simulation results also indicate that there is not much improvement by interleaving when m is large.

6.6 Further Analysis of One-bit-error Branch

In this appendix, we will show that no matter what one-bit-error branches actually are (all 01 or all 10 or combinations of 01 and 10), the pairwise probability can be obtained by assuming a convenient form, e.g. they are all 01.

Suppose the incorrect trellis path is A, and there are X_1 01 branches and X_2 10 branches in path A. Suppose that there are k_{0x_1} 00 symbols, k_{1x_1} 01 symbols, k_{2x_1} 10 symbols, and k_{3x_1} 11 symbols received corresponding to 01 branches, and k_{0x_2} 00 symbols, k_{1x_2} 01 symbols, k_{2x_2} 10 symbols, and k_{3x_2} 11 symbols received corresponding to 10 branches, respectively. Obviously,

$$k_{ix_1} + k_{ix_2} = k_{ix} \qquad \text{for } i = 0, 1, 2, 3$$

and

$$\sum_{i=0}^{3} k_{ix_1} = X_1,$$

$$\sum_{i=0}^{3} k_{ix_2} = X_2.$$

Then the metric for path A is

$$m_e(k_{2x_1} + k_{1x_2}) = 2(k_{2x_1} + k_{1x_2}) + C \tag{6.23}$$

where $C = k_{0x} + k_{3x} + 2k_{0y} + k_{1y} + k_{2y}$. The pairwise error probability is

$$\begin{aligned} P_d^{(A)}(X,Y) &= \sum_{\Omega_A} u(m_c - m_e(k_{2x_1} + k_{1x_2})) \frac{X_1!}{k_{0x_1}!k_{1x_1}!k_{2x_1}!k_{3x_1}!} \frac{X_2!}{k_{0x_2}!k_{1x_2}!k_{2x_2}!k_{3x_2}!} \\ &\times \frac{Y!}{k_{0y}!k_{1y}!k_{2y}!k_{3y}!} \left(\frac{p_e}{3}\right)^{k_1+k_2+k_3} (1-p_e)^{k_0} \end{aligned} \tag{6.24}$$

where

$$\begin{aligned} \Omega_A = \{k_{ix_1}, k_{ix_2}, k_{iy}, i = 0,1,2,3 \mid \quad & 0 \le k_{ix_1}, 0 \le k_{ix_2}, \\ & 0 \le k_{iy}, i = 0,1,2,3, \\ & \sum_{i=0}^{3} k_{ix_1} = X_1, \sum_{i=0}^{3} k_{ix_2} = X_2, \sum_{i=0}^{3} k_{iy} = Y\}. \end{aligned}$$

Note that in the above equation, the summation constraint on k_{1x_2} and k_{2x_2} is identical. In another word, k_{1x_2} can assume the same range of values as the k_{2x_2}. Thus, we can exchange these two variables in (6.24) without affecting the value of $P_d^{(A)}(X,Y)$. The right hand side of the equation is not changed, except that the argument of m_e becomes $k_{2x_1} + k_{2x_2} = k_{2x}$. By definition of multinomial coefficients, it is not difficult to see that $P_d^{(A)}(X,Y)$ is equal to the right hand side of the $P_d(X,Y)$ in (6.22), which is based on the assumption all one-bit-error branches have 01. It is worth mentioning that the advantage of using Equation (6.22) is that the summation involves much fewer terms, thus much less computing time, than that in Equation (6.24).

Chapter 7

Suggestions for Future Work

The main objectives will be the study of coding and detection for frequency hopped spread spectrum communications. The major emphasis will be on slow frequency hop systems, especially using differential phase shift keying (DPSK) modulation scheme.

7.1 Slow Frequency Hopping Systems

For such systems, it remains to evaluate the performance when diversity is also employed, and when nonbinary DPSK is used. Both of these directions should provide performance improvement over the results found in Chapter 2. As well, the use of interleaving will reduce the number of erroneous symbols in a given RS codeword when a hop is jammed. It also allows a long hop length, which reduces the amount of lost data due to the phase reference bit. Some specific problems to be addressed are as follows:

1. Multi-symbol probability distribution of DPSK in Gaussian noise. This will facilitate the performance evaluation of coded systems in the presence of partial band noise jamming or multi-tone jamming plus thermal noise.

2. Multi-symbol probability distribution of DPSK in the presence of tone jamming. This will facilitate the performance evaluation in the presence of multi-tone jamming.

3. Block code system evaluation using M-ary codes. Robust techniques such as erasure generation and erasure correction decoding will be considered.

4. Study the use of constrained sequences to cancel interference and perform error correction.

5. Consider possible alternative convolutional coding techniques and corresponding decoding metric generation problem.

7.2 Interleaving

Building on Chapter 4, we propose to examine the complicated problem of analyzing the performance of coded systems using a particular finite interleaver. The performance degradation, when the designed burst length is exceeded, is worth investigating.

7.3 Coding and Diversity

The objective here will be to to investigate the trade-off between diversity and coding for systems, such as spread spectrum systems, where a low code rate is anticipated. Ultimately, the aim is to determine actual performance trade-offs in terms of signal-to-noise ratios and probability of error. Of more importance is the translation of the trade-offs considered here into an understanding of how it affects system performance on a variety of channels, such as the additive white Gaussian noise, Rayleigh fading and interference channels. Future work will consider these questions and attempt to determine guidelines for this trade-off.

7.4 Implementation of CODECs

We shall concentrate on implementation of practical error correcting codes using current technologies (e.g. VLSI gate array designs) and future technologies (e.g. artificial neural networks).

We shall study and develop new algorithms and/or architectures that take maximal advantage of the circuit regularity and parallelism afforded by VLSI technology. Of special interest, we shall concentrate on the cellular structure which allow cascadability of identical chips to form long codes.

Bibliography

[1] Bhargava, V.K., Blake, I.F., Gulliver, T.A., Li, G., Wang, Q., and Weeks, B., "Coding for frequency hopped spread spectrum satellite communications,"*Final Report prepared for the Department of Communications under SSC Contract No. 36001-8-3529-01-SS*, April 15, 1989.

[2] Wang, Q., Gulliver, T.A., and Bhargava, V.K., "Probability distribution of DPSK in tone interference and applications to SFH/DPSK," accepted for publication in *IEEE J. on Selected Areas in Communications.*

[3] Berlekamp, E.R., "The technology of error correcting codes,"*Proc. IEEE*, Vol. 68, May 1980, pp. 564-593.

[4] Simon, M.K., Omura, J.K., Scholtz, R.A. and Levitt, B.K., *Spread Spectrum Communications, Vol. II*, Rockville: Computer Science Press, 1985.

[5] Pawula, R.F., Rice, S.O., and Roberts, J.H., "Distribution of the phase angle between two vectors perturbed by Gaussian noise," *IEEE Trans. on Communications*, Vol. COM-30, No. 8, August 1982, pp. 1828-1841.

[6] Knuth, D.E., "Efficient balanced codes," *IEEE Trans. on Information Theory*, Vol. IT-32, No. 1, Jan. 1986, pp. 51-53.

[7] vanTilborg, H., and Blaum, M., "On error-correcting balanced codes," *IEEE Trans. on Information Theory*, Vol. IT-35, No. 5, September 1989, pp 1091-1095.

[8] Bhargava, V.K., Haccoun, D., Matyas, R., and Nuspl, P.P., *Digital Communications by Satellite*, New York: Wiley, 1981.

[9] Oppenheim, A.V., and Schafer, R.W., *Digital Signal Processing*, New York: Prentice-Hall, Inc., 1975.

[10] Clark, Jr., G.C., and Cain, J.B., *Error-Correction Coding for Digital Communications*, New York: Plenum Press, 1981.

[11] Ramsey, J.L., "Realization of optimum interleavers," *IEEE Trans. on Information Theory*, Vol. IT-16, 1970, pp. 338-345.

[12] Forney, Jr., G.D., "Burst-correcting codes for the classic bursty channel," *IEEE Trans. on Communications Technology*, Vol. COM-19, 1971, pp. 772-781.

[13] Forney, Jr., G.D., "Interleavers," *U.S. Patent No. 3,652,998*, March 28, 1972.

[14] Richer, I., "A simple interleaver for use with Viterbi decoding," *IEEE Trans. on Communications*, Vol. COM-26, 1978, pp. 406-408.

[15] "Interleaved coding for bursty channels," *Final Report, Phase I, NSF Grant No. ECS-8260180*, Cyclotomics Inc., Berkeley, California, April 12, 1983.

[16] Knuth, D.E., *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*, Reading, Massachusetts: Addison-Wesley Publishing Co., 1981.

[17] Dunscombe E., and Piper, F.C., "Optimal interleaving scheme for convolutional coding," *Electronics Letters*, Vol. 25, 1989, pp. 1517-1518.

[18] Darmon, M.M., and Sadot, P.R., "A new pseudo-random interleaving for antijamming applications," *Proc. of MILCOM'89*, Boston, Mass. November, 1989, pp. 1.2.1-1.2.5.

[19] Chase, D., "Code combining - a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. on Communications*, Vol. COM-33, No. 5, May 1985, pp. 385-393.

[20] Heller, J.A., "Sequential decoding: short constraint length convolutional codes," *JPL Space Programs Summary 37-54*, Vol. III, pp. 171-177; October-November, 1968.

[21] Ryan, W.E. and Wilson, S.G., "Convolutional coding over $GF(q)$ with application to frequency hopping channels," *Proceedings, 1987 Conference on Information Sciences and Systems*, pp.439-445.

[22] Larsen, K.J., "Short convolutional codes with maximal free distance," *IEEE Trans. on Information Theory*, Vol. IT-19, 1973, pp. 371-372.

[23] Odenwalder, J.P., "Optimal decoding of convolutional codes," *Ph.D. Dissertation*, University of California, Los Angeles, 1970.

[24] Wang, Q., Nicholson, R.D., and Onotera, L.Y., "Some practical issues in the design and application of a VLSI FEC chip," *International J. of Satellite Communications*, Vol. 7, No. 3, July - September, 1989, pp. 129-142.

[25] Kasami, T., Fujiwara, T., Takata, T. and Lin, S., "A cascaded coding scheme for error control and its performance analysis," *IEEE Trans. on Information Theory*, Vol. IT-34, No. 3, May 1988, pp. 448-462.

[26] Shaft, P.D., "Low-rate convolutional code applications in spread-spectrum communications," *IEEE Trans. on Communications*, Vol. COM-25, No. 8, August 1977, pp. 815-821.

[27] Ziemer, R.E. and Peterson, R.L., *Digital Communications and Spread Spectrum Systems*, NewYork: MacMillan, 1985.

[28] Wozencraft, J.M. and Jacobs, I.M., *Principles of Communication Engineering*, New York: Wiley, 1965.

[29] Massey, J.L., "Coding and modulation in digital communications," *Proc. of International Zurich Seminar*, 1974.

[30] Trumpis, B.D., "Convolutional coding for M-ary channels," *Ph.D. Dissertation*, University of California, Los Angeles, 1975.

[31] Gong, K.S., "Performance of diversity combining technique for FH/MFSK in worst case partial band noise and multi-tone jamming," *Proc. of MILCOM'83*, pp. 17-21, 1983.

[32] *The Programmable Gate Array Data Book*, Xilinx Inc., San Jose, C.A., 1988.

[33] Reed, I.S., and Solomon, G., "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, Vol. 8, June 1960, pp. 300-304.

[34] Lin, S. and Costello, D.J., Jr., *Error Control Coding*, Englewood Cliffs: Prentice-Hall, 1983.

[35] Michelson, A.M., and Levesque, A.H., *Error-Control Techniques for Digital Communications*, New York: John Wiley & Sons, 1985.

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

[37] Wang, C.C., Truong, T.K., Shao, H.M., Deutsch, L.J., Omura, J.K., and Reed, I.S., "VLSI architecture for computing multiplications and inverses in GF(2^m)," *IEEE Trans. on Computer*, Vol. C-34, August 1985, pp. 709-717.

[38] Bhargava, V.K., Le-Ngoc, T., Shayan, Y.R., and Sheikh, A.U.H., "Control coding systems for mobile communications with applications to Advanced Train Control System (ATCS) and other UHF/VHF mobile systems," *Transport Canada Publication No. 9596E.* Prepared by Binary Communications Inc. under DSS Contrcat No. 10SB.T8200-7-7516, January 1989.

[39] Bhargava, V.K., "Communications and Transportations", technical paper prepared for Transport Canada under their "Visiting Expert" program, July 1989.

Appendix A

Logic-Cell Array Implementation of a (31,k) Reed-Solomon Codec

This algorithm presents designs for a programmable gate array implementation of a user programmable (31,k) Reed-Solomon encoder-decoder. The programmable encoder and decoder algorithms were first implemented in the C programming language, using a Galois Field software development package written for this purpose. The control hardware of five independent modules was simulated in C; based upon the simulation programs a number of designs involving differing amounts of pipelining and different storage architecture and Galois Field bases were developed. One design is currently being implemented using the Xilinx 3000 series Programmable Gate Arrays.

A.1 Introduction

This Appendix examines a number of design options for programmable gate array (PGA) implementation of a (31,k) user programmable Reed-Solomon codec. A key consideration was maintaining a high level of modularity in the design process; this not only simplifies the task of making small changes in the design if necessary, but also allows the same design with minor alterations to be used for larger size Reed-Solomon codecs. Although only the (31,k) RS codec is described in this Appendix, C source code has also been written and tested for the (15, k), (63, k), (127, k) and (255, k) Reed-Solomon codecs. These RS codecs would be relatively easy to implement in PGA's or other hardware if the need arose.

The use of programmable gate arrays is in itself based on the need for a quick turn-around time in the design and testing of a circuit. The design is loaded from a computer terminal or from ROM onto the Xilinx chip; a change in design only requires changing the program loaded into the ROM, in contrast to a period of weeks and considerable more cost required for a custom integrated circuit. If high production of the PGA decoder was felt to be desirable, software exists[32] to convert between the files needed by Xilinx and the standard schematic capture format used in custom integrated circuit contruction.

Although both programmable RS encoders and decoders have been designed, only the decoder designs will be discussed in this appendix. The encoder design is a variation of the standard linear feed-back shift registers used in non-programmable RS encoders, and so is straight-forward. In the process of writing the C language implementations of the programmable RS codecs, C language software tools were developed for the Galois fields GF(2^m), where m = 4 to m = 8. These software tools find the Galois field elements generated by a primitive polynomial, produce the RS generator polynomials for a Galois field, produce Karnough mappings, and test the encoder/decoder programs by introducing random errors into the pipeline between the encoder and the decoder programs.

Section 2 provides background on the algorithm used to decode Reed-Solomon codes. Section 3 gives an outline of some of the options considered in designing the (31,k) codec: the choice of Galois Field basis, the internal bus and register design, the pipelining options and the external logic and memory. Finally Section 4 describes the design that is being implemented at the time of writing.

A.2 Background

A.2.1 Background

Reed-Solomon codes are a class of cyclic random error-correcting non-binary block codes discovered by Reed and Solomon in 1960 [33]. The symbols of a Reed-Solomon code are Galois field $GF(2^m)$ elements, where m is a positive integer. An RS code with symbols from $GF(2^m)$ has a block length of 2^m-1 symbols, and can be written to correct t_e errors, where $1 \leq t_e \leq 2^{m-1}-1$. A t_e error correcting RS code of block length $n = 2^m-1$ has $2t_e$ parity check and $n - 2t_e$ information symbols per block, and a minimum distance
$d = 2t_e+1$. Decoding an RS code involves four basic logical modules[34][35]:

(1) computing the syndrome components S_i, $i = 1,2,...,2t_e$.

(2) determining the error-location polynomial $\partial(X)$.

(3) finding the roots of the error-location polynomial $\partial(X)$.

(4) finding and correcting the error at each error location.

(1) The symbols r_k in the received block of an RS code are the coefficients of the received vector

$$r(X) = r_0 + r_1X + r_2X^2 + ... + r_{n-1}X^{n-1}$$

where k is the position of the symbol rk in the received block. The received vector r(X) is the sum:

$$r(X) = t(X) + e(X)$$

where t(X) is the transmitted vector and e(X) is the vector of errors introduced during transmission. The $2t_e$ syndrome components are found by substituting α^i into the received vector r(X) for $i = 1,2,...,2t_e$:

$$S_i = r(\alpha^i) = t(\alpha^i) + e(\alpha^i) = e(\alpha^i) \qquad (A.1)$$

where α^i is a primitive element of $GF(2^m)$.

(2) The decoder must find the locations of the the errors introduced by transmission; that is, the non-zero coefficients of e(X). Directly solving the system of equations (A.1) is difficult; an alternative approach is to introduce a polynomial, the error- location polynomial $\partial(X)$, whose roots are the locations of the transmission errors.

$$\partial(X) = (1+\beta_1 X)(1+\beta_2 X)\ldots(1+\beta_\mu X)$$

$$= \partial_0 + \partial_1 X + \partial_2 X^2 + \ldots + \partial_\mu X^\mu$$

where β_i is the location of the ith error and μ is the number of errors introduced during transmission. A number of methods exist for determining the error location-polynomial, $\partial(X)$ [34]; Berlekamp's iterative method [36] was used in the programmable (31,k) Reed-Solomon decoder.

(3) The roots β_i of the error-location polynomial must be found. This can be done either by the Chien [35] search or by substituting all the elements α^j in $GF(2^m)$ into the error locator polynomial and noting the elements which give $\partial(\alpha^j) = 0$.

(4) Finding the errors at the error locations requires solving the equations:

$$S_k = Y_1\beta_1 + Y_2\beta_2 + \ldots + Y_\mu\beta_\mu \quad \text{where} \quad k = 1, 2, \ldots, 2t$$

for the μ errors $Y_1,\ldots,Y_\mu$. An easy way of doing this is to first find the function

$$Z(X) = 1 + (S_1 + \partial_1)X + (S_2 + \partial_1 S_1 + \partial_2)X^2 + \ldots$$

$$+ (S_\mu + \partial_1 S_{\mu-1} + \partial_2 S_{\mu-2} + \ldots + \partial_\mu)X^\mu$$

The μ errors Yi can then be found using

$$Y_i = \frac{Z(\beta_i^{-1})}{\prod_{\substack{j=1 \\ j\neq i}}^{\mu}(1+\beta_i\beta_j^{-1})}$$

The transmitted vector t(X) can then be found by adding the error vector e(X) to the received vector r(X)

$$t(X) = e(X) + r(X).$$

A.2.2 The (31,k) RS Decoder Algorithm

Based on the above formulation, an algorithm was written to code and decode a t-error correcting RS code, where t is user programmable and $1 \leq t \leq 2^{m-1}-1$. The algorithm consists of logical modules (1) to (3) as given in the introduction, plus two logical modules from logical module (4) above which determine Z(X) and find and correct the transmitted errors. Each logical module of the algorithm is dependent upon earlier logical modules for intermediate results, but runs independent of earlier logical modules once those intermediate results are received. This independence allows each logical module to be implemented on a separate physical module.

The programmable RS decoding algorithm was initially written and tested using the C programming language. After testing the algorithm, C code was written simulating the hardware controllers needed in the PGA implementation of the programmable RS codec. All references to the number of algorithm operations and bit rate refer to the hardware implemented algorithm.

A.3 Implementation Options

Implementation of the algorithm requires decisions to be made on:

(1) which basis to use to represent the Galois field and the design of the Galois field arithmetic units.

(2) the number of bus lines and registers to provide for logically independent operations.

(3) the amount of pipelining to be used.

(4) the external logic and memory needed to co-ordinate a pipelined codec.

These decisions are made on the basis of how they will effect the trade-off between speed and hardware requirements. These decisions are not independent; decisions made on different aspects of the implementation effect each other.

A.3.1 Basis and Galois Field Arithmetic

The Galois field elements can be represented in either vector or power notation. For example, the element α^{27} from GF(2^5) may be represented as the vector:

$$\alpha^{27} = 1\alpha^0 + 1\alpha^1 + 0\alpha^2 + 1\alpha^3 + 0\alpha^4$$

$$= (1\ 1\ 0\ 1\ 0)$$

or as the integer

$$27 = (1\ 1\ 0\ 1\ 1).$$

The advantages and disadvantages in a representation lie in the implementations they allow for Galois field arithmetic, and in the complexity of the hardware needed to implement the decoder in the representation. The comparison of vector and power representation implementations of Galois field arithmetic is dependent on the size of the Galois field, and on the medium on which the arithmetic unit is implemented. In this appendix only single step Xilinx[32] implementations of arithmetic on the Galois fields GF(2^5) will be discussed. The number of configurable logic blocks (CLB's) of the LCA needed to implement arithmetic on the Galois field GF(2^5) for vector and power representations is given in Table A.1.

Galois Field Adders

In the vector representation, Galois field addition is simply bit-wise integer addition modulo 2. On the Xilinx 3000 series LCA[32] a vector representation adder

requires one configurable logic block for every two bits of the field size, plus input registers, for a total of 8 CLB's for a GF(2^5) adder. The easiest implementation of power representation addition on GF(2^5) on the Xilinx 3000 series LCA is to translate from power to vector representation, add mod 2, and translate back to the power representation. The five function logic of the Xilinx 3000 series LCA allows all of this to be done in the input and output registers, and requires 18 CLB's.

Galois Field Multipliers

Choosing the normal basis for the vector representation allows the Massey-Omura[37] multiplier to be used for the Galois field multiplication. For the GF(2^5) multiplier, the least-complex parallel Massey-Omura multiplier requires 20 CLB's, plus 5 CLB's for the input registers. Galois field multiplication using the power representation is equivalent to integer addition modulo 2^m. For the GF(2^5) multiplier 10 CLB's plus 5 CLB's for the input registers are needed.

Galois Field Inverter

Inversion over GF(2^5) for the normal basis representation normally requires either repeating a shift and multiply over GF(2^5) four times[37], or a parallel inverter requiring four Galois field multipliers. However the five function logic of the Xilinx 3000 series LCA's may be used to translate directly from an element to its inverse; the total space requirement for the normal basis GF(2^5) inverter is 8 CLB's. Inversion over GF(2^5) in the power representation for elements other than α^0 only requires inverting each bit of the integer representation; an extra CLB is required to return α^0 when α^0 is input. The power representation GF(2^5) inverter requires in total 4 CLB's.

Representation	GF Adder (CLB's)	GF Multiplier (CLB's)	GF Inverter (CLB's)
Normal	8	25	8
Power	18	15	4

Table A.1: CLB requirements for GF Arithmetic

The Galois field arithmetic operations needed by the RS decoding algorithm are addition, multiplication and inversion. From Table 3.1 it can be seen that the power representation requires about 10% fewer CLB's to implement the Galois field arithmetic operation than the vector representation. The power representation also has the additional advantage of allowing the Galois field elements to be used as incremental counters; this simplifies the design of some parts of the decoder.

A.3.2 Memory and Bus Lines

During the decoding process the algorithm requires storage space to hold results of intermediate calculations. The intermediate results include results used only in a given module and results to be used in modules after the module in which they are calculated. If there is pipelining the received RS code blocks must also be stored from the first to the last stage in the pipeline. The storage space in bits of required by each module is given in Table A.2; the first row gives the storage space needed to carry out the calculations of the module itself, the second row includes the storage space required from previous modules.

Module	Syndrome	Delta	Location	Z(X)	Error
Storage (bits)	330	410	100	95	100
Total Storage (bits)	330	565	185	415	335

Table A.2: Storage Space Requirements by Module

Dedicating CLB's to each intermediate variable simplifies the design and speeds up the decoder, but at the cost of space. The implementation of the 5-bit buses required by the GF(2^5) can be done either as common buses using Xilinx's tri-state capabilities, or as buses between each set of communicating decoder components or storage block. It was found that the latter scheme allowed for tighter routing.

A.3.3 Pipelining

The independence of the five modules in the decoding algorithm allows a trade-off to be made between the decoder speed and the amount of hardware needed for implementation.

Speed

The number of operations each module requires to complete as a function of the number of correctable errors is given in Fig.A.1. Not only does each module require a different number of operations to complete, but the number of operations varies among modules either linearly or as a square with respect to the number of operations to completion.

Since the decoder must finish decoding each received block in the same amount of time, the pipelining should be made so that each stage in the pipeline takes approximately the same number of modules. The five pipelining options which meet this requirement are given in Table A.3.

Pipeline Option	Logical Modules
no pipelining	• syndrome & delta & location & z(X) & error correction
two stage pipelining	• syndrome & delta • location & z(X) & error correction
three stage pipelining	• syndrome & delta • delta • location & z(X) & error correction
four stage pipelining	• syndrome • delta • location& z(X) • error correction
five stage pipelining	• syndrome • delta • location • z(X) • error correction

Table A.3: Pipelining Options

The overall number of decoder operations for each pipelining option as a function of correctable errors is given in Fig.A.2.

Hardware

Each module uses a different number of CLB's to carry out it's function. The CLB requirements are given in Table A.4; the requirements include CLB storage of results calculated in the module, Galois field arithmetic and control logic, but not storage of arrays passed to other modules. Routing limitations may give rise to higher CLB requirements than those listed in Table A.4.

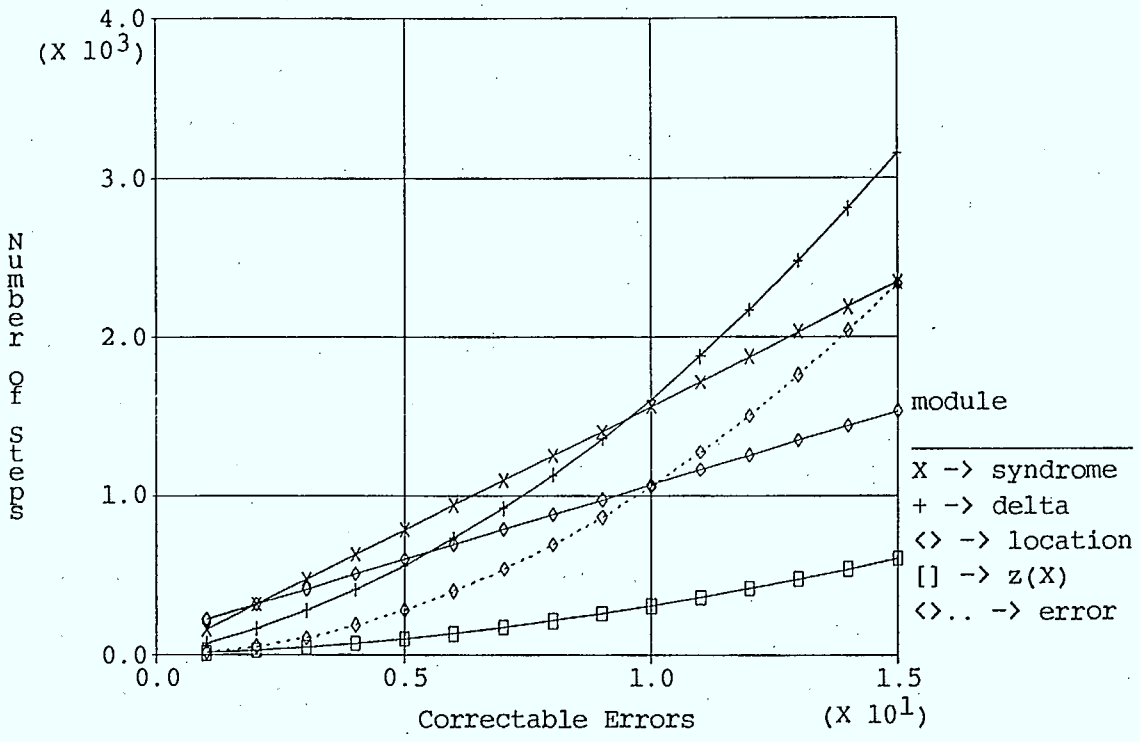


Fig. A.1: Module Steps vs Correctable Errors

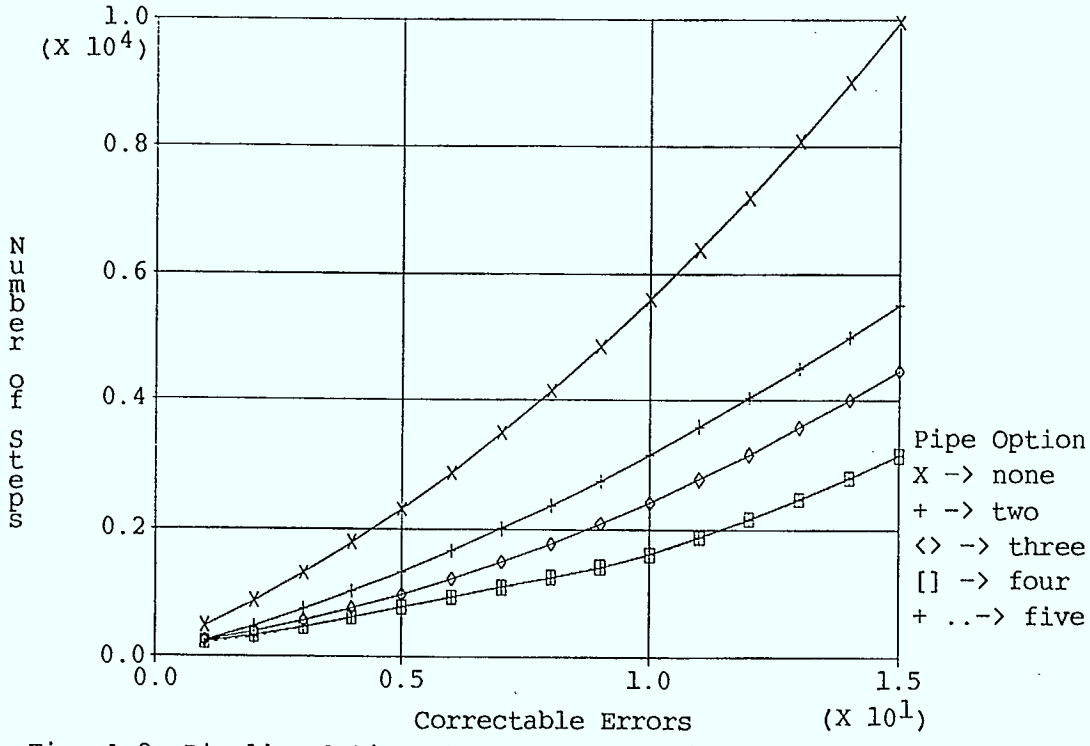


Fig. A.2: Pipeline Option Steps vs Correctable Errors

Module	Syndrome	Delta	Location	Z(X)	Error
CLB requirements	90	320	75	90	160

Table A.4 : CLB requirements by Module

The received block for each stage in the pipeline must also be saved and passed on from the first to last stage, as well as some intermediate calculations. Hardware is minimized if modules are combined which share intermediate calculations. The coordination of the stages and the shared intermediate calculations also becomes more complex as more stages of pipelining are used.

A.3.4 External Logic and Memory

Each of the five stages requires as its input either the received codeword or an array of intermediate values calculated in previous stages; this dependency is shown in Fig.A.3. The received codeword and the syndrome are arrays of 155 bits (31 Galois field elements), the size of each of the other arrays is 75 bits. If no pipeling is used these values can be passed on by using the same memory for each stage. Because the RS decoding algorithm requires both the received codeword and each intermediate value array to be completed before being passed on, each level of pipelining introduced increases both the number of arrays that must be passed on and the number of copies of each passed array that must be stored.

Two strategies for passing arrays between stages have been considered. The first strategy passes the data between the storage elements associated with each stage. Consider the case of five stage pipelining as an example. If some array A[X,t] is calculated in the first stage and used in the fourth stage, the copy of A[X,t] generated at time t = 1 will reach the fourth stage three stage shifts later. This means thatwhen the array A[X,t =1] will be used in the fourth stage there will be an array A[X, t =4]

being calculated in the first stage and two arrays, A[X, t =2] and A[X, t =3] being stored for use in the fourth stage.

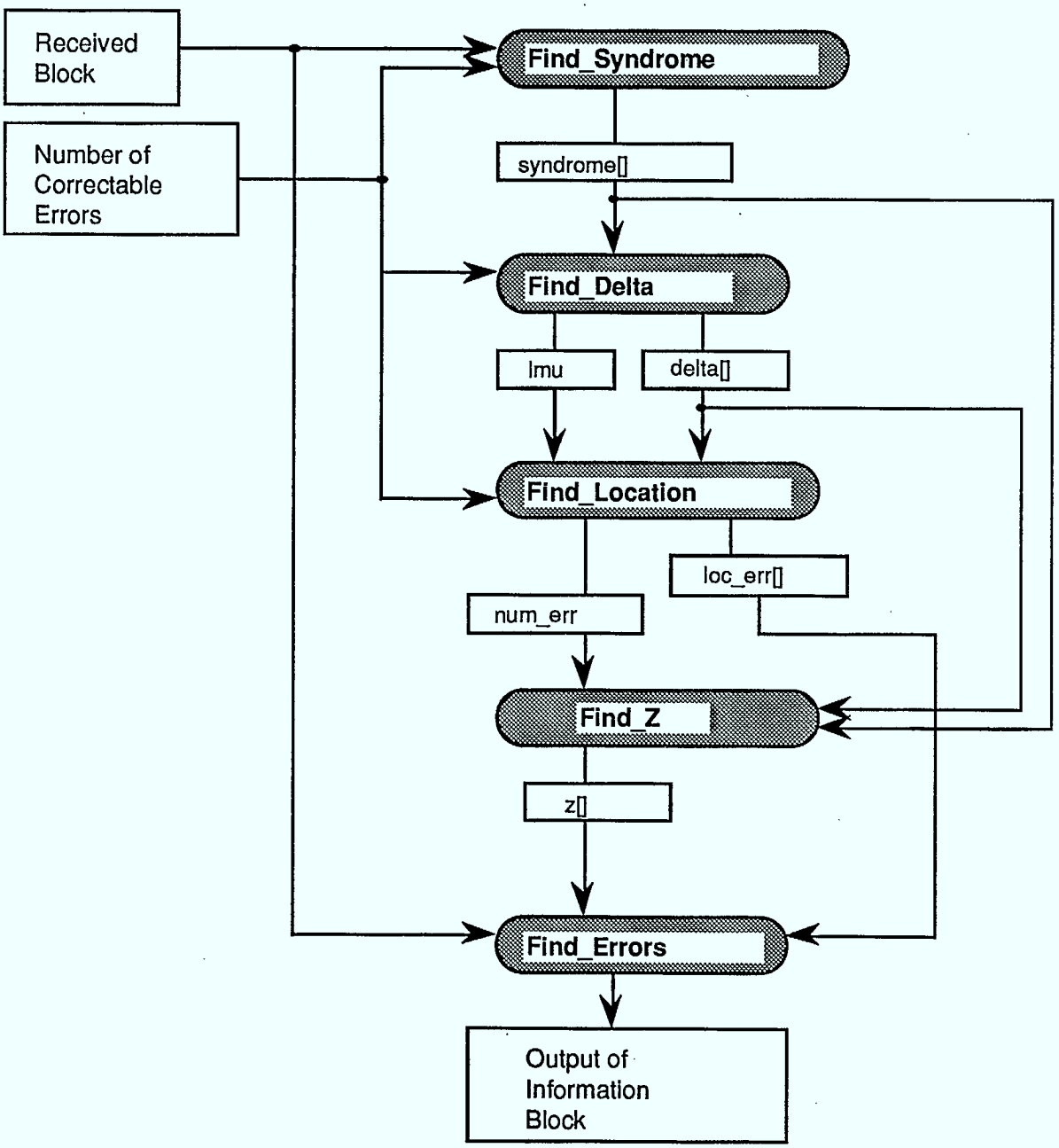


Fig.A.3: Intermediate Array Passing

Since passing an array of 75 or 155 bits in one clock cyclerequires too many input/output pins to be practical, the modules must shift the array in one stage

before the array is to be used to ensure the complete array will be available when required. In some cases this requires the module to have two copies of the array; the copy being used A[X, t =0] and the copy being shifted in A[X, t =1].

Since only the arrays needed by a particular stage are passed to that stage, this strategy minimizes the amount of storage needed. It has the disadvantage however of requiring relatively complex logic to control the passing process, which reduces the modularity of the design and makes design alterations more difficult.

The first strategy requires either shift registers or Xilinx chips with a very large number of input/output pins to store and pass the arrays; either option is expensive to implement in Xilinx, and requires a large number of chips if implemented in standard register chips. The flow of the arrays in the case of the five stage pipelined RS decoder is shown in Fig.A.4.

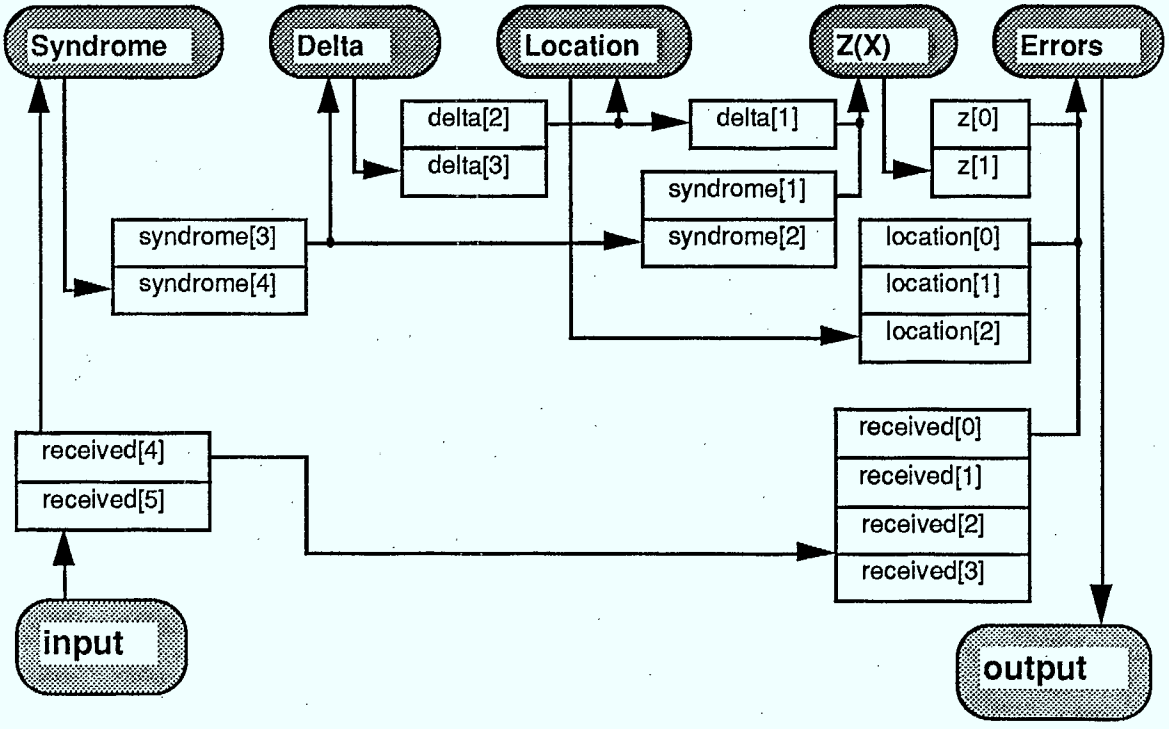


Fig.A.4: Direct Array Passing for a Five Stage Pipelined RS Decoder

The second strategy is to pass a pointer to the data associated with each stage. Since one copy of each array used in the decoding algorithm must exist for each stage, this strategy requires more memory space than the first option. However, it maintains the modularity of the design and keeps the control logic simple. The arrays may also be stored in relatively inexpensive external 8-bit RAM; because the sequence of steps in each module varies with the number of errors to be corrected, it is not possible to time RAM access among the modules. Instead each module has its own RAM thus allowing the same memory design to be used with little change for Galois fields of size 2^8 and smaller. An example of the switching network is shown in Fig.A.5.

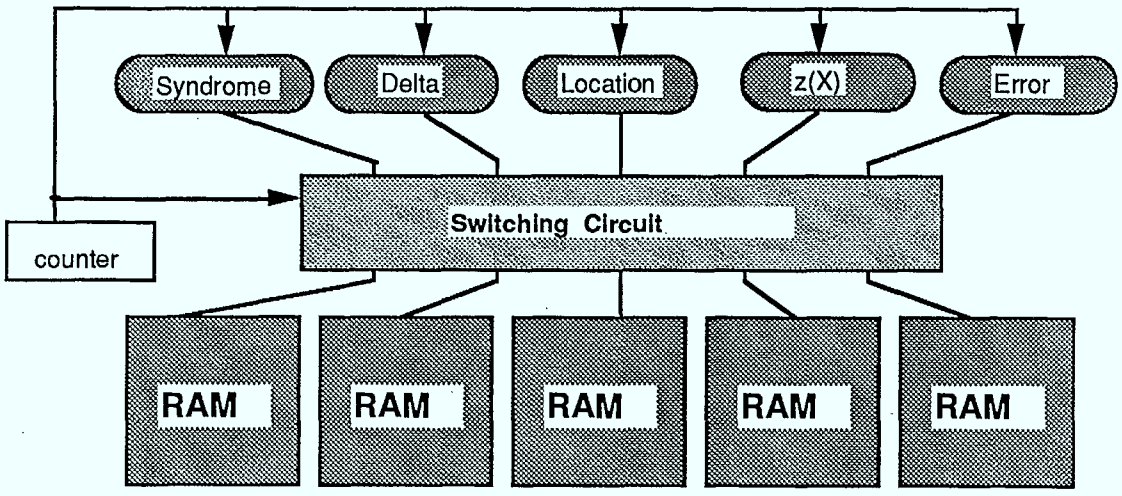


Fig.A.5 External RAM Storage of Arrays for a Five Stage Pipelined RS Decoder

A.4 Implementation

Implementation of some of the stages of the (31,k) RS codec on the Xilinx 3000 Series Programmable Gate Arrays has shown the Xilinx clock speed to be about 3.5 MHz for this design. This clock speed gives the bit rates for each of the pipeline options given in Section A.3 as a function of correctable errors as shown in Fig.A.6. The bit rate increases as more pipelining stages are added until four stage pipelining

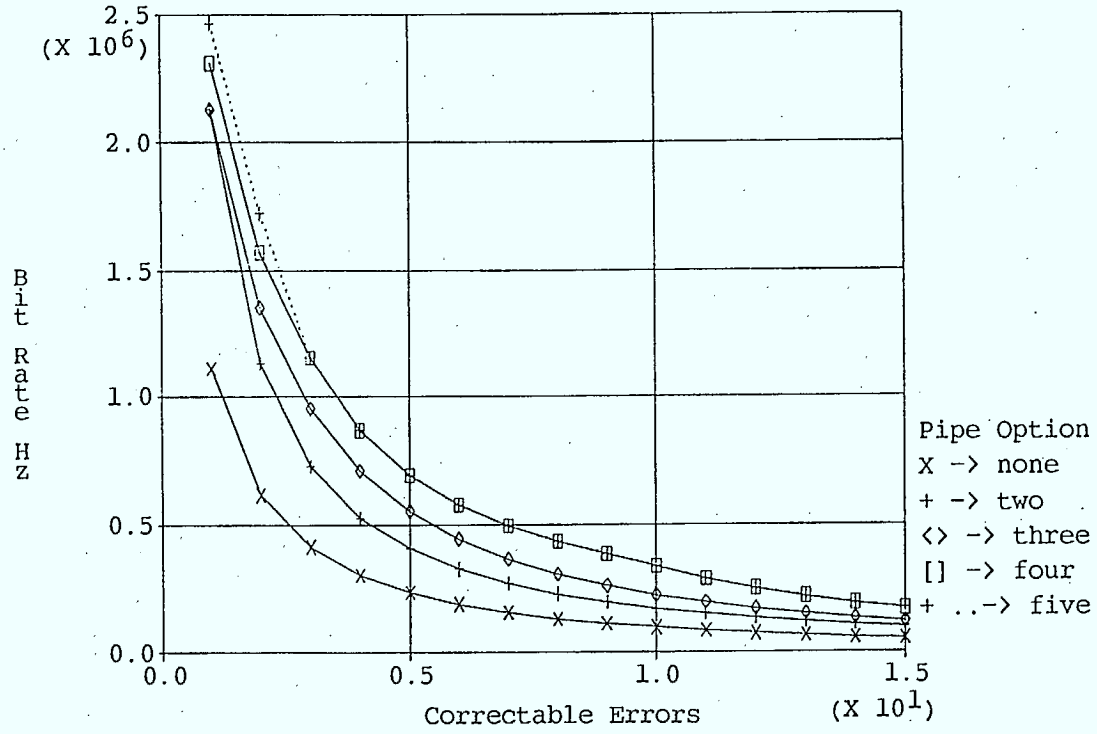


Fig. A.6: Pipeline Option Rates vs Correctable Errors

is reached; adding a fifth pipelining stage only increases the bit rates for low rate codes. However, a fifth stage of pipelining may be desirable in order to increase the modularity of the design.

The number of correctable errors that an incoming received word is coded to correct must also be stored along with the received word if to allow the user to program the (31,k) RS codec without interrupting the transmission. This is done in an array passed along each stage with the received word. The Xilinx 3000 Series PGA comes in chips of five different sizes[32]. The design options can be either implemented on one or two larger chips, or with a separate smaller chip for each module if ease of modification or expansion is desirable. The module designs are saved in software in the design's .LCA files [32]; therefore it is relatively easy to design the options on separate chips and then combine the smaller chip designs onto the larger chips. Xilinx reportedly will soon be releasing the Xilinx 4000 series PGA's, with denser on chip routing, and faster internal switching. If the Xilinx 3000 series PGA's will be upwardly compatible with the 4000 series PGA's, the (31,k) RS Decoder may be implemented with little alteration on the Xilinx 4000 series PGA's, increasing the bit-rates shown in Fig.A.6.

No-Pipelining Option:

Since only one logical module may be carried out in each clock cycle, only one unit of each of the Galois field arithmetic units is needed, and all the storage can be saved in either in registers within the Xilinx chips or preferably, in one external RAM. Because most of the chip space in the (31,k) RS codec is in storage of space and the Galois field arithmetic unit, the no-pipelining option is relatively easy to implement, and requires considerably less hardware than the pipe-lined options. The bit rate of the no-pipelined option is about one-quarter that of the fully pipelined codec (see Fig.A.6).

Two Stage and Three Stage Pipelining Options:

Since only three arrays (see Fig.A.3) need be passed a distance of one stage, directly passing the arrays is the easiest strategy to implement in the two stage pipelining option. Three arrays are also passed in the three stage pipelining option, one of which is passed a distance of two stages; directly passing the arrays is then also the best strategy for storing intermediate calculations. The bit-rate of the two stage option is about half that of the five stage pipelined option, while the bit rate of the three stage option about two-thirds that of the five stage pipelined option. If modification is an important consideration, external RAM should be used instead of directly passing the arrays as it simplifies the task of altering the control logic.

Four Stage and Five Stage Pipelining Options:

External RAM should be used for both the four stage and the five stage (fully) pipelining options to maintain modularity and reduce the complexity of the control logic. The bit-rates of the four stage option and the fully pipelined option are the same for much of the range of correctable errors. The fully pipelined option can be built with each module on its own small Xilinx chip, making it the easiest to modify and test. The module designs of the fully pipelined option may later be transformed into one of the lower level pipelining options with minimal reworking.

A.5 Applications of the CODEC

The (31, k) CODEC can be modified to form a (16, 12) CODEC which is a standard for Advanced Train Control systems[38]. This may be accomplished by considering a (31, 27) code and then shortening it by 15 symbols to obtain a (16, 12) code. The data rate requirement is 4.8 kbps with future upgrades to 9.6 and 19.2 kbps.

With k = 15, we have a (31, k) CODEC which is a standard for Joint Tactical Information Distribution System (JTIDS)[27]. The data rate requirement for this system is 57.6 kbps.

The CODEC is ideally suitable for Meteor Burst Communication Systems[39]. Such systems have recently been proposed for data communications from trucks to dispatch centers.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

University of Victoria,Department of Electrical & Computer Engineering,P.O. Box 1700,VICTORIA, B.C. V8W 2Y2

2. SECURITY CLASSIFICATION (overall security classification of the document. including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Coding for Frequency Hopped Spread Spectrum Satellite Communications (U).

4. AUTHORS (Last name, first name, middle initial)

Bhargava, Vijay K.; Wang, Qiang; Li, Gang; Gulliver, T. Aaron; Dravnieks, Olaf.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
April, 1990	140	39

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report (Final)

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Communications Research Centre
P.O. Box 11490, Station "H"
OTTAWA, ONTARIO K2H 8S2

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)
041LG	36001-8-3529/01-SS

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
Technical Report No. ECE 90-1	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(XX) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availabilty (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

The performance of Reed-Solomon (RS) error correcting codes with slow frequency hopped (SFH) differential phase shift keying (DPSK) signalling is analysed and evaluated under worst case partial band noise and worst case multitone jamming. A representative set of the performance curves is shown. Based on these results, recommendations on the choice of RS code parameters is given.

Two in-hop jamming cancellation schemes for SFH/DPSK systems are proposed. One scheme is based on balanced coding; the other one uses notch filter to cancel jamming tone. The performances of both schemes are illustrated. It is shown that both schemes can work well under some conditions.

Basic principles and techniques for designing interleavers are presented. Block, convolutional and the more recent helical interleavers are considered. Certain questions on the trade-off between diversity and coding for spread spectrum systems, where a low code rate is anticipated are considered.

An error correction scheme is presented, for an M-ary symmetric channel (MSC) characterized by a large error probability p_e. The value of p_e can be near, but smaller than, $1\text{-}1/M$ for which the channel capacity is zero. Such a large p_e may occur, for example, in a jamming environment. Monte Carlo simulation results are presented. For the binary symmetric channel (BSC), it is shown that the overall code rate is larger than $0.6R_0$,where R_0 is the cutoff rate of the channel. For BSC and a large m, a method is presented for BER approximation based on the central limit theorem.

Logic-cell array implementation of a (31,k) "programmable" Reed Solomon CODEC is presented as an Appendix.

Suggestions for future work include investigation of coding and detection for slow frequency hop systems using DPSK, robust techniques for generation of erasures, use of constrained sequences to cancel interference and perform error correction, analysis of coded systems using finite interleavers, trade-offs between coding and diversity and implementation aspects of CODECS.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Frequency hopping
Spread Spectrum
Error-correcting codes
Satellite communications
M-ary FSK modulation
DPSK modulation
Tone cancellation
Interleaving
Diversity

CODING FOR FREQUENCY HOPPED SPREAD SPECTRUM SATELLITE COMMUNICATIONS : FINAL REPORT : PERIOD COVERED : APRIL 1 1989 TO MARCH 31 1990