INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

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

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Improving the Iterated MAP Decoder by Remapping

by

Jason A. Cooke

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of

Master of Applied Science
Ottawa - Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

Acknowledgements

I would like to take this opportunity to make known my sincere appreciation for the contributions of my supervisors, Dr. Peter Galko, of the University of Ottawa, and Dr. John Lodge, of the Communications Canada. Without their guidance and timely advice this thesis would not have been completed.

I also wish to express my heartfelt gratitude to the staff of the Canadian Bureau for International Education, especially Ms. C. Taha, and the Ministry of the Public Service in Jamaica, especially Ms. P. Gallimore, for providing the scholarship which has enabled the furthering of my education.

It is also incumbent on me to say a big thank you to my family here in Canada, who have made my stay here that much more comfortable. Thanks are also due to Susanna for her patience and support.

Finally, I dedicate this work to my mother and father, for everything.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
APP	*a posteriori* probability
MAP	Maximum a posteriori
ML	Maximum Likelihood
m.l.s.e.	Maximum likelihood sequence estimation
m.s.e.	Mean square error
VA	Viterbi Algorithm
SOVA	Soft Output Viterbi Algorithm

List of Symbols

Symbol	Description
A	Set of state transitions for which the coder output is zero.
α_j	j^{th} element of GF(q)
$\alpha_t(m)$	Forward trellis branch transition probability from node m
α_x	Event that there are x errors in the codeword
B_r	Counted bit error rate
$B(r)$	Polynomial function fitted to the collected error statistics, as a function of reliability estimate
$\beta_t(m)$	Reverse trellis branch transition probability from node m
C	Error control code used in digital transmission system
c_i	i^{th} bit in a codeword
c_l	Confidence level
d	Event that a bit is in the 2-D error subfield
D_t	Data input to encoder at time index t
$D_{i'}$	Sequence of binary k-vectors input to encoder from time index i to i'
e_i	Event that an error occurs in the i^{th} bit
$\overline{e_i}$	Event that the i^{th} bit is not in error
E_w	Weighted mean square error
$E(r)$	Probability of bit error given the value of the bit's reliability estimate
$GF(q)$	Galois Field over q
$\gamma_t(m',m)$	Trellis bracnh transition probability from node m' to node m at depth t.
H	Parity check matrix of block code
h_m	m^{th} column of H
k	Number of information bits in a codeword

Symbol	Description
$L_t(k)$	Iterated MAP decoder output after k iterations
M_t	Subset of $(n-k)$-tuples in list of nodes at depth t
n	Number of bits in a codeword
n_e	Number of errors observed
n_k	Event that k bits in an n-bit codeword are in error
$P'(r)$	Remapping function derived from $B'(e)$, as a function of reliability estimate
$P[x_i=0\|Y;C]$	Form of reliability estimator used in this work. Represents the probability that a given bit in the received codeword is zero given the received codeword and the structure of the code
$P_k[x_i=0\|Y;C]$	Reliability estimate of a given bit after the k^{th} application of the MAP algorithm.
p_e	Probability of bit error resulting from transmission over an AWGN channel using antipodal modulation
$P[e_b]$	Probability of bit error
r_i	Reliability estimate of i^{th} bit, in the form of $P[x_i=0\|y_i]$
r_i	Value of the reliability estimate of the i^{th} bit
$\bar{r}_i$	Complement of the reliability estimate of the i^{th} bit, i.e. $\bar{r}_i = 1 - r_i$
$s_m(t)$	m^{th} $(n-k)$-tuple in list of nodes at depth t
S_t	State of trellis at time t
$_{t-1}S_{t'}$	Sequence of state though the trellis corresponding to $_tD_t$
σ	Standard deviation produced by the AWGN channel
$\sigma_t(m',m)$	State transition probability at time t
t	Time index or depth of node in a trellis
t	Event that an error bit arrangement of type t occurs
u	Event that a bit is in the 1-D error subfield

Symbol	Description
$X_t(k)$	Binary n-vector produced by coder at time t during channel input sequence corresponding to the k^{th} path through the trellis
$_iX_t(k)$	Channel input sequence corresponding to the k^{th} path through the trellis
X	Transmitted codeword
x_i	i^{th} bit in transmitted codeword
Y	Received codeword
y_i	i^{th} bit in received codeword
y	Degree of interleaver
Y_t	Real valued n-vector output by the channel corresponding to channel input sequence $X_t(k)$
$_iY_{t'}$	Channel output sequence produced by $_iX_t(k)$
z	Event that a bit is in the No-Error subfield
z_i	Event that the i^{th} bit is in the no-error subfield

Abstract

In this thesis, we present a method of rectifying a deficiency intrinsic to iterative MAP decoding. The decoding of multidimensional product codes using an iterated symbol-by-symbol maximum *a posteriori* (MAP) decoder was recently shown to have good performance. Although iterating the MAP process was found to improve the BER performance, it causes overly optimistic estimates of bit reliability to be output. This inaccuracy reduces the usefulness of the iterative MAP decoder, when it is used as the inner decoder in a concatenated coding scheme. We investigated several modifications to the basic MAP algorithm in an attempt to improve the accuracy of the reliability estimates. The most effective approach found was one termed "remapping" in which the reliability estimates output by the MAP process are mapped to more realistic values using functions generated from error statistics. Remapping was found to significantly improve the accuracy of reliability estimates output by the iterative MAP decoder. However, this improvement comes at a small cost in BER performance.

Chapter 1

Introduction

At the present time, industry and society are becoming increasingly dependent on communications as a means of improving their efficiency. Devices which were once considered to be the playthings of the wealthy have come to be seen as essential business tools. One need look no further than the office of the average small business to see that fax machines and computers with fax modems have become commonplace. The businessman of today feels cut off without his cellular phone and pager. The question of whether or not it is desirable–on a personal level–to be able to be contacted at any time is another matter. The reality is that this capability is now considered to be an essential ingredient for success in the increasingly competitive business environment. Portable laptop computers have spawned a need for the ability to link them to the computer at the office, thus the company networks must have a dial-in access capability. Telecommuting has become a viable alternative for a few and, as communication capabilities improve, the number of people working from home will increase.

All the above-mentioned activities and devices rely on transmission of data from one point to another. This data can take many forms and be transmitted over various media: it might be data from a fax machine over telephone lines; voice data from a cellular phone through the air; a dissertation being e-mailed over the Internet, passing through, among other media, optical fiber lines; etc. The advantages of transmitting all this data in digital form is well recognized. The fundamental reason for this is that, in digital transmission, all the information is reduced to a sequence of discrete symbols whose values are from a small set of possibilities, thus making it easier to decide what was transmitted. This contrasts with analog transmission in which the task of deciding between an infinite number of possible transmitted signal values is confronted. Practically, of course, the possibilities are chosen from within a (hopefully) narrow range.

Digital transmission often uses the "error rate" as an important indicator of the quality of the transmission system. Error rate is simply a measure of the frequency with which errors occur during transmission of the information from the information-source to the information-sink; it is usually expressed as the number of symbol errors per number of transmitted symbols. Different services require different levels of transmission quality. For example, the error rate tolerable in a transmission of speech is orders of magnitude greater than that for a transmission of computer data in which it is mandatory that virtually every bit be received correctly. In using digital transmission techniques, a number of possibilities exist which allow for tailoring the transmission schemes to the transmission environment, and balancing between the cost of any design and its performance. One method of improving the quality of transmission–not unique to digital transmissions–is to increase the power of the transmitted signal. Increasing the transmission power, however, increases the cost of operating the transmission system due to the increase in electric power consumption. In addition, more powerful (and thus more expensive) electronics are required to produce the greater transmission power.

There is another technique which can be used to improve transmission quality–error control coding. Error control coding is a technique that works by using the addition of redundancy to the transmitted data to detect and/or correct some of the errors that occur during transmission. The degree to which errors can be detected or corrected is a reflection of the power of the code, in that the more powerful the code, the greater the proportion of errors it can detect and/or correct. In general, error-controlling capability is increased through greater redundancy in the encoded data stream. However, the number of transmitted symbols per symbol of source data also increases such that we generally need a higher transmission rate if we are to maintain the same rate of information transfer. We therefore see that the use of error control coding usually results in an increase of the bandwidth required for transmission of information. Given that the bandwidth allocated for any given application is usually restricted, we cannot simply add redundancy at our whim, but rather, must balance the error rate considerations against those of the

bandwidth or power restriction. Therefore, in error control coding, we are typically trying to find coding techniques which afford maximum reductions in error rate with minimum increase in bandwidth requirements.

In choosing a decoder, several factors must be taken into account. First, the cost of the decoder is directly related to the complexity of the decoding algorithm which must be implemented since a more complex implementation has greater hardware requirements. Secondly, the delay introduced in converting the transmitted information from its encoded form to its original form must also be considered. The amount of delay tolerable will depend on the application. For example, if the application is two-way voice transmission, an end-to-end delay greater that 400ms is regarded as unacceptable to the average telephone customer [1]. The decoding delay must therefore be considered along with other delay contributors in order to arrive at a final design.

Several coding schemes have been proposed each having its own unique properties and suitability to specific transmission environments. One such scheme is that of concatenated coding, in which the output of one coder (termed the outer code) is fed to the input of another coder (termed the inner code). When decoding, the inner code is decoded first, followed by the decoder for the outer code. Concatenated coding was initially proposed as a method of constructing long block codes, with various good properties, from shorter ones. The resulting code is more powerful than the short codes by themselves, and does not require the level of decoding complexity usually associated with long block codes. The concept has been extended to a number of variations on the types of channel coding/decoding schemes employed in the inner and outer coder/decoders (codecs). One variation is the concatenation of a convolutional code and a Reed-Solomon code, which is decoded using a Viterbi decoder followed by a Reed-Solomon decoder [2].

A possible approach to decoding concatenated codes is to apply hard decision decoding to each level of the code. However, it has been found that the performance of the concatenated code can be significantly improved by not making entirely firm decisions in

decoding the inner code, and then passing these "soft decisions", which include an indication of the reliability of the soft decision, to the outer code decoder [3]. The symbol-by-symbol MAP algorithm–first presented by Bahl *et al* [15]–is one example of a decoding scheme that is capable of providing soft decisions to the outer decoder. The soft decisions which are passed to the outer decoder are in the form of estimates of the probability of individual bits in a code block being zero or one.

It was shown in [5] that the approach of concatenated coding can be extended using more than just two layers of coding to produce so-called "multi-dimensional product codes" from simple block codes. It was also demonstrated that such codes could be decoded very practically in a close to optimal fashion using an algorithm referred to as the iterated maximum *a posteriori* (MAP) filter. The iterated MAP "filter" is based on the symbol-by-symbol MAP algorithm. It was further shown that there is a class of convolutional codes–"separable" convolutional codes–which are directly analogous to the multi-dimensional product codes. As such, it is possible to use the iterated MAP "filter" to decode the separable convolutional codes. The major impetus for the work discussed in this thesis stems from the desire to use the iterative MAP "filter" as the inner decoder for separable convolutional codes in a concatenated coding scheme. In that, the performance of the overall concatenated coding scheme will improve if we can increase the accuracy of the soft decisions passed from the inner iterated MAP "filter" to the outer decoder.

There are several other applications whose performance improves by the delivery of soft decisions to the outer decoder. Those applications which have a source decoder that uses soft decisions will obviously benefit from higher quality soft decisions being input to them. Two examples of such applications are voice decoding and image decoding. Also, in quality monitoring, an estimate of the quality of the decision for each bit is passed to the user. Clearly, improving the accuracy of the reliability estimates output by the iterated MAP decoder will enhance its usefulness in this application.

The decoding algorithm of the MAP filter begins with a block of data for which the likelihood of each bit being a zero or a one is provided without taking into account the encoding. That is, we determine what the *a posteriori* probability that a channel bit value was zero based solely on the received signal in a single symbol interval. The symbol-by-symbol MAP algorithm is then applied sequentially, one dimension followed by another, and new estimates of the probabilities are obtained by taking into account the code employed. These *a posteriori* probabilities can then be converted to decisions on what was transmitted by the simple rule of deciding on whether a "0" or "1" was more probable. However, it was also demonstrated that if the symbol-by-symbol MAP algorithm is applied to these new probability estimates produced by the first application of the MAP filter algorithm to all of the dimensions, the error rate can be further reduced [5]. This iteration can be repeated several times until the error rate ceases to improve significantly. The number of iterations which must be performed before this point is reached has been found to be dependent on the dimension of the multi-dimensional product code as well as the complexity of the component codes of the product code.

It has been found, however, that the iteration of the algorithm introduces two problems. The most serious of these problems being that reliability estimates become increasingly over-optimistic as the number of iterations increases. This problem arises because the MAP algorithm assumes that the error statistics for different bits are unrelated, which while typically true for the initial application of the algorithm in each dimension, is invariably false for subsequent iterations. Secondly, the numerical precision necessary to store the significant part of the estimates produced increases as the number of iterations increases. This is due to the fact that the values of the estimates become increasingly closer to zero or one.

1.1. Thesis Objectives

In this thesis, several modifications to the iterated MAP algorithm will be explored in attempts to mitigate the major undesirable effect of iterating the algorithm, i.e. the

decrease in accuracy of the reliability estimates output. It will be shown that one of these modifications–"remapping"–was able to improve the performance of the iterated MAP "filter".

1.2. Thesis Organization

This thesis is organized as follows: In Chapter 2, a review of the literature is given covering concatenated coding, the benefits of soft decision decoding, the MAP algorithm, and the iterated MAP algorithm. In Chapter 3, we introduce the notion of error subfields and as well as present theoretical analyses of subfield error rate versus reliability estimate. In Chapter 4, the concept of "remapping" as a modification to the iterated MAP algorithm is introduced, and we present simulation results supporting the effectiveness of the remapping modification to the iterated MAP algorithm. Chapter 5 contains brief descriptions of the other modifications that were investigated, but were found to be ineffective in improving the accuracy of the reliability estimates. Finally, conclusions as well as suggestions for further work are presented in Chapter 6.

Chapter 2
Concatenated Coding and the MAP Algorithm

2.1. Overview

It was demonstrated in 1948 by Claude Shannon [6, 7] that it is possible to devise a system of encoding data which can achieve an arbitrarily low error-rate during transmission of information through a channel, so long as the capacity of the channel is not exceeded. Unfortunately, this theory did not provide a practical method for constructing such coding systems. Since 1948 much of the work in field of error control coding has been concerned with the realisation of a system which can achieve almost error–free operation at rates closer to channel capacity in various situations.

A number of coding schemes have been utilised, one of which is that of block coding. Simple block codes can be used to build more powerful block codes called product codes. Product codes can be of any dimension, the simplest non-trivial cases being two-dimensional (in which two block codes produce the product code). In a 1992 paper [5] Lodge *et al* introduced the symbol-by-symbol maximum *a posteriori* (MAP) "filter" as an algorithm for decoding multidimensional product codes. The concept of the MAP filter was extended to the decoding of separable convolutional codes, in which a large convolutional code is formed from component convolutional codes.

The MAP filter produces soft decisions on each bit received, which can then be passed to another signal processing device for further analysis. An example of this is provided by the outer decoder in a two-tier concatenated coding scheme, described in Chapter 4. The soft decisions that are made in this system are in the form of the a *posteriori* probability

that a given bit is "zero". If this value is greater than 0.5, we can say that the soft "decision" indicates that the bit was a zero, and the amount by which it exceeds 0.5 providing the confidence that the decision is correct. Conversely if the probability is less than 0.5, then the soft "decision" may be taken that the bit was a one, and the amount by which the probability is less than 0.5 is an indication of the confidence that the decision is correct. Therefore, if the probability has a value close to 0.5 we would have more uncertainty about the decision than if the value was close to 0.0 or 1.0.

It was found that iterating the MAP filter reduces the bit error rate (BER) of the data stream output by the filter. This is achieved because the iterated MAP filter displays an ability to "lock onto" valid codewords [5]. The process by which this is achieved is not fully understood at the present time and is not explored in this work. Instead, this thesis focuses on dealing with the deleterious effect of iteration; which is that the reliability estimates output by the iterated MAP filter are inaccurate. That is, the estimates become increasingly overly optimistic with each application of the MAP filter. Specifically, in simulations testing the iterated MAP filter, it was found that the average probability of bit error calculated from the reliability estimates was less than the bit error rate obtained by counting the errors which occurred.

The use of the iterated MAP filter as the inner decoder in a concatenated coding scheme can be viewed as accepting the channel output and then using knowledge of the inner code to 'filter' out some of the noise before passing the signal to the outer decoder. Concatenated coding and its place in error control coding are reviewed in the following section, as well as the reasons why it is desirable to use an inner decoder that passes soft decisions to the outer decoder.

2.2. Error Control Coding

Error control coding refers to methods that mitigate against effects of noise in communication systems that might otherwise produce errors. The means of doing so usually consists of adding additional symbols to a data stream which replicate the

information in the data stream from the source in a particular standard format. Most often, this produces a symbol stream which has a higher symbol rate than the symbol stream from the data source, thereby usually requiring a higher transmission bandwidth than would be required to transmit the symbol stream directly from the data source. Error correction or detection is accomplished by monitoring the inconsistency between the received symbol stream and what the encoding process might possibly generate. Error control coding schemes can be designed for either error detection or error correction, or both. In general, however, a code can always detect more errors than it can correct.

There are two main types of codes commonly employed today. These are block codes and convolutional codes. This thesis addresses block codes, and the special form of a block code that is a product code, constructed from other simple block codes.

2.2.1. Block Codes

In block coding the encoder divides the input data stream into blocks of k information symbols, termed *data words*, and uses these k symbol sequences to produce n channel (or code) symbols, termed *codewords*. The process is depicted in Figure 2.1. The resulting encoding process is termed an (n,k) code. Block codes are commonly classified according to the size of the alphabet for symbols. In a binary code, the symbols may only assume binary values. In that case there are 2^k possible data words (or messages), and thus 2^k codewords which are drawn from amongst 2^n possible binary n-tuples.

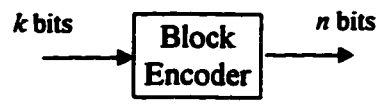


Figure 2.1. An (n,k) binary block encoder

If the data word symbols appear in fixed positions in the codewords (i.e., k symbols of the codeword are the k data symbols), then the code is called *systematic*. One class of systematic codes that are frequently used are the parity codes which we describe next.

2.2.2. Parity Codes

Parity codes are a type of systematic binary block code in which a single bit is added to the information message to always bring the sum of the values of the bits in a codeword to either always an even number or always an odd number (labelled as even or odd parity codes respectively). Therefore, parity codes are (k+1, k) block codes, where the bits, c_1, c_2, ..., c_{k+1}, in each codeword satisfy the parity equation.

$$c_1 \oplus c_2 \oplus \cdots \oplus c_{k+1} = \begin{cases} 0, & \text{for an even parity code;} \\ 1, & \text{for an odd parity code.} \end{cases} \tag{2.1}$$

A (4,3) parity encoder is shown below in Figure 2.2.

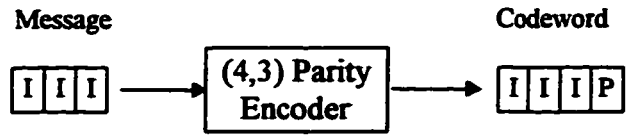


Figure 2.2. A (4,3) parity encoder, where the I represent the input bits, and P is the parity bit added at the end of a codeword by the encoder.

In parity codes, errors are detected when a violation of the parity equation is found to have occurred. If on reception and decoding of an *even* parity codeword, it is found that that the sum of the bits with value "1" is *odd*, it is certain that at least one error has occurred. Note that on detection of an error event, there is no information on which bit of the codeword is in error and it is not possible to correct the error. Therefore the parity code is only an error detecting code.

Parity codes are further limited in that they are only capable of detecting approximately half of all the possible errors events. Specifically, errors can be detected whenever an odd number of errors in the transmission of a codeword are made, but are undetected when an even number of errors occur.

The modifications made to the iterated MAP algorithm in this thesis were investigated using 2-dimensional product codes with component even parity block codes. Parity codes

were used due to their simplicity. In addition, the fact that parity codes are systematic lends itself to the implementation of a more effective decoding strategy.

2.2.3. Product Codes

As mentioned earlier, product codes represent a method by which more powerful block codes can be constructed from simpler ones. The resulting overall block code is more powerful in the sense that they are able to correct and detect more errors than the constituent codes. They are also better able to deal with a contiguous series of errors (burst errors), since they contain a sort of "interleaving" (discussed below).

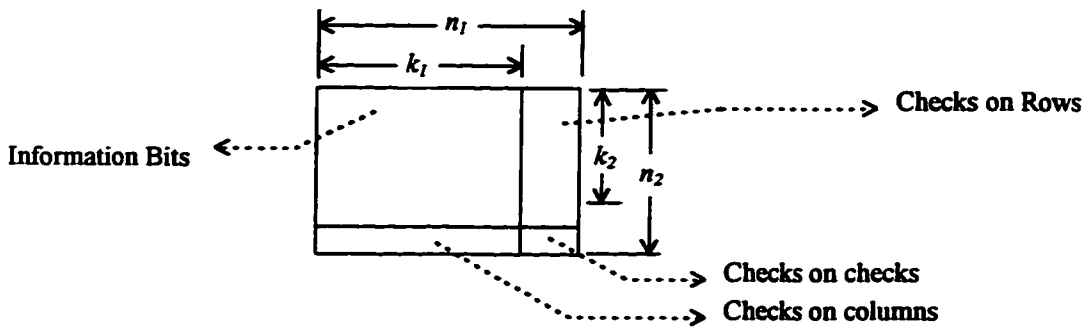


Figure 2.3. The codeword block for a two-dimensional product code with component codes $C_1(n_1, k_1)$ and $C_2(n_2, k_2)$.

In general, a product code is not constrained to have identical block codes as its components, nor must it be confined to two dimensions. We may describe a 2-D product code as forming an $n_1 \times n_2$ matrix as a codeword where the rows are arrived at by applying a block code C_1 to k_2 consecutive k_1 bit data words, and then we construct new codewords from the columns of the matrix by applying a block code C_2 to the n_1 different k_2 long bit columns of the matrix to produce n_2 bit codewords. Figure 2.3 illustrates the process for systematic codes where additional parity bits are appended to datawords.

2.2.4. Interleaving

Interleaving is a method of constructing burst-error correcting codes from simple codes. For example, a code with a burst error correction capacity of y can be constructed from single-bit error correcting codes interleaved to degree y. A block interleaver of degree y arranges y codewords of a component code into y rows each having n columns. The bits are then read out in consecutive columns and transmitted. On reception, the bits are then "de-interleaved", which is the reverse operation of interleaving. In de-interleaving, the received bits are written to an array of n columns and then read out of the array in consecutive rows. In essence, an (n, k) block code interleaved to degree y can be considered to be a (yn, yk) block code, that is, the resulting code is y times as long with y times as many information bits. An interleaver of degree y is shown in Figure 2.4 below.

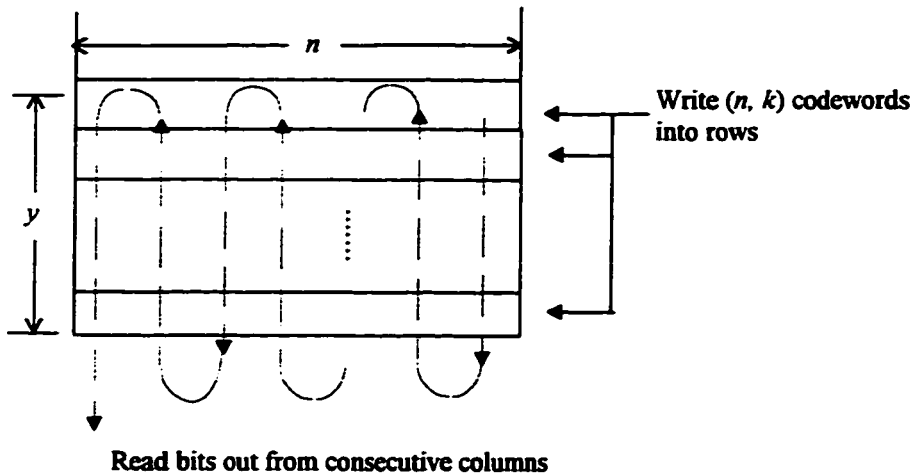


Figure 2.4. A interleaver of degree y.

The interleaver increases the burst-error capability by making any burst of errors which occurs affect only a few bits in each row. Consider a single-error correcting code interleaved to degree y. Any burst error of length y or less will be 'spread' over the codewords, such that each component codeword will have at most one error, which can be corrected by the single error correction component code. It should be noted however, that should an error burst of length greater than y occur entirely within a block, then at

least one of the component codewords will have more than one error, and the error may not be corrected unless the code is inherently capable of correcting more than 1 error.

2.2.5. Convolutional Codes

Convolutional coding is a channel coding scheme in which the output of the encoder, at any given time instant, depends not only on the input of encoder input at that time, but also on previous inputs, i.e., the encoder has *memory*. A convolutional encoder is characterized by three parameters n, k and m, where:

- n represents the number of encoder outputs at each time unit, t,
- k represents the number of encoder inputs at time t, and
- m represents the memory of encoder, or in other words, the number of previous encoder inputs on which the encoder output depends.

Convolutional codes can be contrasted with block codes in that the n outputs at any time unit of an (n,k) block encoder depends only on the previous k encoder inputs at that time unit.

2.2.6. Concatenated Coding

Concatenated coding was initially proposed by Forney [8] as a method of constructing long codes from shorter ones. The resulting code is more powerful and does not require the level of decoding complexity usually associated with long block codes. A generic concatenated coding scheme with two levels of concatenation is shown below.

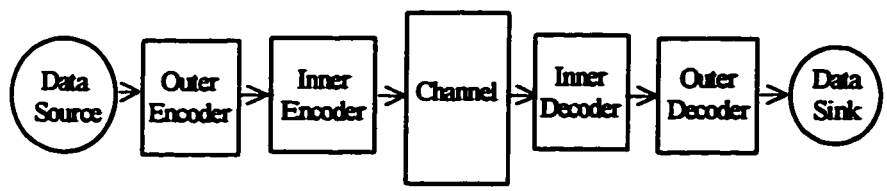


Figure 2.5. Model of a two-level concatenated coding scheme.

The concept has been extended to a number of variations on the types of channel coding/decoding schemes employed in the inner and outer codecs (coder/decoders). One variation which has received some attention in the literature is the concatenation of two convolutional codes, each of which is decoded using a Viterbi decoder.

2.2.7. Separable Convolutional Codes

In the paper by Lodge *et al* [5] the concepts developed for decoding multidimensional product codes were extended to convolutional codes. This is desirable because of the superiority shown by convolutional codes over block codes in many applications in digital communications. The extension to convolutional codes is achieved when it is recognized that a product code can be regarded as a special case of concatenated coding with interleaving between the coding stages. Specifically, a concatenated coding scheme with interleaving in which the interleaver has a number of columns equal to the length of the first component code in the product code, and a number of rows equal to the length of the second component code.

A class of convolutional codes termed 'separable' convolutional codes are analogous to product codes. They can be constructed by using time-division interleaving to interleave small convolutional codes. Separability is the property of being able to view the composite codeword as being composed of codewords from either of the component codes depending on the grouping of the bits of the composite codeword. Separable convolutional concatenated codes will not be considered further in this work. However it is expected that the modifications to the iterated MAP algorithm are as applicable to separable convolutional codes as they are to product codes.

2.2.8. Decoding

A very important consideration in the construction of any code is the manner in which it may be decoded after transmission. It is desirable that decoding be done as fast and simple as possible. Decoding techniques can generally be divided into several categories,

including maximum likelihood (ML) decoding and maximum *a posteriori* (MAP) decoding.

In MAP decoding the largest *a posteriori* probability is used to decide which signal was transmitted. This decision rule results in the minimum probability of decision error [9]. In an M-ary signalling scheme the *a posteriori* probabilities take the following form:

$$P[a_n = m \mid y(t),\ t \in I_{obs}] \quad \text{for } m = 1, 2, \ldots, M \tag{2.2}$$

where $y(t)$ is the received signal on the nth observation interval, I_{obs}, and a_n represents the symbol transmitted on the observation interval which we assume comes from the alphabet $\{1, 2, 3, \ldots, M\}$. The M probabilities represented by (2.2) are compared and the value of m corresponding to the maximum *a posteriori* probability is chosen as the decoder output.

Maximum Likelihood stems from the observation that

$$P\left[a_n = m \mid y(t), t \in I_{obs}\right] = \frac{P[a_n = m] \cdot p(y(t) \mid a_n = m)}{p(y(t))} \tag{2.3}$$

where $p(y(t)|a_n=m)$ is the conditional density function and $p(y(t))$ is the marginal density functions for the sufficient statistics of the received signal. If all M possibilities for a_n are equiprobable, then from (2.3) we observe that (2.2) is maximized by maximising $p(y(t)|a_n=m)$ since $p(y(t))$ does not depend on m.

The quantity $p(y(t)|a_n=m)$ may be regarded as the likelihood of observing $y(t)$ if the transmitted symbol had value m, so that maximising this likelihood is another way of optimally deciding on the transmitted symbol values (optimum for the equiprobable data case). In the additive white Gaussian channel noise case, these likelihoods are monotonic decreasing functions of the distance between the observed signal $y(t)$ and the hypothesized transmitted signals and so this decision rule is equivalent to the rule of picking a value m to match the possible transmitted signal which is nearest to the received signal $y(t)$.

Likelihood is a function of distance between received signal and signal points, thus an ML receiver seeks the closest signal point to the received signal.

2.2.8.1. Viterbi Decoding

The Viterbi algorithm (VA) was introduced by Andrew Viterbi in 1967 [10] as a method of decoding convolutional codes. The algorithm uses a coding trellis and metrics based on the received signals to determine the most probable path through the trellis and thus derive the corresponding transmitted sequence. In other words, it is an optimal algorithm for the decoding of convolutional codes in the sense that it will select the most likely sequence of bits. In general however, it is not optimal in the sense of minimising the bit error rate.

The Viterbi decoder, while optimal, does suffers the disadvantage of being slow in the decoding of convolutional codes with high memory orders. This is so because the number of computations required for the decoding process increases exponentially with the memory order. Memory order refers to the number of bits input to the coder on which each bit output by the coder is dependent.

2.2.8.2. Hard Decision versus Soft Decision Decoding

In binary hard-decision decoding in a concatenated code, the output of the first decoder is quantized into one of two discrete values before being input to the second decoder. This contrasts with binary soft-decision decoding where the decoder output may be described as being quantified into more than two values. Block and convolutional decoders both display improved performance when soft decision decoding is used as opposed to hard decision decoding. In general, when soft-decision decoding is used, a coding gain of 2dB is found [11]. This improvement can be interpreted as resulting from the increased information present in soft-decisions.

2.2.8.3. Soft decisions in Concatenated Coding

Generally in concatenated coding, employing a convolutional code as the inner code, the inner decoder benefits from soft decisions from the demodulator and outputs hard decisions to the outer decoder. We would expect some performance improvement if the outer decoder received soft decisions from the inner decoder. In fact, Hagenauer & Hoeher [3] have shown that significant performance gains can be realized in a

concatenated coding scheme (using convolutional coding) if soft decisions were passed to the outer decoder. This improvement is realized because the outer Viterbi decoder is then able to make better sequence estimations when soft decisions are input.

One method of generating soft decisions, in a concatenated convolutional coding scheme employing two Viterbi decoders is to modify the VA of the inner decoder to produce soft decisions. The VA modified in this way is termed the Soft Output Viterbi Algorithm (SOVA) [3]. However, the SOVA is sub-optimal–in a symbol-by-symbol sense–in that, like the Viterbi Algorithm, the probability of sequence error is minimized rather than the probability of symbol error. The symbol-by-symbol MAP algorithm is optimal in the sense that it minimises the probability of bit error based on the limited information it uses about the received signal. This algorithm can be used in the inner decoder and is in fact used in this work. The symbol-by-symbol MAP algorithm is discussed further in Section 2.3 below.

2.3. The Symbol-by-Symbol MAP Algorithm

2.3.1. Overview

The symbol-by-symbol MAP algorithm is an optimal general algorithm for calculating the *a posteriori* probabilities (APP) of symbols from a Markov source observed through a memoryless channel. The algorithm thus forms the basis of a receiver which minimises the probability of bit error. These APPs serve as estimates of the reliability of the decoded bit. The calculation of the APP is facilitated through the use of a decoding trellis as described in the next section.

2.3.2. Trellis Decoding

In a 1974 paper [12] Wolf described a method of applying trellis decoding to linear (n, k) block codes. This process is critical in the application of the symbol-by-symbol MAP algorithm to block codes.

Consider a binary (n,k) block code. The parity check matrix of this code is an n-k by n matrix H, with the property that those binary n-tuples X that solve $HX=0$ (where 0 is the $(n-k)$-tuple of zeros) are precisely the codewords of the code. Let h_i, for $i=1,2,\ldots,n$ denote the various columns of H.

It is possible to describe the mapping of datawords to codewords in a block code as the traversing of a trellis (a series of sets of nodes interconnected by unidirectional branches from one set to another). Such a trellis description of a block code allows us to apply the techniques of trellis decoding, usually reserved for convolutional codes, to block codes and to generate soft decisions.

To construct such a trellis, we begin with a series of sets of nodes, with the sets indexed by the parameter t termed the *depth* of the various nodes in the set. Branches will connect nodes at depth t to nodes at depth at $t+1$. There are at most than 2^{n-k} nodes in each set. We identify each node in the set of nodes at depth t with a binary $(n-k)$-tuple. Let $s_{m,t}$ –for $m=0,1,\ldots,2^{n-k}-1$– denote the various possible binary $(n-k)$-tuples. Let M_t denote those $s_{m,t}$ corresponding to a node at depth t. At depth $t=0$, there is only one node which is identified with $s_{0,0}$. This is the starting node. From this starting node we construct two branches to two nodes at depth 1 which we identify with the possible values of the first codeword bit. We identify the terminal state of the branches according to the rule

$$s_{1,1}= s_{0,0} + l_1 h_1;\ \ l_1=0,1,$$

and identify the branch leading to $s_{1,1}$ with the value of l_1. From the two nodes at depth $t=1$, we generate the nodes at depth $t=2$ in a similar manner, and so forth for all depths according to the general rule that there are nodes corresponding to each distinct value of

$$s_{m,0} + l_{t+1} h_{t+1};\ \ m \in M_t,\ l_{t+1}=0,1.$$

We proceed in this manner until we have defined nodes to a depth n. The passage through this trellis arrives then at nodes which gives us the result of the product

$$H[l_1, l_2, l_3, \ldots, l_n]^T.$$

Obviously codewords will correspond only to those sequences which terminate in the zero state, so if we remove from the last set of nodes all the other nodes other than 0 and then

remove the nodes at depth $t=n-1$ that lead only to non-zero nodes at depth $t=n$ and process likewise to the nodes at depth $t=n-2$, leading to nodes that had been eliminated, etc., we ultimately produce a trellis where the paths from the first non-zero node to the last zero node represent all the codewords. The process of the removal of nodes is termed expurgation of the trellis.

As an example, consider the formation of the trellis for a binary (4,3) even parity code. For a binary code, there is only one parity check matrix H which is the (1,4) binary matrix $H=[1,1,1,1]$ where each column $h_i=1$ for all I. Since $n=4$ and $k=3$ the trellis in this case (prior to expurgation) will have two nodes at each depth $t>0$ and is that trellis shown in Figure 2.6. After expurgation we obtain the trellis in Figure 2.7.

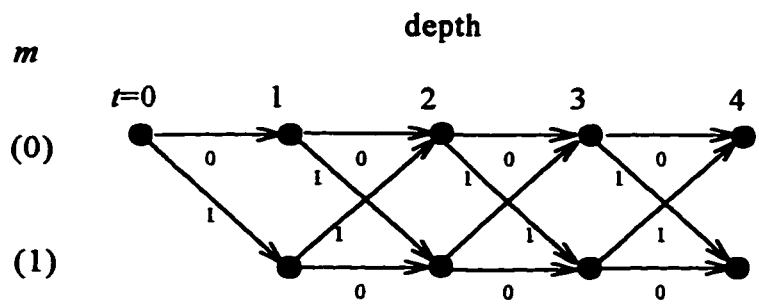


Figure 2.6. Trellis for (4,3) even parity code.

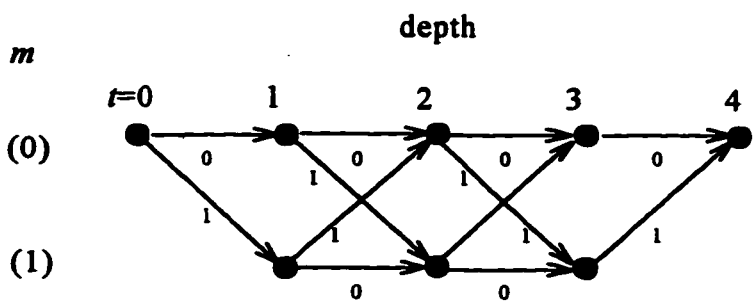


Figure 2.7. Expurgated trellis for (4,3) even parity code.

From Figure 2.7 it can be verified that each unique path through the trellis, moving from the starting node to the ending node, corresponds to a codeword of the (4,3) parity code.

The discussion above describes a trellis for which the state vector is a "parity vector". While such a formulation is straightforward and useful, the resulting trellis is not guaranteed to be minimal, in the sense that it is not necessarily the trellis representing the given code with the fewest states. Techniques for finding a minimal trellis are described in [13, 14].

The decoding trellis and its associated state transition probabilities are used by the symbol-by-symbol MAP algorithm to determine that a given input was transmitted based on the received codeword information. The details of how this accomplished is presented in the following section.

2.4. Iterated Symbol-by-Symbol MAP Algorithm

As previously mentioned, the iterated symbol-by-symbol MAP algorithm has been proposed for the decoding of multidimensional product codes. This algorithm is based on the symbol-by-symbol MAP algorithm proposed by Bahl *et al* [15], except that the recursions are based on conditional probabilities as opposed to the conditional probability density functions used in [15]. This modification simplifies the iterated process by eliminating the need to convert between probability and density functions.

The channel model considered is shown in Figure 2.8 below. Channel coding is performed using a concatenated coding scheme. The exact outer code employed is irrelevant to our discussion of the iterated MAP process. Suffice it to say that the outer encoder passes sequences of data, D_i, to the inner encoder. The inner encoder encodes these data sequences with a given code C. The code C is required to have the property that it can be represented by a trellis of finite duration.

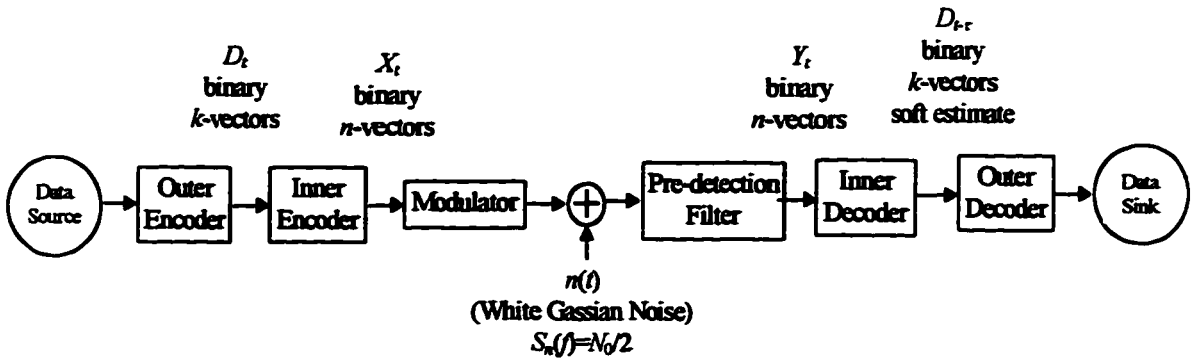


Figure 2.8. Channel model

It is assumed that the inner code is a multidimensional product code, with (n, k) component block codes. Since it is possible, in this case of block codes, for the number of states in the trellis representation of the code to vary with trellis depth, we represent the number of states at depth t by M_t. It is also assumed that at the start and end of the time interval of interest, the trellis is in the zero state.

Now, the sequence of binary digits, ${}_iD_{i'}$, input to the inner block encoder, satisfying the conditions given above, will correspond to a sequence of states given by

$$ {}_{t-1}S_{t'} = \{S_{t-1} = 0, \ldots, S_t = m, \ldots, S_{t'} = 0\}, \tag{2.4}$$

where $m \in \{0,1,\ldots,M_{t-1}\}$.

Given the input sequence the inner encoder will produce the binary output

$$ {}_iX_{i'}(k) = \{x_i(k), \ldots, x_t(k), \ldots, x_{i'}(k)\}. \tag{2.5}$$

The encoder output sequence is transmitted over an AWGN channel producing the real-valued received sequence of sufficient statistics

$$ {}_iY_{i'} = \{Y_i, \ldots, Y_t, \ldots, Y_{i'}\}. \tag{2.6}$$

The sequence is real-valued as a result of the exposure of the transmitted signals to AWGN and the matched filtering of the received signal. This method of linear demodulation gives rise to an infinite number of possible values of the demodulator output. If the AWGN has a power spectral density of $N_0/2$, then the probability

distribution of the noise contribution at the predetection filter output (sampled value) is normal with zero mean and a variance of $N_0/2$.

The real-valued received elements therefore have conditional probability density functions given by

$$p(y_t|x_t) = \frac{1}{2\pi\sigma^2}\exp\left[\frac{-(y_t - x_t)^2}{2\sigma^2}\right]. \tag{2.7}$$

The APP of the transmitted bits can be determined using Bayes' theorem:

$$\begin{aligned} \mathrm{P}[x_t = i|y_t] &= \frac{p(y_t|x_t = i)\cdot \mathrm{P}[x_t = i]}{p(y_t)} = \frac{p(y_t|x_t = i)\cdot \mathrm{P}[x_t = i]}{p(y_t; x_t = 1) + p(y_t; x_t = -1)} \\ &= \frac{p(y_t|x_t = i)\cdot \mathrm{P}[x_t = i]}{p(y_t|x_t = 1)\cdot \mathrm{P}[x_t = 1] + p(y_t|x_t = -1)\cdot \mathrm{P}[x_t = -1]}. \end{aligned} \tag{2.8}$$

Assuming equiprobable symbol values, this simplifies to

$$\mathrm{P}[x_t = i|y_t] = \frac{p(y_t|x_t = i)}{p(y_t|x_t = 1) + p(y_t|x_t = -1)}. \tag{2.9}$$

The APP can therefore be expressed as

$$\mathrm{P}[x_t = i|y_t] = K_t \cdot p(y_t|x_t = i), \tag{2.10}$$

where

$$K_t = \frac{1}{p(y_t|x_t = 1) + p(y_t|x_t = -1)}. \tag{2.11}$$

Note that the value of K_t is independent of the hypothesized value of x_t.

The knowledge of the structure of the code can be used to determine the probability that the kth codeword was transmitted:

$$\mathrm{P}[\mathbf{X} = {}_iX_{i'}(k)|{}_iY_{i'}; C] = \frac{p({}_iY_{i'}|\mathbf{X} = {}_iX_{i'}(k))\cdot \mathrm{P}[\mathbf{X} = {}_iX_{i'}(k)]}{\sum_{j=1}^{K} p({}_iY_{i'}|\mathbf{X} = {}_iX_{i'}(j))\cdot \mathrm{P}[\mathbf{X} = {}_iX_{i'}(j)]}, \tag{2.12}$$

where K is the total number of possible codewords.

If we assume that the modulation used is memoryless, the statistics y_i are all independent and so we find

$$p\left({}_{i}Y_{i'} \mid \mathbf{X}={}_{i}X_{i'}(k)\right)=\prod_{t} p\left(y_t \mid x_t = x_t(k)\right). \tag{2.13}$$

Using (2.10) we find,

$$\begin{aligned} p\left({}_{i}Y_{i'} \mid \mathbf{X}={}_{i}X_{i'}(k)\right) &= \prod_{t} K_t \cdot \mathrm{P}\left[x_t = x_t(k) \mid y_t\right] \\ &= \left(\prod_{t} K_t\right)\cdot\left(\prod_{t} \mathrm{P}\left[x_t = x_t(k) \mid y_t\right]\right). \end{aligned} \tag{2.14}$$

Assuming equiprobable codewords, then using (2.14) in (2.12), we arrive at

$$\mathrm{P}\left[\mathbf{X}={}_{i}X_{i'}(k) \mid {}_{i}Y_{i'}; C\right]=\frac{\prod_{t} \mathrm{P}\left[x_t = x_t(k) \mid y_t\right]}{\sum_{j=1}^{K}\prod_{t} \mathrm{P}\left[x_t = x_t(j) \mid y_t\right]}. \tag{2.15}$$

We want to compute the probability that a given branch of the trellis is traversed by a particular codeword. In order to facilitate this computation we define $\mathrm{B}(m',m)$ as the index set of all the codewords that traverse the trellis branch between states $S_{t-1}=m'$ and $S_t=m$. The state transition probability is given by

$$\mathrm{P}\left[S_{t-1}=m'; S_t=m \mid {}_{i}Y_{i'}\right]=\frac{\sum_{k\in B(m',m)} \mathrm{P}\left[\mathbf{X}={}_{i}X_{i}(k) \mid {}_{i}Y_{i}; C\right]}{\sum_{(q',q)}\left(\sum_{k\in B(q',q)} \mathrm{P}\left[\mathbf{X}={}_{i}X_{i}(k) \mid {}_{i}Y_{i}; C\right]\right)}. \tag{2.16}$$

The computation of the state transition probability is accomplished by a recursion process, that is,

$$\begin{aligned}\sigma_t(m',m) &= \sum_{k\in B(m',m)} \mathrm{P}\left[\mathbf{X}=_i X_{i'}(k)|_i Y_{i'};C\right] \\ &= p\left(S_{t-1}=m';S_t=m,_i Y_{i'}\right) \\ &= p\left(S_{t-1}=m';_i Y_{i'-1}\right)\cdot p\left(S_t=m,Y_t|S_{t-1}=m'\right)\cdot p\left(_{t+1}Y_{i'}|S_t=m\right) \\ &= \alpha_{t-1}(m')\cdot\gamma_t(m',m)\cdot\beta_t(m),\end{aligned} \tag{2.17}$$

where

$$\begin{aligned}\alpha_t(m) &= p\left(S_t;_i Y_{i'}\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} p\left(S_{t-1}=m';S_t=m,_i Y_{i'}\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} p\left(S_{t-1}=m';_i Y_{t-1}\right)\cdot p\left(S_t=m,_t Y_{i'}\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} \alpha_{t-1}(m')\cdot\gamma_t(m',m),\end{aligned} \tag{2.18}$$

$$\begin{aligned}\beta_t(m) &= p\left(_{t+1}Y_{i'}|S_t=m\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} p\left(S_{t+1}=m';_{t+1}Y_{i'}|S_t=m\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} p\left(S_{t+1}=m';Y_{t+1}|S_t=m\right)\cdot p\left(_{t+2}Y_{i'}|S_{t+1}=m'\right) \\ &= \sum_{m'=0}^{M_{t-1}-1} \beta_{t+1}(m')\cdot\gamma_{t+1}(m',m),\end{aligned} \tag{2.19}$$

and

$$\begin{aligned}\gamma_t(m',m) &= \sum_X p\left(S_t=m,Y_t|S_{t-1}=m'\right)\cdot\mathrm{P}\left[X_t=X|S_{t-1}=m',S_t=m\right]\cdot p\left(Y_t|X\right) \\ &= \sum_{j=1}^{n}\mathrm{P}\left[x_{t_j}=x_{t_j}(m',m)|y_{t_j}\right].\end{aligned} \tag{2.20}$$

Note that $\alpha_1(0)=1$, $\alpha_1(m)=0$, $\beta_1(0)=1$, and $\beta_1(m)=0$ for $m\neq 0$.

It is desired to use the MAP algorithm to determine the reliability estimates in the form of the probabilities $\mathrm{P}\left[x_{t_j} = 0|_i Y_i; C\right]$. This is achieved by defining A as the set of state transitions for which the coder output is zero, i.e.

$$A = \{(m', m): x_{t_j}(m', m) = 0\}, \tag{2.21}$$

the reliability estimates are then found by evaluating

$$\mathrm{P}\left[x_{t_j} = 0|_i Y_i; C\right] = \frac{\sum_{(m',m)\in A} \sigma_t(m', m)}{\sum_{(m'm)} \sigma_t(m', m)}. \tag{2.22}$$

2.4.1. Iteration

Iteration of the algorithm means that the MAP algorithm is applied several times to the reliability estimates obtained from the channel output. In a single iteration (or cycle) the MAP algorithm is applied to each codeword in each dimension one dimension at a time. For example, in the case of a two-dimensional product code, such as the one considered in this thesis, the MAP algorithm would be applied to each codeword in each row and then to each codeword in each column, or vice versa. In the case of identical codes in each dimension, as used in this thesis, the choice of which dimension is to be operated on first is arbitrary.

2.4.1.1. Benefits of Iteration

It was found that iterating the algorithm has the effect of lowering the observed BER. It is felt that this is achieved by the algorithm 'locking onto' valid codewords, though the valid codeword which results may not in fact be the codeword that was transmitted. The decrease of BER with multiple iterations is illustrated in Figure 2.9 below.

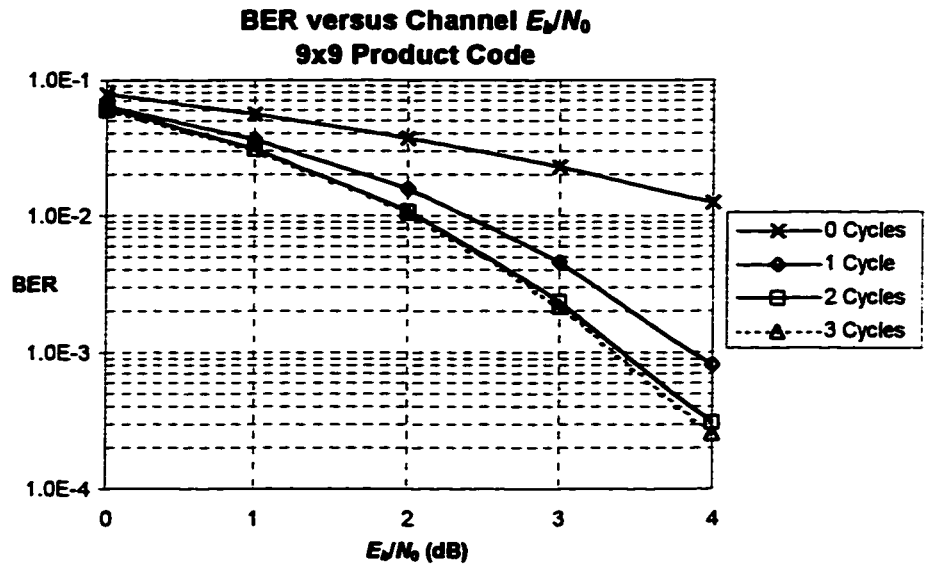


Figure 2.9. BER versus Channel E_b/N_0 for several iterations of the MAP algorithm for a 2D product code using (9,8) parity codes in each dimension.

The symbol E_b/N_0 is the energy per transmitted bit to noise ratio, and serves as a measure of the signal strength used in transmission. In general we expect that the higher this ratio the lower the BER.

The above graph is an illustration of empirical results gathered by simulation. Note that there is significant improvement in the BER when the number of cycles is increased from one to two. However, when the number of iterations is changed from two to three the change in BER is relatively small, and the improvement is even smaller from three to four. This is due to the relative simplicity of the parity block codes which were used in the simulations. It was observed in [5] that when more complex component codes are used in the product code, significant improvements in performance can continue for many iterations. The number of dimensions used in the product code also affects the improvement resulting from iteration. In this work simple two-dimensional product codes were used.

2.4.1.2. Problems with the Iteration Process

Despite the benefits of iterating the MAP algorithm there is a problem associated with its use. The problem arises from the fact that the MAP algorithm assumes that the channel noise on each statistic y_i is statistically independent. This assumption is valid for the first complete iteration of the algorithm if the channel is AWGN. However, for subsequent iterations the assumption is not valid in that the value of the reliability estimate of any one bit in the block now depends, to varying degrees, on the value of all the reliability estimates in the block. This violation of the independence assumption causes the value of the reliability estimates to become inaccurate if more than one iteration of MAP processing is performed.

Once a valid codeword is output, if any more MAP processing is performed on the codeword, the reliability estimates are observed to assume values reflecting increasing certainty of the reliability of the estimate. That is, the reliability estimates, which are in the form of APPs, take on values closer and closer to zero or unity as further iterations are carried out. This convergence to 0 and 1 is over-optimistic.

Given that the APP represents the probability of a transmitted bit being 0, given the received symbol information, if this probability p is greater than ½ and we decide that a 0 was sent, the chance that we made a mistake is the probability that the transmitted bit was a 1, which is 1-p. Similarly, if p is less than 1/2 and we therefore decide that a 1 was sent, then the probability of an error is simply p. If 0's and 1's are equally likely (*a priori*), then the probability of error given the reliability data p is

$$p_e = \begin{cases} 1-p, & p > 1/2; \\ p, & p < 1/2. \end{cases} \tag{2.23}$$

If we average the reliability values for each bit then we will produce the overall error rate (termed the predicted average probability of bit error).

If we apply this result to the "reliability estimates" results produced by repeated application of the MAP algorithm, and compare the results with the empirically determined bit error rate (i.e., determined by counting errors) we obtain results as depicted in Figure 2.10 where the difference between the two values is zero after the first iteration but afterwards grows in % terms (although the BER continues to decrease with further iteration).

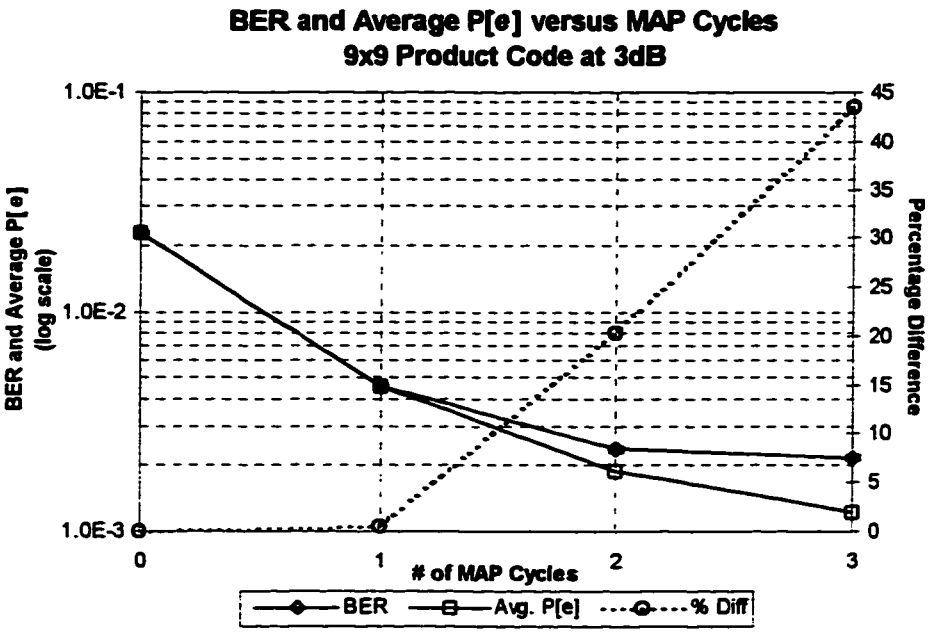


Figure 2.10. Plot of BER and Average Probability of bit error versus number of MAP cycles at channel E_b/N_0 of 3dB.

The BER is an estimate of the probability of bit error obtained by comparing the transmitted bits to those received and then finding the ratio of the number of bits in error to the total number of bits. The average probability of bit error (Avg. P[e]) is obtained by calculating the probability of error assuming the values of the reliability estimates from the MAP algorithm output are correct and then averaging this probability of error over all the reliability estimates observed. In Figure 2.10 it can be seen that the two indices of bit error rate agree when zero and one iteration are performed. However, for more than one iteration, it can be seen that the reliability estimates indicate a lower bit error rate than that which is actually observed. Note also that this deviation increases as more iterations are performed.

2.4.2. Partial Factor Modification

In an effort to mitigate the deleterious effect of iteration, Lodge *et al* [4] proposed a modification to the iterated algorithm referred to as the Partial Factor modification. The idea behind this modification is to preserve as much as possible whatever independence there might be between the component codewords in each dimension of the multidimensional codeword. This is done as shown below.

For systematic codes, such as the parity codes considered in this work, the state transition probabilities [defined in (2.17)], $\sigma_t(m', m)$ can be expressed as

$$\sigma_t(m', m) = \mathrm{P}[x_t = 0 | y_t] \cdot \sigma_t'(m', m). \tag{2.24}$$

Using this expression in (2.22) we find that

$$\mathrm{P}[x_t = 0 | {}_{i}Y_{i}^{t}; C] = \frac{\mathrm{P}[x_t = 0 | y_t] \sum_{(m',m)\in A} \sigma_t'(m', m)}{\sum_{(m',m)} \sigma_t(m', m)}. \tag{2.25}$$

A comparison of the *a posteriori* probabilities after the first application of the MAP algorithm can be made through the likelihood ratio

$$L_t(1) = \frac{\mathrm{P}[x_t = 0 | {}_{i}Y_{i}^{t}; C]}{\mathrm{P}[x_t = 1 | {}_{i}Y_{i}^{t}; C]} = \frac{\mathrm{P}[x_t = 0 | y_t] \cdot \left(\sum_{(m',m)\in A} \sigma_t'(m', m) \right)}{\mathrm{P}[x_t = 1 | y_t] \cdot \left(\sum_{(m',m)\in A} \sigma_t'(m', m) \right)}. \tag{2.26}$$

This can then be expressed as

$$L_t(1) = f_t(1) \cdot L_t(0). \tag{2.27}$$

Thus the output after k iterations of the MAP algorithm can be expressed as

$$L_t(k) = \left[\prod_{q=1}^{k} f_t(q) \right] \cdot L_t(0). \tag{2.28}$$

This form of the expression leads to the observation that the output of the kth iteration can be interpreted as the product of the received likelihood ratio and the refinement factors

from each iteration. The partial factor modification can result in a significant improvement in the accuracy of the reliability estimates as shown by Lodge *et al* [16]. Even with the partial factor modification, however, the issue of reliability estimate inaccuracy is not completely solved.

The divergence between the true P_e and the apparent one resulting from the overly optimistic estimates that remain leads one to speculate that perhaps the iterated MAP algorithm could be improved further if the bias in the reliability estimates could somehow be removed. It is this possibility that we wish to explore in this thesis.

Chapter 3

Theoretical Analysis

3.1. Overview

The method found to improve the accuracy of the reliability estimates is one we term "remapping". The concept is explained in greater detail in Chapter 4 but the salient points will now be mentioned. Remapping is based on our observation that the relationship observed between the reliability estimates (or APP) and the bit error rate (BER) after the first application of the MAP algorithm is different from that provided by (2.23). The method chosen to remedy this difference is to collect the error statistics after MAP processing has been performed and then use our knowledge of the ideal BER/APP relationship to adjust the value of the reliability estimates to values which would make the BER and the average expected P_e from the reliability estimates match. Essentially, what we do is to compute the APP and determine the BER for bits with a given APP so that we can construct a function $f(x)$ to produce a new APP, $\hat{p} = f(p)$, so that the BER for bits with APP=p is given by $f(p) = \min(\hat{p}, 1-\hat{p})$. As will be shown in Chapter 4, the remapping modification to the MAP filter algorithm based on the empirical data was successful in addressing the problem of inaccuracy of the estimates, although there remains room for improvement.

Remapping based on empirical evidence involves gathering statistics showing the distribution of BER against the value of the reliability estimate, fitting a curve to this distribution, and then using this curve to modify the reliability estimates of subsequent transmissions to their proper value. This process was further refined by the use of our knowledge of the structure of the code to group the bits of the codeword into what we call "error subfields". The reader is referred to Section 3.3 below for a detailed explanation of error subfields. Suffice it to say that the BER/APP relationship was

determined for each error subfield and remapping applied to each. This modification allows us to be more precise in determining what the new values of the reliability estimates should be.

It was desired to determine the theoretical relationship between BER and APP in each error subfield in an effort to guide the empirical curve-fitting which needed to be done. A point of special interest is the point where APP=0.5. The behaviour at this point is important because it determines the constraints that might be placed on the fitted polynomial functions. Specifically, it was necessary to determine whether the BER/APP curve was continuous or not at the APP=0.5 point. If continuous, the polynomial curve would be constrained to pass through the (0.5, 0.5) point when the polynomial was fit to the collected statistics. This would result in a continuous remapping function. If discontinuous, then the generated polynomial would not be constrained to pass through the (0.5, 0.5) point. This would result in a discontinuity in the remapping function at the point where the reliability estimate equals 0.5.

We also wished to generate theoretical results against which it would be possible to verify the simulation results. These theoretical results will be generated in the process of doing the following analysis.

3.2. The Channel Model

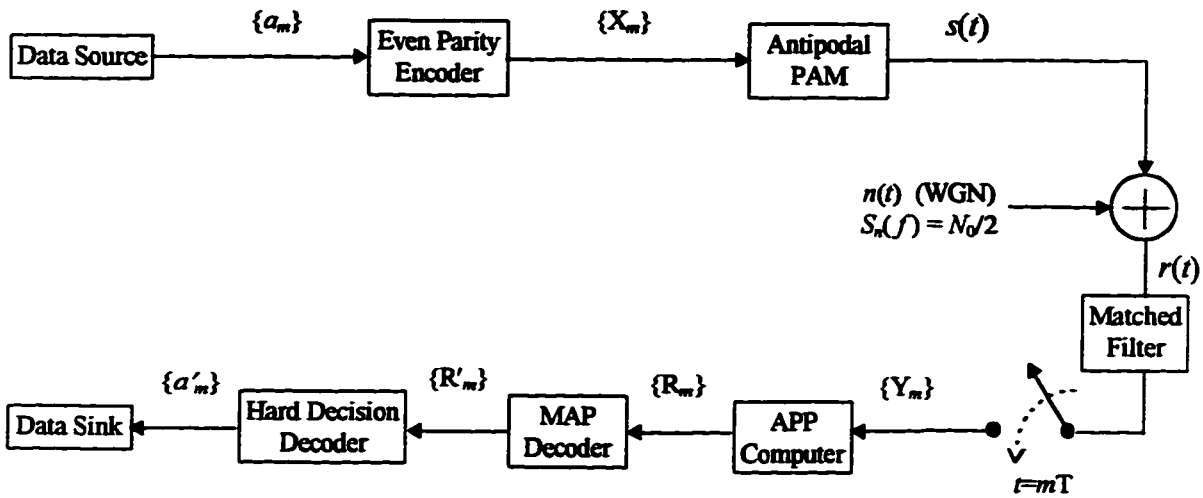


Figure 3.1. Channel Model

Consider the channel model shown in Figure 3.1. In general, if an (n,k) parity code was used, then $\mathbf{X}_m$ would be a binary n-tuple, $\mathbf{Y}_m$, $\mathbf{R}_m$ and $\mathbf{R}'_m$ would be real–valued n-tuples. For example, consider the case in which a (2,1) even parity code is used. In this case $\mathbf{X}_m = [x_1, x_2]$, where $x_i=\{1, 0\}$. Antipodal pulse amplitude modulation (PAM) is a modulation scheme in which the data is transmitted using pulses, $p(t)$ and $-p(t)$. Specifically, binary zeroes are transmitted as $-p(t)$, and binary ones are transmitted as $p(t)$, where $p(t)$ denotes an arbitrary pulse shape. The energy of the pulse is denoted by

$$E_b = \int_{-\infty}^{\infty} |p(t)|^2 dt . \tag{3.1}$$

These pulses are then transmitted over an additive white Gaussian noise channel (AWGN) with a two-sided mean power spectral density given by $S(f)=N_0/2$. On reception, the received signal is passed through a matched filter and sampled, which produces the sufficient statistics consisting of real-valued n-tuples, $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}$ for each transmitted codeword $\mathbf{X}$.

It is possible to decide on the symbol elements of the codeword sent at this point by comparing the value of each y_i to some threshold. However, some measure of the reliability of this decision is often desired. A suitable measure is the *a posteriori* probability (APP) of the transmitted bit. The APP is also referred to as a reliability estimate, denoted by $r_i = P[x_i=0|y_i]$. In other words, r_i is the probability that the ith bit transmitted is zero given the signal received in the ith interval.

The reliability estimates are then input to the MAP decoder which used the structure of the codeword to produce reliability estimates that take into account the coding . The hopefully improved reliability estimates output by the MAP decoder are denoted by the real-valued n-tuple, $R'=\{r_1', r_2', \ldots, r_n'\}$, where $r_i'=P[x_i=0|Y;C]$, and C represents the knowledge of the structure of the code.

The R′ estimates were then used to make the final decision on the transmitted bits, a_n', by performing hard decisions on the r_i'. Since the r_i' were in the form of the probabilities $P[x_i=0 \mid Y;C]$, it can be seen that the best decision-making rule is given by

$$a_i' = \begin{cases} 0, & r_i' > 0.5; \\ 1, & r_i' < 0.5. \end{cases} \tag{3.2}$$

3.3. Error Subfields

Error subfields are used to classify the bits of the received block according to the validity of the rows and columns (in a 2-D product code) as codewords. The process of identifying the subfields is illustrated using the example of a two-dimensional product code formed from identical $(n, n\text{-}1)$ even parity block codes.

The bits of the received block are grouped into three subfields, as described below

(i) No-Error subfield - the bits which are in a row and column which seems error-free (i.e., the parity equation for the rows and column of that bit are satisfied),

(ii) One-dimensional (1-D) error subfield - the bits in a row or column for which the parity equation of exactly one dimension indicates to contain an error, and

(iii) Two-dimensional (2-D) error subfield - those bits that are in a row and column for which the parity equations do not hold.

These divisions of the bits in a received block of data are appropriate to 2-D product codes. Should one have occasion to use a product code with possibly a greater number of dimensions, say m, then, in addition to the no-error subfield, one would be able to identify error subfields up to dimension m for use in the remapping process.

In Figure 3.2, the identification of the subfields is illustrated using a two-dimensional product code constructed from identical (4,3) parity codes in both dimensions. The subfields resulting from the occurrence of an error in the (2,2) position is shown. The single error in this position results in a parity failure in one of the parity equations for the rows and also in one of the parity equations for the columns.

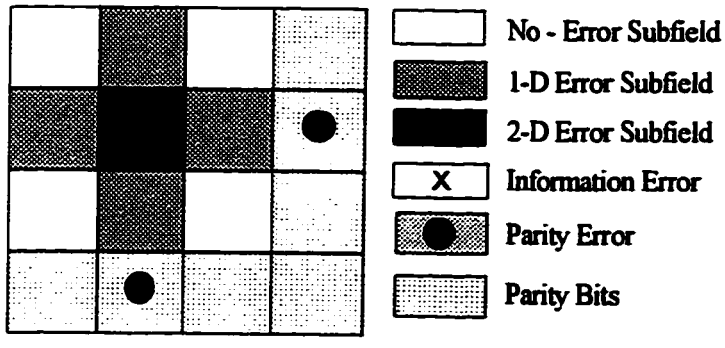


Figure 3.2. Example of subfield formation from a two dimensional product code with an error at element (2,2) of the matrix.

In an effort to determine the error behaviour of the subfields, simulations were carried out to determine the BER for the error subfields and the no-error subfield. The results are shown in Figure 3.3 below. The error statistics were collected for channel E_b/N_0 between 0 dB and 4 dB. Each simulation was performed under identical conditions, in that

(i) two-dimensional product codes were used,

(ii) each product code was constructed with (9,8) even parity codes in each dimension, and

(iii) a single cycle of MAP decoding was performed.

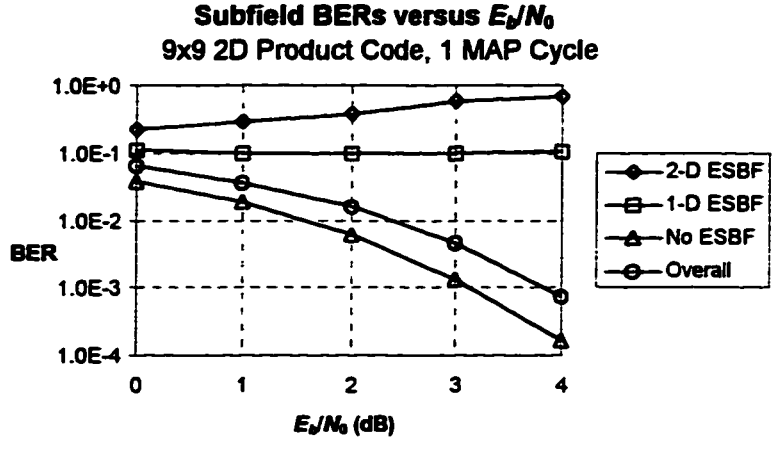


Figure 3.3. Graph of Subfield BER versus Channel E_b/N_0.

The 1-D error subfield can be seen to have a very flat BER between 0 dB and 4 dB, whereas the no-error subfield BER shows a marked decrease as the E_b/N_0 increases. This contrasts with the 2-D error subfield where the BER is found to actually increase with increasing E_b/N_0. The latter point is to be expected when we note that the BER of the overall code (which can be considered an average of the subfields) decreases with an increase in E_b/N_0. It is felt that these differences in behaviour of each of the subfields indicates that there is some exploitable information present given knowledge of the subfield.

3.4. Subfield BER in terms of Channel BER

Before going into the actual derivation of the subfield BERs as a function of the reliability estimate, it is instructive to show how the subfield BERs can be determined as a function of the base BER of the system. In order to simplify the following presentation we define the following variables:

p_e = probability that an arbitrarily chosen received bit is in error,

P[z] = probability that an arbitrarily chosen received bit is the no-error subfield,

P[u] = probability that an arbitrarily chosen received bit is the 1-D error subfield,

P[d] = probability that an arbitrarily chosen received bit is the 2-D error subfield,

P[t] = probability that an error arrangement of type t occurs,

P[α_x] = probability that on reception, there are x errors in the codeword, and

P[e] = probability that a decoded bit is in error.

We know that if we transmit n bits on a symmetric channel with error rate p_e, the probability of receiving the bits with x errors is given by the binomial distribution $\mathrm{P}[\alpha_x] = \binom{n}{x} \cdot (p_e)^x \cdot (1-p_e)^{n-x}$. This then is the probability that we should find x errors in a received word of length n prior to the application of the MAP algorithm.

The following probabilities can be found by considering the possible arrangements of errors in the received word and applying simple binomial theory:

(i) P[$t|\alpha_x$], the probability that arrangement t occurs when there are x errors,

(ii) P[$z|t;\alpha_x$], P[$u|t;\alpha_x$] and P[$d|t;\alpha_x$], the probability that a bit is in a given subfield given arrangement t and α_x errors, and

(iii) P[$e|z;\alpha_x;t$], P[$e|u;\alpha_x;t$] and P[$e|d;\alpha_x;t$], the probability that a bit is in error given that it is in a given subfield, that α_x errors have occurred and that arrangement t has occurred.

We wish to determine:

(i) P[$e|z$], the probability that a bit is in error given that the bit is in the No-Error subfield,

(ii) P[$e|u$], the probability that a bit is in error given that the bit is in the 1-D Error subfield, and

(iii) P[$e|d$], the probability that a bit is in error given that the bit is in the 2-D Error subfield.

Since the procedure for determining each of these probabilities is similar, we will consider the case of deriving P[e|z] from the known quantities as an example.

To begin with, we note that

$$\mathrm{P}[e|z]=\sum_{t,x}\mathrm{P}[e;t;\alpha_x|z]. \tag{3.3}$$

The summand in (3.3) can be expressed as

$$\mathrm{P}[e;t;\alpha_x|z]=\mathrm{P}[e|z;t;\alpha_x]\cdot\mathrm{P}[t;\alpha_x|z], \tag{3.4}$$

where

$$\mathrm{P}[t;\alpha_x|z]=\frac{\mathrm{P}[z;t;\alpha_x]}{\mathrm{P}[z]}=\frac{\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t;\alpha_x]}{\mathrm{P}[z]}. \tag{3.5}$$

We now consider the denominator in (3.5), the probability that a bit is in the No-Error Subfield, this is given by

$$\mathrm{P}[z]=\sum_{t,x}\mathrm{P}[z;t;\alpha_x]=\sum_{t,x}\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t;\alpha_x]=\sum_{t,x}\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]. \tag{3.6}$$

Therefore, replacing (3.6) in (3.5) and placing the resulting expression in (3.4) we get

$$\mathrm{P}[e;t;\alpha_x|z]=\mathrm{P}[e|z;t;\alpha_x]\cdot\frac{\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]}{\mathrm{P}[z]}=\frac{\mathrm{P}[e|z;t;\alpha_x]\cdot\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]}{\sum_{t,x}\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]}, \tag{3.7}$$

all the terms of which are known or are easily determined. Therefore P[e|z] can be expressed as

$$\mathrm{P}[e|z]=\frac{\sum_{t,x}\mathrm{P}[e|z;t;\alpha_x]\cdot\mathrm{P}[z|t;\alpha_x]\cdot\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]}{\sum_{t',x'}\mathrm{P}[z|t';\alpha_{x'}]\cdot\mathrm{P}[t'|\alpha_{x'}]\cdot\mathrm{P}[\alpha_{x'}]}. \tag{3.8}$$

Similarly, the 1-D Subfield error probabilities is given by

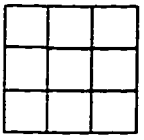
$$\mathrm{P}[e|u]=\frac{\sum_{t,x}\mathrm{P}[e|u;t;\alpha_x]\cdot\mathrm{P}[u|t;\alpha_x]\cdot\mathrm{P}[t|\alpha_x]\cdot\mathrm{P}[\alpha_x]}{\sum_{t',x'}\mathrm{P}[u|t';\alpha_{x'}]\cdot\mathrm{P}[t'|\alpha_{x'}]\cdot\mathrm{P}[\alpha_{x'}]}, \tag{3.9}$$

and the 2-D Subfield error probability by

$$P[e|d] = \frac{\sum_{t,x} P[e|d;t;\alpha_x] \cdot P[d|t;\alpha_x] \cdot P[t|\alpha_x] \cdot P[\alpha_x]}{\sum_{t',x'} P[d|t';\alpha_{x'}] \cdot P[t'|\alpha_{x'}] \cdot P[\alpha_{x'}]}. \tag{3.10}$$

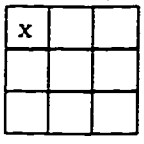
As an example, let us consider these results for a 3×3 product code. There are nine bit positions and thus 2^9 possible error arrangements. Fortunately, the error arrangements can be examined in groups so that there are much less than 2^9 cases to consider. Figure 3.4 below shows all the arrangements which are possible for each of the possible number of errors. For each of these arrangements, Table 3.1 gives their *a priori* probabilities and the probability of a bit being in a particular error subfield.

Case 1 : No Errors



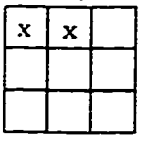
Arrangement 1
No errors in codeword

Case 2 : One Error

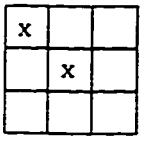


Arrangement 1
One error anywhere in codeword.

Case 3 : Two Errors

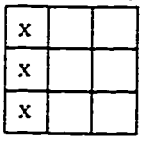


Arrangement 1
Two errors in the same row or column.

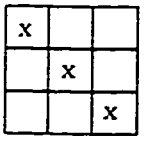


Arrangement 2
Two errors in different rows and columns.

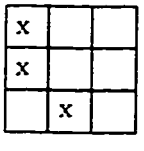
Case 4 : Three Errors



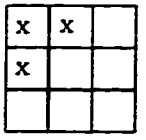
Arrangement 1
Three errors in same row or column.



Arrangement 2
Three errors in different rows and columns.

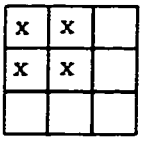


Arrangement 3
Two errors in a row/column, and the third error not in the row/column occupied by the first pair.

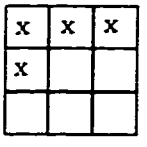


Arrangement 4
Two errors in a row, and the third error sharing a column with one of the first pair.

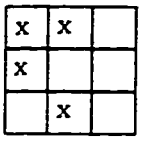
Case 5 : Four Errors



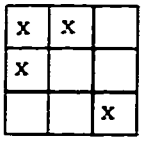
Arrangement 1
Two errors in one row, the other errors in another row and in the columns occupied by the first pair.



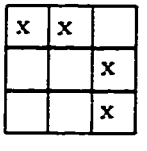
Arrangement 2
Three errors in row/column, and the other error anywhere.



Arrangement 3
Two errors in a row/column, and an error in each of the remaining rows/columns but in the same columns/rows as the first pair.



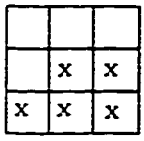
Arrangement 4
Two errors in a row/column, the other two in the remaining rows/columns, one of these in the column/row not occupied by the first pair.



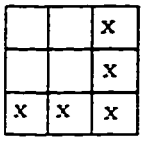
Arrangement 5
Two errors in a row, the other errors in the remaining rows and in the column not occupied by either of the first two.

Figure 3.4. (Part A) The possible arrangements of error subfields in a 3×3 product code.

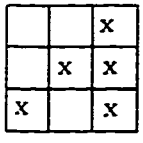
Case 6 : Five Errors



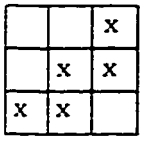
Arrangement 1
Three errors in a row, and the remaining two in a remaining row.



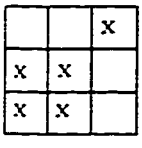
Arrangement 2
Three errors in a same row, and the remaining two in different rows but in the same column.



Arrangement 3
Three errors in the same row, and the remaining two in different rows and columns.

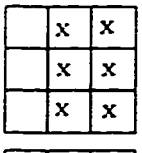


Arrangement 4
Two errors in row, two more in a remaining row but not in the columns of the first pair, and the fifth error in the remaining row but not sharing a column with the first pair.

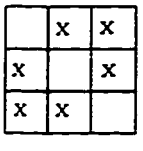


Arrangement 5
Two errors in a row, two in another row and in the same columns as the first pair, and the fifth error in a row and column not occupied by the first four.

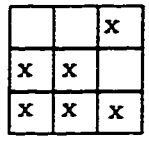
Case 7 : Six Errors



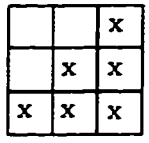
Arrangement 1
Three non-errors in a row.



Arrangement 2
Three non-errors each in a different row and column.

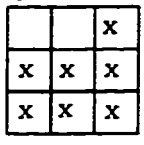


Arrangement 3
Two non-errors in the same row, and the other non-error in a remaining row and column.

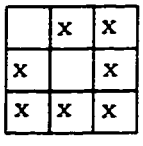


Arrangement 4
Two non-errors in the same row, and the other non-error in a remaining row and in a column occupied by one of the first pair.

Case 8 : Seven Errors

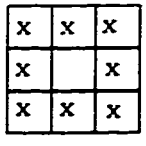


Arrangement 1
Two non-errors in same row.



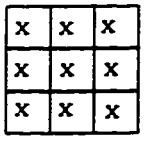
Arrangement 2
Two non-errors in different rows and columns.

Case 9 : Eight Errors



Arrangement 1
One non-error anywhere.

Case 10 : Nine Errors



Arrangement 1
Nine errors in dataword.

Figure 3.4. (Part B) The possible arrangements of error subfields in a 3×3 product code.

Table 3.1 Table showing the relevant probabilities for all possible arrangements of errors in a 3×3 product code.

# of Errors (α_x)	# of Permutations of Errors	Error Arrangement (t)	# of Instances of t	P[$t\|\alpha_x$]	P[$z\|\alpha_x;t$]	P[$u\|\alpha_x;t$]	P[$d\|\alpha_x;t$]
0	1	1	1	1	1	0	1
1	9	1	9	1	4/9	4/9	1/9
2	36	1	18	1/2	3/9	6/9	0
		2	18	1/2	1/9	4/9	4/9
3	84	1	6	1/14	0	6/9	3/9
		2	6	1/14	0	0	1
		3	36	3/7	0	6/9	3/9
		4	36	3/7	4/9	1/9	36/84
4	126	1	9	1/14	1	0	0
		2	36	2/7	1/9	4/9	4/9
		3	36	2/7	3/9	0	6/9
		4	36	2/7	3/9	0	6/9
		5	9	1/14	1/9	4/9	4/9
5	126	1	36	2/7	4/9	4/9	1/9
		2	9	1/14	0	0	1
		3	36	2/7	0	6/9	3/9
		4	36	2/7	4/9	4/9	1/9
		5	9	1/14	4/9	4/9	1/9
6	84	1	6	1/14	4/9	4/9	1/9
		2	6	1/14	1	0	0
		3	36	3/7	3/9	6/9	0
		4	36	3/7	1/9	4/9	4/9
7	36	1	18	1/2	0	6/9	3/9
		2	18	1/2	4/9	4/9	1/9
8	9	1	9	1	0	0	1
9	1	1	1	1	0	0	1

3.5. Subfield Error Probability given Reliability Estimate before MAP Processing

The preceding calculation of the probability of bit error conditioned on the subfield does not provide the information to allow remapping. Instead, for remapping we need to determine the probability of bit error conditioned on both the subfield in which the bit resides as well as the value of the reliability estimate produced by the iterated MAP algorithm. Before considering this after the MAP algorithm is applied, let us see how to work these error rates in the simpler situation we have before MAP remapping. The determination of the probability of bit error in the no error subfield given the reliability estimate, $P[e_i|z_i;r_i]$, will serve as an example of finding the probability of bit in the other subfield(s), where

(i) e_i denotes the event that an error occurs in the ith bit of the codeword,

(ii) z_i denotes the event that the ith bit is in the no-error subfield, and

(iii) r_i denotes the reliability estimate for the ith bit.

By Bayes' rule, $P[e_i|z_i;r_i]$ can be expressed as

$$P[e_i|z_i;r_i] = \frac{P[e_i;z_i|r_i]}{P[z_i|r_i]}. \tag{3.11}$$

Consider the numerator of the quotient in (3.11). We can equate the probability of the joint event e_i, z_i as the sum of the probabilities of the joint events e_i, z_i, α_k over the values of α_k, where α_k denotes the event that k bits of the n-bit codeword are in error. Therefore, $P[e_i|z_i;r_i]$ can be expressed as

$$\begin{aligned} P[e_i;z_i|r_i] &= \sum_{k=0}^{n} P[e_i;z_i;\alpha_k|r_i] = \sum_{k=0}^{n} P[z_i;\alpha_k|e_i;r_i]\cdot P[e_i|r_i] \\ &= \sum_{k=0}^{n} P[\alpha_k|z_i;e_i;r_i]\cdot P[z_i|e_i;r_i]\cdot P[e_i|r_i]. \end{aligned} \tag{3.12}$$

Consider the term $P[\alpha_k|z_i;e_i;r_i]$ in (3.12). Since e_i indicates that the ith bit is in error, the value of r_i is superfluous information (each r_i is independent of the others before MAP processing), thus

$$P[\alpha_k|z_i;e_i;r_i] = P[\alpha_k|z_i;e_i]. \tag{3.13}$$

The form of this probability is now changed to allow for easy determination of its value, i.e.,

$$P[\alpha_k|z_i;e_i] = \frac{P[\alpha_k;z_i|e_i]}{P[z_i|e_i]} = \frac{P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]}{\sum_{k=0}^{n} P[z_i;\alpha_k|e_i]} = \frac{P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]}{\sum_{k=0}^{n} P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]}, \tag{3.14}$$

where $P[z_i|\alpha_k;e_i]$ can be determined by straightforward enumeration for different α_k, and $P[\alpha_k|e_i]$ is given by the binomial distribution.

Consider the term $P[z_i|e_i;r_i]$ in (3.12). It is evident that this can be expressed as

$$P[z_i|e_i;r_i] = \sum_{k=0}^{n} P[z_i;\alpha_k|e_i;r_i] = \sum_{k=0}^{n} P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]. \tag{3.15}$$

Therefore, by using (3.14) and (3.15) in (3.12), the numerator of (3.11), $P[e_i;z_i|r_i]$, can be expressed as

$$\begin{aligned} P[e_i;z_i|r_i] &= \sum_{k=0}^{n} P[\alpha_k|z_i;e_i;r_i]\cdot P[z_i|e_i;r_i]\cdot P[e_i|r_i] \\ &= \sum_{k=0}^{n}\left(\frac{P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]}{\sum_{k=0}^{n} P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]}\right)\cdot\left(\sum_{k=0}^{n} P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i]\right)\cdot\left(P[e_i|r_i]\right) \\ &= P[e_i|r_i]\cdot\sum_{k=0}^{n} P[z_i|\alpha_k;e_i]\cdot P[\alpha_k|e_i] \end{aligned} \tag{3.16}$$

Now, consider the denominator of the quotient in (3.11), $P[z_i|r_i]$. This term can be expressed as

$$\mathrm{P}[z_i|r_i] = \sum_{k=0}^{n} \mathrm{P}[z_i;\alpha_k|r_i] = \sum_{k=0}^{n} \mathrm{P}[z_i|\alpha_k;r_i]\cdot\mathrm{P}[\alpha_k|r_i]. \tag{3.17}$$

The first term of the final summation in (3.17), $\mathrm{P}[z_i;\alpha_k|r_i]$ can be expressed as

$$\begin{aligned}\mathrm{P}[z_i|\alpha_k;r_i] &= \mathrm{P}[z_i;e_i|\alpha_k;r_i] + \mathrm{P}[z_i;\bar{e}_i|\alpha_k;r_i] \\ &= \mathrm{P}[z_i|e_i;\alpha_k;r_i]\cdot\mathrm{P}[e_i|\alpha_k;r_i] + \mathrm{P}[z_i|\bar{e}_i;\alpha_k;r_i]\cdot\mathrm{P}[\bar{e}_i|\alpha_k;r_i] \\ &= \mathrm{P}[z_i|e_i;\alpha_k]\cdot\mathrm{P}[e_i|\alpha_k;r_i] + \mathrm{P}[z_i|\bar{e}_i;\alpha_k]\cdot\mathrm{P}[\bar{e}_i|\alpha_k;r_i],\end{aligned} \tag{3.18}$$

where $\bar{e}_i$ represents the event that the ith bit is not in error. As mentioned previously $\mathrm{P}[z_i|\alpha_k;e_i]$ can be determined by straightforward enumeration and likewise so can $\mathrm{P}[z_i|\alpha_k;\bar{e}_i]$.

Now,

$$\begin{aligned}\mathrm{P}[e_i|\alpha_k;r_i] &= \frac{\mathrm{P}[e_i;\alpha_k|r_i]}{\mathrm{P}[\alpha_k|r_i]} = \frac{\mathrm{P}[\alpha_k|r_i;e_i]\cdot\mathrm{P}[e_i|r_i]}{\mathrm{P}[\alpha_k|r_i]} \\ &= \frac{\mathrm{P}[\alpha_k|e_i]\cdot\mathrm{P}[e_i|r_i]}{\mathrm{P}[\alpha_k|r_i]},\end{aligned} \tag{3.19}$$

and similarly,

$$\begin{aligned}\mathrm{P}[\bar{e}_i|\alpha_k;r_i] &= \frac{\mathrm{P}[\bar{e}_i;\alpha_k|r_i]}{\mathrm{P}[\alpha_k|r_i]} = \frac{\mathrm{P}[\alpha_k|r_i;\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r_i]}{\mathrm{P}[\alpha_k|r_i]} \\ &= \frac{\mathrm{P}[\alpha_k|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r_i]}{\mathrm{P}[\alpha_k|r_i]}.\end{aligned} \tag{3.20}$$

Therefore (3.17) becomes

$$\begin{aligned}\mathrm{P}[z_i|r_i] &= \sum_{k=0}^{n}\mathrm{P}[\alpha_k|r_i]\cdot\left(\mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[e_i|\alpha_k;r] + \mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[e_i|\alpha_k;r]\right) \\ &= \sum_{k=0}^{n}\mathrm{P}[\alpha_k|r_i]\cdot\left(\frac{\mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i;r]\cdot\mathrm{P}[e_i|r] + \mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|\bar{e}_i;r]\cdot\mathrm{P}[\bar{e}_i|r]}{\mathrm{P}[\alpha_k|r_i]}\right) \\ &= \sum_{k=0}^{n}\mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i;r]\cdot\mathrm{P}[e_i|r] + \mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|\bar{e}_i;r]\cdot\mathrm{P}[\bar{e}_i|r].\end{aligned} \tag{3.21}$$

Thus, the No Error subfield error probability given the reliability estimate, $\mathrm{P}[e_i|z_i;r_i]$, is given by

$$\mathrm{P}[e_i|z_i;r_i]=\frac{\mathrm{P}[e_i|r_i]\cdot\sum_{k=0}^{n}\mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]}{\sum_{k=0}^{n}\mathrm{P}[z_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]\cdot\mathrm{P}[e_i|r_i]+\mathrm{P}[z_i|\alpha_k;\bar{e}_i]\cdot\mathrm{P}[\alpha_k|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r_i]}. \tag{3.22}$$

Similarly, the conditional subfield error probabilities of the 1-D and 2-D error subfields can be expressed as

$$\mathrm{P}[e_i|u_i;r_i]=\frac{\mathrm{P}[e_i|r_i]\cdot\sum_{k=0}^{n}\mathrm{P}[u_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]}{\sum_{k=0}^{n}\mathrm{P}[u_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]\cdot\mathrm{P}[e_i|r_i]+\mathrm{P}[u_i|\alpha_k;\bar{e}_i]\cdot\mathrm{P}[\alpha_k|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r_i]}, \tag{3.23}$$

and

$$\mathrm{P}[e_i|d_i;r_i]=\frac{\mathrm{P}[e_i|r_i]\cdot\sum_{k=0}^{n}\mathrm{P}[d_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]}{\sum_{k=0}^{n}\mathrm{P}[d_i|\alpha_k;e_i]\cdot\mathrm{P}[\alpha_k|e_i]\cdot\mathrm{P}[e_i|r_i]+\mathrm{P}[d_i|\alpha_k;\bar{e}_i]\cdot\mathrm{P}[\alpha_k|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r_i]} \tag{3.24}$$

respectively.

Note that the only terms which are dependent on the transmission E_b/N_0 are $\mathrm{P}[\alpha_k|e_i]$ and $\mathrm{P}[\alpha_k|\bar{e}_i]$ through their use of p_e, the probability of bit error. The other terms are fixed for a given code. So, if it is desired to determine the subfield BER at different E_b/N_0, we only need to find p_e. With MAP processing things are much more complicated.

Example 3.1. The (3,2) Even Parity Code

Determining the conditional subfield BER of the (3,2) even parity code now serves as a simple example illustrating the result above. The probability of k errors occurring given that there is, or is not, an error in position i is easily seen to be given by the probabilities of the k errors in 2 positions or the probability of k-1 errors in 2 positions:

$$\mathrm{P}[\alpha_k|e_i]=\begin{cases}0 & ;k=0\\ \binom{2}{k-1}(p_e)^{k-1}(1-p_e)^{3-k} & ;k=1,2,3\end{cases} \tag{3.25}$$

and

$$\mathrm{P}[\alpha_k|\bar{e}_i]=\begin{cases}\binom{2}{k}(p_e)^k(1-p_e)^{2-k} & ;0\le k\le 2\\ 0 & ;k=3\end{cases} \tag{3.26}$$

respectively. These equations are easily understood when one considers the fact that they represent the probability of k-1 errors occurring in the remaining n-1 positions of the codeword.

Table 3.2. Subfield probabilities given k errors and an error, or not, in position i, for a (3,2) even parity code.

k	$\mathrm{P}[z_i\|\alpha_k;e_i]$	$\mathrm{P}[z_i\|\alpha_k;\bar{e}_i]$	$\mathrm{P}[u_i\|\alpha_k;e_i]$	$\mathrm{P}[u_i\|\alpha_k;\bar{e}_i]$
0	-	1.0	-	0.0
1	0.0	0.0	1.0	1.0
2	1.0	1.0	0.0	0.0
3	0.0	-	1.0	-

The No-Error Subfield error rate conditioned on the reliability estimate is therefore given by

$$\begin{aligned}\mathrm{P}[e|z_i;r_i]&=\frac{\mathrm{P}[z_i|\alpha_2;e_i]\cdot\mathrm{P}[\alpha_2|e_i]\cdot\mathrm{P}[e_i|r]}{\mathrm{P}[z_i|\alpha_2;e_i]\cdot\mathrm{P}[\alpha_2|e_i]\cdot\mathrm{P}[e_i|r]+\mathrm{P}[z_i|\alpha_0;\bar{e}_i]\cdot\mathrm{P}[\alpha_0|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r]+\mathrm{P}[z_i|\alpha_2;\bar{e}_i]\cdot\mathrm{P}[\alpha_2|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r]}\\&=\frac{\mathrm{P}[\alpha_2|e_i]\cdot\mathrm{P}[e_i|r]}{\mathrm{P}[\alpha_2|e_i]\cdot\mathrm{P}[e_i|r]+\mathrm{P}[\alpha_0|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r]+\mathrm{P}[\alpha_2|\bar{e}_i]\cdot\mathrm{P}[\bar{e}_i|r]}\\&=\frac{2p_e(1-p_e)\cdot\mathrm{P}[e_i|r]}{2p_e(1-p_e)\cdot\mathrm{P}[e_i|r]+\left((1-p_e)^2+p_e^2\right)\cdot\mathrm{P}[\bar{e}_i|r]}.\end{aligned} \tag{3.27}$$

The corresponding expression for the 1-D error subfield is

$$\begin{aligned}\mathrm{P}[e|u_i;r_i]&=\frac{\mathrm{P}[e_i|r]\cdot\left(\mathrm{P}[\alpha_1|e_i]+\mathrm{P}[\alpha_3|e_i]\right)}{\mathrm{P}[e_i|r]\cdot\left(\mathrm{P}[\alpha_1|e_i]+\mathrm{P}[\alpha_3|e_i]\right)+\mathrm{P}[\bar{e}_i|r]\cdot\mathrm{P}[\alpha_1|\bar{e}_i]}\\&=\frac{\mathrm{P}[e_i|r]\cdot\left(1+p_e^2\right)}{\mathrm{P}[e_i|r]\cdot\left(1+p_e^2\right)+2p_e\cdot(1-p_e)\cdot\mathrm{P}[\bar{e}_i|r]}.\end{aligned} \tag{3.28}$$

The results are shown in Figures 3.5 and 3.7. The scenario analyzed was also simulated with the results shown in Figures 3.6 and 3.8. It can be seen that the analytically derived subfield BER conditioned on reliability estimates is a good match compared to those found by simulation.

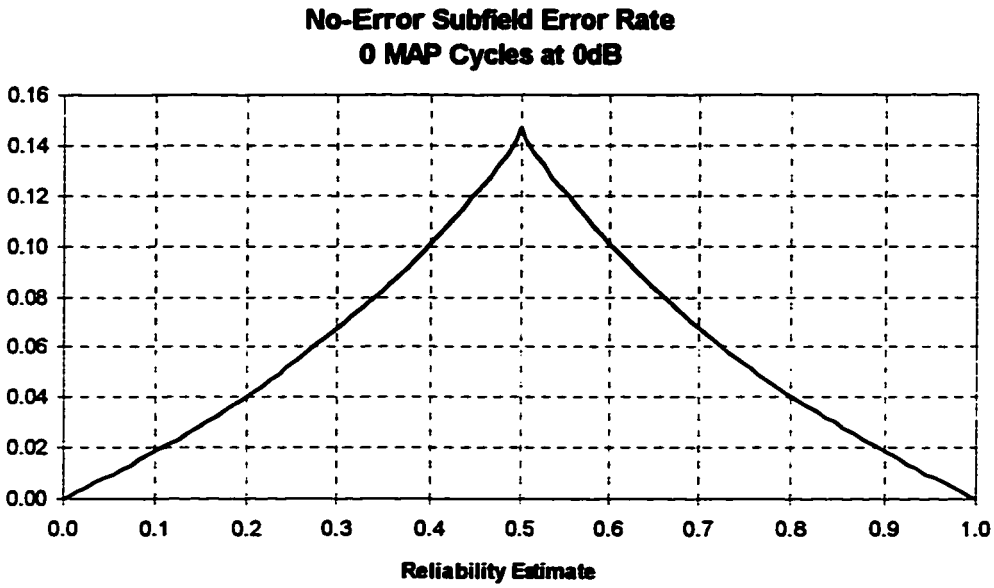


Figure 3.5. Plot of the expected No-Error Subfield BER versus Reliability Estimate before MAP processing at an E_b/N_0 of 0 dB.

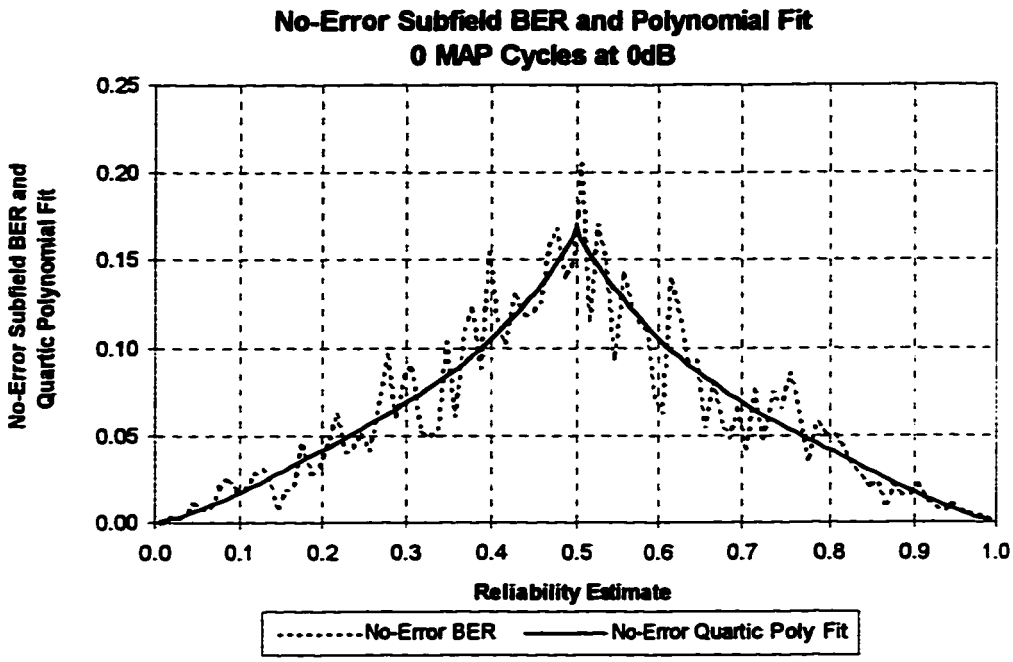


Figure 3.6. Plot of the empirical No-Error Subfield BER versus Reliability Estimate before MAP processing at an E_b/N_0 of 0dB.

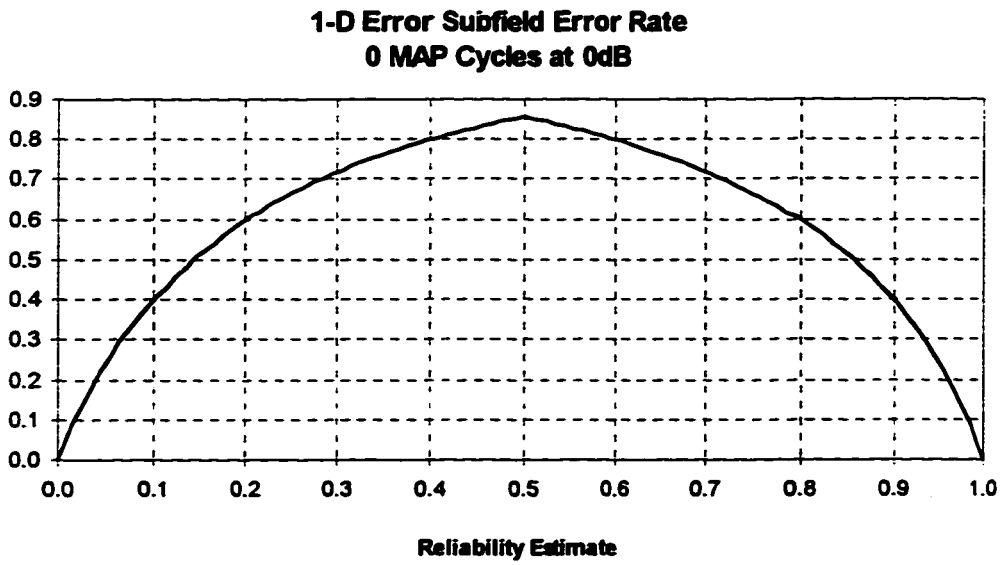


Figure 3.7. Plot of the expected 1-D subfield BER versus Reliability Estimate before MAP processing at an E_b/N_0 of 0dB.

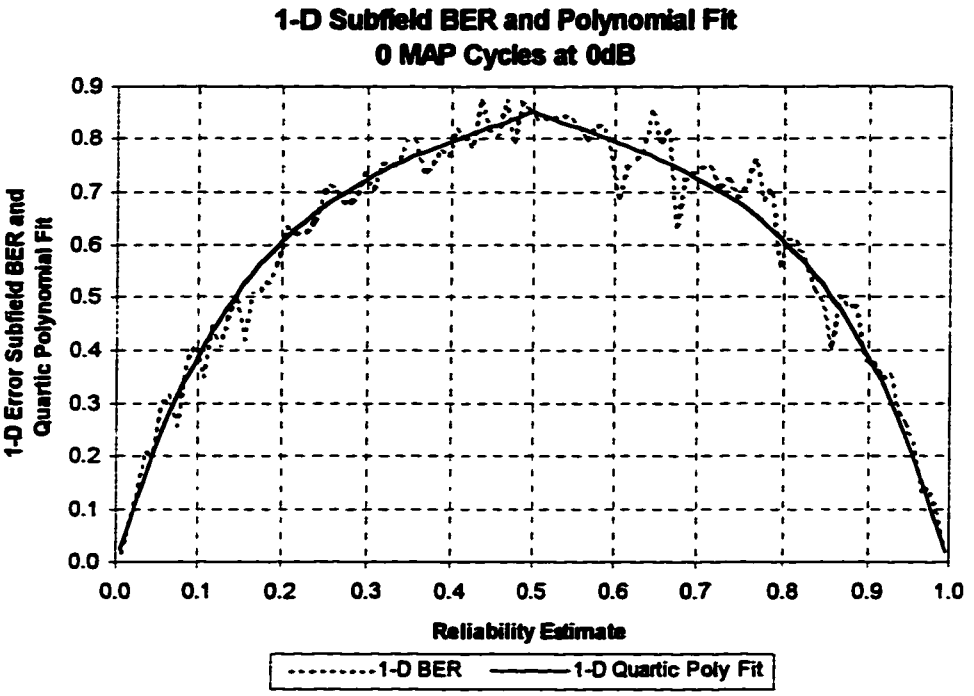


Figure 3.8. Plot of the empirical 1-D subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

Example 3.2. The 3×3 Product Code

Consider the case of the 2-dimensional product code with (3,2) even parity codes as component codes in each dimension. The expressions for the probability of k errors occurring given that there is, or is not, an error in position i are shown below in (3.29) and (3.30) respectively,

$$\mathrm{P}[\alpha_k|e_i] = \begin{cases} 0 & ;k=0 \\ \binom{8}{k-1}(p_e)^{k-1}(1-p_e)^{9-k} & ;1 \le k \le 9 \end{cases} \tag{3.29}$$

$$\mathrm{P}[\alpha_k|\bar{e}_i] = \begin{cases} \binom{8}{k}(p_e)^{k}(1-p_e)^{8-k} & ;0 \le k \le 8 \\ 0 & ;k=9 \end{cases} \tag{3.30}$$

Next, we would like to determine the probability that an arbitrary selected bit is in a particular subfield given that a specific number of errors has occurred. To illustrate the process by which we arrived at these values, we shall explicitly show how we derive the probability that a bit in error (or not) is in the one-dimensional error subfield given that two errors have occurred. These probabilities are denoted by $P[u_i|\alpha_2;e_i]$ and $P[u_i|\alpha_2;\bar{e}_i]$ respectively. When two errors occur, there are two possible subfield arrangements; these are shown in Figure 3.9.

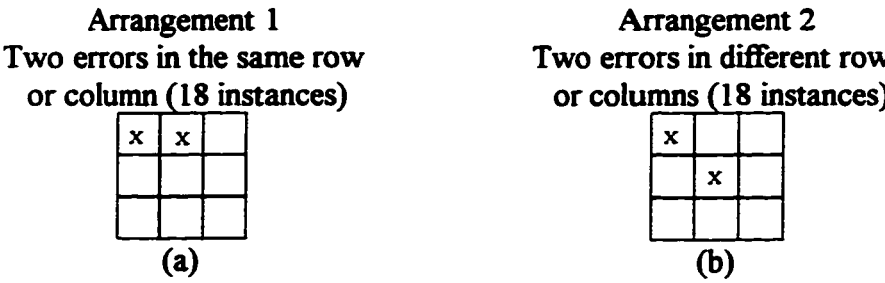


Figure 3.9. The two possible arrangements of double errors in a 3×3 product code block and illustrative instances of each.

If we let t denote the error arrangement, $P[u_i|\alpha_2;e_i]$ can be expressed as

$$\begin{aligned} P[u_i|\alpha_2;e_i] &= \sum_{t=1,2} P[u_i;t|\alpha_2;e_i] = \sum_{t=1,2} P[u_i|t;\alpha_2;e_i]\cdot P[t|\alpha_2;e_i] \\ &= \sum_{t=1,2} P[u_i|t;\alpha_2;e_i]\cdot P[t|\alpha_2]. \end{aligned} \tag{3.31}$$

There are 36 possible permutations of 2 errors in the 9 bit codeword. Of the 36, 18 permutations correspond to each of arrangements 1 and 2. Therefore

$$P[t=1|\alpha_2] = P[t=2|\alpha_2] = 18/36 = 0.5. \tag{3.32}$$

Consider $P[u_i|t=1;\alpha_2;e_i]$. From the example of the arrangements depicted in Figure 3.9(a), it can be seen that the one-dimensional error subfield consists of the two columns containing the errors. More generally, for any of the arrangements represented by Figure 3.9(a), both errors lie in the one-dimensional error subfield. Clearly, given arrangement 1 of the two errors as per Figure 3.9(a), the probability that an error is in the one-dimensional error subfield is $P[u_i|t=1;\ \alpha_2;e_i]=1$.

Consider $P[u_i|t=2;\alpha_2;e_i]$. From the arrangements described by Figure 3.9(b), it can be seen that the one dimensional error subfield does not contain errors. Therefore $P[u_i|t=2;\alpha_2;e_i]=0$.

From this it follows that

$$\begin{aligned} P[u_i|\alpha_2;e_i] &= \sum_{t=1,2} P[u_i;t|\alpha_2;e_i] \\ &= P[u_i|t=1;\alpha_2;e_i]\cdot P[t=1|\alpha_2]+P[u_i|t=2;\alpha_2;e_i]\cdot P[t=2|\alpha_2] \\ &= (0.5)\cdot(1)+(0.5)\cdot(0) \\ &= 0.5. \end{aligned} \tag{3.33}$$

Now, we see from Figure 3.9(a) above, that of the 7 bits not in error, 4 are in the one-dimensional error subfield, therefore $P[u_i|t=1;\alpha_2;\bar{e}_i]=4/7$. The same is true of $P[u_i|t=2;e_2;\bar{e}_i]$ from Figure 3.9(b). Therefore

$$\begin{aligned} P[u_i|\alpha_2;\bar{e}_i] &= P[u_i|t=1;\alpha_2;\bar{e}_i]\cdot P[t=1|\alpha_2]+P[u_i|t=2;\alpha_2;\bar{e}_i]\cdot P[t=2|\alpha_2] \\ &= (0.5)\cdot(4/7)+(0.5)\cdot(4/7) \\ &= 0.571. \end{aligned} \tag{3.34}$$

The other probabilities are derived similiarly, and are listed in Table 3.3 below.

Table 3.3. Subfield probabilities given k errors and an error (or not) in position i, for a 3×3 product code with (3,2) even parity codewords in each dimension.

k	$P[z_i\|\alpha_k;e_i]$	$P[z_i\|\alpha_k;\bar{e}_i]$	$P[u_i\|\alpha_k;e_i]$	$P[u_i\|\alpha_k;\bar{e}_i]$	$P[d_i\|\alpha_k;e_i]$	$P[d_i\|\alpha_k;\bar{e}_i]$
0	-	1	-	0.0	-	0.0
1	0.0	0.556	0.0	0.5	1.0	0.0
2	0.0	0.286	0.5	0.571	0.5	0.143
3	0.143	0.214	0.381	0.243	0.095	0.143
4	0.777	0.168	0.429	0.457	0.214	0.343
5	0.429	0.214	0.457	0.929	0.143	0.286
6	0.762	0.25	0.452	0.381	0.143	0.286
7	0.25	0.25	0.571	0.5	0.571	0.0
8	0.0	1	0.5	0.0	0.5	0.0
9	0.0	-	0.0	-	1.0	-

The resulting analytic plots are shown in Figures 3.10, 3.12 and 3.14 for the 2-D, 1-D and No-Error Subfields respectively. The corresponding plots determined by simulation are shown in Figures 3.11, 3.13 and 3.15 for comparison. Again we see a very close correlation between the analytic and empirical results.

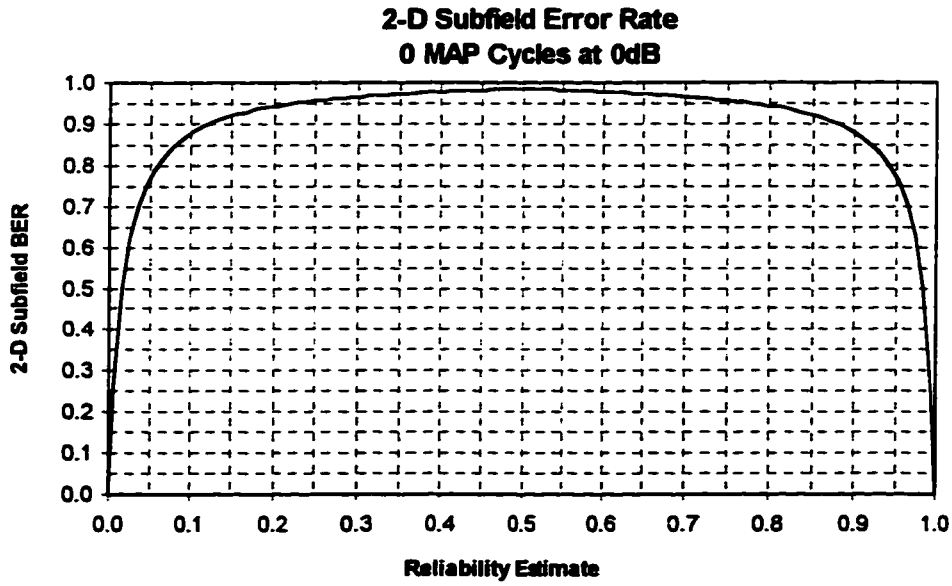


Figure 3.10. Plot of the expected 2-D Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0dB.

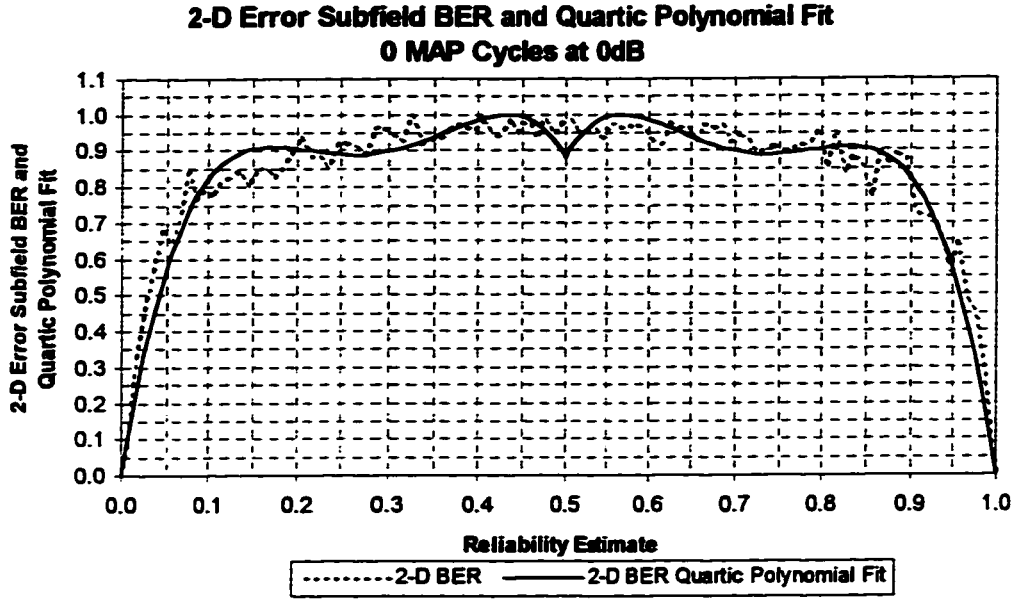


Figure 3.11. Plot of the empirical 2-D Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

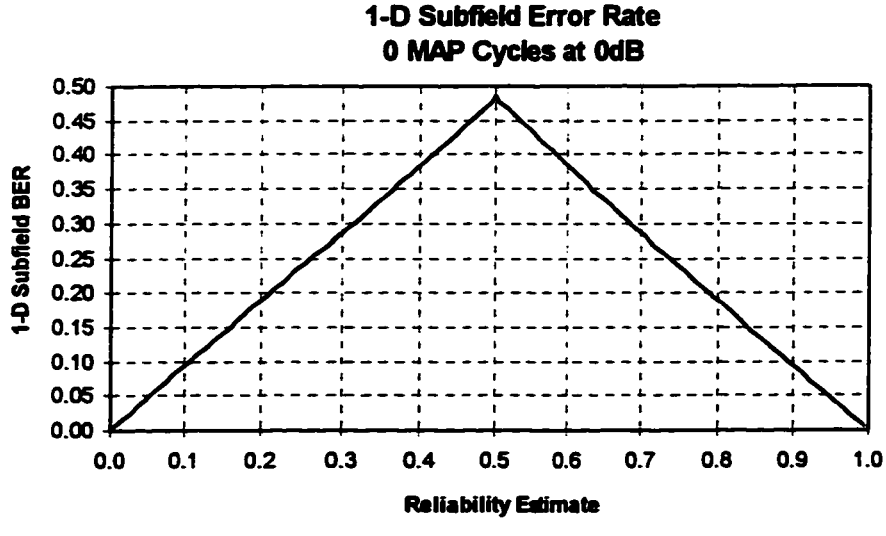


Figure 3.12. Plot of the expected 1-D Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

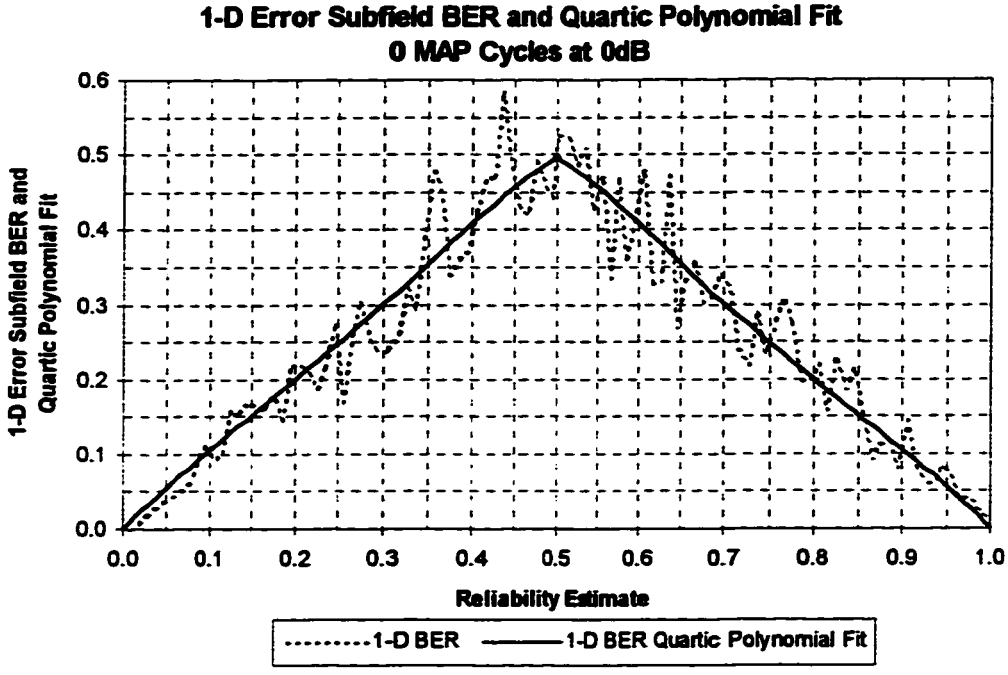


Figure 3.13. Plot of the empirical 1-D Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

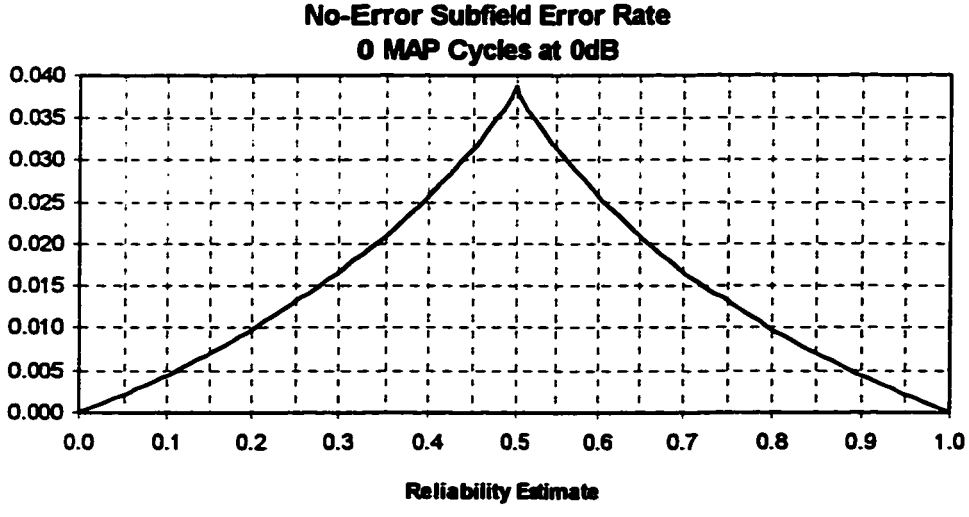


Figure 3.14. Plot of the expected No-Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

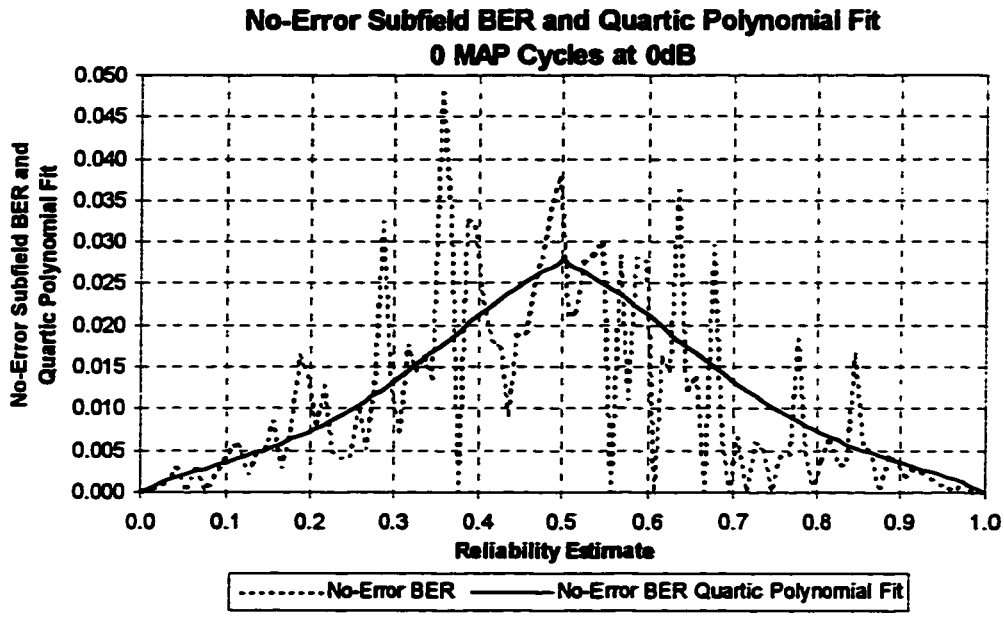


Figure 3.15. Plot of the empirical No-Error Subfield BER given the reliability estimate before MAP processing at an E_b/N_0 of 0 dB.

3.6. Subfield Error Probability given Reliability Estimate for a (3,2) Parity Code after MAP Processing

Let us now turn to the task of determining the relationship between reliability estimates and subfield position and the error rate after application of a single cycle of the MAP algorithm. To illustrate this we shall consider a simple situation provided by using a simple (3,2) even parity code (no product code used). In this case, Y will be a 3-tuple where $y_1=a_1$, $y_2=a_2$, and y_3 is the parity bit serving to make the sum of the number of ones in the codeword Y an even number. The reliability estimates, r_i, prior to any MAP processing, are calculated as before. The MAP process is then applied to adjust these change to reflect the new structure of the code. The value of the reliability estimates after one cycle of MAP processing is given by

$$r_1' = g_1(r_1,r_2,r_3) = \frac{r_1 \cdot r_2 \cdot r_3 + r_1 \cdot \bar{r}_2 \cdot \bar{r}_3}{r_1 \cdot r_2 \cdot r_3 + r_1 \cdot \bar{r}_2 \cdot \bar{r}_3 + \bar{r}_1 \cdot \bar{r}_2 \cdot r_3 + \bar{r}_1 \cdot r_2 \cdot \bar{r}_3}, \tag{3.35}$$

$$r_2' = g_2(r_1,r_2,r_3) = \frac{r_1 \cdot r_2 \cdot r_3 + \bar{r}_1 \cdot r_2 \cdot \bar{r}_3}{r_1 \cdot r_2 \cdot r_3 + r_1 \cdot \bar{r}_2 \cdot \bar{r}_3 + \bar{r}_1 \cdot \bar{r}_2 \cdot r_3 + \bar{r}_1 \cdot r_2 \cdot \bar{r}_3}, \tag{3.36}$$

and

$$r_3' = g_3(r_1,r_2,r_3) = \frac{r_1 \cdot r_2 \cdot r_3 + \bar{r}_1 \cdot \bar{r}_2 \cdot r_3}{r_1 \cdot r_2 \cdot r_3 + r_1 \cdot \bar{r}_2 \cdot \bar{r}_3 + \bar{r}_1 \cdot \bar{r}_2 \cdot r_3 + \bar{r}_1 \cdot r_2 \cdot \bar{r}_3}, \tag{3.37}$$

where $\bar{r}_i = 1 - r_i$.

If we let $x_1 = r_1 \cdot r_2 \cdot r_3$, $x_2 = r_1 \cdot \bar{r}_2 \cdot \bar{r}_3$, $x_3 = \bar{r}_1 \cdot r_2 \cdot \bar{r}_3$ and $x_4 = \bar{r}_1 \cdot \bar{r}_2 \cdot r_3$, then (3.35) - (3.37) can be expressed as

$$r_1' = \frac{x_1 + x_2}{x_1 + x_2 + x_3 + x_4}, \tag{3.38}$$

$$r_2' = \frac{x_1 + x_3}{x_1 + x_2 + x_3 + x_4}, \tag{3.39}$$

and

$$r_3' = \frac{x_1 + x_4}{x_1 + x_2 + x_3 + x_4}. \tag{3.40}$$

These simple relationships can be used to allow us to determine the possible values for (r_1', r_2', r_3').

The values of x_1, x_2, x_3 and x_4 giving rise to a given set of values for r_1', r_2', r_3' are found to be a one dimensional set which can be expressed in term of x_1 as

$$x_2 = \frac{1 + r_1' - r_2' - r_3'}{-1 + r_1' + r_2' + r_3'} \cdot x_1, \tag{3.41}$$

$$x_3 = \frac{1 - r_1' + r_2' - r_3'}{-1 + r_1' + r_2' + r_3'} \cdot x_1, \tag{3.42}$$

and

$$x_4 = \frac{1 - r_1' - r_2' + r_3'}{-1 + r_1' + r_2' + r_3'} \cdot x_1, \tag{3.43}$$

where x_i can take any value in [0,1]. Replacing the x_i with their expansion in terms of r_i and $\bar{r}_i$ in (3.41), we find

$$r_1 \cdot \bar{r}_2 \cdot \bar{r}_3 = \frac{1 + r_1' - r_2' - r_3'}{-1 + r_1' + r_2' + r_3'} \cdot r_1 \cdot r_2 \cdot r_3. \tag{3.44}$$

Letting $b_i = \bar{r}_i / r_i$, (3.44) becomes

$$b_2 \cdot b_3 = \frac{1 + r_1' - r_2' - r_3'}{-1 + r_1' + r_2' + r_3'}. \tag{3.45}$$

Similarly, (3.42) and (3.43) become

$$b_1 \cdot b_2 = \frac{1 - r_1' + r_2' - r_3'}{-1 + r_1' + r_2' + r_3'} \tag{3.46}$$

and

$$b_1 \cdot b_3 = \frac{1 - r_1' - r_2' + r_3'}{-1 + r_1' + r_2' + r_3'}. \tag{3.47}$$

Since $r_i \in [0, 1]$, we note that

$$0 \le b_i \le \infty. \tag{3.48}$$

From this, we can determine the regions of the r_i' for which the expressions (3.45) through (3.47) can be valid. We see that there are two sets of conditions which satisfy (3.48), these are that both the numerator and denominator are non-negative or that they are both negative, i.e.,

$$r_1' - r_2' - r_3' \ge -1 \text{ and } r_1' + r_2' + r_3' \ge 1 \tag{3.49}$$

or

$$r_1' - r_2' - r_3' < -1 \text{ and } r_1' + r_2' + r_3' < 1. \tag{3.50}$$

Similarly, (3.46) yields the following conditions

$$-r_1' + r_2' + r_3' \ge -1 \text{ and } r_1' + r_2' + r_3' \ge 1 \tag{3.51}$$

or

$$-r_1' + r_2' + r_3' < 1 \text{ and } r_1' + r_2' + r_3' < 1; \tag{3.52}$$

and (3.47)

$$-r_1' - r_2' + r_3' \ge -1 \text{ and } r_1' + r_2' + r_3' \ge 1 \tag{3.53}$$

or

$$-r_1' - r_2' + r_3' < 1 \text{ and } r_1' + r_2' + r_3' < 1. \tag{3.54}$$

Upon further examination of the conditions in (3.50), (3.52) and (3.54)–which correspond to the numerator and denominator both being negative–we find that these can only be satisfied when r_1' is less than zero. Since we know that this is not possible, the set r_1', r_2', r_3' of satisfying any of these conditions is empty, so we need only consider further the conditions given in (3.49), (3.51) and (3.53). We find that these conditions restrict the

possible values of the r_i' to the intersection of four half-solid sections which form the interior and surface of the tetrahedron shown in the figure below.

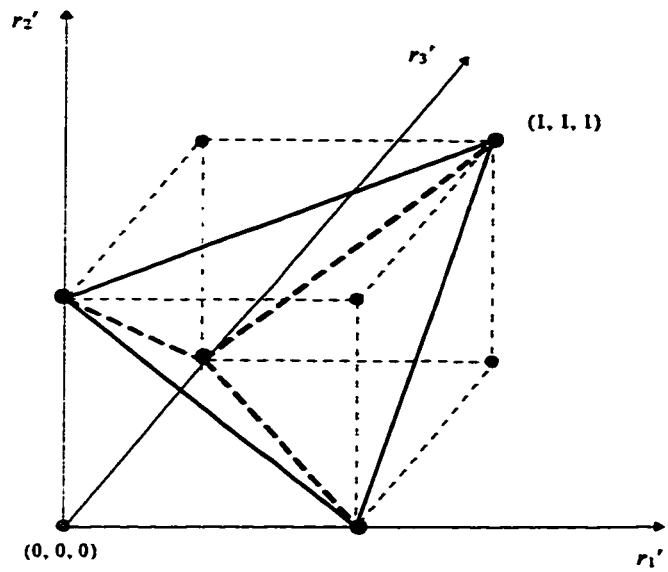


Figure 3.16 Figure showing the region of 3-dimensional space where r_1', r_2', r_3' may lie. The heavy shaded points are the vertices of the tetrahedron.

To determine the probability distribution of (r_1',r_2',r_3'), we are interested in determining the value of (r_1,r_2,r_3) that are mapped to a given point (r_1',r_2',r_3') by the MAP filter. Consider first a point on a surface of the tetrahedral region.

Consider the plane represented by $r_1'+r_2'+r_3'=1$. We would like to determine the values of r_1, r_2, r_3 (or b_1, b_2, b_3) that give rise to a position on this plane. Substituting for the r_i' in (3.38) to (3.40) we see that this results in

$$\frac{3x_1+x_2+x_3+x_4}{x_1+x_2+x_3+x_4}=1, \tag{3.55}$$

which requires that $x_1=r_1 \cdot r_2 \cdot r_3=0$. We see that one, or more, of the r_i is zero.

Let us say that r_i is zero. This means that x_1 and $x_{\alpha+1}$ equal zero. From (3.38) to (3.40) we see that this implies that r_α' is zero. Dividing the equations we get

$$\frac{r'_\beta}{r'_\gamma} = \frac{x_{\beta+1}}{x_{\gamma+1}} = \frac{r_\beta \cdot \bar{r}_\gamma}{r_\gamma \cdot \bar{r}_\beta} = \frac{b_\beta}{b_\gamma}, \tag{3.56}$$

with the final result that

$$b_\beta = \frac{r'_\beta}{r'_\gamma} \cdot b_\gamma . \tag{3.57}$$

We therefore see that there is no unique set of b_1, b_2, b_3 and hence no set of r_1, r_2, r_3 that corresponds to a set of r_1', r_2', r_3' in the case of $r_1'+r_2'+r_3'=1$. Fortunately, however, since we need one of the r_i to be zero to satisfy the condition, and the probability of this happening is zero, the probability of the condition occurring is also zero and it can be ignored in the task of determining the joint density function of (r_1',r_2',r_3'). In a similar manner, we find that the set r_1, r_2, r_3 corresponding to a given point on the surface of the tetrahedron is a one dimension set, which has zero probability. Therefore, we only need to integrate within the tetrahedron, and can exclude the surface in our subsequent working. This is fortunate, as we were unable to successfully integrate numerically any of the functions developed below, when the surface of the figure was included.

Solving (3.45) to (3.47) for the b_i for points on the interior of the tetrahedron, we get

$$b_1 = \sqrt{\frac{(1-r_1'+r_2'-r_3')\cdot(1-r_1'-r_2'+r_3')}{(-1+r_1'+r_2'+r_3')(1+r_1'-r_2'-r_3')}} = h_1(r_1',r_2',r_3'), \tag{3.58}$$

$$b_2 = \sqrt{\frac{(1-r_1'-r_2'+r_3')\cdot(1+r_1'-r_2'-r_3')}{(-1+r_1'+r_2'+r_3')(1-r_1'+r_2'-r_3')}} = h_2(r_1',r_2',r_3'), \tag{3.59}$$

and

$$b_3 = \sqrt{\frac{(1-r_1'+r_2'-r_3')\cdot(1+r_1'-r_2'-r_3')}{(-1+r_1'+r_2'+r_3')(1-r_1'-r_2'+r_3')}} = h_3(r_1',r_2',r_3'). \tag{3.60}$$

We wish to find the joint conditional probability density function of the r_i' in the case where **X=0**. The joint conditional probability density function is given by

$$f_{R|X=0}(r_1';r_2';r_3') = f_{B|X=0}\left(h_1(r_1',r_2',r_3');h_2(r_1',r_2',r_3');h_3(r_1',r_2',r_3')\right)\cdot\left|J(r_1',r_2',r_3')\right|, \tag{3.61}$$

where the function $J(r_1',r_2',r_3')$ denotes the Jacobian determinant of the transformation:

$$J(r_1',r_2',r_3')=\det\begin{bmatrix}\frac{\partial b_1}{\partial r_1'} & \frac{\partial b_1}{\partial r_2'} & \frac{\partial b_1}{\partial r_3'}\\ \frac{\partial b_2}{\partial r_1'} & \frac{\partial b_2}{\partial r_2'} & \frac{\partial b_2}{\partial r_3'}\\ \frac{\partial b_3}{\partial r_1'} & \frac{\partial b_3}{\partial r_2'} & \frac{\partial b_3}{\partial r_3'}\end{bmatrix}. \tag{3.62}$$

The Jacobian determinant is found to be

$$J(r_1',r_2',r_3')=\frac{4}{\sqrt{(-1+r_1'+r_2'+r_3')^5(1-r_1'-r_2'+r_3')(1-r_1'+r_2'-r_3')(1+r_1'-r_2'-r_3')}}. \tag{3.63}$$

To find the joint density of (b_1,b_2,b_3), we note (as derived in Appendix C) that the conditional probability density function of the b_i is given by

$$\begin{aligned} f_{B|X_i=0}(b)&=\begin{cases}\frac{1}{a\sqrt{\pi N_0}\cdot b}\cdot\exp\left[-\left(\frac{\ln(b)+a\sqrt{E_b}}{a\sqrt{N_0}}\right)^2\right], & b>0;\\ 0, & b<0;\end{cases}\\ &=\begin{cases}\frac{1}{4\sqrt{\pi}\cdot\gamma b}\cdot\exp\left[-\left(\frac{\ln(b)+4\gamma^2}{4\gamma}\right)^2\right], & b>0;\\ 0, & b<0;\end{cases}\end{aligned} \tag{3.64}$$

where $a=4\sqrt{E_b}/N_0$ and $\gamma=\sqrt{E_b/N_0}$.

The joint conditional probability joint density function of the b_i, $f_{B|X=0}(b_1,b_2,b_3)$, is simply the product of the individual density functions of the b_i, since the elements of the received symbol are independent before MAP processing. The joint conditional probability density function of the b_i is therefore given by

$$\begin{aligned} f_{B|X=0}(b_1,b_2,b_3)&=f_{B_1|X_1=0}(b_1)\cdot f_{B_2|X_2=0}(b_2)\cdot f_{B_3|X_3=0}(b_3)\\ &=\frac{1}{\left(4\sqrt{\pi}\gamma\right)^3 b_1b_2b_3}\exp\left[\frac{-\left(\ln(b_1)+4\gamma^2\right)^2-\left(\ln(b_2)+4\gamma^2\right)^2-\left(\ln(b_3)+4\gamma^2\right)^2}{16\gamma^2}\right]\end{aligned} \tag{3.65}$$

Substituting for the b_i and evaluating (3.65) we find that

$$f_{R1X=0}(r_1',r_2',r_3')=\begin{cases}\dfrac{1}{\left(4\sqrt{\pi\gamma}\right)^3}\cdot\sqrt{\dfrac{\left(-1+r_1'+r_2'+r_3'\right)^3}{\left(1-r_1'+r_2'-r_3'\right)\left(1-r_1'-r_2'+r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}}\cdot \\ \exp\left[-\left(\ln\left(\sqrt{\dfrac{\left(1-r_1'+r_2'-r_3'\right)\left(1-r_1'-r_2'+r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right]\cdot \\ \exp\left[-\left(\ln\left(\sqrt{\dfrac{\left(1-r_1'-r_2'+r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1-r_1'+r_2'-r_3'\right)}}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right]\cdot \\ \exp\left[-\left(\ln\left(\sqrt{\dfrac{\left(1-r_1'+r_2'-r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1-r_1'+r_2'-r_3'\right)}}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right]\cdot \\ \left|\dfrac{4}{\sqrt{\left(-1+r_1'+r_2'+r_3'\right)^5\left(1-r_1'-r_2'+r_3'\right)\left(1-r_1'+r_2'-r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}}\right|, \\ r_1'+r_2'+r_3'>1,\ r_1'-r_2'-r_3'>-1,\ -r_1'-r_2'+r_3'>-1,\ \&\ -r_1'+r_2'+r_3'>1; \\ 0,\ \text{otherwise.}\end{cases} \tag{3.66}$$

We can further simplify the expression above by noting that within the region of interest the Jacobian determinant is non-negative and thus the square rule is non-negative and real and so can be dropped. This allows (3.69) to be further simplified as

$$f_{R1X=0}(r_1',r_2',r_3')=\begin{cases}\dfrac{1}{\left(4\sqrt{\pi\gamma}\right)^3}\cdot\dfrac{4}{\left(-1+r_1'+r_2'+r_3'\right)\cdot\left(1-r_1'-r_2'+r_3'\right)\cdot\left(1-r_1'+r_2'-r_3'\right)\cdot\left(1+r_1'-r_2'-r_3\right.} \\ \exp\left[-\left(\dfrac{1}{2}\ln\left(\dfrac{\left(1-r_1'+r_2'-r_3'\right)\left(1-r_1'-r_2'+r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right]\bullet \\ \exp\left[-\left(\dfrac{1}{2}\ln\left(\dfrac{\left(1-r_1'-r_2'+r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1-r_1'+r_2'-r_3'\right)}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right]\bullet \\ \exp\left[-\left(\dfrac{1}{2}\ln\left(\dfrac{\left(1-r_1'+r_2'-r_3'\right)\left(1+r_1'-r_2'-r_3'\right)}{\left(-1+r_1'+r_2'+r_3'\right)\left(1-r_1'-r_2'+r_3'\right)}\right)+4\gamma^2\right)^2\Big/\left(16\gamma^2\right)\right], \\ r_1'+r_2'+r_3'>1,\ r_1'-r_2'-r_3'>-1,\ -r_1'-r_2'+r_3'>-1,\ -r_1'+r_2'+r_3'>1; \\ 0,\ \text{otherwise.}\end{cases} \tag{3.67}$$

We can use the joint density function denoted in (3.67) to determine the probability of specific patterns of errors occurring. These probabilities may then be used to determine the probability of the various subfields forming.

Let us now determine an analytic expression for the probability of a bit error after one cycle of MAP processing, given the value of the reliability estimate for that interval and that the bit is in a particular error subfield. We assume that the code being used is a (3,2) even parity code. We can express this probability as

$$\begin{aligned} P[e_i|z_i;r_i'] &= \frac{P[e_i;z_i;r_i']}{P[z_i;r_i']} \\ &= \frac{P[e_i;z_i;r_i';a_{000}]+P[e_i;z_i;r_i';a_{011}]+P[e_i;z_i;r_i';a_{101}]+P[e_i;z_i;r_i';a_{110}]}{P[z_i;r_i';a_{000}]+P[z_i;r_i';a_{011}]+P[z_i;r_i';a_{101}]+P[z_i;r_i';a_{110}]} \\ &= \frac{\begin{pmatrix} P[e_i|z_i;r_i';a_{000}]\cdot P[z_i;r_i';a_{000}]+P[e_i|z_i;r_i';a_{011}]\cdot P[z_i;r_i';a_{011}]+ \\ P[e_i|z_i;r_i';a_{101}]\cdot P[z_i;r_i';a_{101}]+P[e_i|z_i;r_i';a_{110}]\cdot P[z_i;r_i';a_{110}] \end{pmatrix}}{P[z_i;r_i';a_{000}]+P[z_i;r_i';a_{011}]+P[z_i;r_i';a_{101}]+P[z_i;r_i';a_{110}]}. \end{aligned} \tag{3.68}$$

Where the term a_{000} indicates that $a_i=a_\alpha=a_\beta=0$, and a_i is the value of the bit transmitted in that interval. We observe that the probability of error is independent of the subfield if we are given the value of r_i' and the value of the transmitted bit, as is shown in the following table.

Table 3.4. Probability of error conditioned on zero-error subfield, value of reliability estimate and transmitted codeword.

Conditional Subfield Probabilities	$r_i'<1/2$	$r_i'>1/2$
$P[e_i\|z_i;r_i';a_{000}]$	1	0
$P[e_i\|z_i;r_i';a_{000}]$	1	0
$P[e_i\|z_i;r_i';a_{000}]$	0	1
$P[e_i\|z_i;r_i';a_{000}]$	0	1

Therefore (3.68) becomes

$$P[e_i|z_i;r_i'] = \begin{cases} \dfrac{P[z_i;r_i';a_{000}]+P[z_i;r_i';a_{011}]}{P[z_i;r_i';a_{000}]+P[z_i;r_i';a_{011}]+P[z_i;r_i';a_{101}]+P[z_i;r_i';a_{110}]}, & r_i' < 1/2; \\ \dfrac{P[z_i;r_i';a_{101}]+P[z_i;r_i';a_{110}]}{P[z_i;r_i';a_{000}]+P[z_i;r_i';a_{011}]+P[z_i;r_i';a_{101}]+P[z_i;r_i';a_{110}]}, & r_i' > 1/2. \end{cases} \quad (3.69)$$

We can express each term in terms of conditional probabilities, i.e.,

$$\begin{aligned} P[z_i;r_i';a_{000}] &= P[z_i|r_i';a_{000}]\cdot P[r_i'|a_{000}]\cdot P[a_{000}] \\ P[z_i;r_i';a_{011}] &= P[z_i|r_i';a_{011}]\cdot P[r_i'|a_{011}]\cdot P[a_{011}] \\ P[z_i;r_i';a_{101}] &= P[z_i|r_i';a_{101}]\cdot P[r_i'|a_{101}]\cdot P[a_{101}] \\ P[z_i;r_i';a_{110}] &= P[z_i|r_i';a_{110}]\cdot P[r_i'|a_{110}]\cdot P[a_{110}] \end{aligned} \quad (3.70)$$

If we assume equiprobable transmitted codewords, i.e., $P[a_{000}]=P[a_{011}]=P[a_{101}]=P[a_{110}]$, then (3.69) can be expressed as

$$P[e_i|z_i;r_i'] = \begin{cases} \dfrac{P[z_i|r_i';a_{000}]\cdot P[r_i'|a_{000}]+P[z_i|r_i';a_{011}]\cdot P[r_i'|a_{011}]}{\left(\begin{array}{l} P[z_i|r_i';a_{000}]\cdot P[r_i'|a_{000}]+P[z_i|r_i';a_{011}]\cdot P[r_i'|a_{011}]+ \\ P[z_i|r_i';a_{101}]\cdot P[r_i'|a_{101}]+P[z_i|r_i';a_{110}]\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i' < 1/2; \\ \dfrac{P[z_i|r_i';a_{101}]\cdot P[r_i'|a_{101}]+P[z_i|r_i';a_{110}]\cdot P[r_i'|a_{110}]}{\left(\begin{array}{l} P[z_i|r_i';a_{000}]\cdot P[r_i'|a_{000}]+P[z_i|r_i';a_{011}]\cdot P[r_i'|a_{011}]+ \\ P[z_i|r_i';a_{101}]\cdot P[r_i'|a_{101}]+P[z_i|r_i';a_{110}]\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i' > 1/2. \end{cases} \quad (3.71)$$

We now consider the $P[z_i|r_i';a_{i\alpha\beta}]$ terms. These can each be expressed similar to the expansion

$$P[z_i|r_i';a_{000}] = \sum_k P[z_i;\alpha_k|r_i';a_{000}] = \sum_k P[z_i|\alpha_k;r_i';a_{000}]\cdot P[\alpha_k|r_i';a_{000}]. \quad (3.72)$$

The first term in the summation of (3.72) can be easily determined when we note that the zero-error subfield can only occur for α_k=0 and α_k=2. The other conditions let us know whether or not there is an error at position i of the decoded symbol. That is, if $r_i' < 1/2$, we will decode this symbol as 1, but the third condition tells us that a zero was transmitted. Thus the value for the first term is given by the values in the following table.

Table 3.5. Probability of a bit being in the zero-error subfield, given k errors, the value of reliability estimate and the transmitted codeword.

	$P[z_i \mid \alpha_k, r_i', a_{000}]$	
k	$r_i'<1/2$	$r_i'>1/2$
0	0	1
1	0	0
2	1	1
3	0	0

Therefore (3.72) becomes

$$P[z_i|r_i';a_{000}]=\begin{cases}P[\alpha_2|r_i';a_{000}], & r_i'<1/2;\\ P[\alpha_0|r_i';a_{000}]+P[\alpha_2|r_i';a_{000}], & r_i'>1/2.\end{cases} \tag{3.73}$$

Similarly,

$$P[z_i|r_i';a_{011}]=\begin{cases}P[\alpha_2|r_i';a_{011}], & r_i'<1/2;\\ P[\alpha_0|r_i';a_{011}]+P[\alpha_2|r_i';a_{011}], & r_i'>1/2.\end{cases} \tag{3.74}$$

For the cases in which a_i=1, we get

$$P[z_i|r_i';a_{101}]=\begin{cases}P[\alpha_0|r_i';a_{101}]+P[\alpha_2|r_i';a_{101}], & r_i'<1/2;\\ P[\alpha_2|r_i';a_{101}], & r_i'>1/2;\end{cases} \tag{3.75}$$

and

$$P[z_i|r_i';a_{110}]=\begin{cases}P[\alpha_0|r_i';a_{110}]+P[\alpha_2|r_i';a_{110}], & r_i'<1/2;\\ P[\alpha_2|r_i';a_{110}], & r_i'>1/2.\end{cases} \tag{3.76}$$

Equation (3.73)–(3.76) can now be used in (3.71), thereby yielding

$$P[e_i|z_i;r_i'] = \begin{cases} \dfrac{P[\alpha_2|r_i';a_{000}]\cdot P[r_i'|a_{000}]+P[\alpha_2|r_i';a_{011}]\cdot P[r_i'|a_{011}]}{\left(\begin{array}{l} P[\alpha_2|r_i';a_{000}]\cdot P[r_i'|a_{000}]+P[\alpha_2|r_i';a_{011}]\cdot P[r_i'|a_{011}]+ \\ \left(P[\alpha_0|r_i';a_{101}]+P[\alpha_2|r_i';a_{101}]\right)\cdot P[r_i'|a_{101}]+ \\ \left(P[\alpha_0|r_i';a_{110}]+P[\alpha_2|r_i';a_{110}]\right)\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i'<1/2; \\ \dfrac{P[\alpha_2|r_i';a_{101}]\cdot P[r_i'|a_{101}]+P[\alpha_2|r_i';a_{110}]\cdot P[r_i'|a_{110}]}{\left(\begin{array}{l} \left(P[\alpha_0|r_i';a_{000}]+P[\alpha_2|r_i';a_{000}]\right)\cdot P[r_i'|a_{000}]+ \\ \left(P[\alpha_0|r_i';a_{011}]+P[\alpha_2|r_i';a_{011}]\right)\cdot P[r_i'|a_{011}]+ \\ P[\alpha_2|r_i';a_{101}]\cdot P[r_i'|a_{101}]+P[\alpha_2|r_i';a_{110}]\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i'>1/2. \end{cases} \tag{3.77}$$

By a similar process we can determine the equivalent expression for the No-Error Subfield, i.e.

$$P[e_i|u_i;r_i'] = \begin{cases} \dfrac{\left(\begin{array}{l} \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{000}]+ \\ \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{011}] \end{array}\right)}{\left(\begin{array}{l} \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{000}]+ \\ \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{011}]+ \\ \left(P[\alpha_1|r_i';a_{101}]+P[\alpha_3|r_i';a_{101}]\right)\cdot P[r_i'|a_{101}]+ \\ \left(P[\alpha_1|r_i';a_{110}]+P[\alpha_3|r_i';a_{110}]\right)\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i'<1/2; \\ \dfrac{\left(\begin{array}{l} \left(P[\alpha_1|r_i';a_{101}]+P[\alpha_3|r_i';a_{101}]\right)\cdot P[r_i'|a_{101}]+ \\ \left(P[\alpha_1|r_i';a_{110}]+P[\alpha_3|r_i';a_{110}]\right)\cdot P[r_i'|a_{110}] \end{array}\right)}{\left(\begin{array}{l} \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{000}]+ \\ \left(P[\alpha_1|r_i';a_{000}]+P[\alpha_3|r_i';a_{000}]\right)\cdot P[r_i'|a_{011}]+ \\ \left(P[\alpha_1|r_i';a_{101}]+P[\alpha_3|r_i';a_{101}]\right)\cdot P[r_i'|a_{101}]+ \\ \left(P[\alpha_1|r_i';a_{110}]+P[\alpha_3|r_i';a_{110}]\right)\cdot P[r_i'|a_{110}] \end{array}\right)}, & r_i'>1/2. \end{cases} \tag{3.78}$$

Let us consider the terms in (3.77), and as an example $P[\alpha_2|r_i';a_{000}]$ specifically. We first consider the condition that $r_i'<1/2$. Under this condition $P[\alpha_2|r_i';a_{000}]$ is the probability that there is an error in the decoded symbol in addition to the error in the ith bit. This is given by

$$P[\alpha_2|r_i';a_{000}] = 2\int_{1/2}^{1}\int_{0}^{1/2} f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta'$$

$$= 2\int_{1/2}^{1}\int_{0}^{1/2} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta';\quad r_i'<1/2. \tag{3.79}$$

If we now consider the same probability when $r_i'>1/2$, we find that this represents the probability that the two bits besides the ith bit are in error. This is given by

$$P[\alpha_2|r_i';a_{000}] = \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta',\quad r_i'>1/2. \tag{3.80}$$

The other probabilities in (3.77) and (3.78) are all determined similarly and are presented in Table 3.6 below:

Table 3.6. (Part A) Probability of k errors occurring, given the value of reliability estimate and the transmitted codeword.

Term	$r_i'<1/2$	$r_i'>1/2$
$P[\alpha_0\|r_i';a_{000}]$	0	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta'$
$P[\alpha_1\|r_i';a_{000}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta'$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta$
$P[\alpha_2\|r_i';a_{000}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta$
$P[\alpha_3\|r_i';a_{000}]$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{000}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{000}}(r_i')}\,dr_\alpha'\,dr_\beta$	0
$P[\alpha_0\|r_i';a_{011}]$	0	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta$
$P[\alpha_1\|r_i';a_{011}]$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta'$
$P[\alpha_2\|r_i';a_{011}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta$
$P[\alpha_3\|r_i';a_{011}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i'|\mathcal{A}_{011}}(r_i';r_\alpha';r_\beta')}{f_{R_i'|\mathcal{A}_{011}}(r_i')}\,dr_\alpha'\,dr_\beta'$	0

Table 3.6. (Part B) Probability of k errors occurring, given the value of reliability estimate and the transmitted codeword.

Term	$r_i'<1/2$	$r_i'>1/2$
$P[\alpha_1 \mid r_i'; a_{101}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{101}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{101}}(r_i')} dr_\alpha' dr_\beta'$	$\int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{101}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{101}}(r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_2 \mid r_i'; a_{101}]$	$\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{101}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{101}}(r_i')} dr_\alpha' dr_\beta'$	0
$P[\alpha_3 \mid r_i'; a_{101}]$	0	$\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{101}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{101}}(r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_0 \mid r_i'; a_{110}]$	$\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta$	0
$P[\alpha_1 \mid r_i'; a_{110}]$	$2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta'$	$\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_2 \mid r_i'; a_{110}]$	$\int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta$
$P[\alpha_3 \mid r_i'; a_{110}]$	0	$\int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i \mid A_{110}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i \mid A_{110}}(r_i')} dr_\alpha' dr_\beta$

If we observe that those terms in (3.77) and (3.78) of the form $P[r_i' \mid a_{xxx}]$ are in fact the probability density functions of r_i', i.e. $f_{R_i \mid A_{xxx}}(r_i')$, we see that (3.77) becomes

$$P[e_i|z_i;r_i'] = \begin{cases} \dfrac{2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{100}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{011}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta'}{\left(\begin{array}{l} 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{000}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{011}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \\ \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{001}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{010}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \\ \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{100}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{110}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \end{array}\right)}, & r_i' < 1/2; \\ \dfrac{2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{110}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{111}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta'}{\left(\begin{array}{l} 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{100}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{110}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \\ \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{000}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{010}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \\ \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{001}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i';R_\alpha';R_\beta'}\mathcal{A}_{011}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \end{array}\right)}, & r_i' > 1/2 \end{cases} \tag{3.81}$$

and similarly (3.78) becomes

$$P[e_i|u_i;r_i'] = \begin{cases} \dfrac{\left(\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \right.}{\left. \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{011}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{011}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \right)}, & r_i'<1/2; \\ \dfrac{\left(2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{101}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{110}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \right.}{\left. \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{011}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{011}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \right)} \\ \dfrac{\left(\int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{101}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{101}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \right.}{\left. \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{110}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{110}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \right)}, & r_i'>1/2 \\ \dfrac{\left(2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{100}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + 2\int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{011}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \right.}{\left. \int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{101}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{101}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{110}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{110}}(r_i';r_\alpha';r_\beta')dr_\alpha'dr_\beta' \right)} \end{cases} \tag{3.82}$$

We can incorporate the symmetry of the channel to express all the integrals of (3.81) and (3.82) in terms of $f_{R_i|A_{000}}(r_i')$. Consider, for example, $\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_iR_\alpha R_\beta|A_{011}}(r_i')\mathrm{d}R_\alpha \mathrm{d}R_\beta$. When $r_i'<1/2$, this means we will decode bit i as a 1, giving an error in position i. The region of integration over R_α and R_β indicate that we are finding the probability that bit α is in error when bit β is not in error. This is the same as the product of the integration

$\int_0^{\frac{1}{2}}\int_{\frac{1}{2}}^{1} f_{R_i R_\alpha R_\beta | A_{011}}(r_i')\mathrm{d}R_\alpha \mathrm{d}R_\beta$. Similarly, we can express all the entries in Table 3.6 in terms of $f_{R_i | A_{000}}(r_i')$, as shown in Table 3.7 below.

Table 3.7. (Part A) Probability of k errors occurring, given the value of reliability estimate and the transmitted codeword, in terms of the joint probability density function for the case when the all-zero codeword is transmitted.

Term	$r_i'<1/2$	$r_i'>1/2$
$P[\alpha_0 \mid r_i'; a_{000}]$	0	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta'$
$P[\alpha_1 \mid r_i'; a_{000}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta'$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$
$P[\alpha_2 \mid r_i'; a_{000}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$
$P[\alpha_3 \mid r_i'; a_{000}]$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	0
$P[\alpha_0 \mid r_i'; a_{011}]$	0	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta'$
$P[\alpha_1 \mid r_i'; a_{011}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta'$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$
$P[\alpha_2 \mid r_i'; a_{011}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$
$P[\alpha_3 \mid r_i'; a_{011}]$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	0
$P[\alpha_0 \mid r_i'; a_{101}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	0
$P[\alpha_1 \mid r_i'; a_{101}]$	$2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} \mathrm{d}r_\alpha' \mathrm{d}r_\beta$

Table 3.7. (Part B) Probability of k errors occurring, given the value of reliability estimate and the transmitted codeword, in terms of the joint probability density function for the case when the all-zero codeword is transmitted.

Term	$r_i'<1/2$	$r_i'>1/2$
$P[\alpha_2 \mid r_i'; a_{101}]$	$\int_0^{\frac{1}{2}}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$	$2\int_{\frac{1}{2}}^{1}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_3 \mid r_i'; a_{101}]$	0	$\int_0^{\frac{1}{2}}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_0 \mid r_i'; a_{110}]$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$	0
$P[\alpha_1 \mid r_i'; a_{110}]$	$2\int_{\frac{1}{2}}^{1}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_2 \mid r_i'; a_{110}]$	$\int_0^{\frac{1}{2}}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$	$2\int_{\frac{1}{2}}^{1}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$
$P[\alpha_3 \mid r_i'; a_{110}]$	0	$\int_0^{\frac{1}{2}}\int_0^{\frac{1}{2}} \frac{f_{R_\alpha' R_\beta' R_i' \mid A_{000}}(1-r_i'; r_\alpha'; r_\beta')}{f_{R_i' \mid A_{000}}(1-r_i')} dr_\alpha' dr_\beta'$

We can use these new integrals in (3.81) and (3.82) to get simplified expressions for the subfield error probabilities, i.e.

$$P[e_i|z_i;r_i'] = \begin{cases} \dfrac{2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta'}{\left(2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right.} & r_i'<1/2; \\ \left. \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right) & \\ \dfrac{2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta'}{\left(2\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right.} & r_i'>1/2 \\ \left. \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right) & \end{cases} \tag{3.83}$$

and

$$P[e_i|u_i;r_i'] = \begin{cases} \dfrac{\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta'}{\left(2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \right.} & r_i'<1/2; \\ \left. \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right) & \\ \dfrac{\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta'}{\left(2\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{000}}(r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i:R_\alpha:R_\beta|A_{000}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' + \right.} & r_i'>1/2 \\ \left. \int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i:R_\alpha:R_\beta|A_{001}}(1-r_i';r_\alpha';r_\beta')\,dr_\alpha'\,dr_\beta' \right) & \end{cases} \tag{3.84}$$

We observe that there are three integrals which need to be evaluated for the required values of r_i' and $1-r_i'$. These integrals were found to be given by the following relations

$$\int_0^{\frac{1}{2}}\int_0^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* = \begin{cases} \int_{\frac{1}{2}-r_i^*}^{\frac{1}{2}}\int_{\frac{1}{2}-r_i^*}^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^*, & r_i^*<1/2; \\ \int_0^{\frac{1}{2}-r_i^*}\int_{\frac{1}{2}-r_i^*-r_\alpha^*}^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* + \int_{\frac{1}{2}-r_i^*}^{\frac{1}{2}}\int_0^{\frac{1}{2}-r_i^*-r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* & \\ -\int_{r_i^*-\frac{1}{2}}^{\frac{1}{2}}\int_{\frac{1}{2}}^{\frac{1}{2}-r_i^*-r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^*, & r_i^*>1/2; \end{cases} \tag{3.85}$$

$$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\alpha^*dr_\beta^* = \begin{cases} \int_{\frac{1}{2}}^{\frac{1}{2}+r_i^*}\int_{\frac{1}{2}}^{\frac{1}{2}+r_i^*-r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\alpha^*dr_\beta^*, & r_i^*<1/2; \\ \int_{\frac{1}{2}}^{r_i^*}\int_{\frac{1}{2}-r_i^*+r_\alpha^*}^{\frac{1}{2}+r_i^*-r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* + \int_{r_i^*}^{1}\int_{\frac{1}{2}-r_i^*+r_\alpha^*}^{\frac{1}{2}+r_i^*-r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* & \\ -\int_{\frac{1}{2}}^{\frac{3}{2}-r_i^*}\int_{\frac{1}{2}-r_i^*+r_\alpha^*}^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^*, & r_i^*>1/2; \end{cases} \tag{3.86}$$

and

$$\int_{\frac{1}{2}}^{1}\int_0^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\alpha^*dr_\beta^* = \begin{cases} \int_0^{r_i^*}\int_{\frac{1}{2}-r_i^*-r_\alpha^*}^{\frac{1}{2}+r_\alpha^*-r_\beta^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* + \int_{r_\alpha^*}^{\frac{1}{2}}\int_{\frac{1}{2}-r_\alpha^*-r_\beta^*}^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^* - & \\ \int_{\frac{1}{2}-r_i^*}^{\frac{1}{2}}\int_{\frac{1}{2}-r_i^*-r_\alpha^*}^{\frac{1}{2}} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^*, & r_i^*<1/2; \\ \int_{r_i^*-\frac{1}{2}}^{\frac{1}{2}}\int_{\frac{1}{2}}^{\frac{1}{2}-r_i^*+r_\alpha^*} f_{R_i^*R_\alpha^*R_\beta^*\Lambda_{\infty}}\left(r_i^*;r_\alpha^*;r_\beta^*\right)dr_\beta^*dr_\alpha^*, & r_i^*>1/2 \end{cases} \tag{3.87}$$

The integrals were evaluated numerically with the MATHEMATICA™ software. The results as shown in Table 3.8 below.

Table 3.8. Results of integrating the joint probability density function for the case when the all-zero codeword is transmitted, over various regions of the reliability estimates. Calculated at a Channel E_b/N_0 of 0dB.

r_i'	$\int_{\frac{1}{2}}^{1}\int_{\frac{1}{2}}^{1} f_{R_i' R_\alpha' R_\beta' \mathcal{A}_{000}}(r_i'; r_\alpha'; r_\beta')\, dr_\alpha' dr_\beta'$	$\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{1}{2}} f_{R_i' R_\alpha' R_\beta' \mathcal{A}_{000}}(r_i'; r_\alpha'; r_\beta')\, dr_\alpha' dr_\beta'$	$\int_{\frac{1}{2}}^{1}\int_{0}^{\frac{1}{2}} f_{R_i' R_\alpha' R_\beta' \mathcal{A}_{000}}(r_i'; r_\alpha'; r_\beta')\, dr_\alpha' dr_\beta$
0.05	2.85×10^{-4}	2.29×10^{-5}	3.97×10^{-2}
0.10	6.69×10^{-4}	4.90×10^{-4}	3.53×10^{-2}
0.15	1.16×10^{-3}	7.95×10^{-5}	3.35×10^{-2}
0.20	1.80×10^{-3}	1.17×10^{-4}	3.29×10^{-2}
0.25	2.66×10^{-4}	1.65×10^{-4}	3.29×10^{-2}
0.30	3.87×10^{-3}	2.27×10^{-4}	3.34×10^{-2}
0.35	5.64×10^{-3}	3.12×10^{-4}	3.42×10^{-2}
0.40	8.46×10^{-3}	4.34×10^{-4}	3.52×10^{-2}
0.45	2.85×10^{-4}	6.23×10^{-4}	3.55×10^{-2}
0.50	4.58×10^{-2}	1.02×10^{-3}	2.34×10^{-2}
0.55	8.46×10^{-2}	8.79×10^{-3}	2.15×10^{-3}
0.60	1.02×10^{-1}	6.67×10^{-3}	3.30×10^{-3}
0.65	1.22×10^{-2}	5.53×10^{-3}	4.61×10^{-3}
0.70	1.50×10^{-1}	4.78×10^{-3}	6.24×10^{-3}
0.75	1.89×10^{-1}	4.24×10^{-3}	8.43×10^{-3}
0.80	2.51×10^{-1}	3.84×10^{-3}	1.16×10^{-2}
0.85	3.63×10^{-1}	3.51×10^{-3}	1.70×10^{-2}
0.90	6.08×10^{-1}	3.23×10^{-3}	2.76×10^{-2}
0.95	1.45×10^{-0}	2.93×10^{-3}	5.90×10^{-2}

These results were used in (3.83) and (3.84) to generate the theoretical subfield error rates conditioned on reliability estimates. These error rates are listed in Table 3.9 below.

Table 3.9. Theoretical subfield error rates conditioned on reliability estimates. Calculated at a Channel E_b/N_0 of 0dB.

r_i'	$P[e_i\|z_i,r_i']$	$P[e_i\|u_i,r_i']$
0.05	0.05	0.05
0.10	0.10	0.10
0.15	0.15	0.15
0.20	0.20	0.20
0.25	0.25	0.25
0.30	0.30	0.30
0.35	0.35	0.35
0.40	0.40	0.40
0.45	0.45	0.45
0.50	0.50	0.50
0.55	0.45	0.45
0.60	0.40	0.40
0.65	0.35	0.35
0.70	0.30	0.30
0.75	0.25	0.25
0.80	0.20	0.20
0.85	0.15	0.15
0.90	0.10	0.10
0.95	0.05	0.05

These results are shown in the figures below, and we see that they match the simulation results.

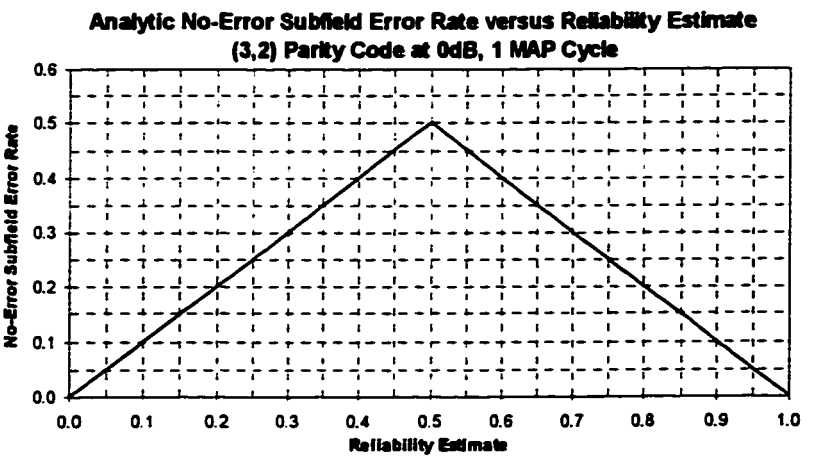


Figure 3.17. Plot of Analytic No-Error Subfield Probability of Error for a (3,2) Parity Code after 1 cycle of MAP processing at 0dB.

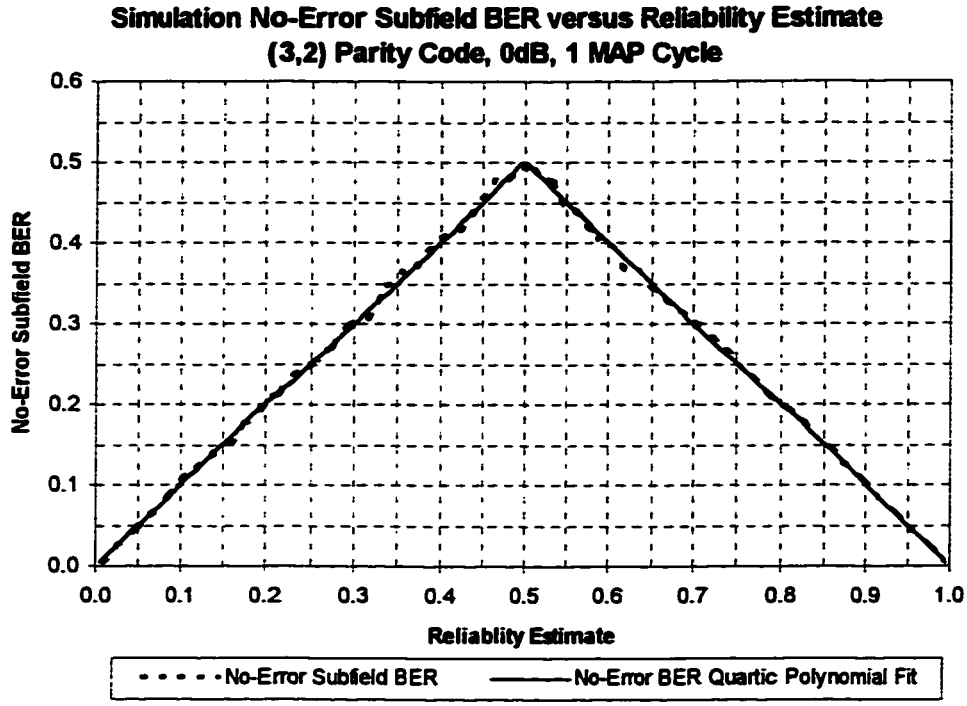


Figure 3.18. Plot of empirical No-Error Subfield BER given the reliability estimate and fitted polynomial for a (3,2) even parity code after 1 cycle of MAP processing at a channel E_b/N_0 of 0 dB.

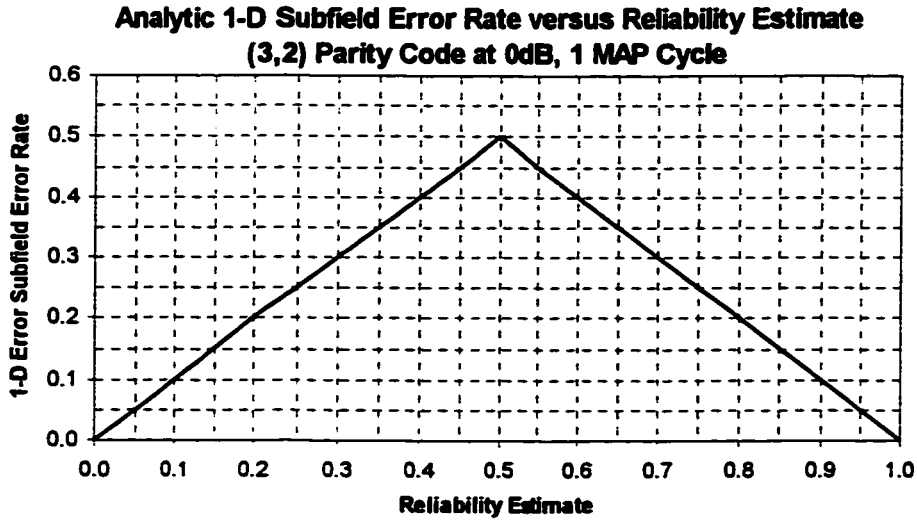


Figure 3.19. Plot of Analytic 1-D Error Subfield Probability of Error for a (3,2) Parity Code after 1 cycle of MAP processing at 0dB.

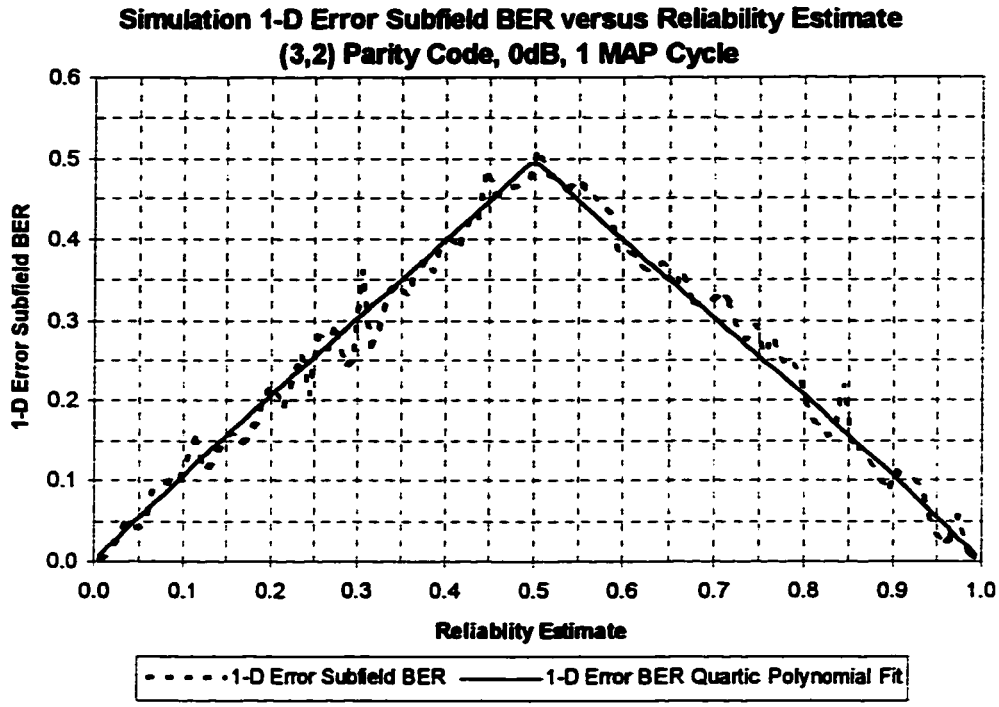


Figure 3.20. Plot of empirical 1-D Error Subfield BER given the reliability estimate and fitted polynomial for a (3,2) even parity code after 1 cycle of MAP processing at a channel E_b/N_0 of 0 dB.

3.7 Summary

The analysis for the relatively simple (3,2) parity code after only 1 MAP cycle proved to be a very involved and tedious process. From this we can extrapolate to the realisation that the corresponding procedure for codes with greater dimensions or for more MAP cycles would be of great difficulty indeed. For instance, if we wish to analyse the 3×3 product code we can find the joint probability density function of the reliability estimates relatively easily since we can use the same procedure as that used in the (3,2) case. However, we will then be faced with the task of having to integrate over a 9^{th}-dimensional space in order to find the probabilities of the 27 possible error patterns. In the case of the 9×9 product codes which were used in this thesis for evaluating improvements to the MAP algorithm, the problem of determining the joint probability density function will be

great and even more so integrating over the 81-dimensional region to determine the probabilities of the 27 possible error patterns.

For these reasons, instead of analytically determining the required Subfield Error Rate versus Reliability Estimate relationships, for use in the "remapping" modification (presented in Chapter 4), we empirically determine them via Monte Carlo simulation.

As mentioned in Section 3.1, one of the objectives in doing the theoretical analysis was to find out whether or not there was a discontinuity in the remapping functions at the point where the APP=0.5. Based on our analytic results as shown in Figures 3.10 and 3.14, we can see that in these cases there will be a discontinuity at the point in question, given that the characteristic does not pass through the (0.5, 0.5) point. We therefore infer from these cases that when the empirically determined characteristics do not seem to pass through the (0.5, 0.5) point, there is in fact a discontinuity present. Furthermore, there is no need to constrain the polynomials, fit to the empirical characteristics, to pass through the (0.5, 0.5) point.

Chapter 4

Remapping Reliability Estimates

4.1. Overview

It was found that there is a significant difference between the observed BER and the estimate derived from the "reliability estimate" produced by iterating the MAP decoding algorithm. Specifically, we expect that the BER versus Reliability Estimate will be a triangle function. However, on examining the actual relationship found on simulating the system, the triangle function was not found. The deviation from the expected was also observed to increase as the number of MAP iterations performed was increased. We felt that this discrepancy might allow the iterated algorithm to be improved by incorporating remapping of the produced "reliability estimates" to bring them into line with the observed BER (i.e., to make them truly correspond to reliability estimates). In this chapter, we shall demonstrate that this approach can indeed improve the performance of the iterated MAP algorithm, adopting empirical data from simulations to determine the necessary remapping functions.

4.2. Subfield BER Distribution

As discussed earlier, the counted BER (B_r) can be expressed in terms of the conditional probability that a bit is zero given the observed value ($P[x_i=0|Y;C]$). This relationship is given by

$$B_r = \begin{cases} P[x_i = 0|Y;C], & 0.0 \le P[x_i = 0|Y;C] < 0.5; \\ 1 - P[x_i = 0|Y;C], & 0.5 \le P[x_i = 0|Y;C] \le 1.0; \end{cases} \tag{4.1}$$

and is shown in Figure 4.1 below. We know, however, that after more than one iteration of the MAP algorithm, the algorithm does not truly produce $P[x=0|Y;C]$, but some skewed reliability estimates. For lack of a better concise term, we will nonetheless refer to the iterated MAP algorithm's output as reliability estimates.

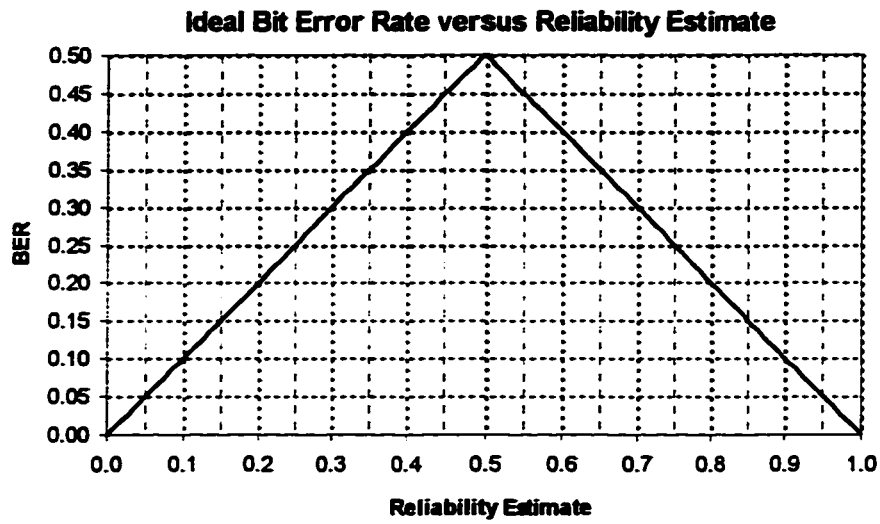


Figure 4.1. Plot showing the ideal relationship between BER and reliability estimates $P[x=0|Y;C]$.

Simulations of a channel were conducted using the iterated MAP algorithm and, when the BER for bits with different reliability data was measured, the results from the first iteration were consistent with the prediction from (4.1), as expected. Figure 4.2 shows the result of a simulation of the use of a 9×9 simple parity code at an E_b/N_0=3dB. The results are based on a run observing 10,000 errors through the system and dividing the values of possible reliability values into 100 bins (0 to 0.01, 0.01 to 0.02, etc.) and counting errors in each bin. There are relatively few bits with reliability values near 0.5 which is responsible for the large fluctuations about the expected result in this case. If we apply a second iteration of the MAP algorithm and plot the bit error rate versus the new "reliability" values, we obtain the result shown in Figure 4.3. We see what appears to be a slight deviation from the curve of Figure 4.1. The deviation becomes even more pronounced if we apply a third iteration of the MAP algorithm as Figure 4.4 demonstrates.

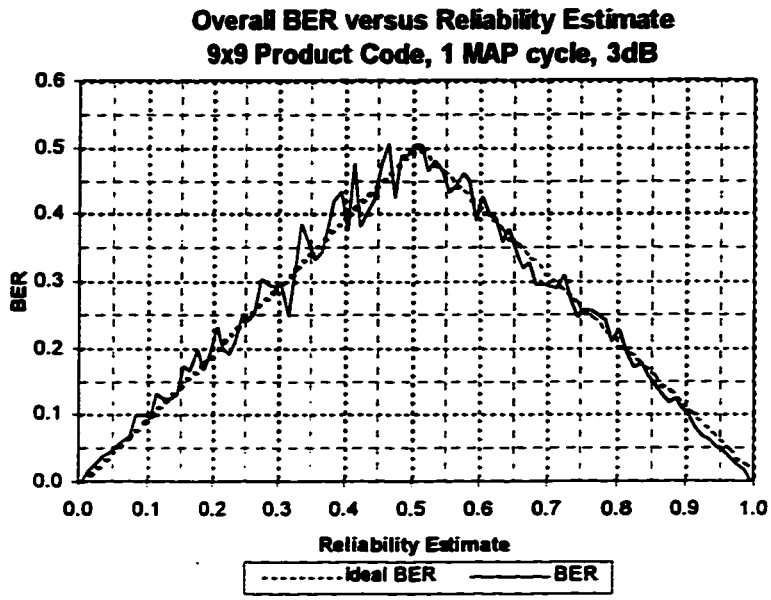


Figure 4.2. Plot of overall counted BER versus reliability estimate after 1 cycle of MAP processing. See the text for information on the details of the simulation.

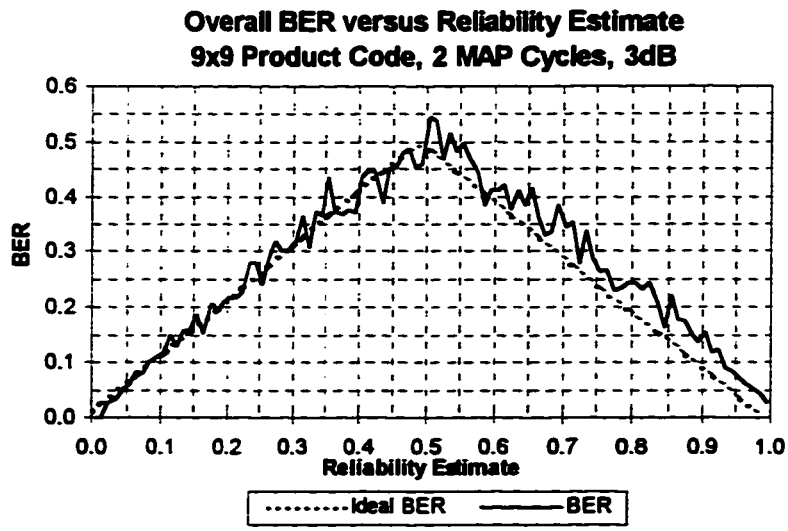


Figure 4.3. Plot of overall counted BER versus reliability estimate after 2 cycles of MAP processing

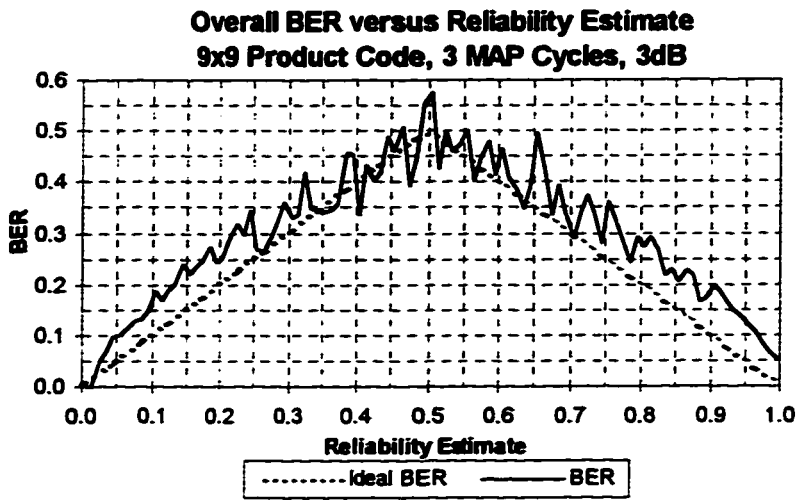


Figure 4.4. Plot of overall counted BER versus reliability estimate after 3 cycles of MAP processing

Thus we see that as the number of MAP cycles increase, the BER versus reliability estimate relationship increasingly diverges from the simple relationship of (4.1). This divergence is a measure of the extent to which the reliability estimates misrepresents the true probability of error in the bits passed to the outer decoder in the concatenated coding scheme.

The BER distribution in the error subfields was also analysed. Deviation from the relationship in (4.1) was also found to occur for the subfield BER distributions. Interestingly, significant deviation is evident even after only one iteration (as shown in the Figures. 4.5 to 4.7 below), and this deviation was also observed to increase with further application of the MAP algorithm. This of course is to expected since the reliability values do not reflect probabilities conditional on the subfield in which a bit is found.

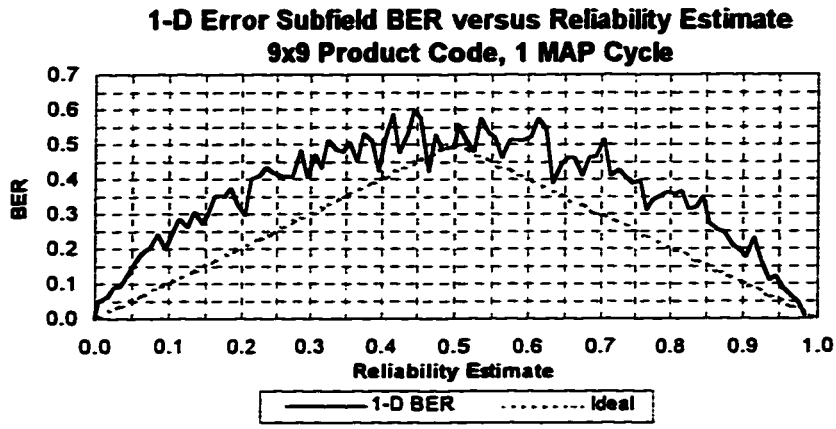


Figure 4.5. Plot of counted BER in the 1-D error subfield versus reliability estimate after 1 cycle of MAP processing.

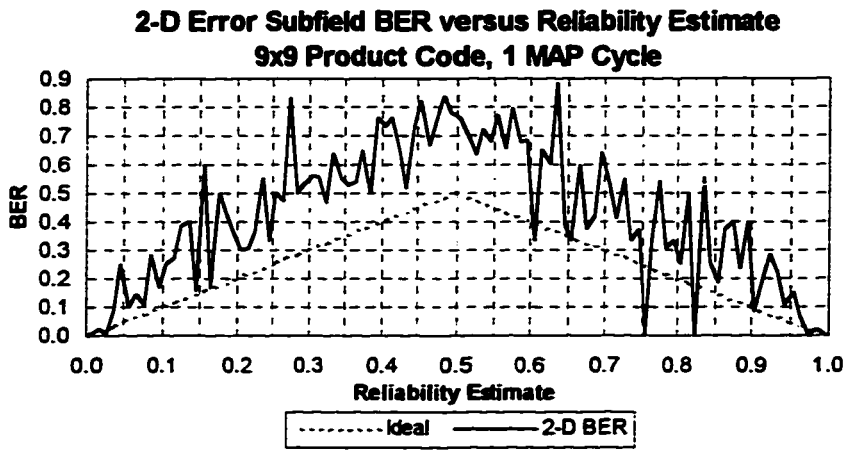


Figure 4.6. Plot of counted BER in the 2-D error subfield versus reliability estimate after 1 cycle of MAP processing.

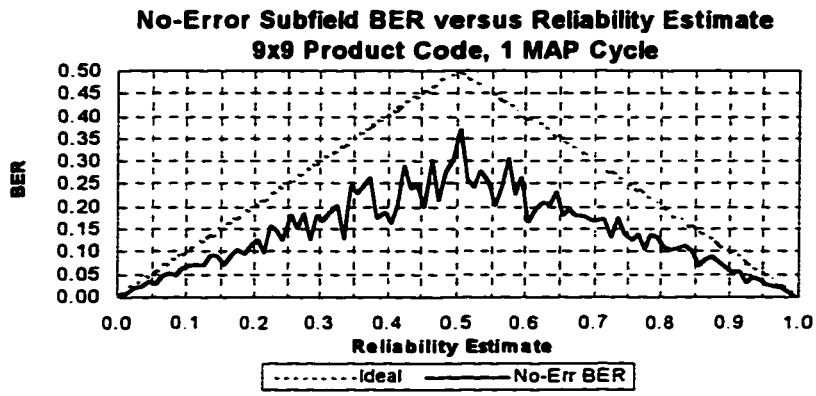


Figure 4.7. Plot of counted BER in the No-Error subfield versus reliability estimate after 1 cycle of MAP processing.

These plots imply as well that there may be information which can be exploited to make the reliability values output more accurate. In the following section this information is used to remap the inaccurate reliability estimates for each subfield to more accurate values.

4.3. Remapping

4.3.1. Overview

In order to apply remapping in any approach, we must find the relationship between the BER and the "reliability" data. As we saw in Chapter 3, determining this analytically can be quite difficult. An alternative to this is to determine the relationship empirically by conducting simulations of the communication system, observing the frequency of errors for small ranges of "reliability values", and fitting a model remapping function to the data. This is the approach we chose to take, using a piecewise polynomial function to relate reliability to BER. As a result, the steps required to implement remapping are

(i) collect error statistics for the subfields,

(ii) fit a polynomial to the statistics collected for each subfield,

(iii) generate remapping functions from the fitted polynomials, and

(iv) modify the reliability estimates in each subfield using the appropriate remapping function.

It should be pointed out here that it is possible to collect error statistics and subsequently apply the resulting remapping functions at a variety of points in the decoding process. For instance, remapping can be carried out after all MAP processing has been completed, or after each iteration of MAP processing or even between stages in a single MAP algorithm iteration. The effect of the choice of points at which remapping is done on the performance of the iterated MAP decoder is explored in Section 4.4.5.

Having determined this relationship, it is then possible to remap each reliability estimate $P_k[x_t = 0|Y;C]$ to a new value, denoted by $P'_k[x_t = 0|Y;C]$ with the same BER as that which was measured. The new estimate $P'_k[x_t = 0|Y;C]$ can be expressed as

$$P'_k[x_t = 0|y_t] = \begin{cases} B(P_k[x_t = 0|Y;C]); & 0.0 \le P_k[x_t = 0|Y;C] < 0.5 \\ 1 - B(P_k[x_t = 0|Y;C]); & 0.5 \le P_k[x_t = 0|Y;C] \le 1.0 \end{cases} \quad (4.2)$$

where $B(\cdot)$ denotes the polynomial derived from performing a best fit to the error statistics collected for each value of $P_k[x_t=0|Y;C]$.

In attempting to fit a curve to the empirical data, several constraints on the remapping function seem to be appropriate. First, when the reliability data is either exactly 0 or 1, the BER is expected to be 0, and the remapping function should reflect this. Secondly, symmetry would dictate that the curve relating BER to "reliability" would be symmetric about the value 0.5. To address these constraints it was felt that a low order polynomial could be used to represent the relationship for reliability values between 0 and 0.5, and then its mirror image for values between 0.5 and 1. This curve should pass through the point (0, 0), and so, it would be a polynomial whose constant term was zero. Various degree polynomials and methods to judge optimality of fit were tried and it was found that a fourth degree polynomial provided a good fit to the data. The fourth degree polynomial had the best compromise between following the data too closely or not. In Figure 4.8 below, we compare a 4th degree and 6th degree polynomial. We can see that the 6th degree polynomial has more inflexion points than the 4th degree polynomial. This is undesirable as we expect the characteristic to have only one point of inflexion at the midpoint of the reliability estimate axis.

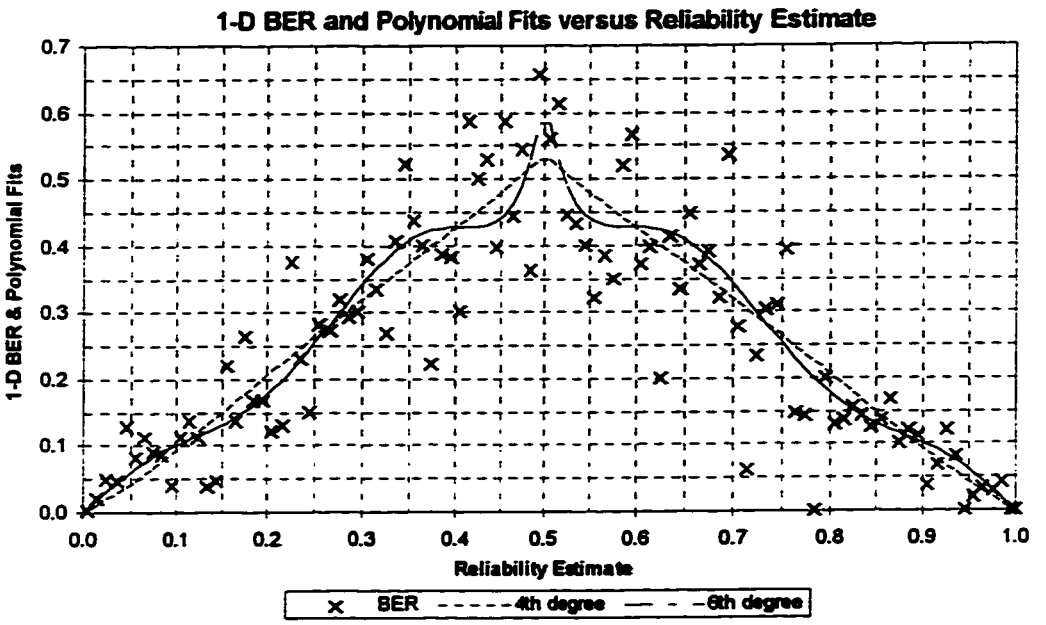


Figure 4.8. Plot of BER and 4th and 6th degree polynomial fit. For the 1-D Error Subfield of a 9×9 product code after 3 MAP Cycles at 0dB.

We now illustrate the remapping process with a simple example; specifically, we will remap the estimates after two iterations of MAP processing. We know that the estimates will seemingly indicate a lower probability of bit error than will actually be observed. For example, after two iterations reliability estimates with values of 0.3 might actually correspond to a true BER of 0.4. From (4.1), the reliability estimate of 0.3 should correspond to a BER of 0.3, indicating that these reliability estimates are incorrect and need to be changed. From (4.1) again it is seen that ideally a BER of 0.4 corresponds to a reliability estimate with value 0.4, therefore the value of the reliability estimate is changed (remapped) from 0.3 to 0.4. In this example, remapping would be applied to all the reliability estimates after the second complete MAP cycle. The various steps involved in remapping are described in further detail in the following sections.

4.3.2. Collect error statistics

Let $P_k[x_t=0|Y;C]$ be the reliability estimate output after the kth application of the MAP algorithm. Hard decisions on each bit are then made from these estimates and the BER observed in each range of values. The BER is found by counting the number of estimates

and the number of errors that occur at each range of values between 1 and 0. These reliability value versus BER histogram plots are the raw data from which we empirically determine the remapping polynomial.

In regard to choice of the number of bins in the histogram, a compromise must be reached between the accuracy of the histogram and the speed of carrying out the simulation. The number of bins must be high enough for the polynomial subsequently generated from it to be a good approximation of the actual BER/Reliability Estimate relationship. On the other hand, should the number of bins be too high the simulations—run in MATLAB™ on a 33MHz 486 PC—will take a prohibitively long time to complete. In addition, we can expect that above some number of bins, there is no practical benefit in having such small bins, and should the bins be too small the number of observations in each bin will be too small to provide statistically significant data. We eventually settled on using 100 bins, after considering 10 and 50 bins as being too crude, and finding simulations generating enough data points for 1000 bins to take too much time.

4.3.3. Polynomial Fitting

4.3.3.1. Polynomial approximation procedure

In order to map the observed probabilities to the improved values, the error statistics on the distribution of errors at the observed reliability estimates are collected. A histogram of the counted BER versus reliability estimate is then plotted. This histogram provides a comparison of the empirically determined probability of bit error (estimated by the counted BER) versus reliability estimate. We then fit a polynomial to this histogram thus determined by minimizing the total weighted mean square error at the data points subject to the constraints that the polynomial pass through the origin.

To be more specific, let us assume that our observations are divided into N bins on (0, 1] so that the ith bin covers the range $\left(\frac{i-1}{N}, \frac{i}{N}\right]$ (for I = 1, 2,..., N). Let e_i be the number

of errors at histogram bin number i, and let n_i be the number of events observed in bin number i. Then the observed BER at bin i, b_i is given by

$$b_i = \frac{e_i}{n_i}. \tag{4.3}$$

Let $b_e(x)$ denote a polynomial function of degree $\leq k$

$$b_e(x) = a_k x^k + a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 = \sum_{j=0}^{k} a_j \cdot x^j. \tag{4.4}$$

We would like to choose the coefficients of $b_e(x)$ so that it is a best fit to the observed error rates in each bin b_i, at the values of x in the midpoint of the range for each bin. In the ith bin, the midpoint is

$$x_i = \frac{i - \frac{1}{2}}{N}.$$

The criterion of best fit that we will use to select the coefficients of the polynomial is to minimize a weighted sum of the square error, given by

$$E = \sum_{i=1}^{N} w_i \cdot \left[b_i - b_e(x_i) \right]^2,$$

where the w_i are "weighting coefficients" used to give greater or lesser weight to the deviation in some bins. As mentioned previously, we must minimise E subject to the constraint that $b_e(0)=0$. This constraint is easily seen to be the same as dictating that $a_0=0$. As a result we simply seek the coefficients $a_1, a_2, \ldots, a_k$ so as to minimize

$$E(\bar{a}) = \sum_{x=1}^{N} w_i \cdot \left[b_i - \left(\sum_{j=0}^{k} a_j \cdot \left(x_j \right) \right) \right]^2,$$

where $\bar{a}$ denotes the set of coefficients $a_1, a_2, \ldots, a_k$.

To select these remaining coefficients let

$$E(\bar{a}) = \sum_{i=1}^{N} w_i \cdot \left[b_i - b_e(\bar{a}, x_i) \right]^2. \tag{4.5}$$

To minimise $E(\bar{a})$, it is necessary and sufficient that $\frac{\partial}{\partial a_l} E(\bar{a}) = 0$; for l=1,2,...,k.

Differentiating with respect to a_l, through (4.5) we have

$$\frac{\partial}{\partial a_l} E(\bar{a}) = \sum_{i=1}^{N} 2 \cdot w_i \cdot \left[b_i - b_e(\bar{a}, x_i) \right] \frac{\partial}{\partial a_l} b_e(\bar{a}, x_i) = 0. \tag{4.6}$$

Thus the condition for optimality becomes

$$2 \cdot \sum_{i=1}^{N} w_i \cdot \left[b_i - b_e(\bar{a}, x_i) \right] \cdot x_i^l = 0. \tag{4.7}$$

Substituting for $b_e(\bar{a}, x_i)$, this can be restated as

$$\sum_{i=1}^{N} x_i^l \cdot w_i \cdot \left[b_i - \left(\sum_{j=1}^{k} a_j \cdot x_i^j \right) \right] = 0, \tag{4.8}$$

which leads to

$$\sum_{i=1}^{N} x_i^l \cdot w_i \cdot \left(\sum_{j=1}^{k} a_j \cdot x_i^l \right) = \sum_{i=1}^{N} x_i^l \cdot b_i . \tag{4.9}$$

Rearranging the order of summation

$$\sum_{j=1}^{k} a_j \sum_{i=1}^{N} w_i \cdot x_i^{j+l} = \sum_{i=1}^{N} x_i^l \cdot w_i \cdot b_i . \tag{4.10}$$

Now, let $\alpha_{l,j} = \sum_{i=1}^{N} w_i \cdot x_i^{j+l}$, and let $\beta_l = \sum_{i=1}^{N} x_i^l \cdot w_i \cdot b_i$, such that

$$\sum_{j=1}^{k} a_j \cdot \alpha_{l,j} = \beta_l \text{ for } l=1,2,\ldots,k. \tag{4.11}$$

Thus we see that the problem of finding the coefficients to minimise $E(\bar{a})$ reduces to solving a set of k simultaneous equations. Solving such a set of linear equations is easily done by any number of convenient methods.

For the moment, we let w_i=1 for all i, so that all "deviations" have equal weight. (We shall consider non-uniform weighting later in Section 4.3.3.3.)

4.3.3.2. Improvements to the approximation procedure

When trying to determine a reliable estimate of the BER/Reliability measure relationship in the 2-D error subfield (and to a lesser extent the 1-D error subfield) we encountered the problem of not having sufficient data points. The problem was particularly acute in the regions near the mid-point of the plot. This is of course to be expected when the probability density functions are considered, in that the majority of the observations will be made at the extreme ends of the histogram. This effect also becomes more pronounced as higher values of channel E_b/N_0 are considered. We therefore felt that a method of getting more information from the collected statistics was needed.

It is possible to use our assumption of a symmetric channel to say that ideally the relationship between BER and the value of the reliability estimate is symmetric about the 0.5 point. That is, reliability estimates equidistant form the 0.5 point are expected to have equal probabilities of bit error. With this assumption, we may average the two collected statistics which are equidistant from the 0.5 point in order to get a better idea of the actual BER/APP relationship.

For example, Figures 4.9 shows the raw data from simulations of a channel for an E_b/N_0 of 2 dB for the 2-D error subfield. If we average results symmetrically about the 0.5 point we obtain the results shown in Figure 4.10 which are obviously smoother than the previous results.

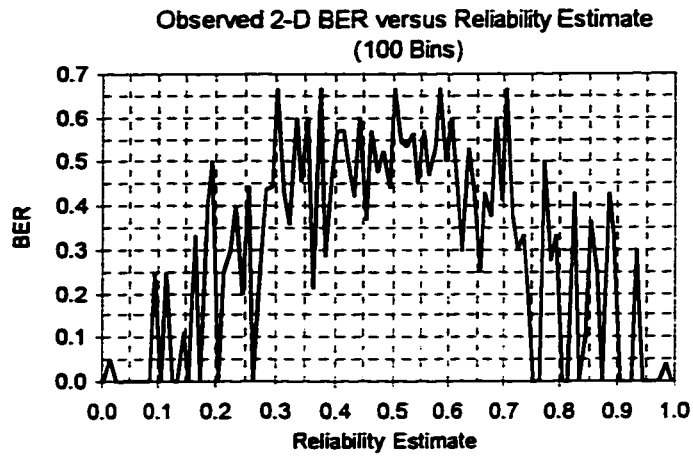


Figure 4.9. Plot of empirical BER versus Reliability Estimate for the 2-D Error Subfield after 2 cycles of MAP processing at a channel E_b/N_0 of 2dB. A total of 563 errors were observed from 1609 bits.

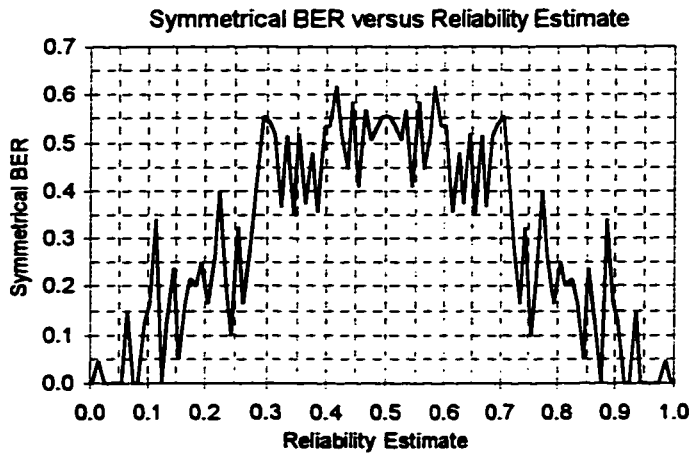


Figure 4.10. Plot of Symmetrical BER versus Reliability Estimate for the 2-D Error Subfield after 2 cycles of MAP processing at a channel E_b/N_0 of 2dB.

Therefore, the polynomial fit to this data can be expected to more reliably describe the BER/Reliability Estimate relationship. This is borne out when one looks at the mean square error between the collected statistics and the fitted polynomials for the two cases. The results in the case of the 2dB channel E_b/N_0 after two MAP iterations considered above are shown in Table 4.1. While this improvement is good, there are further refinements to the process which can be made in order to improve the approximation.

The premise used here is simply that since the actual BER/Reliability Estimate relation is symmetrical about the 0.5 point then it is only necessary to approximate the BER/Reliability Estimate relation from the 0.0 to the 0.5 point in order to fully represent the entire relation over the range from 0.0 to 1.0. As such, since the approximation will be restricted to take place over only half the range, it should be possible to get a better fit for a given order of fitted polynomial.

In this case then, the symmetrical BER is derived first and only the half of the plot from 0.0 to 0.5 need be considered when doing the polynomial approximation. The result is termed the Half Symmetrical BER and is shown in Figure 4.11 below.

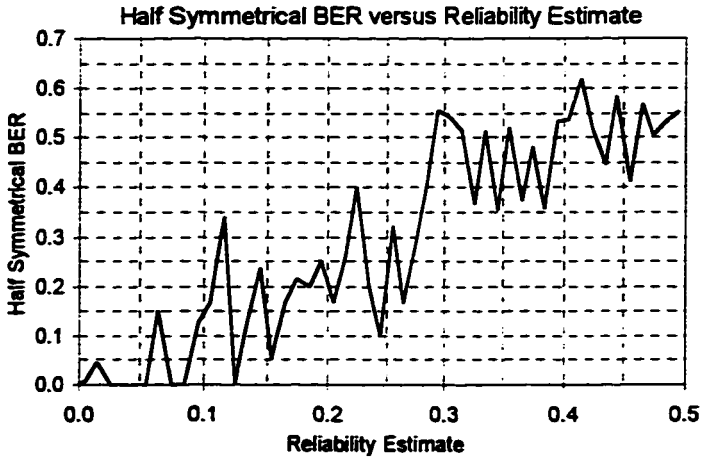


Figure 4.11. Plot of Half Symmetrical BER versus Reliability Estimate for the 2-D Error Subfield after 2 cycles of MAP processing at a channel E_b/N_0 of 2 dB.

The effect of using this method is reflected in a drop in the mean square error as detailed in Table 4.1 below.

Table 4.1. Mean square error and polynomial approximation method used.

Approx. Procedure	Mean Square Error
Normal	0.01880
Symmetrical BER	0.00760
Half Symmetrical BER	0.00756

4.3.3.3. Weighted Error Approximation

In addition to the improvements described in the previous sections, it was felt that a method of tailoring the polynomial approximation process to the nature of the collected statistics was necessary. This was necessary because it was observed that the numbers of errors and estimates present in the bins of the histogram were not constant over the range of reliability estimate. This variation in the number of observations results in a variation of the confidence level which we can attribute to each estimated probability of bit error. Therefore, some method of attaching more weight (or significance) to those observations with greater numbers of observations was felt to be desirable. The resulting polynomial will then pass more closely to those points to which we attach greater confidence as opposed to the polynomial being more 'loose' at those points which have less observations.

The variation in confidence level results from the fact that less observations are made at values of reliability estimates near 0.5 than at values near 0 and 1. This is expected from the distribution of the reliability estimates, as shown in Figure 4.12 below.

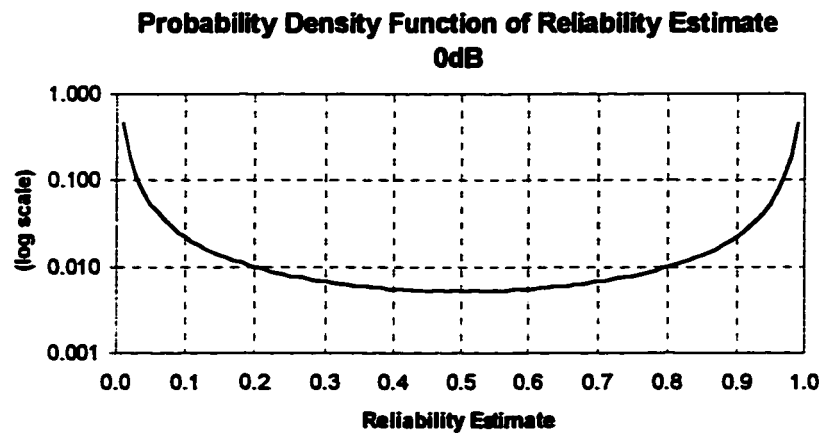


Figure 4.12. Plot of the probability density function of the Reliability Estimate before MAP processing at a Channel E_b/N_0 of 0 dB.

This is further illustrated in the Figure 4.13 below, which is a histogram of the number of observations made in the one-dimensional error subfield versus the value of the reliability

estimates. These statistics were collected during a simulation in which a total of 10,000 errors were observed in all the subfields. It can be seen that the distributions in Figures 4.12 and 4.13 are in fact quite similar.

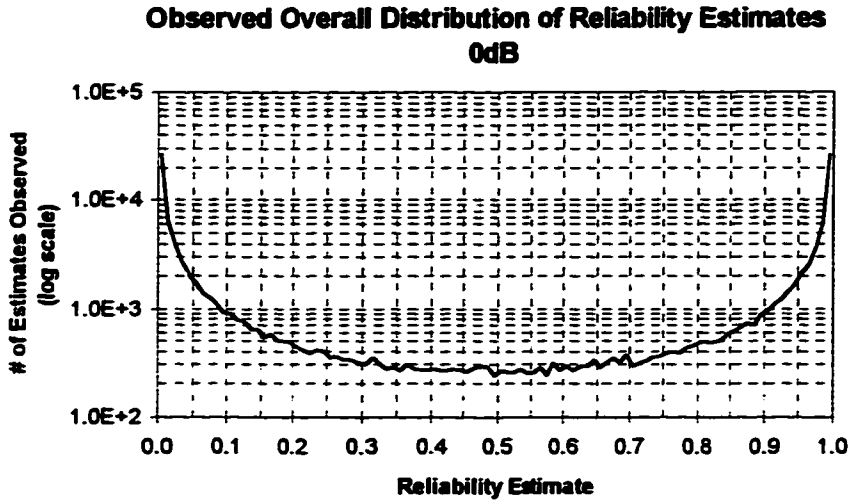


Figure 4.13. Distribution of Reliability Estimates.

As noted above this variation in the distribution of the observations results in a variation of the confidence we can attach to the statistics we collect over the range of the value of the reliability estimates. This is a result of the fact that, under certain conditions, the confidence interval at a given confidence level attached to the statistic can be approximated by a function of the number of errors observed. This expression is now derived.

If n_e errors are observed from a total run of n_b bits, then the observed BER is given by $\hat{p} = n_e/n_b$. From [17] it is shown that the (1–α)×100% Confidence Interval for $\hat{p}$ is given by

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{p(1-p)}{n_b}}, \tag{4.12}$$

where p is the true probability of error and $z_{\alpha/2}$ is the z value corresponding to an area of $\alpha/2$ in the upper tail of a standard normal distribution. The relation assumes errors occur independently. This is not true after more than one MAP iteration, but we assume that the

result is within an order of magnitude of the correct answer. The relation is valid when the distribution can be approximated by a normal distribution. This is true for large n_b, which is considered true under the condition

$$0 < \hat{p} \pm 2\sqrt{\frac{p(1-p)}{n_b}} < 1. \tag{4.13}$$

For large n_b, p can be approximated by $\hat{p}$, leading to the estimated confidence interval

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n_b}}. \tag{4.14}$$

For small $\hat{p}$ (4.14) becomes

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n_b}} \approx \hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}}{n_b}}, \tag{4.15}$$

and the confidence interval width expressed as a percentage of the observed probability of error is given by

$$\frac{\pm 100 \cdot z_{\alpha/2}\sqrt{\frac{\hat{p}}{n_b}}}{\hat{p}} = \frac{\pm 100 \cdot z_{\alpha/2}\sqrt{\frac{n_e/n_b}{n_b}}}{n_e/n_b} = \frac{\pm 100 \cdot z_{\alpha/2}}{\sqrt{n_e}}. \tag{4.16}$$

As an example the observed confidence intervals at a confidence level of 95% for the overall BER based on observing 10,000 errors is shown in Figure 4.14 below.

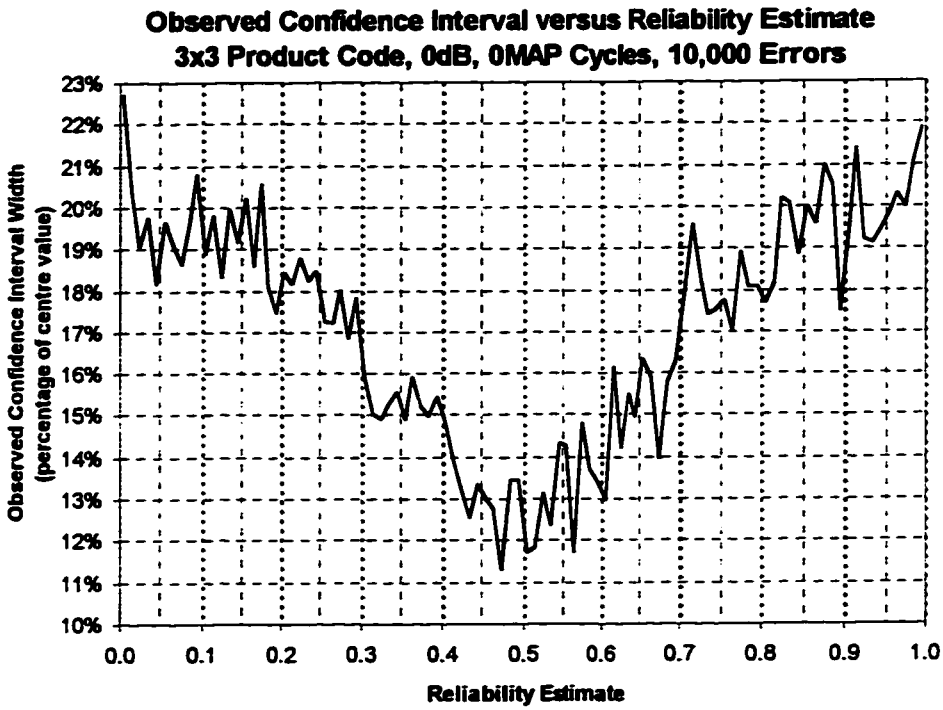


Figure 4.14. Plot of Observed Confidence Intervals at a 95% Confidence Level versus Reliability Estimate for the Overall BER before MAP processing, at a channel E_b/N_0 of 0dB and having 10,000 errors.

Given that the confidence level interval can be approximated by a function of the number of errors observed, it was decided that n_e or some function thereof should serve as the parameter to be used in weighting. The effect of using the number of errors observed raised to various powers as the weighting factor is shown in Figures 4.15 and 4.16 below for the two remapping schemes. Better MAP decoder performance is indicated by the Percentage Difference between Counted BER and Average Probability of Bit Error being close to zero.

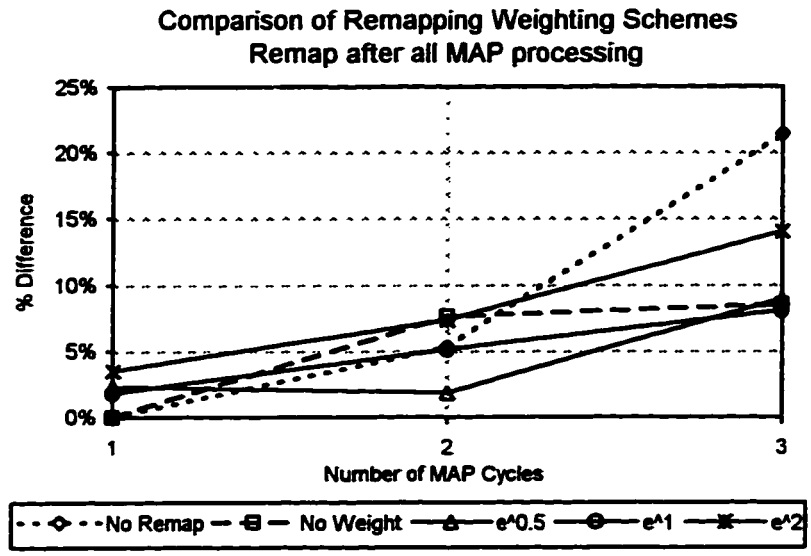


Figure 4.15. Plot comparing Percentage Difference between Counted BER and Average Probability of Bit Error for weighting functions raising n_e to various powers. All simulations were done with remapping after all MAP processing at a channel E_b/N_0 of 2dB using a 9×9 product code.

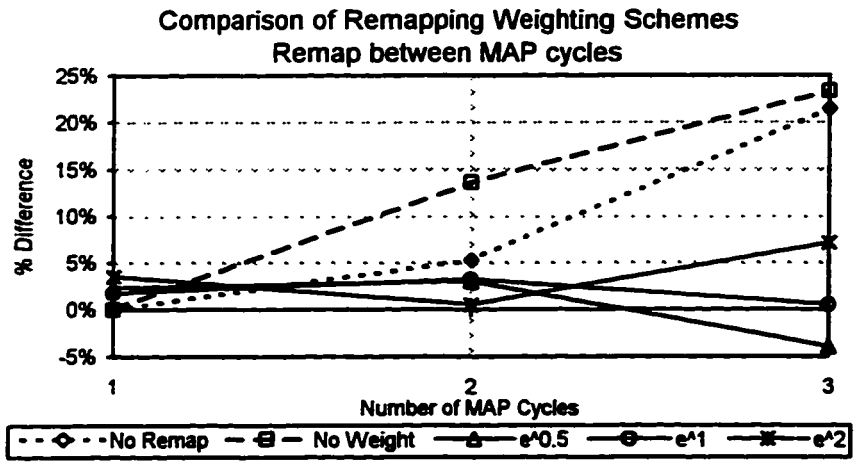


Figure 4.16. Plot comparing Percentage Difference between Counted BER and Average Probability of Bit Error for weighting functions raising n_e to various powers. All simulations were done with remapping between cycles of MAP processing at a channel E_b/N_0 of 2dB using a 9×9 product code.

For remapping after MAP processing, as shown in Figure 4.15, we find that the advantage of doing remapping with weighting only becomes clear when 3 MAP cycles are performed. We note, however, that when $\sqrt{n_e}$ is the weighting function MAP decoder

performance improves–over no remapping–for 2 MAP cycles and so gives the best overall performance in this remapping scheme, with weighting function n_e being a close second.

When remapping between MAP cycles, as shown in Figure 4.16, things are a little more clear than in the previous case. We find that remapping with no weighting performed the worst of the weighting functions, and even worse that doing no remapping, for 2 and 3 MAP cycles. The other weighted remapping schemes all performed better than doing no remapping, with weighting function n_e giving the best performance.

4.3.4. Deriving the remapping function from the polynomial

The final preparatory step in the process is to generate the function which will be used to remap the estimates as they are being decoded. The fitted polynomial described in the previous section serves as the basis for the generation of this "remapping function". The stated purpose of the remapping function is to change the value of reliability estimates output at a specified point in the MAP decoding process to a value which is consistent with the actual BER observed at that point in the process. The method of deriving the remapping function from the fitted polynomial is quite simple and is described below.

Let the polynomial function fitted to collected statistics be denoted by $B_e'(x)$, and let the remapping function derived from $B_e'(x)$ be denoted by $B_e(x)$. Then, $B_e(x)$ is given by

$$B_e(x) = \begin{cases} B'(x); & 0 \le x \le 0.5 \\ 1 - B'(x); & 0.5 < x \le 1 \end{cases} \tag{4.17}$$

This is graphically illustrated in the following figures.

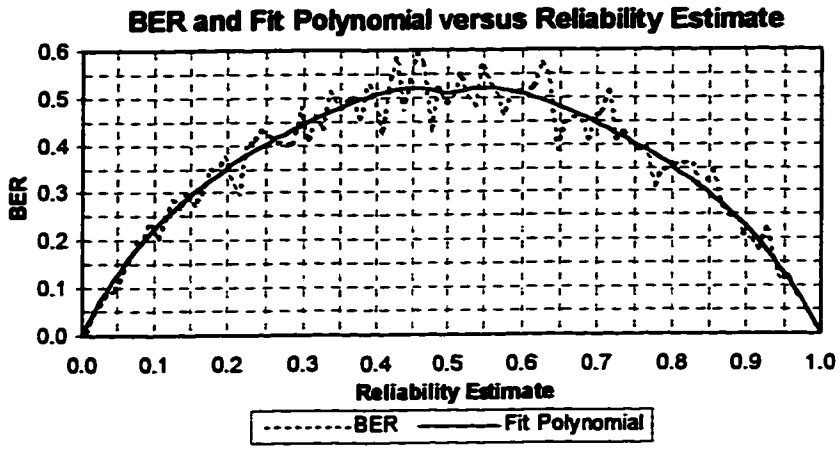


Figure 4.17. Sample plot showing the collect BER statistics and the polynomial approximation versus reliability estimate.

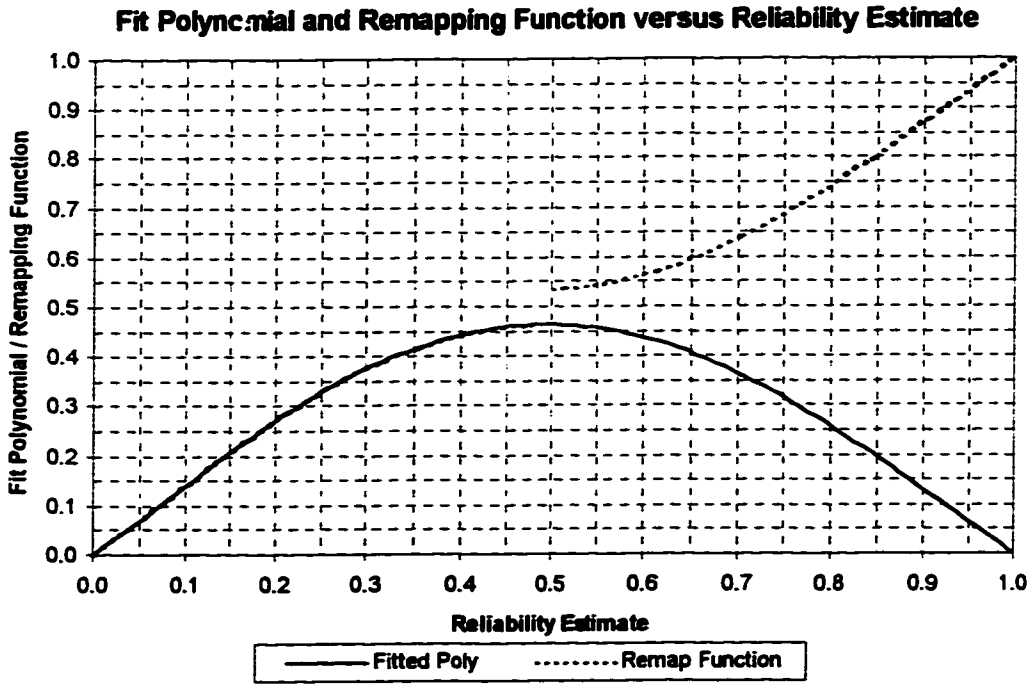


Figure 4.18. Sample plot showing a fit polynomial and the derived remapping function versus reliability estimate.

4.3.5. Application of remapping function to estimates

At this point all the preparatory work necessary for remapping has been completed and the remapping functions can now be applied to the reliability estimates. It is important to note

that the reliability estimates are remapped at the same point in the MAP decoding process at which the remapping functions were generated, and are only applicable at that point. Note that this does not preclude the application of remapping functions derived at several points in the MAP decoding process. In fact, the most significant improvements in the performance of the MAP decoder result when multiple applications of remapping is done. This operation is described in more detail in the following section.

4.3.5.1. Remapping at Different Points in the Decoding Process

If it is desired to carry out remapping at several points in the decoding process, steps must be taken to ensure that the validity of the final estimates output are valid. This process is illustrated in the following example.

Suppose that two iterations of the MAP algorithm will be used to decode the received data stream. To improve the estimates provided, remapping will be carried out on completion of each iteration. The procedure is described below.

(i) Run the MAP algorithm for one iteration, in order to collect the error statistics after one iteration, and from these statistics, generate the functions necessary to do remapping after the first iteration.

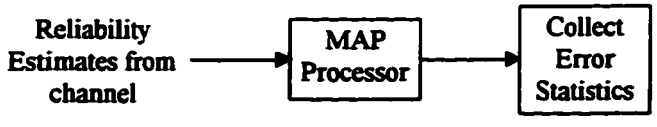


Figure 4.19. Illustration of initial collection of error statistics for remapping between cycles.

(ii) Run the MAP algorithm for two iterations, where remapping is carried out after the first iteration, in order to collect error statistics and generate remapping functions after the second iteration.

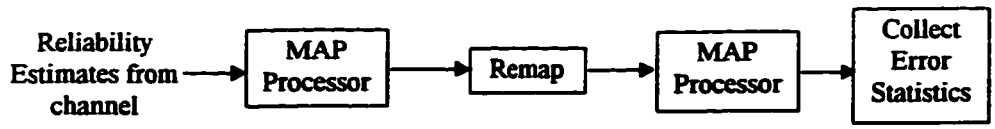


Figure 4.20. Illustration of second collection of error statistics for remapping between cycles.

With the preparatory steps concluded, run the MAP algorithm through two iterations. The remapping functions generated in step one are applied after the first iteration, and the remapping functions generated in step two are applied after the second iteration.

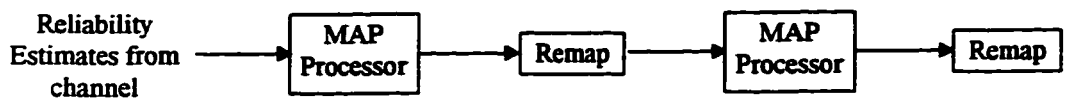


Figure 4.21. Illustration of remapping between cycles.

The above example is an example of what is termed remapping between cycles. The error statistics are collected and the remapping functions applied at the end of each iteration of the MAP algorithm. Other schemes for applying remapping are of course possible and some of these are explored later in this work.

The simulation results in Table 4.2 below illustrate the benefits of remapping the reliability estimates using error statistics collected during the MAP processing. From the results in the table it can be seen that applying remapping improves the accuracy of the estimates over those obtained when no remapping is performed. Note also that remapping between cycles is more effective than remapping only after all of the MAP cycles are completed.

Table 4.2. Results of simulations comparing the effect of the "remapping" schemes with no "remapping".

- *The simulations were carried out using a two-dimensional product code with even parity component codes each 9 bits long.*
- *All simulations carried out at a channel E_b/N_0 of* 2dB.
- *In the table, % Difference is a measure of the accuracy of the reliability estimates, in that it represents the percentage difference between the bit error rate obtained by counting the errors (BER), and the average probability of bit error obtained from the values of the reliability estimates (Average* P[*E*]*).*

Remapping Scheme	# of MAP Iterations	BER	Average P[*E*]	% Difference
No Remapping	1	1.557E-2	1.540E-2	1.10%
	2	1.077E-2	1.020E-2	5.30%
	3	1.056E-2	8.299E-3	21.40%
Remap After	1	1.480E-2	1.482E-2	0.10%
	2	1.068E-2	1.050E-2	1.80%
	3	1.052E-2	9.627E-3	8.50%
Remap Between	2	1.134E-2	1.128E-2	0.50%
	3	1.073E-2	1.058E-2	1.30%

These results are further illustrated in Figure 4.22 below.

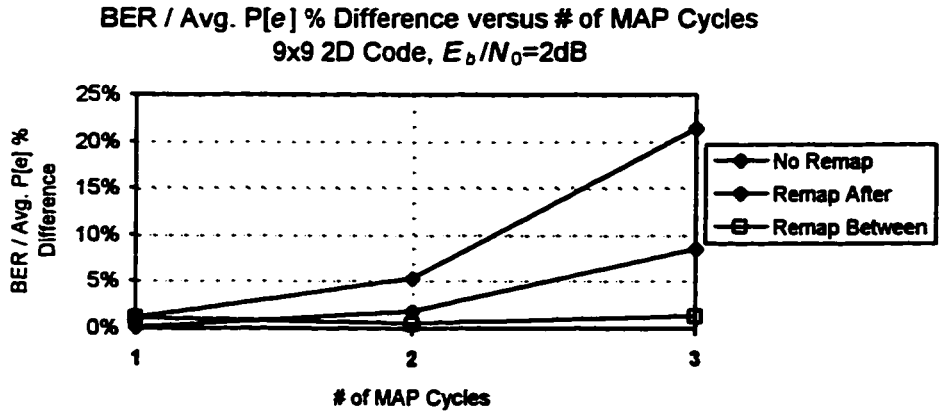


Figure 4.22. Plot of the best results achieved by the weighting scheme $w(x)=n_e(x)$ in each of the two remapping schemes simulated.

As mentioned in Section 4.3.3, the curve fitting to the error statistics collected was done by minimizing a weighted mean square error. It was found that the improvement depended on the weighting scheme used, and that no one weighting scheme was able to

consistently produce the best performance when different remapping schemes were applied. The results in Table 4.3 show the best results achieved for the two remapping schemes investigated as well as results for no remapping and no weighting for comparison. The two remapping schemes shown are

(i) remap after completing all MAP processing, and

(ii) remap after each cycle of MAP processing.

Table 4.3. Results of simulations for various remapping/weighting schemes.

- *The simulations were carried out using a two-dimensional product code with even parity components codes each 9 bits long.*
- *All simulations carried out at a channel E_b/N_0 of 2 dB.*

Cycles	Percentage Difference								
	No Remap	Remap After				Remap Between			
		Weighting Scheme				Weighting Scheme			
		uniform	$\sqrt{n_e}$	n_e	$(n_e)^2$	uniform	$\sqrt{n_e}$	n_e	$(n_e)^2$
1	0.0%	0.0%	2.3%	1.8%	3.5%	0.0%	2.3%	1.8%	3.5%
2	5.3%	7.6%	1.8%	5.1%	7.3%	13.6%	3.0%	3.2%	0.5%
3	21.4%	8.5%	8.9%	8.0%	14.0%	23.3%	-4.0%	1.3%	7.1%

In the Remap After scheme, it was found that not only is the percentage difference between the observed BER and the average probability of bit error reduced, but that the BER also improves slightly, when compared to MAP processing without remapping.

We note that when three MAP cycles are performed all remapping performs better than no remapping, with the sole exception of no weighting when remapping between MAP cycles. For this number of MAP cycles we note that remapping between outperforms remapping after when weighting is performed. We find that the best performance was turned in by remapping between MAP cycles using the number of errors in each bin as the weighting function. This scheme resulted in a percentage difference between observed BER and Average P[E] of only 1.3% as compared to the 23.3% resulting from normal processing. This represents a significant improvement in the performance of the iterated

MAP decoder in providing accurate reliability estimates to be used in the next stage of decoding. And as such, we feel it is well worth the additional processing necessary to incorporate this modification into the iterated MAP decoder, as it may well make the difference between the application of iterated MAP decoding in a concatenated coding scheme or not.

In the second scheme, the resulting BER is slightly higher than that in normal processing, but the percentage difference is lower than that in both normal processing and in scheme (i.) above. Note that there is a trade-off between BER and the accuracy of the reliability estimates to be considered in deciding which scheme should be used.

4.4. Practical Considerations

In an actual implementation of the iterated MAP decoder that incorporated the remapping modification there are a few issues which should be considered. These issues include, but are not limited to

(i) generation of the remapping functions, and

(ii) what action to take when measured E_b/N_0 does not correspond to any stored remapping function.

Each of these issues are now explored further.

4.4.1. Generate Remapping Functions

It will be necessary to have the remapping functions stored in the MAP decoder's memory before actual information is transmitted. For some channels (eg., satellite channels) the AWGN channel model is an accurate model and the expected E_b/N_0 can be estimated and a corresponding range of remapping functions generated. The remapping functions can be generated from simulation of the channel or by training.

In training, a sequence of known data is transmitted and decoded by the iterated MAP decoder from which the error statistics for the current channel conditions can then be

determined. These error statistics can then be used to construct the remapping functions in the manner described previously.

In other applications (eg., mobile communications) it is very likely that the channel characteristics will change over time and from location to location, it is therefore desirable that a training sequence be used in order to determine the remapping functions appropriate to the current channel conditions at the receivers site.

This training sequence can be transmitted at various values of channel E_b/N_0 and the corresponding remapping functions then stored for the values of E_b/N_0 for which the training is carried out. Subsequently, during reception of actual data, the channel E_b/N_0 can be measured and the appropriate remapping functions employed.

This then raises the question of what should be done when it is found that the measured channel E_b/N_0 differs significantly from that for which any of the stored remapping functions were constructed. This issue is considered in the following section.

4.4.2. Measured E_b/N_0 and Stored function mismatch

The occurrence of a mismatch between the measured channel E_b/N_0 and stored function can take place in either of the following two conditions:

(i) measured E_b/N_0 is either greater than the highest, or lower than the lowest, stored function, and

(ii) measured E_b/N_0 is between that of two stored functions.

At the time of writing, we have no general rule on how long the training sequence should be in order to construct accurate remapping functions. We do, however, know that the length required increases with the quality of the channel (on account of the increased time required to observe a satisfactory number of error events). Therefore, if in the first case,

the measured E_b/N_0 is greater than the highest stored function, doing a retraining process would probably not be worth the effort. Especially as the increase in channel quality may well make the need for remapping negligible.

If the measured E_b/N_0 is lower than the lowest stored function, the retraining process will proceed faster, and as such may be carried out, if it is still desired to use the degraded channel.

We therefore see that a good approach is to store a set of remapping functions corresponding to the expected operating range of channel E_b/N_0. If the actual E_b/N_0 exceeds the highest E_b/N_0 for a stored function, use the remapping function with the highest E_b/N_0. If the actual E_b/N_0 drops below the lowest E_b/N_0 for a stored remapping function, then one should retrain.

In case (ii), where the actual E_b/N_0 is between the E_b/N_0 of two stored remapping functions, then an appropriate remapping function can be generated by interpolating between the two stored remapping functions.

At the present time there is no precise knowledge of how long the training sequence should be in order to construct accurate remapping functions. However, it is known that the length required increases with the channel E_b/N_0. Therefore, it is probably best to avoid carrying out the retraining process every time a change in the channel E_b/N_0 is detected. In the first case, it may well be that there is no choice but to do a training sequence for the E_b/N_0 measured.

4.5 Summary

In this chapter the remapping modification to the iterated MAP algorithm was introduced. It was shown to be an effective method in reducing the disparity between the observed BER and the probability of bit error which occurs when the MAP algorithm is used repeatedly.

When the remapping was done between each complete MAP cycle it was found that the percentage difference between the average probability of bit error derived from the reliability estimates and the observed probability of bit error decreased significantly when compared to the iterated MAP decoder without remapping. The best example of this is found when, for 3 MAP cycles, we remap between MAP cycles using a weighted remapping function. In this case, we observed that the percentage difference decreased from 23.3% to 1.3%. There is however a slight increase in the observed BER when compared to the iterated MAP decoder without the remapping modification.

If remapping is done after all MAP processing is complete the percentage difference was also found to be significantly less that that observed when no remapping is done. When the "remap after" scheme is compared to the "remap between" scheme, the BER is observed to be slightly lower when we "remap after". This advantage is however offset by the additional observation that the decrease in percentage difference is not as pronounced as when we "remap between".

We therefore find that we have a choice of which remapping scheme to apply, depending on the relative importance of BER versus quality of reliability estimate.

Chapter 5
Other Modifications

5.1. Overview

In the course of this work two other approaches were tried in the effort to improve the accuracy of the reliability estimates supplied by the iterated MAP decoding process. In this chapter, we present brief descriptions of two other modifications which were investigated as well as their results. Although none of these approaches produced an improvement in the iterated MAP process, they are included here for the sake of completeness. The other modifications explored fall into two categories

(i) identifying error patterns, and

(ii) freezing the values of the reliability estimates.

The first, 'identifying error patterns', was not a modification as such but actually involved studying the output of each cycle of the iterated MAP decoder in order to try and identify the conditions under which errors were occurring. That is to say, an attempt was made to identify any conditions under which errors consistently occurred. For example, whether or not errors occurred only within certain ranges of values of reliability estimates, or under certain arrangements of parity errors. Unfortunately, no such condition(s) could be identified.

In the second category, the possibility that an improvement could be obtained by preventing reliability estimates from tending to 0 or 1 with additional iterations was considered. Specifically, an approach was considered, whereby reliability values were fixed in value according to various criteria. None produced the desired improvement. The details of the procedures attempted are provided in the next section.

5.2. 'Freezing' Estimates

5.2.1. Overview

As described in Section 2.4.1, the iterated MAP decoding algorithm uses the reliability estimates output from the previous cycle as input for the current cycle. The idea in fixing the values from the previous cycle, referred to as 'freezing' here, is that if the values in which there is a high confidence as to their validity are 'frozen', then the other values corresponding to higher uncertainty will have increased accuracy/validity. This, because their values will be based on the 'better' estimates.

Two methods were used to determine which estimates were considered to have a high probability of being valid:

(i) value of estimate, and

(ii) parity equations.

In the first method, those reliability estimates are 'frozen' whose value falls in a certain numerical range which were considered to indicate a high level of validity. In the second method, the estimates that are frozen are those which the parity equations indicate are valid.

5.2.2. Freeze Estimates in range

Given the knowledge of the probability density function of the received signal it is expected that most errors should occur for reliability estimates with values near 0.5. These estimates are those about which one would be the most uncertain about the actual transmitted value.

Now, since the reliability estimates are in the form of the probability, $P[x=0|Y;C]$, their value ranges from 0 to 1. Since we wish to 'freeze' those estimates which are considered 'good', we freeze those estimates with values near to 0 or 1, since these are expected to have a lower probability of error. Those values that lie in the specified range (for example

between 0.0 and 0.2 or between 0.8 and 1.0) are therefore 'frozen' after each cycle of the iterated MAP algorithm.

It was found that 'freezing' probabilities had the effect of increasing the observed BER in all cases, and that this adverse effect increased with the channel E_b/N_0. An example of this is shown in Figure 5.1 below.

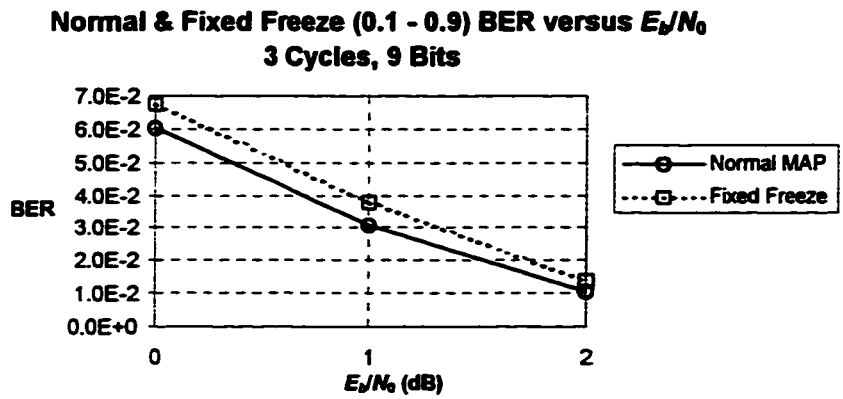


Figure 5.1. Plot of BER versus Channel E_b/N_0 of the unmodified MAP process and the fixed freezing range modification. The 'freezing' range is 0.0 - 0.1 and 0.9 - 1.0

However, when the ranges were tightened, the observed BER was seen to approach that of the unmodified MAP algorithm. Since 'tightening of the ranges' corresponds to decreasing the freezing effect it can be seen that nothing is gained from performing this modification.

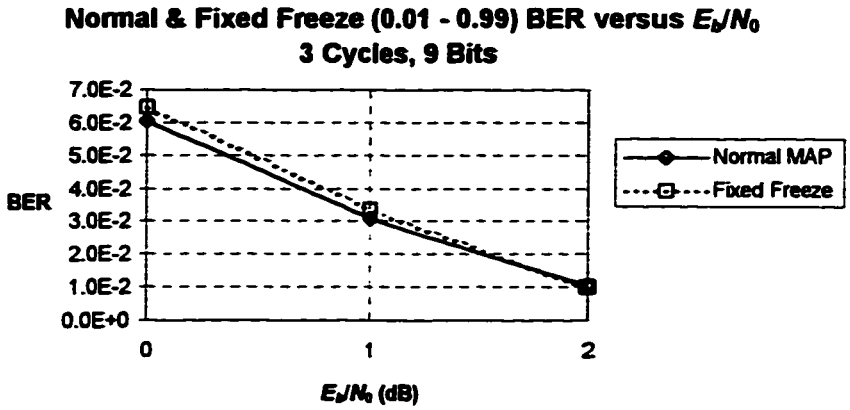


Figure 5.2. Plot of BER versus Channel E_b/N_0 for the unmodified MAP process and the fixed freezing range modification. The 'freezing' range is 0.00 - 0.01 and 0.99 - 1.00.

5.2.2.1. Variable Freezing Ranges

In an effort to improve this modification another variation of the basic freezing idea was investigated. That is, we tried to customise the 'freezing' ranges for each value of channel E_b/N_0 simulated. Specifically, the 'freezing' ranges were tightened at the higher levels of channel E_b/N_0. This further modification resulted in improved performance over the fixed freezing range modification. However, the BER performance still did not equal that of the unmodified iterated MAP algorithm. The results of the simulation using this modification are shown in Figure 5.3 below.

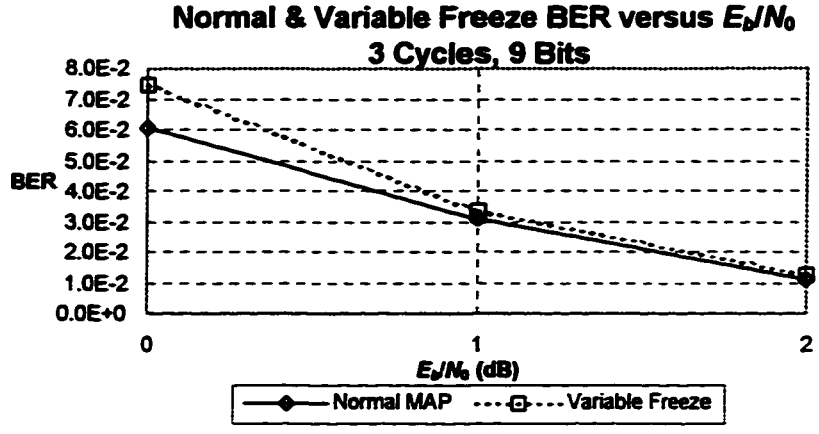


Figure 5.3. Plot of BER versus Channel E_b/N_0 for the unmodified MAP process and the variable freezing range modification.

It can be seen that when these results are compared to Figure 5.1, the deterioration in BER is reduced at the higher levels of channel E_b/N_0. Again, however, the BER of the modified algorithm is still lower-bounded by the BER of the unmodified algorithm.

5.2.3. Freeze Estimates in Subfields

This further modification of the basic 'freezing' idea combines it with subfield error analysis. The parity equations are inspected after each MAP cycle to determine the arrangement of the two error subfields and the no-error subfield as explained in Section 4.2. The contents of the no-error subfield are then 'frozen' for subsequent MAP iterations.

The result of this modification was similar to those obtained in the 'freezing ranges' modification, in that the observed BER increased.

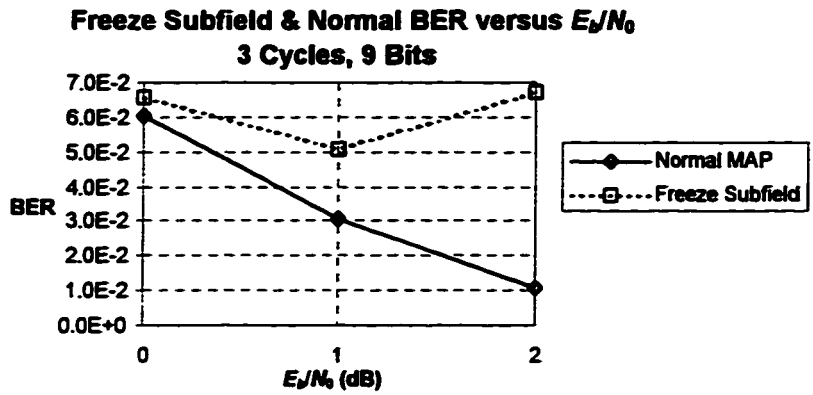


Figure 5.4. Plot of BER versus Channel E_b/N_0 for the unmodified MAP process and the freeze no-error subfield modification.

There is a significant increase in the BER of the modified algorithm when compared to unmodified MAP algorithm, as well as when it is compared with the 'freeze range' modifications described above. This effect is particularly pronounced at higher levels of channel E_b/N_0. This is obviously very unsatisfactory and this approach was abandoned.

5.3. Summary

The search for a readily identifiable pattern to the occurrence of errors proved futile and work on this method of solving the accuracy problem was halted. Also, for the reason mentioned in the previous sections of this chapter, i.e. the deterioration in the observed BER when the reliability estimates are 'frozen', the freezing approach was abandoned and the solution was sought elsewhere.

Chapter 6

Conclusions

6.1. Thesis Summary

This thesis was primarily concerned with determining a method for improving the performance of an iterated MAP decoder through improvements to the accuracy of the reliability estimates output by the decoder. It is important for these estimates to be as accurate as possible, since they are the basis on which another decoder may improve the decisions to be made based on the received signal. This is the case when the MAP decoder is used as the inner decoder of a concatenated coding scheme in which soft decisions are input to the outer decoder. It has been demonstrated that a concatenated coding scheme which passes soft decisions from the inner decoder to an outer decoder capable of exploiting soft decisions exhibits significantly increased decoding power over a scheme which passes hard decisions to the outer decoder. Since the iterated MAP decoder outputs soft decisions it is natural to consider its use in the soft-decision concatenated coding scheme, and it is in this context that this work was carried out.

It was shown in a previous work [5] that when the MAP algorithm is applied repeatedly to a multidimensional product code, the BER decreases with each application until an error floor is reached. However, it was noted that after the second complete pass of MAP processing in each dimension of the code that the accuracy of the reliability estimates began to diminish. Specifically, the reliability estimates became increasingly overly optimistic with each cycle of MAP processing carried out. This property of the iterated MAP decoder is undesirable if it is to be used in a concatenated coding scheme.

It is believed that this behaviour of the iterated MAP decoder can be explained by a closer look at the MAP algorithm itself. In the MAP algorithm there is an assumption that the probability of error occurring in any bit of the block code is independent of the probability

of error in any other bit of the codeword. When the MAP algorithm is applied repeatedly, this assumption is clearly no longer true as the reliability estimate generated for a particular bit in the codeword is calculated using the values for the reliability estimates of every bit in the codeword. As such, the reliability estimate output by the MAP algorithm is dictated by the value of all the reliability estimates of the codeword, and the assumption of independence is clearly invalid after the MAP algorithm has been applied.

This problem is mitigated to some extent when a multidimensional product code is used. Since processing occurs in one dimension at a time, the dependence introduced by processing the current dimension is reduced to some extent, by processing the estimates in the other dimensions. To the extent that the dependence is reduced, the accuracy of the final reliability estimates output may be expected to improve. This leads to the conclusion that as the number of dimensions in the product code is increased, the performance of the decoder will improve, at least initially. However, the problem addressed by this thesis will still occur after a sufficient number of iterations.

The decoder performance is also affected by the power of the component codes used in the product code. In this work, simple parity codes in a two-dimensional product code were used to allow for simple analysis and fast simulation of various modifications to the iterated MAP algorithm.

There was a need for a quantitative measure of the difference between the actual probability of bit error (BER) found and the expected probability of bit error according to the reliability estimates (Avg. P[e]). The measure chosen was the percentage difference between the BER and the Avg. P[e]. The modification made to the iterated MAP decoder should make this percentage difference as small as possible, while preserving the benefit of iteration, i.e., a significant reduction in BER.

One of the modifications attempted was suggested by the realization that the estimates output by the MAP decoder are accurate after the first application of the MAP algorithm.

As such, it was decided to modify the iterated algorithm by keeping constant the values of the estimates which were considered to have a good chance of being accurate after the first complete cycle of MAP processing. Thus this method is called 'freezing'. Two methods of determining which estimates should be accurate were investigated. One was to identify those estimates whose value fell into a certain range near to one or zero. The other involved using our knowledge of the structure of the code to identify the codewords which would be invalid after hard-decision decoding. Unfortunately however, as was shown in Chapter 5, these modifications were ultimately unsuccessful in solving the inaccuracy problem.

The method which was ultimately found to be successful in mitigating the problem of estimate inaccuracy was one which is termed 'remapping'. In this method, the values of the inaccurate reliability estimates are changed to better reflect the actual probabilities of bit error which they are supposed to represent. The values are changed according to a 'remapping function' derived by direct observation of the relationship between reliability estimate and probability of bit error. Simply put, simulations were run and data collected which gave us the probability of bit error for a bit with a given value of reliability estimate. The data was then plotted in a histogram relating probability of bit error and reliability estimate and a quartic polynomial was used to approximate the relationship. Then, in subsequent simulations, the polynomial approximation was used to change the reliability estimates to the new and improved values at the point in the MAP processing where the data was collected.

The knowledge of the structure of the code was exploited by collecting data and remapping in each error subfield. An error subfield is a grouping of the bits in a product code according to whether or not the component codes are found to be valid or not. In a 2-dimensional product code, such as was used in this work, there are three possible subfields into which a bit can be grouped

(i) 2-Dimensional Error Subfield - formed by the intersection of invalid component code in each dimension,

(ii) 1-Dimensional Error Subfield - those bits in a component codeword of one dimension that do not satisfy the parity equations and that are not in a 2-dimensional error subfield, and

(iii) No Error Subfield - those bits of the product code which are not in a 2-dimensional or 1-dimension error subfield.

The polynomial approximation used was to determine the polynomial that minimizes the mean square error between the polynomial and the BER/reliability estimate histogram. The lowest degree of the polynomial which was felt to give a good approximation was four. Additionally, since the channel being modeled is a symmetric channel, it is expected that the histogram will be symmetric about the 0.5 point. This basically allows the averaging of the data about the 0.5 point, and as such gives more data on which the approximation can be carried out.

It was found that in order to get a good polynomial approximation, it is necessary to include some way of assigning a weight to each point of the histogram. This is a result of the fact that the values of BER near to the reliability estimates of 0 or 1 are much more reliable than those values near to reliability estimates of 0.5. For all useful values of E_b/N_0, it is much more likely that the reliability estimates will have values close to 0 or 1. Weighted polynomial approximation schemes were found to give a much closer approximation than non-weighted schemes. The parameter used to weight the values was the number of errors observed at that reliability estimate raised to some power. It was observed that the power applied to the number of errors also affected the accuracy of the approximation, with the best power being 1.0.

The question still remained at which point(s), in the MAP processing, remapping should be carried out. There are several possible remapping schemes, however, only the following two schemes were investigated

(i) remapping after all MAP processing has been completed, and

(ii) remapping after each complete cycle of MAP processing.

It was found that the more effective scheme involved remapping between the applications of the MAP algorithm to all dimensions of the product code, as it resulted in percentage differences very close to zero for multiple cycles of MAP processing. However, this scheme resulted in a slight increase in the final BER when compared to the scheme of remapping after all MAP processing was completed and when no remapping was done at all. This increase in BER was found to have a maximum of 5% and occurred at the third MAP iteration.

The 'remap after' scheme was found to result in a lower BER but less accurate reliability estimates when compared to the 'remap between' scheme. This difference between the two schemes allows us to choose one or the other depending on relative importance of BER versus quality of reliability estimate. If the need for accurate reliability is such that a small increase in BER can be tolerated, then one would choose the 'remap between' remapping scheme. However, if no increase in BER can be tolerated, but one still desires more accurate estimates, then the 'remap after' would be the remapping scheme of choice.

In an effort to verify the remapping procedure, the remapping functions were analytically derived. The remapping functions were verified for a (3,2) parity code at a E_b/N_0, of 0dB for 0 and 1 MAP cycles, and so it can be concluded that the remapping scheme is a valid way of making the reliability estimates more accurate. It was also determined that it was not necessary to constrain the polynomial approximation to pass through the (0.5, 0.5) point.

One disadvantage of remapping is the additional time required to carry out the remapping operation. Another disadvantage is that it is necessary to collect the data necessary for remapping prior to the actual transmission of data. Although it should be possible to analytically determine the remapping functions for simple codes and low numbers of MAP cycles, the difficulty in doing so increases for component codes of any reasonable complexity and high numbers of MAP cycles. The remapping functions can be created by

simulation (in the case of simple, stable channels) or by a training sequence (for more complex, time-varying channels).

6.2 Suggestions for Further Research

It was not possible to derive an analytic relation between subfield error rate and reliability estimate for more than one MAP cycle. If a method of achieving this was actually derived, the results could be used to verify the subfield error rates versus reliability estimate for 2 or more MAP cycles. An added benefit would be that it would no longer be necessary to use the expected long training sequence to get the remapping functions when the decoder is in use.

The effect of remapping between the application of the MAP algorithm in each dimension of the product code could also be investigated. However, it must be pointed out that the author feels that this will not result in a performance improvement great enough to justify the increased complexity, processing power required and additional decoding delay which would result from this approach.

Appendix A

Distribution of Reliability Estimates before MAP

In the following, y_i denotes the output of the matched filter shown in the channel model of Figure 3.1. The probability density function of a given observation, y_i, given a hypothesized value of the corresponding transmitted bit, x_i, is given by

$$f_{Y_i}(y_i|x_i=0)=\frac{1}{\sqrt{\pi\cdot N_0}}\cdot\exp\left(\frac{-\left(y_i+\sqrt{E_b}\right)^2}{N_0}\right). \tag{A.1}$$

The reliability estimate, r_i, is given by

$$\begin{aligned} r_i &= \mathrm{P}[x_i=0|y_i] \\ &= \frac{f_{Y_i}(y_i|x_i=0)\cdot\mathrm{P}[x_i=0]}{f_{Y_i}(y_i)}. \end{aligned} \tag{A.2}$$

We note that the denominator of (A.2) can be expressed as

$$f_{Y_i}(y_i)=f_{Y_i}(y_i|x_i=0)\cdot\mathrm{P}[x_i=0]+f_{Y_i}(y_i|x_i=1)\cdot\mathrm{P}[x_i=1], \tag{A.3}$$

and the reliability estimate is therefore given by

$$r_i=\frac{f_{Y_i}(y_i|x_i=0)\cdot\mathrm{P}[x_i=0]}{f_{Y_i}(y_i|x_i=0)\cdot\mathrm{P}[x_i=0]+f_{Y_i}(y_i|x_i=1)\cdot\mathrm{P}[x_i=1]}. \tag{A.4}$$

Assuming that the values of the a_n are equiprobable, i.e., P[a=0] = P[a=1] = 0.5, then the values of the x_i are also equiprobable and (A.2) simplifies to

$$r_i=\frac{f_{Y_i}(y_i|x_i=0)}{f_{Y_i}(y_i|x_i=0)+f_{Y_i}(y_i|x_i=1)}. \tag{A.5}$$

It is known that the probability density function of y_i conditioned on x_i is given by

$$f_{Y_i}(y_i|x_i=1)=\frac{1}{\sqrt{\pi N_0}}\exp\left[\frac{-\left(y_i-\sqrt{E_b}\right)^2}{N_0}\right] \tag{A.6}$$

and

$$f_{Y_i}(y_i|x_i=0)=\frac{1}{\sqrt{\pi N_0}}\exp\left[\frac{-\left(y_i+\sqrt{E_b}\right)^2}{N_0}\right]. \tag{A.7}$$

Therefore, using this knowledge in (A.5),

$$r_i=\frac{\exp\left(\frac{-\left(y_i+\sqrt{E_b}\right)^2}{N_0}\right)}{\exp\left(\frac{-\left(y_i-\sqrt{E_b}\right)^2}{N_0}\right)+\exp\left(\frac{-\left(y_i+\sqrt{E_b}\right)^2}{N_0}\right)}=\frac{1}{1+\exp\left(\frac{4\sqrt{E_b}y_i}{N_0}\right)}. \tag{A.8}$$

The probability density function of r_i was needed to calculate the probability of error based on the values of the reliability estimates. We can find this from (A.8) by applying the result that states that if Y is a random variable with density function $f_Y(y_i)$ then for $g(\cdot)$ a function which is invertible and everywhere differentiable, the density function of $R=g(Y)$ is given by

$$f_R(r)=\begin{cases}\dfrac{f_Y\left(g^{-1}(r)\right)}{\left|g'\left(g^{-1}(r)\right)\right|}, & r\in \text{range of } g(\cdot);\\ 0, & \text{otherwise.}\end{cases} \tag{A.9}$$

Using our assumption of equiprobable data in (A.3) the probability density function of y can be expressed as

$$
\begin{aligned}
f_r(y) &= \frac{1}{2}\left(f_r(y|x=0)+f_r(y|x=1)\right) \\
&= \frac{1}{2}\left(\frac{1}{\sqrt{\pi N_0}}\exp\left[\frac{-\left(y+\sqrt{E_b}\right)^2}{N_0}\right]+\frac{1}{\sqrt{\pi N_0}}\exp\left[\frac{-\left(y-\sqrt{E_b}\right)^2}{N_0}\right]\right) \\
&= \frac{1}{2\sqrt{\pi N_0}}\left(\exp\left[\frac{-\left(y+\sqrt{E_b}\right)^2}{N_0}\right]+\exp\left[\frac{-\left(y-\sqrt{E_b}\right)^2}{N_0}\right]\right).
\end{aligned}
\tag{A.10}
$$

From (A.9), $g(y)$ has the form $g(y)=(1+\exp(ay))^{-1}$, where

$$
a = \frac{4\sqrt{E_b}}{N_0}. \tag{A.11}
$$

It can be shown that for $0 < r < 1$,

$$
g^{-1}(r) = \frac{1}{a}\ln\left(\frac{1-r}{r}\right) \tag{A.12}
$$

(otherwise the inverse does not exist) and

$$
g'(y) = \frac{-a\cdot e^{ay}}{\left(1+e^{ay}\right)^2}. \tag{A.13}
$$

Combining these two relations, we have that for $0 < r < 1$,

$$
g'\left(g^{-1}(r)\right) = -a\cdot r(1-r). \tag{A.14}
$$

The probability density function of the reliability estimate is therefore given by

$$
f_R(r) = \begin{cases} \dfrac{\left(\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)+\sqrt{E_b}\right)^2\Big/N_0\right]+\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)-\sqrt{E_b}\right)^2\Big/N_0\right]\right)}{2\sqrt{\pi\cdot N_0}\cdot\left|-ar(1-r)\right|}, & 0<r<1; \\ 0, & \text{otherwise.} \end{cases} \tag{A.15}
$$

Since r is always positive and a is a positive constant, the expression $\left|-ar(1-r)\right|$ is always positive and (A.14) reduces to

$$f_R(r)=\frac{\left(\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)+\sqrt{E_b}\right)^2\Big/N_0\right]+\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)-\sqrt{E_b}\right)^2\Big/N_0\right]\right)}{2\sqrt{\pi\cdot N_0}\cdot ar(1-r)}. \tag{A.16}$$

Similarly, it can be shown that the probability density function of the reliability estimate conditioned on the transmitted signal x can be expressed as

$$f_R(r|x=0)=\begin{cases}\dfrac{\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)+\sqrt{E_b}\right)^2\Big/N_0\right]}{\sqrt{\pi\cdot N_0}\cdot ar(1-r)}, & 0<r<1;\\ 0, & \text{otherwise;}\end{cases} \tag{A.17}$$

and

$$f_R(r|x=1)=\begin{cases}\dfrac{\exp\left[-\left(\frac{1}{a}\ln\left(\frac{1-r}{r}\right)-\sqrt{E_b}\right)^2\Big/N_0\right]}{\sqrt{\pi\cdot N_0}\cdot ar(1-r)}, & 0<r<1;\\ 0, & \text{otherwise.}\end{cases} \tag{A.18}$$

These probability density functions are shown in Figure A.1 below for one value of E_b/N_0.

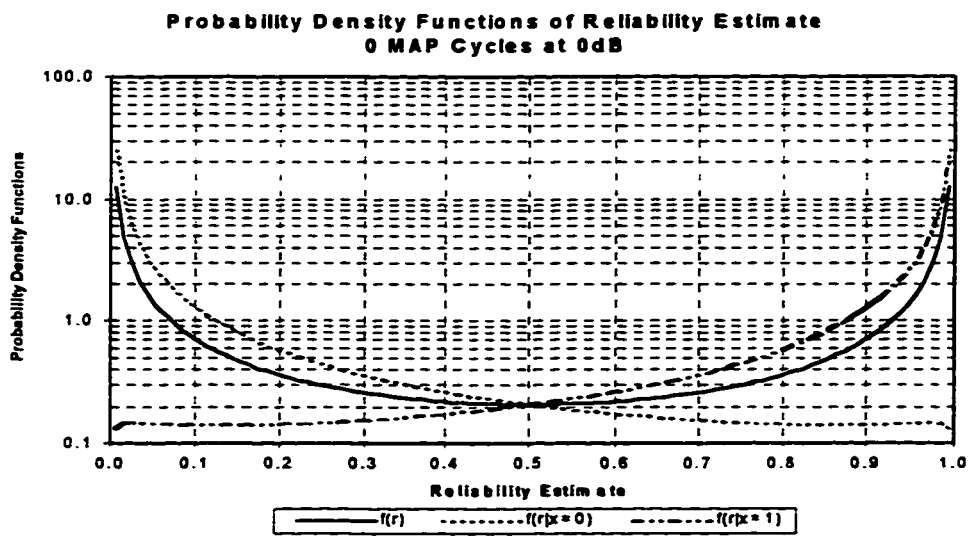


Figure A.1. Plot of the probability density function and conditional probability density functions of the reliability estimate–before MAP processing–of a parity code transmitted in AWGN at an E_b/N_0 of 0 dB.

Appendix B

Probability of Error After MAP

B.1. The (2,1) Even Parity Code

It is known that a (2,1) even parity code is simply to a single repetition code. From communication theory the probability of bit error using such a code and optimally detected antipodal signalling is

$$\mathrm{P}[e_b] = Q\left(\sqrt{4E_b/N_0}\right). \tag{B.1}$$

The P[e_b] of the post-MAP reliability estimates r_i' should be equal to the above expression.

In a (2,1) parity code, the output of the MAP algorithm is given by

$$r_1' = r_2' = \frac{r_1 \cdot r_2}{r_1 \cdot r_2 + \bar{r}_1 \cdot \bar{r}_2}. \tag{B.2}$$

This can be expressed as

$$r_1' = r_2' = \frac{1}{1 + \bar{r}_1 \cdot \bar{r}_2 / r_1 \cdot r_2} = \frac{1}{1 + \left(\frac{1-r_1}{r_1}\right) \cdot \left(\frac{1-r_2}{r_2}\right)}. \tag{B.3}$$

Let $b_i = (1-r_i)/r_i$; since $0 < r_i < 1$, this means $0 < b_i < \infty$.

Therefore

$$r_1' = \frac{1}{1 + b_1 \cdot b_2}. \tag{B.4}$$

Thus, the probability density function of b_i can be expressed in terms of the probability density function of r_i,

$$f_{B_i}(b_i|x_i=0)=\begin{cases}\dfrac{f_{R_i}\left(g^{-1}(b_i)|x_i=0\right)}{\left|g'\left(g^{-1}(b_i)\right)\right|}, & 0\le b_i<\infty;\\ 0, & \text{otherwise;}\end{cases} \tag{B.5}$$

where

$$g(r_i)=b_i=\frac{1-r_i}{r_i} \quad \text{for } 0<r_i<1. \tag{B.6}$$

Therefore $g'(r_i)=-r_i^{-2}$ and $g^{-1}(b_i)=\dfrac{1}{1+b_i}$, which gives

$$g'\left(g^{-1}(b_i)\right)=-(1+b_i)^2. \tag{B.7}$$

Therefore

$$f_{B_i}(b_i|x_i=0)=\begin{cases}\dfrac{f_{R_i}\left(\frac{1}{1+b_i}|x_i=0\right)}{\left|-(1+b_i)^2\right|}, & 0\le b_i<\infty;\\ 0, & \text{otherwise.}\end{cases} \tag{B.8}$$

This expression becomes

$$f_{B_i}(b_i|x_i=0)=\frac{1}{(1+b_i)^2}\cdot\frac{1}{a\sqrt{\pi N_0}}\cdot\frac{1}{\left(\frac{1}{1+b_i}\right)\left(\frac{1}{1-\frac{1}{1+b_i}}\right)}\exp\left[-\left(\ln\left(\frac{1-\frac{1}{1+b_i}}{\frac{1}{1+b_i}}\right)+a\sqrt{E_b}\Big/a\sqrt{N_0}\right)^2\right] \tag{B.9}$$

which reduces to

$$f_{B_i}(b_i|x_i=0)=\begin{cases}\dfrac{1}{ab_i\sqrt{\pi N_0}}\exp\left[-\left(\dfrac{\ln(b_i)+a\sqrt{E_b}}{a\sqrt{N_0}}\right)^2\right] & ;\ b_i>0\\ 0 & ;\ b_i<0\end{cases} \tag{B.10}$$

Let $t=b_1\cdot b_2$, in which case

$$r_i' = \frac{1}{1+t}. \tag{B.11}$$

The probability density function of the random variable T is therefore given by

$$f_T(t|x_i=0) = \int_0^\infty \frac{1}{|b_2|} \cdot f_{B_1,B_2}\left(t/b_2; b_2 | x_i = 0\right) db. \tag{B.12}$$

Since b_1 and b_2 are independent random variables,

$$f_{B_1;B_2}(b_1;b_2) = f_{B_1}(b_1) \cdot f_{B_2}(b_2) \tag{B.13}$$

and (B.12) then becomes

$$f_T(t|x_i=0) = \int_0^\infty \frac{1}{|b_2|} \cdot f_{B_1}\left(t/b_2 | x_i = 0\right) \cdot f_{B_2}(b_2|x_i=0) db_2. \tag{B.14}$$

Using (B.10) in the above expression,

$$\begin{aligned} f_T(t|x_i=0) &= \int_0^\infty \frac{1}{b_2} \cdot \frac{1}{\frac{at}{b_2}\cdot\sqrt{\pi N_0}} \cdot \exp\left[-\left(\frac{\ln\left(t/b_2\right)+a\sqrt{E_b}}{a\sqrt{N_0}}\right)^2\right] \cdot \frac{1}{ab_2\sqrt{\pi N_0}} \cdot \exp\left[-\left(\frac{\ln(b_2)+a\sqrt{E_b}}{a\sqrt{N_0}}\right)^2\right] db_2 \\ &= \frac{1}{a^2\pi N_0 t}\int_0^\infty \frac{1}{b_2}\exp\left[-\frac{1}{a^2N_0}\left(\left(\ln\left(t/b_2\right)+a\sqrt{E_b}\right)^2+\left(\ln(b_2)+a\sqrt{E_b}\right)^2\right)\right] db_2 \end{aligned} \tag{B.15}$$

Using the substitution $z=\ln(b_2)$, (B.15) becomes

$$\begin{aligned} f_T(t|x_i=0) &= \frac{1}{a^2\pi N_o t}\int_{-\infty}^\infty \frac{1}{b_2}\exp\left[-\frac{1}{a^2N_o}\left(\left(\ln(t)-z+a\sqrt{E_b}\right)^2+\left(z+a\sqrt{E_b}\right)^2\right)\right] b_2 dz \\ &= \frac{1}{a^2\pi N_o t}\int_{-\infty}^\infty \exp\left[-\frac{1}{a^2N_o}\left(\left(\ln(t)-z+a\sqrt{E_b}\right)^2+\left(z+a\sqrt{E_b}\right)^2\right)\right] dz \end{aligned} \tag{B.16}$$

Substituting $c = \ln(t) + a\sqrt{E_b}$ we can re-express various terms in (B.16):

$$\left(\ln(t)-z+a\sqrt{E_b}\right)^2 = (c-z)^2 = c^2 - 2cz + z^2 \tag{B.17}$$

and since

$$\left(z+a\sqrt{E_b}\right)^2 = a^2E + 2a\sqrt{E_b}z + z^2, \tag{B.18}$$

$$(c-z)^2+\left(z+a\sqrt{E_b}\right)^2=2z^2+2\left(a\sqrt{E_b}-c\right)z+\left(c^2+a^2E_b\right). \tag{B.19}$$

Using this expression in (B.16)

$$f_T(t|x_i=0)=\frac{1}{a^2\pi N_0 t}\int_{-\infty}^{\infty}\exp\left[-\frac{2}{a^2N_0}\left(z^2+\left(a\sqrt{E_b}-c\right)z+\frac{\left(c^2+a^2E_b\right)}{2}\right)\right]dz. \tag{B.20}$$

By completing the squares,

$$z^2+\left(a\sqrt{E_b}-c\right)z+\frac{\left(c^2+a^2E_b\right)}{2}=\left(z+\frac{a\sqrt{E_b}-c}{2}\right)^2+\frac{c^2+a^2E_b}{2}+\left(\frac{a\sqrt{E_b}-c}{2}\right)^2, \tag{B.21}$$

so (B.20) becomes

$$f_T(t|x_i=0)=\frac{1}{a^2\pi N_0 t}\cdot\exp\left[\frac{2}{a^2N_0}\left(\left(\frac{a\sqrt{E_b}-c}{2}\right)^2-\frac{c^2+a^2E_b}{2}\right)\right]\int_{-\infty}^{\infty}\exp\left[-\frac{2}{a^2N_0}\left(z+\frac{a\sqrt{E_b}-c}{2}z\right)^2\right]dz. \tag{B.22}$$

The expression inside the integral is a scaled version of the probability density function of a Gaussian random variable with a variance given by $\sigma^2=a^2\cdot N_0/2$ and a mean $m=-\dfrac{a\sqrt{E_b}-c}{2}$.

The integral therefore evaluates to

$$\int_{-\infty}^{\infty}\exp\left[-\left(\frac{z-m}{\sigma}\right)^2\right]dz=\sqrt{2\pi}\sigma=\sqrt{2\pi}.\frac{a\sqrt{N_0}}{2}=a\sqrt{\frac{\pi N_0}{2}}. \tag{B.23}$$

The probability density function of t is therefore given by

$$\begin{aligned}f_T(t|x_i=0)&=\frac{1}{a^2\pi N_0 t}\cdot\exp\left[\frac{2}{a^2N_0}\left(\left(\frac{a\sqrt{E_b}-c}{2}\right)^2-\frac{c^2+a^2E_b}{2}\right)\right]\cdot a\sqrt{\frac{\pi N_0}{2}}\\&=\frac{1}{a\sqrt{\pi N_0 t}}\cdot\exp\left[\frac{\left(a\sqrt{E_b}-c\right)^2-2\left(c^2+a^2E_b\right)}{2a^2N_0}\right]\end{aligned} \tag{B.24}$$

The expression inside the exponent is

$$\begin{aligned}\left(a\sqrt{E_b}-c\right)^2-2\left(c^2+a^2E_b\right) &= a^2E_b-2ac\sqrt{E_b}-2a^2E_b-2c^2 \\ &= -\left(c^2+2ac\sqrt{E_b}+a^2E_b\right) \\ &= -\left(c+a\sqrt{E_b}\right)^2.\end{aligned} \tag{B.25}$$

Using the expression for c in (B.24)

$$-\left(c+a\sqrt{E_b}\right)^2 = -\left(\ln(t)+a\sqrt{E_b}+a\sqrt{E_b}\right)^2 = -\left(\ln(t)+2a\sqrt{E_b}\right)^2 \tag{B.26}$$

The probability density function of t is therefore

$$f_T(t|x_i=0) = \frac{1}{a\sqrt{\pi N_0 t}}\cdot\exp\left[-\left(\frac{\ln(t)+2a\sqrt{E_b}}{a\sqrt{2N_0}}\right)^2\right] \tag{B.27}$$

Recall from (B.11) that $r_i' = \frac{1}{1+t} = g(t)$, it is therefore possible to express the probability density function of r_i in terms of the probability density function of t, i.e.

$$f_{R_i'}(r_i'|x_i=0) = \frac{f_T\left(g^{-1}(r_i')|x_i=0\right)}{\left|g'\left(g^{-1}(r_i')\right)\right|}, \tag{B.28}$$

where $g^{-1}(r_i') = \frac{1-r_i'}{r_i'}$ and. $g'(t) = \frac{dr_i'}{dt} = \frac{-1}{(1+t)^2}$

Hence

$$g'\left(g^{-1}(r_i')\right) = -(r_i')^2. \tag{B.29}$$

Equation (B.28) therefore becomes

$$\begin{aligned}f_{R_i'}(r_i'|x_i=0) &= f_T\left(\frac{1-r_i'}{r_i'}|x_i=0\right)\Big/\left|-(r_i')^2\right| \\ &= \frac{1}{(r_i')^2}\cdot\frac{1}{a\sqrt{2\pi N_0\left(\frac{1-r_i'}{r_i'}\right)}}\cdot\exp\left[-\left(\ln\left(\frac{1-r_i'}{r_i'}\right)+2a\sqrt{E_b}\Big/a\sqrt{2N_0}\right)^2\right] \\ &= \frac{1}{a\sqrt{2\pi N_0}(r_i')(1-r_i')}\cdot\exp\left[-\left(\ln\left(\frac{1-r_i'}{r_i'}\right)+2a\sqrt{E_b}\Big/a\sqrt{2N_0}\right)^2\right]\end{aligned} \tag{B.30}$$

which is the final answer.

Now,

$$P[r_i' < \tfrac{1}{2} | x = 0] = \int_0^{\frac{1}{2}} f_{R_i'}(r_i' | x_i = 0) dr_i'$$

$$= \int_0^{\frac{1}{2}} \frac{1}{a(r_i')(1-r_i')\sqrt{2\pi \cdot N_0}} \cdot \exp\left[-\left(\ln\left(\frac{1-r_i}{r_i}\right) + 2a\sqrt{E_b} \Big/ aN_0 \right)^2 \right] dr_i', \tag{B.31}$$

following the same procedure used in (B.4) and (B.5) above, (B.31) evaluates to

$$P[r_i' < \tfrac{1}{2} | x = 0] = Q\left(\frac{-m'}{\sigma'}\right). \tag{B.32}$$

Where $\sigma' = a\sqrt{N_0}$ and $m = -2a\sqrt{E_b}$.

The integral can therefore be evaluated using the Q-function, therefore

$$P[e_b | x_i = 0] = P[r_i' < \tfrac{1}{2} | x_i = 0] = Q\left(\frac{0-m'}{\sigma'}\right) = Q\left(\frac{2a\sqrt{E_b}}{a\sqrt{N_0}}\right) \tag{B.33}$$

$$= Q\left(\sqrt{4E_b/N_0}\right)$$

as expected.

B.2. The 2x2 Product Code

If the component codes of the 2×2 product code are (2,1) even parity codes, then it is easy to apply the results for the parity code to the product code. Consider the MAP processing on the product code after reception as shown in the following figure.

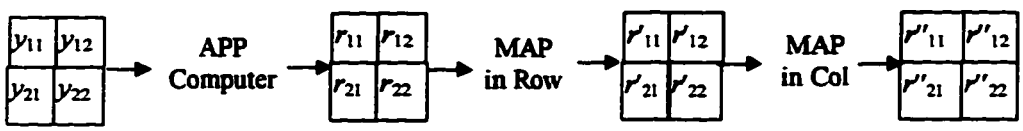


Figure B.1. Model of MAP processing in each dimension of a 2×2 product code.

Up to the point of performing MAP processing in the columns of the product code the results of the previous section hold. After performing MAP processing in the columns the product code can be considered to be a repetition code of length 4, with a corresponding expected P[e_b] given by

$$\mathrm{P}[e_b] = Q\left(\sqrt{\frac{8E_b}{N_0}}\right). \tag{B.34}$$

After MAP processing in the rows the reliability estimates are no longer independent across the rows, however, across the columns the estimates are still independent. Therefore, the same process as used in the previous section can be applied to determine the pdf of the r_{ij}'' .

Note that the probability density function of r_i' is given by

$$f_{R_i}(r_i|x_i=0) = \frac{1}{a(r_i)(1-r_i)\sqrt{2\pi}\cdot\sigma_1}\exp\left[-\left(\left(\ln\left(\frac{1-r_i}{r_i}\right)+m_1\right)\Big/\sqrt{2}\sigma_1\right)^2\right] \tag{B.35}$$

where $\sigma_1 = a\sqrt{\frac{N_0}{2}}$ and $m_1 = a\sqrt{E_b}$. Similarly for r_i',

$$f_{R_i'}(r_i'|x_i=0) = \frac{1}{a(r_i')(1-r_i')\sqrt{2\pi}\cdot\sigma_2}\exp\left[-\left(\left(\ln\left(\frac{1-r_i'}{r_i'}\right)+m_2\right)\Big/\sqrt{2}\sigma_2\right)^2\right] \tag{B.36}$$

where $\sigma_2 = a\sqrt{N_0}$ and $m_2 = 2a\sqrt{E_b}$.

As a result

$$\frac{\sigma_2}{\sigma_1} = \sqrt{2} \tag{B.37}$$

and

$$\frac{m_2}{m_1} = 2. \tag{B.38}$$

Therefore the r_{ij}'' will have a probability density function in the same form, i.e.

$$f_{R_i''}(r_i''|x_i=0) = \frac{1}{a(r_i'')(1-r_i'')\sqrt{2\pi}\cdot\sigma_3}\exp\left[-\left(\left(\ln\left(\frac{1-r_i''}{r_i''}\right)+m_3\right)\Big/\sqrt{2}\sigma_3\right)^2\right], \tag{B.39}$$

where

$$\sigma_3 = \sqrt{2}\sigma_2 = \left(\sqrt{2}\right)\cdot\left(\sqrt{2}\sigma_1\right) = 2\cdot a\sqrt{\frac{N_0}{2}} = 2a\sqrt{N_0} \tag{B.40}$$

and

$$m_3 = 2m_2 = 2\cdot 2m_1 = 4m_1 = 4a\sqrt{E_b} \tag{B.41}$$

The probability of bit error can now be found and is given by

$$\mathrm{P}[e_b|x_i = 0] = \mathrm{P}[r_i' < \tfrac{1}{2}|x_i = 0] = Q\left(\frac{m_3}{\sigma_3}\right) = Q\left(\frac{4a\sqrt{E_b}}{a\sqrt{2N_0}}\right) = Q\left(\sqrt{8E_b/N_0}\right) \tag{B.42}$$

which is the expected result, further confirming the validity of the formulations.

References

[1] Roger L. Freeman, *Telecommunication System Engineering*, Second Ed., John Wiley and Sons, New York, 1989.

[2] S. Lin and D. J. Costello Jr., *Error Control Coding*, Prentice-Hall Inc., N.J., 1983

[3] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications." *Proc. GLOBECOM'89*, Dallas, Texas, pp 47.1.1-47.1.7, Nov. 1989.

[4] J. Lodge, R. Young, P. Hoeher and J. Hagenauer, "Separable MAP 'filters' for the decoding of product and concatenated codes," *Proc. IEEE Int. Conf. Comm.*, pp. 1740-1745, Geneva, Switzerland, May 1993.

[5] J. Lodge, P. Hoeher and J. Hagenauer, "The decoding of multidimensional codes using separable MAP 'filters'," *Proc. Queen's University 16th Biennial Symp. Comm.*, pp. 343-346, May 1992.

[6] C. E. Shannon, "A Mathematical Theory of Communication, Part I," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, July 1948.

[7] C. E. Shannon, "A Mathematical Theory of Communication, Part II," *Bell Syst. Tech. J.*, vol. 27, pp. 623-656, October 1948

[8] G. D. Forney Jr., *Concatenated Codes*, MIT Press, Cambridge, Mass., 1966.

[9] J. G. Proakis, *Digital Communications*, 2nd Edition, McGraw-Hill, 1989

[10] G. D. Forney Jr., "The Viterbi Algorithm." *Proc. IEEE*, vol. 61 pp.268-278, Mar. 1973

[11] J. Lodge and R. Young, "Separable concatenated codes with iterative MAP decoding for Raleigh fading channels," *Proc. of the Fifth Annual Conference on Wireless Comm.*, pp. 703-712, Calgary, Canada, July 1993.

[12] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. IT-24, pp.76-81, January 1978.

[13] G. D. Forney, "Coset Codes - Part II: binary Lattices and Related Codes", *IEEE Transaction. on Inf. Theory*, pp. 1152-1187, Sept. 1988.

[14] V. Sorokine, F. Kschischsang, and V. Durand, "Trellis-Based Decoding of Linear Block Codes", *Information Theory and Applications: Third Canadian Workshop*, pp. 270-286, Springer-Verlag, Berlin, 1994

[15] L. Bahl, J. Cocke, F. Jelinek, J. Raviv. "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate," *IEEE Trans. on Inf. Theory*, Vol. IT-20, pp.284-287, Mar. 1974

[16] J. Lodge, R. Young and P. Guinand, "Separable concatenated codes with iterative MAP filtering," *Information Theory and Applications, Third Canadian Workshop*, pp. 223-240, Rockland, Ontario, Canada, May 1993.

[17] W. Mendehall and R. J. Beaver, *Introduction to Probability and Statistics*, PWS-Kent Publishing Company, Boston, 1990.