A STUDY OF LINEAR ERROR CORRECTING CODES

C. J., Tjhai

Ph.D.
September 2007

A STUDY OF LINEAR ERROR CORRECTING CODES

A thesis submitted to the University of Plymouth
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Cen Jung Tjhai
September 2007

School of Computing, Communications and Electronics
Faculty of Technology
University of Plymouth, UK

A Study of Linear Error Correcting Codes

Cen Jung Tjhai

Abstract

Since Shannon's ground-breaking work in 1948, there have been two main development streams of channel coding in approaching the limit of communication channels, namely classical coding theory which aims at designing codes with large minimum Hamming distance and probabilistic coding which places the emphasis on low complexity probabilistic decoding using long codes built from simple constituent codes. This work presents some further investigations in these two channel coding development streams.

Low-density parity-check (LDPC) codes form a class of capacity-approaching codes with sparse parity-check matrix and low-complexity decoder. Two novel methods of constructing algebraic binary LDPC codes are presented. These methods are based on the theory of cyclotomic cosets, idempotents and Mattson-Solomon polynomials, and are complementary to each other. The two methods generate in addition to some new cyclic iteratively decodable codes, the well-known Euclidean and projective geometry codes. Their extension to non binary fields is shown to be straightforward. These algebraic cyclic LDPC codes, for short block lengths, converge considerably well under iterative decoding. It is also shown that for some of these codes, maximum likelihood performance may be achieved by a modified belief propagation decoder which uses a different subset of n codewords of the dual code for each iteration.

Following a property of the revolving-door combination generator, multi-threaded minimum Hamming distance computation algorithms are developed. Using these algorithms, the previously unknown, minimum Hamming distance of the quadratic residue code for prime 199 has been evaluated. In addition, the highest minimum Hamming distance attainable by all binary cyclic codes of odd lengths from 129 to 189 has been determined, and as many as 901 new binary linear codes which have higher minimum Hamming distance than the previously considered best known linear code have been found.

It is shown that by exploiting the structure of circulant matrices, the number of codewords required, to compute the minimum Hamming distance and the number of codewords of a given Hamming weight of binary double-circulant codes based on primes, may be reduced. A means of independently verifying the exhaustively computed number of codewords of a given Hamming weight of these double-circulant codes is developed and in conjunction with this, it is proved that some published results are incorrect and the correct weight spectra are presented. Moreover, it is shown that it is possible to estimate the minimum Hamming distance of this family of prime-based double-circulant codes.

It is shown that linear codes may be efficiently decoded using the incremental correlation Dorsch algorithm. By extending this algorithm, a list decoder is derived and a novel, CRC-less error detection mechanism that offers much better throughput and performance than the conventional CRC scheme is described. Using the same method it is shown that the performance of conventional CRC scheme may be considerably enhanced. Error detection is an integral part of an incremental redundancy communications system and it is shown that sequences of good error correction codes, suitable for use in incremental redundancy communications systems may be obtained using the Constructions X and XX. Examples are given and their performances presented in comparison to conventional CRC schemes.

Table of Contents

List of Tables

List of Figures

Acknowledgements

This thesis would not be available without the help and support that I obtained whilst I was registered as a PhD student in the University of Plymouth. I would like to express my sincere gratitude to everyone who has helped and supported me in this work.

First of all, it has been a privilege to have Martin Tomlinson as my director of studies. Thanks for all the constant help, support, guidance and encouragement during my research. Thanks also for giving chances to visit other parts of the world as well as your huge effort in getting me as a member of staff in the university.

My other supervisors, Mohammed Zaki Ahmed and Marcel Ambroze, thanks for your support and friendship. I really enjoyed the tour and dinner that we had and to Zaki, thanks for your 'holiday-package' in Kuching during your wedding.

I would like to acknowledge Emmanuel Ifeachor and Joachim Hagenauer–my internal and external examiners respectively, for providing their valuable feedback on this thesis.

The University of Plymouth, for awarding me scholarships in my studies. In fact, I am indebted to Carole Watson, who made the university scholarship possible.

The PlymGRID team of the University of Plymouth, for the high throughput computing resources. Without the grid, some of the results in this work would not be feasible.

Markus Grassl, for providing me with a preprint of Zimmermann (1996) and also for some very useful discussion on coding theory.

My colleagues in Smeaton 209 (Purav Shah, Evangelos Papagiannis, Jing Cai, Xin Xu, Ismail Isnin, Li Yang, and Keiko Takeuchi), for your pleasant company, friendship and of course, parties. Thanks to Purav Shah, for proof-reading parts of this thesis.

Ivy Tiam, for your love and support in any sort of difficulty that I had.

Finally, my beloved family–my parents, sisters and brother, for all your support throughout the years. In particular to my parents, for all the advice, guidance and sacrifices that you have made. Without them, I will not be able to reach this far.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

This study was financed in parts with the aid of the university scholarship, the Overseas Research Student Award Scheme and the Faculty of Technology scholarship.

A programme of advanced study was undertaken, which included the extensive reading of literature relevant to the research project and attendance of international conferences on coding and communications.

The author has published papers in the following peer-reviewed international journals:

1. IET Proceedings Communications (1 paper: volume 1, June 2007, pages 479–488);
2. IEE Proceedings Communicatiosn (2 papers: volume 153, October 2006, pages 581–585; volume 153, April 2006, pages 256–262);
3. Electronics Letters (2 papers: volume 41, no. 3, March 2005, pages 341–343; volume 43, no. 4, February 2007, pages 234–235);

and has presented papers in the following international conferences:

1. IEEE Information Theory Workshop, Rotorua, New Zealand, 28 August – 1 September 2005;
2. IEEE Information Theory Workshop, Chengdu, China, 22–26 October 2006;
3. the 4th International Symposium on Turbo Codes in connection with the 6th International ITG-Conference on Source and Channel Coding, Munich, Germany, 3–7 April 2006;
4. the 10th IEEE International Conference on Communication Systems, Singapore, 30 October – 1 November 2006;
5. the 8th International Symposium on Communication Theory and Applications, Ambleside, Lake District, UK, 17–22 July 2005;
6. the 9th International Symposium on Communication Theory and Applications, Ambleside, Lake District, UK, 16–20 July 2007;
7. IEEE International Magnetic Conference, Nagoya, Japan, 4–8 April 2005; and
8. IEEE International Magnetic Conference, San Diego, USA, 8–12 May 2006;

In addition, the author has also applied for three UK patents (GB0409306.8, GB0428042.6, GB0637114.3) and a US patent (11751313).

Word count of main body of thesis: 62876

Signed : CJ Thai

Date : 25 September 2007

Part I

Introduction and Background

1 Introduction

Suppose that we want to send a message from one end of a communication link to another and we want to make sure that, upon receiving the transmitted message, the receiver makes a decision as to what was transmitted with a minimum probability of error. Errors are inevitable in practical communication systems and they may be caused by noise, interference, distortion and fading. Error correcting codes provide a means of ensuring reliable information transfer from source to destination, and is the subject of this thesis.

1.1 Channel Coding and Reliable Communications

The development of digital communications has made channel coding possible. Harry Nyquist may be regarded as the founder of digital communications. Nyquist's (1924) paper investigated the maximum possible transmission rate that is free from intersymbol interference for a bandlimited channel in telegraph transmission. It was shown by Nyquist (1924) that given a channel of bandwidth B Hz, the maximum theoretical signalling rate without suffering from intersymbol interference is $2B$ symbols per second. This rate of $2B$ symbols per second is commonly known as the Nyquist rate in sampling theorem*. He also found that this rate was only possible for certain pulse shapes such as that of the form $\mathrm{sinc}(t)$.

Another important work in the early development of digital communications was that of Hartley (1928). Ralph Hartley realised that electronics do not have infinite precision and the ability of a receiver to reliably detect the transmitted signal depends on this precision. This, in turn, dictates the maximum rate for reliable transmission over a communications channel. Hartley's (1928) argument may be restated as follows. Assume that the amplitude of the transmitted signal has minimum and maximum values of $-A$ and A respectively, and the precision or accuracy of the receiver is $\pm\delta$. This signal may be divided into slots of size 2δ and there are

$$M = \left(1 + \frac{A}{\delta}\right)$$

slots of this size. If we consider that these M levels may be represented by a binary sequence, then the length of this binary sequence is simply $k = \log_2(M)$. Coupled with the work of Nyquist (1924), this means that signals may be transmitted and reliably distinguished at the receiver at a

*The term *sampling theorem* was first coined by Shannon (1949) and in fact, the idea behind sampling has been known before Shannon's publication. Edmund Taylor Whittaker, an English mathematician, may be regarded as the originator of the sampling theorem. His work on sampling–although it was not called sampling then, was published in the 1915 Proceedings of the Royal Society of Edinburgh, Section A, vol. 35, pages 181–194, entitled "On the functions which are represented by the expansions of the interpolation theory". For a more detailed discussion of the origin of sampling theorem, refer to Luke (1999).

maximum rate of

$$R = 2B\log_2\left(1+\frac{A}{\delta}\right) \quad \text{bits/second} \tag{1.1}$$

$$R = B\log_2\left(1+\frac{A}{\delta}\right)^2 \quad \text{bits/second} \tag{1.2}$$

assuming that the channel bandwidth is B Hz.

Two decades later, Shannon (1948) published his landmark paper on coding and information theory, which showed that the capacity of a channel C of bandwidth B, perturbed by additive white Gaussian noise (AWGN) is given by

$$C = B\log_2\left(1+\frac{S}{N}\right) \quad \text{bits/second} \tag{1.3}$$

where S is the average received signal power and N is the average noise power. We can see that there is a striking similarity between Hartley's (1928) (1.2) and Shannon's (1948) (1.3) formulations; the term A/δ is a voltage ratio, whereas S/N is a power ratio. In the literature, it is common to see (1.3) being referred to as *Shannon-Hartley capacity theorem*.

Shannon (1948) postulated that communications over a noisy channel with arbitrarily small error probability is theoretically possible as long as the transmission rate R is kept below this capacity C. Shannon's (1948) theorem, which is also commonly referred to as the *noisy channel-coding theorem*, shows that the effect of noise on the communication link is not on the reliability of communication, but on how fast we can signal. This essentially means that signal power S and channel bandwidth B are two of the factors that affect the limit of communications, and we may compensate one for the other. We may for example, reduce B at a cost of increased S or reduce S at a cost of increased B to achieve a given reliability. In terms of achieving the capacity C, it is worth noting that increasing the power or bandwidth alone may not be an option. This is because there is a limit on how much power we can increase and as shown in (1.3), the increment is slow since it is logarithmic. Increasing the channel bandwidth, on the other hand, has two contradicting effects since the average noise power is proportional to the bandwidth, i.e.

$$N = \frac{N_0}{2}2B = N_0 B \tag{1.4}$$

where $N_0/2$ is the two-sided noise power-spectral density of the AWGN channel (Proakis; 2001; Sklar; 2001). Using this relationship, we can rewrite (1.3) as

$$C = B\log_2\left(1+\frac{S}{N_0 B}\right) \quad \text{bits/seconds}$$

and the two contradicting effects of increasing B are obvious in this equation.

It is interesting to examine what was the engineering perception to achieving reliable communication prior to Shannon's (1948) formulation. Consider a communication system which transmits a sequence of waveforms corresponding to binary sequences. It is assumed that a one is mapped to the waveform $s(t) = g(t)$ and a zero is mapped to the waveform $s(t) = -g(t)$ where $g(t)$ is an arbitrary waveform which has non zero amplitude at intervals $0 \leq t \leq T$. This type of mapping is

known as the *binary antipodal signalling*. At the receiving end, a signal $y(t) = s(t)+n(t)$ is received, where $n(t)$ represents white Gaussian noise of zero mean and variance σ^2. Matched filtering of the transmitted waveforms is carried out in the receiver prior to detection. A bit error is made if a 0 is sent but a 1 is decided at the receiver and vice versa. For a binary communication system with two equiprobable a priori digits, the probability of bit error P_B may be expressed in terms of the Euclidean distance d_E between the two signal points corresponding to digits 0 and 1, and the noise variance σ^2 (Sklar; 2001)

$$P_B = Q\left(\sqrt{\frac{d_E^2}{4\sigma^2}}\right) \tag{1.5}$$

where $Q(x)$ is the Q-function, i.e.

$$Q(x) = \frac{1}{2\pi}\int_x^{\infty} e^{-u^2/2}du.$$

For binary antipodal signalling, it is sufficient to use just one matched filter and we know that the two signal points are located at $\pm\sqrt{E_b}$, where E_b is the energy per information bit. It is clear that $d_E = 2\sqrt{E_b}$ and following Proakis (2001) and Sklar (2001),

$$\sigma^2 = N_0/2. \tag{1.6}$$

Thus the probability of bit error P_B of bandlimited AWGN channel with binary antipodal signalling is given by

$$P_B = Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \tag{1.7a}$$

Since energy is a product of power and time duration, i.e. $E_b = ST_b = S/R$ where $R = 1/T_b$ is the bit rate, we can also write

$$P_B = Q\left(\sqrt{\frac{2ST_b}{N_0}}\right). \tag{1.7b}$$

From (1.7b), we can see that in order to increase transmission reliability, we may increase the transmitter power or increase the bit duration (i.e. to reduce bit rate). Transmitter power cannot be increased indefinitely, each transmitter has a limit on the average power and there are regulations on this issue. Thus, the only option to have reliable communication is to reduce the transmission rate and in order for $P_B \to 0$, it is necessary that $R_b \to 0$ or $T_b \to \infty$. This was how reliable communication was viewed before 1948.

Shannon's (1948) work completely changed this perception and showed that reliable communication may be achieved by means of channel coding. Given a block of message or information bits, channel coding adds redundancy in a structured and efficient manner to produce a block of longer length, known as codeword. How do the additional redundant symbols help to achieve reliable communication? In order to answer this question, consider the following scenario.

Consider a block of k bits binary information sequence and assume that no channel coding is

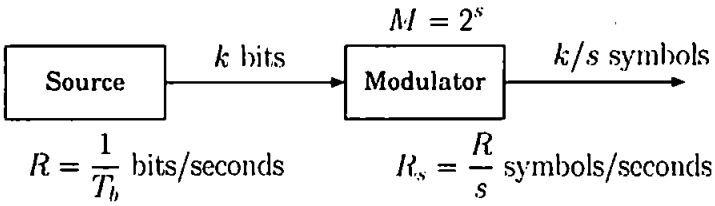


Figure 1.1: Transmitter of a typical communication system without coding

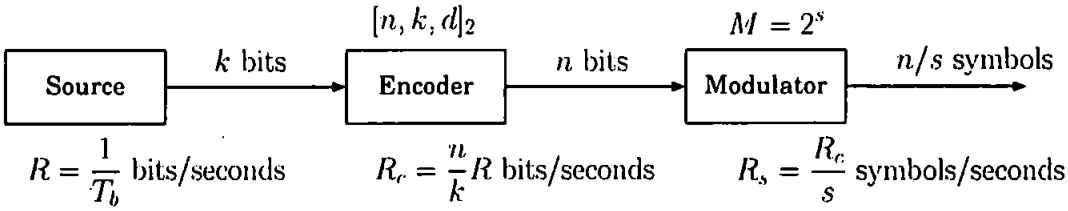


Figure 1.2: Transmitter of a typical communication system with coding

employed, see Figure 1.1 for the block diagram representation of the transmitter of this system. The source emits one bit of information every T_b seconds and we assume that the modulator groups every s information bits, where $s \leq k$, and assigns a waveform to represent these s bits. With $s = 1$, there are $M = 2^k$ possible distinct waveforms, $\{m_1, m_2, \ldots, m_M\}$, and it is assumed that m_1 is the transmitted waveform and that binary antipodal equiprobable signalling is employed. The energy in each information bit is E_b and since each information contains k bits, the total energy is $E = kE_b$ and it is clear that the minimum number of differences among the k bits between any pair in the M waveforms is 1. This result holds for any value of k and it corresponds to a minimum Euclidean distance of $d_E = 2\sqrt{E_b}$. Using an optimum detection criterion, the receiver makes a decision error if the squared Euclidean distance of m_1 with respect to received vector y is larger than that of m_i, for $2 \leq i \leq M$, with respect to the same received vector. For a given value of i, the probability of this error P_e is simply equal to (1.7a). Let $\mathcal{E}_i$, where $2 \leq i \leq M$, denote the event that the squared Euclidean distance of m_i is closer to that of transmitted waveform m_1, with respect to the received vector y. Using the union bound argument (Proakis; 2001), the message error probability P_M is upper-bounded by

$$\begin{aligned} P_M &= \Pr\left(\bigcup_{2 \leq i \leq M} \mathcal{E}_i\right) \\ &\leq \sum_{2 \leq i \leq M} P_e \\ &\leq (M-1)Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \end{aligned} \tag{1.8}$$

The transmitting end of a communication system that uses coding may be seen as in Figure 1.2. Initially, k bits of information is encoded into a codeword of length n bits where $n > k$. We have, as before, the total energy $E = kE_b$ and denoting E_c by the energy per transmitted bit in a codeword,

we also have $E = nE_c$ for the encoded block. Therefore, we can write

$$E_c = \frac{k}{n} E_b. \tag{1.9}$$

A digital modulator groups every s bits in a codeword and a waveform is assigned to these s bits. If E_s is the energy per waveform, we also have $E = \frac{n}{s} E_s$. To make a fair comparison with the uncoded case, we let $s = 1$. The effect of the $n - k$ additional redundant bits is to increase the number of differences between any pair of codewords in the set of 2^k codewords. Let d denote the minimum number of differences, the corresponding Euclidean distance becomes $d_E = \sqrt{d} \cdot 2\sqrt{E_c}$. Using (1.5) and (1.9), the probability of one codeword or message error is

$$P_c = Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) = Q\left(\sqrt{\left(d \cdot \frac{k}{n}\right)\frac{2E_b}{N_0}}\right)$$

and as before, using the union argument, the codeword or message error probability is upper-bounded by

$$P_M \leq (M-1)Q\left(\sqrt{\left(d \cdot \frac{k}{n}\right)\frac{2E_b}{N_0}}\right). \tag{1.10}$$

Comparing (1.7a) and (1.10), we can immediately see that there is a ratio of

$$\gamma_c = d \cdot \frac{k}{n} \tag{1.11}$$

in their difference and this ratio is called the *asymptotic coding gain*. In contrast, the real coding gain is a measure of the amount of power saving that can be attained with coding compared to without coding for a fixed probability of error. In general, the asymptotic coding gain is not equal to the real coding gain, but the quantity γ_c may be regarded as a reliable indicator of how good a code is without the need of running long experiments or simulations.

It is obvious that if a coding scheme for which $d > n/k$ can be devised, reliable communication is possible. Of course, there is a price to pay for this gain. In an uncoded case, we need to transmit k bits for a given duration and with coding, $n - k$ more bits need to be transmitted in the same duration. This means that, assuming that the same type of modulation is employed in both coded and uncoded schemes ($s = 1$), the bandwidth required has been expanded by a factor n/k. It is worth noting that with trellis-coded modulation, of Ungerboeck (1982), $s \neq 1$ and coding does not necessarily require bandwidth expansion.

Consider the block diagram depicted in Figure 1.2 and assuming that we are transmitting information over a channel of bandwidth B Hz at bit rate equal to the channel capacity, $R = C$. There are two important parameters encapsulated in Shannon's capacity equation for AWGN channel, (1.3): the spectral efficiency in bits per second per Hertz and a dimensionless quantity, the signal-to-noise ratio (SNR). Given an AWGN channel of bandwidth B Hz, the spectral efficiency of a coding system that transmits R bits per second is $\eta = R/B$ bits per second per Hertz. Following Costello, Jr. and Forney, Jr. (2007), the spectral efficiency may be written as $\eta = 2r_d$ where r_d is the discrete-time code rate of the coding system. Since $S = E_b R$ and $N = N_0 B$, Shannon's capacity formula in (1.3)

may be rewritten in terms of the discrete-time code rate r_d as

$$2r_d = \log_2\left(1 + 2r_d\frac{E_b}{N_0}\right)$$
$$\frac{E_b}{N_0} = \frac{2^{2r_d} - 1}{2r_d}. \tag{1.12}$$

Equation (1.12) gives a limit of E_b/N_0 (in dB) for various discrete-time code rates to achieve error free communications, for instance $E_b/N_0 \geq 0$ dB for $r_d = 1/2$; $E_b/N_0 \geq 0.86$ dB for $r_d = 3/4$; $E_b/N_0 \geq 1.36$ dB for $r_d = 8/9$; and so forth. This formulation is asymptotic in the sense that it assumes zero probability of error, infinite block length and also infinite input alphabet size (continuous input). For the case $r_d = 0$, we have $E_b/N_0 \geq -1.6$ dB and this value of -1.6 dB represents the *ultimate Shannon's limit* to achieve error free communications. For any communication system, unless the E_b/N_0 operating point is larger than -1.6 dB, reliable information transfer is not guaranteed. Reliable communication may be achieved by means of coding scheme which has $\gamma_c > 1$.

The promise of coding to achieve reliable communications has created a new line of research in both mathematics and communication theory. Many research efforts have been devoted to this subject and to date, there have been many different coding schemes developed. Nowadays, channel coding is an integral part in not only almost every communication systems, but also in storage systems. Section 1.3 gives an outline of some major milestones in the development of coding since Shannon's (1948) landmark paper to date. In order to aid the description of the history of coding as well as the description of the rest of the chapters in this thesis, some useful definitions and notations are given in Section 1.2.

1.2 Definition and Notation

1.1 ***Definition (Linear code).*** Let $\mathbb{F}_q^n$ denote an n-dimensional vector space over a finite field of q elements, $\mathbb{F}_q$. An $[n, k, d]_q$ linear code C is a k-dimensional subset of $\mathbb{F}_q^n$. The quantity d is called the minimum Hamming distance of the code, see Definition 1.6. Each vector in the k-dimensional subset of $\mathbb{F}_q^n$, which has length of n symbols, is called a codeword and may be denoted as $c = (c_0, c_1, \ldots, c_{n-1})$.

The term *linear* arises from the fact that a codeword may be obtained from linear combinations of other codewords and the component-wise sum of all codewords is an all zeros vector. It is assumed that operations such as addition and multiplication are performed under the algebra of $\mathbb{F}_q$.

Note that, throughout this thesis, if the subscript q is dropped from the notation of C, i.e. $[n, k, d]$, we shall assume that it means $[n, k, d]_2$.

1.2 ***Definition (Code rate).*** The code rate of an $[n, k, d]_q$ linear code C is the ratio k/n.

1.3 ***Definition (Hamming weight).*** For a vector $v = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_q^n$, the Hamming weight of

v–denoted by $\mathrm{wt}_H(v)$, is the number of non zero elements in the vector. That is

$$\mathrm{wt}_H(v) = |\{v_i \neq 0 \mid 0 \leq i \leq n-1\}|.$$

1.4 ***Definition (Support).*** The support of a vector $v \in \mathbb{F}_q^n$, $\sup(v)$, denotes a set of coordinates of v for which the value is non zero,

$$\sup(v) = \{i \mid v_i \neq 0,\ 0 \leq i \leq n-1\}.$$

1.5 ***Definition (Hamming distance).*** Let vectors $u, v \in \mathbb{F}_q^n$, the Hamming distance of u and v, denoted by $\mathrm{d}_H(u, v)$, represents the number of coordinates in which u and v do not agree,

$$\mathrm{d}_H(u, v) = \mathrm{wt}_H(u - v) = |\{(u_i - v_i) \neq 0 \mid 0 \leq i \leq n-1\}|.$$

1.6 ***Definition (Minimum Hamming distance).*** The minimum Hamming distance of C, denoted by d, is the smallest value of all Hamming distances between any two distinct codewords of C. Because of the linearity property, d can also be defined as

$$\begin{aligned} d &= \min\{\mathrm{d}_H(c, c') \mid \text{for all } c, c' \in C\} \\ &= \min\{\mathrm{wt}_H(c) \mid \text{for all } c \in C\}. \end{aligned}$$

A code of Hamming distance d is capable of correcting all $0 \leq t \leq \lfloor (d-1)/2 \rfloor$ symbol errors.

1.7 ***Definition (Generator matrix).*** An $[n, k, d]_q$ linear code C has a generator matrix G, which contains k linearly independent codewords of C. A codeword $c \in C$ may be obtained by taking any linear row combination of G, an $k \times n$ matrix.

The matrix G can be transformed into a reduced-echelon or systematic form by elementary row operations and if necessary, some column permutations. In systematic form, the first k coordinates of G is an identity matrix. Hence, for an arbitrary information vector $u \in \mathbb{F}_q^k$, $c = uG$ and $c \in C$ is a systematic codeword where the first k symbols are u and the remaining $n-k$ coordinates contain the redundant symbols.

1.8 ***Definition (Parity-check equation and matrix).*** The parity-check matrix H–an $(n-k) \times n$ matrix, of a code C contains $n-k$ linearly independent vectors of $\mathbb{F}_q^n$ so that $GH^T = 0$. Equivalently, this implies $cH^T = 0$ for all $c \in C$. Each row in the matrix H is called a parity-check equation.

Similar to G, H may be transformed into a reduced-echelon form such that the last $n-k$ columns of H form an identity matrix. Given an arbitrary information vector $u \in \mathbb{F}_q^k$, $c \in C$ may also be obtained from H, which is in reduced echelon form, by taking the component-wise product of u and

the first k columns of H, i.e. the ith parity-check symbol is $u\bar{H}_i^T$ where $\bar{H}_i$ is the ith row of a matrix formed from the first k columns of H.

1.9 ***Definition (Dual code).*** The dual code of an $[n, k, d]_q$ linear code C, denoted by $C^\perp$, is an $[n, n-k, d']_q$ linear code where $c^\perp c^T = 0$ for all $c \in C$ and all $c^\perp \in C^\perp$.

While any linear row combination of G produces c, any linear row combination of H produces $c^\perp$. The matrices H and G are the generator and parity-check matrices of $C^\perp$ respectively.

1.10 ***Definition (Syndrome).*** Given a vector $v \in \mathbb{F}_q^n$, the vector $s \in \mathbb{F}_q^{n-k}$ defined by

$$s = vH^T$$

is the syndrome of a code whose parity-check matrix is H. If the vector $v \in C$, then $s = 0$ otherwise at least one coordinate of s has non zero value.

1.11 ***Definition (Hamming weight enumerator polynomial and weight distribution).*** Given an $[n, k, d]_q$ linear code C, let $A_i = |\{\mathrm{wt}_H(c) = i \mid \text{for all } c \in C\}|$, i.e. the number of codewords of Hamming weight i. The Hamming weight enumerator polynomial of C is given by

$$A_C(z) = \sum_{i=0}^{n} A_i z^i$$

where z is an indeterminate. The distribution of A_i for $0 \le i \le n$ is known as the weight distribution of C.

1.3 Historical Development of Channel Coding

Shannon (1948) did not show how to devise a coding scheme that maximises the quantity γ_c, although he did give an example of encoding 4 bits of information to a codeword of 7 bits long. This coding scheme, which is due to R. Hamming, has $d = 3$ and this translates to $\gamma_c = 1.714$ or equivalently 2.34 dB. Assuming binary antipodal signalling, plotting (1.7a) against E_b/N_0, we will see that $P_B = 10^{-5}$ is achieved at $E_b/N_0 = 9.6$ dB. Compared to the ultimate Shannon's limit, the coding scheme due to Hamming is approximately 8.86 dB away and obviously there must be a better coding scheme that can bring the gap to Shannon's limit closer. This sparked challenges to communication engineers and mathematicians to design codes that maximise d for a given n and k (or equivalently maximise k for a given d and n). The quest to approach Shannon's limit began and this also marked the birth of coding theory.

The scheme introduced by Hamming, which was subsequently called the Hamming codes and was published in Hamming (1950), is a class of binary codes, with parameter $[2^m - 1, 2^m - 1 - m, 3]$ for $m \ge 2$, that is capable of correcting any single error. The example given in Shannon (1948) is for $m = 3$. It is interesting to note that, according to Blake (1973), Fisher[1], before coding theory

[1] Fisher, R. (1942), "The theory of confounding in factorial experiments in relation to the theory of groups", *Ann. Augenics*, 11, 341–353

was even born, the Hamming code of length 15 bits was already known, along with its weight distributions. In addition to Hamming's (1950) work, Golay (1949) introduced a 3-error correcting codes which added 11 bits of redundancy to protect 12 bits of information. The $[23, 12, 7]_2$ Golay code has become a toy code for mathematicians due to the very nice structure the code possess. All patterns up to 3 errors are uniquely represented by the 2^{11} syndromes. A code that satisfies this property, which may be mathematically stated as $\sum_{t=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{t}(q-1)^t = q^{n-k}$, is called a *perfect* or lossless or close-packed code. To date, the Hamming codes, the $[23, 12, 7]_2$ and $[11, 6, 5]_3$ Golay codes are the only non-trivial perfect codes known.

Studies of error-correcting codes in the 1950s were code-specific. A generalisation to the theory of error-correcting codes, including the decoding method (using the standard array), was given by Slepian[2]. The generalised decoding method, however, is only efficient for very short codes. Nevertheless, Slepian's work is an important contribution to coding as it sets an algebraic framework to this subject.

It soon became clear that finding a good coding scheme is not the only problem to approach the Shannon's limit. Given a code that could theoretically give large coding gain, the complexity of the matched filter decoder was often prohibitive to realise this gain. Hamming and Golay codes are good examples. The technologies in the 1950s only allowed exhaustive decoding of some very short codes such as the $[7, 4, 3]_2$ Hamming code. The Golay code which gives $\gamma_c = 5.63$ dB, on the contrary, had impractical decoding complexity. Thus, there are contradicting aims to approach Shannon's limit; long codes which have larger coding gain are undecodable practically, meanwhile short codes which have practical decoders are not effective. It is not surprising that not long after Shannon's (1948) formulation, there was a great deal of effort dedicated towards the construction of codes which have efficient decoder structures.

A simple decoding, but non optimum, method suitable for the technology in the 1950s was the one based on majority voting, the *majority logic decoding*. A class of binary codes that could be decoded in this manner was given by Muller[3] and an efficient majority logic decoding algorithm was introduced by Reed[4]. These codes are subsequently known as the Reed-Muller (RM) codes. Similar to the case of the $[15, 11, 3]_2$ Hamming code, according to Peterson and Weldon, Jr. (1972), the RM codes had already been known by Mitani[5] in 1951. The RM codes are a precursor to a class of majority-logic decodable codes which include the difference set cyclic codes of Weldon, Jr. (1966) and the finite geometry codes.

Albeit having low decoding complexity, the error correcting capability of RM codes is relatively poor and a quest for better codes with simple decoders was necessary. Communication engineers and mathematicians started to associate algebra with the design of codes and the outcomes were fruitful; the discovery of the Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes. The BCH codes were discovered independently by Hocquenghem[6] in 1959 and by Bose and Ray-

[2]Slepian, D. (1956), "A class of binary signalling alphabets", *Bell System Technical Journal*, 35, 203–234

[3]Muller, D. (1954), "Application of Boolean algebra to switching circuit design and to error detection", *IRE Transactions on Electronics and Computers*, 3, 6–12

[4]Reed, I., (1954), "A class of multiple-error-correcting codes and the decoding scheme", *IRE Transactions on Information Theory*, 4, 38–49

[5]Mitani, N. (1951), "On the transmission of numbers in a sequential computer," National Convention of the Institute of Electrical Communication Engineers of Japan

[6]Hocquenghem, A. (1959), "Codes correcteurs d'erreurs", *Chiffres*, 2, 147–156

Chaudhuri[7] in 1960. They are t-error correcting linear codes which have a decoder that is guaranteed to correct all error patterns up to t errors. In addition to having low complexity decoder, the BCH codes also have rich mathematical structure and in fact they form a subclass of codes introduced by Prange in 1957[8] and 1958[9], cyclic codes. The cyclic nature of the BCH codes was first shown by W. W. Peterson in his paper in 1960[10]. This paper was also the first that described the error correction procedure for binary BCH codes. The decoding procedure for non binary BCH codes was attributed to Gorenstein and Zierler in 1961[11]. Both decoding techniques, however, are not efficient and Berlekamp[12] in 1966 and Massey[13] in 1969 set an important milestone in channel coding with their invention of an efficient decoding algorithm for BCH codes.

Another important class of algebraic codes are the RS codes, which were discovered by Irving Reed–who also invented the RM codes, and Gustave Solomon[14]. This class of codes turn out to be a special case of BCH codes and they are optimum in the sense of having the highest error correcting ability attainable by any linear code of the same field. The decoding algorithm developed by Berlekamp and Massey is applicable to RS codes. The fact that RS codes are non binary, optimum and efficiently decodable has attracted a great deal of practical interest. Each non binary symbol in a codeword of an RS code may be mapped to a binary vector and since the optimality of RS codes is defined symbol-wise, this created an opportunity for RS codes to be used in applications that require burst error correction.

All the efficient decoding procedures introduced in the period of the 1950s and 1960s involve quantisation of the received signal to levels which are equal to the field size of the code. This, inevitably, results in a loss as the reliability information of each symbol is excluded by the decoder. It was realised that in order to bring the gap to Shannon's limit closer, this loss–which is typically 2 dB for binary quantisation, must be eliminated. The *hard-decision* decoder is the name given to the decoder that does not take the symbol reliability information into account and the one that does is called a *soft-decision* decoder. In this period, the lack of development in soft-decision decoding was attributed to its complexity, which is of course higher than that of the hard-decision counterpart, and also the state-of-the-art of electronics at the time.

Another important development in channel coding was the invention of convolutional codes by Elias in 1954[15]. All the codes mentioned earlier fall into a type known as block codes. With block codes of length n symbols and dimension k symbols, each user message is partitioned into blocks of

[7]Bose, R and Ray-Chaudhuri, D. (1960), "On a class of error correcting binary group codes", *Information and Control*, 3, 68–79

[8]Prange, E. (1957), "Cyclic error-correcting codes in two symbols", *Report AFCRC-TN-57-103*, Air Force Cambridge Research Center, Bedford, Mass, USA

[9]Prange, E. (1958), "Some cyclic error-correcting codes with simple decoding algorithms", *Report AFCRC-TN-58-156*, Air Force Cambridge Research Center, Bedford, Mass, USA

[10]Peterson, W. (1960), "Encoding and error-correction procedures for the Bose-Chaudhuri codes", *IRE Transactions on Information Theory*, 6, 459–470

[11]Gorenstein, D. and Zierler, N. (1961), "A class of error-correcting codes in p^m symbols", *Journal of the Society for Industrial and Applied Mathematics*, 9, 207–214

[12]Berlekamp, E. (1966), "Nonbinary BCH decoding", Institute of Statistics Mimeo Series 502, Dept. Statistics, University of North Carolina, Chapel Hill, N.C.

[13]Massey, J. (1969), "Shift-register synthesis and BCH decoding", *IEEE Transactions on Information Theory*, 15, 122–127

[14]Reed, I. and Solomon, G. (1960), "Polynomial codes over certain finite fields", *Journal of the Society for Industrial and Applied Mathematics*, 8, 300–304

[15]Elias, P. (1954), "Error-free coding", *IRE Transactions Information Theory*, 4, 29–37

k symbols. Each of these blocks is mapped to a larger block, n symbols, which is then used for transmission. With convolutional or *recurrent* codes, however, partitioning of user messages is avoided. The convolutional encoder reads a small fraction of the user message–say k' symbols, and generates a block of length n' symbols, where $n' > k'$, for transmission. Each of the n' symbols generated does not depend on the current k' symbols of the user message only, but it is also a function of some previous symbols of the user message. The number of symbols in the previous user message that are involved in generating a block for transmission are dictated by the memory or constraint length of a convolutional code. Unlike block codes, the structure of convolutional codes with short memory allows efficient implementation of soft-decision decoding algorithms. Soft-decision decoding of convolutional codes was first introduced by Wozencraft in 1957[16] which was then improved by Fano in 1963[17]. This decoder, known as a *sequential decoder*, is a suboptimal soft-decision decoder. The suboptimality was solved with the introduction of a maximum likelihood decoder, known as the Viterbi algorithm (Viterbi; 1967), which was further developed by Forney, Jr. (1973) using trellis diagrams. The introduction of a trellis diagram for convolutional codes marked another important milestone in channel coding. Efficient optimum soft-decision decoder of convolutional codes would not be possible without the knowledge of their trellis representation. The Viterbi's (1967) algorithm is optimum only in terms of the detected codeword, that is the algorithm returns a codeword, which among all of the possible transmitted codewords, has the highest probability of being correct for a given sequence received from the channel. There also exists optimum soft-decision decoders in terms of symbol, that is for a given coordinate in a received sequence, this decoder returns a symbol that has the highest probability of being correct compared to the other symbol possibilities. The algorithms by Bahl et al. (1974) and Hartmann and Rudolph (1976) are instances of optimum symbol-by-symbol soft-decision decoders. It is worth mentioning that the algorithm by Bahl et al. (1974), which is subsequently known as the BCJR algorithm, also makes use of a trellis diagram. To the best of the author's knowledge, the trellis-based optimum soft-decision decoder for block codes was also first introduced by Bahl et al. (1974). While the trellis representation of convolutional codes is invariant over time, this is not the case for block codes and as a consequence, it prohibits the development of efficient, optimum soft-decision decoders for block codes. It is not surprising that, in the 1990s, one of the active research subjects was the trellis complexity of block codes.

In general, the longer the code the larger the coding gain. Forney, Jr. (1966) introduced a coding scheme which allowed two codes–a non binary outer code and a binary inner code, to be cascaded to form a longer and more powerful binary code. This coding scheme, which is termed *concatenated codes*, restricts the decoding complexity to that of the component codes. This is because the decoding can be done in two stages: inner code decoding in the first stage, followed by outer code decoding to correct the residual errors from the decoding of the inner code. The Consultative Committee on Space Data Systems (CCSDS) adopted Forney, Jr.'s (1966) concatenation as the NASA/ESA Telemetry Standard in 1987 (Costello, Jr. et al.; 1998). The adopted arrangement used a rate 1/2 convolutional code as the inner code and a $[255, 223, 33]_{2^8}$ RS code as the outer code. A generalisation of Forney, Jr.'s (1966) concatenation was introduced by several Russian mathematicians, Blokh and Zyablov (1974) and Zinov'ev (1976). The resulting work has resulted in the construction of many

[16] Wozencraft, J. (1957), "Sequential decoding for reliable communication", *1957 National IRE Convention Record*, 5(2), 11–25

[17] Fano, R. (1963), "A heuristic discussion of probabilistic decoding", *IEEE Transactions on Information Theory*, 9(2), 64–74

codes with attractive asymptotic coding gains.

The introduction of combined coding and M-ary modulation by Ungerboeck (1982) was an important step in the quest to approach Shannon's limit. Before this combined scheme, which is commonly referred to as *trellis-coded modulation* (TCM), it was believed that coding, in bringing the gap to channel capacity closer, required larger bandwidth. It was thought that coding would only be suitable for applications that have large bandwidth availability such as satellite communications; for data transmission over telephone channel, where the bandwidth is around 4 kHz only, coding was considered to be inefficient. The TCM scheme, which puts emphasis on joint coding and modulation, dispelled this belief and introduced significant data rate improvements for transmission using modems over the telephone channel. The pre-TCM modem standard was only able to provide data rate of 9.6 kilo bits per second half-duplex, meanwhile with TCM full duplex data rate as high as 33.6 kilo bits per second (V.34 modem standard) was made possible.

Despite the many achievements accomplished between 1948 and the early 1990s, none of these presented a desired solution to Shannon's challenge–the gap to Shannon's limit was still considerable. History has shown that the challenge of finding good codes has been much more difficult to do than said. According to Costello, Jr. et al. (1998), the best code designed up to the 1990s was the Forney, Jr.'s (1966) concatenation of $[255, 223, 33]_{2^8}$ RS and a rate 1/4 memory 14 convolutional codes decoded using the Big Viterbi Decoder (BVD)[18]. This arrangement with binary antipodal signalling achieves bit error probability of 10^{-5} at $E_b/N_0 = 0.9$ dB, which corresponds to a gap of 2.5 dB to the ultimate Shannon's limit, and albeit this significant gain, the code rate is less than 1/4 and the complexity of BVD is significantly large. A major breakthrough was achieved with the discovery of parallel-concatenated convolutional codes or turbo codes by Berrou et al. (1993). The turbo code was made up of two simple convolutional codes separated by a pseudorandom interleaver, giving an overall code rate of 1/3. Berrou et al. (1993) reported that, with rate 1/2 (achieved by means of puncturing) and block length of 65536 bits, a bit error probability of 10^{-5} was achieved at $E_b/N_0 = 0.7$ dB, a real coding gain of 8.9 dB over uncoded binary antipodal signalling. The outstanding performance of the turbo code, which has brought the gap to ultimate Shannon's limit to as close as 2.3 dB at 10^{-5} of bit error probability, is attributed to the way the code is decoded. The decoder for the turbo code consists of two optimum soft-decision decoders, which use the BCJR algorithm, for the constituent convolutional codes. Each decoder produces soft output, known as extrinsic information, which is fed to the other decoder and vice versa, in an iterative manner. A notable issue on turbo codes is on their performance which shows a sign of flooring at low error probability. This floor is attributed to the weakness of the iterative decoder and the codes themselves.

Since the discovery of Viterbi's (1967) algorithm, many developments have been devoted to convolutional codes. The feasibility of an optimum soft-decision decoder has created opportunities for convolutional codes to be used in many practical applications, namely space communications, broadcasting of digital audio and video, and also mobile communications such as the Global System for Mobile Communications (GSM) and the IS-95 standard of Code Division Multiple Access (Costello, Jr. et al.; 1998). The CCSDS concatenation scheme of RS and convolutional codes was also decoded using the Viterbi algorithm. Realising the benefit of soft input information, researchers began to associate soft output in the context of decoding. A notable work in this direction

[18]Collins, O. (1992), "The subtleties and intricacies of building a constraint length 15 convolutional decoder", *IEEE Transactions Communications*, 40(12), 1810–1819

was that of Hagenauer and Hoeher (1989) for what is known as the Soft-Output Viterbi Algorithm (SOVA). Subsequent work on this area was the use of soft output using the BCJR algorithm, see Lodge et al. (1992) and Lodge et al. (1993). The work on soft-input and soft-output decoding revealed the advantage of iterative decoding in a concatenated scheme and was a precursor to turbo codes.

The success of iterative decoding has brought Gallager's (1962) work back to life. The rediscovery of Gallager's (1962) low-density parity-check (LDPC) codes by MacKay and Neal (1996) marked another major milestone of channel coding in the 1990s. MacKay and Neal (1996) showed that, in addition to turbo codes, LDPC codes were also capacity-approaching codes in the AWGN channel. The decoder of LDPC codes, commonly known as the sum-product (Gallager; 1963) or equivalently belief propagation (Pearl; 1988) decoder, is a form of iterative decoder which can be viewed as a bank of BCJR decoders for $[n, n-1, 2]_2$ single parity-check codes. Each of the BCJR decoders produces soft-output information which is processed and then utilised in the next iteration.

LDPC codes provides an obvious example which shows the role of technology in the field of channel coding. Gallager invented LDPC codes in 1962, but his invention was forgotten for the next three decades because the complexity of decoding LDPC codes was deemed to be too high from a practical standpoint considering the level of technology during that period. While this was true from the application point of view, it did not stop Tanner (1981) studying the algebraic framework of LDPC codes. In addition, Tanner (1981) also linked the subject of LDPC codes with a branch of mathematics called graph theory, and consequently, the term *Tanner graph* now commonly appears in the literature on LDPC codes.

An important contribution to the construction of LDPC codes was made by Luby et al. (2001) who showed that by allowing some irregularity in the structure of the parity-check matrix of the LDPC codes, performance can be improved. Gallager's (1962) original LDPC codes are regular in the sense that, considering the parity-check matrix of the LDPC codes, the number of non zeros in each row are fixed and so are the number of non zeros in each column. It was than shown by Chung, Forney, Jr., Richardson and Urbanke (2001), using the method of Richardson et al. (2001)–*density evolution* to design irregular LDPC codes, a distance as small as 1.8 dB to the ultimate Shannon's limit was attainable at a block length 10^7 bits.

Due to their capacity approaching performance, both turbo codes and LDPC codes have dominated the research topics in modern channel coding. They have also attracted the attention of industry and we have seen, in recent years, the availability of various hardware encoders and decoders for these codes. There seems to be a tendency by industry to favour LDPC codes, instead of turbo codes. This may have been motivated by the fact that Berrou has patented his invention, while Gallager's patent on LDPC codes had expired before even their rediscovery. Thus, since both types of code have similar performance, it is financially more attractive to adopt LDPC codes than turbo codes. In addition, in terms of hardware implementation, LDPC codes are more attractive–the bank of BCJR decoders for the single parity-check codes may be parallelised and each BCJR decoder for a single parity-check code has relatively low complexity. Indeed, implementation complexity has driven recent research and development and also the deployment of LDPC codes. Despite having superior performance, the complexity of encoding irregular LDPC codes is considerable and LDPC codes that have some regular structure are preferred. Recent developments have been geared towards the construction of low encoding complexity LDPC codes. One of the attractive types

is quasi-cyclic LDPC codes for which the encoder may be implemented with linear feedback shift registers. Another attractive type of LDPC codes is the family of Repeat-Accumulate (RA) codes proposed by Divsalar et al. (1998). An irregular extension of RA codes, introduced by Jin et al. (2000), outclassed turbo codes and was selected as the chosen error-correcting codes in the second version of digital video broadcasting (DVB) standard. LDPC codes have also been adopted for the IEEE 802.3an (Ethernet) standard, and are being considered for inclusion in the IEEE 802.16e (WiMax) and 802.11n (WiFi) standards as well as various storage applications (Costello, Jr. and Forney, Jr.; 2007). Turbo codes have also been widely deployed; according to Costello, Jr. and Forney, Jr. (2007), they have been used in deep space communications, third generation mobile communications such as CDMA2000 and Universal Mobile Telecommunications System (UMTS), digital video broadcasting: both Return Channel over Satellite and Terrestrial, satellite communications such as Inmarsat and Eutelsat, as well as in WiMax.

Throughout the journey of the development of channel coding since 1948, we have seen that the line of development has been shifted from algebraic coding, which aims to maximise the asymptotic coding gain γ_c, to iterative decoding of large codes made up of simple constituent codes. The latter line of development puts the emphasis on linear codes of long block length which in general have relatively poor minimum Hamming distance. Classical or algebraic coding is still an active research subject to date and the search for codes that maximise γ_c is still active. The first effort to tabulate a survey of binary linear codes that are known to have maximum γ_c was attributed to Calabi and Myrvaagnes (1964). They gave a table of best known binary linear codes for $1 \leq k \leq n \leq 24$. This survey was later extended by Sloane (1972) to include both linear and non linear binary codes. Using the results of Calabi and Myrvaagnes (1964), those of Sloane (1972) for binary linear codes and contributions from researchers, Helgert and Stinaff (1973) presented an extended Calabi and Myrvaagnes's (1964) table to include all best known binary linear codes for $1 \leq k \leq n \leq 127$. Improvements to Helgert and Stinaff's (1973) table were given by Verhoeff (1987), which were subsequently updated in Brouwer and Verhoeff (1993). A major contribution to this survey was published in Brouwer (1998), which includes not only the best known binary linear codes, but also linear codes over $\mathbb{F}_q$ for $3 \leq q \leq 9$ except $q = 6$. Brouwer's (1998) tables contains codes over

1. $\mathbb{F}_2$ for $1 \leq k \leq n \leq 256$;
2. $\mathbb{F}_3$ for $1 \leq k \leq n \leq 243$;
3. $\mathbb{F}_4$ for $1 \leq k \leq n \leq 128$;
4. $\mathbb{F}_5$ for $1 \leq k \leq n \leq 100$;
5. $\mathbb{F}_7$ for $1 \leq k \leq 10, 1 \leq n \leq 50$;
6. $\mathbb{F}_8$ for $1 \leq k \leq 40, 1 \leq n \leq 85$; and
7. $\mathbb{F}_9$ for $1 \leq k \leq 20, 1 \leq n \leq 121$.

In order to maintain an up-to-date database to assist with the search for best known linear codes, Brouwer set up a website[19] from which one may determine the highest known d for an $[n, k]_q$ linear code. Brouwer's (1998) tables were discontinued on 12 March 2007 and the tables were superseded

[19] http://www.win.tue.nl/~aeb/voorlincod.html

by those of Grassl (2007). At the time of writing, Grassl's (2007) contains entries for $1 \leq k \leq n \leq n'$ where $n' = 256, 243, 256, 130, 100, 130$ and 130 for $q = 2, 3, 4, 5, 7, 8$ and 9 respectively.

As the technology became more powerful and coupled with the success of soft-decision decoding of convolutional codes, from the beginning of the 1990s, there has been some significant development in soft-decision decoding of linear block codes. While an optimum soft-decision decoder is only possible for very short, low rate or high rate codes, many suboptimum soft-decision decoding algorithm arose from these developments. The precursor to the suboptimum decoder was the work of Wagner[20], Chase (1972) and Dorsch (1974). Some of the notable developments from the 1990s are due to Han et al. (1993), Fossorier and Lin (1995), Gazelle and Snyders (1997) and Valembois and Fossorier (2004). While these algorithms are for binary codes, major contributions to soft-decision decoding of non binary codes such as the RS codes are attributed to Sudan (1997), Guruswami and Sudan (1999) and Koetter and Vardy (2003).

1.4 A Note on the Performance Bound of Binary Codes of Finite Block Length

Shannon (1948) defines the channel capacity C of a communication channel with input X and output Y as the maximum of the mutual information between X and Y, denoted by $I(X;Y)$,

$$C = \max_{p(x)} I(X;Y) \tag{1.13}$$

$$= \max_{p(x)} \sum_x \sum_y p(x)p(y|x) \log\left(\frac{p(y|x)}{p(y)}\right) \tag{1.14}$$

where $p(x)$ is the probability density function of x and $p(y|x)$ is the conditional probability density function of y given x. The mutual information between X and Y can also be written in terms of entropy functions (Shannon; 1948; Cover and Thomas; 1991). Correspondingly, (1.13) may be written as

$$C = \max_{p(x)}\{H(Y) - H(Y|X)\}, \tag{1.15}$$

where $H(Y)$ is the entropy of Y and $H(Y|X)$ is the conditional entropy of Y given X.

The most commonly used communication channel model is the bandlimited AWGN channel model, which is a continuous time model, whose input and output waveforms of the channel are related by (Cover and Thomas; 1991)

$$y(t) = (x(t) + n(t)) * c(t), \tag{1.16}$$

where $x(t)$, $n(t)$ and $y(t)$ are the input, Gaussian noise and output waveforms respectively, and $c(t)$ is the impulse response of the channel which is bandlimited to B Hz. In order to obtain the capacity of the continuous-time AWGN channel model, the equivalent discrete-time model is considered. This equivalent discrete-time model is obtained by sampling the waveforms $x(t)$, $n(t)$ and $y(t)$ at

[20] Wagner's algorithm is described in Silverman and Balser (1954)

Nyquist rate of $2B$ samples per second. Thus, (1.16) becomes

$$y_i = x_i + n_i \qquad i = 0, 1, \ldots, L-1 \tag{1.17}$$

and $L = 2BT$ if these samples are transmitted in T seconds. The noise samples n_i are independently, identically distributed Gaussian random variables with zero mean and variance $\sigma^2 = N_0/2$. For continuous random variables x_i and y_i, (1.14) and (1.15) may be respectively written as

$$C = \max_{p(x_i)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x_i) p(y_i|x_i) \log \left(\frac{p(y_i|x_i)}{p(y_i)} \right) dy_i dx_i \tag{1.18}$$

$$= \max_{p(x_i)} \{ h(y_i) - h(y_i|x_i) \}. \tag{1.19}$$

Note that for continuous random variable, differential entropy $h(\cdot)$ is used instead of entropy $H(\cdot)$. The maximisation of the channel capacity is attained when x_i is a Gaussian random variable with zero mean and variance $\sigma_{x_i}^2$ (Shannon; 1948; Cover and Thomas; 1991). As a result, the output y_i is also a Gaussian random variable with mean $\mu_{y_i} = \mu_{x_i} + 0 = 0$ and variance $\sigma_{y_i}^2 = \sigma_{x_i}^2 + \sigma^2$, i.e.

$$p(y_i) = \frac{1}{\sigma_{y_i}\sqrt{2\pi}} \exp\left(-\frac{(y_i - \mu_{y_i})^2}{2\sigma_{y_i}^2} \right) = \frac{1}{(\sigma_{x_i} + \sigma)\sqrt{2\pi}} \exp\left(-\frac{y_i^2}{2(\sigma_{x_i}^2 + \sigma^2)} \right). \tag{1.20}$$

Using (1.20) and the following identities of continuous random variable z of mean μ_z and variance σ_z^2

$$\int_{-\infty}^{\infty} p(z) dz = 1 \tag{1.21}$$

$$\int_{-\infty}^{\infty} (z - \mu_z)^2 p(z) dz = \sigma_z^2, \tag{1.22}$$

the differential entropy $h(y_i)$ is given by

$$\begin{aligned}
h(y_i) &= -\int_{-\infty}^{\infty} p(y_i) \log_e p(y_i) dy_i \\
&= -\int_{-\infty}^{\infty} p(y_i) \log_e \left[\frac{1}{\sigma_{y_i}\sqrt{2\pi}} \exp\left(-\frac{(y_i - \mu_{y_i})^2}{2\sigma_{y_i}^2} \right) \right] dy_i \\
&= -\int_{-\infty}^{\infty} p(y_i) \log_e \left[\frac{1}{\sigma_{y_i}\sqrt{2\pi}} \right] dy_i + \int_{-\infty}^{\infty} p(y_i) \log_e \left[\exp\left(\frac{(y_i - \mu_{y_i})^2}{2\sigma_{y_i}^2} \right) \right] dy_i \\
&= \log_e(\sigma_{y_i}\sqrt{2\pi}) \int_{-\infty}^{\infty} p(y_i) dy_i + \frac{1}{2\sigma_{y_i}^2} \int_{-\infty}^{\infty} (y_i - \mu_{y_i})^2 p(y_i) dy_i \\
&= \log_e(\sigma_{y_i}\sqrt{2\pi}) + \frac{1}{2} \\
&= \frac{1}{2} \log_e(\sigma_{y_i}\sqrt{2\pi})^2 + \frac{1}{2} \log_e e \\
&= \frac{1}{2} \log_e(2\pi e \sigma_{y_i}^2).
\end{aligned} \tag{1.23}$$

From Cover and Thomas (1991), the conditional differential entropy $h(y_i|x_i)$,

$$\begin{aligned} h(y_i|x_i) &= h(x_i + n_i|x_i) = h(n_i|x_i) \\ &= h(n_i) \end{aligned}$$

since the Gaussian random variable n_i of zero mean and variance σ^2 is independent of the random variable x_i. Following (1.23),

$$h(y_i|x_i) = h(n_i) = \frac{1}{2}\log_e(2\pi e\sigma^2) \tag{1.24}$$

and the channel capacity for continuous-time bandlimited AWGN channel is

$$\begin{aligned} C &= \frac{1}{2}\log_e(2\pi e\sigma_{y_i}^2) - \frac{1}{2}\log_e(2\pi e\sigma^2) \\ &= \frac{1}{2}\log_e\left[\frac{\sigma_{y_i}^2}{\sigma^2}\right] \end{aligned}$$

and after changing the basis from e to 2,

$$C = \frac{1}{2}\log_2\left(\frac{\sigma_{x_i}^2 + \sigma^2}{\sigma^2}\right). \tag{1.25}$$

Note that the average transmitted power has to be constrained, otherwise the channel capacity may be infinite (Cover and Thomas; 1991; Proakis; 2001), i.e.

$$S = \frac{1}{T}\sum_{i=0}^{L-1}\mathbb{E}[x_i^2] = \frac{L\sigma_{x_i}^2}{T} \tag{1.26}$$

and this means

$$\sigma_{x_i}^2 = \frac{ST}{L} = \frac{S}{2B}. \tag{1.27}$$

and using (1.27), (1.6) and (1.4), the channel capacity (1.25) can be written as

$$C = \frac{1}{2}\log_2\left(1 + \frac{S}{N}\right) \quad \text{bits/sample.} \tag{1.28}$$

Using Nyquist signalling, $2B$ samples per second can be transmitted without suffering from inter-symbol interference over a channel of bandwidth B Hz. Multiplying (1.28) by $2B$ samples per second, which effectively changes the unit from bits per sample to bits per second, the channel capacity formulation given in (1.3) is obtained. Note that both input and output of the channel are continuous random variables (infinite input and output alphabets), $x_i, y_i \in [-\infty, \infty)$ for $i = 0, 1, \ldots, L-1$. Therefore, (1.3) is also termed the *continuous-input continuous-output channel capacity* for AWGN channel.

For finite input alphabet size or discrete-input and continuous-output AWGN channel, a different formulation of channel capacity exists. Assuming q-ary discrete input $\mathfrak{X} = \{s_0, s_1, \ldots, s_{q-1}\}$ and

continuous output $y \in Y = [-\infty, \infty)$, the channel capacity for this model is given by (Proakis; 2001)

$$C = \max_{p(s_i)} \sum_{i=0}^{q-1} \int_{-\infty}^{\infty} p(s_i)p(y|s_i) \log_2 \left(\frac{p(y|s_i)}{\sum_{j=0}^{q-1} p(s_j)p(y|s_j)} \right) dy \tag{1.29}$$

where for AWGN channel, it is known that

$$p(y|s_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y - s_i)^2 / 2\sigma^2\right). \tag{1.30}$$

The maximum value of C in (1.29) is achieved by a uniform distribution of the input s_i, and thus

$$C = \frac{1}{q} \sum_{i=0}^{q-1} \int_{-\infty}^{\infty} p(y|s_i) \log_2 \left(\frac{p(y|s_i)}{\frac{1}{q}\sum_{j=0}^{q-1} p(y|s_j)} \right) dy. \tag{1.31}$$

For binary-input AWGN (BIAWGN) channel, where $\mathfrak{X} = \{\pm 1\}$, having continuous-output, the channel capacity given in (1.31) can be written as

$$\begin{aligned} C = & \frac{1}{2} \int_{-\infty}^{\infty} p(y|+1) \log_2 \left(\frac{p(y|+1)}{\frac{1}{2}p(y|+1) + \frac{1}{2}p(y|-1)} \right) dy + \\ & \frac{1}{2} \int_{-\infty}^{\infty} p(y|-1) \log_2 \left(\frac{p(y|-1)}{\frac{1}{2}p(y|+1) + \frac{1}{2}p(y|-1)} \right) dy \\ = & \frac{1}{\sqrt{8\pi}\sigma} \int_{-\infty}^{\infty} \exp\left(-(y-1)^2/2\sigma^2\right) \log_2 \left(\frac{2}{1 + \exp\left(-2y/\sigma^2\right)} \right) dy + \\ & \frac{1}{\sqrt{8\pi}\sigma} \int_{-\infty}^{\infty} \exp\left(-(y+1)^2/2\sigma^2\right) \log_2 \left(\frac{2}{1 + \exp\left(2y/\sigma^2\right)} \right) dy. \end{aligned} \tag{1.32}$$

There is no closed-form expression for (1.32) and it has to be evaluated numerically. The channel capacity formulations given in (1.12) and (1.32) give the minimum E_b/N_0 value required to achieve zero error probability. It is possible to obtain the minimum E_b/N_0 value to achieve a given non zero probability of error using the rate distortion theory, see Moon (2005). Figure 1.3 shows the plot of the channel capacity in terms of the minimum E_b/N_0 to achieve a given probability of error–denoted by bit-error rate (BER) for binary-input AWGN and continuous-input AWGN channels, for various code rates. It can be seen from the figure that infinite alphabet input has larger capacity than discrete input; and that the difference in the capacity vanishes as the code rate approaches zero.

While all the Shannon's channel capacity formulations given above assume infinite block length, from a practical point of view, it is important to know the limit for finite block length and non zero error probability. The lower-bound of this limit was derived by Shannon (1959) using the sphere packing argument. It gives the required E_b/N_0 value to achieve a given probability of frame error for a given code with parameters n and k. The exact evaluation of this lower-bound–commonly known as the *sphere packing lower bound*, was initially given by Dolinar et al. (1998) for $k \leq 50$, and later extended by Tomlinson et al. (2002) to include cases for $k \leq 1000$. Using the results of Lenth (1989) and Posten (1994) on the non central t-distribution, which is equivalent to spherical Gaussian distribution, Ahmed et al. (2007) described an exact evaluation of the sphere packing lower bound for any value of k.

The Shannon's (1959) sphere packing lower bound is for the transmission of codes with infinite

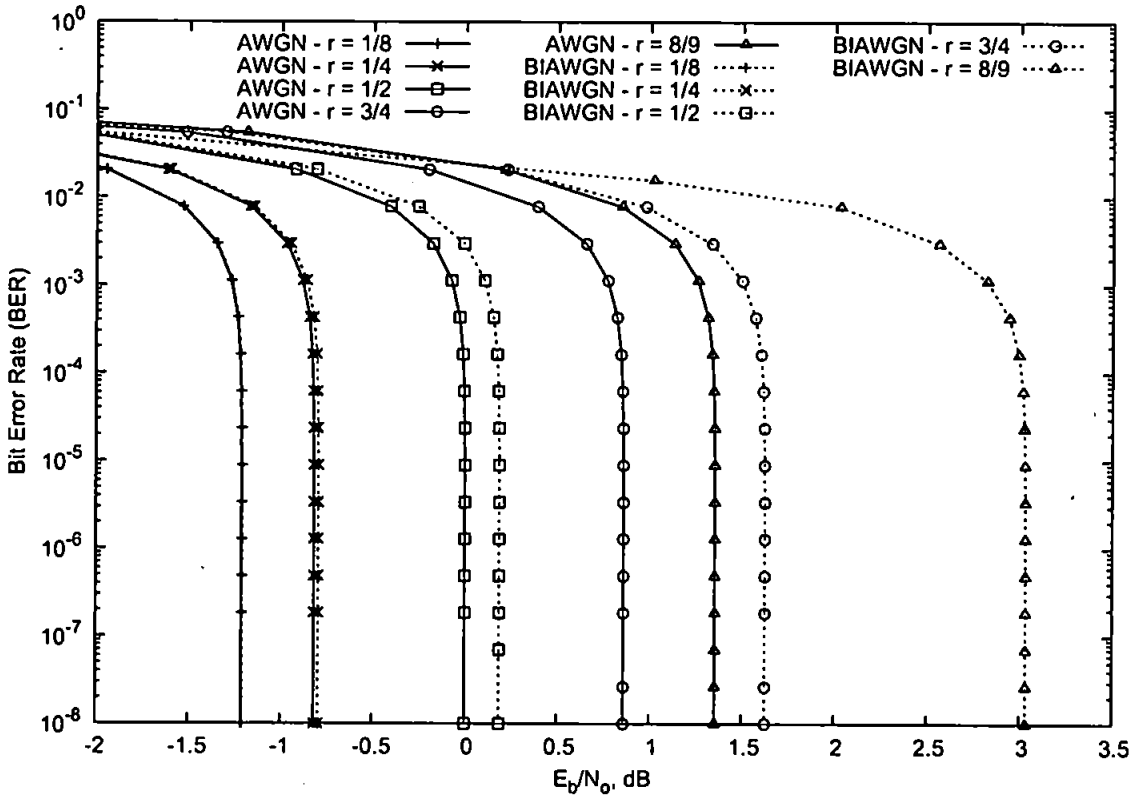


Figure 1.3: The minimum E_b/N_0 to achieve a given probability of bit error for continuous-input AWGN and binary-input AWGN channels
Note that, in the figure legend, BIAWGN refers to that of binary-input whereas AWGN refers to that of continuous-input.

alphabets over AWGN channel. Constraining the transmission to finite alphabets, e.g. binary transmission, inevitably introduces loss–rendering the sphere packing lower-bound relatively loose for finite alphabet transmissions. This loss, which is defined in terms of E_b/N_0 and is dependent on the code rate (larger loss for higher code rate), may be obtained using the result of Butman and McEliece (1974) as shown in Ahmed et al. (2007). Consequently, a tighter approximation for binary transmission may be derived by taking this loss into account.

For the remainder of this thesis, the term *offset sphere packing lower-bound* is used to refer to Shannon's (1959) sphere packing lower-bound constrained for binary transmission. This offset bound is used throughout this thesis as the Shannon's limit for binary transmission over AWGN channel, i.e. a benchmark of how good a code performs with binary transmission over AWGN channel.

1.5 Thesis Aim, Objectives and Organisation

The aim of this thesis is to study the state-of-the-art advancement of error correcting codes from the perspective of classical coding theory–which put the emphasis on code optimality; and probabilistic coding theory–which put the emphasis on ease-of-decoding. The objectives of this thesis are the development of code construction and decoding methods from the classical and probabilistic coding

perspectives; investigation and development of various capacity approaching measures; and the study of the mathematical aspects of classical error correcting codes in relation to their weight distributions.

This thesis contains three main parts. The first part (Chapters 2 and 3) considers the modern approach to channel coding–commonly referred to as probabilistic coding theory, we discuss LDPC codes–the construction methods and improvements to the iterative decoder. The second part (Chapters 4 through to 6) considers classical coding theory aspects; the search for good binary linear codes, some results on double-circulant codes and the soft-decision decoding of linear block codes. The final part of the thesis presents an application of coding to a communication system which has noiseless feedback. The detailed breakdown of this thesis is as follows.

Modern passage to approach the channel capacity is to utilise iterative decoding of long block length linear codes, such as turbo codes and LDPC codes, which are made of some simple component codes. Chapter 2 presents some novel algebraic techniques to construct good short block length LDPC codes. Techniques are presented based on the theory of idempotents, cyclotomic cosets and Mattson-Solomon polynomials in the code construction. The construction of irregular LDPC codes is considered, there is a numerical investigation on the effect of the irregularity of the codes and also the construction of quasi-cyclic codes using a protograph is described.

Chapter 3 describes methods to improve the convergence of the iterative decoder for LDPC codes. It essentially shows that, in the event that the standard iterative decoder which uses low weight parity-check equations is unsuccessful, the higher weight parity-check equations can be of some use. Investigation of the optimal Hartmann and Rudolph (1976) decoding algorithm is also given in this chapter.

One of the main problems in classical coding theory is to search for linear codes that maximise the minimum Hamming distance for a given code rate. To find a code which has the highest known minimum Hamming distance, one may consult to Brouwer's (1998) or Grassl's (2007) tables. Chapter 4 presents some new codes that have larger minimum Hamming distance than the corresponding ones in the tables. In this chapter, we also review the development of algorithms to compute the minimum Hamming distance of binary linear codes, and show how the problem of minimum Hamming distance computation may be solved by turning the minimum Hamming distance computation into a task suitable for parallel computing. In addition, this chapter also presents some results that serve as a continuation of the existing work on determining the largest minimum Hamming distance attainable by all binary cyclic error-correcting codes.

Chapter 5 considers an important class of half rate binary linear codes made up of two circulant matrices–double-circulant codes. Only prime-based double-circulant codes are considered and some interesting results on the weight distributions of these codes are presented. Some useful applications of these results are presented.

The loop in the classical coding part of this thesis is not closed without some studies on how to decode linear codes which are designed by classical coding theory–large minimum Hamming distance for a given code rate. In Chapter 6, there is a detailed discussion on the suboptimum soft-decision decoding algorithm due to Dorsch (1974). Some bounds related to the complexity of extending this algorithm to a achieve maximum likelihood solution are presented and so are the performance curves of some of the best binary linear codes.

The last part of this thesis, which contains Chapter 7, discusses an application of coding in a

type of communication systems commonly known as the incremental redundancy hybrid automatic repeat request (ARQ) system. It is shown how classical coding theory, which was also used in Chapter 4, may be used to construct a sequence of good linear codes for incremental redundancy communication systems. Some novel soft-decision error detection techniques, which can improve performance and throughput considerably compared to the known conventional schemes, are described and analysed in this chapter.

In Chapter 8, the work presented in this thesis is concluded and some directions for future research are given.

1.6 Contributions to Knowledge

The following list summarises the contributions made by the thesis.

Chapter 2

- An algebraic approach, using the theory of idempotents and cyclotomic cosets, to construct binary cyclic LDPC codes. Various conditions are derived to guarantee that the resulting cyclic LDPC codes have a girth of at least 6 and have an explicit minimum Hamming distance. This construction
  1. produces, in addition to the well-known Euclidean and projective geometry LDPC codes, some new binary cyclic codes suitable for iterative decoding;
  2. provides an incremental approach to the minimum Hamming distance and the sparseness of the parity-check matrix of the code.

  These binary cyclic codes, at short block length ($n < 350$), have performance close to the optimum performance of binary codes of the same block length and code rate.
- Algorithm 2.1 provides a procedure to search for all cyclic LDPC codes which have a girth of at least 6 for a given length.
- A different insight to constructing cyclic LDPC codes is given by running the construction in the Mattson-Solomon domain. This is analogous to time and frequency domain representations of a signal.
- Algorithm 2.2 provides a procedure, which works in the Mattson-Solomon domain, to search for all cyclic LDPC codes of a given length that can be iteratively decoded.
- Extending the construction of algebraic LDPC codes using the theory of cyclotomic cosets, idempotents and Mattson-Solomon polynomials to non binary fields.
- Some results and conclusions on the effect of low weight and high weight variable nodes in irregular LDPC codes, which are constructed using the Progressive-Edge-Growth (Hu et al.; 2005) algorithm, under iterative decoding. In addition, the effect of variable node degree ordering in constructing irregular LDPC using the Progressive-Edge-Growth algorithm is also discussed.

Chapter 3

- Results showing the effect of a suboptimum implementation–using a subset of all 2^{n-k} parity-check equations of an $[n, k, d]$ linear code, of the Hartmann and Rudolph (1976) decoding algorithm.

- A modified belief propagation iterative decoder–the codeword-substitution belief propagation decoder (Algorithm 3.1). For some cyclic LDPC codes of short block length, this modification produces near maximum likelihood performance.
- A method to improve the convergence of the iterative decoder by grouping the parity-check equations and the complexity implications.

Chapter 4

- A detailed description, including an example (Example 4.3) and a pseudo-code (Algorithm 4.4), of how the revolving door algorithm may used to split a lengthy single-threaded enumeration process to one that is suitable for parallel computation, in evaluating the minimum Hamming distance and the number of codewords of a given Hamming weight of a linear code.
- The highest minimum Hamming distance attainable by all binary cyclic codes of odd lengths from 129 to 189. This is an extension to the previous work of Chen (1970), Promhouse and Tavares (1978), and Schomaker and Wirtz (1992).
- A method to obtain an improved component code to be used in Construction X in the case where the dimension of the auxiliary component code is relatively small.
- New codes with larger minimum Hamming distance than the corresponding linear codes of the highest known minimum Hamming distance given in Brouwer's (1998). Overall there are 901 new codes.

Chapter 5

- A mathematical description on the choices of defining polynomial to construct pure double-circulant codes based on primes congruent to $\pm 3 \pmod 8$. A necessary condition to produce self-dual pure double-circulant codes is also derived.
- A detailed mathematical proof of the automorphism group of the bordered double-circulant codes based on prime congruent to $\pm 3 \pmod 8$ (quadratic double-circulant codes) in relation to the projective special linear group.
- A more efficient algorithm to compute the minimum Hamming distance and also the number of codewords of a given Hamming weight for self-dual double-circulant and formally self-dual quadratic double-circulant codes, both pure and bordered cases (Lemma 5.7).
- Detailed descriptions, including examples, on the approach to compute the modular congruence of the number of codewords of a given Hamming weight in a double-circulant code based on prime congruent to $\pm 3 \pmod 8$.
- A proof, which is derived using the modular congruence approach, of the incorrect results in Gaborit et al. (2005) on the weight distribution of the $[152, 76, 20]_2$ extended quadratic residue code, and the number of codewords of Hamming weights 30 and 32 in the $[138, 69, 22]_2$ extended quadratic residue code. Corrected results for these extended quadratic residue codes are given in this thesis.
- Weight distributions of the binary $[168, 84, 24]$ extended quadratic residue and quadratic double-circulant codes. The correctness of these distributions is verified with the modular congruence approach.

- A probabilistic approach to determine the minimum Hamming distance in extended quadratic residue and quadratic double-circulant codes.

Chapter 6

- Detailed description of the Dorsch's (1974) algorithm to decode binary linear codes, which uses the revolving door algorithm to efficiently enumerate codewords of low information weight.
- Derivation of a bound on the number of codewords required by Dorsch's (1974) algorithm to achieve maximum likelihood solution.

Chapter 7

- Application of Constructions X, Corollary 7.1, and XX, Corollary 7.2, to produce a sequence of linear codes suitable for incremental redundancy systems.
- A novel cyclic-redundancy-check-less soft-decision error detection method based on the confidence of the output of a list decoder of a linear code.
- A formulation and analysis of the error probability of this soft-decision error detection method.
- A scheme to reduce the probability of error of this soft-decision error detection technique by incorporating a cyclic-redundancy-check.

Part II

Probabilistic Coding

2 LDPC Code Constructions

Since their rediscovery in the 1990s, a great deal of effort has been devoted to the construction of good LDPC codes. This chapter outlines various existing techniques for LDPC code constructions, introduces some new algebraic construction techniques and discusses the author's experiments on irregular LDPC code constructions

Parts of this chapter are published in the following journal papers

1. Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2005), "Cyclotomic idempotent based binary cyclic codes", *Electronics Letters*, **41**(3), pp. 341–343
2. Horan, R., Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2006), "Idempotents, Mattson-Solomon polynomials and binary LDPC codes", *IEE Proceedings Communications*, **153**(2), pp. 256–262

and as well as in the following conference proceedings

1. Horan, R., Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2005), "A finite-field transform domain construction of binary low-density parity-check codes", in *Proceedings IEEE Information Theory Workshop on Coding and Complexity*, Rotorua, New Zealand, pp. 72–76
2. Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006), "GF(2^m) low-density parity-check codes derived from cyclotomic cosets", in *Proceedings 4th International Symposium on Turbo Codes in connection with 6th International ITG-Conference on Source and Channel Coding*, Munich, Germany.

2.1 Background and Notation

LDPC codes are linear block codes whose parity-check matrix–as the name implies, are sparse. These codes can be iteratively decoded using the sum product (Gallager; 1963) or equivalently the belief propagation (Pearl; 1988) soft decision decoder. It has been shown that (e.g. Chung, Forney, Jr., Richardson and Urbanke; 2001), for long block lengths, the performance of LDPC codes is close to the channel capacity. The theory of LDPC codes is related to a branch of mathematics called the graph theory. Some basic definition and notation on graph theory are briefly introduced as follows.

2.1 Definition (Vertex, Edge, Adjacent and Incident). A graph–denoted by $G(V, E)$, consists of an ordered set of vertices and edges.

- **(Vertex)** A vertex is commonly drawn as a node or a dot. The set $V(G)$ consists of vertices of $G(V, E)$ and if v is a vertex of $G(V, E)$, it is denoted as $v \in V(G)$. The number of vertices of $V(G)$ is denoted by $|V(G)|$.

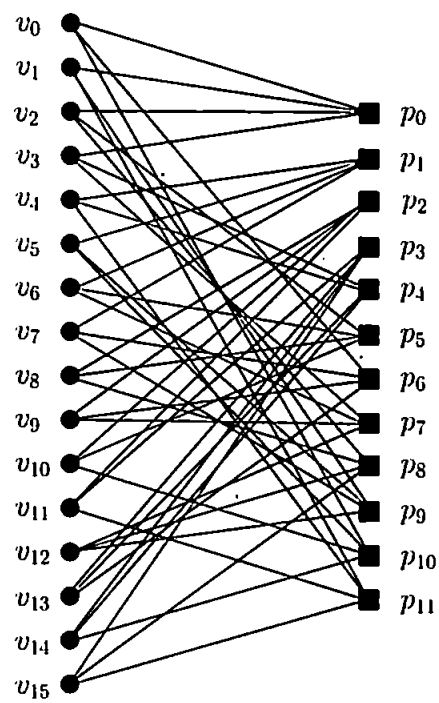
- **(Edge)** An edge (u, v) *connects* two vertices $u \in V(G)$ and $v \in V(G)$ and it is drawn as a line connecting vertices u and v. The set $E(G)$ contains pairs of elements of $V(G)$, i.e. $\{(u, v) \mid u, v \in V(G)\}$.
- **(Adjacent and Incident)** If $(u, v) \in E(G)$, then $u \in V(G)$ and $v \in V(G)$ are adjacent or neighbouring vertices of $G(V, E)$. Similarly, the vertices u and v are incident with the edge (u, v).

2.2 ***Definition (Degree).*** The degree of a vertex $v \in V(G)$ is the number of edges that are incident with vertex v, i.e. the number of edges that are connected to vertex v.

2.3 ***Definition (Bipartite or Tanner graph).*** Bipartite or Tanner graph $G(V, E)$ consists of two disjoint sets of vertices, say $V_v(G)$ and $V_p(G)$, such that $V(G) = V_v(G) \cup V_p(G)$ and every edge $(v_i, p_j) \in E(G)$ such that $v_i \in V_v(G)$ and $p_j \in V_p(G)$ for some integers i and j.

$$H = \begin{array}{c} \begin{array}{cccccccccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \end{array} \\ \left[\begin{array}{cccccccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \end{array}$$

(a) Parity-check matrix



(b) Tanner graph

Figure 2.1: Representations of a $[16, 4, 4]$ LDPC code

An $[n, k, d]$ LDPC code may be represented by a Tanner graph $G(V, E)$. The parity-check matrix H of the LDPC code consists of $|V_p(G)| = n - k$ rows and $|V_v(G)| = n$ columns. The set of vertices $V_v(G)$ and $V_p(G)$ are called *variable* and *parity-check* vertices respectively. Figure 2.1 shows the parity-check and the corresponding Tanner graph of a $[16, 4, 4]$ LDPC code. Let $V_v(G) = (v_0, v_1, \ldots, v_{n-1})$ and $V_p(G) = (p_0, p_1, \ldots, p_{n-k-1})$, it can be seen that for each $(v_i, p_j) \in E(G)$, the ith column and jth row of H, $H_{j,i} \neq 0$, for $0 \leq i \leq n-1$ and $0 \leq j \leq n-k-1$.

2.4 *Definition (Cycle).* A cycle in a graph $G(V, E)$ is a sequence of distinct vertices that starts and ends in the same vertex. For bipartite graph $G(V, E)$, exactly half of these distinct vertices belong to $V_v(G)$ and the remaining half belong to $V_p(G)$.

2.5 *Definition (Girth and Local Girth).* The girth of graph $G(V, E)$ is the length of the shortest cycle in the graph $G(V, E)$. The local girth of a vertex $v \in V(G)$ is the length of shortest cycle that passes through vertex v.

The performance of a typical iteratively decodable code (e.g. LDPC and turbo code) may be partitioned into three regions, namely erroneous, waterfall and error floor regions, see Figure 2.2. The erroneous region occurs at low E_b/N_0 values and is indicated by the inability of the iterative decoder to correctly decode almost all of the transmitted messages. As the signal power is increased, the error rate of the iterative decoder decreases rapidly–resembling a waterfall. The E_b/N_0 value at which the waterfall region starts is commonly known as the *convergence threshold* in the literature. At higher E_b/N_0 values, the error rate starts to flatten–introducing error floor in the frame error rate (FER) curve.

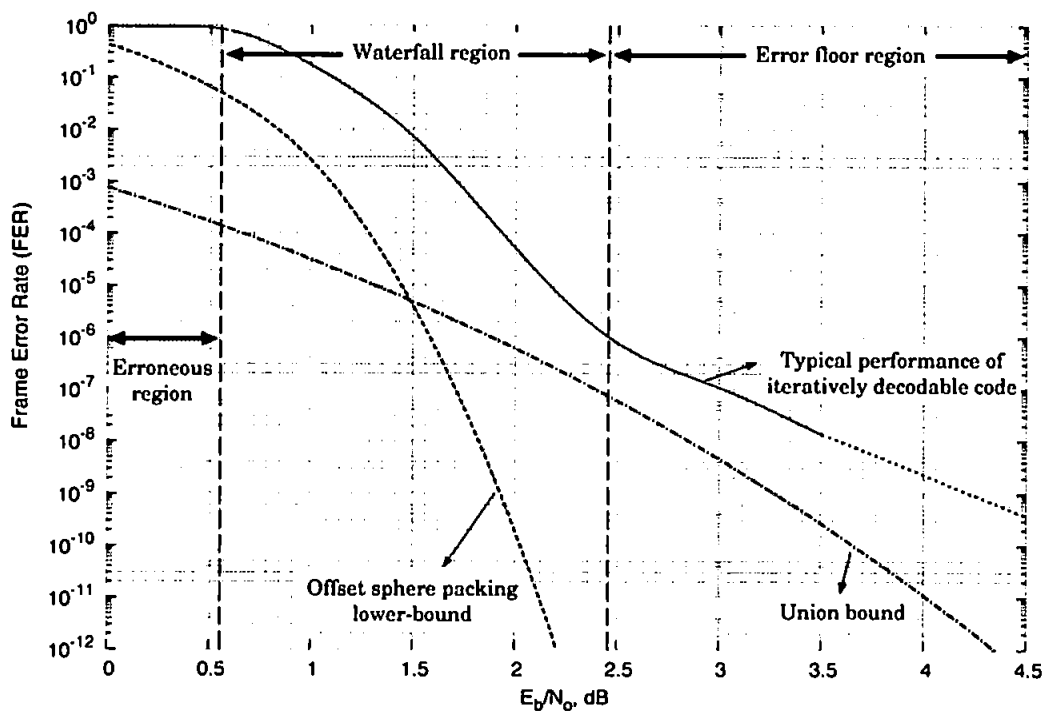


Figure 2.2: Waterfall and error regions on FER performance over AWGN channel

In addition to this FER curve, the offset sphere packing lower bound and the probability of error based on the union bound argument–see Proakis (2001), are also plotted in Figure 2.2. The sphere packing lower bound represents a region where the attainable performance of a coding system is; the performance to the left of this lower bound is not attainable, whereas that to the right may be achieved by some coding and decoding arrangements. The other curve is the union bound of the

probability of error, which is dominated by the low Hamming weight codewords and the number of codewords of these Hamming weights. The larger the minimum Hamming distance of a code, the lower the union bound typically. For iteratively decodable codes which are not designed to maximise the minimum Hamming distance, the union bound intersects with the offset sphere packing lower bound at relatively low E_b/N_0 value.

It may be seen that, with an ideal soft decision decoder, the performance of a coding system would follow the sphere packing lower bound and at higher E_b/N_0 values, the performance floors due to the limitation of the minimum Hamming weight codewords. However, as depicted in Figure 2.2, that there is a relatively wide gap between the union bound and the error floor of a typical iteratively decodable code. This is an inherent behaviour of iteratively decodable codes and it is attributed to the weakness of the iterative decoder. There are other error events, which are not caused by the minimum Hamming weight codewords, that prevent the iterative decoder to reach the union bound.

In terms of the construction technique, LDPC codes may be divided into two categories, namely random and algebraic LDPC codes. LDPC codes may also be classified as regular or irregular codes depending on the structure of the parity-check matrix, see Section 2.1.1 for the definition. Another attractive construction method that has been shown to offer capacity-achieving performance is non-binary construction.

2.1.1 Random Constructions

Gallager (1962) introduced the (n, λ, ρ) LDPC codes where n represents the block-length and the number of non zeros per column and the number of non zeros per row are represented by λ and ρ respectively*. The short notation (λ, ρ) is also commonly used to represent these LDPC codes. The code-rate of the Gallager (λ, ρ) codes is given by

$$R = 1 - \frac{\lambda}{\rho}.$$

An example of the parity-check matrix of the Gallager (λ, ρ) LDPC code is shown in Figure 2.1a. It is a $[16, 4, 4]$ code with λ of 3 and ρ of 4. The parity-check matrix of the (λ, ρ) Gallager codes always have a fixed number of non zeros per column and per row, and because of this property, this class of LDPC codes is termed regular LDPC codes. The performance of the Gallager LDPC codes in the waterfall region is not as satisfactory as that of turbo codes for the same block-length and code-rate. Many efforts have been devoted to improve the performance of the LDPC codes and one example that provides significant improvement is the introduction of the irregular LDPC codes by Luby et al. (2001). The irregular LDPC codes, as the name implies, does not have fixed number of non zeros per column or per row and thus, the level of error-protection varies over a codeword. The columns of a parity-check matrix that have higher number of non zeros provide stronger error protection than those that have less number of non zeros. Given an input block in iterative decoding, errors in the coordinates of this block, whose columns of parity-check matrix have larger number of non zeros, will be corrected earlier, i.e. only small number of iterations are required. In the subsequent

*Note that Gallager actually used the notations of j and k for the number of non zeros per column and per rows respectively. These notations, however, are changed here due to the ambiguity of the symbol k which is also used to denote the code dimension.

iterations, the corrected values in these coordinates will then be utilised to correct errors in the remaining coordinates of the block.

2.6 *Definition (Degree Sequences).* The polynomial $\Lambda_\lambda(x) = \sum_{i\geq 1} \lambda_i x^i$ is called the symbol or variable degree sequence, where λ_i is the fraction of vertices of degree i. Similarly, $\Lambda_\rho(x) = \sum_{i\geq 1} \rho_i x^i$ is the check degree sequence, where ρ_i is the fraction of vertices of degree i.

The degree sequences given in the above definition are usually known as *vertex-oriented degree sequences*. Another representation are *edge-oriented degree sequences* which consider the fraction of edges that are connected to a vertex of certain degree. Irregular LDPC codes are defined by these degree sequences and in this thesis, it is assumed that the degree sequences are vertex-oriented.

Degree sequences

Example 2.1: An irregular LDPC code with the following degree sequences:

$$\Lambda_\lambda(x) = 0.5x^2 + 0.26x^3 + 0.17x^5 + 0.07x^{10}$$
$$\Lambda_\rho(x) = 0.80x^{14} + 0.20x^{15}$$

has 50%, 26%, 17% and 7% of the columns with 2, 3, 5 and 10 ones per columns respectively and 80% and 20% of the rows with 14 and 15 ones per row respectively.

Various techniques have been proposed to design good degree distributions. Richardson et al. (2001) have used *density evolution* to determine the convergence threshold and to optimise the degree distributions. Chung, Richardson and Urbanke (2001) simplified the density evolution approach using *Gaussian approximation*. With the optimised degree distributions, Chung, Forney, Jr., Richardson and Urbanke (2001) showed that the bit error rate performance of a long block-length ($n = 10^7$) irregular LDPC code was within 0.04 dB away from the capacity limit for binary transmission over AWGN channel–see Wozencraft and Jacobs (1965) for the limit, which is equivalent to 0.18 dB to the ultimate Shannon's (1948) limit. The density evolution and Gaussian approximation methods, which make use of the concentration theorem (Richardson and Urbanke; 2001), can only be used to design the degree distributions for infinitely long LDPC codes. The concentration theorem states that the performance of cycle-free LDPC codes can be characterised by the average performance of the ensemble. The cycle-free assumption is only valid for infinitely long LDPC codes and cycles are inevitable for finite block-length LDPC codes. As can be expected, the performance of finite block-length LDPC codes with degree distributions derived based on the concentration theorem varies considerably from the ensemble performance. There are various techniques to design good finite block-length LDPC codes, for instance see Campello et al. (2001), Campello and Modha (2001), Hu et al. (2002) and Tian et al. (2004). In particular, the work of Hu et al. (2002) with the introduction of the Progressive-Edge-Growth (PEG) algorithm to construct both regular and irregular LDPC codes, that of Tian et al. (2004) with the introduction of *extrinsic message degree* and recently, that of Richter and Hof (2006) which improves the original PEG algorithm by introducing some construction constraints to avoid certain cycles involving variable vertices of degree 3, have provided significant contributions to the construction of practical LDPC codes as well as the lowering of the inherent error floor of these codes.

2.1.2 Algebraic Constructions

In general, LDPC codes constructed algebraically have a regular structure in their parity-check matrix. The algebraic LDPC codes offer many advantages over the randomly generated codes. Some of these advantages are

1. The important property such as the minimum Hamming distance can be easily determined or in the worst case, lower- and upper-bounds may be mathematically derived. These bounds are generally more accurate than the likes for random codes.

2. The minimum Hamming distance of algebraic LDPC codes is typically higher than that of the random counterparts. Due to the higher minimum Hamming distance, algebraic codes are not that likely to suffer from an early error floor.

3. The existence of a known structure in algebraic codes offers an attractive and simple encoding scheme. In the case of random codes, in order to perform encoding, a Gaussian-elimination process has to be carried out in the first place and the entire reduced echelon parity-check matrix has to be stored in the memory. On the other hand, algebraically constructed codes such as cyclic or quasi-cyclic codes can be completely described by polynomials. The encoding of cyclic or quasi-cyclic codes may be simply achieved using a linear-feedback shift-register circuit and the memory requirement is minimum. Various efficient techniques for encoding random LDPC codes have been proposed, see Ping et al. (1999) for example, but none of these techniques simplifies the storage requirements. The simplicity of the encoder and decoder structure has led to many algebraically constructed LDPC codes being adopted as industry standards (Costello, Jr. and Forney, Jr.; 2007).

4. Cyclic LDPC codes, have n low Hamming weight parity-check equations and therefore, compared to random codes, these cyclic LDPC codes has k extra equations for the iterative decoding to iterate with and this leads to improved performance.

One of the earliest algebraic LDPC code constructions was introduced by Margulis (1982) using the Ramanujan graphs. Lucas et al. (2000) showed that the well-known Different-Set Cyclic (DSC) (Weldon, Jr.; 1966) and One-Step Majority-Logic Decodable (OSMLD) (Lin and Costello, Jr.; 2004) codes have good performance under the iterative decoding. The iterative soft decision decoder offers significant improvement over the conventional majority-logic decoder. Another class of algebraic codes, is the class of the Euclidean and projective geometry codes which were discussed in detail by Kou et al. (2001). Other algebraic constructions include those that use combinatorial techniques (e.g. Johnson and Weller; 2001, 2002; Johnson; 2004; Vasic and Milenkovic; 2004).

It has been observed by the author that, in general, there is an inverse performance association between the minimum Hamming distance and the iterative decoding convergence. The irregular codes converge well with iterative decoding, but the minimum Hamming distance is relatively poor. On the other hand, the algebraically constructed LDPC codes, which have high minimum Hamming distance, tend not to converge well with iterative decoding. Consequently, compared to the performance of the irregular codes, the algebraic LDPC codes may perform worse in the low SNR region and become better in the high SNR region and this is be attributed to the early error floor of the irregular codes. As will be shown later, for short block-lengths ($n < 350$), cyclic algebraic LDPC codes offer the best performance.

2.1.3 Non-Binary Constructions

LDPC codes may be easily extended so that the symbols take values from finite-field $\mathbb{F}_{2^m}$ and Davey and MacKay (1998) were the pioneers in this work. Given an LDPC code over $\mathbb{F}_2$ with parity-check matrix H, LDPC code over $\mathbb{F}_{2^m}$ for $m \geq 2$ may be constructed by simply replacing every non zero element of H with any non zero element of $\mathbb{F}_{2^m}$ in a random or structured manner. Davey and MacKay (1998) and Hu et al. (2005) have shown that the performance of LDPC codes can be improved by going beyond the binary field. The non binary LDPC codes have better convergence behaviour under iterative decoding. Using some irregular non binary LDPC codes, whose parity-check matrices are derived by randomly replacing the non zeros of the PEG-constructed irregular binary LDPC codes, Hu et al. (2005) demonstrated that a coding gain of 0.25 dB was achieved. It may be regarded that the improved performance is attributable to the improved graph structure in the non binary arrangement. Consider a cycle of length 6 in the Tanner graph of a binary LDPC code, which is represented as the following sequence of pairs of edges $\{(v_0, p_0), (v_3, p_0), (v_3, p_2), (v_4, p_2), (v_4, p_1), (v_0, p_1)\}$. If the corresponding entries in the parity-check matrix are replaced with some non zeros over $\mathbb{F}_{2^m}$ for $m \geq 2$, provided that these six entries are not all the same, the cycle length becomes larger than 6. According to McEliece et al. (1998) and Etzion et al. (1999), the non convergence of iterative decoder are caused by existence of cycles in the Tanner graph representation of the code considered. Cycles–especially those of short lengths, introduce correlations of reliability information exchanged in iterative decoding. Since cycles are inevitable for finite block length codes, it is desirable to have LDPC codes with large girth.

The non-binary LDPC codes also offer an attractive matching for higher-order modulation. The impact of increased complexity of the symbol-based iterative decoder can be moderated as the reliability information from the component codes may be efficiently evaluated using the frequency domain dual codes decoder based on the Fast-Walsh-Hadamard transform (Davey and MacKay; 1998).

2.2 Algebraic LDPC Codes

2.2.1 Binary Cyclic LDPC Codes Derived from Cyclotomic Cosets

Based on the pioneering works of MacWilliams and Sloane (1977) on the Mattson-Solomon polynomials, idempotents and cyclotomic cosets, a class of cyclic codes that is suitable for iterative decoding is constructed. This class of cyclic codes falls into the class of OSMLD codes whose parity-check polynomial is orthogonal on each position–implying the absence of girth of 4 in the underlying Tanner graph, and the corresponding parity-check matrix is sparse, and thus can be used as LDPC codes.

Let the splitting field of $x^n - 1$ over $\mathbb{F}_2$ be $\mathbb{F}_{2^m}$ where n is an odd integer and $m > 1$, and let a generator of $\mathbb{F}_{2^m}$ be α and an integer $r = (2^m - 1)/n$. Let $T_m(x)$ be a set of polynomials of degree at most $n - 1$ with coefficients over $\mathbb{F}_{2^m}$ and for convenience, let $T_1(x)$ be denoted by $T(x)$.

2.7 ***Definition (Mattson-Solomon Polynomial).*** If $a(x) \in T(x)$, the Mattson-Solomon polynomial

of $a(x)$ is the mapping from $a(x)$ to $A(z)$ and is defined by (MacWilliams and Sloane; 1977)

$$A(z) = \mathrm{MS}(a(x)) = \sum_{j=0}^{n-1} a(\alpha^{-rj}) z^j, \tag{2.1}$$

where $A(z) \in T_m(z)$. The inverse Mattson-Solomon polynomials is:

$$a(x) = \mathrm{MS}^{-1}(A(z)) = \frac{1}{n} \sum_{i=0}^{n-1} A(\alpha^{ri}) x^i. \tag{2.2}$$

The polynomial variables x and z are used to distinguish the polynomials in the domain and codomain of the mapping. Assume that $\circ$ be polynomial multiplications $\pmod{x^n - 1}$ and $\cdot$ be the dot product of polynomials, i.e. $(\sum a_i x^i) \cdot (\sum b_i x^i) = \sum a_i b_i x^i$. It follows from the mapping that $\circ$ in the domain is mapped to $\cdot$ in the codomain and vice-versa. This concept is analogous to multiplication and convolution in time and frequency domain, where the Fourier and inverse Fourier transforms are the Mattson-Solomon and inverse Mattson-Solomon polynomials respectively.

2.8 ***Definition (Binary Idempotent).*** Consider $e(x) \in T(x)$, $e(x)$ is an idempotent if the property of $e(x) = e^2(x) = e(x^2) \bmod (x^n - 1)$ is satisfied.

An $[n, k]$ binary cyclic code may be described by the generator polynomial $g(x) \in T(x)$ of degree $n - k$ and the parity-check polynomial $h(x) \in T(x)$ of degree k such that $g(x)h(x) = x^n - 1$. According to MacWilliams and Sloane (1977), as an alternative to $g(x)$, idempotent may also be used to generate cyclic codes. Any binary cyclic code can be described by a unique idempotent $e_g(x) \in T(x)$ which consists of a sum of primitive idempotents. The unique idempotent $e_g(x)$ is known as the *generating idempotent* and as the name implies, $g(x)$ is a divisor of $e_g(x)$, and to be more specific $e_g(x) = m(x)g(x)$ where $m(x) \in T(x)$ contains the repeated factors or non-factors of $x^n - 1$.

2.1 ***Lemma.*** If $e(x) \in T(x)$ is an idempotent, $E(z) = \mathrm{MS}(e(x)) \in T(z)$.

Proof ([cf. MacWilliams and Sloane (1977)].). Since $e(x) = e(x)^2 \pmod{x^n - 1}$, from (2.1) it follows that $e(\alpha^{-rj}) = e(\alpha^{-rj})^2$ for $j = \{0, 1, \ldots, n-1\}$ and some integer r. Clearly $e(\alpha^{-rj}) \in \{0, 1\}$ implying that $E(z)$ is a binary polynomial. ∎

2.9 ***Definition (Cyclotomic Coset).*** Let s be a positive integer, the 2−cyclotomic coset of $s \pmod n$ is given by

$$C_s = \{2^i s \pmod n \mid 0 \le i \le t\},$$

where it shall always be assumed that the subscript s is the smallest element in the set C_s and t is the smallest positive integer such that $2^{t+1}s \equiv s \pmod n$.

For convenience, the term cyclotomic coset shall be used when referring to the 2−cyclotomic coset of an integer for the remainder of this thesis. If $\mathcal{N}$ is the set consisting of the smallest elements of

all possible cyclotomic cosets then it follows that

$$C = \bigcup_{s \in \mathcal{N}} C_s = \{0, 1, 2, \ldots, n-1\}.$$

2.10 Definition (Binary Cyclotomic Idempotent). Let the polynomial $e_s(x) \in T(x)$ be given by

$$e_s(x) = \sum_{0 \le i \le |C_s|-1} x^{C_{s,i}}, \tag{2.3}$$

where $|C_s|$ is the number of elements in C_s and $C_{s,i} = 2^i s \pmod{n}$, the $(i+1)$th element of C_s. The polynomial $e_s(x)$ is called a binary cyclotomic idempotent.

Cyclotomic cosets of 63

Example 2.2: The entire cyclotomic cosets of 63 and their corresponding binary cyclotomic idempotents are as follows

$$\begin{aligned}
C_0 &= \{0\} & e_0(x) &= 1 \\
C_1 &= \{1,2,4,8,16,32\} & e_1(x) &= x + x^2 + x^4 + x^8 + x^{16} + x^{32} \\
C_3 &= \{3,6,12,24,48,33\} & e_3(x) &= x^3 + x^6 + x^{12} + x^{24} + x^{33} + x^{48} \\
C_5 &= \{5,10,20,40,17,34\} & e_5(x) &= x^5 + x^{10} + x^{17} + x^{20} + x^{34} + x^{40} \\
C_7 &= \{7,14,28,56,49,35\} & e_7(x) &= x^7 + x^{14} + x^{28} + x^{35} + x^{49} + x^{56} \\
C_9 &= \{9,18,36\} & e_9(x) &= x^9 + x^{18} + x^{36} \\
C_{11} &= \{11,22,44,25,50,37\} & e_{11}(x) &= x^{11} + x^{22} + x^{25} + x^{37} + x^{44} + x^{50} \\
C_{13} &= \{13,26,52,41,19,38\} & e_{13}(x) &= x^{13} + x^{19} + x^{26} + x^{38} + x^{41} + x^{52} \\
C_{15} &= \{15,30,60,57,51,39\} & e_{15}(x) &= x^{15} + x^{30} + x^{39} + x^{51} + x^{57} + x^{60} \\
C_{21} &= \{21,42\} & e_{21}(x) &= x^{21} + x^{42} \\
\\
C_{23} &= \{23,46,29,58,53,43\} & e_{23}(x) &= x^{23} + x^{29} + x^{43} + x^{46} + x^{53} + x^{58} \\
C_{27} &= \{27,54,45\} & e_{27}(x) &= x^{27} + x^{45} + x^{54} \\
C_{31} &= \{31,62,61,59,55,47\} & e_{31}(x) &= x^{31} + x^{47} + x^{55} + x^{59} + x^{61} + x^{62}
\end{aligned}$$

and $\mathcal{N} = \{0,1,3,5,7,9,11,13,15,21,23,27,31\}$.

2.11 Definition (Binary Parity-Check Idempotent). Let $\mathcal{M} \subseteq \mathcal{N}$ and let the polynomial $u(x) \in T(x)$ be defined by

$$u(x) = \sum_{s \in \mathcal{M}} e_s(x), \tag{2.4}$$

where $e_s(x)$ is an idempotent. The polynomial $u(x)$ is called a binary parity-check idempotent.

The binary parity-check idempotent $u(x)$ can be used to describe an $[n, k]$ cyclic code. Since $(u(x), x^n - 1) = h(x)$, the polynomial $\bar{u}(x) = x^{\deg(u(x))}u(x^{-1})$ and its n cyclic shifts $\pmod{x^n - 1}$ can be used to define the parity-check matrix of a binary cyclic code. In general, $\mathrm{wt}_H(\bar{u}(x))$ is much lower than $\mathrm{wt}_H(h(x))$, therefore a low-density parity-check matrix can be derived from $\bar{u}(x)$.

Let the parity-check polynomial $\bar{u}(x) = x^{\bar{u}_0} + x^{\bar{u}_1} + \ldots + x^{\bar{u}_t}$ of weight $t+1$. Since the code defined by $\bar{u}(x)$ is cyclic, for each non zero coefficient $\bar{u}_i$ in $\bar{u}(x)$, there are other t parity-check polynomials of weight $t+1$, that also have non zero at position $\bar{u}_i$. Furthermore, consider the set of these $t+1$ polynomials that have non zero at position $\bar{u}_i$, there is no more than one polynomial in the set that have non zero at position $\bar{u}_j$ for some integer j. In other words, if we count the number of times the positions $0, 1, \ldots, n-1$ appear in the exponent of the aforementioned set of $t+1$ polynomials, we shall find that all positions except $\bar{u}_i$ appear at most once. These set of $t+1$ polynomials are said to be *orthogonal* on position $\bar{u}_i$. The mathematical expression of this orthogonality is given in the following definition and lemma.

2.12 *Definition (Difference Enumerator Polynomial).* Let the polynomial $f(x) \in T(x)$. The difference enumerator of $f(x)$, denoted as $\mathcal{D}(f(x))$, is defined as

$$\mathcal{D}(f(x)) = f(x)\, f\left(x^{-1}\right) = d_0 + d_1 x + \ldots + d_{n-1}x^{n-1}, \tag{2.5}$$

where it is assumed that $\mathcal{D}(f(x))$ is a modulo $x^n - 1$ polynomial with coefficients taking values from $\mathbb{R}$ (real coefficients).

2.2 *Lemma.* Let d_i for $0 \leq i \leq n-1$ denote the coefficients of $\mathcal{D}(\bar{u}(x))$. If $d_i \in \{0, 1\}$, for all $i \in \{1, 2, \ldots, n-1\}$, the parity-check polynomial derived from $\bar{u}(x)$ is orthogonal on each position in the n-tuple. Consequently

i) the minimum distance of the resulting LDPC code is $1 + \mathrm{wt}_H(\bar{u}(x))$, and

ii) the underlying Tanner Graph has girth of at least 6.

Proof. (i) (Peterson and Weldon, Jr.; 1972, Theorem 10.1) Let a codeword $c(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1}$ and $c(x) \in T(x)$. For each non zero bit position c_j of $c(x)$, where $j \in \{0, 1, \ldots, n-1\}$, there are $\mathrm{wt}_H(u(x))$ parity-check equations orthogonal to position c_j. Each of the parity-check equation must check another non zero bit c_l, where $l \neq j$, so that the equation is satisfied. Clearly, $\mathrm{wt}_H(c(x))$ must equal to $1 + \mathrm{wt}_H(u(x))$ and this is the minimum weight of all codewords. (ii) The direct consequence of having orthogonal parity-check equation is the absence of cycles of length 4 in the Tanner Graphs. Let a, b and c, where $a < b < c$, be three distinct coordinates in an n-tuple, since $d_i \in \{0, 1\}$ for $1 \leq i \leq n-1$, this implies that $b - a \neq c - b$. It is known that $q(b-a) \pmod{n} \in \{1, 2, \ldots, n-1\}$ and thus, $q(b-a) \pmod{n} \equiv (c-b)$ for some integer $q \in \{1, 2, \ldots, n-1\}$. If the integers a, b and c are associated with some variable vertices in the Tanner graph, a length 6 cycle is formed. □

It can be deduced that the cyclic LDPC code with parity-check polynomial $\bar{u}(x)$ is an OSMLD code if $d_i \in \{0, 1\}$, for all $i \in \{1, 2, \ldots, n-1\}$ or a DSC code if $d_i = 1$, for all $i \in \{1, 2, \ldots, n-1\}$, where d_i is the coefficient of $\mathcal{D}(\bar{u}(x))$.

In order to arrive at either OSMLD or DSC codes, the following design conditions are imposed on $\bar{u}(x)$ and therefore, $u(x)$:

Condition 2.1. The idempotent $u(x)$ must be chosen such that $\mathrm{wt}_H(u(x))\,(\mathrm{wt}_H(u(x)) - 1)) \leq n - 1$.

Proof. There are $\mathrm{wt}_H(u(x))$ polynomials of weight $\mathrm{wt}_H(u(x))$ that are orthogonal on position j for some integer j. The number of distinct positions in this set of polynomials is $\mathrm{wt}_H(u(x))\,(\mathrm{wt}_H(u(x)) - 1)$ and this number must be less than or equal to the total number of distinct integers between 1 and $n - 1$. □

Condition 2.2. Following Definition 2.12, let $W = \{i \mid d_i = 1,\ 1 \leq i \leq n - 1\}$, the cardinality of W must be equal to $\mathrm{wt}_H(u(x))\,(\mathrm{wt}_H(u(x)) - 1)$.

Proof. The cyclic differences between the exponents of polynomial $u(x)$ are given by $\mathcal{D}(u(x)) = \sum_{i=0}^{n-1} d_i x^i$, where the coefficient d_i is the number of differences and the exponent i is the difference. The polynomial $u(x)$ and some of its cyclic shifts are orthogonal on position 0 and this means that all of the cyclic differences between the exponents of $u(x)$ (excluding zero) must be distinct, i.e. $d_i \in \{0, 1\}$ for $1 \leq i \neq n - 1$. Since the weight of $u(x)$ excluding x^0 is $\mathrm{wt}_H\,(u(x)) - 1$ and there are $\mathrm{wt}_H\,(u(x))$ cyclic shifts of $u(x)$ that are orthogonal to x^0, the number of distinct exponents in the cyclic differences is $\mathrm{wt}_H\,(u(x))\,(\mathrm{wt}_H\,(u(x)) - 1) = W$. □

Condition 2.3. The exponents of $u(x)$ must not contain a common factor of n, otherwise a degenerate code–a repetition of a shorter cyclic code, is generated.

Proof. If the exponents of $u(x)$ contain a common factor of n, p with $n = pr$, then factors of $u(x)$ divide $x^r - 1$ and form a cyclic code of length r. Every codeword of the longer code is a repetition of the shorter cyclic code.

Condition 2.4. Following (2.4), unless $\mathrm{wt}_H(e_s(x)) = 2$, the binary parity-check idempotent $e_s(x)$ must not be self-reciprocal, i.e. $e_s(x) \neq e_s\left(x^{-1}\right)$.

Proof. The number of non zero coefficients of $\mathcal{D}(e_s(x))$ is equal to $\mathrm{wt}_H(e_s(x))\,(\mathrm{wt}_H(e_s(x)) - 1)$. For a self-reciprocal case, $e_s(x)e_s\left(x^{-1}\right) = e_s^2(x) = e_s(x)$ with $\mathrm{wt}_H(e_s(x))$ non zero coefficients. Following Condition 2.1, the inequality $\mathrm{wt}_H(e_s(x))\,(\mathrm{wt}_H(e_s(x) - 1) \leq \mathrm{wt}_H(e_s(x))$ becomes equality if and only if $\mathrm{wt}_H(e_s(x)) = 2$. □

Condition 2.5. Following (2.4), $u(x)$ must not contain $e_s\left(x^{-1}\right)$, unless $e_s(x)$ is self-reciprocal.

Proof. If $u(x)$ contains $e_s\left(x^{-1}\right)$, then $\mathcal{D}(u(x))$ will contain both $e_s(x)e_s\left(x^{-1}\right)$ and $e_s\left(x^{-1}\right)e_s(x)$, hence some of the coefficients of $\mathcal{D}(e_s(x))$, $d_i \neq \{0, 1\}$ for some integer i. □

Although the above conditions seem overly restrictive, they turn out to be helpful in code construction. Codes may be designed in stage-by-stage by adding candidate idempotents to $u(x)$ checking the above conditions at each stage.

In order to encode the cyclic LDPC codes constructed, there is no need to determine $g(x)$. With α defined as a primitive nth root of unity, it follows from Lemma 2.1 that $u(\alpha^i) \in \{0, 1\}$ for $0 \leq i \leq n-1$. Let $\mathcal{J} = (j_0, j_1, \ldots, j_{n-k-1})$ be a set of integer between 0 and $n - 1$ such that $g(\alpha^j) = 0$, for all $j \in \mathcal{J}$.

Because $u(x)$ does not contain α^j as its roots, it follows that $u(\alpha^j) = 1$, for all $j \in \mathcal{J}$. In $\mathbb{F}_2$, $1 + u(\alpha^j) = 0$ and the polynomial $1 + u(x) = e_g(x)$, the generating idempotent of the code, may be used to generate the codewords as an alternative to $g(x)$.

The number of information symbols of the cyclic LDPC codes can be determined either from the number of roots of $u(x)$ which are also roots of unity, i.e. $n - \mathrm{wt}_H\,(U(z))$, or from the degree of $(u(x), x^n - 1) = h(x)$.

Construction of [63,37,9] cyclotomic idempotent code

Example 2.3: Consider the design of cyclic LDPC code of length 63. The cyclotomic cosets modulo 63 is given in Example 2.2. Let $u(x)$ be defined by C_9, i.e. $u(x) = e_9(x) = x^9(1 + x^9 + x^{27})$. $\mathcal{D}(\bar{u}(x))$ indicates that the parity-check matrix defined by $\bar{u}(x)$ has no cycles of length 4, however, following Condition 2.3, it is a degenerate code consisting of repetition of codewords of length 7.

With $u(x) = e_{23}(x) = x^{23}(1 + x^6 + x^{20} + x^{23} + x^{30} + x^{35})$, the resulting cyclic code is a $[63, 31, 6]$ LDPC code which is non degenerate and its underlying Tanner graph has girth of 6. This code can be further improved by adding $e_{21}(x)$ to $u(x)$. Despite $e_{21}(x)$ is self-reciprocal, its weight is 2 satisfying Condition 2.4. Now, $u(x) = x^{21}(1 + x^2 + x^8 + x^{21} + x^{22} + x^{25} + x^{32} + x^{37})$ and it is a $[63, 37, 9]$ cyclic LDPC code.

Based on the theory described above, an algorithm which exhaustively searches for all non degenerate cyclic LDPC codes of length n which have orthogonal parity-check polynomial, has been developed and it is given in Algorithm 2.1.

Algorithm 2.1 CodeSearch(n, V, *index*)

Input:

$n \Leftarrow$ block length (odd integer)
index $\Leftarrow$ an integer that is initialised to -1
V $\Leftarrow$ a vector that is initialised to $\emptyset$
$S \Leftarrow \mathcal{N}$ excluding 0

Output:

CodesList contains set of cyclic codes which have orthogonal parity-check polynomial

```
T ⇐ V
for (i=index+1; i ≤ |S|; i++) do
  T_prev ⇐ T
  if (Σ_{∀t∈T} |C_{S_t}| ≤ √n, S_t is the t^th element of S) then
    Append i to T
    u(x) = Σ_{∀t∈T} e_{S_t}(x)
    if (u(x) is non degenerate) and (u(x) is orthogonal on each position (Lemma 2.2)) then
      U(z) = MS(u(x))
      k = n − wt_H(U(z))
      C ⇐ an [n, k, 1 + wt_H(u(x))] cyclic code defined by u(x)
      if (k ≥ 1/4) and (C ∉ CodeList) then
        Add C to CodeList
      end if
    end if
    CodeSearch(T, i)
  end if
  T ⇐ T_prev
end for
```

Table 2.1: Examples of 2−cyclotomic coset-based LDPC codes

$[n, k, d]$	Cyclotomic cosets
$[21, 11, 6]$	C_7, C_9
$[63, 37, 9]$	C_{21}, C_{23}
$[93, 47, 8]$	C_3, C_{31}
$[73, 45, 10]$	C_1
$[105, 53, 8]$	C_7, C_{15}
$[219, 101, 12]$	C_3, C_{73}
$[255, 135, 13]$	C_1, C_{119}
$[255, 175, 17]$	C_1, C_{27}
$[273, 191, 18]$	C_1, C_{91}, C_{117}
$[341, 205, 16]$	C_1, C_{55}
$[511, 199, 19]$	C_5, C_{37}
$[511, 259, 13]$	C_1, C_{219}
$[819, 435, 13]$	C_1
$[819, 447, 19]$	C_1, C_{351}
$[1023, 661, 23]$	C_1, C_{53}, C_{341}
$[1023, 781, 33]$	$C_1, C_{53}, C_{123}, C_{341}$

$[n, k, d]$	Cyclotomic cosets
$[1057, 813, 34]$	C_5, C_{43}, C_{151}
$[1387, 783, 28]$	C_1, C_{247}
$[1971, 1105, 21]$	C_1, C_{657}
$[2047, 1167, 23]$	C_1, C_{27}
$[2325, 1335, 28]$	C_1, C_{57}, C_{775}
$[2325, 1373, 30]$	C_1, C_{525}, C_{1085}
$[2359, 1347, 22]$	C_1
$[3741, 2229, 29]$	C_1
$[3813, 2087, 28]$	C_1, C_{369}, C_{1271}
$[4095, 2767, 49]$	$C_1, C_{41}, C_{235}, C_{733}$
$[4095, 3367, 65]$	$C_1, C_{41}, C_{235}, C_{273}, C_{411}, C_{733}$
$[4161, 2827, 39]$	C_1, C_{307}, C_{1387}
$[4161, 3431, 66]$	$C_1, C_{285}, C_{307}, C_{357}, C_{1387}$
$[4681, 2681, 31]$	C_1, C_{51}
$[5461, 3781, 43]$	C_1, C_{77}, C_{579}

Table 2.1 lists some example of codes obtained from Algorithm 2.1. Note that, all codes with code rate less than 0.25 are excluded from the table and codes of longer lengths may also be constructed. It can also be seen that some of the codes in Table 2.1 have the same parameters as the Euclidean and projective geometry codes, which have been shown by Kou et al. (2001) to perform well under iterative decoding.

A key feature of the cyclotomic coset-based construction is the ability to increment the minimum Hamming distance of a code by adding further weight from other idempotents and so steadily decrease the sparseness of the resulting parity-check matrix. Despite the construction method has a feature of producing LDPC codes with no cycles of length 4, it is important to remark that codes that have cycle of length 4 in their parity-check matrix do not necessarily have bad performance under iterative decoding and a similar finding has been demonstrated by Tang et al. (2005). It has been observed that there are many cyclotomic coset-based LDPC codes that have this property and the constraints in Algorithm 2.1 can be easily relaxed to allow the construction of cyclic LDPC codes with girth 4.

2.2.2 Mattson-Solomon Domain Construction of Binary Cyclic LDPC Codes

The $[n, k, d]$ cyclic LDPC codes presented in Section 2.2.1 are constructed by using the sum of idempotents, which are derived from the cyclotomic cosets modulo n, to define the parity-check matrix. A different insight on this construction technique may be obtained by working in the Mattson-Solomon domain.

Let n be a positive odd integer, $\mathbb{F}_{2^m}$ be a splitting field for $x^n - 1$ over $\mathbb{F}_2$, α be a generator for $\mathbb{F}_{2^m}$ and $T_m(x)$ be a polynomial with maximum degree of $n-1$ and coefficients in $\mathbb{F}_{2^m}$. Similar to Section 2.2.1, the notation of $T(x)$ is used as an alternative to $T_1(x)$ and the variables x and z are used to distinguish the polynomials in the domain and codomain. Let the decomposition of $z^n - 1$ into irreducible polynomials over $\mathbb{F}_2$ be contained in a set $\mathcal{F} = \{f_1(z), f_2(z), \ldots, f_t(z)\}$, i.e. $\prod_{1 \le i \le t} f_i(z) = z^n - 1$. For each $f_i(z)$, there is a corresponding primitive idempotent, denoted as $\theta_i(z)$, which can be obtained by (MacWilliams and Sloane; 1977)

$$\theta_i(z) = \frac{z(z^n - 1)f_i'(z)}{f_i(z)} + \delta(z^n - 1) \tag{2.6}$$

where $f_i'(z) = \frac{d}{dz} f_i(z)$, $f_i'(z) \in T(z)$ and the integer δ is defined by

$$\delta = \begin{cases} 1 & \text{if } \deg(f_i(z)) \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

Let the decomposition of $z^n - 1$ and its corresponding primitive idempotent be listed as follows

$$\begin{array}{ccc} u_1(x) & \theta_1(z) & f_1(z) \\ u_2(x) & \theta_2(z) & f_2(z) \\ \vdots & \vdots & \vdots \\ u_t(x) & \theta_t(z) & f_t(z). \end{array}$$

Here $u_1(x), u_2(x), \ldots, u_t(x)$ are the binary idempotents whose Mattson-Solomon polynomials are $\theta_1(z), \theta_2(z), \ldots, \theta_t(z)$ respectively. Assume that $\mathcal{I} \subseteq \{1, 2, \ldots, t\}$, let the binary polynomials $u(x) = \sum_{\forall i \in \mathcal{I}} u_i(x)$, $f(z) = \prod_{\forall i \in \mathcal{I}} f_i(z)$, and $\theta(z) = \sum_{\forall i \in \mathcal{I}} \theta_i(z)$. It is apparent that, since $u_i(x) = \mathrm{MS}^{-1}(\theta_i(z))$, $u(x) = \mathrm{MS}^{-1}(\theta(z))$ and $u(x)$ is an idempotent†.

Recall that $u(x)$ is a low-weight binary idempotent whose reciprocal polynomial can be used to define the parity-check matrix of a cyclic LDPC code. The number of distinct nth roots of unity which are also roots of the idempotent $u(x)$ determines the dimension of the resulting LDPC code. In this section, the design of cyclic LDPC codes are built around several important features of a code. These features, which are listed as follows, may be easily gleaned from the Mattson-Solomon polynomial of $u(x)$ and the binary irreducible factors of $z^n - 1$.

1. Weight of the idempotent $u(x)$
   The weight of $u(x)$ is the number of nth roots of unity which are zeros of $f(z)$. Note that, $f(\alpha^i) = 0$ if and only if $\theta(\alpha^i) = 1$ since idempotent takes only the values 0 and 1 over $\mathbb{F}_{2^m}$. If $u(x)$ is written as $u_0 + u_i x + \ldots + u_{n-1}x^{n-1}$, following (2.2), it can be seen that

   $$u_i = \theta(\alpha^i) \pmod 2 \qquad \text{for } i = \{0, 1, \ldots, n-1\}.$$

   Therefore $u_i = 1$ precisely when $f(\alpha^i) = 0$, giving $\mathrm{wt}_H(u(x))$ as the degree of the polynomial $f(z)$.

†Since the Mattson-Solomon polynomial of a binary polynomial is an idempotent and vice-versa (MacWilliams and Sloane; 1977), the Mattson-Solomon polynomial of a binary idempotent is also a binary idempotent

2. Number of zeros of $u(x)$
   Following (2.1), it is apparent that the number of zeros of $u(x)$ which are roots of unity, which is also the dimension of the code k, is

$$\text{Number of zeros of } u(x) = k = n - \mathrm{wt}_H\left(\theta(z)\right). \tag{2.7}$$

3. Minimum Hamming distance bound
   The lower-bound of the minimum Hamming distance of a cyclic code defined by idempotent $u(x)$ is given by its BCH bound, which is determined by the number of consecutive powers of α, taken cyclically (mod n), which are also roots of the generating idempotent $e_g(x) = 1 + u(x)$. In the context of $u(x)$, it is the same as the number of consecutive powers of α, taken cyclically (mod n), which are not roots of $u(x)$. Therefore, it is the largest number of consecutive non zero coefficients in $\theta(z)$, taken cyclically (mod n).

The method of finding $f_i(z)$ is well-established and using the above information, a systematic search for idempotents of suitable weight may be developed. To be efficient, the search procedure has to start with an increasing order of $\mathrm{wt}_H(u(x))$ and this requires rearrangement of the set $\mathcal{F}$ such that $\deg(f_i(z)) < \deg(f_{i+1}(z))$ for all i. It is worth mentioning that it is not necessary to evaluate $u(x)$ by taking the Mattson-Solomon polynomial of $\theta(z)$, for each $f(z)$ obtained. It is more efficient to obtain $u(x)$ once the desired code criteria–see the design features listed above, are met.

For an exhaustive search, the complexity is of order $\mathcal{O}\left(2^{|\mathcal{F}|}\right)$. A search algorithm, see Algorithm 2.2, has been developed and it reduces the complexity considerably by targeting the search on the following key parameters. Note that this search algorithm, which is constructed in the Mattson-Solomon domain, is not constrained to find cyclic codes that have girth at least 6.

1. Sparseness of the parity-check matrix
   A necessary condition for the absence of cycles of length 4 in the Tanner graph of a cyclic LDPC code is given by the inequality $\mathrm{wt}_H(u(x))\left(\mathrm{wt}_H(u(x)) - 1\right) \leq n - 1$. Since $\mathrm{wt}_H(u(x)) = \deg(f(z))$, a reasonable bound is

$$\sum_{\forall i \in \mathcal{I}} \deg(f_i(z)) \leq \sqrt{n}.$$

   In practice, a bit beyond this limit is considered to enable finding of good cyclic LDPC codes which have girth of 4 in their underlying Tanner graph.

2. Code-rate
   The code-rate is directly proportional to the number of roots of $u(x)$. If R_{min} represents the minimum desired code-rate then it follows from (2.7) that the search may be refined to consider cases where

$$\mathrm{wt}_H(\theta(z)) \leq (1 - R_{min})n.$$

3. Minimum Hamming distance
   If the idempotent $u(x)$ is orthogonal on each position, then the minimum Hamming distance of the resulting code defined by $u(x)$ is equal to $1+\mathrm{wt}_H(u(x))$. However, for cyclic codes with cycles

of length 4, there is no direct method to determine their minimum Hamming distance and their BCH bound provides a useful information on the minimum Hamming distance lower-bound. Let d be the lowest desired minimum Hamming distance and r_θ be the largest number of consecutive non zero coefficients, taken cyclically, of $\theta(z)$. If a cyclic code has r_θ of d, then its minimum Hamming distance is at least $1 + d$. It follows that the search may be further refined to consider the following case

$$r_\theta \geq d - 1.$$

Algorithm 2.2 MSCodeSearch(n, V, $index$)

Input:

$n \Leftarrow$ block length (odd integer)
V $\Leftarrow$ a vector initialised to $\emptyset$
$index \Leftarrow$ an integer initialised to -1
$R_{min} \Leftarrow$ minimum code-rate of interest
$d \Leftarrow$ lowest expected minimum distance
$\delta \Leftarrow$ small positive integer
$\mathbf{F}(z) \Leftarrow \{f_i(z)\}\ \forall i \in I$ sorted in ascending order of the degree
$\mathbf{Q}(z) \Leftarrow \{\theta_i(z)\}\ \forall i \in I$

Output:

CodesList contains set of codes

1: $\mathbf{T} \Leftarrow \mathbf{V}$
2: **for** $(i{=}index{+}1;\ i \leq |\mathcal{I}|;\ i{+}{+})$ **do**
3: $\mathbf{T}_{\text{prev}} \Leftarrow \mathbf{T}$
4: **if** $(\sum_{\forall j \in \mathbf{T}} \deg(f_j(x)) + \deg(f_i(x)) \leq \sqrt{n} + \delta)$ **then**
5: Append i to T
6: $\theta(z) \Leftarrow \sum_{\forall j \in \mathbf{T}} \theta_j(z)$
7: **if** $(\mathrm{wt}_H(\theta(z)) \leq (1 - R_{min})n$ **and** $r_\theta > d)$ **then**
8: $u(x) \Leftarrow \mathrm{MS}^{-1}(\theta(z))$
9: **if** $(u(x)$ is non-degenerate$)$ **then**
10: $C \Leftarrow$ a cyclic code defined by $u(x)$
11: **if** $(C \notin$ CodeList$)$ **then**
12: Add C to CodeList
13: **end if**
14: **end if**
15: **end if**
16: MSCodeSearch(T, i)
17: **end if**
18: $\mathbf{T} \Leftarrow \mathbf{T}_{\text{prev}}$
19: **end for**

In comparison to the construction method described in Section 2.2.1, we can see that the construction given in Section 2.2.1 starts from the idempotent $u(x)$, whereas that given in this section starts from the idempotent $\theta(z)$, which is the Mattson-Solomon polynomial of $u(x)$. Both construction methods are equivalent and the same cyclic LDPC codes may be produced.

Some good cyclic LDPC codes with cycles of length 4 found using Algorithm 2.2, which may also be found using Algorithm 2.1, are tabulated in Table 2.2. A check based on Lemma 2.2 may be easily incorporated in Step 12 of Algorithm 2.2 to filter out cyclic codes whose Tanner graph has

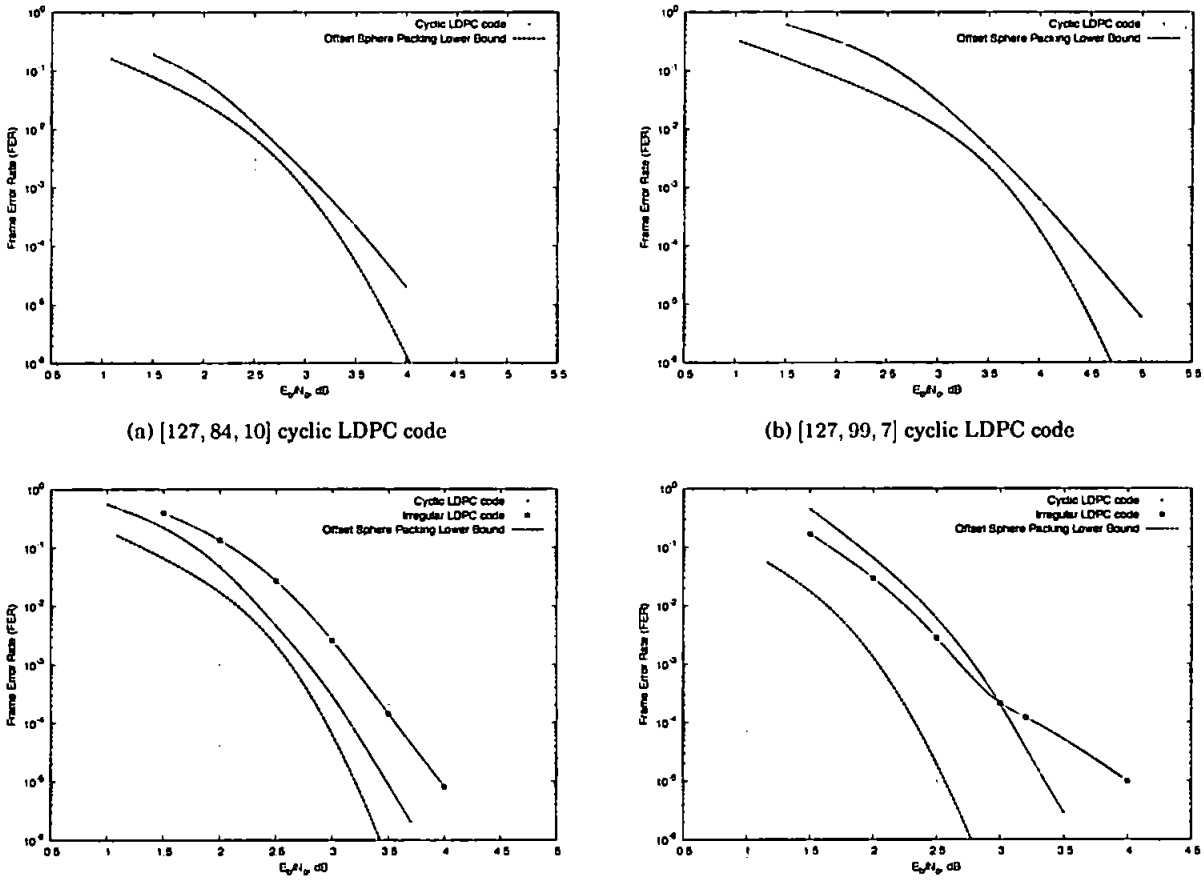
Table 2.2: Several good cyclic LDPC codes with girth of 4

$[n, k, d]$	$u(x)$
$[51, 26, 10]$	$1 + x^3 + x^6 + x^{12} + x^{17} + x^{24} + x^{27} + x^{34} + x^{39} + x^{45} + x^{48}$
$[63, 44, 8]$	$1 + x^9 + x^{11} + x^{18} + x^{21} + x^{22} + x^{25} + x^{27} + x^{36} + x^{37} + x^{42} + x^{44} + x^{45} + x^{50} + x^{54}$
$[117, 72, 12]$	$1 + x + x^2 + x^4 + x^8 + x^{11} + x^{16} + x^{22} + x^{32} + x^{44} + x^{59} + x^{64} + x^{88}$
$[127, 84, 10]$	$1 + x + x^2 + x^4 + x^8 + x^{16} + x^{32} + x^{55} + x^{59} + x^{64} + x^{91} + x^{93} + x^{109} + x^{110} + x^{118}$
$[127, 91, 10]$	$1 + x^2 + x^{10} + x^{18} + x^{29} + x^{32} + x^{33} + x^{49} + x^{50} + x^{54} + x^{58} + x^{65} + x^{74} + x^{76} + x^{78} + x^{86} + x^{87} + x^{88} + x^{92} + x^{93} + x^{95}$
$[127, 92, 7]$	$1 + x^5 + x^{10} + x^{20} + x^{29} + x^{31} + x^{33} + x^{39} + x^{40} + x^{58} + x^{62} + x^{66} + x^{78} + x^{79} + x^{80} + x^{83} + x^{103} + x^{105} + x^{115} + x^{116} + x^{121} + x^{124}$
$[127, 99, 7]$	$1 + x^{13} + x^{16} + x^{18} + x^{22} + x^{26} + x^{39} + x^{42} + x^{45} + x^{46} + x^{49} + x^{57} + x^{65} + x^{68} + x^{70} + x^{78} + x^{80} + x^{90} + x^{91} + x^{92} + x^{96} + x^{97} + x^{102} + x^{103} + x^{105} + x^{108} + x^{111}$

girth of 4.

Figure 2.3 demonstrates the FER performance of several cyclic LDPC codes found by Algorithm 2.2. It is assumed that binary antipodal signalling is employed and the iterative decoder uses the RVCM algorithm (Papagiannis et al.; 2003a). The FER performance is compared against the sphere packing lower bound offset for binary transmission. It can be seen that the codes $[127, 84, 10]$ and $[127, 99, 7]$, despite having cycles of length 4, are around 0.3 dB from the offset sphere packing lower bound at 10^{-4} FER. Figure 2.3c compares two LDPC codes of block size 255 and dimension 175, an algebraic code obtained by Algorithm 2.2 and an irregular code constructed using PEG algorithm (Hu et al.; 2002). It is shown in the figure that, in addition to having an advantage on the minimum Hamming distance, the cyclic LDPC code is 0.4 dB superior to the irregular counterpart, and compared to the offset sphere packing lower bound, it is within 0.25 dB away at 10^{-4} FER. The effect of error floor is apparent in the FER performance of the $[341, 205, 6]$ irregular LDPC code, as shown in Figure 2.3d. The floor of this irregular code is largely attributed to minimum Hamming distance error events. While this irregular code, at low SNR region, has better convergence than does the algebraic LDPC code of the same block length and dimension, the benefit of having larger minimum Hamming distance is obvious as SNR increases. The $[341, 205, 16]$ cyclic LDPC code is approximately 0.8 dB away from the offset sphere packing lower bound at 10^{-4} FER.

It has been shown that short block length ($n \leq 350$) cyclic LDPC codes have outstanding performance; the gap to the offset sphere packing lower bound is relatively close. On the other hand, it has also been observed that, as the block length increases, the algebraic LDPC codes although have large minimum Hamming distance, have a convergence issue and the threshold to the waterfall region is at larger E_b/N_0. The convergence problem arises because as the minimum Hamming distance is increased, the weight of the idempotent $u(x)$, which defines the parity-check matrix, also increases. In fact if $u(x)$ satisfies Lemma 2.2, $\mathrm{wt}_H(u(x)) = d - 1$ where d is the minimum Hamming distance of the code. Large value of $\mathrm{wt}_H(u(x))$ results in a parity-check matrix that is not as sparse



(a) [127, 84, 10] cyclic LDPC code

(b) [127, 99, 7] cyclic LDPC code

(c) [255, 175, 17] cyclic and [255, 175, 6] irregular PEG LDPC codes

(d) [341, 205, 16] cyclic and [341, 205, 6] irregular PEG LDPC codes

Figure 2.3: FER performance of binary cyclic LDPC codes

as that of a good irregular LDPC code of the same block length and dimension.

2.2.3 Non Binary Extension of the Cyclotomic Coset-based LDPC Codes

The code construction technique for the cyclotomic coset-based binary cyclic LDPC codes, which are discussed in Section 2.2.1, may be extended to non binary fields and it is the subject of this section. Similar to the binary case, the non binary construction produces the dual code idempotent which is used to define the parity-check matrix of the associated LDPC code.

Let m and m' be positive integers with $m \mid m'$, so that $\mathbb{F}_{2^m}$ is a subfield of $\mathbb{F}_{2^{m'}}$. Let n be a positive odd integer and $\mathbb{F}_{2^{m'}}$ be the splitting field of $x^n - 1$ over $\mathbb{F}_{2^m}$, so that $n|2^{m'} - 1$. Let $r = (2^{m'} - 1)/n$, $l = (2^{m'} - 1)/(2^m - 1)$, α be a generator for $\mathbb{F}_{2^{m'}}$ and β be a generator of $\mathbb{F}_{2^m}$, where $\beta = \alpha^l$. Let $T_a(x)$ be the set of polynomials of degree at most $n - 1$ with coefficients in $\mathbb{F}_{2^a}$. For the case of $a = 1$, we may denote $T_1(x)$ by $T(x)$ for convenience.

The Mattson-Solomon polynomial and its corresponding inverse, (2.1) and (2.2) respectively, may

be redefined as

$$A(z) = \mathrm{MS}\left(a(x)\right) = \sum_{j=0}^{n-1} a(\alpha^{-rj}) z^j \tag{2.8}$$

$$a(x) = \mathrm{MS}^{-1}\left(A(z)\right) = \frac{1}{n} \sum_{i=0}^{n-1} A(\alpha^{ri}) x^i \tag{2.9}$$

where $a(x) \in T_{m'}(x)$ and $A(z) \in T_{m'}(z)$.

Recall that a polynomial $e(x) \in T_m(x)$ is termed an idempotent if the property $e(x) = e(x)^2 \pmod{x^n - 1}$ is satisfied. Note that $e(x) \neq e(x^2) \pmod{x^n - 1}$ unless $m = 1$. The following definition shows how to construct an idempotent for binary and non binary polynomials.

2.13 *Definition (Cyclotomic Idempotent).* Assume that $\mathcal{N}$ be a set as defined in Section 2.2.1, let $s \in \mathcal{N}$ and let $C_{s,i}$ represent the $(i+1)$th element of C_s, the cyclotomic coset of $s \pmod{n}$. Assume that the polynomial $e_s(x) \in T_m(x)$ is given by

$$e_s(x) = \sum_{0 \le i \le |C_s|-1} e_{C_{s,i}} x^{C_{s,i}}, \tag{2.10}$$

where $|C_s|$ is the number of elements in C_s. In order for $e_s(x)$ to be an idempotent, its coefficients may be chosen in the following manner,

i) if $m = 1$, $e_{C_{s,i}} = 1$,

ii) otherwise, $e_{C_{s,i}}$ is defined recursively as follows

$$\begin{aligned} &\text{for } i = 0, \quad e_{C_{s,i}} \in \{1, \beta, \beta^2, \ldots, \beta^{2^m-2}\}, \\ &\text{for } i > 0, \quad e_{C_{s,i}} = e^2_{C_{s,i-1}}. \end{aligned}$$

The idempotent $e_s(x)$ is referred to as a cyclotomic idempotent.

2.14 *Definition (Parity-Check Idempotent).* Let $\mathcal{M} \subseteq \mathcal{N}$ and let $u(x) \in T_m(x)$ be

$$u(x) = \sum_{s \in \mathcal{M}} e_s(x). \tag{2.11}$$

The polynomial $u(x)$ is an idempotent and it is called a parity-check idempotent.

As in Section 2.2.1, the parity-check idempotent $u(x)$ is used to define the $\mathbb{F}_{2^m}$ cyclic LDPC code over $\mathbb{F}_{2^m}$, which may be denoted by $[n, k, d]_{2^m}$. The parity-check matrix is being made up of n cyclic shifts of $x^{\deg(u(x))} u(x^{-1})$. For non binary case, the minimum Hamming distance d of the cyclic code is bounded by

$$d_0 + 1 \le d \le \min\left(\mathrm{wt}_H(g(x)), \mathrm{wt}_H(1 + u(x))\right)$$

where d_0 is the maximum run of consecutive ones in $U(z) = \mathrm{MS}(u(x))$, taken cyclically mod n.

Based on the description given above, a procedure to construct a cyclic LDPC code over $\mathbb{F}_{2^m}$ may be devised and described as follows.

1. Given some integers m and n, obtain the splitting field ($\mathbb{F}_{2^{m'}}$) of $x^n - 1$ over $\mathbb{F}_{2^m}$. Unless the condition of $m \mid m'$ is satisfied, $\mathbb{F}_{2^m}$ cyclic LDPC code of length n cannot be constructed.
2. Generate the cyclotomic cosets modulo $2^{m'} - 1$ and denote it C'.
3. Derive a polynomial $p(x)$ from C' and let $s \in \mathcal{N}$ be the smallest positive integer such that $|C'_s| = m$. The polynomial $p(x)$ is the minimal polynomial of α^s,
$$p(x) = \prod_{0 \le i < m} \left(x + \alpha^{C'_{s,i}}\right) \tag{2.12}$$
Construct all elements of $\mathbb{F}_{2^m}$ using $p(x)$ as the primitive polynomial.
4. Let C be the cyclotomic cosets modulo n and let $\mathcal{N}$ be a set containing the smallest number in each coset of C. Assume that there exists a non empty set $\mathcal{M} \subset \mathcal{N}$ and following Definition 2.14, construct the parity-check idempotent $u(x)$. The coefficients of $u(x)$ can be assigned following Definition 2.13.
5. Generate the parity-check matrix of C using the n cyclic shifts of $x^{\deg(u(x))}u(x^{-1})$.
6. Compute r and l, then take the Mattson-Solomon polynomial of $u(x)$ to produce $U(z)$. Obtain the code dimension and the lower-bound of the minimum Hamming distance from $U(z)$.

Construction of $[21, 15, \geq 5]_{2^6}$ cyclic LDPC code

Example 2.4: Consider the construction of a $n = 21$ cyclic LDPC code over $\mathbb{F}_{2^6}$. The splitting field of $x^{21} - 1$ over $\mathbb{F}_{2^6}$ is $\mathbb{F}_{2^6}$ and this implies that $m = m' = 6$, $r = 3$ and $l = 1$. Let C and C' denote the cyclotomic cosets modulo n and $2^{m'} - 1$ respectively. It is known that $|C'_1| = 6$ and therefore the primitive polynomial $p(x)$ has roots of α^j, for all $j \in C'_1$, i.e. $p(x) = 1+x+x^6$. By letting $1+\beta+\beta^6 = 0$, all of the elements of $\mathbb{F}_{2^6}$ can be defined. If $u(x)$ is the parity-check idempotent generated by the sum of the cyclotomic idempotents defined by C_s where $s \in \{\mathcal{M} : 5, 7, 9\}$ and $e_{C_{s,0}}$ for all $s \in \mathcal{M}$ be β^{23}, 1 and 1 respectively,

$$u(x) = \beta^{23}x^5 + x^7 + x^9 + \beta^{46}x^{10} + \beta^{43}x^{13} + x^{14} + x^{15} + \beta^{53}x^{17} + x^{18} + \beta^{58}x^{19} + \beta^{29}x^{20}$$

and its Mattson-Solomon polynomial $U(z)$ indicates that it is a $[21, 15, \geq 5]_{2^6}$ cyclic code, whose binary image is a $[126, 90, 8]$ linear code.

A systematic algorithm has been developed to sum up all combinations of the cyclotomic idempotents to search for all possible $\mathbb{F}_{2^m}$ cyclic codes of a given length. Same as Algorithm 2.2, the search algorithm is targeted on the following key parameters:

1. Sparseness of the resulting parity-check matrix
   Since the parity-check matrix is directly derived from $u(x)$ which consists of the sum of the cyclotomic idempotents, only low-weight cyclotomic idempotents are of interest. Let W_{max} be the maximum $\mathrm{wt}_H(u(x))$ then the search algorithm will only choose the cyclotomic idempotents whose sum has total weight less than or equal to W_{max}.

2. High code-rate
   The number of roots of $u(x)$ which are also roots of unity define the dimension of the resulting LDPC code. Let the integer k_{min} be defined as the minimum code dimension, the cyclotomic idempotents that are of interest are those whose Mattson-Solomon polynomial has at least k_{min} zeros.

3. High minimum Hamming distance
   Let the integer d' be the smallest value of the minimum Hamming distance of the code. The sum of the cyclotomic idempotents should have at least $d' - 1$ consecutive powers of β which are roots of unity but not roots of $u(x)$.

Following Definition 2.14 and the Mattson-Solomon polynomial

$$U(z) = \mathrm{MS}\left(\sum_{s\in\mathcal{M}} e_s(x)\right) = \sum_{s\in\mathcal{M}} E_s(z),$$

it is possible to maximise the run of the consecutive ones in $U(z)$ by varying the coefficients of $e_s(x)$. It is therefore important that all possible non zero values of $e_{C_{s,0}}$ for all $s \in \mathcal{M}$ are included to guarantee that codes with the highest possible minimum Hamming distance can be found, or at least to obtain a better estimate of the minimum Hamming distance of a code.

Table 2.3 outlines some examples of $[n, k, d]_{2^m}$ cyclic LDPC codes. The detail parameters are given in the table. The non binary algebraic LDPC codes in this table perform considerably well under iterative decoding as shown in Figure 2.4. For the results in this figure, binary antipodal signalling and AWGN channel are assumed. Furthermore, the RVCM algorithm is employed in the iterative decoder. The FER performance of these non binary codes is compared to the offset sphere packing lower bound.

As mentioned in Section 2.1.2, an inverse relationship between the convergence of iterative decoder and the minimum Hamming distance of a code has been observed. The algebraic LDPC codes, which have larger minimum Hamming distance compared to irregular LDPC codes, do not converge well at long block length. To the best of author's knowledge, the best convergence at long block length can only be realised by irregular LDPC codes with good degree distributions. Figure 2.5 shows the performance of two LDPC codes of block length 5461 bits and code rate 0.6924; one is an irregular code constructed using the PEG algorithm and the other one is an algebraic code of minimum Hamming distance 43 based on cyclotomic coset and idempotent, see Table 2.1. These results are obtained by assuming AWGN channel and binary antipodal signalling with belief propagation iterative decoder with a maximum of 100 iterations. It can be seen that at 10^{-5} FER, the irregular PEG code is superior by approximately 1.6 dB over the algebraic cyclic LDPC code. Despite this disadvantage at long block length, short algebraic LDPC codes are attractive as they offer the best performance and yet have low complexity encoder structure.

Table 2.3: Examples of $[n, k, d]_{2^m}$ cyclic LDPC codes

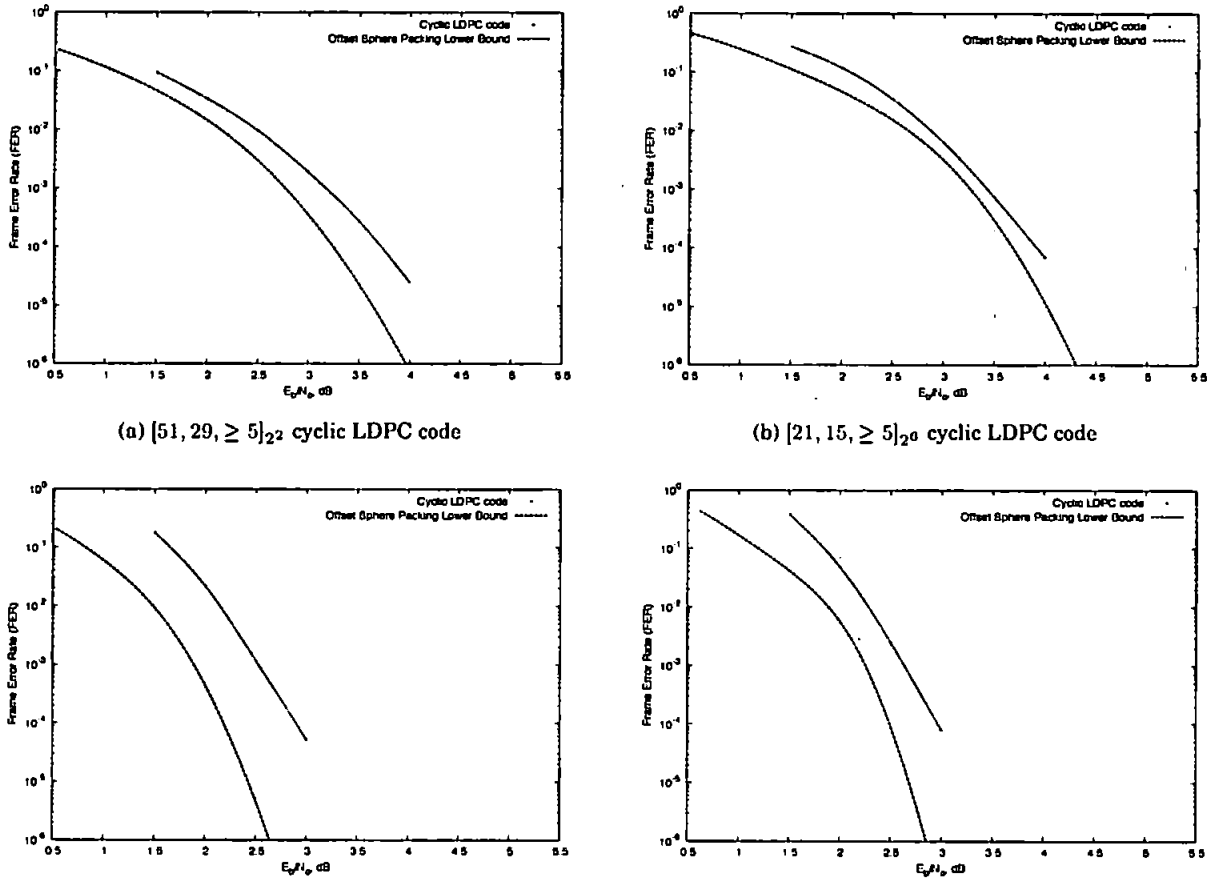
q	$[n, k]$	$u(x)$	d	$d_b^\dagger$	Comment
$\mathbb{F}_{2^2}$	$[51, 29]$	$\beta^2x^3+\beta x^6+\beta^2x^{12}+\beta^2x^{17}+\beta x^{24}+\beta x^{27}+\beta x^{34}+\beta^2x^{39}+\beta x^{45}+\beta^2x^{48}$	≥ 5	10	$m = 2$, $m' = 8$, $r = 5$ and $l = 85$
	$[255, 175]$	$\beta x^7+\beta^2x^{14}+\beta x^{28}+\beta^2x^{56}+x^{111}+\beta x^{112}+x^{123}+\beta^2x^{131}+x^{183}+x^{189}+\beta x^{193}+x^{219}+x^{222}+\beta^2x^{224}+x^{237}+x^{246}$	≥ 17	≤ 20	$m = 2$, $m' = 8$, $r = 1$ and $l = 85$
	$[273, 191]$	$\beta^2x^{23}+\beta x^{37}+\beta x^{46}+\beta^2x^{74}+\beta x^{91}+\beta^2x^{92}+\beta^2x^{95}+\beta^2x^{107}+x^{117}+\beta x^{148}+\beta^2x^{155}+\beta^2x^{182}+\beta x^{184}+\beta x^{190}+x^{195}+\beta x^{214}+x^{234}$	≥ 18	≤ 20	$m = 2$, $m' = 12$, $r = 15$ and $l = 1365$
$\mathbb{F}_{2^3}$	$[63, 40]$	$1+\beta^5x^9+\beta x^{13}+\beta^3x^{18}+\beta^2x^{19}+\beta^2x^{26}+\beta^6x^{36}+\beta^4x^{38}+\beta x^{41}+\beta^4x^{52}$	≥ 6	10	$m = 3$, $m' = 6$, $r = 1$ and $l = 9$
	$[63, 43]$	$\beta^2x^9+\beta^3x^{11}+\beta^4x^{18}+x^{21}+\beta^6x^{22}+\beta^3x^{25}+x^{27}+\beta x^{36}+\beta^5x^{37}+x^{42}+\beta^5x^{44}+x^{45}+\beta^6x^{50}+x^{54}$	≥ 8	≤ 12	$m = 3$, $m' = 6$, $r = 1$ and $l = 9$
	$[91, 63]$	$\beta^6x+\beta^5x^2+\beta^3x^4+\beta^6x^8+\beta x^{13}+\beta^5x^{16}+\beta^5x^{23}+\beta^2x^{26}+\beta^3x^{32}+\beta^5x^{37}+\beta^3x^{46}+\beta^4x^{52}+\beta^6x^{57}+\beta^6x^{64}+\beta^3x^{74}$	≥ 8	≤ 10	$m = 3$, $m' = 12$, $r = 45$ and $l = 585$
$\mathbb{F}_{2^4}$	$[85, 48]$	$1+\beta^{12}x^{21}+\beta^9x^{42}+\beta^6x^{53}+\beta^3x^{69}+\beta^9x^{77}+\beta^{12}x^{81}+\beta^6x^{83}+\beta^3x^{84}$	≥ 7	≤ 12	$m = 4$, $m' = 8$, $r = 3$ and $l = 17$
$\mathbb{F}_{2^5}$	$[31, 20]$	$1+\beta^{28}x^5+\beta^7x^9+\beta^{25}x^{10}+x^{11}+x^{13}+\beta^{14}x^{18}+\beta^{19}x^{20}+x^{21}+x^{22}+x^{26}$	≥ 7	12	$m = 5$, $m' = 5$, $r = l$ and $l = 1$
	$[31, 21]$	$\beta^{23}x^5+\beta^{29}x^9+\beta^{15}x^{10}+\beta x^{11}+\beta^4x^{13}+\beta^{27}x^{18}+\beta^{30}x^{20}+\beta^{16}x^{21}+\beta^2x^{22}+\beta^8x^{26}$	≥ 4	8	$m = 5$, $m' = 5$, $r = 1$ and $l = 1$
$\mathbb{F}_{2^6}$	$[21, 15]$	$\beta^{23}x^5+x^7+x^9+\beta^{46}x^{10}+\beta^{43}x^{13}+x^{14}+x^{15}+\beta^{53}x^{17}+x^{18}+\beta^{58}x^{19}+\beta^{29}x^{20}$	≥ 5	8	$m = 6$, $m' = 6$, $r = 3$ and $l = 1$

† The minimum Hamming distance of the binary image; it is determined using the improved Zimmermann algorithm, Algorithm 4.1.

2.3 Irregular LDPC Codes from Progressive-Edge-Growth Construction

It is shown in Hu et al. (2005) that LDPC codes obtained from the PEG construction method can perform better than other randomly constructed LDPC codes. The PEG algorithm adds edges to each vertex such that the local girth is maximised. The PEG algorithm, which is given in Hu et al. (2005), considers only the variable degree sequence, and the check degree sequence is maintained to be as uniform as possible. In this section, the results of experimental constructions of irregular LDPC codes using the PEG algorithm are presented. Analysis on the effects of the vertex degree ordering and degree sequences have been carried out by means of computer simulations. All simulation results in this section, unless otherwise stated, were obtained by using binary antipodal signalling with the belief propagation decoder with a maximum of 100 iterations and by assuming AWGN channel.

Figure 2.6 shows the FER performance of various $[2048, 1024]$ irregular LDPC codes constructed using the PEG algorithm with different vertex degree ordering. These LDPC codes have variable degree sequence $\Lambda_\lambda(x) = 0.475x^2 + 0.280x^3 + 0.035x^4 + 0.109x^5 + 0.101x^{15}$. Let $(v_0, v_1, \ldots, v_i, \ldots, v_{n-1})$



(a) $[51, 29, \geq 5]_{2^2}$ cyclic LDPC code

(b) $[21, 15, \geq 5]_{2^6}$ cyclic LDPC code

(c) $[85, 48, \geq 7]_{2^4}$ cyclic LDPC codes

(d) $[255, 175, \geq 17]_{2^2}$ cyclic LDPC codes

Figure 2.4: FER performance of non binary cyclic LDPC codes

be a set of variable vertices of an LDPC code. Code 0 and Code 1 LDPC codes were constructed with an increasing vertex degree ordering, i.e. $\deg(v_0) \leq \deg(v_1) \leq \ldots \leq \deg(v_{n-1})$, whereas the remaining LDPC codes were constructed with random vertex degree ordering. Figure 2.6 clearly shows that, unless the degree of the variable vertices is assigned in an increasing order, poor LDPC codes are obtained. In random degree ordering of half rate codes, it is very likely to encounter the situation where, as the construction approaches the end, there are some low degree variable vertices that have no edge connected to them. Since almost all of the variable vertices would have already had edges connected to them, the low degree variable vertices would not have many choice of edges to connect in order to maximise the local girth. It has been observed that, in many cases, these low degree variable vertices are connected to each other, forming a cycle which involves all vertices, and the resulting LDPC codes may have a considerably low minimum Hamming distance. If d variable vertices are connected to each other and a cycle of length $2d$ is formed, then the minimum Hamming distance of the resulting code is d because the sum of these d columns in the corresponding parity-check matrix H is 0^T.

On the other hand, for the alternative construction which starts from an increasing degree of the variable vertices, edges are connected to the low degree variable vertices earlier in the process.

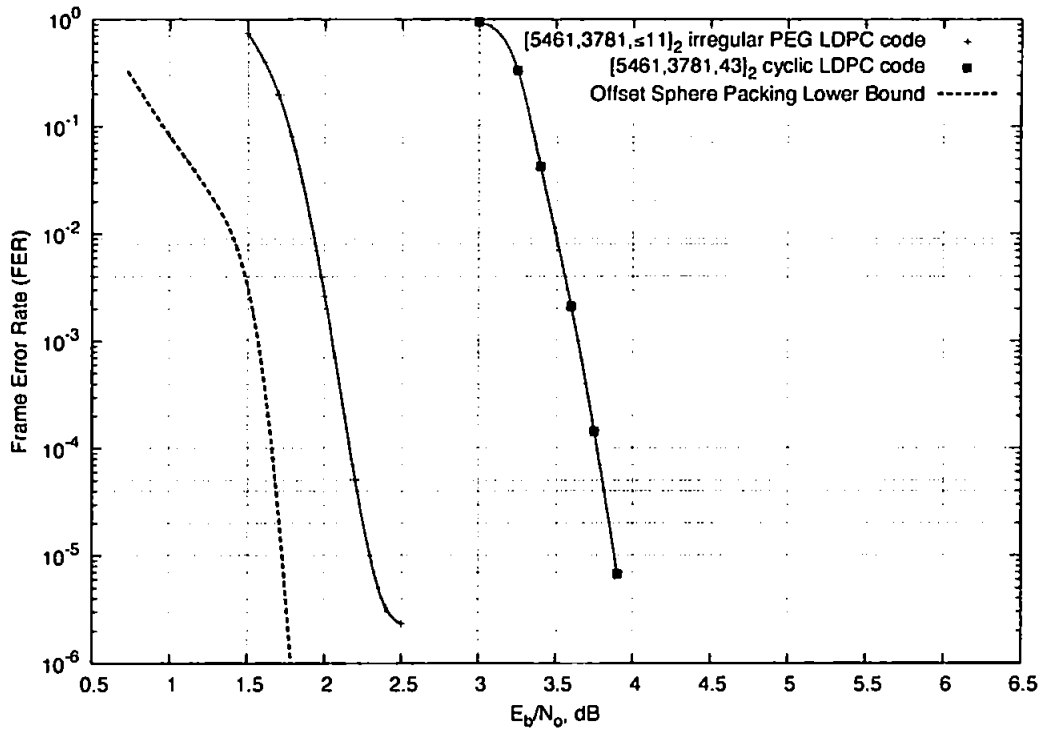


Figure 2.5: FER performance of algebraic and irregular LDPC codes of rate 0.6924 and block length 5461 bits

Short cycles, which involve the low degree variable vertices and lead to low minimum Hamming distance, may be avoided by ensuring these low degree variable vertices to have edges connected to the parity-check vertices which are connected to high degree variable vertices.

It can be expected that the PEG algorithm will almost certainly produce no good LDPC codes if the degree of the variable vertices is assigned in descending order. Throughout this thesis, unless otherwise stated, it is assumed that all PEG-based LDPC codes are constructed with increasing variable vertex degree ordering.

Figure 2.7 shows the effect of low degree variable vertices, especially λ_2 and λ_3, on the FER performance of various $[512, 256]$ PEG-constructed irregular LDPC codes. Table 2.4 shows the variable degree sequences of the simulated irregular codes. Figure 2.7 shows that, with the fraction of high degree variable vertices kept constant, the low degree variable vertices have influence over the convergence in the waterfall region. As the fraction of low degree variable vertices is increased, the FER in the low signal-to-noise ratio (SNR) region improves. On the other hand, LDPC codes with high fraction of low degree variable vertices tend to have low minimum Hamming distance and as expected, these codes exhibit early error floor and this effect is clearly depicted by Code 7 and Code 8 which have the highest fraction of low degree variable vertices among all the codes in Figure 2.7. Among all of the codes, Code 6 and Code 24 appear to have the best performance.

Figure 2.8 demonstrates the effect of high degree variable vertices on the FER performance. These codes are rate 3/4 irregular LDPC codes of length 1024 bits with the same degree sequences, apart from their maximum variable vertex degree. One group has maximum degree of 8 and the

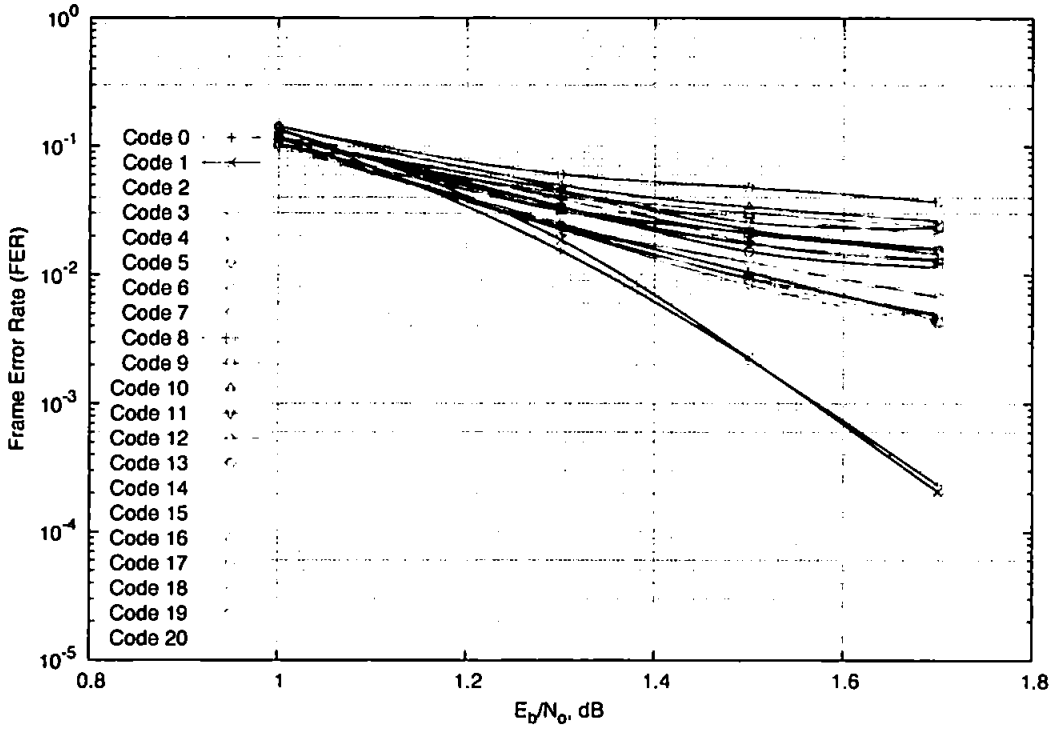


Figure 2.6: Effect of vertex degree ordering in PEG algorithm

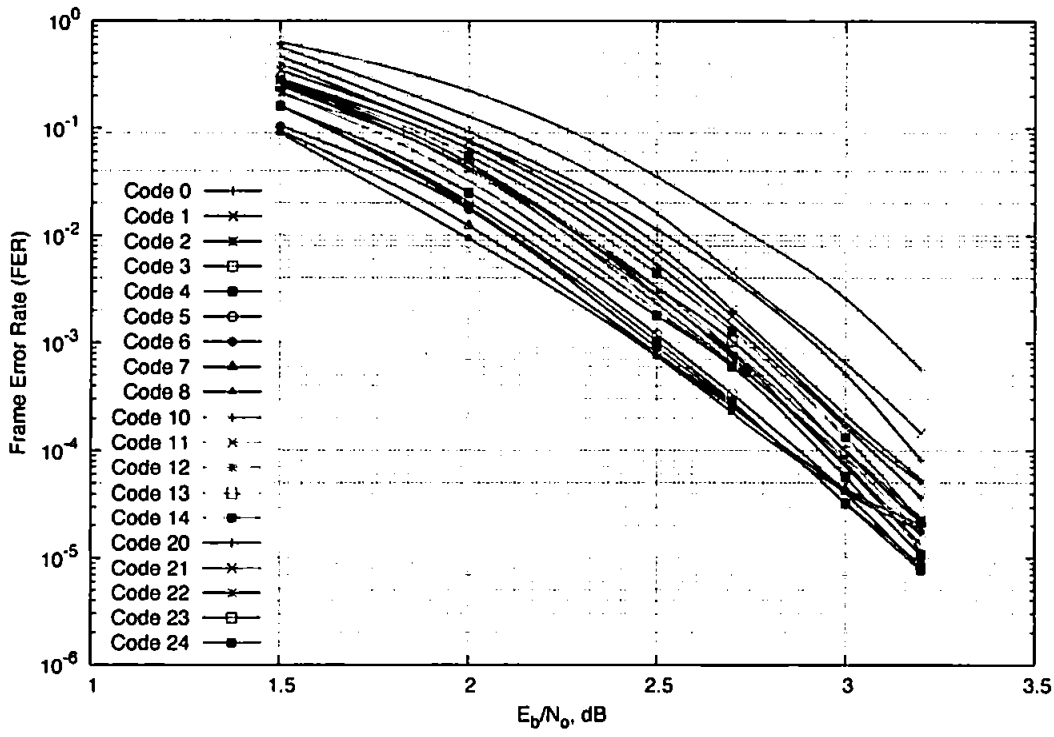


Figure 2.7: Effect of low degree variable vertices

Table 2.4: Variable degree sequences for codes in Figure 2.7.

Code	λ_2	λ_3	λ_4	λ_5	λ_{14}
Code 0	0.150	0.350	0.350	0.050	0.100
Code 1	0.200	0.325	0.325	0.050	0.100
Code 2	0.250	0.300	0.300	0.050	0.100
Code 3	0.300	0.275	0.275	0.050	0.100
Code 4	0.350	0.250	0.250	0.050	0.100
Code 5	0.400	0.225	0.225	0.050	0.100
Code 6	0.450	0.200	0.200	0.050	0.100
Code 7	0.500	0.175	0.175	0.050	0.100
Code 8	0.550	0.150	0.150	0.050	0.100
Code 10	0.150	0.700	0.000	0.050	0.100
Code 11	0.200	0.550	0.100	0.050	0.100
Code 12	0.250	0.400	0.200	0.050	0.100
Code 13	0.300	0.250	0.300	0.050	0.100
Code 14	0.350	0.100	0.400	0.050	0.100
Code 20	0.150	0.000	0.700	0.050	0.100
Code 21	0.200	0.100	0.550	0.050	0.100
Code 22	0.250	0.200	0.400	0.050	0.100
Code 23	0.300	0.300	0.250	0.050	0.100
Code 24	0.350	0.400	0.100	0.050	0.100

other group has maximum degree of 12. From Figure 2.8, it is clear that the LDPC codes with maximum variable vertex degree of 12 converges better under iterative decoding than those codes with maximum variable vertex degree of 8.

Code	λ_1	λ_2	λ_3	λ_4	λ_5	λ_{12}
Code 0	0.000625	0.249375	0.644375			0.105625
Code 1	0.000625	0.249375	0.420000	0.224375		0.105625
Code 2	0.000625	0.249375	0.195000	0.449375		0.105625
Code 3	0.000625	0.249375		0.420000	0.224375	0.105625
Code 4	0.000625	0.249375		0.195000	0.449375	0.105625
Code 5	0.000625	0.249375	0.420000	0.111875	0.111875	0.106250
Code 6	0.000625	0.249375	0.195000	0.224375	0.224375	0.106250
Code 7	0.000625	0.249375	0.420000		0.224375	0.105625
Code 8	0.000625	0.249375	0.195000		0.449375	0.105625
Code 9	0.000625	0.249375	0.449375		0.195000	0.105625
Code 10	0.000625	0.249375	0.449375	0.097500	0.097500	0.105625
Code 11	0.000625	0.249375	0.449375	0.044375	0.150000	0.106250
Code 12	0.000625	0.249375	0.495000		0.150000	0.105000
Code 13	0.000625	0.249375	0.495000	0.075000	0.075000	0.105000
Code 14	0.000625	0.249375	0.495000	0.037500	0.111875	0.105625
Code 15	0.000625	0.249375	0.570000		0.075000	0.105000
Code 16	0.000625	0.249375	0.570000	0.037500	0.037500	0.105000

Table 2.5: Variable degree sequences of LDPC codes in Figure 2.9.

Similar to Figure 2.7, the effect of having various low degree variable vertices is also demonstrated on Figure 2.9. In this case, the LDPC codes are constructed to have linear time encoding complexity, where the parity symbols are commonly described as having a zig-zag pattern (Ping

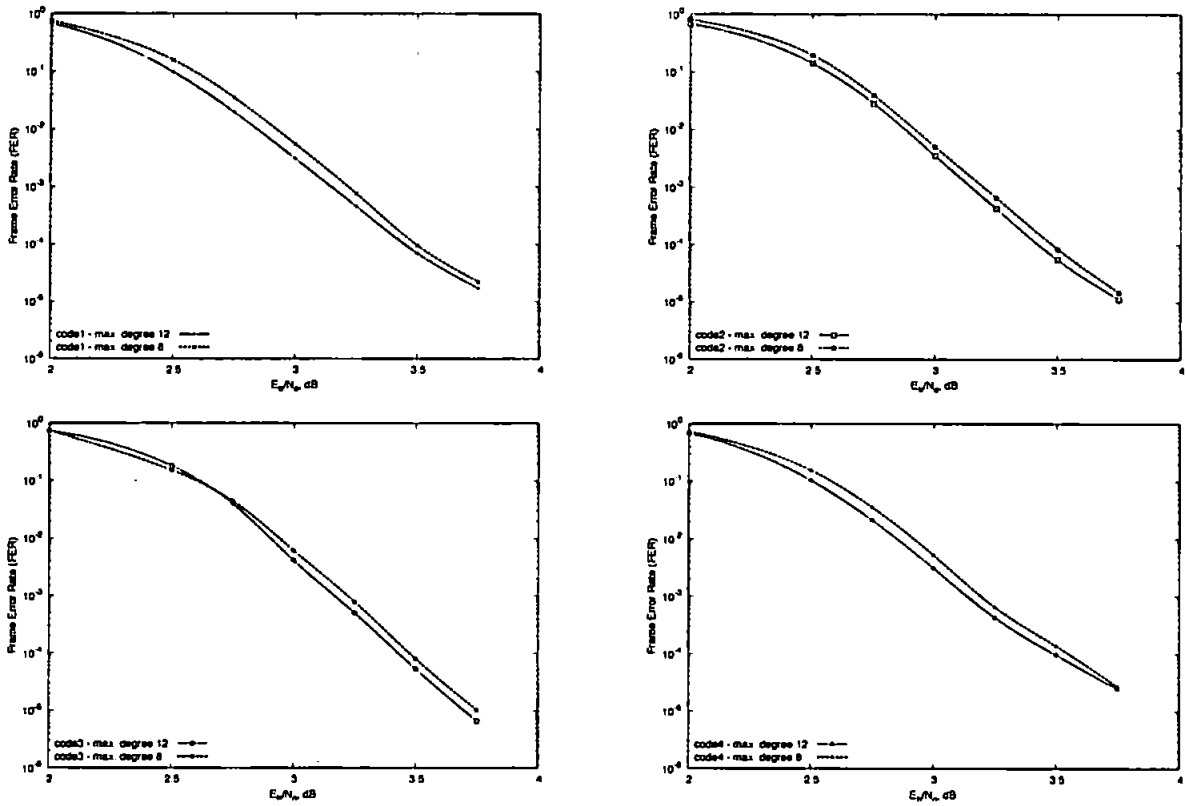


Figure 2.8: Effect of high degree variable vertices

et al.; 1999). In this case, λ_1 and λ_2 of these LDPC codes are fixed and the effect of varying λ_3, λ_4 and λ_5 is investigated. The variable degree sequences of the LDPC codes under investigation, which are rate 3/4 codes of length 1600 bits, are depicted on Table 2.5. The results shows that, as in the previous cases, these low degree variable vertices contribute to the waterfall region on the FER curve. The contribution of λ_i is more significant than that of λ_{i+1} and this may be observed by comparing the FER curves of Code 1 with either Code 3 or Code 4, which has λ_3 of 0.0. It can also be seen that Code 0, which has the most variable vertices of low degree, exhibits high error floor.

In contrast to Figure 2.9, Figure 2.10 shows the effect of varying high degree variable vertices. The LDPC codes considered here all have the same code rate and block length as those in Figure 2.9 and their variable degree sequences are shown in Table 2.6. The results show that:

- The contribution of the high degree variable vertices is in the high SNR region. Consider Code 10 to Code 33, those LDPC codes that have larger λ_{12} tend to be more resilient to error in the high SNR region than those with smaller λ_{12}. At $E_b/N_o = 3.0$ dB, Code 10, Code 11 and Code 12 are inferior to Code 13 and similarly, Code 23 and Code 33 have the best performance in their group.

- Large value of maximum variable vertex degree may not always lead to improved FER performance. For example, Code 5 and Code 6 do not perform as good as Code 4 at $E_b/N_o =$ 3.0 dB. This may be explained as follows. As the maximum variable vertex degree is increased,

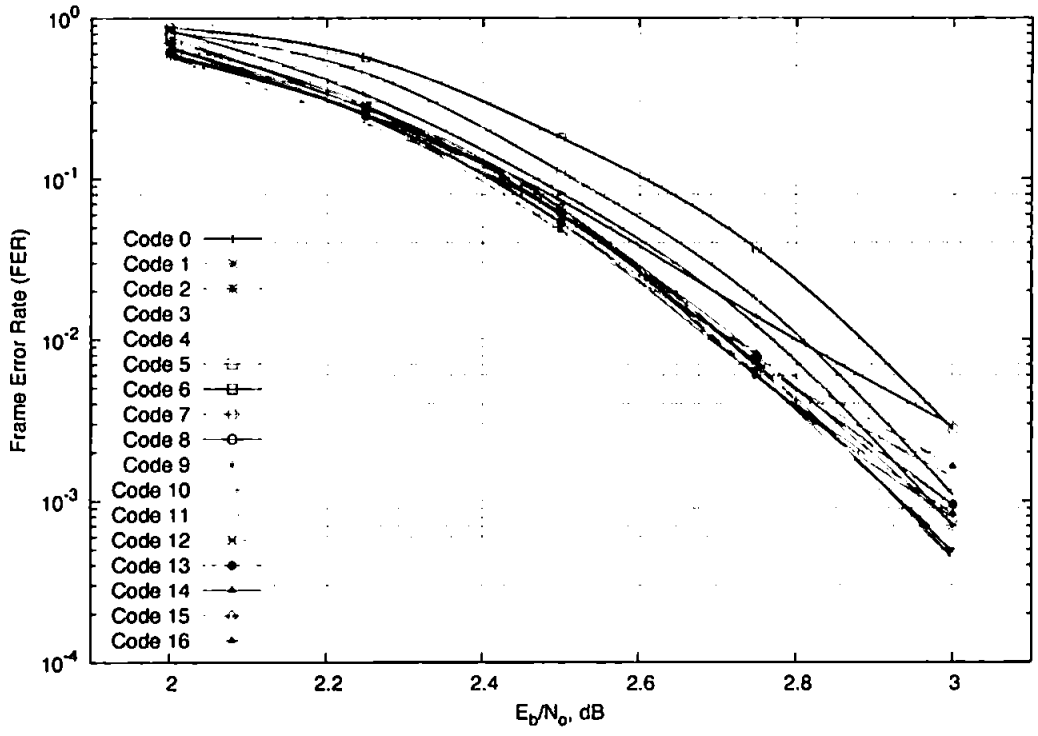


Figure 2.9: Effect of varying low degree variable vertices.

some of the variable vertices have many edges connected to them, in the other words the corresponding symbols are checked by many parity-check equations. This increases the chances of having unreliable information from some of these equations during iterative decoding. In addition, larger maximum variable vertex degree also increases the number of short cycles in the underlying Tanner graph of a code, and following McEliece et al. (1998) and Etzion et al. (1999), short cycles have negative contributions toward the convergence of iterative decoder.

In Appendix A, a method of constructing a quasi-cyclic LDPC code by expanding a small irregular LDPC code is described. Protograph is the name commonly used to refer to a small regular or irregular LDPC code.

2.4 Summary

- The application of cyclotomic cosets, idempotents and Mattson-Solomon polynomials can produce many binary cyclic LDPC codes whose parity-check equations are orthogonal in each position. While some of these cyclic codes have the same parameters as a class of the finite geometry codes, others are new. A key feature of this construction technique is the incremental approach to the minimum Hamming distance and thus, the sparseness of the resulting parity-check matrix of the code.

- Binary cyclic LDPC codes may also be constructed by considering idempotents in the Mattson-Solomon domain. This approach gives a different insight into the cyclotomic coset-based con-

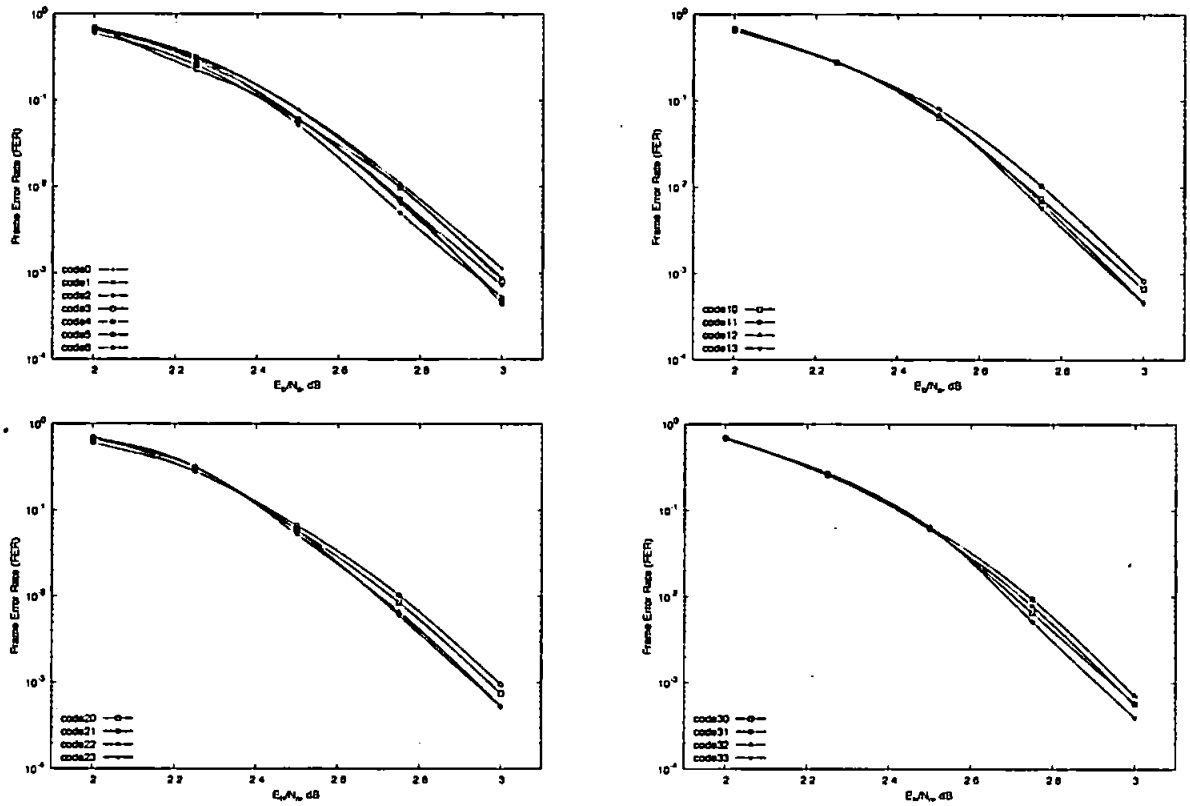


Figure 2.10: Effect of varying high degree variable vertices

struction.

- It has also been shown that, for short algebraic LDPC codes considered in this chapter, the myth of codes which have cycles of length 4 in their Tanner graph do not converge well with iterative decoding is not necessary true.
- The cyclotomic coset based construction can be easily extended to produce good non binary algebraic LDPC codes.
- Good irregular LDPC codes may be constructed using the Progressive-Edge-Growth algorithm. This algorithm adds edges to the variable and check vertices in a way such that the local girth is maximised. Structured LDPC codes, such us those which have quasi-cyclic structure, are of interest to industry due to the simplification of their encoder and decoder.

Table 2.6: Variable degree sequences of LDPC codes in Figure 2.10. Note that $\lambda_1 = 0.000625$ and $\lambda_2 = 0.249375$.

Code	λ_3	λ_4	λ_5	λ_8	λ_9	λ_{10}	λ_{11}	λ_{12}	λ_{13}	λ_{14}
Code 0	0.420000	0.111875	0.111875	0.10625						
Code 1	0.420000	0.111875	0.111875		0.10625					
Code 2	0.420000	0.111875	0.111875			0.10625				
Code 3	0.420000	0.111875	0.111875				0.10625			
Code 4	0.420000	0.111875	0.111875					0.10625		
Code 5	0.420000	0.111875	0.111875						0.10625	
Code 6	0.420000	0.111875	0.111875							0.10625
Code 10	0.434375	0.111875	0.111875					0.091875		
Code 11	0.449375	0.111875	0.111875					0.076875		
Code 12	0.405000	0.111875	0.111875					0.121250		
Code 13	0.390000	0.111875	0.111875					0.136250		
Code 20	0.420000	0.127500	0.111875					0.090625		
Code 21	0.420000	0.142500	0.111875					0.075625		
Code 22	0.420000	0.097500	0.111875					0.120625		
Code 23	0.420000	0.082500	0.111875					0.135625		
Code 30	0.420000	0.111875	0.127500					0.090625		
Code 31	0.420000	0.111875	0.142500					0.075625		
Code 32	0.420000	0.111875	0.097500					0.120625		
Code 33	0.420000	0.111875	0.082500					0.135625		

3 Improvements to Iterative Decoder

Realising an optimum soft decision decoder for any coded system is an non-polynomial-time (NP) hard problem. For a general $[n, k, d]$ linear code over $\mathbb{F}_q$, the optimum decoding complexity is proportional to $\min\{q^k, q^{n-k}\}$. As a consequence, the optimum decoder can only be realised for very high-rate or very low-rate codes.

An iterative decoder is an approximation to the optimum decoder. An iterative decoder may not converge to an optimal solution and this leads to a gap with respect to the optimum performance. This chapter discusses techniques to improve the convergence of iterative decoder, aimed to bring the gap to optimum performance closer. The discussion in this chapter focuses on iterative decoder for LDPC codes, the belief propagation algorithm.

Parts of this chapter appear in a conference proceedings: Tjhai, C., Tomlinson, M., Horan, R., Ambroze, M. and Ahmed, M. (2005), "Near maximum-likelihood performance of some new cyclic codes constructed in the finite field transform domain", in *Proceedings 8th International Symposium Communication Theory and Applications*, Ambleside, Lake District, UK, pp. 194–199.

3.1 Preliminaries

An iterative decoder may be viewed as a bank of soft-input soft-output (SISO) decoders. Each of the SISO decoder produces some reliability information which can then be used by the other SISO decoders in an iterative manner to converge to a solution. Each SISO decoder takes in some probabilities, known as the a-priori-probabilities (AP), and outputs some probabilities, known as the a-posteriori-probabilities (APP).

3.1 Definition (Extrinsic information). The ratio of the a-posteriori-probabilities and the a-priori-probabilities is defined as extrinsic information.

According to Ambroze et al. (2000), the iterative decoding algorithm is mathematically equivalent to the iterative method of solving the following equations

$$P_e^1 = f(P_e^2)$$
$$P_e^2 = g(P_e^1).$$

In the case of belief-propagation iterative decoder of LDPC codes, the function f may be represented as the message passing from the check to variable vertices, the function g as the message passing from the variable to check vertices, and P_e^i is the extrinsic information from the ith function.

Consider an $[n, k, d]_q$ linear code C, let $y = (y_0, y_1, \ldots, y_{n-1})$, where $y_i \in \mathbb{R}$, be the n-th dimensional vector representing the received samples.

3.2 *Definition (Maximum likelihood decoder).* A maximum likelihood decoder produces a codeword, say $\hat{c}$, which has the highest probability of being correct compared to the remaining $q^k - 1$ codewords for a given received sample y, i.e.

$$\hat{c} = \underset{\forall c \in C}{\operatorname{argmax}} \Pr(c|y).$$

Following Bayes rule, $\Pr(c|y)$ may be written as

$$\Pr(c|y) = \frac{\Pr(c)}{\Pr(y)} \Pr(y|c).$$

Assuming a discrete memoryless channel, $\Pr(c) = q^{-k}$ and the term $\Pr(y)$ is a normalising constant. Therefore, finding a codeword that maximises $\Pr(c|y)$ is equivalent to finding one that maximises $\Pr(y|c)$.

In dealing with a hard decision decoder, the distance metric of interest is the Hamming distance, see Definition 1.5, whereas with a soft decision decoder, entities of $\mathbb{R}$ are considered and Euclidean distance is a more important parameter.

3.3 *Definition (Euclidean Distance).* The Euclidean distance of two n-tuple vectors, x and y, denoted as $\mathrm{d}_E(x, y)$, is given by

$$\mathrm{d}_E(x, y) = \frac{1}{n}\sqrt{\sum_{i=0}^{n-1}(x_i - y_i)^2}.$$

Following Definition 1.1, a codeword c may be represented as an n-tuple vector. For transmission purposes, the codeword c is mapped to a real vector $\psi(c) = (\psi(c_0), \psi(c_1), \ldots, \psi(c_{n-1}))$ and assuming discrete memoryless AWGN channel with zero mean and variance σ^2, $\Pr(y|c)$ can be written as

$$\Pr(y|c) = \prod_{i=0}^{n-1} \Pr(y_i|c_i) = \prod_{i=0}^{n-1} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(y_i - \psi(c_i))^2}{2\sigma^2}\right). \tag{3.1}$$

Since $\exp(x)$ is a decreasing function of x, maximising $\Pr(y|c)$ is also equivalent to minimising

$$\sum_{i=0}^{n-1}(y_i - \psi(c_i))^2$$

in the logarithmic domain. Thus, a maximum likelihood solution is the codeword with the minimum Euclidean distance with respect to the received sample y.

It is useful to define the term more likely codeword in order to characterise the convergence of an iterative decoder in relation to maximum likelihood solution.

3.4 Definition (More Likely or mrl Codeword). Let c be a codeword of the code C, which is then modulated to $\psi(c)$ prior transmission. Due to noise corruption, it is received as a vector y and the decoder produces a codeword, c', where $c \neq c'$. On the other hand, $\mathrm{d}_E(y, \psi(c')) < \mathrm{d}_E(y, \psi(c))$. The codeword c' is defined as the more likely codeword.

An iterative decoder, at the end of an iteration, does not always produce a solution which is an element of the code C. Given the transmitted codeword c and received vector y, the output of an iterative decoder, say z, may be categorised as follows

non convergent block: $z \notin C$,

correct codeword: $z = c$,

non mrl codeword: $z \in C$ and $\mathrm{d}_E(y, \psi(z)) > \mathrm{d}_E(y, \psi(c))$,

mrl codeword: $z \in C$ and $\mathrm{d}_E(y, \psi(z)) < \mathrm{d}_E(y, \psi(c))$.

In evaluating the performance of a code numerically, the number of mrl codewords provides significant information on the performance of an iterative decoder. A maximum likelihood decoder produces either a correct codeword or an mrl codewords and thus, the percentage of mrl codewords produced by an iterative decoder gives a performance indication of how close the iterative decoder is from the optimum maximum likelihood performance for the same code. If the iterative decoder produces an mrl codeword c_1, then a maximum likelihood decoder is guaranteed to produce an mrl codeword c_{ML} and hence a decoding error, where

$$\mathrm{d}_E(y, \psi(c_{ML})) \leq \mathrm{d}_E(y, \psi(c_1)) < \mathrm{d}_E(y, \psi(c)),$$

i.e. the codeword c_{ML} is not necessary the same as c_1. Frame errors due to mrl codewords provide a lower-bound on the maximum-likelihood performance of a code.

3.2 Investigation on the Hartmann-Rudolph Decoding Algorithm

Consider a communication system employing channel coding and binary antipodal signalling, given a received vector which is assumed to be made of a transmitted sequence added with Gaussian noise, the job of the decoder in the receiving end is to determine what the transmitted codeword was with minimum probability of making error. Recall that there are two types of optimum decoding algorithm: one that minimises the probability of codeword error and the other which minimises the probability of symbol error. The former type of optimum decoder, which is commonly known as the maximum likelihood decoder, includes the Wagner* algorithm for $[n, n-1, 2]$ single parity-check codes which is the earliest type of soft decision decoder known, and the well-known Viterbi (1967) algorithm. The latter type of optimum decoding algorithm, which minimises symbol error probabilities, includes the BCJR algorithm by Bahl et al. (1974) and the Hartmann and Rudolph

*Description of the Wagner algorithm is available in the paper by Silverman and Balser (1954).

(1976) algorithm. Given a code C, while the optimum BCJR algorithm produces the optimum decision based on the codewords of C, the Hartmann-Rudolph (HR) algorithm is based on the codewords of $C^{\perp}$, the dual of C. As a consequence, the HR decoding algorithm is suitable for high-rate codes where the number of codewords $C^{\perp}$ are much less than those of C.

The optimal HR decoding algorithm for linear code over $\mathbb{F}_q$ aims to minimise the symbol error probabilities and this is equivalent to maximising the probabilities of a symbol being correct, i.e.

$$\underset{s\in\mathbb{F}_q}{\operatorname{argmax}} \Pr(c_i = s|y) \qquad \text{for } i = 0, 1, \ldots, n-1$$

where $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ is a codeword of C and y is the received vector. The probabilities of a symbol c_i being $s \in \mathbb{F}_q$ conditioned to a received vector y is given by

$$\Pr(c_i = s|y) = \sum_{c_i=s,\ \mathbf{c}\in C} \Pr(\mathbf{c}|y). \tag{3.2}$$

Following Bayes rule, (3.2) can be expressed as

$$\begin{aligned}\Pr(c_i = s|y) &= \sum_{c_i=s,\ \mathbf{c}\in C} \frac{\Pr(y|\mathbf{c})\Pr(\mathbf{c})}{\Pr(y)} \\ &= \frac{q^{-k}}{\Pr(y)} \sum_{\mathbf{c}\in C} \Pr(y|\mathbf{c})\delta_{s,\mathbf{c}\cdot\mathbf{e}_i}\end{aligned} \tag{3.3}$$

where $\Pr(\mathbf{c})$ has been replaced with q^{-k} due to the memoryless channel assumption, $\mathbf{e}_i$ is a vector of all zeros except at position i where it is a one and $\delta_{i,j}$ is the Kronecker symbol, defined by

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathbf{c}\cdot\mathbf{e}_i$ in (3.3) is the standard dot product of codeword $\mathbf{c}$ and vector $\mathbf{e}_i$, i.e. $\sum_{m=0}^{n-1} c_m \cdot e_{i,m}$. It is shown by Hartmann and Rudolph (1976) that (3.3) is equivalent to

$$\Pr(c_i = s|y) = \frac{q^{-n}}{\Pr(y)} \sum_{\mathbf{v}\in C^{\perp}} \Pr(y_i|s)\omega^{s\cdot v_i} \prod_{\substack{m=0\\ m\neq i}}^{n-1} \sum_{t=0}^{q-1} \Pr(y_m|t)\omega^{t\cdot v_m} \tag{3.4}$$

where $\omega = e^{j2\pi/q}$, the complex nth root of unity. For codes over $\mathbb{F}_2$, as shown by Hartmann and Rudolph (1976), the detection rule given may be simplified to

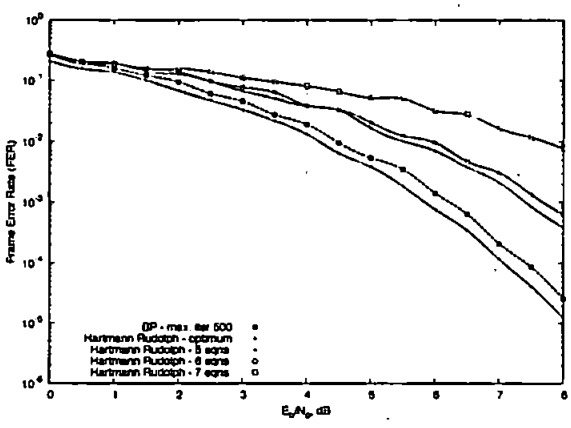
$$\hat{c}_i = \begin{cases} 0 & \text{if } \displaystyle\sum_{\mathbf{v}\in C^{\perp}} \prod_{m=0}^{n-1} \left[\frac{1-\phi_i}{1+\phi_i}\right]^{v_m+\delta_{i,m} \pmod 2} > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{3.5}$$

for $0 \le i \le n-1$, where $\phi_i = \Pr(y_i|1)/\Pr(y_i|0)$ and $\hat{\mathbf{c}} = (\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_{n-1})$ is the output of the HR decoder.

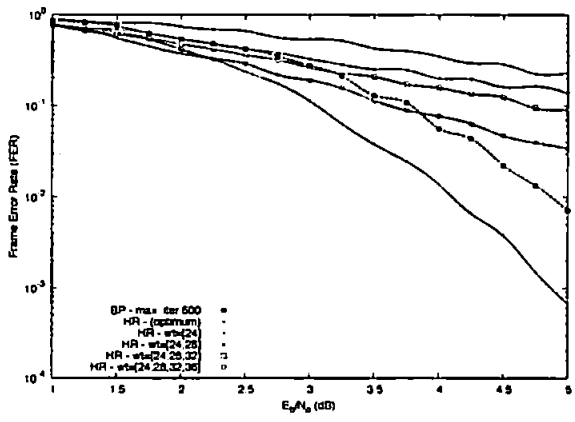
Following (3.5), the HR decoder requires the entire codewords of $C^{\perp}$ in order to arrive at the

optimum solution. The sum-product algorithm of Gallager (1963) for decoding LDPC codes, which is equivalent to the belief propagation algorithm of Pearl (1988), may be viewed as an iterative algorithm to generate all codewords of $C^{\perp}$ to approximate the optimum HR solution. Instead of using all q^{n-k} codewords of the dual code of an $[n, k, d]$ linear code over $\mathbb{F}_q$, the sum-product iterative decoding algorithm considers the codewords of the $n-k$ single parity-check subcodes of the dual code. This iterative approximation, however, introduces some n-dimensional vectors, which are not codewords of C, and as a consequence, there is a gap between the performance of the HR algorithm and that of the sum-product iterative decoding algorithm.

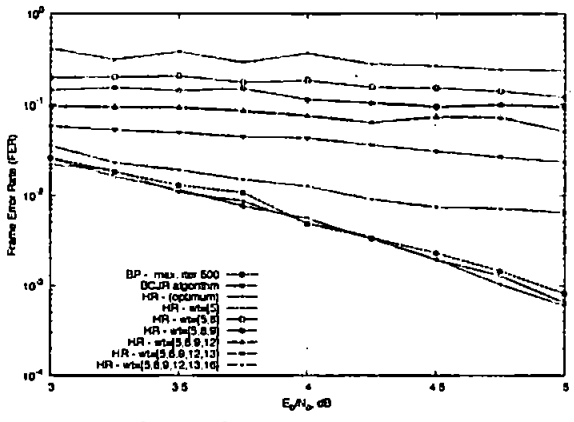
The complexity of the HR decoder is on the enumeration of all codewords of $C^{\perp}$. For an $[n, k, d]$ code over $\mathbb{F}_q$, there are q^{n-k} codewords of the dual and as such, applying the HR algorithm is prohibitive for almost all good linear codes. It is, therefore interesting to investigate the applicability of using some subset of the dual code codewords and also to determine how much the associated coding loss is. The sum-product iterative decoder resembles the HR decoder in a way that both decoders make use of the dual code codewords. While the HR algorithm uses the entire dual code codewords, the sum-product algorithm uses the lowest weight dual code codewords only. Investigation on whether or not the gap between iterative and optimum decoders can be reduced by aiding the iterative decoder with HR algorithm is of interest and worth doing.



(a) $[7, 4, 3]$ Hamming code



(b) $[63, 51, 5]$ BCH code



(c) $[21, 11, 6]$ difference set cyclic code

Figure 3.1: Hartmann Rudolph decoding algorithm: optimum and non optimum performance

Figure 3.1 compares the FER performance of the sum-product iterative decoder and that of the HR decoder–using the entire codewords of the dual or some subset of them. The weight enumerator functions of the codes considered in Figure 3.1 are

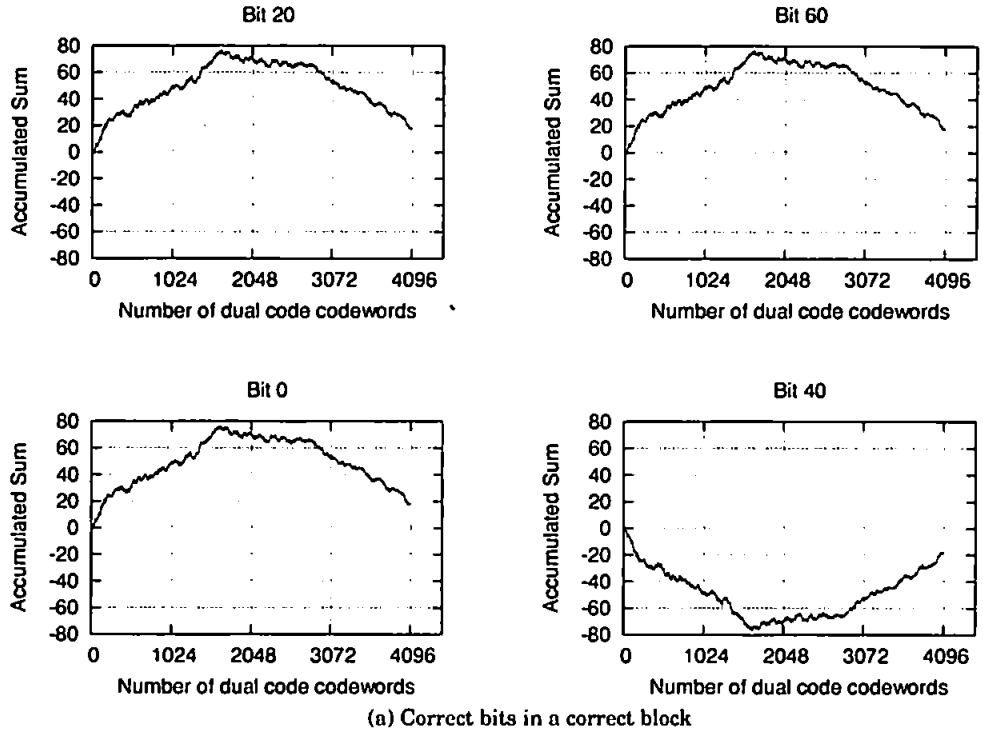
a) $[7,4,3]$ Hamming code: $A(z) = 1 + 7x^7$,

b) $[63,51,5]$ BCH code: $A(z) = 1 + 24x^{210} + 28x^{1512} + 32x^{1071} + 36x^{1176} + 40x^{126}$, and

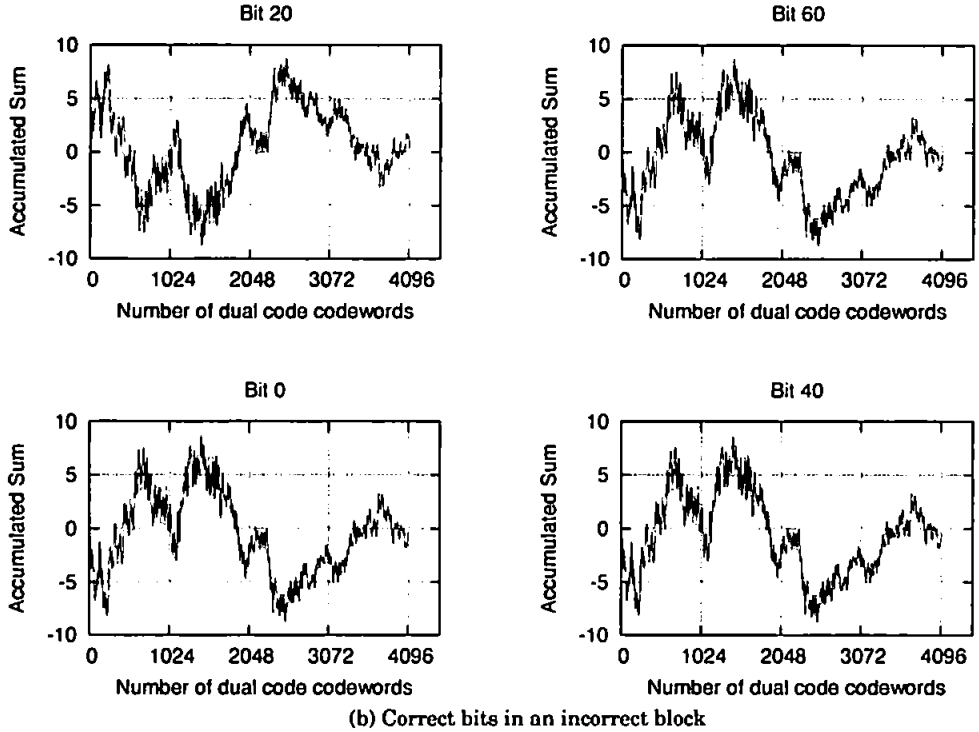
c) $[21,11,6]$ difference set cyclic code: $A(z) = 1 + 168x^6 + 210x^8 + 1008x^{10} + 280x^{12} + 359x^{14} + 21x^{16}$.

It is clearly demonstrated in Figure 3.1 that, unless the entire dual code codewords are considered, the performance of the HR algorithm is poor. The $[7,4,3]$ Hamming code (Figure 3.1a) is a good example. The dual code of this Hamming code is a $[7,3,4]$ cyclic code consisting of a codeword of weight 0 and 7 codewords of weight 7. The performance attained by omitting a single dual code codeword only is around 1.0 dB and more than 1.5 dB away from the iterative and optimum solutions, respectively. It is surprising that a codeword can makes such a significant coding gain and the answer can be explained by looking at Figure 3.2. This figure shows the plot of the summation in (3.5) against the number of dual code codewords used for the correct bit at positions 0, 20, 40, and 60, in both correct and incorrect blocks, at $E_b/N_o = 5.0$ dB. In the case of correct bits in a correct block, the sign of the summation does not change against the number of dual code codewords used. On the other hand, the sign of this summation for a correct bit fluctuates from positive to negative and vice-versa in an incorrect block. Also note that, in the case of an incorrect block, the maximum and minimum values of the summation are much lower than those in the correct block case. From (3.5), a bit is decoded as 0 if the summation is a positive real number and 0 otherwise. Due to the fluctuation as shown in Figure 3.2, especially for bits in an incorrect block, a single codeword can change the sign of the summation from one to another.

A way of improving the iterative decoder by extending the parity-check matrix of an LDPC code has also been investigated. A standard parity-check matrix of an LDPC code contains the low weight codewords of its dual. This parity-check matrix is extended so that it contains some higher weight codewords of the dual code. Initially, the belief propagation algorithm is invoked using the standard parity-check matrix; in the event of non convergence block, the HR algorithm is invoked using the additional parity-check equations of higher weight. However, the numerical results show that the joint iterative and Hartmann-Rudolph decoding scheme does not provide any improvement to the iterative decoder and this is attributed to the fluctuations as shown in Figure 3.2. This investigation, however, produces a different approach to improve the iterative decoder, which is discussed in Section 3.3.



(a) Correct bits in a correct block



(b) Correct bits in an incorrect block

Figure 3.2: Behaviour of the Hartmann-Rudolph decoding algorithm for $[63, 51, 5]$ BCH code at $E_b/N_o = 5.0$ dB

3.3 Codeword-Substitution Belief Propagation Algorithm

For linear codes, $c \cdot c^{\perp} = 0$ for every $c \in C$ and $c^{\perp} \in C^{\perp}$. The standard belief propagation algorithm for decoding LDPC codes extracts extrinsic information from the parity-check matrix which contains $n-k$ low weight parity-check equations†. Since $|C^{\perp}| = q^{n-k}$, one simple way to improve the performance of the belief propagation iterative decoder is to use a parity-check matrix that contains a larger set of parity-check equations. This parity-check matrix will have, in addition to the low weight codewords, higher weight codewords of $C^{\perp}$. The resulting parity-check matrix is dense and this causes performance degradation of the belief propagation algorithm.

Let H be the parity-check matrix of C. The Codeword-Substitution belief propagation algorithm does not extend the number of parity-check equations in H. Instead, a set of parity-check equations H^c is generated by taking the linear combinations of those equations in H. A subset of H^c is substituted into H to generate a modified parity-check matrix $\hat{H}$, which is then used to extract extrinsic information as in the standard belief propagation iterative decoding. The decoding procedures may be described in Algorithm 3.1.

Algorithm 3.1 Codeword-Substitution Belief Propagation Algorithm

Input:
- $y \Leftarrow$ received vector
- $H \Leftarrow$ original parity-check matrix of the code
- $H^c \Leftarrow$ a set of parity-check equations not in H
- $\mathcal{T} \Leftarrow$ number of trials
- $\mathcal{S} \Leftarrow$ number of selections

Output: a codeword with the minimum Euclidean distance

1: Perform belief propagation decoding
2: Let $d_0 \Leftarrow$ decoded output.
3: $\mathrm{d}_E(y, \psi(d_0)) \Leftarrow$ Euclidean distance between d_0 and y.
4: $d' \Leftarrow d_0$
5: $d_E^{min} \Leftarrow \mathrm{d}_E(y, \psi(d_0))$
6: **for** $\tau = 1$ to $\mathcal{T}$, **do**
7: **for** $i = 1$ to maximum number of iterations, **do**
8: Pick $\mathcal{S}$ parity-check equations from H^c.
9: Substitute them into H to generate $\hat{H}$.
10: Based on $\hat{H}$, perform the check nodes (horizontal) and bit nodes (vertical) processing as in standard belief propagation decoding algorithm,
11: $d_\tau \Leftarrow$ denote the decoded output,
12: $\mathrm{d}_E(y, \psi(d_\tau)) \Leftarrow$ Euclidean distance between d_τ and y.
13: **if** $d_\tau H^T = 0$ **then**
14: Stop the algorithm and goto Step 17
15: **end if**
16: **end for**
17: **if** $\left(\mathrm{d}_E(y, \psi(d_\tau)) < d_E^{min}\right)$ and $\left(d_\tau H^T = 0\right)$ **then**
18: $d' \Leftarrow d_\tau$ and $d_E^{min} \Leftarrow \mathrm{d}_E(y, \psi(d_\tau))$
19: **end if**
20: **end for**
21: Output d'.

The selection of the candidate parity-check equations for substitution may be made on a ran-

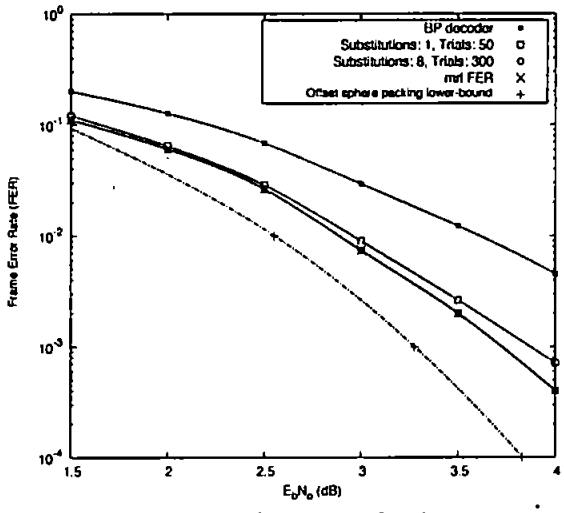
†Note that for cyclic LDPC codes, there are n low weight parity-check equations.

dom basis or may correspond to a predetermined sequence. To evaluate the performance of this algorithm, computer simulations have been run using a class of cyclic codes whose parity-check equations are orthogonal in each position. The properties and the construction of this class of codes are described in Chapter 2. It is assumed that the channel is perturbed by additive white Gaussian noise and binary antipodal signalling is employed, which has a function ψ mapping the symbols 0 and 1 to -1 and $+1$ respectively. Figure 3.3 compares the FER performance of the standard belief propagation and that of the Codeword-Substitution belief propagation decoders on the $[63, 37, 9]$, $[93, 47, 8]$ and $[105, 53, 8]$ cyclic LDPC codes. It can be seen that the Codeword-Substitution decoder, provided enough substitutions and trials are carried out, can achieve maximum likelihood performance as all the frame errors observed are due to mrl codewords. Figure 3.3a shows that, for the $[63, 37, 9]$ cyclic code, the Codeword-Substitution decoder offers a gain of approximately 0.9 dB over the standard decoder and at 10^{-3} FER the performance is within 0.4 dB away from the offset sphere packing lower bound. Table 3.1 shows how close the performance of the Codeword-Substitution decoder is to the maximum likelihood decoder for $[63, 37, 9]$ cyclic LDPC code. Using the standard decoder, more than 50% mrl codewords are found in the low SNR region, but only a few mrl codewords is found in moderate SNR region. With just single substitution and 50 trials, the Codeword-Substitution decoder is able to increase the percentage of mrl codewords significantly. The maximum likelihood performance is achieved with 8 substitutions and 300 trials. Figure 3.3 also shows that the Codeword-Substitution decoder also achieves the maximum likelihood performance for the $[93, 47, 8]$ and $[105, 53, 8]$ cyclic codes. The Codeword-Substitution decoder is approximately 1.1 dB and 2.0 dB better than the standard decoder for the $[93, 47, 8]$ and $[105, 53, 8]$ codes respectively. In addition, the gaps from the offset sphere packing lower bound at 10^{-3} FER are respectively 0.8 dB and 0.9 dB.

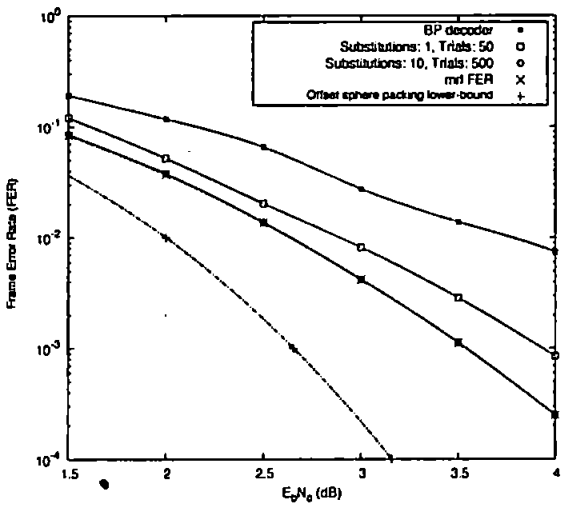
Substitution of the low weight parity-check equations with those of higher weight will inevitably introduce many short cycles in the graph of the LDPC code. However, these cycles do not pose a lasting negative effect on the belief propagation decoder as the substitutions are done at every iteration.

Standard belief propagation decoder						
E_b/N_o, dB	1.5	2.0	2.5	3.0	3.5	4.0
%	73	41	27	23	16	9
Substitutions: 1, Trials: 50						
E_b/N_o, dB	1.5	2.0	2.5	3.0	3.5	4.0
%	90	94	90	82	74	61
Substitutions: 8, Trials: 300						
E_b/N_o, dB	1.5	2.0	2.5	3.0	3.5	4.0
%	100	100	100	100	100	100

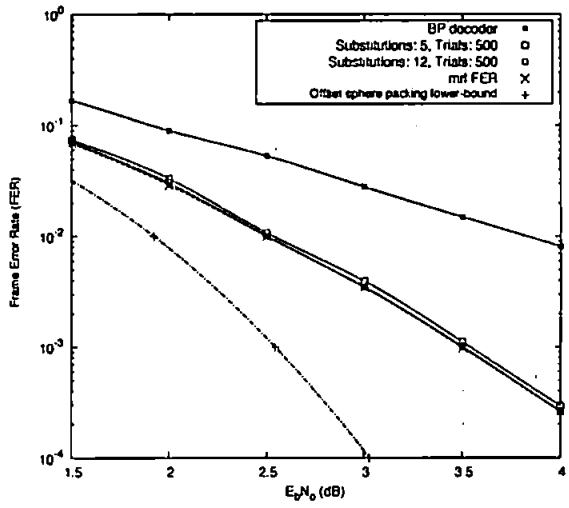
Table 3.1: E_b/N_o against mrl codewords for the $[63, 37, 9]$ cyclic LDPC code



(a) [63, 37, 9] cyclic LDPC code



(b) [93, 47, 8] cyclic LDPC code



(c) [105, 53, 8] cyclic LDPC code

Figure 3.3: FER of the Codeword-Substitution belief propagation decoder

3.4 Other Approaches to Improve the Convergence of Iterative Decoder

There are other approaches that can improve the performance of belief propagation iterative decoder. In this section, two of such methods are described.

3.4.1 Grouping of the Parity-Check Equations

Recall that an LDPC code of length n and dimension k may be seen as a concatenation of $n-k$ single parity-check $[n, n-1, 2]$ codes. The belief propagation iterative decoder, in each iteration, derives extrinsic information independently from each of the single parity-check codes. The reliability of the extrinsic information may be improved by using stronger component codes and this may be achieved by grouping m rows, for $m > 1$, of the parity-check matrix. As a result of the grouping, the LDPC code is seen as a concatenation of $\lceil \frac{n-k}{m} \rceil - 1$ subcodes of length n and dimension $n-m$ and one subcode of length n and dimension $m - m\lceil \frac{n-k}{m} \rceil + (n-k)$. Figure 3.4 compares the FER performance of standard iterative decoding and the one which groups $m = 5$ parity-check equations for a $[1024, 768]$ irregular LDPC code. From this figure, it can be seen that grouping improves the FER performance, however, the gain obtained is negligible compared to the complexity involved.

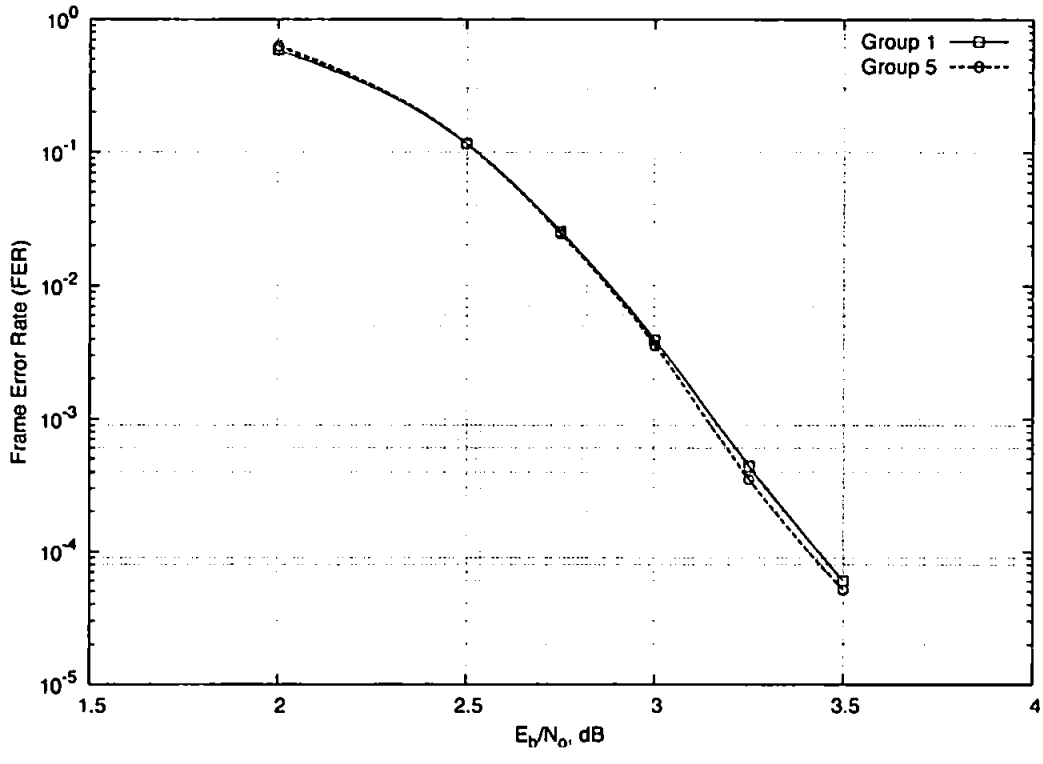


Figure 3.4: Performance improvement by grouping the component codes

3.4.2 The Received Vector Coordinate Modification Algorithm

The received vector coordinate modification (RVCM) algorithm was initially introduced by Papagiannis et al. (2003a). The authors showed that by employing the RVCM algorithm to iteratively decode serial concatenated convolutional codes of low memory, a gain of 0.2 dB in the waterfall region was achieved. Subsequent work on this algorithm appeared in Papagiannis et al. (2004) which shows that, using the same concatenated arrangement, the algorithm can achieve the maximum likelihood performance of serial concatenated convolutional codes in the high signal-to-noise-ratio region; and also in Papagiannis, Ambroze, Tomlinson and Ahmed (2005) which analyses this algorithm in decoding LDPC codes. Despite the RVCM algorithm was shown to give improvement on binary linear codes, the algorithm is applicable to any iteratively decodable $[n, k, d]_q$ linear codes. Other decoding algorithms that are similar to the RVCM algorithm include the work of Pishro-Nik and Fekri (2003) and Varnica and Fossorier (2004).

3.5 Summary

- The Hartmann-Rudolph decoder is an optimum symbol-by-symbol decoder based on the dual of a linear code. The Hartmann-Rudolph decoder requires all codewords of the dual code in order to arrive at optimum solution. It has not been possible to obtain an acceptable suboptimum performance of this decoder by using some subset of the dual code codewords.

- The performance of the belief propagation iterative decoder may be improved by using any subset of $n - k$ (or n in the case of cyclic codes) dual code codewords as the parity-check equations in each iteration. The belief propagation decoder is modified such that in the beginning of each iteration some parity-check equations are substituted with the higher weight dual code codewords and they are reverted at the end of each iteration. For some short cyclic codes this modified decoder, which is called the Codeword-Substitution belief propagation decoder, results in maximum likelihood performance.

- Grouping the parity-check equations in the parity-check matrix of an LDPC code results in stronger component codes. Grouping can improve the performance of iterative decoder, however, the gain obtained is negligible compared to the complexity involved.

Part III

Classical Coding

4 Good Binary Linear Codes

Parts of this chapter are published as the following journal papers:

1. Tjhai, C. and Tomlinson, M. (2007), "Results on binary cyclic codes", *Electronics Letters*, **43**(4), pp. 234–235

2. Tjhai, C., Tomlinson, M., Grassl, M., Horan, R., Ahmed, M. and Ambroze, M. (2006), "New linear codes derived from cyclic codes of length 151", *IEE Proceedings Communincations*, **153**(5), pp. 581–585.

4.1 Introduction

As mentioned in the beginning of this thesis, the quest to Shannon's limit has been approached in two different ways. In the previous part of this thesis, iterative decoding approach, which relies on long codes with low complexity decoder, was studied. In this part of the thesis, we consider the problem formulated by Shannon as that of construction of good codes which maximises the asymptotic coding gain, see (1.11). In this approach, it is assumed that a decoding algorithm always exists for any constructed code.

Computing the minimum Hamming distance of a linear code is, in general, a Nondeterministic Polynomial-time (NP) hard problem, as conjectured by Berlekamp et al. (1978) and later proved by Vardy (1997). Nowadays, it is a common practice to use multi-threaded algorithm which runs on multiple parallel computers (grid computing) for minimum Hamming distance evaluation. Even then, it is not always possible to evaluate the exact minimum Hamming distance for large codes. For some algebraic codes, however, it is possible to obtain the lower- and upper-bounds on this distance. But knowing these bounds are not sufficient as the whole idea is to know explicitly the exact minimum Hamming distance of a constructed code. As a consequence, algorithms for evaluating the minimum Hamming distance of a code is of utmost important in this subject and they are described in the following section.

It is worth mentioning that a more accurate benchmark of how good a code is, is in fact its Hamming weight distribution. While computing the minimum Hamming distance of a code is in general NP-hard, computing the Hamming weight distribution of a code is even more complex. In general, for two codes of the same length and dimension but of different minimum Hamming distance, it can be considerably certain that the one with higher distance is superior to the other one. Unless we are required to decide between two codes of the same parameters, it is not necessary to go down the root of evaluating their Hamming weight distributions.

Since in this chapter, we deal exclusively with the set of binary vector of a given length, i.e. Hamming space, when the terms distance and weight are used, we shall mean the Hamming distance and the Hamming weight respectively.

4.2 Algorithms to Compute the Minimum Distance of Binary Linear Codes

4.2.1 The First Approach to Minimum Distance Evaluation

For a $[n, k, d]$ linear code over $\mathbb{F}_2$ with a reduced-echelon generator matrix $G_{sys} = [I_k|P]$, where I_k and P are $k \times k$ identity and $k \times (n-k)$ matrices respectively, a codeword of this linear code can be generated by taking a linear combination of some rows of G_{sys}. Since, by Definition 1.6, the minimum distance of a code is the minimum non zero weight among all of the 2^k codewords, the brute-force method to compute the minimum distance would be to generate codewords by taking

$$\binom{k}{1}, \binom{k}{2}, \binom{k}{3}, \ldots, \binom{k}{k-1}, \text{ and } \binom{k}{k}$$

linear combinations of the rows in G_{sys}, note on the weight of each codeword generated and return the minimum weight among these $2^k - 1$ codewords. This method gives not only the minimum distance, but also the weight distributions, of a code. It is obvious that, as k grows large, this method becomes infeasible. Having said that, if $n - k$ is not that large, the minimum distance can still be obtained by evaluating the weight distributions of the $[n, n-k, d']$ dual code and returning the corresponding MacWilliams Identity.

It is clear that there are too many unnecessary codeword enumerations involved in the above approach. A better approach which avoids enumerating large amount of unnecessary codewords can be devised. Let $c = (i|p) = (c_0, c_1, \ldots, c_{k-1}|c_k, \ldots, c_{n-2}, c_{n-1})$ be a codeword of a binary linear code of minimum distance d. Let $c' = (i'|p')$ be a codeword of weight d, then if $\mathrm{wt}_H(i') = w$ for some integer $w < d$, $\mathrm{wt}_H(p') = d - w$. This means that at most

$$\sum_{w=1}^{\min\{d-1,k\}} \binom{k}{w} \tag{4.1}$$

codewords are required to be enumerated. In practice, d is unknown and an upper-bound d_{ub} on the minimum distance is required during the evaluation and the minimum Hamming weight found thus far can be used for this purpose. It is clear that once all $\sum_{w'=1}^{w} \binom{k}{w'}$ codewords of information weight w' are enumerated,

- it is known that all possibilities of $d \leq w$ have been considered; and
- if $w < d_{ub}$, it is also known that the minimum distance of the code is at least $w + 1$.

Therefore, as well as having an upper-bound, a lower-bound $d_{lb} = w + 1$ on the minimum distance can also be obtained. The evaluation continues until the condition $d_{lb} \geq d_{ub}$ is met and in this event, d_{ub} is the minimum Hamming distance.

4.2.2 Brouwer's Algorithm for Linear Codes

There is an apparent drawback of the above approach. In general, the minimum distance of a low rate linear code is greater than its dimension. This implies that $\sum_{w=1}^{k} \binom{k}{w}$ codewords would need

to be enumerated. A more efficient algorithm was attributed to Brouwer* and the idea behind this approach is to use a collection of generator matrices of mutually disjoint information sets (Grassl; 2006).

4.1 ***Definition (Information Set).*** Let the set $S = \{0, 1, 2, \ldots, n-\}$ be the coordinates of an $[n, k, d]$ binary linear code with generator matrix G. The set $\mathcal{I} \subseteq S$ of k elements is an information set if the corresponding coordinates in the generator matrix is linearly independent and the submatrix corresponding to the coordinates in $\mathcal{I}$ has rank k, hence it can be transformed into a $k \times k$ identity matrix.

In the other words, it can be said that, in relation to a codeword, the k symbols user message is contained at the coordinated specified by $\mathcal{I}$ and the redundant symbols are stored in the remaining $n - k$ positions. An information set corresponds to a reduced echelon generator matrix and it may be obtained as follows. Starting with a reduced-echelon generator matrix $G_{sys}^{(1)} = G_{sys} = [I_k|P]$, Gaussian-elimination is applied to submatrix P so that it is transformed to a reduced echelon form. The resulting generator matrix now becomes $G_{sys}^{(2)} = [A|I_k|P']$, where P' is a $k \times (n - 2k)$ matrix. Next, submatrix P' is put into reduced echelon form and the process continue until there exists a $k \times (n - lk)$ submatrix of rank less than k, for some integer l. Note that column permutation may be necessary during the transformation to maximise the number of disjoint information sets.

Let $\mathcal{G}$ be a collection of m reduced echelon generator matrices of disjoint information sets, $\mathcal{G} = \left\{G_{sys}^{(1)}, G_{sys}^{(2)}, \ldots, G_{sys}^{(m)}\right\}$. Using these m matrices means that, after $\sum_{w'=1}^{w} \binom{k}{w'}$ enumerations,

- all possibilities of $d \leq mw$ have been considered; and
- if $mw < d_{ub}$, the minimum distance of the code is at least $m(w + 1)$, i.e. $d_{lb} = m(w + 1)$.

In this case, the lower-bound has been increased by a factor of m, instead of 1 compared to the previous approach. For $w \leq k/2$, $\binom{k}{w} \gg \binom{k}{w-1}$ and as a consequence, this lower-bound increment reduces the bulk of computations significantly. If d is the minimum distance of the code, the total number of enumerations required is given by

$$\sum_{w=1}^{\min\{\lceil d/m \rceil - 1, k\}} m \binom{k}{w}. \tag{4.2}$$

Disjoint information sets

Example 4.1: Consider the $[55, 15, 20]_2$ optimal binary linear–a shortened code of the Goppa code discovered by Loeloeian and Conan (1984). The reduced echelon generator matrices of disjoint information sets are given by

$$G_{sys}^{(1)} = \left[\begin{array}{c|c}
100000000000000 & 0101101111101010011101010101010100000000 \\
010000000000000 & 1000111001101011110100100111001000110000 \\
001000000000000 & 1011001111011010111010111111001001000001 \\
000100000000000 & 1010101011101101101111110100010010101010 \\
000010000000000 & 0011110100111110110110100011000101011101 \\
000001000000000 & 0101000010100101111110001110001010001110 \\
000000100000000 & 1001001110100010100110011001010010100111 \\
000000010000000 & 1011000100110100000001110010110011110101 \\
000000001000000 & 1010110101111110101001001011101110100001 \\
000000000100000 & 1010000101001011000001110101111101100010 \\
000000000010000 & 1101101011110001001011111011100101010100 \\
000000000001000 & 1101101110101111110011101111110000011011 \\
000000000000100 & 0000010001010011101110010011001011101101 \\
000000000000010 & 0101100000011101011110010001100111000011 \\
000000000000001 & 0011111001011000111001010001000000111011
\end{array}\right],$$

*Zimmermann (1996) attributed this algorithm to Brouwer as private communication.

$$G_{sys}^{(2)} = \left[\begin{array}{c|c|c}
1011010010110 01 & 100000000000000 & 001111001111011111110110000 \\
0000000110001 10 & 010000000000000 & 101100011101111100001111010 \\
0011111001000 11 & 001000000000000 & 001110100101001111000001 01 \\
0101101001111 01 & 000100000000000 & 110010100101010111001101 1 \\
1111110010001 00 & 000010000000000 & 011001100110010110100010 0 \\
1111100101010 01 & 000001000000000 & 000000010001111111001100 01 \\
1111001000111 10 & 000000100000000 & 101001011011010110001110 1 \\
0000011001111 11 & 000000010000000 & 110100010010101110001000 1 \\
0000001010000 01 & 000000001000000 & 111011000001111111001111 01 \\
1110011001001 00 & 000000000100000 & 110010011110011101101010 111 \\
1000111111001 111 & 000000000010000 & 010000110010000100010101 110 \\
0101100001101 11 & 000000000001000 & 110111010110110101011100 100 \\
0010110111111 11 & 000000000000100 & 010101001100101111111110 \\
1001010110010 11 & 000000000000010 & 011000011100100011001001 1 \\
1101001011101 01 & 000000000000001 & 001010001100010000111110 0
\end{array}\right],$$

and

$$G_{sys}^{(3)} = \left[\begin{array}{c|c|c}
0100101001101000101111100011 10 & 100000000000000 & 1100010001 \\
1111010111100001111101111100 11 & 010000000000000 & 1000110111 \\
0111011011110010110011101101 00 & 001000000000000 & 1101000101 \\
0011000100101010100100011111 01 & 000100000000000 & 0110011100 \\
0001111011010101101101101010 11 & 000010000000000 & 1011000101 \\
1011101010010001001101010010 00 & 000001000000000 & 0110101011 \\
0101010001101000000101111001 10 & 000000100000000 & 0011001111 \\
1001011011101111100001101101 010 & 000000010000000 & 0100000001 \\
1001001101010111100111100110 00 & 000000001000000 & 1101111001 \\
0110011000111001111101110111 11 & 000000000100000 & 0111111010 \\
1011100011111100011101101101 111 & 000000000010000 & 0111010111 \\
1000011011010100001011101101 10 & 000000000001000 & 1000110010 \\
0111010100000100010111011010 00 & 000000000000100 & 1101100011 \\
0100111110100011000100010110 01 & 000000000000010 & 1101111111 \\
1101001111001110011001111011 01 & 000000000000001 & 0000011110
\end{array}\right].$$

Brouwer's algorithm requires 29844 codewords to prove the minimum distance of this code. Meanwhile, for the same proof, 32767 codewords would be needed if only one generator matrix is employed.

4.2.3 Zimmermann's Algorithm for Linear Codes and Some Improvements

A further refinement to the minimum distance algorithm was due to Zimmermann (1996). Similar to Brouwer's approach, a set of reduced-echelon generator matrices are required. While in Brouwer's approach the procedure is stopped once a non-full rank submatrix is reached, Zimmermann's approach proceed further to obtain submatrices with overlapping information sets. Let $G_{sys}^{(m)} = [A_m | I_k | B_{m+1}]$ be the last generator matrix which contains a disjoint information set, to obtain matrices with overlapping information sets, Gaussian-elimination is performed on the submatrix B_{m+1} and this yields

$$G_{sys}^{(m+1)} = \left[\begin{array}{c|c|c|c}
\hat{A}_m & 0 & I_{r_{m+1}} & B_{m+2} \\
\hline
 & I_{k-r_{m+1}} & 0 &
\end{array}\right],$$

where $r_{m+1} = \text{Rank}(B_{m+1})$. Next, $G_{sys}^{(m+2)}$ is produced by running Gaussian-elimination on the submatrix B_{m+2} and so on until all the n coordinates have been exhausted.

From $G_{sys}^{(3)}$ of Example 4.1, it can be seen that the last 10 coordinates do not form an information set since the rank of this submatrix is clearly less than k. Nonetheless, a "partial" reduced echelon

generator matrix can be obtained from $G_{sys}^{(3)}$,

$$G_{sys}^{(4)} = \left[\begin{array}{c|c|c}
011111100101101001010011000110010101010000 & 00000 & 1000000000 \\
110011000001011000001011111000011010111 0 & 00000 & 0100000000 \\
101001000011000101110011100111010000111 0 & 00000 & 0010000000 \\
001001001000110000100100100111110111101111 & 00000 & 0001000000 \\
101111001010100010111001001111000011000 10 & 00000 & 0000100000 \\
101000101101110111100000100010110101010100 & 00000 & 0000010000 \\
010000010001101110111101111110010001111 & 00000 & 0000001000 \\
101110101100101110110100111111111001110 11 & 00000 & 0000000100 \\
011101011111110001001111110100011111 0111 & 00000 & 0000000010 \\
010110101111100100000110010010011010101 0 & 00000 & 0000000001 \\
110000110100000010101100101100100111 1101 & 10000 & 0000000000 \\
100100110000011101101011101000111001 0001 & 01000 & 0000000000 \\
011100000011010011101000101100110001 01110 & 00100 & 0000000000 \\
000100111100001101110100001010101100 1110 & 00010 & 0000000000 \\
111111110011111111100000111110110100010111 & 00001 & 0000000000
\end{array}\right]$$

From $G_{sys}^{(4)}$, it can be seen that the last k columns is also an information set, but $k - \text{Rank}\left(G_{sys}^{(4)}\right)$ coordinates of which overlap with those in $G_{sys}^{(3)}$. The generator matrix $G_{sys}^{(4)}$ then may be used to enumerate codewords with condition that the effect of overlapping information set has to be taken into account.

Say that all codewords with information weight $\leq w$ have been enumerated, it is known that

- for all $G_{sys}^{(i)}$ of full-rank, say there are m of these matrices, all cases of $d \leq mw$ have been considered and each contributes to the lower-bound. As a result, the lower-bound becomes $d_{lb} = m(w+1)$.
- for each $G_{sys}^{(i)}$ that do not have full-rank, the matrices $G_{sys}^{(i)}$ may be joined with column subset of $G_{sys}^{(j)}$, for $j < i$, so that an information set $\mathcal{I}_i$ is obtained, which of course overlaps with the information set $\mathcal{I}_j$. Therefore, for all of these matrices, say M, all cases of $d \leq Mw$ have been considered, but some of which are attributed to other information set and considering these would result in double-counting. According to Zimmermann (1996), for each matrix $G_{sys}^{(m+j)}$ with overlapping information set unless $w \geq k - \text{Rank}(B_{m+j})$ for which the lower-bound becomes $d_{lb} = d_{lb} + \{w - (k - \text{Rank}(B_{m+j})) + 1\}$, there is no contribution to the lower-bound.

As before, let the collection of full-rank reduced-echelon matrices be denoted by

$$\mathcal{G} = \left\{G_{sys}^{(1)}, G_{sys}^{(2)}, \ldots, G_{sys}^{(m)}\right\},$$

and let $\mathcal{G}'$ denote the collection of M rank matrices with overlapping information sets

$$\mathcal{G}' = \left\{G_{sys}^{(m+1)}, G_{sys}^{(m+2)}, \ldots, G_{sys}^{(m+M)}\right\}.$$

All $m + M$ generator matrices are need by the Zimmermann's (1996) algorithm. Clearly, if the condition $w \geq k - \text{Rank}(B_{m+j})$ is never satisfied throughout the enumeration, the corresponding generator matrix contributes nothing to the lower-bound and hence can be excluded (Grassl; 2006). In order to accommodate this improvement, the integer w_{max}, the maximum information weight that would need to be enumerated before the minimum distance is found, needs to be known. This can be accomplished as follows. Say at information weight w, a lower weight codeword is found, i.e. new d_{ub}, starting from $w' = w$, let $\mathcal{X} = \mathcal{G}'$, set $d_{lb} = m(w'+1)$ and then increment it by $(w' - (k - \text{Rank}(B_{m+j})) + 1)$ for each matrix in $\mathcal{G}'$ that satisfies $w' \geq k - \text{Rank}(B_{m+j})$. Each matrix that satisfies this condition is also excluded from $\mathcal{X}$. The weight w' is incremented, d_{lb} is recomputed and at the point when $d_{lb} \geq d_{ub}$, w_{max} is obtained and all matrices in $\mathcal{X}$ are those to be excluded from codeword enumeration.

In some cases, it has been observed that while enumerating codewords of information weight w, a codeword, whose weight coincides with the lower-bound obtained at enumeration step $w-1$, appears. Clearly, this implies that the newly found codeword is indeed a minimum weight codeword; any other codeword of lower weight–if they exist, would have been found in the earlier enumeration steps. This suggests that the enumeration at step w may be terminated immediately. Since the bulk of computation time increases exponentially as the information weight is increased, this termination may result in a reasonable amount of time saving.

Without loss of generality, it can be assumed that $\text{Rank}(\boldsymbol{B}_{m+j}) > \text{Rank}(\boldsymbol{B}_{m+j+1})$. With this assumption, the Zimmermann's (1996) approach to minimum distance evaluation of linear code over $\mathbb{F}_2$–with the improvements, may be written in Algorithm 4.1. The procedure to update w_{max} and $\mathscr{X}$ is given in Algorithm 4.2.

Depending on the code structure, the computation time required by Algorithm 4.1 can be reduced. Say the binary code considered has even weight codewords only, then at the end of codeword enumeration at each step, the lower-bound d_{lb} that has been obtained should be rounded to the next multiple of 2. Similarly, for codes where the weight of every codeword is divisible by 4, the lower-bound should be rounded to the next multiple of 4. Aside from this improvement, there are also other code specific improvements, for example see Section 4.2.4 for cyclic codes and Section 5.4 for double-circulant codes.

4.2.4 Chen's Algorithm for Cyclic Codes

Binary cyclic codes, which was introduced by Prange (1957), form an important class of block codes over $\mathbb{F}_2$. Cyclic codes constitute many well-known error-correcting codes such as the quadratic-residue codes and the commonly used in practice Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes. A binary cyclic code of length n, where n is necessary odd, has the property that if $c(x) = \sum_{i=0}^{n-1} c_i x^i$, where $c_i \in \mathbb{F}_2$ is a codeword of the cyclic code, then $x^j c(x) \pmod{x^n - 1}$, for some integer j, is also a codeword of that cyclic code. That is to say that the automorphism group of a cyclic code contains the coordinate permutation $i \to i+1 \pmod{n}$.

An $[n, k, d]$ binary cyclic code is defined by a generator polynomial $g(x)$ of degree $n-k$ and a parity-check polynomial $h(x)$ of degree k such that $g(x)h(x) = 0 \pmod{x^n - 1}$. Any codeword of this cyclic code is a multiple of $g(x)$, that is $c(x) = u(x)g(x)$ where $u(x)$ is any polynomial of degree less than k. The generator matrix G can be simply formed from the cyclic shifts of $g(x)$, i.e.

$$G = \begin{bmatrix} g(x) \pmod{x^n - 1} \\ xg(x) \pmod{x^n - 1} \\ \vdots \\ x^{k-1}g(x) \pmod{x^n - 1} \end{bmatrix}. \tag{4.4}$$

Since for some integer i, $x^i = q_i(x)g(x) + r_i(x)$ where $r_i(x) = x^i \pmod{g(x)}$, we can write

$$x^k \left(x^{n-k+i} - r_{n-k+i}(x)\right) = x^k q_i(x) g(x)$$

Algorithm 4.1 Minimum distance algorithm: improved Zimmermann's approach

Input: $\mathscr{G} = \left\{G_{sys}^{(1)}, G_{sys}^{(2)}, \ldots, G_{sys}^{(m)}\right\}$ where $|\mathscr{G}| = m$

Input: $\mathscr{G}' = \left\{G_{sys}^{(m+1)}, G_{sys}^{(m+2)}, \ldots, G_{sys}^{(m+M)}\right\}$ where $|\mathscr{G}'| = M$

Output: d (minimum distance)

1: $d' \leftarrow d_{ub} \leftarrow w_{max} \leftarrow k$
2: $d_{lb} \leftarrow w \leftarrow 1$
3: $\mathscr{X} = \emptyset$
4: **repeat**
5: $\quad M \leftarrow M - |\mathscr{X}|$
6: $\quad$ **for all** $i \in \mathbb{F}_2^k$ where $\mathrm{wt}_H(i) = w$ **do**
7: $\quad\quad$ **for** $1 \leq j \leq m$ **do**
8: $\quad\quad\quad d' \leftarrow \mathrm{wt}_H(i \cdot G_{sys}^{(j)})$
9: $\quad\quad\quad$ **if** $d' < d_{ub}$ **then**
10: $\quad\quad\quad\quad d_{ub} \leftarrow d'$
11: $\quad\quad\quad\quad$ **if** $d_{ub} \leq d_{lb}$ **then**
12: $\quad\quad\quad\quad\quad$ Goto Step 36
13: $\quad\quad\quad\quad$ **end if**
14: $\quad\quad\quad\quad w_{max}, \mathscr{X} \leftarrow$ Update w_{max} and $\mathscr{X}$ $(d_{ub}, k, m, \mathscr{G}')$
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **end for**
17: $\quad\quad$ **for** $1 \leq j \leq M$ **do**
18: $\quad\quad\quad d' \leftarrow \mathrm{wt}_H(i \cdot G_{sys}^{(m+j)})$
19: $\quad\quad\quad$ **if** $d' < d_{ub}$ **then**
20: $\quad\quad\quad\quad d_{ub} \leftarrow d'$
21: $\quad\quad\quad\quad$ **if** $d_{ub} \leq d_{lb}$ **then**
22: $\quad\quad\quad\quad\quad$ Goto Step 36
23: $\quad\quad\quad\quad$ **end if**
24: $\quad\quad\quad\quad w_{max}, \mathscr{X} \leftarrow$ Update w_{max} and $\mathscr{X}$ $(d_{ub}, k, m, \mathscr{G}')$
25: $\quad\quad\quad$ **end if**
26: $\quad\quad$ **end for**
27: $\quad$ **end for**
28: $\quad d_{lb} \leftarrow m(w+1)$
29: $\quad$ **for** $1 \leq j \leq M$ **do**
30: $\quad\quad$ **if** $w \geq \{k - \mathrm{Rank}\,(B_{m+j})\}$ **then**
31: $\quad\quad\quad d_{lb} = d_{lb} + \{w - (k - \mathrm{Rank}\,(B_{m+j})) + 1\}$
32: $\quad\quad$ **end if**
33: $\quad$ **end for**
34: $\quad w \leftarrow w + 1$
35: **until** $d_{lb} \geq d_{ub}$ OR $w > k$
36: $d \Leftarrow d_{ub}$

and based on this, a reduced-echelon generator matrix G_{sys} of a cyclic code is obtained,

$$G_{sys} = \left[\begin{array}{c|c} & -x^{n-k} \pmod{g(x)} \\ & -x^{n-k+1} \pmod{g(x)} \\ I_k & -x^{n-k+2} \pmod{g(x)} \\ & \vdots \\ & -x^{n-1} \pmod{g(x)} \end{array}\right]. \tag{4.5}$$

Algorithm 4.2 w_{max}, $\mathscr{X}$ = Update w_{max} and $\mathscr{X}$ $(d_{ub}, k, m, \mathscr{G}')$

Input: d_{ub}, k, m
Input: $\mathscr{G}'\left\{G_{sys}^{(m+1)}, G_{sys}^{(m+2)}, \ldots, G_{sys}^{(m+M)}\right\}$
Output: w_{max} and $\mathscr{X}$

```
1:  X ← G'
2:  w_max ← 1
3:  repeat
4:      d_lb ← m(w_max + 1)
5:      for 1 ≤ j ≤ |G'| do
6:          if w_max ≥ {k − Rank(B_{m+j})} then
7:              Remove G_sys^(m+j) from X if G_sys^(m+j) ∈ X
8:              d_lb = d_lb + {w_max − (k − Rank(B_{m+j})) + 1}
9:          end if
10:     end for
11:     w_max ← w_max + 1
12: until d_lb ≥ d_ub OR w_max > k
13: return w_max and X
```

The matrix G_{sys} in (4.5) may contain more than one mutually disjoint information sets. But because each codeword is invariant under cyclic shift, a codeword generated by information set $\mathcal{I}_i$ can be obtained from information set $\mathcal{I}_j$ by means of simple cyclic shift. For an $[n, k, d]$ cyclic code, there always exists $\lfloor n/k \rfloor$ mutually disjoint information sets. As a consequence of this, using a single information set is sufficient to improve the lower-bound to $\lfloor n/k \rfloor (w+1)$ at the end of enumeration step w. However, Chen (1969) showed that this lower-bound could be further improved by noting that the average number of non zeros of a weight w_0 codeword in an information set is $w_0 k/n$. After enumerating $\sum_{i=1}^{w} \binom{k}{i}$ codewords, it is known that the weight of a codeword restricted to the coordinates specified by an information set is at least $w+1$. Relating this to the average weight of codewords in an information set, an improved lower-bound of $d_{lb} = \lceil (w+1)n/k \rceil$ is obtained. Algorithm 4.3 summarises Chen's (1969) approach to minimum distance evaluation of a binary cyclic code. Note that Algorithm 4.3 takes into account the early termination condition suggested in Section 4.2.3.

For cyclic codes, the lower bound on their minimum distance may be obtained using the BCH bound. That is, assuming that β is a primitive nth root of unity, if there are δ consecutive powers of β (taken cyclically modulo n) in the roots of the generator polynomial of a cyclic code of length n, the minimum distance of the cyclic code is at least $\delta + 1$. With this information, Chen's (1969) algorithm can be terminated as soon as a codeword of weight $\delta + 1$ is found. It is not necessary to enumerate all codewords of information weights less than or equal to w, which may take a considerable amount of time, such that the condition $\lceil (w+1)n/k \rceil \geq \delta + 1$ is satisfied.

It is worth noting that both minimum distance evaluation algorithm of Zimmermann (1996) for linear codes and that of Chen (1969) for cyclic codes, may be used to compute the number of codewords of a given weight. In evaluating the minimum distance d, the algorithm is terminated after enumerating all codewords having information weight i to w, where w is the smallest integer at which the condition $d_{lb} \geq d$ is reached. To compute the number of codewords of weight d, in addition to enumerate all codewords of weight i to w in their information set, all codewords having weight $w+1$ in their information sets, also need to be enumerated. For Zimmermann's (1996) approach, all

Algorithm 4.3 Minimum distance algorithm for cyclic codes: Chen's approach

Input: $G_{sys} = [I_k|P]$ {see (4.5)}
Output: d (minimum distance)

1: $d_{ub} \leftarrow k$
2: $d_{lb} \leftarrow 1$
3: $w \leftarrow 1$
4: **repeat**
5: $\quad d' \leftarrow k$
6: $\quad$ **for all** $i \in \mathbb{F}_2^k$ where $\mathrm{wt}_H(i) = w$ **do**
7: $\quad\quad d' \leftarrow \mathrm{wt}_H(i \cdot G_{sys})$
8: $\quad\quad$ **if** $d' < d_{ub}$ **then**
9: $\quad\quad\quad d_{ub} \leftarrow d'$
10: $\quad\quad\quad$ **if** $d_{ub} \leq d_{lb}$ **then**
11: $\quad\quad\quad\quad$ Goto Step 18
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad d_{lb} \leftarrow \left\lceil \frac{n}{k}(w+1) \right\rceil$
16: $\quad w \leftarrow w + 1$
17: **until** $d_{lb} \geq d_{ub}$ OR $w > k$
18: $d \Leftarrow d_{ub}$

the available information sets, including those that overlap, are utilised and all codewords whose weight matches d are stored. Meanwhile for Chen's (1969) algorithm, only a single information set is utilised, but for each codeword of weight d found, this codeword and all of the $n-1$ cyclic shifts are accumulated. In both approaches, the doubly-counted codewords are removed at the end of enumeration.

4.2.5 Codeword Enumeration Algorithm

The core of all minimum distance evaluation and codeword counting algorithms lies on the codeword enumeration. Given a reduced echelon generator matrix, codewords can be enumerated by taking linear combinations of the rows in the generator matrix. This suggests the need of an efficient algorithm to generate combinations. One of the most efficient algorithm for this purpose is the revolving-door algorithm, see (Bitner et al.; 1976; Nijenhuis and Wilf; 1978; Knuth; 2005). The efficiency of the revolving-door algorithm arises from the property that in going from one combination pattern to the next, there is only an element that is exchanged. An efficient implementation of the revolving-door algorithm is given in Knuth (2005), called *Algorithm R*, which is attributed to Payne and Ives (1979)†.

In many cases, using a single-threaded program to either compute the minimum distance, or count the number of codewords of a given weight, of a linear code may take a considerably amount of time. In this case, we may resort to a multi-threaded approach by splitting the codeword enumeration on multiple computers. The revolving-door algorithm has a nice property that allows such splitting to be neatly realised. Let $a_t a_{t-1} \ldots a_2 a_1$, where $a_t > a_{t-1} > \ldots > a_2 > a_1$, be a pattern of an t out of s combination–C_t^s. A pattern is said to have rank i if this pattern appears as the

†This is the version that the author implemented to compute the minimum distance and to count the number of codewords of a given weight of a binary linear code.

$(i+1)$th element in the list of all C_t^s combinations[‡]. Let $\text{Rank}(a_t a_{t-1} \ldots a_2 a_1)$ be the rank of pattern $a_t a_{t-1} \ldots a_2 a_1$, the revolving-door algorithm has the property that (Knuth; 2005)

$$\text{Rank}(a_t a_{t-1} \ldots a_2 a_1) = \left[\binom{a_t+1}{t} - 1\right] - \text{Rank}(a_{t-1} \ldots a_2 a_1) \tag{4.6}$$

and, for each integer N, where $0 \leq N \leq \binom{s}{t} - 1$, it can be uniquely represented with an ordered pattern $a_t a_{t-1} \ldots a_2 a_1$. As an implication of this and (4.6), if all $\binom{k}{t}$ codewords need to be enumerated, the enumeration can be split into $\left\lceil \binom{k}{t}/M \right\rceil$ blocks where in each block only at most M codewords need to be generated. In this way, the enumeration of each block can be done on a separate computer and this allows a parallelisation of the minimum distance evaluation in addition to the counting of the number of codewords of a given weight. It is clear that, in the ith block, the enumeration would start from rank $(i-1)M$ and the corresponding pattern can be easily obtained following (4.6) and Lemma 4.1 below.

All $a_t a_{t-1} \ldots a_2 a_1$ revolving-door patterns of C_t^s satisfy the property that, if the values in position a_t grow in an increasing order, then for fixed a_t the values in position a_{t-1} grow in a decreasing order, moreover for fixed $a_t a_{t-1}$ the values in position a_{t-2} grow in an increasing order, and so on in an alternating order. This behaviour is evident by observing all revolving-door patterns of C_4^6 (left) and C_5^7 (right) shown in Figure 4.1. From this figure, it can also be observed that

$$C_t^s \supset C_t^{s-1} \supset \ldots \supset C_t^{t+1} \supset C_t^t, \tag{4.7}$$

and this suggests the following lemma.

4.1 ***Lemma (Maximum and Minimum Ranks).*** Consider the $a_t a_{t-1} \ldots a_2 a_1$ revolving-door combination pattern, if the patterns with fixed a_t are considered, the maximum and minimum ranks of such pattern are respectively given by

$$\binom{a_t+1}{t} - 1 \qquad \text{and} \qquad \binom{a_t}{t}.$$

Example 4.2: Say if all C_4^6 revolving-door combination patterns (left portion of Figure 4.1), where $a_4 = 4$, are considered. Lemma 4.1 yields a maximum rank of $\binom{5}{4} - 1 = 4$ and a minimum rank of $\binom{4}{4} = 1$, and it can be seen that these rank values are correct from Figure 4.1.

Maximum and minimum ranks

Example 4.3: Consider combinations C_5^7 generated by the revolving-door algorithm, what is the combination pattern at rank 17? It is known that the combination pattern takes an ordered form of $a_5 a_4 a_3 a_2 a_1$, where $a_i > a_{i-1}$. Starting from a_5, an integer between 0 and 6 has to be found such that

Rank of a revolving-door combination pattern

[‡]Here it is assume that the first element in the list of all C_t^s combinations has rank 0.

C_4^6

Rank	a_4	a_3	a_2	a_1
0	3	2	1	0
1	4	3	1	0
2	4	3	2	1
3	4	3	2	0
4	4	2	1	0
5	5	4	1	0
6	5	4	2	1
7	5	4	2	0
8	5	4	3	2
9	5	4	3	1
10	5	4	3	0
11	5	3	1	0
12	5	3	2	1
13	5	3	2	0
14	5	2	1	0

C_5^7

Rank	a_5	a_4	a_3	a_2	a_1
0	4	3	2	1	0
1	5	4	2	1	0
2	5	4	3	2	0
3	5	4	3	2	1
4	5	4	3	1	0
5	5	3	2	1	0
6	6	5	2	1	0
7	6	5	3	2	0
8	6	5	3	2	1
9	6	5	3	1	0
10	6	5	4	3	0
11	6	5	4	3	1
12	6	5	4	3	2
13	6	5	4	2	0
14	6	5	4	2	1
15	6	5	4	1	0
16	6	4	2	1	0
17	6	4	3	2	0
18	6	4	3	2	1
19	6	4	3	1	0
20	6	3	2	1	0

Figure 4.1: C_4^6 and C_5^7 revolving-door combination patterns

the inequality $\binom{a_5}{5} \le 17 \le \binom{a_5+1}{5} - 1$ is satisfied (Lemma 4.1). It follows that $a_5 = 6$ and using (4.6),

$$\begin{aligned} 17 &= \mathrm{Rank}(6a_4a_3a_2a_1) \\ &= \left[\binom{6+1}{5} - 1\right] - \mathrm{Rank}(a_4a_3a_2a_1) \\ \mathrm{Rank}(a_4a_3a_2a_1) &= 20 - 17 = 3\,. \end{aligned}$$

Next, consider a_4 and as before, an integer between 0 and 5 needs to be found for a_4 such that the inequality $\binom{a_4}{4} \le \mathrm{Rank}(a_4a_3a_2a_1) \le \binom{a_4+1}{4} - 1$ is satisfied. It follows that $a_4 = 4$ and consequently,

$$\begin{aligned} 3 &= \mathrm{Rank}(4a_3a_2a_1) \\ &= \left[\binom{4+1}{4} - 1\right] - \mathrm{Rank}(a_3a_2a_1) \\ \mathrm{Rank}(a_3a_2a_1) &= 4 - 3 = 1\,, \end{aligned}$$

following (4.6).

Next, find a_3, which can only take any non negative integer less than 4, such that the inequality $\binom{a_3}{3} \le \mathrm{Rank}(a_3a_2a_1) \le \binom{a_3+1}{3} - 1$ is satisfied. It follows that $a_3 = 3$ and from (4.6), $\mathrm{Rank}(a_2a_1) = \left[\binom{3+1}{3} - 1\right] - 1 = 2$.

So far it is known that $643a_2a_1$, only a_2 and a_1 are unknown. Since $a_3 = 3$, a_2 can only take integer less than 3. The inequality $\binom{a_2}{2} \le \mathrm{Rank}(a_2a_1) \le \binom{a_2+1}{2} - 1$ is satisfied if $a_2 = 2$ and corre-

spondingly, $\text{Rank}(a_1) = \left[\binom{2+1}{2} - 1\right] - 2 = 0$.

For the last case, the inequality $\binom{a_1}{1} \leq \text{Rank}(a_1) \leq \binom{a_1+1}{1} - 1$ is true if and only if $a_1 = 0$. Thus, 64320 is obtained as the rank 17 C_5^7 revolving-door pattern. Cross-checking this with Figure 4.1, it is clear that 64320 is indeed of rank 17.

From (4.6) and Example 4.3, given a rank N, where $0 \leq N \leq \binom{s}{t} - 1$, an ordered pattern of C_t^s revolving-door combination $a_t a_{t-1} \ldots a_2 a_1$ can be constructed recursively. A software realisation of this recursive approach is given in Algorithm 4.4.

Algorithm 4.4 RecursiveCompute a_i $(\text{Rank}(a_i a_{i-1} \ldots a_2 a_1), i)$

Input: i and $\text{Rank}(a_i a_{i-1} \ldots a_2 a_1)$
Output: a_i

1: Find a_i, where $0 \leq a_i < a_{i+1}$, such that $\binom{a_i}{i} \leq \text{Rank}(a_i a_{i-1} \ldots a_2 a_1) \leq \left[\binom{a_i+1}{i} - 1\right]$
2: **if** $i > i$ **then**
3: Compute $\text{Rank}(a_{i-1} \ldots a_2 a_1) = \left[\binom{a_i+1}{i} - 1\right] - \text{Rank}(a_i a_{i-1} \ldots a_2 a_1)$
4: RecursiveCompute a_i $(\text{Rank}(a_{i-1} \ldots a_2 a_1), i - 1)$
5: **end if**
6: **return** a_i

4.3 Binary Cyclic Codes of Lengths $129 \leq n \leq 189$

The minimum distance of all binary cyclic codes of lengths less than or equal to 99 has been determined by Chen (1969), Chen (1970) and Promhouse and Tavares (1978). This was later extended with evaluation of the minimum distance of binary cyclic codes of lengths from 101 to 127 by Schomaker and Wirtz (1992). This work is extended to include cyclic codes of odd lengths from 129 to 189 in this thesis. The aim of this work is to provide as a reference the highest minimum distance, with the corresponding roots of the generator polynomial, attainable by all cyclic codes over $\mathbb{F}_2$ of odd lengths from 129 to 189. It is well-understood that the coordinate permutation $\sigma : i \to \mu i$, where μ is an integer relatively prime to n, produces equivalent cyclic codes (Berlekamp; 1984, pp. 141f). With the respect to this property, we construct a list of generator polynomials $g(x)$ of all inequivalent and non degenerate (MacWilliams and Sloane; 1977, pp. 223f) cyclic codes of $129 \leq n \leq 189$ by taking products of the irreducible factors of $x^n - 1$. Two trivial cases are excluded, namely $g(x) = x + 1$ and $g(x) = (x^n - 1)/(x+1)$, since they have trivial minimum distance and exist for any odd integer n. The idea is, for each $g(x)$ of cyclic codes of odd length n, the systematic generator matrix is formed and the minimum distance of the code is determined using Chen's algorithm (Algorithm 4.3). However, due to the large amount of cyclic codes and the fact that only those of largest minimum distance for given n and k are of interest, a threshold distance d_{th} in Algorithm 4.3 is utilised. Say for given n and k, there is a list of generator polynomials $g(x)$ of all inequivalent cyclic codes. Starting from the top of the list, the minimum distance of the corresponding cyclic code is evaluated. If a codeword of weight less than or equal to d_{th} is found during the enumeration, the computation is terminated immediately and the next $g(x)$ is then processed. The threshold d_{th}, which is initialised with 0, is updated with the largest minimum distance found so far for the given n and k.

Table B.1 in Appendix B shows the highest attainable minimum distance of all binary cyclic codes of odd lengths from 129 to 189. The number of inequivalent and non degenerate cyclic codes

for a given odd integer n, excluding the two trivial cases mentioned above, is denoted by N_C. Note that Table B.1 does not contain entries for primes $n = 8m \pm 3$. This is because for these primes, 2 is not a quadratic residue modulo n and hence, $\text{ord}_2(n) = n - 1$. As a consequence, $x^n - 1$ factors into two irreducible polynomials only, namely $x+1$ and $(x^n - 1)/(x+1)$ which generate trivial codes. Let β be a primitive nth root of unity, the roots of $g(x)$ of a cyclic code (excluding the conjugate roots) are given in terms of the exponents of β. The polynomial $m(x)$ is the minimal polynomial of β and it is represented in octal format with most significant bit on the left. That is, $m(x) = 166761$, as in the case for $n = 151$, represents $x^{15} + x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + 1$.

4.4 Some New Binary Cyclic Codes of Large Minimum Distance

Constructing an $[n, k]$ linear codes of largest minimum distance is one of the main problems in coding theory. There exists a database containing the lower- and upper-bounds of minimum distance of binary linear codes of lengths $1 \leq n \leq 256$. This database appears in Brouwer (1998) and the updated version was accessible online[§]. The lower-bound corresponds to the largest minimum distance for a given $[n, k]_q$ code that has been found to date. Constructing codes which improves Brouwer's lower-bounds is an on-going research activity in coding theory. Recently, tables of lower- and upper-bounds of not only codes over finite-fields, but also quantum error-correcting codes, have been published by Grassl (2007). These bounds for codes over finite-fields, which are derived from MAGMA (Bosma et al.; 1997), appear to be more up-to-date than those of Brouwer.

It has been presented in Section 4.3 the highest minimum distance attainable by all binary cyclic codes of odd lengths from 129 to 189 and it has been found that none of these cyclic code has larger minimum distance than the corresponding Brouwer's lower-bound for the same n and k. The next step is to consider longer length cyclic codes, $191 \leq n \leq 255$. For these lengths, unfortunately, it has not been feasible to repeat the exhaustive approach of Section 4.3 in a reasonable amount of time. This is due to the computation time to determine the minimum distance of these cyclic codes and also, for some lengths (e.g. 195 and 255), there are tremendous amount of inequivalent cyclic codes. Having said that, it is possible to search for improvements from lower rate cyclic codes of these lengths for which the minimum distance computation can be completed in a reasonable time. Many new cyclic codes that improve Brouwer's lower-bound were found and before they are presented, consider the evaluation procedure that is employed.

As before, let β be a primitive nth root of unity and let Λ be a set containing all distinct (excluding the conjugates) exponents of β. The polynomial $x^n - 1$ can be factorised into irreducible polynomials $f_i(x)$ over $\mathbb{F}_2$, $x^n - 1 = \prod_{i \in \Lambda} f_i(x)$. For notational purpose, the irreducible polynomial $f_i(x)$ is denoted as the minimal polynomial of β^i. The generator and parity-check polynomials, denoted by $g(x)$ and $h(x)$ respectively, are products of $f_i(x)$. Given a set $\Gamma \subseteq \Lambda$, a cyclic code C which has β^i, $i \in \Gamma$, as the

[§]The online database is available at `http://www.win.tue.nl/~aeb/voorlincod.html`.
Note that, since 12th March 2007, A. Brouwer has stopped maintaining his database and hence it is no longer accessible. This database is now superseded by that maintained by Grassl (2007).

non zeros can be constructed. This means the parity-check polynomial $h(x)$ is given by

$$h(x) = \prod_{i \in \Gamma} f_i(x)$$

and the dimension k of this cyclic code is $\sum_{i \in \Gamma} \deg(f_i(x))$, where $\deg(f(x))$ denotes the degree of $f(x)$. Let $\Gamma' \subseteq \Lambda \setminus \{0\}$, $h'(x) = \prod_{i \in \Gamma'} f_i(x)$ and $h(x) = (1+x)h'(x)$. Given C with parity-check polynomial $h(x)$, there exist an $[n, k-1, d']$ expurgated cyclic code, C', which has parity-check polynomial $h'(x)$. For this cyclic code, $\mathrm{wt}_H(c) \equiv 0 \pmod 2$ for all $c \in C'$.For convenience, let the code C be termed the augmented code of C'.

Consider an $[n, k-1, d']$ expurgated cyclic code C', let the set $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_r\}$ where, for $1 \leq j \leq r$, $\Gamma_j \subseteq \Lambda \setminus \{0\}$ and $\sum_{i \in \Gamma_j} \deg(f_i(x)) = k-1$. For each $\Gamma_j \in \Gamma$, $h'(x)$ is obtained and the corresponding code C' is constructed. Having done that, the augmented code can be easily obtained as shown below. Let G be a generator matrix of the augmented code C, and without loss of generality, it can be written as

$$G = \begin{array}{|c|} \hline G' \\ \hline v \\ \hline \end{array} \tag{4.8}$$

where G' is a generator matrix of C' and the vector v is a coset of C' in C. Using the arrangement in (4.8), d' is evaluated by enumerating codewords $c \in C'$ from G'. The minimum distance of C, denoted by d, is simply $\min_{c \in C'}\{d', \mathrm{wt}_H(c+v)\}$ for all codewords c enumerated. Algorithm 4.3 is employed to evaluate d'. Let $d_{Brouwer}$ and $d'_{Brouwer}$ denote the lower-bounds of Brouwer (1998) for linear codes of the same length and dimension as those of C and C' respectively. During the enumerations, as soon as $d \leq d_{Brouwer}$ and $d' \leq d'_{Brouwer}$, the evaluation is terminated and the next Γ_j in Γ is then processed. However, if $d \leq d_{Brouwer}$ and $d' > d'_{Brouwer}$, only the evaluation for C is discarded. Nothing is discarded if both $d' > d'_{Brouwer}$ and $d > d_{Brouwer}$. This procedure continues until improvement is obtained; or the set in Γ has been exhausted, which means that there does not exist $[n, k-1]$ and $[n, k]$ cyclic codes which are improvements to Brouwer's lower-bounds. In cases where the minimum distance computation is not feasible using a single computer, the evaluation is switched to parallel version using grid computers.

Table 4.1 presents the results of the search on new binary cyclic codes for $195 \leq n \leq 255$ described earlier. The cyclic codes in this table are expressed in terms of the parity-check polynomial $h(x)$, which is given in the last column by the exponents of β (excluding the conjugates). Note that the polynomial $m(x)$, which is given in octal with most significant bit on the left, is the minimal polynomial of β. In many cases, the entries of C and C' are combined in a single row and this is indicated by "a/b" where the parameters a and b are for C' and C respectively. The notation "[0]" indicates that the polynomial $(1+x)$ is to be excluded from the parity-check polynomial of C'. Some of the improvements coincide with the lower-bounds in Grassl (2007). They are included in Table 4.1 and are marked by "†".

In the late 1970s, computing the minimum distance of extended Quadratic Residue (QR) codes was posed as a research problem in MacWilliams and Sloane (1977). Since then, the minimum

Table 4.1: New Binary Cyclic Codes

$[m(x)]_8$	n	k	d	$d_{Brouwer}$	$h(x)$
17277	195	† 66/67	42/41	40/40	[0], 3, 5, 9, 19, 39, 65, 67
		† 68/69	40/39	39/38	[0], 1, 3, 13, 19, 35, 67, 91
		† 73	38	37	0, 3, 7, 19, 33, 35, 47
		† 74/75	38/37	36/36	[0], 3, 7, 19, 33, 35, 47, 65
		78	36	35	3, 7, 9, 11, 19, 35, 39, 65
13237042705-30057231362-555070452551	199	99/100	32/31	28/28	[0], 1
6727273	205	† 60	48	46	5, 11, 31
		† 61	46	44	0, 3, 11, 31
3346667657	215	70/71	46/46	44/44	[0], 3, 13, 35
3705317547055	223	74/75	48/47	46/45	[0], 5, 9
3460425444467-7544446504147	229	76	48	46	1
6704436621	233	† 58/59	60/60	56/56	[0], 3, 29
150153013	241	† 49	68	65	0, 1, 21
		73	54	53	0, 1, 3, 25
435	255	48/49	76/75	75/72	[0], 47, 55, 91, 95, 111, 127
		50/51	74/74	72/72	[0], 9, 13, 23, 47, 61, 85, 127
		52/53	72/72	71/68	[0], 7, 9, 17, 47, 55, 111, 127
		54/55	70/70	68/68	[0], 3, 7, 23, 47, 55, 85, 119, 127
		56/57	68/68	67/65	[0], 7, 27, 31, 45, 47, 55, 127
		58	66	64	7, 39, 43, 45, 47, 55, 85, 127
		60	66	64	7, 17, 23, 39, 45, 47, 55, 127
		62/63	66/65	64/63	[0], 11, 21, 47, 55, 61, 85, 87, 119, 127
		64/65	64/63	62/62	[0], 19, 31, 39, 47, 55, 63, 91, 127

distance of the extended QR code for prime 199 has been an open question. For this code, the bounds of the minimum distance were $16 - 32$ in MacWilliams and Sloane (1977) and the lower-bound was improved to 24 in Grassl (2000). Since $199 \equiv -1 \pmod 8$, the extended code is doubly-even self-dual and its automorphism group contains a projective special linear group, which is known to be doubly-transitive (MacWilliams and Sloane; 1977). As a result, the minimum distance of the binary $[199, 100]$ QR code is odd, i.e. $d \equiv 3 \pmod 4$, and hence $d = 23, 27$ or 31. Due to the cyclic property and the rate of this QR code (Chen; 1969), it can be assumed that a codeword of weight d has maximum

information weight of $\lfloor d/2 \rfloor$. If a weight d codeword does not satisfy this property, there must exist one of its cyclic shifts that does. After enumerating all codewords up to (and including) information weight 13 using grid computers, no codeword of weight less than 31 was found, implying that d of this binary $[199, 100]$ QR code is indeed 31. Without exploiting the property that $d \equiv 3 \pmod 4$, additional $\binom{100}{14} + \binom{100}{15}$ codewords would need to be enumerated in order to establish the same result. Accordingly, there exists a $[199, 99, 32]$ expurgated QR and a $[200, 100, 32]$ extended QR codes.

It is interesting that many of the improvements are contributed by low-rate cyclic codes of length 255 and there are 16 cases of this. Furthermore, it is interesting to note the existence of $[255, 55, 70]$ and $[255, 63, 65]$ cyclic codes, which are superior to the BCH codes of the same length and dimension. Both of these BCH codes have minimum distance 63 only.

4.5 Constructing New Codes from Existing Ones

It is difficult to explicitly construct a new code with large minimum distance. However, the alternative approach, which starts from a known code which already has large minimum distance, seems to be more fruitful. Some of these methods are described below and in the following subsections, some new binary codes obtained using these methods, which improve Brouwer's lower-bound, are presented.

4.1 ***Theorem (Construction X).*** Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be $[n, k_1, d_1]$ and $[n, k_2, d_2]$ linear codes over $\mathbb{F}_q$ respectively, where $\mathcal{B}_1 \supset \mathcal{B}_2$ ($\mathcal{B}_2$ is a subcode of $\mathcal{B}_1$). Let $\mathcal{A}$ be an $[n', k_3 = k_1 - k_2, d']$ auxiliary code over the same field. There exists an $[n + n', k_1, \min\{d_2, d_1 + d'\}]$ code $\mathcal{C}_X$ over $\mathbb{F}_q$.

Construction X is due to Sloane et al. (1972) and it basically adds tail, which is a codeword of the auxiliary code $\mathcal{A}$, to $\mathcal{B}_1$ so that the minimum distance is increased. The effect of Construction X can be visualised as follows. Let $G_{\mathcal{C}}$ be the generator matrix of code $\mathcal{C}$. Since $\mathcal{B}_1 \supset \mathcal{B}_2$, $G_{\mathcal{B}_1}$ can be expressed as

$$G_{\mathcal{B}_1} = \left[\begin{array}{c} G_{\mathcal{B}_2} \\ \hline V \end{array}\right],$$

where V is a $(k_1 - k_2) \times n$ matrix which contains the cosets of $\mathcal{B}_2$ in $\mathcal{B}_1$. It can be seen that the code generated by $G_{\mathcal{B}_2}$ has minimum distance d_2, and the set of codewords $\{v + c_2\}$, for all $v \in V$ and all codewords c_2 generated by $G_{\mathcal{B}_2}$, have minimum weight of d_1. By appending non zero weight codewords of $\mathcal{A}$ to the set $\{v + c_2\}$, and all zeros codeword to each codeword of $\mathcal{B}_2$, a lengthened code of larger minimum distance, $\mathcal{C}_X$, is obtained which has a generator matrix given by

$$G_{\mathcal{C}_X} = \left[\begin{array}{c|c} G_{\mathcal{B}_2} & 0 \\ \hline V & G_{\mathcal{A}} \end{array}\right]. \tag{4.9}$$

It can be seen that, for binary cyclic linear codes of odd minimum distance, code extension by annexing an overall parity-check bit is an instance of Construction X. In this case, $\mathcal{B}_2$ is the even-weight subcode of $\mathcal{B}_1$ and the auxiliary code $\mathcal{A}$ is the trivial $[1, 1, 1]_2$ code.

Construction X given in Theorem 4.1 consider a chain of two codes only. It can be generalised by considering a chain of more codes and this is given in Theorem 7.1. There also exists a variant of Construction X which makes use of Construction X twice and it was introduced by Alltop (1984).

4.2 ***Theorem (Construction XX).*** Consider three linear codes of the same length, $\mathcal{B}_1 = [n, k_1, d_1]$, $\mathcal{B}_2 = [n, k_2, d_2]$ and $\mathcal{B}_3 = [n, k_3, d_3]$ where $\mathcal{B}_2 \subset \mathcal{B}_1$ and $\mathcal{B}_3 \subset \mathcal{B}_1$. Let $\mathcal{B}_4$ be an $[n, k_4, d_4]$ linear code which is the intersection code of $\mathcal{B}_2$ and $\mathcal{B}_3$, i.e. $\mathcal{B}_4 = \mathcal{B}_2 \cap \mathcal{B}_3$. Using auxiliary codes $\mathcal{A}_1 = [n_1, k_1 - k_2, d_1']$ and $\mathcal{A}_2 = [n_2, k_1 - k_3, d_2']$, there exists an $[n + n_1 + n_2, k_1, d]$ linear code C_{XX} where $d = \min\{d_4, d_3 + d_1', d_2 + d_2', d_1 + d_1' + d_2'\}$.

The relationship among $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ can be illustrated as a lattice shown below (Grassl; 2006).

$$\begin{tikzcd}
 & \mathcal{B}_1 \arrow[dl] \arrow[dr] & \\
\mathcal{B}_2 \arrow[dr] & & \mathcal{B}_3 \arrow[dl] \\
 & \mathcal{B}_4 &
\end{tikzcd}$$

Since $\mathcal{B}_1 \supset \mathcal{B}_2$, $\mathcal{B}_1 \supset \mathcal{B}_3$, $\mathcal{B}_4 \subset \mathcal{B}_2$ and $\mathcal{B}_4 \subset \mathcal{B}_3$, the generator matrices of $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_1$ can be written as

$$G_{\mathcal{B}_2} = \left[\begin{array}{c} G_{\mathcal{B}_4} \\ \hline V_2 \end{array}\right], \quad G_{\mathcal{B}_3} = \left[\begin{array}{c} G_{\mathcal{B}_4} \\ \hline V_3 \end{array}\right] \quad \text{and} \quad G_{\mathcal{B}_1} = \left[\begin{array}{c} G_{\mathcal{B}_4} \\ \hline V_2 \\ \hline V_3 \\ \hline V \end{array}\right]$$

respectively, where V_i, $i = 2, 3$, is the coset of $\mathcal{B}_4$ in $\mathcal{B}_i$, and V contains the cosets of $\mathcal{B}_2$ and $\mathcal{B}_3$ in $\mathcal{B}_1$. Construction XX starts by applying Construction X to the pair of codes $\mathcal{B}_1 \supset \mathcal{B}_2$ using $\mathcal{A}_1$ as the auxiliary code. The resulting code is $C_X = [n + n_1, k_1, \min\{d_2, d_1 + d_1'\}]$, whose generator matrix is given by

$$G_{C_X} = \left[\begin{array}{c|c} G_{\mathcal{B}_4} & 0 \\ \hline V_2 & \\ \hline V_3 & \\ \hline V & G_{\mathcal{A}_1} \end{array}\right].$$

This generator matrix can be rearranged such that the codewords formed from the first n coordinates are cosets of $\mathcal{B}_3$ in $\mathcal{B}_1$. This rearrangement results in the following generator matrix of C_X,

$$G_{C_X} = \left[\begin{array}{c|c} G_{\mathcal{B}_4} & 0 \\ \hline V_3 & G^{(1)}_{\mathcal{A}_1} \\ \hline V_2 & 0 \\ \hline V & G^{(2)}_{\mathcal{A}_1} \end{array}\right],$$

where $G_{\mathcal{A}_1} = \left[\begin{array}{c} G^{(1)}_{\mathcal{A}_1} \\ \hline G^{(2)}_{\mathcal{A}_1} \end{array}\right]$. Next, using $\mathcal{A}_2$ as the auxiliary code, applying Construction X to the pair $\mathcal{B}_1 \supset \mathcal{B}_3$ with the rearrangement above yields a code C_{XX} whose generator matrix is

$$G_{C_{XX}} = \left[\begin{array}{c|c|c} G_{\mathcal{B}_4} & 0 & 0 \\ \hline V_3 & G^{(1)}_{\mathcal{A}_1} & \\ \hline V_2 & 0 & G_{\mathcal{A}_2} \\ \hline V & G^{(2)}_{\mathcal{A}_1} & \end{array}\right].$$

While Constructions X and XX result in code with increased length, there also exists a technique to obtain a shorter code with known minimum distance lower-bound from a longer code whose minimum distance and also that of its dual code are known explicitly. This technique is due to Sloane et al. (1972) and it is called the Construction Y1.

4.3 ***Theorem (Construction Y1).*** Given an $[n, k, d]$ linear code C, which has an $[n, n-k, d^{\perp}]$ code $C^{\perp}$ as its dual, an $[n-d^{\perp}, k-d^{\perp}+1, \geq d]$ code C' can be constructed.

Given an $[n, k, d]$ code, with standard code shortening, an $[n-i, k-i, \geq d]$ code, where i indicates the number of coordinates to shorten, is obtained. With Construction Y1, however, an additional dimension in the resulting shortened code can be gained. This can be explained as follows. Without loss of generality, it can be assumed that the parity-check matrix of C, which is also the generator matrix of $C^{\perp}$, H contains a codeword $c^{\perp}$ of weight $d^{\perp}$. If the coordinates which form the support of $c^{\perp}$ are deleted from H, then H becomes an $(n-k) \times n-d^{\perp}$ matrix and there is a row which contains all zeros among these $n-k$ rows. Removing this all zeros row yields an $(n-k-1) \times (n-d^{\perp})$ matrix H', which is the parity-check matrix of an $[n-d^{\perp}, n-d^{\perp}-(n-k-1), \geq d] = [n-d^{\perp}, k-d^{\perp}+1, \geq d]$ code C'.

4.5.1 New Binary Codes from Cyclic Codes of Length 151

Among all cyclic codes in Table B.1, those of length 151 have minimum distance that were found to have the highest number of matches against Brouwer's (1998) lower-bounds. This shows that binary cyclic codes of length 151 are indeed good codes. Since 151 is a prime, cyclic codes of this length are

special as all of the irreducible factors of $x^{151} - 1$, apart from $1 + x$, have a fixed degree of 15. Having a fixed degree implies that duadic codes (Leon et al.; 1984), which include quadratic residue codes, also exist for this length. Due to their large minimum distance, they are good candidate component codes for Constructions X and XX.

4.2 ***Definition (Chain of Cyclic Codes).*** A pair of cyclic codes, $C_1 = [n, k_1, d_1]$ and $C_2 = [n, k_2, d_2]$ where $k_1 > k_2$, is nested, denoted $C_1 \supset C_2$, if all roots of C_1 are contained in C_2. Here the roots refer to those of the generator polynomial. By appropriate arrangement of their roots, cyclic codes of the same length may be partitioned into a sequence of cyclic codes $C_1 \supset C_2 \supset \ldots \supset C_t$. This sequence of codes is termed a chain of cyclic codes.

Given all cyclic codes of the same length, it is important to order the roots of these cyclic codes so that optimum chain can be obtained. For all cyclic codes of length 151 given in Table B.1, whose generator polynomial contains $1 + x$ as a factor, an ordering of roots (excluding the conjugate roots) shown in Table 4.2 results in an optimum chain arrangement. Here β is a primitive 151st root of unity. Similarly, a chain which contains cyclic codes, whose generator polynomial does not divide $1 + x$, can also be obtained.

Table 4.2: Order of β in an optimum chain of $[151, k_i, d_i]$ cyclic codes

i	k_i	d_i	Roots of $g(x)$, excluding conjugate roots									
1	150	2	β^0									
2	135	6	β^0	β^1								
3	120	8	β^0	β^1	β^3							
4	105	14	β^0	β^1	β^3	β^5						
5	90	18	β^0	β^1	β^3	β^5		β^{11}				
6	75	24	β^0	β^1	β^3	β^5		β^{11}	β^{15}			
7	60	32	β^0	β^1	β^3	β^5		β^{11}	β^{15}			β^{37}
8	45	36	β^0	β^1	β^3	β^5		β^{11}	β^{15}	β^{23}		β^{37}
9	30	48	β^0	β^1	β^3	β^5		β^{11}	β^{15}	β^{23}	β^{35}	β^{37}
10	15	60	β^0	β^1	β^3	β^5	β^7	β^{11}	β^{15}	β^{23}	β^{35}	β^{37}

All the constituent codes in the chain $C_1 \supset C_2 \supset \ldots \supset C_{10}$ of Table 4.2 are cyclic. Following Grassl (2001), chain of non-cyclic subcodes may also be constructed after a chain of cyclic codes. This is because for a given generator matrix of an $[n, k, d]$ cyclic code (not necessary in row-echelon form), removing the last i rows of this matrix will produce an $[n, k - i, \geq d]$ code which is not necessary cyclic. As a consequence, with respect to Table 4.2, there exists $[151, k, d]$ linear codes, for $15 \leq k \leq 150$.

Each combination of pairs of codes in the $[151, k, d]$ chain is a nested pair which can be used as component codes for Construction X to produce another linear code. There is a chance that the minimum distance of the resulting linear code is larger than that of the best-known codes for the same length and dimension. In order to find the existence of such case, the following approach has been taken. There are $\binom{150-15+1}{2} = \binom{136}{2}$ distinct pair of codes in the above chain of linear codes, and for each pair say $C_1 = [n, k_1, d_1] \supset C_2 = [n, k_2, d_1]$, is combined using Construction X with an auxiliary code $\mathcal{A}$, which is an $[n', k_1 - k_2, d']$ best-known linear code. The minimum distance of the

resulting code C_X is then compared to that of the best-known linear code for the same length and dimension to check for improvement. Two improvements were obtained and they are tabulated in in the top half of Table 4.3.

In the case when $k_1 - k_2$ is small, the minimum-distance of C_1 i.e. d_1, obtained from a chain of linear codes, can be unsatisfactory. It is possible to improve d_1 by augmenting C_1 with a vector v of length n. i.e. add v as an additional row in G_{C_2}. In finding a vector v that can maximise the minimum distance of the enlarged code, the following procedure is adopted. Choose a code $C_2 = [n, k_2, d_2]$ that has sufficiently high minimum distance. Assuming that G_{C_2} is in reduced-echelon format, generate a vector v which satisfies the following conditions:

1. $v_i = 0$ for $0 \leq i \leq k-1$ where v_i is the ith element of v,
2. $\mathrm{wt}_H(v) > d_1$, and
3. $\mathrm{wt}_H(v + G_r) > d_1$ for all $r \in \{0, 1, \ldots, k_2 - 1\}$ where $G_{C_2,r}$ denotes the rth row of G_{C_2}.

The vector v is then appended to G_{C_2} as an additional row. Compute the minimum distance of the resulting code using Algorithm 4.1. A threshold is applied during the minimum distance evaluation and a termination is called whenever: $d_{ub} \leq d_1$, in which case a different v is chosen and Algorithm 4.1 is restarted; or $d_1 < d_{ub} \leq d_{lb}$ which means that an improvement has been found.

Using this approach, two new linear codes are found, namely $[151, 77, 20]$ and $[151, 62, 27]$, which have higher minimum distance than the corresponding codes obtained from a chain of nested cyclic codes. These two codes are obtained by starting from the cyclic codes $[151, 76, 23]$–which has roots $\{\beta, \beta^5, \beta^{15}, \beta^{35}, \beta^{37}\}$ and $[151, 61, 31]$–which has roots $\{\beta, \beta^3, \beta^5, \beta^{11}, \beta^{15}, \beta^{37}\}$, respectively and therefore, $[151, 77, 20] \supset [151, 76, 23]$ and $[151, 62, 27] \supset [151, 61, 31]$. The second half of Table 4.3 shows the ingredients of these new codes.

Note that, when searching for the $[151, 62, 27]$ code, the fact that the $[152, 61, 32]$ code obtained by extending the $[151, 61, 31]$ cyclic code is doubly even, is exploited. The additional vector v is chosen such that extending the enlarged code $[151, 62, d_1]$ yields again a doubly-even code. This implies the congruence $d_1 = 0, 3 \bmod 4$ for the minimum distance of the enlarged code. Hence, it is sufficient to establish a lower bound $d_{lb} = 25$ using Algorithm 4.1 to show that $d_1 \geq 27$.

Table 4.3: New binary codes from Construction X and cyclic codes of length 151

C_1	C_2	$\mathcal{A}$	C_X
Using chain of linear codes			
$[151, 72, 24]$	$[151, 60, 32]$	$[23, 12, 7]$	$[174, 72, 31]$
$[151, 60, 32]$	$[151, 45, 36]$	$[20, 15, 3]$	$[171, 60, 35]$
Using an improved subcode			
$[151, 77, 20]$	$[151, 76, 23]$	$[3, 1, 3]$	$[154, 77, 23]$
$[151, 62, 27]$	$[151, 61, 31]$	$[4, 1, 4]$	$[155, 62, 31]$

What is even more, two different codes are also found, namely $C_2 = [151, 62, 27] \subset C_1$ and $C_3 = [151, 62, 27] \subset C_1$, where $C_1 = [151, 63, 23]$ and $C_4 = C_2 \cap C_3 = [151, 61, 31]$. Using Construction XX, a $[159, 63, 31]$ code is obtained, see Table 4.4.

Table 4.4: New binary code from Construction XX and cyclic codes of length 151

C_1	C_2	C_3	$C_4 = C_2 \cap C_3$	$\mathcal{A}_1$	$\mathcal{A}_2$	C_{XX}
$[151, 63, 23]$	$[151, 62, 27]$	$[151, 62, 27]$	$[151, 61, 31]$	$[4, 1, 4]$	$[4, 1, 4]$	$[159, 63, 31]$

4.5.2 New Binary Codes from Cyclic Codes of Length ≥ 199

It is known from Table 4.1 that there exists a $[199, 100, 31]$ cyclic code. The extended code, obtained by annexing an overall parity-check bit, is a $[200, 100, 32]$ doubly-even self-dual code. As the name implies, being self-dual we know that the dual code has minimum distance 32. Using Construction Y1 (Theorem 4.3), a $[168, 69, 32]$ new binary code is obtained. The minimum distance of the $[168, 69]$ previously considered best-known binary linear code is 30.

Consider cyclic codes of length 205, in addition to a $[205, 61, 46]$ cyclic code (see Table 4.1), there also exists a $[205, 61, 45]$ cyclic code which contains a $[205, 60, 48]$ cyclic code as its even-weight subcode. Applying Construction X (Theorem 4.1) to $[205, 61, 45] \supset [205, 60, 48]$ pair of cyclic codes with a repetition code of length 3 as the auxiliary code, a $[208, 61, 48]$ new binary linear code is constructed, which improves Brouwer's lower-bound by 2.

Furthermore, it was also found that the dual codes of the $[255, 65, 63]$ cyclic code in Table 4.1 and that of its $[255, 64, 64]$ even weight subcode both have minimum distance of 8. Applying Construction Y1 (Theorem 4.3), new binary linear codes $[247, 57, 64]$ and $[247, 58, 63]$, which improve Brouwer's lower-bounds by 2 and 1 respectively, are obtained.

4.6 Summary

- In the search for error-correcting codes with large minimum distance, an efficient algorithm to compute the exact minimum distance of a linear code is important. The evolution of various algorithms to evaluate the minimum distance of a binary linear code, from the naive approach to the efficient Zimmermann's (1996) approach, has been discussed. In addition to these algorithms, Chen's (1969) approach to compute minimum distance of binary cyclic codes has also been described.

- The core of a minimum distance evaluation algorithm is on codeword enumeration. As the weight of the information vector is increased, the number of codewords required grow exponentially. Zimmermann's (1996) algorithm may be improved by omitting generator matrices with overlapping information sets that never contribute to the lower-bound throughout the enumeration; and also early termination in the event of a new minimum distance is found that meets the lower-bound value of the previous enumeration step. In additions, if the code considered has every codeword weight divisible by 2 or 4, the number of codewords that need to be enumerated can also reduced.

- With some simple modification, Chen's (1969) and Zimmermann's (1996) algorithms can also be used to collect and hence, count all codewords of a given weight.

- Given a generator matrix containing an information set, codewords may be efficiently generated by taking linear combinations of rows of this matrix. This implies the faster combinations

can be generated, the less time the minimum distance evaluation algorithm will take. One of such efficient algorithms to generate combinations is called the revolving-door algorithm. This algorithm has a nice property that allows the enumeration of combinations to be easily parallelised. This property has been discussed in detail in this chapter.

- Having an efficient minimum distance computation algorithm, which can also be parallelised, allows extension of the work by Chen (1970), Promhouse and Tavares (1978), and Schomaker and Wirtz (1992), in evaluating the minimum distance of cyclic codes. The highest minimum distance attainable by all binary cyclic codes of odd length from 129 to 189 has been obtained. We found that none of these cyclic codes has minimum distance that exceeds the minimum distance of the best-known linear codes of the same length and dimension, which is given as the lower-bound in Brouwer (1998), but there are 134 cyclic codes that meet the lower-bound, see Appendix B.

- Having an efficient parallelise-able minimum distance computation algorithm also allows the search for the existence of binary cyclic codes of length longer than 189 which are improvements to Brouwer's (1998) lower-bound. It has been found that there are 37 of these cyclic codes, namely

$$[195,66,42],[195,67,41],[195,68,40],\ [195,69,39],[195,73,38],[195,74,38],$$
$$[195,75,37],[195,78,36],[199,99,32],[199,100,32],[205,60,48],[205,61,46],$$
$$[215,70,46],[215,71,46],[223,74,48],\ [223,75,47],[229,76,48],[233,58,60],$$
$$[233,59,60],[255,48,76],[255,49,75],\ [255,50,74],[255,51,74],[255,52,72],$$
$$[255,53,72],[255,54,70],[255,55,70],\ [255,56,68],[255,57,68],[255,58,66],$$
$$[255,60,66],[255,62,66],[255,63,65],\ [255,64,64],[255,65,63],$$

- From the cyclic codes above, using Construction X to lengthen code and Construction Y1 to shorten code, four additional improvements to Brouwer's (1998) lower-bound are found, namely

$$[168,69,32],[208,61,48],[247,57,64],[247,58,63].$$

- Five new linear codes, which are derived from cyclic codes of length 151, have also been constructed. These new codes, which are produced by Constructions X and XX, are

$$[154,77,23],[155,62,31],[159,63,31],[171,60,35],[174,72,31].$$

- Given an $[n,k,d]$ code C, where d is larger than the minimum distance of the best known linear code of the same n and k, it is possible to obtain more codes, whose minimum distance is still larger than that of the corresponding best known linear code, by recursively extending (annexing parity-check), puncturing and/or shortening C. For example, consider the new code $[168,69,32]$; by annexing parity-check bit $[168+i,69,32]$, for $1 \leq i \leq 3$, new codes are obtained; by puncturing a $[167,69,31]$ new code is obtained; by shortening $[168-i,69-i,32]$, for $1 \leq i \leq 5$, new codes are obtained. More improvements are also obtained by shortening the extended and

punctured codes. Overall, using the new codes obtained in this chapter, there are 901 binary linear codes which improve Brouwer's (1998) lower-bound. The improved lower-bounds are tabulated in Tables C.1–C.5 in Appendix C.

5 Double Circulant Codes based on Primes

Parts of this chapter appear in the following conference proceedings:

1. Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006), "On the efficient codewords counting algorithm and the weight distribution of the binary quadratic double-circulant codes", in *Proceedings IEEE Information Theory Workshop*, Chengdu, China, pp. 42–46

2. Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006), "Some results on the weight distributions of the binary double-circulant codes based on primes", in *Proceedings 10th IEEE International Conference on Communication Systems*, Singapore.

5.1 Introduction

Binary self-dual codes form an important class of codes due to their powerful error-correcting capabilities and their rich mathematical structure. As such, this family of codes has been a subject of extensive research for many years. Much of this work is on their classification and the search for the extremal codes (Rains and Sloane; 1998). Many binary self-dual codes are codes with the highest known minimum Hamming distance. Recently, van Dijk et al. (2005) constructed two inequivalent binary self-dual codes of length 160 that have higher minimum Hamming distance than the previously known half-rate code of that length.

Closely related to the self-dual codes are the double-circulant codes. Many good binary self-dual codes can be constructed in double-circulant form. An interesting family of binary, double-circulant codes, which includes self-dual and formally self-dual codes, is the family of codes based on primes. A classic paper for this family was published by Karlin (1969) in which double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8 were considered. Moore's (1976) PhD work investigated the class which is congruent to 3 modulo 8, and his work was later extended by Gulliver and Senkevitch (1999) to longer codes. An extensive discussion on these two types of circulant is also given by MacWilliams and Sloane (1977). The prime-based double-circulant codes can also be constructed over non binary fields, e.g. see Pless (1972) and Beenker (1984) for $\mathbb{F}_3$, and Gaborit (2002) for the generalisation to prime fields.

This chapter considers the binary double-circulant codes based on primes, especially on their Hamming weight distributions, and the remaining of this chapter is organised as follows. Section 5.2 introduces the notation and gives a review of double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8. Section 5.3 describes the construction of double-circulant codes for these primes and Section 5.4 presents an improved algorithm to compute the minimum Hamming distance and also the number of codewords of a given Hamming weight for certain double-circulant

codes. The algorithm presented in this section requires the enumeration of less codewords than that of the commonly used technique (e.g. Gaborit et al.; 2005; van Dijk et al.; 2005). Section 5.5 considers the Hamming weight distribution of the double-circulant codes based on primes. A method to provide an independent verification to the number of codewords of a given Hamming weight in these double-circulant codes is also discussed in this section. In the last section of this chapter, Section 5.6, a probabilistic method—based on its automorphism group, to determine the minimum Hamming distance of these double-circulant codes is described.

Note that, since throughout this chapter, we consider Hamming space only, we shall omit the word 'Hamming' when we refer to Hamming weight and distance.

5.2 Background and Notation

A code C is called *self-dual* if $C = C^{\perp}$, where $C^{\perp}$ is the dual of C. There are two types of self-dual code: *doubly-even* or Type II if the weight of every codeword is divisible by 4; *singly-even* or Type I if the weight of every codeword is divisible by 2. Furthermore, the code length of a Type II code is divisible by 8. In addition to self-dual codes, there also exists formally self-dual, abbreviated fsd, codes. A code is called fsd if $C \neq C^{\perp}$ but $A_C(z) = A_{C^{\perp}}(z)$. A self-dual, or fsd, code is called *extremal* if its minimum distance is the highest possible for the given parameters. The bound of the minimum distance of the extremal codes is (Rains and Sloane; 1998)

$$d \leq 4\left\lfloor \frac{n}{24} \right\rfloor + 4 + \epsilon \tag{5.1}$$

where

$$\epsilon = \begin{cases} -2 & \text{if } C \text{ is Type-I with } n = 2, 4, \text{ or } 6, \\ 2, & \text{if } C \text{ is Type-I with } n \equiv 22 \pmod{24}, \text{ or} \\ 0, & \text{if } C \text{ is Type-I or Type-II with } n \not\equiv 22 \pmod{24}. \end{cases} \tag{5.2}$$

For an fsd code, the minimum distance of the extremal case is upper-bounded by (Gaborit et al.; 2005)

$$d \leq 2\left\lfloor \frac{n}{8} \right\rfloor + 2. \tag{5.3}$$

With this upper-bound, extremal fsd code only exists for $n \leq 30$ and $n \neq 16$ and $n \neq 26$ (Huffman and Pless; 2003). Databases of best-known, not necessary extremal, self-dual codes are given in Rains and Sloane (1998) and Gaborit and Otmani (2007). A table of binary self-dual double-circulant codes is also provided in Rains and Sloane (1998).

As a class, double-circulant codes are $[n, k]$ linear codes, where $k = n/2$, whose generator matrix G consists of two circulant matrices.

5.1 Definition (Circulant Matrix). A circulant matrix is a square matrix in which each row is a cyclic shift of the adjacent row. In addition, each column is also a cyclic shift of the adjacent column and the number of non zeros per column is equal to those per row.

A circulant matrix is completely characterised by a polynomial formed by its first row,

$$r(x) = \sum_{i=0}^{m-1} r_i x^i,$$

which is called the *defining polynomial*. Note that the algebra of polynomials modulo $x^m - 1$ is isomorphic to that of circulants (MacWilliams and Sloane; 1977). Let the polynomial $r(x)$ to have a maximum degree of $m-1$, the corresponding circulant matrix $\boldsymbol{R}$ is an $m \times m$ square matrix of the form

$$\boldsymbol{R} = \begin{bmatrix} r(x) \pmod{x^m - 1} \\ xr(x) \pmod{x^m - 1} \\ \vdots \\ x^i r(x) \pmod{x^m - 1} \\ \vdots \\ x^{m-1} r(x) \pmod{x^m - 1} \end{bmatrix} \tag{5.4}$$

where the polynomial in each row can be represented by an m-dimensional vector, which contains the coefficients of the corresponding polynomial.

Double-circulant codes can be put into two classes, namely *pure*, and *bordered* double-circulant codes, whose generator matrices G_p and G_b are shown in (5.5a)

$$G_p = \left[\begin{array}{c|c} I_k & \boldsymbol{R} \end{array}\right] \tag{5.5a}$$

and (5.5b)

$$G_b = \left[\begin{array}{c|c|c} & 1 \;\ldots\; 1 & \alpha \\ \hline I_k & \boldsymbol{R} & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \end{array}\right] \tag{5.5b}$$

respectively. Here, I_k is a k-dimensional identity matrix, and $\alpha \in \{0, 1\}$.

Two binary linear codes, $\mathscr{A}$ and $\mathscr{B}$, are *equivalent* if there exists a permutation π on the coordinates of the codewords which maps the codewords of $\mathscr{A}$ onto codewords of $\mathscr{B}$. These two codes shall be written as $\mathscr{B} = \pi(\mathscr{A})$. If π transforms C into itself, then it is said that π fixes the code, and the set of all permutations of this kind form the automorphism group of C, denoted as Aut(C). MacWilliams and Sloane (1977) gives some necessary but not sufficient conditions on the equivalence of double-circulant codes, which are restated for convenience in the lemma below.

5.1 Lemma. (cf. (MacWilliams and Sloane; 1977, Problem 7, Ch. 16)) Let $\mathscr{A}$ and $\mathscr{B}$ be double-circulant codes with generator matrices $[I_k|A]$ and $[I_k|B]$ respectively. Let the polynomials $a(x)$ and $b(x)$ be the defining polynomials of A and B. The codes $\mathscr{A}$ and $\mathscr{B}$ are equivalent if any of the following conditions holds:

i) $B = A^T$, or

ii) $b(x)$ is the reciprocal of $a(x)$, or

iii) $a(x)b(x) = 1 \pmod{x^m - 1}$, or

iv) $b(x) = a(x^u)$ where m and u are relatively prime.

Proof.

i) It can be clearly seen that $b(x) = \sum_{i=0}^{m-1} a_i x^{m-i}$. It follows that $b(x) = \pi(a(x))$ where $\pi : i \to m - i \pmod m$ and hence $\mathscr{A}$ and $\mathscr{B}$ are equivalent.

ii) Given a polynomial $a(x)$, its reciprocal polynomial can be written as $\ddot{a}(x) = \sum_{i=0}^{m-1} a_i x^{m-i-1}$. It follows that $\ddot{a}(x) = \pi(a(x))$ where $\pi : i \to m - i - 1 \pmod m$.

iii) Consider the code $\mathscr{A}$, since $b(x)$ has degree less then m, it can be one of the possible data patterns and in this case, the codeword of $\mathscr{A}$ has the form $|b(x)|1|$. Clearly, this is a permuted codeword of $\mathscr{B}$.

iv) If $(u, m) = 1$, then $\pi : i \to iu \pmod m$ is a permutation on $\{0, 1, \ldots, m-1\}$. So $b(x) = a(x^u)$ is in the code $\pi(\mathscr{A})$. ∎

5.3 Code Construction

Consider an $[n, k, d]$ pure double-circulant code, it can be seen that for a given user message, represented by a polynomial $u(x)$ of degree at most $k - 1$, a codeword of the double-circulant code has the form $(u(x)|u(x)r(x) \pmod{x^m - 1})$. The defining polynomial $r(x)$ characterises the resulting double-circulant code. Before the choices of $r(x)$ is discussed, consider the following lemmas and corollary.

5.2 Lemma. Let $a(x)$ be a polynomial over $\mathbb{F}_2$ of degree at most $m - 1$, i.e. $a(x) = \sum_{i=0}^{m-1} a_i x^i$ where $a_i \in \{0, 1\}$. The weight of the polynomial $(1 + x)a(x) \pmod{x^m - 1}$, denoted by $\mathrm{wt}_H((1 + x)a(x))$, is even.

Proof. Let $w = \mathrm{wt}_H(a(x)) = \mathrm{wt}_H(xa(x))$ and $S = \{i : a_{i+1 \bmod m} = a_i \neq 0,\ 0 \leq i \leq m - 1\}$.

$$\begin{aligned}\mathrm{wt}_H((1 + x)a(x)) &= \mathrm{wt}_H(a(x)) + \mathrm{wt}_H(xa(x)) - 2|S| \\ &= 2(w - |S|),\end{aligned}$$

which is even. ∎

5.3 Lemma. An $m \times m$ circulant matrix R with defining polynomial $r(x)$ is non-singular if and only if $r(x)$ is relatively prime to $x^m - 1$.

Proof. Suppose that $r(x)$ is relatively prime to $x^m - 1$, i.e. $(r(x), x^m - 1) = 1$. From the extended Euclid algorithm, it follows that, for some unique polynomials $a(x)$ and $b(x)$, $r(x)a(x)+(x^m-1)b(x) = 1$, which is equivalent to $r(x)a(x) = 1 \pmod{x^m - 1}$ and therefore $r(x)$, under polynomial algebra modulo $x^m - 1$ (and hence R) is invertible, where $r(x)^{-1} = a(x)$. Conversely, for non relatively prime cases, $(r(x), x^m - 1) = t(x)$, where $t(x) \neq 1$. This means $r(x)^{-1}$ does not exists since $r(x)a(x) = t(x) \pmod{x^m - 1}$. □

5.1 Corollary. From Lemma 5.3,

i) if R is non-singular, R^{-1} is an $m \times m$ circulant matrix with defining polynomial $r(x)^{-1}$, and

ii) the weight of $r(x)$ or $r(x)^{-1}$ is odd.

Proof. The proof for the first case is obvious from the proof of Lemma 5.3. For the second case, if $(1 + x) \mid r(x)$ (resp. $(1 + x) \mid r(x)^{-1}$), then $(r(x), x^m - 1) = 1 + x$ (resp. $(r(x)^{-1}, x^m - 1) = 1 + x$). By Lemma 5.2, this rules out the possibility of even weight defining polynomial. Thus, $\mathrm{wt}_H(r(x))$ (resp. $\mathrm{wt}_H(r(x)^{-1})$) is necessary odd. □

5.2 Definition. Let α be a generator of $\mathbb{F}_p$ where p be an odd prime, $r \equiv \alpha^2 \pmod{p}$ is called a quadratic residue modulo p and so is $r^i \in \mathbb{F}_p$ for some integer i. Because the element α has (multiplicative) order $p - 1$ over $\mathbb{F}_p$, $r = \alpha^2$ has order $\frac{1}{2}(p - 1)$. A set of quadratic residue modulo p, Q, and non quadratic residue modulo p, N, are defined as

$$\begin{aligned} Q &= \{r, r^2, \ldots, r^i, \ldots, r^{\frac{p-3}{2}}, r^{\frac{p-1}{2}} = 1\} \\ &= \{\alpha^2, \alpha^4, \ldots, \alpha^{2i} \ldots, \alpha^{p-3}, \alpha^{p-1} = 1\}. \end{aligned} \tag{5.6a}$$

and

$$\begin{aligned} N &= \{n : \forall n \in \mathbb{F}_p,\ n \notin Q \text{ and } n \neq 0\} \\ &= \{nr, nr^2, \ldots, nr^i, \ldots, nr^{\frac{p-3}{2}}, n\} \\ &= \{\alpha^{2i+1} \ : \ 0 \leq i \leq \frac{p-3}{2}\} \end{aligned} \tag{5.6b}$$

respectively*.

*It is obvious that, if $r \in Q$, $r = \alpha^c$ for even c; and if $n \in N$, $n = \alpha^c$ for odd c. Hence, if $n \in N$ and $r \in Q$, $nr = \alpha^{2i}\alpha^{2j+1} = \alpha^{2(i+j)+1} \in N$. Similarly, $rr = \alpha^{2i}\alpha^{2j} = \alpha^{2(i+j)} \in Q$ and $nn = \alpha^{2i+1}\alpha^{2j+1} = \alpha^{2(i+j+1)} \in Q$.
Note that

- $2 \in Q$ if $p \equiv \pm 1 \pmod 8$, and $2 \in N$ if $p \equiv \pm 3 \pmod 8$
- $-1 \in Q$ if $p \equiv 1 \pmod 8$ or $p \equiv -3 \pmod 8$, and $-1 \in N$ if $p \equiv -1 \pmod 8$ and $p \equiv 3 \pmod 8$

5.4 Lemma. Let p be an odd prime, then

i) $p \mid 2^{p-1} - 1$, and

ii) the integer q for $pq = 2^{p-1} - 1$ is odd.

Proof. From Fermat's little theorem, it is known that for any integer a and an odd prime p, $a^{p-1} \equiv 1 \pmod{p}$. This is equivalent to $a^{p-1} - 1 = pq$ for some integer q. Let $a = 2$, it follows that

$$q = \frac{2^{p-1} - 1}{p}$$

which is clearly odd since neither denominator nor numerator contains 2 as a factor. □

5.5 Lemma. Let p be a prime and $j(x) = \sum_{i=0}^{p-1} x^i$, then

$$(1+x)^{2^{p-1}-1} = 1 + j(x) \bmod (x^p - 1).$$

Proof. The polynomial $(1+x)^{2^{p-1}-1}$ can be written as

$$\begin{aligned}(1+x)^{2^{p-1}-1} &= \frac{(1+x)^{2^{p-1}}}{1+x} = \frac{1+x^{2^{p-1}}}{1+x} \\ &= \sum_{i=0}^{2^{p-1}-1} x^i .\end{aligned}$$

From Lemma 5.4, it is known that the integer $q = (2^{p-1} - 1)/p$ and is odd. In terms of $j(x)$, $\sum_{i=0}^{2^{p-1}-1} x^i$ may be written as follows

$$\begin{aligned}\sum_{i=0}^{2^{p-1}-1} x^i = 1 + x\underbrace{\left(1 + x + \ldots + x^{p-1}\right)}_{j(x)} + x^{p+1}\underbrace{\left(1 + x + \ldots + x^{p-1}\right)}_{j(x)} + \ldots + \\ x^{(q-3)p+1}\underbrace{\left(1 + x + \ldots + x^{p-1}\right)}_{j(x)} + x^{(q-2)p+1}\underbrace{\left(1 + x + \ldots + x^{p-1}\right)}_{j(x)} + \\ x^{(q-1)p+1}\underbrace{\left(1 + x + \ldots + x^{p-1}\right)}_{j(x)} \\ = 1 + \underbrace{xj(x)(1+x^p) + x^{2p+1}j(x)(1+x^p) + \ldots + x^{(q-3)p+1}j(x)(1+x^p)}_{J(x)} + x^{(q-1)p+1}j(x).\end{aligned}$$

Since $(1 + x^p) \pmod{x^p - 1} = 0$ for binary polynomial, it is clear that $J(x) = 0$ and

$$\sum_{i=0}^{2^{p-1}-1} x^i = 1 + xx^{(q-1)p}j(x) \pmod{x^p - 1}.$$

Because $x^{ip} \pmod{x^p - 1} = 1$,

$$\sum_{i=0}^{2^{p-1}-1} x^i = 1 + xj(x) \pmod{x^p - 1}$$
$$= 1 + j(x) \pmod{x^p - 1}.$$

□

In the following, for the purpose of designing $[2p, p, d]$ pure double-circulant codes, the choices of the defining polynomial $r(x)$ of prime degree only is considered. Let $p = m$ be a prime, there are 2^p polynomials (including the zero polynomial) over $\mathbb{F}_2$ of degree at most $p - 1$. Let F denote the set of these polynomials, excluding the zero polynomial and the all-ones polynomial, denoted by $j(x) = \sum_{i=0}^{p-1} x^i$. The set F can split into $2^{p-1} - 1$ polynomials of odd weight and and $2^{p-1} - 1$ polynomials of even weight, denoted by F_1 and F_2 respectively.

For $p \equiv \pm 1 \pmod 8$, the polynomial $x^p - 1 = (1 + x)\prod_{i=1}^{t} f_i(x)$, where the polynomials $f_i(x)$ for $1 \leq i \leq t$ and $t \geq 2$, have the same degree, say e. As a consequence, not all polynomials in F are relatively prime to $x^p - 1$. To design a $[2p, p, d]$ pure double-circulant code, $1 + \mathrm{wt}_H(r(x))$ must be at least d and if $r(x) = j(x) = \prod_{i=1}^{t} f_1(x)$, a $[2p, p, 2]$ pure double-circulant code is obtained—the minimum distance of 2 appears when $u(x) = 1 + x$. Another type of double-circulant construction, both pure and bordered, exists for these primes, see Section 5.3.1.

The case for $p \equiv \pm 3 \pmod 8$ is more interesting. For these primes, $x^p - 1 = (1 + x)j(x)$ and as a result, the set F_1 contains all binary polynomials which are relatively prime to $x^p - 1$, and F_2 contains all binary polynomials which contain $(1 + x)$ as a factor. The defining polynomial $r(x)$ can then be chosen from F_1, F_2 or $r(x) = j(x)$. Like in the previous prime case, the latter choice results in a $[2p, p, 2]$ double-circulant code and for other choices of $r(x)$ the defining polynomial can be constrained for a given minimum distance.

For $r(x) \in F_1$, it is known that F_1 forms a multiplicative group of order $|F_1| = 2^{p-1} - 1$ with a generator $a(x)$ and identity 1. This means that $a(x)^{|F_1|} = 1$, where $|F_1|$ is the smallest positive integer that satisfies this equality, and that $r(x)$ may be represented as an integer power of $a(x)$, that is $r(x) = a(x)^i$. The polynomial $u(x)r(x) \neq 0 \pmod{x^p - 1}$ for any given $u(x)$. If $\mathrm{wt}_H(u(x))$ is odd, $\mathrm{wt}_H(u(x)r(x)) = \mathrm{wt}_H(a(x)^{j+i})$, hence it is also odd, and the weight of the resulting codeword is even. Similarly, if $\mathrm{wt}_H(u(x))$ is even, the weight of $u(x)r(x)$ is even (see Lemma 5.2) and the resulting codeword weight is also even. Hence, for $r(x) \in F_1$, an even $[2p, p, d]$ pure double-circulant code is obtained.

Note that F_2 also forms a multiplicative group of order $|F_2| = 2^{p-1} - 1$. Let $b(x)$ be a generator of F_2, it can be written that $b(x) = (1+x)a(x)$ where $a(x)$ is a generator of F_1. Any element of F_2 can be represented by some power of $b(x)$; for some integers i and j, $b(x)^i b(x)^j = ((1+x)a(x))^i((1+x)a(x))^j = ((1+x)a(x))^{i+j} \in F_2$. It is known that $|F_1| = |F_2|$ and $a(x)^{|F_2|} = 1$, hence $b(x)^{|F_2|} = ((1 + x)a(x))^{|F_2|} = (1 + x)^{|F_2|}$, which is equivalent to $1 + j(x) \pmod{x^p - 1}$ by Lemma 5.5. The polynomial $1 + j(x)$ can be considered as the identity of this multiplicative group since, for $b(x)^j \in F_2$, $b(x)^j(1 + j(x)) = ((1 + x)a(x))^j(1 + j(x)) = ((1 + x)a(x))^j + ((1 + x)a(x))^j j(x) = ((1 + x)a(x))^j \pmod{x^p - 1} = b(x)^j \pmod{x^p - 1}$. The polynomial $r(x)$ can be represented as an integer power of $b(x)$, $r(x) = b(x)^i$. If $u(x) = j(x)$, the polynomial $u(x)r(x) = j(x)((1 + x)a(x))^i = 0 \pmod{x^p - 1}$ and the resulting

codeword is $j(x)$, which has odd weight. For any other $u(x)$, $\mathrm{wt}_H(u(x)r(x))$ is even and the resulting codeword has odd (resp. even) weight if $\mathrm{wt}_H(u(x))$ is odd (resp. even). Hence, for $r(x) \in F_2$, even and odd $[2p, p, d]$ pure double-circulant codes can be constructed.

A pure double-circulant self-dual code, which has the property that $r(x)^T = r(x^{-1}) = r(x)^{-1}$, can only be constructed if $r(x) \in F_1$. It is known that $2^{p-1} = 1 \pmod p$ for primes $\pm 3 \pmod 8$, and correspondingly $2^{(p-1)/2} = -1 \pmod p$. Consider the self-dual case, if $r(x) = a(x)^z$, $r(x)^T = a(x^{-1})^z = a(x^{2^{(p-1)/2}})^z = a(x)^{z2^{(p-1)/2}}$. Since $r(x)^{-1} = a(x)^{-z} = a(x)^{2^{p-1}-1-z}$, the equality $2^{p-1} - z - 1 = z(2^{p-1/2})$ is obtained, and it follows that

$$z = \frac{2^{p-1} - 1}{2^{(p-1)/2} + 1} = 2^{(p-1)/2} - 1. \tag{5.7}$$

This means that $a(x)^{jz}$, for $1 \leq j \leq |F_1|/z = 2^{(p-1)/2} + 1$, generates $[2p, p, d]$ self-dual pure double-circulant codes. Section 5.3.2 discusses more details of pure double-circulant codes for these primes.

5.3.1 Double-Circulant Codes from Extended Quadratic Residue Codes

The following is a summary of the extended QR codes as double-circulant codes (Karlin; 1969; MacWilliams and Sloane; 1977; Jenson; 1980). Binary QR codes are cyclic codes of length p over $\mathbb{F}_2$. For a given p, there exists four QR codes:

1. $\bar{\mathcal{Q}}_p, \bar{\mathcal{N}}_p$ which are equivalent and have dimension $\frac{1}{2}(p-1)$, and
2. $\mathcal{Q}_p, \mathcal{N}_p$ which are equivalent and have dimension $\frac{1}{2}(p+1)$.

The $[p+1, \frac{1}{2}(p+1), d]$ extended quadratic residue code, denoted by $\hat{\mathcal{Q}}_p$ (resp. $\hat{\mathcal{N}}_p$), is obtained by annexing an overall parity check to $\mathcal{Q}_p$ (resp. $\mathcal{N}_p$). If $p \equiv -1 \pmod 8$, $\hat{\mathcal{Q}}_p$ (resp. $\hat{\mathcal{N}}_p$) is Type II; otherwise it is fsd.

It is well-known that† $\mathrm{Aut}(\hat{\mathcal{Q}}_p)$ contains the projective special linear group denoted by $\mathrm{PSL}_2(p)$ (MacWilliams and Sloane; 1977). If r is a generator of the cyclic group Q then $\sigma : i \to ri \pmod p$ is a member of $\mathrm{PSL}_2(p)$. Given $n \in N$, the cycles of σ can be written as

$$(\infty)(n, nr, nr^2, \ldots, nr^t)(1, r, r^2, \ldots, r^t)(0) \tag{5.8}$$

where $t = \frac{1}{2}(p-3)$. Due to this property, G, the generator matrix of $\hat{\mathcal{Q}}_p$, can be arranged into circulants as shown in (5.9)

$$G = \begin{array}{c|c|ccccc|c|ccccc|c|} & \infty & n & nr & \ldots & nr^{t-1} & nr^t & 1 & r & \ldots & r^{t-1} & r^t & 0 \\ \hline \infty & 1 & 1 & 1 & \ldots & 1 & 1 & 1 & 1 & \ldots & 1 & 1 & 1 \\ \hline \beta & 0 & & & & & & & & & & & 1 \\ \beta r & 0 & & & & & & & & & & & 1 \\ \vdots & \vdots & & & & L & & & & R & & & \vdots \\ \beta r^{t-1} & 0 & & & & & & & & & & & 1 \\ \beta r^t & 0 & & & & & & & & & & & 1 \\ \hline \end{array} \tag{5.9}$$

†Since $\hat{\mathcal{Q}}_p$ and $\hat{\mathcal{N}}_p$ are equivalent, considering either one is sufficient.

where L and R are $\frac{1}{2}(p-1) \times \frac{1}{2}(p-1)$ circulant matrices. The rows $\beta, \beta r, \ldots, \beta r^t$ in the above generator matrix contain $\bar{e}_\beta(x), \bar{e}_{\beta r}(x), \ldots, \bar{e}_{\beta r^t}(x)$, where $\bar{e}_i(x) = x^i e(x)$ whose coordinates are arranged in the order of (5.8). Note that, (5.9) can be transformed to (5.5b) as follows

$$\begin{bmatrix} 1 & J \\ 0^T & L^{-1} \end{bmatrix} \times \left[\begin{array}{cc|cc} 1 & J & J & 1 \\ 0^T & L & R & J^T \end{array}\right] = \left[\begin{array}{cc|cc} 1 & J + \mathrm{w}(L^T) & J + \mathrm{w}(R^T) & \frac{1}{2}(p+1) \\ 0^T & I_{\frac{1}{2}(p-1)} & L^{-1}R & \mathrm{w}(L^{-1})^T \end{array}\right] \tag{5.10}$$

where J is an all-ones vector and $\mathrm{w}(A) = [\mathrm{wt}_H(A_0) \pmod 2, \mathrm{wt}_H(A_1) \pmod 2, \ldots]$, A_i being the ith row vector of matrix A. The multiplication in (5.10) assumes that L^{-1} exists and following Corollary 5.1, $\mathrm{wt}_H(l^{-1}(x)) = \mathrm{wt}_H(l(x))$ is odd. Therefore, (5.10) becomes

$$G = \left[\begin{array}{c|c|c} & J + \mathrm{w}(R^T) & \frac{1}{2}(p+1) \\ \hline I_{\frac{1}{2}(p+1)} & L^{-1}R & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \end{array}\right] \tag{5.11}$$

In relation to (5.9), consider extended QR codes for primes

1. $p = 8m + 1$, the idempotent $e(x) = \sum_{n \in N} x^n$ and $\beta \in N$. Following MacWilliams and Sloane (1977, Theorem 24, Ch. 16), it is known that $\bar{e}_{\beta r^i}(x)$ where $\beta r^i \in N$, for $0 \leq i \leq t$, contains $2m+1$ quadratic residues modulo p (including 0) and $2m - 1$ non quadratic residues modulo p. As a consequence, $\mathrm{wt}_H(r(x))$ is even, implying $\mathrm{w}(R^T) = 0$ and $r(x)$ is not invertible (cf. Corollary 5.1); and $\mathrm{wt}_H(l(x))$ is odd and $l(x)$ may be invertible over polynomial modulo $x^{\frac{1}{2}(p-1)} - 1$ (cf. Corollary 5.1). Furthermore, referring to (5.5b), it is clear that $\alpha = \frac{1}{2}(p+1) = 4m+1 = 1 \bmod 2$.

2. $p = 8m - 1$, the idempotent $e(x) = 1 + \sum_{n \in N} x^n$ and $\beta \in Q$. Following MacWilliams and Sloane (1977, Theorem 24, Ch. 16), assuming that there exists a set S containing 0 and $4m - 1$ non quadratic residues modulo p, the set $\beta + S$ contains $2m + 1$ quadratic residues modulo p (including 0) and $2m - 1$ non quadratic residues modulo p. It follows that $\bar{e}_{\beta r^i}(x)$ where $\beta r^i \in Q$, for $0 \leq i \leq t$, contains $2m$ quadratic residues modulo p (excluding 0), implying that R is singular (cf. Corollary 5.1); and $2m - 1$ non quadratic residues modulo p, implying L^{-1} may exist (cf. Corollary 5.1). Furthermore, $\mathrm{w}(R^T) = 0$ and referring to (5.5b), it can be seen that $\alpha = \frac{1}{2}(p+1) = 4m = 0 \bmod 2$.

For many $\hat{Q}_p$, L is invertible and Karlin (1969) has shown that $p = 73, 97, 127, 137, 241$ are the known cases where the canonical form (5.5b) cannot be obtained. Consider the case for $p = 73$, with $\beta = 5 \in N$, the defining polynomial of the left circulant $l(x)$ is given by

$$l(x) = x^2 + x^3 + x^4 + x^5 + x^6 + x^{11} + x^{15} + x^{16} + x^{18} + x^{20} + x^{21} + x^{25} + x^{30} + x^{31} + x^{32} + x^{33} + x^{34}.$$

The polynomial $l(x)$ contains some irreducible factors of $x^{\frac{1}{2}(p-1)} - 1 = x^{36} - 1$, i.e. $(l(x), x^{36} - 1) = 1 + x^2 + x^4$, and hence it is not invertible. In addition to form (5.5b), G can also be transformed to (5.5a), and Jenson (1980) has shown that, for $7 \leq p \leq 199$, except $p = 89, 167$, the canonical

form (5.5a) exists.

5.3.2 Pure Double-Circulant Codes for Primes ± 3 Modulo 8

Recall that F_1 is a multiplicative group of order $2^{p-1}-1$ containing all polynomials of odd weight (excluding the all ones polynomial) of degree at most $p-1$, where p is a prime. It is assumed that $a(x)$ is a generator of F_1. For $p \equiv \pm 3 \pmod 8$, the following lemma is obtained.

5.6 Lemma. For $p \equiv \pm 3 \pmod 8$, let the polynomials $q(x) = \sum_{i \in Q} x^i$ and $n(x) = \sum_{i \in N} x^i$. Self-dual pure double-circulant codes with $r(x) = q(x)$ or $r(x) = n(x)$ exist if and only if $p \equiv 3 \pmod 8$.

Proof. Consider $r(x) = q(x)$, for $p \equiv \pm 3 \pmod 8$, $2 \in N$ and hence $q(x)^2 = \sum_{i \in Q} x^{2i} = n(x)$. It is known that $1 + q(x) + n(x) = j(x)$, therefore $q(x)^3 = q(x)^2 q(x) = n(x)q(x) = (1 + q(x) + j(x))q(x) = q(x) + n(x) + j(x) = 1$, i.e. $q(x)^2 = n(x) = q(x)^{-1}$. The polynomial $q(x)^T = q(x^{-1}) = \sum_{i \in Q} x^{-i}$ and for self-dual codes, the condition $q(x)^T = n(x)$ must be satisfied. If $p \equiv -3 \pmod 8$, $-1 \in Q$ and thus $q(x)^T = q(x)$. On the other hand, $-1 \in N$ if $p \equiv 3 \pmod 8$ and thus $q(x)^T = n(x)$. For $r(x) = n(x)$, the same arguments follow. □

Let $\mathscr{P}_p$ denote a $[2p, p, d]$ pure double-circulant code for $p \equiv \pm 3 \pmod 8$ The properties of $\mathscr{P}_p$ can be summarised as follows:

1. For $p \equiv 3 \pmod 8$, since $q(x)^3 = 1$ and $a(x)^{2^{p-1}-1} = 1$, $q(x) = a(x)^{(2^{p-1}-1)/3}$ and $q(x)^T = a(x)^{(2^p-2)/3}$. There are two full-rank generator matrices with mutually disjoint information sets associated with $\mathscr{P}_p$ for these primes. Let G_1 be a reduced echelon generator matrix of $\mathscr{P}_p$, which has the form of (5.5a) with $R = B$ where B is a circulant matrix with defining polynomial $b(x) = q(x)$. The other full-rank generator matrix G_2 can be obtained as follows

$$G_2 = \boxed{B^T} \cdot G_1 = \boxed{B^T \mid I_p}. \tag{5.12}$$

   The self-duality of this pure double-circulant code is obvious from G_2.

2. For $p \equiv -3 \pmod 8$, $(p-1)/2$ is even and hence neither $q(x)$ nor $n(x)$ is invertible, which means that if this polynomial was chosen as the defining polynomial for $\mathscr{P}_p$, there exists only one full-rank generator matrix. However, either $1 + q(x)$ (resp. $1 + n(x)$) is invertible and the inverse is $1 + n(x)$ (resp. $1 + q(x)$), i.e.

$$\begin{aligned}(1 + q(x))(1 + n(x)) &= 1 + q(x) + n(x) + q(x)n(x) \\ &= 1 + q(x) + n(x) + q(x)(1 + j(x) + q(x)) \\ &= 1 + q(x) + n(x) + q(x) + q(x)j(x) + q(x)^2,\end{aligned}$$

   and since $q(x)j(x) = 0$ and $q(x)^2 = n(x)$ under polynomial modulo $x^p - 1$, it follows that

$$(1 + q(x))(1 + n(x)) = 1 \pmod{x^p - 1}.$$

Let G_1 be the first reduced echelon generator matrix, which has the form of (5.5a) where $R = I_p + Q$. The other full-rank generator matrix with disjoint informations sets G_2 can be obtained as follows

$$G_2 = \boxed{I_p + N} \cdot G_1 = \boxed{I_p + N \mid I_p}. \tag{5.13}$$

Since $-1 \in Q$ for this prime, $(I_p+Q)^T = I_p+Q$ implying that the $[2p, p, d]$ pure double-circulant code is fsd, i.e. the generator matrix of $\mathscr{B}_p^{\perp}$ is given by

$$G^{\perp} = \boxed{I_p + Q \mid I_p}.$$

A bordered double-circulant construction of these primes—commonly known as the *quadratic double-circulant* construction, also exists, see Section 5.3.3.

5.3.3 Quadratic Double-Circulant Codes

Let p be a prime that is congruent to ± 3 modulo 8. A $[2(p+1), p+1, d]$ binary quadratic double-circulant code, denoted by $\mathscr{B}_p$, can be constructed using the defining polynomial

$$b(x) = \begin{cases} 1 + q(x) & \text{if } p \equiv 3 \pmod 8, \text{ and} \\ q(x) & \text{if } p \equiv -3 \pmod 8 \end{cases} \tag{5.14}$$

where $q(x) = \sum_{i \in Q} x^i$. Following MacWilliams and Sloane (1977), the generator matrix G of $\mathscr{B}_p$ is

$$G = \begin{array}{c} \begin{array}{cccccccc} l_\infty & l_0 & \dots & l_{p-1} & r_\infty & r_0 & \dots & r_{p-1} \end{array} \\ \left[\begin{array}{c|ccc|c|ccc} 1 & & & & 0 & & & \\ \vdots & & I_p & & \vdots & & B & \\ 1 & & & & 0 & & & \\ \hline 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \end{array}\right] \end{array} \tag{5.15}$$

which is, if the last row of G is rearranged as the first row, the columns indexed by l_∞ and r_∞ are rearranged as the last and the first columns respectively, equivalent to (5.5b) with $\alpha = 0$ and $k = p + 1$. Let $j(x) = 1 + x + x^2 + \ldots + x^{p-1}$, the following are some properties of $\mathscr{B}_p$ (Karlin; 1969):

1. for $p \equiv 3 \pmod 8$, $b(x)^3 = (1+q(x))^2(1+q(x)) = (1+n(x))(1+q(x)) = 1+j(x)$, since $q(x)^2 = n(x)$ ($2 \in N$ for this prime) and $q(x)j(x) = n(x)j(x) = j(x)$ ($\mathrm{wt}_H(q(x)) = \mathrm{wt}_H(n(x))$ is odd). Also, $(b(x)+j(x))^3 = (1+q(x)+j(x))^2(1+q(x)+j(x)) = n(x)^2(1+q(x)+j(x)) = q(x)+n(x)+j(x) = 1$ because $n(x)^2 = q(x)$. Since $-1 \in N$ and $b(x)^T = 1 + \sum_{i \in Q} x^{-i} = 1 + n(x)$, it follows that $b(x)b(x)^T = (1 + q(x))(1 + n(x)) = 1 + j(x)$. There are two generator full-rank matrices with disjoint information sets for $\mathscr{B}_p$. This is because, although $b(x)$ has no inverse, $b(x) + j(x)$ does and the inverse being $(b(x) + j(x))^2$. Let G_1 has the form of (5.5b) where $R = B$, the other

full-rank generator matrix G_2 can be obtained as follows

$$G_2 = \left[\begin{array}{c|c} 1 & 1 \ \dots \ 1 \\ \hline 0 & \\ \vdots & B^T \\ 0 & \end{array}\right] \cdot G_1 = \left[\begin{array}{c|c|c|c} 0 & 1 \ \dots \ 1 & 1 & 0 \ \dots \ 0 \\ \hline 1 & & 0 & \\ \vdots & B^T & \vdots & I_p \\ 1 & & 0 & \end{array}\right]. \tag{5.16}$$

It is obvious that G_2 is identical to the generator matrix of $\mathscr{B}_p^\perp$ and hence, it is self-dual;

2. for $p \equiv -3 \pmod 8$, $b(x)^3 = n(x)q(x) = (1 + j(x) + q(x))q(x) = 1 + j(x)$ since $q(x)^2 = n(x)$ ($2 \in N$ for this prime) and $q(x)j(x) = n(x)j(x) = 0$ ($\mathrm{wt}_H(q(x)) = \mathrm{wt}_H(n(x))$ is even). Also, $(b(x)+j(x))^3 = (q(x)+j(x))^2(1+n(x)) = q(x)^2+q(x)^2n(x)+j(x)^2+j(x)^2n(x) = n(x)+q(x)+j(x) = 1$ because $n(x)^2 = q(x)$. Since $-1 \in Q$ and $b(x)^T = \sum_{i \in Q} x^{-i} = b(x)$, it follows that $\mathscr{B}_p$ is fsd, i.e. the generator matrix of $\mathscr{B}_p^\perp$ is given by $\mathscr{B}_p$ is given by

$$G^\perp = \left[\begin{array}{c|c|c|c} 0 & 1 \ \dots \ 1 & 1 & 0 \ \dots \ 0 \\ \hline 1 & & 0 & \\ \vdots & B & \vdots & I_p \\ 1 & & 0 & \end{array}\right].$$

Since $(b(x) + j(x))^{-1} = (b(x) + j(x))^2$, there exists full-rank two generator matrices of disjoint information sets for $\mathscr{B}_p$. Let G_1 has the form of (5.5b) where $R = B$, the other full-rank generator matrix G_2 can be obtained as follows

$$G_2 = \left[\begin{array}{c|c} 1 & 1 \ \dots \ 1 \\ \hline 0 & \\ \vdots & B^2 \\ 0 & \end{array}\right] \cdot G_1 = \left[\begin{array}{c|c|c|c} 0 & 1 \ \dots \ 1 & 1 & 0 \ \dots \ 0 \\ \hline 1 & & 0 & \\ \vdots & B^2 & \vdots & I_p \\ 1 & & 0 & \end{array}\right]. \tag{5.17}$$

Codes of the form $\mathscr{B}_p$ form an interesting family of double-circulant codes. In terms of self-dual codes, the family contains the longest extremal Type II code known, $n = 136$. Moreover, $\mathscr{B}_p$ is the binary image of the extended QR code over $\mathbb{F}_4$ (Karlin et al.; 1978).

The $[p + 1, \frac{1}{2}(p + 1), d]$ double-circulant codes for $p \equiv \pm 1 \pmod 8$ is fixed by $\mathrm{PSL}_2(p)$, see Section 5.3.1. This linear group $\mathrm{PSL}_2(p)$ is generated by the set of all permutations to the coordinates $(\infty, 0, 1, \dots, p-1)$ of the form

$$y \to \frac{ay + b}{cy + d} \tag{5.18}$$

where $a, b, c, d \in \mathbb{F}_p$, $ad - bc = 1$, $y \in \mathbb{F}_p \cup \{\infty\}$ and it is assumed that $\pm\frac{1}{0} = \infty$ and $\pm\frac{1}{\infty} = 0$ in the arithmetic operations.

It is known from MacWilliams and Sloane (1977) that this form of permutation is generated by following transformations

$$\begin{aligned} S &: y \to y+1 \\ V &: y \to \alpha^2 y \\ T &: y \to -\frac{1}{y} \end{aligned} \tag{5.19}$$

where α is a primitive element of $\mathbb{F}_p$. In fact, V is redundant since it can be obtained from S and T, i.e.

$$V = TS^{\alpha}TS^{\mu}TS^{\alpha} \tag{5.20}$$

for $\mu = \alpha^{-1} \in \mathbb{F}_p$[‡].

The linear group $\mathrm{PSL}_2(p)$ fixes not only the $[p+1, \frac{1}{2}(p+1), d]$ binary double-circulant codes, for $p \equiv \pm 1 \pmod 8$, but also the $[2(p+1), p+1, d]$ binary quadratic double-circulant codes, as shown as follows. Consider the coordinates $(\infty, 0, 1, \ldots, p-1)$ of a circulant, the transformation S leaves the coordinate ∞ invariant and introduces a cyclic shift to the rest of the coordinates and hence S fixes a circulant. Let R_i and L_i denote the ith row of the right and left circulants of (5.15) respectively (it is assumed that the index starts with 0), J and J' denote the last row of the right and left circulant of (5.15) respectively.

Consider the primes $p = 8m+3$, $R_0 = \left(0 \mid 1 + \sum_{i \in Q} x^i\right)$. Let e_i and f_j, for some integers i and j, be even and odd integers respectively. If $i \in Q$, $-1/i = -1 \cdot \alpha^{p-1}/\alpha^{e_1} = \alpha^{f_1} \cdot \alpha^{e_2 - e_1} \in N$ since $-1 \in N$ for these primes. Therefore, the transformation T interchanges residues to non residues and vice versa. In addition, it is also known that T interchanges coordinates ∞ and 0. Applying transformation T to R_0, $T(R_0)$, results in

$$T(R_0) = \left(1 \mid \sum_{j \in N} x^j\right) = R_0 + J.$$

Similarly, for the first row of L, which has 1 at coordinates ∞ and 0 only, i.e. $L_0 = (1 \mid 1)$

$$T(L_0) = L_0 + J'.$$

Let $s \in Q$ and let the set $\hat{Q} = Q \cup \{0\}$, $R_s = \left(0 \mid \sum_{i \in \hat{Q}} x^{s+i}\right)$ and $T\left(\sum_{i \in \hat{Q}} x^{s+i}\right) = \sum_{i \in \hat{Q}} x^{-1/(s+i)}$. Following MacWilliams and Sloane (1977, Theorem 24, Ch. 16), it is known that the exponents of $\sum_{i \in \hat{Q}} x^{s+i}$ contain $2m+1$ residues and $2m+1$ non residues. Note that $s+i$ produces no 0 since both $i, s \in Q$[§]. It follows that $-1/(s+i)$ contains $2m+1$ non residues and $2m+1$ residues. Now consider $R_{-1/s} = \left(0 \mid \sum_{i \in \hat{Q}} x^{i-1/s}\right)$, $i - 1/s$ contains 0[¶], $2m$ residues and $2m+1$ non residues. It is possible

[‡] $TS^{\alpha}TS^{\mu}TS^{\alpha}(y) = TS^{\alpha}TS^{\mu}T(y+\alpha) = TS^{\alpha}TS^{\mu}(-y^{-1}+\alpha) = TS^{\alpha}T\left(-\frac{1}{y+\mu}+\alpha\right) = TS^{\alpha}T\left(\frac{\alpha y+\alpha\mu-1}{y+\mu}\right) = TS^{\alpha}\left(\frac{-\alpha y^{-1}+\alpha\mu-1}{-y^{-1}+\mu}\right) = T\left(\frac{-\alpha(y+\alpha)^{-1}+\alpha\mu-1}{-(y+\alpha)^{-1}+\mu}\right) = T\left(\frac{(\alpha\mu-1)y+\alpha(\alpha\mu-1)-\alpha}{\mu y+(\alpha\mu-1)}\right) = \left(\frac{(-\alpha\mu-1)y^{-1}+\alpha(\alpha\mu-1)-\alpha}{-\mu y^{-1}+(\alpha\mu-1)}\right) = \left(\frac{-\alpha}{-\mu y^{-1}}\right) = \alpha^2 y = V(y)$.

[§] Consider a prime $p = \pm 3 \pmod 8$, $q \in Q$ and an integer a where $(a, p) = 1$. In order for $q + a = 0$ to happen, $a = -q$. The integer a is a residue if $p = 8m-3$ and a non residue if $p = 8m+3$.

[¶] This is because all $i \in Q$ are considered and $1/s \in Q$.

to write $-1/(s+i)$ as

$$-\frac{1}{s+i} = \frac{i/s}{s+i} - \frac{1}{s} = z - \frac{1}{s}.$$

Let $I \subset \hat{Q}$ be a set of all residues such that for all $i \in I$, $i - 1/s \in N$. If $-1/(s+i) \in N$, $z \in \hat{Q}$ and it can be seen that z must belong to I such that $z - 1/s \in N$. This means these non residues cancel each other in $T(\boldsymbol{R}_s) + \boldsymbol{R}_{-1/s}$. On the other hand, if $-1/(s+i) \in Q$, $z \in N$ and it is obvious that $z - 1/s \neq i - 1/s$ for all $i \in \hat{Q}$, implying that all $2m+1$ residues in $T(\boldsymbol{R}_s)$ are disjoint from all $2m+1$ residues (including 0) in $\boldsymbol{R}_{-1/s}$. Therefore, $T(\boldsymbol{R}_s) + \boldsymbol{R}_{-1/s} = \left(0 \mid \sum_{i\in\hat{Q}} x^i\right)$, i.e.

$$T(\boldsymbol{R}_s) = \boldsymbol{R}_{-1/s} + \boldsymbol{R}_0.$$

Similarly, $T(\boldsymbol{L}_s) = \left(0 \mid 1 + x^{-1/s}\right)$ and $\boldsymbol{L}_{-1/s} = \left(1 \mid x^{-1/s}\right)$, which means

$$T(\boldsymbol{L}_s) = \boldsymbol{L}_{-1/s} + \boldsymbol{L}_0.$$

Let $t \in N$, $\boldsymbol{R}_t = \left(0 \mid \sum_{i\in\hat{Q}} x^{t+i}\right)$ and $T\left(\sum_{i\in\hat{Q}} x^{t+i}\right) = \sum_{i\in\hat{Q}} x^{-1/(t+i)}$. It is known that $t+i$ contains 0, $2m$ residues and $2m+1$ non residues (MacWilliams and Sloane; 1977, Theorem 24, Ch. 16), and correspondingly $-1/(t+i)$ contains ∞, $2m$ non residues and $2m+1$ residues. As before, now consider $\boldsymbol{R}_{-1/t} = \left(0 \mid \sum_{i\in\hat{Q}} x^{i-1/t}\right)$. There are $2m+1$ residues (excluding 0) and $2m+1$ non residues in $i-1/t$, and let $I' \subset \hat{Q}$ be a set of all residues such that, for all $i \in I'$, $i - 1/t \in Q$. As before, it is possible to write $-1/(t+i)$ as $z - 1/t$, where $z = (i/t)/(t+i)$. If $-1/(t+i) \in Q$, $z \in I'$ and hence the $2m+1$ residues from $-1/(t+i)$ are identical to those from $i - 1/t$. If $-1/(t+i) \in N$, $z \in N$ and hence all of the $2m$ non residues of $-1/(t+i)$ are disjoint from all $2m+1$ non residues of $i - 1/t$. Therefore, $T(\boldsymbol{R}_t) + \boldsymbol{R}_{-1/t} = \left(1 \mid \sum_{i\in N} x^i\right)$, i.e.

$$T(\boldsymbol{R}_t) = \boldsymbol{R}_{-1/t} + \boldsymbol{R}_0 + \boldsymbol{J}.$$

Similarly, $T(\boldsymbol{L}_t) = \left(0 \mid 1 + x^{-1/t}\right)$ and $\boldsymbol{L}_{-1/t} = \left(1 \mid x^{-1/t}\right)$, which means

$$T(\boldsymbol{L}_t) = \boldsymbol{L}_{-1/t} + \boldsymbol{L}_0 + \boldsymbol{J}'.$$

For primes $p = 8m-3$, $\boldsymbol{R}_0 = \left(0 \mid \sum_{i\in Q} x^i\right)$ and since $-1 \in Q$, $-1/i \in Q$ for $i \in Q$. Thus,

$$T(\boldsymbol{R}_0) = \left(0 \mid \sum_{i\in Q} x^{-1/i}\right) = \boldsymbol{R}_0.$$

Similarly for $\boldsymbol{L}_0$, which contains 1 at coordinates 0 and ∞,

$$T(\boldsymbol{L}_0) = \boldsymbol{L}_0.$$

Consider $\boldsymbol{R}_s = \left(0 \mid \sum_{i\in Q} x^{s+i}\right)$, for $s \in Q$, $T\left(\sum_{i\in Q} x^{s+i}\right) = \sum_{i\in Q} x^{-1/(s+i)}$. There are 0 (when $i = -s \in Q$), $2m-2$ residues and $2m-1$ non residues in the set $s+i$ (MacWilliams and Sloane;

1977, Theorem 24, Ch. 16). Correspondingly, $-1/(s+i) = z - 1/s$, where $z = (i/s)/(s+i)$, contains ∞, $2m-2$ residues and $2m-1$ non residues. Now consider $R_{-1/s} = \left(0 \mid \sum_{i\in Q} x^{i-1/s}\right)$, the set $i - 1/s$ contains 0 (when $i = 1/s \in Q$), $2m-2$ residues and $2m-1$ non residues. Let $I \subset Q$ be a set of all residues such that for all $i \in I$, $i - 1/s \in Q$. If $-1/(s+i) \in Q$ then $z - 1/s \in Q$ which means $z \in Q$ and z must belong to I. This means all $2m-2$ residues of $-1/(s+i)$ and those of $i - 1/s$ are identical. On the contrary, if $-1/(s+i) \in N$, $z \in N$ and this means $z - 1/s \neq i - 1/s$ for all $i \in Q$, and therefore all non residues in $-1/(s+i)$ and $i - 1/s$ are mutually disjoint. Thus, $T(R_s) + R_{-1/s} = \left(1 \mid 1 + \sum_{i\in N} x^i\right)$, i.e.

$$T(R_s) = R_{-1/s} + R_0 + J.$$

Similarly, $T(L_s) = \left(0 \mid 1 + x^{-1/s}\right)$, and this is equivalent to

$$T(L_s) = L_{-1/s} + L_0 + J'.$$

For $t \in N$, $R_t = \left(0 \mid \sum_{i\in Q} x^{t+i}\right)$ and $T(\sum_{i\in Q} x^{t+i}) = \sum_{i\in Q} x^{-1/(t+i)}$. Following MacWilliams and Sloane (1977, Theorem 24, Ch. 16), there are $2m-1$ residues and $2m-1$ non residues in the set $t+i$ and the same distributions are contained in the set $-1/(t+i)$. Considering $R_{-1/t} = \left(0 \mid \sum_{i\in Q} x^{i-1/t}\right)$, there are $2m-1$ residues and $2m-1$ non residues in $i - 1/t$. Rewriting $-1/(t+i) = z - 1/t$, for $z = (i/t)/(t+i)$, and letting $I' \subset Q$ be a set of all residues such that for all $i \in I'$, $i - 1/t \in N$, it is known that if $-1/(t+i) \in N$ then $z - 1/t \in N$ which means that $z \in Q$ and z must belong to I'. Hence, the non residues in $i - 1/t$ and $-1/(t+i)$ are identical. If $-1/(t+i) \in Q$, however, $z \in N$ and for all $i \in Q$, $i - 1/t \neq z - 1/t$, implying that the residues in $-1/(t+i)$ and $i - 1/t$ are mutually disjoint. Thus, $T(R_t) + R_{-1/t} = \left(0 \mid \sum_{i\in Q} x^i\right)$, i.e.

$$T(R_t) = R_{-1/t} + R_0.$$

Similarly, $T(L_t) = \left(0 \mid 1 + x^{-1/t}\right)$, and equivalently

$$T(L_t) = L_{-1/t} + L_0.$$

The effect T to the circulants are summarised as follows

T	for $p \equiv 3 \pmod 8$	for $p \equiv -3 \pmod 8$
$T(R_0)$	$R_0 + J$	R_0
$T(R_s)$	$R_{-1/s} + R_0$	$R_{-1/s} + R_0 + J$
$T(R_t)$	$R_{-1/t} + R_0 + J$	$R_{-1/t} + R_0$
$T(L_0)$	$L_0 + J'$	L_0
$T(L_s)$	$L_{-1/s} + L_0$	$L_{-1/s} + L_0 + J'$
$T(L_t)$	$L_{-1/t} + L_0 + J'$	$L_{-1/t} + L_0$

where $s \in Q$ and $t \in N$. This shows that, for $p \equiv \pm 3 \pmod 8$, the transformation T is a linear combination of at most three rows of the circulant and hence it fixes the circulant. This establishes the following theorem on $\mathrm{Aut}(\mathscr{B}_p)$ (MacWilliams and Sloane; 1977; Gaborit; 2002).

5.1 *Theorem.* The automorphism group of the $[2(p+1), p+1, d]$ binary quadratic double-circulant codes contains $\mathrm{PSL}_2(p)$ applied simultaneously to both circulants.

The knowledge of $\mathrm{Aut}(\mathscr{B}_p)$ can be exploited to deduce the modular congruence weight distributions of $\mathscr{B}_p$ as shown in Section 5.5.

5.4 Evaluation of the Number of Codewords of Given Weight and the Minimum Distance: A More Efficient Approach

In Chapter 4, algorithms to compute the minimum distance of a binary linear code, which can be easily modified to count the number of codewords of a given weight, are described. If the code of interest is not cyclic, the Zimmermann's (1996) algorithm is used. Assuming the code rate of the code is a half and its generator matrix contains two mutually disjoint information sets, each of rank k (the code dimension), Zimmermann's (1996) algorithm requires enumeration of

$$\binom{k}{w/2} + 2\sum_{i=1}^{w/2-1}\binom{k}{i}$$

codewords in order to count the number of codewords of weight w. Here, it is shown that a more efficient approach exists for fsd double-circulant codes for $p \equiv -3 \pmod 8$ and self-dual double-circulant codes. This approach applies to both pure and bordered double-circulant cases.

5.7 *Lemma.* Let $T_m(x)$ be a set of binary polynomials with degree at most m. Let $u_i(x), v_i(x) \in T_{k-1}(x)$ for $i = 1, 2$, and $e(x), f(x) \in T_{k-2}(x)$. The number of weight w codewords of the forms $c_1(x) = (u_1(x)|v_1(x))$ and $c_2(x) = (v_2(x)|u_2(x))$ are equal, where

i) for self-dual pure double-circulant codes, $u_2(x) = u_1(x)^T$ and $v_2(x) = v_1(x)^T$;

ii) for self-dual bordered double-circulant codes, $u_1(x) = (\epsilon|e(x))$, $v_1(x) = (\gamma|f(x))$, $u_2(x) = (\epsilon|e(x)^T)$ and $v_2(x) = (\gamma|f(x)^T)$ where $\gamma = \mathrm{wt}_H(e(x)) \pmod 2$;

iii) for fsd pure double-circulant codes ($p \equiv -3 \pmod 8$), $u_2(x) = u_1(x)^2$ and $v_2(x) = v_1(x)^2$;

iv) for fsd bordered double-circulant codes ($p \equiv -3 \pmod 8$), $u_1(x) = (\epsilon|e(x))$, $v_1(x) = (\gamma|f(x))$, $u_2(x) = (\epsilon|e(x)^2)$, $v_2(x) = (\gamma|f(x)^2)$ where $\gamma = \mathrm{wt}_H(e(x)) \pmod 2$.

Proof.

i) Let $G_1 = [I_k|R]$ and $G_2 = [R^T|I_k]$ be the two full-rank generator matrices with mutually disjoint information sets of a self-dual pure double-circulant code. Assume that $r(x)$ and $r(x)^T$ are the defining polynomials of G_1 and G_2 respectively. Given $u_1(x)$ as an input, G_1 produces a codeword $c_1(x) = (u_1(x)|v_1(x))$, where $v_1(x) = u_1(x)r(x)$. Another codeword $c_2(x)$ can be obtained from G_2 by using $u_1(x)^T$ as an input, $c_2(x) = (v_1(x)^T|u_1(x)^T)$, where $v_1(x)^T = u_1(x)^T r(x)^T = (u_1(x)r(x))^T$. Since the weight of a polynomial and that of its trans-

pose are equal, for a given polynomial of degree at most $k-1$, there exists two distinct codewords of the same weight.

ii) Let G_1, given by (5.5b), and G_2 be two full-rank generator matrices with pairwise disjoint information sets, of bordered self-dual double-circulant codes. It is assumed that the form of G_2 is identical to that given by (5.16) with $R^T = B^T$. Let $f(x) = e(x)r(x)$, consider the following cases:

(a) $\epsilon = 0$ and $\mathrm{wt}_H(e(x))$ is odd, G_1 produces a codeword

$$c_1(x) = (0 \mid e(x) \mid 1 \mid f(x)).$$

Applying $(0 \mid e(x)^T)$ as an information vector to G_2 yields another codeword

$$c_2(x) = (1 \mid e(x)^T r(x)^T \mid 0 \mid e(x)^T) = (1 \mid f(x)^T \mid 0 \mid e(x)^T).$$

(b) $\epsilon = 1$ and $\mathrm{wt}_H(e(x))$ is odd, G_1 produces

$$c_1(x) = (1 \mid e(x) \mid 1 \mid f(x) + j(x)).$$

Applying $(1 \mid e(x)^T)$ as an information vector to G_2, a codeword $c_2(x)$, which has the form

$$c_2(x) = (1 \mid e(x)^T r(x)^T + j(x) \mid 1 \mid e(x)^T) = (1 \mid f(x)^T + j(x) \mid 1 \mid e(x)^T),$$

is obtained.

(c) $\epsilon = 0$ and $\mathrm{wt}_H(e(x))$ is even, G_1 produces a codeword

$$c_1(x) = (0 \mid e(x) \mid 0 \mid f(x)).$$

Applying $(0 \mid e(x)^T)$ as an information vector to G_2, another codeword

$$c_2(x) = (0 \mid e(x)^T r(x)^T \mid 0 \mid e(x)^T) = (0 \mid f(x)^T \mid 0 \mid e(x)^T)$$

is produced.

(d) $\epsilon = 1$ and $\mathrm{wt}_H(e(x))$ is even, G_1 produces

$$c_1(x) = (1 \mid e(x) \mid 0 \mid f(x) + j(x)).$$

Applying $(1 \mid e(x)^T)$ as an information vector to G_2 yields a codeword $c_2(x)$ which has the form

$$c_2(x) = (0 \mid e(x)^T r(x)^T + j(x) \mid 1 \mid e(x)^T) = (0 \mid f(x)^T + j(x) \mid 1 \mid e(x)^T).$$

It is clear that in all cases, $\mathrm{wt}_H(c_1(x)) = \mathrm{wt}_H(c_2(x))$ since $\mathrm{wt}_H(v(x)) = \mathrm{wt}_H(v(x)^T)$ and $\mathrm{wt}_H(v(x) + j(x)) = \mathrm{wt}_H(v(x)^T + j(x))$ for some polynomial $v(x)$. This means that given an information vector, there always exists two distinct codewords of the same weight.

iii) Let G_1, given by (5.5a) with $R = I_p + Q$, and G_2, given by (5.13), be two full-rank generator matrices with pairwise disjoint information sets, of pure fsd double-circulant codes for $p \equiv -3 \pmod 8$. Given $u_1(x)$ as input, G_1 produces a codeword $c_1(x) = (u_1(x)|v_1(x))$, where $v_1(x) = u_1(x)(1 + q(x))$, where as G_2 produces a codeword $c_2(x) = (v_2(x)|u_2(x))$, where $u_2(x) = u_1(x)^2$ and $v_2(x) = u_1(x)^2(1+n(x)) = u_1(x)^2(1+q(x))^2 = v_1(x)^2$. Since the weight of a polynomial and that of its square are the same over $\mathbb{F}_2$, the proof follows.

iv) Let G_1, given by (5.5b) with $B = R$, and G_2, given by (5.17), be two full-rank generator matrices with pairwise disjoint information sets, of bordered fsd double-circulant codes for $p \equiv -3 \pmod 8$. Let $f(x) = e(x)b(x)$, consider the following cases:

(a) $\epsilon = 0$ and $\mathrm{wt}_H(e(x))$ is odd, G_1 produces a codeword

$$c_1(x) = (0 \mid e(x) \mid 1 \mid f(x)).$$

Applying $(0 \mid e(x)^2)$ as an information vector to G_2, another codeword

$$c_2(x) = (1 \mid e(x)^2 n(x) \mid 0 \mid e(x)^2) = (1 \mid f(x)^2 \mid 0 \mid e(x)^2)$$

since $e(x)^2 n(x) = e(x)^2 b(x)^2 = f(x)^2$, is obtained.

(b) $\epsilon = 1$ and $\mathrm{wt}_H(e(x))$ is odd, G_1 produces

$$c_1(x) = (1 \mid e(x) \mid 1 \mid f(x) + j(x)).$$

Applying $(1 \mid e(x)^2)$ as an information vector to G_2 yields a codeword $c_2(x)$ which can be written as

$$c_2(x) = (1 \mid e(x)^2 n(x) + j(x) \mid 1 \mid e(x)^2) = (1 \mid f(x)^2 + j(x) \mid 1 \mid e(x)^2).$$

(c) $\epsilon = 0$ and $\mathrm{wt}_H(e(x))$ is even, G_1 produces a codeword

$$c_1(x) = (0 \mid e(x) \mid 0 \mid f(x)).$$

Applying $(0 \mid e(x)^2)$ as an information vector to G_2, another codeword

$$c_2(x) = (0 \mid e(x)^2 n(x) \mid 0 \mid e(x)^2) = (0 \mid f(x)^2 \mid 0 \mid e(x)^2)$$

is produced.

(d) $\epsilon = 1$ and $\mathrm{wt}_H(e(x))$ is even, G_1 produces

$$c_1(x) = (1 \mid e(x) \mid 0 \mid f(x) + j(x)).$$

Applying $(1 \mid e(x)^2)$ as an information vector to G_2 yields a codeword $c_2(x)$ which can be written as

$$c_2(x) = (0 \mid e(x)^2 n(x) + j(x) \mid 1 \mid e(x)^2) = (0 \mid f(x)^2 + j(x) \mid 1 \mid e(x)^2).$$

It is clear that in all cases, $\mathrm{wt}_H(c_1(x)) = \mathrm{wt}_H(c_2(x))$ since $\mathrm{wt}_H(v(x)) = \mathrm{wt}_H(v(x)^2)$ and $\mathrm{wt}_H(v(x)+j(x)) = \mathrm{wt}_H(v(x)^2+j(x))$ for some polynomial $v(x)$. This means that given an information vector, there always exists two distinct codewords of the same weight. □

From Lemma 5.7, it follows that, in order to count the number of codewords of weight w, only

$$\sum_{i=1}^{w/2} \binom{k}{i}$$

codewords are required to be enumerated and if A_w denotes the number of codewords of weight w,

$$A_w = a_{w/2} + 2 \sum_{i=1}^{w/2-1} a_i, \tag{5.21}$$

where a_i is the number of weight w codewords which have i non zeros in the first k coordinates.

Similarly, the commonly used method to compute the minimum distance of half-rate codes with two full-rank generator matrices of mutually disjoint information sets (e.g. van Dijk et al.; 2005), assuming that d is the minimum distance of the code, as many as

$$S = 2 \sum_{i=1}^{d/2-1} \binom{n}{i}$$

codewords are required to be enumerated. Following Lemma 5.7, only $S/2$ codewords are required for $\mathscr{P}_p$ and $\mathscr{B}_p$ for $p \equiv -3 \pmod 8$, and self-dual double-circulant codes. Note that the bound $d/2-1$ may be improved for singly- and doubly-even codes, but a consideration for the general cases is given here.

5.5 Weight Distributions

The automorphism group of both $[p+1, \frac{1}{2}(p+1), d]$ extended QR and $[2(p+1), p+1, d]$ quadratic double-circulant codes contains the projective special linear group, $\mathrm{PSL}_2(p)$. Let $\mathcal{H}$ be a subgroup of the automorphism group of a linear code, the number of codewords of weight i, denoted by A_i, can be categorised into two classes:

1. a class of weight i codewords which are invariant under some element of $\mathcal{H}$; and

2. a class of weight i codewords which forms an orbit of size $|\mathcal{H}|$, the order of $\mathcal{H}$. In the other words, if c is a codeword of this class, applying all elements of $\mathcal{H}$ to c, $|\mathcal{H}|$ distinct codewords are obtained.

Thus, A_i may be written in terms of congruence as follows

$$\begin{aligned} A_i &= n_i \cdot |\mathcal{H}| + A_i(\mathcal{H}), \\ &\equiv A_i(\mathcal{H}) \pmod{|\mathcal{H}|} \end{aligned} \tag{5.22}$$

where $A_i(\mathcal{H})$ is the number of codewords of weight i fixed by some element of $\mathcal{H}$. This was originally shown in Mykkeltveit et al. (1972), where it was applied to extended QR codes for primes 97 and 103.

5.5.1 The Number of Codewords of a Given Weight in Quadratic Double-Circulant Codes

For $\mathscr{B}_p$, we shall choose $\mathcal{H} = \mathrm{PSL}_2(p)$, which has order $|\mathcal{H}| = \frac{1}{2}p(p^2-1)$. Let the matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ represent an element of $\mathrm{PSL}_2(p)$, see (5.18). Since $|\mathcal{H}|$ can be factorised as $|\mathcal{H}| = \prod_j q_j^{e_j}$, where q_j is a prime and e_j is some integer, $A_i(\mathcal{H}) \pmod{|\mathcal{H}|}$ can be obtained by applying the Chinese-Remainder-Theorem to $A_i(S_{q_j}) \pmod{q_j^{e_j}}$ for all q_j that divides $|\mathcal{H}|$, where S_{q_j} is the Sylow-q_j-subgroup of $\mathcal{H}$. In order to compute $A_i(S_{q_j})$, a subcode of $\mathscr{B}_p$ which is invariant under S_{q_j} needs to be obtained in the first place. This invariant subcode, in general, has a considerably smaller dimension than $\mathscr{B}_p$, and hence its weight distribution can be easily obtained.

For each odd prime q_j, S_{q_j} is a cyclic group which can be generated by some $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in \mathrm{PSL}_2(p)$ of order q_j. Because S_{q_j} is cyclic, it is straightforward to obtain the invariant subcode, from which $A_i(S_{q_j})$ can be computed.

On the other hand, the case of $q_j = 2$ is more complicated. For $q_j = 2$, S_2 is a dihedral group of order 2^{m+1}, where $m+1$ is the maximum power of 2 that divides $|\mathcal{H}|$ (Burnside; 1955). For $p = 8m \pm 3$, it is known that

$$|\mathcal{H}| = \frac{1}{2}(8m \pm 3)\left((8m \pm 3)^2 - 1\right) = 2^2\left(64m^3 \pm 72m^2 + 26m \pm 3\right),$$

which shows that the highest power of 2 that divides $|\mathcal{H}|$ is 2 ($m = 1$). Following Burnside (1955), there are $2^m + 1$ subgroups of order 2 in S_2, namely

$$\begin{aligned} H_2 &= \{1, P\}, \\ G_2^0 &= \{1, T\}, \text{ and} \\ G_2^1 &= \{1, PT\}, \end{aligned}$$

where $P, T \in \mathrm{PSL}_2(p)$, $P^2 = T^2 = 1$ and $TPT^{-1} = P^{-1}$. Let $T = \left[\begin{smallmatrix} 0 & p-1 \\ 1 & 0 \end{smallmatrix}\right]$, which has order 2. It can be shown that any order 2 permutation, $P = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, if a constraint $b = c$ is imposed, $a = -d$ is obtained. All these subgroups, however, are conjugates in $\mathrm{PSL}_2(p)$ (Burnside; 1955) and therefore, the subcodes fixed by G_2^0, G_2^1 and H_2 have identical weight distributions and considering any of them, say G_2^0, is sufficient.

Apart from $2^m + 1$ subgroups of order 2, S_2 also contains a cyclic subgroup of order 4, 2^{m-1} non cyclic subgroups of order 4, and subgroups of order 2^j for $j \geq 3$. Following Mykkeltveit et al. (1972), only the subgroups of order 2 and the non cyclic subgroups of order 4 that make contributions towards $A_i(S_2)$. For $p \equiv \pm 3 \pmod 8$, there is only one non cyclic subgroup of order 4, denoted by G_4, which contains, apart from an identity, three permutations of order 2 (Burnside; 1955), i.e. a Klein 4 group,

$$G_4 = \{1, P, T, PT\}.$$

Having obtained $A_i(G_2^0)$ and $A_i(G_4)$, following the argument in Mykkeltveit et al. (1972), the number of codewords of weight i that are fixed by some element of S_2 is given by

$$A_i(S_2) \equiv 3A_i(G_2^0) - 2A_i(G_4) \pmod 4. \tag{5.23}$$

In summary, in order to deduce the modular congruence of the number of weight i codewords in $\mathscr{B}_p$, it is sufficient to do the following steps

1. compute the number of weight i codewords in the subcodes fixed by G_2^0, G_4 and S_q, for all odd primes q that divide $|\mathcal{H}|$;

2. apply (5.23) to $A_i(G_2^0)$ and $A_i(G_4)$ to obtain $A_i(S_2)$; and then

3. apply the Chinese-Remainder-Theorem to $A_i(S_2)$ and all $A_i(S_q)$, for all odd primes q that divide $|\mathcal{H}|$, to obtain $A_i(\mathcal{H}) \pmod{|\mathcal{H}|}$.

Given $\mathscr{B}_p$ and an element of $\mathrm{PSL}_2(p)$, how can the subcode consisting of the codewords fixed by this element be found? Assume that $Z = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in \mathrm{PSL}_2(p)$ of prime order. Let c_{l_i} (resp. c_{r_i}) and $c_{l_{i'}}$ (resp. $c_{r_{i'}}$) denote the ith coordinate and $\pi_Z(i)$th coordinate (ith coordinate with respect to permutation π_Z), in the left (resp. right) circulant form respectively. The invariant subcode can be obtained by solving a set of linear equations consisting of the parity-check matrix of $\mathscr{B}_p$ (denoted by H), $c_{l_i} + c_{l_{i'}} = 0$ (denoted by $\pi_Z(L)$) and $c_{r_i} + c_{r_{i'}} = 0$ (denoted by $\pi_Z(R)$) for all $i \in \mathbb{F}_p \cup \{\infty\}$, i.e.

$$H_{sub} = \begin{array}{|c|}\hline \\ H \\ \\ \hline \pi_Z(L) \\ \hline \pi_Z(R) \\ \hline \end{array}.$$

The solution to H_{sub} is a matrix of rank $r > (p+1)$, which is the parity-check matrix of the $[2(p+1), 2(p+1)-r, d']$ invariant subcode. For subgroup G_4, which consists of permutations P, T and PT, the following matrix

$$H_{sub} = \begin{array}{|c|}\hline \\ H \\ \\ \hline \pi_P(L) \\ \hline \pi_P(R) \\ \hline \pi_T(L) \\ \hline \pi_T(R) \\ \hline \pi_{PT}(L) \\ \hline \pi_{PT}(R) \\ \hline \end{array}$$

needs to be transformed into a reduced-echelon form to obtain the invariant subcode. Note that the

parity-check matrix of $\mathscr{B}_p$ is assumed to have the following form

$$H = \begin{array}{c|ccc|c|ccc|} & l_\infty & l_0 & \dots & l_{p-1} & r_\infty & r_0 & \dots & r_{p-1} \\ \hline & 0 & & & & 1 & & & \\ & \vdots & & B^T & & \vdots & & I_p & \\ & 0 & & & & 1 & & & \\ \hline & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ \hline \end{array} . \tag{5.24}$$

One useful application of the modular congruence of the number of codewords of weight w is to provide an independent verification to the number of codewords of weight w computed exhaustively. Computing the number of codewords of a given weight in small codes using a single-threaded algorithm is tractable, but for longer codes, it is a common practice to use multiple computers in parallel to render the computation feasible. In order to do so, the codeword enumeration task is distributed among these computers and each computer just needs to evaluate a predetermined chunk of codewords. In the end, the results are added to give the total number of codewords of the desired weight. There are always rooms for bugs/mistakes to be made in this parallel scheme, perhaps the splitting may not be done properly and double-counting and miss-counting are introduced as a result. The importance of this modular congruence will be demonstrated in Section 5.5.2 and before that, consider the following examples which illustrate the application of the modular congruence technique to the weight distribution of the quadratic double-circulant codes of primes 37 and 83.

Modulo congruence weight distribution of $\mathscr{B}_{37}$

Example 5.1: For prime 37, there exists an fsd [76, 38, 12] quadratic double-circulant code, $\mathscr{B}_{37}$. The weight enumerator of an fsd code is given by Gleason's theorem (Rains and Sloane; 1998)

$$A(z) = \sum_{i=0}^{\lfloor \frac{n}{8} \rfloor} K_i (1+z^2)^{\frac{n}{2}-4i} (z^2 - 2z^4 + z^6)^i \tag{5.25}$$

for integers K_i. Hence, in order to compute $A(z)$ of $\mathscr{B}_{37}$, only A_{2i} for $6 \leq i \leq 9$ are required to be computed. Using the technique described in Section 5.4, the number of codewords of desired weights are obtained and then substituted into (5.25). The resulting weight enumerator function is

$$\begin{aligned} A(z) = {} & (1+z^{76}) + 2109 \cdot (z^{12}+z^{64}) + \\ & 86469 \cdot (z^{16}+z^{60}) + 961704 \cdot (z^{18}+z^{58}) + \\ & 7489059 \cdot (z^{20}+z^{56}) + 53574224 \cdot (z^{22}+z^{54}) + \\ & 275509215 \cdot (z^{24}+z^{52}) + 1113906312 \cdot (z^{26}+z^{50}) + \\ & 3626095793 \cdot (z^{28}+z^{48}) + 9404812736 \cdot (z^{30}+z^{46}) + \\ & 19610283420 \cdot (z^{32}+z^{44}) + 33067534032 \cdot (z^{34}+z^{42}) + \\ & 45200010670 \cdot (z^{36}+z^{40}) + 50157375456 \cdot z^{38} . \end{aligned} \tag{5.26}$$

Let $\mathcal{H} = \mathrm{PSL}_2(37)$, it is known that $|\mathcal{H}| = 2^2 \cdot 3^2 \cdot 19 \cdot 37 = 25308$. Consider the odd primes factors q

in the first place. For $q = 3$, $\left[\begin{smallmatrix} 0 & 1 \\ 36 & 1 \end{smallmatrix}\right]$ generates the following permutation of order 3

$$(\infty, 0, 1)(2, 36, 19)(3, 18, 13)(4, 12, 10)(5, 9, 23)(6, 22, 7)(8, 21, 24)$$
$$(11)(14, 17, 30)(15, 29, 33)(16, 32, 31)(20, 35, 25)(26, 34, 28)(27).$$

The corresponding invariant subcode has a generator matrix $G^{(S_3)}$ of dimension 14, which is given by

$$G^{(S_3)} = \left[\begin{array}{l}
10000000000000011101110011111110100100101100101000100111000100010000000000000 \\
01000000000000000010011000101110110001000101111001101110000001000000000100000 \\
00100000000000011101011101010110011010110101001000001111010000000100000000000 \\
00010000000000011100111111111100010000011110101001101011001000001000000000000 \\
00001000000000011101011111101010100101111111111100011111110000000000000000000 \\
00000100000000011111111111111111111111111111111111000000000000000000000000000 \\
00000010000000011101110001111111111101001010101111101110000000000000000000000 \\
00000001000000000000110011101000100011011101100011111000000000001001000000000 \\
00000000100000011011011000101110111000100011101011101000011000111000000000000 \\
00000000010000111001011101110001000111011110100111011111000000000000001000000001 \\
00000000001000011100010001010011101001001011111100111111000000000000010000000010 \\
00000000000100111001011001010111100001000111111111011110000000000000000000001100 \\
00000000000010111101000010010101101011001101010111111000000000000000001010000 \\
00000000000001000011111101111101101010110111111011111111100000000000000010000000
\end{array}\right]$$

and its weight enumerator function is

$$\begin{aligned}
A^{(S_3)}(z) = {} & (1 + z^{76}) + 3 \cdot (z^{12} + z^{64}) + 24 \cdot (z^{16} + z^{60}) + \\
& 54 \cdot (z^{18} + z^{58}) + 150 \cdot (z^{20} + z^{56}) + 176 \cdot (z^{22} + z^{54}) + \\
& 171 \cdot (z^{24} + z^{52}) + 468 \cdot (z^{26} + z^{50}) + 788 \cdot (z^{28} + z^{48}) + \\
& 980 \cdot (z^{30} + z^{46}) + 1386 \cdot (z^{32} + z^{44}) + 1350 \cdot (z^{34} + z^{42}) + \\
& 1573 \cdot (z^{36} + z^{40}) + 2136 \cdot z^{38}.
\end{aligned} \tag{5.27}$$

For $q = 19$, $\left[\begin{smallmatrix} 0 & 1 \\ 36 & 3 \end{smallmatrix}\right]$ generates the following permutation of order 19

$$(\infty, 0, 25, 5, 18, 32, 14, 10, 21, 2, 1, 19, 30, 26, 8, 22, 35, 15, 3)$$
$$(4, 36, 28, 34, 31, 33, 16, 17, 29, 27, 20, 13, 11, 23, 24, 7, 9, 6, 12).$$

The resulting generator matrix of the invariant subcode $G^{(S_{19})}$, which has dimension 2, is

$$G^{(S_{19})} = \left[\begin{array}{l}
1011111111111111111111111111111111111111110000000000000000000000000000000000000 \\
0100000000000000000000000000000000000000001111111111111111111111111111111111111
\end{array}\right]$$

and its weight enumerator function is

$$A^{(S_{19})}(z) = 1 + 2z^{38} + z^{76}. \tag{5.28}$$

For the last odd prime, $q = 37$, a permutation of order 37

$$(\infty, 0, 18, 24, 27, 14, 30, 15, 13, 32, 25, 26, 33, 19, 7, 4, 6, 23, 34,$$
$$1, 12, 29, 31, 28, 16, 2, 9, 10, 3, 22, 20, 5, 21, 8, 11, 17, 35)(36)$$

is generated by $\left[\begin{smallmatrix} 0 & 1 \\ 36 & 35 \end{smallmatrix}\right]$ and it turns out that the corresponding invariant subcode, and hence the weight enumerator function, are identical to those of $q = 19$.

For $q = 2$, subcodes fixed by some element of G_2^0 and G_4 are required. We have $P = \left[\begin{smallmatrix} 3 & 8 \\ 8 & 34 \end{smallmatrix}\right]$ and

$T = \left[\begin{smallmatrix} 0 & 36 \\ 1 & 0 \end{smallmatrix}\right]$, and the resulting order 2 permutations generated by P, T and PT are

$$(\infty,5)(0,22)(1,17)(2,21)(3,29)(4,16)(6,31)(7,18)(8,26)(9,30)(10,25)$$
$$(11,34)(12,14)(13,36)(15)(19,28)(20,24)(23,27)(32)(33,35),$$

$$(\infty,0)(1,36)(2,18)(3,12)(4,9)(5,22)(6)(7,21)(8,23)(10,11)(13,17)$$
$$(14,29)(15,32)(16,30)(19,35)(20,24)(25,34)(26,27)(28,33)(31),$$

and

$$(\infty,22)(0,5)(1,13)(2,7)(3,14)(4,30)(6,31)(8,27)(9,16)(10,34)(11,25)$$
$$(12,29)(15,32)(17,36)(18,21)(19,33)(20)(23,26)(24)(28,35)$$

respectively. It follows that the corresponding generator matrices and weight enumerator functions of the invariant subcodes are

$$G^{(G_2^0)} = \left[\begin{array}{c}
100000000000000000000001010101011000110101010101111010000001011110010000000000000000000 \\
010000000000000000000000101110000110100001100001110000111101010110110100000000100000000000 \\
001000000000000000000010110111101100101111100010000100011111011000000000000000000000 \\
000100000000000000000001100100111101010111100100100110101001000000000000000100000 \\
000010000000000000000010000011011101110111011000000101010101000001000000000000000 \\
000001000000000000000010110010010111010010010011001110000001110100000000000000000000 \\
000000100000000000000011111111111111111111111111111111111111111111000000000000000000000 \\
000000010000000000000001100001111011110010000001001110111100000000000000000100 \\
000000001000000000000001011011001100011111100011001101101000000000000001000000000 \\
000000000100000000000010001001111110101010101111110010001000000000001000000000 \\
000000000010000000000010011001111100001000110111110110001000000000000000000010 \\
000000000001000000000011100000111100001010011111111100101000000000000000000001000 \\
000000000000100000000011000011101010110110101011100001100000000100000000000000 \\
000000000000010000000111100101010100010111111101000101010100000000000100000000 \\
000000000000001000000011001011010110110000001101110101011010000000000000000000 \\
000000000000000100000011100011000101110110010100010011111100100000001000000000000000 \\
000000000000000010000001011001011011010010100010011111100100000001000000000000000 \\
000000000000000001001111011101100000001111001100111100000000000000000000010000000 \\
000000000000000000100111001001101010010101100111101010000000000000000000000001 \\
000000000000000000011001101110110000000111100110011110000000000000000000100000
\end{array}\right],$$

which has dimension 20, with

$$\begin{aligned}
A^{(G_2^0)}(z) = {} & (1+z^{76}) + 21\cdot(z^{12}+z^{64}) + 153\cdot(z^{16}+z^{60}) + \\
& 744\cdot(z^{18}+z^{58}) + 1883\cdot(z^{20}+z^{56}) + 4472\cdot(z^{22}+z^{54}) + \\
& 10119\cdot(z^{24}+z^{52}) + 21000\cdot(z^{26}+z^{50}) + 36885\cdot(z^{28}+z^{48}) + \\
& 58656\cdot(z^{30}+z^{46}) + 85548\cdot(z^{32}+z^{44}) + 108816\cdot(z^{34}+z^{42}) + \\
& 127534\cdot(z^{36}+z^{40}) + 136912\cdot z^{38}
\end{aligned} \tag{5.29}$$

and

$$G^{(G_4)} = \left[\begin{array}{c}
100000000000011011010110111101000110111011111000111000100001000000000000010000000 \\
010000000000000000101001110001100010001110000111010111000001000000000000000000000 \\
001000000000111011010100001010001001110000000100000000000000000000010000000000010100 \\
000100000000111111111111111111111111111111111111111111111111111111000000000000000000000 \\
000010000000000000101001000001100110001000101111101000010000000010100000000000 \\
000001000000110110111100010110011011111101101111100000000100001000000100000000 \\
000000100000011001011010111011000000110111010110100000000100000000000000000000 \\
000000010000000110001100010110011111101111001101000000000000000000010000000000 \\
000000001000000011000011110111001000000100111011111000000000000000000000100000000 \\
000000000100100001100101000010111111000110100000000000000000000110000000000001000 \\
000000000010000010010100010110001100110111011101000000000000000000000000100000011 \\
000000000001110011100011110101001001001001110011110010000000000000010001000000000
\end{array}\right],$$

which has dimension 12, with

$$\begin{aligned} A^{(G_4)}(z) = &\left(1+z^{76}\right)+3\cdot\left(z^{12}+z^{64}\right)+11\cdot\left(z^{16}+z^{60}\right)+\\ &20\cdot\left(z^{18}+z^{58}\right)+51\cdot\left(z^{20}+z^{56}\right)+56\cdot\left(z^{22}+z^{54}\right)+\\ &111\cdot\left(z^{24}+z^{52}\right)+164\cdot\left(z^{26}+z^{50}\right)+187\cdot\left(z^{28}+z^{48}\right)+\\ &224\cdot\left(z^{30}+z^{46}\right)+294\cdot\left(z^{32}+z^{44}\right)+328\cdot\left(z^{34}+z^{42}\right)+\\ &366\cdot\left(z^{36}+z^{40}\right)+464\cdot z^{38}. \end{aligned} \tag{5.30}$$

respectively. Consider the number of codewords of weight 12, from (5.26)–(5.30) it is known that $A_{12}(G_2^0) = 21$ and $A_{12}(G_4) = 3$; applying (5.23),

$$A_{12}(S_2) \equiv 3\cdot 21 - 2\cdot 3 \pmod 4 \equiv 1 \pmod 4$$

and thus, the following set of simultaneous congruences

$$\begin{aligned} A_{12}(S_2) &\equiv 1 \pmod{2^2}\\ A_{12}(S_3) &\equiv 3 \pmod{3^2}\\ A_{12}(S_{19}) &\equiv 0 \pmod{19}\\ A_{12}(S_{37}) &\equiv 0 \pmod{37} \end{aligned}$$

is obtained. Following the Chinese-Remainder-Theorem, a solution to the above congruences, denoted by $A_{12}(\mathcal{H})$, is congruent modulo $\operatorname{lcm}\{2^2, 3^2, 19, 37\}$, where $\operatorname{lcm}\{2^2, 3^2, 19, 37\}$ is the least-common-multiple of the integers 2^2, 3^2, 19 and 37, which is equal to $2^2\cdot 3^2\cdot 19\cdot 37 = 25308$ in this case. Since these integers are pairwise coprime, by the extended Euclid algorithm, it is possible to write

$$\begin{aligned} 1 &= 4\cdot 1582 + \frac{25308}{4}\cdot(-1)\\ 1 &= 9\cdot 625 + \frac{25308}{9}\cdot(-2)\\ 1 &= 19\cdot 631 + \frac{25308}{19}\cdot(-9)\\ 1 &= 37\cdot 37 + \frac{25308}{37}\cdot(-2). \end{aligned}$$

A solution to the congruences above may be given by

$$\begin{aligned} A_{12}(\mathcal{H}) &= 1\cdot\left[(-1)\frac{25308}{4}\right]+3\cdot\left[(-2)\frac{25308}{9}\right]+0\cdot\left[(-9)\frac{25308}{19}\right]+0\cdot\left[(-2)\frac{25308}{37}\right]\\ &= -1\cdot 6327 + -6\cdot 2812\\ &= 2109\\ A_{12} &= 25308 n_{12} + 2109. \end{aligned}$$

Referring to the weight enumerator function, (5.26), it can be immediately seen that $n_{12} = 0$, indicating that A_{12} has been accurately evaluated. Repeating the above procedures for weights larger than 12, Table 5.1 is obtained which shows that the weight distributions of $\mathscr{B}_{37}$ are indeed accurate.

In fact, since the complete weight distributions can be obtained once the first few terms required by Gleason's theorem are known, verification of these few terms is sufficient.

Table 5.1: Modular congruence weight distributions of $\mathscr{B}_{37}$

$i/n-i$	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3^2	$A_i(S_{19})$ mod 19	$A_i(S_{37})$ mod 37	$A_i(\mathcal{H})$ mod 25308	n_i in $A_i = 25308n_i + A_i(\mathcal{H})$
0/76	1	1	1	1	1	0
12/64	1	3	0	0	2109	0
16/60	1	6	0	0	10545	3
18/58	0	0	0	0	0	38
20/56	3	6	0	0	23199	295
22/54	0	5	0	0	22496	2116
24/52	3	0	0	0	6327	10886
26/50	0	0	0	0	0	44014
28/48	1	5	0	0	16169	143278
30/46	0	8	0	0	5624	371614
32/44	0	0	0	0	0	774865
34/42	0	0	0	0	0	1306604
36/40	2	7	0	0	23902	1785996
38	0	3	2	2	7032	1981878

Modulo congruence weight distribution of $\mathscr{B}_{83}$

Example 5.2: Gulliver and Senkevitch (1999) has shown that the $[168, 84, 24]$ doubly-even self-dual quadratic double-circulant code $\mathscr{B}_{83}$ is not extremal since it has minimum distance less than or equal to 28. The weight enumerator of a Type II code of length n is given by Gleason's theorem, which is expressed as (Rains and Sloane; 1998)

$$A(z) = \sum_{i=0}^{\lfloor n/24 \rfloor} K_i(1 + 14z^4 + z^8)^{\frac{n}{8}-3i}\{z^4(1-z^4)^4\}^i \tag{5.31}$$

where K_i are some integers. As shown in (5.31), only the first few terms of A_i are required in order to completely determine the weight distribution of a Type II code. For $\mathscr{B}_{83}$, only the first 8 terms of A_i are required. Using the parallel version of the efficient codeword enumeration method described in Section 5.4, it is determined that all of these 8 terms are 0 apart from $A_0 = 1$, $A_{24} = 571704$ and $A_{28} = 17008194$.

The correctness of the terms A_{24} and A_{28} has to be independently verified. As in the previous example, the modular congruence method can be used for this purpose. For $p = 83$, it is clear that $|\mathcal{H}| = 2^2 \cdot 3 \cdot 7 \cdot 41 \cdot 83 = 285852$. The odd prime cases will be considered in the first place.

For prime $q = 3$, a cyclic group of order 3 S_3 can be generated by $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 1 \end{smallmatrix}\right] \in \mathrm{PSL}_2(83)$ and it is found that the subcode invariant under S_3 has dimension 28 and has 63 and 0 codewords of weights 24 and 28 respectively.

For prime $q = 7$, there exists a matrix $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 10 \end{smallmatrix}\right]$ which generates S_7. The subcode fixed by S_7 has dimension 12 and neither codeword of weight 24 nor 28 is contained in this subcode.

Similarly, for prime $q = 41$, the subcode fixed by S_{41}, which is generated by $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 4 \end{smallmatrix}\right]$ and has dimension 4, contains no codeword of weights 24 and 28.

Finally, for prime $q = 83$, the invariant subcode of dimension 2, contains the all-zeros, the all-ones, $\{\underbrace{0,0,\ldots,0,0}_{84},\underbrace{1,1,\ldots,1,1}_{84}\}$ and $\{\underbrace{1,1,\ldots,1,1}_{84},\underbrace{0,0,\ldots,0,0}_{84}\}$ codewords only. The cyclic group S_{83} is generated by $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 81 \end{smallmatrix}\right]$.

For the case of $q = 2$, we have $P = \left[\begin{smallmatrix} 1 & 9 \\ 9 & 82 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 82 \\ 1 & 0 \end{smallmatrix}\right]$. The subcode fixed by S_2, which has dimension 42, contains 196 and 1050 codewords of weights 24 and 28 respectively. Meanwhile, the subcode fixed by G_4, which has dimension 22, contains 4 and 6 codewords of weights 24 and 28 respectively.

Thus, using (5.23), the number of codewords of weights 24 and 28 fixed by S_2 are

$$A_{24}(S_2) = 3 \cdot 196 - 2 \cdot 4 \equiv 0 \pmod 4, \text{ and}$$
$$A_{28}(S_2) = 3 \cdot 1050 - 2 \cdot 6 \equiv 2 \pmod 4$$

and by applying the Chinese-Remainder-Theorem to all $A_i(S_q)$ for $i = 24, 28$, it follows that

$$A_{24} = n_{24} \cdot 285852 \tag{5.32a}$$

and

$$A_{28} = n_{28} \cdot 285852 + 142926. \tag{5.32b}$$

From (5.32), it is now known that A_{24} and A_{28} are indeed correct, since they have equality for non negative integers n_{24} and n_{28} ($n_{24} = 2$ and $n_{28} = 59$). Using Gleason's theorem, i.e. (5.31), the weight enumerator function of the $[168, 84, 24]$ code $\mathscr{B}_{83}$ is obtained and it is given by

$$\begin{aligned} A(z) =& (z^0 + z^{168}) + 571704 \cdot (z^{24} + z^{144}) + \\ & 17008194 \cdot (z^{28} + z^{140}) + 5507510484 \cdot (z^{32} + z^{136}) + \\ & 1252615755636 \cdot (z^{36} + z^{132}) + 166058829151929 \cdot (z^{40} + z^{128}) + \\ & 13047194638256310 \cdot (z^{44} + z^{124}) + 629048483051034984 \cdot (z^{48} + z^{120}) + \\ & 19087129808556586056 \cdot (z^{52} + z^{116}) + 372099697089030108600 \cdot (z^{56} + z^{112}) + \\ & 4739291490433882602066 \cdot (z^{60} + z^{108}) + 39973673426117369814414 \cdot (z^{64} + z^{104}) + \\ & 225696677517789500207052 \cdot (z^{68} + z^{100}) + 860241109321000217491044 \cdot (z^{72} + z^{96}) + \\ & 2227390682939806465038006 \cdot (z^{76} + z^{92}) + 3935099587279668544910376 \cdot (z^{80} + z^{88}) + \\ & 4755747411704650343205104 \cdot z^{84}. \end{aligned} \tag{5.33}$$

For the complete weight distributions and their congruences of the $[2(p+1), p+1, d]$ quadratic double-circulant codes, for $11 \le p \le 83$, except $p = 37$ as it has been given in Example 5.1, refer to Appendix D.

5.5.2 The Number of Codewords of a Given Weight in Extended Quadratic Residue Codes

The modular congruence approach of Mykkeltveit et al. (1972), which was originally introduced for extended QR codes $\hat{\mathcal{Q}}_p$, has been modified so that it is applicable to the quadratic double-circulant codes. While $\mathscr{B}_p$ contains one non cyclic subgroup of order 4, $\hat{\mathcal{Q}}_p$ contains two distinct non cyclic subgroups of this order, namely G_4^0 and G_4^1. As a consequence, (5.23) becomes (Mykkeltveit et al.; 1972)

$$A_i(S_2) \equiv (2^m+1)A_i(H_2) - 2^{m-1}A_i(G_4^0) - 2^{m-1}A_i(G_4^1) \pmod{2^{m+1}}, \tag{5.34}$$

where $m+1$ is the highest power of 2 that divides $|\mathcal{H}|$. Unlike $\mathscr{B}_p$ where there are two circulants in which each one is fixed by $\mathrm{PSL}_2(p)$, a linear group $\mathrm{PSL}_2(p)$ acts on the entire coordinates of $\hat{\mathcal{Q}}_p$. In order to obtain the invariant subcode, only one set of linear equations is required which contains the parity-check matrix of $\hat{\mathcal{Q}}_p$ arranged in the order $(0,1,\ldots,p-2,p-1)(\infty)$, and $c_i + c_{i'} = 0$ for all $i \in \mathbb{F}_p \cup \{\infty\}$. Note that c_i and $c_{i'}$ are defined in the same manner as in Section 5.5.1.

Before the importance of this modular congruence approach is demonstrated by proving some of the published results on the weight distribution of $\hat{\mathcal{Q}}_{151}$ and the number of codewords of weights 30 and 32 of $\hat{\mathcal{Q}}_{137}$ are incorrect, let us consider the weight distribution of $\hat{\mathcal{Q}}_{167}$.

Modulo congruence weight distribution of $\hat{\mathcal{Q}}_{167}$

Example 5.3: There exists an extended QR code $\hat{\mathcal{Q}}_{167}$ which has identical parameters ($n = 168$, $k = 84$ and $d = 24$) as the code $\mathscr{B}_{83}$. Since $\hat{\mathcal{Q}}_{167}$ can be put into double-circulant form and it is Type-II self-dual, the algorithm in Section 5.4 can be used to compute the number of codewords of weights 24 and 28, denoted by A'_{24} and A'_{28} for convenience, from which Gleason's theorem (5.31) can be applied to derive its weight enumerator function, $A'(z)$. By evaluations, it was found that

$$\begin{aligned} A'_{24} &= 776216 \\ A'_{28} &= 18130188\,. \end{aligned} \tag{5.35}$$

In order to verify the accuracy of A'_{24} and A'_{28}, the modular congruence method originally described in Mykkeltveit et al. (1972) is used. In this case, $\mathrm{Aut}(\hat{\mathcal{Q}}_{167}) \supseteq \mathcal{H} = \mathrm{PSL}_2(167)$ and it is also known that $|\mathrm{PSL}_2(167)| = 2^3 \cdot 3 \cdot 7 \cdot 83 \cdot 167 = 2328648$. Let $P = \left[\begin{smallmatrix} 12 & 32 \\ 32 & 155 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 166 \\ 1 & 0 \end{smallmatrix}\right]$. Let the permutations of orders 3, 7, 83 and 167 be generated by $\left[\begin{smallmatrix} 0 & 1 \\ 166 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 166 & 19 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 166 & 4 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 166 & 165 \end{smallmatrix}\right]$ respectively. The number of codewords of weights 24 and 28 in the various invariant subcodes of dimension k are

	H_2	G_4^0	G_4^1	S_3	S_7	S_{83}	S_{167}
k	42	22	21	28	12	2	1
A_{24}	252	6	4	140	0	0	0
A_{28}	1812	36	0	0	6	0	0

For $\hat{\mathcal{Q}}_{167}$, (5.34) becomes

$$A_i(S_2) \equiv 5 \cdot A_i(H_2) - 2 \cdot A_i(G_4^0) - 2 \cdot A_i(G_4^1) \pmod 8\,. \tag{5.36}$$

It follows that

$$A_{24}(S_2) \equiv 0 \pmod 8$$
$$A_{28}(S_2) \equiv 4 \pmod 8,$$

and thus

$$A'_{24} = n'_{24} \cdot 2328648 + 776216 \tag{5.37a}$$

and

$$A'_{28} = n'_{28} \cdot 2328648 + 1829652 \tag{5.37b}$$

from the Chinese-Remainder-Theorem.

From (5.32a) and (5.37a), it can be seen that $\mathscr{B}_{83}$ and $\hat{\mathcal{Q}}_{167}$ are indeed inequivalent. This is because for integers $n_{24}, n'_{24} \geq 0$, $A_{24} \neq A'_{24}$.

Equation (5.37) establishes that $A'_{24} = 776216$ ($n'_{24} = 0$) and $A'_{28} = 18130188$ ($n'_{28} = 7$). The weight enumerator of $\hat{\mathcal{Q}}_{167}$ is derived from (5.31) and it is given in (5.38). In comparison to (5.33), it may be seen that, $\hat{\mathcal{Q}}_{167}$ is a slightly inferior code than $\mathscr{B}_{83}$ having more codewords of weights 24, 28 and 32.

$$\begin{aligned}
A'(z) =& (z^0 + z^{168}) + 776216 \cdot (z^{24} + z^{144}) + \\
& 18130188 \cdot (z^{28} + z^{140}) + 5550332508 \cdot (z^{32} + z^{136}) + \\
& 1251282702264 \cdot (z^{36} + z^{132}) + 166071600559137 \cdot (z^{40} + z^{128}) + \\
& 13047136918828740 \cdot (z^{44} + z^{124}) + 629048543890724216 \cdot (z^{48} + z^{120}) + \\
& 19087130695796615088 \cdot (z^{52} + z^{116}) + 372099690249351071112 \cdot (z^{56} + z^{112}) + \\
& 4739291519495550245228 \cdot (z^{60} + z^{108}) + 39973673337590380474086 \cdot (z^{64} + z^{104}) + \\
& 225696677727188690570184 \cdot (z^{68} + z^{100}) + 860241108921860741947676 \cdot (z^{72} + z^{96}) + \\
& 2227390683565491780127428 \cdot (z^{76} + z^{92}) + 3935099586463594172460648 \cdot (z^{80} + z^{88}) + \\
& 4755747412595715344169376 \cdot z^{84}.
\end{aligned} \tag{5.38}$$

Modulo congruence weight distribution of $\hat{\mathcal{Q}}_{137}$

Example 5.4: Gaborit et al. (2005) gave A_{2i}, for $22 \leq 2i \leq 32$, of $\hat{\mathcal{Q}}_{137}$ and the consistency of these results will be checked. For $p = 137$, it is clear that $|\mathrm{PSL}_2(137)| = 2^3 \cdot 3 \cdot 17 \cdot 23 \cdot 137 = 1285608$ and $A_{2i}(S_q)$, where $22 \leq 2i \leq 32$, for all primes q dividing $|\mathrm{PSL}_2(137)|$, have to be computed. Let $P = \left[\begin{smallmatrix} 136 & 51 \\ 51 & 1 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 136 \\ 1 & 0 \end{smallmatrix}\right]$. Let $\left[\begin{smallmatrix} 0 & 1 \\ 136 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 136 & 6 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 & 1 \\ 136 & 11 \end{smallmatrix}\right]$ be generators of permutation of orders 3, 17 and 23 respectively. It is not necessary to find a generator of permutation of order 137 as it fixes the all zeros and all ones codewords only. Subcodes that are invariant under G_2^0, G_4^0, G_4^1, S_3, S_{17} and S_{23} are obtained and the number of weight i, for $22 \leq 2i \leq 32$, codewords in these subcodes are then computed. The results are shown as follows, where k denotes the dimension of the corresponding subcode,

	H_2	G_4^0	G_4^1	S_3	S_{17}	S_{23}	S_{137}
k	35	19	18	23	5	3	1
A_{22}	170	6	6	0	0	0	0
A_{24}	612	10	18	46	0	0	0
A_{26}	1666	36	6	0	0	0	0
A_{28}	8194	36	60	0	0	0	0
A_{30}	34816	126	22	943	0	0	0
A_{32}	114563	261	189	0	0	0	0

It follows that

$$A_i(S_2) \equiv 5 \cdot A_i(H_2) - 2 \cdot A_i(G_4^0) - 2 \cdot A_i(G_4^1) \pmod 8,$$

for $\hat{Q}_{137}$, which is identical to that for $\hat{Q}_{167}$ since they both have 3 as the highest power of 2 that divides $|\mathcal{H}|$. Using this formulation, the following congruences

$$\begin{aligned} A_{22}(S_2) &= 2 \pmod 8 \\ A_{24}(S_2) &= 4 \pmod 8 \\ A_{26}(S_2) &= 6 \pmod 8 \\ A_{28}(S_2) &= 2 \pmod 8 \\ A_{30}(S_2) &= 0 \pmod 8 \\ A_{32}(S_2) &= 3 \pmod 8 \end{aligned}$$

are obtained.
Combining all the results using the Chinese-Remainder-Theorem, it follows that

$$\begin{aligned} A_{22} &= n_{22} \cdot 1285608 + 321402 \\ A_{24} &= n_{24} \cdot 1285608 + 1071340 \\ A_{26} &= n_{26} \cdot 1285608 + 964206 \\ A_{28} &= n_{28} \cdot 1285608 + 321402 \\ A_{30} &= n_{30} \cdot 1285608 + 428536 \\ A_{32} &= n_{32} \cdot 1285608 + 1124907 \end{aligned} \tag{5.39}$$

for some non negative integers n_i. Comparing these to the results in Gaborit et al. (2005), it can be immediately seen that $n_{22} = 0$, $n_{24} = 1$, $n_{26} = 16$, $n_{28} = 381$, and both A_{30} and A_{32} were incorrectly reported. By codeword evaluations, it has been established that

$$\begin{aligned} A_{30} &= 6648307504 \\ A_{32} &= 77865259035, \end{aligned}$$

and hence, $n_{30} = 5171$ and $n_{32} = 60566$ in (5.39).

Modulo congruence weight distribution of $\hat{Q}_{151}$

Example 5.5: Gaborit et al. (2005) also gives the weight distribution of $\hat{Q}_{151}$ which has also been incorrectly reported as shown below. For $\hat{Q}_{151}$, $|\mathrm{PSL}_2(151)| = 2^3 \cdot 3 \cdot 5^2 \cdot 19 \cdot 151 = 1721400$ and we have $P = \left[\begin{smallmatrix} 104 & 31 \\ 31 & 47 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 150 \\ 1 & 0 \end{smallmatrix}\right]$. Let $\left[\begin{smallmatrix} 0 & 1 \\ 150 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 150 & 27 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 & 1 \\ 150 & 8 \end{smallmatrix}\right]$ be generators of permutation of orders 3, 5 and 19 respectively. The number of weight i codewords for $i = 20, 24$, in the various fixed subcodes of dimension k are

	H_2	G_4^0	G_4^1	S_3	S_5	S_{19}	S_{151}
k	38	20	19	26	16	4	1
A_{20}	38	2	0	25	15	0	0
A_{24}	266	4	4	100	0	0	0

and $A_i(S_2)$ is again the same as that for primes 167 and 137, see (5.36). Using this equation, $A_{20}(S_2) = A_{24}(S_2) = 2 \pmod 8$ and following the Chinese-Remainder-Theorem, it is found that

$$\begin{aligned} A_{20} &= n_{20} \cdot 1721400 + 28690 \\ A_{24} &= n_{24} \cdot 1721400 + 717250 \end{aligned} \tag{5.40}$$

It follows that A_{20} has been correctly reported in Gaborit et al. (2005), but A_{24} has been incorrectly reported as 717230. Using the method in Section 5.4, it has been established that

$$\begin{aligned} A_{20} &= 28690 \\ A_{24} &= 717250, \end{aligned}$$

and hence $n_{20} = 0$ and $n_{24} = 0$ in (5.40). Since A_{20} and A_{24} are required to derive the complete weight distribution of $\hat{Q}_{151}$ according to Gleason's theorem for Type-II codes (5.31), the weight distribution of $\hat{Q}_{151}$ given in Gaborit et al. (2005) is no longer correct. The corrected weight distribution of this code, given in terms of the weight enumerator function, is

$$\begin{aligned} A(z) = {} & (z^0 + z^{152}) + 28690 \cdot (z^{20} + z^{132}) + \\ & 717250 \cdot (z^{24} + z^{128}) + 164250250 \cdot (z^{28} + z^{124}) + \\ & 39390351505 \cdot (z^{32} + z^{120}) + 5498418962110 \cdot (z^{36} + z^{116}) + \\ & 430930711621830 \cdot (z^{40} + z^{112}) + 19714914846904500 \cdot (z^{44} + z^{108}) + \\ & 542987434093298550 \cdot (z^{48} + z^{104}) + 9222363801696269658 \cdot (z^{52} + z^{100}) + \\ & 98458872937331749615 \cdot (z^{56} + z^{96}) + 670740325520798111830 \cdot (z^{60} + z^{92}) + \\ & 2949674479653615754525 \cdot (z^{64} + z^{88}) + 8446025592483506824150 \cdot (z^{68} + z^{84}) + \\ & 15840564760239238232420 \cdot (z^{72} + z^{80}) + 19527364659006697265368 \cdot z^{76}. \end{aligned} \tag{5.41}$$

5.6 Minimum Distance Evaluation: A Probabilistic Approach

An interesting observation is that the minimum weight codewords of $\hat{\mathcal{Q}}_p$, for $p \equiv \pm 1 \pmod 8$, and $\mathscr{B}_p$, for $p \equiv \pm 3 \pmod 8$ are always contained in one or more of their fixed subcodes. At least, this is true for all known cases ($n \leq 200$) and this is depicted in Table 5.2. It can be seen that the subcode fixed by H_2 appears in all the known cases. In Table 5.2, the column d_U denotes the minimum distance upper-bound of extremal doubly-even self-dual codes of a given length and the last column indicates the various subgroups whose fixed subcodes contain the minimum weight codewords. The highest n for which the minimum distance of extended QR codes is known, is 168 (Grassl; 2000) and in this thesis, it is extended to include $n = 192, 194$, and 200. The minimum distance of these extended QR codes is obtained from the parallel version of the minimum distance algorithm for cyclic codes (QR codes are cyclic) described in Section 4.4. Note that the fact that the code is singly-even ($n = 194$) or doubly-even ($n = 192, 200$) is also taken into account in order to reduce the number of codewords that need to be enumerated, see Sections 4.2.3 and 4.4. This code property is also taken into account for computing the minimum distance of $\mathscr{B}_p$ using the method described in Section 5.4.

Table 5.2: The Minimum Distance of $\hat{\mathcal{Q}}_p$ and $\mathscr{B}_p$ for $12 \leq n \leq 200$

n	p	p mod 8	d	d_U	Subgroups
12	5	−3	4		H_2, G_4
18	17	1	6		H_2, G_4^0, S_3
24	23	−1	8	8	H_2, G_4^0, G_4^1
28	13	−3	6		H_2, G_4, S_3
32	31	−1	8	8	H_2, G_4^0, S_3
40	19	3	8	8	H_2, G_4, S_3
42	41	1	10		H_2, G_4^1, S_5
48	47	−1	12	12	H_2, G_4^1, S_5
60	29	−3	12		H_2, S_3
72	71	−1	12	16	H_2, G_4^1, S_3, S_5
74	73	1	14		H_2, G_4^0, G_4^1, S_3
76	37	−3	12		H_2, G_4, S_3
80	79	−1	16	16	H_2, G_4^0, G_4^1, S_3
88	43	3	16	16	H_2, S_3, S_7
90	89	1	18		H_2, G_4^0, G_4^1, S_3
98	97	1	16		H_2, G_4^0
104	103	−1	20	20	H_2, G_4^0, S_3
108	53	−3	20		H_2, G_4
114†	113	1	16		H_2, G_4^1, S_7
120	59	3	20	24	H_2, G_4, S_5
124	61	−3	20		H_2, G_4, S_3, S_5
128	127	−1	20	24	H_2, S_3

Continued on next page

n	p	p mod 8	d	d_U	Subgroups
136	67	3	24	24	H_2, G_4, S_3, S_{11}
138	137	1	22		H_2, G_4^0, G_4^1
152†	151	−1	20	28	H_2, G_4^0, S_3, S_5
168	167	−1	24	32	H_2, G_4^0, G_4^1, S_3
168	83	3	24	32	H_2, G_4, S_3
192	191	−1	28	36	H_2, G_4^1
194	193	1	28		H_2, G_4^1, S_3
200	199	−1	32	36	H_2, G_4^0, G_4^1, S_3

† Extended duadic code (Leon et al.; 1984) has higher minimum distance

Based on the above observation, a probabilistic approach to minimum distance evaluation is developed. Given $\hat{\mathcal{Q}}_p$ or $\mathscr{B}_p$, the minimum distance of the code is upper-bounded by

$$d \leq \min_{Z=\{G_2^0, G_4^0, G_4^1, S_{q_1}, S_{q_2}, \ldots\}} \{d(Z)\} \tag{5.42}$$

where $d(Z)$ is the minimum distance of the subcode fixed by $Z \in \mathrm{PSL}_2(p)$ and q runs through all odd primes that divide $|\mathrm{PSL}_2(p)|$. Note that for $\mathscr{B}_p$, $G_4^0 = G_4^1$ hence, only one is required. Using (5.42), an upper-bound of the minimum distance of $\hat{\mathcal{Q}}_p$ and $\mathscr{B}_p$ for all codes where $n \leq 450$ is given and this is tabulated in Table 5.3. The various fixed subgroups where the minimum weight codewords are found are given in the last column of this table. As shown in Tables 5.2 and 5.3, there is no extremal extended QR or quadratic double-circulant codes for $136 < n \leq 450$ and the minimum distance (or its upper-bound for $n > 200$) is plotted against the extremal bound in Figure 5.1. From this figure, it is obvious that, as the block length increases, the gap between the extremal bound and the minimum distance widens and it seems that longer block lengths will follow the same trend. Thus, it can be conjectured that $n = 136$ is the longest doubly-even extremal self-dual double-circulant code. It is worth noting that, for extended QR codes, the results obtained using this probabilistic method are the same as those published by Leon (1988).

Table 5.3: The Minimum Distance of $\hat{\mathcal{Q}}_p$ and $\mathscr{B}_p$ for $204 \leq n \leq 450$

n	p	p mod 8	d	d_U	Subgroups
203	101	−3	≤ 24		H_2, G_4, S_5
216	107	3	≤ 24	40	H_2, G_4, S_3
220	109	−3	≤ 30		H_2, S_3
224	223	−1	≤ 32	40	H_2, G_4^0, G_4^1
234†	233	1	≤ 26		H_2, S_{13}
240‡	239	−1	≤ 32	44	H_2, G_4^1
242‡	241	1	≤ 32		H_2, G_4^1, S_3, S_5
258‡	257	1	≤ 34		H_2, G_4^1
264‡	263	−1	≤ 36	48	H_2, G_4^0, S_3
264‡	131	3	≤ 40	48	H_2, G_4
272‡	271	−1	≤ 40	48	H_2, G_4^0, G_4^1, S_3

Continued on next page

n	p	p mod 8	d	d_U	Subgroups
280‡	139	3	≤ 36	48	H_2, S_3
282‡	281	1	≤ 36		H_2, G_4^0, G_4^1, S_3
300‡	149	−3	≤ 36		H_2, G_4
312‡	311	−1	≤ 36	56	H_2, G_4^0, S_3
314‡	313	1	≤ 40		H_2, G_4^1, S_3
316‡	157	−3	≤ 40		H_2, S_3
328‡	163	3	≤ 44	56	H_2, G_4
338‡	337	1	≤ 40		H_2, G_4^1, S_3
348‡	173	−3	≤ 42		H_2, S_3
354‡	353	1	≤ 42		H_2, G_4^1
360‡	359	−1	≤ 40	64	H_2, G_4^0, G_4^1, Z_5
360‡	179	3	≤ 40	64	H_2, G_4, Z_5
364‡	181	−3	≤ 40		H_2, G_4, Z_3
368‡	367	−1	≤ 48	64	$H_2, G_4^0, Z_3,$
384‡	383	−1	≤ 48	68	H_2, G_4^0, Z_3
396‡	197	−3	≤ 44		H_2, Z_{11}
402‡	201	1	≤ 42		H_2, G_4^0, G_4^1, Z_5
410‡	409	1	≤ 48		H_2, G_4^0, Z_3
424‡	211	3	≤ 56	72	H_2, G_4, Z_3, Z_7
432‡	431	−1	≤ 48	76	H_2, G_4^0, G_4^1, Z_3
434‡	433	1	≤ 38		H_2, G_4^0, Z_3
440‡	440	−1	≤ 48	76	H_2, G_4^0, G_4^1, Z_3
450‡	449	1	≤ 56		H_2, G_4^1

† Extended duadic code (Leon et al.; 1984) has higher minimum distance

‡ The minimum distance of the subcode is computed probabilistically

5.7 Summary

- Bordered double-circulant codes based on primes can be classified into two classes: $[p+1, (p+1)/2, d]$ extended QR codes, for primes $\pm 1 \pmod 8$; and $[2(p+1), p+1, d]$ quadratic double-circulant codes, for primes $\pm 3 \pmod 8$. While quadratic double-circulant codes always exist given a prime $p \equiv \pm 3 \pmod 8$, bordered double-circulant code may not exist given a prime $p \equiv \pm 1 \pmod 8$.
- There also exists $[2p, p, d]$ pure double-circulant codes for any prime $p \equiv \pm 3 \pmod 8$.
- Self-dual double-circulant codes exist for primes $p \equiv -1, 3 \pmod 8$ and for other primes, the double-circulant codes are fsd.
- By exploiting the code structure of fsd double-circulant codes for $p \equiv -3 \pmod 8$ and also the self-dual double-circulant codes–both pure and bordered cases, we have shown that, compared to the commonly used method, the number of codewords required to evaluate the minimum

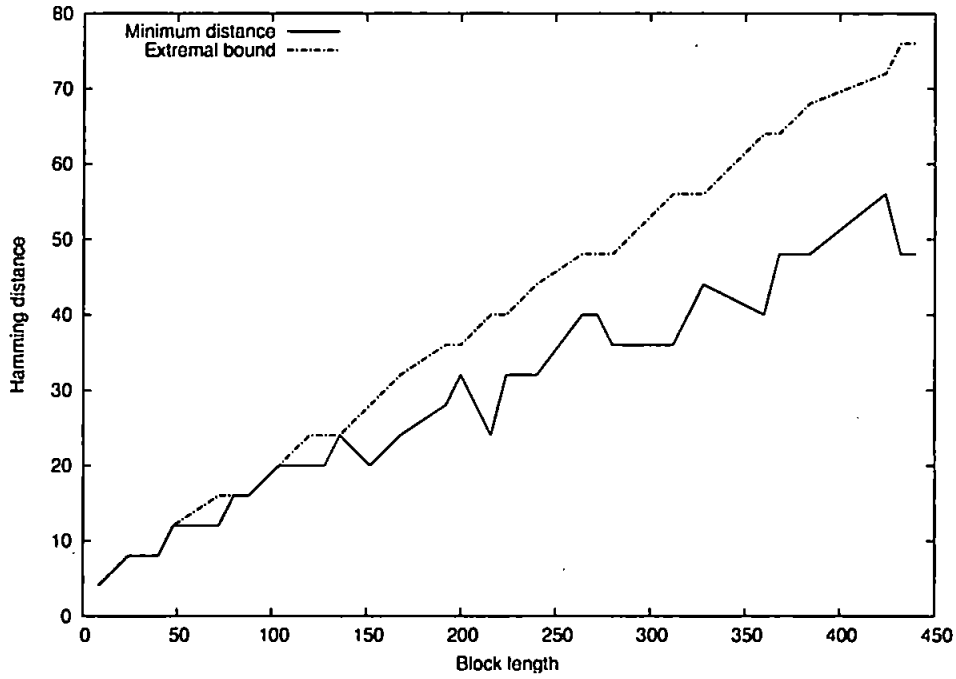


Figure 5.1: Minimum distance and its extremal bound of doubly-even self-dual codes

distance or to count the number of codewords of a given weight can be reduced by a factor around 2.

- The automorphism group of the $[p+1,(p+1)/2,d]$ extended QR code contains the projective special linear group $\mathrm{PSL}_2(p)$ acting on the coordinates $(\infty)(0,1,\ldots,p-2,p-1)$.

- The automorphism group of the $[2(p+1),p+1,d]$ quadratic double-circulant code contains $\mathrm{PSL}_2(p)$, acting on coordinates $(\infty)(0,1,\ldots,p-2,p-1)$, applied simultaneously to left and right circulants.

- The number of codewords of weight i of prime-based double-circulant codes, denoted by A_i, can be written as $A_i = n_i \cdot |\mathrm{PSL}_2(p)| + A_i(\mathrm{PSL}_2(p)) \equiv A_i(\mathrm{PSL}_2(p)) \pmod{|\mathrm{PSL}_2(p)|}$ where $A_i(\mathrm{PSL}_2(p))$ denotes the number of codewords of weight i that are fixed by some element of $\mathrm{PSL}_2(p)$. This result was due to Mykkeltveit et al. (1972) and it was originally introduced for extended QR codes. It has been shown in this chapter that, with some modifications, this modulo congruence method can also be applied to quadratic double-circulant codes.

- The modulo congruence technique is found to be very useful in verifying the number of codewords of a given weight obtained exhaustively by computation. The usefulness of this method has been shown in the case of the extended QR codes for primes 137 and 151 where corrections to some published results on their weight distributions have been provided.

- The weight distribution of the $[168,84,24]$ extended QR code, which was previously unknown, is given in this chapter. There also exists a quadratic double-circulant code with identical parameters (n, k and d) and the weight distribution of this code is also given. The $[168,84,24]$ quadratic double-circulant code appears to be a better code than does the $[168,84,24]$ extended

QR code since it has less low-weight codewords. The correctness of the weight distributions of these two codes are verified by the modulo congruence method.

- The weight enumerator polynomial of an extended QR code of prime p, denoted by $A_{\hat{Q}}(z)$, can be obtained using Gleason's theorem once the first few terms are known. Since $PSL_2(p)$ is doubly-transitive (MacWilliams and Sloane; 1977), knowing $A_{\hat{Q}}(z)$ implies that $A_Q(z)$, the weight enumerator polynomial of the corresponding cyclic QR code, is also known, i.e.

$$A_Q(z) = A_{\hat{Q}}(z) + \frac{1-z}{p+1} A'_{\hat{Q}}(z)$$

  where $A'_{\hat{Q}}(z)$ is the first derivative of $A_{\hat{Q}}(z)$ with the respect to z (van Lint; 1970). As a consequence, the weight distributions of QR codes for primes 151 and 167 are obtained and they are tabulated in Appendix E, Tables E.1 and E.2 respectively.

- A probabilistic method to obtain the minimum distance of double-circulant codes based on primes is presented. This probabilistic approach is based on the observation that the minimum weight codewords are always contained in one or more subcodes fixed by some element of $PSL_2(p)$. Using this approach, it can be conjectured that $n = 136$ is the longest extremal double-circulant self-dual codes.

6 Decoding of Linear Block Codes

Unlike turbo codes or LDPC codes which have a low-complexity soft-decision decoder, constructing a soft-decision decoder for general linear code, which can produce good performance yet have moderate complexity, is a challenge. This chapter describes a flexible suboptimum method to decode general linear block codes. Parts of this chapter appear in the journal paper: Tomlinson, M., Tjhai, C., and Ambroze, M. (2007), "Extending the Dorsch decoder towards achieving maximum likelihood decoding for linear codes", *IET Proceedings Communications*, 1(3), June 2007, pp. 479–488.

6.1 Introduction

It is well-known from the literature that in order to bring the gap to channel capacity closer, a decoder that makes use of channel reliability information, a soft-decision decoder, has to be employed. The hard-decision alternative which as the name implies, quantises each symbol of the received vector to the minimum number of levels with the loss of channel reliability information, is typically around 2 dB inferior. The Viterbi's (1967) algorithm and also the Maximum-A-Posteriori (MAP)* algorithm introduced by Bahl et al. (1974) are optimum soft-decision decoding algorithms for general linear codes. Both of these algorithms require trellis representation of the code considered. For general $[n, k, d]$ linear codes over $\mathbb{F}_q$, there are q^{n-k} states in the trellis and it is obvious that the complexity of these algorithms is excessive for many codes. Trellis independent optimum decoding algorithms exist, (e.g. Hartmann and Rudolph; 1976), but these algorithms also have exponential complexity. As a consequence, many suboptimum soft-decision decoding techniques have been devised.

A type of suboptimum soft-decision decoding algorithm is the reliability-based reprocessing algorithm, which involves reordering the coordinates of the received vector. In general, this type of soft-decision decoding algorithm can be classified into two categories (Fossorier; 2004): most-reliable-positions reprocessing algorithms and least-reliable-positions reprocessing algorithms. In the former case, the most-reliable k information set is determined and some of these k coordinates are then modified. For each modification, a codeword is derived and at the end, the most-likely codeword is chosen. Here the term *most-likely* codeword refers to a codeword which has the smallest distance to the received vector in the Euclidean space. In the latter case, the decoding algorithm uses a different strategy in reprocessing the coordinates. On the basis of the fact that errors are more likely to occur in the least-reliable positions, a typical least-reliable-positions reprocessing algorithm initially obtains a hard-decision vector, denoted by b for convenience, from the received sequence, and then derives some error patterns in these least-reliable positions and subtracts these error patterns from b. Each of the resulting vectors are decoded using an algebraic hard-decision decoder and the most-likely codeword is chosen at the end of the procedure. A notable advantage

*It is also commonly known as the BCJR algorithm, named after the authors.

of the most-reliable-positions reprocessing algorithms is clear, an algebraic decoder, which may not exist for a particular $[n, k, d]$ linear code, is not required.

Various least-reliable-positions reprocessing algorithms have been devised. One of them is the well-known Chase's (1972) algorithm which utilises an algebraic hard decision decoder in conjunction with a search for errors in the least likely positions. The list size of the Chase algorithm is constant and an extension to this algorithm, which generates a dynamic list and which can achieve maximum likelihood decoding if all codewords are processed, has been proposed by Kaneko et al. (1994). Other least-reliable-positions reprocessing approaches include the syndrome-based list decoding algorithms, which are more suitable for high rate codes, proposed by Snyders (1991) and Lous et al. (1993).

Similarly, many most-reliable-positions reprocessing algorithms have been devised. The algorithm of Dorsch (1974) can be considered as the first instance of the most-reliable-positions reprocessing algorithms.

6.2 Dorsch Decoding Algorithm

Dorsch (1974) described a decoding technique that can be applied to any $[n, k, d]$ linear block code using soft-decision quantised to J levels. Assuming an AWGN channel with binary antipodal signalling, the Dorsch decoding algorithm starts by permuting the vector received from the transmission channel, which contains the transmitted codeword–mapped to binary antipodal signal and contaminated by Gaussian noise, in a decreasing reliability order. A hard-decision vector is then derived from this permuted received vector and candidate codewords are derived from a set of k, independent, most likely bits. These codewords are derived using a parity-check matrix whose coordinates are rearranged according to the permutation of the received vector, and which has been reduced to echelon canonical form by elementary row operations. After evaluation of several candidate codewords, the most likely codeword is output.

A soft-decision decoder using a similar principle, but without soft-decision quantisation, has been described by Fossorier and Lin (1995), and it is called the Ordered Statistic Decoder (OSD). An order-i OSD algorithm systematically reprocesses $\sum_{j=0}^{i} \binom{k}{j}$ error patterns in the k most reliable positions of the information set. The complexity of the OSD algorithm depends on the size of the list containing the error patterns and various approaches have been devised to reduce the size of this list (see e.g. Fossorier and Lin; 1996, 1999; Fossorier; 2002; Isaka et al.; 2004).

With the introduction of turbo principles, it was shown in Sweeney and Wesemeyer (2000) that the Dorsch decoder can be modified to produce soft-decision outputs and thus, can be arranged in an iterative scheme to decode product codes. It was also shown in Sweeney and Wesemeyer (2000) that the Dorsch decoder can be easily adapted to decode non binary linear codes.

The power of the Dorsch decoder arises from the relatively unknown property that most codes, on average, can correct almost $n - k$ erasures (Tomlinson et al.; 2007), which is considerably more than the guaranteed number of correctible erasures of $d-1$, or the guaranteed number of correctible hard decision errors of $(d-1)/2$. In this chapter, the Dorsch algorithm for decoding binary codes without quantisation is described. An incremental correlation approach, which features low weight information vectors generated using the efficient revolving door algorithm (Bitner et al.; 1976; Ni-

jenhuis and Wilf; 1978; Knuth; 2005), and a correlation function involving a small number of terms, is presented. This approach is very efficient and allows many codewords to be reprocessed. It is also shown that maximum likelihood decoding is realised provided all codewords are evaluated up to a bounded information weight which may be calculated for each received vector.

6.3 Incremental Correlation Approach to Dorsch Decoding

Throughout this chapter, binary transmission with antipodal signalling over AWGN channel and unquantised received vectors are assumed. Let C denote an $[n, k, d]$ code over $\mathbb{F}_2$, which has codeword

$$c = (c_0, c_1, c_2, \ldots, c_{n-2}, c_{n-1}).$$

In binary antipodal transmission, it is assumed that each symbol c_i of c is mapped to $v_i = \psi(c_i) \in \mathbb{R}$, where the mapping function $\psi(c_i) = 2c_i - 1$. Let the transmitted vector be denoted by $v = (v_0, v_1, \ldots, v_{n-1})$. The received vector $y = (y_0, y_1, \ldots, y_{n-1})$ consists of the transmitted codeword contaminated by independent Gaussian noise, denoted by n, with variance σ^2, i.e. $y_i = v_i + n_i$ for $0 \leq i \leq n-1$. The received vector is also assumed to have been matched-filtered and free from distortion so that

$$\frac{1}{\sigma^2} = 2\frac{E_b}{N_0},$$

where E_b is the energy per information bit and N_0 is the single-sided noise power spectral density. Accordingly, $\sigma^2 = N_0/2E_b$.

Consider the ith position of the received vector y, its a priori log-likelihood ratio is given by, assuming that $\hat{c}_i$ is the ith symbol of the decoder's output,

$$\begin{aligned} L_i &= \log_e\left(\frac{Pr(y_i|\hat{c}_i = 0)}{Pr(y_i|\hat{c}_i = 1)}\right) \\ &= \log_e\left(\frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-(y_i+1)^2/2\sigma^2\right)}{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-(y_i-1)^2/2\sigma^2\right)}\right) \\ &= \log_e\left(\exp\left(\{-(y_i+1)^2 + (y_i-1)^2\}/2\sigma^2\right)\right) \\ &= -\frac{2}{\sigma^2}y_i. \end{aligned} \tag{6.1}$$

From (6.1), it can be seen that, if the vector y is hard decision decoded, the likelihood of the ith being correct is proportional to $|y_i|$.

The basic principle of Dorsch's (1974) algorithm is to treat the k most reliable bits of the received vector as correct and to treat the remaining $n-k$ least reliable bits as erasures. Let μ be a permutation on coordinates of y such that $\mu(y) = (y_{\mu_0}, y_{\mu_1}, \ldots, y_{\mu_{n-1}})$ and $|y_{\mu_i}| > |y_{\mu_j}|$ for integers $0 \leq i, j \leq n-1$ and $i < j$. Let $\mu(H)$ be the parity-check matrix of C whose coordinates have been rearranged according to the permutation μ. By means of elementary row operations, the last $n-k$ columns of $\mu(H)$ may form an identity matrix, i.e. $\mu(H) = [P \mid I_{n-k}]$. In this form, the aforementioned $n-k$ erasures can be directly solved from the remaining k columns of $\mu(H)$ to produce a

codeword $\hat{c} = (\hat{c}_i|\hat{c}_c)$, i.e. $\hat{c}_c = \hat{c}_i P^T$, where $\hat{c}_c$ is an $n-k$ tuple containing the erased bits and $\hat{c}_i$ is a k tuple containing the unerased bits.

In the event that the last $n-k$ columns of $\mu(H)$ cannot form an identity matrix, i.e. the last $n-k$ columns are linearly dependent, and hence these positions cannot be solved as erasures, the matrix $\mu(H)$ has to be rearranged until the last $n-k$ columns of this matrix are linearly independent.As a consequence of this rearrangement, some of the more reliable bits, say s bits[†], have to be erased, s least reliable bits have to be treated as the more reliable bits and the condition $|y_{\mu_i}| > |y_{\mu_j}|$ for $i < j$ is no longer satisfied. For convenience, let π be the final ordering of y such that the last $n-k$ columns of $\mu(H)$ are linearly independent and let this reordered final matrix be denoted by $\pi(H)$. Note that π is also represented as the final ordering even in the event that the initial permutation of y has already produced an identity matrix in the last $n-k$ coordinates of $\mu(H)$ and it is obvious that π is an identity permutation in this case.

The event that the last $n-k$ columns of $\mu(H)$ are solvable as erasures depends on the power of the code and also the positions of erasures. For binary codes, the occurrence of the opposite event is quite common. Only maximum-distance-separable (MDS) codes are capable of solving any $n-k$ erasures, MacWilliams and Sloane (1977), regardless of their positions. Unfortunately, the only MDS codes over $\mathbb{F}_2$ that exist are trivial codes: the single parity-check codes and their dual. However good binary codes are almost MDS, on average.

Without loss of generality, the codeword $\hat{c}$ can be expressed as $(\hat{c}_{\pi_0}, \hat{c}_{\pi_1}, \ldots, \hat{c}_{\pi_{n-1}})$. The first k coordinates $\hat{c}_i = (\hat{c}_{\pi_0}, \hat{c}_{\pi_1}, \ldots, \hat{c}_{\pi_{k-1}})$ are simply obtained from $\pi(y)$, received vector y whose coordinates are permuted according to π, using the following thresholding rule

$$\hat{c}_{\pi_i} = \begin{cases} 1 & \text{if } y_{\pi_i} \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{6.2}$$

where $0 \leq i \leq k-1$. The remaining $n-k$ coordinates $\hat{c}_c = (\hat{c}_{\pi_k}, \hat{c}_{\pi_{k+1}}, \ldots, \hat{c}_{\pi_{n-1}})$, which are erased, are solved using $\pi(H)$. The codeword $\hat{c} = (\hat{c}_i \mid \hat{c}_c)$ is either equal to the transmitted codeword, denoted by c, or needs only small changes to produce a codeword equal to c.

Given a received vector y, it is well-known in Proakis (2001) that a codeword most likely to be transmitted–denoted by c', is the one which has the smallest Euclidean distance, see Definition 3.3. This is equivalent to a codeword c' with the highest cross-correlation with the respect to the received vector

$$\mathcal{X}(c') = \sum_{j=0}^{n-1} v'_j y_j \tag{6.3}$$

where $v'_j = \psi(c'_j)$ and $\mathcal{X}(\bar{c}) < \mathcal{X}(c')$ for all $\bar{c} \in C$, $\bar{c} \neq c'$.

The Dorsch's (1974) decoding algorithm starts by evaluating $\mathcal{X}(\hat{c})$. The cross-correlation of a codeword and the received vector can be written in any order

$$\mathcal{X}(\hat{c}) = \sum_{j=0}^{n-1} \hat{v}_{\pi_j} y_{\pi_j}$$

[†] Depending of the code, s is usually a small integer.

and the cross-correlation can be split into two terms, i.e.

$$\mathcal{X}(\hat{c}) = \sum_{j=0}^{k-1} \hat{v}_{\pi_j} y_{\pi_j} + \sum_{j=k}^{n-1} \hat{v}_{\pi_j} y_{\pi_j}. \tag{6.4}$$

The first term of (6.4) may be written as

$$\sum_{j=0}^{k-1} \hat{v}_{\pi_j} y_{\pi_j} = \sum_{j=0}^{k-1} \mid y_{\pi_j} \mid$$

since the signs of $\hat{c}_{\pi_j}$ and that of y_{π_j} are equal for $0 \leq j \leq k-1$. Thus, this first term is independent of the code and (6.4) can be rewritten as

$$\mathcal{X}(\hat{c}) = \sum_{j=0}^{k-1} \mid y_{\pi_j} \mid + \sum_{j=k}^{n-1} \hat{v}_{\pi_j} y_{\pi_j}. \tag{6.5}$$

Let b be a vector over $\mathbb{F}_2$ which contains the hard-decision vector of y obtained using the same thresholding rule as (6.2). It should be noted that, in general, b is not a codeword. For convenience, let $\hat{z}$ be another vector over $\mathbb{F}_2$ defined as follows

$$\hat{z} = (\hat{z}_i \mid \hat{z}_c) = \hat{c} + b. \tag{6.6}$$

It is worth mentioning that $\hat{z}_i = 0$ since, as mentioned earlier, $\hat{c}_{\pi_j}$ and y_{π_j} have the same sign for $0 \leq j \leq k-1$. In the case of noiseless channel, the maximum attainable cross-correlation is given by

$$\mathcal{X}_{max} = \sum_{j=0}^{n-1} \mid y_{\pi_j} \mid,$$

which occurs when the received vector is equal to the transmitted codeword. Let $\sup(x)$ be the support of vector x and $\overline{\sup}(x)$ be coordinates of x for which the value is 0. Equation (6.5) may be expressed in terms of $\mathcal{X}_{max}$ as follows

$$\begin{aligned}
\mathcal{X}(\hat{c}) &= \sum_{j=0}^{k-1} \mid y_{\pi_j} \mid + \underbrace{\sum_{\pi_j \in \overline{\sup}(\hat{z}_e)} \mid y_{\pi_j} \mid - \sum_{\pi_j \in \sup(\hat{z}_e)} \mid y_{\pi_j} \mid}_{\sum_{j=k}^{n-1} \hat{v}_{\pi_j} y_{\pi_j}} \\
&= \sum_{j=0}^{k-1} \mid y_{\pi_j} \mid + \sum_{\pi_j \in \overline{\sup}(\hat{z}_e)} \mid y_{\pi_j} \mid - \sum_{\pi_j \in \sup(\hat{z}_e)} \mid y_{\pi_j} \mid + \sum_{\pi_j \in \sup(\hat{z}_e)} \mid y_{\pi_j} \mid - \sum_{\pi_j \in \sup(\hat{z}_e)} \mid y_{\pi_j} \mid \\
&= \mathcal{X}_{max} - 2 \sum_{\pi_j \in \sup(\hat{z}_e)} \mid y_{\pi_j} \mid .
\end{aligned} \tag{6.7}$$

This is equivalent to

$$\mathcal{X}(\hat{c}) = \mathcal{X}_{max} - \Delta(\hat{c}),$$

where evidently

$$\Delta(\hat{c}) = 2 \sum_{\pi_j \in \sup(\hat{z}_e)} | y_{\pi_j} | \tag{6.8}$$

is the shortfall from the maximum achievable cross-correlation for the codeword $\hat{c}$. Further observations can be made about the vector $\hat{z}$. Since $\hat{z}_i = 0$, the maximum possible weight $\hat{z}$ is $n - k$ and its average weight is $(n-k)/2$ at low E_b/N_o values. At high E_b/N_o values, the average weight $\hat{z}$ is small because there is a high chance that $\hat{c} = c$. It may be seen from (6.8) that, in general, $\mathrm{wt}_H(\hat{z})$ is directly proportional to the values of the correlation shortfall $\Delta(\hat{c})$ and it is inversely proportional to the cross-correlation value $\mathcal{X}(\hat{c})$.

Of course there is no guarantee that the codeword $\hat{c}$ is always equal to the transmitted codeword c. The code C has additional $2^k - 1$ codewords, one or more of these may produce higher cross-correlation value than $\mathcal{X}(\hat{c})$ and thus, the decoder has to evaluate more codewords in order to find such cases. Let these additional codewords be denoted by $c^{(l)}$ where $l = 1, \ldots, 2^k - 1$. Each of these codewords has an associated binary antipodal vector $v^{(l)}$ and the first term of the cross-correlation given by (6.5), is bound to be

$$\sum_{j=0}^{k-1} y_{\pi_j} v_{\pi_j}^{(l)} < \sum_{j=0}^{k-1} | y_{\pi_j} |$$

because, by definition, there has to be at least one bit among the coordinates $0, 1, \ldots, k-1$ of $c^{(l)}$ that is flipped with the respect to $\hat{c}$. In order for $\mathcal{X}(c^{(l)})$ to be larger than $\mathcal{X}(\hat{c})$, the second term of $\mathcal{X}(c^{(l)})$ has to be larger than the second term of $\mathcal{X}(\hat{c})$ plus the negative contribution from the first term. Since the coordinates of the received vector are ordered, however, the first term has higher magnitudes than the second term and it follows that codewords more likely to produce a higher cross-correlation value than $\mathcal{X}(\hat{c})$ will have small number of changes in the first k coordinates. As a consequence, codewords which have small number of differences in these first k coordinates have to be generated. These codewords are represented by $\bar{c}^{(l)} = \left(\bar{c}_i^{(l)} | \bar{c}_e^{(l)}\right)$ and are referred to as *low information weight* codewords. Thus, the codewords $c^{(l)}$ can be represented by

$$c^{(l)} = \left(c_i^{(l)} | c_e^{(l)}\right) = \hat{c} + \bar{c}^{(l)}. \tag{6.9}$$

The codewords $\bar{c}^{(l)}$ are generated such that $\mathrm{wt}_H\left(\bar{c}_i^{(l)}\right)$ increases as l is incremented. In this way, it becomes less likely for a codeword with larger l, compared to a codeword which has already been found, to have a higher cross-correlation value.

Similar to (6.8), the cross-correlation shortfall may be derived as a function of $\bar{c}^{(l)}$. Let the binary vector $z^{(l)}$ associated with $\bar{c}^{(l)}$ be given by

$$z^{(l)} = b + \hat{c} + \bar{c}^{(l)}, \tag{6.10a}$$

which, following (6.6), may be simplified to

$$z^{(l)} = \hat{z} + \bar{c}^{(l)}. \tag{6.10b}$$

Following (6.7), the cross-correlation of $c^{(l)}$ may be expressed as

$$\mathcal{X}\left(c^{(l)}\right) = \mathcal{X}_{max} - 2 \sum_{\pi_j \in \mathrm{sup}\left(z^{(l)}\right)} | y_{\pi_j} | \tag{6.11}$$

and equivalently,

$$\mathcal{X}\left(c^{(l)}\right) = \mathcal{X}_{max} - \Delta\left(c^{(l)}\right).$$

It is clear that the shortfall from maximum cross-correlation is

$$\Delta\left(c^{(l)}\right) = 2 \sum_{\pi_j \in \mathrm{sup}\left(z^{(l)}\right)} | y_{\pi_j} | \tag{6.12}$$

and substituting for $z^{(l)}$ gives $\Delta\left(c^{(l)}\right)$ as a function of $c^{(l)}$,

$$\Delta\left(c^{(l)}\right) = 2 \sum_{j=0}^{n-1} \left(\hat{z}_{\pi_j} + \bar{c}^{(l)}_{\pi_j} \pmod 2\right) | y_{\pi_j} | . \tag{6.13}$$

It is apparent that instead of determining $\mathcal{X}\left(c^{(l)}\right)$ for each codeword $c^{(l)}$, it is sufficient for the decoder to determine $\Delta\left(c^{(l)}\right)$ for each low information weight codeword $\bar{c}^{(l)}$, and compare the correlation shortfall value to the minimum value obtained so far, denoted by Δ_{min}, starting with $\Delta(\hat{c})$, i.e.

$$\Delta_{min} = \min\left\{\Delta(\hat{c}), \Delta\left(c^{(1)}\right), \Delta\left(c^{(2)}\right), \ldots, \Delta\left(c^{(l)}\right)\right\}.$$

From the above description, it can be seen that it is more efficient for the decoder to compute the cross-correlation of the codeword $\bar{c}^{(l)}$ instead of deriving $c^{(i)}$ by solving $\pi(H)$ and compute the corresponding squared Euclidean distance. Since the codeword $\bar{c}^{(l)}$ has an associated binary vector $z^{(l)}$ whose $\mathrm{wt}_H\left(z_i^{(l)}\right)$ is low, the number of non zero terms that need to be evaluated in (6.12) is typically $(n-k)/2$ rather than $n/2$ if (6.3) is evaluated, which results in an efficient and fast, soft-decision decoder. In practice, unless the dimension of the code is very small, only a fraction of $2^k - 1$, say L, codewords can be enumerated. At the end of decoding, the decoder will produce a codeword $\left(\hat{c} + \bar{c}^{(i)}\right)$ where $\bar{c}^{(i)}$ is the low weight information codeword that produces Δ_{min}.

In terms of implementation, the incremental correlation Dorsch decoder may be realised in an efficient decoder structure and this is depicted in Figure 6.1. The codeword $\bar{c}^{(i)}$ may be efficiently derived from a generator matrix of the code, which can be obtained from $\pi(H)$, using the revolving door algorithm described in Section 4.2.5. The revolving door algorithm has the property that between two successive combinations, there is only one element that is exchanged; one coordinate leaves and another enters the combination. Therefore, given a new codeword, the cross-correlation value for the first k terms can be easily computed by subtracting $2|y_{out}|$ and adding $2|y_{in}|$, where *out* is the coordinate that leaves and *in* is the coordinate that enters the combination. The cross-correlation value for the remaining $n-k$ coordinates can be easily computed once the support of $\left(\hat{z} + \bar{c}^{(l)}\right)$. This incremental correlation Dorsch decoder gives two levels of performance improvements and complexity trade-offs. Firstly, it is obvious that the number of processed low information weight codewords is directly proportional to the decoder's complexity, and also the higher the

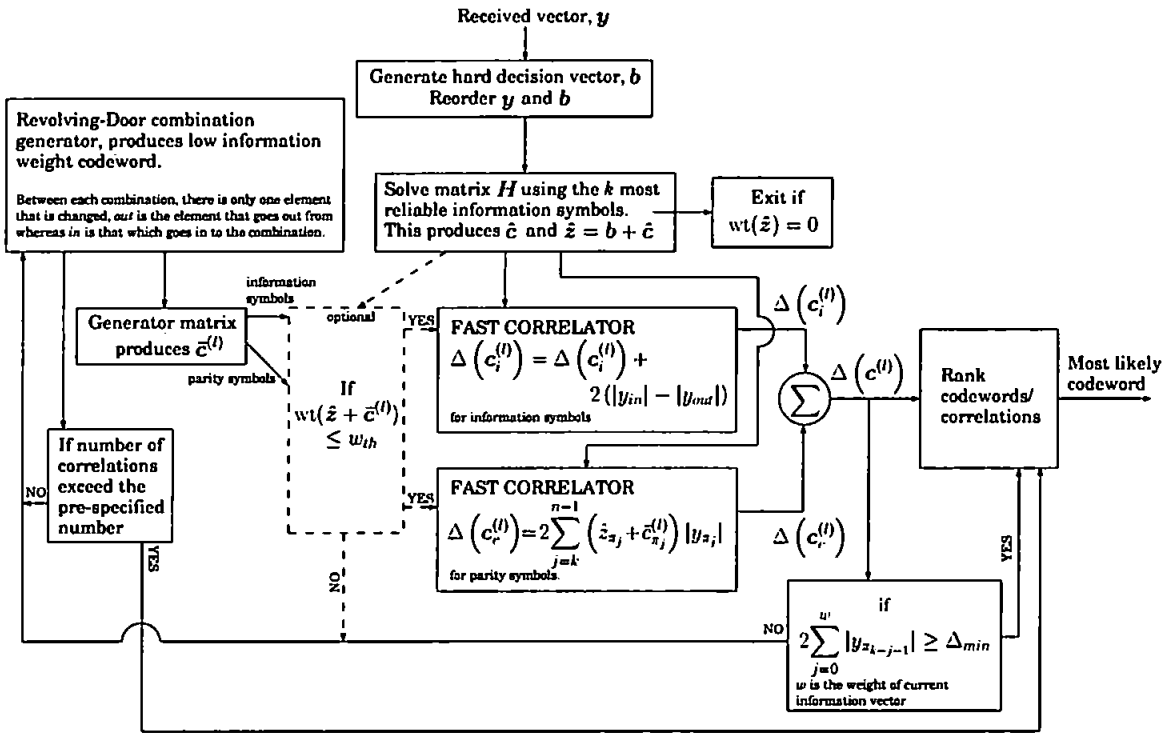


Figure 6.1: The structure of incremental correlation Dorsch decoder

number of processed codewords the more probable the decoder will find the maximum-likelihood codeword. The second level of performance trade-off is given by w_{th}, a threshold for the weight of vector $z^{(l)}$. The higher the weight of $z^{(l)}$, the less likely this codeword is the maximum likelihood solution. One can select a value for w_{th} so that vectors $z^{(l)}$ of weight higher than w_{th} are ignored and only sets of low weight information codewords are considered.

As a summary, the incremental correlation Dorsch decoding algorithm may be outlined in the flowchart depicted in Figure 6.2.

6.4 The Number of Codewords Required to Achieve Maximum-Likelihood Solution

The maximum information weight w_{max} necessary to achieve maximum-likelihood decoding may be upper-bounded from the magnitudes of the received vector and $\Delta(\hat{c})$ initially, and updated by Δ_{min} as the decoding progresses. It is known that

$$\Delta\left(c^{(l)}\right) \geq 2\sum_{j=0}^{w-1} |\, y_{\pi_{k-j-1}} \,| \tag{6.14}$$

where $\mathrm{wt}_H\left(\bar{c}_i^{(l)}\right) = w$. The bound given by (6.14) is reasonably tight since it is possible that there exists a vector $z^{(l)}$ where $\mathrm{wt}_H\left(z_i^{(l)}\right) = w_{max}$ and $\mathrm{wt}_H\left(z_c^{(l)}\right) = 0$. Correspondingly, w_{max} is the

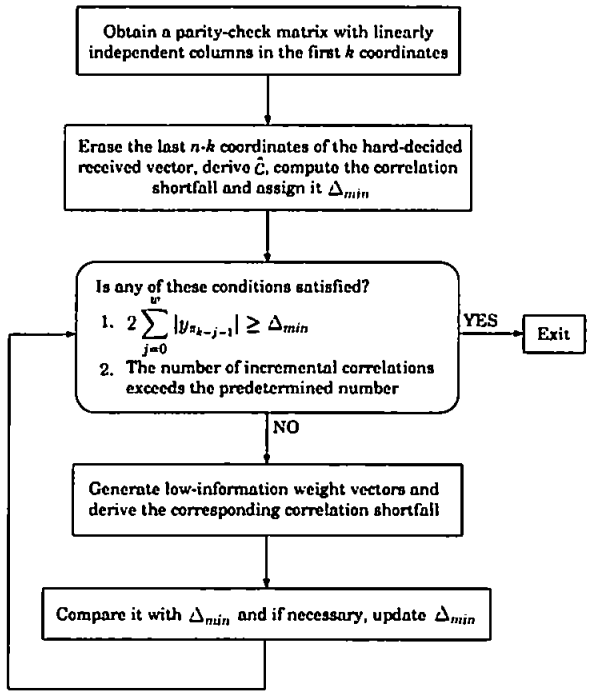


Figure 6.2: Outline of the incremental correlation Dorsch decoding algorithm

smallest integer such that

$$2\sum_{j=0}^{w_{max}-1} \mid y_{\pi_{k-j-1}} \mid \geq \Delta_{min}. \tag{6.15}$$

Nevertheless, for a given integer w, it is not necessary to enumerate all $\binom{k}{w}$ codewords that have information weight w. This is because a bound may be derived for Δ_{min} in terms of the coordinates of the information vector, the weight of the information vector and the magnitude of selected coordinates of the received vector. For an information weight w, let the integers $t \in \{0, 1, 2, \ldots, \tau(w)\}$ and $t' \in \{\tau(w)+1, \tau(w)+2, \ldots, k-w\}$ such that

$$\frac{1}{2}\Delta_{min} < \mid y_{\pi_t} \mid + \sum_{j=0}^{w-2} \mid y_{\pi_{k-j-1}} \mid \tag{6.16a}$$

and

$$\frac{1}{2}\Delta_{min} \geq \mid y_{\pi_{t'}} \mid + \sum_{j=0}^{w-2} \mid y_{\pi_{k-j-1}} \mid . \tag{6.16b}$$

Here, $\tau(w)$ is the maximum position in the received vector y among the first k coordinates such that (6.16a) is satisfied. The position of $\tau(w)$ in the first k coordinates may be illustrated by Figure 6.3. In this figure, it can be clearly seen that the permutation π applied to the coordinates of the received vector results in decreasing order of reliability of the received vector. Using $\tau(w)$, $N_{ML}(y)$, the minimum number of codewords that needs to be enumerated to achieve maximum-likelihood

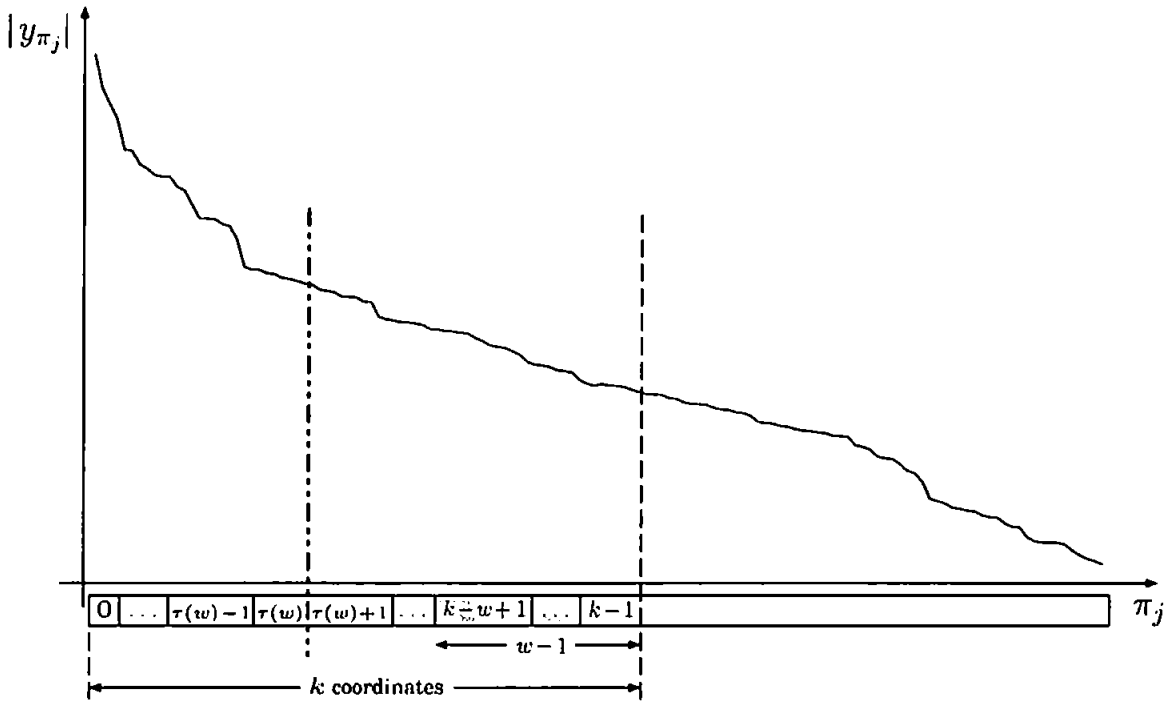


Figure 6.3: Position of $\tau(w)$ in the first k coordinates of a received vector

solution as a function of the received vector y, can expressed as

$$N_{ML}(y) = \sum_{w=0}^{w_{max}} \binom{k-(\tau(w)+1)}{w}. \tag{6.17}$$

For many short codes, it is found by means of simulations that $N_{ML}(y)$ is surprisingly small in comparison to the total number of codewords.

6.5 Numerical Results of Some Binary Codes with Large Minimum Distance

The decoder can be used for any linear code and best results are obtained for codes which have the highest minimum distance for a given block length and dimension, i.e. the best known linear codes which may be obtained from Grassl (2007). In the following, results for some powerful binary codes are presented. The numerical results for each code is compared against the Shannon's sphere packing lower bound offset by binary transmission loss, see Shannon (1959) and Butman and McEliece (1974). This bound is obtained numerically using the method described in Ahmed et al. (2007).

6.5.1 [136, 68, 24] Quadratic Double-Circulant Codes

A particular good class of linear codes is the class of binary self-dual doubly-even quadratic double-circulant codes discussed in Section 5.3.3. This class of codes includes the perfect $[24, 12, 8]$ Golay code, whose coordinates can be rearranged such that the parity-check matrix of this code is in

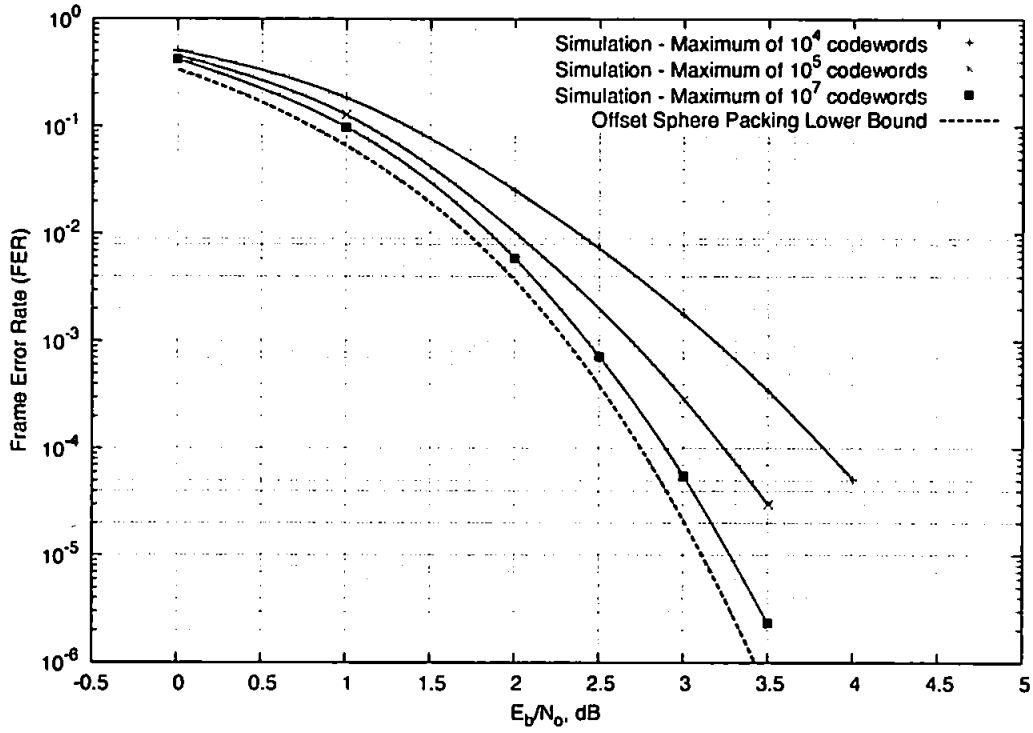


Figure 6.4: FER performance of the $[136, 68, 24]$ quadratic double-circulant code as a function of the number of codewords enumerated

double-circulant form; the $[48, 24, 12]$ extended quadratic residue code which, according to Houghten et al. (2003), is the only binary self-dual doubly-even code for this length; and the $[136, 68, 24]$ quadratic double-circulant code, which is the longest extremal self-dual code known to date.

The $[136, 68, 24]$ quadratic double-circulant code is in bordered double-circulant form, see (5.5b), with defining polynomial of the right circulant given by

$$\begin{aligned} b(x) = 1 + x + x^4 + x^6 + x^9 + x^{10} + x^{14} + x^{15} + x^{16} + x^{17} + x^{19} + x^{21} + x^{22} + x^{23} + \\ x^{24} + x^{25} + x^{26} + x^{29} + x^{33} + x^{35} + x^{36} + x^{37} + x^{39} + x^{40} + \\ x^{47} + x^{49} + x^{54} + x^{55} + x^{56} + x^{59} + x^{60} + x^{62} + x^{64} + x^{65}. \end{aligned}$$

The frame-error-rate (FER) of this code obtained using the incremental correlation Dorsch decoder and the offset sphere packing lower bound are depicted in Figure 6.4. Also shown in this figure is the FER performance of this code as a function of the maximum number of codewords enumerated. It can be seen from Figure 6.4 that the performance of the $[136, 68, 24]$ code is around 0.2 dB away from the offset sphere packing lower bound.

The conclusion from Figure 6.4 is that the performance of the $[136, 68, 24]$ quadratic double-circulant code in conjunction with the incremental correlation Dorsch decoder is within 0.2 dB of the best achievable performance for any $(136, 2^{68})$ binary code (linear or non linear) at 10^{-5} FER. Interestingly, as observed from computer simulations, there is a significant number of maximum-likelihood codeword errors, which have a Hamming distance of 36 or 40 from the transmitted code-

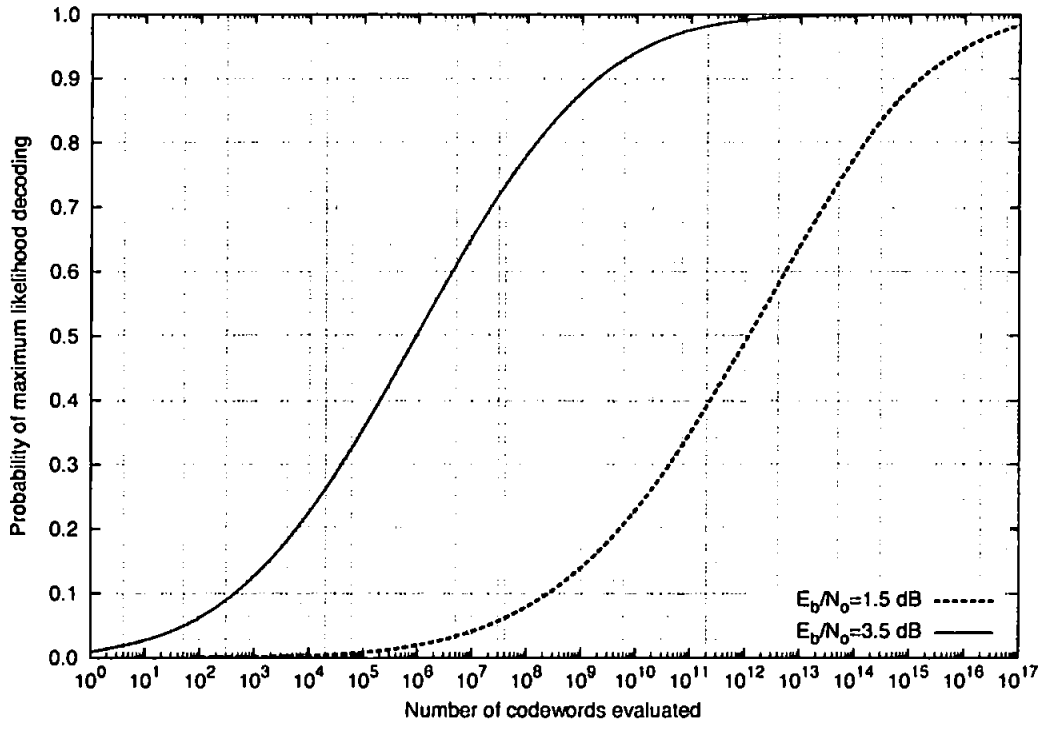


Figure 6.5: Probability of maximum likelihood decoding as a function of the number of codewords evaluated for the $[136, 68, 24]$ quadratic double-circulant code

word. This indicates that bounded distance decoding would not perform well for this code.

For each received vector y, as discussed earlier, $N_{ML}(y)$ codewords are required to be evaluated, see (6.17), in order to guarantee a maximum likelihood decoding. After enumerating a certain number of codewords and knowing $N_{ML}(y)$, the probability of maximum likelihood decoding can be estimated. Figure 6.5 shows the average probability of maximum likelihood decoding of the $[136, 68, 24]$ quadratic double-circulant code as a function of the number of codewords evaluated at operating points $E_b/N_o = 1.5$ dB and $E_b/N_o = 3.5$ dB. From Figure 6.5, it may be seen that evaluating 10^7 codewords per received vector gives 66% of probability of maximum likelihood decoding and for the remaining 35% of the received vectors, even though maximum likelihood solution is not guaranteed, its likelihood is relatively small as depicted in Figure 6.4.

A detailed operation of the decoder may be seen by considering an example of the received vector at the operating point of $E_b/N_o = 2.5$ dB. The magnitudes of the received coordinates, in the order of π, is shown in Figure 6.6. In this particular example, the initial ordering μ results in a singular parity-check matrix in the last $n-k$ least-reliable coordinates, leaving the μ_{68}th coordinate dependent. The permutation π which swaps the coordinates μ_{66} with μ_{68} unlock this dependency issue and this change is evident by inspecting the magnitude of the ordered received vector $\pi(y)$ in Figure 6.6. For this particular received vector, there are 16 coordinates that are in error and 14 of them are on the $n-k$ least-reliable coordinates. Using 10^7 maximum codewords, this decoder produces the minimum value of the cross-correlation shortfall $\Delta_{min} = 13.8$ and this occurs at the 640th codeword, for which the cross-correlation value $\mathcal{X}\left(c^{(640)}\right) = 126.2$ and $\mathcal{X}_{max} = 140$. The weight

of $z^{(640)}$, $\mathrm{wt}_H\left(z^{(640)}\right) = 16$ which corresponds to the 16 erroneously received bits.

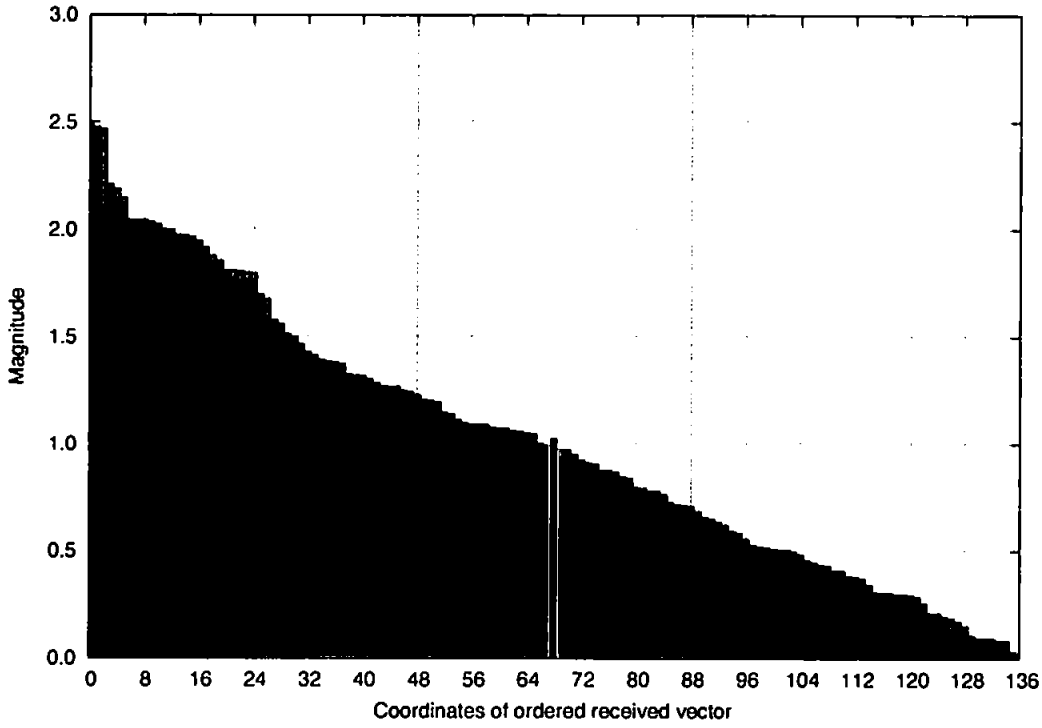


Figure 6.6: Magnitudes of an example received vector, ordered by π, at $E_b/N_o = 2.5$ dB

In practice, it is not necessary to evaluate $\Delta\left(\bar{c}^{(l)}\right)$ for every low information weight codeword $\bar{c}^{(l)}$ generated. In many cases, $\mathrm{wt}_H\left(z^{(l)}\right)$ is sufficiently high to indicate that the corresponding codeword $c^{(l)} = \left(\hat{c} + \bar{c}^{(l)}\right) = \left(z^{(l)} + b\right)$ is unlikely to be the maximum likelihood codeword. Figure 6.7 shows the cumulative probability distributions of $\mathrm{wt}_H\left(z^{(l)}\right)$ for the case where $c^{(l)}$ is equal to the transmitted codeword and the case where $c^{(l)}$ is not equal to the transmitted codeword. The plot in Figure 6.7 is obtained at operating point $E_b/N_o = 3.5$ dB. Considering a decoding rule that discard all $z^{(l)}$ of weight 28 or more. This means that

$$\frac{\sum_{w=0}^{27}\binom{68}{w}}{2^{68}} \times 100\% = 5.7\%$$

of total candidate codewords are considered and the remaining 94.3% candidate codewords may be rejected. From Figure 6.7, the probability of vector $z^{(l)}$ of weight 28 or more produces a correct codeword is approximately 10^{-6}, and from Figure 6.4, there are 2.3 frames in error per 10^6 frames transmitted at $E_b/N_o = 3.5$ dB. As a consequence of this decoding rule, the FER is degraded to 3.3×10^{-6}.

6.5.2 [154, 77, 23] Best Known Linear Code

In Chapter 4, it is shown that there exists a $[154, 77, 23]$ binary linear code which improves the lower-bound on the minimum distance of any half-rate linear code over $\mathbb{F}_2$ of dimension 77 by one, i.e. the previously considered best known linear code was the code $[154, 77, 22]$. The FER performance of

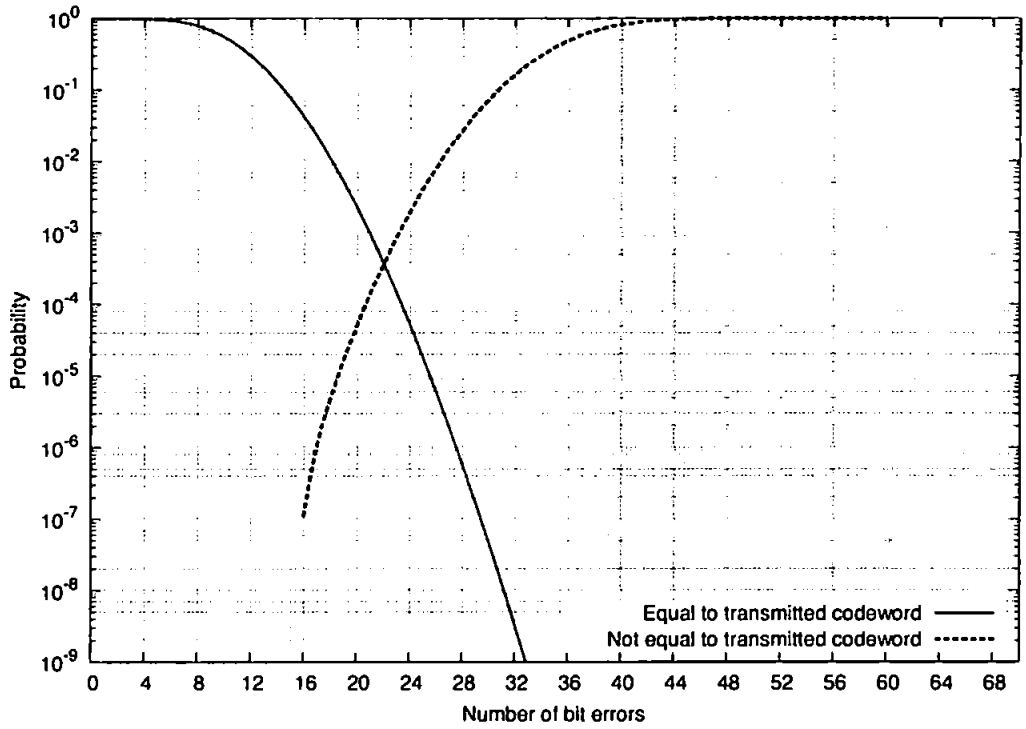


Figure 6.7: Cumulative probability distributions of $\mathrm{wt}_H\left(z^{(l)}\right)$ at $E_b/N_o = 3.5$ dB

this code, which is derived from cyclic code of length 151, under the incremental correlation Dorsch decoder is depicted in Figure 6.8. It may be seen that, at 10^{-5} FER, the $[154, 77, 23]$ best known linear code is approximately 2.5 dB away from the offset sphere packing lower bound.

6.5.3 [255, 175, 17] Cyclotomic Idempotent LDPC Code

This code is from a class of the One-Step Majority-Logic Decodable codes originally used in hard-decision majority-logic decoding (Lin and Costello, Jr.; 2004). It has the same parameter as the code in the family of Euclidean geometry codes which are shown by Kou et al. (2001) that they are iteratively decodable as LDPC codes. This $[255, 175, 17]$ code, which is cyclic, can be easily constructed using the theory of cyclotomic cosets and idempotents as shown in Section 2.2. The parity-check matrix of this cyclotomic idempotent LDPC code consists of 255 cyclic shifts of the following polynomial

$$m(x)h(x) = 1 + x + x^3 + x^7 + x^{15} + x^{26} + x^{31} + x^{53} + x^{63} + x^{98} + x^{107} + x^{127} + x^{140} + x^{176} + x^{197} + x^{215}.$$

Figure 6.9 shows the FER performance of this cyclic LDPC codes under the standard belief propagation and incremental correlation Dorsch decoding with 5.5×10^6 codewords. With this arrangement, the Dorsch algorithm gains around 0.4 dB over the belief-propagation algorithm at 10^{-4} FER. A hybrid decoding approach may also be utilised to decode this LDPC code, that is by executing the Dorsch algorithm after the belief propagation has completed. There is an option of executing

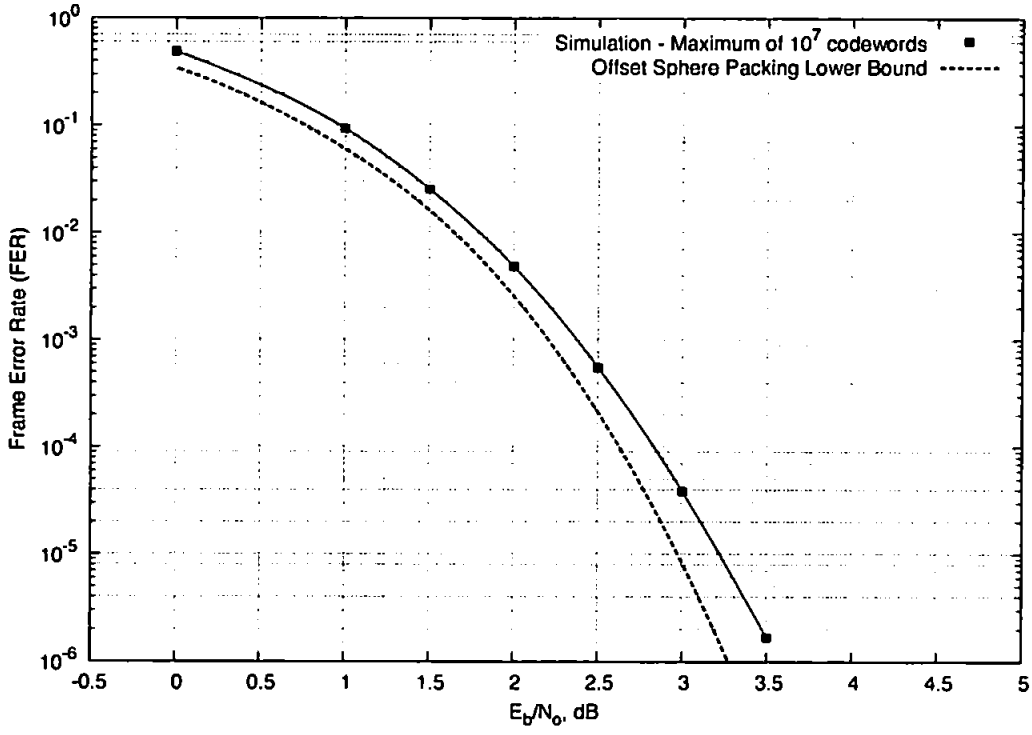


Figure 6.8: FER performance of the [154, 77, 23] best known linear code

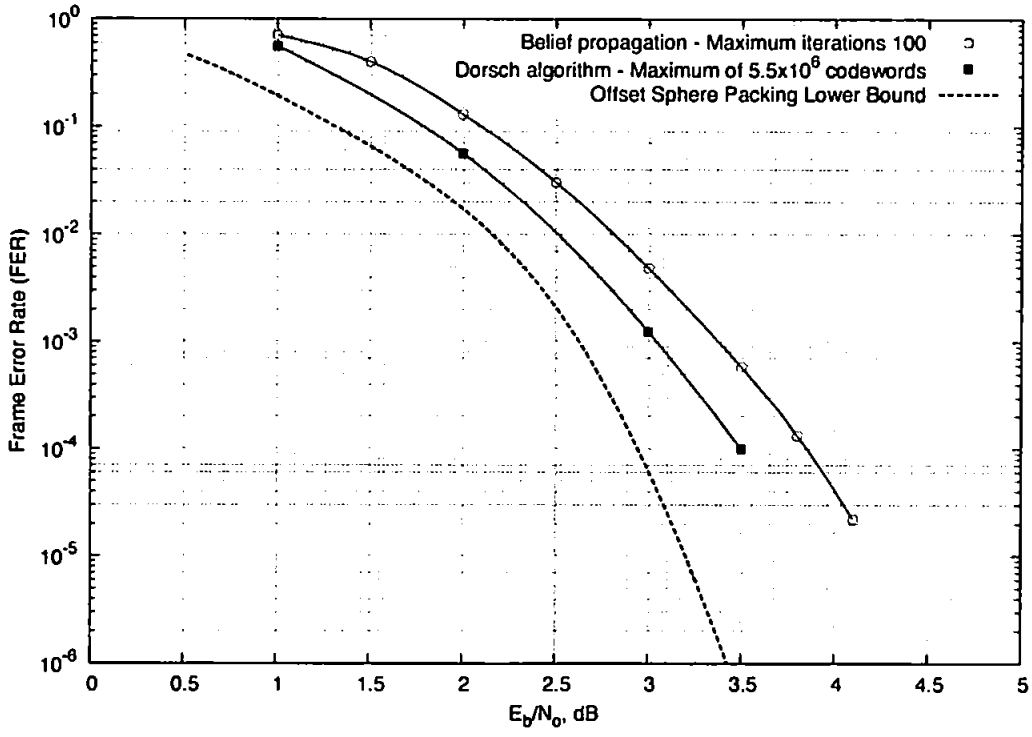


Figure 6.9: FER performance of the [255, 175, 17] cyclic LDPC code

the Dorsch algorithm for the non convergent blocks or for all blocks output by the belief propagation decoder. A more complex hybrid arrangement can also be employed, that is by executing the Dorsch algorithm after every i iterations of belief propagation algorithm, see e.g. Fossorier (2001).

6.5.4 BCH and Goppa Codes

The extended BCH code, $[128, 64, 22]$, is a powerful binary error correcting code, which has appeared many times in the published literature to benchmark various decoders' performance. Figure 6.10 shows the FER performance of this powerful extended BCH code and it is clear that it has a performance around 0.2 dB away from the best $(128, 2^{64})$ binary code over the entire range of E_b/N_o simulated.

For relatively long codes, good FER performance may also be obtained using the incremental correlation Dorsch decoder. This is shown in Figure 6.11 which plots the FER curves of the 4-error correcting BCH code of length 1023, $[1023, 983, 9]$ primitive BCH code, under incremental correlation Dorsch decoding and algebraic hard decision decoding. The offset sphere packing lower bound is also shown in this figure and we can see that the FER performance of this code, using the incremental correlation Dorsch algorithm with a maximum of 10^6 codewords , is around 1.4 dB away from the bound at 10^{-5} FER. Although this may seem excessive compared to the other codes previously considered, there is a gain of approximately 1.5 dB over the hard decision decoder.

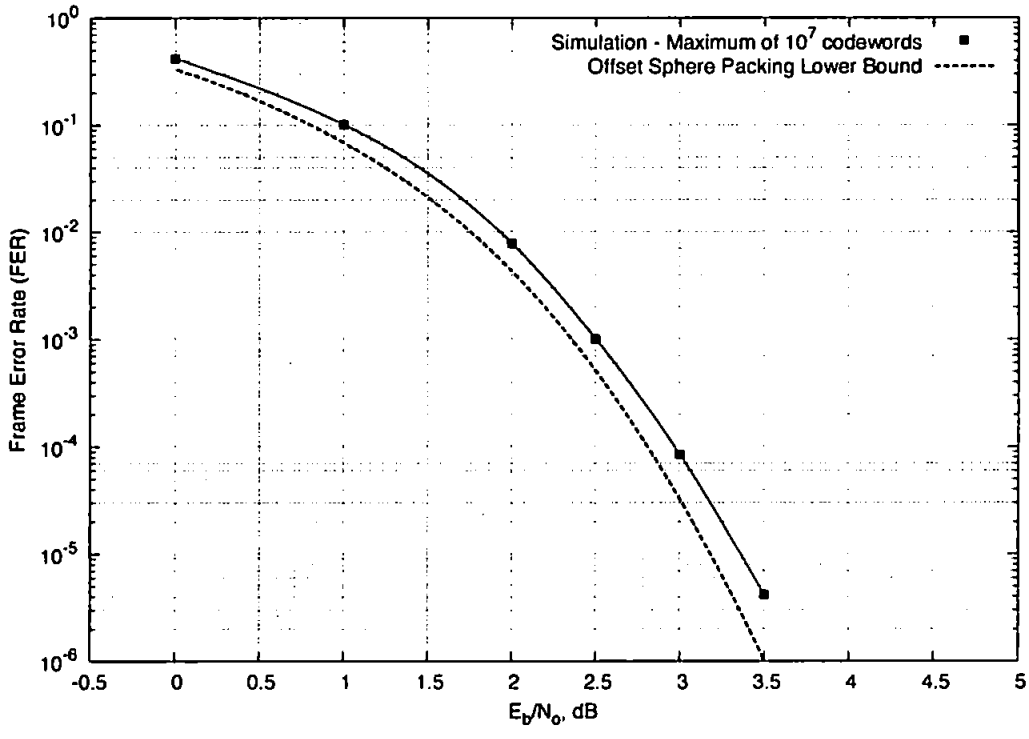


Figure 6.10: FER performance of the $[128, 64, 22]$ extended BCH code

Goppa codes are commonly regarded as better codes than the corresponding primitive BCH codes of the same length and dimension. This is because, compared to a BCH code having the same number of redundant symbols, the Goppa code has one extra information symbol. As an example,

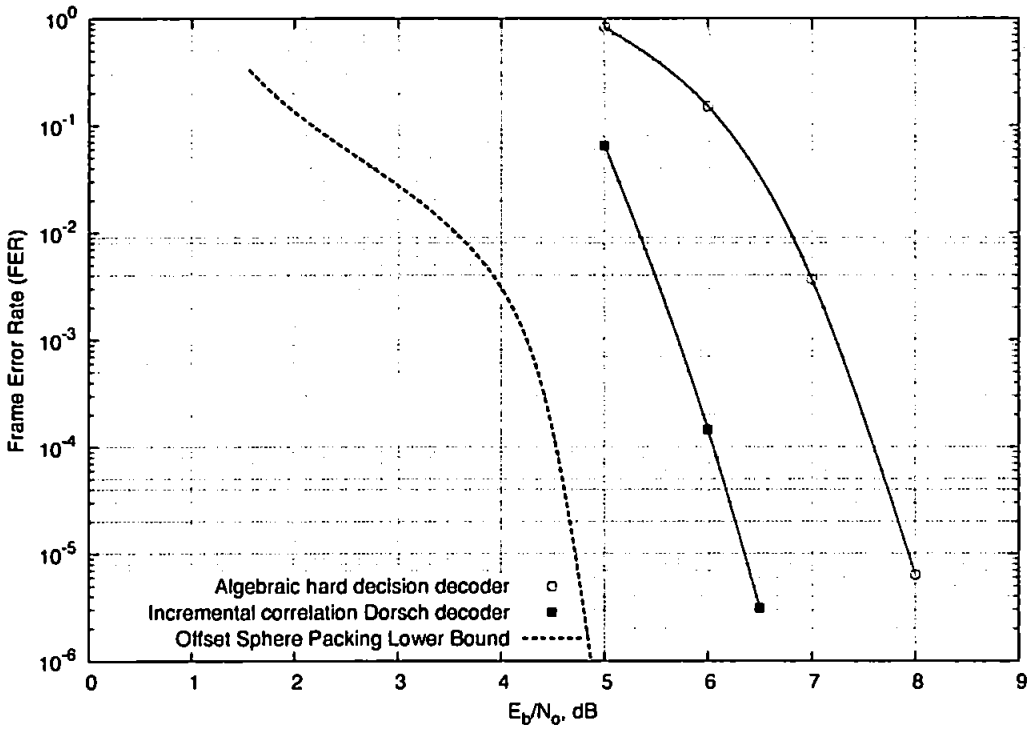


Figure 6.11: FER performance of the $[1023, 983, 9]$ primitive BCH code

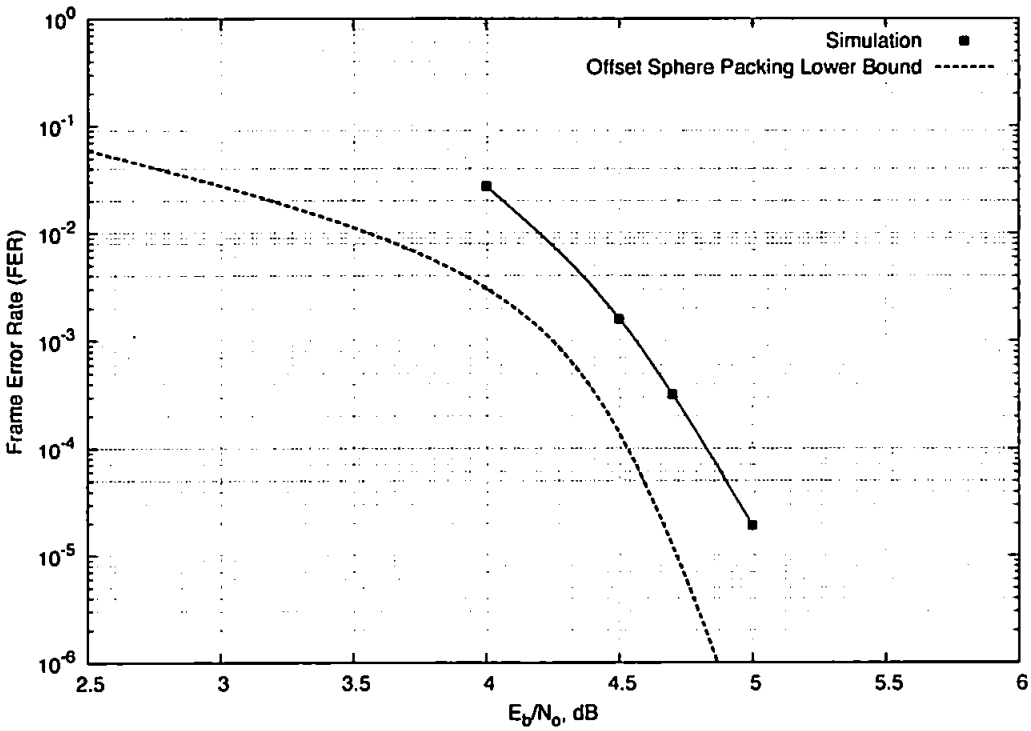


Figure 6.12: FER performance of the $[513, 467, 12]$ extended Goppa code

there exists a $[511, 466, 11]$ primitive BCH code generated by generator polynomial $g(x) = 1+x^2+x^5$, whose roots have order $2^5 - 1 = 31$ which is relatively prime to 511. Using $g(x)$ as the Goppa polynomial, a $[512, 467, 2\deg(g(x)) + 1 = 11]$ Goppa code can be constructed. It is clear that an overall parity-check bit can be annexed to this code to produce a $[513, 467, 12]$ extended Goppa code. The FER performance of this extended Goppa code is shown in Figure 6.12. Compared to the offset sphere packing lower bound, which is also shown in Figure 6.12, at 10^{-4} FER, the realised performance of the code using the incremental correlation Dorsch decoder is within 0.35 dB.

It is relatively simple to produce a list decoder from the incremental correlation Dorsch decoding algorithm. In addition, this incremental correlation Dorsch algorithm may also be straightforwardly modified to decode non binary linear codes such as Reed-Solomon and Algebraic Geometry codes. Note that there also exist an alternative list decoding decoding algorithm for these non binary codes: the Guruswami-Sudhan (GS) decoding algorithm, see Guruswami and Sudan (1999) and Guruswami (2001). The GS decoding algorithm is a type of list decoder that is capable of correcting more errors than a bounded-distance decoder. The GS decoding algorithm is based on polynomial interpolation and Koetter and Vardy (2003) has extended this interpolation concept to produce a soft-decision decoder for Reed-Solomon codes, which is commonly known as the Koetter-Vardy (KV) algorithm.

6.6 Summary

- Dorsch's (1974) decoding algorithm may be extended to approach maximum likelihood decoding by an incremental correlation approach in which for each received vector, a partial summation metric is evaluated as a function of low weight information codewords.
- The upper-bound on the number of codewords that need to be enumerated in order to achieve a maximum likelihood solution may be obtained as the decoding progresses.
- If binary transmission is employed, the extension of this decoding algorithm to non binary codes is straightforward by using the binary image of the non binary code.
- It has been shown that for many powerful codes of short block lengths, performance that is close to the offset sphere packing lower bound can be achieved. To the best of author's knowledge, there are no equivalent, iteratively decodable codes that have similar performance.
- The decoder may be efficiently implemented by employing Nijenhuis and Wilf's (1978) revolving door combination generator to generate low weight information codewords, where for two successive codewords, there is only one element that is flipped in the k most-reliable coordinates. Furthermore, a threshold may be imposed on the low weight information vector in order to allow performance and complexity trade-off.
- The incremental correlation Dorsch decoder may be easily modified to become a *maximum likelihood list decoder*, which as the name implied, generates a list of candidates codewords ranked in the order of increasing squared Euclidean distance to the received vector. A useful application of this maximum likelihood list decoding is in novel CRC-less error detection, which is discussed in Chapter 7.

Part IV

Application of Coding

7 Incremental Redundancy Communications

A typical communication system uses channel coding to ensure reliable transmission of information from source to destination. Channel coding can be employed in two different ways: error correction and error detection. In both ways, a message is split into several data blocks and for each data block, some redundant symbols are obtained (these redundant symbols are functions of the symbols in the data block) and appended to each data block. In transmission, instead of sending strings of data blocks only, strings of data and redundant symbols are transmitted. In the error correction case, these redundant symbols are used by the decoder in the receiving end to correct any errors that may have occurred during transmission. In the error detection case, on the other hand, channel coding is typically used in conjunction with automatic-repeat-request (ARQ) scheme. Upon the receipt of the data and redundant symbols, the decoder recomputes a new set of redundant symbols based on the received data block. If the recomputed redundant symbols are the same as those of the received symbols, successful transmission is declared by sending an acknowledgement (ACK) to the transmitter, otherwise an error is detected and a negative acknowledgement (NACK) results in a retransmission.

While this retransmission scheme can guarantee high data transfer reliability, repeating the transmission of the entire block is certainly not efficient in terms of throughput. A more efficient system can be achieved by retransmitting a new set of redundant symbols. This technique is commonly known as the incremental redundancy (IR) ARQ scheme. In this arrangement, two levels of error protection are employed: the first level employs an error-detecting code, such as a cyclic-redundancy-check (CRC) code, to detect errors in the user message; and the second level employs an error-correcting code which protects both user message and its check.

The idea of incremental redundancy dates back to the work of Davida and Reddy (1972) and later that of Mandelbaum (1974). A historical overview of the development of this retransmission scheme, which is also known as the type-II hybrid ARQ (HARQ), is given in Lin and Costello, Jr. (2004). Since the discovery of turbo codes and the rediscovery of low-density parity-check (LDPC) codes, there is a growing interest in the IR-ARQ scheme using these iteratively-decodable codes (Narayanan and Stuber; 1997; Liu et al.; 2003; Sesia et al.; 2004; Soljanin et al.; 2006). These iteratively decodable codes form a class of low-decoding complexity codes which, in general, have relatively poor minimum distance. The performance gain of this class of codes comes from their large block length and hence, they are not suitable for short packet applications.

This chapter discusses approaches of code construction and error detection for short packet IR-ARQ applications. Only construction of binary, general linear codes is considered in this chapter and the feedback channel of the IR-ARQ scheme is assumed noiseless.

7.1 Overview of Incremental Redundancy Codes

Incremental redundancy codes have the property that all symbols of the higher-rate code are contained in the lower-rate code. There are two approaches to construct codes with this property. The most commonly used approach is to use a good low-rate code, which is then successively punctured to produce higher-rate codes. Iteratively-decodable codes for incremental redundancy are constructed in this manner (Narayanan and Stuber; 1997; Liu et al.; 2003; Soljanin et al.; 2006). and so are the codes that appeared in Davida and Reddy (1972) and Mandelbaum (1974). Using this approach, the minimum distance of the punctured code can collapse unless there is a proper selection of puncturing patterns and to determine good puncturing patterns is non trivial. A second approach is to start with a good high-rate code and successively add parity check symbols to produce longer lower-rate codes with higher minimum distance. Some of the codes obtained from this approach include the codes introduced by Krishna and Morgera (1987), the short codes of optimal weight structure from a computer search carried out by Cygan and Offer (1991), and the $(u|u+v)$ construction of Reed-Muller codes (Wicker and Bartz; 1994a). In this chapter, some methods of constructing incremental redundancy codes based on the second approach are shown.

In incremental redundancy with M transmissions, the property of an $[n,k,d]$ incremental redundancy codes C implies that its codeword $c \in C$ can be partitioned into M subblocks of codeword, i.e. $c = (u|p^{(1)}|p^{(2)}|\ldots|p^{(M)})$, where $c^{(1)} = (u|p^{(1)}) \in C^{(1)}$, $c^{(2)} = (c^{(1)}|p^{(2)}) \in C^{(2)}$, ..., $c = c^{(M)} = (c^{(M-1)}|p^{(M)}) \in C^{(M)} = C$. Here u denotes the information block of length k, $p^{(i)}$ denotes the ith parity block of length r_i and $C^{(i)}$ denotes an $[n_i = k + \sum_{j=1}^{i} r_j, k, d_i]$ code. It is desirable to have $d_i > d_{i-1}$ for $2 \le i \le M$, however, it is not trivial to append parity symbols to increase the minimum distance while maintaining high code-rate of the overall code. Nonetheless, this can be neatly achieved, as shown in Section 7.2, by applying Constructions X and XX to a chain of cyclic codes.

7.2 Juxtaposition Codes: Chain of Cyclic Codes with Constructions X and XX

If C is a cyclic code, there exists a generator polynomial $g(x) \in \mathbb{F}_2[x]$ and a check polynomial $h(x) \in \mathbb{F}_2[x]$ such that $g(x)h(x) = x^n - 1$. Two cyclic codes, C_1 with $g_1(x)$ as the generator polynomial and C_2 with $g_2(x)$ as the generator polynomial, are said to be chained or nested, if $g_1(x)|g_2(x)$, and they are denoted by $C_1 \supset C_2$. With reference to this definition, it is clear that narrow-sense BCH codes of the same length form a chain of cyclic codes.

Let $G = [I_k| - R]$ be an $k \times n$ reduced-echelon generator matrix of a cyclic code with generator polynomial $g(x)$, the ith row of G is $[x^k(x^{n-k+i-1} - r_{n-k+i-1}(x))]$ where $r_j(x) = x^j \bmod g(x)$.

7.1 ***Lemma.*** Consider a chain of cyclic codes, $C_1 \supset C_2$, where $g_1(x) = f_1(x)$ and $g_2(x) = f_1(x)f_2(x)$, the

reduced-echelon generator matrices of these cyclic codes can be written as

$$G_1 = \left[\begin{array}{c|c|c} & & -r_{2,n-k_2}(x) \\ & & -r_{2,n-k_2+1}(x) \\ I_{k_2} & & \vdots \\ & & -r_{2,n-2}(x) \\ & & -r_{2,n-1}(x) \\ \hline 0 & I_{k_1-k_2} & -r_{1,n-k_1+k_2}(x) \\ & & \vdots \\ & & -r_{1,n-1}(x) \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} G_2 \\ \\ \end{array}$$

where G_i is the reduced-echelon generator matrix of C_i, I_k is a $k \times k$ identity matrix and the polynomial $r_{i,j}(x) = x^j \bmod g_i(x)$.

Proof. It is obvious that the first k_2 rows of G_1 is G_2. The proof that the polynomials formed by the last $k_1 - k_2$ rows of G_1 do not contain $f_2(x)$, where $\deg(f_2(x)) = k_1 - k_2$, as a factor, is required. From the $(k_2 + i + 1)$th row of G_1, for $0 \le i \le k_1 - k_2 - 1$, $u_i(x)$ can be expressed as

$$u_i(x) = \frac{x^{k_2+i} - x^{k_1} r_{1,n-k_1+k_2+i}(x)}{f_1(x)}.$$

For $0 \le i \le k_1 - k_2 - 1$, the numerator has degree at most $n - k_2 - 1$ and $\deg(f_1(x)) = n - k_1$, so the maximum degree of $u_i(x)$ is $k_1 - k_2 - 1$ and therefore, $f_2(x) \nmid u_i(x)$ since $\deg(f_2(x)) = k_1 - k_2$. □

Given a chain of two codes, using Construction X (Sloane et al.; 1972), the code with larger dimension can be lengthened to produce a code with increased length and minimum distance. A generalised Construction X which involves more than two codes is given in the following theorem, see also Bierbrauer and Edel (1997).

7.1 ***Theorem (Generalised Construction X).*** Let $\mathcal{B}_i$ be an $[n, k_i, d_i]$ code, given a chain of M codes, $\mathcal{B}_1 \supset \mathcal{B}_2 \supset \ldots \supset \mathcal{B}_M$, and a set of auxiliary codes $\mathcal{A}_i = [n'_i, k'_i, d'_i]$, for $1 \le i \le M - 1$, where $k'_i = k_1 - k_i$, a code $C_X = [n + \sum_{i=1}^{M-1} n'_i, k_1, d]$ code can be constructed, where $d = \min\{d_M, d_{M-1} + d'_{M-1}, d_{M-2} + d'_{M-2} + d'_{M-1}, \ldots, d_1 + \sum_{i=1}^{M-1} d'_i\}$.

From Theorem 7.1, the corollary below follows.

7.1 ***Corollary.*** Let v be a vector of length n formed by the first n coordinates of a codeword of C_X. A

codeword of $\mathcal{C}_X$ is a juxtaposition of codewords of $\mathcal{B}_i$ and $\mathcal{A}_i$, where

$$\begin{array}{lll}
(b_M \mid 0 \mid 0 \mid \ldots \mid 0 \mid 0) & \text{if } v \in \mathcal{B}_M, \\
(b_{M-1} \mid 0 \mid 0 \mid \ldots \mid 0 \mid a_{M-1}) & \text{if } v \in \mathcal{B}_{M-1}, \\
(b_{M-2} \mid 0 \mid 0 \mid \ldots \mid a_{M-2} \mid a_{M-1}) & \text{if } v \in \mathcal{B}_{M-2}, \\
\vdots & \vdots \\
(b_2 \mid 0 \mid a_2 \mid \ldots \mid a_{M-2} \mid a_{M-1}) & \text{if } v \in \mathcal{B}_2, \\
(b_1 \mid a_1 \mid a_2 \mid \ldots \mid a_{M-2} \mid a_{M-1}) & \text{if } v \in \mathcal{B}_1.
\end{array}$$

Note that $b_i \in \mathcal{B}_i$ and $a_i \in \mathcal{A}_i$.

Another lengthening method is Construction XX, which uses a lattice of four codes and makes use of Construction X twice. This construction was introduced by Alltop (1984) and is restated in Theorem 4.2 in Chapter 4.

7.2 *Corollary.* Let v be a vector of length n formed by the first n coordinates of a codeword of $\mathcal{C}_{XX}$. A codeword of $\mathcal{C}_{XX}$ is a juxtaposition of codewords of $\mathcal{B}_i$ and $\mathcal{A}_i$, where

$$\begin{array}{ll}
(b_4 \mid 0 \mid 0) & \text{if } v \in \mathcal{B}_4, \\
(b_3 \mid a_1 \mid 0) & \text{if } v \in \mathcal{B}_3, \\
(b_2 \mid 0 \mid a_2) & \text{if } v \in \mathcal{B}_2. \\
(b_1 \mid a_1 \mid a_2) & \text{if } v \in \mathcal{B}_1.
\end{array}$$

From Corollaries 7.1 and 7.2, assuming that $b_j = (u|p^{(1)})$ and $a_i = (p^{(i+1)})$ for some positive integers j and i, it is obvious that the codes obtained from Constructions X and XX follow the property of incremental redundancy codes. Cyclic codes of length n, with appropriate arrangement of their zeros, can be put in chain form and hence, they are good candidate linear codes for Constructions X and XX. It is known from the literature that a cyclic code can have the highest minimum distance attainable by any $[n, k]$ linear code. Due to their nested structure, it is possible to extend cyclic codes using Construction X or XX to produce more codes which have the highest minimum distance, see Bierbrauer and Edel (1997), Grassl (2001), Tjhai, Tomlinson, Grassl, Horan, Ahmed and Ambroze (2006), and Tjhai and Tomlinson (2007) as examples. It is also known that narrow-sense BCH codes are nested and so are the extended codes.

Chain of BCH codes, Construction X and incremental redundancy codes

Example 7.1: Consider the following chain of extended BCH codes of length 128,

$$\mathcal{B}_1 = [128, 113, 6] \supset \mathcal{B}_2 = [128, 92, 12] \supset \mathcal{B}_3 = [128, 78, 16] \supset \mathcal{B}_4 = [128, 71, 20].$$

Applying Construction X to $[128, 113, 6] \supset [128, 92, 12]$ with an $[32, 21, 6]$ extended BCH code as auxiliary code, a $[160, 113, 12]$ code is obtained and it follows that

$$[160, 113, 12] \supset [160, 92, 12] \supset [160, 78, 16] \supset [160, 71, 20].$$

Using a $[42, 35, 4]$ shortened extended Hamming code as the auxiliary code in applying Construction X to $[160, 113, 12] \supset [160, 78, 16]$ yields

$$[202, 113, 16] \supset [202, 92, 16] \supset [202, 78, 16] \supset [202, 71, 20].$$

Finally, applying Construction X to $[202, 113, 16] \supset [202, 71, 20]$ with the shortened extended Hamming code $[49, 42, 4]$ as the auxiliary code yields

$$[251, 113, 20] \supset [251, 92, 20] \supset [251, 78, 20] \supset [251, 71, 20].$$

A sequence of codes for IR-ARQ $[128, 113, 6]$, $[160, 113, 12]$, $[202, 113, 16]$ and $[251, 113, 20]$ is obtained.

The generator matrix of the $[251, 113, 20]$ code may be written in a form given by

$$G = \left(\begin{array}{c|c|c|c|c||c|c|c}
I_{71} & & & -R_4 & & 0 & 0 & 0 \\ \hline
 & I_7 & & -R_3 & & & & G_{\mathcal{A}_3} \\ \hline
0 & & I_{14} & -R_2 & & & G_{\mathcal{A}_2} & \\ \hline
 & & & I_{21} & -R_1 & G_{\mathcal{A}_1} & &
\end{array}\right). \tag{7.1}$$

On the left hand side of the double bar, the generator matrix of the code $\mathcal{B}_1$ is decomposed along the chain $\mathcal{B}_1 \supset \mathcal{B}_2 \supset \mathcal{B}_3 \supset \mathcal{B}_4$ (see Lemma 7.1). The matrices $G_{\mathcal{A}_i}$, for $1 \leq i \leq 3$ are the generator matrices of the auxiliary codes $\mathcal{A}_i$. In this form, obtaining the generator matrices for the shorter codes in the sequence is straightforward.

Chain of BCH codes, Construction XX and incremental redundancy codes

Example 7.2: Refining the chain of extended BCH codes from Example 7.1, a lattice of extended cyclic codes shown in Fig. 7.1 can be constructed. All, except $\mathcal{B}_2$ and $\mathcal{B}_6$, are extended primitive narrow-sense BCH codes. Let $\hat{\mathcal{B}}_i$ be the cyclic code obtained by removing the overall parity-check of $\mathcal{B}_i$ and let α be a primitive nth root of unity. Let $Z_{\hat{\mathcal{B}}_i}$ denote the representative zeros of $\hat{\mathcal{B}}_i$, the cyclic codes $\hat{\mathcal{B}}_2$ and $\hat{\mathcal{B}}_6$ have zeros $Z_{\hat{\mathcal{B}}_1} \cup \{\alpha^9\}$ and $Z_{\hat{\mathcal{B}}_4} \cup \{\alpha^{13}, \alpha^{15}\}$ respectively. Applying Construction XX to the lattice of $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ with auxiliary codes $\mathcal{A}_1 = [8, 7, 2]$ (single parity-check code) and $\mathcal{A}_2 = [20, 14, 4]$ (shortened extended Hamming code), the following chains are obtained,

$$[136, 113, 8] \supset [136, 92, 12] \supset [136, 78, 14] \supset [136, 71, 20],$$
$$[156, 113, 12] \supset [156, 92, 12] \supset [156, 78, 14] \supset [156, 71, 20].$$

Using $\mathcal{A}_3 = [40, 28, 6]$ and $\mathcal{A}_4 = [47, 35, 6]$ (best known linear codes from Grassl (2007)) as the auxiliary codes in applying Construction XX to the lattice of codes $[156, 113, 12] \supset [156, 85, 14]$ and $[156, 113, 12] \supset [156, 78, 14]$, where the subcodes are obtained by padding zeros to the codes $\mathcal{B}_6$ and

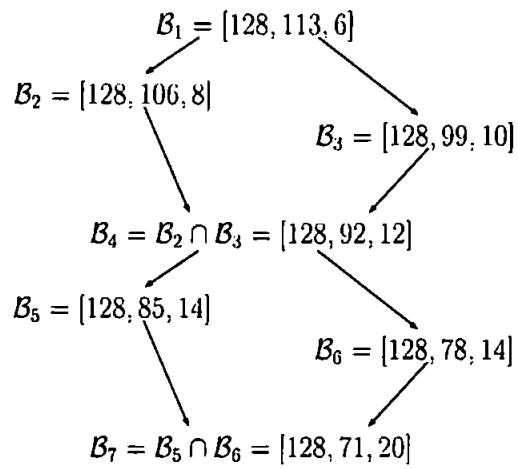


Figure 7.1: Lattice of extended cyclic codes

$\mathcal{B}_5$, it follows that

$$[196, 113, 14] \supset [196, 92, 14] \supset [196, 78, 14] \supset [196, 71, 20],$$
$$[243, 113, 20] \supset [243, 92, 20] \supset [243, 78, 20] \supset [243, 71, 20].$$

A sequence of code for IR-ARQ $[128, 113, 6]$, $[136, 113, 8]$, $[156, 113, 12]$, $[196, 113, 14]$ and $[243, 113, 20]$ is obtained.

The generator matrix of the $[243, 113, 20]$ code can be written in a form given by

$$G = \left(\begin{array}{c|c|c|c|c|c||c|c|c|c}
I_{71} & & & & & -R_5 & 0 & 0 & 0 & 0 \\ \hline
 & I_{14} & & & & -R_4 & & & & \\ \cline{2-6}
 & & I_7 & & & -R_3 & & & & \tilde{G}_{\mathcal{A}_4} \\ \cline{3-6}
0 & & & I_{14} & & -R_2 & & \tilde{G}_{\mathcal{A}_2} & G_{\mathcal{A}_3} & \\ \cline{4-6}
 & & & & I_7 & -R_1 & G_{\mathcal{A}_1} & & &
\end{array}\right). \tag{7.2}$$

On the left hand side of the double bar, the generator matrix of the code $\mathcal{B}_1$ is decomposed along the chain $\mathcal{B}_1 \supset \mathcal{B}_2 \supset \mathcal{B}_4 \supset \mathcal{B}_5 \supset \mathcal{B}_7$ (see Lemma 7.1). The matrices $G_{\mathcal{A}_i}$ and $\tilde{G}_{\mathcal{A}_i}$ generate the corresponding auxiliary codes $\mathcal{A}_i$, where as an effect of Construction XX, the matrices $\tilde{G}_{\mathcal{A}_i}$ do not have full rank.

7.3 IR-ARQ Protocols, Error Detection Mechanisms and their Performance Analysis

The transmission stages of an IR-ARQ scheme with M maximum transmissions is illustrated in Fig. 7.2. For the first transmission, a codeword of $C^{(1)}$ is sent, and for the ith transmission, $2 \leq i \leq M$, parity checks $p^{(i)}$ are transmitted such that the overall concatenated codeword is an element of $C^{(i)}$. An IR-ARQ scheme requires a mechanism of error detection so that either ACK or NACK can be used to provide feedback to the transmitter. An ACK signal is fed back to the transmitter to indicate a successful decoding and no further parity-checks are required, whereas a NACK signal is sent to request for transmission of more parity-checks. Three error detection mechanisms are considered in this chapter; these are error detection based on a CRC, two successive outputs of an error correction decoder, and confidence levels of the soft decision outputs of an error correction decoder.

In this section, three approaches to error detection are described and analysed. For the convenience of analysis, it is assumed that the channel is perturbed by additive white Gaussian noise and that binary antipodal signalling, which maps the coordinates of $c^{(i)} = (c_0^{(i)}, c_1^{(i)}, \ldots, c_{n_i-1}^{(i)}) \in C^{(i)}$ to the real coordinate space $\mathbb{R}^{n_i}$, is employed. This mapping function is defined as $\psi(c^{(i)}) = 2c^{(i)} - 1$.

7.3.1 Error Detection based on Cyclic-Redundancy-Check

The cyclic-redundancy-check (CRC) is the simplest and the most commonly used mechanism for error detection in IR-ARQ schemes. The block diagram of an IR-ARQ scheme employing CRC is shown in Figure 7.3. For an $(k-m)$ bit CRC, the IR-ARQ protocol, in relation to Figure 7.2 can be described as follows:

1. An m bit user message, u, is encoded by a CRC encoder to produce a codeword $x = (u|z)$ of length k bits. This k-tuple is then encoded by the forward error correction (FEC) code $C^{(M)}$ to produce $c = (u|z|p^{(1)}|p^{(2)}|\ldots|p^{(M)})$.

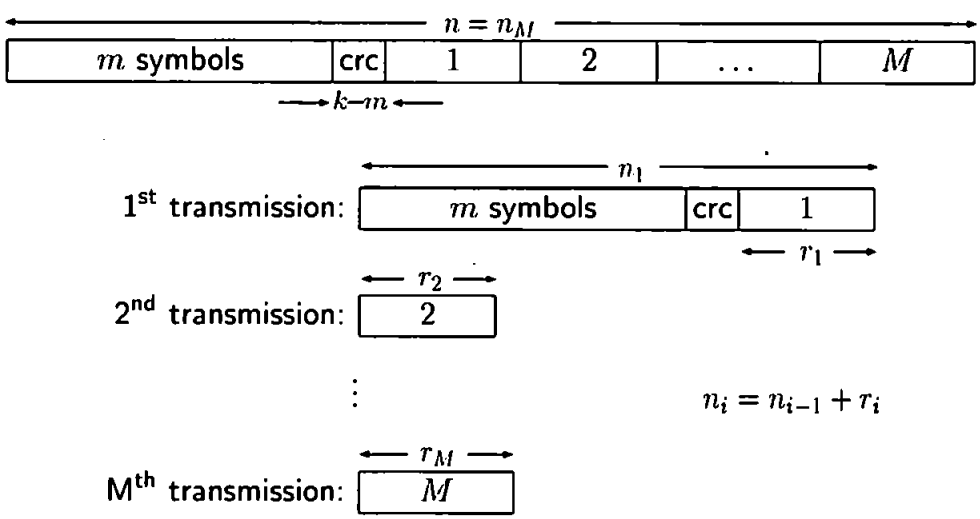


Figure 7.2: Transmission stages of a typical incremental redundancy ARQ scheme

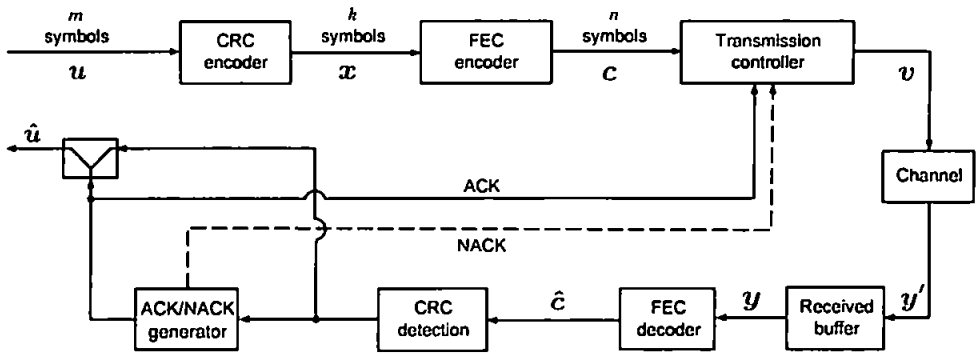


Figure 7.3: Incremental redundancy system employing CRC

2. The codeword $c^{(1)} = (u|z|p^{(1)}) \in C^{(1)}$ is transmitted and received as $y = y^{(1)}$ at the input to the receiver. The vector $y^{(1)}$ is decoded as $C^{(1)}$ using the FEC decoder and $\hat{c}^{(1)}$ is obtained at the end of decoding.

3. If $\hat{c}^{(1)}$ passes the CRC check, an ACK is sent to the transmitter to indicate successful transmission and the process stops here; otherwise $y^{(1)}$ is kept in the buffer and a NACK signal is transmitted to request for additional parity-check bits.

4. In the case of NACK, the transmitter sends parity-check $p^{(2)}$ of length r_2 and received as y'. At the receiving end, the vector $y^{(2)} = (y^{(1)}|y')$ is decoded as $C^{(2)}$ to produce $\hat{c}^{(2)}$.

5. If $\hat{c}^{(2)}$ passes the CRC check, $\hat{c}^{(2)}$ is accepted as the correct codeword and an ACK is sent to the transmitter; otherwise $y^{(2)}$ is kept in the buffer and a NACK signal is transmitted requesting for more parity-check bits. This process then continues until either the CRC is satisfied or the maximum number of transmissions per message M is reached.

The length of the CRC provides a trade-off between throughput and the error probability. A short CRC increases throughput, but the undetected error probability increases and results in an early error-floor which dominates the frame-error-rate (FER) of the system. Let $\mathcal{D}$ be the CRC code and assume that $d' \in \mathcal{D}$ is the correct message. Then the probability of undetected error of CRC is

$$Pr(\gamma_{ue}|d) = \sum_{d'' \in \mathcal{D},\ d'' \neq d'} Pr(d''|d), \tag{7.3}$$

where $Pr(d''|d)$ is the probability of d'' is output by the FEC decoder, given the input d. Using similar arguments, the probability of detected error is given by

$$Pr(\gamma_e|d) = \sum_{d'' \in \mathbb{F}_2^k,\ d'' \notin \mathcal{D}} Pr(d''|d). \tag{7.4}$$

7.3.2 Error Detection based on Two Successive FEC Decoding

In order to overcome the error-floor of CRC, consider a CRC-less approach that makes a feedback decision based on the output of two successive FEC decoding output. Note that since no CRC is

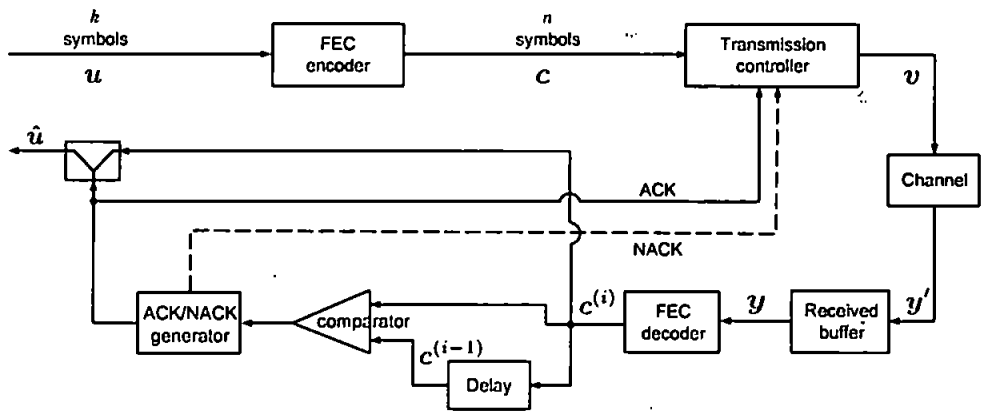


Figure 7.4: CRC-less incremental redundancy system, using output of two successive FEC decoding for error detection

employed, $m = k$. The block diagram of this CRC-less incremental redundancy scheme is shown in Figure 7.4. In relation to Figure 7.2, the IR-ARQ protocol using this CRC-less error detection approach can be described as follows:

1. A k bit user message, u, is encoded by an FEC encoder of the code $C^{(M)}$ to produce a codeword $c = (u|p^{(1)}|p^{(2)}|\ldots|p^{(M)})$.

2. The codeword $c^{(1)} = (u|p^{(1)}) \in C^{(1)}$ is transmitted and received as $y = y^{(1)}$ at the input to the receiver. The vector $y^{(1)}$ is decoded by an FEC decoder of the code $C^{(1)}$ to produce $\hat{c}^{(1)}$ and the corresponding information block is $\hat{u}^{(1)}$. Both $y^{(1)}$ and $\hat{u}^{(1)}$ are kept in the receiver's buffer and a NACK is sent to the transmitter.

3. The transmitter sends parity-check $p^{(2)}$ of length r_2 and it is received as y' at the receiving end. Using a concatenation of received vector $y^{(2)} = (y^{(1)}|y')$, the FEC decoder outputs a codeword $\hat{c}^{(2)} \in C^{(2)}$ which corresponds to an information block $\hat{u}^{(2)}$. If $\hat{u}^{(1)} = \hat{u}^{(2)}$, $\hat{u}^{(1)}$ is accepted as the correct information block and an ACK is sent to the transmitter to indicate successful transmission; otherwise $y^{(2)}$ and $\hat{u}^{(2)}$ are kept in the receiver's buffer and a NACK signal is transmitted to request for additional parity-check bits.

4. In the case of NACK, the transmitter sends parity-check $p^{(3)}$ of length r_3 and received as y^* at the receiving end. The vector $y^{(3)} = (y^{(2)}|y^*)$ is decoded as $C^{(3)}$ to produce $\hat{c}^{(3)}$ with the corresponding information block $\hat{u}^{(3)}$.

5. If $\hat{u}^{(2)} = \hat{u}^{(3)}$, $\hat{u}^{(2)}$ is accepted as the correct information block and an ACK is sent to the transmitter; otherwise both $y^{(3)}$ and $\hat{u}^{(3)}$ are kept in the buffer and a NACK signal is transmitted. This process then continues until either $\hat{u}^{(i-1)} = \hat{u}^{(i)}$, for $4 \leq i \leq M$, or the maximum number of transmissions per message M is reached.

Assuming that a maximum-likelihood (ML) list decoder is employed at the receiving, let the event that this ML decoder outputs a codeword $\hat{c}^{(i)}$, $\hat{c}^{(i)} \neq c^{(i)}$, where $c^{(i)}$ is the transmitted codeword, at ith transmission be denoted by $\mathrm{ML}_i(\hat{c}^{(i)})$. Then the probability of undetected error of this

approach is given by

$$Pr(\gamma_{ue}) = Pr\left(\mathrm{ML}_{i+1}((\hat{c}^{(i)}|\hat{p}^{(i+1)}))\mid \mathrm{ML}_i(\hat{c}^{(i)})\right), \tag{7.5}$$

and the probability of detected error is simply

$$Pr(\gamma_e) = \sum_{\substack{\bar{c}^{(i+1)}=(\bar{c}^{(i)}|\bar{p}^{(i+1)})\in C^{(i+1)}\\ \bar{c}^{(i)}\neq\hat{c}^{(i)}}} Pr\left(\mathrm{ML}_{i+1}(\bar{c}^{(i+1)})\mid \mathrm{ML}_i(\hat{c}^{(i)})\right). \tag{7.6}$$

7.1 *Definition (Cross-correlation).* The cross-correlation between a received vector y and a codeword c is defined as

$$\mathcal{X}(y,\psi(c)) = \sum_{j=0}^{n_i-1}(2c_j-1)y_j.$$

7.2 *Definition (Cross-correlation difference).* Given a received vector y, the cross-correlation difference between two codewords, c and c', is defined as

$$\delta(c,c') = \mathcal{X}(y,\psi(c)) - \mathcal{X}(y,\psi(c')).$$

Let $\hat{c}^{(i)}$ and $\bar{c}^{(i)}$ be the most likely and the next most likely codewords respectively in the list output by the ML list decoder at the ith transmission. Following Definitions 7.1 and 7.2, the cross-correlation difference between these two most likely codewords is given by

$$\begin{aligned}\delta(\hat{c}^{(i)},\bar{c}^{(i)}) &= \mathcal{X}(y^{(i)},\psi(\hat{c}^{(i)})) - \mathcal{X}(y^{(i)},\psi(\bar{c}^{(i)}))\\ &= 2\sum_{j=0}^{n_i-1} y_j^{(i)}(\hat{c}_j^{(i)}-\bar{c}_j^{(i)})\\ &= 2\left(\sum_{j\in S_{\hat{c}}} y_j^{(i)} - \sum_{j\in S_{\bar{c}}} y_j^{(i)}\right),\end{aligned} \tag{7.7}$$

where $S_{\hat{c}} = \{\sup(\hat{c}^{(i)} \ominus \bar{c}^{(i)}) \cap \sup(\hat{c})\}$, $S_{\bar{c}} = \{\sup(\hat{c}^{(i)} \ominus \bar{c}^{(i)}) \cap \sup(\bar{c})\}$, and $\sup(c) = \{i : c_i \neq 0,\ 0 \leq i \leq n-1\}$, the support of codeword c. Now, at the $(i+1)$th transmission, suppose that $(\hat{c}^{(i)}|\hat{p}^{(i+1)}) \in C^{(i+1)}$ is the most likely codeword and $(\bar{c}^{(i)}|\bar{p}^{(i+1)}) \in C^{(i+1)}$ is the next most likely codeword. Then the probability of undetected error is related to $\delta(\hat{c}^{(i)},\bar{c}^{(i)})$ and $\delta(\hat{p}^{(i+1)},\bar{p}^{(i+1)})$. At the ith transmission,

$$\mathcal{X}(y^{(i)},\psi(\hat{c}^{(i)})) > \mathcal{X}(y^{(i)},\psi(\bar{c}^{(i)}))$$

and at the $(i+1)$th transmission,

$$\mathcal{X}(y^{(i)},\psi(\hat{c}^{(i)})) + \mathcal{X}(y',\psi(\hat{p}^{(i+1)})) > \mathcal{X}(y^{(i)},\psi(\bar{c}^{(i)})) + \mathcal{X}(y',\psi(\bar{p}^{(i+1)})),$$

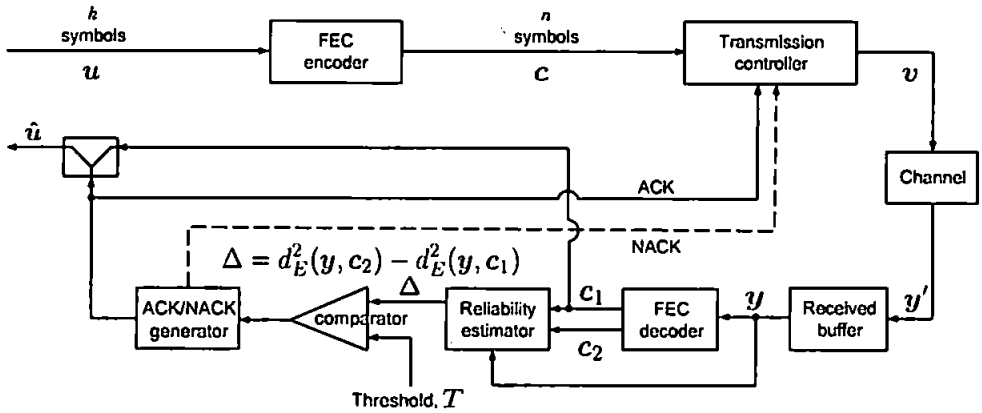


Figure 7.5: CRC-less incremental redundancy system, using confidence of FEC output for error detection

where $y^{(i+1)} = (y^{(i)}|y')$, which means $\delta(\bar{p}^{(i+1)}, \hat{p}^{(i+1)}) < \delta(\hat{c}^{(i)}, \bar{c}^{(i)})$. Therefore, (7.5) can also be expressed as

$$Pr(\gamma_{uc}) = Pr\left(\sum_{j\in S_{\hat{p}}} y_j^{(i)} - \sum_{j\in S_{\bar{p}}} y_j^{(i)} > \sum_{j\in S_{\hat{p}}} y'_j - \sum_{j\in S_{\bar{p}}} y'_j\right), \tag{7.8}$$

where, as before, $S_{\hat{p}} = \{\sup(\hat{p}^{(i+1)} \oplus \bar{p}^{(i+1)}) \cap \sup(\hat{p}^{(i+1)}\}$ and $S_{\bar{p}} = \{\sup(\hat{p}^{(i+1)} \oplus \bar{p}^{(i+1)}) \cap \sup(\bar{p}^{(i+1)}\}$.

7.3.3 Error Detection based on the Confidence of FEC Output

Another approach to error detection without using CRC is based on the reliability of the output of an FEC decoder. Similar to the previous approach, $m = k$ and it is also assumed that an ML list decoder is employed.

7.3 ***Definition (Squared Euclidean distance).*** The squared Euclidean distance between the received vector $y^{(i)} \in \mathbb{R}^{n_i}$ and a codeword $c^{(i)} \in C^{(i)}$, is defined by

$$d_E^2(y^{(i)}, \psi(c^{(i)})) = \frac{1}{n_i}\sum_{j=0}^{n_i-1}(y_j^{(i)} - \psi(c_j^{(i)}))^2.$$

7.4 ***Definition (Confidence of FEC output).*** Given a received vector y and let c and c' be the most likely and the next most likely codewords respectively output by an ML list decoder, the confidence of FEC output is defined as

$$\Delta = d_E^2(y, \psi(c)) - d_E^2(y, \psi(c')).$$

Let $\Delta^{(i)}$ and $T_i \in \mathbb{R}$ be the FEC output confidence (see Definition 7.4) and a predefined threshold respectively, at the ith transmission. The block diagram of this scheme is depicted in Figure 7.5.

Based on Figure 7.2, the IR-ARQ protocol of this approach can be described as follows:

1. An k bit user message, u, is encoded by an FEC encoder of the code $C^{(M)}$ to produce $c = (u|p^{(1)}|p^{(2)}|\ldots|p^{(M)})$.

2. The codeword $c^{(1)} = (u|p^{(1)}) \in C^{(1)}$ is transmitted and received as $y = y^{(1)}$ at the input to the receiver. The vector $y^{(1)}$ is decoded as $C^{(1)}$ using the FEC decoder and $\hat{c}^{(1)}$ is obtained at the end of decoding.

3. If $|\Delta^{(1)}| > T_1$, $\hat{c}^{(1)}$ is accepted as the correct output and transmit an ACK to the transmitter; otherwise a NACK is transmitted to request for additional parity-check and $y^{(1)}$ is kept in the buffer.

4. In the case of NACK, the transmitter sends parity-check $p^{(2)}$ of length r_2 and received as y'. At the receiving end, the vector $y^{(2)} = (y^{(1)}|y')$ is decoded as $C^{(2)}$ to produce $\hat{c}^{(2)}$.

5. If $|\Delta^{(2)}| > T_2$, $\hat{c}^{(2)}$ is accepted as the correct codeword and an ACK is sent to the transmitter; otherwise $y^{(2)}$ is kept in the buffer and a NACK signal is transmitted requesting for more parity-check bits. This process then continues until either the condition $|\Delta^{(i)}| > T_i$, for $3 \leq i \leq M$, is satisfied or the maximum number of transmissions per message M is reached.

Clearly, the threshold T_i provides a trade-off between the throughput and the error probability. The higher the threshold T_i, the lower the probability of error and, consequently, the lower the throughput of the system.

Let $v_j = \psi(c_j)$, $\Delta^{(i)}$ may be expressed as

$$\begin{aligned}
\Delta^{(i)} = \Delta &= d_E^2(y^{(i)}, \psi(\hat{c}^{(i)})) - d_E^2(y^{(i)}, \psi(\bar{c}^{(i)})) \\
&= \frac{1}{n_i}\left\{\sum_{j=0}^{n_i-1}(y_j^{(i)} - \hat{v}_j^{(i)})^2 - (y_j^{(i)} - \bar{v}_j^{(i)})^2\right\} \\
&= \frac{1}{n_i}\left\{\sum_{j=0}^{n_i-1} 2y_j^{(i)}\bar{v}_j^{(i)} - 2y^{(i)}\hat{v}_j^{(i)}\right\} + \frac{1}{n_i}\left\{\sum_{j=0}^{n_i-1}\left(\hat{v}_j^{(i)}\right)^2 - \sum_{j=0}^{n_i-1}\left(\bar{v}_j^{(i)}\right)^2\right\} \\
&= \frac{2}{n_i}\sum_{j=0}^{n_i-1} y_j^{(i)}(\bar{v}_j^{(i)} - \hat{v}_j^{(i)})
\end{aligned}$$

because $\sum_{j=0}^{n_i-1}\left(\hat{v}^{(i)}\right)^2 = \sum_{j=0}^{n_i-1}\left(\bar{v}^{(i)}\right)^2 = n_i$.

Assume that $c^{(i)}$ is the correct codeword at the ith transmission. Since, for $0 \leq j \leq n_i - 1$,

$$y_j^{(i)} = v_j^{(i)} + \bar{n}_j^{(i)} \tag{7.9a}$$

and

$$\begin{aligned}
\hat{v}_j^{(i)} &= \psi(c_j^{(i)} + e_j^{(i)}) \\
&= v_j^{(i)} + 2e_j^{(i)}
\end{aligned} \tag{7.9b}$$

where $e^{(i)} \in C^{(i)}$ and $\bar{n}_j^{(i)}$ is the Gaussian noise at the jth position, $\Delta^{(i)}$ can also be written as

$$\begin{aligned}
\Delta^{(i)} &= \frac{2}{n_i}\sum_{j=0}^{n_i-1}\left(v_j^{(i)}+\bar{n}_j^{(i)}\right)\left(\bar{v}_j^{(i)}-v_j^{(i)}-2e_j^{(i)}\right) \\
&= \frac{2}{n_i}\sum_{j=0}^{n_i-1}\left(v_j^{(i)}\bar{v}_j^{(i)}-\left(v_j^{(i)}\right)^2+\bar{n}_j^{(i)}\left(\bar{v}_j^{(i)}-v_j^{(i)}\right)-2e_j^{(i)}\left(v^{(i)}+\bar{n}_j^{(i)}\right)\right) \\
&= \frac{2}{n_i}\left\{\left(\sum_{j=0}^{n_i-1}v_j^{(i)}\bar{v}_j^{(i)}\right)-n_i+\left(\sum_{j=0}^{n_i-1}\bar{n}_j^{(i)}\left(\bar{v}_j^{(i)}-v_j^{(i)}\right)\right)\right\}- \\
&\quad \frac{2}{n_i}\left\{\left(\sum_{j=0}^{n_i-1}2e_j^{(i)}\left(v_j^{(i)}+\bar{n}_j^{(i)}\right)\right)\right\}
\end{aligned} \tag{7.10}$$

Let $\bar{e}^{(i)} = c^{(i)} \oplus \bar{c}^{(i)}$ and $w_i = |\sup(\bar{e}^{(i)})|$, equation (7.10) becomes

$$\begin{aligned}
\Delta^{(i)} &= \frac{2}{n_i}\left\{\left(\sum_{j:c_j^{(i)}=\bar{c}_j^{(i)}}1\right)+\left(\sum_{j:c_j^{(i)}\neq\bar{c}_j^{(i)}}(-1)\right)-n_i+\left(\sum_{j:c_j^{(i)}\neq\bar{c}_j^{(i)}}\pm 2\bar{n}_j^{(i)}\right)\right\} \\
&\quad -\frac{2}{n_i}\left\{2\sum_{j:c_j^{(i)}\neq\bar{c}_j^{(i)}}\left(v_j^{(i)}+\bar{n}_j^{(i)}\right)\right\} \\
&= -\frac{4}{n_i}\left(w_i-\sum_{j\in\sup(\bar{e}^{(i)})}\pm\bar{n}_j^{(i)}\right)-\frac{4}{n_i}\left(\sum_{j\in\sup(e^{(i)})}\left(v_j^{(i)}+\bar{n}_j^{(i)}\right)\right)
\end{aligned} \tag{7.11}$$

and by definition, $\Delta^{(i)} < 0$.

Without loss of generality, assume that $c^{(i)} = (0)^{n_i}$ (all zeros codeword of length n_i) and $c^{(i)} = \hat{c}^{(i)}$, $e^{(i)} = (0)^{n_i}$ and (7.11) simplifies to

$$\Delta_c^{(i)} = -\frac{4}{n_i}\left(w_i-\sum_{j\in\sup(\bar{e}^{(i)})}\bar{n}_j^{(i)}\right). \tag{7.12}$$

On the other hand, if $\hat{c}^{(i)} \neq c^{(i)}$, equation (7.11) now becomes

$$\begin{aligned}
\Delta_c^{(i)} &= -\frac{4}{n_i}\left(w_i-\sum_{j\in\sup(\bar{e}^{(i)})}\bar{n}_j^{(i)}\right)+\frac{4}{n_i}\left(\sum_{j\in\sup(e^{(i)})}\left(1-\bar{n}_j^{(i)}\right)\right) \\
&= \underbrace{-\frac{4}{n_i}\left(w_i-\sum_{j\in\sup(\bar{e}^{(i)})}\bar{n}_j^{(i)}\right)}_{\Delta_1}+\underbrace{\frac{4}{n_i}\left(\hat{w}_i-\sum_{j\in\sup(e^{(i)})}\bar{n}_j^{(i)}\right)}_{\Delta_2}
\end{aligned} \tag{7.13}$$

where $\hat{w}_i = |\sup(e^{(i)})|$.

7.2 Lemma. The first term of (7.13),

$$\Delta_1 = -\frac{4}{n_i}\left(w_i - \sum_{j\in\sup(\bar{c}^{(i)})} \bar{n}_j^{(i)}\right) \geq 0.$$

Proof. In the case where $\bar{c}^{(i)} = c^{(i)}$, it is obvious that $\Delta_1 = 0$.

For other case, $\bar{c}^{(i)} \neq c^{(i)}$ and $\bar{c}^{(i)} \neq \hat{c}^{(i)}$, by definition it is known that

$$d_E^2(y^{(i)}, \hat{v}^{(i)}) < d_E^2(y^{(i)}, \bar{v}^{(i)}) < d_E^2(y^{(i)}, v^{(i)}).$$

Following Definition 7.3 and (7.9a), it can be written that

$$\frac{1}{n_i}\sum_{j=0}^{n_i-1}\left(y_j^{(i)} - \bar{v}_j^{(i)}\right)^2 < \frac{1}{n_i}\sum_{j=0}^{n_i-1}\left(y_j^{(i)} - v_j^{(i)}\right)^2$$

$$\sum_{j=0}^{n_i-1}\left(v_j^{(i)} + \bar{n}_j^{(i)} - \bar{v}_j^{(i)}\right)^2 < \sum_{j=0}^{n_i-1}\left(\bar{n}_j^{(i)}\right)^2$$

$$\sum_{j=0}^{n_i-1}\left\{\left(v_j^{(i)}\right)^2 + 2v_j^{(i)}\bar{n}_j^{(i)} + \left(\bar{n}_j^{(i)}\right)^2 - 2\bar{v}_j^{(i)}v_j^{(i)} - 2\bar{v}_j^{(i)}\bar{n}_j^{(i)} + \left(\bar{v}_j^{(i)}\right)^2\right\} < \sum_{j=0}^{n_i-1}\left(\bar{n}_j^{(i)}\right)^2$$

Since the assumption $v_j^{(i)} = -1$ for $0 \leq j \leq n_i - 1$ is made,

$$\sum_{j=0}^{n_i-1}\left(\bar{n}_j^{(i)}\right)^2 + \sum_{j=0}^{n_i-1}\left(2 - 2\bar{n}_j^{(i)} + 2\bar{v}_j^{(i)} - 2\bar{v}_j^{(i)}\bar{n}_j^{(i)}\right) < \sum_{j=0}^{n_i-1}\left(\bar{n}_j^{(i)}\right)^2$$

$$\sum_{j=0}^{n_i-1}\left(2 - 2\bar{n}_j^{(i)} + 2\bar{v}_j^{(i)} - 2\bar{v}_j^{(i)}\bar{n}_j^{(i)}\right) < 0$$

By rearranging the above equation, it follows that

$$\sum_{j=0}^{n_i-1}\left(2 + 2\bar{v}_j^{(i)}\right) < \sum_{j=0}^{n_i-1}\left(2\bar{n}_j^{(i)} + 2\bar{v}_j^{(i)}\bar{n}_j^{(i)}\right)$$

$$n_i + \sum_{j=0}^{n_i-1}\bar{v}_j^{(i)} < \sum_{j=0}^{n_i-1}\bar{n}_j^{(i)}\left(1 + \bar{v}_j^{(i)}\right)$$

$$n_i + (2w_i - n_i) < \sum_{j=0}^{n_i-1}\bar{n}_j^{(i)}\left(1 + \bar{v}_j^{(i)}\right)$$

$$w_i < \sum_{j\in\sup(\bar{c}^{(i)})}\bar{n}_j^{(i)},$$

and hence, $\Delta_1 > 0$. ∎

7.3 ***Corollary.*** The second term of (7.13),

$$\Delta_2 = \frac{4}{n_i}\left(\hat{w}_i - \sum_{j\in\sup(c^{(i)})} \bar{n}_j^{(i)}\right) < 0$$

and

$$|\Delta_2| > |\Delta_1|.$$

Proof. The proof follows from Lemma 7.2 and $\Delta_c^{(i)} < 0$. ∎

From Equations (7.12) and (7.13), Lemma 7.2 and Corollary 7.3, it is possible to characterise the behaviour of $\Delta^{(i)}$. It can be said that

- for the case $\hat{c}^{(i)} = c^{(i)}$, decoder produces a correct codeword
  - the noise components in (7.12) at the support of the next most likely codeword $\bar{c}^{(i)}$ are not that severe and as such $w_i > \sum_{j\in\sup(\bar{c}^{(i)})} \bar{n}_j^{(i)}$;
  - consider (7.12), $\delta = |\bar{n}_j^{(i)}| < 0.5$ on average, hence

$$\begin{aligned}\delta_c &= w_i - \sum_{j\in\sup(\bar{c}^{(i)})} \bar{n}_j^{(i)} = w_i(1-\delta)\\ &> \frac{w_i}{2}.\end{aligned}$$

- for the case $\hat{c}^{(i)} \neq c^{(i)}$, decoder produces an incorrect codeword
  - the magnitude of the noise components in (7.13) at the support of the most likely candidate codewords $\bar{c}^{(i)}$ is relatively large as such $w_i < \sum_{j\in\sup(\bar{c}^{(i)})} \bar{n}_j^{(i)}$ and $\hat{w}_i < \sum_{j\in\sup(c^{(i)})} \bar{n}_j^{(i)}$;
  - consider Δ_2 of (7.13), $\delta' = |\bar{n}_j^{(i)}| > 0.5$ on average, hence

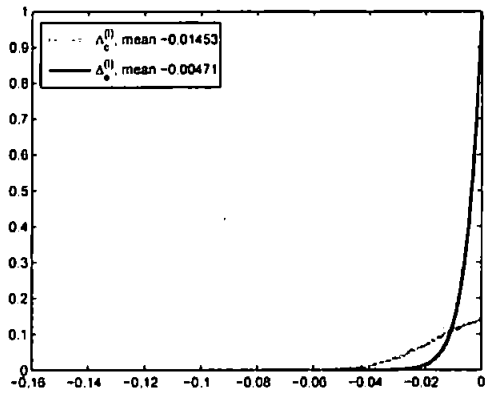
$$\begin{aligned}\delta_c &= \hat{w}_i - \sum_{j\in\sup(c^{(i)})} \bar{n}_j^{(i)} = \hat{w}_i(1-\delta')\\ &< \frac{\hat{w}_i}{2}\end{aligned}$$

  and the same arguments can be applied to Δ_1 of (7.13);

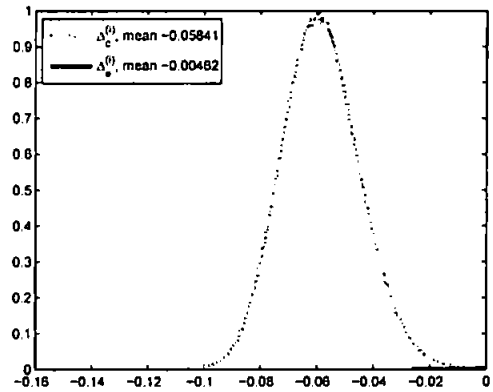
- since on average $|\delta_c| > |\delta_c|^*$, $|\Delta_c^{(i)}| > |\Delta_c^{(i)}|$ and this claim is supported by Figure 7.6.

Figure 7.6 shows the distribution of $\Delta_c^{(i)}$ and $\Delta_c^{(i)}$, under the ordered-reliability soft-decision list decoding using Dorsch's (1974) algorithm, see also Section 6.3, for the extended BCH code [128, 113, 6] as well as the lengthened code [136, 113, 8] at E_b/N_o 2.0 dB and 5.5 dB. This figure clearly shows that as the length of the code or the E_b/N_o is increased, the average value of $|\Delta_c^{(i)}|$ increases; on the contrary, the average value of $|\Delta_c^{(i)}|$ stays on a relatively the same value regardless of the increase

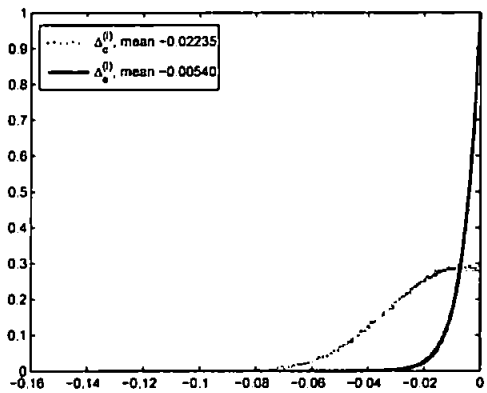
*This difference is even large in the case where $\Delta_1 > 0$.



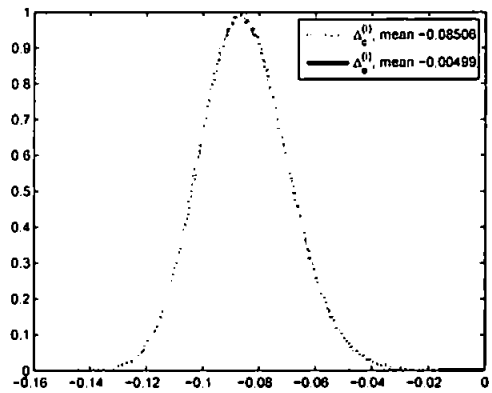
(a) E_b/N_o 2.0 dB, [128, 113, 6]



(b) E_b/N_o 5.5 dB, [128, 113, 6]



(c) E_b/N_o 2.0 dB, [136, 113, 8]



(d) E_b/N_o 5.5 dB, [136, 113, 8]

Figure 7.6: Comparison of $\Delta_c^{(i)}$ and $\Delta_e^{(i)}$ under soft-decision list decoding. They are normalised so that the highest peak between $\Delta_c^{(i)}$ and $\Delta_e^{(i)}$ is 1.0.

of code length and/or the E_b/N_o. It is this characteristic that led to the use of a threshold at the ith transmission,

$$T_i = \frac{4}{n_i}\kappa, \tag{7.14}$$

where $\kappa \in \mathbb{R}$, for error detection. This error detection mechanism produces a

$$\text{decision} = \begin{cases} \text{no error detected} & \text{if } |\Delta^{(i)}| > T_i, \\ \text{error detected} & \text{otherwise.} \end{cases}$$

The probability of undetected error, $Pr(\gamma_{ue})$, and the probability of detected error, $Pr(\gamma_e)$, are respectively given by

$$Pr(\gamma_{ue}) = Pr\left(\left(\sum_{j\in\sup(e^{(i)})} \bar{n}_j^{(i)} - \sum_{j\in\sup(\bar{e}^{(i)})} \bar{n}_j^{(i)}\right) - (\hat{w}_i - w_i) > \kappa\right) \tag{7.15}$$

and

$$Pr(\gamma_e) = Pr\left(\left(\sum_{j\in\sup(e^{(i)})} \bar{n}_j^{(i)} - \sum_{j\in\sup(\bar{e}^{(i)})} \bar{n}_j^{(i)}\right) - (\hat{w}_i - w_i) \leq \kappa\right). \tag{7.16}$$

7.4 Numerical Results

Computer simulations of the CRC and CRC-less IR-ARQ schemes using sequence of codes given in Examples 7.1 and 7.2 have been carried using the incremental correlation Dorsch decoder, see Section 6.3, with 10^6 codewords as the list decoder with hard and soft decision outputs and which has quasi ML performance. For code sequences in Example 7.2, all three approaches to error detection described in Section 7.3 were simulated, whereas for the code sequences in Example 7.1, only the CRC approach was simulated. An 8 bit CRC polynomial $(1+x)(1+x^2+x^5+x^6+x^7)$ was used for the CRC case. The protocol assumes that an ACK is transmitted if no error is detected or a maximum number of transmission has been reached; and a NACK is transmitted otherwise. Fig. 7.7 and 7.8 show the FER performance and the throughput respectively of the simulated IR-ARQ schemes. The CRC approach shows good throughput, but exhibits an early error-floor of the FER, which is caused by undetected error events. Comparing the FER performance and the throughput of the sequence of codes given in Examples 7.1 and 7.2, the sequence of codes of Example 7.1 has lower error probability and higher throughput than that of Example 7.2 at low SNR. This can be explained by comparing the sequences of codes in the two examples; the code $[196, 113, 14]$ in Example 7.2 is weaker in terms of minimum distance than the code $[202, 113, 16]$ in Example 7.1. However, the advantage of having higher-rate codes and also more codes in the sequence becomes evident in the middle SNR region of both figures.

In comparison to the CRC approach, the first CRC-less approach, indicated by Method B in the figures, can only provide gain in the high SNR region and is relatively useless at low SNR. On the other hand, the second CRC-less approach, indicated by Method C in the figures, provides an attractive scheme to trade-off FER performance against throughput. The confidence level factor κ may be set to achieve the required FER and throughput.

7.5 Adding CRC to CRC-less Error Detection Approach

It has been shown graphically that the CRC-less approach, Method C, which compares the confidence of FEC decoder's output with some predefined threshold or confidence level exhibits good performance and attractive throughput. In this approach, the FEC decoder produces a list of L most-likely codewords ranked based on the their Euclidean distance to the received vector. The codeword which is closest to the received vector appears on the top of the list. An undetected error

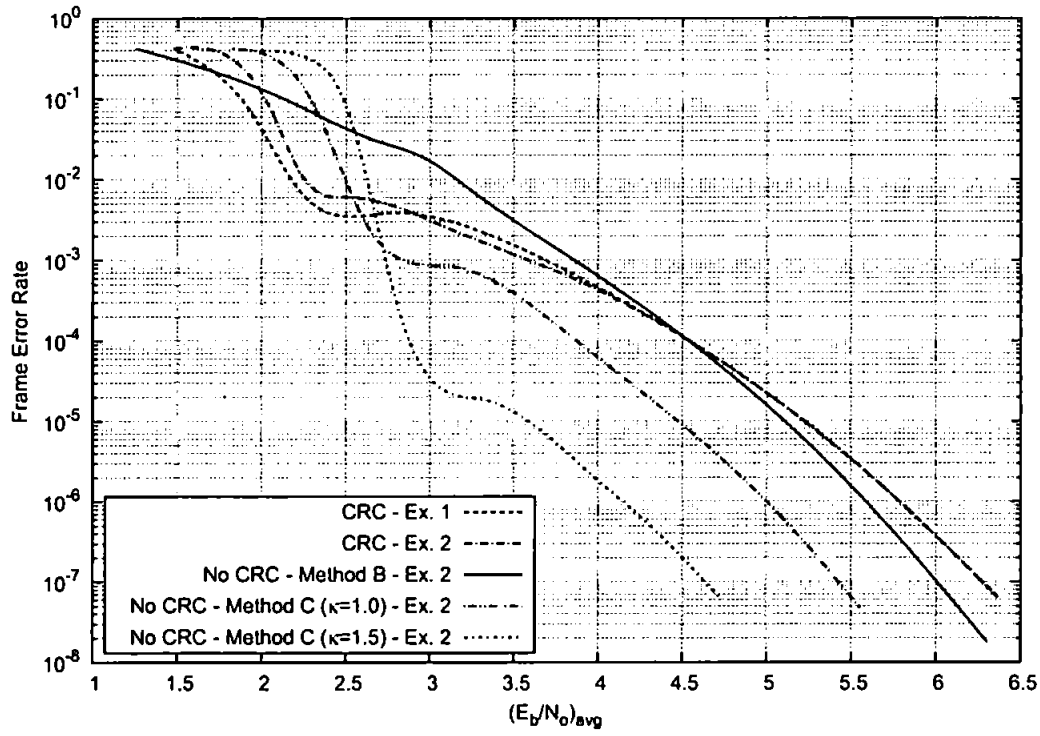


Figure 7.7: FER performance of the IR-ARQ scheme based on extended BCH codes of length 128

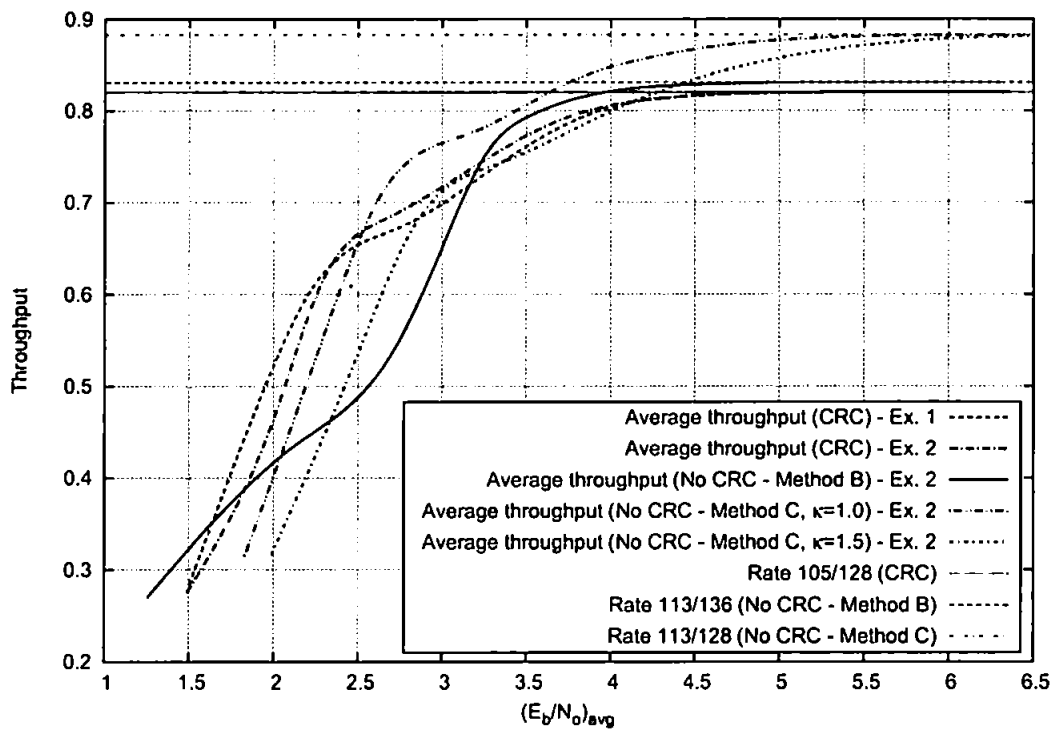


Figure 7.8: Average throughput of the IR-ARQ scheme based on extended BCH codes of length 128

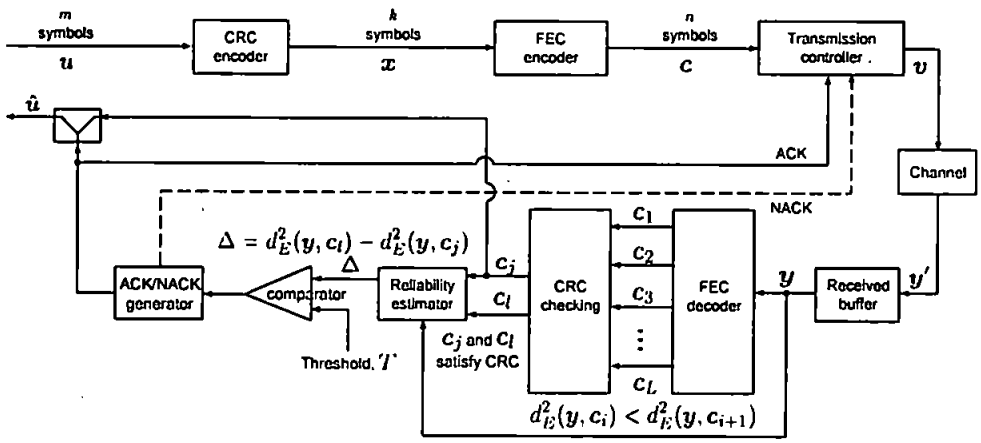


Figure 7.9: Incremental redundancy system, using combined CRC and the confidence of FEC output for error detection

occurs if the closest codeword is not equal to the transmitted codeword; and the first two codewords are not so close to the received vector in the Euclidean space such that their difference in squared Euclidean distance is greater than the predefined threshold. Clearly, one way to improve this is to simply increase the threshold. However, this may result in many correct decisions being discarded. An alternative approach, which can avoid this situation, is to combine CRC with Method C. With CRC, the FEC decoder still produces L most-likely codewords, but before their Euclidean distance is compared, those that do not satisfy the CRC will be discarded. This, in effect, leaves us with L' most-likely and CRC-satisfied codewords, where $L' \leq L$, to process. Assuming that these L' codewords are ranked such that the one closest to the received vector comes first, if the difference in squared Euclidean distance between the first two codewords with respect to the received vector is greater than the threshold, the first codeword is declared correct and an ACK is transmitted; otherwise additional parity checks are requested. The pictorial representation of this approach is illustrated in Figure 7.9, where Δ denotes the confidence level or threshold.

The FER and throughput performance of this combined approach look encouraging, as shown in Figures 7.10 and 7.11 respectively. As expected, the overall FER performance is superior than that of either CRC only or Method C for the same confidence level factor. It has been shown that the throughput of a CRC only system is limited due to the high probability of undetected error. By combining the error detection with Method C, an increased in throughput is evident from Figure 7.11. Due to rate loss attributed to CRC, as can be anticipated, the throughput of the combined system is lower than that of Method C in the higher E_b/N_o region. These results are obtained from computer simulations using code sequences given in Example 7.2 and an 8 bit CRC polynomial $(1+x)(1+x^2+x^5+x^6+x^7)$.

7.6 Summary

- Chain of cyclic codes in conjunction with Constructions X and XX can be use to construct a sequence of linear codes for incremental redundancy two-way communications scheme. Using this construction, a sequence of codes with increasing length and minimum distance is

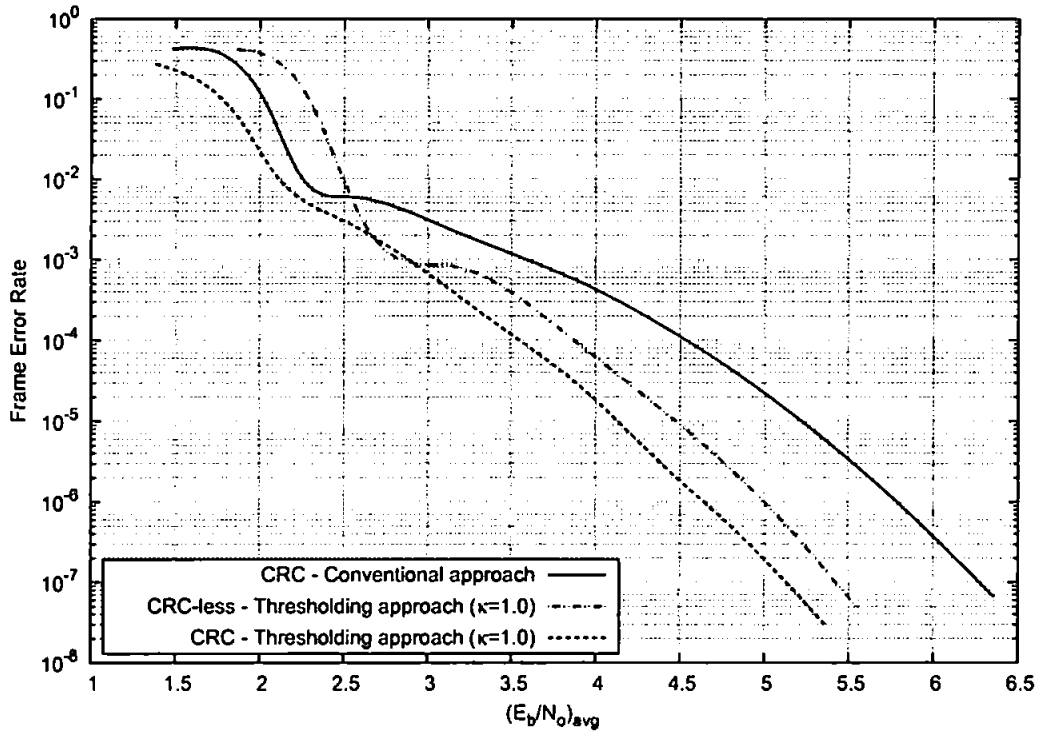


Figure 7.10: FER performance of the approach using CRC combined with confidence of FEC output

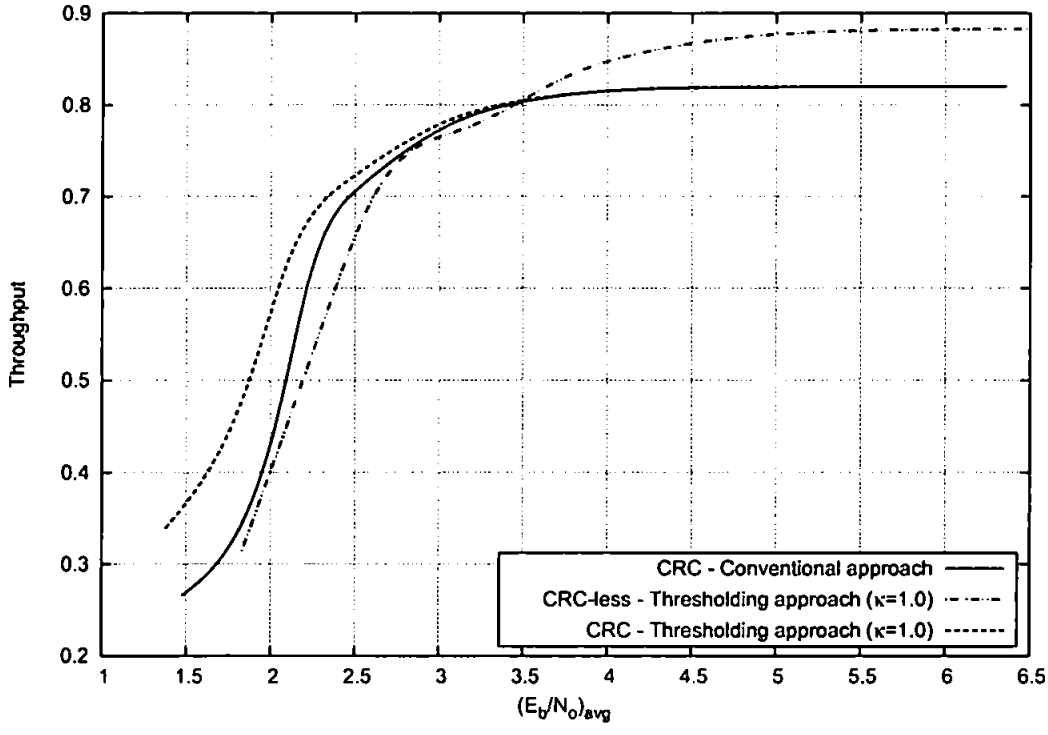


Figure 7.11: Average throughput of the approach using CRC combined with confidence of FEC output

obtained.

- Conventional incremental redundancy scheme uses CRC as a mechanism to decide whether to feed an ACK or NACK back to the transmitter. Two new CRC-less approaches and a hybrid approach to error detection are introduced:

  1. The first CRC-less approach does error detection based on the output of two successive FEC outputs. If two successive FEC outputs return the same frame, a correct codeword is declared.
  2. The second CRC-less approach introduces a confidence level to the FEC output. If the difference of the squared Euclidean distance, with respect to received vector, of two most-likely codewords is greater than the confidence level, a correct codeword is declared. This approach to error detection, by adjusting the confidence level factor, provides a way to trade-off error-rate and throughput of an incremental redundancy communication scheme.
  3. In the hybrid approach, error detection using CRC is combined with the second CRC-less approach (Method C). This combined approach reduces the probability of undetected error of CRC and also that of the second CRC-less approach, and as a consequence, produces the best FER performance among the other error-detection approaches considered in this chapter.

Part V

Conclusions

8 Conclusions and Future Research Directions

Since Shannon's landmark paper in 1948, we have seen two main development streams of channel coding, namely probabilistic coding and classical coding. Probabilistic coding theory is the modern approach to channel coding which puts emphasis on low complexity decoding of long codes which are made up of simple component codes. On the other hand, the classical approach to channel coding focuses on the algebraic structure of codes and particularly, on the maximisation of the asymptotic coding gain. In this thesis, extensive studies have been carried out on these two approaches and various contributions to the field of coding have been made.

Motivated by the fact that the majority of research work on LDPC codes is aimed at long codes, two algebraic construction techniques for binary cyclic LDPC codes of short block length using the theory of cyclotomic cosets, idempotents and Mattson-Solomon polynomials have been developed. An attractive feature of these constructions is the incremental trade-off between the minimum Hamming distance and the sparseness of the parity-check matrix of an LDPC code. Starting with a set of idempotents for a corresponding code of a given minimum Hamming distance, one may include additional idempotents to increase the minimum Hamming distance. Since the sparseness of the parity-check matrix of the code is proportional to the number of idempotents in the set, adding more idempotents in effect reduces the sparseness the parity-check matrix. It is worth noting that these two construction techniques–although they work in different domains, produce the same LDPC code, and are complementary. The construction technique described in Section 2.2.1 is more useful to search for codes of certain minimum Hamming distance with girth of at least 6, whereas the one described in Section 2.2.2 is better suited for construction of cyclic LDPC codes (regardless of the girth) of given code rate and minimum Hamming distance. We may also regard the advantage of having complementary construction methods of different domains as analogous to that of representing a signal in time and frequency domains; one is able to gain a better insight to code construction from different perspective. It is interesting to note that the algebraic construction methods discussed in Chapter 2, in addition to producing some new cyclic LDPC codes, provide simple alternatives to generate codes which have equivalent parameters as the Euclidean and projective geometry codes. We have also shown that non binary cyclic LDPC codes may be easily constructed using the same concept.

While short block length cyclic LDPC codes have superior performance compared to irregular LDPC codes–such as codes constructed using the PEG algorithm, of the same block length and code rate, this is not true for longer block length cyclic LDPC codes. The algebraic structure of the cyclic LDPC codes–assuming that their girth is at least 6, guarantees increases in minimum Hamming distance of these codes as the block length is increased. For these algebraic codes, the minimum Hamming distance is proportional to the weight of the parity-check equations and thus the density

of the corresponding parity-check matrix. Consequently, the advantage of having relatively large minimum Hamming distance becomes counter-productive in the low SNR region where errors are dominant. Given these errors and a large number of participating symbols in each parity-check equation, it is difficult for these parity-check equations to agree on a solution causing convergent problem for the iterative decoder, and this shifts the threshold in the waterfall region for the frame error performance curve to a higher SNR value. However, for an irregular LDPC code of the same length and code rate which has a good variable node degree distribution, the parity-check matrix is still relatively sparse. As a result of this, convergence is not as much of a problem in the low SNR region. The advantage of having large minimum Hamming distance becomes apparent in the high SNR region. Irregular LDPC codes show signs of an error floor in their performance which is partly attributed to minimum Hamming distance error events. Whereas for algebraic LDPC codes, there is no such sign–at least this is true in the frame error region where simulations have been performed.

The minimum Hamming distance of a linear code is one of the important parameters, particularly in classical coding theory. This distance which dictates the number of errors a code can correct, is a benchmark for comparing codes of the same block length and code rate. In this work, the existing algorithms to compute the minimum Hamming distance of linear codes have been studied. Advances in computing technology has enabled the possibility to harness spare computing resources to create a grid computer. It has been shown in detail how to convert a minimum Hamming distance evaluation algorithm to one that can be deployed in the grid of computers by making use of one of the properties of the revolving-door combination generator. The advantage of this is apparent, it enables one to compute the minimum Hamming distance of a longer code that is otherwise impractical to compute, in a reasonable amount of time. A notable example is the evaluation of the previously unconfirmed minimum Hamming distance of the binary extended quadratic residue code for prime 199; a single-threaded evaluation would take months to complete, whereas using grid computing, the evaluation time reduces to several days. In addition to this extended quadratic residue code, grid computing has also enabled many binary cyclic codes to be found which have larger minimum Hamming distance than the previously considered best known linear code.

Several methods to lengthen (Constructions X and XX) codes and to shorten (Construction Y1) codes have been studied, and are aimed at producing codes of large minimum Hamming distance. It was found in this work that Constructions X and XX can be ineffective for a certain set of component codes and accordingly, a more effective strategy has been devised. This has enabled the construction of several codes which have larger minimum Hamming distance than the highest known. Overall, this work has resulted in as many as 901 codes to be determined which are better than the best known linear codes. Another important result that emerged from this work is a tabulation of the highest minimum Hamming distance attainable by all binary cyclic codes of odd lengths from 129 to 189. This table together with those of Chen (1969), Promhouse and Tavares (1978) and Schomaker and Wirtz (1992) serve as useful references for those who are interested in binary cyclic codes.

Double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8 have also been explored. Various choices of defining polynomial of a circulant matrix have been examined and the link to the self-duality of a code was established. Furthermore, simplifications to an algorithm to compute the minimum Hamming distance as well as an algorithm to compute the number codewords of a given Hamming weight for certain double-circulant codes have been established. A

complete proof that the projective special linear group fixes the automorphism group of the double-circulant codes based on primes congruent to ± 3 modulo 8, has been given. Using this in conjunction with the work of Mykkeltveit et al. (1972), the weight distributions of the double-circulant codes for primes congruent to ± 3 modulo 8 have been characterised. There are two useful applications of this result, it provides an independent verification of the number of codewords of a given Hamming weight that have been previously computed exhaustively; and it gives an accurate estimate of the minimum Hamming distance of prime-based double-circulant codes. For the first application, it has been shown that some of results reported in Gaborit et al. (2005) were incorrect; and the weight distributions of two inequivalent double-circulant codes have also been obtained and independently verified, the $[168, 84, 24]_2$ codes, and these were previously unknown. The second application has enabled the conjecture to be made that there are no extremal double-circulant self-dual codes of length longer than 136.

Soft-decision decoding algorithms have been addressed in this work. The optimal algorithm of Hartmann and Rudolph (1976) was initially investigated. The optimum solution of the Hartmann and Rudolph (1976) algorithm requires the complete set of codewords of the dual code and the results from this investigation have revealed that it is not possible to trade off complexity against suboptimum performance by taking a subset of all the dual code codewords. This investigation, however, did lead to an improved iterative decoder for LDPC codes–the codeword substitution belief propagation decoder, which substitutes a subset of the low weight parity check equations for those of higher weight at the beginning of each iteration replacing them at the end of each iteration. Although this modification produces near maximum likelihood performance for short cyclic LDPC codes, it has been found to be unsuitable for long LDPC codes.

For decoding general linear codes, the algorithm by Dorsch (1974) was studied. A detailed description of an incremental correlation approach to this algorithm was given in Chapter 6 as well as the derivation of the upper-bound on the complexity to achieve maximum likelihood performance. Various half-rate binary linear codes of large minimum Hamming distance were simulated and the results showed that the frame-error performance of these codes under the incremental correlation Dorsch decoder is very close to the Shannon's sphere packing lower bound constrained for binary transmission. In the author's opinion, with advances in semiconductor technology, the implementation of this type of decoder will herald a new era in channel coding which emphasises on the optimality of codes rather than the simplicity of decoding. Figure 8.1 plots the E_b/N_0 values of various LDPC and best known linear codes at which FER performance of 10^{-5} is achieved*. Plotting the sphere packing lower bound clearly shows the importance of designing codes for optimality as opposed to that for ease of decoding.

It was shown to be straightforward to build a soft-decision list decoder from of the incremental correlation Dorsch algorithm. The application of this type of soft-decision list decoder has created a new approach to the area of error detection, which so far has been dominated by error detecting codes such as a CRC used with hard-decision detector. Error detection can be realised by comparing the squared Euclidean distance of the two most likely outputs of the list decoder. This is another major contribution that emerged from this work. This idea has been applied to the incremental redundancy communication system and promising results–both in error rate performance and throughput, have been obtained. With this method, increases in throughput are possible as the

*Some of the points are obtained from the figures in Chapter 6

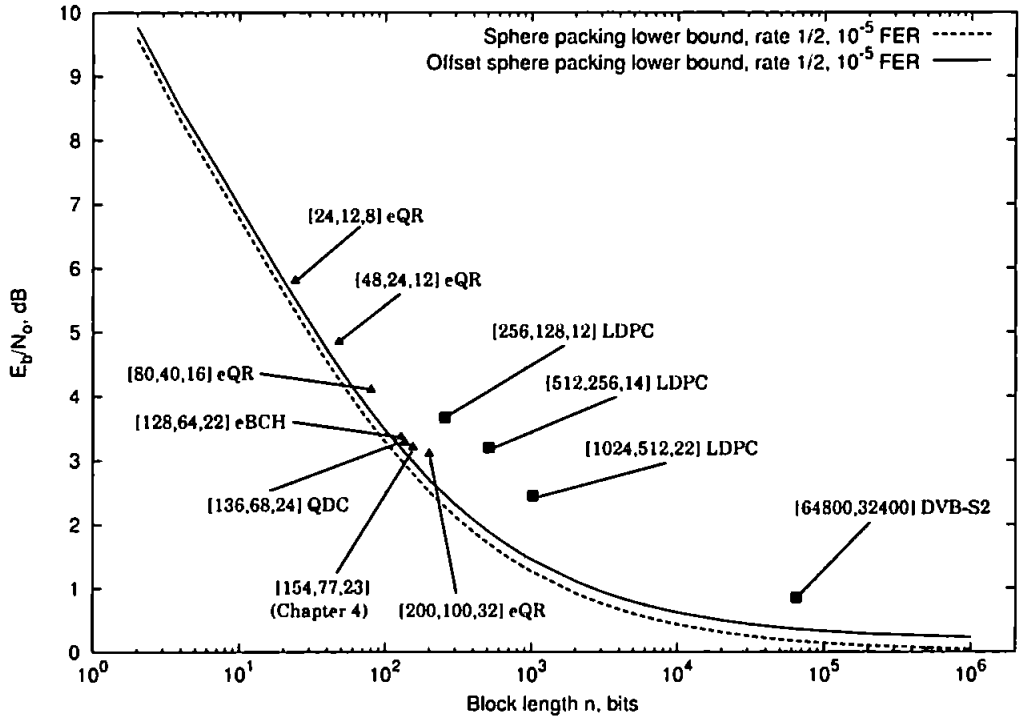


Figure 8.1: Distance to the sphere packing lower bound at 10^{-5} FER of rate 1/2 LDPC codes and best known linear codes

All of the LDPC codes, excluding the that of the DVB-S2, are constructed using the PEG algorithm. All LDPC codes are decoded using the belief propagation decoder with maximum iterations of 100, whereas the best known linear codes are decoded using the incremental correlation Dorsch decoder. Note that 4096, 3×10^6, 3×10^6, 10^7, 10^7, 10^7 and 3×10^7 correlations respectively have been employed for decoding the [24, 12, 8] extended quadratic residue (eQR), [48, 24, 12] eQR, [80, 40, 16] eQR, [128, 64, 22] extended BCH (eBCH), [136, 68, 24] quadratic double-circulant (QDC), [154, 77, 23] (see Chapter 4) and [200, 100, 32] eQR codes.

error detecting code is no longer mandatory. It is also interesting to note that for the case when an error detecting code is employed, it is still possible with the method to trade performance level off for a reduced throughput.

The novel idea of constructing a sequence of codes suitable for incremental redundancy communication has been introduced in this thesis. The sequence of codes can be obtained by recursively applying Construction X, Construction XX, or both constructions, to a chain of cyclic codes. Using these constructions, it is possible to guarantee the sequence of codes, which consists of codes of different block length, to have satisfactory minimum Hamming distances for each code in the sequence. This is not the case for the method of applying puncturing patterns to a low rate code which is the standard method used in incremental redundancy communication.

During the course of this work, several problems have arisen that have not been answered as well as new ideas which suggest several areas for further investigation. These are listed below and are proposed as potentially useful topics for future work.

- Given the existence of algebraic structure in the cyclic LDPC codes, the exact minimum Hamming distance of cyclic LDPC code is known in many cases. In depth mathematical studies

into the weight distributions as well as the automorphism group of these codes would be of interest. The knowledge of the weight distribution of a code would allow one to estimate the error floor from the union bound argument.

- The performance of irregular LDPC codes is limited by an error floor which is not necessarily caused by minimum Hamming weight error events. It has been observed from numerical simulations that for this type of error event, the number of failed parity-check equations oscillates with each iteration as do the number of symbol errors. The configuration of the bipartite graph of the LDPC code that causes this problem should be analysed and it is desirable to have a code construction method that can avoid problem configurations. Instead of tackling this problem from the code construction side, another possible strategy is to modify the iterative decoder to trap and break the oscillations in the number of unsatisfied parity-check equations. Alternatively, matched code-decoder design is another strategy worth investigation.

- The family of irregular LDPC codes which are constructed from a protograph by using circulant expansion i.e. the IRA family, has quasi cyclic structure, in addition to a zig-zag pattern in the parity-check matrix. It may be possible to exploit these constraints to determine the exact minimum Hamming distance of the code. An estimate or bound of the minimum Hamming distance would also be useful.

- Extending the table of the highest minimum Hamming distances attainable by binary cyclic codes of length longer than 189. Tables for cyclic codes over larger fields should also be considered.

- Searching for improvements to the best known linear codes is a useful area of research. Many best known linear codes have been derived from generalised concatenated codes, see Blokh and Zyablov (1974) and Zinov'ev (1976), whose minimum Hamming distance is explicit as long as the minimum Hamming distances of the constituent codes are known. This is mainly attributed to the limitation of single-threaded minimum Hamming distance computation algorithm. It is possible that other family of codes such as cyclic and algebraic geometry codes, may have larger minimum Hamming distances, but it is impractical to verify this distance using a single-threaded approach. However, grid computing now makes this approach a practical proposition.

- A great deal of effort has been devoted to rate $1/m$ quasi cyclic codes, for integers $m \geq 2$, in searching for good linear codes. Little has been devoted to the higher rate codes and it is likely that these codes will also produce linear codes of large minimum Hamming distance.

- Investigations to double-circulant codes of larger fields with the extension of the modular congruence approach to their weight distributions is also a useful area. Having seen the benefits that the modular congruence approach can provide, it would also be useful to consider the potential application of this approach for other family of codes. To be effective, the code considered should have large automorphism group.

- In order to allow the incremental correlation Dorsch algorithm to be used for longer codes, various strategies that reduce the number of required correlations should be investigated.

- In a sequence of codes constructed using either Construction X or XX, the code of longer length is less optimal that that of shorter length. Different construction approaches that maintain the code optimality as the code length is increased should be investigated.
- A simpler derivation of the probability of undetected error given by (7.15) as a function of SNR appears to be possible.

Part VI

Appendices

A Quasi-Cyclic LDPC Codes and Protograph

Despite irregular LDPC codes have lower error rate than the regular counterparts, see Luby et al. (2001), the extra complexity of the encoder and decoder hardware structure has made this class of LDPC codes less attractive from industry point of view. In order to encode an irregular code which has a parity-check matrix H, Gaussian elimination has to be done to transform this matrix into a reduced echelon form. Irregular LDPC codes, as shown in Section 2.3, may also be constructed by constraining the $n-k$ low degree variable vertices of the Tanner graph to form a zig-zag pattern, see Ping et al. (1999). Translating these $n-k$ variable vertices of the Tanner graph into the form of matrix, we have

$$H_p = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ & \vdots & \vdots & & \\ & & 1 & 1 & \\ & & & 1 & 1 \end{bmatrix}. \tag{A.1}$$

The matrix H_p is non singular and the columns of this matrix may be used as the coordinates of the parity-check symbols of an LDPC code.

The use of zig-zag parity does simplify the encoding complexity as the Gaussian elimination process is no longer necessary and the encoding, assuming that

$$H = [H_u | H_p]$$

$$= \begin{array}{c} \begin{array}{ccccccccc} v_0 & v_1 & \cdots & v_{k-2} & v_{k-1} & v_k & v_{k+1} & \cdots & v_{n-2} & v_{n-1} \end{array} \\ \left[\begin{array}{ccccc|ccccc} u_{0,0} & u_{0,1} & \cdots & u_{0,k-2} & u_{0,k-1} & 1 & & & & \\ u_{1,0} & u_{1,1} & \cdots & u_{1,k-2} & u_{1,k-1} & 1 & 1 & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & & \\ u_{n-k-2,0} & u_{n-k-2,1} & \cdots & u_{n-k-2,k-2} & u_{n-k-2,k-1} & & & 1 & 1 & \\ u_{n-k-1,0} & u_{n-k-1,1} & \cdots & u_{n-k-1,k-2} & u_{n-k-1,k-1} & & & & 1 & 1 \end{array}\right] \end{array},$$

can be performed as follows

$$v_k = \sum_{j=0}^{k-1} v_j u_{0,j} \pmod 2$$

$$v_i = v_{i-1} + \sum_{j=0}^{k-1} v_j u_{i-k,j} \pmod 2 \qquad \text{for } k+1 \le i \le n-1 .$$

Nevertheless, zig-zag parity does not give significant reduction in storage space as the matrix H_u still needs to be stored. By introducing some structure in H_u, say quasi-cyclic structure, the storage requirements may be considerably simplified.

A.1 Quasi-Cyclic LDPC Codes

Quasi-cyclic codes have the property that a codeword is a m-sized cyclic shift of any codeword, where m is an integer. The existence of this structure allows a simple feedback shift registers to be employed for efficient encoding of this type of codes. Refer to Chapter 5 (Definition 5.1) for the definition of a circulant matrix. A quasi-cyclic code can be built from concatenation of circulant matrices to define the generator or parity-check matrix.

Example A.1: A quasi-cyclic code with defining polynomials $r_1(x) = 1 + x + x^3$ and $r_2(x) = 1 + x^2 + x^5$, where both polynomials have maximum degree of 7, may have the parity-check matrix in the following form

Quasi-cyclic code

$$H = \left[\begin{array}{ccccccc|ccccccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}\right] .$$

A.1 ***Definition (Permutation Matrix).*** A permutation matrix is a type of circulant matrix where each row or column has weight of 1. A permutation matrix, which is denoted by $P_{m,j}$, has $r(x) = x^j \pmod{x^m - 1}$ as the defining polynomial and it satisfies the property that $P_{m,j}^2 = I_m$ where I_m is an $m \times m$ identity matrix.

Due to the sparseness of permutation matrix, they are usually used to construct quasi-cyclic LDPC codes. The resulting LDPC codes may have their parity-check matrix in the following form

$$H = \begin{bmatrix} P_{m,O_{0,0}} & P_{m,O_{0,1}} & \cdots & P_{m,O_{0,t-1}} \\ P_{m,O_{1,0}} & P_{m,O_{1,1}} & \cdots & P_{m,O_{1,t-1}} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m,O_{s-1,0}} & P_{m,O_{s-1,1}} & \cdots & P_{m,O_{s-1,t-1}} \end{bmatrix}. \tag{A.2}$$

From (A.2), we can see that there exists a $s \times t$ matrix, denoted by O, in H. This matrix is called an *offset matrix* and it represents the exponent of $r(x)$ in each permutation matrix, i.e.

$$O = \begin{bmatrix} O_{0,0} & O_{0,1} & \cdots & O_{0,t-1} \\ O_{1,0} & O_{1,1} & \cdots & O_{1,t-1} \\ \vdots & \vdots & \vdots & \vdots \\ O_{s-1,0} & O_{s-1,1} & \cdots & O_{s-1,t-1} \end{bmatrix}$$

where $0 \leq O_{i,j} \leq m-1$, for $0 \leq i \leq s-1$ and $0 \leq j \leq t-1$. The permutation matrix $P_{m,j}$ has m rows and m columns, and since the matrix H contains s and t of these matrices per row and column respectively, the resulting code is a $[mt, m(t-s), d]$ quasi-cyclic LDPC code over $\mathbb{F}_2$.

In general, some of the permutation matrices $P_{i,j}$ in (A.2) may be zeros matrices. In this case, the resulting quasi-cyclic LDPC code is irregular and $O_{i,j}$ for which $P_{i,j} = O$ may be ignored. If none of the permutation matrices in (A.2) is a zero matrix, the quasi-cyclic LDPC code defined by (A.2) is a (s,t) regular LDPC code.

A.2 Construction of Quasi-Cyclic Codes using Protograph

A protograph is a miniature prototype Tanner graph of arbitrary size, which can be used to construct a larger Tanner graph by means of replicate and permute operations (Thorpe; 2003). A protograph may also be considered as an $[n', k']$ linear code $\mathcal{P}$ of small block length and dimension. A large code may be obtained expanding code $\mathcal{P}$ by an integer factor Q so that the resulting code has parameter $[n = n'Q, k = k'Q]$ over the same field. A simplest way to expand code $\mathcal{P}$ and also to impose structure in the resulting code is by replacing a non zero element of the parity-check matrix of code $\mathcal{P}$ with a $Q \times Q$ permutation matrix, and a zero element with a $Q \times Q$ zero matrix. As a consequence, the resulting code has a quasi-cyclic structure. The procedure may be described in more detail in Example A.2.

LDPC code from a protograph

Example A.2: Consider a code $\mathcal{P} = [4, 2]$ over $\mathbb{F}_2$ as a protograph. The parity-check matrix of code $\mathcal{P}$ is given by

$$H' = \begin{array}{c} \\ c_0 \\ c_1 \end{array} \begin{array}{c} \begin{array}{cccc} v_0 & v_1 & v_2 & v_3 \end{array} \\ \boxed{\begin{array}{cccc} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{array}} \end{array}. \tag{A.3}$$

Let the expansion factor $Q = 5$, the expanded code, which has parameter [20, 10], may have a parity-check matrix given by

$$H = \begin{array}{c|cccc|cccc|cccc|cccc|}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} \\
\hline
c_0 & \mathbf{1} & & & & & \mathbf{1} & & & & & & & & & & \mathbf{1} & & & & \\
c_1 & & 1 & & & & & 1 & & & & & & & & & & 1 & & & \\
c_2 & & & 1 & & & & & 1 & & & & & & & & & & 1 & & \\
c_3 & & & & 1 & & & & & 1 & & & & & & & & & & 1 & \\
c_4 & & & & & 1 & & & & & 1 & & & & & & & & & & 1 \\
\hline
c_5 & & & & & & \mathbf{1} & & & & & \mathbf{1} & & & & & \mathbf{1} & & & & \\
c_6 & & & & & & & 1 & & & & & 1 & & & & & 1 & & & \\
c_7 & & & & & & & & 1 & & & & & 1 & & & & & 1 & & \\
c_8 & & & & & & & & & 1 & & & & & 1 & & & & & 1 & \\
c_9 & & & & & & & & & & 1 & & & & & 1 & & & & & 1 \\
\hline
\end{array}, \quad \text{(A.4)}$$

where the zero elements have been omitted. This protograph construction may also be described using Tanner graph representation as shown in Figure A.1.

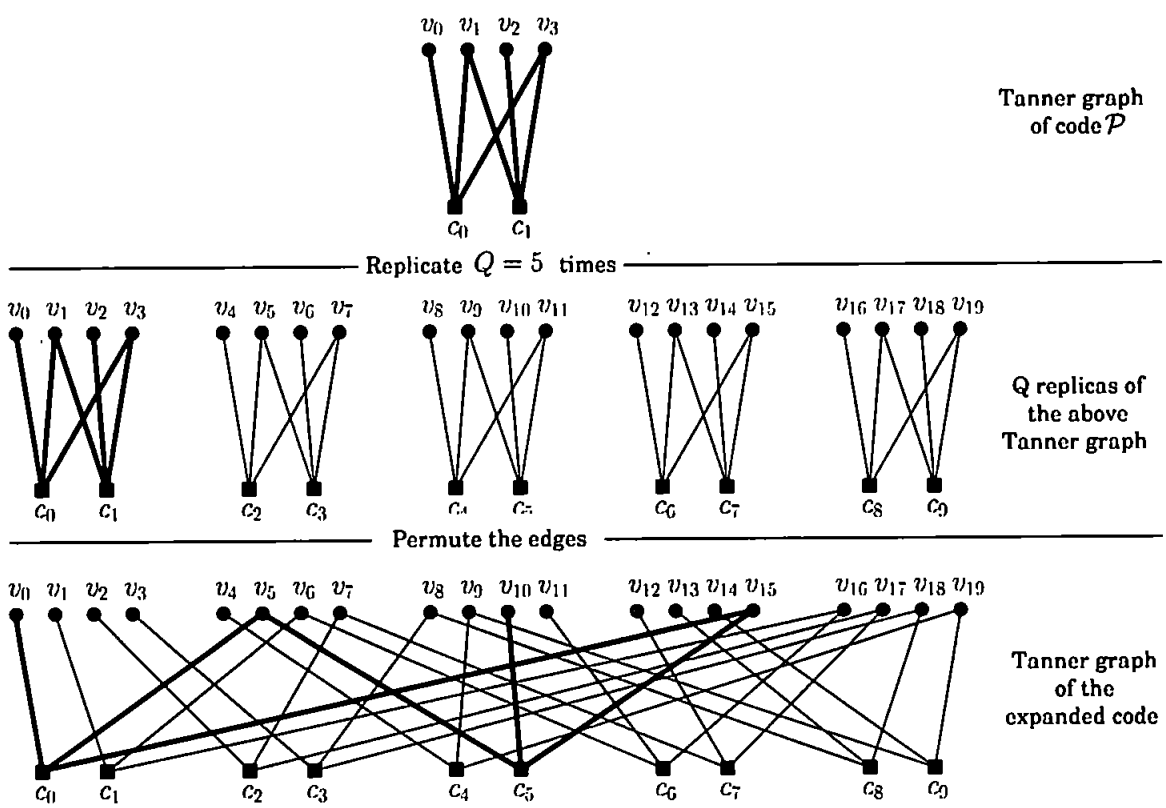


Figure A.1: Code construction using a protograph

Initially the tanner graph of code $\mathcal{P}$ is replicated Q times. The edges of these replicated Tanner graphs are then permuted. The edges may be permuted in many ways and in this particular example, we want the permutation to produce a code which has quasi-cyclic structure. The edges in Figure A.1 or equivalently the non zeros in (A.4) which are printed in bold represent the code $\mathcal{P}$.

The minimum Hamming distance of code $\mathcal{P}$ is 2 and this may be seen from its parity-check matrix, (A.3), where the summation of two column vectors, those of v_1 and v_3, produces a zero vector. The fact that in the expansion, only identity matrices are employed, the expanded code will have the same minimum Hamming distance as the protograph code. This is obvious from (A.4) where the summation of two column vectors, those of v_5 and v_{15}, produces a zero vector. In order to avoid expanded code of small minimum Hamming distance, permutation matrices may be used instead and the parity-check matrix of the expanded code may have the form given by (A.5),

$$H = \begin{array}{c|cccccccccccccccccccc|}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} & v_{17} & v_{18} & v_{19} \\
\hline
c_0 & & & 1 & & & & & & & 1 & & & & & & 1 & & & & \\
c_1 & & & & 1 & & 1 & & & & & & & & & & & 1 & & & \\
c_2 & & & & & 1 & & 1 & & & & & & & & & & & 1 & & \\
c_3 & 1 & & & & & & & 1 & & & & & & & & & & & 1 & \\
c_4 & & 1 & & & & & & & 1 & & & & & & & & & & & 1 \\
c_5 & & & & & & & 1 & & & & & & & 1 & & & 1 & & & \\
c_6 & & & & & & & & 1 & & & & & & & 1 & & & 1 & & \\
c_7 & & & & & & & & & 1 & & 1 & & & & & & & 1 & & \\
c_8 & & & & & & & & & & 1 & & 1 & & & & & & & & 1 \\
c_9 & & & & & & 1 & & & & & & & 1 & & & 1 & & & & \\
\hline
\end{array}, \qquad \text{(A.5)}$$

and the corresponding code defined by this parity-check matrix has minimum Hamming distance of 3. In addition, the cycle structure of the protograph is also preserved in the expanded code if only identity matrices for expansion. Since the protograph is such a small code, the variable vertex degree distribution required to design a good target code, which has much larger size than a protograph does in general, dictates many inevitable short cycles in the protograph. By using an appropriate permutation matrices in expansion, these short cycles may be avoided in the expanded code.

In the following, we describe a construction of a long quasi-cyclic LDPC code for applications in satellite communications. The standard for digital video broadcasting (DVB), which is commonly known as DVB-S2, makes use of a concatenation of LDPC and BCH codes to protect the video stream. The parity-check matrices of DVB-S2 LDPC codes contain a zig-zag matrix on the $n-k$ parity coordinates and quasi-cyclic matrices on the remaining k coordinates. In the literature, the code with this structure is commonly known as the Irregular Repeat Accumulate (IRA) code (Jin et al.; 2000). The code construction described below–using a protograph and greedy PEG expansion, is aimed at improving the performance compared to the rate 3/4 DVB-S2 LDPC code of block length 64800 bits. Let the [64800, 48600] LDPC code that we will construct be denoted by C_1. A protograph code, which has parameter [540, 405], is constructed using the PEG algorithm with a good variable vertex degree distributions obtained from Urbanke (2001),

$$\Lambda_{\lambda_1}(x) = \underbrace{0.00185185x + 0.248148x^2}_{\text{for zig-zag matrix}} + 0.55x^3 + 0.0592593x^5 + 0.0925926x^8 + 0.00555556x^{12} +$$
$$0.00185185x^{15} + 0.0166667x^{19} + 0.00185185x^{24} + 0.00185185x^{28} + 0.0203704x^{35}.$$

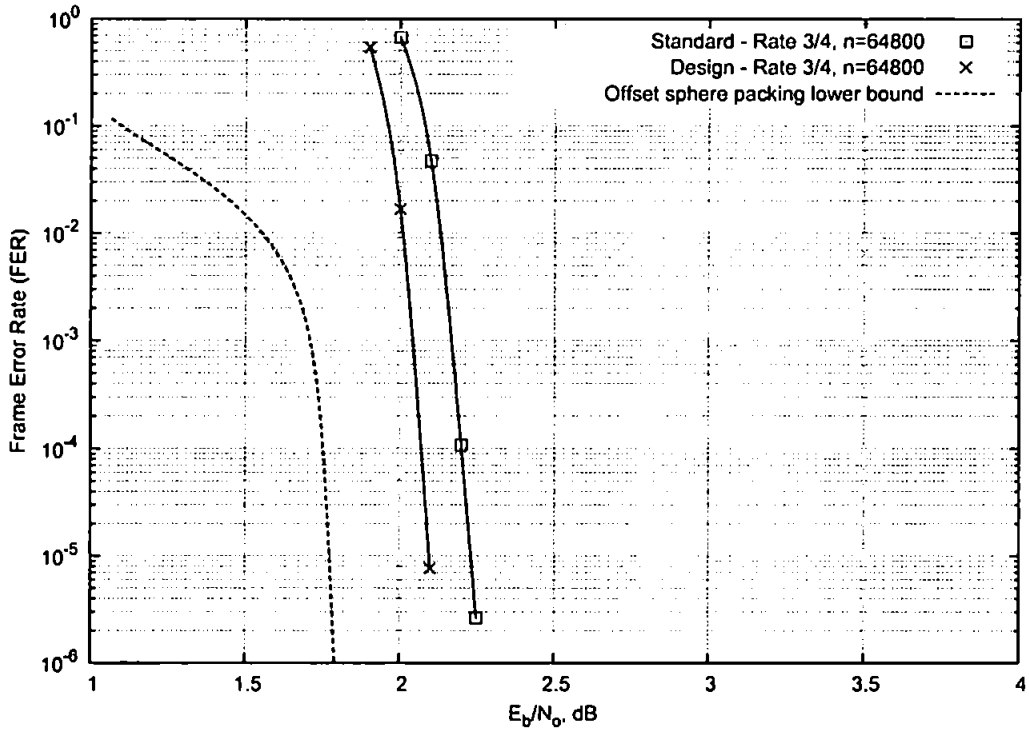


Figure A.2: FER performance of the DVB-S2 and the designed [64800, 48600] LDPC codes

The constructed [540, 405] protograph code has a parity-check matrix $H' = [H'_u \mid H'_p]$ where H'_p is a 135×135 zig-zag matrix, see (A.1), and H'_u is an irregular matrix satisfying $\Lambda_{\lambda_1}(x)$ above. In order to construct a [64800, 48600] LDPC code C_1, we need to expand the protograph code by a factor of $Q = 120$. In expanding the protograph code, we apply the greedy approach to construct the offset matrix O in order to obtain a Tanner graph for the [64800, 48600] LDPC code C_1, which has local girth maximised. This greedy approach examines all offset values, from 0 to $Q-1$, and pick an offset that results in highest girth or if there are more than one choices, one of them is randomly chosen. A 16200×48600 matrix H_u can be easily constructed by replacing a non zero element at coordinate (i, j) in H'_u with a permutation matrix $P_{Q,O_{i,j}}$. The resulting LDPC code C_1 has a parity-check matrix given by $H = [H_u \mid H_p]$, where, as before, H_p is given by (A.1).

In comparison, the rate 3/4 LDPC code of block length 64800 bits specified in the DVB-S2 standard takes a lower Q value, $Q = 45$. The protograph is a [1440, 1080] code which has the following variable vertex degree distributions

$$\Lambda_{\lambda_2}(x) = \underbrace{0.000694x + 0.249306x^2}_{\text{for zig-zag matrix}} + 0.666667x^3 + 0.083333x^{12}.$$

For convenience, we denote the DVB-S2 LDPC code by C_2.

Figure A.2 compares the FER performance of C_1 and C_2 using the belief propagation decoder with a maximum iteration of 100. Binary antipodal signalling and AWGN channel are assumed in the simulation. Note that, although the outer concatenation of BCH code is not used, there is still no sign of error floor at FER as low as 10^{-6}. It may be seen from Figure A.2 that the designed LDPC

code, which at 10^{-5} FER performs approximately 0.35 dB away from the sphere packing lower bound offset for binary transmission loss, is 0.1 dB better than the DVB-S2 code.

B Binary Cyclic Codes of Odd Lengths from 129 to 189

The highest minimum distance attainable by all binary cyclic codes of odd lengths $129 \leq n \leq 189$ is tabulated in Table B.1. The column "Roots of $g(x)$" in Table B.1 denotes the exponents of roots of the generator polynomial $g(x)$, excluding the conjugate roots. All cyclic codes with generator polynomials $1 + x$ and $(x^n - 1)/(1 + x)$, since they are trivial codes, are excluded in Table B.1 and since primes $n = 8m \pm 3$ contain these trivial cyclic codes only, there is no entry in the table for these primes. The number of permutation inequivalent (see Section 4.3 for the details on the permutation) and non degenerate cyclic codes, excluding the two trivial codes mentioned earlier, for each odd integer n is given by N_C. In Table B.1, there is no cyclic code that improves Brouwer's (1998) lower-bound, but there are 134 cyclic codes that meed this lower-bound and they are printed in bold.

Table B.1: The Highest Attainable Minimum Distance of Binary Cyclic Codes of Odd Lengths from 129 to 189

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
$n = 129$, $m(x) = 77277$, $N_C = 388$								
127	**2**	**43**	**84**	**14**	**0,1,19,21,43**	42	30	0,1,3,7,9,11,19,43
115	3	1	73	15	1,3,7,19	31	32	1,7,9,11,13,19,21
114	**6**	**0,1**	**72**	**18**	**0,1,7,9,19**	**30**	**38**	**0,1,3,7,9,11,13,19**
113	4	3,43	71	17	1,3,7,19,43	29	37	1,3,7,11,13,19,21,43
112	**6**	**0,1,43**	70	18	0,1,3,7,19,43	28	40	0,1,3,7,11,13,19,21,43
101	**8**	**1,9**	59	22	1,3,7,9,19	17	43	1,3,7,9,11,13,19,21
100	**10**	**0,1,3**	58	22	0,1,3,7,9,19	**16**	**52**	**0,1,3,7,9,11,13,19,21**
99	8	1,9,43	57	22	1,3,7,9,19,43	15	54	1,3,7,9,11,13,19,21,43
98	**10**	**0,1,3,43**	**56**	**24**	**0,1,5,9,19,21,43**	14	54	0,1,3,7,9,11,13,19,21,43
87	**13**	**1,13,21**	**45**	**29**	**1,3,7,9,11,19**	**2**	**86**	**0,1,3,5,7,9,11,13,19,21**
86	**14**	**0,1,19,21**	**44**	**30**	**0,1,3,7,9,11,19**			
85	13	1,19,21,43	43	29	1,3,7,9,11,19,43			
$n = 133$, $m(x) = 1334325$, $N_C = 198$								
130	**2**	**57**	91	8	1,7,19,57	43	19	1,7,9,15,31
129	**2**	**0,57**	90	10	0,1,19,31,57	42	28	0,1,5,7,9,31
127	**2**	**19,57**	79	14	1,7,31	**40**	**32**	**1,5,7,9,31,57**
126	**2**	**0,19,57**	78	14	0,1,5,9	**39**	**32**	**0,1,5,7,9,31,57**
115	3	1	76	16	1,7,31,57	37	32	1,5,7,9,19,31,57
114	4	0,1	75	16	0,1,7,31,57	36	32	0,1,5,7,9,19,31,57
112	**6**	**31,57**	73	16	1,7,19,31,57	25	19	1,3,5,7,9,31
111	6	0,31,57	72	16	0,1,7,19,31,57	24	38	0,1,3,5,7,9,31
109	6	1,19,57	61	19	1,7,9,31	22	44	1,5,7,9,15,31,57
108	6	0,1,19,57	**60**	**24**	**0,1,3,7,9**	21	44	0,1,5,7,9,15,31,57
97	7	1,31	**58**	**24**	**1,7,9,31,57**	19	48	1,3,5,7,9,19,31,57
96	10	0,1,31	**57**	**24**	**0,1,7,9,31,57**	18	48	0,1,3,5,7,9,19,31,57
94	8	7,31,57	55	24	1,7,9,19,31,57	4	57	1,3,5,7,9,15,31,57
93	10	0,1,31,57	54	24	0,1,7,9,19,31,57	**3**	**76**	**0,1,3,5,7,9,15,31,57**

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
$n = 135$, $m(x) = 1000000001001$, $N_C = 982$								
133	2	45	89	6	1,15,63	45	10	1,7,21,45,63
132	2	0,45	88	6	0,1,15,63	44	10	0,1,7,21,45,63
131	2	63	87	6	1,15,45,63	43	10	1,7,15,21,45
130	2	0,63	86	6	0,1,15,45,63	42	10	0,1,7,15,21,45
129	2	45,63	85	6	1,21,45	41	10	1,7,15,21,63
128	2	0,45,63	84	6	0,1,21,45	40	10	0,1,7,15,21,63
127	2	15,45	83	6	1,15,27,45,63	39	10	1,7,15,21,45,63
126	2	0,15,45	82	6	0,1,21,63	38	10	0,1,7,15,21,45,63
125	2	15,63	81	6	1,21,45,63	37	10	1,3,7,21,45
124	2	0,15,63	80	6	0,1,21,45,63	36	10	0,1,3,7,21,45
123	2	15,45,63	79	6	1,15,21,45	35	12	1,5,7,15,63
122	2	0,15,45,63	78	6	0,1,15,21,45	34	12	0,1,5,7,15,63
121	2	21,45	77	6	1,5,63	33	12	1,5,7,15,45,63
120	2	0,21,45	76	6	0,1,5,63	32	12	0,1,5,7,15,45,63
119	2	21,63	75	6	1,5,45,63	31	12	1,5,7,21,45
118	2	0,21,63	74	6	0,1,5,45,63	30	12	0,1,5,7,21,45
117	2	21,45,63	73	6	1,3,21,45	29	15	1,5,7,21,63
116	2	0,21,45,63	72	6	0,1,3,21,45	28	18	0,1,5,7,21,63
115	2	5,45	71	8	1,5,15,63	27	18	1,5,7,21,45,63
114	2	0,5,45	70	8	0,1,5,15,63	26	18	0,1,5,7,21,45,63
113	4	5,63	69	8	1,5,15,45,63	25	15	1,5,7,21,27,63
112	4	0,5,63	68	8	0,1,5,15,45,63	24	18	0,1,5,7,21,27,63
111	4	5,45,63	67	8	1,5,21,45	23	21	1,5,7,15,21,63
110	4	0,5,45,63	66	8	0,1,5,21,45	22	24	0,1,5,7,15,21,63
109	4	5,27,63	65	8	1,5,15,27,45,63	21	24	1,5,7,15,21,45,63
108	4	0,5,27,63	64	8	0,1,5,21,63	20	24	0,1,5,7,15,21,45,63
107	4	5,15,63	63	8	1,5,21,45,63	19	21	1,5,7,15,21,27,63
106	4	0,5,15,63	62	8	0,1,5,21,45,63	18	24	0,1,5,7,15,21,27,63
105	4	5,15,45,63	61	8	1,5,15,21,45	17	24	1,5,7,15,21,27,45,63
104	4	0,5,15,45,63	60	8	0,1,5,15,21,45	16	30	0,1,3,5,7,21,63
103	4	5,21,45	59	8	1,5,15,21,63	15	30	1,3,5,7,21,27,45
102	4	0,5,21,45	58	8	0,1,5,15,21,63	14	30	0,1,3,5,7,21,45,63
101	4	5,21,63	57	8	1,5,15,21,45,63	13	24	1,5,7,9,15,21,27,45,63
100	4	0,5,21,63	56	8	0,1,5,15,21,45,63	12	30	0,1,3,5,7,21,27,63
99	4	5,21,45,63	55	8	1,3,5,21,45	11	30	1,3,5,7,21,27,45,63
98	4	0,5,21,45,63	54	8	0,1,3,5,21,45	10	36	0,1,3,5,7,15,21,63
97	4	1,45	53	10	1,7,15,63	9	36	1,3,5,7,15,21,27,45
96	4	0,1,45	52	10	0,1,7,15,63	8	36	0,1,3,5,7,15,21,45,63
95	5	1,63	51	10	1,7,15,45,63	7	45	1,3,5,7,15,21,27,63
94	6	0,1,63	50	10	0,1,7,15,45,63	6	54	0,1,3,5,7,15,21,27,63
93	6	1,45,63	49	10	1,7,21,45	5	63	1,3,5,7,15,21,27,45,63
92	6	0,1,45,63	48	10	0,1,7,21,45	4	72	0,1,3,5,7,15,21,27,45,63
91	5	1,27,63	47	10	1,7,15,27,45,63			
90	6	0,1,27,63	46	10	0,1,7,21,63			
$n = 137$, $m(x) = 673573303732676066756 73$, $N_C = 2$								
69	21	1	68	22	0,1			
$n = 141$, $m(x) = 2146417666311013$, $N_C = 30$								
139	2	47	93	4	3,15,47	47	24	1,3,15,47
138	2	0,47	92	6	0,1,47	46	24	0,1,3,15,47
118	2	3	72	21	3,5	26	33	1,3,5
117	2	0,3	71	22	0,3,5	25	36	0,1,3,5
116	4	3,47	70	21	3,5,47	24	33	1,3,5,47
115	4	0,3,47	69	24	0,3,5,47	23	36	0,1,3,5,47
95	3	1	49	22	1,3,15			
94	6	0,1	48	22	0,1,3,15			

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
$n = 143$, $m(x) = 14523676054732450506 1$, $N_C = 16$								
133	2	13	83	11	1	61	24	1,11,13
132	2	0,13	82	12	0,1	60	24	0,1,11,13
131	2	11	73	11	1,13	23	11	1,5
130	2	0,11	72	16	0,1,13	22	22	0,1,5
121	4	11,13	71	13	1,11			
120	4	0,11,13	70	18	0,1,11			
$n = 145$, $m(x) = 3572445367$, $N_C = 40$								
141	2	29	89	14	1,5	57	26	1,5,11,29
140	2	0,29	88	14	0,1,5	56	26	0,1,5,11,29
117	5	1	85	14	1,5,29	33	29	1,3,5,11
116	8	0,1	84	14	0,1,5,29	32	44	0,1,3,5,11
113	5	1,29	61	24	1,5,11	29	46	1,3,5,11,29
112	10	0,1,29	60	24	0,1,5,11	28	46	0,1,3,5,11,29
$n = 147$, $m(x) = 100002000040201$, $N_C = 488$								
145	2	49	96	4	0,1,35,49	48	8	1,3,7,9,21,35
144	2	0,49	95	4	0,1,21,35	47	8	0,1,3,7,9,21,35
143	2	0,21	94	4	1,21,35,49	46	8	1,3,7,9,21,35,49
142	2	21,49	93	4	0,1,21,35,49	45	8	0,1,3,7,9,21,35,49
141	2	35	92	4	0,1,7,35	44	8	0,1,3,7,9,21,35,63
140	2	0,35	91	4	1,21,35,49,63	43	8	1,3,7,9,21,35,49,63
139	2	35,49	90	4	1,7,21,35	42	8	0,1,3,7,9,21,35,49,63
138	2	0,35,49	89	4	0,1,7,21,35	40	9	1,5,9,49
137	2	0,7,21	88	4	1,7,21,35,49	39	12	0,1,5,9,49
136	2	21,35,49	87	4	0,1,7,21,35,49	38	10	0,1,5,9,21
135	2	7,35	86	4	0,1,7,21,35,63	37	12	1,5,9,21,49
134	2	0,7,35	85	4	1,7,21,35,49,63	36	12	0,1,5,9,21,49
133	2	21,35,49,63	84	4	0,1,7,21,35,49,63	35	12	0,1,5,9,35
132	2	7,21,35	82	5	5,9,49	34	12	1,5,9,21,49,63
131	2	0,7,21,35	81	8	0,5,9,49	33	12	0,1,5,9,35,49
130	2	7,21,35,49	80	6	0,5,9,21	32	12	0,1,5,9,21,35
129	2	0,7,21,35,49	79	8	5,9,21,49	31	12	1,5,9,21,35,49
127	2	7,21,35,49,63	78	8	0,5,9,21,49	30	12	0,1,5,9,21,35,49
126	2	0,7,21,35,49,63	77	8	0,5,9,35	29	12	0,1,5,7,9,35
124	3	9,49	76	8	5,9,21,49,63	28	12	1,5,9,21,35,49,63
123	4	0,9,49	75	8	0,5,9,35,49	27	12	1,5,7,9,21,35
122	2	0,9,21	74	8	0,5,9,21,35	26	12	0,1,5,7,9,21,35
121	4	9,21,49	73	8	5,9,21,35,49	25	12	1,5,7,9,21,35,49
120	4	0,9,21,49	72	8	0,5,9,21,35,49	24	12	0,1,5,7,9,21,35,49
119	4	0,9,35	71	8	0,5,7,9,35	23	12	0,1,5,7,9,21,35,63
118	4	9,21,49,63	70	8	5,9,21,35,49,63	22	12	1,5,7,9,21,35,49,63
117	4	0,9,35,49	69	8	5,7,9,21,35	21	12	0,1,5,7,9,21,35,49,63
116	4	0,9,21,35	68	8	0,5,7,9,21,35	19	14	1,3,5,9,49
115	4	9,21,35,49	67	8	5,7,9,21,35,49	18	14	0,1,3,5,9,49
114	4	0,9,21,35,49	66	8	0,5,7,9,21,35,49	17	14	0,1,3,5,9,21
113	4	0,7,9,35	65	8	0,5,7,9,21,35,63	16	21	1,3,5,9,21,49
112	4	9,21,35,49,63	64	8	5,7,9,21,35,49,63	15	28	0,1,3,5,9,21,49
111	4	7,9,21,35	63	8	0,5,7,9,21,35,49,63	14	28	0,1,3,5,9,35
110	4	0,7,9,21,35	61	8	1,3,9,49	13	28	1,3,5,9,21,49,63
109	4	7,9,21,35,49	60	8	0,1,3,9,49	12	35	1,3,5,7,9,21
108	4	0,7,9,21,35,49	59	6	0,1,5,21	11	42	0,1,3,5,7,9,21
107	4	0,7,9,21,35,63	58	8	1,3,9,21,49	10	35	1,3,5,7,9,21,49
106	4	7,9,21,35,49,63	57	8	1,3,9,35	9	56	0,1,3,5,7,9,21,49
105	4	0,7,9,21,35, 49,63	56	8	0,1,3,9,35	8	42	0,1,3,5,7,9,35
103	4	3,9,49	55	8	1,3,9,35,49	7	56	1,3,5,9,21,35,49,63
102	4	0,1,49	54	8	0,1,3,9,35,49	6	56	0,1,3,5,9,21,35,49,63

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
101	4	0,1,21	53	8	0,1,3,9,21,35	5	70	0,1,3,5,7,9,21,35
100	4	1,21,49	52	8	1,3,9,21,35,49	4	63	1,3,5,7,9,21,35,49
99	4	0,1,21,49	51	8	1,3,7,9,35	3	84	0,1,3,5,7,9,21,35,49
98	4	0,1,35	50	8	0,1,3,7,9,35			
97	4	1,21,49,63	49	8	1,3,9,21,35,49,63			
$n = 151$, $m(x) = 166761$, $N_C = 212$								
136	5	1	91	17	1,5,15,37	46	31	1,5,7,11,15,23,37
135	6	0,1	90	18	0,1,5,15,37	45	36	0,1,5,7,11,15,23,37
121	8	1,5	76	23	1,5,15,35,37	31	47	1,5,7,11,15,17,23,37
120	8	0,1,5	75	24	0,1,5,15,35,37	30	48	0,1,5,7,11,15,17,23,37
106	13	1,3,5	61	31	1,3,5,11,15,37	16	60	1,5,7,11,15,17,23,35,37
105	14	0,1,3,5	60	32	0,1,3,5,11,15,37	15	60	0,1,5,7,11,15,17,23,35,37
$n = 153$, $m(x) = 110110001$, $N_C = 2114$								
151	2	51	99	8	1,9,15,17,27	51	19	1,5,9,11,15,17,27
150	2	0,51	98	8	0,1,9,15,17,27	50	24	0,1,5,9,11,15,17,27
145	2	9	97	9	1,5,15	49	24	1,5,9,11,15,17,27,51
144	2	0,9	96	10	0,1,5,15	48	24	0,1,5,9,11,15,17,27,51
143	2	9,51	95	10	1,5,9,51	47	18	1,5,9,11,15,27,33,51
142	2	0,9,51	94	10	0,1,5,9,51	46	18	0,1,5,9,11,15,27,33,51
139	4	9,17	91	9	1,5,15,17	43	19	1,5,9,11,15,17,27,33
138	4	0,9,17	90	10	0,1,5,15,17	42	24	0,1,5,9,11,15,17,27,33
137	4	9,17,51	89	13	1,5,9,57	41	24	1,5,9,11,15,17,27,33,51
136	4	0,9,17,51	88	14	0,1,5,9,57	40	24	0,1,5,9,11,15,17,27,33,51
135	2	9,27,51	87	14	1,5,9,51,57	39	18	1,5,9,11,15,19,51
134	2	0,9,27,51	86	14	0,1,5,9,51,57	38	18	0,1,5,9,11,15,19,51
131	4	9,17,27	83	15	1,5,9,17,57	35	19	1,5,9,11,15,17,27,33,57
130	4	0,9,17,27	82	16	0,1,5,9,17,57	34	24	0,1,5,9,11,15,17,27,33,57
129	4	9,17,27,51	81	16	1,5,9,17,51,57	33	24	1,5,9,11,15,17,27,33,51,57
128	4	0,9,17,27,51	80	16	0,1,5,9,17,51,57	32	30	0,1,5,9,11,15,19,57
127	4	1,51	79	14	1,5,9,15,27,51	31	30	1,5,9,11,15,19,51,57
126	4	0,1,51	78	14	0,1,5,9,15,27,51	30	30	0,1,5,9,11,15,19,51,57
123	4	9,15,17,27	75	16	1,5,9,15,17,27	27	27	1,5,9,11,15,17,19,57
122	4	0,9,15,17,27	74	16	0,1,5,9,15,17,27	26	30	0,1,5,9,11,15,17,19,57
121	5	1,9	73	16	1,5,9,15,17,27,51	25	30	1,5,9,11,15,17,19,51,57
120	6	0,1,9	72	16	0,1,5,9,15,17,27,51	24	34	0,1,5,9,11,15,19,27,57
119	6	1,9,51	71	14	1,5,9,15,27,33,51	23	34	1,5,9,11,15,19,27,33,51
118	6	0,1,9,51	70	14	0,1,5,9,15,27,33,51	22	34	0,1,5,9,11,15,19,27,33,51
115	6	1,9,17	67	16	1,5,9,15,17,27,33	19	42	1,5,9,11,15,17,19,27,57
114	6	0,1,9,17	66	16	0,1,5,9,15,17,27,33	18	42	0,1,5,9,11,15,17,19,27,57
113	8	1,9,57	65	16	1,5,9,15,17,27,33,51	17	48	1,5,9,11,15,17,19,27,51,57
112	8	0,1,9,57	64	18	0,1,5,9,11,57	16	48	0,1,5,9,11,15,17,19, 27,51,57
111	8	1,9,27,51	63	18	1,5,9,19,51,57	15	34	1,5,9,11,15,19,27,33,51,57
110	8	0,1,9,27,51	62	18	0,1,5,9,11,51,57	14	34	0,1,5,9,11,15,19, 27,33,51,57
107	8	1,9,17,57	59	16	1,5,9,15,17,27,33,57	11	51	1,5,9,11,15,17,19,27,33,57
106	8	0,1,9,17,57	58	18	0,1,5,9,11,17,57	10	54	0,1,5,9,11,15,17, 19,27,33,57
105	8	1,9,15,27	57	18	1,5,9,11,17,51,57	9	57	1,5,9,11,15,17,19, 27,33,51,57
104	8	0,1,9,15,27	56	18	0,1,5,9,11,15,27	8	72	0,1,5,9,11,15,17,19, 27,33,51,57
103	8	1,9,15,27,51	55	18	1,5,9,11,15,27,51	7	34	1,3,5,9,11,15,19, 27,33,51,57
102	8	0,1,9,15,27,51	54	18	0,1,5,9,11,15,27,51	6	34	0,1,3,5,9,11,15,19, 27,33,51,57
$n = 155$, $m(x) = 7154113$, $N_C = 2768$								
151	2	31	101	12	1,3,25,31,75	51	24	1,3,9,23,25,31,35,55,75
150	2	0,31	100	12	0,1,9,25,31,75	50	24	0,1,3,9,23,25,31,35,55,75
149	2	0,25	99	10	0,1,9,25,35,75	49	22	0,1,3,5,11,23,25,35,55,75
146	4	25,31	96	12	1,9,25,31,35,75	46	24	1,3,5,11,23,25,31,35,55,75
145	4	0,25,31	95	12	0,1,9,25,31,35,75	45	25	1,3,9,11,23,25,75
144	2	0,25,75	94	10	0,1,11,25,35,55,75	44	28	0,1,3,9,11,23,25,75

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
141	**4**	**25,31,75**	91	12	1,11,25,31,35,55,75	41	25	1,3,9,11,23,25,31,75
140	**4**	**0,25,31,75**	90	12	0,1,11,25,31, 35,55,75	40	30	0,1,3,9,11,23,25,31,75
139	2	0,25,35,75	89	12	0,1,3,11,25	39	30	0,1,3,9,11,23,25,35,75
136	4	25,31,35,75	86	12	9,11,23,25,31	36	31	1,3,9,11,23,25,31,35,75
135	4	0,25,31,35,75	85	14	1,3,9,25,75	35	32	0,1,3,9,11,23,25,31,35,75
134	4	0,1	84	14	0,1,3,9,25,75	34	30	0,1,3,9,11,23,25,35,55,75
131	4	1,31	81	16	1,3,9,25,31,75	31	32	1,3,9,11,23,25,31,35,55,75
130	5	1,25	80	16	0,1,3,9,25,31,75	30	32	0,1,3,9,11,23,25, 31,35,55,75
129	6	0,1,25	79	14	0,1,3,9,25,35,75	29	30	0,1,3,5,9,11,23,25,35,55,75
126	6	1,25,31	76	16	1,3,9,25,31,35,75	26	32	1,3,5,9,11,23,25, 31,35,55,75
125	6	0,1,25,31	75	16	0,1,3,9,25,31,35,75	25	32	0,1,3,5,9,11,23, 25,31,35,55,75
124	6	0,1,25,75	74	14	0,1,3,9,25,35,55,75	24	30	0,1,3,7,9,11,23,25,75
121	8	1,25,31,75	71	16	1,9,11,25,31, 35,55,75	21	32	1,3,5,9,11,15,23, 25,31,35,55,75
120	8	0,1,25,31,75	70	16	0,1,9,11,25, 31,35,55,75	20	32	0,1,3,5,9,11,15,23, 25,31,35,55,75
119	6	0,1,25,35,75	69	16	0,1,9,11,23,25	19	40	0,1,3,7,9,11,23,25,35,75
116	8	1,25,31,35,75	66	16	1,5,9,11,25, 31,35,55,75	16	35	1,3,7,9,11,23,25,31,35,75
115	8	0,1,25,31,35,75	65	16	0,1,3,9,11,25,31	15	40	0,1,3,7,9,11,23,25,31,35,75
114	6	0,1,11	64	20	0,1,9,11,23,25,55	14	60	0,1,3,7,9,11,23,25,35,55,75
111	8	1,25,31,35, 55,75	61	20	1,3,9,23,25,31,75	11	55	1,3,7,9,11,23,25, 31,35,55,75
110	8	0,1,25,31, , 35,55,75	60	22	1,3,9,23,25,35,75	10	60	0,1,3,7,9,11,23, 25,31,35,55,75
109	8	0,1,11,25	59	22	0,1,3,9,23,25,35,75	9	62	0,1,3,5,7,9,11,23, 25,35,55,75
106	8	1,11,25,31	56	24	1,3,9,23,25,31,35,75	6	75	1,3,5,7,9,11,23,25, 31,35,55,75
105	10	1,3,25,75	55	24	0,1,3,9,23, 25,31,35,75	**5**	**80**	**0,1,3,5,7,9,11, 23,25,31,35,55,75**
104	10	0,1,9,25,75	54	22	0,1,3,9,23, 25,35,55,75			
$n = 157$, $m(x) = 352125723713652127$, $N_C = 4$								
105	13	1	53	26	1,3			
104	**14**	**0,1**	52	26	0,1,3			
$n = 159$, $m(x) = 303667410520550411$, $N_C = 16$								
157	**2**	**53**	105	4	3,53	53	32	1,3,53
156	**2**	**0,53**	104	6	0,1,53	52	32	0,1,3,53
107	3	1	55	30	1,3			
106	6	0,1	54	30	0,1,3			
$n = 161$, $m(x) = 150536353761$, $N_C = 156$								
158	**2**	**23**	106	4	1,7,35	56	7	1,3,5,23,69
157	**2**	**0,23**	105	4	0,1,7,35	55	14	0,1,3,5,23,69
155	**2**	**23,69**	103	8	5,7,23,35	51	23	1,5,11,35
154	**2**	**0,23,69**	102	8	0,5,7,23,35	50	28	0,1,5,11,35
150	2	35	100	8	1,7,23,35,69	48	32	3,5,11,23,35
149	2	0,35	99	8	0,1,7,23,35,69	47	32	0,3,5,11,23,35
147	**4**	**23,35**	95	7	1,5	45	32	1,5,11,23,35,69
146	**4**	**0,23,35**	94	14	0,1,5	44	32	0,1,5,11,23,35,69
144	4	23,35,69	92	7	1,3,23	40	23	1,3,5,7,35
143	4	0,23,35,69	91	14	0,1,5,23	39	28	0,1,3,5,7,35
139	2	7,35	89	7	1,5,23,69	37	32	1,3,5,7,23,35
138	2	0,7,35	88	14	0,1,5,23,69	36	32	0,1,3,5,7,23,35
136	4	7,23,35	84	14	1,5,35	34	32	1,3,5,7,23,35,69
135	4	0,7,23,35	83	14	0,1,5,35	33	32	0,1,3,5,7,23,35,69
133	4	7,23,35,69	81	16	5,11,23,35	29	7	1,3,5,11
132	4	0,7,23,35,69	80	18	0,3,11,23,35	28	14	0,1,3,5,11
128	3	1	78	18	1,5,23,35,69	26	7	1,3,5,11,23
127	4	0,1	77	18	0,1,5,23,35,69	25	14	0,1,3,5,11,23
125	6	5,23	73	23	1,5,7,35	18	23	1,3,5,11,35
124	6	0,5,23	72	24	0,1,5,7,35	17	46	0,1,3,5,11,35

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
122	6	1,23,69	70	24	1,5,7,23,35	15	49	1,3,5,11,23,35
121	6	0,1,23,69	69	24	0,1,5,7,23,35	14	56	0,1,3,5,11,23,35
117	4	1,35	**67**	**28**	**1,5,7,23,35,69**	12	49	1,3,5,11,23,35,69
116	4	0,1,35	66	28	0,1,5,7,23,35,69	11	56	0,1,3,5,11,23,35,69
114	8	5,23,35	62	7	1,3,5	4	69	1,3,5,7,11,23,35
113	8	0,5,23,35	61	14	0,1,3,5	**3**	**92**	**0,1,3,5,7,11,23,35**
111	8	1,23,35,69	59	7	1,3,5,23			
110	8	0,1,23,35,69	58	14	0,1,3,5,23			
$n = 165$, $m(x) = 6223427$, $N_C = 4800$								
163	2	**55**	109	12	5,9,29,55,77	55	32	1,5,7,9,15,29,33,55,77
162	2	**0,55**	108	12	0,5,9,29,55,77	54	32	0,1,5,7,9,15,29,33,55,77
161	2	**77**	107	12	5,9,29,33,77	53	32	1,5,7,9,11,15,29,33,77
160	2	**0,77**	106	12	0,5,9,29,33,77	52	32	0,1,5,7,9,11,15,29,33,77
159	2	**55,77**	105	12	5,9,29,33,55,77	51	32	1,5,7,9,11,15,29,33,55,77
158	2	**0,55,77**	104	12	0,5,9,29,33,55,77	50	32	0,1,5,7,9,11,15,29,33,55,77
157	2	33,77	103	12	1,5,9,11,33,77	49	28	1,5,7,9,15,25,29,55,77
156	2	0,33,77	102	12	0,1,9,29,55	48	30	0,1,3,5,7,9,29,55,77
155	2	5	101	12	5,9,15,29,77	47	32	1,5,7,9,15,25,29,33,77
154	2	0,5	100	12	0,1,9,29,77	46	32	0,1,5,7,9,15,25,29,33,77
153	2	5,55	99	12	1,9,29,33,55	45	32	1,5,7,9,15,25,29,33,55,77
152	2	0,5,55	98	12	0,5,9,15,29,55,77	44	32	0,1,5,7,9,15,25,29,33,55,77
151	**4**	**15,77**	97	16	5,9,15,29,33,77	43	32	1,5,7,9,11,15,25,29,33,77
150	**4**	**0,5,77**	96	16	0,5,9,15,29,33,77	42	32	0,1,5,7,9,11,15,25,29,33,77
149	4	15,55,77	95	16	5,9,15,29,33,55,77	41	33	1,3,5,7,9,15,29,77
148	4	0,5,55,77	94	16	0,5,9,15,29,33,55,77	40	38	0,1,3,5,7,9,15,29,77
147	4	5,33,77	93	16	1,3,5,7,55	39	39	1,5,7,9,15,19,29,33,55
146	4	0,5,33,77	92	16	0,1,5,9,29,55	**38**	**44**	**0,1,3,5,7,9,15,29,55,77**
145	4	5,33,55,77	91	16	5,9,19,29,77	37	40	1,3,5,7,9,15,29,33,77
144	4	0,1	90	18	0,1,5,9,29,33	36	44	0,1,5,7,9,15,19,29,33,77
143	4	9,55	89	19	1,3,7,15,55,77	35	44	1,3,5,7,9,15,29,33,55,77
142	4	0,1,55	88	20	0,1,5,9,29,55,77	34	44	0,1,3,5,7,9,15,29,33,55,77
141	5	1,33	87	16	5,9,15,25,29,33,77	33	44	1,3,5,7,9,11,15,29,33,77
140	6	0,29,77	86	20	0,1,5,9,29,33,77	32	44	0,1,3,5,7,9,11,15,29,33,77
139	6	1,33,55	85	20	1,3,7,15,33,55,77	31	44	1,3,5,7,9,11,15,29,33,55,77
138	6	0,29,55,77	84	20	0,1,5,9,29,33,55,77	30	44	0,1,3,5,7,9,15,25,29,77
137	5	29,33,77	83	17	1,5,9,15,29,55	29	44	1,5,7,9,15,19,25,29,33,55
136	6	0,1,33,77	82	20	0,1,5,9,15,29,55	28	44	0,1,3,5,7,9,15,25,29,55,77
135	6	1,33,55,77	81	21	1,5,9,15,29,77	27	48	1,3,5,7,9,15,25,29,33,77
134	6	0,1,33,55,77	**80**	**24**	**0,1,3,5,7,15,77**	26	48	0,1,3,5,7,9,15,25,29,33,77
133	5	1,11,33,77	79	23	1,3,5,7,15,55,77	25	48	1,3,5,7,9,15,25,29,33,55,77
132	6	0,1,11,33,77	78	24	0,1,3,5,7,15,55,77	24	48	0,1,3,5,7,9,15, 25,29,33,55,77
131	7	3,5,77	77	24	1,5,9,15,29,33,77	23	48	1,3,5,7,9,11,15,25,29,33,77
130	8	0,5,9,77	76	24	0,1,5,9,15,29,33,77	22	48	0,1,3,5,7,9,11,15, 25,29,33,77
129	8	1,15,33,55	75	24	1,5,9,15,29,33,55,77	21	48	1,3,5,7,9,11,15, 25,29,33,55,77
128	8	0,5,9,55,77	74	24	0,1,5,9,15, 29,33,55,77	20	48	0,1,3,5,7,9,11, 15,25,29,33,55,77
127	8	5,29,33,77	73	24	1,5,9,11,15,29,33,77	19	44	1,3,5,7,9,15,19,29,33,55
126	8	0,1,5,33,77	72	24	0,1,5,9,11, 15,29,33,77	18	44	0,1,3,5,7,9,15,19,29,55,77
125	8	5,29,33,55,77	71	24	1,3,5,7,15,25,77	17	50	1,3,5,7,9,15,19,29,33,77
124	8	0,1,5,33,55,77	70	24	0,3,5,7,19,29,77	16	50	0,1,3,5,7,9,15,19,29,33,77
123	8	1,9,55	69	24	1,7,9,15,29,55,77	15	55	1,3,5,7,9,15,19,29,33,55,77
122	8	0,1,9,55	68	24	0,1,5,7,9,29,55,77	14	60	0,1,3,5,7,9,15,19, 29,33,55,77
121	8	5,9,15,77	67	24	1,5,9,15,25,29,33,77	13	50	1,3,5,7,9,11,15,19,29,33,77
120	10	0,7,9,77	66	26	0,1,5,9,19,29,33,77	12	50	0,1,3,5,7,9,11,15, 19,29,33,77
119	10	7,9,55,77	65	24	1,5,9,15,25, 29,33,55,77	11	55	1,3,5,7,9,11,15, 19,29,33,55,77
118	10	0,7,9,55,77	64	28	0,1,5,9,19, 29,33,55,77	10	60	0,1,3,5,7,9,11,15, 19,29,33,55,77
117	8	1,5,15,33,77	63	24	1,5,7,9,15,29,55	9	44	1,3,5,7,9,15,19,25,29,33,55
116	10	0,9,29,33,77	62	28	0,1,5,9,11, 19,29,33,77	8	44	0,1,3,5,7,9,15, 19,25,29,55,77

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
115	10	9,29,33,55,77	61	27	1,5,7,9,15,29,77	7	55	1,3,5,7,9,15,19,25,29,33,77
114	10	0,9,29,33,55,77	60	28	0,1,5,7,9,15,29,77	6	66	0,1,3,5,7,9,15, 19,25,29,33,77
113	8	1,9,15,55	59	28	1,5,7,9,15,29,55,77	5	77	1,3,5,7,9,15,19, 25,29,33,55,77
112	10	0,1,9,11,33,77	58	28	0,1,5,7,9, 15,29,55,77	4	88	**0,1,3,5,7,9,15, 19,25,29,33,55,77**
111	11	5,7,9,77	57	32	1,5,7,9,15,29,33,77			
110	12	0,5,7,9,77	56	32	0,1,5,7,9, 15,29,33,77			
$n = 167$, $m(x) = 51226225446671215657424325 23$, $N_C = 2$								
84	23	1	83	24	0,1			
$n = 169$, $m(x) = 1000040002000100004000200010000400020001000040002 0001$, $N_C = 2$								
157	2	13	12	26	0,1			
$n = 171$, $m(x) = 1167671$, $N_C = 802$								
169	**2**	**57**	111	12	1,3,9,19	57	35	1,3,5,7,9,13,19
168	**2**	**0,57**	**110**	**16**	**0,1,3,5,19**	**56**	**36**	**0,1,3,5,7,9,13,19**
163	2	19,57	109	12	1,3,9,19,57	55	36	1,3,5,7,9,13,19,57
162	2	0,19,57	108	16	0,1,3,5,19,57	54	36	0,1,3,5,7,9,13,19,57
153	3	1	99	18	1,3,9,13	45	19	1,3,5,7,9,15,17
152	**6**	**0,1**	98	18	0,1,3,9,13	44	38	0,1,3,5,7,9,15,17
151	5	1,57	97	18	1,3,9,13,57	43	38	1,3,5,7,9,15,17,57
150	**6**	**0,1,57**	96	18	0,1,3,5,7,57	42	38	0,1,3,5,7,9,13,17,57
147	4	9,19	93	20	1,3,5,9,19	39	45	1,3,5,7,9,15,17,19
146	6	0,1,19	92	20	0,1,3,5,9,19	**38**	**48**	**0,1,3,5,7,9,15,17,19**
145	6	1,19,57	91	21	1,3,5,9,19,57	**37**	**48**	**1,3,5,7,9,15,17,19,57**
144	6	0,1,19,57	**90**	**22**	**0,1,3,5,9,19,57**	36	48	0,1,3,5,7,9,15,17,19,57
135	9	1,3	81	19	1,3,5,7,9	27	19	1,3,5,7,9,13,15,17
134	**10**	**0,1,3**	**80**	**26**	**0,1,3,5,7,9**	26	38	0,1,3,5,7,9,13,15,17
133	9	1,3,57	79	23	1,3,5,9,17,57	25	38	1,3,5,7,9,13,15,17,57
132	10	0,1,3,57	78	26	0,1,3,5,7,9,57	24	38	0,1,3,5,7,9,13,15,17,57
129	9	1,3,19	75	27	1,3,5,9,17,19	21	55	1,3,5,7,9,13,15,17,19
128	10	0,1,3,19	**74**	**28**	**0,1,3,5,9,17,19**	20	64	0,1,3,5,7,9,13,15,17,19
127	10	1,9,19,57	73	28	1,3,5,9,17,19,57	**19**	**68**	**1,3,5,7,9,13,15,17,19,57**
126	**12**	**0,1,15,19,57**	72	28	0,1,3,5,9,17,19,57	**18**	**68**	**0,1,3,5,7,9,13,15,17,19,57**
117	10	1,3,9	63	19	1,3,5,7,9,25	7	38	1,3,5,7,9,13,15,17,25,57
116	14	0,1,3,5	62	32	0,1,3,5,9,17,25	6	38	0,1,3,5,7,9,13,15,17,25,57
115	12	1,7,9,57	61	32	1,3,5,9,17,25,57			
114	14	0,1,7,9,57	60	32	0,1,3,5,9,17,25,57			
$n = 175$, $m(x) = 10004102040000 4002041$, $N_C = 242$								
172	**2**	**25**	112	6	3,25	60	8	0,1,5,7,15,25,35,75
171	**2**	**0,25**	111	6	0,3,25	52	7	1,3,25
170	**2**	**0,35**	110	4	0,1,35	51	10	0,1,3,25
169	**2**	**25,75**	109	6	1,25,75	50	10	0,1,3,35
168	**2**	**0,25,75**	108	6	0,1,25,75	49	7	1,3,25,75
167	2	0,25,35	107	6	0,3,25,35	48	14	1,3,25,35
165	2	25,35,75	105	6	1,25,35,75	47	14	0,1,3,25,35
164	2	0,25,35,75	104	6	0,1,25,35,75	45	14	1,3,25,35,75
163	2	5	103	6	1,5	44	14	0,1,3,25,35,75
162	2	0,5	102	6	0,1,5	43	7	1,3,5
160	2	5,25	100	6	1,5,25	42	14	0,1,3,5
159	2	0,5,25	99	6	0,1,5,25	40	7	1,3,5,25
158	2	0,5,35	98	6	0,1,5,35	39	14	0,1,3,5,25
157	2	5,25,75	97	6	1,5,25,75	38	14	0,1,3,5,35
156	2	0,5,25,75	96	6	0,1,5,25,75	37	7	1,3,5,25,75
155	2	0,5,25,35	95	6	0,1,5,25,35	36	14	0,1,3,5,25,75
153	2	5,25,35,75	93	6	1,5,25,35,75	35	14	0,1,3,5,25,35
152	4	7,25	92	7	3,7,25	33	14	1,3,5,25,35,75
151	4	0,7,25	91	8	0,3,7,25	32	14	0,1,3,5,25,35,75

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
150	2	0,7,35	90	6	0,1,5,15	31	10	0,1,3,7,25
149	4	7,25,75	89	8	1,7,25,75	30	14	0,1,3,5,15
148	4	0,7,25,75	88	8	0,1,7,25,75	29	10	1,3,7,25,75
147	4	0,7,25,35	87	8	0,3,7,25,35	28	20	1,3,7,25,35
145	4	7,25,35,75	86	6	0,1,5,15,35	27	20	0,1,3,7,25,35
144	4	0,7,25,35,75	85	8	1,7,25,35,75	26	14	0,1,3,5,15,35
143	4	5,7	84	8	0,1,7,25,35,75	25	20	1,3,7,25,35,75
142	4	0,5,7	83	7	1,5,7	24	20	0,1,3,7,25,35,75
141	2	5,15,25,35,75	82	8	0,1,5,7	23	15	1,3,5,7
140	4	5,7,25	81	6	1,5,15,25,35,75	22	20	0,1,3,5,7
139	4	0,5,7,25	80	8	1,5,7,25	21	14	1,3,5,15,25,35,75
138	4	0,5,7,35	79	8	0,1,5,7,25	20	30	1,3,5,7,25
137	4	5,7,25,75	78	8	0,1,5,7,35	19	30	0,1,3,5,7,25
136	4	0,5,7,25,75	77	8	1,5,7,25,75	18	20	0,1,3,5,7,35
135	4	0,5,7,25,35	76	8	0,1,5,7,25,75	17	30	1,3,5,7,25,75
133	4	5,7,25,35,75	75	8	0,1,5,7,25,35	16	35	1,3,5,7,25,35
132	4	0,5,7,25,35,75	73	8	1,5,7,25,35,75	15	40	0,1,3,5,7,25,35
131	4	5,7,15	72	8	0,1,5,7,25,35,75	13	40	1,3,5,7,25,35,75
130	4	0,5,7,15	71	8	1,5,7,15	12	40	0,1,3,5,7,25,35,75
128	4	5,7,15,25	70	8	0,1,5,7,15	11	25	1,3,5,7,15
127	4	0,5,7,15,25	68	8	1,5,7,15,25	10	50	0,1,3,5,7,15
126	4	0,5,7,15,35	67	8	0,1,5,7,15,25	8	35	1,3,5,7,15,25
124	4	5,7,15,25,35	66	8	0,1,5,7,15,35	7	70	0,1,3,5,7,15,25
123	4	0,5,7,15,25,35	64	8	1,5,7,15,25,35	4	75	1,3,5,7,15,25,35
121	4	5,7,15,25,35,75	63	8	0,1,5,7,15,25,35	3	100	0,1,3,5,7,15,25,35
120	4	0,5,7,15, 25,35,75	61	8	1,5,7,15,25,35,75			
$n = 177$, $m(x) = 23563311065422331671$, $N_C = 16$								
175	2	59	117	4	3,59	59	30	1,3,59
174	2	0,59	116	6	0,1,59	58	30	0,1,3,59
119	3	1	61	28	1,3			
118	6	0,1	60	28	0,1,3			
$n = 183$, $m(x) = 1310103544416375716 37$, $N_C = 16$								
181	2	61	121	4	3,61	61	36	1,3,61
180	2	0,61	120	6	0,1,61	60	36	0,1,3,61
123	3	1	63	34	1,3			
122	6	0,1	62	34	0,1,3			
$n = 185$, $m(x) = 1761557733077$, $N_C = 40$								
181	2	37	113	14	1,5	73	32	1,3,5,37
180	2	0,37	112	14	0,1,5	72	32	0,1,3,5,37
149	5	1	109	16	1,5,37	41	37	1,3,5,19
148	8	0,1	108	16	0,1,5,37	40	48	0,1,3,5,19
145	5	1,37	77	28	1,3,5	37	37	1,3,5,19,37
144	8	0,1,37	76	28	0,1,3,5	36	54	0,1,3,5,19,37
$n = 187$, $m(x) = 36000132706473$, $N_C = 78$								
179	2	33	129	12	3,17,33	59	17	1,3,9,33
178	2	0,33	128	12	0,3,17,33	58	30	0,1,3,23,33
177	2	17	121	12	1,11,17,33	57	11	1,3,9,17
176	2	0,17	120	12	0,1,11,17,33	56	22	0,1,3,9,17
171	2	11,33	107	11	1,3	51	17	1,3,9,11,33
170	2	0,11,33	106	14	0,1,3	50	34	0,1,3,9,11,33
169	4	17,33	99	17	1,3,33	49	38	1,3,9,17,33
168	4	0,17,33	98	22	0,1,3,33	48	38	0,1,3,9,17,33
161	4	11,17,33	97	11	1,3,17	41	48	1,3,9,11,17,33
160	4	0,11,17,33	96	16	0,1,3,17	40	48	0,1,3,9,11,17,33
147	5	1	91	17	1,3,11,33	27	11	1,3,9,23
146	6	0,1	90	22	0,1,3,11,33	26	22	0,1,3,9,23

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
139	9	3,33	89	24	1,3,17,33	19	17	1,3,9,23,33
138	10	0,3,33	88	24	0,1,3,17,33	18	34	0,1,3,9,23,33
137	6	1,17	81	24	1,3,11,17,33	9	55	1,3,9,17,23,33
136	6	0,1,17	80	24	0,1,3,11,17,33	8	66	0,1,3,9,17,23,33
131	10	1,11,33	67	11	1,3,9			
130	10	0,1,11,33	66	22	0,1,3,9			
$n = 189$, $m(x) = 1100111$, $N_C = 175286$								
187	2	63	125	12	0,1,3,5,31,81	63	24	0,1,3,5,7,11,13,31,39,63,81
186	2	0,63	124	12	1,3,5,7,63,81	62	24	0,1,3,5,7,9,11,13,15,31
185	2	0,81	123	12	0,1,3,5,7,63,81	61	24	1,3,5,7,11,13,21, 23,27,63,81
184	2	63,81	122	12	0,1,3,5,7,9	60	24	0,1,3,5,7,9,11,13,31,39,63
183	2	3	121	12	1,3,5,7,9,63	59	24	0,1,3,5,7,9,11,13,15,31,81
182	2	0,3	120	12	0,1,3,5,7,9,63	58	24	1,3,5,7,11,13,21, 31,63,69,81
181	2	3,63	119	12	0,1,3,5,7,69,81	57	27	1,3,5,7,9,11,13,15,21,23
180	2	0,3,63	118	14	1,3,5,31,39,63,81	56	28	0,1,3,5,7,9,11,13,15,21,23
179	2	0,3,81	117	14	0,1,3,5,31,39,63,81	55	27	1,3,5,7,9,11,13,21,23,39,63
178	2	3,63,81	116	14	0,1,3,5,9,31,39	54	31	1,3,5,7,11,13,21, 23,39,45,81
177	2	3,69	115	14	1,3,5,9,31,39,63	53	32	0,1,3,5,7,11,13,21, 23,39,45,81
176	2	0,3,69	114	14	0,1,3,5,9,31,39,63	52	32	1,3,5,7,11,13,21, 23,39,45,63,81
175	2	3,63,69	113	14	0,1,3,5,31,39,69,81	51	32	0,1,3,5,7,11,13,21, 23,39,45,63,81
174	2	0,3,63,69	112	14	1,3,5,31,39,63,69,81	50	32	0,1,3,5,7,9,11,13, 15,21,23,27,81
173	2	0,3,69,81	111	14	0,1,3,5,31, 39,63,69,81	49	32	1,3,5,7,11,13,21, 23,27,39,63,69,81
172	2	3,63,69,81	110	14	0,1,3,5,9,31,39,69	48	32	0,1,3,5,7,11,13,21,23, 27,39,63,69,81
171	2	3,21,69	109	14	1,3,5,9,31,39,63,69	47	32	0,1,3,5,7,11,13,21, 23,39,45,69,81
170	2	0,3,21,69	108	14	0,1,3,5,9, 31,39,63,69	46	32	1,3,5,7,11,13,21, 23,39,45,63,69,81
169	4	1,63	107	14	0,1,3,11,13, 21,39,69,81	45	32	0,1,3,5,7,11,13,21, 23,39,45,63,69,81
168	4	0,1,63	106	14	1,3,11,13,21, 39,63,69,81	44	32	0,1,3,5,7,9,11, 13,21,23,39,45,69
167	4	0,1,81	105	14	1,3,9,11,13,21,39,45	43	32	1,3,5,7,9,11,13,21,23, 27,39,63,69,81
166	4	1,63,81	104	14	0,1,3,5,7,9,11	42	32	0,1,3,5,7,9,11,13,21, 23,39,45,63,69
165	5	1,3	103	14	1,3,5,7,9,11,63	41	32	0,1,3,5,7,9,11,13,21, 23,39,45,69,81
164	6	0,1,3	102	15	1,3,5,7,11,21,81	40	32	1,3,5,7,9,11,13,21,23, 39,45,63,69,81
163	6	1,3,63	101	16	0,1,3,5,7,11,21,81	39	32	0,1,3,5,7,9,11,13,21, 23,39,45,63,69,81
162	6	0,1,3,63	100	16	1,3,5,7,11,21,63,81	38	32	0,1,3,5,7,9,11,13,15, 21,23,39,45,69
161	6	0,1,3,81	99	17	1,3,7,11,27,31,39,81	37	32	1,3,5,7,9,11,13,15,21, 23,27,39,63,69,81
160	6	1,3,63,81	98	18	0,1,3,7,11, 27,31,39,81	36	33	1,3,5,7,11,13,21, 23,31,39,45,81
159	6	3,9,13	97	18	1,3,7,11,27,31, 39,63,81	35	36	0,1,3,5,7,11,13,23, 31,45,69,81,93
158	6	0,1,3,69	96	18	0,1,3,7,11,27, 31,39,63,81	34	33	1,3,5,7,11,13,23,31, 39,45,63,69,81
157	6	1,3,63,69	95	18	0,1,3,7,11,21, 31,39,81	33	36	0,1,3,5,7,11,13,23,31, 39,45,63,69,81

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
156	6	0,1,3,63,69	94	18	1,3,7,11,21, 31,39,63,81	32	36	0,1,3,5,7,11,13,21, 23,31,39,45,69
155	6	0,1,3,69,81	93	18	3,5,7,9,11,13,39,45	31	36	1,3,5,7,11,13,21,23, 27,31,39,63,69,81
154	6	1,3,63,69,81	92	18	0,1,3,7,9, 11,21,31,39	30	39	1,3,5,7,11,13,21, 23,31,39,45,69,81
153	6	3,9,13,69	91	18	1,3,7,9,11,21, 31,39,63	29	42	0,1,3,5,7,11,13,21, 23,31,39,45,69,81
152	6	0,1,3,21,69	90	18	0,1,3,7,9,11, 21,31,39,63	28	45	1,3,5,7,11,13,21,23, 31,39,45,63,69,81
151	6	1,31,63	89	18	0,1,3,7,11,21, 31,39,69,81	27	48	0,1,3,5,7,11,13,21,23, 31,39,45,63,69,81
150	6	0,1,31,63	88	18	1,3,7,11,21,31, 39,63,69,81	26	42	0,1,3,5,7,9,11,13,21, 23,31,39,45,69
149	6	0,1,5,81	87	18	1,3,7,9,11, 21,31,39,45	25	45	1,3,5,7,9,11,13,21,23, 27,31,39,63,69,81
148	6	1,5,63,81	86	18	0,1,3,7,11,13,31,93	24	48	0,1,3,5,7,9,11,13,21,23, 27,31,39,63,69,81
147	7	1,3,5	85	18	1,3,7,9,11,21, 31,39,63,69	23	48	0,1,3,5,7,9,11,13,21,23, 31,39,45,69,81
146	8	0,1,3,5	84	18	0,1,3,7,11,13, 31,39,63	22	48	1,3,5,7,9,11,13,15,21,23, 31,45,63,69,81
145	8	1,3,5,63	83	18	0,1,3,5,11,13, 21,31,81	21	54	0,1,3,5,7,9,11,13,21,23, 31,39,45,63,81,93
144	8	0,1,3,5,63	82	20	1,3,5,7,11,13, 39,63,81	20	54	0,1,3,5,7,9,11,13,15,21, 23,31,45,69,93
143	8	0,1,3,5,81	81	21	1,3,5,7,9,11,13,15	19	57	1,3,5,7,9,11,13,15,21,23, 27,31,63,69,81,93
142	8	1,3,5,63,81	80	22	0,1,3,5,7,9,11,13,15	18	63	1,3,5,7,9,11,13,15,21, 23,31,39,45,69,81
141	8	1,3,7,39	79	21	1,3,5,7,9,11, 13,39,63	17	66	0,1,3,5,7,9,11,13,15,21, 23,31,39,45,69,81
140	10	0,1,3,7,39	78	22	0,1,3,5,7,9, 11,13,39,63	16	69	1,3,5,7,9,11,13,15,21, 23,31,39,45,63,69,81
139	10	1,3,7,39,63	77	22	0,1,3,5,7,9, 11,13,15,81	15	72	0,1,3,5,7,9,11,13,15,21, 23,31,39,45,63,69,81
138	10	0,1,3,7,39,63	76	24	1,3,5,7,11,13, 21,39,63,81	14	66	0,1,3,5,7,9,11,13,15,21, 23,27,31,39,45,69,81
137	10	0,1,3,7,39,81	75	24	0,1,3,5,7,11, 13,21,39,63,81	13	72	1,3,5,7,9,11,13,15,21,23, 27,31,39,63,69,81,93
136	10	1,3,7,39,63,81	74	24	0,1,3,5,7,9, 11,13,15,21	12	72	0,1,3,5,7,9,11,13,15,21, 23,27,31,33,39,63,69,81
135	10	1,3,9,31,39	73	24	1,3,5,7,9,11, 13,21,39,63	11	78	0,1,3,5,7,9,11,13,15,21, 23,31,33,39,45,69,81
134	10	0,1,3,31,39,69	72	24	0,1,3,5,7,9, 11,13,21,39,63	10	81	1,3,5,7,9,11,13,15,21, 23,31,39,45,63,69,81,93
133	10	1,3,31,39,63,69	71	24	0,1,3,5,7,9, 11,13,15,21,81	9	84	0,1,3,5,7,9,11,13,15,21, 23,31,33,39,45,63,69,81
132	10	0,1,3,31, 39,63,69	70	24	1,3,5,7,11,13, 21,39,63,69,81	8	78	0,1,3,5,7,9,11,13,15, 21,23,27,31,33,39,45,69,81
131	10	0,1,3,31, 39,69,81	69	24	1,3,7,9,11, 13,21,31,39,45	7	93	1,3,5,7,9,11,13,15,21,23, 27,31,33,39,45,63,69,81
130	10	1,3,31,39, 63,69,81	68	24	0,1,3,5,7,9, 11,13,21,39,69	6	96	0,1,3,5,7,9,11,13,15,21,23, 27,31,33,39,45,63,69,81

Continue on next page

Table B.1 (*continued*)

k	d	Roots of $g(x)$	k	d	Roots of $g(x)$	k	d	Roots of $g(x)$
129	10	1,3,9,21,31,45	67	24	1,3,5,7,9,11,13, 21,39,63,69	5	90	0,1,3,5,7,9,11,13,15,21, 23,31,33,39,45,69,81,93
128	10	0,1,3,5,7	66	24	0,1,3,5,7,9,11, 13,21,39,63,69	4	81	1,3,5,7,9,11,13,15,21,23, 31,33,39,45,63,69,81,93
127	10	1,3,5,7,63	65	24	0,1,3,5,7,9,11, 13,21,39,69,81	3	108	0,1,3,5,7,9,11,13,15,21,23, 31,33,39,45,63,69,81,93
126	10	0,1,3,5,7,63	64	24	1,3,5,7,9,11,13, 21,39,63,69,81			

C Improved Lower-Bounds of the Minimum Hamming Distance of Binary Linear Codes

The following tables depict the updated lower-bounds of minimum distance of linear codes over $\mathbb{F}_2$. These improvements–there are 901 of them in total, are due to the new binary linear codes described in Chapter 4. In the following tables, entires marked with C refer to cyclic codes, those marked with X, XX and $Y1$ refer to codes obtained from Constructions X, XX and Y1 respectively. Similarly, entries marked with E, P, and S denote $[n, k, d]$ codes otained by extending (aneexing parity-check bit) to $[n-1, k, d']$ codes, puncturing $[n+1, k, d+1]$ codes and shortening $[n+1, k+1, d]$ codes, respectively. Unmarked entries are the original Brouwer's (1998) lower-bounds.

Table C.1: Updated Minimum Distance Lower Bounds of Linear Codes $C = [n, k]$ for $153 \leq n \leq 174$ and $58 \leq k \leq 77$

$n\backslash k$	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	k/n
153	32	32	32	32	29^P	28	28	27	26	26	26	26	25	24	24	24	24	24	24	22	153
154	32	32	32	32	30^P	28	28	28	27	26	26	26	26	24	24	24	24	24	24	23^X	154
155	32	32	32	32	31^X	28	28	28	28	27	26	26	26	25	24	24	24	24	24	24^E	155
156	32	32	32	32	32^E	28	28	28	28	28	27	26	26	26	25	24	24	24	24	24^E	156
157	32	32	32	32	32^E	29	28	28	28	28	28	26	26	26	26	24	24	24	24	24^E	157
158	32	32	32	32	32^E	30	29	28	28	28	28	26	26	26	26	25	24	24	24	24	158
159	32	32	32	32	32^E	31^{XX}	30	29	28	28	28	27	26	26	26	26	25	24	24	24	159
160	32	32	32	32	32^E	32^E	30	30	28	28	28	28	26	26	26	26	26	25	24	24	160
161	32	32	32	32	32^E	32^E	30	30	29	28	28	28	27	26	26	26	26	26	25	24	161
162	33	32	32	32	32	32^E	31^S	30	30	29	28	28	28	27	26	26	26	26	26	24	162
163	34	33	32	32	32	32	32^S	31^S	30	30	29	28	28	28	27	26	26	26	26	25	163
164	34	34	33	32	32	32	32	32^S	31^S	30	30	29	28	28	28	27	26	26	26	26	164
165	34	34	34	33	32	32	32	32	32^S	31^S	30	30	28	28	28	28	27	26	26	26	165
166	34	34	34	34	32	32	32	32	32^S	32^S	31^S	30	28	28	28	28	28	27	26	26	166
167	34	34	34	34	32	32	32	32	32^S	32^S	32^S	31^P	29	28	28	28	28	28	27	26	167
168	34	34	34	34	32	32	32	32	32^S	32^S	32^S	32^{Y1}	30	29^S	28	28	28	28	28	26	168
169	35^S	34	34	34	32	32	32	32	32^E	32^S	32^S	32^E	31^S	30^S	29^S	28	28	28	28	27	169
170	36^E	35^S	34	34	33	32	32	32	32	32^E	32^S	32^E	32^S	31^S	30^S	29^S	28	28	28	28	170
171	36	36^E	35^X	34	34	33	32	32	32	32	32^E	32^E	32^S	32^S	31^S	30^S	29^S	28	28	28	171
172	36	36	36^E	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	28	28	172
173	36	36	36	35	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	28	173
174	36	36	36	36	34	34	34	34	32	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	174

Table C.2: Updated Minimum Distance Lower Bounds of Linear Codes $C = [n, k]$ for $175 \leq n \leq 224$ and $56 \leq k \leq 78$

$n \backslash k$	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	k/n
175	38	36	36	36	36	36	34	34	34	34	33	32	32	32	32	32^E	32^E	32^S	32^S	32^S	31^S	30^S	29^S	175
176	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	32^S	31^S	30^S	176
177	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	32^S	31^S	177
178	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	32^S	178
179	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	179
180	40	38	38	38	38	36	36	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	180
181	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	181
182	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	182
183	40	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	183
184	41	40	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	184
185	42	41^S	40	40	40	38	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	185
186	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	36	34	34	34	34	34	32	32	32	186
187	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	32	32	187
188	42	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	32	188
189	43	42	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	189
190	44	42	42	42	42	42^S	41^S	40	40	38	38	38	38	38	37^S	36	36	36	35	34	34	34	34	190
191	44	43	42	42	42	42	42^S	41^S	40	39	38	38	38	38	38^S	37^S	36	36	36	35	34	34	34	191
192	44	44	43	42	42	42	42	42^S	41^S	40	39^P	38	38	38	38	38^S	37^S	36	36	36	35^S	34	34	192
193	44	44	44	43	42	42	42	42	42^S	41^S	40^P	39^P	38	38	38	38^S	38^S	37^S	36	36	36^S	35^S	34	193
194	44	44	44	44	43	42	42	42	42	42^S	41^P	40^P	39^P	38	38	38	38^S	38^S	37^P	36	36	36^S	35^P	194
195	44	44	44	44	44	43	42	42	42	42	42^C	41^C	40^C	39^C	38	38	38	38^C	38^C	37^C	36	36	36^C	195
196	44	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40^E	38	38	38	38	38^E	38^E	36	36	36	196
197	45	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40	39	38	38	38	38	38^E	36	36	36	197
198	46	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	36	36	36	198
199	46	45^S	44	44	44	44	43	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	36	36	36	199
200	47^S	46^S	45^S	44	44	44	44	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	37	36	36	200
201	48^S	47^S	46^S	45^S	44	44	44	42	42	42	42	42^E	40	40	40	38	38	38	38	38	38	37	36	201
202	48^S	48^S	47^S	46^S	45^P	44	44	43	42	42	42	42^E	40	40	40	39	38	38	38	38	38	38	37	202
203	48^S	48^S	48^S	47^S	46^P	44	44	44	43	42	42	42^E	40	40	40	40	39	38	38	38	38	38	38	203
204	48^S	48^S	48^S	48^S	47^P	45^P	44	44	44	43	42	42	41	40	40	40	40	39	38	38	38	38	38	204
205	48	48^S	48^S	48^S	48^C	46^C	45^S	44	44	44	42	42	42	41	40	40	40	40	39	38	38	38	38	205
206	48	48^S	48^S	48^S	48^E	46^E	46^S	45^S	44	44	43	42	42	42	41	40	40	40	40	39	38	38	38	206
207	48	48	48^S	48^S	48^E	47^P	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	40	38	38	38	207
208	48	48	48	48^S	48^E	48^X	46	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	39	38	38	208
209	49	48	48	48	48^E	48^E	46	46	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	39	38	209
210	50	48	48	48	48	48^E	47^S	46	46	46^S	46^S	45^S	44	44	43	42	42	42	40	40	40	40	39	210
211	50	49	48	48	48	48^E	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	40	40	211
212	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	40	212
213	50	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	213
214	51	50	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^P	44	44	43	42	42	41	40	214
215	52	50	50	50	50	48	48	48	48	48^S	48^S	47^S	46	46^S	46^C	46^C	44	44	44	43	42	42	40	215
216	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^S	46^E	46^E	44	44	44	44	43	42	41	216
217	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^E	46^E	44	44	44	44	44	43	42	217
218	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^E	45^S	44	44	44	44	44	43	218
219	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^S	45^S	44	44	44	44	44	219
220	54	52	52	52	52	50	50	50	50	48	48	48	48	48^S	48^S	48^S	47^S	46^S	45^P	44	44	44	44	220
221	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^S	47^S	46^P	45^P	44	44	44	221
222	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^S	47^P	46^P	44	44	44	222
223	54	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^C	47^C	44	44	44	223
224	55	54	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^E	48^E	45	44	44	224

Table C.3: Updated Minimum Distance Lower Bounds of Linear Codes $C = [n, k]$ for $175 \leq n \leq 224$ and $79 \leq k \leq 100$

$n \backslash k$	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	k/n
175	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	22	21	175
176	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	22	176
177	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	177
178	31^S	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	178
179	32^S	31^S	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	179
180	32^S	32^S	31^S	30^S	29^S	28	28	28	26	26	26	26	26	24	24	24	24	24	23	22	22	22	180
181	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	23	22	22	181
182	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	23	22	182
183	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	23	183
184	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	184
185	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	185
186	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	26	26	26	26	26	24	24	24	24	186
187	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	24	24	187
188	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	24	188
189	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	189
190	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	190
191	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	191
192	34	34	32	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27^S	26	26	26	192
193	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^S	27^S	26	26	193
194	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^S	27^P	26	194
195	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^P	27^P	195
196	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^P	28^P	196
197	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^P	29^P	197
198	36	36	34	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^P	30^P	198
199	36	36	34	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^C	31^C	199
200	36	36	35	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	200
201	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	201
202	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	202
203	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	203
204	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^E	32^E	204
205	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^E	32^E	205
206	38	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^E	32^E	206
207	38	38	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^E	32^E	207
208	38	38	38	38	38	37	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^E	32^E	208
209	38	38	38	38	38	38	37	36	35	34	34	34	34	32	32	32	32	32	32	32^S	32^E	32^E	209
210	38	38	38	38	38	38	38	37	36	35	34	34	34	32	32	32	32	32	32	32^S	32^E	32^E	210
211	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32	32^S	32^E	32^E	211
212	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^S	32^E	32^E	212
213	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^E	32^E	213
214	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^E	214
215	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	215
216	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	32	32	32	32	216
217	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	32	32	217
218	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	32	218
219	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	219
220	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	220
221	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	221
222	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	222
223	44	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	223
224	44	44	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	224

Table C.4: Updated Minimum Distance Lower Bounds of Linear Codes $C = [n, k]$ for $225 \leq n \leq 256$ and $48 \leq k \leq 62$

$n \backslash k$	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	k/n
225	60	60^S	60^S	60^S	59^S	58^S	57^S	56	56	54	54	54	54	52	52	225
226	60	60	60^S	60^S	60^S	59^S	58^S	57^S	56	55^S	54	54	54	52	52	226
227	60	60	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^S	55^S	54	54	52	52	227
228	61	60	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^S	55^P	54	53^S	52	228
229	62	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^P	54	54^S	53^S	229
230	62	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^P	54	54	54^S	230
231	63	61	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^P	54	54	54	231
232	64	62	60	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^P	54	54	54	232
233	64	62	60	60	60	60^S	60^S	60^S	60^S	60^S	60^C	60^C	54	54	54	233
234	64	62	61	60	60	60	60^S	60^S	60^S	60^S	60^E	60^E	55	54	54	234
235	64	63	62	61	60	60	60	60^S	60^S	60^S	60^E	60^E	56	55	54	235
236	65	64	62	62	61	60	60	60	60^S	60^S	60^E	60^E	56	56	54	236
237	66	64	63	62	62	61	60	60	60	60^S	60^E	60^E	56	56	55	237
238	66	65^P	64	63	62	62	61	60	60	60	60^E	60^E	57	56	56	238
239	67	66^P	64	64	63	62	62	61	60	60	60	60^E	58	57	56	239
240	68	67^P	64	64	64	62	62	62	61	60	60	60	58	58	56	240
241	68	68^C	64	64	64	62	62	62	62	61	60	60	58	58	57	241
242	68	68^E	65	64	64	63^S	62	62	62	62	61	60	59	58	58	242
243	68	68^E	66	65	64	64^S	63^S	62	62	62	62	61	60	59	58	243
244	69	68	66	66	65	64	64^S	63^S	62	62	62	62	61	60	59	244
245	70	68	67	66	66	65^S	64^S	64^S	63^S	62	62	62	62	61	60	245
246	70	68	68	67	66	66^S	65^S	64^S	64^S	63^P	62	62	62	62	61	246
247	71	68	68	68	67	66^S	66^S	65^S	64^S	64^{Y1}	63^{Y1}	62	62	62	62	247
248	72	69^S	68	68	68	66^S	66^S	66^S	65^S	64^E	64^E	62	62	62	62	248
249	72	70^S	69^S	68	68	66^P	66^S	66^S	66^S	65^S	64^E	63^S	62	62	62	249
250	72	71^S	70^S	69^P	68	67^P	66^S	66^S	66^S	66^S	65^S	64^S	63^S	62	62	250
251	73^S	72^S	71^S	70^P	69^S	68^P	67^S	66^P	66^S	66^S	66^S	65^S	64^S	63^S	62	251
252	74^S	73^S	72^S	71^P	70^S	69^P	68^S	67^P	66^S	66^S	66^S	66^S	65^S	64^S	63^P	252
253	74	74^S	73^S	72^P	71^S	70^P	69^S	68^P	67^S	66^P	66^S	66^S	66^S	65^S	64^P	253
254	75^P	74^P	74^S	73^P	72^S	71^P	70^S	69^P	68^S	67^P	66^S	66^S	66^S	66^S	65^P	254
255	76^C	75^C	74^C	74^C	72^C	72^C	70^C	70^C	68^C	68^C	66^C	66^S	66^C	66^S	66^C	255
256	76	76^E	74^E	74^E	72	72^E	70^E	70^E	68	68^E	66^E	66^S	66^E	66^S	66^E	256

Table C.5: Updated Minimum Distance Lower Bounds of Linear Codes $C = [n, k]$ for $225 \leq n \leq 256$ and $63 \leq k \leq 76$

$n \backslash k$	63	64	65	66	67	68	69	70	71	72	73	74	75	76	k/n
225	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	225
226	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	226
227	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	227
228	52	52	50	50	50	50	48	48	48	48	48	48^E	48^E	47^P	228
229	52	52	51	50	50	50	49	48	48	48	48	48^E	48^E	48^C	229
230	53^S	52	52	51	50	50	50	48	48	48	48	48^E	48^E	48^E	230
231	54^S	53^S	52	52	51	50	50	48	48	48	48	48	48^E	48^E	231
232	54	54^S	53^S	52	52	51	50	49	48	48	48	48	48	48^E	232
233	54	54	54^S	53^S	52	52	51	50	49	48	48	48	48	48	233
234	54	54	54	54^S	53^S	52	52	51	50	49	48	48	48	48	234
235	54	54	54	54	54^S	53^S	52	52	51	50	49	48	48	48	235
236	54	54	54	54	54	54^S	53^S	52	52	51	50	49	48	48	236
237	54	54	54	54	54	54	54^S	53^S	52	52	51	50	49	48	237
238	55	54	54	54	54	54	54	54^S	53^S	52	52	51	50	49	238
239	56	55	54	54	54	54	54	54	54^S	53^S	52	52	51	50	239
240	56	56	54	54	54	54	54	54	54	54^S	53^P	52	52	51	240
241	56	56	55	54	54	54	54	54	54	54	54^C	52	52	52	241
242	57	56	56	55	54	54	54	54	54	54	54	53	52	52	242
243	58	57	56	56	55	54	54	54	54	54	54	54	53	52	243
244	58	58	56	56	56	55	54	54	54	54	54	54	54	53	244
245	59	58	57	56	56	56	55	54	54	54	54	54	54	54	245
246	60	59	58	57	56	56	56	55	54	54	54	54	54	54	246
247	61	60	59	58	57	56	56	56	55	54	54	54	54	54	247
248	62	61	60	59	58	57	56	56	56	55	54	54	54	54	248
249	62	62	61	60	59	58	57	56	56	56	55	54	54	54	249
250	62	62	62	61	60	59	58	57	56	56	56	55	54	54	250
251	62	62	62	62	61	60	59	58	57	56	56	56	55	54	251
252	62	62	62	62	62	61	60	59	58	56	56	56	56	55	252
253	63^P	62	62	62	62	62	61	60	59	56	56	56	56	56	253
254	64^P	63^P	62	62	62	62	62	61	60	57	56	56	56	56	254
255	65^C	64^C	63^C	62	62	62	62	62	61	58	57	56	56	56	255
256	66^E	64^E	64^E	62	62	62	62	62	62	58	58	56	56	56	256

D Weight Distributions of Quadratic Double-Circulant Codes and their Modulo Congruence

D.1 Primes +3 Modulo 8

D.1.1 Prime 11

We have $P = \left[\begin{smallmatrix} 1 & 3 \\ 3 & 10 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 10 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(11)$, and the permutations of order 3, 5 and 11 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 10 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 10 & 3 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 10 & 9 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(11)| = 2^2 \cdot 3 \cdot 5 \cdot 11 \cdot = 660$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A_{\mathscr{B}_{11}}^{G_2^0}(z) &= \left(1 + z^{24}\right) + 15 \cdot \left(z^8 + z^{16}\right) + 32 \cdot z^{12} \\
A_{\mathscr{B}_{11}}^{G_4}(z) &= \left(1 + z^{24}\right) + 3 \cdot \left(z^8 + z^{16}\right) + 8 \cdot z^{12} \\
A_{\mathscr{B}_{11}}^{S_3}(z) &= \left(1 + z^{24}\right) + 14 \cdot z^{12} \\
A_{\mathscr{B}_{11}}^{S_5}(z) &= \left(1 + z^{24}\right) + 4 \cdot \left(z^8 + z^{16}\right) + 6 \cdot z^{12} \\
A_{\mathscr{B}_{11}}^{S_{11}}(z) &= \left(1 + z^{24}\right) + 2 \cdot z^{12}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{11}$ and their modular congruence are shown in Table D.1.

Table D.1: Modular congruence weight distributions of $\mathscr{B}_{11}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_5)$ mod 5	$A_i(S_{11})$ mod 11	$A_i(\mathcal{H})$ mod 660	n_i†	A_i
0	1	1	1	1	1	0	1
8	3	0	4	0	99	1	759
12	0	2	1	2	596	3	2576
16	3	0	4	0	99	1	759
24	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{660}$

D.1.2 Prime 19

We have $P = \left[\begin{smallmatrix} 1 & 6 \\ 6 & 18 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 18 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(19)$, and the permutations of order 3, 5 and 19 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 18 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 18 & 4 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 18 & 17 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(19)| = 2^2 \cdot 3^2 \cdot 5 \cdot 19 = 3420$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A^{(G_2^0)}_{\mathscr{B}_{19}}(z) &= (1 + z^{40}) + 5 \cdot (z^8 + z^{32}) + 80 \cdot (z^{12} + z^{28}) + 250 \cdot (z^{16} + z^{24}) + 352 \cdot z^{20} \\
A^{(G_4)}_{\mathscr{B}_{19}}(z) &= (1 + z^{40}) + 1 \cdot (z^8 + z^{32}) + 8 \cdot (z^{12} + z^{28}) + 14 \cdot (z^{16} + z^{24}) + 16 \cdot z^{20} \\
A^{(S_3)}_{\mathscr{B}_{19}}(z) &= (1 + z^{40}) + 6 \cdot (z^8 + z^{32}) + 22 \cdot (z^{12} + z^{28}) + 57 \cdot (z^{16} + z^{24}) + 84 \cdot z^{20} \\
A^{(S_5)}_{\mathscr{B}_{19}}(z) &= (1 + z^{40}) + 14 \cdot z^{20} \\
A^{(S_{19})}_{\mathscr{B}_{19}}(z) &= (1 + z^{40}) + 2 \cdot z^{20}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{19}$ and their modular congruence are shown in Table D.2.

Table D.2: Modular congruence weight distributions of $\mathscr{B}_{19}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3^2	$A_i(S_5)$ mod 5	$A_i(S_{19})$ mod 19	$A_i(\mathcal{H})$ mod 3420	n_i†	A_i
0	1	1	1	1	1	0	1
8	1	6	0	0	285	0	285
12	0	4	0	0	760	6	21280
16	2	3	0	0	570	70	239970
20	0	3	4	2	2244	153	525504
24	2	3	0	0	570	70	239970
28	0	4	0	0	760	6	21280
32	1	6	0	0	285	0	285
40	1	1	1	1	1	0	1

† $n_i = \dfrac{A_i - A_i(\mathcal{H})}{3420}$

D.1.3 Prime 43

We have $P = \left[\begin{smallmatrix} 1 & 16 \\ 16 & 42 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 42 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(43)$, and the permutations of order 3, 7, 11 and 43 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 42 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 42 & 8 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 42 & 4 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 42 & 41 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(43)| = 2^2 \cdot 3 \cdot 7 \cdot 11 \cdot 43 = 39732$$

and the weight enumerator polynomials of the invariant subcodes are

$$
\begin{aligned}
A_{\mathscr{B}_{43}}^{(G_2^0)}(z) &= (1+z^{88}) + 44\cdot(z^{16}+z^{72}) + 1232\cdot(z^{20}+z^{68}) + 10241\cdot(z^{24}+z^{64}) + \\
&\quad 54560\cdot(z^{28}+z^{60}) + 198374\cdot(z^{32}+z^{56}) + 491568\cdot(z^{36}+z^{52}) + \\
&\quad 839916\cdot(z^{40}+z^{48}) + 1002432\cdot z^{44} \\
A_{\mathscr{B}_{43}}^{(G_4)}(z) &= (1+z^{88}) + 32\cdot(z^{20}+z^{68}) + 77\cdot(z^{24}+z^{64}) + 160\cdot(z^{28}+z^{60}) + \\
&\quad 330\cdot(z^{32}+z^{56}) + 480\cdot(z^{36}+z^{52}) + 616\cdot(z^{40}+z^{48}) + 704\cdot z^{44} \\
A_{\mathscr{B}_{43}}^{(S_3)}(z) &= (1+z^{88}) + 7\cdot(z^{16}+z^{72}) + 168\cdot(z^{20}+z^{68}) + 445\cdot(z^{24}+z^{64}) + \\
&\quad 1960\cdot(z^{28}+z^{60}) + 4704\cdot(z^{32}+z^{56}) + 7224\cdot(z^{36}+z^{52}) + \\
&\quad 10843\cdot(z^{40}+z^{48}) + 14832\cdot z^{44} \\
A_{\mathscr{B}_{43}}^{(S_7)}(z) &= (1+z^{88}) + 6\cdot(z^{16}+z^{72}) + 16\cdot(z^{24}+z^{64}) + 6\cdot(z^{28}+z^{60}) + \\
&\quad 9\cdot(z^{32}+z^{56}) + 48\cdot(z^{36}+z^{52}) + 84\cdot z^{44} \\
A_{\mathscr{B}_{43}}^{(S_{11})}(z) &= (1+z^{88}) + 14\cdot z^{44} \\
A_{\mathscr{B}_{43}}^{(S_{43})}(z) &= (1+z^{88}) + 2\cdot z^{44}.
\end{aligned}
$$

The weight distributions of $\mathscr{B}_{43}$ and their modular congruence are shown in Table D.3.

Table D.3: Modular congruence weight distributions of $\mathscr{B}_{43}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_7)$ mod 7	$A_i(S_{11})$ mod 11	$A_i(S_{43})$ mod 43	$A_i(\mathcal{H})$ mod 39732	n_i†	A_i
0	1	1	1	1	1	1	0	1
16	0	1	6	0	0	32164	0	32164
20	0	0	0	0	0	0	176	6992832
24	1	1	2	0	0	25069	13483	535731625
28	0	1	6	0	0	32164	418387	16623384448
32	2	0	2	0	0	8514	5673683	225426781470
36	0	0	6	0	0	5676	35376793	1405590745152
40	0	1	0	0	0	26488	104797219	4163803131796
44	0	0	0	3	2	28812	150211729	5968212445440
48	0	1	0	0	0	26488	104797219	4163803131796
52	0	0	6	0	0	5676	35376793	1405590745152
56	2	0	2	0	0	8514	5673683	225426781470
60	0	1	6	0	0	32164	418387	16623384448
64	1	1	2	0	0	25069	13483	535731625
68	0	0	0	0	0	0	176	6992832
72	0	1	6	0	0	32164	0	32164
88	1	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{39732}$

D.1.4 Prime 59

We have $P = \left[\begin{smallmatrix} 1 & 23 \\ 23 & 58 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 58 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(59)$, and the permutations of order 3, 5, 29 and 59 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 58 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 58 & 25 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 58 & 3 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 58 & 57 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(59)| = 2^2 \cdot 3 \cdot 5 \cdot 29 \cdot 59 = 102660$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A_{\mathscr{B}_{59}}^{(G_2^0)}(z) &= (1 + z^{120}) + 90 \cdot (z^{20} + z^{100}) + 2555 \cdot (z^{24} + z^{96}) + \\
&\quad 32700 \cdot (z^{28} + z^{92}) + 278865 \cdot (z^{32} + z^{88}) + 1721810 \cdot (z^{36} + z^{84}) + \\
&\quad 7807800 \cdot (z^{40} + z^{80}) + 26366160 \cdot (z^{44} + z^{76}) + 67152520 \cdot (z^{48} + z^{72}) + \\
&\quad 130171860 \cdot (z^{52} + z^{68}) + 193193715 \cdot (z^{56} + z^{64}) + 220285672 \cdot z^{60} \\
A_{\mathscr{B}_{59}}^{(G_4)}(z) &= (1 + z^{120}) + 6 \cdot (z^{20} + z^{100}) + 19 \cdot (z^{24} + z^{96}) + 132 \cdot (z^{28} + z^{92}) + \\
&\quad 393 \cdot (z^{32} + z^{88}) + 878 \cdot (z^{36} + z^{84}) + 1848 \cdot (z^{40} + z^{80}) + 3312 \cdot (z^{44} + z^{76}) + \\
&\quad 5192 \cdot (z^{48} + z^{72}) + 7308 \cdot (z^{52} + z^{68}) + 8931 \cdot (z^{56} + z^{64}) + 9496 \cdot z^{60} \\
A_{\mathscr{B}_{59}}^{(S_3)}(z) &= (1 + z^{120}) + 285 \cdot (z^{24} + z^{96}) + 21280 \cdot (z^{36} + z^{84}) + \\
&\quad 239970 \cdot (z^{48} + z^{72}) + 525504 \cdot z^{60} \\
A_{\mathscr{B}_{59}}^{(S_5)}(z) &= (1 + z^{120}) + 12 \cdot (z^{20} + z^{100}) + 711 \cdot (z^{40} + z^{80}) + 2648 \cdot z^{60} \\
A_{\mathscr{B}_{59}}^{(S_{29})}(z) &= (1 + z^{120}) + 4 \cdot (z^{32} + z^{88}) + 6 \cdot z^{60} \\
A_{\mathscr{B}_{59}}^{(S_{59})}(z) &= (1 + z^{120}) + 2 \cdot z^{60}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{59}$ and their modular congruence are shown in Table D.4.

Table D.4: Modular congruence weight distributions of $\mathscr{B}_{59}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_5)$ mod 5	$A_i(S_{29})$ mod 29	$A_i(S_{59})$ mod 59	$A_i(\mathcal{H})$ mod 102660	n_i†	A_i
0	1	1	1	1	1	1	0	1
20	2	0	2	0	0	71862	0	71862
24	3	0	0	0	0	76995	372	38266515
28	0	0	0	0	0	0	59565	6114942900
32	1	0	0	4	0	32745	4632400	475562216745
36	2	1	0	0	0	17110	183370922	18824858869630
40	0	0	1	0	0	61596	3871511775	397449398883096
44	0	0	0	0	0	0	45105349212	4630515150103920
48	0	0	0	0	0	0	297404962554	30531593455793640
52	0	0	0	0	0	0	1130177151411	116023986363853260
56	3	0	0	0	0	76995	2505920073120	257257754706576195
60	0	0	3	6	2	85788	3265149944551	335200293307691448
64	3	0	0	0	0	76995	2505920073120	257257754706576195
68	0	0	0	0	0	0	1130177151411	116023986363853260
72	0	0	0	0	0	0	297404962554	30531593455793640
76	0	0	0	0	0	0	45105349212	4630515150103920
80	0	0	1	0	0	61596	3871511775	397449398883096
84	2	1	0	0	0	17110	183370922	18824858869630
88	1	0	0	4	0	32745	4632400	475562216745
92	0	0	0	0	0	0	59565	6114942900
96	3	0	0	0	0	76995	372	38266515
100	2	0	2	0	0	71862	0	71862
120	1	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{102660}$

D.1.5 Prime 67

We have $P = \left[\begin{smallmatrix} 1 & 20 \\ 20 & 66 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 66 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(67)$, and the permutations of order 3, 11, 17 and 67 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 66 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 66 & 17 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 66 & 4 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 66 & 65 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(67)| = 2^2 \cdot 3 \cdot 11 \cdot 17 \cdot 67 = 150348$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A_{\mathscr{B}_{67}}^{(C_2^0)}(z) = {} & (1 + z^{136}) + 578 \cdot (z^{24} + z^{112}) + 14688 \cdot (z^{28} + z^{108}) + \\
& 173247 \cdot (z^{32} + z^{104}) + 1480768 \cdot (z^{36} + z^{100}) + 9551297 \cdot (z^{40} + z^{96}) + \\
& 46687712 \cdot (z^{44} + z^{92}) + 175068210 \cdot (z^{48} + z^{88}) + 509510400 \cdot (z^{52} + z^{84}) + \\
& 1160576876 \cdot (z^{56} + z^{80}) + 2081112256 \cdot (z^{60} + z^{76}) + 2949597087 \cdot (z^{64} + z^{72}) + \\
& 3312322944 \cdot z^{68} \\
A_{\mathscr{B}_{67}}^{(C_4)}(z) = {} & (1 + z^{136}) + 18 \cdot (z^{24} + z^{112}) + 88 \cdot (z^{28} + z^{108}) + 271 \cdot (z^{32} + z^{104}) + \\
& 816 \cdot (z^{36} + z^{100}) + 2001 \cdot (z^{40} + z^{96}) + 4344 \cdot (z^{44} + z^{92}) + \\
& 8386 \cdot (z^{48} + z^{88}) + 14144 \cdot (z^{52} + z^{84}) + 21260 \cdot (z^{56} + z^{80}) + \\
& 28336 \cdot (z^{60} + z^{76}) + 33599 \cdot (z^{64} + z^{72}) + 35616 \cdot z^{68} \\
A_{\mathscr{B}_{67}}^{(S_3)}(z) = {} & (1 + z^{136}) + 66 \cdot (z^{24} + z^{112}) + 682 \cdot (z^{28} + z^{108}) + 3696 \cdot (z^{32} + z^{104}) + \\
& 12390 \cdot (z^{36} + z^{100}) + 54747 \cdot (z^{40} + z^{96}) + 163680 \cdot (z^{44} + z^{92}) + \\
& 318516 \cdot (z^{48} + z^{88}) + 753522 \cdot (z^{52} + z^{84}) + 1474704 \cdot (z^{56} + z^{80}) + \\
& 1763454 \cdot (z^{60} + z^{76}) + 2339502 \cdot (z^{64} + z^{72}) + 3007296 \cdot z^{68} \\
A_{\mathscr{B}_{67}}^{(S_{11})}(z) = {} & (1 + z^{136}) + 6 \cdot (z^{24} + z^{112}) + 16 \cdot (z^{36} + z^{100}) + 6 \cdot (z^{44} + z^{92}) + \\
& 9 \cdot (z^{48} + z^{88}) + 48 \cdot (z^{56} + z^{80}) + 84 \cdot z^{68} \\
A_{\mathscr{B}_{67}}^{(S_{17})}(z) = {} & (1 + z^{136}) + 14 \cdot z^{68} \\
A_{\mathscr{B}_{67}}^{(S_{67})}(z) = {} & (1 + z^{136}) + 2 \cdot z^{68}
\end{aligned}$$

The weight distributions of $\mathscr{B}_{67}$ and their modular congruence are shown in Table D.5.

Table D.5: Modular congruence weight distributions of $\mathscr{B}_{67}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_{11})$ mod 11	$A_i(S_{17})$ mod 17	$A_i(S_{67})$ mod 67	$A_i(\mathcal{H})$ mod 150348	n_i †	A_i
0	1	1	1	1	1	1	0	1
24	2	0	6	0	0	88842	26	3997890
28	0	1	0	0	0	50116	8173	1228844320
32	3	0	0	0	0	37587	1217081	182985731775
36	0	0	5	0	0	136680	95005682	14283914414016
40	1	0	0	0	0	112761	4076381478	612875802567105
44	0	0	6	0	0	13668	99752935189	14997654299809440
48	2	0	9	0	0	20502	1432445445981	215365307912371890
52	0	0	0	0	0	0	12338369112000	1855049119250976000
56	0	0	4	0	0	109344	64817708364545	9745212817192721004
60	0	0	0	0	0	0	210227711554224	31607315976754469952
64	3	0	0	0	0	37587	424499666112161	63822675800631219615
68	0	0	7	14	2	138156	536258660836183	80625417139398579840
72	3	0	0	0	0	37587	424499666112161	63822675800631219615
76	0	0	0	0	0	0	210227711554224	31607315976754469952
80	0	0	4	0	0	109344	64817708364545	9745212817192721004
84	0	0	0	0	0	0	12338369112000	1855049119250976000
88	2	0	9	0	0	20502	1432445445981	215365307912371890
92	0	0	6	0	0	13668	99752935189	14997654299809440
96	1	0	0	0	0	112761	4076381478	612875802567105
100	0	0	5	0	0	136680	95005682	14283914414016
104	3	0	0	0	0	37587	1217081	182985731775
108	0	1	0	0	0	50116	8173	1228844320
112	2	0	6	0	0	88842	26	3997890
136	1	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{150348}$

D.1.6 Prime 83

We have $P = \left[\begin{smallmatrix} 1 & 9 \\ 9 & 82 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 82 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(83)$, and the permutations of order 3, 7, 41 and 83 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 10 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 4 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 82 & 81 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(83)| = 2^2 \cdot 3 \cdot 7 \cdot 41 \cdot 83 \cdot = 285852$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A^{(G_2^0)}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 196 \cdot (z^{24} + z^{144}) + 1050 \cdot (z^{28} + z^{140}) + \\
& 29232 \cdot (z^{32} + z^{136}) + 443156 \cdot (z^{36} + z^{132}) + \\
& 4866477 \cdot (z^{40} + z^{128}) + 42512190 \cdot (z^{44} + z^{124}) + \\
& 292033644 \cdot (z^{48} + z^{120}) + 1590338568 \cdot (z^{52} + z^{116}) + \\
& 6952198884 \cdot (z^{56} + z^{112}) + 24612232106 \cdot (z^{60} + z^{108}) + \\
& 71013075210 \cdot (z^{64} + z^{104}) + 167850453036 \cdot (z^{68} + z^{100}) + \\
& 326369180312 \cdot (z^{72} + z^{96}) + 523672883454 \cdot (z^{76} + z^{92}) + \\
& 694880243820 \cdot (z^{80} + z^{88}) + 763485528432 \cdot z^{84} \\
A^{(G_4)}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 4 \cdot (z^{24} + z^{144}) + 6 \cdot (z^{28} + z^{140}) + \\
& 96 \cdot (z^{32} + z^{136}) + 532 \cdot (z^{36} + z^{132}) + 1437 \cdot (z^{40} + z^{128}) + \\
& 3810 \cdot (z^{44} + z^{124}) + 10572 \cdot (z^{48} + z^{120}) + 24456 \cdot (z^{52} + z^{116}) + \\
& 50244 \cdot (z^{56} + z^{112}) + 95030 \cdot (z^{60} + z^{108}) + 158874 \cdot (z^{64} + z^{104}) + \\
& 241452 \cdot (z^{68} + z^{100}) + 337640 \cdot (z^{72} + z^{96}) + 425442 \cdot (z^{76} + z^{92}) + \\
& 489708 \cdot (z^{80} + z^{88}) + 515696 \cdot z^{84} \\
A^{(S_3)}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 63 \cdot (z^{24} + z^{144}) + 8568 \cdot (z^{36} + z^{132}) + 617085 \cdot (z^{48} + z^{120}) + \\
& 11720352 \cdot (z^{60} + z^{108}) + 64866627 \cdot (z^{72} + z^{96}) + 114010064 \cdot z^{84} \\
A^{(S_7)}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 759 \cdot (z^{56} + z^{112}) + 2576 \cdot z^{84} \\
A^{(S_{41})}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 4 \cdot (z^{44} + z^{124}) + 6 \cdot z^{84} \\
A^{(S_{83})}_{\mathscr{B}_{83}}(z) = {} & (1 + z^{168}) + 2 \cdot z^{84}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{83}$ and their modular congruence are shown in Table D.6.

Table D.6: Modular congruence weight distributions of $\mathscr{B}_{83}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_7)$ mod 7	$A_i(S_{41})$ mod 41	$A_i(S_{83})$ mod 83	$A_i(\mathcal{H})$ mod 285852	n_i†	A_i
0	1	1	1	1	0	6889	0	1
24	0	0	0	0	0	0	2	571704
28	2	0	0	0	0	142926	59	17008194
32	0	0	0	0	0	0	19267	5507510484
36	0	0	0	0	0	0	4382043	1252615755636
40	1	0	0	0	0	214389	580925895	166058829151929
44	2	0	0	4	0	156870	45643181220	13047194638256310
48	0	0	0	0	0	0	2200608997142	629048483051034984
52	0	0	0	0	0	0	66772769854878	19087129808556586056
56	0	0	3	0	0	81672	1301721510043764	372099697089030108600
60	2	0	0	0	0	142926	16579528883596695	4739291490433882602066
64	2	0	0	0	0	142926	139840453892634544	39973673426117369814414
68	0	0	0	0	0	0	789557804450518101	225696677517789500207052
72	0	0	0	0	0	0	3009393355026378047	860241109321000217491044
76	2	0	0	0	0	142926	7792111592501736790	2227390682939806465038006
80	0	0	0	0	0	0	13766213240696824038	3935099587279668544910376
84	0	2	0	6	0	211484	16637096860279621422	4755747411704650343205104
88	0	0	0	0	0	0	13766213240696824038	3935099587279668544910376
92	2	0	0	0	0	142926	7792111592501736790	2227390682939806465038006
96	0	0	0	0	0	0	3009393355026378047	860241109321000217491044
100	0	0	0	0	0	0	789557804450518101	225696677517789500207052
104	2	0	0	0	0	142926	139840453892634544	39973673426117369814414
108	2	0	0	0	0	142926	16579528883596695	4739291490433882602066
112	0	0	3	0	0	81672	1301721510043764	372099697089030108600
116	0	0	0	0	0	0	66772769854878	19087129808556586056

Continued on next page

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_7)$ mod 7	$A_i(S_{41})$ mod 41	$A_i(S_{83})$ mod 83	$A_i(\mathcal{H})$ mod 285852	n_i †	A_i
120	0	0	0	0	0	0	2200608997142	629048483051034984
124	2	0	0	4	0	156870	45643181220	13047194638256310
128	1	0	0	0	0	214389	580925895	166058829151929
132	0	0	0	0	0	0	4382043	1252615755636
136	0	0	0	0	0	0	19267	5507510484
140	2	0	0	0	0	142926	59	17008194
144	0	0	0	0	0	0	2	571704
168	1	1	1	1	0	6889	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{285852}$

D.2 Primes −3 Modulo 8

D.2.1 Prime 13

We have $P = \left[\begin{smallmatrix} 3 & 4 \\ 4 & 10 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 12 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(13)$, and the permutations of order 3, 7 and 13 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 12 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 12 & 3 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 12 & 11 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(13)| = 2^2 \cdot 3 \cdot 7 \cdot 13 = 1092$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A^{G_2^0}_{\mathscr{B}_{13}}(z) &= (1 + z^{28}) + 26 \cdot (z^8 + z^{20}) + 32 \cdot (z^{10} + z^{18}) + 37 \cdot (z^{12} + z^{16}) + 64 \cdot z^{14} \\
A^{G_4}_{\mathscr{B}_{13}}(z) &= (1 + z^{28}) + 10 \cdot (z^8 + z^{20}) + 8 \cdot (z^{10} + z^{18}) + 5 \cdot (z^{12} + z^{16}) + 16 \cdot z^{14} \\
A^{S_3}_{\mathscr{B}_{13}}(z) &= (1 + z^{28}) + 6 \cdot (z^8 + z^{20}) + 10 \cdot (z^{10} + z^{18}) + 9 \cdot (z^{12} + z^{16}) + 12 \cdot z^{14} \\
A^{S_7}_{\mathscr{B}_{13}}(z) &= (1 + z^{28}) + 2 \cdot z^{14} \\
A^{S_{13}}_{\mathscr{B}_{13}}(z) &= (1 + z^{28}) + 2 \cdot z^{14}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{13}$ and their modular congruence are shown in Table D.7.

Table D.7: Modular congruence weight distributions of $\mathscr{B}_{13}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_7)$ mod 7	$A_i(S_{13})$ mod 13	$A_i(\mathcal{H})$ mod 1092	n_i†	A_i
0	1	1	1	1	1	0	1
8	2	0	0	0	546	0	546
10	0	1	0	0	364	1	1456
12	1	0	0	0	273	3	3549
14	0	0	2	2	912	4	5280
16	1	0	0	0	273	3	3549
18	0	1	0	0	364	1	1456
20	2	0	0	0	546	0	546
28	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{1092}$

D.2.2 Prime 29

We have $P = \left[\begin{smallmatrix} 2 & 13 \\ 13 & 27 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 28 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(29)$, and the permutations of order 3, 5, 7 and 29 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 28 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 28 & 5 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 28 & 3 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 28 & 27 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(29)| = 2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 29 = 12180$$

and the weight enumerator polynomials of the invariant subcodes are

$$
\begin{aligned}
A_{\mathscr{B}_{29}}^{(G_2^0)}(z) &= (1+z^{60}) + 28\cdot(z^{12}+z^{48}) + 112\cdot(z^{14}+z^{46}) + 394\cdot(z^{16}+z^{44}) + \\
&\quad 1024\cdot(z^{18}+z^{42}) + 1708\cdot(z^{20}+z^{40}) + 3136\cdot(z^{22}+z^{38}) + 5516\cdot(z^{24}+z^{36}) + \\
&\quad 7168\cdot(z^{26}+z^{34}) + 8737\cdot(z^{28}+z^{32}) + 9888\cdot z^{30} \\
A_{\mathscr{B}_{29}}^{(G_4)}(z) &= (1+z^{60}) + 12\cdot(z^{14}+z^{46}) + 30\cdot(z^{16}+z^{44}) + 32\cdot(z^{18}+z^{42}) + \\
&\quad 60\cdot(z^{20}+z^{40}) + 48\cdot(z^{22}+z^{38}) + 60\cdot(z^{24}+z^{36}) + 96\cdot(z^{26}+z^{34}) + \\
&\quad 105\cdot(z^{28}+z^{32}) + 136\cdot z^{30} \\
A_{\mathscr{B}_{29}}^{(S_3)}(z) &= (1+z^{60}) + 10\cdot(z^{12}+z^{48}) + 70\cdot(z^{18}+z^{42}) + 245\cdot(z^{24}+z^{36}) + 372\cdot z^{30} \\
A_{\mathscr{B}_{29}}^{(S_5)}(z) &= (1+z^{60}) + 15\cdot(z^{20}+z^{40}) + 32\cdot z^{30} \\
A_{\mathscr{B}_{29}}^{(S_7)}(z) &= (1+z^{60}) + 6\cdot(z^{16}+z^{44}) + 2\cdot(z^{18}+z^{42}) + 8\cdot(z^{22}+z^{38}) + 8\cdot(z^{24}+z^{36}) + \\
&\quad 1\cdot(z^{28}+z^{32}) + 12\cdot z^{30} \\
A_{\mathscr{B}_{29}}^{(S_{29})}(z) &= (1+z^{60}) + 2\cdot z^{30}.
\end{aligned}
$$

The weight distributions of $\mathscr{B}_{29}$ and their modular congruence are shown in Table D.8.

Table D.8: Modular congruence weight distributions of $\mathscr{B}_{29}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_5)$ mod 5	$A_i(S_7)$ mod 7	$A_i(S_{29})$ mod 29	$A_i(\mathcal{H})$ mod 12180	n_i†	A_i
0	1	1	1	1	1	1	0	1
12	0	1	0	0	0	4060	0	4060
14	0	0	0	0	0	0	2	24360
16	2	0	0	6	0	2610	24	294930
18	0	1	0	2	0	11020	141	1728400
20	0	0	0	0	0	0	637	7758660
22	0	0	0	1	0	3480	2162	26336640
24	0	2	0	1	0	11600	5533	67403540
26	0	0	0	0	0	0	10668	129936240
28	1	0	0	1	0	6525	15843	192974265
30	0	0	2	5	2	8412	18129	220819632
32	1	0	0	1	0	6525	15843	192974265
34	0	0	0	0	0	0	10668	129936240
36	0	2	0	1	0	11600	5533	67403540
38	0	0	0	1	0	3480	2162	26336640
40	0	0	0	0	0	0	637	7758660
42	0	1	0	2	0	11020	141	1728400
44	2	0	0	6	0	2610	24	294930
46	0	0	0	0	0	0	2	24360
48	0	1	0	0	0	4060	0	4060
60	1	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{12180}$

D.2.3 Prime 53

We have $P = \left[\begin{smallmatrix} 3 & 19 \\ 19 & 50 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 52 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(53)$, and the permutations of order 3, 13 and 53 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 52 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 52 & 8 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 52 & 51 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(53)| = 2^2 \cdot 3^3 \cdot 13 \cdot 53 = 74412$$

and the weight enumerator polynomials of the invariant subcodes are

$$\begin{aligned}
A^{(G_2^0)}_{\mathscr{B}_{53}}(z) = {} & (1 + z^{108}) + 234 \cdot (z^{20} + z^{88}) + 1768 \cdot (z^{22} + z^{86}) + 5655 \cdot (z^{24} + z^{84}) + \\
& 16328 \cdot (z^{26} + z^{82}) + 47335 \cdot (z^{28} + z^{80}) + 127896 \cdot (z^{30} + z^{78}) + \\
& 316043 \cdot (z^{32} + z^{76}) + 705848 \cdot (z^{34} + z^{74}) + 1442883 \cdot (z^{36} + z^{72}) + \\
& 2728336 \cdot (z^{38} + z^{70}) + 4786873 \cdot (z^{40} + z^{68}) + 7768488 \cdot (z^{42} + z^{66}) + \\
& 11636144 \cdot (z^{44} + z^{64}) + 16175848 \cdot (z^{46} + z^{62}) + 20897565 \cdot (z^{48} + z^{60}) + \\
& 25055576 \cdot (z^{50} + z^{58}) + 27976131 \cdot (z^{52} + z^{56}) + 29057552 \cdot z^{54} \\
A^{(G_4)}_{\mathscr{B}_{53}}(z) = {} & (1 + z^{108}) + 12 \cdot (z^{20} + z^{88}) + 12 \cdot (z^{22} + z^{86}) + 77 \cdot (z^{24} + z^{84}) + \\
& 108 \cdot (z^{26} + z^{82}) + 243 \cdot (z^{28} + z^{80}) + 296 \cdot (z^{30} + z^{78}) + 543 \cdot (z^{32} + z^{76}) + \\
& 612 \cdot (z^{34} + z^{74}) + 1127 \cdot (z^{36} + z^{72}) + 1440 \cdot (z^{38} + z^{70}) + 2037 \cdot (z^{40} + z^{68}) + \\
& 2636 \cdot (z^{42} + z^{66}) + 3180 \cdot (z^{44} + z^{64}) + 3672 \cdot (z^{46} + z^{62}) + 4289 \cdot (z^{48} + z^{60}) + \\
& 4836 \cdot (z^{50} + z^{58}) + 4875 \cdot (z^{52} + z^{56}) + 5544 \cdot z^{54} \\
A^{(S_3)}_{\mathscr{B}_{53}}(z) = {} & (1 + z^{108}) + 234 \cdot (z^{24} + z^{84}) + 1962 \cdot (z^{30} + z^{78}) + 9672 \cdot (z^{36} + z^{72}) + \\
& 28728 \cdot (z^{42} + z^{66}) + 55629 \cdot (z^{48} + z^{60}) + 69692 \cdot z^{54} \\
A^{(S_{13})}_{\mathscr{B}_{53}}(z) = {} & (1 + z^{108}) + 6 \cdot (z^{28} + z^{80}) + 2 \cdot (z^{30} + z^{78}) + 8 \cdot (z^{40} + z^{68}) + \\
& 8 \cdot (z^{42} + z^{66}) + 1 \cdot (z^{52} + z^{56}) + 12 \cdot z^{54} \\
A^{(S_{53})}_{\mathscr{B}_{53}}(z) = {} & (1 + z^{108}) + 2 \cdot z^{54}.
\end{aligned}$$

The weight distributions of $\mathscr{B}_{53}$ and their modular congruence are shown in Table D.9.

Table D.9: Modular congruence weight distributions of $\mathscr{B}_{53}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3^3	$A_i(S_{13})$ mod 13	$A_i(S_{53})$ mod 53	$A_i(\mathcal{H})$ mod 74412	n_i †	A_i
0	1	1	1	1	1	0	1
20	2	0	0	0	37206	3	260442
22	0	0	0	0	0	78	5804136
24	3	18	0	0	43407	1000	74455407
26	0	0	0	0	0	10034	746650008
28	3	0	6	0	64395	91060	6776021115
30	0	18	2	0	64872	658342	48988609776
32	3	0	0	0	18603	3981207	296249593887
34	0	0	0	0	0	20237958	1505946930696

Continued on next page

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3^3	$A_i(S_{13})$ mod 13	$A_i(S_{53})$ mod 53	$A_i(\mathcal{H})$ mod 74412	n_i†	A_i
36	3	6	0	0	26871	86771673	6456853758147
38	0	0	0	0	0	315441840	23472658198080
40	1	0	8	0	67257	976699540	72678166237737
42	0	0	8	0	11448	2584166840	192293022909528
44	0	0	0	0	0	5859307669	436002802265628
46	0	0	0	0	0	11412955404	849260837522448
48	1	9	0	0	31005	19133084721	1423731100290057
50	0	0	0	0	0	27645086470	2057126174405640
52	3	0	1	0	1431	34462554487	2564427604488075
54	0	5	12	2	55652	37087868793	2759782492680368
56	3	0	1	0	1431	34462554487	2564427604488075
58	0	0	0	0	0	27645086470	2057126174405640
60	1	9	0	0	31005	19133084721	1423731100290057
62	0	0	0	0	0	11412955404	849260837522448
64	0	0	0	0	0	5859307669	436002802265628
66	0	0	8	0	11448	2584166840	192293022909528
68	1	0	8	0	67257	976699540	72678166237737
70	0	0	0	0	0	315441840	23472658198080
72	3	6	0	0	26871	86771673	6456853758147
74	0	0	0	0	0	20237958	1505946930696
76	3	0	0	0	18603	3981207	296249593887
78	0	18	2	0	64872	658342	48988609776
80	3	0	6	0	64395	91060	6776021115
82	0	0	0	0	0	10034	746650008
84	3	18	0	0	43407	1000	74455407
86	0	0	0	0	0	78	5804136
88	2	0	0	0	37206	3	260442
108	1	1	1	1	1	0	1

† $n_i = \frac{A_i - A_i(\mathcal{H})}{74412}$

D.2.4 Prime 61

We have $P = \left[\begin{smallmatrix} 2 & 19 \\ 19 & 59 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 60 \\ 1 & 0 \end{smallmatrix}\right]$, $P, T \in \mathrm{PSL}_2(61)$, and the permutations of order 3, 5, 31 and 61 are generated by $\left[\begin{smallmatrix} 0 & 1 \\ 60 & 1 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 60 & 17 \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} 0 & 1 \\ 60 & 5 \end{smallmatrix}\right]$, and $\left[\begin{smallmatrix} 0 & 1 \\ 60 & 59 \end{smallmatrix}\right]$ respectively. In addition,

$$|\mathrm{PSL}_2(61)| = 2^2 \cdot 3 \cdot 5 \cdot 31 \cdot 61 \cdot = 113460$$

and the weight enumerator polynomials of the invariant subcodes are

$$
\begin{aligned}
A_{\mathscr{B}_{61}}^{(G_2^0)} &= (1+z^{124}) + 208\cdot(z^{20}+z^{104}) + 400\cdot(z^{22}+z^{102}) + 1930\cdot(z^{24}+z^{100}) + \\
&\quad 8180\cdot(z^{26}+z^{98}) + 26430\cdot(z^{28}+z^{96}) + 84936\cdot(z^{30}+z^{94}) + 253572\cdot(z^{32}+z^{92}) + \\
&\quad 696468\cdot(z^{34}+z^{90}) + 1725330\cdot(z^{36}+z^{88}) + 3972240\cdot(z^{38}+z^{86}) + \\
&\quad 8585008\cdot(z^{40}+z^{84}) + 17159632\cdot(z^{42}+z^{82}) + 31929532\cdot(z^{44}+z^{80}) + \\
&\quad 55569120\cdot(z^{46}+z^{78}) + 90336940\cdot(z^{48}+z^{76}) + 137329552\cdot(z^{50}+z^{74}) + \\
&\quad 195328240\cdot(z^{52}+z^{72}) + 260435936\cdot(z^{54}+z^{70}) + 325698420\cdot(z^{56}+z^{68}) + \\
&\quad 381677080\cdot(z^{58}+z^{66}) + 419856213\cdot(z^{60}+z^{64}) + 433616560\cdot z^{62} \\
A_{\mathscr{B}_{61}}^{(G_4)} &= (1+z^{124}) + 12\cdot(z^{20}+z^{104}) + 12\cdot(z^{22}+z^{102}) + 36\cdot(z^{24}+z^{100}) + \\
&\quad 40\cdot(z^{26}+z^{98}) + 140\cdot(z^{28}+z^{96}) + 176\cdot(z^{30}+z^{94}) + 498\cdot(z^{32}+z^{92}) + \\
&\quad 576\cdot(z^{34}+z^{90}) + 1340\cdot(z^{36}+z^{88}) + 1580\cdot(z^{38}+z^{86}) + 2660\cdot(z^{40}+z^{84}) + \\
&\quad 3432\cdot(z^{42}+z^{82}) + 4932\cdot(z^{44}+z^{80}) + 6368\cdot(z^{46}+z^{78}) + 8820\cdot(z^{48}+z^{76}) + \\
&\quad 10424\cdot(z^{50}+z^{74}) + 12752\cdot(z^{52}+z^{72}) + 14536\cdot(z^{54}+z^{70}) + 15840\cdot(z^{56}+z^{68}) + \\
&\quad 18296\cdot(z^{58}+z^{66}) + 18505\cdot(z^{60}+z^{64}) + 20192\cdot z^{62} \\
A_{\mathscr{B}_{61}}^{(S_3)} &= (1+z^{124}) + 30\cdot(z^{20}+z^{104}) + 10\cdot(z^{22}+z^{102}) + 50\cdot(z^{24}+z^{100}) + 200\cdot(z^{26}+z^{98}) + \\
&\quad 620\cdot(z^{28}+z^{96}) + 960\cdot(z^{30}+z^{94}) + 2416\cdot(z^{32}+z^{92}) + 4992\cdot(z^{34}+z^{90}) + \\
&\quad 6945\cdot(z^{36}+z^{88}) + 15340\cdot(z^{38}+z^{86}) + 25085\cdot(z^{40}+z^{84}) + 34920\cdot(z^{42}+z^{82}) + \\
&\quad 68700\cdot(z^{44}+z^{80}) + 87548\cdot(z^{46}+z^{78}) + 104513\cdot(z^{48}+z^{76}) + 177800\cdot(z^{50}+z^{74}) + \\
&\quad 201440\cdot(z^{52}+z^{72}) + 225290\cdot(z^{54}+z^{70}) + 322070\cdot(z^{56}+z^{68}) + 301640\cdot(z^{58}+z^{66}) + \\
&\quad 316706\cdot(z^{60}+z^{64}) + 399752\cdot z^{62} \\
A_{\mathscr{B}_{61}}^{(S_5)} &= (1+z^{124}) + 3\cdot(z^{20}+z^{104}) + 24\cdot(z^{26}+z^{98}) + 48\cdot(z^{28}+z^{96}) + \\
&\quad 6\cdot(z^{30}+z^{94}) + 150\cdot(z^{32}+z^{92}) + 8\cdot(z^{34}+z^{90}) + 168\cdot(z^{36}+z^{88}) + \\
&\quad 96\cdot(z^{38}+z^{86}) + 75\cdot(z^{40}+z^{84}) + 468\cdot(z^{42}+z^{82}) + 132\cdot(z^{44}+z^{80}) + \\
&\quad 656\cdot(z^{46}+z^{78}) + 680\cdot(z^{48}+z^{76}) + 300\cdot(z^{50}+z^{74}) + 1386\cdot(z^{52}+z^{72}) + \\
&\quad 198\cdot(z^{54}+z^{70}) + 1152\cdot(z^{56}+z^{68}) + 1272\cdot(z^{58}+z^{66}) + 301\cdot(z^{60}+z^{64}) + 2136\cdot z^{62} \\
A_{\mathscr{B}_{61}}^{(S_{31})} &= (1+z^{124}) + 2\cdot z^{62} \\
A_{\mathscr{B}_{61}}^{(S_{61})} &= (1+z^{124}) + 2\cdot z^{62}
\end{aligned}
$$

The weight distributions of $\mathscr{B}_{61}$ and their modular congruence are shown in Table D.10.

Table D.10: Modular congruence weight distributions of $\mathscr{B}_{61}$

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_5)$ mod 5	$A_i(S_{31})$ mod 31	$A_i(S_{61})$ mod 61	$A_i(\mathcal{H})$ mod 113460	n_i†	A_i
0	1	1	1	1	1	1	0	1
20	0	0	3	0	0	90768	0	90768
22	0	1	0	0	0	75640	4	529480
24	2	2	0	0	0	94550	95	10873250
26	0	2	4	0	0	83204	1508	171180884
28	2	2	3	0	0	71858	19029	2159102198
30	0	0	1	0	0	68076	199795	22668808776
32	0	1	0	0	0	75640	1759003	199576556020
34	0	0	3	0	0	90768	13123969	1489045613508
36	2	0	3	0	0	34038	83433715	9466389337938
38	0	1	1	0	0	30256	454337550	51549138453256
40	0	2	0	0	0	37820	2128953815	241551099887720
42	0	0	3	0	0	90768	8619600220	977979841051968
44	0	0	2	0	0	22692	30259781792	3433274842143012
46	0	2	1	0	0	105896	92387524246	10482288501057056
48	0	2	0	0	0	37820	245957173186	27906300869721380
50	0	2	0	0	0	37820	572226179533	64924782329852000
52	0	2	1	0	0	105896	1165598694540	132248827882614296
54	0	2	3	0	0	15128	2081950370302	236218089014480048
56	0	2	2	0	0	60512	3264875882211	370432817595720572
58	0	2	2	0	0	60512	4499326496930	510493584341738312
60	1	2	1	0	0	20801	5452574159887	618649064180799821
62	0	2	1	2	2	102116	5813004046431	659543439108163376
64	1	2	1	0	0	20801	5452574159887	618649064180799821
66	0	2	2	0	0	60512	4499326496930	510493584341738312
68	0	2	2	0	0	60512	3264875882211	370432817595720572
70	0	2	3	0	0	15128	2081950370302	236218089014480048

Continued on next page

i	$A_i(S_2)$ mod 2^2	$A_i(S_3)$ mod 3	$A_i(S_5)$ mod 5	$A_i(S_{31})$ mod 31	$A_i(S_{61})$ mod 61	$A_i(\mathcal{H})$ mod 113460	n_i†	A_i
72	0	2	1	0	0	105896	1165598694540	132248827882614296
74	0	2	0	0	0	37820	572226179533	64924782329852000
76	0	2	0	0	0	37820	245957173186	27906300869721380
78	0	2	1	0	0	105896	92387524246	10482288501057056
80	0	0	2	0	0	22692	30259781792	3433274842143012
82	0	0	3	0	0	90768	8619600220	977979841051968
84	0	2	0	0	0	37820	2128953815	241551099887720
86	0	1	1	0	0	30256	454337550	51549138453256
88	2	0	3	0	0	34038	83433715	9466389337938
90	0	0	3	0	0	90768	13123969	1489045613508
92	0	1	0	0	0	75640	1759003	199576556020
94	0	0	1	0	0	68076	199795	22668808776
96	2	2	3	0	0	71858	19029	2159102198
98	0	2	4	0	0	83204	1508	171180884
100	2	2	0	0	0	94550	95	10873250
102	0	1	0	0	0	75640	4	529480
104	0	0	3	0	0	90768	0	90768
124	1	1	1	1	1	1	0	1

† $n_i = \dfrac{A_i - A_i(\mathcal{H})}{113460}$

E Weight Distributions of Quadratic Residue Codes of Primes 151 and 167

Table E.1: Weight distributions of QR and extended QR codes of prime 151

i	A_i of $[152, 76, 20]$ code	A_i of $[151, 76, 19]$ code
0	1	1
19	0	3775
20	28690	24915
23	0	113250
24	717250	604000
27	0	30256625
28	164250250	133993625
31	0	8292705580
32	39390351505	31097645925
35	0	1302257122605
36	5498418962110	4196161839505
39	0	113402818847850
40	430930711621830	317527892773980
43	0	5706949034630250
44	19714914846904500	14007965812274250
47	0	171469716029462700
48	542987434093298550	371517718063835850
51	0	3155019195317144883
52	9222363801696269658	6067344606379124775
55	0	36274321608490644595
56	98458872937331749615	62184551328841105020
59	0	264765917968736096775
60	670740325520798111830	405974407552062015055
63	0	1241968201959417159800
64	2949674479653615754525	1707706277694198594725
67	0	3778485133479463579225
68	8446025592483506824150	4667540459004043244925
71	0	7503425412744902320620
72	15840564760239238232420	8337139347494335911800
75	0	9763682329503348632684
76	19527364659006697265368	9763682329503348632684

Table E.2: Weight distributions of QR and extended QR codes of prime 167

i	A_i of $[168, 84, 24]$ code	A_i of $[167, 84, 23]$ code
0	1	1
23	0	110888
24	776216	665328
27	0	3021698
28	18130188	15108490
31	0	1057206192
32	5550332508	4493126316
35	0	268132007628
36	1251282702264	983150694636
39	0	39540857275985
40	166071600559137	126530743283152
43	0	3417107288264670
44	13047136918828740	9630029630564070
47	0	179728155397349776
48	629048543890724216	449320388493374440
51	0	5907921405841809432
52	19087130695796615088	13179209289954805656
55	0	124033230083117023704
56	372099690249351071112	248066460166234047408
59	0	1692604114105553659010
60	4739291519495550245228	3046687405389996586218
63	0	15228066033367763990128
64	39973673337590380474086	24745607304222616483958
67	0	91353417175290660468884
68	225696677727188690570184	134343260551898030101300
71	0	368674760966511746549004
72	860241108921860741947676	491566347955348995398672
75	0	1007629118755817710057646
76	2227390683565491780127428	1219761564809674070069782
79	0	1873856945935044844028880
80	3935099586463594172460648	2061242640528549328431768
83	0	2377873706297857672084688
84	4755747412595715344169376	2377873706297857672084688

Part VII

References

References

Ahmed, M., Ambroze, M. and Tomlinson, M. (2007). On computing Shannon's sphere packing bound and applications, *Proc. 9th International Symposium on Communication Theory and Applications*, Ambleside, Lake District, UK. (Cited on pages 20, 21 and 142.)

Alltop, W. O. (1984). A Method of Extending Binary Linear Codes, *IEEE Transactions on Information Theory* **30**(6): 871–872. (Cited on pages 89 and 156.)

Ambroze, M., Wade, G. and Tomlinson, M. (2000). Practical Aspects of Iterative Decoding, *IEE Proceedings Communications* **147**(2): 69–74. (Cited on page 59.)

Bahl, L. R., Cocke, J., Jelinek, F. and Raviv, J. (1974). Optimal decoding of linear codes for minimising symbol error rate, *IEEE Transactions on Information Theory* **IT-20**: 284–287. (Cited on pages 13, 61 and 133.)

Beenker, G. (1984). A Note on Extended Quadratic Residue Codes over GF(9) and Their Ternary Images, *IEEE Transactions on Information Theory* **30**(2): 403–405. (Cited on page 97.)

Berlekamp, E., McEliece, R. and van Tilborg, H. (1978). On the inherent intractability of certain coding problems, *IEEE Transactions on Information Theory* **24**: 384–386. (Cited on page 73.)

Berlekamp, E. R. (1984). *Algebraic Coding Theory*, Revised Edition edn, Aegean Park Press. ISBN 0 894 12063 8. (Cited on page 84.)

Berrou, C., Glavieux, A. and Thitimajshima, P. (1993). Near shannon limit error-correcting coding: Turbo codes, *Proc. IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, pp. 1064–1070. (Cited on page 14.)

Bierbrauer, J. and Edel, Y. (1997). Extending and Lengthening BCH-codes, *Finite Fields and Their Applications* **3**: 314–333. (Cited on pages 155 and 156.)

Bitner, J. R., Ehrlich, G. and Reingold, E. M. (1976). Efficient generation of the binary reflected gray code and its applications, *Communications of the ACM* **19**(9): 517–521. (Cited on pages 81 and 134.)

Blake, I. F. (1973). *Algebraic Coding Theory: History and Development*, Dowden, Hutchinson and Ross, Inc. (Cited on page 10.)

Blokh, E. and Zyablov, V. (1974). Coding of generalized concatenated codes, *Problems of Information Transmission* **10**: 218–222. (Cited on pages 13 and 181.)

Bosma, W., Cannon, J. J. and Playoust, C. P. (1997). The Magma algebra system I: The user language, *Journal of Symbolic Computation* **24**: 235–266. (Cited on page 85.)

Brouwer, A. E. (1998). Bounds on the size of linear codes, *in* V. S. Pless and W. C. Huffman (eds), *Handbook of Coding Theory*, Elsevier, North Holland, pp. 295–461. (Cited on pages 16, 22, 24, 85, 86, 90, 94, 95, 193 and 205.)

Brouwer, A. and Verhoeff, T. (1993). An updated table of minimum-distance bounds for binary linear codes, *IEEE Transactions on Information Theory* **39**(2): 662–677. (Cited on page 16.)

Burnside, W. (1955). *Theory of Group of Finite Order*, 2nd, 1911 edn, Reprinted by Dover, New York. (Cited on page 116.)

Butman, S. and McEliece, R. J. (1974). The ultimate limits of binary coding for a wideband gaussian channel, *JPL Deep Space Network Progress Report* **42-22**: 78–80. (Cited on pages 21 and 142.)

Calabi, L. and Myrvaagnes, E. (1964). On the minimal weight of binary group codes, *IEEE Transactions on Information Theory* **10**(4): 385–387. (Cited on page 16.)

Campello, J. and Modha, D. S. (2001). Extended bit-filling and ldpc code design, *Proc. IEEE Global Communications Conference (GLOBECOM)*, San Antonio, TX, USA, pp. 985–989. (Cited on page 33.)

Campello, J., Modha, D. S. and Rajagopalan, S. (2001). Designing ldpc codes using bit-filling, *Proc. IEEE International Conference on Communications (ICC)*, Helsinki, Finland, pp. 55–59. (Cited on page 33.)

Chase, D. (1972). A class of algorithms for decoding block codes with channel measurement information, *IEEE Transactions on Information Theory* **IT-18**: 170–182. (Cited on pages 17 and 134.)

Chen, C. L. (1969). *Some Results on Algebraically Structured Error-Correcting Codes*, Ph.D dissertation, University of Hawaii, USA. (Cited on pages 80, 81, 84, 87, 93 and 178.)

Chen, C. L. (1970). Computer results on the minimum distance of some binary cyclic codes, *IEEE Transactions on Information Theory* **16**(3): 359–360. (Cited on pages 24, 84 and 94.)

Chung, S. Y., Forney, Jr., G. D., Richardson, T. J. and Urbanke, R. L. (2001). On the design of low-density parity check codes within 0.0045 db of the shannon limit, *IEEE Communications Letters* **3**(2): 58–60. (Cited on pages 15, 29 and 33.)

Chung, S. Y., Richardson, T. J. and Urbanke, R. L. (2001). Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation, *IEEE Transactions on Information Theory* **47**(2): 657–670. (Cited on page 33.)

Costello, Jr., D. and Forney, Jr., G. (2007). Channel coding: The road to channel capacity, *Proceedings of the IEEE* **95**(6): 1150–1177. (Cited on pages 7, 16 and 34.)

Costello, Jr., D. J., Hagenauer, J., Imai, H. and Wicker, S. B. (1998). Applications of error-control coding, *IEEE Transactions on Information Theory* **44**(6): 2531–2560. (Cited on pages 13 and 14.)

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley & Sons, Inc. ISBN 0 471 06259 6. (Cited on pages 17, 18 and 19.)

Cygan, D. and Offer, E. (1991). Short linear incremental redundancy codes having optimal weight structure profile, *IEEE Transactions on Information Theory* **37**(1): 192–195. (Cited on page 154.)

Davey, M. C. and MacKay, D. J. C. (1998). Low-Density Parity-Check Codes over GF(q), *IEEE Communications Letters* **2**: 165–167. (Cited on page 35.)

Davida, G. and Reddy, S. (1972). Forward-error correction with decision feedback, *Information and Control* **21**: 117–133. (Cited on pages 153 and 154.)

Divsalar, D., Jin, H. and McEliece, R. (1998). Coding theorem for 'turbo-like' codes, *Proc. 36th Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, USA, pp. 201–210. (Cited on page 16.)

Dolinar, S., Divsalar, D. and Pollara, F. (1998). Code Performance as a Function of Block Size, *TMO Progress Report* pp. 42–133. Available: `http://tmo.jpl.nasa.gov/tmo/progress_report/`. (Cited on page 20.)

Dorsch, B. G. (1974). A decoding algorithm for binary block codes and J-ary output channels, *IEEE Transactions on Information Theory* **20**: 391–394. (Cited on pages 17, 22, 25, 134, 135, 136, 150, 167 and 179.)

Etzion, T., Trachtenberg, A. and Vardy, A. (1999). Which codes have cycle-free tanner graphs?, *IEEE Transactions on Information Theory* **45**(6): 2173–2181. (Cited on pages 35 and 56.)

Forney, Jr., G. (1973). The Viterbi Algorithm, *Proceedings of the IEEE* **61**(3): 268–278. (Cited on page 13.)

Forney, Jr., G. D. (1966). *Concatenated Codes*, MIT Press. (Cited on pages 13 and 14.)

Fossorier, M. (2001). Iterative reliability-based decoding of low-density parity check codes, *IEEE Journal on Selected Areas in Communications* **JSAC-19**: 908–917. (Cited on page 148.)

Fossorier, M. (2002). Reliability-based soft-decision decoding with iterative information set reduction, *IEEE Transactions on Information Theory* **IT-48**: 3101–3106. (Cited on page 134.)

Fossorier, M. (2004). Reliability-based soft-decision decoding algorithms for linear block codes, *in* S. Lin and D. J. Costello, Jr. (eds), *Error Control Coding: Fundamentals and Applications*, 2nd edn, Pearson Education, Inc, chapter 10. (Cited on page 133.)

Fossorier, M. and Lin, S. (1995). Soft-decision decoding of linear block codes based on ordered statistics, *IEEE Transactions on Information Theory* **41**(5): 1379–1396. (Cited on pages 17 and 134.)

Fossorier, M. and Lin, S. (1996). Computationally efficient soft-decision decoding of linear block codes based upon ordered statistics, *IEEE Transactions on Information Theory* **42**: 738–750. (Cited on page 134.)

Fossorier, M. and Lin, S. (1999). Reliability-based information set decoding of binary linear codes, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E82-A**: 2034–2042. (Cited on page 134.)

Gaborit, P. (2002). Quadratic Double Circulant Codes over Fields, *Journal of Combinatorial Theory, Series A* **97**: 85–107. (Cited on pages 97 and 111.)

Gaborit, P., Nedeloaia, C.-S. and Wassermann, A. (2005). On the weight enumerators of duadic and quadratic residue codes, *IEEE Transactions on Information Theory* **51**(1): 402–407. (Cited on pages 24, 98, 125, 126, 127 and 179.)

Gaborit, P. and Otmani, A. (2007). Tables of self-dual codes. `http://www.unilim.fr/pages_perso/philippe.gaborit/SD/index.html`. (Cited on page 98.)

Gallager, R. (1962). Low-density parity-check codes, *IRE Transactions on Information Theory* **IT-8**: 21–28. (Cited on pages 15 and 32.)

Gallager, R. (1963). *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press. (Cited on pages 15, 29 and 63.)

Gazelle, D. and Snyders, J. (1997). Reliability-based code-search algorithm for maximum likelihood decision decoding of block codes, *IEEE Transactions on Information Theory* **IT-43**: 239–249. (Cited on page 17.)

Golay, M. J. E. (1949). Notes on Digital Coding, *Proceedings of the IEEE* **37**: 657. (Cited on page 11.)

Grassl, M. (2000). On the minimum distance of some quadratic residue codes, *Proc. IEEE International Symposium on Information Theory (ISIT)*, Sorento, Italy, p. 253. (Cited on pages 87 and 128.)

Grassl, M. (2001). New binary codes from a chain of cyclic codes, *IEEE Transactions on Information Theory* **47**(3): 1178–1181. (Cited on pages 91 and 156.)

Grassl, M. (2006). Searching for linear codes with large minimum distance, *in* W. Bosma and J. Cannon (eds), *Discovering Mathematics with MAGMA – Reducing the Abstract to the Concrete*, Heidelberg: Springer, pp. 287–313. (Cited on pages 75, 77 and 89.)

Grassl, M. (2007). Code Tables: Bounds on the parameters of various types of codes. `http://www.codetables.de`. (Cited on pages 17, 22, 85, 86, 142 and 157.)

Gulliver, T. A. and Senkevitch, N. (1999). On a class of self-dual codes derived from quadratic residue, *IEEE Transactions on Information Theory* **45**(2): 701–702. (Cited on pages 97 and 122.)

Guruswami, V. (2001). *List Decoding of Error-Correcting Codes*, Ph.D disertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, United States. (Cited on page 150.)

Guruswami, V. and Sudan, M. (1999). Improved decoding of Reed-Solomon and algebraic-geometry codes, *IEEE Transactions on Information Theory* **45**(6): 1757–1767. (Cited on pages 17 and 150.)

Hagenauer, J. and Hoeher, P. (1989). A viterbi algorithm with soft-decision outputs and its applications, *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dallas, TX, USA, pp. 1680–1686. (Cited on page 15.)

Hamming, R. W. (1950). Error Detecting and Error Correcting Codes, *Bell System Technical Journal* **29**: 147–160. (Cited on pages 10 and 11.)

Han, Y. S., Hartmann, C. R. P. and Chen, C. C. (1993). Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes, *IEEE Transactions on Information Theory* **IT-39**: 1514–1523. (Cited on page 17.)

Hartley, R. (1928). Transmission of information, *Bell System Technical Journal* **7**(3): 535–563. (Cited on pages 3 and 4.)

Hartmann, C. R. P. and Rudolph, L. D. (1976). An optimum symbol-by-symbol decoding rule for linear codes, *IEEE Transactions on Information Theory* **IT-22**(5): 514–517. (Cited on pages 13, 22, 23, 61, 62, 133 and 179.)

Helgert, H. and Stinaff, R. (1973). Minimum-distance bounds of binary linear codes, *IEEE Transactions on Information Theory* **19**(3): 344–356. (Cited on page 16.)

Houghten, S., Lam, C., Thiel, L. and Parker, J. (2003). The extended quadratic residue code is the only (48,24,12) self-dual doubly-even code, *IEEE Transactions on Information Theory* **49**(1): 53–59. (Cited on page 143.)

Hu, X. Y., Eleftheriou, E. and Arnold, D. M. (2002). Irregular Progressive Edge-Growth Tanner Graphs, *Proc. IEEE International Symposium on Information Theory (ISIT)*, Lausanne, Switzerland. (Cited on pages 33 and 45.)

Hu, X. Y., Eleftheriou, E. and Arnold, D. M. (2005). Regular and Irregular Progressive Edge-Growth Tanner Graphs, *IEEE Transactions on Information Theory* **51**(1): 386–398. (Cited on pages 23, 35 and 50.)

Huffman, W. C. and Pless, V. S. (2003). *Fundamentals of Error-Correcting Codes*, Cambridge University Press. ISBN 0 521 78280 5. (Cited on page 98.)

Isaka, M., Fossorier, M. and Imai, H. (2004). On the suboptimality of iterative decoding for turbo-like and ldpc codes with cycles in their tanner graph representation, *IEEE Transactions on Communications* **52**(5): 845–854. (Cited on page 134.)

Jenson, R. (1980). A Double Circulant Presentation for Quadratic Residue Codes, *IEEE Transactions on Information Theory* **26**(2): 223–227. (Cited on pages 104 and 105.)

Jin, H., Khandekar, A. and McEliece, R. (2000). Irregular repeat-accumulate codes, *Proc. 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, pp. 1–8. (Cited on pages 16 and 189.)

Johnson, S. (2004). *Low-Density Parity-Check Codes from Combinatorial Designs*, Ph.D dissertation, School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW 2308, Australia. (Cited on page 34.)

Johnson, S. J. and Weller, S. R. (2001). Construction of low-density parity-check codes from kirkman triple systems, *Proc. IEEE Information Theory Workshop*, Cairns, Australia, pp. 90–92. (Cited on page 34.)

Johnson, S. J. and Weller, S. R. (2002). Codes for iterative decoding from partial geometries, *Proc. IEEE International Symposium on Information Theory (ISIT)*, Lausanne, Switzerland, p. 310. (Cited on page 34.)

Kaneko, T., Nishijima, T., Inazumi, H. and Hirasawa, S. (1994). An efficient maximum likelihood decoding of linear block codes with algebraic decoder, *IEEE Transactions on Information Theory* **IT-40**: 320–327. (Cited on page 134.)

Karlin, M. (1969). New binary coding results by circulants, *IEEE Transactions on Information Theory* **15**(1): 81–92. (Cited on pages 97, 104, 105 and 107.)

Karlin, M., Bhargava, V. K. and Tavares, S. E. (1978). A note on the extended quadratic residue codes and their binary images, *Information and Control* **38**: 148–153. (Cited on page 108.)

Knuth, D. E. (2005). *The Art of Computer Programming, Vol. 4: Fascicle 3: Generating All Combinations and Partitions*, 3rd edn, Addison-Wesley. ISBN 0 201 85394 9. (Cited on pages 81, 82 and 135.)

Koetter, R. and Vardy, A. (2003). Algebraic soft-decision decoding of Reed-Solomon codes, *IEEE Transactions on Information Theory* **49**(11): 2809–2825. (Cited on pages 17 and 150.)

Kou, Y., Lin, S. and Fossorier, M. (2001). Low density parity check codes based on finite geometries: A rediscovery and new results, *IEEE Transactions on Information Theory* **47**: 2711–2736. (Cited on pages 34, 41 and 146.)

Krishna, H. and Morgera, S. (1987). A new error control scheme for hybrid ARQ systems, *IEEE Transactions on Communications* **35**(10): 981–990. (Cited on page 154.)

Lenth, R. (1989). Cummulative distribution function of non-central t distribution, *Journal of Applied Statistics* **38**(1): 185–189. (Cited on page 20.)

Leon, J. S. (1988). A probabilistic algorithm for computing minimum weights of large error-correcting codes, *IEEE Transactions on Information Theory* **34**(5): 1354–. (Cited on page 129.)

Leon, J. S., Masley, J. M. and Pless, V. (1984). Duadic codes, *IEEE Transactions on Information Theory* **30**(5): 709–713. (Cited on pages 91, 129 and 130.)

Lin, S. and Costello, Jr., D. J. (2004). *Error Control Coding: Fundamentals and Applications*, 2nd edn, Pearson Education, Inc. (Cited on pages 34, 146 and 153.)

Liu, R., Spasojević, P. and Soljanin, E. (2003). Punctured turbo code ensembles, *Proc. IEEE Information Theory Workshop*, Paris, France, pp. 249–252. (Cited on pages 153 and 154.)

Lodge, J., Hoeher, P. and Hagenauer, J. (1992). The decoding of multidimensional codes using separable map 'filters', *Proc. 16th Biennial Symposium on Communications*, Kingston, Ontario, Canada, pp. 343–346. (Cited on page 15.)

Lodge, J., Young, R., Hoeher, P. and Hagenauer, J. (1993). Separable map 'filters' for the decoding of product and concatenated codes, *Proc. IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, pp. 1740–1745. (Cited on page 15.)

Loelocian, M. and Conan, J. (1984). A [55, 16, 19] binary Goppa code, *IEEE Transactions on Information Theory* **30**: 773. (Cited on page 75.)

Lous, N. J. C., Bours, P. A. H. and van Tilborg, H. C. A. (1993). On maximum likelihood soft-decision decoding of binary linear codes, *IEEE Transactions on Information Theory* **39**: 197–203. (Cited on page 134.)

Luby, M. G., Shokrolloahi, M. A., Mizenmacher, M. and Spielman, D. A. (2001). Improved Low-Density Parity-Check Codes Using Irregular Graphs, *IEEE Transactions on Information Theory* **47**(2): 585–598. (Cited on pages 15, 32 and 185.)

Lucas, R., Fossorier, M. P. C., Kou, Y. and Lin, S. (2000). Iterative decoding of one-step majority logic decodable codes based on belief propagation, *IEEE Transactions on Communications* **46**(6): 931–937. (Cited on page 34.)

Luke, H. D. (1999). The origins of the sampling theorem, *IEEE Communications Magazine* **37**(4): 106–108. (Cited on page 3.)

MacKay, D. J. C. and Neal, R. M. (1996). Near Shannon limit performance of low-density parity-check codes, *Electronics Letters* **32**(18): 1645–1646. (Cited on page 15.)

MacWilliams, F. J. and Sloane, N. J. A. (1977). *The Theory of Error-Correcting Codes*, North-Holland. (Cited on pages 35, 36, 42, 84, 86, 87, 97, 99, 100, 104, 105, 107, 109, 110, 111, 132 and 136.)

Mandelbaum, D. (1974). An adaptive-feedback coding scheme using incremental redundancy, *IEEE Transactions on Information Theory* **IT-20**(3): 388–389. (Cited on pages 153 and 154.)

Margulis, G. A. (1982). Explicit Constructions of Graphs without Short Cycles and Low Density Codes, *Combinatorica* **2**(1): 71–78. (Cited on page 34.)

McEliece, R. J., MacKay, D. J. C. and Cheng, J.-F. (1998). Turbo decoding as an instance of pearl's "belief propagation" algorithm, *IEEE Journal on Selected Areas in Communications* **16**: 140–152. (Cited on pages 35 and 56.)

Moon, T. K. (2005). *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley-Interscience. (Cited on page 20.)

Moore, E. H. (1976). *Double Circulant Codes and Related Algebraic Structures*, Ph.D dissertation, Dartmouth College, USA. (Cited on page 97.)

Mykkeltveit, J., Lam, C. and McEliece, R. J. (1972). On the weight enumerators of quadratic residue codes, *JPL Technical Report 32-1526* **XII**: 161–166. (Cited on pages 116, 117, 124, 131 and 179.)

Narayanan, K. and Stuber, G. (1997). A novel ARQ technique using the turbo coding principle, *IEEE Communications Letters* **1**: 49–51. (Cited on pages 153 and 154.)

Nijenhuis, A. and Wilf, H. S. (1978). *Combinatorial Algorithms for Computers and Calculators*, 2nd edn, Academic Press, London. (Cited on pages 81, 134 and 150.)

Nyquist, H. (1924). Certain factors affecting telegraph speed, *Bell System Technical Journal* 3(2): 324–346. (Cited on page 3.)

Papagiannis, E., Ambroze, M. A. and Tomlinson, M. (2003a). Analysis of non convergence blocks at low and moderate SNR in SCC turbo schemes, *Proc. 8th International Worksop on Signal Processing for Space Communications (SPSC)*, Catania, Italy, pp. 121–128. (Cited on pages 45 and 70.)

Papagiannis, E., Ambroze, M. A. and Tomlinson, M. (2004). Approaching the ML performance with iterative decoding, *Proc. International Zurich Seminar on Communications*, Zurich, Switzerland, pp. 220–223. (Cited on page 70.)

Papagiannis, E., Ambroze, M., Tomlinson, M. and Ahmed, M. (2005). Improved decoding of low-density parity-check codes by reduction of pseudocodewords, *in* C. E. Palau Salvador (ed.), *Proc. 4th IASTED International Conference Communication Systems and Networks*, ACTA Press, pp. 152–157. (Cited on page 70.)

Payne, W. H. and Ives, F. M. (1979). Combination generators, *ACM Transactions on Mathematical Software* 5(2): 163–172. (Cited on page 81.)

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA. (Cited on pages 15, 29 and 63.)

Peterson, W. and Weldon, Jr., E. J. (1972). *Error-Correcting Codes*, MIT Press. (Cited on pages 11 and 38.)

Ping, L., Leung, W. K. and Phamdo, N. (1999). Low Density Parity Check Codes with Semi-Random Parity Check Matrix, *Electronics Letters* **35**(1): 38–39. (Cited on pages 34, 54 and 185.)

Pishro-Nik, H. and Fekri, F. (2003). Improved decoding algorithms for low-density parity-check codes, *Proc. 3rd International Symposium on Turbo Codes*, Brest, France, pp. 117–120. (Cited on page 70.)

Pless, V. (1972). Symmetry Codes over GF(3) and New Five Designs, *Journal of Combinatorial Theory, Series A* **12**: 119–142. (Cited on page 97.)

Posten, H. (1994). A new algorithm for the non central t distribution function, *Journal of Statistical Computation and Simulation* **51**: 79–87. (Cited on page 20.)

Prange, E. (1957). Cyclic error-correcting codes in two symbols, *Technical Report TN-58-103*, Air Force Cambridge Research Labs, Bedford, Massachusetts, USA. (Cited on page 78.)

Proakis, J. G. (2001). *Digital Communications*, 4th edn, McGraw-Hill, New York. (Cited on pages 4, 5, 6, 19, 20, 31 and 136.)

Promhouse, G. and Tavares, S. E. (1978). The Minimum Distance of All Binary Cyclic Codes of Odd Lengths from 69 to 99, *IEEE Transactions on Information Theory* **24**(4): 438–442. (Cited on pages 24, 84, 94 and 178.)

Rains, E. M. and Sloane, N. J. A. (1998). Self-Dual Codes, *in* V. S. Pless and W. C. Huffman (eds), *Handbook of Coding Theory*, Elsevier, North Holland. (Cited on pages 97, 98, 118 and 122.)

Richardson, T. J., Shokrollahi, M. A. and Urbanke, R. L. (2001). Design of capacity-approaching irregular low-density parity-check codes, *IEEE Transactions on Information Theory* **47**(2): 619–637. (Cited on pages 15 and 33.)

Richardson, T. J. and Urbanke, R. L. (2001). The capacity of low-density parity-check codes under message-passing decoding, *IEEE Transactions on Information Theory* **47**(2): 599–618. (Cited on page 33.)

Richter, G. and Hof, A. (2006). On a construction method of irregular LDPC codes without small stopping sets, *Proc. IEEE International Conference on Communications (ICC)*, Istanbul, Turkey, pp. 1119–1124. (Cited on page 33.)

Schomaker, D. and Wirtz, M. (1992). On Binary Cyclic Codes of Odd Lengths from 101 to 127, *IEEE Transactions on Information Theory* **38**(2): 516–518. (Cited on pages 24, 84, 94 and 178.)

Sesia, S., Caire, G. and Vivier, G. (2004). Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes, *IEEE Transactions on Communications* **52**(8): 1311–1321. (Cited on page 153.)

Shannon, C. E. (1948). A mathematical theory of communication, *Bell System Technical Journal* **27**(3): 379–423. (Cited on pages 4, 5, 8, 10, 11, 17, 18 and 33.)

Shannon, C. E. (1949). Communication in the presence of noise, *Proceedings of the IRE* **37**(1): 10–21. (Cited on page 3.)

Shannon, C. E. (1959). Probability of error for optimal codes in a gaussian channel, *Bell System Technical Journal* **38**(3): 611–656. (Cited on pages 20, 21 and 142.)

Silverman, R. and Balser, M. (1954). Coding for constant-data-rate systems, *IRE Transactions on Information Theory* **4**(4): 50–63. (Cited on pages 17 and 61.)

Sklar, B. (2001). *Digital Communications: Fundamentals and Applications*, 2nd edn, Prentice Hall, Upper Saddle River, New Jersey. (Cited on pages 4 and 5.)

Sloane, N. (1972). A survey of constructive coding theory, and a table of binary codes of highest known rate, *Discrete Mathematics* **3**: 265–294. (Cited on page 16.)

Sloane, N. J., Reddy, S. M. and Chen, C. L. (1972). New binary codes, *IEEE Transactions on Information Theory* **IT-18**: 503–510. (Cited on pages 88, 90 and 155.)

Snyders, J. (1991). Reduced lists of error patterns for maximum likelihood soft decision decoding, *IEEE Transactions on Information Theory* **37**: 1194–1200. (Cited on page 134.)

Soljanin, E., Varnica, N. and Whiting, P. (2006). Punctured vs rateless codes for hybrid ARQ, *Proc. IEEE Information Theory Workshop*, Punta del Este, Uruguay. (Cited on pages 153 and 154.)

Sudan, M. (1997). Decoding of Reed-Solomon codes beyond the error-correction bound, *Journal of Complexity* **13**: 180–193. (Cited on page 17.)

Sweeney, P. and Wesemeyer, S. (2000). Iterative soft-decision decoding of linear block codes, *IEE Proceedings Communications* **147**(3): 133–136. (Cited on page 134.)

Tang, H., Xu, J., Lin, S. and Abdel-Ghaffar, K. A. S. (2005). Codes on Finite Geometries, *IEEE Transactions on Information Theory* **51**(2): 572–596. (Cited on page 41.)

Tanner, R. M. (1981). A Recursive Approach to Low-Complexity Codes, *IEEE Transactions on Information Theory* **IT-27**: 533–547. (Cited on page 15.)

Thorpe, J. (2003). Low-density parity-check (LDPC) codes constructed from protographs, *JPL IPN Progress Report* **42–154**. Available: `http://tmo.jpl.nasa.gov/progress_report/42-154/154C.pdf`. (Cited on page 187.)

Tian, T., Jones, C., Villasenor, J. and Wesel, R. (2004). Selective Avoidance of Cycles in Irregular LDPC Code Construction, *IEEE Transactions on Communications* **52**: 1242–1247. (Cited on page 33.)

Tjhai, C. and Tomlinson, M. (2007). Results on binary cyclic codes, *Electronics Letters* **43**(4): 234–235. (Cited on page 156.)

Tjhai, C., Tomlinson, M., Grassl, M., Horan, R., Ahmed, M. and Ambroze, M. (2006). New linear codes derived from cyclic codes of length 151, *IEE Proceedings Communications* **153**(5): 581–585. (Cited on page 156.)

Tomlinson, M., Tjhai, C., Cai, J. and Ambroze, M. (2007). Analysis of the distribution of the number of erasures correctable by a binary linear code and the link to low weight codewords, *IET Proceedings Communications* **1**(3): 539–548. (Cited on page 134.)

Tomlinson, M., Wade, G., Van Eetvelt, P. and Ambroze, M. (2002). Bounds for finite block-length codes, *IEE Proceedings Communications* **149**(2): 65–69. (Cited on page 20.)

Ungerboeck, G. (1982). Channel Coding with Multilevel/Phase Signals, *IEEE Transactions on Information Theory* **IT-28**(1). (Cited on pages 7 and 14.)

Urbanke, R. (2001). LdpcOpt a fast and accurate degree distribution optimizer for LDPC code ensembles. Available at `http://lthcwww.epfl.ch/research/ldpcopt/`. (Cited on page 189.)

Valembois, A. and Fossorier, M. (2004). Box and match techniques applied to soft-decision decoding, *IEEE Transactions on Information Theory* **50**(5): 796–810. (Cited on page 17.)

van Dijk, M., Egner, S., Greferath, M. and Wassermann, A. (2005). On Two Doubly Even Self-Dual Binary Codes of Length 160 and Minimum Weight 24, *IEEE Transactions on Information Theory* **51**(1): 408–411. (Cited on pages 97, 98 and 115.)

van Lint, J. H. (1970). Coding Theory, *Lecture Notes in Mathematics No. 201*, Springer, Berlin. (Cited on page 132.)

Vardy, A. (1997). The intractability of computing the minimum distance of a code, *IEEE Transactions on Information Theory* **43**: 1759–1766. (Cited on page 73.)

Varnica, N. and Fossorier, M. (2004). Belief propagation with information correction : Imporved near maximum-likelihood decoding of low-density parity-check codes, *Proc. IEEE International Symposium on Information Theory (ISIT)*, Chichago, Illinois, USA, p. 343. (Cited on page 70.)

Vasic, B. and Milenkovic, M. (2004). Combinatorial Constructions of Low-Density Parity-Check Codes for Iterative Decoding, *IEEE Transactions on Information Theory* **50**(6): 1156–1176. (Cited on page 34.)

Verhoeff, T. (1987). An updated table of minimum-distance bounds for binary linear codes, *IEEE Transactions on Information Theory* **33**(5): 665–680. (Cited on page 16.)

Viterbi, A. J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, *IEEE Transactions on Information Theory* **IT-13**: 260–269. (Cited on pages 13, 14, 61 and 133.)

Weldon, Jr., E. J. (1966). Difference-set cyclic codes, *Bell System Technical Journal* **45**: 1045–1055. (Cited on pages 11 and 34.)

Wicker, S. and Bartz, M. (1994a). The design and implementation of Type-I and Type-II hybrid-ARQ protocols based on first-order Reed-Muller codes, *IEEE Transactions on Communications* **42**(2/3/4): 979–987. (Cited on page 154.)

Wozencraft, J. and Jacobs, I. (1965). *Principles of Communication Engineering*, John Wiley & Sons, Inc. (Cited on page 33.)

Zimmermann, K.-H. (1996). Integral hecke modules, integral generalized reed-muller codes, and linear codes, *Technical Report 3-96*, Technische Universität Hamburg-Harburg, Hamburg, Germany. (Cited on pages ix, 75, 76, 77, 78, 80, 93 and 112.)

Zinov'ev, V. (1976). Generalized concatenated codes, *Problems of Information Transmission* **12**(3): 5–15. (Cited on pages 13 and 181.)

Part VIII

Publications

Journals

Horan, R., Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2006). Idempotents, Mattson-Solomon Polynomials and Binary LDPC Codes, *IEE Proceedings Communications* **153**(2): 256–262.

Tjhai, C. and Tomlinson, M. (2007). Results on binary cyclic codes, *Electronics Letters* **43**(4): 234–235.

Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2005). Cyclotomic idempotent-based binary cyclic codes, *Electronics Letters* **41**(6): 341–343.

Tjhai, C., Tomlinson, M., Grassl, M., Horan, R., Ahmed, M. and Ambroze, M. (2006). New linear codes derived from cyclic codes of length 151, *IEE Proceedings Communications* **153**(5): 581–585.

Tomlinson, M., Tjhai, C. and Ambroze, M. (2007). Extending the dorsch decoder towards achieving maximum likelihood decoding for linear codes, *IET Proceedings Communications* **1**(3): 479–488.

Extending the Dorsch decoder towards achieving maximum-likelihood decoding for linear codes

M. Tomlinson, C. Tjhai and M. Ambroze

Abstract: It is shown that the relatively unknown Dorsch decoder may be extended to produce a decoder that is capable of maximum-likelihood decoding. The extension involves a technique for any linear (n, k) code that ensures that $n - k$ less reliable, soft decisions of each received vector may be treated as erasures in determining candidate codewords. These codewords are derived from low information weight codewords and it is shown that an upper bound of this information weight may be calculated from each received vector in order to guarantee that the decoder will achieve maximum-likelihood decoding. Using the cross-correlation function, it is shown that the most likely codeword may be derived from a partial correlation function of these low information weight codewords, which leads to an efficient fast decoder. For a practical implementation, this decoder may be further simplified into a concatenation of a hard-decision decoder and a partial correlation decoder with insignificant performance degradation. Results are presented for some powerful, known codes, including a GF(4) non-binary BCH code. It is shown that maximum-likelihood decoding is realised for a high percentage of decoded codewords and that performance close to the sphere packing bound is attainable for codeword lengths up to 1000 bits.

1 Introduction

In a relatively unknown paper published in 1974, Dorsch described a decoder for linear binary block (n, k) codes using soft decisions quantised to J levels [1]. The decoder is applicable to any linear block code and does not rely upon any particular features of the code, such as being a concatenated code or having a sparse parity check matrix. In the Dorsch decoder, hard decisions are derived from the soft decisions using standard bit-by-bit detection, choosing the binary state closest to the received coordinate. The hard decisions are then ranked in terms of their likelihoods and candidate codewords are derived from a set of k, independent, most likely bits. This is done by producing a new parity check matrix $\boldsymbol{H}_I$ obtained by reordering the columns of the original $\boldsymbol{H}$ matrix according to the likelihood of each coordinate and reducing the resulting matrix to echelon canonical form by elementary row operations. After evaluation of several candidate codewords, the codeword with the minimum soft decision metric is the output from the decoder. With the introduction of turbo principles in [2], it was shown in [3] that the Dorsch decoder can be modified to produce soft decision outputs and thus can be arranged in an iterative scheme to decode product codes. It was also shown in [3] that the Dorsch decoder can easily be adapted to decode non-binary linear codes.

In general, soft-decision decoding algorithms, which involve reordering the coordinates of the received vector (in the literature, these kinds of decoding algorithms are commonly referred to as reliability-based decoding), can be classified into two categories: most-reliable-positions reprocessing algorithms and least-reliable-positions reprocessing algorithms [4, Chapter 10]. In the former case, the most reliable k information set is determined and some of these k coordinates are then modified. For each modification, a codeword is derived and at the end of soft, or hard, decision decoding, the most likely codeword is chosen. In the latter case, the decoding algorithms use quite different strategies in reprocessing the coordinates and tend to be limited to hard decision decoding only. On the basis of the fact that errors are more likely to occur in the least-reliable-positions, the typical least-reliable-reprocessing algorithm derives some error patterns in the least-reliable-positions and subtracts these error patterns from the hard decision vector initially derived from the received vector. The resultant codewords are then decoded using an algebraic hard-decision decoder. The most likely codeword is chosen at the end of the procedure. It is worth mentioning that the least-reliable-positions reprocessing algorithms generally require that many more codewords be reprocessed than the most-reliable-positions reprocessing algorithms in order to find the correct codeword for each received vector.

The Dorsch decoding algorithm can be considered as the precursor of the most-reliable-positions reprocessing algorithms. A decoder using a similar principle, but without soft-decision quantisation, has been described by Fossorier and Lin [5] and is termed the ordered statistic decoder (OSD). An order-i OSD algorithm systematically reprocesses $\sum_{j=0}^{i} \binom{k}{j}$ error patterns in the k-most reliable positions of the information set. The complexity of the OSD algorithm depends on the size of the list containing the error patterns and various approaches have been devised to reduce the size of this list [6–9]. Another variant of the most-reliable-positions reprocessing algorithm was introduced by de Barros *et al.* [10] and de Barros and Dorsch [11]. This algorithm, however, employs a stopping rule, which results in suboptimum performance.

Various least-reliable-positions reprocessing algorithms have also been devised. One of them is the well-known

doi:10.1049/iet-com:20060338

Paper first received 6th June and in revised form 16th October 2006

The authors are with Fixed and Mobile Communications Research, University of Plymouth, PL4 8AA, UK

E-mail: ctjhai@plymouth.ac.uk

Chase algorithm which utilises an algebraic hard-decision decoder in conjunction with a search for errors in the least likely positions [12]. The list size of the Chase algorithm is constant and an extension to this algorithm, which generates a dynamic list and which can achieve maximum-likelihood decoding if all codewords are processed, has been proposed by Kaneko *et al.* [13]. Other least-reliable-positions reprocessing approaches include the syndrome-based list decoding algorithms, which are more suitable for high-rate codes, proposed by Snyders [14] and Lous *et al.* [15].

The power of the Dorsch decoder arises from the relatively unknown property that most codes, on average, can correct almost $n-k$ erasures [16], which is considerably more than the guaranteed number of correctible erasures of $d_{\min}-1$, or the guaranteed number of correctible hard decision errors of $(d_{\min}-1)/2$, where $d_{\min}$ is the minimum Hamming distance of the code. In its operation, the Dorsch decoder needs to correct any combination of $n-k$ erasures, which is impossible unless the code is a maximum distance separable (MDS) code [17]. Dorsch did not discuss this problem, or potential solutions, in his original paper, although at least one solution is implied by the results presented in the paper [1]. In this paper, one solution is described and it is also shown that it is not necessary to keep recalculating each candidate codeword and its associated soft-decision metric in order to find the most likely codeword. Instead, an incremental correlation approach is adopted, which features low information weight codewords and a correlation function involving only a small number of coordinates of the received vector [16]. This approach is very efficient and enables many more codewords to be reprocessed than the original Dorsch decoder and the other published most-reliable-positions reprocessing algorithms. It is shown that maximum-likelihood decoding is realised, provided all codewords are evaluated up to a bounded information weight, which may be calculated for each received vector. It is also shown that maximum-likelihood decoding may be achieved for a predetermined, high percentage of received vectors and is a function of the number of codewords processed for each received vector. The decoder lends itself to a low complexity, parallel implementation involving a concatenation of hard and soft-decision decoding. It produces near maximum-likelihood decoding for codes as long as 1000 bits, depending on code rate. Furthermore, it is shown that complexity may be a traded-off against performance in a flexible manner for implementation of the decoder. Decoding results are presented for some of the most powerful binary codes known and compared to Shannon's sphere packing bound [18]. The extension to non-binary codes is straightforward and this is described in Section 5.

2 Incremental correlation Dorsch decoder

Codewords with binary coordinates having state 0 or 1 are denoted as

$$x = (x_0, x_1, x_2, \ldots, x_{n-1})$$

For transmission, bipolar transmission is used with coordinates having binary state 0 mapped to +1 and having state 1 mapped to −1. Transmitted codewords are denoted as

$$c = (c_0, c_1, c_2, \ldots, c_{n-1})$$

The received vector r consists of n coordinates $(r_0, r_1, r_2, \ldots, r_{n-1})$ equal to the transmitted codeword plus additive white gaussian noise (AWGN) with variance σ^2. The received vector processed by the decoder is assumed to have been matched filtered and free from distortion so that $1/\sigma^2 = 2E_b/N_o$, where E_b is the energy per information bit and N_o is the single sided noise power spectral density. Accordingly, $\sigma^2 = N_o/2E_b$.

The basic principle that is used is that the k most reliable bits of the received vector are initially taken as correct and the $n-k$ least reliable bits are treated as erasures. The parity check equations of the code, as represented by $\boldsymbol{H}$, are used to solve for these erased bits and a codeword $\hat{x}$ is obtained. This codeword is either equal to the transmitted codeword or needs only small changes to produce a codeword equal to the transmitted codeword.

One difficulty is that, depending on the code, the $n-k$ least reliable bits usually cannot all be solved as erasures. This depends on the positions of the erased coordinates and the power of the code. Only MDS codes are capable of solving $n-k$ erasures regardless of the positions of the erasures in the received codeword [17]. Unfortunately, there are no binary MDS codes apart from trivial examples. However, a set of $n-k$ erasures can always be solved from $n-k+s$ least reliable bit positions and, depending on the code, s is usually a small integer. In order to obtain best performance, it is important that the very least reliable bit positions are solved first, since the corollary that the $n-k$ least reliable bits usually cannot all be solved as erasures is that the k most reliable bits, used to derive codeword $\hat{x}$, must include a small number of least reliable bits. However, for most received vectors, the difference in reliability between ranked bit k and ranked bit $k+s$ is usually small.

For any received coordinate, the a priori log-likelihood ratio of the bit being correct is proportional to $|r_i|$. The received vector r with coordinates ranked in order of most likely to be correct is defined as $(r_{\mu_0}, r_{\mu_1}, r_{\mu_2}, \ldots, r_{\mu_{n-1}})$, where $|r_{\mu_0}| > |r_{\mu_1}| > |r_{\mu_2}| > \cdots > |r_{\mu_{n-1}}|$.

The decoder is most straightforward for a binary MDS code. The codeword coordinates $(x_{\mu_0}, x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{k-1}})$ are formed directly from the received vector r using the bitwise decision rule $x_{\mu_i} = 1$ if $r_{\mu_i} < 0$ else $x_{\mu_i} = 0$.

The $n-k$ coordinates $(x_{\mu_k}, x_{\mu_{k+1}}, x_{\mu_{k+2}}, \ldots, x_{\mu_{n-1}})$ are considered to be erased and derived from the k most reliable codeword coordinates $(x_{\mu_0}, x_{\mu_1}, x_{\mu_2}, \ldots, x_{\mu_{k-1}})$ by using the parity check equations.

For a non-MDS code, the $n-k$ coordinates cannot always be solved from the parity check equations because the parity check matrix is not a Cauchy or Vandermonde matrix [17]. To obviate this problem, a slightly different order is defined $(x_{\eta_0}, x_{\eta_1}, x_{\eta_2}, \ldots, x_{\eta_{n-1}})$.

The label of the last coordinate η_{n-1} is set equal to μ_{n-1} and $x_{\eta_{n-1}}$ solved first by flagging the first parity check equation that contains $x_{\eta_{n-1}}$ and then subtracting this equation from all other parity check equations containing $x_{\eta_{n-1}}$. Consequently, $x_{\eta_{n-1}}$ is now only contained in one equation, the first flagged equation.

The label of the next coordinate η_{n-2} is set equal to μ_{n-2} and an attempt is made to solve $x_{\eta_{n-2}}$ by finding an unflagged parity check equation containing $x_{\eta_{n-2}}$. In the event that there is not an unflagged equation containing $x_{\eta_{n-2}}$, η_{n-2} is set equal to μ_{n-3} the label of the next most reliable bit, $x_{\mu_{n-3}}$ and the procedure repeated until an unflagged equation contains $x_{\eta_{n-2}}$. As before this equation is flagged that it will be used to solve for $x_{\eta_{n-2}}$ and is subtracted from all other unflagged equations containing $x_{\eta_{n-2}}$. The procedure continues until all of the $n-k$ codeword coordinates $x_{\eta_{n-1}}, x_{\eta_{n-2}}, x_{\eta_{n-3}}, \ldots, x_{\eta_k}$ have been solved and all $n-k$ equations have been flagged. In effect, the least reliable coordinates are skipped if they

cannot be solved. The remaining k ranked received coordinates are set equal to $(r_{\eta_0}, r_{\eta_1}, r_{\eta_2}, \ldots, r_{\eta_{k-1}})$ in most reliable order where $|r_{\eta_0}| > |r_{\eta_1}| > |r_{\eta_2}| > \cdots > |r_{\eta_{n-1}}|$ and $(x_{\eta_0}, x_{\eta_1}, x_{\eta_2}, \ldots, x_{\eta_{k-1}})$ determined using the bit decision rule $x_{\eta_i}=1$ if $r_{\eta_i} < 0$ else $x_{\eta_i} = 0$. The flagged parity check equations are in upper triangular form and have to be solved in reverse order starting with the last flagged equation. This equation gives the solution to x_{η_k}, which is back substituted into the other equations and $x_{\eta_{k+1}}$ is solved next, back substituted and so on, with coordinate $x_{\eta_{n-1}}$ solved last.

This codeword is denoted as $\hat{x}$ and the mapped version of the codeword is denoted as $\hat{c}$. As is well known in [19], the codeword most likely to be transmitted is the one denoted as $\check{x}$, which has the smallest squared Euclidean distance, $D(\check{x})$, between the mapped codeword, $\check{c}$, and the received vector

$$D(\check{x}) = \sum_{j=0}^{n-1} (r_j - \check{c}_j)^2 \tag{1}$$

$D(\check{x}) < D(x)$ for all other codewords x.

Equivalently, $\check{x}$ is the codeword, after mapping, which has the highest cross-correlation

$$Y(\check{x}) = \sum_{j=0}^{n-1} r_j \times \check{c}_j \tag{2}$$

$Y(\check{x}) > Y(x)$ for all other codewords x.

The decoder may be simplified if the cross-correlation function is used to compare candidate codewords. The cross-correlation is first determined for the codeword $\hat{x}$

$$Y(\hat{x}) = \sum_{j=0}^{n-1} r_j \times \hat{c}_j \tag{3}$$

It is interesting to make some observations about $Y(\hat{x})$. Since the summation can be carried out in any order

$$Y(\hat{x}) = \sum_{j=0}^{n-1} r_{\eta_j} \times \hat{c}_{\eta_j} \tag{4}$$

and

$$Y(\hat{x}) = \sum_{j=0}^{k-1} r_{\eta_j} \times \hat{c}_{\eta_j} + \sum_{j=k}^{n-1} r_{\eta_j} \times \hat{c}_{\eta_j} \tag{5}$$

Considering the first term

$$\sum_{j=0}^{k-1} r_{\eta_j} \times \hat{c}_{\eta_j} = \sum_{j=0}^{k-1} \left| r_{\eta_j} \right| \tag{6}$$

This is because the sign of $\hat{c}_{\eta_j}$ equals the sign of r_{η_j} for $j < k$. Thus, this term is independent of the code and (5) becomes

$$Y(\hat{x}) = \sum_{j=0}^{k-1} |r_{\eta_j}| + \sum_{j=k}^{n-1} r_{\eta_j} \times \hat{c}_{\eta_j} \tag{7}$$

Almost all of the k largest received coordinates (all of the k largest terms for an MDS code) are contained in the first term of (7) and this ensures that the codeword $\hat{x}$, after mapping, has a high correlation with r.

A binary, (hard decision), received vector b may be derived from the received vector r using the bitwise decision rule $b_j = 1$ if $r_j < 0$ else $b_j = 0$ for $j = 0$ to $n - 1$. It should be noted that in general, the binary vector b is not a codeword.

It is useful to define a binary vector $\hat{z}$ as

$$\hat{z} = b \oplus \hat{x} \tag{8}$$

The maximum attainable correlation $Y_{\max}$ is given by

$$Y_{\max} = \sum_{j=0}^{n-1} \left| r_{\eta_j} \right| \tag{9}$$

This correlation value occurs when there are no bit errors in transmission and provides an upper bound to the maximum achievable correlation for $\check{x}$. The correlation $Y(\hat{x})$ may be expressed in terms of $Y_{\max}$ and $\hat{x}$ for

$$Y(\hat{x}) = Y_{\max} - 2 \sum_{j=0}^{n-1} \hat{z}_{\eta_j} \times \left| r_{\eta_j} \right| \tag{10}$$

equivalently

$$Y(\hat{x}) = Y_{\max} - Y_{\Delta}(\hat{x}) \tag{11}$$

where $Y_{\Delta}(\hat{x})$ is the shortfall from the maximum achievable correlation for the codeword $\hat{x}$ and is evidently

$$Y_{\Delta}(\hat{x}) = 2 \sum_{j=0}^{n-1} \hat{z}_{\eta_j} \times \left| r_{\eta_j} \right| \tag{12}$$

Some observations may be made about the binary vector $\hat{z}$. The coordinates $\hat{z}_{\eta_j}$ for $j = 0$ to $(k - 1)$ are always equal to zero. The maximum possible weight of $\hat{z}$ is thus $n - k$ and the average weight is $(n - k)/2$ at low E_b/N_o values. At high E_b/N_o values the average weight of $\hat{z}$ is small because there is a high chance that $\hat{x}$ is equal to the transmitted codeword. It may be seen from (12) that, in general, the lower the weight of $\hat{z}$ the smaller will be $Y_{\Delta}(\hat{x})$ and the larger will be the correlation value $Y(\hat{x})$.

Since there is no guarantee that the codeword $\hat{x}$ is the transmitted codeword, the decoder has to evaluate additional codewords since one or more of these may produce a correlation higher than $\hat{x}$. There are $2^k - 1$ other codewords, which may be derived by considering all other $2^k - 1$ sign combinations of c_{η_j} for $j = 0$ to $k - 1$. For any of these codewords denoted as c_i, the first term of the correlation given in (7) is bound to be smaller since

$$\sum_{j=0}^{k-1} r_{\eta_j} \times c_{i,\eta_j} < \sum_{j=0}^{k-1} \left| r_{\eta_j} \right| \tag{13}$$

This is because there has to be, by definition, at least one sign change of c_{i,η_j} compared to $\hat{c}_{\eta_j}$ for $j = 0$ to $k - 1$. In order for $Y(x_i)$ to be larger than $Y(\hat{x})$, the second term of the correlation $\sum_{j=k}^{n-1} r_{\eta_j} \times c_{i,\eta_j}$, which uses the bits from the solved parity check equations, must be larger than $\sum_{j=k}^{n-1} r_{\eta_j} \times \hat{c}_{\eta_j}$ plus the negative contribution from the first term.

However, the first term has higher received magnitudes than the second term because the received coordinates are ordered. It follows that codewords likely to have a higher correlation than $\hat{x}$ will have small number of differences in the coordinates x_{η_j} for $j = 0$ to $k - 1$. As the code is linear, these differences will correspond to a codeword and codewords may be generated that have low weight in coordinates x_{η_j} for $j = 0$ to $k - 1$. These codewords are represented as $\tilde{x}_i$ and referred to as low information weight codewords since coordinates x_{η_j} for $j = 0$ to $k - 1$ form an information set. Thus, codewords x_i are given by

$$x_i = \hat{x} \oplus \tilde{x}_i \tag{14}$$

and $\tilde{x}_i$ are codewords chosen to have increasing weight in coordinates x_{η_j} for $j = 0$ to $k - 1$ as i is incremented. This means that for increasing i it will become less likely that a codeword will be found that has higher correlation than the correlation of a codeword already found.

The difference in the correlation value $Y_\Delta(x_i)$ as a function of $\tilde{x}_i$ may be derived. First, the binary vector z_i is given by

$$z_i = b \oplus \hat{x} \oplus \tilde{x}_i \tag{15}$$

which may be simplified to

$$z_i = \hat{z} \oplus \tilde{x}_i \tag{16}$$

The cross-correlation $Y(x_i)$ is given by

$$Y(x_i) = Y_{\max} - 2\sum_{j=0}^{n-1} z_{i,\eta_j} \times |r_{\eta_j}| \tag{17}$$

equivalently

$$Y(x_i) = Y_{\max} - Y_\Delta(x_i) \tag{18}$$

The shortfall from maximum correlation, $Y_\Delta(x_i)$, is evidently

$$Y_\Delta(x_i) = 2\sum_{j=0}^{n-1} z_{i,\eta_j} \times |r_{\eta_j}| \tag{19}$$

Substituting for z_i gives $Y_\Delta(x_i)$ as a function of $\tilde{x}_i$.

$$Y_\Delta(x_i) = 2\sum_{j=0}^{n-1} (\hat{z}_j \oplus \tilde{x}_{i,\eta_j}) \times |r_{\eta_j}| \tag{20}$$

It is apparent that instead of the decoder determining $Y(x_i)$ for each codeword, x_i, it is sufficient for the decoder to determine $Y_\Delta(x_i)$ for each codeword $\tilde{x}_i$ and compare the value with the smallest value obtained so far, denoted as $Y_\Delta(x_{\min})$, starting with $Y_\Delta(\hat{x})$

$$Y_\Delta(x_{\min}) = \min\{Y_\Delta(x)\} \tag{21}$$

Thus, it is more efficient for the decoder to compute the correlation (partial sum) of the $\tilde{x}_i$ instead of deriving $(\hat{x} \oplus \tilde{x}_i)$ by solving $\boldsymbol{H}_I$ and computing the squared Euclidean distance. Since codewords $\tilde{x}_i$ produce low weight in z_i, the number of non-zero terms that need to be evaluated in (19) is typically $(n - k)/2$ rather than the $n/2$ terms of (2), which makes for an efficient, fast decoder. Before (20) is evaluated, the Hamming weight of z_i may be compared to a threshold, w_{th}, and the correlation stage bypassed if the Hamming weight of z_i is high. There is an associated performance loss and results are presented in Section 4.

The maximum information weight $w_{\text{info max}}$ necessary to achieve maximum-likelihood decoding may be upper bounded from $Y_\Delta(\hat{x})$ and $|r_{\eta_j}|$ initially, updated by $Y_\Delta(x_{\min})$ as decoding progresses, since

$$Y_\Delta(x_i) \geq \sum_{m=0}^{w_{\text{info}}} \left| r_{\eta_{k-m-1}} \right| \tag{22}$$

This is reasonably tight since there is a possibility of at least one codeword with information weight $w_{\text{info max}}$, for which all of the coordinates of the binary vector z_i corresponding to the parity bits of $\tilde{x}_i$ are zero. Correspondingly $w_{\text{info max}}$ is the smallest integer such that

$$\sum_{m=0}^{w_{\text{infomax}}} \left| r_{\eta_{k-m-1}} \right| \geq Y_\Delta(\hat{x}) \tag{23}$$

The codewords $\tilde{x}_i$ may be most efficiently derived from the $\boldsymbol{G}$ matrix corresponding to the solved $\boldsymbol{H}$ matrix because the maximum information weight given by (23) is small. Each row, i, of the solved $\boldsymbol{G}$ matrix is derived by setting x_{η_j} for $j=0$ to $k - 1$, $j \neq i$ and using the solved parity check equations to determine x_{η_j} for $j = k$ to $n - 1$. The maximum number of rows of the $\boldsymbol{G}$ matrix that need to be combined to produce $\tilde{x}_i$ is $w_{\text{info max}}$.

In terms of implementation, the incremental correlation Dorsch decoder can be realised in an efficient decoder structure and this is depicted in Fig. 1. The low weight information vectors are generated by the revolving-door algorithm, which has the property that between two successive combinations, there is only one element that is changed; one leaves and the other one enters the combination [20, 21]. Therefore given a new codeword, the correlation value for the information portion can be easily computed by subtracting $|r_{out}|$ and adding $|r_{in}|$, where *out* is the coordinate that leaves and *in* is the coordinate that enters the combination. The correlation of the parity check term can be easily computed by knowing the support of $(\hat{z}_j \oplus \hat{x}_{i,\eta_j})$ for $j = k$ to $n - 1$. This incremental correlation version of the Dorsch decoder gives two levels of performance improvements and complexity trade-offs. Firstly, it is obvious that the number of processed low weight information vectors is directly proportional to the decoder's complexity, and also the higher the number of processed low weight information vectors the more probable that the decoder will find the maximum-likelihood codeword. An example of this feature will be discussed in Section 4. The second level of performance trade-off is given by w_{th}, the threshold for the weight of z_i. The higher the weight of z_i, the less probable that this codeword is the maximum-likelihood codeword. One can select a value for w_{th} and vectors z_i that have weights higher than w_{th} will be ignored and only sets of low-weight information vectors will be considered. From simulations, it is observed that the loss associated with this trade-off is insignificant, and the performance gain can be large for a fixed number of evaluated codewords.

3 Number of codewords that need to be evaluated to achieve maximum-likelihood decoding

For each received vector, the decoder needs to evaluate the correlation shortfall for the codewords $\tilde{x}_i$ for information weights up to the maximum information weight of $w_{\text{info max}}$ in order to achieve maximum-likelihood decoding. The number of codewords that need to be evaluated is a function of the received vector. Not all of the codewords having information weight less than or equal to $w_{\text{info max}}$ need be evaluated because a lower bound may be derived for $Y_\Delta(x_i)$ in terms of the coordinates of the information bits, their total weight and the magnitudes of selected coordinates of the received vector. For an information weight of w_{info}, $Y_\Delta(x_i)$ is lower bounded by

$$Y_\Delta(x_i) \geq \left| r_{\eta_j} \right| + \sum_{m=0}^{w_{\text{info}}-1} \left| r_{\eta_{k-m-1}} \right|, \quad 0 \leq j < k - m \tag{24}$$

and

$$\left| r_{\eta_{j_{\max}(w_{\text{info}})}} \right| \geq Y_\Delta(x_i) - \sum_{m=0}^{w_{\text{info}}-1} \left| r_{\eta_{k-m-1}} \right|, \quad 0 \leq j < k - m \tag{25}$$

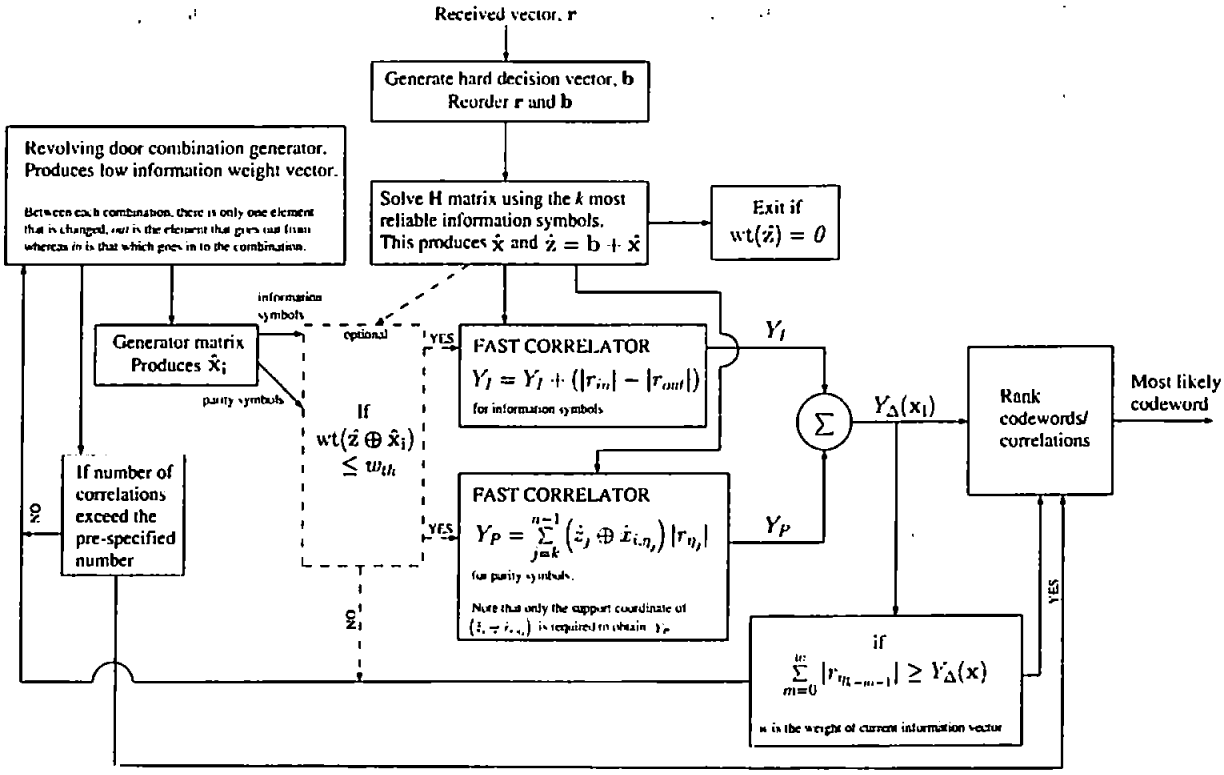


Fig. 1 *Structure of the efficient Dorsch decoder*

where $j_{\min}(w_{\text{info}})$ is defined as the lower limit for j to satisfy (25). The minimum number of codewords that need to be evaluated as a function of the received vector $N(r)$ is given by the total number of combinations

$$N(r) = \sum_{m=0}^{w_{\text{min}}} \binom{k - j_{\min}(m) - 1}{m} \tag{26}$$

For many short codes the minimum number of codewords that need to be evaluated is surprisingly small in comparison to the total number of codewords.

4 Results for some powerful binary codes

The decoder can be used with any linear code and best results are obtained for codes, which have the highest known $d_{\min}$ for a given code-length n and number of information symbols k. The best binary codes are tabulated up to length 256 in Brouwer's database [22] (the updated version of the database is available online at http://www.win.tue.nl/~aeb/voorlincod.html). Non-binary codes, for example, ternary codes of length up to 243 symbols and GF(4) codes of length up to 256 symbols are also tabulated.

A particularly good class of codes are the binary self-dual, double circulant codes first highlighted in a classic paper by Karlin [23]. For example, the (24, 12, 8) extended Golay code is included since it may be put in double circulant form. There is also the (48, 24, 12) bordered double circulant code, based on quadratic residues of the prime 47 and the (136, 68, 24) bordered double circulant code based on quadratic residues of the prime 67. These codes are extremal and are doubly even, only having codeword weights that are a multiple of 4, and in these cases it is necessary that the code-lengths are a multiple of 8 [24]. For higher code rates of length greater than 256 the best codes are tabulated in [17], and some of these include cyclic codes and Goppa codes.

4.1 (136, 68, 24) Double circulant code

This code is a bordered double circulant code based on the identity matrix and a matrix whose rows consist of all cyclic combinations, modulo $1 + x^{67}$, of the polynomial $b(x)$ defined by

$$\begin{aligned} b(x) = {} & 1 + x + x^4 + x^6 + x^9 + x^{10} + x^{14} + x^{15} + x^{16} \\ & + x^{17} + x^{19} + x^{21} + x^{22} + x^{23} + x^{24} + x^{25} + x^{26} \\ & + x^{29} + x^{33} + x^{35} + x^{36} + x^{37} + x^{39} + x^{40} + x^{47} \\ & + x^{49} + x^{54} + x^{55} + x^{56} + x^{59} + x^{60} + x^{62} + x^{64} + x^{65} \end{aligned} \tag{27}$$

The frame error rate (FER) of this code using the extended Dorsch decoder with a maximum number of codewords limited to 1×10^7 is shown in Fig. 2. Also shown in Fig. 2 is Shannon's sphere packing bound [18] offset by the loss for binary transmission [25], which is 0.19 dB for a code rate of 1/2.

It may be seen from Fig. 2 that the performance of the decoder in conjunction with the double circulant code is within 0.2 dB of the best achievable performance for any (136, 68) code at 10^{-5} FER. Interestingly, there is a significant number of maximum-likelihood codeword errors, which have a Hamming distance of 36 or 40 from the transmitted codeword. This indicates that a bounded distance decoder would not perform very well for this code. At the operating points of E_b/N_o equal to 1.5 and 3.5 dB, the probability of the decoder processing each received vector as a maximum-likelihood decoder is shown plotted in Fig. 3 as a function of the number of codewords evaluated. A typical, worst case, practical operating point for this code is E_b/N_o equal to 3.5 dB. It is evident that the higher the E_b/N_o value, the less will be the number of codewords required by the decoder to evaluate. Another illustration of

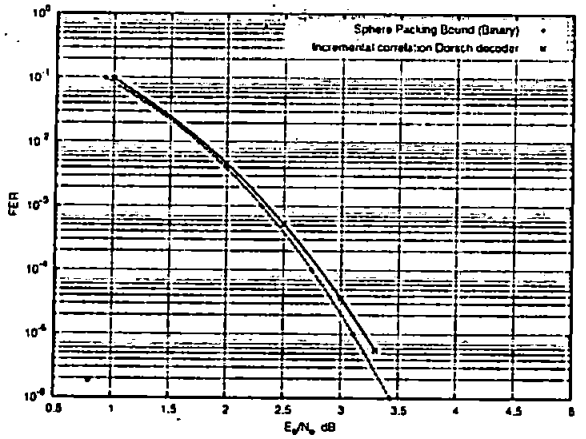


Fig. 2 *FER performance of the (136, 68, 24) double circulant code*

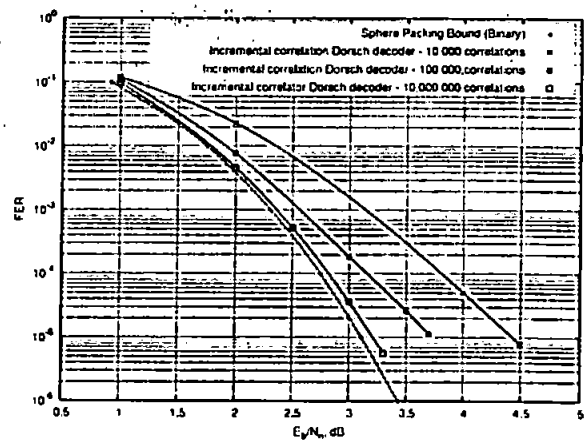


Fig. 4 *FER performance of the (136, 68, 24) code as a function of number of evaluated codewords*

this point is given in Fig. 4, which shows for each received vector, the average number of codewords processed by the decoder before the maximum-likelihood codeword is found as a function of E_b/N_o. For example, at E_b/N_o value of 2.5 dB, on average, approximately 250 codewords are processed per received vector to find the maximum-likelihood codeword.

Of course to guarantee maximum-likelihood decoding, all $2^{68} = 2.95 \times 10^{20}$ codewords need to be evaluated by the decoder. Equation (22) has been evaluated for the double circulant (136, 68, 24) code in computer simulations, at an E_b/N_o of 3.5 dB, for each received vector and the cumulative distribution derived. Fig. 3 shows that by evaluating 10^7 codewords per received vector, 65% of received vectors are guaranteed to be maximum-likelihood decoded. For the remaining 35% of received vectors, although maximum-likelihood decoding is not guaranteed, the probability is very small that the codeword with the highest correlation is not the transmitted codeword or a codeword closer to the received vector than the transmitted codeword. This last point is illustrated by Fig. 5, which shows the FER performance of the decoder as a function of the maximum number of evaluated codewords.

The detailed operation of the decoder may be seen by considering an example of a received vector at E_b/N_o of 2.5 dB. The magnitudes of the received coordinates, ordered in their solved order, is shown in Fig. 6. In this particular example, it is not possible to solve for ordered coordinates 67 and 68 (in their order prior to solving of the parity check matrix) and so these coordinates are skipped and become coordinates 68 and 69, respectively, in the solved order. The transmitted bits are normalised with magnitudes 1 and the σ of the noise is $\simeq 1.07$. The shift in position of coordinate 69 (in original position) to 67 (in solved order) is evident in Fig. 6. The positions of the bits received in error in the same solved order is shown in Fig. 7. It may be noted that the received bit errors are concentrated in the least reliable bit positions. There are a total of 16 received bit errors and only 2 of these errors correspond to the (data) bit coordinates 11 and 34 of the solved $\boldsymbol{G}$ matrix. Evaluation of 10^7 codewords indicates that the minimum value of $Y_\Delta(x_{min})$ is $\simeq 13.8$ and this occurs for the 640th codeword producing a maximum correlation of $\simeq 126.2$ with $Y_{max} \simeq 140$. The weight of z_{min} is 16 corresponding to the 16 received bit errors.

In practice it is not necessary for $Y_\Delta(x_i)$ given by the partial sum, that is, (19), to be evaluated for each codeword. In most cases, the weight of the binary vector z_i is sufficiently high to indicate that this codeword is not the most likely codeword. Shown in Fig. 8 are the cumulative probability distributions for the weight of z_i for the case where x_i is equal to the transmitted codeword and the case where it is not equal to the transmitted codeword. Two operating values for E_b/N_o are shown: 3.5 and 4 dB. Considering the decoding rule that a weight 29 or more for z_i is unlikely

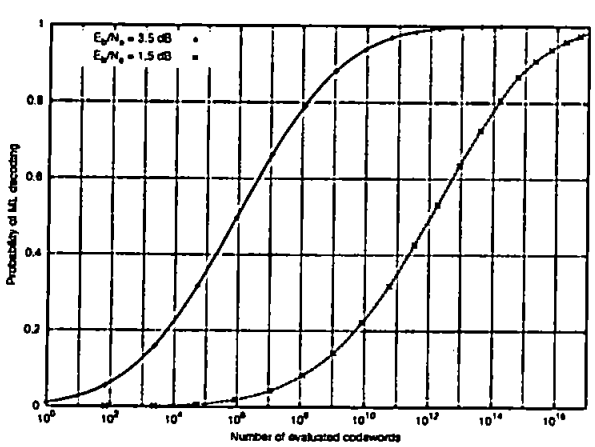


Fig. 3 *Probability of a received vector being maximum likelihood decoded as a function of number of evaluated codewords for the (136, 68, 24) code*

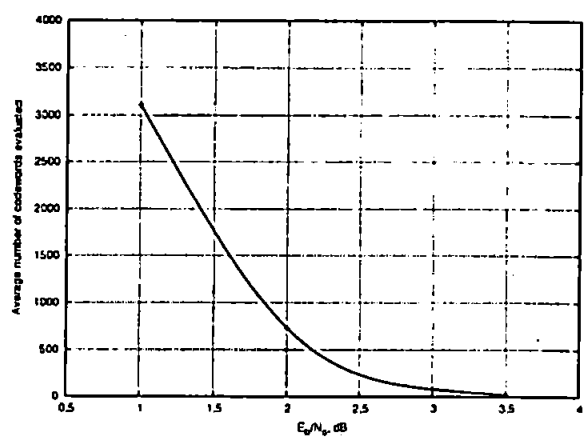


Fig. 5 *Average number of codewords evaluated in order for a received vector to be maximum likelihood decoded as a function of E_b/N_o for (136, 68, 24) code*

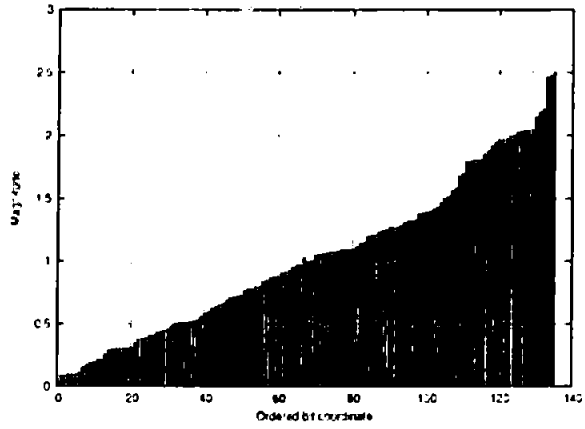


Fig. 6 *Example of received coordinate magnitudes in their solved order for the (136, 68, 24) code at $E_b/N_o = 2.5$ dB for a single received vector*

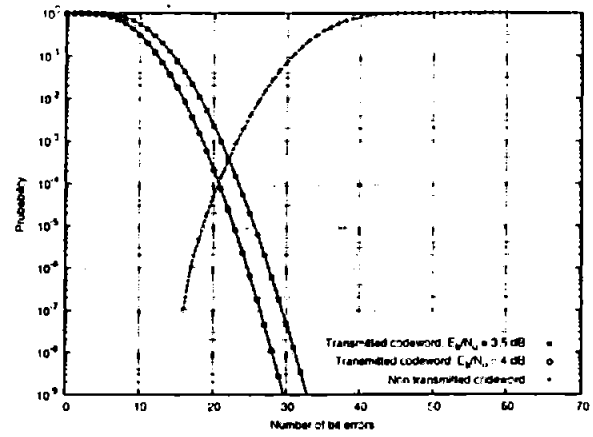


Fig. 8 *Cumulative probability distributions for the number of bit errors for the transmitted codeword and non-transmitted, evaluated codewords for the (136, 68, 24) code*

to be produced by the transmitted codeword means that 95.4% of candidate codewords may be rejected at this point and that the partial sum (19) need only be evaluated for 4.6% of the candidate codewords. In reducing the decoder complexity in this way, the degradation to the FER performance as a result of rejection of a transmitted codeword corresponds to ≃3% increase in the FER and is not significant.

4.2 (255, 175, 17) Euclidean geometry code

This code is an Euclidean geometry (EG) code originally used in hard decision, one step majority logic decoding [4]. In [26], it is shown that finite geometry codes can also be applied as low-density parity check (LDPC) codes using belief propagation, iterative decoding [27, 28]. The (255, 175, 17) code is a cyclic code and its parity check polynomial $p(x)$ may conveniently be generated from the cyclotomic idempotents [29] and is given by

$$p(x) = 1 + x + x^3 + x^7 + x^{15} + x^{26} + x^{31} + x^{53} + x^{63} + x^{98} + x^{107} + x^{127} + x^{140} + x^{176} + x^{197} + x^{215} \quad (28)$$

In cases where the parity check matrix of the code is sparse, the performance of the incremental correlation Dorsch decoder can be improved by utilising the extrinsic information obtained from each parity check equation corresponding to each row of the parity check matrix of the code. For each received vector, the belief propagation algorithm with a single iteration is executed and the a posteriori probabilities at the output of the belief propagation decoder are taken as the input to the incremental correlation Dorsch decoder. The (255, 175, 17) EG code has a relatively sparse parity check matrix, and by taking into account the extrinsic information obtained in this way, an improvement in FER as much as 50%, at $E_b/N_o = 3.5$ dB, has been observed. Fig. 9 shows the FER performance of this EG code obtained using this combined technique in which the maximum number of correlations was set to 5.5×10^6. Also shown in Fig. 9 is the FER performance obtained using the belief propagation decoder on its own with 100 iterations, and the sphere packing bound offset by the binary transmission loss.

The benefits of extrinsic information were first proposed by Fossorier in [30] for decoding LDPC codes, in a more complicated arrangement in which the order-1 or order-2 OSD algorithm is executed after every iteration of the belief propagation decoder.

As a general point, it should be noted that the effect of utilising extrinsic information in the Dorsch decoder is to reduce, on average, the number of codewords that need to be processed in order to find the maximum-likelihood codeword. However, for the occasional received vector, utilising the extrinsic information introduces more errors in the k most likely bits than not utilising the extrinsic information. Accordingly, the advantages of utilising extrinsic

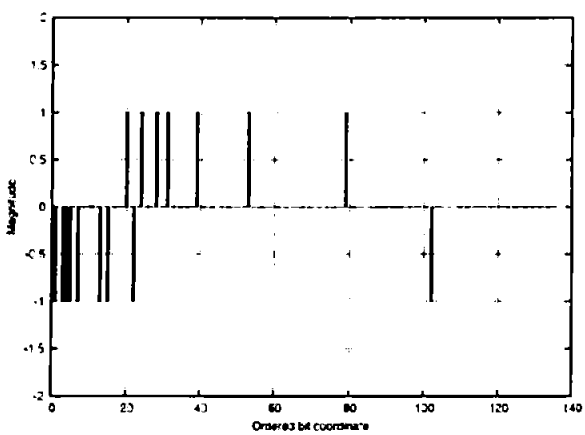


Fig. 7 *Bit error positions for the same received vector and same order as that shown in Fig. 6*

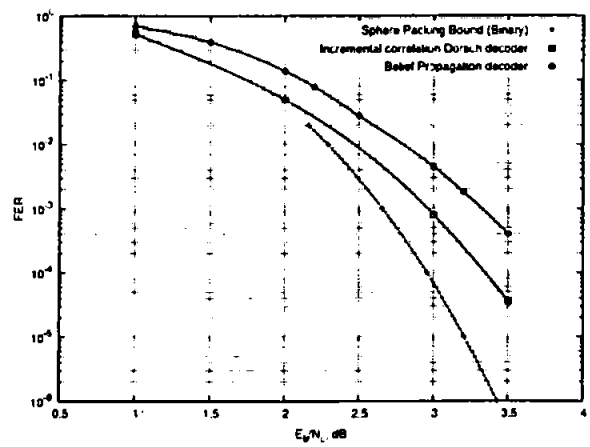


Fig. 9 *FER of the (255, 175, 17) EG code*

information depends upon the particular code being used and the maximum number of codewords processed per received vector. Extrinsic information should not be utilised if the the maximum number of codewords is such that near maximum-likelihood performance is already achieved by the decoder. In this case, if extrinsic information is utilised the performance of the decoder will be degraded.

4.3 (513, 467, 12) Extended binary Goppa code

Goppa codes are frequently better than the corresponding BCH codes because there is an additional information bit and the Goppa code is only one bit longer than the BCH code. For example the (512, 467, 11) binary Goppa code has one more information bit than the (511, 466, 11) BCH code and is generated by the irreducible polynomial $1+x^2+x^5$ whose roots have order 31, which is relatively prime to 511. The $d_{\min}$ of the binary Goppa code is equal to twice the degree of the irreducible polynomial plus 1 [17], and is the same as the (511, 466, 11) BCH code. The Goppa code may be extended by adding an overall parity check, increasing the $d_{\min}$ to 12.

The FER performance of the extended code is shown in Fig. 10 and was obtained using the incremental correlation decoder. Also shown in Fig. 10 is the sphere packing bound offset by the binary transmission loss. It can be seen that the realised performance of the decoder is within 0.3 dB at 10^{-4} FER.

4.4 (1023, 983, 9) BCH code

This code is a standard BCH code that may be found in reference text book tables such as [4]. This example is considered here in order to show that the decoder can produce near maximum-likelihood performance for relatively long codes. The performance obtained is shown in Fig. 11 with evaluation of candidate codewords limited to 10^6 codewords. At 10^{-5} FER, the degradation from the sphere packing bound, offset for binary transmission, is 1.8 dB. Although this may seem excessive, the degradation of hard decision decoding is 3.6 dB as may also be seen from Fig. 11.

5 Extension to non-binary codes

The extension of the decoder to non-binary codes is relatively straightforward and for simplicity binary transmission of the components of each non-binary symbol is assumed. Codewords are denoted as before by x_i but

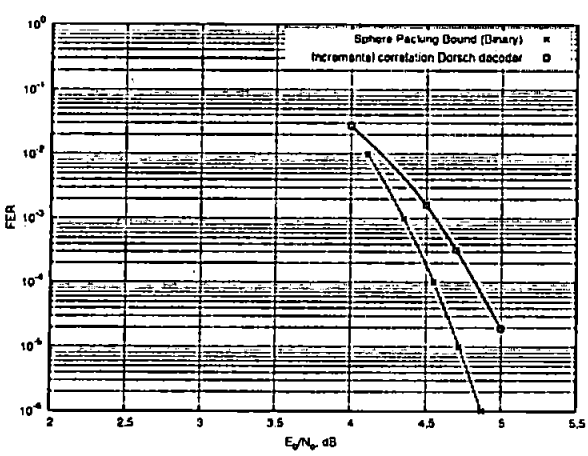


Fig. 10 *FER of the (513, 467, 12) binary Goppa code*

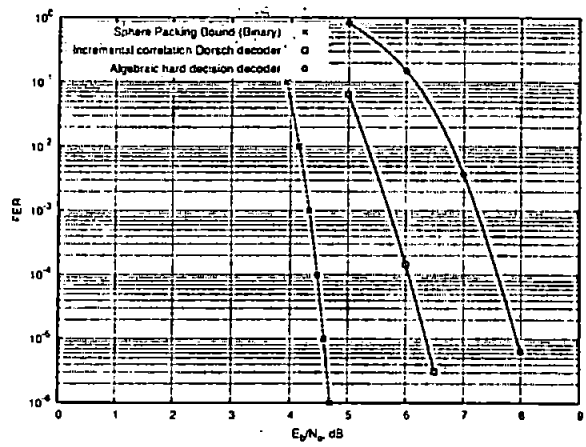


Fig. 11 *FER of the (1023, 983, 9) binary BCH code*

redefined with coefficients, $\gamma_{j,i} \in \mathrm{GF}(2^m)$

$$x_i = (\gamma_{01,i}x_0, \gamma_{1,i}x_1, \gamma_{2,i}x_2, \ldots, \gamma_{n-1,i}x_{n-1}) \tag{29}$$

The received vector r with coordinates ranked in order of those most likely to be correct is redefined as

$$r = \sum_{l=0}^{m-1}(r_{l,\mu_0}, r_{l,\mu_1}, r_{l,\mu_2}, \ldots, r_{l,\mu_{n-1}}) \tag{30}$$

so that the received vector consists of n symbols, each with m values. The maximum attainable correlation $Y_{\max}$ is straightforward and is given by

$$Y_{\max} = \sum_{j=0}^{n-1}\sum_{l=0}^{m-1}\left|r_{l,j}\right| \tag{31}$$

The hard decided received vector r_s is redefined as

$$b = \sum_{j=0}^{n-1}\theta_j x^j \tag{32}$$

where θ_j is the $\mathrm{GF}(2^m)$ symbol corresponding to $\mathrm{sign}(r_{l,j})$ for $l=0$ to $m-1$.

Decoding follows in a similar manner to the binary case. The received symbols are ordered in terms of their symbol magnitudes $|r_{\mu}j|_S$ where each symbol magnitude is defined as

$$\left|r_{\eta_l}\right|_S = \sum_{l=0}^{m-1}\left|r_{l,\eta_l}\right| \tag{33}$$

The codeword $\hat{x}$ is derived from the k coordinates x_{η_l} whose coefficients v_{η_l} are the $\mathrm{GF}(2^m)$ symbols corresponding to $\mathrm{sign}(r_{l,\eta_l})$ for $l=0$ to $m-1$; for $j=0$ to $k-1$ and then using the solved parity check equations for the remaining $n-k$ coordinates.

The vector z_i is given by

$$z_i = b \oplus \hat{x} \oplus \tilde{x}_i \quad \mathrm{mod\ GF}(2^m) \tag{34}$$

which may be simplified as before to

$$z_i = \hat{z} \oplus \tilde{x}_i \quad \mathrm{mod\ GF}(2^m) \tag{35}$$

Denoting the n binary vectors $\rho_{i,l,j}$ corresponding to the n $\mathrm{GF}(2^m)$ coefficients of z_i

$$Y(x_i) = Y_{\max} - Y_{\Delta}(x_i) \tag{36}$$

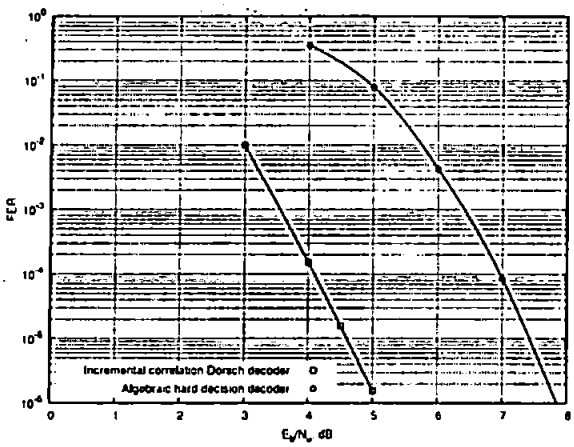


Fig. 12 *FER of the (63, 36, 13) GF(4) BCH code*

where $Y_\Delta(x_i)$, the shortfall from maximum correlation is given by

$$Y_\Delta(x_i) = 2\sum_{j=0}^{n-1}\sum_{l=0}^{m-1} \rho_{i,l,j} \times \left|r_{l,j}\right| \qquad (37)$$

In the implementation of the decoder, as in the binary case, the Hamming weight of the vector z_i may be used to decide whether it is necessary to evaluate the soft-decision metric given by (37) for each candidate codeword.

5.1 Results for the (63, 36, 13) GF(4) BCH Code

This is a non-binary BCH code with the generator polynomial $g(x)$ defined by roots

$$\alpha^1\alpha^4\alpha^{16}\alpha^2\alpha^8\alpha^{32}\alpha^3\alpha^{12}\alpha^{48}\alpha^5\alpha^{20}\alpha^{17}\alpha^6\alpha^{24}\alpha^{33}\alpha^7\alpha^{28}$$
$$\alpha^{29}\alpha^9\alpha^{36}\alpha^{18}\alpha^{10}\alpha^{40}\alpha^{34}\alpha^{11}\alpha^{44}\alpha^{50}$$

The benefit of having GF(4) coefficients is that $g(x)$ does not need to contain the roots

$$\alpha^{14}\alpha^{56}\alpha^{35}\alpha^{22}\alpha^{25}\alpha^{37}$$

which are necessary to constrain $g(x)$ to binary coefficients [17]. Correspondingly, the binary version of this BCH code is the lower rate (63, 30, 13) code with six less information symbols (bits).

The performance of the (63, 36, 13) GF(4) BCH code is shown in Fig. 12 for the binary AWGN channel. Also shown in Fig. 12 is the performance of the code with hard decision decoding. It may be seen that at 10^{-4} FER the performance of the incremental correlation decoder is 2.9 dB better than the performance of the hard-decision decoder.

6 Conclusions

It has been shown that the extended Dorsch decoder may approach maximum-likelihood decoding by an incremental correlation approach in which for each received vector a partial summation metric is evaluated as a function of low information weight codewords. Furthermore, the number of information weight codewords that need to be evaluated to achieve maximum-likelihood decoding may be calculated as an upper upper bound for each received vector. Consequently for each received vector it is known whether the decoder has achieved maximum-likelihood decoding. An efficient decoder structure consisting of a combination of hard decision threshold decoding followed by partial sum correlation was also described, which enables practical decoders to trade-off performance against complexity.

The decoder for non-binary codes was shown to be straightforward for the binary transmission channel and an example given for a GF(4) BCH code. It is readily possible to extend the decoder to non-binary modulation by extensions to the incremental correlation of (37) although this inevitably involves an increase in complexity. Future work will address this area.

Another interesting conclusion is just how well some codes in Brouwer's table perform with maximum-likelihood decoding. In particular, the (136, 68, 24) double circulant, extremal, self-dual code is shown to be an outstanding code.

In the opinion of the authors, the implementation of this type of decoder coupled with the availability of powerful processors will eventually herald a new era in the application of error control coding with the re-establishment of the importance of the optimality of codes rather than the ease of decoding. Certainly this type of decoder is more complex than an iterative decoder, but the demonstrable performance, that is achievable for short codes, is virtually the same as the sphere packing bound.

7 Acknowledgment

This work was partly funded by an Overseas Research Students Award Scheme (ORSAS). The authors would like to thank the reviewers for their comments which have enhanced the quality of presentation of this paper. The rapid and efficient processing of this paper by the Editorial Office is also much appreciated.

8 References

1 Dorsch, B.G.: 'A decoding algorithm for binary block codes and J-ary output channels', *IEEE Trans. Inf. Theory*, 1974, 20, pp. 391–394
2 Berrou, C., Glavieux, A., and Thitimajshima, P.: 'Near shannon limit error-correcting coding: turbo codes'. Proc. IEEE. Int. Conf. Communication, Geneva, Switzerland, May 1993, pp. 1064–1070
3 Sweeney, P., and Wesemeyer, S.: 'Iterative soft-decision decoding of linear block codes', *IEE Proc. Commun.*, 2000, 147, (3), pp. 133–136
4 Lin, S., and Costello, Jr., D.J.: 'Error control coding: Fundamentals and applications' (Pearson Education, Inc. 2nd edn.)2004) (ISBN 0 13 017973 6)
5 Fossorier, M., and Lin, S.: 'Soft-decision decoding of linear block codes based on ordered statistics', *IEEE Trans. Inf. Theory*, 1995, 41, (5), pp. 1379–1396
6 Fossorier, M., and Lin, S.: 'Computationally efficient soft-decision decoding of linear block codes based upon ordered statistics', *IEEE Trans. Inf. Theory*, 1996, 42, pp. 738–750
7 Fossorier, M., and Lin, S.: 'Reliability-based information set decoding of binary linear codes', *IEICE Trans. Fundam.*, 1999, E82-A, pp. 2034–2042
8 Fossorier, M.: 'Reliability-based soft-decision decoding with iterative information set reduction', *IEEE Trans. Inf. Theory*, 2002, IT-48, pp. 3101–3106
9 Isaka, M., Fossorier, M., and Imai, H.: 'On the suboptimality of iterative decoding for turbo-like and ldpc codes with cycles in their tanner graph representation', *IEEE Trans. Commun.*, 2004, 52, (5), pp. 845–854
10 de Barros, D.J., Godoy, Jr., W., and Wille, E.C.G.: 'A new approach to the information set decoding algorithm', *Comput. Commun.*, 1997, 20, pp. 302–308
11 de Barros, D.J., and Dorsch, B.G.: 'An efficient information set algorithm for soft-decision decoding of linear block codes'. Proc. IEEE Int. Symp. Information Theory, Ulm, Germany, 29 June–4 July 1997, p. 302
12 Chase, D.: 'A class of algorithms for decoding block codes with channel measurement information', *IEEE Trans. Inf. Theory*, 1972, IT-18, pp. 170–182

13 Kaneko, T., Nishijima, T., Inazumi, H., and Hirasawa, S.: 'An efficient maximum likelihood decoding of linear block codes with algebraic decoder', *IEEE Trans. Inf. Theory*, 1994, IT-40, pp. 320–327
14 Snyders, J.: 'Reduced lists of error patterns for maximum likelihood soft decision decoding', *IEEE Trans. Inf. Theory*, 1991, 37, pp. 1194–1200
15 Lous, N.J.C., Bours, P.A.H., and van Tilborg, H.C.A.: 'On maximum likelihood soft-decision decoding of binary linear codes', *IEEE Trans. Inf. Theory*, 1993, 39, pp. 197–203
16 Tomlinson, M., Tjhai, C., Ambroze, M., and Ahmed, M.: 'Improved error correction decoder using ordered symbol reliabilities'. UK Patent Application GB0637114.3, 2005
17 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error-correcting codes' (North-Holland, 1977), (ISBN 0 444 85193 3)
18 Shannon, C.E.: 'Probability of error for optimal codes in a gaussian channel', *Bell Syst. Tech. J.*, 1959, 38, (3), pp. 611–656
19 Proakis, J.G.: 'Digital communications' (McGraw-Hill, New York, 2001, 4th edn.), (ISBN 0 071 18183 0)
20 Nijenhuis, A., and Wilf, H.S.: 'Combinatorial algorithms for computers and calculators' (Academic Press, London, 1978, 2nd edn.)
21 Bitner, J.R., Ehrlich, G., and Reingold, E.M.: 'Efficient generation of the binary reflected gray code and its applications', *Commun. ACM*, 1976, 19, (9), pp. 517–521
22 Brouwer, A.E.: 'Bounds on the size of linear codes' in Pless, V.S., and Huffman, W.C. (Eds.): 'Handbook of coding theory' (Elsevier, North Holland, 1998), pp. 295–461
23 Karlin, M.: 'New binary coding results by circulants', *IEEE Trans. Inf. Theory*, 1969, 15, (1), pp. 81–92
24 Conway, J.H., and Sloane, N.J.A.: 'A new upper bound on the minimum distance of self-dual codes', *IEEE Trans. Inf. Theory*, 1990, 36, (6), pp. 1319–1333
25 Butman, S., and McEliece, R.J.: 'The ultimate limits of binary coding for a wideband gaussian channel'. JPL Deep Space Network Progress Report 42-22, 1974, pp. 78–80
26 Kou, Y., Lin, S., and Fossorier, M.: 'Low density parity check codes based on finite geometries: a rediscovery and new results', *IEEE Trans. Inf. Theory*, 2001, 47, pp. 2711–2736
27 Pearl, J.: 'Probabilistic reasoning in intelligent systems: networks of plausible inference' (Morgan Kaufmann, San Mateo, CA, 1988)
28 McEliece, R.J., MacKay, D.J.C., and Cheng, J.F.: 'Turbo decoding as an instance of pearl's belief propagation algorithm', *IEEE J. Sel. Areas Commun.*, 1998, 16, pp. 140–152
29 Tjhai, C., Tomlinson, M., Ambroze, M., and Ahmed, M.: 'Cyclotomic idempotent-based binary cyclic codes', *Electron. Lett.*, 2005, 41, (3), pp. 341–343
30 Fossorier, M.: 'Iterative reliability-based decoding of low-density parity check codes', *IEEE J. Sel. Areas Commun.*, 2001, JSAC-19, pp. 908–917

Results on binary cyclic codes

C. Tjhai and M. Tomlinson

Thirty-seven binary cyclic codes with minimum distance higher than those of the best linear codes given in Brouwer's table are presented. Among these new cyclic codes is the quadratic residue code of length 199, for which is provided an answer to the long-open question regarding its minimum distance. Four new binary linear codes are also obtained by applying Construction X and Construction Y1 to those new cyclic codes. Overall, after taking into account the extended, punctured and shortened codes, there are 869 binary linear codes which are improvements to Brouwer's lower-bound.

Introduction: Let C be a binary linear code with parameters $[n, k, d]$, where n, k and d represent the length, dimension and minimum Hamming distance of the code, respectively. The minimum distance of a code is an important parameter since it determines the error-correcting abilities of the code. Brouwer has constructed a database containing the lower-bounds and upper-bounds of minimum distance of linear codes over finite-fields (updated version available at http://www.win.tue.nl/~aeb/voorlincod.html, accessed 20 December 2006) [1]. The lower-bound corresponds to the largest minimum distance for a given n and k that has been found and verified to date. Constructing codes which are improvements over those in Brouwer's database is an ongoing research activity in coding theory. Recently, tables of lower- and upper-bounds of not only codes over finite-fields, but also quantum error-correcting codes, have been published [2]. These bounds for codes over finite-fields, which are derived from MAGMA [3], appear to be more up-to-date than those of Brouwer.

This Letter presents some results on binary cyclic codes ($195 \le n \le 255$) which have higher minimum distance than Brouwer's lower-bounds, denoted by $d_{Brouwer}$, and also in many cases the lower-bounds in [2], for the same n and k. In addition to the new codes, this Letter also answers a long-standing open question regarding the minimum distance of quadratic residue (QR) codes of length 199.

Idempotents and cyclic codes: A polynomial $\theta(x) \in \mathbb{F}_2/(x^n - 1)$ is called an idempotent if it satisfies the property $\theta(x)^2 = \theta(x)$. The use of idempotents in the study of cyclic codes is well-established, see e.g. [4, 5]. The basis of this theory is the primitive idempotents. Any cyclic code may be described by a unique idempotent, which is a sum of primitive idempotents.

A cyclic code C of odd length n has generator and parity-check polynomials, denoted by $g(x)$ and $h(x)$, respectively, where $g(x)h(x) = 0 \pmod{x^n - 1}$. Let β be a primitive nth root of unity and let Λ be a set containing all distinct (excluding conjugates) exponents of β. The polynomial $x^n - 1$ can be factorised into irreducible polynomials $f_i(x)$ over GF(2), $x^n - 1 = \Pi_{i \in \Lambda} f_i(x)$. For notational purpose, we denote the irreducible polynomial $f_i(x)$ as the minimal polynomial of β^i. For each $f_i(x)$, there exists a corresponding primitive idempotent $\phi_i(x)$, which can be obtained as follows [4]

$$\theta_i(x) = \frac{x^n - 1}{f_i(x)} \left(\delta f_i(x) + x \frac{d}{dx} f_i(x) \right) \tag{1}$$

where $\delta = \deg(f_i(x)) \pmod 2$ and the derivative is taken over GF(2). The notation $\deg(a(x))$ denotes the degree of $a(x)$. Given a set $\Gamma \subseteq \Lambda$ and an idempotent $\theta(x) = \Sigma_{i \in \Gamma} \theta_i(x)$, a cyclic code C defined by $\theta(x)$ has β_i, for $i \in \Gamma$, as roots of its $h(x)$ and it follows that $h(x) = \gcd(\theta(x), x^n - 1)$.

Let $\Gamma' \subseteq \Lambda \setminus \{0\}$, $\theta'(x) = \Sigma_i \subseteq \Gamma' \theta_i(x)$ and $\theta(x) = \theta_0(x) + \theta'(x)$. Given C with idempotent $\theta(x)$, there exists an $[n, k-1, d']$ expurgated cyclic code, C', which is defined by idempotent $\theta'(x)$ and $\mathrm{wt}(c) \equiv 0 \pmod 2$ for all $c \subseteq C'$, where $\mathrm{wt}(x)$ denotes the weight of vector x. For convenience, we call C the augmented code of C'.

Evaluation procedures: Consider an $[n, k-1, d']$ expurgated cyclic code C', let the set $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_r\}$ where, for $1 \le j \le r$, $\Gamma_j \subseteq \Lambda \setminus \{0\}$ and $\Sigma_{i \in \Gamma_j} \deg(f_i(x)) = k - 1$. For each $\Gamma_j \in \Gamma$, we compute $\theta'(x)$ and construct C'. Let G be a generator matrix of the augmented code, C, and without loss of generality, it can be written as

$$G = \frac{G'}{v} \tag{2}$$

where G' is a generator matrix of C' and the vector v is a coset of C' in C. Using the arrangement in (2), we evaluate d' by enumerating codewords $c \subseteq C'$ from G'. The minimum distance of C, d, is simply $\min_c\{d', \mathrm{wt}(c + v)\}$ for all codewords c enumerated. We follow the codeword enumeration algorithm of Chen [6, 7] to evaluate d. Let $d_{Brouwer}$ and $d'_{Brouwer}$ denote the lower-bounds of Brouwer for linear codes of the same length and dimension as those of C and C', respectively. During the enumerations, as soon as $d \le d_{Brouwer}$ and $d' \le d'_{Brouwer}$ the evaluation is terminated and the next Γ_j in Γ is then processed. However, if $d \le d_{Brouwer}$ and $d' > d'_{Brouwer}$ only the evaluation for C is discarded. These procedures continue until improvement is obtained; or the set in Γ has been exhausted, which means that there does not exist $[n, k-1]$ and $[n, k]$ cyclic codes which are improvements to Brouwer's lower-bounds. In cases where it is not feasible to determine the minimum distance of C using a single computer, we switch to a parallel version using grid computers.

Table 1: New binary cyclic codes

$[m(x)]_8$	n	k	d	$d_{Brouwer}$	$\theta(x)$
17277	195	†66/67	42/41	40/40	$[\theta_0] + \theta_3 + \theta_5 + \theta_9 + \theta_{19} + \theta_{39} + \theta_{65} + \theta_{67}$
		†68/69	40/39	39/38	$[\theta_0] + \theta_1 + \theta_3 + \theta_{13} + \theta_{19} + \theta_{35} + \theta_{67} + \theta_{91}$
		†73	38	37	$\theta_0 + \theta_3 + \theta_7 + \theta_{19} + \theta_{33} + \theta_{35} + \theta_{47}$
		†74/75	38/37	36/36	$[\theta_0] + \theta_3 + \theta_7 + \theta_{19} + \theta_{33} + \theta_{35} + \theta_{47} + \theta_{65}$
		78	36	35	$\theta_3 + \theta_7 + \theta_9 + \theta_{11} + \theta_{19} + \theta_{35} + \theta_{39} + \theta_{65}$
13237042705-30057231362-555070452551	199	99/100	32/31	28/28	$[\theta_0] + \theta_1$
6727273	205	†60	48	46	$\theta_5 + \theta_{11} + \theta_{31}$
		†61	46	44	$\theta_0 + \theta_3 + \theta_{11} + \theta_{31}$
3346667657	215	70/71	46/46	44/44	$[\theta_0] + \theta_3 + \theta_{13} + \theta_{15}$
3705317547055	223	74/75	48/47	46/45	$[\theta_0] + \theta_5 + \theta_9$
3460425444467-7544446504147	229	76	48	46	θ_1
6704436621	233	†58/59	60/60	56/56	$[\theta_0] + \theta_3 + \theta_{29}$
150153013	241	†49	68	65	$\theta_0 + \theta_1 + \theta_{21}$
		73	54	53	$\theta_0 + \theta_1 + \theta_3 + \theta_{25}$
435	255	48/49	76/75	75/72	$[\theta_0] + \theta_{47} + \theta_{55} + \theta_{91} + \theta_{95} + \theta_{111} + \theta_{127}$
		50/51	74/74	72/72	$[\theta_0] + \theta_9 + \theta_{13} + \theta_{23} + \theta_{47} + \theta_{61} + \theta_{85} + \theta_{127}$
		52/53	72/72	71/68	$[\theta_0] + \theta_7 + \theta_9 + \theta_{17} + \theta_{47} + \theta_{55} + \theta_{117} + \theta_{127}$
		54/55	70/70	68/68	$[\theta_0] + \theta_3 + \theta_7 + \theta_{23} + \theta_{47} + \theta_{55} + \theta_{65} + \theta_{119} + \theta_{127}$
		56/57	68/68	67/65	$[\theta_0] + \theta_7 + \theta_{27} + \theta_{31} + \theta_{45} + \theta_{47} + \theta_{55} + \theta_{127}$
		58	66	64	$\theta_7 + \theta_{39} + \theta_{43} + \theta_{45} + \theta_{47} + \theta_{55} + \theta_{85} + \theta_{127}$
		60	66	64	$\theta_7 + \theta_{17} + \theta_{23} + \theta_{39} + \theta_{45} + \theta_{47} + \theta_{55} + \theta_{127}$
		62/63	66/65	64/63	$[\theta_0] + \theta_{11} + \theta_{21} + \theta_{47} + \theta_{55} + \theta_{61} + \theta_{85} + \theta_{87} + \theta_{119} + \theta_{127}$
		64/65	64/63	62/62	$[\theta_0] + \theta_{19} + \theta_{31} + \theta_{39} + \theta_{47} + \theta_{55} + \theta_{65} + \theta_{91} + \theta_{127}$

New binary codes: Table 1 presents the results of the search on new binary cyclic codes for $195 \le n \le 255$ described earlier. Note that the polynomial $m(x)$, which is given in octal with the most significant bit on the left, is the minimal polynomial of β. In many cases, the entries for C and C' are combined in a single row and this is indicated by 'a/b' where the parameters a and b are for C' and C, respectively. The notation '$[\theta_0]$' indicates that the primitive idempotent $\theta_0(x)$ is to be excluded from the defining idempotent of C'. In this Letter, some of the improvements coincide with the lower-bounds in [2]. For completeness, they are included in this Letter and are marked by '†'.

In the late 1970s, computing the minimum distance of extended QR codes was posed as a research problem [4]. Since then, the minimum distance of the extended QR code for prime 199 has been an open

question. For this code, the bounds of the minimum distance was $16-32$ [4] and the lower-bound was improved to 24 [8]. Since $199 \equiv -1 \pmod 8$, the extended code is doubly-even self-dual and its automorphism group contains a projective special linear group, which is doubly-transitive. As a result, the minimum distance of the [199, 100] QR code is odd, i.e. $d \equiv 3 \pmod 4$, and hence $d = 23, 27$ or 31. Owing to the cyclic property and the rate of this QR code [6], we can assume that a codeword of weight d has maximum information weight of $[d/2]$. If a weight d codeword does not satisfy this property, there must exist one of its cyclic shifts that does. After enumerating all codewords up to (and including) information weight 13 of the [199, 100] cyclic QR code using grid computers, no codeword of weight less than 31 was found, implying that d is indeed 31. Without exploiting the property that $d \equiv 3 \pmod 4$, additional $\binom{100}{14} + \binom{100}{15}$ codewords would need to be enumerated to establish the same result. Accordingly, we know that there exist [199, 99, 32] QR and [200, 100, 32] extended QR codes. By applying Construction Y1 [9] to the minimum weight codeword of the dual of the [200, 100, 32] extended QR code, a [168, 69, 32] new binary linear code is obtained.

In the case of $n = 205$, in addition to a $[205, 61, 46]^{\dagger}$ cyclic code, there also exists a [205, 61, 45] cyclic code which contains a $[205, 60, 48]^{\dagger}$ cyclic code as its even-weight subcode. Applying Construction X [9] to this pair of nested cyclic codes using a [3, 1, 3] auxiliary code, a $[208, 61, 48]^{\dagger}$ code, which is an improvement to Brouwer's lower-bound, is obtained.

It is interesting that many of the improvements in this Letter are contributed by low-rate cyclic codes of length 255 and there are 16 cases of this. Furthermore, it was found that the dual codes of the [255, 65, 63] cyclic code and that of its [255, 64, 64] even weight subcode both have minimum distance of 8. Applying Construction Y1 to the minimum weight codeword of the dual, we obtain [247, 58, 63] and [247, 57, 64] new binary linear codes. It is also interesting to see the existence of [255, 55, 70] and [255, 63, 65] cyclic codes, which are superior to the BCH codes of the same length and dimension. Both of these BCH codes have minimum distance 63 only.

To summarise, the four binary codes derived from cyclic codes in Table 1, which are improvements to Brouwer's lower-bound, are [168, 69, 32], $[208, 61, 48]^{\dagger}$, [247, 57, 64] and [247, 58, 63]. Given an $[n, k, d]$ code, where $d > d_{Brouwer}$, it is possible to obtain more improvements by recursively extending (annexing parity-check), puncturing and/or shortening the original code. For example, consider the [168, 69, 32] Construction Y1 code above; by annexing parity-check bit $[168+i, 69, 32]$, for $1 \le i \le 3$, new codes are obtained; by puncturing a [167, 69, 31] new code is obtained; by shortening $[168-i, 69-i, 32]$, for $1 \le i \le 5$, new codes are obtained. More improvements are also obtained by shortening the extended and punctured codes. Overall, there are 869 and 621 binary linear codes obtained which are improvements to the lower-bounds of Brouwer and those in [2], respectively.

Acknowledgments: This work is partly funded by an Overseas Research Students Award Scheme. The high throughput computing resources provided by the PlymGRID team of the University of Plymouth are gratefully acknowledged.

20 December 2006
Electronics Letters online no: 20073898
doi: 10.1049/el:20073898

C. Tjhai and M. Tomlinson (*Fixed and Mobile Communications Research, University of Plymouth, Plymouth, PL4 8AA, United Kingdom*)

E-mail: ctjhai@plymouth.ac.uk

References

1 Brouwer, A.E.: 'Bounds on the size of linear codes' *in* Pless V.S., and Huffman W.C. (eds): 'Handbook of Coding Theory' (Elsevier, North Holland, 1998), pp. 295–461
2 Grassl, M.: 'Code tables: bounds on the parameters of various types of codes', http://www.codetables.de, accessed 20 December 2006
3 Bosma, W., Cannon, J.J., and Playoust, C.P.: 'The Magma algebra system I: the user language', *J. Symb. Comput.*, 1997, **24**, pp. 235–266
4 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error-correcting codes' (North-Holland, 1977)
5 Van Lint, J.H.: 'Introduction to coding theory' (Springer-Verlag, 1999, 3rd edn.)
6 Chen, C.L.: 'Some results on algebraically structured error-correcting codes', PhD dissertation, University of Hawaii, USA, 1969
7 Chen, C.L.: 'Computer results on the minimum distance of some binary cyclic codes', *IEEE Trans. Inf. Theory*, 1970, **16**, (3), pp. 359–360
8 Grassl, M.: 'On the minimum distance of some quadratic residue codes'. Proc. IEEE Int. Symp. Information Theory, Sorento, Italy, June 2000, p. 253
9 Sloane, N.J., Reddy, S.M., and Chen, C.L.: 'New binary codes', *IEEE Trans. Inf. Theory*, 1972, **IT-18**, pp. 503–510

New linear codes derived from binary cyclic codes of length 151

C. Tjhai, M. Tomlinson, M. Grassl, R. Horan, M. Ahmed and M. Ambroze

Abstract: The minimum distance of all binary cyclic codes of length 151 is determined. Almost all of these cyclic codes have the same parameters as the best linear codes given in Brouwer's database. A nested chain of linear codes is derived from these cyclic codes and some new binary codes are obtained by applying Constructions X and XX to pairs of codes in this chain. Good candidates for nested codes can also be obtained by enlarging the cyclic codes of high minimum distance. In total, there are 39 new binary linear codes that have a minimum distance higher than codes previously considered to be the best linear codes.

1 Introduction

Let C be a binary linear code with parameters $[n, k, d]$ where n, k and d represent the length, dimension and minimum distance (d_{min}) of the code, respectively. The minimum distance of a code is an important parameter as it determines the error-correcting abilities of the code. Brouwer [1] has constructed a database containing the lower-bounds and upper-bounds of minimum distances for linear codes [Note 1]. The lower-bound corresponds to the largest minimum distance for a given n and k that has been found and verified to date. While there are codes in Brouwer's database whose minimum distance is equal to the upper-bound, particularly those codes with very low and very high code rates, many of them have a minimum distance that is considerably less than, the upper-bound. Constructing error-correcting codes which are improvements over those in Brouwer's database is an on-going research activity in coding theory.

A pair of cyclic codes is nested if all roots of one code are contained in the other one. Here the roots are those of the generator polynomial. By appropriate arrangement of their roots, it is possible to build a chain of nested cyclic codes of the same length [2]. In the other words, cyclic codes may be partitioned into a sequence of cyclic sub-codes $C_1 \supset C_2 \supset C_3 \ldots C_t$, such that a cyclic code C_i contains C_{i+1} and, in general, C_{i-1} has a higher minimum distance than C_i. A good example of a chain of cyclic codes is the primitive BCH codes, provided that the roots of the BCH codes, are chosen such that they start from a fixed position, i.e. a [31, 26, 3] BCH code contains a [31, 21, 5] BCH code, which then contains a [31, 16, 7] BCH code and so on.

It has been shown that the application of Construction X [3, 4] can produce many good linear codes, see [2, 5, 6]. In fact, many of the best known linear codes are obtained by applying Construction X [1]. This construction, which requires a pair of nested codes and an auxiliary code, pads a non-zero codeword of an auxiliary code to each codeword of the sub-code and a zero codeword to each codeword not in the sub-code. Another variation of this construction is known as Construction XX [7], which requires two pairs of nested codes and two auxiliary codes, and makes use of Construction X twice. In order to produce good new codes using these constructions, component codes which can be arranged into a nested form and have a high minimum distance are required. The cyclic codes of length 151 are special as all of the irreducible factors of $x^{151} - 1$, apart from $1 + x$, have a fixed weight equal to 15. Having a fixed weight implies that quadratic-residue and duadic codes can be constructed at length 151, and these families of codes are well known to contain powerful half-rate error-correcting codes. In terms of computing the minimum distance of cyclic codes, the length 151 is just within the current computational reach, and we find that many cyclic codes of length 151 have an exceptionally high minimum distance. Since cyclic codes can be easily arranged into a nested form, the cyclic codes of length 151 are clearly suitable candidates for component codes. In this paper, we present 39 new binary linear codes with minimum distances higher than the previous best codes contained in Brouwer's database. These new codes are constructed by applying Constructions X and XX to nested codes derived from cyclic codes of length 151. The results have been verified independently using the well known computer algebra package, MAGMA [8].

2 Cyclic codes of length 151

The techniques of polynomial factorisation over a finite field are well established. A cyclic code of length n is defined by its generator polynomial, denoted as $g(x)$, which can be obtained by taking the product of the irreducible polynomials that are factors of $x^n - 1$. For binary codes of length $n = 151$, we shall choose as our primitive 151st root, β, where $\beta = \alpha^{217}$ is a root of the polynomial $m(x) = 1 + x^4 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{13} + x^{14} + x^{15}$.

IEE Proceedings online no. 20050560

doi:10.1049/ip-com:20050560

Paper first received 9th October 2005 and in final revised form 10th March 2006

C. Tjhai, M. Tomlinson, R. Horan, M. Ahmed and M. Ambroze are with Fixed and Mobile Comunications Research, University of Plymouth, Plymouth PL4 8AA, UK

M. Grassl is with the Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 76128, Karlsruhe, Germany

E-mail: ctjhai@plymouth.ac.uk

Note 1: An updated version is available online at http://www.win.tue.nl/~aeb/voorlincod.html

Let a polynomial $c(x) = \sum_{i=0}^{n-1} c_i x^i$ be a codeword of a cyclic code of length n. It is well understood that the permutation $\sigma : i \rightarrow \mu i$, where μ is an integer prime to n, produces an equivalent cyclic code [4]. With respect to the permutation σ, it was found that there are 214 inequivalent cyclic codes of length 151.

There exist a number of efficient algorithms to evaluate the d_{min} of a cyclic code without having to determine the weight of each codeword. The first efficient method is due to Chen [9], which is also described in [10] and [6]. The algorithm is as follows. Let G be the systematic generator matrix of a cyclic code, u be a vector of length k over GF (2) and wt (u) be the weight, of vector u. Since G is systematic, a codeword c can be written in the form of $c = (u|p)$, where p is the parity vector of length $n-k$. For $w = 1, 2, \ldots, k$, enumerate all codewords that satisfy $\mathrm{wt}(u) = w$ by taking all possible sums of w rows in G. At each stage, the minimum weight of all codewords c enumerated so far yields an upper-bound on d_{min}, denoted as d_{ub}. The lower-bound, d_{lb}, is equal to $\lceil (w+1)n/k \rceil$, where $\lceil a \rceil$ is the smallest integer greater than or equal to a. The algorithm terminates when $d_{lb} \geq d_{ub}$, and then d_{ub} is the true minimum distance of the cyclic code.

In [11], Coppersmith and Seroussi presented a variation of the above method, in which the enumeration of codewords is done between some minimum distance estimates d_1 and d_2. In this case, they have to enumerate the codewords whose wt(u) is in the range of $\lfloor d_1 k/n \rfloor$ and $\lfloor d_2 k/n \rfloor$, where $\lfloor a \rfloor$ is the greatest integer less than or equal to a. The weight d_1 can be easily obtained (for example from the BCH-bound for cyclic codes or the square-root bound for duadic/quadratic-residue codes) and d_2 has to be chosen such that $d_1 \leq d_{min} \leq d_2$.

We implement an algorithm which uses the best features of the aforementioned methods and we use the revolving-door algorithm [12, 13] in enumerating combinations. The d_{min} of all cyclic codes of length 151 is then evaluated, and Table 1 tabulates the optimum ones. Here, the term optimum cyclic code refers to a cyclic code with the largest d_{min} for a given length and rate. In cases where there is more than one optimum code, only one is chosen. Table 1 shows that the lower-bound of d_{min}, given by the BCH-bound (d_{BCH}), is very poor and the true d_{min} of almost all of these cyclic codes is as large as that of the best known linear codes given in Brouwer's database ($d_{Brouwer}$).

Table 1: Optimum [151, k, d] cyclic codes

k	d	d_{BCH}	$d_{Brouwer}$
150	2	2	2
136	5	3	5
135	6	4	6
121	8	4	8
120	8	4	9
106	13	7	13
105	14	8	14
91	17	8	17
90	18	8	18
76	23	9	23
75	24	9	24
61	31	10	31
60	32	10	32
46	31	17	34
45	36	18	36
31	47	19	47
30	48	20	48
16	60	37	62
15	60	46	64
1	151	151	151

3 The new binary codes

Let $C_1 = [n, k_1, d_1]$ and $C_2 = [n, k_2, d_2]$ be a pair of nested codes, where $C_1 \subset C_2$. Given an auxiliary code, $C_3 = [n_3, k_3, d_3]$, where $k_3 = k_2 - k_1$, Construction X produces $C = [n + n_3, k_2, \min(d_1, d_2 + d_3)]$. The codeword of the resulting code can be partitioned in the form of $|c_2 \; |c_3|$, where c_i is a codeword of C_i. Since $C_1 \subset C_2$, we can partition C_2 into cosets of C_1, i.e. $C_2 = (b_0 + C_1) \cup (b_1 + C_1) \cup \ldots \cup (b_s + C_1)$, where $s = 2^{k_3} - 1$ and b_i is a vector of length n. Given c_2, c_3 is a zero codeword of C_3 if $\mathrm{wt}(b_i) = 0$ or a non-zero codeword of C_3 otherwise.

Construction XX, on the other hand, requires two sub-codes and two auxiliary codes. Let $C_1 = [n, k_1, d_1]$, $C_2 = [n, k_2, d_2]$ and $C_3 = [n, k_3, d_3]$ such that $C_2 \subset C_1$ and $C_3 \subset C_1$. Let $C_2 \cap C_3$ be denoted by C_4 and let the two auxiliary codes be $C_5 = [n', k_1 - k_2, d_5]$ and $C_6 = [n'', k_1 - k_3, d_6]$. Construction XX produces $C = [n + n' + n'', k_1, \min(d_4, d_2 + d_6, d_3 + d_5, d_1 + d_5 + d_6)]$. The resulting codeword can be partitioned into the form of $|c_1|x|y|$. The vector $|x|y|$ is $|c_5|0|$ if $c_1 \in c_2 \backslash c_4$, $|0|c_6|$ if $c_1 \in C_3 \backslash C_4$, or $|0|0|$ otherwise, where c_i denotes a non-zero codeword of C_i and the notation $c \in C_i \backslash C_j$ means that $C_j \subset C_i$ and c is a codeword of C_i, but not of C_j.

The techniques used to obtain the candidate codes for Constructions X and XX are presented in the following Sections.

3.1 Using a chain of nested codes

It is possible to build a chain of cyclic codes such that all roots of the higher-rate codes are contained in the lower-rate codes [2]. It is necessary to order the roots appropriately such that all cyclic codes in the chain have optimum d_{min}. For all cyclic codes in Table 1, whose generator, $g(x)$, is divisible by $1 + x$, an ordering of roots, excluding the conjugate roots, shown in Table 2, results in an optimum chain arrangement. Given $C_i \subset C_{i-1}$ $2 \leq i \leq 10$, we can take the non-cyclic sub-codes, such that

Table 2: Order of β in an optimum chain of [151, k_i, d_i] cyclic codes

i	k_i	d_i	Roots of $g(x)$, excluding conjugate roots									
1	150	2	β^0									
2	135	6	β^0	β^1								
3	120	8	β^0	β^1	β^3							
4	105	14	β^0	β^1	β^3	β^5						
5	90	18	β^0	β^1	β^3	β^5		β^{11}				
6	75	24	β^0	β^1	β^3	β^5		β^{11}	β^{15}			
7	60	32	β^0	β^1	β^3	β^5		β^{11}	β^{15}			β^{37}
8	45	36	β^0	β^1	β^3	β^5		β^{11}	β^{15}	β^{23}		β^{37}
9	30	48	β^0	β^1	β^3	β^5		β^{11}	β^{15}	β^{23}	β^{35}	β^{37}
10	15	60	β^0	β^1	β^3	β^5	β^7	β^{11}	β^{15}	β^{23}	β^{35}	β^{37}

Table 3: New binary codes from construction X; C_1 and C_2 are from the chain of nested codes

C_1	C_2	C_3	C
[151, 60, 32]	[151, 72, 24]	[23, 12, 7]	[174, 72, 31]
[151, 45, 36]	[151, 60, 32]	[20, 15, 3]	[171, 60, 35]

there exists [151, k, d], linear codes for $k_i + 1 \leq k \leq k_{i-1} - 1$ with, $d \geq d_{i-1}$. We then have a chain of nested [151, k, d] linear codes for $1 \leq k < 151$.

Each combination of pairs of codes in the [151, k, d] chain is a nested pair. In order to obtain a new code, each pair, C_1 and C_2, is combined using Construction X with an auxiliary code C_3 obtained from Brouwer's database, which is a code of length n_3 and dimension k_3 with the highest d_{min} currently known. The resulting code and the best code currently known, for the same code length and dimension are compared to see if a larger d_{min} code has been obtained. Some of the new codes obtained, which are improvements on the best codes currently known, as reported in Brouwer's database, are tabulated in Table 3.

3.2 Improved subcodes

In the case when the difference k_3 of the dimensions of the codes C_2 and C_1 is small, d_2, the minimum distance of C_2 obtained from a chain of nested codes, can be unsatisfactory. We can improve d_2 by augmenting C_1 with a vector v of length n. In finding a v that can maximise the d_{min} of the enlarged code, we have adopted, the following procedure.

Choose a code $C_1 = [n, k_1, d_1]$ that has a sufficiently high minimum distance. Let G denote its generator matrix in systematic form. We generate a vector v which satisfies the following conditions: $v_i = 0$ for $0 \leq i \leq k - 1$, where v_i is the ith element of v; $\mathrm{wt}(v) > d_2$ and $\mathrm{wt}(v \ominus G_r) > d_2$ for all $r \in \{0, 1, \ldots, k - 1\}$, where G_r is the rth row vector of G and $\oplus$ denotes the standard binary vector addition. The vector v is then appended to G as an additional row. Although the base code is cyclic, the enlarged code is not, so the d_{min} evaluation method for cyclic codes can no longer be used. We therefore include the improved Zimmermann algorithm [6], which is described below, into our procedure in order to evaluate the d_{min} of the enlarged code. This algorithm requires a set of reduced-echelon generator matrices. Assuming that there are m of these matrices $G^1, \ldots, G^m$, they may be produced as follows.

Table 4: New binary codes from construction X; C_2 is an augmented code of C_1, a cyclic code

C_1	C_2	C_3	C
[151, 76, 23]	[151, 77, 20]	[3, 1, 3]	[154, 77, 23]
[151, 61, 31]	[151, 62, 27]	[4, 1, 4]	[155, 62, 31]

Starting with G^1, we perform the Gauss–Jordan elimination so that $G^1 = [I_1 | A_1]$. It is worth mentioning that, during the Gauss–Jordan elimination process, column permutation is allowed. Clearly, the matrix I_1, has rank $r_1 = k$. Next, the sub matrix A_1 is transformed into the reduced echelon form

Table 6: New binary codes

n	k	d	$d_{Brouwer}$	Construction
169	58	35	34	shortening [171, 60, 35]
170	58	36	35	extending [169, 58, 35]
170	59	35	34	shortening [171, 60, 35]
171	59	36	35	extending [170, 59, 35]
171	60	35	34	construction X (Table 3)
172	60	36	35	extending [171, 60, 35]
153	62	29	28	puncturing [155, 62, 31]
154	62	30	28	Puncturing [155, 62, 31]
155	62	31	28	construction X (Table 4)
156	62	32	29	extending [155, 62, 31]
157	62	32	30	extending [155, 62, 31]
158	62	32	30	extending [155, 62, 31]
159	62	32	30	extending [155, 62, 31]
160	62	32	30	extending [155, 62, 31]
161	62	32	31	extending [155, 62, 31]
159	63	31	30	construction XX (Table 5)
160	63	32	30	extending [159, 63, 31]
161	63	32	30	extending [159, 63, 31]
162	63	32	31	extending [159, 63, 31]
168	66	31	30	shortening [174, 72, 31]
169	66	32	31	extending [168, 66, 31]
169	67	31	30	shortening [174, 72, 31]
170	67	32	31	extending [169, 67, 31]
170	68	31	30	shortening [174, 72, 31]
171	68	32	31	extending [170, 68, 31]
171	69	31	30	shortening [174, 72, 31]
172	69	32	31	extending [171, 69, 31]
172	70	31	30	shortening [174, 72, 31]
173	70	32	31	extending [172, 70, 31]
173	71	31	30	shortening [174, 72, 31]
174	71	32	30	extending [173, 71, 31]
175	71	32	31	extending [173, 71, 31]
174	72	31	30	construction X (Table 3)
175	72	32	30	extending [174, 72, 31]
176	72	32	31	extending [174, 72, 31]
154	77	23	22	construction X (Table 4)
155	77	24	22	extending [154, 77, 23]
156	77	24	22	extending [154, 77, 23]
157	77	24	23	extending [154, 77, 23]*

*This code also be obtained by shortening the [160, 80, 24] code constructed in [14]

Table 5: New binary codes from construction XX, starting from the cyclic code $C_2 \cap C_3$

C_1	C_2	C_3	$C_2 \cap C_3$	C_4	C_5	C
[151, 63, 23]	[151, 62, 27]	[151, 62, 27]	[151, 61, 31]	[4, 1, 4]	[4, 1, 4]	[159, 63, 31]

$A'_1 = [I_2|A_2]$, where I_2 has rank $r_2 \le r_1$, and we now have

$$G^2 = \left[\begin{array}{c|cc} & 0 & I_2 \quad A_2 \\ B & & \\ & I'_1 & 0 \quad 0 \end{array}\right]$$

where I'_1 is a $k - r_2$ dimensional identity matrix. The process then continues by transforming A_2 into a reduced echelon form $A'_2 = [I_3|A_3]$, which has rank $r_3 \le r_2$, and so on until all of the n coordinates have been exhausted. At the end of this process, we have a set of matrices $G^1, \ldots, G^m$ with the corresponding ranks $r_1, \ldots, r_m$, where $r_{i+1} \le r_i$ for $1 \le i < m$.

Let u be a vector of length k. Starting from $w = 1$, enumerate all codewords with $\mathrm{wt}(u) = w$ by taking the sum of all possible w rows of G^i for $1 \le i \le m$. For each G^i, we have an improvement of $\max(0, (w+1) - (k - r_i))$ to the lower-bound of d_{min}, denoted by d_{lb}. In addition to d_{lb}, we also compute the d_{min} upper-bound, d_{ub}, which is the minimum-weight codeword currently available. As in the case of cyclic codes, the revolving-door algorithm is used to enumerate all possible w rows of the generator matrices. This evaluation continues with $w' = w + 1$ and terminates whenever: $d_{ub} \le d_2$, in which case we choose a different v and restart the evaluation from $w = 1$, or $d_{ub} > d_2$ and $d_{ub} \le d_{lb}$, which means that we have an improvement of the minimum distance of C_2. Note that it may be possible to augment this code further while maintaining a minimum distance that is greater than d_2.

Note that all matrices G^i, for $1 \le i \le m$, that do not contribute to d_{lb} throughout the enumerations can be omitted for computational efficiency.

Using the above-mentioned approach, we found linear codes [151, 77, 20] and [151, 62, 27] which have higher minimum distances than the corresponding codes obtained from a chain of nested cyclic codes. These two codes are obtained starting from the cyclic codes [151, 76, 23] and [151, 61, 31], respectively. Therefore, [151, 76, 23] $\subset$ [151, 77, 20] and [151, 61, 31] $\subset$ [151, 62, 27]. Table 4 shows the parameters of the new codes derived from applying Construction X.

When searching for the code [151, 62, 27], we exploit the fact that the code [152, 61, 32], obtained by extending the cyclic code [151, 61, 31], is doubly even. We chose the additional vector v, such that extending the enlarged code [151, 62, d_2] yields again a doubly-even code. This implies the congruence $d_2 = 0, 3 \bmod 4$ for the minimum distance of the enlarged code. Hence it is sufficient to establish a lower-bound $d_{lb} = 25$, using the algorithm described above, to show that $d_2 \ge 27$. We also found two different codes $C_2 = [151, 62, 27]$ and $C_3 = [151, 62, 27]$, such that $C_1 = C_2 + C_3 = [151, 63, 23]$ and $C_2 \cap C_3 = [151, 61, 31]$. Using Construction XX, we get a code [159, 63, 31], see Table 5.

Table 6 lists all 39 new codes that are improvements on the codes given in Brouwer's database. Some of these codes are obtained by adding additional redundancies, shortening and puncturing the new codes given in Tables 3, 4 and 5. The current lower-bound of Brouwer is also included for comparison. The generator matrices of all new codes in Table 6 are available online at http://www.tech.plym.ac.uk/see/research/satcen/codes/. The minimum distances of C_2, in Table 4, C_1, C_2 and C_3, in Table 5, have been verified independently using MAGMA.

4 Performance evaluation

The maximum-likelihood (ML) performance of a given linear code depends on its spectral properties and minimum distance. While many iteratively decodable codes, such as turbo-codes and low-density parity-check (LDPC) codes, have good error performances, their minimum distance is poor and, as such, an early error floor is often observed in the performance of these codes. In this Section, we evaluate the soft-decision decoding performance of one of the many new codes. Although only one code is simulated, the error performance of all of the new codes should be closer to the optimal than that of the previous codes due to the improved minimum distance.

Figure 1 shows the frame error rate (FER) obtained from simulating the new rate 1/2 code of length 154. In the simulation, binary phase-shift key (BPSK) signalling is employed and a reliability-based soft-decision decoder [15] is used. The FER, performance is compared against the sphere-packing-bound (SPB) [16, 17] of the same code rate and block length. The classical Shannon limit, which is defined on the bit error rate (BER), is not used, because it is not an accurate measure due to the assumptions of error-free transmission and unconstrained block length. As

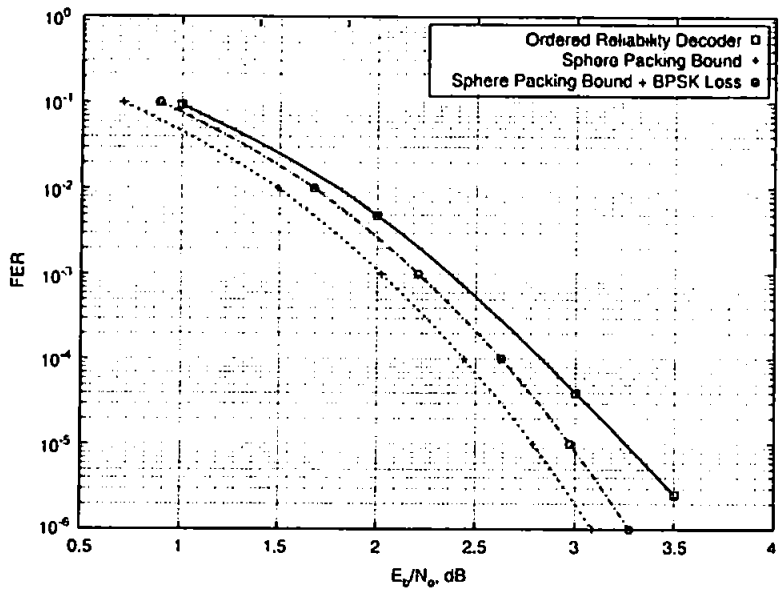


Fig. 1 *Performance of the [154, 77, 23] code using a reliability-based soft-decision decoder*

depicted in Fig. 1, the new [154, 77, 23] code has outstanding performance. Compared to the SPB, the performance at 10^{-4} FER is approximately 0.4 dB out and within 0.2 dB if loss due to BPSK transmission is taken into account. It is worth mentioning that almost all decoding errors observed during the simulation were due to more likely codewords. By definition, a more likely codeword is the codeword output by the decoder which has a closer Euclidean distance to the given received vector than the transmitted codeword and, as such, an ML decoder (if one exists) will also output the same codeword. This implies that the simulated FER performance of the [154, 77, 23] code is empirically close to the ML performance.

5 Acknowledgment

This work is partly funded by an Overseas Research Students (ORS) award. The authors would like to thank the anonymous reviewers for the constructive suggestions on the manuscript.

6 References

1 Brouwer, A.E.: 'Bounds on the size of linear codes', in Pless, V.S., and Huffman, W.C. (Eds.): 'Handbook of coding theory' (Elsevier, North Holland, 1998)
2 Grassl, M.: 'New binary codes from a chain of cyclic codes', *IEEE Trans. Inf. Theory*, 2001, 47, pp. 1178–1181
3 Sloane, N.J., Reddy, S.M., and Chen, C.L.: 'New binary codes', *IEEE Trans. Inf. Theory*, 1972, 18, pp. 503–510
4 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error-correcting codes' (North Holland, 1977), ISBN 0 444 85193 3
5 Bierbrauer, J., and Edel, Y.: 'Extending and lengthening BCH-codes', *Finite Fields Appl.*, 1997, 3, pp. 314–333
6 Grassl, M.: 'Searching for good linear codes', in Bosma, W., and Cannon, J. (Eds.): 'Discovering mathematics with MAGMA' (Springer, Berlin, 2006)
7 Alltop, W.O.: 'A method of extending binary linear codes', *IEEE Trans. Inf. Theory*, 1984, 30, pp. 871–872
8 Bosma, W., Cannon, J.J., and Playoust, C.: 'The Magma algebra system I: the user language', *J. Symb. Comput.*, 1997, 24, pp. 235–266
9 Chen, C.L.: 'Computer results on the minimum distance of some binary cyclic codes', *IEEE Trans. Inf. Theory*, 1970, 16, pp. 359–360
10 Promhouse, G., and Tavares, S.E.: 'The minimum distance of all binary cyclic codes of odd lengths form 69 to 99', *IEEE Trans. Inf. Theory*, 1978, 24, pp. 438–442
11 Coppersmith, D., and Seroussi, G.: 'On the minimum distance of some quadratic residue codes', *IEEE Trans. Inf. Theory*, 1984, 30, pp. 407–411
12 Bitner, J.R., Ehrlich, G., and Reinglod, E.M.: 'Efficient generation of the binary reflected Gray code and its applications', *Commun. ACM*, 1976, 19, pp. 517–521
13 Knuth, D.E.: 'The art of computer programming, vol. 4, fascicle 3: Generating all combinations and partitions (Addison-Wesley Professional, 2005)
14 Van Dijk, M., Egner, S., Greferath, M., and Wassermann, A.: 'On two doubly even self-dual binary codes of length 160 and minumum weight 24', *IEEE Trans. Inf. Theory*, 2005, 51, pp. 408–411
15 Fossorier, M.: 'Reliability-based soft-decision decoding for linear block codes', in Lin, S., and Costello, D.J. (Eds.): 'Error control coding' (Pearson Education, Inc. 2004, 2nd edn.)
16 Shannon, C.E.: 'Probability of error for optimal codes in a gaussian channel', *Bell Syst. Tech. J.*, 1959, 38, pp. 611–656
17 Dolinar, S., Divsalar, D., and Pollara, F.: 'Code performance as a function of block size', JPL TMO progress report 42-133, May 1998, http://tmo.jpl.nasa.gov/progress_report/42-133/133K.pdf, accessed 14 July 2006

Idempotents, Mattson-Solomon polynomials and binary LDPC codes

R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze and M. Ahmed

Abstract: It is shown how to construct an algorithm to search for binary idempotents that may be used to construct binary LDPC codes. The algorithm, which allows control of the key properties of sparseness, code rate and minimum distance, is constructed in the Mattson-Solomon domain. Examples are given of the codes constructed that include equivalent codes to the Euclidean and Projective Geometry codes in addition to some new codes. Codes having cycles of length 4 can also be constructed and are demonstrated to have good performance under iterative decoding.

1 Introduction and background

The use of idempotents in the construction of cyclic error correcting codes is well established and the resulting literature is extensive (for example, see [1, 2 or 3]). The basic building blocks for this theory are the primitive idempotents. Any cyclic code may be described by a unique idempotent and this idempotent is a sum of primitive idempotents. For binary cyclic codes, efficient algorithms exist for the calculation of these primitive idempotents.

Another way of constructing idempotents in the binary case is by using cyclotomic cosets and this property was exploited, in a recent article [4], by four of the current authors. This was also the approach adopted by Shibuya and Sakaniwa in [5], but their aim was to use idempotents to construct parity check matrices for LDPC codes that have no cycles of length 4 in their factor graphs. At the heart of their technique is a lemma that is a variation of a result used by Weldon [6], for the construction of difference set cyclic codes. Using this lemma and a subsequent theorem, they were able to simplify the problem of determining which idempotent that is constructed, using a single cyclotomic coset, does not have cycles of length 4. They then extended this theory to more than one cyclotomic coset.

This approach to the construction of LDPC codes has the great advantage of simplicity, the parity check matrices depend only upon the correct choice of cyclotomic cosets and these are very easily calculated. However, the minimum distance and the code rate were not controlled in this construction method. Also, whilst the absence of four cycles is a desirable objective in the construction of LDPC codes it is not mandatory, since there are some good codes that do not have this property, e.g. see [7] and [8]. An example of such a code is included in this paper. The code rate is an important property of codes but, as Shibuya and Sakaniwa admit in their conclusion, the codes that they construct are 'expected to have a large minimum distance at the expense of rate'. The minimum distance of a code is a crucial property but there is no indication in [5] of how either a single cyclotomic coset, or combinations of more than one cyclotomic coset, should be chosen to guarantee that the code constructed has a large minimum distance.

In order to address the question of how to choose idempotents that will produce good LDPC codes we propose an entirely different route to that of [4] or [5]. As in those articles, we shall deal exclusively with binary cyclic codes. Making effective use of the Mattson-Solomon polynomial, we produce an algorithm that not only allows us to choose, in a systematic way, idempotents with low weight, and therefore a correspondingly sparse parity check matrix, but also with the desirable features that the corresponding codes have a high code rate and a large minimum distance.

2 Binary idempotents

Let $F = GF(2)$, n be an odd positive integer and $\mathcal{F}$ be the splitting field for $x^n - 1$ over F. Let $\alpha \in \mathcal{F}$ be a primitive nth root of unity and let $T(x)$ be the polynomials in $\mathcal{F}[x]$ of degree $\leq n - 1$. If $a(x) \in T(x)$ then the map $\Phi: T \rightarrow T$ is defined by

$$[\Phi(a)](z) = \sum_{j=1}^{n} a(\alpha^j) z^{n-j} \qquad (1)$$

and $\Phi(a)$ is the Mattson-Solomon polynomial of a (see [1]). We use x and z for the polynomial variables to distinguish between the polynomials in the domain and codomain of Φ. If $\circ$ is multiplication of polynomials mod $(x^n - 1)$ and $*$ is defined on $T(z)$ by the rule $(\sum a_i z^i) * (\sum b_i z^i) = (\sum a_i b_i z^i)$ then it is well known [1, 3], that

$$\Phi : (T, +, \circ) \rightarrow (T, +, *)$$

is an isomorphism of rings, in particular it is an isomorphism of the additive groups.

By the term idempotent, we shall mean a polynomial $e(x) \in (T, +, \circ)$ such that $e(x) \circ e(x) = e(x)^2 = e(x)$. If $S(x)$ is the subset of $T(x)$ consisting of polynomials with coefficients in $GF(2)$ (binary polynomials) and $E(x)$ is the subset of $T(x)$ consisting of idempotents, both of these

IEE Proceedings online no. 20050415

doi:10.1049/ip-com:20050415

Paper first received 1st February and in final revised form 24th August 2005

The authors are with Fixed and Mobile Communications Research, University of Plymouth, Plymouth, PL4 8AA, UK

E-mail: ctjhai@plymouth.ac.uk

subsets are additive subgroups of $T(x)$. It is easy to show (see [1]) that

$$\Phi : (S(x), +) \rightarrow (E(z), +) \tag{2}$$

$$\Phi : (E(x), +) \rightarrow (S(z), +) \tag{3}$$

are both isomorphisms and from this it is obvious that

$$\Phi : (S(x) \cap E(x), +) \rightarrow (E(z) \cap S(z), +) \tag{4}$$

is also an isomorphism.

Suppose that $u(x)$ is a binary idempotent used to construct a parity check matrix for a cyclic code. The parity check matrix is constructed from the n-cyclic shifts of $u(x)$ [9], and so for the resulting code to be a LDPC code, $u(x)$ must have low weight.

If $h(x) = \gcd(x^n - 1, u(x))$ and $g(x) = (x^n - 1)/h(x)$, then $g(x)$ is the generator of the cyclic code. If the generator, $g(x)$, has degree $n - k$, the dimension of the code is k and the larger the value of k, the better the code rate. Since $g(x)$ is a divisor of $x^n - 1$, all of the zeros of $g(x)$ are nth roots of unity, and there are $n - k$ of these. Furthermore, $\gcd(g(x), h(x)) = 1$ and $x^n - 1 = h(x)g(x)$, so that the number of distinct nth roots of unity, which are also roots of $u(x)$, is k. The dimension of the code is therefore the number of nth roots of unity, which are also roots of $u(x)$.

The BCH bound of the code is determined by the number of consecutive powers of α, taken cyclically (mod n), which are roots of $g(x)$. For the reasons outlined in the previous paragraph, this is precisely the same as the number of consecutive powers of α, taken cyclically (mod n), which are not roots of $u(x)$.

The important features of the code are therefore determined by:

(a) the weight of the idempotent $u(x)$,

(b) the number of nth roots of unity that are roots of $u(x)$,

(c) the number of consecutive powers of α which are not roots of $u(x)$.

Take $u(x) \in S(x) \cap E(x)$ and let $\Phi(u) = \theta$ be its MS polynomial. The inverse mapping

$$\Phi^{-1} : (S(z) \cap E(z), +) \rightarrow (E(z) \cap S(x), +) \tag{5}$$

is defined as follows: If $A(z) = [\Phi(a)](z)$ is the Mattson-Solomon polynomial of the polynomial $a(x) = a_0 + a_1 x + \cdots + a_n x^{n-1}$ then, for $i = 0, \ldots, n-1$,

$$a_i = \frac{1}{n} A(\alpha^i) \tag{6}$$

(see [1]). Let $h(z) = \gcd(\theta(z), z^n - 1)$ and let $f(z) = (z^n - 1)/h(z)$. The three key properties relating to the idempotent $u(x)$, listed above, are easily gleaned from its Mattson-Solomon polynomial $\theta(z)$, and $f(z)$, as follows.

2.1 The weight of u(x)

The weight of $u(x)$, denoted as $\mathrm{wt}(u(x))$, is the number of nth roots of unity that are zeros of $f(z)$. To see this note that $f(\alpha^i) = 0$ if and only if $\theta(\alpha^i) = 1$, since idempotents take only the values 0 and 1 in $\mathcal{F}$. Now $u = \Phi^{-1}\theta$ and the coefficients of $u(x) = u_0 + u_1 x + \cdots + u_{n-1}x^{n-1}$ are given by

$$u_i = \theta(\alpha^i) \bmod 2 \quad \text{for } i = 0, \ldots, n-1 \tag{7}$$

(cf. (6)). Thus $u_i = 1$ precisely when $f(\alpha^i) = 0$, giving $\mathrm{wt}(u(x))$ as the degree of the polynomial $f(z)$.

2.2 The zeros of u(x)

From the definition of the MS polynomial, (1),

$$\theta(z) = \sum_{j=1}^{n} u(\alpha^j) z^{n-j} \tag{8}$$

and the number of zeros of $u(x)$, which are roots of unity, is clearly $n - \mathrm{wt}(\theta(z))$.

2.3 The BCH bound of the code

The BCH bound of the code is the largest number of consecutive powers of α which are not roots of $u(x)$, i.e. the number of consecutive i, taken cyclically (mod n), such that $u(\alpha^i) = 1$. From (8), this is the largest number of consecutive non-zero coefficients in θ, taken cyclically (mod n).

Using this information, a systematic search for idempotents can now be made in increasing order of weight (sparseness), with accompanying knowledge of the number of roots that are nth roots of unity and the corresponding BCH bound. This algorithm is constructed in the Mattson-Solomon domain.

Let the decomposition of $z^n - 1$ into irreducible (over $F = GF(2)$) polynomials be $z^n - 1 = f_1(z) f_2(z) \ldots f_t(z)$. For $i = 1, \ldots, t$, let $k_i(z) = (z^n - 1)/f_i(z)$ and let $\theta_i(z)$ be the associated primitive idempotent (see [1] or [3]). These are displayed below in an array, together with other idempotents

$$\left.\begin{array}{ccc} u_1(x) & \theta_1(z) & f_1(z) \\ u_2(x) & \theta_2(z) & f_2(z) \\ \vdots & \vdots & \vdots \\ u_t(x) & \theta_t(z) & f_t(z) \end{array}\right\} \tag{9}$$

Here $u_1(x), u_2(x), \ldots, u_t(x)$ are the idempotents whose Mattson-Solomon polynomials are $\theta_1(z), \theta_2(z), \ldots, \theta_t(z)$, respectively. Let $I \subseteq \{1, 2, \ldots, t\}$ and let u, θ and f be defined as $u = \sum_{i \in I} u_i$, $\theta = \sum_{i \in I} \theta_i$ and $f(z) = \prod_{i \in I} f_i(z)$. From the properties of primitive idempotents, if $h(z) = \gcd(\theta(z), z^n - 1)$ then it follows that $\gcd(f(z), h(z)) = 1$ and $z^n - 1 = f(z)h(z)$. The idempotent u will now have the following properties

$$\mathrm{wt}(u) = \sum_{i \in I} \deg(f_i) \tag{10}$$

$$\text{number of zeros of } u = n - \mathrm{wt}(\theta) \tag{11}$$

The BCH bound is determined from $\theta(z)$ as explained in Section 2.3.

Since methods for finding the θ_i and f_i are well documented (see e.g. [10]) a search algorithm can be built around this observation to find a suitable weight idempotent with a known number of zeros and a known BCH bound. The rows of the array (9), are ordered by the degree of the polynomials, i.e. $\deg(f_i) \leq \deg(f_{i+1})$ for all i, and a search can be made in increasing order of weight. When a successful outcome has been obtained, only at this stage is it necessary to evaluate the inverse Fourier transform to find the corresponding idempotent. All of the information required will already be known.

3 Design and implementation

If t denotes the number of binary irreducible polynomials of $z^n - 1$, the complexity of an exhaustive search algorithm is $\mathcal{O}(2^t)$. We reduce this search complexity by targeting the search on the three key parameters: 1) sparseness of the parity-check matrix, 2) code rate and 3) minimum distance.

3.1 Sparseness of the parity-check matrix

Difference-set cyclic codes were introduced by Weldon [6]. These codes have the desirable property that the parity check equations are orthogonal on all bits and have no cycles of length 4 in their factor graphs. A necessary but not sufficient condition for this is that if $v(x)$ is the polynomial that generates the parity check matrix then $\mathrm{wt}(v(x))$ must satisfy the inequality

$$\mathrm{wt}(v(x))(\mathrm{wt}(v(x)) - 1) \leq n \tag{12}$$

where n is the code length. Since the weights of the idempotents $u(x)$ are related to the degrees of the f_i by (10), the inequality of (12) becomes

$$\sum_{\forall i \in I} \deg(f_i) \leq \sqrt{n} \tag{13}$$

We define the parameter δ that is used either to restrict codes to be orthogonal on all bits ($\delta = 0$) or to introduce cycles of length 4 in their factor graph ($\delta > 0$). In this case

$$\sum_{\forall i \in I} \deg(f_i) \leq \sqrt{n} + \delta \tag{14}$$

While all codes that have no cycles of length 4 satisfy condition (12), there are some codes that satisfy this but do have cycles of length 4. These latter codes can be excluded by evaluating the difference enumerator polynomial given by [4, 6]

$$\mathcal{D}(u(x)) = u(x)u(x^{-1}) \tag{15}$$

Note that $\mathcal{D}(u(x))$ is evaluated with real coefficients.

3.2 Code rate

The code rate is directly proportional to the number of roots of $u(x)$. If we let R_{min} represent a minimum desired code rate then, following (11), we can refine the search

Table 1: Examples of the constructed codes having no cycles of length 4

$[n, k]$	$u(x)$	d_{min}
[21, 11]†	$1+x^3+x^4+x^9+x^{11}$	6
[63, 37]‡	$1+x+x^3+x^7+x^{15}+x^{20}+x^{31}+x^{41}$	9
[73, 45]†	$1+x+x^3+x^7+x^{15}+x^{31}+x^{36}+x^{54}+x^{63}$	10
[93, 47]	$1+x^2+x^8+x^{31}+x^{32}+x^{35}+x^{47}$	8
[105, 53]	$1+x^4+x^{30}+x^{32}+x^{45}+x^{46}+x^{53}$	8
[217, 109]	$1+x^7+x^{21}+x^{24}+x^{49}+x^{55}+x^{105}+x^{117}$	9
[255, 175]‡	$1+x+x^3+x^7+x^{15}+x^{20}+x^{31}+x^{53}+x^{63}+x^{98}+x^{107}+x^{127}+x^{140}+x^{170}+x^{197}+x^{215}$	17
[273, 191]†	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{90}+x^{116}+x^{127}+x^{136}+x^{181}+x^{194}+x^{204}+x^{233}+x^{238}+x^{255}$	18
[341, 205]	$1+x^{29}+x^{87}+x^{92}+x^{94}+x^{114}+x^{122}+x^{150}+x^{202}+x^{203}+x^{213}+x^{217}+x^{234}+x^{257}+x^{273}$	16
[465, 233]	$1+x^{15}+x^{16}+x^{45}+x^{47}+x^{105}+x^{109}+x^{225}+x^{233}$	10
[511, 199]	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{82}+x^{100}+x^{127}+x^{152}+x^{165}+x^{201}+x^{255}+x^{296}+x^{305}+x^{331}+x^{403}$	19
[511, 259]	$1+x^{31}+x^{42}+x^{83}+x^{115}+x^{217}+x^{240}+x^{261}+x^{360}+x^{420}+x^{450}+x^{465}$	13
[819, 435]	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{127}+x^{204}+x^{255}+x^{409}+x^{511}$	13
[819, 447]	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{127}+x^{204}+x^{255}+x^{350}+x^{409}+x^{511}+x^{584}+x^{701}$	16
[1023, 781]‡	0, 1, 3, 7, 15, 31, 52, 63, 105, 122, 127, 211, 245, 255, 268, 322, 340, 398, 423, 491, 511, 537, 572, 645, 672, 681, 710, 797, 847, 866, 944, 983	33
[1057, 813]†	0, 1, 3, 7, 15, 31, 54, 63, 109, 127, 138, 219, 255, 277, 298, 338, 348, 439, 452, 511, 528, 555, 597, 677, 697, 702, 754, 792, 879, 905, 924, 990, 1023	34
[1387, 783]	0, 1, 3, 7, 15, 31, 63, 127, 246, 255, 346, 493, 511, 550, 588, 660, 693, 816, 866, 968, 987, 1023, 1101, 1126, 1177, 1256, 1321	28
[1533, 823]	0, 3, 9, 21, 45, 93, 189, 246, 300, 381, 456, 495, 508, 603, 765, 888, 915, 993, 1019, 1209	21
[1971, 1105]	0, 1, 3, 7, 15, 31, 63, 76, 127, 153, 255, 307, 492, 511, 615, 656, 985, 1023, 1231, 1313	21
[2047, 1167]	0, 1, 3, 7, 15, 26, 31, 53, 63, 107, 127, 215, 255, 431, 511, 770, 863, 1023, 1036, 1408, 1541, 1727	23
[2255, 1191]	0, 1, 3, 7, 15, 31, 63, 127, 140, 255, 281, 511, 563, 598, 1023, 1127, 1197, 1426, 1840, 2047	21
[2325, 1373]	0, 1, 3, 7, 15, 31, 63, 108, 127, 217, 255, 435, 511, 524, 871, 1023, 1049, 1084, 1162, 1216, 1424, 1704, 1743, 1770, 1874, 2014, 2047, 2099, 2169	30
[3741, 2229]	0, 1, 3, 7, 15, 31, 63, 127, 136, 255, 273, 354, 511, 547, 642, 709, 1023, 1095, 1285, 1402, 1419, 1870, 1938, 2047, 2191, 2571, 2805, 2839	29
[4095, 3367]‡	0, 1, 3, 7, 15, 31, 40, 63, 81, 127, 163, 234, 255, 272, 327, 410, 469, 511, 516, 545, 655, 732, 821, 866, 939, 1023, 1033, 1091, 1152, 1206, 1311, 1414, 1465, 1564, 1586, 1643, 1733, 1768, 1866, 1879, 2047, 2067, 2164, 2183, 2252, 2305, 2413, 2480, 2623, 2650, 2754, 2829, 2840, 2931, 2980, 3129, 3173, 3287, 3372, 3424, 3467, 3537, 3733, 3759	65
[4161, 3431]†	0, 1, 3, 7, 15, 31, 63, 127, 255, 284, 306, 356, 398, 424, 511, 569, 613, 713, 750, 797, 849, 1023, 1116, 1139, 1227, 1386, 1427, 1501, 1550, 1595, 1604, 1699, 1760, 1846, 2042, 2047, 2080, 2222, 2233, 2258, 2279, 2292, 2455, 2638, 2773, 2855, 2882, 2960, 3003, 3101, 3120, 3191, 3209, 3226, 3399, 3521, 3560, 3640, 3693, 3860, 3900, 4010, 4030, 4085, 4095	66
[5461, 3781]	0, 1, 3, 7, 15, 31, 63, 76, 127, 153, 255, 307, 511, 578, 615, 754, 776, 1023, 1157, 1196, 1231, 1509, 1553, 2047, 2144, 2315, 2393, 2463, 2730, 2768, 3019, 3107, 3118, 3328, 3802, 4095, 4114, 4289, 4394, 4631, 4787, 4927	43

†Equivalent to the Type-I 2-D projective geometry code [12]
‡Equivalent to the Type-I 2-D Euclidean geometry code [12]

bound to

$$\mathrm{wt}(\theta) \leq (1 - R_{min})n \tag{16}$$

3.3 Minimum distance

Let d be the lowest desired minimum distance and let r_θ be the largest number of consecutive non-zero coefficients, taken cyclically (mod n), of θ. Then, following the discussion in Section 2.3, we restrict the search algorithm to those θ for which the BCH bound applies

$$r_\theta \geq d - 1 \tag{17}$$

When a cyclic code has no cycles of length 4, i.e. $u(x)$ is orthogonal on each symbol position of the codeword, then the minimum distance is simply $1 + \mathrm{wt}(u(x))$ [11, Theorem 10.1].

We develop an efficient, but exhaustive, recursive tree search based on the above bounds. The flowchart of this search algorithm is shown in Fig. 5 in Appendix 8.1.

4 Code example and performance

Since the algorithm is an exhaustive search, the code construction method presented in this paper is able to produce, in addition to new codes, equivalent codes to the Euclidean and projective geometry LDPC codes [12]. Some of the codes found using this technique, which have no cycles of length 4 in their factor graph ($\delta = 0$), are presented in Table 1. The girth of these cyclic codes is at least 6. Owing to space limitation, for codes with n larger than 1000, only the exponents of the idempotent $u(x)$ are given. In addition, some good performance cyclic codes that do have cycles of length 4 ($\delta > 0$) are listed in Table 2. Here, the column N_c represents the number of cycles of length 4 per variable node in their factor graph. As the codes are cyclic, N_c is the same for each variable node. Note that d_{min} in these tables refers to the minimum distance of the code.

Table 2: Examples of good performance cyclic codes that have cycles of length 4

$[n, k]$	$u(x)$	d_{min}	N_c
[51, 26]	$1+x^3+x^6+x^{12}+x^{17}+x^{24}+x^{27}+x^{34}+x^{39}+x^{45}+x^{48}$	10	150
[63, 44]	$1+x^9+x^{11}+x^{18}+x^{21}+x^{22}+x^{25}+x^{27}+x^{30}+x^{37}+x^{42}+x^{44}+x^{45}+x^{50}+x^{54}$	8	294
[117, 72]	$1+x+x^2+x^4+x^8+x^{11}+x^{16}+x^{22}+x^{32}+x^{44}+x^{59}+x^{64}+x^{88}$	12	72
[127, 84]	$1+x+x^2+x^4+x^8+x^{16}+x^{32}+x^{55}+x^{59}+x^{64}+x^{91}+x^{93}+x^{109}+x^{110}+x^{118}$	10	126

Throughout the paper, it is assumed that the codewords are transmitted across a noisy communication channel with binary phase-shift keying (BPSK) modulation and the receiver uses a modified belief propagation decoder that approximates the maximum-likelihood decoder [13]. The algorithm of this decoder is outlined in Appendix 8.2. In using the modified belief propagation decoder, the following assumptions are used: maximum number of iterations is 100, i_{max} is set to n and π reorders the coordinates of each received vector in an increasing order of reliability (see Appendix 8.2).

Figure 1 shows the frame-error-rate (FER) performance realised for the [127, 84, 10] cyclic code that has cycles of length 4. Despite having cycles of length 4, the performance of this code is outstanding and, at 10^{-3} FER, it is within 0.2 dB of the sphere-packing-bound constrained for binary transmission.

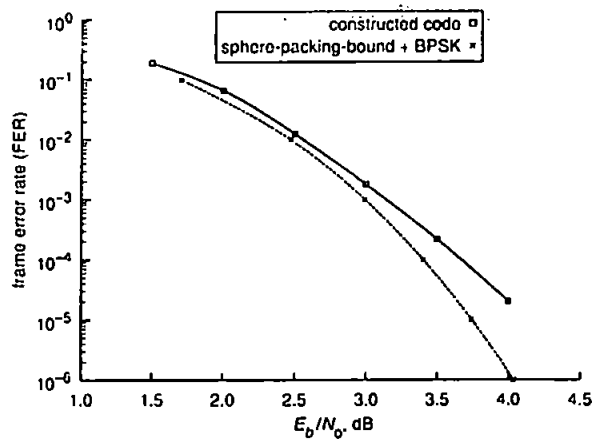


Fig. 1 *Frame error performance of the [127, 84, 10] cyclic code*

Figure 2 shows the performance of two [255, 175] codes, one constructed using our method (equivalent to the Euclidean geometry code) and the other one is an irregular LDPC code constructed using the progressive-edge-growth (PEG) algorithm [14]. We can see that the cyclic code, which achieves a coding gain of around 0.4 dB compared to the equivalent irregular code, performs approximately within 0.15 dB of the sphere-packing-bound constrained for binary transmission at 10^{-3} FER.

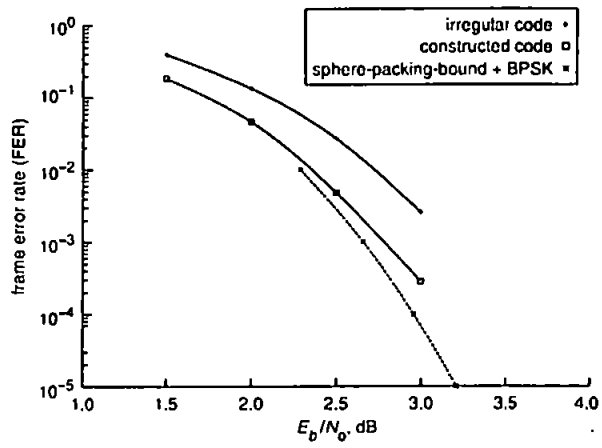


Fig. 2 *Frame error performance of the [255, 175] codes*

The construction method described in this paper can produce LDPC codes with high minimum distance and therefore they do not suffer from an error-floor. Figure 3 demonstrates the performance of the [341, 205, 16] cyclic code, which is inferior to the equivalent irregular PEG code in the low signal-to-noise ratio (SNR) region, but the irregular code exhibits an early error floor owing to its low minimum distance, which is 6.

Some of the codes listed in Table 1 have smaller minimum distances than the other codes having the same code length n. Under iterative decoding, the codes with lower minimum distance can exhibit better performance owing to improved convergence. As an example, consider the [819, 435, 13] and [819, 447, 16] cyclic codes whose performances are depicted in Fig. 4. Although their code rates are comparable, there is a significant difference in their iterative decoding performance where the lower minimum distance code has an advantage at low E_b/N_0 values. It has been observed that the error-floor exhibited by the [819, 435, 13] cyclic code is attributable to minimum distance

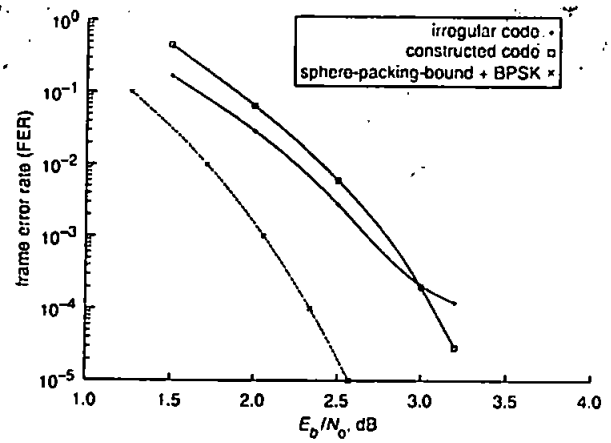


Fig. 3 *Frame error performance of the [341, 205] codes*

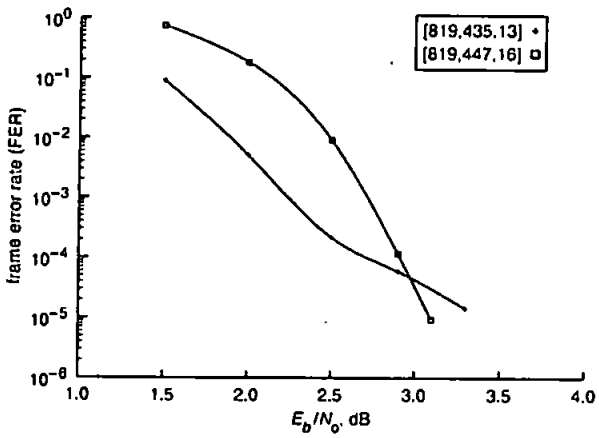


Fig. 4 *Frame error performance of the [819, 435, 13] and [819, 447, 16] cyclic codes*

error events, which implies that a maximum-likelihood decoder will also suffer from the same error floor.

5 Conclusions

A method of constructing binary cyclic codes from the finite-field transform (Mattson-Solomon) domain is able to produce a large number of codes that have high minimum distance and code rate. Codes equivalent to all of the Euclidean and projective geometry LDPC codes [12] can also be generated using this method. The constructed codes have sparse parity check matrices and thus are applicable as LDPC codes. Owing to their cyclic property, these LDPC codes have n parity check equations instead of $n-k$ equations as in the case of random LDPC codes. With these extra parity check equations to iterate with, the performance of the iterative decoder is improved.

In designing cyclic LDPC codes of length n, the described method allows one to increase the minimum distance of the code by combining additional irreducible factors of $z^n - 1$ which, in turn, reduces the sparseness of the parity check matrix. The ability to control the sparseness of the parity check matrix is a trade-off against the minimum distance of the code. In some cases, the constructed codes exhibit cycles of length 4 and still have good performance using iterative decoding.

Simulation results have shown that the relatively short cyclic codes have outstanding performance and are superior to the equivalent irregular LDPC codes constructed using the PEG algorithm. The high minimum distance of these cyclic codes ensures the absence of an early error floor in their performance. We have also shown that there are exceptions to the myth that all codes that have cycles of length 4 in their factor graph are bad under iterative decoding.

To realise the best performance of all of these cyclic codes, a modified belief propagation decoder has been used (see Appendix 8.2). This was used for all of the cyclic codes constructed including those codes having no cycles of length 4.

6 Acknowledgments

This work is partly funded by the Overseas Research Students Award Scheme. The authors wish to thank the anonymous reviewers for their constructive suggestions and comments.

7 References

1 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error correcting codes' (North Holland, New York, 1977)
2 Roman, S.: 'Coding and information theory' (Springer Verlag, New York, 1992)
3 van Lint, J.H.: 'Introduction to coding theory' (Springer-Verlag, Berlin, Germany, 1991, 2nd edn.)
4 Tjhai, C., Tomlinson, M., Ambroze, M., and Ahmed, M.: 'Cyclotomic idempotent-based binary cyclic codes', *Electron. Lett.*, 2005, 41, (6), pp. 341–343
5 Shibuya, T., and Sakaniwa, K.: 'Construction of cyclic codes suitable for iterative decoding via generating idempotents', *IEICE Trans. Fundam.*, 2003, E86-A, (4), pp. 928–939
6 Weldon, E.J.: 'Difference-set cyclic codes', *Bell Syst. Tech. J.*, 1966, 45, pp. 1045–1055
7 Tang, H., Xu, J., Lin, S., and Abdel-Ghaffar, K.A.S.: 'Codes on finite geometries', *IEEE Trans. Inf. Theory*, 2005, 51, (2), pp. 572–596
8 Chen, L., Xu, J., Djurdjevic, I., and Lin, S.: 'Near-shannon-limit quasi-cyclic low-density parity-check codes', *IEEE Trans. Commun.*, 2004, 52, pp. 1038–1042
9 Pless, V.S., Huffman, W.C., and Brualdi, R.A.: 'An introduction to algebraic codes' in Pless, V.S. and Huffman, W.C. (Eds.): 'Handbook of coding theory' (Elsevier, North Holland, Amsterdam, The Netherlands, 1998)
10 van Lint, J.H.: 'Coding theory', *Lect. Notes Math.*, 1971, 201, Springer-Verlag, Berlin, Germany
11 Peterson, W., and Weldon Jr., E.: 'Error-correcting codes' (MIT Press, Cambridge, 1972, 2nd edn.)
12 Kou, Y., Lin, S., and Fossorier, M.: 'Low-density parity-check codes based on finite geometries: a rediscovery and new results', *IEEE Trans. Inf. Theory*, 2001, 47, pp. 2711–2736
13 Tjhai, C., Papagiannis, E., Tomlinson, M., Ambroze, M., and Ahmed, M.: 'Improved iterative decoder for LDPC codes with performance approximating to a maximum-likelihood decoder', UK Patent Application 0409306.8
14 Hu, X.Y., Eleftheriou, E., and Arnold, D.M.: 'Progressive edge-growth tanner graphs', IEEE Global Telecommunications Conf. (GLOBECOM), San Antonio, TX, November 2001
15 Papagiannis, E., Ambroze, M., and Tomlinson, M.: 'Analysis of non convergent blocks at low and moderate SNR in SCCC turbo schemes', Proc. 8th Int. Workshop on Signal Processing for Space Communications, Catania, Italy, 24–26 September 2003, pp. 121–128

8 Appendix

8.1 Code search algorithm

Figure 5 illustrates the flowchart of the efficient, but exhaustive, recursive code search algorithm developed. The inputs to the algorithm are: i) $R_{\min}$ minimum code rate of interest, ii) d lowest expected minimum distance, iii) δ a non-negative integer and iv) $\mathbf{F}(z) = \{f_i(z)\}\ \forall i \in I$ sorted in ascending order of the degree and the corresponding set of primitive idempotent, $\mathbf{Q}(z) = \{\theta_i(z)\}\ \forall i \in I$. As in Section 2, $I \subseteq \{1, 2, \ldots, t\}$. This algorithm outputs two lists of cyclic LDPC codes which correspond to the input parameters, **CodeList4** and **CodeListNo4**. Codes that have cycles of length 4 are in the former, otherwise are in the latter. Prior to execution of this algorithm, **V** and *index* are initialised to $\emptyset$ and -1 respectively. The notation used in the flowchart is identical to that used earlier.

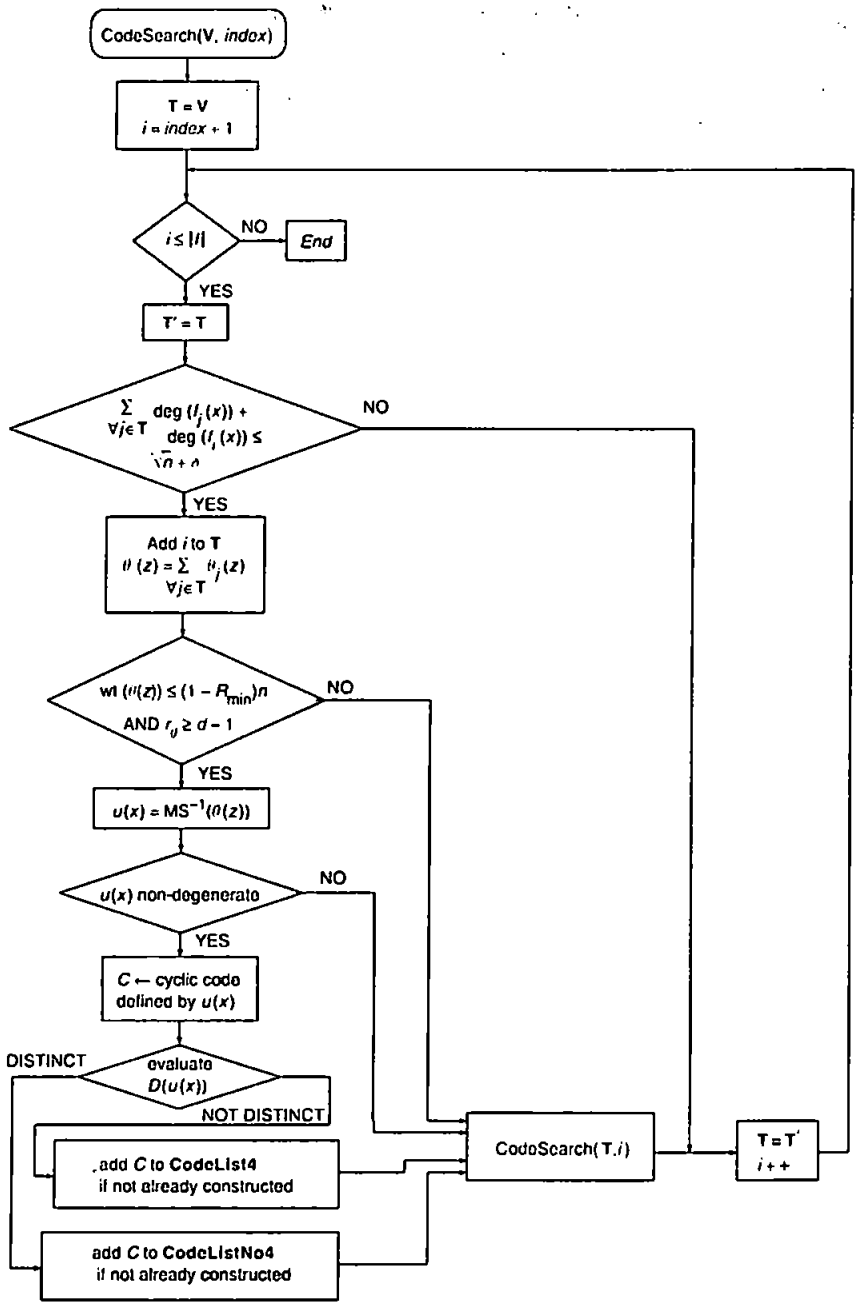


Fig. 5 *Flowchart of the recursive code search algorithm*

The recursive code search algorithm exits in one branch, that is when the counter i, which is initialised to $index+1$, reaches the number of elements in I, $|I|$. Given δ, the algorithm takes combinations of the binary irreducible polynomials of $z^n - 1$ for which the summation of their degrees does not exceed $\sqrt{n}+\delta$. Suppose that $f(z)$ is the product of the combinations of irreducible polynomials, we compute the corresponding primitive idempotent, $\theta(z)$. The idempotent $\theta(z)$ contributes information of the code rate (its weight) and the minimum distance lower bound (its run of consecutive non-zero coefficients) of the resulting cyclic code. The algorithm continues provided that both $\mathrm{wt}(\theta(z)) \leq (1-R_{\min})n$ and $r_0 \geq d-1$ inequalities are satisfied. The inverse Mattson-Solomon polynomial of $\theta(z)$ gives an idempotent $u(x)$, which is the parity check polynomial of the resulting code. Only non-degenerate cyclic codes are of interest in this case. The difference enumerator polynomial $\mathcal{D}(u(x)) = d_0x^0 + d_1x^1 + \cdots + d_{n-1}x^{n-1}$ is evaluated. If the coefficients are distinct (except that of x^0), i.e. $d_i \in \{0,1\}\ \forall i \in \{1,2,\ldots,n-1\}$, then the cyclic code is added to **CodeListNo4**. Otherwise, it is added to **CodeList4**. We assume that both lists contain codes that have not been constructed earlier from different combinations of the irreducible polynomials. The algorithm continues recursively to consider the possibility of adding further weight to the idempotent $u(x)$.

8.2 Received-vector-coordinate-modification decoder

A non-convergent block output from an iterative decoder can be made to converge to a correct solution by restarting the iterative decoder after altering the channel probability of one of the symbols in such a way that the residual error of that specific component becomes zero. Based on this observation, we can build an improved belief propagation iterative decoder. This modified iterative decoder, known as the received-vector-coordinate-modification (RVCM) decoder, was first proposed by Papagiannis *et al.* [15] to improve

the convergence of the binary serially-concatenated turbo codes. This algorithm can be generalised to any iteratively-decodable linear codes over GF(q), as outlined below.

Algorithm 1 RVCM Algorithm for $[n, k]$ Linear Codes over GF(q)

Input:

Received vector $\mathbf{r} = (r_0, r_1, \ldots, r_i, \ldots, r_{n-1})$, where $r_i \in \mathbf{R}$

i_{max}, where $0 \le i_{max} \le n-1$ and $\mathcal{I} = \{0, 1, \ldots, i_{max}-1\}$

$\pi \Leftarrow$ some permutation of i_{max} symbols, i.e. $\pi(\mathcal{I}) = (\pi(0), \pi(1), \ldots, \pi(i_{max}-1))$

Output:

$\mathbf{z}$ = a codeword whose euclidean distance to $\mathbf{r}$ is minimum.

```
Z = 0
for 0 ≤ i ≤ i_max − 1, do
  r' = r,
  for 0 ≤ s ≤ q − 1, do
    r'_π(i) = s,
    z' ⇐ the iterative decoder output for the received vector of r',
    if z' ∈ C then Z = Z ∪ z',
  end for
end for
Output z where d_E(r, Φ(z)) = min_{∀z' ∈ Z} d_E(r, Φ(z')),
```

Here Φ is the function that maps a binary vector to its corresponding BPSK vector and $d_E(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between the vectors $\mathbf{x}$ and $\mathbf{y}$. For the received vector $\mathbf{r}$, let its corresponding reliability vector be $\mathbf{L} = (L_0, L_1, \ldots, L_i, \ldots, L_{n-1})$, so that $L_i = \max_{s=0}^{q-1} p(r_i|s)$ where $p(r_i|s)$ is the probability density function of the channel output r_i conditioned on the transmitted symbol s. There is a permutation π on the set $\{0, 1, \ldots, n-1\}$ such that $L_{\pi(0)} < L_{\pi(1)} < \ldots < L_{\pi(i)} < \ldots < L_{\pi(n-1)}$.

The complexity of the RVCM algorithm depends on i_{max}. One of the major obstacles concerning this algorithm is the difficulty in finding symbol(s), which if modified, will cause the iterative decoder to converge to the ML solution. These symbols are referred as the critical symbols and their distribution is uniform with no sign of vulnerable or favourite symbol positions. Owing to their uniform distribution, however, it is possible that one of the critical symbols could be found if the search is confined to a small set, i.e. keeping the value of i_{max} low.

Cyclotomic idempotent-based binary cyclic codes

C. Tjhai, M. Tomlinson, M. Ambroze and M. Ahmed

It is shown that idempotents based on cyclotomic cosets may be used to construct some new one-step majority-logic codes. The construction method produces the dual code idempotent which can be used to directly define the parity-check matrix of the cyclic code. A feature of the cyclotomic idempotent codes is the incremental approach to the sparseness of the parity-check matrix and the property of parity-check bit orthogonality which is known to be useful in belief-propagation decoding. These codes have high minimum distance and a performance 0.4 dB away from the theoretical limit is demonstrated.

Introduction: It is well known that an (n, k, t) binary cyclic code C may be generated by its idempotent $E_c(x)$ as an alternative to the generator polynomial $g(x)$ [1–3]. Any idempotent $E(x)$ has the key property that $E(x)^2 = E(x)$ with the consequence that the coefficient of the Mattson-Solomon polynomial only take values from GF(2). C may contain several idempotents but only one idempotent $E_c(x)$ that will generate the code and GCD($E_c(x)$; $1+x^n$) is $g(x)$ where GCD denotes the greatest-common-divisor. In this Letter $C^{\perp}$, the dual code of C, is first derived from its idempotent because the minimum weight codeword of $C^{\perp}$ is used to define a sparse parity-check of C.

Difference-set cyclic (DSC) and one-step majority-logic (OSML) codes have been given renewed attention [4] owing to their good performance as low-density parity-check (LDPC) codes. The parity-check equations of the codes are orthogonal on each bit position of the codeword which implies the absence of cycles of length 4. The minimum distance (d_{min}) can be directly determined from the weight of the lowest weight codeword of $C^{\perp}$. Moreover as the code is cyclic, there are n low weight parity-check equations to use in the iterative decoder instead of $n-k$ low weight parity-check equations as in the case of a random LDPC code. This leads to improved performance.

Code construction: The cyclotomic coset of C is defined as follows:

$$C_i = \{i, 2i, 2^2 i, \ldots, 2^r i\} \bmod n \quad \{\mathcal{N} : 1 \le i \le m\} \tag{1}$$

where r is the smallest integer so that $2^{r+1} i \equiv i \bmod n$ and m is the smallest integer so that $\cup_{\forall i \in \mathcal{N}} C_i = \{1, 2, \ldots, n-1\}$. The cyclotomic idempotent, i.e. idempotent given by the cyclotomic coset, is defined as $E_i(x) = \sum_{\forall s \in C_i} x^s$.

In this construction, the parity-check matrix of C is defined by an idempotent $E_s(x) = m(x)\ h(x)$ where $h(x)\ g(x) = 1 + x^n$ and $m(x)$ consists either of repeated factors of $h(x)$ or non-factors of $1 + x^n$. The idempotent $E_s(x)$ consists of at least one cyclotomic idempotent:

$$E_s(x) = \sum_{\{M : i \in \mathcal{N}\}} E_i(x) \tag{2}$$

For a code to be an OSML code all of the cyclic differences between the exponents of $E_s(x)$ must be distinct and, additionally, to be a DSC code all of the cyclic differences must be present. The cyclic differences are given by the difference enumerator polynomial excluding the x^0 component:

$$\mathcal{D}(E_s(x)) = E_s(x) E_s(x^{-1}) \tag{3}$$

Note that $\mathcal{D}(E_s(x))$ is evaluated with real coefficients and $E_s(x^{-1})$ is the code idempotent of $C^{\perp}$.

Based on the fact that all idempotents consist of the sum of a number of cyclotomic idempotents and the product of two cyclotomic idempotents is equal to a sum of cyclotomic idempotents, $\mathcal{D}(E_s(x))$ of OSML codes must be equal to the sum of distinct cyclotomic idempotents and $\mathcal{D}(E_s(x))$ of DSC codes must be equal to the sum of all cyclotomic idempotents. The conditions on $\mathcal{D}(E_s(x))$ are determined by conditions on $E_s(x)$, which are:

Condition 1: The idempotent $E_s(x)$ must be chosen such that wt $(E_s(x))$ (wt $(E_s(x)) - 1) \le n - 1$ where wt $(f(x))$ is the weight of $f(x)$.

Proof: Each of the differences must be distinct and nonzero. The number of distinct nonzero differences is wt $(E_s(x))$ (wt $(E_s(x)) - 1$) and this must be less than or equal to the total number of distinct integers between 1 and $n-1$.

Condition 2: Following (2) unless the wt $(E_i(x)) = 2$, $E_i(x)$ must not be self-reciprocal, i.e. $E_i(x) \ne E_i(x^{-1})$, $\forall i \in \mathcal{M}$.

Proof: The sum of nonzero coefficients of $\mathcal{D}(E_i(x))$ is equal to wt $(E_i(x))$(wt$(E_i(x)) - 1$). For a self-reciprocal component, $E_i(x)\ E_i(x^{-1}) = E_i(x)^2 = E_i(x)$ with wt $(E_i(x))$ coefficients. For the differences to be distinct wt $(E_i(x))$ (wt $(E_i(x)) - 1)) \le$ wt $(E_i(x))$ and hence wt $(E_i(x))$ must be equal to 2 or less.

Condition 3: Following (2), $E_s(x)$ must not contain $E_i(x^{-1})$, $\forall i \in \mathcal{M}$ unless $E_i(x)$ is self-reciprocal.

Proof: If $E_s(x)$ does contain $E_i(x^{-1})$ for $i \in \mathcal{M}$ then $\mathcal{D}(E_s(x))$ will contain both $E_i(x)\ E_i(x^{-1})$ and $E_i(x^{-1})\ E_i(x)$, hence the same differences will be produced.

Condition 4: Following (3), wt $(\mathcal{D}(E_s(x)))$ ignoring the x^0 component must be equal to wt $(E_s(x))$ (wt $(E_s(x)) - 1$).

Proof: For all of the nonzero differences to be distinct, each one must be a separate exponent of $\mathcal{D}(E_s(x))$.

Another condition is that the exponents of $E_s(x)$ must not contain a common factor of n, otherwise a degenerate code is produced. Although the above conditions seem overly restrictive they turn out to be helpful in code construction. After enumerating all of the cyclotomic idempotents for length n, codes may be designed step-by-step by adding candidate idempotents to $E_s(x)$ checking the above conditions at each stage. Moreover, by adding the candidate idempotents in a recursive manner, all cyclic codes with no cycles of length 4 may be generated.

Following MacWilliams [2] we can define C entirely by $E_s(x)$ avoiding the need to determine $g(x)$ and $h(x)$. With α defined as the primitive nth-root of unity, $E_s(\alpha^i)$ for $1 \le i \le n$ is either 1 or 0 as a consequence of the property $E_s(x)^2 = E_s(x)$. With α^j being the roots of $g(x)$, $1 \le j \le n-k$ and as $E_s(x)$ does not contain α^j as roots, it follows that $E_s(\alpha^j) = 1$ for $1 \le j \le n-k$. For GF(2), $1 + E_s(\alpha^j) = 0$ and the code idempotent is $E_c(x) = 1 + E_s(x)$. $E_c(x)$ may be used to generate the code as an alternative to $g(x)$. The dimension k of C is equal to the number of zeros of the Mattson-Solomon polynomial [1] of $E_s(x)$. Note that $d_{min} \le$ wt $(E_c(x))$, i.e. wt $(E_c(x)) + 1$. Because of space limitation, only a limited number of cyclotomic idempotent codes are included in Table 1.

Table 1: Examples of cyclotomic idempotent codes

(n, k)	Cosets	d_{min}	(n, k)	Cosets	d_{min}
(63,37)	{1,21}	9	(219,101)	{3,73}	12
(255,135)	{1,119}	13	(255,175)	{1,27}	17
(273,191)	{1,91,117}	18	(341,205)	{1,55}	16
(511,119)	{5,37}	19	(819,447)	{1,351}	19
(1023,661)	{1,53,341}	23	(1023,781)	{1,53,123,341}	33
(1057,813)	{5,43,151}	34	(1387,783)	{1,247}	28
(1917,1105)	{1,657}	21	(2047,1167)	{1,27}	23
(2325,1335)	{1,75,775}	28	(2325,1373)	{1,525,1085}	30
(2359,1347)	{1}	22	(3741,2229)	{1}	29
(3813,2087)	{1,369,1271}	28	(4095,2767)	{1,41,235,733}	49
(4095,3367)	{1,41,235,273, 411,733}	65	(4161,3431)	{1,285,307,357,1387}	66
(4161,2827)	{1,307,1387}	39	(4681,2681)	{1,51}	31

Simulation results: Computer simulations have been carried out for the (63,37) cyclotomic idempotent code and a (63,37) irregular random code and the frame error rate (FER) performance is shown in Fig. 1. It is clear that, under standard BP decoding, the performance of the cyclotomic idempotent code is superior to that of the irregular code. The modified BP decoder [5], which approximates

the maximum-likelihood decoder, realises the full performance of the cyclotomic idempotent code showing that it is about 0.4 dB away from the best theoretical (63,37) binary code [6] at 10^{-3} FER.

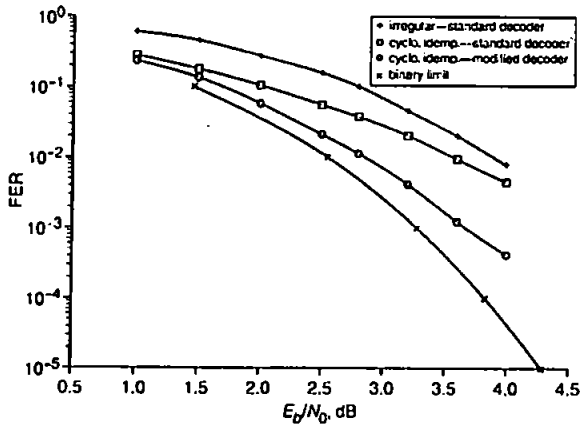


Fig. 1 *FER of (63,37) cyclotomic idempotent and irregular code*

Conclusions: The construction method based on summing the cyclotomic idempotents is able to produce a large number of cyclic codes with no cycles of length 4. Some of these codes are equal to DSC codes, Euclidean-Geometry (EG) and Projective-Geometry (PG) codes which are already well known, but some other codes are new. The ability to increment the d_{min} of the code by adding further weight from other cyclotomic idempotents and so steadily decrease the sparseness of the parity-check matrix is an interesting property. As an example consider n of 1023, with the idempotent defined by C_1, a (1023,511,5) code is produced and adding C_{53} produces (1023,463,10) code. Adding C_{341} to the idempotent results in (1023,661,11) code and finally a (1023,781,16) code is produced by further adding C_{123}.

The cyclotomic idempotent codes as a class are not as good as BCH codes in that there tends to be a BCH code of similar length and d_{min} but with higher rate. However the BCH codes have dense parity-check matrices and thus are not suitable for BP decoding.

As shown by the simulation results the cyclotomic idempotent codes can have outstanding performance.

4 October 2004
Electronics Letters online no: 20057266
doi: 10.1049/el:20057266

C. Tjhai, M. Tomlinson, M. Ambroze and M. Ahmed (*Communication Research Group, University of Plymouth, Drake Circus, Plymouth PL4 8AA, United Kingdom*)

E-mail: cen.tjhai@plymouth.ac.uk

References

1 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error-correcting codes' (North-Holland, 1977)
2 MacWilliams, F.J.: 'A table of primitive binary idempotents of odd length n, $7 \le n \le 511$', *IEEE Trans. Inf. Theory*, 1979, **IT-25**, pp. 118–123
3 Shibuya, T., and Sakaniwa, K.: 'Construction of cyclic codes suitable for iterative decoding via generating idempotents', *IEICE Trans. Fundam.*, 2003, E86-A, (4),
4 Kou, Y., Lin, S., and Fossorier, M.: 'Low density parity check codes based on finite geometries: rediscovery and new results', *IEEE Trans. Inf. Theory*, 2001, 47, pp. 2711–2736
5 Tjhai, C.J., Papagiannis, E., Tomlinson, M., Ambroze, M.A., and Ahmed, M.Z.: 'Improved iterative decoder for LDPC codes with performance approximating to a maximum likelihood decoder', UK Patent Application 0409306.8
6 Shannon, C.E.: 'Probability of error for optimal codes in a gaussian channel', *Bell Syst. Tech. J.*, 1959, 38, pp. 611–656

Conferences

Horan, R., Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M. (2005). A finite-field transform domain construction of binary low-density parity-check codes, *Proc. IEEE Information Theory Workshop on Coding and Complexity*, Rotorua, New Zealand, pp. 72–76.

Ntokas, I., Nutter, P., Middleton, B., Tjhai, C. and Ahmed, M. (2005). Performance evaluation of ldpc codes for patterned media, *Digests of IEEE International Conference on Magnetics (INTERMAG)*, Nagoya, Japan, p. 494.

Ntokas, I., Nutter, P., Middleton, B., Tjhai, C. and Ahmed, M. (2006). Read channel designs for efficient data recovery in storage systems employing imperfect patterned media, *Digests of IEEE International Conference on Magnetics (INTERMAG)*, San Diego, USA, p. 793.

Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006a). GF(2^m) low-density parity-check codes derived from cyclotomic cosets, *Proc. 4th International Symposium on Turbo Codes in connection with 6th International ITG-Conference on Source and Channel Coding*, Munich, Germany.

Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006b). On the efficient codewords counting algorithm and the weight distribution of the binary quadratic double-circulant codes, *Proc. IEEE Information Theory Workshop*, Chengdu, China, pp. 42–46.

Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M. and Ambroze, M. (2006c). Some results on the weight distributions of the binary double-circulant codes based on primes, *Proc. 10th IEEE International Conference on Communication Systems (ICCS)*, Singapore.

Tjhai, C., Tomlinson, M., Horan, R., Ambroze, M. and Ahmed, M. (2005). Near maximum-likelihood performance of some new cyclic codes constructed in the finite-field transform domain, *Proc. 8th International Symposium Communication Theory and Applications*, Ambleside, Lake District, UK, pp. 194–199.

Tjhai, C., Tomlinson, M., Perry, P. and Fossorier, M. (2007). Comparison of Bose-Chaudhuri-Hocquenghem and Goppa codes for error detection, *Proc. 9th International Symposium on Communication Theory and Applications*, Ambleside, Lake District, UK.

Comparison of Bose-Chaudhuri-Hocquenghem and Goppa Codes for Error Detection

C. Tjhai, M. Tomlinson
Fixed and Mobile Communications Research,
University of Plymouth,
Plymouth, PL4 8AA, United Kingdom
{ctjhai,mtomlinson}@plymouth.ac.uk

P. Perry
Mathematics Department,
Hawaii Pacific University,
Honolulu, Hawaii 96813, USA
pperry@hpu.edu

M. Fossorier
Dept. of Electrical Engineering,
University of Hawaii,
Honolulu, Hawaii, USA
marc@spectra.eng.hawaii.edu

***Abstract*—In coding for error detection over the q-ary symmetric channel with cross-over probability p, it is well known that the probability of decoder failure or the probability of undetected error is dictated by the weight distribution of the code. In this paper, some binary and ternary primitive BCH and irreducible Goppa codes of the same parameters are compared in terms of their error detection capabilities. Using the $[128, 85, 14]_2$, $[128, 92, 12]_2$ $[256, 207, 14]_2$ and $[256, 215, 12]_2$ extended BCH and shortened irreducible Goppa codes as case studies, it is found that the shortened irreducible Goppa codes have lower expected value for probability of decoder failure over the interval $p \in [0, 1]$. It is also found that apart from the codes with parameters $[128, 85, 14]_2$, the shortened irreducible Goppa codes are better suited for error detection over the intervals $p \in [0, \frac{1}{2}]$ and $p \in [0, \frac{1}{3}]$. On the other hand, for ternary BCH and Goppa codes, $[80, 60]_3$ and $[242, 222]_3$, BCH codes are preferable due to their superior minimum distance.**

I. Introduction

A typical communication system uses channel coding to ensure reliable transmission of information from source to destination. Channel coding can be employed in two different ways: error correction and error detection. In either case, a message is split into several data blocks and for each data block, some redundant symbols are obtained (these redundant symbols are functions of the symbols in the data block) and appended to each data block. In transmission, instead of sending strings of data blocks only, strings of data and redundant symbols are transmitted. In the error correction case, these redundant symbols are used by the decoder in the receiving end to correct any errors that may have occurred during transmission. In the error detection case, channel coding is used in conjunction with automatic-repeat-request (ARQ) retransmission scheme. Upon the receipt of the data and redundant symbols, the decoder recomputes a new set of redundant symbols based on the received data block. If the recomputed redundant symbols are the same as those of the received, successful transmission is declared, otherwise an error is detected and retransmission is requested. In this paper, we consider the application of channel coding for error detection.

When a decoder requests retransmission, it is certain that some errors have occurred. On the contrary, when a decoder declares successful transmission, this decision may not necessarily be correct. There is a possibility that errors during transmission have transformed the transmitted symbols to a pattern which is not detectable by the decoder as an error. The probability of this event is known as the probability of decoder failure or the probability of undetected error.

Throughout the development of channel coding since the early 1950s, there exists many good linear codes. One of the many popularly used codes is a class known as the Bose-Chaudhuri-Hocquenghem (BCH) codes. Given an extended BCH code of length which is a power of a prime q, there also exists a shortened Goppa code of the same length and dimension. The question addressed here is which class of codes has better error-detecting capability. In the other words, we are interested to know which of the two codes has lower probability of decoder failure. As a preliminary study, we consider some high-rate binary and ternary BCH and Goppa codes and determine their probability of undetected error and also their expected value.

II. Background and Notation

Let $\mathbb{F}_q^n$ denote the space of vectors of length n with elements in $\mathbb{F}_q$. A linear code over $\mathbb{F}_q$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$. We denote $[n, k, d]_q$ as a linear code over $\mathbb{F}_q$ of length n, dimension k and minimum distance d. The Hamming weight enumerator function of a code is defined as $A(z) = \sum_{i=0}^{n} A_i z^i$, where A_i denotes the number of codewords of Hamming weight i. If C is a linear code over $\mathbb{F}_q$, its dual code $C^\perp$ is defined as $C^\perp = \{w \in \mathbb{F}_q^n | \sum_{i=0}^{n-1} v_i w_i = 0, \text{ for all } v \in C\}$. Let $A(z)$ and $B(z)$ be the weight enumerator functions of C and $C^\perp$ respectively, from the MacWilliams Identity for Hamming weight enumerator [1][2], we know that the relationship of $A(z)$ and $B(z)$ is given by

$$A(z) = \frac{(1+(q-1)z)^n}{|C^\perp|} B\left(\frac{1-z}{1+(q-1)z}\right) = \frac{1}{|C^\perp|} \sum_{i=0}^{n} B_i (1-z)^i (1+(q-1)z)^{n-i} \quad (1)$$

where $|C^\perp|$ is the size of $C^\perp$, which is q^{n-k}.

In a typical ARQ system, given a user data $u \in F_q^k$, we map u of length k to $c \in C$ of length n, which is a vector that we transmit. Errors may have occurred during the transmission and the decoder receives $r \in \mathbb{F}_q^n$, where $r = c + e$

and $e \in \mathbb{F}_q^n$. The decoder requests retransmission if $r \notin C$, otherwise successful transmission is declared. The declaration of successful transmission occurs for any r regardless of c where $r \in C$. Therefore, an error will not be detected by the decoder if the error vector e has transformed the transmitted vector c into another codeword of C. Since, we consider linear codes, this means an undetected error will occur if $e \in C$ and $\mathrm{wt}(e) > 0$, where $\mathrm{wt}(x)$ denotes the weight of vector x.

We consider a discrete memoryless channel with q inputs and q outputs, a q-ary symmetric channel (qSC). Let p be the symbol error probability so that the probability of a symbol being received correctly is $1-p$. The probability of a symbol being changed to a specific other symbol is simply $\frac{p}{q-1}$. It is assumed that $0 \leq \frac{p}{q-1} \leq p$. Hence, the probability of undetected error with symbol error probability p of code C, $P_{ue}(C,p)$, is simply

$$\begin{aligned} P_{ue}(C,p) &= P(c \in C \text{ and } \mathrm{wt}(c) > 0) \\ &= \sum_{i=1}^{n} A_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i}. \end{aligned} \tag{2}$$

Using the MacWilliams Identity, $P_{ue}(C,p)$ can also be expressed in terms of $B(z)$:

$$\begin{aligned} P_{ue}(C,p) &= \left[\sum_{i=0}^{n} A_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i}\right] - (1-p)^n \\ &= \left[A\left(\frac{p}{(q-1)(1-p)}\right) - 1\right](1-p)^n \\ &= \left[\frac{(1-p)^{-n}}{|C^\perp|} B\left(\frac{q-qp-1}{q-1}\right) - 1\right](1-p)^n \\ &= -(1-p)^n + \\ &\quad \left[\frac{1}{|C^\perp|}\sum_{i=0}^{n} B_i \left((1-p) - \frac{p}{q-1}\right)^i\right]. \end{aligned} \tag{3}$$

It is obvious from (2) and (3) that the weight distribution of the code or that of the dual code may be used. Optimising A_i or B_i will result in reduced probability of undetected error.

III. The Expected Value of the Probability of Decoder Failure

It is shown in [3], that the expected value of the probability of undetected error can be used to determine how good a code is for error detection. Let p be a random variable with a uniform distribution over some interval $[0,x]$, the expected value of the probability of undetected error with symbol error probability p of code C, $E[P_{ue}(C,p)]$, is defined as [3]

$$E[P_{ue}(C,p)] = \int_0^x f(p) P_{ue}(C,p) \mathrm{d}p, \tag{4}$$

where $f(p)$, the probability density function of the uniformly distributed random variable p, is

$$f(p) = \begin{cases} x^{-1} & \text{if } 0 \leq p \leq x \\ 0 & \text{otherwise.} \end{cases}$$

For C over $\mathbb{F}_q$ [4],

$$\begin{aligned} E[P_{ue}(C,p)] &= \int_0^x \frac{1}{x} \sum_{i=1}^{n} A_i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i} \mathrm{d}p \\ &= \frac{1}{x} \sum_{i=1}^{n} \left(\frac{1}{q-1}\right)^i A_i \beta(x; i+1, n-i+1) \end{aligned} \tag{5}$$

where

$$\beta(x; i+1, n-i+1) = \int_0^x p^i (1-p)^{n-i} \mathrm{d}p$$

is an incomplete Beta function. For the reason which will be stated later, the expression of $E[P_{ue}(C,p)]$ involving $B(z)$ is more useful than (5) in this paper. Equipped with (3), $E[P_{ue}(C,p)]$ can be rewritten in alternative form as [4]

$$\begin{aligned} E[P_{ue}(C,p)] &= -\int_0^x \frac{1}{x}(1-p)^n \mathrm{d}p + \\ &\quad \int_0^x \frac{1}{x}\left[\frac{1}{|C^\perp|}\sum_{i=0}^{n} B_i \left((1-p) - \frac{p}{q-1}\right)^i\right] \mathrm{d}p \\ &= -\left[\frac{1-(1-x)^{n+1}}{x(n+1)}\right] + \\ &\quad \frac{x^{-1}}{|C^\perp|}\sum_{i=0}^{n} B_i \left[\left(1 - \frac{qx}{q-1}\right)^i \frac{qx}{q(i+1)} - \right. \\ &\quad \left. \frac{q-1}{q(i+1)}\left\{\left(1 - \frac{qx}{q-1}\right)^i - 1\right\}\right] \end{aligned} \tag{6}$$

In the binary case, $q = 2$, $E[P_{ue}(C,p)]$ simply reduces to

$$\begin{aligned} E[P_{ue}(C,p)] &= -\left[\frac{1-(1-x)^{n+1}}{x(n+1)}\right] + \\ &\quad \frac{x^{-1}}{|C^\perp|}\sum_{i=0}^{n} B_i \left[\frac{1-(1-2x)^{i+1}}{2(i+1)}\right]. \end{aligned} \tag{7}$$

Given some $[n,k,d]_q$ linear codes, $E[P_{ue}(C,p)]$ can be numerically evaluated to determine which code is best suited for error detecting applications over an interval $[0,x]$ of cross-over probability p, which is assumed to be uniformly distributed. In a later section, $E[P_{ue}(C,p)]$ is used to compare the error detecting performances of some BCH and Goppa codes, over $\mathbb{F}_2$ and $\mathbb{F}_3$, having the same length and code rate for $x = 1$, $x = \frac{1}{2}$ and $x = \frac{1}{3}$.

IV. BCH and Goppa Codes and Their Weight Distributions

In this paper, t-error correcting primitive BCH and irreducible Goppa codes over $\mathbb{F}_2$ and $\mathbb{F}_3$, which are well-studied in the literature, are considered. A brief outline of the theory of these two classes of codes is given below.

A. BCH Codes

A t-error correcting primitive BCH code is a cyclic code of length $q^m - 1$, for some integer m, which is able to correct up to t errors. Let α be a primitive nth root of

unity of $\mathbb{F}_q^m$, a t-error correcting primitive BCH code has a generator polynomial $g(x)$ which has $2t$ consecutive powers of α, i.e. $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$ (The integer b is usually chosen to be 1, and the BCH code of this type is known as a narrow-sense BCH code). The dimension of the BCH code is $q^m - \deg(g(x)) - 1$, where $\deg(f(x))$ denotes the degree of polynomial $f(x)$, and its minimum distance is at least $2t+1$. The parity-check matrix of a $[q^m-1, k, \geq 2t+1]_q$ primitive BCH code, $H_{i,j} = (\alpha^{j(b+i)})$ for $0 \leq i < 2t$ and $0 \leq j < q^m - 1$

$$H = \begin{bmatrix} 1 & \alpha^b & \cdots & \alpha^{(q^m-2)b} \\ 1 & \alpha^{b+1} & \cdots & \alpha^{(q^m-2)(b+1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{b+2t-1} & \cdots & \alpha^{(q^m-2)(b+2t-1)} \end{bmatrix} \quad (8)$$

A primitive BCH code of length $q^m - 1$ may be extended by annexing an overall parity check to increase the minimum distance and length by one. For the binary case, the weight distributions of many primitive BCH codes and the extended codes are well documented, for example see [5], [6], and this is one reason why we have chosen these codes in this paper. We consider the 5- and 6-error correcting extended BCH codes of lengths 128 and 256 over $\mathbb{F}_2$. These codes, whose weight distributions are given in [5], [6], are $C_1 = [128, 85, 14]_2$, $C_2 = [128, 92, 12]_2$, $C_3 = [256, 207, 14]_2$ and $C_4 = [256, 215, 12]_2$ and their dual codes $C_i^\perp$ are $[128, 43, 32]_2$, $[128, 36, 32]_2$, $[256, 49, 64]_2$ and $[256, 41, 64]_2$ respectively. Using the results in [5], [6] and the MacWilliams Identity, the Hamming weight distributions of $C_i^\perp$, for $1 \leq i \leq 4$, are obtained and tabulated in Tables I and II.

In addition to these binary codes, we also construct $C_5 = [80, 60, 8]_3$ and $C_6 = [242, 222, 7]_3$ BCH codes and the Hamming weight distributions of the dual of these ternary codes ($C_5^\perp = [80, 20, 30]_3$, and $C_6^\perp = [242, 20, 126]_3$ respectively), which are obtained using GUAVA [7], are tabulated in Tables V and VI.

B. Goppa Codes

Let $G(z)$ be a polynomial with coefficients from $\mathbb{F}_q^m$ of degree r and let β_j be the jth element of $\mathbb{F}_q^m$. If we now replace $(\alpha^{j(b+i)})$ in (8) with $(\beta_{j+1}^i G(\beta_{j+1})^{-1})$ for $0 \leq i < r$, a new matrix $\mathcal{H}$ is obtained

$$\mathcal{H} = \begin{bmatrix} \frac{1}{G(\beta_1)} & \frac{1}{G(\beta_2)} & \cdots & \frac{1}{G(\beta_{q^m})} \\ \frac{\beta_1}{G(\beta_1)} & \frac{\beta_2}{G(\beta_2)} & \cdots & \frac{\beta_{q^m}}{G(\beta_{q^m})} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\beta_1^{r-1}}{G(\beta_1)} & \frac{\beta_2^{r-1}}{G(\beta_2)} & \cdots & \frac{\beta_{q^m}^{r-1}}{G(\beta_{q^m})} \end{bmatrix}, \quad (9)$$

which is the parity-check matrix of a Goppa code over $\mathbb{F}_q$. The polynomial $G(z) = \sum_{i=0}^{n-1} g_i z^i$ is known as the Goppa polynomial. If $G(z)$ is irreducible over $\mathbb{F}_q^m$, the resulting code is a $[q^m, k, d]_q$ irreducible Goppa code. As a class, Goppa codes contain the narrow-sense primitive BCH codes. From (8) and (9), it can be seen that if we let $G(z) = z^r$, $b = 1$ (narrow-sense) and $\beta_j = \alpha^{j-1}$, the columns of $\mathcal{H}$, for which $\beta_j \neq 0$, become H [1].

TABLE I

WEIGHT DISTRIBUTIONS OF BINARY CODES: $C_1^\perp$, $\Gamma_1^\perp$, $C_2^\perp$ AND $\Gamma_2^\perp$

i	B_i of $C_1^\perp$	B_i of $\Gamma_1^\perp$	B_i of $C_2^\perp$	B_i of $\Gamma_2^\perp$
0	1	1	1	1
26	0	14	0	0
27	0	14	0	0
28	0	315	0	0
29	0	560	0	0
30	0	4103	0	21
31	0	8652	0	28
32	124460	43813	10668	70
33	0	82474	0	406
34	0	367311	0	1778
35	0	659092	0	5068
36	8810752	2553502	16256	14288
37	0	4455458	0	38879
38	0	14939232	0	97748
39	0	25371472	0	228732
40	263542272	74806634	2048256	521696
41	0	124227908	0	1116227
42	0	325466258	0	2314879
43	0	526423780	0	4640031
44	4521151232	1233207653	35551872	8887707
45	0	1940111516	0	16529051
46	0	4083199400	0	29751750
47	0	6227830742	0	51829218
48	44899876672	11839034032	353494848	87414166
49	0	17450194650	0	142753082
50	0	30114370552	0	225845752
51	0	42776504596	0	345723099
52	262118734080	67353199823	2028114816	512303743
53	0	91925083068	0	735020965
54	0	132713265006	0	1020845539
55	0	173505905412	0	1373019683
56	915924097536	230745825270	7216135936	1789255624
57	0	287996387752	0	2259040886
58	0	354384710354	0	2764278666
59	0	420768878768	0	3279535861
60	1931974003456	481152508403	14981968512	3771725846
61	0	541519789580	0	4205861926
62	0	577875305546	0	4546798459
63	0	614231340058	0	4764526042
64	2476672341286	614220896660	19484794406	4839622902

Given a $[q^m - 1, k, d]_q$ primitive BCH code, in many cases, there also exists a $[q^m, k+1, d']_q$ irreducible Goppa code. In order to have codes of the same length and dimension, we can either 1) shorten the irreducible Goppa code; or 2) extend both codes and then shorten the extended Goppa code.

For comparison purposes, shortened extended irreducible binary Goppa codes, Γ_i, of the same parameters (n, k and d) as those of C_i, for $1 \leq i \leq 4$ are constructed. Unlike the BCH codes, the weight distribution of Goppa codes is not widely known. In order to compute $P_{uc}(\Gamma_i, p)$ and its expected value, $A(z)$ for each Goppa code has to be determined. However, since the dimension of Γ_i is larger than $n - k$, it is more efficient to deal with $\Gamma_i^\perp$ to obtain $B(z)$, which can be used to evaluate the probability of decoder value and also its expected value from (3) and (7) respectively. The dual codes $\Gamma_i^\perp$, for $1 \leq i \leq 4$, are $[128, 43, 26]_2$, $[128, 36, 30]_2$, $[256, 49, 66]_2$ and $[256, 41, 76]_2$ respectively. We have computed the Hamming weight distributions of these dual codes exhaustively by splitting the codeword enumerations on grid computers.

TABLE II

WEIGHT DISTRIBUTIONS OF BINARY CODES: $C_3^\perp$, $\Gamma_3^\perp$, $C_4^\perp$ AND $\Gamma_4^\perp$

i	B_i of $C_3^\perp$	B_i of $\Gamma_3^\perp$	B_i of $C_4^\perp$	B_i of $\Gamma_4^\perp$
0	1	1	1	1
64	2380	0	2380	0
66	0	4	0	0
68	0	6	0	0
70	0	32	0	0
71	0	8	0	0
72	0	262	0	0
73	0	240	0	0
74	0	980	0	0
75	0	2480	0	0
76	0	6868	0	12
77	0	16400	0	0
78	0	44676	0	56
79	0	101864	0	40
80	1109760	248305	0	411
81	0	551816	0	792
82	0	1259956	0	2452
83	0	2642280	0	5224
84	20889600	5700092	0	14098
85	0	11507496	0	30056
86	0	23612460	0	68332
87	0	45753672	0	140472
88	362565120	89735664	783360	298732
89	0	167507664	0	585248
90	0	315172736	0	1150032
91	0	568642784	0	2132960
92	4076605440	1029255126	14622720	3949642
93	0	1797840736	0	7006744
94	0	3137142904	0	12340180
95	0	5309417328	0	21018968
96	35858333376	8946696483	148737216	35486775
97	0	14678583272	0	58263776
98	0	23901609576	0	94666664
99	0	38012234984	0	150089568
100	238056837120	59844242312	942120960	235326864
101	0	92238297408	0	361449808
102	0	140446070520	0	548843516
103	0	209723747424	0	817691032
104	1240028198400	308930784216	4793902080	1202968312
105	0	446910570200	0	1739673224
106	0	637057073900	0	2480378992
107	0	892762599672	0	3477843856
108	4908429312000	1231932050062	19134873600	4803654944
109	0	1672490161744	0	6527987208
110	0	2234904493124	0	8730342848
111	0	2939644499088	0	11490935480
112	15280308701440	3805070363875	59846440960	14880225073
113	0	4849553108928	0	18967868984
114	0	6082010272324	0	23787385796
115	0	7511362959416	0	29369663008
116	36375691345920	9128782911738	142149550080	35679435538
117	0	10925384150352	0	42684122136
118	0	12868429439732	0	50255272788
119	0	14925100921104	0	58269207056
120	68422918011904	17038499973682	266972882944	66509177532
121	0	19151788306856	0	74754952568
122	0	21192114533768	0	82725113732
123	0	23086749529024	0	90143029568
124	98661457551360	24762784394308	385367162880	96714994390
125	0	26147370900152	0	102150696568
126	0	27186437005404	0	106243390132
127	0	27825813800936	0	108761419544
128	112615534493670	28045588520464	440281097190	109622652088

This splitting is possible due to the nice property of the Revolving-Door combination generator algorithm [8], which will be described in the following section. The Hamming weight distributions of the $\Gamma_i^\perp$, for $1 \le i \le 4$, are tabulated in Tables I and II.

Similarly, ternary Goppa codes $\Gamma_5 = [80, 60, 5]_3$ and $\Gamma_6 = [242, 222, 5]_3$ have been constructed with the same length and dimension as the ternary BCH codes, $C_5 = [80, 60, 8]_3$ and $C_6 = [242, 222, 7]_3$. The respective dual codes are $\Gamma_5^\perp = [80, 20, 25]_3$ and $\Gamma_6^\perp = [242, 20, 119]_3$ and since their dimensions are relatively small, GUAVA [7] have been used to obtain their weight distributions, see Tables V and VI.

V. Computing the Weight Distributions of a Linear Code

The core of all algorithms to evaluate the minimum distance or to compute the number of codewords of a given weight is codeword enumeration. Given a reduced-echelon generator matrix, codewords can be enumerated by taking linear combinations of rows of the generator matrix. Having an efficient combination generator will certainly speed up the process and one of the most efficient algorithms to do so is the Revolving-Door algorithm [8]. An efficient implementation of this algorithm is given in [9] (Algorithm R), which is attributed to Payne and Ives [10].

When the number of codewords that need to be enumerated is large, it is common practice to resort to a multi-threaded approach by splitting the enumerations on multiple computers. The Revolving-Door algorithm has a nice property that allows the splitting to be realised. Let $a_t a_{t-1} \ldots a_1$, where $a_t > a_{t-1} > \ldots > a_1$, be a pattern of a t out of s combination–C_t^s. A pattern is said to have rank i if this pattern appears as the $(i+1)$th element in the list of all C_t^s combinations[1]. Let $\mathrm{Rank}(a_t a_{t-1} \ldots a_1)$ be the rank of pattern $a_t a_{t-1} \ldots a_1$, the Revolving-Door algorithm has the property that

$$\mathrm{Rank}(a_t a_{t-1} \ldots a_1) = \left[\binom{a_t+1}{t} - 1\right] - \mathrm{Rank}(a_{t-1} \ldots a_1)$$

and consequently, an integer N, $0 \le N \le \binom{s}{t} - 1$, can be represented uniquely with an ordered pattern $a_t a_{t-1} \ldots a_1$ [9],

$$N = \binom{a_t+1}{t} - \binom{a_{t-1}+1}{t-1} + \ldots + (-1)^t \binom{a_2+1}{2} - (-1)^t \binom{a_1+1}{1} - (t \bmod 2) \tag{10}$$

As an implication of this, if all $\binom{k}{t}$ codewords need to be enumerated, we can split the enumeration into $\lceil \binom{k}{t}/M \rceil$ blocks where in each block only at most M codewords need to be generated. We know that at the ith block, the enumeration would start from rank $(i-1)M$ and the corresponding pattern can be easily obtained following Algorithm 1.

VI. Error Detecting Performance

Using (3), the probability of decoder failure of C_i and Γ_i, for $1 \le i \le 4$, is compared and plotted in Figure 1. Parts (a), (c) and (d) of Figure 1 show that C_i and Γ_i have similar error detecting performance in the interval $0 \le p \le \frac{k}{n}$, and Γ_i is slightly better at higher p. On the other hand, Figure 1(b) clearly demonstrates that Γ_2 is a better code for error-detection

[1] Here it is assumed that the first element in the list of all C_t^s combinations has rank 0.

TABLE III

$E[P_{ue}(C,p)]$ OF BCH AND GOPPA CODES OVER THE INTERVALS $[0,x]$

	$E[P_{ue}(C,p)]$	x		
		1	1/2	1/3
$\mathbb{F}_2$	C_1	$7.7519379846 \times 10^{-03}$	$8.9541221659 \times 10^{-14}$	$7.7469280990 \times 10^{-14}$
	Γ_1	$9.5373261925 \times 10^{-07}$	$9.0229906879 \times 10^{-14}$	$7.8503176182 \times 10^{-14}$
	C_2	$7.7519379967 \times 10^{-03}$	$1.2247468664 \times 10^{-11}$	$1.1095246250 \times 10^{-11}$
	Γ_2	$1.2579157334 \times 10^{-11}$	$1.1993692497 \times 10^{-11}$	$1.0714581999 \times 10^{-11}$
	C_3	$3.8910505837 \times 10^{-03}$	$1.5916087892 \times 10^{-15}$	$1.4988010832 \times 10^{-15}$
	Γ_3	$2.2262568958 \times 10^{-11}$	$1.5907414275 \times 10^{-15}$	$1.4970663598 \times 10^{-15}$
	C_4	$3.8910505841 \times 10^{-03}$	$4.1658256705 \times 10^{-13}$	$3.9750147618 \times 10^{-13}$
	Γ_4	$2.2683065929 \times 10^{-11}$	$4.1435518211 \times 10^{-13}$	$3.9415866404 \times 10^{-13}$
$\mathbb{F}_3$	C_5	$2.6022301319 \times 10^{-10}$	$2.3364878628 \times 10^{-10}$	$2.0707456936 \times 10^{-10}$
	Γ_5	$3.2835295473 \times 10^{-10}$	$3.6990871108 \times 10^{-10}$	$4.1146446410 \times 10^{-10}$
	C_6	$2.7890836365 \times 10^{-10}$	$2.7101952821 \times 10^{-10}$	$2.6313069278 \times 10^{-10}$
	Γ_6	$3.0616854727 \times 10^{-10}$	$3.2553989547 \times 10^{-10}$	$3.4491124366 \times 10^{-10}$

Algorithm 1 RecursiveCompute a_i $(\text{Rank}(a_i a_{i-1} \ldots a_1), i)$

Input: i and $\text{Rank}(a_i a_{i-1} \ldots a_1)$

Output: a_i

1: Find a_i, where $0 \le a_i < a_{i+1}$, such that $\binom{a_i}{i} \le \text{Rank}(a_i a_{i-1} \ldots a_1) \le \left[\binom{a_i+1}{i} - 1\right]$

2: **if** $i > i$ **then**

3: Compute $\text{Rank}(a_{i-1} \ldots a_1) = \left[\binom{a_i+1}{i} - 1\right] - \text{Rank}(a_i a_{i-1} \ldots a_1)$

4: RecursiveCompute a_i $(\text{Rank}(a_{i-1} \ldots a_1), i-1)$

5: **end if**

6: **return** a_i

than C_2 over the entire interval $0 \le p \le 1$. It is interesting to note that Γ_2 has a non increasing $P_{ue}(\Gamma_2, p)$ for interval $\frac{1}{2} \le p \le 1$.

The expected value of probability of decoder failure, $E[P_{ue}(C,p)]$, provides an alternative way to determine whether or not a code is good for error detection over a given interval of cross-over probability p. The evaluation results for the binary and ternary codes for $E[P_{ue}(C,p)]$ over the intervals $p \in [0,1]$, $p \in [0, \frac{1}{2}]$ and $p \in [0, \frac{1}{3}]$ are tabulated in Table III. From the results in Table III, for binary codes, we can see that, over the interval $[0,1]$, Goppa codes have superior error-detection performance than the corresponding BCH codes. Apart from the case of $[128, 85, 14]_2$, Goppa codes over $\mathbb{F}_2$ are also better codes than binary BCH codes over the intervals $[0, \frac{1}{2}]$ and $[0, \frac{1}{3}]$. These results are evident from the comparison of the weight distributions of the two different classes of codes. The extended BCH codes over $\mathbb{F}_2$, except the $[128, 85, 14]_2$ code, have larger number of minimum weight codewords than the corresponding Goppa codes of the same parameters, see Table IV.

From the results in Table III, for all ternary cases, the Goppa codes have higher expected value of probability of undetected error than the BCH codes over all intervals. This, as can be expected, is attributable to the Goppa codes having lower minimum distance than the corresponding BCH codes.

TABLE IV

NUMBER OF CODEWORDS OF MINIMUM WEIGHT IN C_i AND Γ_i

i	d	A_d	
		C_i	Γ_i
1	14	341376	436073
2	12	1194816	749112
3	14	159479040	155077077
4	12	152592000	120901024

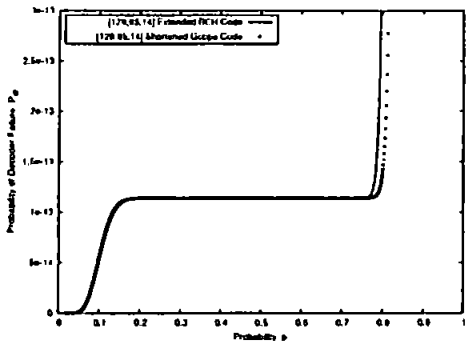
TABLE V

WEIGHT DISTRIBUTIONS OF TERNARY CODES: $C_5^{\perp}$ AND $\Gamma_5^{\perp}$

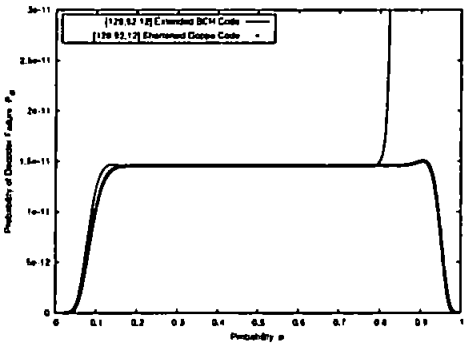
i	B_i of $C_5^{\perp}$	B_i of $\Gamma_5^{\perp}$	i	B_i of $C_5^{\perp}$	B_i of $\Gamma_5^{\perp}$
0	1	1	51	827411040	277195728
25	0	2	52	0	309144814
27	0	10	53	0	326596460
28	0	34	54	983598560	326659346
29	0	62	55	0	308825018
30	3536	218	56	0	275769562
31	0	830	57	694183360	232266736
32	0	2368	58	0	184225808
33	0	6822	59	0	137419354
34	0	18394	60	289896264	96193026
35	0	47926	61	0	63036320
36	436400	118658	62	0	38628490
37	0	281772	63	65663200	22053746
38	0	633114	64	0	11723814
39	3707200	1361306	65	0	5768592
40	0	2784116	66	8014880	2623612
41	0	5435176	67	0	1095830
42	31565040	10091226	68	0	422794
43	0	17835450	69	422400	148026
44	0	30017882	70	0	46746
45	141440480	48050492	71	0	13470
46	0	73161396	72	10180	3500
47	0	105839904	73	0	858
48	440431860	145528364	74	0	132
49	0	190060574	75	0	16
50	0	235646498	76	0	8

REFERENCES

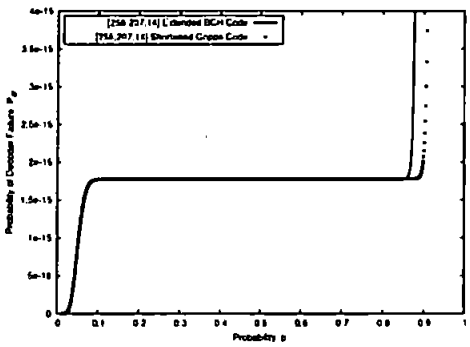
[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.



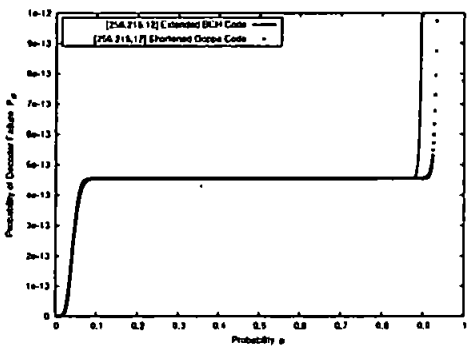
(a) [128, 85, 14]



(b) [128, 92, 12]



(c) [256, 207, 14]



(d) [256, 215, 12]

Fig. 1. Probability of decoder failure of the binary, extended primitive BCH and shortened extended Goppa codes

TABLE VI

WEIGHT DISTRIBUTIONS OF TERNARY CODES: $C_G^\perp$ AND $\Gamma_G^\perp$

i	B_i of $C_G^\perp$	B_i of $\Gamma_G^\perp$	i	B_i of $C_G^\perp$	B_i of $\Gamma_G^\perp$
0	1	1	161	0	192018992
119	0	10	162	1701743516	183149106
120	0	50	163	0	188647060
121	0	84	164	0	173303550
122	0	180	165	0	172381920
123	0	550	166	0	152631080
124	0	1060	167	0	146628608
125	0	2390	168	0	124837850
126	56628	4340	169	0	115472860
127	0	8240	170	0	94260640
128	0	13140	171	726354288	83946240
129	0	24030	172	0	65579594
130	0	40060	173	0	56084640
131	0	68970	174	0	41779700
132	0	109360	175	0	34251970
133	0	177370	176	0	24374790
134	0	271530	177	0	19156820
135	3165360	432890	178	0	13020410
136	0	635240	179	0	9815460
137	0	983270	180	62325648	6378750
138	0	1409560	181	0	4691180
139	0	2145070	182	0	2936376
140	0	2971660	183	0	2103010
141	0	4419210	184	0	1272060
142	0	5984198	185	0	890230
143	0	8721000	186	0	520720
144	109727640	11533820	187	0	360980
145	0	16286960	188	0	201770
146	0	20877500	189	798600	135390
147	0	28681410	190	0	73930
148	0	35564020	191	0	48000
149	0	47023490	192	0	24810
150	0	56208322	193	0	16210
151	0	71784888	194	0	8030
152	0	82597694	195	0	4400
153	882590940	101282440	196	0	1890
154	0	112038570	197	0	1290
155	0	132740810	198	21780	520
156	0	141380040	199	0	300
157	0	161186760	200	0	50
158	0	165437080	201	0	72
159	0	182316770	202	0	20
160	0	180409086			

[2] T. Kløve and V. I. Korzhik, *Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems*. Kluwer Academic Publishers, 1995. ISBN 0 7923 9629 4.

[3] P. Perry and M. Fossorier, "The expected value for the probability of an undetected error," in *Proc. IEEE Information Theory Workshop*, (Chengdu, China), pp. 219–223, 22–26 Oct. 2006.

[4] T. Kløve, "Private communication," 2006.

[5] Y. Desaki and T. Fujiwara and T. Kasami, "The weight distribution of extended binary primitive BCH code of length 128," vol. 43, pp. 1364–1371, Jul. 1997.

[6] T. Fujiwara and T. Kasami, "The weight distribution of (256,k) extended binary primitive BCH code with $k \leq 63$, $k \geq 207$," *Technical Report of IEICE*, vol. IT97-46, pp. 29–33, Sep. 1997.

[7] J. Cramwinckel, E. Roijackers, R. Baart, E. Minkes, L. Ruscio, D. Joyner, and C. Tjhai, "GUAVA, a GAP package for computing with error-correcting codes." Available at `http://www.gap-system.org/Packages/guava.html`.

[8] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms for Computers and Calculators*. Academic Press, London, 2nd ed., 1978.

[9] D. E. Knuth, *The Art of Computer Programming, Vol. 4: Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley, 3rd ed., 2005. ISBN 0 201 85394 9.

[10] W. H. Payne and F. M. Ives, "Combination generators," *ACM Transactions on Mathematical Software*, vol. 5, pp. 163–172, Jun. 1979.

SOME RESULTS ON THE WEIGHT DISTRIBUTIONS OF THE BINARY DOUBLE-CIRCULANT CODES BASED ON PRIMES

C. Tjhai, M. Tomlinson, R. Horan, M. Ahmed and M. Ambroze

Fixed and Mobile Communications Research, University of Plymouth, Plymouth PL4 8AA, UK
email:{ctjhai,mtomlinson,rhoran,mahmed,mambroze}@plymouth.ac.uk

ABSTRACT

This paper presents a more efficient algorithm to count codewords of given weights in self-dual double-circulant and formally self-dual quadratic double-circulant codes over GF(2). A method of deducing the modular congruence of the weight distributions of the binary quadratic double-circulant codes is proposed. This method is based on that proposed by Mykkeltveit, Lam and McEliece, *JPL Tech. Rep.*, 1972, which was applied to the extended quadratic-residue codes. A useful application of this modular congruence method is to provide independent verification of the weight distributions of the extended quadratic-residue and quadratic double-circulant codes. Using this method in conjunction with the proposed efficient codeword counting algorithm, we are able i) to give the previously unpublished weight distributions of the $[76, 38, 12]$ and $[124, 62, 20]$ binary quadratic double-circulant codes; ii) to provide corrections to the published results on the weight distributions of the binary extended quadratic-residue code of prime 151, and the number of codewords of weights 30 and 32 of the binary extended quadratic-residue code of prime 137; and iii) to prove that the $[168, 84, 24]$ extended quadratic-residue and quadratic double-circulant codes are inequivalent.

1. INTRODUCTION

Binary self-dual codes form an important class of codes due to their powerful error-correcting capabilities and their rich mathematical structure. As such, this family of codes has been a subject of extensive research for many years. Much of this work is on their classification and the search for the extremal codes [1]. Many binary self-dual codes are codes with the highest known minimum distance. Recently, van Dijk *et al.* [2], constructed two inequivalent binary self-dual codes of length 160 that have higher minimum distance than the previously known half-rate codes of that length.

Closely related to the self-dual codes are the double-circulant codes. Many good binary self-dual codes can be constructed in double-circulant form. An interesting family of binary, double-circulant codes, which includes self-dual and formally self-dual codes, is the family of codes based on primes. A classic paper for this family was published by Karlin [3] in which double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8 were considered. Moore's PhD work [4] investigated the class which is congruent to 3 modulo 8, and his work was later extended by Gulliver *et al.* [5] to longer codes. An extensive discussion on these two types of circulant is also given by MacWilliams *et al.* [6]. The prime-based double-circulant codes can also be constructed over non binary fields, e.g. see Pless [7] and Beenker [8] for GF(3), and Gaborit [9] for the generalisation to prime fields. The weight distributions of double-circulant codes based on primes congruent ± 1 modulo 8, of lengths from 74 to 152 (except 138), i.e. binary extended Quadratic Residue (QR) codes, may be found in [10], as well as those based on primes congruent to ± 3 modulo 8, of lengths 108 and 120.

This paper considers the weight distributions of the binary double-circulant codes based on primes, and it is organised as follows. Section 2 introduces the notation and gives a review of double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8. Section 3 presents an improved algorithm to compute the number of codewords of given weight in certain double-circulant codes. In order to count codewords of given weight, this algorithm requires the enumeration of less codewords than a recently published technique [2, 10]. Based on the fact that the extended QR codes are invariant under the projective special linear group, Mykkeltveit *et al.* [11] developed a technique to deduce the modular congruences of the number of codewords of a given weight in these codes. In Section 4, we describe the automorphism group of the family of double circulant codes with primes congruent to ± 3 modulo 8 which contains the projective special linear group. Accordingly, we show that, with some modifications, the modular congruence method of Mykkeltveit is also applicable to these double-circulant codes. Using this method in conjunction with that given in Section 3, we compute the weight distributions of the quadratic double-circulant codes of lengths 76 and 124. In Section 5, we prove that some of the results reported by Gaborit *et al.* [10] on the extended QR code of length 138 and 152 are incorrect, and provide corrections to these results. It has been observed that, for some primes, there exist two double-circulant codes from different constructions which have the same parameters, but it is not known if the two codes are equivalent. Section 6 discusses two such codes of length 168, and using the techniques presented in Section 4, determines that these codes are inequivalent. Section 7 concludes the paper.

2. BACKGROUND AND NOTATION

Let $\mathbb{F}_2^n$ denote the space of vectors of length n with elements in GF(2). A binary linear code is a k-dimensional linear subspace of $\mathbb{F}_2^n$. We denote $[n, k, d]$ as a binary linear code of length n, dimension k and minimum distance d. The weight enumerator function

This work was partly funded by an Overseas Research Students (ORS) award scheme. The high throughput computing resources provided by the PlymGRID team of the University of Plymouth are gratefully acknowledged.

1-4244-0411-8/06/$20.00 ©2006 IEEE.

of a code is defined as $A(z) = \sum_{i=0}^{n} A_i z^i$, where A_i denotes the number of codewords of weight i. If C is a binary linear code, its dual code $C^\perp$ is defined as $C^\perp = \{w \in \mathbb{F}_2^n \mid \sum_{i=0}^{n-1} v_i w_i = 0 \pmod 2,\ \text{for all } v \in C\}$.

A code is called self-dual iff $C = C^\perp$. A self-dual code is called Type II, or *doubly even*, if the weight of all codewords are divisible by 4; otherwise it is called Type I, or *singly even*. A Type II self-dual code has a length that is divisible by 8. A code is called formally self-dual (fsd) if its weight enumerator is equal to that of its dual. If an fsd code contains an odd weight codeword, it is called an *odd* fsd code; otherwise it is an *even* fsd code. Unless otherwise stated, when we refer to an fsd code, we shall mean an even fsd code. A self-dual, or fsd, code is called *extremal* if its minimum distance is the highest possible for the given parameters.

As a class, double-circulant codes are $[n, k, d]$ codes, where $k = n/2$, whose generator matrix G consists of two circulant matrices. A circulant matrix R is a square $m \times m$ matrix in which each row (resp. column) is a cyclic shift of the adjacent row (resp. column). Such a matrix R is completely characterised by a polynomial formed from its first row, $r(x) = \sum_{i=0}^{m-1} r_i x^i$, which is called the *defining polynomial*, and the algebra of polynomials modulo $x^m - 1$ is isomorphic to that of circulants.

Double-circulant codes can be put into two classes, namely *pure*, and *bordered double-circulant*, codes, whose generator matrices G_p and G_b are shown in (1) and (2) respectively, where I_k is the k-dimensional identity matrix, and $\alpha \in \{0, 1\}$. For the purpose of this paper, we consider the bordered case only and, unless otherwise stated, we shall assume that the term double-circulant codes refers to (2).

$$G_p = \begin{array}{|c|c|} \hline I_k & R \\ \hline \end{array} \quad (1), \qquad G_b = \begin{array}{|c|c|c|} \hline & 1 \ \dots \ 1 & \alpha \\ \cline{2-3} I_k & R & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \\ \hline \end{array} \quad (2)$$

Two binary linear codes, $\mathscr{A}$ and $\mathscr{B}$, are *equivalent* if there exists a permutation π on the coordinates of the codewords which maps the codewords of $\mathscr{A}$ onto codewords of $\mathscr{B}$. We shall write this as $\mathscr{B} = \pi(\mathscr{A})$. If π transforms C into itself, then we say that π fixes the code, and the set of all permutations of this kind form the automorphism group of C, denoted as $\mathrm{Aut}(C)$. MacWilliams *et al.* [6] gives some conditions on the equivalence of double-circulant codes, which are restated for convenience in the lemma below.

Lemma 1. *Let $\mathscr{A}$ and $\mathscr{B}$ be double-circulant codes with generator matrices $[I|A]$ and $[I|B]$ respectively. Let the polynomials $a(x)$ and $b(x)$ be the defining polynomials of A and B. The codes $\mathscr{A}$ and $\mathscr{B}$ are equivalent if any of the following conditions hold: i) $B = A^T$, or ii) $b(x)$ is the reciprocal of $a(x)$, or iii) $a(x)b(x) = 1 \pmod{x^m - 1}$, or iv) $b(x) = a(x^u)$ where m and u are relatively prime.*

For the purpose of this paper, we call the double-circulant codes based on prime congruent to ± 1 modulo 8 the $[p+1, \frac{1}{2}(p+1), d]$ extended quadratic residue (QR) codes, i.e. $p \equiv \pm 1 \pmod 8$; and, following [9], those based on prime congruent to ± 3 modulo 8 the $[2(p+1), p+1, d]$ quadratic double-circulant codes, i.e. $p \equiv \pm 3 \pmod 8$.

2.1. Extended Quadratic Residue Codes as Double-Circulants

The following is a summary of the extended QR codes as double-circulant codes [3, 6, 12]. Let p be a prime congruent to ± 1 modulo 8 and let Q and N be the sets of quadratic residues and non residues modulo p respectively. Binary QR codes are cyclic codes of length p over GF(2). For a given p, there exists four QR codes: $\mathscr{L}, \mathscr{N}$ which are equivalent and have dimension $\frac{1}{2}(p-1)$, and $\mathscr{L}, \mathscr{N}$ which are equivalent and have dimension $\frac{1}{2}(p+1)$. The $[p+1, \frac{1}{2}(p+1), d]$ extended quadratic residue code, denoted by $\hat{\mathscr{L}}$ (resp. $\hat{\mathscr{N}}$), is obtained by annexing an overall parity check to $\mathscr{L}$ (resp. $\mathscr{N}$). If $p \equiv -1 \pmod 8$, $\hat{\mathscr{L}}$ (resp. $\hat{\mathscr{N}}$) is Type II; otherwise it is fsd.

It is well-known that $\mathrm{Aut}(\hat{\mathscr{L}})$ contains the projective special linear group $\mathrm{PSL}_2(p)$ [6]. If r is a generator of the cyclic group Q then $\sigma : i \to ri \pmod p$ is a member of $\mathrm{PSL}_2(p)$. Given $n \in N$, the cycles of σ can be written as

$$(\infty)(n, nr, nr^2, \dots, nr^t)(1, r, r^2, \dots, r^t)(0) \qquad (3)$$

where $t = \frac{1}{2}(p-3)$. Due to this property, G, the generator matrix of $\hat{\mathscr{L}}$, can be arranged into circulants as shown in (4).

$$G = \begin{array}{c|c|c|c|c|c|c|c|c|} & \infty & n & \dots & nr^t & 1 & \dots & r^t & 0 \\ \hline \infty & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 \\ \hline \beta & 0 & & & & & & & 1 \\ \vdots & \vdots & & L & & & R & & \vdots \\ \beta r^t & 0 & & & & & & & 1 \\ \hline \end{array} \qquad (4)$$

The rows $\beta, \beta r, \dots, \beta r^t$ in the above generator matrix contain $\bar{c}_\beta(x), \bar{c}_{\beta r}(x), \dots, \bar{c}_{\beta r^t}(x)$, where $\bar{c}_i(x) = x^i e(x)$ whose coordinates are arranged in the order of (3). Note that, if $p \equiv 1 \pmod 8$, then $\alpha = 1$, $\beta = n$ and the idempotent $e(x) = \sum_{n \in N} x^n$; otherwise $\alpha = 0$, $\beta = 1$ and $e(x) = 1 + \sum_{n \in N} x^n$. If L is non-singular, (4) can be transformed to (2). For many $\hat{\mathscr{L}}$, L is invertible and Karlin [3] has shown that $p = 73, 97, 127, 137, 241$ are the known cases where the canonical form (2) cannot be obtained. In addition to form (2), G can also be transformed to (1), and Jenson [12] has shown that, for $7 \le p \le 199$, except $p = 89, 167$, the canonical form (1) exists.

2.2. Quadratic Double-Circulant Codes

Let p be a prime that is congruent to ± 3 modulo 8. A binary $[2(p+1), p+1, d]$ quadratic double-circulant code, denoted by $\mathscr{B}$, can be constructed using the following defining polynomials

$$b(x) = \begin{cases} 1 + \sum_{r \in Q} x^r & \text{if } p \equiv 3 \pmod 8, \text{ and} \\ \sum_{r \in Q} x^r & \text{if } p \equiv -3 \pmod 8. \end{cases} \qquad (5)$$

The generator matrix G of $\mathscr{B}$ can be written as follows [6]

$$G = \begin{array}{|c|c|c|c|c|c|c|c|} \hline l_\infty & l_0 & \dots & l_{p-1} & r_\infty & r_0 & \dots & r_{p-1} \\ \hline 1 & & & & 0 & & & \\ \vdots & & I_p & & \vdots & & B & \\ 1 & & & & 0 & & & \\ \hline 0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 \\ \hline \end{array} \qquad (6)$$

which is equivalent to (2) with $\alpha = 0$ and $k = p+1$. If $p \equiv 3 \pmod 8$, $\mathscr{B}$ is Type II; otherwise it is fsd with $B = B^T$. Codes of the form $\mathscr{B}$ form an interesting family of double-circulant codes. In terms of self-dual codes, the family contains the largest extremal Type II code known, $n = 136$.

3. AN IMPROVED ALGORITHM TO COUNT CODEWORDS OF GIVEN WEIGHT FOR DOUBLE-CIRCULANT CODES

An algorithm to count codewords of given weight in half-rate codes, which have two full rank disjoint information sets is described in [2] and [10]. This algorithm requires the enumeration of $\binom{k}{w/2} + 2\cdot\sum_{i=1}^{w/2-1}\binom{k}{i}$ codewords in counting all codewords of weight w. We show in the following that, for self-dual double-circulant and fsd quadratic double-circulant codes, it is sufficient to enumerate $\sum_{i=1}^{w/2}\binom{k}{i}$ codewords only.

Let C_1 and C_2 be a self-dual double-circulant, and a fsd quadratic double-circulant codes respectively, which has defining polynomial $r(x)$. We assume that $\mathrm{wt}(f(x))$ represents the weight of the polynomial $f(x)$ and $T_m(x)$ is a set of binary polynomials with degree at most m.

Lemma 2. *Let $u(x), v(x) \in T_{k-1}(x)$, where $wt(u(x)) = i$, and $a(x), b(x) \in T_{k-2}(x)$. The number of weight w codewords of the forms $c(x) = [u(x)|v(x)]$ and $c'(x) = [v'(x)|u'(x)]$ is equal, where i) $u'(x) = u(x)^T$, $v'(x) = v(x)^T$, in the case of C_1; and ii) $u(x) = [\epsilon|a(x)]$, $v(x) = [\gamma|b(x)]$, $u'(x) = [\epsilon|a(x)^2]$, $v'(x) = [\gamma|b(x)^2]$ and $\gamma = wt(a(x))$ (mod 2), in the case of C_2.*

Proof. i) The proof is immediate from the self-dual property. ii) For C_2, we have $(x+1)|r(x)$, $r(x)^3 = 1 + j(x)$, $d(x)(1+j(x)) = d(x)$ if $\mathrm{wt}(d(x))$ is even, otherwise $d(x)(1+j(x)) = d(x) + j(x)$, where $d(x) \in T_{k-2}(x)$ and $j(x)$ is an *all-ones* polynomial. If $\mathrm{wt}(a(x))$ is odd and $\epsilon = 0$, we have $c(x) = [0|a(x)|1|b(x)]$ where $b(x) = a(x)r(x)$. It follows that $b(x)^2 r(x) = a(x)^2 r(x)^3 = a(x)^2 + j(x)$, which has even weight. By adding the first row of (2), we arrive at $c'(x) = [1|b(x)^2|0|a(x)^2]$, which has the same weight as that of $c(x)$ since the weight of a polynomial and its squared is the same. If $\mathrm{wt}(a(x))$ is even and $\epsilon = 1$, we have $c(x) = [1|a(x)|0|b(x) + j(x)]$ and it follows that $(b(x) + j(x))^2 r(x) = a(x)^2$ since $j(x)^2 r(x) = 0$. We know that $\mathrm{wt}((b(x) + j(x))^2)$ is odd and hence, $c'(x) = [0|(b(x) + j(x))^2|1|a(x)^2]$ which has the same weight as that of $c(x)$. The same result holds for the remaining cases of $\mathrm{wt}(a(x))$ and ϵ. Given $c(x)$ there exists $c'(x)$ of the same weight, thus the number of codewords of the forms $c(x)$ and $c'(x)$ is equal. □

From Lemma 2, it follows that, in order to count codewords of weight w, the enumeration of $\sum_{i=1}^{w/2}\binom{k}{i}$ codewords only is required and

$$A_w = a_{w/2} + 2\sum_{i=1}^{w/2-1} a_i, \tag{7}$$

where a_i is the number of weight w codewords which have i non zeros in the first k coordinates.

4. NUMBER OF CODEWORDS OF GIVEN WEIGHTS IN QUADRATIC DOUBLE-CIRCULANT CODES

Let Q and N be the sets of quadratic residue and non residue modulo p respectively. The linear group $\mathrm{PSL}_2(p)$ is generated by the set of all permutations to the coordinates $(\infty, 0, 1, \ldots, p-1)$ of the form $y \to (ay + b)/(cy + d)$ where $a, b, c, d \in \mathrm{GF}(p)$, $y \in \mathrm{GF}(p) \cup \{\infty\}$ and $ad - bc = 1$. It can be shown that this form of permutation is generated by $S : y \to y + 1$, $V : y \to \rho^2 y$ and $T : y \to -\frac{1}{y}$ transformations, where ρ is a primitive element of $\mathrm{GF}(p)$. In fact, V is redundant, since $V = TS^{\mu}TS^{\mu}TS^{\rho}$, where $\mu \in \mathrm{GF}(p)$ and $(\mu, \rho) = 1$. Consider the coordinates $(\infty, 0, 1, \ldots, p-1)$, the transformation S leaves the coordinate ∞ invariant and introduces a cyclic shift to the rest of the coordinates. Let R_i and L_i denote the ith row of the right and left circulants of (6) respectively, J and J' denote the last row of the right and left circulant of (6) respectively. Using the arguments in [6], it can be shown that $T(R_0) = R_0 + J$, $T(R_s) = R_{-1/s} + R_0$, $T(R_t) = R_{-1/t} + R_0 + J$, $T(L_0) = L_0 + J'$, $T(L_s) = L_{-1/s} + L_0$ and $T(L_t) = L_{-1/t} + L_0 + J'$ for $p \equiv 3 \pmod 8$, and $T(R_0) = R_0$, $T(R_s) = R_{-1/s} + J$, $T(R_t) = R_{-1/t} + R_0$, $T(L_0) = L_0$, $T(L_s) = L_{-1/s} + J'$ and $T(L_t) = L_{-1/t} + L_0$ for $p \equiv -3 \pmod 8$, where $s \in Q$ and $t \in N$. This establishes the following theorem on $\mathrm{Aut}(\mathscr{B})$ [6,9].

Theorem 1. *The automorphism group of the $[2(p+1), p+1, d]$ binary quadratic double-circulant codes contains $PSL_2(p)$ applied simultaneously to both circulants.*

The knowledge of $\mathrm{Aut}(\mathscr{B})$ can be exploited to deduce the modular congruence of A_i of $\mathscr{B}$. If $\mathcal{H} \subseteq \mathrm{Aut}(\mathscr{B})$, then A_i of $\mathscr{B}$ can be categorised into two classes: one which contains all weight i codewords that are invariant under $\mathcal{H}$ and the other which contains the rest. The latter class forms orbits of size $|\mathcal{H}|$, the order of $\mathcal{H}$.

For $\mathscr{B}$, we shall choose $\mathcal{H} = \mathrm{PSL}_2(p)$. Each A_i of $\mathscr{B}$ can be written as $A_i \equiv n_i \cdot |\mathcal{H}| + A_i(\mathcal{H})$, where $|\mathcal{H}| = \frac{1}{2}p(p^2 - 1)$ and $A_i(\mathcal{H})$ is the number of weight i codewords fixed by some elements of $\mathcal{H}$. We can factorise the order of $\mathcal{H}$ into product of distinct primes, i.e. $|\mathcal{H}| = \prod_j q_j^{e_j}$ where q_j are distinct primes. As a result, $A_i(\mathcal{H}) \pmod{|\mathcal{H}|}$ can be obtained by applying the Chinese-Remainder-Theorem to $A_i(S_{q_j}) \pmod{q_j^{e_j}}$ for all q_j that divides $|\mathcal{H}|$, where S_{q_j} is the Sylow-q_j-subgroup of $\mathcal{H}$. In order to compute $A_i(S_{q_j})$, we can find the subcode of C that is fixed by S_{q_j}, and compute the number of codewords of weight i in this subcode.

In order to obtain the subcode fixed by S_{q_j} for all primes q_j that divide $|\mathcal{H}|$, the following method can be used. Let c_{l_i} (resp. c_{r_i}) and $c_{l_{i'}}$ (resp. $c_{r_{i'}}$) denote the ith coordinate and ith permuted coordinate, with the respect to the permutation Z_{q_j}, in the left (resp. right) circulant form respectively. The invariant subcode can be obtained by solving a set of linear equations consisting of the parity-check matrix of $\mathscr{B}$, $c_{l_i} + c_{l_{i'}} = 0$ and $c_{r_i} + c_{r_{i'}} = 0$ for all $i \in \mathrm{GF}(p) \cup \{\infty\}$. The solution is a matrix of rank $r > (p+1)$, which is the parity-check matrix of the $[2(p+1), 2(p+1) - r, d']$ invariant subcode.

Following [6], we represent an element of $\mathrm{PSL}_2(p)$ by a 2×2 matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, where $a, b, c, d \in \mathrm{GF}(p)$ and $ad - bc = 1$. For each odd prime q_j, S_{q_j} is a cyclic group which can be generated by some $Z_{q_j} = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in \mathrm{PSL}_2(p)$ of order q_j. Because S_{q_j} is cyclic, it is straightforward to obtain the invariant subcode, from which we can to compute $A_i(S_{q_j})$.

On the other hand, the case of $q_j = 2$ is more complicated. For $q_j = 2$, S_2 is a dihedral group of order 2^{m+1}, where $m + 1$ is the maximum power of 2 that divides $|\mathcal{H}|$. According to Burnside [13], for $p \equiv \pm 3 \pmod 8$, the highest power of 2 that divides $|\mathcal{H}|$ is 2^2 and hence, $m = 1$. Accordingly, there are 3 subgroups of order 2 in S_2, namely $H_2 = \{1, P\}$, $G_2^0\{1, T\}$ and $G_2^1 = \{1, PT\}$ where $P, T \in \mathrm{PSL}_2(p)$, $P^2 = T^2 = 1$ and $TPT^{-1} = P^{-1}$. Let $T = \left[\begin{smallmatrix} 0 & 1 \\ p-1 & 0 \end{smallmatrix}\right]$, which has order 2. It can be

shown that any order 2 permutation, $P = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, has $b = c$ and $a, d \in \mathrm{GF}(p)$ such that $ad - bc = 1$.

Apart from subgroups of order 2, S_2 also contains a non cyclic subgroup of order 4, denoted by G_4. This subgroup contains, apart from an identity, three permutations of order 2 [13], i.e. a Klein 4 group, $G_4 = \{1, P, T, PT\}$.

Following [11], it can be shown that, in order to compute $A_i(S_2)$, it is only necessary to consider the three subgroups of order 2 and G_4. However, all the three subgroups of order 2 are conjugate in $\mathrm{PSL}_2(p)$ and therefore, the subcodes fixed by G_2^0, G_2^1 and H_2 have identical weight distributions and consider either one of them, say G_2^0, is sufficient. Thus, the number of codewords of weight i in the subcodes fixed by S_2 is given by

$$A_i(S_2) \equiv 3A_i(G_2^0) - 2A_i(G_4) \pmod 4. \tag{8}$$

In summary, in order to deduce the modular congruence of the number of weight i codewords in $\mathscr{H}$, it is sufficient to compute the number of weight i codewords in the subcodes fixed by G_2^0, G_4 and Z_q for all odd primes q that divide $|\mathcal{H}|$. The result follows by applying the Chinese-Remainder-Theorem to the number of weight i codewords in the subcodes.

As examples, we consider the previously unknown weight distributions of the $[76, 38, 12]$ and $[124, 62, 20]$ fsd quadratic double-circulant codes. The weight enumerator of an fsd code is given by Gleason's theorem [1]

$$A(z) = \sum_{i=0}^{\lfloor \frac{n}{8} \rfloor} K_i (1 + z^2)^{\frac{n}{2} - 4i} (z^2 - 2z^4 + z^6)^i \tag{9}$$

for integers K_i. Hence, in order to compute $A(z)$, A_{2i} for $6 \le i \le 9$ and $10 \le i \le 15$ have to be computed for the $[76, 38, 12]$ and $[124, 62, 20]$ codes respectively.

In the case of the $[76, 38, 12]$ code, $p = 37$ and $|\mathrm{PSL}_2(37)| = 2^2 \cdot 3^2 \cdot 19 \cdot 37 = 25308$, and for the $[124, 62, 20]$ code, $p = 61$ and $|\mathrm{PSL}_2(61)| = 2^2 \cdot 3 \cdot 5 \cdot 31 \cdot 61 = 113460$. The elements of $\mathrm{PSL}_2(p)$ which generate the required permutations are as follows $P = \left[\begin{smallmatrix} 3 & 8 \\ 8 & 34 \end{smallmatrix}\right]$, $T = \left[\begin{smallmatrix} 0 & 1 \\ 36 & 0 \end{smallmatrix}\right]$, $Z_3 = \left[\begin{smallmatrix} 0 & 1 \\ 36 & 1 \end{smallmatrix}\right]$, $Z_{19} = \left[\begin{smallmatrix} 0 & 1 \\ 36 & 3 \end{smallmatrix}\right]$ and $Z_{37} = \left[\begin{smallmatrix} 0 & 1 \\ 36 & 35 \end{smallmatrix}\right]$ for $p = 37$, and $P = \left[\begin{smallmatrix} 2 & 19 \\ 19 & 59 \end{smallmatrix}\right]$, $T = \left[\begin{smallmatrix} 0 & 1 \\ 60 & 0 \end{smallmatrix}\right]$, $Z_3 = \left[\begin{smallmatrix} 0 & 1 \\ 60 & 1 \end{smallmatrix}\right]$, $Z_5 = \left[\begin{smallmatrix} 0 & 1 \\ 60 & 17 \end{smallmatrix}\right]$, $Z_{31} = \left[\begin{smallmatrix} 0 & 1 \\ 60 & 5 \end{smallmatrix}\right]$, and $Z_{61} = \left[\begin{smallmatrix} 0 & 1 \\ 60 & 59 \end{smallmatrix}\right]$ for $p = 61$. The number of weight i codewords in various subcodes of dimension k, which are fixed by the $\mathrm{PSL}_2(p)$ permutations are

	G_2^0	G_4	S_3	S_{19}	S_{37}
k	20	12	14	2	2
A_{12}	21	3	3	0	0
A_{14}	0	0	0	0	0
A_{16}	153	11	24	0	0
A_{18}	744	20	54	0	0

and

	G_2^0	G_4	S_3	S_5	S_{31}	S_{61}
k	32	18	22	14	2	2
A_{20}	208	12	30	3	0	0
A_{22}	400	12	10	0	0	0
A_{24}	1930	36	50	0	0	0
A_{26}	8180	40	200	24	0	0
A_{28}	26430	140	620	48	0	0
A_{30}	84936	176	960	6	0	0

for $p = 37$ and 61 respectively. Applying the Chinese-Remainder-Theorem, we have

$$\begin{aligned} A_{12} &= n_{12} \cdot 25308 + 2109 & A_{14} &= n_{14} \cdot 25308 \\ A_{16} &= n_{16} \cdot 25308 + 10545 & A_{18} &= n_{18} \cdot 25308 \end{aligned} \tag{10}$$

for the $[76, 38, 12]$ code, and

$$\begin{aligned} A_{20} &= n_{20} \cdot 113460 + 90768 & A_{22} &= n_{22} \cdot 113460 + 75640 \\ A_{24} &= n_{24} \cdot 113460 + 94550 & A_{26} &= n_{26} \cdot 113460 + 83204 \\ A_{28} &= n_{28} \cdot 113460 + 71858 & A_{30} &= n_{30} \cdot 113460 + 68076 \end{aligned} \tag{11}$$

for the $[124, 62, 20]$ code, where n_i are non negative integers.

Using the algorithm in Section 3, we count codewords of the weights required by Gleason's theorem and then derive the weight distributions of the codes, which are shown as follows, using (9). Since the weight distributions are symmetrical with $A_i = A_{n-i}$, only half terms are given.

$[76, 38, 12]$							
i	A_i	i	A_i	i	A_i	i	A_i
0	1	20	7489059	28	3626095793	36	45200010670
12	2109	22	53574224	30	9404812736	38	50157375456
16	86469	24	275509215	32	19610283420		
18	961704	26	1113906312	34	33067534032		

$[124, 62, 20]$			
i	A_i	i	A_i
0	1	42	977979841051968
20	90768	44	3433274842143012
22	529480	46	10482288501057056
24	10873250	48	27906300869721380
26	171180884	50	64924782329852000
28	2159102198	52	132248827882614296
30	22668808776	54	236218089014480048
32	199576556020	56	370432817595720572
34	1489045613508	58	510493584341738312
36	9466389337938	60	618649064180799821
38	51549138453256	62	659543439108163376
40	241551099887720		

Comparing the above A_i with (10) and (11), we can immediately see that $n_{12} = 0$, $n_{14} = 0$, $n_{16} = 3$ and $n_{18} = 38$ for $[76, 38, 12]$ code; $n_{20} = 0$, $n_{22} = 4$, $n_{24} = 95$, $n_{26} = 1508$, $n_{28} = 19029$ and $n_{30} = 199795$ for $[124, 62, 20]$ code. Clearly (10) and (11) provide an independent check on the accuracy of the weight distributions.

5. CORRECTIONS TO THE WEIGHT DISTRIBUTIONS OF THE EXTENDED QUADRATIC-RESIDUE CODES

In this section, we demonstrate the importance of the modular congruence method in providing independent verification to the number of codewords of given weights enumerated exhaustively. We consider two cases of $[p+1, \frac{1}{2}(p+1), d]$ code $\hat{\mathscr{L}}$, namely $p = 137$ and $p = 151$. Note that, in the case of $\hat{\mathscr{L}}$, the method originally proposed in [11] is used.

5.1. Extended Quadratic-Residue Code of Prime 137

Gaborit *et al.* gave A_{2i}, for $22 \le 2i \le 32$, of $\hat{\mathscr{L}}$ for $p = 137$ in [10] and we will check the consistency of these results. For $[p+1, \frac{1}{2}(p+1), d]$ code $\hat{\mathscr{L}}$, we know that $\mathrm{PSL}_2(p) \subseteq \mathrm{Aut}(\hat{\mathscr{L}})$ and for $p = 137$, we have $|\mathrm{PSL}_2(p)| = 2^3 \cdot 3 \cdot 17 \cdot 23 \cdot 137 = 1285608$ and we need to compute $A_{2i}(S_q)$, where $22 \le 2i \le 32$, for all primes q dividing $|\mathrm{PSL}_2(p)|$. Let $P = \left[\begin{smallmatrix} 0 & 37 \\ 37 & 31 \end{smallmatrix}\right]$, $T = \left[\begin{smallmatrix} 0 & 1 \\ 136 & 0 \end{smallmatrix}\right]$, $Z_3 = \left[\begin{smallmatrix} 0 & 1 \\ 136 & 1 \end{smallmatrix}\right]$, $Z_{17} = \left[\begin{smallmatrix} 0 & 1 \\ 136 & 6 \end{smallmatrix}\right]$, and $Z_{23} = \left[\begin{smallmatrix} 0 & 1 \\ 136 & 11 \end{smallmatrix}\right]$. It is not necessary to find Z_p as it only fixes the *all zeros* and *all ones* codewords. The number of weight i codewords in various fixed subcodes of dimension k are

	H_2	G_4^0	G_4^1	S_3	S_{17}	S_{23}	S_{137}
k	35	19	18	23	5	3	1
A_{22}	170	6	6	0	0	0	0
A_{24}	612	10	18	46	0	0	0
A_{26}	1666	36	6	0	0	0	0
A_{28}	8194	36	60	0	0	0	0
A_{30}	34816	126	22	943	0	0	0
A_{32}	114563	261	189	0	0	0	0

and from the Chinese-Remainder-Theorem, we know that

$$\begin{aligned} A_{22} &= n_{22}\cdot 1285608+321402 & A_{28} &= n_{28}\cdot 1285608+321402 \\ A_{24} &= n_{24}\cdot 1285608+1071340 & A_{30} &= n_{30}\cdot 1285608+428536 \\ A_{26} &= n_{26}\cdot 1285608+964206 & A_{32} &= n_{32}\cdot 1285608+1124907 \end{aligned} \tag{12}$$

for some integers n_i. Comparing these to the results in [10], we can immediately see that $n_{22}=0$, $n_{24}=1$, $n_{26}=16$, $n_{28}=381$, and both A_{30} and A_{32} were incorrectly reported. By codeword evaluations, we have established that $A_{30}=6648307504$ ($n_{30}=5171$) and $A_{32}=77865259035$ ($n_{32}=60566$) in (12).

5.2. Extended Quadratic-Residue Code of Prime 151

For $\hat{\mathscr{L}}$ of $p=151$, $|\mathrm{PSL}_2(p)| = 2^3\cdot 3\cdot 5^2\cdot 19\cdot 151 = 1721400$ and $P=\left[\begin{smallmatrix}1 & 42\\ 42 & 104\end{smallmatrix}\right]$, $T=\left[\begin{smallmatrix}0 & 1\\ 150 & 0\end{smallmatrix}\right]$, $Z_3=\left[\begin{smallmatrix}0 & 1\\ 150 & 1\end{smallmatrix}\right]$, $Z_5=\left[\begin{smallmatrix}0 & 1\\ 150 & 27\end{smallmatrix}\right]$, and $Z_{19}=\left[\begin{smallmatrix}0 & 1\\ 150 & 8\end{smallmatrix}\right]$. The number of weight i codewords in the various fixed subcodes of dimension k are

	H_2	G_A^0	G_A^1	S_3	S_5	S_{19}	S_{151}
k	38	20	19	26	16	4	1
A_{20}	38	2	0	25	15	0	0
A_{24}	266	4	4	100	0	0	0

and we have

$$A_{20} = n_{20}\cdot 1721400 + 28690 \text{ and}$$
$$A_{24} = n_{24}\cdot 1721400 + 717250.$$

It follows that A_{20} is correctly reported in [10], but A_{24} is incorrectly reported as 717230. Using the method in Section 3, we have established that $A_{20}=28690$ and $A_{24}=717250$. Using Gleason's theorem for Type II codes [1], we give the corrected weight distribution of this code.

[152, 76, 20]			
i	A_i	i	A_i
0	1	48	5429874340932985 50
20	28690	52	9222363801696269658
24	717250	56	9845887293733174 9615
28	164250250	60	6707403255207981 11830
32	39390351505	64	2949674479653615754525
36	5498418962110	68	8446025592483506824150
40	430930711621830	72	1584056476023923823242 0
44	19714914846904500	76	1952736465900669726536 8

6. [168, 84, 24] DOUBLE-CIRCULANT CODES

If $(2p+1)$ is a prime for $p\equiv 3 \pmod 8$, there exists $\hat{\mathscr{L}}$ and $\hat{\mathscr{B}}$ which have the same length $2(p+1)$ and dimension $p+1$, and in some cases, the minimum distances are also the same. Some examples of such codes are the [8, 4, 4], [24, 12, 8] and [168, 84, 24] codes. Using Lemma 1, we found that, for the two former codes, $\hat{\mathscr{L}}$ and $\hat{\mathscr{B}}$ are equivalent. We consider the [168, 84, 24] double-circulant codes in this section. Note that the minimum distance of $\hat{\mathscr{L}}$ was shown to be 24 in [14] and that of $\hat{\mathscr{B}}$ was shown to be ≤ 28 in [5] and our computation confirms that it is 24.

The defining polynomials (in hexadecimal) of $\hat{\mathscr{L}}$ and $\hat{\mathscr{B}}$ are *c6ac71e844bcd0c8ddd3c* and *d978c57f4cc8d015ce164* respectively. Although these two polynomials have the same weight, the resulting double-circulant codes do not satisfy any condition in Lemma 1, indicating that they are likely to be inequivalent. The inequivalence of these codes can be confirmed by deducing their $A(z)$. Using the modular congruence method, it can be shown that

$$\bar{A}_{24} = \bar{n}_{24}\cdot 2328648 - +776216 \text{ and } \hat{A}_{24} = \hat{n}_{24}\cdot 285852,$$

where $\bar{A}_{24}$ and $\hat{A}_{24}$ are A_{24} of $\hat{\mathscr{L}}$ and $\hat{\mathscr{B}}$ respectively. For integers $\bar{n}_{24}, \hat{n}_{24}\geq 0$, $\bar{A}_{24}\neq\hat{A}_{24}$ and thus, they are inequivalent.

7. CONCLUSIONS

We have presented a more efficient method of codeword counting algorithms for self-dual double-circulant and fsd quadratic double-circulant codes in addition to a method to deduce the modular congruent of the weight distributions of the quadratic double-circulant codes. This modular congruence method is derived from the classic technique proposed by Mykkeltveit *et al.* in 1972, which was applied to the extended QR codes. Using this method, we are able to deduce the inequivalence of the [168, 84, 24] extended QR and quadratic double-circulant codes, and also to correct the previously published results on the weight distribution of the [138, 69, 22] and [152, 76, 20] extended QR codes.

8. REFERENCES

[1] E. M. Rains and N. J. A. Sloane, "Self-Dual Codes," in *Handbook of Coding Theory*, Elsevier, North Holland, V.S. Pless and W. C. Huffman edition, 1998.

[2] M. van Dijk, S. Egner, M. Greferath, and A. Wassermann, "On Two Doubly Even Self-Dual Binary Codes of Length 160 and Minimum Weight 24," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 408–411, Jan. 2005.

[3] M. Karlin, "New Binary Coding Results by Circulants," *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 81–92, Jan. 1969.

[4] E. H. Moore, *Double Circulant Codes and Related Algebraic Structures*, Ph.D dissertation, Dartmouth College, USA, 1976.

[5] T. A. Gulliver and N. Senkevitch, "On a Class of Self-Dual Codes Derived from Quadratic Residue," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 701–702, Mar. 1999.

[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory Error-Correcting Codes*, North-Holland, 1977.

[7] V. Pless, "Symmetry Codes over GF(3) and New Five Designs," *J. Combin. Theory Ser. A*, vol. 12, pp. 119–142, 1972.

[8] G. Beenker, "A Note on Extended Quadratic Residue Codes over GF(9) and Their Ternary Images," *IEEE Trans. Inform. Theory*, vol. 30, no. 2, pp. 403–405, Mar. 1984.

[9] P. Gaborit, "Quadratic Double Circulant Codes over Fields," *J. Combin. Theory Ser. A*, vol. 97, pp. 85–107, 2002.

[10] Philippe Gaborit, Carmen-Simona Nedeloaia, and Alfred Wassermann, "On the Weight Enumerators of Duadic and Quadratic Residue Codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 402–407, Jan. 2005.

[11] J. Mykkeltveit, C. Lam, and R. J. McEliece, "On the Weight Enumerators of Quadratic Residue Codes," *JPL Technical Report 32-1526*, vol. XII, pp. 161–166, 1972.

[12] R. Jenson, "A Double Circulant Presentation for Quadratic Residue Codes," *IEEE Trans. Inform. Theory*, vol. 26, no. 2, pp. 223–227, Mar. 1980.

[13] W. Burnside, *Theory of Group of Finite Order*, Reprinted by Dover, New York, 2nd, 1911 edition, 1955.

[14] M. Grassl, "On the Minimum Distance of Some Quadratic Residue Codes," in *Proceedings IEEE Int. Symp. Inform. Theory, 25-30 June 2000*, p. 253.

On the Efficient Codewords Counting Algorithm and the Weight Distributions of the Binary Quadratic Double-Circulant Codes

C. Tjhai[1], M. Tomlinson, R. Horan, M. Ahmed and M. Ambroze
Fixed and Mobile Communications Research,
University of Plymouth, Plymouth, PL4 8AA, United Kingdom
{ctjhai,mtomlinson,rhoran,mahmed,mambroze}@plymouth.ac.uk

Abstract — **An efficient algorithm to count all codewords of given weight and a method to deduce the modular congruence of the weight distributions of the binary quadratic double-circulant codes are presented. Using this algorithm, we give the weight distribution of the quadratic double-circulant code of length 168 and that of the extended quadratic-residue code of the same length. The weight distributions of these two inequivalent codes of length 168, which were previously unknown, are independently verified and proved to be accurate using the modular congruence method.**

I. Introduction

Binary double-circulant codes forms an important class of codes due to their rich mathematical structure and their high minimum distance properties. Closely related to these double-circulant codes are the self-dual and formally self-dual codes. The self-dual property of a code places a tight restrictions to its weight distribution. In terms of deducing the complete weight distribution of a code, these restrictions are of great advantage since the number of codewords of certain weights only are required.

An interesting family of binary double-circulant codes, which includes self-dual and formally self-dual codes, is the family of codes based on primes. A classic paper for this family was published by Karlin [1] in which double-circulant codes based on primes congruent to ± 1 and ± 3 modulo 8 were considered. Moore's PhD work [2] investigated the class which is congruent to 3 modulo 8, and his work was later extended by Gulliver *et al.* [3] to longer codes. The prime-based double-circulant codes can also be constructed over non binary fields, e.g. see Pless [4] and Beenker [5] for GF(3), and Gaborit [6] for the generalisation to prime fields.

This paper considers some methods related to codeword counting and the weight distributions of binary double-circulant codes based primes congruent to ± 3 modulo 8, which following [6], we will refer to as the quadratic double-circulant codes. This paper is organised as follows. After introducing notations and summarising the background of the quadratic double-circulant codes in Section II, a more efficient algorithm to count all codewords of a given weight is presented in Section III. This codeword counting algorithm requires the enumeration of less codewords than a recently published technique [7,8]. Based on the fact that the automorphism group of the quadratic double-circulant codes contains the projective special linear group, we present a method of deducing the modular congruence of the weight distribution of this family of codes in Section IV. For many codes, to obtain the number of codewords of a given weight in reasonable time requires parallel computations. In this case, the enumerations may have to be split into many sub processes and errors in any of the sub processes will render the overall computations to be incorrect. Thus, it is important to have an independent check on the results and the modular congruence method can be used for this purpose. For the extended quadratic residue (QR) codes, we have demonstrated the importance of having an independent check in [9], where we were able to correct some published results. Using the efficient codeword counting algorithm and the modular congruence method, we give the weight distribution of the $[168, 84, 24]$ quadratic double-circulant code in Section V and that of the $[168, 84, 24]$ extended QR codes in Section VI. The weight distributions of these two codes were previously unknown. Finally, conclusions are given in Section VII.

II. Background and Notation

Let F_2^n denote the space of vectors of length n with elements in GF(2). A binary linear code is a k-dimensional linear subspace of F_2^n. We denote $[n, k, d]$ as a binary linear code of length n, dimension k and minimum distance d. The weight enumerator function of a code is defined as $A(z) = \sum_{i=0}^{n} A_i z^i$, where A_i denotes the number of codewords of weight i. If C is a binary linear code, its dual code $C^\perp$ is defined as $C^\perp = \{w \in F_2^n | \sum_{i=0}^{n-1} v_i w_i = 0 \pmod 2, \text{ for all } v \in C\}$.

Two binary linear codes, $\mathscr{A}$ and $\mathscr{B}$, are *equivalent* if there exists a permutation π on the coordinates of the codewords which maps the codewords of $\mathscr{A}$ onto codewords of $\mathscr{B}$. We shall write this as $\mathscr{B} = \pi(\mathscr{A})$. If π transforms C into itself, then we say that π fixes the code, and the set of all permutations of this kind form the automorphism group of C, denoted as $\mathrm{Aut}(C)$.

A code is called self-dual if and only if $C = C^\perp$. A self-dual code is called Type II, or *doubly even*, if all codeword

[1]This work was partially supported by an Overseas Research Students (ORS) award scheme.

weights are divisible by 4; otherwise it is called Type I, or *singly even*. A Type II self-dual code has a length that is divisible by 8. A code C is called formally self-dual (fsd) if $C \neq C^{\perp}$, but its weight enumerator is equal to that of its dual. If an fsd code contains an odd weight codeword, it is called an *odd* fsd code; otherwise it is an *even* fsd code [10]. Unless otherwise stated, when we refer to an fsd code, we shall mean an even fsd code. A self-dual, or fsd, code is called *extremal* if its minimum distance is the highest possible for the given parameters.

As a class, double-circulant codes are $[n, k]$ codes, where $k = n/2$, whose generator matrix G consists of two circulant matrices. A circulant matrix R is a square $m \times m$ matrix in which each row (resp. column) is a cyclic shift of the adjacent row (resp. column). Such a matrix R is completely characterised by a polynomial formed from its first row, $r(x) = \sum_{i=0}^{m-1} r_i x^i$, which is called the *defining polynomial*, and the algebra of polynomials modulo $x^m - 1$ is isomorphic to that of circulants.

Double-circulant codes can be put into two classes, namely *pure*, and *bordered* double-circulant, codes, whose generator matrices G_p and G_b are shown in (1) and (2) respectively, where I_k is the k-dimensional identity matrix, and $\alpha \in \{0, 1\}$. We consider the bordered case only in this paper and, unless otherwise stated, we shall assume that the term double-circulant codes refers to (2).

$$G_p = \begin{array}{|c|c|} \hline I_k & R \\ \hline \end{array} \quad (1), \qquad G_b = \begin{array}{|c|c|c|} \hline & 1 \ \ldots \ 1 & \alpha \\ \hline I_k & R & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \\ \hline \end{array} \quad (2)$$

A. Quadratic Double-Circulant Codes

Let p be a prime that is congruent to ± 3 modulo 8. Let Q and N be sets of quadratic and non quadratic residues modulo p respectively. We assume that the polynomials $q(x) = \sum_{i \in Q} x^i$ and $n(x) = \sum_{i \in N} x^i$. A $[2(p+1), p+1, d]$ binary quadratic double-circulant code, denoted by $\mathscr{B}$, can be constructed using the defining polynomial

$$b(x) = \begin{cases} 1 + q(x) & \text{if } p \equiv 3 \pmod 8, \text{ and} \\ q(x) & \text{if } p \equiv -3 \pmod 8. \end{cases} \quad (3)$$

Following [11], the generator matrix G of $\mathscr{B}$ is

$$G = \begin{array}{|c|ccc|c|ccc|} \hline l_\infty & l_0 & \ldots & l_{p-1} & r_\infty & r_0 & \ldots & r_{p-1} \\ \hline 1 & & & & 0 & & & \\ \vdots & & I_p & & \vdots & & B & \\ 1 & & & & 0 & & & \\ \hline 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 \\ \hline \end{array} \quad (4)$$

which is equivalent to (2) with $\alpha = 0$ and $k = p + 1$. Let $j(x) = 1 + x + x^2 + \ldots + x^{p-1}$, the following are some properties of $\mathscr{B}$ [1]:

1. since $(x+1) | b(x)$, $\mathrm{wt}(b(x))$, i.e. the weight of $b(x)$, is even and hence, $b(x)$ does not have inverse, but $(b(x) + j(x))$ does;
2. for $p \equiv \pm 3 \pmod 8$, $2 \in N$ and we have $q(x)^2 = n(x)$, $b(x)^3 = 1 + j(x)$ and $(b(x) + j(x))^3 = 1$;
3. for $p \equiv 3 \pmod 8$, $-1 \in N$ and we have $b(x)^T = 1 + n(x)$ and thus, $\mathscr{B}$ is Type-II by property 2; and
4. for $p \equiv -3 \pmod 8$, $-1 \in Q$ and we have $b(x)^T = b(x)$ and it follows from property 2 that $\mathscr{B}$ is fsd.

Codes of the form $\mathscr{B}$ form an interesting family of double-circulant codes. In terms of self-dual codes, the family contains the longest extremal Type II code known, $n = 136$. Moreover, $\mathscr{B}$ is the binary image of the extended QR code over GF(4) [12].

It is well-known that the automorphism group of the $[(p+1), \frac{1}{2}(p+1), d]$ binary extended quadratic-residue codes contains the projective special group, $\mathrm{PSL}_2(p)$ [11]. This linear group $\mathrm{PSL}_2(p)$ is generated by the set of all permutations to the coordinates $(\infty, 0, 1, \ldots, p-1)$ of the form $y \to (ay + b)/(cy + d)$ where $a, b, c, d \in \mathrm{GF}(p)$, $y \in \mathrm{GF}(p) \cup \{\infty\}$ and $ad - bc = 1$. It can be shown that this form of permutation is generated by following transformations

$$S : y \to y + 1, \quad V : y \to \rho^2 y, \quad \text{and} \quad T : y \to -\frac{1}{y},$$

where ρ is a primitive element of GF(p). In fact, V is redundant, since $V = TS^{\rho}TS^{\mu}TS^{\rho}$, where $\mu \in \mathrm{GF}(p)$ and $(\mu, \rho) = 1$. Consider the coordinates $(\infty, 0, 1, \ldots, p-1)$, the transformation S leaves the coordinate ∞ invariant and introduces a cyclic shift to the rest of the coordinates. Let R_i and L_i denote the ith row of the right and left circulants of (4) respectively, J and J' denote the last row of the right and left circulant of (4) respectively. Using the arguments in [11], it can be shown that the effects of T to the circulants are as follows:

T	for $p \equiv 3 \pmod 8$	for $p \equiv -3 \pmod 8$
$T(R_0)$	$R_0 + J$	R_0
$T(R_s)$	$R_{-1/s} + R_0$	$R_{-1/s} + J$
$T(R_t)$	$R_{-1/t} + R_0 + J$	$R_{-1/t} + R_0$
$T(L_0)$	$L_0 + J'$	L_0
$T(L_s)$	$L_{-1/s} + L_0$	$L_{-1/s} + J'$
$T(L_t)$	$L_{-1/t} + L_0 + J'$	$L_{-1/t} + L_0$

where $s \in Q$ and $t \in N$. This establishes the following theorem on $\mathrm{Aut}(\mathscr{B})$ [11][6].

Theorem 1. *The automorphism group of the $[2(p+1), p+1, d]$ binary quadratic double-circulant codes contains $PSL_2(p)$ applied simultaneously to both circulants.*

The knowledge of $\mathrm{Aut}(\mathscr{B})$ can be exploited to deduce the modular congruence of A_i of $\mathscr{B}$ and this is shown in Section IV.

III. A More Efficient Algorithm To Count Codewords of Given Weights

An algorithm to count codewords of given weight in half-rate codes, which have two full rank disjoint information sets is described in [8] and [7]. This algorithm requires the enumeration of $\binom{k}{w/2} + 2 \cdot \sum_{i=1}^{w/2-1} \binom{k}{i}$ codewords in counting all codewords of weight w. We show in

the following that, for self-dual double-circulant and fsd quadratic double-circulant codes, it is sufficient to enumerate $\sum_{i=1}^{w/2}\binom{k}{i}$ codewords only.

Let C_1 and C_2 be a self-dual double-circulant code, and a fsd quadratic double-circulant code respectively. The set of binary polynomials with degree at most m is denoted by $T_m(x)$.

Lemma 1. *Let $u(x),v(x) \in T_{k-1}(x)$, where $wt(u(x)) = i$, and $a(x),b(x) \in T_{k-2}(x)$. The number of weight w codewords of the forms $c(x) = [u(x)|v(x)]$ and $c'(x) = [v'(x)|u'(x)]$ is equal, where i) $u'(x) = u(x)^T$, $v'(x) = v(x)^T$, in the case of C_1; and ii) $u(x) = [\epsilon|a(x)]$, $v(x) = [\gamma|b(x)]$, $u'(x) = [\epsilon|a(x)^2]$, $v'(x) = [\gamma|b(x)^2]$ and $\gamma = wt(a(x))$ (mod 2), in the case of C_2.*

Proof. i) The proof is immediate from the self-dual property. ii) For C_2, we have $d(x)(1+j(x)) = d(x)$ if $\mathrm{wt}(d(x))$ is even, otherwise $d(x)(1+j(x)) = d(x)+j(x)$, where $d(x) \in T_{k-2}(x)$. If $\mathrm{wt}(a(x))$ is odd and $\epsilon = 0$, we have $c(x) = [0|a(x)|1|b(x)]$ where $b(x) = a(x)r(x)$. It follows that $b(x)^2r(x) = a(x)^2r(x)^3 = a(x)^2+j(x)$, which has even weight. By adding the first row of (2), we arrive at $c'(x) = [1|b(x)^2|0|a(x)^2]$, which has the same weight as that of $c(x)$ since the weight of a polynomial and its square is the same. If $\mathrm{wt}(a(x))$ is even and $\epsilon = 1$, we have $c(x) = [1|a(x)|0|b(x)+j(x)]$ and it follows that $(b(x)+j(x))^2r(x) = a(x)^2$ since $j(x)^2r(x) = 0$. We know that $\mathrm{wt}((b(x)+j(x))^2)$ is odd and hence, $c'(x) = [0|(b(x)+j(x))^2|1|a(x)^2]$ which has the same weight as that of $c(x)$. The same result holds for the remaining cases of $\mathrm{wt}(a(x))$ and ϵ. Given $c(x)$ there exists $c'(x)$ of the same weight, thus the number of codewords of the forms $c(x)$ and $c'(x)$ is equal. □

From Lemma 1, it follows that, in order to count codewords of weight w, the enumeration of $\sum_{i=1}^{w/2}\binom{k}{i}$ codewords only is required and

$$A_w = a_{w/2} + 2\sum_{i=1}^{w/2-1} a_i, \tag{5}$$

where a_i is the number of weight w codewords which have i non zeros in the first k coordinates.

IV. Number of Codewords of Given Weight in Quadratic Double-Circulant Codes

If $\mathcal{H} \subseteq \mathrm{Aut}(\mathscr{B})$, then A_i of $\mathscr{B}$ can be categorised into two classes: one which contains all weight i codewords that are invariant under $\mathcal{H}$ and the other which contains the rest. The latter class forms orbits of size $|\mathcal{H}|$, the order of $\mathcal{H}$ [13][11, pp. 505].

For $\mathscr{B}$, we shall choose $\mathcal{H} = \mathrm{PSL}_2(p)$, which has order $\frac{1}{2}p(p^2-1)$. Each A_i of $\mathscr{B}$ can be written as $A_i = n_i \cdot |\mathcal{H}| + A_i(\mathcal{H})$, where $A_i(\mathcal{H})$ is the number of weight i codewords in a subcode fixed by some elements of $\mathcal{H}$ and n_i is a non negative integer. Since $|\mathcal{H}| = \prod_j q_j^{e_j}$, where q_j are distinct primes, $A_i(\mathcal{H})$ (mod $|\mathcal{H}|$) can be obtained by applying the Chinese-Remainder-Theorem to $A_i(S_{q_j})$ (mod $q_j^{e_j}$) for all q_j that divides $|\mathcal{H}|$, where S_{q_j} is the Sylow-q_j-subgroup of $\mathcal{H}$. In order to compute $A_i(S_{q_j})$, we simply find the subcode of C that is fixed by S_{q_j}, and compute the number of codewords of weight i in this subcode.

In order to obtain the subcode fixed by S_{q_j} for all primes q_j that divide $|\mathcal{H}|$, the following method can be used. Let c_{l_i} (resp. c_{r_i}) and $c_{l_{i'}}$ (resp. $c_{r_{i'}}$) denote the ith coordinate and ith permuted coordinate, with respect to the permutation Z_{q_j}, in the left (resp. right) circulant form respectively. The invariant subcode can be obtained by solving a set of linear equations consisting of the parity-check matrix of $\mathscr{B}$, $c_{l_i} + c_{l_{i'}} = 0$ and $c_{r_i} + c_{r_{i'}} = 0$ for all $i \in \mathrm{GF}(p) \cup \{\infty\}$. The solution is a matrix of rank $r > (p+1)$, which is the parity-check matrix of the $[2(p+1), 2(p+1)-r]$ invariant subcode.

Following [11], we represent an element of $\mathrm{PSL}_2(p)$ by a 2×2 matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, where $a,b,c,d \in \mathrm{GF}(p)$ and $ad-bc = 1$. For each odd prime q_j, S_{q_j} is a cyclic group which can be generated by some $Z_{q_j} = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in \mathrm{PSL}_2(p)$ of order q_j. Because S_{q_j} is cyclic, it is straightforward to obtain the invariant subcode, from which we can compute $A_i(S_{q_j})$.

The case of $q_j = 2$ is more complicated. For $q_j = 2$, S_2 is a dihedral group of order 2^{m+1}, where $m+1$ is the maximum power of 2 that divides $|\mathcal{H}|$. According to Burnside [14], for $p \equiv \pm 3$ (mod 8), the highest power of 2 that divides $|\mathcal{H}|$ is 2^2 and hence, $m = 1$. Accordingly, there are 3 subgroups of order 2 in S_2, namely $H_2 = \{1, P\}$, $G_2^0\{1, T\}$ and $G_2^1 = \{1, PT\}$ where $P, T \in \mathrm{PSL}_2(p)$, $P^2 = T^2 = 1$ and $TPT^{-1} = P^{-1}$. Let $T = \left[\begin{smallmatrix} 0 & 1 \\ p-1 & 0 \end{smallmatrix}\right]$, which has order 2. It can be shown that any order 2 permutation, $P = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, has $b = c$ and $a, d \in \mathrm{GF}(p)$ such that $ad - bc = 1$.

Apart from subgroups of order 2, S_2 also contains a non cyclic subgroup of order 4, which contains, apart from an identity, three permutations of order 2 [14], i.e. a Klein 4 group, $G_4 = \{1, P, T, PT\}$.

Following [13], it can be shown that, in order to compute $A_i(S_2)$, it is only necessary to consider the three subgroups of order 2 and G_4. However, all the three subgroups of order 2 are conjugate in $\mathrm{PSL}_2(p)$ and therefore, the subcodes fixed by G_2^0,G_2^1 and H_2 have identical weight distributions and considering any of them, say G_2^0, is sufficient. Thus, the number of codewords of weight i in the subcodes fixed by S_2 is

$$A_i(S_2) \equiv 3A_i(G_2^0) - 2A_i(G_4) \pmod 4. \tag{6}$$

In summary, in order to deduce the modular congruence of the number of weight i codewords in $\mathscr{B}$, it is sufficient to compute the number of weight i codewords in the subcodes fixed by H_2, G_4 and Z_q, for all odd primes q that divide $|\mathcal{H}|$, and apply the Chinese-Remainder-Theorem to them. In the following section, we apply this method to the $[168, 84, 24]$ binary quadratic double-circulant code as an example.

V. On the Weight Distribution of the [168, 84, 24] Quadratic Double-Circulant Code

Gulliver et al. [3] has shown that the [168, 84, 24] quadratic double-circulant code is not an extremal self-dual code since it has minimum distance less than or equal to 28. The weight enumerator of a Type II code of length n is given by Gleason's theorem, which is expressed as [10]

$$A(z) = \sum_{i=0}^{\lfloor n/24 \rfloor} K_i (1 + 14z^4 + z^8)^{\frac{n}{8}-3i} \{z^4(1-z^4)^4\}^i \quad (7)$$

where K_i are some integers. As shown in (7), only the first few terms of A_i are required in order to completely determine the weight distributions of Type II codes. For the case of [168, 84, 24] code $\mathscr{B}$, only the first 8 terms of A_i are required. Using a parallel version of the efficient codeword enumeration method described in Section III, we determined that all of these 8 terms are 0 apart from $A_0 = 1$, $A_{24} = 571704$ and $A_{28} = 17008194$.

Although computing the number of codewords of weight i is straightforward, it is desirable to have a method of checking the number of codewords computed. The modular congruence method introduced in the previous section can be used for this purpose. Consider the case of the [168, 84, 24] code $\mathscr{B}$, now $p = 83$ and $|\mathcal{H}| = 2^2 \cdot 3 \cdot 7 \cdot 41 \cdot 83 = 285852$. We will consider the odd prime cases in the first place.

For prime $q = 3$, a cyclic group of order 3 S_3 can be generated by $Z_3 = \left[\begin{smallmatrix} 0 & 1 \\ 82 & 1 \end{smallmatrix}\right]$ and we found that the subcode invariant under S_3 has dimension 28 and has 63 and 0 codewords of weights 24 and 28 respectively.

For prime $q = 7$, we have $Z_7 = \left[\begin{smallmatrix} 0 & 1 \\ 82 & 10 \end{smallmatrix}\right]$ which generates S_7. The subcode fixed by S_7 has dimension 12 and there are no codewords of weights 24 and 28 in this subcode.

Similarly, for prime $q = 41$, the subcode fixed by S_{41}, which is generated by $Z_{41} = \left[\begin{smallmatrix} 0 & 1 \\ 82 & 4 \end{smallmatrix}\right]$ and has dimension 4, contains no codeword of weights 24 and 28.

Finally, for prime $q = 83$, the invariant subcode has only the all-zeros, all-ones, $\{\underbrace{0,0,\ldots,0,0}_{84},\underbrace{1,1,\ldots,1,1}_{84}\}$, and $\{\underbrace{1,1,\ldots,1,1}_{84},\underbrace{0,0,\ldots,0,0}_{84}\}$ codewords. The cyclic group S_{83} is generated by $Z_{83} = \left[\begin{smallmatrix} 0 & 1 \\ 82 & 81 \end{smallmatrix}\right]$.

For the case of $q = 2$, we have $P = \left[\begin{smallmatrix} 1 & 0 \\ 9 & 82 \end{smallmatrix}\right]$ and $T = \left[\begin{smallmatrix} 0 & 1 \\ 82 & 0 \end{smallmatrix}\right]$. The subcode fixed by S_2, which has dimension 42, has 196 and 1050 codewords of weights 24 and 28 respectively. On the other hand, the subcode fixed by G_4, which has dimension 22, has 4 and 6 codewords of weights 24 and 28 respectively.

Thus, using (6), the number of codewords of weights 24 and 28 fixed by S_2 are

$$A_{24}(S_2) = 3 \cdot 196 - 2 \cdot 4 \equiv 0 \pmod 4, \text{ and}$$
$$A_{28}(S_2) = 3 \cdot 1050 - 2 \cdot 6 \equiv 2 \pmod 4$$

and by applying the Chinese-Remainder-Theorem to all $A_i(S_q)$ for $i = 24, 28$, we arrive at

$$A_{24} = n_{24} \cdot 285852 \quad (8)$$
$$A_{28} = n_{28} \cdot 285852 + 142926. \quad (9)$$

From (8) and (9), we have now verified A_{24} and A_{28}, since they have equality for non negative integers n_{24} and n_{28} ($n_{24} = 2$ and $n_{28} = 59$). Using Gleason's theorem, i.e. (7), the weight enumerator of the [168, 84, 24] code $\mathscr{B}$, denoted by $A(z)$, is obtained and is given in (10).

$$\begin{aligned} A(z) = {} & (z^0 + z^{168}) + \\ & 571704 \cdot (z^{24} + z^{144}) + \\ & 17008194 \cdot (z^{28} + z^{140}) + \\ & 5507510484 \cdot (z^{32} + z^{136}) + \\ & 1252615755636 \cdot (z^{36} + z^{132}) + \\ & 166058829151929 \cdot (z^{40} + z^{128}) + \\ & 13047194638256310 \cdot (z^{44} + z^{124}) + \\ & 629048483051034984 \cdot (z^{48} + z^{120}) + \\ & 19087129808556586056 \cdot (z^{52} + z^{116}) + \\ & 372099697089030108600 \cdot (z^{56} + z^{112}) + \\ & 4739291490433882602066 \cdot (z^{60} + z^{108}) + \\ & 39973673426117369814414 \cdot (z^{64} + z^{104}) + \\ & 225696677517789500207052 \cdot (z^{68} + z^{100}) + \\ & 860241109321000217491044 \cdot (z^{72} + z^{96}) + \\ & 2227390682939806465038006 \cdot (z^{76} + z^{92}) + \\ & 3935099587279668544910376 \cdot (z^{80} + z^{88}) + \\ & 4755747411704650343205104 \cdot z^{84} \end{aligned} \quad (10)$$

VI. Comparison to the Extended Quadratic Residue Code of Length 168

There exists an extended QR code, denoted by $\hat{\mathscr{L}}$ for convenience, which is also a [168, 84, 24] code. Since $\hat{\mathscr{L}}$ can be put into double-circulant form and it is Type-II self-dual, the algorithm in Section III can be used to compute the number of codewords of weights 24 and 28, A'_{24} and A'_{28}, from which we can use Gleason's theorem to derive its weight enumerator, $A'(z)$. By evaluations, we determined that $A'_{24} = 776216$ and $A'_{28} = 18130188$.

In order to verify the accuracy of A'_{24} and A'_{28}, the modular congruence method originally described in [13] was used. In this case, we have $\mathrm{Aut}(\hat{\mathscr{L}}) \supseteq \mathcal{H} = \mathrm{PSL}_2(p)$. Now, $p = 167$ and $|\mathrm{PSL}_2(p)| = 2^3 \cdot 3 \cdot 7 \cdot 83 \cdot 167 = 2328648$. We have $P = \left[\begin{smallmatrix} 1 & 41 \\ 41 & 12 \end{smallmatrix}\right]$, $T = \left[\begin{smallmatrix} 0 & 1 \\ 166 & 0 \end{smallmatrix}\right]$, $Z_3 = \left[\begin{smallmatrix} 0 & 1 \\ 166 & 1 \end{smallmatrix}\right]$, $Z_7 = \left[\begin{smallmatrix} 0 & 1 \\ 166 & 19 \end{smallmatrix}\right]$, $Z_{83} = \left[\begin{smallmatrix} 0 & 1 \\ 166 & 4 \end{smallmatrix}\right]$, and $Z_{167} = \left[\begin{smallmatrix} 0 & 1 \\ 166 & 165 \end{smallmatrix}\right]$. The number of codewords of weights 24 and 28 in the various invariant subcodes of dimension k are

	H_2	G_4^0	G_4^1	S_3	S_7	S_{83}	S_{167}
k	42	22	21	28	12	2	1
A_{24}	252	6	4	140	0	0	0
A_{28}	1812	36	0	0	6	0	0

For $\hat{\mathscr{L}}$, (6) becomes [13]

$$A_i(S_2) \equiv (2^m+1)A_i(H_2) - 2^{m-1}A_i(G_4^0) - 2^{m-1}A_i(G_4^1) \pmod{2^{m+1}}, \quad (11)$$

where $m+1$ is the highest power of 2 that divides $|\mathcal{H}|$. It follows that $A_{24}(S_2) \equiv 0 \pmod 8$ and $A_{28}(S_2) \equiv 4 \pmod 8$, and thus

$$A'_{24} = n'_{24} \cdot 2328648 + 776216 \quad (12)$$
$$A'_{28} = n'_{28} \cdot 2328648 + 1829652 \quad (13)$$

from the Chinese-Remainder-Theorem.

Equations (12) and (13) establish that $A'_{24} = 776216$ ($n'_{24} = 0$) and $A'_{28} = 18130188$ ($n'_{28} = 7$). The weight enumerator of $\hat{\mathscr{L}}$ of length 168 is derived from (7) and it is given in (14).

$$\begin{aligned} A'(z) = {} & (z^0 + z^{168}) + \\ & 776216 \cdot (z^{24} + z^{144}) + \\ & 18130188 \cdot (z^{28} + z^{140}) + \\ & 5550332508 \cdot (z^{32} + z^{136}) + \\ & 1251282702264 \cdot (z^{36} + z^{132}) + \\ & 166071600559137 \cdot (z^{40} + z^{128}) + \\ & 13047136918828740 \cdot (z^{44} + z^{124}) + \\ & 629048543890724216 \cdot (z^{48} + z^{120}) + \\ & 19087130695796615088 \cdot (z^{52} + z^{116}) + \\ & 372099690249351071112 \cdot (z^{56} + z^{112}) + \\ & 4739291519495550245228 \cdot (z^{60} + z^{108}) + \\ & 39973673337590380474086 \cdot (z^{64} + z^{104}) + \\ & 225696677727188690570184 \cdot (z^{68} + z^{100}) + \\ & 860241108921860741947676 \cdot (z^{72} + z^{96}) + \\ & 2227390683565491780127428 \cdot (z^{76} + z^{92}) + \\ & 3935099586463594172460648 \cdot (z^{80} + z^{88}) + \\ & 4755747412595715344169376 \cdot z^{84} \end{aligned} \quad (14)$$

In comparison to (10), it may be seen that, for $n = 168$, $\hat{\mathscr{L}}$ is a slightly inferior code than $\mathscr{B}$ having more codewords of weights 24, 28 and 32.

VII. Conclusions

We have presented, in addition to a more efficient codeword counting algorithm, a method to deduce the modular congruence of the number of codewords of given weight for the quadratic double-circulant codes. This modular congruence method is derived from the classic technique proposed by Mykkeltveit *et al.* in 1972, which was applied to the extended QR codes. With this method, the modular congruence of the number of codewords of weight i in a $[2(p+1),(p+1),d]$ binary quadratic double-circulant codes can be deduced by applying the Chinese-Remainder-Theorem to the number of codewords of weight i in the invariant subcodes. Each of these subcodes is fixed by some non identity element of the Sylow-q-subgroup of $\mathrm{PSL}_2(p)$ where q is a prime dividing $|\mathrm{PSL}_2(p)|$. The dimension of the subcodes is generally much smaller than that of the original code and hence, counting the number of codewords of given weights in these subcodes is straightforward.

The beauty of this technique is that it provides an independent, invaluable checking tool for the number of codewords of given weights computed exhaustively. Using this technique in conjunction with the more efficient codeword counting algorithm for double-circulant codes, we give the weight enumerator of the $[168, 84, 24]$ binary quadratic double-circulant and extended quadratic-residue codes, which are inequivalent.

Acknowledgments

The authors wish to thank the PlymGRID team of the University of Plymouth, especially to Michael Hess and Pin Hu for their assistance, in providing high throughput computing resources.

References

[1] M. Karlin, "New Binary Coding Results by Circulants," *IEEE Transactions on Information Theory*, vol. 15, pp. 81–92. Jan. 1969.

[2] E. H. Moore, *Double Circulant Codes and Related Algebraic Structures*. Ph.D dissertation, Dartmouth College. USA, 1976.

[3] T. A. Gulliver and N. Senkevitch, "On a Class of Self-Dual Codes Derived from Quadratic Residue," *IEEE Transactions on Information Theory*, vol. 45, pp. 701–702, Mar. 1999.

[4] V. Pless, "Symmetry Codes over GF(3) and New Five Designs," *J. Combin. Theory Ser. A*, vol. 12, pp. 119–142, 1972.

[5] G. Beenker, "A Note on Extended Quadratic Residue Codes over GF(9) and Their Ternary Images," *IEEE Transactions on Information Theory*, vol. 30, pp. 403–405, Mar. 1984.

[6] P. Gaborit, "Quadratic Double Circulant Codes over Fields," *J. Combin. Theory Ser. A*, vol. 97, pp. 85–107, 2002.

[7] P. Gaborit, C.-S. Nedeloaia, and A. Wassermann, "On the Weight Enumerators of Duadic and Quadratic Residue Codes," *IEEE Transactions on Information Theory*, vol. 51, pp. 402–407, Jan. 2005.

[8] M. van Dijk, S. Egner, M. Greferath, and A. Wassermann, "On Two Doubly Even Self-Dual Binary Codes of Length 160 and Minimum Weight 24," *IEEE Transactions on Information Theory*, vol. 51, pp. 408–411, Jan. 2005.

[9] C. Tjhai, M. Tomlinson, R. Horan, M. Ahmed, and M. Ambroze, "Some Results on the Weight Distributions of the Binary Double-Circulant Codes Based on Primes," in *Proc. 10th IEEE ICCS 2006, 30 Oct.–1 Nov., Singapore*, 2006.

[10] E. M. Rains and N. J. A. Sloane, "Self-Dual Codes," in *Handbook of Coding Theory*, Elsevier, North Holland, V.S. Pless and W. C. Huffman ed., 1998.

[11] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977. ISBN 0 444 85193 3.

[12] M. Karlin, V. K. Bhargava, and S. E. Tavares, "A Note on the Extended Quadratic Residue Codes and Their Binary Images," *Inform. and Control*, vol. 38, pp. 148–153, 1978.

[13] J. Mykkeltveit, C. Lam, and R. J. McEliece, "On the Weight Enumerators of Quadratic Residue Codes," *JPL Technical Report 32-1526*, vol. XII, pp. 161–166, 1972.

[14] W. Burnside, *Theory of Group of Finite Order*. Reprinted by Dover, New York, 2nd, 1911 ed., 1955.

Read Channel Designs for Efficient Data Recovery in Storage Systems Employing Imperfect Patterned Media . .

I. T. Ntokas[1], P. W. Nutter[1], B. K. Middleton[1], C. J. Tjhai[2] and M. Z. Ahmed[2]
1. School of Computer Science, The University of Manchester, Manchester, United Kingdom; 2. School of Computing, Communications and Electronics, The University of Plymouth, Plymouth, United Kingdom

Introduction

The general view is that the current use of a continuous magnetic thin-film for data storage will not be suitable for attaining storage densities in excess of 1Tbit/in^2. As such, new storage technologies, such as the use of a patterned medium, must be explored [1].

However, the development of patterned media as viable storage media is limited by the availability of cost-effective fabrication techniques. In addition, the variation in island size and distribution due to the limitations of current fabrication techniques introduces lithography jitter in the replay signal, which invariably affects the ability to recover recorded data [2]. The data recovery process from a magnetic storage system incorporating patterned media was initially analysed by Hughes [3] and more recently in [4], where the effect of the media configuration on the bit-error-rate (BER) performance of the read channel was explored and the BER was shown to highly dependent upon the amount of lithography jitter present. These results will undoubtedly have a considerable effect on the commercial viability of data storage on patterned magnetic media and will define suitable manufacturing processes for the fabrication of such media.

The aim of this paper is to investigate how careful media design and the use of low density parity check (LDPC) codes, can be used to optimise the read channel BER performance in the presence of lithography jitter, as well as offer increased areal density without sacrificing too much BER performance.

Replay and Channel Simulation

A complete simulation of the data recovery process in a patterned magnetic medium storage system has been developed in order to predict the performance of the read channel in terms of BER against system SNR. The complete simulation, including both replay and read channel models has been described in [4, 5]. The replay model adopted [4] takes into account the geometrical (across-track) aspects of both the patterned recording medium and the GMR read sensor. In this analysis, a GMR head of sensor width 20nm, sensor length (along track) 4nm, and shield-to-shield spacing 16nm has been used. The patterned medium comprises square islands of length 12.5nm with a period and track pitch of 25nm, which translates to an areal density of 1Tbit/in^2. A recording medium of perpendicular anisotropy has been assumed, the patterned film is 10nm thick, and the head-medium separation is 10nm. The channel simulation is described in detail in [5], and consists of either a generalised partial response (GPR) filter and Viterbi decoder, in the case of no LDPC code, or a PR4 target and MAP decoder if an LDPC code is adopted. The LDPC codes used are irregular codes constructed using the progressive-edge-growth (PEG) algorithm [6] of block-length 4096 and code-rate 0.8. Lithography jitter is specified as a percentage of the island period, along-track only.

Results-Discussion

We have demonstrated previously that for the specific media/head configurations adopted, a patterned medium with no SUL offers overall better BER performance compared to a patterned medium employing a SUL [5]. In addition, the performance is improved significantly through the use of hexagonally packed islands [7]. These results show that with no error correction scheme employed, the maximum amount of lithography jitter that can be tolerated is 10% of the island period (2.5nm), whilst still maintaining a BER<10^{-5}. Here we demonstrate the advantage of using LDPC codes in terms of improving the BER performance. Fig. 1 illustrates the BER performance when utilizing irregular LDPC codes, a) for varying lithography jitter and b) for varying island period. Fig. 1a demonstrates that even with 20% lithography jitter (5nm) present, a BER<10^{-5} can be achieved at an SNR of ≈17dB. In addition, in the case of no lithography jitter, Fig. 1b illustrates that a reduction of the island period to 20nm (areal density of 1.6Tbit/in^2) still permits a BER of 10^{-5} at an SNR of ≈10dB and even when the period is 17.5nm (areal density of 2.1Tbit/in^2) a BER of 10^{-4} is obtained at an SNR of ≈17dB. In conclusion, even in the presence of large amounts of lithography jitter, an acceptable BER is observed when utilizing irregular LDPC codes. In addition, the use of LDPC codes permits increased areal densities whilst still maintaining acceptable BER performance.

[1] B. D. Terris & T. Thomson, J. Phys. D: Appl. Phys., vol. 38, pp. R199-R222, 2005.
[2] M. M. Aziz et al., IEEE Trans. Mag., vol. 38, pp. 1964-1966, 2002.
[3] G. F. Hughes, IEEE Trans. Mag., vol. 35, pp. 2310-2312, 1999.
[4] P. W. Nutter et al., IEEE Trans. Mag., vol. 40, pp.3551-3558, 2004.
[5] P. W. Nutter et al., IEEE Trans. Mag., vol. 41, pp. 4327-4334, 2005.
[6] X. Hu et. all IEEE Inform. Theory, vol 51, pp.386-398, 2005.
[7] P. W. Nutter et al., IEEE Trans. Mag., vol. 41, pp. 3214-3216, 2005.

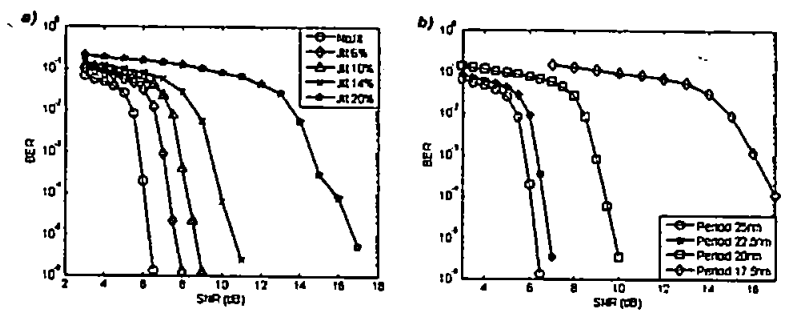


Fig. 1 BER against SNR for a) varying lithography jitter and b) varying island period.

GF(2^m) Low-Density Parity-Check Codes Derived from Cyclotomic Cosets

C. Tjhai, M. Tomlinson, R. Horan, M. Ahmed and M. Ambroze
Fixed and Mobile Communications Research, University of Plymouth, PL4 8AA, United Kingdom
email: {ctjhai,mtomlinson,rhoran,mahmed,mambroze}@plymouth.ac.uk

Abstract

Based on the ideas of cyclotomic cosets, idempotents and Mattson-Solomon polynomials, we present a new method to construct GF(2^m), where $m > 0$, cyclic low-density parity-check codes. The construction method produces the dual code idempotent which is used to define the parity-check matrix of the low-density parity-check code. An interesting feature of this construction method is the ability to increment the code dimension by adding more idempotents and so steadily decrease the sparseness of the parity-check matrix. We show that the constructed codes can achieve performance close to the sphere-packing-bound constrained for binary transmission.

1 Introduction

Since the recent rediscovery of low-density parity-check (LDPC) codes, a great deal of effort has been devoted to constructing LDPC codes that can work well with the belief-propagation iterative decoder. The studies of long block length LDPC codes are very much established. The recent works of [1],[2] have shown that, for long block lengths, the best performing LDPC codes are irregular codes and these codes can outperform turbo codes of the same block length and code-rate. These long LDPC codes have degree distributions which are derived from differential evolution [1] or Gaussian Approximation [3]. It can be shown that, using the concentration theorem [4], the performance of infinitely long LDPC codes of a given degree distribution can be characterised by the average performance of the ensemble based on cycle-free assumption. This assumption, however, does not work for short and moderate block length LDPC codes due to the inevitable existence of cycles in the underlying Tanner graph. Consequently, for a given degree distribution, the performance of short block length LDPC codes varies considerably from the ensemble performance. Various methods exist for the construction of finite block length irregular codes, for example see [5],[6], and [7]. In addition to irregular LDPC codes, algebraic constructions exist and the resulting codes are regular and usually cyclic in nature. Some examples of algebraic LDPC codes are the Euclidean and projective geometry codes [8].

It has been noticed by the authors that, in general, there is a performance association between the minimum distance (d_{min}) of a code and the decoding convergence. The irregular LDPC codes converge very well with iterative decoding, but their d_{min} are reasonably low. On the other hand, the algebraically constructed LDPC codes, which have high d_{min}, tend not to converge well with the iterative decoder. It is not surprising that algebraically constructed codes may outperform the irregular codes. The latter have error-floor which is caused by the d_{min} error-events. On the encoding side, the existence of algebraic structure in the codes is of benefit. Rather than depending on the parity-check or generator matrices for encoding, as in the case of irregular codes, a low-complexity encoder can be built for the algebraic LDPC codes. One such example is the linear shift-register encoder for cyclic LDPC codes. Assuming that n and k denote the codeword and information length respectively, algebraic codes that are cyclic offer another decoding advantage. The iterative decoder has n parity-check equations to iterate with instead of $n-k$ equations, as in the case of non-cyclic LDPC codes, and this leads to improved performance.

It has been shown that the performance of LDPC codes can be improved by going beyond the binary field [6][9]. Hu *et al.* showed that, under iterative decoding, the non binary LDPC codes have better convergence properties than the binary codes [6]. They also demonstrated that a coding gain of 0.25 dB is achieved by moving from GF(2) to GF(2^6). Non binary LDPC codes in which each symbol takes values from GF(2^m) offer an attractive scheme for higher-order modulation. The complexity of the symbol-based iterative decoder can be simplified as the extrinsic information from the component codes can be evaluated using the frequency domain dual codes decoder based on the Fast-Walsh-Hadamard transform.

Based on the pioneering works of MacWilliams on the idempotents and the Mattson-Solomon polynomials [10][11], we present a generalised construction method for algebraic GF(2^m) codes that are applicable

This research is partially funded by an UK Overseas Research Students Award Scheme.

as LDPC codes. The construction for binary codes using idempotents has been investigated by Shibuya and Sakaniwa [12], however, their investigation was mainly focused on half-rate codes. In this paper, we construct some higher code-rate non binary LDPC codes with good convergence properties. We focus on the design of short block length LDPC codes in view of the benefits for thin data-storage, wireless, command/control data reporting and watermarking applications. One of the desirable features in any code construction technique is an effective method of determining the d_{min} and this feature is not present in irregular code construction methods. With our idempotent-based method, the d_{min} of a constructed code can be easily lower-bounded using the well-known BCH bound.

The rest of the paper is organised as follows. In Section 2, we briefly review the theory of the cyclotomic cosets, idempotents and Mattson-Solomon polynomials. Based on the theory, we devise a generalised construction algorithm and present an example in Section 3. We also outline an efficient and systematic algorithm to search for algebraic LDPC codes in Section 3. In Section 4, we demonstrate the performance of the constructed codes by means of simulation and Section 5 concludes this paper.

2 Cyclotomic Coset, Idempotent and Mattson-Solomon Polynomial

We briefly review the theory of cyclotomic cosets, idempotents and Mattson-Solomon polynomials to make this paper relatively self-contained. Let us first introduce some notations that will be used throughout this paper. Let m and m' be positive integers with $m|m'$, so that GF(2^m) is a subfield of GF($2^{m'}$). Let n be a positive odd integer and GF($2^{m'}$) be the splitting field for $x^n - 1$ over GF(2^m), so that $n|2^{m'} - 1$. Let $r = (2^{m'} - 1)/n$, $l = (2^{m'} - 1)/(2^m - 1)$, α be a generator for GF($2^{m'}$) and β be a generator for GF(2^m), where $\beta = \alpha^l$. Let $T_a(x)$ be the set of polynomials of degree at most $n - 1$ with coefficients in GF(2^a).

Definition 2.1: If $a(x) \in T_{m'}(x)$, then the finite-field transform of $a(x)$ is:

$$A(z) = \mathrm{MS}\left(a(x)\right) = \sum_{j=0}^{n-1} a(\alpha^{-rj}) z^j \,, \qquad (1)$$

where $A(z) \in T_{m'}(z)$. This transform is widely known as the Mattson-Solomon polynomial. The inverse transform is:

$$a(x) = \mathrm{MS}^{-1}\left(A(z)\right) = \frac{1}{n} \sum_{i=0}^{n-1} A(\alpha^{ri}) x^i \,. \qquad (2)$$

Definition 2.2: Consider $e(x) \in T_m(x)$, $e(x)$ is an idempotent if the property of $e(x) = e(x)^2 \pmod{x^n - 1}$ is satisfied. In the case of $m = 1$, the property of $e(x) = e(x^2) \pmod{x^n - 1}$ is also satisfied.

An $[n, k]$ cyclic code C can be described by the generator polynomial $g(x) \in T_m(x)$ of degree $n - k$ and the parity-check polynomials $h(x) \in T_m(x)$ of degree k such that $g(x)h(x) = x^n - 1$. It is widely known that idempotents can be used to generate C. Any GF(2^m) cyclic code can also be described by a unique idempotent $e_g(x) \in T_m(x)$ which consists of a sum of primitive idempotents. This unique idempotent is known as the generating idempotent and, as the name implies, $g(x)$ is a divisor of this idempotent, i.e. $e_g(x) = m(x)g(x)$, where $m(x)$ contains the repeated factors or non-factors of $x^n - 1$.

Lemma 2.1: If $e(x) \in T_m(x)$ is an idempotent, $E(z) = \mathrm{MS}(e(x)) \in T_1(z)$.

Proof: (cf. [10, Chap. 8]) Since $e(x) = e(x)^2 \pmod{x^n - 1}$, from (1), it follows that $e(\alpha^{-rj}) = e(\alpha^{-rj})^2$, $\forall j \in \{0, 1, \ldots, n-1\}$ and for some integers r and l. Clearly, $e(\alpha^{-rj}) \in \{0, 1\}$ implying that $E(z)$ is a binary polynomial. □

Definition 2.3: If s is a positive integer, the binary cyclotomic coset of $s \pmod n$ is:

$$C_s = \left\{2^i s \pmod n \mid 0 \le i \le t\right\},$$

where we shall always assume that the subscript, s, is the smallest element in the set C_s, and t is the smallest positive integer with the property that $2^{t+1}s = s \pmod n$. If $\mathcal{N}$ is the set consisting of the smallest elements of all possible cyclotomic cosets then

$$C = \bigcup_{s \in \mathcal{N}} C_s = \{0, 1, 2, \ldots, n-1\}.$$

Lemma 2.2: Let $s \in \mathcal{N}$ and let $C_{s,i}$ represents the ith element of C_s. Let the polynomial $e_s(x) \in T_m(x)$ be given by

$$e_s(x) = \sum_{0 \le i \le |C_s|-1} e_{C_{s,i}} x^{C_{s,i}}, \qquad (3)$$

where $|C_s|$ is the number of elements in C_s and $e_{C_{s,i}}$ is defined below.

i) if $m = 1$, $e_{C_{s,i}} = 1$,
ii) if $m > 1$, $e_{C_{s,i}}$ is defined recursively as follows:

$$\text{for } i = 0, \quad e_{C_{s,i}} \in \{1, \beta, \beta^2, \ldots, \beta^{2^m-2}\},$$
$$\text{for } i > 0, \quad e_{C_{s,i}} = e^2_{C_{s,i-1}}.$$

The polynomial so defined, $e_s(x)$, is an idempotent. We term $e_s(x)$ a *cyclotomic* idempotent.

Definition 2.4: Let $\mathcal{M} \subseteq \mathcal{N}$ and let $u(x) \in T_m(x)$ be

$$u(x) = \sum_{s \in \mathcal{M}} e_s(x) \,, \qquad (4)$$

then (refer to Lemma 2.2) $u(x)$ is an idempotent and we call $u(x)$ a *parity-check* idempotent.

The parity-check idempotent $u(x)$ can be used to describe the code C, the parity check matrix being made up of the n cyclic shifts of the polynomial $x^{\deg(u(x))}u(x^{-1})$.

Note that $(u(x), 1+x^n) = h(x)$[‡] and, in general, $wt(u(x))$[§] is much lower than $wt(h(x))$. Based on this observation and the fact that $u(x)$ contains all the roots of $h(x)$, we can construct cyclic codes that have a low-density parity-check matrix.

Definition 2.5: Let the polynomial $f(x) \in T_1(x)$. The difference enumerator of $f(x)$, denoted as $\mathcal{D}(f(x))$, is defined as follows:

$$\begin{aligned}\mathcal{D}(f(x)) &= f(x)f(x^{-1}) \\ &= d_0 + d_1 x + \ldots + d_{n-1}x^{n-1}, \qquad (5)\end{aligned}$$

where we assume that $\mathcal{D}(f(x))$ is a modulo $x^n - 1$ polynomial with real coefficients.

Lemma 2.3: Let $m = 1$ and let d_i for $0 \le i \le n-1$ denote the coefficients of $\mathcal{D}(u(x))$. If $d_i \in \{0,1\}$, for $i = 1, 2, \ldots, n-1$, the parity-check polynomial derived from $u(x)$ is orthogonal on each position in the n-tuple. Consequently (i) the d_{min} of the resulting C is $1 + wt(u(x))$ and (ii) the underlying Tanner Graph has girth of at least 6.

Proof: (i) (cf. [13, Theorem 10.1]) Let a codeword $c(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1}$ and $c(x) \in T_1(x)$. For each non zero bit position c_j of $c(x)$ where $j \in \{0, 1, \ldots, n-1\}$, there are $wt(u(x))$ parity-check equations orthogonal to position c_j. Each of the parity-check equation must check another non zero bit c_l $l \neq j$ so that the equation is satisfied. Clearly, $wt(c(x))$ must equal to $1 + wt(u(x))$ and this is the minimum weight of all codewords. (ii) The direct consequence of having orthogonal parity-check equation is the absence of cycles of length 4 in the Tanner Graphs. It can be shown that there exists three integers a, b and c, such that $2(b-a) \equiv (c-b)$ for $a < b < c$. If these three integers are associated to the variable nodes in the Tanner Graphs, a cycle of length 6 can be formed between these variable nodes and some check nodes. □

From Lemma 2.3 we can deduce that, for $i = 1, 2, \ldots, n-1$, $u(x)$ is the parity-check polynomial for One-Step Majority-Logic Decodable codes if $d_i \in \{0,1\}$, or the parity-check polynomial for Difference-Set Cyclic codes if $d_i = 1$.

Lemma 2.4: For the non binary GF(2^m) cyclic codes, the d_{min} is bounded by:

$$d_0 < d_{min} \le \min\left(wt(g(x)), 1 + wt(u(x))\right),$$

where d_0 denotes the maximum run of consecutive ones in $U(z)$ taken cyclically modulo n.

Proof: The lower-bound of the d_{min} of a cyclic code, BCH bound is determined from the number of consecutive roots of $e_g(x)$ and from Lemma 2.1, it is equivalent to the run of consecutive ones in $U(z)$. □

[‡] (a, b) denotes the greatest common divisor of a and b

[§] $wt(f(x))$ denotes the weight of polynomial $f(x)$.

3 Code Construction Algorithm

Based on the mathematical theories outlined above, we devise an algorithm to construct GF(2^m) cyclic codes which are applicable for iterative decoding. The construction algorithm can be described in the following procedures:

1) Given the integers m and n, find the splitting field (GF($2^{m'}$)) of $x^n - 1$ over GF(2^m). We can only construct GF(2^m) cyclic codes of length n if and only if the condition of $m|m'$ is satisfied.
2) Generate the cyclotomic cosets modulo $2^{m'} - 1$ and denote it C'.
3) Derive a polynomial $p(x)$ from C'. Let $s \in \mathcal{N}$ be the smallest positive integer such that $|C'_s| = m$. The polynomial $p(x)$ is the minimal polynomial of α^s:
$$p(x) = \prod_{0 \le i < m} \left(x + \alpha^{C'_{s,i}}\right) \qquad (6)$$
Construct all elements of GF(2^m) using $p(x)$ as the primitive polynomial.
4) Let C be the cyclotomic cosets modulo n and $\mathcal{N}$ be a set containing the smallest number in each coset of C. Assume that there exists a non empty set $\mathcal{M} \subset \mathcal{N}$ and following Definition 2.4, construct the parity-check idempotent $u(x)$. The coefficients of $u(x)$ can be assigned following Lemma 2.2.
5) Generate the parity-check matrix of C using the n cyclic shifts of $x^{\deg(u(x))}u(x^{-1})$.
6) Compute r and l, then take the Mattson-Solomon polynomial of $u(x)$ to produce $U(z)$. Obtain the code dimension and the lower-bound of the d_{min} from $U(z)$.

Note that care should be taken to ensure that there is no common factor between n and all of the exponents of $u(x)$, apart from unity, in order to avoid a degenerate code.

Example 3.1: Let us assume that we want to construct a GF(64) $n = 21$ cyclic idempotent code. The splitting field for $x^{21} - 1$ over GF(64) is GF(64) and this implies that $m = m' = 6$, $r = 3$ and $l = 1$. Let C and C' denote the cyclotomic cosets modulo n and $2^{m'} - 1$ respectively. $|C'_1| = 6$ and therefore the primitive polynomial $p(x)$ has roots of α^j, $\forall j \in C'_1$, i.e. $p(x) = 1 + x + x^6$. By letting $1 + \beta + \beta^6 = 0$, where $\beta = \alpha$, all of the elements of GF(64) can be defined. If we let $u(x)$ be the parity-check idempotent generated by the sum of the cyclotomic idempotents defined by C_s where $s \in \mathcal{M} = \{5, 7, 9\}$ and $e_{C_{s,0}}$, $\forall s \in \mathcal{M}$ be β^{23}, 1 and 1 respectively,

$$\begin{aligned}u(x) = {} & \beta^{23}x^5 + x^7 + x^9 + \beta^{46}x^{10} + \beta^{43}x^{13} + \\ & x^{14} + x^{15} + \beta^{53}x^{17} + x^{18} + \beta^{58}x^{19} + \beta^{29}x^{20}\end{aligned}$$

and its Mattson-Solomon polynomial $U(z)$ tells us that the [21, 15] cyclic code over GF(64) has $d_{min} >= 5$.

A systematic algorithm has been developed to sum up all combinations of the cyclotomic idempotents to search for all possible GF(2^m) cyclic codes C of a given length. The search algorithm is targeted on the following key parameters:

1) Sparseness of the resulting parity-check matrix. Since the parity-check matrix of C is directly derived from $u(x)$ which consists of the sum of the cyclotomic idempotents, we are only interested in low-weight cyclotomic idempotents. Let us define W_{max} as the maximum $wt(u(x))$ then the search algorithm will only choose the cyclotomic idempotents whose sum has total weight less than or equal to W_{max}.
2) High code-rate. The number of roots of $u(x)$ which are also roots of unity define the dimension of C and let us define k_{min} as the minimum information length of C. We are only interested in the sum of the cyclotomic idempotents whose Mattson-Solomon polynomial has at least k_{min} zeros.
3) High d_{min}. Let us define d as the minimum value of the d_{min} of C. The sum of the cyclotomic idempotents should have at least $d-1$ consecutive powers of β which are roots of unity but not roots of $u(x)$.

The search algorithm can be relaxed to allow the existence of cycles of length 4 in the resulting parity-check matrix of C. The condition of cycles-of-length-4 is not crucial as we will show later that there are codes that have good convergence properties when decoded using iterative decoder. The same observation can also be found in [14], [15] and [16]. Clearly, by eliminating the cycles-of-length-4 constraint, we can construct more codes.

Following Definitions 2.1 and 2.4:

$$U(z) = \mathrm{MS}\left(\sum_{s\in\mathcal{M}} e_s(x)\right) = \sum_{s\in\mathcal{M}} E_s(z),$$

and hence, it is possible to maximise the run of the consecutive ones in $U(z)$ if the coefficients of $e_s(x)$ are aligned appropriately. It is therefore important that all possible non zero values of $e_{C_{s,0}}$, $\forall s \in \mathcal{M}$, are included in the search in order to guarantee that we can obtain codes with the highest possible d_{min}, or at least to obtain a better estimate of the d_{min}.

4 Code Performance

As an example of the performance attainable from an iterative decoder, computer simulations have been carried out for several GF(2^m), where $m \geq 2$, cyclic LDPC codes. Results for $m = 1$ can be found in [16]. We assume binary phase-shift keying (BPSK) signalling and the iterative decoder used is the modified belief-propagation decoder which approximates the performance of a maximum-likelihood decoder [17][18][19].

Fig. 1 shows the frame-error-rate (FER) performance of the GF(2^4)[85, 48] cyclic LDPC code. The performance of this code is compared against the sphere-packing-bound [20],[21] for binary codes of length 340 bits and code-rate of 0.5647. Since this bound does not take modulation into account, we offset it by a BPSK transmission loss¶. We can see that the performance of the code is within 0.8 dB away from the bound at 10^{-4} FER.

While the code in Fig. 1 does not have cycles of length 4 in the underlying factor graph, there are good convergence codes which do have these short cycles. One such example is the GF(4)[51, 29] cyclic code whose FER performance is shown in Fig. 2. At 10^{-4} FER, this code performs within 0.4 dB away from the sphere-packing-bound for length of 102 bits and code-rate of 0.5686.

The performance of the cyclic codes is also compared against that of LDPC codes from other constructions and this is shown in Fig. 3 and 4. Fig. 3 compares the FER of the GF(64)[21, 15] cyclic code, [126, 90, 4] binary irregular LDPC code constructed using the progressive-edge-growth (PEG) algorithm [6], and [128, 96, 6] binary quasi-cyclic LDPC code. The GF(64)[21, 15] cyclic code has outstanding performance; at 10^{-4} FER, it is around 0.2 dB away from the sphere-packing-bound. Similarly, Fig. 4 demonstrates the performance of the GF(4)[255, 175] cyclic LDPC code in comparison to the binary irregular (PEG construction) and quasi-cyclic LDPC codes of similar block length and code-rate. Compared to the sphere-packing-bound at 10^{-4} FER, the performance of the GF(4)[255, 175] cyclic code is around 0.45 dB away. As shown in Fig. 3 and 4, the algebraically constructed codes have superior performance in comparison to the irregular and quasi-cyclic LDPC codes. Irregular LDPC codes are known to have capacity-approaching performance, however, this applies for long block length codes only. To design an irregular LDPC code, one needs to know the degree distribution, which can be optimised using the density-evolution approach. Due to the infinite block length assumption, this optimisation approach is not accurate for short block length, and as a consequence, it is difficult to design capacity-approaching short irregular LDPC codes. For the case of quasi-cyclic codes, we believe the superior performance of the cyclic codes is due to the extra parity-check equations.

Table I outlines some examples of the non binary GF(2^m) cyclic LDPC codes. The detail parameters are given in the table.

¶In the rest of this paper, when the term sphere-packing-bound is used, we shall refer to that offset by the information theoretical loss associated with binary transmission.

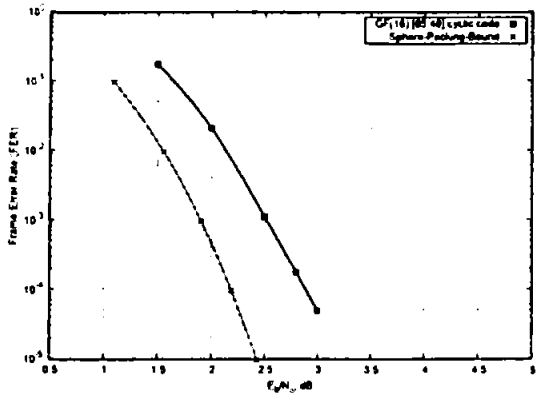


Fig. 1. FER performance of the GF(2^4)[85, 18] cyclic LDPC code

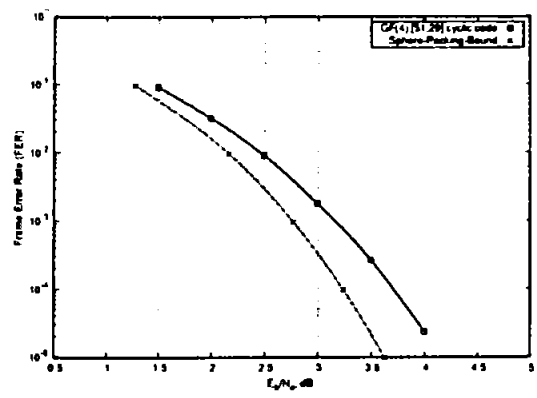


Fig. 2. FER performance of the GF(2^2)[51, 29] cyclic LDPC code

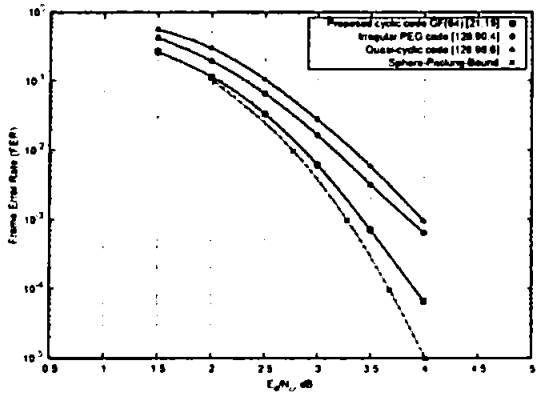


Fig. 3. FER performance of the GF(2^6)[21, 15] cyclic code and irregular and quasi-cyclic LDPC codes of similar block length and code-rate

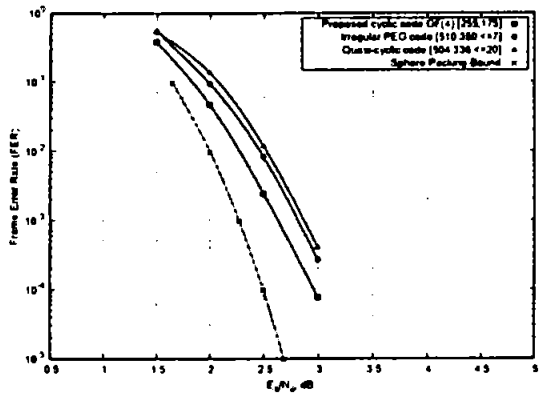


Fig. 4. FER performance of the GF(2^2)[255, 175] cyclic code and irregular and quasi-cyclic LDPC codes of similar block length and code-rate

5 Conclusions

An algebraic construction technique for GF(2^m) ($m > 0$) LDPC codes based on summing the cyclotomic idempotents to define the parity-check polynomial is able to produce a large number of cyclic codes. Since we consider step-by-step summation of the cyclotomic idempotents, we are able to control the sparseness of the resulting parity-check matrix. The lower-bound of the d_{min} and the dimension of the codes can be easily determined from the Mattson-Solomon polynomial of the resulting idempotent. For GF(2) case where the parity-check polynomials are orthogonal on each bit position, we can even determine the true d_{min} of the codes regardless of the code length. In fact, this special class of binary cyclic codes are the Difference-Set Cyclic and the One-Step Majority-Logic Decodable codes, which can be easily constructed using our method. For non-binary cases, if the constructed code has low d_{min}, we can concatenate this code with an inner binary code to trade improvement in d_{min} with loss in code-rate.

Simulation results have shown that these codes can converge well under iterative decoding and their performance is close to the sphere-packing-bound of binary codes for the same block length and code-rate. The excellent performance of these codes coupled with their low-complexity encoder offers an attractive coding scheme for applications that required short block-lengths such as thin data-storage, wireless, command/control data reporting and watermarking.

References

[1] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[2] S. Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke, "On the Design of Low-Density Parity Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Comm. Letters*, vol. 3, pp. 58–60, Feb. 2001.

[3] S. Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation," *IEEE Trans. Inform. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[4] T. J. Richardson and R. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

TABLE I

EXAMPLES OF THE CONSTRUCTED NON BINARY CODE

C	$u(x)$	d_{min}	d_b^1	Comment
GF(4) [51, 29]	$\beta^2x^3+\beta x^6+\beta^2x^{12}, x^{17}+\beta x^{24}+\beta x^{27}+x^{34}+\beta^2x^{39}+\beta x^{46}+\beta^2x^{48}$	5	10	$m = 2$, $m' = 8$, $r = 5$ and $l = 85$
GF(4) [255, 175]	$\beta x^7+\beta^2x^{14}+\beta x^{28}+\beta^2x^{56}+x^{111}+\beta x^{112}+x^{123}+\beta^2x^{131}+x^{183}+x^{189}+\beta x^{193}+x^{219}+x^{222}+\beta^2x^{224}+x^{237}+x^{246}$	≥ 17	≤ 20	$m = 2$, $m' = 8$, $r = 1$ and $l = 85$
GF(4) [273, 191]	$\beta^2x^{23}+\beta x^{37}+\beta x^{46}+\beta^2x^{74}+\beta x^{91}+\beta^2x^{92}+\beta^2x^{95}+\beta^2x^{107}+x^{117}+\beta x^{148}+\beta^2x^{155}+\beta^2x^{182}+\beta x^{184}+\beta x^{190}+x^{195}+\beta x^{214}+x^{234}$	≥ 18	≤ 20	$m = 2$, $m' = 12$, $r = 15$ and $l = 1365$
GF(8) [63, 40]	$1+\beta^5x^9+\beta x^{13}+\beta^3x^{18}+\beta^2x^{19}+\beta^2x^{26}+\beta^6x^{36}+\beta^4x^{38}+\beta x^{41}+\beta^4x^{52}$	≥ 6	10	$m = 3$, $m' = 6$, $r = 1$ and $l = 9$
GF(8) [63, 43]	$\beta^2x^9+\beta^3x^{11}+\beta^4x^{18}+x^{21}+\beta^6x^{22}+\beta^3x^{25}+x^{27}+\beta x^{36}+\beta^5x^{37}+x^{42}+\beta^5x^{44}+x^{45}+\beta^6x^{50}+x^{54}$	≥ 8	≤ 12	$m = 3$, $m' = 6$, $r = 1$ and $l = 9$
GF(8) [91, 63]	$\beta^6x+\beta^5x^2+\beta^3x^4+\beta^6x^8+\beta x^{13}+\beta^5x^{16}+\beta^5x^{23}+\beta^2x^{26}+\beta^3x^{32}+\beta^5x^{37}+\beta^3x^{46}+\beta^4x^{52}+\beta^6x^{57}+\beta^6x^{64}+\beta^3x^{74}$	≥ 8	≤ 10	$m = 3$, $m' = 12$, $r = 45$ and $l = 585$
GF(16) [85, 48]	$1+\beta^{12}x^{21}+\beta^9x^{42}+\beta^6x^{63}+\beta^3x^{69}+\beta^9x^{77}+\beta^{12}x^{81}+\beta^6x^{83}+\beta^3x^{84}$	≥ 7	≤ 12	$m = 4$, $m' = 8$, $r = 3$ and $l = 17$
GF(32) [31, 20]	$1+\beta^{28}x^5+\beta^7x^9+\beta^{25}x^{10}+x^{11}+x^{13}+\beta^{14}x^{18}+\beta^{10}x^{20}+x^{21}+x^{22}+x^{26}$	≥ 7	12	$m = 5$, $m' = 5$, $r = l$ and $l = 1$
GF(32) [31, 21]	$\beta^{23}x^5+\beta^{29}x^9+\beta^{15}x^{10}+\beta x^{11}+\beta^4x^{13}+\beta^{27}x^{18}+\beta^{30}x^{20}+\beta^{16}x^{21}+\beta^2x^{22}+\beta^8x^{26}$	≥ 4	8	$m = 5$, $m' = 5$, $r = 1$ and $l = 1$
GF(64) [21, 15]	$\beta^{23}x^5+x^7+x^9+\beta^{46}x^{10}+\beta^{43}x^{13}+x^{14}+x^{15}+\beta^{53}x^{17}+x^{18}+\beta^{58}x^{19}+\beta^{29}x^{20}$	≥ 5	8	$m = 6$, $m' = 6$, $r = 3$ and $l = 1$

[1] The minimum distance of the binary image; it is determined using the improved Zimmermann algorithm [22].

[5] J. Campello and D. S. Modha, "Extended Bit-Filling and LDPC Code Design," in *Proc. IEEE GLOBECOM Conf.*, vol. 2, pp. 985–989, 25-29 Nov. 2001.

[6] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs," *IEEE Trans. Inform. Theory*, vol. 51, pp. 386–398, Jan. 2005.

[7] A. Ramamoorthy and R. D. Wesel, "Construction of Short Block Length Irregular Low-Density Parity-Check Codes," in *Proc. IEEE Intl. Conf. Commun.*, vol. 1, pp. 410–414, 20-24 Jun. 2004.

[8] Y. Kou, S. Lin, and M. Fossorier, "Low density parity check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.

[9] M. C. Davey and D. J. C. MacKay, "Low-Density Parity-Check Codes over GF(q)," *IEEE Comm. Letters*, vol. 2, pp. 165–167, June 1998.

[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977. ISBN 0 444 85193 3.

[11] F. J. MacWilliams, "A table of primitive binary idempotents of odd length n, $7 \leq n \leq 511$," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 118–123, Jan. 1979.

[12] T. Shibuya and K. Sakaniwa, "Construction of cyclic codes suitable for iterative decoding via generating idempotents," *IEICE Trans. Fundamentals*, vol. E86-A, no. 4, 2003.

[13] W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. MIT Press., 1972.

[14] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-Limit Quasi-Cyclic Low-Density Parity-Check Codes," *IEEE Trans. Commun.*, vol. 52, pp. 1038–1042, Jul. 2004.

[15] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on Finite Geometries," *IEEE Trans. Inform. Theory*, vol. 51, pp. 572–596, Feb. 2005.

[16] R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed, "Idempotents, Mattson-Solomon Polynomials and Binary LDPC Codes," to appear in IEE Proceedings Communications.

[17] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Analysis of non convergence blocks at low and moderate snr in sce turbo schemes," in *Proc. SPSC 2003 8th Intl. Workshop Signal Processing for Space Commun.*, pp. 121–128, 24-26 Sep. 2003.

[18] C. J. Tjhai, E. Papagiannis, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed, "Improved iterative decoder for LDPC codes with performance approximating to a maximum likelihood decoder," UK Patent Application 0409306.8, Apr. 2004.

[19] E. Papagiannis, M. Ambroze, M. Tomlinson, and M. Ahmed, "Improved Decoding of Low-Density Parity-Check Codes with Low, Linearly Increased Added Complexity," in *Proc. 4th IASTED Intl. Conf. Commun. Syst. Networks*, pp. 152–157, 12-14 Sep. 2004.

[20] C. E. Shannon, "Probability of error for optimal codes in a gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, May 1959.

[21] S. Dolinar, D. Divsalar, and F. Pollara, "Code Performance as a Function of Block Size," *TMO Progress Report 42-133*, 15 May 1998. [Online] Available: `http://tmo.jpl.nasa.gov/`.

[22] M. Grassl, "Searching for Good Linear Codes," in *Discovering Mathematics with MAGMA*, Springer, Berlin, W. Bosma and J. Cannon ed., 2006.

In the Proc. of IEEE ISOC ITW2005 on Coding and Complexity; editor M.J. Dinneen; co-chairs U. Speidel and D. Taylor; pages 72-76

A Finite-Field Transform Domain Construction of Binary Low-Density Parity-Check Codes

R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze and M. Ahmed
Fixed and Mobile Communications Research,
University of Plymouth,
Plymouth PL4 8AA,
United Kingdom,
email: {rhoran,ctjhai,mtomlinson,mambroze,mahmed}@plymouth.ac.uk

***Abstract*— A new method of finding binary cyclic codes from the finite-field transform domain is presented. These cyclic codes have sparse parity-check matrix and thus are suitable for iterative decoding. Some interesting properties of the proposed construction method include the knowledge of the minimum distance and the ability to trade the increase in code dimension with a reduction in the parity-check matrix sparsity. By means of simulations, we show that the error correcting performance of the codes under iterative decoding is very close to the sphere-packing-bound constrained for binary transmission.**

I. Introduction and Background

The use of idempotents in the construction of cyclic error correcting codes is well established and the resulting literature is extensive (for example, see [1],[2],[3]). The basic building blocks for this theory are the primitive idempotents. Any cyclic code may be described by a unique idempotent and this idempotent is a sum of primitive idempotents. For binary cyclic codes, efficient algorithms exist for the calculation of these primitive idempotents.

Another way of constructing idempotents in the binary case is by using cyclotomic cosets and it was this property which was exploited by Shibuya and Sakaniwa in [4]. Their goal was to use idempotents to construct parity-check matrices for LDPC codes which have no cycles of length 4 in their bipartite graphs. At the heart of their technique is a lemma which is a variation of a result used by Weldon [5], for the construction of difference set cyclic codes. Using this lemma and a subsequent theorem, they were able to simplify the problem of determining which of the idempotents that are constructed, using a single cyclotomic coset, do not have cycles of length 4. They then extended this theory to more general idempotents.

This approach to the construction of LDPC codes has the great advantage of simplicity, the parity-check matrices depend only upon the correct choice of cyclotomic cosets and these are very easily calculated. However, we believe that this advantage is offset by some fundamental weaknesses.

Whilst the absence of 4 cycles is a desirable objective in the construction of LDPC codes it is not mandatory [6], since there are some good codes which do not have this property. An example of such a code is included in this paper. The code rate is also an important property of codes but, as Shibuya and Sakaniwa admit in their conclusion, the codes which they construct in this way are expected to have a large minimum distance at the expense of rate. The minimum distance of a code is a crucial property but there is no indication of how either a single cyclotomic coset, or combinations of more than one cyclotomic cosets, should be chosen to guarantee that the code constructed has a large minimum distance.

In order to address the question of how to choose idempotents which will produce good LDPC codes we propose an entirely different route. As in [4], we shall deal exclusively with binary cyclic codes. Making effective use of the finite-field transform, which is commonly known as the Mattson-Solomon polynomial, we produce an algorithm which not only allows us to choose, in a systematic way, idempotents with low weight, and therefore a correspondingly sparse parity-check matrix, but also with the desirable features that the corresponding codes have a high code-rate and a large minimum distance.

This paper is organised as follows. In section II we shall review the necessary theory and explain how it will be used to provide an algorithm for the determination of idempotents which may be used to construct good codes. In section III the design and implementation of this algorithm is given and then, in section IV, some of the results are given. Finally, in section V, we draw our conclusions on this approach.

II. Binary Idempotents

Let $F = GF(2)$, n be a positive odd integer and $\mathcal{F}$ be the splitting field for $x^n - 1$ over F. Let $\alpha \in \mathcal{F}$ be a primitive nth root of unity and let $T(x)$ be the polynomials in $\mathcal{F}[x]$ of degree $\leq n-1$. If $a(x) \in T(x)$ then the map $\Phi : T \to T$ is defined by

$$[\Phi(a)](z) = \sum_{j=1}^{n} a(\alpha^j) z^{n-j} \qquad (1)$$

and $[\Phi(a)](z)$ is the Mattson Solomon polynomial of $a(x)$ (see [1]). (We use x and z for the polynomial variables to distinguish between the polynomials in the domain and codomain of Φ.) If $\circ$ is multiplication of polynomials mod $(x^n - 1)$ and $*$ is defined on $T(z)$ by the rule $(\sum a_i z^i) * (\sum b_i z^i) = \sum a_i b_i z^i$ then it is well known [1],[3], that

$$\Phi : (T, +, \circ) \to (T, +, *)$$

is an isomorphism of rings, in particular it is an isomorphism of the additive groups.

A polynomial $e(x) \in (T, +, \circ)$ is called an idempotent if it satisfies $e(x) \circ e(x) = e(x)^2 = e(x)$ property. If $S(x)$ is the subset of $T(x)$ consisting of polynomials with coefficients in $GF(2)$ (binary polynomials) and $E(x)$ is the subset of $T(x)$ consisting of idempotents, both of these subsets are *additive subgroups* of $T(x)$. It is easy to show (see [1]) that

$$\Phi : (S(x), +) \rightarrow (E(z), +) \quad (2)$$

$$\Phi : (E(x), +) \rightarrow (S(z), +) \quad (3)$$

are both isomorphisms and from this it is obvious that

$$\Phi : (S(x) \cap E(x), +) \rightarrow (E(z) \cap S(z), +) \quad (4)$$

is also an isomorphism.

Suppose that $u(x)$ is a binary idempotent which is used to construct a parity-check matrix for a cyclic code. The parity-check matrix is constructed from the n-cyclic shifts of $x^n u(x^{-1})$, and so for the resulting code to be a LDPC code, $u(x)$ must have low weight.

If $h(x) = \gcd(x^n - 1, u(x))$ and $g(x) = (x^n - 1)/h(x)$, then $g(x)$ is the generator of the cyclic code. If the generator, $g(x)$, has degree $n - k$, the dimension of the code is k and the larger the value of k, the better the code rate. Since $g(x)$ is a divisor of $x^n - 1$, all of the zeros of $g(x)$ are nth roots of unity, and there are $n - k$ of these. Further, $\gcd(g(x), h(x)) = 1$ and $x^n - 1 = h(x)g(x)$, so that the number of distinct nth roots of unity which are also roots of $u(x)$ is k. The dimension of the code is therefore the number of nth roots of unity which are also roots of $u(x)$.

The BCH bound of the code is determined by the number of consecutive powers of α, taken cyclically (mod n), which are also roots of $g(x)$. For the reasons outlined in the previous paragraph, this is precisely the same as the number of consecutive powers of α, taken cyclically (mod n), which are not roots of $u(x)$.

The important features of the code are therefore determined by:

1) the weight of the idempotent $u(x)$,
2) the number of nth roots of unity which are roots of $u(x)$,
3) the number of consecutive powers of α which are not roots of $u(x)$.

Take $u(x) \in S(x) \cap E(x)$ and let $[\Phi(u)](z) = \theta(z)$ be its MS polynomial. The inverse mapping

$$\Phi^{-1} : (S(z) \cap E(z), +) \rightarrow (E(z) \cap S(x), +) \quad (5)$$

is defined as follows: If $A(z) = [\Phi(a)](z)$ is the Mattson Solomon polynomial of the polynomial $a(x) = a_0 + a_1 x + \cdots + a_n x^{n-1}$ then, for $i = \{0, \ldots, n-1\}$,

$$a_i = \frac{1}{n} A(\alpha^i) \quad (6)$$

(see [1]). Let $h(z) = \gcd(\theta(z), z^n - 1)$ and let $f(z) = (z^n - 1)/h(z)$. The three key properties relating to the idempotent $u(x)$, listed above, are easily gleaned from its Mattson Solomon polynomial $\theta(z)$, and $f(z)$, as follows:

A. *The weight of* $u(x)$.

The weight of $u(x)$ is the number of nth roots of unity which are zeros of $f(z)$. To see this note that $f(\alpha^i) = 0$ if and only if $\theta(\alpha^i) = 1$, since idempotents take only the values 0 and 1 in $\mathcal{F}$. Now $u(x) = [\Phi^{-1}(\theta)](x)$ and the coefficients of $u(x) = u_0 + u_1 x + \ldots + u_{n-1} x^{n-1}$ are given by

$$u_i = \theta(\alpha^i) \bmod 2 \quad \text{for } i = 0, \ldots, n-1 \quad (7)$$

(*cf.* equation 6). Thus $u_i = 1$ precisely when $f(\alpha^i) = 0$, giving the weight of $u(x)$ as the degree of the polynomial $f(z)$.

B. *The zeros of* $u(x)$.

From the definition of the MS polynomial (equation 1),

$$\theta(z) = \sum_{j=1}^{n} u(\alpha^j) z^{n-j} \quad (8)$$

and the number of zeros of $u(x)$ which are roots of unity is clearly $n - \mathrm{wt}(\theta(z))$.

C. *The BCH bound of the code.*

The BCH bound of the code is the largest number of consecutive powers of α which are not roots of $u(x)$, i.e. the number of consecutive i, taken (mod n), such that $u(\alpha^i) = 1$. From equation 8, this is the largest number of consecutive non-zero coefficients in $\theta(z)$, taken cyclically (mod n).

Using this information, a systematic search for idempotents can now be made in increasing order of weight, with accompanying knowledge of the number of roots which are nth roots of unity and the corresponding BCH bound. This algorithm is constructed in the Mattson Solomon domain.

Let the decomposition of $z^n - 1$ into irreducible (over $F = GF(2)$) polynomials be $z^n - 1 = f_1(z) f_2(z) \ldots f_t(z)$. For $i = 1, \ldots, t$, let $k_i(z) = (z^n - 1)/f_i(z)$ and let $\theta_i(z)$ be the associated primitive idempotent (see [1] or [3]). These are displayed below in an array, together with other idempotents:

$$\left.\begin{array}{lll} u_1(x) & \theta_1(z) & f_1(z) \\ u_2(x) & \theta_2(z) & f_2(z) \\ \vdots & \vdots & \vdots \\ u_t(x) & \theta_t(z) & f_t(z) \end{array}\right\} \quad (9)$$

Here $u_1(x), u_2(x), \ldots, u_t(x)$ are the idempotents whose Mattson Solomon polynomials are $\theta_1(z), \theta_2(z), \ldots, \theta_t(z)$, respectively. Let $I \subseteq \{1, 2, \ldots, t\}$ and let $u(x), \theta(z)$ and $f(z)$ be defined as $u(x) = \sum_{i \in I} u_i(x)$, $\theta(z) = \sum_{i \in I} \theta_i(z)$ and $f(z) = \prod_{i \in I} f_i(z)$. From the properties of primitive idempotents, if $h(z) = \gcd(\theta(z), z^n - 1)$ then it follows that $\gcd(f(z), h(z)) = 1$ and $z^n - 1 = f(z)h(z)$. The idempotent $u(x)$ will now have the following properties.

$$\mathrm{wt}(u(x)) = \sum_{i \in I} \deg(f_i(z)), \quad (10)$$

$$\text{number of zeros of } u(x) = n - \mathrm{wt}(\theta(z)). \quad (11)$$

The BCH bound is determined from $\theta(z)$ as explained in section II-C.

Since methods for finding the $\theta_i(z)$ and $f_i(z)$ are well documented (for example, see [7]) a search algorithm can be built around this observation to find a suitable weight idempotent with a known number of zeros and a known BCH bound. The rows of the array (equation 9), are ordered by the degree of the polynomials, i.e. $\deg(f_i(z)) \leq \deg(f_{i+1}(z))$ for all i, and a search can be made in increasing order of weight. When a successful outcome has been obtained, only at this stage is the *inverse finite-field transform* (MS^{-1}) evaluated to find the corresponding idempotent. All of the information which is required will already be known.

III. Design and Implementation

If t denotes the number of cyclotomic cosets modulo n, the complexity of an exhaustive search algorithm is $\mathcal{O}(2^t)$. We reduce this search complexity by targeting the search on the three key parameters:

A. Sparseness of the parity-check matrix

In [5], Weldon introduced difference-set cyclic codes. These codes have the desirable property that the parity-check equations are orthogonal on all bits and have no cycles of length 4 in their bipartite graphs. A necessary condition for this is that if $v(x)$ is the polynomial which generates the parity-check matrix then the weight of $v(x)$ must satisfy the inequality

$$\mathrm{wt}(v(x))(\mathrm{wt}(v(x)) - 1) \leq n\,, \tag{12}$$

where n is the code length. Since the weights of the idempotents $u(x)$ are related to the degrees of the $f_i(z)$ by equation 10, a reasonable bound is

$$\sum_{i \in I} \deg(f_i(z)) \leq \sqrt{n}\,. \tag{13}$$

In practice we have gone a little beyond this limit and this has enabled us to find some good codes which do have cycles of length 4 in their bipartite graphs.

B. Code-rate

The code-rate is directly proportional to the number of roots of $u(x)$. If we let R_{min} represent our minimum desired code-rate then, following equation 11, we can refine our search bound to

$$\mathrm{wt}(\theta(z)) \leq (1 - R_{min})n\,. \tag{14}$$

C. Minimum distance

Let d be the lowest desired minimum distance and let r_θ be the largest number of consecutive non-zero coefficients, taken cyclically mod n, of $\theta(z)$. Then, following the discussion in section II-C, we restrict our search algorithm to those $\theta(z)$ for which

$$r_\theta \geq d - 1 \tag{15}$$

We develop an efficient, but exhaustive recursive tree-search based on the above bounds. The developed search algorithm, Algorithm 1, is initialised by setting V and *index* to $\emptyset$ and -1 respectively.

Algorithm 1 CodeSearch(V, *index*)

Input:
- $R_{min} \Leftarrow$ minimum code-rate of interest
- $d \Leftarrow$ lowest expected minimum distance
- $\delta \Leftarrow$ small positive integer
- $\mathbf{F}(z) \Leftarrow \{f_i(z)\}\ \forall i \in I$ sorted in ascending order of the degree
- $\mathbf{Q}(z) \Leftarrow \{\theta_i(z)\}\ \forall i \in I$

Output: CodesList contains set of codes

```
T ⇐ V
for (i=index+1; i ≤ Size(I); i++) do
  T_prev ⇐ T
  if (Σ_{∀j∈T} deg(f_j(x)) + deg(f_i(x)) ≤ √n + δ) then
    Append i to T
    θ(z) ⇐ Σ_{∀j∈T} θ_j(z)
    if (wt(θ(z)) ≤ (1 − R_min)n and r_θ ≥ d − 1) then
      u(x) ⇐ MS^{-1}(θ(z))
      if (u(x) is non-degenerate) then
        C ⇐ a cyclic code defined by u(x)
        if (C ∉ CodeList) then
          Add C to CodeList
        end if
      end if
    end if
    CodeSearch(T, i)
  end if
  T ⇐ T_prev
end for
```

IV. Code Example and Performance

The code construction method presented in this paper is able to produce, in addition to some new codes, many well-know cyclic codes for example the Difference-Set Cyclic codes and the Euclidean and Projective Geometry codes. Some of the new codes are presented in Table I.

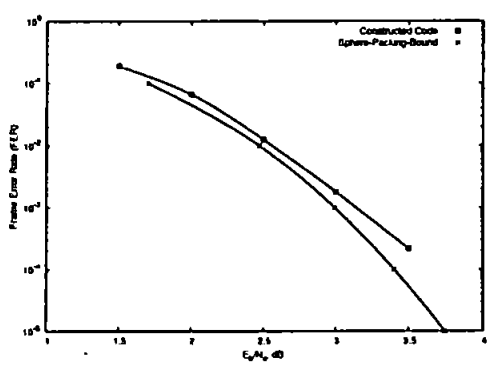


Fig. 1. Frame error performance of the [127, 84] cyclic code

Figure 1 shows the frame-error-rate (FER) performance of the [127, 84] cyclic code. Throughout the paper, it is assume

TABLE I

EXAMPLES OF THE CONSTRUCTED CODES

$[n, k]$	$u(x)$	Minimum distance
$[51,26]^{\dagger}$	$1+x^3+x^6+x^{12}+x^{17}+x^{24}+x^{27}+x^{34}+x^{39}+x^{45}+x^{48}$	10
$[63,44]^{\dagger}$	$1+x^7+x^9+x^{14}+x^{18}+x^{27}+x^{28}+x^{35}+x^{36}+x^{45}+x^{49}+x^{54}+x^{56}$	8
$[93,47]$	$1+x^2+x^8+x^{31}+x^{32}+x^{35}+x^{47}$	8
$[105,53]$	$1+x^4+x^{30}+x^{32}+x^{45}+x^{46}+x^{53}$	8
$[117,72]^{\dagger}$	$1+x+x^2+x^4+x^8+x^{11}+x^{16}+x^{22}+x^{32}+x^{44}+x^{59}+x^{64}+x^{88}$	12
$[127,84]^{\dagger}$	$1+x+x^2+x^4+x^8+x^{16}+x^{32}+x^{55}+x^{59}+x^{64}+x^{91}+x^{93}+x^{109}+x^{110}+x^{118}$	10
$[219,101]$	$1+x^2+x^8+x^{32}+x^{73}+x^{74}+x^{77}+x^{89}+x^{110}+x^{128}+x^{137}$	12
$[255,135]$	$1+x^4+x^{13}+x^{21}+x^{39}+x^{54}+x^{55}+x^{91}+x^{121}+x^{123}+x^{148}+x^{195}$	13
$[255,175]$	$1+x+x^3+x^7+x^{15}+x^{26}+x^{31}+x^{53}+x^{63}+x^{98}+x^{107}+x^{127}+x^{140}+x^{176}+x^{197}+x^{215}$	17
$[341,205]$	$1+x^{29}+x^{87}+x^{92}+x^{94}+x^{114}+x^{122}+x^{156}+x^{202}+x^{203}+x^{213}+x^{217}+x^{234}+x^{257}+x^{273}$	16
$[511,199]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{82}+x^{100}+x^{127}+x^{152}+x^{165}+x^{201}+x^{255}+x^{290}+x^{305}+x^{331}+x^{403}$	19
$[511,259]$	$1+x^{31}+x^{42}+x^{93}+x^{115}+x^{217}+x^{240}+x^{261}+x^{360}+x^{420}+x^{450}+x^{465}$	13
$[819,435]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{127}+x^{204}+x^{255}+x^{409}+x^{511}$	13
$[819,447]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{127}+x^{204}+x^{255}+x^{350}+x^{409}+x^{511}+x^{584}+x^{701}$	16
$[5461,3781]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{76}+x^{127}+x^{153}+x^{255}+x^{307}+x^{511}+x^{578}+x^{615}+x^{754}+x^{776}+x^{1023}+x^{1157}+x^{1196}+x^{1231}+x^{1509}+x^{1553}+x^{2047}+x^{2144}+x^{2315}+x^{2393}+x^{2463}+x^{2730}+x^{2768}+x^{3019}+x^{3107}+x^{3118}+x^{3328}+x^{3802}+x^{4095}+x^{4114}+x^{4289}+x^{4394}+x^{4631}+x^{4787}+x^{4927}$	43

† The resulting parity-check matrix has cycles of length 4.

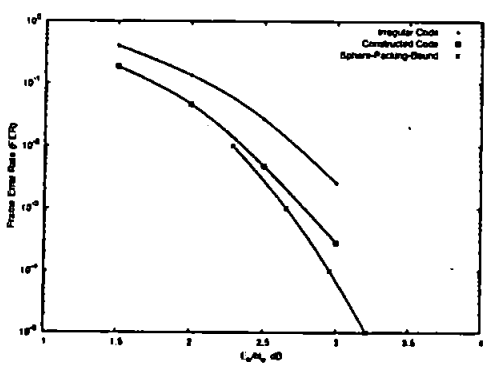


Fig. 2. Frame error performance of the $[255, 175]$ codes

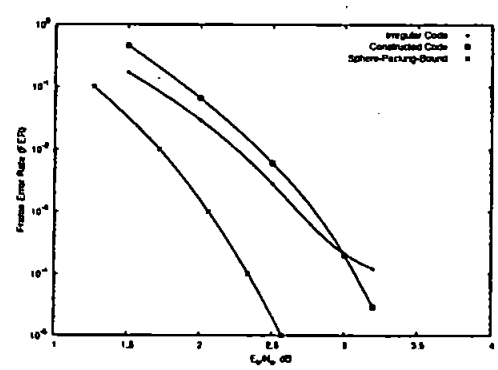


Fig. 3. Frame error performance of the $[341, 205]$ codes

that the codewords are transmitted across a noisy communication channel with BPSK modulation and at the receiving end is the modified Belief-Propagation decoder which approximates the Maximum-Likelihood decoder [8], [9]. The performance of this code is outstanding and at 10^{-3} FER, it is within 0.2dB away from the sphere-packing-bound[1] [10],[11].

Figure 2 shows the performance of two $[255, 175]$ codes: code constructed using our method and irregular computer generated code. We can see that our code, which achieves a coding gain of around 0.4dB compared to the equivalent irregular code, performs approximately within 0.15dB away from the sphere-packing-bound.

Our construction method can produce LDPC codes with high minimum distance and therefore they do not suffer from error-floor. Figure 3 demonstrates that the performance of our $[341, 205]$ code is inferior to the equivalent irregular code in the low signal-to-noise ratio region, however, the irregu-

[1] In this paper, we assume that the sphere-packing-bound has been offset by the information theoretical loss associated with binary transmission.

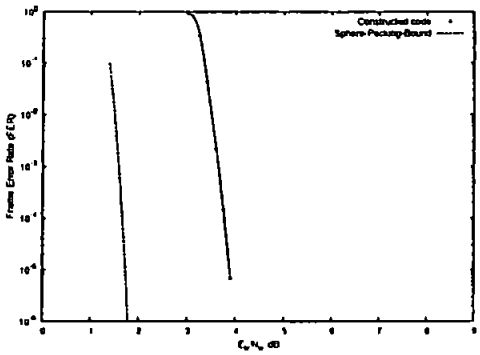


Fig. 4. Frame error performance of the [5461, 3781] code

lar code exhibits early error-floor due to its low minimum-distance.

For long block lengths, our construction method produces LDPC codes with very high minimum distance and as a conseqeunce, the error-floor of the resulting codes is extremely low. However, this low error-floor advantage is counterbalanced by the increase in the signal-to-noise-ratio (SNR) threshold, which is a point at which the FER curve starts to fall rapidly with SNR. This region is commonly known as the waterfall region and the SNR threshold is the starting point of the waterfall region. As an example, consider the [5461, 3781] cyclic LDPC code which has minimum distance of 43 and code-rate of 0.69. Figure 4 demonstrates its FER performance under standard Belief-Propagation iterative decoder with maximum iterations of 100. As seen in this figure, due to its high SNR threshold (approximately at E_b/N_o of 3.0dB), the distance to the sphere-packing-bound is around 1.9dB at 10^{-3} FER. On the other hand, with minimum distance of 43 the approximated error-floor is extremely low.

V. Conclusions

The application of idempotents and Mattson Solomon polynomials in code construction produces a large number of binary cyclic codes which have high minimum distance and code-rate. These codes have sparse parity-check matrix and thus, are applicable as LDPC codes. Due to the cyclic nature, these LDPC codes have n parity-check equations instead of $n-k$ equations as in the case of random LDPC codes. With this extra parity-check equations to iterate with, the performance of the iterative decoder is improved.

In designing cyclic- LDPC codes of length n, our new method allows one to steadily increase the minimum distance of the code by combining additional irreducible factors of z^n-1 which in turn reduces the sparseness of the parity-check matrix. The ability to control the sparseness of the parity-check matrix by trading the code dimension and thus the minimum distance off is an interesting property.

Simulation results have shown that, at short block lengths, our cyclic codes have outstanding performance which are superior to the equivalent irregular LDPC codes. The high minimum distance of these codes ensures the absence of the early error-floor in their performance.

Acknowledgement

The financial assistance from the UK Overseas Research Student Award Scheme is gratefully acknowledged.

References

[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977. ISBN 0 444 85193 3.
[2] S. Roman, *Coding and Information Theory*, Springer-Verlag, 1992.
[3] J. H. van Lint, *Introduction to Coding Theory*, Springer-Verlag, 2nd ed., 1991.
[4] T. Shibuya and K. Sakaniwa, "Construction of cyclic codes suitable for iterative decoding via generating idempotents," *IEICE Trans. Fundamentals*, vol. E86-A, no. 4, 2003.
[5] E. J. Weldon, Jr., "Difference-set cyclic codes," *Bell Syst. Tech. J.*, vol. 45, pp. 1045–1055, Sept. 1966.
[6] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on Finite Geometries," *IEEE Trans. Inform. Theory*, vol. 51, pp. 572–596, Feb. 2005.
[7] J. H. van Lint, *Coding Theory*, Lecture Notes in Mathematics, Springler-Verlag, 1971. ISBN 3 54 005476 6.
[8] C. J. Tjhai, E. Papagiannis, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed, "Improved iterative decoder for LDPC codes with performance approximating to a maximum likelihood decoder." UK Patent Application 0409306.8, Apr. 2004.
[9] E. Papagiannis, M. Ambroze, and M. Tomlinson, "Improved Decoding of Low-Density Parity-Check Codes with Low, Linearly Increased Added Complexity." Submitted to ISIT 2005, Jan. 2005.
[10] C. E. Shannon, "Probability of error for optimal codes in a gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, May 1959.
[11] S. Dolinar, D. Divsalar, and F. Pollara, "Code Performance as a Function of Block Size," *TMO Progress Report*, pp. 42–133, May 1998. Available: http://tmo.jpl.nasa.gov/tmo/progress_report/.

Near Maximum-Likelihood Performance of Some New Cyclic Codes Constructed in the Finite-Field Transform Domain

C. Tjhai, M. Tomlinson, R. Horan, M. Ambroze and M. Ahmed
Fixed and Mobile Communications Research,
University of Plymouth,
Plymouth PL4 8AA,
United Kingdom
email: {ctjhai,mtomlinson,rhoran,mambroze,mahmed}@plymouth.ac.uk

Abstract

It is shown that some well-known and some new cyclic codes with orthogonal parity-check equations can be constructed in the finite-field transform domain. It is also shown that, for some binary linear cyclic codes, the performance of the iterative decoder can be improved by substituting some of the dual code codewords in the parity-check matrix with other dual code codewords formed from linear combinations. This technique can bring the performance of a code closer to its maximum-likelihood performance, which can be derived from the erroneous decoded codeword whose euclidean distance with the respect to the received block is smaller than that of the correct codeword. For $[63, 37]$, $[93, 47]$ *and* $[105, 53]$ *cyclic codes, the maximum-likelihood performance is realised with this technique.*

1. Introduction

Low-density parity-check (LDPC) codes [1],[2] form a class of $[n, k]$ linear block codes, where n is the codeword length and k is the information length, that can approach near capacity performance. The good performance of LDPC codes is attributed to the code representation and the use of an iterative decoder. It is an essential condition that the code representation does not contain more than one parity-check equation checking on the same two or more bit positions, in the other words there is no cycles of length 4 in its underlying parity-check matrix. The avoidance of these short cycles is important to allow convergence of the iterative decoder [3]. The best performance gains to date have been obtained with long LDPC codes, i.e. several thousand bits in length.

There are many applications where short LDPC codes can be potentially useful. Applications such as watermarking, thin data storage, command/control data reporting and packet communications require blocks of data ranging from 32 to 512 bits to be either robustly protected or reliably transmitted. We concentrate on cyclic LDPC codes of similar lengths in this paper. The particular class of cyclic codes we consider are difference set cyclic (DSC) codes [4] and one-step majority logic decodable (OSMLD) codes [5] which have orthogonal parity-check equations on each bit position, thus there are no cycles of length 4. For short block lengths, these cyclic codes* have been shown to outperform the ad-hoc computer design (random) counterpart of the same code-rate and block length [6]. For an $[n, k]$ LDPC code, the ad-hoc computer design code has $n - k$ parity-check equations but the cyclic code has n parity-check equations that can be used by the iterative decoder. Consequently, cyclic codes exhibit better convergence than the random LDPC codes when iteratively decoded.

In this paper, we present a modified Belief-Propagation iterative decoder that can perform near maximum-likelihood (ML) performance for binary transmission over the additive-white-Gaussian-noise (AWGN) channel. It is also shown that, for certain cyclic codes, the modified iterative decoder can achieve ML performance.

The organisation of this paper is as follows. Section 2 gives a brief review on binary cyclic code construction method in the finite-field transform domain. We shall deal exclusively with binary cyclic codes throughout this paper. Section 3 introduces the idea of a more-likely codeword and its relationship to ML and iterative decoders. We present modification to the iterative decoder in section 4 and

*We will refer the OSMLD and DSC codes as cyclic codes from this point onwards

some simulation results of the modified decoder are presented in section 5. Section 6 contains the conclusions.

2. Finite-Field Transform Domain Construction of Cyclic Codes

There are relatively few OSMLD and DSC codes. As shown in [7], we have extended these codes by a construction method that works in the finite-field transform domain, which is also known as the Mattson-Solomon domain. We briefly review the construction method in this section.

Let n be a positive odd integer and GF(2^m) be the splitting field for $1 + x^n$ over GF(2). We assume that α is a generator for GF(2^m) and $T_n(x)$ is a polynomial with coefficients in GF(2^a) and degree $\leq n - 1$. Let us denote $\mathcal{F} = \{f_1(z), f_2(z), \ldots, f_t(z)\}$, where $f_i(z) \in T_1(z)$ is an irreducible polynomial, such that $\prod_{i \leq i \leq t} f_i(z) = 1 + z^n$. For each $f_i(z)$, there is a corresponding primitive idempotent[†], denoted as $\theta_i(z)$, which can be obtained as follows:

$$\theta_i(z) = \frac{z(1+z^n)f_i'(z)}{f_i(z)} + \delta(1+z^n) \qquad (1)$$

where $f_i'(z) = \frac{d}{dz} f_i(z)$, $f_i'(z) \in T_1(z)$ and the integer δ is:

$$\delta = \begin{cases} 1 & \text{if } \deg(f_i(z)) \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases}$$

where $\deg(a(x))$ represents the degree of the polynomial $a(x)$.

Let $a(x) \in T_m(x)$, the finite-field transform or Mattson-Solomon (MS) polynomial of $a(x)$ is:

$$A(z) = \mathrm{MS}(a(x)) = \sum_{j=0}^{n-1} a(\alpha^{-j}) z^j \qquad (2)$$

$$a(x) = \mathrm{MS}^{-1}(A(z)) = \frac{1}{n} \sum_{i=0}^{n-1} A(\alpha^i) x^i \qquad (3)$$

where $A(z) \in T_m(z)$.

Let $\mathcal{I} \subseteq \{1, 2, \ldots, t\}$, we define $f(z) = \prod_{i \in \mathcal{I}} f_i(z)$ and $\theta(z) = \sum_{i \in \mathcal{I}} \theta_i(z)$, where $f(z), \theta(z) \in T_1(z)$. Let us define a binary polynomial $u(x) = \mathrm{MS}(\theta(z))$. Since the MS polynomial of a binary polynomial is an idempotent and vice-versa [8], $u(x)$ is an idempotent with coefficients in GF(2). If we write $u(x) = u_0 + u_1 x + \ldots + u_{n-1} x^{n-1}$ then, from (3)

$$u_i = \frac{1}{n} \theta(\alpha^i), \quad \forall i \in \{0, 1, \ldots, n-1\}. \qquad (4)$$

The idempotent $u(x)$ can be used to describe an $[n, k]$ binary cyclic code which has a parity-check polynomial, $h(x)$, of degree k and a generator polynomial, $g(x)$, of degree $n - k$. The polynomial $h(x)$ is a divisor of the idempotent $u(x)$, i.e. $(u(x), 1 + x^n)^{\ddagger} = h(x)$ and $u(x) = m(x)h(x)$ where $m(x)$ contains the repeated factors and/or non-factors of $1 + x^n$.

From the information above, we can summarise that:

1) The weight of $u(x)$ is equal to the number of nth roots of unity which are roots of $f(z)$. Note that for $0 \leq i \leq n-1$, $\theta(\alpha^i) = 1$ if and only if $f(\alpha^i) = 0$ and from (4), $u_i = 1$ if and only if $\theta(\alpha^i) = 1$. In the other words, $u_i = 1$ precisely when $f(\alpha^i) = 0$, giving $\mathrm{wt}(u(x))^{\S} = \deg(f(z))$. Clearly, $\mathrm{wt}(u(x)) = \sum_{i \in \mathcal{I}} \deg(f_i(z))$.

2) Since $\theta(z) = \mathrm{MS}(u(x))$, the number of zeros of $u(x)$ which are roots of unity is clearly $n - \mathrm{wt}(\theta(z))$. This determines the code dimension.

In general, $\mathrm{wt}(u(x))$ is much lower than $\mathrm{wt}(h(x))$ and as such, we can derive a low-density parity-check matrix from $u(x)$ and apply iterative decoding on it. The parity-check matrix of the resulting code consists of the n cyclic shifts of $x^{\deg(u(x))} u(x^{-1})$. Since the idempotent $u(x)$ is orthogonal on each bit position, the resulting LDPC code has no cycles of length 4 in the bipartite graph and the true minimum-distance, d_{min}, of the code is simply $\mathrm{wt}(1 + u(x))$, see [9, Theorem 10.1] for the proof.

Table 1 shows some examples of cyclic codes derived from this technique. From Table 1, it is clear that our technique can also be used to construct the well-known OSMLD and DSC codes.

3. More Likely Codewords in Relation to ML and Iterative Decoders

Realising an optimum decoder for any coded system is NP-complete [10]. For general $[n, k]$ binary linear codes, the optimum decoding complexity is proportional to $\min\{2^k, 2^{n-k}\}$. Due to this

[†] A binary polynomial, $e(x)$, is an idempotent if the property of $e(x) = e(x)^2 = e(x^2) \bmod 1 + x^n$ is satisfied.

[‡] (a, b) denotes the greatest common divisor of a and b.

[§] $\mathrm{wt}(a(x))$ denotes the weight of polynomial $a(x)$.

Table 1. Some examples of the constructed cyclic codes

$[n,k]$	$u(x)$	d_{min}
$[21,11]$	$1+x^2+x^7+x^8+x^{11}$	6
$[63,37]$	$1+x^1+x^3+x^7+x^{15}+x^{20}+x^{31}+x^{41}$	9
$[73,45]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{36}+x^{54}+x^{63}$	10
$[93,47]$	$1+x^3+x^9+x^{21}+x^{28}+x^{45}+x^{59}$	8
$[105,53]$	$1+x^7+x^8+x^{21}+x^{23}+x^{49}+x^{53}$	8
$[255,175]$	$1+x+x^3+x^7+x^{15}+x^{26}+x^{31}+x^{53}+x^{63}+x^{98}+x^{107}+x^{127}+x^{140}+x^{176}+x^{197}+x^{215}$	17
$[341,205]$	$1+x^1+x^3+x^7+x^{15}+x^{31}+x^{54}+x^{63}+x^{98}+x^{109}+x^{127}+x^{170}+x^{197}+x^{219}+x^{255}$	16
$[511,199]$	$1+x+x^3+x^7+x^{15}+x^{31}+x^{63}+x^{82}+x^{100}+x^{127}+x^{152}+x^{165}+x^{201}+x^{255}+x^{296}+x^{305}+x^{331}+x^{403}$	19
$[511,259]$	$1+x^{31}+x^{42}+x^{93}+x^{115}+x^{217}+x^{240}+x^{261}+x^{360}+x^{420}+x^{450}+x^{465}$	13

complexity, the optimum decoder can only be realised for very short, very high-rate or very low-rate codes. An ML decoder is the optimum decoder in terms of minimising the frame-error-rate (FER). An ML decoder will output a codeword that has the closest euclidean distance [5] to the received block.

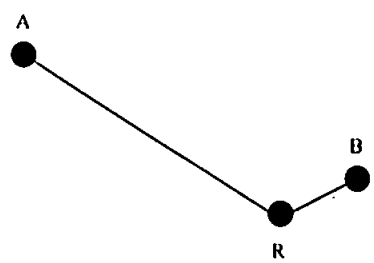


Figure 1. ML decision criterion

The iterative decoder is a suboptimal decoder approximating ML performance. In decoding LDPC codes, the Belief-Propagation (BP) iterative decoder can produce a codeword that is not identical to the transmitted codeword. This is illustrated by the two-dimensional representation of the ML decision criterion shown in Figure 1. Points R and A represent the received block and transmitted/correct codeword respectively. The point B represents a codeword whose euclidean distance with the respect to R is smaller than that of A. If the iterative decoder outputs codeword B then a decoding error is produced, but an ML decoder will also make an error. We classify codeword B as a more likely (mrl) codeword [11],[12]. By counting the number of mrl codewords produced in a simulation, we can derive an mrl-FER curve. A similar technique has been used by Dorsch [13], but the metric was based on the hamming distance rather than the euclidean distance, i.e. hard-decisions rather than soft-decisions.

The significance of the mrl codewords is that an ML decoder either outputs correct codewords or mrl codewords. The percentage of mrl codewords output from the iterative decoder gives us a performance indication of how close the iterative decoder is from the ML decoder for the same code. The mrl-FER provides the lower-bound on the ML performance of a code in comparison to the Maximum-Likelihood-Asymptote (MLA) which provides the upper-bound.

4. Improved Belief-Propagation Decoder

For the $[63,37]$ cyclic code, it has been noticed that the standard BP decoder produces many codewords that are neither correct nor mrl. The number of incorrect codewords is much larger than the number of mrl codewords output. Based on these findings and the fact that every codeword will satisfy all 2^{n-k} parity-check equations, we should be able to improve the performance of the BP decoder by extending the number of parity-check equations in the parity-check matrix, denoted as **H**. However, this is likely to be true if the extended parity-check matrix have low-density and does not contain many short cycles.

For any linear codes, additional parity-check equations can be formed from the linear combinations of the equations in **H**. These additional parity-check equations form the high weight codewords of the dual code and appending them to **H** will introduce many short cycles.

The proposed modified BP decoder does not extend the number of parity-check equations in **H**. Instead, we generate a set of parity-check equations,

denoted as $\mathbf{H}^e$, by taking the linear combinations of those equations in $\mathbf{H}$. A subset of $\mathbf{H}^e$ is substituted into $\mathbf{H}$ resulting in a modified parity-check matrix, labelled as $\hat{\mathbf{H}}$. The overall procedures is described in Algorithm 1 [14]. Note that the selection of the parity-check equations may be made on a random basis or may correspond to a predetermined sequence.

Algorithm 1 Modified Belief-Propagation Iterative Decoder

Input:
- $\mathbf{r} \Leftarrow$ received vector
- $\mathbf{H} \Leftarrow$ original parity-check matrix of the code
- $\mathbf{H}^e \Leftarrow$ a set of parity-check equations not in $\mathbf{H}$
- $\mathcal{T} \Leftarrow$ number of trials
- $\psi \Leftarrow$ number of selections, $\psi \leq n-k$

Output: a codeword with the minimum euclidean distance

```
Perform BP decoding and d_0 ⇐ decoded output.
d_E(d_0, r) ⇐ euclidean distance between d_0 and r.
d' ⇐ d_0 and d_E^min ⇐ d_E(d_0, r)
for τ = 1 to T, do
    for i = 1 to maximum number of iterations, do
        Pick ψ parity-check equations from H^e.
        Substitute them into H to generate Ĥ.
        Based on Ĥ, perform the check nodes (horizontal) and bit nodes (vertical) processing as in standard BP algorithm.
        d_τ ⇐ denote the decoded output.
        d_E(d_τ, r) ⇐ euclidean distance between d_τ and r.
        if (d_τ H^T = 0) then BREAK
    end for
    if (d_E(d_τ, r) < d_E^min) and (d_τ H^T = 0) then
        d' ⇐ d_τ
        d_E^min ⇐ d_E(d_τ, r)
    end if
end for
Output d'.
```

5. Simulation Results

In this section, we present simulation results of the modified BP decoder for some cyclic codes designed using the approach discussed in section 2. The selection of the parity-check equations is made on a random basis. It is assumed that the simulation system employs BPSK modulation mapping the symbols 0 and 1 to -1 and $+1$ respectively.

Figure 2 shows the FER performance of the [63, 37] cyclic code. It is shown that the modified BP decoder, provided enough substitutions and trials are used, can achieve the ML performance as indicated by the mrl-FER and the FER of the modified BP decoder that produce the same curve. Compared to the standard BP decoder, at a FER of 10^{-3} a gain of approximately 0.9 dB is obtained by using the modified BP decoder. In addition, it can be seen that, at a FER of 10^{-3}, the performance of the code is within 0.4 dB of the sphere-packing-bound [15],[16] for a code of length 63 and code-rate of 0.587 after allowing for the coding loss attributable to binary transmission. Table 2 shows how close the performance of the modified BP decoder is to the ML decoder. With the standard BP decoder more than 50% mrl codewords are found in the low signal-to-noise ratio (SNR) region, but in the moderate SNR region we can only find a few mrl codewords. With just single substitution and 50 trials, the modified BP decoder is able to increase the percentage of mrl codewords found to be higher than 50%. ML performance is achieved with 8 substitutions and 300 trials.

Table 2. Percentage of mrl codewords against E_b/N_o (dB) for the [63, 37] cyclic code

Standard BP decoder						
E_b/N_o	1.5	2.0	2.5	3.0	3.5	4.0
%	73	41	27	23	16	9
Substitutions: 1, Trials: 50						
E_b/N_o	1.5	2.0	2.5	3.0	3.5	4.0
%	90	94	90	82	74	61
Substitutions: 8, Trials: 300						
E_b/N_o	1.5	2.0	2.5	3.0	3.5	4.0
%	100	100	100	100	100	100

Figure 3–5 show the FER performance of the [93, 47], [105, 53] and [341, 205] cyclic codes respectively. Both of the [93, 47] and [105, 53] codes achieve ML performance with the modified BP decoder. For the [341, 205] code, the modified BP decoder produces a gain of approximately 0.4dB with the respect to the standard BP decoder. Due to its high minimum-distance and code-length, there are very few mrl codewords observed.Table 3 summarises, at the FER of 10^{-3}, the amount of gain obtained with the modified decoder with the respect to the standard BP decoder and the distance from the sphere packing bound after allowing binary transmission loss.

6. Conclusions

Construction of cyclic LDPC codes using idempotents and MS polynomials can produce a large num-

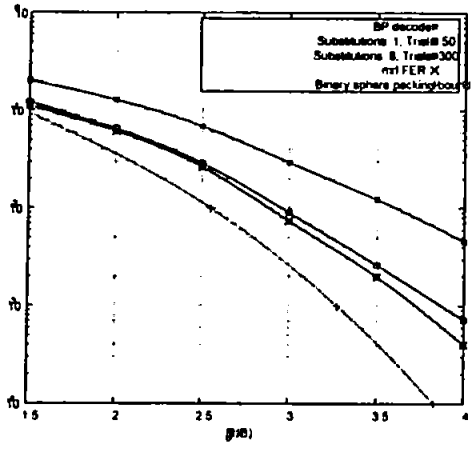


Figure 2. FER performance of the [63, 37] cyclic code.

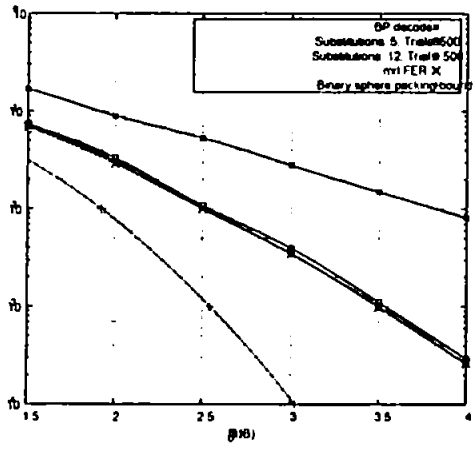


Figure 4. FER performance of the [105, 53] cyclic code.

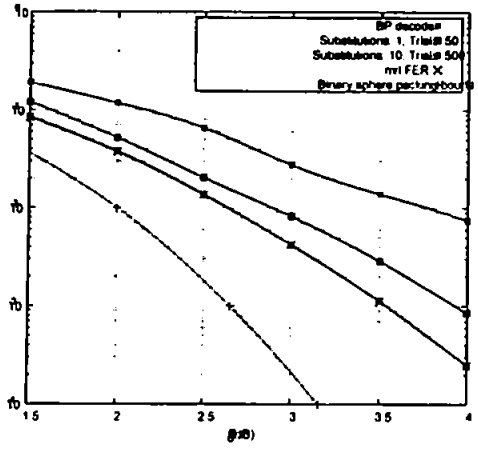


Figure 3. FER performance of the [93, 47] cyclic code.

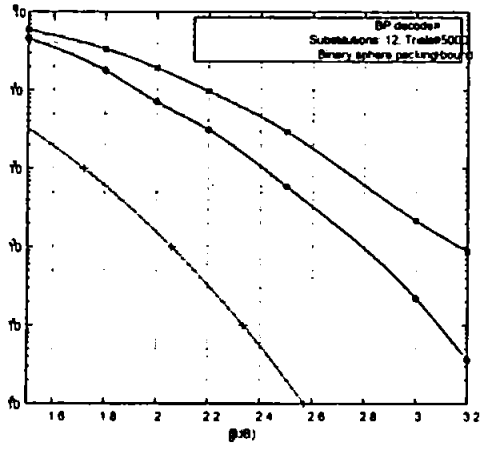


Figure 5. FER performance of the [341, 205] cyclic code.

ber of cyclic codes that are free from cycle of length 4. Some of the codes are already known such as the DSC and OSMLD codes, but others are new. An important feature of this approach is the ability to increase the d_{min} of the codes by taking into account additional irreducible factors of $1 + z^n$ and so steadily decrease the sparseness of the parity-check matrix. As an example, consider that we want to design a cyclic code of length 63. If we let $f(z) = 1 + z + z^6$, we obtain a $(63, 31)$ cyclic code with d_{min} of 7. Now, if the irreducible polynomial $1 + z + z^2$ is also taken into account, the resulting cyclic code is the $[63, 37]$ code which has d_{min} of 9. The row or column weight of the parity-check matrices for the former and latter codes are 6 and 8 respectively.

Table 3. Performance gain with the respect to BP decoder and distance from the sphere-packing-bound at 10^{-3} FER

Codes	Gain[a]	SPB[b]
[63, 37]	0.9 dB	0.4 dB
[93, 47]	1.1 dB	0.8 dB
[105, 53]	2.0 dB	0.9 dB
[341, 205]	0.4 dB	0.7 dB

[a] Coding gain with the respect to BP decoder.

[b] Distance from the sphere-packing-bound offset by binary transmission loss.

By substituting the parity-check equations in the parity-check matrix with other codewords of the dual code derived from their linear combinations, the performance of the BP decoder can be improved. For the [63, 37], [93, 47] and [105, 53] cyclic codes, the modified BP decoder has been shown to achieve ML performance.

Although the substitution method introduces cycles of length 4, these cycles do not pose a lasting negative effect on the iterative decoder. By substituting at every iteration, the effect of these short cycles is broken and simulation results have shown

that this can improve the decoding performance.

Acknowledgement

This work was partly funded by the Overseas Research Students Award Scheme.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Comm.*, vol. 50, pp. 406–414, Mar. 2002.

[4] E. J. Weldon, Jr., "Difference-set cyclic codes," *Bell Syst. Tech. J.*, vol. 45, pp. 1045–1055, Sept. 1966.

[5] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983. ISBN 0 13 283796 X.

[6] R. Lucas, M. P. C. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic decodable codes based on belief propagation," *IEEE Trans. Comm.*, vol. 46, pp. 931–937, June 2000.

[7] R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed, "A Finite-Field Transform Domain Construction of Binary Low-Density Parity-Check Codes," to be presented in ITW 2005, New Zealand, 2005.

[8] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977. ISBN 0 444 85193 3.

[9] W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. MIT Press., 1972.

[10] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, pp. 384–386, May 1978.

[11] M. A. Ambroze, M. Tomlinson, M. Ferrari, and S. Bellini, "Improving iterative product code performance by limited search list decoding," *8th International Workshop on Signal Processing for Space Communications, Catania, Italy*, pp. 129–134, Sep. 2003.

[12] E. Papagiannis, M. A. Ambroze, and M. Tomlinson, "Approaching the ml performance with iterative decoding," *International Zurich Seminar on Communications, Zurich, Switzerland*, pp. 220–223, Feb. 2004.

[13] B. G. Dorsch, "A decoding algorithm for binary block codes and J-ary output channels," *IEEE Trans. Inform. Theory*, vol. 20, pp. 391–394, May 1974.

[14] C. J. Tjhai, E. Papagiannis, M. Tomlinson, M. A. Ambroze, and M. Z. Ahmed, "Improved iterative decoder for LDPC codes with performance approximating to a maximum likelihood decoder." UK Patent Application 0409306.8, Apr. 2004.

[15] C. E. Shannon, "Probability of error for optimal codes in a gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, May 1959.

[16] S. Dolinar, D. Divsalar, and F. Pollara, "Code performance as a function of block size," *TMO Progress Report*, pp. 42–133, May 1998. Available: http://tmo.jpl.nasa.gov/tmo/progress_report/.

PERFORMANCE EVALUATION OF LDPC CODES FOR PATTERNED MEDIA

I. T. Ntokas[1], P. W. Nutter[1], B. K. Middleton[1], C. J.Tjhai[2] and M. Z. Ahmed[2]
(1) School of Computer Science, The University of Manchester, Manchester, UK
(2) School of Computing, Communication and Electronics, University of Plymouth, Plymouth, UK

Introduction

This paper examines the application of low-density-parity-check (LDPC) codes [1] to patterned media storage systems; two areas that have received considerable attention. Recent studies have shown that LDPC codes outperform turbo-codes and are considered as the state-of-the-art in error correction codes (ECC). Patterned media are widely seen as a candidate for ultra high-density recording [2], hence it is interesting to investigate the performance of LDPC codes in this application. We have developed an advanced model of the read channel that allows the investigation of data recovery techniques applicable to patterned media systems. Powerful LDPC codes and their efficient belief-propagation decoder [3] have been incorporated into the system to improve the data recovery process. By comparing the system performance with and without the LDPC codes, we show that LDPC-based error correction scheme can achieve a significant bit-error-rate (BER) gain. We also investigate what BER gain can be achieved by utilizing a soft-input-soft-output (SISO) run-length-limited (RLL) decoder as opposed to the conventional hard output RLL decoder. A final aim of this paper is to analyze the types of errors that occur in the system in relation to the convergence of the iterative decoder.

Channel Simulation

Figure 1 illustrates the data recovery channel simulated.

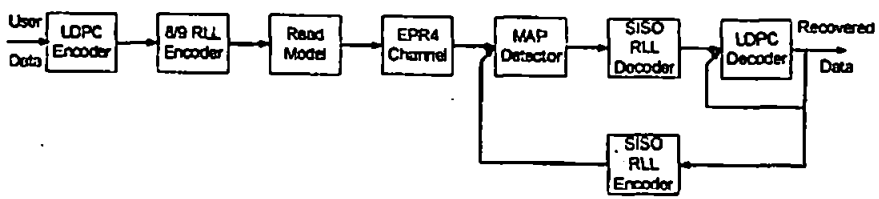


Fig. 1 Read Channel for Magnetic Patterned Media

The LDPC code is an irregular code constructed in such a way to maximise the girth of the corresponding tanner graph. It is constructed with zigzag parity pattern [4] which results in very low encoding complexity. We aim to test the effectiveness of current LDPC codes of different code rates. The 8/9 RLL(0,4/4) modulation encoder is the most appropriate for the system due to its relatively high code rate and since its decoder can be easily converted to SISO decoder which is essential for iterative decoding. The readout signal from patterned media systems is evaluated as outlined in [5]. In the following analysis a GMR head of sensor width 80nm and length 8nm, with shield-to-shield spacing 32nm has been used. The patterned medium comprises square islands of length 25nm, with period and track pitch of 50nm. The EPR4 target has been used and additive white Gaussian noise (AWGN) and media jitter have been considered. Three forms of RLL decoder have been investigated: the conventional hard decision RLL, the soft input hard output RLL that was simulated following the principles of the Viterbi algorithm and the SISO RLL. The SISO RLL encoder has been added in order to feed the MAP detector with more information and further improve the performance of the read channel.

Results-Discussion

Figure 2 illustrates BER performance against signal-to-noise-ratio (SNR) plots for different channel configurations. The BER gain, which is achieved by using the current LDPC scheme, extends 3 dB at a BER of 10^{-4} over the uncoded system. Significant improvement is achieved by increasing the number of LDPC iterations to 5, and an additional slight improvement is observed by utilizing 5 system iterations. Jitter noise degrades the performance of the channel as expected but the use of LDPC contributes to the correction of the errors. A slight improvement in performance is achieved when using the SISO RLL instead of the hard RLL decoder (not illustrated) in terms of symbol-error-rate (SER) against SNR. One symbol consists of 9 bits for the rate 8/9 RLL code and grouping the data in this manner provides a meaningful way of measuring the RLL coding gain. Finally by measuring the Euclidean distances between codewords generated: from the LDPC encoder using random data, at the output of the LDPC decoder, and the re-encoded user data at the output of the LDPC decoder, the types of errors that occur in the system are identified as being due to non convergence of the iterative decoder.

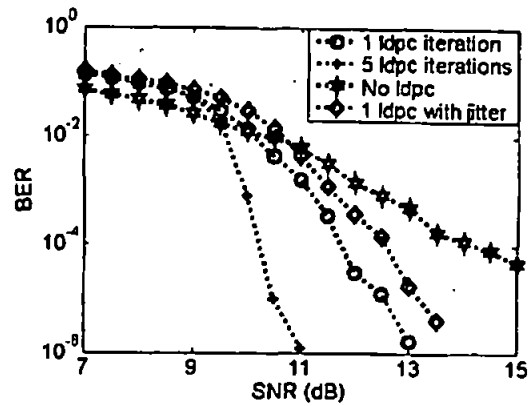


Fig. 2 Channel BER v. SNR

References

[1] R. G. Gallager, *Low Density Parity Check Codes*, MIT Press, USA, 1963.
[2] R. L. White, et. al., IEEE Trans. Mag., vol. 33, pp 990-995, 1997.
[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, USA, 1988.
[4] L. Ping et. al., IEE Electronics Letters, vol. 35, pp. 38–39, 1999.
[5] P. W. Nutter et. al., IEEE Trans. Mag., vol. 40, pp. 3551-3557, 2004.

0-7803-9009-1/05/$20.00 ©2005 IEEE