ITERATIVE CHASE DECODING OF ALGEBRAIC GEOMETRIC CODES

HU WENGUANG

NATIONAL UNIVERSITY OF SINGAPORE

2005

ITERATIVE CHASE DECODING OF ALGEBRAIC GEOMETRIC CODES

HU WENGUANG

(B.Eng.,THU, P.R.CHINA)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgement

I wish to thank Dr. Marc Andre Armand and Dr. Mehul Motani for their patient and invaluable guidance throughout the courses of my project and thesis. I am grateful for, and much enlightened from the numerous discussions with them. I sincerely wish them all the best in all their future endeavors.

I would also like to thank my friend Cai Feng for the fruitful discussions which helped me to solve the some problems in this thesis. My gratitude also goes to National University of Singapore who provides me an excellent research environment and grants me the use of the facilities and scholarship.

I am also grateful to my wife,my parents and my elder brother, who always support and encourage me, without which even the easiest thing would not have been possible for me.

Contents

Abbreviations

AG Code: Algebraic Geometric Code

AWGN Channel: Additive White Gaussian Noise Channel

BCH Code: Bose-Chaudhuri-Hocquenghem Code

BER: Bit Error Rate

BPSK: Binary Phase Shift Keying

BTC: Block Turbo Code

CTC: Convolutional Turbo Code

LLR: Log-Likelihood-Ratio

MAP: Maximum a-Posteriori Probability

MDS Codes: Maximum Distance Separable Codes

MLD: Maximum Likelihood Decoding

OSD: Ordered Statistics Decoding

PBMA: Parallel implementation of Berlekamp-Massey Algorithm

SISO: Soft-Input Soft-Output

SNR: Signal-to-Noise Ratio

RS Code: Reed-Solomon Code

List of Figures

List of Tables

Summary

Error control coding is designed to solve the problem of reliable transmission of information over a noisy channel. BCH codes and Reed-Solomon codes are two kinds of widely used error control codes. In the last two decades, various ideas of algebraic geometry are used in the construction of error control codes and their decoding algorithms. These codes are usually called algebraic geometric codes. Algebraic geometric codes could be considered as a generalization of Reed-Solomon codes. The introduction of turbo codes by Berrou, Glavirux and Thitimajshima also has considerably modified our approach of channel coding in the last ten years. Later the general concept of iterative soft-input soft-output decoding has been extended to block turbo codes(BTC) by Pyndiah. Both BCH codes and Reed-Solomon codes have been used in the block turbo decoding scheme. In this thesis, one-point algebraic geometric codes are used as the component codes of block turbo codes.

This thesis is intended to investigate the soft-decision decoding algorithms of algebraic geometric codes to achieve good performance of digital communications in both AWGN channel and Rayleigh fading channels.

The first part presents the fundamental theory of algebraic geometry, which is important in the construction of AG codes and their decoding algorithms. Codes defined over Hermitian curves and Klein quartic curves are selected as the example of functional AG codes and residual AG codes respectively. Their encoding methods and code parameters are introduced. In addition, the relationship between functional Hermitian codes and generalized Reed-Solomon codes are shown.

In the second part, we use the Chase algorithm with an inner hard-decision decoder, which is a parallel implementation of Berlekamp-Massey algorithm(PBMA),

in the soft-decision decoding of one-point AG codes over Klein quartic curves. The simulations in AWGN channel and Rayleigh fading channel shown that the decoding scheme can achieve remarkable coding gains compared with the hard-decision PBMA decoder.

In the last part, we presented the iterative Chase decoding algorithm for the product codes constructed by one-point AG code over Klein quartic curves. Since the iterative Chase decoder compute the extrinsic information with respect to bit, the product codes are represented in binary codes. Because the AG codes used as component codes are defined over $\mathbb{F}_8$, two concatenation method can be utilized. One is bit concatenation, the other is symbol concatenation. We proposed the codeword validation step used in the decoding algorithm to mitigate the restriction of the PBMA hard-decision decoder of AG codes. The simulation results show that the block turbo code constructed by AG codes can achieve good performance comparable to those of the block turbo codes constructed by BCH codes and Reed-Solomon codes in both AWGN channel and Rayleigh fading channel.

Chapter 1

Introduction

1.1 Error Control Coding

The objective of data transmission is to transfer information from the source through a physical channel to the destination reliably. A canonical digital communication/storage system can be represented by a block diagram shown in Figure 1.1.

Error control coding provides a systematic way of adding redundancy to a message before transmitting it. As a result, even a somewhat corrupted message was received, the redundancy in the message enables the receiver to figure out the original message that the sender intended to transmit. Error control coding is always implemented in the channel encoder and channel decoder modules of a typical digital communication/storage system. As shown in figure 1.1, the channel encoder and decoder in conjunction with the physical channel create a error control channel which provides a reliable data transmission between the data source and the destination.

We will define several basic notations concerning error control coding, which would be necessary to our discussions in all chapters of this thesis.

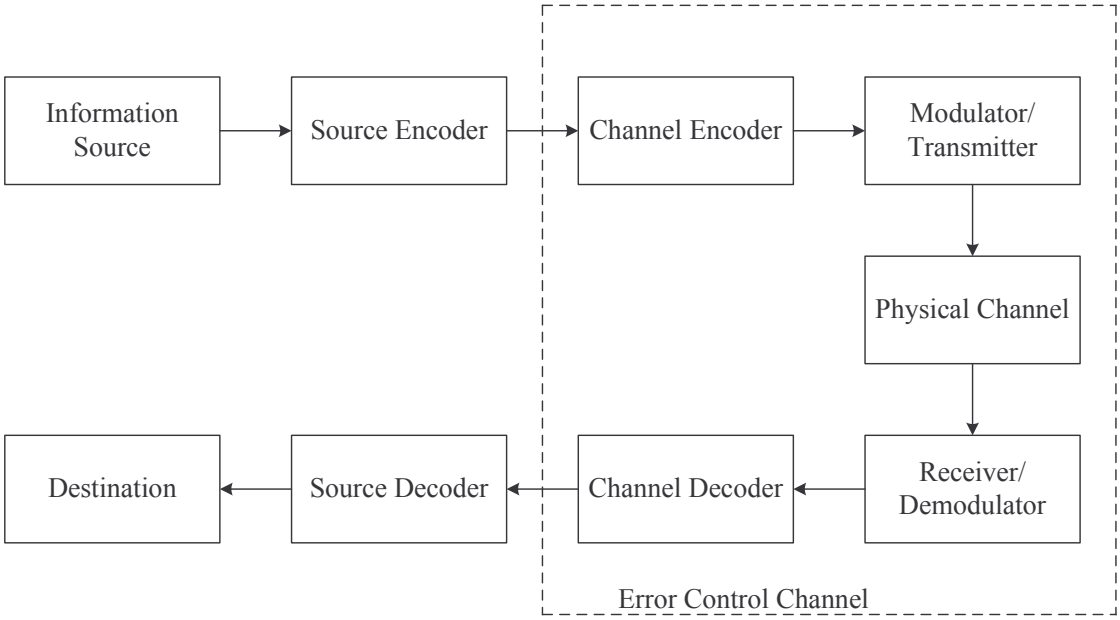

Figure 1.1: Block diagram of a digital communication system

- **Encoding**. An *encoding* function with parameter k and n is a function $E : \Sigma^k \rightarrow \Sigma^n$, which maps a message consisting of k symbols over some alphabet Σ into a longer, redundant string of length n over Σ. The encoded string is called a *codeword*.

- **Decoding**. The receiver gets a possibly distorted copy of the transmitted codeword, and need to figure out the original message which the sender intend to transmit. The *decoding* function $D : \Sigma^n \rightarrow \Sigma^k$ maps the strings of length n, which are the noisy code word, to strings of length k(i.e., what the decoder thinks were the transmitted messages).

- **Code Rate**. The ratio of the number of information symbols to the length of the codeword k/n is called the *Code Rate*. It is a measure of the amount of redundancy added by the encoding.

- **Hamming Distance**. The *Hamming distance* of between two codewords is

the number of coordinates at which they differ. In the rest of this thesis, distance are all referred to Hamming distance. The *minimum distance* of a code is the smallest distance between two distinct codewords. The minimum distance is of great importance to determine the error correcting ability of a code.

The frequently used error control codes presently can be grouped into 2 groups, the convolutional codes and the linear block codes. In this thesis, all the codes are linear block codes. Let's explain several basic concepts of linear block codes first.

A linear block code $\mathbf{C}$ defined in $\mathbb{F}_q$ of with block length n is a linear subspace of $\mathbb{F}_q^n$. Generally, $\mathbf{C}$ has q^k elements, where k is the dimension of the code. As a standard notation, we refer to a q-ary linear block code of block length n, dimension k and minimum distance d as an (n, k, d) code. Linear block codes can be divided into two classes, systematic codes and non-systematic codes. For systematic codes, the redundant symbols are appended to information symbols to obtain a codeword. Such an encoding is said to be systematic. In practical, the information symbols always appear in the first or last k positions of a codeword. The remaining $n - k$ symbols in a codeword are obtained by some function of the information symbols, and can provide the redundancy to be used for error correction.

An (n, k, d) linear block code can be specified in one of two equivalent ways: using the *generator matrix* or the *parity check matrix*.

- An (n, k, d) linear block code can always be described as the set $\{\mathbf{G}\overline{x} : \overline{x} \in \mathbb{F}_q^k\}$ for an $n \times k$ matrix $\mathbf{G}$. Such a $\mathbb{G}$ is called a *generator matrix* of $\mathbf{C}$.

- An (n, k, d) linear block code can also be described as the set $\{\overline{y} : \overline{y} \in \mathbb{F}_q^n$ and $\mathbf{H}\overline{y} = \mathbf{0}\}$ for an $(n - k) \times n$ matrix $\mathbf{G}$. Such a $\mathbf{H}$ is called a *parity check matrix* of $\mathbf{C}$.

Consider a received vector $\overline{r} = \overline{c} + \overline{e}$, where $\overline{c}$ is a valid codeword and $\overline{e}$ is an error pattern introduced by noisy channel. The matrix product $\overline{r}\mathbf{H}$ is the *syndrome vector* of the received vector. It is obvious that $\overline{r}\mathbf{H}$ equals to $\overline{e}\mathbf{H}$. The syndrome vector is solely the function of the error pattern $\overline{e}$ and the parity check matrix $\mathbf{H}$ and is independent of the transmitted codeword $\overline{c}$.

One class of the widely studied and used linear block codes are Reed-Solomon codes. We will introduce some important properties of Reed-Solomon codes first.

1.1.1 Reed-Solomon Codes

Reed-Solomon codes were first described in 1960s by I. S. Reed and G. Solomon.[17] Reed-Solomon codes are defined first as a special case of BCH codes. However, Reed-Solomon codes display some properties that are not found in any of the other BCH codes. Particularly, Reed-Solomon codes are *maximum distance separable*(MDS). They present the best code rate for a given minimum distance. Reed-Solomon codes' initial definition focus on the evaluation of polynomials over the elements in a finite field. This approach has been generalized to an algebraic-geometric definition involving rational curves. There are also some modified Reed-Solomon codes, such as generalize Reed-Solomon codes, punctured Reed-Solomon codes and extended Reed-Solomon codes. Reed-Solomon codes are widely used both in military and commercial. A shortened pair of cross interleaved Reed-Solomon codes provide error control for the digital audio disc. They are also used for the data transmission and communications of man-made satellites.

Reed-Solomon codes are decoded up to half their minimum distance by first finding the error positions as zeros of a polynomial, which is known as the error-locator polynomial. If the error positions are known and their numbers is strictly smaller than half of the minimum distance, error values can be obtained by solving

linear equations involving syndromes. Berlekamp-Massey algorithm is one of the most efficient hard-decision decoding algorithms for Reed-Solomon codes.

1.1.2 Algebraic Geometric Codes

From the theoretical view of error control coding, a significant research objective is to construct asymptotically good codes, whose parameters could achieve the Gilbert-Varshamov lower bound. This bound is defined about the relationship of the code rate, code dimension, minimum distance and the code alphabet. In 1982, M. A. Tsfasman, S. G. Vlådut and T. Zink [18] showed that there exists asymptotically good sequence of geometric Goppa codes satisfying the Tsfasman-Vlådut-Zink bound. This bound is better than the Gilbert-Varshamov bound when the codes are defined over alphabets of size $q \geq 49$. This is a truly remarkable achievement of algebraic geometric codes. In other words, algebraic geometric codes have advantages over the commonly used Reed-Solomon codes in term of the codes parameters. It is possible to construct algebraic geometry codes that have better code rates and error correction capabilities.

However, the use of algebraic-geometric codes is hindered by two significant difficulties. The first difficulty is the abstract nature of the concepts behind the AG codes. The second difficulty is the greater complexity of the decoder for AG codes compared to the Reed Solomon codes decoder. From the end of 1980s, some decoding algorithms of AG codes were provided, and most of them were generalized from decoding algorithms of Reed-Solomon codes. Similar to the decoding algorithm of Reed-Solomon codes, AG codes determine the error positions by finding the error-locator functions on curves. The resulting basic algorithm can decode up to half the designed minimum distance minus the genus of the underlying curve of the AG codes. Later, a technique called *major voting of unknown syndromes*

makes an algorithm which is able to decodes up to half the designed distance. R. Koetter [11] provided a fast decoding algorithm for AG codes, which can be viewed as a parallel implementation of Berlekamp-Massey algorithm.

1.2 Hard-decision and Soft-decision Decoding

In a natural noise environment, the received signals are always continuous. For hard-decision decoding, the input symbols are binary or $\mathbb{F}_{2^m}$ symbols. While for soft-decision decoding, the received values from the channel are directly processed by the decoder in order to estimate a code sequence.

Soft-decision decoding improves error correcting performance of the decoders. However, soft-decision decoding usually leads to significant increase in decoding complexity. There are many soft-decision decoding algorithms developed. For convolutional codes, soft-decision Viterbi algorithm is widely used. The Viterbi algorithm can also be applied to linear block codes with a trellis. For linear block codes such as BCH code and Reed-Solomon,Generalized minimum distance decoding(GMD) algorithm [8], Chase algorithm [4], the ordered statistic decoding algorithm [9] and list-decoding algorithm [12] all can be used for sub-optimum soft-decision decoding.

1.2.1 Chase Decoding

The Chase algorithm [4] is a soft-decision decoding method which approximates optimum sequence decoding of block codes with relatively low computation complexity and a small performance degradation. The Chase algorithm works with a inner hard-decision decoder. According to the received values, a list of error patterns are generated. Each error pattern is added to the hard-decision received

word, and the resulting word is fed into the hard-decoder. Thus a list of candidate codewords are found. Based on the received value we can calculate a metric for each candidate codeword. The candidate with the largest metric will be selected as the output of the Chase decoder. Chase algorithm would improve the BER performance of almost all kind of block codes in both AWGN channel and fading channel.

1.2.2 Turbo Codes and Iterative Decoding

Turbo codes, introduced by Berrou et al. [2] are the new paradigm for error correcting codes. These codes are one of the first successful attempts of achieving error-correcting performance near the theoretical Shannon bound. For a BER of 10^{-5} and code rate 1/2, the authors presented an impressive E_b/N_0 ratio of $0.7dB$ in AWGN channel. Here $N_0/2$ is the variance of the zero mean Gaussian noise in the AWGN channel, and E_b is the power(or energy) of one transmitted bit. The ratio E_b/N_0 is usually called the *signal-to-noise ration per bit*(SNR).

Turbo coding introduces some new concepts such as iterative decoding and random interleaving to achieve remarkable result. The decoding algorithm adopted is a soft-input soft-output(SISO) iteration decoding algorithm, which minimize the error probability. And turbo codes have a weight distribution that approaches that of random codes for long interleavers. Those turbo codes are made from two concatenated recursive convolutional codes. The codes using such coding and decoding scheme are usually called convolutional turbo codes(CTC).

Later, R. Pyndiah [16] [15] investigated an equivalent turbo block code. Product codes and iterative decoding are used as two major technique of block turbo codes(BTC). It is shown that block turbo codes also can achieve good performance similar to convolutional turbo codes.

1.3 Contributions of this Thesis

Our accomplishments and contributions, which are elaborated throughout this thesis, can be briefly listed as follows:

- Using the Chase algorithm collaborating with R. Koetter's parallel Berlekamp-Massey algorithm to implement the soft-decision decoding of AG codes. The BER performance was improved greatly in both AWGN channel and Rayleigh fading channel.

- Present a iterative Chase decoding scheme for product codes constructed by AG codes. Because of the relatively low error correcting ability of the hard-decision decoder, we have to include a codeword validation procedure before we use a list of candidate codewords to generate the soft-output.

1.4 Thesis Outline

- Chapter 2 is devoted to the basic concepts of algebraic geometry and some important definitions for algebraic geometric codes with special emphasis on functional codes over Hermitian curves and residual codes over Klein quartic curves. We also presented some interesting similarities between functional Hermitian codes and Generalized Reed-Solomon codes.

- In chapter 3, the Chase algorithm for algebraic-geometric codes is investigated. R. Koetter's parallel implementation of Berlekamp-Massey algorithm is selected as the hard-decision decoder of Algorithm. In this chapter, simulation results of one-point AG code using the Chase algorithm in both AWGN channel and Rayleigh fading channel are presented.

- In chapter 4, basic concepts and structure of product codes are introduced

including the symbol concatenation and bit concatenation scheme for non-binary product codes. We adopt iterative Chase decoding algorithm to decode product codes construed by AG codes. The simulation results are shown in this chapter, and the BER performance of these product codes are also discussed.

- Chapter 5 draws the remark for this thesis and points out some promising future research areas of the iterative Chase decoding of algebraic-geometric codes.

Chapter 2

Algebraic Geometric Codes

2.1 Introduction

In this chapter, the basic concepts in algebraic geometry required for the understanding of algebraic geometric error-correcting codes will be explained. The aim here is to provide the reader with the most basic knowledge of algebraic geometry for making sense of the codes presented in this report rather than to give a full treatment of the complex subject of algebraic geometry. For a more concise and extensive review of algebraic geometry, the readers are encouraged to read up on [3].

Consider an algebraic curve χ with a subset $\mathcal{P}$ consisting of n points which are enumerated $P_1, \ldots, P_n($ which are the rational points of χ, i.e. points that have coordinates in $\mathbb{F}_q$). Suppose that we have a vector space L over $\mathbb{F}_q$ of functions on χ with values in $\mathbb{F}_q$. Thus $f(P_i) \in \mathbb{F}_q$ for all i and $f \in L$. In this way we can define a code over $\mathbb{F}_q^n$ as the image of the evaluation map below

$$\alpha_{\mathcal{P}} : L \rightarrow \mathbb{F}_q^n \tag{2.1}$$

which is defined by $\alpha_{\mathcal{P}}(f) = (f(P_1), \ldots, f(p_n))$. The evaluation map is linear, so

its image is a linear code. We call the above codes Algebraic Geometric codes(or AG codes, for short). This one of the two different ways to define an Algebraic Geometric code, known as *functional code*. The other class of Algebraic Geometric codes is called *residual code*, which is the dual code of the functional code. We will give the strict definition of AG codes in the following sections.

2.2 Definition of Algebraic-Geometric Codes

Algebraic Geometric codes can be viewed as a generalization of famous Reed-Solomon codes(or RS codes, for short) because RS codes also could be defined under above situation. In the case of RS codes, the algebraic curve χ is the affine line over $\mathbb{F}_q$, the points are n distinct elements of $\mathbb{F}_q$ and L is the vector space of polynomials of degree at most $k-1$ and with coefficients in $\mathbb{F}_q$, assuming $k \leq n$. Let $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ be a set of n distinct elements from $\mathbb{F}_q$, We can define the code C by

$$C = \{(f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{n-1})), f \in L\} \tag{2.2}$$

The vector space L has dimension k, and the polynomials in the vector space have at most $k-1$ zeros, so the nonzero codewords have at least $n-k+1$ non-zeros. This code has parameter $[n, k, n-k+1]$. The length of a RS code is at most q. The major shortcoming of RS codes is that they require an alphabet size at least as large as the block length. There are many applications where codes over a small alphabet are required. AG codes of long block length can be defined over small alphabet. In other words, over the same alphabet, an algebraic geometric code would be longer than an Reed-Solomon code. For example, the code length of the Hermitian code defined over $\mathbb{F}_{q^2}$ is q^3, while code length of the extended Reed-Solomon code defined over the same alphabet is only q^2.

First, we introduce some important notions and theorems about algebraic functions fields that will be necessary for defining algebraic-geometric codes.

2.2.1 Algebraic Function Fields and Algebraic Curves

In the following, $\mathbb{F}$ is the algebraic closure of $\mathbb{F}_q$. $\mathbb{A}^n$ denotes n-dimension affine space with coordinates $x_1, x_2, \ldots, x_n$, and $\mathbb{P}^n$ denotes n-dimension projective space with homogeneous coordinates $x_0, x_1, \ldots, x_n$. $\mathbb{F}[X_1, \ldots, X_m]$ denotes the polynomial ring in m variables with coefficient in $\mathbb{F}$.

Definition 2.1 *Consider a prime ideal I in the ring $\mathbb{F}[X_1, \ldots, X_n]$, the set χ of zeros of I is called an* affine variety.

Definition 2.2 *The ring $\mathbb{F}[X_1, X_2, \ldots, X_n]/I$ is called the* coordinate ring $\mathbb{F}[\chi]$ *of the variety χ.*

Definition 2.3 *The quotient field of the ring $\mathbb{F}[\chi]$ is denoted by $\mathbb{F}(\chi)$. It is called the* function field *of χ. The element of $\mathbb{F}(\chi)$ are called rational functions. The dimension of the variety χ is the transcendence degree of $\mathbb{F}(\chi)$ over $\mathbb{F}$. If the dimension is 1, χ is called an* algebraic curve.

Definition 2.4 *Let χ be a curve defined over $\mathbb{F}_q$, that is to say, the defining equations have coefficient over $\mathbb{F}_q$. Then the points on χ with all coordinates in $\mathbb{F}_q$ are called* rational points.

Given a function field $\mathbb{F}(V)$ and a point set $\mathbb{P}$ associated with the function field, we can define a valuation map $v : \mathbb{F}(V) \times \mathbb{P} \to \mathbb{Z} \bigcup \{\infty\}$, which intuitively tell us how many poles or zeros a function in the function field has at the point. The exact definition of the valuation map can be found in [3]. If $v_P(x) < 0$, we say that

x has a pole at P, and $-v_P(x)$ is the pole order of x at P. If $v_P(x) > 0$, we say that x has a zero at P, and $v_P(x)$ is the zero order of x at P.

The valuation map v_P at any point satisfied the following properties:

- $v_P(a) = \infty$ iff a=0 $v_P(a)$ for all $a \in \mathbb{F}_q \backslash \{0\}$
- $v_P(ab) = v_P(a) + v_P(b)$ for all $a, b \in \mathbb{F}(V) \backslash \{0\}$
- $v_P(a+b) \geq min\{v_P(a), v_P(b)\}$ for all $a, b \in \mathbb{F}(V)$

Consider a function of a function field,we can also define the degree of a point $deg(P)$. When we pick a function $f \in \mathbb{F}(V)$ which has no pole at point P and evaluate the function at P, we get a value in the field $\mathbb{F}_{q^{deg(P)}}$. Points with degree one are the rational points of the curve.

Definition 2.5 *Consider a curve χ in $\mathbb{A}^2$, defined by the equation $F(X,Y) = 0$. Let P be a point on this curve. If at least one of the derivatives F_X or F_Y is not zero at P, then P is called a* nonsingular *point of the curve. A curve is called* nonsingular, regular or smooth *if all the points are nonsingular.*

The number of rational points is important in defining an algebraic geometric codes. A well known result is the Hasse-Weil bound. Let χ be a regular curve defined over $\mathbb{F}_q$ and let N_m be the number of rational points on χ over $\mathbb{F}_{q^m}$. The Hasse-Weil bound provide the inequality below:

$$|N_m - (1 + q^m)| \leq 2g\sqrt{q^m}. \tag{2.3}$$

Here g is the genus of the curve χ, we will give the definition of genus in the next subsection. This inequality actually gives both the upper bound and the lower bound of the number of rational points.

2.2.2 Divisors and Vector Space

Definition 2.6 *Consider an irreducible smooth projective curve χ over $\mathbb{F}$, a* divisor *is a formal sum $D = \sum_{P\in\chi} n_P P$ with $n_P \in \mathbb{Z}$ and n_P is zero for all but a finite number of points P. The support of a divisor is the set of points with nonzero coefficient n_P. And the degree of a divisor can be defined as $deg(D) = \sum_{P\in\chi} n_P deg(P)$*

Definition 2.7 *Let f be a nonzero rational function on χ, we can define the* divisor *of f as $(f) = \sum_{P\in\chi} v_P(f)P$*

Definition 2.8 *Let D be a divisor on a curve χ, we can define a* vector space *$\mathcal{L}(D)$ over $\mathbb{F}$ by $\mathcal{L}(D) = \{f \in \mathbb{F}(\chi)|(f) + D \geq 0\} \bigcup \{0\}$*

The dimension of $L(D)$ is denoted as $l(D)$. The *Theorem of Riemann* says that there exist a nonnegative integer m such that for every divisor G of χ

$$l(G) \geq deg(G) + 1 - m \tag{2.4}$$

and the smallest nonnegative integer with this property is called the *genus* of χ.

In order to determine $l(G)$ we need to know the so-called *differentials*. We can think of the differentials as objects in a form fdh where f and h are rational functions, and dh is the derivation of h. We denote the set of differentials on χ by Ω_χ. At every point P, there exist a *localparameter* that is a function u such that $v_P(u) = 1$, and for every differential ω there exist a function f such that $\omega = fdu$. Based on the definition of differential and local parameter, we can define *residue*, which is also important to the definition of AG codes.

Definition 2.9 *Let P be a point on χ, u is a local parameter at P and ω can be represent by $\omega = fdu$. The function f can be written as $\sum_i a_i u^i$. We define the residue of ω in the point P as $Res_P\omega = a_{-1}$.*

One of the basic results about curves is known as the *residue theorem.*

Theorem 2.1 *If ω is a differential on a smooth projective curve χ, then*

$$\sum_{P \in \chi} Res_P(\omega) = 0. \tag{2.5}$$

Definition 2.10 *The divisor (ω) of a differential ω is defined by*

$$(\omega) = \sum v_P(u)P. \tag{2.6}$$

Similar to the definition of vector space $\mathcal{L}(D)$, we can also define the vector space $\Omega(G)$.

Definition 2.11 *Let D be a divisor on a curve χ, we can define a* vector space *$\Omega(G)$ over $\mathbb{F}$ by*

$$\Omega(G) = \{\omega \in \Omega_\chi | \omega = 0, or(\omega) \geq G\} \tag{2.7}$$

The following theorem, known as the *Riemann-Roch theorem* is not only a central result in algebraic geometry with applications in other research areas, but also the key to the many new results in coding theory.

Theorem 2.2 (Riemann-Roch) *For a divisor G of a curve of genus g*

$$l(G) = deg(G) + 1 - g + l(K - G) \tag{2.8}$$

where K is a canonical divisor.

Here we ignore the definition of canonical divisor. For a canonical divisor always has degree $2g - 2$, we can get the result that for any divisor with $deg(G) > 2g - 2$,

$$l(G) = deg(G) + 1 - g \tag{2.9}$$

Let χ be a nonsingular projective curve over $\mathbb{F}_q$ of genus g and let $P_1, P_2, \ldots, P_n$ be n rational points on χ. Define divisor D as $D = P_1 + P_2 + \cdots + P_n$, and let G

be a divisor with support disjoint from D, and assume $2g-2 < deg(G), n$. We can define the linear code $C_L(D,G)$ over $\mathbb{F}_q$ as the image of the linear map from $L(G)$ to $\mathbb{F}_q^n$

$$\alpha : f \rightarrow (f(P_1), f(P_2), \ldots, f(P_n)) \tag{2.10}$$

where $L(G)$ is define as definition 2.8.

The code defined as above is always called *functional algebraic geometric code.*

Lemma 2.1 *The code $C_L(D,G)$ has dimension $k = deg(G) - g + 1$ and the minimum distance $d \geq d^* = n - deg(G)$.*

Another class of algebraic geometric codes are defined by residue construction. Select same D and G as the construction of *functional algebraic geometric code.* We can define the linear code $C_\Omega(D,G)$ over $\mathbb{F}_q$ as the image of the linear map from $\Omega(G-D)$ to $\mathbb{F}_q^n$

$$\alpha^* : \omega \rightarrow (Res_{P_1}(\omega), Res_{P_2}(\omega), \ldots, Res_{P_n}(\omega)) \tag{2.11}$$

where $\Omega(G-D)$ is define as definition 2.11.

$C_\Omega(D,G)$ is always called *residual algebraic geometric code.*

Lemma 2.2 *The code $C_\Omega(D,G)$ has dimension $k = n - deg(G) + g - 1$ and the minimum distance $d \geq d^* = deg(G) - 2g + 2$.*

It follows the residue theorem that the code $C_L(D,G)$ and $C_\Omega(D,G)$ are dual code each other.

In this thesis, we will focus only on so called one-point AG codes. For the one-point AG code, the divisor $G = mP_0$, which means that the functions of the vector space $L(G) = L(mP_0)$ are allowed to have poles at only one point.

2.3 Hermitian Codes

There are several kinds of algebraic curves commonly used to construct AG codes, such as Klein quartic, elliptic, hyperelliptic curves and Hermitian curves. The AG codes from Hermitian curves are called Hermitian codes. In this section, functional Hermitian codes are select as the example to demonstrate the basic properties and construction methods of functional AG codes.

2.3.1 Basis of the Hermitian Function Field

Consider a Hermitian curve χ defined with equation $x^{q+1} = y^q + y$ over the field $\mathbb{F}_{q^2}$. The curve has genus $q(q-1)/2$ and contains q^3 $\mathbb{F}_{q^2}$-rational points plus the point P_∞ corresponding to the point at infinity on the homogenization of the Hermitian curve. Then we can define the one-point Hermitian codes $C(P, mP_\infty)$, here P is the sum of the q^3 rational points. The set

$$\{x^i y^j, i \geq 0, 0 \leq j \leq q-1, iq + j(q+1) \leq m\} \tag{2.12}$$

is a basis of $L(mP_\infty)$. For $m \geq 2g-1$, the dimension of $L(mP_\infty)$ is $m-g+1$. This basis is called the *standard basis* of the vector space. It is obvious that in above basis, each term has a different pole order with respect to P_∞. Because x and y has pole order q and $q+1$ at P_∞ respectively, $x^i y^j$ has pole order $iq + j(q+1)$.

Consider the case of Hermitian codes when $q = 4, m = 15$, the basis of $\mathcal{L}(15P_\infty)$ with increasing pole order at P_∞ should be:

$$\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3\} \tag{2.13}$$

From Definition 2.8, it is obvious that for a positive integer n, $l((n+1)P_\infty) \geq l(nP_\infty)$. The dimension of the vector space $\mathcal{L}(nP_\infty)$ is the number of the monomials $x^i y^j$ that satisfy $0 \leq j \leq q-1, iq + j(q+1) \leq n$, and each element of the set has a pole

order no larger than n at P_∞. For an integer k, if there is no function in $\mathbb{F}(\chi)$ has pole order k at P_∞, k is called a *gap number* of P_∞. As a result, for a gap number k, the vector space $\mathcal{L}((k-1)P_\infty)$ and $\mathcal{L}(kP_\infty)$ is identical. In the case of one-point Hermitian codes, the number of the gap numbers is the genus of the Hermitian curve χ.

Theorem 2.3 (cf. [3] Theorem 3.32) *Suppose $g > 0$ and P is a closed point of degree one. Then there are exactly g gap numbers $i_1 < i_2 < \cdots < i_g$ of P, and $i_1 = 1$ and $i_g \leq 2g - 1$.*

Let's consider the vector space below:

$$R = \bigcup_{i=0}^{\infty} \mathcal{L}(iP_\infty) \tag{2.14}$$

It is obvious that $\{x^i y^j, j < q\}$ is the basis of above vector space. $iq + j(q+1)$ can be also viewed as a weighted degree function of the monomials $\{x^i y^j, j < q\}$. For example, we consider the case $q = 4$. With the increasing order of the weighted degree, we can display the elements of the basis of R in the following sequence:

$$\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, \underline{x^i, x^{i-1}y, x^{i-2}y^2, x^{i-3}y^3}, \dots\} i = 4, 5, \dots. \tag{2.15}$$

And the weighted degree of above monomial sequence is

$$\{0, 4, 5, 8, 9, 10, 12, 13, 14, 15, 16, \dots\}. \tag{2.16}$$

In this example, there are 6 gap numbers $\{1, 2, 3, 6, 7, 11\}$, while the genus $g = \frac{4(4-1)}{2} = 6$.

In general, the standard basis of a Hermitian function field defined over F_{q^2} can

be described as:

$$1$$

$$x, y$$

$$x^2, xy, y^2$$

$$\dots$$

$$x^{q-1}, x^{q-2}y, \dots, xy^{q-2}, y^{q-1}$$

$$x^q, x^{q-1}y, \dots, x^2y^{q-2}, xy^{q-1}$$

$$\dots$$

2.3.2 Generator Matrix of Hermitian Codes

Given the basis of the vector space $L(mP_\infty)$ provided in the previous subsection, we can construct a generator matrix of the code $C_{\mathcal{L}}(P, mP_\infty)$ using the monomials of equation 2.12.

Example 2.1 *Let $q = 2$ and $m = 4$, the Hermitian curve $x^3 = y^2 + y$ is defined over $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ has 8 rational points as*

$P_1 = (0, 0), P_2 = (0, 1), P_3 = (1, \alpha), P_4 = (1, \alpha^2)$

$P_5 = (\alpha, \alpha), P_6 = (\alpha, \alpha^2), P_7 = (\alpha^2, \alpha), P_8 = (\alpha^2, \alpha)$

The basis of $\mathcal{L}(4P_\infty)$ is $\{1, x, y, x^2\}$.

The generator matrix is

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 \\ 0 & 1 & \alpha & \alpha^2 & \alpha & \alpha^2 & \alpha & \alpha^2 \\ 0 & 0 & 1 & 1 & \alpha^2 & \alpha^2 & \alpha & \alpha \end{pmatrix}$$

Since the corresponding residual code $C_\Omega(P, mP_\infty)$ is the dual code of $C_{\mathcal{L}}(P, mP_\infty)$, we can construct the parity check matrix of the residual code $C_\Omega(P, mP_\infty)$ using

the similar method.

It is obvious that the generator matrix of a functional code is the parity check matrix of the corresponding residual code.

For other classes of AG codes, we can construct the basis of function field in a similar way. With the knowledge of rational points and basis of function field, we could construct the generator matrix for correspond functional code. The matrix is also the parity check matrix for the residual code. From the parity check matrix, we can compute the generator matrix of the residual code easily. In this thesis, all the simulations are implemented on the decoding of one-point residual AG codes over Klein quartic. Actually, all the algorithms discussed in this thesis can also implemented to decode one-point residual Hermitian codes.

2.3.3 The relationship between Hermitian Codes and Generalized Reed-Solomon Codes

The Hermitian codes we defined in the previous subsection has intrinsic relationship to Reed-Solomon codes. In [19], Yaghoobian and Blake presented that Hermitian codes can be expressed as concatenated generalized Reed-Solomon codes. We introduce the definition of generalized Reed-Solomon codes firstly.

Definition 2.12 *We define the* generalized Reed-Solomon codes $GRS_q(k, v)$ *over* $\mathbb{F}_q$ *as*

$$GRS_q(k, v) = \{(v_{q-1}f(\alpha_{q-1}), v_0 f(\alpha_0), \ldots, v_{q-2}f(\alpha_{q-2}) | f(x) \in \mathbb{F}_q[x], deg(f(x) < k\} \tag{2.17}$$

where $\mathbb{F}_q = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$ *and by notation* $\alpha_i = \alpha^i$ *for* $0 \leq i \leq q-2$ *and* $\alpha_{q-1} = 0$*; and* $v = (v_{q-1}, v_0, \ldots, v_{q-2})$ *with* $v_i \in \mathbb{F}_q$ *for* $0 \leq i \leq q-1$.

According to 2.12, we can get the lemma below:

Lemma 2.3 *The functions in the vector space $\mathcal{L}(mP_\infty)$ have the following form.*

$$\mathcal{L}(mP_\infty) = \{f_0(x) + yf_1(x) + \cdots + y^k f_k(x)\} \tag{2.18}$$

where $deg(f_j(x)) < \lfloor \frac{m-j(q+1)}{q} \rfloor + 1$ for $0 \leq j \leq k$, and $k = min(q-1, \lfloor \frac{m}{q+1} \rfloor)$

As a result, if we arrange the rows of the generator matrix of the Hermitian codes in the increasing order of y-degree, we can divide the rows into $k+1$ blocks, where $k+1$ is defined as the lemma above. Consider the Hermitian curve $x^5 = y^4 + y$ and the Hermitian function field $\mathcal{L}(37P_\infty)$, the basis of the function field can be divided into following four groups: $\{1, x, \ldots, x^{10}\}$, $\{y, xy, \ldots, x^8y\}$, $\{y^2, xy^2, \ldots, x^6y^2\}$ and $\{y^3, xy^3, \ldots, x^5y^3\}$.

Now we consider the columns of the generator matrix of the Hermitian codes. Each column corresponds to a rational point(not the point at infinity) of the Hermitian curve. There q^3 rational points for the Hermitian curve $x^{q+1} = y^q + y$. These rational points can be divided into q groups, and in each group the x-coordinate of the q^2 points would be distinct. Since the Hermitian curve is defined over $\mathbb{F}_{q^2}$, these x-coordinate will be $\{0, 1, \alpha, \ldots, \alpha^{q-2}\}$ respectively. For example, the 64 rational points of Hermitian curve $x^5 = y^4 + y$ is shown in table 2.1. It is obvious that the 64 rational points can be divided into 4 groups as we discussed above. As a result, the columns of the generator matrix of Hermitian codes can be divided into q blocks. In each block, the x-coordinates of the rational points are $\{0, 1, \alpha, \ldots, \alpha^{q-2}\}$ respectively.

In conclusion, the generator matrix of Hermitian code could be divided into $(k+1) \times q$ blocks, and each block is a generator matrix of a Generalized Reed-Solomon codes.(In each block, the vector v is determined by the y-degree of the rows of the block, and the y-coordinates of the columns of the block). So the Hermitian codes can be the sum of $k+1$ concatenated generalized Reed-Solomon

X	Y
0	$0, 1, \alpha^5, \alpha^{10}$
$1, \alpha^3, \alpha^6, \alpha^{12}, \alpha^9$	$\alpha, \alpha^2, \alpha^4, \alpha^8$
$\alpha, \alpha^4, \alpha^7, \alpha^{13}, \alpha^{10}$	$\alpha^6, \alpha^7, \alpha^9, \alpha^{13}$
$\alpha^2, \alpha^5, \alpha^8, \alpha^{14}, \alpha^{11}$	$\alpha^{11}, \alpha^{12}, \alpha^{14}, \alpha^3$

Table 2.1: Rational Points of Hermitian Curve

codes(concatenated by q GRS codes). In the case of above example, the generator matrix of the Hermitian codes can be divided into 4×4 blocks as below:

$$\mathbf{G}_{Hermitian} = \begin{pmatrix} \mathbf{G}_{GRS1}(16,10) & \mathbf{G}_{GRS2}(16,10) & \mathbf{G}_{GRS3}(16,10) & \mathbf{G}_{GRS4}(16,10) \\ \mathbf{G}_{GRS1}(16,9) & \mathbf{G}_{GRS2}(16,9) & \mathbf{G}_{GRS3}(16,9) & \mathbf{G}_{GRS4}(16,9) \\ \mathbf{G}_{GRS1}(16,7) & \mathbf{G}_{GRS2}(16,7) & \mathbf{G}_{GRS3}(16,7) & \mathbf{G}_{GRS4}(16,7) \\ \mathbf{G}_{GRS1}(16,6) & \mathbf{G}_{GRS2}(16,6) & \mathbf{G}_{GRS3}(16,6) & \mathbf{G}_{GRS4}(16,6) \end{pmatrix} \tag{2.19}$$

Where the generator matrices $\mathbf{G}_{GRS1}(16,10)$, $\mathbf{G}_{GRS2}(16,10)$, $\mathbf{G}_{GRS3}(16,10)$ and $\mathbf{G}_{GRS4}(16,10)$ are identical, since the y-degree of these basis elements of the function field for these matrices are all zero, and the four ordered rational points groups only differ on the y-coordinates. Among the rest of the generator matrices of GRS codes, there are no any two matrices are identical, since the y-coordinate difference has been taken into account.

2.4 Residual Algebraic Geometric Codes over Klein Quartic Curves

In this section, the construction of a specific residual code will be given in details. The curve and the finite field under considerations is the Klein quartic curve and $\mathbb{F}_8$. The projective form of the Klein quartic curve χ is given by : $X^3Y+Y^3Z+Z^3X = 0$

while $\mathbb{F}_8$ is constructed using the irreducible polynomial $x^3 + x + 1$ over $\mathbb{F}_2$.

2.4.1 Rational Points of Klein Quartic Curve

The length of the code $C_\Omega(D, mP_\infty)$ is dependent on the number of rational points of the curve over the finite fields. Hence finding the number of the rational points over the field of interest is the very first thing to do. Recall the Hasse-Weil bound in Equation 2.3 presented in the previous section, we can calculate the upper bound and the lower bound for the number of rational points in Klein quartic curves. This bound states that the number of rational points n over $\mathbb{F}_8$ would be equal to or less than $q + 1 + 2\lfloor g\sqrt{q^m} \rfloor = 8 + 1 + \lfloor 6\sqrt{8} \rfloor = 24$.

In fact, there are 24 rational points over $\mathbb{F}_8$. Over $\mathbb{F}_2$, three rational points $(1 : 0 : 0)$, $(0 : 1 : 0)$ and $(0 : 0 : 1)$ are easily found. Using the automorphisms described in [3], we can find other 21 rational points over $\mathbb{F}_8$. All the 24 rational points are displayed in Table 2.2.

2.4.2 Codes Definition and Parameters

To define a one-point residual algebraic geometric code, the choice of P_∞ , D and the degree of P_∞ , m must be specified. The curve χ contains 24 rational points over $\mathbb{F}_8$, of which the point $(0 : 1 : 0)$ will be chosen to be P_∞ while the rest of the points would be chosen to be the elements of $Supp(D)$. The degree of P_∞ , m is arbitrarily taken to be 7 and hence the code that is examined in this thesis is the residual code $C_\Omega(D, 7P_\infty)$. The designed minimum distance $d^*(C_\Omega)$ of the code $C_\Omega(D, 7P_\infty)$ is $deg(7P_\infty) - 2g + 2$. The genus of Klein quartic curves is 3. So the designed minimum distance is 3.

Rational points	Coordinates	Rational points	Coordinates
P_∞	$(0,1,0)$	P_{12}	$(\alpha^3,1,\alpha^3)$
P_1	$(1,0,0)$	P_{13}	$(\alpha^3,\alpha^3,1)$
P_2	$(0,0,1)$	P_{14}	$(1,\alpha^3,\alpha^3)$
P_3	$(1,\alpha^2,\alpha^4)$	P_{15}	$(1,1,\alpha)$
P_4	$(\alpha^4,1,\alpha^2)$	P_{16}	$(\alpha,1,1)$
P_5	$(\alpha^2,\alpha^4,1)$	P_{17}	$(1,\alpha,1)$
P_6	$(1,\alpha^5,\alpha^5)$	P_{18}	$(\alpha^5,\alpha,1)$
P_7	$(\alpha^5,1,\alpha^5)$	P_{19}	$(1,\alpha^5,\alpha)$
P_8	$(\alpha^5,\alpha^5,1)$	P_{20}	$(\alpha,1,\alpha^5)$
P_9	$(\alpha,\alpha^2,1)$	P_{21}	$(\alpha^4,\alpha^3,1)$
P_{10}	$(1,\alpha,\alpha^2)$	P_{22}	$(1,\alpha^4,\alpha^3)$
P_{11}	$(\alpha^2,1,\alpha)$	P_{23}	$(\alpha^3,1,\alpha^4)$

Table 2.2: Rational Points of Klein Quartic Curve over $\mathbb{F}_8$

2.4.3 Function Field Basis for Klein Quartic Curve

For the Klein quartic curve over $\mathbb{F}_8$ defined by $X^3Y + Y^3Z = Z^3X$, to define a standard basis for $\mathcal{L}(7P_\infty)$, we must obtain a set of rational functions $\{\phi_i\}$ with pole order i at P_∞, where $0 \leq i \leq 7$. The gap at P_∞ is 1,2 and 4. As a result, the rational function with the lowest pole order at P_∞ is ϕ_3.

To determine these rational functions, we can using the intersection divisors of the coordinate functions. For the case of $X = 0$, the equation of χ is reduced to the following equation.

$$Y^3Z = 0 \tag{2.20}$$

Obviously, the two curves $X = 0$ and χ, intersect at the point $(0, 0, z)$ with multiplicity 3 and at the point $(0, y, 0)$ with multiplicity 1. Therefore, the divisor of $X = 0$, denoted as (X) is represented in projective points, $(X) = 3(0 : 0 : 1) + (0 : 1 : 0)$. Similarly, we can get $(Y) = 3(1 : 0 : 0) + (0 : 0 : 1)$ and $(Z) = 3(0 : 1 : 0) + (1 : 0 : 0)$. As mentioned in previous section, the divisor of a rational function $f = \frac{\Psi}{\Phi}$ is defined as $(f) = (\Psi) - (\Phi)$. Then we can get

$$(\frac{Y}{Z}) = 2(1 : 0 : 0) + (0 : 0 : 1) - 3(0 : 1 : 0). \tag{2.21}$$

It is obvious that $\frac{Y}{Z}$ has pole order 3 at P_∞. So we can choose $\frac{Y}{Z}$ as ϕ_3. Using same method, we can find that

$$(\frac{X}{Z}) = 3(0 : 0 : 1) - (1 : 0 : 0) - 2(0 : 1 : 0). \tag{2.22}$$

Then $\frac{X}{Z}$ has pole order 2 at P_∞, but it can not be used as ϕ_2 since it also has pole order 1 in P_1. Noticed that ϕ_3 has zero order 2 in P_1, we can construct ϕ_5 by multiplying $\frac{X}{Z}$ and $\frac{Y}{Z}$ since the pole of $\frac{X}{Z}$ can be compensated by the zero of $\frac{Y}{Z}$. Let consider the rational function $\frac{Y}{X}$, where

$$(\frac{Y}{X}) = 3(1 : 0 : 0) - 2(0 : 0 : 1) - (0 : 1 : 0). \tag{2.23}$$

ϕ_0	ϕ_3	ϕ_5	ϕ_6	ϕ_7
1	$\frac{Y}{Z}$	$\frac{XY}{Z^2}$	$\frac{Y^2}{Z^2}$	$\frac{Y^3}{Z^2X}$

Table 2.3: Standard Basis of $\mathcal{L}(7P_\infty)$

This rational function has pole order 1 at P_∞ and pole order 2 at P_2. These 3 rational functions can be used as factors to construct all the basis elements of $\mathcal{L}(7P_\infty)$, although $\frac{Y}{X}$ and $\frac{X}{Z}$ can not be included to function field.

So $\phi_5 = \frac{XY}{Z^2}$. Similarly, ϕ_6 is represented as ϕ_3^2, and ϕ_7 could be obtained as the product of ϕ_6 and $\frac{Y}{X}$.

Actually, the functions ϕ_3, ϕ_5 and ϕ_7 generated the ring of functions with poles at P_∞. It is easy to verify that the basis has the relationship below:

$$\phi_i\phi_j = \begin{cases} 0, & i \in \{1,2,4\} \vee j \in \{1,2,4\}; \\ \phi_{i+j} + \phi_{i+j-7}, & i \notin \{1,2,4\} \wedge j \notin \{1,2,4\} \wedge i \not\equiv j \bmod 3; \\ \phi_{i+j}, & \text{otherwise.} \end{cases} \tag{2.24}$$

Since the basis of the function field could be used to generate the parity check matrix for residual AG codes, this relationship is especially useful for the calculation of the syndromes of the one-point residual AG code over Klein quartic curves. This method is adopted in the decoding algorithm of residual AG codes in next chapter. For more details, readers can refer to [11].

As a result, we can get the standard basis of $\mathcal{L}(7P_\infty)$ as Table 2.3.

2.4.4 Parity Check Matrix and Generator Matrix

Using the standard basis for $\mathcal{L}(7P_\infty)$ and the 24 rational points presented in the previous part of this chapter, the parity check matrix $\mathbf{H}$ of the code $C_\Omega(D, 7P_\infty)$ can be obtained by evaluate the element of the basis of the function field $\mathcal{L}(7P_\infty)$ at each rational point.

Note that for P_1 and P_2, the functions ϕ_i $i = 3, 5, 6, 7$ can not be evaluated normally because the denominators are zeros. However, considering the zeros of the divisors, the values of $\phi_i(P_1)$ $i = 3, 5, 6, 7$ should be all zeros. Similarly, the values of $\phi_i(P_2)$ $i = 3, 5, 6$ should be all zeros and $\phi_7(P_2) = 1$. The parity check matrix of the code $C_\Omega(D, 7P_\infty)$ is shown in Table 2.4.4.

Using the parity check matrix $\mathbf{H}$, the generator matrix of the code $C_\Omega(D, 7P_\infty)$ can be determined by the relationship $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. The matrix $\mathbf{G}$ can be obtained by solving a system of linear equations.

Solving the system of equations by Gaussian elimination would show that there is more than one solution, which is consistent with the fact that $\mathbf{G}$ represents a vector space.

To realize systematic encoding, which is more convenient to construct product codes in later chapters, we use linear permutation to make the sub-matrix(the last 18 columns of the generator matrix) an identity matrix. Using such a generator matrix shown in Table 2.5, in a codeword of $C_\Omega(D, 7P_\infty)$ the last 18 symbols are actually the information symbols, and the first 5 symbols are parity check symbols.

2.5 Asymptotically Good AG Codes

As mentioned in the previous chapter, one major advantage of algebraic geometric codes is that those codes can be used to give an asymptotically good sequence of codes with parameters better than the Varshamov-Gilbert bound in a certain range of the code rate and with large enough alphabets. This is one of the primary reasons for the importance of enormous interests in algebraic geometric codes.

A linear block code $\mathbf{C}$ defined over $\mathbb{F}_q$ will be denoted by $(n, k, d)_q$ code, if $\mathbf{C}$ is a subset of $\mathbb{F}_q^n$ with minimum distance d and the code dimension is k. The quotient $\frac{k}{n}$ is called the code rate, and can be denoted by R. The relative minimum distance

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 \\
1 & \alpha^5 & \alpha & \alpha^3 & \alpha^5 \\
1 & \alpha^5 & 1 & \alpha^3 & \alpha^6 \\
1 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 \\
1 & 1 & \alpha^2 & 1 & \alpha^5 \\
1 & \alpha^2 & \alpha^2 & \alpha^4 & \alpha^6 \\
1 & \alpha^5 & \alpha^3 & \alpha^3 & \alpha^3 \\
1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 \\
1 & \alpha^6 & \alpha^4 & \alpha^5 & \alpha^6 \\
1 & \alpha^6 & 1 & \alpha^5 & \alpha^3 \\
1 & \alpha^4 & \alpha^4 & \alpha & \alpha^5 \\
1 & \alpha^3 & \alpha^6 & \alpha^6 & \alpha^6 \\
1 & 1 & \alpha^4 & 1 & \alpha^3 \\
1 & \alpha^6 & \alpha^5 & \alpha^5 & \alpha^5 \\
1 & 1 & \alpha & 1 & \alpha^6 \\
1 & \alpha & \alpha & \alpha^2 & \alpha^3 \\
1 & \alpha & \alpha^6 & \alpha^2 & \alpha^5 \\
1 & \alpha^4 & \alpha^3 & \alpha & \alpha^6 \\
1 & \alpha^2 & \alpha^5 & \alpha^4 & \alpha^3 \\
1 & \alpha^3 & 1 & \alpha^6 & \alpha^5 \\
1 & \alpha & \alpha^5 & \alpha^2 & \alpha^6 \\
1 & \alpha^3 & \alpha^2 & \alpha^6 & \alpha^3
\end{pmatrix}$$

Table 2.4: Parity Check Matrix of (23,18,3) Residual Code over Klein Quartic Curve

$$\begin{pmatrix}
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^2 & 0 & \alpha^5 & 0 & \alpha & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^3 & \alpha^3 & \alpha^5 & \alpha^4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^6 & 1 & \alpha^3 & \alpha^2 & \alpha & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^4 & \alpha & \alpha^6 & \alpha^3 & \alpha^3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^3 & \alpha^5 & 1 & \alpha^5 & \alpha^3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^4 & \alpha^4 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha^6 & \alpha^5 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & \alpha^5 & \alpha^4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^3 & \alpha^5 & \alpha^3 & \alpha^6 & \alpha^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha^4 & \alpha^5 & \alpha & \alpha^3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha & \alpha^5 & \alpha^4 & 1 & \alpha^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\alpha^6 & 0 & \alpha & \alpha^6 & \alpha^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\alpha & \alpha & \alpha & \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & \alpha^2 & \alpha^4 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\alpha^3 & \alpha^4 & \alpha & \alpha^2 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\alpha & \alpha^5 & \alpha^6 & \alpha & \alpha^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & \alpha^2 & 1 & \alpha^5 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Table 2.5: Generator Matrix of (23,18,3) Residual Code over Klein Quartic Curve

$\frac{d}{n}$ is denoted by δ. The dimension k and the minimum distance d of an algebraic geometric code on a curve of genus g with code length n defined over $\mathbb{F}_q$ satisfy

$$k + d \geq n + 1 - g. \tag{2.25}$$

by Lemma 2.1 and Lemma 2.2. Hence

$$R + \delta \geq 1 - \frac{g-1}{n} \tag{2.26}$$

In order to construct a sequence of good codes, we have to find curves with low genus and many rational points.

Definition 2.13 *A sequence of codes C_m ($m \in \mathbb{N}$) with parameters (n_m, k_m, d_m) over a fixed finite field $\mathbb{F}_q$ is called* asymptotically good *if n_m tends to infinity, and $\frac{d_m}{n_m}$ tends to a nonzero constant δ, and $\frac{k_m}{n_m}$ tends to a nonzero constant R for $m \to \infty$.*

Here R can be defined as a function of δ.

$$\alpha(\delta) = \limsup_{m \to \infty} \frac{k_m}{n_m} \tag{2.27}$$

It is easy to see that $\alpha(\delta) = 0$ for $1 - \frac{1}{q} \leq \delta \leq 1$. The Varshamov-Gilbert bound is the fact that

$$\alpha(\delta) \geq 1 - H_q(\delta) \tag{2.28}$$

for $0 \leq \delta \leq 1 - \frac{1}{q}$. Here $H_q(x)$ is the q-ary entropy function defined by

$$\begin{aligned} H_q(0) &= 0 \\ H_q(x) &= x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x), 0 < x \leq 1 - \frac{1}{q} \end{aligned}$$

In [18], it is shown that by using algebraic geometric codes it is possible to prove that

$$\alpha(\delta) + \delta \geq 1 - \frac{1}{\sqrt{q} - 1} \tag{2.29}$$

if q is a square. It is called Tsfasman-Vlådut-Zink bound. It turns out that the TCZ bound is better than VG bound when $q \geq 49$ in a certain range of δ.

Let

$$A(q) = \limsup_{g \to \infty} \frac{N(g)}{g} \tag{2.30}$$

where $N(g)$ is the maximal number of $\mathbb{F}_q$ rational points on a curve of genus g. The Hasse-Weil bound implies that

$$A(q) \leq 2\sqrt{q}. \tag{2.31}$$

Vlådut improved this result to $A(q) \leq \sqrt{q} - 1$. And in [18], Tsfasman, Vlådut and Zink showed that

$$A(q) = \sqrt{q} - 1 \tag{2.32}$$

by studying modular curves over finite field. It is obvious that the above equation can derive the Equation 2.29.

However, the construction using modular curves is difficult, and the actual construction of generator and parity-check matrix is intractable. Many researchers have tried to find a more simple construction. In [7], Feng and Rao suggested that asmptotically good codes can be obtained over so-called generalized Klein curves. Recently, two sequences of asymptotically good codes defined over modular curves were presented by Elkies [6].

2.6 Summary

In this chapter, we presented some necessary definitions and lemmas of algebraic geometry, which are important to the definition and properties of algebraic-geometric codes. We also provide the definition and some important properties of one-points Hermitian codes. And the structure similarities of functional Hermitian codes and

generalized Reed-Solomon codes are discussed. In the last part of this chapter, we discussed the construction method and code parameters of residual algebraic geometric codes over Klein quartic curves, which will used in the simulations of subsequent chapters.

Chapter 3

Chase Decoding of AG Codes

3.1 Introduction

In this chapter, we described the Chase algorithm [4], which is an sub-optimum soft-decision decoding algorithm for block codes. Based on the reliability of the received symbols, the Chase algorithm will generate a list of most likely error patterns. Each error pattern is added to the hard-decision received word and decoded using a hard-decision decoder. A list of candidate codewords are found and scored by computing their metrics with respect to the reliability sequence of the soft-decision received symbols. The codeword with the best metric is selected as the output of the algorithm.

Chase decoding is attractive since it is applicable to all block codes for which a binary decoding method exists. For non-binary block codes, we can represent them in binary form to decode using algorithm. The BER performance of a Chase decoder will greatly depend on the error correcting ability of the hard-decoder used in it. In the simulations of this chapter, the algebraic geometric code defined over Klein quartic curve was decoded using a Chase decoder.

3.2 Parallel Berlekamp-Massey Algorithm

As we mentioned above, the Chase works with a inner hard-decision decoder. In 1998, Ralf Koetter [11] presented a fast parallel decoder for algebraic geometric codes with the computation complexity that is in the same range as the Berlekamp-Massey decoder of Reed-Solomon codes.

Let γ be the first non-gap in the non-gap sequence of the space associated with the AG codes, the PBMA decoder could be viewed as γ independent Berlekamp-Massey decoder processing units. The time requirements of the PBMA decoder are determined by the time requirements for one of the γ processing units. This feature is important to make AG codes a competitive and reasonable alternatives to Reed-Solomon codes in many applications.

3.3 Soft-Decision Decoding

In a digital communication system, we can get the reliability information in the demodulation module. If we use the reliability information in the decoding module, it is called soft-decision decoding.

Generally, the computational overhead of soft-decision decoding is more intensive than hard-decision decoding. There are two main reasons for this. One is that soft-decision decoding usually have to implement real number computation, while hard-decision decoding only needs to implement integer computation. In practical applications of soft-decision decoding, real numbers will be quantized a finite number of bits. The other reason is that in soft-decision decoding, the a-posterior possibilities of the coded symbols given received values have to be computed.

3.4 The Chase Algorithm

In this section, Chase algorithm will be fully discussed. Let's describe the general form of Chase algorithms first.

3.4.1 General Description

Let $\mathbf{C}$ be a binary linear (n, k, d) block code(n, k and d are the code length, code dimension and minimum Hamming distance of the code respectively.), the Chase algorithm is designed to work with any hard-decision decoder that can correct up to $\lfloor (d-1)/2 \rfloor$ errors. Denote $\overline{c} = (c_1, c_2, \ldots, c_n)$ a codeword with binary symbols c_i taking values ± 1. Let $\overline{r} = (r_1, r_2, \ldots, r_n)$ be the received word using BPSK modulation from the physical channel.

$$r_i = c_i + w_i, i = 1, 2, \ldots, n \tag{3.1}$$

where w_i is a zero-mean Gaussian random variable with variance σ^2. The reliability of symbol c_i at the input of decoder is evaluated by the Log-Likelihood-Ratio(LLR) defined as:

$$\Lambda_i = \ln \frac{Pr(c_i = +1|r_i)}{Pr(c_i = -1|r_i)} \tag{3.2}$$

It is obvious that in the case of BPSK,

$$r_i = \frac{\sigma^2}{2} \Lambda_i \tag{3.3}$$

If $r_i \geq 0$, the LLR value is $\ln \frac{Pr(c_i=+1|r_i)}{Pr(c_i=-1|r_i)} = \frac{2}{\sigma^2} r_i$, while if $r_i < 0$, the LLR value would be $\ln \frac{Pr(c_i=-1|r_i)}{Pr(c_i=+1|r_i)} = -\frac{2}{\sigma^2} r_i$. As a result, if normalize with $\frac{2}{\sigma^2}$, the reliability of the received channel values are the amplitudes $|r_i|$.

The hard-decision received word is defined as:

$$\overline{z} = (z_1, z_2, \ldots, z_n) \tag{3.4}$$

where $z_i = 0$ if $r_i \geq 0$, otherwise $z_i = 1$.

In Chase algorithm, a list of error patterns are generated first, and each error pattern is added to the hard-decision received word. And the result word is fed into a hard-decision decoder. Let $\overline{u}$ denote the output of the hard-decision decoder. Then the metric of $\overline{u_j}$ with respect to the received word $\overline{r} = (u_{j1}, u_{j2}, \ldots, u_{jn})$

$$M(\overline{u_j}) = \sum_{i=1}^{n} (-1)^{u_{ji}} r_i \tag{3.5}$$

is computed. The output codeword with the largest metric is selected as the hard output of Chase algorithm. Chase algorithm can be viewed as a Minimum Distance decoding algorithm since the metrics used to select the output codeword from the candidate list are determined by the Euclidean distances between the received word $\overline{r}$ and the candidate codewords $\overline{u_j}$. Here each element of $\overline{u_j}$ should be represent in the form of ± 1. In another word, we use $\overline{y_j}$ to compute the Euclidean distance, where $y_{ji} = (-1)^{u_{ji}}$ for $i = 1, \ldots, n$. The square Euclidean distance between $\overline{r}$ and $\overline{y_j}$ is

$$|\overline{r} - \overline{y_j}|^2 = \sum_{i=1}^{n} (r_i - y_{ji})^2 = \sum_{i=1}^{n} (r_i - (-1)^{u_{ji}})^2 = |\overline{r}|^2 + n - 2\sum_{i=1}^{n} (-1)^{u_{ji}} r_i. \tag{3.6}$$

So we can obtain

$$|\overline{r} - \overline{y_j}|^2 = |\overline{r}|^2 + n - 2M(\overline{u_j}). \tag{3.7}$$

Because the first two items are constant values, the codeword with the largest metric defined in Equation 3.5 is the codeword nearest to the received word among the candidate codewords in the list. In binary transmission over an AWGN channel, the metric $M(\overline{u_j})$ is also called the *correlation* between the generated codeword u_j and the received word $\overline{r}$.

According to the error pattern generation, Chase algorithm can be classified into three types.

Type-I Test all error patterns within a sphere of radius $(d-1)$ from the received word. In practical, we do not have to test all error patterns of binary weight less than d. The binary hard-decision decoder are capable of correct up to $\lfloor (d-1)/2 \rfloor$. The decoder combined with an appropriate test pattern can yield any error pattern that has binary weight up to $(d-1)$.

Type-II Test the error patterns with no more than $\lfloor (d-1)/2 \rfloor$ errors located outside the set of the positions with the $\lfloor d/2 \rfloor$ lowest reliabilities. Similar to Type-I algorithm, we no longer test all possible error patterns of binary weight up to $(d-1)$. All the error patterns generated among the positions with $\lfloor d/2 \rfloor$ lowest reliabilities combined with the hard-decision decoder could yield any error pattern required. So we only have to test $2^{\lfloor d/2 \rfloor}$ error patterns.

Type-III Test those patterns with i 1's located at the i least reliable bit positions, where i is odd and $1 \leq i \leq d-1$. This algorithm has the smallest set of possible error patterns.

With less computation complexity compared to Type-I, and better performance compared to Type-III, Chase algorithm Type-II is the most popular among the three algorithms. A flow chart of Chase algorithm Type-II is shown in figure 3.1.

In the AWGN channel, all the three types Chase algorithm extend the error correcting ability of the hard-decision decoder.

Chase algorithm can generate a list of candidate codeword. This is especially useful in producing soft-output. As a result, Chase algorithm is widely used in the iterative decoding of block turbo codes. We will discuss this in next chapter.

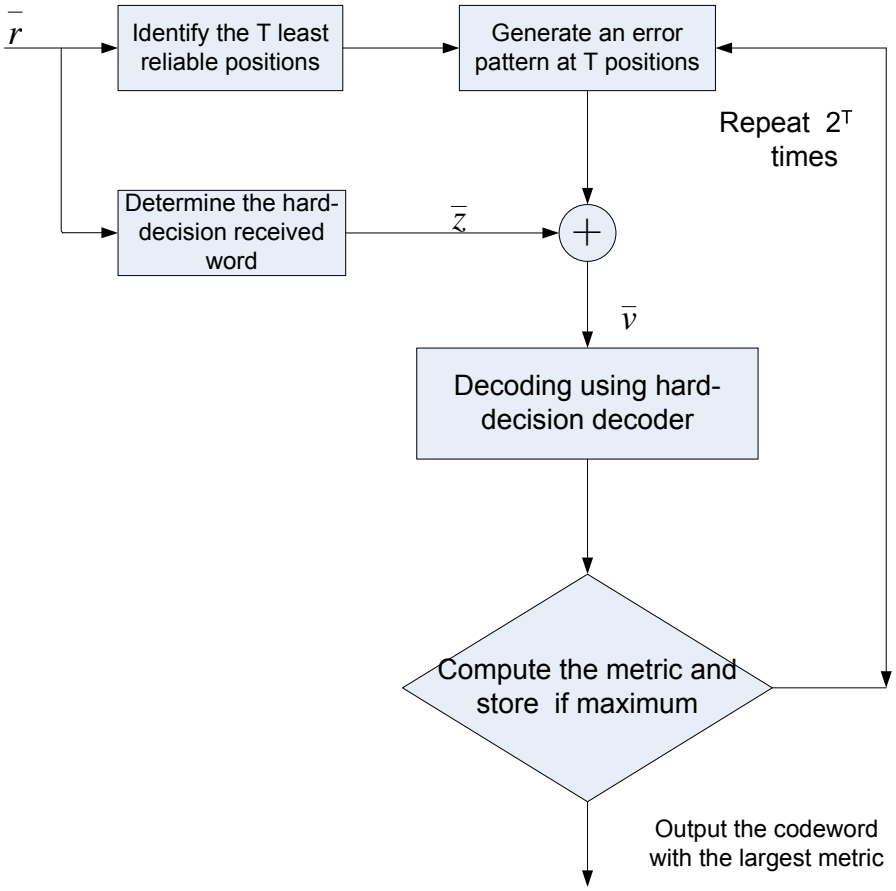

Figure 3.1: Flow Chart of Chase Algorithm

3.4.2 The Chase Algorithm in Fading Channels

Fading occurs in wireless communication systems in the form of a time-variant distortion of the transmitted signal. In this thesis, we only consider the case of flat Rayleigh fading. Flat means that the channel is not frequency selective, and its transfer function in the frequency domain is constant.

In a flat Rayleigh fading channel, we assumed that the transmitted signal x_i is perturbed by Rayleigh distributed multiplicative noise α_i in addition to the white Gaussian noise n_i.

$$v_i = x_i \times \alpha_i + n_i \tag{3.8}$$

Where $x_i = \sqrt{E_b}$ if 0 is transmitted, and $x_i = -\sqrt{E_b}$ if 1 is transmitted. The variance of n_i is denoted by $N_0/2$. When the multiplicative noise is normalized so that $E({\alpha_i}^2) = 1$, the SNR for this case is still $\frac{E_b}{N_0}$.

Similar to the case in AWGN channel, Chase algorithm can improve the error correcting performance of block code greatly. For more details, please refer [4].

3.4.3 The Chase Algorithm for Non-Binary Block Codes

Generally, the Chase algorithm is used for soft-decision decoding of binary linear block codes. For non-binary linear block codes, we can modify the Chase algorithm using two different methods.

The first method is to decode with respect to the reliability of symbols. In this case, the LLR value of each symbol will be expressed as:

$$\Lambda_i = \ln(\frac{Pr(c_i = x_i|r_i)}{Pr(c_i \neq x_i|r_i)}) \tag{3.9}$$

where x_i is not ± 1. If defined in $\mathbb{F}_8$, x_i would be one element of $\mathbb{F}_8$. Then $Pr(c_i \neq x_i|r_i)$ would be a sum of 7 possibilities.

$$\Lambda_i = \ln(\frac{Pr(c_i = x_i|r_i)}{\sum_{j=1,j\neq i}^{8} Pr(c_i = x_j|r_i)}) \tag{3.10}$$

where $x_1, \ldots, x_8$ is the set of eight symbols. And Λ_i could not be computed in a simple form like Equation 3.3. As a result, we have to implement power and log computation in the decoding algorithm. Thus the computation complexity increase greatly.

The second method is to represent the non-binary linear block codes in binary form. In this case, all the codewords are transmitted in binary form. Only when the received word plus a error pattern is fed into a hard-decision decoder, the input word and output codeword would be converted to non-binary form.

Table 3.1: Binary Representation of elements of $\mathbb{F}_8$

Element	Polynomial Representation	Binary Representation
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	α^2+1	101
α^4	$\alpha^2+\alpha+1$	111
α^5	$\alpha+1$	011
α^6	$\alpha^2+\alpha$	110

In this chapter, we implemented the Chase algorithm for the soft-decision decoding of AG codes defined over $\mathbb{F}_8$. In table 3.1 the representations of the elements of $\mathbb{F}_8$ is explained.

In the simulations, PBMA is adopted as the hard-decision decoder. The one-point AG code is a non-binary code. The error correcting ability of the hard-decision decoder is always with respect to symbol errors. As a result, if we represent the word in binary form, whether the hard-decoder can correct the bit errors is not only determined by the bit error numbers, but also determined by the positions of the bit errors. For example, the design minimum distance of (23,18,3) AG codes over Klein quartic is 3, the hard-decision decoder(PBMA) can only correct 1 symbol errors. In binary form, the hard-decoder can correct 3 bit errors if these 3 bits who construct 1 symbol in 8-ary form. While if there are two bits errors who belong to different symbols, the hard-decision decoder might not be able to correct the bit errors. We can show this in the following example. It is known that PMBA decodes based on the syndrome values which are determined only by the code error patterns. So we select zero codeword as the transmitted codeword.

Example 3.1 *Let's consider the PBMA decoder for the residual AG code* $(23, 18, 3)$ *over Klein quartic curve. When there are two symbol errors in the received word at the 1st and 3rd position of the codeword, the error magnitudes are* 1*. In another word, if represented in binary form, there are only two bit errors.*

The output of PBMA decoder is

$$\alpha^5 \quad \alpha^5 \quad 1 \quad \alpha^6 \quad 1 \quad \alpha^5 \quad \alpha^2 \quad \alpha^2 \quad \alpha^4 \quad 1 \quad \alpha^6 \quad \alpha^3$$

$$\alpha^5 \quad \alpha^2 \quad \alpha^2 \quad \alpha^3 \quad \alpha^5 \quad \alpha \quad \alpha \quad \alpha \quad 1 \quad \alpha^6 \quad \alpha^4$$

It shows that in some error patterns the PBMA decoder can not correct 2 bit errors.

However, if we use the Chase Type-II algorithm and select the least reliable bit positions to generate error pattern, we can mitigate the shortcoming of the hard-decision decoder. In AWGN or Rayleigh fading channel, the bit errors caused by channel noise would appear randomly in a word. The error-correcting ability of the PBMA decoder will affect the performance of the Chase decoder compared with those of binary linear block codes using the Chase decoder.

3.5 Simulation Results and Discussion

In this section, the simulation results we have done for the Chase decoding of AG codes would be shown. Because of its reduced complexity and good error-correcting performance among three types of Chase algorithm, only Type-II Chase algorithm is implemented in our simulations. The (23,18,3) residual AG codes over $\mathbb{F}_8$ from Klein quartic curves are selected to demonstrate the Chase decoding of AG codes. Since the codes are not binary, we adopted the second method discussed in the previous section about the Chase decoding of non-binary codes. All the codewords transferred are represented using binary form.

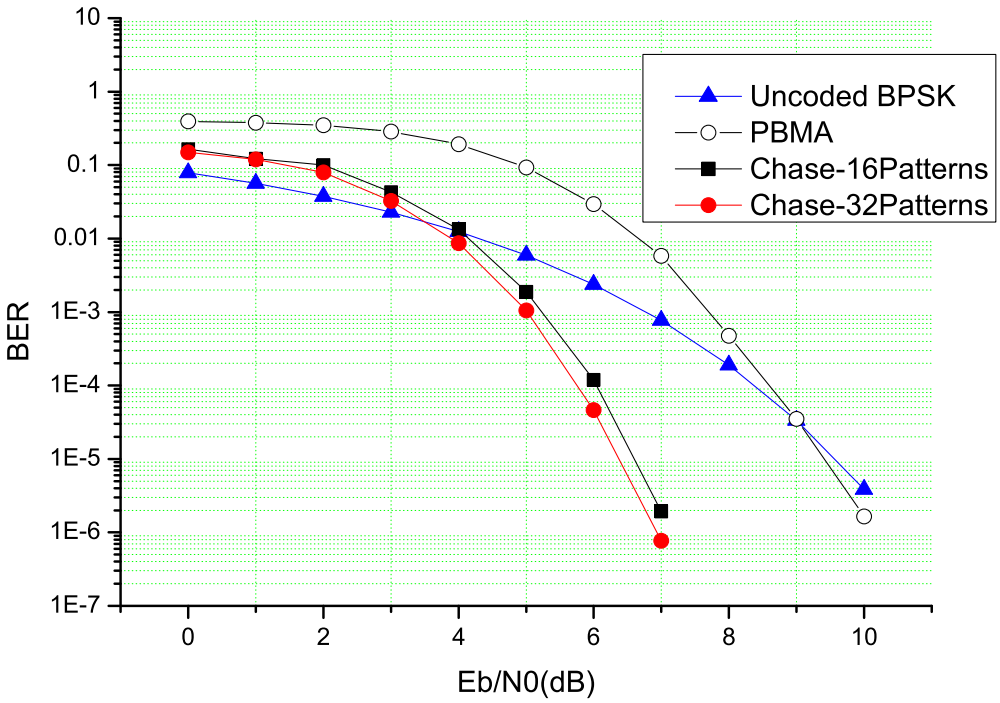

Figure 3.2: Performace of Chase Decoding of (23,18,3) AG codes over Klein quartic curves in AWGN channel

Figure 3.2 indicates the different bit error rate (BER) performance of the (23,18,3) algebraic geometric code using PBMA and the Chase algorithm operating on an AWGN channel with BPSK modulation. In this scenario, the number of test error patterns in the simulations are limited to 16 and 32 respectively. The Chase algorithm obtained a coding gain of about 3dB at $BER = 10^{-6}$ from the PBMA. And the Chase decoder with 32 error patterns can obtain 0.15dB from the Chase decoder with 16 error patterns. This shows that with 16 test error patterns, we can achieve a good trade-off between complexity and performance.

It is observed that the PBMA algorithm can only excel the uncoded BPSK system when the SNR is very high. As we have discussed in the above section, the error correcting ability of the PBMA algorithm is always with respect to symbol errors. However, binary transmissions are used in our simulations. For example, the design minimum distance of (23,18,3) AG codes over Klein quartic is 3, the hard-decision decoder(PBMA) can only correct 1 symbol errors. In binary form, the hard-decoder can correct 3 bit errors if these 3 bits who construct 1 symbol in 8-ary form. While if there are two bits errors who belong to different symbols, the hard-decision decoder might not be able to correct the bit errors. As a result, the BER performance of PBMA algorithm are worse than uncoded BPSK system when the SNR is relatively low. The Chase algorithm is able to improve the error-correcting ability of the PBMA decoder, but it also increases the computation complexity greatly.

The BER performance of the Chase decoder is greatly restricted by the error correcting ability of the hard-decision decoder. Reed-Solomon codes are MDS, while one-point AG codes are not. With similar binary code length and code dimension, the minimum distance of AG codes in binary representation is less than the minimum distance of Reed-Solomon codes in binary representation. As a result,

the performance of the Chase decoder for one-point AG codes is not as good as the performance of the Chase decoder for Reed-Solomon code of similar binary code length and code rate.

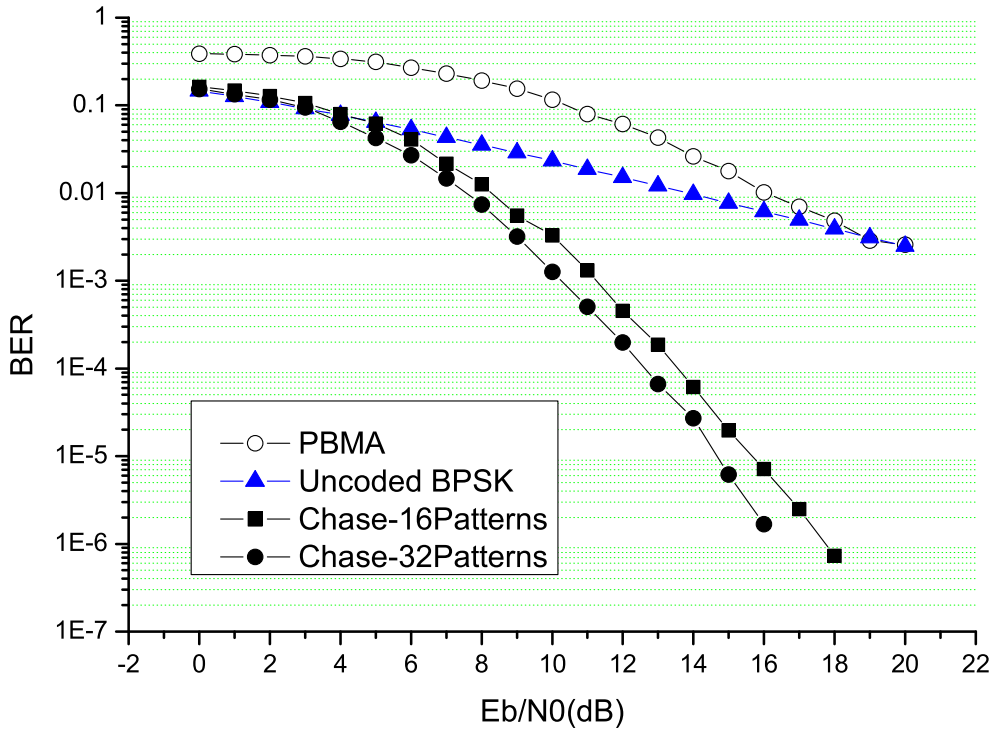

Figure 3.3: Performace of Chase Decoding of (23,18,3) AG codes over Klein quartic curves in Rayleigh fading channel

Figure 3.3 shows the BER performance of one-point AG codes using the Chase algorithm and PBMA operating on flat Rayleigh fading channel with BPSK modulation. The BER curve of PBMA shows that the hard-decision decoder almost can not obtain a reasonable coding gain even in a high SNR, like 20dB. However, if we using the Chase algorithm plus the PBMA decoder, the BER performance is greatly improved. The BER curves of the Chase decoding have a steep descend compared to the curves of corresponding to uncoded BPSK and PBMA hard-decision de-

coding. We can conclude that the Chase algorithm is a very useful soft-decision scheme in flat fading channels.

Besides the Chase algorithm, the ordered statistics decoding(OSD in short) algorithm [9] is also a soft-decision decoding algorithm which must work with a hard-decision decoder of block codes. Similar to the Chase algorithm, the OSD algorithm also has to compute the reliability values of each bit in the received word. However, unlike the Chase algorithm, which deals with the least reliable code positions, the OSD algorithm considers a list of codewords obtained from a set of most reliable information positions. For more details, readers can refer to [9].

3.6 Summary

In this chapter, we described the general information about the Chase algorithm. The Chase algorithm is designed to implement soft-decision decoding for all linear block codes with a hard-decision decoder. We adopt the Chase algorithm in conjunction with PBMA to realize soft-decision decoding for one-point AG codes. From the simulation results, we can conclude that the Chase algorithm could greatly improve the BER performance of one-point AG codes in both AWGN channel and Rayleigh fading channel.

Chapter 4

Block Turbo Codes

4.1 Introduction

In 1993, Berrou[2] introduced a new coding scheme, which can achieve an extraordinary low BER with a SNR per information bit (E_b/N_0) close to Shannon's limit on an AWGN channel. Their coding scheme consists of two recursive systematic convolutional codes concatenated in parallel and which are decoded using iterative maximum-likelihood decoding(MLD) of the component codes. So we can call the codes convolutional turbo codes(CTC) since they use convolutional codes as component codes. Later, R. Pyndiah[16][15] presented a new coding scheme for product codes constructed by linear block codes. We can call these codes block turbo codes(BTC). A new iterative soft-input soft-output decoder is used in this coding scheme. The product codes constructed from BCH codes and Reed-Solomon codes have produced good performance. In this chapter, we will use this scheme in decoding product codes whose component codes are one-point AG codes.

4.2 Product Codes

Codes obtained using multidimensional techniques have random-error and burst-error correcting ability. Product codes are constructed using multidimensional techniques. Product codes, also called iterated codes, are serially concatenated codes which were first introduced by Elias in 1954 [5].

Consider a $k_1 \times k_2$ matrix of information bits, the columns are encoded using an (n_1, k_1, d_1) linear systematic code $\mathbf{C_1}$, and the rows of the resulting $n_1 \times k_2$ bit matrix are encoded using an (n_2, k_2, d_2) linear systematic code $\mathbf{C_2}$. The resultant $(n_1 n_2, k_1 k_2, d_1 d_2)$ product code $\mathbf{C_1} \otimes \mathbf{C_2}$ has code rate $\frac{k_1 k_2}{n_1 n_2}$. The linear codes $\mathbf{C_1}$ and $\mathbf{C_2}$ are called constituent codes or component codes. In this thesis, we always use the same code to encoding both the columns and rows. The construction of a product code is shown in figure 4.1. In the product codes, both column and row component codes should be systematic codes. And the column codes and the row codes should place their systematic bits either both in their first k_1 and k_2 positions, or both in their last k_1 and k_2 positions.

As indicated by Elias [5], product codes can be decoded by sequentially decoding the rows and columns in order to reduce the computation complexity. In order to achieve optimum performance, SISO (Soft-Input Soft-Output) decoding of the column and row codewords and iterative decoding are used in the decoding of product codes. We will introduce these algorithms in the next section.

From Figure 4.1, it is obvious that the code length and code dimension of the product code are $n_1 \times n_2$ and $k_1 \times k_2$ respectively. For the minimum distance, we can prove it is $d_1 \times d_2$ below. The product codes are also linear code, so the minimum distance of the codes equals to minimum Hamming weight of all non-zero codewords.

First, we can obtain the result that any $k_1 \times k_2$ information matrix which is not

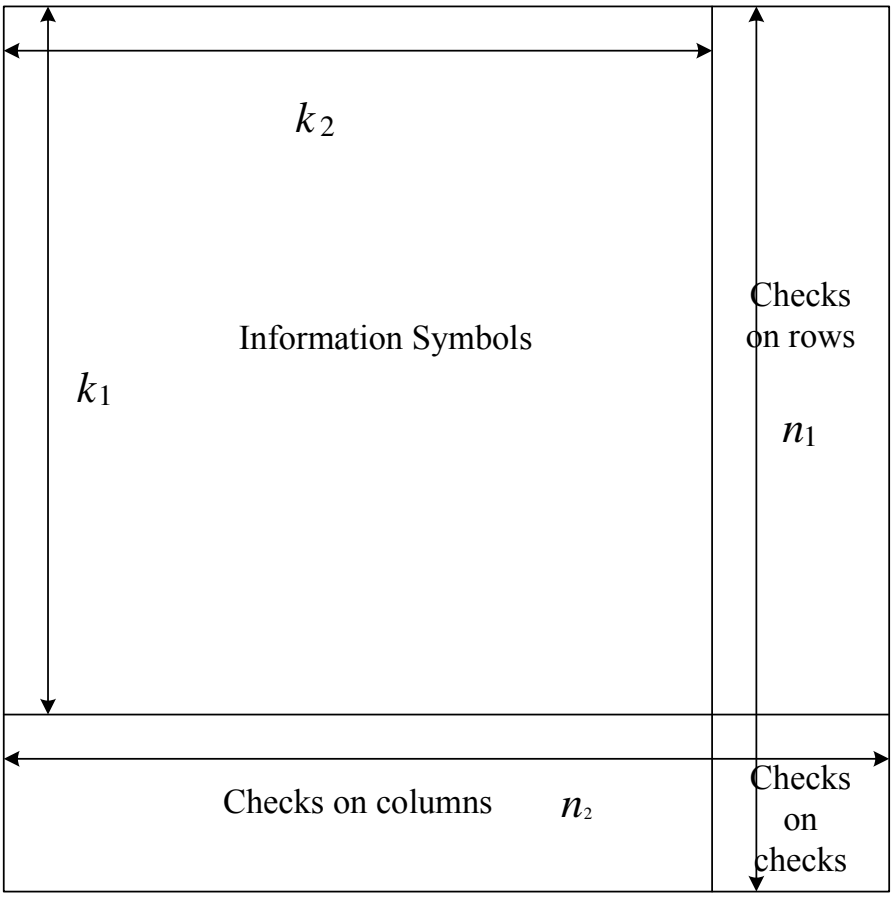

Figure 4.1: Product Codes

all zero, the encoded $n_1 \times n_2$ matrix would have a weight at least $d_1 \times d_2$. Consider the situation after the column encoding, since the $k_1 \times k_2$ information matrix is not all zeros, the resultant $n_1 \times k_2$ matrix has at least 1 column is not all zero. Because the minimum distance of $\mathbf{C_1}$ is d_1, the $n_1 \times k_2$ matrix would have at least d_1 rows are not all zeros. Then consider the case after the row encoding, since the minimum distance of $\mathbf{C_2}$ is d_2, the resultant $n_1 \times n_2$ matrix has at least d_1 rows are not all zeros, and each of these rows would have at least d_2 positions are non-zero. The weight of the $n_1 \times n_2$ matrix is at least $d_1 \times d_2$. From the discussion above, we can conclude that the minimum distance of the product codes $d \geq d_1 \times d_2$.

Second, we can construct a non-zero product codeword with weight $d_1 \times d_2$. We

assume that the code $\mathbf{C_1}$ and $\mathbf{C_2}$ arrange their systematic symbols in their first k_1 and k_2 positions. For code $\mathbf{C_1}$, let $\overline{c_1}$ be the non-zero codeword with minimum Hamming weight d_1, and the non-zero positions in this codeword are $\{x_1, x_2, \ldots, x_{d_1}\}$. For code $\mathbf{C_2}$, let $\overline{c_2}$ be the non-zero codeword with minimum Hamming weight d_2, and the non-zero positions in this codeword are $\{y_1, y_2, \ldots, y_{d_2}\}$. We set the position (x_i, y_j) to 1 in the $n_1 \times n_2$ matrix, where $i = 1, 2, \ldots, d_1$ and $j = 1, 2, \ldots, d_2$. This matrix is a valid product codeword, and the Hamming weight of this codeword is $d_1 \times d_2$.

According to the discussion above, it can be shown that the minimum distance of the product code $\mathbf{C_1} \otimes \mathbf{C_2}$ is $d_1 \times d_2$.

There are two methods have been used for the construction of non-binary product codes. Using the first methods each product code contains $k_1 \times k_2$ Q-ary information symbols as the information matrix. Here the component codes $\mathbf{C_1}$ and $\mathbf{C_2}$ should be define over $\mathbb{F}_Q$ and $Q = 2^q$. This product code can be transformed to binary representation as figure 4.2. In this case, each symbol is represented by q bits. The number of matrix rows is increased q times. Each row represents a row codeword, while q columns represent a column codeword. Here each bit is encoded in row-encoding and column-encoding with the same neighbour bits to form a Q-ary symbol.As a result, this method is always called symbol concatenation.

Using the second method, each product code contains $k_1 q \times k_2 q$ information bits as the information matrix. As shown in figure 4.3, each column of $k_1 \times q$ bits is a column codeword, and each row of $k_2 \times q$ bits is a row codeword. Here each bit is encoded in row-encoding and column-encoding with different neighbour bits to form a Q-ary symbol. This method is called bit concatenation.

With the same component codes, bit-concatenated product code has the same code rate as the symbol-concatenated product code. However, the block length of

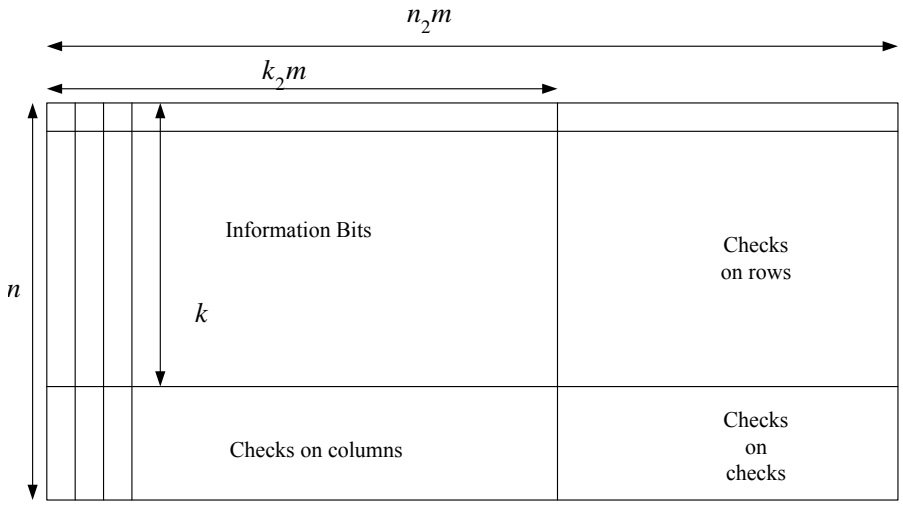

Figure 4.2: Non-binary Product Codes with Symbol Concatenation

bit-concatenated product code is q times of that of symbol-concatenated product code.

In the simulations presented in this chapter, the algebraic geometric codes defined over Klein quartic curve are used as the component codes of the product codes. Both symbol concatenation and bit concatenation will considered.

4.3 Iterative Chase Decoding of Product Codes

In the previous chapter, we have introduced the basic concepts of soft-decision decoding and investigated the performance of the Chase decoder for one-point AG codes. Assume $\overline{r} = (r_1, r_2, \ldots, r_n)$ is the received values from the output of an AWGN channel with transmitted binary codeword $\overline{x} = (x_1, x_2, \ldots, x_n)$.

$$\overline{r} = \overline{x} + \overline{n} \tag{4.1}$$

where the elements of $\overline{n}$ are AWGN samples with standard deviation σ. Let $\overline{y} = (y_1, y_2, \ldots, y_n)$ be the hard-decision codeword. As mentioned in the previ-

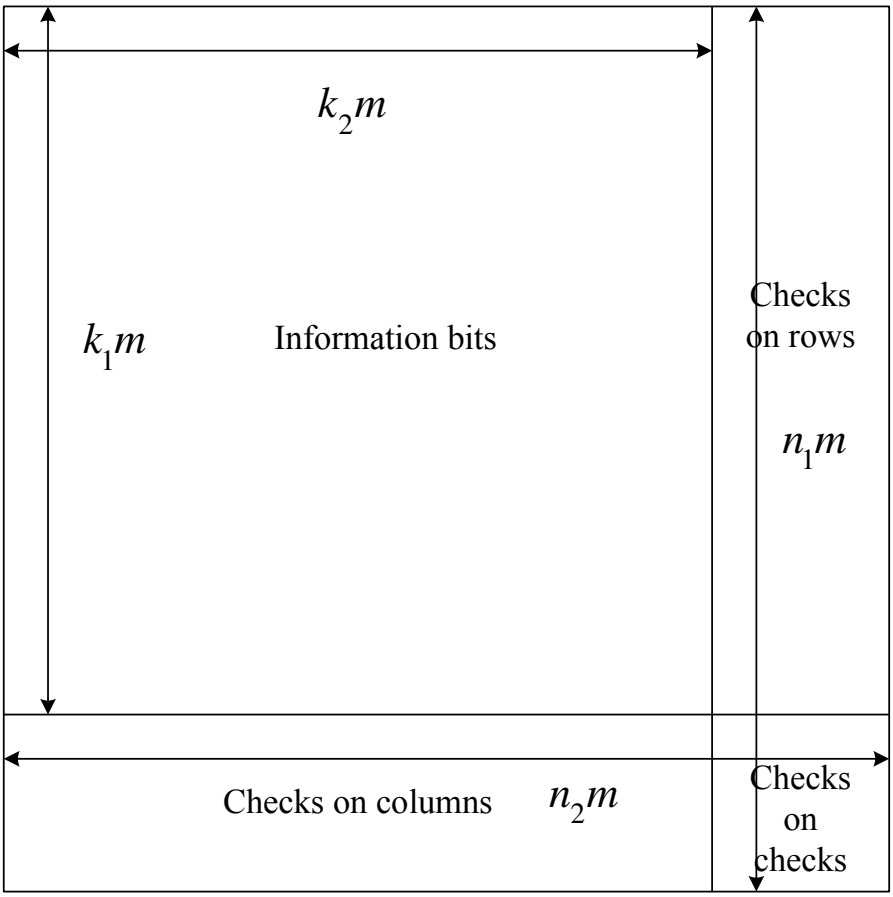

Figure 4.3: Non-binary Product Codes with Bit Concatenation

ous chapter, the reliability of component y_i is defined using the log-likelihood-ratio

$$\Lambda(y_i) = \ln(\frac{Pr(x_i = +1|r_i)}{Pr(x_i = -1|r_i)}) = (\frac{2}{\sigma^2})r_i. \tag{4.2}$$

If we normalize the LLR with respect to $\frac{2}{\sigma^2}$, the reliability of y_i is give by $|r_i|$.

Chase algorithm can yield a decision codeword $\overline{d}$ for each received word $\overline{r}$. In order to compute the soft-output, we have to compute the reliability of each bit of the decision $\overline{d}$ given the received word $\overline{r}$.

4.3.1 Reliability of a Decision Given by Soft-Input Decoder

Let's consider the j-th component d_j of the decision $\overline{d}$, the value of d_j should be 1 or -1. Then from Chase algorithm, we know $\overline{d}$ should be the closest codeword to $\overline{r}$ in Euclidean distance sense. Let $\overline{c}$ the closest codeword to $\overline{r}$ and $d_j \neq c_j$ is satisfied. It is obvious that $\overline{c}$ is farther from $\overline{r}$ than $\overline{d}$.

The reliability of d_j can be expressed as [15]

$$\Lambda(d_j) = \frac{2}{\sigma^2}(|\overline{r} - \overline{c}|^2 - |\overline{r} - \overline{d}|^2) \tag{4.3}$$

After normalization and simplification, we can get the soft-output as

$$\Lambda'(d_j) = r_j + \sum_{i=1, i\neq j, d_i \neq c_i}^{n} r_i d_i = d_j \sum_{i=1, d_i \neq c_i}^{n} r_i {d_i}^n r_i d_i \tag{4.4}$$

The term

$$w_j = d_j \sum_{i=1, i\neq j, d_i \neq c_i}^{n} r_i d_i \tag{4.5}$$

can be viewed as a correction term to the soft-input r_j. This term plays a role as the extrinsic LLR in the classic convolutional turbo codes.

So in the iterative Chase decoding, each element of the soft-output is computed base on the received word $\overline{r}$ and two codewords $\overline{c}$ and $\overline{d}$ from the output candidate codeword list of the original Chase decoder.

In practice, we always use a scale factor α_i in the computation of the soft-output.

$$r'_j = r_j + \alpha_i w_j \tag{4.6}$$

The factor α_i is used to compensate for the difference of variances of r_j and r'_j where i is referred to the iteration stage. The value α_i would be different in different iteration stages.

Consider the case that there is no other codeword in the candidate list that satisfies $d_j \neq c_j$. We can generate an estimation extrinsic LLR value by

$$w_j = \beta_i d_j. \tag{4.7}$$

Here β_i should be an estimation value of $|\log(\frac{Pr(d_j correct)}{Pr(d_j incorrect)})|$.

Based on the soft-output computation method above, we can implement the iterative Chase decoding. The flow chart of iterative chase decoding is shown in figure 4.4. Chase algorithm is not the only SISO algorithm used in the iterative decoding of product codes. A Soft-Input Soft-output ordered statistic decoding algorithm proposed in [10] can also be applied to the iterative decoding of product codes.

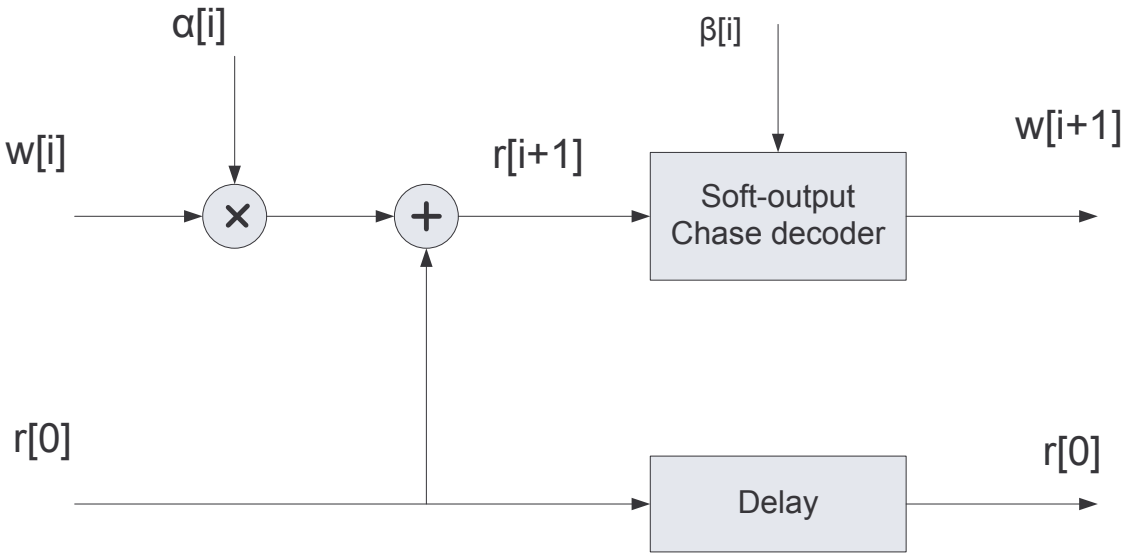

Figure 4.4: The Flowchart of Iterative Chase Decoding

4.3.2 Codeword Validation

In the iterative Chase decoding scheme, the soft-output of each received word is determined by two factors. One is the received symbol values from channel. The other one is the candidate codeword list generated by the Chase decoder. In previous chapter, we have discussed the error correcting ability of PBMA. This algorithm can correct $\lfloor \frac{d-1}{2} \rfloor$ errors, where d is the design minimum distance of the one-point AG codes.

As a result, when decoding using the Chase algorithm, if the symbol errors beyond the error correcting ability, the output candidate codeword would not be closest codeword from the input to the hard-decision decoder. Some output candidate codeword might be far from the input codeword. In iterative Chase decoding algorithm, each candidate codeword would be used to the computation of soft-output. Such fail-decoded codeword would greatly influence the soft-output. As shown in Example 3.1, the PBMA decoder failed to correct the error in that situation, and the output codeword is far from both the received word and transmitted codeword. If such codewords exist in the list of candidate codeword, the extrinsic information generated by Equation 4.5 would be inaccurate.

To reduce the bad influence, we should delete these fail-decoded codeword. A simple method is compare the input and output of the hard-decision decoder, if they have more than d different symbols, we believe it is a fail-decoded codeword and delete it.

4.3.3 Parameter Settings

The setting of the parameters $\overline{\alpha}$ and $\overline{\beta}$ will greatly influence the performance of the iterative Chase decoder. In practice, we can determine there parameters by experiments. In the simulations, both the elements of $\overline{\alpha}$ and $\overline{\beta}$ should be increased gradually from 0 to 1.0.

In [14] and [13], these coefficients could be computed adaptively based on the statistics of the processed codewords. Although the performance would be better, the computation complexity is increased greatly.

In summary, our iterative decoding algorithm for algebraic geometric codes with soft-output based on a set of codewords produced by Type-II algorithm is as follows:

Step 1: Initialization Set iteration counter $i = 0$. For each column or row of the product code, Let $\overline{r}[0]$ be the received channel value $\overline{r}$.

Step 2: Soft Input For each column or row of the product code, $j = 1, \ldots, n$, let $\overline{r_j}[i+1] = \overline{r}[0] + \alpha_i \overline{w_j}[i]$.

Step 3: Chase Algorithm For each column or row of the product code, $j = 1, \ldots, n$, execute the Type-II Chase algorithm with soft-input $\overline{r_j}[i+1]$ and obtain a list of candidate codeword.

Step 4: Codeword Validation Compare each codeword of the list generated by Step 3 with the hard-decision received word. Delete those codewords whose distance from the received word are larger than the designed minimum distance of the component AG code. Arrange the list in descending order with respect to the metric of each codeword.

Step 5: Extrinsic information For each column or row of the product code, generate the extrinsic information. For each bit position of the column or row, if there are at least two codeword different in the position, calculate the extrinsic information using Equation 4.5. If all codewords in the list are identical in a position, calculate the extrinsic information using Equation 4.7.

Step 6: Soft Output Let $i = i + 1$, if i is less than the maximum number of iterations, then go to step 2. Else calculate the soft-output, For $j = 1, \ldots, n$, $\overline{u_i} = \overline{r}[0] + \alpha_i \overline{w_j}[i]$.

4.4 Simulation Results and Discussions

In this section, the simulation results of the iterative Chase decoding of both bit-concatenated and symbol-concatenated one-point AG product codes in AWGN

channel and flat Rayleigh fading channel are shown.

Figure 4.5 shows the BER performance of the bit-concatenated AG block turbo codes in AWGN channel, while Figure 4.6 shows that of the symbol-concatenated AG block turbo codes in AWGN channel. In these simulations, for each column or each row, the number of the test error patterns is 32. The number of the iterations is set to 8.(The column decoding, or the row decoding, is considered as a half iteration) The coefficients $\overline{\alpha}$ and $\overline{\beta}$ are selected as

$$\overline{\alpha} = (0, 0.1, 0.2, 0.25, 0.3, 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.9, 1.0, 1.0) \quad (4.8)$$

$$\overline{\beta} = (0.2, 0.3, 0.4, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.9, 1.0, 1.0, 1.0, 1.0). \quad (4.9)$$

The code rate of our AG product codes is $(\frac{18}{23})^2 = 0.6125$.

Figure 4.7 shows the BER performance of the bit-concatenated AG block turbo codes in AWGN channel, while figure 4.8 shows that of the symbol concatenated AG block turbo codes in Rayleigh fading channel. In these simulations, for each column or each row, the number of the test error patterns is 32. The number of the iterations is set to 8.(The column decoding, or the row decoding, is considered as a half iteration) The coefficients α and β are selected as

$$\overline{\alpha} = (0, 0.1, 0.2, 0.25, 0.3, 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.9, 1.0, 1.0) \quad (4.10)$$

$$\overline{\beta} = (0.2, 0.3, 0.4, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.9, 1.0, 1.0, 1.0, 1.0). \quad (4.11)$$

From the simulation results above, we find that both in fading channel and AWGN, the performance of bit-concatenated product code is slightly better than that of symbol-concatenated product code. The reason might be the bit-concatenated code has longer block length.

The performance of AG product codes are slighted worse than that of Reed-Solomon or BCH product codes. Based on the simulation results of [15], we can get

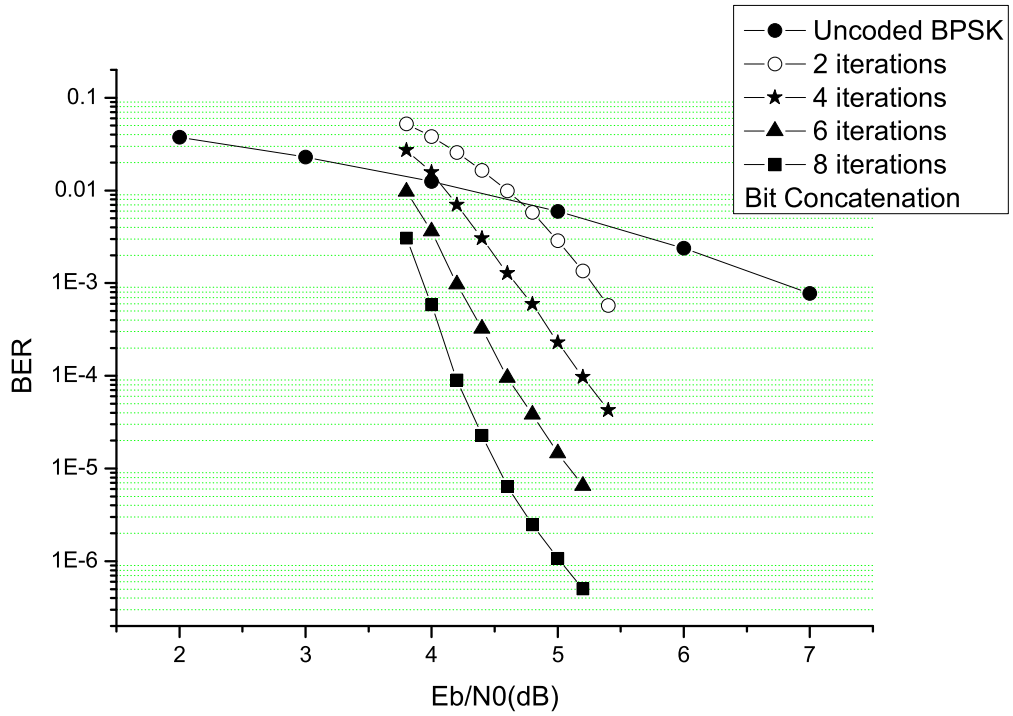

Figure 4.5: Performance of Iterative Chase Decoding of (23,18,3) AG codes over Klein quartic curves in AWGN channel using bit concatenation

the performance of the iterative Chase decoding of Reed-Solomon and BCH product codes. For the iterative Chase decoding BCH product codes, the performance will not improve greatly after 4 iterations. While for the iterative Chase decoding of AG product codes, we have to run at least 8 iterations to get relatively good BER performance. The BER performances of BCH product codes and AG product codes are compared in Table 4.1, where the $\frac{E_b}{N_0}$ column indicates the SNR where the BER achieves 10^{-5}. In [1] and [15], there are more performance curves and simulation results about the iterative Chase decoding of Reed-Solomon product codes and BCH codes.

There are two reasons. One is that the error correcting ability of the hard-

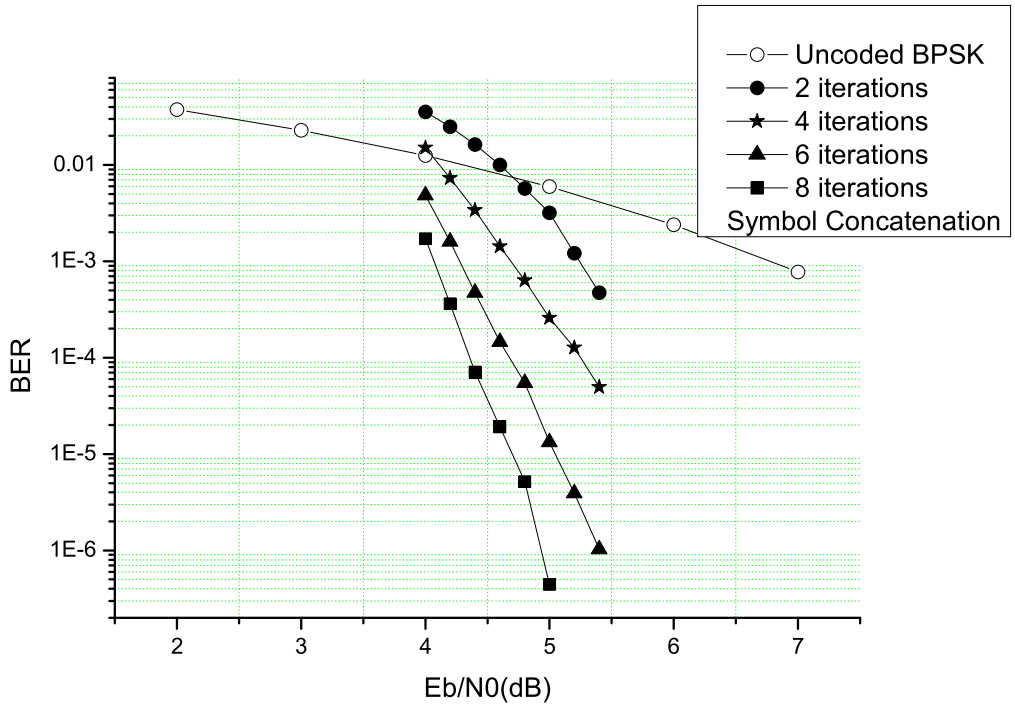

Figure 4.6: Performance of Iterative Chase Decoding of (23,18,3) AG codes over Klein quartic curves in AWGN channel using symbol concatenation

decision decoder restricts the performance of AG codes. The other is that the RS codes are MDS, while AG codes are not MDS, and are even farther from MDS than BCH codes. Let's compare the minimum distance and dimension of the one point AG codes over Klein quartic curves with those of BCH codes with similar binary code length. The binary code length of the one-point AG codes we used in simulations is 69, the code parameters are shown in Table 4.2. We select narrow sense binary BCH code with length 63, whose parameters are shown in Table 4.3.The code rate of the AG code we used as the component codes of the product code in simulation is 0.7826, and the length of its binary representation is 69. As we have discussed in previous chapter, the PBMA hard-decoder can only

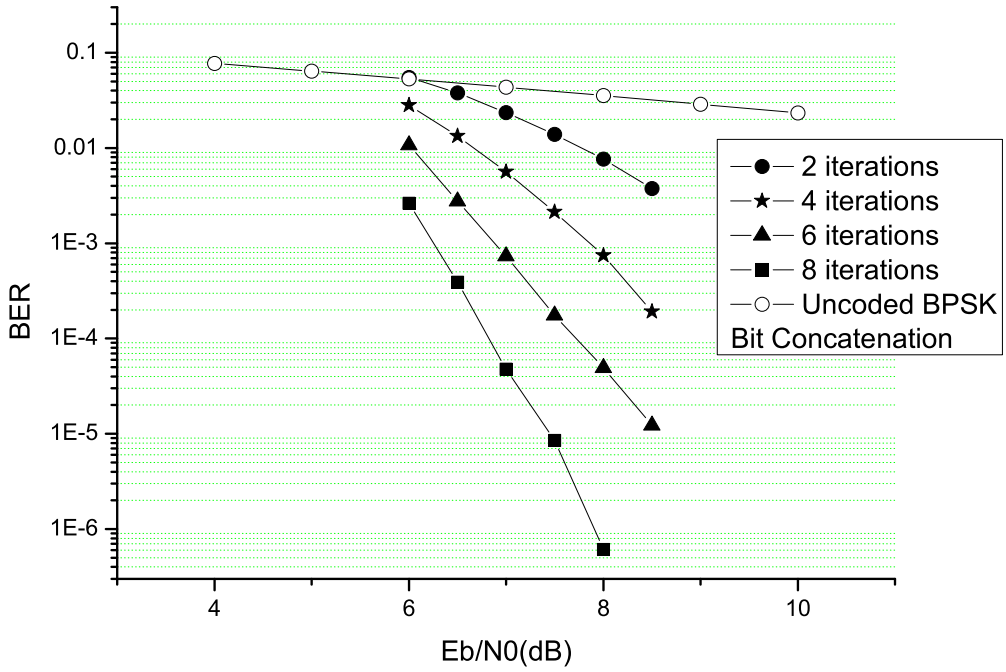

Figure 4.7: Performance of Iterative Chase Decoding of (23,18,3) AG codes over Klein quartic curves in Rayleigh channel using bit concatenation

guarantee to correct 1 bit error in any position. From Table 4.3, we select the $(63, 51)$ BCH code, whose code rate is 0.8095. The hard-decoder of the BCH code and correcting 2 bit errors in without position restriction, although the code length is a bit shorter, and the code rate is a bit higher than those of the AG code we use. We can conclude that the one point AG codes over Klein quartic curves are less MDS than BCH code when the code rate is high. Besides, the hard-decoder for BCH codes can correcting errors with respect to bit, while the hard-decoder for one-point AG codes over Klein quartic curves could only correcting errors with respect to symbols. In a other word, the hard-decoder's bit error correcting ability not only restrict by the bit error numbers, but also by the bit error positions. In previous chapter, Example 3.1 has shown this problem.

Product Codes	Code Rate	$\frac{E_b}{N_0}$	Channel
BCH $(63, 51, 5)^2$	0.8095	2.8	AWGN
AG $(23, 18, 3)^2$	0.7826	4.5	AWGN
BCH $(63, 51, 5)^2$	0.8095	7.0	Rayleigh Fading
AG $(23, 18, 3)^2$	0.7826	7.4	Rayleigh Fading

Table 4.1: Performance Comparison of BCH product codes and AG product codes

dimension k	designed minimum distance d	error correcting ability t
18	3	1
17	4	1
16	5	2
15	6	2
14	7	3
13	8	3
12	9	4
10	10	4
9	11	5
8	12	5

Table 4.2: Code Parameter of One-point AG Codes over $\mathbb{F}_8$

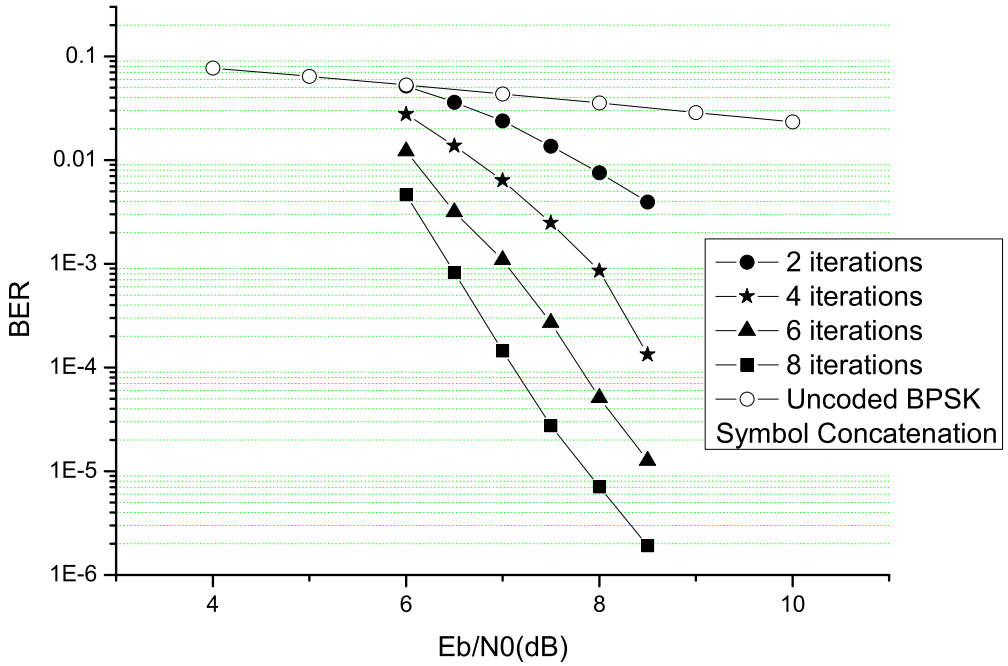

Figure 4.8: Performance of Iterative Chase Decoding of (23,18,3) AG codes over Klein quartic curves in Rayleigh channel using symbol concatenation

One important remark concerning the performance curves above in AWGN channel is that using symbol concatenation, the iterative Chase decoding of the AG product codes can guarantee the mitigation of error phenomena in high SNR region. However, using bit concatenation, this phenomena exists in high SNR region.

The computation complexities of the iterative Chase decoding of the product codes are mainly determined by the complexity of the hard-decision decoder of the component codes and two parameters of the algorithm. One parameter is the number of the error patterns used in the Chase decoding step of each component code. The other parameter is the number of the iterations.

dimension k	designed minimum distance d	error correcting ability t
57	3	1
51	5	2
45	7	3
39	9	4
36	11	5
30	13	6
24	15	7
18	21	10
16	23	11
10	27	13
7	31	15

Table 4.3: Code Parameter of BCH codes

4.5 Summary

In this chapter, we described the basic concepts of product codes. We adopt one-point AG codes as the component codes of product codes. We also implemented iterative Chase Algorithm for the decoding of one-point AG product codes. The simulation results are also discussed in this chapter.

Chapter 5

Conclusion

In this chapter, we attempt to sum up the results and discussions that were put forward in the previous chapters. In addition, the possible future research area is also included in the final part of this chapter.

5.1 Summary of Thesis

This thesis has been intended to investigate iterative Chase decoding algorithm for product codes, whose component codes are one-point AG codes, to achieve a better BER performance.

In Chapter 2, we briefly introduced the basic concepts of algebraic geometry. The definitions of the two class AG codes, functional codes and residual codes, are also explained. Furthermore, the construction method of one-point AG codes is presented.

In Chapter 3, we presented the Chase algorithm and examined the benefits of implement this algorithm for soft-decision decoding of one-point AG codes in both AWGN channel and Rayleigh fading channel. R. Koetter's parallel implementation of Berlekamp-Massey algorithm is used as the hard-decision decoder of AG codes.

Because AG codes is not a MDS codes, and the error-correcting ability of the hard-decision decoder of AG codes is not as good as that of Reed-Solomon codes, the BER performance of the Type-II Chase algorithm for AG codes is a bit worse than that of the Chase algorithm for RS codes with similar code rate.

In Chapter 4, we presented some important knowledge of product codes. We also briefly discuss two methods to construct non-binary product codes. Iterative Chase decoding algorithm for product codes are discussed in this chapter. One-point AG codes are used as the component codes of these product codes. Simulations of these product codes in both AWGN channel and Rayleigh fading channel were implemented. The performances of these product codes were also compared with other product codes constructed by BCH codes or Reed-Solomon codes. We also provided several reasons for the degradation of the performance of AG block turbo codes with respect to the performance of BCH block turbo codes and Reed-Solon block turbo codes.

5.2 Future Work

Although we have managed to reveal and propose the iterative Chase decoding algorithm for product codes composed of one-point AG codes in this thesis, we still highlight some promising work that could be done in the near future.

As we mentioned in previous chapter, the coefficient $overline\alpha$ and $\overline{\beta}$ will determine the performance of iterative Chase decoder. In [14] [13], adaptive methods to determine these coefficients were presented. We can apply similar method in the iterative Chase decoding of AG product codes.

In this thesis, we have shown that when represented in binary form, the one-point AG codes we used in simulations are even less MDS than binary BCH codes. And the hard-decision decoding algorithm of the AG codes with respect to symbols

will restrict the performance of iterative Chase decoding of the AG product codes. In future, we might develop new iterative decoding algorithm for the AG product codes in non-binary form.

As one of the most prominent advantages of algebraic geometric codes is that asymptotically good code could be constructed. In future research, we can employ our algorithm in decoding the product codes, whose component codes are other AG codes with longer code length, and defined over larger alphabet.

Bibliography

[1] O. Aitsab and R. Pyndiah, "Performance of Reed-Solomon Block Turbo Codes," in *Proceedings of IEEE GLOBECOM*, vol. 1/3, London, UK, Nov. 1996, pp. 121–125.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *Proc. of IEEE Intl. Conf. On Communications*, May 1993, pp. 1064–1070.

[3] I. Blake, C. Heegard, T. Høholdt, and V. Wei, "Alegebraic-Geometry Codes," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2596–2618, October 1998.

[4] D. Chase, "A Class of Algorithms for Decoding Block Codes With Channel Measurement Information," *IEEE Transactions on Information Theory*, vol. IT-18, no. 1, pp. 170–182, January 1972.

[5] P. Elias, "Error-Free Coding," *IRE Transcations on Information Theory*, vol. IT-4, pp. 29–37, September 1954.

[6] N. D. Elkies, "Beyond Goppa Codes," in *Proceedings 35th Allerton Conference Communications, Control and Computing*, 1997.

[7] G.-L. Feng and T. Rao, "Improved Geometric Goppa Code, part i: Basic Theory," *IEEE Transactions on Information Theory*, vol. 41, pp. 1678–1693, November 1995.

[8] G. D. Forney, "Generalized Minmum Distance Decoding," *IEEE Transactions on Information Theory*, vol. IT-12, pp. 125–131, April 1966.

[9] M. Fossorier and S. Lin, "Soft-Decision Decoding on Linear Block Codes Based on Ordered Statisics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, September 1995.

[10] ——, "Soft-Input Soft-Output Decoding of Linear Block Codes Based on Ordered Statisics," in *Proceedings of IEEE GLOBECOM*, 1998, pp. 2828–2833.

[11] R. Koetter, "A Fast Parallel Implementation of a Berlekamp-Massey Algorithm for Algebraic-Geometric Codes," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1353–1368, July 1998.

[12] R. Koetter and A. Vardy, "Algebraic Soft-Decision Decoding og Reed-Solomon Codes," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2809–2825, Novermber 2003.

[13] P. Martin and D. P. Taylor, "On Adaptive Reduced-Complexity Iterative Decoding," in *Proceedings of IEEE GLOBECOM*, 2000, pp. 772–776.

[14] A. Picart and R. Pyndiah, "Adapted Iterative Decoding of Product Codes," in *Proceedings of IEEE GLOBECOM*, vol. 5, 1999, pp. 2357–2362.

[15] R. Pyndiah, "Near-Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 1003–1010, August 1998.

[16] R. Pyndiah, A. Glavieux, and S. Jacq, "Near Optimum Decoding of Product Codes," in *Proceedings of IEEE GLOBECOM*, vol. 1/3, San Francisco, CA, Nov.-Dec. 1994, pp. 339–343.

[17] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *SIAM Journal on Applied Mathematics*, vol. 8, no. 300-304, 1960.

[18] M. A. Tsfasman, S. G. Vlådut, and T. Zink, "Modular curves, shimura curves and goppa codes, better than varshamov-gilbert bound," *Math. Nachrichten*, vol. 109, pp. 21–28, 1982.

[19] T. Yaghobian and I. F. Blake, "Hermitian Codes as Generalized Reed-Solomon Codes," *Des., Codes Cryptogr.*, vol. 2, pp. 5–17, 1992.