NAIST-IS-DT0061001

Doctor's Thesis

An Algebraic Maximum Likelihood Decoding Algorithm and a Sub-Optimum Decoding Algorithm for Linear Codes

Daisuke Ikegami

February 7, 2003

Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

Doctor's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Daisuke Ikegami

Thesis committee: Hiroyuki Seki, Professor
Minoru Ito, Professor
Kenji Sugimoto, Professor
Yuichi Kaji, Associate Professor

An Algebraic Maximum Likelihood Decoding Algorithm and a Sub-Optimum Decoding Algorithm for Linear Codes*

Daisuke Ikegami

Abstract

Decoding binary linear block codes is one of the most fundamental and significant topics in the study of error correcting codes. There are two measures to evaluate decoding algorithms: the performance of correcting errors and the complexity of the algorithms. These two measures are in a trade-off relation. A decoding scheme which maximizes the performance is called the maximum-likelihood decoding (MLD), however, the complexity for MLD grows exponentially to the size of codes. At the sacrifice of the performance of MLD, it is possible to reduce the complexity. An efficient decoding scheme whose complexity is reduced is called a sub-optimum decoding. In the first half of this thesis, a new sub-optimum decoding algorithm is discussed. Recently, Fossorier and Lin have proposed a remarkable sub-optimum decoding algorithm which is based on ordered statistics of the communication channel. Their algorithm is simple and shows near optimum performance. By combining techniques of MLD and Fossorier's algorithm, a new sub-optimum decoding algorithm is obtained in this thesis. In the last half of this thesis, a novel MLD algorithm is proposed. Our idea is to reduce the problem of MLD into an integer program with binary variables. Recently, Conti and Traverso have developed an algorithm to solve a certain class of integer programs by making use of Gröbner bases. In this thesis, Conti's algorithm is extended to solve integer programs which represent MLD. The new MLD algorithm first reduces MLD to an integer program, to which an extended Conti's algorithm is applied.

Keywords: error correcting code, maximum likelihood decoding, integer programming, Gröbner basis, soft-decision decoding

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT0061001, February 7, 2003.

線型符号の代数的最尤復号法及び準最尤復号法*

池上 大介

内容梗概

誤り訂正符号理論において、良い復号法を模索することは最も基本的かつ重要な課題である。誤り訂正符号の復号アルゴリズムには、誤り訂正能力と計算量の 2 つの評価尺度があり、これらは一般にトレード・オフの関係にある。最尤復号法は誤り訂正能力を極限まで追求するが、計算量は符号長の指数オーダで増大する。一方、誤り訂正能力を犠牲にして計算量を削減することが可能であり、そのような復号法は準最尤復号法と呼ばれる。本論文の前半では、準最尤復号について議論する。これまで多数の準最尤復号法についての研究が行われているが、中でも Fossorier と Lin によって提案された、通信路の順序統計量に基づく準最尤復号法に注目する。この準最尤復号法は単純で、かつ最尤復号法と極めて近い誤り訂正能力を持つ。本研究では、最尤復号法と Fossorier の復号法を組合せることで、新しいタイプの準最尤復号法を構成した。一方、本論文の後半では、最尤復号の問題を 2 元整数計画問題に帰着させるというアイデアに基づき、新しいタイプの最尤復号アルゴリズムを提案する。近年、 Conti と Traverso はグレブナ基底を利用して、ある種の制約をみたす整数計画問題を解くアルゴリズムを提案した。本論文では、Conti のアルゴリズムを最尤復号に対応する整数計画問題のクラスに拡張し、これを適用することで最尤復号法を実現する手法を示す。

キーワード

誤り訂正符号, 最尤復号, 整数計画, グレブナ基底, 軟判定復号

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DT0061001, 2003 年 2 月 7 日.

Acknowledgements

I am grateful to Prof. Hiroyuki Seki for being my adviser. His technical and general guidance has supported me in my personal and professional growth.

I would like to thank Prof. Minoru Ito for providing me with valuable and beneficial suggestions in this research. The technical guidance and constant encouragement of Prof. Kenji Sugimoto throughout my study.

I consider myself privileged to have supervised by Prof. Yuichi Kaji. His formidable insight and unfailing enthusiasm have proved a fertile source of stimulation and encouragement. He has always been generous with his time, listening carefully and reviewing fairly.

During my three short years in Takayama-cho, I have often benefited from the expertise of others. I would like to thank Yuansheng Tang and Prof. Toru Fujiwara, and Prof. Tadao Kasami for their noteworthy comments. Kiyoshi Shirayanagi gives me kindly innumerable informative comments and discusses about our mathematical problems devotedly. Prof. Carlo Traverso encourages me in the main work very much. Prof. Akio Fujiwara gives me not only important comments for my works but also great fun of study of the quantum information. Prof. Nobuki Takayama gives me apt comments about computing Gröbner bases. Prof. Yoshihide Watanabe always encourages me in my studies. I also thank Toshiyuki Ishida, Masafumi Ito, Takayuki Ishizeki, Tomoharu Shibuya and Ryutaroh Matsumoto for their constructive criticism for my works.

Especially, I remember strongly that Hidefumi Ohsugi, Tomonori Kitamura , Prof. Takayuki Hibi and I share precious time for a common mathematical problem. We are very proud of our variety works about algebra and coding theory.

Of course, life at NAIST has not entirely revolved around the Seki laboratory. I never forget their kindness. I thank again Prof. Hiroyuki Seki for his devoted help when I was in distressfor study and had a terrible stomachache. At various points, I have had cause to be thankful to Yoshizumi Noda, Yuki Kato, Tan Soon Keong, Fumihiko Tamura, Takao Nishi, Yuji Tsuchida, Yuko Ohsumi, Motoharu Baba, Yoshiyuki Nakagawa and Junichi Funasaka.

This thesis is dedicated to my family, whose importance to me shall not try to put into words.

List of publications

1. Publications related to this thesis

1.1 Journal papers

(1) D. Ikegami and Y. Kaji: Maximum likelihood decoding for linear block codes using Gröbner bases, to appear in IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, volume E86-A, number 3, March, 2003.

(2) D. Ikegami and Y. Kaji: On the construction of ideals in the Conti-like MLD algorithm, submitted to IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences.

1.2 International conferences (reviewed)

(3) D. Ikegami and Y. Kaji: The soft-decision MLD of linear block codes, integer programming and Gröbner bases, Proceedings of 2002 International Symposium on Information Theory (ISIT 2002), page 144, Lausanne, Switzerland, June, 2002.

(4) Y. Kaji and D. Ikegami: Decoding linear block codes using the ordered-statistics and the MLD techniques, Proceedings of 2002 International Symposium on Information Theory (ISIT 2002), page 316, Lausanne, Switzerland, June, 2002.

1.3 Workshops

(5) D. Ikegami and Y. Kaji: Decoding linear block codes using the ordered statistics and MLD techniques, Proceedings of the 24th Symposium on Information Theory and Its Applications (SITA 2001), volume II, pages 459–462, December, 2001.

(6) D. Ikegami and Y. Kaji: A soft-decision MLD algorithm for linear block codes using Gröbner bases, Proceedings of the 24th Symposium on Information Theory and Its Applications (SITA 2001), volume II, pages 545–548, December, 2001.

(7) D. Ikegami: Maximum likelihood decoding algorithm using Gröbner bases, Project Research for Advantage of Theory and Application of Gröbner Bases at Research Institute for Mathematical Sciences, Kyoto University (KURIMS Report 1289), pages 110–121, May, 2002 (in Japanese).

(8) D. Ikegami: Gröbner bases of zero dimensional ideals for zero-one integer programming, reprint, also included in Proceedings of the 11th Symposium on Japan Society for Symbolic and Algebraic Computation (JSSAC 2002), September, 2002.

(9) D. Ikegami and Y. Kaji: Sub-optimum soft-decision decoding for linear block codes using Gröbner bases, Proceedings of the 25th Symposium on Information Theory and Its Applications (SITA 2002), volume I, pages 15–18, December, 2002.

2. Other publications

2.4 Journal paper

(10) H. Ohsugi, D. Ikegami, T. Kitamura and T. Hibi: Gröbner bases of certain zero-dimensional ideals arising in coding theory, to appear in Advances in Applied Mathematics, 2003.

2.5 Workshops

(11) T. S. Keong, D. Ikegami and Y. Kaji: An evaluation method of the error performance of linear codes using coset partition, IEICE Technical Report, IT2000-60, volume 100, number 689, pages 91–98, March, 2001 (in Japanese).

(12) Y. Watanabe, Y. Noda, Y. Kato and D. Ikegami: Computation of Gröbner bases of toric ideals with the computer algebra system *Asir*, reprint, also included in Proceedings of the 11th Symposium on Japan Society for Symbolic and Algebraic Computation (JSSAC 2002), September, 2002 (in Japanese).

CONTENTS

List of figures

List of tables

List of algorithms

1. Introduction

1.1 Error correcting codes and their decoding

Error correcting code is a well-known and widely-accepted technique to realize a reliable communication system of digital information over noisy communication channels. Linear block codes are families of error correcting codes which are easy to handle with, and have been studied deeply. The most important problem concerning linear block codes is, both from theoretical and practical viewpoints, realization of an efficient decoding algorithm.

Decoding of an error-correcting code is to correct errors in a received sequence. In other words, decoding is a problem to estimate the *most likely* transmitted codeword from the received sequence which is observed from a noisy channel. Here we introduce some elementary notations for error-correcting codes. We consider binary linear block codes through the thesis. A codeword $\mathbf{c} = (c_1, \ldots, c_n) \in \{0, 1\}^n$ is said to have *length* n. A linear vector space which consists of codewords is also called a *linear block code*. If k is the dimension of the code, we call the code an $[n, k]$ code. Since this code uses n symbols to send k message symbols, it is said to have *rate* $R = k/n$. A codeword $\mathbf{c}$ of an error correcting code is said to be the most likely codeword for a received sequence $\mathbf{r}$ if $\mathbf{c}$ maximizes $\mathcal{P}(\mathbf{r} \mid \mathbf{c})$, which is the probability of receiving $\mathbf{r}$ when the codeword $\mathbf{c}$ was transmitted, among all codewords of the code. Thus the primal purpose of decoding is to find the most likely codeword for a received sequence.

There are two approaches to realizing a decoding algorithm (or decoder) of error correcting codes. The first approach is to try to find the most likely codeword for a received sequence. A decoding scheme which is based on this approach is called the *maximum likelihood decoding (MLD)*. MLD is a powerful decoding scheme but it usually consumes a large amount of computing time and space. Indeed it is known that MLD is an NP-complete problem if arbitrary linear block codes are considered. For this reason, MLD for linear block codes is considered practical only if the code to be used is very small, say length 32 or less. In the second approach to realizing decoders,

we allow a decoder to make some mistakes. That is, a decoder successfully finds the most likely codeword in most cases, but it sometimes fails to find the correct most likely codeword. Of course this will degrade the reliability of communication, but the computational complexity for decoding is expected to be reduced. A decoding scheme which is based on this approach is called a *sub-optimum* decoding. Sub-optimum decoding is important from the practical reason, and a large number of algorithms for sub-optimum decoding have been proposed.

In the rest of this chapter, we briefly review some results on MLD and sub-optimum decodings. Outline of this thesis is also provided.

1.2 Maximum likelihood decoding

A straight-forward method for realizing MLD is by a brute-force exhaustive search. The brute-force approach to MLD of a binary linear block code requires the computation of 2^k conditional probabilities where k is the dimension of the code. The time required for this method rapidly becomes too large to implement a decoder as k increases and more effective methods are therefore needed.

Several researchers have presented techniques for decoding linear block codes. One common approach is by converting the decoding problem into a graph-oriented problem on a trellis diagram which is derived from the code. The best-known and most commonly used MLD algorithm is the Viterbi algorithm [14,34,39,52]. In the context of MLD, a trellis diagram is considered as a weighted and directed graph. The Viterbi algorithm is based on Dijkstra's algorithm which finds a minimum cost path from the start node to every other node of a weighted and directed graph. The Viterbi algorithm is a simple and efficient method for implementing MLD. This algorithm is especially good for convolutional codes [52], since the complexity of the Viterbi algorithm is proportional to the size of the trellis, and convolutional codes usually have a small and simple trellis diagram. However, the Viterbi algorithm is not appropriate for block codes with long length and large dimension because block codes have large, complicated and time-varying trellis diagrams in general. Therefore other efficient algorithms

are strongly needed.

Many efforts have been devoted to finding an efficient algorithm for MLD of linear block codes; a detailed bibliography of classical contributions in this area can be found in [6]. Recently, Han et al. proposed an algorithm which is now known as an A^* algorithm [19]. The A^* algorithm first re-orders received symbols according to their confidence values and then performs a tree search similar to the sequential decoding. The search is guided by a cost function which defines the present cost and estimated future contributions for each node of the search tree. Using the cost function, the algorithm tries to find the most likely codeword (or best leaf node) in an adaptive manner. This algorithm allows MLD to work on long block codes efficiently for high SNR's. In some bad cases which typically occur in low to medium SNR's, however, this algorithm requires both numerous computations and very large memory.

There are efforts to reduce the overall complexity of the trellis decoding by taking advantage of the decomposable structure of certain codes [6, 15, 31, 37]. Lafourcade and Vardy tried to reduce the computational complexity of the Viterbi algorithm by using the optimally sectionalized trellis diagram instead of the naive trellis [32]. It surely reduces the complexity, but the improvement is not so much and the trellis complexity still grows exponentially with the dimension of any sequence of good codes.

Fujiwara et al. noticed that if the trellis is "divided" into small sections, then each section contains many duplicated structures. By making use of this structural property of the trellis, they proposed an efficient algorithm which they call the recursive MLD algorithm [18]. Kaji et al. brought the strategy of a call-by-need computation (lazy evaluation) into the recursive MLD algorithm, and showed that the complexity can be extremely reduced [29].

1.3 Sub-optimum decoding

There are many sub-optimum decoding algorithm nowadays. The generalized minimum distance (GMD) decoding proposed by Forney [13] is one of the oldest sub-optimum decodings. The GMD decoding uses an algebraic decoder to produce a list

of likely codewords. For each codeword in the list, a test is then performed, with respect to a sufficient condition for optimality. The best codeword which passed the test is chosen as the decoded codeword.

Following an idea similar to the GMD decoding, Chase provided an algorithm which testifies a fixed number of error patterns, where the error patterns are generated systematically according to the reliability of received symbols [2]. A naive approach to generating the error patterns is to fix symbol positions, usually least reliable ones, and consider all error patterns which involve errors only in the fixed positions. For this algorithm, the maximum number of considered codewords and the error performance depend on the number of fixed positions. Chase's algorithm has then been modified so that the error patterns cover symbol positions which have less reliabilities than a predetermined threshold [49]. One drawback of this modification is that the maximum number of computations depends on both the choice of a threshold and the signal-to-noise ratio. It is unavoidable for sub-optimum decoding algorithms to have degradation in performance. In the case of the GMD and Chase algorithms, the performance degradation is not so much if the code has small rate. However, the degradation to MLD increases as the code rate increases.

Recently, a more improved algorithm based on the same idea has been proposed [30]. The algorithm tries to search good codewords iteratively. There is no limitation on the search space at the beginning of the algorithm, but at each iteration, a sufficient condition for optimality to terminate the algorithm is tested. After each test, the search space is reduced, and the iteration continues until the search space converges to a unique solution. The algorithm is also equipped with a termination criterion, which works effectively to reduce the computational complexity for short codes. However, the complexity of this algorithm still increases exponentially to the dimension of the code.

Another well-known technique is to perform syndrome decoding on the received sequence, and then use the syndrome information to modify and improve the original hard-decision decoding. In this approach, there is a room to consider a good strategy

for the search of the most likely codeword. In the original study [44], a simple strategy which makes use of the order of symbol reliabilities is considered. Different search schemes based on binary trees and graphs are presented in [35]. For an $[n, k]$ code, the methods presented in [35,44] both require that $n-k$ should be relatively small because the search is carried out over most of the column patterns of the parity check matrix of the code. For very high rate codes, it is possible to reduce the search space of [44] as shown in [43]. Roughly speaking, it is shown in [43] that we can predetermine a necessary and sufficient list of error patterns, based on the parity check matrix of the code and a partial ordering of the reliability measures. However, the technique becomes rapidly impractical whenever $n-k$ exceeds 8. Also we cannot induce enough number of general and effective conditions for error patterns to survive [16,42].

There are studies which make an MLD algorithm a sub-optimum one. For example, a sub-optimum version of the A^* algorithm has been devised where the maximum number of codeword candidates is limited by a threshold [16,20].

Among many sub-optimum decoding algorithms, the author is interested in the soft-decision decoding algorithm based on ordered statistics of the communication channel by Fossorier and Lin [16] because the algorithm achieves near optimum error performance with considerably small decoding complexity. Different from the Viterbi algorithm, the Fossorier's algorithm does not use trellis diagram of the code. In addition, the worst case complexity of the Fossorier's algorithm can be exactly evaluated while it is hard to evaluate the worst case complexity for other sub-optimum algorithms. The Fossorier's algorithm is simple and consists of the following three steps:(1) choose k symbol positions so that the chosen symbol positions constitute information symbol positions in a codeword, and the reliability of symbols at the corresponding positions in the received sequence is as large as possible, (2) make a binary vector of length k, say $\mathbf{a}$, as the hard-decision of the received symbols which correspond to the chosen k positions, and finally (3) encode $\mathbf{a}$ and obtain a codeword, which is the estimation of the transmitted codeword. Fossorier refers to this algorithm as the 0-th order reprocessing algorithm. If the vector $\mathbf{a}$ does not involve an error, then the 0-th order

reprocessing algorithm can estimate the correct transmitted codeword. However, if **a** involves errors, the result is not correct. To avoid this issue, Fossorier also investigated higher-order reprocessing algorithms. In the l-th order reprocessing algorithm, we examine every vector which has length k and whose Hamming distance from **a** is l or less. Thus $\sum_{i=0}^{l} \binom{k}{i}$ encoding operations are executed in the l-th order reprocessing algorithm in general. Fossorier and Lin have also proposed a condition to reduce the number of the computations (Theorem 4 in [16], which also improves the extended distance test introduced in [48]), though the computation of the l-th order reprocessing increases as the dimension k and the level l increase.

1.4 Outline of the thesis

In chapter 2 of this thesis, we consider a new algorithm for sub-optimum decoding [23, 28]. The algorithm takes an approach similar to Fossorier's 0-th order reprocessing algorithm. Instead of choosing k symbol positions, the proposed algorithm chooses t symbol positions with $t \leq k$. The received symbols at the chosen positions are quantized and t symbols in a codeword is decided. The estimation of the remaining part of the codeword is performed by an MLD algorithm because the remaining part can be regarded as a codeword in an punctured $[n - t, k - t]$ linear code. The error performance and the complexity are evaluated analytically as in the case of Fossorier's method. By computer simulation, we show that our extended Fossorier's algorithm has almost the same performance and complexity as the 2-nd order reprocessing. One of the aims of our approach is to provide much flexibility for Fossorier's algorithm. We show that the parameter t considered in the algorithm is more flexible than the Fossorier's parameter l in practice. In addition, since any MLD technique can be applied to our algorithm, if an efficient MLD algorithm is constructed, then the complexity of the proposed algorithm will be further improved.

In chapter 3, we propose a novel and quite different algorithm for MLD [25,26]. The essential idea is that MLD is equivalent to solving a certain integer program with binary variables. This idea itself is not very new. Many coding theorists have noticed the

relation between MLD and integer programming. However, solving an integer program is another very difficult problem in general. The aim of chapter 3 is to reduce MLD into a class of integer programs for which an effective algorithm has been devised. In [5], Conti and Traverso have developed a method of solving an integer program based on a Gröbner basis. The Conti's method consists of three major steps. The first step is to translate a given minimization (or maximization) problem of a target function into a minimization problem of a linear function with nonnegative coefficients. The constraint equations which the solution must satisfy are also modified in this step. The second step is to compute a Gröbner basis for an ideal of a polynomial ring which is defined from the constraint equations, and the final step is to compute the solution which minimizes the target function using the Gröbner basis.

We have extended the Conti's method for integer programs with modulo arithmetics, which is necessary to achieve MLD for binary linear block codes. Our key idea is the following two points: introduction of some binomials for modulo arithmetics in addition to generators of the Conti's proposed ideal, and consideration for an operation of matrices, also known as the Lawrence lifting, for non-positive target functions. Since the integer programs with binary modulo arithmetic include MLD for binary linear block codes on some channels (including the binary symmetric channel and the additive white Gaussian channel (AWGN)), we can obtain a new MLD algorithm which uses our extended Conti's algorithm.

The contribution of this study is that this is the first work which points out the relation between MLD and Gröbner bases. For arbitrary linear codes, the bounded distance decoding algorithm based on Gröbner bases is known [9, 12]. However, to the best of the author's knowledge, the idea of utilizing Gröbner bases in MLD is new. The decoding complexity of our proposed method is larger than other known MLD algorithm if we consider the AWGN channel. This is mainly because the computation of Gröbner bases consumes much time and space. If we can find an efficient algorithm for computing Gröbner bases, then the complexity of our method will be improved. Fortunately, for the binary symmetric channel, we can consider a different transformation of

MLD to integer programs. In this transformation, we can exclude the computation of Gröbner bases as a pre-computation. Hence each decoding operation is more efficient than the above AWGN case. Since the size of Gröbner bases grows exponentially with the length and dimension of the code, reducing the space complexity is an important work in future. We find an interesting class of codes which can be decoded with our algorithm based on Gröbner bases with less complexity [38].

In chapter 4, we introduce another construction of ideals appeared in the Conti-like MLD to reduce the complexity of the decoder [27]. As we mentioned, the Conti-like MLD algorithm which is discussed in chapter 3 needs to compute Gröbner bases of an ideal. We can compute Gröbner bases by Buchberger algorithm [1, 7]. However, unfortunately, the complexity of Buchberger algorithm is a strongly increasing function of the number of variables and the number of generators of ideals. Hence it would be useful to reduce the numbers for computing Gröbner bases. The main result of chapter 4 is to reduce both the number of the variables and generators of the ideal which is needed in the Conti-like MLD algorithm.

In chapter 5, we provide concluding remarks.

2. The ordered-statistics and the MLD techniques

2.1 Summary of this chapter

Fossorier and Lin developed a soft-decision decoding algorithm based on the *ordered-statistics (OS)* [16]. Their algorithm consists of two steps; hard-decision decoding and reprocessing. The hard-decision decoding is based on the ordered reliability values of the received symbols. Assume that an $[n, k]$ binary linear block code C is used over a memoryless channel. In the hard-decision decoding step, we first choose k symbols in the received sequence so that the symbols are as reliable as possible and the symbol positions can constitute information symbol positions of the code C. If the k quantized value do not involve errors, then the constructed codeword is the correct estimation of the transmitted codeword. However, if the quantized values involve errors, then the result of the decoding is not correct. To avoid this issue, the algorithm has the second step called reprocessing. There are $2^k = \sum_{i=0}^{k} \binom{k}{i}$ patterns of information symbols. For a given parameter l called *order*, at most $\sum_{i=0}^{l} \binom{k}{i}$ patterns are tested in the l-th order reprocessing process.

In [16], it is shown that the performance of the order-2 reprocessing is near the optimum MLD for some codes, including Reed-Muller (RM) codes. In this thesis, we will propose a sub-optimum MLD algorithm based on order-0 Fossorier's algorithm for arbitrary binary linear block codes. Our best simulation results (see figure 2.1 on page 16 and table 2.1 on 18) have been achieved for RM codes in practice. The simulation results show that the performance of the proposed decoding is near one of order-2 Fossorier's decoding, or one of MLD for low SNR's, while the complexity of the worst case of the proposed algorithm is smaller than one of order-2 Fossorier's decoding.

2.2 Notation

Suppose an $[n, k, d]$ binary linear block code C with a generator matrix G is used for error control over the additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) signaling. Also suppose each codeword is equally transmitted. Let $\mathbf{c} = (c_1, \ldots, c_n) \in C$ be a codeword. For BPSK transmission, the codeword $\mathbf{c}$ is mapped into the bipolar sequence $\mathbf{z} = (z_1, \ldots, z_n) \in \{-1, 1\}^n$ where $z_i = 2c_i - 1$ for $i = 1 \ldots, n$. After transmission, the received sequence at the output of the sampler in the demodulator is $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{R}^n$ with $r_i = z_i + e_i$ for $i = 1, \ldots, n$, where e_i's are statistically independent Gaussian random variables with zero mean and a fixed variance. See also [41] for details about the AWGN channel with BPSK signaling. We denote by $\mathbf{a} \circ \mathbf{b}$ the concatenation of two vectors $\mathbf{a}$ and $\mathbf{b}$.

2.3 The decoding algorithm based on ordered-statistics

Fossorier's Ordered statistics decoding (OSD) is a probabilistic information set decoding method. The original algorithm can be found in [16, 17]. The OSD method is a reduced-complexity soft decision decoding method for arbitrary binary linear block codes. For RM codes of lengths up to 64 bits, it is shown that the OSD achieve near optimum decoding by the computer simulation.

Their decoding algorithm consists of two major steps: the hard-decision step and the reprocessing step. In the hard-decision step, the decoding begins with reordering the components of the received sequence $\mathbf{r}$ by its absolute values. By this procedure, the decoder finds k information positions which is estimated by the hard-decision. If the hard-decision decoding has no error, then the decoder can obtain the transmitted word to encode the k symbols. Otherwise, the decoder needs to estimate the received sequence again for correcting errors. Fossorier's decoding has a threshold parameter l named *order* to design the performance and the complexity of the latter estimation. The performance and the complexity of the estimation are in a trade off relation. In the second step, which is called the l-th order reprocessing step, the decoder tests at most

$\sum_{i=1}^{l} \binom{l}{i}$ patterns of codewords for finding the most likely codeword in the patterns.

Fossorier and Lin have presented not only the decoding algorithm but also a theoretical evaluation of the algorithm. Both the performance and the complexity of OSD algorithm can be evaluated using the ordered statistics on the noise of the communication channel.

2.4 Proposed decoding algorithm

2.4.1 Overview of the algorithm

The overview of the proposed decoding algorithm is briefly introduced in this section. The details will be discussed in the following subsections. Our proposed algorithm consists of two major steps: (1) the reordering step and (2) the MLD step. Let t be a non-negative integer which is equal to or less than k. In the reordering step, t *most reliable independent* symbols are chosen from the received sequence, as in the ordered-statistic approach in [16]. The chosen symbols are quantized to binary values 0 or 1, and the binary symbols are regarded as the "correct symbols in the transmitted codeword". Remark that the set of codewords which have the chosen symbols at the chosen positions is a coset of linear punctured subcode of the original code C. Hence, in the MLD step, we can apply the soft-decision MLD algorithm over the set of the possible codewords to estimate the remaining symbols in the transmitted codeword. The details of each step, including the discussion of the error probability, are presented in the following.

2.4.2 Reordering step

Let $G = (\mathbf{g}_1, \ldots, \mathbf{g}_n)$ be a generator matrix of the code C where $\mathbf{g}_i$ with $1 \leq i \leq n$ is the column vector of G, and let $\mathbf{r} = (r_1, \ldots, r_n)$ be the received sequence. In this reordering step, t integers $\alpha_1, \ldots, \alpha_t$ are chosen so that

- column vectors $\mathbf{g}_{\alpha_1}, \ldots, \mathbf{g}_{\alpha_t}$ are linearly independent, and

- $|\mathbf{r}_{\alpha_i}|$ with $1 \leq i \leq t$ is as large as possible.

The computation of $\alpha_1, \ldots, \alpha_t$ is easily possible by using G and $\mathbf{r}$. (see section 2.6 and [16] for the detail). The symbols $\mathbf{r}_{\alpha_1}, \ldots, \mathbf{r}_{\alpha_t}$ are called the *t most reliable independent (MRI) symbols* of $\mathbf{r}$. Define a matrix G' as $G' = (\mathbf{g}_{\alpha_1}, \ldots, \mathbf{g}_{\alpha_t}, G^-)$ where G^- is the matrix obtained by deleting column vectors $\mathbf{g}_{\alpha_1}, \ldots, \mathbf{g}_{\alpha_t}$ from G. To make the following discussion easier, we perform elementary operations on row vectors of G', and consider that G' is of the form

$$G' = \begin{pmatrix} 1_t & * \\ 0_{k-t,t} & \bar{G} \end{pmatrix} \tag{2.1}$$

where 1_t is the $t \times t$ identity matrix, $0_{k-t,t}$ is the $(k-t) \times t$ zero matrix and $\bar{G}$ is a $(k-t) \times (n-t)$ matrix. Let C' be the code which is generated by G', and let $\mathbf{r}' = (r'_1, \ldots, r'_n)$ be the sequence which is obtained by reordering symbols in $\mathbf{r}$ as the same corresponding order as G'. Define

$$u_i := \begin{cases} 0 & \text{if } r'_i < 0, \\ 1 & \text{otherwise.} \end{cases}$$

for $1 \leq i \leq t$. That is, $u_1, \ldots, u_t$ are the hard-decision values of the first t MRI symbols. Let $C_{\mathbf{u}}$ be the set of vectors such that $C_{\mathbf{u}} = \{\mathbf{v} : \mathbf{u} \circ \mathbf{v} \in C'\}$, that is, $C_{\mathbf{u}}$ is the set of vectors which are "connectible" to $\mathbf{u} = (u_1, \ldots, u_t)$. In the following MLD step, we perform the MLD of $C_{\mathbf{u}}$ to find the most likely codeword, say $\mathbf{v}_m$, in $C_{\mathbf{u}}$. Since the communication channel is memoryless, $\mathbf{u} \circ \mathbf{v}_m$ is the most likely codeword among all codewords whose first t symbols are $u_1, \ldots, u_t$. Thus, if $\mathbf{u} = (u_1, \ldots, u_t)$ involves no error, then $\mathbf{u} \circ \mathbf{v}_m$ is the most likely codeword in C'.

In the rest of this section, we evaluate the probability that the vector $\mathbf{u}$ involves an error. This probability, denoted P_H in the following, affects the total error performance of the proposed algorithm. The precise analysis of P_H is complicated and beyond the scope of this paper, but it is possible in a similar way to the analysis in [16, Equation 61].

$$P_H \leq \sum_{j=0}^{n-t} P_{t,j}\Big(\sum_{i=0}^{t+j-1} \mathcal{P}\mathrm{e}(t+j-i;n)\Big) \tag{2.2}$$

where, for $1 \leq i \leq n$, $\mathcal{P}\mathrm{e}(i;n)$ is the probability that the hard-decision of r'_i is different from the transmitted symbol (i.e. involves an error), and $P_{t,i}$ is the probability that the (reordered) received sequence $\mathbf{r}'$ contains i symbols satisfying $|r'_j| > |r'_t|$ and $j > t$. The probability (2.2) will be used to evaluate the error performance of the proposed algorithm.

2.4.3 MLD step

In the MLD step, the algorithm compute the most likely codeword in $C_{\mathbf{u}}$ for the remaining part of the received sequence $\bar{\mathbf{r}}' = (r'_{t+1}, \ldots, r'_n)$. In general, any MLD algorithm for linear codes cannot be applied to $C_{\mathbf{u}}$ directly for estimation of $\bar{\mathbf{r}}'$ since $C_{\mathbf{u}}$ is not linear, but the following lemma leads that we can use an MLD algorithm for the estimation.

Lemma 2.1. *Let $\bar{C}$ be the code which is generated by $\bar{G}$, which is appeared in the equation 2.1, and let $\mathbf{y} = (y_t, \ldots, y_n)$ be the vector of length $n - t$ such that*

$$\mathbf{u} \circ \mathbf{y} = (\mathbf{u} \circ \mathbf{0}_{k-t})G',$$

where $\mathbf{0}_{k-t}$ is the zero vector whose length is $k - t$. Then $C_{\mathbf{u}} = \mathbf{y} + \bar{C}$.

Proof. From the definition of the MRI symbols and G', we have $|C_{\mathbf{u}}| = 2^{n-t}$. Since we have also $|\mathbf{y} + \hat{C}| = 2^{n-t}$, in order to prove the lemma, we need to see that if $\mathbf{v} \in C_{\mathbf{u}}$, then $\mathbf{v} - \mathbf{y} \in \hat{C}$. Let A be a $t \times (n - t)$ matrix such that

$$G' = \begin{pmatrix} 1_t & A \\ 0_{k-t,t} & \hat{G} \end{pmatrix}. \tag{2.1'}$$

Then we have $\mathbf{y} = \mathbf{u}A$ from the definition of $\mathbf{y}$. On the other hand, since $\mathbf{v} \in C_{\mathbf{u}}$, $\mathbf{v}$ must be written as

$$\mathbf{v} = \mathbf{u}A + \mathbf{w}\hat{G}$$

where $\mathbf{w}$ is a binary vector whose length is $k - t$. □

Lemma 2.1 shows that the MLD of $C_{\mathbf{u}}$ is possible by using an ML decoder of $\bar{C}$. Let $\hat{\mathbf{r}}' = (\hat{r}'_{t+1}, \ldots, \hat{r}'_n)$ where

$$\hat{r}'_i := \begin{cases} -r'_i & \text{if } y_i = 1 \\ r'_i & \text{otherwise} \end{cases}$$

and let $\bar{\mathbf{v}}_m$ be the most likely codeword in $\bar{C}$ for the sequence $\hat{\mathbf{r}}'$. It is clear that the most likely codeword in $C_{\mathbf{u}}$ for the sequence $\hat{\mathbf{r}}$ is gained as $\mathbf{v}_m = \mathbf{y} + \hat{\mathbf{v}}_m$.

If the vector $\mathbf{u}$ involves no error, then $\mathbf{u} \circ \mathbf{v}_m$ is the most likely codeword in C'. In this case, the probability that the MLD on $\bar{C}$ is erroneous equals the probability that this codeword is different from the transmitted codeword. By applying the union bound [41], the former probability P_S is bounded as

$$P_S \leq ((2^{k-t} - 1)/2) \exp(-\sigma(k-t)(d-t)/(2n)), \tag{2.3}$$

where σ is a constant which is determined by the variance of the noise of the AWGN channel (see also [41, p.407, Equation(5.2.67)]). This probability (2.3) will be recalled to evaluate the error performance of our proposed algorithm.

2.4.4 Proposed algorithm revised

We can summarize the previous discussion in the following decoding algorithm (Algorithm 2.1).

2.5 Evaluation of the algorithm

The error performance and the complexity of our proposed algorithm are evaluated in this section.

2.5.1 Error performance

First, we evaluate the error performance of our proposed decoding algorithm. Let P be the probability of the decoding error of the proposed algorithm. The algorithm

Algorithm 2.1 Proposed decoding algorithm using the ordered-statistics and the MLD technique

Input: The received sequence $\mathbf{r}$, a generator matrix G and a threshold parameter t.

Output: An estimated codeword.

1: Find the most reliable symbols of $\mathbf{r}$.
2: Reorder $\mathbf{r}$ and constitute the sequence $\mathbf{r}'$. Also reorder the column vectors of G, perform elementary row operations and constitute the generator matrix G' of the form in the equation (2.1).
3: Make the hard-decision of the first t symbols of $\mathbf{r}'$, and obtain the vector $\mathbf{u}$.
4: Determine the vector $\mathbf{y}$ satisfying $\mathbf{u} \circ \mathbf{y} = (u_1, \ldots, u_t, 0, \ldots, 0)G'$.
5: Compute $\hat{\mathbf{r}}'$ from $\mathbf{r}'$ and $\mathbf{y}$.
6: Compute the most likely codeword $\mathbf{v}_m \in C_{\mathbf{u}}$ for $\mathbf{r}'$ using the ML decoder of $\bar{C}$.
7: Reorder symbols in $\mathbf{u} \circ \mathbf{v}_m$ to obtain the codeword of the original code C and output the reordered vector.

makes a decoding error if the hard-decision involves errors, or the MLD procedure makes a decoding error though the hard-decision involves no error. Therefore we have

$$P \leq P_H + (1 - P_H)P_S,$$

where P_H and P_S are probability which is given in the equations (2.2) and (2.3). However, unfortunately, this inequality is too complex to compute and the bound is rather quite loose. Hence we evaluate the error performance by computer simulations in this thesis.

The figure 2.1 shows that the performance of the proposed algorithm depends on the choice of the threshold parameter t. We can see from the figure that the proposed algorithm with $t = 20$ or 30 achieves almost the same error performance as the Fossorier's algorithm. Actually, when the SNR is 1.0dB, the word error rate of order-2 Fossorier's algorithm is $10^{-0.428}$, while that of the proposed algorithm is $10^{-0.41}$ for $t = 30$, and $10^{-0.50}$ for $t = 20$.

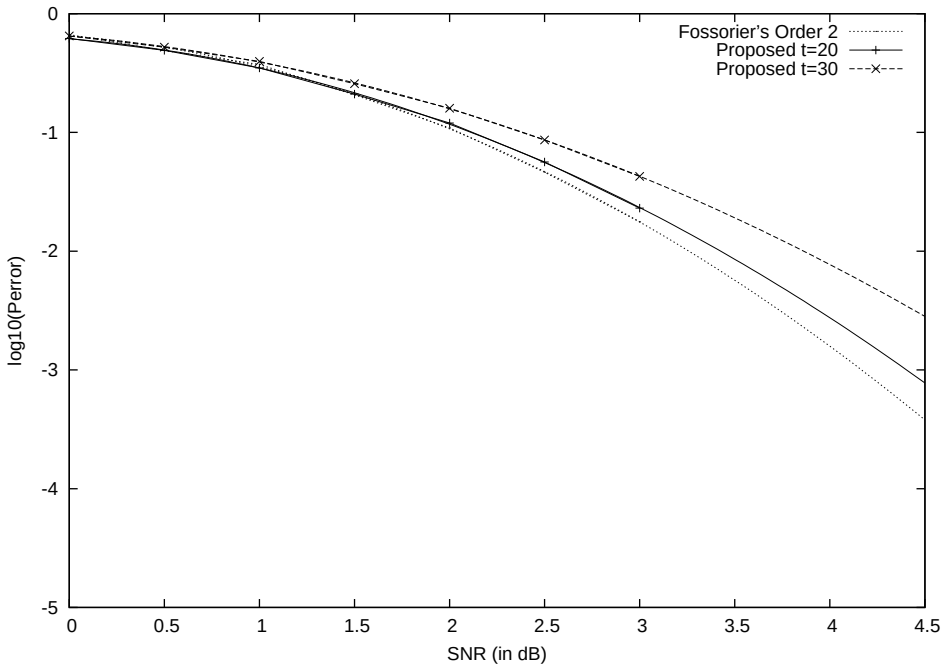

Figure 2.1 Performance of correcting errors for RM [64, 42] code

2.5.2 Complexity

Next, we discuss the complexity of the proposed algorithm. We use the number of additions and comparisons of the reliability information or entries of the received sequence as the measure of the complexity. For the first reordering step, we need at most $n \log_2 n$ operations to choose the MRI symbols. To obtain the hard-decision of t MRI symbols, we need t comparisons of reliability information. Therefore the complexity of the proposed algorithm is upper-bounded by

$$n \log_2 n + t + T_{\mathrm{MLD}},$$

where T_{MLD} is the complexity necessary in the MLD step. Note that T_{MLD} depends on the algorithm to be used for the subcode $\hat{C}$. In the following discussion, we consider to apply the MLD algorithm presented in [29].

By using computer simulation, we evaluated the average of the actual number of operations consumed in the proposed algorithm. Figure 2.2 shows the average number of operations necessary for decoding the [64, 42] RM code. The complexity decreases as the SNR increases because T_{MLD} decreases as the SNR increases.

To compare the proposed algorithm to Fossorier's algorithm, we also discuss the complexity in the worst case. As we saw in the previous section, the proposed algorithm achieves almost the same performance as order-2 Fossorier's algorithm, if we take $t = 20$ or 30. Table 2.1 shows the computer simulation results. In the worst case, the complexity of the proposed algorithm with $t = 30$ is slightly bigger than that of Fossorier's algorithm, while the error performance of the proposed algorithm with $t = 30$ is better than that of Fossorier's algorithm.

2.6 Implementation issue

For the implementation of the proposed algorithm, we need to calculate the t MRI symbols $\mathbf{r}_{\alpha_1}, \ldots, \mathbf{r}_{\alpha_t}$ of the received sequence $\mathbf{r} = (r_1, \ldots, r_n)$, as we have seen in section 2.4.2. We can find the t MRI symbols using the elementary row operations of

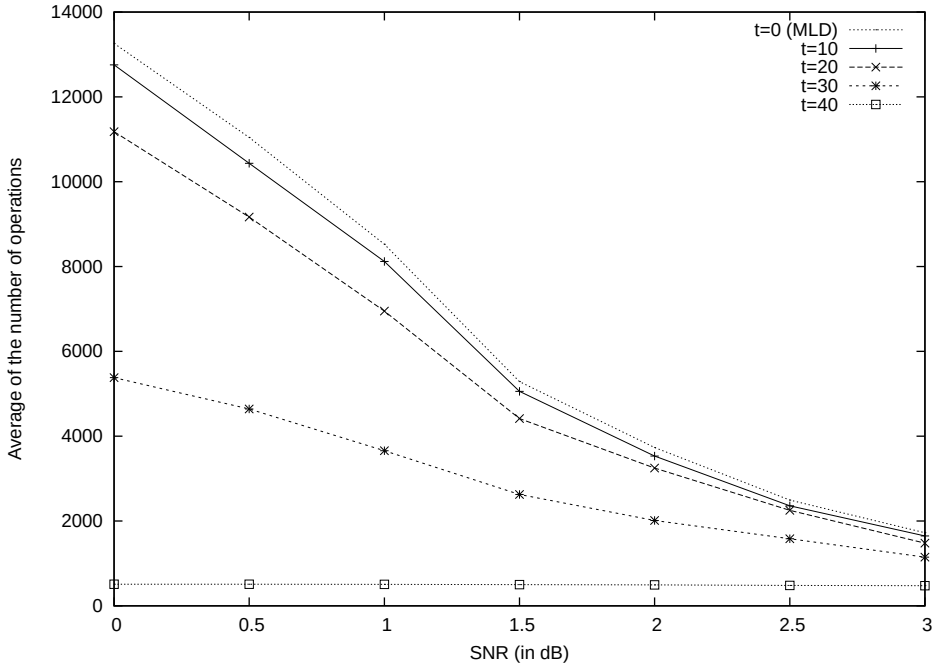

Figure 2.2 Average number of operations of the proposed algorithm for RM [64, 42] code with Kaji's MLD algorithm

Table 2.1 Complexity of the worst case for RM [64, 42] code on SNR=1.0dB

combining Kaji et al. 's [29] and OS	$t = 20$	19,665
	$t = 30$	39,033
Fossorier and Lin [16]	2nd order	20,208
Viterbi [16]		544,640

a generator matrix and the permutation of symbols. In this section, we introduce the method of finding *MRI* symbols in [16].

First, let $\lambda : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be a permutation map such that

$$|r_{\lambda(1)}| \geq |r_{\lambda(2)}| \geq \cdots \geq |r_{\lambda(n)}|.$$

Then we permute the columns of the generator matrix G based on λ. This results in the following matrix:

$$G_\lambda := (\mathbf{g}_{\lambda(1)}, \ldots, \mathbf{g}_{\lambda(n)})$$

Then starting from the most left column $\mathbf{g}_{\lambda(1)}$ of G_λ, we can find t independent columns to apply the elemental transform of rows of G_λ (or Gaussian elimination) with the largest associated reliability values $|r_{\lambda(i)}|$ for $i = 1, \ldots, n$. Therefore we can find the positions of t independent columns $\alpha_1, \ldots, \alpha_t$. The process of elemental transform is also useful to obtain the matrix

$$G' = \begin{pmatrix} 1_t & * \\ 0_{k-t,t} & \hat{G} \end{pmatrix}.$$

2.7 Conclusion of this chapter

In this chapter, we have considered a sub-optimum decoding algorithm. The algorithm uses the techniques of ordered-statistics decoding of Fossorier et al. to determine t information symbols where t is a parameter, and an MLD algorithm to determine the remaining symbols. The error performance and the decoding complexity can be controlled by choosing t. If we choose t smaller, then the error performance makes smaller, but the decoding complexity increases. Further, if we choose t so that the error performance of the proposed algorithm is almost the same as that of Fossorier's algorithm, then the decoding complexity of the proposed algorithm also becomes almost the same as that of Fossorier's algorithm.

Unfortunately, the error performance of the proposed algorithm, with the similar complexity of Fossorier's decoding, is little bit inferior to that of the order-2 Fossorier's

decoding. However, if an MLD algorithm whose complexity is less than Kaji's decoding [29] may appear, then the performance and complexity of our decoding should be improved to apply their MLD technique.

3. Decoding, integer programming and Gröbner bases

3.1 Summary of this chapter

The soft-decision maximum likelihood decoding can be considered as an integer program with binary arithmetics [34]. For binary linear block codes, the soft-decision maximum likelihood decoding is to find a codeword which maximizes inner products with the received sequence. It is equivalent to an integer programming which maximizes the inner products with the received sequence and a binary vector subject to the parity check equations with binary arithmetics.

In 1991, Conti and Traverso constructed an algorithm to solve integer programming using Gröbner bases of the toric ideal in polynomial rings over fields [5]. Their algorithm is interesting from a viewpoint in the theory of Gröbner bases and produces fruitful algebraic results [45].

In this chapter, we extend Conti's algorithm for integer programming with modulo arithmetic, especially with binary arithmetics for the soft-decision decoding [22, 24–26].

However, the proposed algorithm based on Gröbner bases is not efficient since the step to calculate Gröbner bases is computationally hard. There are several algorithms to calculate Gröbner bases, for example, Buchberger algorithm and improved Buchberger algorithms. In addition, since ideals in our algorithm has some algebraic properties such as binomial, zero-dimensional and radical, another algorithm to calculate Gröbner bases can be applied.

The aim of the Gröbner bases approach to the decoding is to find a class of codes which can be decoded efficiently. We consider a class of binary linear block codes of which parity check matrix equals the vertex-edge incident matrix of a finite graph. We explicitly show Gröbner bases in our proposed decoding algorithm for the class [38]. Therefore we can obtain Gröbner bases for the class of codes without using a time-consuming algorithm.

3.2 Preliminaries

3.2.1 MLD as integer programming

An *integer program* is a problem to find a nonnegative integral vector that minimizes (or maximizes) a linear target function subject to linear constraint equations with integral coefficients on the vector. Let $\mathbb{Z}$ and $\mathbb{R}$ be the set of integers and reals. For an $m \times n$ integral matrix $A \in \mathbb{Z}^{m \times n}$, an integral column vector $\mathbf{b} \in \mathbb{Z}^m$ of length m, and a real row vector $\mathbf{w} \in \mathbb{R}^n$, we write $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$ for the integer program to find $\mathbf{u} \in \mathbb{Z}^n_{\geq 0}$ that minimizes the inner product $\mathbf{w} \cdot \mathbf{u} \in \mathbb{R}$ subject to the linear constraint equations $A\mathbf{u} = \mathbf{b}$, where $\mathbb{Z}_{\geq 0}$ denotes the set of nonnegative integers. A solution $\mathbf{u} \in \mathbb{Z}^n_{\geq 0}$ which satisfies $A\mathbf{u} = \mathbf{b}$ is called *optimal* if $\mathbf{u}$ minimizes $\mathbf{w} \cdot \mathbf{u}$ in every solutions. Similarly, for an integer $q \geq 2$, we write $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ for the integer program to find $\mathbf{u} \in \mathbb{Z}^n_q$ that minimizes $\mathbf{w} \cdot \mathbf{u}$ subject to $A\mathbf{u} \equiv \mathbf{b} \mod q$, where in this case $A \in \mathbb{Z}^{m \times n}_q$, $\mathbf{b} \in \mathbb{Z}^m_q$ for $\mathbb{Z}_q = \{0, 1, \ldots, q-1\} \subset \mathbb{Z}_{\geq 0}$. Define the optimal solution for $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ in the same way. Conti-Traverso algorithm, which we will see later, solves integer programs represented as $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$. In this thesis, we will propose an extended Conti-Traverso algorithm to solve integer programs represented as $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$.

The MLD for binary linear block codes can be regarded as an integer program $\mathrm{IP}_{A,\mathbf{w},2}(\mathbf{b})$ as follows. Let C be a binary linear block code of length n and hence there exists a parity check matrix H of C with $C = \{\mathbf{u} \in \{0,1\}^n : H\mathbf{u} \equiv \mathbf{0} \mod 2\}$. A sender chooses a codeword $\mathbf{u} \in C$ uniformly and transmits $\mathbf{u}$ over a noisy memoryless channel. The vector $\mathbf{r}$ observed at the receiver's end is possibly different from $\mathbf{u}$ due to the noise on the channel. The maximum likelihood decoding (MLD) is to estimate the most-likely codeword $\mathbf{u}$ from $\mathbf{r}$. There are two different types of MLD: a hard-decision and a soft-decision. In the hard-decision MLD, the received vector $\mathbf{r}$ is quantized as a binary vector. The hard-decision MLD on the binary symmetric channel is to find an error vector $\mathbf{e} = (e_1, \ldots, e_n) \in \{0,1\}^n$ that has the smallest *Hamming weight* among all vectors in $\{0,1\}^n$ satisfying $H\mathbf{e} \equiv H\mathbf{r} \mod 2$, where the Hamming weight of $\mathbf{e}$ is defined to be $|\{i : e_i \neq 0, i = 1, \ldots, n\}| = e_1 + \cdots + e_n$. Note that the Hamming weight of $\mathbf{e}$ equals the inner product $\mathbf{1} \cdot \mathbf{e}$, where $\mathbf{1} = (1, \ldots, 1)$. Therefore the hard-decision

MLD is equivalent to solve the integer program $\mathrm{IP}_{H,\mathbf{1},2}(H\mathbf{r})$.

On the other hand, with appropriate transformation (see section 3.4.2), the soft-decision MLD can be regarded as a problem to find a binary vector $\mathbf{u} \in \{0,1\}^n$ that maximizes $\mathbf{r} \cdot \mathbf{u}$ subject to $H\mathbf{u} \equiv \mathbf{0} \mod 2$. Since to maximize $\mathbf{r} \cdot \mathbf{u}$ is equivalent to minimize $-\mathbf{r} \cdot \mathbf{u}$, the soft-decision MLD is to solve $\mathrm{IP}_{H,-\mathbf{r},2}(\mathbf{0})$. In the soft-decision MLD, $-\mathbf{r}$ may contain negative components. In the following discussion (in section 3.3 and 3.4.2), we will need a little attention for this issue.

3.2.2 Notation

Conti-Traverso algorithm uses the theory of Gröbner bases, hence we briefly review some notations in this section. Also refer to [1, 7, 8] for more detail.

Let F be a field and $F[X_1, \ldots, X_m]$ be the collection of all polynomials in m variables $X_1, \ldots, X_m$ with coefficients in F. For $f_1, \ldots, f_s \in F[X_1, \ldots, X_m]$, let $\langle f_1, \ldots, f_s \rangle$ be the collection

$$\langle f_1, \ldots, f_s \rangle = \{ \sum_{i=1}^{s} h_i f_i : h_i \in F[X_1, \ldots, X_m] \}.$$

Note that $\langle f_1, \ldots, f_s \rangle$ forms an ideal in $F[X_1, \ldots, X_m]$ and is called the *ideal generated by* $f_1, \ldots, f_s$. On the other hand, for an ideal I, if there exists $f_1, \ldots, f_s \in F[X_1, \ldots, X_m]$ such that $I = \langle f_1, \ldots, f_s \rangle$ then we say that $\{f_1, \ldots, f_s\}$ is a *basis* of I. Note that every ideal of $F[X_1, \ldots, X_m]$ has a finite basis.

A *monomial* in $F[X_1, \ldots, X_m]$ is a product $X_1^{v_1} \cdots X_m^{v_m}$ with $v_i \in \mathbb{Z}_{\geq 0}$ for $1 \leq i \leq m$. To abbreviate, we will sometimes write the above monomial as $\mathbf{X}^{\mathbf{v}}$, where $\mathbf{v} = (v_1, \ldots, v_m)$ is the vector of exponents in the monomial. A *binomial* in $F[X_1, \ldots, X_m]$ is a subtraction of two different monomials and written as $\mathbf{X}^{\mathbf{u}} - \mathbf{X}^{\mathbf{v}}$ with $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_{\geq 0}^m$ and $\mathbf{u} \neq \mathbf{v}$. An ideal in $F[X_1, \ldots, X_m]$ is called a *binomial ideal* if it is generated by only binomials. We will consider only binomial ideals through this part of thesis.

A *monomial order* $\prec$ on $F[X_1, \ldots, X_m]$ is a total order on the set of monomials in $F[X_1, \ldots, X_m]$ that satisfies following conditions: (i) if $\mathbf{X}^{\mathbf{u}_1} \prec \mathbf{X}^{\mathbf{u}_2}$, then $\mathbf{X}^{\mathbf{u}_1+\mathbf{v}} \prec \mathbf{X}^{\mathbf{u}_2+\mathbf{v}}$ for all $\mathbf{u}_1, \mathbf{u}_2, \mathbf{v} \in \mathbb{Z}_{\geq 0}^m$; (ii) $\mathbf{X}^{\mathbf{0}} = 1 \prec \mathbf{X}^{\mathbf{v}}$ for all $\mathbf{v} \in \mathbb{Z}_{\geq 0}^m$. For a monomial order $\prec$ and

a nonzero polynomial $f = \sum_{\mathbf{v}} c_{\mathbf{v}}\mathbf{X}^{\mathbf{v}} \in F[X_1, \ldots, X_m]$ with $c_{\mathbf{v}} \in F$ for any $\mathbf{v} \in \mathbb{Z}^m_{\geq 0}$, the *leading term of f with respect to $\prec$* is the term $c_{\mathbf{v}}\mathbf{X}^{\mathbf{v}}$ with $\mathbf{v}$ the largest in $\{\mathbf{X}^{\mathbf{v}} : c_{\mathbf{v}} \neq 0\}$ with respect to $\prec$. The leading term of f is denoted by $\mathrm{LT}_{\prec}(f)$.

For an ideal $I \subset F[X_1, \ldots, X_m]$, we denote by $\mathrm{LT}_{\prec}(I)$ the set of leading terms of elements of I and by $\langle \mathrm{LT}_{\prec}(I) \rangle$ the ideal generated by the elements of $\mathrm{LT}_{\prec}(I)$. A nonempty finite subset $\mathcal{G} = \{g_1, \ldots, g_t\} \subset F[X_1, \ldots, X_m]$ is said to be a *Gröbner basis* of an ideal I with respect to $\prec$ if and only if

$$\langle \mathrm{LT}_{\prec}(g_1), \ldots, \mathrm{LT}_{\prec}(g_t) \rangle = \langle \mathrm{LT}_{\prec}(I) \rangle.$$

If $\mathcal{G} = \{g_1, \ldots, g_t\}$ is a Gröbner basis of I, then $\mathcal{G}$ generates I. In particular, each g_i with $1 \leq i \leq t$ belongs to the ideal I. Conversely, for a monomial order $\prec$, every nonempty ideal in $F[X_1, \ldots, X_m]$ has a Gröbner basis with respect to $\prec$. By a monomial order $\prec$ and a Gröbner basis $\mathcal{G}$ with respect to $\prec$, the remainder of a polynomial $f \in F[X_1, \ldots, X_m]$ divided by every elements of $\mathcal{G}$ with respect to $\prec$ is uniquely determined according to the division algorithm in $F[X_1, \ldots, X_m]$. The unique remainder of f is called the *normal form* of f by $\mathcal{G}$ and denoted by $\overline{f}^{\mathcal{G}}$. For $f, h \in F[X_1, \ldots, X_m]$, it follows that $\overline{f}^{\mathcal{G}} = \overline{h}^{\mathcal{G}}$ if and only if $f - h \in I$. Especially, $\overline{f}^{\mathcal{G}} = 0$ if and only if $f \in I$. We write $f \equiv h \mod I$ if $\overline{f}^{\mathcal{G}} = \overline{h}^{\mathcal{G}}$.

3.2.3 Conti-Traverso algorithm

In [5], Conti and Traverso have proposed an algorithm to solve $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$. Their algorithm first defines an ideal $\hat{I}_A$, a monomial order $\prec_{\mathbf{w}}$ and a monomial $f_{\mathbf{b}}$ for the given input $A, \mathbf{w}$ and $\mathbf{b}$. The definition $\hat{I}_A, \prec_{\mathbf{w}}$ will be introduced in section 4.4.1 for discussion more technical terms. On the other hand, the definition of $f_{\mathbf{b}}$ is beyond the scope of this thesis and omitted here. The algorithm then computes Gröbner basis $\mathcal{G}_{A,\mathbf{w}}$ of $\hat{I}_A$ with respect to $\prec_{\mathbf{w}}$, and computes the normal form $\overline{f_{\mathbf{b}}}^{\mathcal{G}_{A,\mathbf{w}}}$. The normal form turns out to be a monomial and its exponent shows an optimal solution of $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$.

In the Conti-Traverso algorithm, the computation of $\mathcal{G}_{A,\mathbf{w}}$ consumes much complexity. Gröbner basis $\mathcal{G}_{A,\mathbf{w}}$ depends only on A and $\mathbf{w}$, and independent from $\mathbf{b}$. Thus if

Algorithm 3.1 Conti-Traverso algorithm

Input: $A \in \mathbb{Z}^{m\times n}$, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{w} \in \mathbb{R}^n$

Output: An optimal solution of $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$.

1: Compute a Gröbner basis $\mathcal{G}_{A,\mathbf{w}}$ of $\hat{I}_A$ with respect to $\prec_{\mathbf{w}}$.

2: Compute the normal form of the monomial $f_{\mathbf{b}}$ by $\mathcal{G}_{A,\mathbf{w}}$ with respect to $\prec_{\mathbf{w}}$.

3: Return the exponent vector of the normal form.

we solve $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$ for a given $A, \mathbf{w}$ and $\mathbf{b}$, then we can solve $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b}')$ for a different vector $\mathbf{b}'$ much more efficiently than the first time. At this point readers should recall the hard-decision MLD discussed in the previous section. The hard-decision MLD is equivalent to solve $\mathrm{IP}_{H,\mathbf{w},2}(H\mathbf{r}^t)$, where H and $\mathbf{w}$ do not change for the received vector. Therefore we can pre-compute Gröbner basis before we actually receive a vector from the channel, and we can use the same Gröbner basis in the following communication, which will make the hard-decision MLD much efficient; see section 3.4.1 for more in detail.

3.3 Extension of Conti-Traverso algorithm

We extend Conti-Traverso algorithm so that it can solve $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$. For this sake, we need to extend some lemmata presented in [5, 8] to the modulo q arithmetics. There are two key ideas for the extension: one is to add binomials $X_j^q - 1$'s into the ideal in Conti-Traverso algorithm, and another is to consider the Lawrence type matrix [45]. According to the former idea, Conti-Traverso algorithm is extended to solve an integer programming with q modulo arithmetics. On the other hand, the latter idea is used to extend Conti-Traverso algorithm to work for the target vectors contain negative values. To make discussion simpler, we first consider the case with $\mathbf{w} \in \mathbb{R}^n_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers. The case with arbitrary $\mathbf{w} \in \mathbb{R}^n$ is discussed later.

3.3.1 The case with $\mathbf{w} \in \mathbb{R}^n_{\geq 0}$

Let $A = (a_{i,j})$ be a full row rank $m \times n$ matrix and $a_{i,j} \in \mathbb{Z}_q$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$. Let $\mathbf{X}$ denote m variables $X_1, \ldots, X_m$, and $\mathbf{Y}$ denote n variables $Y_1, \ldots, Y_n$. In the context of MLD, A is the parity check matrix of the code, n equals to the length of the code and m equals to the number of parity check symbols. For a vector $\mathbf{u} \in \mathbb{Z}^n_{\geq 0}$, define $\theta(\mathbf{u}) = A\mathbf{u}^t$ and $\Theta : F[Y_1, \ldots, Y_n] \to F[X_1, \ldots, X_m]$, $\Theta(\mathbf{Y}^{\mathbf{u}}) = \mathbf{X}^{\theta(\mathbf{u})}$ and, for any $f = \sum_{\mathbf{v}} c_{\mathbf{v}} \mathbf{Y}^{\mathbf{v}} \in F[Y_1, \ldots, Y_n]$, $\Theta(f) = f(\Theta(Y_1), \ldots, \Theta(Y_n)) = \sum_{\mathbf{v}} c_{\mathbf{v}} \mathbf{X}^{\theta(\mathbf{v})}$. It follows immediately from the definition that $A\mathbf{u}^t = \mathbf{b}$ if and only if $\Theta(\mathbf{Y}^{\mathbf{u}}) = \mathbf{X}^{\mathbf{b}}$. Under the modulo q arithmetics, we have the following lemma. Let J be a binomial ideal defined by $J = \langle X_1^q - 1, \ldots, X_m^q - 1 \rangle \subset F[X_1, \ldots, X_m]$.

Lemma 3.1. *$A\mathbf{u}^t \equiv \mathbf{b} \mod q$ if and only if $\Theta(\mathbf{Y}^{\mathbf{u}}) \equiv \mathbf{X}^{\mathbf{b}} \mod J$.*

Proof. Let $\mathbf{b} = (b_1, \ldots, b_m)^t \equiv A\mathbf{u}^t \mod q$ with $\mathbf{b} \in \mathbb{Z}_q^m$ and hence $b_i \equiv \sum_{j=1}^n a_{i,j}u_j \mod q$ for $1 \leq i \leq m$. First we show that $X_i^{\sum_{j=1}^n a_{i,j}u_j} \equiv X_i^{b_i} \mod X_i^q - 1$. Since $b_i \equiv \sum_{j=1}^n a_{i,j}u_j \mod q$, there is a nonnegative integer p_i with $\sum_{j=1}^n a_{i,j}u_j = p_i q + b_i$. Hence

$$\begin{aligned} X_i^{\sum_{j=1}^n a_{i,j}u_j} - X_i^{b_i} &= X_i^{p_i q + b_i} - X_i^{b_i} \\ &= X_i^{b_i}(X_i^{p_i q} - 1) \\ &= X_i^{b_i}(X_i^{(p_i-1)q} + X_i^{(p_i-2)q} + \cdots + 1)(X_i^q - 1), \end{aligned}$$

and we have $X_i^{\sum_{j=1}^n a_{i,j}u_j} \equiv X_i^{b_i} \mod X_i^q - 1$. Now we have $X_i^{\sum_{j=1}^n a_{i,j}u_j} \equiv X_i^{b_i} \mod J$ since $X_i^q - 1$ is one of the generators of J, and

$$\Theta(\mathbf{Y}^{\mathbf{u}}) = \prod_{i=1}^m X_i^{\sum_{j=1}^n a_{i,j}u_j} \equiv \prod_{i=1}^m X_i^{b_i} \mod J. \tag{3.4}$$

To show the converse, assume that (3.4) holds. Since generators of J are relatively prime, it follows that $X_i^{\sum_{j=1}^n a_{i,j}u_j} \equiv X_i^{b_i} \mod X_i^q - 1$ for each i with $1 \leq i \leq m$. If $\sum_{j=1}^n a_{i,j}u_j \not\equiv b_i \mod q$, then $X_i^{\sum_{j=1}^n a_{i,j}u_j} \not\equiv X_i^{b_i} \mod X_i^q - 1$. Hence $\sum_{j=1}^n a_{i,j}u_j \equiv b_i \mod q$ for $1 \leq i \leq m$ and the lemma holds. □

Let $\phi_j = \Theta(Y_j) = \prod_{i=1}^{m} X_i^{a_{i,j}}$ for $1 \leq j \leq n$ and I_A be a binomial ideal defined by

$$I_A = \langle \phi_1 - Y_1, \ldots, \phi_n - Y_n, X_1^q - 1, \ldots, X_m^q - 1 \rangle$$
$$\subset F[X_1, \ldots, X_m, Y_1, \ldots, Y_n].$$

We say that a monomial order $\prec_e$ is an *elimination monomial order* if any monomial containing at least one of X_i with $1 \leq i \leq m$ is greater than any monomial containing only Y_j's. Hereafter, we fix an elimination monomial order $\prec_e$ and let $\mathcal{G}$ be a Gröbner basis for the ideal I_A with respect to $\prec_e$. Let $\psi = \mathbf{X}^{\mathbf{b}} = \prod_{i=1}^{m} X_i^{b_i}$, where $\mathbf{b} \equiv A\mathbf{u}^t \mod q$, and $\overline{\psi}^{\mathcal{G}}$ be the normal form of ψ by $\mathcal{G}$ with respect to $\prec_e$.

Lemma 3.2. *The normal form $\overline{\psi}^{\mathcal{G}}$ is a monomial containing only $Y_1, \ldots, Y_n$ and $\Theta(\overline{\psi}^{\mathcal{G}}) \equiv \mathbf{X}^{\mathbf{b}} \mod J$.*

For the proof of Lemma 3.2 we need the following Lemma 3.3 and Lemma 3.4.

Lemma 3.3. *For $f \in F[X_1, \ldots, X_m]$ the following conditions are equivalent:*

1. *There exists $h \in F[\phi_1, \ldots, \phi_m]$ such that $h \equiv f \mod J$.*
2. *$\overline{f}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$.*

In particular, if $\overline{f}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$ then $f \equiv \overline{f}^{\mathcal{G}}(\phi_1, \ldots, \phi_n) \mod J$.

Proof. This lemma is an extension of Proposition 1.8(a, b) in Chapter 8 of [8]. 1 $\Rightarrow$ 2: Let $h \in F[\phi_1, \ldots, \phi_n]$ such that $h \equiv f \mod J$. Since $J \subset I_A$, it follows that $h \equiv f \mod I_A$, and $\overline{h}^{\mathcal{G}} = \overline{f}^{\mathcal{G}}$. Therefore it suffices showing that $\overline{h}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$. To prove this, first note that in $F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$, a monomial $\phi_1^{v_1} \cdots \phi_n^{v_n}$ with $(v_1, \ldots, v_n) \in \mathbb{Z}_{\geq 0}^n$ can be written as follows:

$$\begin{aligned}\phi_1^{v_1} \cdots \phi_n^{v_n} &= (Y_1 + (\phi_1 - Y_1))^{v_1} \cdots (Y_n + (\phi_n - Y_n))^{v_n} \\ &= Y_1^{v_1} \cdots Y_n^{v_n} + B_1(\phi_1 - Y_1) + \cdots + B_n(\phi_n - Y_n)\end{aligned}$$

for some $B_1, \ldots, B_n \in F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$. Therefore we can write the polynomial h as

$$h = C_1(\phi_1 - Y_1) + \cdots + C_n(\phi_n - Y_n) + D$$

with $C_1, \ldots, C_n \in F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$, and $D \in F[Y_1, \ldots, Y_n]$. It follows that $\overline{h}^{\mathcal{G}} = \overline{D}^{\mathcal{G}}$ since each $C_j(\phi_j - Y_j) \in I_A$ for $1 \leq j \leq n$.

Let $\mathcal{G}' = \mathcal{G} \cap F[Y_1, \ldots, Y_n] = \{g_1, \ldots, g_s\}$. By representing D with $g_1, \ldots, g_s$ and its remainder, we can write D as follows:

$$D = E_1 g_1 + \cdots + E_s g_s + r$$

with $E_1, \ldots, E_s, r \in F[Y_1, \ldots, Y_n]$. Consequently we have $\overline{D}^{\mathcal{G}} = \overline{r}^{\mathcal{G}}$.

Now we claim that $r = \overline{r}^{\mathcal{G}}$, in other words, any leading term of elements of $\mathcal{G}$ with respect to $\prec_e$ cannot divide r. For a proof by contradiction, suppose that $\mathcal{G}$ contains a binomial g such that $\mathrm{LT}_{\prec_e}(g)$ is a term which contains only Y_j's and $\mathrm{LT}_{\prec_e}(g)$ divides r. This implies that the binomial g should be in $F[Y_1, \ldots, Y_n]$, because of the elimination property of $\prec_e$ and the fact that $\mathrm{LT}_{\prec_e}(g)$ is the greatest with respect to $\prec_e$. Thus $g \in \mathcal{G}'$ but this is a contradiction because any element of $\mathcal{G}'$ cannot divide r. Therefore we have shown that $\overline{f}^{\mathcal{G}} = \overline{h}^{\mathcal{G}} = \overline{D}^{\mathcal{G}} = \overline{r}^{\mathcal{G}} = r \in F[Y_1, \ldots, Y_n]$.

$2 \Rightarrow 1$: Let $\overline{f}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$ and $\mathcal{G} = \{g_1, \ldots, g_t\}$. By representing f with generators in $\mathcal{G}$ and its remainder, we can write

$$f = Z_1 g_1 + \cdots + Z_t g_t + \overline{f}^{\mathcal{G}} \tag{3.5}$$

with $Z_1, \ldots, Z_t \in F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$. To show variables in g_i explicitly, we say to write

$$g_i(X_1, \ldots, X_m, Y_1, \ldots, Y_n)$$

for g_i. For each $1 \leq j \leq n$, substitute ϕ_j for Y_j in the above formula (3.5). This substitution does not affect the left hand side of (3.5) since $f \in F[X_1, \ldots, X_m]$ does not contain Y_j's. Therefore

$$f = Z_1 g_1(X_1, \ldots, X_m, \phi_1, \ldots, \phi_n) + \cdots + Z_t g_t(X_1, \ldots, X_m, \phi_1, \ldots, \phi_n) + \overline{f}^{\mathcal{G}}.$$

Remark that if we substitute ϕ_j for Y_j, then the generator $\phi_j - Y_j$ in I_A reduces to 0 mod I_A. Hence if $g \in I_A$, then $g(X_1, \ldots, X_m, \phi_1, \ldots, \phi_n) \in J$ because of the definitions of J and I_A. This implies that

$$g_i(X_1, \ldots, X_m, \phi_1, \ldots, \phi_n) \in J,$$

and for $1 \le j \le n$,

$$Z_j(X_1, \ldots, X_m, \phi_1, \ldots, \phi_n) \cdot g_j(X_1, \ldots, X_n, \phi_1, \ldots, \phi_n) \equiv 0 \mod J.$$

Consequently we have $f \equiv \overline{f}^{\mathcal{G}}(\phi_1, \ldots, \phi_n) \mod J$. It is obvious that $\overline{f}^{\mathcal{G}}(\phi_1, \ldots, \phi_n) \in F[\phi_1, \ldots, \phi_n]$. □

Lemma 3.4. *Let $f \in F[\phi_1, \ldots, \phi_n]$ be a monomial in $X_1, \ldots, X_m$. Then $\overline{f}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$ is also a monomial.*

Proof of Lemma 3.4. This lemma is an extension of Proposition 1.8(c) in Chapter 8 of [8]. Lemma 3.3 implies $\overline{f}^{\mathcal{G}} \in F[Y_1, \ldots, Y_n]$. Since $\mathcal{G}$ is a Gröbner basis of the binomial ideal I_A, each polynomial in $\mathcal{G}$ is a binomial. Because the remainder of a monomial by a binomial is a monomial, $\overline{f}^{\mathcal{G}}$ must be a monomial. □

Proof of Lemma 3.2. This lemma is an extension of Proposition 1.6 in Chapter 8 of [8]. Because the matrix A is of full rank, there is a solution of the equations $A\mathbf{u}^t \equiv \mathbf{b} \mod q$. Therefore Lemma 3.1 implies that $\psi = \mathbf{X}^{\mathbf{b}} \equiv \Theta(\mathbf{Y}^{\mathbf{u}}) \mod J$. Note that $\Theta(\mathbf{Y}^{\mathbf{u}})$ is a monomial in $\Theta(Y_j) = \phi_j$ with $1 \le j \le n$ and $\psi \in F[\phi_1, \ldots, \phi_n]$. Lemma 3.4 implies that $\overline{\psi}^{\mathcal{G}}$ is a monomial containing only Y_j's, and Lemma 3.3 implies $\psi \equiv \overline{\psi}^{\mathcal{G}}(\phi_1, \ldots, \phi_n) \mod J$. Remark that

$$\begin{aligned}\Theta(\overline{\psi}^{\mathcal{G}}(Y_1, \ldots, Y_n)) &= \overline{\psi}^{\mathcal{G}}(\Theta(Y_1), \ldots, \Theta(Y_n)) \\ &= \overline{\psi}^{\mathcal{G}}(\phi_1, \ldots, \phi_n) \equiv \psi = \mathbf{X}^{\mathbf{b}} \mod J.\end{aligned}$$

□

Corollary 3.5. *If $\overline{\psi}^{\mathcal{G}} = Y_1^{u_1} \cdots Y_n^{u_n}$, then $\mathbf{u} \in \mathbf{Z}_q^n$ and $A\mathbf{u}^t \equiv \mathbf{b} \mod q$.*

Proof. For $j = 1, \ldots, n$, we have

$$\begin{aligned}Y_j^q - 1 &= Y_j^{q-1}(Y_j - \phi_j) + Y_j^{q-2}\phi_j(Y_j - \phi_j) + \cdots + \phi_j^{q-1}(Y_j - \phi_j) + \phi_j^q - 1 \\ &= \phi_j^q - 1 - (\phi_j - Y_j) \sum_{i=0}^{q-1} Y_j^i \phi_j^{q-1-i}.\end{aligned}$$

Remark that $\phi_j^q - 1 \in J$ since $\phi_j^q - 1 = (\prod_{i=1}^m X_i^{a_{i,j}})^q - 1$. Therefore we have $Y_j^q - 1 \in I_A$ and $\overline{Y_j^q - 1}^{\mathcal{G}} = 0$. First we show that if $\overline{\psi}^{\mathcal{G}} = Y_1^{u_1} \cdots Y_n^{u_n}$, then $u_j \in \mathbb{Z}_q$ for $1 \leq j \leq n$. For the proof by contradiction, assume that there is j with $u_j \geq q$. Let $u_j = pq + r$ where $p, q, r \in \mathbb{Z}_{\geq 0}$ with $r < q$. Define $f = Y_1^{u_1} \cdots Y_n^{u_n} - Y_j^r \prod_{i \neq j} Y_i^{u_i}$. Since $f = (Y_j^{pq} - 1) Y_j^r \prod_{i \neq j} Y_i^{u_i}$ and $\overline{Y_j^{pq} - 1}^{\mathcal{G}} = 0$, we have $\overline{f}^{\mathcal{G}} = 0$. On the other hand, $\mathrm{LT}_{\prec_e}(f) = Y_1^{u_1} \cdots Y_n^{u_n} = \overline{\psi}^{\mathcal{G}}$ since $r < u_j$. Note that $\mathrm{LT}_{\prec_e}(f) = \overline{\psi}^{\mathcal{G}}$ is already a remainder and cannot be divided by $\mathcal{G}$ with respect to $\prec_e$ and this contradicts to $\overline{f}^{\mathcal{G}} = 0$. Then we have $\mathbf{u} \in \mathbb{Z}_q^n$. The rest of the assertion follows from Lemma 3.2 and 3.1. In fact, Lemma 3.2 implies that $\Theta(\overline{\psi}^{\mathcal{G}}) = \Theta(Y_1^{u_1} \cdots Y_n^{u_n}) \equiv \mathbf{X}^{\mathbf{b}} \mod J$, which implies $A\mathbf{u}^t \equiv \mathbf{b} \mod q$ by Lemma 3.1. □

Corollary 3.5 suggests that we can find a solution to $A\mathbf{u}^t \equiv \mathbf{b} \mod q$ by computing the remainder of ψ by $\mathcal{G}$. A different choice of an elimination order $\prec_e$ results in different solution, while we would like to find an optimal solution $\mathbf{u}$ that minimizes $\mathbf{w} \cdot \mathbf{u}$. For this purpose, we need to choose the elimination order appropriately. The following definition is an extension of Definition 1.10 in Chapter 8 of [8].

Definition 1. A monomial order $\prec_{\mathbf{w}}$ on $F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$ is *adapted* to an integer programming $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ if it has the following two properties:

- (Elimination) The monomial order $\prec_{\mathbf{w}}$ is an elimination order.
- (Compatibility with $\mathbf{w}$) For any $\mathbf{u}, \mathbf{v} \in \mathbf{Z}_q^n$ with $\Theta(\mathbf{Y}^{\mathbf{u}}) \equiv \Theta(\mathbf{Y}^{\mathbf{v}}) \mod J$, if $\mathbf{w} \cdot \mathbf{u} < \mathbf{w} \cdot \mathbf{v}$, then $\mathbf{Y}^{\mathbf{u}} \prec_{\mathbf{w}} \mathbf{Y}^{\mathbf{v}}$.

Let $\prec_{\mathbf{w}}$ be a monomial order adapted to $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ and $\tilde{\mathcal{G}}$ be a Gröbner basis of the ideal I_A with respect to $\prec_{\mathbf{w}}$.

Theorem 3.6. *The monomial* $\overline{\psi}^{\tilde{\mathcal{G}}} \in F[Y_1, \ldots, Y_n]$ *will give an optimal solution of* $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$. *In other words, if* $\overline{\psi}^{\tilde{\mathcal{G}}} = Y_1^{u_1} Y_2^{u_2} \cdots Y_n^{u_n}$ *then* $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{Z}_q^n$ *minimizes* $\mathbf{w} \cdot \mathbf{u}$ *subject to* $A\mathbf{u}^t \equiv \mathbf{b} \mod q$.

This theorem leads an extended Conti-Traverso algorithm (Algorithm 3.2) to compute an optimal solution of the integer programming $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ when $\mathbf{w} \in \mathbb{R}_{\geq 0}^n$. For the proof of this theorem, we need another lemma.

Algorithm 3.2 Extended Conti-Traverso algorithm

Input: $A \in \mathbb{Z}_q^{m\times n}$, $\mathbf{b} \in \mathbb{Z}_q^m$, $\mathbf{w} \in \mathbb{R}_{\geq 0}^n$, $q \in \mathbb{Z}_{\geq 0}$ with $q \geq 2$.

Output: $\mathbf{u} \in \mathbb{Z}_q^n$ which is an optimal solution of the integer programming $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$.

1: Compute the Gröbner basis $\tilde{\mathcal{G}}$ of I_A with respect to a fixed adapted monomial order $\prec_{\mathbf{w}}$.
2: Compute the normal form $\overline{\psi}^{\tilde{\mathcal{G}}}$.
3: Return the exponent of $\overline{\psi}^{\tilde{\mathcal{G}}}$.

Lemma 3.7. *If $f \in F[Y_1, \ldots, Y_n]$ and $\Theta(f) \equiv 0 \mod J$ then $f \in I_A$.*

Proof. The proof is similar to the proof of $1 \Rightarrow 2$ in Lemma 3.3. A monomial $Y_1^{u_1} \cdots Y_n^{u_n}$ with $(u_1, \ldots, u_n) \in \mathbb{Z}_q^n$ can be written as

$$Y_1^{u_1} \cdots Y_n^{u_n} = \phi_1^{u_1} \cdots \phi_n^{u_n} + B_1(Y_1 - \phi_1) + \cdots + B_n(\phi_n - Y_n)$$

with $B_1, \ldots, B_n \in F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$. Then f can be written as

$$f(Y_1, \ldots, Y_n) = f(\phi_1, \ldots, \phi_n) + C_1(\phi_1 - Y_1) + \cdots + C_n(\phi_n - Y_n)$$

with $C_1, \ldots, C_n \in F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$. From the assumption, we have

$$\Theta(f) = f(\Theta(Y_1), \ldots, \Theta(Y_n)) = f(\phi_1, \ldots, \phi_n) \equiv 0 \mod J,$$

and $f \in I_A$. □

Proof of Theorem 3.6. Let $\mathbf{u} = (u_1, \ldots, u_n)$ such that $\overline{\psi}^{\tilde{\mathcal{G}}} = Y_1^{u_1} \cdots Y_n^{u_n}$. Corollary 3.5 implies that $\mathbf{u} \in \mathbb{Z}_q^n$. Assume that there is some $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$ such that $\mathbf{v} \neq \mathbf{u}$, $A\mathbf{v}^t \equiv \mathbf{b} \mod q$ and $\mathbf{w}\cdot\mathbf{v} < \mathbf{w}\cdot\mathbf{u}$. Define $f = Y_1^{u_1} \cdots Y_n^{u_n} - Y_1^{v_1} \cdots Y_n^{v_n}$, then from Lemma 3.2 we have

$$\begin{aligned}\Theta(f) &= \Theta(Y_1^{u_1} \cdots Y_n^{u_n}) - \Theta(Y_1^{v_1} \cdots Y_n^{v_n}) \\ &\equiv \mathbf{X}^{\mathbf{b}} - \mathbf{X}^{\mathbf{b}} \equiv 0 \mod J.\end{aligned}$$

Now by Lemma 3.7 it follows that $f \in I_A$ and $\overline{f}^{\mathcal{G}} = 0$. On the other hand $\mathrm{LT}_{\prec_{\mathbf{w}}}(f)$ must be $Y_1^{u_1} \cdots Y_n^{u_n}$ because $\prec_{\mathbf{w}}$ satisfies the compatibility with $\mathbf{w}$, though, the term $Y_1^{v_1} \cdots Y_n^{v_n}$ is already a remainder and cannot be divided by $\tilde{\mathcal{G}}$ with respect $\prec_{\mathbf{w}}$. This contradiction shows that $\mathbf{u}$ minimizes $\mathbf{w} \cdot \mathbf{u}$ subject to $A\mathbf{u}^t \equiv \mathbf{b} \mod q$. □

3.3.2 The case of an arbitrary $\mathbf{w} \in \mathbb{R}^n$

We cannot apply the discussion in the previous section if $\mathbf{w} = (w_1, \ldots, w_n)$ contains a negative value, because we cannot define an adapted monomial order $\prec_{\mathbf{w}}$. In this section, we consider to transform a given problem so that this issue does not occur.

For a given matrix $A \in \mathbb{Z}_q^{m \times n}$, consider an enlarged matrix

$$A' = \begin{pmatrix} A & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{pmatrix},$$

where $\mathbf{0}$ is the $m \times n$-zero matrix and $\mathbf{1}$ is the $n \times n$-identity matrix. The $(m+n) \times 2n$-matrix A' is called the *Lawrence Lifting* of A [45]. Let $\mathbf{c} = (q-1, \ldots, q-1) \in \mathbb{Z}_q^n$ whose components are all $q-1$ and set $\mathbf{b}' = (\mathbf{b}, \mathbf{c}) \in \mathbb{Z}_q^{m+n}$. Let $\mu = \max\{|w_i| : w_i < 0, i = 1, \ldots, n\} \cup \{0\} \in \mathbb{R}_{\geq 0}$ and set $\mathbf{w}_1 = (w_1 + \mu, \ldots, w_n + \mu) \in \mathbb{R}_{\geq 0}^n$, $\mathbf{w}_2 = (\mu, \mu, \ldots, \mu) \in \mathbb{R}_{\geq 0}^n$ and $\mathbf{w}' = (\mathbf{w}_1, \mathbf{w}_2) \in \mathbb{R}_{\geq 0}^{2n}$. Next theorem shows that $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ is equivalent to $\mathrm{IP}_{A',\mathbf{w}',q}(\mathbf{b}')$. Remark that even if $\mathbf{w}$ in $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ contains negative values, $\mathbf{w}'$ in $\mathrm{IP}_{A',\mathbf{w}',q}(\mathbf{b}')$ contains only *nonnegative* values and the algorithm in the previous section is applicable.

Theorem 3.8. *Let* $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{Z}_q^n$. *If* $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{Z}_q^{2n}$ *is an optimal solution of* $\mathrm{IP}_{A',\mathbf{w}',q}(\mathbf{b}')$ *then* $\mathbf{u}_1$ *is an optimal solution of* $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$.

Proof. If $\mathbf{u}$ is an optimal solution of $\mathrm{IP}_{A',\mathbf{w}',q}(\mathbf{b}')$, then $A'\mathbf{u} \equiv \mathbf{b}' \mod q$, in other words,

$$\begin{pmatrix} A & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} \equiv \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \mod q.$$

This linear constraint equations mean that $A\mathbf{u}_1^t \equiv \mathbf{b} \mod q$ and $\mathbf{u}_1 + \mathbf{u}_2 \equiv \mathbf{c} \mod q$. Let $\mathbf{u}_1 = (u_1, \ldots, u_n)$ and $\mathbf{u}_2 = (u_{n+1}, \ldots, u_{2n})$. Then $\mathbf{u}_1 + \mathbf{u}_2 \equiv \mathbf{c} \mod q$ implies that $u_i + u_{n+i} \equiv q - 1 \mod q$ for $i = 1, \ldots, n$. Since each $u_i \in \mathbb{Z}_q$ with $i = 1, \ldots, n$, these equations leads $u_i + u_{n+i} = q - 1$. Let $\mathbf{w}' = (\mathbf{w}_1, \mathbf{w}_2) = (w_1, \ldots, w_{2n}) \in \mathbb{R}_{\geq 0}^n$. Then we

have

$$\begin{aligned}\mathbf{w}' \cdot \mathbf{u} &= \sum_{i=1}^{2n} w_i u_i = \sum_{i=1}^{n} (w_i + \mu) u_i + \sum_{i=n+1}^{2n} \mu u_i \\ &= \sum_{i=1}^{n} w_i u_i + \sum_{i=1}^{2n} \mu u_i = \sum_{i=1}^{n} w_i u_i + \mu n(q-1) \\ &= \mathbf{w} \cdot \mathbf{u}_1 + \mu n(q-1).\end{aligned}$$

Since $\mu n(q-1)$ is a constant independent from $\mathbf{u}$, if $\mathbf{u}$ minimizes $\mathbf{w}' \cdot \mathbf{u}$, then $\mathbf{u}_1$ minimizes $\mathbf{w} \cdot \mathbf{u}_1$. □

3.4 MLD using Gröbner bases

In this section, we will present an application of the extended Conti-Traverso algorithm (Algorithm 3.2) in the previous section for decoding codes. We will show that either hard-decision or soft-decision MLD for binary linear block codes can be achieved by the proposed algorithm based on Gröbner bases. Through the section let $H \in \mathbb{Z}_2^{m \times n}$ be a parity check matrix of the code C. Note that $C = \{\mathbf{c} \in \mathbb{Z}_2^n : H\mathbf{c}^t \equiv 0 \mod 2\}$.

3.4.1 Hard-decision MLD

In this section, we assume that the communication channel is the binary symmetric channel. Let $\mathbf{c} \in C$ be the transmitted vector, $\hat{\mathbf{e}} \in \mathbb{Z}_2^n$ the error vector and $\mathbf{r} = (r_1, \ldots, r_n) \equiv \mathbf{c} + \hat{\mathbf{e}} \mod 2$ be the received vector. The hard-decision MLD is equivalent to the *syndrome decoding* [36]. In other words, the decoder algorithm chooses the candidate error vector $\mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{Z}_2^n$ which minimizes the Hamming weight $|\{i : e_i \neq 0, i = 1, \ldots, n\}|$ subject to $H\mathbf{e}^t \equiv H\mathbf{r}^t \mod 2$.

Let $\mathbf{w} = (1, 1, \ldots, 1)$ whose components are all 1. Then the Hamming weight of $\mathbf{e}$ equals the inner product $\mathbf{w} \cdot \mathbf{e}$. Therefore the hard-decision MLD for binary linear block codes is equivalent to $\mathrm{IP}_{H,\mathbf{w},2}(\mathbf{b})$ with $\mathbf{b} = H\mathbf{r}^t, \mathbf{b} \in \mathbb{Z}_2^m$, and can be solved using the extended Conti-Traverso algorithm (Algorithm 3.2).

Let $\tilde{\mathcal{G}}$ be a Gröbner basis of the ideal I_H with respect to an monomial order $\prec_{\mathbf{w}}$ adapted to $\mathrm{IP}_{H,\mathbf{w},2}(\mathbf{b})$.

Lemma 3.9. *Let $\tilde{\mathcal{G}}' = \tilde{\mathcal{G}} \cap F[Y_1, \ldots, Y_n]$. Then $\tilde{\mathcal{G}}'$ is a Gröbner basis of the ideal $I_H \cap F[Y_1, \ldots, Y_n]$ with respect to $\prec_{\mathbf{w}}$ and the normal form of the monomial $\mathbf{Y}^{\mathbf{r}} = Y_1^{r_1} \cdots Y_n^{r_n}$ by $\tilde{\mathcal{G}}'$ with respect to $\prec_{\mathbf{w}}$ is also a monomial.*

In addition, let $\mathbf{e}$ be the exponents vector of the normal form. Then $\mathbf{e}$ is an optimal solution of $\mathrm{IP}_{H,\mathbf{w},2}(\mathbf{b})$, in other words, $\mathbf{r} - \mathbf{e} \mod 2$ is the most likely codeword.

Proof of Lemma 3.9. The previous assertion is followed by the elimination theorem [7]. The rest of the proof is similar to Theorem 3.6 and omitted. □

Note that the decoder can compute Gröbner bases of I_H beforehand because Gröbner basis is independent of the received vector $\mathbf{r}$.

Example 1. Suppose a code $C = \{(0,0,0), (1,1,1)\}$ and a parity check matrix $H = \left(\begin{smallmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{smallmatrix}\right)$. Then a Gröbner basis $\tilde{\mathcal{G}}$ of the ideal I_H with respect to $\prec_{\mathbf{w}}$ can be calculated using the Buchberger algorithm. We have that $\tilde{\mathcal{G}}' = \tilde{\mathcal{G}} \cap F[Y_1, \ldots, Y_n]$ is $\{Y_1Y_2 - Y_3, Y_1Y_3 - Y_2, Y_2Y_3 - Y_1, Y_1^2 - 1, Y_2^2 - 1, Y_3^2 - 1\}$. Let $\mathbf{r} = (1,0,1)$ be the received vector. The decoder can calculate the normal form of $\mathbf{Y}^{\mathbf{r}} = Y_1Y_3$ by the $\tilde{\mathcal{G}}'$. We find that $\overline{Y_1Y_3}^{\tilde{\mathcal{G}}'} = Y_2$ as expected, which suggests the error pattern $(0,1,0)$ and the most likely codeword $(1,1,1) = (1,0,1) - (0,1,0) \mod 2$.

Algorithm 3.3 Hard-decision MLD using Gröbner bases

Input: $\mathbf{r} = \mathbf{c} + \hat{\mathbf{e}} \mod 2$ and $\tilde{\mathcal{G}}' = \tilde{\mathcal{G}} \cap F[Y_1, \ldots, Y_n]$, where $\tilde{\mathcal{G}}$ is a Gröbner basis of I_H with respect to an adapted monomial order.

Output: The most likely codeword $\mathbf{u} \in C$.

1: Set $f = Y_1^{r_1} Y_2^{r_2} \cdots Y_n^{r_n}$.
2: Compute the normal form $\overline{f}^{\tilde{\mathcal{G}}'}$.
3: Let $\mathbf{e}$ be the exponents vector of the monomial $\overline{f}^{\tilde{\mathcal{G}}'}$.
4: Return $\mathbf{u} = \mathbf{r} - \mathbf{e} \mod 2$.

3.4.2 Soft-decision MLD

In this section, we present another application of the extended Conti-Traverso algorithm (Algorithm 3.2) for a soft-decision decoding. Suppose that the channel is additive white Gaussian noise (AWGN) channel with binary phase-shift keying (BPSK) signaling . In other words, if the codeword $\mathbf{c} = (c_1, \ldots, c_n) \in C$ is transmitted, the modulator maps $\mathbf{c}$ into a bipolar sequence represented by $(l_1, \ldots, l_n) \in \{-1, 1\}^n$ with $l_i = 2c_i - 1$ for $i = 1, \ldots, n$. The received sequence $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{R}^n$ can be represented by $r_i = l_i + z_i$ for $i = 1, \ldots, n$, where each z_i is a statistically independent real valued Gaussian random variable with 0 mean and a fixed variance.

For soft-decision MLD, the decoder finds the candidate codeword $\mathbf{u}$ which maximizes $\mathbf{r} \cdot \mathbf{u}$ subject to $H\mathbf{u}^t \equiv \mathbf{0} \mod 2$ (see [34]). In other words, the soft-decision MLD for binary linear block codes is equivalent to $\mathrm{IP}_{H,-\mathbf{r},2}(\mathbf{0})$. By Theorem 3.8, the integer programming $\mathrm{IP}_{H,-\mathbf{r},2}(\mathbf{0})$ is equivalent to $\mathrm{IP}_{A,\mathbf{w},2}(\mathbf{b})$, where $A \in \mathbb{Z}_2^{(m+n)\times 2n}$ is the Lawrence lifting of H, $\mathbf{b} = (0, 0, \ldots, 0, 1, 1, \ldots, 1) \in \mathbb{Z}_2^{m+n}$, $\mathbf{w} = (\mu - r_1, \ldots, \mu - r_n, \mu, \mu, \ldots, \mu) \in \mathbb{R}_{\geq 0}^{2n}$ and $\mu = \max\{r_i : r_i > 0\} \cup \{0\}$ which can be solved by the extended Conti-Traverso algorithm using the Gröbner basis of I_A.

Example 2. Suppose a code

$$C = \{(0, 0, 0), (1, 1, 0), (0, 1, 1), (1, 0, 1)\}$$

and a parity check matrix $H = (1, 1, 1)$. If the received sequence is $\mathbf{r} = (-2, -3, 9)$, then the most likely codeword is $(1, 0, 1)$. In this case, $m = 1$ and $n = 3$. Now $\mu = 9$ and $\mathbf{w} = (11, 12, 0, 9, 9, 9)$. We can compute the Gröbner basis $\tilde{\mathcal{G}}$ of I_A with respect to $\prec_{\mathbf{w}}$ using the Buchberger algorithm. Then we find that $\tilde{\mathcal{G}}$ contains 18 binomials and the normal form of $\mathbf{X}^{\mathbf{b}} = X_2X_3X_4$ by $\tilde{\mathcal{G}}$ turns out to be $Y_1Y_3Y_5$ as expected, giving the optimal solution $(1, 0, 1, 0, 1, 0)$ and the most likely codewords $(1, 0, 1)$ appeared at the left half 3 components.

Let $\tilde{\mathcal{G}}$ be a Gröbner basis of the ideal I_A with respect to an adapted monomial order $\prec_{\mathbf{w}}$. As same as Lemma 3.9, the following lemma (Lemma 3.10) can be proved. Lemma 3.10 leads a variant soft-decision algorithm of Algorithm 3.4.

Algorithm 3.4 Soft-decision MLD using Gröbner bases

Input: The parity check matrix $H \in \mathbb{Z}_2^{m\times n}$ and the received sequence $\mathbf{r} \in \mathbb{R}^n$.

Output: The most likely codeword $\mathbf{u} \in C$.

1: Find $\mu = \max\{r_i : r_i > 0\} \cup \{0\}$.

2: Set $\mathbf{w} = (\mu - r_1, \ldots, \mu - r_n, \mu, \mu, \ldots, \mu) \in \mathbb{R}^{2n}_{\geq 0}$.

3: Let $A = \left(\begin{smallmatrix} H & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{smallmatrix}\right)$ be the Lawrence matrix of H.

4: Compute a Gröbner basis $\tilde{\mathcal{G}}$ of I_A with respect to an adapted monomial order $\prec_{\mathbf{w}}$.

5: Set $f = X_{m+1} \cdots X_{m+n}$.

6: Compute the normal form $\overline{f}^{\tilde{\mathcal{G}}}$.

7: Return $\mathbf{u} = (u_1, \ldots, u_n)$ with

$$\overline{f}^{\tilde{\mathcal{G}}} = Y_1^{u_1} \cdots Y_n^{u_n} Y_{n+1}^{u_{n+1}} \cdots Y_{2n}^{u_{2n}}.$$

Algorithm 3.5 Modified soft-decision MLD using Gröbner bases

Input: The parity check matrix $H \in \mathbb{Z}_2^{m\times n}$ and the received sequence $\mathbf{r} \in \mathbb{R}^n$.

Output: The most likely codeword $\mathbf{u} \in C$.

1: Find $\mu = \max\{r_i : r_i > 0\} \cup \{0\}$.

2: Set $\mathbf{w} = (\mu - r_1, \ldots, \mu - r_n, \mu, \mu, \ldots, \mu) \in \mathbb{R}^{2n}_{\geq 0}$.

3: Let $A = \left(\begin{smallmatrix} H & \mathbf{0} \\ \mathbf{1} & \mathbf{1} \end{smallmatrix}\right)$ be the Lawrence matrix of H.

4: Compute a Gröbner basis $\tilde{\mathcal{G}}'$ of $I_A \cap F[Y_1, \ldots, Y_n]$ with respect to an adapted monomial order $\prec_{\mathbf{w}}$.

5: Choose an *initial* codeword $\mathbf{c} = (c_1, \ldots, c_n) \in C$, and let $\bar{c}_i = 1 - c_i \mod 2$ for $i = 1, \ldots, n$.

6: Set $f = Y_1^{c_1} \cdots Y_n^{c_n} Y_{n+1}^{\bar{c}_1} \cdots Y_{2n}^{\bar{c}_n}$.

7: Compute the normal form $\overline{f}^{\tilde{\mathcal{G}}}$.

8: Return $\mathbf{u} = (u_1, \ldots, u_n)$ with

$$\overline{f}^{\tilde{\mathcal{G}}} = Y_1^{u_1} \cdots Y_n^{u_n} Y_{n+1}^{u_{n+1}} \cdots Y_{2n}^{u_{2n}}.$$

Lemma 3.10. *Let* $\tilde{\mathcal{G}}' = \tilde{\mathcal{G}} \cap F[Y_1, \ldots, Y_{2n}]$. *Then* $\tilde{\mathcal{G}}'$ *is a Gröbner basis of the ideal* $I_A \cap F[Y_1, \ldots, Y_{2n}]$ *with respect to* $\prec_{\mathbf{w}}$. *Choose* $\mathbf{c} = (c_1, \ldots, c_n) \in C$ *and let* $\bar{c}_i = 1 - c_i \mod 2$ *for* $i = 1, \ldots, n$. *Let* $f = Y_1^{c_1} \cdots Y_n^{c_n} Y_{n+1}^{\bar{c}_1} \cdots Y_{2n}^{\bar{c}_n}$ *and the normal form* $\overline{f}^{\tilde{\mathcal{G}}'} = Y_1^{u_1} \cdots Y_n^{u_n} Y_{n+1}^{u_{n+1}} \cdots Y_{2n}^{u_{2n}}$. *Then the left hand-side vector* $(u_1, \ldots, u_n)$ *is the most likely codeword.*

Proof. Since the assertion is followed by Lemma 3.9 straightforwardly, the proof is omitted. □

Example 3 (Continued). The reduced Gröbner basis $\tilde{\mathcal{G}}' = \tilde{\mathcal{G}} \cap F[Y_1, \ldots, Y_{2n}]$ in Example 2 consists of 14 binomials. Let the initial codeword $\mathbf{c} = (c_1, c_2, c_3) = (1, 1, 0)$ at Step 5 in Algorithm 3.5. Then $\bar{\mathbf{c}} = (\bar{c}_1, \bar{c}_2, \bar{c}_3) = (0, 0, 1)$ and we can compute the normal form of $Y_1^{c_1} Y_2^{c_2} Y_3^{c_3} Y_4^{\bar{c}_1} Y_5^{\bar{c}_2} Y_6^{\bar{c}_3} = Y_1 Y_2 Y_6$ by $\tilde{\mathcal{G}}'$ which turns out to be $Y_1 Y_3 Y_5$ as expected. If we let another codeword for the initial codeword at Step 5, for example, $(0, 1, 1) \in C$, we find that the normal form of the monomial $Y_2 Y_3 Y_4$ related $(0, 1, 1)$ is also $Y_1 Y_3 Y_5$.

3.5 Discussion on complexity

In this section, we discuss the complexity of the proposed algorithms. On the hard-decision MLD algorithm (Algorithm 3.3), the complexity depends on the calculation of the normal form at step 2. In the case of the modified soft-decision MLD algorithm (Algorithm 3.5), the complexity depends on not only the calculation of the normal form but also the calculation of the Gröbner basis at step 4. Unfortunately, it is difficult to estimate the complexity necessary for these operations even though it guarantees that the algorithms terminate definitely. Experimentally, the computation of a Gröbner basis using the basic Buchberger algorithm takes tremendously long time and consumes a huge amount of storage space for codes with practical length and dimension. In section 4, we will discuss the complexity of computing Gröbner bases in detail.

In the following, for the evaluation of the space-complexity of the proposed algorithms, we try to estimate the number of the polynomials in the Gröbner basis in the

algorithms. Note that the Gröbner basis which we consider in the proposed algorithms consists of only binomials. Let $\tilde{\mathcal{G}}'$ be the Gröbner basis of the ideal $I_H \cap F[Y_1, \ldots, Y_n]$ with respect to the adapted monomial order in the hard-decision decoding algorithm. Since $Y_j^2 - 1 \in I_H$ for each $j = 1, \ldots, n$, if $\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} \in \tilde{\mathcal{G}}'$ with $\mathbf{u} \neq \mathbf{v} \neq \mathbf{0}$, then both $\mathbf{u}$ and $\mathbf{v}$ are in $\mathbf{Z}_2^n$ when $\tilde{\mathcal{G}}'$ is reduced. Therefore the number of the binomials in the reduced Gröbner basis $\tilde{\mathcal{G}}'$ is less than $2^{2n-1} + n$. This estimation may be loose in practice, in fact, $|\tilde{\mathcal{G}}|' = 6$ in Example 1, while the bound is $2^{2n-1} + n = 35$. As the same reason, the number of the binomials in the Gröbner basis in the modified soft-decision algorithm (Algorithm 3.5) is less than $2^{4n-1} + 2n$. The number of the binomials in the Gröbner basis for a special class of codes can be estimated tightly [38].

3.6 Conclusion of this chapter

In this chapter, an extended Conti-Traverso algorithm to solve integer programs with modulo arithmetics have been proposed. Furthermore, we have proposed the hard-decision and soft-decision MLD algorithms for binary linear block codes based on the extended Conti-Traverso algorithm using Gröbner bases. The proposed MLD algorithms are not as efficient as the known algorithms because of the complexity to calculate Gröbner bases. Future study should include an investigation of algorithms for computing Gröbner bases which are more efficient than the Buchberger algorithm or the changing ordering algorithm [11].

4. Generators of ideals in the Conti-like MLD algorithm

4.1 Finding generators of ideals in the Conti-like MLD

In chapter 3, we proposed the Conti-like MLD algorithm based on Gröbner bases of a certain ideal in a polynomial ring. In this chapter, we are interested in computing a Gröbner basis for an ideal which is appeared in the Conti-like MLD. To compute Gröbner bases can be accomplished applying Buchberger algorithm [1,7], however unfortunately, the complexity of Buchberger algorithm is a strongly increasing function of the number of variables and one of generators of ideals. Hence it would be useful to reduce both the numbers for calculating Gröbner bases. The main result of this chapter is to obtain another generators of the ideal which is appeared in the Conti-like MLD for reducing both the numbers of generators and one of the variables.

The Conti-like MLD algorithm consists in the first step of computing a Gröbner basis of the ideal which is determined in section 3. The second step then involves a reduction of a monomial by the Gröbner basis. The latter step is a subject of investigation of its own, and can be solved efficiently to use a *reduced Gröbner basis* (see [1,7] for the definition), so we concentrate on the first step only to speed up.

The conception in section 4 is derived from the study of Conti-Traverso algorithm by Hosten and Sturmfels [21], and by Di Biase and Urbanke [10]. The basic problem in Conti-Traverso algorithm is the complexity of computing Gröbner bases, as similar as the Conti-like MLD algorithm. Their concept to conquest of this problem is to consider a certain ideal which is defined from the integral kernel of the coefficient matrix of the integer program, and then to compute the generators of an ideal which is defined by the kernel. The difference between the Hosten-Sturmfels method and the Di Biase-Urbanke method lies at the latter step. On the Other hand, their methods are in common at the former step.

In the case of Conti-like MLD, the coefficient matrix is the parity check matrix as we have discussed in chapter 3, and its kernel over binary field forms a generator matrix. We consider an ideal which is defined by a generator matrix of the code as similar as

Hosten-Sturmfels or Biase-Urbanke, and show that the complexity of the computation of Gröbner bases in the Conti-like MLD can be improving theoretically [27].

4.2 Preliminaries

Let us fix every notation for the whole of section 3.

> *F is a field. $F[X_1, \ldots, X_m]$ is a polynomial ring in m variables $X_1, \ldots, X_m$ with coefficients in F.* $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$ *is an integer program with a coefficient matrix A, a target vector $\mathbf{w}$ and a right-hand vector $\mathbf{b}$.* $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ *is an integer program with a coefficient matrix A, a target vector $\mathbf{w}$ and a right-hand vector $\mathbf{b}$ using modulo q arithmetics.*

For $f \in F[X_1, \ldots, X_m]$ and an ideal $I \subset F[X_1, \ldots, X_m]$, the *ideal quotient* $(I : f^\infty)$ is the set

$$\{g \in F[X_1, \ldots, X_m] : f^r g \in I \text{ for some } r \in \mathbb{Z}_{\geq 0}\}.$$

For $\mathbf{u} = (u_1, \ldots, u_m) \in \mathbb{Z}^m_{\geq 0}$, a *support of* $\mathbf{u}$ is the set $\{i : u_i \neq 0\}$. Every $\mathbf{u} \in \mathbb{Z}^m$ can be written uniquely as $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ where $\mathbf{u}^+$ and $\mathbf{u}^-$ are non-negative and have disjoint supports. A *graded reverse lexicographic order* $\prec_{\mathrm{grevlex}}$ is defined as follows:

$$\mathbf{X}^{\mathbf{u}} \prec_{\mathrm{grevlex}} \mathbf{X}^{\mathbf{v}} \Leftrightarrow |\mathbf{u}| = \sum_{i=1}^{m} u_i < |\mathbf{v}| = \sum_{i=1}^{m} v_i, \text{ or}$$

$|\mathbf{u}| = |\mathbf{v}|$ and, in $\mathbf{v} - \mathbf{u}$, the right-most nonozero entry is negative. A *weighted order* $\prec_{\mathbf{w}}$ with $\mathbf{w} \in \mathbb{R}^m_{\geq 0}$ is defined as follows:

$$\mathbf{X}^{\mathbf{u}} \prec_{\mathbf{w}} \mathbf{X}^{\mathbf{v}} \Leftrightarrow \mathbf{w} \cdot \mathbf{u} = \sum_{i=1}^{m} w_i u_i < \mathbf{w} \cdot \mathbf{v} = \sum_{i=1}^{m} w_i v_i, \text{ or}$$

$\mathbf{w} \cdot \mathbf{u} = \mathbf{w} \cdot \mathbf{v}$ and $\mathbf{u} \prec_{\mathrm{grevlex}} \mathbf{v}$. Both $\prec_{\mathrm{grevlex}}$ and $\prec_{\mathbf{w}}$ with $\mathbf{w} \in \mathbb{R}^m_{\geq 0}$ form monomial orders.

4.3 Integer programming and Gröbner bases

In this section, we will see Conti-Traverso algorithm [5] and the Conti-like MLD algorithm [26] for discussion of their difference.

4.3.1 Conti-Traverso algorithm

Conti and Traverso have proposed an algorithm to solve $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$, as we introduced in section 3.2.3. Their algorithm uses the theory of Gröbner bases, hence we briefly review some basic notation in this section. Refer [1, 7, 8] for more detail.

Conti-Traverso algorithm first defines an ideal $\hat{I}_A$, a monomial $f_{\mathbf{b}}$ or the given input A and $\mathbf{b}$. We will see the definition of $\hat{I}_A$ soon in section 4.4.1. The algorithm then computes a Gröbner basis $\mathcal{G}$ of $\hat{I}_A$ with respect to the weighted order $\prec_{\mathbf{w}}$ with the target vector $\mathbf{w}$. After the computation, the algorithm computes the normal form of $f_{\mathbf{b}}$ by the basis $\mathcal{G}$.

The computation of Gröbner bases can be accomplished by Buchberger algorithm. However, unfortunately, it takes much time and space to compute $\mathcal{G}$ by Buchberger algorithm. There are two improved methods for computing Gröbner bases for Conti-Traverso algorithm: one is proposed by Hosten and Sturmfels [21], and another is proposed by Di Biase and Urbanke [10]. In this thesis, we refer [45] for their methods since the key ideas of them are summarized in [45].

Conti-Traverso algorithm have been further developed in [40,46,47,50,51] with the development of the theory of toric ideals. The ideals $\hat{I}_A$ and $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$ are called *toric ideals* [47]. Algorithm 4.1 is a modified Conti-Traverso algorithm with an ideal which consists of fewer variables and generators than one of the original. Note that the original Conti-Traverso algorithm and the modified algorithm can be extended in the case where $\mathbf{w}$ has negative values, however the details are beyond of this thesis and omitted.

Algorithm 4.1 Modified Conti-Traverso algorithm

Input: $A \in \mathbb{Z}^{m\times n}$, $\mathbf{b} \in \mathbb{Z}^m$, $\mathbf{w} \in \mathbb{R}^n_{\geq 0}$

Output: An optimal solution of $\mathrm{IP}_{A,\mathbf{w}}(\mathbf{b})$.

1: Compute a Gröbner basis $\mathcal{G}_{A,\mathbf{w}}$ of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$ with respect to the weighted order $\prec_{\mathbf{w}}$, which is determined by $\mathbf{w}$.

2: Compute the normal form of the monomial $f_{\mathbf{b}}$ by $\mathcal{G}_{A,\mathbf{w}}$ with respect to $\prec_{\mathbf{w}}$.

3: Return the exponent vector of the normal form.

4.3.2 MLD Using Gröbner bases

In section 3.3, we proposed a maximum likelihood decoding algorithm which is based on an extended Conti-Traverso algorithm. The proposed decoding algorithm is called the *Conti-like MLD* algorithm in this section.

For the soft-decision MLD (Algorithm 4.2), at step 2, we need a Gröbner basis of $I_{\Lambda(H)} \cap F[Y_1, \ldots, Y_n]$, where $\Lambda(H)$ is the Lawrence lifting of the parity check matrix H. We will reviewed the definition of I_A for an integeral matrix A in section 4.4.2.

On the other hand, for the hard-decision MLD (Algorithm 4.3), we does not need to compute Gröbner bases since it should be given as input of the algorithm. However, we also consider the complexity of computing Gröbner bases of the hard-decision MLD algorithm in this section for practical application.

Algorithm 4.2 The Conti-like MLD algorithm for the soft-decision (Algorithm 3.5 revised)

Input: The received vector $\mathbf{r} \in \mathbb{R}^n$ and the parity check matrix H.

Output: The most likely codeword.

1: Prepare a vector $\mathbf{w}$.
2: Compute a Gröbner basis $\mathcal{G}$ of $I_{\Lambda(H)}$ with the weighted order $\prec_{\mathbf{w}}$.
3: Compute the normal form of a monomial by $\mathcal{G}$.
4: Return the left-half of the exponent vector of the normal form.

Algorithm 4.3 The Conti-like MLD algorithm for the hard-decision (Algorithm 3.3 revised)

Input: The received vector $\mathbf{r} \in \{0, 1\}^n$ and $\mathcal{G}$, where $\mathcal{G}$ is a Gröbner basis of $I_H \cap F[Y_1, \ldots, Y_n]$ and a parity check matrix H.

Output: An estimated error vector.

1: Prepare a monomial $f_{\mathbf{r}}$.
2: Compute the normal form of $f_{\mathbf{r}}$ by $\mathcal{G}$.
3: Return the exponent vector of the normal form.

4.4 Computing Gröbner bases

4.4.1 In the original Conti-Traverso algorithm

In order to focus upon the computation of a Gröbner basis of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$ at the step 1 of the modified Conti-Traverso algorithm (Algorithm 4.1), we introduce the definition of $\hat{I}_A$. Let A be a non-negative integral $m \times n$ matrix and $\mathbf{a}_i \in \mathbb{Z}^m_{\geq 0}$ be the i-th column vector of A with $i = 1, \ldots, n$. Define $\hat{I}_A \subset F[X_1, \ldots, X_m, Y_1, \ldots, Y_n]$ as follows:

$$\hat{I}_A := \langle Y_i - \mathbf{X}^{\mathbf{a}_i} : i = 1, \ldots, n \rangle.$$

We can find a Gröbner basis of

$$\hat{I}_A \cap F[Y_1, \ldots, Y_n]$$

after computing a Gröbner basis of $\hat{I}_A$ by Buchberger algorithm since the elimination theorem [1, 7] leads the former Gröbner basis equals polynomials which consisits of Y_i's in the latter Gröbner basis with an suitable monomial order.

However, the previous method is harder than to compute the Gröbner basis directory in general, since the number of variables and the number of generators of $\hat{I}_A$ are larger than one of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$, respectively.

There are two methods for the direct computation which is proposed by Hosten and Sturmfels [21], and by Di Biase and Urbanke [10]. In this section, we refer [45] for describing their methods. For computing Gröbner bases, Buchberger algorithm needs generators of the ideal as input. However, it is not clear to write the generators of $I_A \cap F[Y_1, \ldots, Y_n]$ explicitly. Then we meet a problem how to determine the generators. First we introduce a basic lemma which exposes the generators theoretically.

Lemma 4.1.

$$\begin{aligned}\hat{I}_A \cap F[Y_1, \ldots, Y_n] &= \mathrm{Span}_F\{\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} : A\mathbf{u} = A\mathbf{v}, \mathbf{u}, \mathbf{v} \in \mathbb{Z}^n_{\geq 0}\} \\ &= \langle \mathbf{Y}^{\mathbf{u}^+} - \mathbf{Y}^{\mathbf{u}^-} : A\mathbf{u} = 0, \mathbf{u} \in \mathbb{Z}^n, \mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-, \mathbf{u}^+, \mathbf{u}^- \in \mathbb{Z}^n_{\geq 0} \rangle,\end{aligned}$$

where $\mathrm{Span}_F\{f_1, \ldots, f_s\}$ *for* $f_1, \ldots, f_s \in F[Y_1, \ldots, Y_n]$ *means a vector space over* F *which is generated by* $f_1, \ldots, f_s$ *as a* F*-basis.*

Proof. See Lemma 4.1, Corollary 4.3 and Algorithm 4.5 in [45]. □

Unfortunately, this lemma is not useful to compute the generators since it is hard to find pairs $(\mathbf{u}, \mathbf{v})$ which satisfies $A\mathbf{u} = A\mathbf{v}$, or since the number of vectors where $A\mathbf{u} = 0$ is infinite in general. Therefore we needs other techniques to find generators explicitly.

First we introduce Hosten-Sturmfels' method. Let $\mathrm{Ker}_{\mathbb{Z}} A = \{\mathbf{u} \in \mathbb{Z}^n : A\mathbf{u} = 0\}$. The kernel $\mathrm{Ker}_{\mathbb{Z}} A$ forms a lattice over $\mathbb{Z}$. Let $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_s\} \subset \mathbb{Z}^n$ be a lattice basis of $\mathrm{Ker}_{\mathbb{Z}} A$. We can compute E explicitly as *Hermite normal form* by the LLL algorithm which is found by Lenstra, Lenstra and Lovász [33]. See also Algorithm 2.3.1 and Algorithm 2.4.10 in [4] for details on the LLL algorithm. Then we associate the following ideal $\hat{J}_E$ in $F[Y_1, \ldots, Y_n]$ with $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$;

$$\hat{J}_E = \langle \mathbf{Y}^{\mathbf{e}_i^+} - \mathbf{Y}^{\mathbf{e}_i^-} : i = 1, \ldots, s, \mathbf{e}_i = \mathbf{e}_i^+ - \mathbf{e}_i^-, \mathbf{e}_i^+, \mathbf{e}_i^- \in \mathbb{Z}^n_{\geq 0} \rangle.$$

Since $A\mathbf{e}_i^+ = A\mathbf{e}_i^-$ for every $i = 1, \ldots, s$, Lemma 4.1 says that $\hat{J}_E$ is a subideal of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$. The next lemma shows that an ideal quotation of $\hat{J}_E$ helps to find the generators of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$.

Lemma 4.2 ([21]).

$$\hat{I}_A \cap F[Y_1, \ldots, Y_n] = (\hat{J}_E : (Y_1 \cdots Y_n)^\infty)$$

Proof. See Lemma 12.2 in [45]. □

Hosten and Sturmfels have developed an algorithm to compute generators of the ideal quotation $(\hat{J}_E : (Y_1 \cdots Y_n)^\infty)$. Refer Algorithm 12.3 in [45] for detail.

On the other hand, Di Biase and Urbanke have developed another algorithm to compute generators of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$ using the subideal $\hat{J}_E$ by the following lemma.

Lemma 4.3 ([10]). *If there exists $\mathbf{e}_i \in E$ whose coordinates are all positive for some i, then $\hat{J}_E = \hat{I}_A \cap F[Y_1, \ldots, Y_n]$.*

Proof. See Lemma 12.4 in [45]. □

In general, since every vector $\mathbf{e}_i \in E$ has some negative coordinates, while the lattice basis E of $\mathrm{Ker}_{\mathbb{Z}} A$ is not defined uniquely, Lemma 4.3 may not be applied. However, Lemma 4.3 can be useful to compute generators of $\hat{I}_A \cap F[Y_1, \ldots, Y_n]$ for combining a simple algorithm. Their algorithm is also described as Algorithm 12.6 in [45].

On both Hosten-Sturmfels and Di Biase-Urbanke algorithms, the subideal $\hat{J}_E$ plays an important role to compute generators of $I_A \cap F[Y_1, \ldots, Y_n]$. In the next section, we will consider an analogy of Lemma 4.1 and $\hat{J}_E$ for reducing the complexity of the Conti-like MLD algorithm.

4.4.2 In the Conti-like MLD

In this section, our concern turns into an integer program $\mathrm{IP}_{A,\mathbf{w},2}(\mathbf{b})$ for the Conti-like MLD algorithm from original Conti-Traverso algorithm. We focus the ideal I_A, which has been defined in section 3.3. Note that the following discussion in this section can be extended for $\mathrm{IP}_{A,\mathbf{w},q}(\mathbf{b})$ with an arbitrary prime $q \geq 2$ easily.

Let A be a binary $m \times n$ matrix and $\mathbf{a}_i \in \{0,1\}^n$ be the i-th column vector of A. Remind that the definition of the ideal I_A is following;

$$I_A := \langle Y_i - \mathbf{X}^{\mathbf{a}_i}, X_j^2 - 1 : i = 1, \ldots, n, j = 1, \ldots, m \rangle.$$

Assume $m < n$. Then there exists a binary $(n-m) \times n$ matrix A^* which satisifies $A^* A^t \equiv 0 \mod 2$, where A^t is the transpose of A. Especially, it is well-known that, for a parity check matrix H of binary codes, H^* becomes a generator matrix of the codes.

Let $\mathbf{a}_i^*$ be the i-th *row* vector of A^* and define a binomial ideal $J_{A^*} \subset F[Y_1, \ldots, Y_n]$ as follows;

$$J_{A^*} := \langle Y_i^{\mathbf{a}_i^*} - 1, Y_j^2 - 1 : i = 1, \ldots, n-m, j = 1, \ldots, n \rangle.$$

The key theorem of this section is following.

Theorem 4.4.

$$I_A \cap F[Y_1, \ldots, Y_n] = J_{A^*}$$

Remark that Theorem 4.4 looks like Lemma 4.3, but, our theorem does not need any assumption without only $m < n$. For the proof of the theorem, we needs another auxiliary lemma.

Lemma 4.5. *A binomial* $\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} \in I_A \cap F[Y_1, \ldots, Y_n]$ *for* $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^n_{\geq 0}$ *if and only if* $A\mathbf{u} \equiv A\mathbf{v} \mod 2$.

Proof. By the definition of the binomial, we have

$$\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} = \prod_{i:u_i=1} (Y_i - \mathbf{X}^{\mathbf{a}_i}) - \prod_{i:v_i=1} (Y_i - \mathbf{X}^{\mathbf{a}_i}) + \mathbf{X}^{A\mathbf{u}} - \mathbf{X}^{A\mathbf{v}}.$$

Therefore if $\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} \in I_A$, then $\mathbf{X}^{A\mathbf{u}} - \mathbf{X}^{A\mathbf{v}}$ must be in the ideal $\langle X_i^2 - 1 : i = 1, \ldots, n \rangle$. It leads $A\mathbf{u} \equiv A\mathbf{v} \mod 2$ immediately.

On the other hand, if $A\mathbf{u} \equiv A\mathbf{v} \mod 2$, then the statement $\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} \in I_A$ is followed straightly by Lemma 3.1 in section 3.3. □

Proof of Theorem 4.4. The statement of the theorem is reduced into the following two claims:

Claim 1:

$$I_A \cap F[Y_1, \ldots, Y_n] = \langle \mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}}, Y_i^2 - 1 : A\mathbf{u} \equiv A\mathbf{v} \mod 2, i = 1, \ldots, n \rangle.$$

Claim 2:

$$\langle \mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}}, Y_i^2 - 1 : A\mathbf{u} \equiv A\mathbf{v} \mod 2, i = 1, \ldots, n \rangle = J_{A^*}.$$

First, we have $Y_i^2 - 1 \in I_A$ with $i = 1, \ldots, n$ by Lemma 3.7. Lemma 4.5 leads the rest of proof of Claim 1.

We have $A\mathbf{a}_i^* \equiv 0 \mod 2$ with $i = 1, \ldots, n - m$ from the definition of $\mathbf{a}_i^*$. Therefore it is clear that the left hand ideal in claim 2 includes J_{A^*}. Conversely, suppose that $A\mathbf{u} \equiv A\mathbf{v} \mod 2$ for $\mathbf{u}, \mathbf{v} \in \{0, 1\}^n$. Then there exists $\mathbf{l}, \mathbf{u}', \mathbf{v}' \in \{0, 1\}^n$ such that

$$\begin{aligned} \mathbf{u} &\equiv \mathbf{u}' + \mathbf{l} \mod 2, \\ \mathbf{v} &\equiv \mathbf{v}' + \mathbf{l} \mod 2, \text{ and} \\ A\mathbf{u}' &\equiv A\mathbf{v}' \equiv \mathbf{0} \mod 2. \end{aligned}$$

We can write $\mathbf{u}'$ as $\mathbf{u}' = \sum_{i=1}^{n-m} p_i \mathbf{a}_i^*$ with some $p_i \in \{0,1\}$ with $i = 1,\ldots,n-m$. As similar manner, we can write also $\mathbf{v}' = \sum_{i=1}^{n-m} q_i \mathbf{a}_i^*$ for $q_i \in \{0,1\}$. Then we obtain

$$\begin{aligned}\mathbf{Y}^{\mathbf{u}} - \mathbf{Y}^{\mathbf{v}} &= \mathbf{Y}^{\mathbf{l}}(\mathbf{Y}^{\mathbf{u}'} - \mathbf{Y}^{\mathbf{v}'}) \\ &\equiv \mathbf{Y}^{\mathbf{l}}\Big(\prod_{i:p_i=1} \mathbf{Y}^{\mathbf{a}_i^*} - \prod_{i:q_i=1} \mathbf{Y}^{\mathbf{a}_i^*}\Big) \qquad \mod \langle Y_i^2 - 1 : i = 1,\ldots,n\rangle\end{aligned}$$

Since $\prod_{i:p_i=1} \mathbf{Y}^{\mathbf{a}_i^*} - \prod_{i:q_i=1} \mathbf{Y}^{\mathbf{a}_i^*} \in \langle \mathbf{Y}^{\mathbf{a}_i^*} - 1 : i = 1,\ldots,n-m\rangle$, we have proved Claim 2. □

The Conti-like MLD can be applied for the hard-decision MLD and the soft-decision MLD. In the hard-decision case, we consider an ideal I_H, where H is the parity check matrix of the code (See section 3.3 for details). Since H^* is a generator matrix of the code, the number of variables and one of generators of J_{H^*} is smaller than one of I_H (Table 4.2).

Table 4.2 Number of variables and generators in the hard-decision MLD

	variables	generators
section 3.3, [26]	$2n-k$	$3n-k$
proposed in this section	n	$n+k$

On the other hand, in the soft-decision case, we consider an ideal $I_{\Lambda(H)}$, where $\Lambda(H)$ is the Lawrence lifting of H. Since $\Lambda(H)^*$ is the $k \times 2n$ matrix $(H^* H^*)$, the number of variables and one of generators of $J_{\Lambda(H)^*}$ is also smaller than one of $I_\Lambda(H)$ (Table 4.3).

Table 4.3 Number of variables and generators in the soft-decision MLD

	variables	generators
section 3.3, [26]	$4n-k$	$6n-k$
proposed in this section	$2n$	$2n+k$

4.5 Conclusion of this chapter

We present another construction of generators of an ideal which is appeared in the Conti-like MLD [26]. We show that the number of variables and the number of generators can be reduced for applying Buchberger algorithm to the Conti-like MLD. However, unfortunately, the improved Conti-like MLD takes much time and spaces because of the complexity of Buchberger algorithm. Another computation method for Gröbner bases is needed in future works.

5. Conclusion

In this thesis, two different approaches to realizing decoding algorithms have been discussed. Both approaches can be used for arbitrary binary linear block codes.

In chapter 2, we investigated a sub-optimum algorithm which makes use of the techniques of the ordered-statistics decoding and the maximum likelihood decoding. The algorithm can be regarded as an extension of the OSD algorithm with 0-th order reprocessing. The aim of our extension is to provide much flexibility for the OSD decoding. By adjusting the parameter t, we can control both decoding complexity and error correcting performance simultaneously. We presented analytical estimation of the complexity, and evaluated the complexity and the error performance by computer simulation. Combining Kaji's call-by-need decoding algorithm, we could show that our algorithm achieves almost the same performance and complexity as the OSD algorithm with 2-nd order reprocessing. We remark that our algorithm can employ an arbitrary MLD algorithm. Therefore, the complexity of the proposed algorithm can be further improved if a new efficient MLD algorithm is found in the future.

In chapter 3, we constructed a novel MLD algorithm based on an idea of utilizing Gröbner bases to solve integer programs with modulo arithmetics. For this sake, we extended Conti-Traverso algorithm to a modulo arithmetics case. Since an MLD of a binary linear block code can be regarded as an integer program with binary arithmetic, our extended Conti-Traverso algorithm induces a novel MLD algorithm. This approach provides a new viewpoint in coding theory and Gröbner bases theory. Unfortunately, our MLD algorithm based on Gröbner bases is not efficient in practice because computing Gröbner bases is hard. In the case of the hard-decision MLD, the Gröbner bases can be obtained by a pre-computation, which avoids the complexity issues of computing Gröbner bases. For another direction for this problem, we could find a class of codes which can be decoded efficiently with the Gröbner bases method.

In chapter 4, we have continued discussion for the Conti-like MLD. In this chapter, we have obtained another construction of generators of ideals in the Conti-like MLD for practical application. We can tie the results about decoding in the chapter together

with the theory of toric ideals, which is a popular topic in the recent algebraic geometry.

Bibliography

[1] Thomas Becker, Volker Weispfenning, and Heinz Kredel. *Gröbner bases – A Computational Approach to Commutative Algebra.* Springer-Verlag, 1993.

[2] David Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Transaction on Information Theory*, IT-18, pages 170–182, January 1972.

[3] Arjeh M. Cohen, Hans Cuypers, and Hans Sterk, editors. *Some Tapas of Computer Algebra*, Volume 4 of *Algorithms and Computation in Mathematics*. Springer, 1998.

[4] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1993.

[5] Pasqualina Conti and Carlo Traverso. Buchberger algorithm and integer programming. In Harold F. Mattson, Teo Mora, and T. R. N. Rao, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-9)*, number 539 in Lecture Notes in Computer Science. Springer-Verlag, pages 130–139, October 1991.

[6] John H. Conway and Neil J. A. Sloane. Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice. *IEEE Transaction on Information Theory*, IT-32, pages 41–50, January 1986.

[7] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer, second edition, 1997.

[8] David Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry.* Springer, 1998.

[9] Mario de Boer and Ruud Pellikaan. *Gröbner Bases for Decoding*, chapter 11, pages 260–275. In Cohen et al. [3], 1998.

[10] Fausto Di Biase and Rüdiger Urbanke. An algorithm to calculate the kernel of certain polynomial ring homomorphisms. *Experimental Mathematics*, 4, pages 227–234, 1995.

[11] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(3), pages 329–344, 1993.

[12] Jeanne Fitzgerald. *Applications of Gröbner bases to linear codes*. PhD thesis, Louisiana State Univ., 1996.

[13] G. David. Forney Jr. Generalized minimum distance decoding. *IEEE Transaction on Information Theory*, IT-12, pages 125–131, April 1966.

[14] G. David. Forney Jr. The Viterbi algorithm. In *Proceedings of the IEEE*, Volume 61, pages 268–278, 1973.

[15] G. David. Forney Jr. Coset codes II: Binary lattices and related codes. *IEEE Transaction on Information Theory*, 34, pages 1152–1187, September 1988.

[16] Marc P. C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes on ordered statistics. *IEEE Transaction on Information Theory*, 41(5), pages 1379–1396, September 1995.

[17] Mark P. C. Fossorier and Shu Lin. Soft decision decoding of linear block codes based on ordered statistics for the rayleigh fading channel with coherent detection. *IEEE Transactions on Communications*, COM-45, pages 12–14, January 1997.

[18] Toru Fujiwara, H. Yamamoto, Tadao Kasami, and Shu Lin. A trellis-based recursive maximum likelihood decoding algorithm for linear codes. *IEEE Transaction on Information Theory*, 44(2), pages 714–729, March 1998.

[19] Yunghsiang Sam Han, Carlos R. P. Hartman, and Chih-Chieh Chen. Efficient priority-first search maximum-likelihood soft-decision decoding of linear block

codes. *IEEE Transaction on Information Theory*, 39(5), pages 1514–1523, September 1993.

[20] Yunghsiang Sam Han, Carlos R. P. Hartman, and Kishan G. Mehrotra. Further results on decoding linear block codes using a generalized Dijkstra's algorithm. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 1994)*, Trondheim, Norway. page 342, June 1994.

[21] Serkan Hosten and Bernd Sturmfels. *GRIN: An Implementation of Gröbner Bases for Integer Programming*, pages 267–276. Number 920 in Springer Lecture Notes in Computer Science. Springer, 1995.

[22] Daisuke Ikegami. Maximum likelihood decoding algorithm using Gröbner bases (in japanese). In Hidehumi Ohsugi, editor, *Report on RIMS, Kyoto University*, number 1289, pages 110–121, May 2002.

[23] Daisuke Ikegami and Yuichi Kaji. Decoding linear block codes using the ordered statistics and MLD techniques. In *The 24th Symposium on Information Theory and Its Applications(SITA 2001)*, pages 459–462, December 2001.

[24] Daisuke Ikegami and Yuichi Kaji. A soft-decision MLD algorithm for linear block codes using Gröbner bases. In *The 24th Symposium on Information Theory and Its Applications(SITA 2001)*, pages 545–548, December 2001.

[25] Daisuke Ikegami and Yuichi Kaji. The soft-decision MLD of linear block codes, integer programming and Gröbner bases. In *Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT 2002)*, Lausanne, Switzerland. the IEEE Information Theory Society, page 316, July 2002.

[26] Daisuke Ikegami and Yuichi Kaji. Maximum likelihood decoding for linear block codes using Gröbner bases. *to appear in IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(3), March 2003.

[27] Daisuke Ikegami and Yuichi Kaji. On the construction of ideals in the Conti-like MLD algorithm. submitted to IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2003.

[28] Yuichi Kaji and Daisuke Ikegami. Decoding linear block codes using the ordered-statistics and the MLD techniques. In *Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT 2002)*, Lausanne, Switzerland. the IEEE Information Theory Society, page 144, July 2002.

[29] Yuichi Kaji, Hitoshi Tokushige, and Tadao Kasami. An improved search algorithm for the adaptive and recursive MLD algorithm. In *2001 IEEE International Symposium on Information Theory*, Washington, D.C. page 334, June 2001.

[30] Toshinobu Kaneko, Toshihisa Nishijima, Hiroshige Inazumi, and Shigeichi Hirasawa. Simplified correlation decoding by selecting codewords using erasure information. *IEEE Transaction on Information Theory*, 40(2), pages 320–327, March 1994.

[31] Tadao Kasami, Toyoo Takata, Toru Fujiwara, and Shu Lin. On complexity of trellis structure of linear block codes. *IEEE Transaction on Information Theory*, 39(3), pages 1057–1064, May 1993.

[32] Alec Lafourcade and Alexander Vardy. Optimum sectionalization of a trellis. *IEEE Transaction on Information Theory*, 13(3), pages 689–702, May 1996.

[33] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematics Annals*, 261, pages 515–534, 1982.

[34] Shu Lin, Tadao Kasami, Toru Fujiwara, and Marc Fossorier. *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*. Kluwer Academic Publishers, 1998.

[35] Niels J. C. Lous, Patrick A. H. Bours, and Henk C. A. van Tilborg. On maximum likelihood soft-decision decoding of binary linear codes. *IEEE Transaction on*

Information Theory, 39, pages 197–203, January 1993.

[36] F. Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*, Volume 16 of *North-Holland Mathematical Library*. North-Holland, ninth edition, 1996.

[37] Douglas J. Muder. Minimal trellises for block codes. *IEEE Transaction on Information Theory*, 34, pages 1049–1053, September 1988.

[38] Hidefumi Ohsugi, Daisuke Ikegami, Tomonori Kitamura, and Takayuki Hibi. Gröbner bases of certain zero-dimensional ideals arising in coding theory. to appear, Advances in Applied Mathematics, 2002.

[39] Jim K. Omura. On the Viterbi decoding algorithm. *IEEE Transaction on Information Theory*, IT-15, pages 177–179, 1969.

[40] Loc Pottier. Gröbner bases of toric ideals. Technical report, INRIA Sophia Antipolis, Rapport de recherche 2224, 1994. Manuscript available from `http://citeseer.nj.nec.com/pottier94grbner.html`.

[41] John G. Proakis. *Digital Communications*. McGraw-Hill, Inc., second edition, 1989.

[42] Jakov Snyders. On survivor error patterns for maximum likelihood soft decoding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT 1991)*, Budapest, Hungary. page 192, June 1991.

[43] Jakov Snyders. Reduced lists of error patterns for maximum likelihood soft decoding. *IEEE Transaction on Information Theory*, 39, pages 197–203, January 1993.

[44] Jakov Snyders and Yair Be'ery. Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes. *IEEE Transaction on Information Theory*, 35, pages 963–975, September 1989.

[45] Bernd Sturmfels. *Gröbner Bases and Convex Polytopes*, Volume 8 of *University Lecture*. American Mathematical Society, 1995.

[46] Bernd Sturmfels and Rekha Thomas. Variation of cost functions in integer programming. Technical report, School of Operations Research, Cornell University, 1994. Manuscript is available from `http://www.math.washington.edu/~thomas/articles.html`.

[47] Bernd Sturmfels, Robert Weismantel, and Günter M. Ziegler. Gröbner bases of lattices, corner polyhedra, and integer programming. *Beiträge zur Algebra und Geometrie*, 36, pages 281–298, 1995.

[48] Dana J. Taipale and Michael B. Pursley. An improvement to generalized-minimum-distance decoding. *IEEE Transaction on Information Theory*, 37, pages 167–172, January 1991.

[49] Hatsukazu Tanaka and Koji Kakigahara. Simplified correlation decoding by selecting codewords using erasure information. *IEEE Transaction on Information Theory*, IT-29, pages 743–748, September 1983.

[50] Rekha Thomas. A geometric Buchberger algorithm for integer programming. *Mathematics of Operations Research*, 20, pages 864–884, 1995.

[51] Regina Urbaniak, Robert Weismantel, and Günter M. Ziegler. A variant of buchberger's algorithm for integer programming. *SIAM Journal on Discrete Mathematics*, 1(10), pages 96–108, 1997.

[52] Andrew J. Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transaction on Information Theory*, IT-13, pages 260–269, 1967.