NAIST-IS-DT9661011

Doctor's Thesis

Study on Iterative Soft-Decision Decoding Algorithms for Binary Linear Block Codes

Takuya Koumoto

February 8, 1999

Department of Information Science
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

Takuya Koumoto

Thesis committee: Toru Fujiwara, Professor
Masaki Koyama, Professor
Hiroyuki Seki, Professor
Yuichi Kaji, Associate Professor

Study on Iterative Soft-Decision Decoding Algorithms for Binary Linear Block Codes[*]

Takuya Koumoto

Abstract

For iterative soft-decision decoding algorithms for binary linear block codes, error performance and computational complexity are very important factors. In this thesis, sufficient conditions of the optimality of decoded codeword and a sufficient condition which is used for ruling-out some useless iterations of bounded distance decoding which reduces the computational complexity of iterative soft-decision decoding algorithms are shown. New iterative soft-decision decoding algorithms which uses the sufficient conditions of the optimality of decoded codeword to achieve good error performance with small computational complexity are also shown.

First, the sufficient conditions are derived based on: (1) partial knowledge of the distance profile of the code, and (2) previously generated candidate codewords. When the number of previously generated candidate codewords is greater than 1, the conditions presented here are less stringent than those derived by Taipale and Pursley[4] and Kaneko et. al.[5]. The condition based on three previously generated candidate codewords is the first such sufficient condition that has ever been derived. It is shown by computer simulation that these sufficient condions reduce the computational complexity of iterative decoding algorithms without degradation of error performance.

Second, a sufficient condition to rule-out some useless test error patterns is derived. In an iterative decoding algorithm, such as Chase Type-II decoding algorithm[2] and its improved versions, candidate codewords for a received vector are generated

[*]Doctor's Thesis, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9661011, February 8, 1999.

for test based on a bounded-distance decoder and a set of test error patterns. It is desirable to remove useless test error patterns in these decoding algorithms. If it is assured by the sufficient condition that a test error pattern never generates a better candidate codeword than already generated candidate codewords, then the bounded distance decoding can be skipped. This significantly reduces the decoding operations in Chase type-II decoding algorithm or decoding iterations in its improvements.

Finally, the author also presents a new low-weight trellis-based soft-decision iterative decoding algorithm for binary linear block codes. The algorithm is devised based on a set of optimality conditions and the generation of a sequence of candidate codewords for optimality test. The initial candidate codeword is generated by a simple decoding method. The subsequent candidate codewords, if needed, are generated by a chain of low-weight trellis searches, one at a time. Each search is conducted through a low-weight trellis diagram centered around the latest candidate codeword and results in an improvement over the previous candidate codewords that have been already tested. This improvement is then used as the next candidate codeword for test of optimality. The decoding iteration stops whenever a candidate codeword is found to satisfy a sufficient condition on optimality or the latest low-weight trellis search results in a repetition of a previously generated candidate codeword. A divide-and-conquer technique is also presented for codes that are not spanned by their minimum weight codewords to achieve better error performance and small computational complexity. The proposed decoding algorithm has been applied to some well-known codes of lengths 48, 64 and 128. Simulation results show that the proposed algorithm achieves either practically optimal error performance for the example codes of length 48 and 64 or near optimal error performance for the (128,29,32) RM code with significant reduction in computational decoding complexity.

Keywords:

linear block code, iterative soft-decision decoding, maximum-likelihood decoding, optimality test, low-weight subtrellis, Chase algorithm, sufficient condition,

Acknowledgments

I am deeply indebted to many people for the advice, feedback and support they gave to me in the course of this work. I would especially like to thank Professor Toru Fujiwara and Tadao Kasami for their invaluable support, discussions and encouragement throughout the work.

I am grateful to Professors Masaki Koyama and Hiroyuki Seki for their suggestions on the work. I am also obliged to Associate Professor Yuichi Kaji for his helpful comments and suggestions. I would like to thank Professors Shu Lin and Marc P. C. Fossoire of Hawaii University for their comments and suggestions. I would like to thank Professor Toyoo Takata of Iwate Prefectural University for his technical support. I would like to thank Dr. Hajime Watanabe for his invaluable support.

Finally, I would like to thank all the members of Fujiwara Laboratory of Nara Institute of Science and Technology.

List of Publications

Journal Paper

[1] T. Koumoto, T. Kasami and S. Lin, “A Sufficient Condition for Ruling Out Some Useless Test Error Patterns in Iterative Decoding Algorithms,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E81-A, No. 2, pp. 321–326, February 1998.

[2] T. Koumoto, T. Takata, T. Kasami and S. Lin, “A Low-Weight Trellis Based Iterative Soft-Decision Decoding Algorithm for Binary Linear Block Codes,” to appear in IEEE Transactions on Information Theory, March 1999.

International Conferences

[3] T. Kasami, T. Koumoto, T. Takata and S. Lin, “The Effectiveness of the Least Stringent Sufficient Condition on the Optimality of Decoded Codewords,” Proceedings of the 3rd International Symposium on Communication Theory & Applications, pp.324–333, Charlotte Mason College, Ambleside, Lake District, UK, July 1995.

[4] T. Kasami, T. Koumoto, T. Takata and S. Lin, “The Least Stringent Sufficient Conditions on the Optimality of Decoded Codewords,” Proceedings of the 1995 IEEE International Symposium on Information Theory, p.470, Whistler, Canada, September 1995.

[5] T. Koumoto, T. Takata, T. Kasami and S. Lin, “An Iterative Soft-Decision Decoding Algorithm,” Proceedings of the International Symposium on Information Theory and Its Applications, pp. 806–810, Canada, September 1996.

Workshops

[6] T. Kasami, T. Takata, T. Koumoto, T. Fujiwara, H. Yamamoto and S. Lin, "The Least Stringent Sufficient Condition on Optimality of Suboptimal Decoded Codewords," Technical Report of IEICE, IT94-82, The Institute of Electronics, Information and Communication Engineers, Japan, January 1995.

[7] T. Koumoto, H. Nagano, T. Takata, T. Fujiwara, T. Kasami and S. Lin, "A New Iterative Soft-Decision Decoding Algorithm," Technical Report of IEICE, IT95-28, The Institute of Electronics, Information and Communication Engineers, Japan, July 1995.

[8] T. Koumoto, H. Nagano, T. Takata, T. Fujiwara, T. Kasami and S. Lin, "An Iterative Soft-Decision Decoding Algorithm," Proceedings of the 18th Symposium on Information Theory and Its Applications, pp.557–560, Japan, October 1995.

[9] T. Koumoto, T. Takata, T. Kasami and S. Lin, "Condition for Reducing the Number of Iterations of Iterative Decoding," Technical Report of IEICE, IT95-73, The Institute of Electronics, Information and Communication Engineers, Japan, March 1996.

[10] T. Koumoto and T. Kasami, "An Iterative Decoding Algorithm Based on Information of Decoding Failure," Proceedings of the 20th Symposium on Information Theory and Its Applications, pp.325–328, Japan, December 1997.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The iterative decoding algorithms are studied for long years to achieve good error performance with small computational complexity. There are some iterative decoding algorithms which uses algebraic decoder internally. In such decoding algorithms, the computational complexity has been likely to measured by **order** of code parameter rather than average number of addition equivalent operations. We should use the average computational complexity to measure the computational complexity of decoding algorithms with the same **order**. Then, to keep the average computational complexity small on iterative decoding algorithms, the importance of termination condition grows. Taipale and Puersley have proposed a sufficient condition on optimality of a decoded codeword[4] and Kaneko, Nishijima and Hirasawa have improved it for two candidate codewords[20]. Those sufficient conditions are used to reduce the average computational complexity of iterative decoding algorithms with no degradation of error performance.

The iterative decoding algorithms proposed in [1]–[9] are based on the generation of a sequence of candidate codewords by means of a simple decoder using the reliability information of the received symbols, and then choosing the best (or most likely) one among the generated candidate codewords as the decoded codeword. In these algorithms, the number of iterations for generating candidate codewords can be reduced without degrading the error performance by applying a sufficient condition to test the optimality of a candidate codeword when it is generated and terminating the iteration process as soon as the sufficient condition is satisfied. A less stringent sufficient condition on optimality, whose computational complexity is reasonably smaller than that of the procedure for generating the next candidate codeword, provides a faster termination of the decoding iteration process and hence reduces the computational complexity and decoding delay. In chapter 2, sufficient conditions on the optimality of a candidate codeword are investigated based on: (1) partial knowledge of the distance profile of the code; and (2) h or fewer previously generated candidate codewords. These conditions can be incorporated in any of the iterative soft-decision decoding algorithms based on the generation of a sequence of candidate codewords to reduce the number of decoding iterations significantly. The effectiveness of these conditions depends largely on the iterative decoding algorithm and signal to noise ratio (SNR) as well as the distance profile of the code. It is shown that the new sufficient conditions are very effective in terminating the decoding iteration process except for high block error probability region.

In such an iterative decoding algorithm as Chase type-II decoding algorithm[2], candidate codewords for a received vector are generated by using an algebraic decoder and a set of test error patterns that is formed based on the reliability measures of the received symbols. The candidate codeword that has the largest correlation metric with the received vector is chosen as the decoded codeword. This decoding is a bounded-distance decoding and achieves asymptotic error performance. However, during the decoding process, some of the test error patterns may result in decoding failures if a bounded-distance algebraic decoding is used and some may produce the same candidate codeword (repetition). These result in unnecessary decoding operations and hence prolongs the decoding delay. Therefore, it is desirable to remove these useless or redundant error patterns before or during the decoding process.

To the author's knowledge, the first study on a sufficient condition for ruling out some useless test error patterns to reduce the number of bounded-distance t decodings was made by Kaneko, Nishijima and Hirasawa in [20]. We call the sufficient condition present in [20] the KNH condition. Chapter 3 presents a more effective sufficient condition for ruling out some useless test error pattern than the KNH condition. The ruling out is based on the following conditions: (1) Under a reasonable restriction on the order of generating test error patterns (EG condition stated in Sec. 3.3), a necessary and sufficient condition (stated in Lemma 1) that the decoded codeword, $dec(\boldsymbol{e})$, by a bounded-distance decoding for a nonzero test error pattern $\boldsymbol{e}$ is different from all candidate codewords that have been generated already, and (2) the Hamming distance between the latest candidate codeword generated already, denoted $\boldsymbol{c}$, and $dec(\boldsymbol{e})$ is at least the minimum distance of the code. If test error patterns are assumed to be generated in binary order and the condition (1) above is only taken into consideration, the sufficient condition presented in this paper is reduced to the KNH condition as stated in Sec. 3.4.1. In this sense, the new condition is an extension and improvement of the KNH condition.

The complexity for testing the ruling out condition is considerably smaller than that of a bounded-distance algebraic decoding. Each time when a new test error pattern is generated, it is tested based on this sufficient condition. If the condition holds, then this error pattern can not produce a candidate codeword with a correlation metric larger than those of the candidate codewords generated already and hence it is useless. In this case, the error pattern is ruled out for decoding and the next test error pattern is generated unless the test error patterns have been exhausted. This reduces the number of bounded-distance decoding operations and the decoding delay.

The computational complexity of iterative decoding algorithms is reduced by above sufficient conditions. The remaining problem is the error performance. The application of trellis-based maximum likelihood decoding (MLD) algorithms is limited due to the prohibitively large trellises for codes of long block lengths. To overcome the state and branch complexity problems of large trellises for long block codes, several new approaches have been proposed [21, 22, 23, 24, 5, 25, 26, 16]. Most recently, Moorthy, Lin and Kasami have shown that the minimum-weight subtrellis of a code is sparsely connected and has much simpler state and branch complexities than the full code trellis[27]. Based on this fact, they proposed a minimum-weight subtrellis-based iterative decoding algorithm for linear block codes to achieve suboptimum error performance with a drastic reduction in decoding complexity compared with a trellis-based MLD algorithm, using a full code trellis.

In chapter 4, a new low-weight subtrellis based iterative decoding algorithm which overcomes the major shortcomings of the Moorthy-Lin-Kasami (MLK) algorithm as described above is shown. A low weight subtrellis, we mean a subtrellis of the code trellis that consists of only codewords of low weights, say minimum and next to the minimum weights. The new algorithm is different from the MLK algorithm in the generation of candidate codewords, optimality test, and termination of the decoding process. It has the following important properties. The initial candidate codeword is first generated by a simple decoding method that guarantees a successful decoding, such as the zero-th or the first-order decoding based on the ordered statistics of the received symbols proposed in [17] or combined with an algebraic decoding, called hybrid method. These decodings are very simple and always produce a decoded codeword (no decoding failure). Subsequent candidate codewords, if needed, are generated by a chain of low weight subtrellis searches. Each such search is centered around the current candidate codeword. The current candidate codeword is excluded in the search to prevent repetition and to reduce the possibility of being trapped into a local optimum. The codeword with the largest correlation metric resulting from this low weight subtrellis search is then used as the next candidate codeword. Candidate codewords are generated in the order of improving correlation metrics.

For codes that are spanned by their minimum weight codewords, the proposed decoding algorithm is very effective. However, for codes that are not spanned by their minimum weight codewords, minimum weight (or next to the minimum weight) subtrellis search does not provide a big enough search space and results in a significant degradation in error performance compared with MLD. To overcome this problem, a divide-and-conquer technique to partition the code space into cosets with respect to a subcode which is spanned by the minimum weight codewords is used. Then the low weight subtrellis search is performed over the cosets in the partition, one at a time, based on a likely order. The search is shifted from one coset to another until the ML codeword is found or all the cosets are exhausted. This results in a good coverage of the entire code space and good error performance, while maintains low weight subtrellis searches to keep the decoding complexity down.

Chapter 2

The Effectiveness of the Least Stringent Sufficient Condition on the Optimality of Decoded Codewords

2.1 Introduction

The number of iterations of an iterative optimal or suboptimal decoding scheme [1]–[8] for binary linear block codes can be reduced without any effect on its error performance by testing a sufficient condition on the optimality of a candidate codeword. In this chapter, the least stringent sufficient condition on the optimality of a decoded codeword is investigated under the assumption that the available information on the code is restricted to (1) the minimum weight and a few small weights and (2) for a given positive integer h, h or fewer already generated candidate codewords. The least stringent sufficient conditions of optimality for $1 \leq h \leq 3$, denoted $Cond_h$, are presented. $Cond_1$ where only the minimum weight is considered is the same as the one derived by Taipale and Pursley in [4], $Cond_2$ is less stringent than the one given by Kaneko et al. [5], and $Cond_1$ and $Cond_2$ are derived from $Cond_3$ as special cases.

As examples, we consider the Chase algorithm II [2] modified by introducing an early termination condition $Cond_h$ for $\mathrm{RM}_{5,1}$, $\mathrm{RM}_{5,2}$, $\mathrm{RM}_{6,2}$, and $\mathrm{RM}_{6,3}$, where

$\text{RM}_{m,r}$ denotes the r-th order Reed-Muller code of length 2^m. Majority-logic decoding with randomly breaking ties is used to generate candidate codewords. We also consider new iterative decoding schemes presented in [6]–[8]. For an AWGN channel using BPSK signaling, the ratio of the number of decoded codewords for which the sufficient condition is satisfied to that of decoded codewords which are optimal, and the average reduction in the number of iterations, for $1 \leq h \leq 3$, are evaluated by simulation.

2.2 Sufficient Conditions on the Optimality of a Decoded Codeword

Suppose a binary block code C of length N with distance profile $W \triangleq \{0, w_1 = d_{\min}, w_2, \ldots\}$ is used for error control over an AWGN channel using BPSK signaling. A codeword $\boldsymbol{c}$ is mapped into a bipolar sequence $\boldsymbol{x}$. Suppose $\boldsymbol{x}$ is transmitted and $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ is a received sequence at the output of a matched filter in the receiver. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_N)$ be the binary hard-decision sequence obtained from $\boldsymbol{r}$.

Let V^N denote the set of all binary N-tuples. For $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ in V^N, define the following:

$$D_1(\boldsymbol{u}) \triangleq \{i : u_i \neq z_i, \text{ and } 1 \leq i \leq N\}, \tag{2.2.1}$$

$$D_0(\boldsymbol{u}) \triangleq \{1, 2, \ldots, N\} - D_1(\boldsymbol{u}), \tag{2.2.2}$$

$$n(\boldsymbol{u}) \triangleq |D_1(\boldsymbol{u})|, \tag{2.2.3}$$

$$L(\boldsymbol{u}) \triangleq \sum_{i \in D_1(\boldsymbol{u})} |r_i|. \tag{2.2.4}$$

For a subset U of V^N, let $\underline{L}[U]$ be defined as $\underline{L}[U] \triangleq \min_{\boldsymbol{u} \in U} L(\boldsymbol{u})$. If U is empty, then $\underline{L}[U]$ is defined as ∞ (infinity). For maximum likelihood decoding (MLD), the decoder finds the optimal codeword $\boldsymbol{c}_{\text{opt}}$ [5], for which

$$L(\boldsymbol{c}_{\text{opt}}) = \underline{L}[C]. \tag{2.2.5}$$

For two codewords $\boldsymbol{c}$ and $\boldsymbol{c}'$, $\boldsymbol{c}$ is said to be better than $\boldsymbol{c}'$ if $L(\boldsymbol{c}) \leq L(\boldsymbol{c}')$. A candidate codeword $\boldsymbol{c}$ is said to be the best if $L(\boldsymbol{c})$ is the minimum among the candidate codewords that have been generated already.

Let $d_H(\boldsymbol{u}, \boldsymbol{v})$ denote the Hamming distance between $\boldsymbol{u}$ and $\boldsymbol{v}$. For $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, ..., $\boldsymbol{u}_h \in V^N$, positive integers $d_1, d_2, \ldots, d_h$ and a subset U of V^N, let $U(\boldsymbol{u}_1, d_1;\, \boldsymbol{u}_2, d_2;\, \ldots;\, \boldsymbol{u}_h, d_h)$ (or $U_{d_1,d_2,\ldots,d_h}$ if $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_h$ are clear from the context) be defined as

$$U(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h) \triangleq \{\boldsymbol{u} \in U : d_H(\boldsymbol{u}, \boldsymbol{u}_i) \geq d_i \text{ for } 1 \leq i \leq h\}. \tag{2.2.6}$$

Then we have the following lemma.

__Lemma 2.1__ **:** At a stage of an iterative decoding algorithm, suppose that
(i) candidate codewords $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_h$ have been generated, and let $\boldsymbol{u}_{\text{best}}$ denote the best of all the candidate codewords that have been generated already,
(ii) for $d_1, d_2, \ldots, d_h \in W - \{0\}$,

$$L(\boldsymbol{u}_{\text{best}}) = \underline{L}[\cup_{i=1}^{h}\{\boldsymbol{u} \in C : d_H(\boldsymbol{u}, \boldsymbol{u}_i) < d_i\}], \tag{2.2.7}$$

and (iii) any candidate codeword that will be generated in a later stage belongs to a subset U of V^N.

Then, $\boldsymbol{u}_{\text{best}}$ is the best candidate codeword that can be generated by the decoding algorithm, if

$$L(\boldsymbol{u}_{\text{best}}) \leq \underline{L}[U(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h)]. \tag{2.2.8}$$

△△

The above condition can be used as a termination condition of an iterative decoding algorithm without any effect of the error probability. If $C \subseteq U$ in (2.2.8), then the condition (2.2.8) is a sufficient condition that $\boldsymbol{u}_{\text{best}}$ is the optimal. For instance, if C is an even weight code, the set of all even weight binary N-tuples, denoted V^N_{even}, can be chosen as U. Of course, the complexity to find the right-hand side of (2.2.8) must be less than the average reduction of the complexity of the succeeding iterations due to the introduction of the termination condition.

Condition (2.2.7) holds always for $d_i = w_1 = d_{\min}$ with $1 \leq i \leq h$. In the decoding algorithms proposed in [6, 7], "minimum-weight sub-trellis search around a codeword $\boldsymbol{u}$" which gives the best codeword of $\{\boldsymbol{v} \in C : d_H(\boldsymbol{v}, \boldsymbol{u}) \leq w_1\}$ is used. If such a search around $\boldsymbol{u}_i$ is to be done, then we can set $d_i = w_2$.

For a positive integer h, let B^h be the set of all binary sequences of length h. For $1 \leq i \leq h$ and $\alpha \in B^h$, let $pr_i(\alpha)$ denote the i-th bit of α. For $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_h$

and $\boldsymbol{u}$ in V^N and $\alpha \in B^h$, let D_α and q_α be defined as

$$D_\alpha \triangleq \bigcap_{i=1}^{h} D_{pr_i(\alpha)}(\boldsymbol{u}_i), \tag{2.2.9}$$

$$n_\alpha \triangleq |D_\alpha|, \tag{2.2.10}$$

$$q_\alpha \triangleq |D_1(\boldsymbol{u}) \cap D_\alpha|. \tag{2.2.11}$$

It is shown in [10] that $\boldsymbol{u} \in V^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h)$ if and only if

$$\sum_{\alpha \in B^h} (-1)^{pr_i(\alpha)} q_\alpha \geq \delta_i \triangleq d_i - n(\boldsymbol{u}_i), \text{ for } 1 \leq i \leq h. \tag{2.2.12}$$

Let Q denote the set of those 2^h-tuples $\boldsymbol{q} = (q_{00\cdots0}, q_{0\cdots01}, \ldots, q_{11\cdots1})$'s over nonnegative integers which satisfy (2.2.12). We say, $\boldsymbol{q} = (q_{00\cdots0}, q_{0\cdots01}, \ldots, q_{11\cdots1}) \in Q$ is minimal if and only if there is no $\boldsymbol{q}' = (q'_{00\cdots0}, q'_{0\cdots01}, \ldots, q'_{11\cdots1}) \in Q$ such that $\boldsymbol{q} \neq \boldsymbol{q}'$ and $q_\alpha \geq q'_\alpha$ for $\alpha \in B^h$. Let $Q_{\min}$ denote the set of minimal tuples in Q. For $\boldsymbol{q} \in Q_{\min}$, $q_{11\cdots1} = 0$.

For $\alpha = a_1a_2\cdots a_h \in B^h$, let $\bar{\alpha}$ denote $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_h \in B^h$, where if $a = 0$, then $\bar{a} = 1$ and otherwise, $\bar{a} = 0$. For each unordered pair $(\alpha, \bar{\alpha})$ with $\alpha \in B^h$, we choose α or $\bar{\alpha}$ and let B_0^h denote the set of 2^{h-1} chosen binary sequences.

For $\alpha \in B_0^h$, let y_α be defined as $y_\alpha \triangleq q_\alpha - q_{\bar{\alpha}}$. Then, the inequality of (2.2.12) can be rewritten as follows:

$$\sum_{\alpha \in B_0^h} (-1)^{pr_i(\alpha)} y_\alpha \geq \delta_i. \tag{2.2.13}$$

For $\boldsymbol{q} \in Q_{\min}$,

$$q_{\bar{\alpha}} = 0, \text{ if } y_\alpha \geq 0, \tag{2.2.14}$$

$$q_\alpha = 0, \text{ if } y_\alpha \leq 0. \tag{2.2.15}$$

Thus the number of variables in the key inequalities (2.2.12) is reduced to half.

For simplicity, we assume that the bit positions $1, 2, \ldots, N$ are ordered according to the following increasing order of reliability, i.e. $|r_i| \leq |r_j|$, for $1 \leq i < j \leq N$. For a subset X of $\{1, 2, \ldots, N\}$ and a positive integer $j \leq |X|$, let $X^{(j)}$ denote the set of j smallest integers in X. For a nonpositive integer $j, X^{(j)} \triangleq \phi$ and for $j \geq |X|, X^{(j)} \triangleq X$. Then, the following lemma holds [10].

Lemma 2.2 : **(1)** If $Q_{\min} \neq \phi$, then

$$\underline{L}[V^N_{d_1,d_2,\ldots,d_h}] = \min_{\boldsymbol{q}\in Q_{\min}} \sum_{i\in\bigcup_{\alpha\in B^h} D_\alpha^{(q_\alpha)}} |r_i|. \tag{2.2.16}$$

(2) If C is an even weight code, then

$$\underline{L}[V^N_{\text{even } d_1,d_2,\ldots,d_h}] = \underline{L}[V^N_{d_1,d_2,\ldots,d_h}]. \tag{2.2.17}$$

2.3 Examples

Without loss of generality, assume that $\delta_i \geq \delta_j$ for $i < j$ in Examples 2.1 and 2.2.

Example 2.1: Let $h = 2$. It is proved in [10] that

$$\underline{L}[V^N_{d_1,d_2}] = \sum_{i\in(D_{00}\cup D_{01}^{(\lfloor(\delta_1-\delta_2)/2\rfloor)})^{(\delta_1)}} |r_i|. \tag{2.3.1}$$

Consider a special case where $\boldsymbol{u}_1 = \boldsymbol{u}_2$ and $d_1 = d_2 = d_{\min} = w_1$, that is, $h = 1$. Then, $D_{00} = D_0(\boldsymbol{u}_1)$, $D_{01} = \phi$, $\delta_1 = \delta_2$ and equality (2.3.1) reduces to the following formula derived in [4]: $\underline{L}[V^N_{d_1}] = \sum_{i\in D_0(\boldsymbol{u}_1)^{(\delta_1)}} |r_i|$.

For $\boldsymbol{u}_1 \neq \boldsymbol{u}_2$, the right-hand side of (2.3.1) can be shown to be tighter than the lower bound $\sum_{i\in D_0(\boldsymbol{u}_2)^{(\lceil(\delta_1+\delta_2)/2\rceil)}} |r_i|$ given in [5].

Example 2.2: For $h = 3$, let δ'_2, δ'_3 and $\delta^{(1)}$ be defined as $\delta'_2 \triangleq \lfloor(\delta_1 - \delta_2)/2\rfloor$, $\delta'_3 \triangleq \lfloor(\delta_1 - \delta_3)/2\rfloor$, $\delta^{(1)} \triangleq \min(\delta'_2, \delta'_3)$. Let L_1 be defined as

$$L_1 \triangleq \min_{0\leq\delta\leq\delta^{(1)}} \sum_{i\in(D_{000}\cup D_{001}^{(\delta'_3-\delta)}\cup D_{010}^{(\delta'_2-\delta)})^{(\delta_1-\delta)}\cup D_{011}^{(\delta)}} |r_i|. \tag{2.3.2}$$

If the number of indices i's in the above summation is smaller than δ_1, define L_1 as ∞.

Consider the parities of δ_i with $1 \leq i \leq 3$. By the parity of an integer m, we mean the eveness or oddness of m. If δ_i belongs to the major parity of δ_1, δ_2 and δ_3, then $\triangle_i \triangleq 0$, otherwise $\triangle_i \triangleq 1$. Let δ''_2, δ''_3, $\delta^{(2)}$ and $\delta^{(3)}$ be defined as $\delta''_i \triangleq (\delta_1 + \triangle_1 - \delta_i - \triangle_i)/2$, $\delta^{(2)} \triangleq \max\{0, (\delta_2 + \triangle_2 + \delta_3 + \triangle_3)/2 - n_{000}\}$,

$\delta^{(3)} \triangleq \min\{n_{100},\, n_{010} - \delta_2'',\, n_{001} - \delta_3'',\, (\delta_2 + \triangle_2 + \delta_3 + \triangle_3)/2\}$. If $\delta^{(3)} \geq \delta^{(2)}$, let L_2 be defined as

$$L_2 \triangleq \min_{\delta^{(2)} \leq \delta \leq \delta^{(3)}} \sum_{i \in D_{000}^{((\delta_2+\triangle_2+\delta_3+\triangle_3)/2-\delta)} \cup D_{001}^{(\delta_3''+\delta)} \cup D_{010}^{(\delta_2''+\delta)}} |r_i|. \tag{2.3.3}$$

Otherwise, $L_2 \triangleq \infty$. Then, $\underline{L}[V^N_{d_1,d_2,d_3}]$ is given by $\min\{L_1, L_2\}$ [10].

$\triangle\triangle$

In the following Example 3 and 4, let $cond_h(d_1, d_2, \ldots, d_h)$ denote the termination condition (2.2.8) in which $U = V^N$, $\boldsymbol{u}_h$ is the latest candidate codeword and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{h-1}$ are the $(h-1)$ best candidate codewords other than $\boldsymbol{u}_h$ among all candidate codwrods that have been generated already.

For simplicity, if the number of candidate codewords generated already is less than h, then $cond_h(d_1, d_2, \ldots, d_h)$ is defined to be false. To show the difference between the effectiveness of $cond_h(d_1, d_2, \ldots, d_h)$ and that of $cond_{h-1}(d_2, d_3, \ldots, d_h)$ as early ternimation conditions, we define the following condition $\Delta_h(d_1, d_2, \ldots, d_h)$. Condition $\Delta_h(d_1, d_2, \ldots, d_h)$ is true, if and only if (i) at the current iteration step, $cond_h(d_1, d_2, \ldots, d_h)$ holds and $cond_{h-1}(d_2, d_3, \ldots, d_h)$ does not hold and (ii) at any prior step, no condition $cond_{h'}(d_{h-h'+1}, d_{h-h'+2}, \ldots, d_h)$ with $1 \leq h' \leq h$ holds, where $cond_0 \triangleq$ false.

Example 2.3:

We consider the Chase algorithm II [2] modified by introducing an early termination condition $cond_h(w_1, \ldots, w_1)$ with $1 \leq h \leq 3$ (abbreviated to $cond_h$), and have analyzed the effectiveness of the termination condition by simulation for $\mathrm{RM}_{5,1}$, $\mathrm{RM}_{5,2}$, $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$. The case that the hard-decision received vector $\boldsymbol{z}$ is a codeword is not accounted for in this example and Examples 4.

In the simulation, test error patterns $\boldsymbol{e}_j$ with $1 \leq j \leq 2^t$, whose nonzero components are confined in the first t most unreliable positions, were generated in the increasing order of $L(\boldsymbol{e}_j)$, where $t \triangleq \lfloor w_1/2 \rfloor$, and the j-th candidate codeword $\boldsymbol{c}_j$ was generated by majority-logic decoding of $\boldsymbol{z} + \boldsymbol{e}_j$ with randomly breaking ties. Some of $\boldsymbol{c}_j$'s were the same.

For $1 \leq h \leq 3$, if i is the smallest index such that condition $\Delta_h(w_1, \ldots, w_1)$ (abbreviated to Δ_h) holds for $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_i\}$, then the reduction rate $r_{h,i}$ of iterations is defined as $(2^t - i)/(2^t - 1)$. If there is no such an index i, let $r_{h,2^t} \triangleq 0$. For

$1 \leq j \leq 2^t$, let $\#\Delta_{h,j}$ denote the number of occurrences of event that Δ_h holds at the j-th iteration, let $\#\Delta_h$ denote the occurrence number of events that condition Δ_h holds, and let $\#MLD$ denote the number of occurrences of event that the decoded codeword by the Chase algorithm II is optimal. Figrure 2.1 shows the ratio $r_h \triangleq (\sum_{j=h}^{2^t} r_{h,j}\#\Delta_{h,j})/(\#MLD - \sum_{j=1}^{h-1}\#\Delta_j)$.

Next, we consider the ratio of the number of the decoded codewords for which condition Δ_h holds to that of decoded codewords which are optimal. Figure 2.2 shows ratios $\rho_h \triangleq \#\Delta_h/(\#MLD - \sum_{i=1}^{h-1}\#\Delta_i)$ for $1 \leq h \leq 3$.

Simulation results show that $cond_2$ is effective in all cases and $cond_3$ is slightly more effective than $cond_2$. The effectiveness of $cond_2$ over $cond_1$ is relatively small for $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$.

Example 2.4: In the decoding algorithm in [6, 7], "minimum-weight sub-trellis search" (MWTS) is introduced to improve Chase algorithm II. At most one MWTS is performed and conditions $cond_1(w_i)$ with $1 \leq i \leq 2$ are effectively employed.

In an iterative decoding algorithm presented in [8, 9], a modified minimum weight sub-trellis search (MWTS_0) around a codeword $\boldsymbol{u}$ is used to find the best codeword of $\{\boldsymbol{v} \in C : d_H(\boldsymbol{v}, \boldsymbol{u}) = w_1\}$. An initial candidate codeword $\boldsymbol{u}_0$ for $\mathrm{RM}_{m,r}$ is generated by majority-logic decoding with randomly breaking ties and unless $cond_1(w_1)$ is true, a MWTS_0 around $\boldsymbol{u}_0$ is performed. Then a MWTS_0 around the latest candidate codeword is iterated until the codewords generated already satisfy (i) a sufficient condition on optimality as an early termination condition or (ii) the final condition that the search results in a previous candidate codeword.

Part of the simulation results [8] are shown in Figures 2.3 and 2.4. Optimality conditions $cond_1(w_1)$, $cond_2(w_2, w_i)$ and $cond_3(w_2, w_2, w_i)$ with $1 \leq i \leq 2$ (abbreviated to $cond'_h$) are used in the simulation. Let $\#\Delta'_h$ denote the number of occurrences of event that $\Delta_h(w_2, \ldots, w_2)$ holds. Figure 2.3 shows ratios $\rho'_h = \#\Delta'_h$ / ($\#MLD$ $- \sum_{i=1}^{h-1}$ $\#\Delta_{i,2}$) with $1 \leq h \leq 3$ for $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$. For $\mathrm{RM}_{6,2}$ (or $\mathrm{RM}_{6,3}$), the average numbers of iterations of MWTS_0 are 1.45, 0.668, 0.188, 0.0253(or 1.49, 0.549, 0.118, 0.0131) at E_b/N_0 = 3,4,5,6 in dB, respectively, when $cond_1(w_1)$ and $cond_1(w_2)$ are used.

Figure 2.4 shows the block error probabilities compared with those of hard-decision majority-logic decoding, the Chase algorithm II, the decoding algorithm in [6, 7] and maximum-likelihood decoding for $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$.

Example 2.5: Reliable estimation of the error probability P_E of maximum likelihood (ML) decoding of a block code by simulation is time-consuming at relatively high SN ratios where P_E is very small and no tight upper bound on P_E is available. A front decoder which has less complexity but lower performance than ML decoder can be used as follows. A slower ML decoder is called only if the decoded codeword by the front decoder satisfies no sufficient condition on the optimality. The block error probabilites of ML decoding shown in Figure 2.4 were evaluated by simulation where the iterative MWTS$_0$ decoding algorithm [8] was used as a front decoder.

2.4 On Tighter Bounds

If the complete distance profile W is available, then tighter bounds can be derived. In place of $V^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h)$, the following set $V_W^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h)$ is to be considered:

$$\begin{aligned} &V_W^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h) \triangleq \\ &\quad \{\boldsymbol{u} \in V^N : \text{ for } 1 \leq i \leq h, \text{ there is } d_i' \in W \text{ such that} \\ &\qquad\qquad d_i' \geq d_i \text{ and } d_H(\boldsymbol{u}, \boldsymbol{u}_i) = d_i'\}. \end{aligned}$$

If $\boldsymbol{u}_i \in C$ for $1 \leq i \leq h$, then $V \subseteq V_W^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h) \subseteq V^N(\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h)$. A preliminary study shows that even for $h = 1$, $\underline{L}[V_W^N(\boldsymbol{u}_1, d_1)]$ is tighter than $\underline{L}[V^N(\boldsymbol{u}_1, d_1)]$ if $d_1 < n(\boldsymbol{u}_1)$.

2.5 Conclusion

In this chapter, two sufficient conditions on optimality of a candidate codeword for a given received sequence are derived. These sufficient conditions are stronger than all the previously known sufficient conditions. To show their effectiveness, these sufficient conditions were applied to the Chase decoding algorithm II and a newly proposed iterative decoding algorithm. Simulation results show that they are very effective in terminating the decoding process except for high block error probability region. For low SNR, sufficient conditions based on four or more already generated candidate codewords are under study.

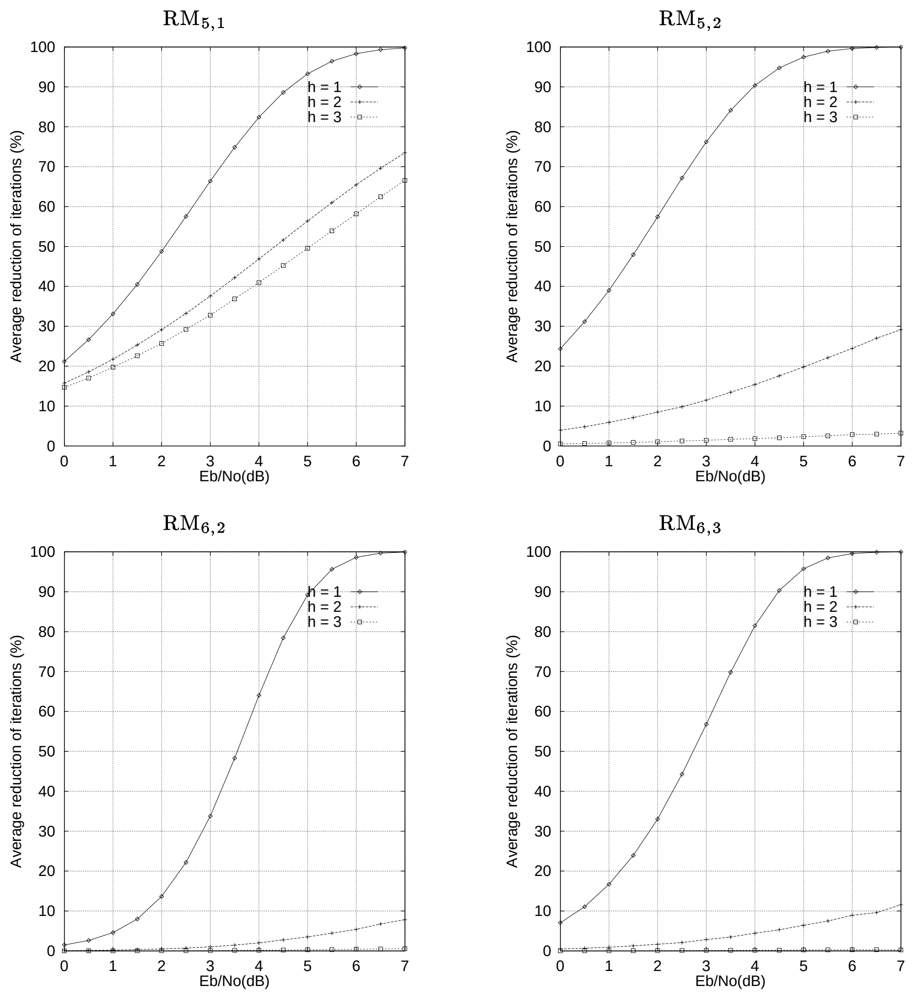

Figure 2.1: Simulation results on the average of reduction rates of iterations r_1, r_2, r_3 for $\mathrm{RM}_{5,1}$, $\mathrm{RM}_{5,2}$, $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$.

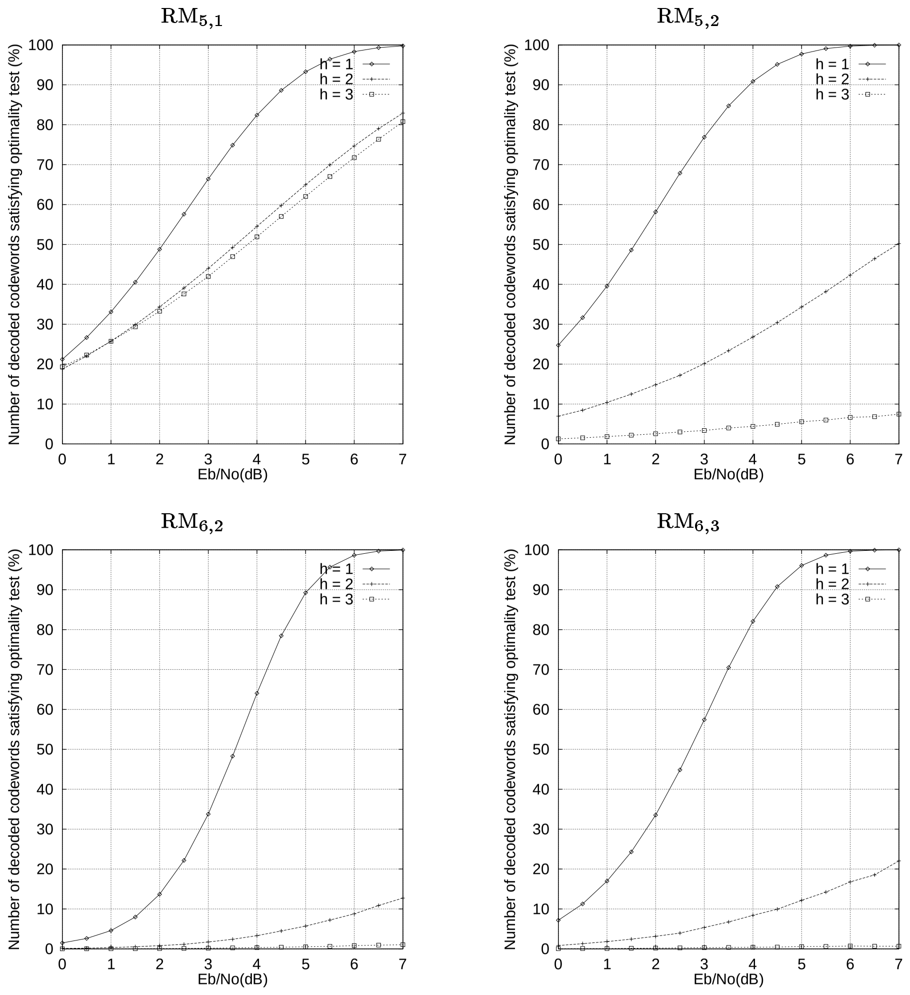

Figure 2.2: Simulation results on the ratios ρ_h of the number of those codewords decoded by Chase decoding algorithm II which satisfy optimality test $cond_h$ to that of decoded codewords which are optimal for RM$_{5,1}$, RM$_{5,2}$, RM$_{6,2}$ and RM$_{6,3}$.

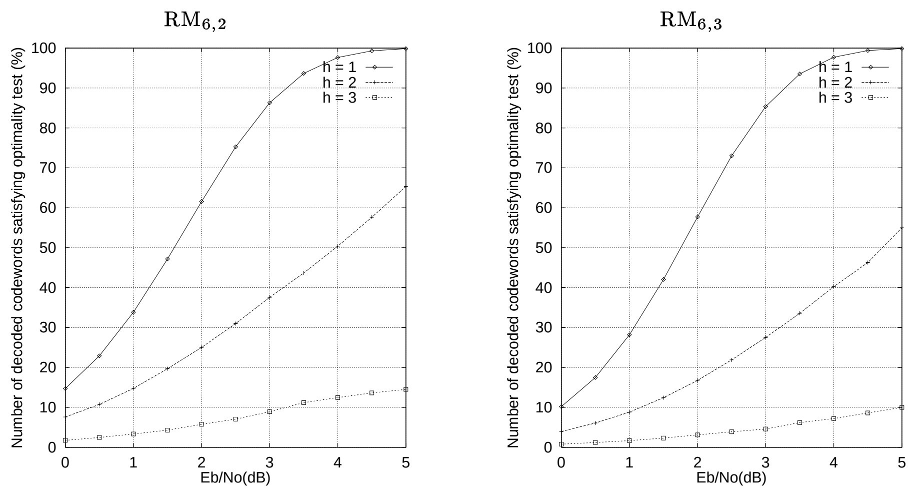

Figure 2.3: Simulation results on the ratios ρ'_h of the number of the decoded codewords which satisfy optimality test $cond'_h$ to that of optimal decoded codewords for the iterative MWTS_0 decoding of $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$.

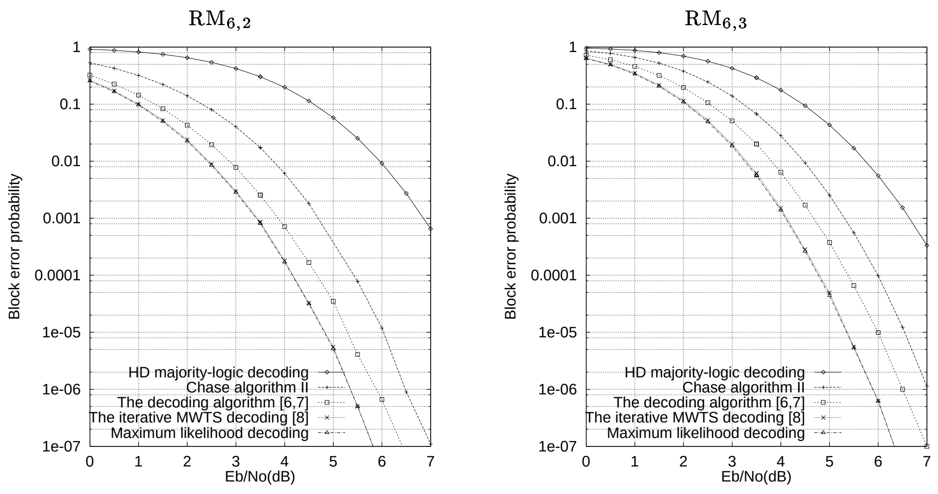

Figure 2.4: Block error probabilities for $\mathrm{RM}_{6,2}$ and $\mathrm{RM}_{6,3}$.

Chapter 3

A Sufficient Condition for Ruling Out Some Useless Test Error Patterns in Iterative Decoding Algorithms

3.1 Introduction

In such an iterative decoding algorithm as Chase type-II decoding algorithm[2], candidate codewords for a received vector are generated by using an algebraic decoder and a set of test error patterns that is formed based on the reliability measures of the received symbols. The candidate codeword that has the largest correlation metric with the received vector is chosen as the decoded codeword. This decoding is a bounded-distance decoding and achieves asymptotic error performance. However, during the decoding process, some of the test error patterns may result in decoding failures if a bounded-distance algebraic decoding is used and some may produce the same candidate codeword (repetition). These result in unnecessary decoding operations and hence prolongs the decoding delay. Therefore, it is desirable to remove these useless or redundant error patterns before or during the decoding process.

Recently, several iterative decoding algorithms have been proposed [20, 6, 6, 19, 5, 4, 13]. In these algorithms, a sufficient condition on optimality is applied to each candidate codeword when it is generated. If the sufficient condition is satisfied, the tested candidate codeword is the most likely codeword for the received vector and the decoding process stops. Otherwise, another candidate codeword is generated by using a new test error pattern. This process repeats until either the most likely codeword is found or the test error patterns are exhausted. In some iterative decoding algorithms[20, 6, 6, 19], rules are provided to remove those error patterns that produce the same candidate codeword (not decoding failures). Hence, successful decodings generate only different candidate codewords for test. However, these rules do not guarantee that the candidate codewords are generated in the order of increasing correlation metric. As a result, some candidate codewords may have smaller correlation metrics than those which were generated earlier and hence useless in decoding decision.

To the authors' knowledge, the first study on a sufficient condition for ruling out some useless test error patterns to reduce the number of bounded-distance t decodings was made in [20]. Let us call the sufficient condition present in [20] the K-N-H condition. This paper presents a more effective sufficient condition for ruling out some useless test error pattern than the K-N-H condition. In this paper, the ruling out is based on the following conditions: (1) Under a reasonable restriction on the order of generating test error patterns (EG condition stated in Sec. 3), a necessary and sufficient condition (stated in Lemma 3.1) that the decoded codeword, $dec(\boldsymbol{e})$, by a bounded-distance decoding for a nonzero test error pattern $\boldsymbol{e}$ is different from all candidate codewords that have been generated already, and (2) the Hamming distance between the latest candidate codeword generated already, denoted $\boldsymbol{c}$, and $dec(\boldsymbol{e})$ is at least the minimum distance of the code. If test error patterns are assumed to be generated in binary order and the condition (1) above is only taken into consideration, the sufficient condition presented in this paper is reduced to the K-N-H condition as stated in Sec. 4.1. In this sense, our new condition is an extension and improvement of the K-N-H condition.

The complexity for testing the ruling out condition is considerably smaller than that of a bounded-distance algebraic decoding. Each time when a new test error pattern is generated, it is tested based on this sufficient condition. If the condition holds, then this error pattern can not produce a candidate codeword with a correlation metric larger than those of the candidate codewords generated already and hence it is useless. In this case, the error pattern is ruled out for decoding and the next test error pattern is generated unless the test error patterns have been exhausted. This reduces the number of bounded-distance decoding operations and the decoding delay. We apply the ruling out condition to Chase Type-II decoding algorithm, and the reduction rates of this ruling out condition are evaluated by simulation and compared with those of the early terminating rule based on the Taipale-Pursley sufficient condition on optimality [5] for various Reed-Muller (RM) codes and extended BCH codes of lengths 64 and 128. Results show that the ruling out condition presented in this paper is more effective than the Taipale-Pursley early termination condition.

3.2 Definitions

Suppose a binary block code C of length n with the minimum Hamming distance $d_{\min}$ is used for error control over the AWGN channel using BPSK signaling. Let $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ and $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ be the soft-decision and binary hard-decision received sequences respectively.

Let V^n denote the set of all binary n-tuples. The correlation between $\boldsymbol{u} \in V^n$ and the received sequence $\boldsymbol{r}$ is given by $M(\boldsymbol{u}) = \sum_{j=1}^{n} r_j(2u_j - 1)$. For any $\boldsymbol{u} \in V^n$, $M(\boldsymbol{z}) \geq M(\boldsymbol{u})$. For a set X, let $|X|$ denote the cardinal number of X. For an n-tuple $\boldsymbol{u} \in V^n$, define the following:

$$D_0(\boldsymbol{u}) \triangleq \{j : u_j = z_j, \text{ and } 1 \leq j \leq n\}, \tag{3.2.1}$$

$$D_1(\boldsymbol{u}) \triangleq \{1, 2, \ldots, n\} \backslash D_0(\boldsymbol{u}), \tag{3.2.2}$$

$$n_\alpha(\boldsymbol{u}) \triangleq |D_\alpha(\boldsymbol{u})|, \text{ for } \alpha \in \{0, 1\}, \tag{3.2.3}$$

$$L(\boldsymbol{u}) \triangleq \sum_{j \in D_1(\boldsymbol{u})} |r_j|. \tag{3.2.4}$$

Then $M(\boldsymbol{u})$ can be expressed as follows:

$$M(\boldsymbol{u}) = M(\boldsymbol{z}) - 2L(\boldsymbol{u}), \tag{3.2.5}$$

where $L(\boldsymbol{u})$ is the correlation discrepancy of $\boldsymbol{u}$. For a subset U of V^n, let $\underline{L}[U]$ be defined as

$$\underline{L}[U] \triangleq \min_{\boldsymbol{u} \in U} L(\boldsymbol{u}). \tag{3.2.6}$$

If U is empty, then $\underline{L}[U]$ is defined as ∞(infinity). A candidate codeword $\boldsymbol{c}$ for the received vector $\boldsymbol{r}$ is the optimal MLD solution if and only if $L(\boldsymbol{c}) = \underline{L}[C]$. If $\boldsymbol{z}$ is a codeword, then $\boldsymbol{z}$ is the optimal codeword.

For a given received sequence $\boldsymbol{r}$, a candidate codeword $\boldsymbol{c}$ is said to be better (or more likely) than another candidate codeword $\boldsymbol{c}'$ if $L(\boldsymbol{c}) \leq L(\boldsymbol{c}')$. A candidate codeword $\boldsymbol{c}$ is said to be the best if $L(\boldsymbol{c})$ is the minimum among a specified set of candidate codewords. Let $d_H(\boldsymbol{u}, \boldsymbol{v})$ denote the Hamming distance between two n-tuples, $\boldsymbol{u}$ and $\boldsymbol{v}$, and let $w_H(\boldsymbol{u})$ denote the Hamming weight of $\boldsymbol{u}$.

3.3 Test Error Patterns

In the iterative decoding algorithms presented in [2, 20, 6, 6], the set of test error patterns for a received sequence $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ is formed based on the reliability measures of the received symbols in $\boldsymbol{r}$. The choice of this set determines the error performance and effectiveness of these decoding algorithms. It is obvious that only the most probable error patterns for a given received sequence $\boldsymbol{r}$ should be used as the test error patterns and they should be used in likelihood order to generate the candidate codewords for optimality test.

In the design of test error patterns, there are two parameters, T and t with $1 \leq T \leq n$ and $0 \leq t \leq \lfloor (d_{\min} - 1)/2 \rfloor$. A test error pattern $\boldsymbol{e}$ is a binary n-tuple with nonzero components confined to the T least reliable positions of $\boldsymbol{r}$. For simplicity of discussion, we assume that the bit positions, 1, 2, $\ldots, n$ are ordered according to the reliability order given as follows:

$$|r_i| \leq |r_j|, \text{ for } 1 \leq i < j \leq n. \tag{3.3.1}$$

In this case, the T least reliable positions are simply the first T bit positions. Let E_T denote the set of these error patterns. For each $\boldsymbol{e} \in E_T$, the last $n-T$ components are all zero. In both the Chase type-II algorithm and the iterative decoding algorithm proposed in [6, 6], T is set to $\lfloor d_{\min}/2 \rfloor$.

For a binary n-tuple $\boldsymbol{u} \in V^n$, let $dec(\boldsymbol{u})$ denote the codeword given by bounded-t distance decoding of $\boldsymbol{z} + \boldsymbol{u}$. If a decoding failure occurs, we write "$dec(\boldsymbol{u}) = *$". For a test error pattern $\boldsymbol{e} \in E_T$, let $l(\boldsymbol{e})$ denote the last nonzero bit position. The candidate codeword associated with $\boldsymbol{e}$ is the codeword $dec(\boldsymbol{e})$ given by bounded t-distance decoding of $\boldsymbol{z} + \boldsymbol{e}$. If a decoding failure occurs, the candidate $dec(\boldsymbol{e})$ is not defined. If $dec(\boldsymbol{e}) \neq *$, then

$$w_H(\boldsymbol{z} + \boldsymbol{e} + dec(\boldsymbol{e})) \leq t. \tag{3.3.2}$$

For m positive integers $i_1, i_2, \ldots, i_m$ such that $1 \leq i_1 < i_2 < \cdots < i_m \leq T$, let $\boldsymbol{v}(i_1, i_2, \ldots, i_m)$ denote the test error pattern whose i-th component is one if and only if there is an index $1 \leq j \leq m$ with $i = i_j$. Let $l(\boldsymbol{v}(i_1, i_2, \ldots, i_m))$ denote i_m. For two different test error patterns $\boldsymbol{e} = \boldsymbol{v}(i_1, i_2, \ldots, i_m)$ and $\boldsymbol{e}' = \boldsymbol{v}(i'_1, i'_2, \ldots, i'_{m'})$, we write "$\boldsymbol{e}' <_E \boldsymbol{e}$", if $m' \leq m$ and there is a subset $\{j_1, j_2, \ldots, j_{m'}\}$ of $\{i_1, i_2, \ldots, i_m\}$ such that $j_1 < j_2 < \cdots < j_{m'}$ and $i'_s \leq j_s$ for $1 \leq s \leq m'$.

We assume that test error patterns are generated sequentially from $\mathbf{0}$(the zero tuple) in such a way that the following condition holds.

EG condition: if $\boldsymbol{e} \in E_T$ is generated at a stage, then every test error pattern $\boldsymbol{e}'$ such that $\boldsymbol{e}' <_E \boldsymbol{e}$ has been generated at a preceding stage.

For an example, if test error patterns are generated in binary order or in the increasing order of correlation discrepancy $L(\boldsymbol{e})$, then the above condition is satisfied.

For $\boldsymbol{e} = \boldsymbol{v}(i_1, i_2, \ldots, i_m) \in E_T$, let $U_t[\boldsymbol{e}]$ denote the set of such binary n-tuples $\boldsymbol{u}$'s that $\boldsymbol{z} + \boldsymbol{u}$ satisfies the following two conditions: (i) the nonzero bit positions among the first $l(\boldsymbol{e})$ components are exactly $i_1, i_2, \ldots, i_m$, and (ii) the weight of the last $n - l(\boldsymbol{e})$ components is t. Then we have the following lemma. The if part has been stated in [6]. For the ruling out condition, we need the only-if part to guarantees that there is no degradation of error-performance.

Lemma 3.1: For a nonzero test error pattern $\boldsymbol{e} \in E_T$, $dec(\boldsymbol{e})$ is a new candidate codeword different from all candidate codewords that have been generated already, if and only if $dec(\boldsymbol{e}) \in U_t[\boldsymbol{e}]$.

(Proof) We prove the only-if part. Define $\boldsymbol{c} \triangleq dec(\boldsymbol{e})$. Since a bounded-$t$ distance decoding is used, if $w_H(\boldsymbol{z} + \boldsymbol{c}) \leq t$, then $\boldsymbol{c} = dec(\mathbf{0})$. This contradicts the EG condition. Hence, $w_H(\boldsymbol{z} + \boldsymbol{c}) > t$. Let $\boldsymbol{z} + \boldsymbol{c} = \boldsymbol{v}(i'_1, i'_2, \ldots, i'_p)$, where $p = w_H(\boldsymbol{z} + \boldsymbol{c})$. Define $\boldsymbol{e}' \triangleq \boldsymbol{v}(i'_1, i'_2, \ldots, i'_{m'})$ where $m' = p - t > 0$. Then

$$w_H(\boldsymbol{z} + \boldsymbol{e}' + \boldsymbol{c}) = t. \tag{3.3.3}$$

From the uniqueness of bounded-t distance decoding,

$$\boldsymbol{c} = dec(\boldsymbol{e}'). \tag{3.3.4}$$

Let $\boldsymbol{e} = \boldsymbol{v}(i_1, i_2, \ldots, i_m)$. Define

$$J \triangleq \{i'_1, i'_2, \ldots, i'_p\} \cap \{i_1, i_2, \ldots, i_m\}. \tag{3.3.5}$$

It follows from (3.3.2) that

$$m' = w_H(\boldsymbol{z} + \boldsymbol{c}) - t \leq |J|. \tag{3.3.6}$$

Let $j_1, j_2, \ldots, j_{m'}$ be the m' smallest integers in J. Then, $i'_s \leq j_s$ for $1 \leq s \leq m'$. Hence, $l(\boldsymbol{e}') \leq l(\boldsymbol{e}) \leq T$, that is, $\boldsymbol{e}' \in E_T$ and $\boldsymbol{e}' <_E \boldsymbol{e}$. Since $dec(\boldsymbol{e}') = dec(\boldsymbol{e}) = \boldsymbol{c}$, it follows from the assumption of Lemma that $\boldsymbol{e} = \boldsymbol{e}'$, and therefore $\boldsymbol{c} \in U_t[\boldsymbol{e}]$.

$\triangle\triangle$

3.4 A sufficient condition for ruling out useless error patterns

3.4.1 General Case

For $\boldsymbol{e} \in E_T$ and $\boldsymbol{u}, \boldsymbol{v} \in V^n$, let $d_H^{[1]}(\boldsymbol{u}, \boldsymbol{v})$ and $d_H^{[2]}(\boldsymbol{u}, \boldsymbol{v})$ denote the Hamming distances between $\boldsymbol{u}$ and $\boldsymbol{v}$ in the first $l(\boldsymbol{e})$ components and the last $n - l(\boldsymbol{e})$ components, respectively. For $\boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in V^n$, let $D_1^{[j]}(\boldsymbol{u})$, $D_0^{[j]}(\boldsymbol{u})$, $n^{[j]}(\boldsymbol{u})$ with $j \in \{1, 2\}$ be defined as follows:

$$D_0^{[1]}(\boldsymbol{u}) \triangleq \{i : u_i = z_i, \text{ and } 1 \leq i \leq l(\boldsymbol{e})\}, \tag{3.4.1}$$

$$D_0^{[2]}(\boldsymbol{u}) \triangleq \{i : u_i = z_i, \text{ and } l(\boldsymbol{e}) < i \leq n\}, \tag{3.4.2}$$

$$D_1^{[1]}(\boldsymbol{u}) \triangleq \{1, 2, \ldots, l(\boldsymbol{e})\} \backslash D_0^{[1]}(\boldsymbol{u}), \tag{3.4.3}$$

$$D_1^{[2]}(\boldsymbol{u}) \triangleq \{l(\boldsymbol{e}) + 1, l(\boldsymbol{e}) + 2, \ldots, n\} \backslash D_0^{[2]}(\boldsymbol{u}), \tag{3.4.4}$$

$$n_0^{[1]}(\boldsymbol{u}) \triangleq |D_0^{[1]}(\boldsymbol{u})| = l(\boldsymbol{e}) - n_1^{[1]}(\boldsymbol{u}). \tag{3.4.5}$$

$$n_0^{[2]}(\boldsymbol{u}) \triangleq |D_0^{[2]}(\boldsymbol{u})| = n - l(\boldsymbol{e}) - n_1^{[2]}(\boldsymbol{u}). \tag{3.4.6}$$

$$n_1^{[j]}(\boldsymbol{u}) \triangleq |D_1^{[j]}(\boldsymbol{u})| = d_H^{[j]}(\boldsymbol{u}, \boldsymbol{z}), \tag{3.4.7}$$

For a given codeword $\boldsymbol{c} \in C$, $\boldsymbol{u} \in V^n$, $\alpha \in \{0,1\}$ and $j \in \{1,2\}$, let $q_\alpha^{[j]}(\boldsymbol{u})$ be defined as

$$q_\alpha^{[j]}(\boldsymbol{u}) \triangleq |D_1^{[j]}(\boldsymbol{u}) \cap D_\alpha^{[j]}(\boldsymbol{c})|. \tag{3.4.8}$$

Then we have that

$$0 \leq q_\alpha^{[2]}(\boldsymbol{u}) \leq n_\alpha^{[2]}(\boldsymbol{c}), \text{ for } \alpha \in \{0,1\}, \tag{3.4.9}$$

$$\begin{aligned} d_H^{[2]}(\boldsymbol{u}, \boldsymbol{z}) &= |D_1^{[2]}(\boldsymbol{u})| \\ &= |D_1^{[2]}(\boldsymbol{u}) \cap D_0^{[2]}(\boldsymbol{c})| + |D_1^{[2]}(\boldsymbol{u}) \cap D_1^{[2]}(\boldsymbol{c})| \\ &= q_0^{[2]}(\boldsymbol{u}) + q_1^{[2]}(\boldsymbol{u}), \end{aligned} \tag{3.4.10}$$

$$\begin{aligned} d_H^{[2]}(\boldsymbol{u}, \boldsymbol{c}) &= |D_1^{[2]}(\boldsymbol{u}) \cap D_0^{[2]}(\boldsymbol{c})| + |D_0^{[2]}(\boldsymbol{u}) \cap D_1^{[2]}(\boldsymbol{c})| \\ &= q_0^{[2]}(\boldsymbol{u}) + n_1^{[2]}(\boldsymbol{c}) - q_1^{[2]}(\boldsymbol{u}). \end{aligned} \tag{3.4.11}$$

For $\boldsymbol{u} \in U_t[\boldsymbol{e}]$, it follows from the definition of $U_t[\boldsymbol{e}]$, (3.4.5), (3.4.8), (3.4.10) and (3.4.11) that

$$\begin{aligned} d_H(\boldsymbol{u}, \boldsymbol{c}) &= d_H^{[1]}(\boldsymbol{u}, \boldsymbol{c}) + d_H^{[2]}(\boldsymbol{u}, \boldsymbol{c}) \\ &= d_H^{[1]}(\boldsymbol{z} + \boldsymbol{e}, \boldsymbol{c}) + d_H^{[2]}(\boldsymbol{u}, \boldsymbol{c}) \\ &= d_H^{[1]}(\boldsymbol{e}, \boldsymbol{z} + \boldsymbol{c}) + n_1^{[2]}(\boldsymbol{c}) + q_0^{[2]}(\boldsymbol{u}) - q_1^{[2]}(\boldsymbol{u}) \\ &= d_H(\boldsymbol{e}, \boldsymbol{z} + \boldsymbol{c}) + q_0^{[2]}(\boldsymbol{u}) - q_1^{[2]}(\boldsymbol{u}), \end{aligned} \tag{3.4.12}$$

$$d_H^{[2]}(\boldsymbol{u}, \boldsymbol{z}) = q_0^{[2]}(\boldsymbol{u}) + q_1^{[2]}(\boldsymbol{u}) = t. \tag{3.4.13}$$

For $\boldsymbol{e} \in E_T$ and $\boldsymbol{c} \in C$, define the following set of n-tuples in V^n:

$$V_t(\boldsymbol{e}, \boldsymbol{c}) \triangleq \{\boldsymbol{u} \in U_t[\boldsymbol{e}] : d_H(\boldsymbol{u}, \boldsymbol{c}) \geq d_{\min}\}. \tag{3.4.14}$$

Then, it follows from (3.2.6), Lemma 3.1 and (3.4.14), that we have Theorem 1.

Theorem 1: Suppose that $\boldsymbol{c}$ is the latest candidate codeword generated, $\boldsymbol{c}_{\text{best}}$ is the best among those candidate codewords already generated, the order of generation of test error patterns satisfies the condition EG and the next test error pattern is $\boldsymbol{e}$. Then $\boldsymbol{e}$ is useless and can be ruled out if the following condition holds:

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{L}[V_t(\boldsymbol{e}, \boldsymbol{c})]. \tag{3.4.15}$$

△△

Theorem 1 gives a sufficient condition to rule out useless test error patterns.

The complexity for testing the above condition is to be considerably smaller than that for bounded-t decoding. Evaluation of the bound of (3.4.15) is given below.

It follows from (3.4.9) and (3.4.12) to (3.4.14) that for $\boldsymbol{u} \in U_t[\boldsymbol{e}]$, $\boldsymbol{u} \in V_t(\boldsymbol{e}, \boldsymbol{c})$ if and only if

$$q_0^{[2]}(\boldsymbol{u}) + q_1^{[2]}(\boldsymbol{u}) = t, \tag{3.4.16}$$

$$q_0^{[2]}(\boldsymbol{u}) - q_1^{[2]}(\boldsymbol{u}) \geq \delta, \tag{3.4.17}$$

$$0 \leq q_\alpha^{[2]}(\boldsymbol{u}) \leq n_\alpha^{[2]}(\boldsymbol{c}), \text{ for } \alpha \in \{0, 1\}, \tag{3.4.18}$$

where

$$\delta = d_{\min} - d_H(\boldsymbol{e}, \boldsymbol{z} + \boldsymbol{c}). \tag{3.4.19}$$

Conversely, for a pair of nonnegative integers q_0 and q_1 such that

$$q_0 + q_1 = t, \tag{3.4.20}$$

$$q_0 - q_1 \geq \delta, \tag{3.4.21}$$

$$n_\alpha^{[2]}(\boldsymbol{c}) \geq q_\alpha \geq 0, \text{ for } \alpha \in \{0, 1\}, \tag{3.4.22}$$

there is $\boldsymbol{u} \in V_t(\boldsymbol{e}, \boldsymbol{c})$ such that

$$q_\alpha^{[2]}(\boldsymbol{u}) = q_\alpha, \text{ for } \alpha \in \{0, 1\}. \tag{3.4.23}$$

Let Q denote the set of pairs of nonnegative integers (q_0, q_1) satisfying (3.4.20) to (3.4.22). The following lemma summarizes the above result.

Lemma 3.2: For $\boldsymbol{u} \in U_t[\boldsymbol{e}]$, $\boldsymbol{u} \in V_t(\boldsymbol{e}, \boldsymbol{c})$ if and only if there is a pair (q_0, q_1) in Q such that $q_\alpha^{[2]}(\boldsymbol{u}) (\triangleq |D_1^{[2]}(\boldsymbol{u}) \cap D_\alpha^{[2]}(\boldsymbol{c})|) = q_\alpha$ for $\alpha \in \{0, 1\}$.

△△

In [20], test error patterns are assumed to be generated in binary order, and the K-N-H condition can be readily expressed in the notations in this paper as

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{L}(U_t[\boldsymbol{e}]). \tag{3.4.24}$$

From Lemma 3.2, we can see the difference of effectiveness between (3.4.15) and (3.4.24).

From (3.4.5), (3.4.9) and (3.4.20) to (3.4.23), we have that

$$\overline{\delta} \geq q_1 \geq \underline{\delta}, \tag{3.4.25}$$

$$\underline{\delta} = \max\{0, t + l(\boldsymbol{e}) + n_1^{[2]}(\boldsymbol{c}) - n\}, \tag{3.4.26}$$

$$\overline{\delta} = \min\{n_1^{[2]}(\boldsymbol{c}), \lfloor (t-\delta)/2 \rfloor, t\}. \tag{3.4.27}$$

Since the Hamming distances between $\boldsymbol{e}$ and $\boldsymbol{z} + \boldsymbol{c}$ in the first T components and the last $n - T$ components are at most T and t respectively,

$$d_H(\boldsymbol{e}, \boldsymbol{z} + \boldsymbol{c}) \leq T + t. \tag{3.4.28}$$

3.4.2 Special Case

Consider the special case where $T = \lfloor d_{\min}/2 \rfloor$ and $t = \lfloor (d_{\min} - 1)/2 \rfloor$. The Chase algorithm II[2] is this case. From (3.4.19) and (3.4.28),

$$\delta > 0, \tag{3.4.29}$$

and therefore,

$$\overline{\delta} = \min\{n_1^{[2]}(\boldsymbol{c}), \lfloor (t-\delta)/2 \rfloor\}. \tag{3.4.30}$$

Suppose that $d_{\min} \leq n/2$. Then since $l(\boldsymbol{e}) \leq T$ and $n_1^{[2]}(\boldsymbol{c}) \leq d_H(\boldsymbol{c}, \boldsymbol{z}) \leq T + t \leq d_{\min}$,

$$\underline{\delta} = 0. \tag{3.4.31}$$

Consequently, Q is not empty if and only if

$$t \geq \delta. \tag{3.4.32}$$

Assume that (3.4.32) holds. Then we have

$$\begin{aligned} Q = \{(t - q_1, q_1) \quad & : 0 \leq q_1 \leq \overline{\delta} = \\ & \min\{n_1^{[2]}(\boldsymbol{c}), \lfloor (t-\delta)/2 \rfloor\}\}. \end{aligned} \tag{3.4.33}$$

For a subset X of $\{1, 2, \ldots, n\}$ and a positive integer $j \leq |X|$, let $X^{(j)}$ denote the set of j smallest integers in X. For a non-positive integer j, $X^{(j)} \triangleq \phi$ and for $j \geq |X|$, $X^{(j)} \triangleq X$.

Suppose that $T = \lfloor d_{\min}/2 \rfloor$, $t = \lfloor (d_{\min} - 1)/2 \rfloor$ and $d_{\min} \leq n/2$. Now we evaluate $\underline{L}(V_t(\boldsymbol{e}, \boldsymbol{c}))$. Let $\boldsymbol{u} \in V_t(\boldsymbol{e}, \boldsymbol{c})$. Since $\boldsymbol{u} \in U_t[\boldsymbol{e}]$, it follows from (3.2.4) that

$$\begin{aligned} L(\boldsymbol{u}) &= L(\boldsymbol{e}) + \sum_{i \in D_1^{[2]}(\boldsymbol{u})} |r_i| \\ &= L(\boldsymbol{e}) + \sum_{i \in D_1^{[2]}(\boldsymbol{u}) \cap D_0^{[2]}(\boldsymbol{c})} |r_i| \\ &\quad + \sum_{i \in D_1^{[2]}(\boldsymbol{u}) \cap D_1^{[2]}(\boldsymbol{c})} |r_i|. \end{aligned} \tag{3.4.34}$$

From Lemma 3.2, (3.2.6), (3.4.33), (3.4.34) and the definition of $X^{(j)}$, we have that

$$\begin{aligned} &\underline{L}(V_t(\boldsymbol{e}, \boldsymbol{c})) = \\ &L(\boldsymbol{e}) + \min_{0 \leq q_1 \leq \bar{\delta}} \left\{ \sum_{i \in [D_0^{[2]}(\boldsymbol{c})]^{(t-q_1)}} |r_i| \right. \\ &\qquad \left. + \sum_{i \in [D_1^{[2]}(\boldsymbol{c})]^{(q_1)}} |r_i| \right\} \\ &\qquad = \sum_{j=1}^{m} |r_{i_j}| + \sum_{i \in D(\boldsymbol{c})} |r_i|, \end{aligned} \tag{3.4.35}$$

where $\boldsymbol{e} = \boldsymbol{v}(i_1, i_2, \ldots, i_m)$ and

$$D(\boldsymbol{c}) \triangleq (D_0^{[2]}(\boldsymbol{c}) \cup [D_1^{[2]}(\boldsymbol{c})]^{(\bar{\delta})})^{(t)}. \tag{3.4.36}$$

From (3.4.35) and (3.4.36), we see that the evaluation of $\underline{L}(V_t(\boldsymbol{e}, \boldsymbol{c}))$ is quite simple.

3.5 Application

In the following, we consider the effectiveness of the sufficient condition given in Theorem 1 for ruling out useless test error patterns.

Consider the following modified Chase algorithm-II$_e$. In algorithm-II$_e$, each time a nonzero new test pattern $\boldsymbol{e}$ is generated in an order satisfying the EG condition stated in Section 3 (unless no candidate codeword has been generated yet), the following condition is tested:

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{L}[V_t(\boldsymbol{e}, \boldsymbol{c})], \tag{3.5.1}$$

where $\boldsymbol{c}$ is the latest candidate codeword and $\boldsymbol{c}_{\text{best}}$ is the best among all candidate codewords that have been generated already. If (3.5.1) holds, then the bounded-t algebraic decoding of $\boldsymbol{z} + \boldsymbol{e}$ is skipped and the next test error pattern is generated unless test error patterns have been exhausted. Otherwise, the bounded-t decoding is applied to $\boldsymbol{z} + \boldsymbol{e}$. For $\boldsymbol{z} \notin C$, let $N_e(\boldsymbol{z})$ be the number of bounded-t decodings performed for $\boldsymbol{z}$ in algorithm-II$_e$. Then $1 \leq N_e(\boldsymbol{z}) \leq 2^T$. Let us define a reduction rate of the number of bounded-t decodings in algorithm-II$_e$, denoted $r_d(\text{II}_e)$, as the average of $100 \times (1 - ((N_e(\boldsymbol{z}) - 1)/(2^T - 1)))$ over the random vector $\boldsymbol{z} \notin C$ when test error patterns $\boldsymbol{e}$'s are generated in the increasing order of $L(\boldsymbol{e})$.

For comparison, we consider the following modified Chase algorithm-II$_s$. Algorithm-II$_s$ is the Chase algorithm-II modified by introducing an early termination condition:

$$L(\boldsymbol{c}) \leq \sum_{i \in D_0(\boldsymbol{c})^{(\delta')}} |r_i|, \tag{3.5.2}$$

where $\boldsymbol{c}$ is the latest candidate codeword, and $\delta' \triangleq d_{\min} - n_1(\boldsymbol{c})$, $D_0(\boldsymbol{c})$ and $n_1(\boldsymbol{c})$ are defined by (3.2.1) and (3.2.3) respectively. The condition (3.5.2) is a sufficient condition on optimality of a candidate codeword $\boldsymbol{c}$ due to Taipale and Pursley[4]. In algorithm-II$_s$, the ruling out condition of (3.5.1) is not used. Let $r_d(\text{II}_s)$ be the average of $100 \times (1 - ((N_s(\boldsymbol{z}) - 1)/(2^T - 1)))$ where $N_s(\boldsymbol{z})$ denotes the number of bounded-t decodings performed for $\boldsymbol{z}$ in algorithm-II$_s$.

Let $r_d(\text{II}_e)$ and $r_d(\text{II}_s)$ denote the reduction rates of the number of bounded-t decodings in algorithm-II$_e$ and algorithm-II$_s$ respectively. Let RM$_{m,r}$ and EBCH(n, k) denote an r-th order RM code of length 2^m and an extended (n, k) BCH code of length n and dimension k respectively. Figures 3.1 to 3.4 show the values of $r_d(\text{II}_e)$ and $r_d(\text{II}_s)$ evaluated by simulation for RM$_{6,2}$, RM$_{7,3}$, EBCH(64,24) and EBCH(128,64) codes respectively. We see that the ruling out condition of (3.5.1) is more effective than the early termination conditions (3.5.2) for these example codes.

3.6 Conclusion

We have derived a condition to rule out useless test error patterns in the generation of candidate codewords in a Chase-type decoding algorithm. This rule-out condition reduces many unnecessary decoding iterations and computations.

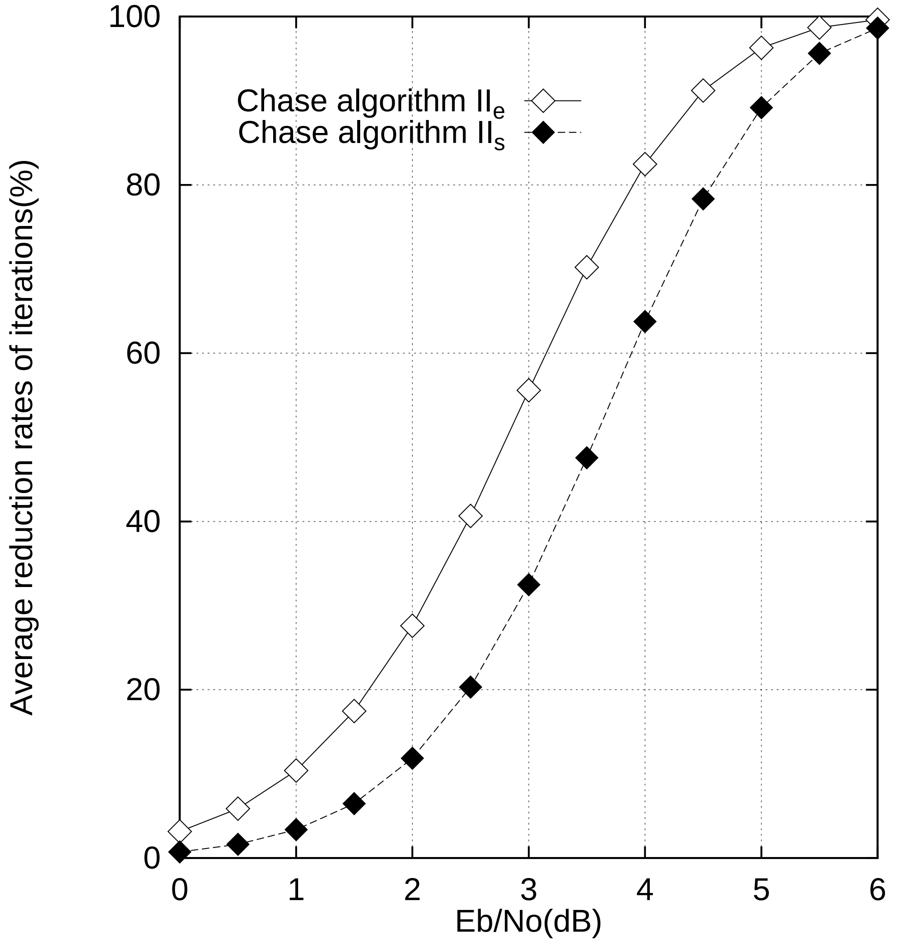

Figure 3.1: Average reduction rates $r_d(\mathrm{II}_e)$ and $r_d(\mathrm{II}_s)$ of iterations for $\mathrm{RM}_{6,2}$

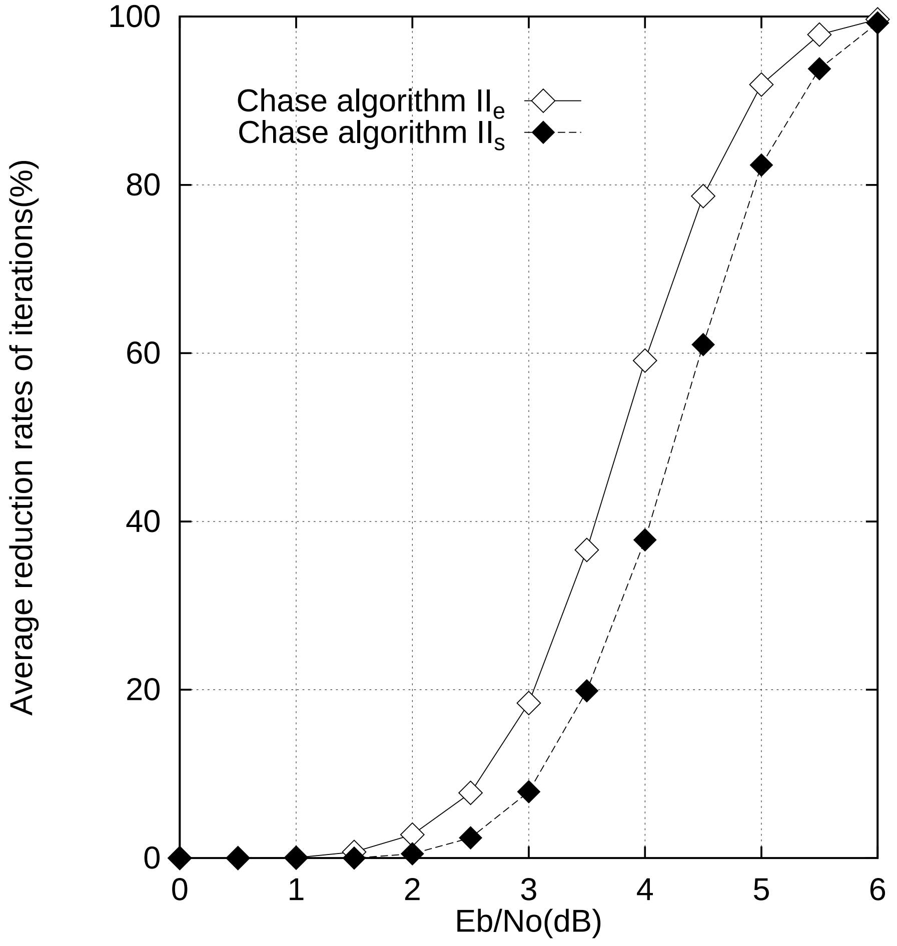

Figure 3.2: Average reduction rates $r_d(\mathrm{II}_e)$ and $r_d(\mathrm{II}_s)$ of iterations for $\mathrm{RM}_{7,3}$

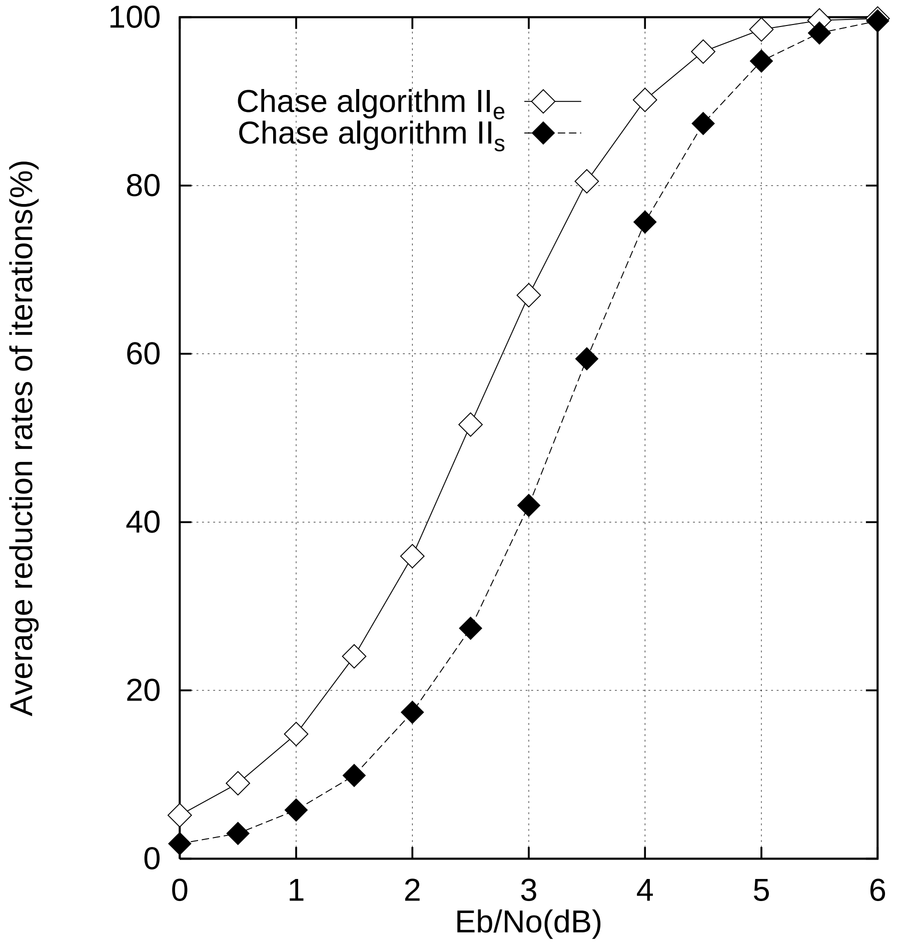

Figure 3.3: Average reduction rates $r_d(\mathrm{II}_e)$ and $r_d(\mathrm{II}_s)$ of iterations for EBCH$(64, 24)$

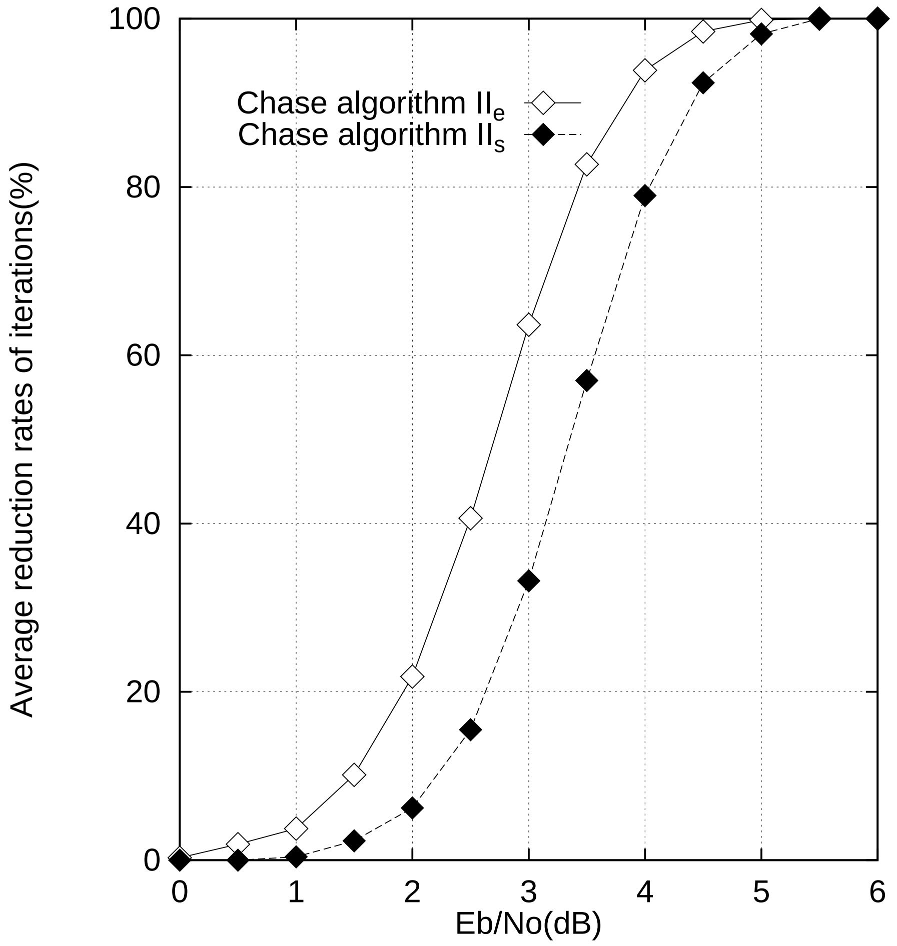

Figure 3.4: Average reduction rates $r_d(\mathrm{II}_e)$ and $r_d(\mathrm{II}_s)$ of iterations for EBCH$(128, 64)$

Chapter 4

A Low-Weight Trellis Based Iterative Soft-Decision Decoding Algorithm

4.1 Introduction

The application of trellis-based maximum likelihood decoding(MLD) algorithms is limited due to the prohibitively large trellises for codes of long block lengths. To overcome the state and branch complexity problems of large trellises for long block codes, several new approaches have been proposed [21]–[16]. Most recently, Moorthy, Lin and Kasami have shown that the minimum-weight subtrellis of a code is sparsely connected and has much simpler state and branch complexities than the full code trellis[27]. Based on this fact, they proposed a minimum-weight subtrellis-based iterative decoding algorithm for linear block codes to achieve suboptimum error performance with a drastic reduction in decoding complexity compared with a trellis-based MLD algorithm, using a full code trellis. The Moorthy-Lin-Kasami(MLK) algorithm is devised based on the following: (1) generation of a sequence of candidate codewords based on a set of test error patterns using the Chase algorithm-II[2] and an algebraic decoder; (2) two test conditions, one to test the optimality of a candidate codeword and the other to test whether the most likely(ML) codeword is at a distance no greater than the minimum distance from the tested candidate codeword; and (3) a minimum weight trellis search to find the ML codeword. The

MLK decoding algorithm is simple indeed and provides good error performance with large reduction in decoding complexity. However, for long codes, there is a significant performance degradation compared to MLD for low to medium SNR and it has several major shortcomings. First, the algebraic decoder used in the MLK algorithm may fail to decode. Second, some test error patterns may result in the same candidate codeword and hence useless decoding iterations unless some preprocessing is done to rule out the repetitions. Third, there is no guarantee that the candidate codewords are generated in the order of increasing improvement in terms of the correlation metric. Fourth, the sufficient conditions for optimality and the nearest neighbor tests are derived based on only the current candidate codeword, and the information of previously tested candidate codewords is discarded. This discarded information may help to narrow down the search of the ML codeword and reduce the possibility that it slips through the tests without being detected. Finally, the performance degradation is big for codes whose minimum weight codewords do not span the codes.

In this chapter, we present a new low-weight subtrellis based iterative decoding algorithm which overcomes the major shortcomings of the MLK algorithm as described above. By a low weight subtrellis, we mean a subtrellis of the code trellis that consists of only codewords of low weights, say minimum and next to the minimum weights. The new algorithm is different from the MLK algorithm in the generation of candidate codewords, optimality test, and termination of the decoding process. It has the following important features. The initial candidate codeword is first generated by a simple decoding method that guarantees a successful decoding, such as the zero-th or the first-order decoding based on the ordered statistics of the received symbols proposed in [17] or combined with an algebraic decoding, called hybrid method. These decodings are very simple and always produce a decoded codeword (no decoding failure). Subsequent candidate codewords, if needed, are generated by a chain of low weight subtrellis searches. Each such search is centered around the current candidate codeword. The current candidate codeword is excluded in the search to prevent repetition and to reduce the possibility of being trapped into a local optimum. The codeword with the largest correlation metric resulting from this low weight subtrellis search is then used as the next candidate codeword. Candidate codewords are generated in the order of improving correlation metrics.

In the new algorithm, three conditions are used to stop the decoding process. The first condition, called the termination condition, is used to test whether further generation of candidate codewords will improve the correlation metric and it also prevents repetition. The other two conditions are used to test the optimality of a candidate codeword. When a candidate codeword other than the initial candidate codeword is generated, the termination condition is tested. If the condition holds, the decoder outputs either the current candidate codeword or the past candidate codeword just before the current one whichever has larger correlation metric. Otherwise, the two sufficient conditions on optimality are tested. If any of the two optimality conditions is satisfied, then the ML codeword is either the current candidate codeword or the past candidate codeword just before the current one. If none of the optimality conditions is satisfied, a new candidate codeword is generated. This test based on the current and past information and use of more than one sufficient condition on optimality reduces the number of decoding iterations. Furthermore, the two sufficient conditions on optimality are derived from the current and the previous candidate codewords, and they are less stringent than the conditions given in [5] and [4]. The only case that the decoder may not output the ML codeword is when the decoding process is terminated by the termination condition.

For codes that are spanned by their minimum weight codewords, the proposed decoding algorithm is very effective. However, for codes that are not spanned by their minimum weight codewords, minimum weight (or next to the minimum weight) subtrellis search does not provide a big enough search space and results in a significant degradation in error performance compared with MLD. To overcome this problem, we use a divide-and-conquer technique to partition the code space into cosets with respect to a subcode which is spanned by the minimum weight codewords. Then the low weight subtrellis search is performed over the cosets in the partition, one at a time, based on a likely order. The search is shifted from one coset to another until the ML codeword is found or all the cosets are exhausted. This results in a good coverage of the entire code space and good error performance, while maintains low weight subtrellis searches to keep the decoding complexity down.

Simulation results show that this new decoding algorithm provides a significant improvement in error performance over the MLK algorithm. For all the Reed-Muller(RM) codes of length 64, extended (64,24,16) and (64,45,8) BCH(EBCH) codes, and the extended (48,24,12) quadratic residue(EQR) code (in which the optimal bit order given in [28] is used), the new algorithm with only minimum weight subtrellis searches provides practically optimal error performance, i.e., the error performance curves of these codes based on the new decoding algorithm fall on top of their respective MLD error performance curves. For the (128,29,32) RM code, the new algorithm only gives a performance degradation less than 0.3 dB for bit-error-rates(BER) from 10^{-5} to 10^{-1} (or SNR over the range from 0 to 4 dB), while the MLK algorithm gives a performance degradation of 1.6 dB. Furthermore, for all these codes, the new decoding algorithm requires much less computational complexity than the Viterbi [21] and RMLD [16] algorithms based on full code trellises. Both the new algorithm and the MLK algorithm require about the same order of average computational complexity.

4.2 Preliminaries

Suppose a binary (N, K, d_H) linear block code C with minimum Hamming distance d_H is used for error control over the AWGN channel using BPSK signaling. Let $W = \{w_0 = 0, w_1, w_2, \ldots, w_m\}$ be the **weight profile** of C, where $w_1 = d_H$ is the minimum weight and $w_1 < w_2 < \cdots < w_m \leq N$. Suppose $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ is the received sequence at the output of the matched filter of the receiver. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_N)$ be the binary hard-decision sequence obtained from $\boldsymbol{r}$ with $z_i = 1$ for $r_i > 0$ and $z_i = 0$ otherwise for $1 \leq i \leq N$.

Let V_N denote the vector space of all the binary N-tuples. For a binary N-tuple $\boldsymbol{u} = (u_1, u_2, \ldots, u_N) \in V_N$, the correlation between $\boldsymbol{u}$ and $\boldsymbol{r}$ is given by

$$M(\boldsymbol{u}) \triangleq \sum_{i=1}^{N} r_i(2u_i - 1). \tag{4.2.1}$$

It is easy to see that $M(\boldsymbol{z}) = \sum_{i=1}^{N} |r_i|$ and for any $\boldsymbol{u} \in V_N$, $M(\boldsymbol{z}) \geq M(\boldsymbol{u})$. For an N-tuple $\boldsymbol{u} \in V_N$, define the following index set:

$$D_1(\boldsymbol{u}) \triangleq \{i : u_i \neq z_i \text{ and } 1 \leq i \leq N\}. \tag{4.2.2}$$

Then $M(\boldsymbol{u})$ can be expressed in terms of $M(\boldsymbol{z})$ and $D_1(\boldsymbol{u})$ as follows:

$$M(\boldsymbol{u}) = M(\boldsymbol{z}) - 2L(\boldsymbol{u}), \tag{4.2.3}$$

where

$$L(\boldsymbol{u}) = \sum_{i \in D_1(\boldsymbol{u})} |r_i|, \tag{4.2.4}$$

which is called the **correlation discrepancy** of $\boldsymbol{u}$ with respect to $\boldsymbol{z}$. Then MLD can be stated as follows: Find the codeword $\boldsymbol{c}_{\text{opt}} \in C$ for which $L(\boldsymbol{c}_{\text{opt}}) = \min_{\boldsymbol{c} \in C} L(\boldsymbol{c})$. The codeword $\boldsymbol{c}_{\text{opt}}$ is then the **most likely**(ML) codeword.

For $\boldsymbol{u}$ and $\boldsymbol{v}$ in V_N, $\boldsymbol{u}$ is said to be better than $\boldsymbol{v}$ if $L(\boldsymbol{u}) < L(\boldsymbol{v})$. Let $\mathbf{better}(\boldsymbol{u}, \boldsymbol{v})$ denote the better one of $\boldsymbol{u}$ and $\boldsymbol{v}$. For a nonempty subset U of V_N, define

$$\boldsymbol{L}[U] \triangleq \min_{\boldsymbol{u} \in U} L(\boldsymbol{u}). \tag{4.2.5}$$

For convenience, $\boldsymbol{L}[\phi] \triangleq \infty$, where ϕ denotes the empty set. It is clear that for $\boldsymbol{c} \in C$, $\boldsymbol{c} = \boldsymbol{c}_{\text{opt}}$ if and only if $L(\boldsymbol{c}) = \boldsymbol{L}[C]$. If $\boldsymbol{z} \in C$, then it follows from (4.2.3) that $\boldsymbol{z} = \boldsymbol{c}_{\text{opt}}$.

4.2.1 A Sufficient Condition for Optimality

A key element in the proposed decoding algorithm is a set of sufficient conditions for testing the optimality of a candidate codeword. In the following, we give a general sufficient condition for optimality in terms of correlation discrepancy and a set of reference codewords.

For $\boldsymbol{u} \in V_N$ and a positive integer d, define the following region:

$$V_N(\boldsymbol{u}, d) \triangleq \{\boldsymbol{v} \in V_N : d(\boldsymbol{v}, \boldsymbol{u}) \geq d\}, \tag{4.2.6}$$

where $d(\boldsymbol{v}, \boldsymbol{u})$ denote the Hamming distance between $\boldsymbol{v}$ and $\boldsymbol{u}$. For $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_h \in V_N$ and positive integers $d_1, d_2, \ldots, d_h$, define

$$\begin{aligned} \underline{\boldsymbol{L}}[\boldsymbol{u}_1, d_1; \boldsymbol{u}_2, d_2; \ldots; \boldsymbol{u}_h, d_h] &\triangleq \min\{L(\boldsymbol{u}) : \boldsymbol{u} \in V_N \text{ and } d(\boldsymbol{u}, \boldsymbol{u}_i) \geq d_i \text{ for } 1 \leq i \leq h\} && (4.2.7) \\ &= \boldsymbol{L}[\cap_{i=1}^{h} V_N(\boldsymbol{u}_i, d_i)], && (4.2.8) \end{aligned}$$

where $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_h$ are called reference words.

Let Q be a subset of C. Define $\widetilde{V_N}(\boldsymbol{u}, d) \triangleq V_N \backslash V_N(\boldsymbol{u}, d) = \{\boldsymbol{v} \in V_N : d(\boldsymbol{v}, \boldsymbol{u}) < d\}$. For a codeword $\boldsymbol{c}_i$ in Q and a positive integer d_i with $1 \leq i \leq h$, define $S_i \triangleq Q \cap \widetilde{V_N}(\boldsymbol{c}_i, d_i)$. Then,

$$Q \backslash S_i \subseteq V_N(\boldsymbol{c}_i, d_i). \tag{4.2.9}$$

S_i is called a search region with center $\boldsymbol{c}_i$ and covering distance d_i from $\boldsymbol{c}_i$. Then $Q\backslash \cup_{i=1}^{h} S_i \subseteq \cap_{i=1}^{h} V_N(\boldsymbol{c}_i, d_i)$. It follows from (4.2.5) that

$$\boldsymbol{L}[\cap_{i=1}^{h} V_N(\boldsymbol{c}_i, d_i)] \leq \boldsymbol{L}[Q\backslash \cup_{i=1}^{h} S_i]. \tag{4.2.10}$$

For a codeword $\boldsymbol{c} \in Q$, suppose

$$L(\boldsymbol{c}) \leq \boldsymbol{L}[\cup_{i=1}^{h} S_i], \tag{4.2.11}$$

and

$$L(\boldsymbol{c}) \leq \underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1; \boldsymbol{c}_2, d_2; \ldots; \boldsymbol{c}_h, d_h]. \tag{4.2.12}$$

Then it follows from (4.2.8), (4.2.10) to (4.2.12) that

$$L(\boldsymbol{c}) \leq \boldsymbol{L}[Q]. \tag{4.2.13}$$

This says that the joint condition of (4.2.11) and (4.2.12) is sufficient for the codeword $\boldsymbol{c}$ to be the **best** codeword (or the codeword with the least correlation discrepancy) in Q.

Consider the case of $Q = C$. Let $\boldsymbol{c}$ be the best codeword found in the region $\cup_{i=1}^{h} S_i$. If $L(\boldsymbol{c})$ satisfies the condition of (4.2.12), then $\boldsymbol{c}$ is the most likely codeword $\boldsymbol{c}_{\text{opt}}$ with respect to the received sequence $\boldsymbol{r}$. The choices of h, the reference codewords, the search regions and the covering distances are crucial in designing the decoding algorithm to be presented in the next section.

Expressions for evaluating $\underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1; \boldsymbol{c}_2, d_2; \ldots; \boldsymbol{c}_h, d_h]$ have been derived in [28] for $h = 1, 2$ and 3 and are given in Appendix A.

4.2.2 Low Weight Subtrellis Search

Let w_k be the k-th nonzero weight in the weight profile W of C. Let $C_{w_k}(\mathbf{0})$ denote the subcode of C that consists of all the nonzero codewords with weights up to and including w_k. The codewords in $C_{w_k}(\mathbf{0})$ form a subtrellis diagram of the full trellis T of C. This subtrellis diagram is called the w_k-weight subtrellis centered around the all-zero codeword and is denoted $T_{w_k}(\mathbf{0})$. $T_{w_k}(\mathbf{0})$ can be obtained by purging the full code trellis T [27] or by direct construction [29]. If $k = 1$, $C_{w_1}(\mathbf{0})$ consists of all the minimum-weight codewords of C and $T_{w_1}(\mathbf{0})$ is called the minimum-weight subtrellis centered around $\mathbf{0}$. For any $\boldsymbol{c} \in C$, define the following subcode of C:

$$C_{w_k}(\boldsymbol{c}) \triangleq \{\boldsymbol{c} + \boldsymbol{v} : \boldsymbol{v} \in C_{w_k}(\mathbf{0})\} = \{\boldsymbol{v} \in C : 0 < d(\boldsymbol{v}, \boldsymbol{c}) \leq w_k\}. \tag{4.2.14}$$

$C_{w_k}(\boldsymbol{c})$ consists of all the codewords in C that are at distances w_1 to w_k from the codeword $\boldsymbol{c}$. For convenience, $C_0(\boldsymbol{c}) \triangleq \phi$. It is clear that $\boldsymbol{v}$ in $C_{w_k}(\boldsymbol{c})$ if and only if $\boldsymbol{c}$ in $C_{w_k}(\boldsymbol{v})$. The subtrellis diagram for $C_{w_k}(\boldsymbol{c})$, denoted $T_{w_k}(\boldsymbol{c})$, is isomorphic to $T_{w_k}(\mathbf{0})$ and can be obtained by adding $\boldsymbol{c}$ to the codewords in $T_{w_k}(\mathbf{0})$. $T_{w_k}(\boldsymbol{c})$ is called the w_k-weight subtrellis centered around $\boldsymbol{c}$.

Let $\varphi_{w_k}(\boldsymbol{c})$ denote the codeword in $C_{w_k}(\boldsymbol{c})$ with the least correlation discrepancy, i.e.,

$$L(\varphi_{w_k}(\boldsymbol{c})) = \boldsymbol{L}[C_{w_k}(\boldsymbol{c})] \tag{4.2.15}$$

Then $\varphi_{w_k}(\boldsymbol{c})$ can be found by searching through the w_k-weight subtrellis $T_{w_k}(\boldsymbol{c})$ using a trellis-based decoding algorithm, such as the Viterbi algorithm or the recursive MLD algorithm [16]. This search operation is called a w_k-weight subtrellis search, denoted w_k-WTS$(\boldsymbol{c})$.

In the proposed decoding algorithm, for a given reference codeword $\boldsymbol{c}$, $C_{w_k}(\boldsymbol{c})$ will be used as the search region with covering distance w_k.

4.3 Iterative Decoding Algorithm I

Decoding begins with the computation of the syndrome of the hard-decision received sequence $\boldsymbol{z}$. If the syndrome of $\boldsymbol{z}$ is zero, then $\boldsymbol{z}$ is the ML codeword $\boldsymbol{c}_{\text{opt}}$. Otherwise starts the decoding iteration process by generating a sequence of candidate codewords for testing and search. Decoding process is terminated when the ML codeword $\boldsymbol{c}_{\text{opt}}$ is found or a termination condition is satisfied. In the following, we first present a method for generating the candidate codewords. Then we discuss a termination condition and an optimality test. Finally, we present the first decoding algorithm.

4.3.1 Generation of Candidate Codewords

Let $\boldsymbol{c}_0$ denote a codeword in C that is obtained by decoding the received sequence $\boldsymbol{r}$ using a simple decoding method. This codeword $\boldsymbol{c}_0$ serves as the initial candidate codeword to start the decoding iteration process. The subsequent candidate codewords, denote $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_i, \ldots$, are generated by a chain of w_k-weight trellis searches in the following order:

$$\boldsymbol{c}_1 = \varphi_{w_k}(\boldsymbol{c}_0),\ \boldsymbol{c}_2 = \varphi_{w_k}(\boldsymbol{c}_1), \ldots, \boldsymbol{c}_i = \varphi_{w_k}(\boldsymbol{c}_{i-1}), \ldots, \tag{4.3.1}$$

where for $i \geq 1$, $\boldsymbol{c}_i$ is the codeword with the least correlation discrepancy found by searching through the w_k-weight subtrellis centered around $\boldsymbol{c}_{i-1}$ using the w_k-WTS($\boldsymbol{c}_{i-1}$) search procedure.

To generate the initial candidate codeword $\boldsymbol{c}_0$, the **first-order** decoding based on the ordered statistics of the received symbols proposed in [17] is used. This decoding is very simple and never fails to produce a decoded codeword. In general, it produces a good initial candidate codeword with small correlation discrepancy which makes the decoding iteration process to converge to the ML codeword fast to reduce the decoding delay and the computational complexity. Another method to generate the initial codeword is to combine an algebraic decoder and an ordered statistic decoder, and take the better one of the two decoded codewords as the initial candidate codeword. If the algebraic decoder fails to decode, then the output of the ordered statistics decoder is used as the initial candidate codeword. This **hybrid** method produces a very good codeword and makes the decoding iteration process converge to the ML codeword faster for higher SNR as it will be shown later in Section 5. Of course, this hybrid method increases the decoding cost, an additional algebraic decoder.

In order to keep the decoding complexity down, the weight w_k should be kept small. Small w_k results in small state and branch complexities of the w_k-weight subtrellis $T_{w_k}(\boldsymbol{c}_i)$ and hence reduces the number of computations needed to search through $T_{w_k}(\boldsymbol{c}_i)$ at the i-th decoding iteration. If C is spanned by the minimum weight codewords, then $w_k = w_1$ is enough for the proposed algorithm to achieve practically optimum error performance provided the length of C is not too long.

4.3.2 A Termination Condition

The above generation of candidate codewords provides a condition to terminate the decoding iteration process in case that the ML codeword can not be found. This termination condition is derived based on the properties of the candidate codewords.

It follows from (4.2.14), the definition of φ_{w_k} and (4.3.1) that $\boldsymbol{c}_{i-1} \notin C_{w_k}(\boldsymbol{c}_{i-1})$

and $\boldsymbol{c}_i = \varphi_{w_k}(\boldsymbol{c}_{i-1}) \in C_{w_k}(\boldsymbol{c}_{i-1})$. Therefore for $i > 0$,

$$\boldsymbol{c}_i \neq \boldsymbol{c}_{i-1}. \tag{4.3.2}$$

The next two lemmas and two corollaries characterize the properties of the candidate codewords generated in the manner of (4.3.1).

Lemma 4.1: For $i \geq 2$,

$$L(\boldsymbol{c}_i) \leq L(\boldsymbol{c}_{i-2}). \tag{4.3.3}$$

Proof: It follows from (4.2.14) and (4.3.1) that $\boldsymbol{c}_{i-2} + \boldsymbol{c}_{i-1}$ and $\boldsymbol{c}_{i-1} + \boldsymbol{c}_i$ are both in $C_{w_k}(\mathbf{0})$. This implies that both $\boldsymbol{c}_{i-2}$ and $\boldsymbol{c}_i$ are in $C_{w_k}(\boldsymbol{c}_{i-1})$. Then it follows from the definition of φ_{w_k} and (4.3.1) that

$$L(\boldsymbol{c}_i) = L(\varphi_{w_k}(\boldsymbol{c}_{i-1})) \leq L(\boldsymbol{c}_{i-2}).$$

This proves the lemma. △△

Lemma 4.1 gives the following ordering of the candidate codewords, $\boldsymbol{c}_0$, $\boldsymbol{c}_1$, ..., $\boldsymbol{c}_i$, ..., in terms of correlation discrepancy:

$$\begin{aligned} &L(\boldsymbol{c}_0) \geq L(\boldsymbol{c}_2) \geq \cdots \geq L(\boldsymbol{c}_{2l}) \geq \cdots, \\ &L(\boldsymbol{c}_1) \geq L(\boldsymbol{c}_3) \geq \cdots \geq L(\boldsymbol{c}_{2l+1}) \geq \cdots. \end{aligned} \tag{4.3.4}$$

This simply says that the candidate codewords are generated in the order of the improvement in correlation metric alternately.

Lemma 4.2: If $j \geq 2$ and $L(\boldsymbol{c}_j) < L(\boldsymbol{c}_{j-2})$, then for $0 \leq i < j$,

$$\boldsymbol{c}_i \neq \boldsymbol{c}_j. \tag{4.3.5}$$

Proof: See Appendix B. △△

Corollary 4.1: If $L(\boldsymbol{c}_i) < L(\boldsymbol{c}_{i-2})$ for $2 \leq i \leq j$, then the candidate codewords, $\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_j$ are distinct.
Proof: From (4.3.2), $\boldsymbol{c}_0 \neq \boldsymbol{c}_1$. From Lemma 4.2, $\boldsymbol{c}_{i'} \neq \boldsymbol{c}_i$ for $0 \leq i' < i$ and $2 \leq i \leq j$. These two facts imply that all the candidate codewords are distinct. △△

A direct consequence of Lemma 4.1 and (4.3.4) is Corollary 4.2. This corollary gives a condition to terminate the decoding iteration process.

Corollary 4.2: If j_0 is the smallest integer such that

$$L(\boldsymbol{c}_{j_0}) = L(\boldsymbol{c}_{j_0-2}), \tag{4.3.6}$$

then

$$\min\{L(\boldsymbol{c}_i) : 0 \leq i \leq j_0\} = \min\{L(\boldsymbol{c}_{j_0-1}), L(\boldsymbol{c}_{j_0})\}. \tag{4.3.7}$$

$\triangle\triangle$

For simplicity, suppose the zero codeword is transmitted and $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$ is the received sequence. For given two different codewords $\boldsymbol{c}^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_N^{(1)})$ and $\boldsymbol{c}^{(2)} = (c_1^{(2)}, c_2^{(2)}, \ldots, c_N^{(2)})$, consider the probability $p(\epsilon)$ that $|L(\boldsymbol{c}^{(1)}) - L(\boldsymbol{c}^{(2)})|/d \leq \epsilon$, where ϵ is a small nonnegative number and $d \triangleq d(\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)})$. From (4.2.1) and (4.2.3), $L(\boldsymbol{c}^{(1)}) - L(\boldsymbol{c}^{(2)}) = \sum_{i=1}^{N} r_i(c_i^{(2)} - c_i^{(1)})$. Since $r_1, r_2, \ldots, r_N$ are independent gaussian random variables with mean $m = -1$ and variance $\sigma^2 = N_o/(2E_b)$, where E_b/N_o is the signal-to-noise ratio per bit, $(L(\boldsymbol{c}^{(1)}) - L(\boldsymbol{c}^{(2)}))/d$ is also a random variable with a normal distribution. The average is $\sum_{i=1}^{N}(c_i^{(1)} - c_i^{(2)})/d$ and the variance is σ^2/d. Clearly, $p(\epsilon)$ approaches to zero as ϵ approaches to zero. On the other hand, if $\boldsymbol{c}_{j_0} = \boldsymbol{c}_{j_0-2}$, then $\varphi_{w_k}(\boldsymbol{c}_{j_0}) = \varphi_{w_k}(\boldsymbol{c}_{j_0-2}) = \boldsymbol{c}_{j_0-1}$ by a deterministic w_k-WTS$(\boldsymbol{c}_{j_0}, C)$, and the repetition occurs.

Let g denote the number of candidate codewords that have been generated by the w_k-WTS procedure. Based upon the above consideration, we use the following simple condition, denoted $\textbf{Cond}_R$, to terminate the decoding iteration process in the simulation presented in Section 4.5: $g \geq 2$ and

$$L(\boldsymbol{c}_{\text{latest}}) = L(\boldsymbol{c}_{\text{latest}}^{(-2)}), \tag{4.3.8}$$

where $\boldsymbol{c}_{\text{latest}} (= \boldsymbol{c}_g)$ denotes the latest generated candidate codeword in (4.3.1) and $\boldsymbol{c}_{\text{latest}}^{(-2)} (= \boldsymbol{c}_{g-2})$ denotes the candidate codeword that was generated two iterations before $\boldsymbol{c}_{\text{latest}}$. Then Corollary 4.1 guarantees no repetition except for the last one. According to Corollary 4.2, the decoder outputs $\textbf{better}(\boldsymbol{c}_{\text{latest}}, \boldsymbol{c}_{\text{latest}}^{(-1)})$ as the decoded codeword and stops the decoding process.

4.3.3 Search Regions, Reference Codewords and Optimality Test

Let G denote the set of candidate codewords that have been generated, including the initial candidate codeword $\boldsymbol{c}_0$. Then, $g = |G| - 1$. Let $\boldsymbol{c}_{\text{best}}$ denote the best candidate codeword in G. Each time a candidate codeword is generated by w_k-WTS, the termination condition $\mathbf{Cond}_R$ is tested. If $\mathbf{Cond}_R$ holds, $\boldsymbol{c}_{\text{best}}$ is the decoded codeword and the decoding process is terminated. In this case, $\boldsymbol{c}_{\text{best}}$ may not be the ML codeword $\boldsymbol{c}_{\text{opt}}$. Otherwise, an optimality test for $\boldsymbol{c}_{\text{best}}$ is performed.

Since $\boldsymbol{c}_i = \varphi_{w_k}(\boldsymbol{c}_{i-1})$ for $1 \le i \le g$, it follows from (4.2.15) that

$$L(\boldsymbol{c}_{\text{best}}) = \boldsymbol{L}[\{\cup_{i=0}^{g-1} C_{w_k}(\boldsymbol{c}_i)\} \cup \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_g\}]. \tag{4.3.9}$$

Define $h \triangleq \min\{h_d, g+1\}$, where h_d is a design parameter and $1 \le h_d \le 3$. For the optimality test of $\boldsymbol{c}_{\text{best}}$, we use the conditions (4.2.11) and (4.2.12) with $Q = C$ and $\boldsymbol{c} = \boldsymbol{c}_{\text{best}}$. Then the question is how to choose the reference codewords $\boldsymbol{c}_i$ with $1 \le i \le h$. Intuitively, we should choose a candidate codeword $\boldsymbol{c}_j$ with a small $L(\boldsymbol{c}_j)$ as a reference codeword, and $\boldsymbol{c}_{\text{latest}}$ should be included as a reference codeword for renewing the iteration in case that the optimality test fails. From (4.2.11), we see that a requirement of the search region S_i around $\boldsymbol{c}_i$ is to satisfy the following condition:

$$L(\boldsymbol{c}_{\text{best}}) \le \boldsymbol{L}[\cup_{i=1}^{h} S_i]. \tag{4.3.10}$$

From (4.3.9), we see that the following choice of S_i is reasonable:

$$S_i = C_{d_i}(\boldsymbol{c}_i) \cup \{\boldsymbol{c}_i\}, \tag{4.3.11}$$

where

$$d_i \triangleq \begin{cases} w_k, & \text{for } \boldsymbol{c}_i \neq \boldsymbol{c}_{\text{latest}}, \\ w_0 = 0, & \text{for } \boldsymbol{c}_i = \boldsymbol{c}_{\text{latest}}. \end{cases} \tag{4.3.12}$$

For $d = w_s$ with $0 \le s \le m$, let $d^{(+1)}$ denote w_{s+1}(i.e., $d^{(+1)} = w_{s+1}$). It follows from (4.2.6), (4.2.14), (4.3.11), (4.3.12) and the definition of weight profile W that

$$C \backslash S_i \subseteq V_N(\boldsymbol{c}_i, d_i^{(+1)}). \tag{4.3.13}$$

It follows from (4.3.9) and (4.3.11) that

$$L(\boldsymbol{c}_{\text{best}}) \le \boldsymbol{L}[\cup_{i=1}^{h} S_i]. \tag{4.3.14}$$

From (4.2.11) to (4.2.13), (4.3.13) and (4.3.14), we obtain the following sufficient condition for $\boldsymbol{c}_{\text{best}} = \boldsymbol{c}_{\text{opt}}$:

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1^{(+1)}; \boldsymbol{c}_2, d_2^{(+1)}; \ldots; \boldsymbol{c}_h, d_h^{(+1)}]. \tag{4.3.15}$$

That is, if $\boldsymbol{c}_{\text{best}}$ satisfies the condition of (4.3.15), it is the most likely codeword $\boldsymbol{c}_{\text{opt}}$.

We use two sequences of reference codewords, denoted $\text{ref}_h^{(1)} = (\boldsymbol{c}_1^{(1)}, \boldsymbol{c}_2^{(1)}, \ldots, \boldsymbol{c}_h^{(1)})$ and $\text{ref}_h^{(2)} = (\boldsymbol{c}_1^{(2)}, \boldsymbol{c}_2^{(2)}, \ldots, \boldsymbol{c}_h^{(2)})$, to evaluate the optimality condition (right-hand side) of (4.3.15). They enforce each other. The reference codewords are chosen from G. For $0 \leq i \leq g+1$ and a subset G' of G, let $\text{best}_i(G')$ denote the sequence of i best candidate codewords in G' arranged in the order of their generation ($\text{best}_0(G')$ denotes the null sequence). Define

$$\text{ref}_h^{(1)} \triangleq (\text{best}_{h-1}(G \backslash \{\boldsymbol{c}_{\text{latest}}\}), \boldsymbol{c}_{\text{latest}}), \tag{4.3.16}$$

$$\text{ref}_h^{(2)} \triangleq \text{best}_h(G \backslash \{\boldsymbol{c}_{\text{latest}}\}). \tag{4.3.17}$$

Note that $\text{ref}_h^{(2)}$ is defined only for $h \leq g$, i.e., $h_d \leq g$. From (4.3.7) of Corollary 4.2, we have that for $h \geq 2$, $\boldsymbol{c}_{\text{best}} \in \text{ref}_h^{(1)}$. It follows from (4.3.12) that

$$d_h^{(1)} = w_0 = 0, \tag{4.3.18}$$

$$d_i^{(j)} = w_k, \text{ except for } j = 1 \text{ and } i = h. \tag{4.3.19}$$

It follows from (4.3.16) to (4.3.19) that the sufficient conditions for optimality of $\boldsymbol{c}_{\text{best}}$ given by (4.3.15) based on $\text{ref}_h^{(1)}$ and $\text{ref}_h^{(2)}$ are:

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{\boldsymbol{L}}^{(1)} \triangleq \underline{\boldsymbol{L}}[\boldsymbol{c}_1^{(1)}, w_{k+1}; \ldots; \boldsymbol{c}_{h-1}^{(1)}, w_{k+1}; \boldsymbol{c}_h^{(1)}, w_1], \tag{4.3.20}$$

$$L(\boldsymbol{c}_{\text{best}}) \leq \underline{\boldsymbol{L}}^{(2)} \triangleq \underline{\boldsymbol{L}}[\boldsymbol{c}_1^{(2)}, w_{k+1}; \ldots; \boldsymbol{c}_h^{(2)}, w_{k+1}], \tag{4.3.21}$$

where $\boldsymbol{c}_h^{(1)} = \boldsymbol{c}_{\text{latest}}$. Sequence $\text{ref}_h^{(1)}$ is a reasonable choice, except that the last argument in the right-hand side of (4.3.20) is w_1. To compensate this shortcoming, $\text{ref}_h^{(2)}$ is introduced. Since either (4.3.20) or (4.3.21) is sufficient for $\boldsymbol{c}_{\text{best}}$ to be the ML codeword $\boldsymbol{c}_{\text{opt}}$, the following combined test for optimality, denoted $\mathbf{Cond}_{OP}$, is performed:

$$\mathbf{Cond}_{OP}: \; L(\boldsymbol{c}_{\text{best}}) \leq \max\{\underline{\boldsymbol{L}}^{(1)}, \underline{\boldsymbol{L}}^{(2)}\}, \text{ where } \underline{\boldsymbol{L}}^{(2)} = 0 \text{ for } g < h_d. \tag{4.3.22}$$

4.3.4 Algorithm I-w_k

Suppose $\boldsymbol{z} \notin C$. Then the decoding iteration begins. Let $f(\boldsymbol{r}, C)$ denote the initial candidate codeword, and let bG denote the sequence $\text{best}_{\min\{h,g\}}(G \backslash \{\boldsymbol{c}_{\text{best}}\})$.

The iterative decoding algorithm I-w_k consists of the following steps:

Step 1: Generate $f(\boldsymbol{r}, C)$. Initialize $\boldsymbol{c}_{\text{latest}} \leftarrow f(\boldsymbol{r}, C)$, $g \leftarrow 0$, $h \leftarrow 1$, $G \leftarrow \{\boldsymbol{c}_{\text{latest}}\}$, $\boldsymbol{c}_{\text{best}} \leftarrow f(\boldsymbol{r}, C)$, $bG \leftarrow \phi$.

Go to **Step 2**.

Step 2: Test condition $\textbf{Cond}_{OP}$. If $\textbf{Cond}_{OP}$ holds, output $\boldsymbol{c}_{\text{best}}$ and stop. Otherwise, update bG and go to **Step 3**.

Step 3: Generate $\boldsymbol{c}_{\text{latest}}$ by the w_k-WTS$(\boldsymbol{c}^{(-1)}_{\text{latest}}, C)$ procedure where $\boldsymbol{c}^{(-1)}_{\text{latest}}$ denotes the last $\boldsymbol{c}_{\text{latest}}$. Update the global variables, g, h, G, $\boldsymbol{c}_{\text{best}}$ and $\boldsymbol{c}_{\text{latest}}$. Test condition $\textbf{Cond}_{R}$. If $\textbf{Cond}_{R}$ holds, output $\boldsymbol{c}_{\text{best}}$ and stop. Otherwise, go to **Step 2**.

Note that in this algorithm, two candidate codewords(may be the same), $\boldsymbol{c}_{\text{best}}$ and $\boldsymbol{c}_{\text{latest}}$, and a few candidate codewords in G (at most a total of $h + 1$ candidate codewords) need to be stored.

4.4 Iterative Decoding Algorithm II

As it will be shown in Section 4.5, Algorithm I-w_1 gives very good error performance for codes that are spanned by their minimum weight codewords. Searches based on a minimum weight subtrellis result in a large reduction in computational complexity. However, Algorithm I-w_1 results in a large performance degradation for codes which are not spanned by their minimum weight codewords. To reduce the performance degradation, we may use a larger weight subtrellis for search. This increases the decoding complexity. In this section, we present a modification of Algorithm I for application to codes that are not spanned by their minimum weight codewords. This algorithm is devised to keep the performance degradation small compared to MLD without using a larger subtrellis for search. The basic concept is to divide a code into cosets based on a properly chosen subcode and then apply Algorithm I to each coset in a specific order.

Let C be a binary (N, K) linear block code with weight profile, $W = \{w_0 = 0, w_1, w_2, \ldots\}$, where w_1 is the minimum weight and $w_1 < w_2 < \cdots$. Let C_0 be a binary (N, K_0) linear subcode of C with weight profile, $W_0 = \{w_{00} = 0, w_{01}, w_{02}, \ldots\}$, where w_{01} is the minimum weight of C_0 and $w_{01} < w_{02} < \cdots$. Then $W_0 \subseteq W$ and $w_1 \leq w_{01}$. Let w_{0k} be the k-th weight in W_0. Suppose the codewords of weights w_{01} to w_{0k} in C_0 span C_0. Partition C with respect to C_0. Then C/C_0 consists of 2^{K-K_0} cosets of C_0. Let d_{C/C_0} denote the minimum distance between different cosets in C/C_0. Note that

$$\min\{d_{C/C_0}, w_{01}\} = w_1. \tag{4.4.1}$$

Let $C_{0,w_{0k}}(\mathbf{0})$ denote the subcode of C_0 that consists of all the nonzero codewords of weights up to and including w_{0k}, and let $T_{C_{0,w_{0k}}}(\mathbf{0})$ denote the minimal trellis for $C_{0,w_{0k}}(\mathbf{0})$. For a codeword $\boldsymbol{c}$ in a coset B of C/C_0, define

$$C_{0,w_{0k}}(\boldsymbol{c}) \triangleq \{\boldsymbol{c} + \boldsymbol{v} : \boldsymbol{v} \in C_{0,w_{0k}}(\mathbf{0})\}. \tag{4.4.2}$$

Then the w_{0k}-weight subtrellis for $C_{0,w_{0k}}(\boldsymbol{c})$ centered around $\boldsymbol{c}$, denoted $T_{C_{0,w_{0k}}}(\boldsymbol{c})$, can be obtained by adding $\boldsymbol{c}$ to all the codewords in $T_{C_{0,w_{0k}}}(\mathbf{0})$. The best codeword in $C_{0,w_{0k}}(\boldsymbol{c})$ can be found by searching through $T_{C_{0,w_{0k}}}(\boldsymbol{c})$. This search is called a w_{0k}-weight search of B around $\boldsymbol{c}$ and is denoted w_{0k}-WTS$(\boldsymbol{c}, C_0)$.

Decoding algorithm II consists of iterative w_{0k}-weight subtrellis searches of all the cosets $B \in C/C_0$ using Algorithm I. Let **Search-in**(B) denote the procedure of searching the coset B. The searches are performed in a serial manner, one coset at a time, and the searches are shifted from one coset to another in a specific order and based on certain conditions. In searching a coset, the candidate codewords are generated in the same manner as described in Section 4.3.1. Lemmas 4.1 and 4.2 and Corollaries 4.1 and 4.2 apply to any coset in C/C_0.

4.4.1 Coset Ordering and Generation of Initial Candidate Codewords

To start the search of a coset $B \in C/C_0$, an initial candidate codeword, denoted $f(\boldsymbol{r}, B)$, must be generated. At the beginning of the decoding process, 2^{K-K_0} initial candidate codewords, one for each coset $B \in C/C_0$, are generated. Let G_{I} denote the set of 2^{K-K_0} initial candidate codewords. These initial candidate codewords are ordered in the increasing order of correlation discrepancies. Then the cosets in C/C_0 are numbered according to this ordering, the first coset corresponds to the least correlation discrepancy and the last coset corresponds to the largest correlation discrepancy. During the decoding process, cosets in C/C_0 will be searched in this order.

The zero-th order decoding in [17] can be modified for decoding the cosets in C/C_0. For a given received sequence $\boldsymbol{r} = (r_1, r_2, \ldots, r_N)$, let M_K denote the location set of the **most reliable basis** of the column space of a generator matrix for C, and Λ_{N-K_0} denote the location set of the **least reliable basis** for the column space of a parity-check matrix of C_0. Then it follows from Theorem 1 in [31] that

$$|M_K \cap \Lambda_{N-K_0}| = K - K_0. \tag{4.4.3}$$

Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_N)$ be the hard-decision received vector obtained from $\boldsymbol{r}$. Then there is an unique codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_N) \in B$ for which

$$c_i = z_i \text{ for } i \in M_K \cap \bar{\Lambda}_{N-K_0}, \tag{4.4.4}$$

where $\bar{\Lambda}_{N-K_0} \triangleq \{1, 2, \ldots, N\} \backslash \Lambda_{N-K_0}$. This codeword, denoted $f(\boldsymbol{z}, B)$, is simply the decoded codeword obtained by zero-th order decoding of coset B. Then $f(\boldsymbol{z}, B)$ is chosen as the initial candidate codeword for coset B.

Hybrid method as described in Section 4.3.1 can be used to generate an initial candidate codeword for the most likely coset $B \in C/C_0$.

4.4.2 Termination Conditions and Optimality Tests

Let G denote the set of candidate codewords in C that have been generated already by Algorithm II. Note that $G_{\mathrm{I}} \subseteq G$. Define

$$G_{\mathrm{WTS}} \triangleq \{\boldsymbol{c} \in G : \text{ the } w_{0k}\text{-WTS}(\boldsymbol{c}, C_0) \text{ has been performed already}\}.$$

Let $\boldsymbol{c}_{\mathrm{best}}$ denote the best candidate codeword in G and $\boldsymbol{c}_{\mathrm{latest}}$ denote the latest candidate codeword. Note that $\boldsymbol{c}_{\mathrm{latest}} \notin G_{\mathrm{WTS}}$. There are two termination conditions in Algorithm II, one for terminating the search of a coset B in C/C_0, called the **local termination condition**, and the other for terminating the search of the entire code C and making a decoding decision, called the **global termination condition** and denoted $\mathbf{Cond}_E$. $\mathbf{Cond}_E$ is the condition that all the cosets in C/C_0 have been searched and the ML codeword has not been identified. In such a case, the decoder outputs $\boldsymbol{c}_{\text{best}}$ as the decoded codeword and stops the decoding process.

Let B be the coset in C/C_0 that is currently being searched. Let R be the set consisting of B and all the cosets in C/C_0 that have been searched. Let G_B denote the set of candidate codewords in B that have been generated already by **Search-in**(B) and define $g_B \triangleq |G_B| - 1$. For a codeword $\boldsymbol{c} \in B$, let $\varphi_{C_0,w_{0k}}(\boldsymbol{c})$ denote the best codeword obtained by searching $T_{C_0,w_{0k}}(\boldsymbol{c})$ using the w_{0k}-WTS$(\boldsymbol{c}, C_0)$ procedure as described in Section 4.3. The following local termination condition $\mathbf{Cond}_{B,R}$ is used in **Search-in**(B) as in Algorithm I:

$$\mathbf{Cond}_{B,R} : g_B \geq 2 \text{ and } L(\boldsymbol{c}_{\text{latest}}) = L(\boldsymbol{c}_{\text{latest}}^{(-2)}).$$

When this condition holds, the search is shifted to the next coset.

4.4.3 Optimality Tests

There are two optimality tests, one for testing the optimality of $\boldsymbol{c}_{\text{best}}$ as the ML codeword in the entire code C, called the **global optimality test** and denoted $\mathbf{Cond}_{C,OP}$, and the other for testing whether $\boldsymbol{c}_{\text{best}}$ is the best codeword in the coset B currently being searched, called the **local optimality test** and denoted $\mathbf{Cond}_{B,OP}$. When $\mathbf{Cond}_{C,OP}$ holds, the decoder outputs $\boldsymbol{c}_{\text{best}}$ as the decoded codeword and stops the decoding process. When $\mathbf{Cond}_{B,OP}$ holds, none of the remaining codewords in B is better than $\boldsymbol{c}_{\text{best}}$ and search must be shifted from B to the next coset in C/C_0.

Since for $\boldsymbol{c} \in G_{\text{WTS}}$, w_{0k}-WTS$(\boldsymbol{c}, C_0)$ has been performed,

$$L(\varphi_{0,w_{0k}}(\boldsymbol{c})) = \boldsymbol{L}[C_{0,w_{0k}}(\boldsymbol{c})]. \tag{4.4.5}$$

Hence, we have that

$$L(\boldsymbol{c}_{\text{best}}) = \boldsymbol{L}[\cup_{\boldsymbol{u}\in G_{\text{WTS}}} C_{0,w_{0k}}(\boldsymbol{c}) \cup G]. \tag{4.4.6}$$

Define

$$d \triangleq \min\{d_{C/C_0}, w_{0(k+1)}\}, \tag{4.4.7}$$

$$h \triangleq \min\{h_d, |G|\}. \tag{4.4.8}$$

It follows from (4.4.1) and (4.4.7) that $d \geq w_1$ and $d > w_1$ if and only if

$$d_{C/C_0} > w_{01}. \tag{4.4.9}$$

For the optimality condition $\mathbf{Cond}_{C,OP}$, if $d > w_1$, then we use the same two sequences of reference codewords $\mathrm{ref}_h^{(1)} = (\boldsymbol{c}_1^{(1)}, \boldsymbol{c}_2^{(1)}, \ldots, \boldsymbol{c}_h^{(1)})$ and $\mathrm{ref}_h^{(2)} = (\boldsymbol{c}_1^{(2)}, \boldsymbol{c}_2^{(2)}, \ldots, \boldsymbol{c}_h^{(2)})$, as given by (4.3.16) and (4.3.17) in Algorithm I. (Note that $\mathrm{ref}_h^{(2)}$ is defined only if $|G| > h_d$.) Then the search region $S_i^{(j)}$ around $\boldsymbol{c}_i^{(j)}$ for $1 \leq i \leq h$ and $j \in \{1, 2\}$ is given as follows:

$$S_i^{(j)} \triangleq \begin{cases} \{\boldsymbol{c}_i^{(j)}\}, & \text{for } \boldsymbol{c}_i^{(j)} \notin G_{\mathrm{WTS}}, \\ C_{0,w_{0k}}(\boldsymbol{c}_i^{(j)}) \cup \{\boldsymbol{c}_i^{(j)}\}, & \text{for } \boldsymbol{c}_i^{(j)} \in G_{\mathrm{WTS}}. \end{cases} \tag{4.4.10}$$

It follows from (4.4.1), (4.4.2) and (4.4.7) that

$$C \backslash S_i^{(j)} \subseteq V_N(\boldsymbol{c}_i^{(j)}, d_i^{(j)}), \tag{4.4.11}$$

where

$$d_i^{(j)} = \begin{cases} w_1, & \text{for } \boldsymbol{c}_i^{(j)} \notin G_{\mathrm{WTS}}, \\ d, & \text{for } \boldsymbol{c}_i^{(j)} \in G_{\mathrm{WTS}}. \end{cases} \tag{4.4.12}$$

From (4.4.6), (4.4.10) and (4.4.11), we have the following sufficient condition for $\boldsymbol{c}_{\text{best}}$ to be the ML codeword:

$$\mathbf{Cond}_{C,OP} : L(\boldsymbol{c}_{\text{best}}) \leq \max\{\underline{\boldsymbol{L}}_C^{(1)}, \underline{\boldsymbol{L}}_C^{(2)}\}, \tag{4.4.13}$$

where for $j \in \{1, 2\}$,

$$\underline{\boldsymbol{L}}_C^{(j)} \triangleq \underline{\boldsymbol{L}}[\boldsymbol{c}_1^{(j)}, d_1^{(j)}; \ldots; \boldsymbol{c}_{h-1}^{(j)}, d_{h-1}^{(j)}; \boldsymbol{c}_h^{(j)}, d_h^{(j)}]. \tag{4.4.14}$$

For the case of $d > w_1$, Algorithm II is more effective than the case of $d_{C/C_0} = w_1$. If $d = w_1$ or $|G| \leq h_d$,

$$\mathbf{Cond}_{C,OP} : L(\boldsymbol{c}_{\text{best}}) \leq \underline{\boldsymbol{L}}_C^{(1)}. \tag{4.4.15}$$

For the local optimality condition $\mathbf{Cond}_{B,OP}$, we use the following two sequences of reference codewords, $\mathrm{ref}_{B,h}^{(1)} = (\boldsymbol{c}_1^{(1)}, \boldsymbol{c}_2^{(1)}, \ldots, \boldsymbol{c}_h^{(1)})$ and $\mathrm{ref}_{B,h}^{(2)} = (\boldsymbol{c}_1^{(2)}, \boldsymbol{c}_2^{(2)}, \ldots, \boldsymbol{c}_h^{(2)})$: (1) If $g_B + 1 \geq h$, then $\mathrm{ref}_{B,h}^{(1)} \triangleq \mathrm{ref}_h^{(1)}$, and otherwise

$$\mathrm{ref}_{B,h}^{(1)} \triangleq (\mathrm{best}_{h-g_B-1}(G \backslash G \cap B), \mathrm{ref}_{g_B+1}^{(1)}). \tag{4.4.16}$$

(2) If $g_B \geq h$, then $\text{ref}_{B,h}^{(2)} = \text{ref}_h^{(2)}$, and otherwise, if $|G| > h_d$,

$$\text{ref}_{B,h}^{(2)} \triangleq (\text{best}_{h-g_B}(G \backslash G \cap B), \text{ref}_{g_B}^{(2)}). \tag{4.4.17}$$

Then the search region $S_{B,i}^{(j)}$ in B around $\boldsymbol{c}_i^{(j)}$ is defined the same way as $S_i^{(j)}$ given by the right-hand side of (4.4.10). It follows from (4.4.2), (4.4.7) and (4.4.10) that

$$B \backslash S_{B,i}^{(j)} \subseteq V_N(\boldsymbol{c}_i^{(j)}, d_i^{(j)}), \tag{4.4.18}$$

where

$$d_i^{(j)} \triangleq \begin{cases} d_{C/C_0}, & \text{for } \boldsymbol{c}_i^{(j)} \notin B, \text{ (from (4.4.7))}, \\ w_{0(k+1)}, & \text{for } \boldsymbol{c}_i^{(j)} \in B \cap G_{\text{WTS}}, \\ w_{01}, & \text{for } \boldsymbol{c}_i^{(j)} \in B \backslash B \cap G_{\text{WTS}}. \end{cases} \tag{4.4.19}$$

From the definition of $S_{B,i}^{(j)}$, (4.4.6) and (4.4.18), we have the following sufficient condition that none of the remaining codewords in B is better than $\boldsymbol{c}_{\text{best}}$:

$$\textbf{Cond}_{B,OP} : L(\boldsymbol{c}_{\text{best}}) \leq \max\{\underline{\boldsymbol{L}}_B^{(1)}, \underline{\boldsymbol{L}}_B^{(2)}\}, \tag{4.4.20}$$

where $\underline{\boldsymbol{L}}_B^{(j)}$ with $j \in \{1, 2\}$ is defined as

$$\underline{\boldsymbol{L}}_B^{(j)} \triangleq \underline{\boldsymbol{L}}[\boldsymbol{c}_1^{(j)}, d_1^{(j)}; \boldsymbol{c}_2^{(j)}, d_2^{(j)}; \ldots; \boldsymbol{c}_h^{(j)}, d_h^{(j)}]. \tag{4.4.21}$$

If $|G| \leq h_d$,

$$\textbf{Cond}_{B,OP} : L(\boldsymbol{c}_{\text{best}}) \leq \underline{\boldsymbol{L}}_B^{(1)}. \tag{4.4.22}$$

In searching a coset $B \in C/C_0$, if none of the conditions, $\textbf{Cond}_{B,R}$ and $\textbf{Cond}_{B,OP}$, holds, search in B continues until one of the condition holds.

4.4.4 Algorithm II-w_{0k}

Algorithm-II is simply a procedure of using Algorithm-I to search each coset in C/C_0 to find the ML codeword. In the process of searching a coset B, the global termination conditions, $\textbf{Cond}_E$ and the global optimality test $\textbf{Cond}_{C,OP}$ are used to stop the entire decoding procedure, and the local termination condition $\textbf{Cond}_{B,R}$ and the local optimality test $\textbf{Cond}_{B,OP}$ are used to shift the search from the current coset B to the next coset in C/C_0.

In the decoding procedure of Algorithm II-w_{0k}, global variables g_B, G_B and bG_B for $B \in C/C_0$ are used besides G, bG, $\boldsymbol{c}_{\text{best}}$, $\boldsymbol{c}_{\text{latest}}$ as defined in Section 4.3.4. bG_B denotes $\text{best}_{\min\{h, g_B\}}(G \backslash \{\boldsymbol{c}_{\text{latest}}\})$. Assume that $\boldsymbol{z} \notin C$. The decoding procedure of Algorithm II-w_{0k} consists of the following steps:

step 1 : Generate the initial candidate codewords $f(\boldsymbol{r}, B)$ for all the cosets $B \in C/C_0$. Order the cosets according to the order of increasing correlation discrepancies of the initial candidate codewords. Initialize $G \leftarrow \{f(\boldsymbol{r}, B) : B \in C/C_0\}$, $\boldsymbol{c}_{\text{best}} \leftarrow \text{best}_1(G)$, $\boldsymbol{c}_{\text{latst}} \leftarrow \boldsymbol{c}_{\text{best}}$, $h \leftarrow \min\{h_d, 2^{K-K_0}\}$ and $bG \leftarrow \text{best}_{\min\{h, 2^{K-K_0}-1\}}(G \backslash \{\boldsymbol{c}_{\text{latest}})$. If condition, $\mathbf{Cond}_{C,OP}$, holds, then output $\boldsymbol{c}_{\text{best}}$ and stop. Otherwise, initialize the search of the first coset B_1 with $G_{B_1} \leftarrow \{f(\boldsymbol{r}, B_1)\}$, $bG_B \leftarrow \phi$ and go to **step 2**.

step 2 : Execute **Search-in**(B_i). Each time, when a new candidate codeword is generated, update the global variables, G, bG, G_{B_i}, g_B, h, bG_{B_i}, $\boldsymbol{c}_{\text{best}}$ and $\boldsymbol{c}_{\text{latest}}$. First test the local termination condition $\mathbf{Cond}_{B_i,R}$. If $\mathbf{Cond}_{B_i,R}$ holds, then go to **step 3**. Otherwise, test the global optimality test $\mathbf{Cond}_{C,OP}$. If $\mathbf{Cond}_{C,OP}$ holds, output $\boldsymbol{c}_{\text{best}}$ and stop the decoding process. Otherwise, test the local optimality test $\mathbf{Cond}_{B_i,OP}$. If $\mathbf{Cond}_{B_i,OP}$ holds, exit B_i and go to **step 3**. Otherwise, go to **step 2**.

step 3 : If all the cosets in C/C_0 have been exhausted, then output $\boldsymbol{c}_{\text{best}}$ and stop. Otherwise, call **Search-in**(B_{i+1}).

Algorithm II is basically based on the divide-and-conquer technique to cover a large search space using low weight subtrellis searches. It is effective only when 2^{K-K_0} is not too big. This algorithm is devised primarily for codes that are not spanned by their minimum weight codewords, i.e., **unequal error protection(UEP)** codes[32].

4.5 Simulation Results: Error Performance and Computational Complexity

Both Algorithm I and Algorithm II with $h_d = 3$ and minimum-weight subtrellis searches(w_1-WTS) have been applied to some well known codes of length 48, 64 and 128. Some simulation results in error performance and computational complexity are shown in Figures 4.1–4.9, Tables 4.1 and 4.2. The computational complexity is evaluated in terms of average number of additions and comparisons of metrics, called addition equivalent operations, that are required for the generation of the initial candidate codeword(s), the optimality tests, and low weight subtrellis searches.

The initial candidate codewords are generated with the first-order decoding based on the ordered statistics of the received symbols[17]. For convenience, we call it the order-1 decoding. For minimum-weight subtrellis search, the recursive MLD algorithm (RMLD) proposed in [16] is used. This algorithm is more effective than the Viterbi algorithm with optimal sectionalization[21]. Table 4.1 lists the numbers of addition equivalent operations required by RMLD algorithm for the full code trellises and their low weight subtrellises of some example codes.

For the (48,24,12) extended quadratic residue(EQR) code, Algorithm I-w_1 achieves practically the same error performance as MLD with a significant reduction in computational complexity as shown in Figures 4.1 and 4.2 and, Table 4.1. From Figure 4.2 and Table 4.1, we see that the average number of addition equivalent operations required by Algorithm I-w_1 are 17% and 1.43% of that required by the RMLD algorithm (13.3% and 1.12% of that required by the Viterbi algorithm with optimum sectionalization) at SNR = 0.0 dB and 5.0 dB, respectively. Figure 4.3 shows the relative frequency of the number of iterations. The maximum number of iterations is only 5 at SNR=0.0, 1,0 and 2.0 dB. The numbers of occurrences of this event are only 30, 5 and 3 at SNR=0.0, 1,0 and 2.0 dB, respectively over 100,000 trials.

For the (64,22,16) RM code, the (64,42,8) RM code, and the (64,45,8) extended BCH(EBCH) code, simulation results show that Algorithm I-w_1 achieves practically optimum error performance (i.e., error performance curve falls on top of the MLD error performance curve) with significant reductions in decoding computational complexity. Table 4.2 gives a comparison between Algorithm I-w_1 and the MLK algorithm[27] in terms of performance degradation with respect to MLD and computational complexity for the (64,22,16) and (64,42,8) RM codes, and also shows the simulation results for the (64,45,8) EBCH code. We see that for the two RM codes, Algorithm I-w_1 outperforms the MLK algorithm.

For the (128,29,32) RM codes, Algorithm I-w_1 results in a very small degradation in error performance compared to MLD, less than 0.28 dB with the hybrid order-1 initial decoding as shown in Figure 4.4. With MLK decoding algorithm, the degradation in error performance compared to MLD is 1.7 dB. For the entire range of SNR greater than 0.0 dB, the average number of decoding iterations required to decode a received word is small, less than 2.8 for SNR=0.0 dB and less than 1.3 for SNR=4 dB as shown in Figure 4.5. The average numbers of addition equivalent operations required at SNR=0.0 dB and 4.0 dB are 41,277 and 18,317, respectively with the hybrid-1 initial decoding. From Figure 4.5 and Table 4.1, the average numbers of addition equivalent operations are about 0.9% and 0.4% of that required by the RMLD algorithm based on the full code trellis at SNR = 0.0 dB and 4.0 dB, respectively. If the Viterbi decoding algorithm based on the full code trellis is used to decode this code, then the total number of addition equivalent operations required is more than 6 millions. For this code, Algorithm I-w_1 provides an excellent trade-off between error performance and decoding complexity.

Now consider the (64,24,16) EBCH code which contains the (64,22,16) RM code as a proper subcode. The minimum weight codewords do not span the code but span the (64,22,16) RM subcode. For this code, Algorithm I-w_1 results in a large performance degradation compared to optimum MLD as shown in Figure 4.6. The degradation(0.9 dB) is worse than that of the MLK algorithm with w_1-weight trellis search, 0.6 dB at SNR=4.0 dB[27]. The weight next to the minimum weight is w_2=18. Performance degradation is reduced if w_2-weight trellis search is used in Algorithm I. If the first order hybrid decoding(hybrid-1) is used to generated the initial candidate codeword, Algorithm I-w_2 results in a 0.2 dB performance degradation compared to MLD at 10^{-6} BER as shown in Figure 4.6. Suppose Algorithm II is used to decode this code. Let C be the extended (64,24,16) BCH code and C_0 be the (64,22,16) RM code. The partition C/C_0 consists of 4 cosets of C_0. The first two weights in the weight profile W_0 of C_0 are w_{01}=16 and w_{02}=24, respectively. The minimum distance between cosets in C/C_0 is d_{C/C_0}=18. With the order-1 decoding to generate the initial candidate codeword for each coset, Algorithm II-w_1 achieves practically optimum error performance as shown in Figure 4.6. The degradation is 9.2×10^{-3} dB at SNR=4.0 dB.

From Figure 4.7 and Table 4.1, we find that the average numbers of addition equivalent operations required by Algorithm II-w_1 to decode the (64,24,16) EBCH code are about 1/9 and 1/315 of that required by the RMLD algorithm based on the full code trellis at SNR = 0.0 dB and 5.0 dB, respectively. For this code, Algorithm II-w_1 gives a better coverage of code space using minimum-weight trellis search shifted among the cosets than Algorithm I-w_2 does. The performance improvement of Algorithm II-w_1 over Algorithm I-w_2 is at the expense of some increase in numbers of decoding iterations as shown in Figure 4.7. For example, at SNR=3.0 dB, Algorithm II-w_1 with the order-1 initial decoding requires an average of 2.1 decoding iterations, while Algorithm I-w_2 with the hybrid-1 decoding requires an average of only 0.72 decoding iterations. For the same SNR $= 3.0$ dB, while Algorithm I-w_2 requires an average of 22,306 addition equivalent operations, Algorithm II-w_1 requires an average of 7,564 addition equivalent operations. The numbers of addition equivalent operations of the MLK algorithm with w_1-weight trellis search and w_2-weight trellis search are about 60,000 and 40,000, respectively at SNR=3.0 dB.

Finally consider the extended (128,36,32) BCH code which contains the (128,29,32) RM code as a proper subcode. The minimum weight codewords do not span the code but span the (128,29,32) RM subcode. For this code, the optimum error performance is very hard to evaluated. For computation, we use the union bound on MLD. The weight next to the minimum weight is $w_2 = 36$. Algorithm II-w_1 is used to decode this code. Let C be the extended (128,36,32) BCH code and C_0 be the (128,29,32) RM code. The partition C/C_0 consists of 128 cosets of C_0. The first two weights in the weight profile W_0 of C_0 are w_{01}=32 and w_{02}=48. The minimum distance between cosets in C/C_0 is d_{C/C_0}=36. With the order-1 decoding to generate the initial candidate codeword for each coset, Algorithm II-w_1 achieves good error performance as shown in Figure 4.8. The average numbers of iterations and addition equivalent operations are shown in Figure 4.9. For example, at SNR=4.0 dB, Algorithm II-w_1 with the order-1 initial decoding requires an average of 80 decoding iterations and 1.4×10^6 addition equivalent operations.

4.6 Conclusion

In this chapter, we have presented a soft-decision decoding algorithm for binary linear block codes based on iterative low-weight trellis searches to achieve practically optimum or near optimum error performance with a significant reduction in decoding complexity. Two specific algorithms have been proposed. One algorithm is specifically designed for codes whose minimum weight codewords do not span the codes, such as UEP codes. Both algorithms can be improved in several ways. The effect of quantization to the minimum weight subtrellis search is to be studied.

Code	Full code		Low weight subcode	
	VMLD[25]	RMLD[16]	w_1 RMLD[30]	w_1 and w_2 RMLD[30]
(48,24,12) EQR	548,113	431,503	45,551	—
(64,22,16) RM	101,786	78,209	3,331	—
(64,42,8) RM	538,799	326,017	5,651	—
(128,29,32) RM	—	4,573,304	14,595	—
(64,24,16) EBCH	316,608	271,745	3,331	30,211
(64,45,8) EBCH	891,819	893,489	32,307	—

Table 4.1: Numbers of addition equivalent operations of the Viterbi decoding with optimal sectionalization(VMLD) and RMLD

Code	SNR(dB)	Algorithm I-w_1		MLK[27]	
		DEG(dB)	NAO	DEG(dB)	NAO
(64,22,16) RM	3.0	0.02	3179	0.4	20,000
	4.0	0.01	1732	0.6	14,000
	5.0	0	806	0.6	5,000
(64,42,8) RM	3.0	0.02	4210	1.3	5,400
	4.0	0.009	1790	0.7	1,200
	5.0	0.004	768	0.6	240
(64,45,8) EBCH	3.0	0.03	20336		
	4.0	0.02	6248		
	5.0	0.02	1351		

Table 4.2: Degradations(DEG) and numbers of addition equivalent operations(NAO) of Algorithm I-w_1 with hybrid-1 initial decoding and the MLK algorithm with w_1-weight trellis search

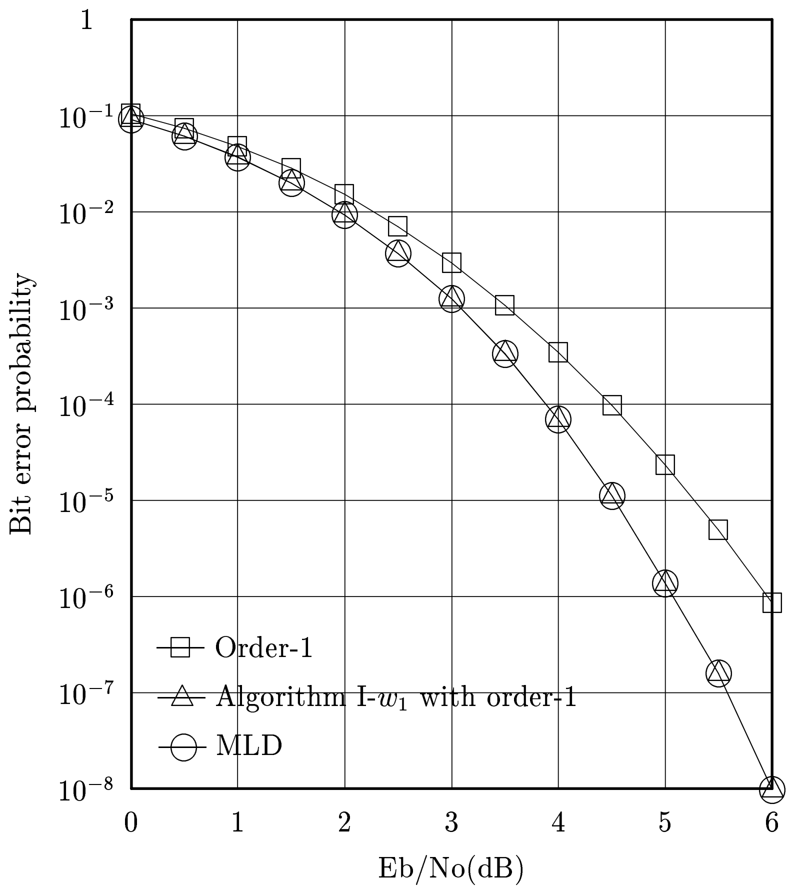

Figure 4.1: Bit error probability for the (48,24,12) EQR code.

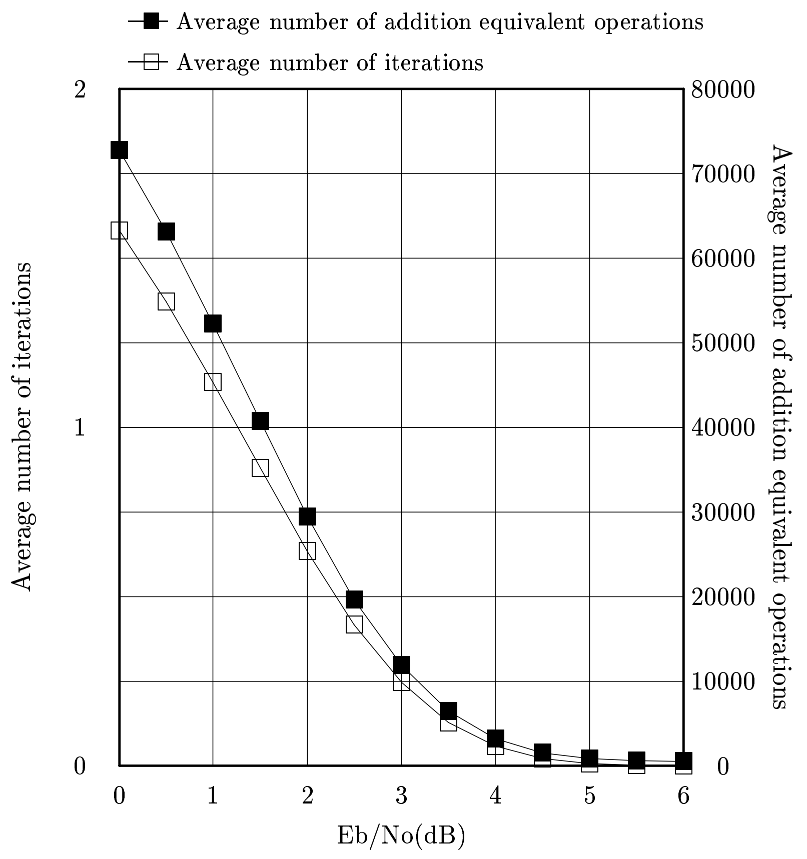

Figure 4.2: Average numbers of iterations and addition equivalent operations for the (48,24,12) EQR code.

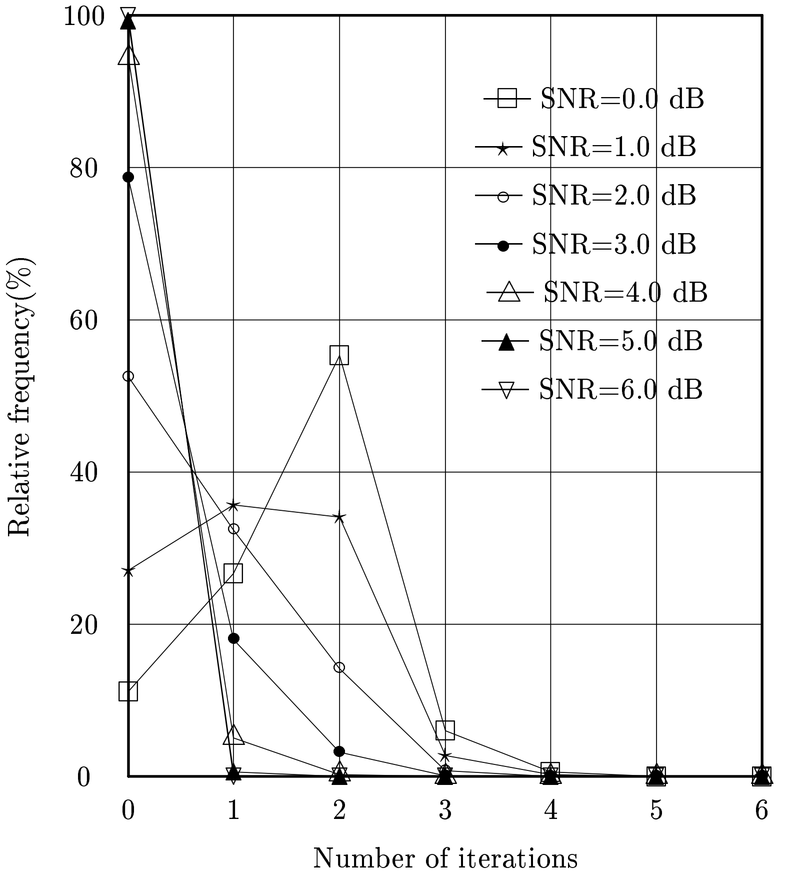

Figure 4.3: Histogram of the numbers of iterations for the (48,24,12) EQR code.

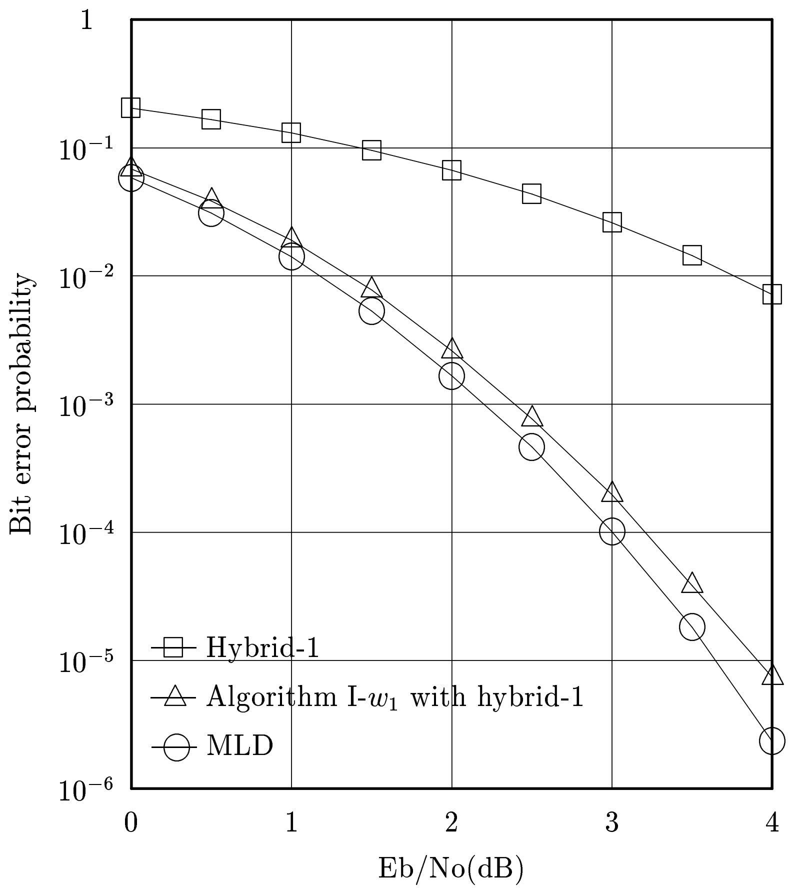

Figure 4.4: Bit error probability for the (128,29,32) RM code.

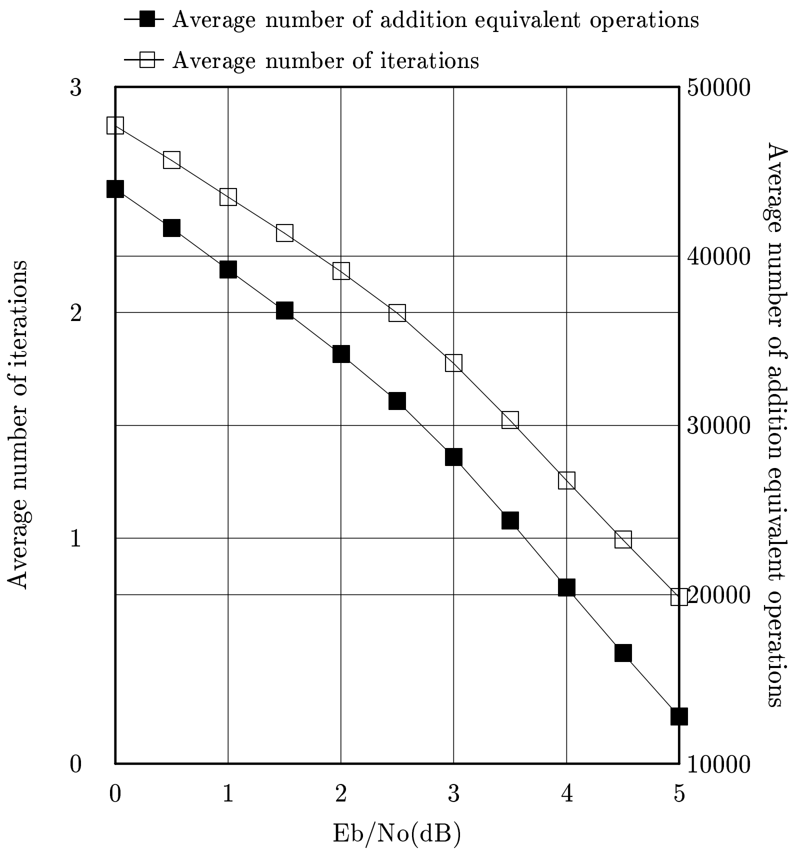

Figure 4.5: Average numbers of iterations and addition equivalent operations for the (128,29,32) RM code.

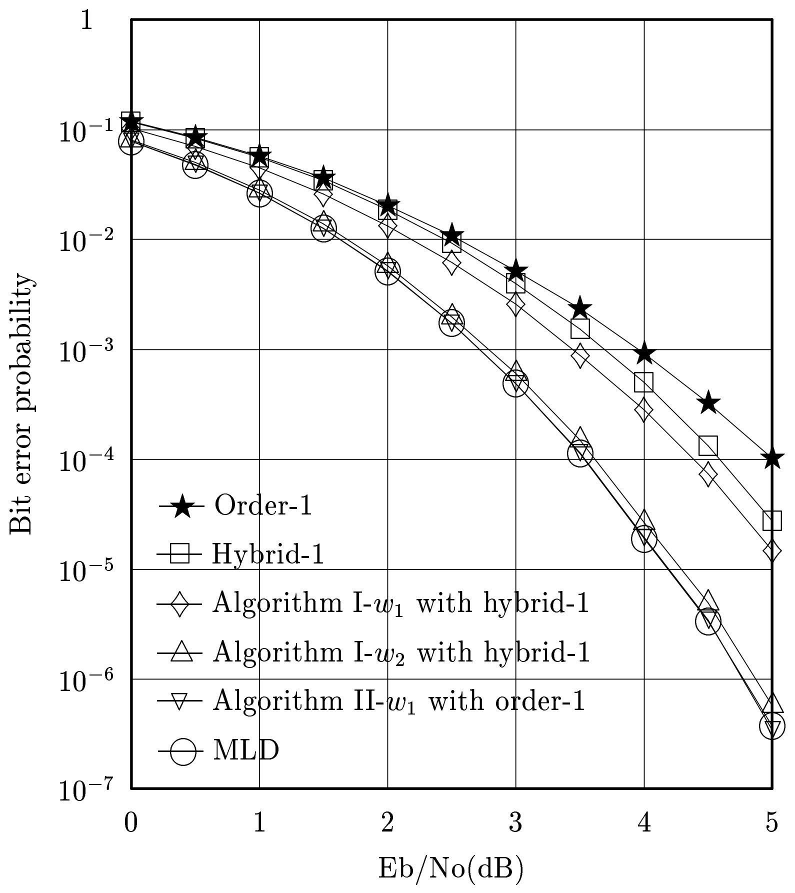

Figure 4.6: Bit error probability for the (64,24,16) EBCH code.

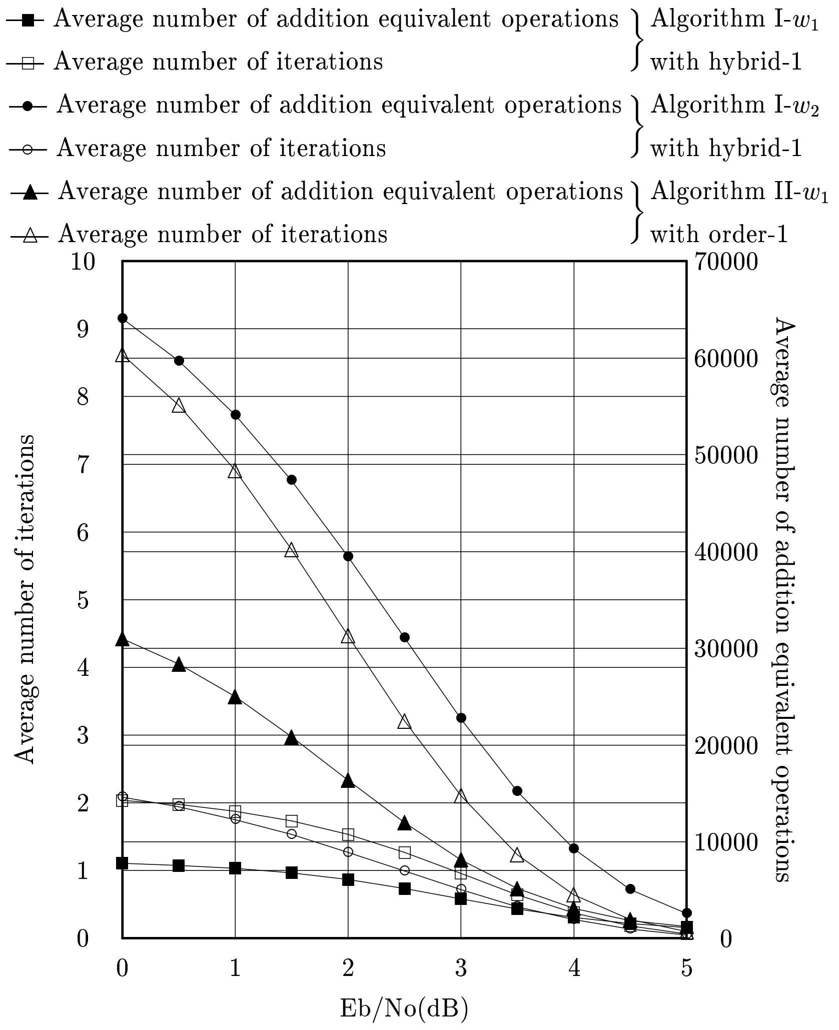

Figure 4.7: Average numbers of iterations and addition equivalent operations for the (64,24,16) EBCH code.

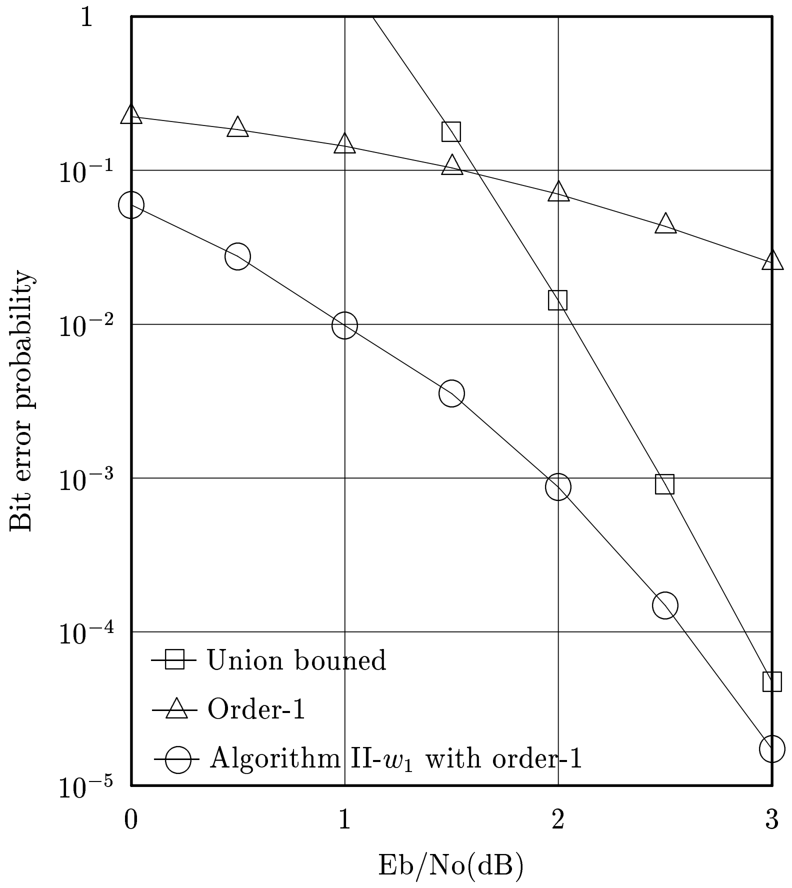

Figure 4.8: Bit error probability for the (128,36,32) EBCH code.

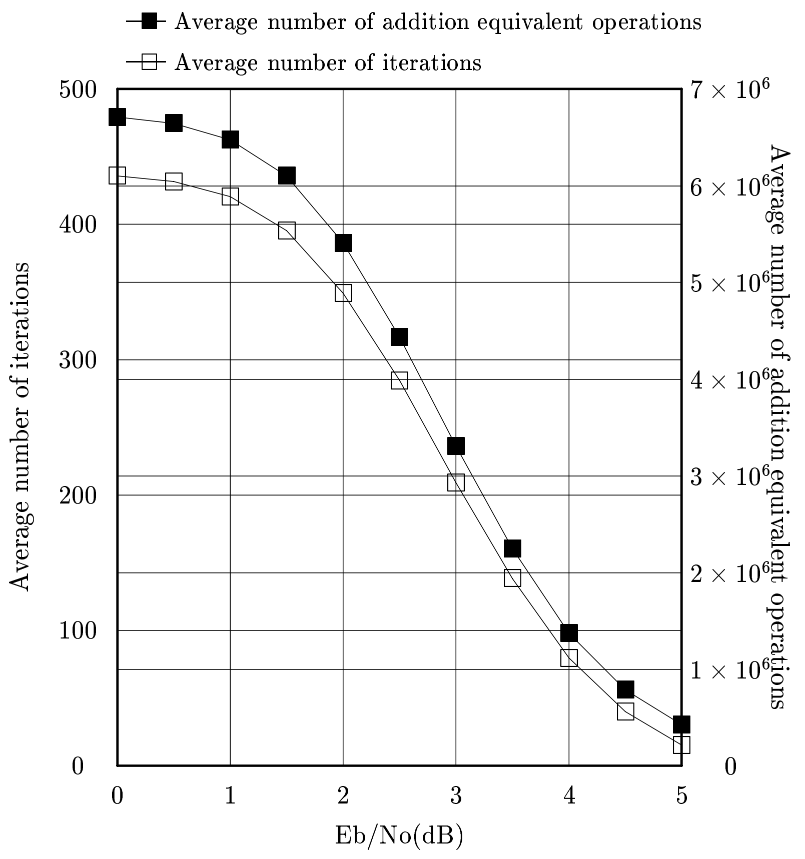

Figure 4.9: Average numbers of iterations and addition equivalent operations for the (128,36,32) EBCH code.

Chapter 5

Conclusion

In this thesis, two new sufficient conditions are derived. The sufficient conditions on optimality of decoded codeword are stronger than all the previously known sufficient conditions. To show their effectiveness, these sufficient conditions were applied to the Chase decoding algorithm II and a newly proposed iterative decoding algorithm. Simulation results show that they are very effective in terminating the decoding process except for high block error probability region.

We have also derived a condition to rule out useless test error patterns in the generation of candidate codewords in a Chase-type decoding algorithm. This rule-out condition reduces many unnecessary decoding iterations and computations.

Finally, we have presented a new soft-decision decoding algorithm for binary linear block codes based on iterative low-weight trellis searches to achieve practically optimum or near optimum error performance with a significant reduction in decoding complexity. Two specific algorithms have been proposed. One algorithm is specifically designed for codes whose minimum weight codewords do not span the codes, such as UEP codes. Both algorithms can be improved in several ways. The effect of quantization to the minimum weight subtrellis search is to be studied.

Bibliography

[1] G. D. Forney,Jr., “Generalized Minimum Distance Decoding,” *IEEE Trans. Info. Theory*, IT-12, pp.125–131, Apr. 1966.

[2] D. Chase, “A Class of Algorithms for Decoding Block Codes with Channel Measurement Information,” IEEE Transactions on Information Theory, Vol. 18, No. 1, pp. 170–182, Jan. 1972.

[3] H. Tanaka and K. Kakigahara, “Simplified Correlation Decoding by Selecting Possible Codewords Using Erasure Information,” *IEEE Trans. Info. Theory*, IT-29, pp.743–748, Sep. 1983.

[4] D. J. Taipale and M. B. Pursley, “An Improvement to Generalized Minimum-Distance Decoding,” IEEE Trans. Information Theory, Vol. 37, pp. 167–172, Jan. 1991.

[5] T. Kaneko, T. Nishijima, H. Inazumi and S. Hirasawa, “An Efficient Maximum-Likelihood-Decoding Algorithm for Linear Block Codes with Algebraic Decoder,” IEEE Transactions on Information Theory, Vol. 40, No. 2, pp. 320–327, Mar. 1994.

[6] H. T. Moorthy, S. Lin and T. Kasami, “Soft-Decision Decoding of Binary Linear Block Codes Based on an Iterative Search Algorithm,” *IEEE Trans. Info. Theory*, IT-43, pp. 1030–1040, May 1997.

[7] S.Lin, H.T.Moorthy and T.Kasami, “An Efficient Soft-Decision Decoding Scheme for Binary Linear Block Codes,” *Proc. of the 3rd International Symposium on Communication Theory & Applications*, 10–14 Jul. 1995, Charlotte Mason Colledge, Ambleside, Lake District, UK.

[8] T.Koumoto, H.Nagano, T.Takata, T.Fujiwara, T.Kasami and S.Lin, "A New Iterative Soft-Decision Decoding Algorithm," *Technical Report of IEICE*, IT95-28, The Inst. of Electronics, Information and Communication Engineers, Japan, Jul. 1995.

[9] T. Koumoto, T. Takata, T. Kasami and S. Lin, "An Iterative Soft-Decision Decoding Algorithm," *Proc. of International Symp. on Information Theory and Its Applications*, pp. 806–810, Canada, Sep. 1996.

[10] T. Kasami, T. Takata, T. Koumoto, T. Fujiwara, H. Yamamoto and S. Lin, "The Least Stringent Sufficient Condition on Optimality of Suboptimal Decoded Codewords," *Technical Report of IEICE*, IT94-82, The Inst. of Electronics, Information and Communication Engineers, Japan, Jan. 1995.

[11] B. Shen, K. K. Tzeng and C. Wang, "A Bounded-Distance Decoding Algorithm for Binary Linear Block Codes Achieving the Minimum Effective Error Coefficient," *IEEE Trans. Info. Theory*, IT-42, pp. 1987–1991, Nov. 1996.

[12] T. Koumoto and T. Kasami, "Analysis and Improvement on GMD like Decoding Algorithms," *Technical Report of IEICE*, IT98-54, The Inst. of Electronics, Information and Communication Engineers, Japan, May 1998. A revised version is presented in Proc. of International Symp. on Information Theory and its Applications, Mexico City, Oct. 1998.

[13] T. Kasami, T. Koumoto, T. Takata and S. Lin, "The Effectiveness of the Least Stringent Sufficient Condition on the Optimality of Decoded Codewords," *Proc. of the 3rd International Symposium on Communication Theory & Applications*, pp.324–333, Jul. 1995, Charlotte Mason College, Ambleside, Lake District, UK.

[14] T. Koumoto, T. Kasami and S. Lin, "A Sufficient Condition for Ruling Out Some Useless Test Error Patterns in Iterative Decoding Algorithms," IEICE Transactions on Fundamentals, Vol. E81-A, No. 2, pp. 321–326, Feb. 1998.

[15] T. Koumoto and T. Kasami, "An Iterative Decoding Algorithm Based on Information of Decoding Failure," Proceedings of the 20th Symposium on Information Theory and Its Applications, pp.325–328, Japan, Dec. 1997.

[16] T. Fujiwara, H. Yamamoto, T. Kasami and S. Lin, “A Trellis-Based Recursive Maximum Likelihood Decoding Algorithm for Linear Block Codes,” IEEE Transactions on Information Theory, Vol. 44, No. 2, pp. 714–729, Mar. 1998.

[17] M. P. C. Fossorier and S. Lin, “Soft-Decision Decoding of Binary Linear Block Codes Based on Ordered Statistics,” IEEE Transactions on Information Theory, Vol. 44. No. 5, pp. 1217–1234, Sep. 1995.

[18] T. Kasami and T. Koumoto, “Computational Complexity for Computing Sufficient Conditions on the Optimality of a Decoded Codeword,” NAIST-IS-TR98008 ISSN 0919-9527, Jul. 1998.

[19] M. Kobayashi, A. Ogino, T. Kohnosu and S. Hirasawa, “On Reducing Complexity of Soft-Decision Decoding,” *Proc. of the 17th Symp. on Information Theory and Its Applications*, pp.333–336, Dec. 1994.

[20] T. Kaneko, T. Nishijima and S. Hirasawa, “Efficient Maximum-Likelihood-Decoding using Bounded Distance Decoding,” *Technical Report of IEICE*, IT94-34, The Inst. of Electronics, Information and Communication Engineers, Japan, Jul. 1994.

[21] A. Vardy and Y. Be’ery, “Maximum Likelihood Soft-Decision Decoding of BCH Codes,” IEEE Transactions on Information Theory, Vol. 40, pp. 546–554, Mar. 1994.

[22] Y. Berger and Y. Be’ery, “Soft Trellis-Based Decoder for Linear Block Codes,” IEEE Transactions on Information Theory, Vol. 40, pp. 764–773, May 1994.

[23] L. Ekroot and S. Dolinar, “A* decoding of block codes,” IEEE Transactions on Communications, Vol. 44, No. 9, pp. 1052–1056, Sep. 1996.

[24] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, “Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes,” IEEE Transactions on Information Theory, Vol. 39, No. 5, pp. 1514–1523, Sep. 1993.

[25] A. Lafourcade and A. Vardy, “Optimum Sectionalization of a Trellis,” IEEE Transactions on Information Theory Vol. 42, No. 3, pp. 689–702, May 1996.

[26] R. J. McEliece, "On the BCJR Trellis for Linear Block Codes," IEEE Transactions on Information Theory, Vol. 42, No. 4, pp. 1072–1092, Jul. 1996.

[27] H. T. Moorthy, S. Lin and T. Kasami, "Soft-Decision Decoding of Binary Linear Block Codes Based on an Iterative Search Algorithm," IEEE Transactions on Information Theory, Vol. 43, No. 3, pp. 1030–1040, May 1997.

[28] Y. Berger and Y. Be'ery, "The Twisted Squaring Construction, Trellis Complexity, and Generalized Weights of BCH and QR codes," IEEE Transactions on Information Theory, Vol. 42, No. 6, pp. 1817–1827, Nov. 1996.

[29] T. Kasami, T. Koumoto, T. Takata and S. Lin, "The Least Stringent Sufficient Conditions on the Optimality of Decoded Codewords," Proceedings of the 1995 IEEE International Symposium on Information Theory, p.470, Whistler, Canada, September 1995. Also submitted to IEEE Transactions on Information Theory, 1996.

[30] T. Kasami, K. Koumoto, T. Fujiwara, H. Yamamoto, Y. Desaki and S. Lin, "Low Weight Subtrellises for Binary Linear Block Codes and Their Application," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E80-A, No. 11, pp. 2095–2103, Nov. 1997.

[31] M. P. C. Fossorier, S. Lin and J. Synders, "On Maximum Likelihood Soft-Decision Syndrome Decoding," IEEE Transactions on Information Theory, Vol. 44, No. 1, pp. 388–398, Jan. 1998.

[32] B. Masnick and J. Wolf, "On Linear Unequal Error Protection Codes," IEEE Transactions on Information Theory, Vol. 13, No. 4, pp. 600–607, Jul. 1967.

Appendix A

Expressions for Evaluating $\underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1; \boldsymbol{c}_2, d_2; \ldots; \boldsymbol{c}_h, d_h]$ for $1 \leq h \leq 3$

Let X be a subset of the index set $I = \{1, 2, \ldots, N\}$ for the positions of components in an N-tuple. For any positive integer $j \leq |X|$, let $X^{(j)}$ denote the subset of j indices, i_1, i_2, ..., i_j in X for which the received symbols, r_{i_1}, r_{i_2}, ..., r_{i_j} have the smallest reliability measures, i.e., for $1 \leq l \leq j$,

$$|r_{i_l}| \leq |r_{i_k}|, \tag{A.1}$$

for $i_k \in X \backslash X^{(j)}$. For a non-positive integer j, $X^{(j)} \triangleq \phi$(the **empty set**) and for $j \geq |X|$, $X^{(j)} \triangleq X$.

For $h = 1$, let $\delta_1 \triangleq d_1 - n(\boldsymbol{c}_1)$ where $n(\boldsymbol{c}_1)$ is the cardinality of $D_1(\boldsymbol{c}_1)$ defined by (4.2.2). Then

$$\underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1] = \sum_{i \in D_0(\boldsymbol{c}_1)^{(\delta_1)}} |r_i|, \tag{A.2}$$

where

$$D_0(\boldsymbol{c}_1) = \{1, 2, \ldots, N\} \backslash D_1(\boldsymbol{c}_1). \tag{A.3}$$

This sufficient condition on optimality with a single reference codeword was first derived in [4].

For $h = 2$, let $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ be two candidate codewords. Let $\delta_1 \triangleq d_1 - n(\boldsymbol{c}_1)$ and $\delta_2 \triangleq d_2 - n(\boldsymbol{c}_2)$. Suppose $\delta_1 \geq \delta_2$. Define the following three index sets:

$$D_{00} \triangleq D_0(\boldsymbol{c}_1) \cap D_0(\boldsymbol{c}_2), \tag{A.4}$$

$$D_{01} \triangleq D_0(\boldsymbol{c}_1) \cap D_1(\boldsymbol{c}_2), \tag{A.5}$$

$$I(\boldsymbol{c}_1, \boldsymbol{c}_2) \triangleq (D_{00} \cup D_{01}^{(\lfloor(\delta_1-\delta_2)/2\rfloor)})^{(\delta_1)}. \tag{A.6}$$

Then,

$$\underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1; \boldsymbol{c}_2, d_2] = \sum_{i \in I(\boldsymbol{c}_1, \boldsymbol{c}_2)} |r_i|. \tag{A.7}$$

Now we consider the case with $h = 3$. For a binary 3-tuple, (b_1, b_2, b_3) in V_3, define the following index set:

$$D_{b_1 b_2 b_3} \triangleq D_{b_1}(\boldsymbol{c}_1) \cap D_{b_2}(\boldsymbol{c}_2) \cap D_{b_3}(\boldsymbol{c}_3), \tag{A.8}$$

where $D_{b_i}(\boldsymbol{c}_i)$ is given by (4.2.2) for $b_i = 1$ and by (A.3) for $b_i = 0$. Let $n_{b_1 b_2 b_3} \triangleq |D_{b_1 b_2 b_3}|$ and $\delta_i \triangleq d_i - n(\boldsymbol{c}_i)$ for $i = 1$, 2 and 3. Without loss of generality, assume that $\delta_i \geq \delta_j$ for $i < j$. Define the following integers:

$$\delta_{12} \triangleq \min\{\delta_1, \lfloor(\delta_1 - \delta_2)/2\rfloor\}, \tag{A.9}$$

$$\delta_{13} \triangleq \min\{\delta_1, \lfloor(\delta_1 - \delta_3)/2\rfloor\}, \tag{A.10}$$

$$\delta^{(1)} \triangleq \max\{0, \delta_{12} - n_{010}, \delta_{13} - n_{001}\}, \tag{A.11}$$

$$\delta^{(2)} \triangleq \min\{n_{011}, \delta_{12}, \delta_{13}, n_{000} + \delta_{12} + \delta_{13} - \delta_1\}. \tag{A.12}$$

If $\delta^{(1)} \leq \delta^{(2)}$, then let $\underline{\boldsymbol{L}}_1$ be defined as

$$\underline{\boldsymbol{L}}_1 \triangleq \min_{\delta^{(1)} \leq \delta \leq \delta^{(2)}} \sum_{i \in (D_{000} \cup D_{001}^{(\delta_{13}-\delta)} \cup D_{010}^{(\delta_{12}-\delta)})^{(\delta_1-\delta)} \cup D_{011}^{(\delta)}} |r_i|. \tag{A.13}$$

Otherwise, $\underline{\boldsymbol{L}}_1$ is defined as ∞.

Consider the parities (even or odd) of δ_i with $1 \leq i \leq 3$. If all the parities are the same, then define $\varepsilon_i \triangleq 0$ for $1 \leq i \leq 3$. Otherwise, there is an index j such that the parity of δ_j is different from the parities of other two δ_i's. Define $\varepsilon_j \triangleq 1$ and $\varepsilon_i \triangleq 0$ for $i \neq j$. Let δ_{12}', δ_{13}', $\delta^{(2)}$ and $\delta^{(3)}$ be defined as

$$\delta_{12}' \triangleq (\delta_1 + \varepsilon_1 - \delta_2 - \varepsilon_2)/2, \tag{A.14}$$

$$\delta_{13}' \triangleq (\delta_1 + \varepsilon_1 - \delta_3 - \varepsilon_3)/2, \tag{A.15}$$

$$\delta^{(3)} \triangleq \max\{0, \lceil(\delta_2 + \delta_3)/2\rceil - n_{000}\}, \tag{A.16}$$

$$\delta^{(4)} \triangleq \min\{n_{100}, n_{010} - \delta_{12}', n_{001} - \delta_{13}', \lceil(\delta_2 + \delta_3)/2\rceil\}. \tag{A.17}$$

If $\delta^{(3)} \leq \delta^{(4)}$, then let $\underline{\boldsymbol{L}}_2$ be defined as

$$\underline{\boldsymbol{L}}_2 \triangleq \min_{\delta^{(3)} \leq \delta \leq \delta^{(4)}} \sum_{i \in D_{000}^{(\lceil(\delta_2+\delta_3)/2\rceil-\delta)} \cup D_{010}^{(\delta'_{12}+\delta)} \cup D_{001}^{(\delta'_{13}+\delta)} \cup D_{100}^{(\delta)}} |r_i|. \tag{A.18}$$

Otherwise, define $\underline{\boldsymbol{L}}_2$ as ∞. Then,

$$\underline{\boldsymbol{L}}[\boldsymbol{c}_1, d_1; \boldsymbol{c}_2, d_2; \boldsymbol{c}_3, d_3] = \min\{\underline{\boldsymbol{L}}_1, \underline{\boldsymbol{L}}_2\}. \tag{A.19}$$

Appendix B

Proof of Lemma 3.2

From (4.3.2), $\boldsymbol{c}_{j-1} \neq \boldsymbol{c}_j$. Suppose there exists an index i such that $0 \leq i \leq j-2$ and

$$\boldsymbol{c}_i = \boldsymbol{c}_j. \tag{B.1}$$

There are two cases to be considered: (i) $j - i$ is even, and (ii) $j - i$ is odd.

Case I: $j - i$ is even.

From the hypothesis of (B.1) and (4.3.3) of Lemma 4.1, we have the following chain of equalities,

$$L(\boldsymbol{c}_i) = L(\boldsymbol{c}_{i+2}) = \cdots = L(\boldsymbol{c}_{j-2}) = L(\boldsymbol{c}_j), \tag{B.2}$$

which contradicts the given condition of the lemma that $L(\boldsymbol{c}_{j-2}) \neq L(\boldsymbol{c}_j)$. Hence the hypothesis of (B.1) is invalid.

Case II: $j - i$ is odd.

It follows from the definition of φ_{w_k}, (4.3.1) and (B.1) that

$$\boldsymbol{c}_{j-1} \in C_{w_k}(\boldsymbol{c}_j) = C_{w_k}(\boldsymbol{c}_i). \tag{B.3}$$

(B.3) and (4.3.1) imply that

$$L(\boldsymbol{c}_{i+1}) \leq L(\boldsymbol{c}_{j-1}). \tag{B.4}$$

From (B.4) and (4.3.3) of Lemma 4.1, we have $L(\boldsymbol{c}_i) \geq L(\boldsymbol{c}_{i+2}) \geq \cdots \geq L(\boldsymbol{c}_{j-1}) \geq L(\boldsymbol{c}_{i+1}) \geq \cdots \geq L(\boldsymbol{c}_{j-2}) \geq L(\boldsymbol{c}_j)$. Since

$$L(\boldsymbol{c}_i) = L(\boldsymbol{c}_j),$$

then

$$L(\boldsymbol{c}_{j-2}) = L(\boldsymbol{c}_j),$$

which again contradicts the given condition of the lemma that $L(\boldsymbol{c}_{j-2}) \neq L(\boldsymbol{c}_j)$. Therefore, the hypothesis of (B.1) is again invalid.