



**Karlsruhe Institute of Technology**  
Communications Engineering Lab  
Prof. Dr.-Ing. Laurent Schmalen



# Realization of Channel Decoding Using Optimization Techniques

Bachelor Thesis

**Yanxia Lu**

Advisor : Prof. Dr.-Ing. Laurent Schmalen  
Supervisor : Dr.-Ing. Holger Jäkel

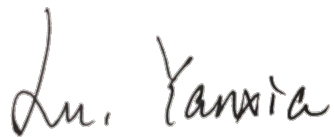
Start date : 15.06.2022  
End date : 15.12.2022



# Declaration

With this statement I declare that I have independently completed the above bachelor's thesis. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis was not previously submitted to another academic institution and has also not yet been published.

Karlsruhe, 15.12.2022

A handwritten signature in black ink that reads "Lu. Yanxia". The signature is written in a cursive style with a large initial 'L'.

Yanxia Lu





## **Acknowledgment**

First of all, I'm very grateful to Prof. Dr.-Ing. Laurent Schmalen for granting me the opportunity to write a bachelor thesis in the institute of Communications Engineering Lab (CEL).

I would like to express my deepest gratitude to my supervisor Dr.-Ing. Holger Jäkel for his patience and support the whole time. He has generously provided me his knowledge and expertise as well as timely feedback and constructive suggestions.

Many thanks should go to the kind colleagues and classmates in CEL, who have offered me help and created a warm working environment. My sincere thanks goes of course also to the institute, where I am provided with a laptop and convenience to work there.

Lastly, I would like to thank my family for their support as well.



# Contents

<b>1. Introduction</b>	<b>5</b>
<b>2. Background</b>	<b>7</b>
2.1. Key Concepts and Notations	7
2.1.1. Binary Linear Block Codes	7
2.1.2. BPSK	8
2.1.3. Channel model	8
2.1.4. MAP and ML Decoding	9
2.2. LDPC Codes	10
2.3. A Brief Introduction of Popular Decoding Algorithms	11
<b>3. Decoding with Optimization Techniques</b>	<b>15</b>
3.1. ML Decoding as Mathematical Optimization Problems	15
3.1.1. Linear Code Linear Program	15
3.1.2. LP Formulation of ML Decoding	16
3.1.3. Decoding with Optimization Problem Solvers	19
3.1.4. Motivation	20
3.2. Interior Point Decoding	21
3.2.1. Formulation of Optimization Problem in IP Decoding	21
3.2.2. Minimization with Gradient Descent	24
3.2.3. Minimization with Newton Method	25
3.2.4. interior point (IP) Decoding Algorithm	25
3.2.5. Internal Decoder	27
3.3. Proximal Decoding	28
3.3.1. Optimization Problem	28
3.3.2. Proximal Gradient Method	30
3.3.3. Proximal Decoding Algorithm	31
<b>4. Simulation Results</b>	<b>33</b>
4.1. Interior Point Decoding Simulation Results and Analysis	33
4.1.1. Validation and Analysis	33
4.1.2. Influence of Scaling Factor $t$	35
4.1.3. Iterations Required to Converge to Valid Codewords	37
4.1.4. IP + sum-product algorithm (SPA) Vs. IP + min-sum algorithm (MSA)	39
4.1.5. Approximate Gradient Descent Vs. Newton Method with Approximate Hessian	47
4.1.6. Experimental Study on the Performance Improvement	50
4.2. Proximal Decoding Simulation Results and Analysis	56
4.2.1. Comparison of Results	56
4.2.2. Influence of $\omega$ and $\gamma$	58
4.2.3. Iterations Required to Converge to Valid Codewords	67
4.2.4. Experimental Study on the Performance Improvement	71
<b>5. Conclusion</b>	<b>75</b>

<b>A. Abbreviations</b>	<b>77</b>
<b>Bibliography</b>	<b>79</b>

# 1. Introduction

This thesis focuses on decoding binary linear codes with optimization techniques. Among all kinds of linear codes, low-density parity-check (LDPC) codes are considered capacity-approaching codes and have been widely applied to communications. Algorithms based on the message-passing (MP) concept, namely the belief propagation (BP) algorithms, have been mostly used to decode LDPC codes. However, another approach grounded on optimization methods has also shown competitive performance. Linear programming (LP) decoding [FWK05] that approximates maximum likelihood (ML) decoding was proposed. The LP relaxation from LP decoding has been further studied in pursuit of reducing complexity and improving performance. Inspired by LP decoding, other algorithms based on convex optimization have also been suggested. To explore possibilities of decoding with optimization methods, two algorithms are primarily introduced and implemented, which employ well-established convex optimization methods, namely the interior point (IP) method [Wad07, Wad08, Wad10] and the proximal gradient method [WT21, WT22].

The thesis starts with theoretical background including notations and important concepts. LDPC codes as well as popular algorithms for decoding LDPC codes are briefly introduced. In the third chapter, decoding algorithms with optimization techniques are elaborated. The LP relaxation of ML decoding from LP decoding is presented in detail, for it represents the fundamentals of transforming ML decoding to optimization problems. This LP relaxation is furthermore used in many decoding algorithms using optimization techniques. Next, a literature review regarding the topic of this thesis is given. At the end of chapter 3, formulations and structure of IP decoding as well as proximal decoding are presented. Chapter 4 consists of simulation results from the implementation of IP decoding and proximal decoding. The influence of important parameters is investigated. Decoding performance is analyzed and discussed in comparison with the results from previous studies, which is followed by suggestions for improvement. Finally, a conclusion is given to summarize the whole thesis.



## 2. Background

This chapter introduces the preliminaries of this thesis, including essential concepts, the channel model, as well as the code class that is mainly involved. Last but not least, a literature review of popular decoding algorithms is presented.

### 2.1. Key Concepts and Notations

The objective of this section is to review key concepts and notations used throughout the thesis. This work focuses on channel decoding in digital signal transmission. To be more specific, it deals with decoding methods which can produce a good approximation of transmitted symbols and afterwards an estimation of transmitted codewords, regardless of the noise from the channel. Therefore, referring to the general framework for digital communications in [Moo20], the system model is simplified to an end-to-end transmission model as shown in Figure 2.1.

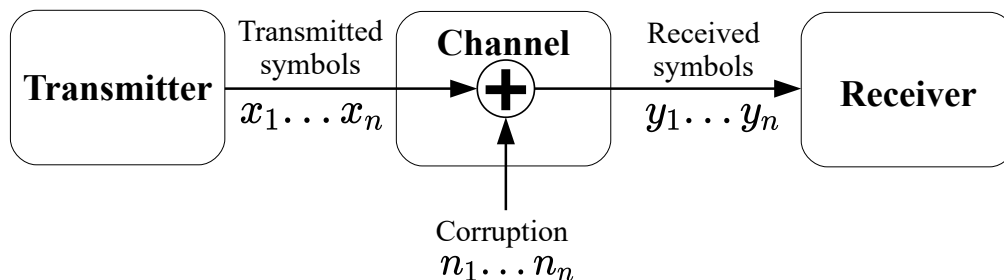


Figure 2.1: System Model

The following parts clarify preliminaries of this thesis in terms of code and modulation scheme as well as channel model.

#### 2.1.1. Binary Linear Block Codes

A binary linear  $(n, k)$  block code is defined over binary field  $\mathbb{F}_2$  with code length  $n$ , code dimension  $k$  and  $2^k$  codewords  $\mathbf{c} \in \mathbb{F}_2^n$ . The code rate is  $R = k/n$ . Moreover, the code is closed under linear operations, which means the linear combination of two codewords is also a codeword [Moo20]. In block coding, a block or frame of  $k$  information bits is encoded by channel encoder into a codeword  $\mathbf{c}$  consisting of  $n$  bits, with  $n > k$ . The  $n - k$  redundant bits are introduced for error detection and correction in the decoding process.

For linear codes, information for decoding is included in a binary  $(n - k) \times n$  parity-check matrix  $\mathbf{H}$ . A valid codeword satisfies the condition  $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}^T$ . The set of all valid codewords is denoted by  $\mathcal{C}$ , where

$$\mathcal{C} = \left\{ \mathbf{c} \in \mathbb{F}_2^n \mid \mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}^T \right\}. \quad (2.1)$$

The estimated codeword is denoted by  $\hat{\mathbf{c}}$ . The decoding performance will be measured by bit error rate (BER) and frame error rate (FER), which are defined as:

$$\text{FER} = P(\hat{\mathbf{c}} \neq \mathbf{c}), \quad (2.2)$$

and

$$\text{BER} = P(\hat{c}_j \neq c_j), j = 1, 2, \dots, n, \quad (2.3)$$

respectively. In this thesis, they will be calculated in the following way:

- FER= number of wrongly decoded frames / total number of frames.
- BER= number of wrongly decoded bits / total number of bits.

### 2.1.2. BPSK

Before transmission, the codeword  $\mathbf{c} \in \mathbb{F}_2^n$  is mapped to a vector of modulation symbols, denoted by  $\mathbf{x} \in \mathbb{R}^n$ . The simulations in this work are based on binary phase-shift keying (BPSK) modulation scheme. The mapping from bits to modulation symbols is  $0 \rightarrow 1$  and  $1 \rightarrow -1$ , therefore it holds,

$$\mathbf{x} = 1 - 2\mathbf{c} \in \{1, -1\}^n \quad (2.4)$$

Let  $\mathcal{X}$  denote an assemble including all  $\mathbf{x}$  that are subject to this linear mapping:

$$\mathcal{X} = \{1 - 2\mathbf{c} \mid \mathbf{c} \in \mathcal{C}\}. \quad (2.5)$$

To imply the relation between  $\mathbf{x}$  and  $\mathbf{c}$ ,  $\mathbf{x}(\mathbf{c})$  is also used throughout this thesis.

### 2.1.3. Channel model

Memoryless binary additive white Gaussian noise (AWGN) is used as channel model. It is simple to implement in the first place. Secondly, as a classical channel model, it has been applied in a few studies that are in consistency with this thesis. In these studies, simulation results for AWGN channel were published for reference. Assuming that a matched filter is applied to received signals, the communication channel model can be expressed with vector  $\mathbf{x}(\mathbf{c})$  and random process  $\mathbf{n}$  [Bos13] as:

$$\mathbf{y} = \mathbf{x}(\mathbf{c}) + \mathbf{n}, \mathbf{y} \in \mathbb{R}^n. \quad (2.6)$$

In an AWGN channel, the noise is denoted by random variable  $\mathbf{n}$  that is Gaussian distributed with mean zero. The variance per noise sample is equal to  $\sigma^2$ . To summarize,

$$n_i \sim \mathcal{N}(0, \sigma^2),$$



$$\sigma^2 = \frac{N_0}{2}, \quad (2.7)$$

where  $N_0$  denotes noise power spectral density [RU08, Moo20]. Because for BPSK the modulated signals are real, the noise energy for one dimension is  $\frac{N_0}{2}$  in normalized unit bandwidth. Assumed that the average energy per transmitted symbol is  $E_s$  and average energy pro information bit is  $E_b$ , the relation between  $E_s$  and  $E_b$  is [RU08]:

$$kE_b = nE_s \quad (2.8)$$

Therefore, the (bit) signal-to-noise ratio (SNR) with BPSK modulation is given as follows:

$$\frac{E_b}{N_0} = \frac{E_s}{\frac{k}{n} \cdot 2\sigma^2} \quad (2.9)$$

With the convention of normalized unit symbol power  $E_s = 1$  [RU08], the variance of noise can be described in dependence on SNR:

$$\sigma^2 = \frac{E_s}{2 \cdot \frac{k}{n} \cdot \frac{E_b}{N_0}} = \frac{1}{2 \cdot \frac{k}{n} \cdot \frac{E_b}{N_0}} \quad (2.10)$$

Therefore, given a fixed code rate, the variance of noise  $\sigma^2$  is determined by  $E_b/N_0$ . After going through the channel, the received corrupted vector  $\mathbf{y}$  is then detected. This thesis concentrates on the last step of signal processing, i.e., the estimation of transmitted codeword, namely  $\hat{\mathbf{c}}$ , before mapping  $n$  code bits back to  $k$  information bits.

#### 2.1.4. MAP and ML Decoding

Given the transmitted vector  $\mathbf{x}$  and received vector  $\mathbf{y}$  under the premise of binary AWGN channel with BPSK modulation, conditional density  $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$  is the so-called likelihood function [Moo20], which is used to make a decision in decoding problems. In reference to [RU08, Moo20], an estimated signal  $\hat{\mathbf{x}}$  with minimum block error probability corresponds to the code with a maximum a posteriori probability (APP), which is also named maximum a posteriori (MAP) decoding rule. In summary, MAP decoding finds

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \mathcal{X}} P(\mathbf{x}|\mathbf{y}). \quad (2.11)$$

According to Bayes' theorem, this rule is equal to [Moo20]

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) P(\mathbf{x}). \quad (2.12)$$

Assumed that each modulated codeword  $\mathbf{x} \in \mathcal{X}$  is transmitted with the same probability,  $P(\mathbf{x})$  doesn't depend on  $\mathbf{x}$ . Thus, as shown in [RU08, Moo20]:

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \hat{\mathbf{x}}^{\text{ML}} \quad (2.13)$$

This decision rule can also be characterized by log-likelihood ratio (LLR). According to [RU08, Moo20], the LLR is defined as

$$l(y_j) = \ln \frac{p_{Y|X}(y_j|x_j = 1)}{p_{Y|X}(y_j|x_j = -1)}, \quad j = 1, 2, \dots, n. \quad (2.14)$$

The LLR can be transformed to  $\frac{2y}{\sigma^2}$  and be treated as soft information whereas  $\text{sign}(l(\mathbf{y}))$  indicates the hard information in decoding [Moo20]. Furthermore, to make hard decisions (HD) is to exploit hard information as a means to yield bit values. To be more specific, the decoder decides between 0 and 1 by comparing the value of received signals with a threshold [VL98]. On the other hand, a decoder that makes soft decisions (SD) does not rush to a binary decision but returns an erasure or possibility information [VL98].

## 2.2. LDPC Codes

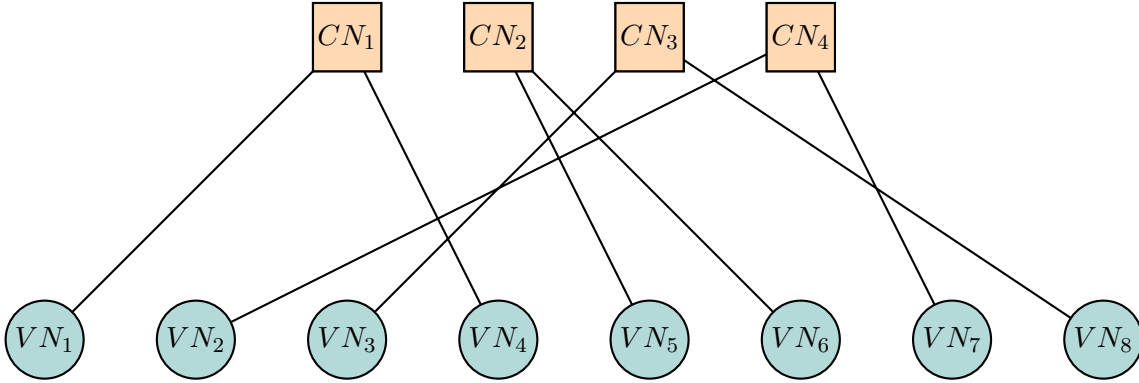
LDPC codes are widely applied in communication standards, for instance, in IEEE standard 802.11ax-2021 [IEE21] and DVB-S2 [W<sup>+</sup>09]. Due to their practical value, LDPC codes have become a popular topic in research. IEEE Xplore [Xpl] shows over 10,000 searching results with keyword of "Low density parity check" or "LDPC" ranging from 1962 till today, including over 2,000 journals and over 7,000 conference papers. Among these studies, there are over 7,000 results with an extra keyword "decoding". Moreover, under the topic of this thesis, LDPC codes are mostly used for the demonstration of decoding performance [FWK05, Wad07, Wad10, WT21]. For a good comparison with references, this thesis also takes LDPC codes as an example of linear block codes to conduct simulations.

LDPC codes are binary linear block code proposed by Gallager in 1962 [Gal62]. They are specified by a sparse parity-check matrix  $\mathbf{H}$  that is comprised of mostly 0's and only a small fixed number of 1's. LDPC codes are proved to be the so-called capacity achieving codes that closely approach the channel capacity [Moo20, Bos13, Joh06]. Spielman has pointed out that with increasing code length towards infinite, LDPC codes could approach the Shannon limit with linear decoding complexity [CF07, SS96, Spi96]. However, LDPC codes were neglected for more than 30 years until David J.C. MacKay and Radford M. Neal rediscovered Gallager's work in 1996 [MN96]. At that time Turbo codes were not long ago officially presented in 1993 [BGT93]. Both have shown near Shannon-limit performance [MN96, BGT93, CFRU01] and both play an important role in wireless communications. LDPC codes outperform turbo codes in regard to error floor and performance with lower complexity in the range of higher code rate [Mac99, HDAES12, TSR17].

Nowadays the codes invented by Gallager [Gal62] are referred as  $(\omega_c, \omega_r)$ -regular LDPC codes, because the number of 1's in each row  $\omega_r$  and that of each column  $\omega_c$  of  $\mathbf{H}$  matrix are constant. Otherwise, they are called irregular LDPC codes. Irregular LDPC codes with improved performance were introduced by Luby et al. in [LMS<sup>+</sup>97, LASMS98, LMSS98] and later studied in [LMSS01, RSU01]. Analogous to binary regular LDPC codes, non-binary regular LDPC codes were afterwards invented [DM98], which showed promising empirical performance for the Gaussian channel, but are computationally expensive to decode [FU98]. This thesis focuses only on decoding binary LDPC codes, since the algorithms for non-binary LDPC codes are often extended from or inspired by those for binary LDPC codes. Without specific notation, LDPC codes in this work are meant to be binary.

An LDPC code can be visually presented by a subclass of bipartite graphs, namely the Tanner graph [Tan81]. It contains two connected node types that are in a one-to-one correspondence with columns and rows of parity-check matrix  $\mathbf{H}$  [Tan81, Wib96, KFL01]. To be more specific, the vertices of one type are named variable nodes (VNs), which

$$\mathbf{H} = \begin{pmatrix} \textcircled{1} & 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \textcircled{1} & \textcircled{1} & 0 & 0 \\ 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & \textcircled{1} \\ 0 & \textcircled{1} & 0 & 0 & 0 & 0 & \textcircled{1} & 0 \end{pmatrix} \begin{matrix} \text{CN}_1 \\ \text{CN}_2 \\ \text{CN}_3 \\ \text{CN}_4 \end{matrix}$$



**Figure 2.2:** An Example of Tanner Graph

correspond to columns of  $\mathbf{H}$ . Each VN represents a bit or symbol of a codeword. The vertices of another type named check nodes (CNs) match the rows of  $\mathbf{H}$ , respectively. In order to meet the condition  $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}^T$ , the sum of all bits connected to each CN must be zero. Therefore, each CN characterizes a constraint function for parity check. There are in total  $n$  VNs and  $n - k$  CNs. VNs and CNs are connected through edges, which correspond to the position of 1's in  $\mathbf{H}$ . An example of regular  $\mathbf{H}$  matrix together with its corresponding Tanner graph is shown in Figure 2.2 for a better illustration. In this demonstration, the number of 1's in each row is 2, specifying the row weight ( $\omega_r$ ) of  $\mathbf{H}$ . Meanwhile, this number indicates the node degree  $d_C$  of each CN. The relation between column weight ( $\omega_c$ ) and VNs follows the same pattern. In a word, each VN has degree  $d_V = \omega_c$ , whereas each CN has degree  $d_C = \omega_r$ . In this way, the Tanner graph is able to provide information of the codes on behalf of  $\mathbf{H}$  matrix at the level of graph theory, which is beneficial to decoding. For example, the graphical representation of the codes is crucial for message-passing algorithms, that are to be mentioned in the next section.

### 2.3. A Brief Introduction of Popular Decoding Algorithms

In the thesis [Gal62], Gallager also presciently put forward with a bit-flipping algorithm and additionally an iterative MP algorithm with the use of APP, which is also called probabilistic decoding, for decoding binary regular LDPC codes. This MP decoding algorithm is the predecessor of iterative decoding [WLK95, HOP96] and a particular instance of later introduced sum-product algorithm (SPA) [Tan81], also known as BP [Pea88, Wib96, MN96]. Similar ideas with the use of APP also appeared in Massey's work in 1963 for APP decoding and threshold decoding [Mas63].

Tanner [Tan81] has developed SPA (or later called BP [Pea88]) and its approximation min-sum algorithm (MSA) at the cost of performance degradation. They are recommended as decoding algorithms for "codes on graphs" [Wib96, KFL01, Big05] such as LDPC codes. As mentioned above, they all trace back to the iterative MP algorithm introduced in Gallager's work [Gal62]. Richardson and Urbanke have very well explained in their book about the relationship between these algorithms under different names [RU08]: "iterative decoding is a generic term referring to decoding algorithms which proceed in iterations. A subclass of iterative algorithms are message-passing algorithms. Message-passing algorithms are iterative algorithms which obey the message-passing paradigm." In summary, one uses MP algorithms to decode codes that can be defined on graphs and BP algorithms are a significant representation of message-passing algorithms.

After reviewing the origin of BP algorithms, below is an introduction of how they work. One iteration of a typical BP decoder, after initializing VN-to-CN message with the a priori probability information of each VN, includes the following processes [DFB14]:

- Update CNs with VN-to-CN message. The message from  $VN_j$  ( $j = 1, 2, \dots, n$ ) to a CN consists of reliability information of all the VNs that are connected to this CN except for  $VN_j$  itself.
- Update VNs with CN-to-VN message along with the initialized a posteriori information. Analogously, the CN-to-VN message from  $CN_j$  ( $j = 1, 2, \dots, n$ ) to a VN includes reliability information of all CNs that are connected to this VN except for  $CN_j$  itself.
- Get the posteriori information of each VN.
- Decide according to the sign of the posterior information a sequence  $\hat{\mathbf{x}} \in \{+1, -1\}^n$  and map it to an estimated codeword  $\hat{\mathbf{c}} \in \{0, 1\}^n$ . Stop the loop if  $\hat{\mathbf{c}}$  is a valid codeword or if the number of iterations has reached a pre-set maximum value. Otherwise, the algorithm will go into the next iteration.

The performance of BP algorithms results from their association with MAP decoding process. BP algorithms can optimally decode codes defined by cycle-free tanner graphs like an MAP decoder. However, under the condition with cycles, BP algorithms are suboptimal but still provide an excellent approximation of the MAP decoder [Wib96, RU08, DFB14].

Although standard BP algorithms have shown excellent decoding performance near Shannon limit [RU01], their computational complexity has made them less practical in implementation [CF02a]. As stated in [RU08], "the ML decision problem for the binary symmetric channel (BSC) is NP-complete". Considering the unfeasible task of implementing the ML decoder for actual applications, it is more pragmatic to develop algorithms which can achieve a satisfying balance between complexity and performance. In addition, standard BP algorithms also have disadvantages of numerical instability [DFB14] and being sensitive to the quantization effect [PL00]. However, BP has inspired a great many studies in direction of reducing computational complexity while more or less maintaining BP performance. A common approximation technique of standard BP is to use logarithms of probabilities or LLR rather than probabilities themselves [RVH95, PL00, HEAD01, CF02a, CDE<sup>+</sup>05]. It simplifies multiplications to additions and helps to solve the problem of numerical instability. There are numerous BP or MSA based studies in decoding LDPC codes, as in [FM97, FMI99, CRU01, CF02b, SLG04, ZZB05, JZZX06, PSE<sup>+</sup>07, HHW10, ISFR11, BSB14], to name a few.

Algorithms born from BP decoding have dominated the field of decoding for a while until another new approach increasingly gained acknowledgment. Unlike MP algorithms that rely mainly on iterative exchange of probability information between symbols, this approach is to apply mathematical optimization theory to channel decoding. The concept is to exploit distance and constraints defined in codes and to transform them into optimization problems. For example, Feldman et al. [FWK05] have made prominent contribution by proposing LP decoding. They convert ML decoding into an LP problem, which can be solved by various optimization techniques. Moreover, realizing the potential of this approach, more studies have come into existence expanding the domain from linear to nonlinear and convex problems. These options are competitive alternatives to BP and BP-based algorithms. More will be elaborated in Chapter 3.



## 3. Decoding with Optimization Techniques

As mentioned before, there emerged a new method different from, however related to, BP family, called LP decoding. It was established from an LP relaxation of the optimization problem underlying the decoding process. An LP decoder is an approximate ML decoder suggested by Feldman et al. [FKW03, Fel03]. It is best known for arbitrary binary linear code transmitted through discrete memoryless symmetric channel [FWK05]. This novel idea in channel decoding has motivated decoding algorithms with various optimization techniques. The advantage of this approach arises from the fact that the framework of optimization theory provides a powerful analytical tool to help understand and analyze the performance of finite-length codes [FWK03]. This chapter elaborates on the theoretical background of LP decoding which represents the principle of decoding with optimization techniques. Algorithms used to solve this LP problem, known as LP solvers [FWK05, TNS08, BLDR13], along with decoding algorithms with nonlinear or convex optimization methods are briefly reviewed. Furthermore, two algorithms based on convex optimization methods are presented.

### 3.1. ML Decoding as Mathematical Optimization Problems

#### 3.1.1. Linear Code Linear Program

The term LP decoding was proposed by Feldman et al. [FK02] in 2002 and analyzed in [FKW02, EH03, HE05] for turbo-like Repeat-Accumulate (RA) codes, shortly before its application for LDPC codes in work [Fel03]. The generalization of LP for arbitrary binary linear code defined as linear code linear program (LCLP) are then presented and theoretically proved in [FWK03, FKW03, FWK05].

The LP decoder is connected with BP in some ways and empirically shows approaching-BP performance [FWK05]. Under the model of binary erasure channel (BEC), it was shown in [FWK05] that the pseudocodewords defined in LP decoding correspond to "stopping sets" of BP decoders. Therefore, the LP decoder performs the same as a BP decoder on the BEC [FWK05]. LP pseudo-codewords are also associated with "graph covers" defined by Koetter and Vontobel [KV03]. In addition, the LP decoder outperformed MSA for the case of a LDPC code in [FWK05] with ML certificate property. That means, once the LP decoder yields a codeword, the codeword is the ML codeword.

Superior to the analysis for BP (and BP-based) algorithms, an inverse-exponential word error rate (WER) for a constant-rate code has been derived in [FMS<sup>+</sup>07] under LP decoding. The LP decoder can thus correct a constant fraction of errors [FMS<sup>+</sup>07]. Based on the theoretical background of LP, the analytic bounds of error probability and error-correcting performance of LP decoding have been derived and analyzed as in [HE05, KV06, FMS<sup>+</sup>07, DDKW08, ADS12]. Consequently, the performance of LP decoding can be analytically proved to be close to BP algorithms.

All in all, the LP decoding has advantages over BP decoding in terms of finite-length analysis and ML-guaranteed outputs because of its well-structured theoretical background

based on mathematical optimization theory. Therefore, the LP decoding is regarded as an alternative of BP algorithms. It can also be used in combination with BP as in [WJW05].

### 3.1.2. LP Formulation of ML Decoding

The fundamental principle behind LP decoding lies in the LP relaxation [BBV04] to the ML problem in decoding process. As mentioned in Section 2.1.4, an ML decoder finds:

$$\hat{\mathbf{x}}^{\text{ML}} = \arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}). \quad (3.1)$$

Given the preliminaries of binary AWGN channel and BPSK modulation, it applies:

$$\mathbf{c} \in \mathcal{C}, \quad \mathcal{X} \in \{-1, +1\}^n \quad \text{and} \quad \mathbf{y} \in \mathbb{R}^n.$$

Assuming that the input codewords have uniform prior, it holds:

$$\arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}). \quad (3.2)$$

As defined in Eq. (2.14), the LLR vector is  $\mathbf{l}$ . The ML codeword corresponds to the solution of the optimization problem [FWK05]:

$$\begin{aligned} & \text{minimize} \quad \sum_{j=1}^n l_j c_j, \\ & \text{subject to} \quad \mathbf{c} \in \mathcal{C}. \end{aligned} \quad (3.3)$$

Furthermore, Feldman et al. reformulated this LP problem by extending the domain from  $\mathcal{C}$  to a convex polytope [FWK05]. They brought in the concept of "codeword polytope", which is defined by [FWK05]:

$$\text{poly}(\mathcal{C}) = \left\{ \sum_{\mathbf{c} \in \mathcal{C}} \epsilon_{\mathbf{c}} \mathbf{c} : \epsilon_{\mathbf{c}} \geq 0, \sum_{\mathbf{c} \in \mathcal{C}} \epsilon_{\mathbf{c}} = 1 \right\}. \quad (3.4)$$

The  $\text{poly}(\mathcal{C})$  is within the  $n$ -hyper-cube  $[0, 1]^n$  and the vertices of  $\text{poly}(\mathcal{C})$  are exactly codewords [FWK05]. LP obtains its optimum at a vertex [Sch98], which means the solution of this LP corresponds to a ML codeword. The points in  $\text{poly}(\mathcal{C})$  are defined in [FWK05] as  $\mathbf{f} = (f_1, \dots, f_n)$ , where  $f_j = \sum_{\mathbf{c} \in \mathcal{C}} \epsilon_{\mathbf{c}} c_j$ . From the definition we can see that  $f_i$  is no more restricted to integers but a real number i.e.,  $f_j \in [0, 1]$ . This linear inequality constraint is considered as the box constraint [BBV04] in convex optimization.

Now the ML decoding problem in Eq. (3.3) can be formulated in LP with broader constraint [FWK05]:

$$\begin{aligned} & \text{minimize} \quad \sum_{j=1}^n l_j f_j, \\ & \text{subject to} \quad \mathbf{f} \in \text{poly}(\mathcal{C}). \end{aligned} \quad (3.5)$$

Since this LP, equivalent to ML problem, is NP hard [FWK05], Feldman et al. applied an LP relaxation [FWK05] to the exact LP. Instead of observing global constraints, which



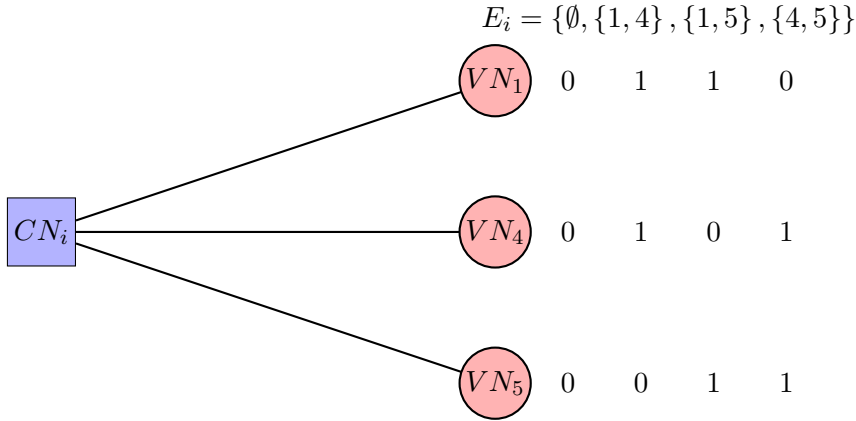
is finite but exponential in the code length  $n$  [FWK05], they exploited parity check at each CN and defined a local code as "the set of binary vectors that have even weight on its neighborhood variables" [FWK05]. For the convenience of explanation, let  $\mathcal{I} = \{1, \dots, n-k\}$  and  $\mathcal{J} = \{1, \dots, n\}$  denote the rows and columns of parity-check matrix  $\mathbf{H}$ , respectively. In addition, a few sets are defined as:

$$\begin{aligned} A(i) &:= \{j | j \in \mathcal{J}, \mathbf{H}_{i,j} = 1\}, \text{ CN index } i \in \mathcal{I}, \\ E_i &:= \{S \subseteq A(i) : |S| \text{ even}\}, \\ B(j) &:= \{i | i \in \mathcal{I}, \mathbf{H}_{i,j} = 1\}, \text{ VN index } j \in \mathcal{J}. \end{aligned} \quad (3.6)$$

Since  $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}^T$  holds for codewords. We know that for  $i \in \mathcal{I}$ , parity check at  $CN_i$  requires

$$\sum_{j \in A(i)} f_j = 0. \quad (3.7)$$

The local code at each CN can be constructed by setting  $f_{j,local} = 1$  if  $j \in S$ ,  $f_{j,local} = 0$  if  $j \in A(i) \setminus S$ , and arbitrary otherwise [FWK05]. These integer codewords are vertices of the local polytope. Analogous to global polytope, the other points in local polytope are the convex combination of local codewords. Because there are even number of elements in  $S$ , it is obvious that every local codeword defined at a CN satisfies the global parity check at this CN. The solution of global LP defined in  $\text{poly}(\mathcal{C})$  is thus the intersection of local solutions defined in local polytopes [FWK05].



**Figure 3.1:** A Simplified Example at  $CN_i$  with Degree 3

Simply for illustration, Fig. 3.1 shows a CN called  $CN_i$  of node degree 3. Its VN neighbors are, for example,  $VN_1$ ,  $VN_4$  and  $VN_5$ . There are only 4 combinations with even weight, i.e.,  $E_i = \{\emptyset, \{1, 4\}, \{1, 5\}, \{4, 5\}\}$ . Under each combination, the value of  $f_1$ ,  $f_4$  and  $f_5$  are formed correspondingly. For instance, when  $S = \{1, 4\}$ , we can set  $f_1 = f_4 = 1$  and  $f_5 = 0$ . The other bits except for  $f_1$ ,  $f_4$  and  $f_5$  can be chosen arbitrarily. Explanation in this example can be slightly different from [FWK05]. For they have required the local codewords to be scaled, and we only focus on the integer solutions for a more intuitive illustration.

Through the relaxation, the number of constraints have been reduced from exponential in code length to exponential in degree of each CN [FWK05], denoted as  $d_C$ . For the

sake of eliminating redundant constraints due to the overlap of local codewords defined at different CNs, authors of [FWK05] have derived the following constraint based on the parity polytope of Jeroslow [Jer75, Yan88].

This constraint requires that  $\forall i \in \mathcal{I}$  and  $S_{\text{odd}} \subseteq A(i)$ , with  $|S_{\text{odd}}|$  an odd number [FWK05],

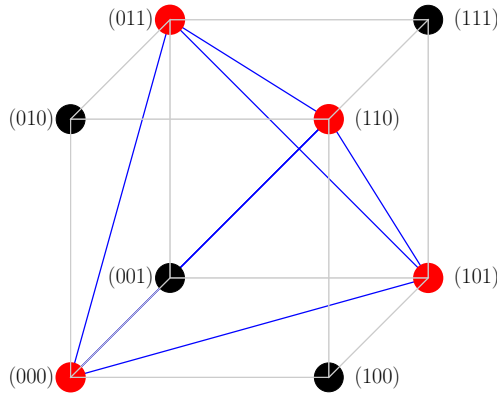
$$\sum_{j \in S_{\text{odd}}} f_j + \sum_{j \in (A(i) \setminus S_{\text{odd}})} (1 - f_j) \leq |A(i)| - 1, \quad (3.8)$$

or equivalently [FWK05],

$$\sum_{j \in S_{\text{odd}}} (1 - f_j) + \sum_{j \in (A(i) \setminus S_{\text{odd}})} f_j \geq 1. \quad (3.9)$$

It is simple to see that this constraint can also be expressed as in [TS06, TNS08]:

$$\sum_{j \in S_{\text{odd}}} f_j - \sum_{j \in (A(i) \setminus S_{\text{odd}})} f_j \leq |S_{\text{odd}}| - 1. \quad (3.10)$$



**Figure 3.2:** The codeword polytope can be represented as convex hull inside the unit hypercube [FWK05].

The conception behind this constraint is to "forbid bad configuration" of  $\mathbf{f}$  [FWK05]. Eq. (3.9) is relatively easy to be understood with the aid of Fig. 3.2 [FWK05]. The VN combinations of a CN of degree 3 are represented as vertices of a unit cube. The forbidden combinations with odd number of 1's are marked black, whereas the valid ones are marked red. Geometrically, this constraint indicates that solution that satisfies parity check lies in the blue convex hull constructed by red valid vertices, whose  $l_1$  distance to the forbidden configurations is at least 1 [FWK05].

This constraint is also referred to as parity constraint [Wad08, Wad10], parity-check constraint [TS06] or parity inequality [TNS08]. It is to be noted that the set  $S_{\text{odd}}$  here is defined to have odd number of elements. Since elaboration hereafter is only about set  $S$  with odd number of elements, we denote simply  $S$  for  $S_{\text{odd}}$ . Along with the box constraint

for each CN, the relaxed LP problem becomes more explicit [FWK05]:

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n l_j f_j, \\
& \text{subject to} && f_j \in [0, 1], \forall j \in \mathcal{J}, \\
& && \sum_{j \in S} (1 - f_j) + \sum_{j \in (A(i) \setminus S)} f_j \geq 1, \forall i \in \mathcal{I}, S \subseteq A(i), |S| \text{ odd}.
\end{aligned} \tag{3.11}$$

### 3.1.3. Decoding with Optimization Problem Solvers

With the LP formulation introduced in Section 3.1.2, decoding algorithms can be developed by implementing the corresponding LP solvers. The complexity of such an LP solver depends on the amount of variables and linear constraints defined on the codeword polytope [FWK05]. The ellipsoid algorithm and the simplex algorithm were considered in [FWK05] as possible LP solvers for LDPC codes, since the complexity of the LP is linear in block length. However, for arbitrary binary linear codes, the complexity of this LP formulation in [FWK05] grows exponentially with the maximum degree of each check node, potentially polynomial in block length [Yan88, FWK05, YWF08], which is unfavorable for codes like high-density parity-check (HDPC) codes. Despite the convenience of analysis, LP decoding has larger computational complexity while the error-correcting performance is not improved when being compared to BP algorithms [YFW06b].

#### Low-complexity LP-based Algorithms

Accordingly, intense studies have been done with focus on achieving higher efficiency of above-mentioned LP problem. Vontobel and Koetter [VK06, VK07] have derived an approximation of dual LP, which can be solved with linear complexity in block length by adopting coordinate-ascent or sub-gradient-based methods. This work has been further studied in [Bur08, Bur09] and extended to non-binary fields in [GB10, GB13, PF10]. In an independent work, Flanagan et al. [FSBG09] have developed a non-binary LP formulation that originates from LP decoding [FWK05]. Apart from that, two new LP formulations with reduced complexity have been proposed respectively in [CS07, YWF08] by exploiting the structure of graphic representation of the codes.

#### Improved LP-based Algorithms with Integer Optimization Methods

Another line of work targets a boost of performance with methods for integer programming, possibly at cost of complexity. Before the LP decoding, it has been proved in [BBLK98] that decoding linear block codes can be treated as an integer optimization problem and solved by an Omura-type algorithm, which is a suboptimal modification of the conventional LP solver, namely the simplex method [Sch98]. The success of LP decoding has encouraged further discussions in applying integer programming techniques such as cutting plane methods and branch-and-bound methods to the LP formulation from LP decoding [FWK05] that approximates ML decoding.

Facet guessing algorithms that remove fractional pseudocodewords have been proposed in [DGW09]. Another algorithm called adaptive linear programming (ALP) based on cutting plane method has been developed by Taghavi et al. [TS06, TNS08] involving an adaptive addition of violated constraints (also called cuts) for the sake of reducing the size of the LP problem to the extent of being independent of the node degree [TS06, TNS08]. The error rate can be improved by generating cuts from redundant parity checks (RPC) [TS06, TNS08]. It has been adapted to an ML decoder in [DYW07] by adding integer constraints. The ALP decoding has been modified in [TSS08, TSS11] and further improved in [ZS11, ZS12]. Other variations of ALP with effective cut-search algorithms, which are applicable to HDPC codes, were presented in [TRH<sup>+</sup>08, TRH<sup>+</sup>09, TRH<sup>+</sup>10] showing better FER than LP and BP decoding for certain tested BCH and LDPC codes. An application of modified ALP decoders to HDPC codes has been demonstrated in [YLB11]. Besides, multistage LP decoding based on branch-and-bound methods has been investigated in [YFW06a, ZM11] for better error-correcting performance.

### Nonlinear and Convex Optimization Approaches

Yang et al. [YFW06a] proposed a box-constrained quadratic programming decoder for LDPC codes of short or medium size with less complexity than the BP decoder. A few researches on IP methods have been conducted. Vontobel has in his work [Von08] discussed about efficient implementations of IP algorithms for LP decoding. Wadayama [Wad07, Wad08, Wad10] suggested the suboptimal IP decoding, in which the LP decoding was relaxed to a convex optimization problem. Taghavi et al. [TSS11] implemented a class of interior-point method with primal-dual path-following method to solve a sequence of linear programs in LP decoding, whereas Wadayama [Wad09] realized an LP decoder with primal path-following method. Recently, an algorithm called proximal decoding has been presented by Wadayama et al. [WT21], using one of proximal gradient methods to perform an approximate MAP decoding. Another method for convex optimization, known as alternating direction method of multipliers (ADMM) [BPC<sup>+</sup>11], has also been actively applied in various ADMM-based algorithms [BLDR13, ZS13, JWMC15, WJM15, DGK<sup>+</sup>16, LD16, JMG16, WB21, JLMH21] to solve decoding problems.

#### 3.1.4. Motivation

To get a better understanding of decoding with optimization techniques, this thesis will elaborate on and implement the IP decoding [Wad07, Wad08, Wad10] as well as the proximal decoding [Wad09]. There are many reasons that these two haven't been chosen. Firstly, with the intention to investigate more than one method on codes with moderate length with limited time and resources, low complexity was prioritized when there must be a trade-off between complexity and performance. IP decoding and proximal decoding stand out for their simple formulation and feasible complexity. Both have linear complexity in code length  $n$  in each iteration regarding LDPC codes [Wad10, WT21], which is more practical compared to other available algorithms. Secondly, there are not as many studies on decoding with convex or nonlinear optimization methods as those established on LP. IP methods and the proximal gradient methods are well-known methods for convex optimization [BBV04, PB<sup>+</sup>14], yet represent a relatively new perspective for decoding.

Ultimately, in addition to clear formulations and simulation results for reference based on AWGN channel, they provide detailed pseudocode and configurations, which are important for comparing results.

Unfortunately, investigation on other methods was not possible due to limited time for the bachelor thesis. For the reference of further studies on this topic, a few other methods that have been demonstrated on binary LDPC codes are also quite attractive with affordable complexity, especially the decomposition methods such as ADMM [BPC<sup>+</sup>11, BLDR13, ZS13, JWMC15, WJM15, DGK<sup>+</sup>16, LD16, JMG16, WB21, JLMH21]. Besides, other IP algorithms, for instance, with primal-dual path-following method [TSS08, TSS11] and the ALP [TS06, TNS08] decoding are also good options for investigation.

## 3.2. Interior Point Decoding

In [Wad07, Wad08, Wad10], Wadayama presented IP decoding for linear vector channels applying a kind of IP methods, namely with the barrier method [BBV04]. This is an approximate decoding algorithm with low complexity and suboptimal performance. It has shown us the possibility of applying convex optimization methods to decoding problems. The idea is to construct a convex "merit function" [Wad10] which is composed of the objective function and the barrier function. The intention is to convert the LP problem with inequality constraints derived in [FWK05] into an unconstrained convex optimization problem, specifically a convex minimization problem.

### 3.2.1. Formulation of Optimization Problem in IP Decoding

The objective function of a linear vector channel is defined in [Wad10] as the squared Euclidean distance between the received and the transmitted vectors:

$$f(\mathbf{c}) := \|\mathbf{y} - (\mathbf{A}\mathbf{c} + \mathbf{b})\|^2, \quad (3.12)$$

where  $\mathbf{A}$  is an interference matrix and  $\mathbf{b}$  is an offset vector of the channel,  $\mathbf{c} \in \mathcal{C}$ . Given the premises of this thesis, the presented model is simplified to:

$$f(\mathbf{c}) := \|\mathbf{y} - \mathbf{x}(\mathbf{c})\|^2. \quad (3.13)$$

As stated in [Moo20] the ML estimation over AWGN channel is the  $\hat{\mathbf{x}}$  with minimum squared Euclidean distance. From this point of view, the objective function helps to approach the ML word.

Additionally, the barrier function is designed to restrain the solutions inside  $\text{poly}(\mathcal{C})$ . All the points of the relaxed polytope meet the condition in Eq. (3.11). Therefore the interior set [Wad10] satisfies the constraints without the equality sign. Points inside relaxed  $\text{poly}(\mathcal{C})$ , i.e., the interior points, denoted by  $\tilde{\mathbf{c}}$ , satisfy box constraint [Wad10]

$$0 < \tilde{c}_j < 1, \quad \forall j \in \mathcal{J} \quad (3.14)$$

as well as parity constraint [Wad10]

$$1 + \sum_{j \in S} (\tilde{c}_j - 1) - \sum_{j \in A(i) \setminus S} \tilde{c}_j < 0, \quad \forall i \in \mathcal{I}, S \subseteq A(i), |S| \text{ odd}, \quad (3.15)$$

In contrast to  $E_i$  in Eq. (3.6), the set including the subsets of odd size in  $A(i)$  is defined [Wad10] as:

$$T_i := \{S \subseteq A(i) : |S| \text{ odd}\}.$$

Through the characteristics of the logarithm function, the convex, differentiable barrier function containing both constraints is established in [Wad10] as:

$$\begin{aligned} B(\tilde{\mathbf{c}}) := & - \sum_{i \in \mathcal{I}} \sum_{S \in T_i} \ln \left[ - \left( 1 + \sum_{j \in S} (\tilde{c}_j - 1) - \sum_{j \in A(i) \setminus S} \tilde{c}_j \right) \right] \\ & - \sum_{j \in \mathcal{J}} \ln[-(-\tilde{c}_j)] - \sum_{j \in \mathcal{J}} \ln[-(\tilde{c}_j - 1)]. \end{aligned} \quad (3.16)$$

When any constraint in the barrier function is violated, the "barrier" would be an infinite value. The merit function that combines the objective function and the barrier function is given by [Wad10]:

$$\psi(\tilde{\mathbf{c}}) := t f(\tilde{\mathbf{c}}) + B(\tilde{\mathbf{c}}), \quad (3.17)$$

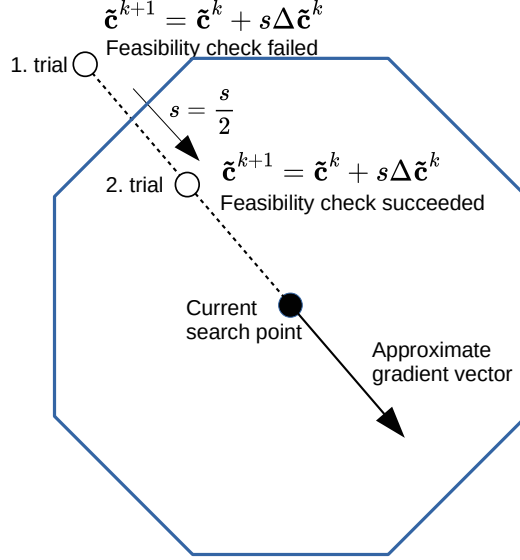
where  $t$  is a constant and  $t > 1$ . Given the merit function, the LP problem in Eq. (3.11), namely

$$\begin{aligned} & \text{minimize} && \sum_{j=1}^n l_j f_j, \\ & \text{subject to} && f_j \in [0, 1], \forall j \in \mathcal{J}, \\ & && \sum_{j \in S} (1 - f_j) + \sum_{j \in (A(i) \setminus S)} f_j \geq 1, \forall i \in \mathcal{I}, S \subseteq A(i), |S| \text{ odd} \end{aligned} \quad (3.18)$$

is transformed in [Wad10] to

$$\text{minimize} \quad \psi(\tilde{\mathbf{c}}). \quad (3.19)$$

The feasible region of this problem is among interior points. In order to search within this region, a feasibility check [Wad07, Wad08, Wad10] must be conducted along with the update of candidate point. Feasibility check decides whether the updated word  $\tilde{\mathbf{c}}^{k+1}$  is an interior point by checking if the word meets the constrains in Eq. (3.15). Update step size is initialized as  $s_0$ . As demonstrated in Fig. 3.3 [Wad07], as long as the constraints are not met, reduce the step size to its half. The process of searching for a proper step size along the update direction is called line search [BBV04]. A line search for  $s$  has been implemented to make sure that the updated search point is still an interior point while minimizing the overall merit function  $\psi(\tilde{\mathbf{c}})$ . However, this process doesn't evaluate objective function  $f(\tilde{\mathbf{c}})$ . Therefore, it can not be guaranteed that the objective function continuously decreases and the searching direction may not converge towards the optimal solution [Wad10]. But the computational complexity is hence reduced to a great extend [Wad10].



**Figure 3.3:** A Demonstration of Feasibility Check [Wad07]

There are some approximations in this algorithm for the purpose of reducing complexity from polynomial to linear [Wad10]. The feasibility check [Wad10] is one of them. The parity constraint concerns all the possible combinations of  $S$  set. It requires

$$1 + \sum_{j \in S} (\tilde{c}_j - 1) - \sum_{j \in A(i) \setminus S} \tilde{c}_j < 0, \quad \forall i \in \mathcal{I}, S \subseteq A(i), |S| \text{ odd.} \quad (3.20)$$

Therefore checking all the combinations calls for quite some computational effort. Instead, the authors [Wad07, Wad08, Wad10] check only the maximum among all combinations and make sure that

$$\max_{S \in \mathcal{T}_i} \left[ 1 + \sum_{l \in S} (c_l - 1) - \sum_{l \in A(i) \setminus S} c_l \right] < 0. \quad (3.21)$$

The same approximation technique has also been practiced in the minimization process of merit function in the inner loop, which is an essential component of this algorithm and will be discussed in detail next. The minimization problem of merit function  $\psi(\tilde{\mathbf{c}})$  in the inner loop, can be solved by many methods, among which are the gradient descent and Newton method. Because of the polynomial computational complexity in calculating exact gradient and Newton step, the authors of [Wad07, Wad08, Wad10] have suggested the so-called approximate gradient and approximate Hessian with greatly reduced complexity from  $O(2^{\omega_r n})$  to  $O(\omega_r n)$  [Wad07, Wad08, Wad10].

### 3.2.2. Minimization with Gradient Descent

The minimization of  $\psi(\tilde{\mathbf{c}})$  can be solved iteratively with classical normalized gradient descent method [Wad07]:

$$\begin{aligned}\tilde{\mathbf{c}} &= \arg \min_{\tilde{\mathbf{c}} \in \text{poly}(\mathcal{C})} \psi(\tilde{\mathbf{c}}), \\ \tilde{\mathbf{c}}^{k+1} &= \tilde{\mathbf{c}}^k - sK\mathbf{g},\end{aligned}\tag{3.22}$$

where  $\mathbf{g}$  is the exact gradient of the merit function  $\psi(\tilde{\mathbf{c}})$ ,  $s$  is the step size and  $K$  is a normalization constant [Wad07], where

$$K = \frac{1}{\max_{j \in \mathcal{J}} |g_j|}.\tag{3.23}$$

For any  $j \in \mathcal{J}$ , the exact gradient of  $\psi(\tilde{\mathbf{c}})$  is given by [Wad07, Wad08, Wad10] :

$$\begin{aligned}g_j &= t \frac{\partial}{\partial \tilde{c}_j} f(\tilde{\mathbf{c}}) + \frac{\partial}{\partial \tilde{c}_j} B(\tilde{\mathbf{c}}) \\ &= 4t(y_j + 2\tilde{c}_j - 1) + \sum_{i \in \mathcal{I}} \sum_{S \in T_i} \tau_j^{(i,S)}(\tilde{\mathbf{c}}) - \frac{1}{\tilde{c}_j} - \frac{1}{\tilde{c}_j - 1},\end{aligned}\tag{3.24}$$

where

$$\tau_j^{(i,S)}(\tilde{\mathbf{c}}) := \frac{I[j \in A(i) \setminus S] - I[j \in S]}{1 + \sum_{l \in S} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S} \tilde{c}_l},\tag{3.25}$$

and  $I$  is the indicator function.

The approximation is to allow [Wad07]

$$\sum_{S \in T_i} \tau_j^{(i,S)}(\tilde{\mathbf{c}}) \approx \tau_j^{(i,S^{(i)})}(\tilde{\mathbf{c}}),\tag{3.26}$$

where

$$S^{(i)} := \arg \max_{S \in T_i} \left[ 1 + \sum_{l \in S} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S} \tilde{c}_l \right].\tag{3.27}$$

For  $S^{(i)}$  is the combination which maximizes the negative denominator and is therefore more dominant among all the combinations.

The approximate gradient  $g_j^{\text{approx}}$ ,  $\forall j \in \mathcal{J}$  is thus defined in [Wad07, Wad08, Wad10] by:

$$g_j^{\text{approx}} = 4t(y_j + 2\tilde{c}_j - 1) + \sum_{i \in \mathcal{I}} \tau_j^{(i,S^{(i)})}(\tilde{\mathbf{c}}) - \frac{1}{\tilde{c}_j} - \frac{1}{\tilde{c}_j - 1}.\tag{3.28}$$

The normalization constant  $K$  is changed to [Wad07]:

$$K^{\text{approx}} = \frac{1}{\max_{j \in \mathcal{J}} |g_j^{\text{approx}}|}.\tag{3.29}$$

Moreover, the update rule becomes [Wad07]:

$$\tilde{\mathbf{c}}^{k+1} = \tilde{\mathbf{c}}^k - sK^{\text{approx}}\mathbf{g}^{\text{approx}}.\tag{3.30}$$



### 3.2.3. Minimization with Newton Method

Since gradient descent method converges not as fast as Newton method, the algorithm is improved in the articles [Wad08, Wad10] by applying Newton method with approximated Hessian matrix. The minimization of  $\psi(\tilde{\mathbf{c}})$  can also be solved as follows [Wad10]:

$$\begin{aligned}\tilde{\mathbf{c}} &= \arg \min_{\tilde{\mathbf{c}} \in \text{poly}(\mathcal{C})} \psi(\tilde{\mathbf{c}}), \\ \tilde{\mathbf{c}}^{k+1} &= \tilde{\mathbf{c}}^k + s\mathbf{d},\end{aligned}\tag{3.31}$$

where  $\mathbf{d}$  is the Newton step and  $s$  is the step size. With Newton equation [Wad10], we have

$$\mathbf{G}\mathbf{d} = -\mathbf{g},\tag{3.32}$$

where  $\mathbf{G}$  is the exact Hessian matrix and  $\mathbf{g}$  is the exact gradient. The gradient  $\mathbf{g}$  can be obtained as discussed in the last part. To acquire  $\mathbf{d}$ , we need the Hessian matrix. Analogous to approximate gradient, the exact  $\mathbf{G}$  of merit function [Wad10], namely

$$t \frac{\partial^2}{\partial \tilde{c}_p \partial \tilde{c}_q} f(\tilde{\mathbf{c}}) + \sum_{i \in \mathcal{I}} \sum_{S \in T_i} \tau_p^{(i,S)}(\tilde{\mathbf{c}}) \tau_q^{(i,S)}(\tilde{\mathbf{c}}) + I[p=q] \left( \frac{1}{\tilde{c}_p^2} + \frac{1}{(\tilde{c}_p - 1)^2} \right)\tag{3.33}$$

is simplified to  $\mathbf{G}^{\text{approx}}$  [Wad10], namely

$$t \frac{\partial^2}{\partial \tilde{c}_p \partial \tilde{c}_q} f(\tilde{\mathbf{c}}) + I[p=q] \sum_{i \in \mathcal{I}} \tau_p^{(i,S^{(i)})}(\tilde{\mathbf{c}}) \tau_q^{(i,S^{(i)})}(\tilde{\mathbf{c}}) + I[p=q] \left( \frac{1}{\tilde{c}_p^2} + \frac{1}{(\tilde{c}_p - 1)^2} \right),\tag{3.34}$$

which only includes the diagonal elements where  $p = q$  for  $p \in \mathcal{J}$  and  $q \in \mathcal{J}$ .

Applied to AWGN channel, it exists:

$$\mathbf{G}^{\text{approx}} = 8t + I[p=q] \sum_{i \in \mathcal{I}} \tau_p^{(i,S^{(i)})}(\tilde{\mathbf{c}}) \tau_q^{(i,S^{(i)})}(\tilde{\mathbf{c}}) + I[p=q] \left( \frac{1}{\tilde{c}_p^2} + \frac{1}{(\tilde{c}_p - 1)^2} \right).\tag{3.35}$$

With the approximation, there are only diagonal elements left. According to [Wad10], the approximated Newton step  $\mathbf{d}^{\text{approx}}$  can be obtained by solving

$$\mathbf{d}^{\text{approx}} = (\mathbf{G}^{\text{approx}})^{-1} (-\mathbf{g}^{\text{approx}}).\tag{3.36}$$

Therefore, the update rule becomes [Wad10]:

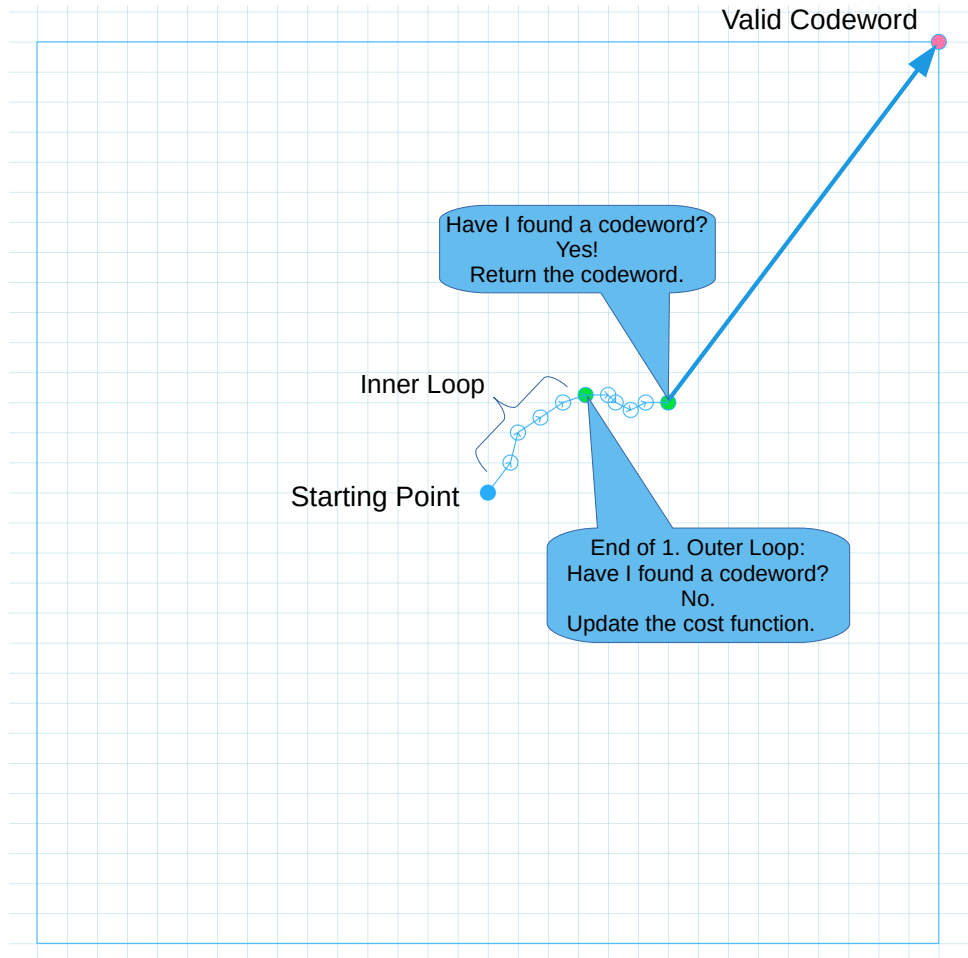
$$\tilde{\mathbf{c}}^{k+1} = \tilde{\mathbf{c}}^k - s (\mathbf{G}^{\text{approx}})^{-1} \mathbf{g}^{\text{approx}}.\tag{3.37}$$

### 3.2.4. IP Decoding Algorithm

The idea of IP decoding is to start with an interior point and to move iteratively towards vertices through minimizing the convex function  $\psi(\tilde{\mathbf{c}})$  while staying inside  $\text{poly}(\mathcal{C})$ . Additionally with the aid of an internal decoder, for example SPA or MSA, the IP decoder can converge to valid codewords.

This algorithm is composed of an outer loop and an inner loop. The loop outside, with maximum  $O_{\text{max}}$  iterations, renews  $\psi(\tilde{\mathbf{c}})$  by increasing the weight of  $f(\tilde{\mathbf{c}})$ , namely parameter

$t$ . In this way, the searching direction is adjusted iteratively towards an estimation of transmitted word that is closest to the received word. To be more specific, increasing factor  $t$  by multiplying initial  $t_0$  with  $\alpha$ , where  $\alpha > 1$ , the weight of the objective function  $f(x)$  grows. As  $t \rightarrow \infty$ , the solution converges to the optimum which minimizes  $f(x)$  [Wad10]. On the other hand, the inner loop, with maximum  $I_{\max}$  iterations, minimizes the overall merit function which includes the box constraint and the parity constraint. In total, there are maximum  $O_{\max} \cdot I_{\max}$  iterations. An illustration of iterations can be found in Fig. 3.4.



**Figure 3.4:** A 2D Illustration of Iterations

The barrier method finds a good solution, however, requires suitable starting point [BBV04]. The authors of [Wad07, Wad10] chose  $\tilde{c}^0 := (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$  of length  $n$  as the starting point. It has been proved to be an interior point in [Wad07, Wad10]. The framework of IP decoding is summarized in Algorithm 1. Details of the implementation are explained in [Wad10].

---

**Algorithm 1** IP Decoding [Wad10]

---

**Input:** received word  $\mathbf{y} \in \mathbb{R}^n$

**Output:** estimated codeword  $\hat{\mathbf{c}} \in \mathbb{F}_2^n$

Step 1: Let starting point  $\tilde{\mathbf{c}}^0 := (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$  and  $t := t_0$ .

Step 2: Repeat the outer loop for  $O_{\max}$  times.

Step 2.1: Repeat the inner loop for  $I_{\max}$  times, with  $k$  as an iteration counter.

Step 2.1.1: Set  $s := s_0$ .

Step 2.1.2: Minimize  $\psi(\tilde{\mathbf{c}})$  via gradient descent or Newton method:

$$\tilde{\mathbf{c}}^{k+1} = \tilde{\mathbf{c}}^k + s\Delta\tilde{\mathbf{c}}.$$

Step 2.1.3: Check feasibility of  $\tilde{\mathbf{c}}^{k+1}$ . If  $\tilde{\mathbf{c}}^{k+1}$  is not an interior point, reduce step size as  $s := \frac{1}{2}s$  and update  $\tilde{\mathbf{c}}^{k+1}$ .

Step 2.2: Run internal decoder for  $L_{\max}$  iterations and let it output  $\hat{\mathbf{c}}$ .

Step 2.3: Exit if  $\mathbf{H} \cdot \hat{\mathbf{c}}^T = \mathbf{0}^T$ .

Step 2.4: Set  $t = \alpha t$ ,  $\alpha > 1$ .

---

### 3.2.5. Internal Decoder

An internal LDPC decoder is placed between inner loop and outer loop which helps to reduce number of iterations and to output codewords [Wad07, Wad10]. The internal decoder makes use of outputs from inner loop to speed up decoding process. This decoder contributes to the delivery of binary outputs without losing too much information as in HD. When the internal decoder converges to a valid codeword, it will break the loops. To be noticed, this decoder gives no outputs back to the loops. BP decoders such as SPA and scaled MSA have been placed inside IP decoder in [Wad07, Wad10]. As briefly introduced in Section 2.3, both SPA and MSA follow the basic message-passing routines between VNs and CNs with slight difference.

1. Initialization [Wad10]:

$$\text{LLR } \lambda_j := \ln \left( \frac{1 - \tilde{c}_j}{\tilde{c}_j} \right).$$

For all the  $(i, j)$  pair, where  $\mathbf{H}_{i,j} = 1$ :  $\xi_{i \rightarrow j} = 0$ ,  $\eta_{j \rightarrow i} = \lambda_j$ .

2. CN update with message from VN [Wad10]:

$$\text{SPA: } \xi_{i \rightarrow j} = 2 \tanh^{-1} \left\{ \prod_{j' \in A(i) \setminus j} \tanh \frac{\eta_{j' \rightarrow i}}{2} \right\}.$$

$$\text{MSA: } \xi_{i \rightarrow j} = \kappa \left( \prod_{j' \in A(i) \setminus j} \text{sign}(\eta_{j' \rightarrow i}) \right) \cdot \min_{j' \in A(i) \setminus j} |\eta_{j' \rightarrow i}|,$$

where  $\kappa$  is a damping factor.

3. VN update with message from CN [Wad10]:

$$\eta_{j \rightarrow i} = \lambda_j + \sum_{i' \in B(j) \setminus i} \xi_{i' \rightarrow j}.$$

4. Get the posterior information [Wad10]:

$$\eta_j = \lambda_j + \sum_{i \in B(j)} \xi_{i \rightarrow j}.$$

5. Make a decision with sign of  $\eta_j$  [Wad10]:

$$\hat{c}_j = \begin{cases} 0, & \eta_j \geq 0, \\ 1, & \eta_j < 0. \end{cases}$$

It is to be noted that the LLR here is not defined conventionally as  $\ln \frac{p[y_j|c_j=0]}{p[y_j|c_j=1]}$ . Since  $\tilde{c}_j \in (0, 1)$ ,  $\tilde{c}_j$  is approximately seen as  $p[y_j|\tilde{c}_j=1]$  [Wad07]. Accordingly,  $p[y_j|\tilde{c}_j=0]$  would be  $1 - \tilde{c}_j$ . Besides, scaling parameter  $\kappa$  is named damping factor [Wad10], set between 0.7 and 0.9 to improve performance of MSA. Other than that, the adopted SPA and MSA are just in standard form.

### 3.3. Proximal Decoding

Proximal decoding was proposed recently in [WT21] for multiple-input multiple-output (MIMO) Channels detection by applying proximal gradient method [PB<sup>+</sup>14]. This innovative decoding algorithm achieves the same time complexity as BP decoders for LDPC codes, namely  $O(n)$  [WT21]. Meanwhile, it showed better performance than the combination of minimum mean square error (MMSE) detector and BP decoder in numerical experiments [WT21]. The framework of proximal decoding is very similar to IP decoding. They both firstly establish a cost function that contains box constraint and parity constraint, which will be minimized with various optimization methods iteratively. Different from IP decoding, a proximal decoder employs a much simpler implementation without extra internal decoder. In this section, an optimization problem based on approximate MAP decoding is formulated. Next, proximal gradient method that was used to solve this problem is reviewed. Last but not least, the algorithm suggested in proximal decoding [WT21] is presented.

#### 3.3.1. Optimization Problem

##### Polynomial Penalty Function

[WT21] introduced a polynomial penalty function in sum-of-squares (SOS) form

$$h(\mathbf{x}) := \sum_{j \in \mathcal{J}} (x_j^2 - 1)^2 + \sum_{i \in \mathcal{I}} \left( \left( \prod_{j \in A(i)} x_j \right) - 1 \right)^2, \quad \mathbf{x} \in \mathbb{R}^n. \quad (3.38)$$

Similar to the barrier function defined in IP decoding [Wad10] and formulated in Eq. (3.16), penalty function  $h(\mathbf{x})$  also contains a term for box constraint [WT21]

$$\sum_{j \in \mathcal{J}} (x_j^2 - 1)^2 \quad (3.39)$$

and a term for parity constraint [WT21]

$$\sum_{i \in \mathcal{I}} \left( \left( \prod_{j \in A(i)} x_j \right) - 1 \right)^2. \quad (3.40)$$

This function transfers the constraints into a minimization problem. For  $\mathbf{x} \in \mathcal{X}$ , both terms are equal to zero. Therefore, penalty function  $h(\mathbf{x})$  is equal to zero for valid codewords and a non-negative value otherwise [WT21].

### Approximate MAP Decoding

As discussed in Section 2.1.4 MAP decoding is formulated as:

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \mathcal{X}} p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) P(\mathbf{x}). \quad (3.41)$$

Assuming that all codewords have equal probability,  $P(\mathbf{x})$  is defined in [WT21] as:

$$P(\mathbf{x}) := \frac{1}{|\mathcal{X}|} \sum_{\mathbf{a} \in \mathcal{C}} \delta(\mathbf{x} - \mathbf{a}), \quad (3.42)$$

where  $\delta$  refers to Dirac delta function. We know that

$$\begin{cases} \forall \mathbf{x} \in \mathcal{X}, & h(\mathbf{x}) = 0, \\ \text{otherwise,} & h(\mathbf{x}) \geq 0. \end{cases}$$

$P(\mathbf{x})$  can thus be approximated as [WT21]:

$$P(\mathbf{x}) = \frac{1}{Z} \exp(-\gamma h(\mathbf{x})), \quad (3.43)$$

where  $Z$  is normalizing constant and  $\gamma > 0$ . When  $\gamma \rightarrow \infty$ , there exists [WT21]:

$$\frac{1}{Z} \exp(-\gamma h(\mathbf{x})) \rightarrow \frac{1}{|\mathcal{X}|} \sum_{\mathbf{a} \in \mathcal{C}} \delta(\mathbf{x} - \mathbf{a}). \quad (3.44)$$

Besides,  $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$  for AWGN channel with BPSK modulation is given by:

$$p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = b_1 \exp\left(-b_2 \|\mathbf{y} - \mathbf{x}\|^2\right), \quad (3.45)$$

where  $b_1$  and  $b_2$  are positive constants. Provided that

$$P(\mathbf{x}) \propto \exp(-\gamma h(\mathbf{x})), \quad (3.46)$$

along with

$$p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) \propto \exp\left(-\|\mathbf{y} - \mathbf{x}\|^2\right). \quad (3.47)$$

the approximate MAP decoding can be formulated as [WT21]:

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \max_{\mathbf{x} \in \mathcal{X}} \exp\left(-\|\mathbf{y} - \mathbf{x}\|^2 - \gamma h(\mathbf{x})\right). \quad (3.48)$$

To maximize probability is equivalent to minimizing its negative log-likelihood. Therefore, the approximate MAP rule from [WT21] for AWGN channel becomes:

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\|\mathbf{y} - \mathbf{x}\|^2 + \gamma h(\mathbf{x})\right). \quad (3.49)$$

Since  $f(\mathbf{x})$  was already used to denote the squared Euclidean distance between the received and the transmitted vectors in IP decoding, this rule can be expressed as:

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg \min_{\mathbf{x} \in \mathcal{X}} (f(\mathbf{x}) + \gamma h(\mathbf{x})). \quad (3.50)$$

### 3.3.2. Proximal Gradient Method

Proximal gradient method assesses the proximal operator of a function [PB<sup>+</sup>14]. The proximal operator of a scaled arbitrary function  $\theta g$  is defined by [PB<sup>+</sup>14]:

$$\mathbf{prox}_{\theta g}(\mathbf{v}) := \arg \min_{\mathbf{x} \in \mathbb{R}^n} \left(g(\mathbf{x}) + \left(\frac{1}{2\theta}\right) \|\mathbf{x} - \mathbf{v}\|_2^2\right), \quad \mathbf{v} \in \mathbb{R}^n, \quad (3.51)$$

where  $\theta > 0$ . As the definition shows, the proximal operator of  $\theta g$  is a point with minimum  $g(\mathbf{x})$  while approaching vector  $\mathbf{v}$ . The parameter  $\theta$  regulates the weight of  $g(\mathbf{x})$ . This form reminds us of the concept of the barrier method used in IP decoding but with a fixed weight instead, which is related to  $\theta$ . Given a small  $\theta$  and a differentiable  $g(\mathbf{x})$ , the proximal operator can be regarded as a gradient step [PB<sup>+</sup>14]:

$$\mathbf{prox}_{\theta g}(\mathbf{v}) \approx \mathbf{v} - \theta \nabla g(\mathbf{v}), \quad (3.52)$$

where  $\theta$  can be interpreted as step size.

Proximal decoding applies this method to minimize  $h(\mathbf{x})$ . Specifically,

$$\mathbf{prox}_{\gamma h}(\mathbf{x}) \approx \mathbf{x} - \gamma \nabla h(\mathbf{x}). \quad (3.53)$$

The minimization of  $f(\mathbf{x}) + \gamma h(\mathbf{x})$  is divided into two steps [WT21]:

$$\begin{aligned} \mathbf{r}^{k+1} &:= \mathbf{s}^k - \omega \nabla f(\mathbf{s}^k), \\ \mathbf{s}^{k+1} &:= \mathbf{prox}_{\gamma h}(\mathbf{r}^{k+1}) = \mathbf{r}^{k+1} - \gamma \nabla h(\mathbf{r}^{k+1}). \end{aligned} \quad (3.54)$$

The first step is a standard gradient descent with step-size parameter  $\omega$ . The second step is the proximal operator of function  $\gamma h$ .  $h(\mathbf{x})$  is minimized while  $\mathbf{s}^{k+1}$  maintains close to the updated  $\mathbf{r}^{k+1}$  from the first step. Additionally, for  $\nabla f(\mathbf{x})$  it holds [WT21]:

$$\nabla f(\mathbf{x}) \propto (\mathbf{x} - \mathbf{y}). \quad (3.55)$$

$\nabla h(\mathbf{x})$  is given by [WT21]:

$$\begin{aligned}\nabla h(\mathbf{x}) &= \left( \frac{\partial}{\partial x_1} h(\mathbf{x}), \dots, \frac{\partial}{\partial x_n} h(\mathbf{x}) \right)^\top, \\ \frac{\partial}{\partial x_j} h(\mathbf{x}) &= 4(x_j^2 - 1)x_j + \frac{2}{x_j} \sum_{i \in B(j)} (Q(i)^2 - Q(i)), \\ Q(i) &:= \prod_{j \in A(i)} x_j.\end{aligned}\tag{3.56}$$

The update rule of proximal gradient method is simple and straightforward. It is recommended for large scale, non-smooth minimization problems [PB<sup>+</sup>14]. Besides, the performance of proximal decoding depends greatly on the choice of  $\omega$  and  $\gamma$  [WT21]. Due to the characteristic of function  $h$ , there are non-codeword stationary points [WT21], which means the solution may not converge to the optimum. However, it is still a promising algorithm for its high throughput and low complexity [WT21].

### 3.3.3. Proximal Decoding Algorithm

At last, the algorithm is implemented as shown in Algorithm 2. Since  $\|\nabla h(\mathbf{x})\|_2$  can cause divergence if  $x_j$  is far from -1 or 1 [WT21]. Thus, box projection [WT21] was applied after the second step of minimization:

$$\mathbf{s}^{k+1} := \Pi_\eta(\mathbf{s}^{k+1}),\tag{3.57}$$

where  $\Pi_\eta$  refers to projection onto  $n$ -dimensional hypercube  $B_\eta := [-\eta, \eta]^n$  with  $\eta$  slightly bigger than 1 [WT21]. To be noted,  $\mathbf{s}^0 := \mathbf{0}$  was employed. But zero-forcing estimation could also be adopted [WT21].

---

#### Algorithm 2 Proximal Decoding [WT21]

---

**Input:** received word  $\mathbf{y} \in \mathbb{R}^n$

**Output:** estimated codeword  $\hat{\mathbf{c}} \in \mathbb{F}_2^n$

Step 1: Set  $\mathbf{s}^0 := \mathbf{0}$ .

Step 2: Repeat the following steps for  $O_{\max}$  times, with  $k$  as an iteration counter.

Step 2.1:  $\mathbf{r}^{k+1} = \mathbf{s}^k - \omega \nabla f(\mathbf{s}^k)$ .

Step 2.2:  $\mathbf{s}^{k+1} = \mathbf{r}^{k+1} - \gamma \nabla h(\mathbf{r}^{k+1})$ .

Step 2.3:  $\mathbf{s}^{k+1} := \Pi_\eta(\mathbf{s}^{k+1})$ .

Step 2.4:  $\hat{c}_j = \begin{cases} 0, & \text{if } s_j > 0, \\ 1, & \text{if } s_j < 0. \end{cases}$

Step 2.5: Exit if  $\mathbf{H} \cdot \hat{\mathbf{c}}^\top = \mathbf{0}^\top$ .

---





## 4. Simulation Results

Simulations are done with all-zero assumption [FWK05]. It is assumed that the codeword sent contains only zeros. Maximum number of frame errors is 100 within  $10^8$  frames in total.

### 4.1. Interior Point Decoding Simulation Results and Analysis

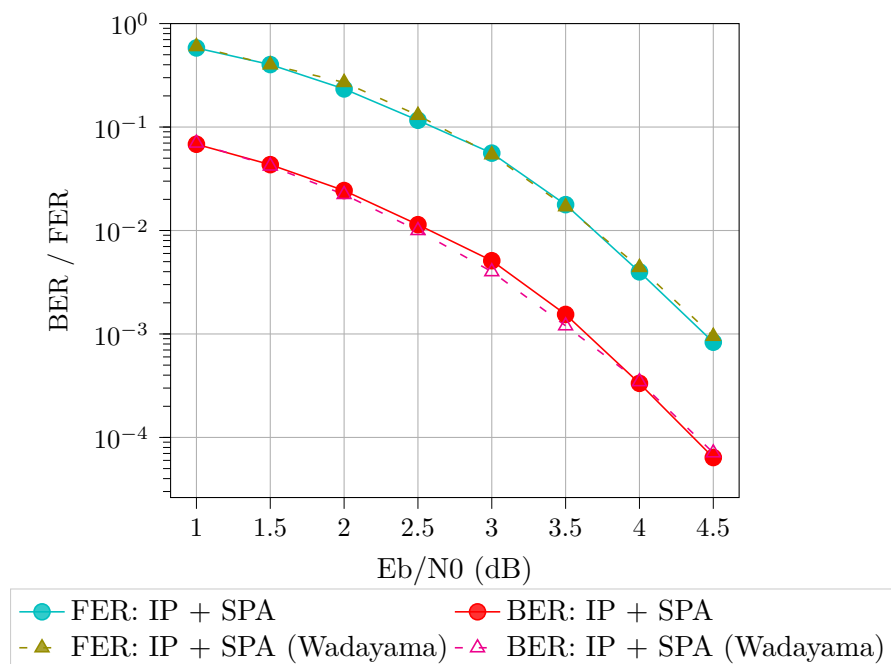
This part implements and validates the IP decoding [Wad07, Wad10]. The simulation results will be compared with the results from the previous studies. The influence of significant variables is examined by conducting experiments with controlled variables. Finally, ideas for performance improvement will be put forward and discussed.

#### 4.1.1. Validation and Analysis

Arrangement of the simulation is identical to [Wad07]:

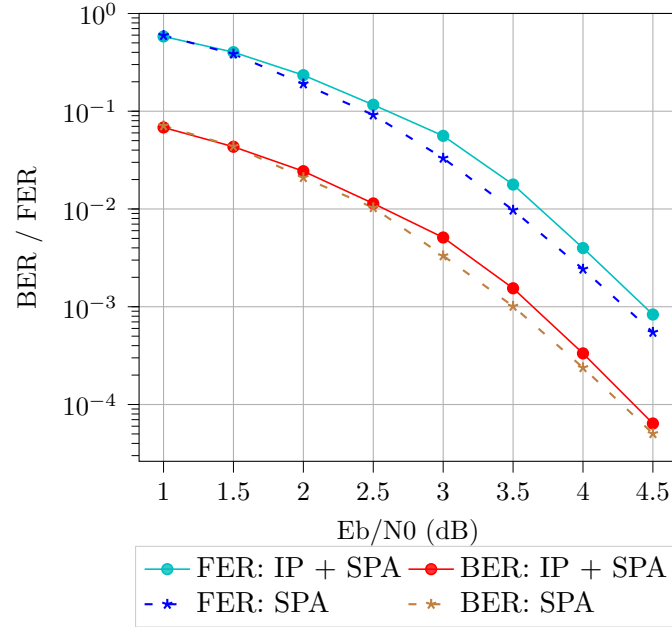
- Code: LDPC code 96.3.963 ( $n = 96, k = 48$ ) by MacKay [Mac].
- Parameters:  $t_0 = 10, \alpha = 2, s_0 = \frac{1}{2}, I_{\max} = 30, O_{\max} = 5, x_0 = (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$ .
- Minimization method in the inner loop: approximate gradient descent.
- Internal decoder: SPA,  $L_{\max} = 50$ .

The simulation result is as shown in Fig. 4.1, which approaches the result from [Wad07].

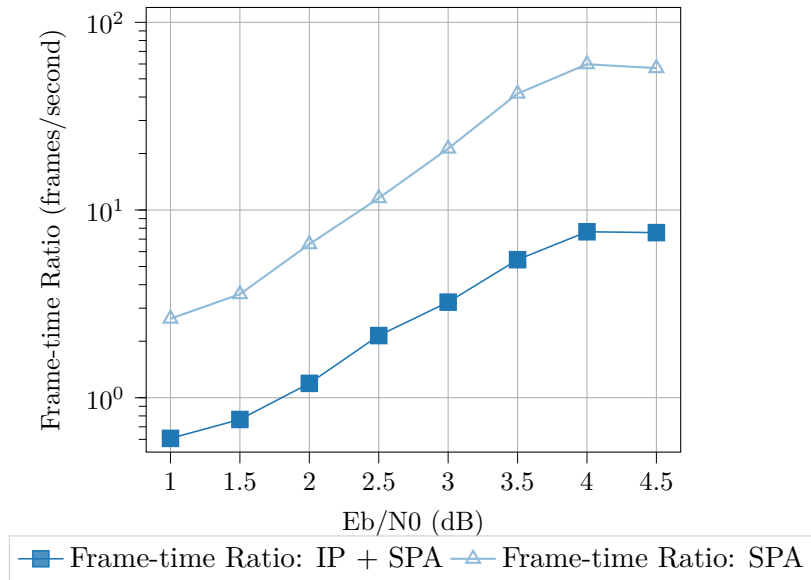


**Figure 4.1:** Comparison of Our Results with the Results from [Wad07] for Code 96.3.963 ( $n = 96, k = 48$ ) with  $I_{\max} = 30, O_{\max} = 5, L_{\max} = 50$

Besides, the author has also made a comparison between IP + SPA with the previous arrangement and SPA alone with 100 iterations [Wad07]. The comparison of results can be seen in Fig. 4.2. Our result is also similar to the result from [Wad07]: the gap between curves of FER is slightly bigger than between curves of BER. The difference by  $10^{-4}$  bit error rate is roughly 0.1 dB.



**Figure 4.2:** Comparison between IP Decoder with Internal SPA ( $I_{\max} = 30$ ,  $O_{\max} = 5$ ,  $L_{\max} = 50$ ) and SPA ( $L_{\max} = 100$ ) for Code 96.3.963 ( $n = 96$ ,  $k = 48$ )



**Figure 4.3:** Speed Comparison between IP Decoder with Internal SPA ( $I_{\max} = 30$ ,  $O_{\max} = 5$ ,  $L_{\max} = 50$ ) and SPA ( $L_{\max} = 100$ ) for Code 96.3.963 ( $n = 96$ ,  $k = 48$ )

As discussed in [Wad07, Wad10], the performance of above mentioned IP + SPA is still inferior to that of SPA only but within a small distance. Fig. 4.3 shows a comparison of decoding speed, where both simulations are done on an Laptop with processor Intel<sup>®</sup> Core<sup>™</sup> i7-8550U CPU @ 1.80GHz × 8 with OS Ubuntu 22.04.1 LTS. The measurement for speed is frame-time ratio or the so-called throughput [Wad07] in frames/second. The author hasn't published speed comparison for code 96.3.963. As we can see from Fig. 4.3, SPA is faster than IP + SPA for short code with  $n = 96$ .

#### 4.1.2. Influence of Scaling Factor $t$

The scaling factor  $t$  plays an important role in the barrier method applied in IP decoding. To observe the influence of  $t$ , we firstly put the internal decoder aside for it breaks the loops. Simulations of this part employ LDPC code 96.3.963 ( $n = 96, k = 48$ ) by MacKay [Mac] and apply approximate gradient descent for the minimization of merit function in the inner loop.

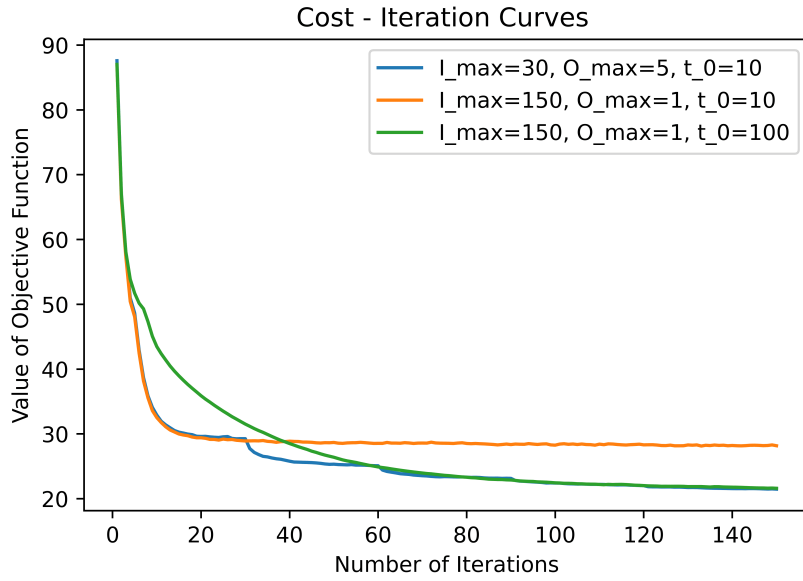
Parameter  $t$  is the weight of the objective function  $f(\tilde{\mathbf{c}})$  in the overall merit function  $\psi(\tilde{\mathbf{c}})$ . It is initialized with  $t_0$  and its multiple  $\alpha$  controls the speed at which  $t$  grows. Firstly,  $\alpha$  is fixed to 2 as suggested in original implementation, in order to observe the influence of  $t_0$ . Intuitively, when  $t_0$  is very big, the objective function has a relative large proportion in the overall merit function, which may weaken the role of barrier function from the very beginning. This results in more iterations in the inner loop [BBV04]. On the other hand, if  $t_0$  is too small, it takes in general more iterations in the outer loop [BBV04] to increase the weight of the objective function to a certain level. Table 4.1 shows the comparison of the same scaling schedules as in [Wad07] under the condition SNR = 4 dB:

Setting	$I_{\max}$	$O_{\max}$	$t_0$	$\alpha$
A	30	5	10	2
B	150	1	10	2
C	150	1	100	2

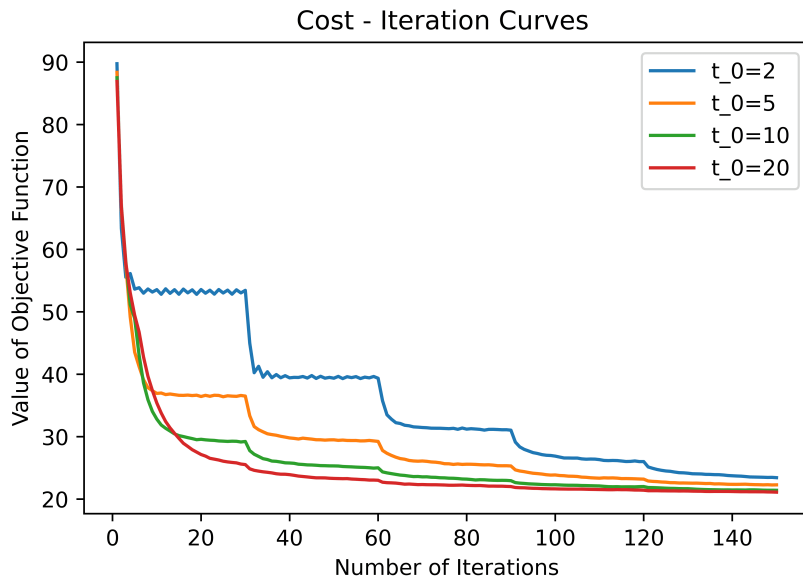
**Table 4.1:** Values of 3 Settings

All three settings have 150 iterations in total. Setting  $A$  and  $B$  have the same  $t_0$  whereas  $A$  has more iterations in outer loop, which increases  $t$  to a larger value than that of setting  $B$ . Setting  $C$  has a much bigger  $t_0$  while  $t$  is not further increased, since there is only one iteration in the outer loop. The ultimate goal is to locate a codeword that is the closest to the received vector. Therefore, the objective function can be used to measure the quality of different scaling schedules. Besides, LDPC code 96.3.963 by MacKay [Mac] is applied for the simulation.

The experiment is set at 4 dB. We get similar results to those in [Wad07] with average value from 1000 trials. As we can see from Fig. 4.4, setting  $A$  (blue line) and  $B$  (orange line) converge faster than setting  $C$  (green line), which shows that a smaller  $t_0$  contributes to faster convergence. Meanwhile,  $A$  outperforms  $B$  for it reduces the objective function to a lower value as the number of iterations increases, which is due to a larger end value of  $t$ .



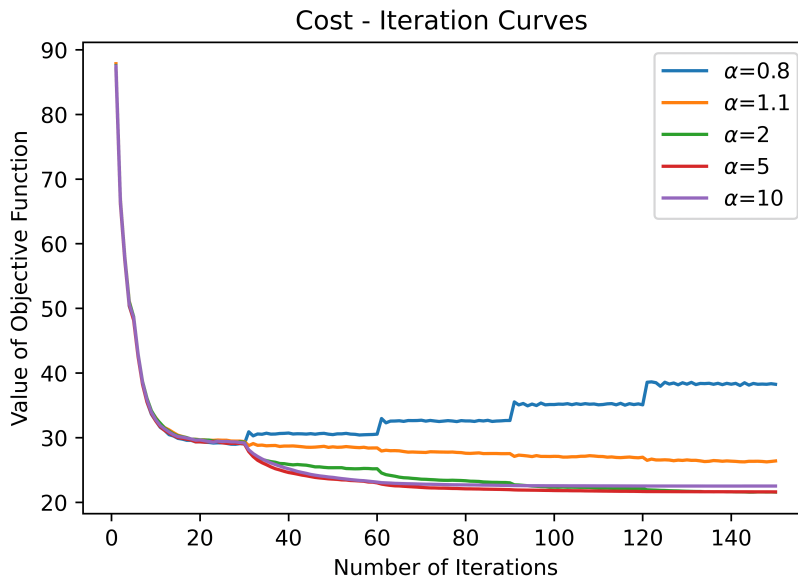
**Figure 4.4:** The Average Value of the Objective Function from 1000 trials with Different Schedules (with Approximate Gradient Descent,  $\alpha = 2$ ) for Code 96.3.963 ( $n = 96$ ,  $k = 48$ ) at 4 dB



**Figure 4.5:** The Average Value of the Objective Function from 1000 trials with Very Small  $t_0$  (with Approximate Gradient Descent,  $I_{\max} = 30$ ,  $O_{\max} = 5$ ,  $\alpha = 2$ ) for Code 96.3.963 ( $n = 96$ ,  $k = 48$ ) at 4 dB

However, given  $I_{\max} = 30$ ,  $O_{\max} = 5$  and  $\alpha = 2$ , if  $t_0$  is too small, as shown in Fig. 4.5, it is expected to see obvious oscillation in the inner loop and steps between the inner loop and the outer loop. Because of the small  $t$ , during updates in the inner loop, there isn't effective reduction of the value of the objective function. These gaps disappear when  $t_0$  is

larger than 10. Therefore, we come to the conclusion that gradually increasing  $t$  to a large value while starting with a relatively small  $t_0$  is most beneficial to the overall performance, which agrees with the conclusion made in [Wad07]. Generally speaking, initializing  $t$  with  $t_0 = 10$  is a good choice for it reduces the value of objective function to a flat area faster than the case by  $t_0 = 20$  with a value smaller than the value with  $t_0 = 5$ .



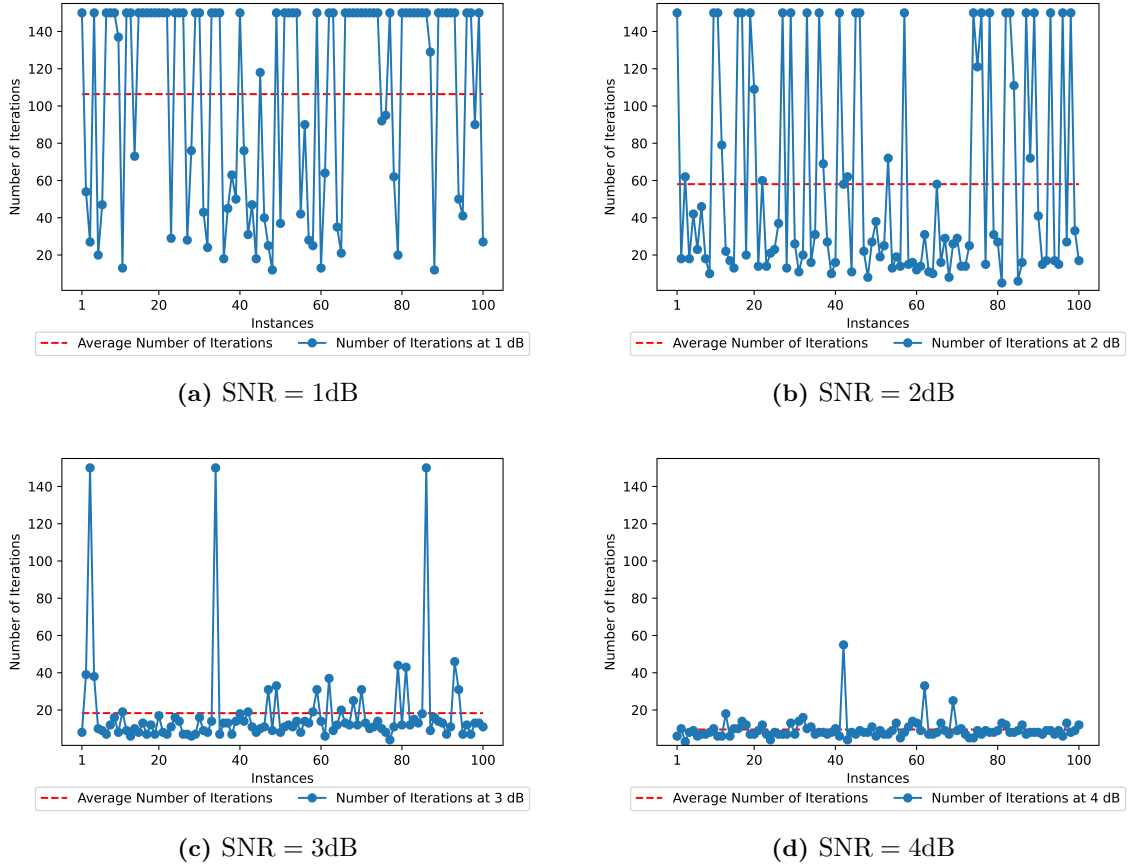
**Figure 4.6:** The Average Value of the Objective Function from 1000 trials with Various  $\alpha$  (with Approximate Gradient Descent,  $I_{\max} = 30$ ,  $O_{\max} = 5$ ,  $t_0 = 10$ ) for Code 96.3.963 ( $n = 96$ ,  $k = 48$ ) at 4 dB

As for the parameter  $\alpha$ , as stated in [BBV04], with Newton method, the choice of  $\alpha$  seems not critical as long as it is bigger than 1 and  $\alpha$  near 1 would cost a few more Newton steps. In practice, in the case of gradient descent, as shown in Fig. 4.6, with  $I_{\max} = 30$ ,  $O_{\max} = 5$  and  $t_0 = 10$ , there seems indeed not to be significant difference between various  $\alpha$  over 2 in terms of the converging speed and the value of the objective function. The curve with  $\alpha = 1.1$  glides very slowly down. For  $\alpha$  below 1, the weight of the objective function is rather reduced instead of being increased. This results in a more dominant barrier function and thus turns the searching direction away from received signal  $\mathbf{y}$ , which leads to a bigger value of the objective function as shown in Fig. 4.6.

### 4.1.3. Iterations Required to Converge to Valid Codewords

When we put back the internal decoder, in our case an SPA or MSA decoder, it helps to cut redundant iterations [Wad07]. The internal decoder takes outputs from the inner loop to decode and breaks both the inner loop and the outer loop when a valid codeword is decoded. However, it interferes neither with the update of the merit function in the outer loop nor the minimization iterations in the inner loop. As soon as the objective function is reduced to "the flat part of the cost curve", the internal SPA can successfully deliver a valid codeword and call a halt to the IP algorithm [Wad07]. To investigate the effectiveness of the internal decoder concerning the reduction of overall iterations, we now

take an example of internal SPA and put it right after Step 2.1.3 in Algorithm 1. The SPA iterates thus inside the inner loop. Simulations of this part employ LDPC code 96.3.963 ( $n = 96, k = 48$ ) by MacKay [Mac], applying approximate gradient descent in the inner loop with the setting  $t_0 = 10, \alpha = 2, I_{\max} = 30, O_{\max} = 5, L_{\max} = 50$ . 100 realizations were observed respectively from SNR = 1 dB to SNR = 4 dB.

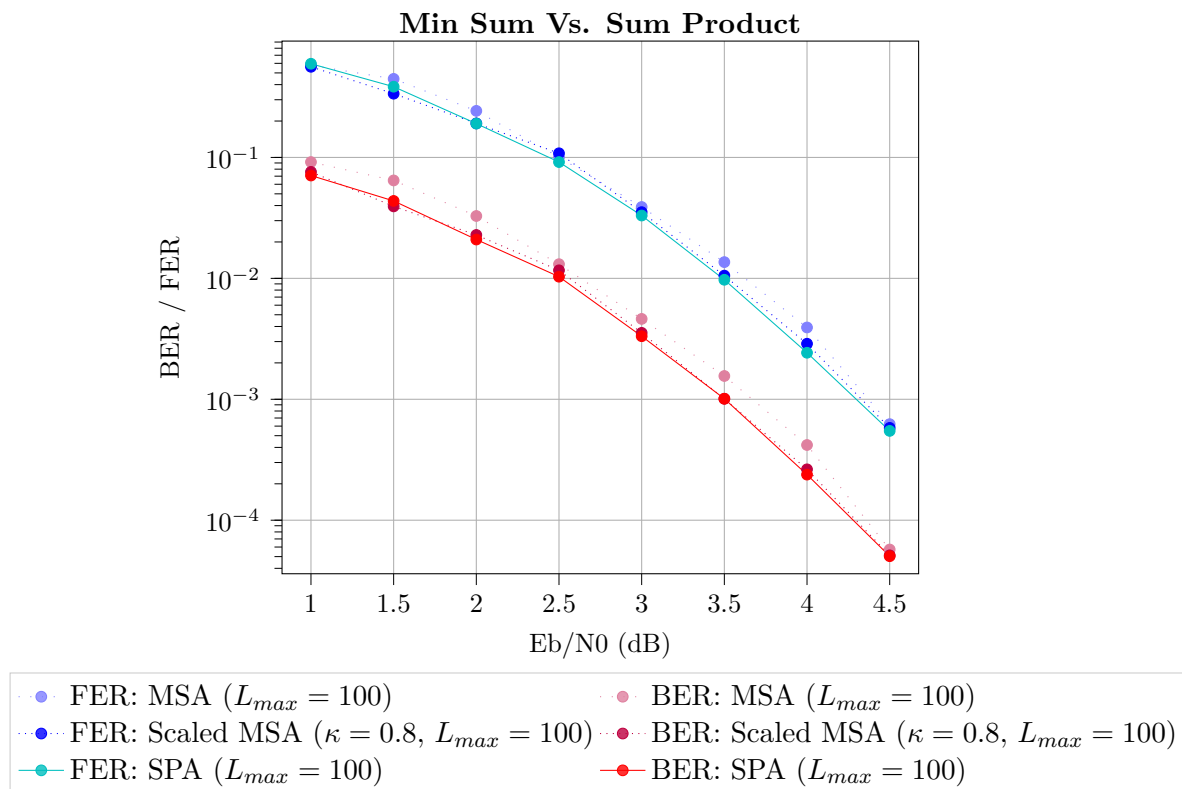


**Figure 4.7:** Number of Iterations Required to Output A Codeword (with Approximate Gradient Descent,  $t_0 = 10, \alpha = 2, I_{\max} = 30, O_{\max} = 5, L_{\max} = 50$ ) for Code 96.3.963 ( $n = 96, k = 48$ )

As shown in Fig. 4.7, with the increase of SNR, there are fewer iterations needed to yield a codeword. Above 3 dB, most of the instances can be successfully decoded within 30 iterations. In other words, if we put SPA back to Step 2.2 in Algorithm 1 as supposed in IP decoding, above 3 dB there is principally only one iteration in the outer loop needed.

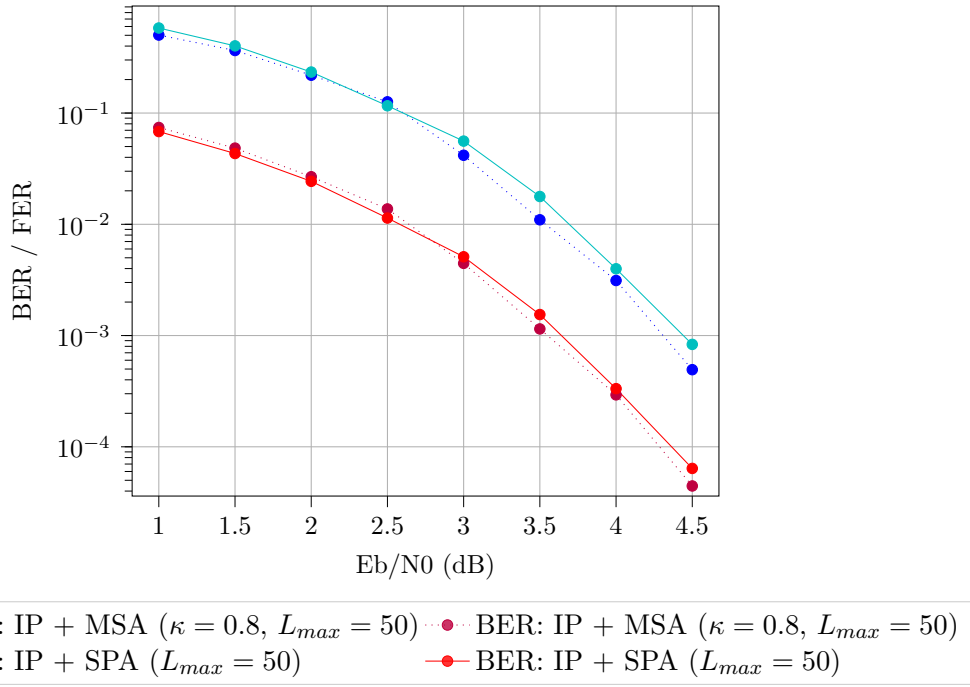
#### 4.1.4. IP + SPA Vs. IP + MSA

In work [Wad10], a scaled MSA was adopted as internal decoder with damping factor  $\kappa$  between 0.7 and 0.9. A scaled MSA, or known as normalized MSA has better performance than standard MSA [YHB04, CDE<sup>+</sup>05, MPK<sup>+</sup>17]. However, the author of [Wad10] didn't publish performance of IP + MSA for AWGN channel. A comparison among SPA, MSA and scaled MSA ( $\kappa = 0.8$ ) for short code 96.3.963 is shown below in Fig. 4.8. All three are initialized with  $\text{LLR} = \frac{2y}{\sigma^2}$ .

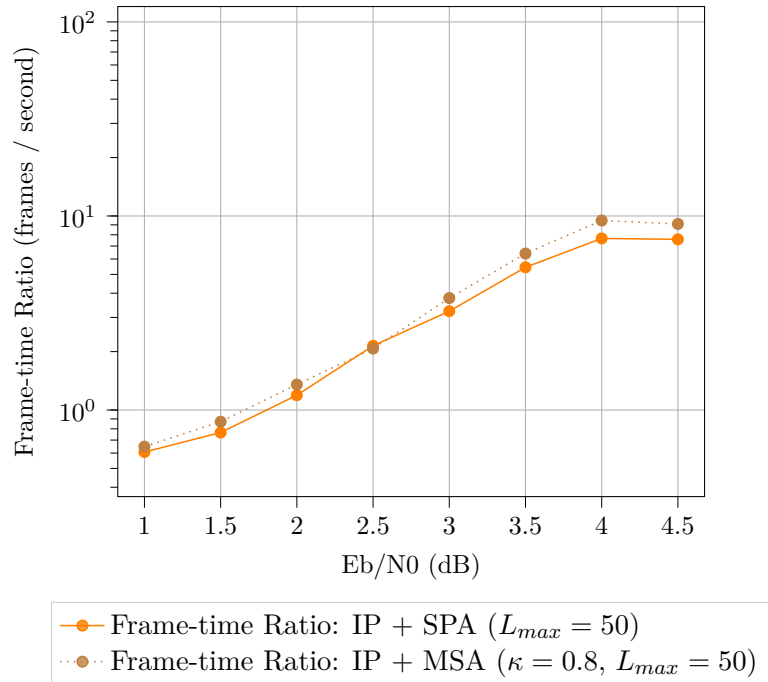


**Figure 4.8:** Comparison of MSA and SPA ( $L_{max} = 100$ ) for Code 96.3.963 ( $n = 96, k = 48$ )

As mentioned in Section 2.3, MSA is an approximation of SPA with reduced computational complexity. As Fig. 4.8 shows, the difference between standard MSA and SPA is very small for short code, and the gap almost disappears with damping factor. Without specific notice, the MSA hereafter refers to normalized MSA, and the factor  $\kappa$  is fixed to 0.8. Now we compare the performance of IP + MSA and IP + SPA. Approximate gradient descent is used for the minimization of merit function in the inner loop.



**Figure 4.9:** Performance Comparison of IP decoders with Internal MSA and SPA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96, k = 48$ )



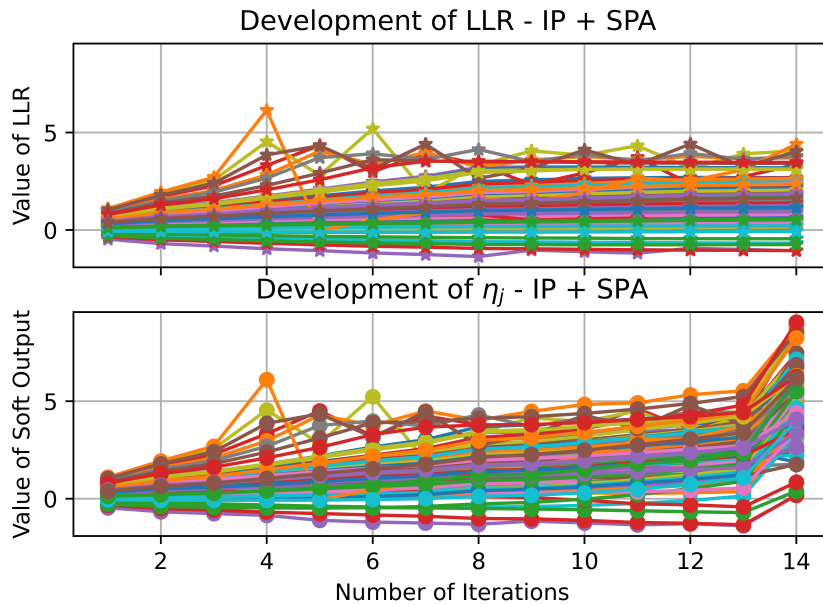
**Figure 4.10:** Speed Comparison of IP Decoders with Internal MSA and SPA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96, k = 48$ )



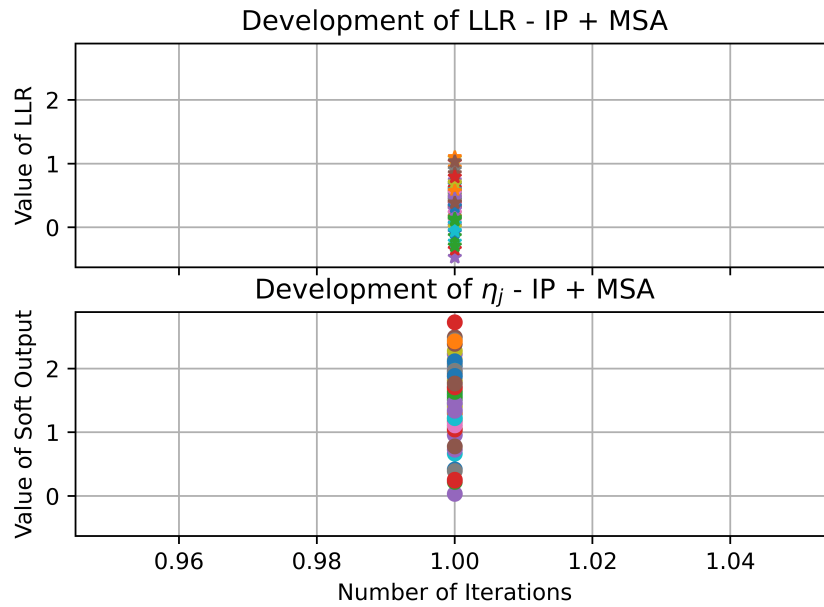
Surprisingly, as shown in Fig. 4.9, starting from 3 dB, the IP + MSA ( $\kappa = 0.8$ ) outperforms the IP + SPA. As expected, algorithm IP + MSA is a little faster than IP + SPA as shown in Fig. 4.10. With the intention to identify the possible reason, the LLR  $\lambda_j := \ln\left(\frac{1-\tilde{c}_j}{\tilde{c}_j}\right)$  defined in [Wad07] was firstly investigated. We put the SPA or MSA again right after the update of gradient descent, in order to collect every output from the inner loop. The prior LLR  $\lambda_j$ , which is the initialization of BP decoder as well as the posterior information  $\eta_j$  are explored.  $\eta_j$  includes the prior  $\lambda_j$  and information through message exchange between VNs and CNs:

$$\eta_j = \lambda_j + \sum_{i \in B(j)} \xi_{i \rightarrow j} \quad (4.1)$$

Since IP + MSA starts to overtake IP + SPA at 3 dB, the development of LLR and  $\eta_j$  of one word are plotted in Fig. 4.11 and Fig. 4.12 at 3 dB for IP + SPA and IP + MSA. Both come from the same instance. It helps us to see what happened to LLR and  $\eta_j$  during iterations. Each curve stands for a symbol. Therefore, there are 96 curves in total. Since we assume that an all-zero codeword has been transmitted, there are  $n$  curves standing for  $n$  samples for the same bit. The curves are supposed to go towards the same direction and stop when internal decoders have successfully decoded a valid codeword. Given that 0 is the threshold for decision at the end of each iteration in BP decoders. So it is expected to see that as soon as  $\eta_j$  for all  $j \in \mathcal{J}$  is equal to or above 0, the algorithm is terminated by the internal decoder.

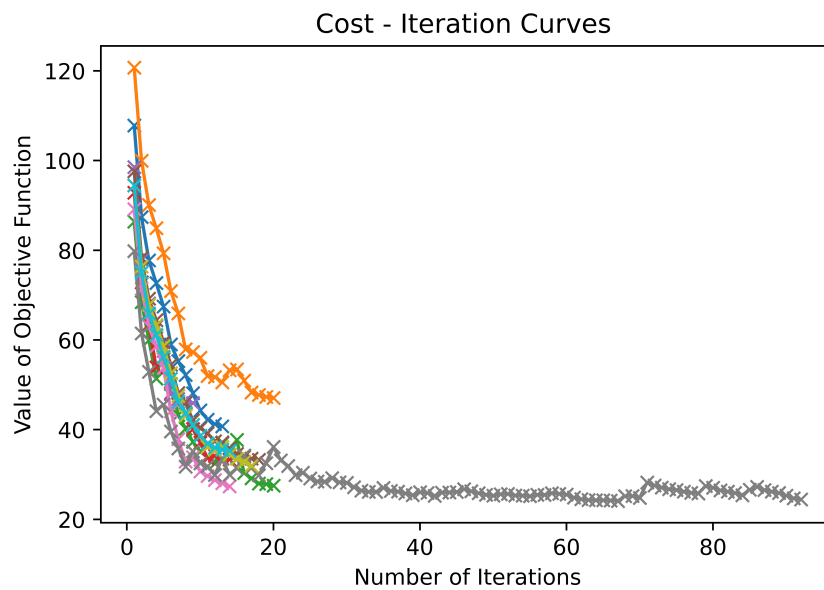


**Figure 4.11:** The Development of LLR and Soft Output  $\eta_j$  in IP + SPA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96$ ,  $k = 48$ )



**Figure 4.12:** The Development of LLR and Soft Output  $\eta_j$  in IP + MSA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96, k = 48$ )

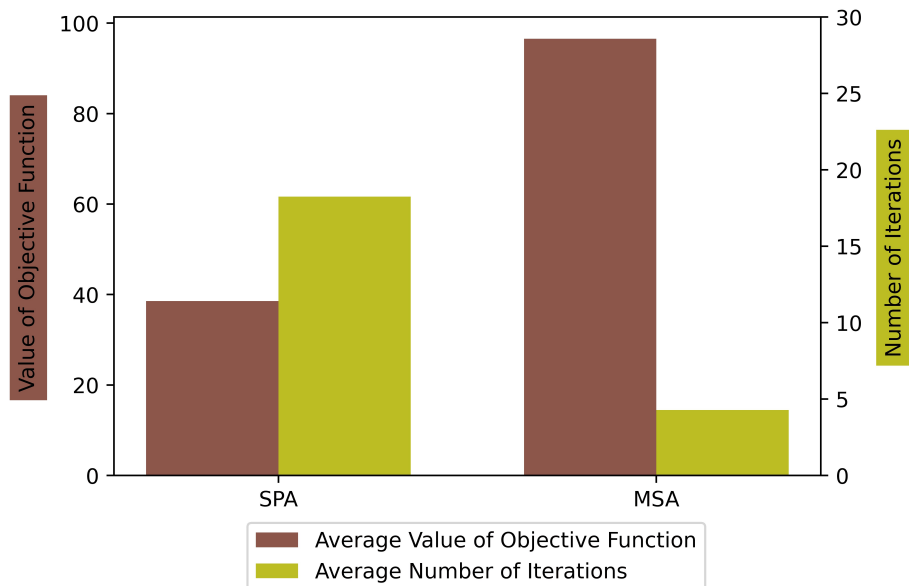
As we can see in Fig. 4.11 and Fig. 4.12, IP + MSA deviates rapidly after the first iteration from zero towards positive values and converges much faster to a valid codeword than the IP + SPA.



**Figure 4.13:** 10 Curves of the Objective Function from IP + SPA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96, k = 48$ )

As illustrated in Fig. 4.4 before, the value of the objective function drops quickly to a lower value and then goes slowly down or maintains more or less stable at this level. SPA

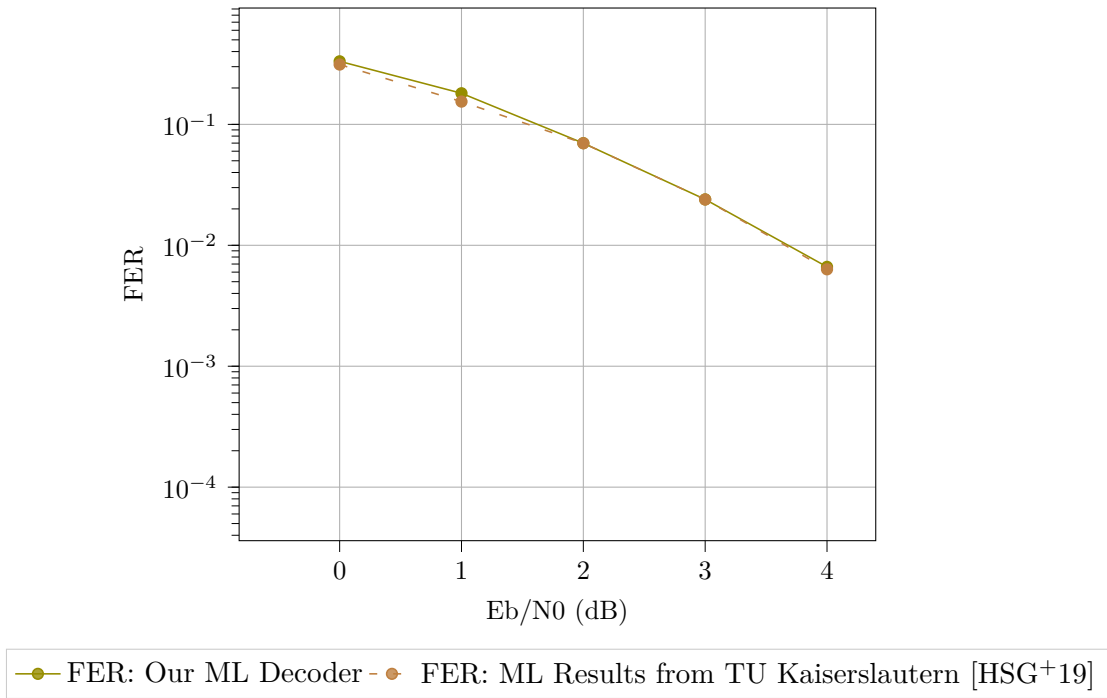
can yield a codeword as soon as the value of the objective function reaches a flat part of the curve [Wad07]. We confirm this conclusion by plotting 10 instances at 3 dB as shown in Fig. 4.13. The curves stop mostly before 20 iterations at a value between 30 to 40.



**Figure 4.14:** Comparison of the Average End Value of the Objective Function and the Average Number of Iterations from 100 trials between IP + SPA and IP + MSA Using Approximate Gradient Descent for Code 96.3.963 ( $n = 96, k = 48$ )

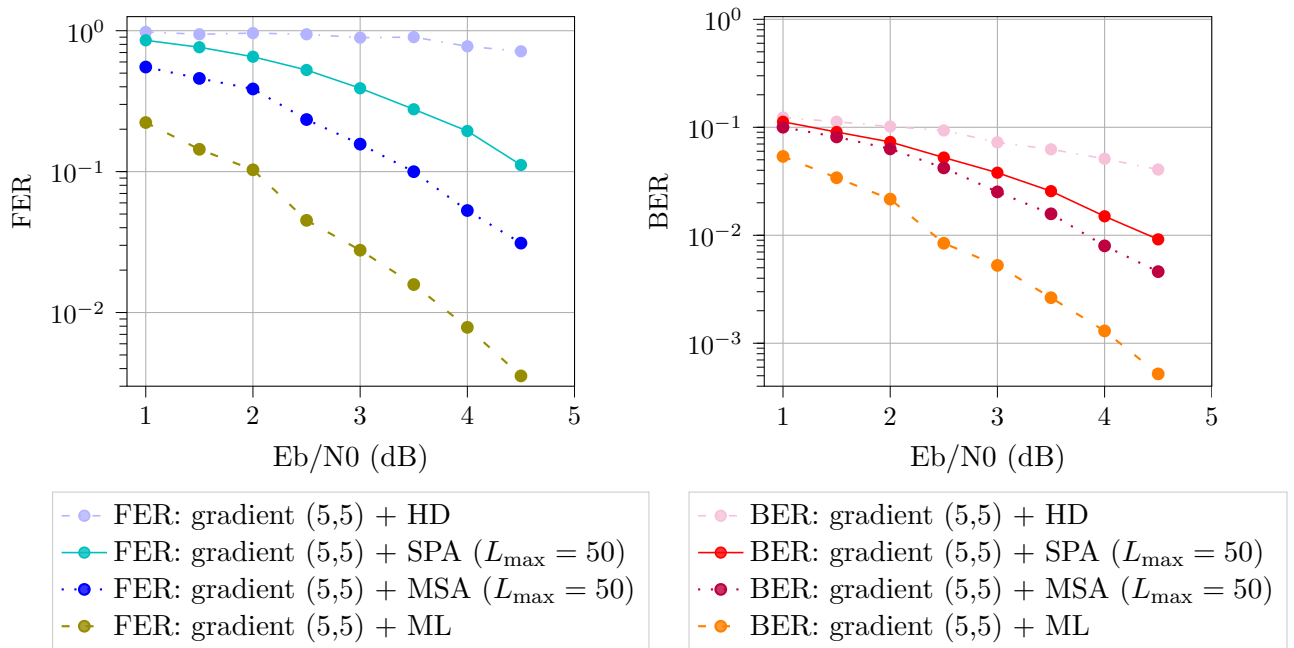
Fig. 4.14 shows the average result of 100 trials. According to the statistics in Fig. 4.14, the internal SPA decodes successfully when the value of the objective function is reduced to between 30 to 40. On the contrary, the internal MSA can deliver a codeword when the value of the objective function is still as high as over 80 and thus requires less iterations.

To perceive influencing factors on the performance of internal decoders, a HD, SPA, MSA or ML decoder is placed outside the outer loop. Now they won't interrupt the loops and decode only when all loops inside and outside are finished. The notation from [Wad10] is followed: for example, gradient (30,5) denotes IP method applying approximate gradient descent, with  $I_{\max} = 30, O_{\max} = 5$ . Code CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ ) is adopted. First of all, our ML decoder was validated with the results from [HSG<sup>+</sup>19] for code CCSDS ( $n = 32, k = 16$ ). As seen in Fig. 4.15, the curves overlap mostly with some minor deviations.

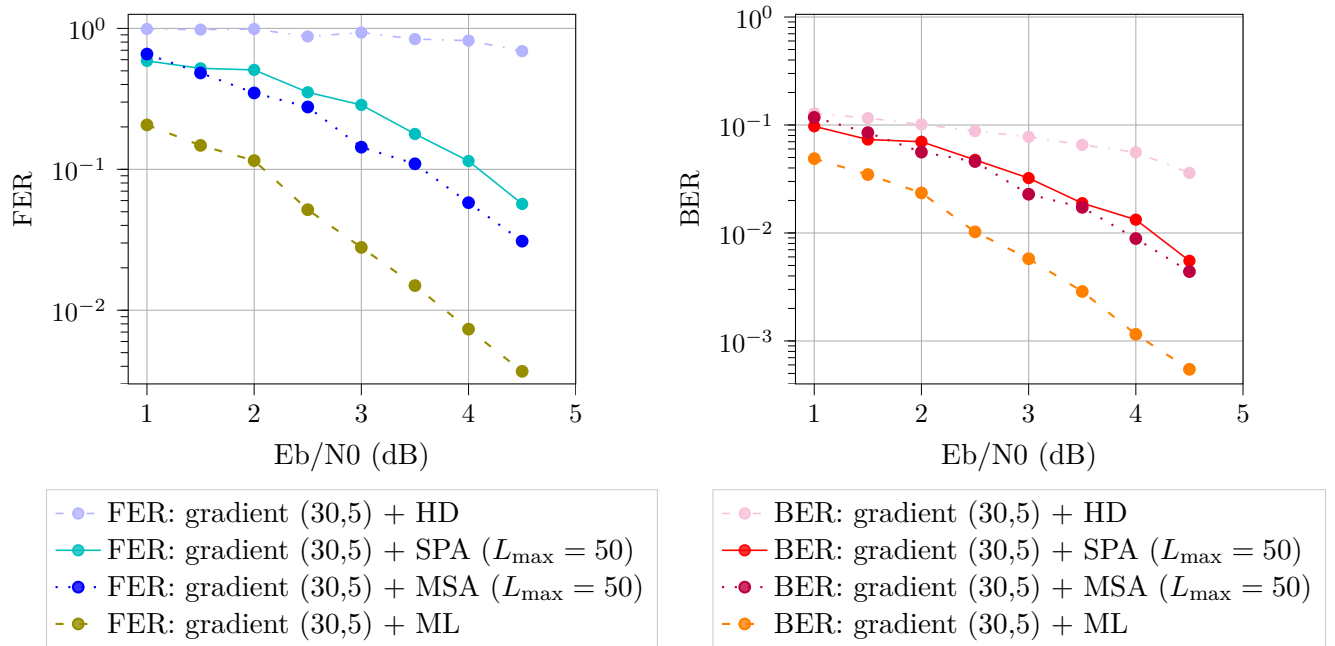


**Figure 4.15:** Validation of ML Decoder for Code CCSDS [HSG+19] ( $n = 32, k = 16$ )

The results are shown in the following figures. Unfortunately, the result with SPA has suffered severe performance degradation when  $I_{\max} = 5$ , as shown in Fig. 4.16.

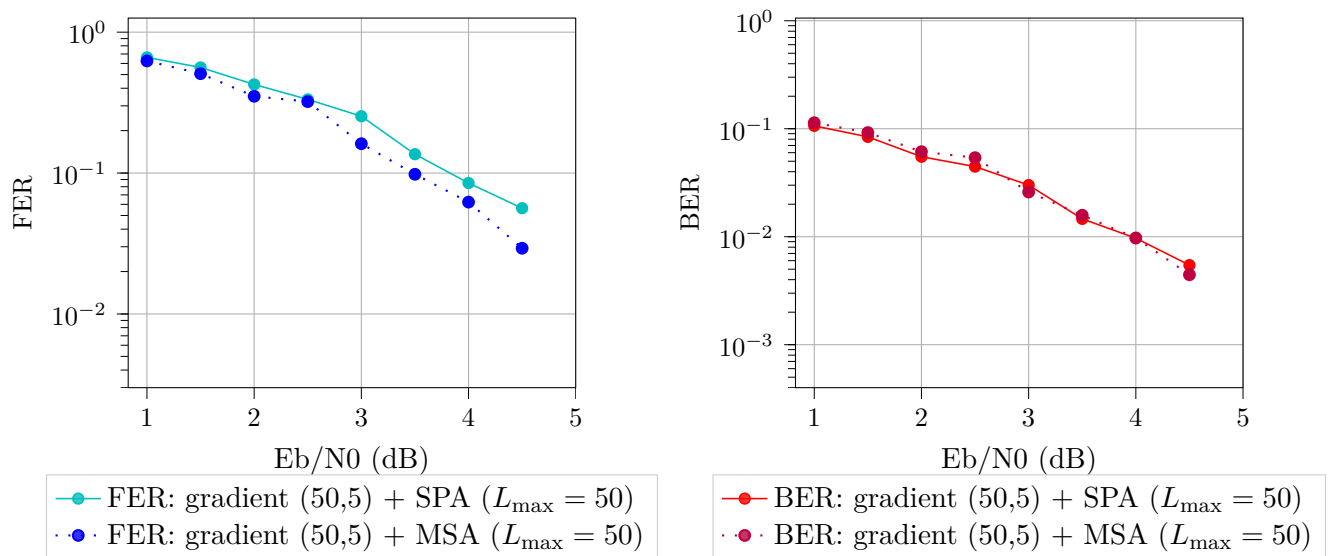


**Figure 4.16:** Comparison of Performance of Various Decoders outside The Outer Loop for Code CCSDS ( $n = 32, k = 16, I_{\max} = 5, O_{\max} = 5$ )



**Figure 4.17:** Comparison of Performance of Various Decoders outside The Outer Loop for Code CCSDS ( $n = 32$ ,  $k = 16$ ,  $I_{\max} = 30$ ,  $O_{\max} = 5$ )

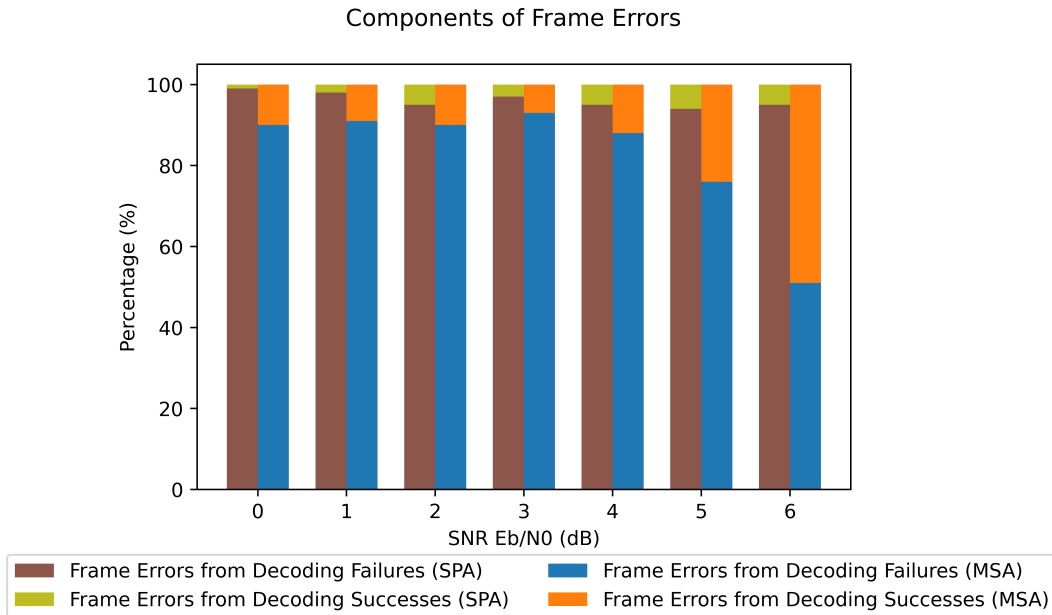
Because the curves for HD and ML don't show much change, Fig. 4.18 focuses only on results for SPA and MSA outside the outer loop.



**Figure 4.18:** Comparison of Performance of Various Decoders outside The Outer Loop for Code CCSDS ( $n = 32$ ,  $k = 16$ ,  $I_{\max} = 50$ ,  $O_{\max} = 5$ )

As shown in Fig. 4.17, with  $I_{\max} = 30$ , the curve of BER for SPA is close to that for MSA but there is still an obvious gap between the curves of FER for both decoders. As  $I_{\max}$  is increased to 50, as shown in Fig. 4.18, the gap is further narrowed. It is obvious that the performance of BP decoders are influenced by the process of interior point method since they take outputs from interior point decoder. At cost of more iterations, SPA has been improved whereas there is little positive effect on MSA. At some points, more iterations even drag MSA's FER performance down, for example at 2.5 dB and 3.5 dB with  $I_{\max} = 30$  and in the curve from 2.5 to 4 dB with  $I_{\max} = 50$ .

Even though SPA outperforms MSA in AWGN channel as shown in Fig. 4.8, the outputs from interior point decoder, namely the interior points, are no longer Gaussian distributed. They are concentrated in the zone between 0 and 1, or -1 and 1 in terms of received signals from AWGN channel with BPSK, which may hold back the performance of BP decoders. Under this precondition, an internal MSA decoder may perform better than an internal SPA decoder. Based on the experiments done, MSA can yield a better performance than SPA with configurations as low as  $I_{\max} = 5$ ,  $O_{\max} = 5$ . To be noticed, in these experiments the BP decoders are placed outside interior point decoder. There are 25 iterations in total. Meanwhile, combined with Fig. 4.14 and Fig. 4.16 to Fig. 4.18, it has also indirectly shown that the choice of internal decoders has impact on decoding speed. From this point of view, MSA is more suitable because it has less computational complexity than SPA due to simplified CN updates [CDE<sup>+</sup>05, MPK<sup>+</sup>17] while assisting IP method better with less iterations.



**Figure 4.19:** Components of Frame Errors between IP + SPA and IP + MSA Using Approximate Gradient Descent for Code CCSDS ( $n = 32$ ,  $k = 16$ ,  $I_{\max} = 30$ ,  $O_{\max} = 5$ )

At last, a statistic was done concerning the components of frame errors. Frame errors from decoding successes are the frame errors among converged valid codewords that delivered by the internal decoder. As we see in Fig. 4.19, the decoding accuracy of internal MSA is not as good as internal SPA. Even though internal MSA performs better and faster combined with IP decoder, there is still room for improvement.

#### 4.1.5. Approximate Gradient Descent Vs. Newton Method with Approximate Hessian

Furthermore, an improved IP decoder with the approximated Newton method instead of the approximate gradient descent is investigated. Newton method is commonly used for minimization problem. The Newton step is the steepest descent direction and thus a good option for updating search direction [BBV04]. IP decoding with Newton method applying the approximated Hessian matrix in the inner loop was proposed in [Wad10] as a better solution than with the approximate gradient descent for linear vector channels. MSA was applied as the internal decoder [Wad10]. According to [Wad10], Newton method is supposed to reduce the number of iterations. As shown in [Wad10], IP decoding with the approximate Newton method, where  $I_{\max} = 5$ ,  $O_{\max} = 5$ , outperforms IP decoder with approximate gradient descent, where  $I_{\max} = 20$ ,  $O_{\max} = 5$ . However with the approximate Newton method,  $I_{\max} = 20$ ,  $O_{\max} = 5$  brings only insignificant improvement compared to  $I_{\max} = 5$ ,  $O_{\max} = 5$  [Wad10]. Comparisons for the AWGN channel between these two minimization methods haven't been published by the authors from [Wad10]. Thus, this section intends to fill the gap. The simulations are conducted with the same setting and code as before. The notation from [Wad10] is followed: for example, Newton (30, 5) denotes IP method applying the approximated Newton method, with  $I_{\max} = 30$ ,  $O_{\max} = 5$ . The settings for comparisons are as follows.

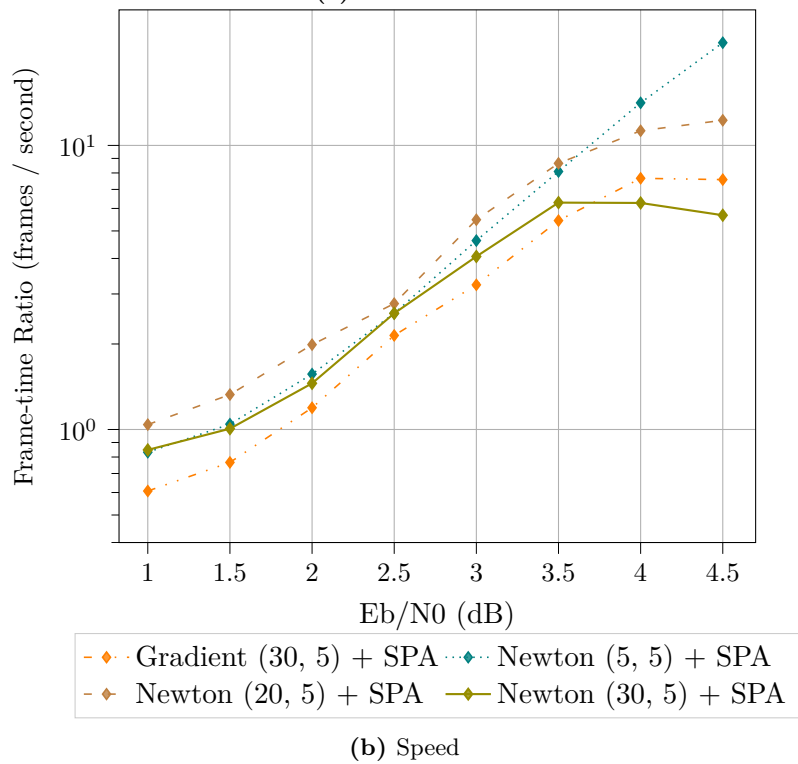
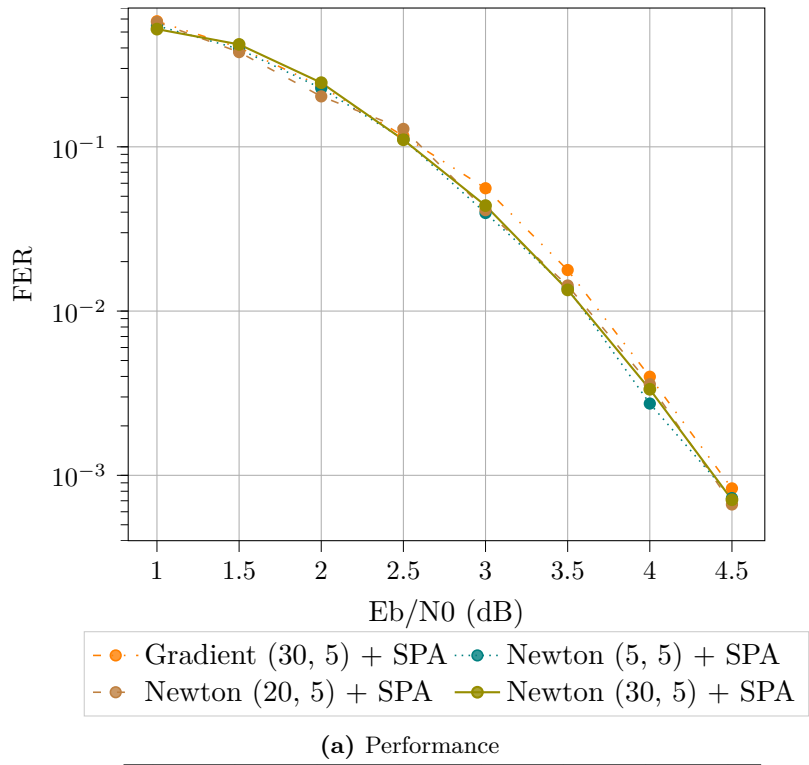
##### Gradient Descent Vs. Newton Method with Internal SPA

- Code: LDPC code 96.3.963 ( $n = 96$ ,  $k = 48$ ) by MacKay [Mac].
- Parameters:  $t_0 = 10$ ,  $\alpha = 2$ ,  $s_0 = \frac{1}{2}$ ,  $I_{\max} = 30$ ,  $O_{\max} = 5$ .
- Starting point:  $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}\right)$ .
- Internal decoder: SPA,  $L_{\max} = 50$ .

##### Gradient Descent Vs. Newton Method with Internal MSA

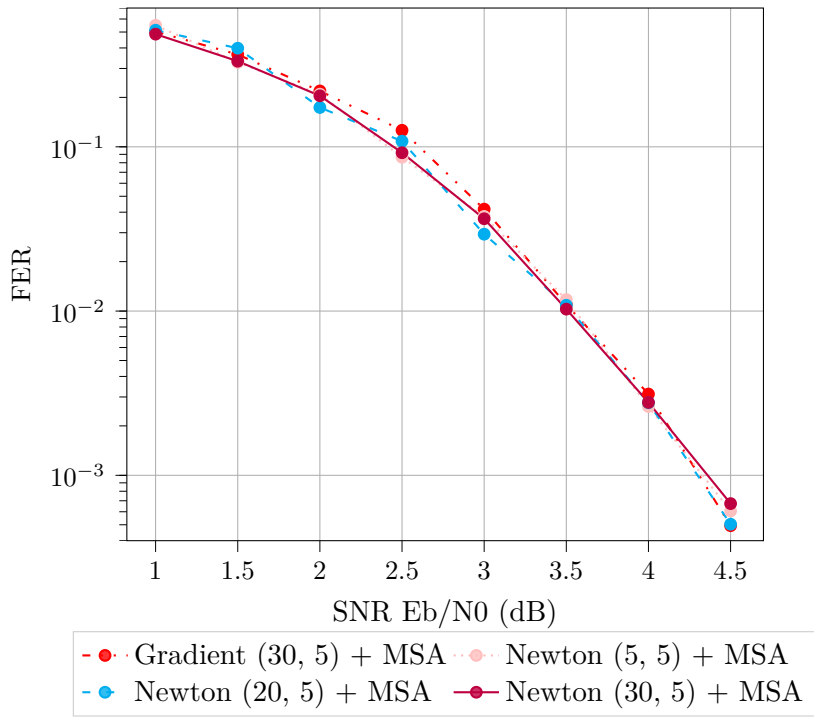
- Code: LDPC code 96.3.963 ( $n = 96$ ,  $k = 48$ ) by MacKay [Mac].
- Parameters:  $t_0 = 10$ ,  $\alpha = 2$ ,  $s_0 = \frac{1}{2}$ ,  $I_{\max} = 30$ ,  $O_{\max} = 5$ .
- Starting point:  $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2}\right)$ .
- Internal decoder: MSA,  $L_{\max} = 50$ .

As shown in Fig. 4.20 and Fig. 4.21, IP decoding with the approximated Newton method proposed by [Wad10] can indeed cut down the number of iterations. Due to limited samples, the results may have minor errors from their actual values. Generally speaking, Newton (5,5) performs better than gradient (30, 5) with higher speed due to less iterations. However, Compared to Newton (5, 5), Newton (20, 5) and Newton (30, 5) show little improvement while processing less frames per second.

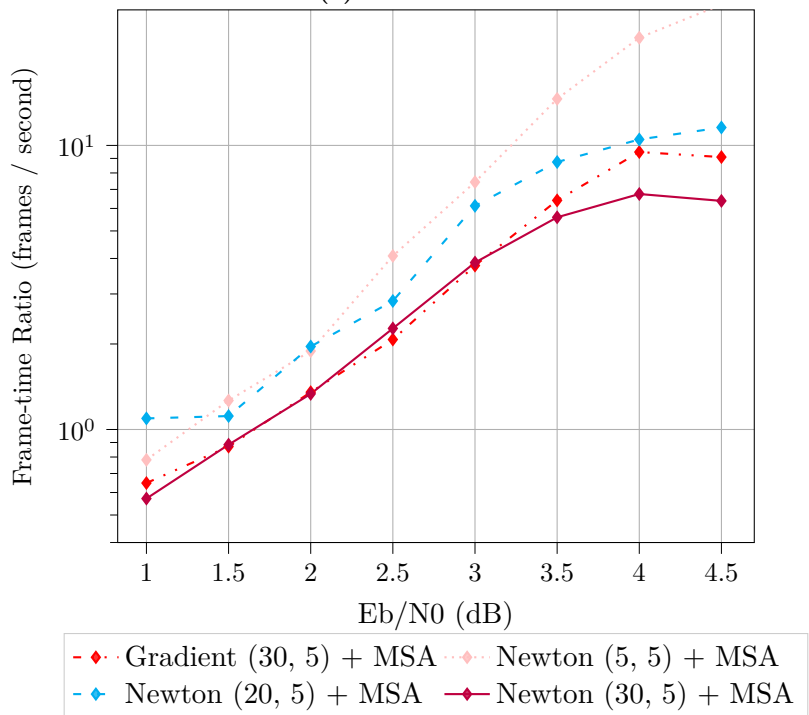


**Figure 4.20:** Approximate Gradient Descent Vs. Approximate Newton Method with Internal SPA for Code 96.3.963 ( $n = 96$ ,  $k = 48$ )



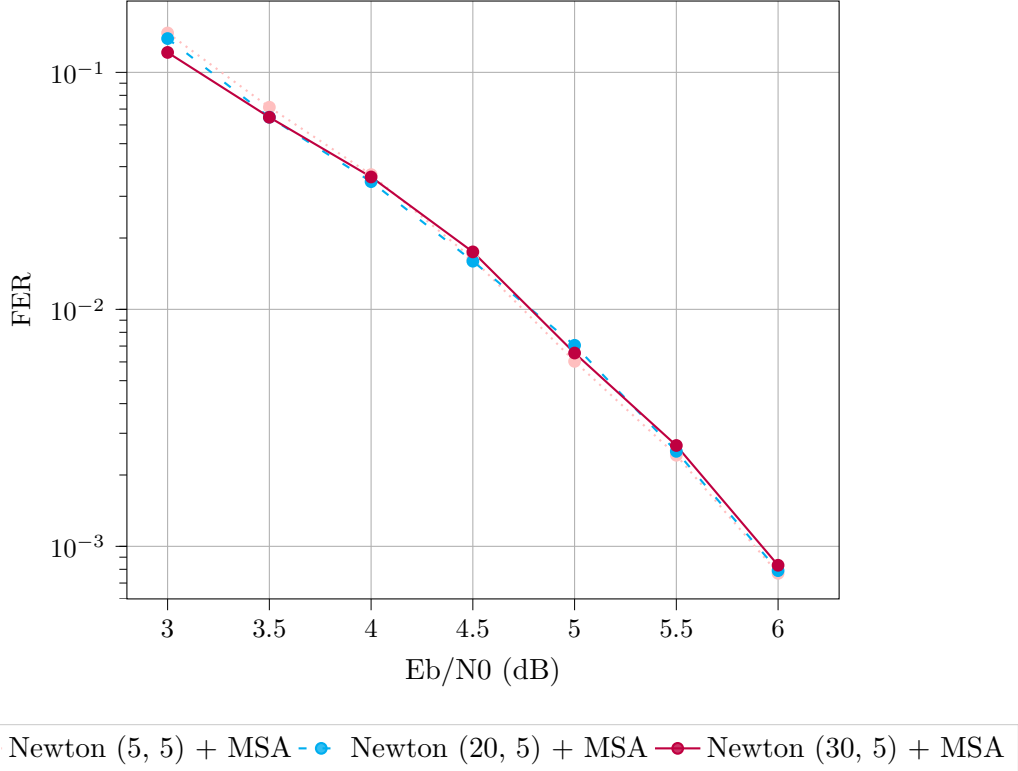


(a) Performance



(b) Speed

**Figure 4.21:** Approximate Gradient Descent Vs. Approximate Newton Method with Internal MSA for Code 96.3.963 ( $n = 96$ ,  $k = 48$ )



**Figure 4.22:** A Test of Approximate Newton Method with Internal MSA at Higher SNR for Code CCSDS ( $n = 32, k = 16$ )

Out of the question that whether there would be significant improvement at higher SNR, a test with shorter code CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ ) was conducted with internal MSA as shown in Fig. 4.22. IP decoder applying the approximated Newton method with more iterations still don't show advantage against the configurations with less iterations.

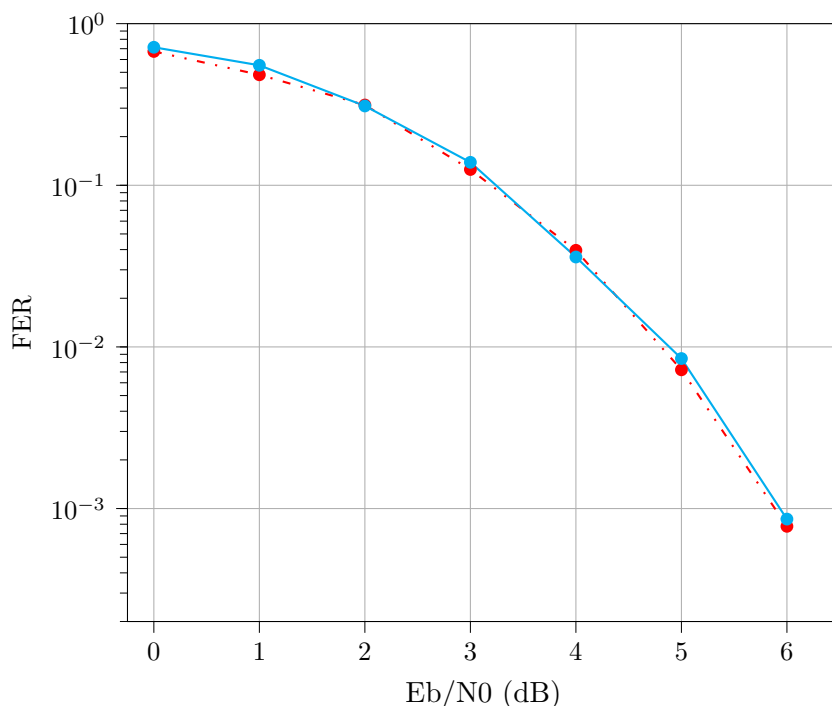
#### 4.1.6. Experimental Study on the Performance Improvement

Finally, improvement ideas will be proposed. The code used in simulations is short code CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ ). In pursuit of performance improvement, a few experiments have been done regarding both time complexity and error rate. For instance, an extra line search and interpolation between iterations were implemented with the attempt to accelerate the algorithm. However, they didn't make much difference in terms of error rate while taking longer than the original algorithm. A possible reason for that is because the step size is restrained by the condition of being interior points. Changing starting point that is nearer to the received symbols, in order to reduce iterations, also didn't make progress. Besides, the author has adopted in [Wad07, Wad10] prior LLR as:

$$\lambda_j = \ln \frac{(1 - \tilde{c}_j)}{\tilde{c}_j}. \quad (4.2)$$

The conventional LLR for AWGN channel is  $2y_j/\sigma^2$  in relevance to both received words and the channel according to [Moo20, Ana01, JZXZ07, ZJJ11]. One experiment is to map

outputs from the inner loop from  $(0, 1)$  to  $(1, -1)$  as  $\tilde{y}_j$  and to initialize the internal MSA with  $\beta_j = 2\tilde{y}_j/\sigma^2$ . The experiment showed slightly worse performance for most of the points than the original one as seen in Fig. 4.23. Next, an experiment with compensated approximate gradient will be introduced.



-•- Newton (5, 5) + MSA initialized with  $\gamma_j$  —•— Newton (5, 5) + MSA initialized with  $\beta_j$

**Figure 4.23:** A Test of IP Decoder with Internal MSA Initialized with Different LLR for Code CCSDS ( $n = 32, k = 16$ )

### Approximate Gradient with Compensation

A possible idea is to improve the approximate gradient. As the author also mentioned in [Wad10], the new search candidate calculated with approximate gradient "may not converge to the optimal point". The approximation was on term

$$\sum_{S \in T_i} \tau_j^{(i,S)}(\mathbf{c}),$$

which is a component of the exact gradient as shown in Eq. (3.24). It is reduced to

$$\tau_j^{(i,S^{(i)})}(\tilde{\mathbf{c}}) := \frac{I[j \in A(i) \setminus S^{(i)}] - I[j \in S^{(i)}]}{1 + \sum_{l \in S^{(i)}} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S^{(i)}} \tilde{c}_l}, \quad (4.3)$$

where

$$S^{(i)} := \arg \max_{S \in T_i} \left[ 1 + \sum_{l \in S} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S} \tilde{c}_l \right] \quad (4.4)$$

For the convenience of explanation, we define

$$D := 1 + \sum_{l \in S} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S} \tilde{c}_l, \quad (4.5)$$

for it is the denominator. In [Wad07, Wad10]  $\tau_j^{(i, S^{(i)})}(\tilde{\mathbf{c}})$  is found in the following steps:

1. Separate VNs into 2 groups to maximize  $D$  :

$$\begin{cases} \text{outside } S^{(i)}, & \text{if } \tilde{c}_l - 1 \leq \tilde{c}_l, \\ \text{inside } S^{(i)}, & \text{if } \tilde{c}_l - 1 > \tilde{c}_l. \end{cases}$$

2. If the number of VNs inside  $S^{(i)}$  is an even number, find a VN with the least cost to switch side between inside  $S^{(i)}$  and outside  $S^{(i)}$ , namely

$$\tilde{c}_{l, \min} = \arg \min_{l \in A(i)} |2\tilde{c}_l - 1|. \quad (4.6)$$

3. Correct  $D$  with difference caused by swapping the VN as

$$D_{\max} = D - \min |2\tilde{c}_l - 1| \quad (4.7)$$

As required in Eq. (3.15),  $D$  is a negative value for all interior points. With the maximum negative denominator, the combination  $S^{(i)}$  occupies a large proportion in  $\sum_{S \in T_i} \tau_j^{(i, S)}(\mathbf{c})$  if the numerator isn't zero. As a result, choosing  $\tau_j^{(i, S^{(i)})}(\tilde{\mathbf{c}})$  to represent the sum of all combinations may cause a bias over positive or negative side. Therefore, an idea for improvement is to add a correction factor to compensate the approximate gradient in order to offset the possible bias.

Regarding low complexity two simple compensation strategies were considered:

- Compensate with a combination whose  $\tau_j^{(i, S)}(\mathbf{c})$  has the opposite sign of  $\tau_j^{(i, S^{(i)})}(\mathbf{c})$ . Disadvantage is if the value is small in comparison to  $\tau_j^{(i, S^{(i)})}(\mathbf{c})$ , the effect of compensation would be minor.
  - Compensate with a combination whose  $\tau_j^{(i, S)}(\mathbf{c})$  also has a relative big share in  $\sum_{S \in T_i} \tau_j^{(i, S)}(\mathbf{c})$ . Disadvantage is that if the sign is the same as  $\tau_j^{(i, S^{(i)})}(\mathbf{c})$ , the bias would be enhanced instead of being weakened.
1. Compensate with a combination whose  $\tau_j^{(i, S)}(\mathbf{c})$  has the opposite sign of  $\tau_j^{(i, S^{(i)})}(\mathbf{c})$ .

As we analyze  $\tau_j^{(i, S^{(i)})}(\tilde{\mathbf{c}})$ , we see that the numerator is 1 for VNs outside  $S^{(i)}$ , -1 for VNs inside  $S^{(i)}$  and 0 for other indices except for  $A(i)$ . The denominator is  $D_{\max}$  under this combination, which is the maximum negative value among all combinations. We know that for any combination the denominator is always negative and the numerator can be

positive or negative as the combination changes. Thus, the sign of numerator dominates the sign. Since  $S$  has odd number of elements, for a CN with even CN degree, the possible combinations are symmetric. For instance, for a CN with  $d_C = 6$ , the number of elements inside  $S$  can only be 1, 3 or 5. The number of elements outside  $S$  would be 5, 3 or 1, respectively. The implementation of compensation with the opposite sign is very simple. By exploiting the symmetric characteristic, we swap the VNs inside  $S^{(i)}$  with the VNs outside, which is also a valid combination in  $T_i$  with the opposite sign in numerator.

Let's name the numerator of  $\tau_j^{(i,S^{(i)})}(\tilde{\mathbf{c}})$  as  $N^{S^{(i)}}$  and the denominator as  $D^{S^{(i)}}$ . In addition, we name the compensation combination  $S^1$ , numerator and denominator of its  $\tau_j^{(i,S^1)}(\tilde{\mathbf{c}})$  as  $N^1$  and  $D^1$  respectively. Then we know the relation between  $N^{S^{(i)}}$  and  $N^1$  is

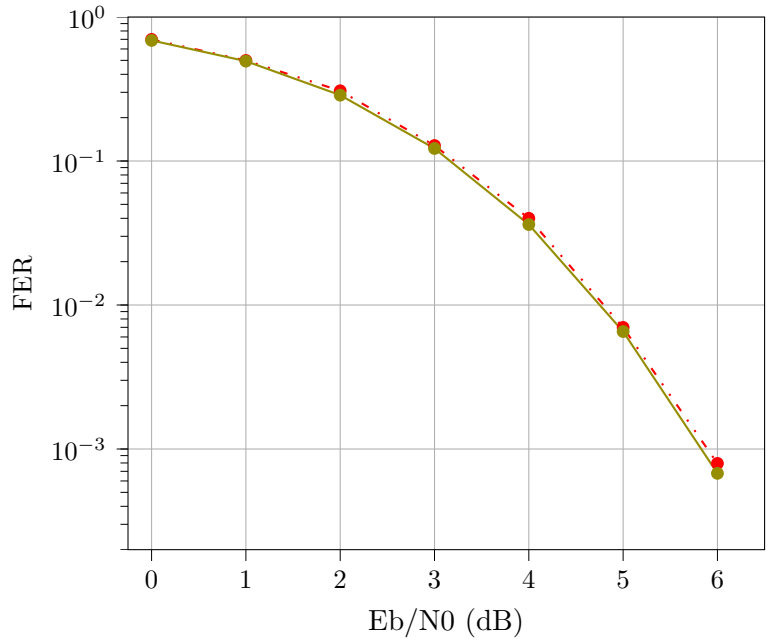
$$N^1 = -N^{S^{(i)}}. \quad (4.8)$$

Concerning  $D^1$ , the VNs inside  $S^{(i)}$  are outside  $S^1$  and vice versa. Therefore, it holds

$$\begin{aligned} D^1 &= 1 + \sum_{l \in S^1} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S^1} \tilde{c}_l \\ &= 1 + \sum_{l \in A(i) \setminus S^{(i)}} (\tilde{c}_l - 1) - \sum_{l \in S^{(i)}} \tilde{c}_l \\ &= 1 + \sum_{l \in A(i) \setminus S^{(i)}} \tilde{c}_l - |A(i) \setminus S^{(i)}| - \sum_{l \in S^{(i)}} \tilde{c}_l \\ &= 1 + \sum_{l \in A(i) \setminus S^{(i)}} \tilde{c}_l - |A(i) \setminus S^{(i)}| - \sum_{l \in S^{(i)}} \tilde{c}_l + |S^{(i)}| - |S^{(i)}| \\ &= 1 - |A(i) \setminus S^{(i)}| - |S^{(i)}| - \left( \sum_{l \in S^{(i)}} \tilde{c}_l - |S^{(i)}| - \sum_{l \in A(i) \setminus S^{(i)}} \tilde{c}_l \right) \\ &= 1 - |A(i)| - \left( \sum_{l \in S^{(i)}} (\tilde{c}_l - 1) - \sum_{l \in A(i) \setminus S^{(i)}} \tilde{c}_l \right) \\ &= 1 - |A(i)| - (D^{S^{(i)}} - 1) \\ &= 2 - |A(i)| - D^{S^{(i)}} \end{aligned} \quad (4.9)$$

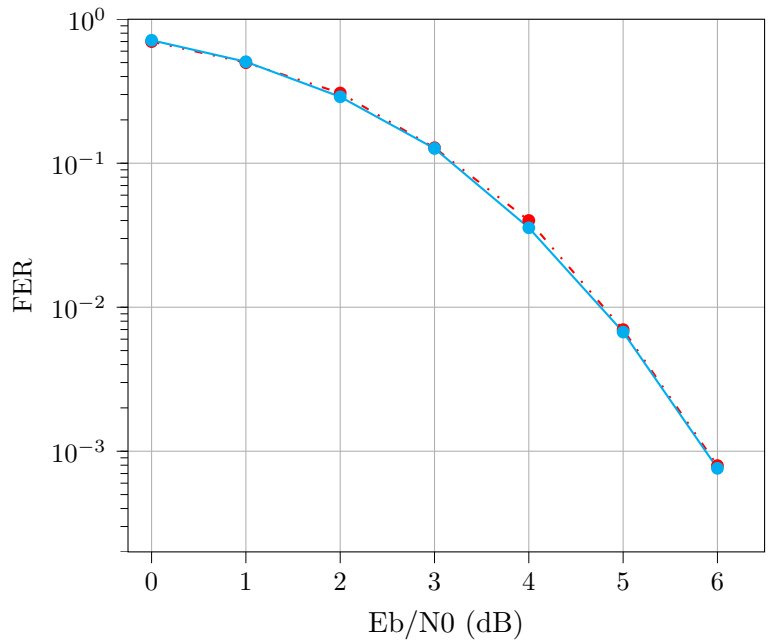
Thus, the compensation term can simply be calculated from  $D^{S^{(i)}}$  and cost only one more computation. Sadly, the compensation wasn't as efficient as expected. Fig. 4.24 shows the average result from 8 realizations, there is consistent but negligible improvement by compensating with a term of the opposite sign.

2. Compensate with a combination whose  $\tau_j^{(i,S)}(\mathbf{c})$  also has a relative big share. Because maximum and minimum are the most easiest terms to find. Another valid combination is introduced by  $D^2 = D_{\max} - \max |2\tilde{c}_l - 1|$ .  $D^2$  is not as big as  $D^{S^{(i)}}$  but is still relatively big compared to  $D$  from other combinations. Moreover,  $N^2$  can be found easily analogous to  $N^{S^{(i)}}$ , the only difference is to find  $\tilde{c}_{l,\max}$  instead of  $\tilde{c}_{l,\min}$ . The average result from 7 realizations, as shown in Fig. 4.25, hardly shows any improvement in comparison to the curve without compensation.



-•- Newton (5, 5) + MSA —•— Newton (5, 5) (Compensated with Method 1) + MSA

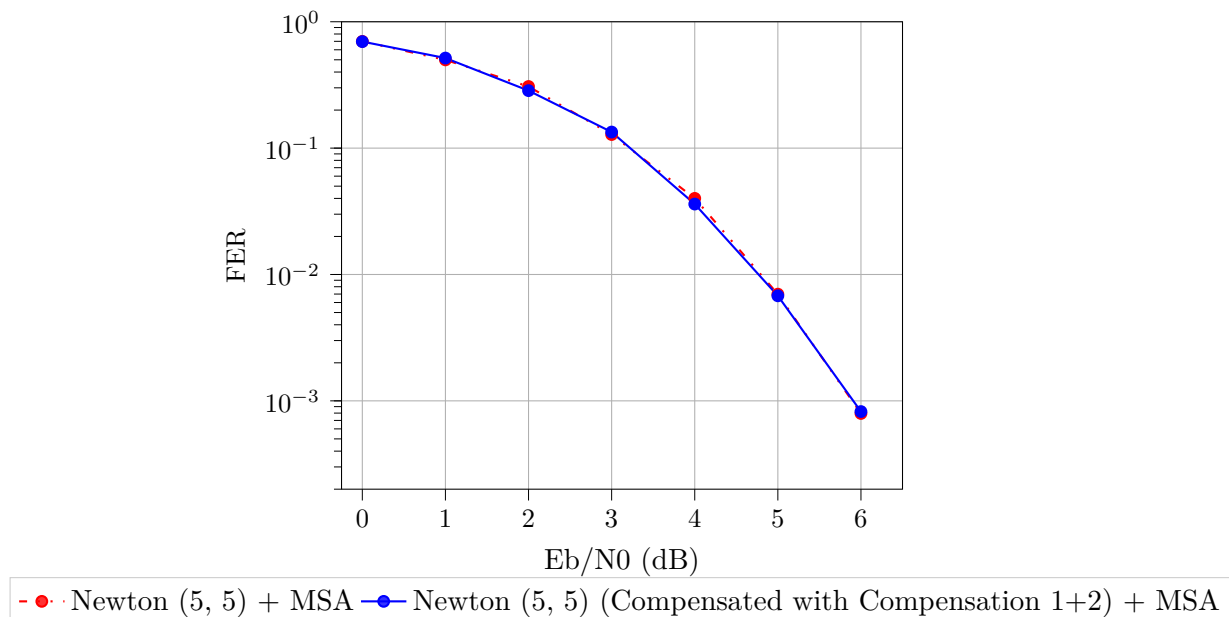
**Figure 4.24:** Result with Compensation 1 for Code CCSDS ( $n = 32$ ,  $k = 16$ )



-•- Newton (5, 5) + MSA —•— Newton (5, 5) (Compensated with Method 2) + MSA

**Figure 4.25:** Result with Compensation 2 for Code CCSDS ( $n = 32$ ,  $k = 16$ )

Out of curiosity, if both compensation combinations are added, we see the average result from 8 trials in Fig. 4.26. It is difficult to see but it is slightly worse than applying each compensation alone from 5 dB on.



**Figure 4.26:** Result with Compensation 1+2 for Code CCSDS ( $n = 32$ ,  $k = 16$ )

It should be noted that the accuracy of data shown in figures is unfortunately not guaranteed, since the results came from the average of only a few trials. The statement made based on the limited samples might be wrong. Besides, in order to create the same operating environment as much as possible, the simulations were completed in software Spyder (Python 3.9) on an Laptop with processor Intel® Core™ i7-8550U CPU @ 1.80GHz  $\times$  8 with OS Ubuntu 22.04.1 LTS. However, there may still be bias or errors in the experiment. More samples are needed to obtain more accurate data.

### Future Work

Due to the time limit, more exploration couldn't be realized. But some ideas might be worth trying for the future work. For example, the soft information from internal BP decoders  $\eta_j$  contains LLR and information from the message exchange between VNs and CNs. Maybe we can somehow make use of it and feed it back to the inner loop. Other than the barrier method, there are possibilities to reform the merit function proposed in IP decoding into its dual problem and to solve with other methods. Since there are in total 3 components inside the overall merit function, namely the objective function, a logarithmic function for parity constraint and a logarithmic function for box constraint. The overall merit function could be divided differently and solved by applying, for example, decomposition methods with Lagrange multipliers or ADMM [BPC<sup>+</sup>11]. What's more, if a more accurate method is applied with less approximation, there might be no need for an internal decoder. For instance, an exact primal-dual IP method was suggested in [Wad09] and a low-complexity IP method in combination with ALP decoding was presented in [TSS11]. [Von08] has also mentioned a few IP algorithms for solving LP decoding problems.

## 4.2. Proximal Decoding Simulation Results and Analysis

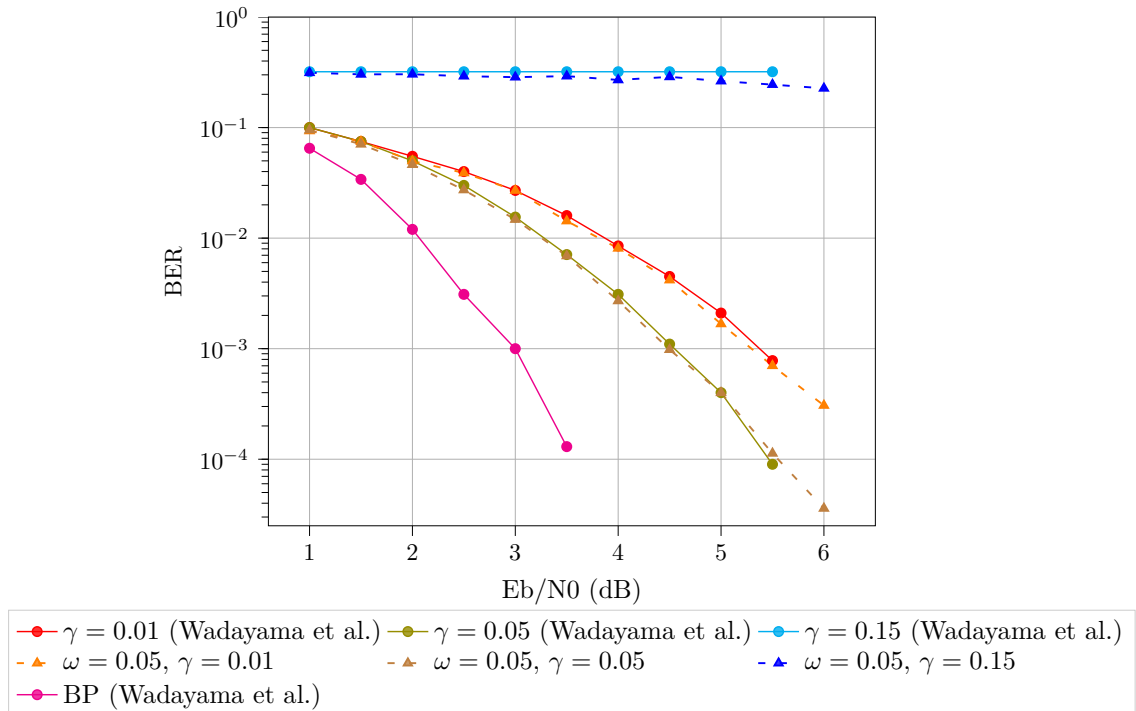
This part realizes and validates the proximal decoding [WT21, WT22]. The simulation results will be compared with the previous studies. The influence of significant variables will be looked into by conducting experiments with controlled variables. Finally, ideas for performance improvement will be put forward and discussed.

### 4.2.1. Comparison of Results

The parameters for simulation are as follows:

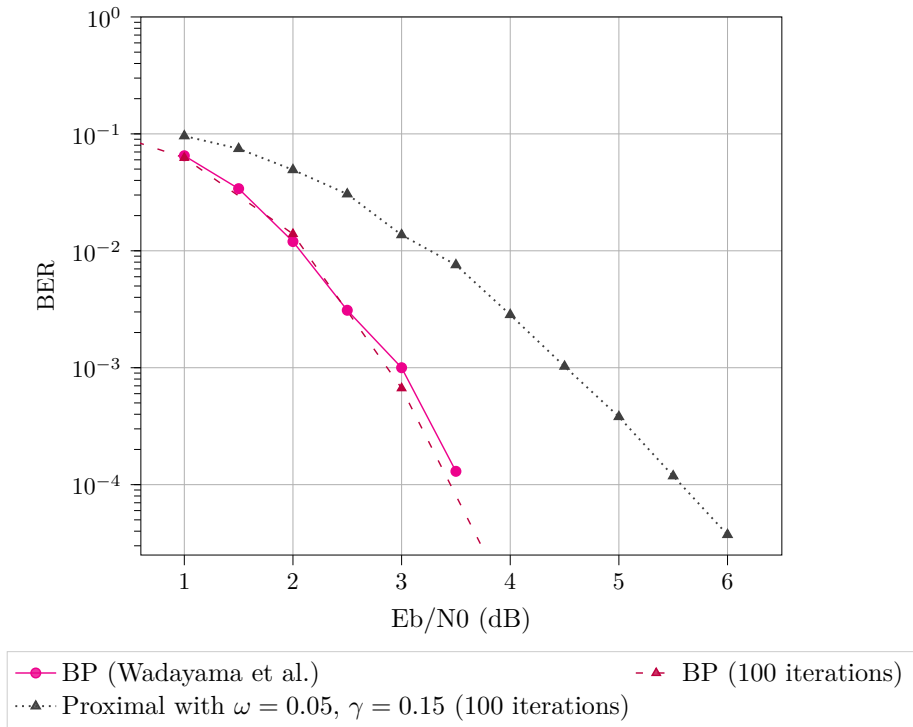
- Code: (3,6)-regular LDPC code named MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ ) [Mac].
- Parameters:  $\omega = 0.05$  (not given in [WT21, WT22]).
- Starting point:  $s^0 = \mathbf{0}$ .
- Box projection:  $B_\eta := [-1.5, 1.5]^n$ .
- Maximum number of iterations  $O_{\max} = 200$  (not given in [WT21, WT22]).

Some parameters are not given in the the previous studies. These were determined through experiments, which will be discussed in depth afterwards. The comparison of results is shown in Fig. 4.27. Additionally, the BP decoder in Fig. 4.27 is compared with my implementation of SPA in Fig. 4.28, which is used to compare decoding throughput with proximal decoder in Fig. 4.29.

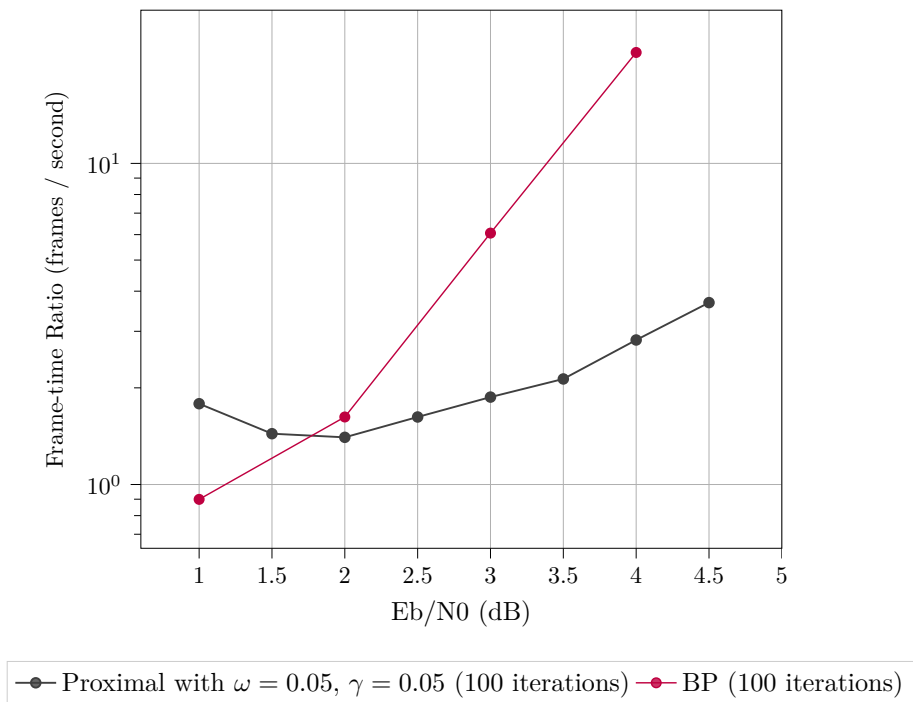


**Figure 4.27:** Comparison of BER Results for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )





**Figure 4.28:** Performance Comparison of BP and Proximal Decoder for MacKay 204.33.484 ( $n = 204, k = 102$ )



**Figure 4.29:** Speed Comparison of BP and Proximal Decoder for MacKay 204.33.484 ( $n = 204, k = 102$ )

### 4.2.2. Influence of $\omega$ and $\gamma$

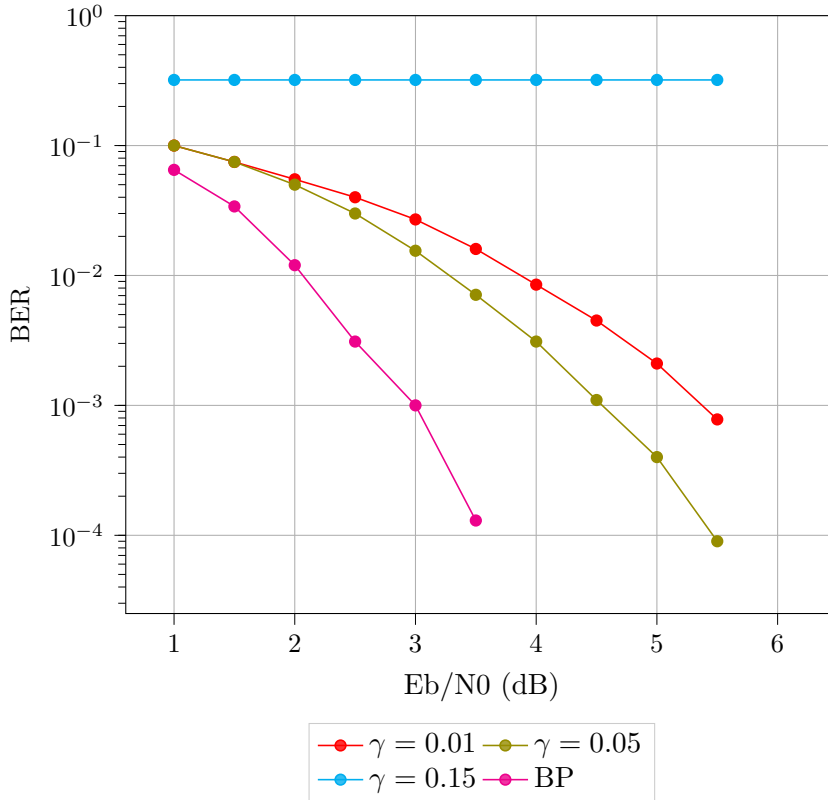
Firstly, let's review the minimizing processes

$$\begin{aligned}\mathbf{r}^{k+1} &:= \mathbf{s}^k - \omega \nabla f(\mathbf{s}^k), \\ \mathbf{s}^{k+1} &:= \mathbf{r}^{k+1} - \gamma \nabla h(\mathbf{r}^{k+1}),\end{aligned}\tag{4.10}$$

where

$$\begin{aligned}f(\mathbf{x}) &:= \|\mathbf{y} - \mathbf{x}\|^2, \\ h(\mathbf{x}) &:= \sum_{j \in \mathcal{J}} (x_j^2 - 1)^2 + \sum_{i \in \mathcal{I}} \left( \left( \prod_{j \in A(i)} x_j \right) - 1 \right)^2.\end{aligned}\tag{4.11}$$

Parameter  $\omega$  and  $\gamma$  are crucial for the proximal decoder, for they control the convergence behavior. By analyzing update steps, we see that  $\omega$  regulates the step size towards a candidate closer to received signal  $\mathbf{y}$  whereas  $\gamma$  controls the step size towards a point with lower parity-check cost, namely lower  $h(\mathbf{x})$ . However, there seems not to be an analytical method to determine the best values of  $\omega$  and  $\gamma$ . [WT22] has found out through experiments that  $\gamma = 0.05$  showed the best BER performance for AWGN channel with (3,6)-regular LDPC code ( $n = 204$ ,  $k = 102$ ).



**Figure 4.30:** BER Results from [WT22] for (3,6)-Regular LDPC Code with  $n = 204$ ,  $k = 102$

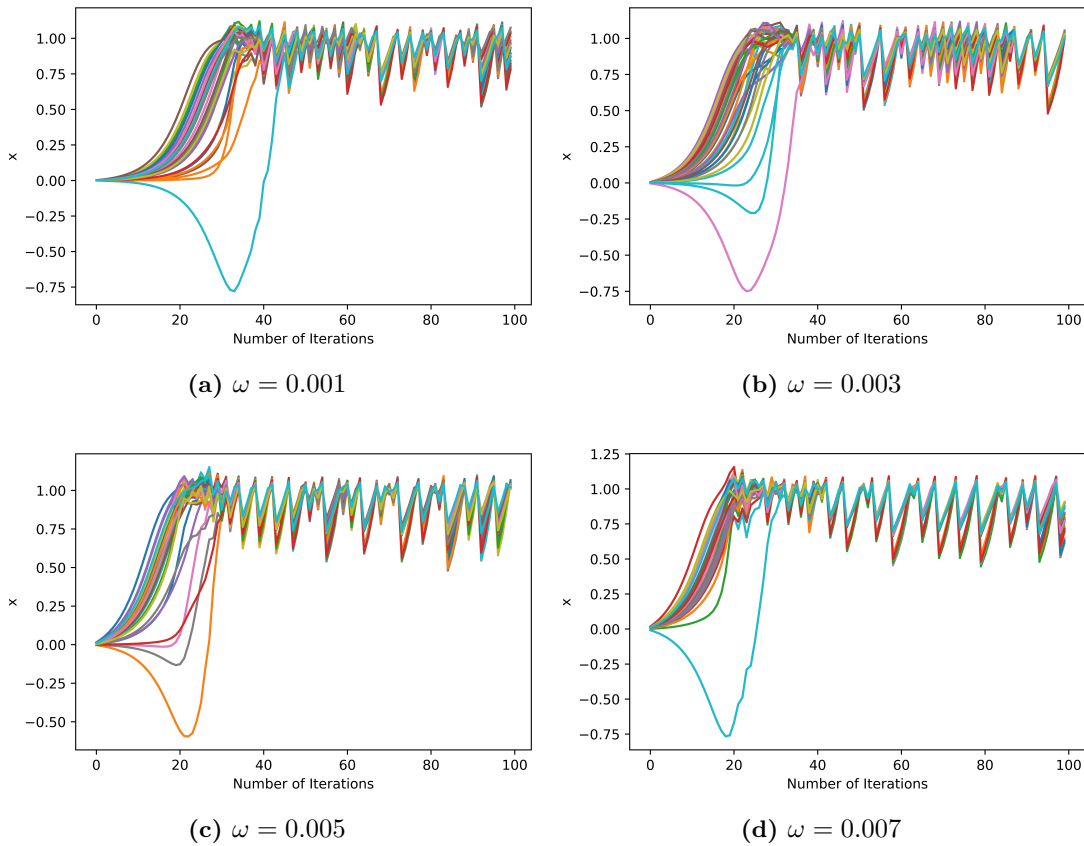
As seen in Fig. 4.30, when  $\gamma$  is larger than 0.1, BER doesn't improve at all as SNR increases. Nevertheless, below 0.1, the curve with  $\gamma = 0.05$  performs better than the

curve with  $\gamma = 0.01$ . Information about  $\omega$  for AWGN channel was not given in [WT22]. A comparison with identical parameters is not possible. Therefore, in this subsection, we will explore the influence of  $\omega$  and  $\gamma$  comprehensively. Short code CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ ) is used for simulations.

First of all, we fix  $\gamma$  at 0.05 to investigate the influence of  $\omega$ . The setting is listed as follows:

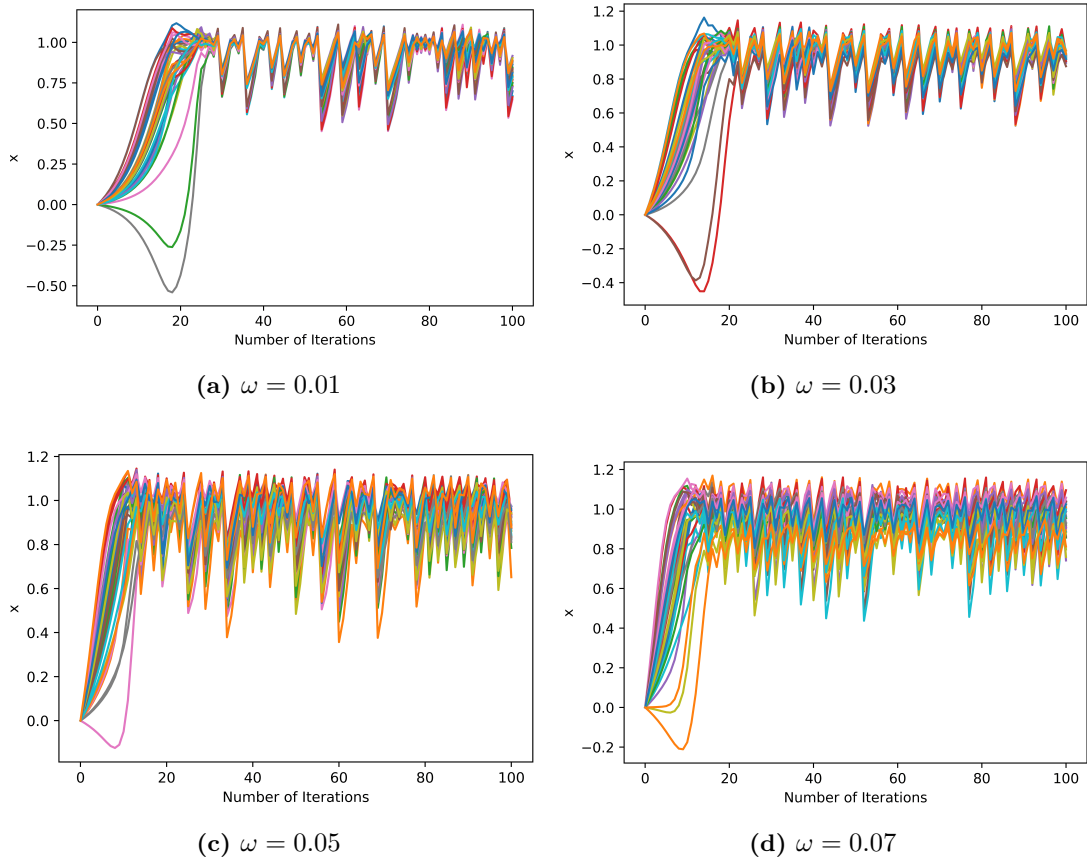
- $s^0 = \mathbf{0}$ .
- $\gamma = 0.05$ .
- SNR = 4 dB.
- Box projection:  $B_\eta := [-1.5, 1.5]^n$ .
- Code: LDPC code named CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ ).
- Maximum number of iterations  $O_{\max} = 100$  without exit option.

In order to see a development from the starting point, the value of iteration 0 in all the following plots is set to  $S^0$ , i.e.,  $\mathbf{0}$ . Fig. 4.31 shows the behavior of very small  $\omega$ . All curves go towards 1 but oscillate around 1 between 0.5 and 1.1. Curves with bigger  $\omega$  reach 1 faster.



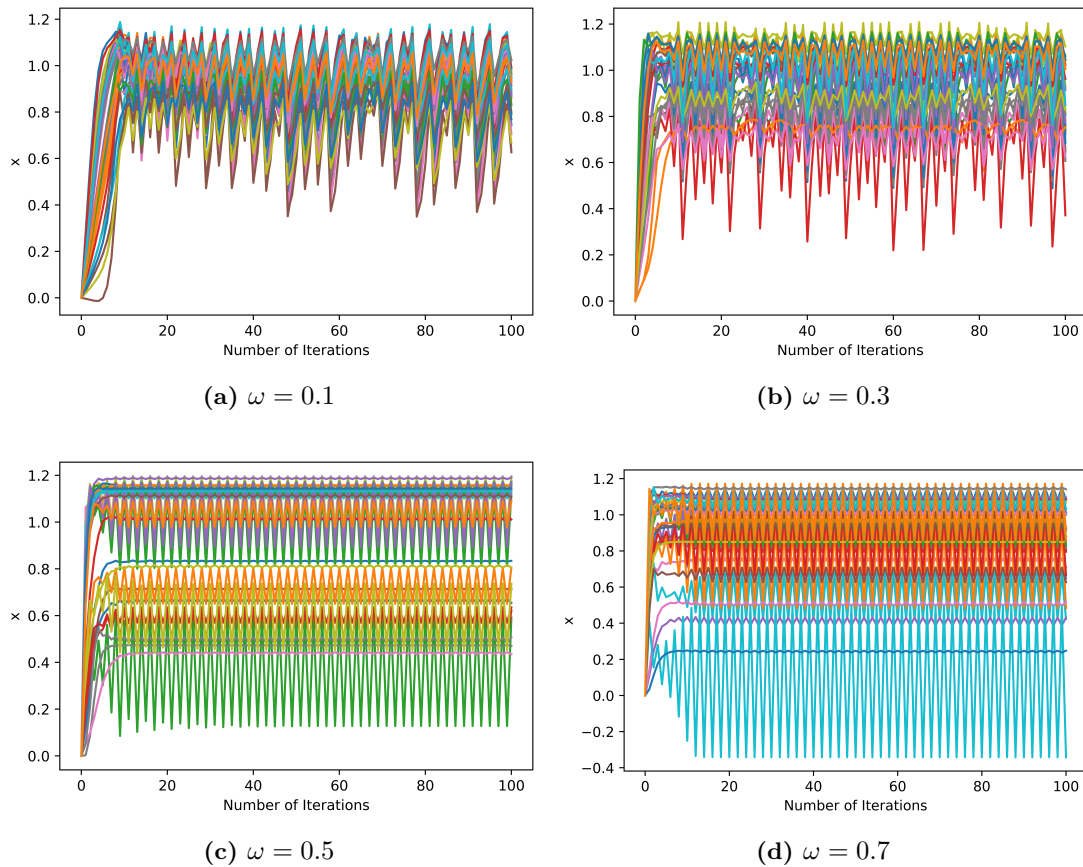
**Figure 4.31:**  $\omega$  below 0.01,  $\gamma = 0.05$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32, k = 16$ )

In Fig. 4.32, the development of curves with  $\omega$  between 0.01 and 0.1 is similar to the situation where  $\omega$  is below 0.01. The curves also oscillate around 1 between 0.5 and 1.1. Besides, the curves reach 1 for the first time even sooner. Curves with  $\omega$  below 0.05 have relative lower frequency of oscillation and stay more together to each other than curves with  $\omega$  above 0.05.



**Figure 4.32:**  $\omega$  between 0.01 and 0.1,  $\gamma = 0.05$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )

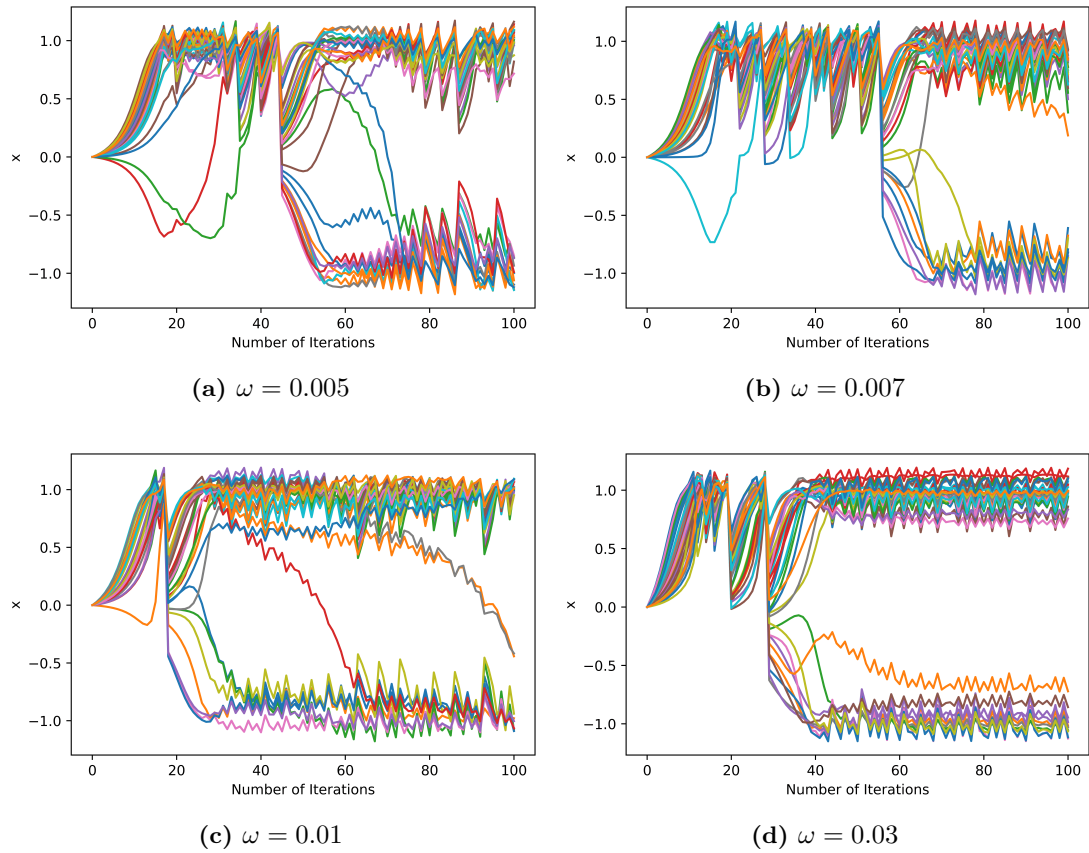
Interesting phenomenon happened when  $\omega$  is increased to be between 0.1 and 1, as shown in Fig. 4.33. The curves don't incline to be near 1 anymore and show instability. They either oscillate intensively over a large range or converge extremely fast after only a few iterations to a random value.



**Figure 4.33:**  $\omega$  between 0.1 and 1,  $\gamma = 0.05$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )

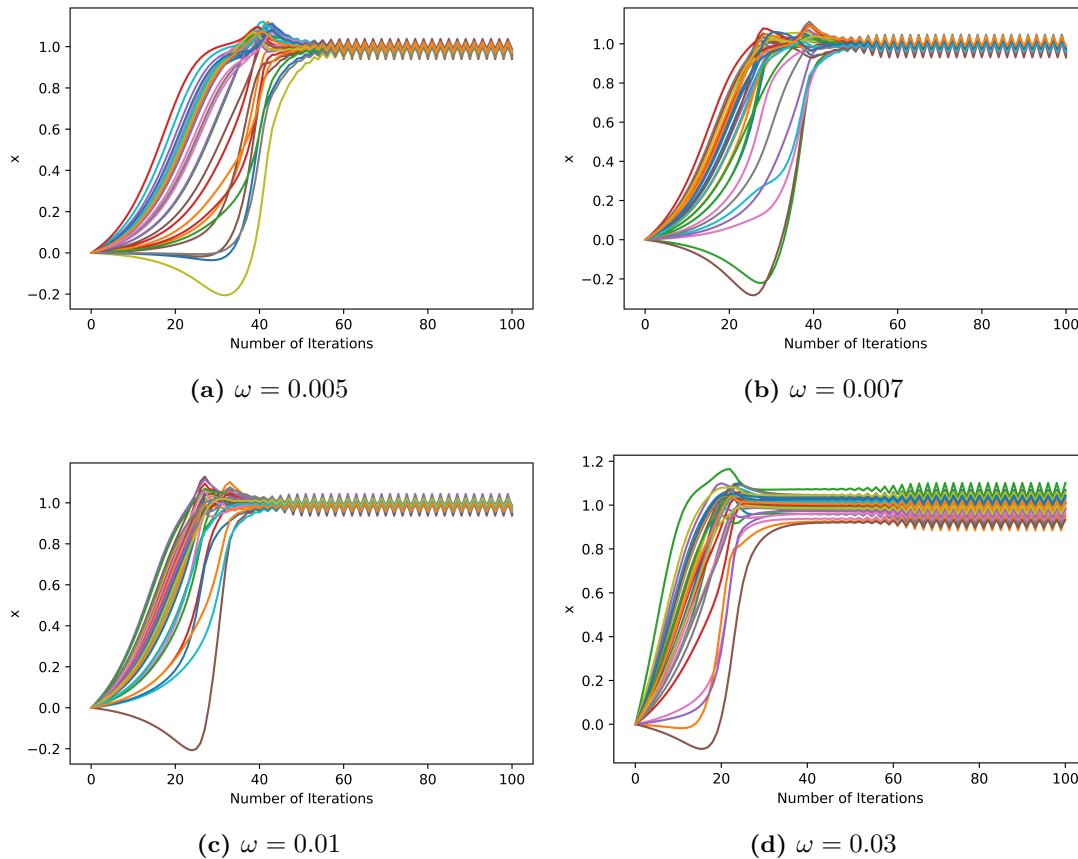
As we see from figures above with  $\gamma = 0.05$ , even though they all oscillate,  $\omega$  between 0.005 and 0.03 show better performance compared to other configurations. They converge fast within 20-30 iterations while having less frequent oscillation. Furthermore, curves with lower  $\omega$  in the range between 0.005 and 0.03 converge slower but oscillate less frequently with more focus near 1. Now we concentrate on  $\omega$  between 0.005 and 0.03 while changing  $\gamma$  a little bit to decide which combination performs better.

We observe from Fig. 4.34 that increasing  $\gamma$  from 0.05 to 0.07 is not a good idea. The curves oscillate extremely and cross the threshold of HD, i.e.,  $\mathbf{0}$ . Therefore, we change  $\gamma$  back to be smaller than 0.05.



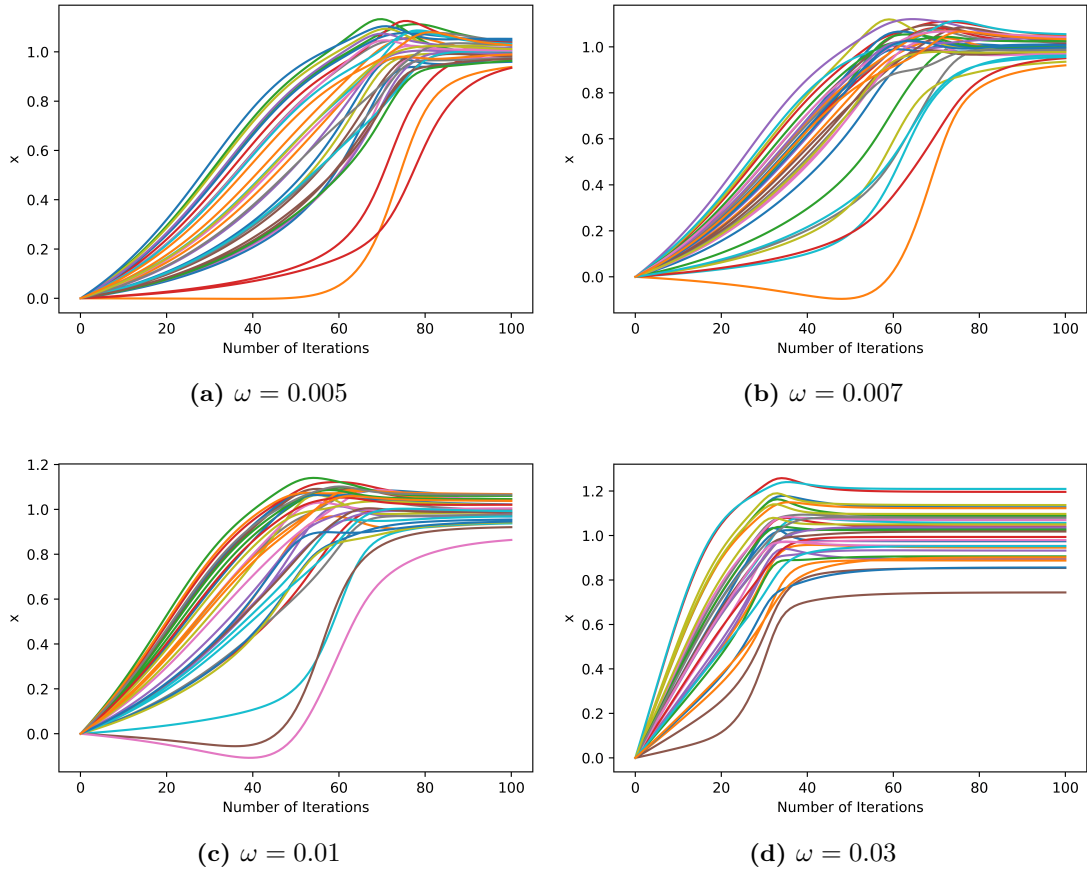
**Figure 4.34:**  $\omega$  between 0.005 and 0.03,  $\gamma = 0.07$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )

All the configurations in Fig. 4.35 show quite good performance regardless of an oscillation within  $[0.9, 1.1]$ , which is a smaller range than  $[0.5, 1.1]$  by  $\gamma = 0.05$ . However, with  $\gamma = 0.03$  the curves with  $\omega$  between 0.05 and 0.03 need 30-50 to converge instead of 20-30 by  $\gamma = 0.05$ . This comparison shows a trade-off between converging speed and accuracy (if required). In general,  $\omega = 0.03$  behaves better, for the curves converge within 30 iterations while not having much more noise compared to the other 3 settings.



**Figure 4.35:**  $\omega$  between 0.005 and 0.03,  $\gamma = 0.03$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )

Fig. 4.36 shows no more oscillation. However, the converging speed of all 4 configurations is greatly reduced. The number of iterations required for convergence is more or less doubled.



**Figure 4.36:**  $\omega$  between 0.005 and 0.03,  $\gamma = 0.01$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )

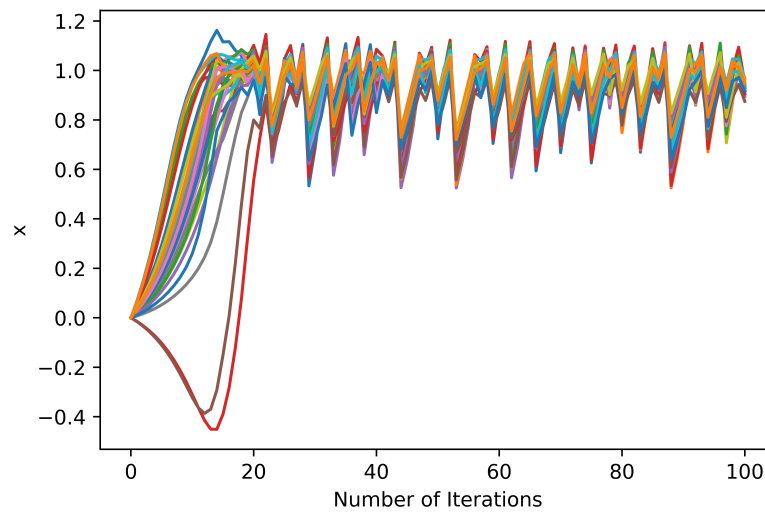
After comparing combinations of various  $\omega$  and  $\gamma$ , some conclusions can be drawn concerning their influence to decoding performance for code CCSDS [HSG<sup>+</sup>19] ( $n = 32$ ,  $k = 16$ ). For  $\omega$  above 0.1 and  $\gamma$  above 0.05 the oscillation threatens decoding accuracy since the range of oscillation is near or beyond zero, which is the threshold for HD. Small range of oscillation close to 1 and -1 doesn't have impact on decoding accuracy but might be a potential cause of instability. Under 0.05, a smaller  $\gamma$  can oppress oscillations better than a bigger one. To avoid any oscillation,  $\gamma$  must be small, for example, below 0.03. However, a smaller  $\gamma$  requires a lot more iterations. A bigger  $\gamma$  below or equal to 0.05 is more beneficial since the converging speed is more crucial to decoding, provided that the small range of oscillation is acceptable for making HD.

In addition,  $\omega$  controls how scattered the curves are around 1 and -1, namely the accuracy of converged end value. With a bigger  $\omega$  (below 0.1), the lines look more scattered, especially with a smaller  $\gamma$ , as seen in the comparison of Fig. 4.36 and Fig. 4.35. Parameter  $\omega$  is also related to the speed of convergence to some extent, but not as significantly as  $\gamma$  is. For instance, as shown in Fig. 4.35, with  $\gamma = 0.03$ , to set  $\omega = 0.03$ , three times as large as

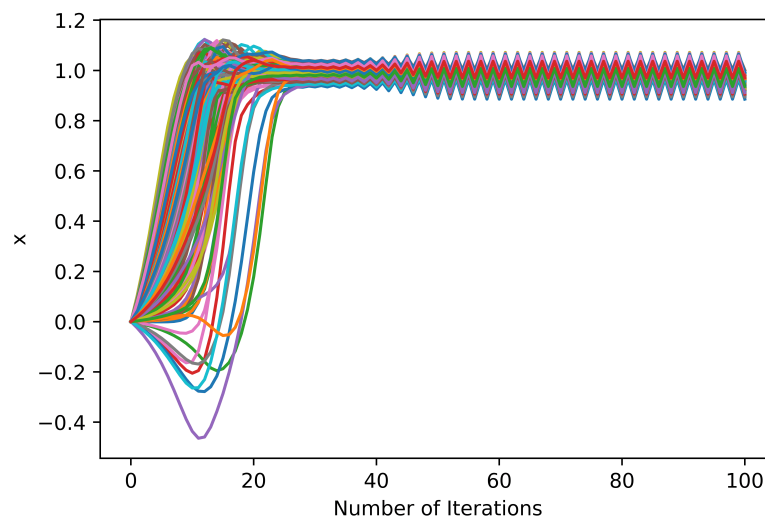


0.01, the curves converge only a few iterations faster than those with  $\omega = 0.01$ . As long as the lines are not too close to 0, namely with a value below 0.1, the accuracy of HD after convergence should not be affected. Nevertheless, if a smaller  $\omega$  doesn't slow down convergence too much, it is safer to choose a relatively small  $\omega$ , such as below 0.05.

In summary,  $\gamma$  dominates the speed of convergence. For a trade-off between speed and oscillation that doesn't affect HD, 0.03 and 0.05 both show advantages. Regarding  $\omega$ , values between 0.01 and 0.05 are stable with reasonable number of iterations. The smaller  $\omega$  is, the more focused the curves are around 1 and -1, at cost of a few more iterations to converge. Under this range, to balance speed and accuracy, a bigger  $\gamma$  with a relatively small  $\omega$  would be better, for example  $\gamma = 0.05, \omega = 0.03$ . See Fig. 4.37 and Fig. 4.38 for the convergence behavior under configuration  $\omega = 0.03, \gamma = 0.05$  for different codes.

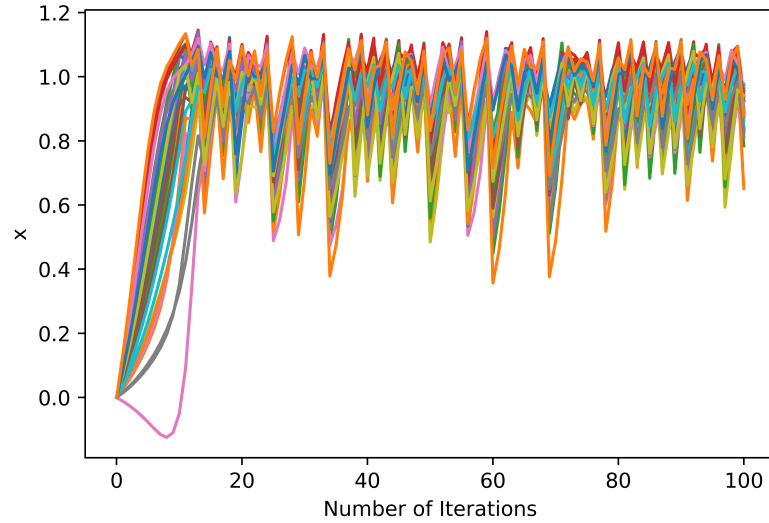


**Figure 4.37:**  $\omega = 0.03, \gamma = 0.05, O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32, k = 16$ )

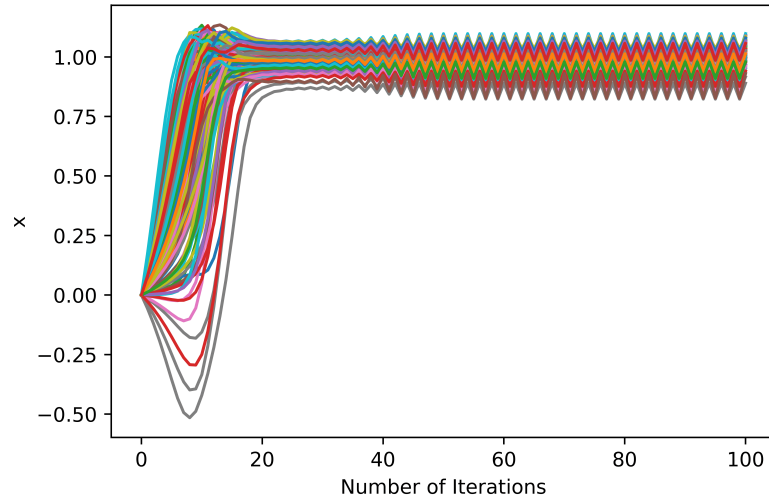


**Figure 4.38:**  $\omega = 0.03, \gamma = 0.05, O_{\max} = 100$  at 4 dB for MacKay 204.33.484 ( $n = 204, k = 102$ )

For the sake of speed and less iterations, convergence behavior under configuration  $\omega = 0.05$ ,  $\gamma = 0.05$  is shown in Fig. 4.39 and Fig. 4.40.



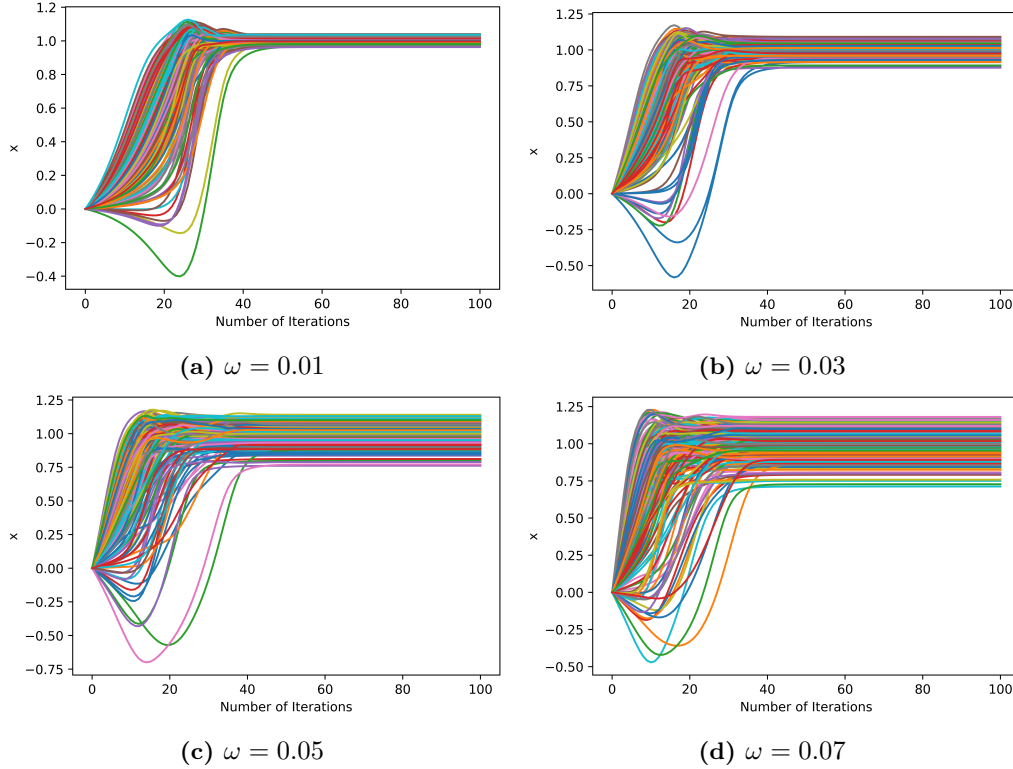
**Figure 4.39:**  $\omega = 0.05$ ,  $\gamma = 0.05$ ,  $O_{\max} = 100$  at 4 dB for Code CCSDS ( $n = 32$ ,  $k = 16$ )



**Figure 4.40:**  $\omega = 0.05$ ,  $\gamma = 0.05$ ,  $O_{\max} = 100$  at 4 dB for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )

It is noticed that the behavior of the same configuration for different codes can also be different. Both arrangements work better with code MacKay 204.33.484 [Mac, HSG<sup>+</sup>19]. It is interesting to observe that the behavior of 204.33.484 ( $n = 204$ ,  $k = 102$ ) with  $\gamma = 0.05$  looks similar to the behavior of CCSDS [HSG<sup>+</sup>19] ( $n = 32$ ,  $k = 16$ ) with  $\gamma = 0.03$ .

Out of curiosity, one more group of experiment was done with  $\omega$  between 0.01 and 0.1 under  $\gamma = 0.03$  for MacKay 204.33.484, as shown in Fig. 4.41. With  $\gamma = 0.03$ , MacKay 204.33.484 doesn't show any oscillation, just like CCSDS with  $\gamma = 0.01$ . As far as observed, it seems that to achieve the same behavior of convergence, bigger  $\omega$  and  $\gamma$  can be chosen for longer code. However, to make a firm statement, more investigations are required.



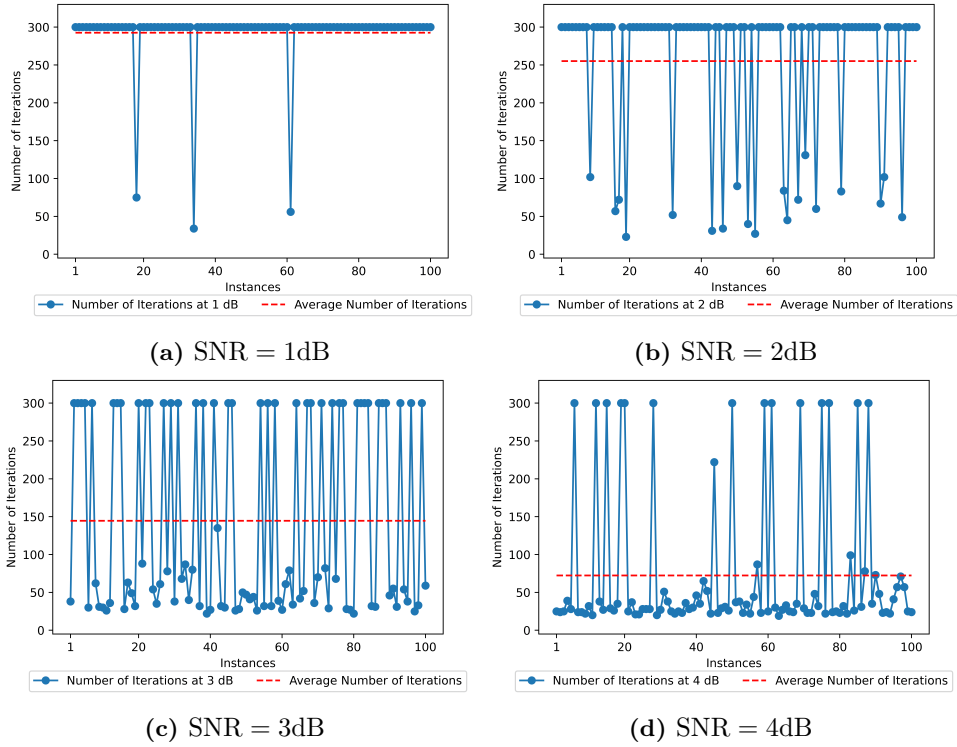
**Figure 4.41:**  $\omega$  between 0.01 and 0.1,  $\gamma = 0.03$ ,  $O_{\max} = 100$  at 4 dB for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )

### 4.2.3. Iterations Required to Converge to Valid Codewords

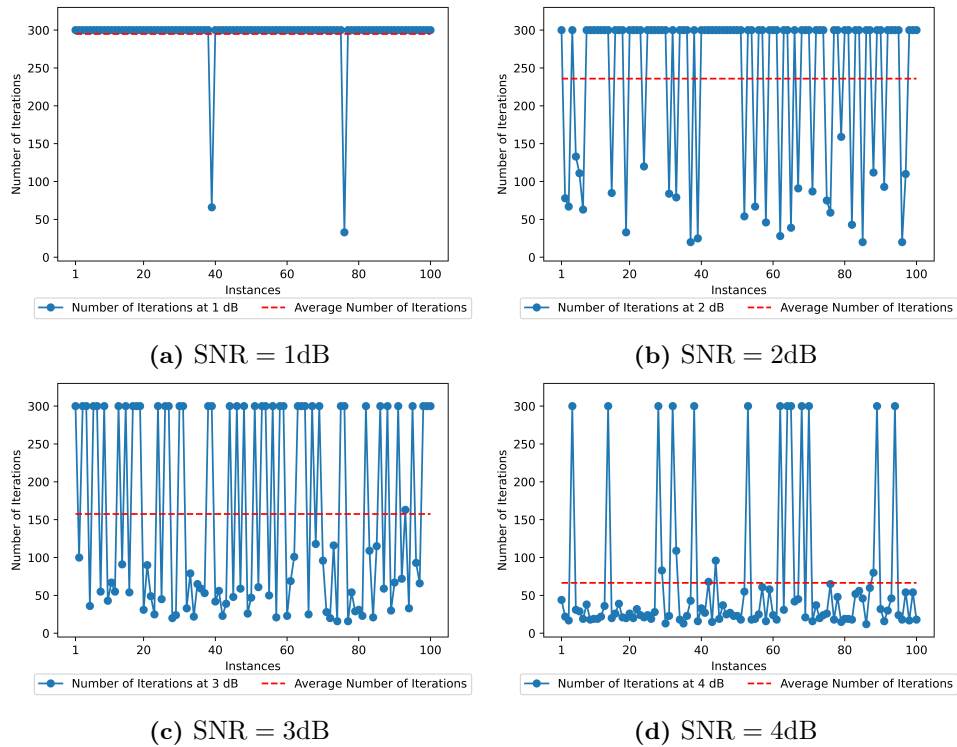
The proximal decoding for AWGN channel was conducted on a (3,6)-regular Mackay LDPC code with  $n = 204$ ,  $k = 102$  in [WT22]. For a comparison, a (3,6)-regular LDPC code called MacKay 204.33.484 [Mac, HSG<sup>+</sup>19] is employed in this thesis. No specification was given in [WT22] regarding the number of iterations. Therefore, the number of iterations required to converge to codewords is investigated in this part. After exploring the convergence behavior under various  $\omega$  and  $\gamma$ , we see that the convergence behavior can be slightly different for short code CCSDS and MacKay 204.33.484. Therefore, the investigation was done on code MacKay 204.33.484 for a more precise results comparison with [WT22].

Firstly, 3 settings with  $\gamma = 0.05$  are explored:

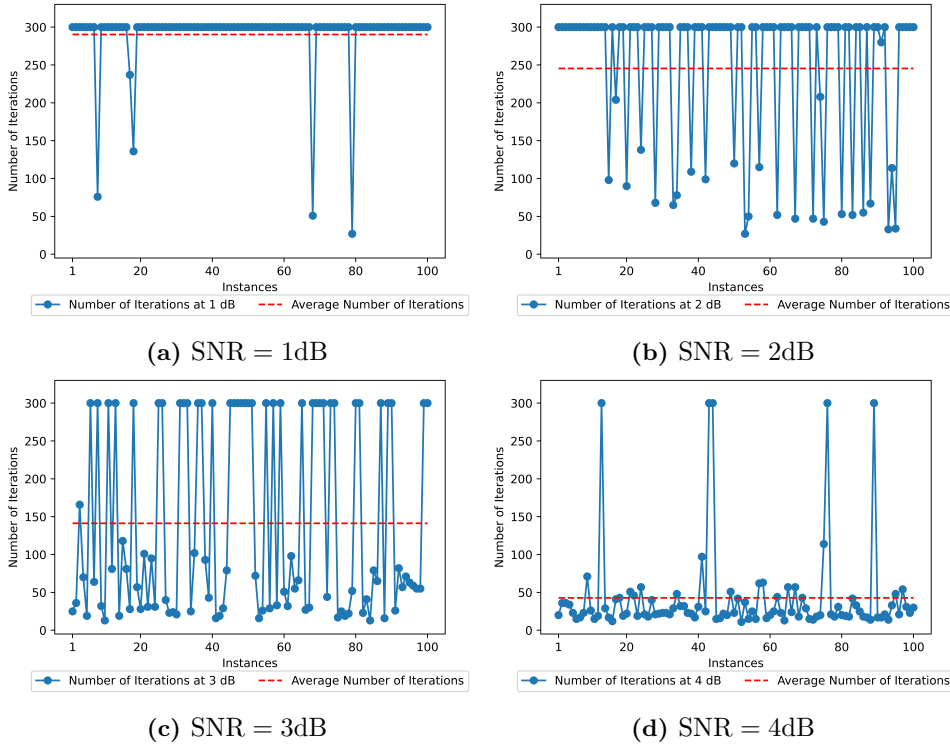
- A:  $\omega = 0.01, \gamma = 0.05$ .
- B:  $\omega = 0.03, \gamma = 0.05$ .
- C:  $\omega = 0.05, \gamma = 0.05$ .



**Figure 4.42:** Required Number of Iterations with Setting A for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )



**Figure 4.43:** Required Number of Iterations with Setting B for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )



**Figure 4.44:** Required Number of Iterations with Setting C for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )

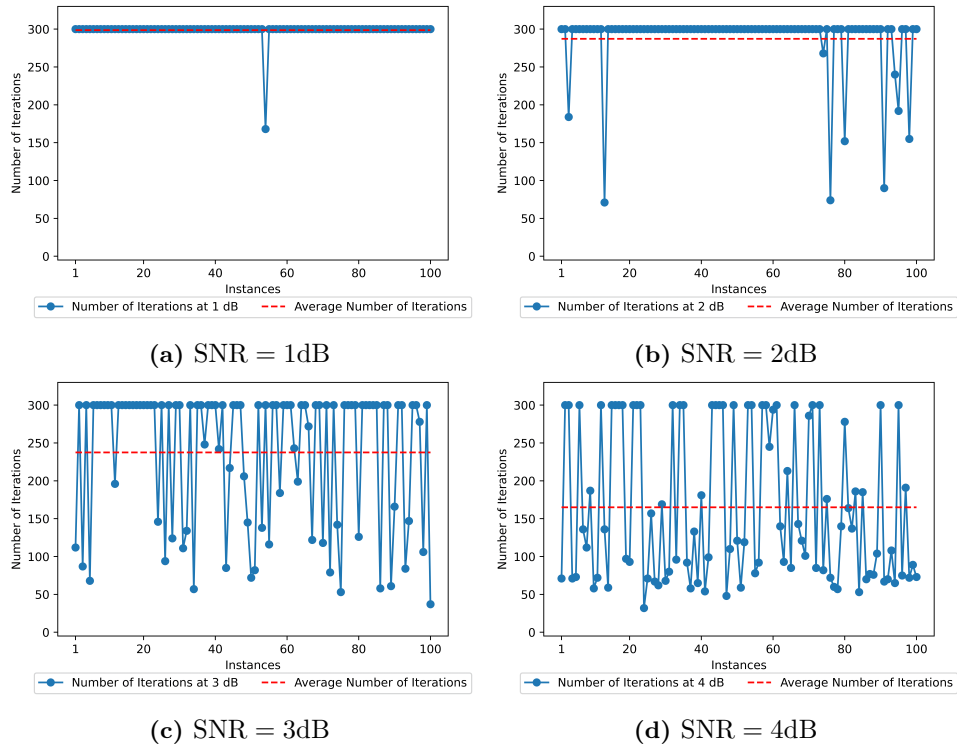
Fig. 4.42 to Fig. 4.44 show the average number of iterations needed for setting A to C to converge to valid codewords based on 100 samples. Within 3 dB, no consistent and apparent advantage can be seen for any setting. At 4 dB, both A and B require more than 50 iterations to converge while setting C requires less than 50, specifically, about 20 iterations less than A and B.

Secondly, 2 settings with  $\gamma = 0.01$  are explored:

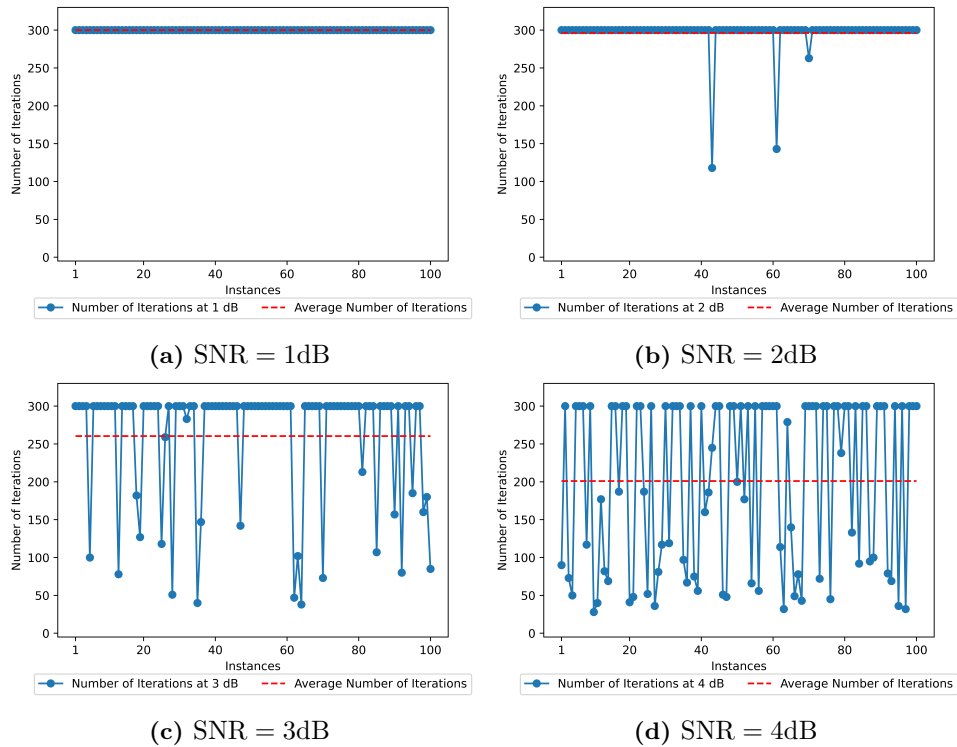
- D:  $\omega = 0.03, \gamma = 0.01$ .
- E:  $\omega = 0.05, \gamma = 0.01$ .

Fig. 4.45 and Fig. 4.46 show the average number of iterations required for setting D and E to converge based on 100 samples. Setting D with  $\omega = 0.03, \gamma = 0.01$  requires generally less iterations than setting E with  $\omega = 0.05, \gamma = 0.01$ . The advantage of setting D over E is more obvious as SNR increases.

In general, both D and E require a lot more iterations than all settings with  $\gamma = 0.05$ . For instance, E requires almost twice as many iterations as by C at 3 dB and almost five times as many at 4 dB. This observation of 100 samples confirms the conclusion made in the last part, namely  $\gamma$  dominates converging speed. Meanwhile, we observe that with a  $\gamma$  as small as 0.01, setting with a smaller  $\omega$  converges faster to codewords, maybe because a smaller  $\omega$  can reduce the spread of distribution and help the decoder to make decisions faster. Nevertheless, configurations with  $\gamma = 0.01$  are far from being optimal.



**Figure 4.45:** Required Number of Iterations with Setting D for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )



**Figure 4.46:** Required Number of Iterations with Setting E for MacKay 204.33.484 ( $n = 204$ ,  $k = 102$ )

#### 4.2.4. Experimental Study on the Performance Improvement

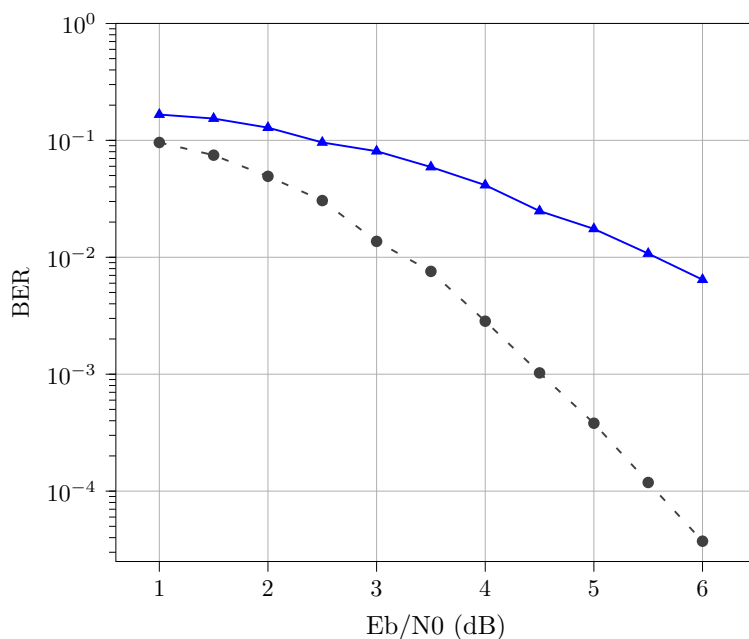
A few experiments were conducted with the intention to improve the performance of the proposed proximal decoding in [WT22]. For instance, an extrapolation between the updates of  $\mathbf{s}$  and  $\mathbf{r}$  as suggested in accelerated proximal gradient method [PB<sup>+</sup>14] was once implemented:

$$\mathbf{s}^{k+1} := \mathbf{s}^k + \beta^k (\mathbf{s}^k - \mathbf{s}^{k-1}), \quad (4.12)$$

, where  $\beta^k$  is the extrapolation parameter:

$$\beta^k = \frac{k}{k+3}. \quad (4.13)$$

However, the implementation didn't reduce execution time or improve the performance. Implementations with varying  $\gamma$  and/or  $\omega$  through line search were also not successful. Besides, if the starting point  $\mathbf{s}^0$  is changed from  $\mathbf{0}$  to  $\mathbf{y}$ , we get worse performance as shown in Fig. 4.47.



-●  $\omega = 0.05, \gamma = 0.05, s^0 = \mathbf{0}$  (100 iterations) —▲  $\omega = 0.05, \gamma = 0.05, s^0 = \mathbf{y}$  (100 iterations)

**Figure 4.47:** Setting  $\mathbf{y}$  as the starting point performs worse than  $\mathbf{0}$  for code MacKay 204.33.484 ( $n = 204, k = 102$ ).

Next, an improvement attempt with ADMM will be presented in detail.

#### Implementation with ADMM

ADMM can be regarded as a special instance of proximal algorithms [PB<sup>+</sup>14, BPC<sup>+</sup>11]. Since proximal gradient was used in proximal decoding, this part explores the possibility of applying ADMM to the derived cost function in [WT21, WT22].

The goal of the proximal decoding is to find

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathcal{X}} (f(\mathbf{x}) + \gamma h(\mathbf{x})). \quad (4.14)$$

The function to minimize can be perfectly divided into two parts, namely

$$f(\mathbf{x}) := \|\mathbf{y} - \mathbf{x}\|^2, \quad (4.15)$$

and

$$h(\mathbf{x}) := \sum_{j \in \mathcal{J}} (x_j^2 - 1)^2 + \sum_{i \in \mathcal{I}} \left( \left( \prod_{j \in A(i)} x_j \right) - 1 \right)^2. \quad (4.16)$$

In the proximal decoding [WT21, WT22], these two parts are also minimized separately in two steps:

$$\begin{aligned} \mathbf{r}^{k+1} &:= \mathbf{s}^k - \omega \nabla f(\mathbf{s}^k), \\ \mathbf{s}^{k+1} &:= \mathbf{r}^{k+1} - \gamma \nabla h(\mathbf{r}^{k+1}). \end{aligned} \quad (4.17)$$

This structure suits perfectly the formulation of standard ADMM [PB<sup>+</sup>14, BPC<sup>+</sup>11], namely

$$\begin{aligned} &\text{minimize} && f(\mathbf{w}) + h(\mathbf{z}), \\ &\text{subject to} && \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{z} = \mathbf{e}, \end{aligned} \quad (4.18)$$

with  $\mathbf{w} \in \mathbb{R}^o$ ,  $\mathbf{z} \in \mathbb{R}^m$ ,  $\mathbf{A} \in \mathbb{R}^{p \times o}$ ,  $\mathbf{B} \in \mathbb{R}^{p \times m}$  and  $\mathbf{e} \in \mathbb{R}^p$ . The iterations include:

$$\mathbf{w}^{k+1} := \arg \min_{\mathbf{w}} \left( f(\mathbf{w}) + \left( \frac{\rho}{2} \right) \|\mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{z}^k - \mathbf{e} + \mathbf{u}^k\|_2^2 \right), \quad (4.19)$$

$$\mathbf{z}^{k+1} := \arg \min_{\mathbf{z}} \left( h(\mathbf{z}) + \left( \frac{\rho}{2} \right) \|\mathbf{A}\mathbf{w}^{k+1} + \mathbf{B}\mathbf{z} - \mathbf{e} + \mathbf{u}^k\|_2^2 \right), \quad (4.20)$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + \mathbf{A}\mathbf{w}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{e}, \quad (4.21)$$

where  $\rho$  is the penalty parameter with  $\rho > 0$  and  $\mathbf{u}$  is the scaled dual variable [BPC<sup>+</sup>11].

We replace the  $\mathbf{w}$  with  $\mathbf{r}$ , the  $\mathbf{z}$  with  $\mathbf{s}$  and introduce a new variable called  $\mathbf{u}$  to adjust the old iteration steps to the ADMM formulation. The new problem is thus defined as:

$$\begin{aligned} &\text{minimize} && f(\mathbf{r}) + h(\mathbf{s}), \\ &\text{subject to} && \mathbf{r} - \mathbf{s} = \mathbf{0}, \end{aligned} \quad (4.22)$$

and the iterations are accordingly

$$\mathbf{r}^{k+1} := \arg \min_{\mathbf{r}} \left( f(\mathbf{r}) + \left( \frac{\rho}{2} \right) \|\mathbf{r} - \mathbf{s}^k + \mathbf{u}^k\|_2^2 \right), \quad (4.23)$$

$$\mathbf{s}^{k+1} := \arg \min_{\mathbf{s}} \left( h(\mathbf{s}) + \left( \frac{\rho}{2} \right) \|\mathbf{r}^{k+1} - \mathbf{s} + \mathbf{u}^k\|_2^2 \right), \quad (4.24)$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + \mathbf{r}^{k+1} - \mathbf{s}^{k+1}. \quad (4.25)$$



Furthermore, in the minimization steps,  $\gamma = 0.05$ ,  $\omega = 0.05$  are employed as step size. Even though ADMM is very similar to the original proximal gradient method, the intention of trying this method is to introduce new parameters, which might regulate better and thus achieve better performance or speed. The iteration steps are derived simply by transforming the cost function from [WT21, WT22] to ADMM formulation in [PB<sup>+</sup>14, BPC<sup>+</sup>11]. The effectiveness is yet to be investigated. The first and biggest problem was to determine the parameters. Due to the non-convexity of the function  $h$  in [WT22], the effectiveness of ADMM is highly dependent on the initialization of  $\rho$  and  $u$  [BPC<sup>+</sup>11].

Different values ranging from 0.001 to 20 with step of 5 were tried. Firstly,  $\mathbf{u} = \mathbf{0}$  was fixed, various  $\rho$  were applied. When a candidate of  $\rho$  appeared,  $\rho$  was then fixed while various  $\mathbf{u}$  were employed. However, this process was not efficient or successful. Inconsistent changes as one parameter was increased or decreased happened. Thus, it was difficult to decide whether a bigger or smaller parameter would perform better. It was observed that with  $\mathbf{u}$  bigger or smaller than  $\mathbf{1}$ , the algorithm was unstable and sometimes didn't converge. It could be related to the non-convexity of the cost function, or there might be mistakes in the implementation. No firm conclusions can be made so far.

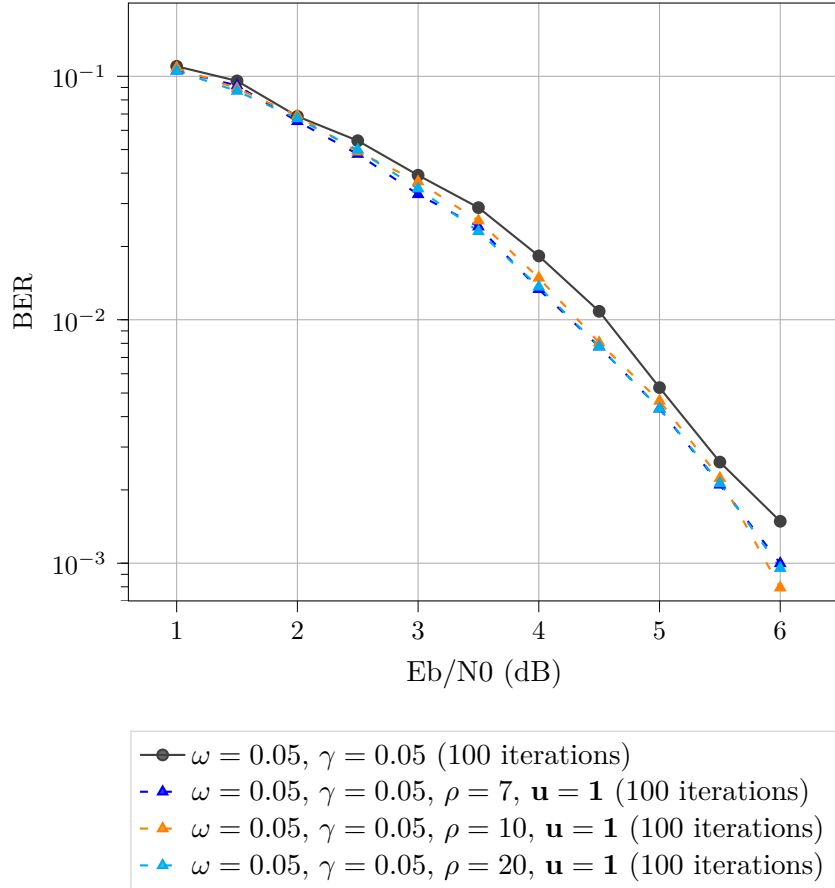
Additionally, the initial value of  $\mathbf{r}$  and  $\mathbf{s}$  are also factors that might affect the result in the implementation. Without knowing where to start, the starting point of the proximal decoding was considered. A "warm start" was employed [BPC<sup>+</sup>11] by initializing  $\mathbf{r}$  and  $\mathbf{s}$  with one gradient step ahead from the starting point of the proximal decoding, namely  $\mathbf{0}$ . To be more specific,

$$\begin{aligned}\mathbf{r}^0 &= \mathbf{0} - \omega \nabla f(\mathbf{0}) \\ \mathbf{s}^0 &= \mathbf{0} - \gamma \nabla f(\mathbf{0})\end{aligned}\tag{4.26}$$

with

$$f(\mathbf{x}) := \|\mathbf{y} - \mathbf{x}\|^2.\tag{4.27}$$

Moreover, it has been observed that it worked better if  $\mathbf{r}$  and  $\mathbf{s}$  were initialized differently. Therefore, after the initialization shown in Eq. (4.26),  $\mathbf{s}$  and  $\mathbf{u}$  were firstly updated, in order to take one more step ahead of  $\mathbf{r}$ . Specifically, the  $\mathbf{r}$  update in Eq. (4.23) was put from the first order to the last. Simulation results for code CCSDS [HSG<sup>+</sup>19] ( $n = 32$ ,  $k = 16$ ) are shown in Fig. 4.48. With  $\mathbf{u} = \mathbf{1}$ , the curve with  $\rho$  above 10, i.e., 20, and the curve with  $\rho$  smaller than 10, i.e., 7, behave almost the same showing slightly better performance than the curve with  $\rho = 10$  except for at 6 dB. At the point when BER is equal to  $10^{-2}$ , ADMM outperforms the proximal gradient method by about 0.25 dB. In general, the algorithm with ADMM is about 0.1-0.2 dB better than proximal decoder. It is possible that with better configuration, the algorithm with ADMM can yield even better results. However, more investigations must be done to confirm this statement.



**Figure 4.48:** Comparison of ADMM and Proximal Gradient Method for Code CCSDS [HSG<sup>+</sup>19] ( $n = 32, k = 16$ )

## Future Work

Choices of parameter  $\gamma$  and  $\omega$  are crucial to the performance of the proximal decoding. Through experiments we have only found a good enough combination of  $\gamma$  and  $\omega$ , because each digital level was only divided into four grades. For example between 0.01 and 0.1, only 0.01, 0.03, 0.05, 0.07 were examined. Moreover, to fully understand the behavior under each value, more samples and data are required. As also suggested in [WT21], deep learning techniques can be applied to optimize the choices of  $\gamma$  and  $\omega$ .

Due to limited time, more variations of ADMM could not be explored with the cost function established in the proximal decoding [WT22], for example with different update orders or adding more updates in one iteration as introduced in [BPC<sup>+</sup>11]. Another interesting implementation of ADMM which is independent of the choice of  $\rho$  and  $u$  was introduced in [ZS13]. It involves Euclidean projection onto convex domain, namely the Poly( $\mathcal{C}$ ). Within convex domain, the problem of initializing  $\rho$  and  $u$  doesn't exist any more.

## 5. Conclusion

In the present thesis, fundamentals of decoding with optimization techniques are reviewed. Two algorithms are primarily introduced and implemented for LDPC codes and the AWGN channel, which are the IP decoding [Wad07, Wad08, Wad10] and the proximal decoding [WT21, WT22]. Both algorithms can be realized with low complexity, which is achieved by practicing approximations. The specialty of these two algorithms lies in their application of convex optimization methods. Simulation results have shown almost the same performance and have verified the conclusions in previous works. Ideas for improvement have been proposed, more investigations and correction are nevertheless required. To implement more exact convex optimization methods with less approximations or to employ simplification, is a trade-off between complexity and error-correcting performance. After exploring low-complexity algorithms, methods with better performance can be the direction of future work.

Concerning the application of optimization methods, LP decoding [FWK05] as well as LP-based algorithms are mostly studied. However, there are fewer algorithms proposed with focus on nonlinear domain. This thesis contributes to the validation and verification of previous studies in the field of channel decoding with convex optimization techniques. This allows us to confirm the possibility of applying methods besides LP solvers for binary linear block codes. Besides the methods covered in this thesis, there are many more convex optimization techniques to be explored.



## A. Abbreviations

<b>ADMM</b>	alternating direction method of multipliers
<b>ALP</b>	adaptive linear programming
<b>APP</b>	a posteriori probability
<b>AWGN</b>	additive white Gaussian noise
<b>BEC</b>	binary erasure channel
<b>BER</b>	bit error rate
<b>BP</b>	belief propagation
<b>BPSK</b>	binary phase-shift keying
<b>BSC</b>	binary symmetric channel
<b>CN</b>	check node
<b>FER</b>	frame error rate
<b>HDPC</b>	high-density parity-check
<b>HD</b>	hard decisions
<b>IP</b>	interior point
<b>LCLP</b>	linear code linear program
<b>LDPC</b>	low-density parity-check
<b>LLR</b>	log-likelihood ratio
<b>LP</b>	linear programming
<b>MAP</b>	maximum a posteriori
<b>MIMO</b>	multiple-input multiple-output
<b>ML</b>	maximum likelihood
<b>MMSE</b>	minimum mean square error
<b>MP</b>	message-passing
<b>MSA</b>	min-sum algorithm
<b>RPC</b>	redundant parity checks
<b>SD</b>	soft decisions
<b>SNR</b>	signal-to-noise ratio
<b>SOS</b>	sum-of-squares
<b>SPA</b>	sum-product algorithm
<b>VN</b>	variable node
<b>WER</b>	word error rate



# Bibliography

- [ADS12] S. Arora, C. Daskalakis, and D. Steurer, “Message-passing algorithms and improved lp decoding,” *IEEE Transactions on Information Theory*, vol. 58, no. 12, pp. 7260–7271, 2012.
- [Ana01] A. Anastasopoulos, “A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution,” in *GLOBECOM’01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 2, 2001, pp. 1021–1025 vol.2.
- [BBLK98] M. Breitbart, M. Bossert, R. Lucas, and C. Kempter, “Letter soft-decision decoding of linear block codes as optimization problem,” *European transactions on telecommunications*, vol. 9, no. 3, pp. 289–293, 1998.
- [BBV04] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes. 1,” in *Proceedings of ICC ’93 - IEEE International Conference on Communications*, vol. 2, 1993, pp. 1064–1070 vol.2.
- [Big05] E. Biglieri, *Coding for wireless channels*. Springer Science & Business Media, 2005.
- [BLDR13] S. Barman, X. Liu, S. C. Draper, and B. Recht, “Decomposition methods for large scale lp decoding,” *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 7870–7886, 2013.
- [Bos13] M. Bossert, “Kanalcodierung,” in *Kanalcodierung*. Oldenbourg Wissenschaftsverlag, 2013.
- [BPC<sup>+</sup>11] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [BSB14] A. Balatsoukas-Stimming and A. Burg, “Density evolution for min-sum decoding of ldpc codes under unreliable message storage,” *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, 2014.
- [Bur08] D. Burshtein, “Iterative approximate linear programming decoding of ldpc codes with linear complexity,” in *2008 IEEE International Symposium on Information Theory*, 2008, pp. 1498–1502.
- [Bur09] —, “Iterative approximate linear programming decoding of ldpc codes with linear complexity,” *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 4835–4859, 2009.
- [CDE<sup>+</sup>05] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, “Reduced-complexity decoding of ldpc codes,” *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, 2005.

- [CF02a] J. Chen and M. Fossorier, “Near optimum universal belief propagation based decoding of low-density parity check codes,” *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 406–414, 2002.
- [CF02b] J. Chen and P. Fossorier, “Density evolution for bp-based decoding algorithms of ldpc codes and their quantized versions,” in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 2, 2002, pp. 1378–1382 vol.2.
- [CF07] D. J. Costello and G. D. Forney, “Channel coding: The road to channel capacity,” *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.
- [CFRU01] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 db of the shannon limit,” *IEEE Communications letters*, vol. 5, no. 2, pp. 58–60, 2001.
- [CRU01] S.-Y. Chung, T. Richardson, and R. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, 2001.
- [CS07] M. Chertkov and M. Stepanov, “Pseudo-codeword landscape,” in *2007 IEEE International Symposium on Information Theory*, 2007, pp. 1546–1550.
- [DDKW08] C. Daskalakis, A. G. Dimakis, R. M. Karp, and M. J. Wainwright, “Probabilistic analysis of linear programming decoding,” *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3565–3578, 2008.
- [DFB14] D. Declercq, M. Fossorier, and E. Biglieri, *Channel Coding: Theory, Algorithms, and Applications: Academic Press Library in Mobile and Wireless Communications*. Academic Press, 2014.
- [DGK<sup>+</sup>16] I. Debbabi, B. L. Gal, N. Khouja, F. Tlili, and C. Jego, “Fast converging admm-penalized algorithm for ldpc decoding,” *IEEE Communications Letters*, vol. 20, no. 4, pp. 648–651, 2016.
- [DGW09] A. G. Dimakis, A. A. Gohari, and M. J. Wainwright, “Guessing facets: Polytope structure and improved lp decoder,” *IEEE Transactions on Information Theory*, vol. 55, no. 8, pp. 3479–3487, 2009.
- [DM98] M. Davey and D. MacKay, “Low density parity check codes over  $gf(q)$ ,” in *1998 Information Theory Workshop (Cat. No.98EX131)*, 1998, pp. 70–71.
- [DYW07] S. C. Draper, J. S. Yedidia, and Y. Wang, “Ml decoding via mixed-integer adaptive linear programming,” in *2007 IEEE International Symposium on Information Theory*, 2007, pp. 1656–1660.
- [EH03] G. Even and N. Halabi, “Improved bounds on the word error probability of ra (2) codes with linear programming based decoding,” in *Proc. 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.
- [Fel03] J. Feldman, “Decoding error-correcting codes via linear programming,” Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
- [FK02] J. Feldman and D. Karger, “Decoding turbo-like codes via linear programming,” in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 2002, pp. 251–260.



- [FKW02] J. Feldman, D. R. Karger, and M. Wainwright, “Linear programming-based decoding of turbo-like codes and its relation to iterative approaches,” in *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, vol. 40, no. 1. The University; 1998, 2002, pp. 467–477.
- [FKW03] J. Feldman, D. R. Karger, and M. J. Wainwright, “Lp decoding.” in *proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 2. The University; 1998, 2003, pp. 951–960.
- [FM97] B. J. Frey and D. MacKay, “A revolution: Belief propagation in graphs with cycles,” *Advances in neural information processing systems*, vol. 10, 1997.
- [FMI99] M. Fossorier, M. Mihaljevic, and H. Imai, “Reduced complexity iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673–680, 1999.
- [FMS<sup>+</sup>07] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, “Lp decoding corrects a constant fraction of errors,” *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 82–89, 2007.
- [FSBG09] M. F. Flanagan, V. Skachek, E. Byrne, and M. Greferath, “Linear-programming decoding of nonbinary linear codes,” *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4134–4154, 2009.
- [FU98] G. Forney and G. Ungerboeck, “Modulation and coding for linear gaussian channels,” *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2384–2415, 1998.
- [FWK03] J. Feldman, M. Wainwright, and D. R. Karger, “Using linear programming to decode linear codes,” in *37th annual Conference on Information Sciences and Systems (CISS’03)*, 2003.
- [FWK05] J. Feldman, M. Wainwright, and D. Karger, “Using linear programming to decode binary linear codes,” *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 954–972, 2005.
- [Gal62] R. Gallager, “Low-density parity-check codes,” *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.
- [GB10] D. Goldin and D. Burshtein, “Approximate iterative lp decoding of ldpc codes over  $gf(q)$  in linear complexity,” in *2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, 2010, pp. 000 960–000 964.
- [GB13] ———, “Iterative linear programming decoding of nonbinary ldpc codes with linear complexity,” *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 282–300, 2013.
- [HDAES12] A. E. S. Hassan, M. Dessouky, A. Abou Elazm, and M. Shokair, “Evaluation of complexity versus performance for turbo code and ldpc under different code rates,” *Proc. SPACOMM*, pp. 93–98, 2012.
- [HE05] N. Halabi and G. Even, “Improved bounds on the word error probability of  $ra(2)$  codes with linear-programming-based decoding,” *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 265–280, 2005.

- [HEAD01] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, “Efficient implementations of the sum-product algorithm for decoding ldpc codes,” in *GLOBE-COM’01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 2, 2001, pp. 1036–1036E vol.2.
- [HHW10] W. Han, J. Huang, and F. Wu, “A modified min-sum algorithm for low-density parity-check codes,” in *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*, 2010, pp. 449–451.
- [HOP96] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, 1996.
- [HSG<sup>+</sup>19] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, “Database of Channel Codes and ML Simulation Results,” [www.uni-kl.de/channel-codes](http://www.uni-kl.de/channel-codes), 2019.
- [IEE21] “Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 1: Enhancements for high-efficiency wlan,” *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–767, 2021.
- [ISFR11] M. R. Islam, D. S. Shafiullah, M. M. A. Faisal, and I. Rahman, “Optimized min-sum decoding algorithm for low density parity check codes,” *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 12, 2011.
- [Jer75] R. G. Jeroslow, “On defining sets of vertices of the hypercube by linear inequalities,” *Discrete Mathematics*, vol. 11, no. 2, pp. 119–124, 1975.
- [JLMH21] X. Jiao, H. Liu, J. Mu, and Y.-C. He, “ $l_2$ -box admm decoding for ldpc codes over isi channels,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3966–3971, 2021.
- [JMG16] X. Jiao, J. Mu, and J. Guo, “A comparison study of ldpc decoding using accelerated admm and over-relaxed admm,” in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 191–195.
- [Joh06] S. J. Johnson, “Introducing low-density parity-check codes,” *University of Newcastle, Australia*, vol. 1, p. 2006, 2006.
- [JWMC15] X. Jiao, H. Wei, J. Mu, and C. Chen, “Improved admm penalized decoder for irregular low-density parity-check codes,” *IEEE Communications Letters*, vol. 19, no. 6, pp. 913–916, 2015.
- [JZXZ07] M. Jiang, C. Zhao, E. Xu, and L. Zhang, “Reliability-based iterative decoding of ldpc codes using likelihood accumulation,” *IEEE Communications Letters*, vol. 11, no. 8, pp. 677–679, 2007.
- [JZZX06] M. Jiang, C. Zhao, L. Zhang, and E. Xu, “Adaptive offset min-sum algorithm for low-density parity check codes,” *IEEE communications letters*, vol. 10, no. 6, pp. 483–485, 2006.
- [KFL01] F. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-

- product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.
- [KV03] R. Koetter and P. O. Vontobel, “Graph-covers and iterative decoding of finite length codes,” in *Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics*. Citeseer, 2003, pp. 1–5.
- [KV06] ———, “On the block error probability of lp decoding of ldpc codes,” *arXiv preprint cs/0602086*, 2006.
- [LASMS98] M. Luby, M. Amin Shokrollahi, M. Mizenmacher, and D. Spielman, “Improved low-density parity-check codes using irregular graphs and belief propagation,” in *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, 1998, pp. 117–.
- [LD16] X. Liu and S. C. Draper, “The admm penalized decoder for ldpc codes,” *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 2966–2984, 2016.
- [LMS<sup>+</sup>97] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical loss-resilient codes,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 150–159.
- [LMSS98] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, “Analysis of low density codes and improved designs using irregular graphs,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 249–258.
- [LMSS01] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, “Improved low-density parity-check codes using irregular graphs,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585–598, 2001.
- [Mac] D. J. MacKay, “Encyclopedia of sparse graph codes.” [Online]. Available: <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>
- [Mac99] D. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.
- [Mas63] J. L. Massey, *Threshold decoding*. MIT Press, 1963.
- [MN96] D. J. MacKay and R. M. Neal, “Near shannon limit performance of low density parity check codes,” *Electronics letters*, vol. 33, no. 6, pp. 457–458, 1996.
- [Moo20] T. K. Moon, *Error correction coding: mathematical methods and algorithms*. John Wiley & Sons, 2020.
- [MPK<sup>+</sup>17] S. Myung, S.-I. Park, K.-J. Kim, J.-Y. Lee, S. Kwon, and J. Kim, “Offset and normalized min-sum algorithms for atsc 3.0 ldpc decoder,” *IEEE Transactions on Broadcasting*, vol. 63, no. 4, pp. 734–739, 2017.
- [PB<sup>+</sup>14] N. Parikh, S. Boyd *et al.*, “Proximal algorithms,” *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.
- [Pea88] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.
- [PF10] M. Punekar and M. F. Flanagan, “Low complexity linear programming decoding of nonbinary linear codes,” in *2010 48th Annual Allerton Conference*

- on *Communication, Control, and Computing (Allerton)*, 2010, pp. 6–13.
- [PL00] L. Ping and W. Leung, “Decoding low density parity check codes with finite quantization bits,” *IEEE Communications Letters*, vol. 4, no. 2, pp. 62–64, 2000.
- [PSE<sup>+</sup>07] S. Papaharalabos, P. Sweeney, B. G. Evans, P. T. Mathiopoulos, G. Albertazzi, A. Vanelli-Coralli, and G. E. Corazza, “Modified sum-product algorithms for decoding low-density parity-check codes,” *IET communications*, vol. 1, no. 3, pp. 294–300, 2007.
- [PVF13] M. Punekar, P. O. Vontobel, and M. F. Flanagan, “Low-complexity lp decoding of nonbinary linear codes,” *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3073–3085, 2013.
- [RSU01] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.
- [RU01] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.
- [RU08] —, *Modern coding theory*. Cambridge university press, 2008.
- [RVH95] P. Robertson, E. Villebrun, and P. Hoeher, “A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain,” in *Proceedings IEEE International Conference on Communications ICC '95*, vol. 2, 1995, pp. 1009–1013 vol.2.
- [Sch98] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [SLG04] E. Sharon, S. Litsyn, and J. Goldberger, “An efficient message-passing schedule for ldpc decoding,” in *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, 2004, pp. 223–226.
- [Spi96] D. Spielman, “Linear-time encodable and decodable error-correcting codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1731, 1996.
- [SS96] M. Sipser and D. Spielman, “Expander codes,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.
- [Tan81] R. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.
- [TNS08] M. H. Taghavi N. and P. H. Siegel, “Adaptive methods for linear programming decoding,” *IEEE Transactions on Information Theory*, vol. 54, no. 12, pp. 5396–5410, 2008.
- [TRH<sup>+</sup>08] A. Tanatmis, S. Ruzika, H. W. Hamacher, M. Punekar, F. Kienle, and N. Wehn, “A separation algorithm for improved lp-decoding of linear block codes,” in *2008 5th International Symposium on Turbo Codes and Related Topics*, 2008, pp. 37–42.

- [TRH<sup>+</sup>09] —, “Valid inequalities for binary linear codes,” in *2009 IEEE International Symposium on Information Theory*, 2009, pp. 2216–2220.
- [TRH<sup>+</sup>10] —, “A separation algorithm for improved lp-decoding of linear block codes,” *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3277–3289, 2010.
- [TS06] M. H. Taghavi and P. H. Siegel, “Adaptive linear programming decoding,” in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1374–1378.
- [TSR17] B. Tahir, S. Schwarz, and M. Rupp, “Ber comparison between convolutional, turbo, ldpc, and polar codes,” in *2017 24th International Conference on Telecommunications (ICT)*, 2017, pp. 1–7.
- [TSS08] M. H. Taghavi, A. Shokrollahi, and P. H. Siegel, “Efficient implementation of linear programming decoding,” in *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008, pp. 402–409.
- [TSS11] —, “Efficient implementation of linear programming decoding,” *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 5960–5982, 2011.
- [VK06] P. O. Vontobel and R. Koetter, “Towards low-complexity linear-programming decoding,” in *4th International Symposium on Turbo Codes & Related Topics; 6th International ITG-Conference on Source and Channel Coding*. VDE, 2006, pp. 1–9.
- [VK07] —, “On low-complexity linear-programming decoding of ldpc codes,” *European transactions on telecommunications*, vol. 18, no. 5, pp. 509–517, 2007.
- [VL98] J. H. Van Lint, *Introduction to coding theory*. Springer Science & Business Media, 1998, vol. 86.
- [Von08] P. O. Vontobel, “Interior-point algorithms for linear-programming decoding,” in *2008 Information Theory and Applications Workshop*, 2008, pp. 433–437.
- [W<sup>+</sup>09] N. Wells *et al.*, “Dvb-t2 in relation to the dvb-x2 family of standards,” *Advanced Television Systems Committee (ATSC) Inc*, vol. 130, 2009.
- [Wad07] T. Wadayama, “Interior point decoding for linear vector channels,” 2007. [Online]. Available: <https://arxiv.org/abs/0705.3990>
- [Wad08] —, “Interior point decoding for linear vector channels,” in *Journal of Physics: Conference Series*, vol. 95, no. 1. IOP Publishing, 2008, p. 012009.
- [Wad09] —, “An lp decoding algorithm based on primal path-following interior point method,” in *2009 IEEE International Symposium on Information Theory*, 2009, pp. 389–393.
- [Wad10] —, “Interior point decoding for linear vector channels based on convex optimization,” *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 4905–4921, 2010.
- [WB21] H. Wei and A. H. Banihashemi, “Admm check node penalized decoders for ldpc codes,” *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3528–3540, 2021.

- [Wib96] N. Wiberg, “Codes and decoding on general graphs,” *Ph. D thesis, Department of electrical engineering, linköping university Sweden*, 1996.
- [WJM15] H. Wei, X. Jiao, and J. Mu, “Reduced-complexity linear programming decoding based on admm for ldpc codes,” *IEEE Communications Letters*, vol. 19, no. 6, pp. 909–912, 2015.
- [WJW05] M. Wainwright, T. Jaakkola, and A. Willsky, “Map estimation via agreement on trees: message-passing and linear programming,” *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.
- [WLK95] N. Wiberg, H.-A. Loeliger, and R. Kotter, “Codes and iterative decoding on general graphs,” in *Proceedings of 1995 IEEE International Symposium on Information Theory*, 1995, pp. 468–.
- [WT21] T. Wadayama and S. Takabe, “Proximal decoding for ldpc-coded massive mimo channels,” in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 232–237.
- [WT22] T. WADAYAMA and S. TAKABE, “Proximal decoding for ldpc codes,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2022.
- [Xpl] “IEEE Xplore,” [Accessed 13-September-2022]. [Online]. Available: <https://ieeexplore.ieee.org/Xplore/home.jsp>
- [Yan88] M. Yannakakis, “Expressing combinatorial optimization problems by linear programs,” in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988, pp. 223–228.
- [YFW06a] K. Yang, J. Feldman, and X. Wang, “Nonlinear programming approaches to decoding low-density parity-check codes,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1603–1613, 2006.
- [YFW06b] ———, “Nonlinear programming approaches to decoding low-density parity-check codes,” *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1603–1613, 2006.
- [YHB04] M. Yazdani, S. Hemati, and A. Banihashemi, “Improving belief propagation on graphs with cycles,” *IEEE Communications Letters*, vol. 8, no. 1, pp. 57–59, 2004.
- [YLB11] A. Yufit, A. Lifshitz, and Y. Be’ery, “Efficient linear programming decoding of hdpc codes,” *IEEE Transactions on Communications*, vol. 59, no. 3, pp. 758–766, 2011.
- [YWF08] K. Yang, X. Wang, and J. Feldman, “A new linear programming approach to decoding linear block codes,” *IEEE Transactions on Information Theory*, vol. 54, no. 3, pp. 1061–1072, 2008.
- [ZJJ11] S. Zengyou, Z. Jin, and D. Juan, “Research of ldpc decoding based on llr bp algorithm,” in *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, vol. 2, 2011, pp. 889–892.
- [ZM11] X. Zhang and P. Ma, “Modified multistage linear programming decoding of ldpc codes,” in *2011 IEEE 13th International Conference on Communication*

*Technology*, 2011, pp. 104–108.

- [ZS11] X. Zhang and P. H. Siegel, “Adaptive cut generation for improved linear programming decoding of binary linear codes,” in *2011 IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 1638–1642.
- [ZS12] —, “Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes,” *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6581–6594, 2012.
- [ZS13] —, “Efficient iterative lp decoding of ldpc codes with alternating direction method of multipliers,” in *2013 IEEE International Symposium on Information Theory*, 2013, pp. 1501–1505.
- [ZZB05] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, “On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (ldpc) codes,” *IEEE transactions on communications*, vol. 53, no. 4, pp. 549–554, 2005.