Prédiction et estimation de très faibles taux d'erreur pour des chaînes de communication codées.

Shahkar Kakakhail

N° d'ordre : 2009-XX-XXX

THÈSE

présentée

à l'Université de Cergy-Pontoise
École Nationale Supérieure de l'Électronique et de ses Applications

pour obtenir le grade de :

Docteur en Science de l'Université de Cergy-Pontoise
Spécialité : Sciences et Technologies de l'Information et de la Communication

Par

Shahkar Kakakhail

Équipe d'accueil :
Équipes Traitement des Images et du Signal (ETIS) - CNRS UMR 8051

Titre de la thèse :

Prédiction et estimation de très faibles taux d'erreurs pour les chaînes de communications codées

Soutenue le 0 0 2009 devant la commission d'examen composée de :

Pr.	A. B	Examinateur
Pr.	C. D	Examinateur
Pr.	E. F	Rapporteur
Pr.	G. H	Rapporteur
Mr.	Vincent Heinrich	Co-encadrant
Dr.	Sylvain Reynal	Co-encadrant
Pr.	David Declercq	Directeur de thèse

Author's Publications

■ INTERNATIONAL CONFERENCES

[C_1] S. Kakakhail, S. Reynal, D. Declercq, V. Y. Heinrich, "Fast simulation for the performance evaluation of LDPC codes using fast flat histogram method", in Proc. of IEEE Sarnoff Symposium'08, Princeton, NJ USA, Apr. 2008

[C_2] S. Kakakhail, S. Reynal, D. Declercq, V. Y. Heinrich, "Efficient performance evaluation of forward error correcting codes", in Proc. of IEEE ICCS'08, Guangzhou, China, Nov. 2008

[C_3] S. Kakakhail, S. Reynal, D. Declercq, V. Y. Heinrich, "An efficient pseudo-codeword search algorithm for belief propagation decoding of LDPC codes", in Proc. of IEEE ICUMT'09, St. Petersburg Russia, Oct. 2009

Dedication

To Ali Kakakhail, my brother
and
to Jamila and Shehryar Kakakhail, my angel parents

Acknowledgements

I carried out my thesis under the auspices of (a) ETIS (Equipes Traitement des Images et du Signal) at ENSEA (Ecole Nationale Supérieure de l'Electronique et de ses Applications) and (b) STMicroelectronics, Crolles. I can say without exaggeration that the three years spent in my PhD make the most memorable time of my life. During this span of time, I have observed a tremendous positive change in my personal and professional development. I am of the opinion that everyone that we come across in life, influences us one way or the other. The bunch of people that I came to know during my PhD were simply wonderful. While I remain the sole responsible for imprecisions and omissions, there are a lot of people to whom I am indebted.

My foremost thanks go to Sylvain Reynal and David Declercq, who were the first that I made contact with. They have always been cool, encouraging, kind and patient. They will always remain special for me for the special reason that they showed confidence in me when I needed it the most. I take great pride in having worked with these two gentlemen, who are the specialists of their respective fields. I have learnt a lot from them.

My utmost gratitude goes to Vincent Heinrich, my ST supervisor, who is now an excellent friend. His cool and calm personality had been a source of inspiration for me. I have learnt a lot from him. I am really grateful to Pascal Urard, my ST manager. His dynamic and energetic personality taught me a great deal about the corporate sector.

My thanks and appreciation go to Yann Costes for his dedicated and continuous

help regarding the use of linux clusters at the University of Cergy-Pontoise.

I feel greatly indebted to all my colleagues at both ETIS and STMicroelectronics who made my life colorful. I feel lucky to have known such gentle and nice people.

I am thankful to my brother Ali, a special child, for illuminating my life with his presence. He has taught me the virtue of forbearance and patience.

I would like to thank my mother and father for their unconditional and boundless love. Whatever I am, I owe to my angel parents.

Above all, I am thankful to God for His guidance and blessings.

Abstract

The time taken by standard Monte Carlo (MC) simulation to calculate the Frame Error Rate (FER) increases exponentially with the increase in Signal-to-Noise Ratio (SNR). Importance Sampling (IS) is one of the most successful techniques used to reduce the simulation time. In this thesis, we investigate an advanced version of IS, called Adaptive Importance Sampling (AIS) algorithm to efficiently evaluate the performance of Forward Error Correcting (FEC) codes at very low error rates.

First we present the inspirations and motivations behind this work by analyzing different approaches currently in use, putting an emphasis on methods inspired by Statistical Physics. Then, based on this qualitative analysis, we present an optimized method namely Fast Flat Histogram (FFH) method, for the performance evaluation of FEC codes which is generic in nature. FFH method employs Wang Landau algorithm and is based on Markov Chain Monte Carlo (MCMC). It operates in an AIS framework and gives a good simulation gain. Sufficient statistical accuracy is ensured through different parameters. Extention to other types of error correcting codes is straight forward.

We present the results for LDPC codes and turbo codes with different codelengths and rates showing that the FFH method is generic and is applicable for different families of FEC codes having any length, rate and structure. Moreover, we show that the FFH method is a powerful tool to tease out the pseudocodewords at high SNR region using Belief Propagation as the decoding algorithm for the LDPC codes.

Résumé

Dans cette thèse, nous abordons le sujet d'optimisation des méthodes utlisées pour l'évaluation de performance des codes correcteurs d'erreurs. La durée d'une simulation Monte Carlo pour estimer le taux d'erreurs dans un système de communication augmente exponentiellement avec l'accroissement du Rapport Signal sur Bruit (RSB). Importance Sampling (IS) est une des techniques qui permettent à réduire le temps de ces simulations. Dans ce travail, on a étudié et mis en oeuvre une version avancée d'IS, appelé Adaptive Importance Sampling (AIS), pour l'évaluation efficace des codes correcteurs d'erreurs aux taux d'erreur très bas.

D'abord, nous présentons les inspirations et motivations en analysant différentes approches actuellement mises en pratique. On s'intéresse plus particulièrement aux méthodes inspirées de la physique statistique. Ensuite, basé sur notre analyse qualitative, nous présentons une méthode optimisée, appelé la méthode de Fast Flat Histogram (FFH) qui est intrinsèquement très générique. La méthode emploie l'algorithme de Wang-Landau, l'algorithme de Metropolis-Hastings et les chaines de Markov. Elle fonctionne dans le cadre de l'AIS et nous donne un gain de simulation satisfaisant. Différents paramètres sont utilisés pour assurer une précision statistique suffisante. L'extension vers d'autres types de codes correcteurs d'erreurs est directe.

Nous présentons les résultats pour les codes LDPC et turbocodes ayant différentes tailles et différents rendements. Par conséquent, nous montrons que la méth-

ode FFH est générique et valable pour une large gamme des rendements, tailles et structures. De plus, nous montrons que la méthode FFH est un outil puissant pour trouver des pseudocodewords dans la région de RSB élévé en appliquant l'algorithme de décodage Belief Propagation aux codes LDPC.

Contents

List of Figures

List of Tables

Abbreviations

FEC	Forward Error Correcting
FER	Frame Error Rate
BER	Bit Error Rate
LDPC	Low Density Parity Check (code)
SNR	Signal-to-Noise Ratio
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MMC	Multicanonical Monte Carlo
IS	Importance Sampling
MS/MT	Mean Shifting / Mean Translation
AIS	Adaptive Importance Sampling
DAIS	Dual Adaptive Importance Sampling
FFH	Fast Flat Histogram
BEC	Binary Erasure Channel
BIAWGN	Binary Input Additive White Gaussian Noise (channel)
BSC	Binary Symmetric Channel
TS	Trapping Sets
BP	Belief Propagation
LP	Linear Programming (decoding)
BPSK	Binary Phase Shift Keying
i.i.d.	independant and identically distributed
ML	Maximum Likelihood
MPA	Message Passing Algorithm
SPA	Sum Product Algorithm
MSA	Min-Sum Algorithm
PDF	Probability Density Function
WL	Wang Landau
GAR	Global Acceptance Ratio

1

Introduction - Context and Background

1.1 Context and background

In his seminal paper [1], Claude E. Shannon laid the mathematical foundations of the modern information theory. He introduced the concept of redundant channel coding as a method to achieve reliable communication on a noisy channel with known capacity. Shannon's mathematical proofs were related to random coding which are impractical owing to their complexity. Tremendous amount of work since then, has been devoted to designing good practical codes aiming at achieving the Shannon limit.

In 1993, Turbo codes [2] were discovered which proved to be a major breakthrough for a reliable communication through channel coding. This discovery showed the potential of iterative decoding as a means of approaching the channel capacity. Another powerful class of capacity approaching codes, called Low Density Parity Check (LDPC) codes were originally presented by Gallager in his PhD thesis in 1963 [3], but received little attention at that time. In addition, the large computational demand required in decoding long LDPC codes prevented their widespread use until major advances were made in computing, which eventually allowed a cost-effective decoding implementation. With the advent of turbo codes, LDPC codes got rediscovered [4] and regained due attention.

For most practical block lengths it is generally agreed that both turbo and LDPC

codes can offer similar performance which, for many applications, is far greater than all previously-known codes. For very large block lengths, in contrast to turbo codes, certain types of LDPC codes can reach the capacity limit on the bi-AWGN channel [5]. One disadvantage of LDPC codes compared to turbo codes is their quadratic-time encoding complexity in the general case, although there are certain types of LDPC codes which can partially avoid this drawback [6],[7].

There are two primary methods to gauge the performance of error correcting codes on a particular channel. The first method comprises of developing both lower and upper bounds on the probability of codeword error versus Signal-to-Noise Ratio (SNR). These bounds can be very informative in certain regions of SNR but may lead to a very loose estimate in other regions. The second commonly used method to find a code's performance is Monte Carlo (MC) simulation. This is generally a simpler and more accurate way to predict the performance of a code, especially in the lower SNR region. The problem with utilizing MC is the large computational time necessary to obtain an error estimate in the higher SNR region.

With the advancement in the code design and better decoders, it has become very important to gauge the performance of the communication system at very low error rates (higher SNR region). On the one hand, there are novel applications operating at very low error probability and standard Monte Carlo (MC) simulation takes extremely long to gauge their performance. On the other hand, new codes and decoders compete for a better performance in the error floor region where the performance evaluation is curbed and it is quite difficult to study the system properties. The problem stems from the fact that there is no analytical characterization of the error rate performance of codes on graphs (LDPC codes and turbo codes) employing iterative decoding, with the exception of a few channels and decoding algorithms for which such a characteri-

zation reduces to a more or less tractable combinatorial problem [8, 9, 10].

In this work, we mainly focus on adressing this problem of performance evaluation in the high SNR region. After extensive study of the existing methods and keeping in view the strengths and weaknesses therein, a powerful method existing in statistical physics was studied. This method, referred to as Fast Flat Histogram (FFH) method [11] in statistical physics, was brought into the the domain of channel coding. In statistical physics, this method proves to be very efficient for the evaluation of density of states for spin models having any number of interaction per spin. The most conspicuous feature of the method is its genericity. The principle is the same as that of the standard Monte Carlo method. However, the MC steps are controlled and depend on the history of the previous steps, a feature which makes it a Markov Chain Monte Carlo (MCMC) method.

1.1.1 Thesis organization

The rest of the dissertation is organized as follows: Chapter 2 gives an introduction to linear block codes with main focus on LDPC codes and the Message Passing Algorithm (MPA). The notions of Near Codewords, Trapping Sets, Stopping Sets and Absorbing Sets are presented which is followed by a discussion on pseudo-codewords. The chapter aims at introducing the basics briefly but comprehensively. Chpater 3 details the existing methods giving a brief introduction to each one followed by an overview. The minute details of the methods are not given. Chapter 4 covers the context, background, thoeretical details and implementation issues of Fast Flat Histogram (FFH) method when applied for the first time in the domain of coding theory. Chapter 5 details an application of FFH method to find the pseudo-codeword spectra employing Belief Propagation (BP) as the decoding algorithm. Chapter 6 concludes this work giving the perspective for future work.

2

LDPC Codes, Error Floor Region, Trapping Sets and Pseudo-codewords

SINCE the main emphasis of this work will remain on LDPC codes (through results for Turbo codes have also been presented), it is important to give a brief overview of this class of binary linear block codes.

2.1 Binary Linear Block Codes

Consider a system with a source vector **u** of k bits which are equally likely to be a '0' or '1' and to be transmitted over a noisy channel. Channel coding is a technique used to add redundancy to data so that imperfections in a communication channel will be less likely to destroy the transmitted information. There are two primary types of coding [12] - convolutional coding and block coding. Convolutional codes have been widely used in practice because they can be represented in a simple trellis structure which leads to an efficient decoding strategy known as the Viterbi [5] algorithm. Although convolutional codes have been tremendously useful for past applications, they fall short of the Shannon bound [1], a feat that coding researchers have always strived to achieve. Block codes also have many practical applications, especially when certain restrictions are placed on their construction, i.e. cyclic error detection codes such as

the famous Cyclic Redundancy Check (CRC) codes. This work deals with linear block codes, a subset of the class of all block codes.

A binary vector **x** is said to be in the code *C* described by its associated *generator matrix* **G**, if **x = uG** for some k length bit vector, **u**. The generator matrix is a $k \times n$ matrix of '1's and '0's that will describe a code with 2^k codewords if all rows of **G** are linearly independent. The code *C* can be seen as a k dimensional vector subspace of the set of all *n-tuples* that is spanned by the k linearly independent rows of **G**. A linear code has the property that $(\mathbf{u_i} + \mathbf{u_j})\mathbf{G} = \mathbf{u_i}\mathbf{G} + \mathbf{u_j}\mathbf{G}$.

$$\mathbf{H} = \begin{bmatrix} 1&1&1&1&0&0&0&0&0&0 \\ 1&0&0&0&1&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&1&0 \\ 0&0&1&0&1&0&0&1&0&1 \\ 0&0&0&1&0&0&1&0&1&1 \end{bmatrix}$$

Figure 2.1: The parity check matrix **H**

A *parity check matrix* **H** is an $(n-k) \times n$ matrix of '1's and '0's which also completely describes the codebook of a linear block code: $\mathbf{x} \in C$ if $\mathbf{xH^T} = \mathbf{0}$, i.e. **x** is within the null space of **H**. The term $\mathbf{xH^T} = \mathbf{0}$ is referred to as the 'syndrome' of the code. Syndrome checking consists of verifying whether the syndrome for a vector is null (a condition which ensures that the vector is included in the codebook). The code rate (a dimensionless quantity) is defined as $R = k/n$. Since both **G** and **H** can completely describe a code, there must be a way to convert from one form to the other. Consider the case, which is common in the design of LDPC codes, where we start with a parity check matrix and need to find its associated generator matrix in order to encode the data. In the following derivation of **G**, if $\mathbf{C_2}$ is a square, invertible binary

matrix, it is assumed that the $n-k$ columns of $\mathbf{C_2}$ are linearly independent. If they are not, we can permute columns in $\mathbf{H}$ until this condition is met. Beginning with the parity-check matrix $\mathbf{H}$ we write

$$\mathbf{H} = [\mathbf{C_1} : \mathbf{C_2}]$$

We form the *systematic* version of $\mathbf{H}$ in the next step, which allows for the systematic $\mathbf{G}$ to be constructed from the submatrices of $\mathbf{H}$.

$$\mathbf{H} = [\mathbf{C_2^{-1}C_1} : \mathbf{I}]$$

$$\Rightarrow \mathbf{G} = [\mathbf{I} : (\mathbf{C_2^{-1}C_1})^{\mathbf{T}}] \tag{2.1}$$

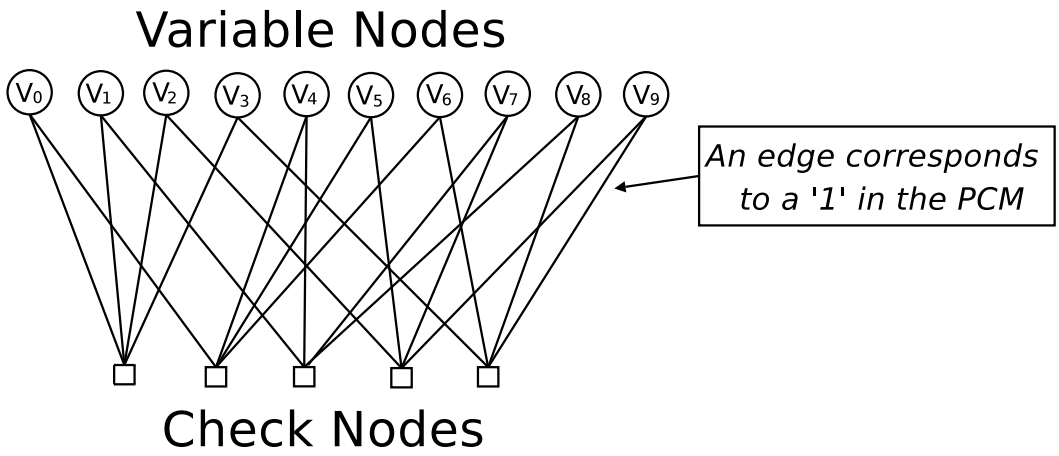

Figure 2.2: The Tanner graph representation of variable and check nodes

Linear codes have a 1-1 mapping from $\mathbf{H}$ to a graph form, called a Tanner graph [13]. Each '1' in $\mathbf{H}$ corresponds to an edge in the graph. Each column of the $\mathbf{H}$ matrix is represented by a *variable node* (v_i) in the graph. The i^{th} row has a check node (c_i) counterpart, which is connected to the variable nodes corresponding to the columns of $\mathbf{H}$ with a '1' in the i^{th} row. Tanner graphs are characterized by a 'bipartite' structure.

The two parties are check nodes and variable nodes, and nodes from one party are never directly connected to other nodes of the same party. A subgraph of a Tanner graph can be constructed from any subset of the n variable nodes and $n - k$ check nodes. The nodes in this subset will be called *active nodes* and the edges connecting the active nodes are known as *active edges*. A tree can be constructed from any subgraph within the Tanner graph. The root of a tree is defined as the variable node at the top of the tree from which all edges descend.

2.1.1 LDPC Codes

Low-density parity-check (LDPC) codes, a subset of linear block codes, carry this name because their parity check matrices, **H**, are characterized by a very small number of '1's compared to '0's. We will see that it is this low-density property of the code that allows a practical decoding algorithm and hence justifies the utility of this special class of codes.

R.G. Gallager first proposed LDPC codes and a few decoding algorithms in his doctoral dissertation in the early 1960's [3]. The large computational demand required in decoding long LDPC codes prevented their widespread use until major advances were made in computing, which eventually allowed a cost-effective decoding implementation. After turbo codes [2] were discovered in the early nineties, D.J.C. MacKay re-discovered and popularized LDPC codes in the late nineties [4]. Turbo codes and LDPC codes are special, not only because they can approach very close to the virtually error-free transmission limit, but mainly because a computationally efficient, so-called iterative, decoding scheme is readily available. When operating at moderate noise values, these decoding algorithms show an unprecedented ability to correct errors, a remarkable feature that has attracted a lot of theoretical attention [14],[15], [16], [17], [18], [19]. (Notice also statistical physics-inspired approach [20] that offered an im-

portant insight into the extraordinary performance of the iterative decoding).

For most practical block lengths, it is generally agreed that both turbo and LDPC codes can offer similar performance, which for many applications, is far greater than all previously-known codes. For very large block lengths, in contrast to turbo codes, certain types of LDPC codes can reach the capacity limit on the bi-AWGN channel [5]. One disadvantage of LDPC codes compared to turbo codes is their quadratic-time encoding complexity in the general case, although there are certain types of LDPC codes which can partially avoid this drawback [6],[7]. There are some new standards which are incorporating LDPC codes such as the new digital video broadcast (DVB-S2) standard [21], which concatenates an LDPC code with a BCH code. The 10-gigabit Ethernet standard will also make use of LDPC codes.

An ensemble of codes is defined as a family of codes that satisfy certain specifications. Typically, for LDPC codes, the ensembles are defined with respect to a certain degree profile. The degree profile of an LDPC code specifies the fraction of edges that are connected to variable and check nodes of a certain degree. The degree distribution polynomial $\lambda(x) = \sum_{d=1}^{d_v} \lambda(d) x^{d-1}$ says that fraction $\lambda(d)$ of the edges are connected to degree d variable nodes and d_v is the maximum degree variable node. A similar polynomial is constructed for the check nodes: $\rho(x) = \sum_{d=1}^{d_c} \rho(d) x^{d-1}$. These two polynomials are related because a code will have the same number of total edges,$|E|$, coming from both the check and variable nodes: $|E| \int_0^1 \lambda(x)dx = n$ and $|E| \int_0^1 \rho(x)dx = n - k$.

Often LDPC codes have a *regular* degree distribution, for example rate-1/2 regular 3,6 codes have all variable nodes with $d_v = 3$ and all check nodes with $d_c = 6$. $\{3, 6\}$ codes are very common in the literature and when decoding with the message

passing algorithm are shown to have the best waterfall threshold, i.e. the SNR region where the bit error rate begins to decrease very rapidly, over all rate-1/2, regular degree code ensembles [4]. Many examples in this work look at codes of this type with varying block length.

2.2 Decoding

The decoding algorithm are defined in two ways:

2.2.1 Hard decision decoding

Hard decision decoding is based on making a hard decision for all symbols exactly at the output of the channel. The decisions are given as inputs to the decoder and messages are passed between the nodes in the graph based on these decisions. The parity nodes checks whether the decisions verify its parity equation. As output towards the variable node, the parity nodes send the updated decisions towards the variable nodes which verify the parity equation. The variable nodes then takes a decision based on the input received at various links. A simple decision criteria can be to decide on the value for which the most number of messages in its favour and in case of a tie, we consider its initial value. This process is repeated iteratively until a code-word has been decoded or the allowed maximum number of iterations is reached.

2.2.2 Soft decision decoding

Soft decision decoding is same as the hard decision decoding, except for the messages travelling between the nodes are probability densities or log ratios of the probabilty densities in place of the actual values of the symbols. At the output of the channel,

based on the received data, a priori probability densities are calculated for all symbols which is given as input to the decoder.

The Message Passing Algorithm (MPA), also known as the Sum Product Algorithm (SPA) or belief propagation (BP) [22], [4], is the iterative decoding method generally used in LDPC codes on the AWGN channel. Traditionally, the goal of a decoder is to find the most likely codeword that was sent at the transmitter. Unfortunately, for long codes, no algorithm is known to exist that achieves this goal, except for the brute-force approach which makes 2^k correlations and comparisons for each decoding. The objective of the MPA decoder, on the other hand, is to maximize the a posterior probability (MAP) that a specific bit x_i was most likely a '0' or a '1', given the channel output vector, $\mathbf{y}$.

The exact a posteriori probability that code bit $x_i = 0$ is

$$\begin{aligned} P(x_i = 0|\mathbf{y}) &= \sum_{\mathbf{x} \in C : x_i = 0} P(\mathbf{x}|\mathbf{y}) \\ &= \sum_{\mathbf{x} \in C : x_i = 0} \frac{P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})}{P(\mathbf{y})} \\ &= \sum_{\mathbf{x} \in C : x_i = 0} K P(\mathbf{y}|\mathbf{x}) \end{aligned} \tag{2.2}$$

The constant K in Eq. (2.2) contains the $P(\mathbf{x})$ and $P(\mathbf{y})$ terms which are both independent of the index variable i, since we are assuming equally likely prior probabilities on the codewords $\mathbf{x}$. The problem with Eq. (2.2) is that the number of elements belonging to the set $\mathbf{x} \in C : x_i = 0$ is 2^{k-1}.

2.2.2.1 Message Passing Algorithm

For decoding purposes, there is no better way than to reconstruct the codeword that was most likely transmitted and then to compare the likelihoods of all possible codewords. However, this Maximum Likelihood (ML) algorithm becomes intractable already for codewords that are tens of bits long. The message passing algorithm, as its name suggests, operates by passing messages along edges in the Tanner graph. These messages represent some measure of probability that each bit is a '0' or '1'. The outgoing message at each node is a function of all incoming messages at the node, except for the message along the edge of the outgoing message. This property led researchers to label the messages as containing extrinsic ('from outside') information [23]. The low-density property of LDPC codes ensures a small number of messages (hence computations) are required at each node.

See Fig. 2.3 [24] for an illustration of both the variable and check node message flow for a $\{3, 6\}$ regular code.

There are three types of messages:

• Variable-to-Check Node Messages ($\uparrow q_{ij}(0)$)
The notation here for $q_{ij}(0)$ says that this message goes from the i^{th} variable node to the j^{th} check node and it passes the probability that this variable node is equal to '0'. This notation follows that used by [25]. When passing likelihood ratios, the '0/1' in parenthesis can be omitted because the ratio contains both pieces of information.
The messages are

$$q_{ij}(0) = P(x_i = 0|y_i, S_i, M_i(\sim j))$$

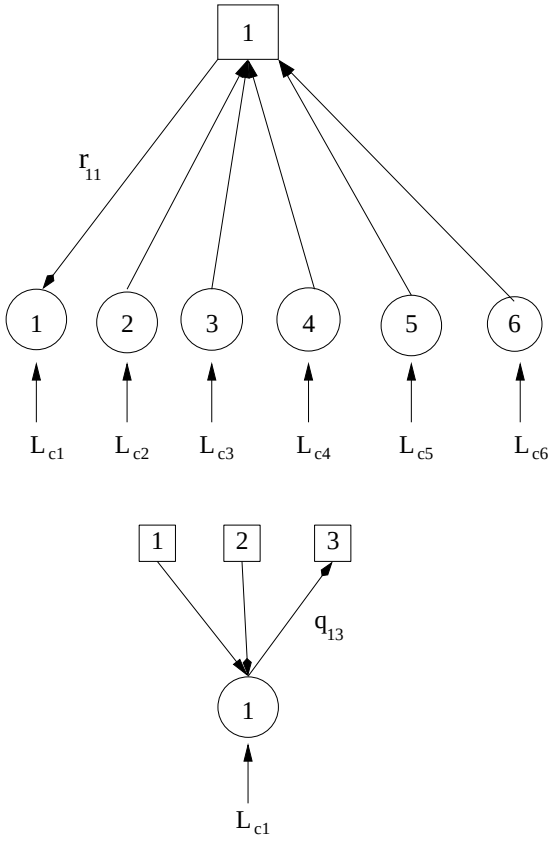

Figure 2.3: Message flow in MPA for a $\{3, 6\}$ code

$$= \frac{P(S_i|x_i = 0, y_i, M_i(\sim j))P(x_i = 0|y_i)}{P(S_i)}$$

$$= K_{ij}P(x_i = 0|y_i) \prod_{j' \in C_i \setminus j} r_{j'i}(0)$$

$$q_{ij}(1) = K_{ij}P(x_i = 1|y_i) \prod_{j' \in C_i \setminus j} r_{j'i}(1) \tag{2.3}$$

Here S_i is the event that all checks involving x_i are satisfied. $M_i(\sim j))$ means all messages from check nodes connected to variable node i, except for the message from the j^{th} check node. The K_{ij} are normalizing constants. C_i are the check nodes connected to the i^{th} variable node.

• Check-to-Variable Node Messages ($\downarrow r_{ji}(0)$)

Fact: A Property of Independent Binary Random Variable R.V.'s [3]:

Consider a sequence of M independent binary r.v.'s a_i such that $P(a_i = 1) = p_i$. The probability that the sequence has an *even* number of ones is given by:

$$\frac{1}{2} + \frac{1}{2}\prod_{i=1}^{M}(1 - 2p_i)$$

Satisfying a parity check in a block code is equivalent to counting binary r.v.'s and finding an even number of ones. Thus, the MPA makes use of this fact by producing the following check-to-variable message:

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i' \in V_{j/i}}(1 - 2q_{i'j})(1) \tag{2.4}$$

where $V_{j/i}$ is the set of all variable nodes connected to check j, except for the i^{th} one.

• Observation (channel evidence) Messages

For a memoryless channel, each received bit y_i is conditionally independent of all others. The transmitted bits, x_i , are also assumed to be equally likely. Thus,

$$P(x_i|y_i) = \frac{P(y_i|x_i)P(x_i)}{P(y_i)} \tag{2.5}$$

The Log Likelihood Ratio (LLR) of the channel data for the AWGN case is

$$Lc_i = \log\frac{P(x_i = 0|y_i)}{P(x_i = 1|y_i)} = 4\frac{E_s}{N_0}y_i \tag{2.6}$$

These three types of messages are shown in Fig. 2.3 [24].

Working in the probability domain (Eqs. (2.3), (2.4) and (2.5)) is more computationally burdensome than working in the log domain. Probabilities must be nor-

malized, an extra step, and independent events are multiplied as opposed to added in the log domain, a less expensive computation. There is also more numerical accuracy in the log domain, as very small probabilities that are multiplied become larger, more manageable exponents of probabilities that are added together (i.e., $(10^{-10})(10^{-10}) \rightarrow (-10)+(-10)$). Thus, the MPA is usually implemented in the log domain.

q_{ij}	$\longrightarrow$	$Lq_{ij} = \sum_{C_i \backslash j} Lr_{ji} + Lc_i$
r_{ji}	$\longrightarrow$	$Lr_{ji} = 2\tanh^{-1}\left[\prod_{V_j \backslash i} \tanh(0.5 Lq_{ij})\right]$
$P(x_i = 0)$	$\longrightarrow$	$LQ_i = \sum_{C_i} Lr_{ji} + Lc_i$

Table 2.1: MPA equations - Probability domain $\rightarrow$ Log domain

The MPA message equations in the log likelihood ratio domain are given in Table 2.1 [24].

A commonly used approximation to the full belief propagation message passing algorithm, called the Min-Sum Algorithm (MSA) [16] is carried out in the same way, except the message originating at the check nodes, Lr_{ji} in Table 2.1 is instead given by the much less complex:

$$Lr_{ji} = \prod_{V_j \backslash i} \text{sign}(Lq_{ij}) \min_{V_j \backslash i} |Lq_{ij}| \tag{2.7}$$

The message passing algorithm is an iterative procedure with operations described by the following pseudocode, using Table 2.1.

1. Initialize 'up' messages $Lq_{ij} = Lc_i \; \forall i, j \; s.t. \; H_{ij} = 1$

2. Update 'down' messages Lr_{ji}

3. Update 'up' messages Lq_{ij}

4. Marginalize: update LQ_i. Set

$$\hat{x} = \begin{cases} 0: & LQ_i > 0 \\ 1: & LQ_i < 0 \end{cases}$$

5. If $\hat{\mathbf{x}}\mathbf{H}^{\mathbf{T}} \neq \mathbf{0}$ perform another operation of message passing (go back to step 2).

If the maximum number of iterations has already been performed, then stop and $\hat{\mathbf{x}}$ is a non-codeword decoder failure. On the other hand, if $\hat{\mathbf{x}}\mathbf{H}^{\mathbf{T}}$, then a valid codeword has been detected.

There are variants on this algorithm which trade complexity for accuracy in computing the probability for each bit. One algorithm adds some postprocessing to the MPA to close the gap between BP and ML [26]. Other versions of the MPA, such as the min-sum algorithm simplify the computations at the check nodes, which is by far the most expensive operation in the algorithm. Some of these schemes trade roughly 0.5 dB of error performance in the threshold region for greatly reduced decoding complexity [27].

2.3 Error Analysis

To analytically determine how well an (n, k) linear block code performs on the AWGN channel, it is necessary to integrate a Gaussian density over all of the region ε, in an n-dimensional signal space that would not decode to the intended codeword. At high SNR, code performance when using an ML decoder is accurately described by a union

bound using the complete weight spectrum of codewords

$$P_f < \sum_{d=d_{\min}}^{d_{\max}} a_d Q(\sqrt{\frac{2dE_s}{N_0}}) < \sum_{d=d_{\min}}^{d_{\max}} \frac{a_d}{2} \exp(\frac{-dE_s}{N_0}) \tag{2.8}$$

where $d_{\min}$ is the Hamming distance of the nearest codewords and $d_{\max}$ is the Hamming distance of the farthest codewords in signal space. a_d is the corresponding multiplicity for codewords at Hamming distance d. In the large deviations theory literature, the nearest error regions contain points which are nearest in n-dimensional space to the correct signal point, and are called *minimum rate points* [28], [29].

The minimum rate points are those points in the error region that have the smallest Euclidean distance $d_{\varepsilon_{\min}}$ from the correct codeword. This Euclidean distance is just $\sqrt{d_{\min}}$ for ML decoding. It is documented in the literature [24] that when using the MPA decoder on LDPC codes, the nearest error regions are usually not valid codewords but are instead Trapping Sets (TS) [15]. The reason only the closest error events are dominant in the high-SNR region is because the argument of the exponential contains a multiplier of E_s/N_0. Thus, the contribution of the second-closest error events is decaying at a rate exponentially faster than the closest events.

$$\begin{aligned} P_f &< \sum_{d=d_{\min}}^{d_{\max}} a_d \exp(-d\frac{E_s}{N_0}) \\ &= a_{d_{\min}} \exp(-d_{\min}\frac{E_s}{N_0}) \Big[1 + \frac{a_{d_{\min}+1}}{a_{d_{\min}}} \exp(\frac{-E_s}{N_0}) + \ldots + \\ &\quad + \frac{a_{d_{\max}}}{a_{d_{\min}}} \exp[-(d_{\max} - d_{\min})\frac{E_s}{N_0}]\Big] \end{aligned} \tag{2.9}$$

All of the exponential terms in the brackets on the RHS of equation (2.9) will go to zero for sufficiently large E_s/N_0. So, at high SNR, only the error events associated with codewords at Hamming distance $d_{\min}$ are necessary in the union bound sum of

(2.9). This high-SNR behavior of the decoder leads to the following formal definition of a *dominant* error event.

Definition [24] *Let the reference position in n-dimensional space be the all-zeros (in GF(2)) codeword mapped to the all-ones vector in $\Re^n$. Every decoding rule will induce an error region surrounding the all-ones point. Consider an n-sphere centered at the all-ones point and inflating it from this point. At some radius, $d_{\varepsilon_{\min}}$, the n-sphere will first touch one or more points in the error region. There is a binary n-length vector which the decoder would output for channel outputs, $\mathbf{y}$, that land in these nearest error regions. These binary n-length vectors, which may or may not be valid codewords, will be considered dominant error events.*

Even though the task of describing code performance in the low and high SNR regions is the same, it is easier to think about the problem differently for each case. In the low SNR region, it is best to think of the mean of a random variable $(E[I_e(\mathbf{y})])$ as describing P_f, where $I_e(\mathbf{y})$ is the indicator function that evaluates to one if $\mathbf{y}$ is in the error region and zero otherwise. That is what a Monte Carlo simulation is calculating - the expected value of a random variable.

In the high-SNR region, on the other hand, it is easier to see the problem in a geometrical sense, as the above definition of a dominant error event demonstrates. In particular, locating the closest points of the error boundary and their shapes will give the information needed to calculate P_f. This is precisely what a union bound on ML decoding is doing - adding up the probability contribution from each error half-space. When employing ML decoding at high enough SNR, only the half-spaces between the codewords at the minimum Hamming distance contribute a significant percentage of the error probability. Thus, at high SNR, determining code performance is nearly

equivalent to the problem of finding all of the nearest error regions in n-dimensional space. Since an ML decoder is not used with large LDPC codes, the nearest error regions are typically not caused by valid codewords, but are instead a consequence of the suboptimal MPA decoder.

2.4 Trapping Sets and Stopping Sets

While measuring the performane of a forward error correcting code, one can typically divide the performance curve into three regions: low SNR region, waterfall region and error floor region [15], [14]. Error floor region corresponds to the performance measurement taken at a relatively high SNR ratio. It is a phenomenon characterized by an abrupt degradation of the coding scheme performance, as measured by the Bit Error Rate (BER) or Frame Error Rate (FER), from the waterfall regime of moderate signal-to-noise ratio (SNR) to the absolutely different error-floor asymptotic achieved at high SNR.

The transient behavior and the error floor asymptotic originate from the suboptimality of decoder, i.e., the ideal maximum-likelihood (ML) curve would not show such a dramatic change in the BER/FER with the SNR increase. While the slope of the BER/FER curve in the waterfall region is the same for almost all the codes in the ensemble, there can be a huge variation in the slopes for different codes in the error floor region [30]. The deterioration power of noise is quite low in this region and it becomes extremely cumbersome to get enough error events to gauge the performance with high confidence level.

The importance of error-floor analysis was recognized in the early stages of the turbo codes revolution [31], and it soon became apparent that LDPC codes are also not immune from the error-floor deficiency [15], [32]. Consequently, despite the appeal

of these codes for many high data rate communications and data storage applications, their wide-scale deployment has been hindered by incomplete understanding of finite-length effects and error floors.

Better understanding of the performance of finite-length LDPC codes in the low BER/FER regime has both theoretical as well as practical implications. From a theoretical standpoint, it provides a deeper understanding of the convergence of the message passing algorithms. For practical storage and wireline applications, such predictions provide a useful engineering tool in estimating performance and designing LDPC codes.

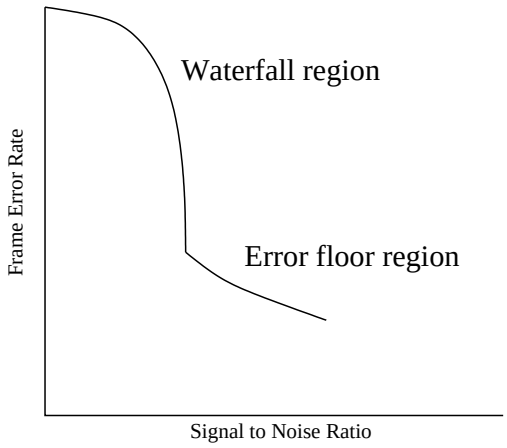

Figure 2.4: The performance curve of an error correcting code is divided into three regions: low SNR region, waterfall region and error floor region

The main approaches to the error-floor analysis problem proposed to date include: (i) a heuristic approach of the importance sampling type [15], utilizing theoretical considerations developed for a typical randomly constructed LDPC code performing over the very special binary-erasure channel [33], and (ii) deriving lower bounds

for BER [34].

To estimate the error-floor asymptotic in the modern high-quality systems is a notoriously difficult task. Typical required BER values are 10^{-12} for an optical communication system, 10^{-15} for hard drive systems in personal computers. However, direct numerical methods, e.g., Monte Carlo simulations, cannot be used to determine the BER below 10^{-9} though emulation of LDPC codes can be accelerated using Field-Programmable Gate Array (FPGA) platform [35], [36].

Both Turbo codes and LDPC codes employ an iterative decoding algorithm which is suboptimal in nature thus not computing the exact Maximum Likelihood (ML) decoding rule. For iterative decoding on the AWGN channel, MacKay and Postol [14] were the first to discover that certain *Near Codewords* are to be blamed for the high error floor in the Margulis code. Richardson reproduced their results [15] and developed a computation technique to predict the performance of a given LDPC code in the error floor domain.

Richardson also characterized the troublesome noise configurations leading to the error floor using combinatorial objects termed Trapping Sets (TS) and described a technique (of a Monte-Carlo importance sampling type) to evaluate the error rate associated with a particular class of trapping sets. Previously, these TS were termed as *Near Codewords* in [14]. A related concept of 'elementary trapping sets' was given in [37]. Milenkovic et al. [38] studied the asymptotic distribution of trapping sets in regular and irregular ensembles. Wang et al. [39] proposed an algorithm to exhaustively enumerate certain trapping sets.

It is extremely difficult to enumerate all of these error events; a brute force search

over an impractically large space is the only way to enumerate all of them. A good computational technique is presented by T. Richardson which is as follows [15]: Some large fixed number of decoding iterations (say 200) is performed unless the decoder converges to a codeword earlier. If it has not converged after the fixed number of iterations then we do some further iterations (say 20) and identify the trapping set as the union of all bits which do not decode correctly during those 20 iteration. Richardson pointed out that that the trapping set definition depends on the decoder input space and the decoding algorithm. In addition, if the channel is the BEC and the decoder is belief propagation then the trapping sets are precisely the stopping sets [33].

Motivated by empirical observations of the non-codeword outputs of LDPC decoders, the notion of stopping sets was first introduced by Forney, et al. [18] in 2001. A formal definition of stopping sets was given by Di Changyan, et al. [33]. They demonstrated that the bit and block error probabilities of iteratively decoded LDPC codes on the binary erasure channel (BEC) can be determined exactly from the stopping sets of the parity-check matrix. (Here, a stopping set S is a subset of the set of variable nodes such that all neighboring check nodes of S are connected to S at least twice).

The intuition behind stopping sets begins with an understanding of message-passing algorithms. Information given to a specific variable node from a neighboring check node is derived from all other variable nodes connected to that check node. If two variable nodes with erasures are connected to a common check node, then the check node is not able to determine the value of either of them. For this reason, the check nodes connected to a stopping set are incapable of resolving erasures if every variable node in the stopping set begins with an erasure.

An analysis of LDPC code performance on the BEC is purely combinatorial and

analytical results can be determined. The name follows from the inability of the message passing decoder (in the BEC case this is a bit-flipping decoder) to correct a block of data when erasures are found in each bit of a stopping set. However, an ML decoder could correct the message block if erasures were found in a non-codeword stopping set. The MPA decoding rule for the erasure channel is for each parity check node to replace an erasure bit with either a '1' or '0' to satisfy the parity if only one erasure occurs in the variable nodes connected to that parity check node. Thus, if two or more of the bits contain an erasure, then this parity check cannot help correct any erasures until another check node resolves some of these erasures. If there is a stopping set of bits such that every check node connected to those bits is connected at least twice, then the decoder stops making any progress once all erasures in the block have been corrected except for those in the stopping set bits.

Stopping sets on the erasure channel led other researchers to believe that an equivalent notion could extend to other channels including the AWGN channel model. The literature is undecided on the name of these bit vectors which cause the message passing decoder to fail. They have been referred to as 'near codewords' [14], 'pseudo-codewords' [40] and 'trapping sets' [15]. The near codeword and trapping set viewpoints both classify a bit vector with a pair (a, b), where a is the Hamming weight of the bit vector and b is the number of unsatisfied checks, i.e. the Hamming weight of the syndrome $\mathbf{xH^T}$. Alternatively, from a Tanner graph perspective, a TS could be defined as the a nonzero variable nodes of $\mathbf{x}$ and all of the check nodes connected by one edge to those a variable nodes. A valid codeword is a TS with $b = 0$. The term trapping set has caught on most widely in the literature.

Definition [24]: *During the decoding process, a history of the hard decision* $\hat{\mathbf{x}}_l$ *of the message estimate must be saved at each iteration* l *and if the maximum number of it-*

erations $I_{\max}$ occurs and no valid codeword has been found, the TS will be defined as the $\hat{\mathbf{x}}_\mathbf{l}$ which satisfies $\min_l w_H(\hat{\mathbf{x}}_\mathbf{l}\mathbf{H}^T)$ where $l = 1 \ldots I_{\max}$.

The MPA is an extremely useful suboptimal decoding algorithm, but it will only correctly compute the exact marginal probability for each code bit if no cycles exist in its Tanner graph [22],[41]. The reason why the MPA cannot perform the exact bit MAP operation for a general graph is because dependent information arises in the messages. This dependence occurs when the messages travel through cycles in the graph. A cycle of length c occurs when a closed path with c edges exists between a node and itself.

The girth of a graph is defined as the length of the shortest cycle in the graph. This implies if the girth of the graph is six, the first three message passing iterations would not contain any dependent information in the messages. At the fourth iteration, all nodes that are involved in length-six cycles would introduce some degree of dependence in the next group of messages emanating from those nodes. TS arise when this dependence occurs in a very severe form, where certain bits are involved in multiple shorter cycles. Since all long LDPC codes contain cycles, it is inevitable that TS of some type exist. The TS behave differently in the error floor region, where most of the errors will saturate to their final TS state early in iteration number and stay in this state regardless of the number of subsequent iterations. This type of behavior better reflects the notion of getting 'trapped' in an error state.

For hard-decision decoding algorithms [3], [42], [43], [44],[45], the following types of decoder failures corresponding to different types of trapping sets were reported in [46].

1) Fixed-pattern: After a finite number of iterations, the error positions at the

output of the decoder remain unchanged.

2) Oscillatory-pattern: After a finite number of iterations, the error positions at the output of the decoder oscillate periodically within a small set of variable nodes.

3) Random-like: Error positions change with iterations in a seemingly random fashion. The errors seem to propagate in the Tanner graph and result in a larger number of errors at the output of the decoder even if the initial error pattern has only a small weight.

The problem of locating trapping sets is a relatively recent problem and there are three searching methods worth mentioning. The first [15] is just to generate noise with the nominal density in the high SNR region and keep track of error events, most of which will be the lower weight trapping sets or codewords. This method is inefficient because it requires decoding huge numbers of messages that do not result in errors.

The second method of searching for dominant error events [47] only works for codes having a small number of very dominant trapping sets. The search method relies on the code having a small number of minimum length cycles. For codes with a more uniform cycle length distribution at each variable node, which is characteristic of most long codes, this method tends to miss many dominant trapping sets.

The third work comes from [24] where two search techniques are described - one combinatorial, employing graph theory arguments and the other using the power of the MPA itself to locate dominant error events. The combinatorial search has a limited scope; it is only practical for finding TS with $a < 10$ or so. The decoder search, however, leads to a much more general search technique, having applicability for most

LDPC codes.

2.5 Absorbing Sets

To characterize the error events, Zhang *et al.* introduced the notion of an absorbing set [48] which is used to describe an error event that can occur when the message passing decoding fails to converge to a codeword after a large number of iterations. An absorbing set is defined as [48]: Let T_H be the bipartite Tanner graph corresponding to the parity check matrix H of the given code. We say that the subset of a bit nodes and their neighboring b check nodes in T_H constitute an (a, b) set if in the subgraph induced by these a bit nodes, exactly $b > 0$ check nodes have odd degrees, each of these a bit nodes is connected to more even-degree checks than odd-degree checks, and all remaining check nodes outside of the induced subgraph have even degree with respect to T_H. We say that an (a, b) set is an (a, b) absorbing set if for all $a', a' < a$, it does not contain an (a', b) set as its subgraph.

Zhang *et al.* choose to use the absorbing set as defined above in order to explicitly distinguish the convergence of the decoder to a non-codeword from its oscillatory behavior. The name 'absorbing sets' was given due to their attractive nature. They also demonstrated that the occurrence of the absorbing set increases with the decrease in the codelength.

A theoretical analysis of the absorbing sets is given in [49] where it is argued that the absorbing sets are related to (but not entirely equivalent to) previously introduced combinatorial structures, including stopping sets, trapping sets, near codewords and pseudo-codewords. The notion of absorbing sets was introduced to qualitatively describe the convergent non-codeword state of the message passing algorithms, when

the transmission channel is Additive White Gaussian Noise (AWGN).

In the asymptotic limit given by the bit flipping algorithm, the configuration described as a fully absorbing set is stable, since each bit node receives strictly more messages from the neighboring checks that reinforce its value than messages that suggest the opposite bit value. In particular, a fully absorbing set can be viewed as a near codeword as defined in [14], though the reverse is not true, since a near codeword does not necessarily describe a stable configuration. The trapping set definition introduced in [15] also does not explicitly capture the convergent behavior since it refers to the union of all bits that are not eventually correct, and thus permits a situation in which the decoder oscillates among a finite number of states. Although stopping sets [33] also describe stable configurations, they are defined in the context of a binary erasure channel, and cannot be directly applied to an AWGN channel.

2.6 Pseudo-codewords

Decoding errors for iterative message-passing algorithms are also often attributed to pseudocodewords [50]. Work on relating pseudocodewords to stopping sets for the BEC [18], the binary symmetric channel (BSC) and the AWGN channel [51] has revealed a relationship between pseudocodeword weight and stopping set size. However, the current notions of stopping sets and pseudocodewords do not completely characterize the performance and non-codeword outputs of iterative decoders on the BSC and AWGN channels.

In his dissertation, Niclas Wiberg provides the foundation for analyzing these errors by turning to an analysis of computation trees [16]. Even with these insights, theoretical analyses of the convergence of iterative message-passing decoding have thus far been scarce. (A notable exception is the work done on density evolution [5], [42],

which considers ensembles of LDPC codes rather than individual codes.) Meanwhile, linear programming (LP) decoding [52] has strong heuristic ties to iterative message-passing decoding by way of graph cover decoding, and its analysis has proven much more tractable [53]. The common finding across all analyses of these decoders is that pseudocodewords play a significant role in determining convergence of the decoder and in understanding the non-codeword outputs that arise.

Three types of pseudocodewords for LDPC codes are found in the literature: graph cover pseudocodewords, linear programming pseudocodewords and computation tree pseudocodewords [54]. Kelley and Sridhara studied pseudo-codewords [55] arising from graph covers and derived bounds on the minimum pseudo-codeword weight in terms of the girth and the minimum left-degree of the underlying Tanner graph. The bounds were further investigated by Xia and Fu [56]. Smarandache and Vontobel [57] found pseudo-codeword distributions for the special cases of codes from Euclidean and projective planes. Pseudo-codeword analysis has also been extended to the convolutional LDPC codes by Smarandache et al. [58].

Linear Programming decoding, introduced in [52], [59] is a close relative of BP which can be viewed as a relaxed version of Maximum Likelihood (ML) decoding. For any realistic code (with loops), the BP algorithm is approximate attempting to solve iteratively nonlinear equations, called BP equations which describe extrema (e.g. minima are of main interest) of the Bethe free energy [60]. Relation of the LP decoding to the Bethe free energy approach [60] and thus to BP equations and decoding, was noticed in [59], and the point was elucidated further in [40], [61], [53], [62], [63], [64]. In short, LP may be considered as large SNR asymptotic limit of BP, where the later is interpreted as an extremum of the Bethe free energy functional. The failures of the LP decoder can be understood in terms of the vertices of the so-called fundamental

polytope which are also known as pseudo-codewords [59].

Vontobel and Koetter introduced a theoretical tool known as graph cover approach [53] and used it to establish connections between the LP and the message passing decoders using the notion of the fundamental polytope. They showed that the pseudo-codewords arising from the Tanner graph covers are identical to the pseudo-codewords of the LP decoder. Vontobel and Koetter also studied the relation between the LP and the min-sum decoders [61].

Both BP and LP are computationally efficient but suboptimal, i.e. incapable of matching performance of the Maximum-Likelihood (ML). Even though BP and LP decodings are suboptimal with respect to ML at all SNRs, the difference in FER is only order one in the water-fall regime of small SNRs. The situation becomes significantly worse in the error-floor domain of moderate to large SNRs where FER for BP/LP is parametrically, i.e. orders of magnitude, larger than FER for ML. Length of the error-correction code brings another dimension into the problem. The longer the code the lower is the value of FER where the waterfall-to-error-floor transition happens. On the other hand, standard Monte-Carlo (MC) numerics is incapable to determine BER below 10^{-9}. Therefore, understanding and describing the error-floor by an alternative, and hopefully more insightful method is in great demand [15].

One such useful insight came through recent efforts [65], [66], [67], [62], [64] to understanding error-floor in terms of the most probable of the dangerous configurations of the noise, so-called instantons, contributing most to FER. BP/LP decodes the instantons into the so-called non-codeword pseudo-codewords [17], [16], [18], [15], [40]. It was recognized that for moderate and large SNRs splitting of the two (FER vs SNR) curves, representing ML decoding and approximate BP/LP decoding, is due

to the pseudo-codewords, which are confused by the suboptimal algorithm for actual codewords of the code. Describing BP/LP error-flooor translates into finding pseudo-codewords with low effective distance.

It is well established that the distance between codewords significantly impacts the probability of decoding errors, and thus it is important to further explore the effect of the distance between pseudocodewords and codewords. For binary linear codes, the classical problem of finding distances between codewords is significantly simplified by looking instead at the weights of codewords, which is made possible by the algebraic structure of the code. To parallel the classical case, we consider the (effective Hamming) weight [18] of a pseudocodeword. It should be noted that this notion of weight was originally motivated by the definition of the generalized weight of a computation tree configuration, as given by Wiberg in [16].

Definition 2.2 (See Forney, et al. [18], Corollary 3.1). On the additive white Gaussian noise channel, the (effective Hamming) weight of a nonzero vector $\mathbf{x} = (x_1, \ldots, x_n)$ of nonnegative rational numbers is given by

$$w(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{(\sum_{i=1}^{n}) x_i^2}$$

Using the weight measure of Definition 2.2, Forney, et al. [18] show that the minimum weight of a vertex of the fundamental polytope [40] determines bounds on linear programming decoding performance. It is important to note that these results deal only with the overall probability of word error when decoding; they say nothing about the probability of word error caused by a given pseudocodeword.

2.7 Difference between Trapping Sets and Pseudocodewords

Although there is a tendency to use the terms trapping sets and pseudocodewords interchangably, yet there is a subtle difference between the two. Both try to say what went wrong in the iterative decoding but the specifics are different. Trapping sets are decoder dependent so different decoders have different trapping sets and researchers have put efforts (as referenced earlier) to classify the low-weight trapping sets for a variety of decoders. For a given iterative decoding algorithm, it is clear what the relevance of trapping sets are with respect to decoding failures. However, it is not yet clear, how to know which trapping sets are important and which ones are not. One can always try to list all (a, b) trapping sets and try out empirically which ones cause problems.

On the other hand, pseudocodewords are defined independently of the iterative decoding algorithm that is used. They are mainly characterized by the fundamental polytope. However, their immediate implication for the decoding behavior needs to be looked at from case-to-case. For the BEC channel characterized by stopping sets, to every pseudocodeword there corresponds one stopping set (given by the support of the pseudocodeword). Consequently, for every stopping set, there is at least one pseudocodeword whose support equals that stopping set.

3

Exiting Methods for the Performance Evaluation at Low Error Rates

IT is not feasible to get an exact analytical expression for the probability of error in a long code, because the error regions have an extremely complex n-dimensional shape. Instead, we generally resort to simulation methods. When analyzing the probability of bit or block error versus SNR for a long code, there are typically two regions of interest. The first region is the low SNR regime. When it is only necessary to calculate a block error performance down to roughly 10^{-5}, a Monte Carlo simulation will provide an efficient and accurate result.

The second region of interest lies in the higher SNR region, or error floor, where only a few rare, but dominant error events contain nearly all of the error probability mass. This is the region of the performance curve that is typically out of reach unless the code has special properties that allow a simple error calculation.

A number of methods exist for the efficient performance evaluation of FEC codes at low error rates. These methods can be broadly classified into two categories: methods that take into account the code structure and other characteristics and methods which are generic in nature being independant of the code structure.

3.1 Monte Carlo Methods

The Monte Carlo method [68] of simulation was first used as a way to solve multidimensional integrals. In its most general form, it is a tool to find the expectation of a function of a random variable:

$$E[g(x)] \simeq \frac{1}{L}\sum_{l=1}^{L} g(x_l) \tag{3.1}$$

where the x_l are independent random samples drawn from the distribution that describes the random variable x. The total number of random samples is given by L. Eq. 3.1 is justified by the law of large numbers, as sample averages of i.i.d. random variables converge in the mean-square sense to their mean as the number of samples (L) in the average increases. In the performance analysis of codes, the usual metric of interest is the probability of frame error P_f. This probability can be given as

$$P_f = \int_{\varepsilon} f(\mathbf{y})d\mathbf{y} = \int_{\Re^n} I_e(\mathbf{y})f(\mathbf{y})d\mathbf{y} = E[I_e(\mathbf{y})] \tag{3.2}$$

where I_e is the indicator function for the error region ε in n-dimensional signal space, and $f(\mathbf{y})$ has a Gaussian distribution for the AWGN channel. The Monte Carlo estimate becomes

$$\hat{P}_{f_{MC}} = \frac{1}{L}\sum_{l=1}^{L} I_e(\mathbf{y}_l) \tag{3.3}$$

An estimate is considered unbiased if the expected value of the estimate is equal to the true value being estimated.

$$E[\hat{P}_{f_{MC}}] = \frac{1}{L}\sum_{l=1}^{L} E[I_e(\mathbf{y}_l)] = \frac{1}{L}\sum_{l=1}^{L} P_f = P_f(\text{unbiased}) \tag{3.4}$$

3.2 Importance Sampling

In Importance Sampling (IS), we statistically bias the sample generation (noise realizations) in a manner that produces the disired result more frequently (error events). Instead of accumulating 1 for each error event, a 'weight' is accumulated for each error to restore an unbiased estimate of P_f. This strategy, if done correctly, will lead to a greatly reduced simulation time of the estimate compared to standard MC. For a comprehensive treatment of IS, see [69],[70].

Some examples of IS methods are given in the following.

3.2.1 Importance Sampling by biasing density function

This IS method is briefly presented as follows [24]:

A biasing density function $f^*(\mathbf{y})$ is introduced into the MC estimator. The desired error probability can be rewritten as

$$P_f \triangleq E[I_e(\mathbf{y})] = \int_{\Re^n} I_e(\mathbf{y}) f(\mathbf{y}) d\mathbf{y} = \int_{\Re^n} I_e(\mathbf{y}) \frac{f(\mathbf{y})}{f^*(\mathbf{y})} f^*(\mathbf{y}) d\mathbf{y} = E_*[I_e(\mathbf{y}) w(\mathbf{y})] \tag{3.5}$$

which gives an alternate estimator

$$\hat{P}_{f_{IS}} = \frac{1}{L} \sum_{l=1}^{L} I_e(\mathbf{y}_l) w(\mathbf{y}_l) \tag{3.6}$$

The $\mathbf{y}_l$ are now generated according to $f^*(\mathbf{y})$, the biased density. If $\mathbf{y}_l$ lands in the error region as determined by the decoder, then the weight function $w(\mathbf{y}_l) = \frac{f(\mathbf{y})}{f^*(\mathbf{y})}$ is accumulated to find the estimate of P_f. MC can be seen as a special case of this more general procedure, with $f^*(\mathbf{y}) = f(\mathbf{y})$. It can be shown that the IS estimator is also

unbiased

$$E[\hat{P}_{f_{IS}}] = \frac{1}{L}\sum_{l=1}^{L} E_*[I_e(\mathbf{y}_l)w(\mathbf{y}_l)] = E_*[I_e(\mathbf{y})w(\mathbf{y})] = \int_{-\infty}^{\infty} I_e(\mathbf{y})\frac{f(\mathbf{y})}{f^*(\mathbf{y})}f^*(\mathbf{y})d\mathbf{y}$$

$$= P_f(\text{unbiased}) \tag{3.7}$$

3.2.1.1 Mean Shifting/Translation IS technique

Mean-shifting (MS) or Mean-Translation (MT) is a popular IS technique [71],[28] where the IS density f^* has the same properties as the nominal density f, except for its mean which is shifted to lie on the boundary of the error region. It is the most efficient IS scheme. In general, MT performs efficiently when the error region geometry is simple. For this reason, MT is often implemented in a divide-and-conquer manner for multi-dimensional systems, that is, the error region is partitioned into simple subregions and the error probability is estimated for each subregion with MT.

A natural partitioning scheme for coded systems is partitioning-by-codeword. This technique has been used on a Hamming (7, 4) code and trellis codes with maximum likelihood decoding criterion [72]. An immediate drawback of this partitioning-by-codeword scheme is its requirement of codebook information. The codebook size becomes prohibitively large as the code length increases. On the other hand, if the codebook size is manageable, the ML performance can be analytically approximated via the union bound technique. Thus, it is somewhat unrealistic to attempt to obtain ML performance via IS simulations.

In [73], short block codes with message-passing decoding were considered with a partitioning scheme that is slightly different than that in [72]. The authors have shown how IS can be applied to evaluate the performance of optimal MAP bit-per-bit

decoders, and of non-optimal (turbo-like) iterative decoders. Though efficient for short codes, this IS scheme still requires the codebook information, which disqualifies it for long codes.

An IS scheme for linear block codes with message-passing decoding that assumes no code-book knowledge was proposed in [74] where a block length of 96 was used making use of the code structure to produce noise events more frequently (application to loop-free decoding trees was given in [75]). Another direct application of an IS technique for LDPC codes is given in [47]. For turbo codes, IS technique was studied in [76].

The choice of the IS density is, quite naturally, critical to the success of the simulation. For the IS to be most effective, the optimal density should neither be underbiased nor overbiased. The optimal density is well known, but it is a function of the probability of error and therefore cannot be used [77]. Conventional IS (CIS) uses a density that is obtained by simply increasing the variance of the underlying density [78]. The improvement in performance obtained with this technique is limited by the memory length of the system, which makes it impractical for most systems of interest.

Improved Importance Sampling (IIS) uses a mean translation of the underlying density which overcomes the effects of memory [71]. In addition, the use of the tail of several different pdf's has been explored in [79], [80],[81]. Two types of importance sampling methods for rare event sampling are presented in [82]. The first approach selects importance sampling distributions by minimizing the variance of importance sampling estimator. The secod approach selects importance sampling distributions by minimizing the cross entropy to the optimal importance sampling distribution.

Although many IS densities have been proposed, their performance varies from system to system. This problem is further complicated by the fact that determining the suitability of an IS density to a particular communications system is similar in complexity to finding the probability of error itself. These problems have prevented IS from gaining wide-scale acceptance despite its promise of decreasing simulation times by several orders of magnitude.

A graph searching technique that can efficiently find the dominant trapping sets and low-weight codewords is presented in [83], [24]. Multiple error impulses are applied on specific nodes in the graph to tease out the dominant TS. Biasing function is calculated based on these TS and IS is then employed by producing noise events in the dominant TS regions to get errors more frequently and FERs are calculated.

In [84], the authors present an importance sampling method for the evaluation of low FER performance of LDPC codes under iterative decoding. It relies on a combinatorial characterization of the absorbing sets [48]. The biased density in the importance sampling scheme is a mean-shifted version of the original Gaussian density which is suitably centered between a codeword and a dominant absorbing set. This choice of biased density yields an unbiased estimator for the FER with a variance lower by several orders of magnitude than the standard Monte Carlo estimator.

3.3 Error rate estimation using cycle enumeration

For Binary Sysmmetric Channels (BSC), an efficient error rate estimation was presented in [46] which was further modified in [85], [86]. A combinatorial approach is adopted and the method is mainly based on efficient enumeration of input vectors with small distances to a reference vector whose elements are selected to be the most reliable values from the input alphabet. Several techniques, including modified cycle

enumeration, are employed to reduce the complexity of the enumeration. The error rate estimate is derived by testing the input vectors of small distances and estimating the contribution of larger distance vectors.

The method proposed in [46] is valid for hard-decision iterative algorithms. Their particularity is that only binary messages are used rendering them quite simple. Some examples are the so-called Gallager algorithms A (GA) and B (GB) [3], [42], [43], their variants [44] and Majority-Based (MB) algorithms [45]. The method is based on enumerating the initial error patterns of smallest weight that cannot be all corrected by the decoder. By using this information, the contribution of all the other initial error patterns with larger weights to the total FER and BER is estimated.

Consider a given LDPC code with block length n decoded by a given hard-decision iterative algorithm over a BSC with crossover probability ε. Denote the set of all the error patterns of weight i by S_i, and those that cannot be corrected by the decoder by E_i. Clearly, $|S_i| = \binom{n}{i}$. Suppose that the decoder can correct all the error patterns of weight $J-1$ and smaller, i.e.,$|E_i| = 0, \forall i < J$. Also suppose that there are $|E_J| = 0$ error patterns that the decoder fails to correct. The FER is then equal to

$$\mathrm{FER} = \sum_{i=J}^{n} \frac{|E_i|}{|S_i|} p_i = \sum_{i=j}^{n} |E_i| \varepsilon^i (1-\varepsilon)^{(n-i)} \tag{3.8}$$

where i is the weight of the initial error pattern at the input of the decoder, and p_i is the probability of having i errors at the output of the channel (or the input of the decoder).

3.4 Instanton Analysis

In 1989, Nicolas Sourlas established a relationship between statistical mechanics especially spin model theory and error correction theory [20]. He presented codes which could, under certain circumstances, be the only known codes at that time to achieve Shannon's well-known code performance bounds. Sourlas also showed that the error-correcting codes are mathematically equivalent to some theoretical spin-glass models and it is possible to use their equivalence to analyse them using the methods of statistical mechanics [87],[88].

Some other statistical physics methods recently used in the coding theory context are [89], [90], [91], [92], [60], [93]. Instanton analysis constitutes a method, aiming to estimate a low probability event, and is known under the names of instanton calculus, saddle-point or optimal fluctuation and is common in theoretical physics. It was suggested first in the context of disordered systems [94] and reinvented in the quantum field theory context [95].

It was pointed out by Sourlas [20] that the error-correction problem can be conveniently reformulated in terms of equilibrium statistical physics. (See e.g. more recent discussions in the error-correction literature [60],[96],[97]). Specifically it was shown that formalism and approaches developed in the context of disordered systems, e.g. spin-glass, can be, though with essential modifications, applied to the coding theory [87],[88],[92].

Instanton analysis or instanton amoeba method, introduced in [65], [66] is named after a theoretical particle in quantum physics that lasts for only an instant, occupying a localized portion of space-time [98]. Statistical physics uses the word instanton to describe a microscopic configuration which, in spite of its rare occurrence, contributes

most to the macroscopic behavior of the system [94]. In a nutshell, an instanton is a configuration of the noise which is positioned in between a codeword (say zero codeword) and another pseudo-codeword (which is not necessarily a codeword). Incremental shift (allowed by the channel) from this configuration toward the zero codeword leads to correct decoding (into the zero-codeword) while incremental shift in an opposite direction leads to a failure. In principle, one can find this dangerous configuration of the noise by exploring the domain of correct decoding surrounding the zero codeword, and finding borders of this domain – the so-called error-surface. If the channel is continuous, the error-surface consists of continuous patches while configuration of the noise maximizing the error probability over a patch is called an instanton.

Instanton amoeba scheme is an efficient numerical scheme, which is *ab initio* by construction, i.e., the scheme requires no additional assumptions (e.g., no sampling). The numerical scheme is also accurate at producing configurations whose validity, as of actual optimal noise configurations, can be verified theoretically and that provide a tight lower bound for BER. The instanton scheme is also generic, in that there are no restrictions related to the channel or decoding. The method finds instanton/pseudo-codeword by means of a simplex (amoeba) optimization. The algorithm is initialized with a random simplex and many sequential attempts are required to built the instanton/pseudo-codeword frequency spectra of the code. The instanton-amoeba method is general but also computational resources consuming.

When SNR is large, FER as an integral over output configurations is approximated by

$$\mathrm{FER} \sim \sum_{\mathrm{inst}} V_{\mathrm{inst}} \times P(\boldsymbol{x}_{\mathrm{inst}} \mid \mathbf{1}) \tag{3.9}$$

where $\boldsymbol{x}_{\mathrm{inst}}$ are the special instanton configurations of the output maximizing $P(\boldsymbol{x}|\mathbf{1})$ under the $\chi_{\mathrm{error}} = 1$ condition, and V_{inst} combines combinatorial and phase-volume

factors [99]. Generically, there are many instanton configurations that are all local maxima of $P(\boldsymbol{x}|\mathbf{1})$ in the noise space. Individual contributions into FER decrease significantly with SNR increase. At large SNR, only instanton with the highest $P(\boldsymbol{x}|\mathbf{1})$ is relevant.

The numerical method to find the instanton is given in [99]. The AWGN channel is defined by,

$$P(\boldsymbol{x}|\boldsymbol{\sigma}') = \prod_i p(x_i|\sigma_i'), p(x|\sigma) \propto \exp\left(- \frac{(x-\sigma)^2 s^2}{2} \right) \tag{3.10}$$

If the detected signal at a bit is x, the respective log-likelihood at the bit is $h = \ln(p(x|1)/p(x|-1))/2s^2 = x$ i.e., it is measured in the units of SNR, s^2. For the AWGN channel, finding the instanton means minimizing $l^2 = \sum_i (1-x_i)^2$ with respect to the noise vector $\mathbf{1} - \boldsymbol{x}$ in the N-dimensional space, under the condition that the decoding terminates with an error. Instanton estimation of FER at the higher SNR, $s \gg 1$ is $\sim \exp(-l_{\text{inst}}^2 . s^2/2)$ while at moderate values of SNR, many terms from the right-hand-side of Eq. 3.9 can contribute to FER comparably.

The downhill simplex method (amoeba) [100] can be used to find the minimum of a function of more than one independent variable [101]. A simplex is the geometrical figure consisting, in N dimensions, of $N+1$ points (or vertices) and all their interconnecting line segments, polygonal faces, etc. For example, in two dimensions, a simplex is a triangle. For multidimensional minimization, the algorithm is given a starting guess, that is, an N-vector of independent variables as the first point to try. The algorithm is then supposed to make its own way downhill through the unimaginable complexity of an N-dimensional topography, until it encounters a (local, at least) minimum. The downhill simplex method must be started not just with a single point, but with $N+1$ points, defining an initial simplex.

The instanton-amoeba evaluation is repeated many times, always starting from a new set for initial simplex chosen randomly. The length l, as a function of noise configuration inside the area of unsuccessful decoding, has multiple minima each corresponding to an instanton. Multiple attempts of the instanton-amoeba evaluations gives the instanton with the minimal l_{inst} plus the whole spectra of higher valued l_{inst}. Based on instanton, error floors are lowered in [102] where instantons are found and then a new Tanner code is constructed which is not prone to these instantons by construction.

3.5 Adaptive Importance Sampling

Another approach to simulate the error probability, which is not dependent on graph topologies, is the advanced version of IS known as Adaptive Importance Sampling (AIS) [103], [104], [105], [106]. AIS is quite attractive due to its potential for removing the burden of selecting a good IS density which is often system-specific. The key to this technique is the recognition that the subset of the simulation samples that yield an error event are distributed according to the (unknown) unconstrained optimal IS density.

The samples obtained may be used to estimate properties of the unconstrained optimal IS density and iteratively render the IS density closer to optimal in the sense that the measured properties (from the current simulation) are made to match those of the unconstrained optimal density. This opens a wide range of possibilities for adaptation rules, since the possible properties of interest range from the simple (e.g., the mean of the IS density) to the complex (e.g., the complete IS density). This approach has the advantage that the mechanics of the simulation remain the same for any system. This is extremely important for investigations into the sensitivity of the probability of error, to various system parameters that is usually determined by performing a series

of simulations with perturbed parameters. Some advances in the automatic selection of the IS density appear in [107].

In AIS, the probability density function is biased in a controlled way during multiple iterations thus making it possible to visit the tails of the noise distribution (Gaussian distribution e.g.). The system response (the output of the BP decoder, for example) is evaluated and stored in the process. AIS is especially interesting for its genericity in the sense that it can be adapted to any stationary memoryless channel (AWGN, BSC, etc.), to any type of decoder (Gallager B, BP, BCJR, etc.) and to any class of codes (regular and irregular LDPC codes, Turbocodes, etc.). An example of AIS technique, which does not take into account the graph structure of LDPC codes, has been successfully employed in [108] where the authors use a Dual Adaptive Importance Sampling (DAIS) technique using Multicanonical approach [109] based on Berg's recursion equations [110]. DAIS has been tested on a regular very short LDPC code of 96 bits.

3.5.1 Dual Adaptive Important Sampling

Dual Adaptive Importance Sampling (DAIS) evalutates the performance of an LDPC code using an AWGN channel using a novel technique based on Multicanonical Monte Carlo (MMC) simulations without a priori knowledge of how to bias. The main idea behind the technique is that a biased distribution must be chosen using some knowledge of which noise realizations most likely generate errors. This task is difficult when iterative decoding algorithms are used, since codeword errors are correlated to the noise distribution among the bits in a highly complex way.

The authors apply the Multicanonical Monte Carlo (MMC) simulation technique of [109] as the basis for their technique to compute very low error rates. They demon-

strate the DAIS technique using a (96, 50) LDPC code and Sum-Product Decoding (SPD) [111] with up to 50 decoder iterations achieving BER of 10^{-19}. Like standard IS [74], MMC increases the number of events in the tail of the Probability Density Function (PDF) being computed by sampling from a biased PDF [112]. The advantage of MMC is that it adaptively iterates to this biased PDF with little a priori knowledge needed of how to bias. The iterative procedure uses a control quantity to update the next iteration's biased PDF so that, as the iteration number increases, there tends to be an approximately equal number of hits in each control-quantity histogram bin [109].

A non-mathematical brief overview of the method is as follows (many steps involved are completely described in the next chapter): The noise vector is represented by a scalar control quantity which is calculated such that only destructive noise components contribute to the control quantity. The noise probability space is divided into a number of partitions called bins such that any noise realization generated corresponds to a bin through the scalar quantity. Instead of using physical intuition to guess the biasing PDF, this Multicanonical Monte Carlo algorithm iterates over a sequence of biasing PDFs which approach the optimal one.

The bias is determined by the vector $\mathbf{P}$ which keeps on evolving during the algorithm. Random walk is employed using Metropolis algorithm and noise realizations are produced in a controlled way such that the tails of the AWGN distribution are sufficiently explored. Two distinct histograms are produced during the algorithm which keep track of the noise samples produced and the errors produced which were not corrected by the decoder. The data of these two histograms and the vlaues of the vector $\mathbf{P}$ are manipulated to get the required results.

The result however, is not accurate due to the undersampling during the previ-

ously described ‘unconstrained’ simulation. In fact, Berg’s recursion equations [110] are used to update $\mathbf{P}$ values. With the increase in the number of samples, the update procedure becomes inefficient (will be discussed in detail in Chapter 4). As a result, in DAIS algorithm, one has to launch a second ‘constrained’ simulation. The random walker is constrained in the distribution where the result is a lot of errors after decoding. Mathematical manipulations are employed to get the result for constrained simulation which are then scaled through the process of curve-fitting to get the required final results.

3.5.2 Fast Flat Histogram Method

Fast Flat Histogram (FFH) method is inspired from statistical physics [11] while its mathematical framework remains that of an adaptive importance sampling. Like DAIS, FFH also iterates over biasing probability density functions gradually approaching the optimum one. The main emphasis of this work remained the implementation, validation and improvement of FFH method so the following chapters are dedicated to its elaboration and its successful application in the domain of information theory [113],[114],[115].

4

Fast Simulation for the Performance Evaluation of FEC Codes using Fast Flat Histogram Method [1]

IN digital communication systems, the quality of transmission system is usually measured by Frame Error Rate (FER) i.e., the ratio between the decoded frames that contain errors to the total number of frames sent through the system. With the advancement in the code design and better decoders, it has become very important to gauge the performance of the system at very low error rates. On the one hand, there are novel applications operating at very low error probability and standard Monte Carlo (MC) simulation takes extremely long to gauge their performance. On the other hand, new codes and decoders compete for a better performance in the error floor region where the performance evaluation is curbed and it is quite difficult to study the system properties.

The LDPC codes make a class of error correcting codes which are graphically represented by Tanner graph [13]. In the high Signal-to-Noise Ratio (SNR) region, the probability of error is dominated by decoder failures which do not correspond to erroneous codewords [108]. This is due to the convergence of decoder towards pseudo-

[1] This chapter was presented in parts in [113], [114]

codewords [50] or Trapping Sets (TS) [15]. Since the errors in the high SNR region are dominated by TS, a lot of work is oriented to study the LDPC codes performance in terms of TS. The main problem with the methods that aim at reducing simulation time based on the graph structure is that the identification of TS becomes extremely cumbersome if the graph connectivity is irregular or the code is long. In addition, the TS are not easily generalizable when the errors in the channel are produced owing to the presence of Additive White Gaussian Noise (AWGN) and an iterative soft decision decoding is employed.

In this work, we will initially focus on LDPC codes used on AWGN channel decoded with standard Belief Propagation (BP). We propose to bypass the limitations of DAIS [108] by using another AIS approach inspired by statistical physics called Fast Flat Histogram (FFH) method. We will show in particular that our method is still robust for relatively large codeword lengths up to 2640 coded bits. For turbo codes, we shall present the results for MPEG size 188 bytes. Since the code is duo-binary, the code-length is that of 3008 bits.

The rest of the chapter is organized as follows: Section 4.1 describes the FFH method detailing how it is applied to regular and irregular LDPC codes. Section 4.2 gives a comparison between DAIS and FFH showing that unlike DAIS, FFH is not dependent on histogram entries and is thus easily extendible to any codelength owing to a different update procedure. Section 4.3 gives the results for some typical test codes and quasi-cyclic codes from IEEE 802.11 standard using AWGN channel and an iterative soft decision decoder with BP in the probability domain. Section 4.4 details the application of FFH method to turbo codes. Section 4.5 gives some results on the statistical precision of our algorithm.

4.1 Fast Flat Histogram Method

4.1.1 Rationale

In statistical physics, one of the most important quantities is the density of states $g(E)$, i.e., the number of all possible states or configurations for an energy level E of the system. An estimation of this quantity through computer simulations is of great interest since it plays a major role for the study of phase transitions and critical phenomena.

Berg *et al.* [109], [110], [116], [117] presented the multicanonical ensemble method in which we have to estimate the density of states $g(E)$ first, then perform a random walk with a flat histogram in the desired region in the phase space. In a multicanonical simulation, the density of states need not necessarily be very accurate, as long as the simulation generates a relatively flat histogram and overcomes the barriers in energy space. This is because the algorithm employs a subsequent re-weighting which does not depend on the accuracy of the density of the states as long as the histogram can cover all important energy levels with sufficient statistics. If the density of states could be calculated very accurately, then the problem would have been solved in the first place and we need not perform any futher simulation such as with the multicanonical method.

Almost all recursive methods update the density of states by using the histogram data directly only after enough histogram entries are accumulated [109], [116], [118], [119], [120], [121], [122]. Due to the exponential growth of the density of states in energy space, this process is not efficient because the histogram is accumulated linearly. In [123], [124], the authors modify the density of states at each step of the random walk allowing them to approach the true density of states much faster than conventional methods especially for large systems. They also accumulate histogram entries

during the random walk but they only use it to check whether the histogram is flat enough to go to the next level random walk with a finer modification factor. The total number of configurations increases exponentially with the size of the system; however, the total number of possible energy levels increases linearly with the size of system. It is thus easy to calculate the density of states with a random walk in energy space for a large system.

The Fast Flat Histogram method [11] employing Wang Landau algorithm [123, 124] was introduced to estimate the density of states $g(E)$, i.e., the number of all possible states for an energy level E of the system. The algorithm is based on the observation that if a random walk in energy space is performed by flipping spins randomly for a spin system and the probability to visit a given energy level E is proportional to the reciprocal of the density of states $1/g(E)$, then a flat histogram is generated for the energy distribution. This is accomplished by modifying the estimated density of states in a systematic way to produce a flat histogram over the allowed range of energy and simultaneously making the density of states converge to the true value.

Wang-Landau algorithm has been used very efficiently in many statistical problems. Similar to the Metropolis algorithm, it is a generic algorithm, independent on the details of the physical system. Subsequently, there has been numerous studies on the algorithm itself and many proposals for improvements were put forward [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135] and studies of the efficiency and convergence of this method [128], [133]. Particularly, C. Zhou and R. N. Batt have given a mathematical analysis of the WL algorithm, proving its convergence and identifying sources of errors and strategies for optimization. Some theoretical aspects of the saturation of error are discussed in [136].

4.1.2 Description

The basic skeleton of our technique is the same as that in DAIS [108], that is we aim at increasing the number of events in the tails of the Probability Density Function (PDF) by sampling from a biased PDF [112]. However, our technique is somewhat different and solves some issues confronted by DAIS as will be explained in section 4.2.

We recall the notations from [108]. Let Γ be the n-dimensional probability space of the noise in the n bits of a codeword. The noise vector $\mathbf{z} = (z_1, z_2, ..., z_n)$ is a multivariate Gaussian with joint PDF $\rho(\mathbf{z}) = \prod_{l=1}^{n} \rho_l(z_l)$. The transmitted bit vector is represented by $\mathbf{b} = (b_1, b_2, ..., b_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_n)$ represents the received codeword. The algorithm is controlled by a scalar control quantity V given as $V(\mathbf{z}) = \left[\frac{1}{n}\sum_{l=1}^{n}[H(q_l z_l)z_l]^2\right]^{1/2}$ where $q_l = (-1)^{b_l}$ while b_l is the transmitted bit in the l^{th} position and $H(x) = 1$ if $x < 0$ and $H(x) = 0$ otherwise. $V(\mathbf{z})$ is contructed such that a noise component z_l contributes to V only if it may produce a bit error at the input to the decoder.

Given a range $[V_{\min}, V_{\max}]$ for V, Γ is partitioned into L subsets $\Gamma_k = \{\mathbf{z} \in \Gamma \mid V_{k-1} \leq V(\mathbf{z}) < V_k\}$, where $V_k = V_{\min} + k\Delta V$, $1 \leq k \leq L$ and $\Delta V = V_k - V_{k-1} = (V_{\max} - V_{\min})/L$ is the width of each bin in the partition of $[V_{\min}, V_{\max}]$. The number of bins depends on the code length and on the signal-to-noise ratio. We observe that an optimized number of bins is obtained by $L = 10^{\frac{1}{\sigma}} \times \frac{n}{10}$ where σ represents the standard deviation corresponding to the E_b/N_0 value.

Let P_k be the probability of selecting a realization $\mathbf{z}$ from ρ such that $\mathbf{z} \in \Gamma_k$ [112, 110]. Then,

$$P_k = \int_{\Gamma} \chi_k(\mathbf{z}) \frac{\rho(\mathbf{z})}{\rho^*(\mathbf{z})} \rho^*(\mathbf{z}) d\mathbf{z} \approx \frac{1}{N} \sum_{i=1}^{N} \chi_k(\mathbf{z}^{*,i}) \frac{\rho(\mathbf{z}^{*,i})}{\rho^*(\mathbf{z}^{*,i})} \tag{4.1}$$

where $\rho^*(\mathbf{z})$ is a positive biasing PDF, $\chi_k = 1$ if $\mathbf{z} \in \Gamma_k$ and $\chi_k(\mathbf{z}) = 0$ otherwise. $\mathbf{z}^{*,i}$ are N random sample points in Γ selected according to the PDF $\rho^*(\mathbf{z})$. The variance of the estimate of Eq. (4.1) is zero if the optimal biasing PDF $\rho^*_{opt}(\mathbf{z}) = \chi_k(\mathbf{z})\rho(\mathbf{z})/P_k$ is used. However, $\rho^*_{opt}(\mathbf{z})$ depends on P_k which is initially unknown. In standard IS, one uses physical intuition to guess a biasing PDF that is close to ρ^*_{opt}. Like DAIS, the FFH method instead iterates over a sequence of biasing PDFs $\rho^{*,j}$ that approach ρ^*_{opt}. We define $\rho^{*,j}$ for j^{th} iteration by $\rho^{*,j}(\mathbf{z}) = \rho(\mathbf{z})/(c^j P^j_k)$ where k is such that $\mathbf{z} \in \Gamma_k$ is satisfied. The quantities P^j_k satisfy $P^j_k > 0$ and $\sum_{k=1}^{M} P^j_k = 1$ and c^j is an unknown constant that ensures $\int_\Gamma \rho^{*,j}(\mathbf{z})d\mathbf{z} = 1$. The vector P_k completely determines the bias and is initialized with $1/L, \forall k = 1, ..., L$.

By employing Metropolis algorithm [137], we produce a random walk of *samples* $\mathbf{z}^{*,i}$ whose PDF equals $\rho^{*,j}(\mathbf{z})$. We consider a Markov chain of transitions consisting of small steps in the noise space. Each transition goes from $\mathbf{z}^{*,i} = \mathbf{z}^*_a \in \Gamma_{k_a}$ to $\mathbf{z}^*_b = (\mathbf{z}_a + \epsilon\Delta\mathbf{z}) \in \Gamma_{k_b}$ where $\Delta\mathbf{z}$ is random and symmetric, i.e., it does not favor any direction in Γ and the transition is accepted with probability π_{ab}. ϵ here is the perturbation constant. If a transition from $\mathbf{z}^{*,i}$ to $\mathbf{z}^*_b$ is accepted, we set $\mathbf{z}^{*,i+1} = \mathbf{z}^*_b$, else we set $\mathbf{z}^{*,i+1} = \mathbf{z}^{*,i} = \mathbf{z}^*_a$. The ratio π_{ab}/π_{ba} equals $\rho^{*,j}(\mathbf{z}^*_b)/\rho^{*,j}(\mathbf{z}^*_a)$ which is the *detailed balance equation* that ensures that the limiting (stationary) PDF for infinitely many steps of this random walk is $\rho^{*,j}$ [137].

We consider the perturbation of the noise component in each bit $z^*_{a,l}$ of $\mathbf{z}^*_a$ separately and accept it or reject it independently with the probability

$$\min[\rho(z^*_{b,l})/\rho(z^*_{a,l}), 1]$$

We pick each perturbation Δz_l from a zero mean symmetric PDF. We obtain a trial state $\mathbf{z}^*_b$ in which only some of the components are different from their previous values

in $\mathbf{z}_a^*$. Then we compute k_b, the bin corresponding to $\mathbf{z}_b^*$ and finally accept the step from $\mathbf{z}_a^*$ to $\mathbf{z}_b^*$ with the probability $\min[P_{k_a}^j)/P_{k_b}^j), 1]$. The compound transition probability thus becomes

$$\pi_{ab} = \left\{ \prod_{l=1}^{n} \min\left[\frac{\rho(z_{b,l}^*)}{\rho(z_{a,l}^*)}, 1 \right] \right\} \min\left[\frac{P_{k_a}^j}{P_{k_b}^j}, 1 \right] \tag{4.2}$$

The Asymptotically Optimal Acceptance Rate AOAR $\alpha \triangleq (\text{number of accepted steps}) \div (\text{total number of steps})$ for a Metropolis algorithm for target distributions with IID components is 0.234 [138]. The perturbation constant ϵ is adjusted so as to keep α close to this value. The noise realizations are recorded in the histogram $H^{*,j}$ where $H_k^{*,j} = \sum_{i=1}^{N} \chi_k(\mathbf{z}^{*,i})$ is the number of $\mathbf{z}^{*,i}$ in iteration j that fall into Γ_k. To keep a record of errors in bin k, we produce an error histogram $G_k^{*,j}$. P_k is updated *on the fly* such that when k bin is visited, P_k is modified by the refinement parameter $f > 1$, i.e. $P_k \rightarrow P_k \cdot f$ [123, 124]. In practice, we have to use the log domain $\ln P_k \rightarrow \ln P_k + \ln f$ in order to fit all possible P_k into double precision numbers. If the random walk rejects a possible move and stays in the same bin k, we modify the same P_k with the modification factor to keep the detailed balance equation in equilibrium.

The above procedure is complemented by the implementation details of Metropolis algorithm [137] as follows:

The AWGN Probability Density Function PDF ρ is defined as

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(- \frac{(x^2 - \mu)}{2\sigma^2} \right)$$

The following steps are to be repeated for each noise component in a noise vector

1. Pick a noise component x_a in the noise vector (this operation is to be repeated

for all noise components and is referred to as *a priori* choice [139], or "proposed update").

2. Add to it a random quantity obtained within a scale of standard deviation to get x_b (i.e., *attempt* a move *update*);

3. Calculate $\rho(x_b)/\rho(x_a)$.

4. Draw a number r at random between 0 and 1 and

   • accept the attempted move if $r < \rho(x_b)/\rho(x_a)$, i.e., the next configuration in the Markov chain has x_b as the noise component.

   • otherwise reject the move, i.e., the next configuration in the Markov chain is the same as the current configuration having the noise component x_a.

The previous set of operations is termed a *Monte Carlo step*. A *Monte Carlo sweep* (MCS) corresponds to N attempts on a noise vector containing N noise components.

While using the log domain, we have to respect the upper and lower limits which can be represented by an exponential function as a double precision number. For our machines, this limit is slightly more than e^{700} and is slightly less than e^{-740}. However, we fix e^{700} and e^{-740} as our upper and lower limits respectively. During the execution of the WL iterations, P_k is modified in the log domain such that the modification factor is added to the P_k value. When the WL iteration is over, the values of P_k are such that they largely exceeds the upper and lower limits. To counteract this problem, we need to bring the P_k values within the boundaries so that they may be represented as double precision numbers once the anti-log is used. Since the P_k values correspond to different bin numbers represented by k, it is extremely important to keep the ratio between the P_k values. We copy the P_k values to a temporary storage so that we

may perform operations on these values without changing the order and sequence of the original P_k vlaues. Now we perform the sorting operation using the bubble sort algorithm. Once we get the sorted sequence, we look for the median of the sorted P_k values. We subtract this median value from every P_k vlaue in the original storage. The subtraction operation is chosen since we are in the log domain. The desired effect is that of division by the same number once an antilog operation is performed on all the P_k values. After the subtraction operation in the log domain, most of the P_k values are within the calculable limits. A check is run on all the values. If any P_k value is found to be superior than 700, it is changed to 700. On the lower side, if any values is found to be inferior than -740, it is changed to -740. Now, the antilog operation can be carried out to get the P_k values which can be used in the normalization procedure. We point out that the evolution of P_k is a Markov process, although the WL algorithm is not, because it makes references to its entire history.

The histogram $H_k^{*,j}$ is checked after about each $L \times 10$ Monte Carlo (MC) sweeps. When the histogram is flat (flatness criterion is the same as in [123, 124]), the modification factor is reduced to a finer one using the function $f_{j+1} = \sqrt{f_j}$ ($f_{\text{init}} = e = 2.7182818$), the histogram is reset and the next iteration of random walk is started where P_k are now modified with the finer modification factor.

We continue doing so until the histogram is flat again and then we begin the next Wang-Landau (WL) iteration with a finer f and so on. We stop the random walk when the change from one WL iteration to the other is “quite small”. The above detailed random walk can also be carried out in a parallel fashion by dividing the range $[V_{\min}, V_{\max}]$ into W partitions and then exploring each partition separately, combining the results in the end.

It is important to note that before starting the random walk for any WL iteration, the noise vectors are initialized to the values obtained from an AWGN direct sample generator (employing Box-Muller method for example). Throughout this work, the intial value of modification factor $f_{\mathrm{init}} = e = 2.7182818$ has been used. If f_{init} is too small, the random walk will spend an extremely long time to reach all possible levels. However, too large a choice of f_{init} will lead to large statistical errors [123], [124]. In addition, it is very clear that the modification factor acts as one of the most important control parameters for the accuracy of the algorithm and also determines how many MC sweeps are necessary for the whole simulation. The accuracy of the results also depends on the complexity and size of the system, criterion of the flat histogram and other details of the implementation of the algorithm.

Here, the flatness criterion deserves some discussion. It is impossible to obtain a perfectly flat histogram and the phrase "flat histogram" here means that the histogram $H^{*,j}$ for the whole range is not less than $x\%$ of the average histogram $\langle H^{*,j} \rangle$ where $x\%$ is chosen according to the size and complexity of the system and the desired accuracy of the results ($x = 90$ in this work). There are other ways to measure the flatness of the histogram. For example, one may simply want to compute the percentage of histogram bins that departs from the baseline by more than a given amount, or one may estimate it from the standard deviation of the logarithm of the histogram. Incidentally, estimation of histogram flatness is a feature shared by all estimators working in an iterative manner.

The convergence of the algorithm towards the flat noise samples distribution is somewhat tedious to prove on rigorous grounds [128], [140], yet the intuitive picture is that, as soon as the noise sample distribution has become flat, the noise samples having the same V level occur with the same frequency and thus - for a Markov chain

of infinite length - the effect is just to translate the whole curve P_k vertically by a global amount. If the sample distribution is flat in the last step, we also have that P_k is an estimator for the probability desnsity, with a relative uncertainty which ideally amounts to $\sqrt{\ln f}$ [128]. The situation is actually somewhat more intricate, because other parameters impinge on the global uncertainty, including the number of entries in the histogram at the end of each iteration, and correlations between successive measurements. In addition, the maximum accuracy affordable with the method was also reported to be limited by construction, irrespective of the number of MCS performed as a whole [141],[142], yet in the meantime it was also suggested that subtle choices of parameters may greatly help in taming several sources of error [128].

The square root function has been chosen to reduce the modification factor and f approaches 1 as the number of iterations approaches ∞. In fact, any function may be used as long as it decreases f monotonically to 1 [123],[124]. A simple and efficient formula is $f_{i+1} = f_i^{1/n}$ where $n > 1$. The value of n can be chosen according to the available CPU time and the expected accuracy of the simulation. The choice of $n = 2$ yields good accuracy in a relatively short time, even for large systems.

It is extremely important to determine the optimum $[V_{\min}, V_{\max}]$ interval since the accuracy and speed of the simulation depends heavily on it. Our aim is to explore the whole of probabilty space Γ using random walk [143]. $[V_{\min}, V_{\max}]$ is initialized to $[0, 1]$ and this interval is divided into L bins. Now the random walk is performed to determine the optimum $[V_{\min}, V_{\max}]$ interval. The value of P_k is updated for every Markov Chain transition during the walk. After a number of steps (we use $L \times E_b/N_0$ steps), the walk is ceased and the farthest bins on either side are detected which were approached by the random walk. These two bins on either side determine the optimum $[V_{\min}, V_{\max}]$ interval.

Let P_{err} be the probability that a received word with noise realization $\mathbf{z}$ selected from ρ leads to an error and $P_{\text{err},k}$ the probability that $\mathbf{z}$ leads to an error *and* falls into bin k. Then

$$P_{\text{err},k} = P_{\text{err}|k} P_k \tag{4.3}$$

$$P_{\text{err}} = \sum_{k=1}^{M} P_{\text{err},k} \tag{4.4}$$

where $P_{\text{err}|k}$ is the conditional probability of an error *given* that $\mathbf{z}$ falls into bin k. We can approximate $P_{\text{err}|k} \approx P_{\text{err}|k}^{j_{\max}} = \sum_{j=1}^{j_{\max}} G_k^{*,j} / \sum_{j=1}^{j_{\max}} H_k^{*,j}$ after $j_{\max}$ iterations where $j_{\max}$ is the iteration when f gets very close to 1 and we stop further refinement of the modification factor. Using (4.3) and (4.4), we get P_{err}.

4.2 Comparison between DAIS and FFH Method

In this section, we explain in detail the main difference between FFH method and DAIS of [108]. We also point out why the FFH method partially solves the limitations confronted by DAIS.

DAIS is using Berg's recursion equations which are explained as follows:
To update P_k^j at the end of iteration j, P_1^{j+1} is initially set to an arbitrary positive value. Then, the recursion equations are [112, 110]

$$P_{k+1}^{j+1} = \frac{P_k^{j+1} P_{k+1}^j}{P_k^j} \left(\frac{H_{k+1}^{*,j}}{H_k^j} \right)^{\hat{g}_k^j} \tag{4.5}$$

where

$$\hat{g}_k^j = \frac{g_k^j}{\sum_{l=1}^{j} g_k^l}, \qquad g_k^l = \frac{H_k^{*,l} H_{k+1}^{*,l}}{H_k^{*,l} + H_{k+1}^{*,l}} \tag{4.6}$$

where in addition, $\hat{g}_k^j = 0$ if $g_k^j = 0$ and $g_k^l = 0$ if $H_k^{*,l} + H_{k+1}^{*,l} = 0$. The exponent $0 \leq \hat{g}_k^j \leq 1$ hence depends on all previous iterations. Finally, P_k^{j+1} is normalized so that $\sum_{k=1}^{L} P_k^{j+1} = 1$.

The histograms are accumulated linearly so obviously the value $\hat{g}_k^j$ will decrease with larger histograms rendering the update process of P_k very slow. Intrinsically, Berg's equations converge to the optimum value of P_k without sampling enough smaller noise realizations leading to errors. To overcome this problem of undersampling, one has to launch another simulation constrained in a different V range.

The above mentioned problem is solved by FFH method employing Wang-Landau Algorithm since the dynamic update of P_k is independant of the histogram values. Modifying P_k at each step of the random walk allows one to approach the optimum value of P_k in a very quick and efficient manner. The noise samples leading to errors are accumulated with enough statistics hence there is no need to carry out another simulation. FFH method thus works for powerful codes having low error floors at low SNR and for codes having large block-lengths.

4.3 Simulation Results for LDPC codes

For all our test-benches, Sum-Product (Belief Propagation in probability domain) decoding algorithm has been used. BPSK modulation is employed using symmetric signal levels of $+1$ and -1 for logical 0s and 1s respectively. An all zeros codeword is transmitted since the code is linear and the noise is symmetric.

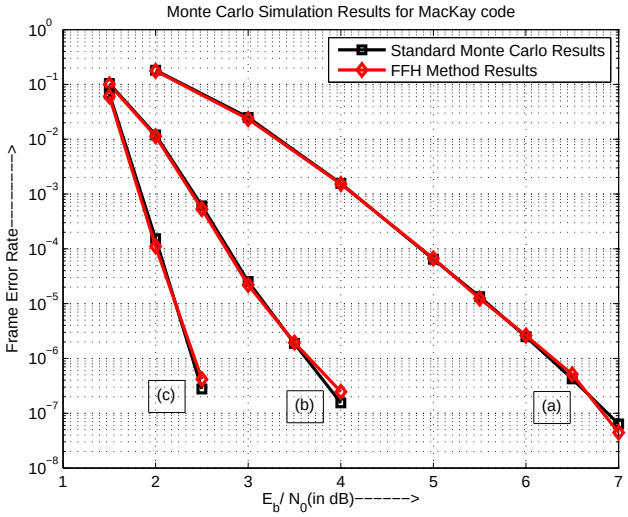

Figure 4.1: Results for MacKay codes (a) $n = 120$, $R = 0.47$ regular (b) $n = 504$, $R = 0.5$ irregular (c) $n = 2640$, $R = 0.5$ regular

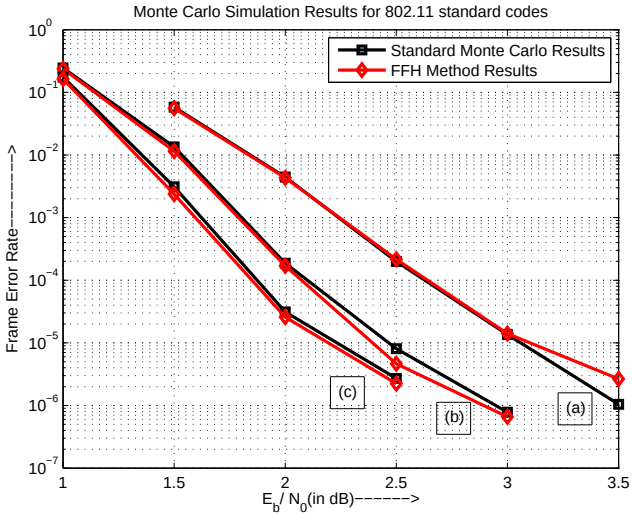

Figure 4.2: Results for IEEE 802.11 standard Quasi-cyclic irregular codes (a) $n = 648$, $R = 0.5$ (b) $n = 1296$, $R = 0.5$, (c) $n = 1944$, $R = 0.5$

Our test bench comprises of three codes obtained from [4] and three quasi-cyclic LDPC codes used in IEEE 802.11 standard [144]. The simulation results are shown in

Code (E_b/N_0)	MC Codewords (100 Err. Fr.)	FFH Codewords	Simul. Gain
(120,0.5,reg) (7)	3594801754	63636462	56.49
(504,0.5,irr) (4)	1274520408	12081492	105.49
(2640,0.5,reg) (2.5)	1445433333	8312500	173.88
(648,0.5,irr) (3.5)	161136000	6465688	24.92
(1296,0.5,irr) (3)	324935483	6645033	48.9
(1944,0.5,irr) (2.5)	78796296	6002175	13.13

Table 4.1: Simulation Gain for LDPC codes

Figs. 4.1 & 4.2. The first remark is that our simulation results stick to the MC curves for all codes and for all FERs. This was though expected since the FFH method implements an unbiased estimation of FER. The efficiency of the FFH method as compared to standard MC can be measured by the simulation gain [69] i.e., the ratio of codewords simulated in MC simulation to those simulated in FFH method. The simulation gain for all the codes in test-benches is given in Table 4.1. The gain is quite impressive and the result is a considerable reduction in simulation time. They are determined for the highest E_b/N_0 for the particular code as indicated by the values within parenthesis in column 1. The codewords simulated in case of standard MC simulation are for 100 erroneous frames. Our current experience suggests that the simulation gain increases with decreasing FER but the dependence of FFH on the number of codewords or code length is unknown at this time and is a subject of continuing research.

4.4 Application of FFH to Turbo codes

Before presenting the results of application of FFH to turbo codes, some state of the art in the context of turbo codes is given.

Turbo codes [2] make a class of FEC codes which are graphically represented through a trellis diagram. In the high SNR region, the probability of error is dominated mainly by the distance properties of turbo codes which depend on the interleaver design between the two constituent encoders. For turbo codes, an IS method based on the distance properties and error patterns was presented in [145].

FFH is especially interesting for its genericity in the sense that it can be adapted to any stationary memoryless channel (AWGN, BSC, etc.), to any type of decoder (turbo decoder, Gallager B, BP, etc.) and to any class of codes (binary and duo-binary turbo codes, regular and irregular LDPC codes, etc.). Here, we focus on duo-binary turbo codes [146] used on AWGN channel decoded with standard duo-binary turbo decoder. We have employed an improved FFH method with a new self-adaptive procedure to determine the optimum sampling domain corresponding to the code, a more stringent stopping criterion and some additional steps to tailor the traditional Wang Landau algorithm according to our objectives. We show the application of the improved FFH method for duo-binary turbo codes used in DVB-RCS standard [147] with varying code rates thus validating the genericity of the method for FEC codes.

4.4.1 Improvements in the algorithm

4.4.1.1 Increase in robustness

In case of rejection of a possible move while going through Metropolis MC step, a very significant additional step is to permute the components of the noise vector and to

add this permuted sequence to the transmitted codeword on which decoding is carried out for error correction. It is important to note that the random walk is performed with the new permuted sequence so as not to disturb the detailed balance equilibrium. We keep on permuting the noise components until a possible move is accepted.

The preceding step stems from the fact that different sequences of the same components of a noise vector lead to different decoding outputs since we are employing a message passing decoding algorithm. The orientation of the permuted noise components remain the same leading to the same V value and consequently staying in the same bin. Without effecting the basic modification of P_k values, we are thus able to check the system response of all entries in histogram $H^{*,j}$ thus adding to the robustness of the method. It is to be noted that if the proposed noise vectors which move the system outside the permitted $[V_{\min}, V_{\max}]$ interval are systematically rejected, the P_k value would increase at edges by an unwanted excessive amount. This problem is counteracted by adopting the n-fold way [148], i.e., leaving P_k value unchanged whenever a move update attempts to take the system outside the allowed interval.

4.4.1.2 Self-adaptive optimum V interval

It is extremely important to determine the optimum $[V_{\min}, V_{\max}]$ interval with the optimum number of bins since the accuracy and speed of the simulation depend heavily on it. Following is a self adaptive procedure to determine this interval which intrinsically takes into account the code length and the code error correcting capacity. Lines of similarity can be drawn between our procedure of determining the optimum $[V_{\min}, V_{\max}]$ interval and Domain Sampling Run of [129].

$[V_{\min}, V_{\max}]$ is initialized to $[0, 1]$ and this interval is divided into 1000 bins. Let

the Global Acceptance Ratio (GAR) correspond to the ratio of the number of accepted noise vectors to the number of noise vectors produced in total. We initialize GAR with a value (0.99 in our case). The bins are initialized with $P_k = 1/1000, \forall k = 1, ..., 1000$. Now the random walk is performed to produce noise vectors for which the corresponding bins are visited with the consequent update of the P_k value. With the bin filling, we start getting rejections for the proposed move. At each step, we calculate the GAR and as soon as we obtain its pre-defined value, the walk is ceased and the farthest bins on either side are detected which were approached by the random walk. These two bins on either side determine the $[V_{\min}, V_{\max}]$ interval. While determining the interval, the noise vectors produced are not added to the codewords and no decoder runs are performed. Their sole purpose is to locate the bins naturally accessible for the code.

The $[V_{\min}, V_{\max}]$ interval is to be divided into a suitable number of bins. Our choice of number of bins depends on the bin width, an important parameter for the accuracy control [149]. We define B, a control parameter to determine the optimum number of bins using the relation $L = (V_{\max} - V_{\min})B$ (rounded off to the nearest integer). The bins are now initialized with $P_k = 1/L, \forall k = 1, ..., L$. The random walk is performed to produce noise vectors for which the corresponding bins are visited with the consequent update of the P_k value. The noise vectors are added to the codeword (in a permuted sequence in case of rejection) in the channel and decoding is performed for the noisy received vector. If we do not get errors and we reach a flat histogram, we reiterate over the above two steps by again choosing a natural $[V_{\min}, V_{\max}]$ interval for $\mathrm{GAR} = \mathrm{GAR} - \Delta\mathrm{GAR}$ where $\Delta\mathrm{GAR} = 0.01$. With each step, we increase the number of bins by $B_{i+1} = 1.5B_i$. If we get errors before reaching a flat histogram, we take the current $[V_{\min}, V_{\max}]$ interval with the optimum number of bins. We then continue on to perform the WL iterations within this optimum interval.

4.4.1.3 Improved stopping criterion

For each and every WL iteration, we calulate the P_{err} value. We stop the random walk when $\max_k \mid (P_{\text{err}}^{j} - P_{\text{err}}^{j+1})/P_{\text{err}}^{j} \mid < 0.1$. This criterion should be satisfied for 2 consecutive WL iterations so as to avoid all surges and to ensure the convergence of the simulation. In addition, such a stringent criterion ensures enough samples for both histograms thus ensuring a good level of accuracy. The P_{err} value has been chosen as the convergence parameter since it takes into account both noise sample and error histograms over all previous WL iterations.

4.4.2 Simulation Results for Turbo codes

The standard Digital Video Broadcast with Return Channel via Satellite (DVB-RCS) [147] 8-state, 188 bytes (MPEG-size) duo-binary turbo code was used as a test bench. Maximum possible iterations are 8 with 4 quantization bits. BPSK modulation is employed using symmetric signal levels of $+1$ and -1 for logical 0s and 1s respectively. An all zeros codeword is transmitted since the code is linear and the noise is symmetric.

DVB-RCS was chosen as a test bench since it's performance is well known. It employs a powerful and complex coding system so the error correcting capacity is very high. The error floors are steep and the Minimum Hamming Distance (MHD) is high. The simulation results for 5 different code rates for MPEG-size DVB-RCS are shown in Fig. 4.3. The first remark is that our simulation results stick to the MC curves for all code rates and for all FERs.

The efficiency of the FFH method as compared to standard MC can be measured by the simulation gain [69] i.e., the ratio of codewords simulated in MC simulation to those simulated in FFH method. The simulation gain for all code rates for MPEG-size

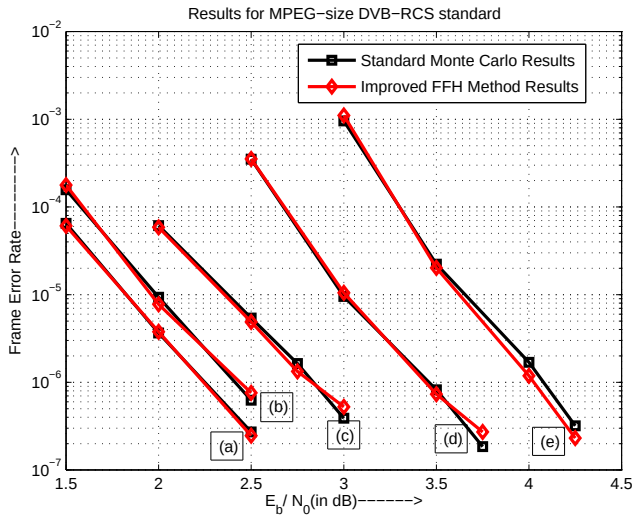

Figure 4.3: Results for MPEG-size DVB-RCS standard (a) $R = 0.5$ (b) $R = 0.4$ (c) $R = 0.5$ (d) $R = 0.67$ (e) $R = 0.75$

DVB-RCS is given in Table 4.2.

Code Rate (E_b/N_0)	MC Codewords	FFH Codewords	Simul. Gain
0.33 (2.5 dB)	367476540	24315960	15.11
0.4 (2.5 dB)	159946900	15247500	10.49
0.5 (3 dB)	256226600	30425510	8.42
0.67 (3.75 dB)	539287910	38629450	13.96
0.75 (4.25 dB)	313316840	30530120	10.26

Table 4.2: Simulation Gain for Turbo codes

The gain is good and the result is a considerable reduction in simulation time. They are determined for a high E_b/N_0 value for the particular code rate as indicated by the values within parenthesis in column 1. For FFH method, the decoder runs include codeword simulations for the permuted sequence of noise samples in case of random walk step rejection. Our current experience suggests that the simulation gain increases

with decreasing FER and the FFH method is intricately dependent on the code length, rate and error correcting capacity.

The results are reported in terms of simulation gain which is a measure of the simulation length. All simulations were performed using different linux clusters containing machines of different flop rates. To report the results on simultion time reduction, both Monte Carlo and FFH simulations have to be performed on machines with the same flop rates. To show that we have to pay the price of FFH algorithm execution itslef, the MC and FFH simulations were performed on the same machine. For 188 bytes, duo-binary turbo code of rate 0.5 with channel SNR 3 dB, FFH takes around 47 hours as compared to around 176 hours of MC simulation showing a simulation time gain of around 3.74. FFH method is characterized by the acceptance of only 23.4% of the noise samples produced. In addition, once a noise realization is obtained, it can be accepted or rejected depending upon the Markov chain step.

4.5 Statistical Precision

To measure the statistical precision of our results and to compare them with that of Monte Carlo, we performed a large number of simulations for different code rates. Our test-bench was the duo-binary turbo-code of DVB-RCS standard with MPEG size (188 bytes). We performed 30 simulations per code rate for 5 code rates with different seeds for the random number generator.

Let X_i represent the FER obtained then $\bar{X}$ stands for the mean (average) value $\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$. The standard deviation is then given as

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})}{n}}$$

For comparison purposes, we choose the relative density which is defined as

$$\sigma_{\mathrm{rel}} = \frac{\sigma}{\bar{X}} \times 100$$

which is represented in terms of percentage. The results we obtained are presented as under:

From the results (Table 4.3 & 4.4), it is evident that in terms of precision, the FFH method is still less precise when compared to the standard MC.

Rate = 0.33 (2 dB)	Rate = 0.4 (2 dB)	Rate = 0.5 (2.5 dB)	Rate = 0.67 (3 dB)	Rate = 0.75 (3.5 dB)
3.67E-006	9.33E-006	4.73E-006	1.33E-005	2.05E-005
3.80E-006	7.71E-006	4.70E-006	1.08E-005	2.05E-005
3.31E-006	8.11E-006	4.67E-006	1.00E-005	2.51E-005
3.78E-006	9.99E-006	5.09E-006	8.18E-006	2.20E-005
3.18E-006	7.68E-006	5.23E-006	1.01E-005	1.85E-005
4.05E-006	8.58E-006	5.75E-006	8.84E-006	2.04E-005
3.55E-006	7.56E-006	4.66E-006	9.86E-006	1.96E-005
4.24E-006	8.74E-006	4.26E-006	1.10E-005	2.45E-005
4.23E-006	7.89E-006	4.75E-006	1.03E-005	2.24E-005
4.10E-006	9.03E-006	4.62E-006	9.80E-006	2.34E-005
3.49E-006	8.65E-006	3.94E-006	1.14E-005	2.50E-005
4.05E-006	9.81E-006	4.97E-006	1.02E-005	2.10E-005
4.17E-006	8.36E-006	3.83E-006	1.00E-005	2.10E-005
2.84E-006	7.73E-006	4.75E-006	1.10E-005	1.93E-005
3.43E-006	7.44E-006	5.60E-006	9.93E-006	1.89E-005
3.89E-006	7.70E-006	5.18E-006	8.39E-006	2.47E-005
3.50E-006	9.44E-006	5.07E-006	1.07E-005	1.98E-005
3.66E-006	8.35E-006	4.37E-006	1.01E-005	1.97E-005
4.27E-006	8.45E-006	4.45E-006	1.02E-005	1.94E-005
3.72E-006	7.75E-006	5.19E-006	1.14E-005	2.43E-005
3.42E-006	7.65E-006	5.26E-006	1.08E-005	1.88E-005
3.71E-006	6.82E-006	4.22E-006	1.02E-005	2.37E-005
3.59E-006	7.44E-006	5.12E-006	1.09E-005	2.34E-005
3.59E-006	9.30E-006	4.56E-006	7.91E-006	1.91E-005
3.42E-006	9.24E-006	4.15E-006	9.29E-006	2.12E-005
3.04E-006	7.95E-006	5.09E-006	9.27E-006	2.35E-005
4.02E-006	8.58E-006	5.15E-006	1.10E-005	1.98E-005
4.20E-006	7.31E-006	4.45E-006	8.33E-006	2.06E-005
3.70E-006	7.73E-006	4.82E-006	1.15E-005	2.06E-005
3.69E-006	8.44E-006	4.50E-006	1.08E-005	2.04E-005
Average 3.71E-006	Average 8.29E-006	Average 4.77E-006	Average 1.02E-005	Average 2.14E-005
Rel. St. Dev. 9.70%	Rel. St. Dev. 9.45%	Rel. St. Dev. 9.45%	Rel. St. Dev. 11.05%	Rel. St. Dev. 9.60%

Table 4.3: Monte Carlo Simulation Results for Duo-Binary Turbo codes for MPEG size (188 bytes) and different code rates

Rate = 0.33 (2 dB)	Rate = 0.4 (2 dB)	Rate = 0.5 (2.5 dB)	Rate = 0.67 (3 dB)	Rate = 0.75 (3.5 dB)
4.53E-006	1.13E-005	4.41E-006	1.10E-005	2.11E-005
4.94E-006	8.11E-006	4.94E-006	1.42E-005	1.85E-005
3.18E-006	1.25E-005	5.37E-006	9.21E-006	3.07E-005
4.32E-006	9.07E-006	4.88E-006	9.05E-006	2.13E-005
4.04E-006	1.01E-005	4.20E-006	9.10E-006	1.84E-005
4.55E-006	8.13E-006	4.99E-006	8.75E-006	2.38E-005
5.27E-006	9.31E-006	5.34E-006	1.13E-005	2.40E-005
4.93E-006	8.05E-006	4.89E-006	8.63E-006	1.70E-005
4.39E-006	1.01E-005	5.86E-006	1.06E-005	2.04E-005
5.39E-006	1.05E-005	4.05E-006	1.05E-005	1.74E-005
4.64E-006	9.59E-006	5.64E-006	1.07E-005	1.94E-005
5.65E-006	9.21E-006	3.27E-006	1.02E-005	1.83E-005
4.24E-006	9.46E-006	4.95E-006	1.02E-005	1.89E-005
4.91E-006	9.18E-006	5.51E-006	1.30E-005	1.94E-005
3.81E-006	8.58E-006	4.45E-006	9.55E-006	2.02E-005
4.42E-006	7.60E-006	3.02E-006	9.52E-006	2.37E-005
3.55E-006	1.04E-005	4.73E-006	1.15E-005	2.34E-005
2.40E-006	8.90E-006	4.41E-006	9.79E-006	2.07E-005
4.60E-006	9.39E-006	4.70E-006	9.41E-006	1.91E-005
3.61E-006	1.04E-005	4.33E-006	1.02E-005	1.70E-005
4.41E-006	9.06E-006	4.76E-006	9.82E-006	1.78E-005
4.62E-006	1.30E-005	4.90E-006	8.59E-006	1.93E-005
4.35E-006	8.40E-006	4.51E-006	9.49E-006	2.20E-005
4.66E-006	1.06E-005	5.54E-006	9.53E-006	2.04E-005
4.37E-006	7.01E-006	3.88E-006	9.15E-006	1.96E-005
3.52E-006	1.06E-005	5.08E-006	1.02E-005	1.31E-005
4.73E-006	1.09E-005	3.83E-006	1.14E-005	2.09E-005
4.37E-006	8.65E-006	4.30E-006	1.08E-005	2.24E-005
3.94E-006	1.19E-005	2.53E-006	1.03E-005	2.01E-005
3.73E-006	1.00E-005	5.37E-006	8.85E-006	1.88E-005
Average 4.34E-006	Average 9.67E-006	Average 4.62E-006	Average 1.02E-005	Average 2.02E-005
Rel. St. Dev. 15.27%	Rel. St. Dev. 14.24%	Rel. St. Dev. 16.48%	Rel. St. Dev. 12.20%	Rel. St. Dev. 14.89%

Table 4.4: Fast Flat Histogram Method Results for Duo-Binary Turbo codes for MPEG size (188 bytes) and different code rates

5

An Efficient Pseudo-Codeword Search Algorithm for Belief Propagation Decoding of LDPC Codes

Abstract

We introduce the use of Fast Flat Histogram (FFH) method employing Wang Landau Algorithm in an adaptive noise sampling framework using Random Walk to find out the pseudo-codewords and consequently the pseudo-weights for the Belief Propagation (BP) decoding of LDPC codes over an Additive White Gaussian Noise (AWGN) channel. The FFH method enables us to tease out pseudo-codewords at very high Signal-to-Noise Ratios (SNRs) exploring the error floor region of a wide range of codes varying in length and structure. We present the pseudo-weight (effective distance) spectra for these codes and analyze their respective behavior [1].

5.1 Introduction

Low Density Parity Check (LDPC) codes [3] make a class of forward error correcting codes which employ a computationally efficient iterative decoding scheme based on a

[1] The contents of this chapter were presented in [115]

message passing algorithm. The decoding process is, however, known to be subject to decoding failures due to the so-called pseudo-codewords. The failures can cause the high Signal-to-Noise Ratio (SNR) performance of message passing decoding to be worse than that predicted by the maximum likelihood decoding union bound in the error floor regime [15] which is characterized by very low error rates. Standard Monte Carlo (SMC) simulation which consists of simulating a system by generating random inputs according to a probability distribution and then evaluating the system response, becomes extremey cumbersome at such high SNRs. With the advancement in the code design and better decoders, it has become very important to gauge the performance of the system in the error floor regime.

To explore the error floor phenomenon, a physics inspired approach coined as instanton amoeba was proposed and developed in [65], [150], [99]. The scheme is generic in that there are no restrictions related to decoding or channel. Chertkov *et al.* [64] presented the pseudo-codeword landscape using an efficient pseudo-codeword search algorithm detailed in [62]. The algorithm is mainly valid for Linear Programming (LP) decoding [52] and the authors reported that a direct attempt to extend the LP-based pseudo-codewords search algorithm to Belief Propagation (BP) decoding [3],[4] did not yield desirable results.

In the pseudo-codeword literature, LP decoding has been predominantly used as it proposes to relax the polytope, expressing $\boldsymbol{\sigma}$ in terms of a linear combination of local codewords. If the LP decoding does not decode to a correct codeword then it usually yields a non-codeword pseudo-codeword which is a special configuration of beliefs containing some rational values [61]. Pseudo-codewords are not codewords in general but codewords are pseudo-codewords [58]. The nature of pseudo-codewords with different origins is further investgated in [54],[55].

To characterize the pseudo-codewords, the notion of fundamental polytope was introduced in [40] which is the most important concept relevant to pseudocodewords found through LP decoding. It was also argued that the large minimum distance of the code does not determine the performance of the code if the code has low pseudo-weight spectrum. For the sum-product decoding, if the messages are converged, then the vector formed by the marginal probabilities of having a bit position in the state 1 is a fundamental polytope vector. More about this vector and the Bethe variational free energy can be found in [60].

In this work, we investigate the use of a physics inspired algorithm known as Fast Flat Histogram (FFH) method [11] which has already been implemented for the efficient performance evaluation of forward error correcting codes [113], [114]. The method consists of a random walk scheme employing Wang-Landau algorithm in adaptive noise sampler framework. We employ our scheme to find the pseudo-codewords in the high SNR region using the BP decoding algorithm on an AWGN channel.

The rest of the work is organized as follows: Section 5.2 sets the background for the use of FFH method employing BP decoding for the search of pseudo-codewords. Section 5.3 gives our numerical scheme and its validation. Section 5.4 reports the results for our test-bench containing regular, irregular and cyclic LDPC codes. Section 5.5 concludes our work.

5.2 Notations and background

The background and notations for this work remains the same as in [99],[64]. Sending a codeword $\boldsymbol{\sigma} = \{\sigma_i = \pm 1; i = 1, \ldots, N\}$ into a noisy channel results with the probability $P(\boldsymbol{x}|\boldsymbol{\sigma})$ in corruption of the original signal, $\boldsymbol{x} \neq \boldsymbol{\sigma}$. The decoding goal is

to infer the original message from the received output $\boldsymbol{x}$. Assuming that coding and decoding are fixed, one studies Frame Error Rate (FER) to characterize performance of the scheme $\mathrm{FER} = \int d\boldsymbol{x} \chi_{\mathrm{error}}(\boldsymbol{x}) P(\boldsymbol{x}|\mathbf{1})$, where $\chi_{\mathrm{error}} = 1$ if an error is detected and $\chi_{\mathrm{error}} = 0$ otherwise. In symmetric channel, FER is invariant with respect to the original codeword, thus all-(+1) codeword can be assumed for the input.

AWGN channel is defined by,

$$P(\boldsymbol{x}|\boldsymbol{\sigma}') = \prod_i p(x_i|\sigma_i'), p(x|\sigma) \propto \exp\left(-\frac{(x-\sigma)^2 s^2}{2}\right) \tag{5.1}$$

If the detected signal at a bit is x, the respective likelihood at the bit is $h = p(x|1)/p(x|-1)$. These likelihoods are translated into beliefs $b_i(\sigma_i)$ which are defined as trial probabilities for bit i to be in the state σ_i. Belief Propagation constitutes in iteratively propagataing messages though different nodes of the code graph following message update rules and computing beliefs using certain non-linear equations called BP equations. The BP equations are equations for extrema of the Bethe free energy [60].

At high SNR, the difference between the performance of Maximum Likelihood (ML) decoding and approximate decoding (BP, LP, etc.) is due to the pseudo-codewords. This performance is gauged in terms of Frame Error Rate (FER) which calculates the probability of a decoding failure. For AWGN channel, the actual asymptotics of the performance curves (FER vs. SNR) of ML and BP decoding at very high SNRs (s^2), in the so-called error-floor region, are $\mathrm{FER}_{\mathrm{ML}} \sim \exp(-d_{\mathrm{ML}} \cdot s^2/2)$ and $\mathrm{FER}_{\mathrm{BP}} \sim \exp(-d_{\mathrm{BP}} \cdot s^2/2)$ where d_{ML} is the Hamming Distance of the code and d_{BP} is the effective distance of the code, specific to BP decoding.

BP decoding turns into LP decoding at SNR $\to \infty$. In the high SNR (error

floor) region, the values of FER are inaccessible by Monte-Carlo simulations. It is in this context that we use FFH method which consists of a Markov Chain Monte Carlo (MCMC) sampler capable of sampling the noise vectors from the tails of the AWGN probability density function. Suppose a pseudo-codeword $\tilde{\boldsymbol{\sigma}} = \{\tilde{\sigma}_i = b_i(1); i = 1, \ldots, N\}$ corresponding to the most damaging configuration of the noise (instanton) is found. Then the effecive distance is given by the same formula $d_{\text{eff}} = (\sum_i \tilde{\sigma}_i)^2 / \sum_i \tilde{\sigma}_i{}^2$ as in [64],[40]. This definition of the effective distance was first described in [18] where the formulas derived by Wiberg *et al.* [16], [17] for AWGN channels were extended to non-binary codes, Binary Erasure Channel (BEC) and Binary Symmetric Channel (BSC).

5.3 Fast Flat Histogram Method

5.3.1 Description

The basic skeleton of our technique is the same as that in [113],[114]. Let Γ be the n-dimensional probability space of the noise in the n bits of a codeword. The noise vector $\boldsymbol{z} = (z_1, z_2, ..., z_n)$ is a multivariate Gaussian with joint pdf $\rho(\boldsymbol{z}) = \prod_{l=1}^{n} \rho_l(z_l)$. The transmitted bit vector is represented by $\boldsymbol{t} = (t_1, t_2, ..., t_n)$ and $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ represents the received codeword. The algorithm is controlled by a scalar control quantity V given as $V(\boldsymbol{z}) = \left[\frac{1}{n}\sum_{l=1}^{n}[t_l - z_l]^2\right]^{1/2}$ where t_l and z_l are the transmitted bit and the noise value in the l^{th} position respectively. This definition of $V(\boldsymbol{z})$ is different from the one that we used in [113],[114].

Given a range $[V_{\min}, V_{\max}]$ for V, Γ is partitioned into L subsets $\Gamma_k = \{\boldsymbol{z} \in \Gamma | V_{k-1} \leq V(\boldsymbol{z}) < V_k\}$, where $V_k = V_{\min} + k\Delta V$, $1 \leq k \leq L$ and $\Delta V = V_k - V_{k-1} = (V_{\max} - V_{\min})/L$ is the width of each bin in the partition of $[V_{\min}, V_{\max}]$.

Let P_k be the probability of selecting a realization $\boldsymbol{z}$ from ρ such that $\boldsymbol{z} \in \Gamma_k$ [108]. Then,

$$P_k = \int_{\Gamma} \chi_k(\boldsymbol{z}) \frac{\rho(\boldsymbol{z})}{\rho^*(\boldsymbol{z})} \rho^*(\boldsymbol{z}) d\boldsymbol{z} \approx \frac{1}{N} \sum_{i=1}^{N} \chi_k(\boldsymbol{z}^{*,i}) \frac{\rho(\boldsymbol{z}^{*,i})}{\rho^*(\boldsymbol{z}^{*,i})} \tag{5.2}$$

where $\rho^*(\boldsymbol{z})$ is a positive biasing pdf, $\chi_k = 1$ if $\boldsymbol{z} \in \Gamma_k$ and $\chi_k(\boldsymbol{z}) = 0$ otherwise. $\boldsymbol{z}^{*,i}$ are N random sample points in Γ selected according to the pdf $\rho^*(\boldsymbol{z})$. The variance of the estimate of (5.2) is zero if the optimal biasing pdf $\rho^*_{\text{opt}}(\boldsymbol{z}) = \chi_k(\boldsymbol{z})\rho(\boldsymbol{z})/P_k$ is used. However, $\rho^*_{\text{opt}}(\boldsymbol{z})$ depends on P_k which is initially unknown. In standard IS, one uses physical intuition to guess a biasing pdf that is close to ρ^*_{opt}. The FFH method instead iterates over a sequence of biasing pdfs $\rho^{*,j}$ that approach ρ^*_{opt}. We define $\rho^{*,j}$ for j^{th} iteration by $\rho^{*,j}(\boldsymbol{z}) = \rho(\boldsymbol{z})/(c^j P_k^j)$ where k is such that $\boldsymbol{z} \in \Gamma_k$ is satisfied. The quantities P_k^j satisfy $P_k^j > 0$ and $\sum_{k=1}^{L} P_k^j = 1$ and c^j is an unknown constant that ensures $\int_{\Gamma} \rho^{*,j}(\boldsymbol{z}) d\boldsymbol{z} = 1$. The vector P_k completely determines the bias and is initialized with $1/L, \forall k = 1, ..., L$.

Our aim is to explore the whole of probabilty space Γ using random walk [143]. By employing Metropolis algorithm [137], we produce a random walk of *samples* $\boldsymbol{z}^{*,i}$ whose pdf equals $\rho^{*,j}(\boldsymbol{z})$. We consider a Markov chain of transitions consisting of small steps in the noise space. Each transition goes from $\boldsymbol{z}^{*,i} = \boldsymbol{z}^*_a \in \Gamma_{k_a}$ to $\boldsymbol{z}^*_b = (\boldsymbol{z}_a + \epsilon \Delta \boldsymbol{z}) \in \Gamma_{k_b}$ where $\Delta \boldsymbol{z}$ is random and symmetric, i.e., it does not favor any direction in Γ and the transition is accepted with probability π_{ab}. Here, ϵ is the perturbation constant. If a transition from $\boldsymbol{z}^{*,i}$ to $\boldsymbol{z}^*_b$ is accepted, we set $\boldsymbol{z}^{*,i+1} = \boldsymbol{z}^*_b$, else we set $\boldsymbol{z}^{*,i+1} = \boldsymbol{z}^{*,i} = \boldsymbol{z}^*_a$. The ratio π_{ab}/π_{ba} equals $\rho^{*,j}(\boldsymbol{z}^*_b)/\rho^{*,j}(\boldsymbol{z}^*_a)$ which is the *detailed balance equation* that ensures that the limiting (stationary) pdf for infinitely many steps of this random walk is $\rho^{*,j}$ [137].

We consider the perturbation of the noise component in each bit $z^*_{a,l}$ of $\boldsymbol{z}^*_a$ sepa-

rately and accept it or reject it independently with the probability

$$\min[\rho(z^*_{b,l})/\rho(z^*_{a,l}), 1]$$

We pick each perturbation Δz_l from a zero mean symmetric pdf. We obtain a trial state $\boldsymbol{z}^*_b$ in which only some of the components are different from their previous values in $\boldsymbol{z}^*_a$. Then we compute k_b, the bin corresponding to $\boldsymbol{z}^*_b$ and finally accept the step from $\boldsymbol{z}^*_a$ to $\boldsymbol{z}^*_b$ with the probability $\min[P^j_{k_a})/P^j_{k_b}), 1]$. The compound transition probability thus becomes

$$\pi_{ab} = \left\{ \prod_{l=1}^{n} \min\left[\frac{\rho(z^*_{b,l})}{\rho(z^*_{a,l})}, 1\right] \right\} \min\left[\frac{P^j_{k_a}}{P^j_{k_b}}, 1\right] \tag{5.3}$$

The Asymptotically Optimal Acceptance Rate AOAR $\alpha \triangleq$ (number of accepted steps)/(total number of steps) for a Metropolis algorithm for target distributions with IID components is 0.234 [138]. The perturbation constant ϵ is adjusted so as to keep α close to this value. The noise realizations are recorded in the histogram $H^{*,j}$ where $H^{*,j}_k = \sum_{i=1}^{N} \chi_k(\boldsymbol{z}^{*,i})$ is the number of $\boldsymbol{z}^{*,i}$ in iteration j that fall into Γ_k. Each noise vector is used in the channel to deteriorate the transmitted codeword which is then fed into the decoders to verify if the errors are corrected within the specifies number of decoder iterations.

P_k is updated *on the fly* such that when k bin is visited, P_k is modified by the refinement parameter $f > 1$, i.e. $P_k \rightarrow P_k \cdot f$ [123, 124]. In practice, we have to use the log domain $\ln P_k \rightarrow \ln P_k + \ln f$ in order to fit all possible P_k into double precision numbers. If the random walk rejects a possible move and stays in the same bin k, we modify the same P_k with the modification factor to keep the detailed balance equation in equilibrium.

In case of rejection of a possible move, a very significant additional step is to permute the components of the noise vector and to add this permuted sequence to the transmitted codeword on which decoding is carried out for error correction. It is important to note that the random walk is performed with the new permuted sequence so as not to disturb the detailed balance equilibrium. We keep on permuting the noise components until a possible move is accepted. The preceding step stems from the fact that different sequences of the same components of a noise vector lead to different decoding outputs since we are employing a message passing decoding algorithm. The orientation of the permuted noise components remain the same leading to the same V value and consequently staying in the same bin. Without effecting the basic modification of P_k values, we are thus able to check the system response of all entries in histogram $H^{*,j}$ thus adding to the robustness of the method.

It is to be noted that if the proposed noise vectors which move the system outside the permitted $[V_{\min}, V_{\max}]$ interval are systematically rejected, the P_k value would increase at edges by an unwanted excessive amount. This problem is counteracted by adopting the n-fold way [148], i.e., leaving P_k value unchanged whenever a move update attempts to take the system outside the allowed interval.

The histogram $H_k^{*,j}$ is checked after about each $10L$ Monte Carlo (MC) sweeps. When the histogram is flat (flatness criterion is the same as in [123, 124]), the modification factor is reduced to a finer one using the function $f_{j+1} = \sqrt{f_j}$ ($f_{\text{init}} = e = 2.7182818$), the histogram is reset and the next iteration of random walk is started where P_k are now modified with the finer modification factor. We continue doing so until the histogram is flat again and then we begin the next Wang-Landau (WL) iteration with a finer f and so on. The above detailed random walk can also be carried out in a parallel fashion by dividing the range $[V_{\min}, V_{\max}]$ into W partitions and then

exploring each partition separately, combining the results in the end.

It is extremely important to determine the optimum $[V_{\min}, V_{\max}]$ interval with the optimum number of bins since the accuracy and speed of the simulation depend heavily on it. Following is a self adaptive procedure to determine this interval which intrinsically takes into account the code length and the code error correcting capacity. Lines of similarity can be drawn between our procedure of determining the optimum $[V_{\min}, V_{\max}]$ interval and Domain Sampling Run of [129].

$[V_{\min}, V_{\max}]$ is initialized to $[0, 5]$ and this interval is divided into 1000 bins. Let the Global Acceptance Ratio (GAR) correspond to the ratio of the number of accepted noise vectors to the number of noise vectors produced in total. We initialize GAR with a value (0.3 in our case). The bins are initialized with $P_k = 1/1000, \forall k = 1, ..., 1000$. Now the random walk is performed to produce noise vectors for which the corresponding bins are visited with the consequent update of the P_k value. With the bin filling, we start getting rejections for the proposed move. At each step, we calculate the GAR and as soon as we obtain its pre-defined value, the walk is ceased and the farthest bins on either side are detected which were approached by the random walk. These two bins on either side determine the $[V_{\min}, V_{\max}]$ interval. While determining the interval, the noise vectors produced are not added to the codewords and no decoder runs are performed. Their sole purpose is to locate the bins naturally accessible for the code.

5.4 Results and discussion

Our test-bench consists of six codes namely Tanner [155, 64, 20] code [13], Margulis $p = 7$ [672, 336, 16] code [151]; [648, 324, 15], [1296, 648, 23] and [1944, 972, 27] codes from the 802.11 draft [144] and the [504, 252, 13] irregular Progressive Edge Growth (PEG) code [152]. The MHDs of the last four codes are measured through

the improved impulse method [153]. BPSK modulation is employed using symmetric signal levels of +1 and -1 for logical 0s and 1s respectively. An all zeros codeword is transmitted since the code is linear and the noise is symmetric. We employ 1000 decoding iterations in our BP decoding so the pseudo-codewords correspond to the instantons which could survive such a high number of decoding iterations.

Figs. 5.1 & 5.2 depict the frequency spectra for the codes under study. For [155, 64, 20] and [672, 336, 16] codes, our conclusions are the same as in [64]. We observe that the two codes demonstrate qualitatively different features for the pseudo-codeword frequency spectra. Pseudo-codeword spectrum for [155, 64, 20] code starts with the lowest effective distance ≈ 10.004 and grows up going through the fundamental polytope pseudo-codewords at effective distance ≈ 19.98 (convergence to valid codewords). In case of [672, 336, 16] code, the spectrum starts at effective distance ≈ 12.056 but grows abruptly to the fundamental polytope pseudocodewords at effective distance ≈ 15.66 (valid codewords).

For the 802.11 cyclic LDPC codes, we observe that there is a significant increase in effective distance with the increase in code-length. As compared to the preceding two codes, these three cyclic codes exhibit a relatively high number of fundamental polytope pseudo-codewords. Convergence to invalid codewords increases with the increase in code-length. The least effective distances of the fundamental polytope pseudo-codewords for [648, 324, 15], [1296, 648, 23] and [1944, 972, 27] codes that we found are 14.64 (valid codeword), 26.48 (invalid codeword) and 64.24 (invalid codeword) respectively. In the case of irregular PEG code, the pseudo-codeword spectrum is similar to the cyclic codes. The least effective distance ≈ 9.855 and the least effective distance of fundamental polytope pseudo-codeword is 12.74 (valid codeword). The valid codeword fundamental polytope pseudo-codeword were termed as *unde-*

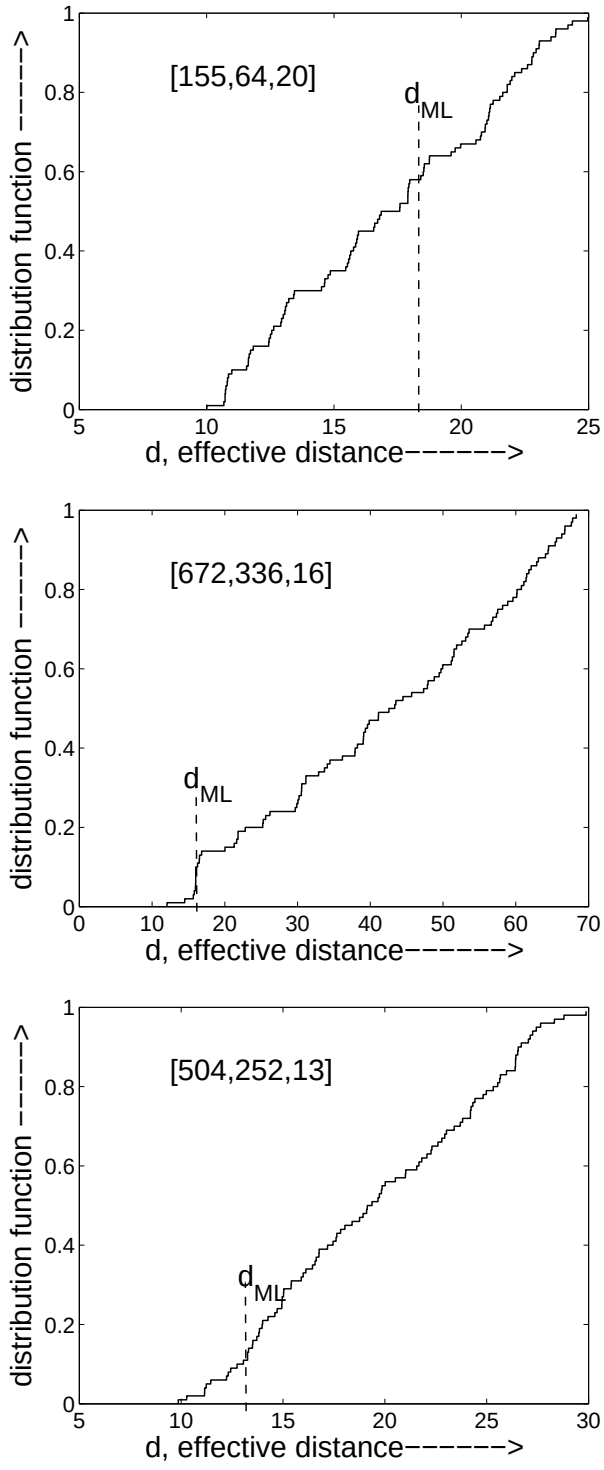

Figure 5.1: The frequency spectrum of the effective distance contructed using FFH method as the pseudo-codeword search algorithm for the codes: Tanner [155, 64, 20]; Margulis $p = 7$ [672, 336, 16]; irregular progressive edge growth [504, 252, 13].

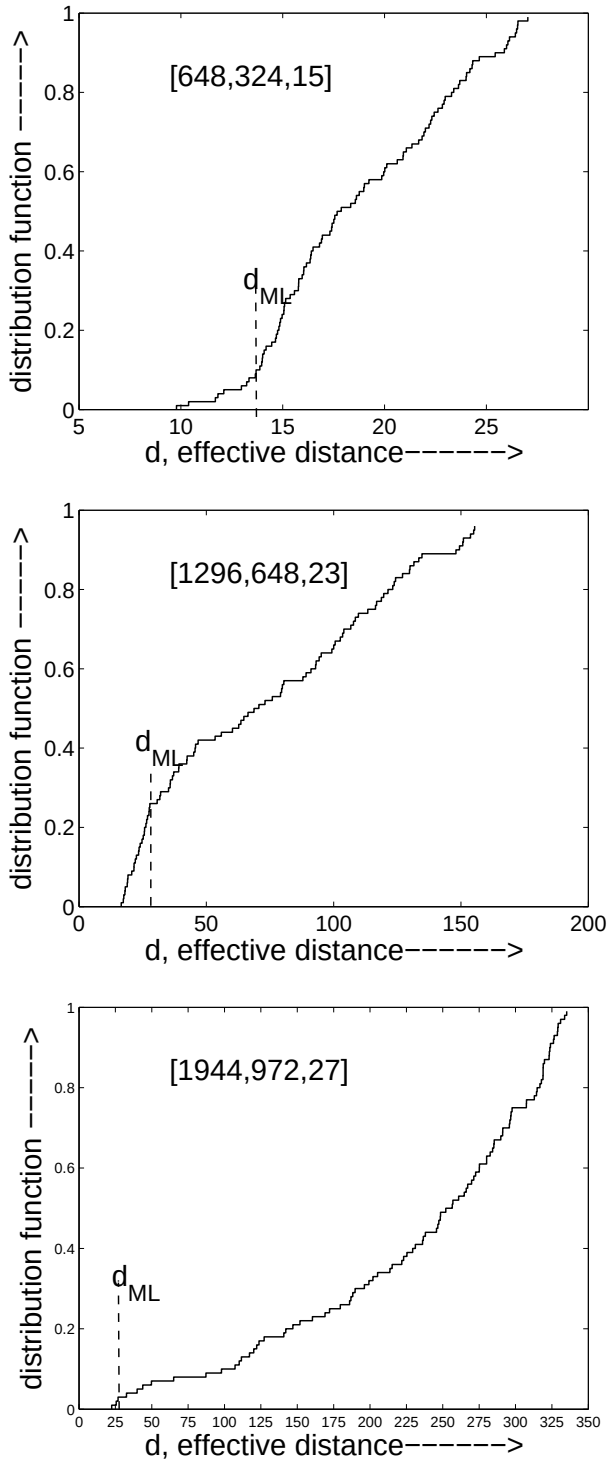

Figure 5.2: The frequency spectrum of the effective distance contructed using FFH method as the pseudo-codeword search algorithm for the codes: 802.11 standard [648, 324, 15], [1296, 648, 23] and [1944, 972, 27].

tected errors in [14].

We also applied our method to Margulis $p = 11$ [2640, 1320, 40] code and found the least effective distance ≈ 54.055. We observe that the fundamental polytope pseudo-codewords are in dominance however the messages converge to invalid codewords. The observations made by Koetter *et al.* [40] that the pseudo-weights are far more important than the Hamming Distance are further bolstered when we analyze the pseudo-codeword spectra in our case. For example, the Hamming Distance of [1944, 972, 27] code is only slightly higher than [1296, 648, 23] code, however there is a big difference in their pseudo-codeword spectra. Similarly, the Hamming Distance of [155, 64, 20] code is higher than [672, 336, 16] code, the latter performs better in terms of pseudo-weight. How this pseudo-weight spectrum depends on the code-length or other code properties is a subject of ongoing research.

5.5 Conclusions

In this paper, we have proposed the use of Fast Flat Histogram method to find the pseudocodewords for different codes using BP decoding. The FFH method is a powerful tool to explore the code performance at very high SNRs (in the error floor region) which is otherwise computationally intractable using standard Monte Carlo simulation. Since the decoder failures in the error floor region are mostly due to pseudo-codewords, the FFH method is an excellent means to study the code behavior at high SNRs. Our future work consists of integrating the FFH method in multiple error impulse framework to increase its effectiveness.

5.6 Acknowledgements

We wish to thank Pascal O. Vontobel for the fruitful discussions reltated to this work. His website www.pseudocodewords.info also proved very useful. We are grateful to

Mikhail G. Stepanov for providing the matrices of Margulis [672, 336, 16] and Tanner [155, 64, 20].

6

Conclusions and perspectives

6.1 Conclusions

We started the work with a detailed study of the state-of-the-art. The inference that we drew from this study of scientific literature is that the already existing methods, which depend on the study of the code structure, face constraints when the code is long and/or irregular. In terms of genericity, Adaptive Importance Sampling (AIS) methods seemed very promising. Dual Adaptive Importance sampling [108] laid the basic foundations for the use of AIS methods in the domain of coding theory. It successfully employed Metropolis Hastings algorithm to produce a random walk of noise samples. The backbone of the AIS methods is the recursive update process. DAIS uses Berg's equations which are not efficient for the sampling purposes. To counteract the problem of undersampling, we propose the use of Fast Flat Histogram method.

In this work, we have presented the successful migration of Fast Flat Histogram Method from Statistical Physics to Information Theory for the efficient performance evaluation of forward error correcting codes. The main characteristic feature of the method is that it is generic in nature i.e., valid for any FEC code, communication channel, decoding algorithm etc. We presented the results for the LDPC codes and Turbo codes on AWGN channel employing message passing algorithm. The FFh method

allows us to pick up the noise samples from the tail of the Gaussian distribution, resulting in rare event sampling. Using FFH, we have shown that performance evaluation of FEC codes can be carried out relatively faster.

6.2 Perspectives

The following points constitute interesting perspectives to the work presented in this thesis.

- A lot of work has been dedicated to the improvement of Wang-Landau algorithm itself. It would be quite interesting to study these improvements and to employ them in the information theory framework to increase the efficiency and to reduce the statistical errors.

- Regarding the precision of the results obtained with FFH method, a lot still remains to be desired.

6.3 Lessons Learnt

After the study of existing methods and problems therein, we chose to go for a method which could be applicable to all codes. Naturally, such a method could be an advanced and modified version of Monte Carlo simulation. FFH is a Markov Chain Monte Carlo method employing an efficient combination of Random Walk (Metropolis-Hastings algorithm) and Wang-Landau algorithm. However, after the detailed study and implementation of FFH method, we observe that like Monte Carlo method, it also suffers from constraints.

6.3 Lessons Learnt

The deteriorating effects of noise samples decrease significantly (according to the exponential function) with the increase in SNR. Once we cross the waterfall region and enter the error floor region, the errors are mostly due to (a) the sub-optimal nature of the decoder and (b) the weaknesses in the graph-structure. In the error floor region, FFH is more eficient as compared to standard Monte Carlo. However, with a slight increase in SNR in the error floor region, even FFH method takes time (though less than MC) to explore the tails of probability distribution.

“After all these years, I do not know what I may appear to the world, but to myself, I seem to have been only a boy playing on the sea-shore and diverting myself in, now and then, finding a smoother pebble or a prettier shell than ordinary whilst the great ocean of truth lay all undiscovered before me.” — Isaac Newton

Bibliography

[1] Claude E. Shannon. "A mathematical theory of communication". *Bell System Technical Journal*, 27, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. "Near Shannon limit error-correcting coding and decoding: turbo-codes". In *ICC '93*, pages 1064–1070, Geneva, Switzerland, May 1993.

[3] R. Gallager. *"Low-Density Parity-Check Codes"*. PhD thesis, MIT, 1963.

[4] David J. C. MacKay. "Good error correcting codes based on very sparse matrices". *IEEE Trans. Inf. Theory*, 45(2):399–431, March 1999.

[5] T. Richardson, M. Shokrollahi, and R. Urbanke. "Design of capacity approaching irregular low-density parity-check codes". *IEEE Trans. Inf. Theory*, 47(2):619–637, Feb. 2001.

[6] T. Richardson and R. Urbanke. "Efficient encoding of low-density parity-check codes". *IEEE Trans. Inf. Theory*, 47(2):638–656, Feb. 2001.

[7] M. Yang, W. E. Ryan, and Y. Li. "Design of efficiently encodable moderate-length high-rate irregular LDPC codes". *IEEE Trans. Commun.*, 52(4):564–571, Apr. 2004.

[8] D. Burshtein and G. Miller. "Bounds on the performance of Belief Propagation decoding". *IEEE Trans. Inf. Theory*, 48:112–122, Jan. 2002.

[9] T. Richardson, A. Shokrollahi, and R. Urbanke. "Design of Capacity-approaching Irregular Low-Density Parity-Check Codes". *IEEE Trans. Inf. Theory*, 47:619–637, Feb. 2001.

[10] M. Sipser and D. Spielman. "Expander Codes". *IEEE Trans. Inf. Theory*, 42:1710–1723, Nov. 1996.

[11] S. Reynal and H. T. Diep. "Fast flat histogram algorithm for generalized spin models". *Phys. Rev.*, E 72(56710), 2005.

[12] S. Wilson. *"Digital Modulation and Coding"*. Prentice-Hall, 1996.

[13] R. M. Tanner. "A recursive approach to low complexity codes". *IEEE Trans. Inf. Theory*, vol 27, Sep. 1981.

[14] D. MacKay and M. Postol. "Weaknesses of Margulis and Ramanujan LDPC codes". *Electronic Nodes in Theoretical Computer Science*, 74:808–821, 2003.

[15] T. Richardson. "Error floors of LDPC codes". *Allerton Conference*, 2001.

[16] N. Wiberg. *"Codes and decoding on general graphs"*. PhD thesis, Linkoping University, 1996.

[17] N. Wiberg, H-A. Loeliger, and R. Koetter. "Codes and iterative decoding on general graphs". *Europ. Trans. Telecommunications*, 6(513), 1995.

[18] Jr. G.D. Forney, R. Koetter, F.R. Kschischang, and A. Reznik. "On the effective weights of pseudocodewords for codes defined on graphs with cycles". *Codes, Systems and graphical models (Minneapolis, MN, 1999) IMA Vol; Math. Appl.*, 123:101–112, 2001.

[19] Y. Weiss and W. T. Freeman. "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs". *IEEE Trans. Inf. Theory*, 47(2):736–744, 2001.

[20] N. Sourlas. "Spin-glass models as error-correcting codes". *Nature*, 338:693–695, 1989.

[21] ETSI EN 302 307 v1.1.2.

[22] J. Pearl. *"Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference"*. Morgan Kaufmann Publishers Inc., 1988.

[23] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inf. Theory*, 42:429–445, Mar. 1996.

[24] Chad A. Cole. *"Determining Performance of LDPC Codes at Low Error Probability"*. PhD thesis, University of Virginia, Aug. 2006.

[25] W.E. Ryan. "An introduction to LDPC codes". http://www.ece.arizona.edu/%7Eryan/New%20Folder/ryan-crc-ldpc-chap.pdf, Aug. 2003.

[26] M. Fossorier. "Iterative reliability-based decoding of LDPC codes". *IEEE JSAC*, 19:908–917, 2001.

[27] X.Y. Hu, E. Eleftheriou, D.M. Arnold, and A. Dholakia. "Efficient implementation of the sum-product algorithm for decoding LDPC codes". In *IEEE GlobeComm'01*, page 1036–1036E, Nov. 2001.

[28] J.-C. Chen, D. Lu, J. Sadowsky, and K. Yao. "On importance sampling in digital communications - part I: fundamentals". *IEEE JSAC*, 11(3):289–299, April 1993.

[29] J.S. Sadowsky and J.A. Bucklew. "On large deviation theory and asymptotically efficient Monte-Carlo estimation". *IEEE Trans. Inf. Theory*, 36(3):579–588, May 1990.

[30] T. Richardson and R. Urbanke. *"Modern Coding Theory"*. Cambridge University Press, Mar. 2008.

[31] S. Benedetto and G. Montorsi. "Unveiling Turbo codes: Some results on parallel concatenated coding schemes". *IEEE Trans. Inf. Theory*, 42, Mar. 1996.

[32] Y. Mao and A. H. Banihashemi. "A heuristic search for good low-density parity-check codes at short block lengths". In *IEEE International Conference on Communications*, volume 1, page 41, New York, 2001.

[33] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R.L. Urbanke. "Finite-length analysis of LDPC codes on the BEC". *IEEE Trans. Inf. Theory*, 48(6):1570–1579, June 2002.

[34] P.O. Vontobel and R. Koetter. "Lower bounds on the minimum pseudo-weight of linear codes". In *IEEE Intern. Symp. on Inf. Theory*, page 70, Chicago IL, July 2004.

[35] L. Sun, H. Song, Z. Keirn, and B.V.K.V. Kumar. "Field programmable gate array (FPGA) for iterative code evaluation". *IEEE Trans. on Magnetics*, 42(2):226–231, Feb. 2006.

[36] L. Yang, H. Liu, and R. Shi. "Code construction and FGPA implementation of capacity approaching low error-floor LDPC decoder". *IEEE Trans. on Circuits and Systems*, 2006.

[37] S. Laendner and O. Milenkovic. "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes". In *Wireless Comm.*, Hawaii, USA, 2005.

[38] O. Milenkovic, E. Soljanin, and P. Whiting. "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles". *IEEE Trans. Inform. Theory*, 53(1):39–55, 2007.

[39] C.C. Wang, S.R. Kulkarni, and H.V. Poor. "Exhaustion of error-prone patterns in LDPC codes". *IEEE Trans. Inform. Theory*, 55(5):1976–1998, May 2009.

[40] R. Koetter and P.O. Vontobel. "Graph-covers and iterative decoding of finite-length codes". In *3rd Intl. Symp. on Turbo Codes and Realted Topics*, pages 75–82, Brest, France, Sept. 2003.

[41] F. Kschischang, B. Frey, and H.A. Loeliger. "Factor graphs and the sum-product algorithm". *IEEE Trans. Inf. Theory*, 47(2):498–519, Feb 2001.

[42] T. Richardson and R. Urbanke. "The capacity of low-density parity-check codes under message-passing decoding". *IEEE Trans. Inf. Theory*, 47(2):599–618, Feb. 2001.

[43] L. Bazzi, T. J. Richardson, and R. L. Urbanke. "Exact threshold and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A". *IEEE Trans. Inf. Theory*, 50:2010–2021, Sept. 2004.

[44] N. Miladinovic and M.P.C. Fossorier. "Improved bit-flipping decoding of low-density parity-check codes". *IEEE Trans. Inf. Theory*, 51(4):1594–1606, April 2005.

[45] P. Zarrinkhat and A. H. Banihashemi. "Threshold values and convergence properties of majority-based algorithms for decoding regular low-density parity-check codes". *IEEE Trans. Commun.*, 52:2087–2097, Dec. 2004.

[46] H. Xiao and A.H. Banihashemi. "Estimation of bit and frame error rates of low-density parity-check codes on binary symmetric channels". *IEEE Trans. Commun.*, 55:2234–2239, Dec. 2007.

[47] E. Cavus, C.L. Haymes, and B. Daneshrad. "An IS simulation technique for very low BER performance evaluation of LDPC codes". In *IEEE Int. Conf. Comm.*, page 1095–1100, Istambul, Turkey, June 2006.

[48] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright. "Investigation of error floors of structured low-density parity-check codes via hardware simulation". In *GLOBECOM 2006*, San Francisco, CA, Oct.-Nov. 2006.

[49] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic. "Analysis of absorbing sets for array-based LDPC codes". In *IEEE Int. Conf. Comm.*, Glasgow, United Kingdom, June 2007.

[50] B.J. Frey, R. Koetter, and A. Vardy. "Skewness and pseudocodewords in iterative decoding". In *IEEE Intern. Symp. on Inf. Theory*, June 1998.

[51] C. Kelley, D. Sridhara, J. Xu, and J. Rosenthal. "Pseudocodeword weights and stopping sets". In *IEEE Intern. Symp. on Inf. Theory*, page 67, Chicago, IL, 2004.

[52] J. Feldman, M. Wainright, and D.R. Karger. "Using linear programming to decode linear codes". In *Conference on Information Sciences and Systems*, The John Hopkins University, Mar. 2003.

[53] P.O. Vontobel and R. Koetter. "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes". *IEEE Trans. Inf. Theory*, May 2007.

[54] N. Axvig, D. Dreher, K. Morrison, E. Psota, L.C. Perez, and J. Walker. "A universal theory of decoding and pseudocodewords". In *Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, Sep. 2007.

[55] C. Kelley and D. Sridhara. "Pseudocodewords of Tanner graphs". *IEEE Trans. Inf. Theory,*, 53(11):4013–4038, Nov. 2007.

[56] S.T. Xia and F.W. Fu. "Minimum pseudoweight and minimum pseudocodewords of LDPC codes". *IEEE Trans. Inform. Theory*, 54(1):480–485, Jan. 2008.

[57] R. Smarandache and P.O. Vontobel. "Pseudo-codeword analysis of Tanner graphs from projective and Euclidean planes". *IEEE Trans. Inform. Theory*, 53(7):2376–2393, 2007.

[58] R. Smarandache, A. Pusane, P. O. Vontobel, and Jr. D. J. Costello. "Pseudo-codewords in LDPC convolutional codes". In *IEEE Intern. Symp. on Inf. Theory*, Seattle, USA, July 2006.

[59] J. Feldman, M. Wainwright, and D.R. Karger. "Using linear programming to decode binary linear codes". *IEEE Trans. Inform. Theory*, 51(3):954–972, Mar. 2005.

[60] J.S. Yedidia, W.T. Freeman, and Y. Weiss. "Constructing free energy approximations and generalized belief propagation algorithms". *IEEE Trans. Inf. Theory*, IT-51(2282), 2005.

[61] P.O. Vontobel and R. Koetter. "On the relationship between LP decoding and Min-Sum Algorithms decoding". In *ISITA*, Parma, Italy, 2004.

[62] M. Chertkov and M.G. Stepanov. "An efficient pseudo-codeword search algorithm for linear programming decoding of LDPC codes". *IEEE Trans. Inf. Theory*, IT-54(4):1514–1520, Apr. 2008.

[63] M. Chertkov and V. Chernyak. "Loop Calculus Helps to Improve Belief Propagation and Linear Programming Decodings of Low-Density-Parity-Check codes". *invited talk at 44th Allerton Conference*, Sep. 2006.

[64] M. Chertkov and M. Stepanov. "Pseudo-codeword landscape". In *IEEE Intern. Symp. on Inf. Theory*, Nice, France, June 2007.

[65] V. Chernyak, M. Chertkov, M. Stepanov, and B. Vasic. "Error correction on a tree: an instanton approach". *Phys. Rev. Lett.*, 93(198702), 2004.

[66] M. Stepanov, V. Chernyak, M. Chertkov, and B. Vasic. "Diagnosis of weaknesses in modern error correction codes: a physics approach". *Phys. Rev. Lett.*, 95(228701), 2005.

[67] M. Stepanov and M. Chertkov. "The error floor of LDPC codes in the Laplacian channel". In *43rd Allerton Conference on Communication, Control and Computing*, Monticello, IL USA, Sep. 2005.

[68] D.P. Landau and K. Binder. *"A Guide to Monte Carlo Methods in Statistical Physics"*. Cambridge University Press, 2000.

[69] R. Srinivasan. *"Importance Sampling - Applications in Communications and Detection"*. Springer-Verlag, 2002.

[70] P.J. Smith, M. Shafi, and H. Gao. "Quick simulation: A review of importance sampling techniques in communications systems". *IEEE JSAC*, 15(4):597–613, May 1997.

[71] D. Lu and K. Yao. "Improved importance sampling technique for efficient simulation of digital communication systems". *IEEE J. Sel. Area Commun.*, 6(1):67–15, 1988.

[72] J.S. Sadowsky. "A new method for Viterbi decoder simulation using importance sampling". *IEEE Trans. Commun.*, 38:1341–1351, Sept. 1990.

[73] M. Ferrari and S. Bellini. "Importance sampling simulation of concatenated block codes". *IEE Proc. Communications*, 147(5):245–251, Oct. 2000.

[74] B. Xia and W. E. Ryan. "On importance sampling for linear block codes". *Proc. IEEE International Conference on Communications (ICC '03)*, pages 2904–2908, 2003.

[75] B. Xia and W.E. Ryan. "Importance sampling for loop-free decoding trees with application to low-density parity-check codes". In *IEEE Intern. Symp. on Inf. Theory*, Yokohama, Japan, July 2003.

[76] T. Sakai and K. Shibata. "A study on importance sampling for turbo codes". *IEICE Trans. Fundamentals*, E87-A(10):2503–2511, Oct. 2004.

[77] M.C. Jeruchim. "Techniques for estimating the bit error rate in the simulation of digital communication systems". *IEEE J. Select. Areas Commun.*, SAC-2:153–170, Jan. 1984.

[78] K.S. Shanmugam and P. Balaban. "A modified Monte-Carlo simulation technique for the evaluation of error rate in digital communication systems". *IEEE Trans. Commun.*, 28:1916–1924, Nov. 1980.

[79] N.C. Beaulieu. "A composite importance sampling technique for digital communication system simulation". *IEEE Trans. Commun.*, 38:393–396, Apr. 1990.

[80] N.C. Beaulieu. "An investigation of Gaussian tail and Rayleigh tail density functions for importance sampling digital communication system simulation". *IEEE Trans. Comrnun.*, 38:1288–1292, Sep. 1990.

[81] H.J. Schlebusch. "Nonlinear importance sampling techniques for efficient simulation of communication systems". In *Int. Conf. Commun.*, pages 631–635, Apr. 1990.

[82] M. Hsieh. "Adaptive Monte Carlo methods for rare event simulations". In *Winter Simulation Conf.*, Piscataway, NJ USA, 2002.

[83] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi. "A general method for finding low error rates of LDPC codes". *CoRR*, vol abs/cs/0605051, 2006.

[84] L. Dolecek, Z. Zhang, M. Wainwright, V. Anantharam, and B. Nikolic. "Evaluation of the Low Frame Error Rate Performance of LDPC Codes Using Importance Sampling". In *Information Theory Workshop*, Lake Tahoe, California, Sep. 2007.

[85] Hua Xiao and A.H. Banihashemi. "Error rate estimation of finite-length low-density parity-check codes on binary symmetric channels using cycle enumeration". In *IEEE Intern. Symp. on Inf. Theory*, Nice, France, July 2007.

[86] Hua Xiao and A.H. Banihashemi. "Error rate estimation of finite-length LDPC Codes decoded by soft-decision iterative algorithms". In *IEEE Intern. Symp. on Inf. Theory*, Toronto, Canada, July 2008.

[87] N. Sourlas. "Spin glasses, error-correcting codes and finite-temperature decoding". *Europhysics Letters*, 25:159–164, 1994.

[88] N. Sourlas. "Statistical mechanics and capacity-approaching error-correcting codes". *Physica A*, 302:14–21, 2001.

[89] P. Rujan. "Finite temperature error-correcting codes". *Phys. Rev. Lett.*, 70:2968–2971, 1993.

[90] H. Nishimori. "Optimum decoding temperature for error-correcting codes". *Journal of the Physical Society of Japan*, 62:2973–2975, 1993.

[91] A. Montanari. "Turbo codes: The phase transition". *Eur. Phys. J. B*, 18:121–136, 2000.

[92] A. Montanari. "The glassy phase of Gallager codes". *Eur. Phys. J. B*, 23:121–136, 2001.

[93] S. Franz, M. Leone, A. Montanari, and F. Ricci-Tersenghi. "Dynamic phase transition for decoding algorithms". *Physical Review E*, 66:0461201–17, 2002.

[94] I. M. Lifshitz. "Energy spectrum structure and quantum states of disodered condensed systems". *Usp. Fiz. Nauk*, 83(4):617–663, 1964.

[95] L. N. Lipatov. "Divergence of the perturbation-theory series and the quasiclassical theory". *Zh. Eksp. Teor. Fiz*, 72(2):411–427, 1977.

[96] J. Yedidia, W. Freeman, , and Y. Weiss. "Generalized Belief Propagation". *Advances in Neural Information Processing Systems*, 13:689–695, 2000.

[97] J. Yedidia, W. Freeman, and Weiss Y. "Bethe free energy, Kikuchi approximations and Belief Propagation algorithms". In *SCTV01*, 2001.

[98] A.A. Belavin, A.M. Polyakov, A.S. Schwartz, and Y.S. Tyupkin. "Pseudoparticle solutions of the Yang-Mills equations". *Phys. Rev. Lett.*, 59B, 1975.

[99] M.G. Stepanov and M. Chertkov. "Instanton analysis of Low-Density-Parity-Check codes in the error floor regime". In *IEEE Intern. Symp. on Inf. Theory*, Seattle, USA, July 2006.

[100] J.A. Nelder and R. Mead. "A simplex method for function minimization". *Computer Journal*, 7:308–313, 1965.

[101] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. "*Numerical recipes in C: the art of scientific computing*". Cambridge University Press, Oct. 1992.

[102] S.K. Chilappagari, M. Chertkov, M.G. Stepanov, and B. Vasic. “Instanton-Based Techniques for Analysis and Reduction of Error Floors of LDPC Codes”. *IEEE JSAC*, 27(6):855–865, Aug. 2009.

[103] J.S. Stadler and S. Roy. “Adaptive importance sampling”. *IEEE J. Sel. Area Commun.*, 11(3):309–316, 1993.

[104] G.L. Ang. “Kernel method in importance sampling density estimation”. In *ICOSSAR*, pages 1193–1200, San Francisco CA, USA, Aug. 1989.

[105] C.G. Bucher. “Adaptive sampling-An iterative fast Monte Carlo procedure”. *Structural Safety*, 5:119–126, June 1988.

[106] Y. Ibrahim. “An adaptive importance sampling strategy”. *Mech. Comput. in 1990’s*, pages 1288–1292, 1990.

[107] M. Devetsikiotis and J.K. Townsend. “An algorithmic approach to the optimization of importance sampling parameters in digital communication system simulation”. *IEEE Trans. Commun.*, 1991.

[108] R. Holzlohner, A. Mahadevan, C. Menyuk, J. Morris, and J. Zweck. “Evaluation of the very low BER of FEC codes using dual adaptive importance sampling”. *IEEE Comm. Letters*, vol 2, Feb. 2005.

[109] B. A. Berg and T. Neuhaus. “The multicanonical ensemble: a new approach to simulate first-order phase transitions”. *Phys. Rev. Lett.*, vol 68:9–12, 1992.

[110] B. A. Berg. “Algorithmic aspects of multicanonical Monte Carlo simulations”. *Nucl. Phys. Proc. Suppl.*, vol 63:982–984, 1998.

[111] D.J.C. MacKay and R. M. Neal. “Near Shannon limit performance of low density parity check codes”. *Electron. Lett.*, 33:457–458, 1997.

[112] R. Holzlohner and C. R. Menyuk. “Use of multicanonical Monte Carlo simulations to obtain accurate bit error rates in optical communications systems”. *Opt. Lett.*, vol 28:1894–1896, Oct. 2003.

[113] S. Kakakhail, S. Reynal, D. Declercq, and V. Y. Heinrich. “Fast simulation for the performance evaluation of LDPC codes using Fast Flat Histogram Method”. In *IEEE Sarnoff Symposium*, Princeton NJ, USA, Apr. 2008.

[114] S. Kakakhail, S. Reynal, D. Declercq, and V. Y. Heinrich. “Efficient Performance Evaluation of Forward Error Correcting Codes”. In *IEEE ICCS’08*, Guangzhou, China, Nov. 2008.

[115] S. Kakakhail, S. Reynal, D. Declercq, and V. Y. Heinrich. “An efficient pseudo-codeword search algorithm for belief propagation decoding of LDPC codes”. In *IEEE ICUMT’09*, St. Petersburg Russia, Oct. 2009.

[116] B.A. Berg and T. Celik. “New approach to spin-glass simulations”. *Phys. Rev. Lett.*, 69(15):2292–2295, 1992.

[117] B.A. Berg and T. Neuhaus. “Multicanonical algorithms for first-order phase transitions”. *Phys. Rev. Lett.*, B 267:249, 1991.

[118] B.A. Berg, U. Hansmann, and T. Neuhaus. “Simulation of an ensemble with varying magnetic field: a numerical determination of the order-order interface tension in the D=2 Ising model”. *Phys. Rev. Lett.*, B 47(497), 1993.

[119] B.A. Berg and W. Janke. “Multioverlap Simulations of the 3D Edwards-Anderson Ising Spin Glass”. *Phys. Rev. Lett.*, 80(21):4771–4774, May 1998.

[120] U. Hansmann. “Effective way for determination of multicanonical weights”. *Phys. Rev. Lett.*, E 56(5):6200–6203, Nov. 1997.

[121] U. Hansmann and Y. Okamoto. “Monte Carlo simulations in generalized ensemble: Multicanonical algorithm versus simulated tempering”. *Phys. Rev. Lett.*, E 54(5863), 1996.

[122] W. Janke, B.A. Berg, and A. Billoire. “Multi-overlap simulations of free-energy barriers in the 3D Edwards-Anderson Ising spin glass”. *Comput. Phys. Commun.*, 176:121–122, 1999.

[123] F. Wang and D. P. Landau. “Determining the density of states for classical statistical models: a random walk algorithm to produce a flat histogram”. *Phys. Rev.*, E 64(056101), 2001.

[124] F. Wang and D. P. Landau. “Efficient, multiple range random walk algorithm to calculate the density of states”. *Phys. Rev. Lett.*, 86(2050), 2001.

[125] P. Poulain, F. Calvo, R. Antoine, M. Broyer, and P. Dugourd. “Performance of Wang-Landau algorithm for continuous systems”. *Phys. Rev.*, E 73(056704), 2006.

[126] B.J. Schulz, K. Binder, and M. Muller. “Flat histogram method of Wang-Landau and N-fold way”. *Int. J. Mod. Phys.*, C13(477), 2002.

[127] D. Jayasri, V.S.S. Sastry, and K.P.N. Murthy. “Wang-Landau Monte Carlo simulation of isotropic-nematic transition in liquid crystals”. *Phys. Rev.*, E 72(036702), 2005.

[128] C. Zhou and R. N. Bhatt. "Understanding and improving the Wang-Landau algorithm". *Phys. Rev. Lett.*, E 72(025701), 2005.

[129] A. Troster and C. Dellago. "Wang-Landau sampling with self-adaptive range". *Phys. Rev.*, E 71(066705), 2005.

[130] C. Zhou, T.C. Schulthess, S. Torbrgge, and D.P. Landau. "Wang-Landau algorithm for continuous models and joint density of states". *Phys. Rev. Lett.*, 96(120201), 2006.

[131] S. Trebst, D.A. Huse, and M. Troyer. "Optimising the ensemble for equilibrium in broad-histogram Monte Carlo simulations". *Phys. Rev.*, E 70(046701), 2004.

[132] P. Dayal, S. Trebst, S. Wessel, D. Wrtz, M. Troyer, S. Sabhapandit, and S.N. Coppersmith. "Performance limitations of flat-histogram methods". *Phys. Rev. Lett.*, 92(097201), 2004.

[133] H. K. Lee, Y. Okabe, and D.P. Landau. "Convergence and refinement of the Wang-Landau algorithm". *Comput. Phys. Commun.*, 175(36), 2006.

[134] J. Viana Lopes M.D. Costa and J.M.B. Lopes Dos Santos. "Analytical study of tunneling times in flat histogram Monte Carlo". *Europhys. Lett.*, 72:802–808, 2005.

[135] J. Viana Lopes, M. D. Costa, J.M.B. Lopes Dos Santos, and R. Toral. "Optimized multicanonical simulations: a proposal based on classical fluctuation theory". *Phys. Rev.*, E 74(046702), 2006.

[136] R.E. Belardinelli and V.D. Pereyra. "Wang-Landau algorithm: A theoretical analysis of the saturation of error". *Journal of Chemical Physics*, 127(184105), 2007.

[137] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller, and E. Teller. "Equation of state calculations by fast computing machines". *J. Chem. Phys.*, vol 21:1087–1092, 1953.

[138] G. O. Roberts, A. Gelman, and W. R. Gilks. "Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms". *Ann. Appl. Probab.*, vol 7:110–20, 1997.

[139] W. Krauth. "Cluster Monte Carlo algorithms". In *New Optimization Algorithms in Physics*. Wiley-VCh, 2004.

[140] S. Reynal. *"Phase transitions in long-range spin models: the power of generalized ensembles"*. PhD thesis, University of Cergy-Pontoise, June 2005.

[141] Q. Yan and J.J. de Pablo. “Fast calculation of the density of states of a fluid by Monte Carlo simulations”. *Phys. Rev. Lett.*, 90(035701), 2003.

[142] J.S. Wang and R. H. Swendsen. “Transition matrix Monte Carlo method”. *J. Stat. Phys.*, 106(245), 2002.

[143] W. Krauth. “Introduction to Monte Carlo Algorithms”. *Advances in computer simulation, lectures held at Eotvos summer school in Budapest, Hungary*, 1996.

[144] IEEE P802.11n/D1.05. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Enhancements for Higher Throughput (Draft)*, Oct. 2006.

[145] U. Gunawardana, K. C. Nguyen, and R. Liyana-Pathirana. “Fast simulation of turbo codes over AWGN channels”. In *ISCIT ’07*, pages 1176–1181, Sydney, Australia, Oct. 2007.

[146] C. Berrou and M. Jézéquel. “Non binary convolutional codes for turbo coding”. *Elect. Letters*, vol 35(1):39–40, Jan. 1999.

[147] European Telecommunications Standards Institute. “Interaction channel for satellite distribution systems”. V1.3.1 301 790, ETSI EN, Mar. 2003.

[148] A. B. Bortz, M. H. Calos, and J. L. Lebowitz. “A new algorithm for Monte Carlo simulation of Ising spin systems”. *J. Comput. Phys.*, 17(10), 1975.

[149] Y. W. Li, T. Wust, D. P. Landau, and H. Q. Lin. “Numerical integration using Wang-Landau sampling”. *Comput. Phys. Commun.*, 177:524–529, 2007.

[150] M.G. Stepanov, V. Chernyak, M. Chertkov, and B. Vasic. “Diagnosis of weaknesses in modern error correction codes: a physics approach”. *Phys. Rev. Lett.*, 95(228701), 2005.

[151] G.A. Margulis. “Explicit construction of graphs without short cycles and low density codes”. *Combinatorica*, 2(71), 1982.

[152] D.J.C. MacKay. Encyclopedia of sparse graph codes. http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C.

[153] D. Declercq and M. Fossorier. “Improved Impulse method to evaluate the low weight profile of sparse binary linear codes”. In *IEEE Intern. Symp. on Inf. Theory*, Toronto, Canada, July 2008.